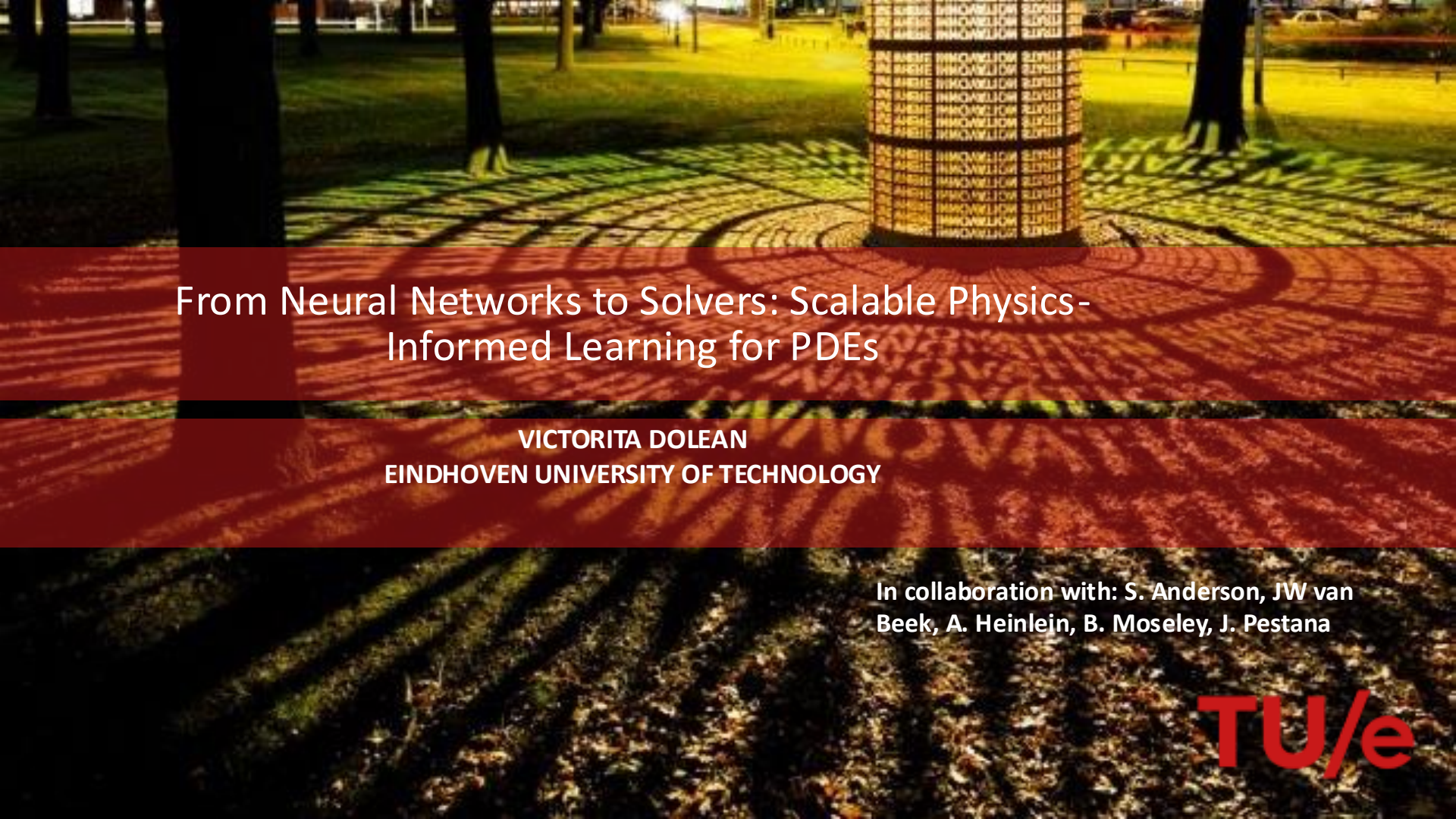


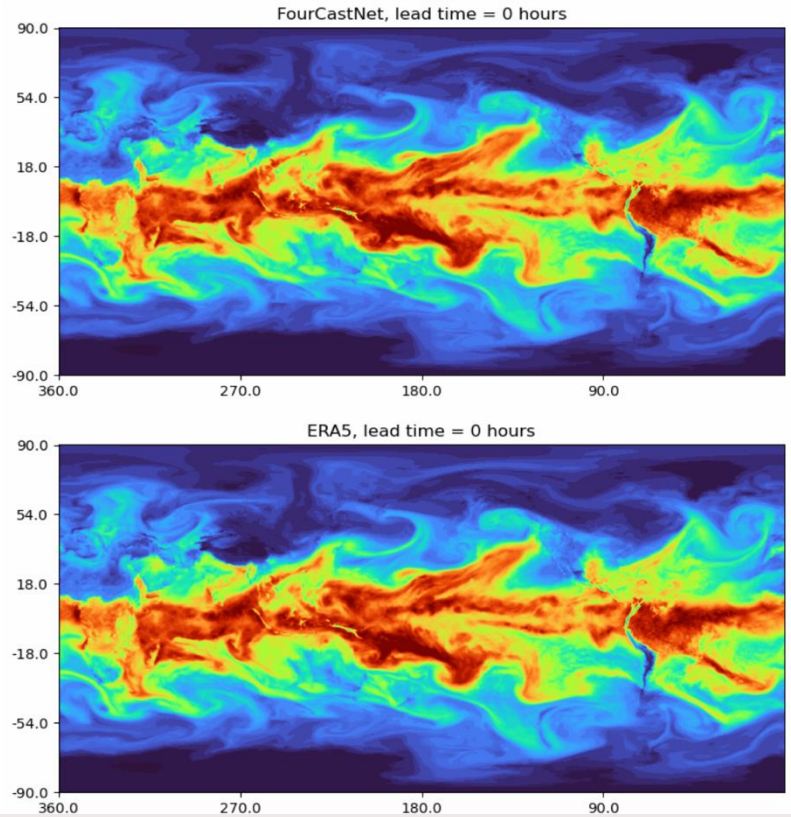
From Neural Networks to Solvers: Scalable Physics-Informed Learning for PDEs

VICTORITA DOLEAN
EINDHOVEN UNIVERSITY OF TECHNOLOGY

In collaboration with: S. Anderson, JW van Beek, A. Heinlein, B. Moseley, J. Pestana

TU/e

AI for science: a revolution?

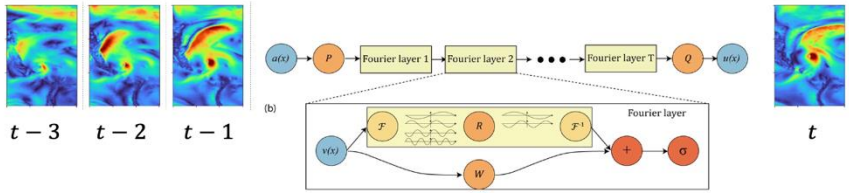


The Washington Post
Democracy Dies in Darkness

WEATHER Extreme Weather Climate Capital Weather Gang Environment Climate Lab

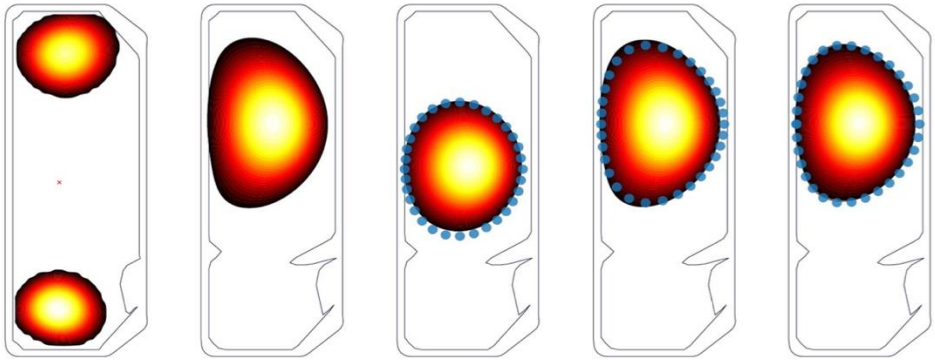
How Big Tech AI models nailed forecast for Hurricane Lee a week in advance

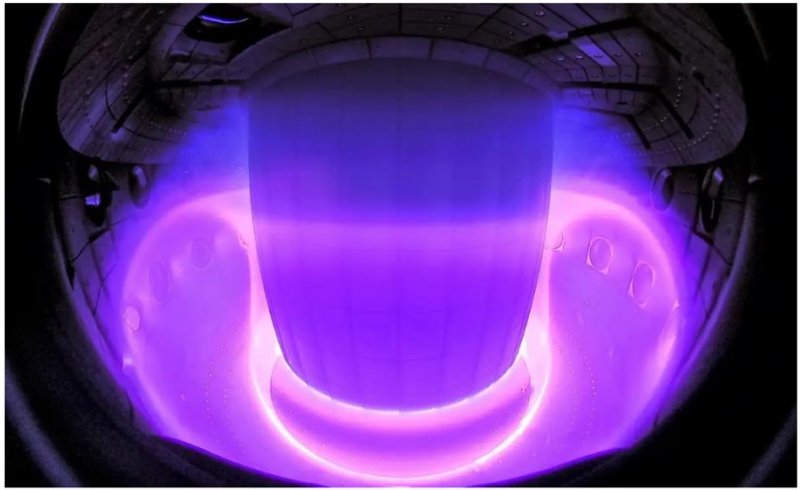
U.S. and European weather agencies are escalating their engagement with artificial intelligence as the technology rapidly advances



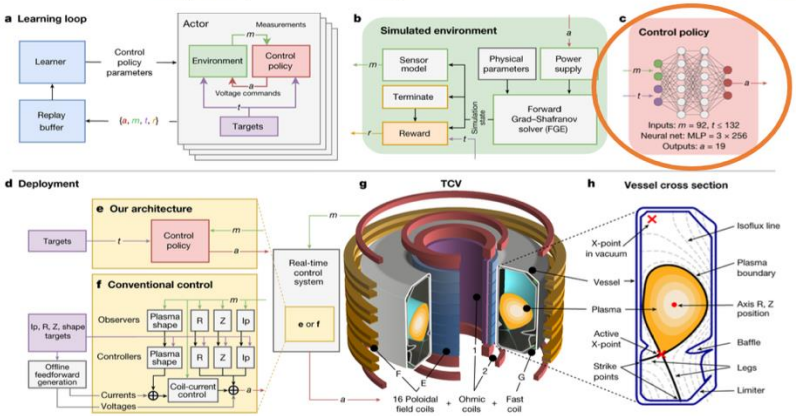
Pathak et al, FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators, ArXiv (2022)

AI for science: a revolution?





Variable Configuration Tokamak (TCV) in Lausanne, Switzerland
Source: DeepMind & SPC/EPFL



nature

Explore content · About the journal · Publish with us

nature › articles › article

Article | Open access | Published: 16 February 2022

Magnetic control of tokamak plasmas through deep reinforcement learning

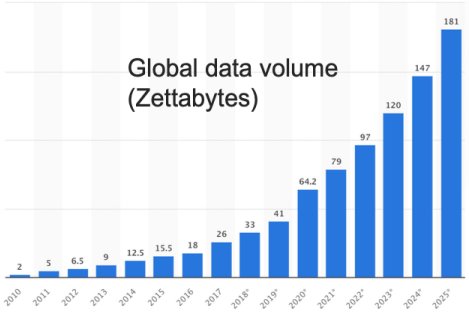
Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, … Martin Riedmiller + Show authors

Nature **602**, 414–419 (2022) | Cite this article

Why now?

Neural networks date back to the 1950's – so why is deep learning so popular today?

Rapidly increasing amounts of data

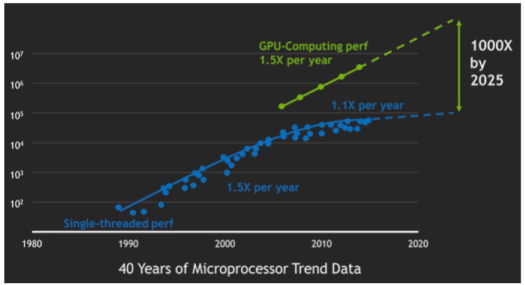


Source: Statista

IMAGENET

WIKIPEDIA
The Free Encyclopedia

Hardware improvements



Source: NVIDIA

- Graphical processing units (GPUs)
- Highly optimised for deep learning (massively parallel)

Software improvements

PyTorch

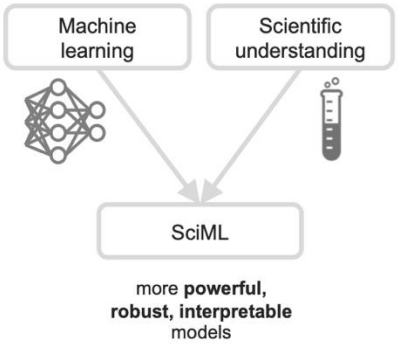
TensorFlow

Keras

- Mature deep learning frameworks
- Better training algorithms
- Deeper and more sophisticated architectures

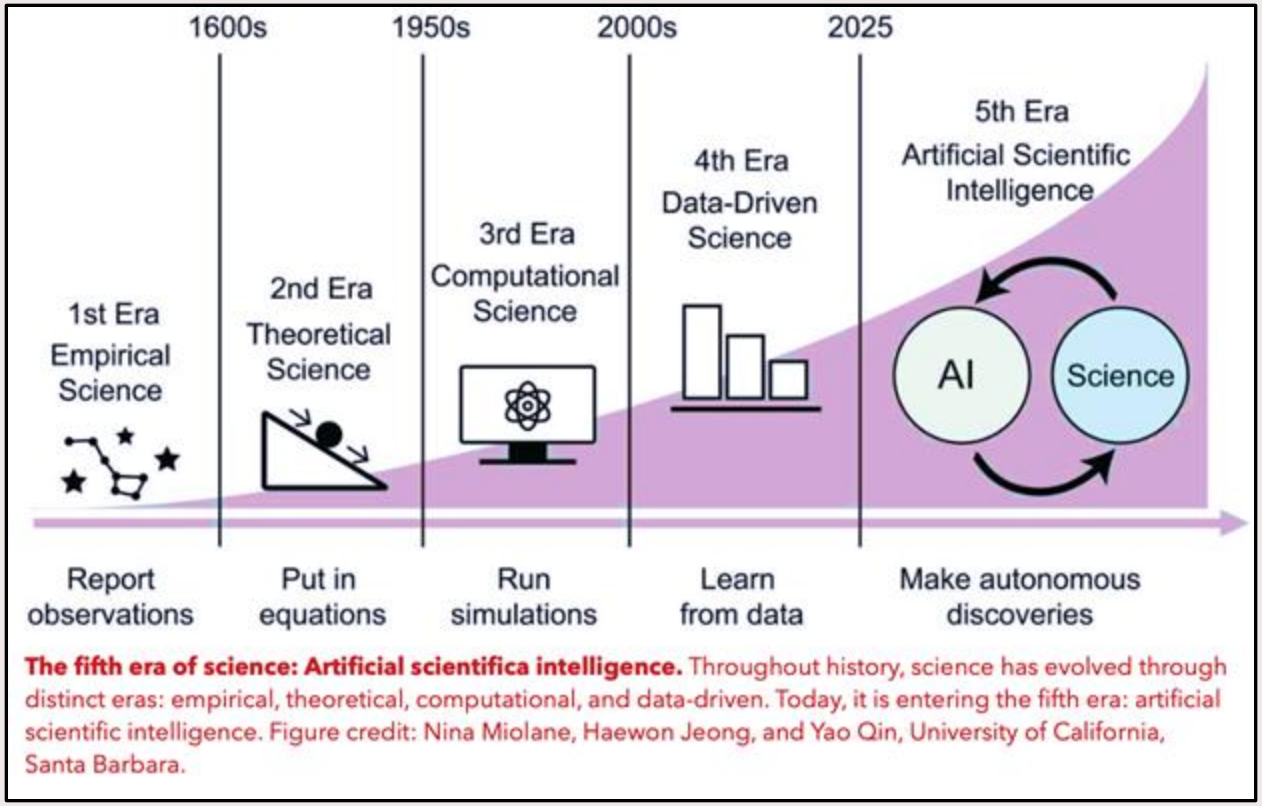
Scientific machine learning

Hamiltonian neural networks
Solver-in-the-loop
Learned sub-grid processes
Physics-constrained Gaussian processes
Physics-informed neural networks
Hidden physics models
AI Feynman
DeepONets
PDE-NetAlgorithm unrolling
AlphaFold
Learned regularisation
Differentiable simulation
Physics-informed neural operators
Fourier neural operators
Encoding conservation laws
Encoding physical symmetries
Neural ODEs



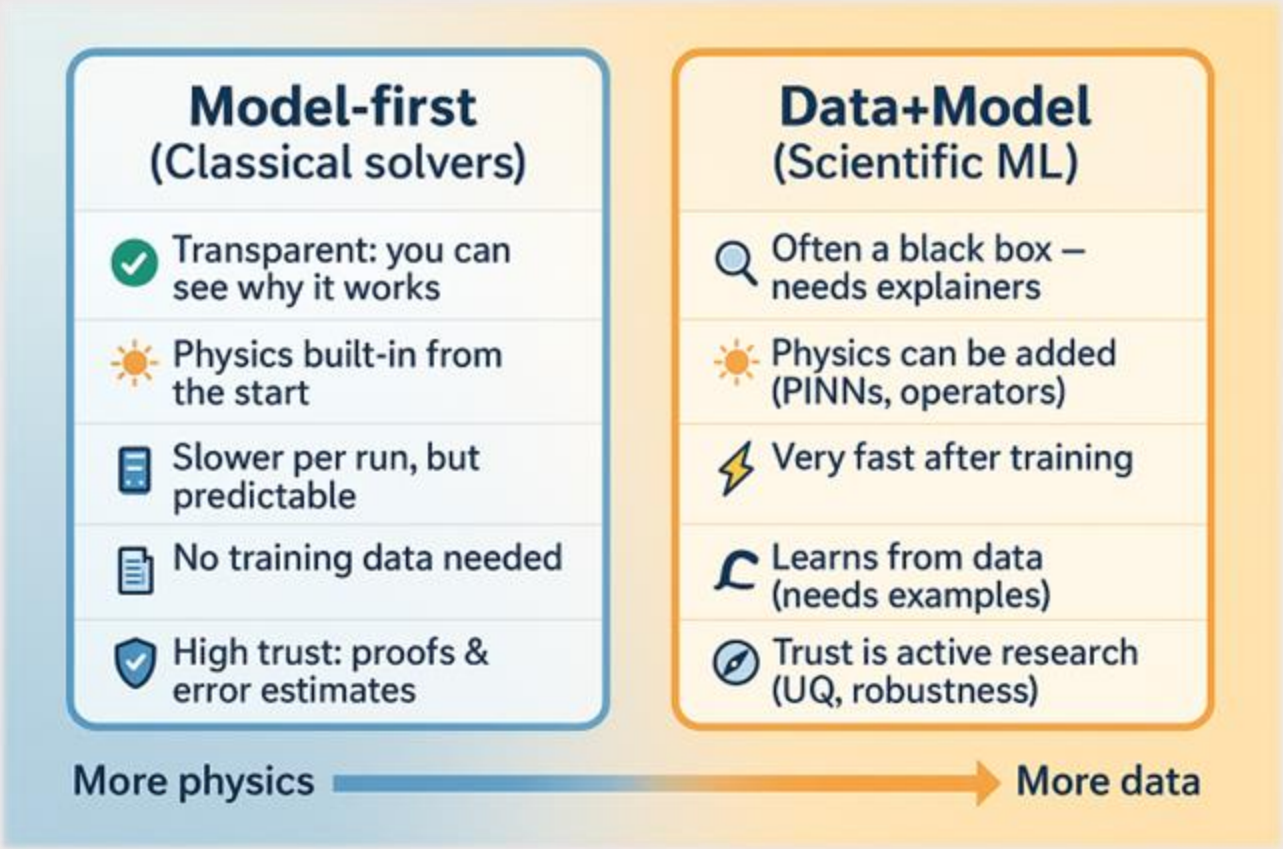
Scientific Machine Learning

(part of Artificial Scientific Intelligence: AI4Science – Science4AI)

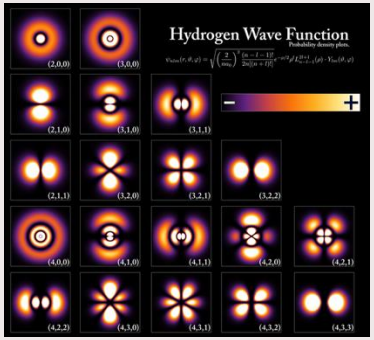


The fifth era of science: Artificial scientifica intelligence. Throughout history, science has evolved through distinct eras: empirical, theoretical, computational, and data-driven. Today, it is entering the fifth era: artificial scientific intelligence. Figure credit: Nina Miolane, Haewon Jeong, and Yao Qin, University of California, Santa Barbara.

Classical Solvers vs Scientific ML

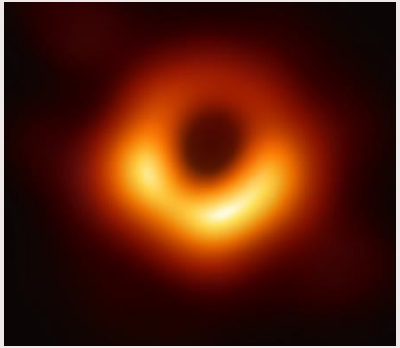


Importance of PDEs



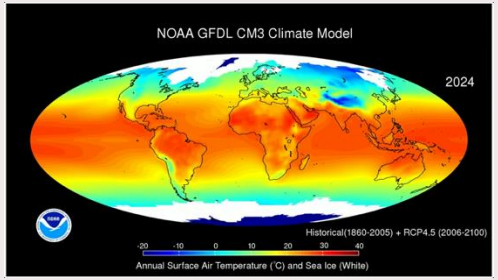
Source: Wikipedia

Schrödinger equation



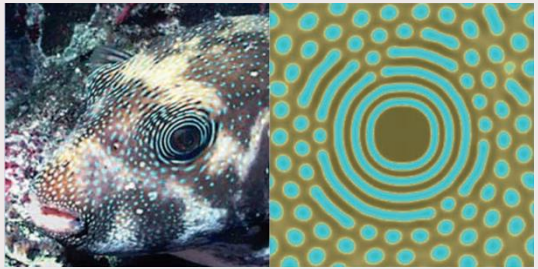
Source: The Event Horizon Telescope (2019)

Einstein field equations



Source: NOAA

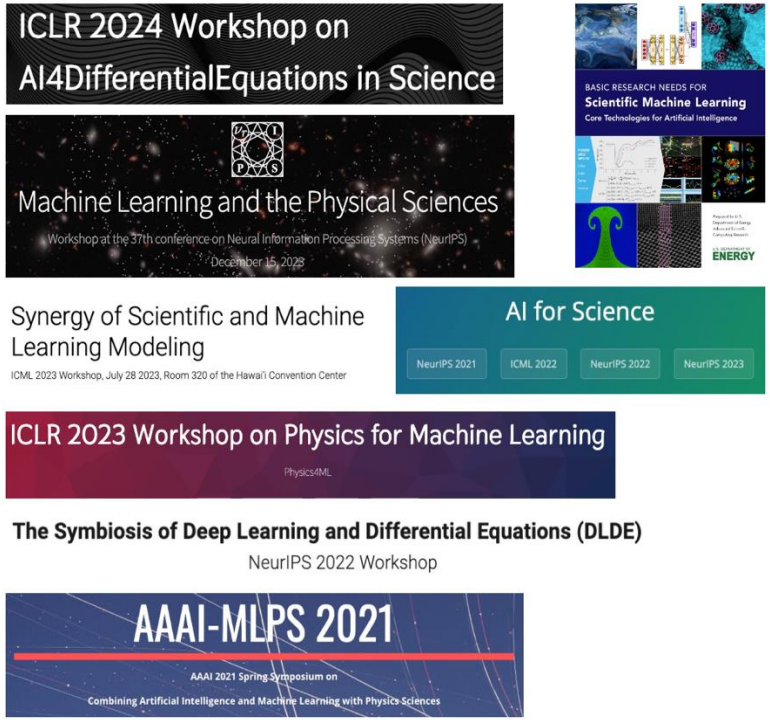
Navier-Stokes equations

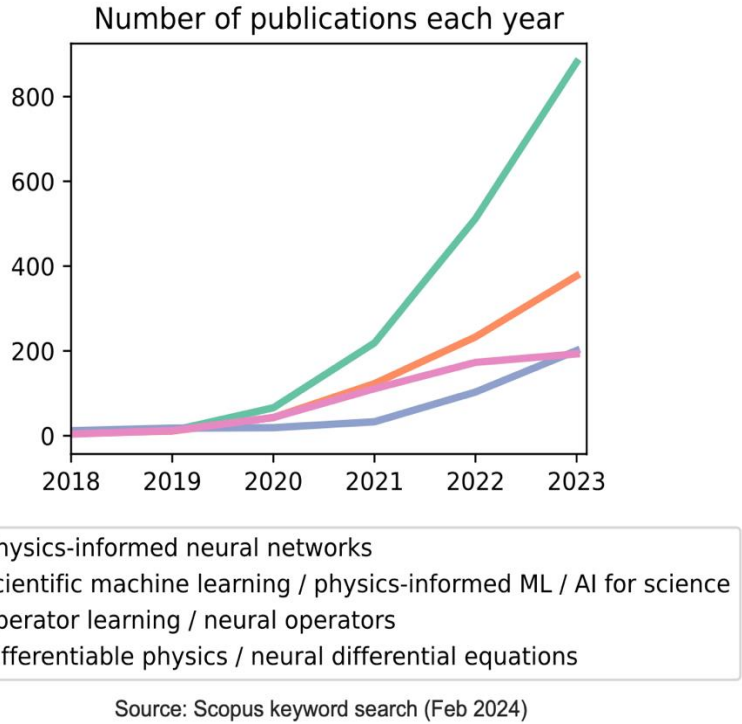


Source: Kondo and Miura, Science (2010)

Reaction-diffusion equation

A rapidly growing field

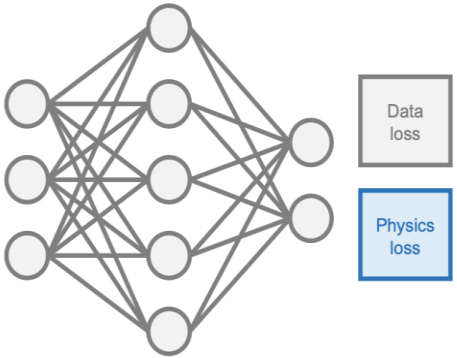




Source: Scopus keyword search (Feb 2024)

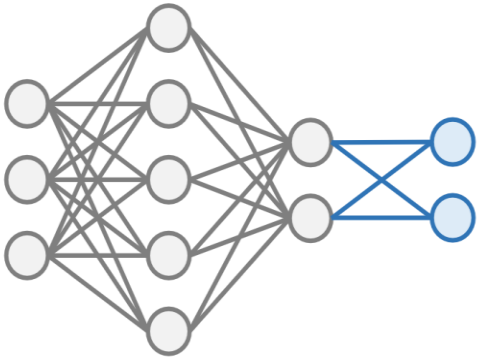
Ways to incorporate scientific principles into machine learning

Loss function



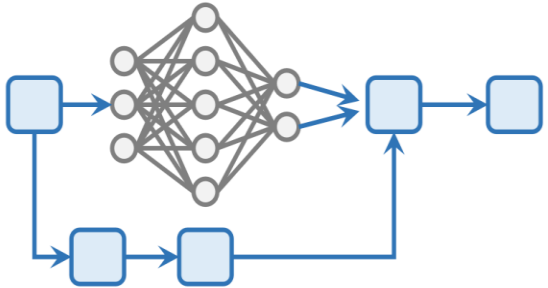
Example:
Physics-informed neural networks (add governing equations to loss function)

Architecture



Example:
Encoding symmetries / conservation laws (e.g. energy conservation, rotational invariance)

Hybrid approaches



Example:
Neural differential equations (incorporating neural networks into PDE models)

Can physics-informed neural networks (PINNs) beat finite difference / finite element methods?

Can physics-informed neural networks beat the finite element method?

Tamara G Grossmann ✉, Urszula Julia Komorowska, Jonas Latz, Carola-Bibiane Schönlieb

IMA Journal of Applied Mathematics, Volume 89, Issue 1, January 2024, Pages 143–174, https://doi.org/10.1093/imamat/hxae011
Published: 23 May 2024 **Article history** ▾

Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks

Petr Karnakov, Sergey Litvinov, Petros Koumoutsakos ✉ Author Notes

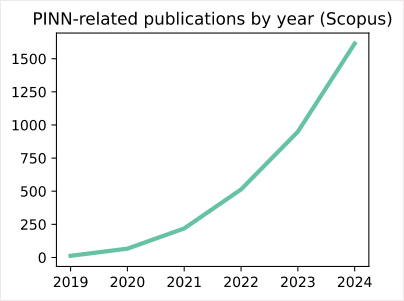
PNAS Nexus, Volume 3, Issue 1, January 2024, pgae005, https://doi.org/10.1093/pnasnexus/pgae005
Published: 11 January 2024 **Article history** ▾

7. Discussion and conclusions

After having investigated each of the PDEs on its own, let us now discuss and draw conclusions from the results as a whole. Considering the solution time and accuracy, PINNs are not able to beat FEM in our study. In all the examples that we have studied, the FEM solutions were faster at the same or at a higher accuracy.

Conclusion

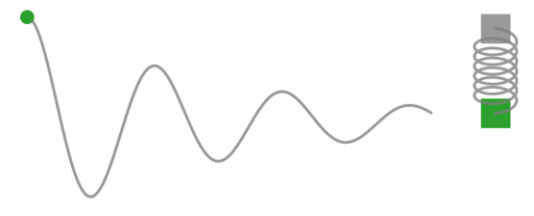
We introduce the ODIL framework for solving inverse problems for PDEs by casting their discretization as an optimization problem and applying optimization techniques that are widely available in machine-learning software. The concept of casting the PDE as is closely related to the neural network formulations proposed by (15–17) and recently revived as PINNs. However, the fact that we use the discrete approximation of the equations allows for ODIL to be orders of magnitude more efficient in terms of computational cost and accuracy compared to the PINN for which complex flow problems "remain elusive" (71).



TITLE-ABS-KEY ("physics-informed neural network" OR "physics informed neural network")

What is a physics-informed neural network?

Problem: damped harmonic oscillator

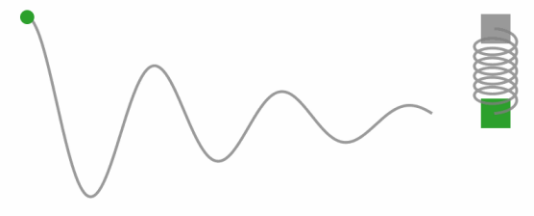


$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

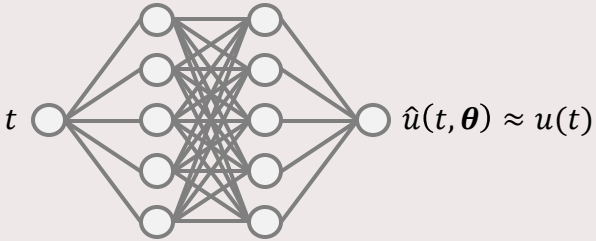
Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, JCP (2018)
Lagaris et al, Artificial neural networks for solving ordinary and partial differential equations, IEEE (1998)

What is a physics-informed neural network?

Problem: damped harmonic oscillator



$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$

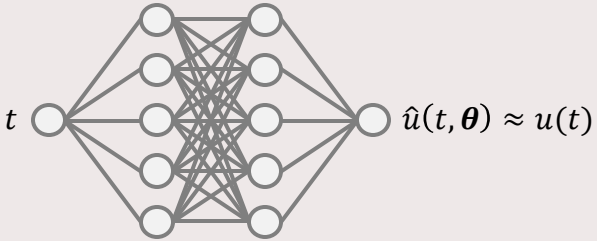


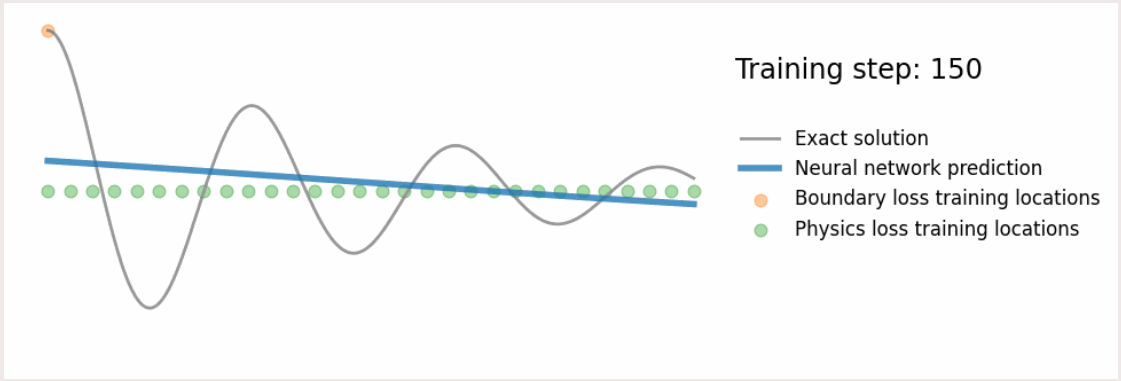
Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, JCP (2018)
Lagaris et al, Artificial neural networks for solving ordinary and partial differential equations, IEEE (1998)

What is a physics-informed neural network?

Problem: damped harmonic oscillator

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = 0$$





Boundary loss

$$L(\theta) = \lambda_1(\hat{u}(t=0, \boldsymbol{\theta}) - 1)^2 + \lambda_2\left(\frac{d\hat{u}}{dt}(t=0, \boldsymbol{\theta}) - 0\right)^2$$

Physics loss (aka PDE residual)

$$+ \frac{1}{N_p}\sum_{i}^{N_p}\left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k\right]\hat{u}(t_i, \boldsymbol{\theta})\right)^2$$

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

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

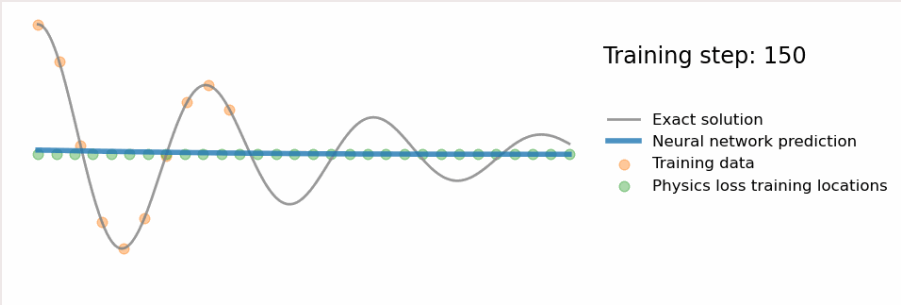
Scalability challenges of PINNs

Advantages of PINNs

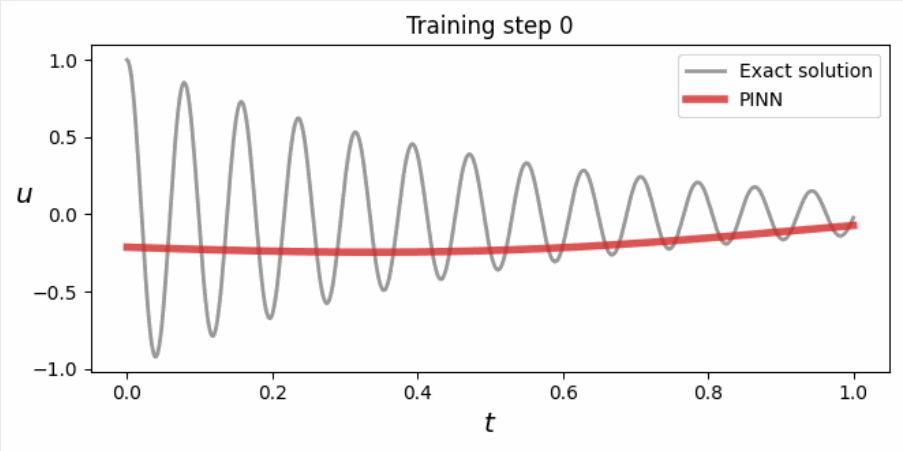
- **Mesh-free**
- Can solve forward and inverse problems, and seamlessly incorporate **observational data**
- Mostly **unsupervised**
- Can perform well for high-dimensional PDEs

Limitations of PINNs

- **Computational cost** often high (especially for forward-only problems)
- Can be hard to **optimise**
- Challenging to **scale** to high-frequency, multi-scale problems



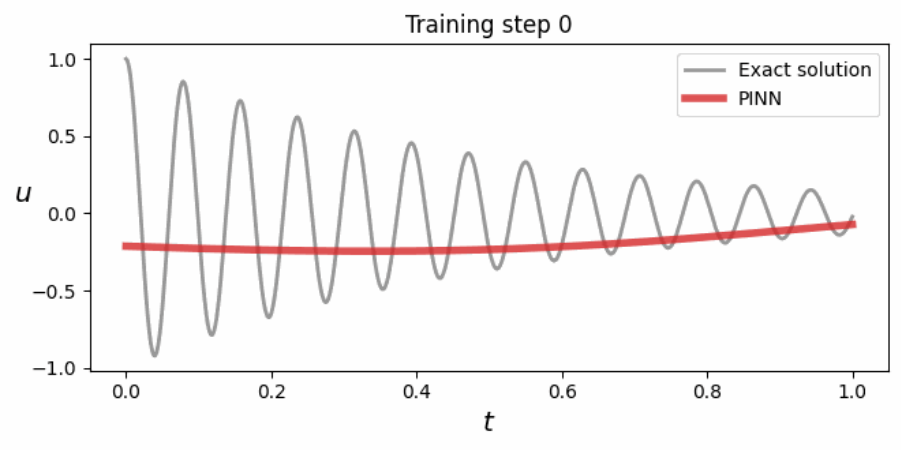
Scaling PINNs to high frequency / multiscale problems



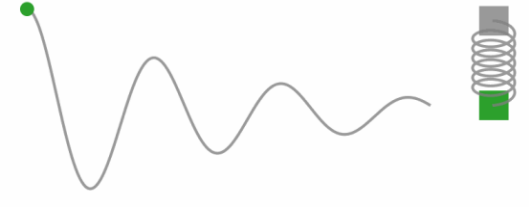
Problem: PINNs **struggle** to solve high-frequency / multiscale problems

Damped harmonic oscillator

Scaling PINNs to high frequency / multiscale problems



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



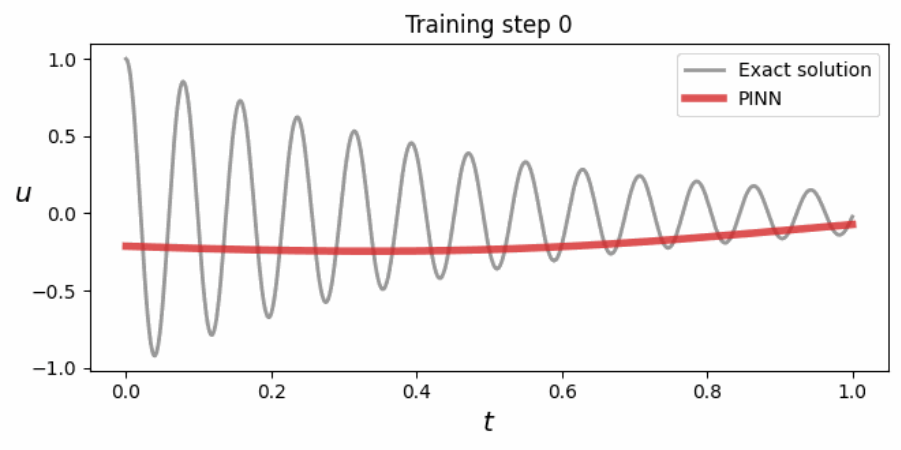
Damped harmonic oscillator

Spectral bias:

NNs tend to converge much **slower** on high frequencies than on low frequencies

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

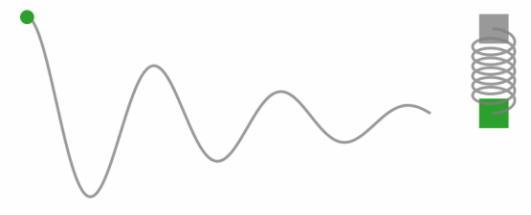
Scaling PINNs to high frequency / multiscale problems



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

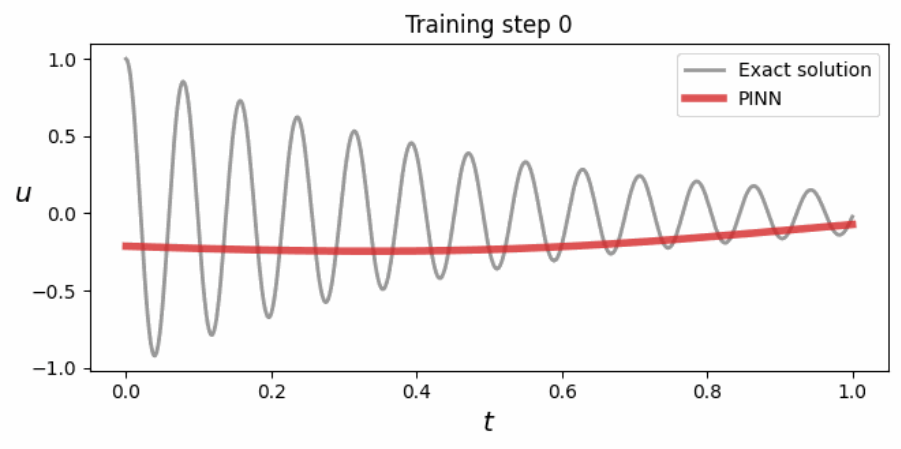
As higher frequencies are added:

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

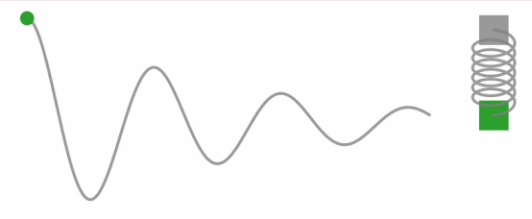


Damped harmonic oscillator

Scaling PINNs to high frequency / multiscale problems



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



Damped harmonic oscillator

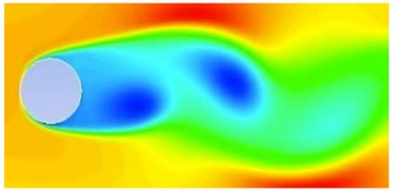
As higher frequencies are added:

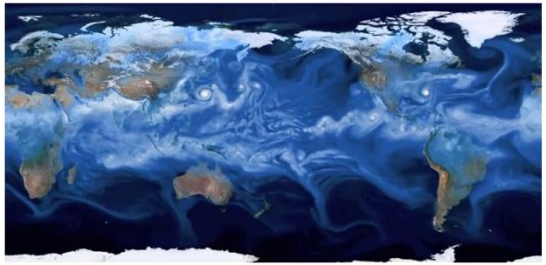
- More collocation points required ($\propto \omega^d$)
- Larger neural network required ($\propto f(\omega)$)
- Spectral bias slows convergence ($\propto s(\omega)$)

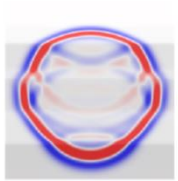
$\Rightarrow$ Empirically, cost of training often $\sim \mathcal{O}(\omega^d f(\omega) s(\omega))$

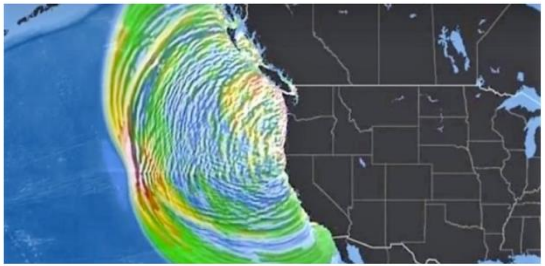
c.f. FD simulation, where cost of simulation can scale like $\sim \mathcal{O}(\omega^d)$

Scaling to more complex problems









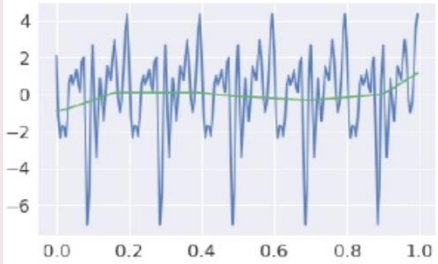
It is often challenging to **scale** traditional scientific algorithms to:

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

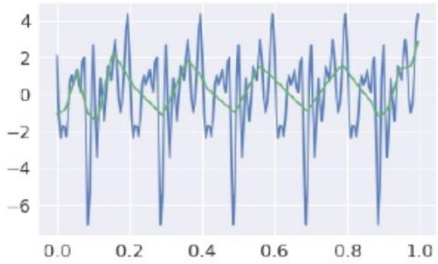
How do PINNs cope in this setting?

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

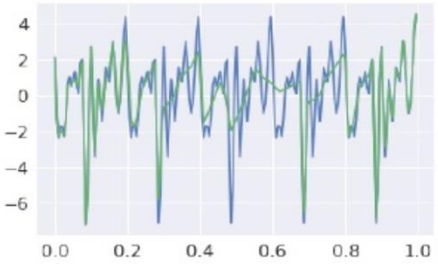
Spectral bias issue



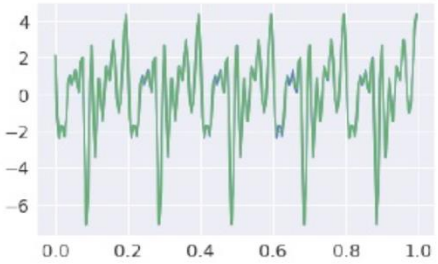
(a) Iteration 100



(b) Iteration 1000



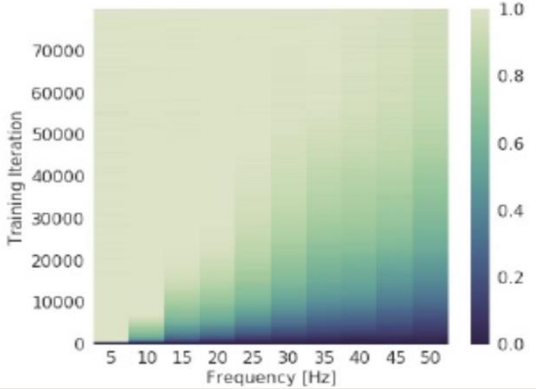
(c) Iteration 10000



(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)

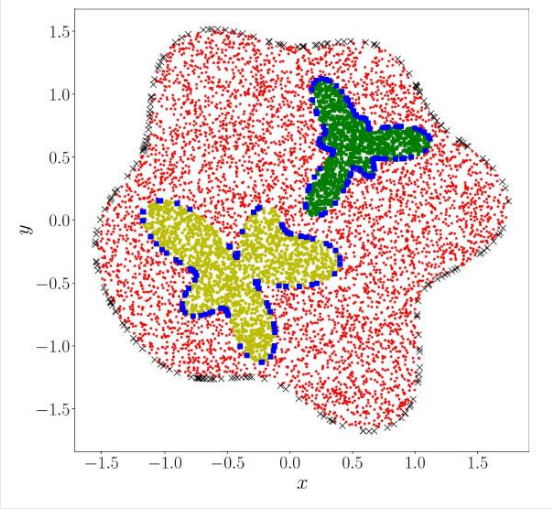
This behaviour can limit the model's ability to scale effectively, especially when the goal is to approximate complex scientific data with both large-scale and fine-scale dynamics.

Divide and conquer approaches tackle spectral bias

Partition of Unity Networks (POUNets) offer several mechanisms to mitigate spectral bias, especially in the context of scaling SciML models.

- **Partitioning the domain** into smaller regions,
- **Allowing localized learning** of high-frequency features.
- **Leveraging multiscale representations**, and
- **Ensuring smooth transitions across regions**

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:

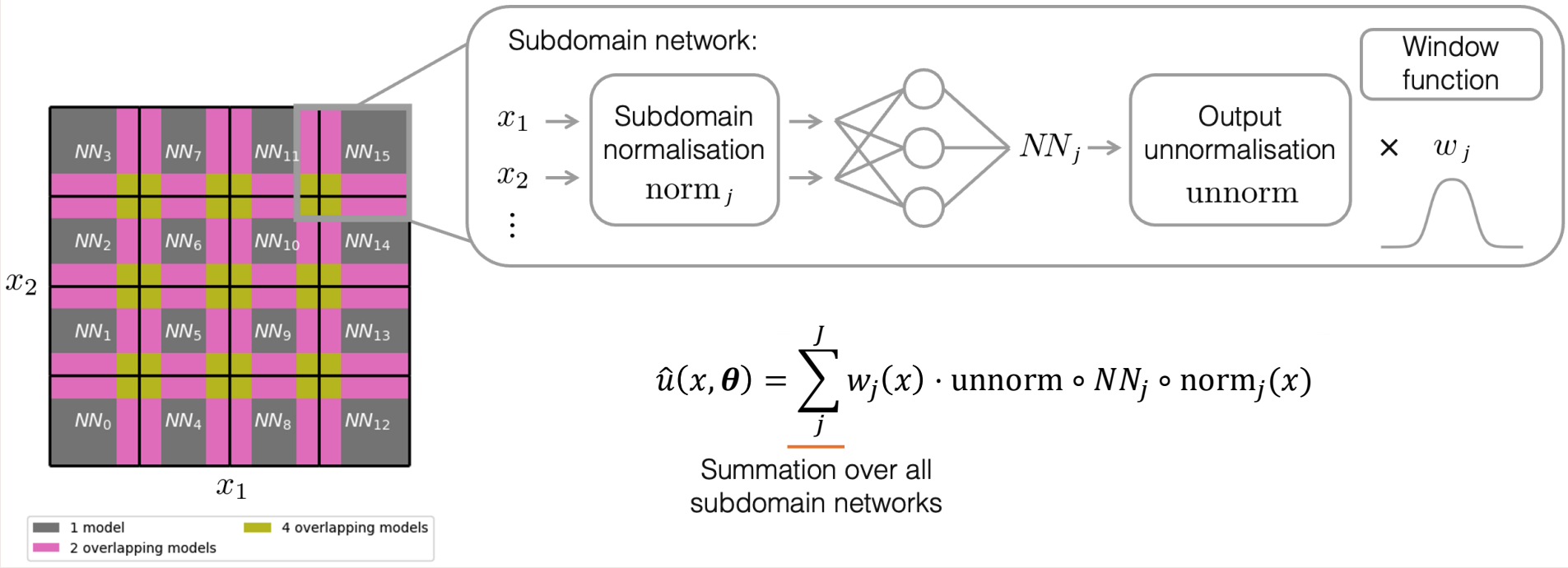
1. Divide modelling domain into many smaller **subdomains**

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

Finite basis PINNs (FBPINNs)



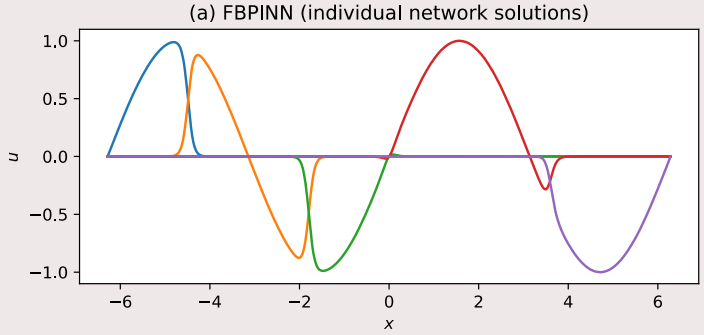
$$\hat{u}(x, \boldsymbol{\theta}) = \sum_{j}^{J} w_j(x) \cdot \text{unnorm} \circ NN_j \circ \text{norm}_j(x)$$

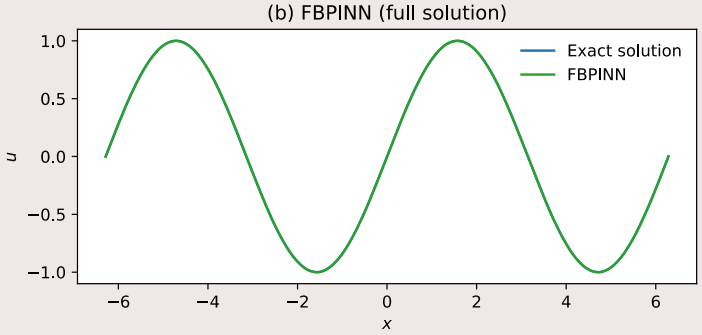
Summation over all subdomain networks

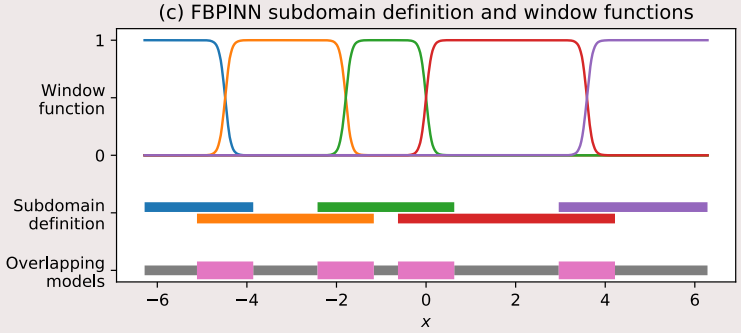
Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

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

FBPINNs in 1D







$$\hat{u}(x, \boldsymbol{\theta}) = \sum_j^J w_j(x) \cdot \text{unnorm} \circ NN_j \circ \text{norm}_j(x)$$

Window function

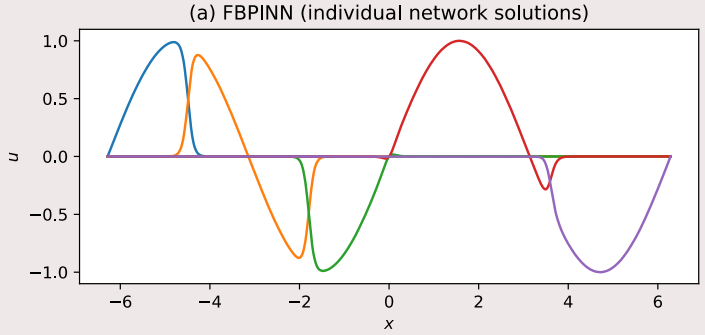
Subdomain network

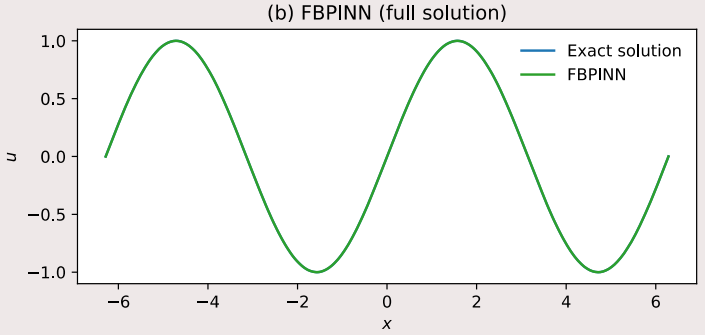
Individual subdomain normalisation

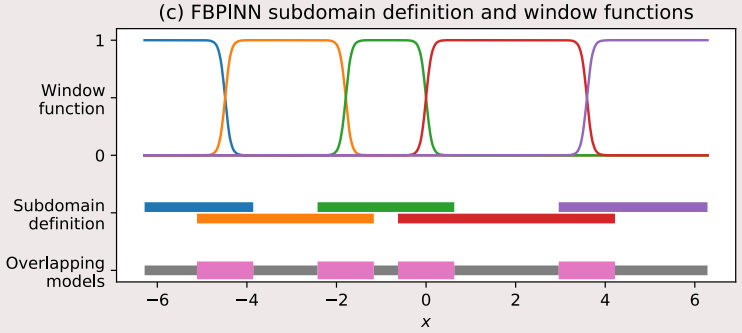
Idea: use **overlapping** subdomains and a **globally** defined solution ansatz

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

FBPINNs in 1D







$$\hat{u}(x, \boldsymbol{\theta}) = \sum_j^J w_j(x) \cdot \text{unnorm} \circ NN_j \circ \text{norm}_j(x)$$

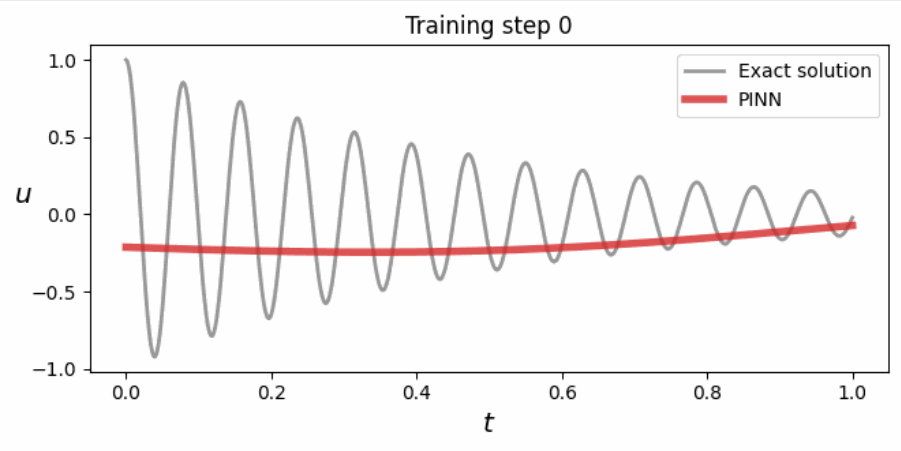
Window function | Subdomain network | Individual subdomain normalisation

Notes:

- FBPINNs can be trained with same loss function as PINNs
- And can simply be thought of as a “custom architecture”

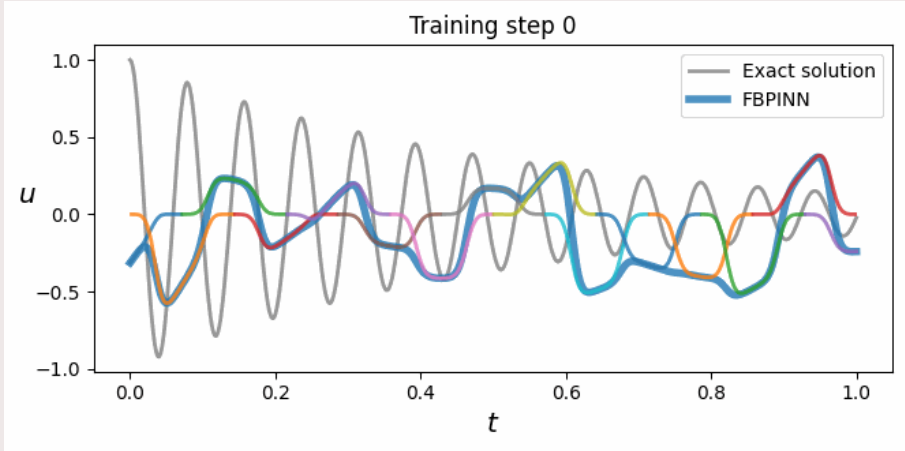
Moseley et al, Finite Basis Physics-Informed Neural Networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, ACM (2023)

FBPINNs vs PINNs



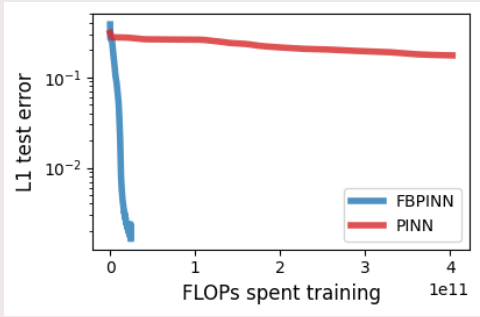
Problem: PINNs **struggle** to solve high-frequency / multiscale problems

Damped harmonic oscillator

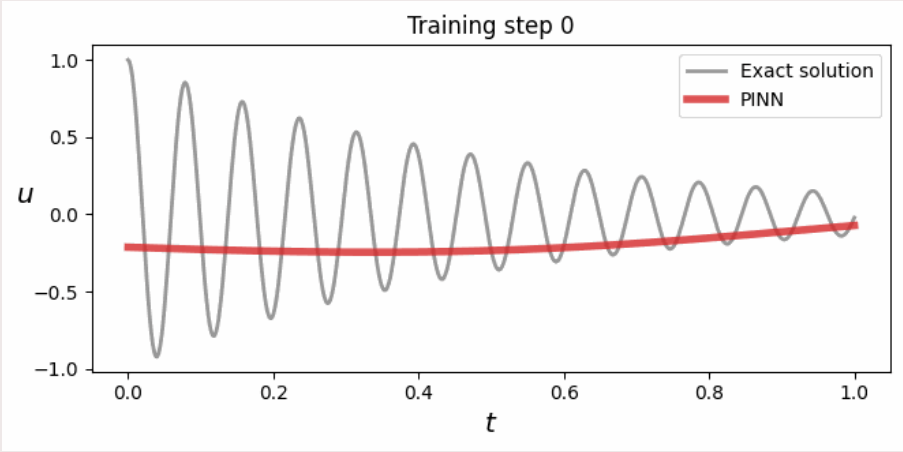


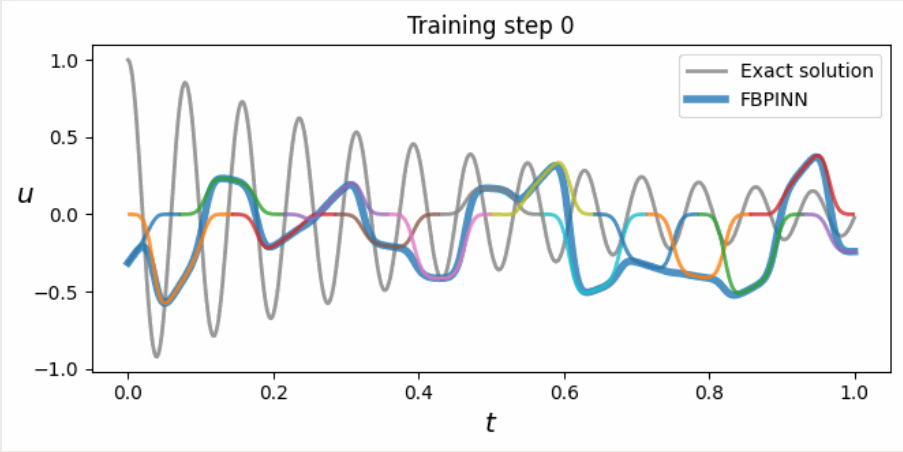
FBPINN solution

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



FBPINNs vs PINNs



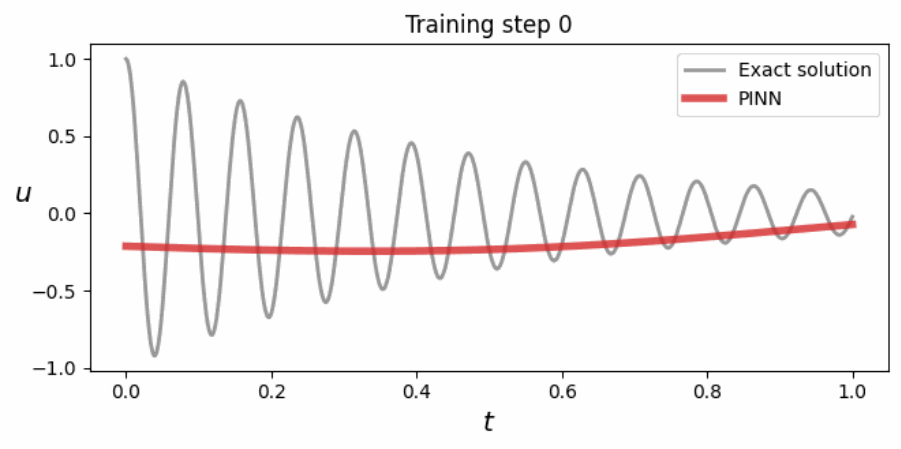


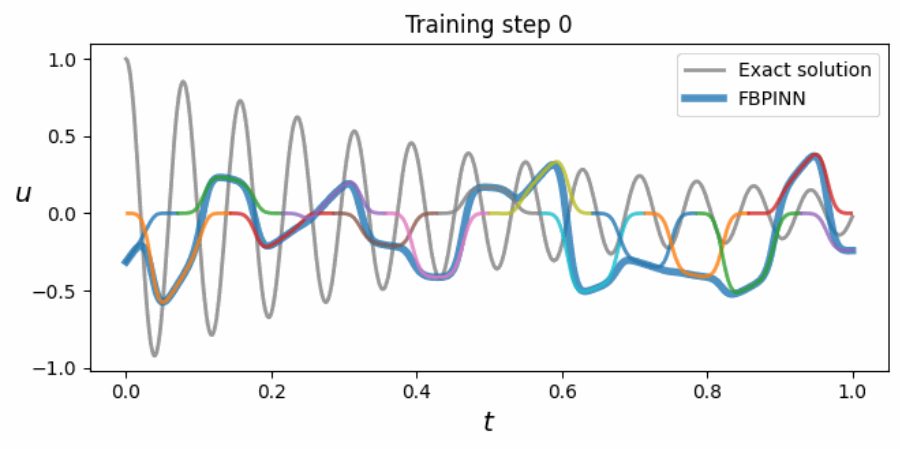
As higher frequencies are added:

- More collocation points required ($\propto \omega^d$)
- Larger neural network required ($\propto f(\omega)$)
- Spectral bias slows convergence ($\propto s(\omega)$)

$\Rightarrow$ Empirically, cost of training often $\sim \mathcal{O}(\omega^d f(\omega) s(\omega))$

FBPINNs vs PINNs





As higher frequencies are added:

- More collocation points required ($\propto \omega^d$)
- Larger neural network required ($\propto f(\omega)$)
- Spectral bias slows convergence ($\propto s(\omega)$)

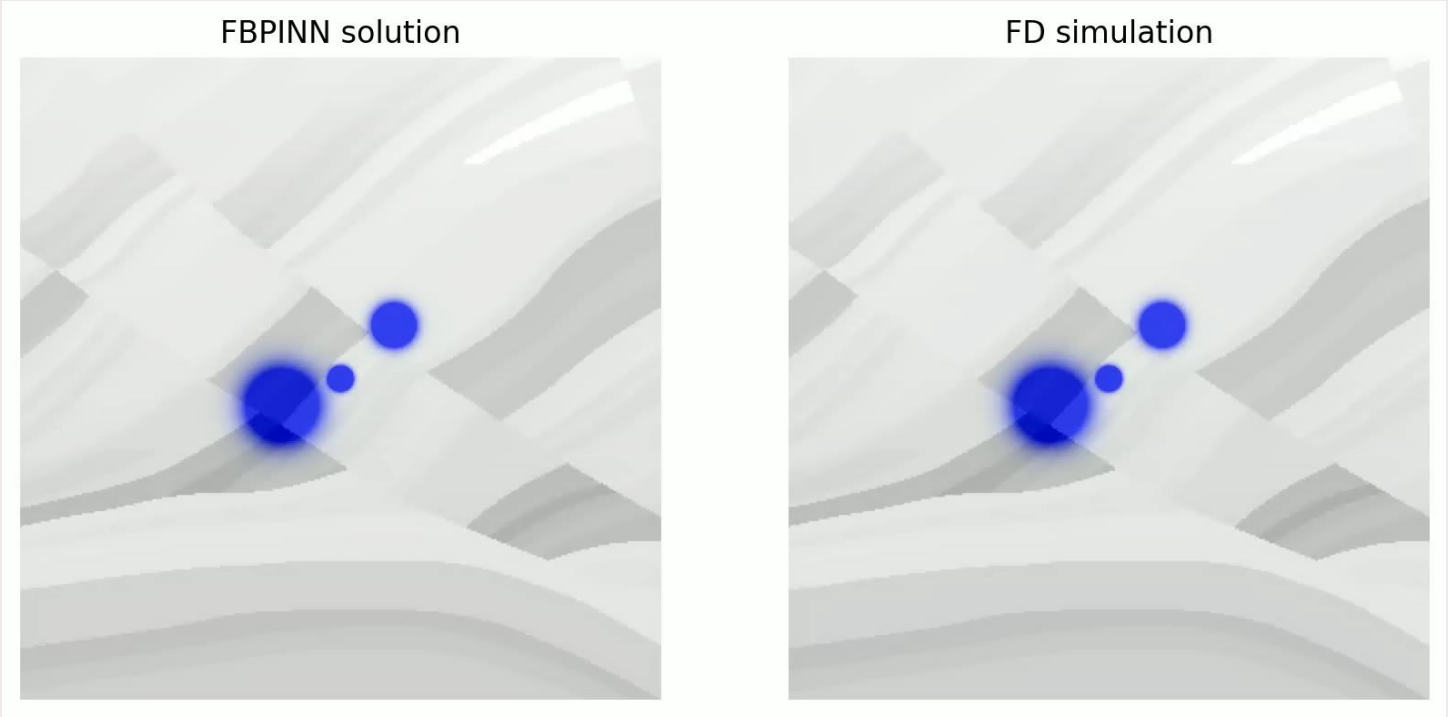
$\Rightarrow$ Empirically, cost of training often $\sim \mathcal{O}(\omega^d f(\omega) s(\omega))$

As higher frequencies are added:

- More collocation points required ($\propto \omega^d$)
- Same size network can be used in each subdomain
- Domain decomposition alleviates spectral bias

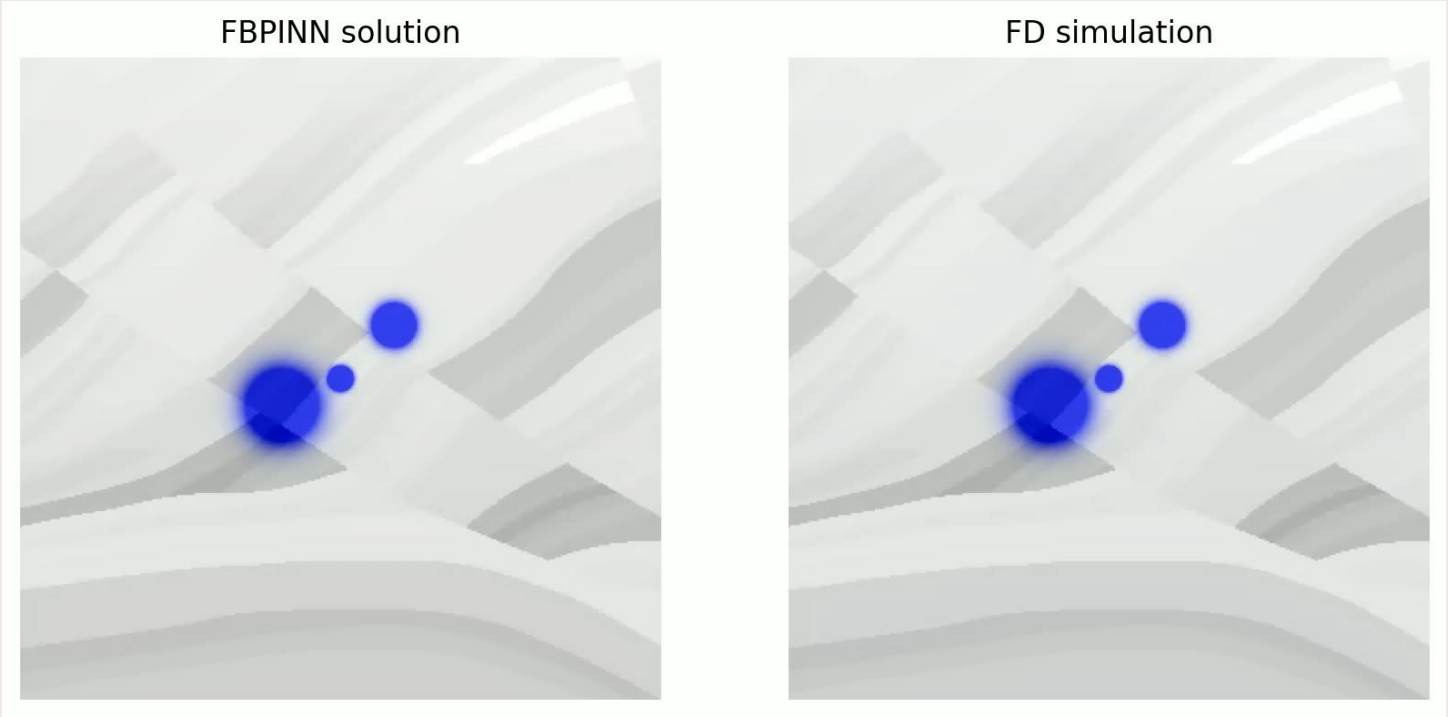
$\Rightarrow$ Empirically, cost of training can be closer to $\sim \mathcal{O}(\omega^d)$

Multi-scale simulation with FBPINNs



Number of subdomains: 60 x 60 x 60 = 216,000
Total number of trainable parameters: 9 M

Multi-scale simulation with FBPINNs

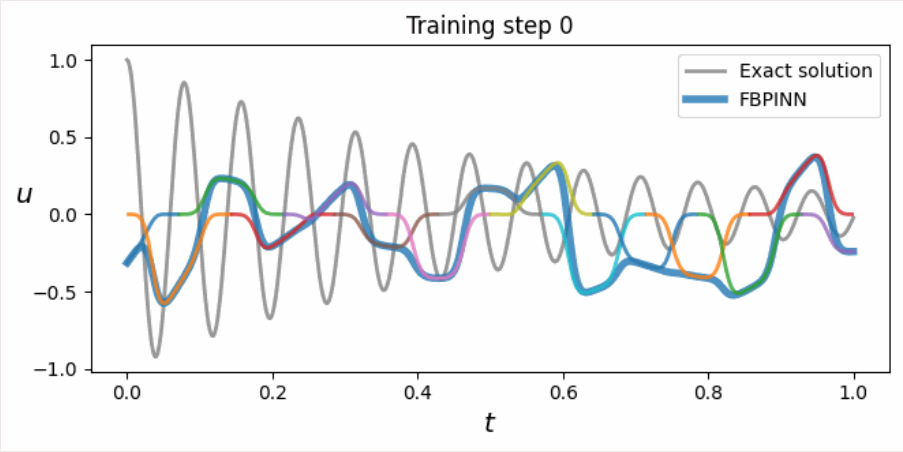


Training time: ~**2 hrs** on GPU (with optimised code)

Number of subdomains: 60 x 60 x 60 = 216,000
Total number of trainable parameters: 9 M

FD simulation time: ~5 mins on CPU!

Why are FBPINNs still slow?



As higher frequencies are added:

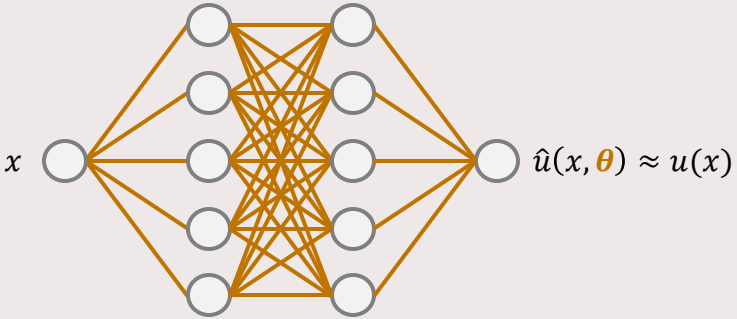
- More collocation points required ($\propto \omega^d$)
- Same size network can be used in each subdomain
- Domain decomposition alleviates spectral bias

$\Rightarrow$ Empirically, cost of training can be closer to $\sim \mathcal{O}(\omega^d)$

BUT gradient descent is a slow optimiser (non-convex loss requires lots of iterations + backprop introduces lots of overhead)

..can we **avoid** gradient descent altogether?

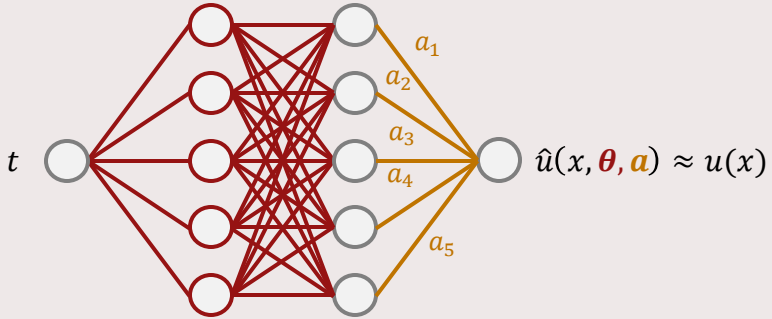
Idea – Extreme learning machines



Neural network

All weights **trainable**

$$\hat{u} = NN(x, \boldsymbol{\theta})$$



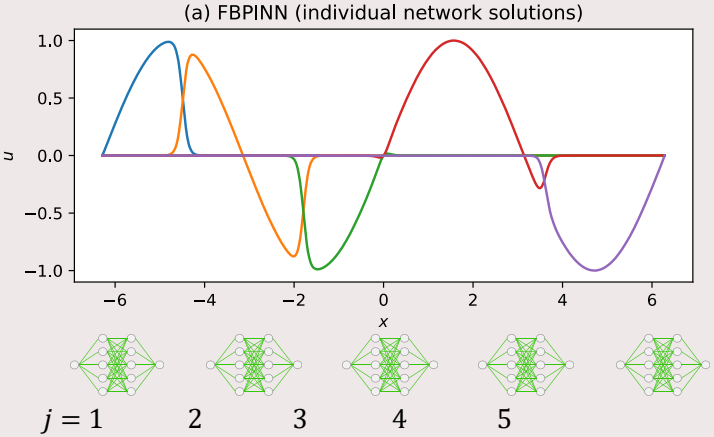
Extreme learning machine

Hidden weights are **randomly** initialised and **fixed**
Only last layer **trainable**

$$\hat{u} = \sum_{k}^{K} a_k \phi(x, \boldsymbol{\theta}_k)$$

Huang, G. Bin, Zhu, Q. Y., & Siew, C. K. Extreme learning machine: Theory and applications. Neurocomputing. (2006).

FBPINN



$$\hat{u}(x, \boldsymbol{\theta}) = \sum_{j}^{J} w_j(x)\, NN_j(x, \boldsymbol{\theta}_j)$$

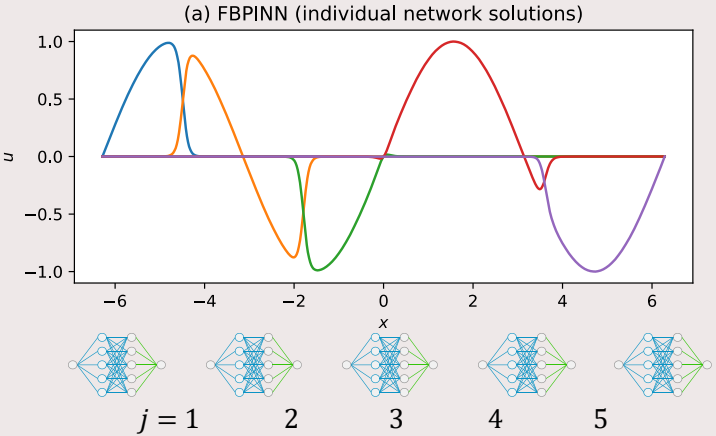
FBPINN

Window function

Subdomain network

(ignoring normalization functions for simplicity)

Extreme learning machine (ELM) FBPINNs



$$\hat{u}(x, \boldsymbol{a}) = \sum_{j}^{J} w_j(x) \sum_{k}^{K} a_{jk} \phi(x, \boldsymbol{\theta}_{jk})$$

ELM-FBPINN

ELM **in each** subdomain

J = total number of subdomains

K = number of basis functions per subdomain

Dwivedi, V., and Srinivasan, B. Physics Informed Extreme Learning Machine (PIELM)–A rapid method for the numerical solution of partial differential equations. Neurocomputing. (2020).

Dong, S., and Li, Z. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. Computer Methods in Applied Mechanics and Engineering. (2021).

Extreme learning machine (ELM) FBPINNs

$$\hat{u}(x, \boldsymbol{a}) = \sum_j^J w_j(x) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk})$$

ELM-FBPINN

ELM **in each** subdomain

J = total number of subdomains
K = number of basis functions per subdomain
N = number of collocation points

$$L = \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \hat{u}(t_i, \boldsymbol{\theta}) \right)^2$$

(ignoring boundary loss for simplicity)

$$= \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \sum_j^J w_j(t_i) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk}) \right)^2$$

$$= \sum_i^N \left(\sum_j^J \sum_k^K a_{jk} \mathcal{N} w_j(t_i) \phi(t_i, \boldsymbol{\theta}_{jk}) \right)^2 = \left\| \begin{pmatrix} \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix} \begin{pmatrix} a_{00} \\ \vdots \\ a_{JK} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\|^2$$

Extreme learning machine (ELM) FBPINNs

$$\hat{u}(x, \boldsymbol{a}) = \sum_j^J w_j(x) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk})$$ ELM-FBPINN

ELM **in each** subdomain

J = total number of subdomains
K = number of basis functions per subdomain
N = number of collocation points

$$L = \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \hat{u}(t_i, \boldsymbol{\theta}) \right)^2$$

ELM-FBPINN

$$= \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \sum_j^J w_j(t_i) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk}) \right)^2$$

$$= \sum_i^N \left(\sum_j^J \sum_k^K a_{jk} \mathcal{N} w_j(t_i) \phi(t_i, \boldsymbol{\theta}_{jk}) \right)^2 = \left\| \begin{pmatrix} \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix} \begin{pmatrix} a_{00} \\ \vdots \\ a_{JK} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\|^2$$

Extreme learning machine (ELM) FBPINNs

$$\hat{u}(x, \boldsymbol{a}) = \sum_j^J w_j(x) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk})$$

ELM-FBPINN

ELM **in each** subdomain

J = total number of subdomains
K = number of basis functions per subdomain
N = number of collocation points

$$L = \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \hat{u}(t_i, \boldsymbol{\theta}) \right)^2$$

$$= \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \sum_j^J w_j(t_i) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk}) \right)^2$$

$$= \sum_i^N \left(\sum_j^J \sum_k^K a_{jk} \mathcal{N} w_j(t_i) \phi(t_i, \boldsymbol{\theta}_{jk}) \right)^2 = \left\| \begin{pmatrix} \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix} \begin{pmatrix} a_{00} \\ \vdots \\ a_{JK} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\|^2$$

Assuming $\mathcal{N}$ is a **linear** operator, $\mathcal{N} = \left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right]$

Extreme learning machine (ELM) FBPINNs

$$\hat{u}(x, \boldsymbol{a}) = \sum_j^J w_j(x) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk})$$

ELM-FBPINN

ELM **in each** subdomain

J = total number of subdomains
K = number of basis functions per subdomain
N = number of collocation points

$$L = \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \hat{u}(t_i, \boldsymbol{\theta}) \right)^2$$

$$= \sum_i^N \left(\left[m \frac{d^2}{dt^2} + \mu \frac{d}{dt} + k \right] \sum_j^J w_j(t_i) \sum_k^K a_{jk} \phi(x, \boldsymbol{\theta}_{jk}) \right)^2$$

$$= \sum_i^N \left(\sum_j^J \sum_k^K a_{jk} \mathcal{N} w_j(t_i) \phi(t_i, \boldsymbol{\theta}_{jk}) \right)^2 = \left\| \begin{pmatrix} \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0) \phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N) \phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix} \begin{pmatrix} a_{00} \\ \vdots \\ a_{JK} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right\|^2$$

$$\equiv \|M\boldsymbol{a} - \boldsymbol{f}\|^2$$

$M: N \times JK \quad \boldsymbol{a}: JK \quad \boldsymbol{f}: N$

Quadratic optimisation

This is a linear least squares problem!

$$L(\boldsymbol{a}) = \|M\boldsymbol{a} - \boldsymbol{f}\|^2$$

The global minima is given by solving

$$A\,\boldsymbol{a}^* = \boldsymbol{b} \quad \text{(normal equation)}$$

Where,

$$A = M^T M \qquad JK \times JK$$
$$\boldsymbol{b} = M^T \boldsymbol{f}$$

Quadratic optimisation

This is a linear least squares problem!

$$L(\boldsymbol{a}) = \|M\boldsymbol{a} - \boldsymbol{f}\|^2$$

The global minima is given by solving

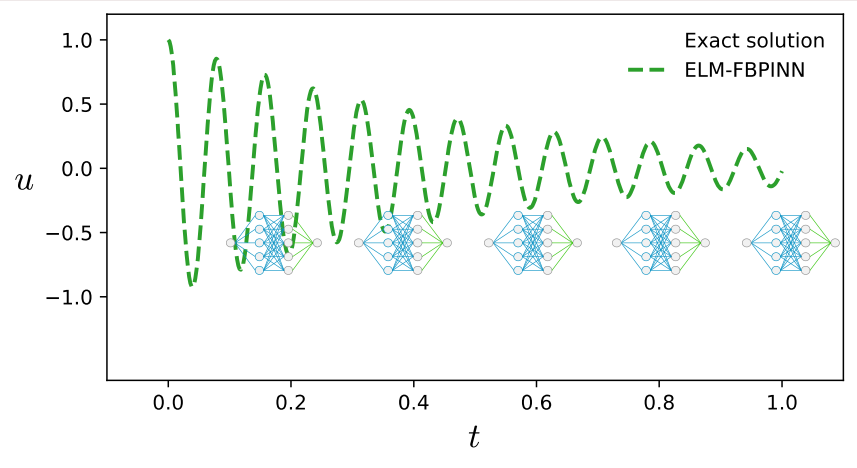
$$A\,\boldsymbol{a}^* = \boldsymbol{b} \qquad \text{(normal equation)}$$

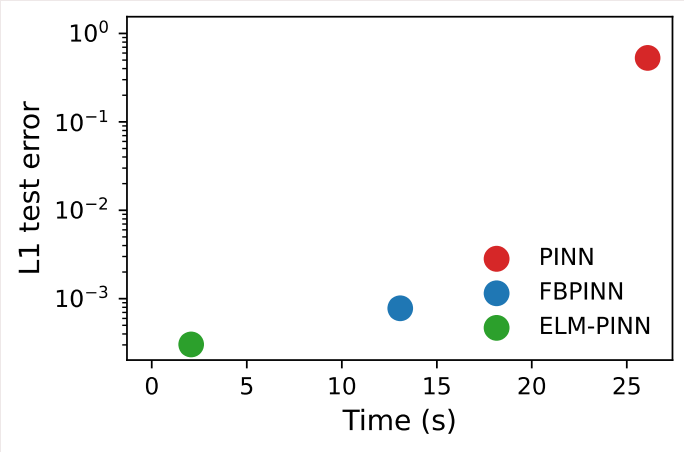
Where,

$$A = M^T M \qquad JK \times JK$$
$$\boldsymbol{b} = M^T \boldsymbol{f} \qquad JK$$

- By using a linear combination of fixed basis functions, we have turned the loss function from non-convex to convex (quadratic)
- I.e., we can now use **linear solvers** to train ELM-FBPINNs, instead of gradient descent! ✅

Example – 1D harmonic oscillator





FBPINN / ELM-FBPINN:
20 subdomains
1 hidden layer, 8 hidden units (=basis functions)
Tanh activation

PINN:
2 hidden layers, 64 hidden units
Tanh activation

PINN / FBPINN: Adam optimiser, 0.001 learning rate
ELM-FBPINN: Conjugate gradient linear solver

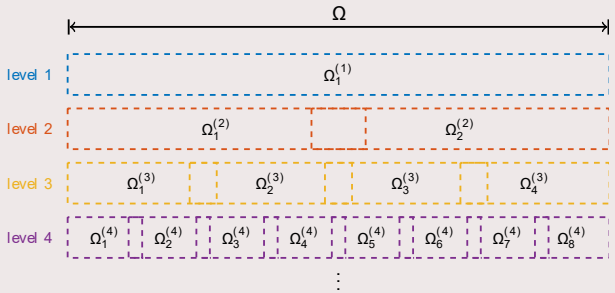
Anderson, S., Dolean, V., Moseley, B., & Pestana, J. ELM-FBPINN: efficient finite-basis physics-informed neural networks. ArXiv. (2024).

Example – 2D multi-scale Laplace

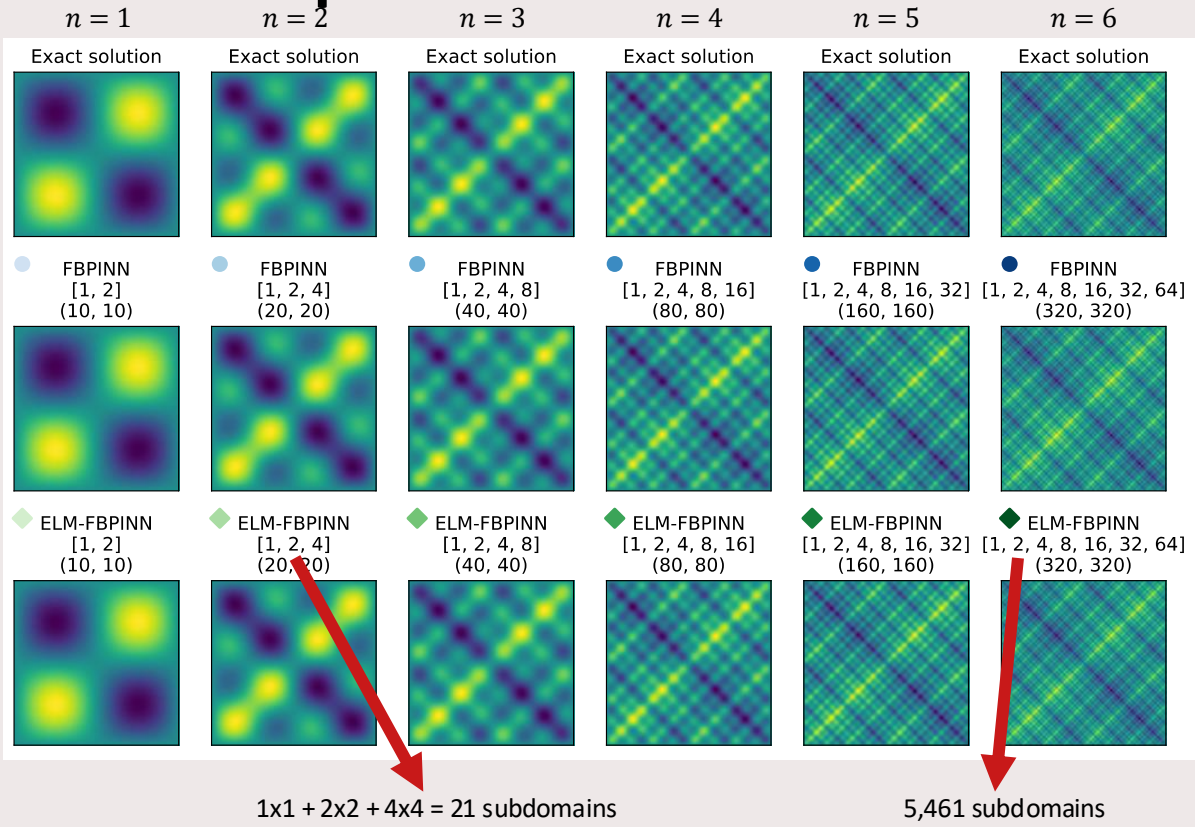
Multi-scale problem:

$$\nabla^2 u(x_1, x_2) = -\frac{2}{n}\sum_{i}^{n}\left(2^i\pi\right)^2 \sin\left(2^i\pi x_1\right)\sin\left(2^i\pi x_2\right)$$

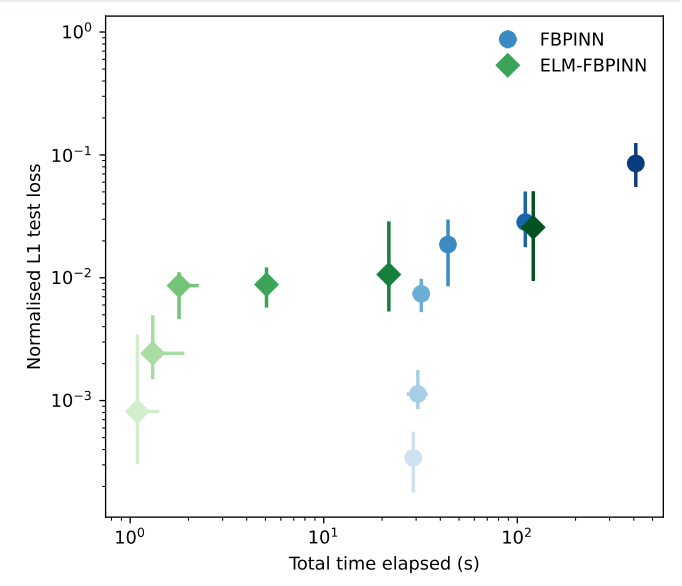
Multilevel domain decomposition:

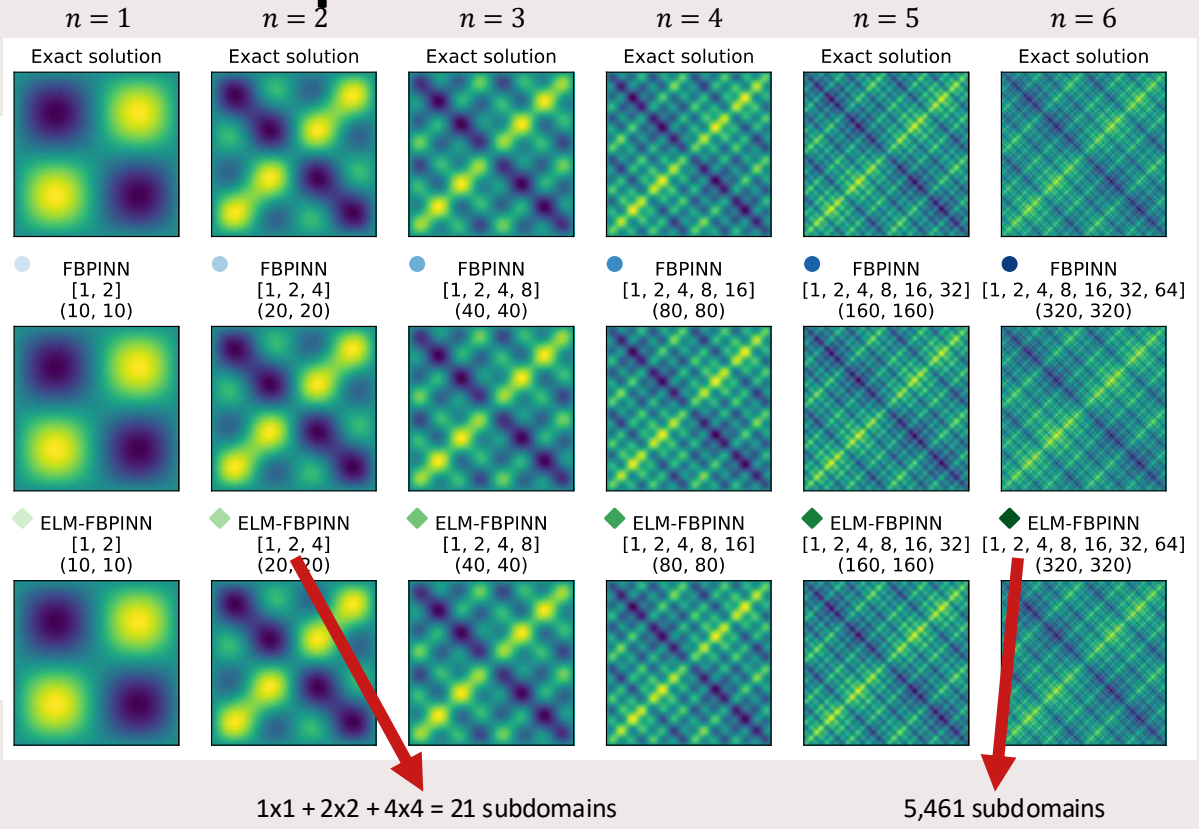


Dolean, V., et al, Multilevel domain decomposition-based architectures for physics-informed neural networks, CMAME (2024)



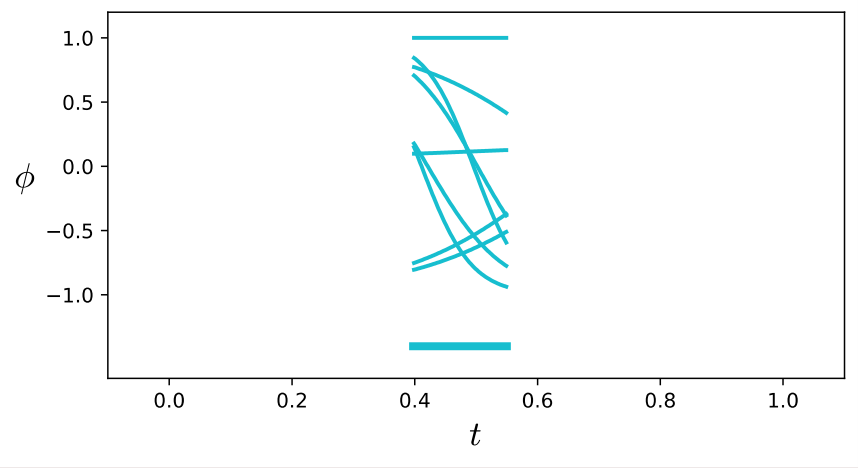
Example – 2D multi-scale Laplace

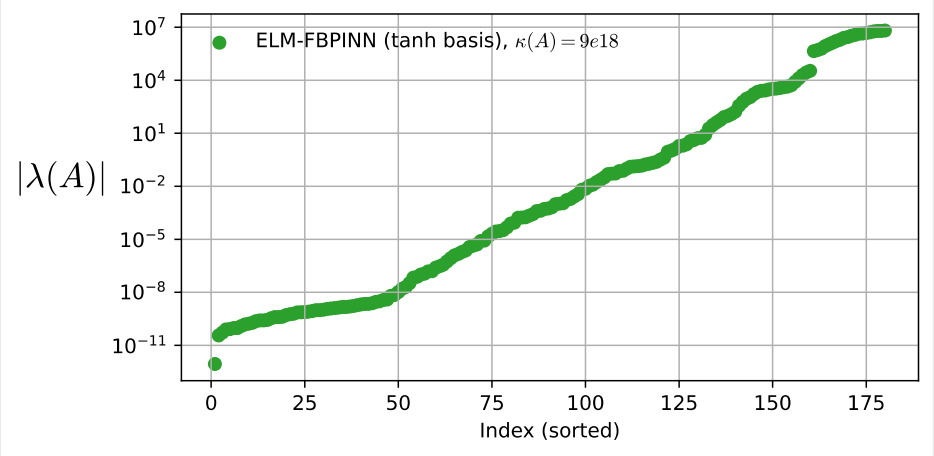




Challenges with ELM-FBPINNs

Challenge 1: Linear dependence between basis functions $\Longrightarrow$ poorly conditioned matrix $A\,\boldsymbol{a}^* = \boldsymbol{b}$





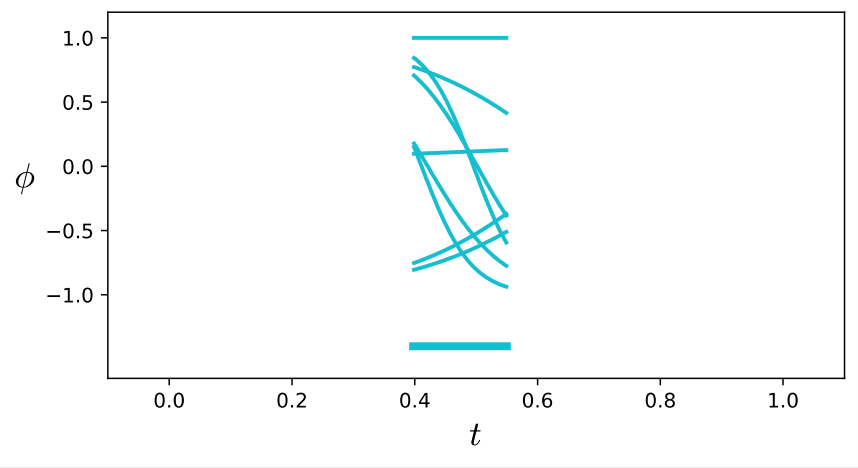
Conjugate gradient solver requires ~5000 iterations

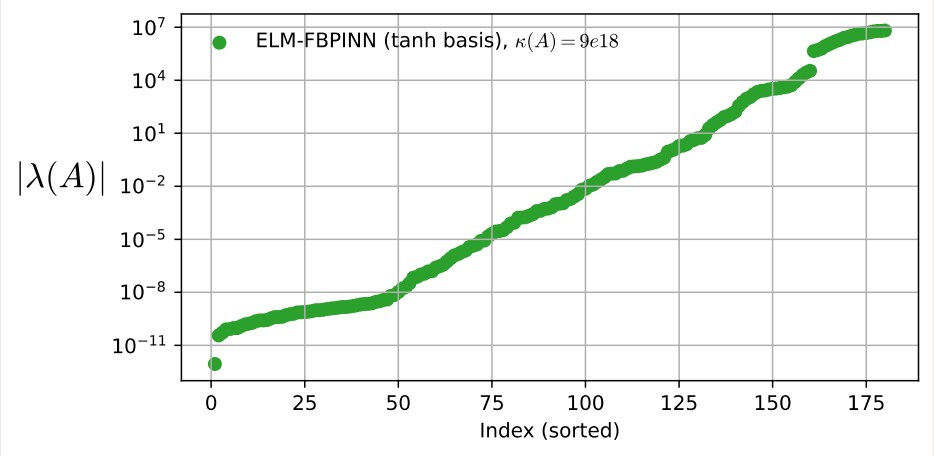
$$M = \begin{pmatrix} \mathcal{N} w_j(t_0)\phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0)\phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N)\phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N)\phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix},$$

$$A = M^T M$$

Challenges with ELM-FBPINNs

Challenge 1: Linear dependence between basis functions $\Longrightarrow$ poorly conditioned matrix $A\,\boldsymbol{a}^* = \boldsymbol{b}$





Possible solution: use **principal component analysis / preconditioning**, see:

Shang, Y., Heinlein, A., Mishra, S., & Wang, F. Overlapping Schwarz Preconditioners for Randomized Neural Networks with Domain Decomposition. ArXiv. (2024).

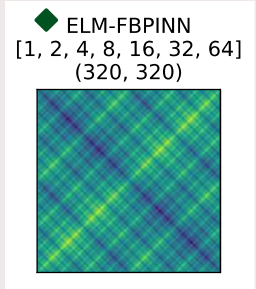
$$M = \begin{pmatrix} \mathcal{N} w_j(t_0)\phi(t_0, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_0)\phi(t_0, \boldsymbol{\theta}_{JK}) \\ \vdots & \ddots & \vdots \\ \mathcal{N} w_j(t_N)\phi(t_N, \boldsymbol{\theta}_{00}) & \dots & \mathcal{N} w_j(t_N)\phi(t_N, \boldsymbol{\theta}_{JK}) \end{pmatrix},$$

$$A = M^T M$$

Challenges with ELM-FBPINNs

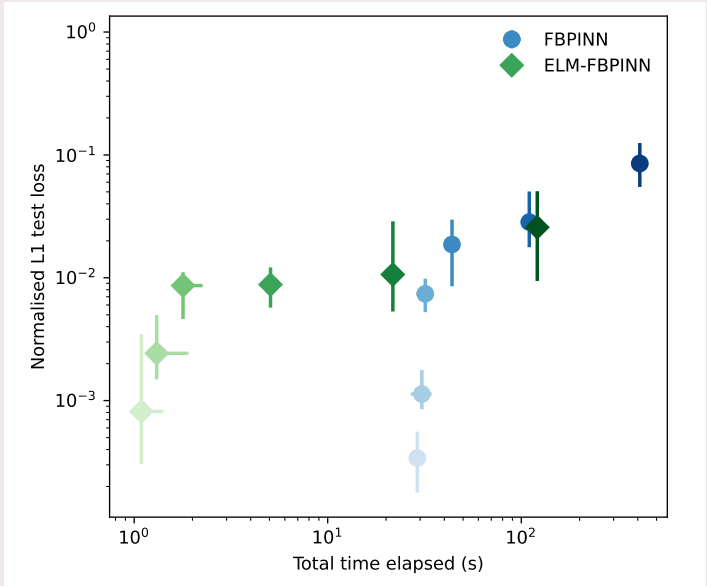
Challenge 2: **Big matrix!** $A\,\boldsymbol{a}^* = \boldsymbol{b}$

$A: JK \times JK \quad \boldsymbol{b}: JK$



$J = 5{,}461, \qquad K = 6$

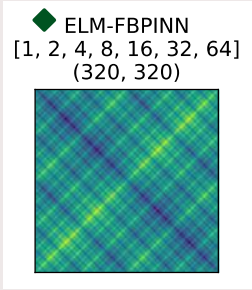
$A: 32{,}766 \times 32{,}766 =$ **1 billion elements!**



Challenges with ELM-FBPINNs

Challenge 2: **Big matrix!** $A\,\boldsymbol{a}^* = \boldsymbol{b}$

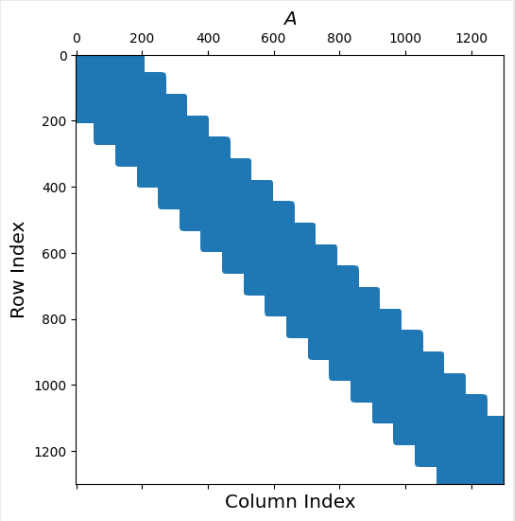
$A: JK \times JK \quad \boldsymbol{b}: JK$



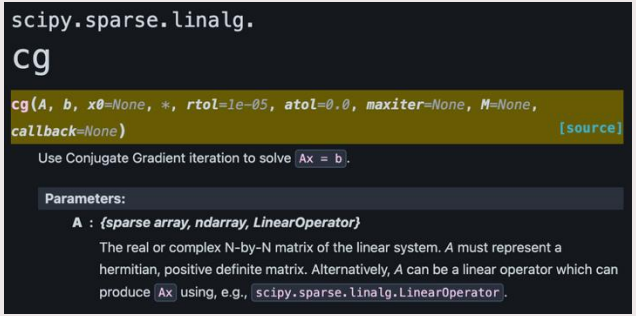
$J = 5{,}461, \qquad K = 6$

$A: 32{,}766 \times 32{,}766 =$ **1 billion elements!**

Solution: exploit **sparsity**



Use sparse solver which only uses matvec products



Can physics-informed neural networks (PINNs) beat finite difference / finite element methods?

Can physics-informed neural networks beat the finite element method?

Tamara G Grossmann ✉, Urszula Julia Komorowska, Jonas Latz, Carola-Bibiane Schönlieb

IMA Journal of Applied Mathematics, Volume 89, Issue 1, January 2024, Pages 143–174, https://doi.org/10.1093/imamat/hxae011
Published: 23 May 2024 **Article history** ▾

7. Discussion and conclusions

After having investigated each of the PDEs on its own, let us now discuss and draw conclusions from the results as a whole. Considering the solution time and accuracy, PINNs are not able to beat FEM in our study. In all the examples that we have studied, the FEM solutions were faster at the same or at a higher accuracy.

Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks

Petr Karnakov, Sergey Litvinov, Petros Koumoutsakos ✉ Author Notes

PNAS Nexus, Volume 3, Issue 1, January 2024, pgae005, https://doi.org/10.1093/pnasnexus/pgae005
Published: 11 January 2024 **Article history** ▾

Conclusion

We introduce the ODIL framework for solving inverse problems for PDEs by casting their discretization as an optimization problem and applying optimization techniques that are widely available in machine-learning software. The concept of casting the PDE as is closely related to the neural network formulations proposed by (15–17) and recently revived as PINNs. However, the fact that we use the discrete approximation of the equations allows for ODIL to be orders of magnitude more efficient in terms of computational cost and accuracy compared to the PINN for which complex flow problems “remain elusive” (71).

Are PINNs becoming FEM?

ELM-FBPINN

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = f$$

$$\hat{u}(t, \boldsymbol{a}) = \sum_j^J w_j(t) \sum_k^K a_{jk}\phi(t, \boldsymbol{\theta}_{jk})$$

$$L(\boldsymbol{a}) = \sum_i^N \left(\left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k \right] \hat{u}(t_i, \boldsymbol{a}) - f(t_i) \right)^2$$
$$= \|M\boldsymbol{a} - \boldsymbol{f}\|^2$$
$$\Rightarrow A\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{0}$$

$A: JK \times JK$ $\quad \boldsymbol{b}: JK$
(sparse & symmetric)

Finite element method

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + ku = f$$

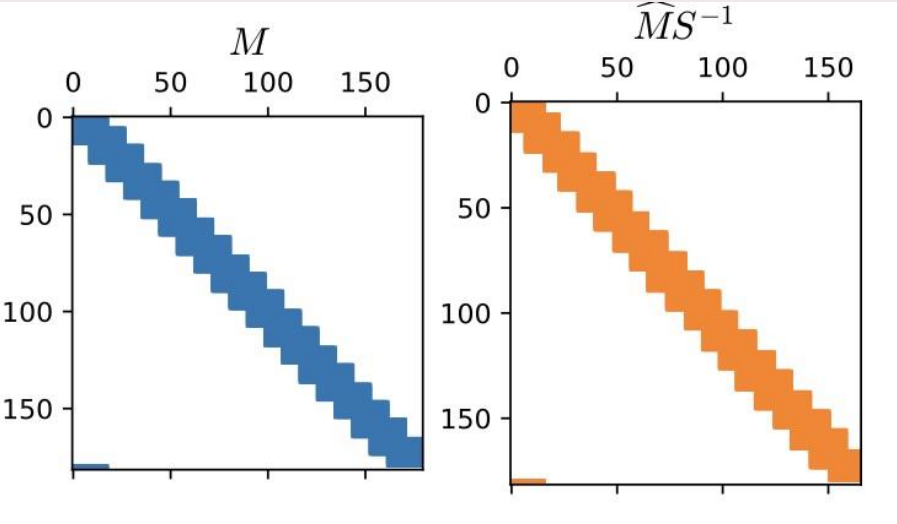
$$\hat{u}(t, \boldsymbol{a}) = \sum_j^J a_j\phi_j(t)$$

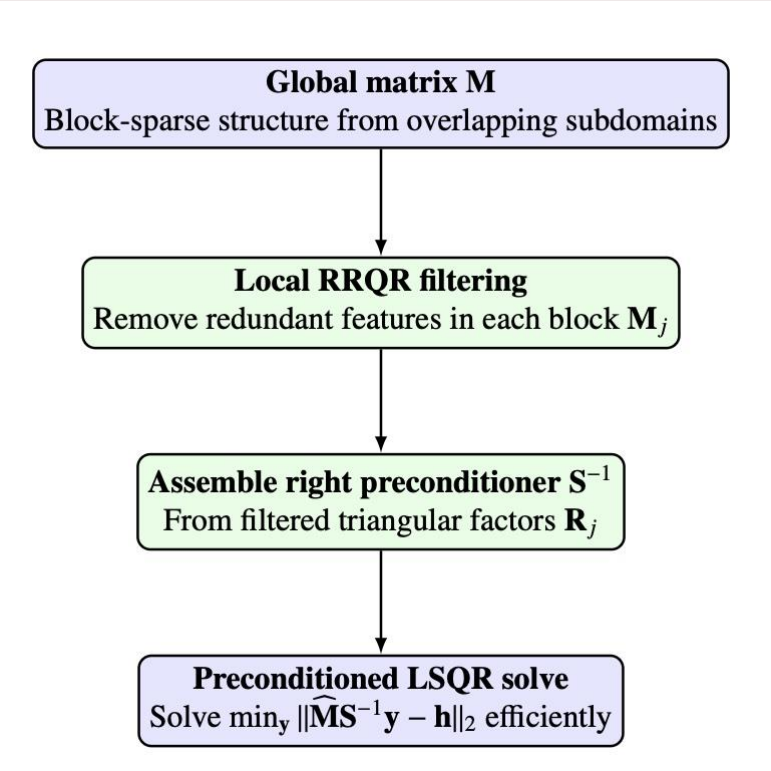
$$\int_0^T \phi_i(t) \left[m\frac{d^2}{dt^2} + \mu\frac{d}{dt} + k \right] \hat{u}(t, \boldsymbol{a}) dt = \int_0^T \phi_i(t)\, f dt \quad \forall i = 0, .., J$$
$$\Rightarrow (-mL + \mu D + kM)\boldsymbol{a} = \boldsymbol{b}$$

$L, D, M: J \times J$ $\quad \boldsymbol{b}: J$
(sparse & symmetric)

Can we do better i.e. how about preconditioning?

Idea: filter the redundant features in each block then precondition directly the least squares problem
Sparsity is preserved.





Local feature filtering for scalable and well-conditioned Random Feature Methods
JW van Beek, V Dolean, B Moseley, CMAME, 2025.

Takeaways

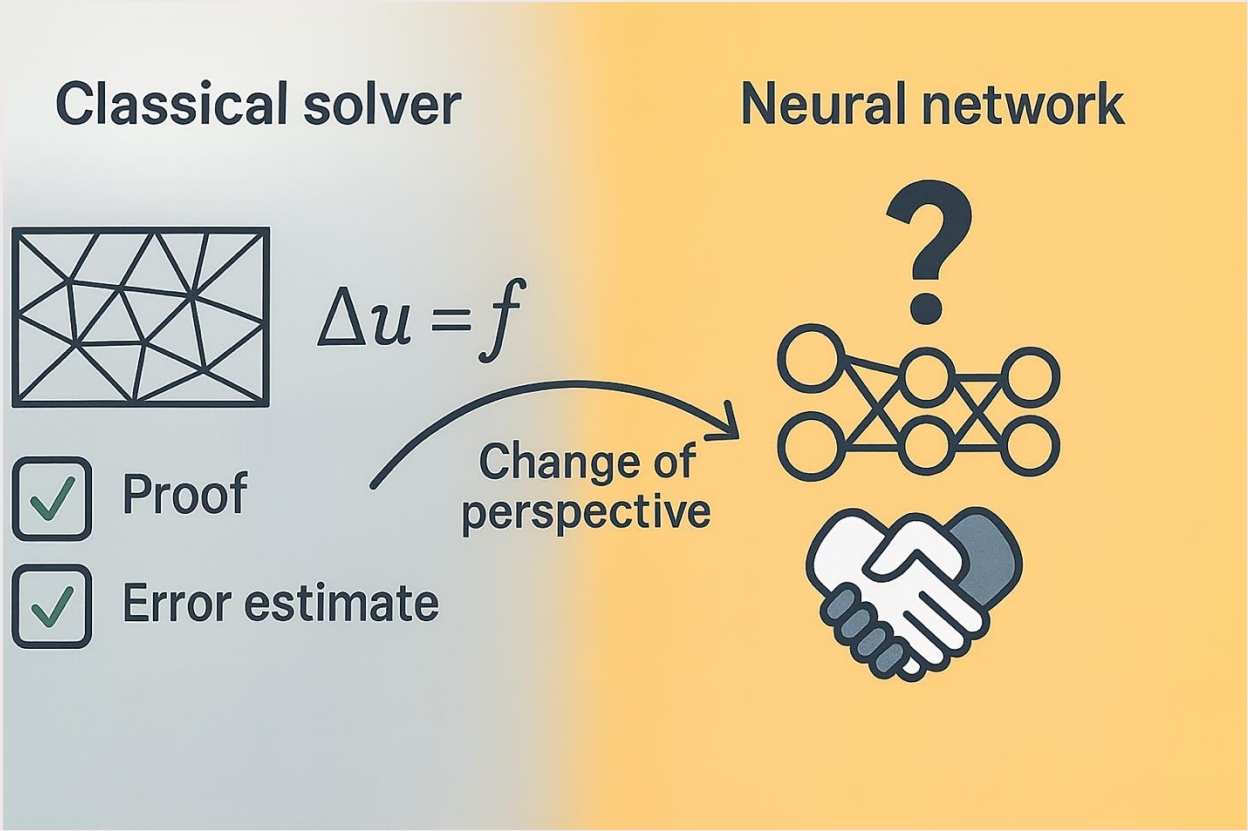
- 💡 **From optimization to linear algebra:** reframing learning as a *numerical linear algebra* problem enables new algorithmic insights and requires new methodologies.

- ⚙️ **Scalability & robustness:** extending to *multilevel* and *overlapping* decompositions could improve performance in large-scale or high-dimensional settings.

- ⚡ **Acceleration & theory:** GPU implementations and convergence-rate analysis are key next steps.

- 🔄 **Beyond linear problems:** the framework naturally extends to nonlinear PDEs via Newton-type iterations.

A change in perspective



Hybrid future

The integration of machine learning (Keplerian paradigm) and more general artificial intelligence technologies with physical modelling based on first principles (Newtonian paradigm) will impact scientific computing in engineering in fundamental ways. (Stefan Kurz, ETH Zurich)