Intuitive Explanation of Physics-Informed Neural Networks (PINNs)

For Finite Element Experts

Core Concept: PINNs as Constrained Optimizers

PINNs train neural networks to satisfy:

- Governing PDE equations
- Boundary conditions (BCs)
- Initial conditions (ICs)

through a **composite loss function**. The network learns by minimizing violations of these physical constraints.

Key Components of PINN Loss Function

The total loss ($\mathcal{L}_{\text{total}}$) combines multiple constraint violations:

$$\mathcal{L}_{\text{total}} = \underbrace{w_{\text{PDE}}\mathcal{L}_{\text{PDE}}}_{\text{PDE residual}} + \underbrace{w_{\text{BC}}\mathcal{L}_{\text{BC}}}_{\text{Boundary violation}} + \underbrace{w_{\text{IC}}\mathcal{L}_{\text{IC}}}_{\text{Initial condition}} \tag{1}$$

1. PDE Loss ($\mathcal{L}_{\text{PDE}}$)

Measures how well the solution satisfies the governing PDE at *collocation points* inside the domain:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{i=1}^{N_{\text{PDE}}} \|\mathcal{N}[u_\theta(\mathbf{x}_i)] - f(\mathbf{x}_i)\|^2 \tag{2}$$

- $\mathcal{N}$: Differential operator (e.g., ∇^2, $\frac{\partial}{\partial t}$)
- u_θ: NN prediction with parameters θ
- f: Source term
- Derivatives computed via **automatic differentiation**

2. Boundary Condition Loss ($\mathcal{L}_{\text{BC}}$)

Enforces boundary constraints at sampled boundary points:

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{j=1}^{N_{\text{BC}}} \|B[u_\theta(\mathbf{x}_j)] - g(\mathbf{x}_j)\|^2 \tag{3}$$

- B: Boundary operator (Dirichlet, Neumann, etc.)
- g: Prescribed boundary value

3. Initial Condition Loss ($\mathcal{L}_{\text{IC}}$)

For time-dependent problems, enforces initial state:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{k=1}^{N_{\text{IC}}} \|u_\theta(\mathbf{x}_k, t=0) - h(\mathbf{x}_k)\|^2 \tag{4}$$

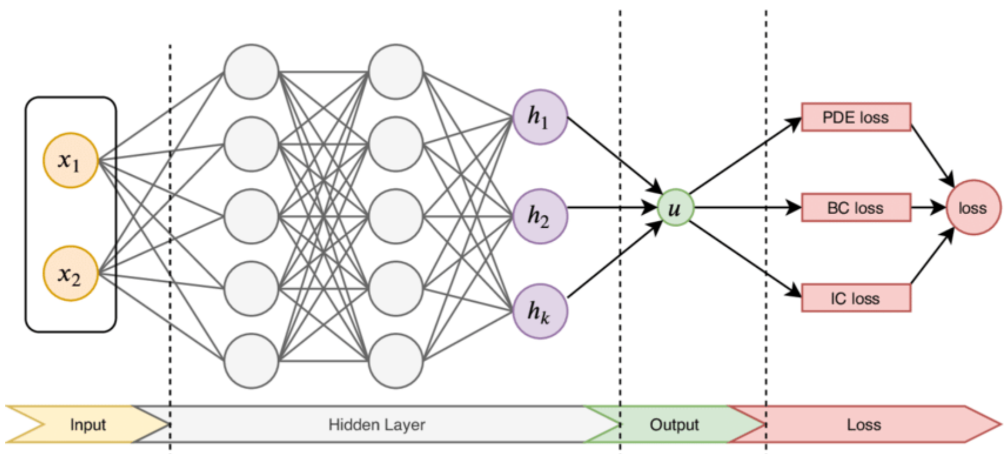


Figure 1: PINN architecture showing loss components

Critical Insight: Soft vs. Hard Constraints

Why Derivatives Work in PINNs

- **Automatic differentiation** computes *exact derivatives* of u_θ (not finite differences)
- PDE loss directly compares NN's derivatives to physical laws
- Gradient descent propagates PDE violation errors backward through:

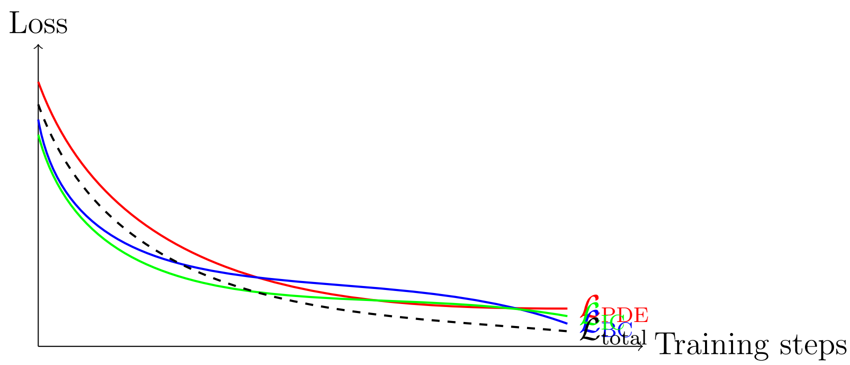
$$\frac{\partial \mathcal{L}_{\text{PDE}}}{\partial \theta} = \frac{2}{N} \sum \left(\mathcal{N}[u_\theta] - f \right) \cdot \frac{\partial \mathcal{N}[u_\theta]}{\partial \theta} \tag{5}$$

- The NN **simultaneously** learns function values *and* derivatives

Soft Constraints (Standard PINNs)	**Hard Constraints**
Constraints enforced via loss terms	Constraints *built into* network architecture
BCs/ICs appear as penalty terms	BCs/ICs are *exactly satisfied* by construction
u_θ can violate constraints during training	No constraint violation possible
Weights (w_{BC}, w_{IC}) require tuning	No weighting needed
Easier to implement	Requires specialized architecture

Table 1: Constraint implementation strategies

Training Dynamics Visualization



Key Advantages for FEM Experts

- **No meshing**: Collocation points can be randomly sampled
- **Unified framework**: Handles both forward and inverse problems
- **Adaptive refinement**: Loss guides where to add collocation points
- **High-dimensional problems**: Avoids curse of dimensionality

Conclusion: Loss as Physics Interpreter

The PINN loss function acts as a *physics interpreter* that:

1. Translates PDEs into optimization constraints
2. Quantifies violations of physical laws

3. Balances multiple constraints through weighting
4. Provides training signals via automatic differentiation

The network becomes a **continuous function approximator** that satisfies physics in the weak sense defined by the composite loss.