Wang & Lai, *J. Comput. Phys.* (2024)
Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Stanford
Doerr
School of
Sustainability

Do AI models truly learn multiscale physics, and how?

Ching-Yao Lai

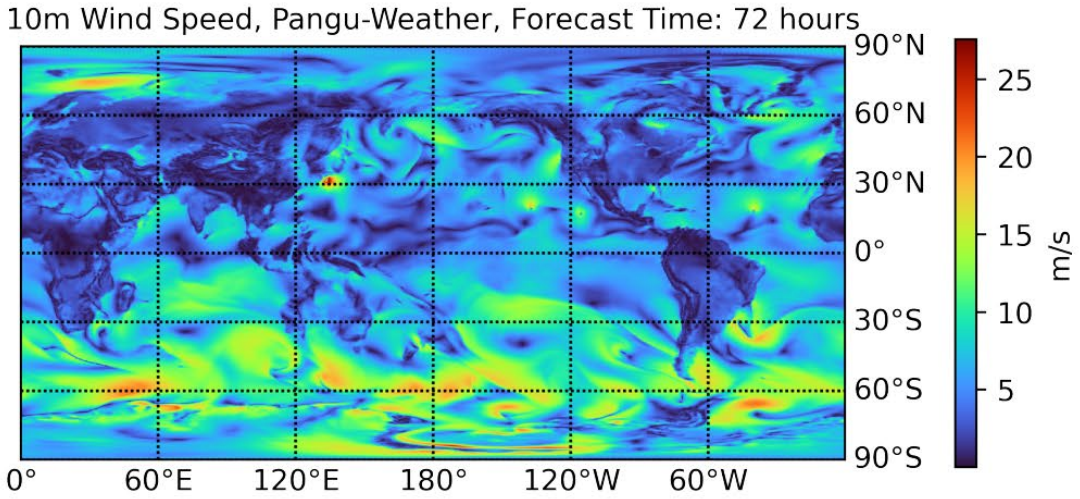
Stanford University

Image generated by Generative AI

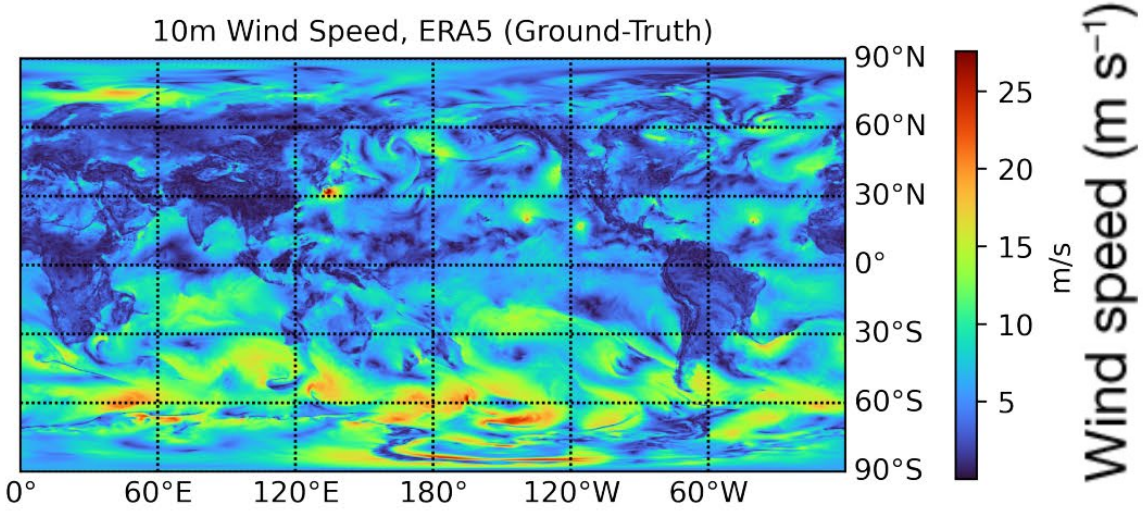
Some progress and challenges

AI weather forecast

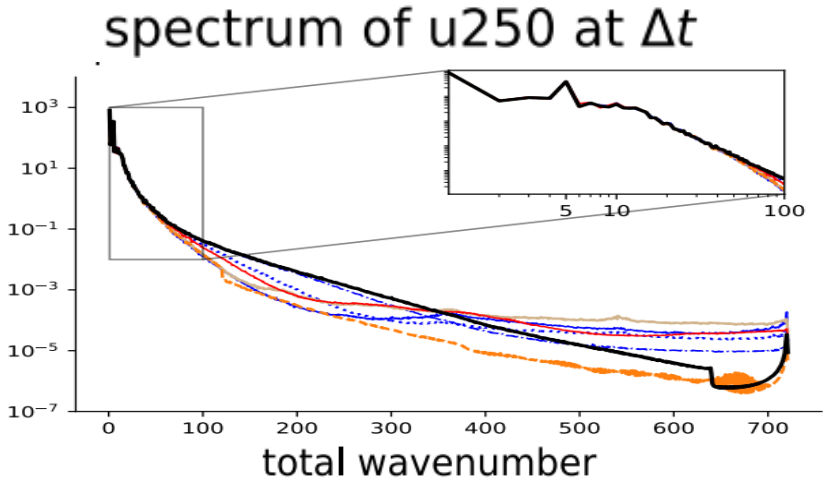
Bi et al (2023) Nature

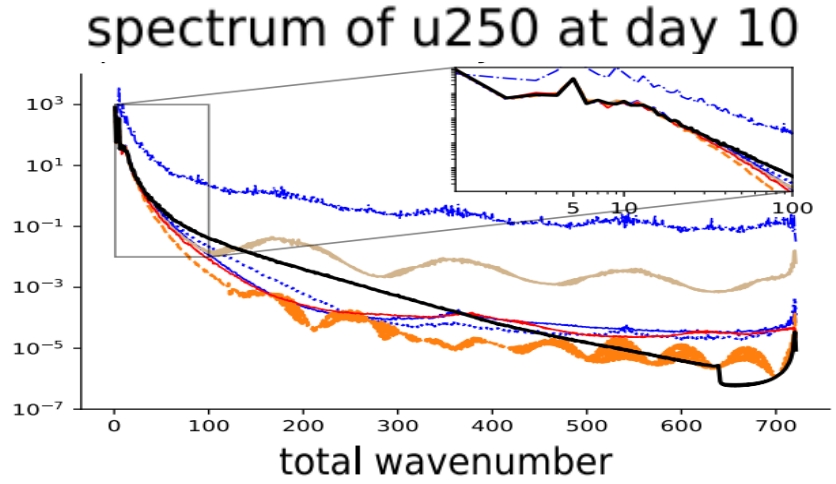


Training data (Re-analysis)



- Spectral bias of NN:





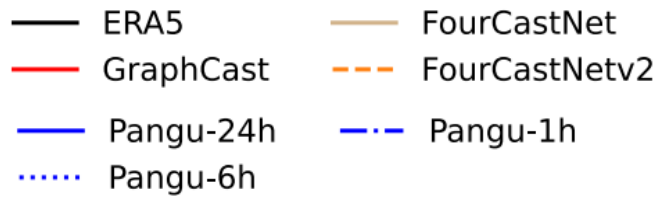
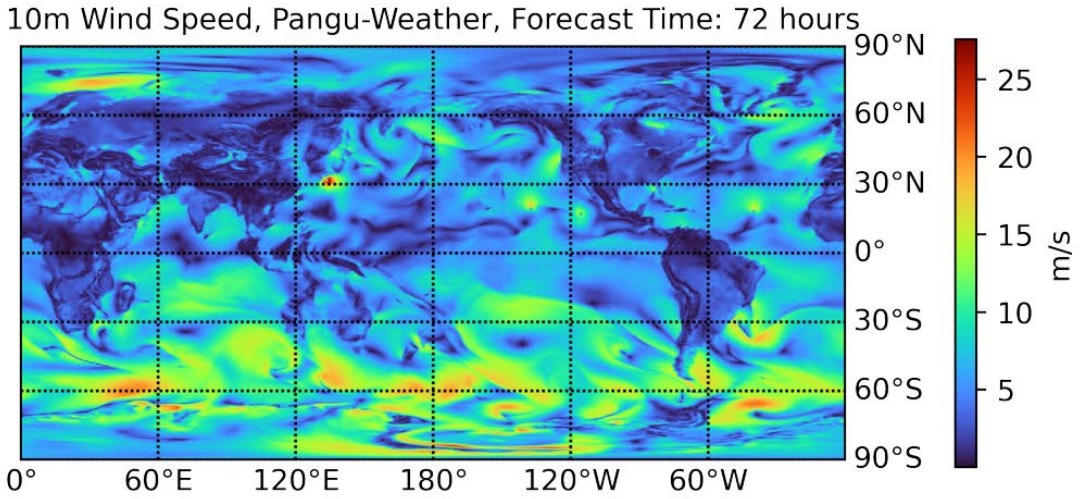


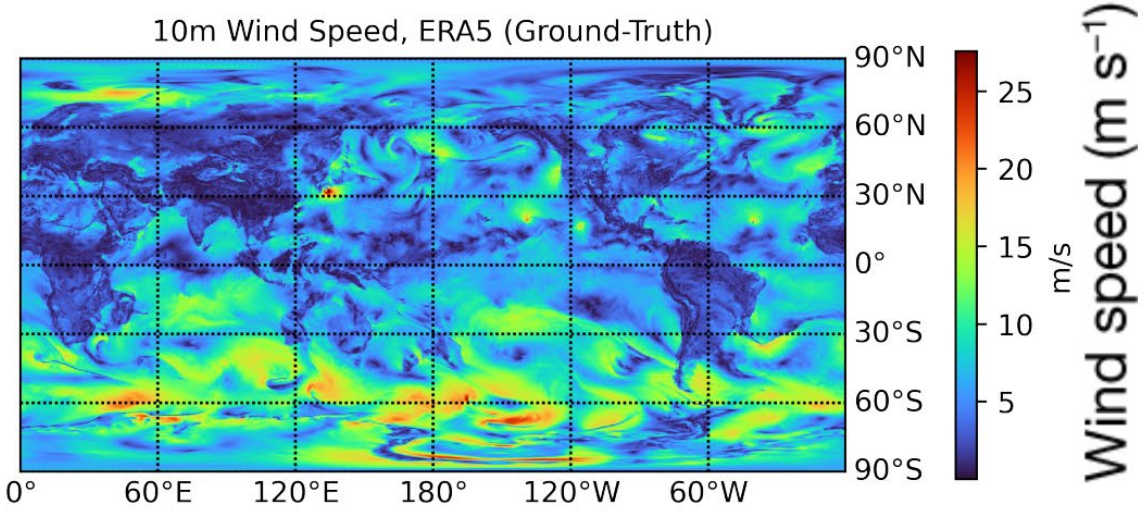
Image source: Qiang Sun & Pedram Hassanzadeh

Spectral bias is bad for multiscale problems

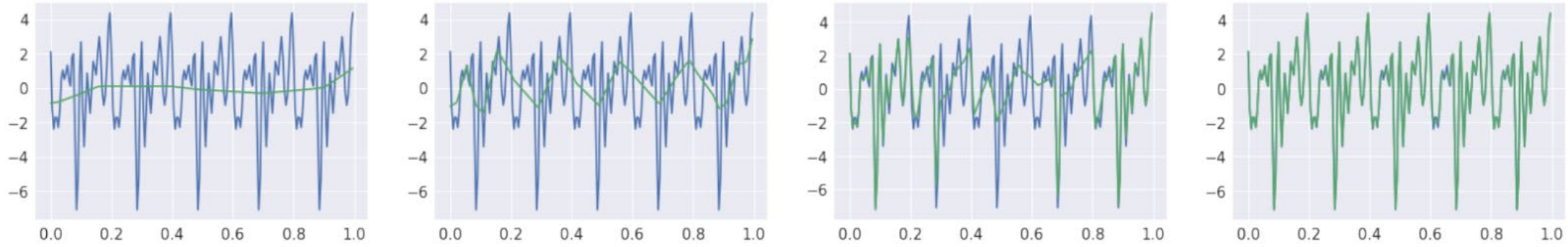
AI weather forecast Bi et al (2023) Nature



Training data (Re-analysis)



- Spectral bias of NN: Rahaman et al (2019), Frequency Principle: Xu et al (2020)



(a) Iteration 100 (b) Iteration 1000 (c) Iteration 10000 (d) Iteration 80000

Spectral bias: A Fourier analysis of NN

Xu et al. (2022)

For a single-hidden-layer NN with input $x \in \mathbb{R}$ and output $u \in \mathbb{R}$

$$u(x) = \sum_{j=1}^{m} w_j^{(1)} \sigma(w_j^{(0)} x + b_j), \quad \text{where} \quad \theta_j = \{w_j^{(0)}, w_j^{(1)}, b_j\}$$

The mean square loss measures the distance between the NN output $u(x)$ and ground truth $u_g(x)$

$J \equiv \int_{-\infty}^{\infty} \frac{1}{2}(u(x) - u_g(x))^2 dx = \int_{-\infty}^{\infty} \frac{1}{2}|\hat{u}(k) - \hat{u}_g(k)|^2 dk$, where $\hat{u}(k)$ denotes the Fourier transform of $u(x)$

The loss at frequency k is $J(k) = \frac{1}{2}|\hat{u} - \hat{u}_g|^2$

For tanh activation function, the gradient of loss at $J(k)$ with respect to NN parameters θ_j is

$$\left|\frac{\partial J(k)}{\partial \theta_j}\right| \approx A(k) e^{-\left|\frac{\pi k}{2w_j^{(0)}}\right|}$$ where $A(k) \in [0, +\infty)$ is the amplitude of $\hat{u} - \hat{u}_g$

This gradient decay rapidly with frequency k!! This is a simple explanation of the spectral bias.

Capturing high-frequency functions

Yongji Wang
Postdoc

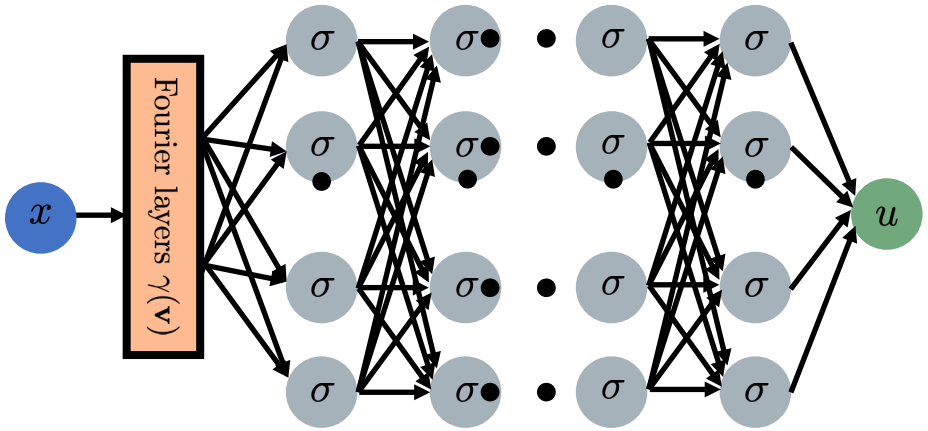


- For multiscale problems, it is important to rescale the NN weights so that it actives the learning of high-frequency (large k) functions

$$\left|\frac{\partial J(k)}{\partial \theta_j}\right| \approx A(k) e^{-\left|\frac{\pi k}{2w_j^{(0)}}\right|}$$

We make $w_j^{(0)} \propto k$

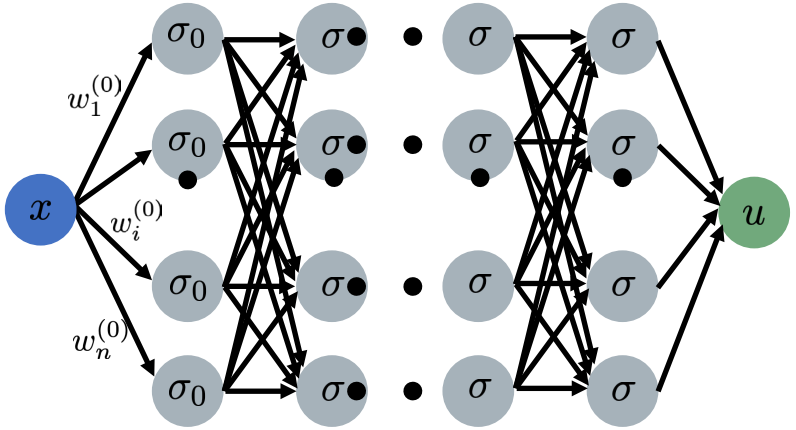
Fourier Feature nets Tancik *et. al.* (2020)



$$\Phi(\mathbf{x}) = \begin{bmatrix} \cos(\mathbf{Bx}) \\ \sin(\mathbf{Bx}) \end{bmatrix} \quad \mathbf{B} \sim \mathcal{N}(0, \kappa^2)$$

Gaussian distribution with standard deviation κ

Our implementation: Wang & Lai, J. Comput. Phys. (2024)

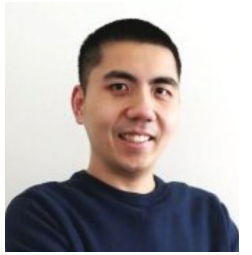


First layer: $\sigma_0(x) = \sin\left(\kappa w_i^{(0)} x + b_i^{(0)}\right)$

κ: scaling factor

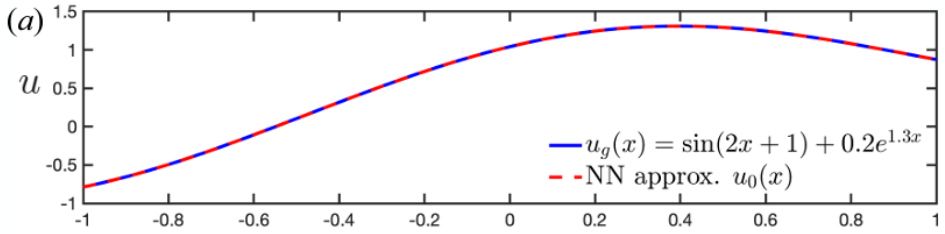
Errors of neural networks

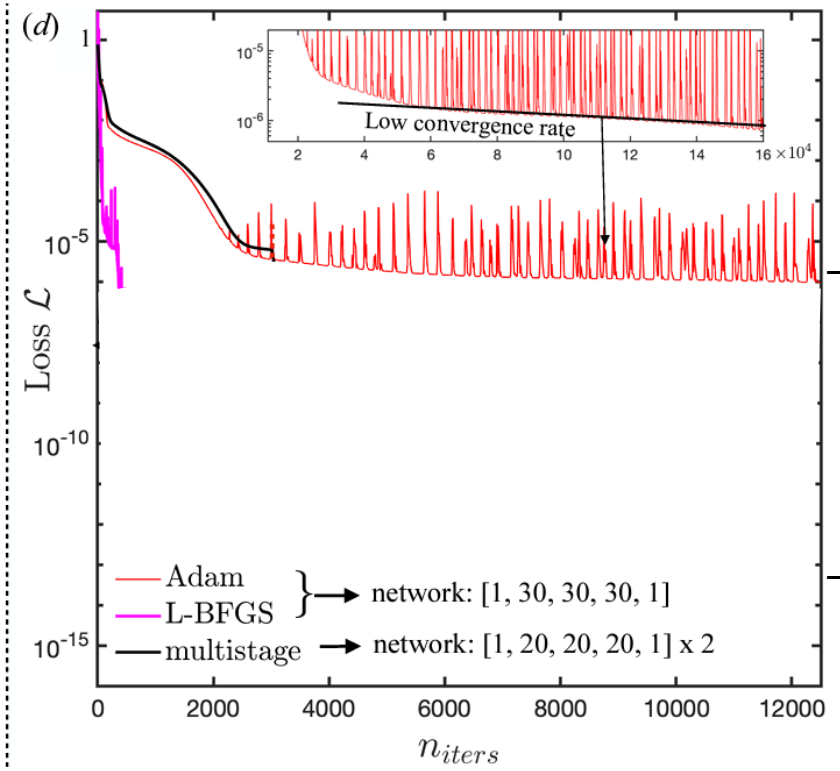
Yongji Wang
Postdoc



- NN's function approximation errors

  NN as universal function approximators: Hornik et al (1990)





→ Regular NN

→ Multistage NN

Main idea

Wang & Lai, J. Comput. Phys. (2024)

- When NN training error is small, the learning becomes really slow.
  Inspired by the principals of perturbation theory, MSNN includes a superposition of multi-stage NNs, with each stage using a new NN to fit the residue rescaled to O(1) from the previous NN.

$$u_c^{(n)}(x) = u_0(x) + \epsilon_1 u_1(x) + \epsilon_2 u_2(x) + \ldots + \epsilon_n u_n(x) = \sum_{j=0}^{n} \epsilon_j u_j(x)$$

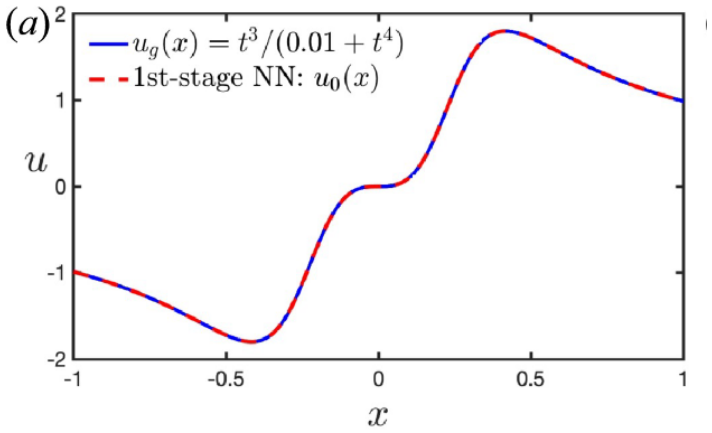
where $1 \gg \epsilon_1 \gg \epsilon_2 \gg \ldots \gg \epsilon_n$

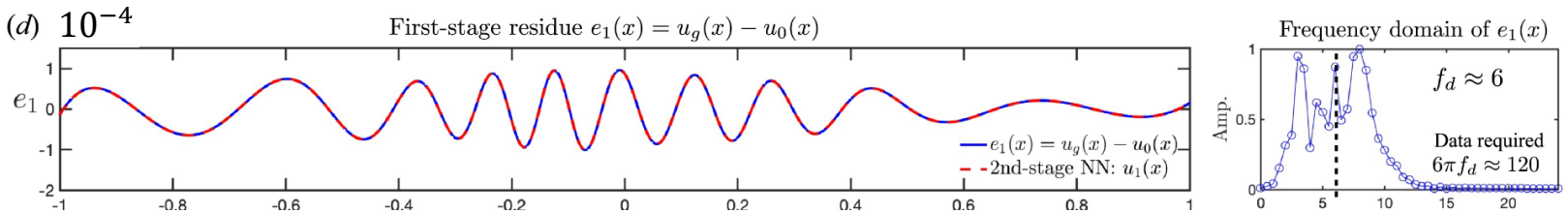
$u_0(x)$, $u_1(x)$ and $u_n(x)$ are all of order $\mathcal{O}(1)$ NNs of different stages

Related work with similar flavors: Sequential residual learning: Krishnanunni & Bui-Thanh arXiv:2211.06860, Multi-level neural networks: Aldirany et al. (2024), Stacked networks: Howard et al. arXiv:2311.06483, Precision Machine Learning: Michaud et al. arXiv:2210.13447…etc

Multistage-NN (MSNN)

Wang & Lai, J. Comput. Phys. (2024)





Can multistage-NN be used to tackle the spectral bias in multiscale problems?

Jakin Ng
Student



Example:

How precisely can a NN approximate a 2D fluid flow?

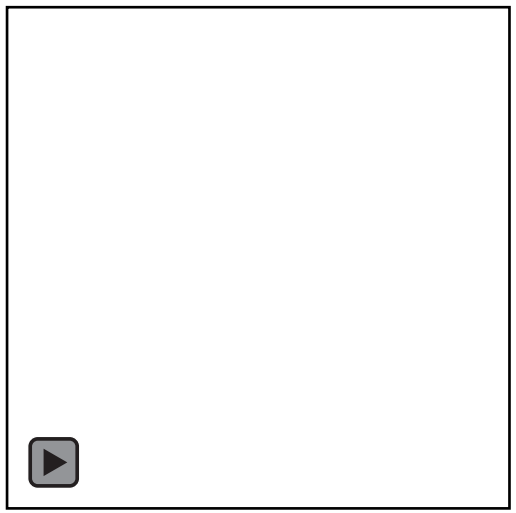
2D incompressible Navier-Stokes eqn:

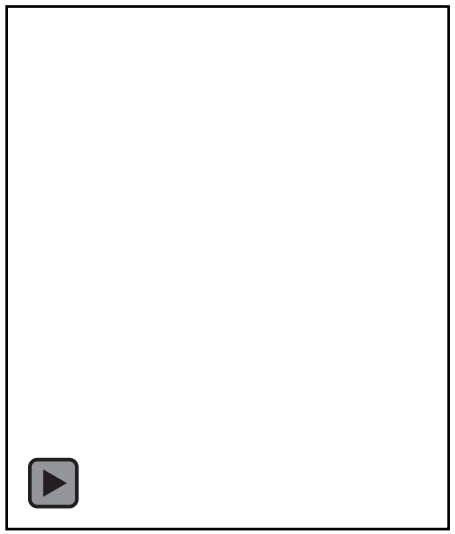
ω: vorticity
ψ: stream function

$$\frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} = \frac{1}{\mathrm{Re}}\nabla^2\omega$$

$$\nabla^2\psi = -\omega$$

Training data:
Numerical solution at $\mathrm{Re} = 2000$

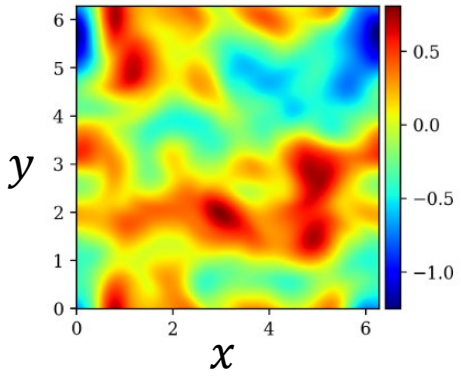




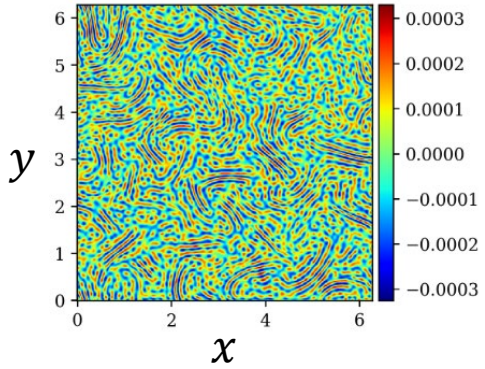
Energy spectrum: $E(k,t) = \sum_{k-\Delta k \le |\boldsymbol{k}| \le k+\Delta k} \frac{1}{2}k^2|\tilde{\psi}(\boldsymbol{k},t)|^2$

E.g. 2D incompressible Navier-Stokes

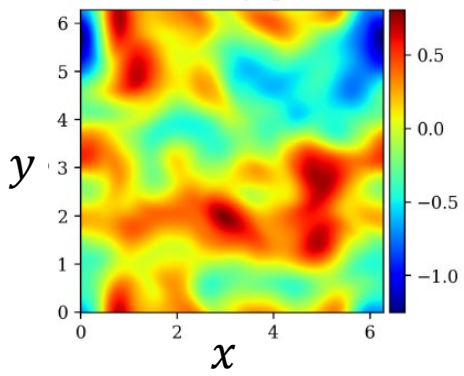
Target $\psi(x, y)$



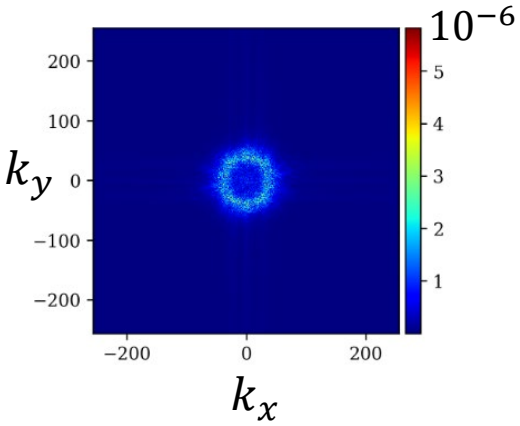
Error



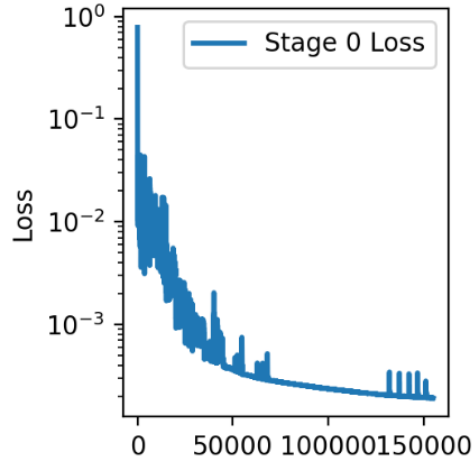
NN $\psi(x, y)$



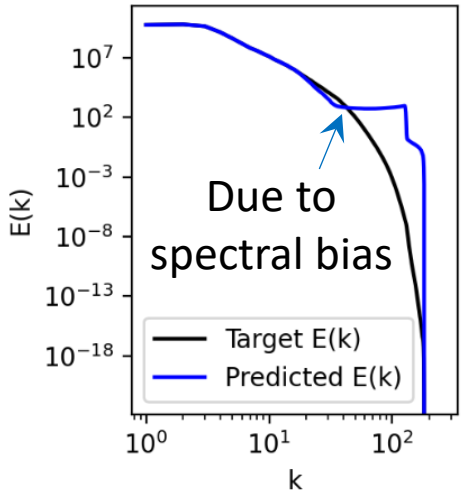
Error in Fourier domain



Loss convergence



Energy spectrum

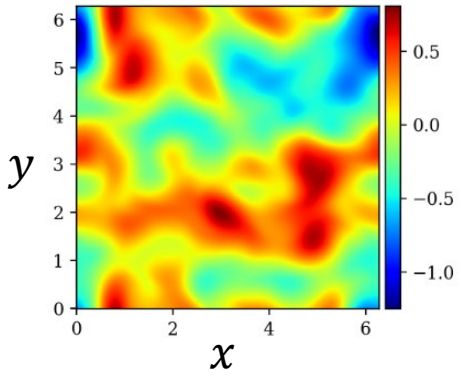


E.g. 2D incompressible Navier-Stokes

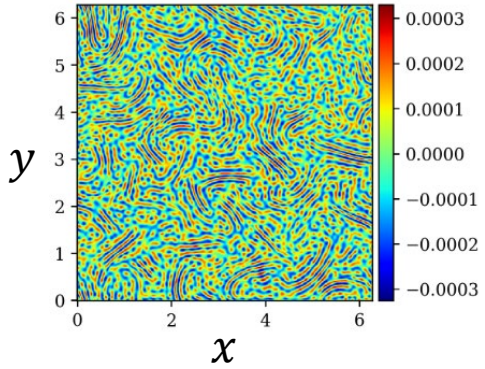
Stage 0

Stage 1

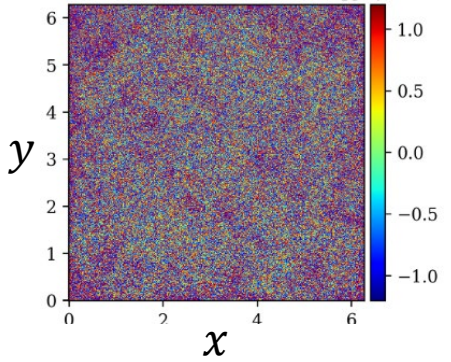
Target $\psi(x, y)$



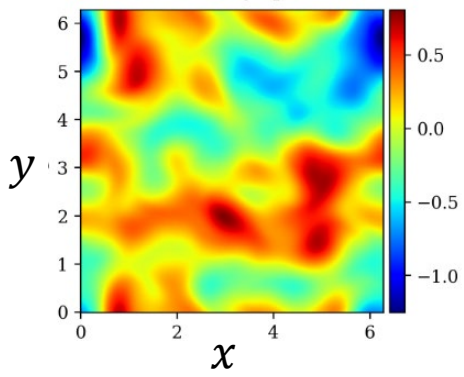
Error



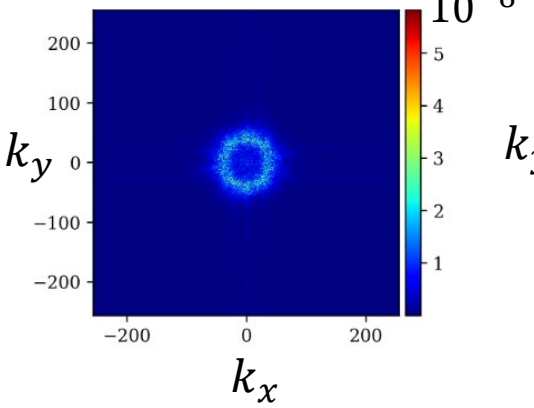
Error



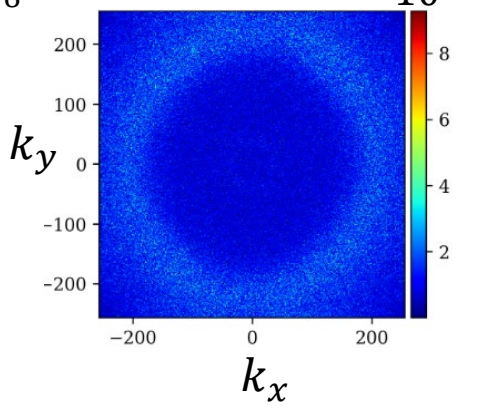
NN $\psi(x, y)$



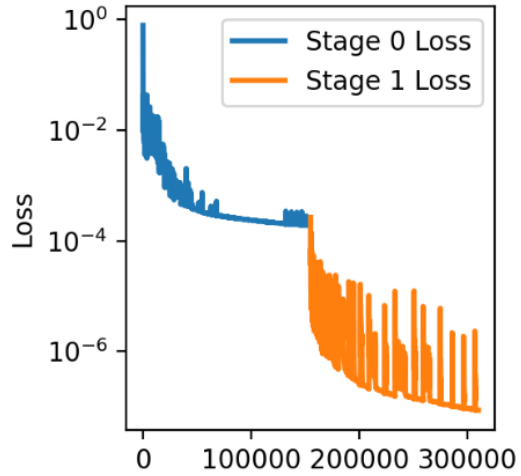
Error in Fourier domain



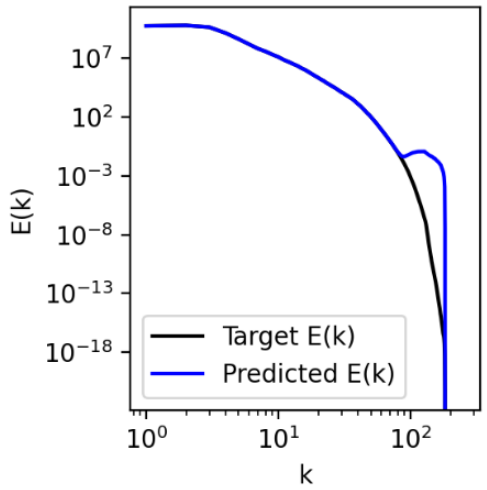
Error in Fourier domain



Loss convergence

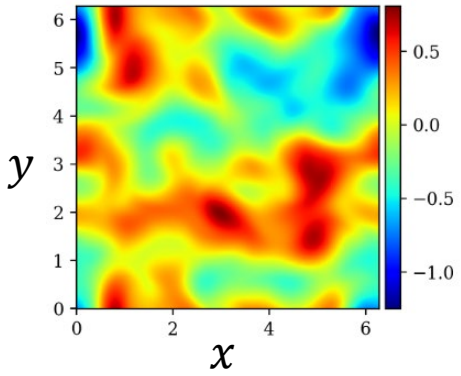


Energy spectrum



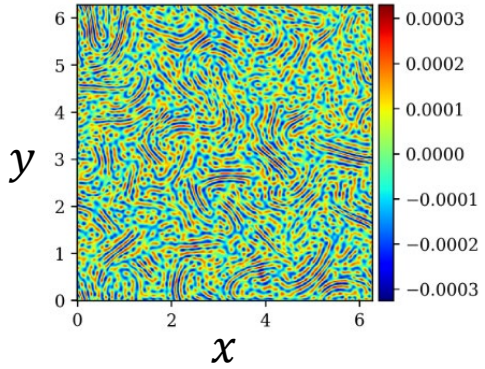
E.g. 2D incompressible Navier-Stokes

Target $\psi(x, y)$



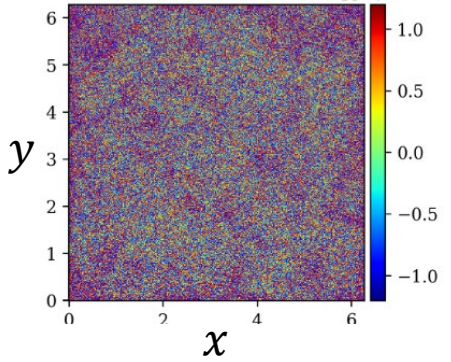
Stage 0

Error



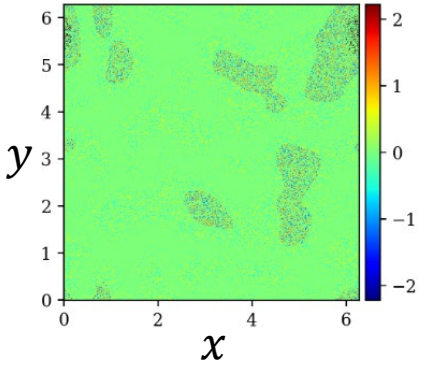
Stage 1

Error 10^{-7}

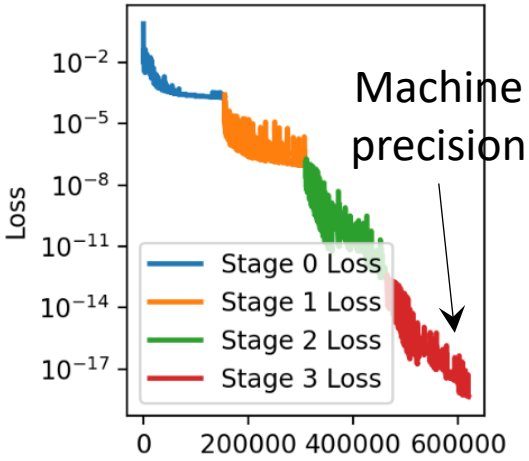


.... Stage 3

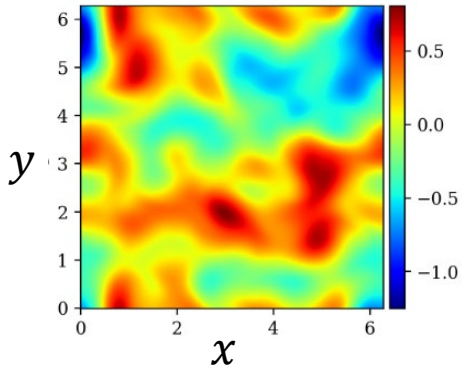
Error 10^{-16}



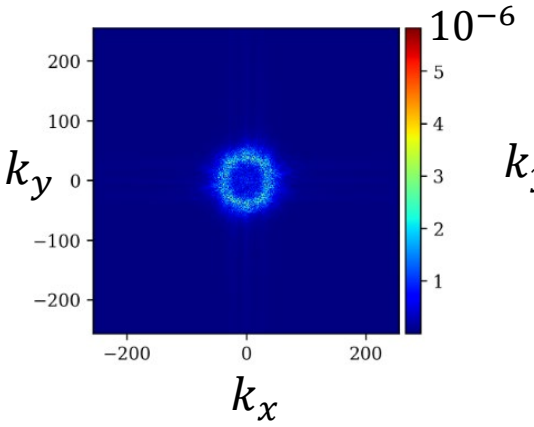
Loss convergence



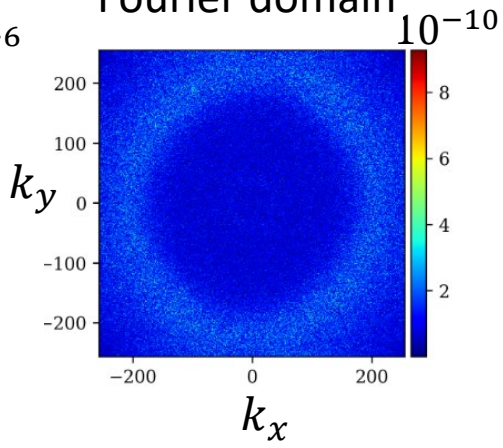
NN $\psi(x, y)$



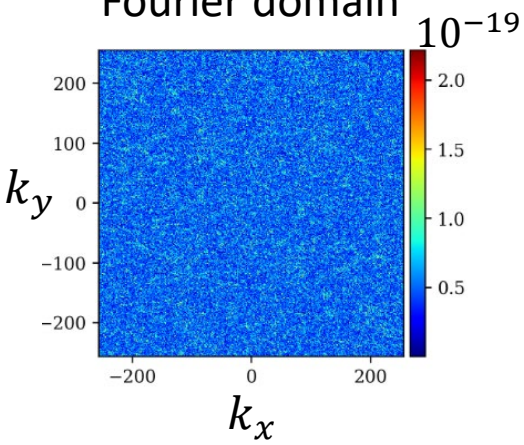
Error in Fourier domain



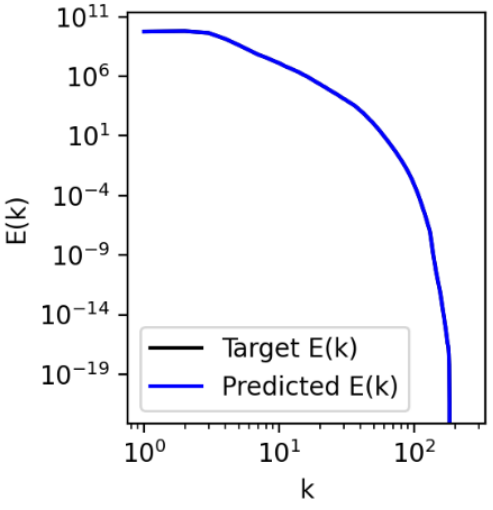
Error in Fourier domain



Error in Fourier domain



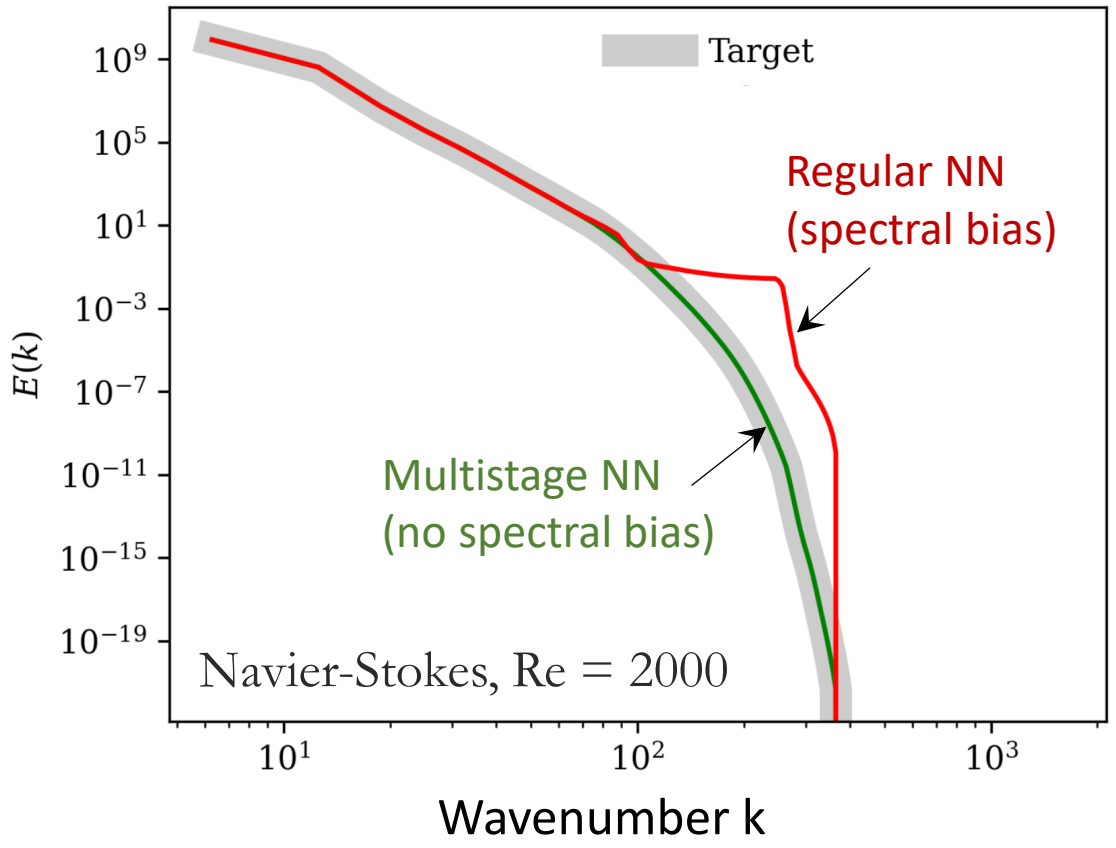
Energy spectrum



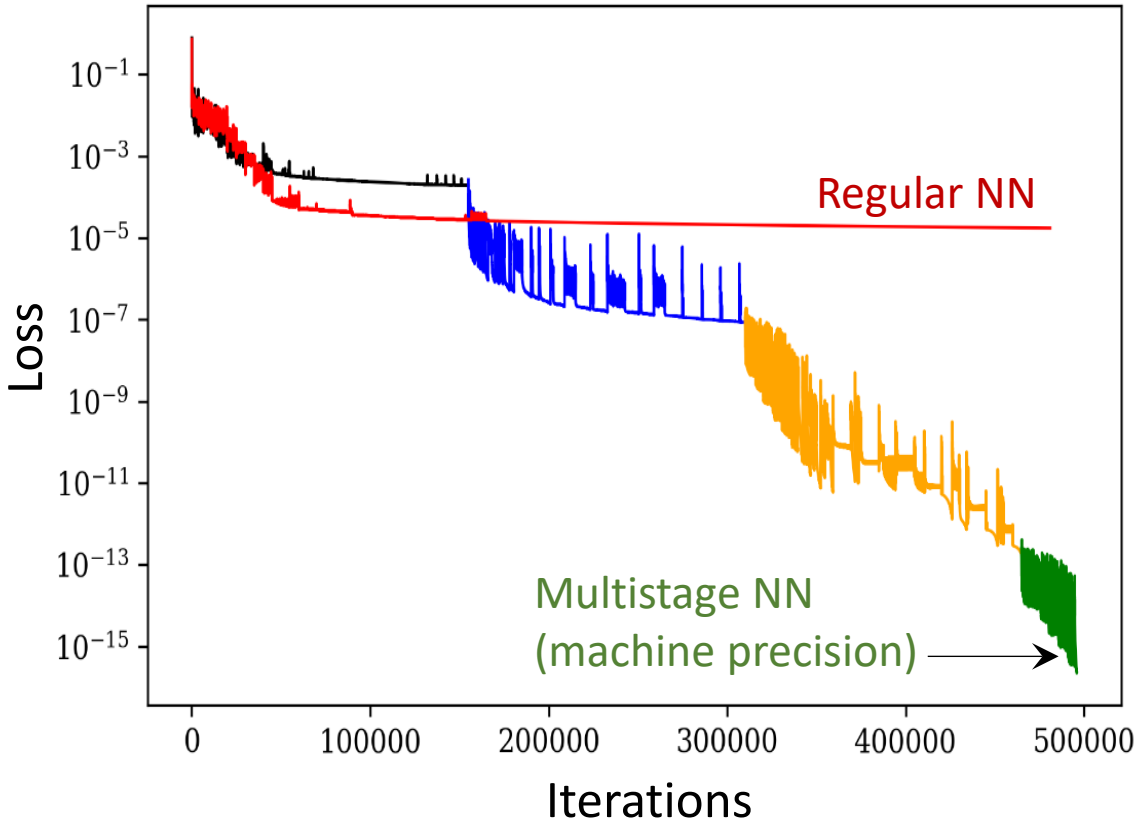
Multistage NN can tackle the spectral bias & accelerate loss convergence

Wang & Lai, J. Comput. Phys. (2024)
Ng, Wang & Lai, ICML workshop (2024), arXiv:2407.17213

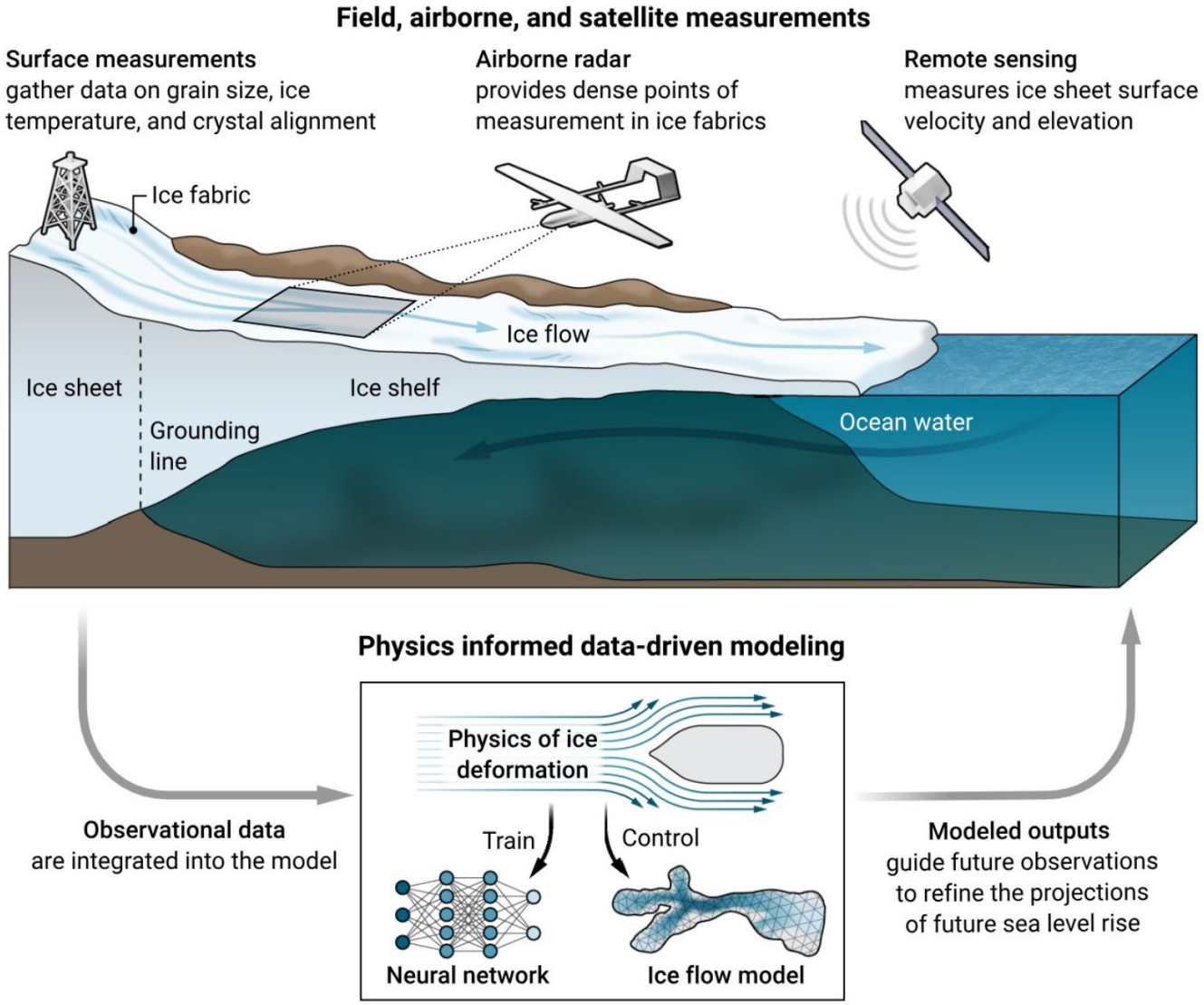
Energy spectrum



Loss convergence



Example: Learning viscosity model for glacial ice



Method:
We inject only the physics that we believe are correct, e.g. **conservation of momentum**, into the loss function, and leaving the uncertain physics, e.g. the **viscosity model** $\mu =?$, to be determined through data assimilation using observations.

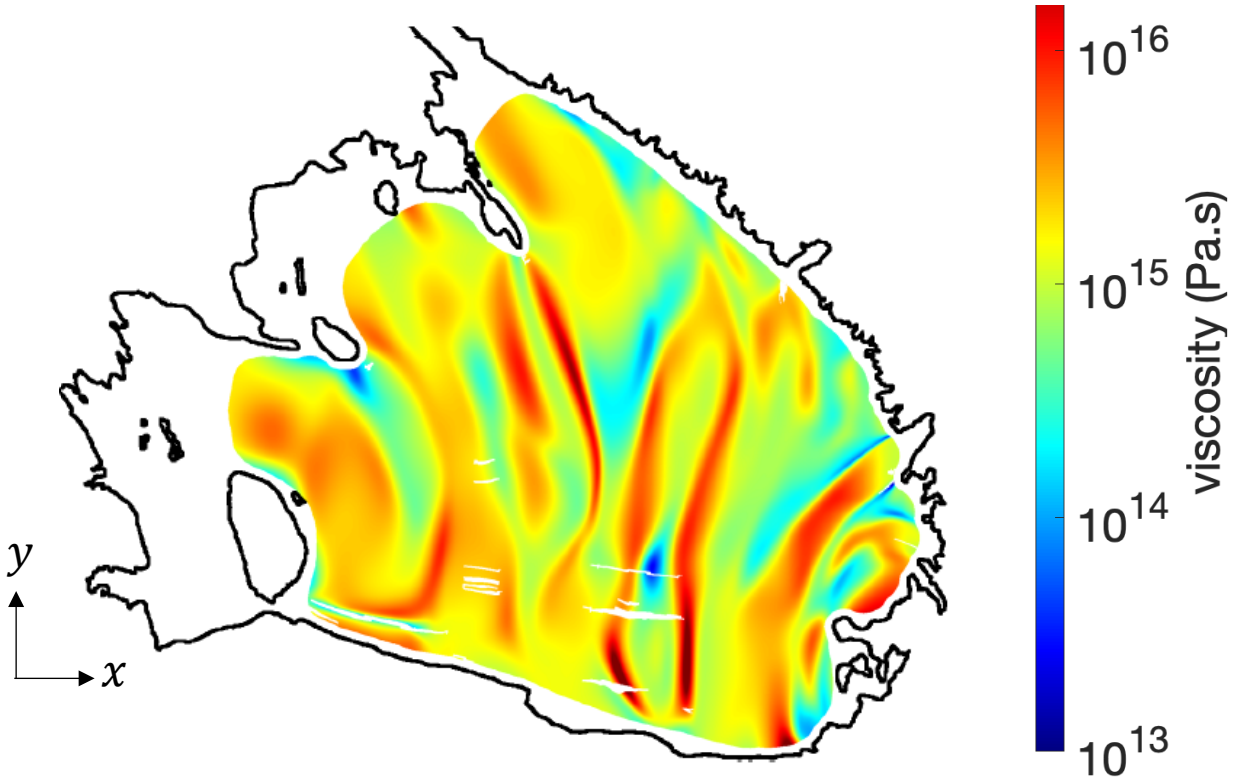
Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Example: Learning viscosity model for glacial ice

Ice effective viscosity μ

NN representation: 40k parameters

Uniform discretized grid: 2300k parameters



Method:

We inject only the physics that we believe are correct, e.g. **conservation of momentum**, into the loss function, and leaving the uncertain physics, e.g. the **viscosity model** $\mu = ?$, to be determined through data assimilation using observations.

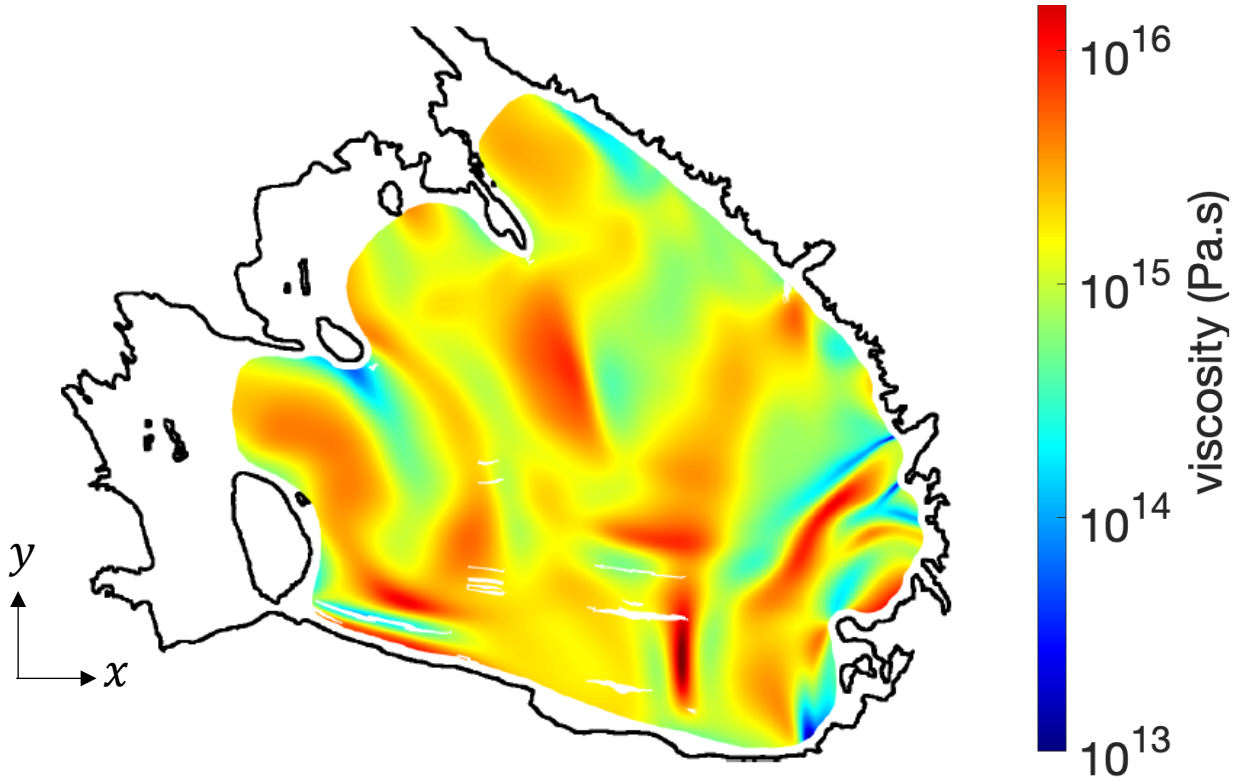
Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Example: Learning viscosity model for glacial ice

Ice effective viscosity μ

NN representation: 40k parameters

Uniform discretized grid: 2300k parameters

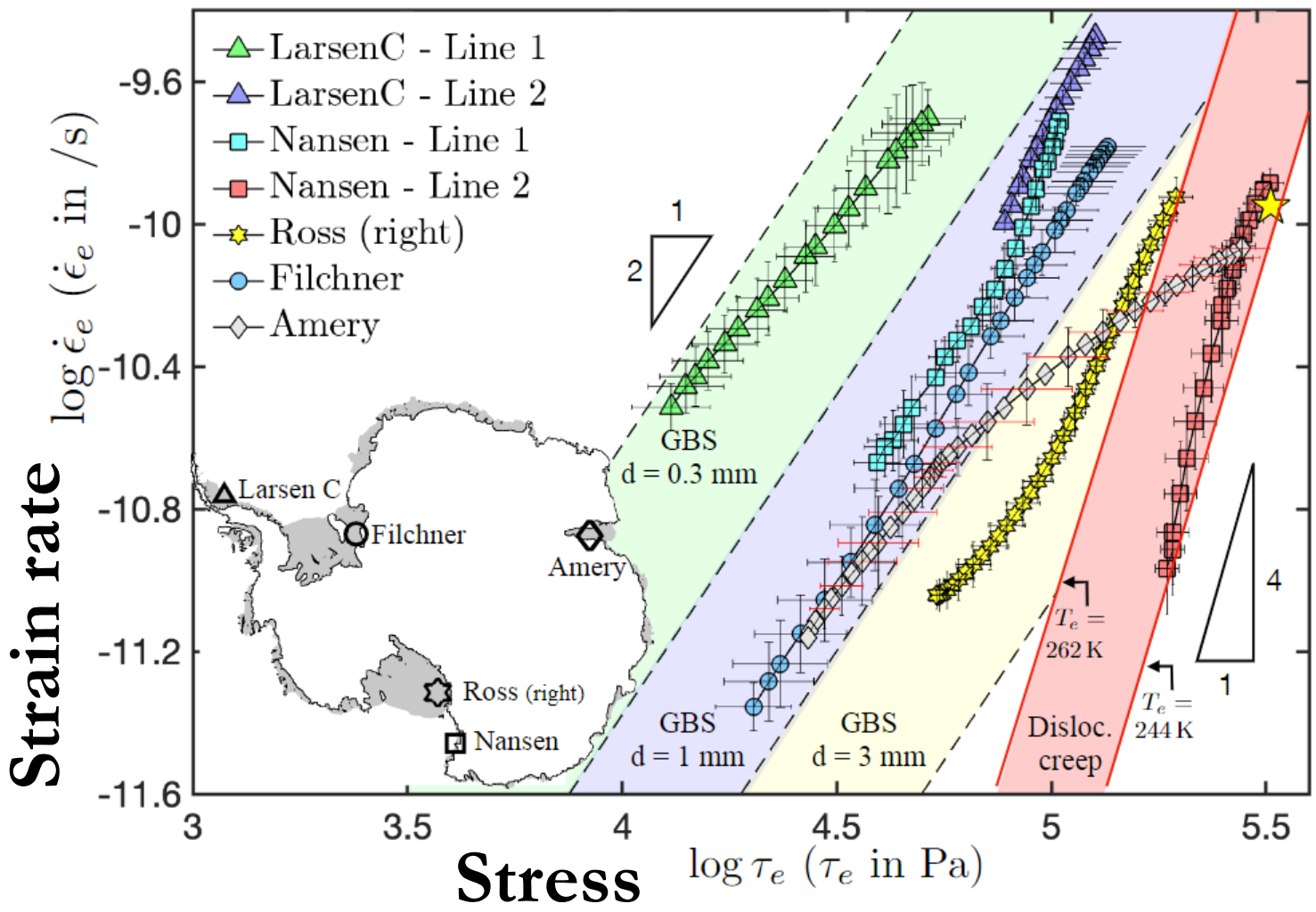


Method:

We inject only the physics that we believe are correct, e.g. **conservation of momentum**, into the loss function, and leaving the uncertain physics, e.g. the **viscosity model** μ =?, to be determined through data assimilation using observations.

Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Strain rate $\propto$ stress^n with stress exponent 1<n<4



- In conventional ice sheet models, n=3 is assumed.
- But here we observe: n increases with stress, important for predicting the fate of ice-sheet and sea-level rise.

Wang, Lai, Prior, Cowen-Breen, *Science* (2025)

Summary

- Neural networks can struggle to represent physics with scales spanning several orders of magnitudes due to a combination of its optimization dynamics and the data spectrum.
- We had tackled this issue with residual-correction idea and applied it to multiscale geophysical inverse problems.

Future work

- **Generalize to operators:** How can multistage neural operators potentially help mitigate the error propagation of autoregressive predictions?

Acknowledgement

References:

Wang & Lai, "Multi-stage neural networks: Function approximator of machine precision", *J. Comput. Phys.* (2024)

Wang, Lai, Prior, Cowen-Breen "Deep learning the flow law of Antarctic ice shelves", *Science* (2025)

Ng, Wang & Lai, *ICML workshop* (2024), arXiv:2407.17213

Yongji Wang
Postdoc



Jakin Ng
Student



Charlie Cowen-Breen
Student



Code available at:
https://github.com/YaoGroup/MultistageNN

Discussion questions

- In the context of weather forecast or climate predictions, when does "missing the small scales" matter?
- Lots of AI emulators seem to be blurry but stable and capable of predicting the trajectories/statistics. Why is that the case? Does it truly find the coarse-grained reduced-order model for the system?

ANNUAL REVIEWS

Lai et al., invited review article in *Annu. Rev. Condens. Matter Phys* (2025). doi.org/10.1146/annurev-conmatphys-043024-114758

Annual Review of Condensed Matter Physics

Machine Learning for Climate Physics and Simulations

Ching-Yao Lai,[1] Pedram Hassanzadeh,[2] Aditi Sheshadri,[3] Maike Sonnewald,[4] Raffaele Ferrari,[5] and Venkatramani Balaji[6]

[1]Department of Geophysics, Stanford University, Stanford, California, USA; email: cyaolai@stanford.edu

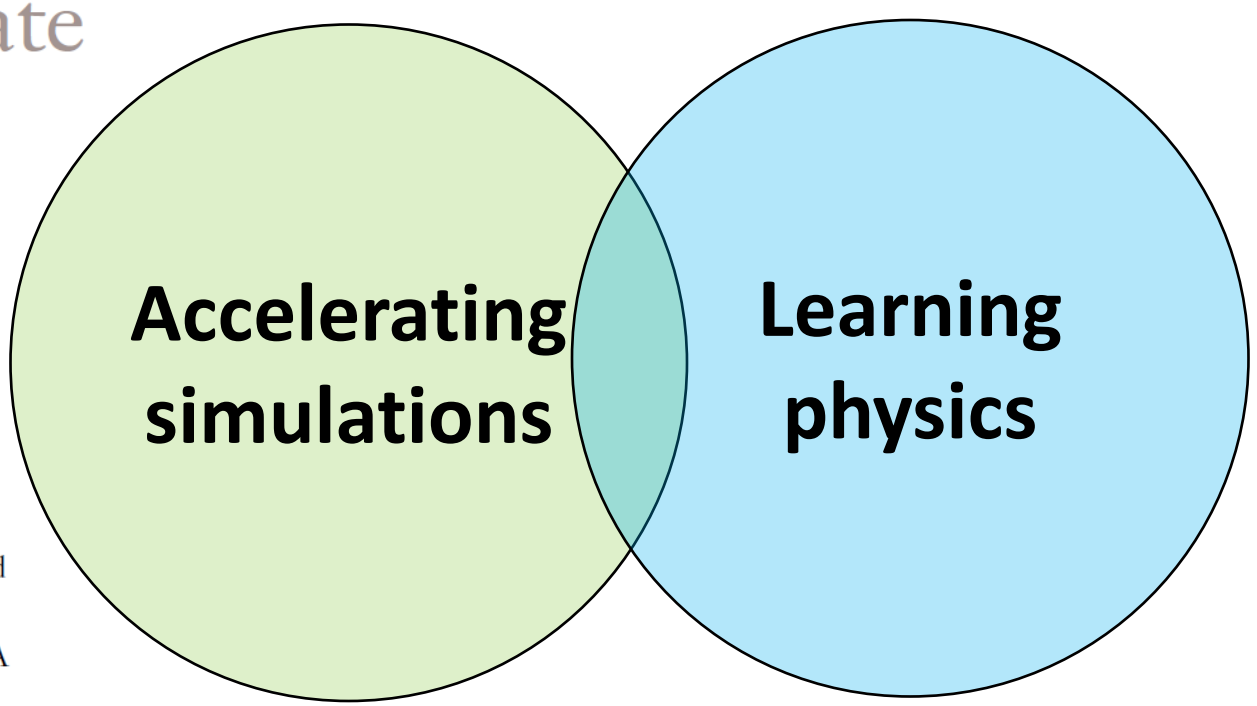
[2]Department of Geophysical Sciences and Committee on Computational and Applied Mathematics, University of Chicago, Chicago, Illinois, USA

[3]Department of Earth System Science, Stanford University, Stanford, California, USA

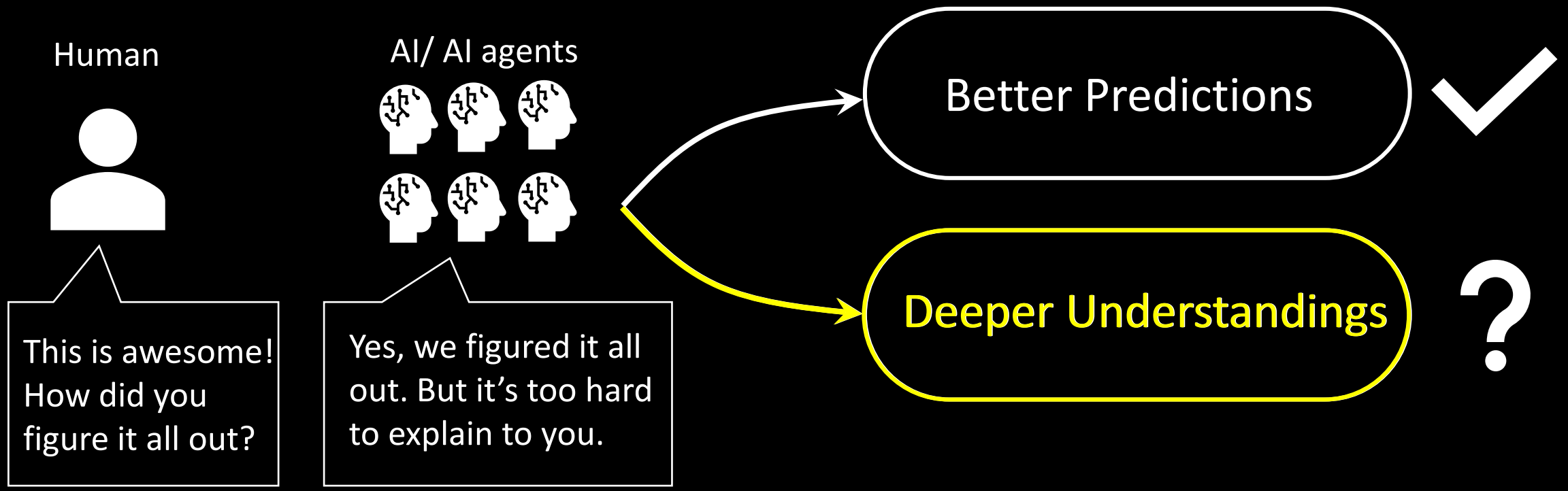
[4]Department of Computer Science, University of California, Davis, California, USA

[5]Department of Earth, Atmospheric, and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

[6]Schmidt Sciences, New York, NY, USA



- Imagine an extreme scenario: AI emulators/agents can make faster, more accurate predictions (more consistent observations) of various physical processes than traditional physics-based simulations.

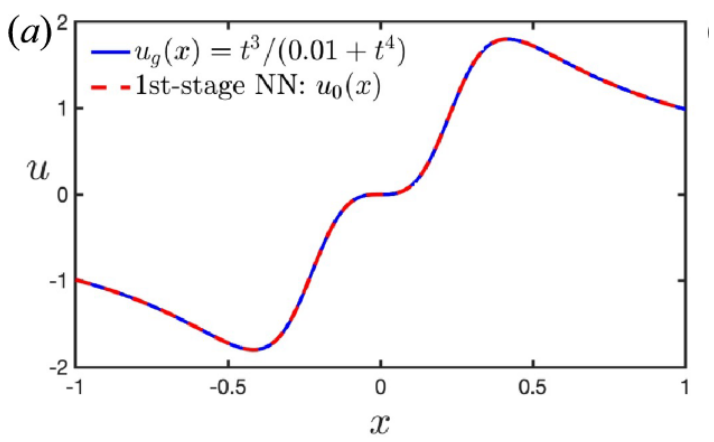


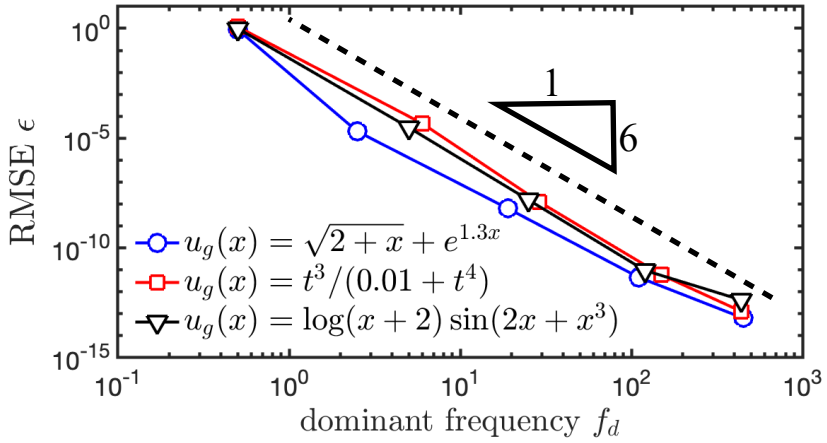
- Predictions ≠ Understandings

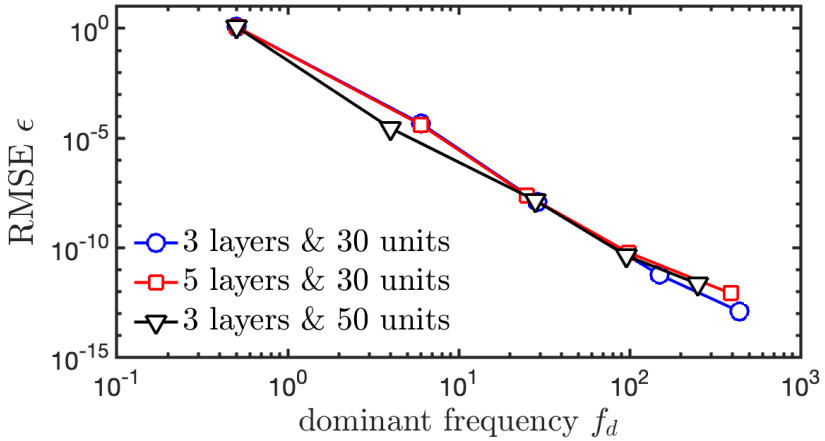
Isaac M. Held (2005), "The Gap between Simulation and Understanding in Climate Modeling"

Backup Slides

Scaling laws: Smaller errors, higher frequencies







Observed universal power-law:

$$f_d \propto \epsilon^{-1/\alpha} \text{ where } \alpha = 6$$

Error of $u_g(x)$ $\longrightarrow$ ϵ

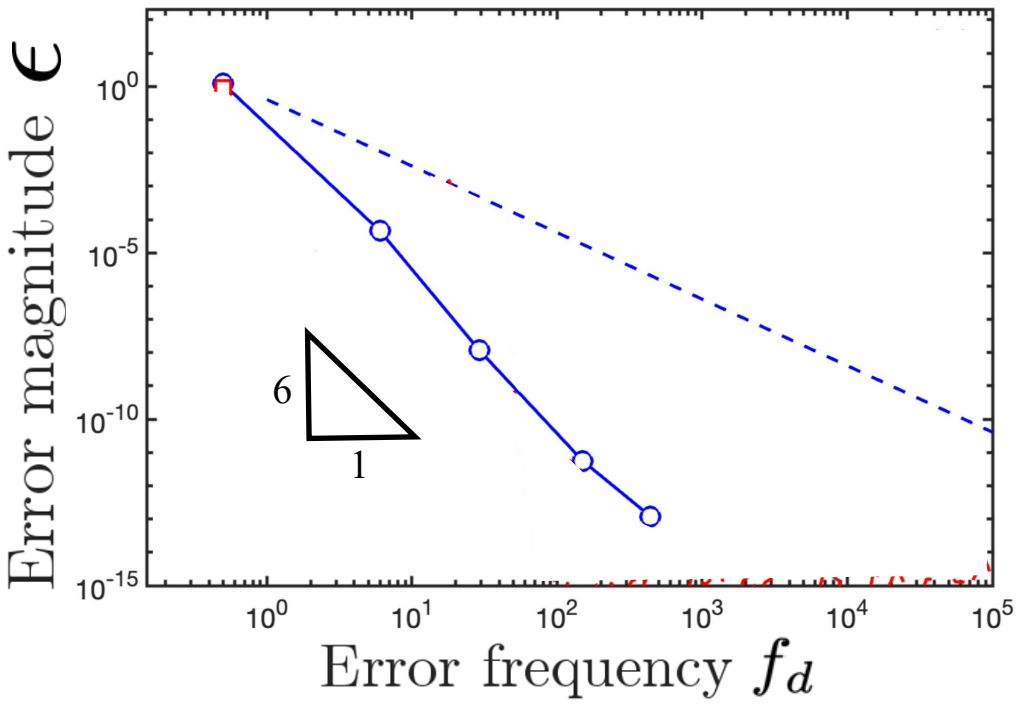
Error of $\frac{d}{dx}u_g(x)$ $\longrightarrow$ $\epsilon(2\pi f_d)$

Error of $\frac{d^n}{dx}u_g(x)$ $\longrightarrow$ $\epsilon(2\pi f_d)^n \sim \epsilon^{1-n/\alpha}$

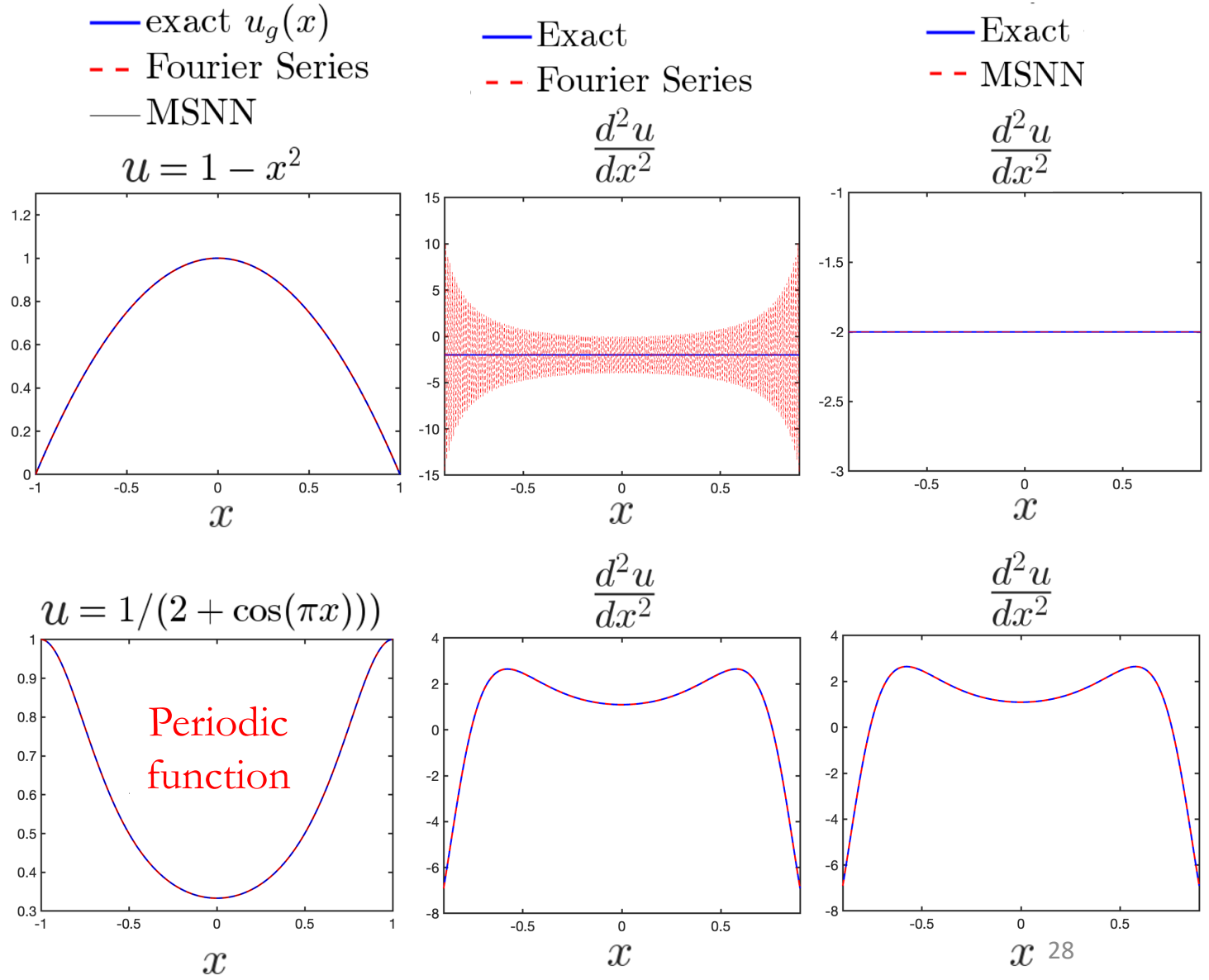
Multistage NN vs Fourier Series

Fourier Series
MSNN

Blue: Non-periodic function $u = 1 - x^2$



$$f_d \propto \epsilon^{-1/\alpha} \quad \text{where } \alpha = 6$$



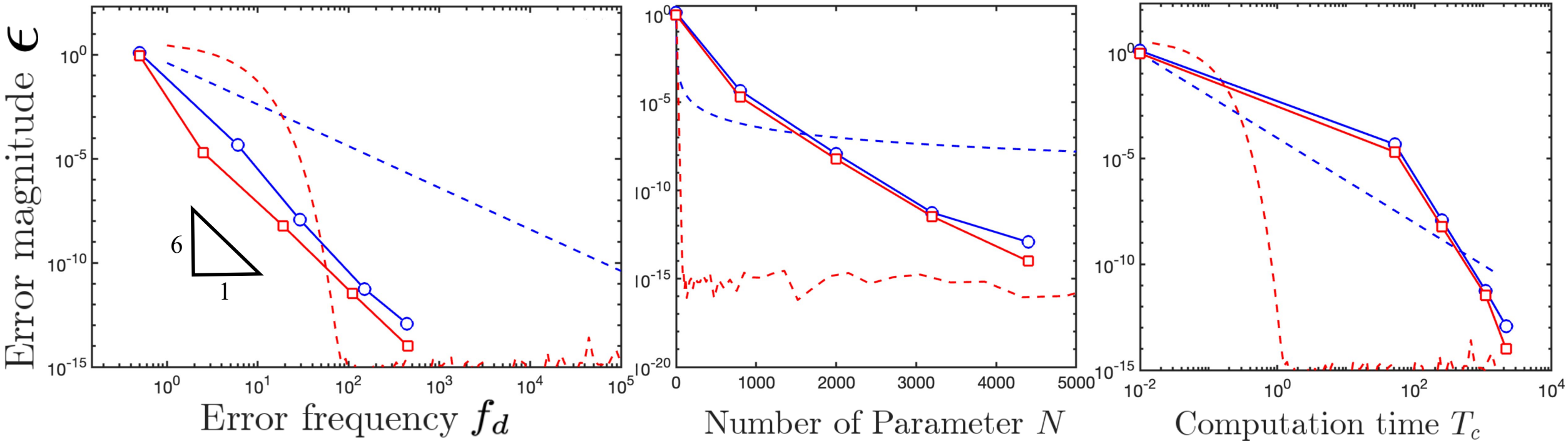
Multistage NN vs Fourier Series

- - Fourier Series

—○— MSNN

Red: Periodic function $u = 1/(2 + \cos(\pi x))$

Blue: Non-periodic function $u = 1 - x^2$

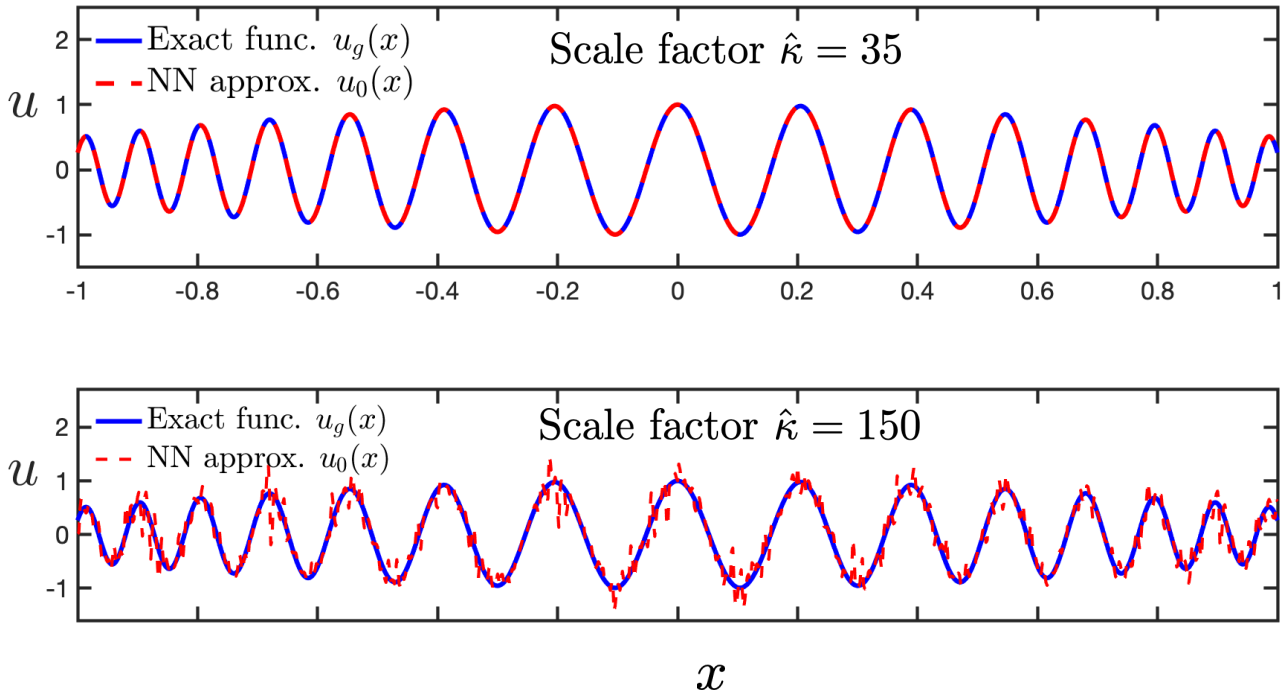


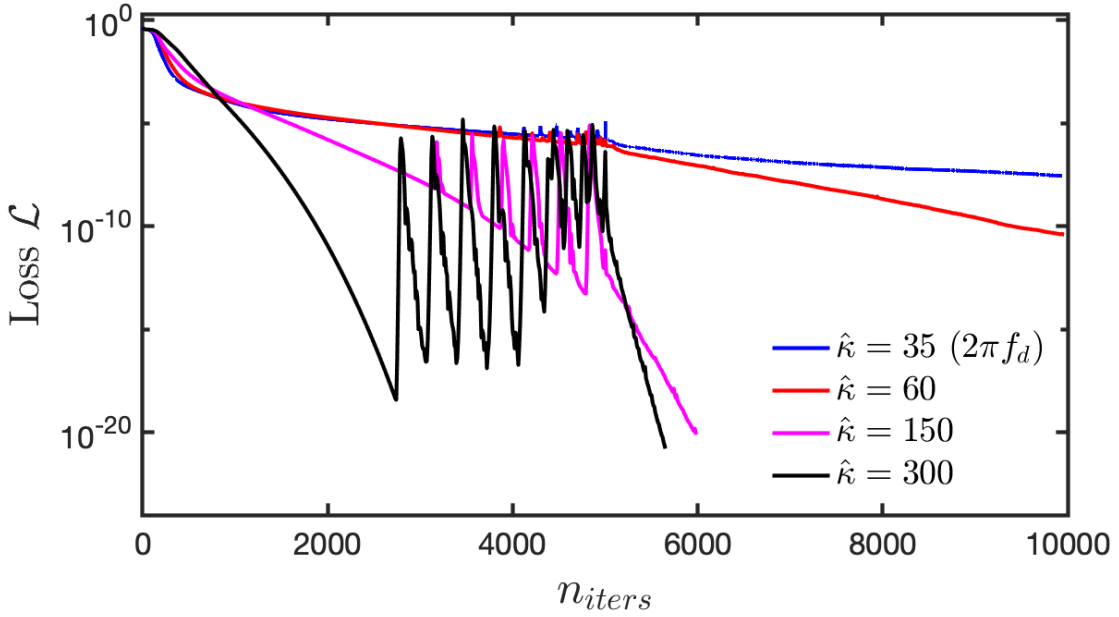
Optimal range of κ

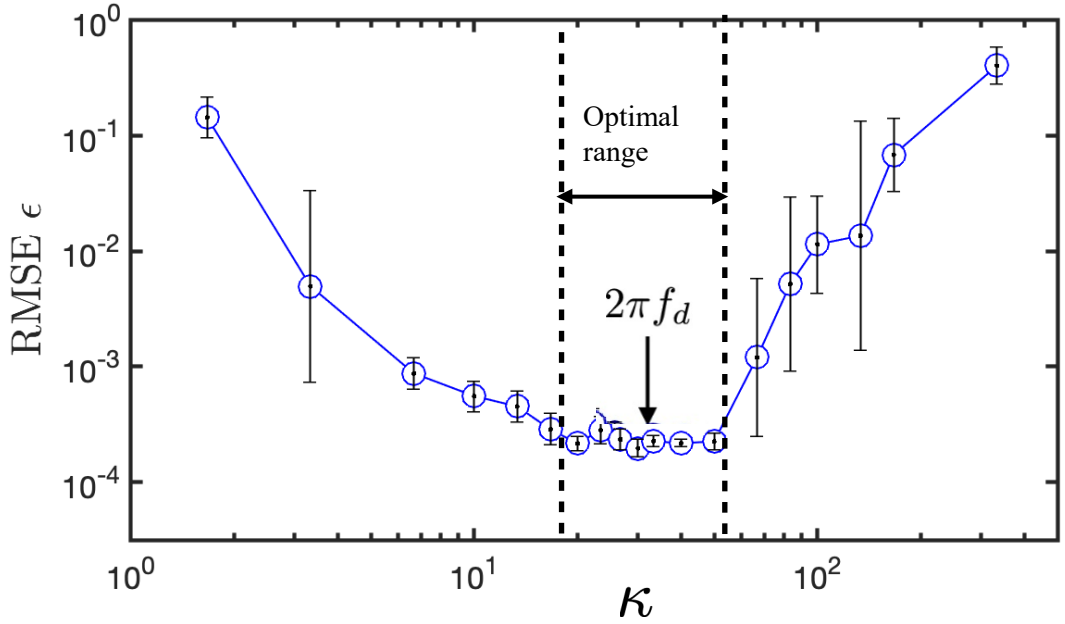
Our implementation:

First layer: $\sigma_0(x) = \sin\left(\kappa w_i^{(0)} x + b_i^{(0)}\right)$

κ: scaling factor

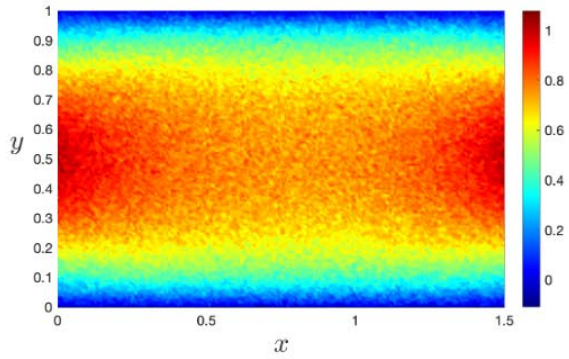




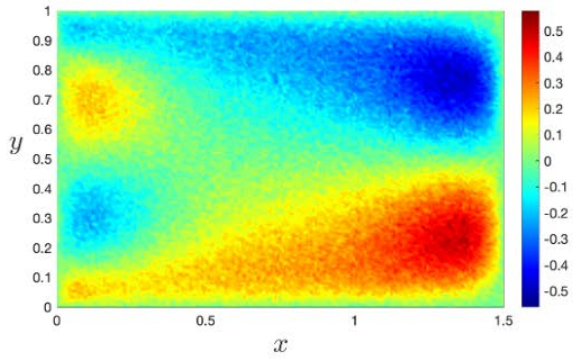


PINN inversion using training data with **noise (20%)**

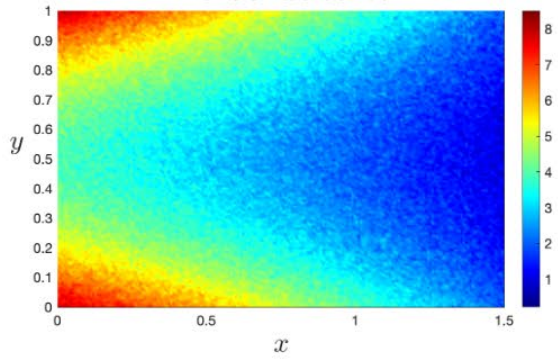
noise data $\tilde{u}$



noise data $\tilde{v}$



noise data $\tilde{h}$

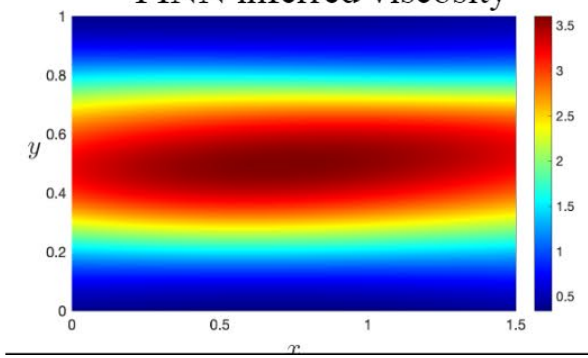


$$\tilde{q} = q(1 + 0.2X)$$

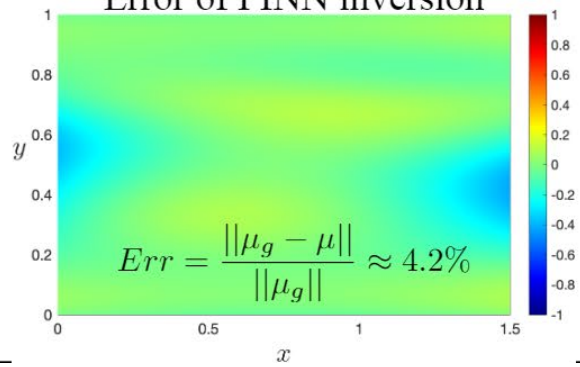
q represents u, v, h

random variable $X \sim N(0, 1)$

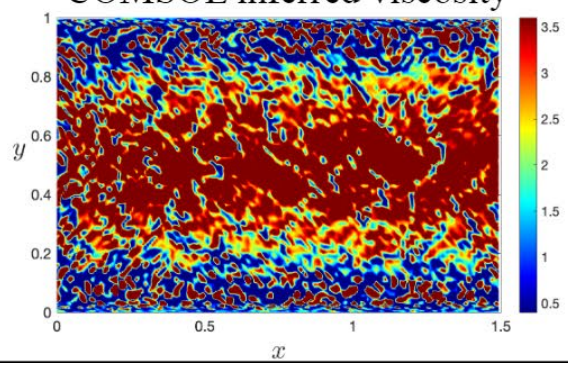
PINN inferred viscosity



Error of PINN inversion

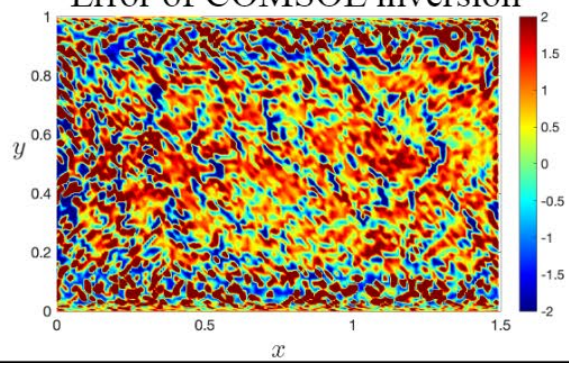


COMSOL inferred viscosity



$$Err = \frac{||\mu_g - \mu||}{||\mu_g||} \approx 63\%$$

Error of COMSOL inversion



We note that achieving a low prediction error of $O(10^{-11})$ for solving 2D partial differential equations via classical numerical method, such as finite difference, would require an extensive number of grid points. For instance, considering the central difference method along the *x*-direction, to reach 10^{-11}, we would need a grid size in the x-direction of $h_{(x)} \sim O(\sqrt{10^{-11}}) \sim O(10^{-5})$, namely 10^6 grid points for each time step. Even with a 4th-order Runge-Kutta method along the *t* direction, the step size in *t*-direction would need to be $h_{(x)} \sim O(\sqrt{(10^{-11})}^{1/4} \sim O(10^{-3})$, requiring 10^3 time steps. Consequently, the total number of grid points needed to achieve this accuracy across the entire domain would be on the order of $O(10^9)$.

In contrast, our approach utilizes fully connected neural network with 4-hidden layers consisting of 30 units for each stage of the training. Thus, the total number of weights and biases used to express the solution is only around $3 \times 4 \times 30^2 \approx 10^4$, which is five order of magnitude less than the number of grid points used in a discretized solution. This demonstrates the efficiency and effectiveness of the multi-stage PINN in achieving accurate solution with significantly fewer parameters compared to classical numerical methods.