UNIVERSITY OF IOANNINA
Department of Physics

5th Summer School of Hel.A.S.

Solution of a Simple Physics Problem with Physics Informed Neural Networks (PINNs)

Practical Work Group 1

Alexander Stork
Konstantinos-Xanthos Argyropoulos
Maria Vergi
Pelagia Flotsiou
Stefania Athanasakis

Ioannina, 2024

Outline

1.1 What is a Neural Network?

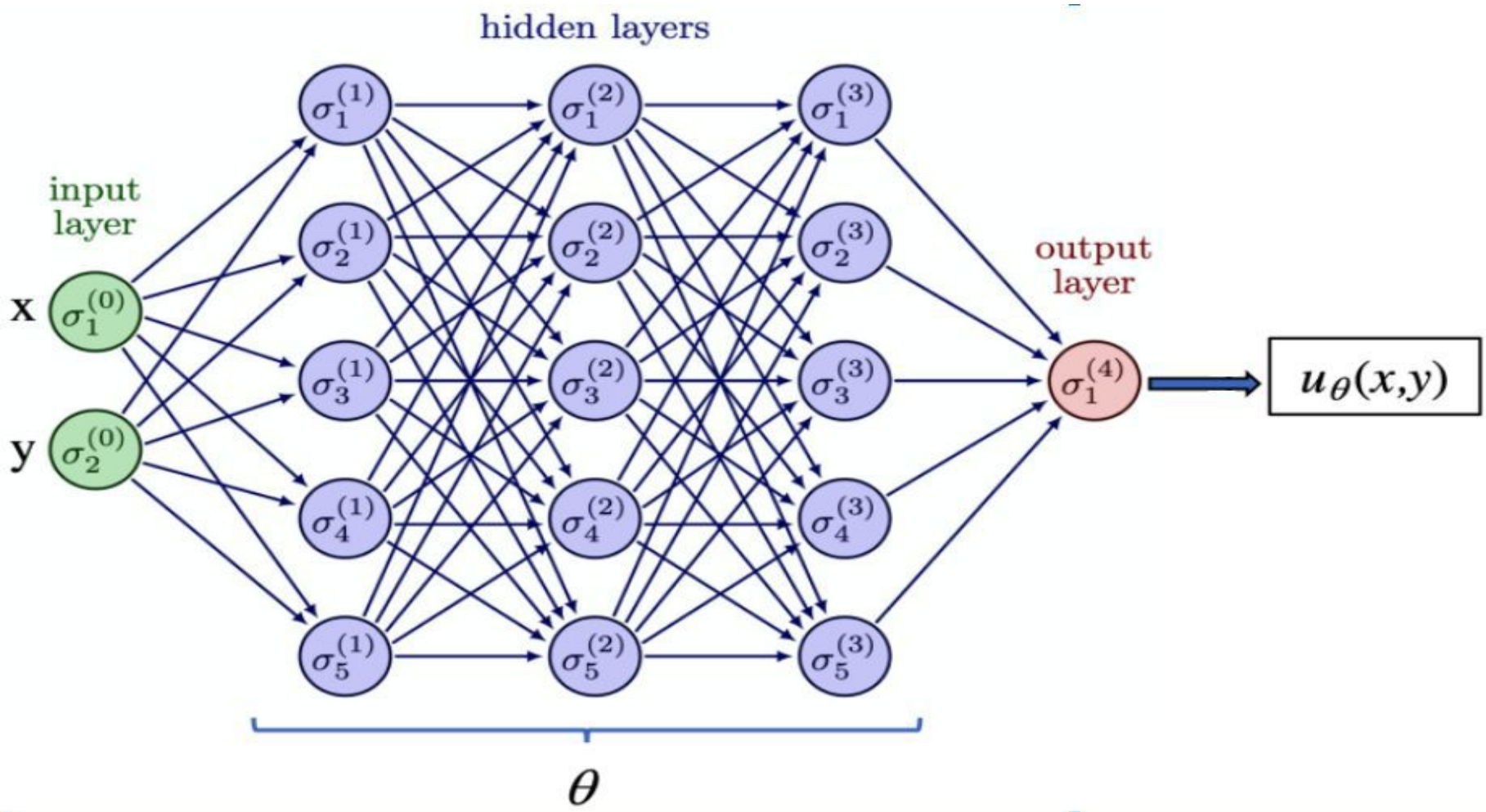
- It is a **computational model** designed to mimic the way the human brain processes information.

- It consists of **interconnected layers of units** (**neurons**),
  which work together to recognize patterns and make predictions.

- Neural networks are used in **machine learning** and **AI** for tasks like image recognition and speech processing.

1.1 Structure of a Neural Network

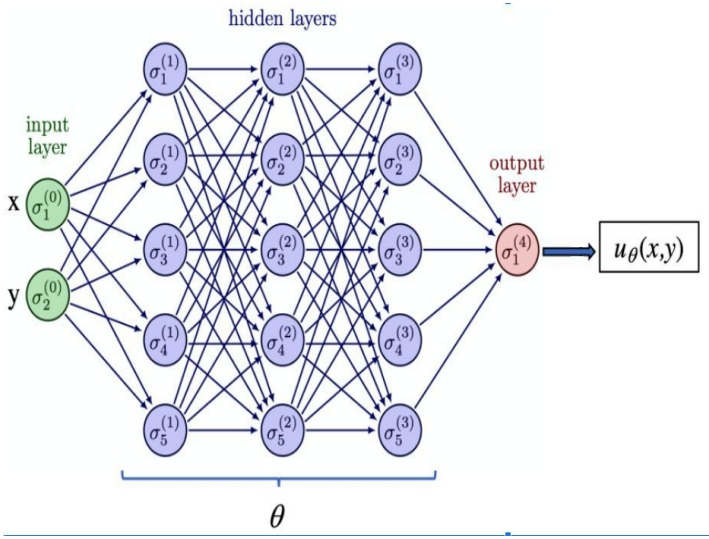
Neurons are the basic units of a neural network.



1.1 Structure of a Neural Network

Input Layer

1. The green circles on the left represent the input layer.
2. This layer receives the initial data.
3. In this case, there are two input nodes, features x and y.



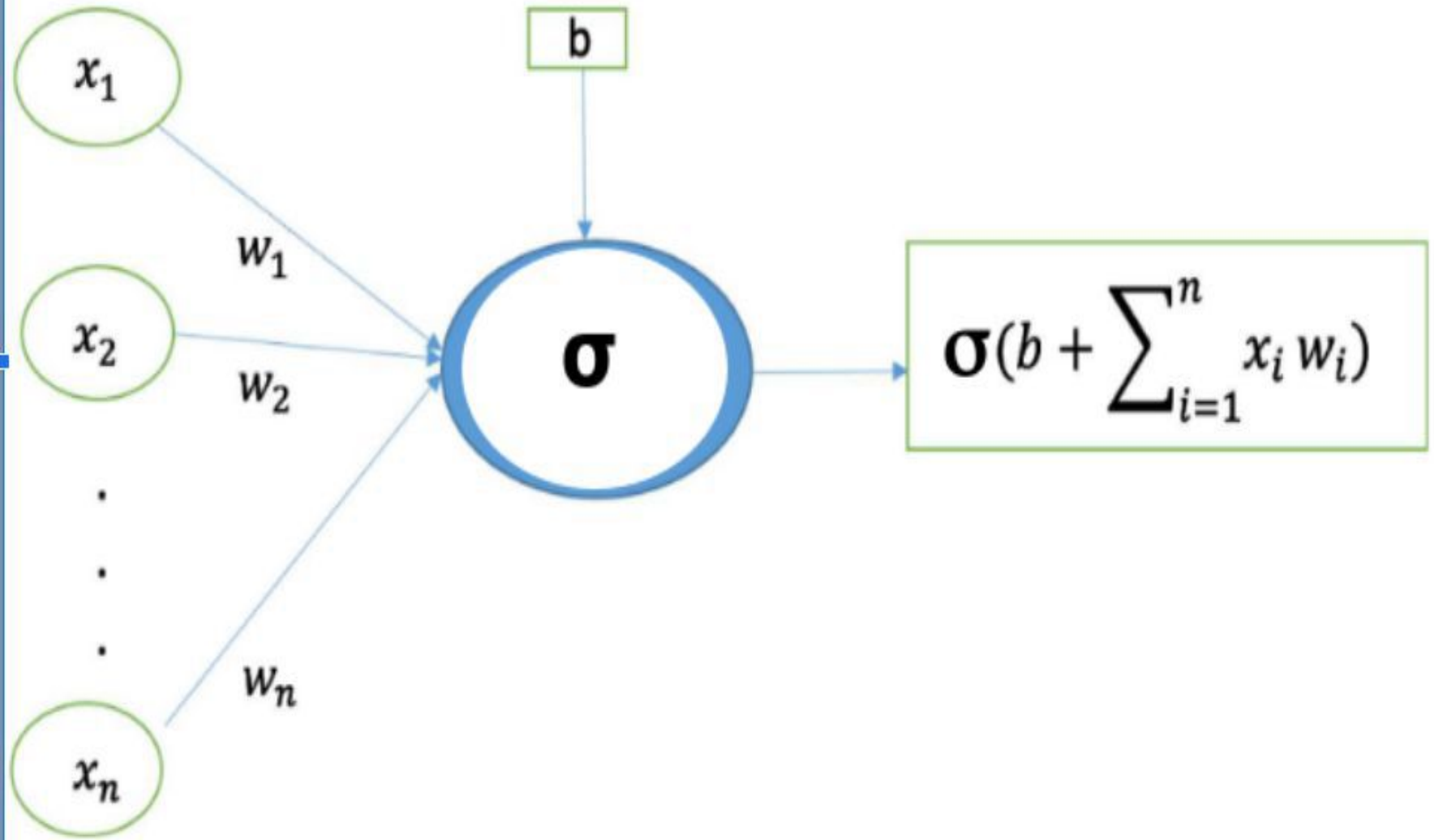
Hidden Layers

1. The blue circles represent hidden layers.
2. These layers process the input data through multiple neurons.
3. Each neuron in a hidden layer is connected to every neuron in the previous layer (fully connected).
4. The network shown has 3 hidden layers.

Output Layer

The output layer produces the final result of the network's computations, represented as uθ. This could be a classification label, a predicted value, etc.

1.2 Structure of a Neuron



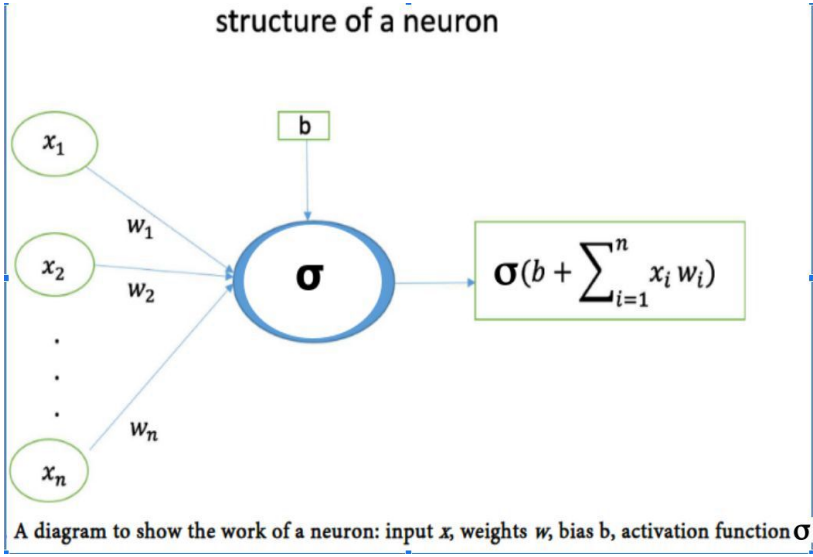
A diagram to show the work of a neuron: input x, weights w, bias b, activation function σ

1.2 Structure of a Neuron

Neuron: The basic unit of a neural network.

Inputs

The neuron receives multiple inputs (x1, x2, ..., xn). Each input has an associated weight (w1, w2, ..., wn) that **determines** its **influence** on the neuron's output.



A diagram to show the work of a neuron: input *x*, weights *w*, bias b, activation function σ

Weighted Sum and Bias

The neuron computes a weighted sum of the inputs, Σ (xi*wi), and adds a bias term, b. This is a way of shifting the activation function,σ, to better fit the data. The network learns by adjusting the weights and biases associated with each neuron.

Activation Function (σ)

Final Output

The final output of the neuron is σ(b + Σ(xi*wi)), which is then passed to neurons in the next layer.

1.3 Activation Function

Role of Activation Function

- a mathematical function
- introduce non-linearity into the model
- Transformation of the output of a neuron by applying a specific rule
- The network learns and models complex patterns in the data

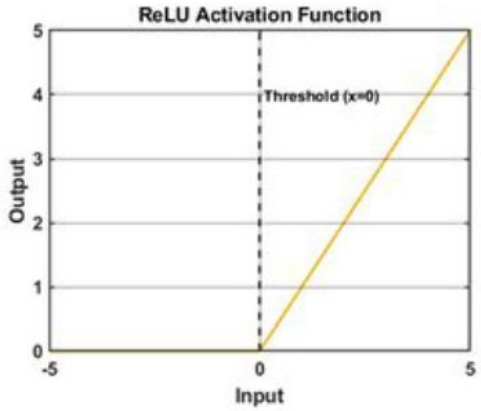
$$\mathcal{N}^\ell(\boldsymbol{x}) = \sigma(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}^\ell),$$

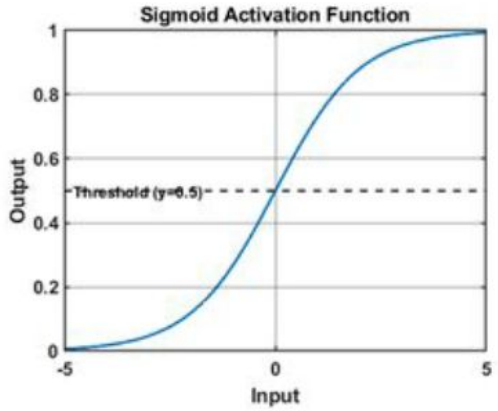
W = weight matrix
x = input (e.g., spatial/temporal coordinates)
b = the bias vector
Activation function σ

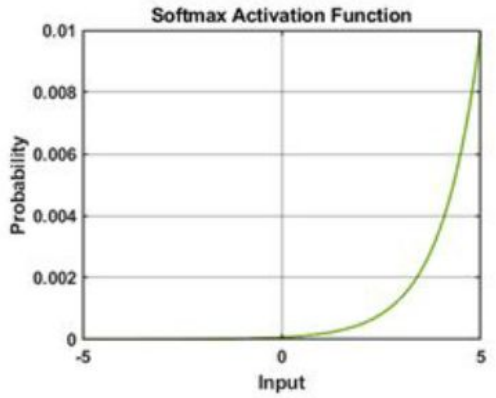
- σ --> allows complex patterns in the solution space, especially for non-linear PDEs
- Solving problems like image recognition & differential equation modeling

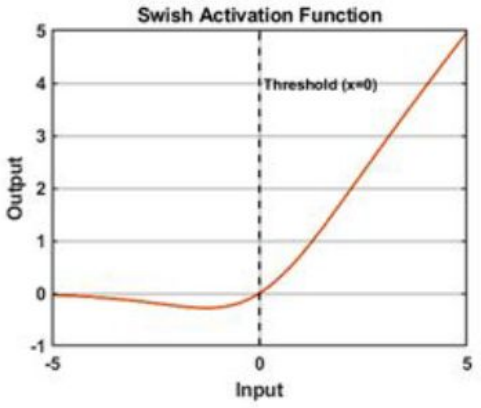
1.3 Activation Function

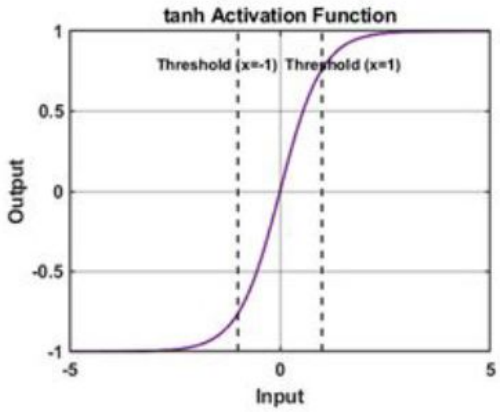
- Common Functions: ReLU, tanh, and sigmoid
- tanh(x) is preferred for approximating continuous physical phenomena

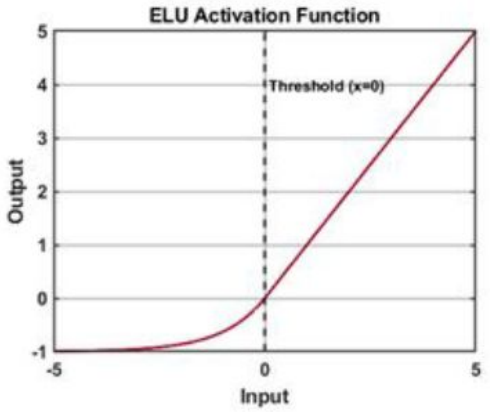










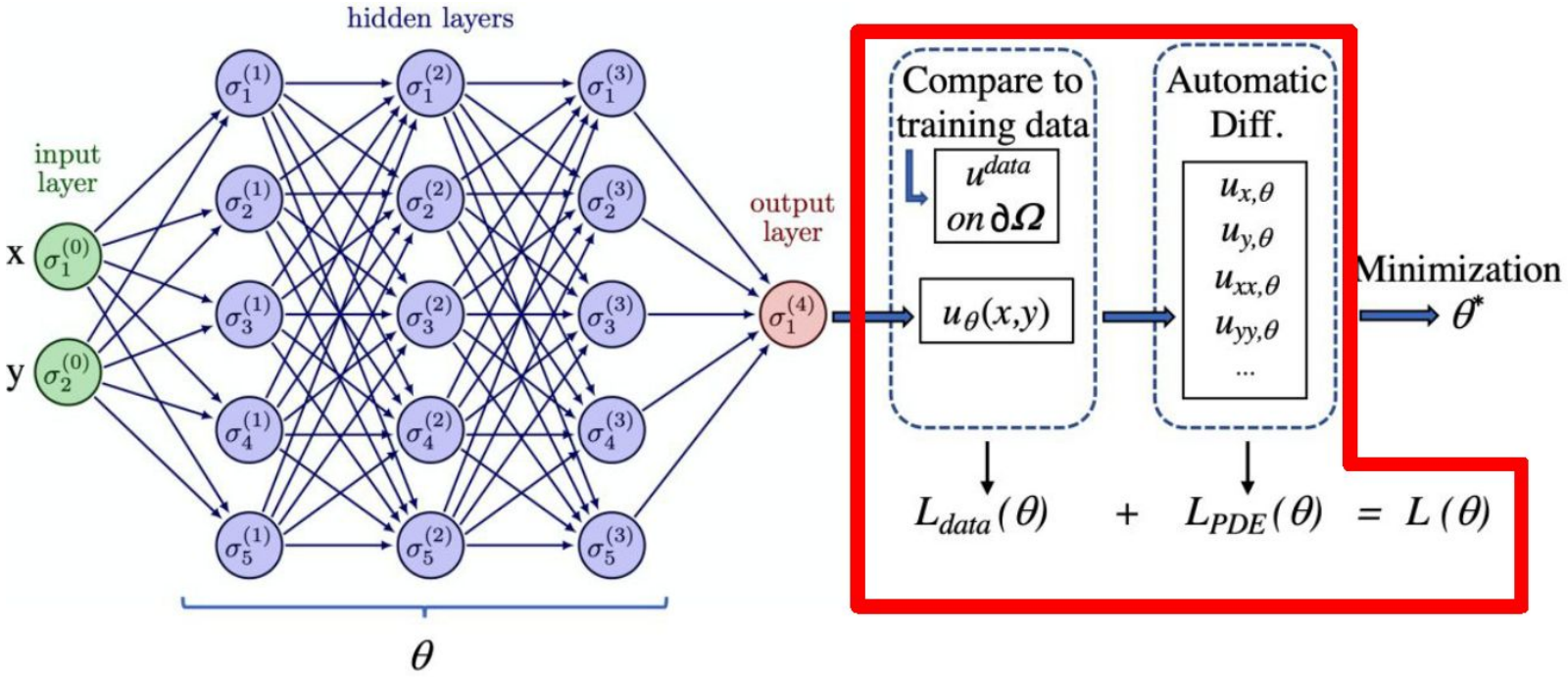


1.4 Loss Function

What is a Loss Function?

- Quantifies the difference between the predicted solution *u(θ)* and the expected solution (value compares with the true value)

- considering both data and physics constraints



1.4 Loss Function

$$L(\theta) = \omega_{data} L_{data}(\theta) + \omega_{PDE} L_{PDE}(\theta),$$

ω_{data} and ω_{PDE} weighting factors that balance the importance of the data loss and the PDE loss

- **Data Loss L_{DATA}:** how well the network's predictions fit the data (e.g., observed data points).

- **Physics Loss L_{PDE} :** Measures how well the predicted solution satisfies the underlying physical laws (e.g., PDE residuals).

Importance of Loss Function

- Lower loss --> better solution, aligns with the observed data and physical behavior

1.5 Optimization

Loss function and Activation Function together ensure that the optimization process can successfully find the best parameters θ.

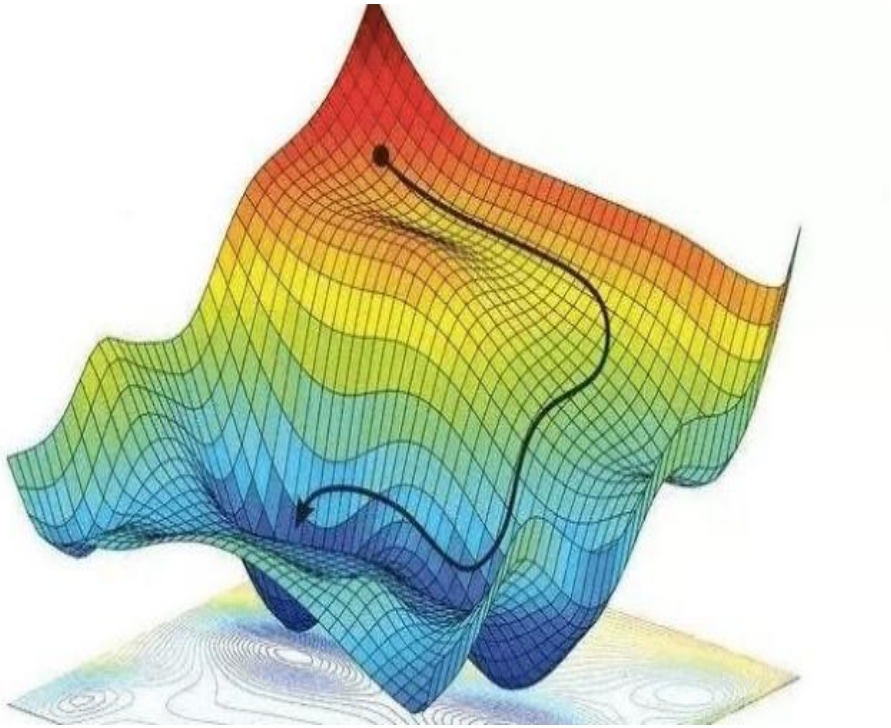
Optimization is the process of adjusting the model's parameters $\theta=(W,b)$ to minimize the total loss function.

Role of Functions in Optimization:

- **Loss Function:** Guides the optimization process by providing a measure of how close the model is to the desired outcome.
- **Activation Function:** Affects how gradients (the rate of change of the loss function with respect to the parameters) are calculated during backpropagation. This, in turn, influences the optimizer's effectiveness.

1.6 Gradient Descent Algorithm

- **Gradient descent algorithm** computes the gradient of the loss function with respect to each weight, determining how much to change each weight to reduce the error.
- Each point on the surface corresponds to a **set of weights, biases**, and the height (z-axis) represents the **error** or **loss** for that parameter set.
- The goal of optimization is to find the **global minimum**, which is the lowest point on the surface (where the loss is minimized).

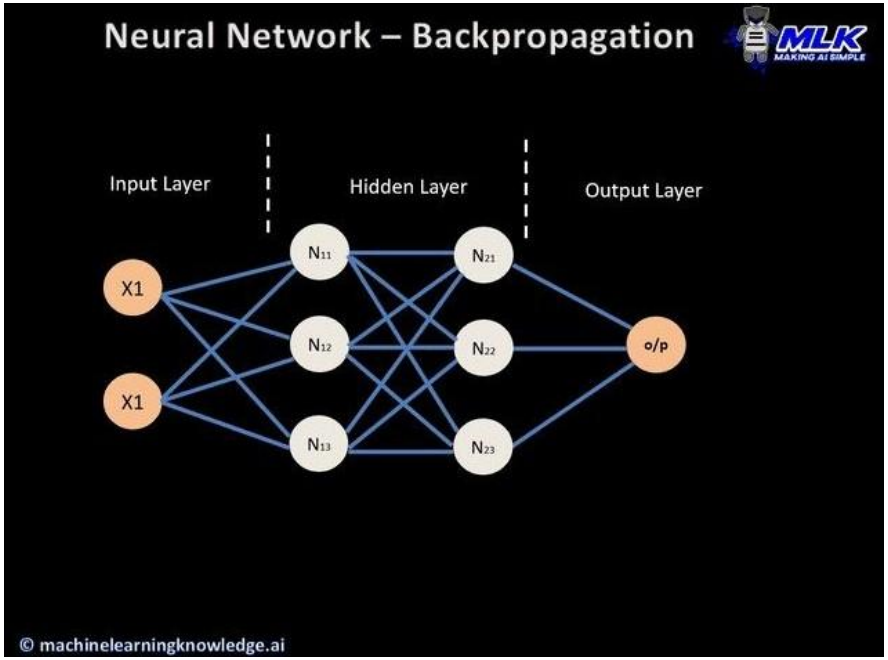


The 3D surface represents the **loss function** in an optimization problem.

1.7 Backpropagation

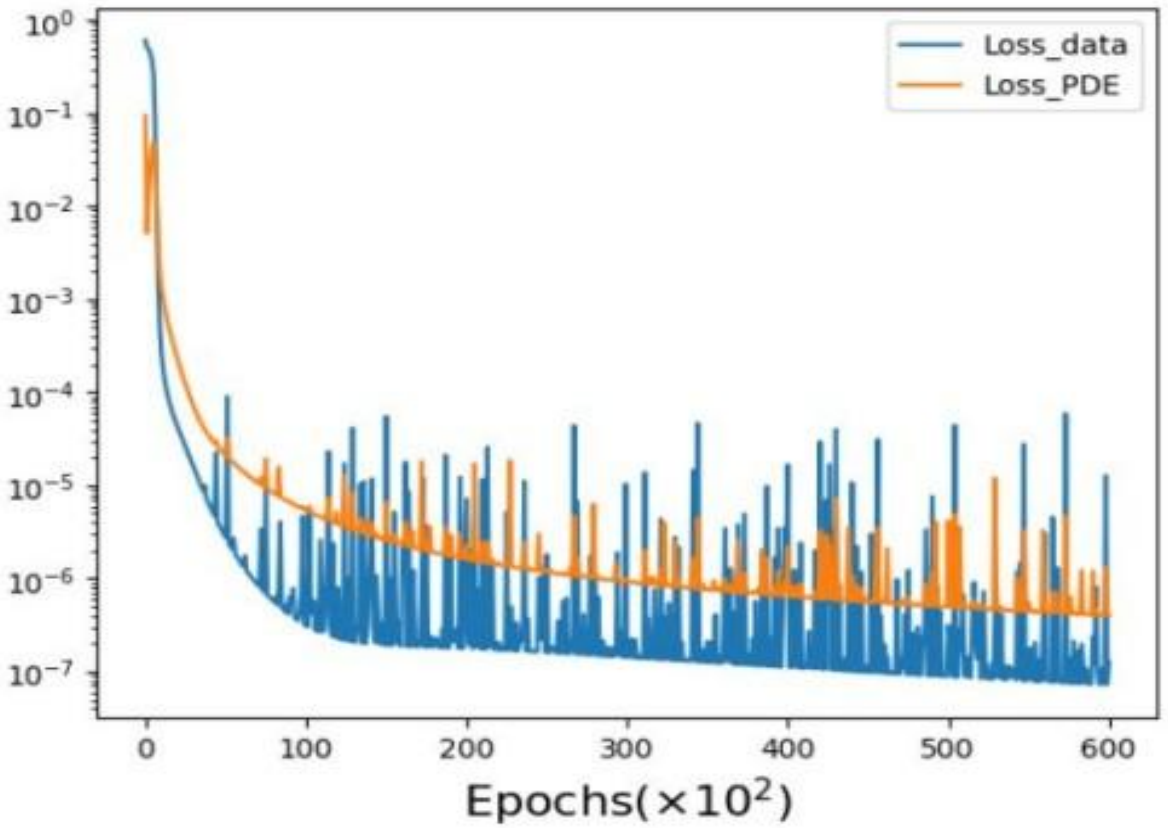
- The input data passes through the network, layer by layer, and an output is generated. This is the **forward pass** where the network makes a **prediction** based on the current weights.

- After the forward pass, the network calculates the **loss** or error.



- **Backpropagation** is the process of propagating this error **backward** through the network to update the weights

1.8 Loss Function Convergence

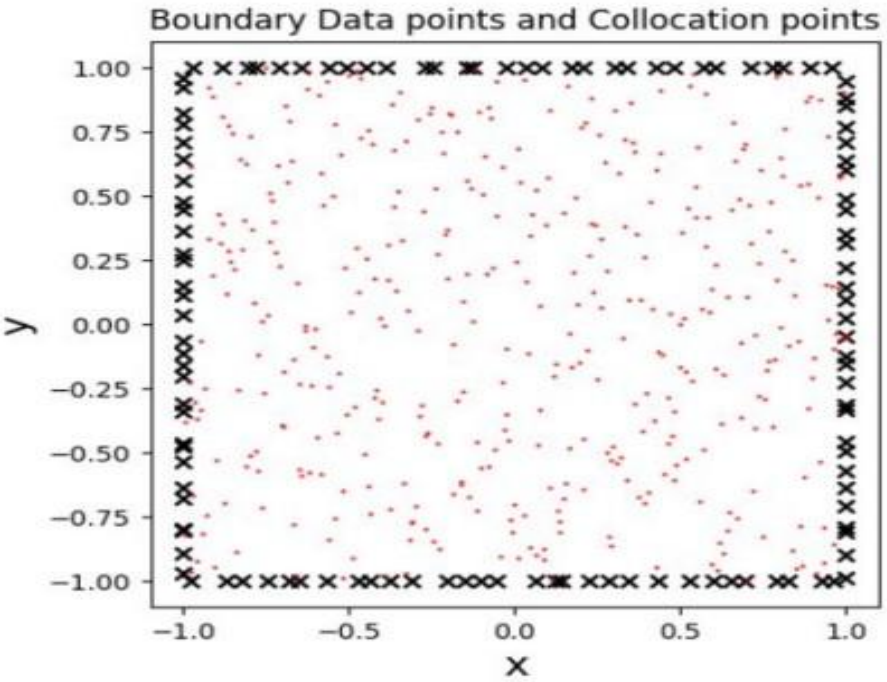


- As the training progresses (over epochs), both loss terms decrease, indicating that the model is learning both the data and the physical constraints effectively.

1.9 PINNS are mesh-free method!

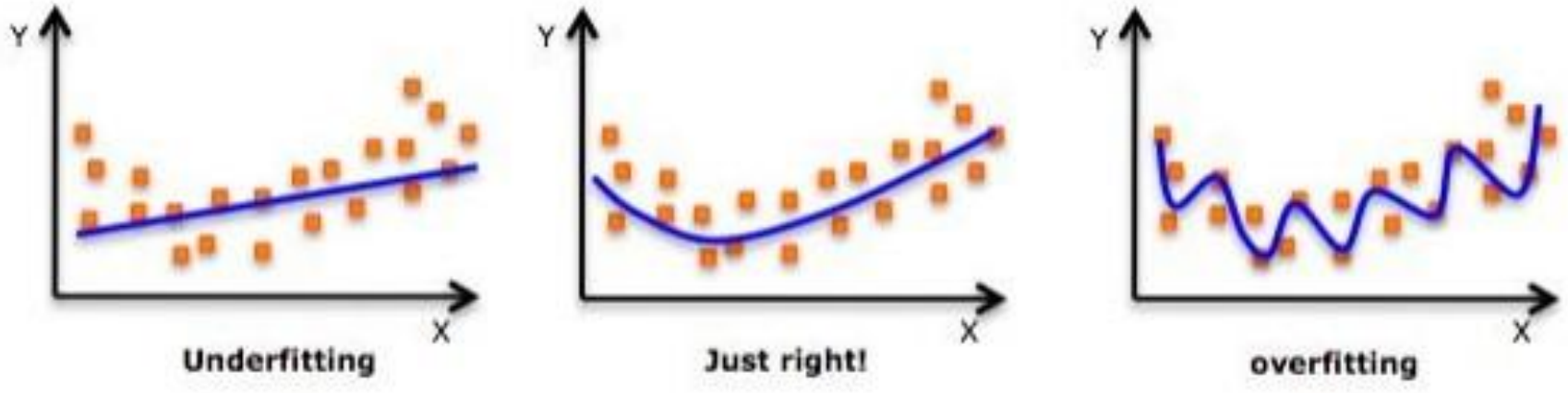
- In conventional numerical methods, solving PDEs requires creating a **mesh/grid** that divides the domain into small elements.

  Problem: This can be computationally expensive, especially in higher dimensions or when dealing with complex geometries (like those found in fluid dynamics)



- Instead of relying on a predefined grid, PINNs operate directly on **scattered points** in space, such as **collocation points** (red dots) shown in the graph.

1.10 Be careful with the fitting!



- **Underfitting:** The model predictions (blue line) are far from the data points (orange), indicating poor learning.
- **Just Right**: The model fits the data well.
- **Overfitting:** The model is too complex and fits the noise in the data.

2.1 The Problem

- (Linear) Partial Deferential Equation: **1 D Diffusion Equation**

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2}$$

- Spatial Domain:

$$x \in [0, L = 1]$$

- Diffusion coefficient:

$$D = 0.1$$

- Initial Condition:

$$u(x,0) = f(x) = 50, \forall x$$

- Dirichlet Boundary Conditions:

$$u(0,t) = u(L,t) = 60, \forall t > 0$$

- Analytical Solution:

$$u(x,t) = \sum_{n=1}^{\infty}\left(\frac{2}{L}\int_0^L f(\xi)\sin\left(\frac{\pi n}{L}\xi\right)d\xi\right)\sin\left(\frac{\pi n}{L}x\right)\exp\left(-D\left(\frac{\pi n}{L}\right)^2 t\right)$$

2.2 Code

1) Class for Neural Network (NN):
   - Forward Propagation
     - ✓ Activation Function: ***tanh(x)***
     - ✓ Weights: Xavier Initialization
     - ✓ Biases: Zero
   - Residuals for:
     - ✓ Boundary Condition: $BC_{1,2} = u_{1,2} - 60$
     - ✓ Initial Condition Residual: $IC = u_{IC} - 50$
     - ✓ PDE (on training points): $PDE = u_t - au_{xx}$

2) Grid: Random coordinates generation for boundaries/initial condition

3) NN Architecture: layers, (Adam) optimizer, epochs

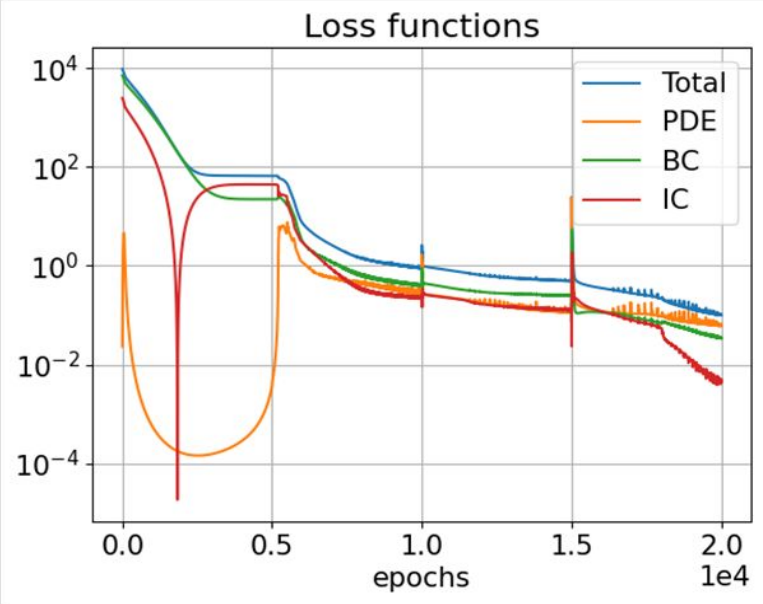
4) Training:
   - **Loss = weighted sum of residuals**
   - Back Propagation

5) Predictions of Initial Condition and NN Solution

6) Computation of Analytical Solution

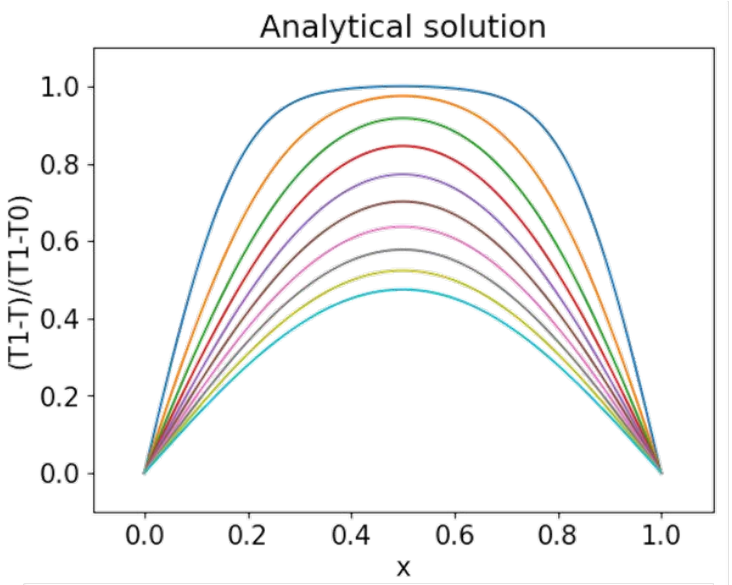
7) Error between the two methods:

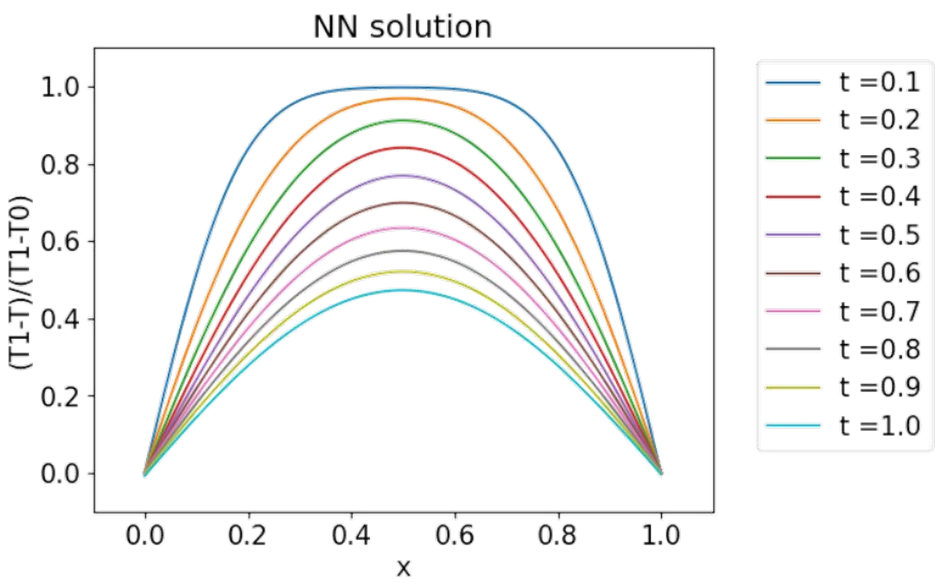
$$MSE = \frac{\sum_i \left(u_i - u_{i,an}\right)^2}{N}$$

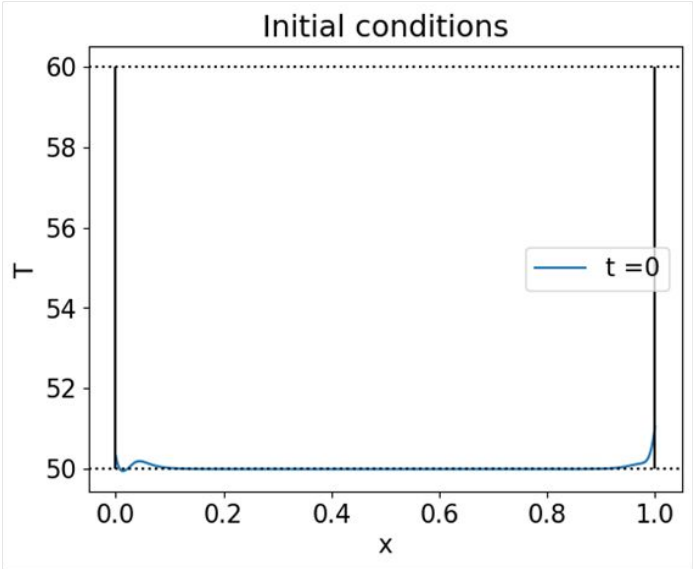


2.3 Results

❑ *20,000 Epochs*







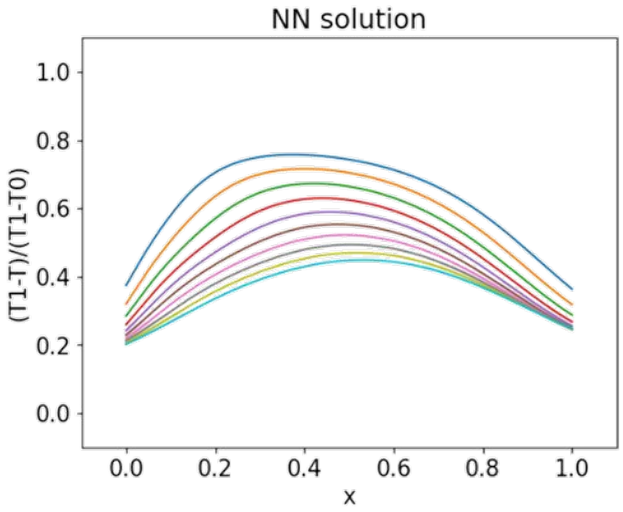

```
ti = 0.0 mean error = 0.06700099449025884
ti = 0.1 mean error = 1.8267733262831113e-05
ti = 0.2 mean error = 1.242470181339765e-05
ti = 0.3 mean error = 1.1600099963025466e-05
ti = 0.4 mean error = 1.1473538192104069e-05
ti = 0.5 mean error = 9.581188435158434e-06
ti = 0.6 mean error = 7.224438634077675e-06
ti = 0.7 mean error = 5.511291316463746e-06
ti = 0.8 mean error = 3.97876654485256e-06
ti = 0.9 mean error = 2.6446457811408243e-06
ti = 1.0 mean error = 5.430226726385929e-06
```

✓**Good Agreement between the two methods.**

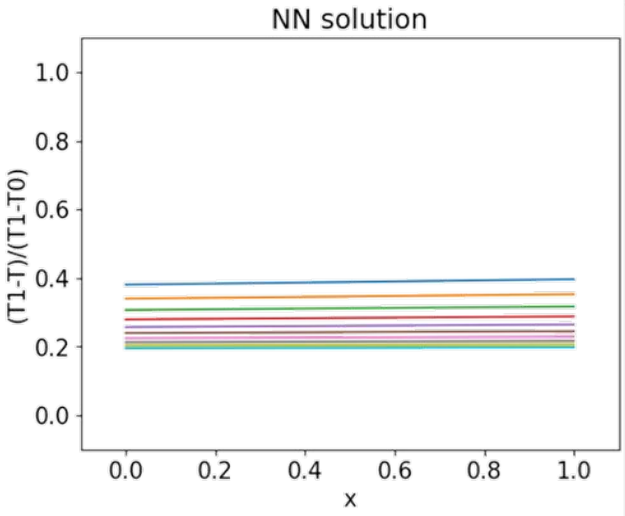
2.3 Results

❑ *Tinkering with the code…*

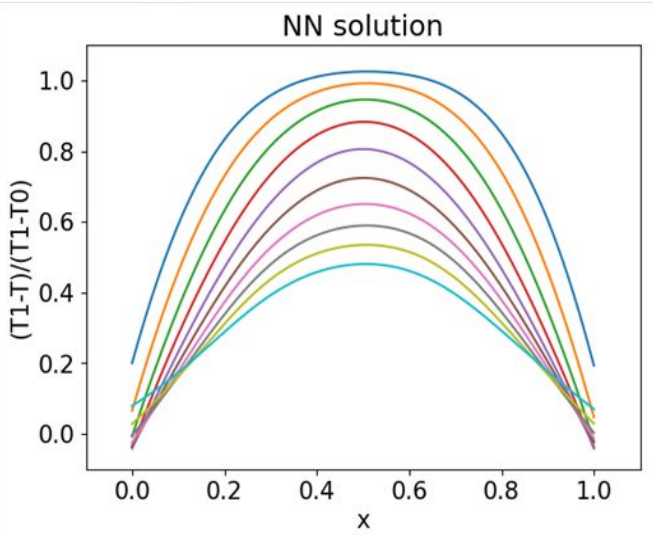
• 5,000 Epochs



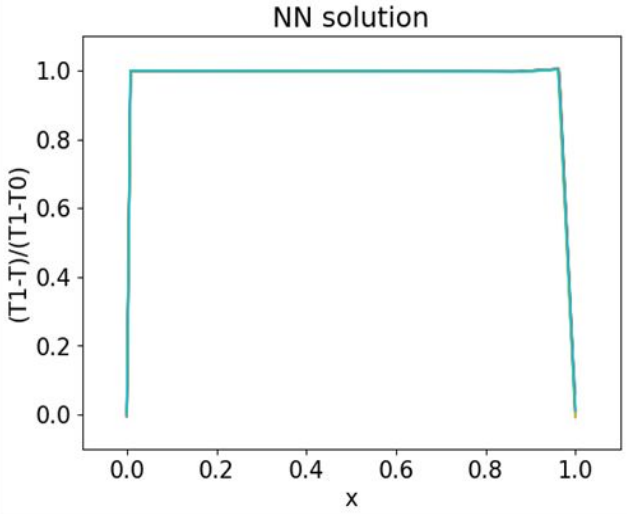
• 2,32,1



• 2,64,1



• 20,000 Epochs/ReLU



			2×10^4 Epochs		
Epochs	**Duration**	**MSE**	**Layers**	**Duration (min)**	**MSE**
5×10^2	4 sec	10	2,32,1	2.13	10^{-2}–10^{-1}
10^3	15 sec	10^{-1}	2,64,1	2.55	10^{-4}
5×10^3	1.16 min	10^{-2}	2,32,32,32,1	4.25	10^{-5}–10^{-6}
10^4	2.40 min	10^{-5}	2,32,32,32,32,32,1	7.20	10^{-4}–10^{-5}
2×10^4	4.25 min	10^{-5}–10^{-6}	2,64,64,64,1	8.48	10^{-6}
3×10^4	6.34 min	10^{-6}	2,64,64,64,64,1	11.25	10^{-6}

3. Conclusions

1. PINNs are easy to use → They naturally incorporate physical laws, reducing the need for extensive pre-processing of data.
2. PINNs can perform well with limited data, leveraging known physics to fill in gaps.
3. PINNs have the potential to bridge gaps between machine learning and traditional scientific fields, fostering new interdisciplinary approaches.
4. The main advantage of PINNs is its numerical simplicity. There is no need of discretization, like in a finite-difference method.
5. The training process can be long for some problems. The rules for the choice of the network architecture are not well determined. The accuracy is good but not as good compared to traditional schemes.

4. Bibliography

1. Hubert Baty, A hands-on introduction to PINNs, 2024
2. Jorge F. Urbán, Petros Stefanou, Clara Dehman,Modelling Force-Free Neutron Star Magnetospheres using Physics-Informed Neural Networks, José A. Pons, 2023
3. Dimitropoulos, I. Contopoulos, V. Mpisketzis, E. Chaniadakis, The pulsar magnetosphere with machine learning: methodology, Volume 528, Issue 2, 2024