PINNs Architecture for Force Equivalence Procedure

Susmita khatun

December 2024

Equations Overview

1. **Resistance (Res):**

$$\text{Res} = \frac{\rho}{2\pi R^2} \cdot \left(\theta - \left(\frac{th}{R} - \cos\theta \right) \cdot \frac{1}{\sqrt{1 - \left(\frac{th}{R} - \cos\theta\right)^2}} \cdot \arctan\left(\sqrt{\frac{1 - \left(\frac{th}{R} - \cos\theta\right)}{1 + \left(\frac{th}{R} - \cos\theta\right)}} \cdot \tan\frac{\theta}{2} \right) \right).$$

2. **Voltage Dynamics:**

$$V = \text{Res} \cdot \frac{4\pi T d^2}{V} \left(-\sin\theta \cdot \frac{d\theta}{dt} - \frac{\cos\theta}{V} \cdot \frac{dV}{dt} \right).$$

3. **Radius Relation:**

$$R = \frac{d}{\sin\theta}.$$

Proposed PINNs Architecture

The goal of the architecture is to predict θ given voltage (V) and other parameters (ρ, T, d, th) while ensuring that the predictions respect the above equations.

Input Features

- V (voltage): Scalar or time series input.
- ρ, T, d, th: Material properties/constants.
- Initial conditions for $\theta, \frac{d\theta}{dt}, \frac{dV}{dt}$.

Output

Predicted θ at time t.

Loss Function

The loss function combines physics and data losses:

$$\mathcal{L}_{\text{physics}} = \left| V - \text{Res} \cdot \frac{4\pi T d^2}{V} \left(-\sin\theta \cdot \frac{d\theta}{dt} - \frac{\cos\theta}{V} \cdot \frac{dV}{dt} \right) \right|^2 + \left| R - \frac{d}{\sin\theta} \right|^2.$$

$$\mathcal{L}_{\text{data}} = \|\theta_{\text{predicted}} - \theta_{\text{observed}}\|^2.$$

$$\mathcal{L} = \mathcal{L}_{\text{physics}} + \lambda \mathcal{L}_{\text{data}}.$$

Architecture Design

Layer Type	**Input Shape**	**Output Shape**	**Activation**	**Comments**
Input Layer	$[V, \rho, T, d, th, \frac{d\theta}{dt}, \frac{dV}{dt}]$	$[n]$	None	$Capturessysteminputs.$
Dense Layer 1	$[n]$	[128]	Tanh	$Encodesnon-linearrelationships.$
Dense Layer 2	[128]	[128]	Tanh	$Addsdepthforfeatureextraction.$
Dense Layer 3	[128]	[64]	Tanh	$Refinesfeaturerepresentation.$
Dense Layer 4	[64]	[1]	Linear	$Outputspredicted\theta.$

Training Process

1. **Initialization:**
   - Normalize inputs and scale outputs.
   - Initialize network weights with small random values.
2. **Optimization:**
   - Use Adam optimizer with a small learning rate (e.g., 10^{-4}).
   - Implement gradient clipping to handle stiff equations.
3. **Regularization:**
   - Penalize large weight magnitudes to improve generalization.
4. **Validation:**
   - Verify physical consistency of predictions (θ, R, Res).

Extensions

1. **Dynamic Learning Rate:** Adjust λ during training based on the magnitude of the physics error.
2. **Neural ODE Extension:** Integrate $\frac{d\theta}{dt}$ and $\frac{dV}{dt}$ using Neural ODE layers for better time-dependent modeling.

3. **Uncertainty Quantification:** Use Bayesian PINNs to provide confidence intervals for θ.

4. **Dimensionality Reduction:** Employ autoencoders to reduce the complexity of high-dimensional inputs.