www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 16(3), May-Aug 2025, 11-24. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i3.1121

Implementation of an artificial neural network for the position control of a seesaw driven with a thrust propeller in open loop

*Mario Oscar Ordaz Oliver [1], María Angélica Espejel Rivera [1], Javier Hernández Pérez *[2], Evelin Gutiérrez Moreno [2], Jesús Patricio Ordaz Oliver[3], Amadeo Manuel Hernández Hernández [1]*

[1] Departamento de Ingeniería Eléctrica y Electrónica, Tecnológico Nacional de México, Campus Pachuca.
[2] Departamento de Ingeniería Mecatrónica, Universidad Politécnica de Pachuca.
[3] Centro de Investigación en Tecnologías de la Información y Sistemas, Universidad Autónoma del Estado de Hidalgo.
mario.oo@pachuca.tecnm.mx, maria.er@pachuca.tecnm.mx, jahdez@upp.edu.mx, evgutierrez@upp.edu.mx, jesus_ordaz@uaeh.edu.mx, amadeo.hh@pachuca.tecnm.mx

Abstract. This project details the implementation of an artificial neural network (ANN) as the principal element of a system engineered to identify and predict future control states of a thrust-propelled seesaw in an open-loop configuration. The primary objective was to maintain the seesaw in a balanced 90° position. The system's dynamic behaviour was analysed under minimal external disturbances, facilitating development and evaluation in a controlled environment. Experimental data were captured via a programmable Arduino UNO board transmitting over the serial port and recorded in an Excel file for subsequent processing. A Kalman filter was applied to refine the data, from which a random subset was selected to train the neural network. A comprehensive analysis of the results is presented herein, demonstrating the ANN's satisfactory performance in the control task.
Keywords: Kalman filter; open loop, Arduino UNO; stabilization.

Article Info
Received April 17, 2025
Accepted Jul 1, 2025

1 Introduction

Control systems have extensive and relevant applications across various research and industry areas, as their ability to confer stability to systems significantly enhances their performance. Control engineering has evolved into a deeply multidisciplinary field, both in its theoretical foundations and practical applications. A neural network represents a computational paradigm based on the structure and operation of the biological nervous system. Its conception is based on performing tasks like information processing and analysis, primarily in configuration identification, knowledge acquisition, and decision-making. Computational algorithms, including neural networks, have experienced a constant increase in their implementation in the field of automatic and intelligent control. These algorithms represent a form of nonlinear and adaptive control. They are mostly used to manage systems that exhibit great complexity, proving particularly effective in handling highly complex dynamic systems. This choice is supported by the intrinsic difficulty of modeling the relationships between the inputs and outputs of such systems through conventional approaches (Prieto et al., 2016; Levine & Aparicio, M, 2013). This article addresses their specific application in the context of a seesaw driven with a thrust propeller, a system of great importance in engineering due to its dynamic behavior highly sensitive to various variables (Morales-Narciso et al., 2023). The introduction of a neural network as a control strategy aims to leverage the inherent ability of this technique to adapt and learn from data; to manage the complex interactions and dependencies present in the seesaw system. In its basic configuration, a neural network is structured from interconnected nodes called artificial neurons. Each neuron receives multiple input signals, which are weighted and processed through an activation function, resulting in a corresponding output. These intraneuronal connections, along with their respective weighted values known as synaptic weights, are adapted during the training process, allowing the network to acquire the ability to perform specific tasks. Armenta (2022) and his thesis on the control of robotic manipulators using artificial neural networks introduce a PID (Proportional, Integral, Derivative) control scheme with adaptive compensation through neural networks for trajectory tracking. The neural network presented has been created using MATLAB software, primarily to make anticipatory projections regarding the positioning of the mentioned seesaw system. It is important to mention that this system has been conceived with a cost-effective approach, using readily available and low-cost materials. Another important feature is that the assembly was carried out with conventional tools. The electronic part of the system was designed simply due to the use of easily accessible

market components. For example, an Arduino UNO programmable board was used as a central component, along with a recycled power source from a computer. This choice of components and materials reflects the intention to develop an efficient and affordable system without compromising the quality and functionality needed for the control and positioning of the seesaw in question. In their article titled "Módulo didáctico de control de posición de un balancín eólico" engineers Estivet Alejandro Giraldo and Quintero Sebastián Herrera Gil describe in detail the approach they adopted for controlling the position of a seesaw driven with a thrust propeller. They highlight the use of PID control as an essential part of their methodology. In addition to detailing the proposed mechanical architecture, the engineers also analyze its effectiveness. Through their work, they explore how the designed system and PID approach have been combined to successfully achieve the position control of the seesaw driven with a thrust propeller. The publication provides a comprehensive view of their research and development, offering a deep understanding of the implementation and results obtained (Giraldo & Herrera, 2018).

To provide a comprehensive understanding of the proposed approach, the structure of this paper is organized to guide the reader through the different stages of the study. Following this introduction, **Section 2** presents the methods, techniques, and instruments used, beginning with a description of the physical mechanism and the architectural methodology behind the system. It continues with the explanation of the implemented algorithm, the structure and functioning of the artificial neural network, the process of experimental data acquisition for training, and the details of the control logic implemented in MATLAB. **Section 3** is dedicated to the presentation and analysis of the results, including graphical representations obtained prior to training and the experimental validation of the model. Finally, **Section 4** discusses the main conclusions drawn from the study, emphasizing the effectiveness and limitations of the neural network-based open-loop control strategy.

2 Methods, techniques, and instruments

This section outlines the methodological framework, techniques, and tools employed in the development and implementation of the control system for the thrust-propeller-driven seesaw. Emphasis is placed on the integration of cost-effective components and widely accessible technologies to construct a functional prototype capable of demonstrating intelligent control strategies. The design and experimentation processes combined hardware and software elements, including the use of an Arduino UNO board for data acquisition, MATLAB for the development of the neural network, and standard tools for physical assembly. The procedures adopted aimed to balance simplicity, reproducibility, and effectiveness, ensuring that the system remains accessible for educational and research applications while maintaining the technical rigor necessary for meaningful experimentation and analysis.

2.1 Mechanism and architectural methodology

The design and construction of the seesaw system followed a methodological approach focused on simplicity, low cost, and functional efficiency. Emphasis was placed on minimizing mechanical friction to enhance the dynamic response of the mechanism. However, due to the inherent instability of the system, an open-loop control configuration was required to maintain the balance position. To address this, a neural network was trained to anticipate the positioning of the seesaw at 90 degrees. The output of the network is transmitted to the motor via an Arduino UNO board, enabling effective actuation of the system. The mechanical structure was built using readily available market materials and assembled with conventional tools, striking a balance between affordability and performance. The final design, as illustrated in Figure (1), provides a robust and replicable platform for the implementation of intelligent control strategies.

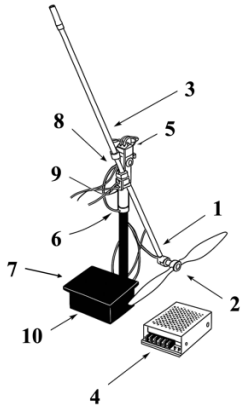

Fig. 1. Architecture of the seesaw driven with a thrust.

The construction of the thrust-propeller-driven seesaw system required a careful selection of components to ensure functionality, efficiency, and accessibility. Each element was chosen based on its performance characteristics, compatibility with the system's architecture, and availability in the market. The integration of these components supports the mechanical stability, electronic control, and overall responsiveness of the system. Furthermore, an emphasis was placed on cost-effectiveness and sustainability, incorporating recycled elements where feasible without compromising operational quality. The following is a detailed description of the primary components used in the design and implementation of the system:

1. Motor: The Brushless A2212 KV2200 motor is used to drive the system. In this context, the Brushless A2212 KV2200 was chosen, a component widely used in aeromodelling applications. Its notable features include detachable mounts, an associated controller, and a mounting platform. The choice of this motor is based on its ability to generate an airflow against gravitational force, focusing on one end of the mechanism.
2. 10×45 clockwise propeller: The propellers are indispensable parts of the aircraft that develop the necessary force to lift and manipulate the aircraft during flight time. For the present work, the selected 1045 propellers are made of polymer, with the characteristic of being self-locking, that is, they do not require adjustment with a tool when mounted on the engine shaft.
3. Aluminum tube for the structure of 0.5 m length: This element is used as a balance beam and has a symmetrical geometry with respect to the pivot point.
4. Power Supply: In summary, the decision to use a power supply from an obsolete computer is a conscious and eco-friendly response to the system's energy needs. At the same time, this choice aligns the project with recycling and sustainability considerations.
5. Sensor (potentiometer): The integration of a sensor serves a dual purpose: data collection and acting as a support point. In the system configuration, a potentiometer was incorporated due to its suitability for this application. The integration of this potentiometer sensor proves to be a strategic component in the design, fulfilling both an essential role in data acquisition and an important physical function as a support point.
6. Copper tube for the structure 0.5 m long: This component is used to support the weight of the seesaw, the sensor, the brushless motor with its propeller and the electronic speed controller. It is oriented vertically to distribute the different connection cables easily.
7. Base: In the system, the base plays a crucial role in promoting stability and minimizing disturbances. Its main function is to provide a solid and secure anchoring point for the entire system.
8. 30A Electronic Speed Controller (ESC): Electronic Speed Controllers regulate the power sent to the brushless motor. This control is achieved by modulating the electrical signal that activates the motor's coils, allowing the speed and direction of rotation to be adjusted. By varying the frequency and voltage, the ESC can increase or decrease the power received by the motor, which influences its performance.
9. Bullet connectors of 0.002m: Bullet connectors are used to establish a secure and efficient connection between the ESC and the brushless motor. Their main function is to allow the transmission of electrical current between the controller and the motor, ensuring that the power is delivered optimally. These connectors are easy to assemble and disassemble, making maintenance and repairs easier. Additionally, their design minimizes electrical resistance and the risk of loose connections, contributing to more reliable system performance. They are also capable of handling high currents, making them ideal for high-power applications such as electric vehicles and drones.
10. Wooden base of 0.06×0.12×0.15m: The base system plays a crucial role in promoting stability and minimizing disturbances. Its main function is to provide a solid and secure anchoring point for the entire system.

The mechanical and architectural design of the seesaw system was guided by principles of simplicity, affordability, and functional efficiency. The integration of readily available materials and standard fabrication techniques enabled the construction of a robust structure capable of supporting the control objectives of the project. The selected configuration not only facilitated ease of assembly and maintenance but also provided a stable platform for evaluating the performance of the neural network-based control strategy. This foundational architecture serves as a reliable framework for subsequent experimental procedures and control system testing, validating its suitability for research and educational purposes alike.

2.2 Description of the Implemented Algorithm

The Kalman filter is an estimation algorithm widely used in signal processing and control theory (Grewal & Andrews, 2014). It is particularly useful for state estimation in systems where measurements are affected by noise or uncertainty and when there is prior knowledge about the system's temporal evolution. The filter relies on a mathematical model that describes how the state of a system evolves over time. This model can be expressed through differential equations, discrete state equations, or other representations that capture the system's dynamics (Katlin, 2012).

The state of the system comprises a set of variables representing the properties to be estimated. These variables can be physical, such as the position and velocity of an object, or conceptual, such as the internal parameters of a model. Typically, the state is represented as a vector containing these variables.

Using the dynamic model of the system, the Kalman filter predicts the future state by considering the previous state and any available system inputs. This prediction provides an estimate of the next state, accounting for both the expected value and the inherent uncertainty. Additionally, the filter forecasts how the error in state estimation will propagate over time by calculating the error covariance matrix, which reflects how uncertainties in the state variables will evolve.

The Kalman filter operates iteratively as time progresses, continuously refining the state estimate with each new measurement. This approach makes the algorithm highly effective and efficient at reducing the impact of noise and uncertainty in state estimation (Kumari et al., 2021).

The Kalman filter operates in two interrelated phases: prediction and update. In the prediction stage, the evolution of the system state is modeled by discrete state equations of the form:

$$X_{\tilde{K}} = AX_{K-1} + W_K \tag{1}$$

Where: **X** represents the state vector, **A** denotes the state transition matrix and **W** reflects the noise of the process. Simultaneously, the prediction covariance matrix $\mathbf{P_K}$ is calculated, which quantifies the uncertainty related to the state estimation and is defined as:

$$P_{\tilde{K}} = AP_{K-1}A^T + Q_K \tag{2}$$

Where $\mathbf{Q_K}$ is the covariance matrix of the process noise at iteration **K**. In the update phase, the $\mathbf{Z_K}$ measurements acquired from the real system are incorporated, which are expressed as follows:

$$Z_K = Hx_{\tilde{k}} + v_k \tag{3}$$

Here, **H** is defined as the observation matrix linking the state to the measurements, while $\boldsymbol{\nu_K}$ is defined as the residual vector at iteration **K**. From the discrepancy between measurements and predictions, the residual $\mathbf{Y_K}$ is computed. Then, the Kalman gain $\mathbf{K_K}$ is evaluated, which weights the contribution of measurements in updating the estimated state.

$$K_K = P_{\tilde{K}}H^T(HP_{\tilde{K}}H^T + R_K)^{-1} \tag{4}$$

Where: $\mathbf{R_K}$ refers to the covariance matrix of the measurement noise at iteration **K**. This matrix is critical as it captures the uncertainty and variability associated with the system measurements at each time step or iteration. The $\mathbf{R_K}$ matrix quantifies how reliable or accurate the measurements are at time **K** and is a crucial component in the Kalman filtering process for improving the estimation of the system state in the presence of noise and measurement errors.

The updated estimate of the state $\mathbf{X_K}$ and the covariance matrix $\mathbf{P_K}$ are obtained using the following equations:

$$X_K = X_{\tilde{K}} + K_K y_K \tag{5}$$

With:

$$y_K = Z_K - H_{X_{\tilde{k}}} \tag{6}$$

And:

$$P_K = (I - K_K H)P_{\tilde{K}} \tag{7}$$

The Kalman filter stands out for its ability to generate accurate and adaptive estimates, as well as its ability to consider the uncertainty present in the measurements and the process itself. Its applicability extends to a wide variety of fields, from navigation to sensor fusion and control of complex systems.

2.3 Structure of a Neural Network

An Artificial Neural Network is a mathematical model inspired by the biological functioning of neurons and the structure of the brain (Zakaria et al., 2014). It can also be viewed as an intelligent system that approaches tasks differently from contemporary computers. Despite the rapid information processing capabilities of modern computers, certain highly complex tasks, such as pattern recognition and classification, require excessive time and effort, even with today's most powerful communication systems. Figure (2) shows the structure of an artificial neural network:

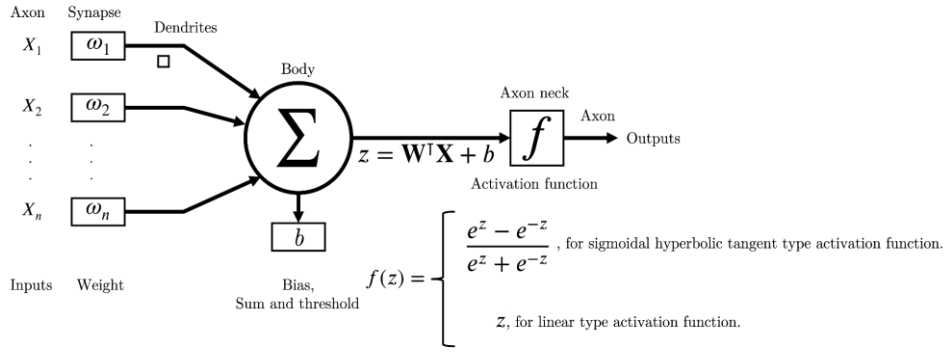

Fig. 2. Structure of an Artificial Neural Network.

where:

1. Dendrites: These are the branches of the neuron that receive signals or stimuli from other neurons. These signals are transmitted to the cell body.
2. Body (Soma): This is the main part of the neuron that contains the nucleus and other essential structures. It is here that the signals received from the dendrites are integrated.
3. Axon: This is a long extension of the neuron that transmits electrical signals (action potentials) from the cell body to other neurons or cells. The axon may be covered by a myelin sheath, which increases the speed of signal transmission.
4. Synapse: It is the functional connection between two neurons. The synapse allows for the transmission of signals from one neuron to another through chemical substances called neurotransmitters.
5. Axon neck: This part of the axon connects the cell body to the main axon and can be crucial for the proper propagation of nerve impulses.
6. Outputs Axon: Refers to the neuron's main axon that carries the nerve impulse to other neurons or target cells.
7. Inputs: These are the signals or stimuli that arrive at the dendrites and are processed within the neuron.
8. Weight: In the context of artificial neural networks, weights are numerical values assigned to the connections between neurons. These weights determine the strength and influence of one neuron on another in the transmission of signals.
9. Activation Function: Also known as “Transfer Function”, it is a mathematical function applied to the weighted sum of a neuron's inputs. It determines whether the neuron activates (produces an output) or not by calculating against a threshold or within a specific range.
10. Outputs: These are the resulting signals that a neuron sends to other neurons or cells after processing the inputs and applying the activation function.

In an artificial neural network, such as those used in machine learning, these concepts are modeled similarly to how they function in biological neurons. Inputs are weighted by the weights, summed, and an activation function is applied to determine the output of each neuron in the network.

2.4 Collection of Experimental Data for Training the Neural Network

To carry out data acquisition, a setup was implemented in which an Arduino UNO board played a central role. This board served as the main controller, orchestrating, and executing the operations required to capture information related to the behavior of the pendulum.

The adopted methodology was based on generating a Pulse Width Modulated (PWM) signal, a technique that allows for the regulation of the energy supplied to a device. In this context, PWM was used to precisely control the amount of energy imparted to the pendulum at regular intervals. This strategy proves particularly effective for managing the amplitude of the pendulum's movement and for analyzing its behavior under controlled conditions. The pendulum was activated in a controlled environment,

meaning that uniform and predictable conditions were established for observing its movement. Once in motion, a sensor (in this case, a potentiometer) was used to collect the essential data.

The captured data consisted of numerical values representing the instantaneous position of the pendulum. These values were recorded on a scaled range from 1 to 1023. Through this scale, it was possible to infer the angle of the pendulum at each moment, which allowed for understanding the variations in its position. This data was recorded in an Excel file. The use of an Arduino UNO board, the generation of PWM pulses, and the application of a sensor for data collection facilitated the analysis of a pendulum's behavior in a controlled environment. The collected values provided information regarding the pendulum's angular position at various moments, which is crucial for analyzing and understanding its movement patterns and behaviors.

2.5 Description of MATLAB Code

The MATLAB code developed for this project constitutes a fundamental component in the design and implementation of the intelligent control strategy. It encompasses a structured sequence of operations dedicated to the preprocessing, training, and validation of the artificial neural network responsible for regulating the position of the thrust-propeller-driven seesaw. The code integrates advanced filtering techniques—such as Kalman filtering—to reduce noise and improve the quality of the input data, as well as predictive algorithms that enhance the network's capacity for accurate estimation. Through these procedures, the system can anticipate and adjust the seesaw's position with high precision, even in the presence of variable dynamics. This computational architecture not only supports the neural network's learning process but also plays a critical role in ensuring robust and adaptive performance during system operation.

The following MATLAB script was developed as part of the data preprocessing stage, aimed at enhancing the quality of the experimental measurements used for training the neural network. This preprocessing step is essential for reducing the influence of noise and improving the reliability of the input data. The algorithm implements a simplified predictive filtering approach inspired by Kalman filtering techniques, applied individually to each signal recorded during the experimental sessions. The output of this process is a set of smoothed signals that more accurately represent the system's behavior, providing a robust foundation for the subsequent stages of training and validation. The implementation is shown below.

```
clearvars
close all
clc
format long
filename = 'Reading.xlsx';
data = importdata(filename);
[numRows, numCols] = size(data);
totalSteps = length(data);
A = 1;
h = 0.1;
t = 0:h:(totalSteps*h)-h;
for j = 1:numCols
    ZK = data(:,j)';
    RK = var(ZK);
    QK = 1023 / 8;
    XK = 0;
    PK = zeros(1, totalSteps);
    XK_estimated = zeros(1, totalSteps);
    Pk = 0;
    for k = 1:totalSteps
        XK = A * XK;
        Pk = (A^2) * (Pk + QK);
        YK = ZK(1,k) - XK;
        KK = Pk / (RK + Pk);
        XK = XK + KK * YK;
        Pk = (1 - KK) * Pk;
        XK_estimated(1,k) = XK;
```

```
        PK(1,k) = Pk;
    end
    X_estimated(j,:) = XK_estimated;
end
```

The purpose of this MATLAB code is to apply a simplified first-order predictive filter, based on Kalman filtering principles, to a set of time-series signals obtained from an external file named Reading.xlsx. This procedure is designed to reduce measurement noise and produce smoothed signal estimations suitable for subsequent processing stages, such as the training of an artificial neural network in the context of a control system.

The code initiates by clearing the workspace, closing all open figures, and resetting the command window to ensure a clean environment for data processing. It then loads the input dataset, which is assumed to contain multiple signal recordings arranged column-wise. The script computes the total number of data points and prepares a corresponding time vector based on a predefined sampling interval. Each column of the dataset is interpreted as a distinct signal to be individually filtered.

For each signal, the algorithm estimates its variance and applies a recursive filtering process. This process involves iteratively predicting the next signal value using a model of exponential decay governed by a constant coefficient. The prediction is updated at each step by computing the estimation error and adjusting it with a dynamic gain factor derived from the variance of the signal and an assumed measurement noise level. The algorithm progressively refines its internal model to minimize prediction error, resulting in a smoothed output signal that is less affected by transient fluctuations or sensor noise.

The filtered signals are stored in a separate matrix for potential use in downstream tasks, such as system identification or control strategy implementation. Overall, the code demonstrates a practical application of predictive filtering techniques to enhance data reliability in noisy dynamic environments, particularly within experimental systems where real-time control and accurate state estimation are required.

To support the training and evaluation process of the neural network, it was necessary to define a temporal reference for the dataset and implement a method for randomized data selection. The following code fragment illustrates the generation of a time vector consistent with the sampling interval used during data acquisition, as well as the creation of a randomized set of indices. This procedure is intended to introduce variability into the training dataset, thereby enhancing the generalization capabilities of the neural network and reducing the likelihood of overfitting.

```
time = 0:h:(fin*h)-h;
dim = length(time);
pos = randperm(45, 41);
```

This MATLAB code fragment serves to define the temporal reference of the dataset and to randomly select a subset of data indices for subsequent computational tasks, such as neural network training or validation. The `time` vector is constructed to span the full duration of the experimental data, with a uniform sampling interval defined by the variable `h`, ensuring temporal alignment with the filtered measurements. The total number of discrete time steps is stored in the variable `dim`, representing the resolution of the temporal domain. Additionally, the use of the `randperm` function enables the randomized selection of 41 unique indices from a range of 1 to 45, which introduces stochastic variability into the data handling process. This randomization is essential for promoting generalization during training and minimizing bias in the selection of training or testing samples.

To proceed with the integration of the neural network into the system, the input and output data must first be prepared and normalized. The following MATLAB code segment handles this preprocessing step, ensuring that the data is structured and scaled appropriately for the neural network model. It creates input and output matrices, normalizes them, and then defines the neural network architecture for subsequent training. The code is designed to facilitate the prediction of the system's behavior based on experimental data, which has been filtered and organized in a manner conducive to efficient learning. Below is the MATLAB code responsible for this data preparation and network initialization.

```
IN=[time time time time time time time time time time time time time time time
time time time time time time time time time time time time time time time
time  time  time  time  time  time  time  time  time;  A(2,pos(1,1))*ones(1,612)
```

```
        A(2,pos(1,2))*ones(1,612)    A(2,pos(1,3))*ones(1,612)    A(2,pos(1,4))*ones(1,612)
        A(2,pos(1,5))*ones(1,612)    A(2,pos(1,6))*ones(1,612)    A(2,pos(1,7))*ones(1,612)
        A(2,pos(1,8))*ones(1,612)   A(2,pos(1,9))*ones(1,612)   A(2,pos(1,10))*ones(1,612)
        A(2,pos(1,11))*ones(1,612)  A(2,pos(1,12))*ones(1,612)  A(2,pos(1,13))*ones(1,612)
        A(2,pos(1,14))*ones(1,612)  A(2,pos(1,15))*ones(1,612)  A(2,pos(1,16))*ones(1,612)
        A(2,pos(1,17))*ones(1,612)  A(2,pos(1,18))*ones(1,612)  A(2,pos(1,19))*ones(1,612)
        A(2,pos(1,20))*ones(1,612)  A(2,pos(1,21))*ones(1,612)  A(2,pos(1,22))*ones(1,612)
        A(2,pos(1,23))*ones(1,612)  A(2,pos(1,24))*ones(1,612)  A(2,pos(1,25))*ones(1,612)
        A(2,pos(1,26))*ones(1,612)  A(2,pos(1,27))*ones(1,612)  A(2,pos(1,28))*ones(1,612)
        A(2,pos(1,29))*ones(1,612)  A(2,pos(1,30))*ones(1,612)  A(2,pos(1,31))*ones(1,612)
        A(2,pos(1,32))*ones(1,612)  A(2,pos(1,33))*ones(1,612)  A(2,pos(1,34))*ones(1,612)
        A(2,pos(1,35))*ones(1,612)  A(2,pos(1,36))*ones(1,612)  A(2,pos(1,37))*ones(1,612)
        A(2,pos(1,38))*ones(1,612)                              A(2,pos(1,39))*ones(1,612)
        A(2,pos(1,40))*ones(1,612)];
        OUT=[B(pos(1,1),:)   B(pos(1,2),:)   B(pos(1,3),:)   B(pos(1,4),:)   B(pos(1,5),:)
        B(pos(1,6),:)    B(pos(1,7),:)    B(pos(1,8),:)    B(pos(1,9),:)    B(pos(1,10),:)
        B(pos(1,11),:)   B(pos(1,12),:)   B(pos(1,13),:)   B(pos(1,14),:)   B(pos(1,15),:)
        B(pos(1,16),:)   B(pos(1,17),:)   B(pos(1,18),:)   B(pos(1,19),:)   B(pos(1,20),:)
        B(pos(1,21),:)   B(pos(1,22),:)   B(pos(1,23),:)   B(pos(1,24),:)   B(pos(1,25),:)
        B(pos(1,26),:)   B(pos(1,27),:)   B(pos(1,28),:)   B(pos(1,29),:)   B(pos(1,30),:)
        B(pos(1,31),:)   B(pos(1,32),:)   B(pos(1,33),:)   B(pos(1,34),:)   B(pos(1,35),:)
        B(pos(1,36),:) B(pos(1,37),:) B(pos(1,38),:) B(pos(1,39),:) B(pos(1,40),:)];
        [INN minIN maxIN OUTN minOUT maxOUT]=premnmx(IN, OUT);
 net=newff(minmax(IN),[2               22               17               12               1],
{'purelin','tansig','tansig','tansig','purelin'},'trainscg');
```

This MATLAB code segment prepares the input and output data for training a neural network, specifically designed to predict the behavior of a dynamic system based on experimental data. The input matrix, denoted as `IN`, consists of concatenated time data and signal values, which are replicated 40 times for different signal sequences from the dataset stored in the variable `A`. The output matrix, referred to as `OUT`, contains the filtered predictions of the system, which were previously calculated and stored in the matrix `B` after applying the Kalman filter.

The purpose of organizing these matrices is to structure the data in a format suitable for training a feedforward neural network. Subsequently, the code normalizes both the input and output data using the `premnmx` function, which scales the data to a range of [-1, 1] for improved network performance. This normalization process is applied to both the inputs (`IN`) and outputs (`OUT`), with the minimum and maximum values being calculated for the scaling process.

After preprocessing the data, the code defines and initializes a neural network using the `newff` function. This function creates a feedforward network with an architecture consisting of three layers. The first hidden layer contains 22 neurons, the second hidden layer includes 17 neurons, and the third layer consists of 12 neurons, while the output layer contains a single neuron. The activation functions used for the network are linear (`'purelin'`) for both the input and output layers and the hyperbolic tangent sigmoid function (`'tansig'`) for the hidden layers.

The network is configured to use the scaled conjugate gradient backpropagation algorithm (`trainscg`) for training, which is an efficient optimization method for large neural networks. This setup ensures that the network is ready for training with the preprocessed and normalized data, aiming to model the relationship between the inputs and outputs of the system. The purpose of this code is to preprocess the time-series data and the filtered output, normalize it, and establish a neural network model that will be trained to predict the system's behavior. This trained neural network can then be employed for further analysis or control applications where precise predictions of the system's dynamics are necessary.

To illustrate the final stage of the modeling process, the following MATLAB code segment presents the initialization, configuration, training, and evaluation of the neural network using previously normalized input and output data. This portion of the code also includes the generation of predictions for an untrained experimental dataset and their subsequent comparison with filtered measurements through graphical representations.

```
net=init(net);
net.trainParam.goal= 1.88e-30;
net.trainParam.min_grad = 1e-40;
net.trainParam.epochs=1000000;
net.trainParam.max_fail=10;
[net,tr]=train(net, INN, OUTN);
%%
in1=A(2,pos(1,41))*ones(1,612);
in1=A(2,pos(1,41))*ones(1,612);
IN1 =[time; in1];
inN = tramnmx (in1, minIN, maxIN);
fitn = sim(net,inN);
fitN=postmnmx(fitn, minOUT, maxOUT);
figure(1)
 title('Experimental data vs ANN prediction')
 plot(time, B(pos(1,41),:))
 hold on
 plot(time, fitN)
 legend('Experimental Data, ANN Prediction')
 xlabel('time (s)')
 ylabel('Angular Position (degrees)')
 grid on
figure(2)
plotregression(B(pos(1,41),:),fitN)
grid on
```

The purpose of this MATLAB code segment is to initialize, configure, train, and evaluate an artificial neural network designed to model the angular position dynamics of a seesaw system. The network is first initialized to ensure consistent and reproducible training behavior. The training parameters are set with high precision, including a stringent performance goal, a very small minimum gradient threshold, an extended maximum number of training epochs, and a limit on the number of validation failures to avoid overfitting.

Once the training process is carried out using the normalized input and output datasets, a new input signal corresponding to an untrained data series is selected and normalized using the previously established parameters. This input is then passed through the trained neural network to obtain a prediction of the system's response, which is subsequently denormalized to return it to its original scale. The predicted output is compared against the experimentally filtered signal to assess the model's ability to generalize beyond the training data.

The evaluation is visually supported by two graphical outputs. The first figure presents a time-domain comparison between the predicted angular position and the corresponding experimental data, highlighting the accuracy of the network's prediction. The second figure displays a regression analysis plot, which provides a quantitative measure of the linear correlation between the predicted and actual responses. This final stage of the process is critical in validating the effectiveness of the neural network model in estimating system behavior under conditions not explicitly included during training.

3 Results

This section presents the main results and findings of this research work, which includes a series of graphs associated with the training of the artificial neural network (ANN), as well as the validation of the training by means of the prediction of the control signal (PWM) to bring the angular position of the seesaw to the desired value of 90º in open loop.

3.1 Graphs obtained prior to training

Upon the completion of the neural network's design and training process, the results obtained were satisfactory. These outcomes were evaluated by comparing the network's predictive performance against experimental tests that were not included during the training phase, thereby validating its generalization capability.

Figure (3) presents a comparative analysis between the experimental response corresponding to test number 26, executed with a constant PWM signal of 50.2, and the prediction generated by the artificial neural network. The vertical axis represents the angular position in degrees, while the horizontal axis denotes time in seconds. The experimental data exhibit a dynamic transient response characterized by a rapid increase in angular position, reaching a peak value slightly above 110°, followed by a gradual settling toward a steady-state value close to 90°. In contrast, the neural network prediction remains constant over time and approximates a steady angular position slightly above 90°, capturing the equilibrium value of the system without reflecting its dynamic behavior. This graphical representation allows for an assessment of the network's performance in estimating the final operating condition of the system under the given input conditions.

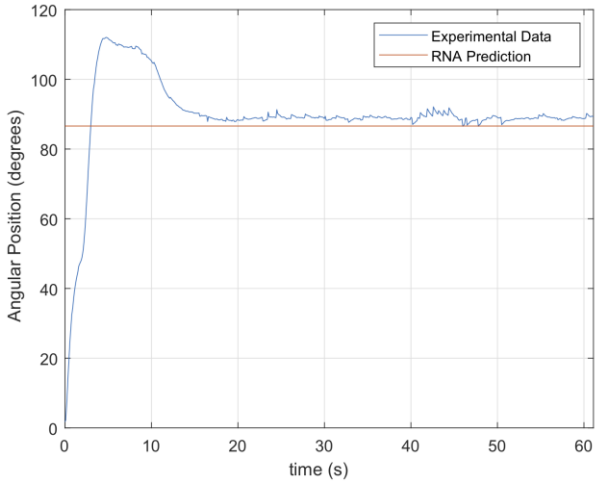

Fig. 3. Comparison between Experimental Response and Neural Network Prediction for Test 26 with Constant PWM Input.

Figure (4) presents a comparative analysis between the experimental angular position response and the predicted output generated by the artificial neural network for test number 28, which was conducted under a constant PWM input of 50. The experimental curve exhibits a dynamic transient followed by a gradual convergence toward a steady-state value, reflecting the typical behavior of the system under study. In contrast, the neural network prediction maintains a nearly constant value throughout the duration of the experiment, slightly deviating from the actual steady-state level observed in the measured data. This discrepancy highlights a minor limitation in the network's generalization capability for this specific test case, suggesting that while the model successfully captures the overall trend, it may require further adjustment to improve prediction accuracy under certain operating conditions.

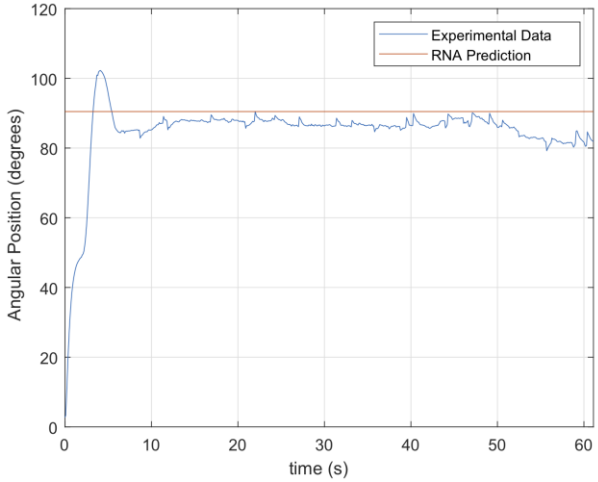

Fig. 4. Comparison between Experimental Response and Neural Network Prediction for Test 28 with Constant PWM Input.

Figure (5) illustrates the comparison between the experimental results and the response predicted by the artificial neural network (ANN) for the open-loop angular position control of the one-degree-of-freedom aeropendulum. As previously described, this test corresponds to experimental run number 46, in which a constant PWM input was applied to the system. The time axis spans from 0 to 60 seconds, while the vertical axis represents the angular position in degrees, ranging from 0 to 700. The experimental response is depicted by the blue curve, which shows an initial transient phase with a brief overshoot, followed by a stabilization around an angular position slightly below 100 degrees. This behavior reflects the physical constraints and natural dynamics of the aeropendulum reaching a quasi-steady state under the given input conditions. Conversely, the red curve corresponds to the ANN prediction, which remains constant at approximately 650 degrees throughout the entire duration of the test. The evident disparity between the predicted and actual responses underscores a significant error in the network's estimation. This suggests that, despite prior training, the ANN was unable to generalize effectively to this input scenario. Specifically, it failed to account for the system's inherent speed limitations and saturation effects, resulting in an overly optimistic and physically unrealistic output. This result reveals clear deficiencies in the network's modeling capabilities and indicates potential shortcomings in the training process, such as limited or unbalanced training data, inadequate model complexity, or assumptions that did not accurately reflect the system's behavior.

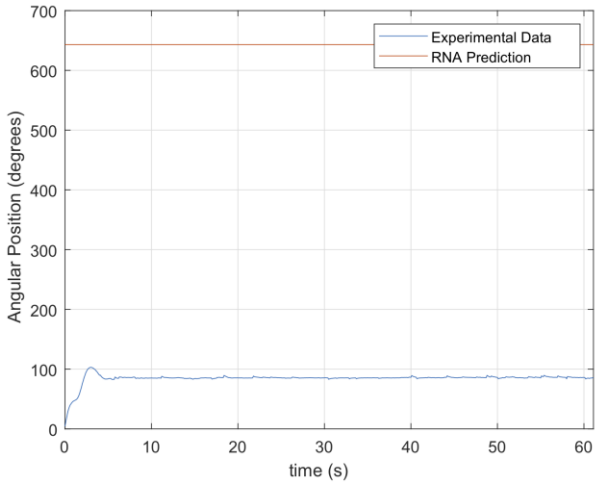

Fig. 5. Comparison between Experimental Response and Neural Network Prediction for Test 46 with Constant PWM Input.

In the case of Figure (6), a constant PWM signal of 49.3 was applied to the system to evaluate the performance of the trained artificial neural network (ANN) under open-loop control conditions. The resulting system behavior exhibited a clear convergence toward the desired target of 90°, with minimal steady-state error and a stable response over time. This outcome reflects a successful validation of the ANN's ability to model the nonlinear dynamics of the aeropendulum for this operating condition. The close agreement between the experimental trajectory and the ANN's predicted behavior confirms that, within certain ranges of the input space, the network can generate accurate estimations of the system's angular position. This supports the notion that the ANN effectively internalized the system's dynamic characteristics during training and was able to generalize this knowledge in real-world conditions. Overall, the results shown in Figure (6) provide strong evidence of the neural network's capacity to deliver reliable control predictions when operating within trained conditions, thus reinforcing its viability for open-loop control tasks in nonlinear systems such as the aeropendulum.

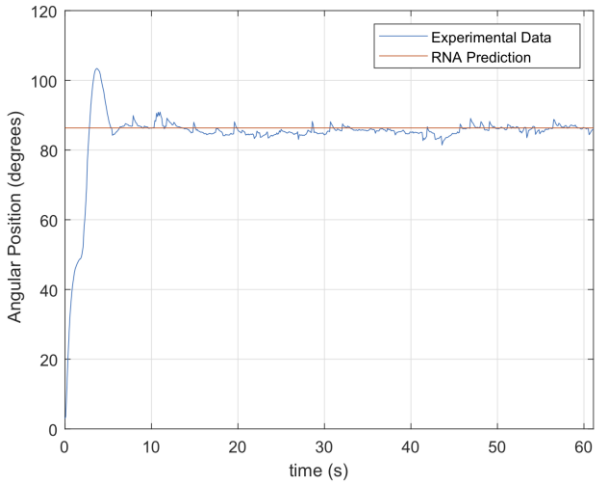

Fig. 6. Comparison between Experimental Response and Neural Network Prediction for Test 47 with Constant PWM Input.

Figure (7) presents the system response under a constant PWM input of 49.2, sustained throughout the entire test duration. This experimental setup was designed to further assess the consistency and reliability of the trained artificial neural network (ANN) in predicting the system's behavior under conditions slightly different from those previously tested. The results obtained exhibit a stable and repeatable system response, with the angular position converging toward a value close to the expected target. The minimal deviation observed reinforces the robustness of both the control strategy and the underlying ANN model. The system's ability to maintain predictable behavior under this configuration highlights the generalization capacity of the ANN and its effectiveness in capturing the nonlinear dynamics of the aeropendulum. Moreover, the consistency of the response across varying but comparable input values suggests that the ANN can serve as a reliable component in the control architecture, particularly in open-loop scenarios where no feedback correction is available. These findings provide additional support for the validity of the modeling and control approach adopted in this study, and they underline the potential of neural network-based methods for real-time control in nonlinear electromechanical systems.

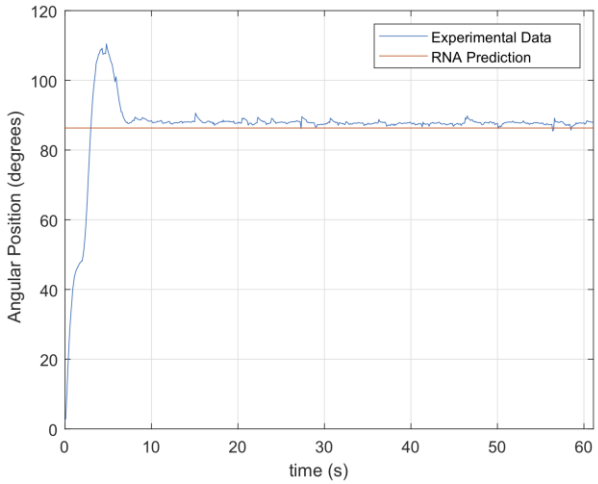

Fig. 7. Comparison between Experimental Response and Neural Network Prediction for Test 48 with Constant PWM Input.

Figure (8) illustrates the system response under the application of a constant PWM input of 49.5, with the objective of further validating the predictive accuracy of the artificial neural network (ANN) model. The results obtained in this trial are notably satisfactory, exhibiting a clear convergence of the angular position toward the reference value of 90 degrees. As observed in previous validation scenarios, the system demonstrates a stable and consistent response, with minimal oscillation and negligible steady-state error. The proximity of the experimental outcome to the expected target reinforces the model's capacity to accurately estimate the system's nonlinear dynamics under open-loop control conditions. The strong correlation between the predicted and actual behavior in this configuration emphasizes the precision and reliability of the ANN, particularly in capturing

the relationship between input signals and the resulting angular displacement. Furthermore, the result confirms the repeatability of the network's performance across different, yet closely spaced, PWM values, highlighting its robustness and practical applicability. Collectively, the findings shown in Figure 8 further validate the design and training approach adopted for the ANN, supporting its suitability for real-time control tasks in systems with nonlinear and complex dynamics such as the aeropendulum.

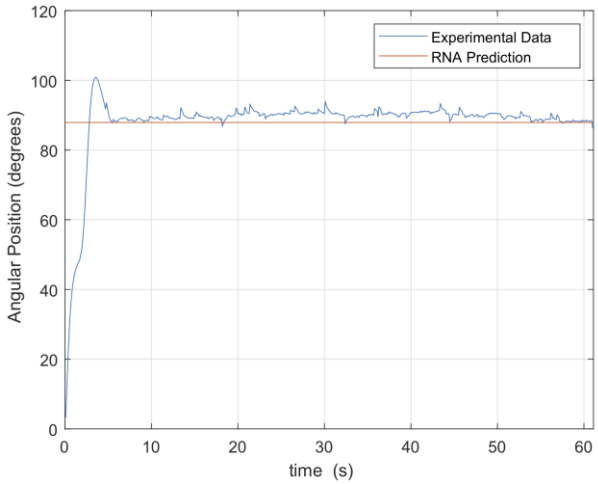

Fig. 8. Comparison between Experimental Response and Neural Network Prediction for Test 49 with Constant PWM Input.

Figure (9) presents the system's response under the application of a constant PWM signal of 49.7. In this case, the angular position successfully reached the target value of 90 degrees with high precision, demonstrating the effective performance of the trained artificial neural network (ANN) in this specific control scenario. The close alignment between the desired and actual inclination angle reinforces the ANN's capability to generalize accurately within the operating range for which it was trained. The minimal steady-state error and the absence of significant oscillations or instability further validate the model's robustness and fidelity in capturing the underlying nonlinear dynamics of the aeropendulum. This result not only confirms the predictive reliability of the ANN under slightly varied input conditions but also strengthens the overall conclusion that the neural network-based approach is a viable and effective method for open-loop control in nonlinear systems. The experiment depicted in Figure 9 thus serves as additional empirical evidence supporting the soundness of the design, training, and implementation strategy employed in the development of the ANN controller.

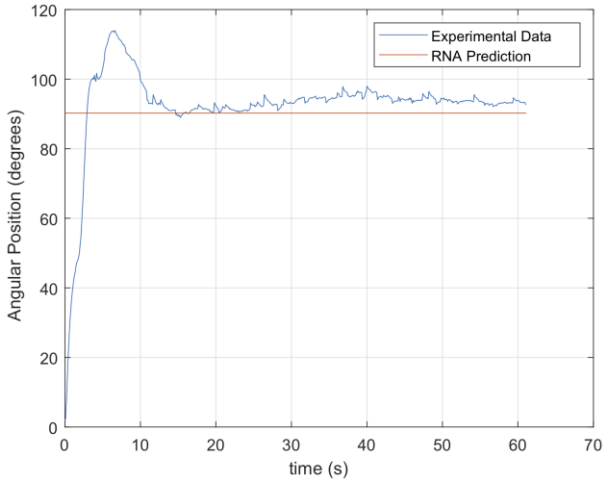

Fig. 9. Comparison between Experimental Response and Neural Network Prediction for Test 49 with Constant PWM Input.

The validation of the angular position prediction of the aeropendulum using an artificial neural network (ANN) provides strong evidence of the model's generalization capabilities. Through the experimental results, it is demonstrated that the ANN is capable

not only of accurately estimating the system's output for various constant PWM inputs, but also of doing so across a range of operating conditions within the trained domain. This capacity for generalization is fundamental, as it enables the implementation of an open-loop control strategy based solely on the network's learned input-output mapping. In other words, the ANN can generate an effective control signal to drive the system toward a desired angular position without requiring real-time feedback of the output signal or the computation of an explicit error term. This represents a significant advantage in scenarios where sensor feedback is limited, costly, or infeasible. Furthermore, the observed consistency between predicted and experimental behaviors across multiple validation cases confirms that the neural model encapsulates the essential dynamics of the nonlinear system with sufficient accuracy. As a result, the ANN-based approach becomes a viable and efficient alternative for real-time control in applications where conventional feedback-based methods may not be practical.

4 Conclusions

The implementation of artificial neural networks for the control of a seesaw system actuated by a thrust propeller in an open-loop configuration has yielded a set of diverse and insightful results. Experimental trials and detailed data analysis have revealed that the performance of the neural network is highly dependent on the specific conditions under which the system operates. One of the most relevant observations was the presence of a significant discrepancy between the predictions generated by the neural network and the actual measurements in certain scenarios. This divergence underscores the importance of thoroughly considering the system's dynamic characteristics and environmental variables during both the design and training phases. Nevertheless, under alternative configurations, the approach proved to be highly effective. For instance, when a particular constant was employed, the system demonstrated a considerable convergence toward the target inclination of 90 degrees, suggesting successful learning and control behavior. Furthermore, in a different experimental case, the system exhibited robust and predictable behavior under another constant value, validating the effectiveness of the adopted control strategy. In this instance, there was a high degree of consistency between theoretical expectations and the practical responses of the system, reinforcing the reliability of the methodology. In an additional scenario, the use of a specific constant yielded highly satisfactory results, once again highlighting the neural network's capability to accurately regulate the position of the rocking mechanism. These findings emphasize the importance of selecting appropriate training parameters and system configurations to maximize control precision. Overall, the results obtained reflect the relevance of adapting the neural network model to the characteristics and operating conditions of the system. While certain limitations were observed in specific cases, the favorable outcomes in others underscore the potential of neural networks to enhance the control of nonlinear and dynamically sensitive systems. These findings reinforce the multidisciplinary nature of control engineering and the practical viability of integrating intelligent computational strategies to address complex technical challenges.

References

Armenta, M. A. M. (2022). *Control de robots manipuladores usando redes neuronales*.

Catlin, D. E. (2012). *Estimation, control, and the discrete Kalman filter* (Vol. 71). Springer Science & Business Media. https://doi.org/10.1007/978-1-4612-4528-5

Giraldo Quintero, E. A., & Herrera Gil, S. (2018). *Módulo didáctico de control de posición de un balancín eólico*.

Grewal, M. S., & Andrews, A. P. (2014). *Kalman filtering: Theory and practice with MATLAB*. John Wiley & Sons.

Kumari, N., Kulkarni, R., Ahmed, M. R., & Kumar, N. (2021). Use of Kalman filter and its variants in state estimation: A review. In *Artificial Intelligence for a Sustainable Industry 4.0* (pp. 213–230). Springer. https://doi.org/10.1007/978-3-030-77070-9_13

Levine, D. S., & Aparicio IV, M. (2013). *Neural networks for knowledge representation and inference*. Psychology Press.

Morales-Narciso, L. M., Mendoza-Reyes, I., Romero-Flores, R. N., Calderón-López, J. V., López-García, A. M., Flores-Álvarez, M. A., … Ordaz, P. (2023, November). Design, construction and control of a seesaw system driven with a thrust propeller. In *2023 XXV Robotics Mexican Congress (COMRob)* (pp. 122–127). IEEE. https://doi.org/10.1109/COMRob60035.2023.10349707

Prieto, A., Prieto, B., Ortigosa, E. M., Ros, E., Pelayo, F., Ortega, J., & Rojas, I. (2016). Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing, 214*, 242–268. https://doi.org/10.1016/j.neucom.2016.06.014

Zakaria, M., Mabrouka, A. S., & Sarhan, S. (2014). Artificial neural network: A brief overview. *Neural Networks, 1*, 2.