UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

EDER HENRIQUE NUNES DA SILVA

PARALLEL COMPUTING FOR REAL-TIME CONTROL: REINFORCEMENT LEARNING-BASED MODEL PREDICTIVE CONTROL OF INVERTED PENDULUM SYSTEMS

DISSERTATION

CORNÉLIO PROCÓPIO
2025

EDER HENRIQUE NUNES DA SILVA

PARALLEL COMPUTING FOR REAL-TIME CONTROL: REINFORCEMENT LEARNING-BASED MODEL PREDICTIVE CONTROL OF INVERTED PENDULUM SYSTEMS

Computação Paralela para Controle Em Tempo Real: Controle Preditivo Baseado Em Modelos de Aprendizado por Reforço de Sistemas de Pêndulo Invertido

Dissertação apresentada como requisito para obtenção do título de Mestre em Engenharia Elétrica, do Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Dr. Cristiano Marcos Agulhari

CORNÉLIO PROCÓPIO

2025

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio

EDER HENRIQUE NUNES DA SILVA

PARALLEL COMPUTING FOR REAL-TIME CONTROL: REINFORCEMENT LEARNING-BASED MODEL PREDICTIVE CONTROL OF INVERTED PENDULUM SYSTEMS

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Sistemas Eletrônicos Industriais.

Data de aprovação: 07 de Agosto de 2025

Dr. Cristiano Marcos Agulhari, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Luciano Antonio Frezzato Santos, Doutorado - Escola Politécnica da USP

Dr. Wesley Angelino De Souza, Doutorado - Universidade Tecnológica Federal do Paraná

RESUMO

SILVA, Eder. **Computação Paralela para Controle em Tempo Real: Controle Preditivo de Sistemas de Pêndulo Invertido com Base em Aprendizado por Reforço**. 2025. 78 f. Dissertation (Master's Degree em Engenharia Elétrica) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2025.

Este trabalho investiga a aplicação da computação paralela em sistemas de controle, propondo uma estrutura de Controle Preditivo Baseado em Aprendizado por Reforço (RLMPC). O foco está no desafio de controlar um sistema de pêndulo invertido, por meio da integração entre o Aprendizado por Reforço (RL) e o Controle Preditivo Modelado (MPC), com o objetivo de otimizar o desempenho e garantir a estabilidade em tempo real. Como parte desta dissertação, foi projetado e construído um sistema físico de pêndulo invertido utilizando componentes industriais, assegurando robustez, confiabilidade e condições realistas de operação. Essa plataforma experimental foi essencial para a validação prática da estrutura proposta. A estrutura RLMPC emprega paralelização para resolver problemas de otimização online, possibilitando respostas rápidas e decisões eficazes. A implementação paralela mostrou-se eficiente no atendimento às exigências computacionais, viabilizando a aplicação em sistemas com restrições de tempo real. Este estudo contribui para o avanço da teoria de controle ao combinar abordagens baseadas em modelos e orientadas a dados, promovendo maior adaptabilidade frente às incertezas dos ambientes industriais.

Palavras-chave: Computação em cluster. Aprendizado por reforço. Controle preditivo baseado em modelo. Pêndulo invertido. Processamento paralelo.

ABSTRACT

SILVA, Eder. **Parallel Computing for Real-Time Control: Reinforcement Learning-Based Model Predictive Control of Inverted Pendulum Systems**. 2025. 78 p. Dissertation (Master's Degree in Course Name) – Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2025.

This work investigates the application of parallel computing in control systems, proposing a Reinforcement Learning-Based Model Predictive Control (RLMPC) framework. The focus lies on addressing the challenge of controlling an inverted pendulum system by integrating Reinforcement Learning (RL) and Model Predictive Control (MPC) to optimize performance and ensure real-time stability. As part of this dissertation, a physical inverted pendulum system was designed and assembled using industrial grade components that ensure robustness, reliability, and realistic operating conditions. This experimental platform played a key role in the validation of the proposed control structure in practice. The RLMPC framework employs parallelization to solve online optimization problems, enabling fast responses and effective decision-making. The parallel implementation proved to be efficient in meeting computational demands, allowing deployment in systems with real-time constraints. This study contributes to control theory by combining model-based and data-driven approaches, enhancing adaptability in uncertain industrial environments.

Keywords: Cluster computing. Reinforcement learning. Model predictive control. Inverted pendulum. Parallel processing.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 OVERVIEW

The ability to control complex and unpredictable systems, like the inverted pendulum, has long been a central challenge in control engineering. This iconic system, known for its rapid, nonlinear behavior and inherent instability, serves as a crucial benchmark for developing and evaluating advanced control strategies (BOUBAKER, 2013). While traditional methods such as linear models and Model Predictive Control (MPC) have proven effective in controlled environments, they often struggle when faced with the dynamic and uncertain conditions prevalent in real-world applications (BUSCHERMÖHLE *et al.*, 2024; SINHA *et al.*, 2022). This limitation stems from their reliance on precise system models, which are often unavailable or difficult to maintain.

Recent advancements in Artificial Intelligence (AI) and Machine Learning (ML) offer promising solutions to these limitations (JANIESCH *et al.*, 2021). Specifically, Reinforcement Learning (RL), a branch of ML where an agent learns optimal control policies through direct interaction with its environment, is particularly well-suited for nonlinear systems like the inverted pendulum (DONGE *et al.*, 2023). Unlike traditional control paradigms, RL operates without an explicit system model, making it highly adaptable to unknown dynamics and disturbances (MOOS *et al.*, 2022). However, a standalone RL approach often lacks the critical guarantees of stability and constraint satisfaction, which are non-negotiable for safe and reliable real-world deployment (TURCHETTA, 2021).

To bridge this crucial gap, this dissertation introduces a novel Reinforcement Learning-based Model Predictive Control (RLMPC) framework. This innovative approach synergistically combines the adaptive, model-free learning capabilities of RL with the robust stability and constraint-handling strengths of MPC (LIN *et al.*, 2023). At each control step, MPC performs multiple real-time simulations using a deterministic model of the system, seeking the optimal input that minimizes a predefined cost while satisfying constraints. The MPC component ensures that the system rigorously adheres to its physical limits and maintains stability across diverse operating conditions (EKAPUTRI; SYAICHU-ROHMAN, 2012). Simultaneously, the RL component empowers the controller to dynamically learn and adapt to unknown system dynamics and disturbances (HWANGBO *et al.*, 2017). This integration of model-free and model-based le-

arning strikes an optimal balance between adaptability and robustness, representing a significant advancement over conventional control methodologies (GINZBURG-GANZ *et al.*, 2024).

A key and distinctive aspect of this research is the physical construction of an industrial-grade inverted pendulum system. This system is meticulously designed to replicate the complexities of real-world industrial environments, featuring high-precision actuators, reliable incremental rotary encoders, and robust mechanical components. This ensures continuous, stable operation and the ability to withstand external disturbances and significant mechanical loads, providing a realistic platform for rigorous testing and validation of advanced control strategies. By building with industrial equipment, we aim to bridge the gap between academic research and practical industrial demands, fostering solutions directly applicable to real-world scenarios.

The foundation of the proposed control framework lies in state-space modeling, which effectively captures the complex nonlinear dynamics of the inverted pendulum. Critical state variables, such as cart position, cart velocity, pendulum angle, and angular velocity, are precisely modeled to represent the system's behavior (University of Michigan, 2025). To address measurement noise and provide accurate real-time feedback, a Kalman Filter (KF) is integrated for states estimation, substantially improving control performance in noisy operational conditions (FARAGHER, 2012).

Implementing the RLMPC framework in real-time presents considerable computational challenges, primarily due to the inherent complexity of online optimization and policy learning. To overcome these hurdles, this dissertation leverages parallel computing. By distributing computationally intensive tasks across multiple processes, the RLMPC framework efficiently handles large-scale optimization problems and processes sensor data in real time.

To enable real-time optimization, it is proposed the use of a supercomputer, implemented as a heterogeneous computing cluster. This approach aims not only to provide the necessary processing capacity to efficiently distribute optimization tasks but also to ensure economic viability and low-cost implementation.

The cluster will consist of 36 processing cores, carefully selected to maximize performance and flexibility. The heterogeneous nature of the cluster will allow the integration of different types of hardware, optimizing the execution of specific real-time optimization algorithms. This distributed architecture will enable the system to process large volumes of data and perform complex calculations in parallel, a fundamental requirement for rapid decision-making in dynamic environments. This high-performance computing infrastructure is crucial for the

system to respond quickly to dynamic changes and maintain operational stability, even under demanding conditions. The distributed processing capability and flexibility of the heterogeneous cluster make it particularly well-suited for practical industrial applications, where real-time optimization is a competitive advantage.

This dissertation begins by detailing the filtering process, which is essential for noise reduction within the system. We then systematically transition from the foundational Linear Quadratic Regulator (LQR) control method to Model Predictive Control (MPC), ensuring that readers from diverse backgrounds can readily grasp the underlying principles. The motivation for adopting an RLMPC approach stems from the need to harness MPC's anticipative capabilities and constraint-handling strengths, particularly for systems characterized by fast dynamics.

Significant advantages of predictive control are often offset by the challenges of solving non-convex constrained optimization problems in real time and the absence of general stability guarantees. This work directly addresses these limitations. The proposed framework leverages parallelized learning and hardware-in-the-loop testing to enhance the feasibility and robustness of predictive control in demanding industrial settings.

Crucially, the use of parallel computing addresses the trade-off between model fidelity and computational tractability by enabling a fast, iterative nonlinear MPC algorithm. Through the parallel reuse of parameter trajectories across control cycles, the framework achieves substantial reductions in computation time while preserving system stability and responsiveness, key requirements for real-time industrial deployment.

2 BACKGROUND

2.1 DYNAMICAL SYSTEMS

The state-space representation for a given system is not unique, except that the number of state variables is the same for any of the different state-space representations of the same system (OGATA, 2010). Therefore, a nth-order system may be described by

$$\begin{aligned}
dx_1(t) &= f_1(t; x_1(t), x_2(t), x_3(t), \ldots, x_n(t); u_1(t), u_2(t), u_3(t), \ldots, u_r(t))dt, \\
dx_2(t) &= f_2(t; x_1(t), x_2(t), x_3(t), \ldots, x_n(t); u_1(t), u_2(t), u_3(t), \ldots, u_r(t))dt, \\
dx_3(t) &= f_3(t; x_1(t), x_2(t), x_3(t), \ldots, x_n(t); u_1(t), u_2(t), u_3(t), \ldots, u_r(t))dt, \\
&\vdots \\
dx_n(t) &= f_n(t; x_1(t), x_2(t), x_3(t), \ldots, x_n(t); u_1(t), u_2(t), u_3(t), \ldots, u_r(t))dt,
\end{aligned} \tag{1}$$

where $x(t)$ and $u(t)$ denote the state and input vectors, respectively, which will be formally defined in the following section. Subject to certain continuity conditions on the functions f_i, they uniquely determine the dynamic system over the time interval $t_0 \leq t \leq t_f$, with initial time t_0 and final time t_f.

The output function $y(t)$ describes the system's response as a function of the state variables $x_i(t)$ and input variables $u_j(t)$. In a general state-space representation, the output equation is given by

$$y(t) = g(t, x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_r(t)), \tag{2}$$

where the function $g(\cdot)$ characterizes how the system's states and inputs map to the output, depending on the specific system dynamics and measurement equations.

In state-space analysis, there are three types of variables that are involved in the modeling of dynamic systems: input variables, output variables, and state variables (GREWAL; ANDREWS, 2014). The variables $x_1, \ldots, x_n$ are the *state variables* and represent the degrees of freedom of the dynamic system. These variables are collected into the *state vector* $x(t)$, defined as

$$x(t) = \begin{bmatrix} x_1(t) & x_2(t) & x_3(t) & \ldots & x_n(t) \end{bmatrix}^T,$$

which denotes the system state on $\mathbb{R}^n$. Furthermore, the state vector is subject to the constraint $x(t) \in \mathbb{X}$, where $\mathbb{X} \subseteq \mathbb{R}^n$ represents the admissible state-space of the system.

The variables $u_1, \ldots, u_r$ are the input coupling coefficients, which are collected into a single *input vector* $u(t)$, defined as

$$u(t) = \begin{bmatrix} u_1(t) & u_2(t) & u_3(t) & \ldots & u_r(t) \end{bmatrix}^T,$$

representing the control inputs on $\mathbb{R}^r$. Similarly, the input vector is constrained by $u(t) \in \mathbb{U}$, where $\mathbb{U} \subseteq \mathbb{R}^r$ defines the set of permissible control inputs.

The variables $y_1, \ldots, y_m$ are the *measured values* of the dynamic system, which are grouped into the *measurement vector* or *output vector* $y(t)$, defined as

$$y(t) = \begin{bmatrix} y_1(t) & y_2(t) & y_3(t) & \ldots & y_m(t) \end{bmatrix}^T,$$

where $y(t) \in \mathbb{R}^m$ represents the system output on $\mathbb{R}^m$. A convenient choice of state variables is to consider the outputs and their derivatives as states. However, in practice, this approach is often unsuitable due to the noise effects inherent in real-world systems, which can render higher-order derivative terms inaccurate (NISE, 2010).

To illustrate such a choice, consider a single input, single output nth-order system given by the following differential equation

$$\frac{d^n y(t)}{dt^n} + a_1(t)\frac{d^{n-1} y(t)}{dt^{n-1}} + \ldots + a_{n-1}(t)\frac{dy(t)}{dt} + a_n(t)y(t) = b(t)u(t), \tag{3}$$

let us define

$$\begin{aligned} y(t) &= x_1(t), \\ dy(t) &= x_2(t)dt, \\ d^2y(t) &= x_3(t)dt^2, \\ &\vdots \\ d^n y(t) &= x_n(t)dt^n, \end{aligned} \tag{4}$$

then Equation (4) can be written as

$$\begin{aligned} dx_1(t) &= x_2(t)dt, \\ dx_2(t) &= x_3(t)dt, \\ dx_3(t) &= x_4(t)dt, \\ &\vdots \\ dx_n(t) &= [-a_n x_1(t) - \ldots - a_1 x_n(t) + b(t)u(t)]dt, \end{aligned} \tag{5}$$

which, in turn, is equivalent to

$$\begin{aligned} dx_1(t) &= x_2(t)dt, \\ dx_2(t) &= x_3(t)dt, \\ dx_3(t) &= x_4(t)dt, \\ &\vdots \\ dx_{n-1}(t) &= x_n(t)dt, \\ dx_n(t) &= [a(x(t)) + b(x(t))u(t)]dt, \end{aligned} \tag{6}$$

where $a(x(t))$ and $b(x(t))$ are nonlinear terms that encapsulate the coefficients $a_1, \ldots, a_n$. The system can now be written compactly as

$$dx(t) = f(x(t),u(t))dt. \tag{7}$$

As noted earlier, the representation of a given system is not unique. In this dissertation, the form provided in Equation (7) was selected as standard. The state-space representation offers a structured framework for modeling and analyzing the behavior of dynamic systems. By expressing the system in terms of state variables, inputs, and outputs, this approach is both versatile and well-suited to a wide range of applications.

Sampling or discretizing nonlinear continuous-time state-space ODEs is a standard approach for obtaining forms suitable for computer-based control (NAIDU, 2003). The resulting discrete-time systems are typically described by the state-space difference equation

$$x_{k+1} = f_d(x_k,u_k), \tag{8}$$

where $x_k \in \mathbb{X} \subseteq \mathbb{R}^n$ represents the state of the discrete-time system, and $u_k \in \mathbb{U} \subseteq \mathbb{R}^r$ denotes the discrete control input at time $k \in \mathbb{N}$. Here, $\mathbb{X}$ and $\mathbb{U}$ are the admissible sets of states and control inputs, respectively. These systems satisfy the 1-step Markov property, as the state at time $k+1$ depends only on the state and control input at time k. In addition, we assume that system (8) satisfies the following conditions.

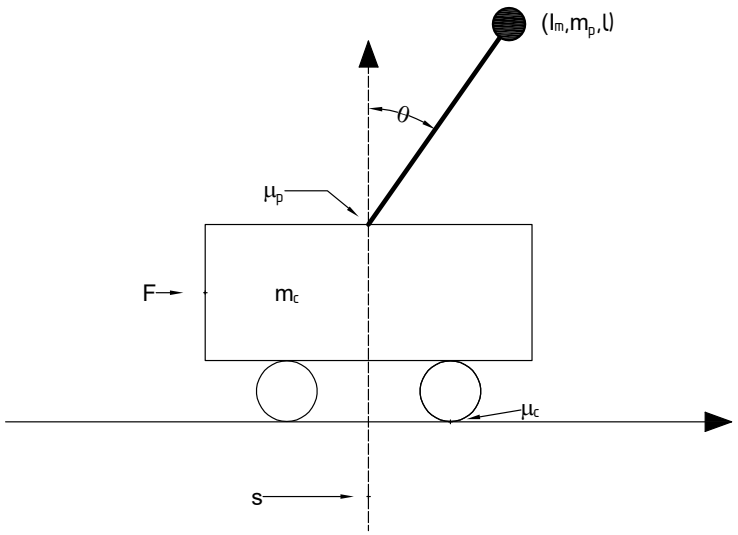
Assumption 1 ((LIN *et al.*, 2024))**.** *The function $f_d : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is continuous in the states and inputs, and satisfies $f_d(0,0) = 0$. Under the constraints $x_k \in \mathbb{X}$ and $u_k \in \mathbb{U}$, the system described by* (8) *is stabilizable at the equilibrium point $x_k = 0$, $u_k = 0$. Additionally, the system state x_k is assumed to be measurable.*

2.2 MATHEMATICAL MODELING OF THE INVERTED PENDULUM

The inverted pendulum problem originates from James Clerk Maxwell's 1868 seminal work, On Governors (Royal Society (London), 1869), where he first mathematically described it to explore the instability inherent in control systems. Its significance was re-established in the mid-20th century, particularly during the 1950s and 1960s, alongside the development of modern control theory. During this period, breakthroughs in electronics and computing transformed the inverted pendulum into a standard testbed for developing and evaluating control techniques (KOELEWIJN *et al.*, 2018; MINOUCHEHR; HOSSEINI-SANI, 2015; BALULA, 2016). For the current investigation, we employ the classical model articulated by (COŞKUN, 2020).

This mechanical system comprises a pendulum affixed to a cart, designed to swing freely in the vertical plane while the cart moves along a horizontal rail. The overarching control goal is twofold: to maintain the pendulum in its inverted (upright) position and to guide the cart to a predefined target location.

Figure 1 – Inverted pendulum system.



Source: Created by the author, based on (YILDIRAN, 2023).

Figure 1 illustrates the inverted pendulum system, composed of a cart that travels on a horizontal track and a pendulum rod that pivots freely from the cart's center. The system's dynamics are such that an external force, $F(t)$, dictates the cart's position, concurrently with gravity and damping forces acting upon the pendulum.

The behavior of the system is characterized by several physical parameters, including the masses of the cart and pendulum, the length of the pendulum rod, and the damping coefficients associated with the cart and pendulum. These parameters, along with their respective units, are summarized in Table 1.

Table 1 – Physical parameters of the inverted pendulum.

Symbol	Description	Unit
m_c	Mass of the cart	kg
m_p	Mass of the pendulum rod	kg
ℓ	Length of the pendulum rod	m
g	Acceleration due to gravity	$\mathrm{m/s^2}$
μ_c	Cart damping coefficient	N · m/s
μ_p	Pendulum damping coefficient	N · m · s
I_m	Rotary inertia of the pendulum rod	kg · m ^{2}

Source: Created by the author.

The motion of the inverted pendulum system is governed by nonlinear differential equations derived using Newton's second law, which relates forces and accelerations (TIGA *et al.*, 2019; GHANAVATI *et al.*, 2011). The horizontal movement of the cart is described considering the applied force $F(t)$, the damping force due to the coefficient of the cart μ_c, and the coupling effect of the pendulum motion. The resulting equation is

$$\left(m_c + m_p\right) \ddot{s}(t) + \mu_c \dot{s}(t) + m_p \ell \ddot{\theta}(t) \cos(\theta(t)) - m_p \ell \left(\dot{\theta}(t)\right)^2 \sin(\theta(t)) = F(t). \tag{9}$$

The rotational dynamics of the pendulum arises from the torques acting on the pendulum rod, including the gravitational acceleration, the damping due to the pendulum coefficient μ_p, and the reaction forces from the cart motion. Applying Newton's second law for rotational motion yields

$$\left(I_m + m_p \ell^2\right) \ddot{\theta}(t) + \mu_p \dot{\theta}(t) + m_p \ell \ddot{s}(t) \cos(\theta(t)) - m_p g \ell \sin(\theta(t)) = 0. \tag{10}$$

In these Equations (9) and (10), $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ represent the first and second derivatives of the pendulum angle θ, respectively. The variable $s(t)$ (m) denotes the position of the cart, $\dot{s}(t)$ (m/s) is its velocity, $\theta(t)$ (rad) is the angle of the pendulum measured from the vertical, $\dot{\theta}(t)$ (rad/s) is the angular velocity and F (N) is the external force applied to the cart.

The state-space representation provides a structured framework for analyzing and modeling dynamic systems. By defining state variables, the nonlinear equations can be reformulated into a state-space format. The state variables include $x_1 = s(t)$ representing the position of the cart, $x_2(t) = \dot{s}(t)$ for its velocity, $x_3(t) = \theta(t)$ for the angle of the pendulum, and $x_4(t) = \dot{\theta}(t)$

for the angular velocity of the pendulum. These variables capture the essential dynamics of the system, enabling a systematic analysis of the interactions between the cart and pendulum. The dynamic coupling is evident, as the horizontal acceleration of the cart $\dot{x}_2(t)$ directly influences the angular motion of the pendulum $\dot{x}_4$, and vice versa, highlighting the complexity of the system.

To model these interactions explicitly, the nonlinear functions h_1, h_2, g_1, and g_2 are defined as follows,

$$h_1(x(t)) = \frac{-gm_p^2\ell^2 \sin(x_3)\cos(x_3) - (I_m + m_p\ell^2)\mu_c x_2}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2}$$

$$+\frac{\mu_p m_p \ell \cos(x_3) x_4 + (I_m + m_p\ell^2) m_p \ell \sin(x_3) x_4^2}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2},$$

$$g_1(x(t)) = \frac{(I_m + m_p\ell^2)}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2},$$

$$h_2(x(t)) = \frac{(m_c + m_p) g m_p \ell \sin(x_3) + \mu_c m_p \ell \cos(x_3) x_2}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2}$$

$$-\frac{\mu_p(m_c + m_p)x_4 - m_p^2\ell^2 \sin(x_3)\cos(x_3) x_4^2}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2},$$

$$g_2(x(t)) = \frac{m_p\ell\cos(x_3)}{(m_c + m_p)(I_m + m_p\ell^2) - (m_p\ell\cos(x_3))^2},$$

the dynamic equations (9) and (10) can then be expressed in a compact state-space form

$$\mathscr{H} := \begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = h_1(x(t)) + g_1(x(t))F(t) \\ \dot{x}_3(t) = x_4(t) \\ \dot{x}_4(t) = h_2(x(t)) + g_2(x(t))F(t) \end{cases} \tag{11}$$

This formulation encapsulates the intricate interactions between the cart and pendulum, providing a robust framework for systematic analysis and allowing the design of control strategies tailored to the system's behavior.

2.3 KALMAN FILTER

The Kalman filter (KF), named after Rudolf Emil Kálmán, provides a recursive solution to the discrete-data linear filtering problem (KALMAN, 1960). It is essential for estimating and predicting the state of dynamic systems, even in noisy environments, making it indispensable in aerospace, robotics, and signal processing. Its recursive structure enables real-time data processing, enhancing precision in control and decision-making for complex systems. By minimizing uncertainty, the KF has transformed navigation, tracking, and autonomous technologies, driving innovation across science and engineering. With its mathematical elegance and practical versatility, it continues to inspire groundbreaking research and applications across disciplines.

The KF utilizes the most recent system state values to update estimated values non-iteratively during subsequent sampling periods (MANKAR; VADIRAJACHARYA, 2015). Additionally, KF algorithms extend static state estimation techniques and are commonly used in signal processing and systems control (PEI *et al.*, 2019). The filter addresses two main types of error: prediction error (uncertainty in the system model) and measurement error (inaccuracy in observed measurements).

Despite various presentations in literature, KF is frequently used for stochastic estimation from noisy sensor measurements with Gaussian noise (LAARAIEDH, 2012). It is statistically efficient concerning any quadratic function of estimate errors, forming a recursive process that minimizes the sum of squared differences between measured and estimated values (SOUZA *et al.*, 2021). These estimators have emerged from the need to develop state estimators as computationally efficient as possible, suitable for real-time implementation (MANKAR; VADIRAJACHARYA, 2015).

The KF estimates the state $x \in \mathbb{R}^n$ of a discrete-time system governed by a linear stochastic difference equation (LAARAIEDH, 2012). The state x_k at sampling instant k evolves from the prior state at time $k-1$ as follows (JANABI-SHARIFI *et al.*, 2000)

$$x_k = \phi_k x_{k-1} + \Upsilon_k u_k + w_k, \tag{12}$$

where x_k represents the state at time k, x_{k-1} represents the state at the previous time step, $\phi_k \in \mathbb{R}^{n \times n}$ is the state transition matrix, $\Upsilon_k \in \mathbb{R}^{n \times r}$ is the input transition matrix, u represents known control inputs, and w is the process noise vector with zero mean and covariance Q. The observation model is

$$z_k = H x_k + v_k, \tag{13}$$

where z_k is the measurement vector at time k, $H \in \mathbb{R}^{n \times n}$ maps the state vector into the measurement domain, and v_k is the measurement noise vector with zero mean and covariance R.

In most linear dynamic systems applications, including the one considered in this work, the w_k and v_k noises are uncorrelated white Gaussian noises caused by modeling errors and measurement inaccuracies (SOUZA *et al.*, 2021). The process and measurement noise covariance matrices $\bar{Q}_k$ and $\bar{R}_k$ are characterized as

$$E\langle w_k w_k^T \rangle = \bar{Q}_k, \quad E\langle v_k v_k^T \rangle = \bar{R}_k.$$

The KF utilizes the properties of Gaussian functions to optimally estimate the state of a system. Both the model's predictions and the measurements are represented as probability density functions (PDFs), which are Gaussian distributions. These PDFs capture the likelihood of the true state being near their respective means, with the associated variances indicating uncertainty (FARAGHER, 2012).

To compute the best estimate of the true state, the KF fuses information from the model prediction and the measurement by mathematically combining their respective Gaussian PDFs. Specifically, if y_1(prediction) and y_2 (measurement) are Gaussian PDFs, their product is also a Gaussian PDF. The prediction PDF is defined as

$$y_1(r; \mu_1, \sigma_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(r-\mu_1)^2}{2\sigma_1^2}},$$

and the measurement PDF is given by

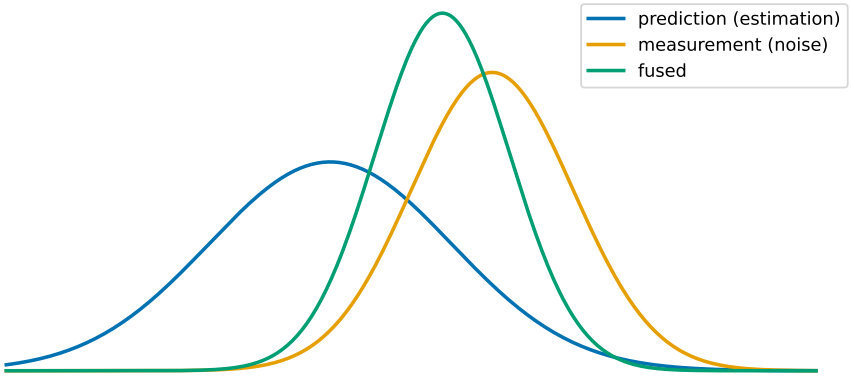
$$y_2(r; \mu_2, \sigma_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(r-\mu_2)^2}{2\sigma_2^2}}.$$

The product of these two Gaussian functions results in a fused Gaussian function

$$y_{\text{fused}}(r; \mu_1, \sigma_1, \mu_2, \sigma_2) = \frac{1}{2\pi\sqrt{\sigma_1^2\sigma_2^2}} e^{-\left(\frac{(r-\mu_1)^2}{2\sigma_1^2} + \frac{(r-\mu_2)^2}{2\sigma_2^2}\right)},$$

which combines the information from both the prediction and measurement, as illustrated in Figure 2. The fused Gaussian function has a mean that represents the optimal state estimate and a variance smaller than either of the individual variances. This reduction in uncertainty provides a more accurate estimate without increasing complexity. This principle underpins the recursive nature of the KF, where update equations are derived from the mathematical fusion of the prediction and measurement PDFs.

Figure 2 – Visualization of the Kalman Filter's fusion: blue for prediction, yellow for measurement, and green for the optimal fused estimate.



Source: Curves inspired by the work of Faragher (2012) (FARAGHER, 2012).

The resulting fused PDF remains Gaussian, characterized by an updated mean and variance

$$y(r;\mu_{\text{fused}},\sigma_{\text{fused}}) = \frac{1}{\sqrt{2\pi\sigma^2_{\text{fused}}}} e^{-\left(\frac{(r-\mu_{\text{fused}})^2}{2\sigma^2_{\text{fused}}}\right)}.$$

Each iteration of the KF involves two main steps: *prediction* and *update*, ensuring a continuously refined estimate of the system state.

Prediction step

The initial estimate $\hat{x}_{k|k-1}$, the a priori is based on the previous estimate $\hat{x}_{k-1|k-1}$, weighted by the state transition matrix ϕ_k, and control input matrix Υ_k

$$\hat{x}_{k|k-1} = \phi_k \hat{x}_{k-1|k-1} + \Upsilon_k u_k, \tag{14}$$

the a priori error covariance matrix $P_{k|k-1}$ is updated from its previous value $P_{k-1|k-1}$, and uses the transition matrix ϕ_k and process noise $\bar{Q}_k$

$$P_{k|k-1} = \phi_k P_{k-1|k-1} \phi_k^T + \bar{Q}_k. \tag{15}$$

Update step

The Kalman gain $\bar{K}_k$ is computed based on the a priori error covariance $P_{k|k-1}$, the observation matrix H, and measurement noise $\bar{R}_k$

$$\bar{K}_k = P_{k|k-1} H^T [H P_{k|k-1} H^T + \bar{R}_k]^{-1},$$

the posteriori state estimate $\hat{x}_{k|k}$ is calculated using the a priori estimate $\hat{x}_{k|k-1}$, Kalman gain $\bar{K}_k$, and measurement z_k

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \bar{K}_k [z_k - H\hat{x}_{k|k-1}].$$

The a posteriori error covariance $P_{k|k}$ is updated from $P_{k|k-1}$ and H

$$P_{k|k} = [I - K_k H]P_{k|k-1}.$$

The filter aims to ensure that $P \to 0$ as $k \to \infty$, though P may converge to a nonzero steady-state value in general (GREWAL; ANDREWS, 2014).

2.4 LINEAR QUADRATIC REGULATOR

The Linear Quadratic Regulator (LQR) is a foundational optimal control strategy for systems represented in state-space form. It is particularly effective for high-order systems or those with multiple inputs and outputs (MIMO), such as systems involving multiple sensors or actuators. By computing an optimal state-feedback gain matrix, LQR efficiently balances the trade-off between system performance and control effort. As a cornerstone of modern control theory, LQR provides an optimal solution for regulating linear dynamic systems with quadratic cost functions. Its development is closely tied to mid-20th-century advancements in optimal control and estimation theory. In 1960, Rudolf Emil Kálmán introduced seminal work on the design equations for LQR, offering a systematic method for determining optimal control laws for linear systems (KALMAN *et al.*, 1960).

At its core, LQR optimizes dynamic system behavior by minimizing a quadratic cost function that balances performance and control effort. This cost function, applicable to both continuous and discrete-time systems, penalizes deviations from desired states and excessive actuator usage. By solving the system dynamics in state-space form, LQR derives an optimal feedback control law to achieve efficient and effective regulation. For continuous-time systems, the cost function is expressed as

$$J(x(t),u(t)) = \int_0^\infty (\|x(t)\|_Q^2 + \|u(t)\|_R^2)dt. \tag{16}$$

Here, $x(t)$ represents the state variables of the system's output vector, $u(t)$ represents the control input vector, Q is a positive semi-definite matrix penalizing deviations of $x(t)$, and R is a positive definite matrix penalizing control effort. The goal is to minimize this cost function, ensuring the system output $x(t)$ closely tracks a reference $r(t)$ while minimizing control energy. When $r(t) \neq 0$, the cost function generalizes to $\|x(t) - r(t)\|_Q^2 + \|u(t)\|_R^2$.

For discrete-time systems, where decisions are made at discrete intervals, the cost function is given by

$$J_\infty(x_k,\mathbf{u}_{\infty,k}) = \sum_{k=0}^{\infty} \ell(x_k,u_k), \tag{17}$$

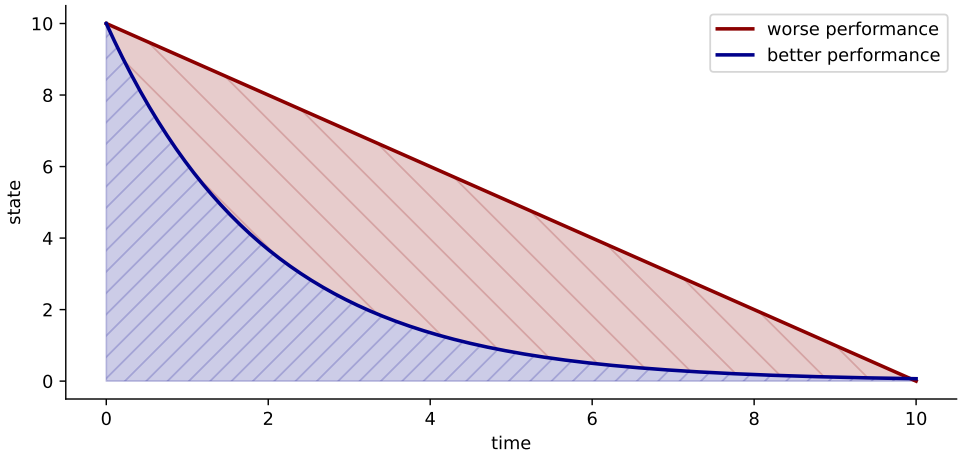
with a stage cost typically of the form

$$\ell(x_k,u_k) = \|x_k\|_Q^2 + \|u_k\|_R^2. \tag{18}$$

In this case, x_k represents the system's output and u_k represents the control input at step k. Q and R are equivalent penalty matrices for the discrete system. This discrete formulation is particularly suited for sampled-data environments, providing an optimal balance between system performance and control effort (CISNEROS, 2021). The LQR also plays a significant role in stochastic control extensions, as discussed by Athans (ATHANS, 1971).

Since states can take on both negative and positive values, squaring the terms in the cost function ensures that negative values do not offset the total cost. This approach also penalizes larger errors more heavily than smaller ones, aligning with practical control requirements where large deviations must be corrected quickly. The quadratic structure of the cost function provides a definitive minimum value, ensuring a well-posed optimization problem when paired with linear system dynamics (LEWIS *et al.*, 2012). The integral in the cost function from Equation (16) captures this behavior and is visually represented in Figure 3.

Figure 3 – Example of cost curves of Linear Quadratic Regulator.

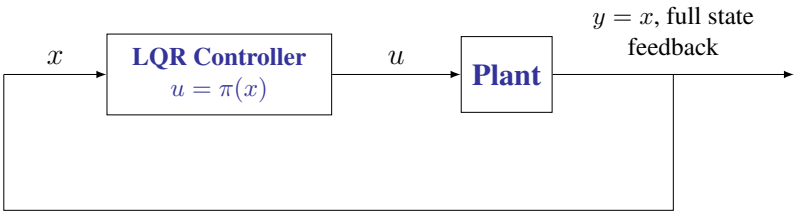


Source: Inspired by MATLAB's video on optimal control (MATLAB, 2025b).

An intuitive way to understand this concept is to consider a system where all states should return to zero, driving the system to its equilibrium point. A system starting from a nonzero state achieves better performance when it returns to equilibrium faster, as this corresponds to a lower cost. The area under the state trajectory curve quantifies this: a smaller area signifies quicker convergence to equilibrium and improved performance, while a larger area indicates prolonged deviations and a higher cost (ATHANS, 1971).

Balancing performance and control effort in LQR is achieved through careful selection of the weighting matrices Q and R. The matrix Q, positive semi-definite, emphasizes error reduction in critical states by assigning larger weights to their deviations. This encourages rapid stabilization of important states. Conversely, the positive definite matrix R penalizes excessive control effort, discouraging overuse of actuators to avoid inefficiencies or wear. The quadratic structure of the cost function ensures a well-defined minimum, enabling the calculation of optimal feedback gains that balance system performance and efficiency

Figure 4 – Schematic of Linear Quadratic Regulator controller.



Source: Diagram inspired by MathWorks (MATHWORKS, 2025).

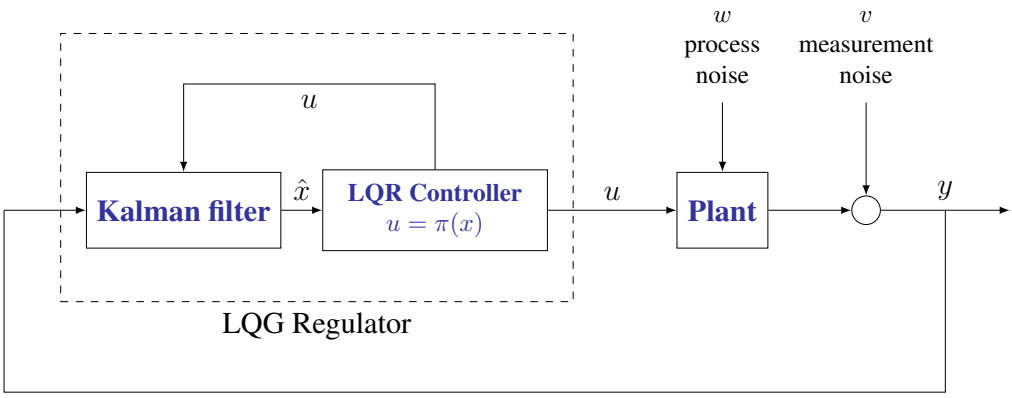
Designers iteratively refine these matrices based on simulations and system constraints. For example, diagonal entries in Q prioritize critical states, while R can account for actuator costs or limitations. By squaring the terms and ensuring nonnegative contributions, R helps to allocate control efforts effectively within practical constraints (LEWIS; VRABIE, 2009). A control policy, defined as a mapping from state-space to control space $\pi(\cdot)\mathbb{R}^n \rightarrow \mathbb{R}^r$, specifies the control action for each state x_k as

$$u_k = \pi(x_k). \tag{19}$$

These mappings are commonly implemented as feedback controllers, which adjust control actions dynamically based on the current state. In many systems, not all states are directly measurable. In such cases, an observer, like a Kalman filter, estimates system states from sensor

measurements. This allows the LQR framework to be extended to systems with partial state feedback, forming the Linear Quadratic Gaussian (LQG) controller.

Figure 5 – Schematic of Linear Quadratic Regulator controller with Kalman filter.



Source: Diagram inspired by MathWorks (MATHWORKS, 2025).

However, the original LQG theory does not inherently handle constraints on states, outputs, or control inputs. Despite this limitation, it provides a robust framework for optimal control of systems with linear dynamics and Gaussian noise (QIN; BADGWELL, 1997).

The LQR is a powerful and versatile control technique. Its ability to balance system performance and control effort through adjustable weighting matrices, coupled with the mathematical rigor of quadratic optimization, makes it a cornerstone of modern control applications. When integrated with state estimation methods like Kalman filters, the framework extends to broader applications, including systems with noisy or incomplete measurements.

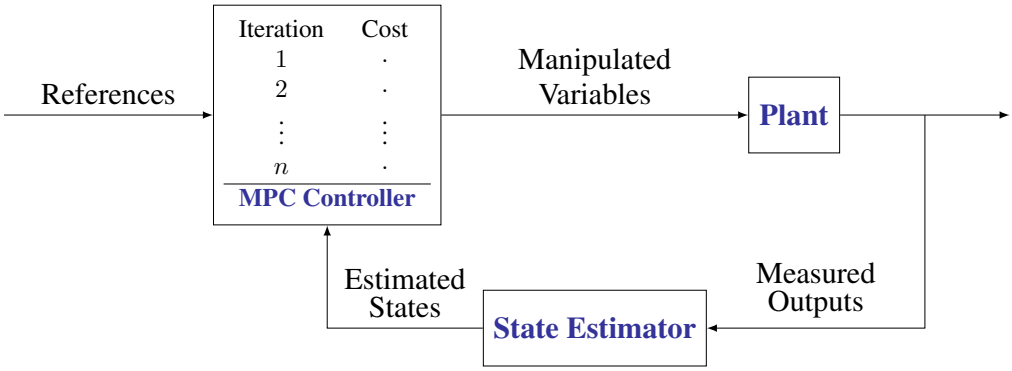
2.5 MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) is a versatile and widely adopted control methodology known for its ability to handle nonlinear and multivariable dynamics, enforce input and state constraints, and adapt to changing reference trajectories (BERBERICH *et al.*, 2022; BONGARD *et al.*, 2022).

Originally developed in the 1980s for process industries, MPC was designed to address challenges in controlling multivariable systems with constraints (HOLKAR; WAGHMARE, 2010). In the 1990s, its applications had expanded to sectors such as automotive, aerospace, and energy, where it optimized vehicle dynamics, flight control, and energy management (QIN; BADGWELL, 2003).

Advances in computational efficiency during the 2000s further broadened its adoption, enabling the control of nonlinear and time-varying systems in robotics, manufacturing, and other complex domains (HOLKAR; WAGHMARE, 2010; BERBERICH *et al.*, 2022).

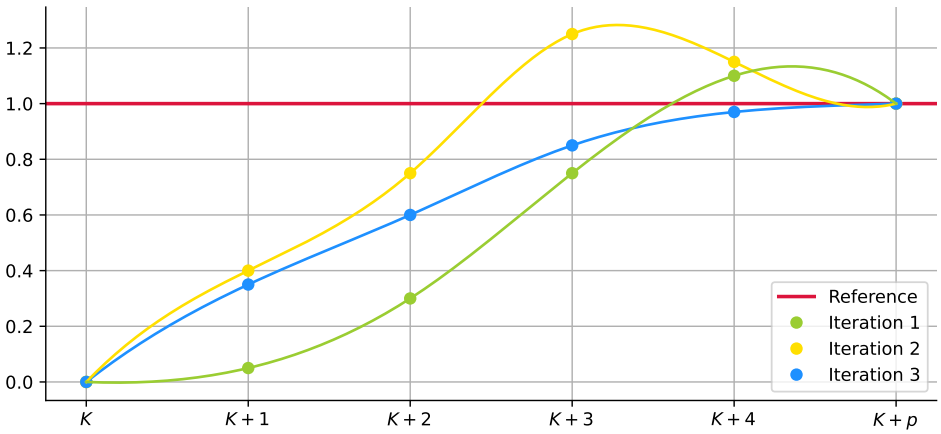
Figure 6 – Schematic of Model Predictive Control.



Source: Diagram inspired by MathWorks (MATHWORKS, 2025).

The principle of MPC is to operate by solving a constrained optimization problem within a receding horizon framework. At each sampling interval, MPC predicts future system behavior over a predefined prediction horizon (N) using a dynamic system model and the latest state measurements (PICHE *et al.*, 2000). By simulating potential future outcomes, MPC computes a sequence of control inputs that minimize a cost function, for instance the LQR cost function presented in Section 2.4, while satisfying system constraints. From the computed set of control inputs, the first sample of the one that generates the best predicted response is applied, and then the process is repeated, ensuring real-time adaptability. The operation of MPC is illustrated in Figure 7, which provides a visual representation of its receding horizon framework.

Figure 7 – Basic principle of model predictive control.



Source: Inspired by MATLAB's video (MATLAB, 2025a).

The optimization process minimizes a cost function over the prediction horizon, balancing system performance and efficiency. This cost function typically penalizes deviations from desired setpoints and excessive control efforts (REBLE, 2013). Using current state information, MPC predicts future outputs and computes control actions to minimize the deviation between predicted and desired trajectories while respecting input limits and safety constraints (PICHE *et al.*, 2000).

The explicit constraint management in MPC makes it particularly effective for multi-input, multi-output (MIMO) systems. It achieves this by solving an online optimization problem, often a quadratic program that adjusts control actions to guide the system toward desired trajectories. Variants such as adaptive, gain-scheduled, and nonlinear MPC extend its applicability to systems with specific requirements, including nonlinear dynamics, time-varying constraints, and varying computational budgets.

Despite challenges such as solving nonconvex optimization problems in real-time, MPC is valued for its closed-loop stability and performance guarantees. Advances in scalability and robustness have further enabled its deployment in fields like robotics, industrial automation, and energy systems, where it balances performance, constraint handling, and real-time adaptability. The optimization problem in MPC is defined as

$$\min_{\mathbf{u}_k} J(x_k,\mathbf{u}_k) = \sum_{i=0}^{N-1} \ell(x_{i|k},u_{i|k}) + F(x_{N|k}), \tag{20}$$

where $N \in \mathbb{N}_{>0}$ is the prediction horizon, $F(\cdot)$ is the terminal cost, $\mathbf{u}_k = \{u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}\}$ is the predicted control sequence of length N at time k. The terminal cost $F : \mathbb{X} \to \mathbb{R}_{\geq 0}$ ensures stability by penalizing deviations at the horizon endpoint. For a given current state $x_{0|k} = x_k$, the optimization problem is solved subject to the following constraints

$$\begin{aligned} x_{i+1|k} &= f(x_{i|k},u_{i|k}), \quad 0 \leq i \leq N-1, \\ u_{i|k} &\in \mathbb{U}, \quad 0 \leq i \leq N-1, \\ x_{i|k} &\in \mathbb{X}, \quad 0 \leq i \leq N-1, \\ x_{N|k} &\in \mathbb{X}_\Omega, \end{aligned} \tag{21}$$

where $f(x_{i|k},u_{i|k})$ represents system dynamics, $\mathbb{U}$ is the set of admissible control inputs, and $\mathbb{X}$ is the set of feasible states (CHEE *et al.*, 2023).

We obtain the policy $\pi(x_k)$ implicitly defined by (20). This process operates in a receding horizon framework, where only the first control action $u_{0|k}$ of the optimized sequence

is implemented in the system. The optimization is repeated at the next time step with updated state information.

Moreover, parameters of the running and terminal costs should be designed to satisfy

$$F(x_{N|k}) \geq \sum_{i=N}^{\infty} \ell(x_{i|k}, \pi(x_k)), \forall x \in \mathbb{X}_\Omega, \tag{22}$$

such that $F(\cdot)$ is a local Lyapunov function for $\pi(x_k)$ in $\mathbb{X}_\Omega$. To ensure recursive feasibility and stability, the prediction horizon N and the terminal constraints $\mathbb{X}_\Omega$ must be chosen to guarantee safe operation and optimal performance.

Determining the finishing cost in MPC is critical for ensuring stability and optimality of the control policy. However, this task is challenging due to complex system dynamics, inherent uncertainties, and significant computational demands. Traditional methods often struggle to accurately estimate the cost-to-go from the terminal state to the desired goal, especially in nonlinear or high-dimensional systems.

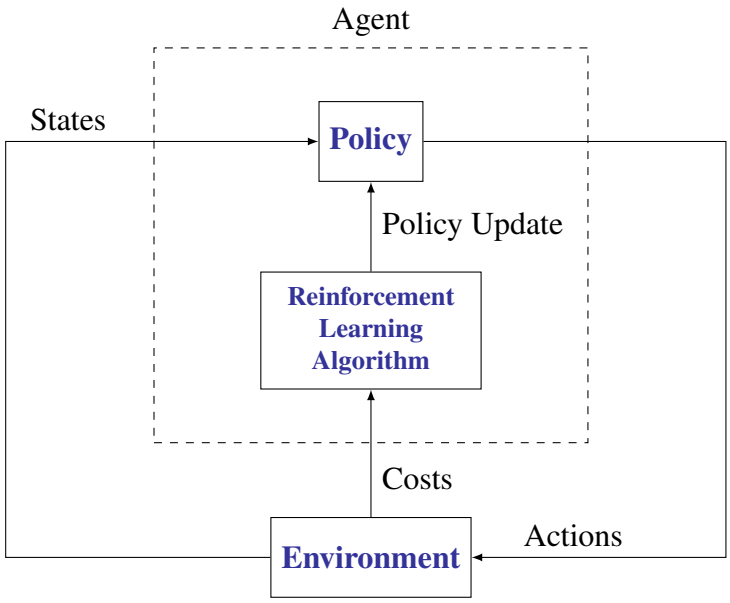
By leveraging RL, the cost-to-go function can be approximated through data-driven learning, enabling the system to better adapt to uncertainties and dynamic changes in its environment. Furthermore, RL can significantly reduce the computational burden of MPC by pre-computing or learning the finishing cost offline. This approach streamlines the online optimization process, making it more efficient and scalable for real-time applications.

2.6 REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a sophisticated machine learning framework designed to solve problems modeled as Markov Decision Processes (MDPs) (ZANON; GROS, 2020; SAWANT *et al.*, 2023). At its core, RL involves an agent that learns to make sequential decisions through iterative interactions with its environment. By observing the current state, taking actions, and receiving feedback in the form of rewards or penalties, the agent refines its strategy to maximize cumulative rewards over time.

Unlike supervised learning, which relies on labeled datasets, RL employs evaluative feedback to guide the agent's decision-making process. This approach enables adaptive, goal-oriented behavior, akin to survival strategies observed in nature (LEWIS; VRABIE, 2009).

Figure 8 – Schematic of Reinforcement Learning.



Source: Diagram inspired by MathWorks (MATHWORKS, 2025).

The RL framework has been instrumental in developing intelligent systems capable of addressing complex, dynamic problems across diverse domains, including robotics, finance, healthcare, and autonomous systems (ZANON; GROS, 2020; KIUMARSI *et al.*, 2014). Initially, RL was applied to optimal control tasks, such as discrete-time systems and linear quadratic regulator (LQR) problems, using techniques such as policy iteration (PI) (KIUMARSI *et al.*, 2014). Over time, its scope has expanded to encompass both discrete-time and continuous-time systems.

In modern applications, deep neural networks (DNNs) are widely employed as function approximators for RL policies. These networks enable efficient representation of action-value functions or direct policy mappings (ZANON; GROS, 2020; SAWANT *et al.*, 2023). However, the nonlinear and high-dimensional characteristics of DNNs introduce challenges, particularly in ensuring stability and safety in critical systems (ZANON; GROS, 2020; SAWANT *et al.*, 2023). Approaches like Nonlinear Model Predictive Control (NMPC) address these challenges by solving real-time optimization problems while respecting system constraints.

Central to RL is the optimization of value functions, which quantify the expected cumulative reward from a given state. In this work, the RL is applied to compute a valid terminal cost function $V_\pi(x_k)$, essential to properly execute the MPC control strategy.

For a policy π, the value function $V_\pi(x_k)$ is recursively defined as (LIN *et al.*, 2023),

$$V_\pi(x_k) = \ell(x_{i|k}, u_{i|k}) + V_\pi(x_{k+1}), \tag{23}$$

where $\ell(x_{i|k}, u_{i|k})$ is the immediate cost, and $V_\pi(x_{k+1})$ represents future rewards.

The Bellman equation, a cornerstone of RL, formalizes this recursion, defining the optimal value function $V_\pi^*(x_k)$ as

$$V_\pi^*(x_k) = \min_{u_{i|k}} \ell(x_{i|k}, u_{i|k}) + V_\pi^*(x_{k+1}), \tag{24}$$

the corresponding optimal policy $\pi^*(x_k)$ specifies the action minimizing cost

$$\pi^*(x_k) = \arg\min_{u_{i|k}} \ell(x_{i|k}, u_{i|k}) + V_\pi^*(x_{k+1}). \tag{25}$$

Despite its theoretical elegance, RL faces practical challenges, including high computational complexity, the curse of dimensionality, and difficulties in approximating solutions for continuous domains. Techniques such as Approximate Dynamic Programming (ADP), and Q-learning have been developed to address these issues. ADP, for instance, leverages temporal-difference (TD) learning to iteratively update value functions

$$V_\pi(x_k) \leftarrow V_\pi(x_k) + \alpha[V_{TD}(x_k) - V_\pi(x_k)], \tag{26}$$

where α is the learning rate and $V_{TD}(x_k)$ is the TD estimate. Similarly, Q-learning iteratively approximates the Q-function, enabling policy optimization in continuous spaces through parameterized models such as DNNs (BRADTKE, 1992). The versatility of RL is evident in its wide-ranging applications, from enabling adaptive navigation in robotics to optimizing treatments in healthcare and strategies in finance under uncertainty. As RL continues to evolve, hybrid approaches that combine model-based and model-free techniques are emerging to address challenges related to scalability, accuracy, and generalization. Future research is likely to focus on enhancing computational efficiency, improving safety guarantees, and exploring advances in transfer learning and deep reinforcement learning, further solidifying the role of RL in intelligent decision-making systems.

3 SYSTEM CONSTRUCTION

3.1 MECHANICAL SYSTEM DESIGN

This project began with extensive research and analysis of inverted pendulum models from academic literature. To guarantee robustness and reliability, all components were manufactured using 100% industrial grade materials. This ensures superior mechanical strength, durability, and performance in real-world conditions, while also adhering to industrial safety standards.

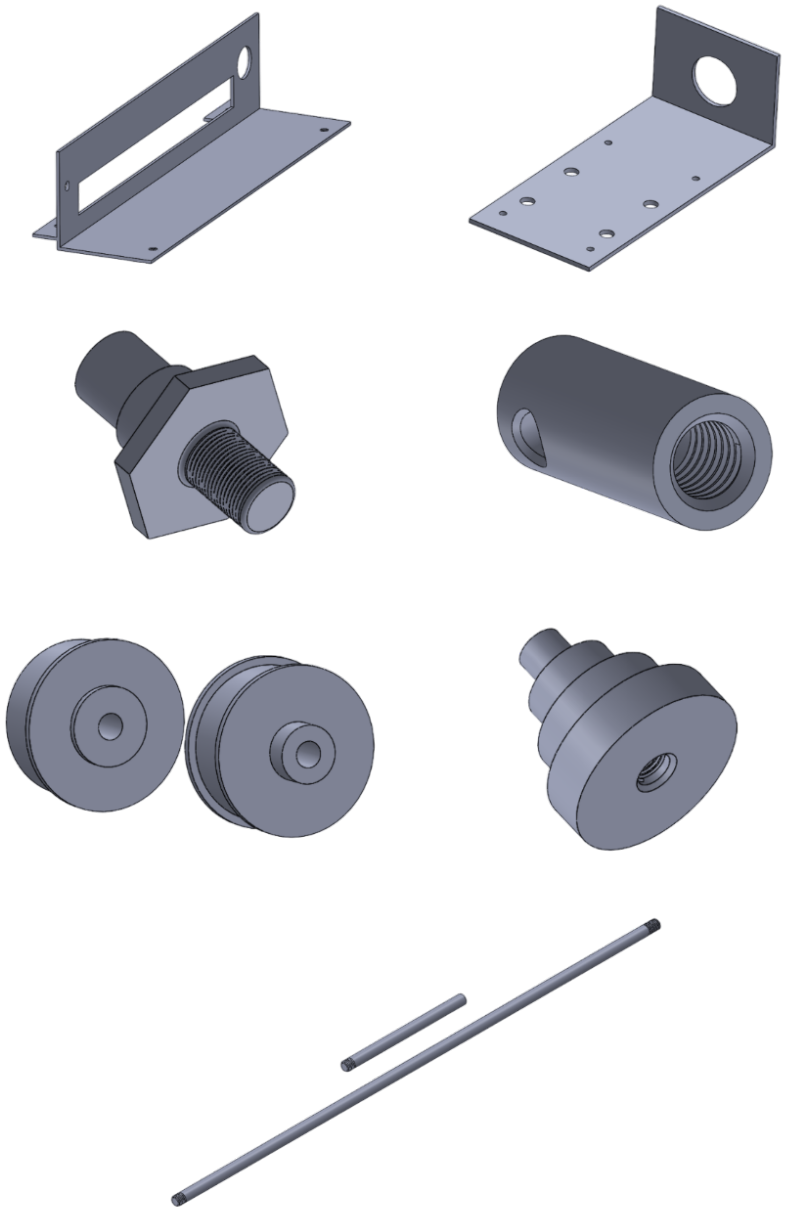
The system was designed to be a vital bridge between academic research and industrial demands. It provided a solid foundation for developing and validating cutting-edge technologies, including advanced control algorithms and artificial intelligence. Beyond promoting design autonomy, the project offered a comprehensive academic experience, covering the entire mechatronic system development cycle, from initial technical drafting to practical validation. This fosters an experimental environment that goes beyond theoretical boundaries, enabling the exploration of transformative, real-world applications.

An essential aspect of the project involved the preparation of mechanical drawings, with a focus on the design and fabrication of mechanical components. To ensure dimensional precision, structural integrity, and high-quality surface finishes, we diverse manufacturing techniques were employed, such as such as milling, turning, and welding. The components developed were designed for seamless integration with commercially available parts, enhancing flexibility and ensuring compatibility with existing industrial solutions.

All components were modeled using the software SolidWorks with an educational license (Dassault Systèmes, 2024), we meticulously modeled all components. A critical step before fabrication involved performing computational simulations to identify any structural weaknesses and thoroughly validate each part's performance. These simulations were foundational to guaranteeing both design reliability and overall manufacturing efficiency.

Comprehensive simulations were performed to analyze component behavior, identify critical stress points, and optimize geometries. This proactive validation reduced material waste and mitigated the risk of costly rework, increasing manufacturing accuracy. For complex systems like the inverted pendulum, this process is essential for achieving weight reduction and fostering innovation. It provides robust virtual verification long before any physical production begins.

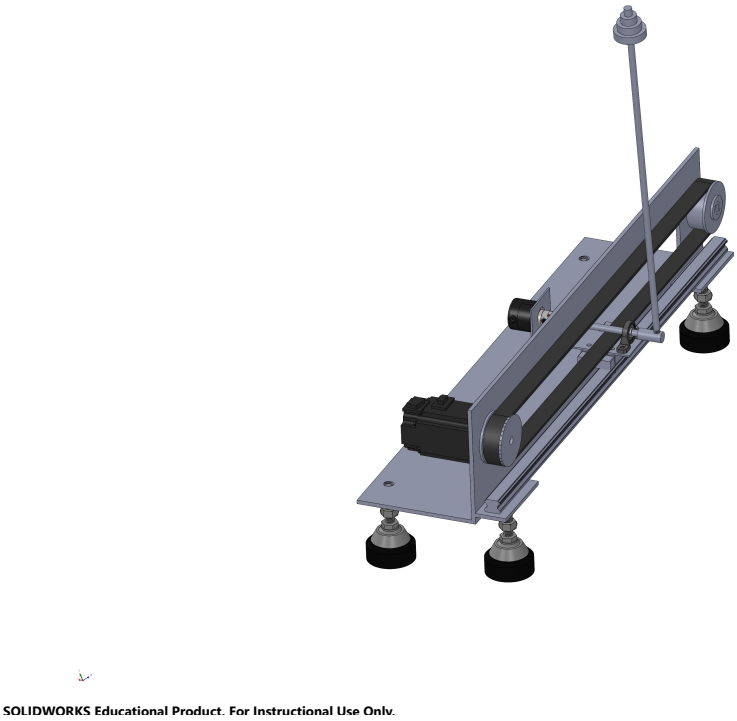
Figure 9 – Mechanical components modeled in SolidWorks, arranged individually to illustrate the complete set of parts designed for the system.



Source: Created by the author.

After the machining phase, additional adjustments were required to ensure the precise fitting and proper alignment of the components. These refinements were performed with the technical support of the machining team, whose experience contributed significantly to the feasibility of the final assembly. The necessary adaptations to the design were based on practical recommendations, taking into account the constraints of the manufacturing process. Following these adjustments, several practical tests were conducted to validate the mechanical assembly and confirm that the system met the technical requirements defined for the project.

Figure 10 – Final mechanical assembly of the system modeled in SolidWorks, illustrating the complete integration of all components as designed.



Source: Created by the author.

The adoption of a system entirely designed and manufactured according to industrial standards not only enhances operational safety but also provides a high level of reliability, ensuring consistent and reproducible performance under real-world conditions.

3.2 ELECTRICAL ARCHITECTURE

A Delta AS218PX-A Programmable Logic Controller (PLC), programmed using ISP-Soft (Delta Electronics, 2024), was implemented to control the pendulum system. PLCs are widely adopted in industrial environments due to their robustness and reliability. Designed to operate with a high level of safety, these controllers offer resistance to electromagnetic interference, temperature variations, and humidity, key characteristics for critical applications (Delta Electronics, 2018). Furthermore, they include integrated fault diagnostic functions, which facilitate predictive maintenance and contribute to the continuous availability and reliability of the system.

In contrast to generic controllers and data acquisition boards (e.g., Arduino-based systems), PLCs stand out by complying with industrial standards, offering certified safety

(e.g., SIL, ATEX), adherence to IEC 61131-3 (International Electrotechnical Commission, 2013), support for industrial communication protocols (e.g., PROFIBUS, CAN, Modbus TCP/IP, EtherCAT), robust construction for harsh environments, and deterministic real-time performance, essential for continuous, fail-safe operation.

The motor selected for this project was a 400W Delta servo motor (model ECMA-C10604RS), paired with a matching 400W servo drive (model ASD-A2-0421-M). Both components were configured and tuned using ASDA-Soft (Delta Electronics, 2024), the manufacturer's official and freely available software.

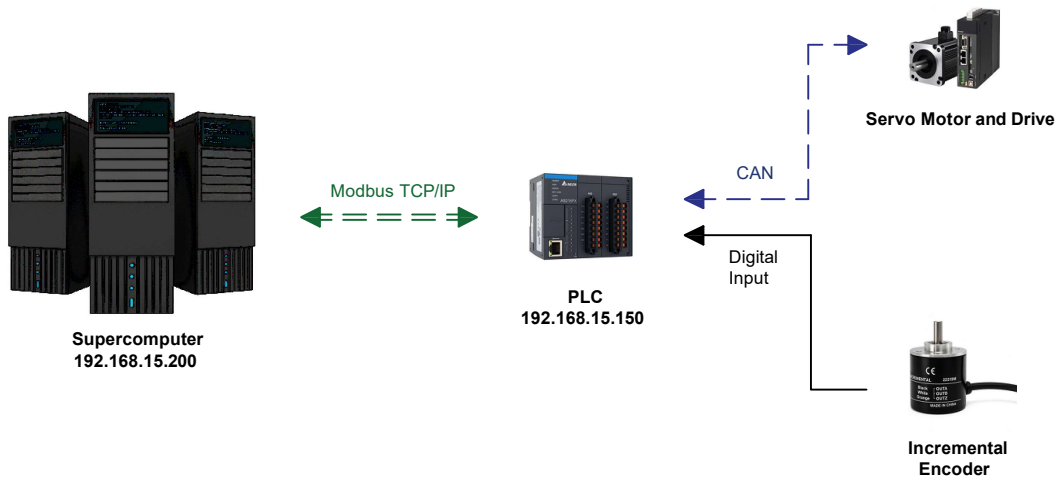
The communication between the PLC and the servo drive is carried out using the Controller Area Network (CAN) communication protocol, with fast response times and immunity to noise, overcoming the classic pulse train (PING *et al.*, 2016). This setup offers advanced features such as auto-tuning, high pulse-per-revolution resolution, no resonance, and self-correction mechanisms during operation.

Servo motors outperform DC motors with encoders by delivering superior torque control, higher precision, and more robust feedback. Compared to stepper motors with encoders, servos eliminate step loss, mechanical resonance, and high-speed torque drop-ensuring reliable, stable performance in dynamic industrial applications. With closed-loop control, servos maintain accuracy under load variations, making them ideal for pendulum systems where precision and responsiveness are critical.

The cart position is measured using the encoder integrated into the servo motor itself, allowing for smooth and stable motion in combination with the servo drive through fine torque control. Sensors form a critical component of the pendulum system architecture, with an Omron incremental rotary encoder (1024 PPR) providing high-resolution pendulum angle feedback. This precision enables real-time closed-loop control, ensuring accurate states estimation and stable system performance. The encoder's robust design and fine granularity make it ideal for demanding control applications where measurement accuracy directly impacts response quality.

Such architecture ensures synchronization, precision, and responsiveness, critical aspects to maintain dynamic balance and enable real-time control performance. A tight integration of electrical interfaces (motor drivers, encoders, power supply) and mechanical interfaces (cart, servo motor, pendulum) ensures stable, high-performance operation. This integration meets academic and industrial standards for real-time control and system responsiveness, as illustrated in the system topology in Figure 11.

Figure 11 – Network topology and system integration, including PLC, servo motor, encoders, sensors, and external processing system.



Source: Created by the author

The key specifications and manufacturers of the electrical components used for signal acquisition and control are summarized in Table 2.

Table 2 – Main electrical components used in the construction of the inverted pendulum system.

Quantity	Description	Manufacturer
1 pc	PLC AS218PX-A	Delta Electronics
1 pc	Servomotor 400W ECMA-C10604RS	Delta Electronics
1 pc	Servodrive 400W ASD-A2-0421-M	Delta Electronics
1 pc	Power Supply DRL-24V120W1EN	Delta Electronics
1 pc	Encoder 1024PR, CWZ5B-12-24V-PNP	Omron

Source: Created by the author.

The system topology prioritizes reliable communication, robust protocols and clear separation of control, actuation, and sensing-enhancing scalability and modular maintenance. Encoder-based position data is processed by the PLC, while an external supercomputing station handles advanced calculations for continuous adaptive control. This embedded external computing approach enables not only real-time adjustments, but also predictive optimization and sophisticated algorithms.

By combining industrial-grade robustness with high-performance computing, this framework advances cyber-physical systems for real-world control applications and cutting-edge research environments.

3.3 ASSEMBLY AND INTEGRATION

This section provides a detailed account of the inverted pendulum's operation and assembly. The system is built upon a precisely engineered mechanical architecture, crucial for achieving dynamic stability and high-precision control. Central to its design is a movable carriage, accurately mounted on a linear guide, which establishes the basis for the pendulum's controlled movement. The rigid pendulum is securely affixed to this carriage, with its shaft interconnected to an incremental rotary encoder. This encoder provides continuous angular displacement measurement and furnishes real-time feedback on the pendulum's angular position to the PLC.

The actuation of the carriage is achieved through a toothed belt, which is driven by a gear directly coupled to the shaft of a Delta servomotor. This servomotor plays a critical role in managing the system's dynamics, enabling precise positional adjustments of the carriage necessary for stabilizing the pendulum in its upright configuration. The linear guide is securely mounted on a solid steel base, with the servomotor positioned at one end to optimize mechanical stability and torque transmission.

Upon receiving input data, the PLC forwards them to the supercomputer for advanced computation and analysis. The supercomputer then returns the processed results to the PLC, which, in turn, transmits the necessary corrective control actions to the servo drive. This close integration of mechanical and electrical subsystems, supported by a robust structural foundation, guarantees consistent and dependable operational performance.

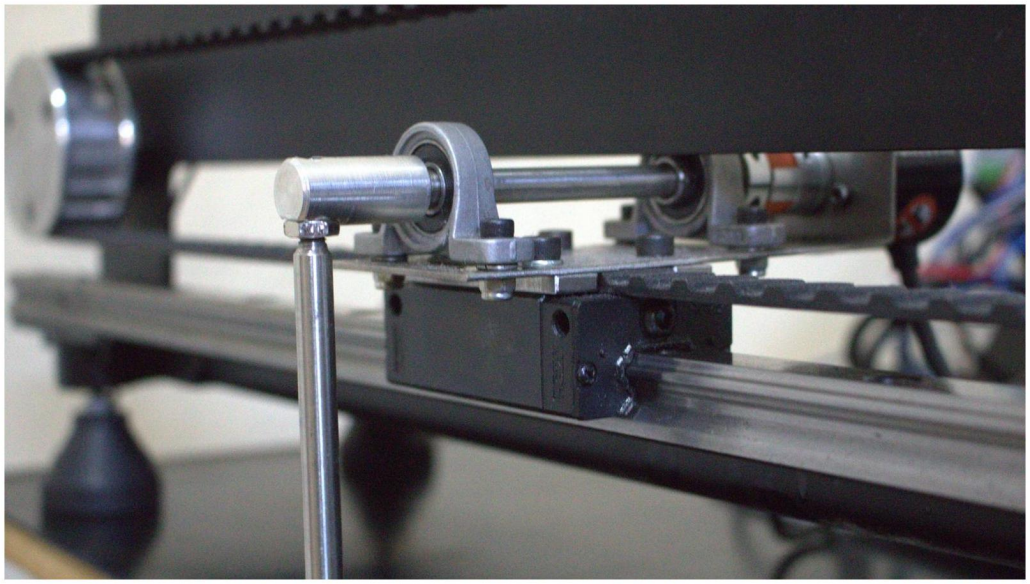
The overarching objective of the system is to maintain the pendulum in its inherently unstable vertical equilibrium, while simultaneously and actively rejecting external disturbances, such as environmental vibrations or perturbations in carriage motion. This architectural design thus facilitates robust disturbance rejection and high-fidelity dynamic control, even when operating under challenging conditions.

Stabilization is achieved through the implementation of advanced control algorithms, which are capable of rapidly adjusting the carriage's position in direct response to variations in the pendulum's angular orientation.

The main structure is fabricated from a 3/16"thick 1020 carbon steel sheet, precision-bent into an 'L' profile to provide structural integrity and mechanical support for the entire system. This material was selected for its high mechanical strength, dimensional stability, and

durability under dynamic loading conditions. A 20 mm × 700 mm linear guide was welded to the front face of the base, allowing for accurate and stable carriage movement along the axis, as shown in Figure 12.

Figure 12 – Front view of the linear base, highlighting the guide rail for precise and stable horizontal carriage motion.



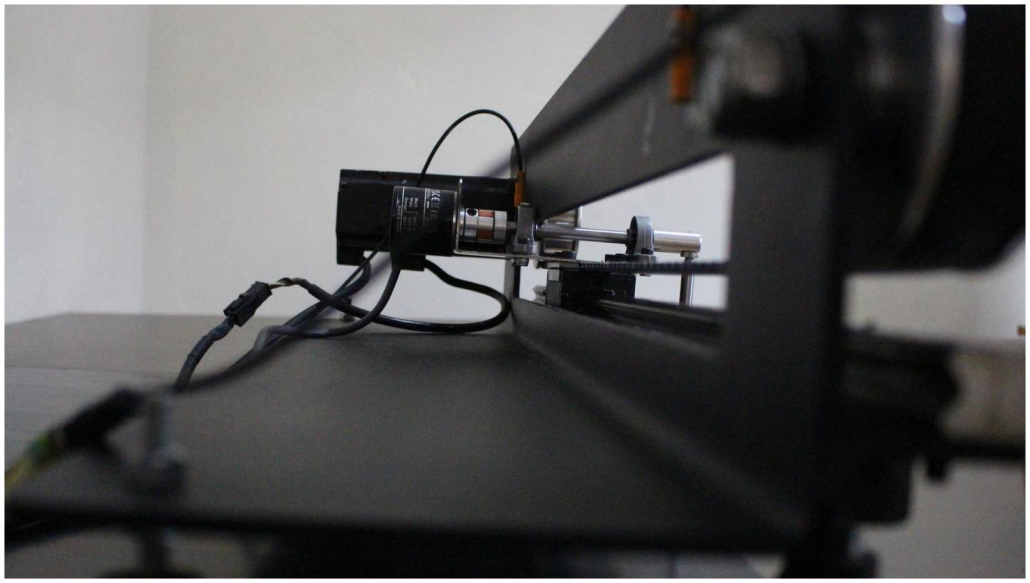
Source: Photograph taken by the author.

The carriage, responsible for driving the pendulum's base, is mounted on skids that provide low-friction movement along the linear rail. A 1/8"stainless steel plate, also bent into an 'L' shape, is bolted to the skids and serves as the mounting interface for the encoder and pendulum linkage.

For safety and system protection, two magnetic sensors were installed at both ends of the linear guide. These sensors act as physical end stops, ensuring that the carriage does not exceed the mechanical boundaries of the rail, even if the software limits are inadvertently surpassed. This redundant safety measure helps to prevent damage to both the carriage and the supporting structure in the event of unexpected software or hardware faults.

Two 8 mm KP08 bearing units support the encoder and pendulum shafts, maintaining precise alignment and smooth rotation. A flexible shaft coupler (5x8 mm) connects the encoder to the pendulum horizontal shaft, minimizing the effects of misalignment and enabling accurate torque transmission, as shown in Figure 13.

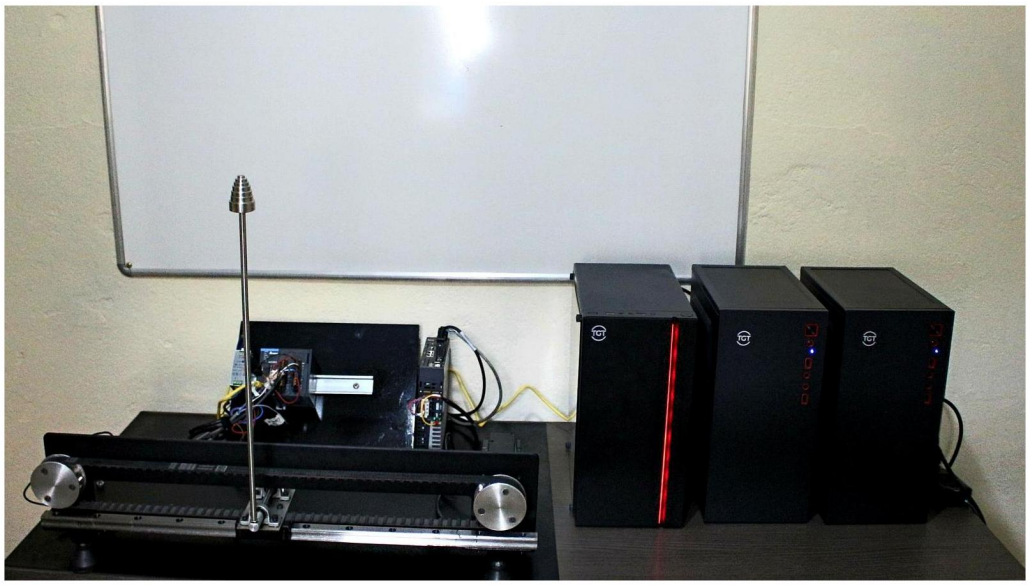
Figure 13 – Carriage assembly, including the encoder and pendulum mounting support.



Source: Photograph taken by the author.

The pendulum arm is constructed from stainless steel, selected for its stiffness, corrosion resistance, and low weight. A strategically placed mass near the tip balances sensitivity and inertia, allowing for quick response while keeping the rotational inertia low, a key factor for achieving precise control. Figure 14 illustrates the complete physical system developed for control system validation

Figure 14 – Complete experimental setup developed for validation of the control strategy.



Source: Photograph taken by the author.

To minimize friction and ensure linear travel accuracy, the guide rails were meticulously aligned during assembly. Rubber damping elements were installed underneath the base to absorb ambient vibrations and enhance structural stability. Design decisions such as the use of lightweight materials for moving components and robust alloys for load-bearing structures contribute to reduced response times, increased energy efficiency, and long-term durability.

To provide full transparency regarding the system's fabrication, Table 3 presents all materials purchased and used in the assembly and construction of the inverted pendulum, excluding any components previously discussed in earlier sections.

Table 3 – Materials and components used in the construction of the inverted pendulum system.

Quantity	Description	Manufacturer
1 pc	Structural Base 1040 steel sheet bent in L-shape	Custom-made
1 pc	Pendulum Shaft Machined stainless steel	Custom-made
1 pc	Pendulum Weight Solid steel	Custom-made
1 pc	Motor Gear Milled aluminum	Custom-made
1 pc	Cart Shaft 1020 steel	Custom-made
1 pc	Gear Shaft 1020 steel	Custom-made
1 pc	Gear Milled aluminum	Custom-made
4 pcs	Extra-Mini 1/4"Rubber Foot 70 KG	Vibramatt
1 pc	Synchronous Belt 570H	Powergrip
1 pc	Ball Bearing 6203-Z/C3	SKF
1 pc	Linear Guide 20mm x 700mm	Prado
1 pc	Linear Guide Carriage H20	Prado
2 pcs	Bearing Block KP08 8mm	Generic
1 pc	Flexible Shaft Coupler d20l25 5x8mm	Generic
1 pc	Cable Carrier/Chain Transmission Chain 10x11mm	Generic
3 pcs	Allen Screw, Cylindrical Head, M3 x 8 Alloy Steel	Generic
4 pcs	Allen Screw, Cylindrical Head, M4 x 12 Alloy Steel	Generic
4 pcs	Allen Screw, Cylindrical Head, M5 x 10 Stainless Steel	Generic
4 pcs	Allen Screw, Cylindrical Head, M5 x 20 Alloy Steel	Generic
4 pcs	Allen Screw, Cylindrical Head, M6 x 20 Alloy Steel	Generic
4 pcs	Set Screw, M6 x 8 Alloy Steel	Generic
1 pc	Internal Retaining Ring, 40mm Spring Steel	Generic
1 pc	External Retaining Ring, 17mm Spring Steel	Generic
1 pc	Lock Nut, Hexagonal, M14 Zinc-coated	Generic
4 pcs	Hex Nut, M4 Zinc-coated	Generic

Source: Created by the author.

The result is a highly functional, scalable platform that not only addresses academic research needs but also serves as a practical benchmark for the development of next-generation control systems in industrial applications.

3.4 CONTROL INTERFACE AND SOFTWARE ENVIRONMENT

The control system is encoded using Python 3.13 (Python Software Foundation, 2024) for its development, chosen for its inherent simplicity, readability, and a rich ecosystem of libraries that are critical for data analysis, simulation, and real-time control. The system operates on Ubuntu 22.04.1 LTS (Canonical Ltd., 2022), a selection driven by its long-term support, stability, and extensive compatibility with industrial communication protocols and scientific computing tools.

A 1000 Mbps Ethernet network allows the Modbus/TCP communication between the computing server and the PLC, implemented via the `pyModbusTCP` library (FAUTH, 2024). As mentioned earlier, the PLC and servo drive communicate using the CAN protocol, specifically chosen for its optimization for real-time, low-latency, and deterministic control, rendering it ideal for high-precision motion systems.

The software architecture prioritizes flexibility, scalability, and computational efficiency, thereby supporting advanced control methodologies such as RL and MPC. Core Python libraries include NumPy (OLIPHANT; CONTRIBUTORS, 2024) for numerical computation, Pandas (TEAM, 2024) for structured data manipulation, and Matplotlib (HUNTER; CONTRIBUTORS, 2024) for data visualization. Redis 7.4 (Redis Ltd., 2024) is employed for temporary data storage and real-time inter-process communication, acting as a high-speed buffer between control processes. After processing, relevant data stored in Redis is subsequently transferred to a MySQL database (Oracle Corporation, 2024) managed via the XAMPP platform (Apache Friends, 2024), for long-term archival, analysis, and reporting. To ensure portability across different Database Management Systems (DBMS), the implementation strictly adheres to ANSI SQL standards, minimizing vendor-specific dependencies and facilitating seamless migration if required.

A foundational principle of this dissertation is the exclusive reliance on open-source software or free software, which ensures accessibility, reproducibility, and cost-effectiveness across both academic and industrial contexts.

3.5 SAFETY AND SECURITY CONSIDERATIONS

Operational safety, particularly crucial during real-time control of inherently unstable systems like the inverted pendulum, is ensured through strictly integrated software safeguards. These mechanisms encompass actuator disabling upon control signal loss or latency overflow,

alongside real-time monitoring scripts that continuously evaluate sampling time thresholds. Should these thresholds be exceeded, a predefined fallback routine is triggered to halt actuator operation and transition the system to a safe state.

Regarding cybersecurity, communication between the computing cluster and the PLC is strictly confined to the local network, thereby exposing no external interfaces and minimizing the attack surface. Furthermore, all communication protocols, such as Modbus/TCP, are encapsulated within isolated network namespaces and fortified by firewall rules. While Modbus/TCP lacks inherent security features, its deployment within an internal, physically isolated network effectively mitigates common vulnerabilities.

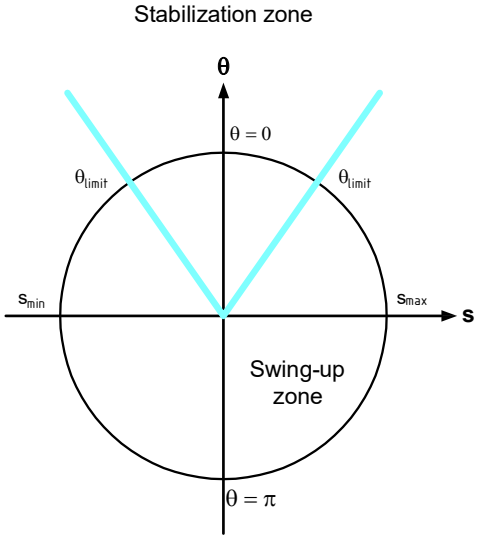
Collectively, these integrated safety and security strategies safeguard the physical experimental platform, uphold the integrity of the control logic, and guarantee the reliability of the real-time system.

4 SYSTEM MODELING AND DYNAMICS

4.1 LINEARIZATION AND DISCRETIZATION

Following (TIGA *et al.*, 2019), the proposed control strategy involves switching between backstepping control for the swing-up phase and linear state feedback control for the stabilization phase. The global control law F is divided into F_{swp} and F_{stb} for the swing-up and stabilization zones, respectively, as depicted in Figure 15.

Figure 15 – Swing-up and stabilization zones.



Source: Created by the author, based on (TIGA *et al.*, 2019).

In the stabilization zone, defined by $|x_3| \leq 0.2$ radians, the system can be linearized using small-angle approximations: $\sin(\theta(t)) \approx \theta(t)$, $\cos(\theta(t)) \approx 1$, and $\dot{\theta}(t) \approx 0$. This linearization allows the system to be expressed by the following linear state equation,

$$\frac{d}{dt}x(t) = A_c x(t) + B_c u(t), \tag{27}$$

where A_c represents the continuous-time state matrix in $\mathbb{R}^{4\times4}$, and B_c is the continuous-time input matrix in $\mathbb{R}^{4\times1}$. Rewriting the matrix form Equations (11), we obtain the following state and output equations (OGATA, 2010; TIGA *et al.*, 2019).

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-\mu_c(I_m+m_p\ell^2)}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} & \frac{-gm_p^2\ell^2}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} & \frac{\mu_p m_p\ell}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\mu_c m_p\ell}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} & \frac{gm_p\ell(m_c+m_p)}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} & \frac{-\mu_p(m_c+m_p)}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} \end{bmatrix},$$

$$B_c = \begin{bmatrix} 0 \\ \frac{I_m+m_p\ell^2}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} \\ 0 \\ \frac{-m_p\ell}{(m_c+m_p)(I_m+m_p\ell^2)-(m_p\ell)^2} \end{bmatrix}.$$

The discretization of state-space models involves transforming continuous differential equations into discrete difference equations, a crucial step for digital implementation. The Tustin method, also known as the bilinear transformation, was chosen due to its well-established ability to provide a robust and accurate approximation. It achieves this by mapping the continuous-time stable region (left-half of the s_L-plane) to the discrete-time stable region (interior of the unit circle in the z-plane).

This method replaces the continuous-time derivative with a discrete approximation, defined as,

$$s_L = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}. \tag{28}$$

Here, T is the sampling period, s_L is the Laplace operator (representing the continuous domain), and z is the Z-transform variable (representing the discrete domain). Applying this transformation to the state-space model yields the discrete equivalent,

$$x_{k+1} = A_d x_k + B_d u_k, \tag{29}$$

the discrete matrices A_d and B_d are calculated using the following formulas,

$$A_d = \left(I - \frac{T}{2}A_c\right)^{-1}\left(I + \frac{T}{2}A_c\right), B_d = \left(I - \frac{T}{2}A_c\right)^{-1} TB_c. \tag{30}$$

This method effectively converts the continuous system representation into a form suitable for numerical analysis and real-time application.

4.2 SYSTEM IDENTIFICATION AND PARAMETER ESTIMATION

To support the development of a model-based control strategy, a preliminary system identification and parameter estimation procedure was performed on the inverted pendulum platform. This stage was essential for constructing a baseline dynamic model by approximating the physical characteristics of the system, thereby enabling early stage simulations and guiding the controller design. Although the initial model was subsequently refined through empirical validation and online adjustments, the identification process provided a robust foundation for reliable control implementation.

As detailed in Chapter 3, the system constants were derived from a virtual model and verified against the physical system. The identified parameters include: cart mass $m_c = 0.360$kg, pendulum mass $m_p = 0.179$kg, pendulum length $\ell = 0.463$m, , and gravitational acceleration $g = 9.810\text{m/s}^2$.

The system was modeled in discrete time, with state updates occurring at a fixed sampling interval of $T = 50\ ms$. This interval was selected based on a series of experiments aimed at evaluating the performance of the data acquisition and control subsystems. The selected sampling rate represents a balanced compromise between temporal resolution and computational demand, ensuring that sensing, estimation, and control tasks are executed reliably within each control cycle.

To establish the dynamic model of the pendulum, the moment of inertia I_m was first calculated. For a homogeneous slender rod pivoted at one end, an approximation equivalent to treating the pendulum as a point mass at its center of gravity the moment of inertia is given by (MERIAM; KRAIGE, 2007; HALLIDAY *et al.*, 2011)

$$I_m = \frac{1}{3} m_p \ell^2, \tag{31}$$

where $m_p = 0.179$,kg is the pendulum mass and $\ell = 0.463$,m is its length. Substituting these values yields

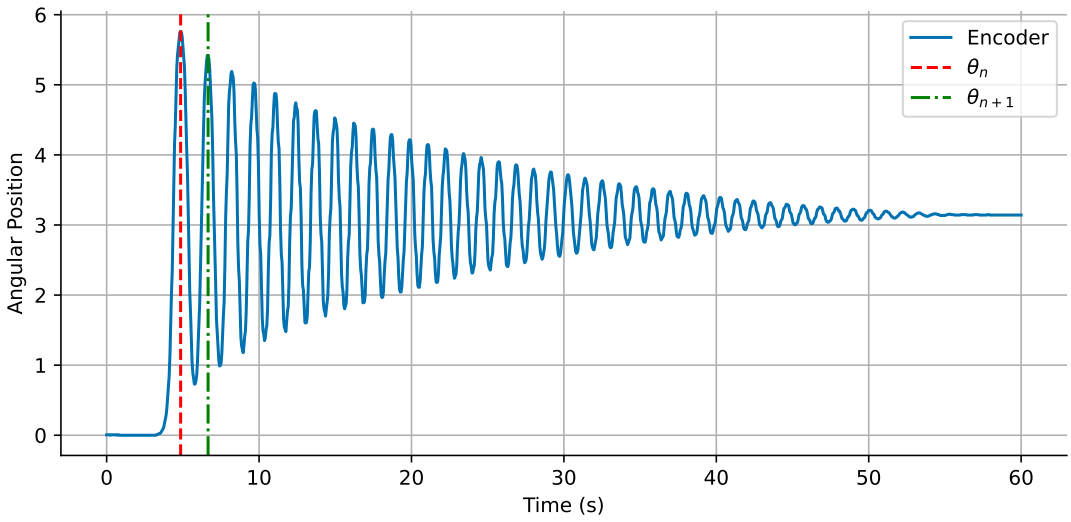
$$I_m = \frac{1}{3}(0.179\text{kg})(0.463\text{m})^2 \approx 0.013\text{kg} \cdot \text{m}^2.$$

The viscous friction coefficient was estimated through the logarithmic decrement method, commonly employed in vibration analysis (RAO, 2011; INMAN, 2014). The damping factor λ, defined as the amplitude ratio between two successive oscillation cycles,

$$\lambda = \frac{\theta_{n+1}}{\theta_n}, \tag{32}$$

was determined experimentally. Figure 16 illustrates the decay in oscillatory amplitude employed for this analysis. Upon release, the pendulum swings to the side opposite the initial displacement, the peak angles are equal in magnitude but opposite in sign.

Figure 16 – Decay of the pendulum's oscillatory amplitude between two consecutive cycles.



Source: Author's data.

From the measured peak values, $\theta_n = 5.762$ and $\theta_{n+1} = 5.418$, the damping factor is given by

$$\lambda = \frac{5.418}{5.762} \approx 0.940,$$

which indicates relatively mild energy dissipation in the pendulum's motion. The damping rate γ was obtained from the damping factor λ and the oscillation period T_s, defined as the time between two consecutive peaks. For the present case, $T_s = 1.8\text{s}$, leading to

$$\gamma = -\frac{\ln(0.940)}{1.8\text{s}} \approx 0.034\text{s}^{-1}.$$

With γ known, the viscous friction coefficient of the pendulum was computed as

$$\mu_p = 2I_m\gamma = 2(0.013\text{kg} \cdot \text{m}^2)(0.034\text{s}^{-1}) \approx 0.001\text{N} \cdot \text{m} \cdot \text{s},$$

thus quantifying the rotational friction opposing the pendulum's motion.

Similarly, viscous friction in the cart was modeled as a force proportional to its velocity, with the coefficient μ_c expressed as (MERIAM; KRAIGE, 2007)

$$\mu_c = \frac{m_c a}{v}, \tag{33}$$

where $m_c = 0.360\text{kg}$ is the cart mass, a is its linear acceleration, and v the corresponding velocity. Since the servo drive provides only torque and angular position measurements, without direct

access to velocity or force data, the Kalman filter was implemented to estimate the cart's velocity. To ensure that viscous effects could be isolated, the guide rails were lubricated to minimize the influence of dry friction. A series of experiments were performed at different velocity settings, yielding mean values of $\bar{a} = 0.009\text{m/s}^2$ and $\bar{v} = 0.261\text{m/s}$. These values were then used to compute the viscous friction coefficient. Substituting the averaged experimental values yields,

$$\mu_c = \frac{(0.360\text{kg})(0.009\text{m/s}^2)}{0.261\text{m/s}} \approx 0.011\text{N} \cdot \text{m/s}.$$

This empirically derived coefficient characterizes the viscous friction opposing the cart's motion. Its accuracy relies on both the quality of the velocity estimation provided by the Kalman filter and the controlled experimental setup. For clarity and completeness, Table 4 summarizes all identified system parameters.

Table 4 – Physical parameters of the inverted pendulum.

Symbol	Description	Value	Unit
m_c	Mass of the cart	0.360	kg
m_p	Mass of the pendulum rod	0.179	kg
ℓ	Length of the pendulum rod	0.463	m
g	Acceleration due to gravity	9.810	m/s^2
μ_c	Cart damping coefficient	0.011	$\text{N} \cdot \text{s/m}$
μ_p	Pendulum damping coefficient	0.001	$\text{N} \cdot \text{m} \cdot \text{s}$
I_m	Rotary inertia of the pendulum rod	0.013	$\text{kg} \cdot \text{m}^2$

Source: Created by the author.

4.3 MODEL VALIDATION

Following the identification of the system parameters, a fine-tuning stage was performed using the least squares method to address residual discrepancies between simulation and experimental results. Although the initial model provided a good approximation for the system behavior, discrepancies emerged when comparing the simulated responses to experimental measurements, particularly in long-duration tests and during transitions.

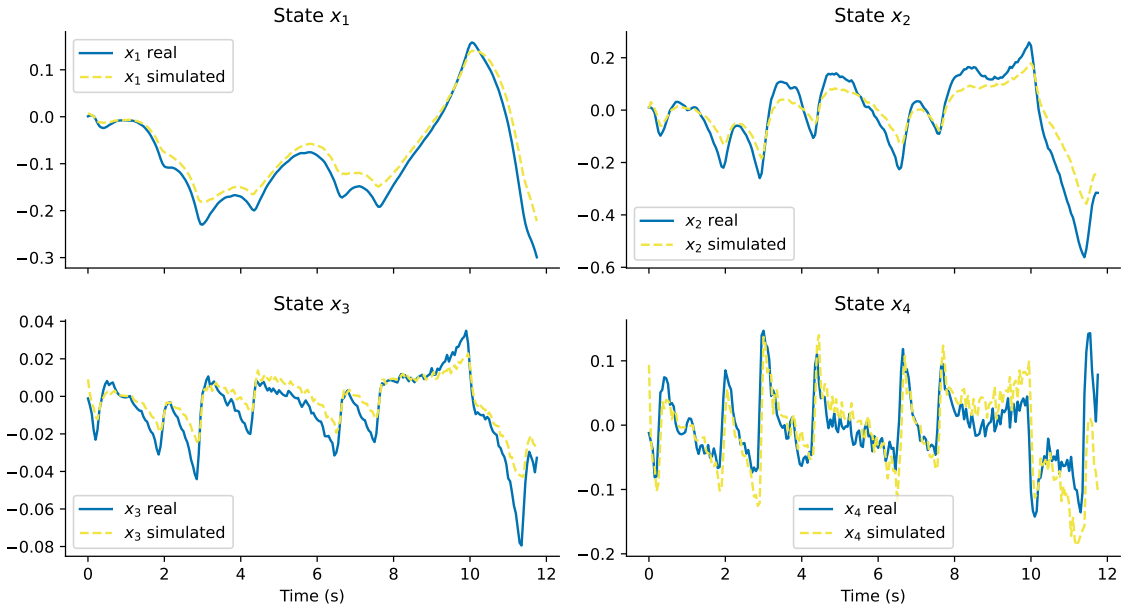
The least squares optimization allowed for systematic refinement of model coefficients based on experimental input-output data, thereby increasing the accuracy of state predictions and improving alignment with the physical response. The resulting discrete-time state-space model is described by the matrices,

$$A_d = \begin{bmatrix} 9.9881 \times 10^{-1} & 2.0725 \times 10^{-2} & 0 & 0 \\ 0 & 9.0577 \times 10^{-1} & 1.8155 \times 10^{-4} & 2.5754 \times 10^{-1} \\ 0 & 4.2402 \times 10^{-2} & 9.9713 \times 10^{-4} & 4.3144 \times 10^{-2} \\ 0 & -3.0689 \times 10^{-1} & 2.2730 \times 10^{-5} & 1.6153 \times 10^{-1} \end{bmatrix},$$

$$B_d = \begin{bmatrix} 5.8437 \times 10^{-5} \\ 1.1186 \times 10^{-5} \\ 8.6361 \times 10^{-5} \\ 9.3370 \times 10^{-4} \end{bmatrix}.$$

To validate the model, predefined input signals u_k were applied to both the real system and the simulation model, starting from the same initial state. In the physical setup, these signals were transmitted via a PLC, while in the simulation, the same inputs propagated the model dynamics governed by discrete-time matrices A_d and B_d. The resulting state trajectories were compared to evaluate prediction accuracy.

Figure 17 – Comparison between experimental and simulated trajectories starting from identical initial states and subjected to the same control input u.



Source: Author's data.

As shown in Figure 17, the simulated responses align with the closed-loop experimental measurements, particularly near the upright equilibrium. This demonstrates the model's reliability in capturing both transient and steady-state behavior, which is critical for subsequent controller design.

Small mismatches in amplitude and phase were observed, especially under higher excitation or abrupt changes. These were attributed to unmodeled nonlinearities, time delays, and sensor noise in the real system. Nonetheless, the integration of a Kalman filter for state estimation and the online adaptation of certain parameters mitigated these effects, enhancing the robustness and applicability of the model for real-time control.

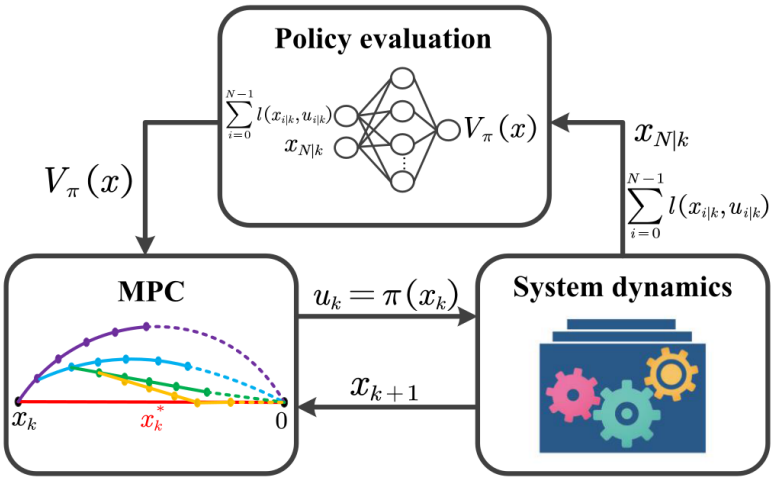
The successful validation confirms that the refined model, calibrated using the least squares approach, is sufficiently accurate for use in control design and system analysis. It provides a reliable foundation for predictive simulations and for evaluating control strategies under realistic operational conditions.

5 REINFORCEMENT LEARNING-BASED PREDICTIVE CONTROL

5.1 OVERVIEW OF THE RLMPC FRAMEWORK

The proposed Reinforcement Learning-based Model Predictive Control (RLMPC) framework integrates RL and MPC to optimize control strategies. By iteratively refining the value function and control policy through a Policy Iteration framework, RLMPC ensures stability, constraint satisfaction, and adaptability in dynamic environments. Figure 18 illustrates the RLMPC pipeline, which follows a cyclic process composed of policy evaluation, value function learning, and controller updates.

Figure 18 – RLMPC pipeline.



Source: Image taken from (LIN *et al.*, 2023).

A key innovation lies in incorporating a heuristic term, derived from the current policy, as the terminal cost within the MPC framework. This term captures long-term rewards, helping the controller balance immediate costs with future benefits. The control input from this policy serves as the current strategy, while system interaction costs are collected as training data to iteratively refine the value function, ensuring alignment with the Bellman equation.

To approximate the value function efficiently in high-dimensional or continuous state-action spaces, RLMPC employs Value Function Approximation (VFA) techniques, particularly DNNs. This approach mitigates challenges such as the curse of dimensionality and computational complexity.

The RLMPC process follows an iterative, closed-loop structure consisting of three main stages:

- **Policy Evaluation**: The current policy is assessed using data collected from system interactions.
- **Value Function Learning**: The value function is updated through DNN-based approximation techniques, enabling generalization across the state space.
- **Controller Update**: The refined value function is incorporated into the MPC cost formulation, guiding the improvement of the control policy.

This cycle ensures continuous refinement of the control strategy, enabling adaptability and optimal performance in dynamic environments. By incorporating RL-based terminal costs, RLMPC overcomes the limitations of traditional MPC, which often results in locally optimal policies. This hybrid approach enhances control performance across short and long horizons while ensuring stability and constraint satisfaction.

Future research directions include real-time adaptation of the value function, robust RL techniques for handling uncertainties, and transfer learning to facilitate deployment across diverse environments with minimal retraining.

5.2 VALUE FUNCTION APPROXIMATION WITH NEURAL NETWORKS

In the Linear Quadratic Regulator (LQR) framework, the value function $V_\pi(x_k)$, defined in (23), is represented using a complete basis set of quadratic functions in the components of x_k.

For nonlinear systems, when the value function is sufficiently smooth, the Weierstrass Approximation Theorem ensures that any continuous function on a compact domain can be uniformly approximated by algebraic polynomials. As established in (PINKUS, 2000), the density of polynomials in the space of continuous functions provides a theoretical foundation for value function approximation in reinforcement learning and optimal control. This justifies the use of parametric approximators, such as orthogonal polynomials or neural networks, especially in high-dimensional problems where closed-form solutions are intractable.

Crucially, this theorem guarantees the existence of a dense basis set $\{\Phi_i(x_k)\}$ such that

$$V_\pi(x_k) = \sum_{i=1}^{\infty} W_i \Phi_i(x_k) = \sum_{i=1}^{p} W_i \Phi_i(x_k) + \sum_{i=p+1}^{\infty} W_i \Phi_i(x_k), \tag{34}$$

where $\Phi(x_k) = [\Phi_1(x_k),\Phi_2(x_k),\ldots,\Phi_p(x_k)]^T$ mapping $\mathbb{R}^n \to \mathbb{R}^p$ is the basis vector, $W = [W_1,W_2,\ldots,W_p]^T \in \mathbb{R}^p$ is the weight vector, and the approximation error is given by

$$e(x_k) = V_\pi(x_k) - \hat{V}_\pi(x_k,W) = \sum_{i=p+1}^{\infty} W_i\Phi_i(x_k). \tag{35}$$

This error converges uniformly to zero as $p \to \infty$. The formulation thus separates the true value function into an approximate component $\hat{V}_\pi(x_k,W) = W^T\Phi(x_k)$ and a residual error $e(x_k)$.

While the classical Weierstrass theorem traditionally employs polynomial bases, the neural network community has demonstrated that other basis functions, such as sigmoids, hyperbolic tangents, and Gaussian radial basis functions, can also serve this purpose (LESHNO *et al.*, 1993). These functions effectively approximate continuous functions, with the approximation error $e(x_k)$ bounded on compact sets (LEWIS; VRABIE, 2009). In this context, p represents the number of hidden layer neurons in the neural network.

5.3 TRAINING THE VALUE FUNCTION

Given a sample set $S_k = \{x_{k1}, x_{k2}, \ldots, x_{kq}\} \subset \mathbb{X}$ with q sampled states ($q \in \mathbb{N}_{>0}$), each element serves as an initial state. Propagating one step under policy π, the incurred costs are recorded as training data. For each $x_{k_j} \in S_k$, the optimal cost $J(x_{k_j},\mathbf{u}^*_{k_j})$ is computed using the mathematical model (LIN *et al.*, 2023), without applying $\mathbf{u}^*_{k_j}$ to the real system, ensuring training data safety. The subscript k_j is used to distinguish these states from the actual system state.

Using these costs $J(x_{k_j},\mathbf{u}^*_{k_j})$, $j \in \mathbb{N}_{>0}$, the network can be trained similarly to Temporal Difference (TD) learning. The goal is to approximate the TD target with $V_\pi(x_k)$. Here, we directly use $\hat{V}_\pi(x_{k_j},W)$ to approximate the targets $J(x_{k_j},\mathbf{u}^*_{k_j})$

$$\hat{V}_\pi(x_{k_j},W) \leftarrow \hat{V}_\pi(x_{k_j},W) + \alpha[J(x_{k_j},\mathbf{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j},W)], \tag{36}$$

for $\forall j \in \mathbb{N}_{[1,q]}$.The minimization is achieved by adjusting the weight vector W using a small step size $\alpha \in (0,1)$. The objective is to minimize

$$E(x_{k_j}) = \tfrac{1}{2}e(x_{k_j})^2, \quad \text{where} \quad E(x_{k_j}) = J(x_{k_j},\mathbf{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j},W). \tag{37}$$

Using stochastic gradient descent algorithm (LIN *et al.*, 2023), the weights W are updated by following the negative gradient of the error function with respect to W

$$W \leftarrow W - \frac{1}{2}\alpha\nabla_W[J(x_{k_j},\mathbf{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j},W)]^2, \tag{38}$$

which simplifies to

$$W \leftarrow W + \alpha[J(x_{k_j},\mathbf{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j},W)]\nabla_W \hat{V}_\pi(x_{k_j},W), \tag{39}$$

thus, the weight update becomes

$$W = W + \alpha[J(x_{k_j},\mathbf{u}^*_{k_j}) - W^T\Phi(x_{k_j})]\Phi(x_{k_j}). \tag{40}$$

where ∇_W denotes the partial derivatives of $E(x_{k_j})$ with respect to the components of W. This step adjusts W to reduce the squared error.

Once W converges, the approximation $\hat{V}_\pi(x_k,W)$ approaches $V_\pi(x_k)$ for all $x_k \in \mathbb{X}$, provided sufficient training data and an appropriate basis vector are used. This approach requires the storage of only a p-dimensional vector W, enhancing computational efficiency and mitigating the curse of dimensionality.

5.4 MPC FORMULATION WITH LEARNED TERMINAL COST

While classical Model Predictive Control (MPC) optimizes a finite-horizon cost function (typically over N steps), this approach is fundamentally limited to generating locally optimal policies for problem (20). To incorporate global system behavior and enhance closed-loop performance, we augment the MPC formulation with a terminal cost derived from reinforcement learning.

Following the stability analysis in (BOCCIA *et al.*, 2014) and motivated by recent RLMPC approaches, we introduce the learned value function $V_\pi(x_{N|k})$ as the terminal cost. Such addition results in the modified optimization problem of finding the optimal u_k solution of

$$\min_{\mathbf{u}_k} J(x_k,\mathbf{u}_k) = \sum_{i=0}^{N-1} \ell(x_{i|k},u_{i|k}) + V_\pi(x_{N|k}), \tag{41}$$

subject to

$$\begin{aligned}
x_{i+1|k} &= f(x_{i|k},u_{i|k}), \quad 0 \le i \le N-1,\\
u_{i|k} &\in \mathbb{U}, \quad 0 \le i \le N-1,\\
x_{i|k} &\in \mathbb{X}, \quad 0 \le i \le N-1,\\
x_{N|k} &\in \mathbb{X}_\Omega,
\end{aligned}$$

where $\mathbf{u}_k = \{u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}\}$ is the decision variable, $\ell(x_{i|k},u_{i|k})$ is the stage cost, for instance the LQR cost function presented in Section 2.4, and $V_\pi(x_{N|k})$ is the terminal cost obtained from the RL value function. We define the MPC control policy generated under a fixed

$V_\pi(\cdot)$ as fixed policy. At each time k, solving the optimization problem above yields a control sequence $\mathbf{u}_k^* = \{u_{0|k}^*, u_{1|k}^*, \ldots, u_{N-1|k}\}^*$, where only the first control action $u_{0|k}^*$ is applied to system (8).

The incorporation of the RL-based terminal cost $V_\pi(x_{N|k})$ allows MPC to account for long-term rewards, addressing the limitation of locally optimal policies. By leveraging the value function learned through RL, the MPC framework adapts to propagate global information into the optimization process. This hybrid approach ensures stability while improving performance across both short and long horizons.

The resulting control strategy bridges the strengths of model-based and model-free methods, offering enhanced scalability and flexibility for dynamic systems. Future works could explore adapting $V_\pi(\cdot)$ during runtime to refine the terminal cost based on real-time learning, enabling more robust and adaptive control under uncertainties.

6 PARALLEL COMPUTATION FOR REAL-TIME CONTROL

6.1 COMPUTATIONAL INFRASTRUCTURE

A supercomputer often operates as a cluster computing system, where numerous interconnected computers, or nodes, function collectively as a single, high-performance machine (DONGARRA *et al.*, 2020). These nodes execute tasks in parallel using high-speed local area networks (LANs) and distributed memory, enabling massive scalability and efficient processing of large-scale computations.

Unlike traditional monolithic supercomputers, cluster based systems often use off-the-shelf hardware (such as x86 CPUs and GPUs), making them cost-effective and flexible. Examples like Fugaku and Summit demonstrate how clustering enables extreme performance for scientific research, AI, and complex simulations (ALMOMANI *et al.*, 2019)

Each computer in the cluster is powered by a 2.7 GHz multicore processor, mounted on an LGA 2011 compatible motherboard. With 12 physical cores, these processors deliver exceptional parallel processing capabilities, making them well-suited for handling computationally intensive workloads. Additionally, each node is equipped with 16 GB of RAM, providing sufficient memory to support advanced computational algorithms, such as reinforcement learning and predictive control.

The computational infrastructure, whose components are enumerated in Table 5, is architected to deliver high performance, ample memory capacity, and efficient networking capabilities, thereby ensuring the cluster's aptitude for addressing complex computational tasks with both speed and reliability.

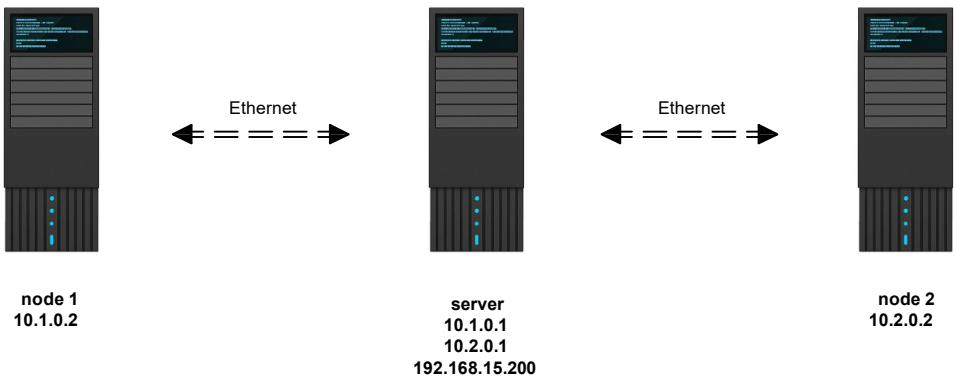
Table 5 – Materials and components used in the construction of the compute cluster.

Quantity	Description	Model/Material
3 pcs	Processor: E5-2697 v2, 12 cores, 2.7 GHz	Intel Xeon
3 pcs	Motherboard: X79 VG2, LGA 2011 socket	ZSUS
3 pcs	SSD: 128 GB, 2.5"SATA	Goldenfir
6 pcs	RAM: 8 GB DDR3	Hynix
3 pcs	Power Supply: 100-N1-0400-L1, 400W	EVGA
2 pcs	Network Adapter: RTL8125B, 2.5 Gigabit	Fenvi
1 pc	Network Adapter: IO-PCE8125E-2GLAN, 2.5 Gigabit	IO Crest

Source: Created by the author.

As mentioned earlier, communication between the service and the nodes is established via high-speed local area networks (LANs) configured in a star topology, as visually represented in Figure 19. This design critically enables seamless communication and rapid data transfer, which are indispensable for effective distributed computing paradigms. The primary mechanism for this inter-node communication is the Message Passing Interface (MPI), specifically implemented through OpenMPI (Open MPI Project, 2021), a widely recognized and utilized standard in the domain of parallel computing.

Figure 19 – A Cluster Computing Layout.



Source: Created by the author

The server is provisioned with a dual-port 2.5-gigabit network adapter. In contrast, node 1 and node 2 are equipped with single-port 2.5-gigabit network adapters, which nonetheless provide efficient and stable data transfer capabilities. This deliberate network configuration is optimized to facilitate low-latency, high-speed message exchanges, a characteristic paramount for the synchronized and efficient execution of distributed algorithms.

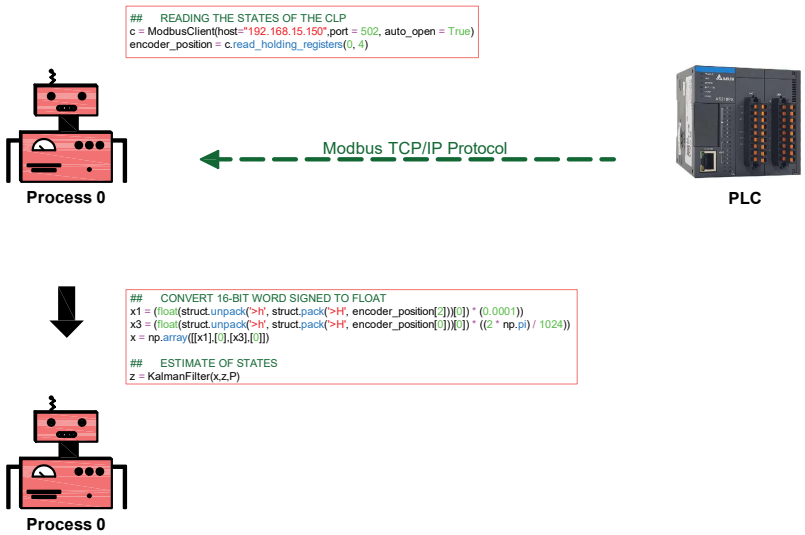
6.2 PARALLEL PROGRAMMING ARCHITECTURE

To simplify understanding, task distribution across computational cores is illustrated using representations of robotic agents with code snippets. The system operates under a master-worker model implemented in Python using the `mpi4py` library (DALCIN *et al.*, 2023), which provides an interface to the MPI. In this architecture, **Process 0** acts as the central coordinator of the primary control loop, responsible for managing task distribution, synchronizing worker processes, and ensuring coherent system execution.

The **Process 0** is responsible for distributing computational tasks, synchronizing worker processes, and ensuring consistent execution and low-latency communication throughout the system. The cycle begins with real-time data acquisition, where **Process 0** requests essential sensors data (states) from the PLC using the `pyModbusTCP` library (FAUTH, 2024).

The data is received in 16-bit word signed format, which represents integers. To convert these values into floating-point numbers for further processing, a custom conversion function was implemented using Python's `struct` module. This data is then processed by a Kalman filter for states estimation, as described in Section 2.3, generating robust and accurate estimates that are essential to mitigate measurement noise and improve the reliability of the control system, as illustrated in Figure 20.

Figure 20 – Process 0 Requesting real-time data from the PLC and States Estimation via Kalman Filter.

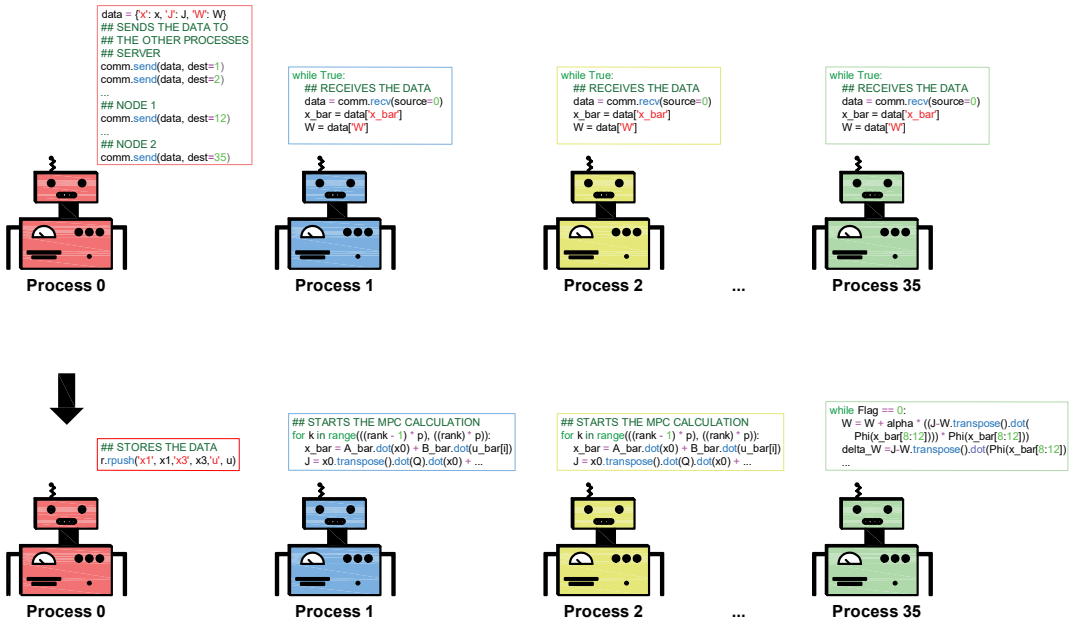


Source: Created by the author, based on (CUTAJAR, 2021).

Once the system states are estimated, the updated state vectors and control parameters are distributed to the worker processes to initialize their parallel computations. Each worker process (Process $1,2,\ldots,35$) executes a distinct segment of either the reinforcement learning task, as detailed in Section 5.3, or the control optimization task, as described in Section 5.4, operating in parallel to enhance computational efficiency.

While the worker processes carry out their computations, **Process 0** concurrently stores the estimated states values in the Redis database using the `redis-py` library (MCCURDY; CONTRIBUTORS, 2021), ensuring efficient data logging and availability for external monitoring in real-time, as shown in Figure 21.

Figure 21 – Distribution of states vectors and control parameters from Process 0 to worker processes.

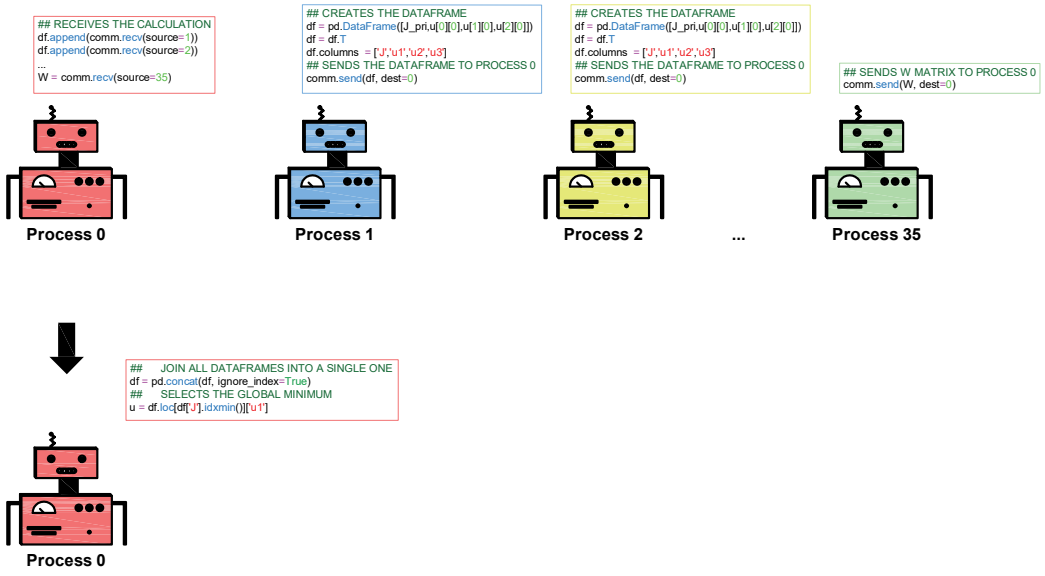


Source: Created by the author, based on (CUTAJAR, 2021).

After completing their assigned computations, the worker processes transmit their results, including the cost values J or matrix W, back to **Process 0**. As the core of the control system, **Process 0** performs optimal action determination by evaluating the control policy and minimizing the cost function J. The selected optimal action, corresponding to the first control value u_k as mentioned in Section 2.5.

The optimal action u_k is converted into a 16-bit signed word using the `struct` module and immediately transmitted to the PLC via the `pyModbusTCP` library.

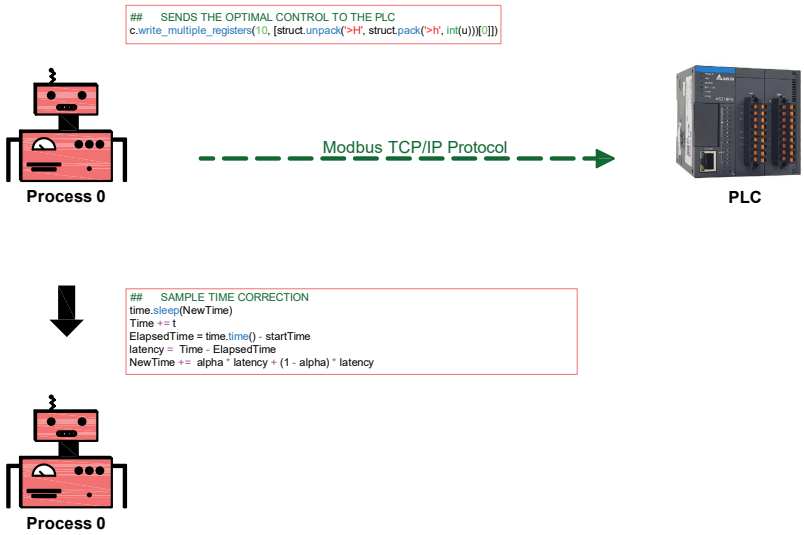
Figure 22 – Worker processes return results to Process 0 for optimal action determination and PLC transmission.



Source: Created by the author, based on (CUTAJAR, 2021).

A vital function of **Process 0** is sampling time correction, actively compensating for communication latency to guarantee uniform sampling intervals, illustrated in Figure 23.

Figure 23 – Sampling time correction performed by Process 0.



Source: Created by the author, based on (CUTAJAR, 2021).

This continuous, real-time loop of estimation, optimization, and actuation, maintained through rapid system cycling after each command dispatch, is enabled by the sophisticated parallel processing structure. This architecture is crucial for meeting stringent real-time constraints, ensuring the control system remains stable and highly responsive by significantly reducing overall computation time.

6.3 PERFORMANCE EVALUATION

The system employs a closed-loop control architecture in which computational outputs from reinforcement learning and control algorithms are executed on a high-performance computing cluster and transmitted to the physical system via the Modbus/TCP protocol (Modbus Organization, 2012). This setup enables real-time actuation, such as controlling the servomotor that drives the cart, while sensor data (e.g., pendulum angle and cart position) is continuously sent back to the cluster for analysis.

Real-world validation is conducted by comparing the observed behavior of the physical prototype with the simulated predictions of the control system. To assess the computational performance of the cluster, a key experiment involved progressively decreasing the number of available processes and analyzing the resulting execution times, thereby quantifying the benefits of parallelization within the control loop.

However, as discussed in previous chapters, the inverted pendulum is a highly unstable system that demands rapid and precise control actions. When the sampling time increases significantly, due to delays introduced by optimization or neural network training, the system may become uncontrollable, leading to divergent behavior. This invalidates direct comparisons in speedup analysis, as the resulting control behavior would differ between the sequential and parallel executions.

To address this limitation and still enable cluster performance evaluation, the experiment was structured in two stages. First, the system was executed using all available cores on the cluster, and the resulting states and control values were recorded in a Redis database. In the second stage, as the number of available processes was progressively reduced, the physical actuator was disabled to avoid unsafe conditions, since the control delays introduced by prolonged sampling periods could compromise the stability of the system.

To ensure a fair comparison, both the sequential and parallel systems executed the same control computations, yielding identical results due to an unchanged mathematical formulation.

While the parallel implementation fully leveraged the cluster's computational power with dynamic sampling time corrections for communication latency, all latency and synchronization overheads were intentionally factored into the speedup analysis, irrespective to the number of processes.

To facilitate a meaningful evaluation, the number of processes was varied from 1 to 36, and for each configuration, the time required to complete 20 control cycles was recorded. Each cycle consisted of an optimization stage followed by a decision dispatch, all executed within a tight real-time constraint of $50\ ms$ per iteration. The results confirm that parallelization significantly reduced the computation time per cycle, enabling more complex optimization problems to be solved within the fixed sampling window.

Table 6 – Performance improvement with increasing number of processes.

Processes	Total Time (s)	Avg. Time per Cycle (ms)	Speedup
1	21.5668	1078.34	$1.00\times$
4	4.3222	216.11	$4.99\times$
8	2.3046	115.23	$9.36\times$
12	1.7452	87.26	$12.36\times$
16	1.3842	69.21	$15.58\times$
20	1.1682	58.41	$18.46\times$
24	1.0777	53.89	$20.01\times$
28	1.0482	52.41	$20.59\times$
32	1.0356	51.78	$20.83\times$
36	1.0000	50.00	$21.57\times$

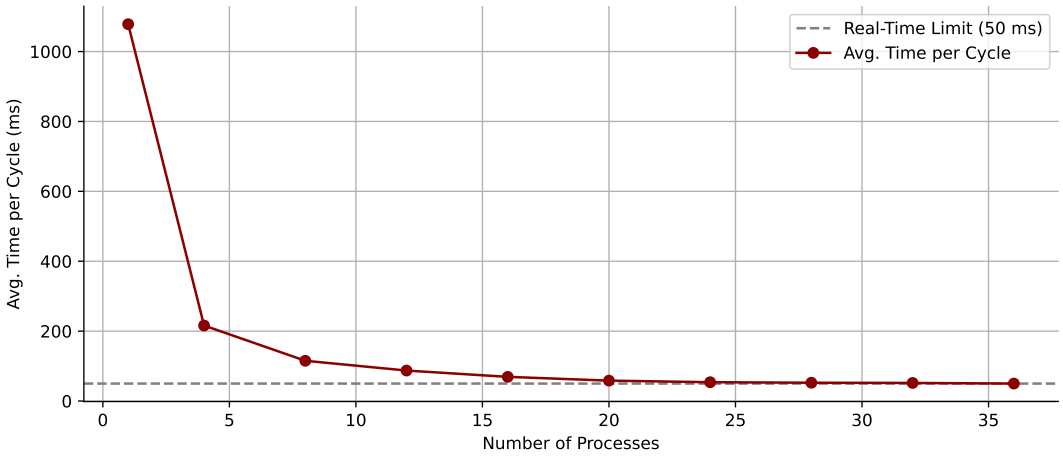
Source: Created by the author.

As presented in Table 6, the execution time with a single process reached approximately $1078\ ms$ per cycle, which is more than twenty times the available $50\ ms$ sampling window, thus rendering the control loop infeasible in real time.

When four processes were used, the average time per cycle decreased to $216\ ms$, and continued to drop to 115 ms with eight processes, showing substantial improvement, though still slightly above the real-time threshold.

The control system only became viable for real-time operation when twelve or more processes were allocated, with the sampling period dropping below $100\ ms$. Beyond sixteen processes, the speedup curve began to plateau, with execution times converging near the $50\ ms$ mark when thirty-six processes were used.

Figure 24 – Average execution time per control cycle versus number of processes.

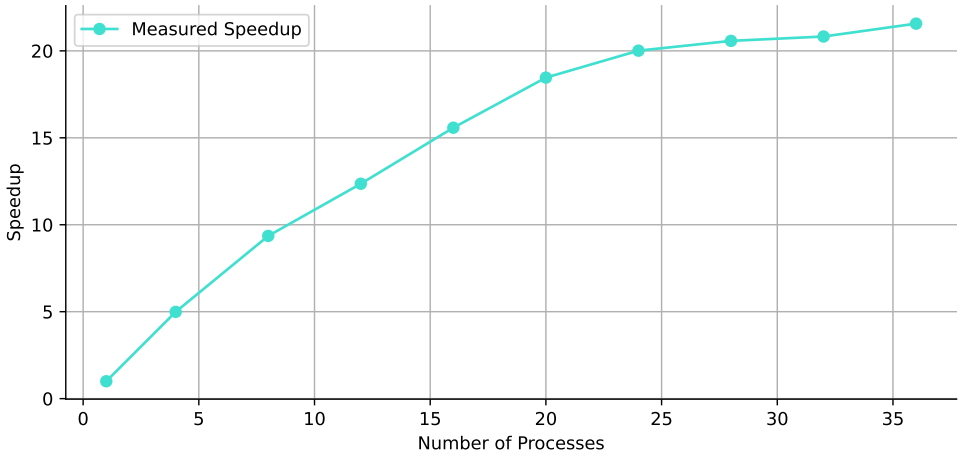


Source: Created by the author.

This saturation suggests that the computational workload per process becomes too small to justify the communication overhead inherent in distributed computing. Such a trend is typical in high-performance computing scenarios, where the cost of inter-process communication increasingly offsets the benefits of parallelism.

The diminishing returns observed beyond 20 processes are consistent with the principles of Amdahl's Law (AMDAHL, 1967), which states that the theoretical speedup of a parallelized system is limited by the portion of the process that remains sequential. This theoretical limit is visually reflected in the plateauing behavior of the speedup curve in Figure 25.

Figure 25 – Speedup Analysis with Decoupled Actuator to Ensure Comparability between Sequential and Parallel Executions



Source: Created by the author.

The initial gains from parallelism are substantial, reaching a speedup of approximately $21.6\times$ when all 36 cores are used. However, further increases in process count beyond 20 yield diminishing returns, reflecting practical trade-offs between computational acceleration and communication overhead.

These findings demonstrate that while the cluster-based implementation effectively accelerates the control loop, careful tuning of the number of processes is required to achieve optimal performance. Assigning excessive cores may not only fail to improve speed but could also introduce unnecessary complexity and synchronization delays.

Ultimately, this experiment validates the efficacy of parallel computing in embedded real-time control systems. By integrating parallel optimization within the fixed $50\ ms$ sampling constraint and analyzing performance across various core configurations, we demonstrate that real-time feasibility can be achieved without compromising computational accuracy, provided the system architecture is appropriately scaled. The evaluation methodology incorporated all relevant computational penalties, including sampling time adjustments and inter-process synchronization, ensuring a realistic assessment of system performance. By measuring total execution time under different process counts, the results offer a fair and accurate benchmark of the benefits and limitations of parallelization in time-sensitive control applications.

6.4 REAL-TIME TIMING COMPENSATION AND LATENCY CONTROL

To ensure the stability and reliability of our real-time control system, we implemented and evaluated a dual-strategy approach to mitigate timing deviations resulting from execution delays, system jitter, and CPU scheduling variability. This method combines a static safeguard with a dynamic adaptive mechanism to prevent latency accumulation and maintain precise, periodic execution.

The first component is a fixed waiting window of $15\ ms$, inserted at the end of each control cycle. This static buffer serves as a foundational safeguard, absorbing minor latency variations, creating a synchronization point for parallel processes, and reducing excessive CPU load. By preventing the gradual buildup of timing errors, it ensures a consistent sampling frequency throughout operation.

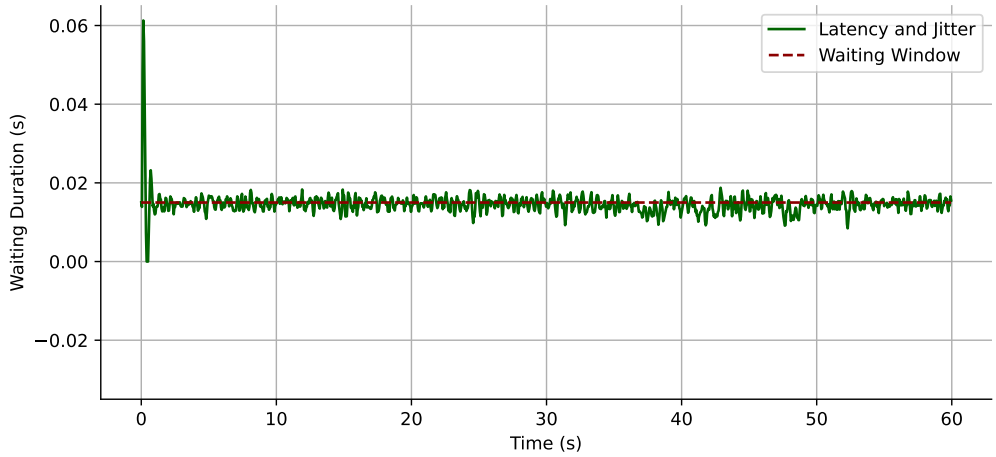
The second component is an adaptive latency smoother designed to handle larger fluctuations. It implements an Exponentially Weighted Moving Average (EWMA) filter to process the raw latency measured during each cycle, defined as the deviation between the actual

and expected execution times. Since raw latency data can be affected by high-frequency noise introduced by system jitter, direct use of these measurements may destabilize the controller. The EWMA filter mitigates this by removing transient deviations while preserving the underlying trend. Tuned with a smoothing factor α, the filter balances responsiveness and noise suppression. The resulting smoothed latency value is then used to inform the timing compensation for the next control cycle.

The application of EWMA for latency smoothing is supported by previous research. For instance, (WU *et al.*, 2008) demonstrated that EWMA-based controllers maintain system stability and performance even in environments with variable communication delays, validating the filter's applicability to real-time control systems.

In practice, the fixed $15\ ms$ waiting window effectively absorbed most variations in execution time. The observed average cycle latency remained close to the $15\ ms$ target, with a standard deviation of approximately $3\ ms$. This low variability indicates minimal jitter and confirms a stable, predictable timing profile, as illustrated in Figure 26.

Figure 26 – Real-Time Execution Timing: Latency, Jitter, and Synchronization Buffer



Source: Created by the author

The EWMA filter further improved system robustness by attenuating high-frequency latency spikes caused by OS scheduling noise or background tasks. As a result, the timing compensation mechanism responded smoothly to persistent latency shifts without overreacting to transient disturbances.

Overall, the evaluation confirmed that the dual-strategy approach met its performance objectives. Sampling periodicity was preserved across thousands of control iterations, cumulative time drift was effectively eliminated, and CPU usage remained moderate and stable, with no

sustained spikes.

In conclusion, the synergistic use of a fixed waiting window and an adaptive EWMA filter offers a precise and robust solution for maintaining temporal integrity in real-time control systems operating under variable execution conditions.

7 EXPERIMENTAL RESULTS AND ANALYSIS

The validation strategy for the proposed RLMPC framework was meticulously crafted to integrate online optimization and real-time execution on industrial hardware. Specifically, a PLC and a servo motor were incorporated into the experimental setup, enabling a realistic assessment of the framework's performance in conditions closely resembling real-world industrial environments.

This hybrid validation methodology served a dual purpose: to ensure theoretical soundness through software-based optimization while rigorously verifying the practical feasibility and robustness of the system under physical constraints and dynamic uncertainties.

The evaluation focused on three interdependent and equally critical dimensions:

1. **Control Accuracy:** We conducted a comprehensive assessment of the system's capability to stabilize the inverted pendulum, follow predefined reference trajectories, and endure a range of external disturbances. These tests confirmed the controller's robustness and precision, even in the presence of unmodeled dynamics and noise.

2. **Computational Efficiency:** Execution time was a key metric in determining the real-time applicability of the framework. By leveraging parallelization techniques, particularly for reinforcement learning parameter tuning and optimization stages, the system achieved control loop executions well within the sampling period. This demonstrates the effectiveness of the proposed framework in overcoming one of the main challenges of MPC, solving constrained optimization problems in real time.

3. **Practical Applicability:** The framework's seamless integration with industrial-grade hardware components was critically evaluated. A thorough evaluation was conducted on communication latency, the reliability of data exchange between the optimization engine and the PLC, and the responsiveness of the actuation system. The results validated the system's operational reliability and highlighted its potential for deployment in fast, nonlinear mechatronic and electrical systems.

To contextualize these findings, this chapter presents a comparative analysis with established baseline methods, underscoring the performance improvements enabled by the

proposed architecture. This comprehensive evaluation not only validates the results but also aligns them with current advancements.

To evaluate the control performance of the proposed RLMPC framework, we adopted five quantitative metrics: stabilization time, settling time, maximum overshoot, control effort, and computational time. Together, these indicators provide a comprehensive assessment of the system's accuracy, efficiency, and real-time feasibility.

Experimental results confirmed the effectiveness and consistency of the RLMPC in stabilizing the inverted pendulum across a wide range of initial conditions, even in the presence of unmodeled dynamics and external disturbances.

7.1 DESIGN PARAMETERS FOR ESTIMATION AND CONTROL

To ensure robust estimation and control performance, the process and measurement noise covariance matrices of the Kalman filter were defined as,

$$\bar{Q} = \begin{bmatrix} 10^{-5} & 0 \\ 0 & 10^{-3} \end{bmatrix} \quad \bar{R} = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 10^{-2} \end{bmatrix}$$

The measurement model was expressed through the mapping matrix H, which relates the system states to the measurement domain,

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

The state transition and input matrices, used in the prediction stage of the filter, were defined respectively as,

$$\phi = \begin{bmatrix} 1 & 0.05 \\ 0 & 1 \end{bmatrix} \qquad \Upsilon = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix}$$

For the controller design, the standard quadratic cost function of the Model Predictive Control (MPC) framework was adopted. The state and input weighting matrices were configured as,

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10^{-3} & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 10^{-3} \end{bmatrix}$$

and

$$R = \begin{bmatrix} 0.04 \end{bmatrix}$$

This choice aligns with the classical LQR tuning approach, where the weighting matrices were adjusted to heavily penalize large cart displacements and pendulum deviations while applying relatively smaller penalties to velocity states.

The RLMPC was configured with a prediction horizon of $N = 3$. The algorithmic parameters were tuned as $\alpha = 10^{-3}$, $\varepsilon = 10^{-3}$, and $p = 800$, ensuring numerical stability and adequate exploration balance in the learning phase.

The feature mapping $\Phi(x)$, designed to capture nonlinearities of the inverted pendulum dynamics, was constructed using polynomial terms up to cubic order and relevant cross-products of the state variables: $\Phi(x) = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_1^2 \quad x_2^2 \quad x_3^2 \quad x_4^2 \quad x_1x_2 \quad x_1x_3 \quad x_1x_4 \quad x_2x_3 \quad x_2x_4 \quad x_3x_4 \quad x_1^3 \quad x_2^3 \quad x_3^3 \quad x_4^3 \quad x_1^2x_2 \quad x_1^2x_3 \quad x_1^2x_4 \quad x_2^2x_1 \quad x_2^2x_3 \quad x_2^2x_4 \quad x_3^2x_1 \quad x_3^2x_2 \quad x_3^2x_4 \quad x_4^2x_1 \quad x_4^2x_2 \quad x_4^2x_3]^T$ This enriched feature set allowed the controller to approximate nonlinear cost-to-go functions more accurately, enhancing performance under highly unstable pendulum dynamics.

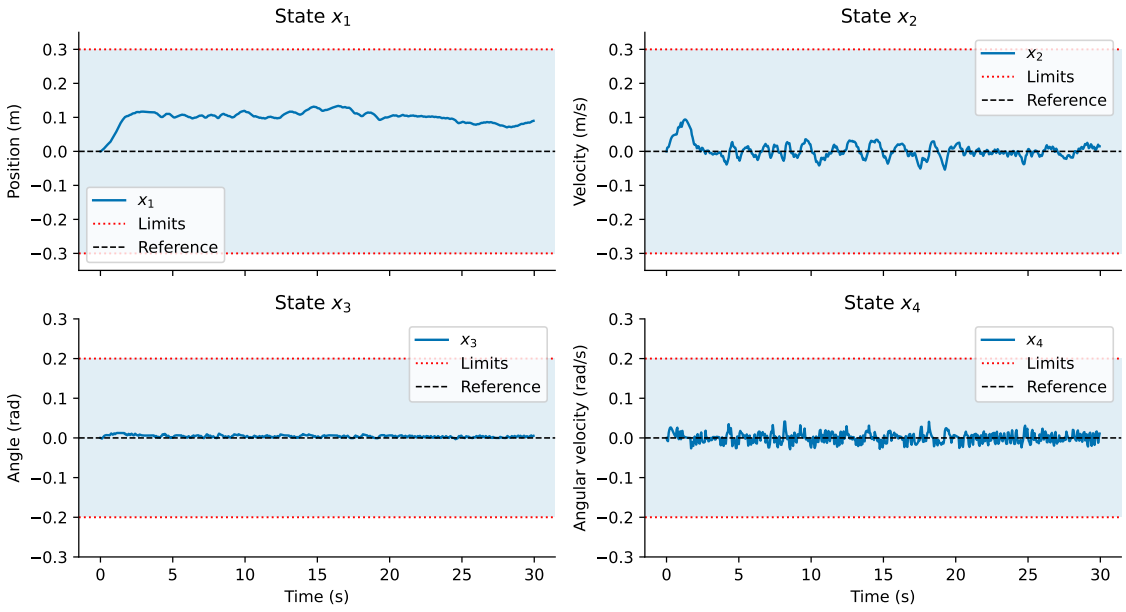
7.2 EXPERIMENTAL RESULTS

The experiments began with the pendulum rod initially positioned at a near-zero angle, enabling the controller to immediately operate within its stabilization zone. This initial condition promoted rapid convergence to the upright equilibrium with minimal oscillation, a common challenge in inverted pendulum systems.

The control objective is to steer the system toward this equilibrium using minimal control effort, which involves driving the pendulum angle x_3 and cart position x_1 to zero while damping residual oscillations. The controller also respected physical constraints, including track limits, angle bounds, and actuator saturation.

Consequently, it generates smooth, bounded control inputs that correct deviations efficiently without inducing additional instability. As illustrated in Figure 27, the time evolution of the states (x_1 to x_4) demonstrates rapid stabilization with negligible overshoot, confirming the efficacy of the RLMPC framework.

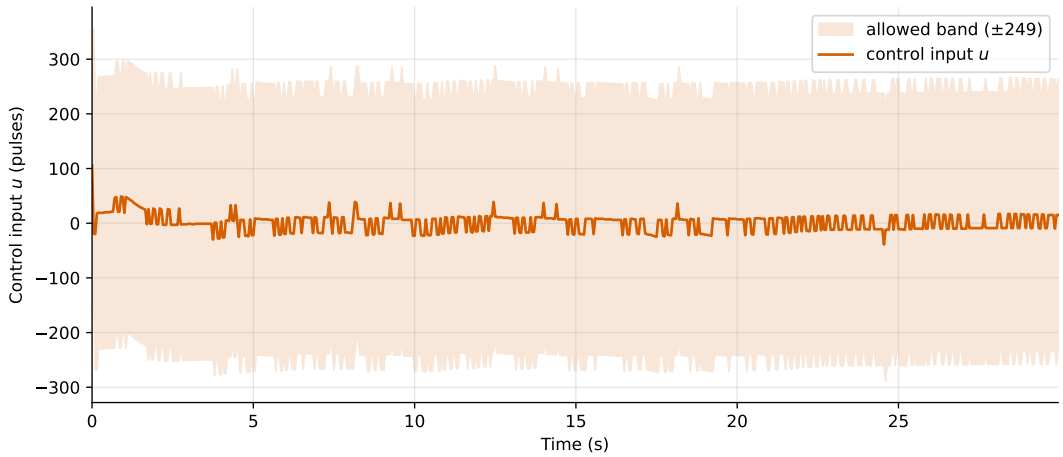
Figure 27 – Time evolution of the system states: cart position x_1, cart velocity x_2, pendulum angle x_3, and angular velocity x_4.



Source: Created by the author

Figure 28 presents the control input $u(t)$ over time. The control effort remained within acceptable bounds, with no excessive or high-frequency actuation, highlighting the controller's efficiency and suitability for real-world applications.

Figure 28 – Time evolution of the control input u.



Source: Created by the author

The controller successfully maintained the pendulum's balance with minimal oscillation. The primary performance metric, the pendulum angle x_3, was held tightly around the upright position, with a mean of 0.00423 rad. The angular excursion was limited to a maximum of 0.02454 rad and a minimum of -0.00614 rad, which corresponds to less than $1.5°$ of movement.

Throughout the stabilization period, the system's other states were kept within tight bounds. The cart's position x_1 and velocity x_2 remained stable, with standard deviations of 0.02061 m and 0.02796 m/s, respectively. Similarly, the pendulum's angular velocity x_4 and the control input u were well-behaved, demonstrating that the overall system was effectively regulated by the controller. It is noteworthy that the cart position x_1 did not converge to zero. This behavior can be attributed to the specific tuning of the Q and R matrices, in which the pendulum angle state x_3 was prioritized during the test.

The RLMPC controller demonstrated robust and precise stabilization by consistently maintaining the pendulum angle within a narrow band of the upright equilibrium. Its effective transient and steady-state performance across multiple trials underscores its practical applicability for challenging real-world control problems.

The experimental results confirmed the computational efficiency of the proposed controller, which consistently executed the full control loop in as little as $34\ ms$ by leveraging 36 parallel processes. This execution time remains well below the stringent $50\ ms$ sampling period required for the real-time stabilization of the inverted pendulum system, thereby affirming the framework's viability for time-critical industrial applications.

The remaining $16\ ms$ were systematically allocated for sampling time correction due to process synchronization and latency, ensuring temporal alignment and coherence throughout the process. This sustained, deterministic performance under continuous operation demonstrates the framework's robustness and stability, both essential characteristics for deployment in safety-critical and high-availability industrial environments. Collectively, these results highlight the success of the RLMPC approach in addressing one of the most significant challenges of traditional MPC: the real-time solution of constrained optimization problems within tight execution deadlines.

7.3 LIMITATIONS

While the RLMPC framework demonstrates promising results, its current implementation presents several limitations that offer clear avenues for future improvement.

- **Computational Assumptions - Homogeneous Cluster Requirement:** All experiments were run using identical hardware and consistent network conditions. Real-world heterogeneous computing environments could introduce significant performance bottlenecks and load imbalances, which would require developing adaptive scheduling algorithms.

- **Parameter Sensitivity - Predictive Horizon and Input Precision:** The controller's performance is sensitive to the choice of predictive horizon and the numerical precision of control inputs. While longer prediction horizons improve long-term decision-making, they also significantly increase the computational burden. Similarly, higher precision in control signals demands more processing power. These trade-offs must be carefully managed to ensure that the framework remains within real-time execution bounds without sacrificing control accuracy.

- **System Latencies - Communication Delays:** Despite optimizations applied to communication protocols and hardware interfaces, residual latencies were observed in the communication between the PLC and the server. While acceptable for the inverted pendulum system, these latencies could become a substantial impediment in larger, more complex, or geographically distributed control implementations that rely on wider-distance network communication.

- **MPI fault-tolerance limitation:** A current limitation of the proposed system lies in the fault-handling model adopted by MPI (Message Passing Interface). By default, MPI follows a "fail-stop"behavior, in which the failure of a single process, such as the loss of a node, results in the termination of all remaining processes within the communicator. This occurs because inter-process communication is essential for ensuring the consistency and progress of the parallel application. Without dedicated fault-tolerance mechanisms, a single node failure may lead to deadlocks or abrupt termination, as dictated by the default error handler `MPI_ERRORS_ARE_FATAL`. Consequently, the current implementation lacks resilience to partial hardware failures, which can jeopardize the stability of long-running optimization routines. Addressing this limitation will require the integration of advanced fault-tolerance strategies, which is identified as a key direction for future work.

8 CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

This dissertation detailed the design, implementation, and experimental validation of a Reinforcement Learning-based Model Predictive Control (RLMPC) framework. The successful real-time stabilization of an inverted pendulum a benchmark for highly nonlinear control demonstrates a significant step toward deploying intelligent controllers for complex industrial applications operating under stringent performance constraints.

This dissertation presented the design, implementation, and experimental validation of a Reinforcement Learning-based Model Predictive Control (RLMPC) framework for real-time stabilization of an inverted pendulum system, a classical, highly nonlinear benchmark in control engineering. The work responds to the growing demand for intelligent controllers capable of managing nonlinear dynamics under stringent real-time constraints, which are common in modern industrial applications.

The proposed RLMPC framework unites adaptive reinforcement learning, optimal trajectory planning via model predictive control, and parallel computing techniques to fulfill strict execution deadlines. This integrated approach was implemented on standard industrial hardware, including a PLC and servo motor, proving its feasibility beyond simulation and confirming its readiness for practical, safety-critical environments.

The experimental results confirmed the controller's exceptional robustness and precision. The system consistently maintained the pendulum angle near its upright equilibrium with minimal overshoot, while all system states and control actions showed low variability. By utilizing parallelization, the control algorithm reliably achieved cycle times as low as $50\ ms$, satisfying the real-time requirements essential for industrial deployment without compromising control performance.

At the algorithmic level, the RLMPC introduces a novel integration of reinforcement learning-derived value functions as terminal costs within the MPC structure, following Policy Iteration (PI) principles. This approach ensures monotonic improvement of the closed-loop value function and guarantees stability, even as prediction horizons are dynamically adjusted to balance optimality and computational burden.

In essence, this research validates that learning-based predictive controllers can achieve

both theoretical rigor and high performance on real industrial platforms. The developed RLMPC framework offers a robust, scalable, and adaptable solution for intelligent real-time control, paving the way for its application to a wide range of challenging nonlinear systems in modern industry.

8.2 FUTURE WORK

Based on the encouraging outcomes of this study, we outline future directions aimed at improving the parallelized RLMPC framework and extending its use to increasingly challenging control environments.

1. **Algorithmic Improvements:** Building upon the discrete inverted pendulum model, future research will explore the integration of Physics-Informed Neural Networks (PINNs) into the RLMPC framework. This extension aims to enable control beyond the conventional stabilization region, accommodating broader state-space dynamics. Although PINNs offer strong capabilities for solving differential equations, their training remains a challenge due to the sensitive tuning of loss function weights. To overcome this, we propose parallelizing the training process, which could enhance optimization efficiency and support improved tuning of reinforcement learning parameters such as learning rates, reward shaping, and exploration strategies.

2. **Adaptive Parallelization:** The current implementation assumes a homogeneous computing environment; however, real-world industrial applications often rely on heterogeneous hardware. Future efforts will focus on designing adaptive load-balancing algorithms and dynamic task allocation mechanisms. These improvements aim to preserve real-time computational efficiency and ensure robust performance across clusters with varying processing capabilities. Investigating middleware and runtime solutions capable of intelligent workload distribution based on node performance and network latency will be central to this research.

3. **Hardware Advancements:** To meet the demands of high-frequency control applications, we plan to investigate the horizontal scalability of the computing cluster. By distributing the most computationally intensive components of the RLMPC algorithm across a larger and more diverse cluster, we aim to support faster system dynamics and extend the prediction

horizon. Additionally, enhancements in sensing resolution, such as highest resolution encoder, will improve the granularity of control signals.

4. **Industrial Validation:** Although the current experiments confirm the feasibility of the proposed framework, its long-term reliability under sustained industrial operation must still be evaluated. Future work will include extended validation campaigns in real production environments, including continuous operation studies over months. These trials will assess the effects of component degradation, environmental variability, and system drift on control performance. The outcomes will provide essential insights into robustness, maintainability, and lifecycle costs, key metrics for industrial adoption.

CONCLUDING REMARKS

This research demonstrated that parallelized RLMPC, when carefully designed using cluster computing and distributed control strategies, is capable of meeting the rigorous demands of industrial control applications. Despite persistent challenges in parameter tuning and deployment, the findings confirm that parallelized learning-based approaches are transitioning from theoretical interest to viable solutions in practical control scenarios. The future work outlined here aims not just to improve the framework, but to transform it into a plug-and-play solution for the next generation of intelligent control systems, pushing the boundaries of autonomous control in real-world industrial settings.

REFERENCES

ALMOMANI, Ammar *et al.* Summit supercomputer architecture and early performance. *In*: **Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. *[S.l.]*: IEEE, 2019. p. 308–316.

AMDAHL, Gene M. Validity of the single processor approach to achieving large-scale computing capabilities. *In*: **AFIPS Conference Proceedings**. *[S.l.]*: Springer, 1967. v. 30, p. 483–485.

Apache Friends. **XAMPP: Apache Distribution Containing MySQL, PHP, and Perl**. *[S.l.]*, 2024. Available at: https://www.apachefriends.org. Accessed on: Jun. 9, 2025.

ATHANS, Michael. The role and use of the stochastic linear-quadratic-gaussian problem in control system design. **IEEE Transactions on Automatic Control**, v. 16, n. 6, p. 529–552, 1971.

BALULA, Samuel. **Nonlinear Control of an Inverted Pendulum**. 2016. (Master's Thesis) — Instituto Superior Técnico (Técnico Lisboa), Lisbon, 2016.

BERBERICH, Julian; KÖHLER, Johannes; MÜLLER, Matthias A.; ALLGÖWER, Frank. Linear tracking MPC for nonlinear systems–part i: The model-based case. **IEEE Transactions on Automatic Control**, v. 67, n. 9, p. 4390–4405, 2022.

BOCCIA, Andrea; GRÜNE, Lars; WORTHMANN, Karl. Stability and feasibility of state-constrained linear MPC without stabilizing terminal constraints. **Systems & Control Letters**, v. 72, p. 14–21, 2014.

BONGARD, Joscha; BERBERICH, Julian; KÖHLER, Johannes; ALLGÖWER, Frank. Robust stability analysis of a simple data-driven model predictive control approach. **IEEE Transactions on Automatic Control**, v. 68, n. 5, p. 2625–2637, 2022.

BOUBAKER, Olfa. The inverted pendulum benchmark in nonlinear control theory: A survey. **International Journal of Advanced Robotic Systems**, v. 10, n. 5, p. 233, 2013.

BRADTKE, Steven J. **Reinforcement Learning Applied to Linear Quadratic Regulation**. Amherst, MA, 1992.

BUSCHERMÖHLE, Philipp; JOUINI, Taouba; LILGE, Torsten; MÜLLER, Matthias A. Disturbance feedback-based model predictive control in uncertain dynamic environments. **arXiv**, 2024. Preprint. Available at: https://arxiv.org/abs/2404.09893. Accessed on: Jun. 9, 2025.

Canonical Ltd. **Ubuntu 22.04.1 LTS Documentation**. *[S.l.]*, 2022. Available at: https://ubuntu.com. Accessed on: Jun. 9, 2025.

CHEE, Kong Yao; HSIEH, M. Ani; MATNI, Nikolai. Learning-enhanced nonlinear model predictive control using knowledge-based neural ordinary differential equations and deep ensembles. *In*: PMLR. **Learning for Dynamics and Control Conference**. *[S.l.]*, 2023. p. 1125–1137.

CISNEROS, Pablo Sebastian Gonzalez. **Quasi-Linear Model Predictive Control: Stability, Modelling and Implementation**. 2021. (PhD Thesis) — Technische Universität Hamburg, Hamburg, 2021.

COŞKUN, Serdar. Non-linear control of inverted pendulum. **Çukurova Üniversitesi Mühendislik Fakültesi Dergisi**, v. 35, n. 1, p. 27–38, 2020.

CUTAJAR, James. **Learn Parallel Computing in Python: Discover Multithreading, Multiprocessing, Concurrency & Parallel Programming with Practical and Fun Examples in Python**. 2021. Udemy. Available at: https://www.udemy.com/course/parallel-computing-in-python/. Accessed on: Jun. 1, 2025.

DALCIN, Lisandro *et al.* **mpi4py: MPI for Python (Version 4.0.3)**. *[S.l.]*, 2023. Available at: https://mpi4py.readthedocs.io/. Accessed on: Jun. 9, 2025.

Dassault Systèmes. **SolidWorks: 3D CAD Design Software**. *[S.l.]*, 2024. Version 2024. Available at: https://www.solidworks.com. Accessed on: Jun. 9, 2025.

Delta Electronics. **ASDA-Soft: Servo Drive Configuration Software**. *[S.l.]*, 2024. Available at: https://www.deltaww.com. Accessed on: Jun. 9, 2025.

Delta Electronics. **Compact Modular Mid-range PLC AS Series: Hardware Manual**. *[S.l.]*, 2018. Available at: https://www.deltronics.ru/images/manual/AS300_AS200_HM_EN_20180209.pdf. Accessed on: Jul. 8, 2025.

Delta Electronics. **ISPSoft: PLC Programming Software**. *[S.l.]*, 2024. Version 3.x. Available at: https://www.deltaww.com. Accessed on: Jun. 9, 2025.

DONGARRA, Jack; MEUER, Hans; STROHMAIER, Erich; SIMON, Horst. **Top500 Supercomputer Sites**. 2020. Available at: https://www.top500.org. Accessed on: Jun. 9, 2025.

DONGE, Vrushabh S.; LIAN, Bosen; LEWIS, Frank L.; DAVOUDI, Ali. Data-efficient reinforcement learning for complex nonlinear systems. **IEEE Transactions on Cybernetics**, v. 54, n. 3, p. 1391–1402, 2023.

EKAPUTRI, Cahyantari; SYAICHU-ROHMAN, Arief. Implementation model predictive control (MPC) algorithm-3 for inverted pendulum. *In*: IEEE. **2012 IEEE Control and System Graduate Research Colloquium**. *[S.l.]*, 2012. p. 116–122.

FARAGHER, Ramsey. Understanding the basis of the kalman filter via a simple and intuitive derivation. **IEEE Signal Processing Magazine**, v. 29, n. 5, p. 128–132, 2012. Lecture notes.

FAUTH, Sébastien. **pyModbusTCP: Modbus client for Python (Version 0.3.0)**. *[S.l.]*, 2024. Available at: https://github.com/sourceperl/pyModbusTCP. Accessed on: Jun. 9, 2025.

GHANAVATI, Meysam; MAJD, Vahid Johari; GHANAVATI, Malek. Control of inverted pendulum system by using a new robust model predictive control strategy. *In*: IEEE. **2011 International Siberian Conference on Control and Communications (SIBCON)**. *[S.l.]*, 2011. p. 27–32.

GINZBURG-GANZ, Elinor; SEGEV, Itay; BALABANOV, Alexander; SEGEV, Elior; NAVEH, Sivan Kaully; MACHLEV, Ram; BELIKOV, Juri; KATZIR, Liran; KEREN, Sarah; LEVRON, Yoash. Reinforcement learning model-based and model-free paradigms for optimal control problems in power systems: Comprehensive review and future directions. **Energies**, v. 17, n. 21, p. 5307, 2024.

GREWAL, Mohinder S.; ANDREWS, Angus P. **Kalman Filtering: Theory and Practice with MATLAB**. *[S.l.]*: John Wiley & Sons, 2014.

HALLIDAY, David; RESNICK, Robert; WALKER, Jearl. **Fundamentals of Physics**. 9th. ed. *[S.l.]*: Wiley, 2011.

HOLKAR, K. S.; WAGHMARE, Laxman M. An overview of model predictive control. **International Journal of Control and Automation**, v. 3, n. 4, p. 47–63, 2010.

HUNTER, John D.; CONTRIBUTORS. **matplotlib: Python 2D plotting library (Version 3.8.4)**. *[S.l.]*, 2024. Available at: https://matplotlib.org. Accessed on: Jun. 9, 2025.

HWANGBO, Jemin; SA, Inkyu; SIEGWART, Roland; HUTTER, Marco. Control of a quadrotor with reinforcement learning. **IEEE Robotics and Automation Letters**, v. 2, n. 4, p. 2096–2103, 2017.

INMAN, Daniel J. **Engineering Vibration**. 4th. ed. Upper Saddle River, NJ: Pearson, 2014.

International Electrotechnical Commission. **IEC 61131-3: Programmable Controllers – Part 3: Programming Languages**. Geneva, Switzerland, 2013. Standard.

JANABI-SHARIFI, Farrokh; HAYWARD, Vincent; CHEN, C.-S. J. Discrete-time adaptive windowing for velocity estimation. **IEEE Transactions on Control Systems Technology**, v. 8, n. 6, p. 1003–1009, 2000.

JANIESCH, Christian; ZSCHECH, Patrick; HEINRICH, Kai. Machine learning and deep learning. **Electronic Markets**, v. 31, n. 3, p. 685–695, 2021.

KALMAN, Rudolph Emil. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, v. 82, n. 1, p. 35–45, 1960.

KALMAN, Rudolf Emil *et al.* Contributions to the theory of optimal control. **Boletín de la Sociedad Matemática Mexicana**, v. 5, n. 2, p. 102–119, 1960.

KIUMARSI, Bahare; LEWIS, Frank L.; MODARES, Hamidreza; KARIMPOUR, Ali; NAGHIBI-SISTANI, Mohammad-Bagher. Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. **Automatica**, v. 50, n. 4, p. 1167–1175, 2014.

KOELEWIJN, Patrick J. W.; CISNEROS, Pablo S. G.; WERNER, Herbert; TOTH, Roland. LPV control of a gyroscope with inverted pendulum attachment. **IFAC-PapersOnLine**, v. 51, n. 26, p. 49–54, 2018.

LAARAIEDH, Mohamed. Implementation of kalman filter with python language. **arXiv**, 2012. Preprint. Available at: https://arxiv.org/abs/1204.0375. Accessed on: Jun. 9, 2025.

LESHNO, Moshe; LIN, Vladimir Ya; PINKUS, Allan; SCHOCKEN, Shimon. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. **Neural Networks**, v. 6, n. 6, p. 861–867, 1993.

LEWIS, Frank L.; VRABIE, Draguna. Reinforcement learning and adaptive dynamic programming for feedback control. **IEEE Circuits and Systems Magazine**, v. 9, n. 3, p. 32–50, 2009.

LEWIS, Frank L.; VRABIE, Draguna; SYRMOS, Vassilis L. **Optimal Control**. *[S.l.]*: John Wiley & Sons, 2012.

LIN, Min; SUN, Zhongqi; XIA, Yuanqing; ZHANG, Jinhui. Reinforcement learning-based model predictive control for discrete-time systems. **IEEE Transactions on Neural Networks and Learning Systems**, 2023.

LIN, Min; XIA, Yuanqing; SUN, Zhongqi; DAI, Li. Learning-based model predictive control under value iteration with finite approximation errors. **International Journal of Robust and Nonlinear Control**, v. 34, n. 4, p. 2946–2971, 2024.

MANKAR, O. S.; VADIRAJACHARYA, K. Kalman filter analysis in dynamic state of power system. **International Research Journal of Engineering and Technology (IRJET)**, v. 2, p. 1170–1173, 2015.

MATHWORKS. **Optimal Control**. 2025. Available at: https://www.mathworks.com/discovery/optimal-control.html. Accessed on: Jan. 25, 2025.

MATLAB. **Introduction to Control Systems**. 2025. Available at: https://www.youtube.com/watch?v=cEWnixjNdzs. Accessed on: Jan. 25, 2025.

MATLAB. **Introduction to Optimal Control Design**. 2025. Available at: https://www.youtube.com/watch?v=E_RDCFOlJx4&t=676s. Accessed on: Jan. 25, 2025.

MCCURDY, Andy; CONTRIBUTORS. **redis: Python Client for Redis (Version 6.0.0)**. *[S.l.]*, 2021. Available at: https://pypi.org/project/redis/6.0.0/. Accessed on: Jun. 9, 2025.

MERIAM, J. L.; KRAIGE, L. G. **Engineering Mechanics: Dynamics**. 6th. ed. Hoboken, NJ: Wiley, 2007.

MINOUCHEHR, Niloufar; HOSSEINI-SANI, Seyyed Kamal. Design of model predictive control of two-wheeled inverted pendulum robot. *In*: IEEE. **2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)**. *[S.l.]*, 2015. p. 456–462.

Modbus Organization. **Modbus Application Protocol Specification V1.1b3**. 2012. Available at: https://modbus.org/specs.php. Accessed on: Jun. 9, 2025.

MOOS, Janosch; HANSEL, Kay; ABDULSAMAD, Hany; STARK, Svenja; CLEVER, Debora; PETERS, Jan. Robust reinforcement learning: A review of foundations and recent advances. **Machine Learning and Knowledge Extraction**, v. 4, n. 1, p. 276–315, 2022.

NAIDU, Desineni Subbaram. **Optimal Control Systems**. Boca Raton, FL, USA: CRC Press, 2003. (Electrical Engineering Textbook Series).

NISE, Norman S. **Control Systems Engineering**. 6th. ed. *[S.l.]*: Wiley, 2010.

OGATA, Katsuhiko. **Modern Control Engineering**. 5th. ed. Upper Saddle River, NJ: Prentice Hall, 2010.

OLIPHANT, Travis E.; CONTRIBUTORS. **NumPy: Fundamental package for scientific computing in Python (Version 1.26.4)**. *[S.l.]*, 2024. Available at: https://numpy.org. Accessed on: Jun. 9, 2025.

Open MPI Project. **Open MPI: Open Source High Performance Message Passing Library (Version 4.1.2)**. *[S.l.]*, 2021. Available at: https://www.open-mpi.org. Accessed on: Jun. 9, 2025.

Oracle Corporation. **MySQL: Open-Source Relational Database Management System**. *[S.l.]*, 2024. Available at: https://www.mysql.com. Accessed on: Jun. 9, 2025.

PEI, Yan; BISWAS, Swarnendu; FUSSELL, Donald S.; PINGALI, Keshav. An elementary introduction to Kalman filtering. **Communications of the ACM**, v. 62, n. 11, p. 122–133, 2019.

PICHE, Stephen; SAYYAR-RODSARI, Bijan; JOHNSON, Doug; GERULES, Mark. Nonlinear model predictive control using neural networks. **IEEE Control Systems Magazine**, v. 20, n. 3, p. 53–62, 2000.

PING, Yu; ZHANG, Yanlong; DU, Xianjun; QIANG, Minghui. An application of the CAN-bus based multi-axis servo control system. **International Journal of Smart Home and Sustainable Technology**, v. 17, n. 13, 2016.

PINKUS, Allan. Weierstrass and approximation theory. **Journal of Approximation Theory**, v. 107, n. 1, p. 1–66, 2000.

Python Software Foundation. **Python 3.13 Documentation**. *[S.l.]*, 2024. Available at: https://docs.python.org/3.13/. Accessed on: Jun. 9, 2025.

QIN, S. Joe; BADGWELL, Thomas A. An overview of industrial model predictive control technology. **AIChE Symposium Series**, v. 93, n. 316, p. 232–256, 1997.

QIN, S. Joe; BADGWELL, Thomas A. A survey of industrial model predictive control technology. **Control Engineering Practice**, v. 11, n. 7, p. 733–764, 2003.

RAO, Singiresu S. **Mechanical Vibrations**. 5th. ed. Singapore: Prentice Hall, 2011.

REBLE, Marcus. **Model Predictive Control for Nonlinear Continuous-Time Systems with and without Time-Delays**. *[S.l.]*: Logos Verlag, 2013.

Redis Ltd. **Redis 7.4 Documentation**. *[S.l.]*, 2024. Available at: https://redis.io/docs. Accessed on: Jun. 9, 2025.

Royal Society (London). **Proceedings of the Royal Society of London**. London: Royal Society, 1869. v. 17.

SAWANT, Shambhuraj; ANAND, Akhil S.; REINHARDT, Dirk; GROS, Sebastien. Learning-based MPC from big data using reinforcement learning. **arXiv**, 2023. Preprint. Available at: https://arxiv.org/abs/2301.01667. Accessed on: Jun. 9, 2025.

SINHA, Rohan; HARRISON, James; RICHARDS, Spencer M.; PAVONE, Marco. Adaptive robust model predictive control via uncertainty cancellation. **arXiv**, 2022. Preprint. Available at: https://arxiv.org/abs/2212.01371. Accessed on: Jun. 9, 2025.

SOUZA, Darielson A. De; BATISTA, Josias G.; VASCONCELOS, Felipe J. S.; REIS, Laurinda L. N. Dos; MACHADO, Gabriel F.; COSTA, Jonatha R.; JÚNIOR, José N. N.; SILVA, José L. N.; RIOS, Clauson S. N.; JÚNIOR, Antônio B. S. Identification by recursive least squares with kalman filter (RLS-KF) applied to a robotic manipulator. **IEEE Access**, v. 9, p. 63779–63789, 2021.

TEAM, The pandas Development. **pandas: Data analysis and manipulation tool (Version 2.2.2)**. *[S.l.]*, 2024. Available at: https://pandas.pydata.org. Accessed on: Jun. 9, 2025.

TIGA, Amira; GHORBEL, Chekib; BRAIEK, Naceur Benhadj *et al.* Nonlinear/linear switched control of inverted pendulum system: Stability analysis and real-time implementation. **Mathematical Problems in Engineering**, v. 2019, 2019.

TURCHETTA, Matteo. **Safety and Robustness in Reinforcement Learning**. 2021. (PhD Thesis) — ETH Zurich, Zurich, 2021.

University of Michigan. **Control Tutorials for MATLAB and Simulink – Inverted Pendulum: State-Space Methods for Controller Design**. 2025. Available at: https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace. Accessed on: Aug. 7, 2025.

WU, M.-F.; LIN, C.-H.; WONG, D. S.-H.; JANG, S.-S.; TSENG, S.-T. Performance analysis of EWMA controllers subject to metrology delay. **IEEE Transactions on Semiconductor Manufacturing**, v. 21, n. 3, p. 494–503, 2008.

YILDIRAN, Uğur. Adaptive control of an inverted pendulum by a reinforcement learning based LQR method. **Sakarya University Journal of Science**, v. 27, n. 6, p. 1311–1321, 2023.

ZANON, Mario; GROS, Sébastien. Safe reinforcement learning using robust MPC. **IEEE Transactions on Automatic Control**, v. 66, n. 8, p. 3638–3652, 2020.