NARX Velocity Observer for Reaction Pendulum

January, 2025

Author:
Pawel Wiktor

Email:
pawelwiktor.kontakt@gmail.com

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

Table of Contents

1 Neural Network as a Reaction Pendulum Velocity Observer

1.1 Theoretical Introduction - Velocity Observer

In dynamic system, such as a Reaction Pendulum, full observation of the system's state is not always feasible due to hardware limitations, costs, or diffuculties in directly measuring all relevant physical quantities. On such example is the pendulum's velocity, which can be challenging to measure in practice without specialized sensors. In such cases, observers are used to estimate the missing states based on available measurements of other variables and the mathematical mode of the system.

State observer
A velocity observer is a specific case of a general state observer. A state observer is an algorithm (or mathematical model) that reconstructs unobservable states of a dynamic system based on:

- **System inputs (e.g., control signals)** - describing the external signals influecing the system.
- **Measurements of part of the system state** - in this case, the pendulum's angular position and the disk's angular velocity
- **System model** - utilizing equations that describe the system's dynamics

Neural Networks as Observers
In traditional approaches, observers are based on differential equations or discrete mathematical models, such as Luenberger observers or Kalman filters. An alternative solution is to use neural networks, which offer the following adventages:

- **Nonlinear modeling** - they can approximate nonlinear realtionships in systems where classical models may be challenging to formulate
- **Data - driven learning** - they learn from input and output data, eliminating the need for a precise mathematical mode of the system
- **Time - delay handling** - they can account for temporal delays in the data through appropriate architectures, scuh as NARX (Nonlinear Autoregressive with External Input)

1.2 NARX Network

NARX (Nonlinear autoregressive with external input) networks can learn to predict one time series given past values of the same time series, the feedback input, and another time series called the external (or exogenous) time series.

$$narxnet(inputDelays, feedbackDelays, hiddenSizes, feedbackMode, trainFcn) \tag{1}$$

takes these arguments:
$inputDelays$ - row vector of increasing 0 or positive input delays
$feedbackDelays$ - row vector of increasing 0 or positive feedback delays
$hiddenSizes$ - row vector of one more hidden layer sizes
$feedbackMode$ - type of feedback
$trainFcn$ - backpropagation training function

For estimating the pendulum's velocity, the NARX network learns from historical data:

- **Inputs:** The control signal, the pendulum's angular position and the disk's angular velocity
- **Output:** The pendulum's velocity

Advantages and Applications of the Velocity Observer:

- **Avoiding costly hardware:** The observer eliminates the need for a velocity sensor, which can reduce costs and increase system reliability.
- **Improved noise tolerance**: By leveraging multiple inputs and nonlinear relationships, NARX networks can function effectively in the presence of disturbances and measurement noise.
- **Control applications:** Estimating the pendulum's velocity can be used in feedback loops for precise system control.

1.2.1 NARX network structure and configuration

In this section, an example structure of the NARX network will be presented, for which the first stage of training will be performed.

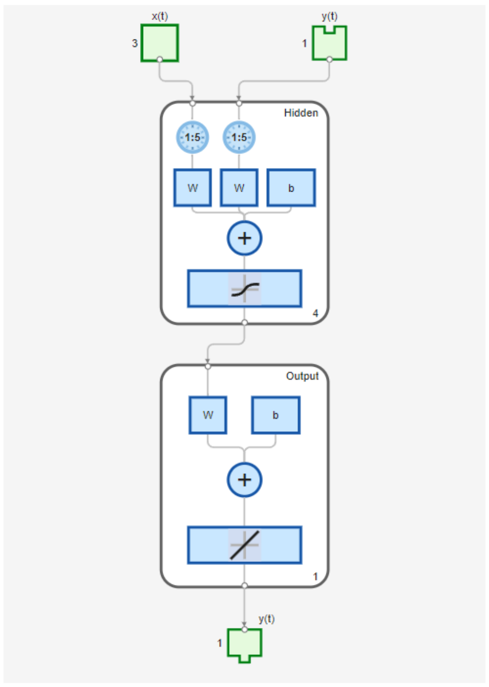

Fig. 1. Example Neural Network structure

The image Fig.1 shows a diagram of a neural network consisting of hidden and output layers, where the input is the data vector $x(t)$, and the output is the value $y(t)$. Below is a detailed description of each element visible in the diagram:

- **Input** $x(t)$: In the upper left corner, there is a block labeled x(t), representing an input vector of size 3. This indicates that the network is fed three different variables as input (e.g., disk speed, control, angular position of the pendulum).
- **Hidden Layer**
  - Layer size: number of neurons mapping the input.
  - Inputs to the layer: Each neuron in the hidden layer processes data from three inputs x(t).
  - Elements in the hidden layer:
    * Weights (W): Each neuron in the hidden layer is assigned weights, which are multiplied by the input data. These weights determine the influence of each input on the outputs of the neurons.
    * Bias (b): A bias value is added to the weights to help model more complex relationships.
  - Activation: The result of the summation ($weight \times input + bias$) is passed through an activation function. Here we see a graphical activation function, which is likely nonlinear (e.g sigmoid). This function introduces nonlinearity into the model, which allows for better representation of complex relationships in the data.
- **Output Layer**
  - Layer size: number of neurons mapping the output
  - Inputs to the layer: The outputs of the input hidden layer neurons are processed by a single output neuron.
  - Elements in the output layer:

    * Weights (W): Defines the influence of each of the input hidden neurons on the output value.
    * Bias (b): A bias value is added to the calculation.
  - Activation: The activation function in the output layer, which is likely linear or nonlinear depending on the characteristics of the problem (e.g. linear for regression).

- **Output** $y(t)$) At the bottom right is the output labeled y(t), representing the estimated value (for example, the angular velocity of a pendulum). This is the result of the entire network.

Additional information:

- Connections between layers: Arrows between blocks show the flow of data from $x(t)$ through the hidden layer and to the output layer.
- Colors and symbols: Each block (e.g. W, b, activation function) is represented graphically to separate different processing steps.

2 Experiments

In this section there is a validation for 6 independently created Neural Networks. Validation were performed on dataset described below:

- data_single_30deg.mat - single pendulum swing of angular position equal to 30°
- data_double_-25deg_25deg.mat - double pendulum swing of angular position equal to $-25°$ and 25° diffed by 20 [s]
- data_double_15deg_20deg.mat - double pendulum swing of angular position equal to 15° and 25° diffed by 20 [s]
- data_triple_-5deg_15deg_-25deg.mat - triple pendulum swing of angular position equal to $-5°$, 15° and $-25°$ diffed by 20 [s]

Training dataset's describtion can be found below:

- data_positive_max_swing.mat - single pendulum swing of angular position equal to 30°
- data_2_max_swings.mat - double pendulum swing of angular position equal to $-30°$ and 30°
- data_complex.mat - multiple pendulum swing of angular position equal from $-30°$ to 30° omitting 0°

2.1 NARX NN Training

In table 1 there is a describtion of configuration parameters of each created NARX NN.

Table 1. Configuration parameters

Version of NN	No of Neurons	No of epochs	Training data	Elapsed time of training
net1	4	1000	positive_max_swing	$3[s]$
net2	4	1000	2_max_swing	$5[s]$
net3	4	1000	complex	$1:42[min]$
net4	4	5000	complex	$8:11[min]$
net5	8	5000	complex	$14:46[min]$
net6	16	5000	complex	$33:08[min]$

2.2 Validation

In this section there are figures presenting validation performance and table with summary of errors.

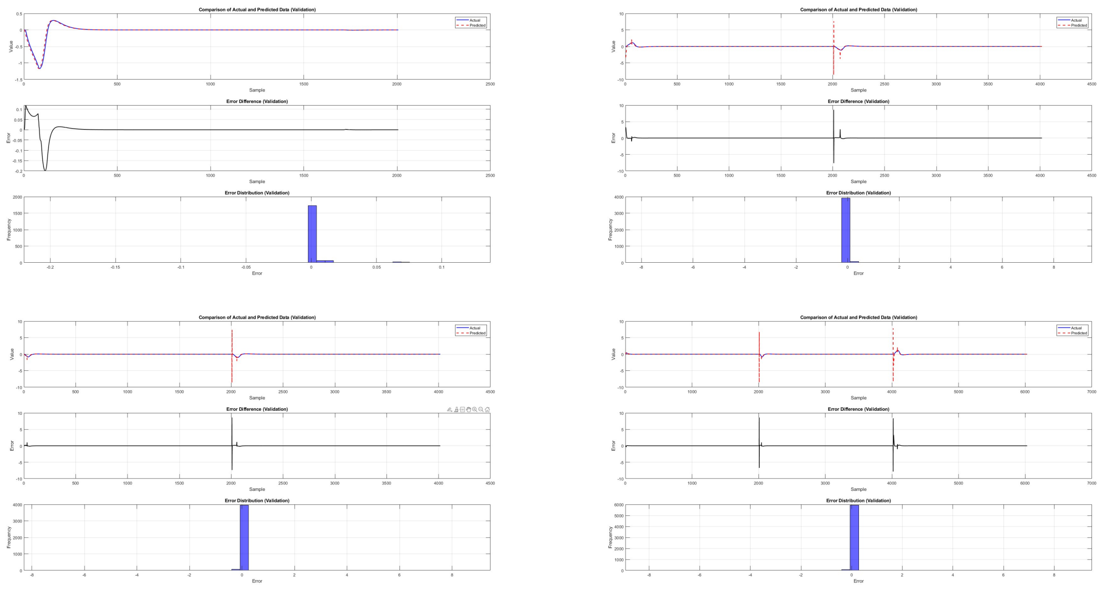

Fig. 2. Validation for net1

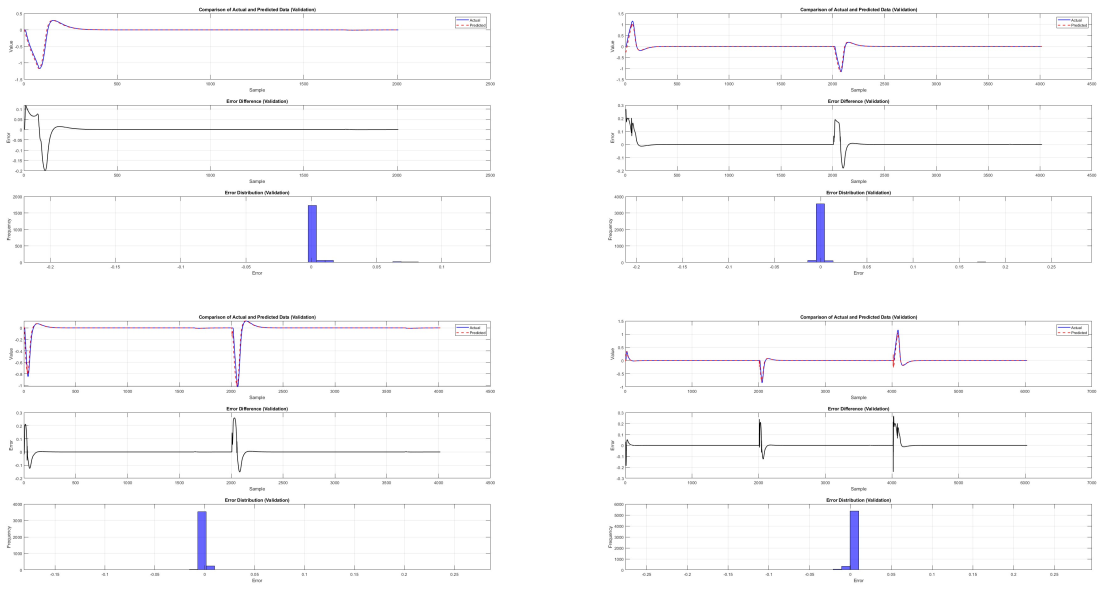

Fig. 3. Validation for net2

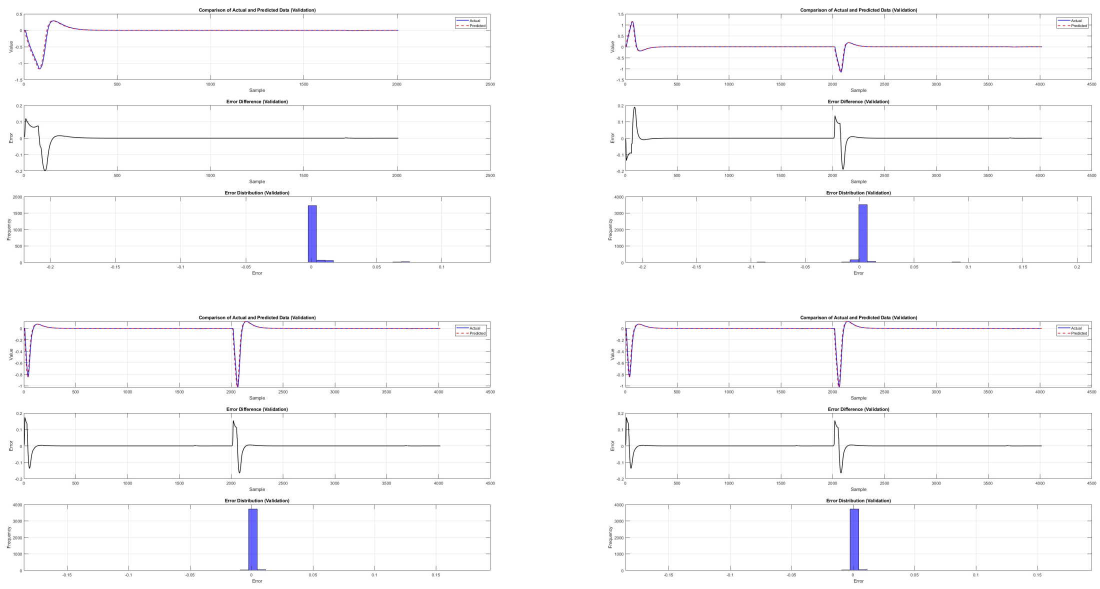

Fig. 4. Validation for net3

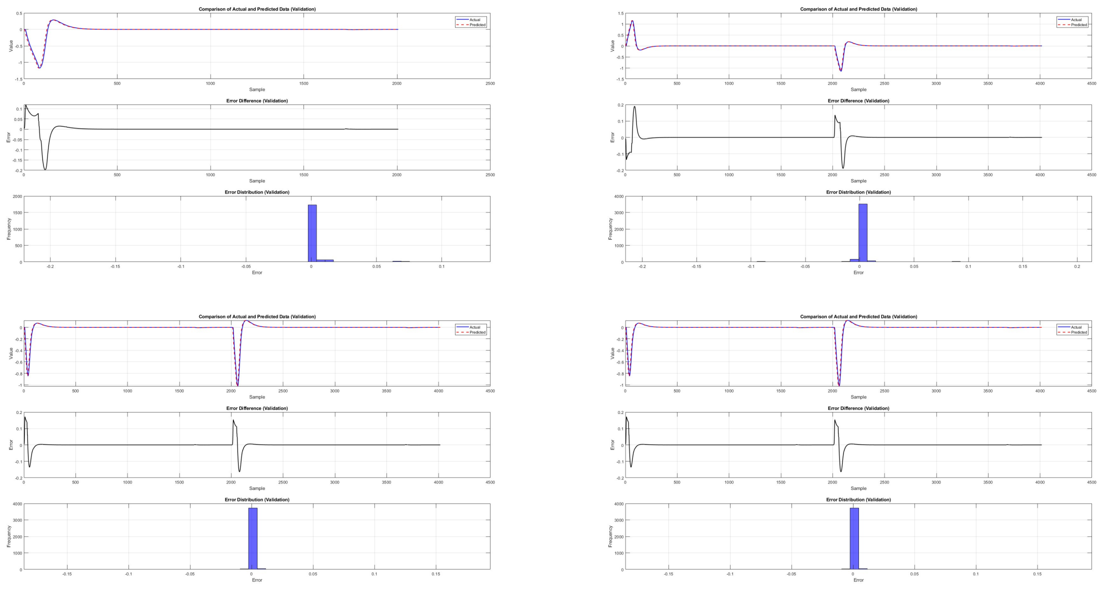

Fig. 5. Validation for net4

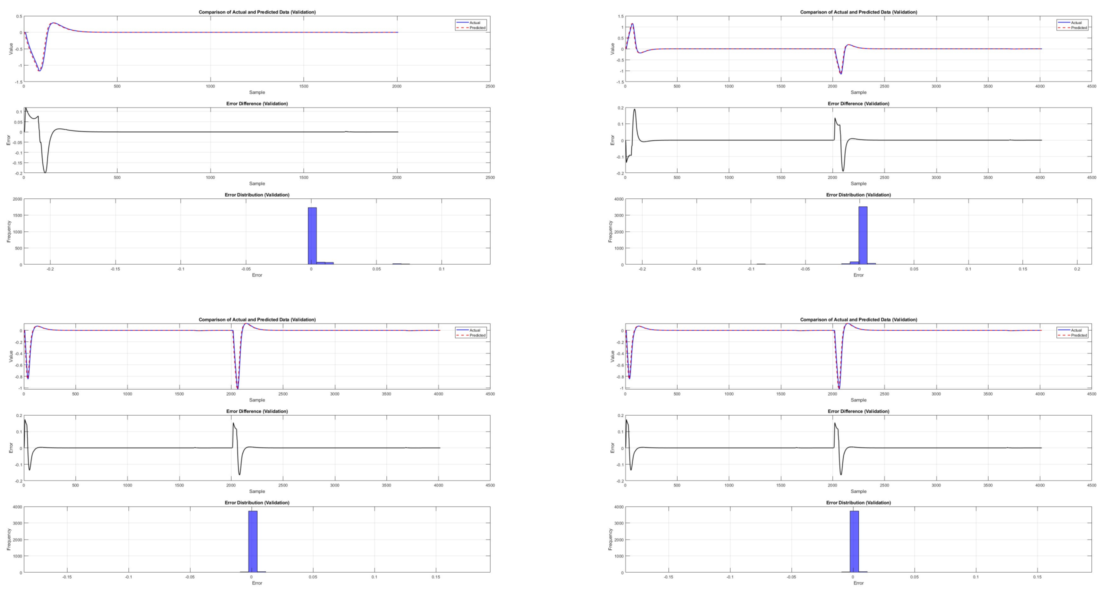

Fig. 6. Validation for net5

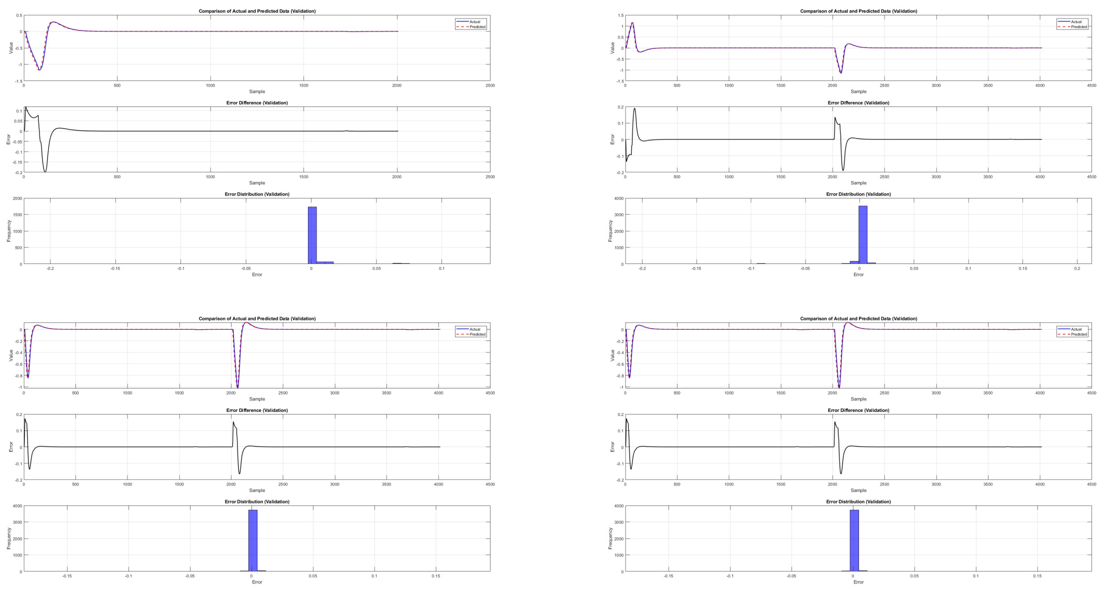

Fig. 7. Validation for net6

Table 2. Quality indicators

Version of NN	No of validation	Mean Squared Error (MSE)	Mean Absolute Error (MAE)
net1	1	$7.76 \cdot 10^{-4}$	$7.31 \cdot 10^{-3}$
	2	$5.72 \cdot 10^{-2}$	$2.05 \cdot 10^{-2}$
	3	$3.49 \cdot 10^{-2}$	$1.06 \cdot 10^{-2}$
	4	$5.62 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$
net2	1	$7.76 \cdot 10^{-4}$	$7.31 \cdot 10^{-3}$
	2	$1.29 \cdot 10^{-3}$	$8.72 \cdot 10^{-3}$
	3	$9.56 \cdot 10^{-4}$	$6.36 \cdot 10^{-3}$
	4	$7.29 \cdot 10^{-4}$	$5.28 \cdot 10^{-3}$
net3	1	$7.84 \cdot 10^{-4}$	$7.40 \cdot 10^{-3}$
	2	$7.69 \cdot 10^{-4}$	$6.72 \cdot 10^{-3}$
	3	$5.74 \cdot 10^{-4}$	$5.18 \cdot 10^{-3}$
	4	$4.78 \cdot 10^{-4}$	$4.38 \cdot 10^{-3}$
net4	1	$7.77 \cdot 10^{-4}$	$7.35 \cdot 10^{-3}$
	2	$7.67 \cdot 10^{-4}$	$6.72 \cdot 10^{-3}$
	3	$5.77 \cdot 10^{-4}$	$5.17 \cdot 10^{-3}$
	4	$4.79 \cdot 10^{-4}$	$4.39 \cdot 10^{-3}$
net5	1	$7.76 \cdot 10^{-4}$	$7.32 \cdot 10^{-3}$
	2	$7.69 \cdot 10^{-4}$	$6.72 \cdot 10^{-3}$
	3	$5.78 \cdot 10^{-4}$	$5.16 \cdot 10^{-3}$
	4	$4.78 \cdot 10^{-4}$	$4.38 \cdot 10^{-3}$
net6	1	$7.76 \cdot 10^{-4}$	$7.31 \cdot 10^{-3}$
	2	$7.68 \cdot 10^{-4}$	$6.70 \cdot 10^{-3}$
	3	$5.78 \cdot 10^{-4}$	$5.16 \cdot 10^{-3}$
	4	$4.78 \cdot 10^{-4}$	$4.37 \cdot 10^{-3}$

Table 3. Average Quality indicator

Version of NN	Average MSE	Average MAE
net1	$3.73 \cdot 10^{-2}$	$1.34 \cdot 10^{-2}$
net2	$9.38 \cdot 10^{-4}$	$6.92 \cdot 10^{-3}$
net3	$6.54 \cdot 10^{-4}$	$5.92 \cdot 10^{-3}$
net4	$6.50 \cdot 10^{-4}$	$5.91 \cdot 10^{-3}$
net5	$6.50 \cdot 10^{-4}$	$5.90 \cdot 10^{-3}$
net6	$6.50 \cdot 10^{-4}$	$5.89 \cdot 10^{-3}$

2.3 Conclusions

Key Findings

1. **NARX Neural Networks for Velocity Estimation:**
   - NARX (Nonlinear Autoregressive with External Input) networks were effectively used to estimate the velocity of a reaction pendulum.
   - The network architecture demonstrated its capability to predict velocity based on inputs such as control signals, angular position, and disk velocity without requiring a direct velocity sensor.
2. **Performance of NARX Networks:**
   - Six versions of NARX networks were trained and validated using various datasets (e.g., simple single pendulum swings and complex multiple swings).
   - The complexity of the network (e.g., increasing the number of neurons and epochs) improved performance in terms of both Mean Squared Error (MSE) and Mean Absolute Error (MAE).
   - The best-performing networks (net4, net5, net6) achieved consistently low error rates, with average MSEs and MAEs around 6.50×10^{-4} and 5.90×10^{-3}, respectively.

3. **Training and Validation Results:**
    - Networks trained on more complex datasets (e.g., *complex*) required significantly longer training times but exhibited better performance during validation.
    - Simpler networks (e.g., net1, net2) showed higher error rates, particularly when dealing with complex validation scenarios.
4. **Advantages of the Approach:**
    - The NARX network-based observer eliminates the need for expensive velocity sensors, making the solution cost-effective.
    - Its ability to handle nonlinear relationships and temporal delays makes it robust to noise and disturbances, ideal for dynamic systems.

Practical Implications

- **Scalability:** The results indicate the potential scalability of NARX networks for other dynamic systems requiring state observation without direct measurement.
- **Optimization:** While more complex networks improve accuracy, they also demand greater computational resources and time. An optimal trade-off should be considered based on the application requirements.

Summary of Limitations

- **Complexity vs. Performance:** As network complexity increases (e.g., number of neurons and epochs), the training time grows significantly, raising questions about real-time applicability in systems with constrained computational resources.

Summary

The document demonstrates that NARX neural networks are highly effective for estimating the velocity of a reaction pendulum, offering a cost-efficient and accurate alternative to traditional sensors. Further exploration and optimization could expand the applications of this approach to other fields where dynamic state observation is critical.