An introduction to physics-informed neural networks

A hands on approach

Emir Esenov

Electric Power Engineering
Chalmers University of Technology
Gothenburg, Sweden

September 9, 2025

CHALMERS

Agenda

What we will do

- Take an abstract look at the PINN framework and how it evolves from supervised learning
- Show simple examples
- Provide the audience with a Jupyter notebook available in Google Colab to experiment with code

What we will not do

- Explain neural networks (weights, biases, activation functions, backpropagation,...)
- Look at complicated systems of differential equations trained on big datasets
- Explain all the code (do it at your own pace afterwards if you're interested, can use ChatGPT to help explain,...)

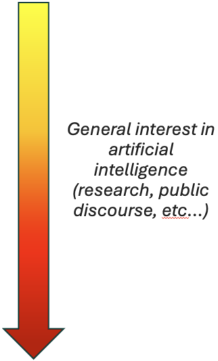
Neural networks: groundwork → breakthrough

- **1940s–2000s: Groundwork.** From early perceptrons to backprop and convolutional neural networks. Good ideas, limited by data, compute, and training issues.
- **2006–2011: Pieces click.** Better activations (ReLU), big labeled datasets, and GPUs make deep nets practical.
- **2012: AlexNet.** Scales the recipe and wins ImageNet by a large margin. Sets off an AI takeoff that is still ongoing.

Neural networks: 2012-onward

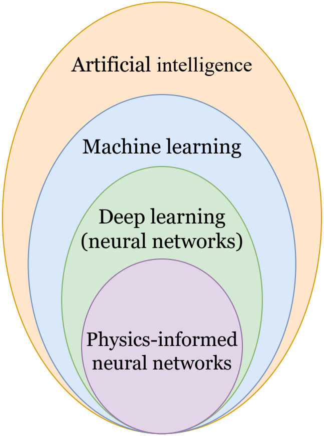
Some notable events

- 2012–2016: AlexNet wins ImageNet. DQN hits "human-level" Atari. ResNet becomes a default backbone for vision tasks. AlphaGo beats the world champion at Go.
- 2017-2020: Transformer architecture ("Attention Is All You Need"). BERT uses self-supervised pretraining for language understanding. GPT-3. Diffusion models. AlphaFold (protein folding).
- 2020–onward: AlphaFold2 protein structures at near experimental accuracy. Stable diffusion, text-to-image goes mainstream. ChatGPT. GPT-4. LLama, Sora, DeepSeek, ...



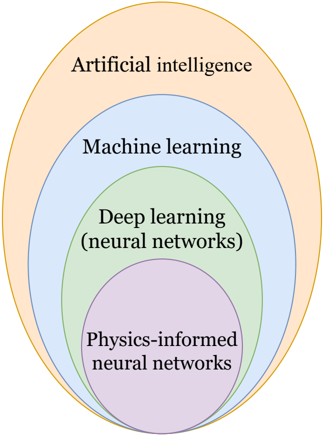
Terminology: AI, machine learning, deep learning, ...

- AI (how to make machines perform human-level intelligence tasks): broad field researched for decades. Includes symbolic reasoning systems, planning algorithms, search algorithms, evolutionary computation, ...



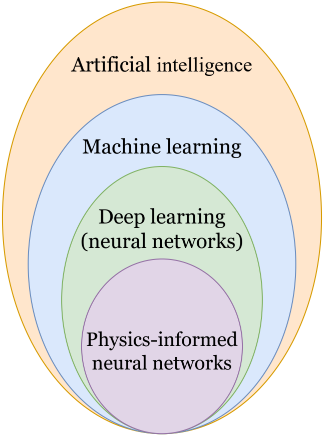
Terminology: AI, machine learning, deep learning, ...

- AI (how to make machines perform human-level intelligence tasks): broad field researched for decades. Includes symbolic reasoning systems, planning algorithms, search algorithms, evolutionary computation, ...
- Other forms of AI research includes ethics, safety, interpretability, philosophy of intelligence, ...



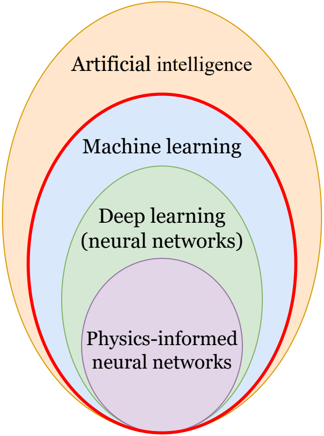
Terminology: AI, machine learning, deep learning, . . .

- AI (how to make machines perform human-level intelligence tasks): broad field researched for decades. Includes symbolic reasoning systems, planning algorithms, search algorithms, evolutionary computation, . . .
- Other forms of AI research includes ethics, safety, interpretability, philosophy of intelligence, . . .
- Historically, AI research started with rule-based and symbolic approaches (GOFAI), well before machine learning became dominant. Today, machine learning/deep learning is almost synonymous with AI due to its dominance.



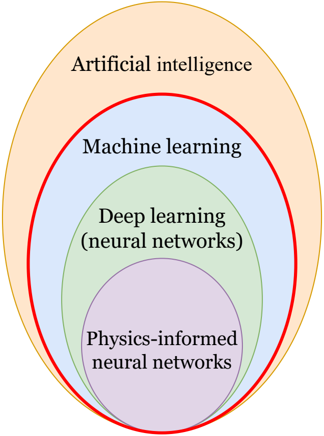
Terminology: AI, machine learning, deep learning, . . .

- Machine learning: use algorithms or mathematical models that learn patterns from data using statistical or probabilistic methods



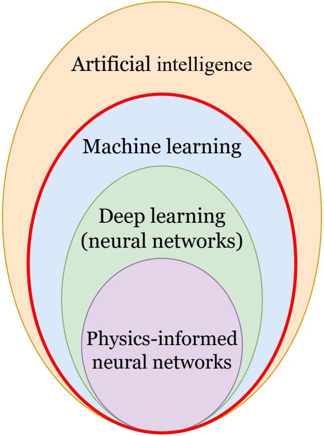
Terminology: AI, machine learning, deep learning, . . .

- Machine learning: use algorithms or mathematical models that learn patterns from data using statistical or probabilistic methods
- Deep learning: uses artificial neurons to create neural networks which learn patterns from data



Terminology: AI, machine learning, deep learning, . . .

- Machine learning: use algorithms or mathematical models that learn patterns from data using statistical or probabilistic methods
- Deep learning: uses artificial neurons to create neural networks which learn patterns from data
- Physics-informed neural networks (PINNs): an extension of neural networks tailored to analyzing differential equations



Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.

Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.

Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.
    - **Classification**: outputs take discrete values for categorization purposes, not covered here.

Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.
  - **Classification**: outputs take discrete values for categorization purposes, not covered here.
- **Supervised learning ("learn by example")**: learn from data points containing input (domain) and output (solution). Data may be collected from human annotation, experimental measurements, . . .

Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.
  - **Classification**: outputs take discrete values for categorization purposes, not covered here.
- **Supervised learning ("learn by example")**: learn from data points containing input (domain) and output (solution). Data may be collected from human annotation, experimental measurements, . . .
  - **Unsupervised learning, semi-supervised learning, reinforcement learning**: alternative approaches, not covered here.

Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.
  - **Classification**: outputs take discrete values for categorization purposes, not covered here.
- **Supervised learning ("learn by example")**: learn from data points containing input (domain) and output (solution). Data may be collected from human annotation, experimental measurements, . . .
  - **Unsupervised learning, semi-supervised learning, reinforcement learning**: alternative approaches, not covered here.
- **Parametric models**: uses a fixed set of parameters to define its structure and behavior. Parameters can be adjusted to create variations of the original model.

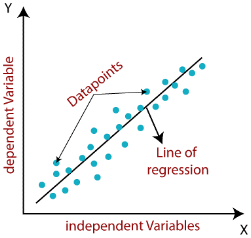
Machine learning: supervised learning for regression tasks with parametric models

Today's focus, looking at a subfield of machine learning

- **Machine learning** and **deep learning** are two very broad fields with many different applications. Today we will look at a specific framework relevant for understanding PINNs.
- **Regression**: numerical tasks. Our focus today, leads us nicely to PINNs.
  - **Classification**: outputs take discrete values for categorization purposes, not covered here.
- **Supervised learning ("learn by example")**: learn from data points containing input (domain) and output (solution). Data may be collected from human annotation, experimental measurements, . . .
  - **Unsupervised learning, semi-supervised learning, reinforcement learning**: alternative approaches, not covered here.
- **Parametric models**: uses a fixed set of parameters to define its structure and behavior. Parameters can be adjusted to create variations of the original model.
  - **Non-parametric models**: doesn't assume a fixed set of parameters, typically more flexible models that scale with data size. Not covered here.

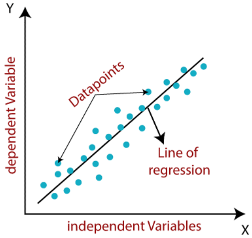
Supervised learning for regression: problem setup

- Data: pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$.



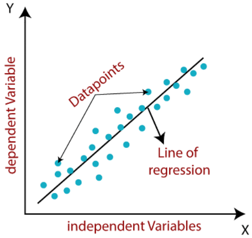
Supervised learning for regression: problem setup

- Data: pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$.
- Goal: learn a mapping $f : \mathbb{R}^d \to \mathbb{R}$ so that $f(x_i)$ is close to y_i for seen data and predicts well for new x.



Supervised learning for regression: problem setup

- Data: pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$.
- Goal: learn a mapping $f : \mathbb{R}^d \to \mathbb{R}$ so that $f(x_i)$ is close to y_i for seen data and predicts well for new x.
- Curve–fitting view: assume $y_i = g(x_i) + \varepsilon_i$ (unknown function g plus noise ε_i).



Parametric models

- Choose a *parametric* family $\{f_\theta \mid \theta \in \mathbb{R}^p\}$ (parameters θ).

Parametric models

- Choose a *parametric* family $\{f_\theta \mid \theta \in \mathbb{R}^p\}$ (parameters θ).
- Examples: linear regression, polynomials, splines, neural networks.

Parametric models

- Choose a *parametric* family $\{f_\theta \mid \theta \in \mathbb{R}^p\}$ (parameters θ).
- Examples: linear regression, polynomials, splines, neural networks.
- Prediction for an input x: $\hat{y} = f_\theta(x)$.

Parametric models

- Choose a *parametric* family $\{f_\theta \mid \theta \in \mathbb{R}^p\}$ (parameters θ).
- Examples: linear regression, polynomials, splines, neural networks.
- Prediction for an input x: $\hat{y} = f_\theta(x)$.
- Training (curve-fitting) $=$ find θ that makes $\hat{y}_i = f_\theta(x_i)$ fit the data well.

Loss function: Mean Squared Error (MSE)

- Measure discrepancy per example with squared error $(f_\theta(x_i) - y_i)^2$.

Loss function: Mean Squared Error (MSE)

- Measure discrepancy per example with squared error $(f_\theta(x_i) - y_i)^2$.
- Dataset loss (empirical MSE):

$$\mathrm{MSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)^2.$$

Loss function: Mean Squared Error (MSE)

- Measure discrepancy per example with squared error $(f_\theta(x_i) - y_i)^2$.
- Dataset loss (empirical MSE):

$$\mathrm{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i\big)^2.$$

- Why MSE? Smooth and differentiable, heavily penalizes large errors.

Empirical risk minimization and gradients

- Learning problem: $\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i \big)^2.$

Empirical risk minimization and gradients

- Learning problem: $\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i \big)^2.$
- Gradient of the objective:

$$\nabla_\theta \mathrm{MSE}(\theta) = \frac{2}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i \big)\, \nabla_\theta f_\theta(x_i).$$

Empirical risk minimization and gradients

- Learning problem: $\theta^\star = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i \big)^2.$
- Gradient of the objective:

$$\nabla_\theta \mathrm{MSE}(\theta) = \frac{2}{N} \sum_{i=1}^{N} \big(f_\theta(x_i) - y_i \big) \, \nabla_\theta f_\theta(x_i).$$

- For neural networks, $\nabla_\theta f_\theta(x_i)$ is computed efficiently via backpropagation algorithm.

Empirical risk minimization and gradients

- Learning problem: $\theta^\star = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)^2.$
- Gradient of the objective:

$$\nabla_\theta \mathrm{MSE}(\theta) = \frac{2}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)\,\nabla_\theta f_\theta(x_i).$$

- For neural networks, $\nabla_\theta f_\theta(x_i)$ is computed efficiently via backpropagation algorithm.
- The gradient gives is a vector operation that gives the direction of steepest ascent.

Empirical risk minimization and gradients

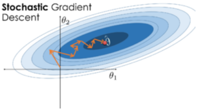
- Learning problem: $\theta^\star = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)^2.$
- Gradient of the objective:

$$\nabla_\theta \mathrm{MSE}(\theta) = \frac{2}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)\,\nabla_\theta f_\theta(x_i).$$

- For neural networks, $\nabla_\theta f_\theta(x_i)$ is computed efficiently via backpropagation algorithm.
- The gradient gives is a vector operation that gives the direction of steepest ascent.
- Update our parameters in the opposite (negative) direction $\rightarrow$ one step toward minimizing the loss function $\rightarrow$ improves our parametric model

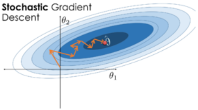
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).



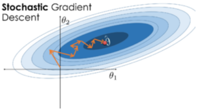
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):



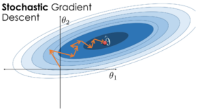
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):
  - Pick a mini-batch $\mathcal{B} \subset \{1, \ldots, N\}$.



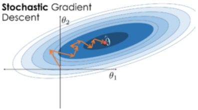
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):
  - Pick a mini-batch $\mathcal{B} \subset \{1, \ldots, N\}$.
  - Compute $\widehat{\mathrm{MSE}}(\theta) = \dfrac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (f_\theta(x_i) - y_i)^2$.



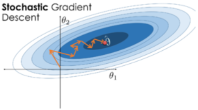
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):
  - Pick a mini-batch $\mathcal{B} \subset \{1, \ldots, N\}$.
  - Compute $\widehat{\mathrm{MSE}}(\theta) = \dfrac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (f_\theta(x_i) - y_i)^2$.
  - Update $\theta \leftarrow \theta - \eta \nabla_\theta \widehat{\mathrm{MSE}}(\theta)$ (learning rate $\eta > 0$).



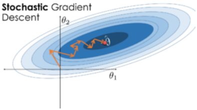
Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):
  - Pick a mini-batch $\mathcal{B} \subset \{1, \dots, N\}$.
  - Compute $\widehat{\mathrm{MSE}}(\theta) = \dfrac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (f_\theta(x_i) - y_i)^2$.
  - Update $\theta \leftarrow \theta - \eta \nabla_\theta \widehat{\mathrm{MSE}}(\theta)$ (learning rate $\eta > 0$).
- Stop when the loss stops improving or after a fixed budget of epochs.



Training with (stochastic) gradient descent

- Initialize θ (e.g., randomly).
- Repeat for many iterations (epochs):
  - Pick a mini-batch $\mathcal{B} \subset \{1, \ldots, N\}$.
  - Compute $\widehat{\mathrm{MSE}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (f_\theta(x_i) - y_i)^2$.
  - Update $\theta \leftarrow \theta - \eta \nabla_\theta \widehat{\mathrm{MSE}}(\theta)$ (learning rate $\eta > 0$).
- Stop when the loss stops improving or after a fixed budget of epochs.
- Next: a short Python demo training such a network to fit noisy data and plotting predictions and MSE over epochs.



Code example

Code example

From NN regression to physics-informed neural networks

- What if our regression task is a physical problem governed by differential equations?

From NN regression to physics-informed neural networks

- What if our regression task is a physical problem governed by differential equations?
- Neural networks fit data well but are purely data-driven: no built-in physical interpretation.

From NN regression to physics-informed neural networks

- What if our regression task is a physical problem governed by differential equations?
- Neural networks fit data well but are purely data-driven: no built-in physical interpretation.
- Experimental data can be noisy, sparse, or not cover the full domain.

From NN regression to physics-informed neural networks

- What if our regression task is a physical problem governed by differential equations?
- Neural networks fit data well but are purely data-driven: no built-in physical interpretation.
- Experimental data can be noisy, sparse, or not cover the full domain.
- We often know physics a priori (differential equations, boundary/initial conditions).

From NN regression to physics-informed neural networks

- What if our regression task is a physical problem governed by differential equations?
- Neural networks fit data well but are purely data-driven: no built-in physical interpretation.
- Experimental data can be noisy, sparse, or not cover the full domain.
- We often know physics a priori (differential equations, boundary/initial conditions).
- Idea of PINNs: keep NN regression and add MSE-style penalties that encode the physics.

What is a PINN?

- Use the same model $f_\theta(x)$ to approximate the solution.

What is a PINN?

- Use the same model $f_\theta(x)$ to approximate the solution.
- Build a loss that combines data-fit and physics terms:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{data}}} \sum_i \big(f_\theta(x_i) - y_i\big)^2}_{\text{data MSE (optional)}} + \underbrace{\frac{1}{N_{\text{col}}} \sum_j r_\theta(x_j)^2}_{\text{ODE residual MSE}} + \underbrace{\frac{1}{N_{\text{bc}}} \sum_k b_\theta(z_k)^2}_{\text{BC/IC MSE}}.$$

What is a PINN?

- Use the same model $f_\theta(x)$ to approximate the solution.
- Build a loss that combines data-fit and physics terms:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{data}}} \sum_i \big(f_\theta(x_i) - y_i\big)^2}_{\text{data MSE (optional)}} + \underbrace{\frac{1}{N_{\text{col}}} \sum_j r_\theta(x_j)^2}_{\text{ODE residual MSE}} + \underbrace{\frac{1}{N_{\text{bc}}} \sum_k b_\theta(z_k)^2}_{\text{BC/IC MSE}}.$$

- Residuals are built from f_θ:

$$r_\theta(x) = \mathcal{N}[\,f_\theta\,](x), \qquad b_\theta(z) = \mathcal{B}[\,f_\theta\,](z).$$

What is a PINN?

- Use the same model $f_\theta(x)$ to approximate the solution.
- Build a loss that combines data-fit and physics terms:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{data}}}\sum_i \big(f_\theta(x_i) - y_i\big)^2}_{\text{data MSE (optional)}} + \underbrace{\frac{1}{N_{\text{col}}}\sum_j r_\theta(x_j)^2}_{\text{ODE residual MSE}} + \underbrace{\frac{1}{N_{\text{bc}}}\sum_k b_\theta(z_k)^2}_{\text{BC/IC MSE}}.$$

- Residuals are built from f_θ:

$$r_\theta(x) = \mathcal{N}[\, f_\theta\,](x), \qquad b_\theta(z) = \mathcal{B}[\, f_\theta\,](z).$$

- Train with gradient descent as before; derivatives inside r_θ come from automatic differentiation.

How the physics term is built

- Differential equation on a domain: $\mathcal{N}[u](x) = 0$ with boundary/initial conditions $\mathcal{B}[u](z) = 0$.

How the physics term is built

- Differential equation on a domain: $\mathcal{N}[u](x) = 0$ with boundary/initial conditions $\mathcal{B}[u](z) = 0$.
- With our model, replace u by f_θ and define

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \big(\mathcal{N}[\, f_\theta \,](x_j)\big)^2, \quad \mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_k \big(\mathcal{B}[\, f_\theta \,](z_k)\big)^2.$$

How the physics term is built

- Differential equation on a domain: $\mathcal{N}[u](x) = 0$ with boundary/initial conditions $\mathcal{B}[u](z) = 0$.
- With our model, replace u by f_θ and define

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \Big(\mathcal{N}[\, f_\theta \,](x_j)\Big)^2, \quad \mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_k \Big(\mathcal{B}[\, f_\theta \,](z_k)\Big)^2.$$

- N_{col}: number of collocation points in the domain (usually sampled randomly or on a grid inside the physical domain).

How the physics term is built

- Differential equation on a domain: $\mathcal{N}[u](x) = 0$ with boundary/initial conditions $\mathcal{B}[u](z) = 0$.
- With our model, replace u by f_θ and define

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \Big(\mathcal{N}[\, f_\theta \,](x_j)\Big)^2, \quad \mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_k \Big(\mathcal{B}[\, f_\theta \,](z_k)\Big)^2.$$

- N_{col}: number of collocation points in the domain (usually sampled randomly or on a grid inside the physical domain).
- N_{bc}: number of boundary (or initial) condition points (sampled on the boundary/initial surface).

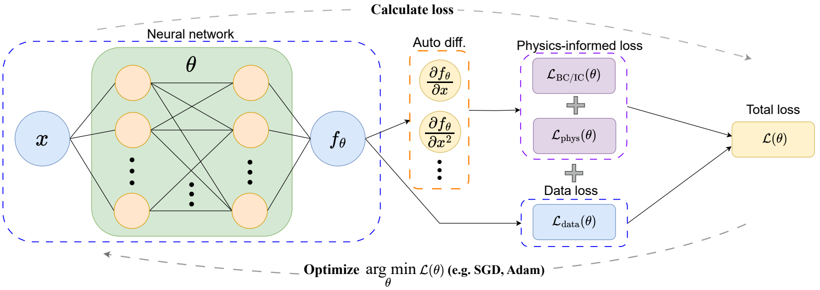
How the physics term is built

- Differential equation on a domain: $\mathcal{N}[u](x) = 0$ with boundary/initial conditions $\mathcal{B}[u](z) = 0$.
- With our model, replace u by f_θ and define

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \Big(\mathcal{N}[\, f_\theta \,](x_j)\Big)^2, \quad \mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_k \Big(\mathcal{B}[\, f_\theta \,](z_k)\Big)^2.$$

- N_{col}: number of collocation points in the domain (usually sampled randomly or on a grid inside the physical domain).
- N_{bc}: number of boundary (or initial) condition points (sampled on the boundary/initial surface).
- If measurements (x_i, y_i) exist, add the usual data MSE.

Schematic of a PINN



Example: 1D Poisson problem

- On $x \in [0, 1]$: $u''(x) = q(x)$, where $q(x) = -\alpha\, \pi^2 \sin(\pi x)$, with Dirichlet $u(0) = u(1) = 0$.

Example: 1D Poisson problem

- On $x \in [0, 1]$: $u''(x) = q(x)$, where $q(x) = -\alpha\,\pi^2 \sin(\pi x)$, with Dirichlet $u(0) = u(1) = 0$.
- Use $f_\theta(x)$ for the solution; via autodiff compute $f_\theta''(x)$.

Example: 1D Poisson problem

- On $x \in [0, 1]$: $u''(x) = q(x)$, where $q(x) = -\alpha\,\pi^2 \sin(\pi x)$, with Dirichlet $u(0) = u(1) = 0$.
- Use $f_\theta(x)$ for the solution; via autodiff compute $f''_\theta(x)$.
- Physics residual: $r_\theta(x) = f''_\theta(x) - q(x)$.

Example: 1D Poisson problem

- On $x \in [0,1]$: $u''(x) = q(x)$, where $q(x) = -\alpha\,\pi^2 \sin(\pi x)$, with Dirichlet $u(0) = u(1) = 0$.
- Use $f_\theta(x)$ for the solution; via autodiff compute $f_\theta''(x)$.
- Physics residual: $r_\theta(x) = f_\theta''(x) - q(x)$.
- Loss terms:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \Big(f_\theta''(x_j) - q(x_j)\Big)^2, \qquad \mathcal{L}_{\text{bc}} = \tfrac{1}{2}\Big(f_\theta(0)\Big)^2 + \tfrac{1}{2}\Big(f_\theta(1)\Big)^2.$$

Example: 1D Poisson problem

- On $x \in [0,1]$: $u''(x) = q(x)$, where $q(x) = -\alpha\, \pi^2 \sin(\pi x)$, with Dirichlet $u(0) = u(1) = 0$.
- Use $f_\theta(x)$ for the solution; via autodiff compute $f''_\theta(x)$.
- Physics residual: $r_\theta(x) = f''_\theta(x) - q(x)$.
- Loss terms:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{col}}} \sum_j \Big(f''_\theta(x_j) - q(x_j) \Big)^2, \qquad \mathcal{L}_{\text{bc}} = \tfrac{1}{2}\Big(f_\theta(0)\Big)^2 + \tfrac{1}{2}\Big(f_\theta(1)\Big)^2.$$

- (Optional) With noisy samples (x_i, y_i), include $\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_i \Big(f_\theta(x_i) - y_i \Big)^2$.

Training a PINN

- Sample domain collocation points $\{x_j\}$ and boundary points $\{z_k\}$ (and measurement points if available).

Training a PINN

- Sample domain collocation points $\{x_j\}$ and boundary points $\{z_k\}$ (and measurement points if available).
- Forward pass with f_θ: evaluate residuals and all loss terms.

Training a PINN

- Sample domain collocation points $\{x_j\}$ and boundary points $\{z_k\}$ (and measurement points if available).
- Forward pass with f_θ: evaluate residuals and all loss terms.
- Combine: $\mathcal{L} = \mathcal{L}_{\text{phys}} + \mathcal{L}_{\text{bc}} + \mathcal{L}_{\text{data}}$.

Training a PINN

- Sample domain collocation points $\{x_j\}$ and boundary points $\{z_k\}$ (and measurement points if available).
- Forward pass with f_θ: evaluate residuals and all loss terms.
- Combine: $\mathcal{L} = \mathcal{L}_{\text{phys}} + \mathcal{L}_{\text{bc}} + \mathcal{L}_{\text{data}}$.
- Backpropagate $\nabla_\theta \mathcal{L}$ and update θ with (stochastic) gradient descent or variants (e.g., Adam optimizer).

Training a PINN

- Sample domain collocation points $\{x_j\}$ and boundary points $\{z_k\}$ (and measurement points if available).
- Forward pass with f_θ: evaluate residuals and all loss terms.
- Combine: $\mathcal{L} = \mathcal{L}_{\text{phys}} + \mathcal{L}_{\text{bc}} + \mathcal{L}_{\text{data}}$.
- Backpropagate $\nabla_\theta \mathcal{L}$ and update θ with (stochastic) gradient descent or variants (e.g., Adam optimizer).
- Monitor physics and boundary losses; visualize $f_\theta(x)$ against a reference when available.

Next: PINN code demo (1D Poisson)

- Define f_θ (MLP) and the known source $q(x)$.

Next: PINN code demo (1D Poisson)

- Define f_θ (MLP) and the known source $q(x)$.
- Build MSE-style losses for the ODE residual and boundary conditions using f_θ.

Next: PINN code demo (1D Poisson)

- Define f_θ (MLP) and the known source $q(x)$.
- Build MSE-style losses for the ODE residual and boundary conditions using f_θ.
- Train with gradient descent; plot predictions and the loss over iterations.

Code example

Code example

When should you consider PINNs?

- **Scarce or noisy data:** PINNs shine when measurement data is limited but governing equations are well known.
- **Hybrid scenarios:** Combine experimental data with simulation-based physics constraints.
- **Forward & inverse problems:**
  - Forward: approximate the solution of DEs directly.
  - Inverse: infer hidden parameters (e.g., material properties) from partial observations.

PINNs in research

- **Engineering:** fluid dynamics, structural mechanics, heat transfer, power systems.
- **Physics:** quantum mechanics, plasma physics, electromagnetics.
- **Biology/medicine:** blood flow modeling, tumor growth, protein folding.
- **Energy systems:** battery modeling, renewable energy integration, smart grids.
- Rapidly growing community: papers, workshops, benchmark suites, open-source libraries (DeepXDE, JAX/torch PINN frameworks).

Thank you & Resources

Thank you!

Source code and presentation available at:

`github.com/emiresenov/SESBC-Webinar`