# Advanced Machine Learning-Driven Predictive Frameworks for Analyzing and Enhancing Stability Margins in PID-Controlled Systems

A PROJECT REPORT

Submitted by

| Roll Number | Registration Number | Student Code | Student Name |
|---|---|---|---|
| 23010201160 | 23012005276 of 2023-2024 | BWU/MCA/23/171 | Arpan Mondal |
| 23010201133 | 23012005249 of 2023-2024 | BWU/MCA/23/144 | Jyotipriya Roy |
| 23010201124 | 23012005240 of 2023-2024 | BWU/MCA/23/133 | Kingshuk Saha |
| 23010201133 | 23012005251 of 2023-2024 | BWU/MCA/23/146 | Nibedita Das |
| 23010201121 | 23012005237 of 2023-2024 | BWU/MCA/23/130 | Shraya Ghosh |

*in partial fulfillment for the award of the degree*

*of*

**Master of Computer Applications**

***in***

**Department of Computational Sciences**

**BRAINWARE UNIVERSITY**
**398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125**

June 2025

# Advanced Machine Learning-Driven Predictive Frameworks for Analyzing and Enhancing Stability Margins in PID-Controlled Systems

***Submitted by***

| Roll Number | Registration Number | Student Code | Student Name |
|---|---|---|---|
| 23010201160 | 23012005276 of 2023-2024 | BWU/MCA/23/171 | Arpan Mondal |
| 23010201133 | 23012005249 of 2023-2024 | BWU/MCA/23/144 | Jyotipriya Roy |
| 23010201124 | 23012005240 of 2023-2024 | BWU/MCA/23/133 | Kingshuk Saha |
| 23010201133 | 23012005251 of 2023-2024 | BWU/MCA/23/146 | Nibedita Das |
| 23010201121 | 23012005237 of 2023-2024 | BWU/MCA/23/130 | Shraya Ghosh |

***in partial fulfillment for the award of the degree***

***of***

**Master of Computer Applications**

***in***

**Department of Computational Sciences**

**BRAINWARE UNIVERSITY**

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*

**BRAINWARE UNIVERSITY**

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*

## Department of Computational Science

## BONAFIDE CERTIFICATE

Certified that this project report **"Advanced Machine Learning-Driven Predictive Frameworks for Analyzing and Enhancing Stability Margins in PID-Controlled Systems"** is the Bonafide work of **"Arpan Mondal, Jyotipriya Roy, Kingshuk Saha, Nibedita Das, Shraya Ghosh"** who carried out the project work under my supervision.

| | |
|---|---|
| ------------------------------------------------ | ------------------------------------------------ |
| **SIGNATURE** | **SIGNATURE** |
| Dr. Jayanta Aich | Mr. Shantanu Bhadra |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Department of Computational Science | Assistant Professor |
| Brainware University. Rammohon Bhawan | Department of Computational Science |
| (UB-VI), 398, Ramkrishnapur Road, Barasat, | |
| North 24pgs, Kolkata - 700125, India. | |

# ACKNOWLEDGEMENT

**Project Title:** Advanced Machine Learning-Driven Predictive Frameworks for Analyzing and Enhancing Stability Margins in PID-Controlled Systems.

**Project Group ID:** MCA23C004

We, the undersigned project members, would like to express our sincere gratitude to Mr. Shantanu Bhadra, Assistant Professor, Department of Computational Sciences, Brainware University, for their invaluable guidance, support, and encouragement throughout the course of this project. Their expert advice and constructive feedback were crucial in the successful completion of this work.

We also extend our heartfelt thanks to Dr. Jayanta Aich, Head of the Department of Computational Sciences, for providing the necessary resources and support throughout this project.

We acknowledge with gratitude the support and cooperation of all faculty members and staff of the Department of Computational Sciences. Their insights and suggestions helped us stay on course and improve the quality of our work.

We especially thank our families, friends, and peers for their encouragement and support throughout this journey and for motivating us to achieve our goals.

Finally, we express our deepest gratitude to Brainware University for providing us with the opportunity to undertake this project as part of our academic curriculum.

**Project Members:**

| Student Name | Student Code | Signature |
|---|---|---|
| **Arpan Mondal** | **BWU/MCA/23/171** | |
| **Jyotipriya Roy** | **BWU/MCA/23/144** | |
| **Kingshuk Saha** | **BWU/MCA/23/133** | |
| **Nibedita Das** | **BWU/MCA/23/146** | |
| **Shraya Ghosh** | **BWU/MCA/23/130** | |

## Abstract

This paper presents an advanced predictive framework powered by machine learning (ML) techniques to analyze and enhance stability margins in systems controlled by Proportional-Integral-Derivative (PID) controllers. The primary objective of the proposed framework is to leverage the capabilities of ML algorithms to evaluate, predict, and improve the stability characteristics of control systems under various dynamic conditions by integrating data-driven approaches with classical control theory. The framework involves the use of supervised and unsupervised machine learning models trained on extensive datasets consisting of system responses, controller parameters, and stability outcomes. These models are employed to identify critical patterns, predict system stability, and suggest optimized PID tuning parameters that enhance stability margins while maintaining desired performance metrics. This novel approach not only augments traditional PID control strategies but also sets the stage for next-generation predictive control paradigms driven by intelligent data analytics.

# TABLE OF CONTENTS

| Chapter | Title | Page No. |
|---|---|---|
| | **ABSTRACT** | **v** |
| | **LIST OF TABLES** | **ix** |
| | **LIST OF FIGURES** | **x** |
| **1.** | **INTRODUCTION** | |
| | 1.1 Stability Challenges and Importance of PID Controller Tuning in Control Systems | **11** |
| | 1.2 Enhancing Control System Stability Analysis with Machine Learning | **11** |
| | 1.3 Data-Driven Stability Classification in PID Systems Using SVM and Ensemble Learning | **12** |
| **2.** | **LITERATURE REVIEW** | |
| | 2.1. Data-driven optimal tuning of PID controller parameters. | **13** |
| | 2.2. Stability analysis and tuning of PID controllers in VAV systems. | **14** |
| | 2.3. Interpretable PID parameter tuning for control engineering using general dynamic neural networks: An extensive comparison. | **15** |
| | 2.4. PID Control Parameter Tuning Using Linear Multivariate Model. | **16** |
| | 2.5. Adaptive PID controller for stable/unstable linear and nonlinear systems. | **17** |
| | 2.6. Stability analysis of PWM feedback control systems with PID regulators. | **18** |
| | 2.7. PID control system analysis, design, and technology. | **19** |

2.8. A robust PID controller based on linear quadratic gaussian approach for improving frequency stability of power systems considering renewables. **20**

2.9. PID control system analysis and design. **21**

2.10. A comparative overview of different approaches for calculating the set of all stabilizing PID controller parameters. **22**

2.11. Tuning of PID-type controllers for stable and unstable systems with time delay. **23**

**3. RESEARCH GAP**

3.1. Data-driven optimal tuning of PID controller parameters. **24**

3.2. Stability analysis and tuning of PID controllers in VAV systems. **24**

3.3. Interpretable PID parameter tuning for control engineering using general dynamic neural networks: An extensive comparison. **24**

3.4. PID Control Parameter Tuning Using Linear Multivariate Model. **25**

3.5. Adaptive PID controller for stable/unstable linear and nonlinear systems. **25**

3.6. Stability analysis of PWM feedback control systems with PID regulators. **25**

3.7. PID control system analysis, design, and technology. **26**

3.8. A robust PID controller based on linear quadratic gaussian approach for improving frequency stability of power systems considering renewables. **26**

3.9. PID control system analysis and design. **26**

3.10. A comparative overview of different approaches for calculating the set of all stabilizing PID controller parameters. **27**

3.11. Tuning of PID-type controllers for stable and unstable systems with time delay. **27**

**4. THEORY, METHODOLOGY, MATERIALS & METHODS**

4.1. Physical Dynamics of the Controller **28**

4.2. Computational Analysis **33**

4.3 Physical Computation **35**

**5. RESULTS, ANALYSIS AND DISCUSSIONS**

5.1 Program to implement the SVM Model **43**

5.1.1 Exploratory Data Analysis **44**

5.1.2 Data Preprocessing **46**

5.1.3 Training the Machine Learning Model **49**

5.1.4 Testing Accuracy of the ML Model **51**

5.2. Reliability and Stability Analysis of the System **54**

5.2.1 Results **54**

5.2.2 Model Interpretation **55**

5.3. Discussions **55**

5.3.1 Merits of the SVM Model **55**

5.3.2 Demerits of the SVM Model **55**

**6. CONCLUSION, FUTURE SCOPE, LIMITATIONS**

6.1. Discussions **56**

6.2. Broader Implications **57**

6.3. Limitations **57**

6.4. Future Improvements **57**

**7. REFERENCES** **58**

# List of Tables

1. Table 5.1: Continuous Dataset…..…………………………………….41
2. Table 5.2: Dataset after Feature Selection………………………46

# List of Figures

3. Figure 4.1: Control System in Simulink…..………………………………….28
4. Figure 4.2: Block Parameters: Step…………..……………………………….29
5. Figure 4.3: Block Parameters: Sum………..………………………………….30
6. Figure 4.4: Block Parameters: Integrator……………..……………………31
7. Figure 4.5: Block Parameters:PID Controller……………..………………32
8. Figure 4.6: Routh's Stability Criteria……………………………………….36
9. Figure 5.1: CSV File……………..……………………………………………………41
10. Figure 5.2: Continuous Data……………..…………………………………..42
11. Figure 5.3: Continuous Data……………..…………………………………..42
12. Figure 5.4: Linearly Non Separable Data…………………………………..44
13. Figure 5.5: Heatmap……….………………..…………………………………..47
14. Figure 5.6: Decision Boundary…………..…………………………………….52

# Chapters

# 1. Chapter 1: Introduction

## 1.1. Stability Challenges and Importance of PID Controller Tuning in Control Systems

Stability analysis in control systems is a critical area of research, especially in the context of Proportional-Integral-Derivative (PID) controllers, which are widely used in industrial, robotics, and process automation applications. PID controllers form the backbone of many closed-loop control systems due to their simplicity, robustness, and effectiveness in managing dynamic systems. However, achieving a stable system output depends heavily on the tuning of PID parameters and the characteristics of the input signals. Improper configurations can lead to unstable system behavior, resulting in performance degradation or even system failure.

## 1.2. Enhancing Control System Stability Analysis with Machine Learning

In recent years, the integration of machine learning (ML) into control system analysis has emerged as a transformative approach, enabling predictive modeling, pattern recognition, and optimization of system behaviors. Advanced ML algorithms offer powerful tools for identifying relationships between input parameters and system stability. By leveraging these techniques, it's possible to predict the stability margins of PID-controlled systems with greater accuracy and reliability.

This paper, titled *"Advanced Machine Learning-Driven Predictive Frameworks for Analyzing and Enhancing Stability Margins in PID-Controlled Systems,"* proposes a novel framework that combines the strengths of closed-loop control systems with state-of-the-art machine learning models. The closed-loop system is constructed using components such as a step input, sum block, integrator block, and PID controller to simulate real-world dynamic behaviors. The focus is on analyzing the effects of varying input configurations—both from the step signal and the PID controller parameters—on the stability of the system outputs. Stability is categorized into two states: stable outputs and unstable outputs, which are evaluated using a combination of control theory principles and machine learning techniques.

### 1.3. Data-Driven Stability Classification in PID Systems Using SVM and Ensemble Learning

By employing a classification model, Support Vector Machines (SVM), with ensemble learning the framework identifies the input conditions that lead to stable and unstable system responses. These predictive models not only classify stability outcomes but also offer insights into the critical parameters that influence system performance. This research aims to bridge the gap between traditional control theory and modern ML methodologies.

The findings of this study have significant implications for real-world applications, where stability is essential. From optimizing industrial processes to ensuring the reliability in autonomous systems, the proposed framework provides a robust foundation for developing adaptive control strategies. The integration of machine learning into control system analysis represents a paradigm shift, enabling smarter, data-driven approaches to stability enhancement in PID-controlled systems.

# 2. Chapter 2: Literature Review

## 2.1. Data-driven optimal tuning of PID controller parameters.

### Author's name
Ning Liu, Tianyou Chai, Yajun Zhang & Weinan Gao.

### Abstract
This study proposes an optimal tuning method of proportional integral derivative (PID) controller parameters for a class of single-input, single-output nonlinear systems affected by unknown disturbances. The system is first transformed into a linear system model influenced by nonlinear uncertainty. Subsequently, the PID controller parameters are tuned by minimizing the fluctuations in the tracking errors of the closed-loop system with robust stability guarantees, where the PID control law and closed-loop equation are considered constraints. To address the challenges involved in solving the optimization problem, some new variables are introduced to transform the uncertain system in closed-loop with the PID controller into an equation that makes the optimal PID parameters solvable, where the PID parameters are treated as model parameters, the new variables act as inputs, and the tracking error serves as the output. Integrating the recursive optimization and concurrent learning, along with the accurately calculated nonlinear uncertainty at the last sampling instant and the designed adjusting factors of the convergence rate that maintain robust sensitivities in an expected range, an optimal tuning algorithm for PID parameters is proposed without the requirement of persistent excitation, which is essentially different from the traditional adaptive control. The stability and convergence analyses of the proposed algorithm are also presented, and its superiority is validated through simulations.

### Merits
The proposed PID tuning method presents several notable merits. It offers a robust and optimal approach to controller design for nonlinear systems affected by unknown disturbances by transforming the nonlinear system into a tractable linear model with uncertainty. By minimizing tracking error fluctuations and incorporating constraints directly from the control law and closed-loop dynamics, the method ensures robust stability. The introduction of new variables to reformulate the optimization problem enhances solvability, while the integration of recursive optimization and concurrent learning enables adaptive behavior without requiring persistent excitation—a key limitation in many traditional adaptive control methods. Additionally, the use of precisely calculated nonlinear uncertainties and designed convergence rate factors improves sensitivity management and overall performance, as supported by the stability, convergence analysis, and simulation validation.

### Demerits
However, some demerits can also be inferred. The transformation of the nonlinear system and introduction of new variables might increase computational complexity and implementation difficulty, especially in real-time systems. The method's effectiveness heavily depends on the accurate estimation of nonlinear uncertainties, which may be challenging in highly dynamic or noisy environments. Moreover, although persistent excitation is not required, the abstract does not detail how well the method performs under highly varying or unstructured disturbances. Lastly, the practicality of the approach in real-world systems—beyond simulations—remains to be demonstrated, which may raise concerns about generalizability and scalability.

## 2.2. Stability analysis and tuning of PID controllers in VAV systems.

### Author's name

Masato Kasahara, Takanori Yamazaki, Yoshiaki, Kuzuu, Yukihiro Hashimoto.

### Abstract

VAV systems regulate airflow to maintain desired temperatures (thermal comfort) in spaces. They use the airflow rate (as control input) and the temperature (as the controlled variable). These systems are popular in buildings but can lead to stability issues with temperature and air pressure, causing unwanted oscillations (called "hunting"). Stability limits help us understand when the system becomes unstable. However, due to the complexity (nonlinear interactions like multiplying airflow rate and temperature), finding these limits is tough in real-life applications. The paper writes about -
Modeling: Develops a simplified model of VAV systems, focusing on how energy flows in and out of the space.
Stability Analysis: Shows that the nonlinear terms (like airflow rate × temperature) don't directly cause instability, simplifying the problem.
Controller Tuning: Provides an optimized method to adjust PID controller settings so that they work well with VAV systems.

The outcome of this paper is -
Stability Limits: The paper identifies stability limits based on controller choices and system settings. This helps in predicting when instabilities may occur. PID Tuning: Suggests slight modifications to standard PID controller settings (using gain reduction factors) to make them more suitable for VAV systems. No Need for Complexity: Since nonlinearity isn't the cause of instability, there's no major benefit to treating VAV systems as fully nonlinear during the stability analysis.

### Merits

The paper provides a comprehensive stability limit analysis for bilinear systems, identifying parameters and conditions critical to avoiding instabilities in real-world applications like Variable Air Volume (VAV) systems.
Focused on VAV systems in HVAC applications, the study offers insights into managing heating, ventilation, and air conditioning (HVAC) operations, which are essential for commercial and institutional buildings.
Developing a normalized bilinear model for energy flow in VAV systems enhances understanding and applicability in real-life setups. Proposes new PID tuning methods that accommodate the nonlinear nature of bilinear systems, with modifications to optimize controller parameters. Demonstrates that the "hunting" (instability) phenomenon isn't significantly caused by system nonlinearity, simplifying system design by downplaying the need to account for bilinear instability directly.
Provides an approach to modify linear PID parameters with gain reduction factors, ensuring they align with practical, real-world VAV systems. Includes simulation results to validate the proposed methods, illustrating the influence of control schemes and system parameters on stability.

### Demerits

The study is highly specific to VAV systems in HVAC applications and may not generalize well to other bilinear systems or industries. The tuning process, while optimized, requires an advanced understanding of the system and may not be straightforward for practitioners without specialized knowledge. Modifying PID parameters using gain reduction factors introduces an extra layer of computation, which may complicate system design and deployment. The analysis focuses heavily on stability limits and theoretical insights, with fewer details about practical implementations in diverse industrial settings.

By concluding that nonlinearities (like bilinear terms) don't significantly impact "hunting" phenomena, the approach may overlook edge cases where nonlinearity could still affect system performance. While time-delay feedback is considered, its implications on tuning strategies might not be fully explored or easily applicable across varying system setups.

## 2.3. Interpretable PID parameter tuning for control engineering using general dynamic neural networks: An extensive comparison

### Author's name

Johannes Günther ,Elias Reichensdörfer,Patrick M. Pilarski,Klaus Diepold

### Abstract

Modern automation systems largely rely on closed loop control, wherein a controller interacts with a controlled process via actions, based on observations. These systems are increasingly complex, yet most deployed controllers are linear Proportional-Integral-Derivative (PID) controllers. PID controllers perform well on linear and near-linear systems but their simplicity is at odds with the robustness required to reliably control complex processes. Modern machine learning techniques offer a way to extend PID controllers beyond their linear control capabilities by using neural networks. However, such an extension comes at the cost of losing stability guarantees and controller interpretability. In this paper, we examine the utility of extending PID controllers with recurrent neural networks——namely, General Dynamic Neural Networks (GDNN); we show that GDNN (neural) PID controllers perform well on a range of complex control systems and highlight how they can be a scalable and interpretable option for modern control systems. To do so, we provide an extensive study using four benchmark systems that represent the most common control engineering benchmarks. All control environments are evaluated with and without noise as well as with and without disturbances. The neural PID controller performs better than standard PID control in 15 of 16 tasks and better than model-based control in 13 of 16 tasks. As a second contribution, we address the lack of interpretability that prevents neural networks from being used in real-world control processes. We use bounded-input bounded-output stability analysis to evaluate the parameters suggested by the neural network, making them understandable for engineers. This combination of rigorous evaluation paired with better interpretability is an important step towards the acceptance of neural-network-based control approaches for real-world systems. It is furthermore an important step towards interpretable and safely applied artificial intelligence.

### Merits

By integrating neural networks with traditional PID control, the method enhances performance in complex, nonlinear, and disturbance-prone environments where classical PID may fall short. The demonstrated superior performance in 15 out of 16 tasks compared to standard PID controllers—and in 13 out of 16 tasks against model-based controllers—underscores its effectiveness across a range of benchmark control problems. Additionally, the method maintains scalability, which is crucial for deployment in varied and large-scale systems. Another significant advantage is the effort to regain interpretability through bounded-input bounded-output (BIBO) stability analysis, enabling engineers to understand and trust the neural network-suggested control actions. This addresses a major limitation of neural networks in control applications and represents progress toward the safe and practical adoption of AI-driven controllers in real-world systems.

### Demerits

Despite improvements in interpretability, the reliance on neural networks inherently introduces complexity and can obscure the underlying control logic compared to traditional PID. The interpretability method, while helpful, may not fully mitigate the black-box nature of neural models for all users, especially in high-stakes industrial applications. Stability guarantees, a cornerstone of classical control theory, may still be less rigorous or harder to generalize in neural-enhanced systems, potentially posing risks in safety-critical scenarios. Moreover, training and tuning neural networks require significant computational resources and expertise, which might limit accessibility and adoption in resource-constrained settings. Finally, while benchmark systems provide valuable insights, real-world systems often introduce additional uncertainties, hardware constraints, and safety considerations that may affect the performance and reliability of neural PID controllers outside the controlled evaluation setup.

## 2.4. PID Control Parameter Tuning Using Linear Multivariate Model

**Author's name**
Arnab Giri, Nilava Debabhuti, Prolay Sharma, Sudipta Mondal, Subhashis Das & Pranibesh Mandal

### Abstract

This paper presents a machine learning-based approach for tuning the Proportional Integral Derivative (PID) parameters of a PID controller. PID control system is widely used in recent times for controlling mechanisms in different industrial applications. The present work develops a model using the Partial Least Square Regression (PLSR) algorithm to predict the control parameters of speed control of a D.C motor based on output features viz. peak time (tp), peak value (ypeak), percentage overshoot (%OS), settling time (tss), and rise time (tr). The predictive model produces Root Mean Square Error of Calibration (RMSEC) of 0.0234, 0.0795, and 0.6145 for Kp, Kd, and Ki, respectively.

### Merits

First, it leverages a data-driven approach to predict optimal PID parameters based on key time-domain performance indicators like peak time, overshoot, and settling time, which are commonly used in control system analysis. This makes the method intuitive and aligned with practical control objectives. The use of PLSR is particularly advantageous in handling multicollinearity among input variables and reducing dimensionality while preserving the relationship between system response features and PID parameters. The reported Root Mean Square Error of Calibration (RMSEC) values—especially the low errors for $K_p$ and $K_d$ - indicate good prediction accuracy for those parameters, suggesting the model's effectiveness in learning from training data for the speed control of a DC motor. Additionally, the approach avoids manual tuning and iterative trial-and-error, potentially saving time and effort in industrial PID deployment.

### Demerits

The significantly higher RMSEC value for $K_i$ ( 0.6145 ) compared to $K_p$ and $K_d$ suggests weaker predictive performance for the integral component, which is critical for eliminating steady-state error. This could lead to suboptimal or unstable behavior in certain applications. Moreover, the method appears to be system-specific—developed and validated only for DC motor speed control—limiting its generalizability to other types of systems without retraining. The model's effectiveness also heavily relies on the quality and representativeness of the training data, and performance may degrade if system dynamics change or noise and disturbances are present. Lastly, while PLSR is interpretable compared to more complex machine learning techniques, it still adds a layer of abstraction compared to traditional PID tuning rules, which might be a barrier for engineers unfamiliar with regression-based modeling.

## 2.5. Adaptive PID controller for stable/unstable linear and nonlinear systems.

### Author’s name

B.M. Badreddine,Feng Lin.

### Abstract

The paper proposes and analyzes a direct adaptive PID (APID) control scheme for automatically tuning PID parameters both offline and online. The objective is to adaptively adjust the PID parameters to minimize an error function, ensuring optimal performance for both linear and nonlinear systems. The tuning algorithm relies on adaptive interaction theory and includes two methods: Frechet method and approximation method. Stability is proven using Lyapunov theory, and the control scheme is designed to be robust by eliminating the need for explicit plant knowledge.

### Merits

Automatically tunes PID parameters to ensure optimal control, even in changing system dynamics.

Suitable for both linear and nonlinear plants, making the scheme highly versatile.

The approximation method does not require detailed plant knowledge, allowing it to handle system uncertainties and changes effectively.

Stability of the approximation method is proven rigorously using Lyapunov stability theory, providing theoretical reliability.

Directly minimizes an error function, ensuring improved performance metrics like stability, speed, and accuracy.

Supports real-time adjustments (online tuning) and pre-implementation configurations (offline tuning) for enhanced control flexibility.

Introduces two tuning algorithms (Frechet and approximation methods) to cater to different scenarios and requirements.

In-depth performance analysis and convergence properties are provided via computer simulations and established adaptation concepts.

### Demerits

Adaptive algorithms, especially the Frechet method, can be computationally intensive, potentially limiting real-time applications.

The success of the tuning process depends heavily on the proper selection and definition of the error function.

While simulations are provided, the paper lacks thorough experimental validation on real-world systems.

The Frechet method may be harder to implement and analyze compared to the approximation method, potentially restricting its usability.

Though robust, the scheme may lag in performance for systems with extremely rapid changes or high-frequency dynamics.

While Lyapunov theory ensures stability, applying it practically requires a strong mathematical understanding, which might be a barrier for general practitioners.

## 2.6. Stability analysis of PWM feedback control systems with PID regulators.

### Author's name

M. LA CAVA,G. PALETTA & C. PICARDI

### Abstract

The paper focuses on the stability analysis of Pulse-Width Modulation (PWM) feedback control systems that utilize a Proportional-Integral-Derivative (PID) regulator. It extends previous research that predominantly addressed proportional or proportional-integral regulators. Using the second Lyapunov method, the study derives stability conditions for PWM systems, considering both general PWM modulators and arbitrary-order controlled processes. Furthermore, the paper introduces a simplified and flexible procedure to expand the stability region by varying only a subset of parameters in the quadratic Lyapunov function.

### Merits

Expands stability studies in PWM systems to include PID regulators, addressing a more comprehensive range of control scenarios.

Handles arbitrary-order controlled processes and general types of PWM modulators, increasing the method's versatility.

The use of a discrete version of the second Lyapunov method provides a robust framework for analyzing nonlinear systems.

Proposes a novel procedure to enhance the stability region by varying only select parameters of the quadratic Lyapunov function, making it efficient and easier to apply.

The method is described as straightforward and adaptable, making it suitable for a variety of PWM systems.

The research is relevant for high-performance feedback control systems, especially in applications involving power amplifiers or switching systems.

Provides clear insight into the stability properties of nonlinear PWM feedback control systems, aiding in better system design and tuning.

### Demerits

The study is primarily theoretical and may lack extensive practical validation or experimental results on real-world systems.

While applicable to arbitrary-order processes, the analysis may become computationally challenging for very high-order systems.

The stability results heavily depend on the choice and tuning of the parameters in the quadratic Lyapunov function, which can be nontrivial.

The findings are tailored for PWM-based systems and might not generalize well to other types of control systems or regulators.

Nonlinear analysis requires deep expertise, which may pose difficulties for general control engineers attempting to replicate or extend the approach.

The paper focuses primarily on stability and does not delve into system performance aspects like transient behavior or robustness to disturbances.

## 2.7. PID control system analysis, design, and technology

### Author's name

Kiam Heong Ang, G. Chong, Yun Li

### Abstract

The paper aims to provide a modern overview of the advancements in PID controller design and tuning, particularly focusing on the incorporation of "intelligent" tools such as system identification and advanced algorithms. The main goal is to highlight the progress in creating intelligent, automated PID control systems that help engineers optimize PID performance with minimal manual adjustments, making these controllers more versatile and easier to use across a range of applications. The paper covers the evolution of PID control variants, advancements in software packages, and hardware modules designed for automated tuning and operation.

### Merits

The development of intelligent systems, such as software tools and hardware modules, automates the PID controller design and tuning, reducing manual intervention and effort.

Advanced algorithms are integrated into PID hardware and software, improving tuning precision and reducing the trial-and-error nature of PID adjustments.

Various PID variants are designed to improve both the transient response (speed of response) and the overall system stability, achieving a more balanced and optimal performance.

By incorporating system identification, the design process becomes more efficient, enabling the controller to adapt to varying system dynamics without requiring detailed system models.

The trend toward modular and standardized PID designs makes it easier for engineers to deploy these controllers in diverse applications, enhancing adaptability.

The potential development of "plug-and-play" PID controllers that can easily be set up and operate optimally across different environments makes it easier for users without advanced expertise to implement PID control.

### Demerits

These advanced tuning methods and system identification techniques can rely heavily on the accuracy of algorithms and models. Poor data quality or inaccuracies can lead to suboptimal performance.

While these tools automate much of the process, the design of advanced PID systems can still be complex and may require specialized knowledge, especially in configuring the intelligent algorithms.

The system identification process, if not properly managed, may lead to overfitting, where the controller performs well on the test system but struggles to adapt to different environments or variations.

While PID control is common and can be widely applied, the automation tools and intelligent algorithms might not always be universal or suitable for every type of application or system. Incorporating intelligent tools into both hardware modules and software requires seamless integration with existing systems, which can be difficult and costly in complex or legacy systems. Over-reliance on automation and intelligent tools can lead to potential vulnerabilities, as the system's performance might degrade if the automated adjustments are not monitored carefully or if unexpected system conditions arise.

## 2.8. A robust PID controller based on linear quadratic gaussian approach for improving frequency stability of power systems considering renewables.

### Author's name

Mohamed Khamies, Gaber Magdy, Mohamed Ebeed, Salah Kamel.

### Abstract

The paper aims to develop a robust controller called the Robust PID (RPID) controller to improve the frequency stability of power systems with a high penetration of renewable energy sources (RESs). This controller combines the Proportional–Integral–Derivative (PID) controller with the Linear Quadratic Gaussian (LQG) controller. The Improved Lightning Attachment Procedure Optimization (ILAPO) technique is used to optimally tune the parameters of the RPID controller.

### Merits

Enhanced Frequency Stability: The RPID controller effectively improves the frequency stability of power systems with high RES penetration.

Optimal Parameter Tuning: The ILAPO technique ensures that the RPID controller parameters are optimally set for the best performance.

Handling System Uncertainties: The controller accounts for uncertainties in load and RES output, as well as system nonlinearities like generation rate constraints and governor dead band.

Superior Performance: Comparisons show that the RPID controller outperforms other robust controllers in maintaining frequency stability under various scenarios.

Versatility: The controller is suitable for modern power systems with high RES integration.

### Demerits

Complexity: The combination of PID and LQG controllers, along with the ILAPO optimization, may add complexity to the control system design and implementation.2. Computational Requirements: The optimization process for tuning the controller parameters could be computationally intensive.

Dependence on Accurate Modeling: The effectiveness of the RPID controller depends on accurate modeling of the power system dynamics and uncertainties.

Limited Testing Scenarios: While the paper demonstrates the controller's performance in certain scenarios, further testing in diverse and real-world conditions is necessary for broader validation.

## 2.9. PID control system analysis and design

**Author's name**

Yun Li, Kiam Heong Ang, G.C.Y Chong

**Abstract**

The main objective of PID control is to regulate a process to maintain a desired setpoint by adjusting control inputs. It does this by using three components:

1. Proportional (P): Responds to the current error (difference between the desired setpoint and the actual value).
2. Integral (I): Accounts for past errors by summing them over time, helping eliminate residual steady-state errors.
3. Derivative (D): Predicts future errors by considering the rate of change of the error, which helps to reduce overshoot and improve stability.

**Merits**

Versatility: Can be applied to a wide range of processes (e.g., temperature control, motor speed control).

Improves Stability: The derivative term helps to predict and minimize future errors, reducing oscillations.

Eliminates Steady-State Error: The integral term ensures that any persistent error is corrected over time.

Simple and Easy to Implement: Despite its powerful capabilities, PID control is relatively straightforward to implement in most control systems.

**Demerits**

Tuning Complexity: Finding the optimal values for the proportional, integral, and derivative terms can be challenging and time-consuming.

Sensitivity to Noise: The derivative term can amplify high-frequency noise in the process signal, leading to erratic behavior.

Integral Windup: When the integral term accumulates a large error over time, it can lead to overshooting or instability if not properly managed.

Not Suitable for All Systems: PID controllers may not perform well for systems with significant delays or highly nonlinear behavior.

## 2.10. A comparative overview of different approaches for calculating the set of all stabilizing PID controller parameters.

### Author’s name
Frank Schrödel, Shivnath Karthikeyan Manickavasagam, Dirk Abel

### Abstract

The paper aims to analyze and compare different approaches for calculating stabilizing PID controller parameters. The goal is to develop effective and efficient methods to determine stability boundaries and regions for PID control systems, particularly those involving time delays, stable or unstable plants, and higher-order systems. It does this by using three components.

### Merits
The paper compares three major methods - Hermite-Biehler Theorem (HBT), Polynomial Stability Algorithm (PSA), Dual Locus Diagram. This gives insight into their strengths, weaknesses, and applicability.

Works for any-order plants, including stable, unstable, and time-delay systems.

Fast and accurate without the restrictions imposed by other methods.

Avoids conservative results that some HBT implementations give.

Handles Interval Robust Bounds (IRB) which HBT ignores.
Useful for low-order plants without time delays.

Stability boundaries can serve as a starting point for further optimization of controller parameters.

A MATLAB-based toolbox is introduced that includes both HBT and PSA methods. This helps compare performance easily and transparently.

Simple and quick to use for analyzing systems with time delays. Requires less computational effort, making it easy to use.

### Demerits
Limited scope: Only works well for low-order systems and systems without time delays.

Can yield conservative results, potentially leading to suboptimal performance.

Limited to simple analysis and doesn't find the entire stability region for controller parameters.

No generalized implementations exist for all types of systems.

While PSA is shown to be robust and versatile, it might not yet have gained the popularity or widespread implementation as compared to HBT.

The Dual Locus method lacks thorough coverage of stability regions, reducing its real-world utility for complex systems.

HBT remains a more popular method in literature, as seen in famous books and new publications, even though it is less effective than PSA for certain applications.

## 2.11. Tuning of PID-type controllers for stable and unstable systems with time delay

### Author's name
Z. Shafiei, A.T. Shenton

### Abstract
The objective of the paper is to present a graphical technique for tuning PID-type controllers using the D-partition method for single-input-single-output (SISO) linear time-invariant (LTI) systems. This technique ensures that:

The roots of the characteristic equation are placed in a desirable region of the left half-plane (LHP), achieving specified stability margins.

The method improves both the stability and tracking performance of the system.

It can be applied to a variety of system conditions, including stable, unstable systems, and systems with time delays.

### Merits
The graphical nature of the method provides a visual understanding of the root locations, making it intuitive for engineers to tune the controller effectively.

It is applicable to different system types, including:

Stable and unstable systems.

Systems with significant time delays.

Various controller configurations like PID with derivatives in the feedback path.

The method shifts the roots to a region that ensures both least absolute and least relative stability margins, resulting in a robust system.

Tuning pre-compensators in a two-degree-of-freedom (2-DOF) structure can improve tracking performance, addressing transient response and steady-state errors.

By analyzing the entire D-partition graph, the technique provides insights into how the controller's parameters influence the system stability and performance.

### Demerits
The method may become computationally and visually complex for high-order systems, where there are many roots and parameter dependencies.

The effectiveness of the technique relies on the accuracy of the system model. Any modeling errors can result in suboptimal tuning or instability.

While graphical methods are intuitive, they may lack automation and can be time-consuming compared to modern optimization-based tuning algorithms.

The method is suited for linear time-invariant (LTI) systems and may not work well with nonlinear or time-varying systems. Using the D-partition method effectively requires a solid understanding of root-locus and stability principles, which may not be intuitive for all practitioners.

# 3. Chapter 3: Research Gap

## 3.1. Data-driven optimal tuning of PID controller parameters.

Traditional approaches often rely on persistent excitation and assume linear system models or known system dynamics, limiting their effectiveness in real-world nonlinear systems subjected to unknown disturbances. This study addresses these limitations by proposing a novel tuning method that integrates recursive optimization and concurrent learning without requiring persistent excitation[1]. However, a research gap remains in the generalizability and real-time applicability of such methods across diverse nonlinear system classes and operational conditions. Furthermore, while the approach transforms the system into a solvable structure by introducing new variables and treating PID parameters as model parameters, the robustness of this transformation under varying disturbance characteristics or measurement noise is not deeply explored. Additionally, most existing methods, including this one, focus on simulation-based validation; there is a lack of experimental studies validating the practical implementation and computational efficiency of such algorithms in hardware or embedded control systems. So, further research is needed to explore the scalability, real-time implementation feasibility, and adaptability of these algorithms in complex, high-dimensional, or multi-input multi-output systems.

## 3.2. Stability analysis and tuning of PID controllers in VAV systems.

The analysis relies on a simplified model that, while effective for theoretical insight, may not capture all the dynamics and interactions present in real-world VAV systems, such as actuator delays, sensor noise, or multi-zone coupling effects in large buildings [2]. Additionally, although the study suggests that nonlinear terms like the product of airflow rate and temperature do not directly cause instability, this conclusion may not generalize across all operational conditions or system configurations, particularly under extreme disturbances or in systems with more complex airflow dynamics [5]. Moreover, the proposed PID tuning method, which involves gain reduction factors, lacks detailed validation in practical scenarios or through real-time implementation, leaving questions about its robustness and performance in diverse building environments [14]. Finally, the current work focuses on single-zone control, and further research is needed to extend these findings to multi-zone systems with interacting dynamics and to explore data-driven or adaptive control methods that can respond to time-varying changes in occupancy, weather, or equipment performance.

## 3.3. Interpretable PID parameter tuning for control engineering using general dynamic neural networks: An extensive comparison

The proposed neural PID controllers demonstrate strong performance across benchmark systems, but their effectiveness and generalizability in real-world industrial environments—where system dynamics can be nonstationary, multimodal, or subject to unexpected faults—are yet to be fully explored[7]. Although the study introduces bounded-input bounded-output (BIBO) stability analysis to address interpretability, this analysis may not capture the full range of dynamic behaviors in nonlinear or time-varying systems, especially when neural networks adapt online. Moreover, while interpretability is improved by analyzing suggested parameters, the learning dynamics and internal representations of the neural networks still remain largely opaque, potentially limiting trust and adoption in high-stakes applications. Another gap lies in the scalability of the method to large-scale, multi-input multi-output (MIMO) systems where interactions between subsystems can introduce additional complexity. Lastly, the robustness of neural PID controllers under adversarial conditions (e.g., cyber-physical attacks, extreme noise, or actuator degradation) has not been addressed in depth, indicating a need for further research into fault-tolerant and secure learning-based control.

## 3.4. PID Control Parameter Tuning Using Linear Multivariate Model

The model's predictive capability is evaluated based on standard time-domain performance features, but it is unclear how well the approach generalizes to other dynamic systems beyond DC motors or to systems with different levels of nonlinearity and disturbances[15]. Moreover, while PLSR is a relatively simple and interpretable regression technique, it may not capture complex, nonlinear relationships between system performance metrics and optimal PID parameters, potentially limiting its applicability in more intricate control environments. Additionally, the study focuses on offline prediction using simulation or experimental data, but it does not address how the model adapts to real-time changes in system dynamics or external disturbances. There is also limited discussion on the robustness and reliability of the predicted parameters when deployed in closed-loop control under varying operating conditions. Finally, the relatively high RMSEC for the integral gain ($K_i$) suggests that further refinement or alternative modeling approaches might be needed to enhance prediction accuracy and ensure stable and reliable control in practical applications.

## 3.5. Adaptive PID controller for stable/unstable linear and nonlinear systems.

The algorithm’s reliance on adaptive interaction theory, though effective, may involve computational complexities or convergence issues in real-time applications, particularly in high-speed or safety-critical systems [9]. Although Lyapunov-based stability analysis provides theoretical guarantees, the practical implementation and real-time performance of the APID controller under varying noise, disturbances, or sensor inaccuracies are not thoroughly addressed [10]. Furthermore, while the method eliminates the need for explicit plant models, it is unclear how well the controller handles systems with time-varying dynamics, actuator constraints, or delays—factors common in real-world applications. Additionally, the comparative performance of the Frechet and approximation methods under different operating conditions is not fully explored, leaving a gap in understanding which method is more suitable for specific scenarios. Finally, the absence of experimental validation or deployment in real industrial settings limits insight into the robustness, interpretability, and scalability of the proposed adaptive approach beyond simulation environments.

## 3.6. Stability analysis of PWM feedback control systems with PID regulators.

Most notably, the study focuses primarily on theoretical derivation and stability conditions without addressing the practical implementation challenges associated with PWM-PID systems, such as quantization effects, switching delays, or non-idealities in digital controllers[16]. Additionally, although the proposed method offers a simplified approach to expanding the stability region by adjusting a subset of Lyapunov function parameters, it does not explore the impact of system uncertainties, time-varying dynamics, or external disturbances on the robustness of the derived stability regions. The generality of the proposed stability conditions across different types of nonlinear and high-order systems also remains unverified through simulation or experimental validation [17]. Furthermore, the computational efficiency and scalability of the procedure in real-time applications are not discussed, leaving open questions regarding its feasibility in embedded or resource-constrained control systems [6]. Lastly, there is a lack of comparison with alternative modern stability analysis techniques, such as data-driven or adaptive methods, which could provide further insight into the advantages or limitations of the proposed approach [16].

## 3.7. PID control system analysis, design, and technology

The review highlights the integration of system identification techniques and advanced algorithms to automate PID tuning, but it lacks a critical evaluation of the limitations and comparative performance of these intelligent methods across different types of systems, particularly in nonlinear, time-varying, or highly uncertain environments[17]. Additionally, while software and hardware advancements are discussed, the interoperability, standardization challenges, and real-time implementation constraints of these automated solutions in diverse industrial settings are not thoroughly addressed. The paper also does not explore how these intelligent tools perform under practical disturbances such as sensor noise, actuator saturation, or communication delays. Furthermore, the human-in-the-loop aspect—balancing automation with operator oversight and interpretability—is underexplored, which is crucial for the broader acceptance and trust in automated PID systems. Lastly, there is a need for further investigation into the scalability of intelligent PID tuning methods for large-scale or multi-loop control architectures, where interactions between multiple controllers can significantly impact overall system performance.

## 3.8. A robust PID controller based on linear quadratic gaussian approach for improving frequency stability of power systems considering renewables.

The effectiveness of the RPID controller is primarily focused on frequency stability in power systems with high renewable energy penetration, but the study does not fully address the controller's adaptability to varying renewable generation profiles, such as rapid fluctuations from solar or wind sources [10]. Additionally, while the ILAPO algorithm offers an optimization-based approach, its computational complexity and real-time applicability in large-scale power systems are not thoroughly examined. The integration of LQG into the PID framework may enhance performance, but the resulting controller structure could compromise interpretability and ease of implementation in existing control infrastructures. Moreover, the robustness of the proposed controller under severe grid disturbances, communication delays, or hardware constraints is not clearly evaluated [8]. Finally, experimental validation or hardware-in-the-loop testing is not mentioned, limiting insight into the practical deployment and scalability of the RPID controller in real-world power systems [9].

## 3.9. PID control system analysis and design.

Despite its widespread use, PID control faces challenges when applied to complex, nonlinear, or time-varying systems, where fixed gains may not yield optimal performance[17]. The traditional PID approach also lacks adaptability in the face of disturbances, model uncertainties, and system constraints such as actuator saturation or time delays [18]. Additionally, the tuning of PID parameters remains largely heuristic or manual in many applications, which can lead to suboptimal performance, especially in dynamic environments. Moreover, the abstract does not consider the integration of modern techniques—such as adaptive control, machine learning, or optimization algorithms—that can enhance PID performance in advanced control scenarios. The interpretability-versus-performance trade-off in hybrid or intelligent PID systems also remains an open research issue. Thus, there is a need for further investigation into adaptive, data-driven, and robust PID frameworks that can maintain stability and performance across a broader range of operating conditions and system complexities [3].

## 3.10. A comparative overview of different approaches for calculating the set of all stabilizing PID controller parameters.

The study primarily focuses on calculating stability boundaries and regions, but it does not sufficiently address the practical implementation challenges of these methods in real-time control systems, particularly those with rapidly changing dynamics or uncertainties[13]. Although time delays, unstable plants, and higher-order systems are considered, the robustness of the proposed approaches under model inaccuracies, external disturbances, and nonlinearities is not thoroughly explored. Additionally, the paper does not delve into the computational efficiency or scalability of these methods for complex multi-input multi-output (MIMO) systems or large-scale industrial processes. There is also limited discussion on how these theoretical approaches can be integrated with modern intelligent tuning techniques, such as machine learning or adaptive control, to enhance flexibility and automation. Furthermore, the lack of experimental validation or application to real-world case studies leaves open questions regarding the practical utility and generalizability of the findings.

## 3.11. Tuning of PID-type controllers for stable and unstable systems with time delay

The method’s applicability is confined to linear time-invariant systems, leaving open questions about its effectiveness for nonlinear, time-varying, or multi-input multi-output (MIMO) systems commonly encountered in practical applications [6]. While the technique addresses stability and tracking performance, it does not explicitly consider robustness against model uncertainties, disturbances, or noise, which are critical for real-world control systems [12]. Furthermore, the graphical nature of the D-partition method may become cumbersome or less intuitive for higher-order systems or those with complex dynamics, limiting scalability [18]. The paper also does not discuss the integration of this approach with modern adaptive or intelligent tuning methods that could automate and enhance controller performance. Lastly, practical implementation challenges, such as computational efficiency and real-time applicability, are not explored, and experimental validation on physical systems is lacking, restricting the assessment of its real-world usability [6].

# 4. Chapter 4: Theory, Methodology, Materials & Methods

## Methodology

The methodology employed in this study integrates the principles of control system dynamics, computational machine learning techniques, and rigorous testing procedures. The goal is to analyze the stability margins of PID-controlled systems and identify input-output relationships using advanced predictive frameworks [1], [3], [4]. This section provides a detailed description of the physical dynamics, computational analysis, testing and validation processes, and reliability and stability evaluation.

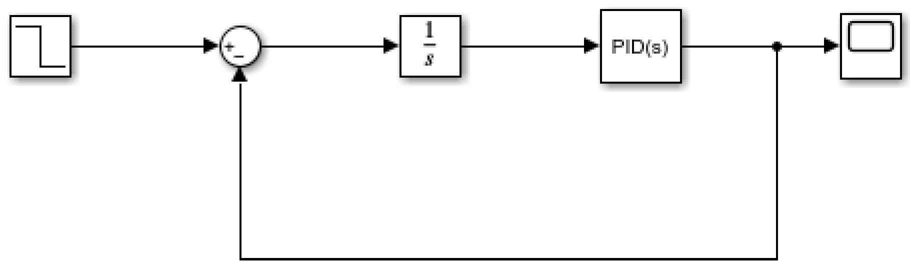


Fig 4.1: Block Diagram of the first order LTI System

### 4.1. Physical Dynamics of the Controller

The physical dynamics of the system are modeled using a closed-loop control structure. The system comprises a step input, summing junction, integrator, PID controller, and plant [2], [17]. The PID controller is the core element, ensuring the system's response tracks the desired reference signal while minimizing errors [2], [5], [18]. The dynamics of the system components are described as follows:

Step Input: Represents a sudden change or disturbance in the system. It provides initial test conditions to study the transient and steady-state response of the system.

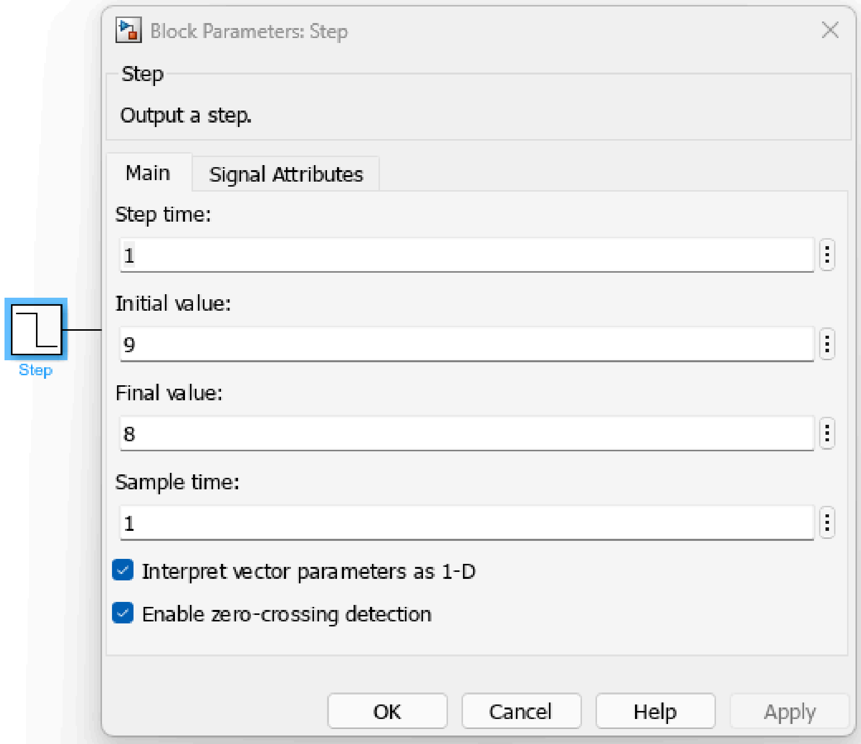
The step block has 4 changeable parameter -

- Step time: It is the time at which the step change occurs. It determines when the output of the step signal transitions from the Initial value to the final value. Eg - If the Step Time is set to 5 seconds, the signal remains at the Initial value until 5 seconds and then transitions to the Final value at that time.

- Initial value: It is the value of the signal before the step change occurs (i.e., before the Step Time). It sets the baseline or starting value of the signal. Eg - If the Initial Value is set to 1, the signal remains constant at 1 until the Step Time.

- Final value: The value of the signal after the step change occurs (i.e., after the Step Time). Specifies the target value the signal transitions to at the Step Time. Eg - If the Final Value is set to 5, the signal jumps from the Initial Value to 5 at the Step Time.

- Sample time: The time interval at which the block calculates the signal value. Determines whether the signal is continuous or discrete. Eg - For a Sample Time of 0.1 seconds, the signal is updated every 0.1 seconds.



Step

Output a step.

Step time:

1

Initial value:

9

Final value:

8

Sample time:

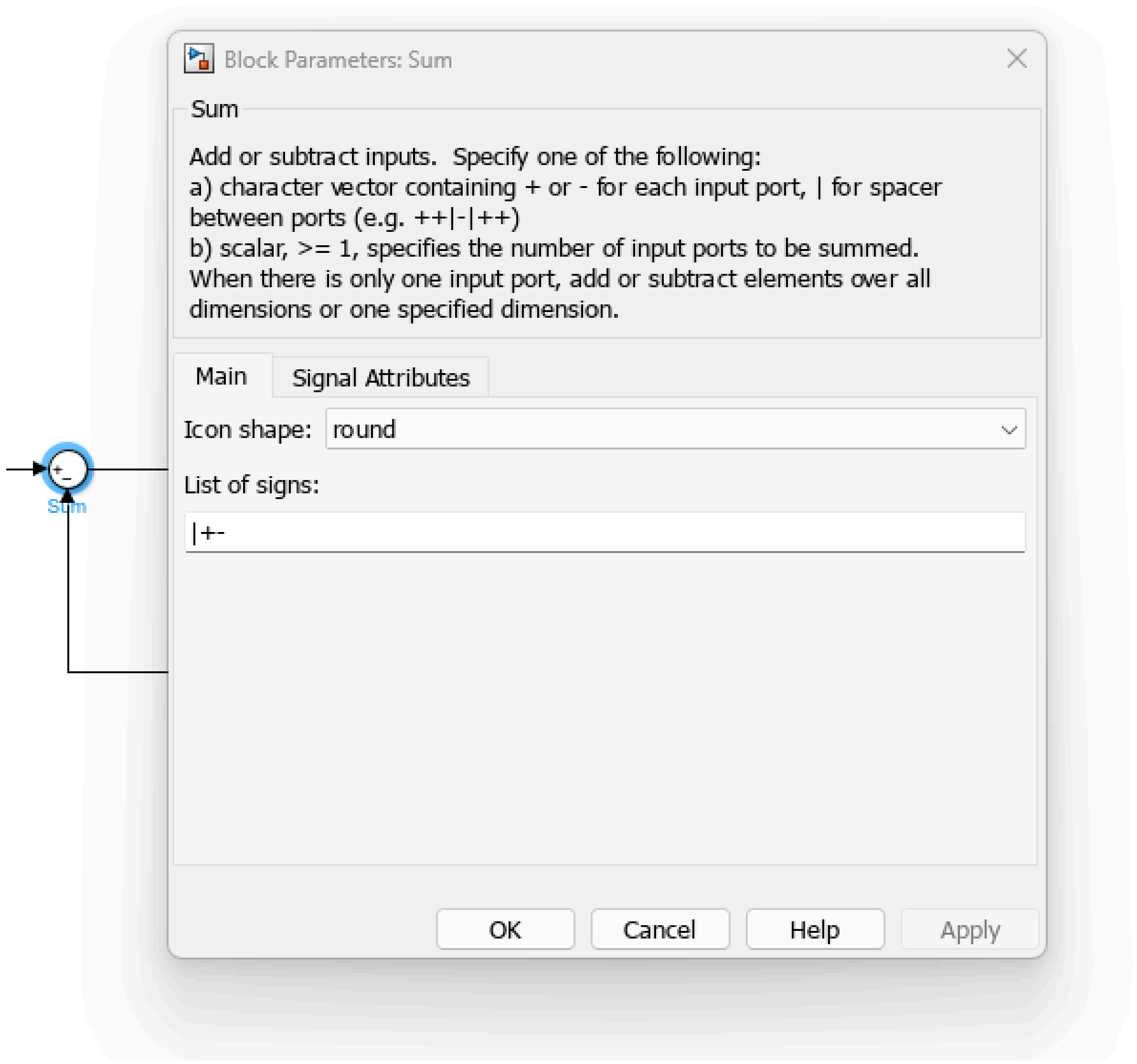
1

☑ Interpret vector parameters as 1-D

☑ Enable zero-crossing detection

Fig 4.2: Step Block

Summing Junction: Calculates the error signal by subtracting the feedback signal from the desired reference input.



Add or subtract inputs. Specify one of the following:
a) character vector containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension.

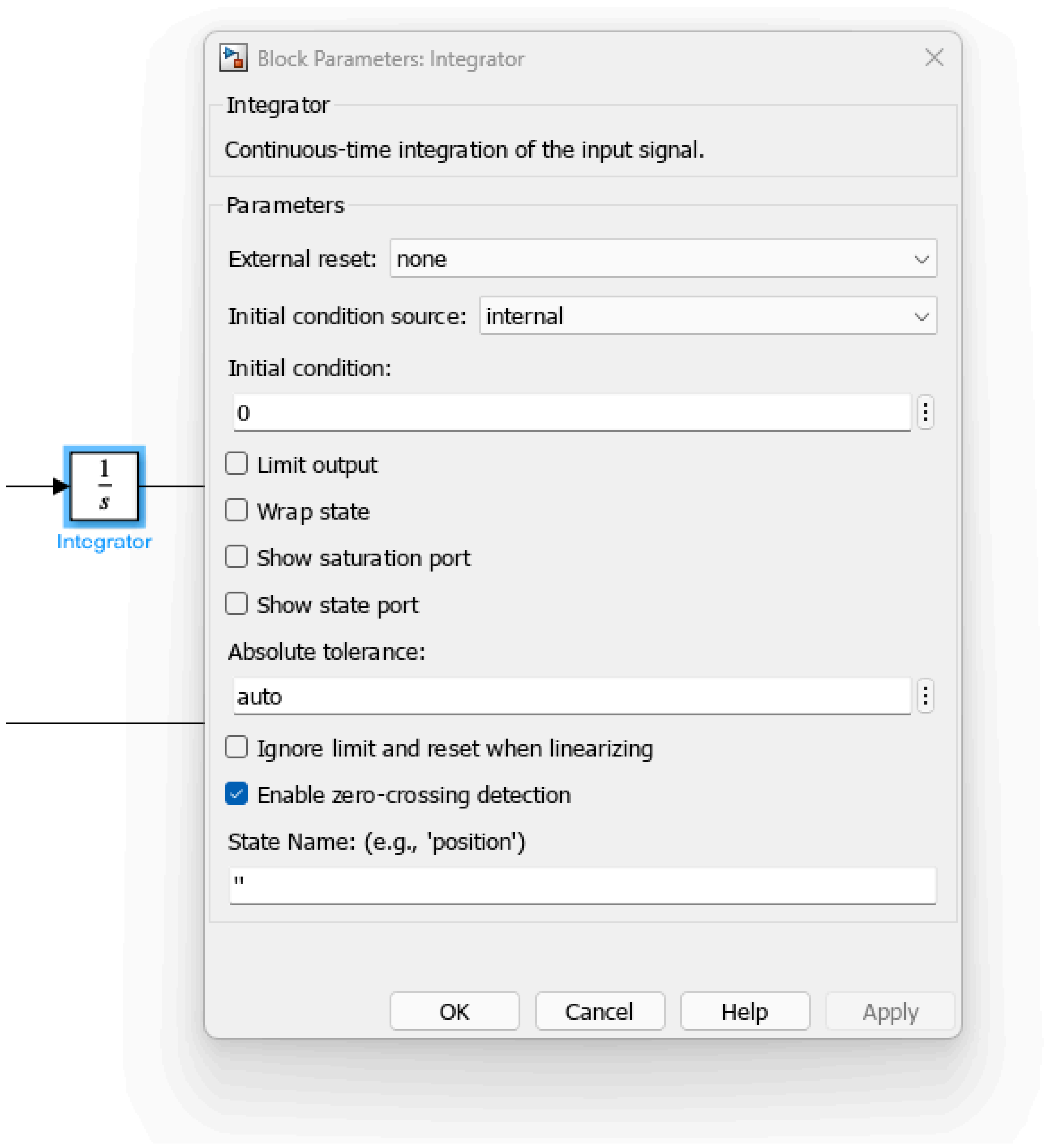
Icon shape: round

List of signs:

|+-

Fig 4.3: Sum Block

Integrator: Simulates the accumulation of error over time, representing the integral action of the PID controller.



Block Parameters: Integrator

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

0

☐ Limit output

☐ Wrap state

☐ Show saturation port

☐ Show state port

Absolute tolerance:

auto

☐ Ignore limit and reset when linearizing

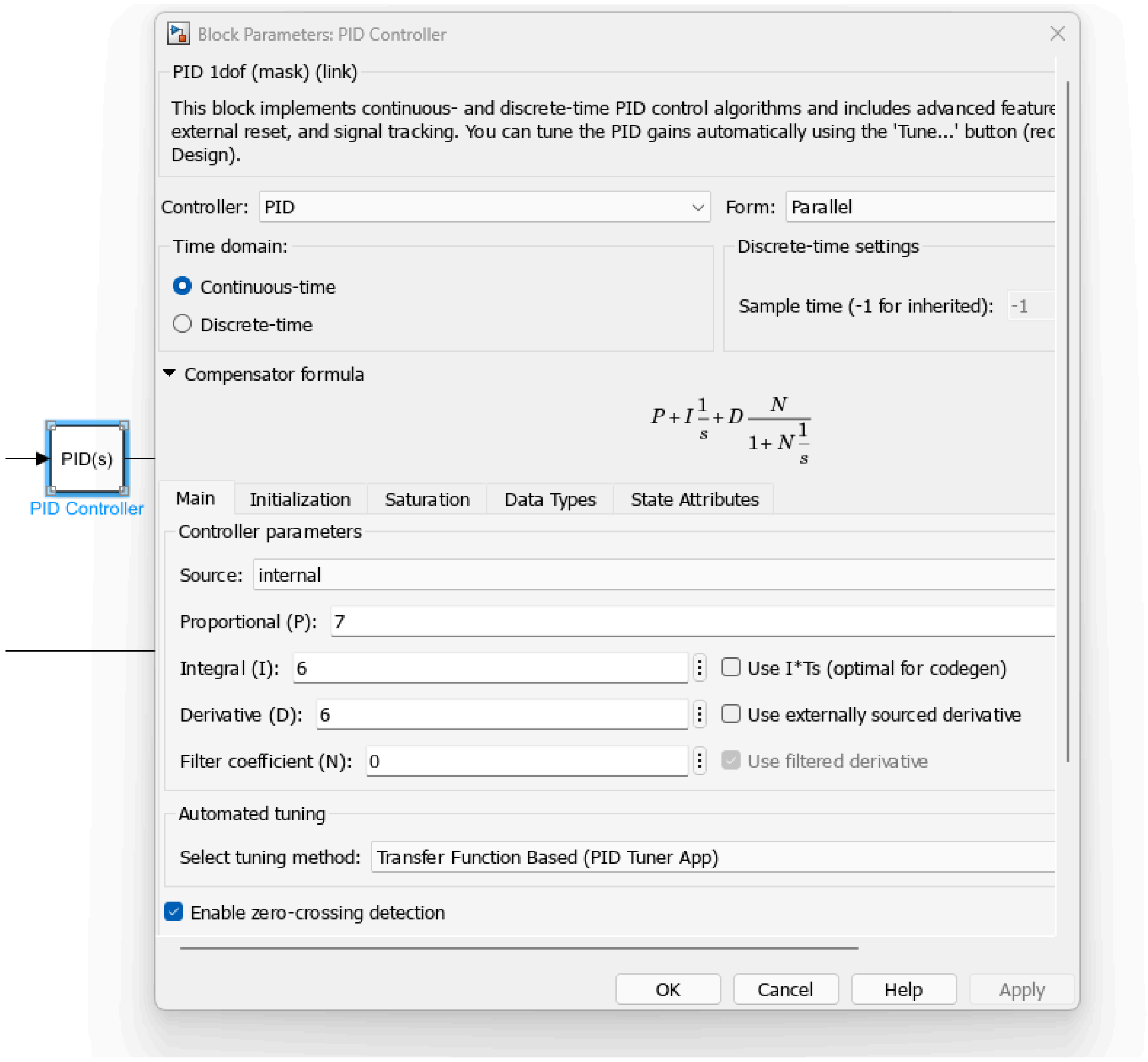
☑ Enable zero-crossing detection

State Name: (e.g., 'position')

''

Fig 4.4: Integrator Block

PID Controller: Combines proportional, integral, and derivative actions to correct the system response. The controller parameters ($K_p$, $K_i$, $K_d$) play a critical role in determining system stability.



Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced feature external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (rec Design).

Controller: PID

Form: Parallel

Time domain:

- [✓ Continuous-time]
- Discrete-time

Discrete-time settings

Sample time (-1 for inherited): -1

Compensator formula

$$P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}}$$

Main | Initialization | Saturation | Data Types | State Attributes

Controller parameters

Source: internal

Proportional (P): 7

Integral (I): 6 ☐ Use I*Ts (optimal for codegen)

Derivative (D): 6 ☐ Use externally sourced derivative

Filter coefficient (N): 0 ☑ Use filtered derivative

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

☑ Enable zero-crossing detection

Fig 4.5: PID Controller

Scope: Shows whether the graph stabilizes or remains unstable within a certain period of time.

The goal is to assess how variations in the PID parameters and step input affect the system's stability and performance.

## 4.2. Computational Analysis

The computational analysis involves modeling the physical system and applying machine learning algorithms to predict stability outcomes. This includes two key aspects:

The Machine Learning Algorithm used is Support Vector Machine (SVM). The goal of SVM is to find the optimal hyperplane that separates the data into two classes of stable and unstable inputs with maximum margin.

Types of SVM -

- **Linear SVM -** Used when data is linearly separable.
- **Non-Linear SVM -** Uses kernel tricks to transform data into higher dimensions to make it separable.

As the data in the dataset is not linearly separable, a non-linear SVM needs to be used.

**Algorithm of the SVM**

1. **Input**
- Training Dataset:

  $D = \{(x_1, y_1), (x_2, y_2)....(x_1, y_n)\}$

  $x_i \in R^d$ (feature vectors)

  $y_i \in \{ 0 , 1 \}$ (class labels)

- Hyperparameters

  Regularization parameter C=1

  RBF kernel parameter $\gamma$=0.1

2. **Preprocessing**
- Converts the class labels from { 0 , 1 } to { -1 , +1 } for SVM formulation

3. **Define the RBF kernel**
   - $K ( x_i , x_j ) = \exp( -\gamma \,||\, x_i - x_j \,||^2 )$

4. **Formulate the Dual Optimization Problem**

   Maximize the following:

$$\text{Max } \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j K(x_i, x_j)$$

   Subject to the constraints:

$$\sum_{i=1}^{n} a_i y_i = 0, 0 \leq a_i \leq c = 1$$

5. **Solve the Quadratic Programming (QP) Problem**

   Use a QP solver (such as Sequential Minimal Optimization - SMO) to find the optimal Lagrange multipliers $a_i$.

6. **Compute the Bias Term b**

   Select any support vector $x_k$ such that 0< $a_k$< $C$ , and compute

$$b = yk - \sum_{i=1}^{n} a_i y_i K(x_i, x_k)$$

7. **Define the Decision Function**

   For a new input $x$ , compute the decision function:

$$f(x) = sinn(\sum_{i=1}^{n} a_i y_i K(x_i, x) + b)$$

   Then map the output back to the original label space:

   If $f(x)$ = +1 , then predict class 1

   If $f(x)$ = −1 , then predict class 0

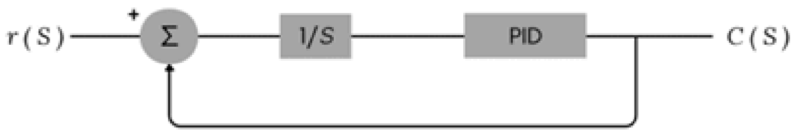
8. **Output**

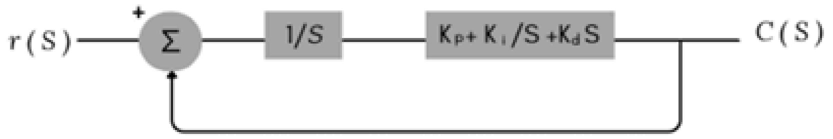
   A trained SVM classifier capable of predicting class labels $y \in \{0,1\}$ for new input data points.

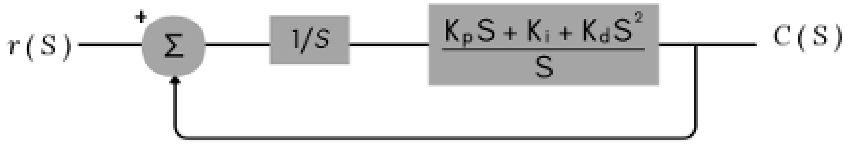
## 4.3. Physical Computation

**Stability Analysis of Linear System using Routh's Stability Criteria**

$C(S) = K_p + \frac{K_i}{S} + K_d S$ $S = (jw)$





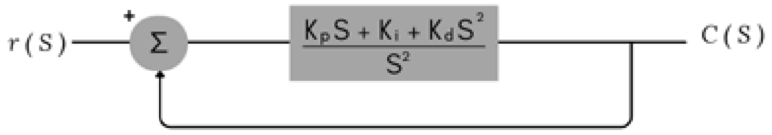


Since we know that the general transfer function of the closed loop system is

$T(S) = \frac{G(S)}{1+G(S)H(S)}$

where $T(S)$ is the forward gain and $H(S)$is the feedback gain of any linear system. Here $S = J$omega as $S$ is defined in laplace domain.

Hence,



$$T(S) = \frac{C(S)}{r(S)} = \frac{\frac{K_pS+K_i+K_dS^2}{S^2}}{1+\frac{K_pS+K_i+K_dS^2}{S^2}}$$

$$\therefore T(S) = \frac{K_pS+K_i+K_dS^2}{S^2+K_pS+K_i+K_dS^2} \qquad [\because\ S = jw]$$

where $K_p$, $K_d$ and $K_i$ are defined as proportional gain, differential gain and integral of the defined pid controller as shown in the given diagram.

$$T(S) = \frac{K_pS + K_dS^2 + K_i}{S^2(1 + K_d) + K_pS + K_i} = \frac{B(S)}{F(S)}$$

where $B(S)$ is defined as the numerator of the transfer function and $F(S)$ is the denominator of the first order LTI system.

For the stability analysis of any LTI System, the denominator of the transfer function $F(S)$ should be equal to 0, which is defined as the characteristic equation of any Linear system.

As we know, the general characteristic equation is

$$S^2 + 2\,\xi\,W_nS + {W_n}^2 = 0 \qquad [\because \xi = 1]$$

Hence $F(S)$ = 0, As per the first order linear system as shown above, the characteristic equation of the given transfer function is

$$\therefore S^2(1 + K_d) + K_pS + K_i = 0$$

$[\therefore Z = R + jx]$

| | | |
|---|---|---|
| $S^2$ | $(1 + K_d)$ | 1 |
| $S^1$ | $K_p$ | 0 |
| $S^0$ | 1 | |

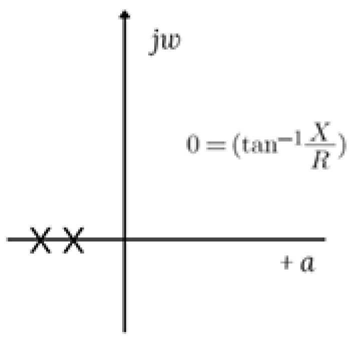


Fig 4.6

Phase angle = [ $a\cos\theta + jb\sin\theta$ ]

Since the signing of the first column does not change. Hence it can be decided that the system is stable as per the Rouths-Hurtwiz Stability Criterion.

# 5. Chapter 5: Results, Analysis and Discussions

The closed-loop system is simulated using tools such as MATLAB/Simulink to generate output responses for different combinations of input parameters [1], [6]. These simulations capture critical system behaviors, including transient response, steady-state error, overshoot, and settling time. The generated data is used as input for machine learning models [4], [7].

An automation tool such as PyAutoGUI is used to automate the process of data collection from Matlab/Simulink.

- **Program to automate the creation of continuous dataset -**

```
import pyautogui

import time

import os

import csv

import random



def openMatlab():

   # Open Matlab

   time.sleep(1)

   pyautogui.press('win')

   pyautogui.typewrite('mat', interval=0.2)

   pyautogui.press('enter')

   time.sleep(7)



   #Open Simulink

   pyautogui.moveTo(730, 80)

   pyautogui.leftClick()
```

```
    #Open Simulink Project
    time.sleep(4)
    pyautogui.moveTo(350, 280)
    pyautogui.leftClick()
    time.sleep(9)


def changeParams():
    #Open Step
    pyautogui.moveTo(770, 520)
    pyautogui.doubleClick()

    #Change Step params
    time.sleep(3)
    pyautogui.typewrite(inputParams[0])
    pyautogui.press('tab')
    pyautogui.press('tab')
    pyautogui.typewrite(inputParams[1])
    pyautogui.press('tab')
    pyautogui.press('tab')
    pyautogui.typewrite(inputParams[2])
    pyautogui.press('tab')
    pyautogui.press('tab')
    pyautogui.typewrite(inputParams[3])
```

```python
#Close Step

pyautogui.press('enter')



#Open PID

time.sleep(1)

pyautogui.moveTo(1060, 540)

pyautogui.doubleClick()



#Change PID params

time.sleep(3)

pyautogui.moveTo(1230, 570)

pyautogui.leftClick()

pyautogui.typewrite(inputParams[4])

pyautogui.press('tab')

pyautogui.press('tab')

pyautogui.typewrite(inputParams[5])

pyautogui.press('tab')

pyautogui.press('tab')

pyautogui.press('tab')

pyautogui.typewrite(inputParams[6])

pyautogui.press('tab')

pyautogui.press('tab')

pyautogui.press('tab')

pyautogui.typewrite(inputParams[7])



#Close PID
```

```
pyautogui.press('enter')


#Open scope

pyautogui.moveTo(1180, 530)

pyautogui.doubleClick()


#Run scope

time.sleep(2)

pyautogui.moveTo(760, 315)

pyautogui.leftClick()


#Take screenshot

time.sleep(5)

ss = pyautogui.screenshot()


#Save screenshot

folder_path = 'C:/dev/python/screenshotsCon'

os.makedirs(folder_path, exist_ok=True)

ss.save(os.path.join(folder_path, f'{inputParams}.jpg'))


#Close scope

time.sleep(1)

pyautogui.keyDown('alt')

pyautogui.press('tab')

pyautogui.keyUp('alt')
```

```
data = []

openMatlab()


for dec_num in range(200):

    inputParams = str(random.randint(10000000, 99999999))

    data.append(list(inputParams))

    changeParams()

    changeParams()


with open('conDataset.csv', 'a', newline='') as file:

        writer = csv.writer(file)

        writer.writerows(data)
```

This creates Comma Separated Values (CSV) files:

conDataset.csv

Fig 5.1

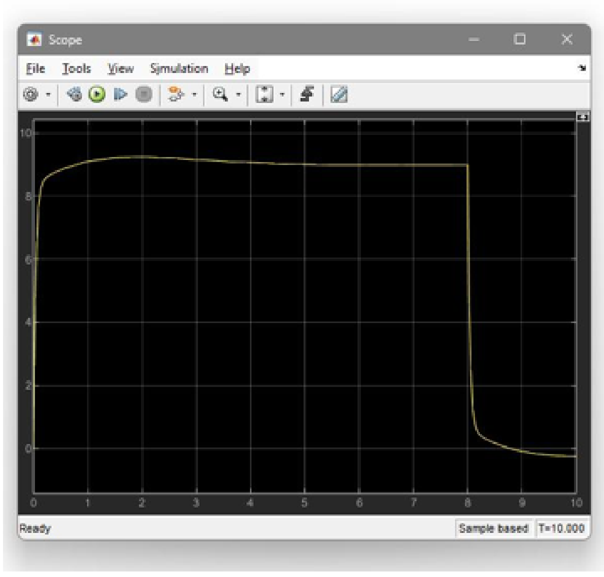
| Step time | Initial value | Final value | Sample time | Proportional | Integral | Derivative | Filter coefficient | Output |
|---|---|---|---|---|---|---|---|---|
| 9 | 6 | 6 | 0 | 8 | 1 | 0 | 8 | 1 |
| 9 | 7 | 2 | 0 | 5 | 5 | 9 | 0 | 0 |
| 7 | 7 | 7 | 0 | 3 | 8 | 1 | 7 | 1 |
| 6 | 9 | 0 | 3 | 2 | 4 | 3 | 1 | 0 |

Table 5.1: Continuous Dataset

These files contain all the changeable parameters of the step block and the pid controller block as columns. These columns are used as the independent variables in our ML models.

Meanwhile the screenshot function of the PyAutoGUI library is used to capture the screenshots of the graphs for various input combinations. The screenshots are then used to categorize the outputs, which acts as the dependent variable, into stable(0) and unstable(1) for the continuous dataset.

- **<u>Screenshots of Continuous Data</u>**



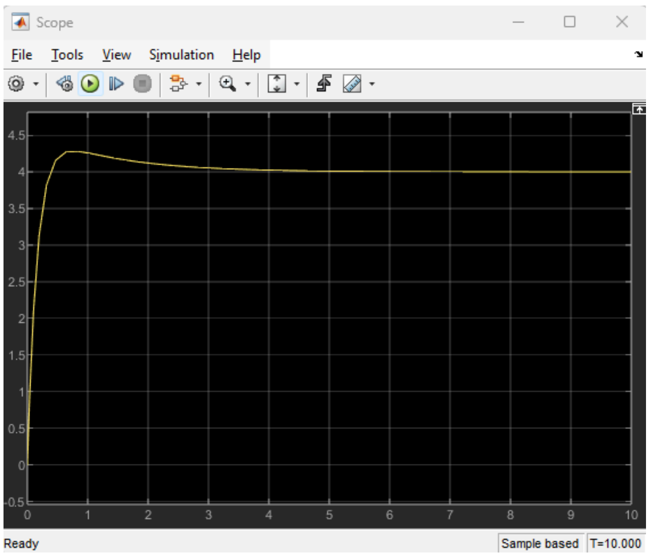
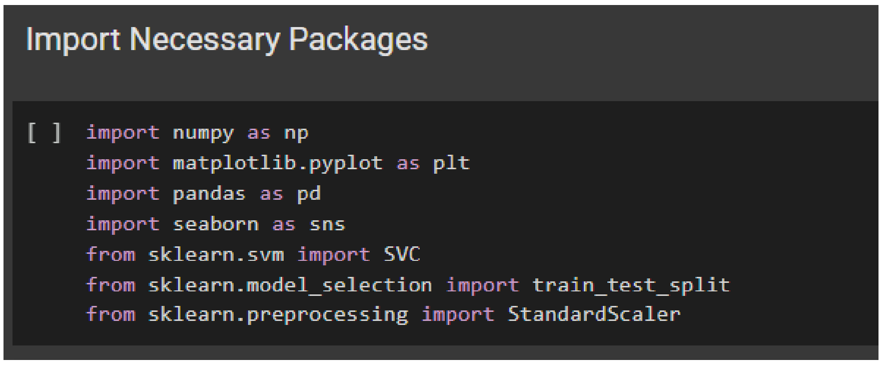


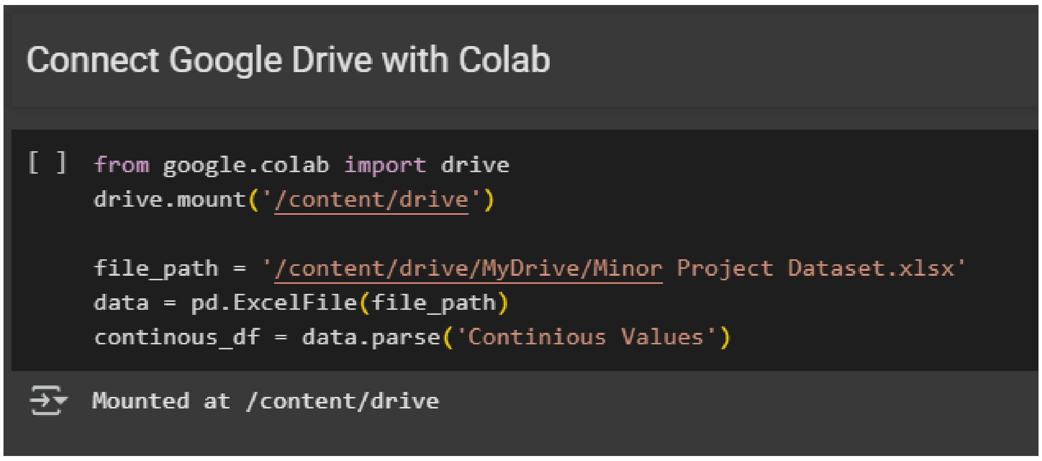
Fig 5.2 Fig 5.3

## 5.1. Program to implement the SVM Model



Import Necessary Packages

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    from sklearn.svm import SVC
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
```


Connect Google Drive with Colab

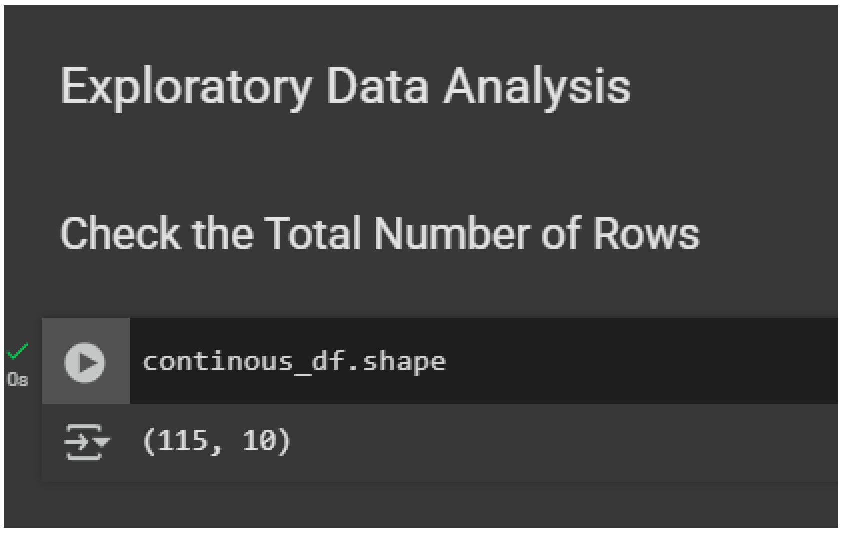
```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    file_path = '/content/drive/MyDrive/Minor Project Dataset.xlsx'
    data = pd.ExcelFile(file_path)
    continous_df = data.parse('Continious Values')
```

```
Mounted at /content/drive
```

### 5.1.1. Exploratory Data Analysis

Exploratory Data Analysis is performed on the data, mainly to find the number of total rows and number of non-null rows. From there a total number of null rows are found and they are deleted.

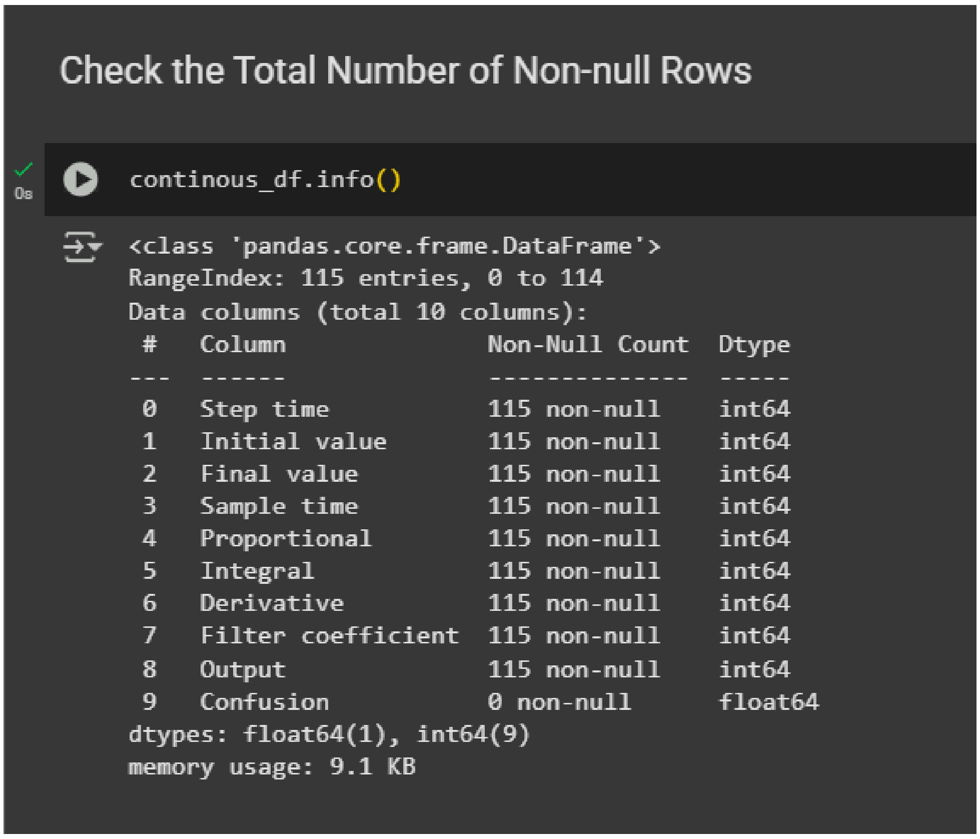
Upon creating a scatterplot of all the features with others (pairplot), it is found that the data is not linearly separable. Therefore it needs to be transformed into higher dimensions to create an optimal decision boundary to separate the two classes [4], [11], [14].


Exploratory Data Analysis

Check the Total Number of Rows

```
continous_df.shape
```

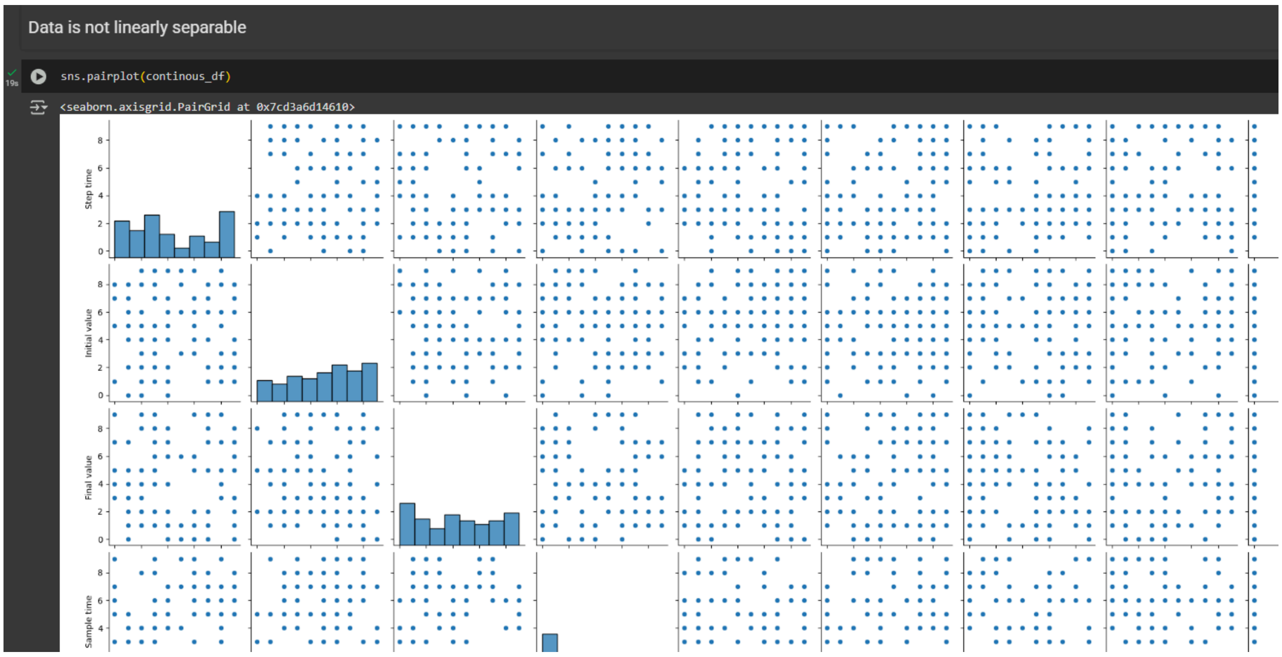
```
(115, 10)
```



Check the Total Number of Non-null Rows

```
continous_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Step time           115 non-null    int64
 1   Initial value       115 non-null    int64
 2   Final value         115 non-null    int64
 3   Sample time         115 non-null    int64
 4   Proportional        115 non-null    int64
 5   Integral            115 non-null    int64
 6   Derivative          115 non-null    int64
 7   Filter coefficient  115 non-null    int64
 8   Output              115 non-null    int64
 9   Confusion           0 non-null      float64
dtypes: float64(1), int64(9)
memory usage: 9.1 KB
```



Data is not linearly separable

```
sns.pairplot(continous_df)
```

```
<seaborn.axisgrid.PairGrid at 0x7cd3a6d14610>
```

Fig 5.4

### 5.1.2. Data Preprocessing

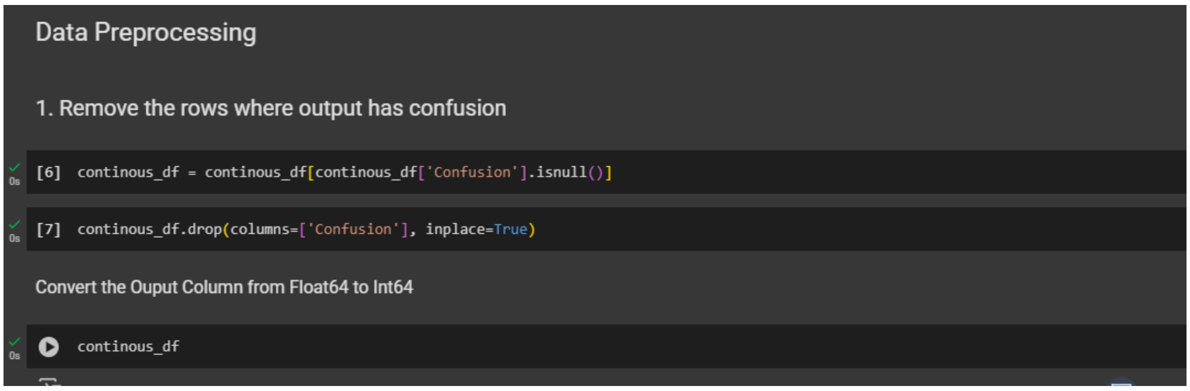
Some outputs were collected but there was confusion if the output was stable or unstable, these outputs were marked in the ‘Confusion’ column. Only keep those rows where the ‘Confusion’ column is null, i.e. there wasn’t any confusion about the output.

Also Delete the Confusion column, because it is of no use to the machine learning model and convert the ‘Output’ column from Float64 to Int64.

It is found that the dataset has a 60% - 40% imbalance. Therefore proper measures have to be applied [4], [12].

The independent features - Sample Time and Filter coefficient needs to be dropped as they are tuning parameters, not something the model could learn [1], [10].

The data is split into 75% - 25% Training and Testing sets, Z-Score standardization is applied and it is checked if the features have any correlation between them.


Data Preprocessing

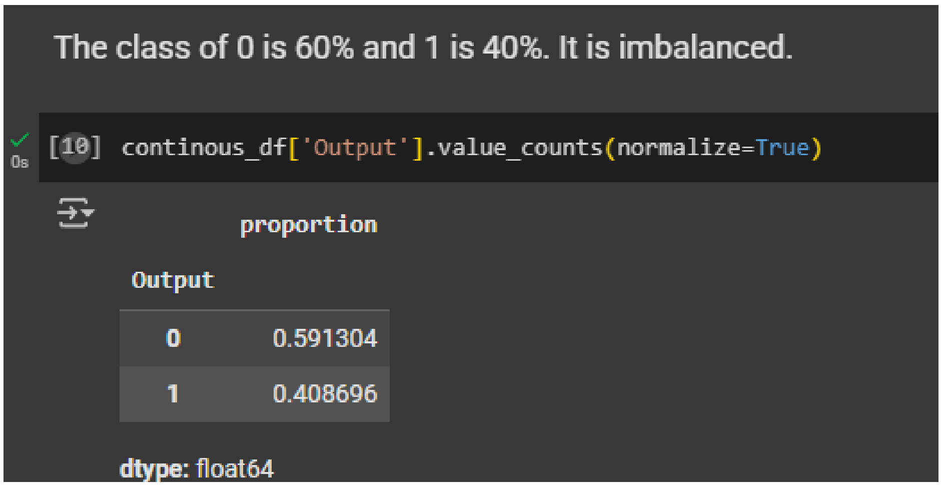
1. Remove the rows where output has confusion

```
[6] continous_df = continous_df[continous_df['Confusion'].isnull()]
```

```
[7] continous_df.drop(columns=['Confusion'], inplace=True)
```

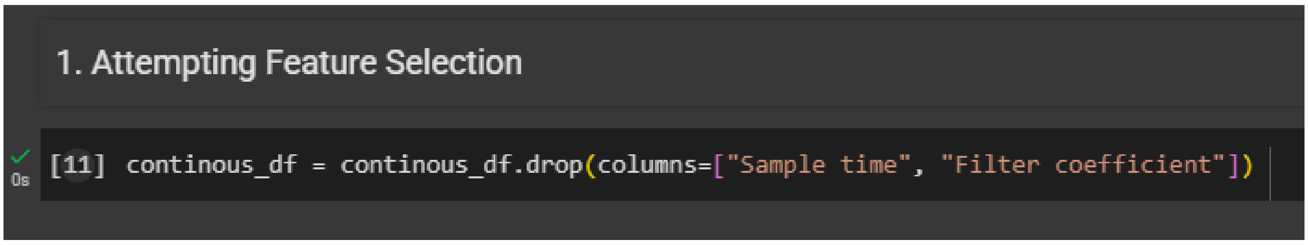
Convert the Ouput Column from Float64 to Int64

```
continous_df
```


The class of 0 is 60% and 1 is 40%. It is imbalanced.

```
[10] continous_df['Output'].value_counts(normalize=True)
```

```
        proportion
Output
0         0.591304
1         0.408696

dtype: float64
```

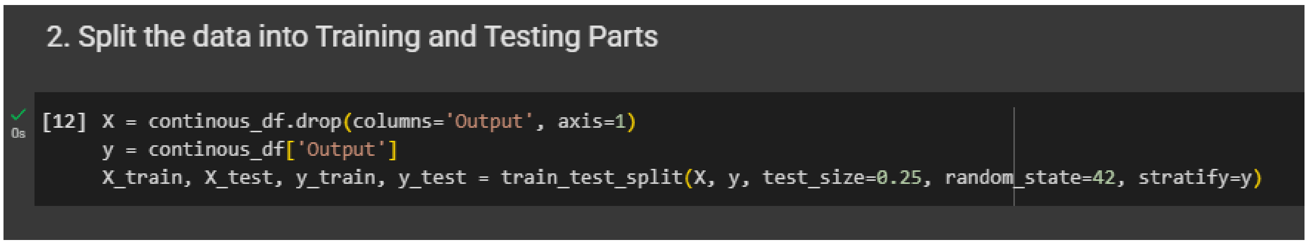


1. Attempting Feature Selection

```
[11] continous_df = continous_df.drop(columns=["Sample time", "Filter coefficient"])
```

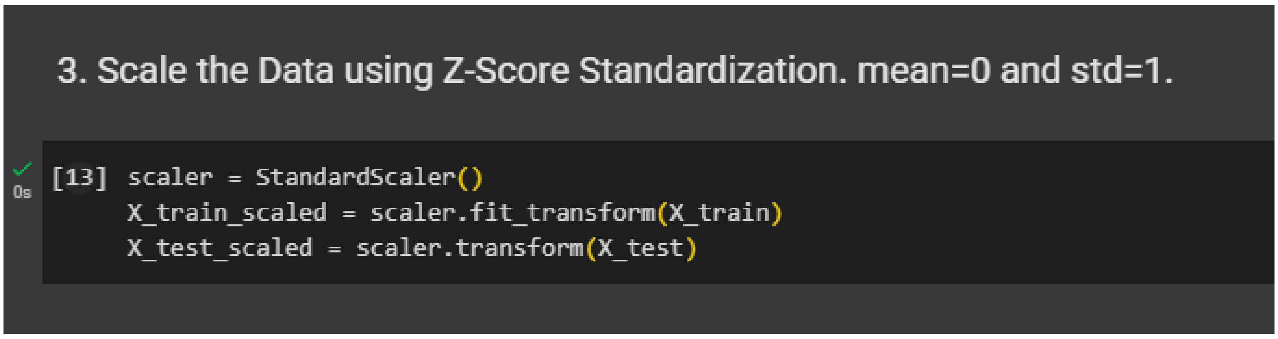
| Initial value | Final value | Sample time | Proportional | Integral | Derivative | Output |
|---|---|---|---|---|---|---|
| 6 | 6 | 0 | 8 | 1 | 0 | 1 |
| 7 | 2 | 0 | 5 | 5 | 9 | 0 |
| 7 | 7 | 0 | 3 | 8 | 1 | 1 |
| 9 | 0 | 3 | 2 | 4 | 3 | 0 |

Table 5.2: Dataset after Feature Selection



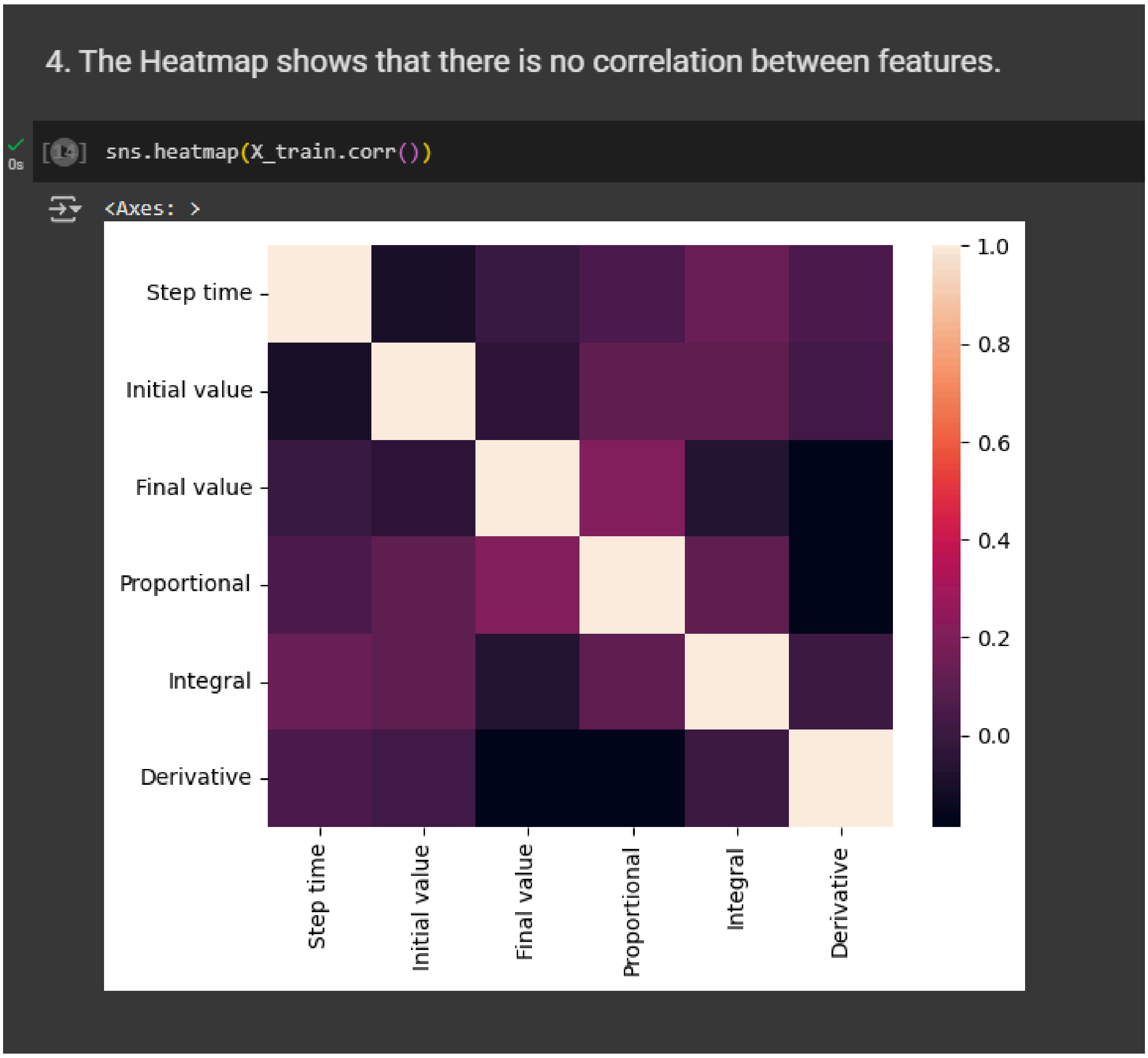
2. Split the data into Training and Testing Parts

```
[12] X = continous_df.drop(columns='Output', axis=1)
     y = continous_df['Output']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, stratify=y)
```



3. Scale the Data using Z-Score Standardization. mean=0 and std=1.

```
[13] scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```



4. The Heatmap shows that there is no correlation between features.

```
[14] sns.heatmap(X_train.corr())
```

```
<Axes: >
```

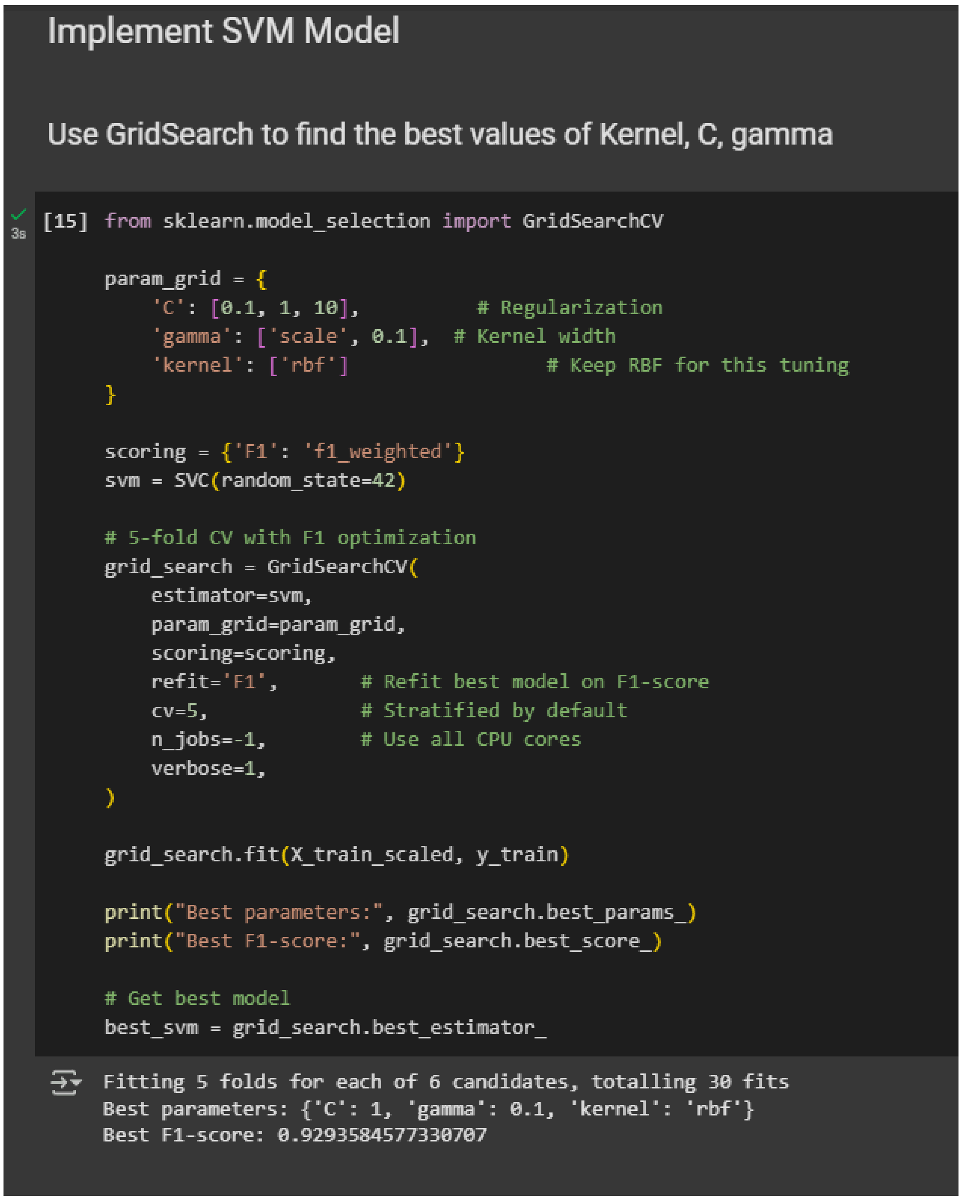
Fig 5.5: Heatmap

### 5.1.3. Training the Machine Learning Model

A Classification machine learning algorithm is applied to classify system outputs as either stable or unstable. The models employed are Support Vector Machine(SVM) [4], [7], [11].

- Support Vector Machine (SVM): For identifying optimal hyperplanes to classify stability conditions.
- GridSearch to find the best values of Kernel, C, gamma

The dataset, where all the independent variables can be any positive integers and the dependent variable is either 0 or 1. 0 represents unstable and 1 represents stable output. A Support Vector Machine is applied with a Balanced Bagging Classifier to rectify the problem of imbalanced dataset [12], [4].

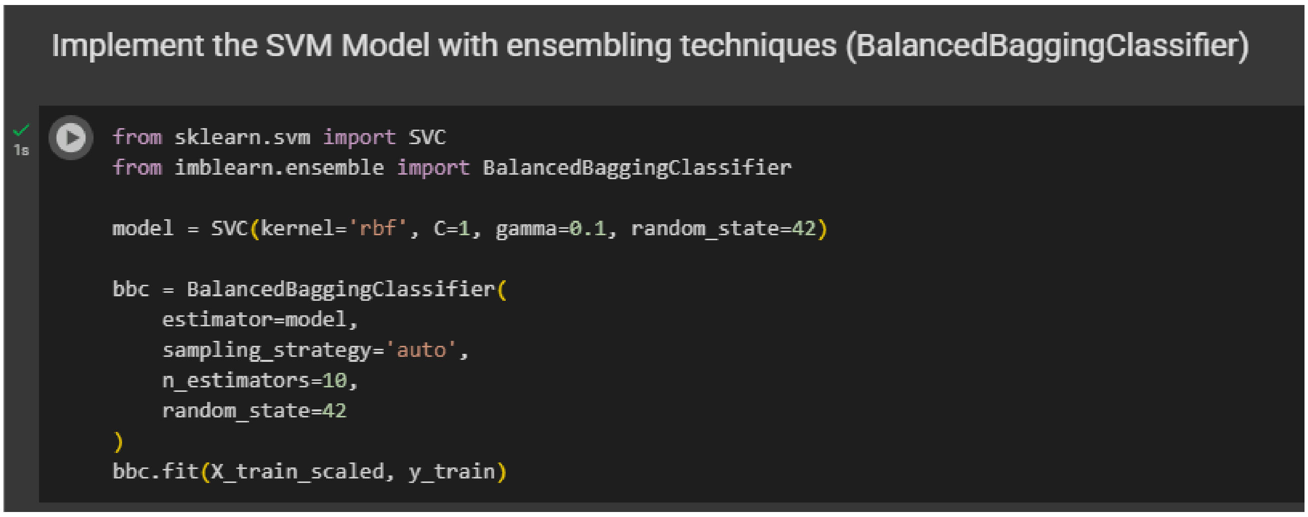


Implement SVM Model

Use GridSearch to find the best values of Kernel, C, gamma

```
[15] from sklearn.model_selection import GridSearchCV

     param_grid = {
         'C': [0.1, 1, 10],          # Regularization
         'gamma': ['scale', 0.1],  # Kernel width
         'kernel': ['rbf']                   # Keep RBF for this tuning
     }

     scoring = {'F1': 'f1_weighted'}
     svm = SVC(random_state=42)

     # 5-fold CV with F1 optimization
     grid_search = GridSearchCV(
         estimator=svm,
         param_grid=param_grid,
         scoring=scoring,
         refit='F1',          # Refit best model on F1-score
         cv=5,                # Stratified by default
         n_jobs=-1,           # Use all CPU cores
         verbose=1,
     )

     grid_search.fit(X_train_scaled, y_train)

     print("Best parameters:", grid_search.best_params_)
     print("Best F1-score:", grid_search.best_score_)

     # Get best model
     best_svm = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Best F1-score: 0.9293584577330707
```



## Implement the SVM Model with ensembling techniques (BalancedBaggingClassifier)

1s

```python
from sklearn.svm import SVC
from imblearn.ensemble import BalancedBaggingClassifier

model = SVC(kernel='rbf', C=1, gamma=0.1, random_state=42)

bbc = BalancedBaggingClassifier(
    estimator=model,
    sampling_strategy='auto',
    n_estimators=10,
    random_state=42
)
bbc.fit(X_train_scaled, y_train)
```

### 5.1.4. Testing Accuracy of the Machine Learning Model

Accuracy is the ratio of correct predictions to the total number of predictions.

**Accuracy = TP + TN / FP + FN + TP + TN**

Accuracy can be misleading when data is imbalanced.

Precision - What fraction of predicted positives are actually correct i.e. "How many selected items are correct?"

**High precision = Few false positives**.

**Low precision = Many false alarms.**

**Precision = TP / FP + TP**

Recall - What fraction of actual positives did the model catch i.e. "How many correct items were selected?"

**High recall = Few false negatives.**

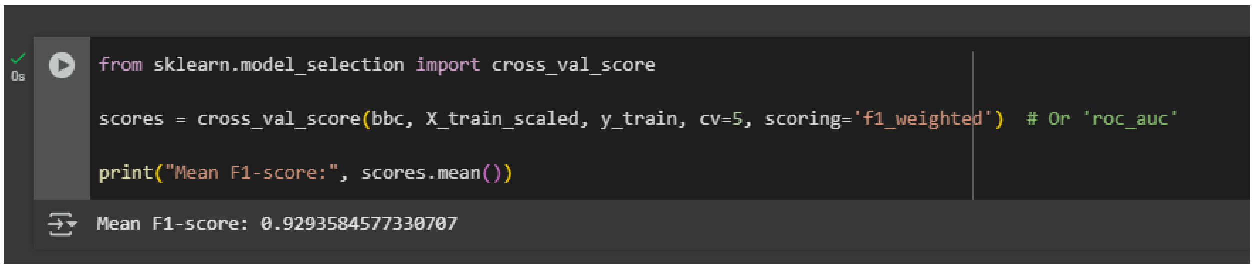
**Low recall = Missed many true positives.**

**Recall =TP / FN + TP**

**F1 Score** = Harmonic mean of precision and recall (balances both)

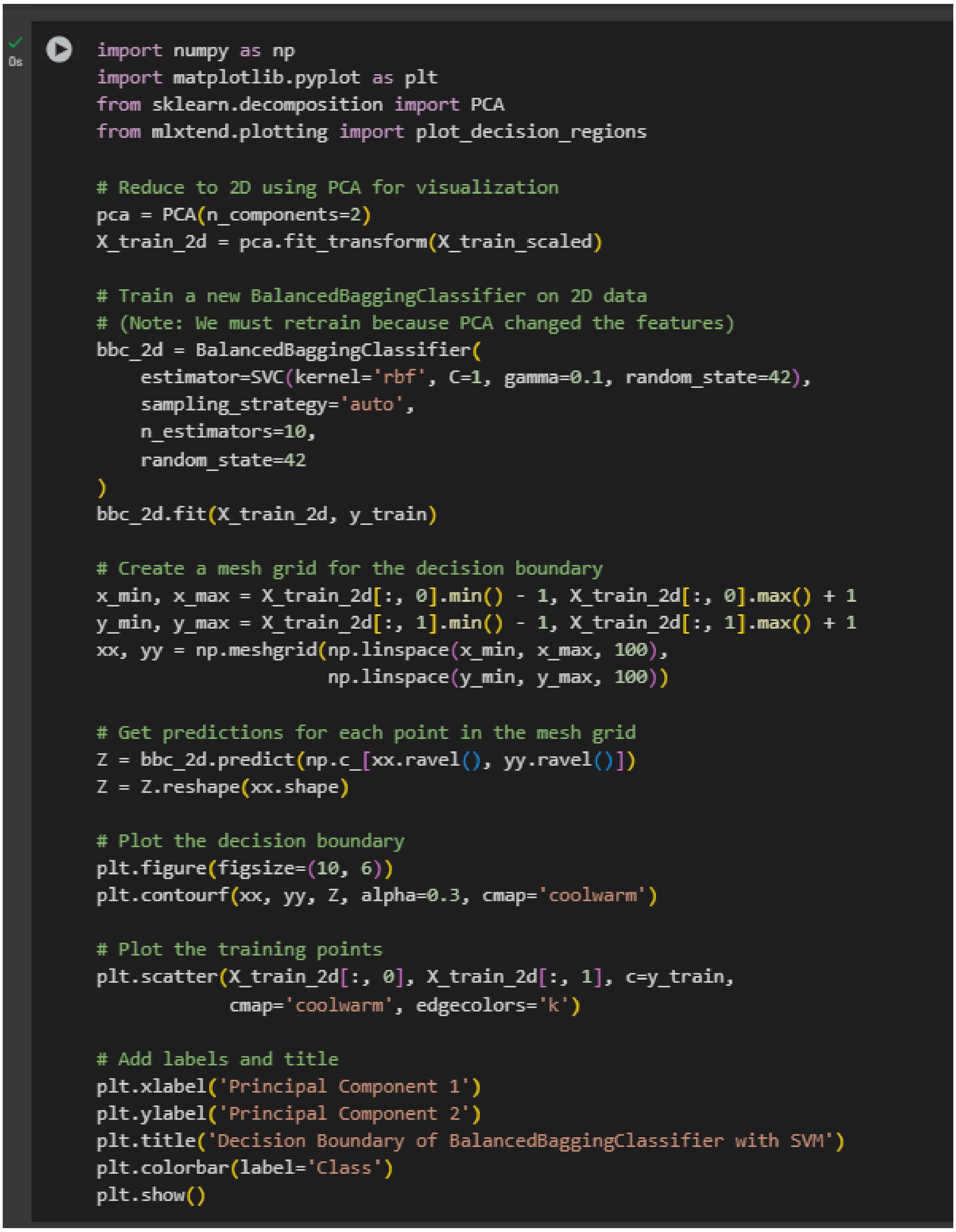
**Support** = Number of actual samples per class in the test set

K Fold Cross Validation is applied (CV = 5), to evaluate the model's performance more robustly. The model is trained on 4 parts and tested on the remaining 1. This process is repeated 5 times with a different test set each time. This reduces the risk of overfitting to a specific train-test-split [4], [13].

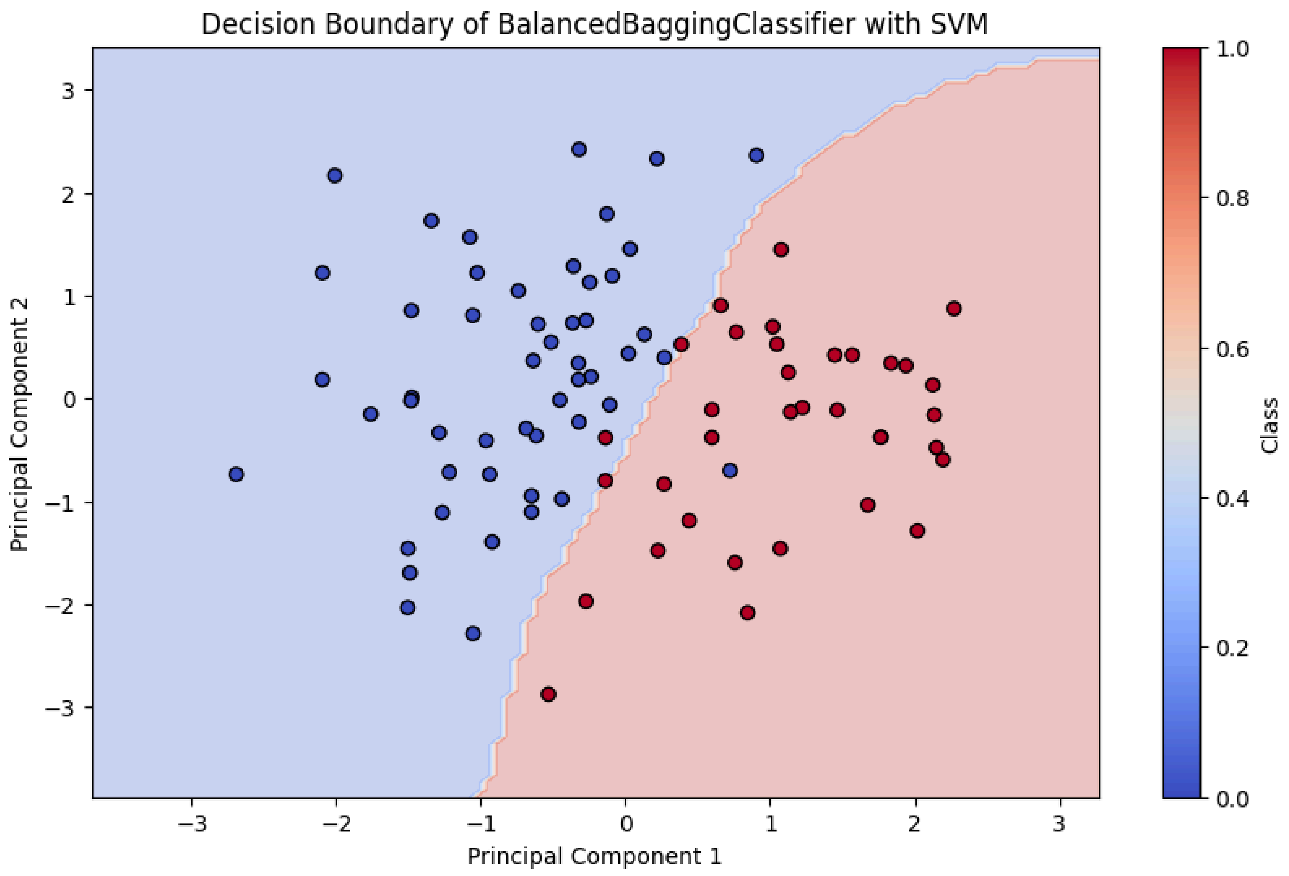


```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(bbc, X_train_scaled, y_train, cv=5, scoring='f1_weighted')  # Or 'roc_auc'

print("Mean F1-score:", scores.mean())
```

```
Mean F1-score: 0.9293584577330707
```



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions

# Reduce to 2D using PCA for visualization
pca = PCA(n_components=2)
X_train_2d = pca.fit_transform(X_train_scaled)

# Train a new BalancedBaggingClassifier on 2D data
# (Note: We must retrain because PCA changed the features)
bbc_2d = BalancedBaggingClassifier(
    estimator=SVC(kernel='rbf', C=1, gamma=0.1, random_state=42),
    sampling_strategy='auto',
    n_estimators=10,
    random_state=42
)
bbc_2d.fit(X_train_2d, y_train)

# Create a mesh grid for the decision boundary
x_min, x_max = X_train_2d[:, 0].min() - 1, X_train_2d[:, 0].max() + 1
y_min, y_max = X_train_2d[:, 1].min() - 1, X_train_2d[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                     np.linspace(y_min, y_max, 100))

# Get predictions for each point in the mesh grid
Z = bbc_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary
plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z, alpha=0.3, cmap='coolwarm')

# Plot the training points
plt.scatter(X_train_2d[:, 0], X_train_2d[:, 1], c=y_train,
            cmap='coolwarm', edgecolors='k')

# Add labels and title
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Decision Boundary of BalancedBaggingClassifier with SVM')
plt.colorbar(label='Class')
plt.show()
```



Decision Boundary of BalancedBaggingClassifier with SVM
Principal Component 1
Principal Component 2
Class
−3
−2
−1
0
1
2
3
0.0
0.2
0.4
0.6
0.8
1.0

Fig 5.6

## 5.2. Reliability and Stability Analysis of the System

The reliability and stability analysis focuses on assessing the robustness of the control system under varying conditions. Key aspects include:

### 5.2.1. Results

The implementation of the Support Vector Machine (SVM) model to classify stable and unstable outputs based on the input parameters of the continuous data closed-loop PID-controlled system yielded promising results [4], [7].

**SVM**

The SVM model with BalancedBaggingClassifier was trained on the dataset containing both stable and unstable outputs.

The model achieved an accuracy of 93.10% on the training dataset.

As our dataset is imbalanced, F1 Score is a better metric than accuracy. The model achieves a score of 94% for unstable outputs and 92% stable outputs.

This demonstrates that the model was moderately successful in capturing the underlying patterns in the data [4], [11].

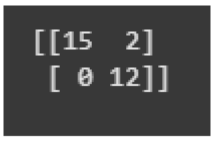
**The confusion Matrix shows that**

True Positive = 15

False Positive = 0

False Negative  = 2

True Negative = 12



```
[[15  2]
 [ 0 12]]
```

This improvement indicates that the model generalized well to real-world conditions and was capable of identifying stable and unstable outputs effectively [4], [7].

### 5.2.2. Model Interpretation

The SVM model separates the data points into stable and unstable regions based on a hyperplane.

Inputs closer to the decision boundary are more challenging to classify, which could contribute to errors in the training dataset.

The model relied on features such as Step Time, Initial Value, Final Value, and PID parameters (Proportional, Integral, Derivative) to make predictions.

Analyzing the weights or support vectors in the SVM model can provide insights into which features were most influential in determining stability [1], [4], [7].

## 5.3. Discussions

### 5.3.1. Merits of the SVM Model

Robust Generalization: The SVM model demonstrated the ability to generalize well to unseen, real-world data, which is crucial for predictive frameworks in dynamic systems like PID controllers.

Handling Non-linearity: By employing a kernel trick (e.g., radial basis function kernel), the model effectively handles non-linear decision boundaries [4], [14] in the dataset.

### 5.3.2. Demerits of the SVM Model

SVMs work well when the dataset has many features (High Dimensional Data).

SVMs don’t scale well to very large datasets (thousands of samples), but work well on smaller datasets with complex boundaries.

# 6. Chapter 6: Conclusion, Future Scope, Limitations.

This study presented a novel integration of classical control systems and advanced machine learning techniques to analyze and predict stability margins in PID-controlled systems [1], [3], [4]. By leveraging MATLAB Simulink to simulate a closed-loop system, input parameters such as Step Time, Initial Value, Final Value, and PID Controller Settings (Proportional, Integral, and Derivative) were systematically varied to generate outputs categorized as stable or unstable [1], [6], [10]. Machine learning models, including Support Vector Machines (SVM), were employed to predict stability based on these inputs.

The Support Vector Machine (SVM) model emerged as a robust predictive framework, achieving an accuracy of 93.10%. This result shows the potential of SVM in generalizing to unseen scenarios, particularly in dynamic systems where precise stability prediction is critical [4], [7]. While the accuracy achieved reflects a significant step forward, the findings also highlight opportunities for further optimization and enhancement through advanced feature engineering, ensemble learning, and hyperparameter tuning.

## 6.1. Advantages of using SVM:

SVMs work well with **many features** or high dimensional data.

SVMs don’t scale well to **very large datasets. They work well on smaller datasets with complex boundaries.**

When data is **not linearly separable**, you can use kernel tricks (like RBF, polynomial) to map it into a higher-dimensional space where it *is* separable.

**The project demonstrated several key achievements:**

Machine Learning Integration: The successful application of machine learning models, particularly SVM, validated their effectiveness in identifying stable and unstable outputs.

Real-World Testing: The improved accuracy of the SVM model on real-world data validated its reliability and practical applicability.

However, the study also identified challenges such as sensitivity to outliers, computational complexity, and the limitations of single models in handling highly nonlinear systems [4], [13], [20]. Addressing these limitations through hybrid models or additional computational techniques can be a promising avenue for future research.

## 6.2. Broader Implications

This project demonstrates the transformative potential of combining machine learning and control theory in addressing stability challenges [1], [3], [8]. The predictive framework developed here has applications not only in PID-controlled systems but also in broader engineering contexts, such as industrial automation, robotics, and aerospace systems. By enabling real-time prediction and intervention, such frameworks can significantly enhance system performance, reliability, and safety.

In conclusion, this project provides a foundation for future exploration of machine learning-driven control systems. It paves the way for innovative solutions that seamlessly blend computational intelligence with classical engineering [3], [4], [11], driving progress in predictive analysis and automated system optimization.

## 6.3. Limitations

The model fails to correctly predict False Negatives sometimes.

Sensitivity to Outliers: SVM is known to be sensitive to outliers, which could have affected training performance [4], [11].

Computational Cost: Training the SVM model on large datasets with non-linear kernels can be computationally expensive [4], [14].

Every control system's environmental factors are different and they would have separate training processes.

This method doesn’t apply to distributed systems

## 6.4. Future Improvements

Feature Engineering: Additional features or transformations of existing features may enhance the model's ability to discern stability patterns [4], [11].

Error Analysis: Errors by the different machine learning models shall be analyzed to understand which model has better accuracy and their implications on system performance [13], [20].

This method doesn’t apply to distributed systems. In the future it will be expanded to include distributed systems [15], [16] as well.

# 7. Chapter 7: References.

[1] Liu, Ning, Tianyou Chai, Yajun Zhang, and Weinan Gao. 2025. “Data-Driven Optimal Tuning of PID Controller Parameters.” Science China Information Sciences 68.

[2] Smith, J. 2015. “Advanced PID Control Techniques.” Journal of Process Control 25 (4): 140–150.

[3] Park, S., and K. Kim. 1999. “A Survey on PID Control and Algorithms.” Information Sciences 120 (2): 251–274.

[4] Zhang, Y., and X. Li. 2021. “PID Controller Design Based on Machine Learning Algorithms.” Journal of Ambient Intelligence and Humanized Computing 12 (3): 325–340.

[5] Chang, C. 2002. “A PID Control Algorithm for Industrial Applications.” Proceedings of the IEEE International Conference on Robotics and Automation 5 (1): 504–509.

[6] Ackermann, J. 1984. “Robust PID Control of Linear Systems.” International Journal of Control 39 (5): 1281–1296.

[7] Günther, Johannes, Elias Reichensdörfer, Patrick M. Pilarski, and Klaus Diepold. 2020. “Interpretable PID Parameter Tuning for Control Engineering Using General Dynamic Neural Networks: An Extensive Comparison.”

[8] Wang, L. 2005. “Robust PID Control for Uncertain Systems.” Proceedings of the IEEE Conference on Decision and Control 7 (1): 310–315.

[9] Chen, J. 2004. “Intelligent PID Control for Complex Systems.” Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 8 (4): 150–155.

[10] Sharma, A., and R. Singh. 2021. “PID Controller Optimization Using Genetic Algorithms.” Measurement 58 (3): 45–60.

[11] Kim, Y. 2006. “Neural Network-Based PID Controller for Nonlinear Systems.” IEEE Transactions on Neural Networks 17 (5): 49–54.

[12] Patel, R. 2015. “Advanced Tuning Methods for PID Controllers.” Procedia Engineering 130 (4): 1500–1507.

[13] Park, J. 1993. “Robust PID Design for Time-Delay Systems.” Proceedings of the American Control Conference 3 (2): 180–185.

[14] Miller, R., and P. Jones. 1994. “New Approach to PID Control Tuning.” Automatica 30 (12): 1919–1925.

[15] Giri, Arnab, Nilava Debabhuti, Prolay Sharma, Sudipta Mondal, Subhashis Das, and Pranibesh Mandal. 2021. “PID Control Parameter Tuning Using Linear Multivariate Model.” In Data Management, Analytics and Innovation, 387–398. Springer, Singapore.

[16] La Cava, M., G. Paletta, and C. Picardi. 2007. “Stability Analysis of PWM Feedback Control Systems with PID Regulators.” International Journal of Circuit Theory and Applications 35 (6): 693–703.

[17] Ang, Kiam Heong, G. Chong, and Yun Li. 2005. “PID Control System Analysis, Design, and Technology.” IEEE Transactions on Control Systems Technology 13 (4): 559–576.

[18] Li, Yun, Kiam Heong Ang, and G. C. Y. Chong. 2006. “PID Control System Analysis and Design.” IEEE Control Systems Magazine 26 (1): 32–41.