Neural Tangent Kernel (NTK) + Preliminary Results

Madison Cooley | Feb. 9, 2022

Outline

- Overview & motivation
- How do NNs act as their width goes to infinity?
- NTK definition

- Simple PDE modeling problem
- Do PINNs exhibit similar behaviour?

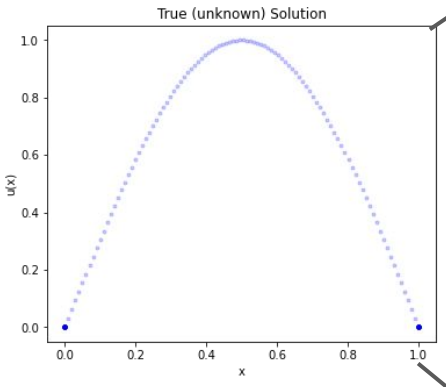
- Strong PINN definition
- Preliminary Results

Overview & Motivation

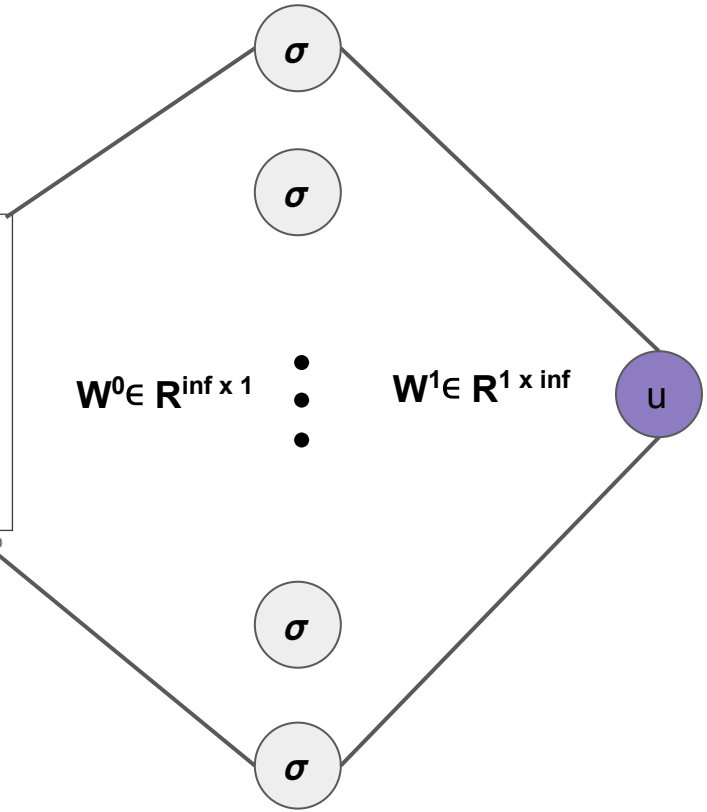
- The Neural Tangent Kernel (NTK) was proposed in 2018 by Jacot et al. for standard neural networks (NNs) [1]
- NTK provides a lens to evaluate the training performance/evolution of NNs
  - How fast is convergence? Which components converge quickest? [3]
- Many NNs and PINNs fail to provide accurate solutions on certain problems
  - Understanding why this occurs can facilitate better mitigation strategies
- Researchers propose methods that empirically fix the problem
  - Often lack theoretical justification
  - Don't address the root cause of the problem [2]
- NTK provides a way to develop methods that directly fix (some of) these issues
  - E.g. **Sifan Wang, Xinling Yu, Paris Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, 2020**

[1] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In Advances in neural information processing systems, 2018.
[2] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536, 2020.
[3] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks.

How do NNs act as their width goes to infinity?



$x \in \mathbf{R}^{N \times 1}$



?

$$u(x, \theta) = \frac{1}{\sqrt{N}} W^{(1)} \cdot \sigma(W^{(0)} x)$$

How do NNs act as their width goes to infinity?

- (Jacot et al. 2018)
  - NNs are equivalent to Gaussian processes in the infinite width limit (at initialization)
  - And training evolution can be described by the NTK

$$u(x, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \boldsymbol{W}^{(1)} \cdot \sigma(\boldsymbol{W}^{(0)} x)$$

Each hidden layer converges to an i.i.d centered Gaussian process [1]

[1] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In Advances in neural information processing systems, 2018.

The Neural Tangent Kernel (NTK)

Defined as the nn parameters are changing during training (for any time t)

$$Ker_t(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{x}', \boldsymbol{\theta}(t)\right)}{\partial \boldsymbol{\theta}} \right\rangle,$$

The Neural Tangent Kernel (NTK)

Defined as the nn parameters are changing during training (for any time t)

loss

input

epoch

$$Ker_t(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{x}', \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} \right\rangle,$$

$\{W^0, b^0, W^1 \ldots\}$

Property 1: Kernel converges to a deterministic kernel

- *At random initialization*
- *As width goes to infinity*

The Neural Tangent Kernel (NTK)

Defined as the nn parameters are changing during training (for any time t)

$$Ker_t(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{x}', \boldsymbol{\theta}(t)\right)}{\partial \boldsymbol{\theta}} \right\rangle,$$

Property 1: Kernel converges to a deterministic kernel

- *At random initialization*
- *As width goes to infinity*

$$\lim_{d_1 \to \infty} Ker_0(\boldsymbol{x}, \boldsymbol{x}') =$$

$$\lim_{d_1 \to \infty} \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{x}', \boldsymbol{\theta}(0)\right)}{\partial \boldsymbol{\theta}} \right\rangle = \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$$

The Neural Tangent Kernel (NTK)

Defined as the nn parameters are changing during training (for any time t)

$$Ker_t(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{x}', \boldsymbol{\theta}(t)\right)}{\partial \boldsymbol{\theta}} \right\rangle,$$

Property 1: Kernel converges to a deterministic kernel

- *At random initialization*
- *As width goes to infinity*

$$\lim_{d_1 \to \infty} Ker_0(\boldsymbol{x}, \boldsymbol{x}') =$$

$$\lim_{d_1 \to \infty} \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{x}', \boldsymbol{\theta}(0)\right)}{\partial \boldsymbol{\theta}} \right\rangle = \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$$

The Neural Tangent Kernel (NTK)

Defined as the nn parameters are changing during training (for any time t)

$$Ker_t(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{x}', \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} \right\rangle,$$

in probability

Property 1: Kernel converges to a deterministic kernel

- *At random initialization*
- *As width goes to infinity*

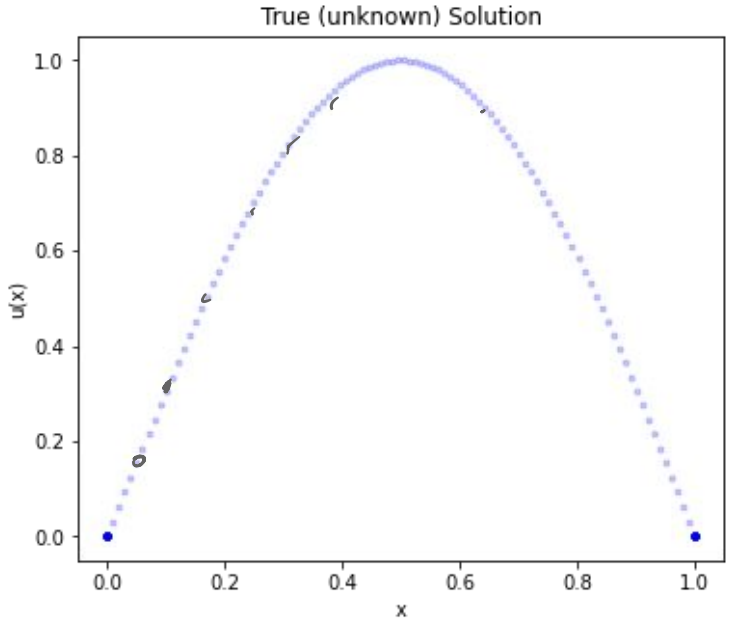
Property 2: Remains constant during training

$$\lim_{d_1 \to \infty} Ker_0(\boldsymbol{x}, \boldsymbol{x}') =$$

$$\lim_{d_1 \to \infty} \left\langle \frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}}, \frac{\partial f(\boldsymbol{x}', \boldsymbol{\theta}(0))}{\partial \boldsymbol{\theta}} \right\rangle = \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$$

Modeling Problem

1D Poisson w. Fabricated solution u(x) = sin(π x)

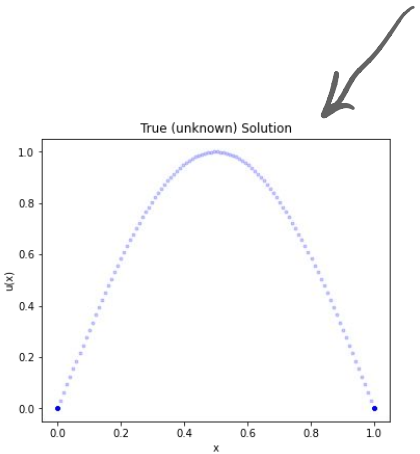


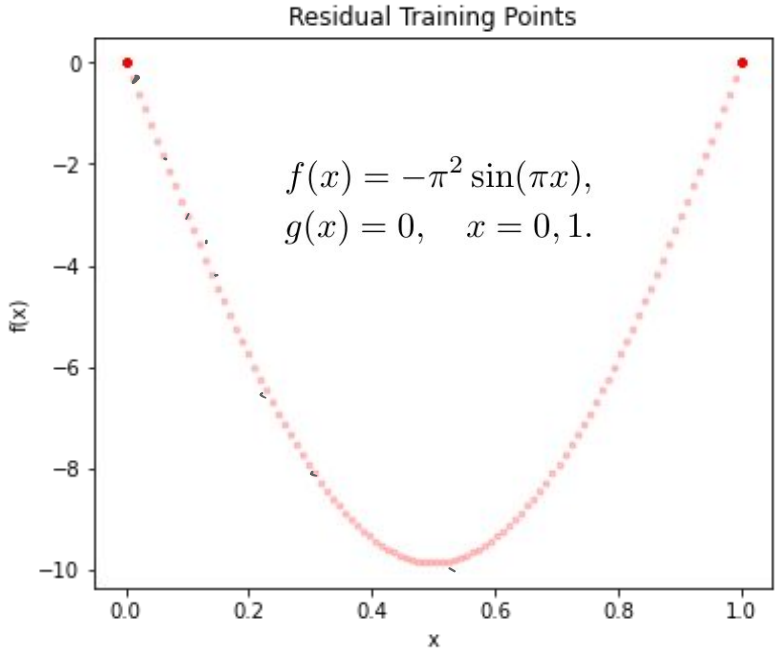
$$u_{xx}(x) = f(x), \quad \forall x \in \Omega$$
$$u(x) = g(x), \quad x \in \partial\Omega.$$

$$f(x) = -\pi^2 \sin(\pi x), \quad x \in [0, 1]$$
$$g(x) = 0, \quad x = 0, 1.$$

Modeling Problem

1D Poisson w. Fabricated solution u(x) = sin(π x)





Modeling Problem

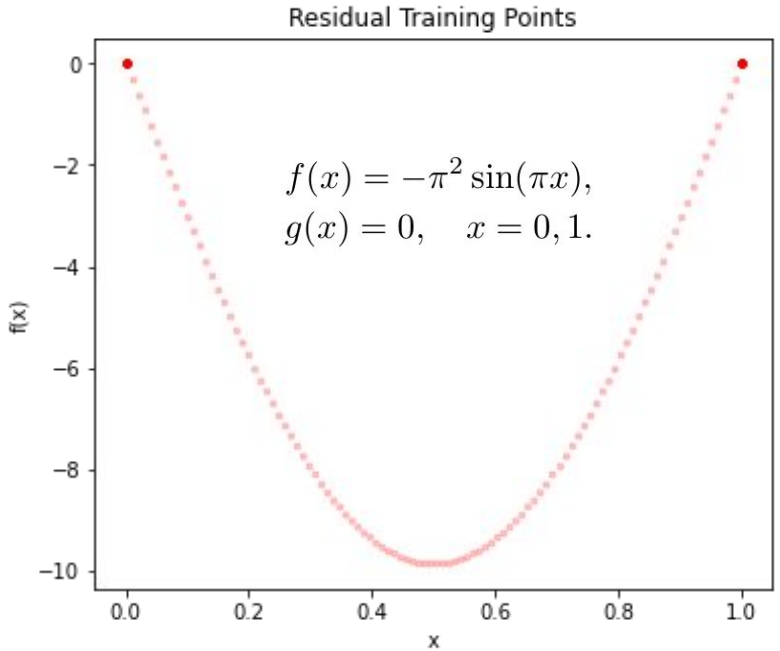
1D Poisson w. Fabricated solution **u(x) = sin(π x)**

PINNs approximate w. **u(x, θ)** $\approx u(\cdot)$

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$

$$= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 +$$

$$\frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2$$



Modeling Problem

1D Poisson w. Fabricated solution u(x) = sin(π x)

?

$$u_{xx}(x) = f(x), \quad \forall x \in \Omega$$
$$u(x) = g(x), \quad x \in \partial\Omega.$$

$$u(x, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \boldsymbol{W}^{(1)} \cdot \sigma(\boldsymbol{W}^{(0)} x)$$

Modeling Problem

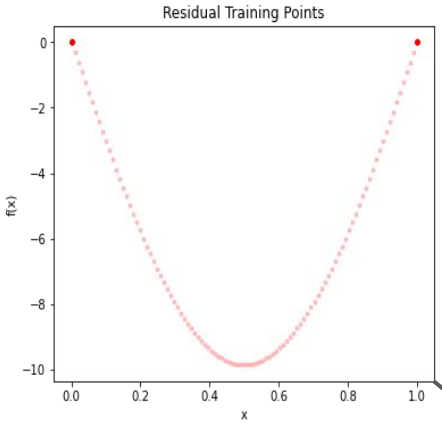
1D Poisson w. Fabricated solution u(x) = sin(π x)

$$u_{xx}(x,\boldsymbol{\theta}) = \frac{1}{\sqrt{N}}\boldsymbol{W}^{(1)} \cdot \left[\ddot{\sigma}(\boldsymbol{W}^{(0)}x\,) \odot \boldsymbol{W}^{(0)} \odot \boldsymbol{W}^{(0)}\right],$$

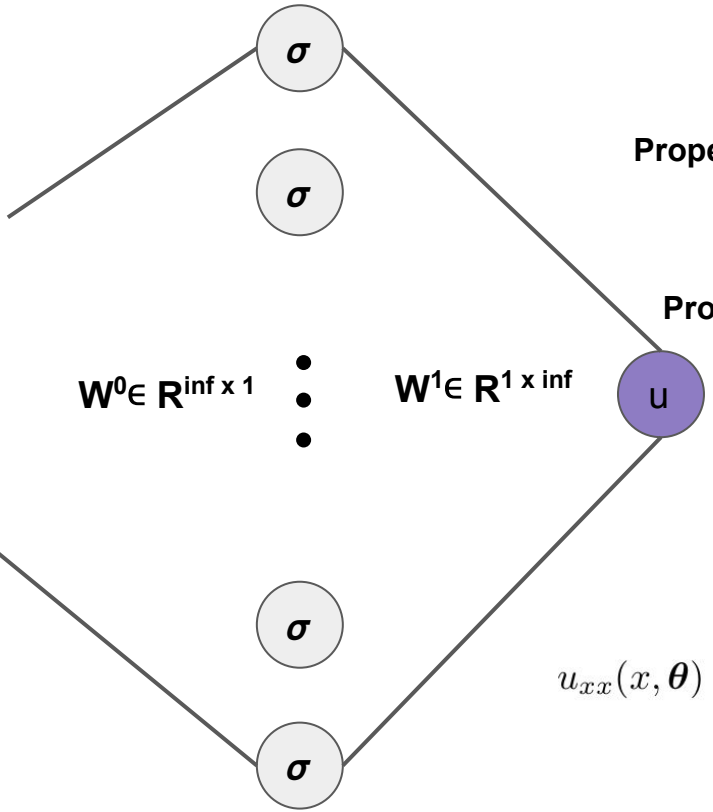
$$u_{xx}(x) = f(x), \quad \forall x \in \Omega$$
$$u(x) = g(x), \quad x \in \partial\Omega.$$

$$u(x,\boldsymbol{\theta}) = \frac{1}{\sqrt{N}}\boldsymbol{W}^{(1)} \cdot \sigma(\boldsymbol{W}^{(0)}x\,)$$

PINNs



$x \in \mathbf{R}^{N \times 1}$



$$u(x, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \boldsymbol{W}^{(1)} \cdot \sigma(\boldsymbol{W}^{(0)} x)$$

Property 1: NTK converges to a deterministic kernel

- *At random initialization*
- *As width goes to infinity*

Property 2: NTK remains constant during training

?

$$u_{xx}(x, \boldsymbol{\theta}) = \frac{1}{\sqrt{N}} \boldsymbol{W}^{(1)} \cdot \left[\ddot{\sigma}(\boldsymbol{W}^{(0)} x) \odot \boldsymbol{W}^{(0)} \odot \boldsymbol{W}^{(0)} \right]$$

Do PINNs exhibit similar behaviour?

- An infinitely wide NN is a Gaussian process [Jacot et al. 2020]
  - Q1: Are (infinitely wide) PINNs?
- How does the PDE residual behave in the infinite width limit
  - Q2: Converges to a deterministic kernel?
  - Q3: Remains constant during training?

WHEN AND WHY PINNS FAIL TO TRAIN: A NEURAL TANGENT KERNEL PERSPECTIVE

A PREPRINT

Sifan Wang
Graduate Group in Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
sifanw@sas.upenn.edu

Xinling Yu
Graduate Group in Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
xlyu@sas.upenn.edu

Paris Perdikaris
Department of Mechanichal Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
pgp@seas.upenn.edu

Do PINNs exhibit similar behaviour

- An infinitely wide NN is a Gaussian process [Jacot et al. 2020]
  - Q1: Are (infinitely wide) PINNs?
- How does the PDE residual behave in the infinite width limit
  - Q2: Converges to a deterministic kernel?
  - Q3: Remains constant during training?

Yes

WHEN AND WHY PINNS FAIL TO TRAIN: A NEURAL TANGENT KERNEL PERSPECTIVE

A PREPRINT

Sifan Wang
Graduate Group in Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
sifanw@sas.upenn.edu

Xinling Yu
Graduate Group in Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
xlyu@sas.upenn.edu

Paris Perdikaris
Department of Mechanichal Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
pgp@seas.upenn.edu

NTK of PINNs

Under gradient flow

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla \mathcal{L}(\boldsymbol{\theta}) \longrightarrow \cdots$$

$$\cdots \longrightarrow \begin{bmatrix} \frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \end{bmatrix} = - \overbrace{\begin{bmatrix} \boldsymbol{K}_{uu}(t) & \boldsymbol{K}_{ur}(t) \\ \boldsymbol{K}_{ru}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix}}^{\text{NTK}} \cdot \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}$$

NTK of PINNs

Under gradient flow

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla \mathcal{L}(\boldsymbol{\theta})$$

, $\theta_{t+1} = \theta_t - \eta_t \cdot \nabla L(\theta_t)$

$$\Rightarrow \frac{\theta_{t+1} - \theta_t}{\eta_t} = -\nabla L(\theta_t)$$

↑ infinitely small

NTK of PINNs

Under gradient flow

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla \mathcal{L}(\boldsymbol{\theta})$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta}) \\ &= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 + \\ &\quad \frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2\end{aligned}$$

NTK of PINNs

Under gradient flow

***We are mainly interested in how u(.) and uxx(.) evolve along the trajectory of gradient descent**

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$

$$= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 +$$

$$\frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2$$

NTK of PINNs

Under gradient flow

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$

$$= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 +$$

$$\frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2$$

***We are mainly interested in how u(.) and uxx(.) evolve along the trajectory of gradient descent**

$$\frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} = \frac{\partial u}{\partial \theta} \cdot \frac{d\theta}{dt} = \frac{du}{d\theta} \cdot -\nabla L(\theta)$$

$$\frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \quad (u_{xx})$$

$$\Rightarrow -\left(u(\cdot) - g(\cdot)\right) \cdot \left\langle \frac{du(\,)}{d\theta}, \frac{du(\,)}{d\theta} \right\rangle$$

$$- \left(u_{xx} - f(\cdot)\right) \cdot \left\langle \frac{du_{xx}(\,)}{d\theta}, \frac{du(\,)}{d\theta} \right\rangle$$

NTK of PINNs

Under gradient flow

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$
$$= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 +$$
$$\frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2$$

***We are mainly interested in how u(.) and uxx(.) evolve along the trajectory of gradient descent**

$$\frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt}$$

$$\frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt}$$

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{K}_{uu}(t) & \boldsymbol{K}_{ur}(t) \\ \boldsymbol{K}_{ru}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix} \cdot \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}$$

NTK of PINNs

Under gradient flow

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_b(\boldsymbol{\theta}) + \mathcal{L}_r(\boldsymbol{\theta})$$
$$= \frac{1}{2}\sum_{i=1}^{N_b} |u(\boldsymbol{x}_b^i, \boldsymbol{\theta}) - g(\boldsymbol{x}_b^i)|^2 +$$
$$\frac{1}{2}\sum_{i=1}^{N_r} |u_{xx}(\boldsymbol{x}_r^i, \boldsymbol{\theta}) - f(\boldsymbol{x}_r^i)|^2$$

***We are mainly interested in how u(.) and uxx(.) evolve along the trajectory of gradient descent**

$$\frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt}$$

$$\frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt}$$

$$(\boldsymbol{K}_{uu})_{ij}(t) = \left\langle \frac{du(\boldsymbol{x}_b^i, \boldsymbol{\theta}(t))}{d\boldsymbol{\theta}}, \frac{du(\boldsymbol{x}_b^j, \boldsymbol{\theta}(t))}{d\boldsymbol{\theta}} \right\rangle$$

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{K}_{uu}(t) & \boldsymbol{K}_{ur}(t) \\ \boldsymbol{K}_{ru}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix} \cdot \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}$$

Q1: Does $u_{xx}(x, \boldsymbol{\theta})$ converge in distribution to another centered Gaussian process?

Shown by Jacot et al.

$$u(x, \boldsymbol{\theta}) \xrightarrow{\mathcal{D}} \mathcal{GP}(0, \Sigma^{(1)}(x, x'))$$

$$u_{xx}(x, \boldsymbol{\theta}) \xrightarrow{\mathcal{D}} \mathcal{GP}(0, \Sigma^{(1)}_{xx}(x, x'))$$

Q1: Does $u_{xx}(x, \boldsymbol{\theta})$ converge in distribution to another centered Gaussian process?

Shown by Jacot et al.

*Shown in this work

$$u(x, \boldsymbol{\theta}) \xrightarrow{\mathcal{D}} \mathcal{GP}(0, \Sigma^{(1)}(x, x'))$$

$$u_{xx}(x, \boldsymbol{\theta}) \xrightarrow{\mathcal{D}} \mathcal{GP}(0, \Sigma^{(1)}_{xx}(x, x'))$$

- Implies model problem induces a joint GP between u (the function values) and the PDE residual at initialization
- PINNs-GP correspondence for linear PDEs

Q2: Does the NTK of PINNs converge to a deterministic kernel?

Theorem 4.3. *For a physics-informed network with one hidden layer at initialization, and in the limit as the layer's width $N \to \infty$, the NTK $\boldsymbol{K}(t)$ of the PINNs model defined in equation 3.9 converges in probability to a deterministic limiting kernel, i.e,*

$$\boldsymbol{K}(0) = \begin{bmatrix} \boldsymbol{K}_{uu}(0) & \boldsymbol{K}_{ur}(0) \\ \boldsymbol{K}_{ru}(0) & \boldsymbol{K}_{rr}(0) \end{bmatrix} \to \begin{bmatrix} \Theta_{uu}^{(1)} & \Theta_{ur}^{(1)} \\ \Theta_{ru}^{(1)} & \Theta_{rr}^{(1)} \end{bmatrix} := \boldsymbol{K}^*, \tag{4.7}$$

Q2: Does the NTK of PINNs converge to a deterministic kernel?

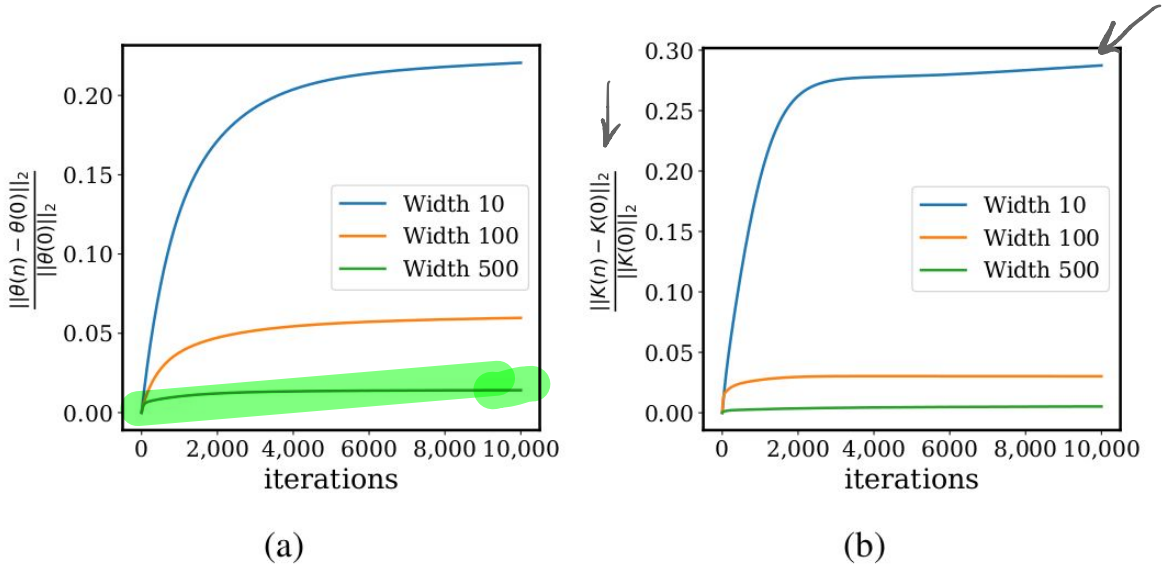
Theorem 4.3. *For a physics-informed network with one hidden layer at initialization, and in the limit as the layer's width $N \to \infty$, the NTK $\boldsymbol{K}(t)$ of the PINNs model defined in equation 3.9 converges in probability to a deterministic limiting kernel, i.e,*

$$\boldsymbol{K}(0) = \begin{bmatrix} \boldsymbol{K}_{uu}(0) & \boldsymbol{K}_{ur}(0) \\ \boldsymbol{K}_{ru}(0) & \boldsymbol{K}_{rr}(0) \end{bmatrix} \to \begin{bmatrix} \Theta_{uu}^{(1)} & \Theta_{ur}^{(1)} \\ \Theta_{ru}^{(1)} & \Theta_{rr}^{(1)} \end{bmatrix} := \boldsymbol{K}^*, \tag{4.7}$$

Q3: Does the NTK of PINNs remain constant during training?

$$\lim_{N \to \infty} \sup_{t \in [0,T]} \|\boldsymbol{K}(t) - \boldsymbol{K}(0)\|_2 = 0,$$

Some numerical results



Combining the results of Q2 and Q3

Q2: Does the NTK of PINNs converge to a deterministic kernel?
Q3: Does the NTK of PINNs remain constant during training?

$$\boldsymbol{K}(t) \approx \boldsymbol{K}(0) \approx \boldsymbol{K}^*, \quad \forall t,$$

Combining the results of Q2 and Q3

Q2: Does the NTK of PINNs converge to a deterministic kernel?
Q3: Does the NTK of PINNs remain constant during training?

$$\boldsymbol{K}(t) \approx \boldsymbol{K}(0) \approx \boldsymbol{K}^*, \quad \forall t,$$

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{K}_{uu}(t) & \boldsymbol{K}_{ur}(t) \\ \boldsymbol{K}_{ru}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix} \cdot \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}$$

Combining the results of Q2 and Q3

Q2: Does the NTK of PINNs converge to a deterministic kernel?
Q3: Does the NTK of PINNs remain constant during training?

$$\boldsymbol{K}(t) \approx \boldsymbol{K}(0) \approx \boldsymbol{K}^*, \quad \forall t,$$

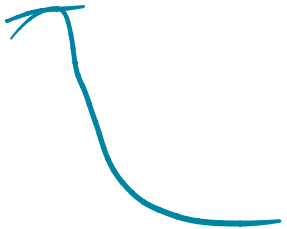
$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b, \boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t))}{dt} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{K}_{uu}(t) & \boldsymbol{K}_{ur}(t) \\ \boldsymbol{K}_{ru}(t) & \boldsymbol{K}_{rr}(t) \end{bmatrix} \cdot \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathcal{L}u(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}$$

$$\approx -\boldsymbol{K}^* \begin{bmatrix} u(\boldsymbol{x}_b, \boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r, \boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r). \end{bmatrix}$$

Combining the results of Q2 and Q3

Q2: Does the NTK of PINNs converge to a deterministic kernel?
Q3: Does the NTK of PINNs remain constant during training?

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx -\boldsymbol{K}^* \begin{bmatrix} u(\boldsymbol{x}_b,\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r). \end{bmatrix}$$



Combining the results of Q2 and Q3

Q2: Does the NTK of PINNs converge to a deterministic kernel?
Q3: Does the NTK of PINNs remain constant during training?

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx -\boldsymbol{K}^* \begin{bmatrix} u(\boldsymbol{x}_b,\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r). \end{bmatrix}$$

$$\begin{bmatrix} u(\boldsymbol{x}_{test},\boldsymbol{\theta}(\infty)) \\ u_{xx}(\boldsymbol{x}_{test},\boldsymbol{\theta}(\infty)) \end{bmatrix} \approx \boldsymbol{K}^*_{test}(\boldsymbol{K}^*)^{-1} \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

Spectral Bias in PINNs

Spectral Bias in PINNs

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx -\dot{\boldsymbol{K}}(0) \begin{bmatrix} u(\boldsymbol{x}_b,\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix} \longrightarrow \quad \cdots$$

$$\cdots \longrightarrow \quad \boldsymbol{Q}\left(\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix} \right) \approx -e^{-\boldsymbol{\Lambda} t} \boldsymbol{Q} \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

Spectral Bias in PINNs

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx -\boldsymbol{K}(0) \begin{bmatrix} u(\boldsymbol{x}_b,\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx \left(I - e^{-\boldsymbol{K}(0)t}\right) \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

$$\cdots \longrightarrow \boldsymbol{Q}\left(\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}\right) \approx -e^{-\boldsymbol{\Lambda}t}\boldsymbol{Q} \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

Spectral Bias in PINNs

$$\frac{du}{dt} = \qquad u = e^{-Kt}$$

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx -\boldsymbol{K}(0) \begin{bmatrix} u(\boldsymbol{x}_b,\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ u_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} \approx \left(I - e^{-\boldsymbol{K}(0)t}\right) \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

$$\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix} \approx \left(I - e^{-\boldsymbol{K}(0)t}\right) \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

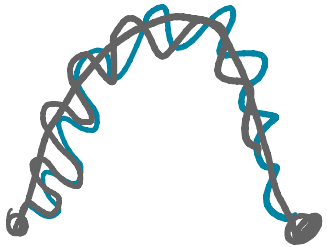
$$\approx -\boldsymbol{Q}^T e^{-\boldsymbol{\Lambda}t} \boldsymbol{Q} \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix},$$

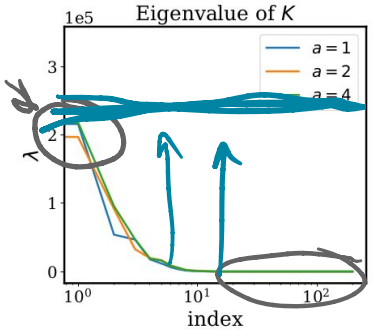
$$\boldsymbol{Q}\left(\begin{bmatrix} \frac{du(\boldsymbol{x}_b,\boldsymbol{\theta}(t))}{dt} \\ \frac{du_{xx}(\boldsymbol{x}_r,\boldsymbol{\theta}(t))}{dt} \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}\right) \approx -e^{-\boldsymbol{\Lambda}t} \boldsymbol{Q} \cdot \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}$$

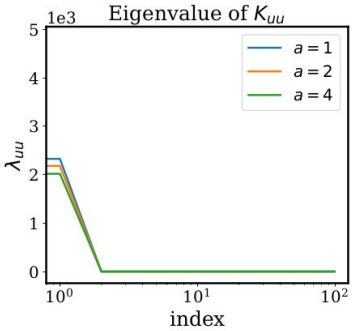
u u_{xx}

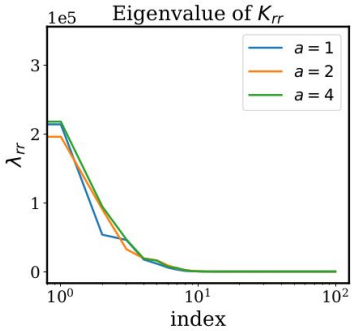
Spectral Bias in PINNs

$$\begin{bmatrix} \boldsymbol{q}_1^T \\ \boldsymbol{q}_2^T \\ \vdots \\ \boldsymbol{q}_N^T \end{bmatrix} \left(f(\boldsymbol{X}_{\text{train}}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_{\text{train}} \right) = \begin{bmatrix} e^{-\lambda_1 t} & & & \\ & e^{-\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{-\lambda_N t} \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_1^T \\ \boldsymbol{q}_2^T \\ \vdots \\ \boldsymbol{q}_N^T \end{bmatrix} \boldsymbol{Y}_{\text{train}}$$









How is the NTK of PINNs practically useful?

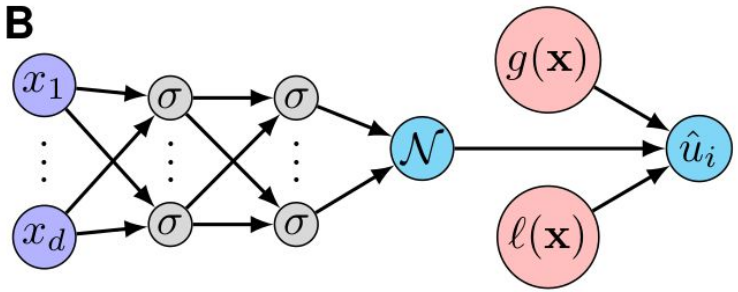
- Define the avg. convergence rate as the mean of all eigenvalues
- Reweight loss terms given these computed averages
- Balances out loss terms such that one doesn't dominate
- Not a fool-proof solution though

$$L = \lambda_1 \mathcal{L}_r + \lambda_2 \mathcal{L}_B$$

Strong PINNs

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_b$$

- Enforce boundary conditions exactly by modifying architecture
- As opposed to through an additional term in the loss function



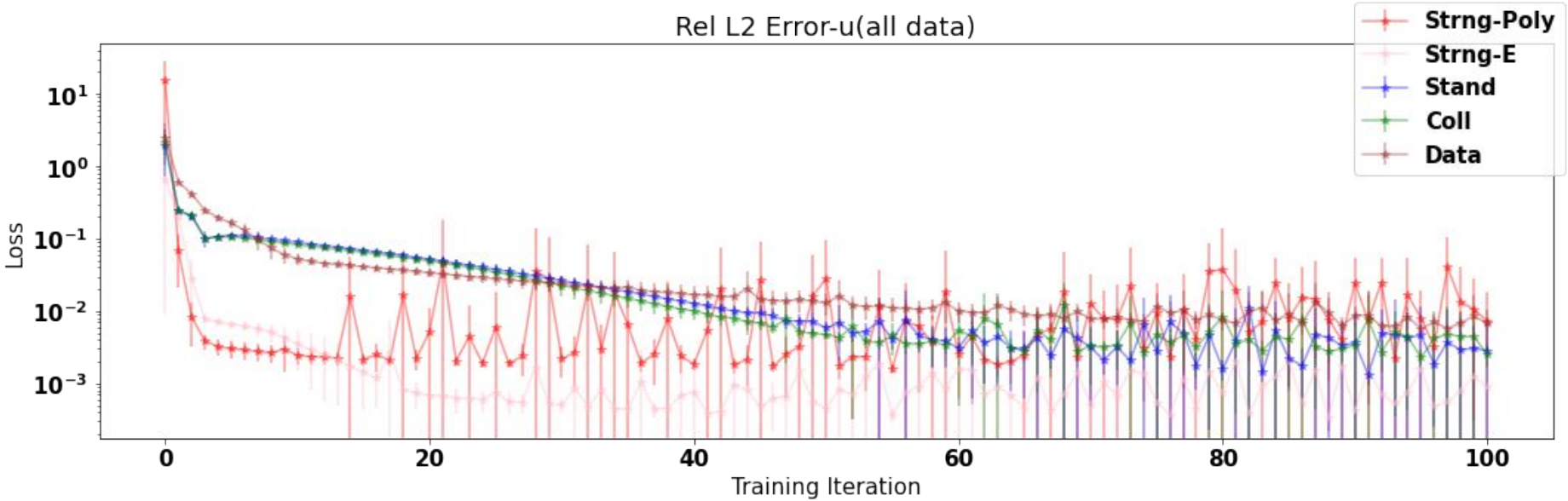
$$\hat{u}_i(\mathbf{x};\boldsymbol{\theta}_u) = g(\mathbf{x}) + \ell(\mathbf{x})\mathcal{N}(\mathbf{x};\boldsymbol{\theta}_u),$$

$$\begin{cases} \ell(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma_D, \\ \ell(\mathbf{x}) > 0, & \mathbf{x} \in \Omega - \Gamma_D \end{cases}$$

choose $\ell(x)$ as $(x-a)(b-x)$ or $(1-e^{a-x})(1-e^{x-b})$

Figure from [Lu et al. 2021]

Preliminary Strong PINN Results



Preliminary Strong PINN Results