Introduction to Physics-Informed Neural Networks

Dr. Ben Moseley

Mini Lecture @ Imperial College London

March 26th 2024

Lecture overview

- The importance of partial differential equations (PDEs)
- What is a neural network?
- What is a physics-informed neural network?
- How can we train them to solve PDEs?

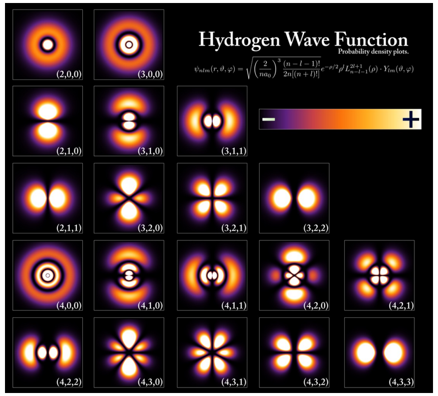
Lecture overview

- The importance of partial differential equations (PDEs)
- What is a neural network?
- What is a physics-informed neural network?
- How can we train them to solve PDEs?

Learning objectives

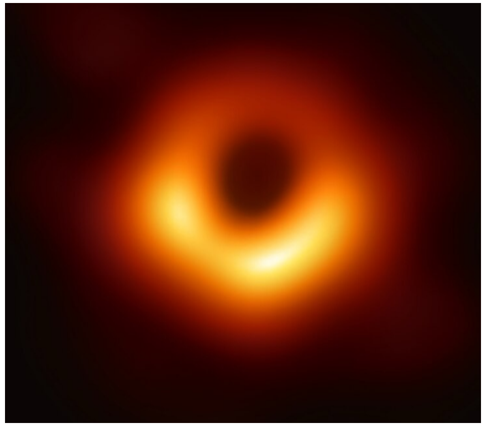
- Explain what a physics-informed neural network is, and how to train it

The importance of partial differential equations



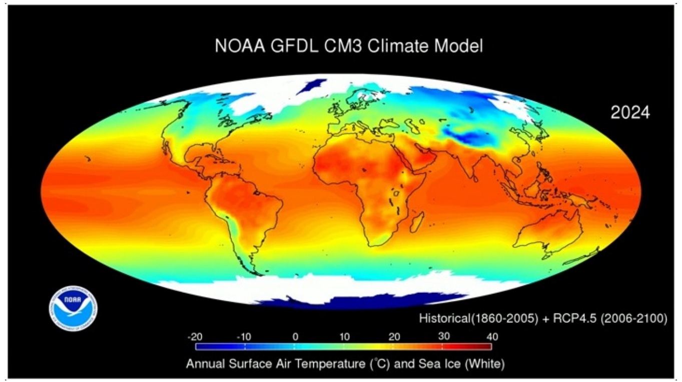
Source: Wikipedia

Schrödinger equation



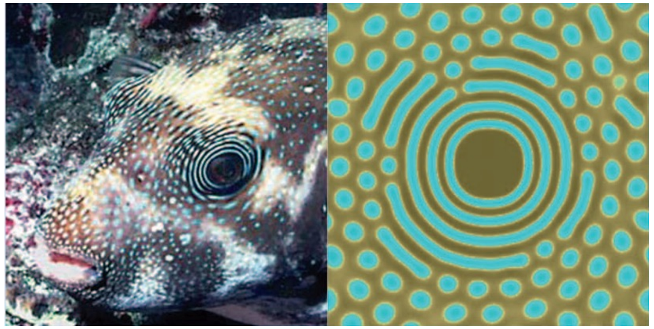
Source: The Event Horizon Telescope (2019)

Einstein field equations



Source: NOAA

Navier-Stokes equations



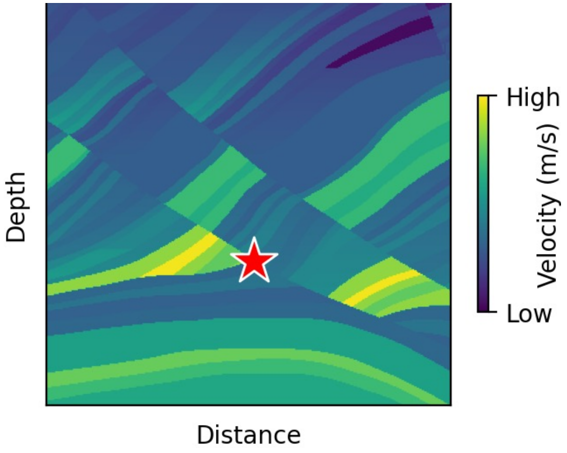
Source: Kondo and Miura, Science (2010)

Reaction-diffusion equation

Solving PDEs

Solving PDEs is:

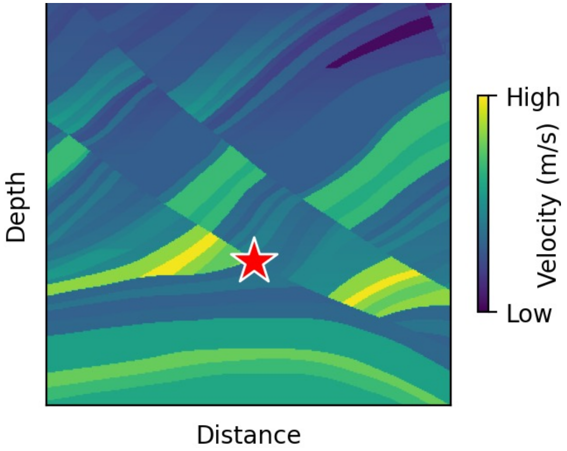
- Crucial for practically all domains of science
- Essential for understanding the behaviour of complex scientific phenomena



Solving PDEs

Solving PDEs is:

- Crucial for practically all domains of science
- Essential for understanding the behaviour of complex scientific phenomena



Wave equation:

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

u = acoustic pressure
c = velocity
s = source function

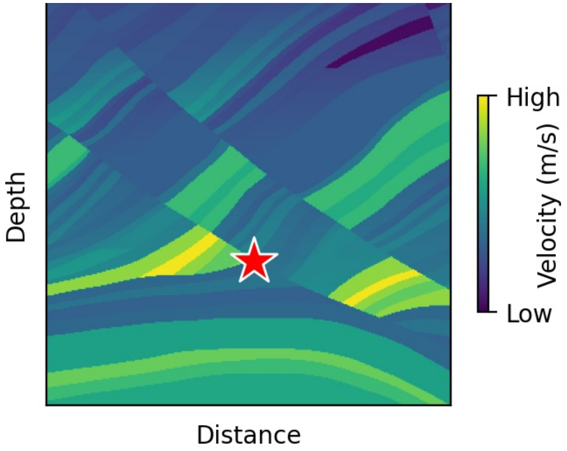
Initial conditions:

$$u(x,y,t=0) = 0$$
$$u_t(x,y,t=0) = 0$$

Solving PDEs

Solving PDEs is:

- Crucial for practically all domains of science
- Essential for understanding the behaviour of complex scientific phenomena



Wave equation:

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

u = acoustic pressure
c = velocity
s = source function

Initial conditions:

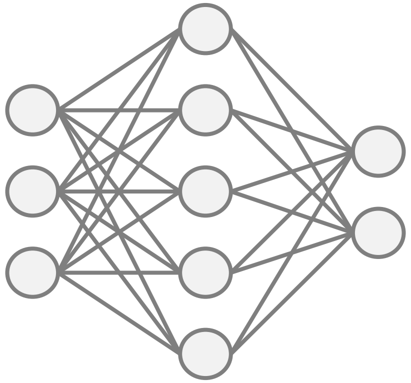
$$u(x,y,t=0) = 0$$
$$u_t(x,y,t=0) = 0$$

Physics-informed neural networks offer a way to **solve** PDEs

What is a neural network?

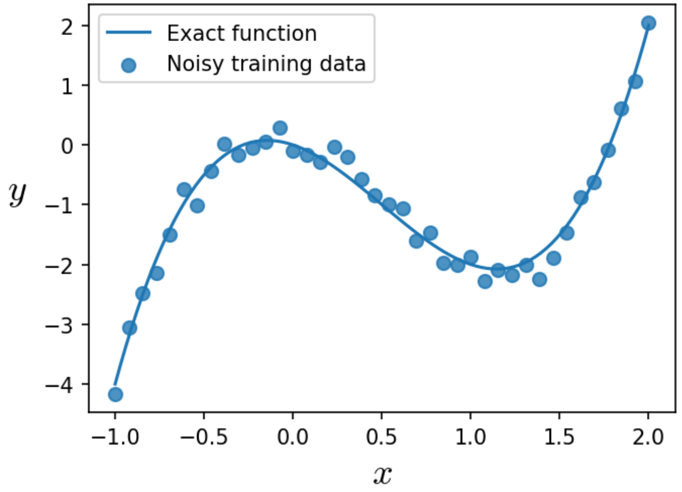
Neural networks are simply **flexible functions** fit to data

$\boldsymbol{x}$



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

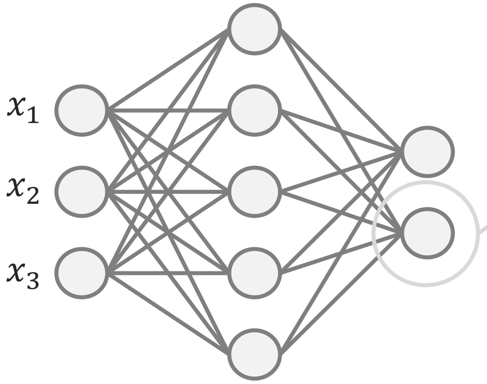
Example data:



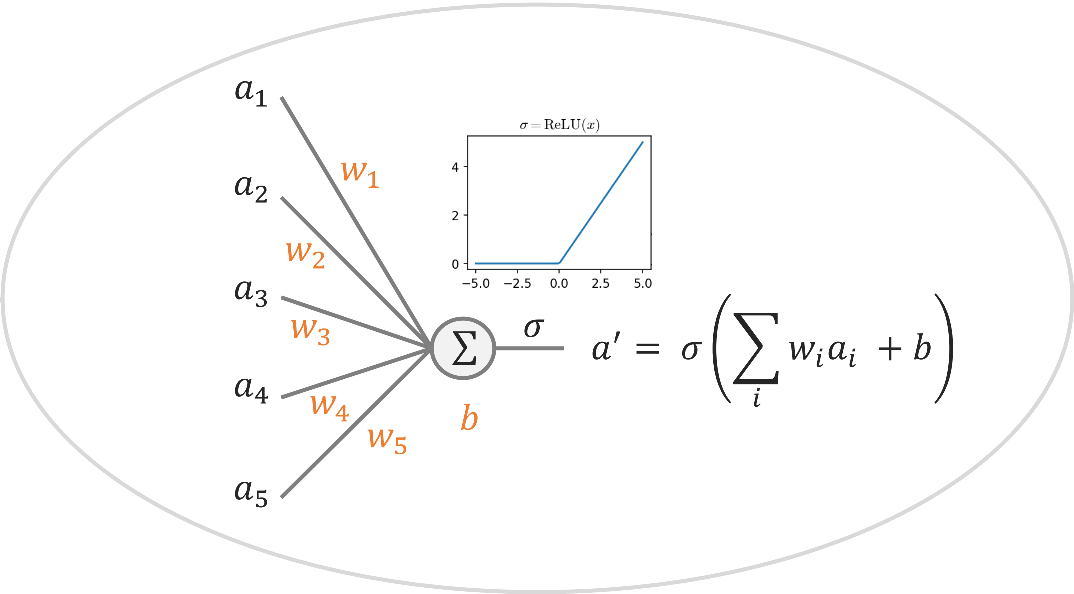
Goal: given training data, tune the parameters $\boldsymbol{\theta}$ so that the network approximates the true function, i.e.,

$$NN(\boldsymbol{x}, \boldsymbol{\theta}) \approx \boldsymbol{y}(\boldsymbol{x})$$

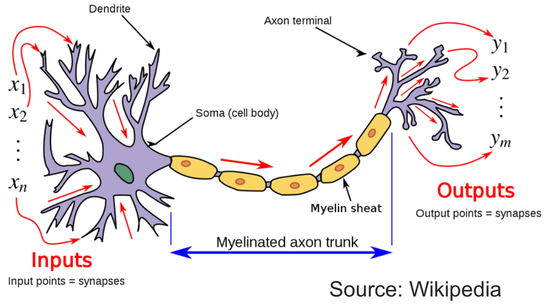
What is a neural network?



2-layer **multi-layer perceptron**



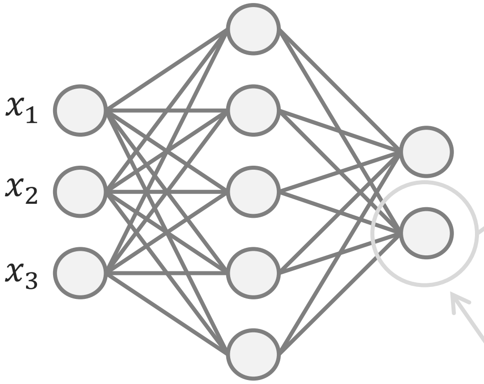
$$a' = \sigma\left(\sum_i w_i a_i + b\right)$$



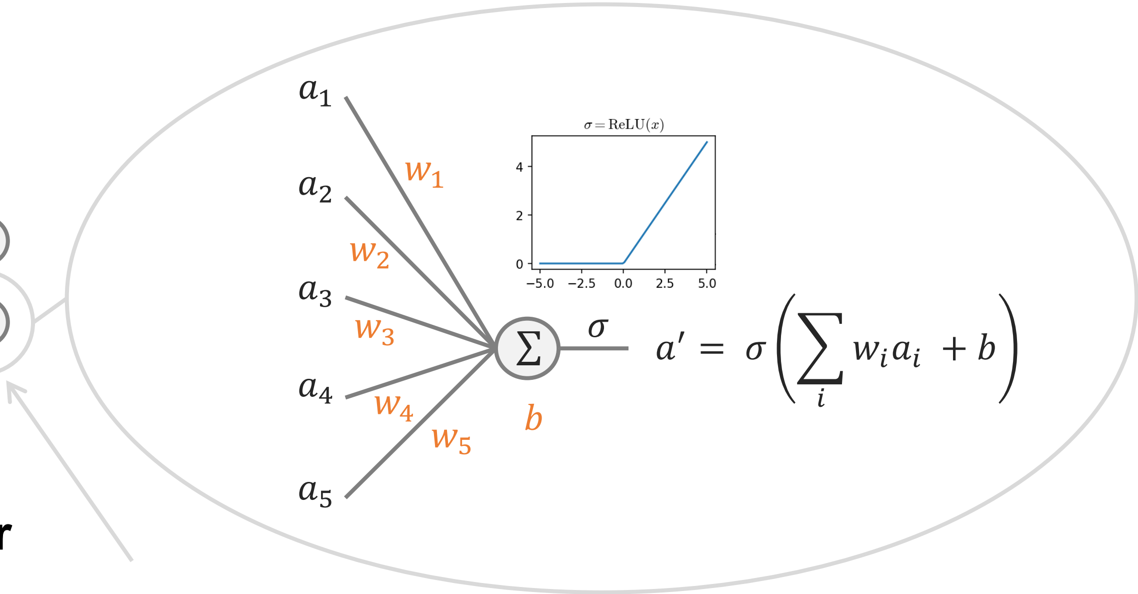
Source: Wikipedia

Biological neuron

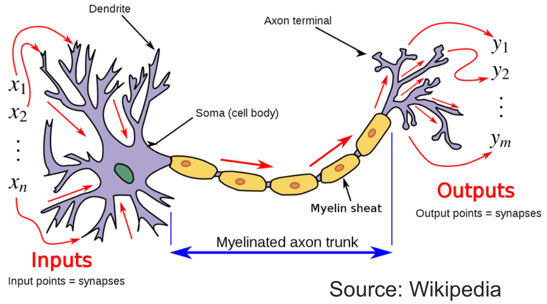
What is a neural network?



2-layer **multi-layer perceptron**



$$a' = \sigma\left(\sum_i w_i a_i + b\right)$$



Source: Wikipedia

Biological neuron

For last layer:

$$\begin{pmatrix} a_1' \\ a_2' \end{pmatrix} = \sigma\left(\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)$$

Entire network:

$$NN(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(W_2\sigma(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

How do we train neural networks?

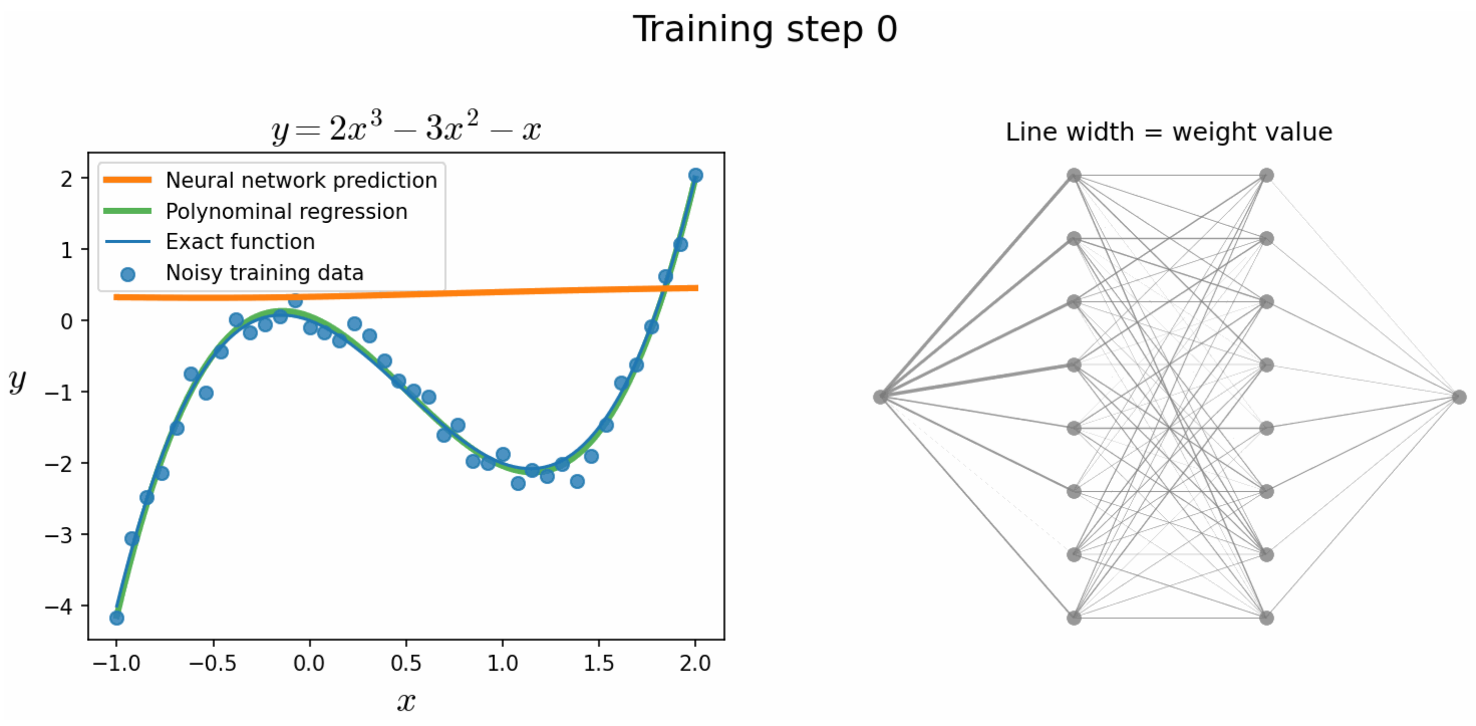
We tune the parameters so that they **minimise** some **loss function**, for example

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i}^{N}(NN(x_i, \boldsymbol{\theta}) - y_i)^2$$

Typically, by using **gradient descent**:

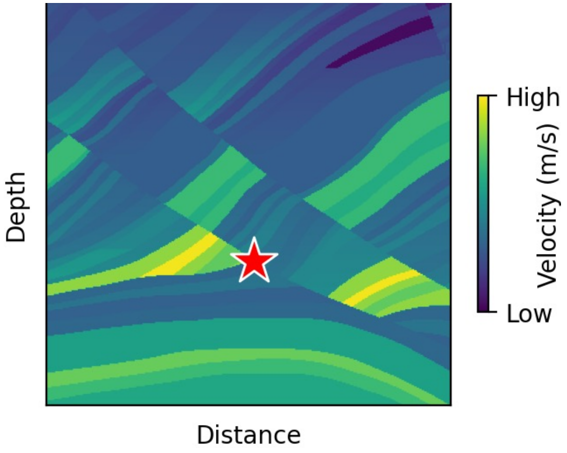
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

γ = step size, e.g. 0.001



Using neural networks for simulation

Q: How could we solve this PDE using neural networks?



Wave equation:

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

u = acoustic pressure
c = velocity
s = source function

Initial conditions:

$$u(x,y,t=0) = 0$$
$$u_t(x,y,t=0) = 0$$

What is a PINN?

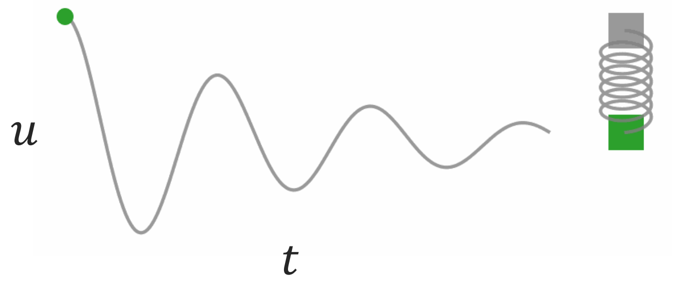
Damped harmonic oscillator:

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

Initial conditions:

$$u(t=0) = 1$$
$$u_t(t=0) = 0$$

$u =$ displacement
$m =$ mass of oscillator
$\mu =$ coefficient of friction
$k =$ spring constant



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 PINN?

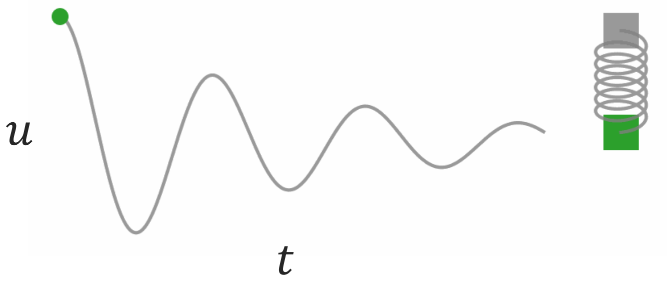
Damped harmonic oscillator:

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

Initial conditions:

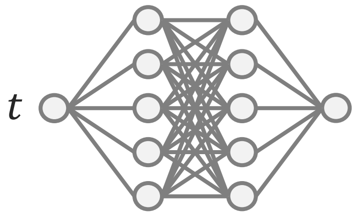
$$u(t=0) = 1$$
$$u_t(t=0) = 0$$

u = displacement
m = mass of oscillator
μ = coefficient of friction
k = spring constant



Key idea: use a neural network to directly approximate the solution

$$NN(t, \boldsymbol{\theta}) \approx u(t)$$



$$NN(t, \boldsymbol{\theta}) \approx u(t)$$

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 PINN?

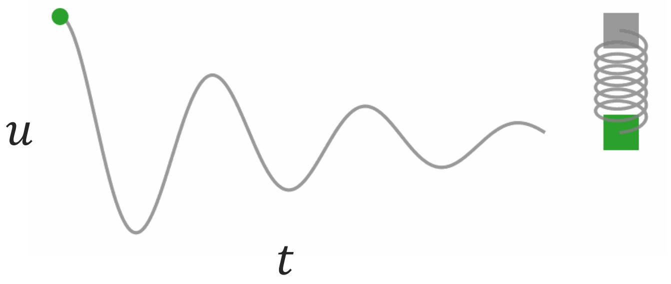
Damped harmonic oscillator:

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

Initial conditions:

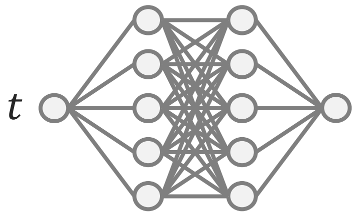
$$u(t=0)=1$$
$$u_t(t=0)=0$$

u = displacement
m = mass of oscillator
μ = coefficient of friction
k = spring constant



Key idea: use a neural network to directly approximate the solution

$$NN(t,\boldsymbol{\theta})\approx u(t)$$



$$NN(t,\boldsymbol{\theta})\approx u(t)$$

Train the network using the loss function:

Boundary loss $L_b(\boldsymbol{\theta})$

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

Physics loss $L_p(\boldsymbol{\theta})$ (aka PDE residual)

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

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)

What is a PINN?

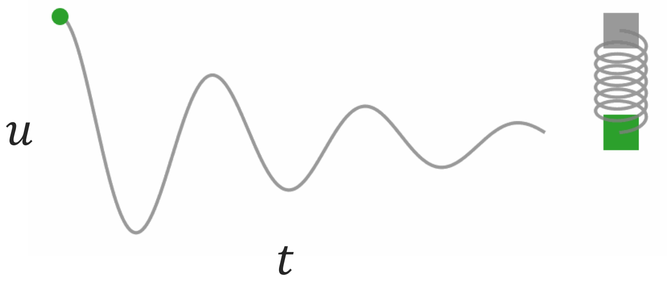
Damped harmonic oscillator:

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

Initial conditions:

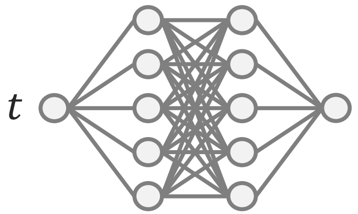
$$u(t=0)=1$$
$$u_t(t=0)=0$$

u = displacement
m = mass of oscillator
μ = coefficient of friction
k = spring constant



Key idea: use a neural network to directly approximate the solution

$$NN(t,\boldsymbol{\theta})\approx u(t)$$



$$NN(t,\boldsymbol{\theta})\approx u(t)$$

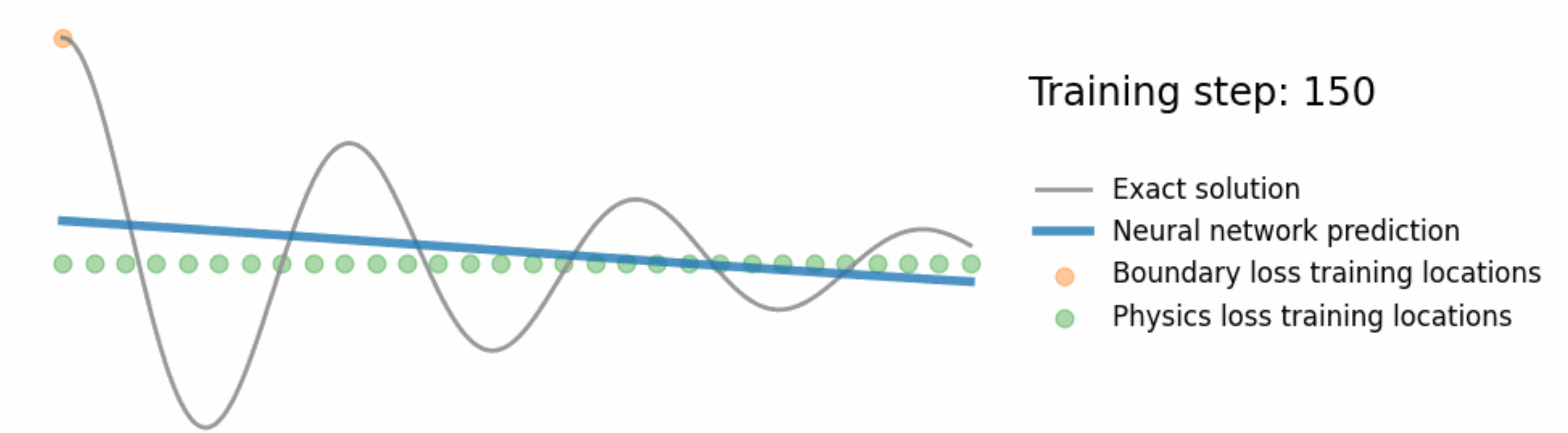
Train the network using the loss function:

Boundary loss $L_b(\boldsymbol{\theta})$

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

Physics loss $L_p(\boldsymbol{\theta})$ (aka PDE residual)

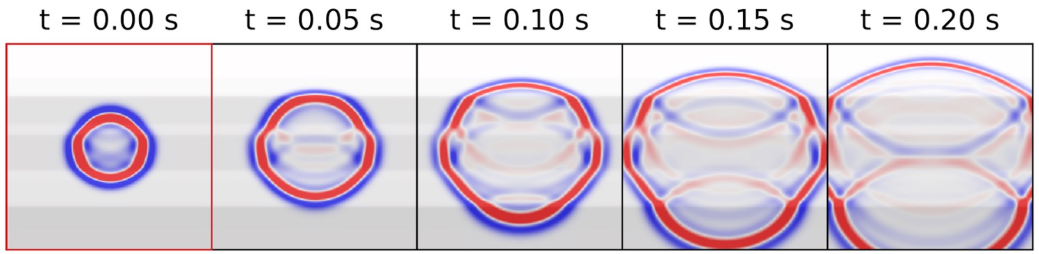
$$+\frac{1}{N_p}\sum_i^{N_p}\left(\left[m\frac{d^2}{dt^2}+\mu\frac{d}{dt}+k\right]NN(\underline{t_i},\boldsymbol{\theta})\right)^2$$



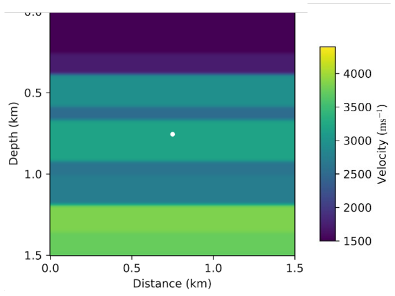
Raissi et al, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, JCP (2018)
Lagaris et al, Artificial neural networks for solving ordinary and partial differential equations, IEEE (1998)

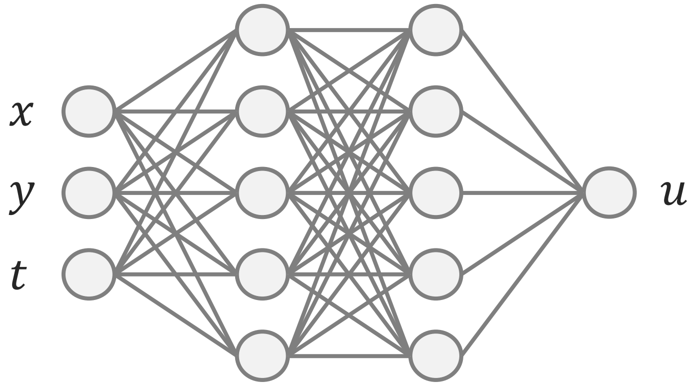
PINNs for solving wave equation

Ground truth FD simulation



Velocity model, $c(x, y)$





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

$$L_b(\boldsymbol{\theta}) = \frac{\lambda}{N_b} \sum_{j}^{N_b} \left(NN(x_j, y_j, t_j, \boldsymbol{\theta}) - \underline{u_{FD}(x_j, y_j, t_j)} \right)^2$$

Boundary data from FD simulation (first 0.02 seconds)

$$L_p(\boldsymbol{\theta}) = \frac{1}{N_p} \sum_{i}^{N_p} \left(\left[\nabla^2 - \frac{1}{c(x_i)^2} \frac{\partial^2}{\partial t^2} \right] NN(\underline{x_i, y_i, t_i}, \boldsymbol{\theta}) \right)^2$$

Physics loss training points randomly sampled over entire x-y-t domain (up to 0.2 seconds)

PINNs for solving wave equation

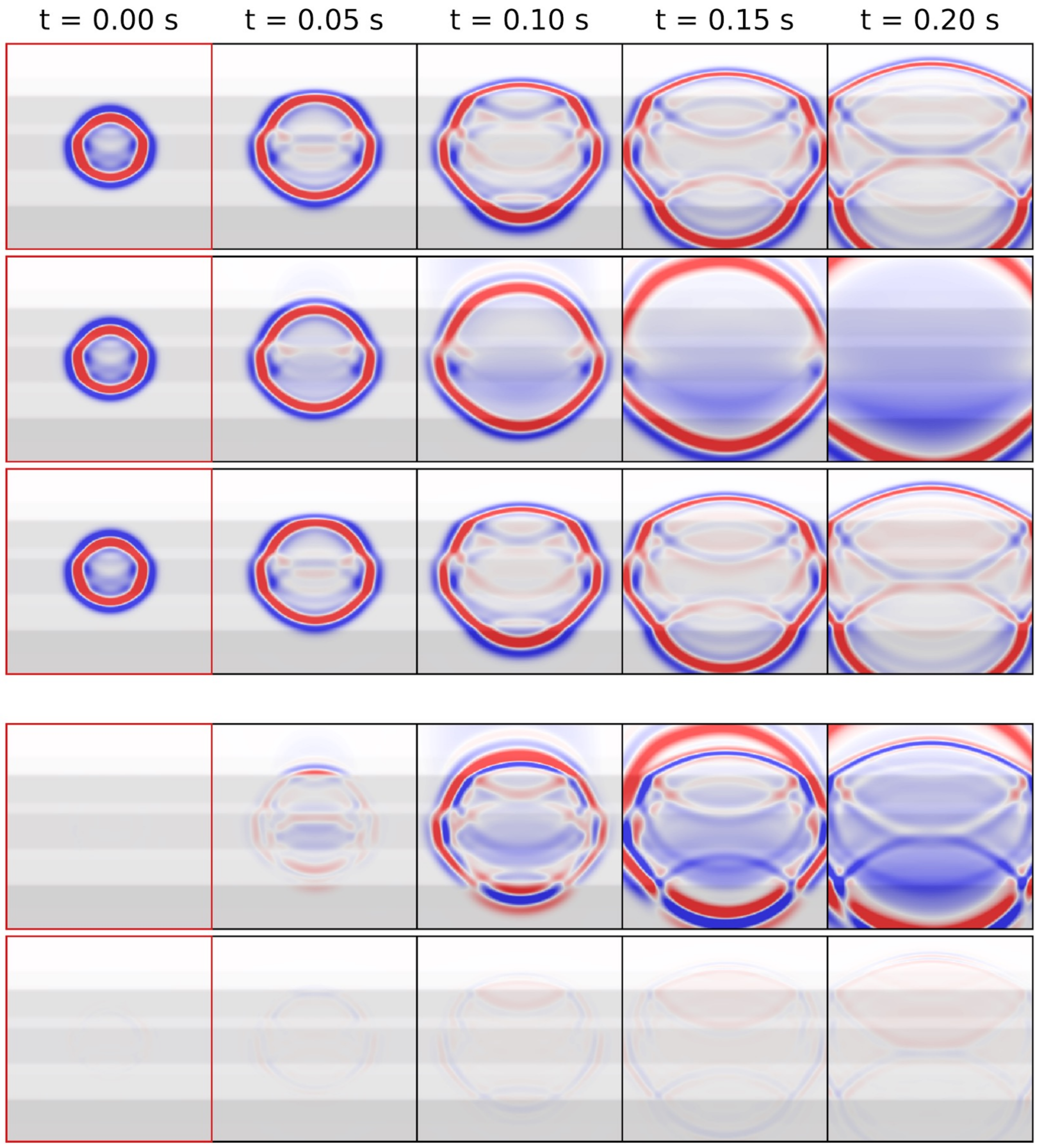
Ground truth FD simulation

"Naïve" NN

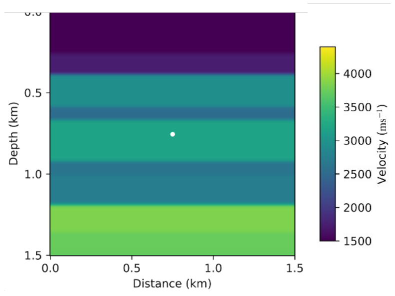
PINN

Difference (NN)

Difference (PINN)



Velocity model, $c(x, y)$



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

Mini-batch size $N_b = N_p = 500$ (random sampling)
Fully connected network with 10 layers, 1024 hidden units
Softplus activation
Adam optimiser

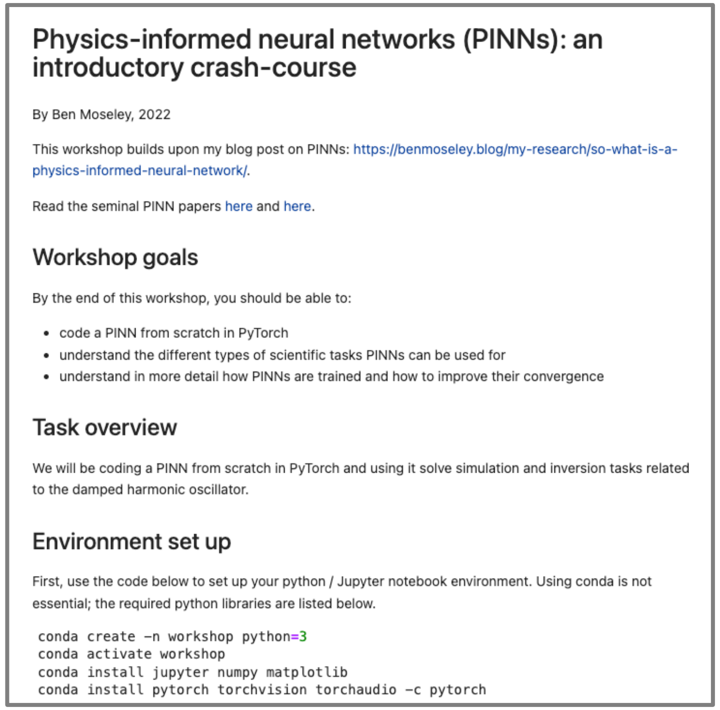
Lecture summary

- PINNs are a method for solving partial differential equations
- They use a neural network to **directly approximate** the solution to the PDE

Lecture **slides** + **homework** coding task (Jupyter notebook) on solving the 1D harmonic oscillator with PINNs:

benmoseley.blog/teaching

SCAN ME



Physics-informed neural networks (PINNs): an introductory crash-course

By Ben Moseley, 2022

This workshop builds upon my blog post on PINNs: https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/.

Read the seminal PINN papers here and here.

Workshop goals

By the end of this workshop, you should be able to:

- code a PINN from scratch in PyTorch
- understand the different types of scientific tasks PINNs can be used for
- understand in more detail how PINNs are trained and how to improve their convergence

Task overview

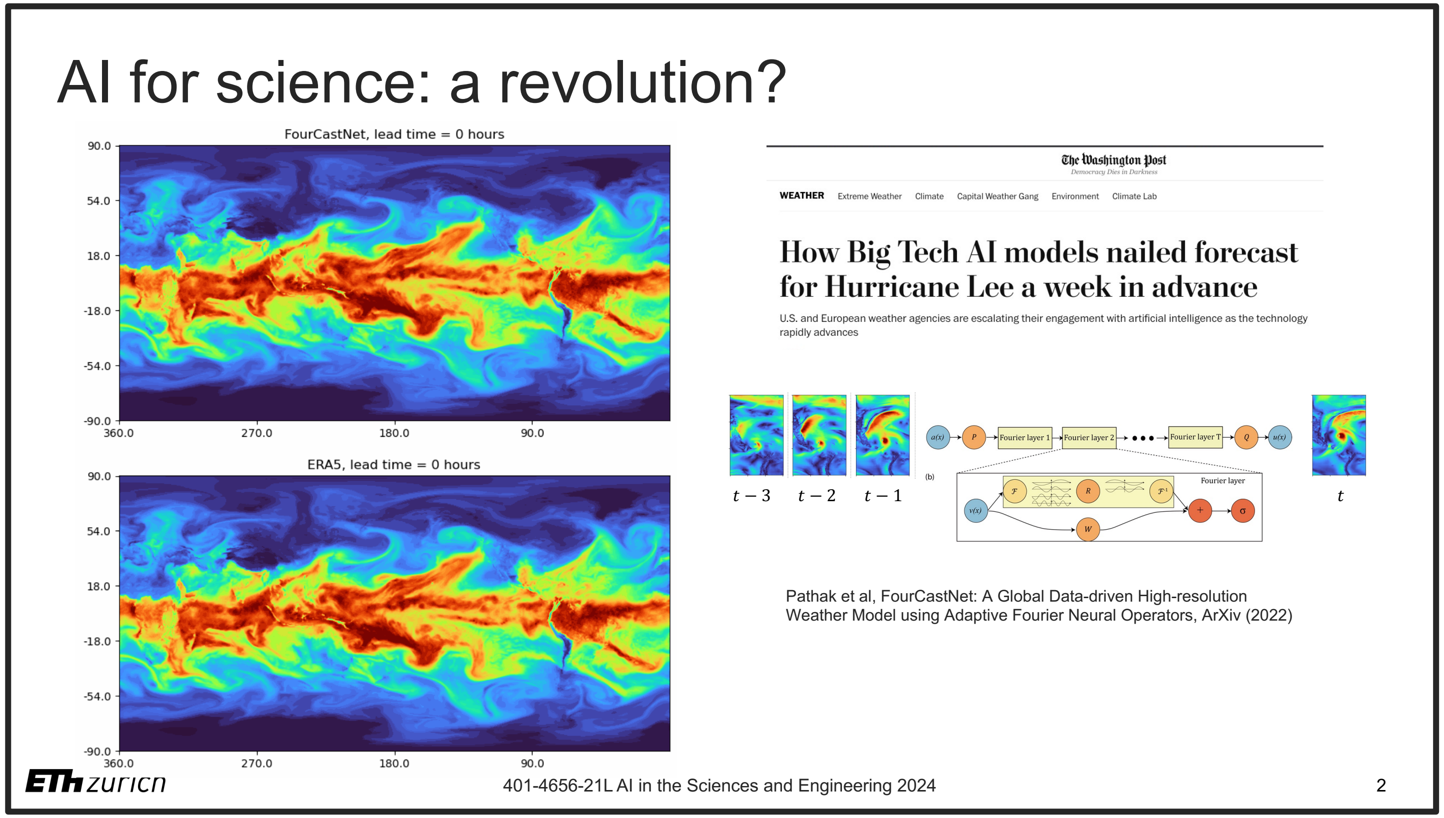
We will be coding a PINN from scratch in PyTorch and using it solve simulation and inversion tasks related to the damped harmonic oscillator.

Environment set up

First, use the code below to set up your python / Jupyter notebook environment. Using conda is not essential; the required python libraries are listed below.

```
conda create -n workshop python=3
conda activate workshop
conda install jupyter numpy matplotlib
conda install pytorch torchvision torchaudio -c pytorch
```

AI in the Sciences and Engineering



ETH zürich

YouTube @CAMLabETHZurich

- Aware of advanced **applications** of AI in science
- Understand key scientific machine learning **concepts** and themes

SCAN ME

51K YouTube views
4.5/5 student feedback score