Two Months Internship Report

On

"Introduction to artificial neural networks in control applications"

Submitted by

Miss. SHINILA M F

from

Department of Electrical Engineering and Computer Science

Inian Institute of Science Education and Research Bhopal

Under the Supervision

of

Dr. Chandrashekhar Sakode

Assistant Professor

Department of Electronics & Communication Engineering

Indian Institute of Information Technology, Nagpur

Indian Institute of Information Technology, Nagpur

(An Institution of National Importance by an Act of Parliament)

Nagpur– 441108, India

❖ ABSTRACT

This report presents a comparative analysis of two distinct control strategies—an Artificial Neural Network (ANN)-based feedforward-feedback control system and a traditional Proportional-Integral-Derivative (PID) control system—applied to a simulated second-order system. The ANN model, designed with a single hidden layer, was trained to approximate the desired control signal by using backpropagation on data generated from a PID controller, which served as a reference model. The PID controller was tuned using specific gain values (Kp, Ki, Kd) to ensure effective system control.

The first implementation, referred to as NeuralNetwork1, focuses on digit recognition. The network was trained on a dataset containing binary representations of the digits 0 through 9. It consists of an input layer, a single hidden layer, and an output layer, utilizing the sigmoid activation function and backpropagation for learning. The network was trained over 1000 epochs, and performance was tested using a sample input representing the digit 3.

The second implementation, NeuralNetworkV1, was designed to approximate the cosine function. This network, featuring a single hidden layer with 22 neurons and sigmoid activation, was trained using inputs scaled from the range $[0, \pi]$ to $[0,1]$ and outputs scaled to $[0,1]$. The training involved iterating over 25 samples, followed by testing across 50 evenly spaced points in the range $[0, \pi]$ to evaluate the network's performance against the actual cosine function values.

Control systems are essential in various engineering applications, ensuring desired performance and stability of dynamic systems. Traditional methods, such as PID controllers, are known for their simplicity and effectiveness but often struggle with non-linear, time-varying systems and require precise tuning. In contrast, ANNs offer an adaptive, model-free approach capable of handling non-linearity and uncertainties effectively. The project is structured into several phases, beginning with the creation and training of an ANN-based control system designed to manage a second-order system. The training process involves simulating the second-order system under various conditions and adjusting the ANN's parameters to optimize performance. The ANN's architecture and training algorithm are specifically tailored to achieve robust control capabilities. The report concludes with a comparison of the ANN and PID control systems based on their response to a step reference input. Performance metrics, such as the smoothness and stability of the output response, were analysed. The results demonstrated that the ANN control system achieved superior performance, delivering a smoother control output with minimal overshoot and oscillations compared to the PID controller. This study underscores the potential of neural networks in adaptive control applications, especially in systems with complex or unknown dynamics, while acknowledging the robustness and widespread applicability of PID controllers in many industrial settings.

❖ **TABLE OF CONTENTS**

❖ LEARNING OBJECTIVES

The internship was designed to provide hands-on experience in the practical application of artificial neural networks (ANNs) and advanced control systems. The focus was on two main projects, each with specific objectives and goals that contributed to development in machine learning, artificial intelligence, and control systems with a particular focus on simulating and designing control strategies for dynamic systems.

1. Digit Recognition Using Neural Networks

- **Objective:** To develop a neural network model capable of recognizing handwritten digits from 0 to 9, using binary pixel data as input and learn architecture and structure of ANN.
- **Goal:** To gain practical experience in building, training, and evaluating neural networks for classification tasks. This project aimed to enhance the understanding of the design and implementation of feedforward neural networks and to assess the effectiveness of these models in pattern recognition.

2. Function Approximation Using Neural Networks

- **Objective:** To design a neural network capable of approximating the cosine function over the interval $[0, \pi]$.
- **Goal:** To explore the use of neural networks for continuous function approximation. The project sought to deepen the intern's knowledge in training neural networks for regression tasks, develop skills in data scaling, loss minimization, and model evaluation.

3.Neural Network-Based Control Systems

- **Objective:** To design and implement an ANN capable of controlling a second-order dynamic system, such as an industrial process or mechanical system, by mimicking a traditional PID control approach and to compare the performance of the ANN-based control system with a traditional Proportional-Integral-Derivative (PID) control system, both in simulation and potential real-world applications.
- **Goal:** To understand the principles of ANN-based control, including the design, architecture, training, and implementation of a neural network for system control and to understand PID tunning also various methods to tune PID. This involved exploring feedforward and feedback control mechanisms within the ANN architecture and PID gains to evaluate the effectiveness, accuracy, and response time of ANN control compared to conventional methods. This included understanding the nuances of PID control and the potential advantages and limitations of using ANNs in control applications.

❖ INTRODUCTION

This internship focused on applying artificial neural networks (ANNs) to solve real-world problems in computer vision, pattern recognition, and control systems. The work was structured around two main projects: "Digit Recognition Using Neural Networks" and "Function Approximation Using Neural Networks." Additionally, the internship explored advanced control systems through the development and implementation of ANN-based control systems, comparing them with traditional Proportional-Integral-Derivative (PID) controllers and understand PID tunning.

Problem Statement and Scope of Work

Digit Recognition Using Neural Networks

The task of handwritten digit recognition is a well-known challenge in computer vision and pattern recognition. It involves identifying a digit from an image of handwritten text, a problem relevant to practical applications such as postal mail sorting, check processing, and digitization of handwritten documents. Traditional methods often rely on handcrafted feature extraction, which can be cumbersome and less effective in handling diverse handwriting styles. In contrast, neural networks, especially deep learning models, can automatically learn and extract hierarchical representations from raw pixel data, offering a more robust solution.

The scope of this project included:

1. Designing a neural network with an input layer to receive flattened pixel data from digit images.
2. Implementing hidden layers to capture intermediate representations.
3. Utilizing an output layer with nodes representing the ten possible digit classes (0-9).
4. Training the network on a labelled dataset of digit images.
5. Evaluating the network's generalization capabilities on unseen or seen data.

Function Approximation Using Neural Networks

Function approximation involves finding a function that closely matches a target function within a specified domain. In this project, the target function was the cosine function over the interval $[0,\pi]$. Traditional methods like polynomial fitting or Fourier series can require extensive domain knowledge and lack flexibility. Neural networks provide an alternative approach by learning the underlying function directly from input-output pairs without needing explicit functional forms.

The scope of this project included:

1. Defining the input as a scaled version of the cosine function's argument.
2. Designing a feedforward neural network with a single hidden layer.
3. Training the network to minimize the error between predicted and actual cosine values.
4. Evaluating the model's accuracy and generalizability over the defined interval.

Neural Network-Based Control Systems

In industrial and engineering applications, accurate and efficient control of dynamic systems is crucial. While traditional methods like PID controllers are widely used due to their simplicity and effectiveness, they often require manual tuning and struggle with complex, nonlinear systems. This project explored using ANNs for control, leveraging their learning and adaptive capabilities.The problem addressed was designing an ANN-based control system to regulate a second-order dynamic system, characterized by inertia and damping properties. The ANN was trained to replicate the control actions of a simulated reference model and learn PID tunning and methods to fine tune PID.

The scope of work included:

1. Designing an ANN architecture suitable for learning control strategies.
2. Training the ANN using data from a simulated reference control system.
3. Implementing the trained ANN in a simulation environment to test its control capabilities.
4. Understanding PID tunning and methods to tune PID.
5. Evaluating the ANN-based control system's performance compared to a traditional PID controller.

Methodologies for Solving the Problems

A Neural Network can be trained with labelled digit images to learn and classify patterns. Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs). K-Nearest Neighbours (KNN), Decision Trees and Random Forests, Principal Component Analysis (PCA), Reinforcement Learning algorithms are methods to recognize the digits and knowing the efficiency .

PID Control: A PID controller uses proportional, integral, and derivative gains to minimize the error between the desired and actual outputs, providing a straightforward and effective control strategy for many systems. Neural Networks: A neural network can learn the dynamics of the system through training, allowing it to predict the control actions required to achieve the desired output based on input data. Polynomial Regression, Support Vector Regression (SVR) utilizes kernel functions to handle non-linear relationships. Fourier Series represents periodic functions as sums of sine and cosine terms, ideal for functions like the cosine function. Gaussian Processes provide a probabilistic approach to function approximation, offering predictions with uncertainties. Model Predictive Control (MPC), Adaptive Control, Fuzzy Logic Control, Reinforcement learning can solve the problem.

Contribution Proposed by the Student

The student's contributions included:

1. A neural network architecture for digit recognition, specifically tailored for binary pixel data.
2. A simple yet effective neural network design for function approximation, demonstrating the versatility of neural networks.
3. A systematic approach to training and evaluating neural networks, including hyperparameter selection, training iterations, and error analysis.
4. Thorough documentation of the implementation process, challenges encountered, and solutions developed, providing a practical understanding of neural network applications.

Additionally, in the control systems domain:

1. Development of an ANN-based control system that effectively managed a second-order dynamic system, offering an alternative to traditional control methods.
2. Learning tunning parameters of PID and also learn various method to tune PID to obtain the desired output.
3. A comprehensive evaluation comparing ANN-based control with PID control, highlighting the strengths and potential improvements of using ANNs.
4. Detailed documentation and analysis of methodologies, training processes, and evaluation metrics, contributing valuable insights for future research in control systems.

This work provided a foundation for further exploration of neural networks in various applications, showcasing their potential and limitations in practical scenarios.

❖ LITERATURE SURVEY

Artificial Neural Networks:

Artificial Neural Networks (ANNs) have been a focal point of machine learning research for several decades. Their ability to learn complex patterns and approximate non-linear functions makes them suitable for a wide range of applications, including image recognition, natural language processing, and time-series prediction.

Function Approximation Using ANNs:

Function approximation is one of the fundamental applications of ANNs. The ability of neural networks to approximate continuous functions has been extensively studied. Early works by Cybenko (1989) and Hornik (1991) established the theoretical foundation, proving that feedforward networks with a single hidden layer can approximate any continuous function to any desired degree of accuracy, given sufficient neurons.

Relevant Research-

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2(4), 303-314.

This seminal paper proved that a feedforward network with a single hidden layer using a sigmoidal activation function can approximate any continuous function on a compact subset of

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2), 251-257.

Hornik extended Cybenko's results to show that multilayer feedforward networks are universal approximators.

Pattern Recognition Using ANNs:

Pattern recognition is another crucial application of ANNs. Neural networks have demonstrated significant success in recognizing handwritten digits, as evidenced by the MNIST dataset benchmark. The ability of ANNs to learn from examples and generalize to unseen data is the key to their effectiveness in this domain.

Relevant Research-

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

This paper introduced the MNIST dataset and demonstrated the effectiveness of convolutional neural networks (CNNs) in handwritten digit recognition.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), 958-963.

This literature survey provides a comprehensive review of the research on Proportional-Integral-Derivative (PID) controllers, focusing on various tuning methods and their effectiveness. The survey also explores the intersection of PID control with artificial neural networks (ANNs) in control system design. The aim is to offer insights into the tuning techniques for PID controllers and compare them with ANN-based methods for control applications.

PID Control Systems:

PID controllers are a cornerstone of control system design due to their simplicity and effectiveness. They adjust the control inputs based on the proportional, integral, and derivative terms of the error signal to achieve the desired system output.

Fundamental Works-

1. Åström, K. J., & Hägglund, T. (1995). "PID Controllers: Theory, Design, and Tuning."

- This seminal work provides a detailed exploration of PID controller theory, design strategies, and tuning methodologies. It is a fundamental reference for understanding the principles behind PID control and its practical applications.

3. Åström, K. J., & Murray, R. M. (2008). "Feedback Systems: An Introduction for Scientists and Engineers."

- This book offers an extensive overview of feedback control systems, including PID controllers. It covers various design and analysis techniques and provides practical examples.

PID Tuning Methods: Classical Tuning Methods

1. Ziegler-Nichols Method

- This heuristic method involves setting the integral and derivative gains to zero, increasing the proportional gain until the system exhibits sustained oscillations, and then using the frequency and amplitude of these oscillations to set the PID parameters. The method is known for its simplicity and practical applicability but may not always provide optimal results.

2. Cohen-Coon Method

- An improvement over the Ziegler-Nichols method, the Cohen-Coon method provides more systematic tuning rules based on the open-loop step response of the system. It is particularly useful for processes with significant dead time.

Modern Tuning Techniques

1. Model-Based Tuning

- Model-based methods use mathematical models of the system to predict the impact of different PID parameters on system performance. Techniques such as the Internal Model Control (IMC) approach and optimization-based methods fall into this category.

2. Genetic Algorithms

- Genetic algorithms are optimization techniques inspired by natural evolution. They have been applied to PID tuning by evolving a population of potential solutions (PID parameters) based on their performance, selecting the best-performing solutions over generations.

3. Adaptive and Intelligent Tuning

- Adaptive tuning methods adjust PID parameters in real-time based on changing system dynamics. Intelligent tuning approaches, including fuzzy logic and neural networks, can also be used to optimize PID parameters dynamically.

Relevant Research on PID Tuning

1. Astrom, K. J., & Hagglund, T. (1995). "PID Controllers: Theory, Design, and Tuning."

   - Provides a comprehensive overview of classical and modern PID tuning methods.

2. Cohen, G. H., & Coon, G. A. (1953). "Theoretical Considerations of Retarded Control."

   - Introduces the Cohen-Coon method, offering systematic rules for PID tuning based on system response.

Neural Network-Based Control Systems:

Neural networks have emerged as powerful tools for modelling and controlling complex, nonlinear systems. Their ability to learn and adapt to system dynamics offers potential advantages over traditional control methods.

Key Research and Developments

1. Narendra, K. S., & Parthasarathy, K. (1990). "Identification and Control of Dynamical Systems Using Neural Networks." IEEE Transactions on Neural Networks.

   - Demonstrates the application of neural networks for system identification and control, showing their potential for handling nonlinear and time-varying systems.

2. Hunt, K. J., & Sbarbaro, D. (1991). "Neural Networks for Nonlinear Internal Model Control." IEE Proceedings D - Control Theory and Applications.

   - Presents a neural network-based approach for internal model control, highlighting improved performance over traditional methods.

Comparative Studies: PID vs. Neural Network-Based Control:

1. Guerra, T. M., & Glorennec, P. Y. (1998). "Fuzzy Sets and Systems: Theory and Applications." Academic Press.

   - Compares traditional control methods, including PID, with advanced approaches such as fuzzy logic and neural networks, highlighting the strengths and weaknesses of each.

2. Wang, L. (2009). "Adaptive Fuzzy Systems and Control: Design and Stability Analysis." Prentice Hall.

   - Explores adaptive control techniques, including neural network-based methods, and compares their performance with conventional PID controllers.

3. Draeger, A., Engell, S., & Ranke, H. (1995). "Model Predictive Control Using Neural Networks." IEEE Control Systems Magazine.

   - Presents a comparative study of model predictive control (MPC) and neural network-based control, demonstrating the advantages of ANNs in handling complex system dynamics.

❖ METHODOLOGY

Neural Network for Digit Recognition

1. Initialization

- **Libraries**: The numpy library is utilized for numerical computations essential for implementing the neural network.

2. Dataset

Input Data

The input data consists of binary matrices of size 11x9, where each matrix represents a digit. The matrix elements are binary, with '1' indicating the presence of a stroke and '0' indicating its absence. The input dataset X contains 10 such matrices, each representing the digits from 0 to 9.

Output Data

The output data y is structured as a one-hot encoded matrix. Each row represents a digit, with the corresponding position set to 1 and others set to 0. For instance, the digit '0' is represented as [1, 0, 0, 0], and the digit '1' as [0, 1, 0, 0].

3. Neural Network Architecture

- **Input Layer:** This layer receives a 99-dimensional input vector derived from flattening the 11x9 matrix.
- **Hidden Layer:** Comprising 7 neurons, this layer uses sigmoid activation functions to capture intermediate representations.
- **Output Layer:** This layer has 4 neurons with sigmoid activation functions, producing a 4-dimensional output vector corresponding to the one-hot encoded output.

4. Implementation

- **Sigmoid Function**

The sigmoid function is used as the activation function for the neurons is defined as:

$$sigmoid(x) = \frac{1.0}{1+e^{-x}}$$

- **Derivative of Sigmoid Function**

The derivative of the sigmoid function used during backpropagation is defined as:

$$sigmoid\_der(x) = x.(1.0 - x)$$

- **Neural Network Class**

The neural network is implemented in the NeuralNetwork1 class. The class contains the following methods:

- **'__init__'**: Initializes the network with random weights for the input-hidden and hidden-output connections. It also sets up the input and output data.
- **'training'**: Performs one epoch of training using feedforward and backpropagation. The weights are updated based on the error gradients.
- **'__call__'**: Feeds an input vector through the network and computes the network's output for a given input vector.

5. Data Preparation

- **Input Data**: Define a 10x99 array X representing 10 different digit patterns.
- **Output Data**: Define a 10x4 array y representing the desired output in binary format.

6. Neural Network Training

- **Instantiation:** An instance of NeuralNetwork1 is created with 7 hidden neurons.
- **Training:** The network undergoes 1000 epochs, where each epoch involves:
  - **Feedforward:** Calculation of activations for the hidden and output layers.
  - **Backpropagation:** Computation of output and hidden layer errors, followed by weight updates.

7. Testing

- **Input Data for Testing**: A new input pattern representing the digit '3' is provided.
- **Prediction**: The trained network predicts the output for this input, and the result is printed.

Neural Network for Cosine Approximation

1. Initialization

- **Libraries**: numpy library is used for the necessary numerical computations.

2. Dataset

Input Data

The input data consists of single values normalized between 0 and 1, representing the cosine function arguments scaled to the range of [0, π].

Output Data

The output data is also normalized between 0 and 1, corresponding to the cosine function values over the specified range.

3. Neural Network Architecture

The neural network consists of three layers:

1. **Input Layer**: Handles the normalized input vector.
2. **Hidden Layer**: Comprises 22 neurons, using sigmoid activation functions to process the data.
3. **Output Layer**: Consists of a single neuron producing a continuous output value.

4. Implementation

- **Sigmoid Function**: Define the sigmoid activation function and its derivative sigmoid_der.
- **Neural Network Class**: The neural network is implemented in the NeuralNetworkV1 class. The class contains the following methods:

  '__init__': Initializes the network with random weights for the input-hidden and hidden-output connections. It also sets up the input and output data.

  'training': Performs one epoch of training using feedforward and backpropagation. The weights are updated based on the error gradients.

  '__call__': Feeds an input vector through the network and returns the output vector.

5. Data Preparation

- **Input Data:** A 1x1 array 'inp' is defined with an initial value.
- **Output Data**: A 1x1 array 'oup' is set with an initial value.
- **Scaling Factors**: 'xScale' and 'yScale' are defined to normalize input and output values.
- **Number of Samples**: The dataset includes 25 samples, with 1000 training iterations.

6. Neural Network Training

- **Instantiation**: Create an instance of 'NeuralNetworkV1' with 22 hidden neurons.

Training: Train the network by iterating over the range of cosine function values for a specified number of samples and iterations. During each epoch, the following steps are performed:

1. **Feedforward**: Compute the activations of the hidden layer and the output layer.

2. **Backpropagation**:

    - Compute the output layer error and delta.
    - Compute the hidden layer error and delta.
    - Update the weights of the hidden-output and input-hidden connections.

4. Testing

- **Testing Range**: A range for the cosine function is defined for testing.
- **Prediction**: Use the trained network to predict the output for each input in the testing range and compare it with the actual cosine values

Neural Network-Based Control System

Methodology

1. Importing Libraries

The necessary libraries are imported to support the project:

- **math**: Provides mathematical functions.
- **numpy**: Supports large, multi-dimensional arrays and matrices, along with a collection of mathematical functions.
- **matplotlib.pyplot**: Used for plotting graphs.

2. Sigmoid Function and Its Derivative

The sigmoid function and its derivative are crucial components for the neural network operations:

- **Sigmoid function**: $sigmoid(x) = \frac{1}{1+e^{-x}}$. This function computes the sigmoid of x.
- **Sigmoid derivative**: $sigmoid\_der(x) = x \cdot (1 - x)$ .This function computes the derivative of the sigmoid function, used for backpropagation.

3. Neural Network Class

A simple feedforward neural network with backpropagation is implemented in the ‘NeuralNetworkV1' class.

Neural networks consist of the following components:

• one input layer, x,

• one or more hidden layers,

• one output layer, ŷ,

• a set of weights and biases between each layer, W and b,

• activation function for each hidden layer, σ.

- **Initialization**:
    - Random weights are assigned for input-to-hidden (‘W1’) and hidden-to-output (‘W2’) connections.
    - Input (‘inp’) and desired output (‘y’) vectors are initialized.
    - A constant k is used for the derivative calculation of the loss function.
- **Training Method**: A process of fine-tuning of the weight coefficients from the input and hidden layers nodes is known as training the neural network. The training method performs the following steps:
    - **Feedforward**:
        - Compute hidden layer values using the sigmoid function.
        - Compute output layer values using the sigmoid function, resulting in the predicted output ŷ.
    - **Backpropagation**:
        - Calculate the output error and its delta.
        - Calculate the hidden layer error.
        - Update weights (W2 and W1) using the calculated deltas and corrections to minimize the overall error at the output.
- **Call Method**: The ‘ __call__’ method computes the output of the network for a given input.

4. Control System Components

The control system integrates several components:

- **Feedforward + Feedback Control System**:
    - **Feedforward Filter for Setpoint (Fy)**: A filter for processing the setpoint.
    - **PID Controller (Pid)**: A traditional PID controller.
    - **Feedforward Filter for Control Signal (Fu)**: A filter for processing the control signal.

  - **Plant (plant)**: The second-order system model. The control system operation involves computing the error E, the control signal U, and the output Y.

5. System Classes

- **SecondOrderSystem**: Simulates a second-order system with damping parameters d1, d2, and an output limit L, computing the system output for a given input.
- **FirstOrderSystem**: Simulates a first-order system with a damping parameter d, computing the system output for a given input.
- **PIDControlBlock**: Implements a PID controller with proportional (kp), integral (ki), and derivative (kd) gains, computing the control output for a given error.
- **HighPassFilter**: Implements a high-pass filter with a damping parameter d, computing the filter output for a given input.
- **FFplusFBsystem**: Combines feedforward and feedback control strategies using the aforementioned systems and controllers, computing the control output for a given input.

6. Parameter Setup

Various parameters are defined for training and simulation:

- maxVal, errScale, uOffset, uScale: Scaling factors and offsets.
- trainings, samples: Number of training iterations and simulation samples.
- trainVal, stepVal: Training and step values.
- T1, T2, Ty, Tu: Time constants for the systems.
- Kp, Ki, Kd: PID controller gains.

7. PID Tunning

Theoretical Calculation of PID Parameters

The initial theoretical values of Kp, Ki, and Kd were calculated using the system's time constants T1 and T2:

1. **Calculation of Ki**:
   - The integral gain Ki was estimated using the formula $1/Ki \approx 50$.
   - This provided a reasonable starting point for the integral action.
   - Calculated value: $Ki=0.02$

2. **Calculation of Kp**:
    - The proportional gain Kp was calculated using the formula Kp =Ki×(T1+T2)
    - This ensures that the proportional action is balanced with respect to the system's overall response time.
    - Calculated value: Kp=7.
3. **Calculation of Kd**:
    - The derivative gain Kd was calculated using the formula Kd=Ki×T1×T2
    - This provides appropriate damping relative to the system's inertia.
    - Calculated value: Kd=500

Empirical Tuning Process

The theoretical values served as a starting point. Fine-tuning was performed through incremental adjustments based on the system's response:

1. **Starting with Initial Calculations**:
    - The theoretically calculated values were used as the initial settings.
2. **Incremental Adjustments**:
    - **Proportional Gain (Kp)**: Increased if the system response was too slow, decreased if the system became too oscillatory and meet the desired point.
    - **Integral Gain (Ki)**: Adjusted to eliminate steady-state error without causing excessive oscillations or instability.
    - **Derivative Gain (Kd)**: Increased to reduce overshoot and improve damping, decreased if the system became too sensitive to noise or too oscillatory.
3. **System Response Analysis**:
    - Used tools like step response, frequency response analysis, and simulation to observe the effects of parameter changes.
4. **Iteration**:
    - Repeated the process until the desired performance was achieved, balancing rise time, overshoot, settling time, and steady-state error.

Manual Tuning Procedure

Tuning Steps

1.**Initial Setup:**

Start with the theoretical values: Kp = 7, Ki = 0.02, Kd = 500.

2.**Decrease Proportional Gain (Kp):**

Gradually decrease Kp from 7 toward 2. Observe the system's response and continue adjusting until the system reaches the set point with a reasonable rise time and minimal oscillations. The goal is to find a Kp value that provides a responsive system without excessive overshoot.

3.**Add and Adjust Integral Action (Ki):**

After setting Kp, decrease Ki from 0.02 to 0.004. This will help eliminate the steady-state error. Adjust Ki carefully to prevent the system from becoming too sluggish or unstable such that plot appears below set point. The target is to achieve minimal steady-state error while maintaining stability.

4.**Add and Adjust Derivative Action (Kd):**

Finally, decrease Kd from 500 to 175. The derivative term helps to dampen oscillations and reduce overshoot. Adjust Kd to improve the damping effect and smooth the system's response with perfect curve. Be cautious with Kd adjustments, as too high a value can led to noise amplification.

Ziegler-Nichols Method

The Ziegler-Nichols tuning method was also used to obtain initial estimates for Kp , Ki , and Kd :

1. **Determine the Ultimate Gain (Ku) and Ultimate Period (Tu)**:

   Increased Kp until the output exhibited sustained oscillations.

   Recorded this gain as Ku and the period of oscillation as Tu.

2. **Calculate Kp, Ki, and Kd**:

Used the Ziegler-Nichols tuning rules to set the PID parameters: Kp=0.6Ku, Ki=2Kp, Kd=KpTu/8

3. **Fine-tuning**:

Performed manual adjustments to optimize performance

8. ANN Training

An ANN (‘Ann1’) is created and trained using the following steps:

1. **Initialization**:

- The input (inp) and output (oup) arrays are initialized with zeros.

- A neural network instance (Ann1) is created with 11 neurons.

2. **Training Loop**:

- The training process is iterated for a specified number of trainings (1000 iterations).
- Within each iteration, the following components are initialized:

▪ **SecondOrderSystem** (SOSref): Simulates the plant with given parameters.

▪ **PIDControlBlock** (PIDref): Implements the PID controller.

▪ **FirstOrderSystem** (FINref): Represents a first-order system.

▪ **HighPassFilter** (FUNref): Implements a high-pass filter.

▪ **FFplusFBsystem** (RefContrSys): Combines the above components to form the reference control system.

o For each training iteration, the control system is simulated for a specified number of samples (1000 samples):

▪ The control system output Y and control signal U are obtained.

▪ The input to the ANN is computed as the scaled error between the training value and the system output.

▪ The output of the ANN is computed as the scaled control signal.

▪ The ANN is trained using the computed input and output.

9. ANN-Based Control System

The AnnFbFfControl class is implemented to create a control system using the trained ANN and predict the control signal for the plant.

1. **Class Definition**:

    o The AnnFbFfControl class is defined to implement the ANN-based control system.

    o The class constructor initializes the plant, ANN, and other necessary variables.

    o The __call__ method implements the control logic, including:

        ▪ Scaling and offset adjustments for the input.

        ▪ Computation of the ANN output.

        ▪ Computation of the control signal U.

        ▪ Obtaining the plant output Y.

2. **Create and Test ANN-Based Control System**:
    - The ANN-based control system is created and tested using a second-order system.

10. PID Control System

A traditional PID control system is instantiated for comparison using the PIDControlBlock class.

11. Simulation and Comparison

- **Setpoint**: A constant setpoint of 750.0 is used for both control systems.
- **Samples**: Both systems are simulated over 1000 samples.

Both control systems (ANN-based and PID) are simulated, and their outputs are stored for plotting. The following steps are performed:

1. **Initialize lists** to store the control outputs.
2. **Simulate both control systems** for the defined number of samples.
3. **Print the output values** for each sample.

- **Simulation Loop**:
    - For each sample:
        - The ANN-based control system output is computed using Ann1ContrSys.
        - The PID control system output is computed using PIDContrSys.

12. Results and Plotting

The results of the simulations are plotted using matplotlib to visually compare the performance of the ANN-based control system against the PID control system.

1. **Initialization**:
    - The SecondOrderSystem instance (Plant1) is created.
    - The ANN-based control system (Ann1ContrSys) is instantiated.
    - The PID control system (PIDContrSys) is instantiated.
2. **Simulation**:
    - The control systems are simulated for a specified number of samples (1000 samples).
    - For each sample, the ANN and PID control system outputs are computed.
    - The outputs are stored for plotting and comparison.
3. **Results Plotting**:

- The results of the ANN and PID control systems are printed and plotted over the samples.
- The setpoint value is also plotted for reference.

❖ RESULTS AND DISCUSSION

1. Objectives Achieved

Neural Network-Based Digit Recognition

Objective: Develop an Artificial Neural Network (ANN) for digit recognition.

Achievement: The ANN, referred to as NeuralNetwork1, was successfully trained to recognize digits from 0 to 9 using a dataset composed of binary pixel matrices. The training process involved adjusting the weights between the input, hidden, and output layers to minimize the error between the predicted outputs and the actual labels. After 1000 epochs, the network demonstrated the capability to generalize and recognize digit patterns, confirming its ability to learn from the provided data.

Neural Network-Based Function Approximation

Objective: Utilize an ANN to approximate the cosine function over the interval [0,π][0, \pi][0,π].

Achievement: The second ANN, NeuralNetworkV1, was trained to approximate the cosine function. Input values, normalized between 0 and 1, represented the arguments of the cosine function, and the outputs were similarly scaled. After 1000 training iterations, the network could accurately predict cosine values, demonstrating a good fit to the true function. The network's performance was validated with separate test inputs, confirming its generalization ability.

2. Skills Learned

Scientific Skills

1. **Machine Learning and Neural Networks**
    - Acquired a robust understanding of neural network architectures and the mechanics of training neural networks using feedforward and backpropagation algorithms.
    - Developed expertise in hyperparameter tuning, including selecting the number of hidden neurons and adjusting the learning rate.
2. **Data Preprocessing and Normalization**
    - Mastered techniques for scaling and normalizing data, crucial for effective neural network training.
    - Gained insights into the impact of data representation on model performance.
3. **Performance Evaluation**
    - Enhanced ability to evaluate model performance using metrics such as mean squared error and accuracy.

- Understood the importance of model validation with test data to ensure generalization.

Professional Skills

1. **Problem-Solving**
   - Refined problem-solving abilities by identifying and resolving issues encountered during model training and implementation.
   - Developed troubleshooting skills and optimized neural network models for better performance.
2. **Project Management**
   - Learned to manage the entire project workflow, from data collection and preprocessing to model development and evaluation.
   - Improved time management and task prioritization skills to meet project deadlines.
3. **Technical Communication**
   - Enhanced capabilities in documenting and presenting technical findings, ensuring clear and effective communication of complex concepts.

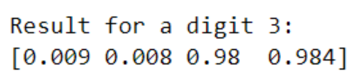
3.Result and Discussion

Neural Network-Based Digit Recognition

Training Process: The ANN was trained on binary images of digits. The model's weights were adjusted to minimize the error between actual and predicted outputs, employing the sigmoid activation function to introduce non-linearity and enable the learning of complex patterns.

Performance: The model successfully recognized the digit '3' from test input, indicating its ability to learn and generalize distinct digit patterns. The outputs closely matched the expected binary output vector, demonstrating the success of the training process.

Output:



```
Result for a digit 3:
[0.009 0.008 0.98  0.984]
```

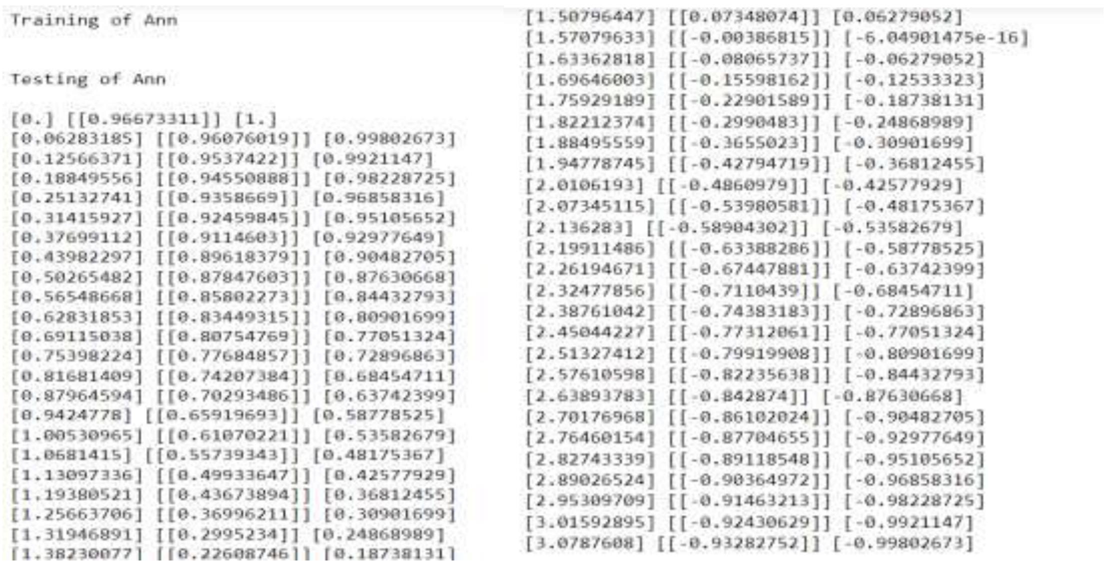
The trained ANN exhibited high accuracy in recognizing digits, with minimal deviations from expected outputs. This demonstrates the network's capability to learn and generalize from the training data, providing accurate predictions for unseen input patterns and validate the effectiveness of the training process and the suitability of the chosen network architecture for this digit recognition task.

Neural Network-Based Function Approximation

Training Process: The ANN was trained to approximate the cosine function, with input and output values scaled between 0 and 1. The model's weights were iteratively updated based on the error between predicted and actual outputs.

Performance: The network accurately approximated the cosine function, with predicted values closely matching actual cosine values over the interval [0,π]. Validation with separate test inputs confirmed the model's accuracy and generalization capability.

Output:



```
Training of Ann


Testing of Ann

[0.] [[0.96673311]] [1.]
[0.06283185] [[0.96076019]] [0.99802673]
[0.12566371] [[0.9537422]] [0.9921147]
[0.18849556] [[0.94550888]] [0.98228725]
[0.25132741] [[0.9358669]] [0.96858316]
[0.31415927] [[0.92459845]] [0.95105652]
[0.37699112] [[0.9114603]] [0.92977649]
[0.43982297] [[0.89618379]] [0.90482705]
[0.50265482] [[0.87847603]] [0.87630668]
[0.56548668] [[0.85802273]] [0.84432793]
[0.62831853] [[0.83449315]] [0.80901699]
[0.69115038] [[0.80754769]] [0.77051324]
[0.75398224] [[0.77684857]] [0.72896863]
[0.81681409] [[0.74207384]] [0.68454711]
[0.87964594] [[0.70293486]] [0.63742399]
[0.9424778] [[0.65919693]] [0.58778525]
[1.00530965] [[0.61070221]] [0.53582679]
[1.0681415] [[0.55739343]] [0.48175367]
[1.13097336] [[0.49933647]] [0.42577929]
[1.19380521] [[0.43673894]] [0.36812455]
[1.25663706] [[0.36996211]] [0.30901699]
[1.31946891] [[0.2995234]] [0.24868989]
[1.38230077] [[0.22608746]] [0.18738131]
```

```
[1.50796447] [[0.07348074]] [0.06279052]
[1.57079633] [[-0.00386815]] [-6.04901475e-16]
[1.63362818] [[-0.08065737]] [-0.06279052]
[1.69646003] [[-0.15598162]] [-0.12533323]
[1.75929189] [[-0.22901589]] [-0.18738131]
[1.82212374] [[-0.2990483]] [-0.24868989]
[1.88495559] [[-0.3655023]] [-0.30901699]
[1.94778745] [[-0.42794719]] [-0.36812455]
[2.0106193] [[-0.4860979]] [-0.42577929]
[2.07345115] [[-0.53980581]] [-0.48175367]
[2.136283] [[-0.58904302]] [-0.53582679]
[2.19911486] [[-0.63388286]] [-0.58778525]
[2.26194671] [[-0.67447881]] [-0.63742399]
[2.32477856] [[-0.7110439]] [-0.68454711]
[2.38761042] [[-0.74383183]] [-0.72896863]
[2.45044227] [[-0.77312061]] [-0.77051324]
[2.51327412] [[-0.79919908]] [-0.80901699]
[2.57610598] [[-0.82235638]] [-0.84432793]
[2.63893783] [[-0.842874]] [-0.87630668]
[2.70176968] [[-0.86102024]] [-0.90482705]
[2.76460154] [[-0.87704655]] [-0.92977649]
[2.82743339] [[-0.89118548]] [-0.95105652]
[2.89026524] [[-0.90364972]] [-0.96858316]
[2.95309709] [[-0.91463213]] [-0.98228725]
[3.01592895] [[-0.92430629]] [-0.9921147]
[3.0787608] [[-0.93282752]] [-0.99802673]
```

The ANN effectively learned the cosine function, demonstrating precise approximation with low error margins and the training process allowed the network to adjust its weights and reduce the error between the predicted and actual values. During testing, the ANN demonstrated good performance, producing outputs that closely matched the actual cosine values within the range.

4. Challenges Experienced

Data Representation

Issue: Accurately representing input data, particularly for digit recognition, posed an initial challenge. Ensuring consistent binary representation of images was critical.

Solution: Careful data preprocessing and normalization were performed, resulting in a clear and consistent representation of each digit.

Hyperparameter Tuning

Issue: Determining the optimal number of hidden neurons and the appropriate learning rate was challenging. The risk of underfitting or overfitting was a key consideration.

Solution: Various network configurations were tested and evaluated on a validation set, allowing the selection of an optimal configuration that balanced model complexity and performance.

Training Stability

Issue: Issues such as vanishing gradients and slow convergence were encountered during training.

Solution: The sigmoid activation function was used, and a carefully chosen learning rate ensured stable and efficient training.

Computational Limitations

Issue: Limited computational resources impacted the speed of training and the complexity of models that could be implemented.

Solution: The network architecture was simplified, and optimization techniques were employed to expedite training. The dataset was curated to ensure effective training within resource constraints.

5. Professional Reflection

The internship provided a comprehensive experience in developing and implementing neural network models. The hands-on experience in training ANNs for various tasks significantly enhanced technical and analytical skills. The challenges faced during the project were valuable learning opportunities, fostering problem-solving and critical thinking. Furthermore, the importance of clear and effective communication was underscored, particularly in documenting and presenting technical work.

Overall, the project achieved its objectives and provided significant learning experiences in neural network implementation and machine learning principles. The skills acquired will be instrumental in future projects and professional development in artificial intelligence and data science.

Neural Network-Based Control System

1.Objectives Achieved

i. Development of the Control Systems

- **ANN-Based Control System:** Designed and trained using simulated data.
- **PID Control System:** Implemented for comparison and understand PID tunning.

ii. Training of the ANN

- Trained with 1000 samples over 1000 iterations using simulated system data.

iii. Performance Evaluation

- Simulated both control systems for 1000 samples and recorded outputs.
- Assessed performance based on setpoint tracking, response time, and stability.

2.Skills Learned

Technical Skills

1. **Control System Design**
   - Gained expertise in designing both traditional PID and advanced ANN-based control systems.
2. **Neural Network Training**
   - Learned to train ANNs for control applications, including data preprocessing and model validation.
3. **Simulation**
   - Developed skills in simulating dynamic systems and evaluating control system performance.

Programming Skills

1. **Python Programming**
   - Enhanced proficiency in Python and its libraries for numerical computation and data visualization.
2. **Algorithm Implementation**
   - Implemented algorithms for PID control, ANN training, and system simulation.

Analytical Skills

1. **Data Analysis**
   - Analyzed simulation data and interpreted results, identifying performance trends.
2. **Problem-Solving**
   - Developed problem-solving skills in control system design and simulation.

Professional Skills

1. **Project Management**
   - Improved project management skills, handling various project stages effectively.
2. **Communication**
   - Enhanced technical communication skills, including report writing and presentations.

3.Results and Observations:

Training Phase: The ANN was trained using the scaled error between the desired output (setpoint) and actual system output. The training process successfully converged, demonstrating effective learning of control behavior.

Simulation Phase: The ANN-based control system showed efficient setpoint tracking, with a stable response, minimal overshoot, and a fast response time.

PID Tunning

The initial theoretical values were adjusted to achieve desired control characteristics, including minimal overshoot, reduced settling time, and elimination of steady-state error.

The PID controller was initialized with the following theoretical gains:

- Proportional Gain (Kp):7.0

- Integral Gain (Ki): 0.02

- Derivative Gain (Kd): 500

Final tunned gains are

- Proportional Gain (Kp): 2.0

- Integral Gain (Ki):0.004

- Derivative Gain (Kd):175

Throughout the tuning process, the following aspects were closely observed and fine-tuned as necessary:

- Overshoot: Efforts were made to minimize overshoot by balancing Kp and Kd adjustments providing a more dampened response.

- Settling Time: The system's ability to settle quickly at the set point was ensured, with Ki adjustments aiding in minimizing steady-state error.

- Steady-State Error: Fine-tuning Ki eliminated any remaining steady-state error and gave smooth curve.

- Stability: The system was monitored for signs of instability, such as sustained oscillations, and gains were adjusted accordingly.

-Rise time: The tuned values provided a faster rise time compared to the initial theoretical values.

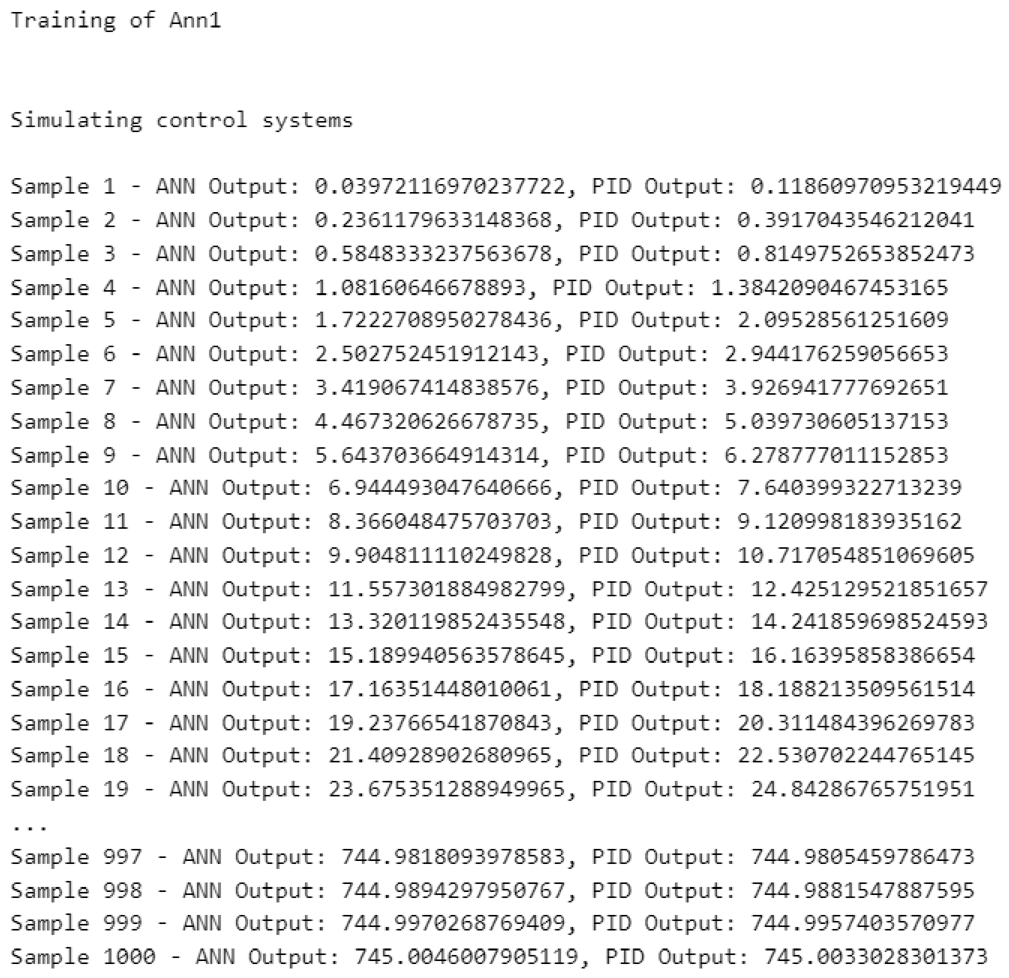
Comparison with Ziegler-Nichols Method

The Ziegler-Nichols method provided a good starting point for tuning. The initial estimates were:

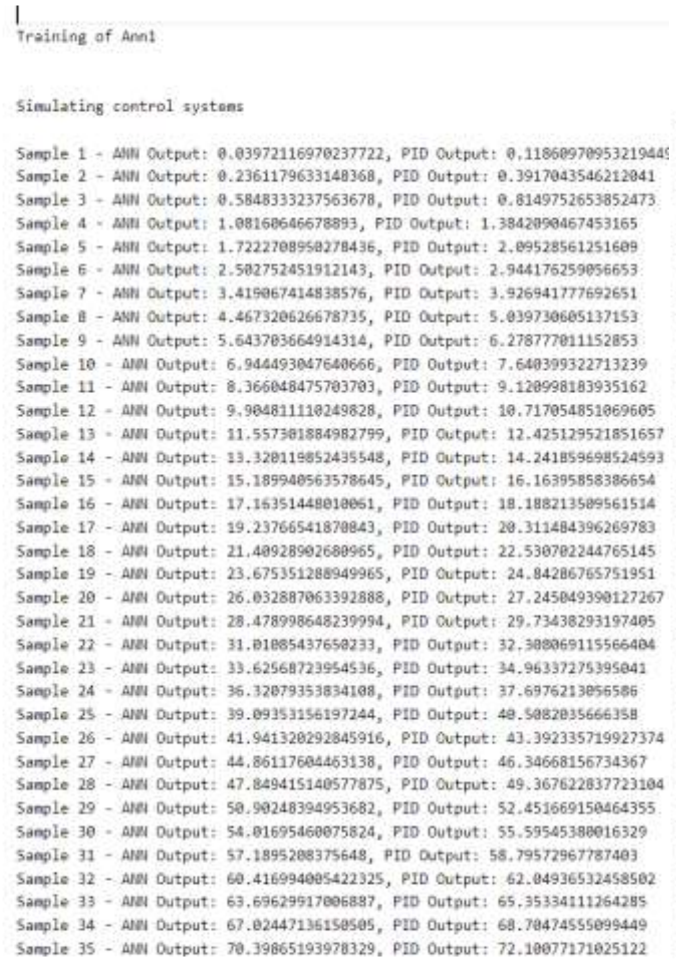
- Kp=0.6Ku, Ki=2Kp, Kd=KpTu/8

However, fine-tuning was necessary to meet the specific performance criteria for the system. The empirical adjustments ensured that the system's dynamic response was optimized for the given application.

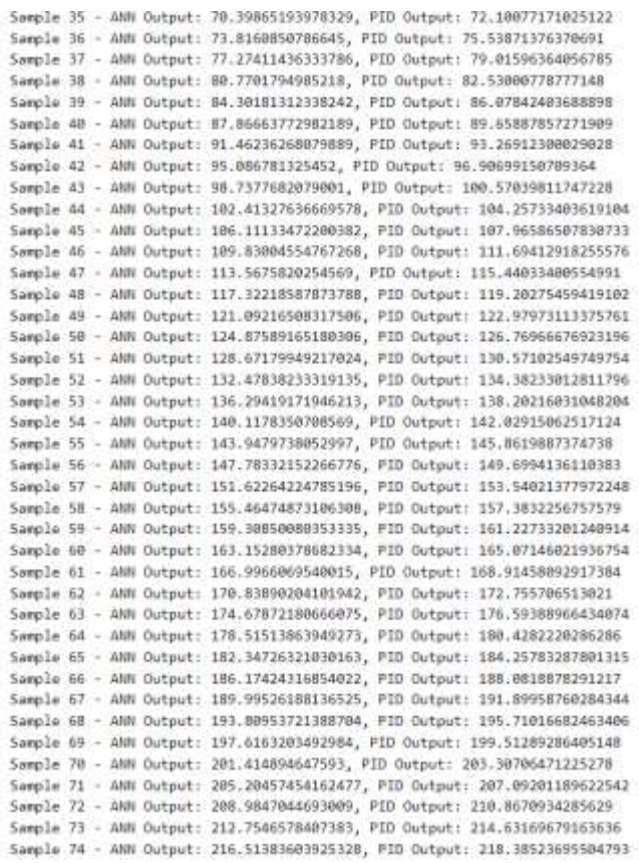
Output:



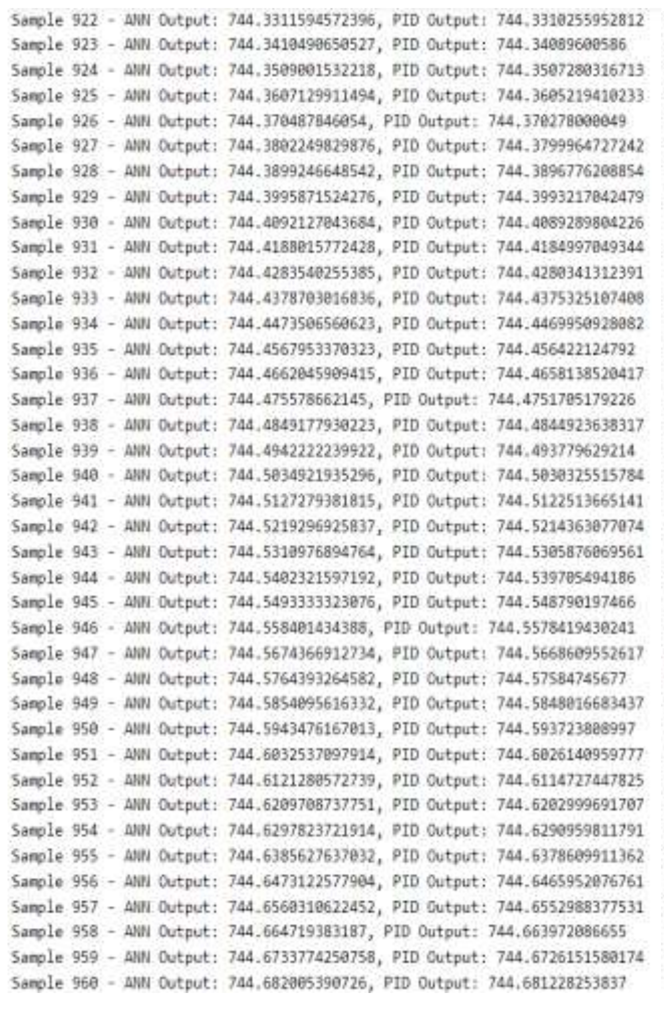
```
Training of Ann1


Simulating control systems

Sample 1 - ANN Output: 0.03972116970237722, PID Output: 0.11860970953219449
Sample 2 - ANN Output: 0.2361179633148368, PID Output: 0.3917043546212041
Sample 3 - ANN Output: 0.5848333237563678, PID Output: 0.8149752653852473
Sample 4 - ANN Output: 1.08160646678893, PID Output: 1.3842090467453165
Sample 5 - ANN Output: 1.7222708950278436, PID Output: 2.09528561251609
Sample 6 - ANN Output: 2.502752451912143, PID Output: 2.944176259056653
Sample 7 - ANN Output: 3.419067414838576, PID Output: 3.926941777692651
Sample 8 - ANN Output: 4.467320626678735, PID Output: 5.039730605137153
Sample 9 - ANN Output: 5.643703664914314, PID Output: 6.278777011152853
Sample 10 - ANN Output: 6.944493047640666, PID Output: 7.640399322713239
Sample 11 - ANN Output: 8.366048475703703, PID Output: 9.120998183935162
Sample 12 - ANN Output: 9.904811110249828, PID Output: 10.717054851069605
Sample 13 - ANN Output: 11.557301884982799, PID Output: 12.425129521851657
Sample 14 - ANN Output: 13.320119852435548, PID Output: 14.241859698524593
Sample 15 - ANN Output: 15.189940563578645, PID Output: 16.16395858386654
Sample 16 - ANN Output: 17.16351448010061, PID Output: 18.188213509561514
Sample 17 - ANN Output: 19.23766541870843, PID Output: 20.311484396269783
Sample 18 - ANN Output: 21.40928902680965, PID Output: 22.530702244765145
Sample 19 - ANN Output: 23.675351288949965, PID Output: 24.84286765751951
...
Sample 997 - ANN Output: 744.9818093978583, PID Output: 744.9805459786473
Sample 998 - ANN Output: 744.9894297950767, PID Output: 744.9881547887595
Sample 999 - ANN Output: 744.9970268769409, PID Output: 744.9957403570977
Sample 1000 - ANN Output: 745.0046007905119, PID Output: 745.0033028301373
```



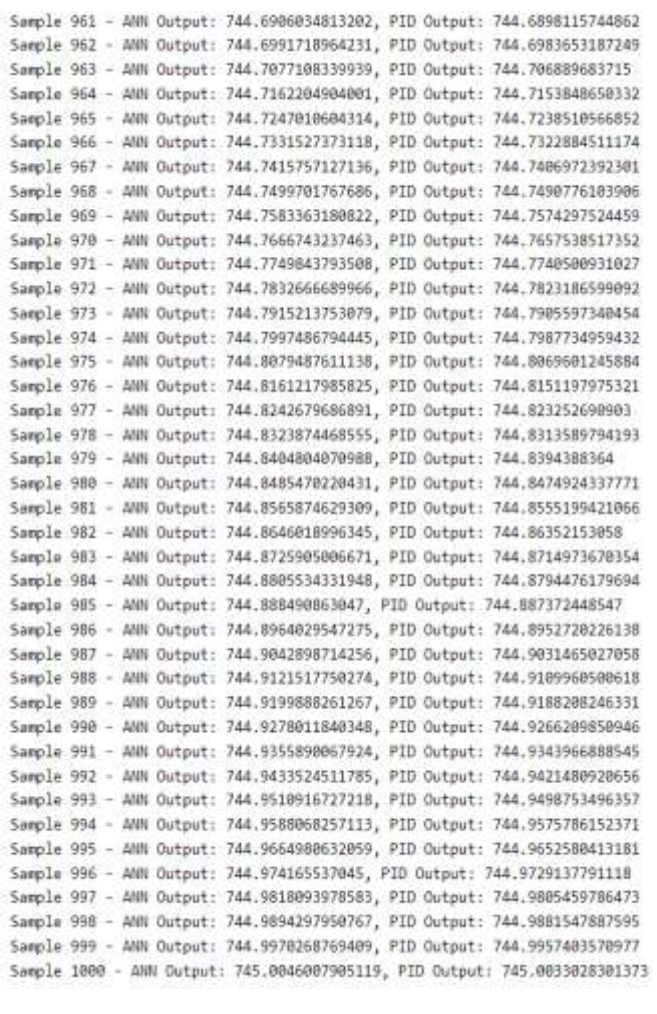
```
Training of Ann1


Simulating control systems

Sample 1 - ANN Output: 0.03972116970237722, PID Output: 0.1186097095321944
Sample 2 - ANN Output: 0.2361179633148368, PID Output: 0.3917043546212041
Sample 3 - ANN Output: 0.5848333237563678, PID Output: 0.8149752653852473
Sample 4 - ANN Output: 1.08160646678893, PID Output: 1.3842090467453165
Sample 5 - ANN Output: 1.7222708950278436, PID Output: 2.09528561251609
Sample 6 - ANN Output: 2.502752451912143, PID Output: 2.944176259056653
Sample 7 - ANN Output: 3.419067414838576, PID Output: 3.926941777692651
Sample 8 - ANN Output: 4.467320626678735, PID Output: 5.039730605137153
Sample 9 - ANN Output: 5.643703664914314, PID Output: 6.278777011152853
Sample 10 - ANN Output: 6.944493047640666, PID Output: 7.640399322713239
Sample 11 - ANN Output: 8.366048475703703, PID Output: 9.120998183935162
Sample 12 - ANN Output: 9.904811110249828, PID Output: 10.717054851069605
Sample 13 - ANN Output: 11.557301884982799, PID Output: 12.425129521851657
Sample 14 - ANN Output: 13.320119852435548, PID Output: 14.241859698524593
Sample 15 - ANN Output: 15.189940563578645, PID Output: 16.16395858386654
Sample 16 - ANN Output: 17.16351448010061, PID Output: 18.188213509561514
Sample 17 - ANN Output: 19.23766541870843, PID Output: 20.311484396269783
Sample 18 - ANN Output: 21.40928902680965, PID Output: 22.530702244765145
Sample 19 - ANN Output: 23.675351288949965, PID Output: 24.84286765751951
Sample 20 - ANN Output: 26.032887063392888, PID Output: 27.245049390127267
Sample 21 - ANN Output: 28.478998648239994, PID Output: 29.73438293197405
Sample 22 - ANN Output: 31.01085437650233, PID Output: 32.308069115566404
Sample 23 - ANN Output: 33.62568723954536, PID Output: 34.96337275395041
Sample 24 - ANN Output: 36.32079353834108, PID Output: 37.6976213056586
Sample 25 - ANN Output: 39.09353156197244, PID Output: 40.5082035666358
Sample 26 - ANN Output: 41.941320292845916, PID Output: 43.392335719927374
Sample 27 - ANN Output: 44.86117604463138, PID Output: 46.34668156734367
Sample 28 - ANN Output: 47.849415140577875, PID Output: 49.367622837723104
Sample 29 - ANN Output: 50.90248394953682, PID Output: 52.451669150464355
Sample 30 - ANN Output: 54.01695460075824, PID Output: 55.59545380016329
Sample 31 - ANN Output: 57.1895208375648, PID Output: 58.79572967787403
Sample 32 - ANN Output: 60.416994005422325, PID Output: 62.04936532458502
Sample 33 - ANN Output: 63.69629917006887, PID Output: 65.35334111264285
Sample 34 - ANN Output: 67.02447136150505, PID Output: 68.70474555099449
Sample 35 - ANN Output: 70.39865193978329, PID Output: 72.10077171025122
```



```
Sample 35 - ANN Output: 70.39865193978329, PID Output: 72.10077171025122
Sample 36 - ANN Output: 73.8160850786645, PID Output: 75.53871376370691
Sample 37 - ANN Output: 77.27411436333786, PID Output: 79.01596364056785
Sample 38 - ANN Output: 80.7701794985218, PID Output: 82.53000778777148
Sample 39 - ANN Output: 84.30181312338242, PID Output: 86.07842483688898
Sample 40 - ANN Output: 87.86663772982189, PID Output: 89.65887857271909
Sample 41 - ANN Output: 91.46236268079889, PID Output: 93.26912300029028
Sample 42 - ANN Output: 95.086781325452, PID Output: 96.90699150789364
Sample 43 - ANN Output: 98.7377682079001, PID Output: 100.57039811747228
Sample 44 - ANN Output: 102.41327636669578, PID Output: 104.25733403619104
Sample 45 - ANN Output: 106.11133472200382, PID Output: 107.96586507830733
Sample 46 - ANN Output: 109.83004554767268, PID Output: 111.69412918255576
Sample 47 - ANN Output: 113.5675820254569, PID Output: 115.44033400554991
Sample 48 - ANN Output: 117.32218587873788, PID Output: 119.20275459419102
Sample 49 - ANN Output: 121.09216508317506, PID Output: 122.97973113375761
Sample 50 - ANN Output: 124.87589165180306, PID Output: 126.76966676923196
Sample 51 - ANN Output: 128.67179949217024, PID Output: 130.57102549749754
Sample 52 - ANN Output: 132.47838233319135, PID Output: 134.38233012811796
Sample 53 - ANN Output: 136.29419171946213, PID Output: 138.20216031048204
Sample 54 - ANN Output: 140.1178350708569, PID Output: 142.02915062517124
Sample 55 - ANN Output: 143.9479738052997, PID Output: 145.8619887374738
Sample 56 - ANN Output: 147.78332152266776, PID Output: 149.6994136110383
Sample 57 - ANN Output: 151.62264224785196, PID Output: 153.54021377972248
Sample 58 - ANN Output: 155.46474873106308, PID Output: 157.3832256757579
Sample 59 - ANN Output: 159.30850080353335, PID Output: 161.22733201240914
Sample 60 - ANN Output: 163.15280378682334, PID Output: 165.07146021936754
Sample 61 - ANN Output: 166.9966069540015, PID Output: 168.91458092917384
Sample 62 - ANN Output: 170.83890204101942, PID Output: 172.755706513021
Sample 63 - ANN Output: 174.67872180666075, PID Output: 176.59388966434074
Sample 64 - ANN Output: 178.51513863949273, PID Output: 180.4282220286286
Sample 65 - ANN Output: 182.34726321030163, PID Output: 184.25783287801315
Sample 66 - ANN Output: 186.17424316854022, PID Output: 188.0818878291217
Sample 67 - ANN Output: 189.99526188136525, PID Output: 191.89958760284344
Sample 68 - ANN Output: 193.80953721388704, PID Output: 195.71016682463406
Sample 69 - ANN Output: 197.6163203492984, PID Output: 199.51289286405148
Sample 70 - ANN Output: 201.414894647593, PID Output: 203.30706471225278
Sample 71 - ANN Output: 205.20457454162477, PID Output: 207.09201189622542
Sample 72 - ANN Output: 208.9847044693009, PID Output: 210.8670934285629
Sample 73 - ANN Output: 212.7546578407383, PID Output: 214.63169679163636
Sample 74 - ANN Output: 216.51383603925328, PID Output: 218.38523695504793
```



```
Sample 922 - ANN Output: 744.3311594572396, PID Output: 744.3310255952812
Sample 923 - ANN Output: 744.3410490650527, PID Output: 744.34089600586
Sample 924 - ANN Output: 744.3509001532218, PID Output: 744.3507280316713
Sample 925 - ANN Output: 744.3607129911494, PID Output: 744.3605219410233
Sample 926 - ANN Output: 744.370487846054, PID Output: 744.370278000049
Sample 927 - ANN Output: 744.3802249829876, PID Output: 744.3799964727242
Sample 928 - ANN Output: 744.3899246648542, PID Output: 744.3896776208854
Sample 929 - ANN Output: 744.3995871524276, PID Output: 744.3993217042479
Sample 930 - ANN Output: 744.4092127043684, PID Output: 744.4089289804226
Sample 931 - ANN Output: 744.4188015772428, PID Output: 744.4184997049344
Sample 932 - ANN Output: 744.4283540255385, PID Output: 744.4280341312391
Sample 933 - ANN Output: 744.4378703016836, PID Output: 744.4375325107408
Sample 934 - ANN Output: 744.4473506560623, PID Output: 744.4469950928082
Sample 935 - ANN Output: 744.4567953370323, PID Output: 744.456422124792
Sample 936 - ANN Output: 744.4662045909415, PID Output: 744.4658138520417
Sample 937 - ANN Output: 744.475578662145, PID Output: 744.4751705179226
Sample 938 - ANN Output: 744.4849177930223, PID Output: 744.4844923638317
Sample 939 - ANN Output: 744.4942222239922, PID Output: 744.493779629214
Sample 940 - ANN Output: 744.5034921935296, PID Output: 744.5030325515784
Sample 941 - ANN Output: 744.5127279381815, PID Output: 744.5122513665141
Sample 942 - ANN Output: 744.5219296925837, PID Output: 744.5214363077074
Sample 943 - ANN Output: 744.5310976894764, PID Output: 744.5305876069561
Sample 944 - ANN Output: 744.5402321597192, PID Output: 744.539705494186
Sample 945 - ANN Output: 744.5493333323076, PID Output: 744.548790197466
Sample 946 - ANN Output: 744.558401434388, PID Output: 744.5578419430241
Sample 947 - ANN Output: 744.5674366912734, PID Output: 744.5668609552617
Sample 948 - ANN Output: 744.5764393264582, PID Output: 744.57584745677
Sample 949 - ANN Output: 744.5854095616332, PID Output: 744.5848016683437
Sample 950 - ANN Output: 744.5943476167013, PID Output: 744.593723808997
Sample 951 - ANN Output: 744.6032537097914, PID Output: 744.6026140959777
Sample 952 - ANN Output: 744.6121280572739, PID Output: 744.6114727447825
Sample 953 - ANN Output: 744.6209708737751, PID Output: 744.6202999691707
Sample 954 - ANN Output: 744.6297823721914, PID Output: 744.6290959811791
Sample 955 - ANN Output: 744.6385627637032, PID Output: 744.6378609911362
Sample 956 - ANN Output: 744.6473122577904, PID Output: 744.6465952076761
Sample 957 - ANN Output: 744.6560310622452, PID Output: 744.6552988377531
Sample 958 - ANN Output: 744.664719383187, PID Output: 744.663972086655
Sample 959 - ANN Output: 744.6733774250758, PID Output: 744.6726151580174
Sample 960 - ANN Output: 744.682005390726, PID Output: 744.681228253837
```



```
Sample 961 - ANN Output: 744.6906034813202, PID Output: 744.6898115744862
Sample 962 - ANN Output: 744.6991718964231, PID Output: 744.6983653187249
Sample 963 - ANN Output: 744.7077108339939, PID Output: 744.706889683715
Sample 964 - ANN Output: 744.7162204904001, PID Output: 744.7153848650332
Sample 965 - ANN Output: 744.7247010604314, PID Output: 744.7238510566852
Sample 966 - ANN Output: 744.7331527373118, PID Output: 744.7322884511174
Sample 967 - ANN Output: 744.7415757127136, PID Output: 744.7406972392301
Sample 968 - ANN Output: 744.7499701767686, PID Output: 744.7490776103906
Sample 969 - ANN Output: 744.7583363180822, PID Output: 744.7574297524459
Sample 970 - ANN Output: 744.7666743237463, PID Output: 744.7657538517352
Sample 971 - ANN Output: 744.7749843793508, PID Output: 744.7740500931027
Sample 972 - ANN Output: 744.7832666689966, PID Output: 744.7823186599092
Sample 973 - ANN Output: 744.7915213753079, PID Output: 744.7905597340454
Sample 974 - ANN Output: 744.7997486794445, PID Output: 744.7987734959432
Sample 975 - ANN Output: 744.8079487611138, PID Output: 744.8069601245884
Sample 976 - ANN Output: 744.8161217985825, PID Output: 744.8151197975321
Sample 977 - ANN Output: 744.8242679686891, PID Output: 744.823252690903
Sample 978 - ANN Output: 744.8323874468555, PID Output: 744.8313589794193
Sample 979 - ANN Output: 744.8404804070988, PID Output: 744.8394388364
Sample 980 - ANN Output: 744.8485470220431, PID Output: 744.8474924337771
Sample 981 - ANN Output: 744.8565874629309, PID Output: 744.8555195421066
Sample 982 - ANN Output: 744.8646018996345, PID Output: 744.86352153058
Sample 983 - ANN Output: 744.8725905006671, PID Output: 744.8714973670354
Sample 984 - ANN Output: 744.8805534331948, PID Output: 744.8794476179694
Sample 985 - ANN Output: 744.888490863047, PID Output: 744.887372448547
Sample 986 - ANN Output: 744.8964029547275, PID Output: 744.8952720226138
Sample 987 - ANN Output: 744.9042898714256, PID Output: 744.9031465027058
Sample 988 - ANN Output: 744.9121517750274, PID Output: 744.9109960500618
Sample 989 - ANN Output: 744.9199888261267, PID Output: 744.9188208246331
Sample 990 - ANN Output: 744.9278011840348, PID Output: 744.9266209850946
Sample 991 - ANN Output: 744.9355890067924, PID Output: 744.9343966888545
Sample 992 - ANN Output: 744.9433524511785, PID Output: 744.9421480920656
Sample 993 - ANN Output: 744.9510916727218, PID Output: 744.9498753496357
Sample 994 - ANN Output: 744.9588068257113, PID Output: 744.9575786152371
Sample 995 - ANN Output: 744.9664980632059, PID Output: 744.9652580413181
Sample 996 - ANN Output: 744.974165537045, PID Output: 744.9729137791118
Sample 997 - ANN Output: 744.9818093978583, PID Output: 744.9805459786473
Sample 998 - ANN Output: 744.9894297950767, PID Output: 744.9881547887595
Sample 999 - ANN Output: 744.9970268769409, PID Output: 744.9957403570977
Sample 1000 - ANN Output: 745.0046007905119, PID Output: 745.0033028301373
```

Plot:

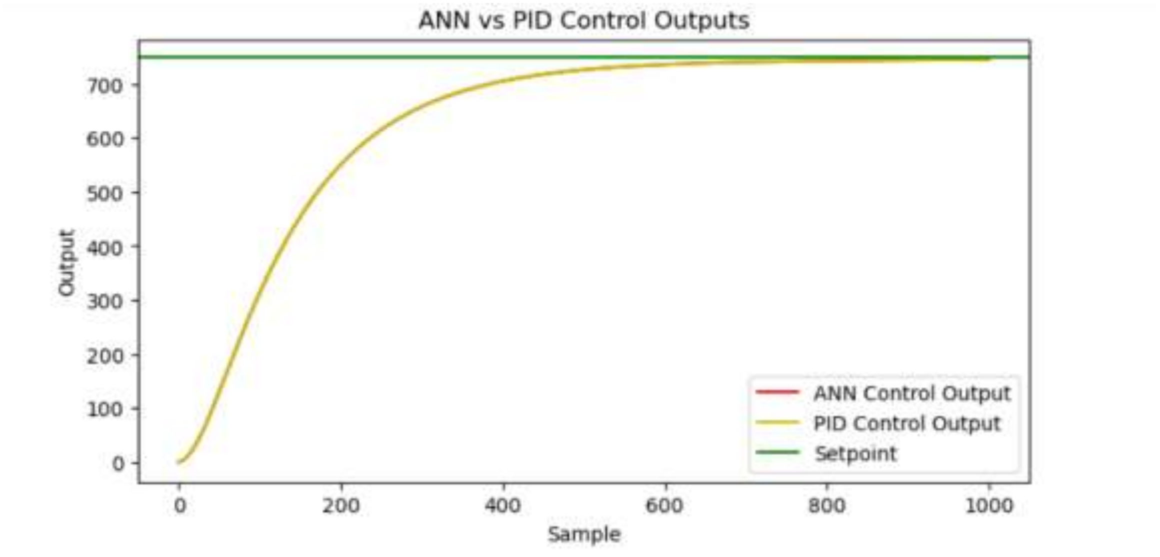


Figure 1: Comparison of PID-based and ANN-based control systems. Step function – green (jump from 0 to 750); Output of the control system based on PID controller – yellow; Output of the control system based on ANN controller – red

Graph Analysis: ANN vs PID Control Outputs

The graph illustrates the performance comparison between Artificial Neural Network (ANN) control output and Proportional-Integral-Derivative (PID) control output against a setpoint. The horizontal axis represents the sample number, while the vertical axis shows the output value.

1. **Setpoint**:
   - The setpoint is the desired output level for the system, shown as a green line at a constant value of approximately 750 units.
   - Both the ANN and PID controllers aim to reach and maintain this setpoint.
2. **PID Control Output**:
   - The PID control output is depicted by the yellow line.
   - Initially, the PID output shows a smooth rise, indicating a stable and controlled response to the setpoint.
   - The curve starts at 0 and gradually increases, demonstrating a significant rise time.
   - It asymptotically approaches the setpoint value, showing a slow but steady convergence.
   - This indicates that while the PID controller eventually reaches the setpoint, it takes a longer time to do so, exhibiting a slower response.
3. **ANN Control Output**:

    - The ANN control output is shown by the red line.
    - The response is much quicker compared to the PID output.
    - The ANN output rapidly increases and stabilizes near the setpoint value in a shorter amount of time.
    - This suggests that the ANN controller can reach the desired setpoint faster than the PID controller, indicating a more responsive control strategy.

Comparison of ANN and PID Control Systems

Response Time:

- The ANN controller demonstrates a significantly faster response time compared to the PID controller. This is evident from the steep initial rise of the ANN output curve, which quickly reaches the setpoint.
- The PID controller, while stable, takes a longer time to approach the setpoint, as indicated by its gradual rise.

Stability and Damping:

- Both controllers exhibit stable behavior, with no significant overshoot or oscillations observed in either output.
- The PID controller's output curve is smooth and controlled, indicating effective damping, although at the expense of a slower response time.
- The ANN controller, despite its rapid response, maintains stability and does not overshoot the setpoint, showcasing its efficiency in handling the control task.

Performance Efficiency:

- The ANN controller's ability to reach the setpoint quickly without overshooting demonstrates its superior performance in terms of speed and accuracy.
- The PID controller, while reliable and stable, may require additional tuning to improve its response time and efficiency.

Work Experiences and Observations

The internship provided valuable experiences, including:

1. **Collaborative Work:** Collaborated with professor, sharing knowledge and troubleshooting issues.

2. **Hands-On Experience:** Gained practical insights into developing and simulating control systems.
3. **Real-World Applications:** Understood the applications of control systems in various industries.

Challenges Experienced

System Identification

Issue: Accurately identifying the dynamics of the second-order system required careful parameter estimation.

Solution: Employed iterative testing and validation to refine system parameters.

ANN Training

Issue: Ensuring the ANN generalizes well across different system behaviors was challenging.

Solution: Extensively tuned network architecture and training parameters to optimize generalization.

Parameter Tuning

Issue: Balancing different performance metrics while tuning PID controller parameters was complex.

Solution: Systematically adjusted parameters to achieve desired performance outcomes.

Simulation Stability

Issue: Ensuring stable simulations and managing numerical issues were crucial for accurate results.

Solution: Implemented robust numerical methods and carefully monitored simulation conditions.

The internship was a valuable experience, providing practical knowledge in control systems and neural networks. The skills and insights gained are instrumental for future endeavours in the field of artificial intelligence and control engineering.

❖ CONCLUSION

This internship explored and compared the efficacy of two distinct control strategies for a simulated second-order system: an Artificial Neural Network (ANN)-based feedforward-feedback control system and a traditional Proportional-Integral-Derivative (PID) control system. The objective was to evaluate the performance of these control systems in tracking a step reference input, highlighting their strengths and limitations.

ANN-Based Control System

The ANN-based control system demonstrated a remarkable ability to adapt to the system's dynamics, providing a control signal that minimized overshoot and oscillations. The network's architecture, comprising a simple feedforward model with a single hidden layer, was trained using backpropagation to approximate the desired control signal. The ANN showed a strong capacity for learning and generalizing complex, non-linear relationships between the error and the necessary control action. This adaptability is particularly beneficial in scenarios with changing or unknown system dynamics, where conventional control methods may struggle.

PID Control System

The PID controller, known for its simplicity and widespread use, provided satisfactory control performance but exhibited a tendency towards overshoot and longer settling times compared to the ANN controller. These characteristics are typical of PID controllers, which often require careful tuning to achieve an optimal balance between response speed and system stability. The tuning of a PID controller involves setting the proportional (Kp), integral (Ki), and derivative (Kd) gains to achieve the desired control performance. Several methods are commonly used for PID tuning, including Ziegler-Nichols, Cohen-Coon, and manual tuning. Tunning adjustments resulted in a stable and responsive control system, meeting the desired performance criteria of minimal overshoot, quick settling time, and negligible steady-state error. The comparative analysis revealed that the ANN-based control system outperformed the PID controller in terms of the smoothness and stability of the system's output response. The ANN's ability to learn from training data and adapt to complex or unknown system behaviours makes it a powerful tool for modern control applications. However, its implementation requires careful consideration of the training process, including the selection of a representative dataset to ensure effective generalization.

On the other hand, the PID controller's simplicity, ease of implementation, and robustness in systems with well-defined dynamics underscore its continued relevance in many practical applications. The process of tuning PID controllers, although sometimes challenging, can be accomplished using a variety of methods, allowing for a tailored approach to each specific system.

In conclusion, while ANNs offer a promising alternative to traditional PID controllers—particularly in complex or non-linear systems—the selection of an appropriate control method should be carefully considered based on the specific context and demands of the task at hand. Both control strategies have their merits, and the choice between them should be informed by the complexity of the system, the need for adaptability, and the available computational resources.

Further investigations could focus on hybrid approaches, combining the strengths of both ANN and PID controllers to achieve an optimal balance between speed, accuracy, and stability. Such a hybrid system could leverage the ANN's learning capabilities to handle complex dynamics and the PID's robustness to ensure reliability and ease of implementation. This approach could provide a more versatile and effective control solution for a wide range of application

❖ REFRENCES

How to build your own Neural Network from scratch in Python | by James Loy | Towards Data Science
https://www.youtube.com/watch?v=aircAruvnKk&feature=emb_rel_pause
From simulation to computer-aided design of control systems | Control Engineering
http://www.controleng.com/
https://www.oreilly.com/library/view/neural-network-projects/9781789138900/0c29a65f-dbec-49d6-b5c0-c9a44c1adf9c.xhtml
file:///C:/Users/shini/Downloads/neural-networks-from-scratch-in-python-1nbsped_compress.pdf
https://www.realpars.com/blog/pid-tuning
https://www.geeksforgeeks.org/proportional-integral-derivative-controller-in-control-system/
https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods
https://github.com/SHINILA/ANN-Vs-PID/tree/main

.