Federal University of Campina Grande
Center for Electrical Engineering and Informatics
Graduate Program in Electrical Engineering

Feedback control of an aeropendulum based on a data-driven dynamic model

Arthur Dimitri Brito Oliveira

Campina Grande, Paraíba, Brazil

Arthur Dimitri Brito Oliveira

Feedback control of an aeropendulum based on a data-driven dynamic model

Master's Thesis submitted to the Coordination of the Electrical Engineering Graduate Program of the Federal University of Campina Grande - Campina Grande Campus - relative to the Information Processing specialization, as part of the necessary requirements to obtain the degree of Master of Science in Electrical Engineering.

Antonio Marcus Nogueira Lima, Dr. - UFCG:
Advisor

Rafael Bezerra Correia Lima, D.Sc. - UFCG:
Advisor

Campina Grande, Paraíba, Brazil, 26th October 2023

O48f Oliveira, Arthur Dimitri Brito.

Feedback control of an aeropendulum based on a data-driven dynamic model / Arthur Dimitri Brito Oliveira – Campina Grande, 2023.

91 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2023.

"Orientação: Prof. Dr. Antonio Marcus Nogueira Lima, Prof. Dr. Rafael Bezerra Correia Lima."

Referências.

1. Electrical Engineering. 2. Model Refinement. 3. Data Acquisition. 4. Parameter Estimation. 5. Sparse Identification. 6. Feedback Control. 7. Feedback Linearization. 8. Cascade Control. I. Lima, Antonio Marcus Nogueira. II. Lima, Rafael Bezerra Correia. III. Título.

CDU 621.3:004(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS GONÇALVES CRB-15/093

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

ATA DA DEFESA PARA CONCESSÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA, REALIZADA EM 20 DE SETEMBRO DE 2023 **(Nº756)**

CANDIDATO: **ARTHUR DIMITRI BRITO OLIVEIRA**. COMISSÃO EXAMINADORA: ALEXANDRE CUNHA OLIVEIRA, D.Sc., UFCG, Presidente da Comissão e Examinador interno, ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG, RAFAEL BEZERRA CORREIA LIMA, D.Sc., UFCG, Orientadores, SAULO OLIVEIRA DORNELLAS LUIZ, Dr, UFPE, Examinador externo. TÍTULO DA DISSERTAÇÃO: **Feedback control of an aeropendulum based on a data-driven dynamic model**. HORA DE INÍCIO: **08h00** – LOCAL: **Sala Virtual, conforme Art. 5º da PORTARIA SEI Nº 01/PRPG/UFCG/GPR, DE 09 DE MAIO DE 2022**. Em sessão pública, após exposição de cerca de 45 minutos, o candidato foi arguido oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua dissertação, obtendo conceito APROVADO. Face à aprovação, declara o presidente da Comissão, achar-se o examinando, legalmente habilitado a receber o Grau de Mestre em Engenharia Elétrica, cabendo a Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o mesmo faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, Filipe Emmanuel Porfírio Correia, e os membros da Comissão Examinadora presentes. Campina Grande, 20 de Setembro de 2023

Filipe Emmanuel Porfírio Correia
Secretário

ALEXANDRE CUNHA OLIVEIRA, D.Sc., UFCG
Presidente da Comissão e Examinador interno

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador

RAFAEL BEZERRA CORREIA LIMA, D.Sc., UFCG
Orientador

SAULO OLIVEIRA DORNELLAS LUIZ, Dr, UFPE
Examinador externo

ARTHUR DIMITRI BRITO OLIVEIRA
Candidato

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Dissertação de Mestrado do candidato **ARTHUR DIMITRI BRITO OLIVEIRA**, assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da dissertação e tomaram conhecimento do teor deste documento.

Referência: Processo nº 23096.072684/2023-20 SEI nº 3817180

To my family. My safe haven even when everything around me falls apart.

Thanks

I thank God for the health and grace to face these days. I thank my family for their unconditional support, incorruptible values instilled in me, and encouragement to prioritize my studies and finish this stage. To my advisors, Antonio Marcus and Rafael Lima, my gratitude for their continuous presence and advice and for motivating me even when I was discouraged. To my friends, thank you for making my days lighter and understanding my absences. I would also like to thank CAPES for the financial support and COPELE-DEE-UFCG for their administrative support during my master's degree.

Abstract

This work aims to refine the aeropendulum model by improving the thrust force description and data acquisition, looking for a better closed-loop response. We propose a low-level acquisition scheme that captures previously neglected electrical quantities using an Arduino Mega. Using the parameters estimated, we develop a simulation model and apply state-of-the-art data-driven identification algorithms to refine the aerodynamic behavior of the plant. We also assess different control approaches with gradual levels of refinement regarding the actuator. Concerning the model refinement, the predicted dynamics for simulated data using the sparse identification algorithm exhibited an MSE (Mean Squared Error) on the order of 10×10^{-10} regarding the dynamics compared to the test set in simulated examples. The system performance in closed-loop indicates that the feedback linearization strategy is the most suitable approach due to the almost constant thrust generation and linear behavior.

Keywords: model refinement, data acquisition, parameter estimation, sparse identification, feedback control, feedback linearization, cascade control.

Resumo

Este trabalho tem como objetivo aperfeiçoar o modelo do aeropêndulo, melhorando a descrição da força de propulsão e a aquisição de dados e o desempenho em malha-fechada. Foi proposto um esquema de aquisição de baixo nível usando um Arduino Mega que captura dados relativos ao atuador previamente negligenciadas. Usando os parâmetros estimados, desenvolveu-se um modelo de simulação e aplicou-se algoritmos de identificação estado-da-arte para refinar o comportamento aerodinâmico da planta. Também avaliou-se diferentes abordagens de controle por realimentação com níveis graduais de refinamento. No que diz respeito ao refinamento do modelo, a dinâmica prevista para os dados simulados utilizando o algoritmo de identificação esparsa apresentou um MSE (Mean Squared Error) da ordem de 1×10^{-10} comparado ao conjunto de teste em exemplos simulados. O desempenho do sistema em malha fechada indica que a estratégia de linearização por realimentação é a abordagem mais adequada devido à quase constante força de propulsão e comportamento linear da planta.

Palavras-chave: refinamento de modelos, aquisição de dados, estimação de parâmetros, identificação esparsa, controle por realimentação, linearização por realimentação, controle em cascata.

Summary

List of Abbreviations, Symbols, Abbreviation, and Acronyms

Abbreviations

MPC: Model Predictive Control	8
PSO: Particle Swarm Optimization	5
TOV1: Timer/Counter1 Overflow Flag	35–37

Initialisms

NARMAX: Nonlinear Autoregressive Models with Moving Avg. and Exog. Input	7
SINDy: Sparse Identification of Nonlinear Dynamics	8
SINDyC: Sparse Identification of Nonlinear Dynamics with Control	8
WSINDy: Weak Sparse Identification of Nonlinear Dynamics with Control	9

List of Tables

List of Figures

Chapter 1

Introduction

This chapter focuses on the problem contextualization, the necessary control design process revision of the aeropendulum experimental platform, the need for complete data acquisition of the system under study, and the model refinement.

1.1 Contextualization

Modeling is a vital aspect of science and engineering. By building a model, one focuses on explaining the relationship between variables. Regarding model-based control design processes, one focuses on building a model of a system, simulating the plant and controller, validating the model, and then deploying the controller. The fundamental step in this process is obtaining a controlled plant's mathematical model.

Amongst the various existing modelings, the mathematical ones can describe dynamic systems, for instance, through differential equations. The objective is to obtain a suitable description of the system's dynamics. The discovery of analytical models is possible through physical or data-driven modeling. Physical modeling uses first principles to model components and requires a good understanding of the system behavior. When there are limitations in determining models from first principles due to complexity or lack of accuracy in describing the plant's behavior, data-driven modeling can provide an appropriate system representation.

The system identification theory comprises a collection of data-driven methods to estimate a given model's parameters through input stimulation and output response. The system identification paradigm shifted from exact to approximate modeling throughout the years. The classical system identification theory assumes that the system's dynamics are previously established. From the approximate modeling standpoint, identifying a system would not necessarily mean finding its exact mathematical model but rather determining what parameters are indispensable from a control-design perspective. Although commonly used for linear dynamics, nonlinear systems represent a challenge for traditional system identification.

An aeropendulum is a nonlinear system commonly used in a didactic context as an example process for other propelling systems, such as drones. Electrical, mechanical, and aerodynamic parts comprise the plant. The actuator of the system is a DC motor, which produces the electromagnetic torque for the propulsion system. Due to the absence of sensors for the electrical quantities, the dynamics between the electrical input and the thrust force are treated as a black box. This lack of modeling might lead to mismatches between the dynamic behavior of the plant and simulated systems. Consequently, there may be closed-loop and open-loop steady-state and transient errors due to the incomplete modeling of the system.

In the literature, most works model the aeropendulum's mechanical dynamics but have deficiencies regarding the actuator's representation, which severely impacts the control strategies. By studying the problem formulation and identifying the possible drawbacks, we can model input-output relationships for the actuator that improve the system's representation. The project's revision includes taming the black box modeling using different refinement levels and improving the collection of electrical quantities concerning the actuator.

The electrodynamic conversion and hidden dynamics during the experimental platform operation could benefit from additional improvements in the data acquisition scheme. The hidden dynamics are ascribed to the unexplicit relationship between the electrical excitation and the thrust force produced but a chain of transductions. Holistic monitoring of the plant leaves space to apply the Sparse Identification of Nonlinear Dynamics (SINDy) to refine the model and obtain physically explainable governing equations from measured data. Given that it explores the connection between input and output variables, we can use the SINDy algorithms to leverage the aerodynamic effects modeling where the traditional modeling cannot improve its description.

1.2 Objectives

This work aims to control an aeropendulum plant using refined data-driven models obtained through the SINDy algorithm and traditional modeling. We run through previous works' projects, aiming to diagnose and fix the discrepancy between the time-domain requirements used in the controller design phase and the closed-loop performance. Pursuing a better description of the system, we want to reconsider some of the hypotheses assumed for the electrical subsystem and its coupling to the mechanical subsystem. The objective is to gradually refine the actuator's description and evaluate different control approaches, exploring data-driven system identification procedures.

1.2.1 Specific objectives

In order to accomplish the general objectives, this work enumerates the following specific objectives:

- Study the SINDy and Weak SINDy mathematical formulations.
- Replicate the frameworks and apply the identification to a forced system.
- Develop signal acquisition schemes for the plant, document and master its hardware and software aspects.
- Propose different levels of refinement in modeling the electrical-aerodynamic behavior using the SINDy algorithm and traditional estimation approaches.
- Improve the electrical characterization of the actuator and develop a simulation model to gain confidence in future design steps.
- Explore noise mitigation strategies and pre-processing steps that could help improve the data quality.

- Establish time-domain requirements, refine the thrust force description in simulation and the experimental platform, and compare the performance in closed-loop with the temporal requirements.

1.3 Structure

This work follows this structure: Chapter 2 has a bibliographical revision of the most relevant works. Chapter 3 contains a mathematical description of the plant and the principal methods used in this work. In Chapter 4, the acquisition schemes and the instrumentation improvements are made. Chapter 5 documents the advances in using the SINDy and WSINDy algorithms in the context of the aeropendulum. In Chapters 5 and 6, respectively, the control strategies' results and final considerations are outlined.

Chapter 2

Literature Review

This chapter focuses on the literature review. First, there is an outline of the classical system identification theory and its limitations regarding nonlinear dynamics. Then, there is a discussion about the state-of-the-art sparse identification methods. Later, in the end, we detail the limitations of previous works using the aeropendulum plant, and the intent of this work is examined after the discussion.

2.1 Background on previous works with the aeropendulum experimental platform

An aeropendulum is a nonlinear dynamic model commonly used in graduate and undergraduate electrical engineering courses. It has a practical appeal, useful in studying control systems-related topics. Previous works have focused on the construction [1] and validation [2] of the proposed plant, taking as the starting point a project from the University of Arizona [3].

Two recent works have already focused on the control and parameter estimation of the model. [4] used the physical modeling of the plant and determined the parameters using the PSO (Particle Swarm Optimization) algorithm applied to step response data of the aeropendulum. Lucena, Luiz and Lima[5] also used first-principles modeling and applied a nonlinear optimization algorithm for estimating the parameters. Nevertheless, they assume some hypotheses and make some simplifications that might considerably affect the desired closed-loop performance.

What is common to all works is that they consider steady-state simplifications about the thrust force produced by the propeller-motor set. Enikov and Campa[3] and Barros and Lima[4] do not model the actuator, thus neglecting the electrical and aerodynamic interactions. Lucena, Luiz and Lima[5] attempts to describe a direct relationship between the input duty cycle and the thrust but lacks a better description of the electrical-aerodynamic information of the motor propeller set. Additionally, they consider that one can infer the thrust force from the steady state angular position. Consequently, these abstractions result in unmodeled dynamics between PWM actuation and the thrust force. Therefore, it is necessary to consider the chain of events that involves the actuator feed to the generated propulsion force.

The previously mentioned works have refined and contributed to the feedback control strategies but made abstractions regarding the actuator dynamics due to instrumentation limitations. If one cannot measure the armature current, the back-EMF (electromotive force) effects are not modeled, impacting the rotor's speed estimation. This work aims to describe the system's hidden dynamics better, contributing to measuring the electrical quantities, describing the thrust force, and discovering electrical and aerodynamic parameters. We want to explore different control strategies and evaluate the results regarding the designed temporal requirements. The expectation is that the ensemble of the neglected modeling parts might describe the system more appropriately and result in better performance of feedback controllers.

2.2 Background on system identification

Model-based control approaches rely on appropriate modeling of the system in such a way that the closed-loop performance meets performance criteria [6]. Likewise, obtaining a plant model that sufficiently generalizes its behavior is necessary. In light of this, the system identification theory is essential in modeling. It comprises mathematical tools and algorithms for obtaining dynamic models from measurement data.

The classical system identification approach assumes that the system's dynamics are already established through a set of poles and zeros [7]. The modeling effort focuses on representing the noise that additively contaminates the measurements. The objective of prefixing input and output terms is to find the most suitable parametric vector by minimizing or maximizing an objective function.

Most system identification techniques, such as subspace identification [8] and prediction-error method [7], are suitable for representing linear systems. Nevertheless, nonlinear dynamics characterize the vast majority of real-world systems. If the dynamical equations are nonlinear, linearization techniques can be used to apply linear state-space control theory [9]. However, it implies challenges when moving the system through different operating points and representing unknown dynamics.

2.3 Background on data-driven nonlinear dynamical representations

Moving from linear identification tools to nonlinear identification methods also implies adaptations in the experiment design, model selection, and determining appropriate cost functions. In this context, NARMAX (Nonlinear Autoregressive Models with Moving Average and Exogenous Input) models were introduced [10] as transparent models that give insights into the system. Compared to linear models, nonlinear ones are represented in a complex high dimensional space and are harder to characterize [11].

By admitting that a NARMAX model can explain the data, any model that belongs to the NARMAX class might be suitable to represent the dynamical behavior. Nonetheless, the main challenge encountered is selecting the appropriate model structure since it is necessary to determine which regressors best represent the system.

With neural networks and deep learning, black-box nonlinear data-driven mapping for complex systems became simpler. Regarding time-series data, it is possible to use different architectures to exploit the relationship between inputs and outputs [12] [13]. While there is no need to make previous assumptions about the system's structure, which might be sufficient for the required type of characterization, the resulting deep learning models lack physical explainability and require extensive data sets [14]. One of the major drawbacks of black box modeling is that the focus relies only on the input-output relationship, extinguishing the possibility of describing better specific parts of the system.

From a model-based control perspective, the model should be accurate and capable of generalizing and parsimonious. In model predictive control (MPC) strategies, a plant representation predicts the system's future behavior and computes the optimal control signal. At each sampling time, an optimization problem is solved. Consequently, there is a need for faster simulations and minimal forms to describe the dynamics [15], which are also known as reduced-order models.

2.3.1 Parsimonious data-driven modelling

The Sparse Identification of Nonlinear Dynamics (SINDy) method is an alternative to not neglect the physical interpretation of the dynamics obtained from measurement data [16]. It bypasses the troublesome combinatorial search for all possible structures, extracting a few relevant terms from the library of candidate functions [17]. Initially proposed for non-forced systems, it benefits from symbolic regression to form a linear combination of candidate functions to represent the data [18].

The comparison between the SINDy method and the parametric system identification theory is inevitable since they are classified as data-driven approaches and lead to a least-squares problem. Nonetheless, they yield to this problem with different premises and models. The regressor vector used to describe the classical system identification theory plant is fully parametric. The library function in the SINDy method is not parametric and acts as a weighting factor to the set of candidate functions. The SINDy method tries to find a sparse model, described in terms of ODEs, with few prior hypotheses.

External forcing can be included in the SINDy algorithm, resulting in the SINDyC (SINDy with Control) formulation [19]. With this inclusion, one of its significant applications is incorporating the SINDyC framework to model predictive control (MPC) schemes. Most MPC applications require accurate model updating to determine the optimal real-time control inputs. In this matter, some works have shown superior performance of SINDYc against other deep learning approaches [20] regarding execution time.

Noise sensitivity is a problematic feature of the SINDy algorithm. Presently, changes in the sparsity-promoting algorithms have been made by focusing on improving noise robustness. The reweighted l1-regularized least-squares solver leverages the accuracy and robustness of the algorithm in tandem with noise mitigation [21]. The weak formulation of differential equations, called Weak SINDy (WSINDy), is also an alternative to replace pointwise derivative approximations and to cope with noisy measurements [22].

The original SINDy algorithm and its variations have been applied to various systems. From reduced-order models in aerospace engineering [23] to robotic applications [24]. In most of them, problems such as noisy measurements, derivatives computation, and design of experiments should be addressed. Regarding noise mitigation, Chartrand[25] proposed some enlightenment in approximating pointwise derivatives using total variation regularized differentiation. Nevertheless, using the SINDy algorithm in these real applications leads to unmodeled dynamics and a lack of physical explainability. The absence of retrieved explainable models is mainly due to the necessity of computing approximate derivatives and the distortion of the original signal, which naturally leads to the use of the WSINDy.

Minor parameter differences between models and real systems can cause deviations in the exact dynamics. Identifying the model mismatch is often challenging, especially when physical explainability is required. The Discrepancy SINDy [26] framework takes advantage of the previous SINDy formulation, leveraging the previous knowledge about the system and aiming for a more accurate system modeling. Nonetheless, it assumes that all state derivatives can be measured, which can lead to dynamic inaccuracies when numerically computed due to the bias-variance trade-off. Therefore, addressing the discussion of an integral version of the algorithm is essential, particularly when considering the application of the algorithm to determine parameter refinement for the aeropendulum system.

2.4 Final considerations

This chapter revisited the contributions and advancements in the sparse identification of nonlinear dynamics and control strategies concerning the aeropendulum platform. It described the limitations of the SINDy algorithm and the challenges regarding modeling the aeropendulum system's hidden dynamics. The next chapter will detail the mathematical formulation of the models and algorithms.

Chapter 3

Theoretical background

This chapter describes the dynamical system used in this work and the mathematical formulation of sparse identification of nonlinear dynamics algorithms. Furthermore, the challenges of computing derivatives on noisy data are outlined and addressed.

3.1 The plant

The Figure 3.1 illustrates the free-body diagram of the system used in this work. The rotating rod with length L connects the bearing axis to the DC motor. The motor is connected to the drive gear, while the driven gear is attached to a propeller. A thrust force f opposes the force weight component $mg\sin(\psi)$. This thrust is proportional to the square of the speed of the propeller. The torque fL leads to the angular displacement concerning the vertical axis.

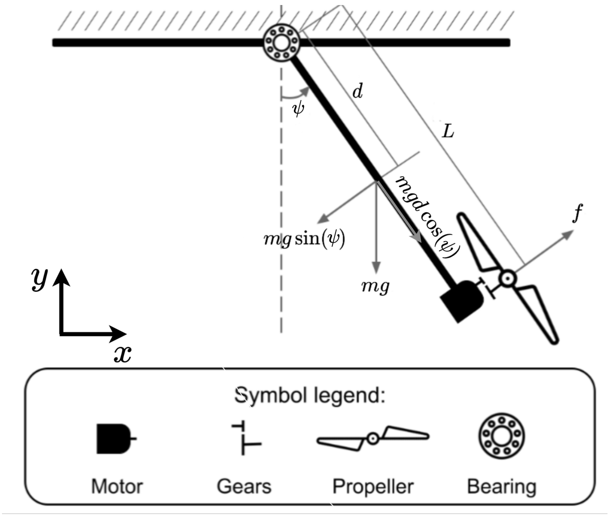
By employing Newton's second law of motion, one can arrive at the differential equation that describes the pendulum's rotational dynamics. By defining Ω as the aeropendulum angular speed, the state-space representation is expressed as

$$\frac{d\psi}{dt} = \Omega, \tag{3.1}$$

$$\frac{d\Omega}{dt} = -\frac{mgd}{J}\sin(\psi) - \frac{C}{J}\Omega + \frac{L}{J}f, \tag{3.2}$$

where J is the inertia moment of the bar, c is the viscous friction related to the rotating

Figure 3.1: Free body diagram of the aeropendulum system and its components. The y and x axis denote the coordinate reference system. The bearing is fixed to the vertical support.



Source: Lucena, Luiz and Lima[5].

axis, and d is the distance from the pivot point to the center of mass of the set.

The thrust force is given by

$$f = K_q \omega_2^2, \tag{3.3}$$

where ω_2 is the speed of the propeller, and K_q is the thrust coefficient [27]. The relationship of ω_1 and ω_2 is expressed by

$$\omega_2 = \frac{N_2}{N_1}\omega_1, \tag{3.4}$$

where $N_2 : N_1$ is the gearbox ratio.

In order to give a better description of the thrust force, one should take into account the electrical aspects of the plant. The equivalent circuit of the DC motor's armature is composed of a series association of its voltage source V_a, the armature resistance R_a and inductance L_a and the armature excitation $e_a = K_\omega \omega_1$. K_ω stands for the back-EMF constant, and the angular speed on the rotor axis is ω_1. The differential equation for the armature current i_a is given by

$$\frac{di_a}{dt} = -\frac{R_a}{L_a}i_a - \frac{K_\omega}{L_a}\omega_1 + \frac{1}{L_a}v_a. \tag{3.5}$$

Considering that the armature inductance has a negligible value, the angular speed ω_1

can be rewritten in terms of the armature current and the input voltage as

$$\omega_1 = \frac{1}{K_\omega} v_a - \frac{R_a}{K_\omega} i_a. \tag{3.6}$$

Because there was no sensor to measure the armature current in the experimental set-up, Lucena, Luiz and Lima[5] uses (3.5) to omit i_a from the expression of the thrust force. The estimated thrust in steady-state can be written as a function of the duty cycle of the input voltage by

$$f_T(u_{ss}) = K_0 \left(2b_0^2 + 4a_0 \frac{K_i}{R_a} K_f u_{ss} - 2b_0 \sqrt{{b_0}^2 + 4a_0 \frac{K_i}{R_a} K_f u_{ss}} \right). \tag{3.7}$$

Considering that the armature current can be measured, the objective is to include it in the expression of the thrust force. The propeller's transient response is assumed to be faster than the pendulum rod's speed, and the propeller's moment of inertia is also neglected. Hence, by knowing the expression given by (3.4), the thrust force is modeled as a proportional relationship between ω_2 and K_T by

$$f = K_q \omega_2^2 = K_q \left(\frac{N_1}{N_2} \omega_1 \right)^2. \tag{3.8}$$

In this work, we rewrite (3.9) in terms of (3.5), the equivalence is given by

$$f = K_q \left(\frac{N_1}{N_2 K_\omega} \right)^2 v_a^2 - 2R_a K_q \left(\frac{N_1}{N_2 K_\omega} \right)^2 v_a i_a + K_q \left(\frac{N_1}{N_2} \frac{R_a}{K_\omega} \right)^2 i_a^2, \tag{3.9}$$

and then (3.2) can be rewritten as

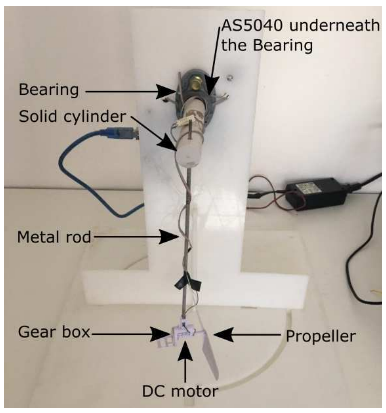
$$\frac{d\Omega}{dt} = -\frac{mgd}{J} \sin(\psi) - \frac{C}{J}\Omega + \frac{L}{J} K_q \left[\left(\frac{N_1}{N_2 K_\omega} \right)^2 v_a^2 - 2R_a \left(\frac{N_1}{N_2 K_\omega} \right)^2 v_a i_a + \left(\frac{N_1}{N_2} \frac{R_a}{K_\omega} \right)^2 i_a^2 \right]. \tag{3.10}$$

3.1.1 The experimental platform

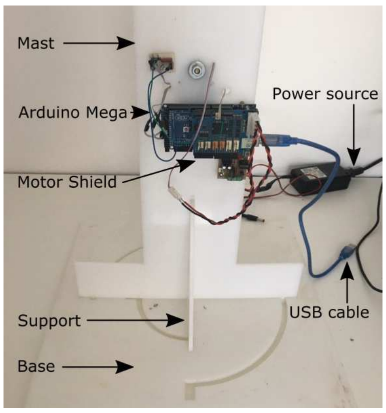
Figure 3.2 depicts the experimental setup. The supports for the pendulum (the base, support, and mast) are made of polystyrene. The propulsion system comprises a DC motor, a motor shield (dual full-bridge driver), a gearbox, and a propeller. An AS5040 magnetic rotary sensor measures the angular displacement caused by the excitation of the propulsion system. The Arduino Mega 2560 generates both the actuation and the clock and chip selection signals and delivers the angular position through USB communication to the central PC station.

Figure 3.2: Pictures of the experimental platform used in this work. In (a), there is a front photograph of the plant and its main mechanical parts. The picture (b) shows the back side of the setup and its hardware components.

(a)



(b)

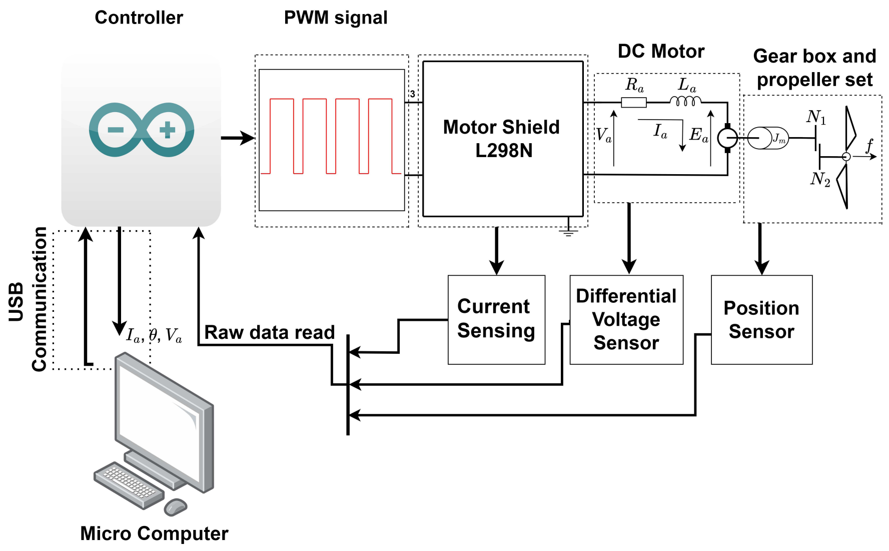


Source: Lucena, Luiz and Lima[5].

The aeropendulum consists of four parts as depicted in Figure 3.3. A PWM signal commands the motor shield's H-bridge and delivers voltage to the motor armature. There is a nonlinear voltage drop across the H-bridge, so the effective armature voltage should be measured. The acquisition routine is written in the C language, and due to hardware limitations of the controller board, the code implemented must use low-level programming. The microcomputer compiles the code and writes it to the Arduino Mega board. The board then acquires the data and sends it through serial communication, dumping it into a text file.

In the next section, we will fundament the basis for the sparse identification algorithms. These algorithms will be useful for determining aerodynamic parameters, especially considering the physical explainability required from the simulation model.

Figure 3.3: Aeropendulum system diagram: electrical, mechanical, and aerodynamic components.



Source: prepared by the author.

3.2 Sparse Identification Algorithms

This section describes the main sparse identification algorithms for nonlinear dynamics: the SINDy, WSINDy, and Discrepancy SINDy frameworks. First, the mathematical formulations are described, followed by the pseudo algorithms for each algorithm and a case study to demonstrate their effectiveness.

3.2.1 SINDy

SINDy, which stands for Sparse Identification of Nonlinear Dynamics, is an algorithm used to identify nonlinear dynamics. With a wide range of applications, from biology to engineering, it assumes that the underlying dynamics can be discovered from input-output data. Hence, it considers that the analytical expressions that describe the system are expressed as a combination of candidate functions.

Mathematical formulation - SINDy

Suppose that a nonlinear system is represented by

$$\frac{d\mathbf{x}(\mathbf{t})}{dt} = \mathbf{F}(\mathbf{x}(\mathbf{t})), \;\; \mathbf{x}(0) = x_0, \tag{3.11}$$

where $\mathbf{x}(\mathbf{t}) \in \mathbb{R}^{m \times n}$ represents the system state variables measured at time different time samples from t_1 to t_m. $\mathbf{F}(\mathbf{x}(\mathbf{t}))$ maps the state variables to the derivatives of the state variables. Assuming that this mapping is unknown, in order to approximate it numerically, a set of measurements of each state variable is given by

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix}. \tag{3.12}$$

One can compose a set of candidate functions to represent $\mathbf{F}(\mathbf{x})$ by

$$\Theta(\mathbf{X}) = \begin{bmatrix} \theta_1(\mathbf{X}) & \theta_2(\mathbf{X}) & \dots & \theta_p(\mathbf{X}) \end{bmatrix}. \tag{3.13}$$

Supposing $\theta_1(\mathbf{X})$ is the second power of the state variables. We construct $\theta_1(\mathbf{X})$ as the multiplication between the state variables, including the variables themselves, in such a way that

$$\Theta(X) = \begin{bmatrix} {x_1}^2(t_1) & x_1(t_1)x_2(t_1) & \dots & {x_3}^2(t_1) & \dots & {x_n}^2(t_1) \\ {x_1}^2(t_2) & x_1(t_2)x_2(t_2) & \dots & {x_3}^2(t_2) & \dots & {x_n}^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ {x_1}^2(t_n) & x_1(t_n)x_2(t_n) & \dots & {x_3}^2(t_n) & \dots & {x_n}^2(t_n) \end{bmatrix}. \tag{3.14}$$

The matrix columns expressed in (3.13) represent candidate functions applied to each measured state variable. We can express the nonlinear mapping $\mathbf{F}(\mathbf{x}(\mathbf{t}))$ as a linear combination of the library of candidate functions evaluated in the sampling intervals. By defining the sparse vectors of coefficients as Ξ, the symbolic regression formulation is then given by

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi^T, \tag{3.15}$$

where $\Xi = [\boldsymbol{\xi}_1 \quad \boldsymbol{\xi}_2 \quad \dots \quad \boldsymbol{\xi}_n]$. Each of the $\boldsymbol{\xi}_k$ terms is a sparse array of coefficients, and determine which of the nonlinear terms are necessary to describe the corresponding dynamics. The matrix of state derivatives $\dot{\mathbf{X}}$ can be numerically computed or measured. The matrix of unknown coefficients is sparse on the assumption that the system dynamics can be expressed in terms of a small subset of the library of candidate functions $\Theta(\mathbf{X})$. The goal is to minimize the loss between the true and approximate dynamics, penalizing large terms of the $\hat{\Xi}$. This minimization can be achieved through LASSO regression by

$$\hat{\Xi} = \arg\min_{\hat{\Xi}} ||\dot{\mathbf{X}} - \Theta(\mathbf{X})\Xi^T||_2 + \alpha||\Xi||_1. \tag{3.16}$$

With the computation of the sparse matrix Ξ, each row of the governing equations can be written in terms of $\dot{\mathbf{x}}_\mathbf{k} = \boldsymbol{f_k}(\mathbf{x}) = \Theta(\mathbf{x}^T)\boldsymbol{\xi}_k$. The original SINDy paper [16] uses the sequential thresholded least squares (STLS), making terms below a certain threshold null. The threshold parameter λ is the hyperparameter that controls the learning process by zeroing the terms at each iteration that are below the defined value.

SINDy with control

One can generalize the SINDy approach to support control inputs, here represented as $\mathbf{u}(\mathbf{t})$ by

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})). \tag{3.17}$$

The new formulation requires generating more candidate functions for the control signal $\Theta(\mathbf{X}, \mathbf{U})$, where $\mathbf{X}$ represents the matrix containing all the state variables, and $\mathbf{U}$ represents the matrix of control signals. (3.18) shows how to represent the derivatives of the state variables using the candidate library function Θ and the sparse matrix $\hat{\Xi}$. In the case of the aeropendulum, considering that both the armature current and voltage can be measured, the library function would include the angular position, velocity, and external forcing terms.

$$\dot{\mathbf{X}} = \Theta(\mathbf{X}, \mathbf{U})\hat{\Xi}^T \tag{3.18}$$

Pseudo algorithm - SINDy

The pseudo algorithm for the SINDy formulation can be seen on Algorithm 1. The preprocessing step includes the derivatives' computation and the library function's construction. Initially, an estimate $\hat{\Xi}$ is computed. While there are terms above a certain value λ, the higher matrix coefficients are then zero, which is applied for each state variable. An improved estimation of the sparse matrix is computed for each of these state variables. When the algorithm has converged, that is, when there are no more terms above the threshold value, the output is a sparse matrix of coefficients that better describe the derivatives of the state variables.

Algorithm 1 The STLs routine to determine the exact state variables' derivatives. It receives as inputs the derivatives $\dot{X}$, the library function $\Theta(X, U)$, and a threshold parameter λ. The output is a sparse matrix Ξ.

```
1:  function STLS(($\dot{\mathbf{X}}$, $\Theta(\mathbf{X}, \mathbf{U})$, $\lambda$, $N$))
2:      $\hat{\Xi} \leftarrow \Theta^{\dagger}\dot{\mathbf{X}}$
3:      while not converged do
4:          $k \leftarrow k + 1$
5:          $I_{smal} = (abs(\Xi < \lambda)$
6:          $\Xi(I_{small}) = 0$
7:          for i = 1:N do
8:              $I_{big} \leftarrow I_{small}(:, i)$
9:              $\Xi(I_{big}, i) \leftarrow \Theta(:, I_{big})^{\dagger}\dot{\mathbf{X}}(:, i)$
10:         end for
11:     end while
12:     return $\Xi$
13: end function
```

3.2.2 Weak SINDy

The WSINDy algorithm (Messenger and Bortz[22]) proposes using the weak form of differential equations towards the noise sensitivity problem. The weak formulation turns a differential equation into an integral equation. It makes the method more robust and less susceptible to errors due to the need for computing derivatives.

Consider that the measurement data is contaminated with noise. Given that the states are $y \in \mathbb{R}^{m\times n}$, with m sampling instants $\mathbf{t}$ from t_1 to t_m, the noisy measurements can be expressed by

$$\mathbf{y}(\mathbf{t}) = \mathbf{x}(\mathbf{t}) + \epsilon(\mathbf{t}), \tag{3.19}$$

where $\mathbf{y}(\mathbf{t})$, $\mathbf{x}(\mathbf{t}) \in \mathbb{R}^{m\times n}$, and $\epsilon(\mathbf{t}) \in \mathbb{R}^{m\times n}$.

For any smooth test function $\phi : \mathbb{R} \to \mathbb{R}$ supported on the interval $(a, b) \subset [0, t_m]$, (3.11) admits the weak formulation, when $0 \le a \le b \le t_m$, given by

$$\phi(b)\mathbf{x}(b) - \phi(a)\mathbf{x}(a) - \int_a^b \phi'(\varsigma)\mathbf{x}(\varsigma)d\varsigma = \int_a^b \mathbf{F}(\mathbf{x}(\varsigma))d\varsigma, \tag{3.20}$$

consisting of the data-driven version of the Galerkin method for solving $\mathbf{F}$.

The function mapping $\hat{\mathbf{F}} : \mathbb{R}^n \to \mathbb{R}^n$ extracted from the noisy measurements is given by

$$\hat{\mathbf{F}}(\mathbf{y}(\mathbf{t})) = \sum_{j=1}^{J} \xi_J \theta_j(\mathbf{y}(\mathbf{t})), \tag{3.21}$$

where the library function $\{\theta_l(\cdot), l = 1, \cdots, J\}$ is a set of candidate basis functions used to represent $\hat{\mathbf{F}}$, an estimate for $\mathbf{F}$, and it is expressed by

$$\Theta(\mathbf{y}(\mathbf{t})) = \begin{bmatrix}\theta_1(\mathbf{y}(t)) & \theta_2(\mathbf{y}(t)) & \ldots & \theta_J(\mathbf{y}(t))\end{bmatrix}. \tag{3.22}$$

where each of the library function terms represents a trigonometric, polynomial function, or a product between each of the terms.

One can include the excitation input when composing the library of candidate functions Θ. With that, the WSINDy formulation supports control inputs and hence can be called WSINDy with control (WSINDyC). In (3.20), when ϕ is non-constant and supported in the interval (a, b), that is $\phi(a) = \phi(b) = 0$, one might define the generalized residual $R(\Xi, \phi)$ in respect to a specific test function by substituting $\hat{\mathbf{F}}$ with a candidate function, and by replacing $\mathbf{x}$ with the noise-contaminated $\mathbf{y}$ as

$$R(\Xi, \phi) = \int_a^b \phi'(\varsigma)\mathbf{y}(\varsigma)d\varsigma + \int_a^b \phi(\varsigma) \sum_{j=1}^{J} \xi_j \theta_j(\mathbf{y}(\varsigma))d\varsigma. \tag{3.23}$$

The proposed method is a data-driven version of the Galerkin method for solving $\hat{\mathbf{F}}$. The discrete-time version of (3.23) can be determined by

$$R(\Xi, \phi_k) := (\mathbf{G}\Xi - \mathbf{b})_k \in \mathbb{R}^{1\times n}. \tag{3.24}$$

The Gram matrix $\mathbf{G}$ and the approximate dynamics $\mathbf{b}$ are determined through the use of the integration matrices $\mathbf{V}$ and $\mathbf{V'}$. They can be determined by

$$\mathbf{V}_k m = \Delta t \phi_k(t_m), \tag{3.25}$$

$$\mathbf{V'}_k m = \Delta t \phi'_k(t_m). \tag{3.26}$$

Hence, (3.24) can be rewritten as

$$R(\Xi, \phi_k) := \mathbf{V}\Theta(\mathbf{y})\Xi - \mathbf{V'y}. \tag{3.27}$$

Defining the covariance matrix as $\Sigma = V'(V')^T$ and using it as a weighting factor, the solution to the generalized least-squares problem can be given by

$$\Xi = \left((\mathbf{G}\Xi - \mathbf{b})^T \Sigma^{-1} (\mathbf{G}\Xi - \mathbf{b}) + \gamma^2 ||\Xi||_2^2\right). \tag{3.28}$$

Different hyperparameters must be tuned, and they control the learning process. K is the number of test functions, p controls the normalization of the polynomials, s is the shift parameter, r_{whm} corresponds to the width-at-half-max parameter. More details about using these parameters in the formulation can be found on [22].

Defining the test function basis

In order to minimize integration errors, a test function space S with Φ_k sufficiently localized is proposed. The space S, which consists of piece-wise polynomials, is given by:

$$\phi(t) = \begin{cases} C(t-a)^p(b-t)^q, & t \in (a,b) \\ 0, & otherwise \end{cases} \tag{3.29}$$

The conditions $a < b$ and $p, q \geq 1$ are satisfied here. The normalization factor C is determined by

$$C = \frac{1}{p^p q^q} \left(\frac{p+q}{b-a} \right)^{p+1}. \tag{3.30}$$

The location of the test functions proposed in (3.29) depends on the strategy adopted toward the problem. It might be a uniform or adaptive grid.

3.2.3 Discrepancy SINDy

The Discrepancy SINDy framework (Kaheman et al.[26]) assumes that the model mismatch can be described in terms of a library of candidate functions. The compensation makes it possible to fix the model mismatch directly from data measurements, providing a better description of the system state variables' derivatives.

Consider that the measurements of a dynamical system are expressed as

$$\boldsymbol{\Upsilon_0}(\mathbf{t}) = \mathbf{F}(\mathbf{X}(\mathbf{t}); \mu) = \dot{\mathbf{X}}, \tag{3.31}$$

where $\mathbf{F}(\mathbf{x}(\mathbf{t}))$ represents the nonlinear mapping between the input and output variables, and μ represents the model's parameters. Suppose that an approximate model is given by

$$\boldsymbol{\Upsilon_m}(\mathbf{t}) = \mathbf{F}_m(\mathbf{X}(\mathbf{t}); \mu_1), \tag{3.32}$$

where μ_1 is the approximate set of parameters. $\mu \neq \mu_1$ and $\mathbf{F}(\mathbf{X}(\mathbf{t}; \mu) \neq \mathbf{F}_m(\mathbf{X}(\mathbf{t}); \mu_1)$.

By constructing a discrepancy model $\delta\boldsymbol{\Upsilon}(\mathbf{t})$, expressed in terms of the measurement data $\mathbf{X}(\mathbf{t})$ and the parameters μ_2, the mismatch is given by

$$\delta\boldsymbol{\Upsilon}(\mathbf{t}) = \mathbf{G}(\mathbf{X}(\mathbf{t}); \mu_2) == \boldsymbol{\Upsilon_0}(\mathbf{t}) - \boldsymbol{\Upsilon_m}(\mathbf{t}). \tag{3.33}$$

The discrepancy of the dynamics is collected and expressed in terms of a matrix

$$\delta\boldsymbol{\Upsilon} = [\delta\boldsymbol{\upsilon_1}(\mathbf{t}) \quad \delta\boldsymbol{\upsilon_2}(\mathbf{t}) \ldots \delta\boldsymbol{\upsilon_n}(\mathbf{t})]^T, \tag{3.34}$$

where $\delta\boldsymbol{\Upsilon} \in \mathbb{R}^{m \times n}$.

We use the SINDy formulation to represent $\mathbf{G}(\mathbf{X}(t); \mu_1)$ as a combination of different candidate functions $\Theta(\mathbf{X})$. Hence, the regression problem is expressed by

$$\delta\boldsymbol{\Upsilon} = \Theta(\mathbf{X})\Xi. \tag{3.35}$$

We solve the least squares problem for Ξ. The computation of the derivatives for the difference in the derivatives might lead to inaccuracies. If one desires not to compute the derivatives, the problem can be adapted to the integral formulation. By integrating both sides of (3.35), we have

$$\int \boldsymbol{\Upsilon} dx = \int \left(\boldsymbol{\Upsilon_0} - \mathbf{F}_m(\mathbf{X}(\mathbf{t}); \mu_1)\right) dt = \mathbf{X}(t) - \gamma(t), \tag{3.36}$$

where $\gamma(t)$ corresponds to the data obtained from the approximate model. The residual for the discrepancy problem is expressed by

$$R(w, \phi) = \int_a^b \left(\phi'(\varsigma)[x(\varsigma) - \gamma(\varsigma)]] + \phi(\varsigma) \left(\sum_{j=1}^{J} \xi_j \theta_j(x(\varsigma)) \right) \right) d\varsigma. \tag{3.37}$$

This way, the minimization problem could be determined through least squares, which is the best modification to match the discrepant model with the reference one.

3.3 Study case

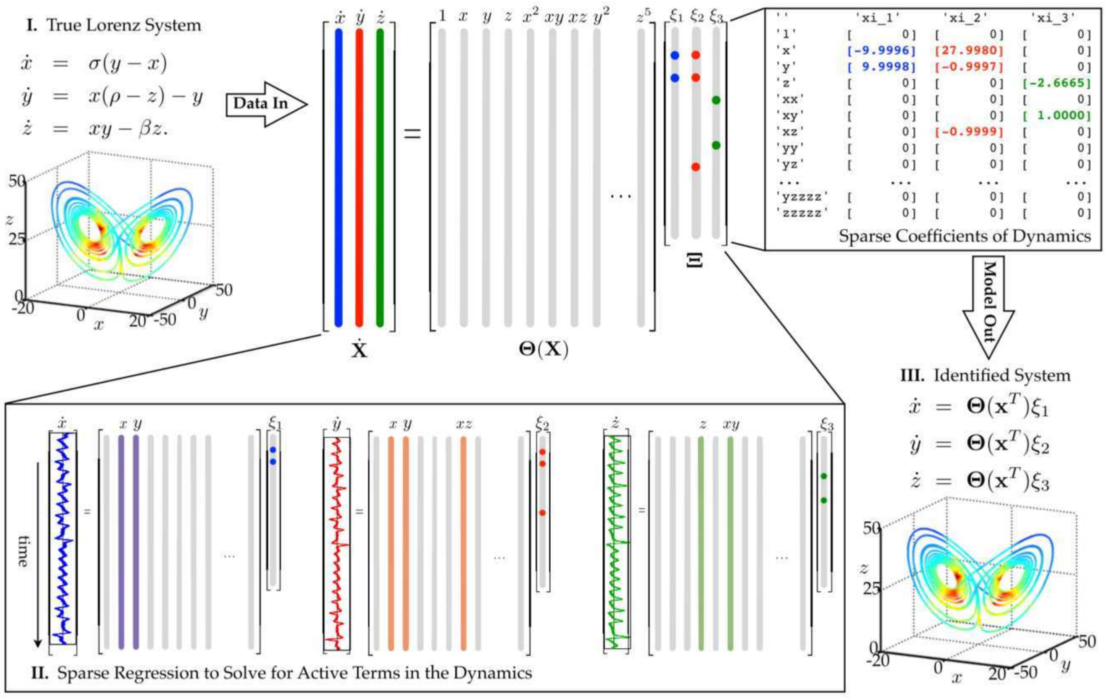
As illustrated in the Figure 3.4, the identification scheme firstly involves data collection of state measurements and derivatives. Subsequently, there is a construction of a library of candidate functions. Finally, a sparse regression identifies the fewest terms on the dynamics that describe the data. For the Lorenz system, we will detail the use of all the sparse promoting algorithms to find the best representation for the system.

3.3.1 Solution using SINDy

As Figure 3.4 illustrates, firstly, the data is generated by the analytical equations that describe the Lorenz attractor. Secondly, the data is used to formulate the regression problem, where the library function consists of the candidate terms based on the data input. Active terms in the dynamics are few and activated by the sparse matrix of coefficients Ξ.

As depicted in the bottom part of the figure, the number of active terms to describe the governing equations is less than three. Lastly, regarding the candidate library function and the sparse matrix, the identified system is used to predict the state variables' derivatives for the test data.

Figure 3.4: Schematic of the SINDy algorithm applied to the Lorenz Equations. The first section represents the data generated from the analytical equations. The second one represents the library function and the sparse matrix that activates the necessary candidate functions. The last corresponds to $\Theta(X)\Xi$, related to the state variables' derivatives.


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
Data In
$\dot{\mathbf{X}}$
$\Theta(\mathbf{X})$
Ξ
Sparse Coefficients of Dynamics
Model Out
III. Identified System
$\dot{x} = \Theta(\mathbf{x}^T)\xi_1$
$\dot{y} = \Theta(\mathbf{x}^T)\xi_2$
$\dot{z} = \Theta(\mathbf{x}^T)\xi_3$
time
II. Sparse Regression to Solve for Active Terms in the Dynamics

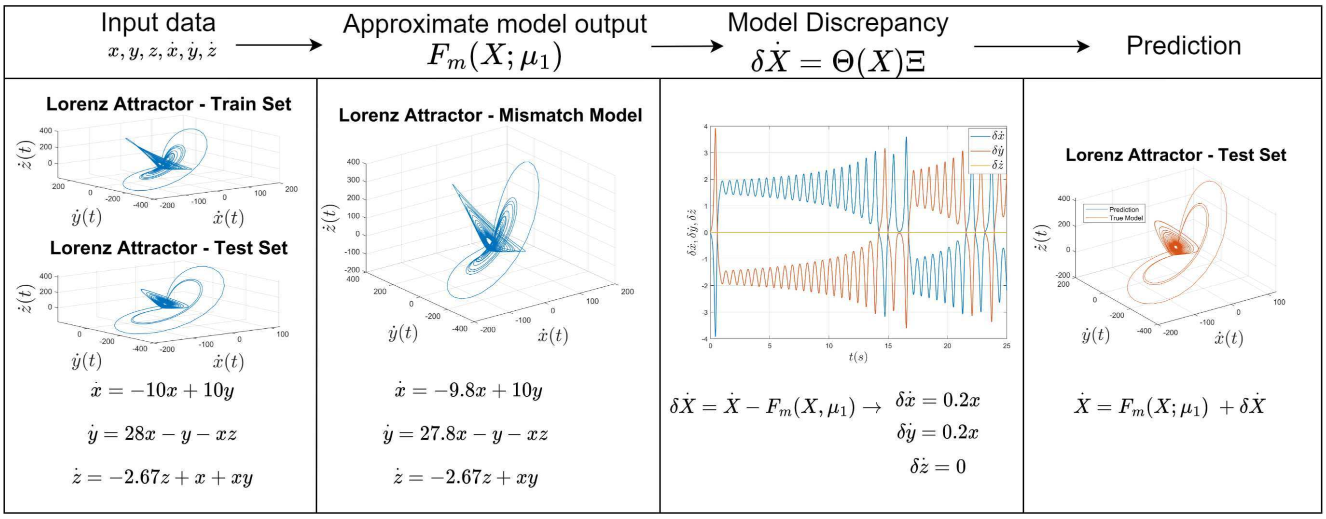
```
''         'xi_1'      'xi_2'      'xi_3'
'1'       [      0]  [      0]  [      0]
'x'       [-9.9996]  [27.9980]  [      0]
'y'       [ 9.9998]  [-0.9997]  [      0]
'z'       [      0]  [      0]  [-2.6665]
'xx'      [      0]  [      0]  [      0]
'xy'      [      0]  [      0]  [ 1.0000]
'xz'      [      0]  [-0.9999]  [      0]
'yy'      [      0]  [      0]  [      0]
'yz'      [      0]  [      0]  [      0]
...          ...        ...        ...
'yzzzz'   [      0]  [      0]  [      0]
'zzzzz'   [      0]  [      0]  [      0]
```

Source: Brunton, Proctor and Kutz[16].

3.3.2 System identification using Discrepancy SINDy

As Figure 3.5 depicts, firstly, the train and test datasets are generated by the reference governing equations using different initial conditions. The mismatching model's output is computed using the same initial conditions of the training set. The discrepancy terms affect the derivatives $\dot{x}$ and $\dot{y}$. Following the data generation, the difference between the dynamics is evaluated, namely $\delta\dot{X}$. The regression problem is formulated and solved for Ξ using the discrepancy data as input. Lastly, the dynamics are predicted for the test set using the approximate model $F_m(X;\mu_1)$ and the correction terms $\delta\dot{X}$.

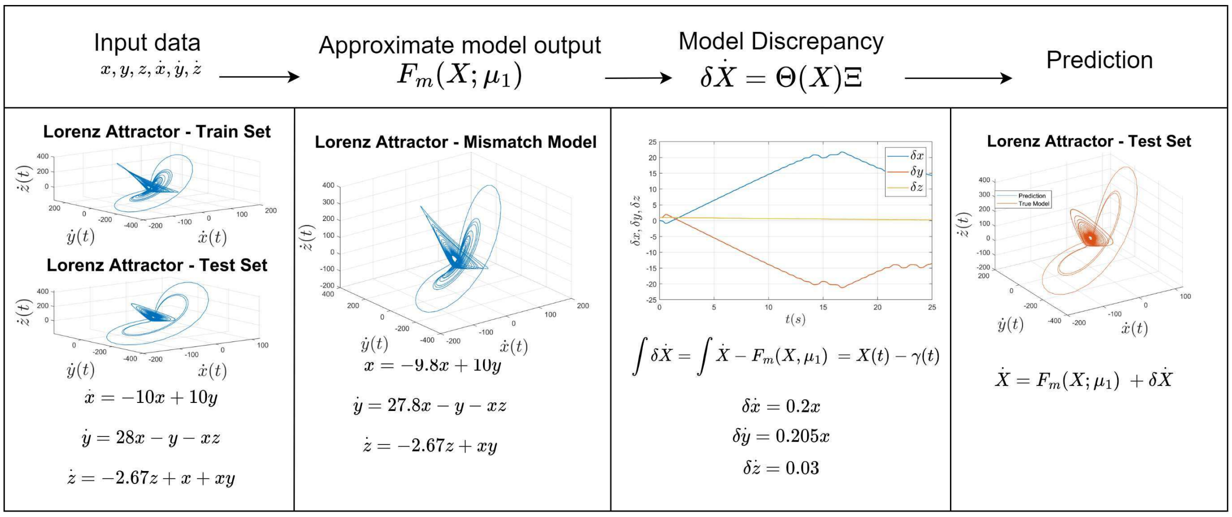
Figure 3.5: Schematic of the Discrepancy SINDy algorithm applied to the Lorenz Equations. The first step is to generate the training and test sets. After that, the approximate model output data is generated, and the regression problem is solved. With the terms that need to be added to the approximate model, the test set dynamics are predicted.


Input data
$x, y, z, \dot{x}, \dot{y}, \dot{z}$
Approximate model output
$F_m(X;\mu_1)$
Model Discrepancy
$\delta\dot{X} = \Theta(X)\Xi$
Prediction
Lorenz Attractor - Train Set
Lorenz Attractor - Test Set
$\dot{x} = -10x + 10y$
$\dot{y} = 28x - y - xz$
$\dot{z} = -2.67z + x + xy$
Lorenz Attractor - Mismatch Model
$\dot{x} = -9.8x + 10y$
$\dot{y} = 27.8x - y - xz$
$\dot{z} = -2.67z + xy$
$\delta\dot{X} = \dot{X} - F_m(X,\mu_1) \rightarrow$
$\delta\dot{x} = 0.2x$
$\delta\dot{y} = 0.2x$
$\delta\dot{z} = 0$
Lorenz Attractor - Test Set
Prediction
True Model
$\dot{X} = F_m(X;\mu_1) + \delta\dot{X}$
$\dot{x}(t)$
$\dot{y}(t)$
$\dot{z}(t)$
$t(s)$

3.3.3 System identification using Discrepancy WSINDy

As Figure 3.6 depicts, firstly, the inputs to the problem are computed using the analytical equations that describe the Lorenz attractor and different initial conditions. The mismatch model that incorrectly describes the derivatives $\dot{x}$ and $\dot{y}$ is used to generate the approximate model output. The model discrepancy is then computed, integrating the difference between the derivatives and generating the approximate difference in the state variables. The minimization problem is solved for Ξ using the approximate difference in the trajectories. One Ξ is computed and is used to predict the dynamics of the test set.

Figure 3.6: Schematic of the WSINDy algorithm applied to the Lorenz Equations. The first step is to generate the training and test sets. After that, the approximate model output data is generated, and the discrepancy between the dynamics is computed. By formulating and solving the regression problem.


Input data
$x, y, z, \dot{x}, \dot{y}, \dot{z}$
Approximate model output
$F_m(X;\mu_1)$
Model Discrepancy
$\delta\dot{X} = \Theta(X)\Xi$
Prediction
Lorenz Attractor - Train Set
Lorenz Attractor - Test Set
$\dot{x} = -10x + 10y$
$\dot{y} = 28x - y - xz$
$\dot{z} = -2.67z + x + xy$
Lorenz Attractor - Mismatch Model
$x = -9.8x + 10y$
$\dot{y} = 27.8x - y - xz$
$\dot{z} = -2.67z + xy$
δx
δy
δz
$t(s)$
$\int \delta\dot{X} = \int \dot{X} - F_m(X,\mu_1) = X(t) - \gamma(t)$
$\delta\dot{x} = 0.2x$
$\delta\dot{y} = 0.205x$
$\delta\dot{z} = 0.03$
Prediction
True Model
$\dot{X} = F_m(X;\mu_1) + \delta\dot{X}$

3.4 Derivatives computation

The state variables' derivatives computation is necessary to formulate the SINDy algorithm, and it is a problematic feature for at least two reasons. First, as previously detailed in subsection 3.1.1, there is only an angular position sensor. Placing an angular velocity sensor would be problematic in the context of the aeropendulum due to the way the rod is coupled to the mast. Even so, the state space formulation requires angular velocity data.

Computing derivatives from measurement data is a common problem in determining dynamical models. Similarly, on the SINDy algorithm, the numerical differentiation is crucial since it composes the left-hand side of (3.15). Regarding the aeropendulum plant, differentiation techniques should be applied to determine the angular velocity even if the integral formulation of differential equations is adopted.

3.4.1 Total Variation Regularized Differentiation - TVRegDiff

Approximating derivatives from noise-contaminated data is troublesome. Due to the noise variance, traditional smoothing techniques are prone to noise amplification [28]. Instead, total variation regularization proposes regularizing the differentiation process itself [25].

The Tikhonov regularization is one of the possible solutions to approximate derivatives from noisy sensor data. Given a function H in $[0, L]$, its derivative is determined as the minimizer derivative of a function h on the same interval by

$$H(u) = \alpha R(u) + DF(Au - h). \tag{3.38}$$

The data fidelity term is $DF(Au - f)$, associated with the discrepancy between the integral of the derivative and the original function f. The first term $\alpha R(u)$ penalizes the irregularity in the derivative of u and controls the balance with the data fidelity term. Rewriting then (3.38), we have:

$$H(u) = \alpha \int_0^L |u'| + \frac{1}{2} \int_0^L |Au - h|^2. \tag{3.39}$$

More noise implies a higher total variation. Hence, controlling the total variation of the derivative allows the approximate signal to capture more features of the data and control its scale of fluctuations. With this formulation of the derivative problem, the main accomplishment is noise suppression.

It is worth mentioning that the choice of the regularization parameter is not straightforward, especially when there is no ground truth in the derivatives. One of the possible ways of determining α is trial and error (Chartrand[25]) through visual inspection of the noise level on the derivatives. Another option is to use a multi-objective optimization framework for automatically choosing the regularization parameter [29]. This approach considers the trade-off between faithfulness and smoothness of the estimated derivatives.

3.5 Final considerations

This chapter described the mathematical model of the experimental platform and detailed the formulations of the SINDy, WSINDy, and Discrepancy SINDy algorithms. Schematics were used to describe the extraction of analytical equations from measurement data and the missing terms for approximate models. The derivative approximation problem was addressed using the Total Variation Regularized Differentiation (TVRegDiff) algorithm. The next chapter will describe the proposed acquisition schemes, the instrumentation improvements, and the refinement of the thrust force description.

Chapter 4

Estimation and Instrumentation Improvements

This chapter details the enhancements made in the instrumentation and data acquisition of the aeropendulum system. Previous studies have primarily concentrated on applying control strategies yet needed a comprehensive depiction of the electrical and aerodynamic aspects. Consequently, they treated the interactions between the actuator's input and the mechanical behavior of the plant as a black box.

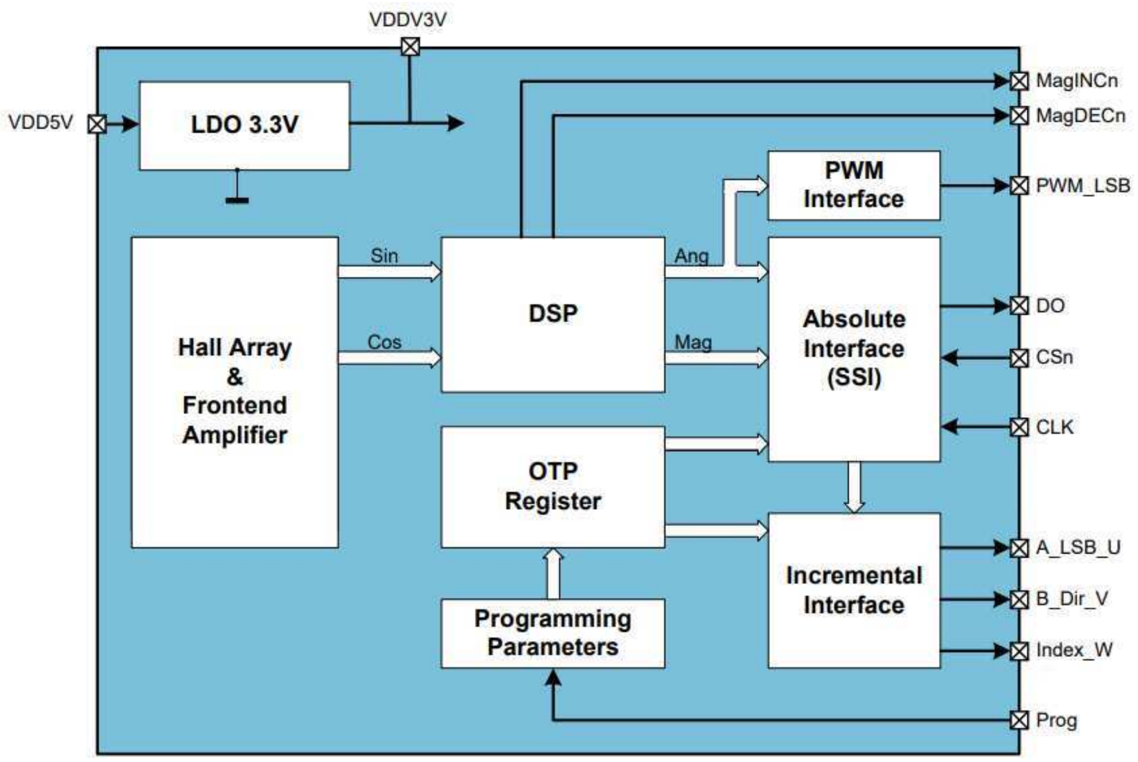
Oliveira, Lima and Lima[30] discusses and addresses the improvements detailed here. Initially, we detail the need for refining the data acquisition methodologies and explore the possible interfaces for capturing angular position data and electrical quantities. Subsequently, we present a comparison and rationale concerning the sensor's interface choice. Following this, we describe the advancements achieved in parameter estimation. In the end, we detail the development of the simulation model.

4.1 Angular position data - acquisition interfaces

For the angular position information, we use the AS5040 magnetic rotary encoder for the angular displacement concerning the vertical axis. As depicted in Figure 4.1, the orientation of the magnetic field is sensed, processed, and results in a 10-bit binary code proportional to the angular displacement. This value can be accessed via a Synchronous Serial Interface

(SSI), the Pulse Width Modulated signal at the PWM output, or the Incremental interface. The focus here is on both the SSI and PWM interfaces.

Figure 4.1: AS5040 sensor block diagram.



Source: AS5040 sensor datasheet [31].

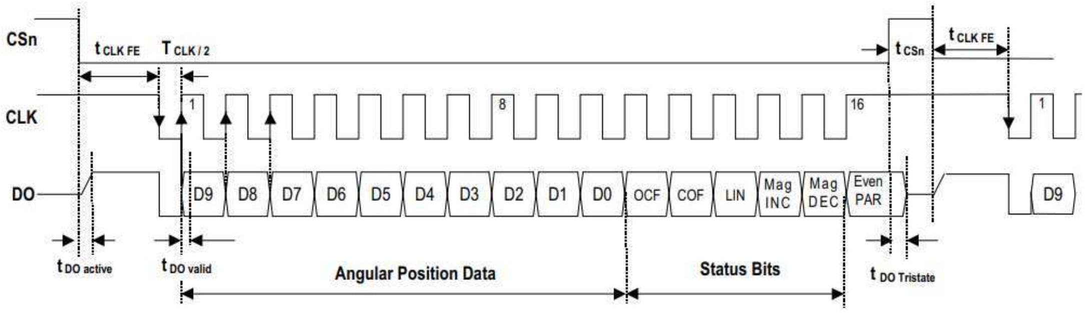
Absolute Interface (SSI)

As shown in Figure 4.2, a sequence of signals must be followed. After the first positive edge of the Clock signal, the data stream begins with the least significant byte. For the 10-bit position data, 22 clock signal pulses are necessary. To generate these signals in hardware, one can rely on the PWM frequency of the microcontroller board. One of the drawbacks is the necessity of 22 pulses to complete the acquisition cycle, which might significantly impact the sampling rate.

PWM Interface

The magnetic sensor also provides a PWM signal, with its duty cycle proportional to the measured angle. The signal frequency is 975,6 Hz, with 10-bit resolution and a minimum

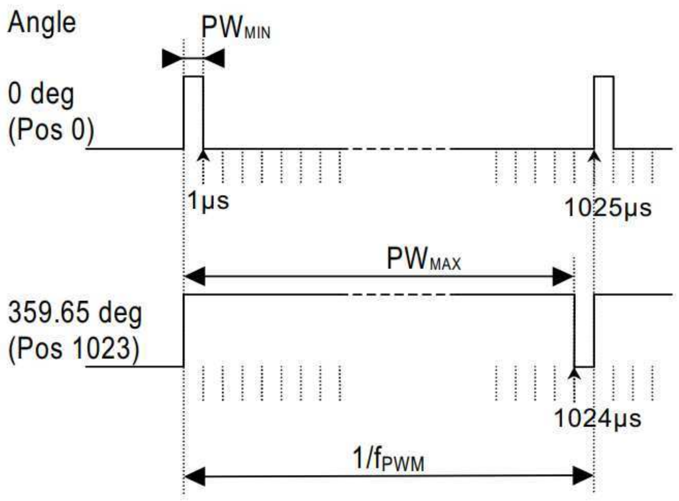
Figure 4.2: AS5040 sensor SSI interface timing diagram.



Source: AS5040 sensor datasheet [31].

pulse width of 1 μs as shown in Figure 4.3.

Figure 4.3: AS5040 sensor PWM interface output illustration.

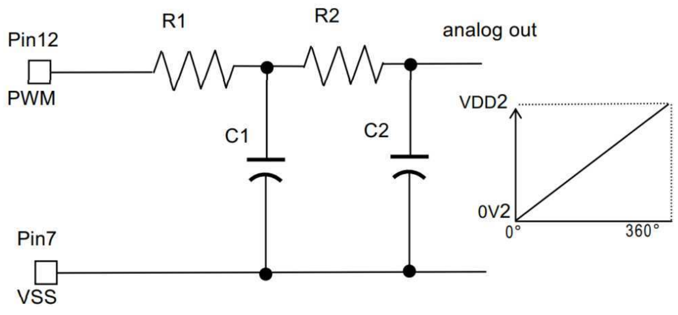


Source: AS5040 sensor datasheet [31].

An averaged signal can be acquired through low-pass filtering detailed in Figure 4.4. In this case, the minimum angle would be 0°, corresponding to 0 V, while the maximum angle would be 360°, and the voltage 5 V. Once the PWM signal is filtered, the microcontroller A/D converter could sample this signal.

The main advantage of using this PWM interface is that the sampling time would be restricted only by the interruption time, not the number of pulses. Hence, the sampling rate is 22 times faster than the other approach and could positively impact the implementation of the future control loop.

Figure 4.4: Low-pass filter for the PWM output signal. R_1, $R_2 \geq 4K7$. C_1, $C_2 \geq 1\mu F/6V$



Source: AS5040 sensor datasheet [31].

4.2 Armature Current - I_a

The system is equipped with a Rev3 motor shield for the DC motor control using the L298 IC. Nevertheless, the motor shield has a built-in feature to measure the current going through the motor reading the **SNSx** pins. On each pin, a voltage will be proportional to the circulating current. The maximum possible current is 2 A, and the calibrated voltage is 3,3V.

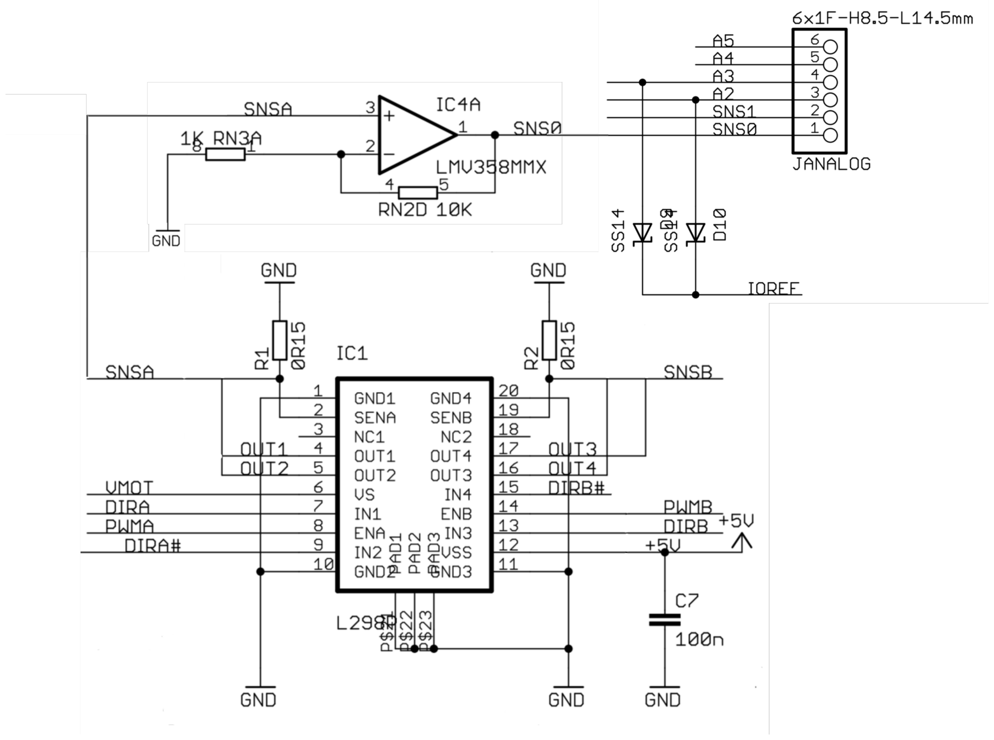
As Figure 4.5 shows, the sensing pins **SENA** and **SENB** are connected to $0,15\ \Omega$ resistors. Then, there is an amplifier to process those voltage signals. The purpose is to deliver at the analog pins **A0** and **A1** of the Arduino board a signal with the desired level of amplification without affecting the input signal's impedance.

4.2.1 Measurement of the armature voltage

There is a voltage drop between the H-bridge that controls the DC motor and the effective delivered armature voltage, and it varies non-linearly with the circulating current. Hence, it is necessary to measure this differential voltage in real-time. For this, the Arduino Mega board has built-in differential measurements, enabled by selecting the proper bits in the ADMUX register. Since the voltage limits applied to the DC motor do not trespass the 2,5 V, selecting the **REFS0** and **REFS1** bits, we select the 2,56 V as the voltage reference for conversion. For selecting the input channels for the differential operation, the **MUX** bits are selected as **0b010011**, which designates the analog pins **A3**, **A1** as the inputs.

Figure 4.6 shows the sequence of duty cycle $\tau(\%)$ steps applied to the motor and the

Figure 4.5: Schematic of the motor shield Rev3.



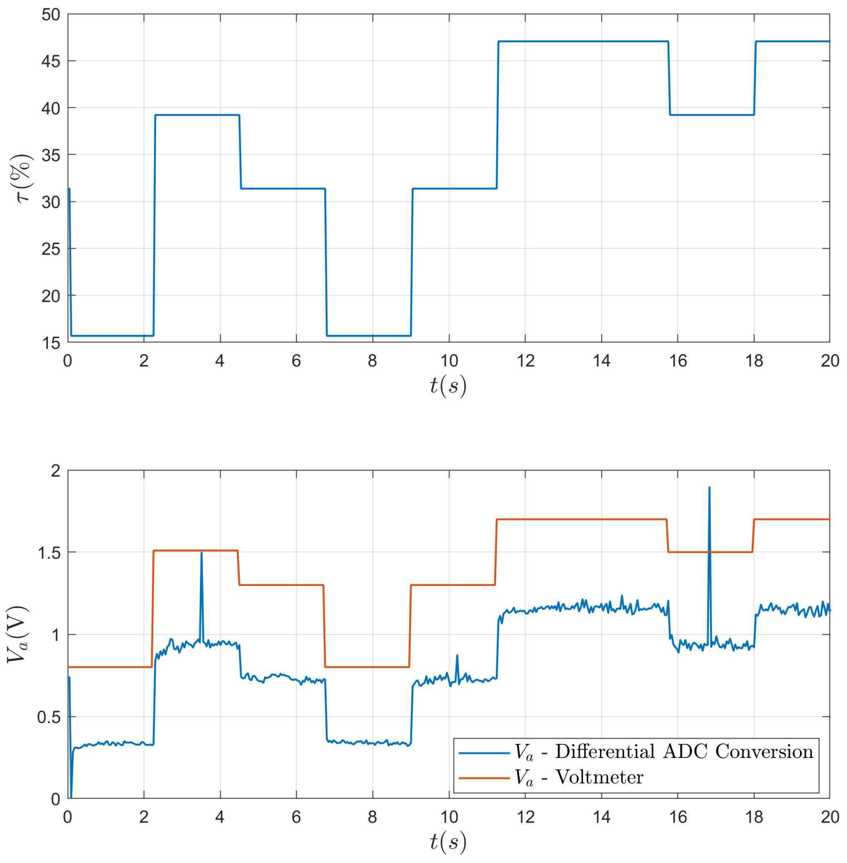
Source: Adapted from the Arduino Rev3 Motor shield datasheet [32].

corresponding voltages measured. The format of the voltage curves is compatible. However, there is a discrepancy between the values measured by the voltmeter and those acquired by the differential measurement mode in the microcontroller board. We could not characterize the voltage offset between the curves in orange and blue, which has led to a curve fitting that relates the voltage input delivered to the H-bridge and the voltage delivered to the armature current circuit.

4.3 Acquisition schemes

Developing an acquisition scheme running solely on the Arduino board would yield two significant benefits. First, it would grant greater control over the sampling rate, enabling the utilization of different acquisition modes, such as the PWM built-in interface in the position sensor. Second, it would ease low-level hardware configuration, empowering the triggering of events based on Arduino Timers, thereby ensuring regular sampling intervals. Moreover, interrupt-based events would allow for sampling diverse signals due to hardware

Figure 4.6: Differential voltage measurements. The first plot shows the voltage steps applied to the H-bridge, and the second one shows the voltage values measured both with the voltmeter and the ADC.

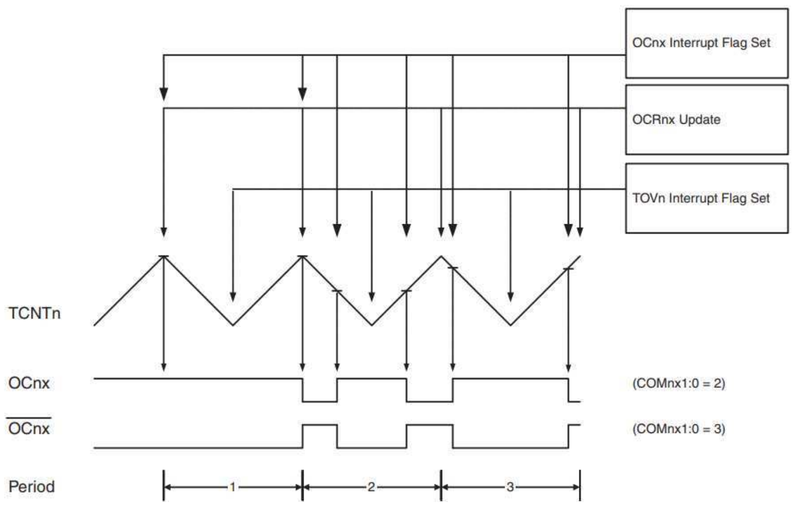


Source: Prepared by the author.

phase correction.

As previously detailed, variables, such as the electrical ones, could not be correctly measured mainly due to the high-level acquisition scheme proposed. The armature current and voltage values must be collected to properly characterize the dynamics between the actuator and the mechanical parts. Figure 4.7 shows that a hardware-generated PWM signal controls the H-bridge and delivers power to the armature current circuit. The onboard PWM signal bases the acquisition logic by triggering all the sampling events. For the aeropendulum application data collection, it is necessary to generate symmetrical output waveforms. This symmetry is required to guarantee that interruptions occur when the sampled signals are at a high level. In this case, a phase correction on the PWM signal is necessary.

Figure 4.7: PWM phase corrected mode timing diagram.



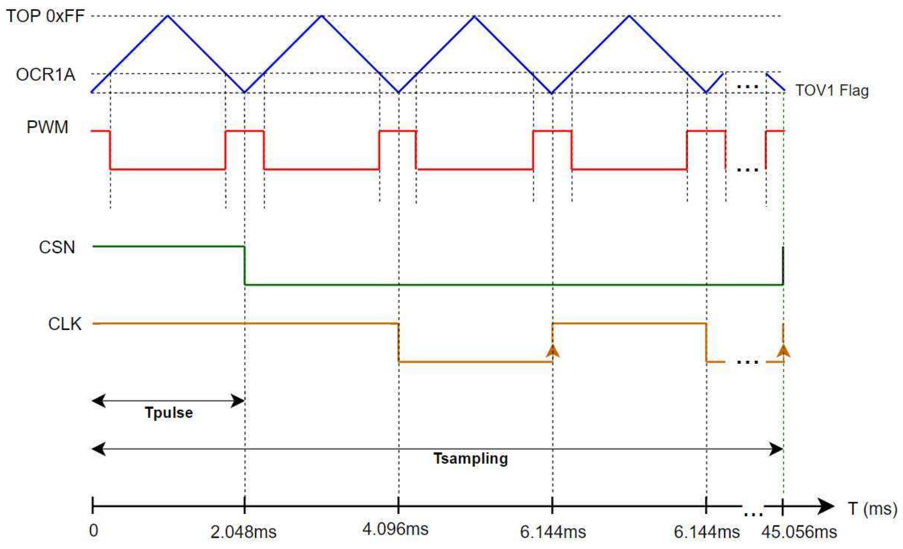
Source: ATmega 640 Microcontroller Datasheet [33].

The phase-correct PWM mode provides a dual-slope waveform generation, and the Timer/Counter Control Register WGMx bits as **0b001** configures this setting. The counter counts from a bottom to a top value, defined as **OxFF**. Each time the counter reaches the BOTTOM value, it enables the Timer/Counter Overflow Flag. In the case of the acquisition scheme proposed here, this interrupt flag bases the signal generation and data collection.

4.3.1 First strategy - SSI interface

Figure 4.8 depicts the proposed timing diagram. By phase-correcting the PWM signal of the microcontroller, there is a guarantee that the PWM-dependent signals, such as the armature current and voltage, will be sampled at their high levels. During the TOV1 overflow interruption, the armature current, voltage data, and filtered position information are gathered. The sampling rate is influenced by the PWM frequency of the Arduino board and the number of necessary pulses to complete the data-acquisition procedure, resulting in a sampling time of $T_s = \frac{1}{490} \times 22 = 45,056$ ms. The system's natural oscillating frequency is 5,46 rad/s, and the default sampling frequency of 490 Hz meets and exceeds the recommended data acquisition frequency. A set of voltage values is applied to carry out the data acquisition process.

Figure 4.8: Timing diagram proposed. The curve in blue represents the Arduino's Timer1 counter. In red is the phase-correct PWM signal. The chip selection and clock signals are in green and orange, respectively. The TOV1 overflow flag generates an interruption each time the counter reaches zero. After the first positive edge of the clock signal, the 10-bit data stream begins, and it takes 45.056ms to complete the data acquisition cycle.



Source: Prepared by the author.

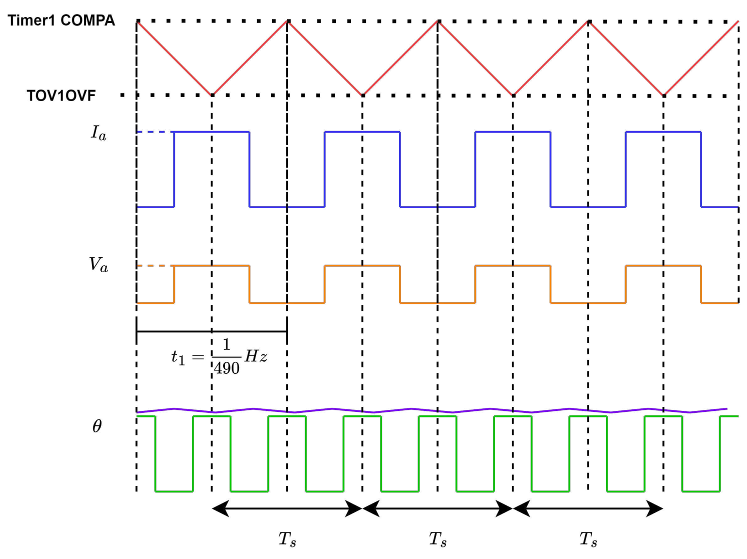
Each time the counter from Arduino's Timer One reaches zero, it corresponds to one pulse completed. Twenty-two pulses are necessary to read the 10-bit resolution angular position, which is this approach's main drawback. Given that the sampling time depends on the number of pulses to acquire the angular position, which cannot be changed, if it is necessary to increase the sampling rate, it would be necessary to increase the PWM frequency of the board.

4.3.2 Second strategy - PWM interface

As detailed in section 4.1, the PWM interface outputs a signal with a duty cycle proportional to the angular position of the magnetic rotary encoder. A low-pass filter with the specifications previously defined should filter the angular position signal. Then, an analog input from the microcontroller should read the filtered angular position. With that, we can design an acquisition scheme that does not rely on chip selection and clock signals.

As Figure 4.9 depicts, Whenever the counter of Arduino's timer one reaches zero, an overflow interruption called TOV1OVF is triggered. During this interruption, the armature current, voltage data, and filtered position information are gathered. The sampling rate is directly influenced only by the PWM frequency of the Arduino board, resulting in a sampling time of $T_s = \frac{1}{490} = 2,04$ ms for the default operation frequency of 490 Hz.

Figure 4.9: The proposed acquisition timing diagram. The curve in red represents the Arduino's Timer1 counter. The blue, yellow, and purple curves represent the armature current I_a, the armature voltage V_a, and the low-pass filtered angular position θ. The TOV1 interruption flag is enabled each time the counter reaches the bottom value. Those interruptions occur each 2,048 ms.

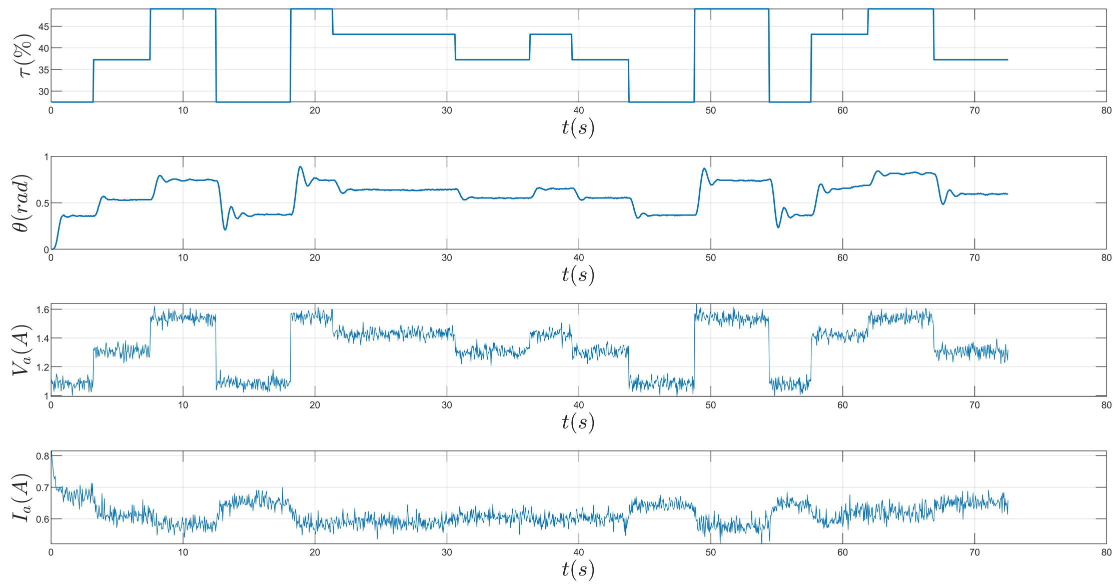


Source: Prepared by the author.

4.3.3 SSI and PWM data acquired

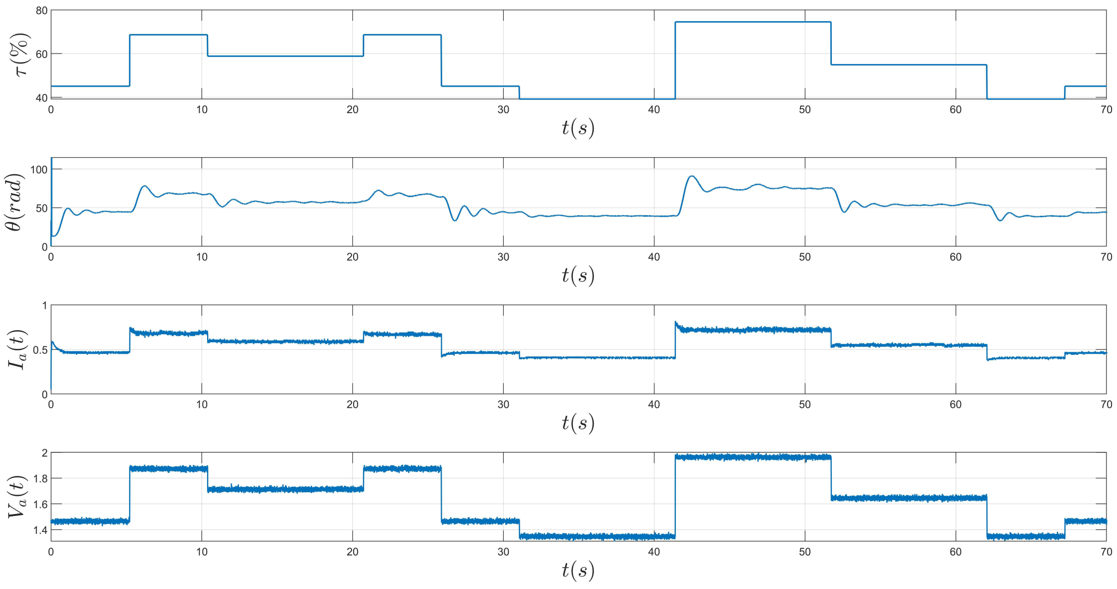
Figure 4.10 and Figure 4.11 depict the data collected from the system. More specifically, the duty cycle $\tau(\%)$, the angular displacement θ(rad), the armature current I_a(t), and the armature voltage $V_a(t)$ using both acquisition schemes. They are equivalent in data collected but differ regarding the sampling rate. While the SSI acquisition scheme requires 46,056 ms and yields multiple voltages and current values discarded, the PWM averaging scheme can collect system information every 2,048 ms. The control loop might benefit from a faster sampling rate and more informative data collection. The chosen acquisition for the plant is the PWM one for simplicity in the implementation and the faster sampling rate.

Figure 4.10: Data collected using the SSI interface and the acquisition timing diagram proposed in subsection 4.3.1 for seventy seconds of collection.



Source: Prepared by the author.

Figure 4.11: Data collected using the PWM interface and the acquisition timing diagram proposed in subsection 4.3.2 for an interval of seventy seconds of collection.



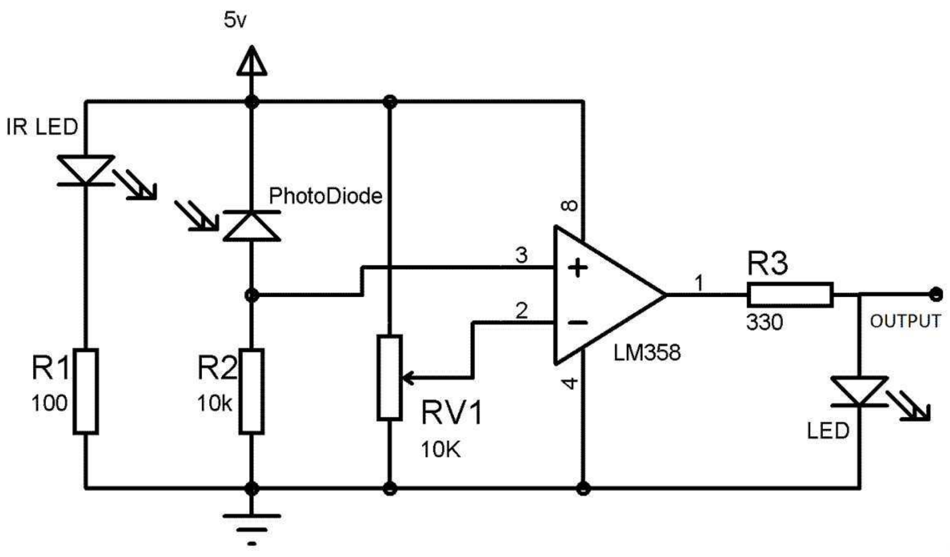
Source: Prepared by the author.

4.3.4 Propeller speed acquisition

The rotor speed is vital to the estimation algorithms and control strategies. Nonetheless, due to size limitations and mechanical restrictions, it is necessary to have a measuring module that does not affect the pendulum's weight. Given this background, we use an IR sensor module based on the LM358 IC. Since it captures the speed of the propeller, we can use the gearbox's gear ratio to infer the rotor's speed.

Figure 4.12 shows the electrical diagram of the sensor. The comparator's output is low when no IR radiation is on the photodiode. Once a reflecting object is in front of the IR set, the light reflected by the object results in a voltage drop across the photodiode, increases the voltage across the R2 resistor, and increases the output of the comparator is high.

Figure 4.12: IR module circuit diagram.

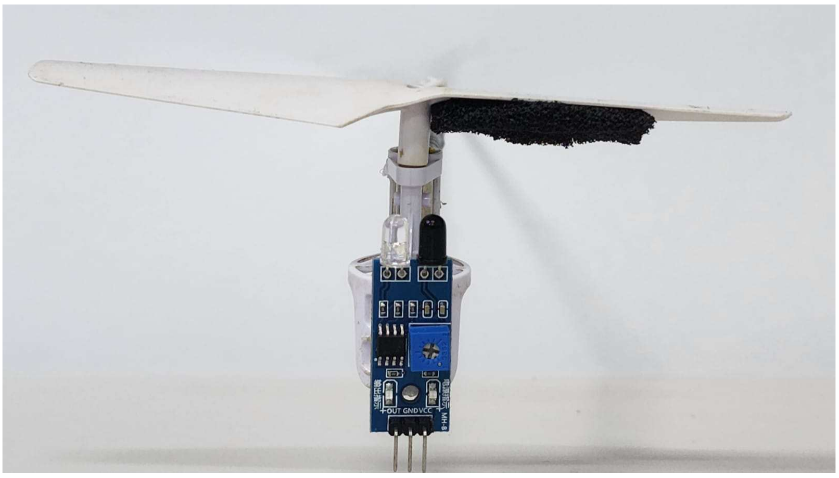


Source: LM358 IC datasheet [34].

The idea is to use the IR module coupled to the motor-gear set and measure the rotor speed indirectly from the propeller speed using the gear ratio. Figure 4.13 shows the measurement setup. A black piece of foam in the back of the propeller blade blocks the light reflection. The foam was scraped so it was close to the propeller and would not affect its aerodynamics. This way, each completed revolution generates a voltage pulse.

Different strategies are commonly used to measure the speed of propellers using IR sensors. Most lack accuracy due to high-level programming and fixed-size observation win-

Figure 4.13: IR sensor module coupled to the propeller-gear set. The black foam is attached to one of the propeller blades, and it is used to determine when a revolution is complete.



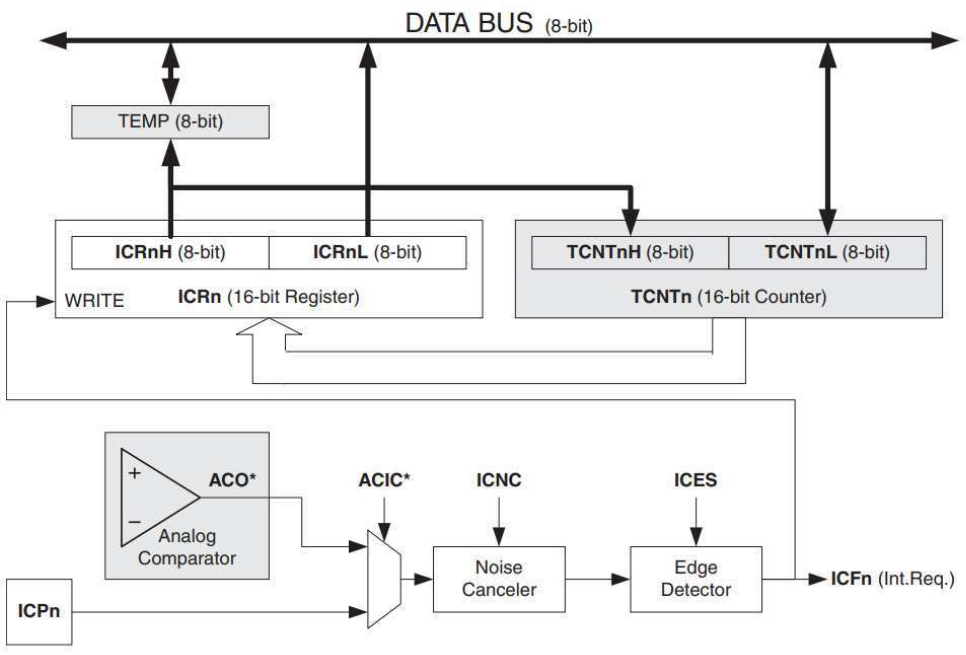
Source: Prepared by the author.

dows, that is, counting the number of pulses over the same time interval. This approach is significantly inaccurate at lower speeds. Hence, the strategy used here is based on the input capture mode using the Timer 4 in the Arduino Microcontroller. Figure 4.14 depicts the input capture unit scheme.

The ICP4 pin receives the voltage pulses coming from the IR sensor. The value of the timer counter control register is initially set to one, which enables positive edge pulse detection. The current time tick goes to the IRC4 register each time the event occurs. The value of the ICE4 bit is inverted each time a transition occurs, and the pulse width is then measured as a difference of the correspondent time ticks between a rising and a falling edge. Based on the microcontroller's pre-scaler settings, it is possible to determine the time related to the number of time ticks.

The DC motor was excited with a sequence of voltage steps, and the corresponding speed was collected, as depicted in Figure 4.15. The measured propeller speed presents various spikes greater than 500 RPM, compromising the speed measurement's accuracy for future applications. This measurement noise can be linked to hardware problems since the value of the ICR4 register presented counter overflows. As an alternative, the code uses the **micros()** counter to determine the time between rising and falling edges, limiting the pulse width's precision to 4 μs. These results indicate that another approach must aim for a more accurate propeller speed estimation.

Figure 4.14: Input-capture unit block diagram. Using one of the IPCn pins and enabling edge detection, the value of the TCNTn register is transferred and delivered to the data bus.



Source: ATmega 640 Microcontroller Datasheet [33].

4.4 Estimation improvements

This section details the improvements made concerning estimating the electrical parameters, specifically the armature equivalent ones, and the nonlinear relationship between the PWM action and the armature voltage. In addition, there is an improvement in the description of the thrust force.

4.4.1 Armature resistance - R_a

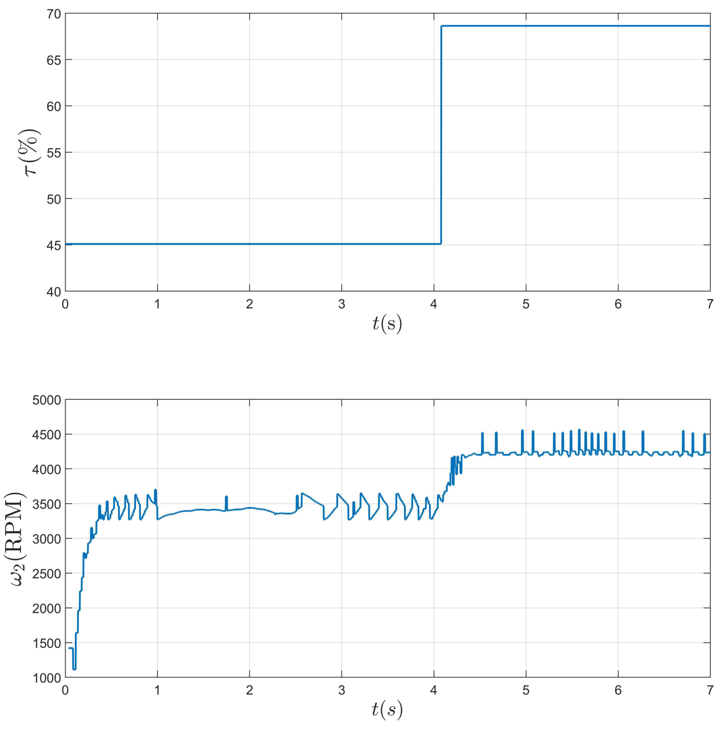
The Figure 4.16 depicts the electrical equivalent of a DC motor's armature.

The back-electromotive force is expressed by:

$$e_a = v_a - R_a i_a - L\frac{di_l}{dt} \tag{4.1}$$

where, v_a is the armature voltage, R_a and L_a the armature resistance and inductance, respectively. By blocking the rotor, since $e_a = K_\omega \omega^2$, the angular speed in the motor axis

Figure 4.15: The duty cycle $\tau(\%)$ steps applied to the motor, and the corresponding propeller angular speed ω_2 in RPM collected.



Source: Prepared by the author.

will be zero, and so will be the back EMF. Considering that the inductance is negligible, one can find the armature resistance by

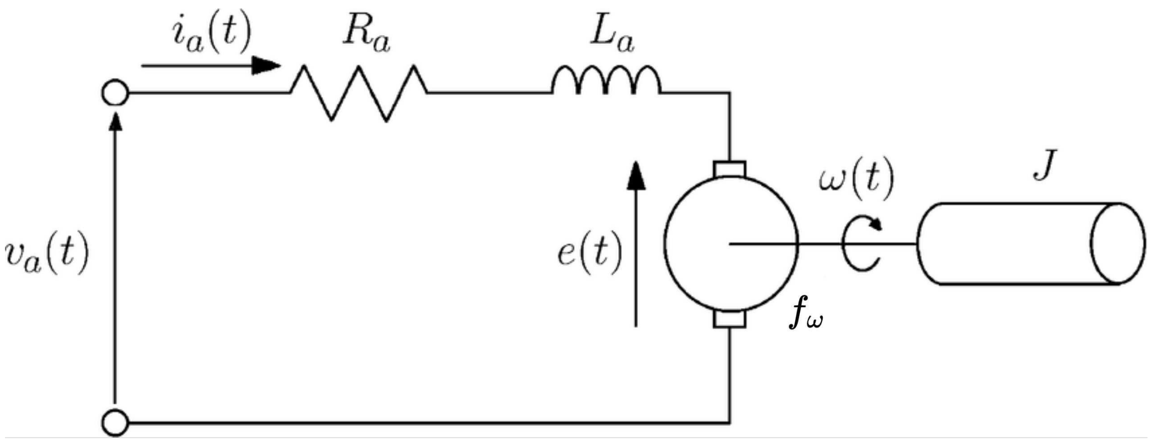
$$R_a = \frac{v_a}{i_a}, \tag{4.2}$$

where by varying the armature voltage, v_a, applied to the motor, and monitoring the values of i_a, one can determine the mean R_a. The experimental results are presented in Table 4.1, along with the mean $\hat{R}_a(\Omega)$ and its standard deviation σ_{R_a}.

4.4.2 Estimation of the armature inductance L_a

To estimate the value of L_a, we induced a voltage step (V_a) and monitored the resulting I_a, as Figure 4.17 shows. By analyzing the time interval it took for I_a to achieve 63% of its steady-state value, we determined the electrical time constant as $\tau_e = \frac{L}{R}$, resulting in a

Figure 4.16: Illustration of the equivalent armature circuit.



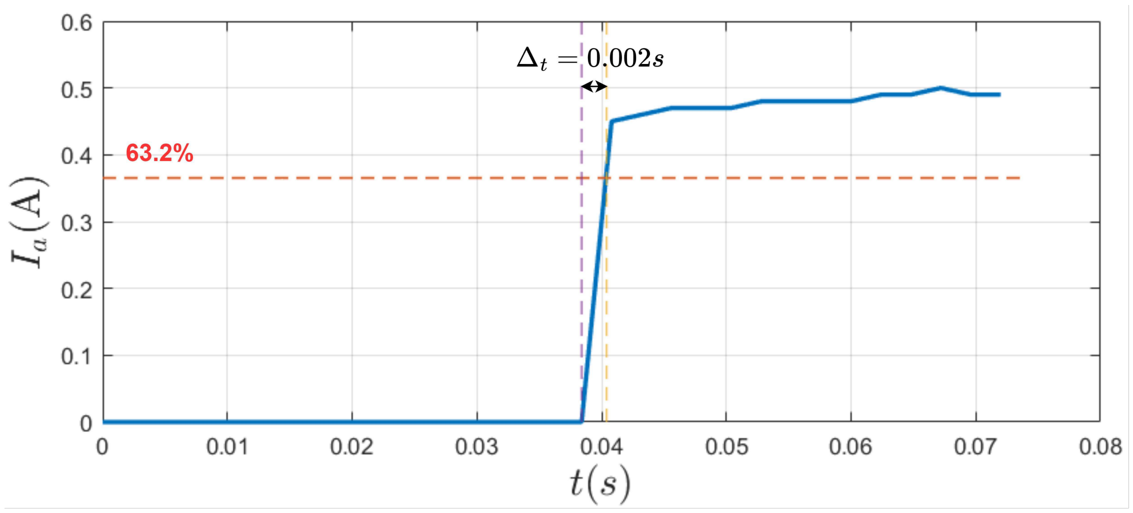
Source: Prepared by the author.

Table 4.1: Armature voltage v_a and armature current i_a values collected, and the estimated R_a.

v_a (V)	$i_a(A)$	$R_a(\Omega)$
0,48	0,44	1,08
0,60	0,55	1,08
0,29	0,25	1,17
1,42	1,30	1,09
1,47	1,33	1,11
0,35	0,30	1,17
0,89	0,85	1,05
0,74	0,69	1,07
1,31	1,17	1,12
1,54	1,40	1,10
0,78	0,74	1,05
1,58	1,57	1,00
0,79	0,73	1,09
0,99	0,96	1,03
1,76	1,67	1,05
0,57	0,52	1,09
0,37	0,35	1,07
1,93	1,89	1,02
1,37	1,32	1,04
2,44	2,28	1,07
1,09	0,99	1,10
$\hat{R}_a$		
1,08		
σ_{R_a}		
0,04		

calculated $\hat{\tau}_e$ of $0,002$ s. Using the given $\hat{R}_a$ value, we approximated $\hat{L}_a$ to be $2,2$ mH.

Figure 4.17: Armature current (I_a) time response for a step of $\tau = 50\%$ in the armature voltage (V_a).



Source: prepared by the author

4.4.3 Back Electromagnetic force constant

Considering that the armature inductance value L_a is negligible, hence that there is a minimal voltage drop across the inductor, we can estimate the back-EMF constant, K_ω. As Table 4.2 details, we monitor the values of v_a and i_a. Once the propeller is at its steady-state speed, we measure its speed using a strobe meter and employ the gear ratio to infer the rotor's speed. After that, we use the measured velocity and compute the approximate $\hat{K}_\omega = \frac{v_a - R_a i_a}{\omega}$. We found the mean value for $\hat{K}_\omega$ and the standard deviation σ_{K_ω} of the approximations.

4.4.4 Armature voltage estimation

As subsection 4.2.1 details, it was not possible to diagnose the instrumentation problems and perform differential voltage measurements. Since the voltage's operating region is restricted, the idea would be to obtain a relationship between the voltage applied to the H-bridge and the armature voltage after the voltage drop.

Table 4.2: Experimental results for determining the back-EMF constant using the electrical quantities collected in steady state and the rotor's speed, converted using the gear ratio information.

v_a **(V)**	i_a **(A)**	ω_1 **- RPM (strobe)**	K_ω
1,08	0,29	1.158	0,0006619934856
1,182	0,33	1.235	0,0006684402018
1,272	0,35	1.306	0,0006847902019
1,377	0,4	1.402	0,0006742762768
1,483	0,45	1.485	0,0006715175383
1,561	0,5	1.561	0,0006538620054
1,666	0,55	1.638	0,0006543423154
1,781	0,61	1.734	0,0006470698432
1,864	0,65	1.805	0,0006438833666
1,974	0,7	1.885	0,0006462672905
			$\hat{K}_\omega$
			0,0006606442526
			σ_{K_ω}
			0,00001377310893

Different voltage values were collected by applying different duty cycle levels, as shown in Figure 4.18.

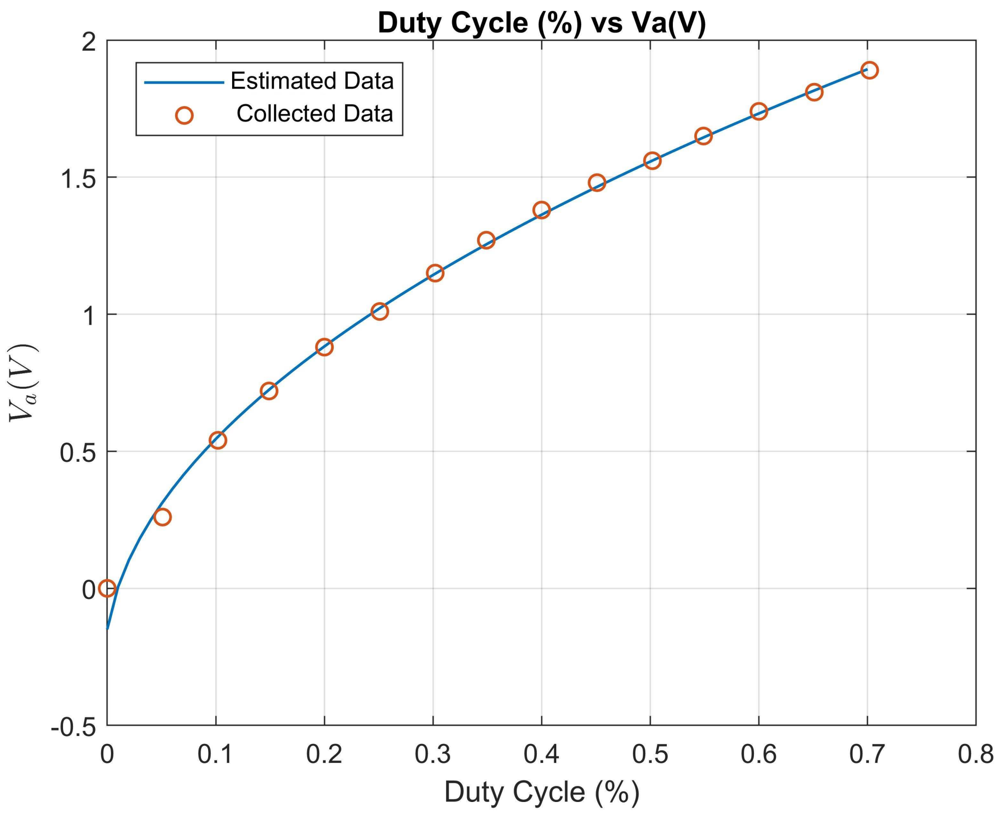


Figure 4.18: The armature voltage V_a as a function of the duty cycle τ; acquired and estimated data through (4.3).

Using the polynomial curve fitting method, which led to an MSE of $2,6 \times 10^{-5}$, the relationship between duty cycle and armature voltage is described by 4.3. This interpolation should be helpful since it enables the knowledge of the armature voltage inside a limited range of applied duty cycle values. It is given by:

$$V_a = k_0 + k_1\sqrt{k_2\tau + k_3} \tag{4.3}$$

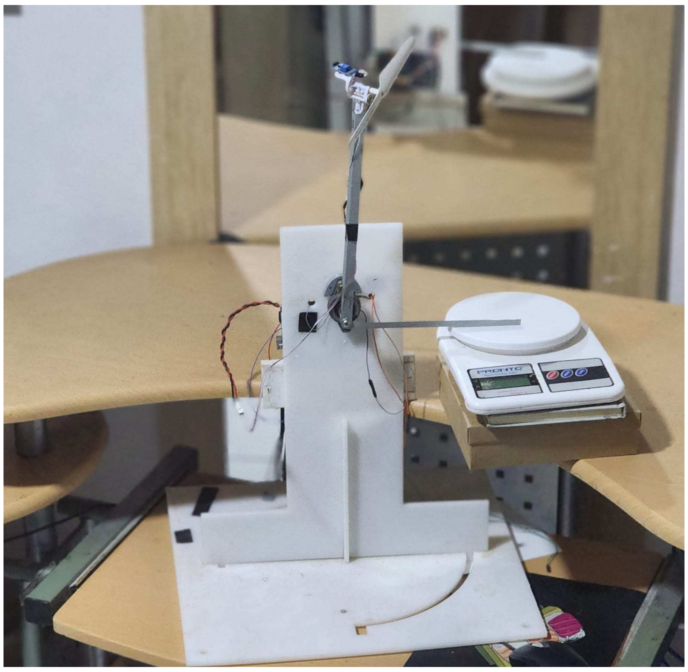
where, $k_0 = -0,2914,\ k_1 = 6,8 \times 10^{-4},\ k_2 = 1,469 \times 10^7, k_3 = 4,3629 \times 10^4$. This equation makes it feasible to determine the voltage applied to the DC motor based on the τ applied to the H-bridge.

4.4.5 Thrust force interpolation

The expression $f = K_q\omega_2^2$ describes the thrust force as a function of the speed of the propeller. Thus, it is necessary to determine the thrust coefficient. For that, it is necessary to perform tests using wind tunnels (Czyż et al.[35]), which are expensive and often inaccessible. Another way to characterize the thrust of a propeller is by using a DC motor, a test stand, and a load cell (Prasetiyo[36],KÓSA et al.[37]). The propeller's thrust force is transmitted to the load cell using a structure that connects the rotating rod to the propeller.

In this work, we try an approach based on this idea. We perform a curve fitting to determine the relationship between the armature voltage and the force produced by the propeller. The fitting should be further used to convert the signal from the controller to an actuation value applied to the DC motor. As Figure 4.19 depicts, the load cell is calibrated when the DC motor is off. We gradually increase the armature voltage from 0 V to the maximum value to which the angular displacement is maximal, and the load cell measures the corresponding weights. Finally, we perform a curve-fitting on the collected values, as Figure 4.20 shows.

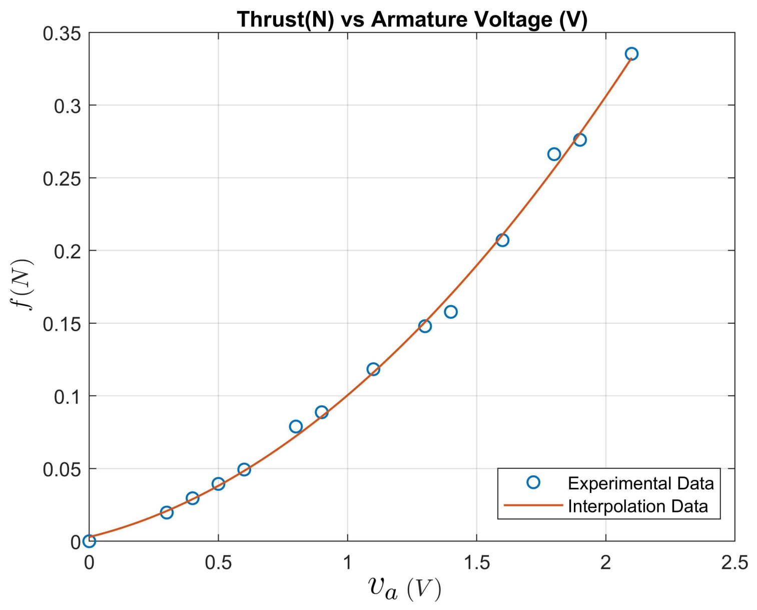
Figure 4.19: The adapted thrust bench and the load cell to estimate the thrust force.



Source: Prepared by the author.

The fitting curve is expressed by Equation 4.4.

Figure 4.20: The multiple voltage values v_a and the estimated thrust force points F collected. The curve in orange represents the mathematical interpolation from the data.



Source: Prepared by the author.

$$f = 0,054v_a^2 + 0,0435v_a + 0,0029 \tag{4.4}$$

In order to convert the thrust signal from the controller, the inverse function is expressed through Equation 4.5.

$$v_a = 0,4028 + 9,2593\sqrt{0,0013 + 0,2160f} \tag{4.5}$$

4.5 Simulation model

We can develop a simulation model that mimics the plant's behavior based on the estimated electrical and mechanical parameters previously determined [5]. It reduces the risk of operating the plant in inadequate conditions, enhances the prototyping process, and yields the rapid implementation of multiple control approaches. Applying simulation methods gives a deeper understanding of the system's behavior and dynamics, thus providing confidence to deal with the real system.

Concerning the possible control techniques, the simulated models facilitate the rapid

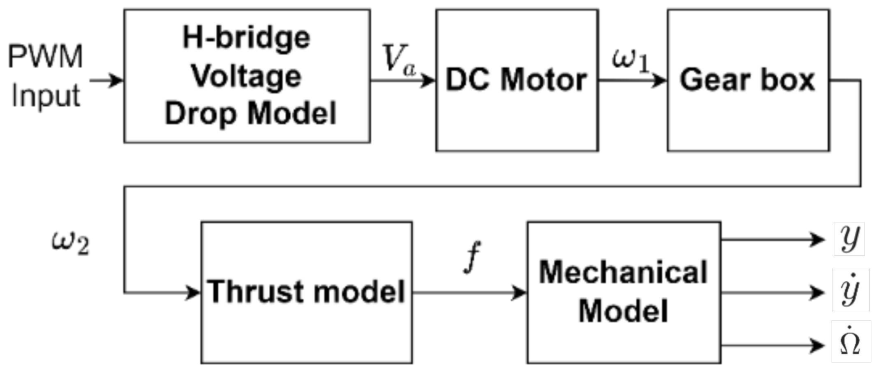
Table 4.3: The parameters used in the simulation model

	Model Parameters
Electrical	$R_a = 1,08\ \Omega\ L_a = 2,2\ \text{mH}$
Mechanical and Aerodynamic	$K_\omega = 6,6 \times 10^{-4}\ \text{Vs/rad},\ K_t = 0,235\ \text{Nm/A}$
	$J_m = 5,2 \times 10^{-5}\ Kgm^2,\ B_m = 0,01\ Ns/m^2$
	$\frac{c}{J} - 13,82,\ \frac{mgd}{J} - 131,75,\ \frac{L}{J} - 462,83$
	$\frac{N_1}{N_2} = 55/9,\ K_q = 1 \times 10^{-4}$

implementation and evaluation of different approaches, such as small-signal linearization, input-output linearization, and pole allocation. This fast implementation accelerates comparing control strategies, optimizing parameters, and identifying the most effective approach to enhance closed-loop performance.

Figure 4.21 depicts the aeropendulum model represented as a block diagram. The ODEs that describe the motor dynamics are included on the electric motor block, which outputs the rotor's angular speed, armature current, derivatives, and input voltage. The gearbox and the propeller dynamics are used to describe the aerodynamic effects that produce the thrust force. The aeropendulum block receives the corresponding thrust force as input and outputs the angular displacement. The mechanical parameters of the plant were obtained from Lucena, Luiz and Lima[5], where a nonlinear optimization method was used. The parameters used for the simulation model are detailed in Table 4.3.

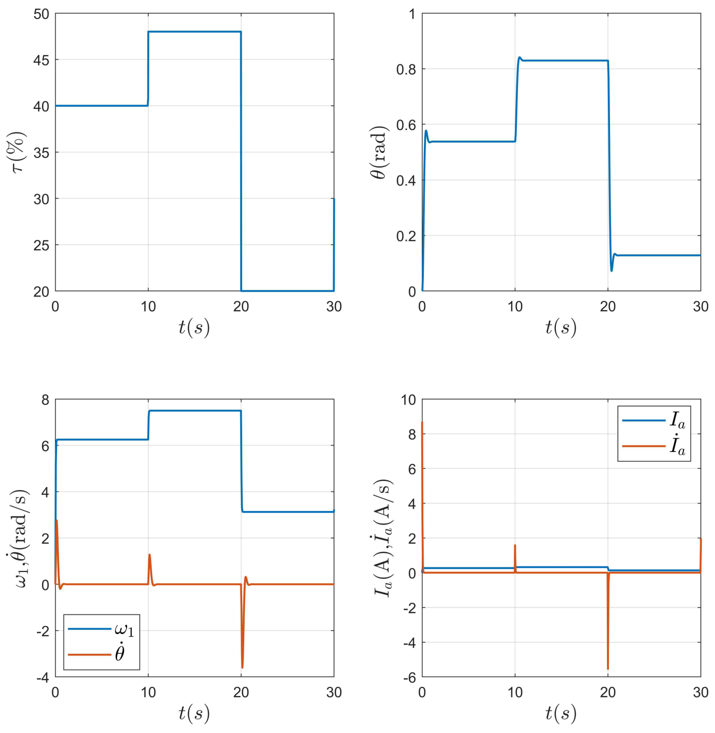
Figure 4.21: Simulation model composed of the motor, gearbox, aerodynamic conversion, and mechanical blocks. The generated data from the simulation is sent to the workspace.



Source: Prepared by the author.

As Figure 4.22 depicts, for a sequence of duty cycle steps applied to the armature equivalent circuit, the angular displacement of the aeropendulum θ, the angular speed concerning the vertical axis, $\dot{\theta}$, and the rotor's angular speed, ω_1, as well as the armature current information I_a are collected.

Figure 4.22: Data collected and sent from the simulation to the workspace.



Source: Prepared by the author.

4.6 Final considerations

This chapter described the improvements regarding the instrumentation of the experimental platform and the refinements on the estimation of electrical and aerodynamic aspects. It also discussed the proposed acquisition schemes, the reason for using one at the expense of the other, and the implementation of a simulated model to experiment and gain confidence in the problem solution. The next chapter will address the use of the SINDy algorithm variants to characterize the dynamic behavior of the experimental plant.

Chapter 5

Parameter Estimation Using the SINDy-based Frameworks

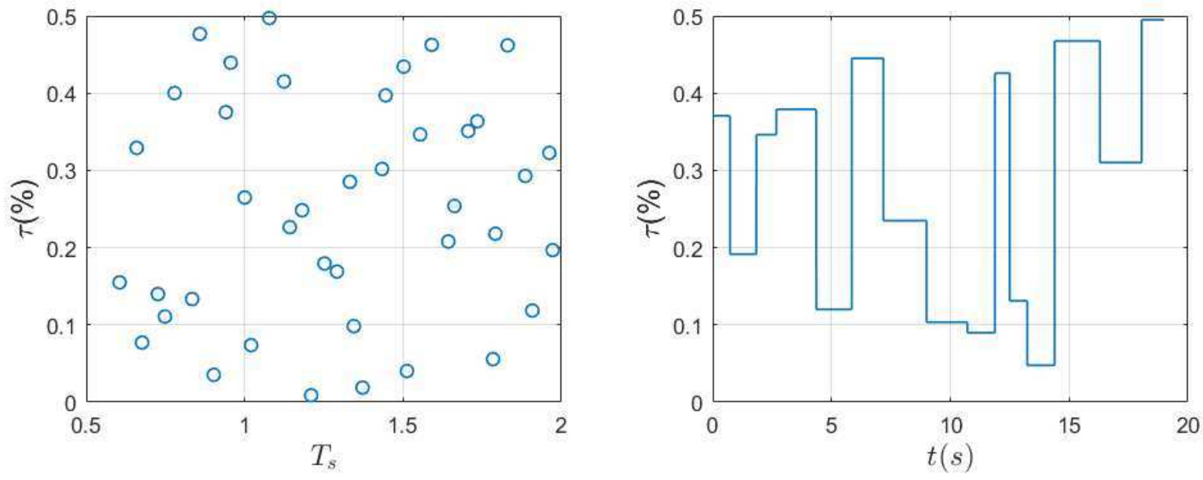
This chapter describes the results regarding using SINDy-related algorithms to determine the electrical and aerodynamic parameters of the aeropendulum plant. The objective is to evaluate the capability of the sparse identification algorithms to determine an improved data-driven model, specifically in terms of electrical-aerodynamic characterization. Furthermore, we want to evaluate the impact of pre-processing steps on the accuracy of the discovered model.

5.1 Input signal design

A vital element of the parameter estimation procedure entails shaping the experiment's framework, commonly called the Design of Experiment (DoE). When addressing the identification of nonlinear systems, a critical stride involves crafting an excitation signal with informative qualities and the ability to manifest the hidden dynamics [38]. Although extensively debated in the domain of system identification theory, this subject has garnered limited attention in the context of the SINDy algorithm, as underscored by prior investigations ([39]), thus warranting further exploration.

To extract valuable insights via DoE, one can adopt either of two distinct approaches: the model-based or the model-free route. In the present context, an apt model-free DoE technique involves leveraging the Latin Hypercube distribution to engender the excitation signal. Delimiting the input space to $\tau \in [\tau_{max}, \tau_{min}]$ and $T_s \in [T_{s_{min}}, T_{s_{max}}]$, it is partitioned into ten discrete intervals, and divided into ten intervals, which correspond to the prior information related to the actuation limits and operating characteristics. The visualization presented in Figure 5.1 shows the configuration of the input space design points and the excitation signal on the right.

Figure 5.1: The excitation signal. Each point represents a step signal with amplitude τ and a duration time.



Source: prepared by the author.

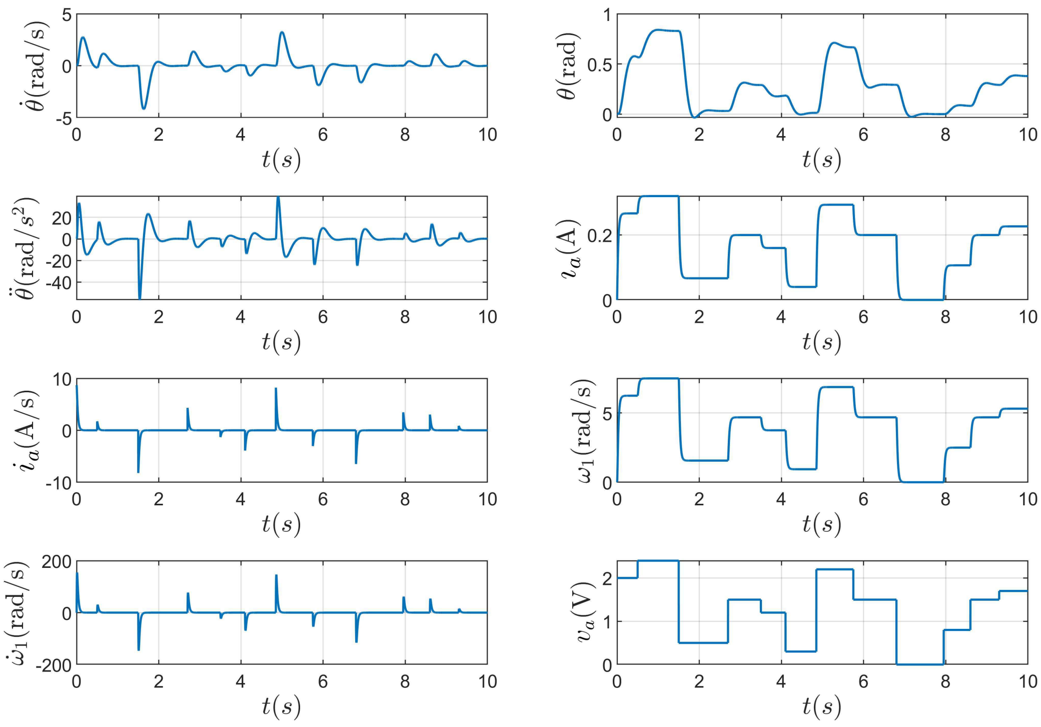
5.2 Sparse system identification using noise-free measurements

Using the simulation model previously described, we utilize the Latin Hypercube (LHC) input space design to excite the system. The simulation data is used because it provides knowledge of dealing with data of different qualities and which approaches are most effective. For this first case, we consider that all state derivatives can be measured, and no noise is added to the measurements.

5.2.1 Identifying the state variables' derivatives of the Aeropendulum - SINDy and WSINDy

Figure 5.2 depicts the state variable's derivatives collected from the simulation and the measurement variables, corresponding to 75% of the simulation results. 25% was left to the test set. A library function $\Theta(\theta, \dot{\theta}, \omega_1, v_a)$ containing polynomial terms of second order and trigonometric terms was constructed, with products between the measurement variables. The dynamics, that is, the derivatives of the state variables, were not approximated, and $\frac{dX}{dt} = \begin{bmatrix} \dot{\theta} & \ddot{\theta} & \dot{\omega}_1 & \dot{i}_a \end{bmatrix}$ was composed directly from the simulation data output.

Figure 5.2: Simulation results from the simulated model. In the left column, the derivatives of the state variables. On the right side are the state variables of interest.



Source: prepared by the author.

Using the SINDy algorithm, the threshold parameter $\lambda = 0.9$, and the library function Θ, the sparse matrix Ξ. The library function comprises the state variables, the input signal, and the cross-products of these variables and their powers and trigonometric composites. Table 5.1 exemplifies the representation of a dynamic system. The column labels $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$, and $\dot{x}_4$ refer to the state variables' derivatives, and the non-zero terms in each correspondent row represent which terms accompany the library function candidates. This way, from the table, we interpret that $\dot{x}_1 = x_1$, $\dot{x}_2 = x_1+2x_2+u_1$, $x_3 = 2x_3+x_4+u_1$, and $\dot{x}_4 = 3x_3+x_1x_2+1$.

Table 5.1: Example of the sparse matrix that represents the derivatives of the state variables $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$, and $\dot{x}_4$.

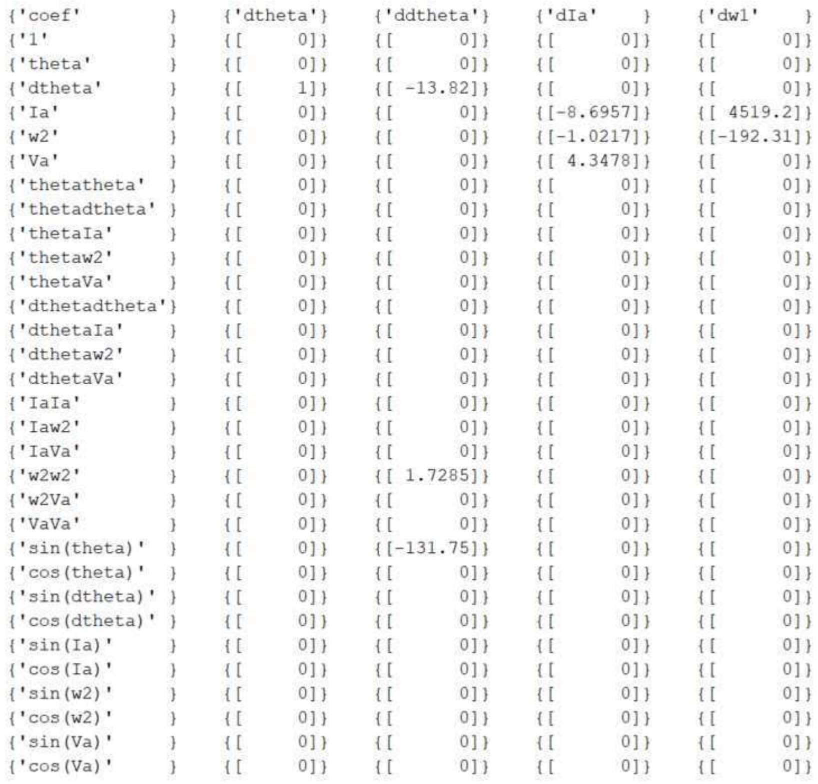
	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	$\dot{x}_4$
1	0	0	0	1
x_1	1	1	0	0
x_2	0	2	0	0
x_3	0	0	2	3
x_4	0	0	1	0
u_1	0	1	1	0
x_1x_2	0	0	0	1
x_1x_3	0	0	0	0
x_1x_4	0	0	0	0
x_1u_1	0	0	0	0

The sparse matrix was estimated and illustrated in Figure 5.3. Each column represents the sparse vector for the respective state variables' derivatives such that 'dtheta' refers to $\dot{\theta}$, 'ddtheta' represents $\dot{\theta}$, 'dIa' denotes $\dot{i}_a$, and 'dw1' is $\dot{\omega}_1$. For example, to describe the angular acceleration $\ddot{\theta} = -\frac{mgd}{J}\sin(\psi) - \frac{C}{J}\Omega + \frac{L}{J}f$, the angular velocity column in the figure, the candidate terms related to the square of the speed of the propeller, and the sine of the angular position are active. All the coefficients that could correctly describe the derivatives of the state variables were discovered with no spurious terms.

When it comes to using the WSINDy algorithm, the only difference is that it is unnecessary to compute the approximate derivatives to solve the least-squares problem. We use the hyperparameters, which are a set of values that control the learning process, $K = 120$, $p = 2$, $s = 16$, $r_{whm} = 30$ recommended in the original paper (Messenger and Bortz[22]). Getting information from the problem domain, we chose the threshold as $\lambda = 0.9$ and the regularization factor $\gamma = 0.000001$. As Figure 5.4 depicts, the sparse representation for the system was retrieved.

Talking specifically about the aerodynamic terms, we want to analyze the use of the SINDy algorithm to model the propeller's thrust coefficient. Since the term that accompanies ω_1^2 is present in the dynamic equation of $\ddot{\theta}$, using the corresponding value of the sparse matrix, $K_q = \frac{1.729}{\frac{L}{J}(\frac{N_1}{N_2})^2} = 0.0001$. Hence, the SINDy and WSINDy algorithms could correctly identify the propeller behavior and determine the parameters that describe the other derivatives.

Figure 5.3: The sparse matrix Ξ estimated using the SINDy algorithm. Each column represents the approximation of the state variables' derivatives, and each non-zero term activates a nonlinear dynamic that models the behavior. The reference analytical equations are $\dot{\theta} = \dot{\theta}$, $\ddot{\theta} = -13.82\dot{\theta} + 1.73\omega_2^2$, $\dot{i}_a = -8.7i_a - 1.02\omega_2 + 4.35v_a$.



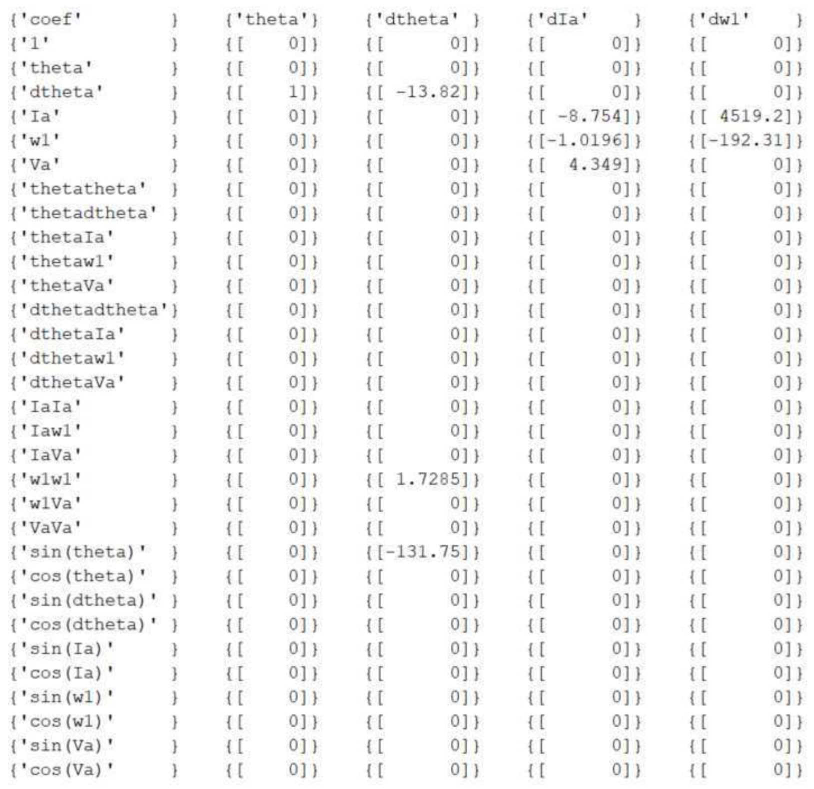
```
{'coef'         }  {'dtheta'}  {'ddtheta'}  {'dIa'    }  {'dw1'    }
{'1'            }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'theta'        }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'dtheta'       }  {[      1]}  {[ -13.82]}  {[      0]}  {[      0]}
{'Ia'           }  {[      0]}  {[      0]}  {[-8.6957]}  {[ 4519.2]}
{'w2'           }  {[      0]}  {[      0]}  {[-1.0217]}  {[-192.31]}
{'Va'           }  {[      0]}  {[      0]}  {[ 4.3478]}  {[      0]}
{'thetatheta'   }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'thetadtheta'  }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'thetaIa'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'thetaw2'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'thetaVa'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'dthetadtheta' }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'dthetaIa'     }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'dthetaw2'     }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'dthetaVa'     }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'IaIa'         }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'Iaw2'         }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'IaVa'         }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'w2w2'         }  {[      0]}  {[ 1.7285]}  {[      0]}  {[      0]}
{'w2Va'         }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'VaVa'         }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'sin(theta)'   }  {[      0]}  {[-131.75]}  {[      0]}  {[      0]}
{'cos(theta)'   }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'sin(dtheta)'  }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'cos(dtheta)'  }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'sin(Ia)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'cos(Ia)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'sin(w2)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'cos(w2)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'sin(Va)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
{'cos(Va)'      }  {[      0]}  {[      0]}  {[      0]}  {[      0]}
```

Source: prepared by the author.

5.2.2 Prediction on the test set

We use the test set data to evaluate the generalization capabilities of the models identified using SINDy and WSINDy. Since the derivatives of the state variables can be represented as $\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi^T$, we construct the library function using the test set and multiply it by the sparse matrix simulation results estimated. Figure 5.5 depicts the predicted derivatives of the state variables by the SINDy and WSINDy sparse representations and the expected ground truth. One can notice that the derivatives were correctly predicted, with an MSE

Figure 5.4: The sparse matrix Ξ estimated using the integral formulation WSINDy. Each column represents the approximation of the state variables' derivatives, and each non-zero term activates a nonlinear dynamic that models the behavior. The reference analytical equations are $\dot{\theta} = \dot{\theta}$, $\ddot{\theta} = -13.82\dot{\theta} + 1.73\omega_2^2$, $\dot{i}_a = -8.7i_a - 1.02\omega_2 + 4.35v_a$.



```
{'coef'         }  {'theta'}  {'dtheta' }  {'dIa'    }  {'dw1'    }
{'1'            }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'theta'        }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'dtheta'       }  {[    1]}  {[ -13.82]}  {[       0]}  {[       0]}
{'Ia'           }  {[    0]}  {[       0]}  {[ -8.754]}  {[ 4519.2]}
{'w1'           }  {[    0]}  {[       0]}  {[-1.0196]}  {[-192.31]}
{'Va'           }  {[    0]}  {[       0]}  {[  4.349]}  {[       0]}
{'thetatheta'   }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'thetadtheta'  }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'thetaIa'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'thetaw1'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'thetaVa'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'dthetadtheta'}   {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'dthetaIa'     }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'dthetaw1'     }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'dthetaVa'     }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'IaIa'         }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'Iaw1'         }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'IaVa'         }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'w1w1'         }  {[    0]}  {[ 1.7285]}  {[       0]}  {[       0]}
{'w1Va'         }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'VaVa'         }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'sin(theta)'   }  {[    0]}  {[-131.75]}  {[       0]}  {[       0]}
{'cos(theta)'   }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'sin(dtheta)'  }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'cos(dtheta)'  }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'sin(Ia)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'cos(Ia)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'sin(w1)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'cos(w1)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'sin(Va)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
{'cos(Va)'      }  {[    0]}  {[       0]}  {[       0]}  {[       0]}
```

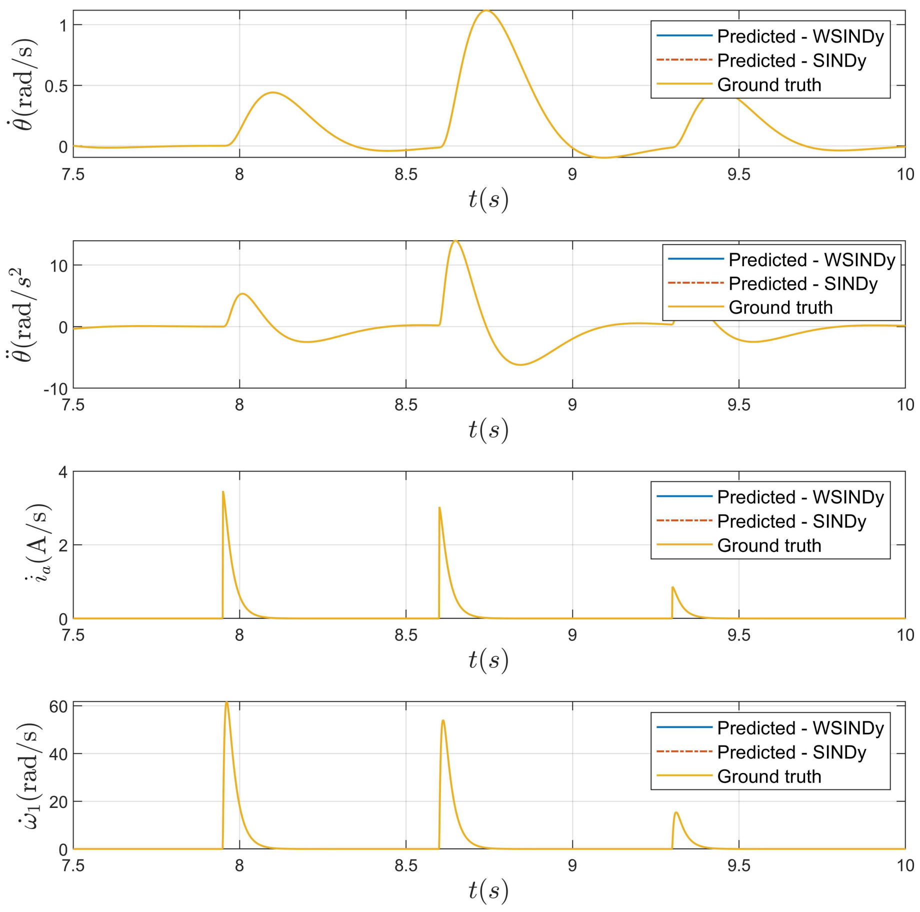
Source: prepared by the author.

of 10e-25 for the SINDy and 6e-10 for the WSINDy.

5.3 Sparse system identification under noisy measurements

Most of the time, the derivatives of the system must be computed since they cannot be measured, and the major problem of measurement data is the presence of noise. We add noise to the measurement data so that the SNR (Signal-to-noise-ratio) is 50 dB, thus making the scenario closer to reality.

Figure 5.5: The time derivatives $\dot{\theta}$, $\ddot{\theta}$, $\dot{i}_a$, and $\dot{\omega}_1$ as predicted by using the sparse matrix Ξ from the SINDy and WSINDy algorithms on the test set and the noise-free ground truth derivatives.



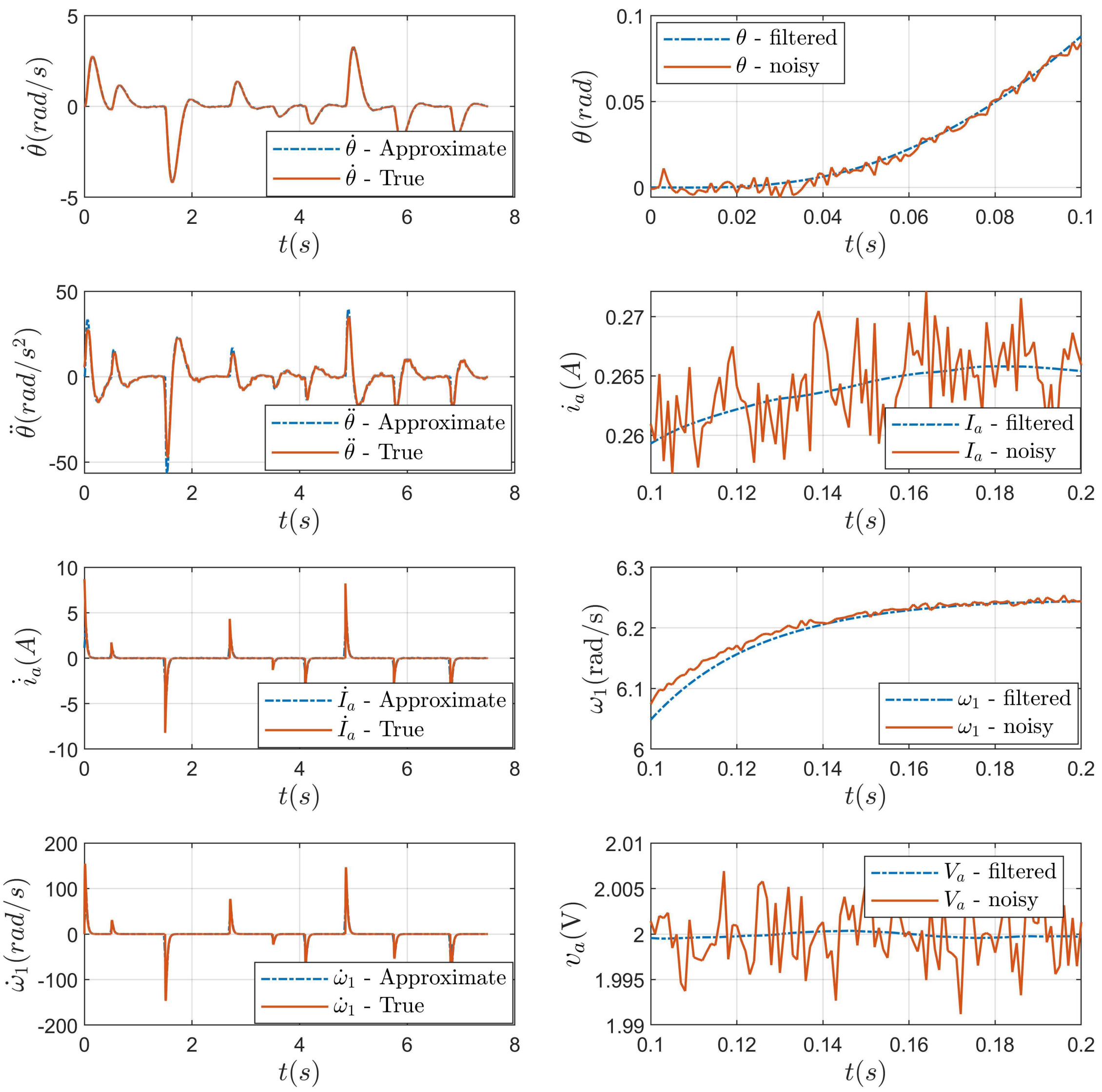
Source: prepared by the author.

5.3.1 Data pre-processing and derivatives computation

As Figure 5.6 depicts, the original state variables from simulation data were noise-contaminated. The noisy variables obtained from the simulation were filtered using a moving average filter with a window length of 40 samples. On the right side of the figure, we see the comparison between the noisy signals and the filtered ones. We can see that the input signals were properly filtered.

The derivatives were computed using the TVRegDiff algorithm. According to the

Figure 5.6: On the right side, the noise-contaminated variables θ, i_a, ω_1, and v_a and the corresponding filtered variables. The computed derivatives using the TVRegDiff algorithm are compared to the noise-free ground truth on the left side.



Source: prepared by the author.

variance-bias tradeoff, we chose the regularization factor empirically, with the regularization parameter varying from $\alpha = 0.001$ to $\alpha = 0.00005$. We compare the approximate derivatives and the analytical ground truth on the left side of Figure 5.6. Due to the bias-variance tradeoff, it is noticeable that the approximate angular acceleration $\ddot{\theta}$ had some of its peak values attenuated, and it is also visible when it comes to the derivative of the armature current I_a. Nonetheless, the other approximate derivatives present adherence to their respective ground truths.

5.3.2 Identifying and predicting the state variable's derivatives of the Aeropendulum - SINDy

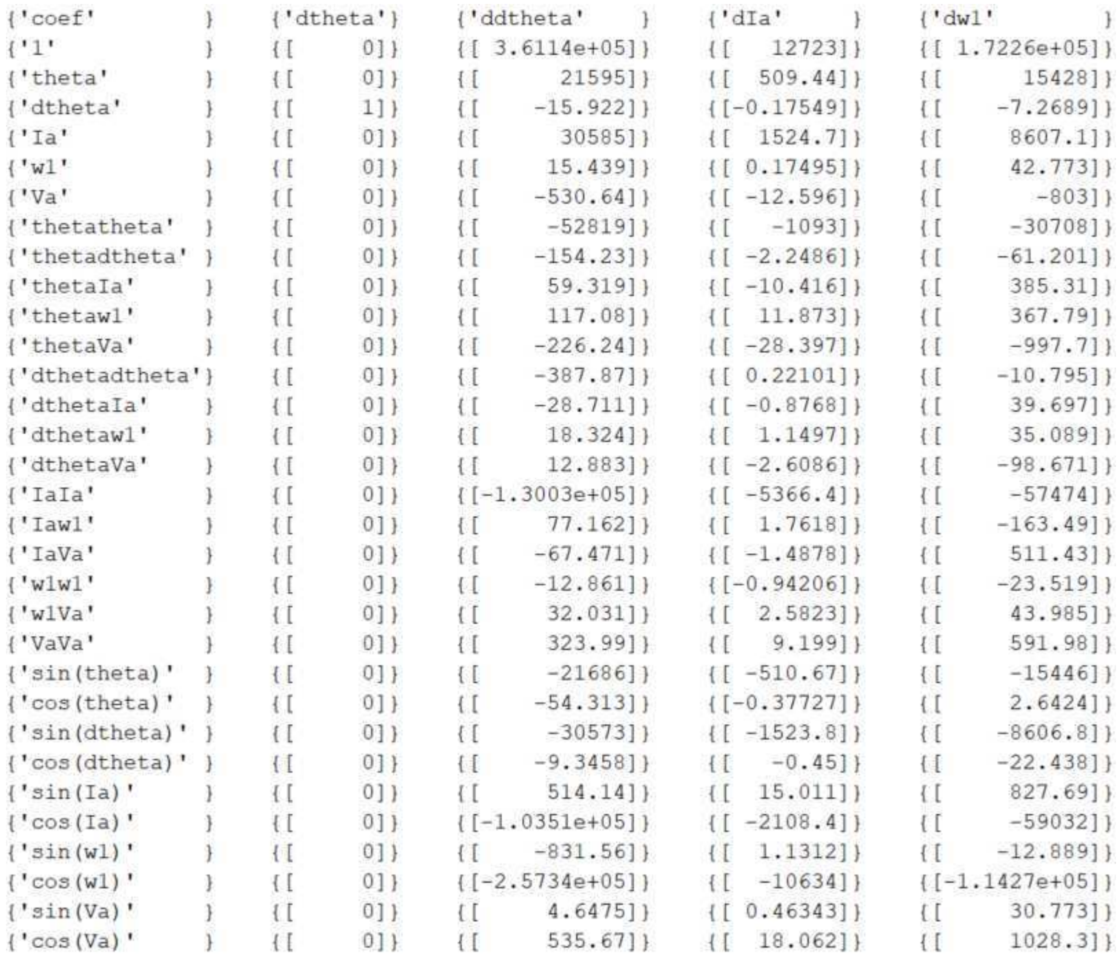
Figure 5.7 depicts the matrix simulation results estimated by solving the STLS problem. The matrix is not sparse, and the terms of interest do not correspond to the expected values. Even though noise mitigation strategies were applied, they might have led to a mischaracterization of the derivatives. Since the SINDy algorithm is extremely dependent on the data quality, it is probably the source of the non-sparse identified derivatives.

Once the representation of the dynamics Ξ is simulation results estimated, we can use it to predict the behavior on the test set. The 25% of the simulation results is used to compose the $\frac{dX_{test}}{dt} = \begin{bmatrix} \dot{\theta} & \ddot{\theta} & \dot{\omega}_1 & \dot{i}_a \end{bmatrix}$, and the library function $\Theta(X_{test})$. Figure 5.8 depicts the expected and predicted derivatives. It is noticeable that the representation poorly describes the angular acceleration $\ddot{\theta}$, with MSE$(\ddot{\theta}, \hat{\ddot{\theta}}) = 3.7$. The prediction of the rotor's speed and armature current derivatives is also deficient, with $\dot{\omega}_1$. MSE$(\dot{\omega}_1, \hat{\dot{\omega}_1}) = 3.7$ and MSE$(\dot{i}_a, \hat{\dot{i}}) = 3.7$.

5.3.3 Identifying and predicting the state variables' derivatives of the Aeropendulum - WSINDy

When using the WSINDy algorithm to determine the hidden dynamics, the difference is that only the $\dot{\theta}$ must be approximated due to the integral formulation. It is worth mentioning that, to our knowledge, until the present moment, previous works did not apply the combination of these methods. We use the same filtered data for the angular position to compose the library function $\Theta(\mathbf{X})$. Defining $\lambda = 0.9$, and $\gamma = 0.0014$, we find the sparse matrix depicted in Figure 5.9. We notice that it is not as sparse as it should be. However, the terms that describe the derivatives are close to what they should be, especially looking at the angular acceleration representation and the armature current derivative. The hypothesis

Figure 5.7: The sparse matrix Ξ estimated using the SINDy algorithm and the approximate derivatives from filtered noisy data. Each column represents the approximation of the state variables' derivatives, and the coefficients are the weights that activate each candidate nonlinear function from the library. The reference analytical equations are $\dot{\theta} = \dot{\theta}$, $\ddot{\theta} = -13.82\dot{\theta} + 1.73\omega_2^2$, $\dot{i}_a = -8.7i_a - 1.02\omega_2 + 4.35v_a$.

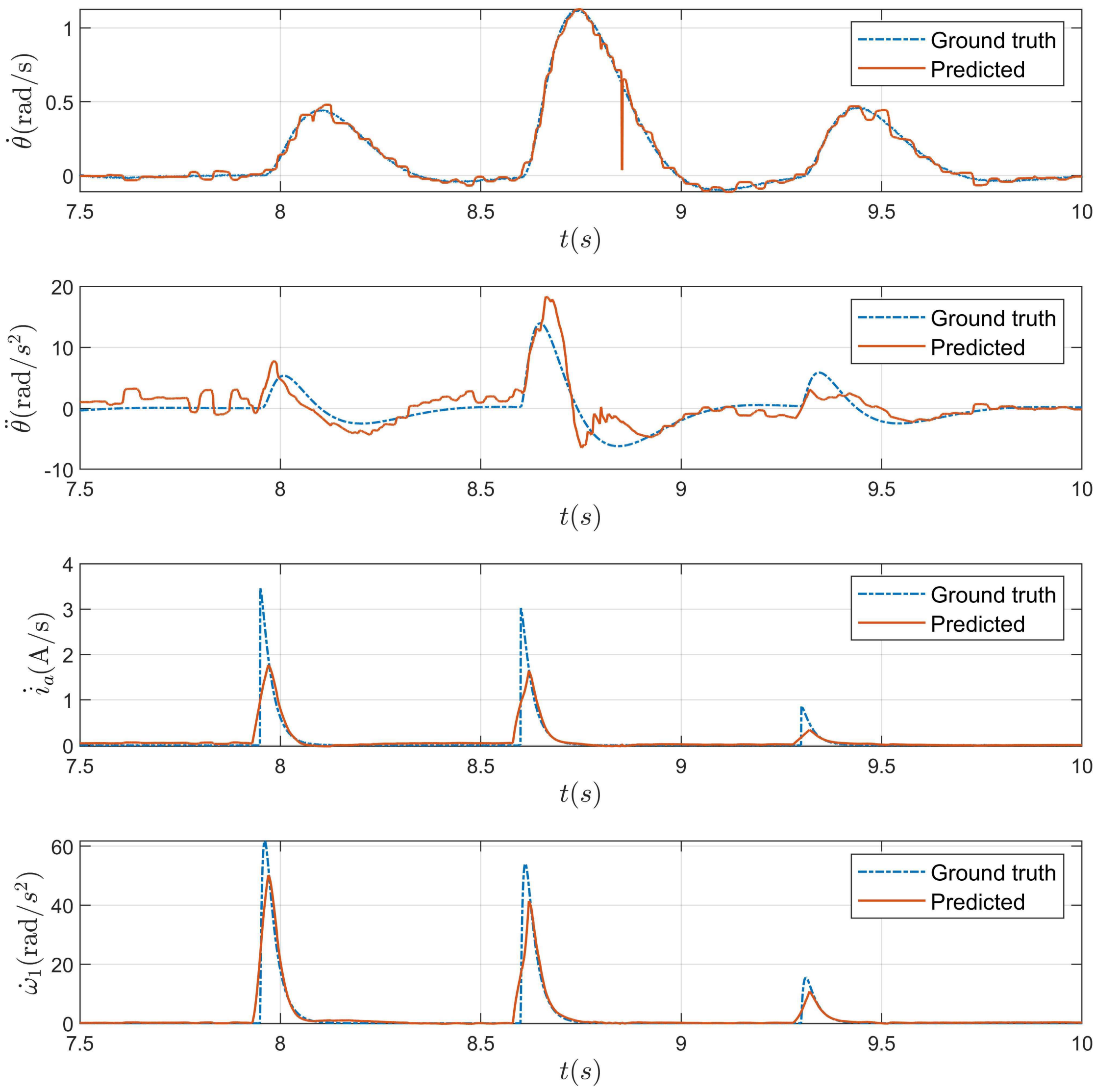

```
{'coef'         }   {'dtheta'}   {'ddtheta'     }   {'dIa'       }   {'dw1'          }
{'1'            }   {[      0]}  {[ 3.6114e+05]}    {[    12723]}    {[ 1.7226e+05]}
{'theta'        }   {[      0]}  {[      21595]}    {[   509.44]}    {[      15428]}
{'dtheta'       }   {[      1]}  {[    -15.922]}    {[-0.17549]}     {[    -7.2689]}
{'Ia'           }   {[      0]}  {[      30585]}    {[   1524.7]}    {[     8607.1]}
{'w1'           }   {[      0]}  {[     15.439]}    {[ 0.17495]}     {[     42.773]}
{'Va'           }   {[      0]}  {[    -530.64]}    {[ -12.596]}     {[       -803]}
{'thetatheta'   }   {[      0]}  {[     -52819]}    {[    -1093]}    {[     -30708]}
{'thetadtheta'  }   {[      0]}  {[    -154.23]}    {[ -2.2486]}     {[    -61.201]}
{'thetaIa'      }   {[      0]}  {[     59.319]}    {[ -10.416]}     {[     385.31]}
{'thetaw1'      }   {[      0]}  {[     117.08]}    {[  11.873]}     {[     367.79]}
{'thetaVa'      }   {[      0]}  {[    -226.24]}    {[ -28.397]}     {[     -997.7]}
{'dthetadtheta'}    {[      0]}  {[    -387.87]}    {[ 0.22101]}     {[    -10.795]}
{'dthetaIa'     }   {[      0]}  {[    -28.711]}    {[ -0.8768]}     {[     39.697]}
{'dthetaw1'     }   {[      0]}  {[     18.324]}    {[  1.1497]}     {[     35.089]}
{'dthetaVa'     }   {[      0]}  {[     12.883]}    {[ -2.6086]}     {[    -98.671]}
{'IaIa'         }   {[      0]}  {[-1.3003e+05]}    {[ -5366.4]}     {[     -57474]}
{'Iaw1'         }   {[      0]}  {[     77.162]}    {[  1.7618]}     {[    -163.49]}
{'IaVa'         }   {[      0]}  {[    -67.471]}    {[ -1.4878]}     {[     511.43]}
{'w1w1'         }   {[      0]}  {[    -12.861]}    {[-0.94206]}     {[    -23.519]}
{'w1Va'         }   {[      0]}  {[     32.031]}    {[  2.5823]}     {[     43.985]}
{'VaVa'         }   {[      0]}  {[     323.99]}    {[   9.199]}     {[     591.98]}
{'sin(theta)'   }   {[      0]}  {[     -21686]}    {[ -510.67]}     {[     -15446]}
{'cos(theta)'   }   {[      0]}  {[    -54.313]}    {[-0.37727]}     {[     2.6424]}
{'sin(dtheta)'  }   {[      0]}  {[     -30573]}    {[ -1523.8]}     {[    -8606.8]}
{'cos(dtheta)'  }   {[      0]}  {[    -9.3458]}    {[   -0.45]}     {[    -22.438]}
{'sin(Ia)'      }   {[      0]}  {[     514.14]}    {[  15.011]}     {[     827.69]}
{'cos(Ia)'      }   {[      0]}  {[-1.0351e+05]}    {[ -2108.4]}     {[     -59032]}
{'sin(w1)'      }   {[      0]}  {[    -831.56]}    {[  1.1312]}     {[    -12.889]}
{'cos(w1)'      }   {[      0]}  {[-2.5734e+05]}    {[  -10634]}     {[-1.1427e+05]}
{'sin(Va)'      }   {[      0]}  {[     4.6475]}    {[ 0.46343]}     {[     30.773]}
{'cos(Va)'      }   {[      0]}  {[     535.67]}    {[  18.062]}     {[     1028.3]}
```

Source: prepared by the author.

is that the filtering should be improved not to include unmodeled dynamics. When talking specifically of the propeller's thrust coefficient, using the term that accompanies ω_1^2, $K_q = \frac{2.8}{\frac{L}{J}(\frac{N_1}{N_2})^2} = 0.00017$, which is 70% higher than the expected coefficient.

Figure 5.10 shows the predicted derivatives using the sparse matrix determined by the WSINDy algorithm. Compared to the SINDy algorithm predictions, it is perceptible that it improved the angular acceleration predictions and the armature current derivative. $\text{MSE}(\ddot{\theta}, \hat{\ddot{\theta}}) = 0.18$. The prediction of the armature current derivative improved, having $\text{MSE}(\dot{i}_a, \hat{\dot{i}_a}) = 0.03$. From the performance on the test set, we can say that even though the

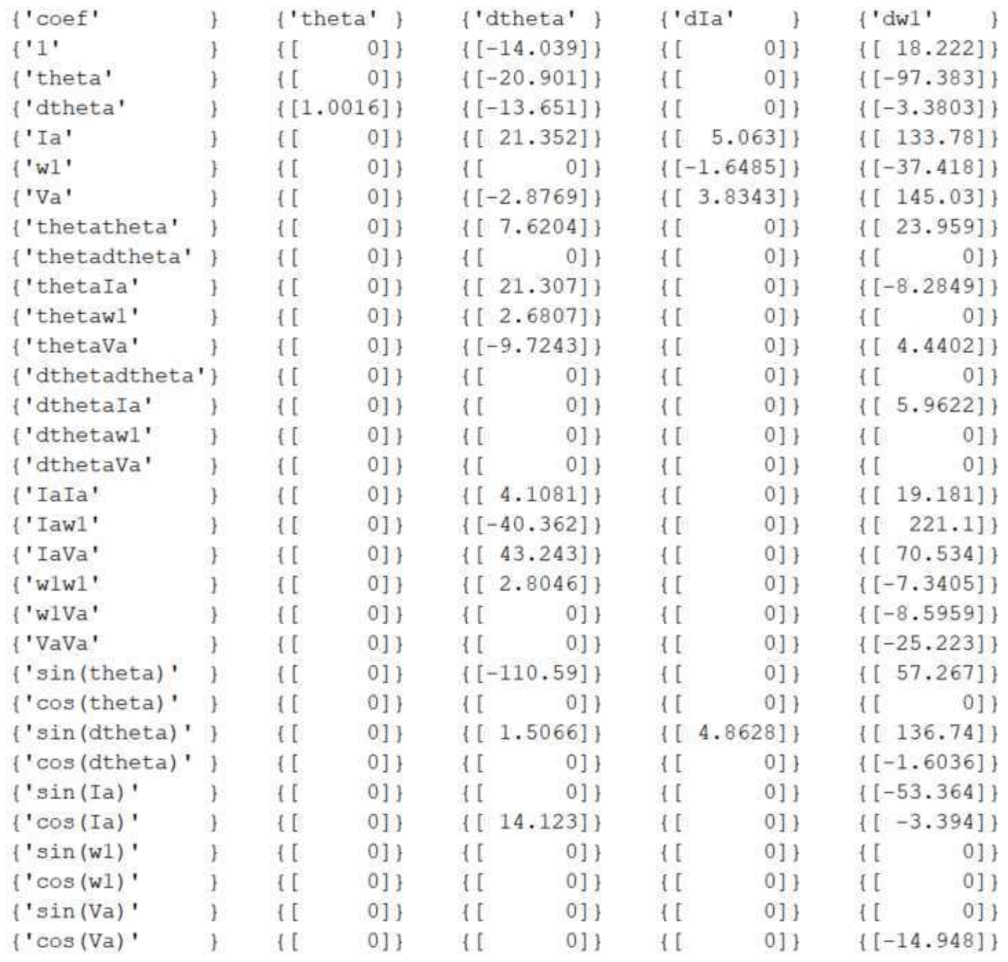
Figure 5.8: The time derivatives $\dot{\theta}$, $\ddot{\theta}$, $\dot{i}_a$, and $\dot{\omega}_1$ as predicted by using the sparse matrix Ξ from the SINDy algorithm on the test set and the noise-free ground truth derivatives.



Source: prepared by the author.

discovered dynamics lacked physical explainability, the generalization capabilities for the angular acceleration and electrical dynamics indicate that it is a homolog representation of the system data. Further hyperparameter adjustments should be made to deal with the noisy measurements and their effect on the identification procedures.

Figure 5.9: The sparse matrix Ξ estimated using the WSINDy algorithm and the approximate derivatives from filtered noisy data. Each column represents the approximation of the state variables' derivatives, and the non-zero coefficients represent the weighing factors that activate the candidate nonlinear dynamics. The reference analytical equations are $\dot{\theta} = \dot{\theta}$, $\ddot{\theta} = -13.82\dot{\theta} + 1.73\omega_2^2$, $\dot{i}_a = -8.7i_a - 1.02\omega_2 + 4.35v_a$.



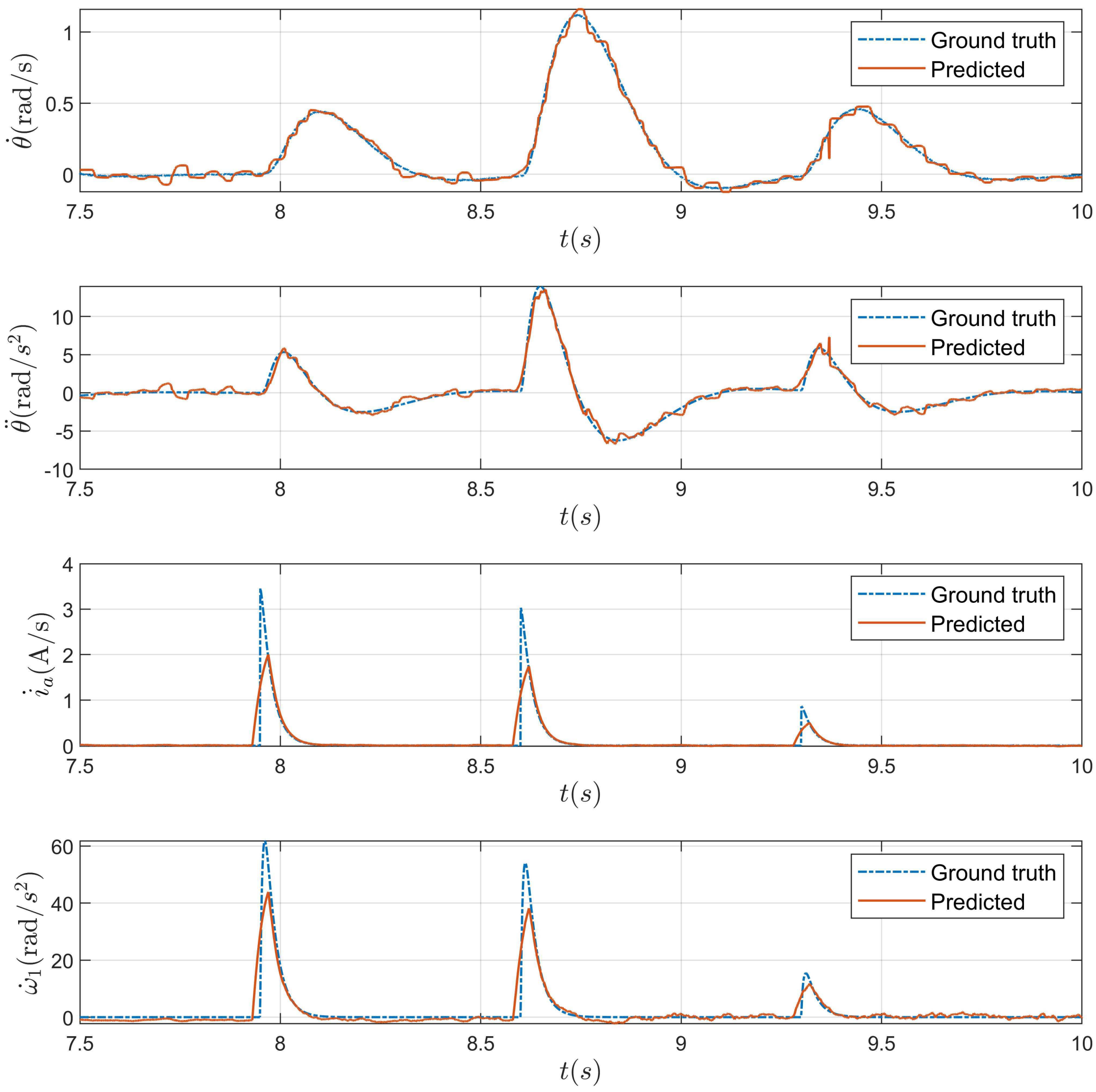
```
{'coef'         }    {'theta' }    {'dtheta' }    {'dIa'     }    {'dw1'     }
{'1'            }    {[      0]}   {[-14.039]}    {[       0]}    {[ 18.222]}
{'theta'        }    {[      0]}   {[-20.901]}    {[       0]}    {[-97.383]}
{'dtheta'       }    {[1.0016]}    {[-13.651]}    {[       0]}    {[-3.3803]}
{'Ia'           }    {[      0]}   {[ 21.352]}    {[  5.063]}    {[ 133.78]}
{'w1'           }    {[      0]}   {[       0]}   {[-1.6485]}    {[-37.418]}
{'Va'           }    {[      0]}   {[-2.8769]}    {[ 3.8343]}    {[ 145.03]}
{'thetatheta'   }    {[      0]}   {[ 7.6204]}    {[       0]}    {[ 23.959]}
{'thetadtheta'  }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'thetaIa'      }    {[      0]}   {[ 21.307]}    {[       0]}    {[-8.2849]}
{'thetaw1'      }    {[      0]}   {[ 2.6807]}    {[       0]}    {[      0]}
{'thetaVa'      }    {[      0]}   {[-9.7243]}    {[       0]}    {[ 4.4402]}
{'dthetadtheta' }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'dthetaIa'     }    {[      0]}   {[       0]}   {[       0]}    {[ 5.9622]}
{'dthetaw1'     }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'dthetaVa'     }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'IaIa'         }    {[      0]}   {[ 4.1081]}    {[       0]}    {[ 19.181]}
{'Iaw1'         }    {[      0]}   {[-40.362]}    {[       0]}    {[  221.1]}
{'IaVa'         }    {[      0]}   {[ 43.243]}    {[       0]}    {[ 70.534]}
{'w1w1'         }    {[      0]}   {[ 2.8046]}    {[       0]}    {[-7.3405]}
{'w1Va'         }    {[      0]}   {[       0]}   {[       0]}    {[-8.5959]}
{'VaVa'         }    {[      0]}   {[       0]}   {[       0]}    {[-25.223]}
{'sin(theta)'   }    {[      0]}   {[-110.59]}    {[       0]}    {[ 57.267]}
{'cos(theta)'   }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'sin(dtheta)'  }    {[      0]}   {[ 1.5066]}    {[ 4.8628]}    {[ 136.74]}
{'cos(dtheta)'  }    {[      0]}   {[       0]}   {[       0]}    {[-1.6036]}
{'sin(Ia)'      }    {[      0]}   {[       0]}   {[       0]}    {[-53.364]}
{'cos(Ia)'      }    {[      0]}   {[ 14.123]}    {[       0]}    {[ -3.394]}
{'sin(w1)'      }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'cos(w1)'      }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'sin(Va)'      }    {[      0]}   {[       0]}   {[       0]}    {[      0]}
{'cos(Va)'      }    {[      0]}   {[       0]}   {[       0]}    {[-14.948]}
```

Source: prepared by the author.

5.4 Discrepancy SINDy and WSINDy - finding mismatches in the model

We now consider a model mismatch in the angular acceleration, specifically on the propeller's thrust coefficient and the pendulum's mass. Hence, the approximate model is $\hat{\ddot{\theta}} = -131.5\sin(\psi) - 13.82\dot{\theta} + 1.65\omega_1^2$, instead of $\ddot{\theta} = -131.75\sin(\psi) - 13.82\dot{\theta} + 1.73\omega_1^2$. Figure 5.11 displays the phase diagram that evidences the difference between the approximate and reference derivatives.

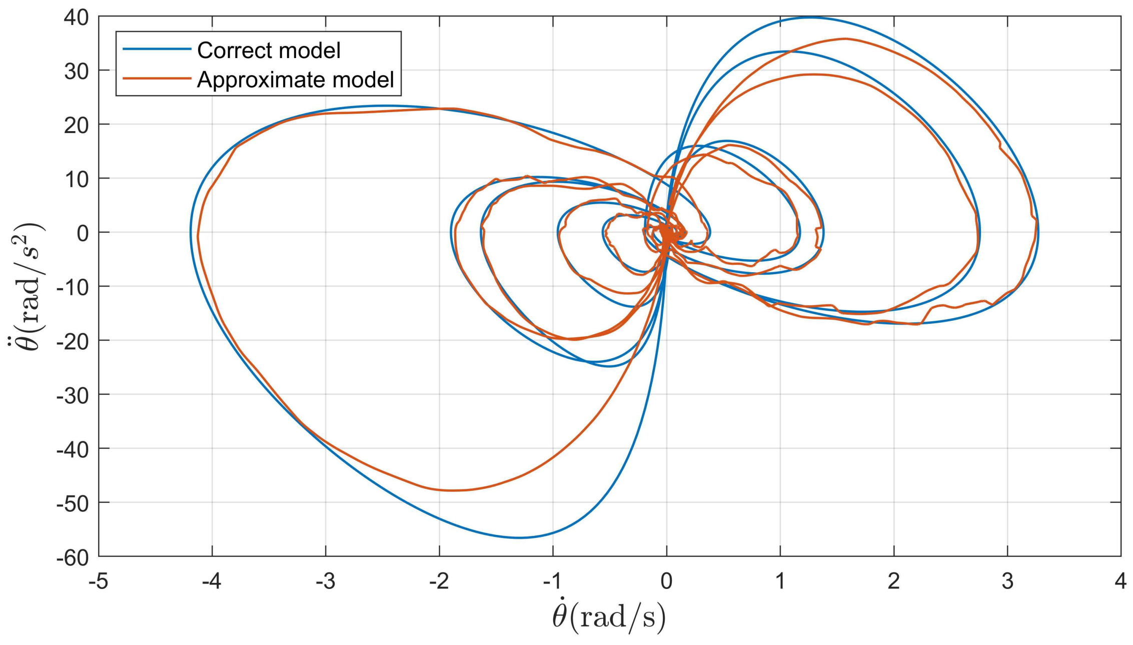
Figure 5.10: The state variables' derivatives $\dot{\theta}$, $\ddot{\theta}$, $\dot{i}_a$, and $\dot{\omega}_1$ predicted by the sparse matrix Ξ from the WSINDy algorithm on the test set and the noise-free ground truth derivatives.



Source: prepared by the author.

The same previous variables are collected from the simulation, and we construct the library function and the difference between the approximate and exact dynamics $\delta\dot{\mathbf{X}}$ using the SINDy algorithm. For the WSINDy algorithm, we integrate the approximate derivatives and compute the difference between the trajectories $\delta\mathbf{X}$. Figure 5.12 depicts the difference between the behavior of each derivative. The rest of the terms are null since there is only a discrepancy in the angular acceleration modeling.

Figure 5.11: The phase diagram of both the approximate and correct model.

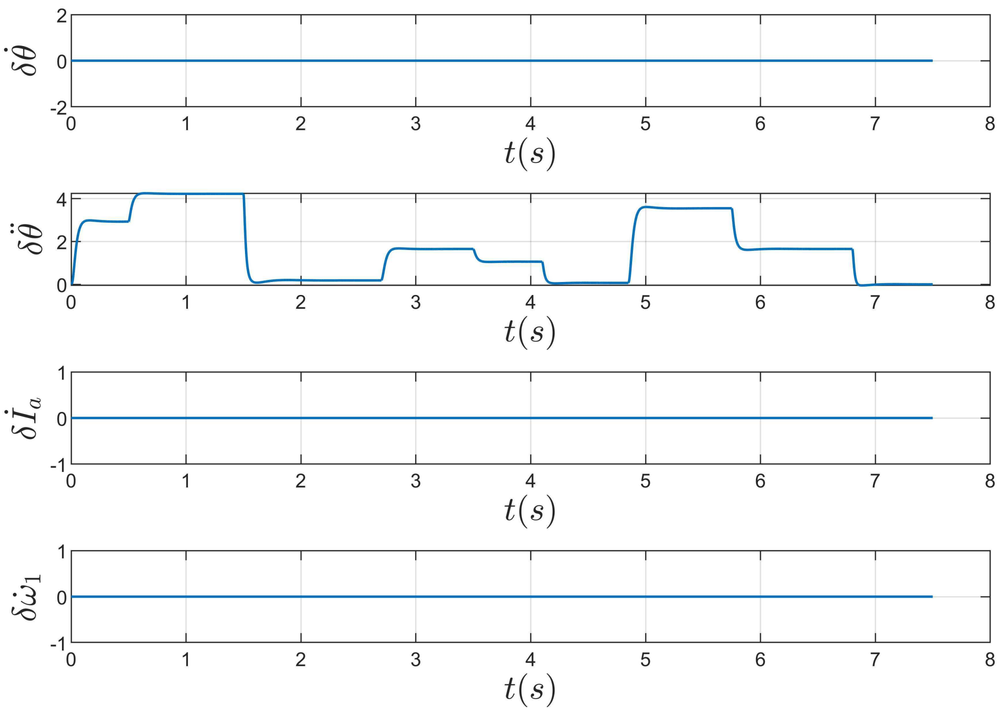


Source: prepared by the author.

Figure 5.13 shows the sparse matrix representing the parameter difference between the two models simulation results estimated by solving the STLS problem using the SINDy algorithm. The solution is sparse and only points out differences in the modeling of the angular acceleration, as expected. The rest of the dynamics do not need any change. On the other hand, Figure 5.14 depicts the sparse matrix simulation results estimated using the WSINDy algorithm. The reason for using the integral version of the algorithm is to evaluate the feasibility of the integral formulation for the discrepancy SINDy. The same correction terms were simulation results estimated for the angular acceleration. Nonetheless, smaller coefficients were incorrectly identified in the other dynamics, such as in the angular velocity and the derivative of the armature current. They should have a small impact in correcting the approximate derivatives.

We can correct the approximate model and predict the test set using the sparse matrix representing the discrepancy. Figure 5.15 shows the discrepant model and the corrected predictions using the SINDy and WSINDy formulations. Both corrected models using SINDy

Figure 5.12: The difference between the derivatives of the correct model and the approximate one. The discrepancy is not null for the angular acceleration $\ddot{\theta}$ since the model inaccuracy is only present in it.



Source: prepared by the author.

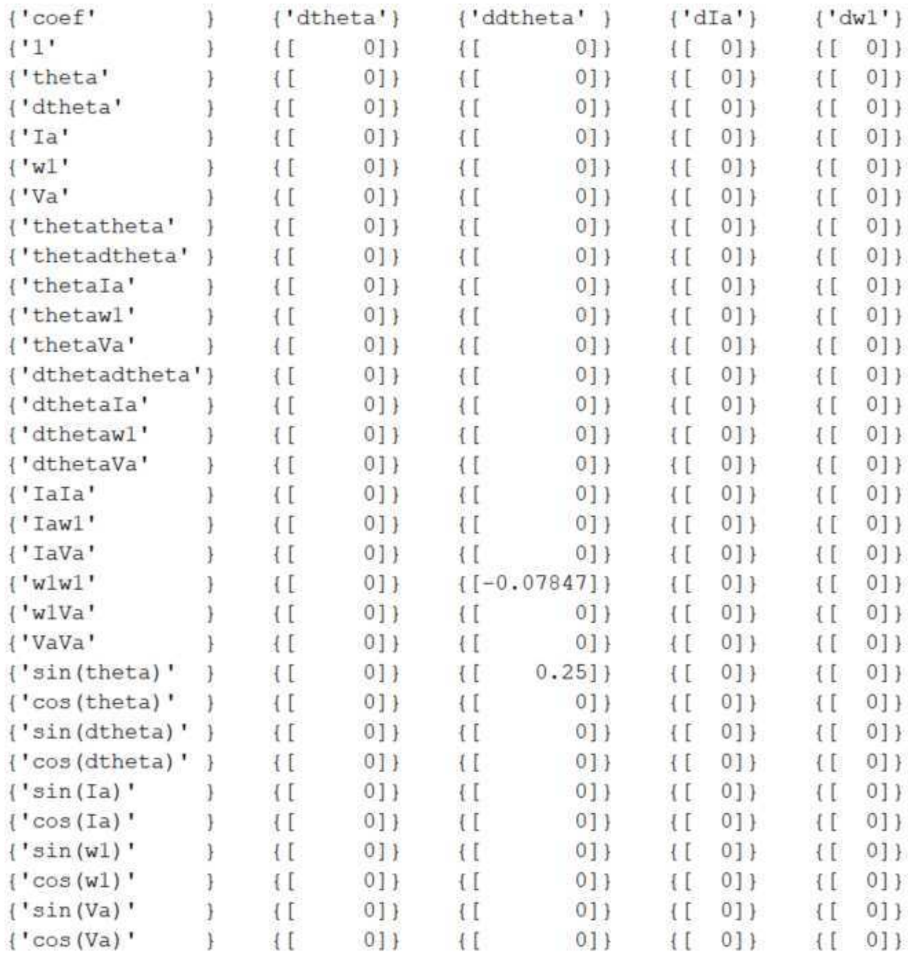
and WSINDy could correctly predict the derivatives. The SINDy and WSINDy resulted in MSEs of 2.55×10^{-29} and 1.7410×10^{-12}, respectively.

5.5 Experimental platform identification

Now, we want to collect the data from the experimental platform and try to characterize the behavior of the aerodynamic part using the WSINDy algorithm. As detailed in the previous sections, the approximate derivatives taken from noisy data are prone to mischaracterization, and the SINDy algorithm needs to perform satisfactorily in this case, which naturally leads to the integral formulation.

More specifically, we want to evaluate the capabilities of the WSINDy to deal with the noise sources associated with the experimental platform and describe the propeller's

Figure 5.13: The discrepancy sparse matrix Ξ estimated using the SINDy algorithm. Each column represents the approximation of the state variables' derivatives, and the non-zero coefficients represent the weighing factors that should be added to the approximate model and activate the candidate nonlinear functions.



```
{'coef'          }    {'dtheta'}    {'ddtheta' }    {'dIa'}    {'dw1'}
{'1'             }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'theta'         }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'dtheta'        }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'Ia'            }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'w1'            }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'Va'            }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'thetatheta'    }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'thetadtheta'   }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'thetaIa'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'thetaw1'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'thetaVa'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'dthetadtheta'}      {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'dthetaIa'      }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'dthetaw1'      }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'dthetaVa'      }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'IaIa'          }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'Iaw1'          }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'IaVa'          }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'w1w1'          }    {[      0]}    {[-0.07847]}    {[  0]}    {[  0]}
{'w1Va'          }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'VaVa'          }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'sin(theta)'    }    {[      0]}    {[     0.25]}    {[  0]}    {[  0]}
{'cos(theta)'    }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'sin(dtheta)'   }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'cos(dtheta)'   }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'sin(Ia)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'cos(Ia)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'sin(w1)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'cos(w1)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'sin(Va)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
{'cos(Va)'       }    {[      0]}    {[        0]}    {[  0]}    {[  0]}
```

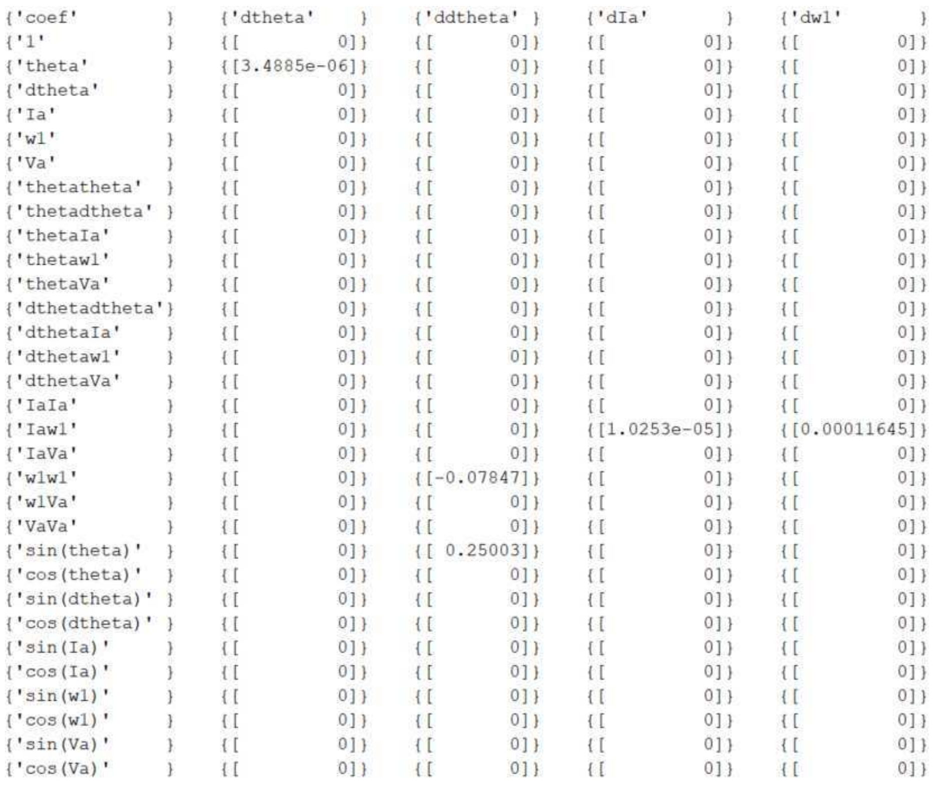
Source: prepared by the author.

characteristics. In addition, we want to explore different candidate functions and see if they can better describe the thrust force as a function of the propeller's speed.

5.5.1 Data collection and preprocessing

We obtained the raw signals V_a, θ, I_a, and ω_1 using the PWM-based acquisition scheme. As Figure 5.16 depicts on its right side, the state variables were collected and filtered. The approximate derivatives are posteriorly computed on the left side using the TVRegDiff algorithm. It is noticeable that most of the spikes in the collected data are related to measurement noise. Due to the open loop operation, the rotor's speed varies, and the

Figure 5.14: The discrepancy sparse matrix Ξ estimated using the WSINDy algorithm. Each column represents the approximation of the state variables' derivatives, and the non-zero coefficients represent the weighing factors that should be added to the approximate model and activate the candidate nonlinear functions. Some non-zero spurious terms are identified.



```
{'coef'        }   {'dtheta'     }   {'ddtheta' }   {'dIa'        }   {'dw1'        }
{'1'           }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'theta'       }   {[3.4885e-06]}   {[       0]}   {[           0]}   {[           0]}
{'dtheta'      }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'Ia'          }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'w1'          }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'Va'          }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'thetatheta'  }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'thetadtheta' }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'thetaIa'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'thetaw1'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'thetaVa'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'dthetadtheta'}   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'dthetaIa'    }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'dthetaw1'    }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'dthetaVa'    }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'IaIa'        }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'Iaw1'        }   {[           0]}   {[       0]}   {[1.0253e-05]}   {[0.00011645]}
{'IaVa'        }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'w1w1'        }   {[           0]}   {[-0.07847]}   {[           0]}   {[           0]}
{'w1Va'        }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'VaVa'        }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'sin(theta)'  }   {[           0]}   {[ 0.25003]}   {[           0]}   {[           0]}
{'cos(theta)'  }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'sin(dtheta)' }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'cos(dtheta)' }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'sin(Ia)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'cos(Ia)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'sin(w1)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'cos(w1)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'sin(Va)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
{'cos(Va)'     }   {[           0]}   {[       0]}   {[           0]}   {[           0]}
```

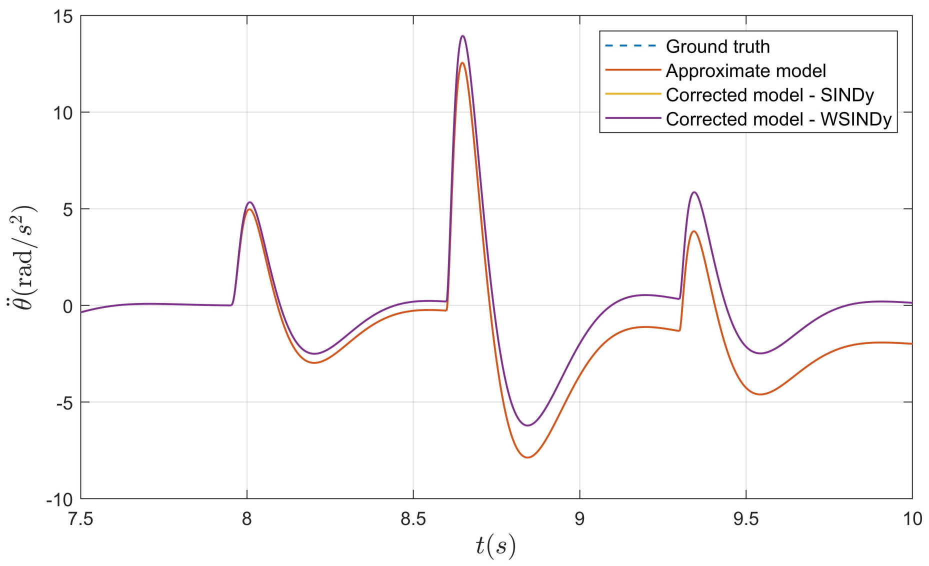
Source: prepared by the author.

system suffers environmental influences, leading to undesired angular displacements and smaller spikes in the armature current and angular speed.

5.5.2 Identification using WSINDy and prediction on the test set

We conducted a hyperparameter tuning to find the sparsification and regularization terms for the integral formulation of the sparsification algorithm. Figure 5.17 depicts the sparse matrix that represents the dynamics for the test set using the WSINDy algorithm. The solution is not sparse for the angular acceleration, the armature current derivative, and the rotor's speed derivative. Most terms expected to be null, especially on the angular acceleration description, are spurious. Considering this representation, we can say that the model lacks physical explainability and cannot describe the real dynamics. Hence, we

Figure 5.15: Derivative $\ddot{\theta}$ predicted using both the SINDy and WSINDy Ξ matrices to correct the approximate models compared to the ground truth signal.



Source: prepared by the author.

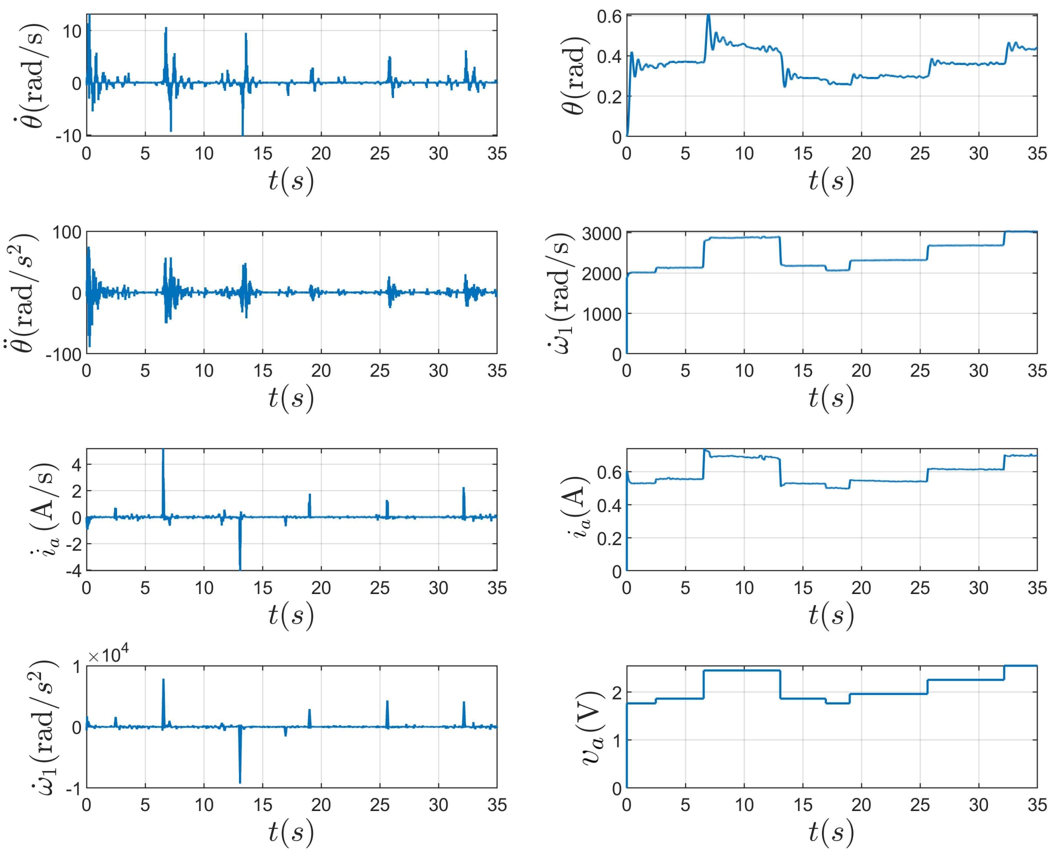
could not find satisfactory results to describe the thrust effects regarding the propeller's characterization.

We used the sparse matrix estimated to know if the model could still sufficiently generalize on the test set. Figure 5.18 depicts the derivatives predicted. We notice, especially regarding the prediction of $\ddot{\theta}$, that the predicted behavior draws apart the ground truth behavior. It also happens with $\dot{i}_a$ and $\dot{\omega}_1$.

5.6 General discussion about the results and possibility of using the WsINDy algorithm.

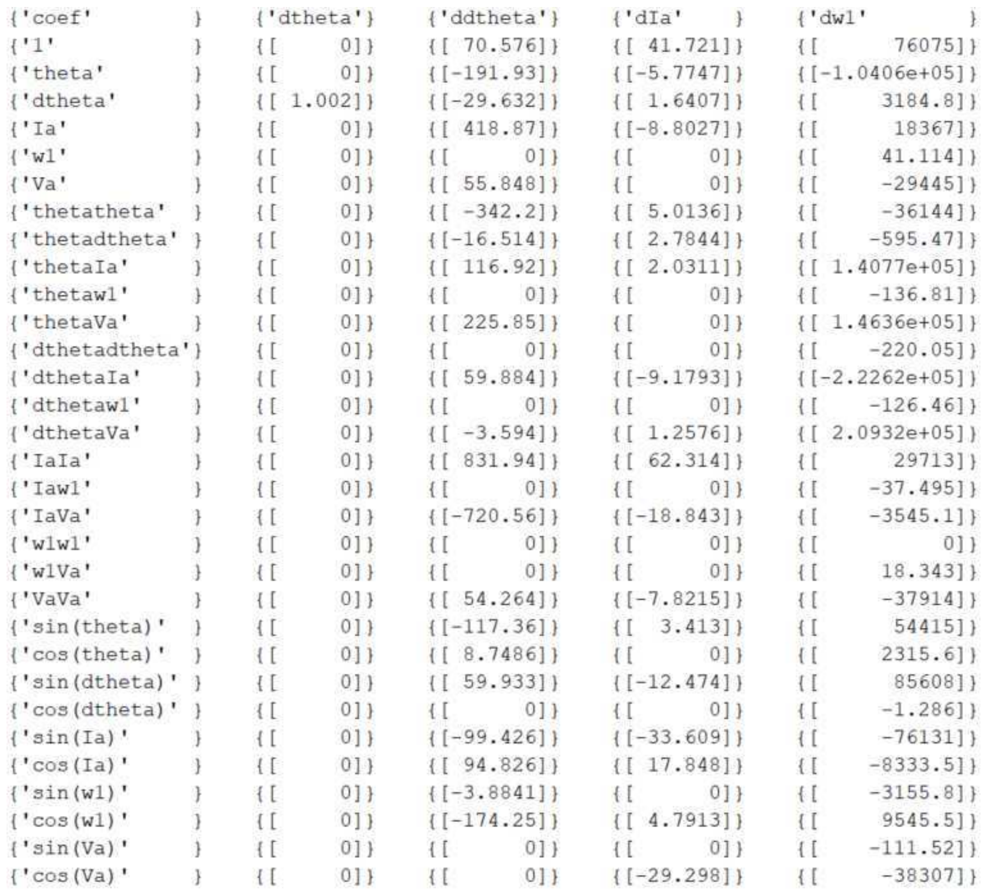
From the bibliographical review and motivating examples, we inferred that the WSINDy algorithm was more noise-robust when compared to the traditional SINDy. By addressing the problem using signal processing and regularized differentiation, we expected to leverage the identification capabilities of the algorithm by providing higher-quality data. Nonetheless, this did not happen. A closer look at the results for the WSINDy algorithm in the simulated example contaminated with noise and using the experimental data shows that it

Figure 5.16: On the right side, the filtered noise-contaminated variables θ, i_a, ω_1, and V_a. The computed derivatives using the TVRegDiff algorithm are compared to the noise-free ground truth on the left side.



Source: prepared by the author.

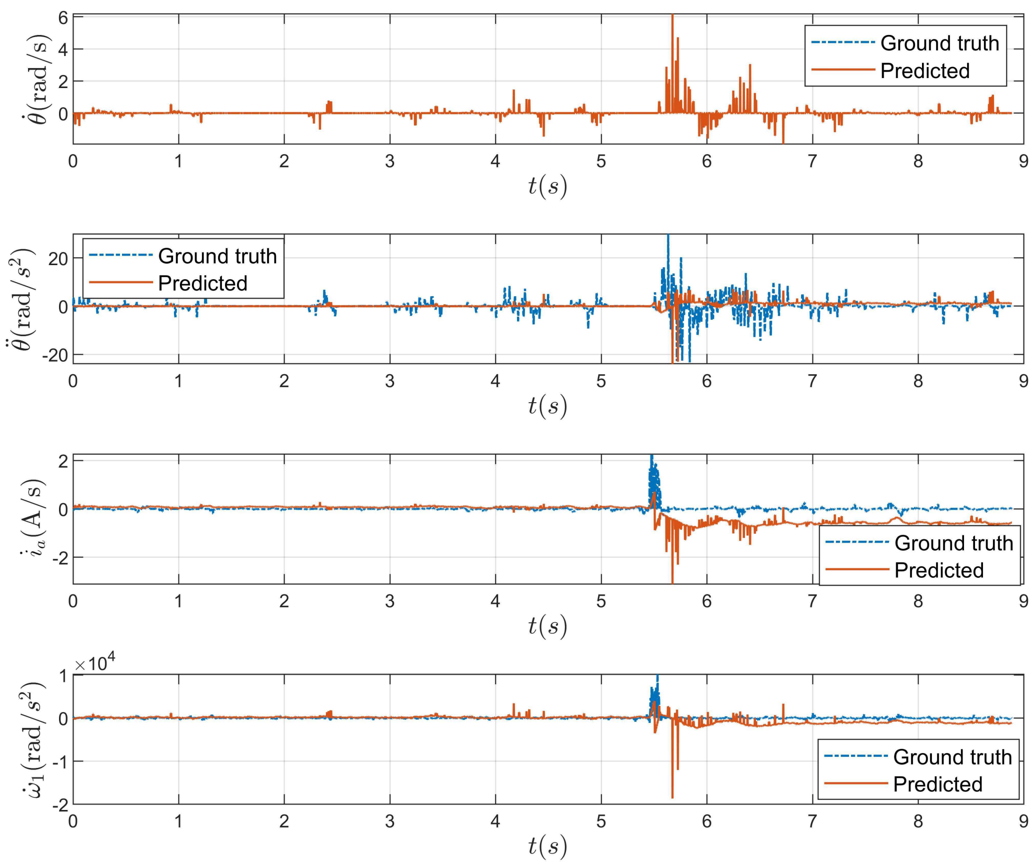
Figure 5.17: The sparse matrix Ξ estimated using the WSINDy algorithm and the approximate derivatives from the experimental platform data.



```
{'coef'         }    {'dtheta'}    {'ddtheta'}    {'dIa'     }    {'dw1'           }
{'1'            }    {[      0]}    {[ 70.576]}    {[ 41.721]}    {[        76075]}
{'theta'        }    {[      0]}    {[-191.93]}    {[-5.7747]}    {[-1.0406e+05]}
{'dtheta'       }    {[ 1.002]}    {[-29.632]}    {[ 1.6407]}    {[       3184.8]}
{'Ia'           }    {[      0]}    {[ 418.87]}    {[-8.8027]}    {[        18367]}
{'w1'           }    {[      0]}    {[      0]}    {[      0]}    {[       41.114]}
{'Va'           }    {[      0]}    {[ 55.848]}    {[      0]}    {[       -29445]}
{'thetatheta'   }    {[      0]}    {[ -342.2]}    {[ 5.0136]}    {[       -36144]}
{'thetadtheta'  }    {[      0]}    {[-16.514]}    {[ 2.7844]}    {[      -595.47]}
{'thetaIa'      }    {[      0]}    {[ 116.92]}    {[ 2.0311]}    {[ 1.4077e+05]}
{'thetaw1'      }    {[      0]}    {[      0]}    {[      0]}    {[      -136.81]}
{'thetaVa'      }    {[      0]}    {[ 225.85]}    {[      0]}    {[ 1.4636e+05]}
{'dthetadtheta'}    {[      0]}    {[      0]}    {[      0]}    {[      -220.05]}
{'dthetaIa'     }    {[      0]}    {[ 59.884]}    {[-9.1793]}    {[-2.2262e+05]}
{'dthetaw1'     }    {[      0]}    {[      0]}    {[      0]}    {[      -126.46]}
{'dthetaVa'     }    {[      0]}    {[ -3.594]}    {[ 1.2576]}    {[ 2.0932e+05]}
{'IaIa'         }    {[      0]}    {[ 831.94]}    {[ 62.314]}    {[        29713]}
{'Iaw1'         }    {[      0]}    {[      0]}    {[      0]}    {[      -37.495]}
{'IaVa'         }    {[      0]}    {[-720.56]}    {[-18.843]}    {[      -3545.1]}
{'w1w1'         }    {[      0]}    {[      0]}    {[      0]}    {[            0]}
{'w1Va'         }    {[      0]}    {[      0]}    {[      0]}    {[       18.343]}
{'VaVa'         }    {[      0]}    {[ 54.264]}    {[-7.8215]}    {[       -37914]}
{'sin(theta)'   }    {[      0]}    {[-117.36]}    {[  3.413]}    {[        54415]}
{'cos(theta)'   }    {[      0]}    {[ 8.7486]}    {[      0]}    {[       2315.6]}
{'sin(dtheta)'  }    {[      0]}    {[ 59.933]}    {[-12.474]}    {[        85608]}
{'cos(dtheta)'  }    {[      0]}    {[      0]}    {[      0]}    {[       -1.286]}
{'sin(Ia)'      }    {[      0]}    {[-99.426]}    {[-33.609]}    {[       -76131]}
{'cos(Ia)'      }    {[      0]}    {[ 94.826]}    {[ 17.848]}    {[      -8333.5]}
{'sin(w1)'      }    {[      0]}    {[-3.8841]}    {[      0]}    {[      -3155.8]}
{'cos(w1)'      }    {[      0]}    {[-174.25]}    {[ 4.7913]}    {[       9545.5]}
{'sin(Va)'      }    {[      0]}    {[      0]}    {[      0]}    {[      -111.52]}
{'cos(Va)'      }    {[      0]}    {[      0]}    {[-29.298]}    {[       -38307]}
```

Source: prepared by the author.

Figure 5.18: The derivatives $\dot{\theta}$, $\ddot{\theta}$, $\dot{i}_a$, and $\dot{\omega}_1$ for the experimental platform test set predicted by the sparse matrix Ξ from the WSINDy algorithm. The ground truth signals are the derivatives taken from the input test signals.



Source: prepared by the author.

could not correctly identify the hidden dynamics. The inaccuracy is more noticeable in the practical case, where not even the terms of interest in some dynamics could be identified.

Some considerations might been drawn. The first is regarding the preprocessing step. Even though a regularization parameter is used to control the bias-variance tradeoff, the signal has oscillations that might mischaracterize the dynamics when the approximate derivatives are computed, which requires a more careful optimization approach. The second one is related to the hyperparameter tuning. A vast combination of parameters could be set to tame the low-quality data, but it requires additional study time. The last one is regarding the actuator and plant themselves. The identification algorithm considers measurement noise due to sensor imprecision. However, in addition to the measurement noise originating from the sensors, steady-state oscillations occur on the experimental platform when a constant duty cycle is applied to the actuator input. These oscillations result in spikes on the derivatives that insert other nonlinear dynamics that might obscure the real ones.

From the results obtained using the experimental platform and the WSINDy algorithm, we can tell that the algorithm cannot correctly characterize the propeller's thrust coefficient. Many signal preprocessing strategies should be made before affirming that the algorithm is inappropriate to identify the dynamics. Moreover, the angular position data has low quality, which might have led to a cascade of mischaracterizations of the dynamics. These challenges regarding the identification of the aerodynamic behavior of the plant indicate that another strategy to model the thrust force should be carried out posteriorly.

5.7 Final considerations

This chapter detailed the input-signal design procedure, evaluating the SINDy and WSINDy algorithms in different scenarios and gaining confidence in operating the identification framework. The first was noise-free simulated data; the expected derivatives were correctly predicted. In the second one, when performing under noisy measurements, preprocessing steps were performed to mitigate the undesired identified terms and resulted in satisfactory results using the integral formulation. When we used experimental data, the results did not match the expectations. The WSINDy algorithm could not identify the hidden dynamics nor generalize on the test set. These results indicate that further investigations should be followed to evaluate the actuation and other strategies to model the thrust force. The next chapter will detail the control strategies and the data-driven model used to perform the feedback control of the platform.

Chapter 6

Control System Design

In this chapter, we define the time-domain requirements for the solution and evaluate different control approaches, improving the description of the previously electrical-aerodynamic unmodeled dynamics through simulated and experimental platforms.

6.1 Motivation for the improvements

Lucena, Luiz and Lima[5] reports a mismatch in the closed-loop performance. Through the revision of the design workflow, it was evident that there was no error in the controller's project but in the system representation. Previous works have tackled the control problem using different strategies but either neglected the electrical-aerodynamic conversion and unmodeled dynamics or made suppositions that compromised the closed-loop performance. By improving the actuator's description, we evaluate the most appropriate way to control the plant using different strategies, first in simulation and then in the experimental platform.

The simulated experiments and those using the experimental platform are in a public GitHub repository [1].

6.2 Time domain requirements specification

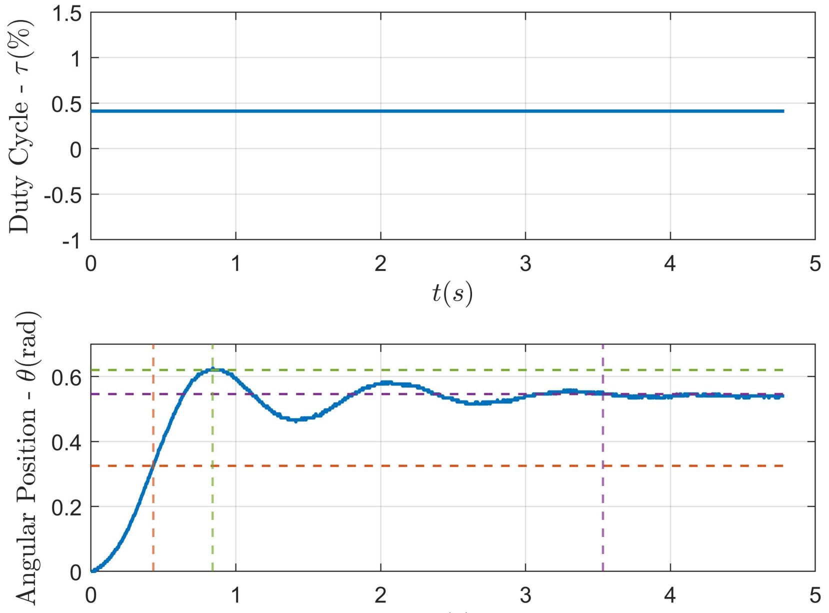
In the initial control design stages, knowing the physical constraints and limitations within the aeropendulum's mechanical system is essential. We can extract time domain

[1]https://github.com/dimitriarthur/aeropendulum_project

parameters and define the control specifications through the system's response information. These temporal aspects hold significant implications for control efficacy, providing insights into the mechanical system's inherent temporal capabilities.

Figure 6.1 depicts the system response. Through this initial test, we acknowledge reasonable specifications for the rise, settling, and peak times. We evaluate the time constants regarding the step response using a duty cycle of $\tau = 43.14\%$. The rise, peak, and settling time were $t_r = 0.43$ s, $t_p = 0.84$ s, $t_s = 3.53$ s respectively. One can see that the steady-state angular position is $\theta = 0.54$ rad. Hence, we can define the temporal requirements for the closed-loop response as in Table 6.1. These values were chosen so that the physical limits related to the step response data.

Figure 6.1: Step response of the system to a duty cycle of 43.14%.



Source: prepared by the author.

Table 6.1: Time domain requirements for the closed-loop response.

Description	**Value**
t_r - Rise Time	0.5 s
t_p - Peak Time	0.9 s
t_s - Settling Time	3.6 s
M_p - Overshoot	20 %

6.3 Gain adjustment

MATLAB's Control System Toolbox provides apps and methods to project and optimize a linearized version of control systems [40]. Here, we use the PID controller adjustment using two degrees of freedom to control the trade-off between robustness and time response. More specifically, we set the optimization to focus on the reference tracking. The gains are automatically computed as long as the PID controller blocks are used in the Simulink diagram.

6.4 PID controller implementation on the microcontroller

We use the PID Library, publicly available on the Arduino's library manager [41]. It consists of a solid PID implementation that tames most common problems, such as the derivative kick, windup-induced lags, and actuation limits. The discrete-time version of the controller is implemented in the library using the Forward Euler method.

6.5 Small-signal linearization

Since the aeropendulum is a nonlinear system, one can benefit from the linear control theory by linearizing the system around an operating point. Under the scope of small signal linearization, it is possible to treat the system as linear.

By linearizing the model for a specific angular position y_0, the linearized version of the system is described by

$$\frac{d\Delta y}{dt} = \Delta\Omega, \tag{6.1}$$

$$\frac{d\Delta\Omega}{dt} = -\frac{mgd}{J}\cos(y_0)\Delta y - \frac{C}{J}\Delta\Omega + \frac{L}{J}\Delta f. \tag{6.2}$$

Therefore, the transfer function that relates the angular position and the thrust force applied is given by

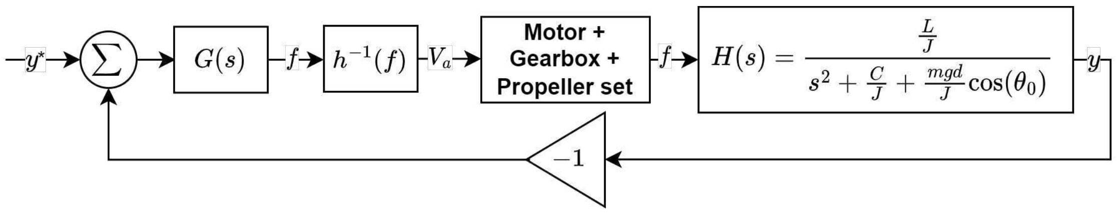
$$H(s) = \frac{\frac{L}{J}}{s^2 + \frac{C}{J}s + \frac{mgd}{J}\cos(y_0)}. \tag{6.3}$$

6.5.1 Block diagram representation

The linearization is only applied to the mechanical description of the problem. The design decision is based on the fact that we ideally consider that there is a source of mechanical conjugate.

Using the linearized version of the system expressed in Equation 6.3, we created a block diagram that represents the system as depicted in Figure 6.2. The control allocation considers the inverse relationship between the desired thrust force from the controller output and the armature voltage to excite the actuator. A thrust force is then produced by the motor and propeller set, leading to the desired angular displacement.

Figure 6.2: Small-signal linearization closed loop diagram.



y^*
$\sum$
$G(s)$
f
$h^{-1}(f)$
V_a
Motor + Gearbox + Propeller set
f
$H(s) = \frac{\frac{L}{J}}{s^2 + \frac{C}{J} + \frac{mgd}{J}\cos(\theta_0)}$
y
-1

Source: prepared by the author.

6.5.2 Type of controller

Since the transfer function has two poles, we infer that the system's dynamic has a potential for an oscillatory behavior. We can mitigate or control this behavior using a derivative term. Therefore, a PD or PID controller would be more suitable for this system than a simple PI controller. However, a PID controller would be more appropriate to address steady-state errors or improve steady-state accuracy.

6.5.3 Simulation results

Using the Control Toolbox, the controller gains were computed $k_p = 0.04$, $k_i = 0.993$, and $k_d = -0.0086$. The closed-loop response in simulation is shown in Figure 6.3. The settling time was $t_s = 1.16$ s, the rise-time $t_r = 0.26$ s, peak-time $0.58s$, and the overshoot $M_p = 9.98\ \%$. Comparing these values with the temporal requirements from Table 6.1, we can say that the simulation results satisfied the closed-loop specifications.

Figure 6.3: Closed-loop performance for the small signal linearized simulated model at $\theta = 0.4$ rad. θ represents the angular displacement, and τ the control action in terms of the duty cycle.

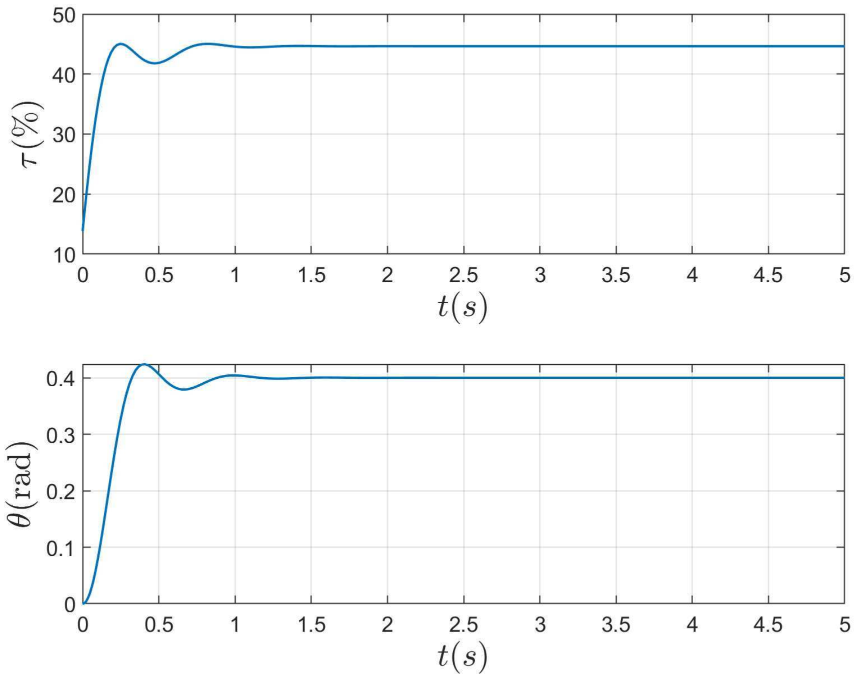


Figure 6.4: Source: prepared by the author.

6.5.4 Experimental platform results

Figure 6.5 depicts the closed-loop performance of the experimental platform. The temporal requirements were satisfied given that $t_s = 0.4\ s$, $t_r = 0.25\ s$. It is worth emphasizing that before reaching the steady-state value, there are oscillations that can be related to vibrations of the aeropendulum mast. Another source of oscillations might be that there is no back-EMF control but an inverse of the desired thrust to the necessary armature voltage. It does not guarantee that the speed of the propeller will be constant, nor the thrust force produced.

The model's small-signal linearized version should accurately approximate the nonlinear system in the equilibrium point $\theta = 0.4$ rad. It should present nonlinear effects until the system reaches the equilibrium point. Nevertheless, these nonlinearities do not affect the system's closed-loop performance until it reaches the desired position. The fact that the nonlinearity is not that severe might positively impact this since it is just comprised of a trigonometric term.

Figure 6.5: Closed loop performance for the system linearized using small signal linearization at $\theta = 0.4$ rad. θ represents the angular displacement, and τ the control action in terms of the duty cycle.

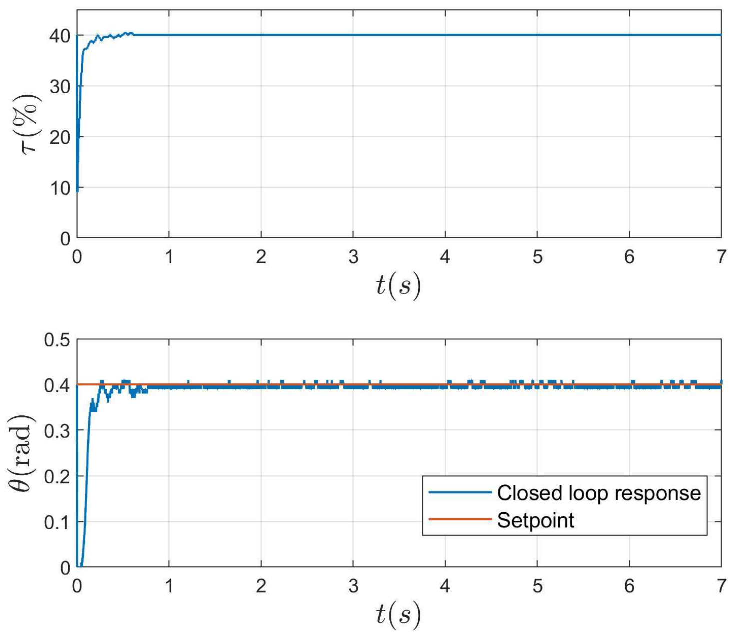


Figure 6.6: Source: prepared by the author.

6.6 Input-output feedback linearization

The feedback linearization strategy would be suitable to move the system through different operating points and cancel the nonlinearities. By recalling (6.2), we know that $\frac{d\Omega}{dt} = -\frac{mgd}{J}\sin(y) - \frac{C}{J}\omega + \frac{L}{J}f$. As detailed in [42], considering the system's input as the thrust f, it will be chosen as a function of v to cancel the nonlinearity of the plant, and it is given by

$$f = \frac{J}{L}\left[\frac{mgd}{J}\sin(y) + v\right]. \tag{6.4}$$

By substituting (6.4) in (6.2), the equivalent linear version of the system is described by:

$$\frac{d\Omega}{dt} = -\frac{mgd}{J}\sin(y) - \frac{C}{J}\omega + \frac{J}{L}\frac{L}{J}\left[\frac{mgd}{J}\sin(y) + v\right], \tag{6.5}$$

$$\frac{d\Omega}{dt} = -\frac{C}{J}\omega + v. \tag{6.6}$$

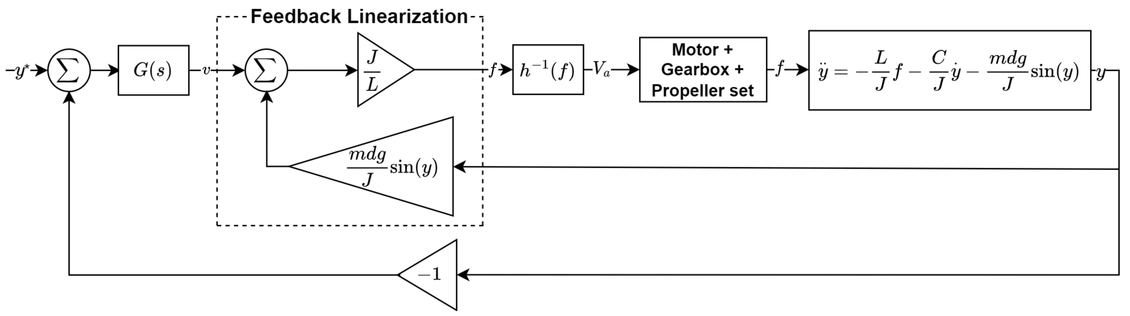
Therefore, the transfer function for the linearized version of the system is:

$$H(s) = \frac{Y(s)}{V(s)} = \frac{1}{s(s+\frac{C}{J})} \tag{6.7}$$

6.6.1 Block diagram representation

We treat the system's input as V_a, performing the inverse computation $h^{-1}(f)$ to perform the control allocation. Using this feedback linearization scheme, the plant becomes linear throughout all the angular displacements of interest.

Figure 6.7: Diagram for the feedback linearization applied to the aeropendulum platform. The position controller $G(s)$ outputs the control action, and the linearization block calculates the value to cancel the nonlinearity. The pseudo-control action v is converted to the required V_a, and the generated thrust is input to the aeropendulum.



Source: prepared by the author.

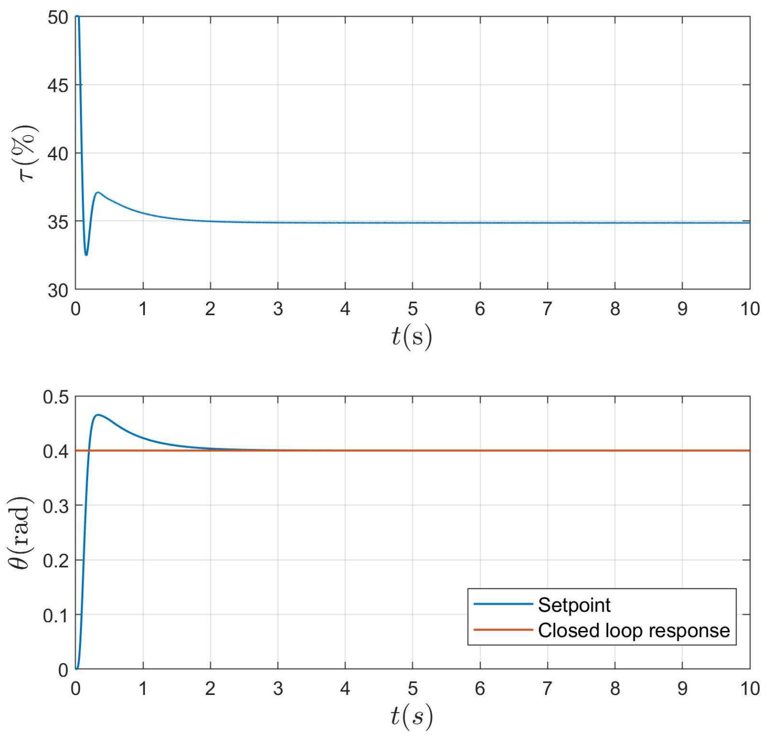
6.6.2 Choice of the type of the controller

Since the linearized version of the plant detailed in Equation 6.7 has a double integrator, we can specify a PD controller for the plant not to insert any additional pole. [43] analyzes the stabilization and command following performance of a double integrator plant, comparing different types of controllers, from classical PID to Continuous Sliding Mode controllers. In terms of stabilization, the PD controller performs as well as the other controllers for all prescribed criteria. Regarding the analysis of the command following, the PD controller presents a fair performance compared to the alternatives and improves performance when an integrator is added. Nonetheless, adding an integrator improves the command following performance [?]. Hence, to tackle the position control problem of the linearized plant, we chose a PID controller for the application.

6.6.3 Simulation results

We used the simulated model and a PID controller with gains $k_p = 291$, $k_d = 436$, and $k_i = 15$ to gain confidence and evaluate the approach's effectiveness. Figure 6.8 depicts the closed loop performance for the reference tracking of $\theta = 0.4$ rad, and the control action in terms of $\tau(\%)$. The temporal requirements specified in Table 6.1 were satisfied, with $t_s = 1.56$ s , $t_r = 0.11$ s, $t_p = 0.33$ s, $M_p(\%) = 16.34$.

Figure 6.8: Closed loop performance of the simulated model linearized using the feedback linearization technique.

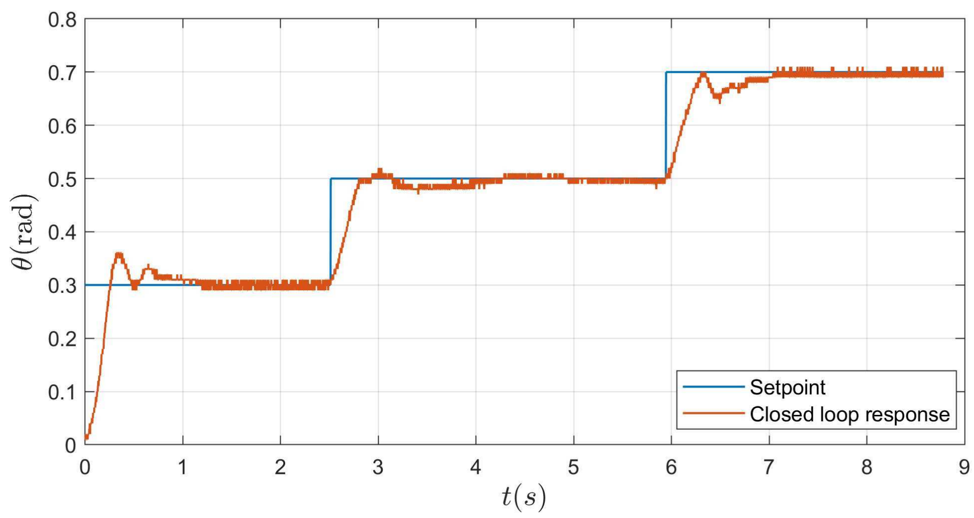


Source: prepared by the author.

6.6.4 Experimental results

Figure 6.9 depicts the closed-loop performance of the experimental platform. We applied different step reference values, and the result is satisfactory regarding reference tracking. Regarding temporal requirements, all specifications were satisfied for the different reference steps with zero steady-state error. Nonetheless, it is worth emphasizing that there is a more intense overshoot or unexpected oscillation in some of them, as in the first reference step with $M_p = 12\%$. These oscillations might come from two different sources. The first one is mechanical, where the vibrations are more intense than the weight of the mast can handle. The other is related to the fact that this strategy has no back-EMF control, which means that the actuator is prone to vary its speed, thus oscillating the thrust force produced.

Figure 6.9: Closed loop response of the feedback linearization version of the system .

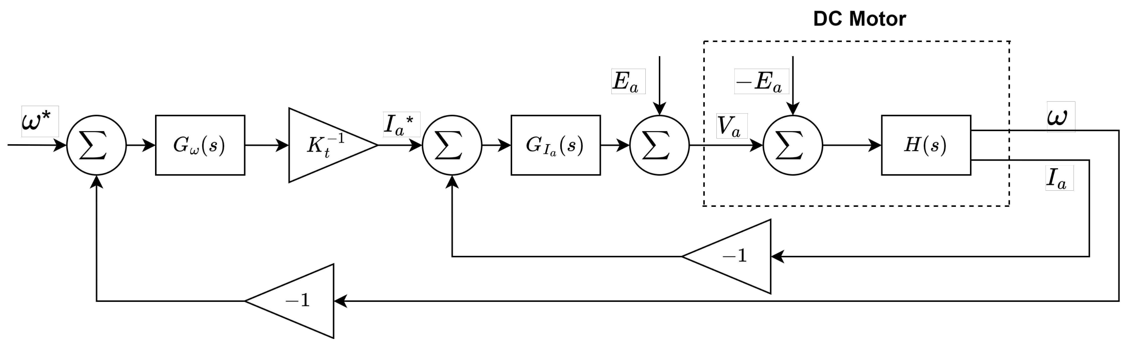


Source: prepared by the author.

6.7 Cascade controller

The velocity of a DC motor can be indirectly controlled by managing the armature current, and this strategy is named cascade control. As depicted in Figure 6.10, the inner loop determines the setpoint of the armature current based on the comparison between the current value and the reference one. The outer loop acts to match the reference speed and the current one. In both loops, a PI controller is employed within a closed-loop system, with $k_{p_\omega} = 0.1$, $k_{i_\omega} = 1.5$, $k_{p_I} = 1.2$, $k_{i_I} = 0.5$. The gains were computed using Simulink's Control Toolbox. There are saturation mechanisms to restrict the delivered current and the voltage supplied to the motor.

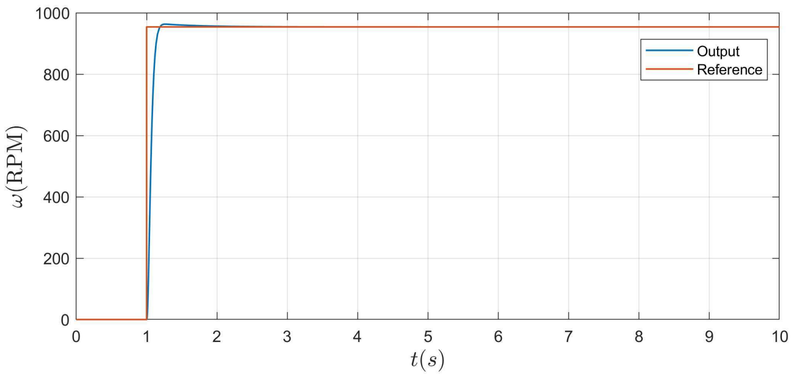
Figure 6.10: Cascade Control.



Source: prepared by the author.

To demonstrate the effectiveness of the control strategy using the simulated model, we set a step reference speed of 950 RPM and monitored the speed of the rotor. Figure 6.13 depicts that the rotor's speed followed the reference value, respecting the temporal requirements ($t_s \leq 0.8\ s$, $t_r \leq 0.2\ s$, $Mp(\%) \leq 10\%$).

Figure 6.11: Reference speed and the output of the cascade controller for the angular speed.

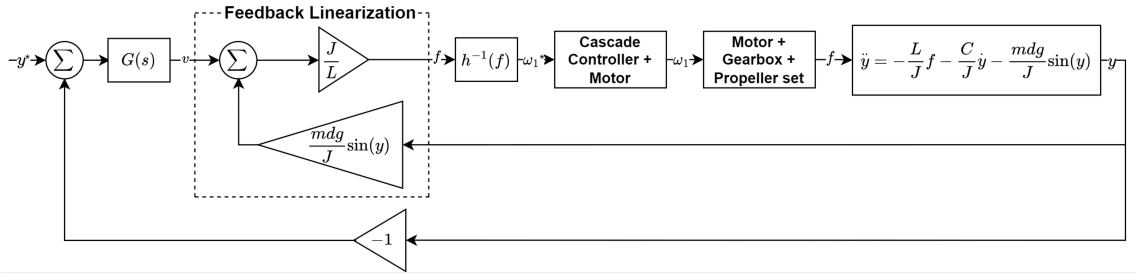


Source: prepared by the author.

6.8 Input-ouput feedback linearization and cascade control

As previously described, the feedback linearization technique is beneficial to transform the nonlinear system into a fully linear one. At the same time, the cascade control strategy makes it possible to control the rotor's speed. Combining these two strategies makes it possible to move the system throughout different operating regions and guarantee that there will be fewer oscillations in the thrust force produced by the propeller. Figure 6.12 depicts the block diagram of the proposed strategy. First, the control action for the position error is computed. The pseudo control action v is then converted to the reference speed ω_1, which serves as the reference speed for the cascade controller. The output of the inner loop is the desired rotor speed, which is converted to the desired thrust force.

Figure 6.12: The block diagram of the feedback linearization strategy combined with a cascade control strategy.

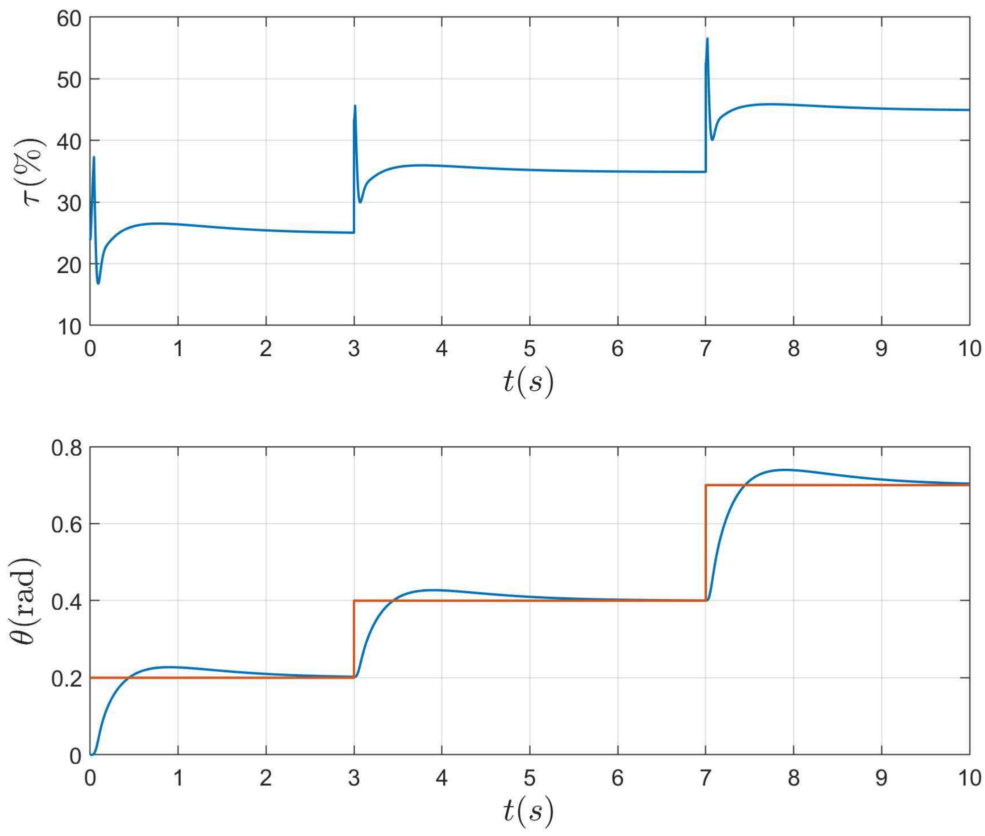

Feedback Linearization
y^*
$G(s)$
v
$\frac{J}{L}$
f
$h^{-1}(f)$
ω_1^*
Cascade Controller + Motor
ω_1
Motor + Gearbox + Propeller set
f
$\ddot{y} = -\frac{L}{J}f - \frac{C}{J}\dot{y} - \frac{mdg}{J}\sin(y)$
y
$\frac{mdg}{J}\sin(y)$
-1

Source: prepared by the author.

6.8.1 Simulation results

The diagram illustrated by Figure 6.12 was implemented in a simulated environment. Figure 6.13 depicts the closed-loop response using the position controller with the same gains used in the feedback linearization without the cascade controller. The system tracked the reference values for three different operating points, $\theta_1 = 0.2$ rad, $\theta_2 = 0.4$ rad, and $\theta_3 = 0.7$ rad, with zero steady-state error. The temporal requirements were satisfied, with $t_s \leq 3.6$ s for all step reference values, $t_r \leq 0.5$, and $M_p(\%) \leq 20\%$. Given that saturation limits were applied to the reference rotor's speed and the armature current values, we see that the duty cycle did not exceed the maximum actuation value.

Figure 6.13: Cascade Control.



Source: prepared by the author.

6.9 Final considerations

This chapter discussed different control approaches using the simulated model and the experimental platform. The simulation environment made it possible to gain trust and test different controllers, while the experimental platform assured the effectiveness of the control design procedures.

The position control using the small-signal linearization was successfully implemented using a PID controller, with the simulated and experimental platforms meeting the time-domain requirements specification. In the aeropendulum platform, some oscillations could be attributed to vibrations on the mast and variations in the propeller's speed. The limitation of this approach is that each desired angular position must be taken into the linearization procedure, altering the gains and requiring a gain scheduling strategy.

The input-output feedback linearization strategy was also successfully implemented. A PID controller was chosen as a design decision, and the closed-loop performance, both in the simulated model and the experimental platform met the desired time-domain specifications. Oscillations were already expected since there is no back-EMF control.

A cascade control was proposed to tame the oscillations due to the variation in the rotor's speed. Both PI controllers were tuned to meet the desired performance criteria. The time-domain specifications were satisfied in the simulation environment, with no exceeding the actuation limits. Future work would be valuable to deal with the practical aspects of this strategy regarding the experimental platform.

In the next chapter, we make the final considerations and recommend directions for future work.

Chapter 7

Final Considerations

In this work, we have addressed the aeropendulum problem, which earlier works overlooked by making assumptions and not modeling the actuator's electrical dynamics. We proposed and implemented a low-level data collection to acquire the electrical information. We improved the system's overall modeling by estimating the electrical parameters and the nonlinear relationship between the PWM input and the armature voltage. Concerning the simulation model, we have included electrical and aerodynamic effects.

Searching for a better way to describe the aerodynamic behavior of the propeller, we employed the SINDy-based algorithms to help in this process. We explored noise mitigation strategies and derivatives approximation methods to improve the identification results, and it was evident that further effort must be employed to filter and assure high data quality. The WSINDy algorithm was shown to be the best option due to the absence of derivatives approximation. In simulation, the thrust coefficient was correctly estimated, and the model correctly predicted the dynamics on the test set. A sparse model could not be retrieved using data from the experimental platform.

The development of an acquisition scheme that runs solely on the Arduino made it possible to assess different control approaches with gradual levels of improvement. All the proposed strategies satisfy the specified temporal requirements, and there was a notable improvement regarding the system's performance. The feedback linearization technique is the most suitable since it enables the system to move smoothly throughout different operating regions. Nonetheless, the experimental platform claims a back-EMF control strategy, such as the cascade control, given that undesired oscillations are mainly due to motor speed variations.

7.1 Future work

For future work, we suggest:

- Diagnose and fix the measurements of the armature voltage values. This improvement will help to describe the electrical behavior of the actuator better.
- Improve the speed measurement of the propeller using the IR sensor and deal with measurement spikes.
- Enhance the DC motor parameter estimation, especially the mechanical ones, and update the simulated model.
- Revise the data preprocessing steps of the sparse identification algorithms and propose model selection criteria; use the real propeller speed instead of the estimated one.
- Evaluate other polynomial candidate functions to describe the thrust force, including non-integer exponents.
- Apply the feedback linearization with the cascade control approach to the experimental platform and assess the closed-loop performance.

Bibliography

1 SILVA, Yago Luiz Monteiro. *Projeto, Construção e Controle de um Aeropêndulo*. 2018. Monography, UFCG.

2 JúNIOR, Alexsandro Ferreira de Barros. *Construção e Projeto de Controle de um Aeropêndulo utilizando Model-Based Design*. 2019. Monography, UFCG.

3 ENIKOV, E.T.; CAMPA, Giampiero. Mechatronic aeropendulum: Demonstration of linear and nonlinear feedback control principles with matlab/simulink real-time windows target. *Education, IEEE Transactions on*, v. 55, p. 538–545, 11 2012.

4 BARROS, Stayner; LIMA, Rafael. Controlador pid Ótimo baseado em pso aplicado a um aeropêndulo. In: . [S.l.]: SBA Sociedade Brasileira de Automática, 2020.

5 LUCENA, Ellen R.; LUIZ, Saulo O. D.; LIMA, Antonio M. N. Modeling, parameter estimation, and control of an aero-pendulum. *Procedings do XV Simpósio Brasileiro de Automação Inteligente*, SBA Sociedade Brasileira de Automática, 2021.

6 GRIESEBNER, Klaus. *Model-based Controller Development*. 2017. MSc Dissertation , Halmstad University.

7 LJUNG, L.; SÖDERSTRÖM, T. Theory and practice of recursive identification. MIT press Cambridge, MA, 1983.

8 OVERSCHEE, Peter Van; MOOR, Bart De. Subspace identification for linear systems. theory, implementation, applications. *Springer Science & Business Media*, xiv, 01 1996.

9 HOF, Paul M.J. Van den. Lecture notes in system identification: Data-driven modeling of dynamic systems. Department of Electrical Engineering Eindhoven University of Technology, February 2020.

10 CHEN, S.; BILLINGS, S. A. Representations of non-linear systems: the narmax model. *International Journal of Control*, Taylor & Francis, v. 49, n. 3, p. 1013–1032, 1989.

11 SCHOUKENS, Johan; LJUNG, Lennart. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, v. 39, n. 6, p. 28–99, 2019.

12 GUO, Yu; WANG, Fei; LO, James Ting-Ho. Nonlinear system identification based on recurrent neural networks with shared and specialized memories. p. 2054–2059, 2017.

13 GONZALEZ, Jesus; YU, Wen. Non-linear system modeling using lstm neural networks. *IFAC-PapersOnLine*, v. 51, p. 485–489, 06 2018.

14 YING, Xue. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, v. 1168, p. 022022, 02 2019.

15 LöHNING, Martin et al. Model predictive control using reduced order models: Guaranteed stability for constrained linear systems. *Journal of Process Control*, v. 24, n. 11, p. 1647–1659, 2014. ISSN 0959-1524.

16 BRUNTON, Steven; PROCTOR, Josh; KUTZ, J. Sparse identification of nonlinear dynamics (sindy). 04 2016.

17 BRUNTON, Steven L.; KUTZ, J. Nathan. 7 data-driven methods for reduced-order modeling. De Gruyter, p. 307–344, 2020.

18 STANKOVIć, Alex M.; SARIć, Aleksandar A.; SARIć, Andrija T.; TRANSTRUM, Mark K. Data-driven symbolic regression for identification of nonlinear dynamics in power systems. p. 1–5, 2020.

19 BRUNTON, Steven; PROCTOR, Josh; KUTZ, J. Sparse identification of nonlinear dynamics (sindy). 04 2016.

20 KAISER, Eurika; KUTZ, J.; BRUNTON, Steven. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, v. 474, 11 2017.

21 CORTIELLA, Alexandre; PARK, K C; DOOSTAN, Alireza. *Sparse Identification of Nonlinear Dynamical Systems via Reweighted l_1-regularized Least Squares.* 2020.

22 MESSENGER, Daniel A.; BORTZ, David M. Weak sindy for partial differential equations. *Journal of computational physics*, v. 443, 2021.

23 SUN, Chong; TIAN, Tian; ZHU, Xiao cheng; DU, Zhaohui. Sparse identification of nonlinear unsteady aerodynamics of the oscillating airfoil. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, v. 235, p. 095441002095987, 09 2020.

24 BASTOS, VINICIUS BENITES. *Virtual Environments Assisted by Machine Learning for Modeling and Test of Robotic Platforms.* 2021. PhD Thesis, UNICAMP.

25 CHARTRAND, Rick. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 5 2011.

26 KAHEMAN, Kadierdan et al. Learning discrepancy models from experimental data. *arXiv.org perpetual*, 2019.

27 CORKE, Peter. *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®.* [S.l.]: Springer, 2011. v. 73. 1-495 p. (Springer Tracts in Advanced Robotics, v. 73). ISBN 978-3-642-20143-1.

28 SILVA, Brian de et al. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, The Open Journal, v. 5, n. 49, p. 2104, 2020.

29 BREUGEL, Floris Van; KUTZ, J. Nathan; BRUNTON, Bingni W. Numerical differentiation of noisy data: A unifying multi-objective optimization framework. *IEEE Access*, v. 8, p. 196865–196877, 2020.

30 OLIVEIRA, Arthur D. B.; LIMA, Rafael B. C.; LIMA, Antonio M. N. Data-driven system identification of an aeropendulum. *Anais do Simpósio Brasileiro de Automação Inteligente (SBAI 2023)*, oct. 2023.

31 GROUP, OSRAM. *AS5040 Product Document*. 2019.

32 ARDUINO. *Arduino Rev3 Motorhsield*. 2015.

33 CORPORATION, Atmel. *ATmega 640 Microcontroller Product Document*. 2014.

34 MOTOROLA. *Motorola Analog IC Device Data*. 2022.

35 CZYŻ, Zbigniew; KARPIŃSKI, Paweł; SKIBA, Krzysztof; WENDEKER, Mirosław. Wind tunnel performance tests of the propellers with different pitch for the electric propulsion system. *Sensors*, MDPI AG, v. 22, n. 1, p. 2, dez. 2021.

36 PRASETIYO, Erwan Eko. A simple brushless motor and propeller test stand for experiment from home. *Journal of Physics: Conference Series*, IOP Publishing, nov. 2021.

37 KÓSA, Patrik et al. Experimental measurement of a UAV propeller's thrust. *Tehnicki vjesnik - Technical Gazette*, Mechanical Engineering Faculty in Slavonski Brod, v. 29, n. 1, fev. 2022.

38 DEFLORIAN, Michael; ZAGLAUER, Susanne. Design of experiments for nonlinear dynamic system identification. *IFAC Proceedings Volumes*, Elsevier BV, v. 44, n. 1, p. 13179–13184, jan. 2011.

39 FASEL, Urban et al. SINDy with control: A tutorial. *60th IEEE Conference on Decision and Control (CDC)*, IEEE, dez. 2021.

40 MATHWORKS. *Tune 2-DOF PID Controller (PID Tuner)*. 2015. Available at: <https://la.mathworks.com/help/control/ug/tune-2-dof-pid-controller-pid-tuner.html>.

41 BEAUREGARD, Brett. *Arduino PID Library*. 2017. Available at: <https://playground.arduino.cc/Code/PIDLibrary/>.

42 LUCENA, Ellen Ribeiro. *Modelagem e Projeto de Controladores para Aeropêndulo com Desenvolvimento de Interface Gráfica Didática*. 2021. Monography, UFCG.

43 RAO, Venkatesh G.; BERNSTEIN; S., Dennis. Naive control of the double integrator. *IEEE Control Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 21, n. 5, p. 86–97, oct. 2001.