Neural Network as a Friction Coefficient Model

January, 2025

Author:
Pawel Wiktor

Email:
pawelwiktor.kontakt@gmail.com

GitHub:
https://github.com/pawelwiktor-github

Table of Contents

1 Creating Neural Network as Friction Coefficient Model

In this task, a *Reaction Pendulum* model was developed to account for the nonlinearities associated with friction. The model is based on a simplified neural network structure, whose parameters have been optimized to minimize the error between experimental data and simulation results. A three-layer neural network with a sufficient number of neurons to represent nonlinearities in the system was used.

1.1 Theoretical Introduction

1.1.1 Components of Neural Networks

Neural networks consist of many elements that work together to enable learning from data and making complex predictions or classifications. Here is a description of the main components of the network:

- **Weights:** Weights are fundamental elements of neural networks, corresponding to the strength of connections between neurons in different layers. Each connection between neurons transmits a signal that is multiplied by the weight of that connection. Weights are adjusted during the learning process of the network, allowing the model to learn data representations and perform the assigned functions. Weight initialization is an initial step that can have a significant impact on the effectiveness and speed of learning.
- **Activation Function:** The activation function in a neuron decides whether and how strongly a neuron should "activate,"i.e., pass the signal further. Activation functions introduce nonlinearity into the network processing process, which is crucial for the network's ability to learn and model complex patterns and dependencies in data. Without activation functions, a neural network would only be a linear transformer, limited to simple tasks. Popular activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh (hyperbolic tangent), and softmax.
- **Loss Function:** The loss function, also known as the cost function, measures the difference between the model's predictions and the actual labels or values. It is a key element of the learning process because it provides information on how well the model is performing the task it is supposed to do. The loss function is used during propagation, where the network aims to minimize this loss by adjusting weights. Popular loss functions include mean squared error (MSE) for regression tasks and cross-entropy for classification tasks.
- **Weight Optimizer:** The optimizer is a mechanism used to update weights in the network based on the gradient calculations of the loss function. Its task is to find optimal weight values that minimize the loss function. There are many optimization algorithms, each with different properties and suitable for different types of problems. The most commonly used optimizers include SGD (Stochastic Gradient Descent), Adam (Adaptive Moment Estimation), and RMSprop (Root Mean Square Propagation). These algorithms differ in how they adjust the learning rate and the application of various techniques, such as momentum, which help accelerate learning and avoid the traps of local minima of the loss function.

1.1.2 Main Stages of the Neural Network Learning Process:

Each step of the Neural Network Learning Process, from network initialization to data propagation to updating weights, plays a key role in achieving high-quality results and the model's ability to generalize.

- **Network Initialization:** Choosing the network architecture (determining the number of layers, the number of neurons in each layer, and the activation functions) and initialization of weights and biases (initial weights and biases are usually initialized randomly, although there are more advanced initialization methods that can accelerate learning).
- **Data Division:** Data are divided into sets: training (to train the model), validation (for tuning hyperparameters and avoiding overfitting), and test (for final performance assessment of the model).
- **Forward Propagation:** Input data is passed through the network from input to output. At each neuron, a weighted sum of input signals is calculated and an activation function is applied to generate an output signal to the next layer.

- **Loss Calculation:** After processing the input data through the network and receiving the result, the loss function (e.g., mean squared error or cross-entropy) is calculated, which measures the difference between the predicted result and the actual one.
- **Backpropagation:** The error calculated at the output of the network is then propagated back through the network, from output to input. During this process, for each weight in the network, its derivative of the loss function (gradient of the loss function) is calculated, which indicates how changing that weight affects the final error. This allows determining the direction and magnitude in which each weight should be changed to minimize the error.
- **Weight Update:** Based on the calculated gradients and using a specified learning rate, weights are updated in such a way as to minimize the loss function. Here, the opposite directions of the derivatives are used to reduce the value of the loss function (i.e., minimize the error).
- **Repeating the Process:** The process from forward propagation to weight update is repeated multiple times (in so-called epochs) on different sets of input data. In each epoch, the network aims to learn to reduce the error for the data presented to it, leading to the gradual improvement of the model.
- **Validation and Testing:** After training is completed, the model is tested on validation and test data to assess its performance and generalization on data it has not seen during training.

1.1.3 Friction force

The frictional force between two rotating bodies can be described as a graph of the dependence on the relative velocity of the rotating body. Assumed to be the sum of Stribeck, Coulomb, and viscous components.

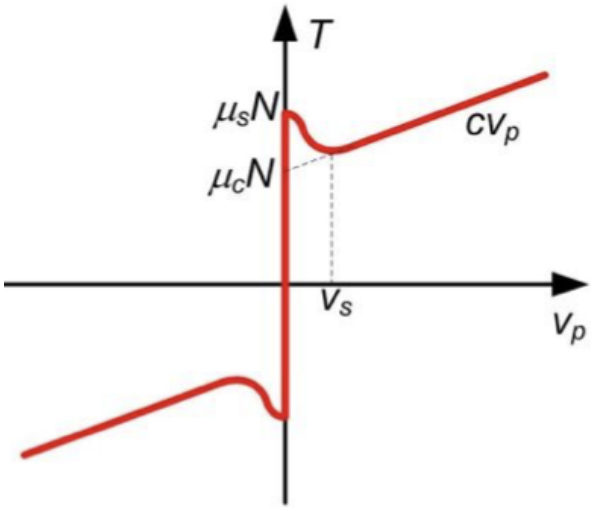

Fig. 1. Friction force

The Stribeck friction, is the negatively sloped characteristics taking place at low velocities. The Coulomb friction, T_c, results in a constant force at any velocity. The viscous friction, T_v, opposes motion with the force directly proportional to the relative velocity. The sum of the Coulomb and Stribeck frictions at the vicinity of zero velocity is often referred to as the breakaway friction. The friction is approximated with the following equations:

$$T(v) = \left(T_c + (T_s - T_c) \left(\frac{v}{v_s} \right)^2 + T_v |v| \right) \text{sign}(v) \tag{1}$$

where:

- T_s – static friction force
- T_c – Coulomb friction force

- T_v – viscous friction force
- v_s – Stribeck velocity

2 Experiments

As part of the tests conducted, the pendulum was swung successively by angles of 45 degrees and 90 degrees in both directions (left and right), which made it possible to obtain four sets of experimental data describing the position and velocity of the pendulum. The collected data were then used to carry out the process of optimizing the model describing the dynamics of the studied system.

2.1 Chosing activation function

Based on iterations through each dataset from Reaction Pendulum, the tests were conducted to choose best activation function for same friction coefficient model.

Firstly the number of neurons, friction model and pendulum model with friction was defined (1).

```
%% Creating NN

% Neural network parameters
num_neurons = 3; % Number of neurons

% Friction model
model_friction = @(p, vel) ...
    sign(vel) .* (p(1) + (p(2) - p(1)) * exp(-(abs(vel) / p(3)).^p(4))) + ... % Dry friction and the Stribeck effect
    p(5) * vel; % Viscous friction

% Pendulum model including friction
model_function = @(p, vel, pos) ...
    p(6) * pos + p(7) * vel + model_friction(p(1:5), vel);
```

Code snippet 1. NN configuration definition

As a validation criteria (quality indicators), mean squared error (3) and mean absolute error (4) were calculated.

$$\text{error} = y - y_{nn} \tag{2}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{error})^2 \tag{3}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{error}| \tag{4}$$

where:

- y – experimental position value
- y_{nn} – predicted position value by NN
- n – total number of samples
- i – actual dataset

2.1.1 Sigmoid activation

The sigmoid activation function is defined as:

$$\text{activation} = \Theta(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function is smooth and continuously differentiable, making it suitable for gradient-based optimization methods like backpropagation. Since friction in a reaction pendulum is often nonlinear, the sigmoid function helps introduce non-linearity to the neural network, allowing it to learn complex relationships between input forces and friction coefficients. The output of the sigmoid function lies within a fixed range, preventing extreme or unbounded values, which is useful when modeling friction coefficients. The sigmoid function's output can be interpreted as a probability, which can be helpful in classification-like decisions within the friction modeling process. The sigmoid function provides smooth transitions between low and high values, which is beneficial when modeling dynamic friction changes in reaction pendulum systems.

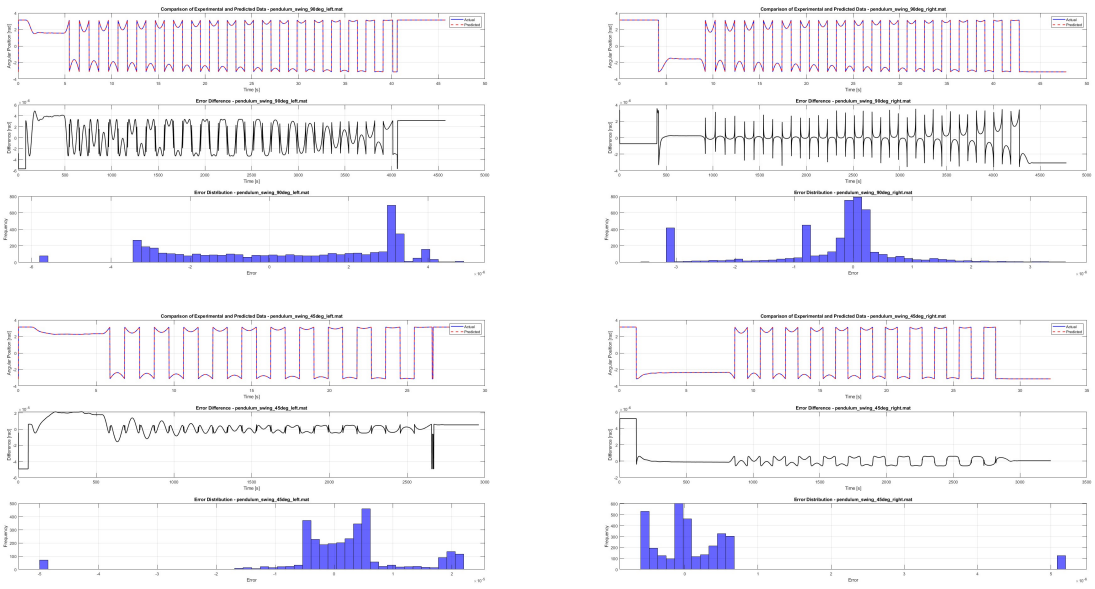

Fig. 2. Validation for Sigmoid activation function

2.1.2 Tanh activation

The hyperbolic tangent activation function is defined as:

$$\text{activation} = \Theta(x) = \tanh(x)$$

Unlike the sigmoid function, which outputs values in the range $(0, 1)$, the tanh function provides an output in the range $(-1, 1)$. This helps in making the optimization process more stable since the activations are zero-centered. The gradient of the tanh function is steeper compared to the sigmoid function, which helps in faster learning and reduces the vanishing gradient problem to some extent. The friction coefficient in a reaction pendulum exhibits nonlinear characteristics. The tanh function enables the neural network to model complex relationships between forces and friction more effectively. Since the tanh function is continuous and differentiable, it ensures that small changes in input produce smooth variations in the output. This is particularly useful when modeling friction forces in dynamic systems. The tanh function is symmetric about the origin, making it well-suited for cases where input values can be both positive and negative, such as in the bidirectional forces affecting the reaction pendulum.

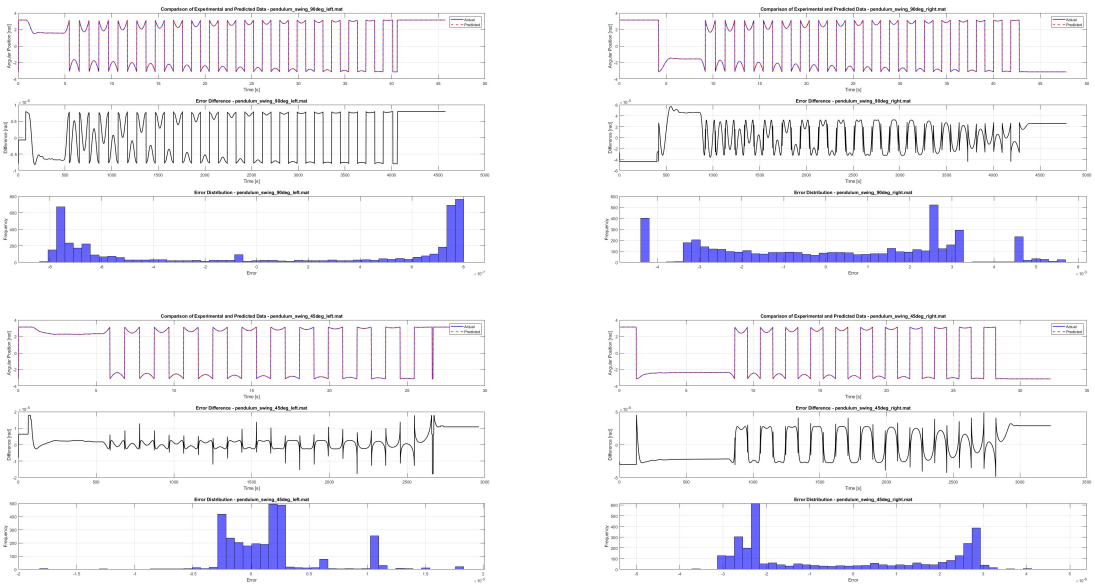

Fig. 3. Validation for Hyperbolic Tangent activation function

2.1.3 ReLu activation

The Rectified Linear Unit (ReLU) activation function is defined as:

$$\text{activation} = \Theta(x) = \max(0, x)$$

The ReLU function is computationally simple as it only involves a comparison operation, making it faster to evaluate compared to sigmoid and tanh functions. Unlike sigmoid and tanh, which saturate at extreme values leading to very small gradients, ReLU maintains a strong gradient for positive inputs, enabling faster learning in deep networks. Since ReLU outputs zero for negative inputs, it introduces sparsity in the network, meaning some neurons remain inactive. This can improve generalization and efficiency in friction modeling. Friction in a reaction pendulum is highly nonlinear, and ReLU helps capture these relationships effectively by allowing non-linearity while avoiding saturation effects. ReLU enables deeper neural networks to learn complex patterns more effectively than sigmoid or tanh, which tend to struggle with vanishing gradients in deep architectures.

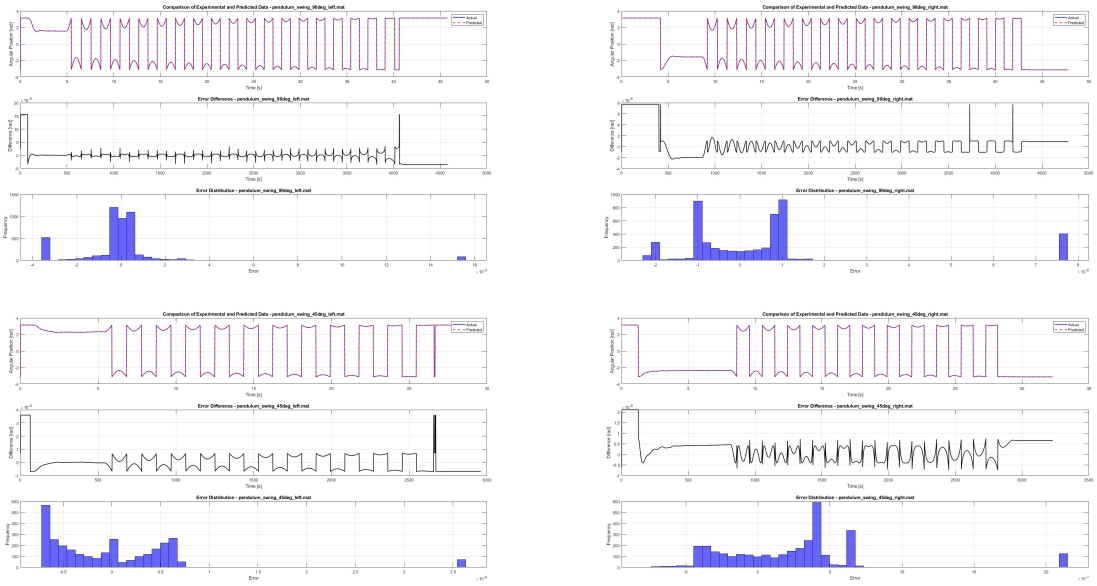

Fig. 4. Validation for Rectified Linear Unit activation function

2.1.4 Conclusions

The average quality indicators were calculated for combined 4 experimental datasets (Table. 1).

Table 1. Average Quality indicator

Activation Function	Average MSE	Average MAE
Sigmoid	$3.41 \cdot 10^{-11}$	$2.69 \cdot 10^{-6}$
Hyperbolic Tangent	$1.95 \cdot 10^{-10}$	$7.55 \cdot 10^{-6}$
Rectified Linear Unit	$1.53 \cdot 10^{-10}$	$3.36 \cdot 10^{-6}$

Based on the evaluation of the Mean Squared Error (MSE) and Mean Absolute Error (MAE) across four experimental datasets, the impact of different activation functions—Sigmoid, Tanh, and ReLU—on the performance of the Neural Network friction coefficient model was analyzed.

Sigmoid Activation:

- Achieved the lowest MSE (3.41×10^{-11}) and lowest MAE (2.69×10^{-6}), indicating strong performance in capturing friction dynamics.
- However, due to the sigmoid function's tendency to saturate for large positive or negative inputs, it may limit the ability of deeper networks to learn effectively.
- Suitable for scenarios where small input variations significantly impact the friction coefficient.

Tanh Activation:

- Exhibited the highest MSE (1.95×10^{-10}) and highest MAE (7.55×10^{-6}), indicating lower accuracy compared to the other activation functions.
- While tanh avoids the output bias issue of sigmoid by being zero-centered, it still suffers from gradient vanishing problems, which can hinder training efficiency.
- Recommended for datasets with significant negative values or symmetric input distributions.

ReLU Activation:

- Provided a balanced performance with MSE (1.53×10^{-10}) and MAE (3.36×10^{-6}), slightly worse than sigmoid but better than tanh.
- Due to its ability to mitigate the vanishing gradient problem and promote sparsity, ReLU is highly efficient for deeper networks.
- Particularly effective for complex, nonlinear friction relationships with larger datasets.

Considering both the error metrics and computational efficiency, the Sigmoid activation proved to be the most accurate in this specific friction modeling task. However, for larger and more complex networks, ReLU may be a better choice due to its better scalability and non-saturating nature. Tanh was the least effective for this particular dataset but could still be useful in scenarios requiring symmetric output distributions.

Further optimization could be explored by testing variants such as Leaky ReLU or combining different activation functions across layers to maximize predictive performance.

2.2 Chosing optimization algorithm

Based on previous conclusions, the Sigmoid activation function were chosen. Using the same number of neurons, and model equations with Sigmoid activation, four optimization algorithm were tested on previously mentioned same datasets.

For each optimization algorithm, the same cost function were defined (Equation. 5).

$$\text{cost\_function}(p) = \sum \left(\text{position} - \text{model\_function}(p, \text{velocity}, \text{position})\right)^2 \tag{5}$$

As a validation criteria (quality indicators), mean squared error (3) and mean absolute error (4) were calculated.

2.2.1 Quasi-Newton

The quasi-Newton method uses second-order derivative approximations, leading to faster convergence compared to first-order methods. Well-suited for optimizing differentiable cost functions like the Mean Squared Error (MSE) in NN-based friction modeling. Provides a structured update of parameters, ensuring stability when fine-tuning the friction coefficient model.

Table 2. Configuration Parameters for `fminunc` (Quasi-Newton)

Parameter	Value
Display	`iter`
Algorithm	`quasi-newton`

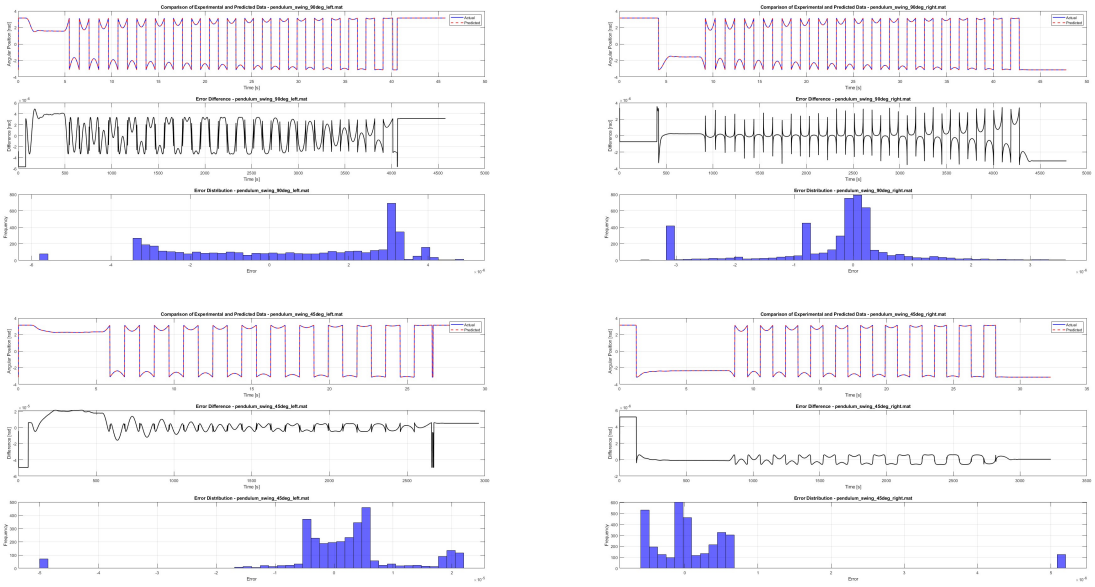

Fig. 5. Validation for Quasi-Newton optimization algorithm

2.2.2 Genetic-Algorithm

Genetic-Algorithm avoids local minima by searching for a global optimum, making it effective for highly nonlinear friction models. Unlike gradient-based methods, GA can optimize functions where derivatives are unavailable or noisy. The stochastic nature of GA helps in handling measurement noise in experimental friction datasets.

Table 3. Configuration Parameters for Genetic Algorithm (GA)

Parameter	Value
Display	`iter`
Population Size	50
Max Generations	100

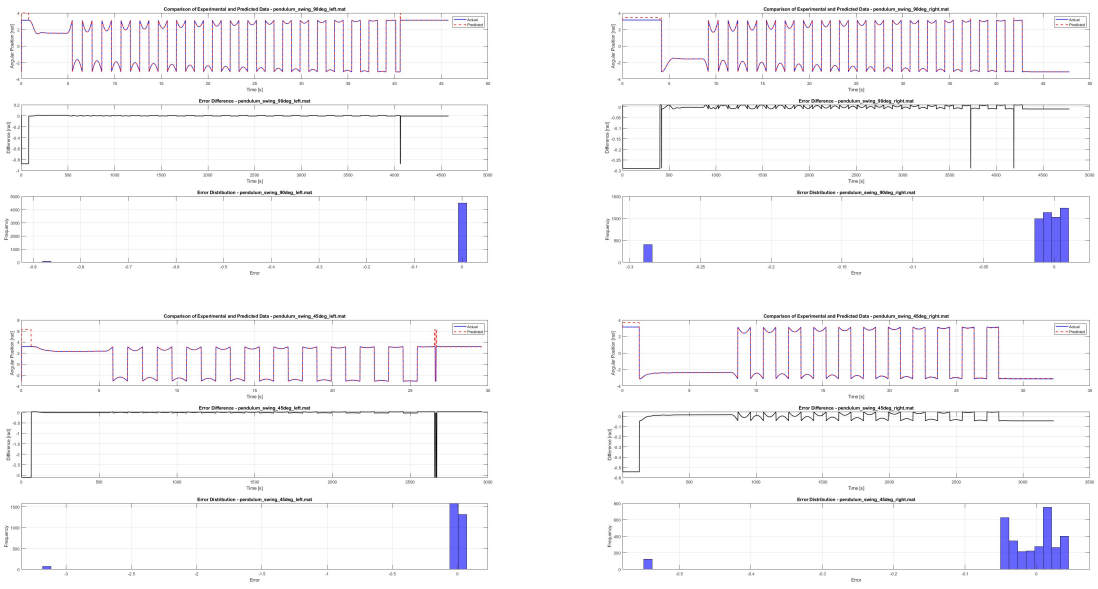

Fig. 6. Validation for Genetic-Algorithm optimization algorithm

2.2.3 Simulated Annealing

Simulated Annealing employs probabilistic acceptance of worse solutions to escape local minima, beneficial in friction models with complex landscapes. Effective even with limited data, making it a practical choice for experimental reaction pendulum measurements. Can be adapted to different types of cost functions, making it a versatile choice for NN-based friction modeling.

Table 4. Configuration Parameters for Simulated Annealing

Parameter	Value
Display	`iter`
Max Iterations	100

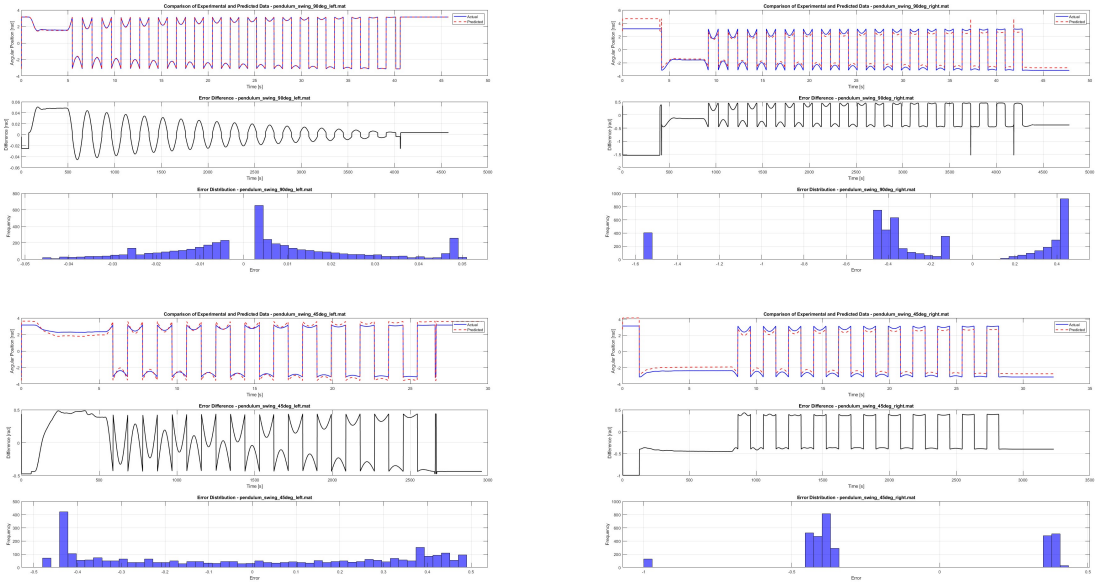

Fig. 7. Validation for Simulated Annealing optimization algorithm

2.2.4 Particle Swarm Optimization

Particle Swarm Optimization performs well in optimizing large parameter spaces, useful for tuning complex NN models. The swarm-based approach balances exploration and exploitation, leading to quick convergence compared to GA. Can be implemented in parallel computing environments for faster performance in real-time friction modeling.

Table 5. Configuration Parameters for Particle Swarm Optimization (PSO)

Parameter	Value
Display	`iter`
Swarm Size	30
Max Iterations	100

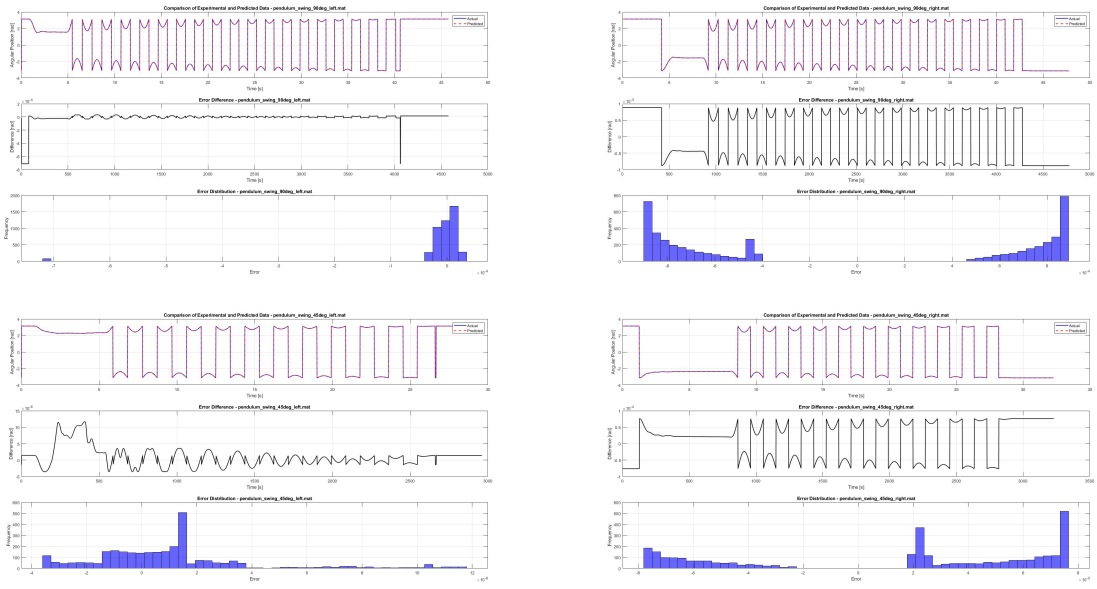

Fig. 8. Validation for Particle Swarm Optimization algorithm

2.2.5 Conclusions

The average quality indicators were calculated for combined 4 experimental datasets (Table. 6).

Table 6. Average Quality indicator

Optimization Algorithm	Average MSE	Average MAE
Quasi-Newton	$3.41 \cdot 10^{-11}$	$2.69 \cdot 10^{-6}$
Genetic-Algorithm	$6.59 \cdot 10^{-2}$	$4.68 \cdot 10^{-2}$
Simulated Annealing	$5.08 \cdot 10^{-2}$	$1.65 \cdot 10^{-1}$
Particle Swarm Optimization	$1.51 \cdot 10^{-7}$	$2.10 \cdot 10^{-4}$

Based on the evaluation of Mean Squared Error (MSE) and Mean Absolute Error (MAE) across four experimental datasets, the performance of different optimization algorithms: Quasi-Newton, Genetic Algorithm (GA), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) were analyzed.

Quasi-Newton Optimization

- Achieved the lowest MSE (3.41×10^{-11}) and lowest MAE (2.69×10^{-6}), indicating high accuracy in tuning the friction model parameters.
- The gradient-based approach ensures fast convergence, making it suitable for differentiable cost functions like MSE.

- However, it may struggle with non-smooth and highly nonlinear friction models.

Genetic Algorithm (GA)

- Exhibited higher MSE (6.59×10^{-2}) and higher MAE (4.68×10^{-2}), indicating lower accuracy compared to Quasi-Newton.
- The global search capability of GA helps avoid local minima, making it effective for highly nonlinear friction models.
- However, GA requires longer computation time and does not guarantee optimal results in every run.

Simulated Annealing (SA)

- Produced an MSE of 5.08×10^{-2} and MAE of 1.65×10^{-1}, performing slightly worse than GA.
- Works well for escaping local minima, which can be beneficial for friction models with complex cost function landscapes.
- However, the stochastic nature of SA can lead to inconsistent results, depending on initialization and cooling parameters.

Particle Swarm Optimization (PSO)

- Achieved an MSE of 1.51×10^{-7} and MAE of 2.10×10^{-4}, performing better than GA and SA but slightly worse than Quasi-Newton.
- Balances exploration and exploitation, allowing for faster convergence than GA.
- Works well in high-dimensional search spaces, making it effective for tuning NN friction coefficient models.
- The primary limitation is that convergence speed depends on hyperparameter tuning (swarm size, inertia weight, etc.).

Considering both error metrics and computational efficiency: the Quasi-Newton proved to be the most accurate and computationally efficient for this specific friction modeling task. PSO performed well and is a good alternative when dealing with high-dimensional parameter spaces. GA and SA are recommended when the friction model is highly nonlinear and prone to local minima, but their performance is less consistent.

For further improvements, a hybrid optimization approach (e.g., combining GA with gradient-based fine-tuning) or testing additional algorithms (such as Adam or L-BFGS) could be explored.

3 Summary

This study presented a Neural Network-based Friction Coefficient Model for a Reaction Pendulum, evaluating different activation functions and optimization algorithms. Based on the results obtained from Mean Squared Error (MSE) and Mean Absolute Error (MAE) across four experimental datasets, the following conclusions were drawn.

The impact of three different activation functions: Sigmoid, Tanh, and ReLU were analyzed. Sigmoid activation achieved the lowest MSE and MAE, making it the best choice for accuracy. ReLU performed well but was slightly less accurate than Sigmoid. It remains a viable alternative for deeper networks. Tanh showed the highest error values, making it the least effective activation function in this context.

Considering accuracy and computational efficiency, Sigmoid was the best activation function for this specific friction coefficient model.

The effectiveness of four different optimization algorithms: Quasi-Newton, Genetic Algorithm (GA), Simulated Annealing (SA), and Particle Swarm Optimization (PSO) were assessed. Quasi-Newton provided the lowest error values, proving to be the most accurate and computationally efficient. PSO performed well and is a good alternative, especially for high-dimensional search spaces. GA and SA were less consistent and had significantly higher error values. They are better suited for highly nonlinear models prone to local minima.

Considering both error metrics and computational efficiency, Quasi-Newton was the best optimization algorithm, with PSO being a viable alternative.

This study demonstrated the effectiveness of Neural Networks in modeling friction coefficients in a Reaction Pendulum. The proposed approach successfully captured nonlinear behaviors and provided an efficient framework for optimization. Future improvements can further refine the model for broader applications in dynamics and control systems.