ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

SEMI-HEURISTIC OPTIMIZATION TECHNIQUES PERFORMANCE IN ROBUST CONTROL SYSTEMS DESIGN

M.Sc. THESIS

Mohammed ELBADRI

Department of Control and Automation Engineering

Control and Automation Engineering Program

JULY 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

SEMI-HEURISTIC OPTIMIZATION TECHNIQUES PERFORMANCE IN ROBUST CONTROL SYSTEMS DESIGN

M.Sc. THESIS

Mohammed ELBADRI
(504221113)

Department of Control and Automation Engineering

Control and Automation Engineering Program

Thesis Advisor: Assoc. Prof. Dr. İlker ÜSTOĞLU
Eş Danışman: Asst. Prof. Dr. İsmail BAYEZİT

JULY 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DAYANIKLI KONTROL SİSTEMLERİ TASARIMINDA YARI-BULUŞSAL OPTİMİZASYON TEKNİKLERİ PERFORMANSI

YÜKSEK LİSANS TEZİ

Mohammed ELBADRI
(504221113)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. İlker ÜSTOĞLU
Eş Danışman: Asst. Prof. Dr. İsmail BAYEZİT

TEMMUZ 2025

Mohammed ELBADRI, a M.Sc. student of ITU Graduate School student ID 504221113 successfully defended the thesis entitled "SEMI-HEURISTIC OPTIMIZATION TECHNIQUES PERFORMANCE IN ROBUST CONTROL SYSTEMS DESIGN", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. İlker ÜSTOĞLU** Istanbul Technical University	.....
Co-advisor :	**Asst. Prof. Dr. İsmail BAYEZİT** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Ali TATAR** Istanbul Technical University	.....
	Assoc. Prof. Dr. Muzaffer METİN Yıldız Technical University	.....
	Assoc. Prof. Dr. Levent UCUN Yıldız Technical University	.....

Date of Submission : **30 May 2025**
Date of Defense : **11 July 2025**

To my twin sister,

FOREWORD

This thesis is presented in partial fulfillment of the requirements for obtaining a Master of Science degree in Control and Automation Engineering from Istanbul Technical University.

The research described in this document was carried out from early June 2023 to January 2025 in the Model-based Design and Control laboratory at Istanbul Technical University, under the mentorship of Assis. Prof. Dr. İsmail Bayezit.

Throughout this research experience, I have benefited from the support, guidance, and encouragement of numerous individuals. I am truly grateful to my supervisor, Assoc. Prof. Dr. İlker Üstoğlu, and my co-supervisor, Assis. Prof. Dr. İsmail Bayezit, whose knowledge and insights were extremely valuable. I also appreciate my colleagues at the Model-based Design and Control laboratory for their teamwork and camaraderie.

I would like to extend special gratitude to my family, whose steadfast faith in me made this achievement possible.

JULY/2025

Mohammed ELBADRI
Validation Engineer

TABLE OF CONTENTS

ABBREVIATIONS

AAS : Annealing Adaptive Search.
Adam : Adaptive Moment Estimation.
AI : Artificial Intelligence.
ANN : Artificial Neural Networks.
ANFIS : Adaptive Neuro-Fuzzy Inference System.
AT : Auto-Tuned.
CCF : Controllable Canonical Form.
CRB : Complex Root Boundary.
EOM : Equations of Motion.
FLC : Fuzzy Logic Control.
GA : Genetic Algorithms.
GD : Gradient Descent.
IAE : Integral of Absolute Error.
IQC : Integral Quadratic Constrain.
IRB : Infinite Root Boundary.
ISE : Integral of Squared Error.
ITAE : Integral of Time-multiplied Absolute Error.
ITSE : Integral of Time-multiplied Squared Error.
LFT : Linear Fractional Transformation.
LQR : Linear Quadratic Regulator.
LQT : Linear Quadratic Tracker.
MIMO : Multi-Input Multi Output.
MPC : Model Predictive Control.
NCS : Network Control Systems.
NN : Neural Network.
PAL : Paraconsistent Annotated Logic.
PAS : Pure Adaptive Search.
PD : Proportional-Derivative Controller.
PID : Proportional-Integral-Derivative Controller.
PPO : Proximal Policy Optimization (Reinforcement Learning).
PRS : Pure Random Search.
PSO : Particle Swarm Optimization.
RL : Reinforcement Learning.
RRB : Real Root Boundary.
SAF-PID : Self Adaptive PID.
SA : Simulated Annealing.
SGD : Stochastic Gradient Descent.
SMC : Sliding Mode Control.

SYMBOLS

$A(q), B(q), C(q)$	: State-Space Matrices with Parameter Uncertainties q.
A_c	: Closed-Loop System State Matrix.
β	: Desired Performance Level.
B_p, B_r	: Viscous Damping Coefficients.
c_1, c_2	: Acceleration Coefficients In PSO.
D_e, D_o, N_e, N_o	: Even/Odd Numerator/Denominator Polynomials.
ε	: Gradient Length.
γ	: Discount Factor In Reinforcement Learning.
$\hat{m}_t, \hat{v}_t$	: Bias-Corrected Estimates Of Gradients.
$J(\theta)$	: Expected Return In Reinforcement Learning.
$J(q, K)$	: Cost Function Associated With Controller K Under Uncertainty q.
$J_{\mathbf{IAE}}, J_{\mathbf{ISE}}, J_{\mathbf{ITAE}}, J_{\mathbf{ITSE}}$	: Integral Performance Error Functions.
J_p, J_r	: Pendulum And Rotary Arm Moments Of Inertia.
k_m	: Back-Emf Constant.
K, K_c, K_p, K_i, K_d	: Controller Gains Including PID Parameters.
$\mathscr{K}, \mathscr{K}_{RS}$	: Sets Of Stabilizing And Robust Controllers.
L_p, L_r	: Pendulum And Rotary Arm Lengths.
m_p, m_r	: Pendulum And Rotary Arm Masses.
m_t, v_t	: Past Gradient And Squared Gradient.
$M(s, q)$	: Transfer Function Matrix With Uncertainty Presence.
N	: Number Of Samples In Monte Carlo Methods.
$\nabla, \nabla f(x_k)$	: Gradient And Gradient Of The Objective Function.
$O(n^2)$	: Algorithm Runtime.
ω, ω^*	: Frequency, Singular Frequency.
P	: Positive Symmetric Matrix.
$P^{++}, P^{+-}, P^{-+}, P^{--}$	: Kharitonov Polynomials.
$P_c(s)$	: Closed Loop Polynomial.
p_i	: Personal Best Position Of Particle i In PSO.
Q, q, q_i	: Uncertainty Parameter Box And Variables.
r_1, r_2	: Random Factors In PSO.
r_t	: Reward At Time Step t.
R_m	: The Motor Terminal Resistor.
$\rho, \rho(K)$	: Quadratic Cost And Cost Of A Controller K.
T, V	: Kinetic Energy, Potential Energy
τ	: Rotary Arm Motor Torque.
θ, α	: Rotary Inverted Pendulum Angles.
u	: System Input Signal.
U_0	: Number Of Unstable Poles.
v_i, x_i	: Particle Velocity and Position.
V_m	: Motor Voltage Input.
x_k	: System Parameter At Iteration k (Gradient Descent).
$x_{\mathbf{high}}, x_{\mathbf{low}}, x_{\mathbf{mid}}$	: Interval Bounds And Midpoint In The Bisection Method.

LIST OF TABLES

Page

LIST OF FIGURES

SEMI-HEURISTIC OPTIMIZATION TECHNIQUES PERFORMANCE IN ROBUST CONTROL SYSTEMS DESIGN

SUMMARY

Designing a robust controller for systems with parameter uncertainties is a complex and demanding task since they are prone to changes regardless of the external impact forced upon these systems. This change might not only yield poor controller performance, but it can also lead to instability based on the change magnitude. Traditional deterministic control approaches may not provide a convenient solution due to combined computational complexity, and probabilistic approaches may also fall short in providing efficient and satisfactory solutions due to their time-intensive behavior and not guarantee of convergence.

To address this challenge, we propose a semi-heuristic approach that works as a mild algorithm, leveraging the advantages of both deterministic and heuristic approaches and overcoming the drawbacks obtained from these two types of control strategies. The proposed semi-heuristic approach exploits the Kharitonov theorem to establish an initial stable controller using system nominal values. Then this controller is used as a starting point for the gradient descent algorithm. The random search technique propagates based on cost function/s implemented for the whole of the uncertainty region. The iterative optimization process by gradient descent incorporates user-defined performance criteria, our approach provides a robust controller with respect to the presence of uncertainties for systems with interval polynomial characteristic equations.

We further enhanced the semi-heuristic approach by utilizing a more effective random optimization technique known as the Adaptive Moment Estimation (Adam) optimizer. We applied this proposal to the rotary inverted pendulum system, which is a well-known nonlinear and unstable system in control theory. We managed to demonstrate the efficiency of the semi-heuristic approaches in stabilizing a linearized rotary inverted pendulum model with a parametric approach used for uncertainty representation.

Extensive numerical simulations done on the proposed semi-heuristic approaches and the designed model validate the effectiveness of our proposed algorithms for both of the systems considered during this thesis study. Consequently, the semi-heuristic approaches emerge as a moderate solution to our initial problem introduced by the deterministic and stochastic approaches. We further compared the controller results with common control strategies in state-of-the-art and proved their superiority in addressing robust control problems with uncertain parameters.

DAYANIKLI KONTROL SİSTEMLERİ TASARIMINDA YARI-BULUŞSAL OPTİMİZASYON TEKNİKLERİ PERFORMANSI

ÖZET

Parametre belirsizlikleri olan sistemler için sağlam bir denetleyici tasarlamak, bu sistemlere uygulanan dış etkilerden bağımsız olarak değişikliklere eğilimli oldukları için karmaşık ve zorlu bir görevdir. Bu değişiklik yalnızca zayıf denetleyici performansına yol açmakla kalmayıp, aynı zamanda değişiklik büyüklüğüne bağlı olarak istikrarsızlığa da yol açabilir. Geleneksel deterministik kontrol yaklaşımları, birleşik hesaplama karmaşıklığı nedeniyle uygun bir çözüm sağlamayabilir ve olasılıksal yaklaşımlar da zaman alıcı davranışları ve yakınsamayı garantilememeleri nedeniyle verimli ve tatmin edici çözümler sağlamada yetersiz kalabilir. Parametre belirsizliği içeren dinamik sistemlerde kararlı ve yüksek performanslı kontrolör tasarımı, kontrol mühendisliğinde çözülmesi güç bir problem olarak karşımıza çıkmaktadır. Sistem parametrelerinin dışsal etkenlere bağlı olmaksızın zamanla değişmesi, klasik kontrolör yapılarını yetersiz kılmakta ve sistemin kararsızlığa sürüklenmesine sebep olabilmektedir. Bu belirsizliklerin göz ardı edilmesi, sistem yanıtında kararsız salınımlar, aşırı yükselmeler veya performans kayıpları gibi istenmeyen durumlara yol açabilir. Bu nedenle, kontrol sistemlerinin tasarım aşamasında bu belirsizliklerin dikkate alınması, hem kararlılık hem de performans açısından büyük önem arz etmektedir.

Bu zorluğun üstesinden gelmek için, hem deterministik hem de sezgisel yaklaşımların avantajlarından yararlanan ve bu iki tür kontrol stratejisinden elde edilen dezavantajların üstesinden gelen hafif bir algoritma olarak çalışan yarı sezgisel bir yaklaşım öneriyoruz. Önerilen yarı sezgisel yaklaşım, sistem nominal değerlerini kullanarak başlangıçta kararlı bir denetleyici oluşturmak için Kharitonov teoremini kullanır. Daha sonra bu denetleyici, gradyan iniş algoritması için bir başlangıç noktası olarak kullanılır. Rastgele arama tekniği, belirsizlik bölgesinin tamamı için uygulanan maliyet fonksiyonuna göre yayılır. Gradyan inişle yinelemeli optimizasyon süreci kullanıcı tanımlı performans kriterlerini içerir; bu sayede önerilen yaklaşım, aralık polinom karakteristik denklemlerine sahip sistemler için belirsizliklerin varlığına karşı sağlam bir denetleyici sunar.

Uyarlanabilir Moment Tahmini (Adam) optimizasyonu olarak bilinen daha etkili bir rastgele optimizasyon tekniği sunarak yarı sezgisel yaklaşımı daha da geliştirdik. Bu öneriyi kontrol teorisinde iyi bilinen doğrusal olmayan ve kararsız bir sistem olan dönel ters sarkaç sistemine uyguladık. Belirsizlik gösterimi için kullanılan parametrik bir yaklaşımla doğrusal hale getirilmiş dönel ters sarkaç modelini sabitlemede yarı sezgisel yaklaşımların verimliliğini göstermeyi başardık. Uygulanan bu yöntem, geleneksel tekniklerle karşılaştırıldığında daha düşük hata, daha kısa yerleşme süresi

ve daha istikrarlı bir kontrol davranışı sunmuştur.

Önerilen yarı sezgisel yaklaşımlar ve tasarlanan model üzerinde yapılan kapsamlı sayısal simülasyonlar, bu tez çalışması sırasında ele alınan her iki sistem için önerilen algoritmalarımızın etkinliğini doğrulamaktadır. Sonuç olarak, yarı sezgisel yaklaşımlar, deterministik ve stokastik yaklaşımlar tarafından ortaya konulan ilk problemimize orta düzeyde bir çözüm olarak ortaya çıkmaktadır. Kontrolör sonuçlarını, son teknolojideki yaygın kontrol stratejileriyle karşılaştırdık ve belirsiz parametrelere sahip sağlam kontrol problemlerini ele almadaki üstünlüklerini çeşitli metriklerle ortaya koyduk. Bu karşılaştırmalar, önerilen yöntemin hem zamansal hem de frekans alanındaki tepkiler açısından daha dengeli sonuçlar verdiğini göstermektedir.

Bu tez çalışmasında, deterministik ve sezgisel kontrol yaklaşımlarının güçlü yönlerini birleştiren yarı sezgisel bir kontrol yöntemi geliştirilmiştir. Geliştirilen yöntem, sistemin nominal modeline dayalı olarak deterministik bir başlangıç noktası elde etmekte, ardından bu başlangıç kontrolörü üzerinde rastgele arama temelli optimizasyon algoritmaları ile kararlılığı ve performansı geliştirmeye yönelik bir iyileştirme süreci yürütmektedir. Bu bağlamda, ilk aşamada Kharitonov Teoremi kullanılarak sistemin aralıklı karakteristik polinom yapısı üzerinden kararlılık sağlanmış, daha sonra bu yapı üzerinden gradyan inişi (Gradient Descent) ve uyarlamalı moment tahmini (Adam) algoritmaları kullanılarak parametre belirsizliği içeren geniş bir çalışma aralığında optimizasyon gerçekleştirilmiştir.

Tez kapsamında iki farklı kontrol problemine çözüm sunulmuştur. İlk uygulama, 5. dereceden bir aralıklı polinom sisteme uygulanmış olup PID kontrolör yapısı temel alınarak gerçekleştirilmiştir. Bu uygulamada, Munro-Soylemez yöntemiyle elde edilen kararlı kazanç aralıkları sayesinde başlangıç kontrolörünün hızlı bir şekilde belirlenmesi sağlanmıştır. Bu başlangıç kazançları, daha sonra kullanıcı tarafından tanımlanan maliyet fonksiyonlarına göre optimize edilmiş ve böylece belirsizlik bölgesinin tamamı için tatmin edici bir kontrolör elde edilmiştir. Bu süreçte kullanılan hata fonksiyonları (IAE, ISE, ITAE vb.) sistem yanıtının şekli üzerinde önemli rol oynamış ve elde edilen sonuçlar klasik otomatik ayarlı PID kontrolörlerle karşılaştırıldığında üstün performans göstermiştir.

İkinci uygulama ise kontrol teorisinde oldukça yaygın olarak kullanılan ve doğrusal olmayan yapısıyla bilinen dönel ters sarkaç (rotary inverted pendulum) sistemi üzerine gerçekleştirilmiştir. Bu sistemin doğrusal hale getirilmiş hali üzerinden yapılan çalışmada, çift-PD kontrolör mimarisi uygulanmıştır. İlk aşamada nominal sarkaç kütlesi ile kararlılık sağlanmış, ardından Adam algoritması kullanılarak sistem parametrelerindeki belirsizlikler dikkate alınarak kontrolör kazançları optimize edilmiştir. Bu yaklaşım, sistemin doğrusal olmayan yapısına rağmen, belirli bir performans kriterini sağlama konusunda başarılı olmuş ve klasik yöntemlerle elde edilemeyecek sonuçlar ortaya koymuştur. Ayrıca, optimizasyon sürecinde kullanılan farklı maliyet fonksiyonları sayesinde sistem davranışı kullanıcı tercihine göre

şekillendirilmiştir.

Yarı sezgisel yaklaşımın temel avantajı, deterministik yöntemlerin sağladığı kararlılık garantisini, sezgisel algoritmaların esnekliği ve geniş arama yeteneği ile birleştiriyor olmasıdır. Özellikle, klasik sezgisel yöntemlerde karşılaşılan yakınsamama, yerel optimumlara takılma ve yüksek hesaplama yükü gibi sorunlar, deterministik başlangıç noktası ve uygun maliyet fonksiyonları kullanılarak minimize edilmiştir. Ayrıca, bu yaklaşımda algoritmaların birbirinden bağımsız çalışabilmesi ve kullanıcı tanımlı performans seviyelerine göre adım adım gelişim sağlaması, hem teorik hem de uygulamalı kontrol problemleri için büyük bir esneklik sağlamaktadır.

Tezde gerçekleştirilen sayısal simülasyonlar ve karşılaştırmalı analizler, önerilen yarı sezgisel kontrolörlerin hem kararlılık hem de izleme performansı açısından etkili olduğunu ortaya koymuştur. Özellikle, tasarlanan kontrolörlerin belirsizlik içeren tüm parametre alanı üzerinde başarılı bir şekilde çalıştığı ve sistem davranışında istikrarlı sonuçlar elde ettiği gözlemlenmiştir. Ayrıca, klasik yöntemlerle tasarlanmış kontrolörler ile yapılan karşılaştırmalarda, önerilen yöntemlerin daha düşük hata değerleri ve daha kısa yerleşme süresi gibi üstünlükleri olduğu gösterilmiştir. Bu durum, önerilen yöntemin yalnızca teorik düzeyde değil, aynı zamanda pratik uygulamalarda da etkili bir çözüm sunduğunu göstermektedir.

Sonuç olarak, bu tez çalışması, deterministik ve stokastik kontrol stratejilerinin eksikliklerine karşı orta yol olarak değerlendirilebilecek yeni bir yaklaşım sunmaktadır. Geliştirilen yarı sezgisel yöntemler, yalnızca kontrol mühendisliği literatürüne değil, aynı zamanda optimizasyon problemlerine genel bir bakış açısı kazandırma açısından da değerlidir. Elde edilen sonuçlar, bu yöntemin geniş bir uygulama yelpazesinde geçerli ve etkili olduğunu kanıtlamakta ve gelecekte farklı sistemlerde uygulanabilirliği konusunda umut vaat etmektedir. Bu kapsamda, önerilen yöntemin daha karmaşık sistemlerde, çok değişkenli kontrol problemlerinde ve donanım uygulamalarında test edilmesi, gelecek çalışmalara ışık tutacaktır.

1. INTRODUCTION

This chapter introduces the subject of the semi-heuristic approach in robust control design. It also provides the main objective and purpose of this thesis study. Then it formulate the research questions which worked as motivation towards this study and the contributions provided through the implementation of the proposed methodologies. Finally, the chapter is concluded by outlining the thesis structure.

The traditional optimization methods often rely on deterministic algorithms constrained by specific mathematical assumptions, such as differentiability and convexity. These constraints limit their applicability to real-world, non-convex, and multi-modal problems. Semi-heuristic optimization offers a promising alternative by combining deterministic and probabilistic strategies to explore solution spaces more effectively.

This thesis delves into semi-heuristic optimization techniques, exploring their potential to improve system performance in challenging environments. By integrating concepts from both deterministic and random search algorithms, we aim to establish a robust framework capable of overcoming traditional limitations and delivering near-optimal solutions across diverse applications.

1.1 Purpose Of The Study

Optimization is a cornerstone of modern science and engineering, underpinning advancements in diverse fields such as operations research, artificial intelligence, and industrial design. Real-world optimization problems often involve complex objectives, high-dimensional search spaces, and intricate constraints, making them challenging to solve efficiently. Addressing these challenges is crucial for improving system performance and achieving optimal solutions in various applications. During this thesis study, we demonstrated semi-heuristic techniques that utilized deterministic and probabilistic approaches to define a better optimization methodology to approach a user-defined level of performance.

Traditional optimization techniques, including gradient-based methods and combinatorial search algorithms, are often limited by their reliance on strict mathematical assumptions such as differentiability, convexity, and linearity. These limitations make them unsuitable for non-convex, discontinuous, or multi-modal problem spaces, which frequently arise in real-world scenarios. Furthermore, these methods can suffer from local optima entrapment and computational inefficiency.

Semi-heuristic optimization methods have emerged as a promising alternative to overcome the shortcomings of classical approaches. Combining the rigor of mathematical optimization with the flexibility of heuristic strategies, semi-heuristic methods leverage domain knowledge and adaptive mechanisms to explore complex search spaces effectively. These methods strike a balance between exploration and exploitation, enhancing their capability to find near-optimal solutions efficiently.

1.2 Problem Formulation

Objectives like system stability and model performance are essential elements during controller design. It is according to the system and the nature of the problem that one of the assets is leveraged upon the other. However, a trade-off point is often met by the controller designer. Assuming that the system parameters are fixed and not varying, we can obtain a rigid mathematical formulation for the system in either the frequency/time domain, or both. Accordingly, the designed controller can have specific parameters or gains depending on the controller topology. It can also be expected to derive a system to stability using classical control approaches in the case of observability and controllability.

Nevertheless, this straight forward modeling and controller design is not the usual case in practice where we face changes in the system model from linearization, stochastic behavior, noises, or even disturbances. These changes lead to system uncertainty which it is required to be introduced in system modeling and taken into account while designing our controller. Despite the fact that these uncertainties can be colored and anticipated for some of the physical applications, it is not possible to do the same for others. Knowing that we shall design a controller that covers all of these uncertainties. This makes the trade-off between system stability and performance

harder to maintain at a specific level. Many control theories provide deterministic approaches to solve this issue, but it is often computationally expensive. Another side of the state-of-the-art utilizes random optimization where we don't require to have a determined mathematical equation to describe the system concerned. These contradictions raise the following interesting research problems:

- Can we introduce a hybrid control approach that leverages the benefits of deterministic and stochastic approaches?
- Is it possible to constrain the random approaches in order to guarantee convergence?
- Can the physical system comply with this hybrid model?

1.3 Objectives Of The Thesis

The primary objective of this thesis is to investigate the principles and applications of semi-heuristic optimization in control system design. In particular, the study focuses on the methodologies introduced in two different random search algorithms. The first algorithm lays the groundwork for semi-heuristic techniques, while the second presents significant advancements in the field, considering a common control design problem such as the rotary inverted pendulum. This thesis explores their theoretical underpinnings, algorithmic frameworks, and practical implementations.

1.4 Overview Of Contributions

In this study, we designed two semi-heuristic approaches that are suitable for stabilizing and enhancing the performance of two control problems. Each one of them is resolved with a predefined user level of performance and guaranteed stability. The academic contributions that can be inferred from this study are:

- A comprehensive review of existing optimization techniques and the evolution of semi-heuristic methods that can be used for interval polynomial systems.
- An in-depth analysis of applying gradient descent to a random PID optimization technique.

- Development and validation of novel algorithms inspired by the principles of semi-heuristic optimization to control an inverted pendulum model.
- Application of these algorithms to real-world problems, demonstrating their efficiency and robustness.

1.5 Thesis Organization

The remainder of this thesis starts with addressing uncertainty modeling, random optimization, and rotary inverted pendulum state-of-the-art in Chapter 2. Then semi-heuristic justification is well addressed in Chapter 3 by discussing the major drawbacks of deterministic and random search approaches. After this literature being discussed, a thorough mathematical foundation is shown in Chapter 4, which includes uncertainty control approaches, random search techniques' topologies, cost functions, and rotary inverted pendulum mathematical modeling. Chapter 5 introduces the decoupled algorithms based on gradient descent and adaptive moment estimation approaches, while Chapter 6 evaluates the proposed methods through experiments and model testing. Finally, the thesis is concluded by addressing the findings, highlighting strengths, limitations, and potential future research directions in Chapter 7.

2. LITERATURE REVIEW

This chapter highlights parameter uncertainty impacts on control systems design and how uncertainty can be taken into account during this modeling process. Then, random optimization approaches state-of-the-art is presented in general, and their classification based on propagation nature. The provided literature is used later in the next chapter to introduce the semi-heuristic approach importance. Finally, the chapter is concluded by summarizing the control algorithms used for rotary inverted pendulums in literature—a benchmark problem in control theory due to its non-linear and unstable nature.

2.1 Systems With Parameter Uncertainty

System stability and performance are pivotal points during the design of a controller for any system. While the stability of the system ensures that all signals converge to zero in the absence of external effects, the performance dictates the design's ability to follow the input references despite of the applied disturbances [1]. Both metrics are impacted with the appearance of uncertainties that can emerge from the system itself or the surrounding environment. Regardless of the uncertainty root, further robustness is required to ensure the system's stability and performance are admissible under all bounded circumstances. Therefore, the terminology of robust stability and robust system performance is given for systems designed to maintain these characteristics in spite of uncertainties [1].

2.1.1 Uncertainty sources

As we mentioned before, the uncertainty in system parameters can roughly be separated into exogenous and endogenous uncertainties according to their causing nature. We will briefly highlight some of the major causes of the system uncertainties as shown below:

(a) Measurement noise

Measurement noise is an extrinsic system signal that often disturbs system's

stability. It can be caused by undistinguished sensors with low quality or wear-and-tear effect. The impact introduced by the measurement noise variates according to its frequency; the noise with very wide range usually called white noise and modeled stochastically with Gaussian distribution, while the noise with specific band of frequency known as colored noise and it can be modeled stochastically or deterministically according to its nature [2]. Considering the impact of measurement noise upon systems, we can classify it as one of the major exogenous causes of the uncertainty.

(b) System linearization and nonlinearities

In many of the conventional control system design, the nominal operating point of a system is taken into account then a linear time-invariant system is considered to develop the stabilizing controller. Following that a generalized form is developed to cover system nonlinearities. For example, during the linearization process of the rotary inverted pendulum in Chapter 4 we assumed that $\sin(\theta) \approx \theta$ for θ around zero. However, whenever the value of θ differs a lot from zero, the linearized model might not be valid for the new operating points and can cause the whole system to lose stability and performance. Accordingly, a better representation of the system shall include uncertainties obtained from system linearization.

(c) Stochastic parameters

Systems can sometimes contain signals with stochastic behaviors. In such a case, the system modeling can be limited to stochastic approaches since these signals cannot be deterministically described. An example of such systems is a mechanical vibration system in which the stiffness of the material or structure varies randomly over time due to factors such as material degradation, temperature fluctuations, or microstructural changes [3]. In these systems, the stiffness parameter is modeled as a random process, leading to stochastic differential equations that govern the system dynamics. Because randomness is embedded within the inherent properties of the system rather than coming from external disturbances, a purely deterministic model would fail to accurately capture the behavior of the system. Therefore, stochastic modeling becomes necessary to properly analyze and design control strategies for such systems.

(d) System order reduction

Mathematical models for dynamic systems are obtained to mimic the system behavior at all operating frequencies. However, some of the systems have complex mathematical representation with high model order, making them difficult to control. One of the examples for that is the robotic model arms with rigid body dynamics and flexible modes [4]. The flexible mode are ignored while designing controller for the robotic arms since they are operating within high frequencies. Although the nominal case might not be affected by this reduction, the uncertainties in the intrinsic parameters of the model are clear in high frequencies.

(e) Network time delays

Network time delays are exogenous uncertainty parameters in networked control systems (NCS). Its effect might be clear in industrial control systems design, where a set of processes/plants are cascaded and critically dependent on accurate timing. The delay is often mapped between the system and the actuator or the system and the sensor. Regardless of the delay root, it generates uncertainty in the system, and either a deterministic or stochastic representation for the delay shall be considered to better represent NCS [5].

2.1.2 Uncertainty objectives and constraints

Prior to design a control system, several objectives are set to be accomplished. Major design criteria for a system with unity feedback, plant disturbance, and measurement noise can be listed as follows:

(1) The system should have minimal steady-state error when the input signals and disturbances are steps and ramps of arbitrary and unknown magnitude and slope.

(2) The output signals follow reference input signals without excessive use of control energy

(3) Design must ensure a reasonable trade-off between the sensitivity function and the complementary sensitivity function. In other words, a built-in trade-off between reference, disturbance, and noise errors [1].

(4) The closed-loop system poles are located within the left half plane [6].

With the introduction of uncertainty in intrinsic and extrinsic system parameters, the following changes can be applied to our control objectives:

(1) The system should have global minimal steady-state error for all of the uncertainty region when the input signals and disturbances are steps and ramps of arbitrary and unknown magnitude and slope.

(2) The output signals follow reference input signals without excessive use of control energy, and the designed controller shall be valid for changing system parameters.

(3) Closed-loop pole spread is located within the left half plane for all uncertainty region.

2.1.3 Uncertainty modeling

Given that we are aware of the root causes of the uncertainty parameters and how they intervene with our control design objectives, we are supposed to model them mathematically as a first step to design our control system. The approach that we are eager to take during the modeling directly dictates the possible controlling methods. As we mentioned before that the systems can sometimes contain signals with stochastic behaviors, which leads to stochastic parametric uncertainty; the modeling of such systems forces us to use stochastic mathematical models. Among the most used models for systems with uncertain parameters, we shall cover the following:

2.1.3.1 The Parameter Modeling Approach

The parametric approach is one of the oldest models to be used for modeling systems with uncertainty [1]. The model upholds the utilization of the transfer function in the frequency domain and the state-space representation in the time domain. It introduces a set of uncertain parameters -typically a one-dimensional array- modeled as time-varying functions and constants in time and frequency domains, respectively [6]. Each of the parameters is bounded with a one-dimensional array of values that accumulate to define the parameter space. Any possible representation of the system

corresponds to a single point within this parameter space.
During this thesis study we will utilize this approach of modeling uncertainty since it is less complex and easier to obtain from conventional system representations.

2.1.3.2 The Linear Fractional Transformation (LFT) Approach

The Linear Fractional Transformation (LFT) approach is a powerful modeling framework widely used in robust control theory to represent uncertain systems systematically [7]. An LFT decomposes a system into two parts: a known nominal system and a structured uncertainty block interconnected via feedback loops. The nominal part is a deterministic transfer function, while the uncertainty block can encapsulate both parametric and dynamic uncertainties, allowing a unified treatment of multiple uncertainty sources.

The LFT framework offers several advantages:

- It enables a compact and flexible description of multi-variable uncertainty structures.
- It facilitates the application of advanced robust analysis tools such as μ-analysis and μ-synthesis.
- It can capture a wide range of uncertainties, including time delays, unmodeled dynamics, and correlated parameter variations.

Despite these strengths, the LFT approach introduces considerable complexity in modeling and computational burden in analysis. Consequently, for the purposes of this thesis study — where the system uncertainty is simple and predominantly parametric — the LFT model was not adopted.

2.1.3.3 The Unstructured Uncertainty Approach

The unstructured uncertainty modeling approach describes system variations without relying on a specific known structure or parameterization [8]. Instead, it models uncertainties as additive or multiplicative deviations around a nominal plant:

- **Additive Uncertainty:** represents an unknown but bounded perturbation added to the nominal transfer function.
- **Multiplicative Uncertainty:**modifies the nominal behavior proportionally across frequencies.

Unstructured uncertainty is especially useful when:

- High-frequency unmodeled dynamics (such as sensor noise, actuator nonlinearities) are significant.
- Detailed modeling of parameter dependencies is impractical.

However, this method tends to be conservative because it assumes the worst-case uncertainty across the entire frequency range. For the relatively simple and structured uncertainty of the rotary inverted pendulum considered in this thesis, unstructured uncertainty modeling was not employed.

2.1.3.4 The Structured Uncertainty Approach

Structured uncertainty modeling extends parametric uncertainty concepts by considering specific relationships between uncertain elements [9]. Rather than treating uncertainties independently, structured approaches model interdependencies and block structures explicitly, leading to less conservative analysis.

In a structured uncertainty setting:

- Uncertainties are often grouped into block-diagonal matrices.
- Each block may represent parametric variation, dynamic uncertainty, or even time delays.

Structured uncertainty models enable the use of powerful robustness analysis techniques such as μ-analysis, which precisely quantifies the smallest destabilizing perturbation. These models are crucial in multi-input multi-output (MIMO) systems or when correlated parameter variations are present.

Several specialized uncertainty modeling techniques offer alternative strategies to improve robustness analysis:

- **Kharitonov's Theorem:** A fundamental result that asserts that the stability of a family of polynomials with interval uncertainties can be guaranteed by checking only four special vertex polynomials [1].
- **Edge Theorem:** Provides conditions under which the stability of a convex polytope of polynomials can be ensured by examining only the polynomials defined along the edges of the polytope [10].
- **Integral Quadratic Constraints (IQCs):** A general framework that models uncertainties and nonlinearities by satisfying certain quadratic inequalities. IQC theory unifies robust analysis and control synthesis into a tractable convex optimization problem [11].

Among these models, Kharitonov's Theorem was utilized in this thesis. Its simplicity and analytical strength align well with the problem scope and computational constraints of this study.

2.2 Random Search Based Control Algorithms

Stochastic algorithms introduce randomness into optimization processes, enabling the exploration of complex solution spaces and providing robust solutions for control systems with inherent uncertainties. These methods are particularly effective in addressing the challenges posed by non-linear and unstable systems, such as the rotary inverted pendulum. Some of these stochastic approaches offer the benefit of exploring intricate design spaces without relying on gradient information, making them suitable for black-box and non-differentiable systems. Nonetheless, a growing body of research also points out their fundamental limitations and the necessity for more advanced hybrid techniques, as we are going to justify that in chapter 3.

In [12], Rajamäki classifies random search techniques into various categories, including Pure Random Search (PRS), Pure Adaptive Search (PAS), and Annealing Adaptive Search (AAS). Each method presents different degrees of exploitation

and exploration. PRS functions by uniformly sampling the feasible space, which, although straightforward, becomes ineffective in high-dimensional systems due to the exponentially greater number of required samples. PAS enhances PRS by focusing sampling on increasingly promising areas. AAS incorporates simulated annealing concepts to balance exploration and convergence. The author notes that while these strategies are solid alternatives to deterministic algorithms, they often experience slow convergence rates and limited scalability.

The theoretical foundations and practical applications of random search techniques can be expanded in control scenarios [13]. The author suggests that while approaches like PRS provide simplicity and theoretical convergence assurances under minimal conditions, their practicality is constrained in high-dimensional optimization due to inefficiency. PAS and AAS are recognized for their adaptive features, with AAS being especially suitable for control tasks involving uncertainty and noise.

Within control systems, randomized algorithms have been applied to robust controller tuning, particularly in the presence of uncertain parameters. Many researchers suggest that these techniques are utilized for probabilistic performance assessment, where traditional worst-case evaluations would be computationally intense. Randomized algorithms facilitate the estimation of controller performance amid uncertainty, making them attractive for safety-critical applications that require high levels of assurance [14].

Nonetheless, random search methods are not without their critiques. Rajamäki identifies the lack of reproducibility and sensitivity to algorithm parameters as significant issues. The stochastic characteristics of these techniques imply that outcomes can differ greatly across multiple runs, complicating benchmarking and validation processes. Furthermore, improper tuning of hyperparameters such as sampling distributions, population size, or cooling procedures can severely impair performance.

Recent advancements aim to mitigate these shortcomings; meta-control strategies have also been introduced, permitting dynamic adjustments of algorithm parameters based on performance feedback, enhancing adaptability to evolving optimization landscapes.

An additional notable area of progress involves the application of probabilistic models to steer the search, as demonstrated in Bayesian optimization and surrogate-assisted approaches [14]. These methods strive to enhance sample efficiency and convergence rates, especially in evaluating costly objective functions, such as those encountered in high-fidelity control simulations.

The literature presents a nuanced viewpoint on random search methods in control systems. While they provide a strong alternative to deterministic techniques, particularly under conditions of uncertainty and complexity, they also introduce considerable challenges relating to efficiency, scalability, and reliability. For the remainder of this section, we will cover major random optimization paths for control system design.

2.2.1 Gradient propagation based optimization

Gradient propagation-based optimization encompasses a set of optimization techniques that utilize gradient computations to iteratively refine parameters or decision variables with the aim of optimizing an objective function. These methods are essential to contemporary in control theory, especially in contexts where differentiability can be applied to facilitate convergence.

Gradient-based techniques like Gradient Descent, Stochastic Gradient Descent (SGD), and their variants have become prevalent due to their effectiveness in managing high-dimensional optimization challenges [15,16]. These approaches take advantage of the local curvature of the objective function landscape to consistently move towards an optimal solution [17,18].

The notion of gradient propagation is further developed through adjoint sensitivity methods and optimal control theory. These methods help ascertain how input variations influence system outputs and cost functions over time, making them particularly significant for applications such as trajectory optimization and system identification [19].

The main advantage of gradient-based techniques is their rapid convergence when faced with well-conditioned and smooth problems. Generally, they offer greater

computational efficiency compared to gradient-free approaches, such as random search or evolutionary algorithms, which often require substantial sampling. Nonetheless, their effectiveness is influenced by the selection of step sizes (learning rates), and they may settle into local optima within non-convex scenarios.

A notable drawback of gradient-based optimization is its dependence on the differentiability of the objective function. In real-world control situations characterized by discontinuities, black-box dynamics, or noise, gradient data may either be inaccessible or unreliable. This shortcoming has prompted the creation of hybrid approaches that blend gradient-based methods with heuristic or random search strategies to achieve a balance between convergence speed and robustness [20].

Gradient propagation-based optimization continues to serve as a fundamental component of contemporary optimization and control methodologies. Its theoretical principles and practical efficacy are driving progress in diverse areas in the control theory domain. However, its shortcomings in effectively addressing non-smooth or non-differentiable systems underscore the ongoing need for hybrid, adaptive, and domain-specific advancements.

2.2.2 Artificial intelligence based controllers

Artificial Intelligence (AI)-driven random controllers have become a key research focus in control theory, offering effective strategies for managing intricate, uncertain, and nonlinear systems. They can be identified as advanced propagation techniques from but not limited to gradient-based optimization approach. The intersection between AI and control theory has been available in the literature for a very long time [21]. These strategies encompass reinforcement learning (RL), fuzzy logic control (FLC), and neural networks (NNs), which integrate randomness and learning within the control framework that can adapt to varying system dynamics and uncertainties.

Reinforcement Learning has attracted significant interest in control applications due to its capability to learn optimal control strategies by interacting with its environment [22]. Within RL, an agent learns to make decisions that enhance cumulative rewards over time. Approaches such as Q-learning, Deep Q-Networks, and policy

gradient methods have been utilized across various control challenges, including robotic motion planning, energy management, and autonomous driving. RL is particularly effective when the system dynamics are either unknown or only partially observable. Nevertheless, RL techniques often necessitate extensive exploration and may experience instability during training, especially in high-dimensional or continuous action contexts.

Fuzzy Logic Control presents another AI-oriented random control approach that addresses uncertainty and imprecision through linguistic rules instead of mathematical formulations [23]. Fuzzy logic controllers are particularly well-suited for systems where accurate modeling is challenging or unnecessary. They employ "if-then" rules to correlate inputs to control actions, facilitating resilient and interpretable control solutions in fields like process control, consumer electronics, and automotive engineering. Despite their effectiveness in managing vagueness, FLCs can face challenges associated with rule explosion in high-dimensional contexts and often require expert knowledge for the design and adjustment of rules.

In addition to the methods mentioned above, Neural Networks are extensively utilized in control theory, serving both as function approximators and as direct controllers. Their capability to approximate intricate nonlinear relationships makes them ideal for modeling unknown dynamics, system identification, and executing control laws. Within adaptive and learning-centered control frameworks, NNs can be trained in real-time to modify control strategies based on observed results. Importantly, when used in conjunction with RL (as in deep reinforcement learning), NNs have shown enhanced efficacy in dynamic and partially structured environments. However, their effectiveness hinges on proper training and data quality, and they can be susceptible to overfitting and a lack of generalization if not managed carefully.

When comparing these AI-driven methods to traditional gradient-based control techniques, several fundamental distinctions arise. Gradient-based methods depend on analytical models and necessitate that the system be differentiable. They provide strong theoretical assurances regarding convergence and stability but lack flexibility when confronted with uncertainties or model inaccuracies. Conversely, AI-driven

strategies offer increased adaptability and resilience in unpredictable or complex scenarios but frequently do not have theoretical guarantees for convergence and demand considerable computation or training data.

Additionally, gradient-based optimization tends to be more effective when a reliable model exists, and gradients can be computed precisely. AI-driven techniques, particularly RL and neural networks, shine in black-box environments but present challenges related to the exploration-exploitation trade-off, interpretability, and computational demands.

2.2.3 Population based optimization

This random optimization approach is the final approach to discuss in this section. It shows essential differences compared to gradient-based optimization methods and AI-driven controllers. Population-based optimization techniques encompass a wide range of stochastic algorithms that utilize a group of candidate solutions to navigate the search space in pursuit of optimal or near-optimal outcomes [24]. These methods are particularly advantageous in control applications characterized by non-convex, multi-modal, or gradient-less optimization landscapes. Notable population-based strategies include Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), each featuring unique mechanisms for exploration and exploitation.

Genetic Algorithms, inspired by the principles of natural evolution, function by evolving a set of solutions through selection, crossover, and mutation [25]. GAs are recognized for their effectiveness in managing complex, discontinuous search environments and have been applied to various control tasks, including PID tuning, adaptive control, and system identification . GAs are proficient in global exploration and can navigate around local minima due to their stochastic characteristics [26]. However, GAs often necessitate meticulous parameter tuning and can be resource-intensive, especially for larger problems.

One of the other common population-based optimization methods is the Particle Swarm Optimization. It employs a group of particles that traverse the search space

while being influenced by their own best-known positions and the global best found by the swarm [27]. PSO has achieved success in numerous control applications due to its straightforwardness and its capacity to quickly converge on effective solutions. It is notably efficient in optimizing continuous parameters and has been extensively utilized in the design of controllers and multi-objective optimization. Nonetheless, PSO can experience premature convergence and stagnation if the swarm lacks diversity or if control parameters, such as inertia weight and acceleration coefficients, are improperly configured.

In addition to the previous algorithms, Simulated Annealing emerges as a population-based optimization approach that is used in random optimization with system control design. It imitates the annealing process used in metallurgy, where a controlled cooling process allows particles to settle into a state of minimal energy [28]. SA employs a probabilistic acceptance criterion to navigate the solution space, permitting it to escape local minima and gradually concentrate on promising areas. In control tasks, SA has been used to optimize cost functions with multiple local minima, such as those encountered in tuning nonlinear controllers or creating robust control systems. SA is capable of managing complex optimization challenges; however, SA typically converges at a slower rate and necessitates a well-crafted cooling schedule for optimal effectiveness.

In contrast to traditional gradient-based optimization methods, population-based techniques offer a considerable benefit in addressing non-differentiable, noisy, or black-box control issues. Gradient-based methods work efficiently when gradients are accessible and the problem is convex or well-defined, but they often struggle with discontinuities or nonlinearity. On the other hand, population-based strategies are less reliant on the nature of the problem and can navigate intricate landscapes with greater efficiency.

Nevertheless, these techniques have their drawbacks. They frequently involve many hyperparameters that need empirical tuning and may require significant computational resources due to their iterative nature. Additionally, their stochastic dynamics can result in variability in outcomes and inconsistencies across different runs.

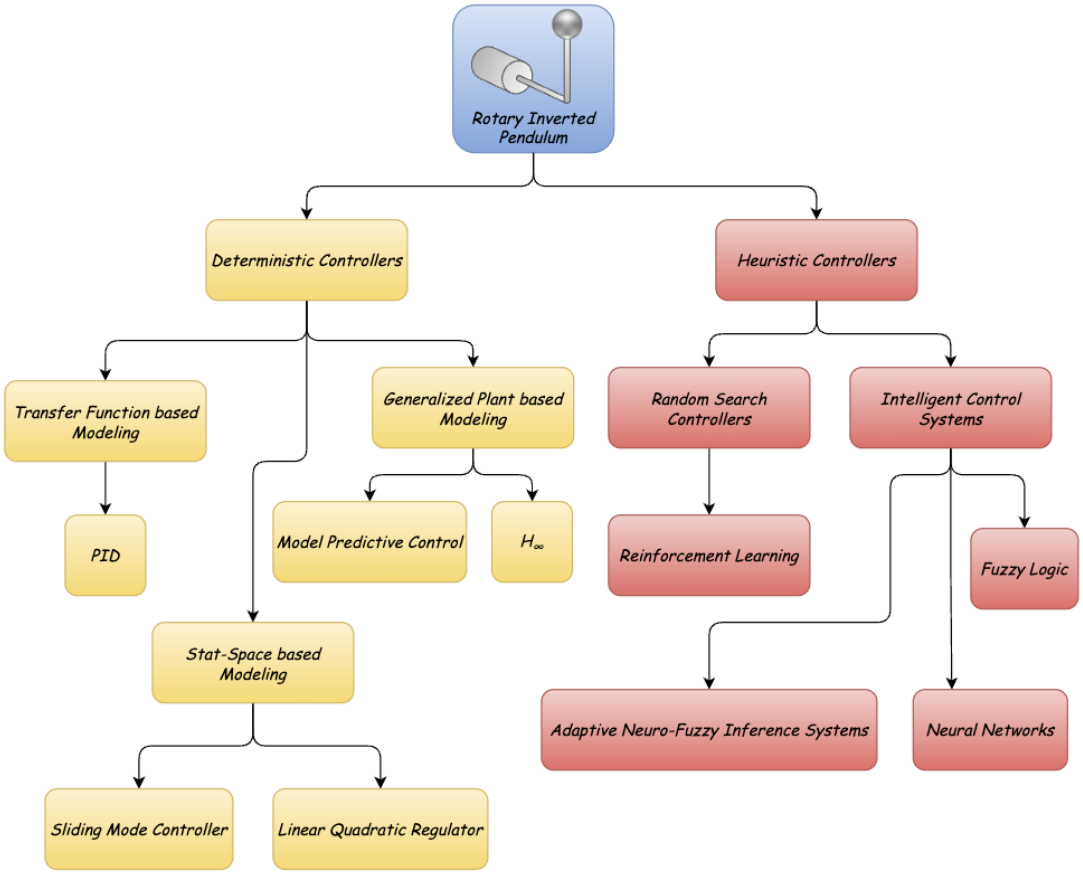


Figure 2.1 : Rotary Inverted Pendulum Control Algorithms

Consequently, recent research has concentrated on developing hybrid approaches that merge population-based global search with gradient-based local refinement to achieve a balance between convergence speed and the quality of solutions.

2.3 Control Strategies For Inverted Pendulums

The inverted pendulum is a quintessential benchmark in control theory and robotics, representing a class of underactuated systems with complex dynamics. Effective stabilization and control of such systems necessitate sophisticated strategies beyond conventional linear controllers. This section explores advanced control methodologies, including intelligent control systems along with robust and adaptive approaches. Figure 2.1 provides the classification of different algorithms used for inverted pendulum control in the literature. At the end of this section, the practical applications of the inverted pendulum application in briefly mentioned to highlight the importance of this problem.

2.3.1 Intelligent control systems

Intelligent control systems are mainly developed based on the AI-driven controller design, which was discussed earlier in section 2.2.2. For this thesis study, we will review the work done using encompassing fuzzy logic controllers, neural networks, and hybrid models like Adaptive Neuro-Fuzzy Inference Systems (ANFIS). However, a variety of examples can be obtained from the state-of-the-art for such a control problem. These systems are adept at handling nonlinearities and uncertainties inherent in dynamic systems. The selected algorithms in this thesis study covers the following approaches:

(1) Fuzzy Logic Controllers

FLCs utilize linguistic rules to model human reasoning, making them suitable for systems with uncertain dynamics, as shown in [29]. In the study, a gyroscopic inverted pendulum utilizing a gyroscopic stabilizer is examined to create a control technique aimed at minimizing the impact of roll motion on the system. The gyroscopic inverted system was modeled using MATLAB/Simulink, and a fuzzy logic-based control strategy was developed. The suggested control method was executed and tested on the actual system. They can demonstrate improved stability and response times compared to traditional PID controllers.

(2) Neural Network Controllers

Neural networks, with their learning capabilities, can approximate complex non-linear functions. Artificial neural networks have been utilized over the past several decades to execute tasks through a comparative learning process. Some areas where ANNs are applied include fitting input-output models, system identification, control, and pattern recognition. However, they can face difficulties in situations involving uncertainty. The paraconsistent logics family presents a robust approach to addressing uncertainty and contradictory information, attracting researchers' interest for its applications and implications in the field of artificial intelligence. In [30], a new activation function based on the paraconsistent annotated logic using two-value annotations (PAL2v) rules, a variant

of paraconsistent logics, facilitating the development of a novel paraconsistent neural net, which is utilized for model identification in control of a closed-loop rotary inverted pendulum system.

(3) Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

ANFIS combines the learning ability of neural networks with the interoperability of fuzzy logic. In [31], the dynamic model of a linear inverted pendulum system was developed using the Newton-Euler method, alongside the design of a conventional PID controller, fuzzy logic control, and self-adaptive fuzzy-proportional-integral-derivative (SAF-PID) control algorithms for system regulation. The goal of these controllers is to maintain the vertical position of the inverted pendulum arms on the moving cart and to stabilize the cart at a designated equilibrium point.

2.3.2 Robust control approaches

This section covers the robust control approaches with deterministic behavior that are common within the research community in stabilizing the inverted pendulum. The robust control strategies are designed to maintain system performance in the presence of uncertainties and external perturbations, which were discussed in the first section of this chapter. We can mentioned the following controller topologies:

(1) Sliding Mode Control (SMC)

The research on the inverted pendulum system highlights a diverse array of controllers available. One prominent type of optimal controller is the sliding mode controller, which is recognized for its strong robustness traits, but it faces challenges during the non-robust reachability phase. In [32], a hybrid controller is introduced for robust management of the rotary inverted pendulum. This design employs LQR as the foundational controller to achieve optimal performance while utilizing a sliding mode controller to effectively handle matched uncertainties. The effectiveness of the proposed controller is demonstrated through real-time validation on a rotary inverted pendulum facing input disturbances. The results

obtained from simulations and experiments indicate that the proposed controller performs well and exhibits robustness in the presence of matched uncertainties.

(2) Model Predictive Control (MPC)

MPC involves solving an optimization problem at each control step, considering future behavior to determine optimal control actions. In [33], a Model Predictive Control (MPC) topology was used to control the rotary inverted pendulum. Due to the unique design of this model, there are discrepancies in both the model and the parameter identification. However, the findings indicate that the MPC controller has superior performance in trajectory tracking and in maintaining stability close to the desired set point.

(3) Linear Quadratic Regulator (LQR)

In [34], a control strategy for the rotary inverted pendulum using a Linear Quadratic Regulator (LQR) controller was designed to stabilize the pendulum in the upright position, known as the unstable equilibrium, and a Linear Quadratic Tracker (LQT) controller was developed to follow a specified trajectory. Additionally, the stability of the closed-loop system is examined to ensure the reliability of the created controller. Simulations are performed within the MATLAB/Simulink environment, and the proposed control strategies have been implemented on hardware that the authors designed. The analysis and results obtained from the system showcase the effectiveness of the control methods, including maintaining stability at the unstable equilibrium point, following the desired trajectory, and demonstrating the robustness and efficiency of the approaches.

Table 2.1 : Comparison of Control Strategies for Inverted Pendulums.

Control Strategy	Complexity	Robustness	Real-Time Implementation	Adaptability
PID	Low	Low	High	Low
LQR	Medium	Medium	High	Low
SMC	Medium	High	Medium	Medium
MPC	High	High	Medium	High
FLC	Medium	Medium	Medium	High
NN	High	High	Low	High
ANFIS	High	High	Medium	High

2.3.3 Comparative analysis of control strategies

A comprehensive analysis of various control strategies provides insights into their relative advantages and limitations. Table 2.1 summarizes key attributes of different controllers applied to inverted pendulum systems.

The study of inverted pendulum systems continues to be a pivotal aspect of control theory and robotics. The development and comparative analysis of various control strategies, from traditional PID controllers to advanced intelligent systems, provide valuable insights into managing complex, non-linear, and underactuated systems. Ongoing research and technological advancements are expected to yield even more sophisticated and efficient control methodologies, broadening the scope of applications in modern engineering.

2.3.4 Applications in robotics and control theory

The principles derived from inverted pendulum studies have been instrumental in advancing various applications in robotics and control theory. On top of these applications, we have the Bipedal robot locomotion. The dynamics of bipedal robots are analogous to inverted pendulums. Insights from inverted pendulum control have facilitated the development of robust balance and mobility control systems in humanoid robots, enabling more natural and stable walking patterns. Additional Application is the self-balancing Segway devices and self-balancing scooters utilize control strategies akin to those of inverted pendulums to maintain stability during motion. These applications underscore the practical relevance of inverted pendulum control methodologies in consumer technology [35]. Finally, we can also mention

the underactuated robotic systems, where the inverted pendulum models serve as foundational examples in the study of underactuated robotic systems, where the number of actuators is less than the degrees of freedom. Understanding and controlling such systems are crucial for the development of agile and efficient robots [36].

3. DETERMINISTIC AND STOCHASTIC APPROACHES LIMITATIONS

In the previous chapter, we introduced the state-of-the-art for uncertainty modeling and its control approaches. We also summarized different control strategies applied to the rotary inverted pendulum. We discussed briefly the deterministic approaches and stochastic approaches advantages and disadvantages when used in control system. However, the necessity of the semi-heuristic approaches has not been elaborated yet. Therefore we decided to introduce this chapter, which will present the main drawbacks obtained from using deterministic/stochastic control approaches with supported mathematical equations. We will also see in the remaining study of this thesis how the semi-heuristic approaches will overcome these drawbacks.

3.1 Deterministic Control Algorithms Complexity

The process of designing a suitable controller for a system is a complex and challenging task that demands meticulous consideration of various factors. These factors may include the inherent dynamics of the system, control objectives, and the influence of external disturbances that can significantly impact the performance of the system [37,38]. It can further be more complicated with the presence of uncertainties in the system parameters, as it requires the selection of robust control parameters to maintain stability with respect to all these uncertainties [2]. Therefore, robust control theory emerged as a necessary field especially after the limitations faced while using optimal control methods such as linear quadratic Gaussian optimal control [6] which has drawn significant attention from the researchers after the introduction of H_2 and H_∞ sensitivity concepts [39].

Systems with varying parameters can be represented with symbolic state space matrices as follows:

$$\begin{cases} \dot{x}(t) = A(q)\,x(t) + B(q)\,u(t) \\ y(t) = C(q)\,x(t) \end{cases} \tag{3.1}$$

where the system uncertain parameters are denoted by $q \in \mathbb{Q}$, while $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $y \in \mathbb{R}^p$ correspond to system states, input and output signals, respectively. Assuming

that we can stabilize the given system above by using a feed-forward controller K, we can compute the updated space state representation as follows:

$$\begin{cases} \dot{x}(t) = [A(q) + B(q)KC(q)]x(t) + B(q)Kr(t) \\ y(t) = C(q)x(t) \end{cases} \tag{3.2}$$

where $r(t)$ is the closed-loop reference signal and the term $[A(q) + B(q)KC(q)]$, also denoted as A_c is the closed-loop system state A matrix. Accordingly, a suitable controller can be driven to ensure stability around the system's nominal values. However, a desired level of performance, such as settling time or steady-state error, may still need to be achieved, which defers according to system's nature and usage conditions. This necessity upgrades our requirement from just finding a stable controller to obtaining a robust controller K_{RS} with a predetermined level of performance (β). It is important to note that the set of controllers $\mathscr{K}_{RS}$ that satisfy the needed level of performance are a subset of the stabilizing set of controllers $\mathscr{K}$. Moreover, the level of performance (β) can be evaluated via a suitable cost function like the Linear Quadratic Lyapunov inequality [40]:

$$A_c^T(q)P + PA_c(q) + C^T(q)K^T RKC(q) + Q \leq 0 \tag{3.3}$$

with Q and R are the given weighting matrices that penalize the state and input, respectively. Where P is defined as a positive symmetric matrix that satisfy the Lyapunov inequality. Furthermore, since we are trying to control a system with uncertainties, the level of performance shall be determined by the worst performance (the lowest Guaranteed Quadratic Cost) for the uncertain parameters combination. Thus, we can define a controller K Quadratic Cost $\rho(K)$ as described by the following equation [41]:

$$\rho(K) = \sup_{q \in \mathbb{Q}} J(q,K) \leq \beta \tag{3.4}$$

where the cost function J can be calculated utilizing 4 different types of error functions: Integral of Absolute Error (IAE), Integral of Squared Error (ISE), Integral of Time multiplied Absolute Error (ITAE) and Integral of Time multiplied Squared Error (ITSE). The error obtained within these functions is defined by:

$$e(t) = r(t) - y(t) \tag{3.5}$$

Combining the previous equations will create a condition such that for all $q \in \mathbb{Q}$, if the inequalities hold, a solution will exist. These matrix inequalities can be expressed as follows:

$$\begin{cases} A_c^T(q)P + PA_c(q) + C^T(q)K^T RKC(q) + Q \leq 0 \\ P = P^T \\ \rho(K) \leq \beta \end{cases} \tag{3.6}$$

Solving the above inequalities may yields to a Bilinear Matrix Inequality (BMI) problem where both P and K are independent variables. BMIs are well-known to be a very difficult problem to solve. Therefore, heuristic approaches can be introduced rather than solving the problem deterministically. Despite the convenient solution that the deterministic approaches provide, it has drawbacks such as computation complexity and inability to obtain a firm mathematical representation for some systems' behaviors where random approaches play an alternative.

Introducing uncertainty for a deterministic model can lead to controllers with varying parameters, since each sampled value within the uncertainty range corresponds to a different characteristic equation. There are two main approaches to follow in this case. The first approach is to ensure the average error of sampled uncertainties converges to a local minimum, while the second approach grantees that the worst-case scenario error is bounded to a prespecified error value [1].

3.2 Random Search Control Approaches Drawbacks

Random search-based control techniques, including stochastic optimization methods, have been widely studied in the field of control systems, as we discussed in the previous chapter, particularly in complex or nonlinear applications where it is difficult to obtain a firm mathematical representation. Approaches such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) utilize random sampling from the solution space as opposed to relying on gradient-based techniques. These algorithms are considered as population-based optimization approaches, which have already been discussed in section 2.2.3. Although these methods are popular and adaptable, they have significant limitations that can affect their performance in specific control scenarios. Random search control methods are fundamentally stochastic and are utilized in situations where the system model is either not defined, significantly

nonlinear, or discontinuous. These techniques are especially beneficial for global optimization challenges, where conventional gradient-based methods struggle.

As an illustration, in Genetic Algorithms, a group of potential solutions develops through processes such as selection, crossover, and mutation. The general framework of GA includes the assessment of a predefined fitness function [42]. The optimization in this algorithm proceeds by selecting individuals that yield better performance and applying stochastic operators to produce new offspring. Similarly, Particle Swarm Optimization mimics social behavior in swarms using the following mathematical model [43]:

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (p_i - x_i^{(t)}) + c_2 r_2 (g - x_i^{(t)}), \quad x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{3.7}$$

where x_i is the position of particle i, v_i its velocity, p_i the best known position of i, and g the global best position found so far. Simulated Annealing, inspired by the annealing process in metallurgy, explores the solution space by accepting worse solutions with a probability that decreases over time [44]:

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{T}\right) \tag{3.8}$$

where ΔE is the change in the objective function and T is the current temperature. Despite their flexibility, random search-based control methods are associated with several significant drawbacks. On top of them sits the computational inefficiency; random optimization techniques frequently demand numerous function evaluations to reach an optimal or near-optimal solution, particularly in high-dimensional environments where significant numbers of algorithm parameters are utilized. In addition to that, the absence of gradient data causes a loss of directional guidance in the search process, resulting in ineffective exploration, which yields expected convergence time $\propto O(n^2)$. This inefficiency is particularly problematic in real-time control scenarios or when evaluating the objective function is computationally expensive.

Another main drawback can be seen within the random optimization techniques is the premature convergence. In techniques such as Genetic Algorithms (GA) or Particle Swarm Optimization (PSO), there is a possibility of quickly settling into suboptimal

solutions, especially when there is a decrease in diversity within the population [45]:

$$\text{Diversity}(t) = \frac{1}{n}\sum_{i=1}^{n} \|x_i^{(t)} - \bar{x}^{(t)}\|^2 \to 0 \tag{3.9}$$

Without mechanisms to maintain or reintroduce diversity, the algorithm may get stuck in local minima. Unlike gradient-based methods, where convergence happens under certain conditions, random search methods often lack strict convergence guarantees. Even when convergence is proven (as in SA under logarithmic cooling schedules), it may require impractically long run times.

It is also necessary to mention that performance effectiveness for random search techniques is greatly influenced by algorithmic parameters, including mutation rate, crossover probability (GA), inertia weight (PSO), and cooling schedule (SA). Incorrect tuning can significantly impair performance. Identifying the suitable parameter combinations is typically specific to the problem at hand and necessitates empirical tuning. This necessity in particular provides a pivotal drawback that cannot be seen in the random search algorithms unless a physical system is used with it. Another essential drawback is due to the stochastic nature, multiple executions of the same algorithm on the same problem may result in different solutions. This stochasticity complicates reproducibility and benchmarking. There are some efforts to address these limitations by random stochastic hybridization with deterministic techniques or the adoption of Bayesian or surrogate-based optimization [46]. These directions may improve convergence efficiency and reproducibility.

Although the drawbacks mentioned in this section are defined mainly for non-gradient optimization techniques, the gradient-based approaches suffer from similar constrains such as premature convergence and sensitivity to hyperparameters. However, each of them can be addressed correctly by introducing a predetermined performance metric and adaptive algorithm parameters, as we will see in the algorithms proposed in chapter 5. It is also necessary to elaborate that the computational complexity for the random optimization is considerably manageable compared to the computational inefficiency developed within the deterministic approaches.

After we have discussed the major types of control approaches drawbacks in the previous sections, we are ready to introduce semi-heuristic approaches. Semi-heuristic optimization techniques combine deterministic principles with heuristic search approaches in order to create moderate progress that merges some of the benefits of each approach and resolves some of the drawbacks of each approach. They have multiple applications in the realm of control system design based on the problem that they are targeting. These techniques strive to fuse the resilience and flexibility of heuristic algorithms with the efficiency in convergence and organization found in classical optimization methods. By doing so, they provide a well-rounded strategy for addressing complex control issues that are frequently nonlinear, high-dimensional, or computationally intensive. Table 3.1 shows the comparison points between deterministic, stochastic, and semi-heuristic control approaches.

Table 3.1 : Comparison of Deterministic, Random, and Semi-Heuristic Approaches

Feature	Deterministic Approaches	Random Approaches	Semi-Heuristic Approaches
Nature of Search	Relies on fixed mathematical rules for predictable outcomes.	Uses randomness to explore solutions broadly without strict patterns.	Combines structured precision with the exploratory flexibility of randomness.
Exploration of Space	Effective in structured spaces relying on system models.	Capable of exploring multi-modal and non-linear spaces.	Balances structured exploration with adaptive random elements.
Handling of Uncertainty	Suitable for bounded uncertainties with well-defined models.	Effective in noisy, dynamic environments with uncertain parameters.	Addresses both static and dynamic uncertainties robustly.
Convergence	Guaranteed under ideal conditions with predefined constraints.	Probabilistic, often requiring extensive iterations for success.	Combines fast convergence with adaptability to uncertainties.
Scalability	Struggles with high-dimensional problems due to computational complexity.	Scales well but may compromise precision for efficiency.	Balances scalability with precision through hybrid techniques.
Adaptability	Performs best with static, predefined models.	Highly adaptable to dynamic conditions and system changes.	Adapts well to both static and dynamic optimization problems.
Applications	Linear control systems, H_∞ loop-shaping, and stability analysis.	Dynamic systems, high-dimensional optimization, and noisy environments.	High-complexity systems requiring robustness and adaptability.
Advantages	Precise, repeatable results with strong theoretical foundation.	Robust in uncertain conditions, flexible, and exploratory.	Combines precision and adaptability for efficient optimization.
Disadvantages	Lacks flexibility for dynamic conditions; computationally intensive.	Computationally demanding and lacks consistent repeatability.	Complex design requirements due to hybrid nature.

4. THEORETICAL FOUNDATION

With the state-of-the-art discussed in the previous chapter, we will set the necessary theoretical foundations for this thesis study within this chapter. We shall start by addressing the basics of controlling systems with parameter uncertainty, the mathematical foundations for optimization algorithms to be utilized, and error functions used to bound these optimization methods. Finally, the modeling and linearization of the rotary inverted pendulum shall be visited. All the thoroughly discussed concepts will be later used in Chapter 5 and Chapter 6.

4.1 Control of Systems with Parameter Uncertainty

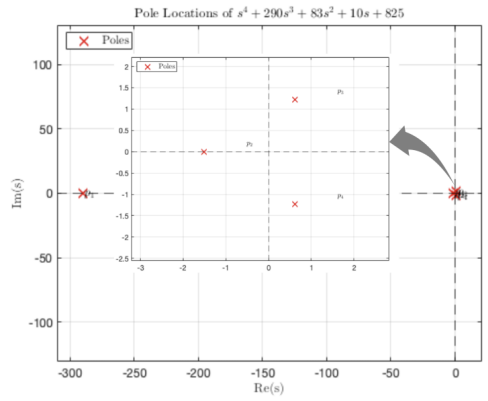
This section initiates the basics of the control system design within the parametric uncertainty presents considering the parametric approach for modeling uncertainty. At the end of this section, we shall describe Kharitanov's theorem, which will be utilized in later chapters.

4.1.1 Parametric modeling approach

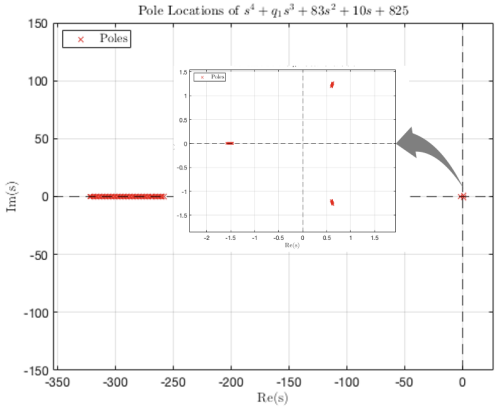
As mentioned earlier, the parametric approach is used in this thesis study to model uncertainty. By modeling the uncertain parameters, we will be able to anticipate these parameters impact into the conventional system representation. Let us consider the following 4th-order system as an example:

$$G(s) = \frac{s+2}{s^4 + 290s^3 + 83s^2 + 10s + 825} \tag{4.1}$$

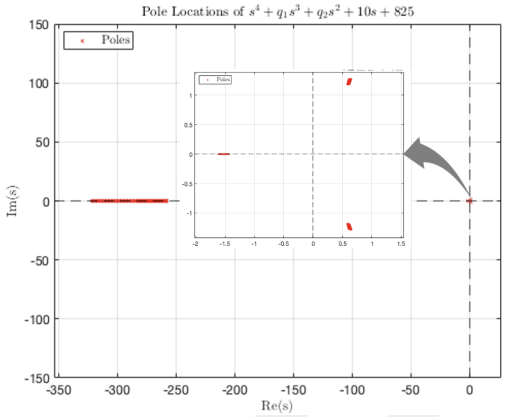
The system has an unstable nature with two complex poles in the right half plane and two stable poles in the left half plane as represented in Figure.4.1(a). The same system can be represented in the Controllable Canonical Form (CCF) state-space model as



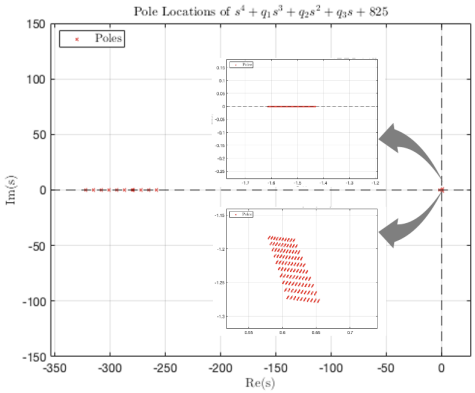
(a) Nominal System Parameter



(b) 1-D Parametric Uncertainty



(c) 2-D Parametric Uncertainty



(d) 3-D Parametric Uncertainty

Figure 4.1 : Pole-Spread for unstable system with different uncertainty space dimensions

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -825 & -10 & -83 & -290 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}(t), \tag{4.2}$$

$$\mathbf{y}(t) = [2 \quad 1 \quad 0 \quad 0]\mathbf{x}(t)$$

where the stability of the system can be measured by state-space representation A matrix eigenvalues. In the case of parametric uncertainty, we assume that the matrices in the state-space representation or the transfer function depend on a vector of real uncertain parameters:

$$q \triangleq [q_1 \quad \cdots \quad q_n]^T \tag{4.3}$$

The vector q, is interpreted as undefined n-Dimensional array of parameter. Each of these parameter is bounded with physical limitations which accumulate to create n-Dimensional uncertainty plane known as Q. Given the initial system transfer

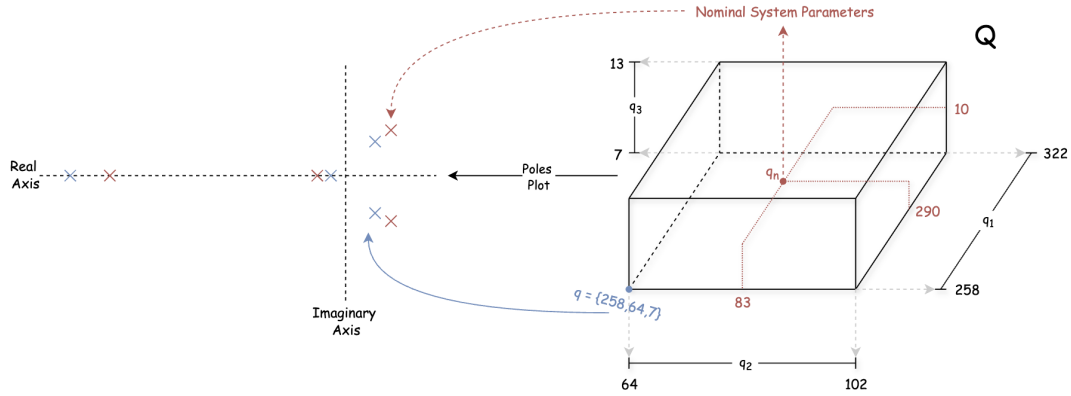


Figure 4.2 : Uncertainty Plane and its corresponding pole spread

function in equation (4.1), we can map the uncertainty in characteristic equation coefficients as:

$$G(s) = \frac{s+2}{s^4 + q_1 s^3 + q_2 s^2 + q_3 s + 825} \tag{4.4}$$

where the boundaries of the unknown uncertain parameters is limited by,

$$Q = \{q \in \mathbb{R}, 258 \leq \mathrm{q_1} \leq 322, 64 \leq \mathrm{q_2} \leq 102, 7 \leq \mathrm{q3} \leq 13\}$$

similarly, the state-space representation of the system can be modeled as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -825 & -q_3 & -q_2 & -q_1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathbf{u}(t), \\ \mathbf{y}(t) &= [2 \quad 1 \quad 0 \quad 0]\mathbf{x}(t) \end{aligned} \tag{4.5}$$

Although the state-space model matrices have uncertain parameters entered as constant values, the parameters can be mapped as time-varying functions, which would be easier to capture system's uncertainty nature. The only added limitation is that all the outputs of these functions are bounded by Q for $t \geq 0$. However, when it comes to transfer function representation we shall define them as unknown constants within Q range.

The impact of introducing uncertainty to the system model can be clearly seen within Figure 4.2. Each points within the parameter box can lead to different system poles and zeros. The wider the range is, the more fluctuating system behavior can be observed. Therefore, uncertainty impact on system stability and performance can be directly

affected with this uncertainty. In addition to the modeling change, the poles plot of the systems is no longer represented with a single nominal location set rather they are spreading within the frequency domain according to the corresponding uncertainty parameter box. This representation is known as **pole-spread**. The pole-spread for 50 samples within Q for first, second, and third order of uncertainty is shown in Figure 4.1(b), 4.1(c) and 4.1(d), respectively.

The transfer function model with uncertainties can be expanded to include multiple transfer functions within the transfer function matrix M as

$$C(s) = M(s,q)R(s) \tag{4.6}$$

The given uncertainty parametric model in equation (4.4) shows that each q is independent of other uncertain parameters, and all of them are nonzero unknown constants. This type of characteristic equation is known as **interval polynomial**, which will be later utilized in Kharitonov's theorem.

4.1.2 Kharitonov theorem

Theorem 1. *[1] Let $P(s)$ be a real polynomial of degree n with interval coefficients defined as:*

$$P(s) = q_0 + q_1 s + q_2 s^2 + \cdots + q_n s^n,$$

where $q_i \in [q_i^-, q_i^+], \forall i = 0, \ldots, n$. Then the entire family of polynomials described by $P(s)$ is stable if and only if the following four Kharitonov polynomials are Hurwitz stable:

$$\begin{aligned}
P^{++}(s) &= q_0^+ + q_1^+ s + q_2^- s^2 + q_3^- s^3 + q_4^+ s^4 + q_5^+ s^5 + \cdots \\
P^{+-}(s) &= q_0^+ + q_1^- s + q_2^- s^2 + q_3^+ s^3 + q_4^+ s^4 + q_5^- s^5 + \cdots \\
P^{-+}(s) &= q_0^- + q_1^+ s + q_2^+ s^2 + q_3^- s^3 + q_4^- s^4 + q_5^+ s^5 + \cdots \\
P^{--}(s) &= q_0^- + q_1^- s + q_2^+ s^2 + q_3^+ s^3 + q_4^- s^4 + q_5^- s^5 + \cdots
\end{aligned}$$

Kharitonov's theorem provides a great opportunity to validate Hurwitz stability in uncertainty presence. Instead of checking the stability within the whole region of

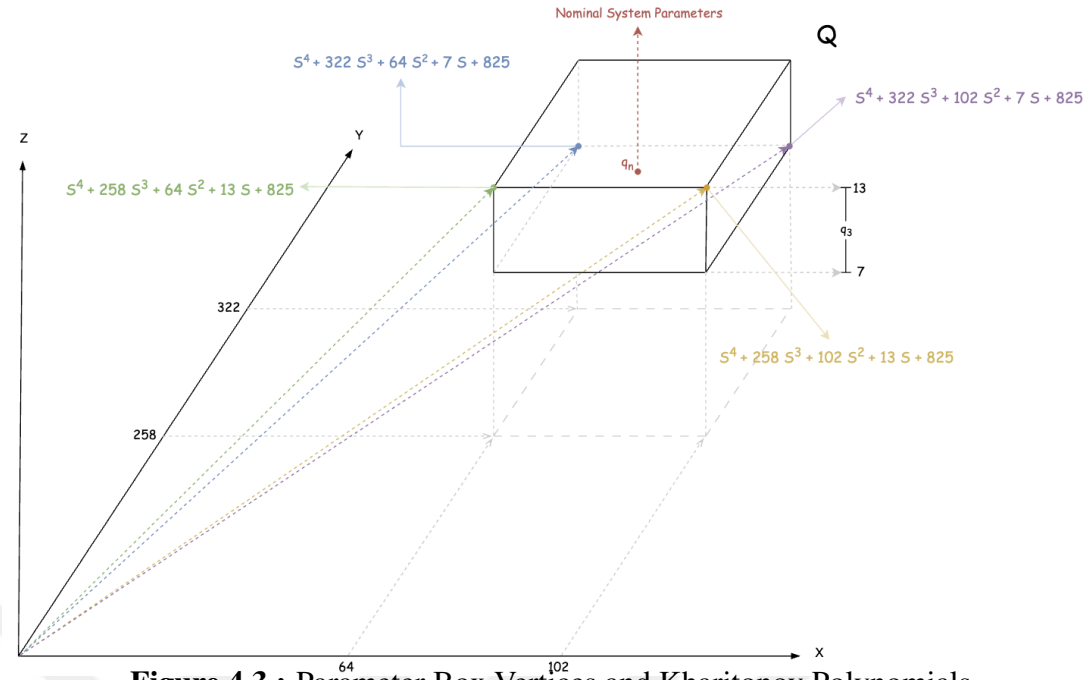


Figure 4.3 : Parameter Box Vertices and Kharitonov Polynomials

uncertainty or increasing the sampling rate within the uncertain parameters box, testing for the stability of Kharitonov polynomials is sufficient to prove system stability for the whole uncertainty region. The visual representation of this theorem can be interpreted by saying that the stability of a set of interval polynomials for any polynomial coefficients in Q can be guaranteed if the polynomials created from the extreme Kharitonov vertices in Q are stable. This visual representation can be seen in Figure 4.3 for the uncertain system described in equation (4.4).

4.1.3 PID controller design for system with uncertainty

Whenever systems with uncertain parameters are concerned, multiple characteristic polynomials must be considered while dealing with stability, as emphasized with the parametric uncertainty modeling in the sections above. Kharitonov's theorem provides a great aid in limiting our polynomials of interest, but the greater picture is not clear unless we implement it within a closed-loop system and define applicable controller parameters. Therefore, we considered a PID controller in this section in order to present the stability analysis for a system with parametric uncertainty and a commonly used controller. In [47], Munro and Soylemez showed that it is possible to obtain the stabilizing range of PID controller parameters by determining the stable proportional

gain K_p range at first then grid to extract the integral gain K_i and the derivative gain K_d.

Let's consider a system $G(s) = N(s)/D(s)$ and input $u = s^2$, we can decompose the system into even and odd parts as follows:

$$G(u) = \frac{N_e(u) + N_o(u)}{D_e(u) + D_o(u)} \tag{4.7}$$

then, we can define the following polynomials

$$X(u) \triangleq D_o N_e - D_e N_o \tag{4.8}$$

$$Y(u) \triangleq D_o N_o u + D_e N_e \tag{4.9}$$

$$Z(u) \triangleq N_e^2 + N_o^2 u \tag{4.10}$$

the positive real roots of X(u) can be denoted by $u_1^*, u_2^*, \ldots, u_\gamma^*$. Assume $u_0 \triangleq -\infty$, $u_{\gamma^*+1} \triangleq 0$, and $u_{\gamma^*+2} \triangleq \infty$ we can define the points x_i such that

$$x_i(u_i^*) \triangleq \frac{Y(u_i^*)}{Z(u_i^*)} \tag{4.11}$$

once $x_i(u_i^*)$ is obtained for $i = 0, 1, \ldots, (\gamma+2)$, the points can be relabeled as $x_i \leq x_{i+1}$. From here, the number of unstable closed-loop system poles for a given gain such as $\frac{-1}{K_p} \in [x_{i-1}, x_i]$ can be determined by:

$$U_i = U_0 + \sum_{t=1}^{i-1} r_t \tag{4.12}$$

where U_0 is the number of unstable roots of $D(s)$ and r_t is calculated as follow:

$$r_t \triangleq \begin{cases} (1-(-1)^l)\mathrm{Sgn}\left(X(u_i^*)^{(l)}\right) & \text{for } 0 < u_i^* < \infty \\ \mathrm{Sgn}(X_0) & \text{for } u_i^* = 0 \\ -\mathrm{Sgn}(X_1); & \text{for } u_i^* = \infty \end{cases} \tag{4.13}$$

where $X(u_i^*)^{(l)}$ represents the first nonzero derivative of $X(u)$ for $u = u_i^*$ while X_0 and X_1 are the first and last coefficients of $X(u)$ respectively.

Using the methodology shown above will allow us to find the stabilizing interval of K_p then grid it, in order to obtain K_i and K_d where for each grid we have 2D stabilizing region. Solving for all sampled K_p gives us complete analysis of the stabilizing PID controllers.

One way to obtain K_i and K_d is by using decoupling at singular frequencies analysis [48]. For the system $G(s)$ that is mentioned before with PID controller in the feed-forward path, the closed-loop characteristic polynomial can be given by:

$$P_c(s,K_p,K_i,K_d) = N(s)(K_i + K_p s + K_d s^2) + sD(s). \tag{4.14}$$

Let $s = j\omega$ and decompose $P_c(s,K_p,K_i,K_d)$ to real and imaginary parts in a matrix form as follow:

$$\begin{bmatrix} R_P \\ I_P \end{bmatrix} = \begin{bmatrix} R_N & -R_N\omega^2 \\ I_N & -I_N\omega^2 \end{bmatrix} \begin{bmatrix} K_i \\ K_d \end{bmatrix} + \begin{bmatrix} -K_p I_N\omega + I_D\omega \\ K_p R_N\omega + R_D\omega \end{bmatrix} \tag{4.15}$$

it is obvious that for all ω the matrix multiplied with $[Ki\ Kd]^T$ is singular so that there are only two possibilities that exist. Either R_p and I_p represent two separate parallel lines (no solution exists) or these 2 lines are identical (a solution exists) that satisfies

$$\frac{R_N}{I_N} = \frac{-R_N\omega^2}{-I_N\omega^2} = \frac{-K_p I_N\omega + I_D\omega}{K_p R_N\omega + R_D\omega} \tag{4.16}$$

This equation can be rephrased as follows:

$$R_N(K_p R_N + R_D) + I_N(K_p I_N + I_D) = 0 \tag{4.17}$$

then for fixed K_p value we can find the singular frequencies ω^* by solving equation (4.15) for ω. Obtaining ω^* will enable us to plot the boundaries of the stabilizing region, which can be classified into the following three cases [47]:

(1) Real Root Boundary (RRB): Occurs when $\omega = 0$.

(2) Complex Root Boundary (CRB): Occurs at a critical frequency denoted by ω^*, where a pair of complex-conjugate roots cross the imaginary axis.

(3) Infinite Root Boundary (IRB): Describes the condition where roots cross the imaginary axis through infinity. This can be analyzed by comparing the number of zeros (z) and the number of poles (p) of the transfer function.

If $p > (z+1)$, then the closed-loop coefficient of its highest order does not depend on K_i and K_d. Hence, there are no IRB exist. All of the above methodology is specified for numeric plants with no uncertainty. However, in the presence of parametric

uncertainties, it is possible to apply this method for finding stabilizing controllers that compensate all possible uncertainties. This can be done by utilizing Kharitonov's Polynomial Theorem described earlier in **Theorem 1**.

The expected outcome of applying Kharitonov's Polynomial Theorem into Munro and Soylemez's proposed method is the formation of four stable $K_i - K_d$ regions. The overlapping area of these regions is identified as the robust stabilizing region that can compensate for the uncertainties introduced into the system. To further increase the efficiency of this approach and reduce computational complexity, Anderson et al. were able to show that it is not necessary to test for all Kharitonov's polynomials [49] using the following theorem:

Theorem 2. *For a given polynomial $P(s)$ of order n, the minimum acceptable test set of Kharitonov polynomials required to prove Hurwitz stability is:*

$$\begin{aligned}
&\{P^{++}\} && \text{for } n = 3\\
&\{P^{++},\ P^{+-}\} && \text{for } n = 4\\
&\{P^{++},\ P^{+-},\ P^{-+}\} && \text{for } n = 5\\
&\{P^{++},\ P^{+-},\ P^{-+},\ P^{--}\} && \text{for } n > 5
\end{aligned}$$

Figure 4.4 demonstrates the utilization of Munro and Soylemez's method with **Theorem 1** and **Theorem 2** to find the stabilizing *Integral Gain - Derivative Gain* region, which is shown in the gray shaded area. The region represents the intersection of all three stabilizing regions founded by drawing root boundaries. It is also necessary to mention that the given figure doesn't include proportional gain values to ensure stability; rather, it is given at the beginning of the analysis. This particular point leaves room for improvement, where we need to utilize a method to obtain the best proportional gain value since our objective is to find an optimum set of $[K_p, K_i, K_d]$. In this case, a random search algorithm implementation can provide a good solution, as we will elaborate in the following chapter.

4.2 Random Optimization Approaches

Semi-heuristic optimization methods are hybrid techniques that integrate deterministic and heuristic strategies to achieve robust and efficient solutions in complex search

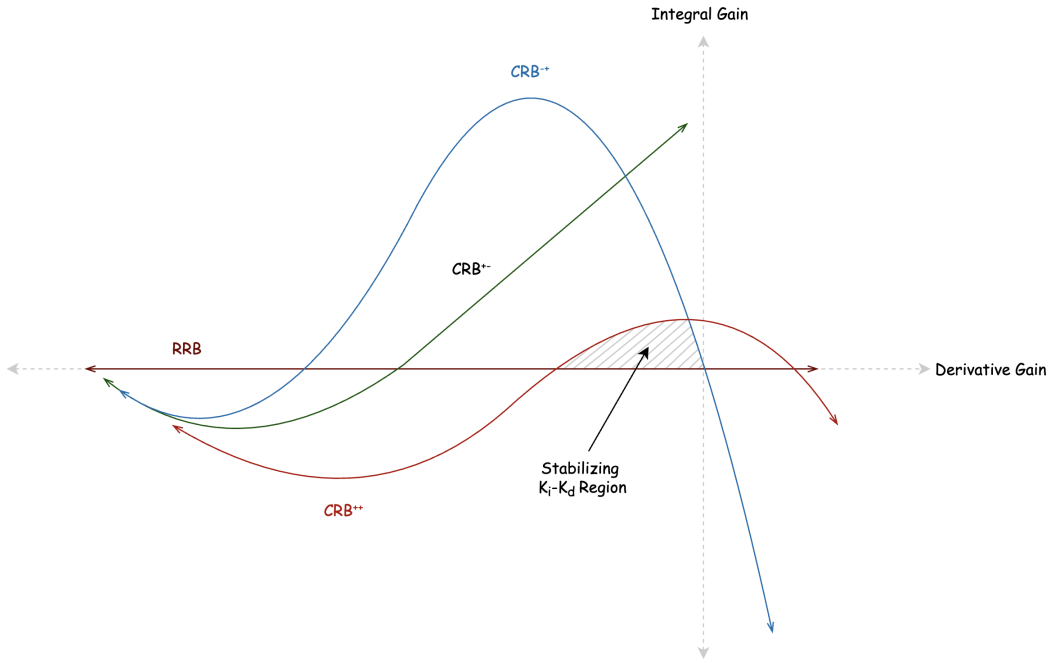


Figure 4.4 : Stabilizing *Integral Gain - Derivative Gain* region for fifth-order characteristic polynomial using Kharitonov Theorem

spaces as described in previous chapter. In this section we shall describe the mathematical basis behind the random optimization approaches used later in chapter 5 to construct our proposed semi-heuristic approaches. Then the cost functions used to bound these algorithms will be elaborated briefly.

4.2.1 Gradient descent algorithm

Batch based gradient descent is a random search algorithm that minimizes an objective function $f(x)$ iteratively:

$$x_{k+1} = x_k - \eta \nabla f(x_k) \tag{4.18}$$

where η is the learning rate and $\nabla f(x_k)$ is the gradient at iteration k [50]. The learning rate dictates the speed of the algorithm while finding local minimums. A high learning rate can decrease processing time effectively, but it might also lead to divergence if the associated gradient is large enough around local minimums. However, a suitable increase in learning rate and gradient is necessary to overcome local minimums and settle within global minimums. Variants like stochastic gradient descent and mini-batch gradient descent improve convergence and computational efficiency. However, it will not be considered in this thesis study for simplicity.

4.2.2 Adaptive moment estimation algorithm

The Adaptive Moment Estimation (Adam Optimizer) is a first-order gradient-based optimization algorithm designed for stochastic objective functions. It combines the advantages of two popular methods: adaptive gradient descent, which handles sparse gradients well, and root mean square propagation, which works well in non-stationary settings [51].

Adam maintains exponentially decaying averages of past gradients (m_t) and squared gradients (v_t), which estimate the first and second moments respectively. In order to correct the bias introduced during initialization, bias-corrected estimates ($\hat{m}_t$, $\hat{v}_t$) are computed.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla f(x_t) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla f(x_t))^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ x_{t+1} &= x_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \end{aligned} \tag{4.19}$$

These enhancements enable faster convergence by reducing computational timing. It also upholds robustness in non-convex optimization problems.

4.2.3 Cost functions

Semi-heuristic approaches aim to leverage the mathematical rigor of deterministic methods while incorporating the exploratory power of heuristics. In control system design, performance evaluation often relies on error functions, which quantify the deviation of system outputs from desired values. The following are the primary error functions used in chapter 5:

- **Integral of Absolute Error (IAE):**

$$J_{\text{IAE}} = \int_0^\infty |e(t)|dt \tag{4.20}$$

  this function penalizes the magnitude of error over time, emphasizing consistent performance.

- **Integral of Squared Error (ISE):**

$$J_{\text{ISE}} = \int_0^\infty e(t)^2 dt \tag{4.21}$$

squaring the error highlights large deviations, making ISE suitable for systems where significant errors are undesirable.

- **Integral of Time-multiplied Absolute Error (ITAE):**

$$J_{\text{ITAE}} = \int_0^\infty t|e(t)|dt \tag{4.22}$$

by weighting errors with time, ITAE prioritizes reducing steady-state errors.

- **Integral of Time-multiplied Squared Error (ITSE):**

$$J_{\text{ITSE}} = \int_0^\infty te(t)^2 dt \tag{4.23}$$

similar to ITAE, but with greater emphasis on reducing large errors over time.

These error metrics are widely used to define objective functions in optimization problems, guiding the search for optimal controller parameters. The major advantages and disadvantages for these four error functions are described in Table 4.1.

Table 4.1 : Comparison of Error-Based Cost Functions

Error Function	**Advantages**	**Disadvantages**
IAE	captures total accumulated error effectively	Less responsive to large or late errors.
ISE	Strongly penalizes large errors	Sensitive to short-duration peaks
ITAE	Improves steady-state accuracy and prioritizes late-stage error reduction.	Underweights early errors
ITSE	Balances size and timing of error and promotes long-term accuracy.	May over-penalize small but persistent errors.

4.3 Inverted Pendulum Mathematical Modeling

This section describes the mathematical modeling of the rotary inverted pendulum model shown in Figure 4.5, which includes equations of motion derivation (EOM), system linearization, and uncertainty modeling using the parametric approach.

4.3.1 Nonlinear mathematical modeling

The kinematics of the rotary inverted pendulum system can be described by using the Euler-Lagrange method:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i - DP \tag{4.24}$$

where Q_i is the generalized force vector, $\frac{\partial D_j}{\partial \dot{q}_i}$ is the dissipative energy from damping, q_i is the system angles and L, the Lagrangian can be defined as

$$L = T - V \tag{4.25}$$

in which T represents the kinetic energy and V represents the potential energy. Both are mathematically defined by equations 4.26 and 4.27, respectively.

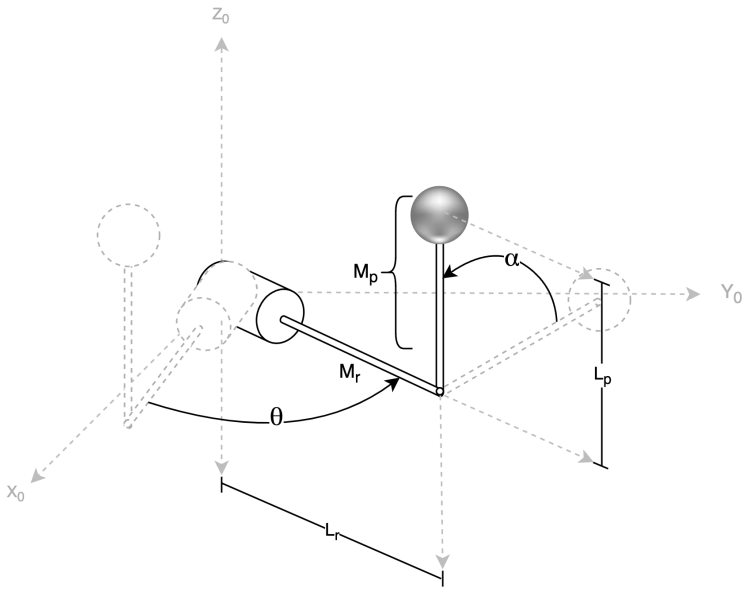


Figure 4.5 : Rotary Inverted Pendulum

$$T = \frac{1}{2}J_{\mathrm{r}}\dot{\theta}^2 + \frac{1}{2}J_p\dot{\alpha}^2 + \frac{1}{2}m_pL_r^2\dot{\theta}^2 + \frac{1}{2}m_pl_p^2\dot{\alpha}^2 + \frac{1}{2}m_pl_p^2\dot{\theta}^2\sin^2\alpha + m_pL_rl_p\dot{\theta}\dot{\alpha}\cos\alpha \tag{4.26}$$

$$U = -m_pgl_p\cos\alpha \tag{4.27}$$

where all parameters of the above equation are described in Table 4.2. Using (4.24), we can define 2 Euler-Lagrange equations according to the system-dependent variables θ and α through the following equations

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\theta}} - \frac{\partial L}{\partial\theta} = Q_1 - \frac{\partial D_r}{\partial\dot{\theta}} \tag{4.28}$$

$$\frac{d}{dt}\frac{\partial L}{\partial\dot{\alpha}} - \frac{\partial L}{\partial\alpha} = Q_2 - \frac{\partial D_p}{\partial\dot{\alpha}} \tag{4.29}$$

using (4.26) and (4.27) we can derive the system equations of motion (EOM) as

$$\begin{aligned}\left(m_pL_r^2 + \frac{1}{4}m_pL_p^2 - \frac{1}{4}m_pL_p^2\cos^2(\alpha) + J_r\right)\ddot{\theta} - \left(\frac{1}{2}m_pL_pL_r\cos(\alpha)\right)\ddot{\alpha} \\ + \left(\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha)\right)\dot{\alpha}\dot{\theta} + \left(\frac{1}{2}m_pL_pL_r\sin(\alpha)\right)\dot{\alpha}^2 \\ = \tau - B_r\dot{\theta}\end{aligned} \tag{4.30}$$

$$\begin{aligned}\left(J_p + \frac{1}{4}m_pL_p^2\right)\ddot{\alpha} - \left(\frac{1}{2}m_pL_pL_r\cos(\alpha)\right)\ddot{\theta} - \frac{1}{2}m_pL_pg\sin(\alpha) \\ - \left(\frac{1}{4}m_pL_p^2\cos(\alpha)\sin(\alpha)\right)\dot{\theta}^2 = -B_p\dot{\alpha}\end{aligned} \tag{4.31}$$

Table 4.2 : Rotary Inverted Pendulum System Parameter

Symbol	Description
θ	Rotary arm horizontal angle
α	Pendulum vertical angle
m_r	Rotary arm mass
m_p	Pendulum mass
L_r	Rotary arm length
L_p	Pendulum link length
J_r	Rotary arm moment of inertia
J_p	Pendulum moment of inertia
g	Earth gravity
τ	Rotary arm motor torque
B_r	Rotary arm viscous damping coefficient
B_p	Pendulum viscous damping coefficient

where τ, B_r and B_p are defined in Table 4.2 as well. The generalized force vector Q_1 is replaced by driving motor torque τ while the generalized force vector Q_2 is set to zero since there are no driving forces in the vertical direction.

4.3.2 System linearization

Although (4.30) provides a detailed mathematical representation for one of the system equations of motion, the torque component remains ambiguous. Since we utilized Quanser QUBE-Servo 2 rotary inverted pendulum system for experimental purposes, we can express the generated torque equation as shown below:

$$\tau = \frac{k_m(V_m - k_m\dot{\theta})}{R_m} \tag{4.32}$$

where k_m is the back-emf constant, V_m is the motor voltage input and R_m is the motor terminal resister. Using this formula the first equation of motion can be reshaped as

$$\left(m_pL_r^2 + \frac{1}{4}m_pL_p^2 - \frac{1}{4}m_pL_p^2\cos^2(\alpha) + J_r\right)\ddot{\theta} - \left(\frac{1}{2}m_pL_pL_r\cos(\alpha)\right)\ddot{\alpha} + \left(\frac{1}{2}m_pL_p^2\sin(\alpha)\cos(\alpha)\right)\dot{\alpha}\dot{\theta} + \left(\frac{1}{2}m_pL_pL_r\sin(\alpha)\right)\dot{\alpha}^2 + \left(\frac{{k_m}^2}{R_m} + B_r\right)\dot{\theta} = \frac{k_m}{R_m}V_m \tag{4.33}$$

Linearization for the rotary inverted pendulum can be done for selected α and θ values. For our case, we preferred to set α to 180° and θ to 0°. Therefore, the linearized EOM can be written as

$$\left(m_pL_r^2 + J_r\right)\ddot{\theta} - \left(\frac{1}{2}m_pL_pL_r\right)\ddot{\alpha} + \left(\frac{{k_m}^2}{R_m} + B_r\right)\dot{\theta} = \frac{k_m}{R_m}V_m \tag{4.34}$$

$$\left(J_p + \frac{1}{4}m_pL_p^2\right)\ddot{\alpha} - \left(\frac{1}{2}m_pL_pL_r\right)\ddot{\theta} - \left(\frac{1}{2}m_pL_pg\right)\alpha = -B_p\dot{\alpha} \tag{4.35}$$

4.3.3 Uncertainty modeling

Due to the nature of the rotary inverted pendulum system, various parameters can lead to system instability. Therefore, during mathematical modeling, these parameters shall either be considered as uncertain parameters or system disturbances in order to study how the designed controller can mitigate their effects. One of the most changing parameters that we considered as our only uncertain parameter is the pendulum mass (m_p). Pendulum mass covers both the pendulum link mass and the sphere object mass

attached at the top of the pendulum link. It will be denoted by Q for the remain of this thesis study. Considering this selection, we can have the system model represented in the following subsections:

4.3.3.1 State-space representation

Given the linearized EOM in the previous subsection, we can select the system state matrix $\mathbf{x}(t)$ as $\begin{bmatrix}\theta(t) & \dot{\theta(t)} & \alpha(t) & \dot{\alpha(t)}\end{bmatrix}^T$ and output matrix $\mathbf{y}(t)$ as $\begin{bmatrix}\theta(t) & \alpha(t)\end{bmatrix}^T$, which means the system state-space first equation can be obtained as:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{\left(\frac{k_m^2}{R_m}+B_r\right)\left(J_p+\frac{1}{4}QL_p^2\right)}{\Delta} & \frac{\frac{1}{4}Q^2L_p^2L_rg}{\Delta} & \frac{\frac{1}{2}QL_pL_rB_p}{\Delta} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\frac{1}{2}QL_pL_rk_m}{R_m\Delta} & \frac{-\frac{1}{2}QL_pL_r\left(\frac{k_m^2}{R_m}+B_r\right)}{\Delta}+\frac{\frac{1}{2}QL_pg}{J_p+\frac{1}{4}QL_p^2} & \frac{\frac{1}{4}Q^2L_p^2L_r^2B_p}{\Delta}-\frac{B_p}{J_p+\frac{1}{4}QL_p^2} \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{\frac{k_m}{R_m}\left(J_p+\frac{1}{4}QL_p^2\right)}{\Delta} \\ 0 \\ \frac{\frac{1}{2}QL_pL_rk_m}{R_m\Delta} \end{bmatrix} V_m(t) \tag{4.36}$$

where,

$$\Delta = \left(QL_r^2+J_r\right)\left(J_p+\frac{1}{4}QL_p^2\right)-\left(\frac{1}{2}QL_pL_r\right)^2$$

Using Quanser QUBE-Servo 2 rotary inverted pendulum system parameters [52] defined in Table 4.3, we can compute the final state-space representation as shown below:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{346.23}{Q+5.06\times10^{-5}} & \frac{-0.94}{Q+1.01\times10^{-5}} & \frac{-0.27}{Q+4.22\times10^{-4}} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{456.27Q+3.61}{Q+0.03} & \frac{-0.93}{Q+5.06\times10^{-5}} & \frac{-0.36Q-0.002}{Q^2+0.03Q} \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{2.76}{Q+0.03} \\ 0 \\ \frac{2.73}{Q+0.03} \end{bmatrix} V_m(t), \qquad \mathbf{Y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{x}(t) \tag{4.37}$$

4.3.3.2 Transfer function representation

Table 4.3 : QUBE-Servo 2 System Parameters

Parameter	Value
K_m	$0.042\ V \cdot s/rad$
R_m	$8.4\ \Omega$
m_r	$0.095\ Kg$
L_r	$0.085\ m$
L_p	$0.129\ m$
J_r	$5.7198 \times 10^{-5}\ kg \cdot m^2$
J_p	$0.0014Q\ kg \cdot m^2$
g	$9.81\ m/s^2$
B_r	$0.0015\ N \cdot m \cdot s/rad$
B_p	$0.0005\ N \cdot m \cdot s/rad$

Assume that the initial states are zeros for the inverted pendulum system. The transfer matrix representation of the system defined in (4.37) can be obtained concerning state-space matrices as follows:

$$\mathbf{Y}(s) = \begin{bmatrix} \dfrac{0.3955s^2 + \dfrac{0.0356}{Q}s - 45.1092}{(Q+0.0045)\Gamma} \\ \dfrac{0.3909s^2 - 1.2243 \times 10^{-9}s}{(Q+0.0045)\Gamma} \end{bmatrix} V_m(s) \tag{4.38}$$

where,

$$\Gamma = s^4 + \frac{0.1868Q + 4.0777 \times 10^{-4}}{Q^2 + 0.0045Q}s^3 - \frac{65.1827Q^2 + 0.516Q - 0.0122}{Q^2 + 0.0045Q}s^2 - \frac{15.4273}{Q + 0.0045}s$$

5. METHODOLOGY AND KEY DEVELOPMENTS

After completing the necessary mathematical and theoretical background for this thesis study, we shall deep dive into the proposed methodologies, which will fabricate the utilization of semi-heuristic approaches within uncertainty systems control design. The first part of this chapter covers how a common controller topology, like a PID controller, can be manipulated to include these approaches. Lately, a second part will include the usage of an advanced optimization technique (Adam) in a real physical system (rotary inverted pendulum).

5.1 Gradient Descent Based Semi-Heuristic Optimization

In chapter 2, we have introduced the deterministic approaches' inability to solve some complex control problems and the heuristic approaches' drawbacks toward time efficiency. In this section, we will introduce a semi-heuristic approach that utilizes the PID designing approach for systems with parametric approach modeled uncertainty shown in section 4.1.3 and batched gradient descent described in section 4.2.1.

The proposed gradient descent-based heuristic approach is divided into two main sub-algorithms; ***Algorithm 1*** starts with nominal system parameters and provides an initial stable controller for predetermined levels of performance λ and γ. ***Algorithm 2*** in the other hand, utilizes the provided initial controller as a gradient descent starting point, then converges to a predetermined level of performance α for all selected uncertainties. Following this process, we will repeatedly select different k_p values, hence different initial controllers, and utilize ***Algorithm 2*** to obtain their corresponding robust controllers. The outcome of our approach is determined by identifying the robust controller that exhibits the best performance according to the specified cost function. Figure 5.2 represents the interaction between those two algorithms along with system characteristics and algorithm parameters.

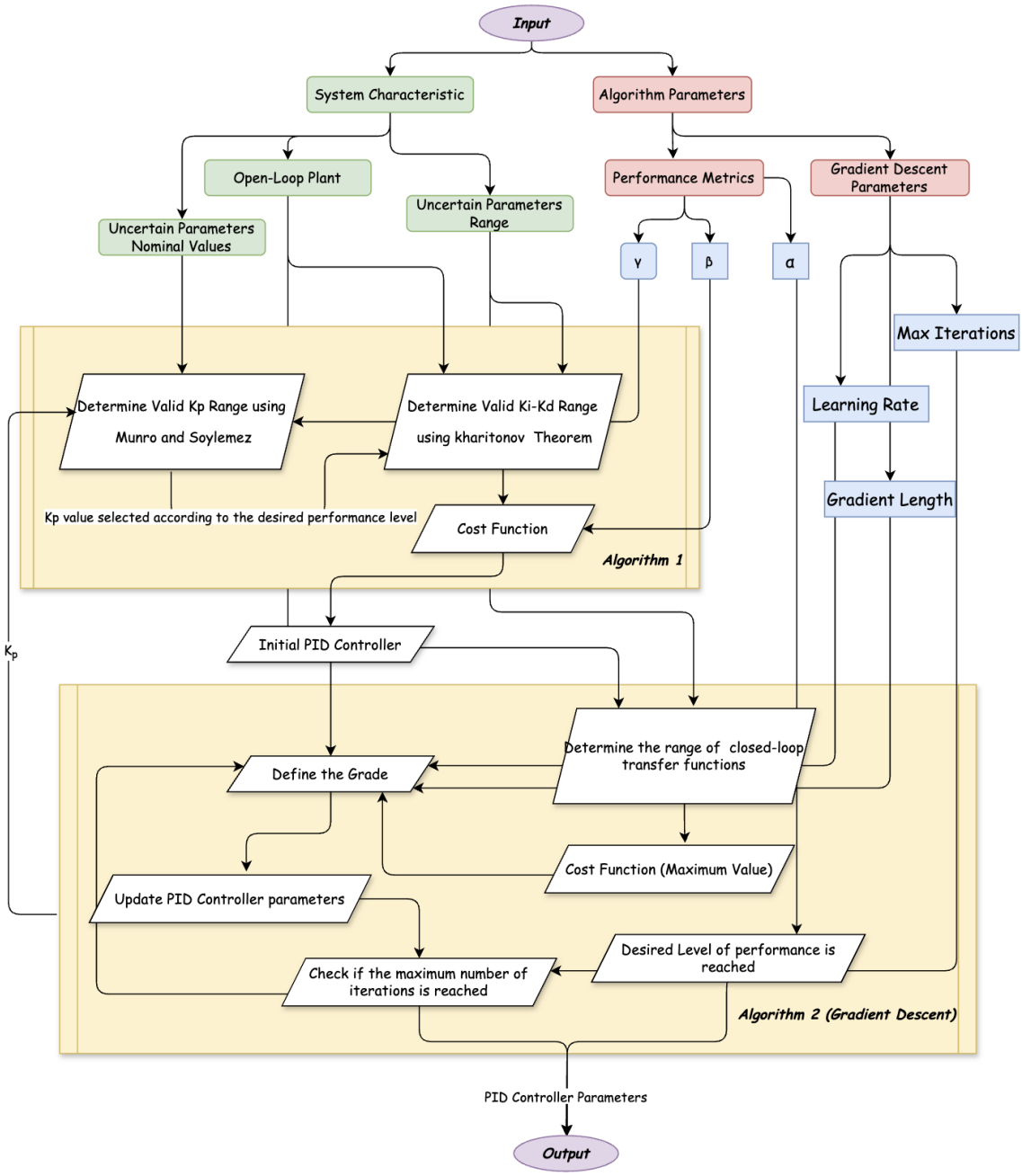


Figure 5.1 : Gradient Descent Based Semi-Heuristic Optimization

Algorithm 1 *(Initial PID Parameters)*

1. *Select the nominal uncertain parameters* q_{ni}

2. *Using the following equation that is derived from equation (4.17), find the stable* K_p *range*

$$K_p = \frac{-I_D I_N + R_D R_N}{R_N^2 + I_N^2} \tag{5.1}$$

3. *Select K_p value that corresponds to an error less than the determined level of performance 1 (γ) using IAE as a cost function.*

4. *Using minimal Kharitonov's Polynomials obtained from* ***Theorem 1*** *and* ***Theorem 2****, determine the stabilizing $K_i - K_d$ surface.*

5. *By implementing a cost function from equation (4.20), (4.21), (4.22) or (4.23), select a $K_i - K_d$ pair that provides a determined level of performance 2 (β).*

6. *The selected PID controller K_{init} is the output of the algorithm.*

Equation (5.1) provides a method to start our random search with a stable controller, which will give us confidence while using Munro-Soylemaz method mentioned in the previous chapter. Defined level of performance α, on the other hand, prevents the algorithm from starting with K_p value that may lead to a set of controllers with poor performance. Similarly, the level of performance β ensures that the selected controller parameters have a good expected performance before starting the gradient descent based searching algorithm. We can see clearly from the stable above that the predetermined levels of performance play a crucial role in enhancing K_{init} performance gradually towards obtaining a robust controller.

Algorithm 2 *(Gradient Descent)*

1. *Determine the range of parameter box Q*

2. *Grid the parameter box according to variation in the uncertain parameters and determine the corresponding transfer function for each q_i combination.*

3. *Determine the variation (grade) in each PID parameter using one of the cost functions specified in equation (4.20), (4.21), (4.22) or (4.23) as well as the gradient descent learning rate. This operation can be determined using the equations below:*

$$K_{c+1} = K_c - a\frac{J(K_c + \Delta) - J(K_c)}{\Delta} \tag{5.2}$$

where J is the selected cost function, K_c is the PID controller parameters, a is the learning rate and Δ is the gradient length.

4. *Update the PID controller parameters according to each grade value of the parameter.*

5. *If the maximum number of iterations is reached or a predefined level of performance 3 (α) is achieved, exit the algorithm.*

6. *Otherwise, define a new grade and start repeating the algorithm from step 2.*

Following ***Algorithm 1***, the initial identified controller parameters are used as a starting point for ***Algorithm 2***. However, prior to that, the parameter uncertainty plane should be gridded into finite user-defined points. These points will be used to compute the worst-case scenario for a global cost function selected in step 3. The selected cost function is mapped in equation (5.2), which is directly obtained from equation (4.18) after generalizing for all PID controller gains. The new defined level of performance α on the other hand, is the ultimate target for system performance that is required by the designer. Therefore, previously introduced levels of performance can be interpreted as intermediate steps to this final level. Nevertheless, the final level of the performance is computed for the whole (gridded) uncertainty region, while the previously defined levels are only applicable for nominal system parameters.

Both of the proposed algorithms are coherently combined to merge the restricted deterministic approach of Kharitonov's theorem while bounding the nominal system stabilizing range and the flexibility of the gradient descent when iterating into a robust and suitable solution. The user-defined levels of performance effectively smooth the intersection of the proposed semi-heuristic algorithm parts.

5.2 Adam Based Semi-Heuristic Optimization

The optimization method described in the previous chapter can be useful in designing a stable PID controller for a system with an interval polynomial and a predetermined/designed level of performance. In this section, we will provide an enhancement on the previously mentioned algorithm by selecting a better random search technique and implementing it on the rotary inverted pendulum control.

The reason behind the selection of the rotary inverted pendulum is that the pendulum is an essential benchmark in control theory and robotics, with its nonlinear and unstable nature that was suitable to challenge this semi-heuristic approach. The newly proposed approach shown in Figure 5.2 is decoupled by the following algorithms:

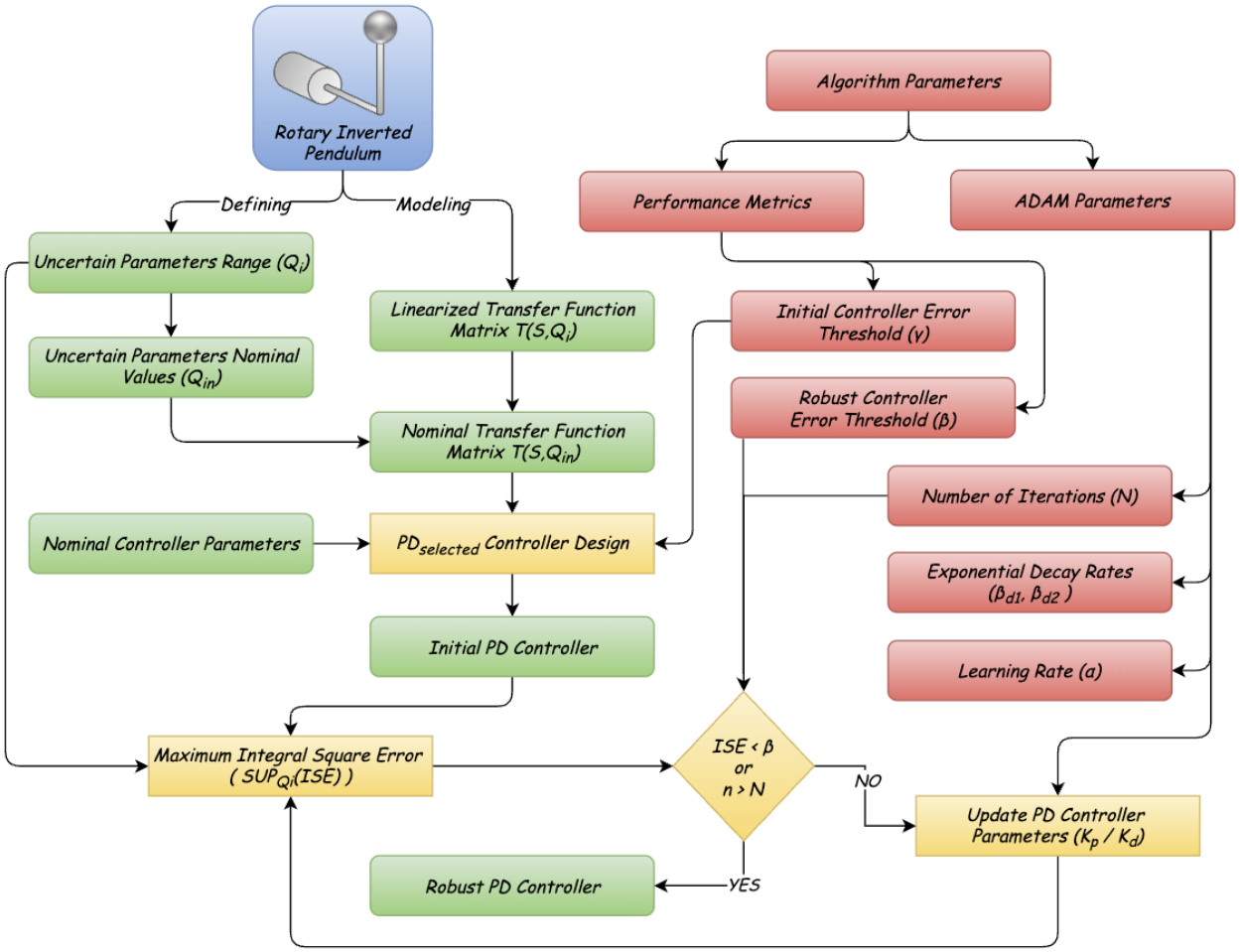


Figure 5.2 : Adam Based Semi-Heuristic Optimization

Algorithm 3 *(Nominal PD Controller)*

1. *Compute the nominal transfer function matrix $T(S, Q_i)$ using system nominal uncertain parameters Q_{ni}*

2. *Using the Dual PD Controller structure shown in Figure 5.3, define arbitrary stable controller coefficients for the system angles*

3. *Select one of the PD controllers to be optimized while fixing the other controller to its initial values*

4. *Modify K_p and K_d values for the selected angle until a controller with ISE error less than the initial controller error threshold (γ) is obtained.*

5. *The selected PD controller K_{init} is the output of the algorithm.*

The dual-PD controller provided in Figure (5.3) creates the basis of the system control design. The stabilizing controllers selected in step 2 are user defined approaches where a deterministic/stochastic approach can be used to define the controller gains. The provided error threshold (γ) is similar to the predetermined levels of performance defined in gradient descent based optimizer; it provides good initialization for the search algorithm to obsolete controllers with poor performance.

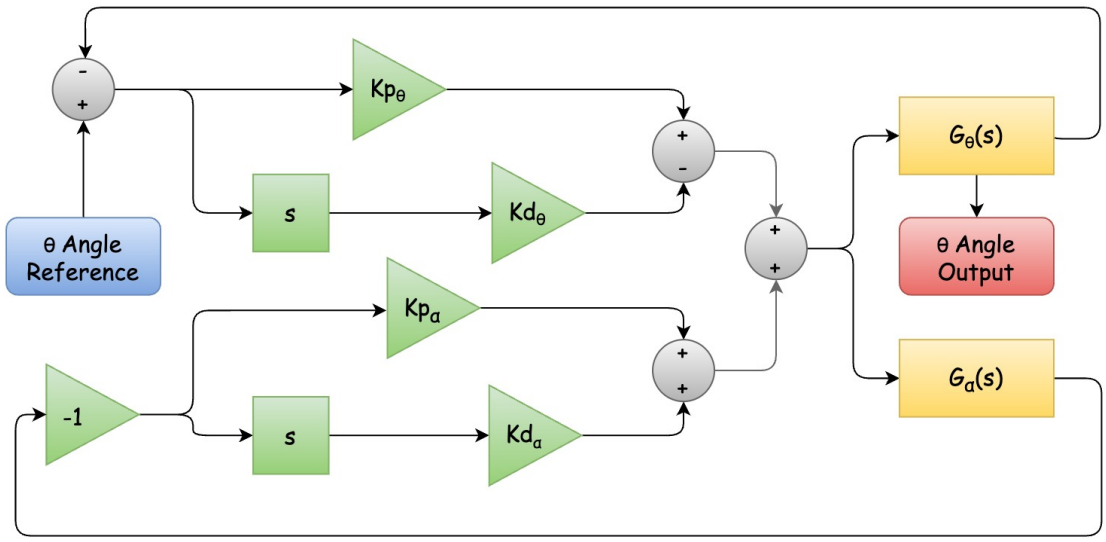


Figure 5.3 : Dual-PD controller for rotary inverted pendulum

Algorithm 4 *(Adam Optimizer)*

1. *Determine the range of parameter box Q*
2. *Grid the parameter box according to variation in the uncertain parameters and determine the corresponding transfer function for each q_i combination.*
3. *Determine the variation (grade) in each PD parameter using the cost function specified in equation (4.21) and the Adam optimizer defined in equation (4.19).*
4. *Update the PD controller parameters according to each grade value of the parameter.*
5. *If the maximum number of iterations is reached or an ISE error less than the robust controller error threshold (β) is achieved, terminate the search.*
6. *Otherwise, define a new grade and start repeating the algorithm from step 3.*

Adam optimizer search algorithm provides a better search technique compared to the gradient descent used in the previous section for the reasons specified in section 4.2.2, which made it suitable for this semi-heuristic approach. Although ***Algorithm 3*** selects one controller per angle, the same semi-heuristic approach can be used to define robust controller performance for each angle separately and compare the controllers' final performance, which will be clear during experimental result analysis in chapter 7.

6. EXPERIMENTAL EVALUATION

In the previous chapter, we proposed two approaches for semi-heuristic optimization based on mathematical foundations derived in chapter 4. In this chapter, we shall thoroughly discuss all the simulation and experimental results obtained during this thesis study. The chapter is divided into two main parts, similar to chapter 5; the first part shows the experimental results obtained during stabilizing an interval polynomial system using the gradient descent based optimization technique - described in section 5.1 - and the second part present experimental data accumulated by utilizing the Adam based optimization approach - derived in section 5.2-.

6.1 Interval Polynomial Control

In chapter 4, we have already mentioned that the parametric approach will be the modeling approach to be used for uncertainty design, and we also mentioned that the proposed gradient descent based algorithm utilizes ***Theorem 1*** which only applicable for interval type polynomials -defined in section 4.1.1-. Hence, we can use the system with the open-loop function defined in the following equation:

$$G(s) = \frac{s+2}{s^4 + q_3 s^3 + q_2 s^2 + q_1 s + 825} \tag{6.1}$$

Since the proposed gradient descent-based optimization approach uses a PID controller with unity feedback, the system closed-loop transfer function can be given by the following equation:

$$G_{cl}(s) = \frac{K_d s^3 + (2K_d + K_p)s^2 + (K_i + 2K_p)s + 2K_i}{s^5 + q_3 s^4 + (K_d + q_2)s^3 + (2K_d + K_p + q_1)s^2 + (K_i + 2K_p + 825)s + 2K_i} \tag{6.2}$$

given that the uncertain parameters vector is demonstrated by $q_i = [q_1, q_2, q_3]$ where $q_1 \in [258, 322]$, $q_2 \in [64, 102]$ and $q_3 \in [7, 13]$. Uncertain parameters nominal values are defined as $q_n = [290, 83, 10]$. The pole spread of the system given in equation 6.1 can be shown in Figure 6.1. From the widely scattered poles location, we can observe the impact of uncertainty presence in system modeling. Accordingly, a wide

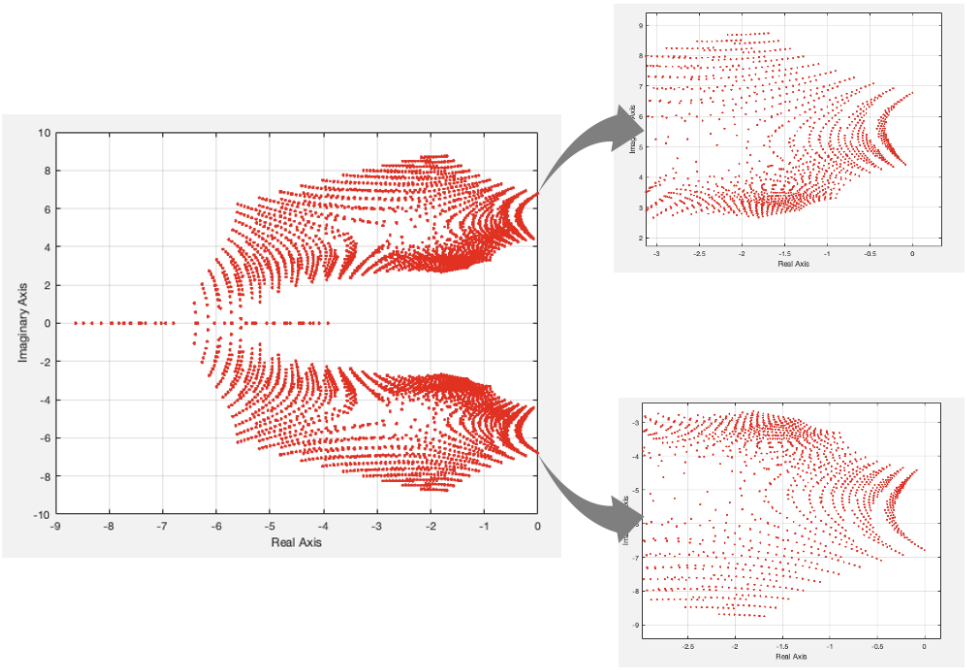


Figure 6.1 : Open-Loop system with interval polynomial characteristic equation pole spread

variation of the controller performance can be expected when it comes to computing error/cost functions. The following subsections represent ***Algorithm 1*** and ***Algorithm 2*** experimental results.

6.1.1 Initial controller design

Following the steps of ***Algorithm 1*** defined in section 5.1, we can find the stable initial controller. First, the open-loop system with nominal values can be obtained from equation 6.1 by substituting nominal parameters as:

$$G_n(s) = \frac{s+2}{s^4+10s^3+83s^2+290s+825} \tag{6.3}$$

Hence, the closed-loop transfer function can be given by:

$$G_{cl}(s) = \frac{K_d s^3 + (2K_d + K_p)s^2 + (K_i + 2K_p)s + 2K_i}{s^5 + 10s^4 + (K_d + 83)s^3 + (2K_d + K_p + 290)s^2 + (K_i + 2K_p + 825)s + 2K_i} \tag{6.4}$$

Using equation (5.1) we can find the stabilizing function for $K_p(\omega)$ as:

$$K_p(\omega) = \frac{(q_3 - 2)\,\omega^4 - (q_1 - 2q_2)\,\omega^2 - 1650}{\omega^2 + 4} \tag{6.5}$$

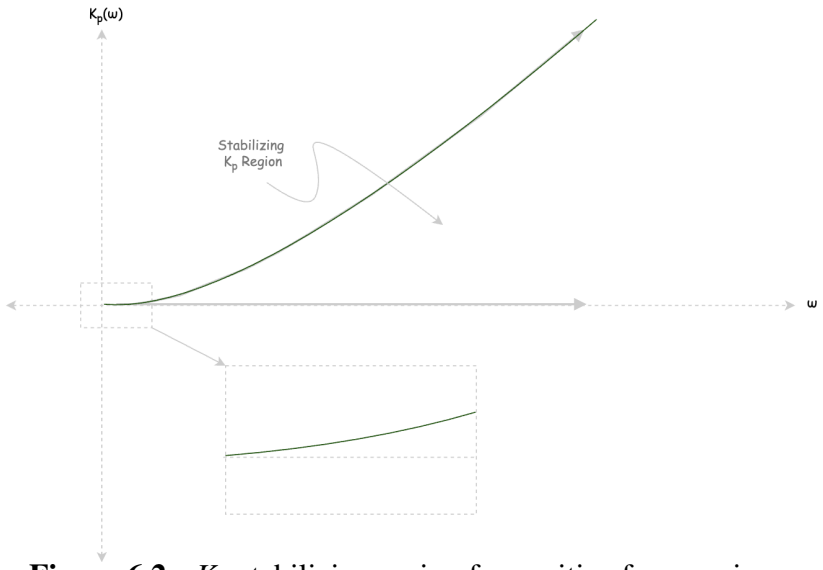


Figure 6.2 : K_p stabilizing region for positive frequencies

The function plot and the stabilizing K_p region against positive frequencies (ω) are shown in Figure (6.2). From the figure, we can interpret the valid K_p range as the positive numbers plane ($K_p \in [0,\infty)$) since the function $K_p(\omega)$ doesn't cross ω axis twice.

The following step in ***Algorithm 1*** requests the selection of a K_p gain that verifies the first desired level of performance (γ). The algorithm specifies that the nominal system parameters and the IAE error function should be used during the selection, but it doesn't determine the $K_i - K_d$ range to be used for this process, which is left for the user's definition. Table 6.1 shows different K_p values associated with different $K_i - K_d$ ranges and their corresponding IAE error.

Table 6.1 : IAE for selective K_p and different $K_i - K_d$ ranges

K_p	IAE_{min} $0 \leq K_i \leq 2000$ $0 \leq K_d \leq 100$	IAE_{min} $0 \leq K_i \leq 2000$ $-50 \leq K_d \leq 50$	IAE_{min} $-1000 \leq K_i \leq 1000$ $-50 \leq K_d \leq 50$
$K_p = 0$	0.476374	0.530421	0.530421
$K_p = 5$	0.460064	0.511379	0.511379
$K_p = 11$	0.442059	0.491304	0.491304
$K_p = 22$	0.413008	0.461612	0.461612
$K_p = 47$	0.353352	0.425356	0.425356
$K_p = 100$	0.263460	0.297289	0.417499
$K_p = 230$	0.235114	0.317546	0.417688
$K_p = 512$	0.331913	0.8965	0.8965

Assuming that $\gamma = 0.9$, $K_p = 0$ satisfies the 1^{st} level of performance, and it can be selected as our initial proportional gain.

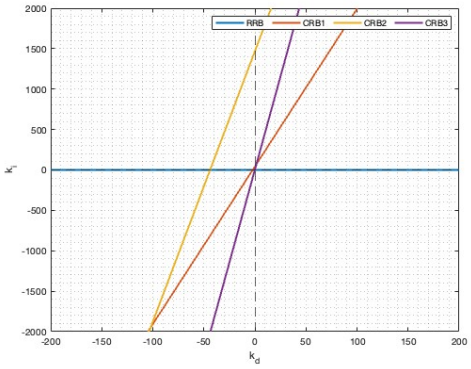
Following the next step of the proposed algorithm, we shall specify or narrow the valid $K_i - K_d$ range using Munro-Soylemez method described in section 4.1.3. The method is defining different root boundary polynomials that can be used to limit $K_i - K_d$ range. The complex root boundaries in particular depend on utilizing ***Theorem 1*** to find dedicated singular frequencies ω^*. Table 6.2 shows different singular frequencies for each K_p value available in Table 6.1 considering all set of Kharitonov polynomials. Despite we are able to compute CRBs for all Kharitonov polynomials, we can utilize ***Theorem 2*** to minimize the required set of polynomials for robust stability by neglecting P^{--} since the closed-loop characteristic polynomial is a 5^{th} order polynomial. Using the three CRBs obtained from the other polynomials and the real root boundary polynomial we can obtain the stabilizing $K_i - K_d$ region as shown in Figure 6.3. The stable region is represented with the intersectional area in the first quarter (positive-positive) plane which we can see from the figure that it decreases with increase of selected K_p value.

Knowing the stable region we can move to the next step in the algorithm which is selecting stabilizing $K_i - K_d$ pair according to error functions defined in section 4.2.3. Figure 6.4(a) , 6.4(b), 6.4(c), and 6.4(d) show IAE, ISE, ITAE and ITSE for fixed $K_p = 0$, respectively. The area highlighted in red at the bottom of each figure represents the stabilizing $K_i - K_d$ region obtained in Figure 6.3(a).

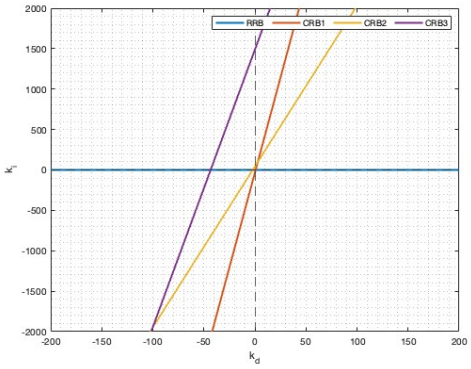
As we agreed earlier to use $K_p = 0$ for initial controller we can find a stabilizing pair assuming the 2^{nd} level of performance (β) is equal to 0.7 and IAE is used for error

Table 6.2 : Singular frequencies for selective K_p values

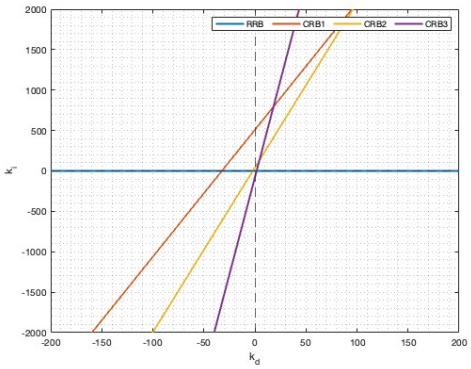
K_p	ω^* from P^{++}	ω^* from P^{+-}	ω^* from P^{-+}	ω^* from P^{--}
$K_p = 0$	4.416732	6.780666	3.865945	5.784630
$K_p = 5$	4.461086	6.849722	3.910466	5.859121
$K_p = 11$	4.513986	6.931873	3.963721	5.947846
$K_p = 22$	4.610052	7.080501	4.060839	6.108619
$K_p = 47$	4.824013	7.409201	4.278822	6.464925
$K_p = 100$	5.258265	8.068344	4.726687	7.180041
$K_p = 230$	6.225408	9.512138	5.737257	8.736910
$K_p = 512$	7.977827	12.093670	7.570970	11.472543



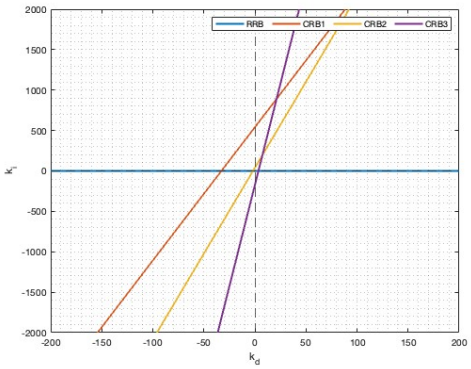
(a) Stable $K_i - K_d$ range for $K_p = 0$



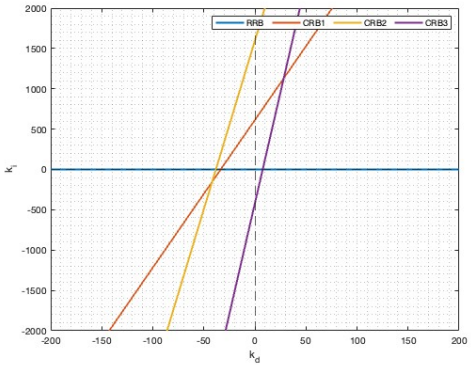
(b) Stable $K_i - K_d$ range for $K_p = 5$



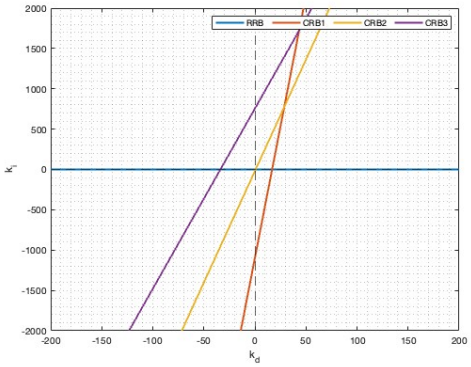
(c) Stable $K_i - K_d$ range for $K_p = 11$



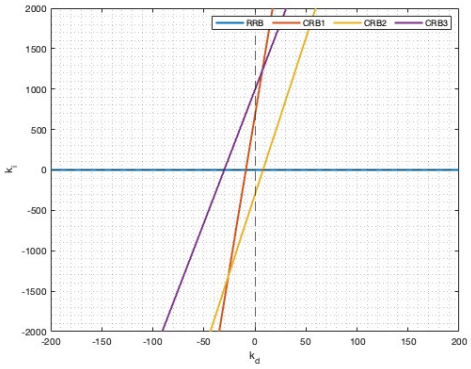
(d) Stable $K_i - K_d$ range for $K_p = 22$



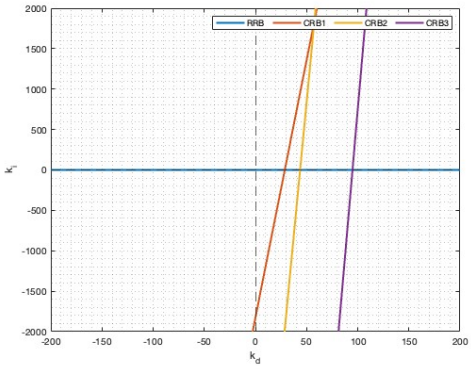
(e) Stable $K_i - K_d$ range for $K_p = 47$



(f) Stable $K_i - K_d$ range for $K_p = 100$

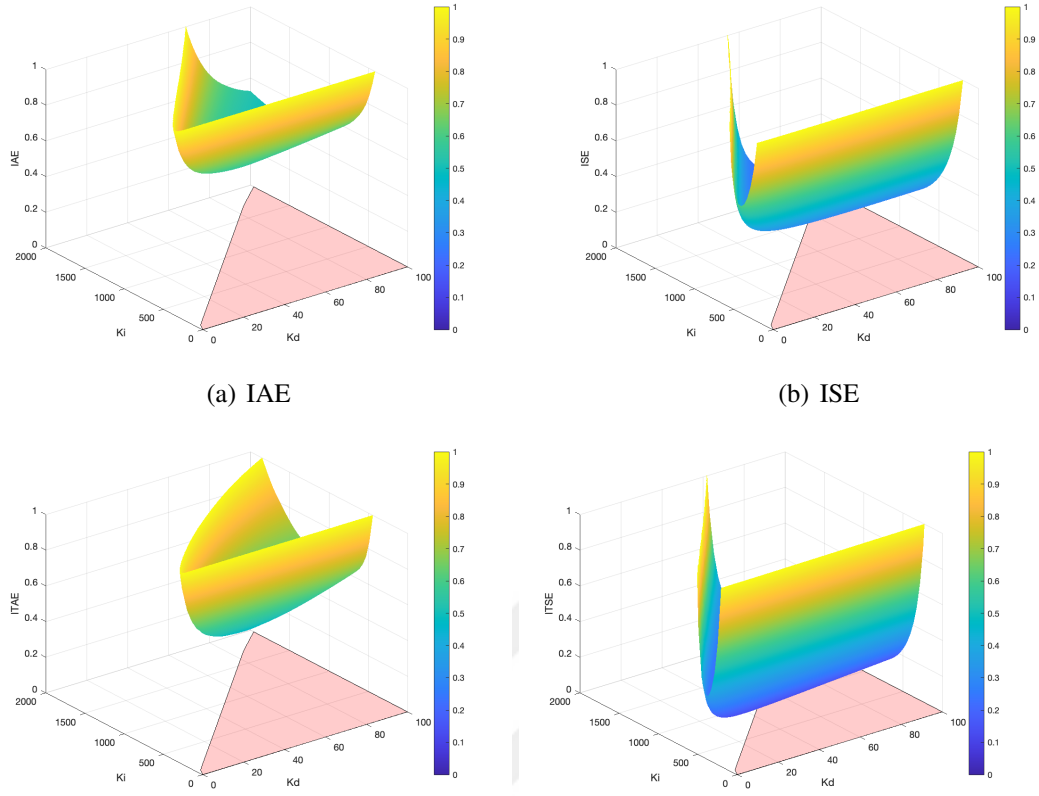


(g) Stable $K_i - K_d$ range for $K_p = 230$



(h) Stable $K_i - K_d$ range for $K_p = 512$

Figure 6.3 : Stabilizing $K_i - K_d$ ranges for different K_p values



(a) IAE (b) ISE

(c) ITAE (d) ITSE

Figure 6.4 : Error signals with respect to corresponding $K_i - K_d$ pair for $K_p = 0$

measurement as follows:

$$K_{init} = [0, 1020.1, 73.8] \tag{6.6}$$

The selected controller corresponds to $J_{IAE}(K_{init}, q_n) = 0.645$. Since the controller obtained in equation (6.6) is approved, we are ready to start with ***Algorithm 2*** and move from nominal system values to the whole uncertainty plane.

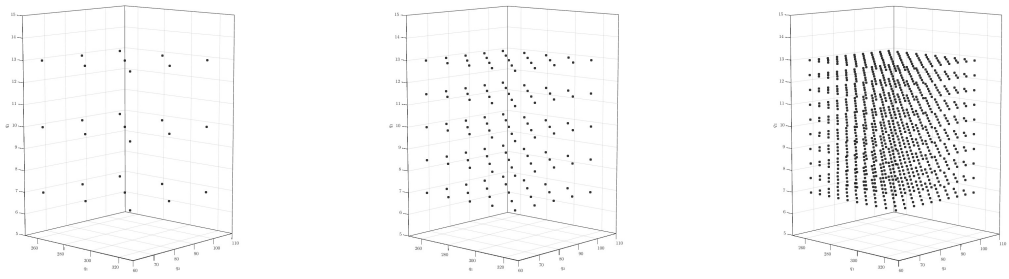
6.1.2 Robust controller design

Algorithm 2 provides the guidance to introduce the complete uncertainty range within the control design process. At the initial steps, the designer shall specify the range of the parameter box Q that encloses all uncertainty points. It is clear to us what the extreme boundaries of the system are from the beginning of the section 6.1. However, the exact number of samples to be used is not introduced in the algorithm, which makes it flexible according to the gradient descent performance in later steps. For instance, we used three different numbers of samples that correspond to the parameter boxes

shown in Figure 6.5. Moving forward, the number of samples defined in the previous step shapes the characteristic polynomial set required in step 2.

After preparing the system's uncertain parameters, we are ready to move towards the gradient descent parameters, where we need to define the learning rate and the gradient length. The gradient length $\Delta = 10^{-5}$ for the remainder of this thesis study, as it shows a very small impact on results per algorithm definition. However, different learning rates are used along with the proposed sample counts defined earlier. The results snapshot shown in Table 6.3 includes various K_{init}, learning rate (α) and total number of samples ($Q_{samples}$) used during gradient descent propagation. The reason behind using this approach is to understand the minimum number of samples required to have acceptable results using gradient descent, and to find the most suitable learning rate. From Table 6.3 we can see that the progress is determined with respect to the ISE function defined in equation (4.20).

During this study, we considered all cost functions defined in section 4.2.3 in order to view the total propagation of the proposed algorithm. The experimental data up to 50 iterations are shown in Appendix A. At each row group a different initial controller K_{init} obtained from ***Algorithm 2*** is used to start the search. Figure 6.6 shows the change in the PID controller parameters at each iteration starting from the same initial controller defined in equation (6.6) while the same gradient descent parameters are used. We can also observe that the use of different error functions led to considerable changes in some of the PID parameters (K_p and K_d).



(a) 27 Sampling Points (b) 125 Sampling Points (c) 1000 Sampling Points

Figure 6.5 : Parameter box Q density according to the defined number of samples

Table 6.3 : ***Algorithm 2*** propagation for different K_{init}, learning rates and number of samples

Parameters				Iteration									
				1		5		10		25		50	
	K_{init}	α	$Q_{samples}$	IAE	K_{pid}	IAE	K_{pid}	IAE	K_{pid}	IAE	K_{pid}	IAE	K_{pid}
K_p	0				333.709		394.5973		395.2664		319.0045		208.2009
K_i	1020.1	1	27	0.6459	996.0305	0.5304	1057.0065	0.485	1123.4746	0.4467	1265.5438	0.416	1414.7791
K_d	73.8				423.4671		318.047		262.2033		224.1593		175.5917
K_p	0				33.709		394.5973		395.2664		319.0045		208.2009
K_i	1020.1	100	125	0.6459	996.0305	0.5304	1057.0065	0.485	1123.4746	0.4467	1265.5438	0.416	1414.7791
K_d	73.8				423.4671		318.047		262.2033		224.1593		175.5917
K_p	0				333.709		394.5973		395.2664		319.0045		208.2009
K_i	1020.1	$1.5k$	$1k$	0.6459	996.0305	0.5304	1057.0065	0.485	1123.4746	0.4467	1265.5438	0.416	1414.7791
K_d	73.8				423.4671		318.047		262.2033		224.1593		175.5917
K_p	47				132.4677		129.9438		105.7272		128.0319		99.8144
K_i	980.23	100	27	0.4612	1025.6996	0.4574	1071.6869	0.4562	1121.6376	0.4326	1238.4448	0.4101	1384.6937
K_d	188.17				166.4983		117.1416		83.7907		109.0122		140.0160
K_p	100				52.9689		122.1963		97.8722		115.5033		104.7381
K_i	505	1	125	0.4821	1028.4667	0.485	1071.6869	0.4386	1128.7042	00.4455	1245.7230	0.4047	1388.3096
K_d	70.92				206.5338		117.1416		109.3554		87.9168		115.4838
K_p	250				−160.842		111.6246		322.5523		684.7429		991.2385
K_i	1010.18	$1.5k$	125	0.9805	1009.4222	1.1915	1093.0296	0.9552	1176.0285	0.7033	1363.2869	0.5644	1584.5543
K_d	63.12				1733.1486		1758.1607		1742.1221		1671.806		1534.249
K_p	22				223.8855		251.8012		229.1826		154.04		78.4152
K_i	1007.9	$50k$	$1k$	0.5298	1011.3364	0.4764	1068.4906	0.4599	1125.7829	0.4285	1251.2625	0.3995	1394.1941
K_d	201				263.9485		180.0708		164.1314		137.5018		93.5337
K_p	100				52.9689		122.1963		97.8722		115.5033		104.7381
K_i	505	$50k$	$1k$	0.4821	1028.4667	0.485	1078.2653	0.4386	1128.7042	0.4455	1245.723	0.4047	1388.3096
K_d	70.92				206.5338		81.8892		109.3554		87.9168		115.4838

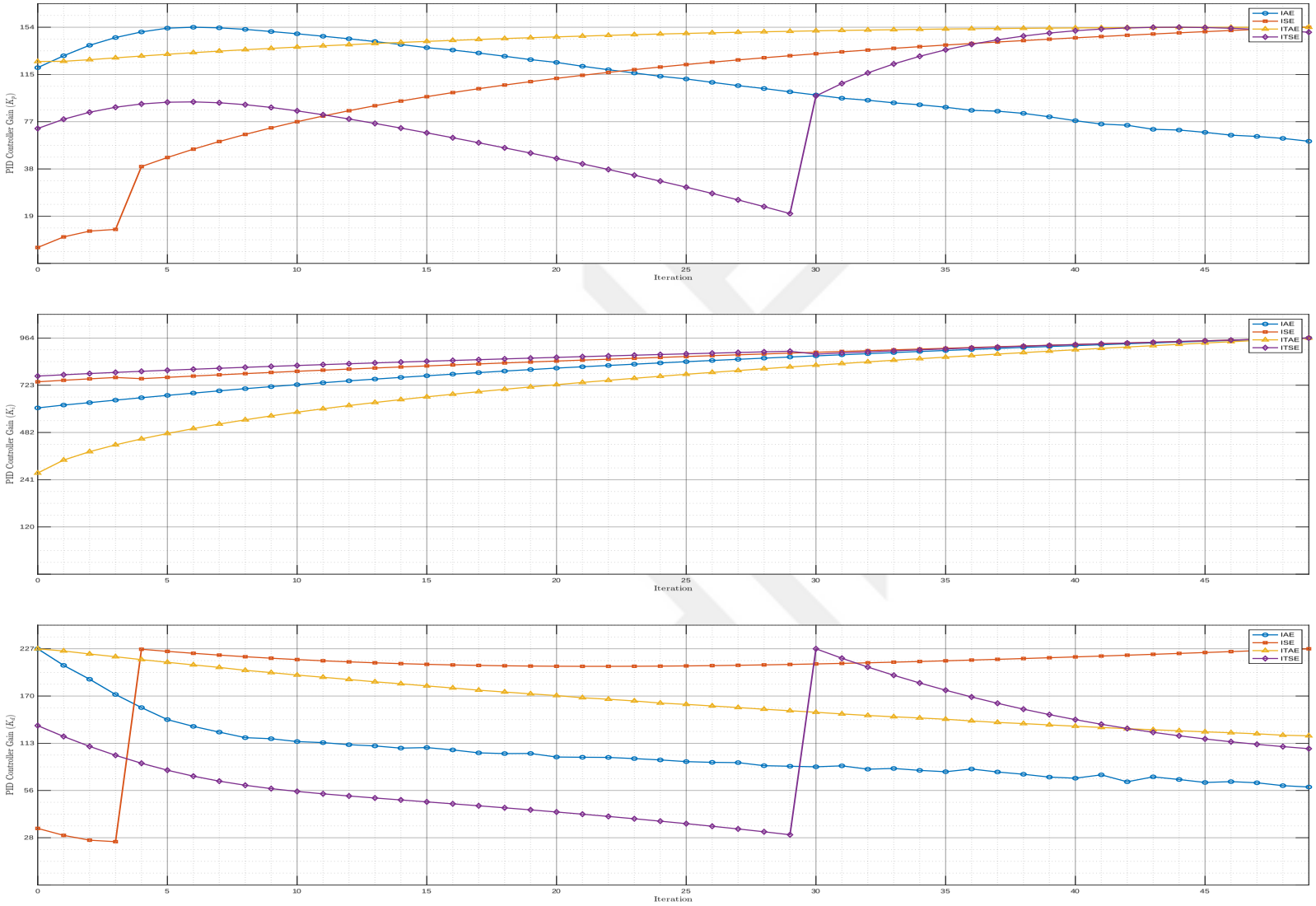


Figure 6.6 : PID Controller parameters change along with gradient descent propagation for IAE, ISE, ITAE, and ITSE functions

Table 6.4 : ***Algorithm 2*** final output based on iteration size and global level of performance

Parameters		Error Functions				Algorithm Output	
Iterations	γ	IAE	ISE	ITAE	ITSE	***Algorithm 1***	***Algorithm 2***
19	0.6	0.459	0.2055	0.5925	0.0776	0 1020.1 73.8	1696.4672 1533.45 1453.858
23	0.5	0.4454	0.2035	0.4967	0.1101	5 893.16 88.945	1535.2096 1577.5 1125.9283
10	0.45	0.4599	0.2157	0.5084	0.1001	22 1007.9 201	953.4517 1244.142 643.7874
26	0.3	0.4248	0.2044	0.3000	0.087	100 505 70.92	321.6593 1351.7616 225.092
50	0.1	0.4047	0.1934	0.6188	0.0725	100 505 70.92	1996.258 2232.7872 2499.6158

The next step in **Algorithm 2** defines the termination points for the gradient descent process obtained from the previous step. This step can be completed by reaching a third level of performance (global performance metric) or the maximum number of iterations. Table 6.4 shows the expected output according to various levels of performance and iteration sizes. It is necessary to clarify that the level of performance defined in the second column refers to the desired ITAE error function, and the output of ***Algorithm 1*** is the initial controller used to obtain the robust controller. In other words, the final output obtained from ***Algorithm 1***. We can observe from the table that the third and fifth results are obtained due to the iteration size limitation, while the other results met the performance metrics.

6.1.3 Comparison with deterministic controller

In order to challenge the proposed gradient descent based optimization approach, we compared our robust controller performance with a controller obtained from Ziegler-Nichols method using system nominal values. We used again the four error functions defined earlier in section 4.2.3 to measure the Auto-tuned (AT) PID controller. Figure 6.7 highlights that the proposed controller performance overcame the AT-PID controller for all used error measurement approaches.

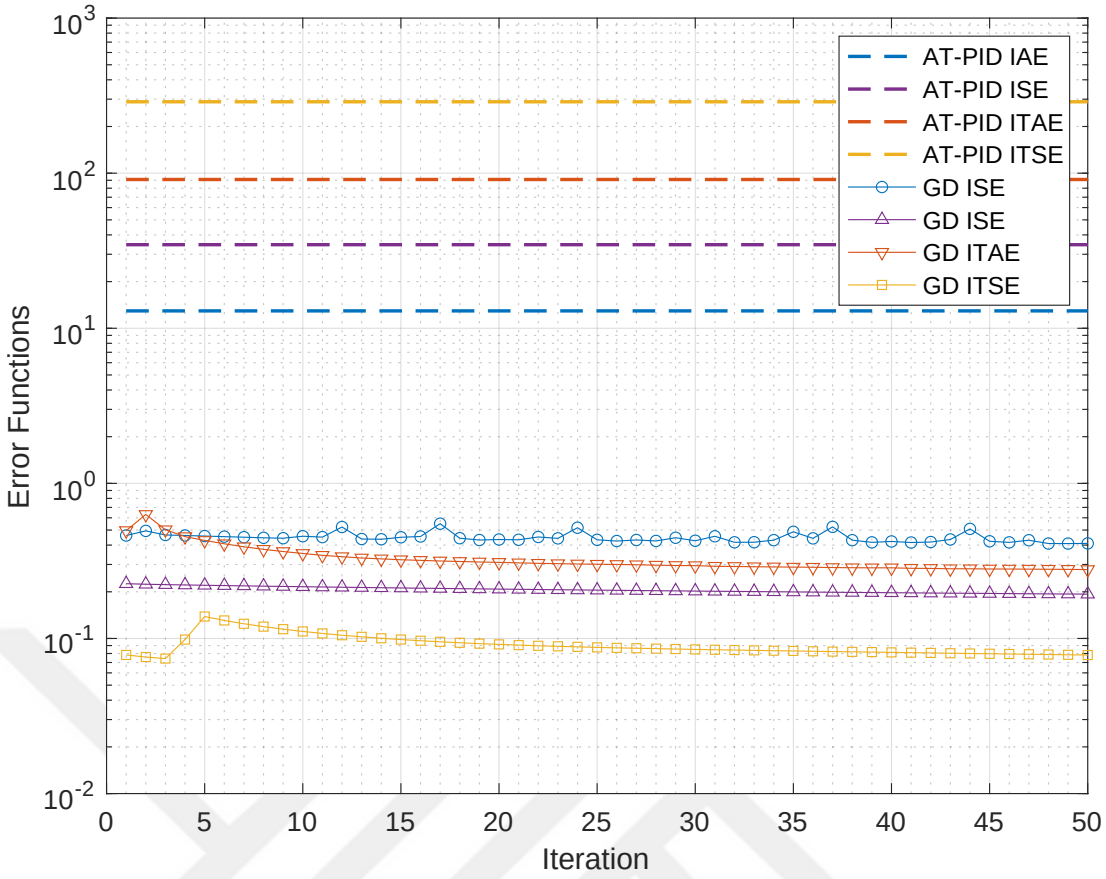


Figure 6.7 : Gradient descent based PID controller compared to AT-PID Controller

6.2 Rotary Inverted Polynomial Control

This section highlights the experimental data and simulation results obtained from controlling the rotary inverted pendulum model with the proposed semi-heuristic approaches. The model utilized for this test is linearized earlier in section 4.3.2. Linearization for the rotary inverted pendulum is done for α around 180° and θ around 0°. The control approach, on the other hand, is proposed in section 5.2, which consists of decoupled ***Algorithm 3*** and ***Algorithm 4***.

The remainder of this section is structured similarly to the previous section, in which we introduce a stable initial controller for the system's nominal parameters. Then we generalize the control strategy for the entire uncertainty region.

The uncertainty of the system is modeled at the end of chapter 4 using the parametric approach modeling type as shown in equation (4.38). We considered the pendulum

mass (m_p) our only uncertain parameter. It is necessary to clarify that the pendulum mass covers both the pendulum link mass and the sphere object mass attached at the top of the pendulum link. The nominal mass taken into account is $24gm$ while the total range of the parameter box can be visualized as a single line with ends at 18 and $30gm$. Accordingly, we can obtain the pole-spread as represented in Figure 6.8.

The system, as expected, has an unstable nature with one pole located on the right half plane, which remains on this side for all of the uncertainty region. Additionally, we have a pair of stable complex roots and a pole at the zero. While the complex pair changed with the left half plane the pole at the zero remains in its location for different values of pendulum mass.

6.2.1 Initial controller design

Similar to the previous section, ***Algorithm 3***, which is associated with obtaining the initial stable controller $K_i nit$ utilizes the system's nominal parameters. Therefore, in the first step of the algorithm, we should compute the nominal system Transfer function matrix $T(s,Q)$ that is represented as follows:

$$T(s,Q_n) = \begin{matrix} 0.3955s^2 + 1.4240\,s - 45.1092 \\ \hline 0.0295\,(s^4 + 6.8563\,s^3 - 117.3132\,s^2 - 523.9797\,s) \\ 0.3909s^2 - 1.2243 \times 10^{-9}s \\ \hline 0.0295\,(s^4 + 6.8563\,s^3 - 117.3132\,s^2 - 523.9797\,s) \end{matrix} \tag{6.7}$$

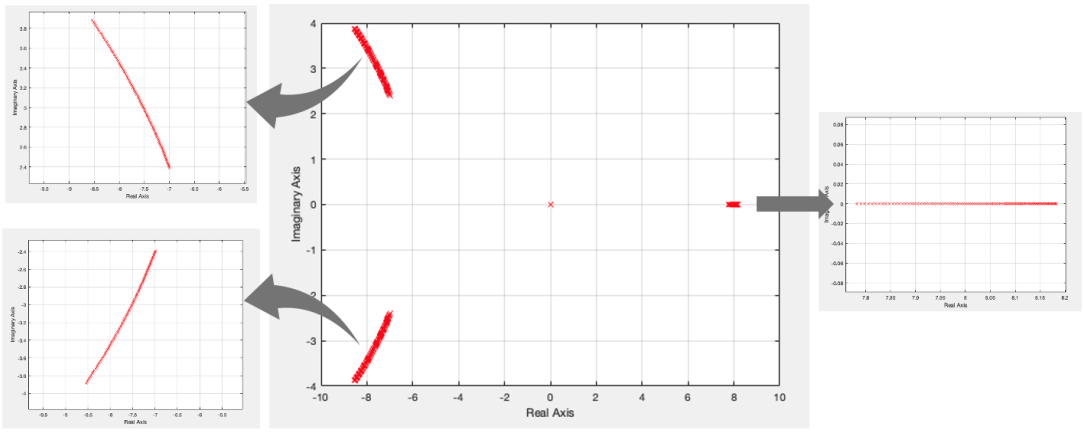


Figure 6.8 : Rotary inverted pendulum pole-spread for 1-D parameter uncertainty

Following the identification of the nominal system, we shall use the dual-PD controller shown in Figure 5.3 to stabilize the nominal system. The following parameters can be utilized for system nominal values:

$$K_{p_\theta} = -2, \quad K_{d_\theta} = -2, \quad K_{p_\alpha} = 30, \quad K_{d_\alpha} = 2 \tag{6.8}$$

Figure 6.9 shows the step response for the rotary inverted pendulum with the controller parameters obtained in the previous step. The controller stabilize the given model; however, there are considerable fluctuations in the system's transient response, which will impact our ISE calculation in later steps.

In the next step defined by ***Algorithm 3***, only one of the PD controller pair can be picked for future steps. For this thesis study, we selected the PD gains related to $alpha$ angle. The selected controller shall meet the predetermined level of performance γ, which is equal to 10 for our current cost function and system parameters. The ISE error function for the system θ and α sub-transfer functions is shown in Figure 6.10(a) and Figure 6.10(b), respectively. It is also necessary to mention that the range of $Kp - Kd$ is selected around our stabilizing controller found in the previous step. It is possible to utilize another deterministic method to narrow the search area however, we can compensate for this with a good random search technique. Adding to that, the nature of our system is stable for a small range of $K_p - K_d$ values.

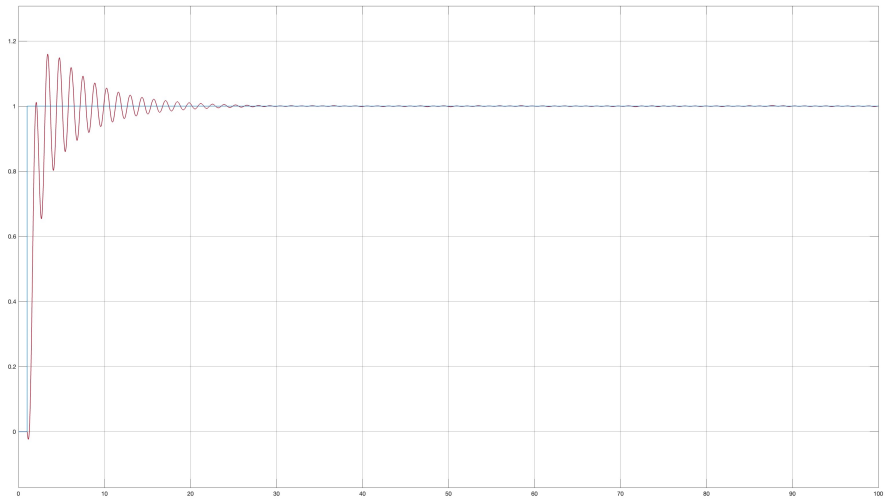


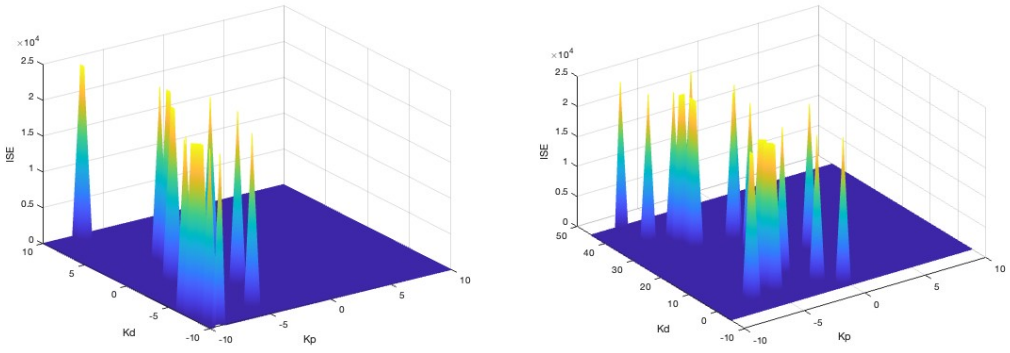
Figure 6.9 : Rotary inverted pendulum nominal system unit step response

In Figure 6.10(a) and 6.10(b), we can observe tremendous error values for multiple controller pairs in our targeted $K_p - K_d$ range. These spikes can impact the propagation of the search algorithm utilized in ***Algorithm 4*** and lead to divergence, as we can see in the next section. However, a minimal error is also available through different gains. Since the initial controller error threshold (γ) is met for the controller in equation (6.8), the current algorithm can be terminated, and we can start generalizing the search for all of the uncertainty region.

6.2.2 Adam-based controller design

Adam optimizer is the main search technique used within ***Algorithm 4*** in order to reach a robust controller for all of the uncertainty region. Prior to start the algorithm, we have to determine the total range of the uncertain parameters Q, which will be used by our cost function (ISE) in order to converge towards robustness. Since we have one uncertain parameter m_p that spans for small range (0.018 - 0.03), we selected 25 samples within Q to be utilized for global cost calculation.

Figure 6.11 and Figure 6.12 represent the change in K_p and K_d, respectively, starting from the initial controller developed in the previous section. Multiple learning rates and gradient lengths have been used for obtaining the results shown in the figures. The algorithm shows similar results while approaching the final robust controller parameters, however, it needed to be guided when it encountered misleading spikes shown in Figure 6.10. The cost function propagation for different initial controllers is shown in Figure 6.13. We can see the impact of the misleading spikes on the



(a) ISE for θ sub-transfer function (b) ISE for α sub-transfer function

Figure 6.10 : ISE for rotary inverted pendulum nominal system

algorithm propagation. However, the corrective steps taken during the study helped the algorithm to reconsider its previous direction and apply different steps that can lead to convergence. Despite these actions in some of the cases ($K_\alpha = 30, \quad \alpha = 0.75$), the algorithm error rate is still increasing. However, for other cases where the algorithm parameters are initialized correctly a good results have been obtained as shown in Table 6.5. With the robust controller error threshold defined as 8, we can terminate the algorithm since the desired performance is reached.

Table 6.5 : ***Algorithm 4*** progress for selected learning rate and gradient length

Parameters		Iterations					
		1		25		50	
α	β	PD	ISE	PD	ISE	PD	ISE
0.1	0.9	30 2.5	10	30.52 3.034	7.38	30.44 2.94	7.43
0.7	0.75	30 2.5	10	31.83 4.33	9.15	30.561 3.06	7.38

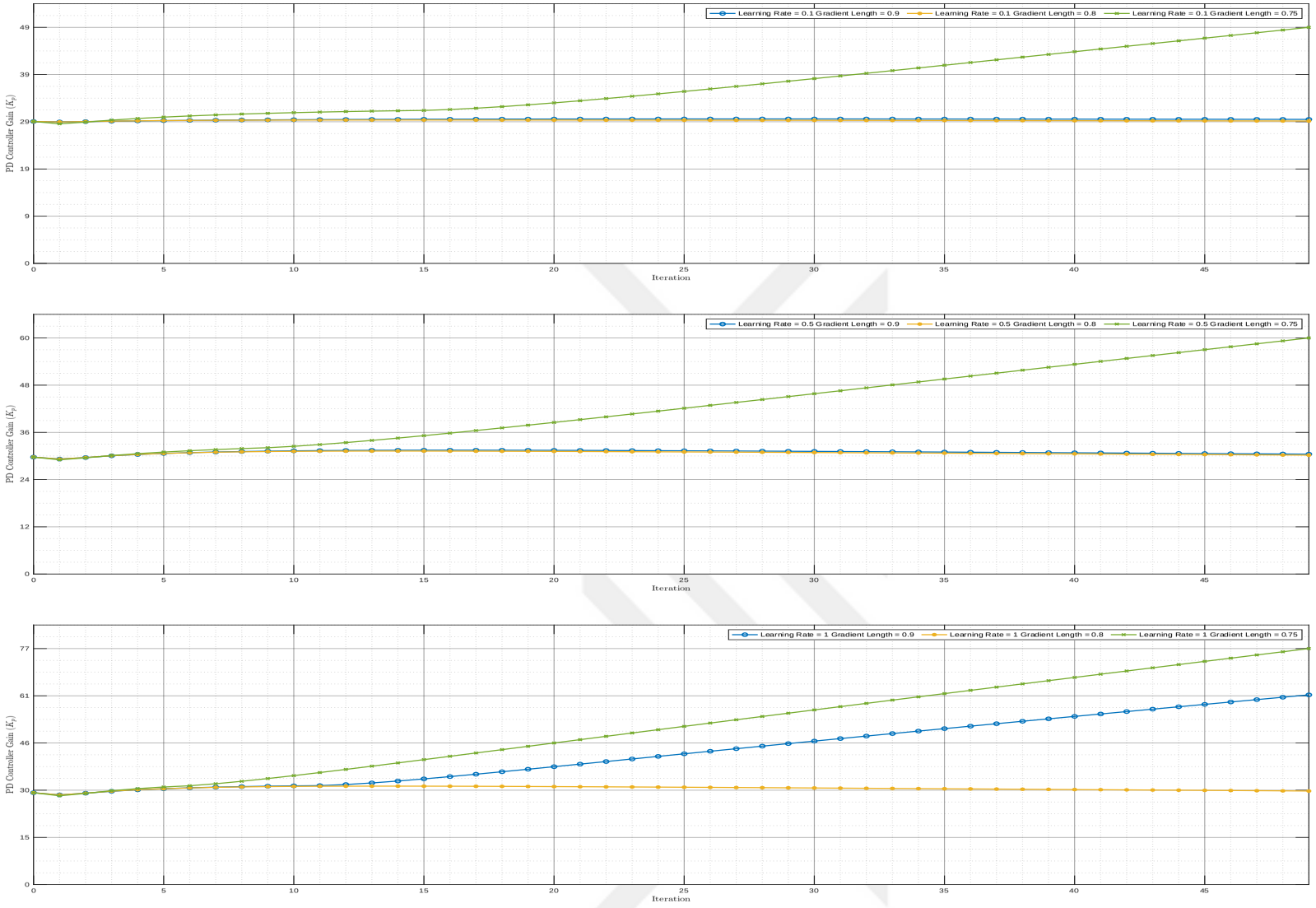


Figure 6.11 : K_p gain change along with Adam propagation for selected learning rates

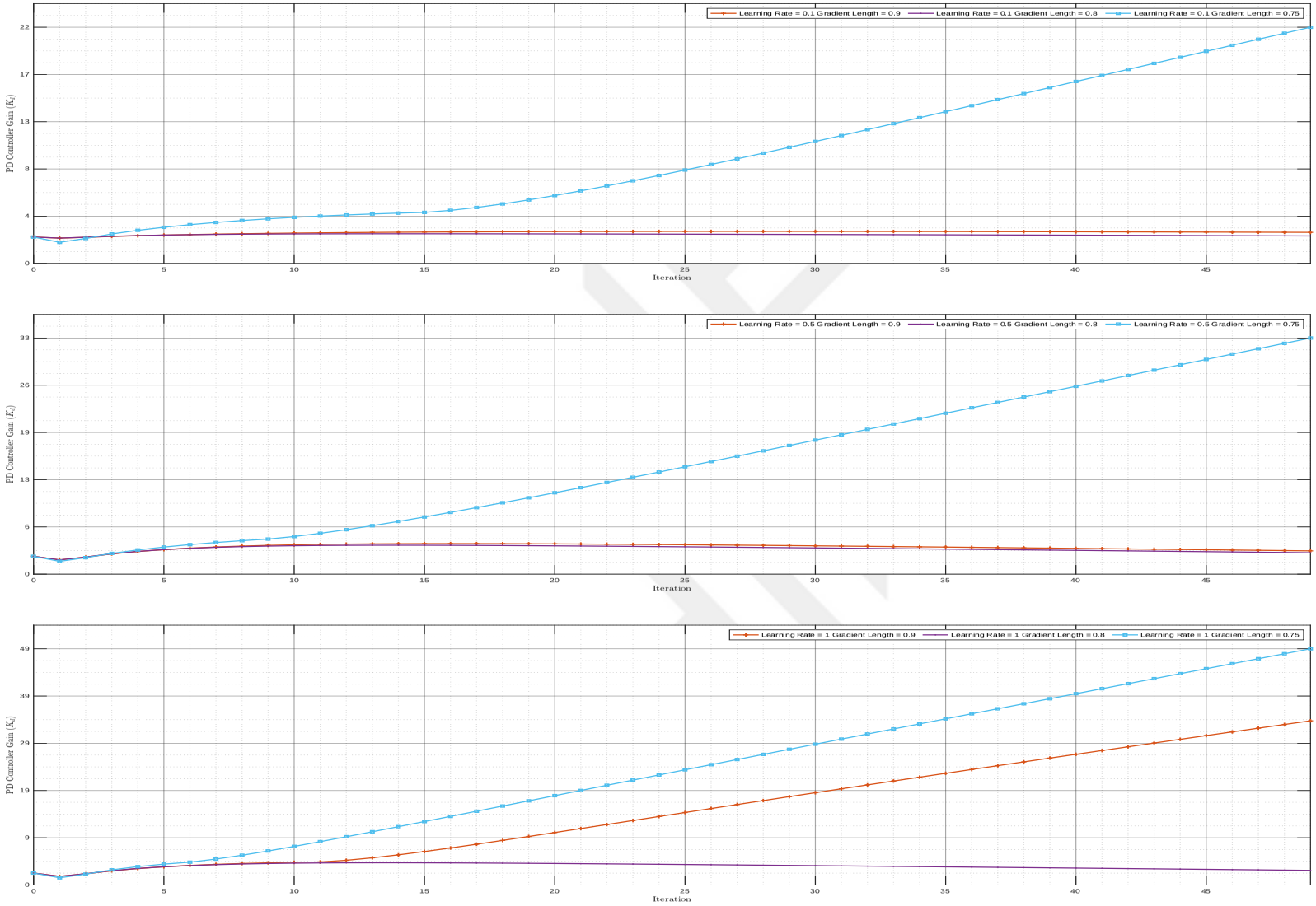


Figure 6.12 : K_d gain change along with Adam propagation for selected learning rates

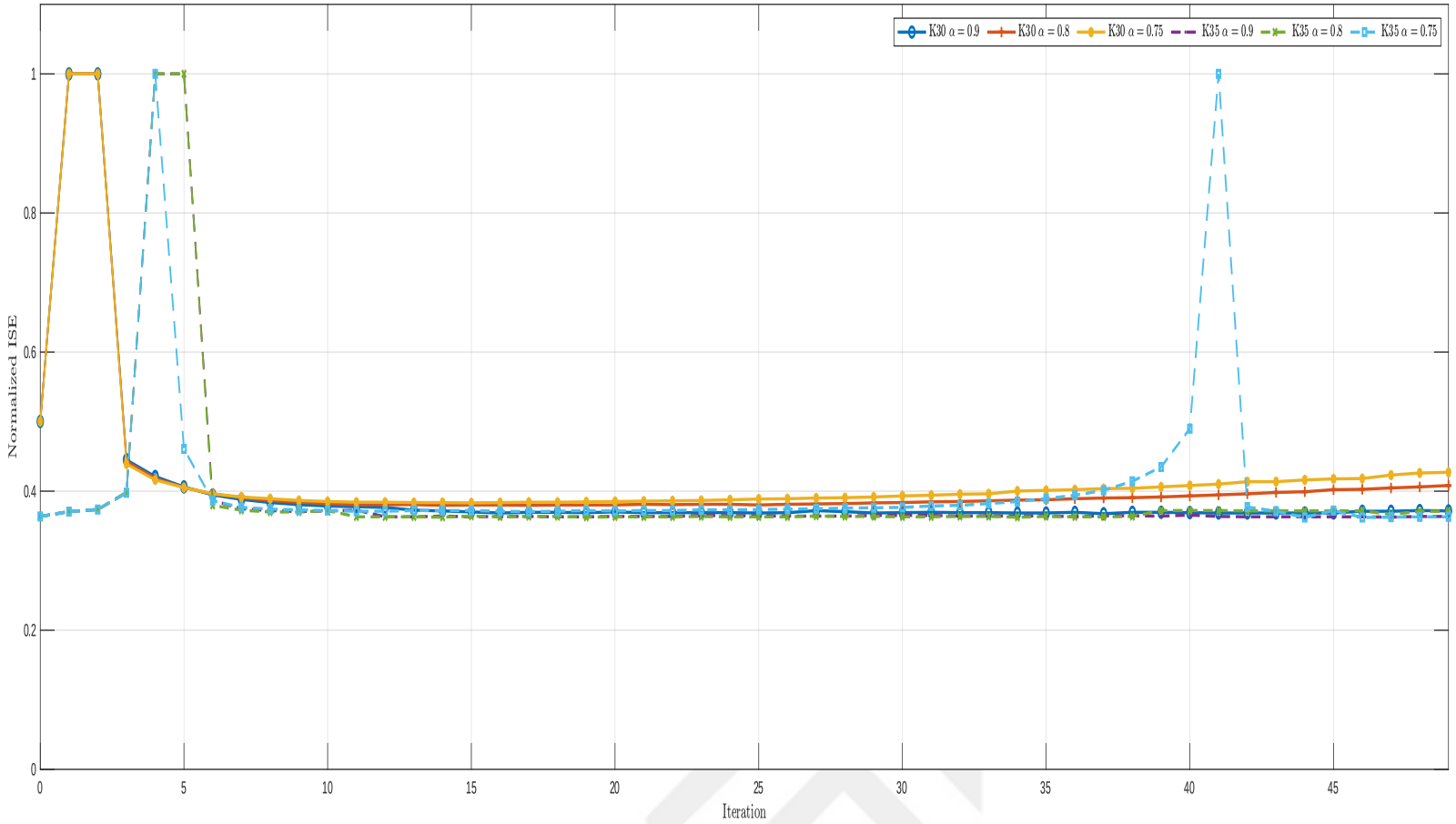


Figure 6.13 : Normalized ISE error for ***Algorithm 4*** progress for different initial PD controllers

7. DISCUSSION AND CONCLUSIONS

System stability and model performance are crucial components during the design of controllers. Depending on the design problem, a moderate point between these two objectives is reached by the designer. Assuming the system parameters remain constant and do not fluctuate, we can derive a rigid mathematical model for the system and the controller used to stabilize the system. We can even achieve this goal using a common controller topology. Nonetheless, this straightforward modeling and controller design is not the typical scenario in practice, where we encounter variations in the system model due to several reasons as shown in section 2.1.1. These variations introduce system uncertainty, which must be incorporated into system modeling and considered during controller design via one of the uncertainty modeling methodologies shown in section 2.1.3. The presence of uncertain parameters in the system to be controlled further complicates the trade-off between maintaining system stability and performance at a satisfactory level.

While many control theories offer deterministic solutions to address this challenge, such approaches are frequently computationally intensive. Alternatively, a modern approach employs random optimization, which does not necessitate a definitive mathematical equation to describe the relevant system. Both of them have several drawbacks that can block the controller designing process, which are shown explicitly in chapter 3. The same chapter initiates the necessary ground to introduce the semi-heuristic optimization approaches that can be considered as a moderate solution for deterministic and stochastic control strategies.

In this thesis study we have proposed two semi-heuristic approaches that address control problems in uncertainty presence. The uncertainty modeling for the systems desired to be controlled is done using the parametric approach, as shown in chapter 4. Each model provided a total description of system parameters using a transfer

function/transfer function matrix in the frequency domain. The proposed approaches utilized a decoupled algorithm pair; the first algorithm in each finds the stabilizing controller for nominal system parameters, while the second algorithm propagates gradually towards a robust controller considering a sampled range of the uncertainty region. Deterministic control topology is applied on the system's nominal values in the first algorithm within each approach to extract an initial stable controller, then a random optimization technique is utilized by the second algorithm to obtain robust controller parameters starting from where the first algorithm is concluded. Predetermined levels of performance are used to smooth the transition between the algorithms pair and to ensure that a global level of performance is met.

The gradient descent based optimization approach was tested on an interval polynomial 5^{th} order system in section 6.1. ***Algorithm 1*** of the proposed approach computed an initial stable controller using IAE as cost function and the PID controller topology. Within its step, Munro-Soylemez's method is utilized with Kharitonov's theorem to obtain the stabilizing gain ranges, which speed the convergence process significantly. Then ***Algorithm 2*** utilized the derived initial controller to propagate towards a robust solution using the gradient descent algorithm. A good result was obtained for all error functions described in section 4.2.3 using various learning rates and gradient lengths. In order to show the derived controller's performance, we compared it with the Auto-tuned PID controller and showed that the AT-PID controller provides poor performance if generalized for all uncertainty regions, unlike our design controller.

Although the results obtained from the gradient descent-based approach are interesting and the approach is easy to implement, there are a couple of limitations that can constrain the overall picture. One of these limitations is ***Algorithm 1*** exclusive compatibility with systems with an interval polynomial characteristic equation. In case of considering an affine linear or multi-linear characteristic polynomial system, the algorithm may require different approaches to determine the stable $K_i - K_d$ range, such as the generalized Kharitonov theorem. Another limitation is the random search technique itself. The gradient descent can get stuck on local critical points or diverge

according to the selected learning rate. Therefore, enhanced adaptive gradient descent can be used to improve the learning rate instead of using a fixed value, as well as implementing a stochastic gradient descent.

The other introduced approach is implemented on the rotary inverted pendulum problem in section 6.2. The first algorithm of this approach (***Algorithm 3***) used a dual-PD controller to stabilize the rotary inverted pendulum system when the nominal pendulum mass is considered. In spite of the considerable chances of driving the system into instability -as shown in the Figure 6.10-, ***Algorithm 4*** managed to converge into a robust controller parameter that meets the desired level of performance using Adam optimizer. The rotary inverted pendulum problem is known for its unstable and non-linear behavior, making it suitable to challenge the performance of semi-heuristic approaches in robust controller design. Despite we needed to perform corrective actions for the second algorithm, a valid result is obtained at the end, as shown in the previous chapter.

Finally, we would like to highlight that this thesis study proves the effectiveness and the practicality of the semi-heuristic approaches tackling control problems with parameter uncertainty, which introduces these approaches as a valid solution for the limitations arising in deterministic and stochastic approaches.

REFERENCES

[1] **Bhattacharyya, S.P. and Keel, L.H.**, (1995). Robust control: the parametric approach, Advances in control education 1994, Elsevier, pp.49–52.

[2] **Sun, N.**, **Liang, D.**, **Wu, Y.**, **Chen, Y.**, **Qin, Y. and Fang, Y.** (2019). Adaptive control for pneumatic artificial muscle systems with parametric uncertainties and unidirectional input constraints, *IEEE Transactions on Industrial Informatics*, *16*(2), 969–979.

[3] **Wang, S. and Han, K.** (2018). Stochastic response analysis for nonlinear vibration systems with adjustable stiffness property under random excitation, *PloS one*, *13*(8), e0200922.

[4] **Benner, P.**, **Ohlberger, M.**, **Cohen, A. and Willcox, K.** (2017). *Model reduction and approximation: theory and algorithms*, SIAM.

[5] **Wang, F.Y. and Liu, D.** (2008). *Networked control systems*, Springer.

[6] **Petersen, I.R. and Tempo, R.** (2014). Robust control of uncertain systems: Classical results and recent developments, *Automatica*, *50*(5), 1315–1335.

[7] **Zhou, K.**, **Doyle, J.C. and Glover, K.** (1998). *Essentials of Robust Control*, Prentice Hall, Upper Saddle River, NJ, USA.

[8] **Doyle, J.C.**, **Glover, K.**, **Khargonekar, P.P. and Francis, B.A.** (1992). Robust and Optimal Control, *Prentice-Hall*.

[9] **Packard, A. and Doyle, J.** (1993). Structured singular value theory: connections with state-space robustness analysis, *Automatica*, *29*(1), 71–109.

[10] **Bartlett, A.C.**, **Rao, C.R. and Wilson, D.C.** (1988). Root locations of polynomials satisfying bivariate polynomial constraints: an edge theorem, *IEEE Transactions on Automatic Control*, *33*(10), 985–988.

[11] **Megretski, A. and Rantzer, A.** (1997). System analysis via Integral Quadratic Constraints, *IEEE Transactions on Automatic Control*, *42*(6), 819–830.

[12] **Rajamäki, J.** (2018). Random Search Algorithms for Optimal Control.

[13] **Zabinsky, Z.B.** *et al.* (2009). Random search algorithms, *Department of Industrial and Systems Engineering, University of Washington, USA*, 34.

[14] **Tempo, R.**, **Calafiore, G.**, **Dabbene, F.** *et al.* (2013). *Randomized algorithms for analysis and control of uncertain systems: with applications*, volume 7, Springer.

[15] **Mustapha, A.**, **Mohamed, L. and Ali, K.** (2020). An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field, *Smart Applications and Data Analysis: Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26, 2020, Proceedings 3*, Springer, pp.349–359.

[16] **Zhang, G.**, **Martens, J. and Grosse, R.B.** (2019). Fast convergence of natural gradient descent for over-parameterized neural networks, *Advances in Neural Information Processing Systems*, *32*.

[17] **Bottou, L.**, **Curtis, F.E. and Nocedal, J.** (2018). Optimization methods for large-scale machine learning, *SIAM review*, *60*(2), 223–311.

[18] **Boyd, S.P. and Vandenberghe, L.** (2004). *Convex optimization*, Cambridge university press.

[19] **Betts, J.T.** (2010). *Practical methods for optimal control and estimation using nonlinear programming*, SIAM.

[20] **Yu, T. and Zhu, H.** (2020). Hyper-parameter optimization: A review of algorithms and applications, *arXiv preprint arXiv:2003.05689*.

[21] **Linkens, D.A.** (1990). AI in control systems engineering, *The Knowledge Engineering Review*, *5*(3), 181–214.

[22] **Meyn, S.** (2022). *Control systems and reinforcement learning*, Cambridge University Press.

[23] **Zadeh, L.A.**, (2023). Fuzzy logic, Granular, fuzzy, and soft computing, Springer, pp.19–49.

[24] **Kashani, A.R.**, **Camp, C.V.**, **Rostamian, M.**, **Azizi, K. and Gandomi, A.H.** (2022). Population-based optimization in structural engineering: a review, *Artificial Intelligence Review*, 1–108.

[25] **Wang, Q.**, **Spronck, P. and Tracht, R.** (2003). An overview of genetic algorithms applied to control engineering problems, *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, volume 3, IEEE, pp.1651–1656.

[26] **Bodenhofer, U.** (2003). Genetic algorithms: theory and applications, *Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter*, *2004*.

[27] **Gad, A.G.** (2022). Particle swarm optimization algorithm and its applications: a systematic review, *Archives of computational methods in engineering*, *29*(5), 2531–2561.

[28] **Ekinci, S.**, **Izci, D. and Hekimoğlu, B.** (2021). Optimal FOPID speed control of DC motor via opposition-based hybrid manta ray foraging optimization and simulated annealing algorithm, *Arabian Journal for Science and Engineering*, *46*(2), 1395–1409.

[29] **Masoumian, A.** *et al.* (2017). Design and Analysis of a Fuzzy Control System for an Inverted Pendulum, *International Journal of Engineering*, *30*(5), 751–759.

[30] **de Carvalho, A.**, **Justo, J.F.**, **Angélico, B.A.**, **de Oliveira, A.M. and da Silva Filho, J.I.** (2021). Rotary inverted pendulum identification for control by paraconsistent neural network, *IEEE Access*, *9*, 74155–74167.

[31] **Abut, T. and Soyguder, S.** (2022). Two-loop controller design and implementations for an inverted pendulum system with optimal self-adaptive fuzzy-proportional–integral–derivative control, *Transactions of the Institute of Measurement and Control*, *44*(2), 468–483.

[32] **Chawla, I. and Singla, A.** (2021). Real-time stabilization control of a rotary inverted pendulum using LQR-based sliding mode controller, *Arabian Journal for Science and Engineering*, *46*(3), 2589–2596.

[33] **Huynh, P.H.**, **Nguyen, M.H.**, **Pham, N.P.**, **Duong, H.V.P.**, **Nguyen, H.H.**, **Le, D.C.**, **Nguyen, M.K.**, **Bui, N.L.**, **Le, N.P.L. and Nguyen, V.D.H.** (2024). Model Predictive Control for Rotary Inverted Pendulum: Simulation and Experiment, *Journal of Fuzzy Systems and Control*, *2*(3), 215–222.

[34] **Nguyen, V.D.**, **Vo, M.T.**, **Tran, M.D.**, **Dang, Q.D.**, **Nguyen, V.D.H.**, **Nguyen, T.D.**, **Le, T.H.L.**, **Nguyen, T.M.N.**, **Nguyen, T.V.** *et al.* (2023). Trajectory Tracking and Stabilization Control of Rotary Inverted Pendulum based on LQR and LQT techniques: Simulation and Experiment, *Journal of Technical Education Science*, *18*(1), 1–11.

[35] **Humaidi, A.J.**, **Hameed, M.R.**, **Hasan, A.F.**, **Al-Obaidi, A.S.M.**, **Azar, A.T.**, **Ibraheem, I.K.**, **Al-Dujaili, A.Q.**, **Al Mhdawi, A.K. and Abdulmajeed, F.A.** (2023). Algorithmic design of block backstepping motion and stabilization control for segway mobile robot, 557–607.

[36] **Spong, M.W.** (2005). Underactuated mechanical systems, 135–150.

[37] **Wang, S.**, **Tao, L.**, **Chen, Q.**, **Na, J. and Ren, X.** (2020). USDE-based sliding mode control for servo mechanisms with unknown system dynamics, *IEEE/ASME Transactions on Mechatronics*, *25*(2), 1056–1066.

[38] **Gambhire, S.**, **Kishore, D.R.**, **Londhe, P. and Pawar, S.** (2021). Review of sliding mode based control techniques for control system applications, *International Journal of dynamics and control*, *9*, 363–378.

[39] **Zames, G.** (1981). Feedback, optimal sensitivity, and plant uncertainty via multiplicative seminorms, *IFAC Proceedings Volumes*, *14*(2), 1171–1175.

[40] **Jiao, J.**, **Trentelman, H.L. and Camlibel, M.K.** (2019). A suboptimality approach to distributed linear quadratic optimal control, *IEEE Transactions on Automatic Control*, *65*(3), 1218–1225.

[41] **Petersen, I.R.**, **McFarlane, D.C. and Rotea, M.A.** (1998). Optimal guaranteed cost control of discrete-time uncertain linear systems, *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, *8*(8), 649–657.

[42] **Goldberg, D.E.** (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[43] **Kennedy, J. and Eberhart, R.** (1995). Particle swarm optimization, *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE, pp.1942–1948.

[44] **Kirkpatrick, S.**, **Gelatt, C.D. and Vecchi, M.P.** (1983). Optimization by simulated annealing, *Science*, *220*(4598), 671–680.

[45] **Hansen, N. and Ostermeier, A.** (2001). Completely derandomized self-adaptation in evolution strategies, *Evolutionary Computation*, *9*(2), 159–195.

[46] **Rios, L.M. and Sahinidis, N.V.** (2013). Derivative-free optimization: A review of algorithms and comparison of software implementations, *Journal of Global Optimization*, *56*(3), 1247–1293.

[47] **Söylemez, M.T.**, **Munro, N. and Baki, H.** (2003). Fast calculation of stabilizing PID controllers, *Automatica*, *39*(1), 121–126.

[48] **Munro, N. and Soylemez, M.** (2000). Fast calculation of stabilizing PID controllers for uncertain parameter systems, *IFAC Proceedings Volumes*, *33*(14), 549–554.

[49] **Anderson, B.**, **Jury, E. and Mansour, M.** (1987). On robust Hurwitz polynomials, *IEEE Transactions on Automatic Control*, *32*(10), 909–913.

[50] **Ruder, S.** (2016). An overview of gradient descent optimization algorithms, *arXiv preprint arXiv:1609.04747*.

[51] **Kingma, D.P.** (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

[52] **Quanser Inc.** (2024). *QUBE-Servo 2*, `https://www.quanser.com/products/qube-servo-2/`, accessed: 2025-05-28.

APPENDICES

APPENDIX A : Gradient Descent based Optimization Experimental Results Tables

APPENDIX A : Gradient Descent based Optimization Experimental Results Tables

In this part of the thesis, extensive data obtained from a gradient descent-based optimization approach for up to 50 iterations is shown. The tables are associated with the error functions defined in section 4.2.3 as follows:

- Table A.1 - Table A.5: covers the IAE error based progress.
- Table A.6 - Table A.10: covers the ISE error based progress.
- Table A.11 - Table A.15: covers the ITAE error based progress.
- Table A.16 - Table A.20: covers the ITSE error based progress.

The initial controller k_init for each row group in this stable is the same since the algorithm defined this controller (***Algorithm 1***) is only dependent on IAE error as shown in Chapter 5.

Table A.1 : ***Algorithm 2*** progress while using IAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 1-10)

Parameter	Iteration									
	1	2	3	4	5	6	7	8	9	10
k_p	333.7090	353.9508	371.8207	385.0007	394.5973	401.0285	402.5894	401.5648	398.9690	395.2664
k_i	996.03	1013.13	1027.63	1042.676	1057	1071.16	1084.7	1098.24	1111.32	1123.47
k_d	423.4671	393.7430	368.8790	341.5159	318.0470	296.4916	284.3321	274.0536	264.1315	262.2033
IAE	0.6459	0.6024	0.5739	0.5509	0.5304	0.5144	0.5016	0.4955	0.4898	0.485
k_p	304.2746	325.2371	341.6426	354.8100	363.6222	365.2103	363.0252	359.8189	355.2251	349.7075
k_i	999.54	1015.97	1030.95	1044.9	1058.85	1073.022	1086.3	1099.199	1111.86	1123.3
k_d	383.4989	350.8948	324.1844	301.4930	275.8572	257.3453	247.9897	242.4030	234.1932	229.9601
IAE	0.6143	0.5915	0.5583	0.5344	0.5155	0.5005	0.4917	0.4867	0.4822	0.4781
k_p	272.4177	293.7098	308.3598	316.5744	316.5272	312.6183	308.4391	303.2242	298.9616	292.5055
k_i	1004.1084	1019.9685	1034.3492	1049.2591	1063.0136	1076.0900	1088.6945	1100.8105	1112.5535	1123.4945
k_d	335.2283	297.4036	267.5463	239.0657	223.9325	216.7630	210.4080	211.0103	203.1332	200.8919
IAE	0.5807	0.5729	0.5344	0.5094	0.4929	0.4854	0.4807	0.4766	0.4733	0.4696
k_p	223.8855	245.6087	256.8311	256.4234	251.8012	247.3790	243.5617	239.5945	233.2172	229.1826
k_i	1011.3364	1026.4962	1041.8781	1056.0741	1068.4906	1080.4990	1092.1096	1103.4727	1114.6015	1125.7829
k_d	263.9485	223.8386	192.1409	181.6496	180.0708	178.1498	176.3154	168.7712	168.0338	164.1314
IAE	0.5298	0.5385	0.4989	0.4819	0.4764	0.4730	0.4697	0.4666	0.4631	0.4599
k_p	132.4677	153.6914	148.1065	139.8500	129.9438	123.4850	116.2049	109.7305	86.3969	105.7272
k_i	1025.6996	1038.1409	1049.6539	1060.7904	1071.6869	1082.2929	1092.7470	1103.2276	1111.5166	1121.6376
k_d	166.4983	129.7339	122.4542	115.7772	117.1416	113.1232	107.5576	94.8703	129.4190	83.7907
IAE	0.4612	0.4939	0.4648	0.4614	0.4574	0.4534	0.4497	0.4457	0.4430	0.4562
k_p	52.9689	113.2001	123.7937	89.6384	122.1963	69.1791	119.4665	121.4944	111.5239	97.8722
k_i	1028.4667	1044.4716	1057.5201	1065.8576	1078.2653	1084.6567	1099.5306	1109.4717	1119.8854	1128.7042
k_d	206.5338	134.6140	92.2703	146.5291	81.8892	199.3788	125.9074	106.5132	100.5191	109.3554
IAE	0.4821	0.6103	0.4670	0.4611	0.4850	0.4751	0.5649	0.4495	0.4429	0.4386

Table A.2 : ***Algorithm 2*** progress while using IAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 11-20)

Parameter	Iteration									
	11	12	13	14	15	16	17	18	19	20
k_p	391.4507	387.2298	382.9032	378.1979	373.1608	367.7796	363.7200	358.6277	353.1409	347.3947
k_i	1135.3688	1146.81	1158.195	1168.57	1178.83	1188.62	1198.24	1207.49	1216.739	1225.643
k_d	257.0953	255.2577	251.4960	249.2746	245.3437	246.3300	242.1418	236.8395	235.5607	235.7650
IAE	0.4816	0.4778	0.4747	0.4716	0.4689	0.4662	0.4638	0.4615	0.4590	0.4568
k_p	343.5128	337.7055	332.3583	327.4078	322.6283	316.7752	311.0391	305.5125	300.3698	294.4649
k_i	1134.36	1145.08	1155.45	1165.69	1175.94	1185.8	1194.63	1203.4	1211.73	1219.945
k_d	229.3482	228.5586	227.9137	224.6975	220.2798	215.6823	214.9769	214.8588	211.666	211.4957
IAE	0.4748	0.4717	0.4688	0.4660	0.4631	0.4600	0.4574	0.4551	0.4530	0.4509
k_p	285.9869	279.5516	273.0349	268.2075	262.2398	255.7456	252.5630	248.7070	243.7070	239.0363
k_i	1133.7252	1143.7205	1153.4798	1163.1525	1172.3243	1181.1741	1189.9134	1198.7708	1207.2727	1215.8298
k_d	199.8254	197.3302	197.0522	191.5880	186.8983	192.8988	188.1039	185.9530	182.5114	180.8342
IAE	0.4663	0.4633	0.4605	0.4578	0.4551	0.4527	0.4508	0.4487	0.4466	0.4444
k_p	222.0924	216.1056	214.5368	205.4683	200.5788	195.9668	189.7273	187.0673	180.1690	173.2969
k_i	1135.8843	1145.6494	1156.1334	1164.6237	1173.5716	1182.1580	1190.5905	1199.0265	1206.9524	1214.9035
k_d	162.2769	166.7389	155.9662	159.2550	157.9322	152.7559	156.5710	148.1258	145.0872	143.3838
IAE	0.4568	0.4539	0.4520	0.4496	0.4468	0.4447	0.4427	0.4408	0.4388	0.4366
k_p	66.0385	115.9849	111.6846	87.9113	108.5899	60.1908	114.5818	128.5291	110.6683	117.9753
k_i	1128.1964	1138.4668	1147.9014	1155.2859	1164.5009	1169.6506	1180.3077	1188.2038	1195.4306	1204.0902
k_d	170.0716	113.2291	96.1817	131.4473	80.6264	198.3516	140.2249	109.2376	129.2005	95.9322
IAE	0.4503	0.5232	0.4377	0.4362	0.4492	0.4544	0.5496	0.4431	0.4313	0.4345
k_p	102.4889	81.0020	97.8759	56.9234	110.3804	110.0198	74.2949	110.6641	63.6126	119.1824
k_i	1138.0121	1145.5780	1155.2325	1160.8943	1171.0828	1180.3733	1186.1664	1196.4210	1201.2041	1209.9648
k_d	93.4699	123.7458	80.4259	172.0690	109.8083	89.8584	150.1259	83.3219	191.3564	140.8975
IAE	0.4387	0.4350	0.4461	0.4462	0.5284	0.4305	0.4373	0.4726	0.4474	0.5331

Table A.3 : ***Algorithm 2*** progress while using IAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 21-30)

Parameter	Iteration									
	21	22	23	24	25	26	27	28	29	30
k_p	342.6367	336.0676	330.1297	324.6320	319.0045	314.4125	308.6508	303.0449	298.1083	292.4839
k_i	1234.545	1242.56	1250.397	1258.027	1265.54	1272.986	1280.2	1287.198	1294.20	1301.08
k_d	229.3784	228.9583	228.7006	226.5642	224.1593	221.1560	219.8245	219.3659	213.8588	212.8538
IAE	0.4546	0.4525	0.4504	0.4485	0.4467	0.4448	0.4433	0.4415	0.4399	0.4382
k_p	289.7055	285.7417	280.4253	277.8659	271.6091	266.2039	261.9133	255.8215	247.5584	244.3185
k_i	1228.15	1236.2	1243.898	1251.87	1259.27	1266.39	1273.73	1280.6	1286.83	1293.92
k_d	213.8088	207.9844	211.1867	201.1543	200.1540	198.5446	193.7647	188.4229	192.0482	183.9084
IAE	0.4489	0.4474	0.4454	0.4439	0.4420	0.4402	0.4385	0.4366	0.4350	0.4333
k_p	234.6314	226.3505	222.3723	217.0279	209.7674	205.9997	202.0041	201.3252	196.1762	192.0693
k_i	1224.3369	1231.9253	1239.6734	1247.2427	1253.7401	1260.7416	1267.5390	1274.8504	1281.5542	1288.231
k_d	173.7728	177.5292	172.7198	165.9823	168.7721	165.8467	169.5821	162.5968	161.8428	158.3483
IAE	0.4424	0.4403	0.4383	0.4363	0.4349	0.4331	0.4318	0.4310	0.4294	0.428
k_p	167.3794	163.1435	159.4280	157.0770	154.0400	150.8728	147.7042	135.2968	137.3698	119.4982
k_i	1222.3293	1229.6786	1236.8853	1244.2003	1251.2625	1258.3498	1265.3936	1271.6947	1278.6168	1284.2182
k_d	143.0974	139.6364	140.0640	138.9832	137.5018	135.5054	125.7913	135.4000	116.1681	130.2599
IAE	0.4344	0.4327	0.4311	0.4298	0.4285	0.4272	0.4257	0.4235	0.4236	0.4210
k_p	185.5930	112.6394	68.0453	119.2084	128.0319	107.8565	119.9239	90.4799	117.8585	83.0308
k_i	1209.7053	1218.1881	1222.5338	1230.5997	1238.4448	1244.1681	1251.9373	1257.1760	1264.7430	1269.168
k_d	140.6742	85.8658	183.8764	137.6611	109.0122	135.6171	99.6565	145.7495	95.9805	153.3217
IAE	0.4328	0.4512	0.4420	0.5177	0.4326	0.4248	0.4315	0.4254	0.4461	0.4266
k_p	131.2500	116.2533	118.3830	89.2357	115.5033	72.1303	121.7519	134.5939	121.2314	121.7282
k_i	1217.8294	1224.5812	1232.5629	1237.8595	1245.7230	1250.0643	1257.9959	1264.8536	1270.8896	1278.3363
k_d	113.5537	125.4506	98.9418	141.4020	87.9168	186.5907	144.0773	119.6039	125.1574	107.8016
IAE	0.4376	0.4271	0.4271	0.4272	0.4455	0.4383	0.5116	0.4316	0.4210	0.4197

Table A.4 : ***Algorithm 2*** progress while using IAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 31-40)

Parameter	Iteration									
	31	32	33	34	35	36	37	38	39	40
k_p	287.0822	281.5594	278.1437	273.7083	270.4204	266.2129	260.9400	259.5057	255.7734	249.8936
k_i	1307.73	1314.312	1321.074	1327.56	1334.096	1340.17	1346.035	1352.375	1358.345	1363.955
k_d	211.8568	213.5867	207.6103	208.8687	205.4431	203.0134	207.9061	202.5471	198.4901	193.3616
IAE	0.4367	0.4352	0.4341	0.4326	0.4315	0.4302	0.4290	0.4285	0.4270	0.4257
k_p	236.4686	233.4201	226.2152	222.6222	218.2030	214.0614	207.7301	202.8571	200.3287	195.8584
k_i	1299.97	1307.03	1312.99	1319.104	1325.1	1331.025	1336.7	1342.4	1348.18	1353.62
k_d	189.8497	181.1065	184.5100	180.4237	176.6732	170.0928	170.9929	172.8704	168.1532	161.1528
IAE	0.4320	0.4304	0.4290	0.4276	0.4263	0.4251	0.4241	0.4227	0.4217	0.4207
k_p	186.1810	182.1831	177.1414	174.7338	168.3581	167.3625	158.5855	151.6181	147.0833	138.9855
k_i	1294.5190	1300.9717	1307.1142	1313.6394	1319.3736	1325.4668	1331.0994	1336.7746	1342.6042	1347.9331
k_d	156.2588	153.8197	153.4886	147.9693	148.4312	140.1418	138.7079	137.8083	133.3012	135.1256
IAE	0.4266	0.4251	0.4237	0.4225	0.4214	0.4203	0.4193	0.4174	0.4159	0.4143
k_p	122.9959	104.6518	115.2437	69.3675	118.4442	122.7880	106.1371	115.8270	97.6765	112.7855
k_i	1291.3555	1296.3500	1303.9733	1307.4962	1315.7128	1322.5378	1327.1821	1333.3472	1337.7815	1343.2793
k_d	108.8407	132.5153	88.0462	185.7847	130.8924	109.5158	129.1092	106.3744	126.8732	105.3724
IAE	0.4200	0.4172	0.4218	0.4334	0.5043	0.4160	0.4137	0.4155	0.4116	0.4153
k_p	115.7314	100.4296	109.3711	66.8245	112.1912	59.8194	115.8181	136.0448	115.7117	127.8425
k_i	1276.4900	1281.8881	1289.1032	1292.9401	1303.8619	1307.2431	1314.9946	1321.1113	1325.8810	1332.8408
k_d	107.1262	117.5075	87.3969	169.7305	82.7115	196.1161	150.3779	115.0180	141.7535	108.1408
IAE	0.4559	0.4169	0.4174	0.4304	0.4873	0.4420	0.5243	0.4302	0.4167	0.4222
k_p	102.5530	112.7873	90.9171	108.6369	98.4038	96.4910	79.5389	85.9131	38.8180	102.9053
k_i	1283.4016	1289.8151	1294.5709	1300.0902	1305.6524	1311.6891	1316.8059	1322.9979	1326.4036	1334.5687
k_d	124.9577	101.3193	125.6874	109.0073	111.3832	97.6562	108.9123	75.0533	193.8281	140.0987
IAE	0.4184	0.4198	0.4166	0.4219	0.4121	0.4104	0.4096	0.4084	0.4329	0.5441

Table A.5 : ***Algorithm 2*** progress while using IAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 41-50)

Parameter	Iteration									
	41	42	43	44	45	46	47	48	49	50
k_p	243.2849	237.5261	235.5176	228.5624	227.3644	223.3910	218.6228	216.4245	213.1091	208.2009
k_i	1369.2	1374.47	1380.27	1385.18	1390.55	1395.57	1400.34	1405.45	1410.242	1414.78
k_d	191.4182	197.4203	185.0052	193.8262	189.0862	183.7594	185.2603	183.2145	178.0531	175.5917
IAE	0.4242	0.4229	0.4224	0.4212	0.4203	0.4193	0.4184	0.4176	0.4169	0.416
k_p	188.0815	185.2629	177.5833	174.2137	170.9143	167.3219	167.9456	168.9474	163.6130	159.1373
k_i	1358.61	1364.23	1369.16	1374.6	1379.52	1384.26	1389.03	1394.243	1398.73	1403.4
k_d	163.8840	155.9908	158.4965	155.4259	151.6537	156.2916	157.4853	147.9849	147.6082	141.1201
IAE	0.4196	0.4183	0.4170	0.4158	0.4147	0.4141	0.4136	0.4133	0.4125	0.4116
k_p	36.6444	118.8536	127.2229	97.1382	124.0811	106.7334	113.9808	93.0880	111.3106	87.487
k_i	1354.0927	1358.6032	1365.4361	1369.1325	1375.1493	1379.2087	1383.9705	1387.8548	1393.3691	1397.2024
k_d	117.924	138.8695	104.7358	154.6829	111.4530	124.7274	104.8229	134.9263	102.0086	134.8793
IAE	0.4131	0.4126	0.4152	0.4154	0.4355	0.4089	0.4079	0.4071	0.4152	0.4066
k_p	94.5212	111.1963	94.6613	98.4346	51.0067	105.3311	95.9625	91.2810	87.6445	78.4152
k_i	1347.5893	1353.5387	1358.0687	1364.1742	1367.0844	1376.5229	1381.0172	1385.5037	1389.8376	1394.1941
k_d	132.8769	105.9334	116.0300	80.7798	182.4891	110.3809	110.7118	106.9588	100.9495	93.5337
IAE	0.4098	0.4180	0.4082	0.4071	0.4290	0.5083	0.4039	0.4021	0.401	0.3995
k_p	104.3617	116.9308	67.5887	119.4132	136.8242	109.0321	132.4143	116.7537	123.7804	99.8144
k_i	1337.4857	1344.4698	1347.6104	1354.8476	1361.1290	1364.9033	1370.6981	1374.9961	1380.8072	1384.6937
k_d	135.7919	86.8843	194.9423	149.6492	110.3755	154.6077	118.5856	131.6862	107.8070	140.016
IAE	0.4156	0.4186	0.4347	0.5088	0.4232	0.4165	0.4300	0.4095	0.4091	0.4101
k_p	118.5700	87.3215	117.3222	94.0174	112.2430	71.0224	116.5887	95.8461	113.6885	104.7381
k_i	1341.6633	1345.6383	1351.9929	1356.1678	1362.6523	1365.4827	1374.9456	1378.9785	1383.8649	1388.3096
k_d	99.6831	152.5404	104.5372	135.8067	90.6972	179.3920	105.7903	129.9988	114.6676	115.4838
IAE	0.4233	0.4166	0.4421	0.4113	0.4192	0.4222	0.4842	0.4085	0.4127	0.4047

Table A.6 : ***Algorithm 2*** progress while using ISE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 1-10)

Parameter	Iteration									
	1	2	3	4	5	6	7	8	9	10
k_p	16.6101	27.6372	33.7058	35.4978	101.0641	110.5185	119.1982	127.1994	134.6004	141.466
k_i	1028.767	1036.877	1044.56	1051.91	1045.447	1052.54	1059.4	1066.05	1072.515	1078.8
k_d	64.8760	56.8881	51.3509	49.6259	270.3282	267.9178	265.6900	263.6396	261.7619	260.0519
ISE	0.2354	0.2278	0.2235	0.2214	0.2267	0.2261	0.2233	0.2208	0.2186	0.2167
k_p	20.1894	30.2277	35.6436	37.2746	37.3878	37.4360	37.4531	37.4371	95.6113	104.5338
k_i	1028.66	1036.7	1044.3	1051.663	1058.8	1065.849	1072.786	1079.62	1074.462	1081.19
k_d	65.0894	57.5577	52.6067	51.1637	51.4547	51.7542	52.0208	52.2578	244.8405	242.2670
ISE	0.2338	0.2271	0.2233	0.2214	0.2203	0.2193	0.2183	0.2174	0.2167	0.2218
k_p	24.5669	33.4501	38.1387	39.6234	39.8708	39.9854	40.0675	40.1184	40.1385	40.1284
k_i	1028.5363	1036.4922	1044.0713	1051.3708	1058.5020	1065.5223	1072.4404	1079.2602	1085.9852	1092.6189
k_d	65.4553	58.5546	54.3169	53.1696	53.4266	53.7800	54.1058	54.4033	54.6729	54.9148
ISE	0.2321	0.2264	0.2231	0.2215	0.2204	0.2193	0.2184	0.2174	0.2165	0.2156
k_p	32.7868	39.6646	43.1936	44.4920	44.9400	45.2022	45.4133	45.5928	45.7440	45.8680
k_i	1028.3108	1036.1112	1043.5902	1050.8372	1057.9328	1064.9147	1071.7934	1078.5739	1085.2596	1091.8541
k_d	66.4314	60.8956	57.9383	57.2575	57.5262	57.9610	58.3972	58.8118	59.2020	59.5674
ISE	0.2294	0.2252	0.2228	0.2214	0.2204	0.2194	0.2184	0.2175	0.2165	0.2157
k_p	52.1106	55.1420	56.8299	57.8320	58.5372	59.1156	59.6321	60.1110	60.5618	60.9884
k_i	1027.8158	1035.2734	1042.5337	1049.6429	1056.6278	1063.5020	1070.2731	1076.9461	1083.5248	1090.0128
k_d	70.1482	68.3029	67.8254	68.0848	68.6460	69.3085	69.9955	70.6790	71.3489	72.0018
ISE	0.2255	0.2237	0.2224	0.2213	0.2203	0.2193	0.2184	0.2174	0.2165	0.2157
k_p	93.8314	93.0351	93.0981	93.5565	94.2199	94.9939	95.8271	96.6901	97.5654	98.4427
k_i	1026.8102	1033.6956	1040.5248	1047.2738	1053.9344	1060.5039	1066.9825	1073.3713	1079.6725	1085.8882
k_d	86.7415	90.5118	93.0607	95.0539	96.7467	98.2610	99.6624	100.9880	102.2601	103.4925
ISE	0.2260	0.2227	0.2215	0.2205	0.2195	0.2186	0.2176	0.2168	0.2159	0.2151

Table A.7 : ***Algorithm 2*** progress while using ISE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 11-20)

Parameter	Iteration									
	11	12	13	14	15	16	17	18	19	20
k_p	147.8507	153.8012	159.3576	164.555	169.4243	173.9929	178.2854	182.3237	186.128	189.7162
k_i	1084.94	1090.92	1096.78	1102.5	1108.12	1113.618	1119.02	1124.32	1129.54	1134.67
k_d	258.5048	257.1156	255.8793	254.7907	253.8446	253.0353	252.3575	251.8054	251.3733	251.0553
ISE	0.2149	0.2133	0.2119	0.2106	0.2094	0.2083	0.2073	0.2063	0.2055	0.2046
k_p	112.6913	120.1774	127.0686	133.4285	139.3110	144.7626	149.8235	154.5293	158.9112	162.997
k_i	1087.71	1094.04	1100.2	1106.2	1112.1	1117.798	1123.41	1128.9	1134.3	1139.59
k_d	239.8972	237.7265	235.7505	233.9644	232.3631	230.9411	229.6924	228.6110	227.6901	226.923
ISE	0.2193	0.2170	0.2151	0.2133	0.2118	0.2104	0.2091	0.2080	0.2069	0.2060
k_p	40.0884	40.0190	39.9204	39.7928	39.6366	39.4518	39.2386	38.9972	38.7274	97.9183
k_i	1099.1645	1105.6251	1112.0035	1118.3024	1124.5246	1130.6724	1136.7483	1142.7545	1148.6932	1142.8041
k_d	55.1291	55.3161	55.4759	55.6087	55.7143	55.7930	55.8447	55.8694	55.8670	248.079
ISE	0.2147	0.2139	0.2130	0.2122	0.2114	0.2107	0.2099	0.2092	0.2085	0.2089
k_p	45.9653	46.0364	46.0817	46.1017	46.0967	46.0670	46.0129	45.9346	45.8323	45.7063
k_i	1098.3606	1104.7820	1111.1212	1117.3810	1123.5639	1129.6724	1135.7088	1141.6754	1147.5743	1153.4075
k_d	59.9083	60.2249	60.5175	60.7863	61.0315	61.2532	61.4515	61.6267	61.7787	61.9077
ISE	0.2148	0.2139	0.2131	0.2123	0.2115	0.2108	0.2100	0.2093	0.2086	0.2079
k_p	61.3927	61.7758	62.1387	62.4818	62.8058	63.1111	63.3982	63.6676	63.9196	64.1545
k_i	1096.4133	1102.7292	1108.9633	1115.1183	1121.1968	1127.2010	1133.1333	1138.9959	1144.7908	1150.52
k_d	72.6366	73.2530	73.8511	74.4312	74.9937	75.5387	76.0665	76.5776	77.0720	77.5502
ISE	0.2148	0.2140	0.2132	0.2124	0.2117	0.2109	0.2102	0.2095	0.2088	0.2081
k_p	99.3155	100.1799	101.0336	101.8752	102.7040	103.5196	104.3219	105.1110	105.8870	106.6502
k_i	1092.0207	1098.0726	1104.0462	1109.9438	1115.7676	1121.5200	1127.2028	1132.8182	1138.3681	1143.8542
k_d	104.6941	105.8705	107.0256	108.1622	109.2819	110.3863	111.4764	112.5530	113.6166	114.668
ISE	0.2142	0.2135	0.2127	0.2119	0.2112	0.2105	0.2098	0.2091	0.2085	0.2078

Table A.8 : ***Algorithm 2*** progress while using ISE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 21-30)

Parameter	Iteration									
	21	22	23	24	25	26	27	28	29	30
k_p	193.1049	196.3092	199.3430	202.2188	204.9485	207.5427	210.0114	212.3636	214.6078	216.7517
k_i	1139.72	1144.693	1149.59	1154.43	1159.19	1163.89	1168.54	1173.125	1177.66	1182.13
k_d	250.8458	250.7389	250.7290	250.8104	250.9777	251.2254	251.5485	251.9418	252.4006	252.9203
ISE	0.2039	0.2031	0.2024	0.2018	0.2012	0.2006	0.2000	0.1994	0.1989	0.1984
k_p	166.8135	170.3821	173.7239	176.8580	179.8017	182.5706	185.1793	187.6411	189.9679	192.1710
k_i	1144.8	1149.91	1154.94	1159.9	1164.79	1169.6	1174.36	1179.05	1183.68	1188.25
k_d	226.3023	225.8209	225.4711	225.2453	225.1361	225.1360	225.2376	225.4337	225.7175	226.0823
ISE	0.2051	0.2043	0.2035	0.2028	0.2021	0.2014	0.2008	0.2003	0.1997	0.1992
k_p	105.8884	113.2211	119.9878	126.2480	132.0518	137.4427	142.4580	147.1308	151.4902	155.5626
k_i	1148.6858	1154.4085	1159.9868	1165.4332	1170.7585	1175.9721	1181.0823	1186.0964	1191.0209	1195.8616
k_d	245.6777	243.4539	241.4040	239.5242	237.8104	236.2582	234.8631	233.6203	232.5246	231.5705
ISE	0.2130	0.2110	0.2092	0.2076	0.2062	0.2049	0.2038	0.2027	0.2018	0.2009
k_p	45.5566	45.3834	45.1867	44.9667	44.7234	44.4568	44.1668	43.8535	43.5167	43.1564
k_i	1159.1769	1164.8844	1170.5318	1176.1208	1181.6530	1187.1300	1192.5532	1197.9241	1203.2442	1208.5146
k_d	62.0137	62.0967	62.1567	62.1937	62.2077	62.1986	62.1663	62.1107	62.0318	61.9292
ISE	0.2072	0.2066	0.2059	0.2053	0.2047	0.2041	0.2035	0.2029	0.2023	0.2017
k_p	64.3729	64.5749	64.7609	64.9312	65.0861	65.2258	65.3507	65.4608	65.5565	65.6379
k_i	1156.1855	1161.7889	1167.3320	1172.8165	1178.2440	1183.6160	1188.9339	1194.1992	1199.4131	1204.577
k_d	78.0122	78.4584	78.8889	79.3039	79.7037	80.0883	80.4580	80.8130	81.1532	81.479
ISE	0.2075	0.2068	0.2062	0.2056	0.2050	0.2044	0.2038	0.2032	0.2027	0.2021
k_p	107.4008	108.1391	108.8654	109.5800	110.2832	110.9754	111.6567	112.3276	112.9881	113.6387
k_i	1149.2784	1154.6424	1159.9476	1165.1958	1170.3884	1175.5267	1180.6123	1185.6463	1190.6301	1195.5649
k_d	115.7074	116.7354	117.7522	118.7582	119.7536	120.7388	121.7141	122.6795	123.6355	124.5821
ISE	0.2072	0.2066	0.2060	0.2054	0.2048	0.2042	0.2037	0.2031	0.2026	0.2021

Table A.9 : ***Algorithm 2*** progress while using ISE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 31-40)

Parameter	Iteration									
	31	32	33	34	35	36	37	38	39	40
k_p	218.8025	220.7667	222.6506	224.4596	226.1990	227.8736	229.4877	231.0455	232.5507	234.0068
k_i	1186.56	1190.94	1195.27	1199.56	1203.8	1207.994	1212.15	1216.27	1220.35	1224.39
k_d	253.4965	254.1250	254.8019	255.5234	256.2861	257.0867	257.9218	258.7890	259.6853	260.61
ISE	0.1979	0.1974	0.1970	0.1965	0.1961	0.1956	0.1952	0.1948	0.1944	0.1940
k_p	194.2604	196.2455	198.1347	199.9359	201.6561	203.3018	204.8789	206.3928	207.8485	209.2503
k_i	1192.78	1197.245	1201.66	1206.035	1210.36	1214.64	1218.88	1223.072	1227.23	1231.34
k_d	226.5217	227.0296	227.6005	228.2289	228.9097	229.6383	230.4104	231.2218	232.0688	232.9480
ISE	0.1987	0.1982	0.1977	0.1972	0.1968	0.1963	0.1959	0.1955	0.1951	0.1947
k_p	159.3712	162.9376	166.2809	169.4189	172.3677	175.1422	177.7560	180.2217	182.5508	184.7539
k_i	1200.6237	1205.3119	1209.9304	1214.4832	1218.9738	1223.4052	1227.7806	1232.1025	1236.3734	1240.5957
k_d	230.7525	230.0647	229.5011	229.0556	228.7220	228.4941	228.3658	228.3311	228.3839	228.5186
ISE	0.2001	0.1994	0.1987	0.1980	0.1974	0.1969	0.1963	0.1958	0.1953	0.1948
k_p	42.7724	42.3646	41.9328	41.4768	40.9963	40.4911	99.7220	106.5899	112.9448	118.8383
k_i	1213.7367	1218.9118	1224.0410	1229.1255	1234.1663	1239.1645	1232.8166	1237.8021	1242.6743	1247.4427
k_d	61.8030	61.6529	61.4786	61.2801	61.0569	60.8088	247.6656	245.4395	243.3672	241.4456
ISE	0.2012	0.2006	0.2001	0.1996	0.1990	0.1985	0.1992	0.2029	0.2014	0.2001
k_p	65.7052	65.7587	65.7984	65.8246	65.8373	65.8368	65.8231	65.7963	65.7566	65.7041
k_i	1209.6922	1214.7597	1219.7808	1224.7565	1229.6880	1234.5763	1239.4223	1244.2270	1248.9913	1253.7162
k_d	81.7904	82.0875	82.3704	82.6393	82.8942	83.1352	83.3623	83.5758	83.7755	83.9615
ISE	0.2016	0.2011	0.2006	0.2001	0.1996	0.1991	0.1986	0.1981	0.1977	0.1972
k_p	114.2796	114.9110	115.5332	116.1463	116.7507	117.3465	117.9339	118.5132	119.0845	119.648
k_i	1200.4518	1205.2920	1210.0865	1214.8364	1219.5428	1224.2065	1228.8285	1233.4098	1237.9511	1242.4534
k_d	125.5196	126.4482	127.3681	128.2794	129.1824	130.0773	130.9640	131.8430	132.7142	133.5778
ISE	0.2016	0.2011	0.2006	0.2001	0.1996	0.1991	0.1987	0.1982	0.1978	0.1974

Table A.10 : ***Algorithm 2*** progress while using ISE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 41-50)

Parameter	Iteration									
	41	42	43	44	45	46	47	48	49	50
k_p	235.4170	236.7842	238.1112	239.4006	240.6545	241.8752	243.0647	244.2249	245.3574	246.4638
k_i	1228.4	1232.36	1236.29	1240.19	1244.06	1247.89	1251.7	1255.47	1259.22	1262.93
k_d	261.5555	262.5249	263.5144	264.5221	265.5464	266.5855	267.6381	268.7028	269.7782	270.863
ISE	0.1936	0.1933	0.1929	0.1925	0.1922	0.1918	0.1915	0.1911	0.1908	0.1905
k_p	210.6026	211.9088	213.1726	214.3970	215.5848	216.7386	217.8608	218.9535	220.0188	221.0584
k_i	1235.4	1239.46	1243.47	1247.44	1251.38	1255.286	1259.159	1263	1266.8	1270.59
k_d	233.8562	234.7904	235.7481	236.7266	237.7238	238.7376	239.7662	240.8078	241.8609	242.924
ISE	0.1943	0.1939	0.1935	0.1931	0.1928	0.1924	0.1921	0.1917	0.1914	0.1911
k_p	186.8409	188.8208	190.7019	192.4918	194.1975	195.8256	197.3820	198.8722	200.3010	201.6733
k_i	1244.7713	1248.9022	1252.9903	1257.0371	1261.0442	1265.0130	1268.9449	1272.8410	1276.7026	1280.5306
k_d	228.7294	229.0110	229.3583	229.7662	230.2301	230.7457	231.3087	231.9153	232.5617	233.2446
ISE	0.1944	0.1940	0.1935	0.1931	0.1928	0.1924	0.1920	0.1916	0.1913	0.1909
k_p	124.3145	129.4113	134.1617	138.5950	142.7371	146.6112	150.2383	153.6374	156.8260	159.8198
k_i	1252.1154	1256.6997	1261.2018	1265.6274	1269.9814	1274.2685	1278.4927	1282.6575	1286.7665	1290.8225
k_d	239.6715	238.0415	236.5519	235.1989	233.9784	232.8862	231.9176	231.0681	230.3328	229.7068
ISE	0.1989	0.1978	0.1969	0.196	0.1952	0.1945	0.1938	0.1931	0.1926	0.192
k_p	65.6388	65.5608	65.4702	65.3671	65.2515	65.1234	64.9829	64.8301	64.6649	64.4873
k_i	1258.4025	1263.0511	1267.6627	1272.2381	1276.7781	1281.2834	1285.7548	1290.1929	1294.5984	1298.972
k_d	84.1340	84.2929	84.4382	84.5701	84.6885	84.7934	84.8848	84.9627	85.0272	85.0782
ISE	0.1968	0.1963	0.1959	0.1955	0.1951	0.1947	0.1942	0.1938	0.1935	0.1931
k_p	120.2039	120.7523	121.2935	121.8275	122.3546	122.8749	123.3886	123.8957	124.3965	124.891
k_i	1246.9174	1251.3440	1255.7339	1260.0879	1264.4067	1268.6909	1272.9412	1277.1584	1281.3429	1285.4956
k_d	134.4340	135.2830	136.1248	136.9595	137.7874	138.6085	139.4229	140.2308	141.0323	141.8275
ISE	0.1969	0.1965	0.1961	0.1957	0.1953	0.1949	0.1945	0.1942	0.1938	0.1934

Table A.11 : ***Algorithm 2*** progress while using ITAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 1-10)

Parameter	Iteration									
	1	2	3	4	5	6	7	8	9	10
k_p	1523.47	1524.73	1536.9	1550.79	1564.56	1577.69	1589.99	1601.56	1612.32	1622.3
k_i	838.56	946.38	1015.84	1072.47	1121.99	1166.56	1207.28	1244.95	1280	1312.86
k_d	1779.59	1763.03	1740.53	1719.64	1698.03	1677.74	1658.16	1639.31	1617.46	1600.12
ITAE	1.2966	1.3629	1.1706	1.0704	0.9978	0.9387	0.8892	0.8468	0.8091	0.7751
k_p	1337.04	1349.9	1365.59	1380.67	1394.66	1407.33	1419.22	1430.6	1441.33	1451.5
k_i	860.37	945.04	1009.62	1064.7	1113.2	1157.05	1196.62	1232.35	1265.2	1295.5
k_d	1550.13	1523.05	1498.12	1472.46	1447.73	1425.9	1403.62	1382.24	1359.15	1340.63
ITAE	1.1541	1.2512	1.1087	1.0140	0.9399	0.8799	0.8300	0.7876	0.7514	0.7193
k_p	1144.7759	1163.0091	1179.2999	1194.1269	1207.9495	1220.5898	1232.0112	1242.3687	1251.7107	1259.6261
k_i	885.8068	957.2101	1016.2987	1066.5636	1110.2073	1149.0483	1184.3353	1216.6577	1246.7250	1275.2174
k_d	1290.1265	1254.9397	1222.2577	1193.0544	1163.8761	1136.2679	1109.7251	1086.7425	1061.4330	1039.3412
ITAE	1.0063	1.1222	1.0006	0.9111	0.8431	0.7879	0.7421	0.7026	0.6694	0.6391
k_p	863.2091	887.8798	907.3137	922.2563	932.8239	940.7541	947.4344	950.7644	953.6469	953.4517
k_i	923.7451	975.2949	1019.7673	1059.4999	1096.6570	1130.5951	1161.1330	1190.8567	1217.6492	1244.142
k_d	919.0748	873.6767	832.8350	799.4097	767.1445	738.8948	711.7297	688.0043	664.8361	643.7874
ITAE	0.7903	0.9114	0.8137	0.7387	0.6815	0.6334	0.5941	0.5609	0.5324	0.5084
k_p	434.1155	478.5397	492.0331	480.5283	471.7998	461.5713	451.8339	442.1306	432.9524	424.1596
k_i	979.9132	1006.0585	1034.8439	1067.0672	1094.9241	1121.1836	1144.5096	1166.0068	1185.3517	1203.4508
k_d	465.5313	394.4235	345.8092	333.0064	317.6212	304.1024	293.2441	283.7559	276.5316	264.4023
ITAE	0.4961	0.6302	0.5024	0.4537	0.4285	0.4080	0.3901	0.3756	0.3635	0.3532
k_p	-19.1260	649.8761	-80.8602	202.8550	366.8775	480.2207	575.9009	660.9675	738.1107	808.8083
k_i	1026.3764	1042.2304	1034.6700	1218.3864	1381.7903	1486.2108	1549.5746	1596.6846	1634.6670	1667.1162
k_d	346.7926	173.5416	3152.4230	3153.8318	3118.8554	3089.4584	3071.0259	3056.1327	3041.2701	3027.0354
ITAE	0.3716	2.0739	1.8837	5.7080	3.8130	2.8737	2.4529	2.2052	2.0232	1.8782

Table A.12 : ***Algorithm 2*** progress while using ITAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 11-20)

Parameter	Iteration									
	11	12	13	14	15	16	17	18	19	20
k_p	1632.0118	1641.3254	1650.2409	1658.9607	1667.0892	1674.9970	1682.4612	1689.6968	1696.4672	1702.8633
k_i	1343.25	1371.68	1398.47	1423.58	1447.71	1470.47	1492.39	1513.2	1533.45	1552.94
k_d	1581.4935	1565.1937	1548.1735	1530.7570	1515.9528	1499.6679	1484.2018	1467.6521	1453.8581	1439.9930
ITAE	0.7460	0.7196	0.6970	0.6752	0.6560	0.6386	0.6223	0.6068	0.5925	0.5795
k_p	1461.2762	1470.2360	1478.8963	1486.7050	1494.1233	1500.7911	1507.0984	1512.9329	1517.9052	1523.1428
k_i	1323.79	1350.53	1375.52	1399.71	1422.48	1444.51	1465.57	1485.84	1505.7	1524.27
k_d	1320.3784	1301.9716	1285.6848	1264.8901	1249.5810	1231.0638	1215.9226	1200.0597	1183.6727	1167.2946
ITAE	0.6919	0.6669	0.6443	0.6246	0.6049	0.5884	0.5716	0.5577	0.5436	0.5304
k_p	1267.0286	1273.0333	1278.7278	1282.9308	1287.5448	1290.7214	1292.6488	1294.7313	1296.0714	1296.6973
k_i	1301.6040	1327.1709	1350.8652	1374.0957	1395.4655	1416.5581	1437.3684	1456.7687	1475.6267	1494.026
k_d	1018.9795	998.5954	977.5034	960.8809	940.7114	922.1088	906.4421	890.9616	876.3115	861.3058
ITAE	0.6127	0.5894	0.5681	0.5484	0.5315	0.5152	0.4999	0.4864	0.4746	0.4631
k_p	952.1256	949.2407	946.1232	942.4448	936.6311	930.5668	923.6869	917.2007	911.3467	905.8912
k_i	1269.0633	1293.1949	1315.8020	1337.1769	1358.1877	1377.9221	1397.0308	1414.9038	1431.7553	1447.6903
k_d	626.5209	611.7977	598.2386	583.1344	572.3287	560.3107	552.3148	548.1067	542.6575	531.7085
ITAE	0.4865	0.4685	0.4525	0.4387	0.4257	0.4145	0.4042	0.3949	0.3875	0.3804
k_p	413.8513	406.1088	398.0970	386.8732	380.0087	372.9427	366.8628	359.2352	351.0576	348.1761
k_i	1219.7365	1234.9317	1249.3906	1259.4943	1269.2233	1278.8193	1287.5427	1295.6654	1303.0298	1311.0969
k_d	264.7510	258.7065	254.3948	256.1149	251.6767	249.0589	244.1630	240.9302	246.1579	240.5373
ITAE	0.3436	0.3368	0.3302	0.3259	0.3218	0.3188	0.3159	0.3136	0.3112	0.3094
k_p	874.1971	934.9361	991.6647	1044.9091	1095.0382	1142.3924	1187.2087	1229.8051	1270.3095	1308.9225
k_i	1695.6105	1721.5274	1745.2645	1767.1921	1787.8011	1807.4575	1826.2511	1843.9281	1861.0341	1877.6321
k_d	3013.5973	2998.8905	2985.0459	2971.1121	2956.5848	2942.6191	2928.2974	2914.5319	2899.9555	2886.065
ITAE	1.7583	1.6566	1.5683	1.4912	1.4229	1.3617	1.3065	1.2566	1.2112	1.1693

Table A.13 : ***Algorithm 2*** progress while using ITAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 21-30)

Parameter	Iteration									
	21	22	23	24	25	26	27	28	29	30
k_p	1709.1540	1714.8244	1720.4910	1725.7982	1730.7101	1735.0385	1739.7335	1743.8540	1747.5217	1750.9618
k_i	1571.615	1589.99	1607.48	1624.47	1641.23	1657.7	1673.22	1688.6	1703.88	1718.61
k_d	1426.9489	1410.6877	1399.4049	1386.4956	1370.4873	1360.6981	1348.8009	1336.8429	1324.1958	1312.7767
ITAE	0.5677	0.5561	0.5442	0.5349	0.5249	0.5146	0.5069	0.4987	0.4909	0.4828
k_p	1527.5402	1531.1220	1535.2096	1538.5424	1541.1805	1543.6722	1546.1877	1548.3032	1549.6768	1551.0541
k_i	1542.6	1560.7	1577.5	1594.12	1610.66	1626.61	1641.7	1656.67	1671.56	1685.93
k_d	1153.9110	1139.2045	1125.9283	1110.1723	1098.6963	1085.4100	1073.9838	1062.1923	1050.2832	1038.4887
ITAE	0.5183	0.5074	0.4967	0.4868	0.4773	0.4689	0.4601	0.4527	0.4454	0.4384
k_p	1297.0425	1296.7073	1296.4099	1295.6591	1294.0907	1292.3763	1290.4676	1288.0928	1285.7030	1282.9666
k_i	1511.6892	1529.0147	1545.2885	1561.2088	1576.8533	1591.9229	1606.6208	1620.9697	1634.6042	1647.9661
k_d	846.6518	836.5633	824.1781	811.3857	801.5972	789.9084	780.6183	770.8558	763.1177	753.4153
ITAE	0.4524	0.4423	0.4341	0.4257	0.4178	0.4109	0.4038	0.3978	0.3918	0.3868
k_p	899.7220	893.4605	887.9965	882.7226	876.8630	870.8536	864.8752	859.7586	854.5723	849.437
k_i	1463.1043	1477.9407	1491.7353	1504.9639	1517.7768	1529.9506	1541.7087	1552.8999	1563.8972	1574.4747
k_d	525.5403	523.7613	518.7732	508.7619	503.3099	500.0790	499.2946	498.1495	495.1800	491.0411
ITAE	0.3732	0.3674	0.3622	0.3571	0.3522	0.3475	0.3437	0.3403	0.3372	0.3341
k_p	342.2656	335.8105	332.7025	326.6111	324.5739	321.6593	314.1078	307.7843	301.2550	289.5115
k_i	1318.7407	1325.9378	1333.1408	1339.3502	1345.4455	1351.7616	1357.5461	1363.0606	1368.6467	1373.9466
k_d	235.6561	236.9593	230.3331	234.0637	232.6219	225.0920	222.6031	219.6523	209.6109	211.3056
ITAE	0.3073	0.3052	0.3035	0.3022	0.3010	0.3000	0.2987	0.2969	0.2955	0.2948
k_p	1345.8805	1381.1874	1414.9296	1447.3212	1478.5497	1508.5068	1537.3933	1565.2177	1592.0349	1617.9316
k_i	1893.5038	1908.8868	1924.0117	1938.6799	1952.7512	1966.5949	1979.9899	1993.1346	2006.0774	2018.6745
k_d	2871.6547	2857.5816	2843.8921	2829.3176	2814.7213	2802.5406	2789.6489	2775.1725	2760.5302	2746.4314
ITAE	1.1308	1.0952	1.0622	1.0315	1.0024	0.9757	0.9511	0.9277	0.9049	0.8833

Table A.14 : ***Algorithm 2*** progress while using ITAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 31-40)

Parameter	Iteration									
	31	32	33	34	35	36	37	38	39	40
k_p	1754.6650	1757.8381	1760.6145	1763.0835	1765.8573	1768.3886	1770.4940	1772.1135	1773.7273	1775.2284
k_i	1732.7	1746.66	1760.6	1774.16	1787.02	1799.82	1812.54	1825.21	1837.46	1849.48
k_d	1301.1737	1288.6286	1277.1029	1267.3085	1258.2551	1248.7898	1235.9321	1224.6344	1215.8066	1205.9462
ITAE	0.4758	0.4689	0.4618	0.4552	0.4496	0.4445	0.4391	0.4329	0.4278	0.4231
k_p	1551.9041	1552.7016	1553.1530	1553.5547	1553.4259	1552.8668	1552.3356	1551.5303	1550.9053	1550.1366
k_i	1699.91	1713.55	1726.86	1739.9	1752.7	1765.27	1777.66	1789.61	1801.18	1812.68
k_d	1027.0760	1017.3312	1007.0368	997.4282	986.1661	976.6076	968.2070	963.3047	954.2372	946.0581
ITAE	0.4316	0.4254	0.4197	0.4141	0.4087	0.4033	0.3984	0.3941	0.3905	0.3862
k_p	1279.9845	1276.9384	1273.8089	1270.1048	1266.2025	1262.3377	1258.3751	1253.6189	1249.2238	1244.9087
k_i	1661.2085	1673.7676	1686.1147	1698.2589	1709.8387	1721.1402	1732.2427	1742.9034	1753.1454	1763.3202
k_d	748.1463	741.7682	731.4549	723.0043	719.9545	713.3992	705.6234	702.9823	699.4924	693.5609
ITAE	0.3815	0.3773	0.3729	0.3679	0.3642	0.3610	0.3572	0.3540	0.3511	0.3483
k_p	844.2883	839.2416	834.1020	829.1886	824.4992	817.5819	811.5168	806.8409	802.7290	799.4412
k_i	1584.7439	1594.3561	1603.8961	1613.0777	1622.2038	1627.9584	1633.7241	1640.1102	1646.3590	1652.5409
k_d	488.2984	484.6317	481.5110	481.2432	478.0483	478.4511	480.1851	480.9058	483.2151	476.7976
ITAE	0.3311	0.3283	0.3255	0.3232	0.3210	0.3193	0.3177	0.3163	0.3150	0.3139
k_p	283.2248	275.8133	271.2003	269.6435	263.6675	262.1521	255.6861	253.3121	240.7009	244.9157
k_i	1378.8907	1383.2488	1387.7513	1392.2358	1396.8388	1401.5962	1405.6792	1410.2338	1414.2094	1418.7571
k_d	203.5483	204.2368	205.3440	199.7757	203.0174	196.6897	198.4469	185.1248	198.5610	183.0435
ITAE	0.2920	0.2912	0.2900	0.2893	0.2884	0.2879	0.2866	0.2858	0.2854	0.2854
k_p	1642.8878	1667.0409	1690.4163	1713.0050	1734.9124	1756.0998	1776.6745	1796.6130	1815.9765	1834.7558
k_i	2031.0598	2043.2561	2055.2209	2066.9502	2078.4751	2089.8405	2101.0416	2112.0230	2122.8085	2133.4975
k_d	2732.4370	2720.2804	2708.0958	2695.4328	2681.8649	2667.8979	2656.2911	2644.1041	2632.0411	2620.0142
ITAE	0.8629	0.8439	0.8264	0.8098	0.7937	0.7780	0.7628	0.7492	0.7361	0.7235

Table A.15 : ***Algorithm 2*** progress while using ITAE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 41-50)

Parameter	Iteration									
	41	42	43	44	45	46	47	48	49	50
k_p	1776.6821	1777.8785	1778.8699	1779.7622	1780.3249	1780.7254	1781.1835	1781.5522	1781.7590	1781.8113
k_i	1861.13	1872.78	1884.1	1895.19	1906.2	1917.09	1927.8	1938.24	1948.49	1958.58
k_d	1196.0895	1186.8757	1177.5438	1169.3869	1161.8054	1153.8281	1146.8600	1139.0542	1128.2913	1123.8390
ITAE	0.4185	0.4137	0.4094	0.4051	0.4014	0.3978	0.3941	0.3909	0.3874	0.3834
k_p	1549.0498	1547.8917	1546.2847	1544.6731	1542.7464	1540.8903	1539.3077	1537.6724	1535.7356	1533.8866
k_i	1823.92	1834.87	1845.63	1856.28	1866.7	1876.94	1886.9	1896.67	1906.2	1915.48
k_d	937.9230	929.3153	924.0203	915.3639	913.6385	910.4477	905.0513	898.1930	890.0407	885.7512
ITAE	0.3823	0.3784	0.3747	0.3716	0.3680	0.3657	0.3633	0.3607	0.3578	0.3547
k_p	1239.891	1235.7602	1231.8240	1228.4335	1224.9051	1221.1923	1217.6971	1213.3510	1209.4635	1205.7286
k_i	1773.2265	1782.7512	1791.9965	1801.0133	1809.7758	1818.4911	1827.1829	1835.4570	1843.4588	1851.4277
k_d	695.0443	694.0839	694.8981	692.3219	690.3521	687.7467	680.6876	679.4883	677.3376	672.8864
ITAE	0.3455	0.3431	0.3410	0.3390	0.3371	0.3351	0.3332	0.3308	0.3290	0.3273
k_p	794.6948	790.8298	786.0872	782.3929	778.5222	774.7441	773.8874	770.6774	768.4051	766.5677
k_i	1658.0665	1663.8882	1669.4350	1674.5328	1680.0133	1685.2032	1691.0263	1696.2313	1701.2292	1706.2052
k_d	477.4655	473.9585	475.5838	473.1543	471.6689	479.2955	471.3283	472.5756	472.8698	469.7314
ITAE	0.3129	0.3119	0.3108	0.3099	0.3089	0.3081	0.3081	0.3066	0.3059	0.3053
k_p	234.5419	236.1666	231.2020	223.7869	222.8266	221.9264	217.9100	214.6800	212.7962	203.8562
k_i	1422.1564	1426.2317	1429.4966	1432.9693	1436.1422	1439.3406	1442.7973	1445.8560	1448.9412	1452.1572
k_d	189.3320	182.6423	176.8699	181.0975	180.7784	175.5650	173.7855	173.5443	166.5017	168.0728
ITAE	0.2834	0.2828	0.2815	0.2814	0.2805	0.2803	0.2799	0.2794	0.2790	0.278
k_p	1853.0245	1870.7917	1888.0308	1904.7752	1920.9697	1936.8514	1952.3143	1967.4024	1982.0367	1996.258
k_i	2143.9990	2154.3230	2164.6093	2174.7354	2184.8243	2194.6186	2204.3095	2213.8494	2223.3491	2232.7872
k_d	2607.4920	2594.4965	2581.0246	2567.3634	2556.5927	2545.7188	2532.9855	2520.1064	2509.4746	2499.6158
ITAE	0.7114	0.6995	0.6878	0.6760	0.6653	0.6558	0.6464	0.6365	0.6270	0.6188

Table A.16 : ***Algorithm 2*** progress while using ITSE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 1-10)

Parameter	Iteration									
	1	2	3	4	5	6	7	8	9	10
k_p	110.1492	117.7269	123.4202	127.5088	130.1864	131.6007	131.8819	131.1640	129.5952	127.3367
k_i	1011.1905	1017.8245	1024.0624	1029.9662	1035.5844	1040.9569	1046.1181	1051.0979	1055.9226	1060.6152
k_d	191.5417	178.4438	166.5490	155.8246	146.2757	137.9215	130.7664	124.7715	119.8337	115.7852
ITSE	0.0883	0.1048	0.0996	0.0956	0.0925	0.0900	0.0881	0.0867	0.0855	0.0846
k_p	14.1729	332.9940	347.2831	360.3694	372.4354	383.6237	394.0481	403.8010	412.9583	421.5836
k_i	1025.0054	973.0807	984.0922	994.5174	1004.4323	1013.8973	1022.9619	1031.6669	1040.0468	1048.1308
k_d	52.2358	780.6969	773.1519	765.9954	759.1798	752.6671	746.4262	740.4312	734.6603	729.0951
ITSE	0.0814	0.1771	0.1754	0.1681	0.1616	0.1560	0.1510	0.1465	0.1425	0.1388
k_p	18.3927	244.1481	258.8262	271.9179	283.7136	294.4271	304.2208	313.2213	321.5293	329.2265
k_i	1024.9833	988.8351	998.9050	1008.3947	1017.3892	1025.9540	1034.1410	1041.9925	1049.5430	1056.8215
k_d	53.4821	562.8425	553.2282	544.2064	535.6918	527.6188	519.9367	512.6044	505.5888	498.8626
ITSE	0.0804	0.1391	0.1609	0.1530	0.1465	0.1408	0.1359	0.1316	0.1278	0.1244
k_p	26.4501	147.7439	160.4431	171.1540	180.2827	188.1096	194.8368	200.6152	205.5607	209.7648
k_i	1024.9379	1006.7614	1015.0039	1022.7325	1030.0320	1036.9653	1043.5806	1049.9165	1056.0044	1061.8705
k_d	55.9768	321.3977	308.8024	297.1306	286.2515	276.0747	266.5366	257.5915	249.2066	241.3580
ITSE	0.0791	0.0936	0.1313	0.1240	0.1180	0.1132	0.1091	0.1056	0.1027	0.1001
k_p	45.7272	40.3606	33.1018	161.2341	175.0480	186.8572	197.0806	206.0078	213.8485	220.7596
k_i	1024.8180	1029.2015	1033.3838	1012.3720	1020.9452	1028.9831	1036.5745	1043.7853	1050.6659	1057.2561
k_d	62.8733	56.8362	52.8476	368.7838	356.6819	345.4483	334.9491	325.0897	315.8013	307.0324
ITSE	0.0783	0.0759	0.0742	0.0984	0.1385	0.1307	0.1244	0.1192	0.1148	0.111
k_p	86.9103	82.7483	77.9748	72.8681	67.5448	62.0334	56.3256	50.3977	44.2161	37.7345
k_i	1024.4771	1029.2354	1033.8639	1038.3818	1042.7970	1047.1113	1051.3240	1055.4333	1059.4361	1063.3284
k_d	87.9105	83.9060	80.5830	77.4700	74.3417	71.1095	67.7353	64.1920	60.4479	56.4588
ITSE	0.0857	0.0823	0.0812	0.0800	0.0789	0.0778	0.0766	0.0753	0.0740	0.0726

Table A.17 : ***Algorithm 2*** progress while using ITSE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 11-20)

Parameter	Iteration									
	11	12	13	14	15	16	17	18	19	20
k_p	124.5492	121.3749	117.9262	114.2826	110.4955	106.5955	102.5987	98.5121	94.3372	90.0717
k_i	1065.1949	1069.6767	1074.0715	1078.3870	1082.6283	1086.7985	1090.8998	1094.9334	1098.9001	1102.8002
k_d	112.4198	109.5309	106.9424	104.5226	102.1812	99.8603	97.5242	95.1512	92.7273	90.2432
ITSE	0.0837	0.0829	0.0821	0.0814	0.0806	0.0799	0.0791	0.0783	0.0776	0.0768
k_p	429.7302	437.4438	444.7633	451.7228	458.3516	464.6757	470.7180	476.4987	482.0359	487.3457
k_i	1055.9440	1063.5082	1070.8424	1077.9634	1084.8860	1091.6234	1098.1876	1104.5892	1110.8377	1116.9421
k_d	23.7197	718.5202	713.4845	708.6021	703.8634	699.2601	694.7847	690.4305	686.1912	682.0614
ITSE	0.1355	0.1324	0.1296	0.1270	0.1246	0.1223	0.1203	0.1183	0.1165	0.1147
k_p	336.3799	343.0454	349.2701	355.0942	360.5521	365.6737	370.4850	375.0089	379.2656	383.2732
k_i	1063.8527	1070.6576	1077.2545	1083.6592	1089.8859	1095.9469	1101.8533	1107.6150	1113.2410	1118.7393
k_d	492.4030	486.1905	480.2088	474.4436	468.8827	463.5153	458.3318	453.3241	448.4845	443.8064
ITSE	0.1213	0.1185	0.1160	0.1137	0.1116	0.1096	0.1078	0.1061	0.1046	0.1032
k_p	213.3014	216.2318	218.6082	220.4764	221.8777	222.8503	223.4305	223.6531	223.5521	223.1602
k_i	1067.5368	1073.0225	1078.3442	1083.5166	1088.5526	1093.4640	1098.2612	1102.9536	1107.5499	1112.0576
k_d	234.0279	227.2028	220.8709	215.0207	209.6397	204.7131	200.2227	196.1467	192.4595	189.132
ITSE	0.0979	0.0960	0.0944	0.0929	0.0917	0.0906	0.0897	0.0888	0.0881	0.0874
k_p	226.8621	232.2511	237.0032	241.1812	244.8374	248.0162	250.7557	253.0897	255.0481	256.6581
k_i	1063.5883	1069.6893	1075.5816	1081.2844	1086.8143	1092.1858	1097.4118	1102.5036	1107.4714	1112.3245
k_d	298.7438	290.9055	283.4941	276.4909	269.8811	263.6524	257.7939	252.2959	247.1489	242.3432
ITSE	0.1078	0.1049	0.1025	0.1003	0.0984	0.0967	0.0952	0.0938	0.0926	0.0916
k_p	130.2206	141.3632	150.6128	158.3356	164.7845	170.1416	174.5431	178.0944	180.8799	182.9699
k_i	1049.0752	1056.0846	1062.6742	1068.9116	1074.8476	1080.5216	1085.9653	1091.2050	1096.2627	1101.1576
k_d	275.1420	262.5341	250.8635	240.0170	229.9205	220.5253	211.8001	203.7251	196.2870	189.4756
ITSE	0.0797	0.1174	0.1111	0.1060	0.1019	0.0985	0.0956	0.0932	0.0912	0.0895

Table A.18 : ***Algorithm 2*** progress while using ITSE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 21-30)

Parameter	Iteration									
	21	22	23	24	25	26	27	28	29	30
k_p	85.7115	81.2503	76.6810	71.9946	67.1809	62.2277	57.1204	51.8410	46.3669	40.6685
k_i	1106.6340	1110.4013	1114.1018	1117.7349	1121.2998	1124.7954	1128.2203	1131.5724	1134.8494	1138.0479
k_d	87.6919	85.0667	82.3612	79.5682	76.6792	73.6842	70.5706	67.3227	63.9195	60.3329
ITSE	0.0760	0.0752	0.0744	0.0735	0.0727	0.0718	0.0708	0.0699	0.0689	0.0678
k_p	492.4428	497.3403	502.0500	506.5828	510.9486	515.1563	519.2142	523.1300	526.9107	530.5627
k_i	1122.9103	1128.7494	1134.4664	1140.0672	1145.5575	1150.9427	1156.2275	1161.4164	1166.5137	1171.5233
k_d	678.0360	674.1104	670.2803	666.5420	662.8917	659.3263	655.8425	652.4377	649.1091	645.8543
ITSE	0.1131	0.1116	0.1101	0.1087	0.1074	0.1062	0.1050	0.1039	0.1029	0.1018
k_p	387.0476	390.6033	393.9534	397.1097	400.0830	402.8831	405.5191	407.9992	410.3314	412.5226
k_i	1124.1172	1129.3816	1134.5385	1139.5936	1144.5520	1149.4185	1154.1977	1158.8936	1163.5100	1168.0505
k_d	439.2838	434.9112	430.6836	426.5965	422.6456	418.8271	415.1371	411.5723	408.1294	404.8053
ITSE	0.1018	0.1006	0.0994	0.0983	0.0973	0.0963	0.0954	0.0945	0.0937	0.093
k_p	222.5090	221.6289	220.5483	219.2934	217.8883	216.3543	214.7104	212.9730	211.1561	209.2715
k_i	1116.4836	1120.8341	1125.1147	1129.3300	1133.4845	1137.5819	1141.6254	1145.6178	1149.5618	1153.4593
k_d	186.1326	183.4276	180.9830	178.7651	176.7417	174.8829	173.1611	171.5519	170.0340	168.5889
ITSE	0.0868	0.0862	0.0857	0.0852	0.0847	0.0842	0.0838	0.0833	0.0829	0.0825
k_p	257.9450	258.9322	259.6421	260.0961	260.3145	260.3170	260.1225	259.7490	259.2136	258.5327
k_i	1117.0712	1121.7189	1126.2747	1130.7448	1135.1349	1139.4503	1143.6959	1147.8759	1151.9943	1156.0548
k_d	237.8683	233.7127	229.8638	226.3076	223.0286	220.0104	217.2353	214.6850	212.3409	210.1843
ITSE	0.0906	0.0897	0.0890	0.0883	0.0876	0.0870	0.0865	0.0860	0.0855	0.085
k_p	184.4262	185.3049	185.6598	185.5435	185.0079	184.1045	182.8829	181.3904	179.6702	177.7609
k_i	1105.9062	1110.5233	1115.0220	1119.4140	1123.7097	1127.9186	1132.0488	1136.1074	1140.1008	1144.034
k_d	83.2795	177.6833	172.6644	168.1917	164.2251	160.7162	157.6111	154.8527	152.3845	150.1527
ITSE	0.0881	0.0869	0.0859	0.0850	0.0842	0.0836	0.0829	0.0824	0.0819	0.0814

Table A.19 : ***Algorithm 2*** progress while using ITSE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 31-40)

Parameter	Iteration									
	31	32	33	34	35	36	37	38	39	40
k_p	136.7326	146.9205	155.5395	162.8682	169.1061	174.4009	178.8656	182.5892	185.6437	188.0891
k_i	1123.1933	1129.0928	1134.6779	1139.9952	1145.0808	1149.9629	1154.6648	1159.2058	1163.6023	1167.8685
k_d	283.8814	272.5240	261.9382	252.0386	242.7667	234.0816	225.9551	218.3674	211.3042	204.7543
ITSE	0.0788	0.1085	0.1035	0.0993	0.0959	0.0930	0.0905	0.0884	0.0867	0.0852
k_p	534.0920	537.5042	540.8046	543.9978	547.0885	550.0808	552.9787	555.7859	558.5059	561.1419
k_i	1176.4488	1181.2936	1186.0610	1190.7539	1195.3752	1199.9276	1204.4135	1208.8354	1213.1954	1217.4958
k_d	642.6709	639.5568	636.5100	633.5286	630.6107	627.7546	624.9588	622.2218	619.5420	616.9182
ITSE	0.1009	0.0999	0.0991	0.0982	0.0974	0.0966	0.0959	0.0951	0.0945	0.0938
k_p	414.5796	416.5087	418.3157	420.0061	421.5851	423.0576	424.4284	425.7018	426.8821	427.9733
k_i	1172.5186	1176.9172	1181.2494	1185.5178	1189.7251	1193.8738	1197.9660	1202.0041	1205.9899	1209.9256
k_d	401.5969	398.5014	395.5160	392.6378	389.8643	387.1927	384.6205	382.1450	379.7637	377.4739
ITSE	0.0923	0.0916	0.0909	0.0903	0.0897	0.0892	0.0887	0.0882	0.0877	0.0872
k_p	207.3293	205.3375	203.3030	201.2310	199.1261	196.9916	194.8302	192.6441	190.4348	188.2034
k_i	1157.3124	1161.1227	1164.8915	1168.6202	1172.3099	1175.9617	1179.5763	1183.1548	1186.6977	1190.2058
k_d	167.2007	165.8562	164.5443	163.2558	161.9831	160.7200	159.4615	158.2032	156.9420	155.6749
ITSE	0.0821	0.0817	0.0812	0.0808	0.0805	0.0801	0.0797	0.0793	0.0789	0.0785
k_p	257.7212	256.7934	255.7622	254.6393	253.4355	252.1604	250.8226	249.4296	247.9881	246.5039
k_i	1160.0607	1164.0149	1167.9202	1171.7789	1175.5932	1179.3652	1183.0965	1186.7889	1190.4436	1194.0622
k_d	208.1966	206.3600	204.6573	203.0725	201.5906	200.1979	198.8819	197.6314	196.4362	195.2876
ITSE	0.0846	0.0842	0.0838	0.0834	0.0831	0.0827	0.0823	0.0820	0.0817	0.0813
k_p	175.6958	173.5031	171.2057	168.8222	166.3671	163.8516	161.2841	158.6708	156.0162	153.3234
k_i	1147.9114	1151.7369	1155.5132	1159.2429	1162.9281	1166.5704	1170.1711	1173.7314	1177.2523	1180.7345
k_d	148.1088	146.2103	144.4214	142.7122	141.0590	139.4426	137.8483	136.2643	134.6817	133.0938
ITSE	0.0809	0.0804	0.0800	0.0795	0.0791	0.0786	0.0782	0.0777	0.0773	0.0769

Table A.20 : ***Algorithm 2*** progress while using ITSE as cost function for initial $Kp = 0$, 5, 11, 22, 47 and 100, respectively (iterations: 41-50)

Parameter	Iteration									
	41	42	43	44	45	46	47	48	49	50
k_p	189.9772	191.3542	192.2627	192.7430	192.8341	192.5742	192.0006	191.1494	190.0551	188.7501
k_i	1172.0167	1176.0582	1180.0026	1183.8589	1187.6350	1191.3382	1194.9749	1198.5509	1202.0714	1205.5409
k_d	198.7075	193.1524	188.0749	183.4570	179.2758	175.5035	172.1076	169.0520	166.2985	163.8081
ITSE	0.0839	0.0828	0.0818	0.0810	0.0803	0.0797	0.0791	0.0787	0.0782	0.0778
k_p	563.6971	566.1744	568.5765	570.9060	573.1654	575.3571	577.4833	579.5461	581.5476	583.4896
k_i	1221.7385	1225.9254	1230.0584	1234.1391	1238.1693	1242.1504	1246.0840	1249.9714	1253.8141	1257.6134
k_d	614.3489	611.8330	609.3693	606.9567	604.5939	602.2801	600.0141	597.7951	595.6220	593.4941
ITSE	0.0931	0.0925	0.0919	0.0914	0.0908	0.0903	0.0898	0.0893	0.0888	0.0883
k_p	428.9793	429.9038	430.7503	431.5221	432.2227	432.8551	433.4223	433.9273	434.3728	434.7616
k_i	1213.8128	1217.6534	1221.4490	1225.2011	1228.9113	1232.5809	1236.2113	1239.8039	1243.3597	1246.8799
k_d	375.2733	373.1591	371.1288	369.1800	367.3100	365.5164	363.7967	362.1482	360.5686	359.0554
ITSE	0.0868	0.0864	0.0860	0.0856	0.0852	0.0849	0.0846	0.0842	0.0839	0.0836
k_p	185.9509	183.6777	181.3841	179.0704	176.7364	174.3821	172.0072	169.6113	167.1940	164.755
k_i	1193.6797	1197.1200	1200.5272	1203.9019	1207.2444	1210.5554	1213.8351	1217.0839	1220.3023	1223.4905
k_d	154.3999	153.1149	151.8186	150.5095	149.1867	147.8493	146.4963	145.1272	143.7413	142.338
ITSE	0.0781	0.0778	0.0774	0.0770	0.0767	0.0763	0.0759	0.0756	0.0752	0.0748
k_p	244.9820	243.4268	241.8421	240.2311	238.5965	236.9406	235.2655	233.5726	231.8634	230.1391
k_i	1197.6456	1201.1951	1204.7116	1208.1960	1211.6491	1215.0717	1218.4645	1221.8281	1225.1631	1228.4702
k_d	194.1777	193.0996	192.0473	191.0156	190.0001	188.9968	188.0024	187.0142	186.0295	185.0463
ITSE	0.0810	0.0807	0.0803	0.0800	0.0797	0.0794	0.0791	0.0788	0.0785	0.0782
k_p	150.5946	147.8308	145.0327	142.2004	139.3337	136.4319	133.4942	130.5197	127.5071	124.4551
k_i	1184.1787	1187.5856	1190.9556	1194.2891	1197.5866	1200.8485	1204.0749	1207.2662	1210.4225	1213.5441
k_d	131.4951	129.8816	128.2500	126.5979	124.9231	123.2240	121.4989	119.7466	117.9657	116.1551
ITSE	0.0764	0.0760	0.0756	0.0751	0.0747	0.0742	0.0738	0.0734	0.0729	0.0725

CURRICULUM VITAE

Mohammed ELBADRI

EDUCATION:

- **B.Sc.:** 2022, Istanbul University, Engineering Faculty, Electrical and Electronic Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2021 Erasmus+ Mundus Exchange program at Lodz University of Technology
- 2022-2023 System engineer at BorgWarner Inc.
- 2023-Present Cybersecurity validation engineer at Phinia Inc.

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Mohammed Elbadri Ahmed Hassan**, Ahmad Irham Jambak, Ismail Bayezit, Mehmet Turan Soylemez, Robust Control Performance in Uncertain Environments: A Semi-Heuristic Gradient Descent Approach. *2023 8th International Conference on Instrumentation, Control, and Automation (ICA)*, August 9-10, 2023, Jakarta, Indonesia