TÉCNICO LISBOA

COSMIC: fast closed-form identification from large-scale data for LTV systems leading to optimal spacecraft attitude control

Maria Cruz de Carvalho

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Professor Cláudia Alexandra Magalhães Soares
Doctor Pedro António Duarte Marques Lourenço

Examination Committee

Chairperson: Professor José Fernando Alves da Silva
Supervisor: Doctor Pedro António Duarte Marques Lourenço
Member of the Committee: Professor António Pedro Rodrigues Aguiar

December 2021

À avó Olinda, de onde estiver a ver

Agradecimentos

Antes de mais, quero agradecer aos meus supervisores por todo o apoio que me deram ao longo do desenvolvimento deste projeto. Professora Cláudia, muito obrigada pelas ideias e pela paciência que teve. Pedro, muito obrigada por toda a paciência também e pelo esforço e tempo dedicado a este trabalho, aprendi muito. Sem o vosso apoio, conhecimento e dedicação esta tese não teria sido possível.

Tenho também de agradecer à GMV pelo apoio à realização deste trabalho. Em especial, um obrigada ao João Branco pela oportunidade e ao João Franco pelo apoio prestado ao longo dos meses de trabalho e pelas contruibuições para a sua realização.

Quero agradecer ao Técnico por todas as oportunidades que me proporcionou, onde eu cresci e aprendi mais do que nunca. Em especial, agradeço ao AeroTéc, onde tive oportunidade de tornar muitas ideias realidade, conheci pesssoas que mudaram o meu percurso e continuarão a mudar a minha vida e onde pude recarregar (e descarregar) energias.

À reunião das quartas que tanto ajudou, com muita Yakisoba e Lobby da Agenda pelo meio, Sei Lá como isto teria sido sem vocês e a SA'19. Obrigada aos companheiros de zoom, Hugo e Teresa, pela partilha de glórias e tristezas ao longo destes meses. Nada disto seria igual se não fosse também pela Catarina e pela Rita, na nossa casinha e a crescer juntas nestes cinco anos.

À minha família, irmão, avós e tias, obrigada por tudo. Em especial, o meu maior obrigada vai para os meus pais, por acreditarem sempre em mim, serem o meu maior pilar e pelo apoio e amor sem condição. Quando for grande, quero ser como vocês.

Aos leitores desta tese, muito obrigada pelo interesse!

Resumo

O presente trabalho apresenta um método em forma fechada para identificação de sistemas lineares variantes no tempo a partir de dados, em tempo discreto. O problema de aprendizagem é formulado como um problema de mínimos quadrados regularizado, onde o regularizador promove transições suaves entre instantes de uma trajetória. Além disso, desenvolvemos um algoritmo em forma fechada com garantias de que a solução encontrada é ótima e uma complexidade que varia linearmente com o número de instantes considerados por trajetória. O algoritmo COSMIC atinge o resultado desejado mesmo na presença de grandes volumes de dados, o que não é verdade para *solvers* de uso generalizado e até para um método de *coordinate descent* especialmente projetado para este problema. Para provar a viabilidade de aplicação a sistemas reais, começamos por realizar a validação num sistema sintético massa-mola-amortecedor e verificamos que o modelo de sistema estimado pode ser usado para encontrar o controlo ótimo ao longo da trajetória. O nosso algoritmo é ainda implementado num simulador de baixa fidelidade da missão Comet Interceptor da ESA, cujo objetivo exige que as câmaras que leva a bordo estejam apontadas numa direção extremamente precisa. Assim, concluímos que esta tese fornece uma nova e melhor abordagem a técnicas clássicas de identificação de sistemas para sistemas lineares variantes no tempo, provando ainda que o algoritmo desenvolvido é uma base sólida para aplicações na indústria espacial e um passo em frente para a incorporação de algoritmos que beneficiam dos dados num ambiente científico onde as garantias de segurança são essenciais.

Palavras-chave: forma fechada, identificação de sistemas, linear variante no tempo, Espaço

Abstract

The present work introduces a closed-form method for identification of discrete-time linear time-variant systems from data. We formulate the learning problem as a regularized least squares problem where the regularizer favors smooth solutions within a trajectory. Further, we develop a closed-form algorithm with guarantees of optimality and a complexity that varies linearly with the number of instants considered per trajectory. The COSMIC algorithm achieves the desired result even in the presence of large volumes of data, too large for general purpose solvers and for a specially designed coordinate descent method to reach a valid solution. To prove its applicability to real world systems, we start by performing the validation in synthetic spring-mass-damper systems and guarantee that the estimated system model can be used to find the optimal control path for such systems. Our algorithm was implemented in a Low Fidelity Simulator for a simplified version of the Comet Interceptor mission from European Space Agency (ESA), that requires precise pointing of the on-board cameras in a fast dynamics environment. Thus, we conclude that this thesis provides a new and better approach to classical system identification techniques for linear time-variant systems, while proving to be a solid base for applications in the Space industry and a step forward to the incorporation of algorithms that leverage data in such a safety-critical scientific environment.

Keywords: closed-form, system identification, linear time-variant, Space

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

$\delta\iota$	Integral of attitude error.
$\delta\theta$	Angular displacement.
λ_k	Regularization parameter for the k-th instant.
ω	Angular velocity.

Roman symbols

$\mathbf{A}$	Dynamics matrix.
$\mathbf{B}$	Control matrix.
$\mathbf{C}$	Optimization variable.
$\mathbf{D}$	Data matrix.
$\mathbf{Q}_k, \mathbf{R}_k$	Control weights matrices.
$\mathbf{u}$	Input vector.
$\mathbf{x}$	State vector.
f	Cost function.
L	Total number of simulations for the k-th instant.
N	Total number of instants.

Subscripts

ℓ	Trajectory considered.
k	Instant considered.

List of Acronyms

AI Artificial Intelligence

AOCS Attitude and Orbital Control System

CA Closest Approach

DKE Dynamics and Kinematics Environment

ESA European Space Agency

FoV Field of View

GNC Guidance, Navigation and Control

LQR Linear Quadratic Regulator

LTI Linear Time-Invariant

LTV Linear Time-Variant

ML Machine Learning

SBCD Stochastic Block Coordinate Descent

Chapter 1

Introduction

1.1 Motivation

Aerospace industry is at the forefront of innovation, being one of the precursors of multiple technologies that have made life on Earth better. However, this does not come without a cost, be it human, environmental or monetary. Therefore, the Space industry tends to demand strong guarantees and proof of work from new technologies to be used in Space missions.

Guidance, Navigation and Control (GNC) and Attitude and Orbital Control Systems (AOCSs) are crucial to any Space mission, thus being subject to intensive research efforts [1]. Nevertheless, several new tools used daily in other industries have yet to come through as standard in Space missions.

Artificial Intelligence (AI) and Machine Learning (ML) techniques are now commonplace in our day to day lives but have yet to make a lasting impression in the Space industry, considerably due to the inherent difficulty to validate in the context of safety-critical activities and the lack of explainable and verifiable methods to be presented to a more cautious audience.

In order to keep up with new advances in the industry and with the disruption caused by the democratization of Space, new applications and the emergence of new private entities challenging governmental players, it is urgent to bridge the differences between control theory and ML, to exploit the potential of faster and better control algorithms, which make use of data to enhance their performance and adapt to the environment. The integration of ML techniques and GNC can be helpful in a broad set of problems, from building robust control frameworks that address parameter varying systems to applying verification and validation techniques to a system in a more efficient way.

We focus on identifying a system's behavior, which constitutes a problem in many areas of engineering, from biological relations to physical systems' dynamics, and is the first step to any control design. It is imperative to have a system model that correctly represents its interaction with the environment and allows for more accurate predictions of its response to external stimuli.

Typically, this issue is addressed by considering previous knowledge about the system, applying first principles of physics and measuring whenever possible. However, for more intricate systems, this approach fails because it may become impossible to measure all the parameters needed to characterize

a behavior completely or the system is too complex to be modeled by simple equations. In order to address these limitations, a data-driven approach to the system identification problem, where input-output data is used to find a model that describes a particular system, has become more common [2]. Finding more efficient and comprehensive universal algorithms to deal with a large amount of data is essential to derive reasonable data-driven solutions and constitutes a pressing issue.

Moreover, system modeling represents a significant cost in complex engineering projects, sometimes up to 50% of the total cost, whether it be because of the obvious cost of the hours dedicated to this problem by specialized professionals or the length and expenses of the experiments needed to obtain a good model. Thus, it is essential to create practical system identification tools that adapt to a wide range of problems and achieve a solution in a time-constrained setting. Such tool can be especially useful in a Space mission project that represents huge cost efforts for entire countries and agencies [3].

1.2 Mission overview

The Comet Interceptor mission aims to explore a comet entering the Solar System for the first time by obtaining precise images of the celestial body that will allow its chemical characterization, the assessment of its structure and the interaction of its components [4]. The Closest Approach (CA) between the spacecraft and the comet will happen at 1000 km and the relative velocity of the bodies will be between 10 km/s and 70 km/s, which evidences the need for a control system that can maintain the pointing of the comet cameras within a reasonable range for image collection and is robust to unknown disturbances.

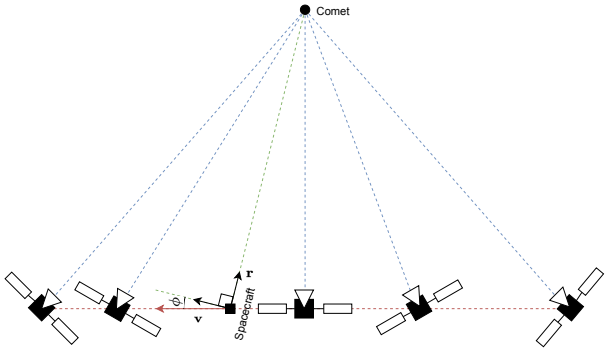


Figure 1.1: Comet Interceptor Scenario. The spacecraft is represented in different moments of the trajectory, with the scientific instruments pointed at the comet.

Although the solution proposed in the Concurrent Design Facility Study Report for the Comet Interceptor is different, we are going to analyze a mission profile that regards the pointing as a function of the attitude of the spacecraft, without the aid of external pointing correction structures, as these might get blocked by the spacecraft body. Thus, the pointing requirement becomes an attitude control problem, where we wish to achieve a spacecraft orientation in relation to the comet that allows it to be in the main camera line of sight constantly. The expected trajectory of the spacecraft in relation to the target is shown in Figure 1.1, where the red body axis must be aligned with the direction of the comet. This

requirement gets harder to meet the closer the spacecraft gets to the target, as the dynamics become faster and the actuation needed to maintain the attitude increases.

A couple of problems could arise, from the lack of actuator power to deficiencies in the control to accompany the movement, or even external perturbations that impact the spacecraft and have such an impact that render the mission unfeasible. In order to prevent the scenarios referred, a robust GNC framework must be built and deployed, which can be done by resorting to optimal control schemes and new methodologies for the estimation of the disturbances. The previously described system can be modeled, for controller design purposes, as a linear time-variant.

1.3 Problem statement

The present work aims to address the shortcomings of classical system identification techniques for Linear Time-Variant (LTV) systems in order to create models that are simultaneously accurate enough to represent a wide range of phenomena while being simple enough to use with well-known and widely used control techniques, leading to a more robust control algorithm. The main challenges of this proposal are the amount of data needed to characterize one trajectory due to its changing dynamics and guaranteeing a solution for the problem in a finite-time setting.

1.4 Related work

A large body of work has been developed in order to incorporate information provided by measurements in system identification and controller synthesis, whether these come from more traditional sensors, such as inertial measurement units, or image and video [5], providing guarantees and robustness.

The results described in [6] establish the grounds for system identification from observed data. Earlier, the work of Kumpati et al. [7] already proposed a system identification strategy based on Neural Networks. Nevertheless, data availability and computational and algorithmic tools were limited, so performance and speed were constrained.

Recently, there has been more focus on providing guarantees that a problem can be solved to a given precision in finite time, which is crucial to bridge the gap to optimal control. Hardt et al. [8] provides a solution to a linear dynamical system identification learning problem through gradient descent techniques, for a Linear Time-Invariant (LTI) system, with a polynomial number of samples. On Dean et al. [9], a three-step controller design method is proposed. The first step corresponds to the identification of an LTI system through ordinary least squares, allowing the estimation error characterization and its employment in a robust controller design, through System Level Synthesis [10]. Oymak et al [11] take it a step further by proposing to learn a realization of an LTI system from a single input/output trajectory. Sarkar et al. [12] detail a new statistical analysis of the ordinary least squares estimator for LTI systems while achieving near-optimal finite time solutions, providing finite time guarantees for LTI systems. Although these works present significant achievements, not many real world systems can be represented

as linear time-invariant systems and generalizing these findings to LTV systems is essential for a broader application of the results.

The literature is not as extensive regarding the direct application of data-driven system identification to linear time-variant systems. However, this problem has been a concern for many years. Dudul et al [13] apply Feedforward Neural Networks to identify an LTV system, assuming a transfer function characterization. Lin et al. [14] proposes an episodic block model for an LTV system, where parameters within a block are kept constant, followed by the exploration of a meta-learning approach for system identification divided into two steps: an offline initialization process and online adaptation. The solution establishes an error bound for both initialization and online learning, while the simulations outperform a least squares estimator without regularization for small samples.

Formetin et al [15] have recently proposed a system identification procedure that takes control specifications into account in the form of regularization. Although following a different representation of the system that the one we adopt to follow, the solution presented in their work derives a model for the system and a controller by the minimization of a cost function that accounts for the controller error and the authors study a Bayesian control design to upgrade from the nominal version achieved by using the convex optimization for the best estimation of the model. Validation of the controller is then performed from a probabilistic perspective. The model estimated is intended to be used solely for the model-based controller, in a setting very similar to the one we are proposing.

Additionally, when linear systems no longer satisfy the users' and system's needs, nonlinear system identification comes into play. This class of problems presents many new challenges that may not be common in linear system identification, for example, the need for thorough experimental processes for data collection to guarantee that all behaviors are covered or even the difficulty of proposing a model structure to represent the system [16].

Mania et al [17] states the difficulties of learning or estimating nonlinear systems with continuous-time states and inputs due to insufficient data. They present an active learning methodology for data collection and results production that guarantees a solution in finite time. Given past observations, a system is estimated, then the system is run and more data is collected. In the end, using the data collected, the system is re-estimated. This result is particularly interesting due to its online application and incorporation of new data in the learning process.

However, when presented with a nonlinear model, one can decide to go forward to identify the best linear approximation if it is taken into account the error associated with this assumption. We can choose to implement a linear system identification for nonlinear systems due to its simplicity and reasonably good approximation for many applications or even as just an initial estimation [16] [18]. When choosing how to proceed with the system identification, we must recognize the trade-off between the versatility of nonlinear models and the simplicity of linear ones. It is important to refer that LTV systems, in particular discrete-time, are a powerful class of models and perform well when used to approximate nonlinear dynamics and can be very useful for controller design and analysis, as we propose to do in this thesis [19].

Going more profound in the domain literature, we can find work from the Advanced Concepts Team at ESA that regards a direct application of Machine Learning algorithms to classical Space problems, such

as autonomous landing [20] or interplanetary transfers [21]. Both of these works build on deep learning algorithms, such as deep neural networks, and can be acknowledged as a step forward in integrating these technologies in the Space industry. However, we can also recognize that simpler models, which allow for a better common knowledge understanding, would be beneficial for more acceptance in the community.

Izzo et al. [21] introduces a novel method for the generation of optimal trajectories, which allows for the training of optimal control trajectories and presents a possible solution to one of the biggest challenges of direct application of ML to optimal control: the need to solve the optimal control problem to generate an optimal trajectory before training a new model. Additionally, this work once again uses G&CNETS, previously proposed by the same team and with already proven results and stability [22]. We opt out of this approach and design a generic algorithm for system identification that does not propose to learn the optimal control path directly, but that can be applied to optimal controller design and is not domain specific, offering a generic approach to LTV system identification.

1.5 Contributions

We present a closed-form solution for the LTV system identification problem, with a complexity that varies linearly with the number of instants considered for the modeling. We formulate the problem as a regularized least squares optimization problem and build the solution and its demonstration on the grounds of the convex optimization framework. The constrained variation of the system's dynamics constitutes the perfect setting for smoothness imposing regularization, which to the best of our knowledge has never been done, and we can then tune the learning model to adapt to different characteristics throughout the trajectory.

This solution represents the main contribution of this work and we wish to emphasize its benefits, when compared to generic tools, such as `cvxpy`, and iterative minimization algorithms, based on coordinate descent, especially for large-scale problems, and its integration with well studied controller design structures, such asLinear Quadratic Regulator (LQR). To support our work, we use simulation environments and apply our findings to simple systems, such as a spring-mass-damper LTV, and to real world case studies from the Space industry, such as the Comet Interceptor mission that we presented before.

1.6 Thesis outline

To support the work we intend to accomplish throughout this thesis, we present the theoretical basis needed to thoroughly comprehend the results in **Chapter 2** as well as the validation system we intend to use in the following chapters.

With this background set, **Chapter 3** presents the problem we wish to address and the possible solutions, from a problem agnostic application to more specific ones that leverage on the problem representation. Here, we derive a necessary and sufficient condition to guide data collection to guarantee a solution and we prove to have a closed-form solution for the LTV system identification problem and

validate it. Additionally, we prove that it is a better option than other well-known methods to solve convex optimization problems, mainly when applied to large scale cases.

Chapter 4 states the solution the optimal control problem for the discrete LTV and we implement it using the model estimated in the previous Chapter to prove that it can be used to feed classical controller design structures, such as the LQR.

In **Chapter 5**, to complete the analysis of the closed-form solution and its applicability to real world problems, we employ it in a Low Fidelity Simulator of the Comet Interceptor mission, after discussing the functionalities of the simulator.

Finally, **Chapter 6** provides a summary of the main achievements of this work and suggests possible future applications.

Chapter 2

Background

This Chapter sets the basis for the work we perform in the following chapters regarding the notation that we intend to use and the theoretical background needed to fully understand the developments. Also, we present the validation model that is used throughout this thesis to complete multiple performance assessments.

2.1 Notation

For the fulfillment of this work, vectors are going to be found in bold non-italic lowercase $\mathbf{a} \in \mathbb{R}^n$, while matrices are to be found in bold non-italic uppercase $\mathbf{A} \in \mathbb{R}^{n \times m}$. Scalar constants, $A \in \mathbb{R}$, and scalar variables, $a \in \mathbb{R}$, are represented in italic uppercase and italic lowercase, respectively.

The identity matrix is represented by $\mathbf{I}$ and a matrix or vector of zeros is $\mathbf{0}$. We assume that their dimensions are clear from the context.

Throughout this work, whenever we use the operator $\boldsymbol{\nabla}$, we refer to it as an expanded gradient with respect to matrix quantities, such that for matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, we have

$$\boldsymbol{\nabla}_{\mathbf{X}} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{X}_{11}} & \cdots & \frac{\partial}{\partial \mathbf{X}_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{X}_{m1}} & \cdots & \frac{\partial}{\partial \mathbf{X}_{mn}} \end{bmatrix}. \tag{2.1}$$

The Frobenius norm of matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is

$$\|\mathbf{X}\|_F = \left(\mathrm{Tr}(\mathbf{X}^T\mathbf{X})\right)^{1/2} = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{x}_{ij}^2\right)^{1/2} \tag{2.2}$$

with $\mathbf{x}_{ij}$ being the element of line i and column j. From [23], its derivative is

$$\frac{\mathrm{d}\|\mathbf{X}\|_F}{\mathrm{d}\mathbf{X}} = 2\mathbf{A}^T(\mathbf{A}\mathbf{X}). \tag{2.3}$$

2.2 Theoretical background

2.2.1 Convex optimization

A function is defined as a convex function if its domain is a convex set and, for all x and y, with $0 \leq \theta \leq 1$, we have

$$f(\theta \mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}). \tag{2.4}$$

Generically, we can derive first and second order conditions to evaluate the convexity of a function. If f is differentiable, f is convex if and only if, with a convex domain and for all $\mathbf{x}$ and $\mathbf{y}$ in this domain, we have the first order condition as

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (x - y). \tag{2.5}$$

If we get $\nabla f(\mathbf{x}) = 0$, then $f(\mathbf{y}) \geq f(\mathbf{x})$, which proves that $\mathbf{x}$ is a global minimizer of f. This results proves to be one of the strongest properties of convex functions, as we can get global information from local information [24].

An optimization problem has the form

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0 \\ & h_i(\mathbf{x}) = 0 \end{aligned} \tag{2.6}$$

with f_0 as the objective function, for what we wish to find a minimum, while subject to strict rules we call constrains, addressed as f_i. Essentially, an optimization problem consists on making the best choice out of a set of possible choices. If no constraints are present, the problem is said to be unconstrained.

If the problem is a minimization and the objective function and constraints are convex, we have a convex optimization problem. When the convex optimization problem is unconstrained, for $\mathbf{x}$ to be optimal, the first-order condition reduces to the necessary and sufficient condition [24]

$$\nabla f_0(\mathbf{x}) = 0. \tag{2.7}$$

The second order condition can be stated as

$$\nabla^2 f_0(\mathbf{x}) \succeq 0. \tag{2.8}$$

2.2.2 Least squares estimation

The least squares problem is a particular case of unconstrained convex optimization and it can be used in multiple settings and variations. For the purpose of this work, we will detail the regularized least squares. The ordinary least squares is formulated as

$$\underset{\mathbf{x}}{\text{minimize}} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2, \tag{2.9}$$

and it can be reduced to solving a set of linear equations $\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{Ab}$ [24].

The regularized least squares is defined by adding extra terms to the cost function that penalize or emphasize special characteristics of the problem. Let $g(\mathbf{x})$ be the regularization function and λ the coefficient that regulates the influence g has in the estimation. We can rewrite (2.9) as

$$\underset{\mathbf{x}}{\text{minimize}} \quad \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \lambda g(\mathbf{x}). \tag{2.10}$$

2.2.3 Linear systems

When presented with the same input, a linear time-invariant system will always output the same response, independently of the time. In practice, both dynamics and control matrices are constant. In contrast, a linear time-variant system response will depend on time, as the dynamics and control matrix will vary with time. Taking a time-bounded trajectory as an example, we can consider that we only need to find the system characterization once for the LTI but as many times as the time steps encapsulated by the trajectory for the LTV. In essence, a system identification problem for a linear time-variant setting is significantly computationally more demanding.

2.2.4 Linear Quadratic Regulator

A feedback control system can use state feedback to regulate the system's output and track a reference input in the presence of disruptions and uncertainty. The design of the controller can be addressed from different viewpoints and one of the most common and beneficial in the Space industry is an optimal control setup. This approach tries to find the control inputs that best fit a given criteria. To accomplish the objectives of the present work, we address the controller design from an optimization perspective, by attempting to minimize a cost function and focus our analysis on the LQR for a discrete LTV system [25].

Considering a discrete-time LTV system

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k), \tag{2.11}$$

the LQR problem resides in finding the control signal $\mathbf{u}^*(k) = -\mathbf{K}_k\mathbf{x}(k)$ that minimizes the LQR cost, which is defined, for LTV discrete systems, as

$$J = \frac{1}{2}\mathbf{x}^T(N-1)\mathbf{H}\mathbf{x}(N-1) + \frac{1}{2}\sum_{k=0}^{N-2}\mathbf{x}^T(k)\mathbf{Q}_k\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}_k\mathbf{u}(k), \tag{2.12}$$

with $\mathbf{H} \geq 0$, $\mathbf{Q}_k \geq 0$ and $\mathbf{R}_k > 0$.

Additionally, let us represent $c_k(\mathbf{x}(k), \mathbf{u}(k))$ as

$$c_k(\mathbf{x}(k), \mathbf{u}(k)) = \frac{1}{2}\left(\mathbf{x}^T(k)\mathbf{Q}_k\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}_k\mathbf{u}(k)\right). \tag{2.13}$$

Matrices $\mathbf{Q_k}$ and $\mathbf{R_k}$ allow for the adjustment of the solution to better adapt to the objectives of the system. The terms regulated by these matrices represent the output energy and the control signal

energy, respectively. Although the LQR intends to find a controller that minimizes both energies, they vary inversely, which means that decreasing one energy will increase the other. The function of $\mathbf{Q_k}$ and $\mathbf{R_k}$ is to allow for a tuning of the trade-off [26]. One of the key challenges of the LQR is the choice of these matrices.

It is important to understand the implications of varying $\mathbf{Q_k}$ and $\mathbf{R_k}$ to ease the tuning phase of the controller design. On the one hand, when we define $\mathbf{R_k}$ much larger than $\mathbf{Q_k}$, the optimal value of the cost can be achieved by employing smaller control inputs, while the state response will be more significant. On the other hand, when we define $\mathbf{R_k}$ much smaller than $\mathbf{Q_k}$, the optimal solution is found for smaller outputs, even at the control input expense. Moreover, we can individually define the entries of the matrices to manipulate how much a state variable is regarded or not and the influence they have on the control input.

2.2.5 Metrics for closed-loop performance evaluation

Closed-loop performance can be evaluated through multiple perspectives. We introduce five metrics used to address the controller performance for the various design parameters so that the best ones can be chosen [27]. The error is the difference between the state at a given time instant and the desired value.

The metrics used in this work are:

- **Rise time, t_r:** Rise time is defined as the time it takes for the output to reach an error that is less than 10% and it is usually required to be small;
- **Sum of the steady state error:** Stands for the error accumulated along the trajectory, since the point where rise time is reached until the end and it must be minimized;
- **Maximum control input:** The maximum value demanded from the actuators, it is crucial for physical systems with actuator constraints. We want to reduce it whenever possible;
- **Sum of the control input energy:** The quantity that stems directly from the cost function of the LQR and it must be as small as possible;
- **Percentage of overshoot** Defined as the peak value divided by the final value and should be kept below 20%.

2.3 Models for validation: spring, mass and damper systems

To validate the methods that are proposed, a spring-mass-damper system is reproduced, first as LTI and then as LTV. Let us consider a classical spring-mass-damper system as

$$\ddot{z} = -\frac{C_s}{m}\dot{z} - \frac{C_d}{m}z + \frac{1}{m}f \tag{2.14}$$

with z as the mass position in relation to the resting point at $z = 0$, C_s is the spring constant, C_d is the damping coefficient and f represents the effects of external inputs on the system.

If we denote $\mathbf{x} = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}$ as the state, the continuous time state space system can be addressed as

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{C_s}{m} & -\frac{C_d}{m} \end{bmatrix}}_{\mathbf{A}_c} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{\mathbf{B}_c} u(t), \tag{2.15}$$

To represent this scenario as a discrete-time linear system, a discretization of the problem was performed, assuming piecewise-constant inputs $u(k)$, with m and Δt constants, resulting in

$$\mathbf{x}(k+1) = \underbrace{e^{\mathbf{A}_c \Delta t}}_{\mathbf{A}(k)} \mathbf{x}(k) + \underbrace{\mathbf{A}_c^{-1}(e^{\mathbf{A}_c \Delta t} - \mathbf{I})\mathbf{B}_c}_{\mathbf{B}(k)} u(k), \tag{2.16}$$

which is comparable to (2.11). We can approach this system as a LTI, by maintaining parameters C_s and C_d constants, and as LTV, by varying these same parameters with time. For this particular case, we defined

$$\begin{cases} C_{s_k} = \cos\left(1.5\omega_0 k + \frac{\pi}{4}\right)^2 C_s \\ C_{d_k} = \left[1.5 + \cos\left(\omega_0 k\right)\right] C_d \end{cases} . \tag{2.17}$$

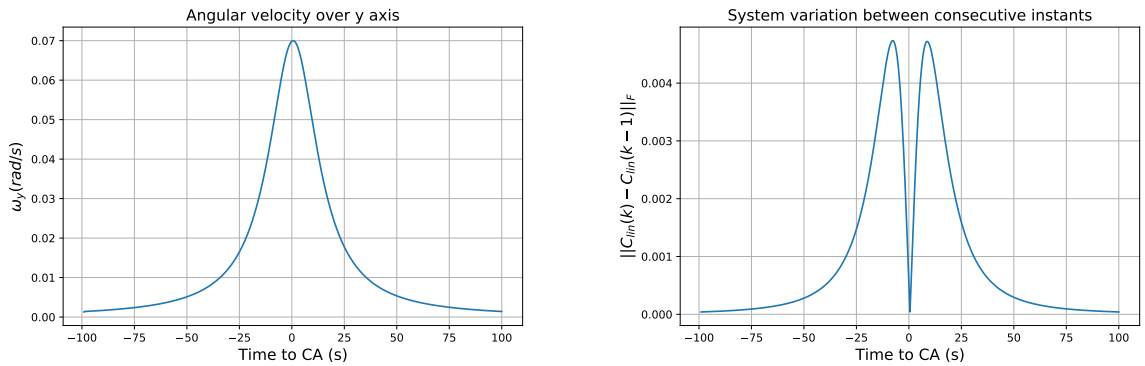
Chapter 3

System identification from data

The present Chapter addresses the main contributions of this work. We start by formulating the problem as a regularized least squares, supporting the formulation with the mission characteristics, deriving a necessary and sufficient condition on data to achieve a solution. We then proceed to prove that we can solve the problem in a closed-form, achieving a linear complexity of the number of data points per trajectory. To support the claims made, we provide a validation study and a benchmarking analysis.

3.1 Model for realistic LTV systems

As a first approach to the attitude modeling problem for the Comet Interceptor mission, we analyzed the angular velocity profile that the spacecraft must meet to perform its mission correctly, Figure 3.1(a). This reference angular velocity is the nominal trajectory as explained later, on Section 5.2.3. This can be used to derive a time-variant linearization of the attitude dynamics, thus allowing the analysis of the variation between instants, as in Figure 3.1(b).



(a) Angular velocity evolution.

(b) Variation between instants.

Figure 3.1: System variation throughout the trajectory.

It is clear that the closer the body is to the point of CA, the faster it needs to change its attitude. Moreover, it can be observed that the variation of the system characterization, $\mathbf{C}$, between consecutive

instants, in a discrete-time setting, is constrained and does not vary indefinitely. This key observation induces a requirement on our model such that, in consecutive time instants, the model does not vary too much.

Let us address a discrete linear time-variant system defined as

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k), \tag{3.1}$$

such that $k \in [0, ..., N-2]$, with N being the total number of instants considered to be part of one trajectory. In a given instant k, the state is $\mathbf{x}(k) \in \mathbb{R}^p$ and the control input is $\mathbf{u}(k) \in \mathbb{R}^q$. System parameters $\mathbf{A}(k) \in \mathbb{R}^{p\times p}$ and $\mathbf{B}(k) \in \mathbb{R}^{p\times q}$ are, respectively, the dynamics and control matrices that define the system's response and the unknown variables we aim to derive from the data.

Taking into account all the data available, we need to additionally define the matrices that contain the state information, $\mathbf{X}(k) \in \mathbb{R}^{p\times L}$ as

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}_1(k) & \mathbf{x}_2(k) & ... & \mathbf{x}_L(k) \end{bmatrix}$$

and the control information, $\mathbf{U}(k) \in \mathbb{R}^{q\times L}$ as

$$\mathbf{U}(k) = \begin{bmatrix} \mathbf{u}_1(k) & \mathbf{u}_2(k) & ... & \mathbf{u}_L(k) \end{bmatrix},$$

of all the L different simulations for the k-th instant of the trajectory. Moreover, we define $\mathbf{X'}(k)$ as $\mathbf{X'}(k) = \mathbf{X}(k+1)$.

To find the proper solution for system (3.1), we start by defining the optimization variable $\mathbf{C}(k) \in \mathbb{R}^{(p+q)\times p}$ as

$$\mathbf{C}(k) = \begin{bmatrix} \mathbf{A}^T(k) \\ \mathbf{B}^T(k) \end{bmatrix},$$

with $\mathbf{C} = [\mathbf{C}(k)]_{k=0}^{k=N-2}$, such that $\mathbf{C} \in \mathbb{R}^{(N-1)(p+q)\times p}$.

Additionally, to allow for a least squares representation of the identification problem, we also need to define $\mathbf{D}(k) \in \mathbb{R}^{L\times(p+q)}$ as

$$\mathbf{D}(k) = \begin{bmatrix} \mathbf{X}^T(k) & \mathbf{U}^T(k) \end{bmatrix},$$

with $\mathbf{D} = [\mathbf{D}(k)]_{k=0}^{k=N-2}$, allowing to have

$$\mathbf{V} = \begin{bmatrix} \mathbf{D}(0) & \mathbf{0} & ... & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}(1) & ... & \mathbf{0} & \mathbf{0} \\ ... & ... & ... & ... & ... \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{D}(N-3) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{0} & \mathbf{D}(N-2) \end{bmatrix},$$

which results in $\mathbf{V} \in \mathbb{R}^{(N-1)L \times (N-1)(p+q)}$ and ultimately the problem can be formulated as

$$\underset{\mathbf{C}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{VC} - \mathbf{X'}\|_F^2. \tag{3.2}$$

According to the key observation in the beginning of the Chapter, to limit the variation of the system between instants, thus encoding the domain knowledge that a system will not change drastically from one time step to the next, we add a term to the cost function, with regularization parameters $\lambda_k > 0$, turning problem (3.2) into the regularized problem. By allowing for the λ_k to vary throughout the trajectory, we impose more flexibility to the problem formulation, covering a wider range of problems.

As such, we are solving

Problem 1.

$$\underset{\mathbf{C}}{\textit{minimize}} \quad f(\mathbf{C}) := \frac{1}{2}\|\mathbf{VC} - \mathbf{X'}\|_F^2 + \frac{1}{2}\sum_{k=1}^{N-2} \|\lambda_k^{1/2}(\mathbf{C}(k) - \mathbf{C}(k-1))\|_F^2. \tag{3.3}$$

Remark 1. *The problem we are solving can be seen as a trade off between how close the optimization variable is to the data and how much we allow for it to change between instants. Thus, we can address the tuning of parameter λ_k as the tuning of the relative importance of each objective. Higher values of λ_k are congruent with little variation of the system behavior between instants and will emphasize the weight of the second term has in the cost function and lower values will allow for more drastic changes between k and $k+1$.*

Moreover, to aid the analysis, we can define the second term of f writing all the weighted difference equations $\lambda_k^{1/2}\left(\mathbf{C}(k) - \mathbf{C}(k-1)\right)$ as a matrix product

$$\underbrace{\begin{bmatrix} \lambda_1^{1/2} & 0 & ... & 0 & 0 \\ 0 & \lambda_2^{1/2} & ... & 0 & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & \lambda_{N-3}^{1/2} & 0 \\ 0 & 0 & ... & 0 & \lambda_{N-2}^{1/2} \end{bmatrix}}_{\boldsymbol{\Upsilon}^{1/2}} \underbrace{\begin{bmatrix} -\mathbf{I} & \mathbf{I} & \mathbf{0} & ... & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} & ... & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ ... & ... & ... & ... & ... & ... & ... \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} & -\mathbf{I} & \mathbf{I} \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} \mathbf{C}(0) \\ \mathbf{C}(1) \\ ... \\ \mathbf{C}(N-2) \end{bmatrix}, \tag{3.4}$$

allowing Problem 1 to be written as

Problem 2.

$$\underset{C}{\textit{minimize}} \quad f(\mathbf{C}) := \frac{1}{2}\|\mathbf{VC} - \mathbf{X'}\|_F^2 + \frac{1}{2}\|\boldsymbol{\Upsilon}^{1/2}\mathbf{FC}\|_F^2. \tag{3.5}$$

This statement makes explicit that we are solving an unconstrained convex optimization problem. As Problem 2 is an unconstrained minimization of a quadratic function, a solution is proven to exist and is well-defined [24]. In order to lighten the notation, we use $\boldsymbol{\nabla}$ to refer to $\boldsymbol{\nabla}_{\mathbf{C}}$ and $\boldsymbol{\nabla}_{\mathbf{k}}$ to refer to $\boldsymbol{\nabla}_{\mathbf{C}(k)}$. Solving problem (3.5) amounts to solving the equation

$$\boldsymbol{\nabla} f(\mathbf{C}) = \mathbf{0}. \tag{3.6}$$

As for 2.3, the following linear equation solves the problem,

$$\begin{aligned}\nabla f(\mathbf{C}) &= \mathbf{V}^T(\mathbf{V}\mathbf{C}^* - \mathbf{X'}) + \mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F}\mathbf{C}^* = \mathbf{0} \\ &\Leftrightarrow \left(\mathbf{V}^T\mathbf{V} + \mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F}\right)\mathbf{C}^* = \mathbf{V}^T\mathbf{X'} \quad . \\ &\Leftrightarrow \mathbf{C}^* = \left(\mathbf{V}^T\mathbf{V} + \mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F}\right)^{-1}\mathbf{V}^T\mathbf{X'}\end{aligned} \tag{3.7}$$

We now derive a condition on data that guarantees that the information collected is enough to reach a solution and correctly identify the system.

Informal statement of Theorem 1 *[When is the dataset large enough?] When the collected data is sufficiently varied, there is a unique solution to Problem 1. Further, after each collected trajectory, it is possible to identify if there is enough information for attaining a unique solution by computing the sum of the required trajectories covariances and testing it for positive definiteness.*

To prove Theorem 1, we will require two lemmas.

Lemma 1. *Let* $\mathbf{B} \succeq 0$, $\mathbf{B} \in \mathbb{R}^{m\times m}$ *and* $\mathbf{x} \neq 0$. *If* $\mathbf{x} \in Im(\mathbf{B})$ *then* $\mathbf{x}^T\mathbf{B}\mathbf{x} > 0$.

Proof. If the result of the Lemma does not hold, we would have that $\mathbf{x} \in$ Im$(\mathbf{B})$ and $\mathbf{x} \in$ Ker$(\mathbf{B})$. However, since Im$(\mathbf{B})\cap$Ker$(\mathbf{B}) = \emptyset$, the only option would be to have $\mathbf{x} = \mathbf{0}$, which violates the conditions of the Lemma, thus proving that the inference holds. □

Lemma 2. *The following two expressions are equivalent:*

(i) Both (a) and (b) hold:

(a) $\mathbf{A} \succeq \mathbf{0}$, $\mathbf{B} \succeq \mathbf{0}$;

(b) $\forall \mathbf{v} \in$ *Ker*$(\mathbf{B}) \setminus \{\mathbf{0}\} \quad \mathbf{v}^T\mathbf{A}\mathbf{v} > 0$;

(ii) $\mathbf{A} + \mathbf{B} \succ \mathbf{0}$.

Proof. The proof that (ii) implies (i) follows trivially from the fact that both $\mathbf{A}$ and $\mathbf{B}$ are positive semidefinite.

To prove the other way around, we first remember that if $\mathbf{B} \succeq 0$, then Im$(\mathbf{B}) \perp$ Ker$(\mathbf{B})$.

Let $\mathbf{v} = \mathbf{v_I} + \mathbf{v}_K$, $\mathbf{x} \neq \mathbf{0}$, with $\mathbf{v_I} \in$ Im$(\mathbf{B})$ and $\mathbf{v_K} \in$ Ker$(\mathbf{B})$.

If $\mathbf{v_I} \neq \mathbf{0}$, then

$$(\mathbf{v_I} + \mathbf{v}_K)^T(\mathbf{A} + \mathbf{B})(\mathbf{v_I} + \mathbf{v}_K) = (\mathbf{v_I} + \mathbf{v}_K)^T\mathbf{A}(\mathbf{v_I} + \mathbf{v}_K) + (\mathbf{v_I} + \mathbf{v}_K)^T\mathbf{B}(\mathbf{v_I} + \mathbf{v}_K). \tag{3.8}$$

From the conditions of the Lemma, we know that the first term of the sum holds, as $\mathbf{v}^T\mathbf{A}\mathbf{v} > 0$. For the second term, we recall Lemma 1, yielding that $\mathbf{v}^T\mathbf{B}\mathbf{v} > 0$, thus we infer that the LHS product is greater than zero. From this, we get

$$\mathbf{A} + \mathbf{B} \succ 0.$$

If $\mathbf{v_I} = \mathbf{0}$, (3.8) is turned into

$$\mathbf{v}_K(\mathbf{A} + \mathbf{B})\mathbf{v}_K = \mathbf{v}_K\mathbf{A}\mathbf{v}_K > \mathbf{0}. \tag{3.9}$$

Thus, we prove that $\mathbf{A}+\mathbf{B} \succ \mathbf{0}$. □

Theorem 1. *[Existence and uniqueness of the solution of Problem 2] Let the collected dataset* $\mathbf{D}$ *be comprised of* L *trajectories drawn independently*

$$\mathbf{D} = \{(\mathbf{x}_\ell(k), \mathbf{u}_l(k)) : k \in \{0, ..., N-2\}, \ell \in \{1, ..., L\}\}$$

and let $\mathbf{\Sigma}_\ell$ *be the empirical covariance of trajectory* ℓ*, i.e.,*

$$\mathbf{\Sigma}_\ell = \frac{1}{N}\sum_{k=0}^{N-1} \begin{bmatrix} \mathbf{x}_\ell(k) \\ \mathbf{u}_\ell(k) \end{bmatrix} \begin{bmatrix} \mathbf{x}_\ell(k) \\ \mathbf{u}_\ell(k) \end{bmatrix}^T, \tag{3.10}$$

with $\ell \in \{1, ..., L\}$*.*

Then, the following statements are equivalent:

(i) the solution of Problem 1 exists and is unique;

(ii) the empirical covariance of the data, $\mathbf{\Sigma} = \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 + ... + \mathbf{\Sigma}_L$*, as defined in* (3.10)*, is positive definite.*

Proof. Initially, let us define $\bar{\mathbf{x}}(k) = vec(\mathbf{X}(k))$ and $\bar{\mathbf{c}}(k) = vec(\mathbf{C}(k))$ and rewrite Problem 1

$$\begin{aligned} \underset{\bar{\mathbf{c}}}{\text{minimize}} \quad f(\bar{\mathbf{c}}) :=& \frac{1}{2}\sum_{k=0}^{N-2} \|\bar{\mathbf{x}}(k+1) - (\mathbf{D}(k) \otimes \mathbf{I})\bar{\mathbf{c}}(k)\|^2 + \\ & \frac{1}{2}\sum_{k=1}^{N-2} \|\lambda_k^{1/2}(\bar{\mathbf{c}}(k) - \bar{\mathbf{c}}(k-1))\|^2. \end{aligned} \tag{3.11}$$

Thus, applying the second-order condition (2.8), which states that a function f is convex if and only if its domain is convex and its Hessian is positive semidefinite. The Hessian of the cost function of our problem as stated in (3.11) is

$$\boldsymbol{\nabla^2} f(\bar{\mathbf{c}}) = \underbrace{\begin{bmatrix} \ddots & & \\ & \mathbf{D}^T(k)\mathbf{D}(k) \otimes \mathbf{I} & \\ & & \ddots \end{bmatrix}}_{\mathbf{\Pi}} + \underbrace{\begin{bmatrix} \lambda_1\mathbf{I} & -\lambda_1\mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ -\lambda_1\mathbf{I} & (\lambda_1+\lambda_2)\mathbf{I} & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & (\lambda_{N-3}+\lambda_{N-2})\mathbf{I} & -\lambda_{N-2}\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \cdots & -\lambda_{N-2}\mathbf{I} & \lambda_{N-2}\mathbf{I} \end{bmatrix}}_{\mathbf{\Gamma}} \tag{3.12}$$

and we must guarantee that $\boldsymbol{\nabla^2} f(\bar{\mathbf{c}}) \succ 0$, which is equivalent to say that the solution exists and is unique. From here on, we will use this knowledge to infer about the conditions on data.

Let us define the kernel of $\mathbf{\Gamma}$ as

$$\text{Ker}(\mathbf{\Gamma}) = \left\{ \begin{bmatrix} \bar{\mathbf{c}} \\ \vdots \\ \bar{\mathbf{c}} \end{bmatrix} : \bar{\mathbf{c}} \right\}. \tag{3.13}$$

From Lemma 2, if we have $\mathbf{\Pi} + \mathbf{\Gamma} \succeq 0$, then we know that

$$\begin{bmatrix} \bar{\mathbf{c}}^T & \cdots & \bar{\mathbf{c}}^T \end{bmatrix} \mathbf{\Pi} \begin{bmatrix} \bar{\mathbf{c}} \\ \vdots \\ \bar{\mathbf{c}} \end{bmatrix} > \mathbf{0}. \tag{3.14}$$

Therefore,

$$\begin{aligned} \bar{\mathbf{c}}^T \left(\sum_{k=1}^{N-2} \mathbf{D}^T(k)\mathbf{D}(k) \otimes \mathbf{I} \right) \bar{\mathbf{c}} > \mathbf{0} \Leftrightarrow & \sum_{k=1}^{N-2} \mathbf{D}^T(k)\mathbf{D}(k) \succ 0 \\ \Leftrightarrow & \sum_{k=1}^{N-2} \begin{bmatrix} \mathbf{X}^T(k) & \mathbf{U}^T(k) \end{bmatrix}^T \begin{bmatrix} \mathbf{X}^T(k) & \mathbf{U}^T(k) \end{bmatrix} \succ 0 \\ \Leftrightarrow & \sum_{k=1}^{N-2} \Big(\begin{bmatrix} \mathbf{x}_1^T(k) & \mathbf{u}_1^T(k) \end{bmatrix}^T \begin{bmatrix} \mathbf{x}_1^T(k) & \mathbf{u}_1^T(k) \end{bmatrix} + \\ & \cdots + \begin{bmatrix} \mathbf{x}_\ell^T(k) & \mathbf{u}_\ell^T(k) \end{bmatrix}^T \begin{bmatrix} \mathbf{x}_\ell^T(k) & \mathbf{u}_\ell^T(k) \end{bmatrix} \Big) \succ 0 \\ \Leftrightarrow & \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 + \cdots + \mathbf{\Sigma}_L \succ 0 \end{aligned} \tag{3.15}$$

Thus, we found the necessary and sufficient conditions on the data and proved that statements *(i)* and *(ii)* are equivalent. □

Theorem 1 allows to efficiently collect data, as we can address the covariance of the data as we collect it and verify its positive definiteness. When this condition is verified, we can stop the data collection and advance for the training.

Equation (3.7) makes evident that in order to find the optimal value for $\mathbf{C}$, it is required to invert a matrix

$$\boldsymbol{\mathcal{A}} := \mathbf{V}^T\mathbf{V} + \mathbf{F}^T\mathbf{\Upsilon}\mathbf{F},$$

$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{(N-1)(p+q)\times(N-1)(p+q)}$. This result shows that the approach requires the computation of a matrix heavily dependent on the number of instants considered in the trajectory being studied and the dimension of the system, meaning the result will become harder to obtain with an increased sampling rate and system complexity. When presented with a high sampling rate system, the size of matrices involved in the calculations can be very high, so solving directly the regularized least squares problem is not feasible. Regardless of this shortcoming, it is still important to understand the conditions in which the matrix $\boldsymbol{\mathcal{A}}$ is invertible, since it will impact the numerical solution of the optimization problem. Theorem 2 addresses this point.

Theorem 2. *Let $\mathbf{\Upsilon} \neq \mathbf{0}$. Then, matrix $\boldsymbol{\mathcal{A}}$ is invertible if there exists a set of $p+q$ pairs $(\ell, k) \in \{1, ..., L\} \times \{0, ..., N-2\}$ such that the corresponding $\begin{bmatrix} \mathbf{x}_\ell^T(k) & \mathbf{u}_\ell^T(k) \end{bmatrix}$ are all linearly independent.*

Proof. The proof is made by contraposition, by proving that if the problem does not have a unique solution, then conditions of the theorem do not hold.

Let us consider then that $\boldsymbol{\mathcal{A}}$ is not invertible. Then

$$\begin{aligned} \underset{\|\boldsymbol{\eta}\|=1}{\exists \boldsymbol{\eta}:} \quad & \boldsymbol{\eta}^T \boldsymbol{\mathcal{A}} \boldsymbol{\eta} = 0 \Leftrightarrow \boldsymbol{\eta}^T \mathbf{V}^T \mathbf{V} \boldsymbol{\eta} + \boldsymbol{\eta}^T \mathbf{F}^T \boldsymbol{\Upsilon} \mathbf{F} \boldsymbol{\eta} = 0 \\ & \Leftrightarrow \|\mathbf{V}\boldsymbol{\eta}\|^2 + \|\boldsymbol{\Upsilon}^{1/2}\mathbf{F}\boldsymbol{\eta}\|^2 = 0 \\ & \Leftrightarrow \mathbf{V}\boldsymbol{\eta} = \mathbf{0} \wedge \boldsymbol{\Upsilon}^{1/2}\mathbf{F}\boldsymbol{\eta} = \mathbf{0} \end{aligned} \quad . \tag{3.16}$$

Defining $\boldsymbol{\eta}(k) = \begin{bmatrix} \boldsymbol{\eta}_A^T(k) & \boldsymbol{\eta}_B^T(k) \end{bmatrix}^T$ and $\boldsymbol{\eta} = [\boldsymbol{\eta}(k)]_{k=0}^{k=N-2}$, it is possible to write, for all k,

$$\boldsymbol{\Upsilon}^{1/2}\mathbf{F}\boldsymbol{\eta} = \mathbf{0} \Leftrightarrow \begin{cases} \boldsymbol{\eta}_A(k) - \boldsymbol{\eta}_A(k-1) = \mathbf{0} \Leftrightarrow \boldsymbol{\eta}_A(k) = \boldsymbol{\eta}_A(k-1) := \bar{\boldsymbol{\eta}}_A \\ \boldsymbol{\eta}_B(k) - \boldsymbol{\eta}_B(k-1) = \mathbf{0} \Leftrightarrow \boldsymbol{\eta}_B(k) = \boldsymbol{\eta}_B(k-1) := \bar{\boldsymbol{\eta}}_B \end{cases} \tag{3.17}$$

where the fact that $\boldsymbol{\Upsilon}^{1/2}$ is invertible was used. This yields

$$\boldsymbol{\eta}^* := \underbrace{\begin{bmatrix} \mathbf{I} \\ \dots \\ \mathbf{I} \end{bmatrix}}_{\boldsymbol{\mathcal{N}}} \underbrace{\begin{bmatrix} \bar{\boldsymbol{\eta}}_A \\ \bar{\boldsymbol{\eta}}_B \end{bmatrix}}_{\bar{\boldsymbol{\eta}}} \tag{3.18}$$

denoting the null space of $\mathbf{F}$. In order for (3.16) to hold, $\boldsymbol{\eta}^*$ has to be contained by the null space of $\mathbf{V}$ whenever $\lambda \neq 0$, i.e.

$$\mathbf{V}\boldsymbol{\eta}^* = \mathbf{0} \Leftrightarrow \mathbf{V}\boldsymbol{\mathcal{N}}\bar{\boldsymbol{\eta}} = \mathbf{0} \tag{3.19}$$

that can be reduced to

$$\forall_k \forall_\ell \quad \mathbf{x}_\ell^T(k)\bar{\boldsymbol{\eta}}_A + \mathbf{u}_\ell^T(k)\bar{\boldsymbol{\eta}}_B = 0. \tag{3.20}$$

The solution of (3.20) implies that either $\bar{\boldsymbol{\eta}} = \mathbf{0}$, contradicting the hypothesis of the proof ($\|\boldsymbol{\eta}\| = 1$), or that the vectors $\begin{bmatrix} \mathbf{x}_\ell^T(k) & \mathbf{u}_\ell^T(k) \end{bmatrix}$ in the dataset are linearly dependent, i.e., it is not possible to find $p+q$ linearly independent vectors for all $\ell = 1, ..., L$ and $k = 0, ..., N-2$. Thus, if $\lambda_k \neq 0$ for all k and the matrix $\boldsymbol{\mathcal{A}}$ is singular, the condition of the Theorem cannot hold and, by contraposition, if it holds, then $\boldsymbol{\mathcal{A}}$ is invertible and there exists a unique solution to Problem 2.

This concludes the proof of sufficiency of the conditions to the existence of solution. □

Remark 2. *The conditions for the existence and uniqueness of solution set forth by Theorem 1 can be clearly seen as equivalent to this new result, as both require the complete set of* $\begin{bmatrix} \mathbf{x}_\ell^T(k) & \mathbf{u}_\ell^T(k) \end{bmatrix}$ *vectors within the dataset to span* $\mathbb{R}^{p+q}$*.*

Solving (3.7) is the path adopted by classic solvers to perform its operations, which becomes a disadvantage of this technology, as it is not capable of keeping up with the complexity of the estimation. Hence, addressing this shortcoming is essential to configure the system identification problem of linear time-variant systems as a regularized least squares problem and is the main objective of this Chapter. To this end, as made explicit by the diagram in 3.2, we present two distinct methods that exploit the specific

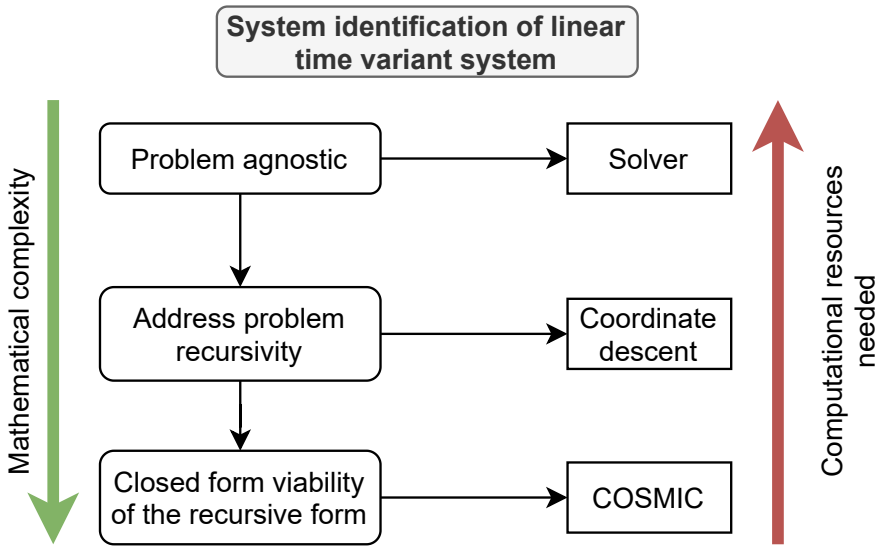


Figure 3.2: Possible approaches for the system identification problem in a convex optimization setting.

characteristics of this problem, representing a step above in terms of formal complexity, as they will require more mathematical manipulation, but will also decrease the computational resources required to find the solution.

3.2 Further analysis

We can regard (3.5) for each k instant independently, such that it can be formulated as

Problem 3.

$$\underset{\mathbf{C}}{\textit{minimize}} \quad f(\mathbf{C}) := \underbrace{\frac{1}{2}\sum_{k=0}^{N-2}\|\mathbf{D}(k)\mathbf{C}(k) - \mathbf{X'}^{T}(k)\|_F^2}_{h(\mathbf{C})} + \underbrace{\frac{1}{2}\sum_{k=1}^{N-2}\|\lambda_k^{1/2}(\mathbf{C}(k) - \mathbf{C}(k-1))\|_F^2}_{g(\mathbf{C})}. \tag{3.21}$$

Accordingly, let us once again think of the derivative of the cost function as a sum of the derivatives of its parcels and have for the k-th instant

$$\boldsymbol{\nabla}_{\mathbf{k}} f(\mathbf{C}) = \boldsymbol{\nabla}_{\mathbf{k}} h(\mathbf{C}) + \boldsymbol{\nabla}_{\mathbf{k}} g(\mathbf{C}), \tag{3.22}$$

knowing that the gradient at each instant k is $\boldsymbol{\nabla}_{\mathbf{k}} f(\mathbf{C})$. The gradient of $h(\mathbf{C})$ at k is

$$\boldsymbol{\nabla}_{\mathbf{k}} h(\mathbf{C}) = \mathbf{D}^T(k)(\mathbf{D}(k)\mathbf{C}(k) - \mathbf{X'}^T(k)). \tag{3.23}$$

The gradient of $g(\mathbf{C})$ is

$$\boldsymbol{\nabla} g(\mathbf{C}) = \mathbf{F}^T \boldsymbol{\Upsilon} \mathbf{F} \mathbf{C}. \tag{3.24}$$

With $\mathbf{F}$ as in (3.4), we can compute $\mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F}$

$$\mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F} = \begin{bmatrix} \lambda_1\mathbf{I} & -\lambda_1\mathbf{I} & \mathbf{0} & ... & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\lambda_1\mathbf{I} & (\lambda_1+\lambda_2)\mathbf{I} & -\lambda_2\mathbf{I} & ... & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ ... & ... & ... & ... & ... & ... & ... \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & -\lambda_{N-3}\mathbf{I} & (\lambda_{N-3}+\lambda_{N-2})\mathbf{I} & -\lambda_{N-2}\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} & -\lambda_{N-2}\mathbf{I} & \lambda_{N-2}\mathbf{I} \end{bmatrix}. \tag{3.25}$$

Replacing (3.23) in (3.22) and solving (3.24), we get $\boldsymbol{\nabla}_{\mathbf{k}} f(\mathbf{C}) =$

$$\begin{cases} \mathbf{D}^T(k)(\mathbf{D}(k)\mathbf{C}(k) - \mathbf{X'}^T(k)) + \lambda_{k+1}\mathbf{C}(k) - \lambda_{k+1}\mathbf{C}(k+1) = \mathbf{0}, & \text{for } k = 0 \\ \mathbf{D}^T(k)(\mathbf{D}(k)\mathbf{C}(k) - \mathbf{X'}^T(k)) - \lambda_k\mathbf{C}(k-1) + (\lambda_k+\lambda_{k+1})\mathbf{C}(k) - \lambda_{k+1}\mathbf{C}(k+1) = \mathbf{0}, & \text{for } k \in [1, ..., N-3] \\ \mathbf{D}^T(k)(\mathbf{D}(k)\mathbf{C}(k) - \mathbf{X'}^T(k)) - \lambda_k\mathbf{C}(k-1) + \lambda_k\mathbf{C}(k) = \mathbf{0}, & \text{for } k = N-2 \end{cases} \cdot \tag{3.26}$$

We have, thus, to solve a linear system of equations.

3.2.1 Stochastic Block Coordinate Descent

The first approach to solve (3.26) is a coordinate descent procedure. Given an optimization variable $\boldsymbol{\epsilon} \in \mathbb{R}^{m\times n}$, the block coordinate descent proposes that, instead of optimizing the cost function for the whole variable, the variable is divided into blocks, $\boldsymbol{\epsilon}_i$, and the problem is solved for that block, maintaining the other blocks constant [28]. This process should be repeated until a solution is reached, selecting different blocks for each update of the optimization variable. The stopping criteria requires for the derivative of the cost function to be below a predefined threshold.

The block selection can be performed sequentially or randomly. For the purpose of this work, we opt to randomize the instants of a trajectory and select one from such random order, up until all instances are updated. From then on, we restart the update step with a new random order. This approach guarantees that all instances are updated before repeating the process for an instant without updating the instances immediately before or after, reducing the potential excessive use of computational resources.

The Stochastic Block Coordinate Descent (SBCD) algorithm shortcomings are linked to the difficulty to update the cost function derivative efficiently: after a sharp decrease in the first iterations, the derivative update tends to reach a plateau, while the cost function is still decreasing, but at an unsuitable rate to achieve a solution in due course. Moreover, the update of the derivative is heavily dependent on the regularization parameter.

This method can be adapted to solve (3.26), by solving the equations to find the optimal value of a

particular block in those specific conditions and getting

$$\mathbf{C}^{+}(k) =$$

$$\begin{cases} \left(\mathbf{D}^T(k)\mathbf{D}(k) + \lambda_{k+1}I\right)^{-1}\left(\mathbf{D}^T(k)\mathbf{X'}^T(k) + \lambda_{k+1}\mathbf{C}(k+1)\right) & \text{for } k = 0 \\ \left(\mathbf{D}^T(k)\mathbf{D}(k) + (\lambda_k + \lambda_{k+1})I\right)^{-1}\left(\mathbf{D}^T(k)\mathbf{X'}^T(k) + \lambda_k\mathbf{C}(k-1) + \lambda_{k+1}\mathbf{C}(k+1)\right) & \text{for } k \in [1, ..., N-3] \\ \left(\mathbf{D}^T(k)\mathbf{D}(k) + \lambda_k I\right)^{-1}\left(\mathbf{D}^T(k)\mathbf{X'}^T(k) + \lambda_k\mathbf{C}(k-1)\right) & \text{for } k = N-2 \end{cases} . \tag{3.27}$$

In each iteration of the SBCD, $\mathbf{C}(k)$ is updated by a small amount $\boldsymbol{\delta} \in \mathbb{R}^{(p+q)\times p}$ and its new value can be written as

$$\mathbf{C}^{+}(k) = \mathbf{C}(k) + \boldsymbol{\delta}_i. \tag{3.28}$$

By replacing the previous value of $\mathbf{C}(k)$ with the new one in (3.26), we can get a new value of $\boldsymbol{\nabla}_{\mathbf{k}} f(\mathbf{C})$ in terms of a much simpler quantity δ, by updating the terms $\boldsymbol{\nabla}_{\mathbf{k}} h(\mathbf{C})$ and $\boldsymbol{\nabla}_{\mathbf{k}} g(\mathbf{C})$.

Algorithm 1 describes in detail the implementation of this approach. By evaluating the value of the derivative at the new $\mathbf{C}$, we can infer if the optimality condition was reached or not. However, this approach is iterative, which means that it may not be able to reach a solution if the proper limits are not found. Nevertheless, the SBCD approach enables different loss functions for the data fidelity term, like, e.g., the Huber M-estimator, robust to outlier measurements.

3.3 Closed-form solution

Taking into account the drawbacks of the SBCD approach, we opted to go even deeper in the evaluation of (3.6) and (3.26) and propose a closed-form solution to the problem. Given the previously defined $\mathbf{D}$, let us have the following auxiliary variables

$$\begin{cases} \mathbf{S}_{00} = \mathbf{D}^T(0)\mathbf{D}(0) + \lambda_1\mathbf{I} \\ \mathbf{S}_{N-2,N-2} = \mathbf{D}^T(N-2)\mathbf{D}(N-2) + \lambda_{N-2}\mathbf{I} \\ \mathbf{S}_{kk} = \mathbf{D}^T(k)\mathbf{D}(k) + (\lambda_k + \lambda_{k+1})\mathbf{I} \\ \mathbf{S}_{k,k-1} = \mathbf{S}_{k-1,k} = -\lambda_k\mathbf{I} \\ \mathbf{S}_{k,k+1} = -\lambda_{k+1}\mathbf{I} \\ \boldsymbol{\Theta}_k = \mathbf{D}^T(k)\mathbf{X'}^T(k) \end{cases} . \tag{3.29}$$

Remark 3. *Our closed-form solution, with a complexity that scales linearly with the number of time steps considered stems, from the key observation that the linear system of equations to be solved in* (3.26) *is in fact a block tridiagonal linear system. By considering the LU factorization method, our problem has a closed-form solution with linear complexity [29].*

In the next section, we apply the LU factorization to (3.26). In general, it consists on a forward pass and then a backward pass, in this case with a complexity that scales linearly with the number of time

Algorithm 1 Stochastic Block Coordinate Descent

Input: $\mathbf{D}$, $\mathbf{X'}$, $[\lambda_k]_{k=1}^{k=N-2}$, ε
Output: $\mathbf{C}^*$

$\mathbf{C_0} \leftarrow$ initialized with random values
$\mathbf{S}_{00} \leftarrow \mathbf{D}^T(0)\mathbf{D}(0) + \lambda_1\mathbf{I}$
$\mathbf{S}_{N-2,N-2} \leftarrow \mathbf{D}^T(N-2)\mathbf{D}(N-2) + \lambda_{N-2}\mathbf{I}$
for $k \in [1, ..., N-3]$ **do**
    $\mathbf{S}_{kk} \leftarrow \mathbf{D}^T(k)\mathbf{D}(k) + (\lambda_k + \lambda_{k+1})\mathbf{I}$
end for

for $k \in [0, ..., N-2]$ **do**
    $\mathbf{\Theta}_k \leftarrow \mathbf{D}^T(k)\mathbf{X'}^T(k)$
    $\boldsymbol{\nabla}\mathbf{h_0}(k) \leftarrow \mathbf{D}^T(k)(\mathbf{D}(k)^T\mathbf{C}_0(k) - \mathbf{X'}^T(k))$
end for

$\boldsymbol{\nabla} g_0 \leftarrow \mathbf{F}^T\boldsymbol{\Upsilon}\mathbf{F}\mathbf{C}_0$
$\boldsymbol{\nabla} f_0 \leftarrow \boldsymbol{\nabla} h_0 + \boldsymbol{\nabla} g_0$
$t \leftarrow 0$

while $\|\boldsymbol{\nabla} f(\mathbf{C}_t)\|_F^2 > \varepsilon$ **do**
    $RndOrd = random(0, ..., N-2)$
    for $i \in RndOrd$ **do**
        if i is 0 **then**
            $\mathbf{C}_{t+1}(i) \leftarrow \mathbf{\Theta}_{ii}^{-1}(\mathbf{\Theta}_i + \lambda_{i+1}\mathbf{C}_t(i+1))$
        else if i is $N-2$ **then**
            $\mathbf{C}_{t+1}(i) \leftarrow \mathbf{\Theta}_{ii}^{-1}(\mathbf{\Theta}_i + \lambda_i\mathbf{C}_t(i-1))$
        else
            $\mathbf{C}_{t+1}(i) \leftarrow \mathbf{\Theta}_{ii}^{-1}(\mathbf{\Theta}_i + \lambda_i\mathbf{C}_t(i-1) + \lambda_{i+1}\mathbf{C}_t(i+1))$
        end if

        $\boldsymbol{\delta}(i) \leftarrow \mathbf{C}_{t+1}(i) - \mathbf{C}_t(i)$

        $\boldsymbol{\nabla} h_{t+1}(i) \leftarrow \boldsymbol{\nabla} h_t + \mathbf{D}^T(i)\mathbf{D}(i)\boldsymbol{\delta}(i)$
        if i is 0 **then**
            $\boldsymbol{\nabla} g_{t+1}(i) \leftarrow \boldsymbol{\nabla} g_t(i) + \lambda_{i+1}\boldsymbol{\delta}(i)$
            $\boldsymbol{\nabla} g_{t+1}(i+1) \leftarrow \boldsymbol{\nabla} g_t(i+1) - \lambda_{i+1}\boldsymbol{\delta}(i)$
        else if i is $N-2$ **then**
            $\boldsymbol{\nabla} g_{t+1}(i-1) \leftarrow \boldsymbol{\nabla} g_t(i-1) - \lambda_i\boldsymbol{\delta}(i)$
            $\boldsymbol{\nabla} g_{t+1}(i) \leftarrow \boldsymbol{\nabla} g_t(i) + \lambda_i\boldsymbol{\delta}(i)$
        else
            $\boldsymbol{\nabla} g_{t+1}(i-1) \leftarrow \boldsymbol{\nabla} g_t(i-1) - \lambda_i\boldsymbol{\delta}(i)$
            $\boldsymbol{\nabla} g_{t+1}(i) \leftarrow \boldsymbol{\nabla} g_t(i) + (\lambda_i + \lambda_{i+1})\boldsymbol{\delta}(i)$
            $\boldsymbol{\nabla} g_{t+1}(i+1) \leftarrow \boldsymbol{\nabla} g_t(i+1) - \lambda_{i+1}\boldsymbol{\delta}(i)$
        end if

        for $i \in [i-1, i, i+1]$ **do**
            $\boldsymbol{\nabla} f_{t+1} \leftarrow \boldsymbol{\nabla} h_{t+1} + \boldsymbol{\nabla} g_{t+1}$ ▷ Only the necessary
        end for
        $t \leftarrow t+1$
    end for
end while
    return $\mathbf{C}^* \leftarrow \mathbf{C}_t$

steps considered in the system model.

Forward pass

For the forward pass, we solve $\mathbf{LY} = \mathbf{\Theta}$, where $\mathbf{L}$ is

$$\mathbf{L} = \begin{bmatrix} \mathbf{\Lambda}_0 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{1,0} & \mathbf{\Lambda}_1 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_{N-3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}_{N-2,N-3} & \mathbf{\Lambda}_{N-2} \end{bmatrix},$$

and $\mathbf{\Lambda}_k$ and $\mathbf{\Theta}_{k,\ell}$ are square submatrices of size $(p+q) \times (p+q)$. To start the process, the unknown variables are initialized with

$$\begin{cases} \mathbf{\Lambda}_0 = \mathbf{S}_{00} \\ \mathbf{Y}_0 = \mathbf{\Lambda}_0^{-1} \mathbf{\Theta}_0 \end{cases},$$

and for the subsequent time steps $k = 1, ..., N-2$, we incrementally compute

$$\begin{cases} \mathbf{\Omega}_k = \mathbf{S}_{k,k-1} \mathbf{\Lambda}_{k-1}^{-1} \mathbf{S}_{k-1,k} \\ \mathbf{\Lambda}_k = \mathbf{S}_{kk} - \mathbf{\Omega}_k \\ \mathbf{Y}_k = \mathbf{\Lambda}_k^{-1} (\mathbf{\Theta}_k - \mathbf{S}_{k,k-1} \mathbf{Y}_{k-1}). \end{cases} \tag{3.30}$$

Backward pass

Given the results of the forward pass, we can address the backward propagation by solving $\mathbf{MC} = \mathbf{Y}$, with

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{\Lambda}_0^{-1} \mathbf{S}_{0,1} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{\Lambda}_{N-3}^{-1} \mathbf{S}_{N-3,N-2} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{3.31}$$

The optimization variable in the last instant is initialized with $\mathbf{C}(N-2) = \mathbf{Y}_{N-2}$. For $i = N-3, ..., 0$,

$$\mathbf{C}(k) = \mathbf{Y}_k - \mathbf{\Lambda}_k^{-1} \mathbf{S}_{k,k+1} \mathbf{C}(k+1). \tag{3.32}$$

Next, we state the main result of this work, about the convergence in a fixed, finite number of algebraic operations enumerated in Algorithm 2, i.e., as a closed-form.

Theorem 3. *[Algorithm 2 solves* (3.21) *in closed-form with linear complexity]*

Algorithm 2 yields a closed-form solution for the system identification problem of the linear time-variant system from data, with linear complexity on the number of time steps. Namely, the number of

Algorithm 2 COSMIC - closed-form system identification

Input: $\mathbf{D}$, $\mathbf{X'}$, $[\lambda_k]_{k=1}^{k=N-2}$
Output: $\mathbf{C}^*$

$\mathbf{S}_{00} \leftarrow \mathbf{D}^T(0)\mathbf{D}(0) + \lambda_1\mathbf{I}$
$\mathbf{S}_{N-2,N-2} \leftarrow \mathbf{D}^T(N-2)\mathbf{D}(N-2) + \lambda_{N-2}\mathbf{I}$
for $k \in [1, ..., N-3]$ **do**
    $\mathbf{S}_{kk} \leftarrow \mathbf{D}^T(k)\mathbf{D}(k) + (\lambda_k + \lambda_{k+1})\mathbf{I}$
end for

for $k \in [0, ..., N-2]$ **do**
    $\mathbf{\Theta}_k \leftarrow \mathbf{D}^T(k)\mathbf{X'}^T(k)$
end for

$\mathbf{\Lambda}_{00} \leftarrow \mathbf{S}_0$
$\mathbf{Y}_0 \leftarrow \mathbf{\Lambda}_0^{-1}\mathbf{\Theta}_0$

for $k \in [1, ..., N-2]$ **do** ▷ forward pass
    $\mathbf{\Lambda}_k \leftarrow \mathbf{S}_{kk} - \lambda_k^2\mathbf{\Lambda}_{k-1}^{-1}$
    $\mathbf{Y}_k \leftarrow \mathbf{\Lambda}_k^{-1}(\mathbf{\Theta}_k + \lambda_k\mathbf{Y}_{k-1})$
end for

$\mathbf{C}(N-2) \leftarrow \mathbf{Y}_{N-2}$

for $k \in [N-3, ..., 0]$ **do** ▷ backward pass
    $\mathbf{C}(k) \leftarrow \mathbf{Y}_k + \lambda_{k+1}\mathbf{\Lambda}_k^{-1}\mathbf{C}(k+1)$
end for
    return $\mathbf{C}^* \leftarrow \mathbf{C}$

multiplication operations to be performed by Algorithm 2 is

$$c = (N-1)\left((p+q)^3 + (2p+3)(p+q)^2\right), \tag{3.33}$$

which is linear on the number of time steps N, and cubic on the size of state space and control space, and does not depend on L, the size of the dataset used for learning matrices $[\mathbf{A}(k), \mathbf{B}(k)]_{k=0}^{N-2}$.

Proof. We must prove that solving $\nabla f(\mathbf{C}) = 0$ is equivalent to solving

$$\mathbf{LMC} = \mathbf{\Theta}, \tag{3.34}$$

where $\mathbf{L}$ is a lower triangular matrix with non zero diagonal and first subdiagonal blocks, and $\mathbf{M}$ is an upper triangular matrix with diagonal identity submatrices and first superdiagonal non-zero.

Let us assume a system starting from the results in (3.26). We can rearrange the system of linear equations to address it as

$$\underbrace{\begin{bmatrix} \mathbf{D}^T(0)\mathbf{D}(0)\mathbf{C}(0) + \lambda(-\mathbf{C}(0) + \mathbf{C}(1)) \\ \vdots \\ \mathbf{D}^T(k)\mathbf{D}(k)\mathbf{C}(k) + \lambda(-\mathbf{C}(k-1) + 2\mathbf{C}(k) - \mathbf{C}(k+1)) \\ \vdots \\ \mathbf{D}^T(N-2)\mathbf{D}(N-2)\mathbf{C}(N-2) + \lambda(-\mathbf{C}(N-3) + \mathbf{C}(N-2) \end{bmatrix}}_{\mathbf{LMC}} = \underbrace{\begin{bmatrix} \mathbf{D}^T(0)\mathbf{X'}^T(0) \\ \vdots \\ \mathbf{D}^T(k)\mathbf{X'}^T(k) \\ \vdots \\ \mathbf{D}^T(N-2)\mathbf{X'}^T(N-2) \end{bmatrix}}_{\mathbf{\Theta}},$$

which directly yields the definition of $\mathbf{\Theta}$, just like it was defined in (3.29). Decomposing the LHS of the previous definition, we can identify the dependence on $\mathbf{C}$ and single it out. Thus, we get

$$\underbrace{\begin{bmatrix} \mathbf{S}_{00} & -\lambda_1\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \ddots & \ddots & \vdots & & \\ \ddots & -\lambda_k\mathbf{I} & \mathbf{S}_{kk} & -\lambda_{k+1}\mathbf{I} & \ddots \\ & & \vdots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\lambda_{N-2}\mathbf{I} & \mathbf{S}_{N-2,N-2} \end{bmatrix}}_{\mathbf{LM}} \begin{bmatrix} \mathbf{C}(0) \\ \vdots \\ \mathbf{C}(k) \\ \vdots \\ \mathbf{C}(N-2) \end{bmatrix}. \tag{3.35}$$

Additionally, from the assumption of the form of matrices $\mathbf{L}$ and $\mathbf{M}$, as previously defined, we get

$$\mathbf{LM} = \begin{bmatrix} \mathbf{\Lambda}_0 & \mathbf{S}_{01} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{S}_{01} & \mathbf{\Lambda}_1 + \mathbf{\Omega}_1 & \mathbf{S}_{12} & \vdots & \vdots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \mathbf{S}_{k,k-1} & \mathbf{\Lambda}_k + \mathbf{\Omega}_k & \mathbf{S}_{k,k+1} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \mathbf{S}_{N-4,N-3} & \mathbf{\Lambda}_{N-3} + \mathbf{\Omega}_{N-3} & \mathbf{S}_{N-3,N-2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{S}_{N-2,N-3} & \mathbf{\Lambda}_{N-2} + \mathbf{\Omega}_{N-2} \end{bmatrix}, \tag{3.36}$$

yielding the results we stated in the formulation of the problem by making it equal to (3.35). Hence, we prove that the problem can be solved by a LU factorization and that solving equation (3.6), taking into account the nuances of the k-th instants, is equivalent to solving (3.34).

For the complexity result, we analyze the multiplication operations involved in the presented algorithm as follows. For the *forward pass* we need to invert all $N-1$ $\mathbf{\Lambda}_k$ matrices. If we use traditional inversion methods, the number of operations for each inversion is $(p+q)^3$. Multiplications by a scalar involve $2(p+q)^2$. The multiplication of the two matrices in Step 11 of Algorithm 2, $\mathbf{\Lambda}_k^{-1}$, with a matrix $(p+q)\times p$ costs $p(p+q)^2$, gives a total number of operations for all $N-1$ steps of

$$c_{\text{fwd}} = (N-1)\left((p+q)^3 + (p+2)(p+q)^2\right),$$

while the *backward pass* demands multiplication of a scalar by a matrix of size $(p+q)\times(p+q)$ accounting

for $(p+q)^2$ operations, and a multiplication between a matrix of size $(p+q)\times(p+q)$, and one of size $(p+q)\times p$, yielding $p(p+q)^2$ operations. Thus, the total number of backward pass operations is

$$c_{\text{bwd}} = (N-1)\left((p+1)(p+q)^2\right).$$

□

3.3.1 Preconditioning

To account for eventual irregularities in the data collected that may difficult the mathematical operations needed to compute a solution, namely the multiple matrices inverses that need to be calculated, we perform a preconditioning process when necessary. The proposed procedure can be compacted by a new definition of matrices $\mathbf{S}$ and $\mathbf{\Theta}$ that regulate the formulation of the upper and lower triangular matrices [30]. As such, indicated with the superscript $\mathbf{PC}$, we redefine (3.29) and get

$$\begin{cases} \mathbf{S}^{\mathbf{PC}}{}_{kk} = \mathbf{I} \\ \mathbf{S}^{\mathbf{PC}}{}_{i,j} = \mathbf{S}_{ii}^{-1}\mathbf{S}_{i,j} \quad \text{for } i \neq j \\ \mathbf{\Theta}^{\mathbf{PC}}{}_{k} = \mathbf{S}_{kk}^{-1}\mathbf{\Theta}_{k} \end{cases}. \tag{3.37}$$

Applying these new formulation to Algorithm 2, we get a more robust solution to the problem, circumventing numerical destabilization caused by large matrix condition numbers. For the purpose of this work, we found that it was not necessary to apply preconditioning to the data. We note that applying preconditioning will change the result of Theorem 3, although not changing the linear complexity in N.

3.4 COSMIC performance evaluation

3.4.1 Setup

To demonstrate previous stated theoretical results, a simulation and testing environment was developed, using Simulink and Python. The schematic in 3.3 presents the flow of the tests performed and the validation process. The green blocks represent previous knowledge the user generates, while the yellow blocks are collected and then fed to the COSMIC. In blue blocks we have the outputs that are feasible to be analyzed and will dictate the performance of the algorithm.

The system is first simulated and excited in a Simulink model to allow for data collection and it is the one described in Section 2.3, both in LTI and LTV setups, and to guarantee sufficient data we input a sine function with different amplitude for each run of the simulation, as shown in Figure 3.4. It is also worth noting that in each simulation run, the initial conditions take a different value. For the purpose of these tests, state variables could take any value between 0 m and 6 m and 0 m/s and 6 m/s, respectively. The validation is performed with parameter λ_k constant throughout the trajectory, as it would yield a heavy parametric study and, for this particular system, we found the usage of different conditions for different

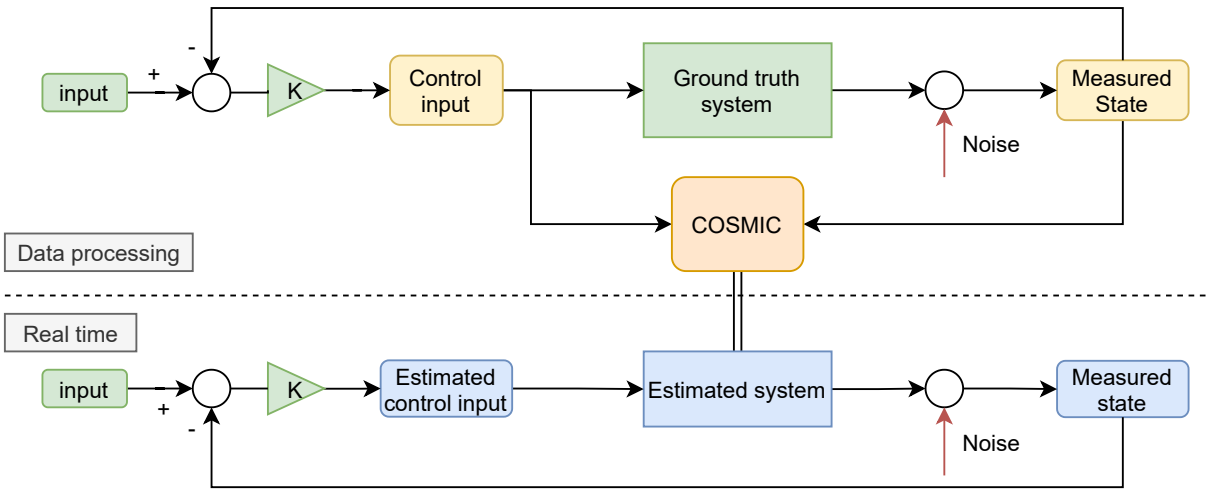


Figure 3.3: Schematic representing the validation procedure for performance evaluation.

parts of the trajectory not to be particularly constructive, thus being represented by λ.

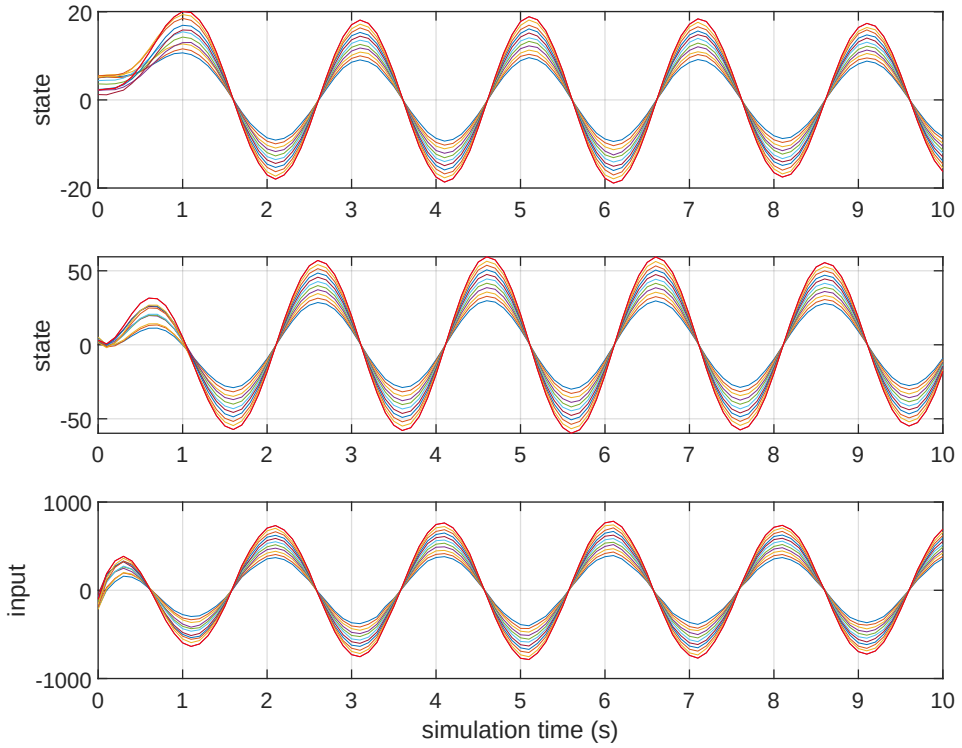


Figure 3.4: Example of the data collected, each color is one of the different L simulations.

Afterwards, this data is used as input into COSMIC and multiple models are trained, in a Python environment, to understand the behavior of the proposed algorithm in different settings, by varying multiple parameters, such as the λ or the noise that disturbs the state.

Finally, presented in the schematic 3.3 as the real time component of the workflow, the estimated

model is excited with previously selected inputs from which we know the expected result, and we are able to evaluate the performance. The metrics used are the estimation error and the prediction error.

Taking into account the formerly defined system, once again from Section 2.3, we have a ground truth that we can compare our results to, both for the LTI system, which is known for all instants, and for the LTV system, that is also computed in the Simulink model and then input to the Python environment. The estimation error is defined as

$$\|\hat{\mathbf{C}} - \mathbf{C}_{\mathbf{gt}}\|_F \tag{3.38}$$

and indicates how close the model is to the true system. We can also verify the evolution of this metric over time.

Regarding the prediction power of COSMIC, we propose the evaluation of the predicted state error when compared to the true state from a simulated trajectory that was not part of the training data set. As such, we address the metric by defining the norm of the error as

$$\|\hat{\mathbf{x}}(k+1) - \mathbf{x}(k+1)\|_2 = \|\hat{\mathbf{A}}(k)\mathbf{X}^T(k) + \hat{\mathbf{B}}(k)\mathbf{U}^T(k) - \left(\mathbf{A}(k)\mathbf{X}^T(k) + \mathbf{B}(k)\mathbf{U}^T(k)\right)\|_2, \tag{3.39}$$

allowing for the assessment of the error propagation. Moreover, for the following validation study, we assumed that the measurements were disturbed by noise, which can be characterized by $\omega \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = 0$ and variance σ^2.

3.4.2 Parametric study

To infer how the performance of the algorithm depends on the measurement noise and parameter λ, we opted to develop a parametric study where, for each noise variance, we tested the estimation for a reasonable range of values for parameter λ.

To characterize COSMIC's effectiveness, we start by the estimation error, from (3.38), for each pair of parameter λ and variance. We perform this experiment for the LTI and LTV settings and the results are expressed in tables 3.1 and 3.2.

	λ	1	10	100	10^3	10^4	10^5
σ^2	0.60^2	190.230	108.367	63.092	35.751	9.403	0.826
	0.10^2	62.131	41.223	16.489	1.806	0.042	0.0007
	0.06^2	44.659	24.618	5.672	0.343	0.006	7.826×10^{-5}
	0.03^2	22.258	6.518	0.688	0.026	0.0004	4.256×10^{-6}
	0.01^2	2.200	0.209	0.012	0.0004	5.654×10^{-6}	5.844×10^{-7}

Table 3.1: Error between estimated $\hat{\mathbf{C}}$ and the ground truth evolution with noise and parameter λ for the LTI system.

Comparing the results from each experiment, it is clear that there is not much difference between the results for the LTI and the LTV trials, which indicates the good performance of the algorithm but also that the noise interferes with the invariance of the LTI system. It is interesting to verify that as the parameter λ increases, the estimation error for the LTI becomes smaller than the estimation error of the LTV, which

	λ	1	10	100	10^3	10^4	10^5
σ^2	0.60^2	185.009	107.004	63.007	35.703	9.642	0.871
	0.10^2	64.301	42.793	16.992	1.982	0.053	0.003
	0.06^2	47.144	25.508	5.860	0.385	0.008	0.002
	0.03^2	23.376	6.611	0.701	0.031	0.001	0.001
	0.01^2	2.275	0.206	0.012	0.0007	0.0003	0.001

Table 3.2: Error between estimated $\hat{\mathbf{C}}$ and the ground truth evolution with noise and parameter λ for the LTV system.

confirms the theoretical inferences, because higher values of λ implicate less variation between instants, which is closer to the dynamics of the time-invariant system. The evaluation from this point forward is only presented for the LTV system, even if it was performed for both settings.

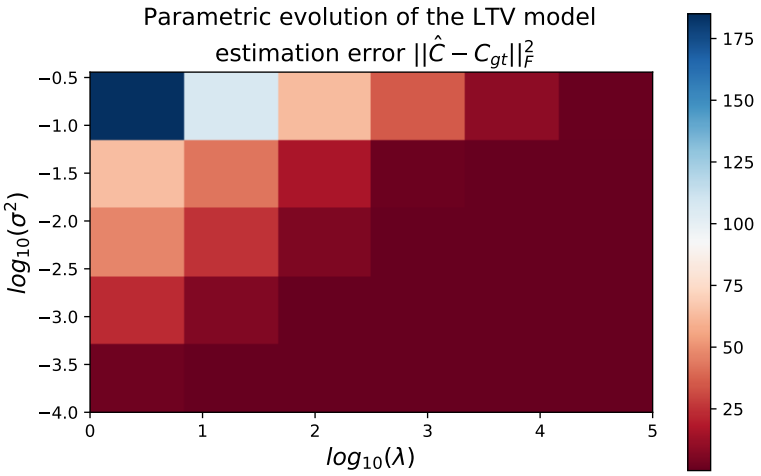
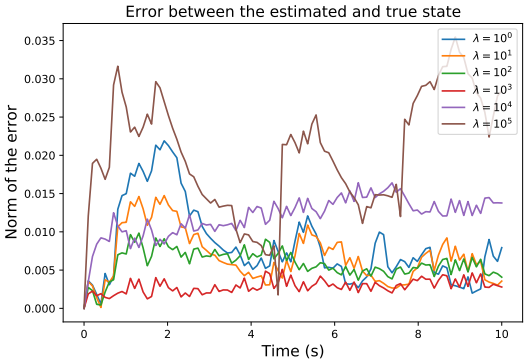


Figure 3.5: State estimation performance of the multiple methods tested, compared with the true state.

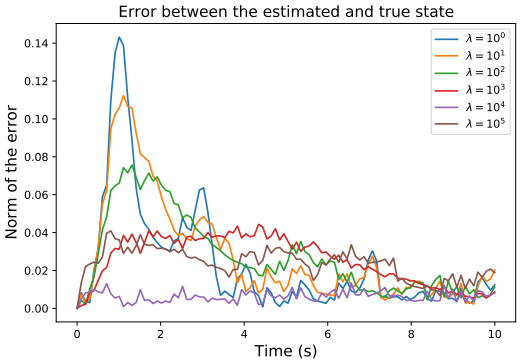
Figure 3.5 exposes the variation of the estimation error with the noise added to the measurements for the LTV system, summarizing the results from Table 3.2, and it is clear that for a noise standard deviation of about 10% of the maximum initial conditions, the algorithm is heavily affected. Reducing this value, the estimation error drops significantly. Once again, it is evident that smaller values for the parameter λ lead to a worse performance of the algorithm, as the variation between instants is not significant enough to allow such values, resulting much better with higher values of λ.

Regarding the prediction power, we used the models estimated to derive the results from Table 3.2 and Figure 3.5 to compute a new trajectory and evaluate the state error, by inputting a previously known trajectory not disturbed by noise, to assess how much it disturbed the prediction. As expected, the results show higher prediction errors for the higher variance, as evident in Figure 3.6.

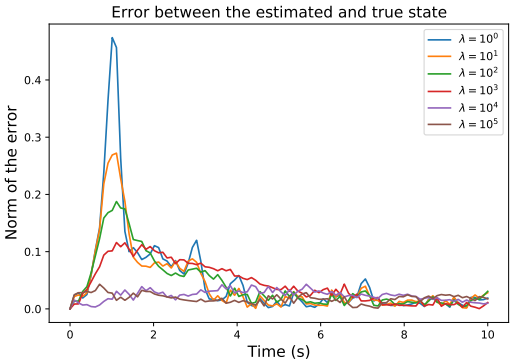
Moreover, this test made explicit certain areas where the prediction power decreases (because the state error increases) and these are congruent for all the tests performed. From the *a priori* knowledge



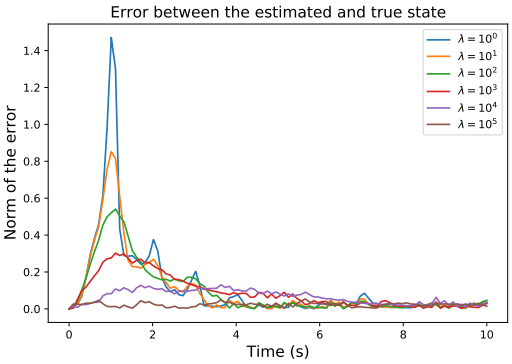
(a) $\sigma^2 = 0.01^2$



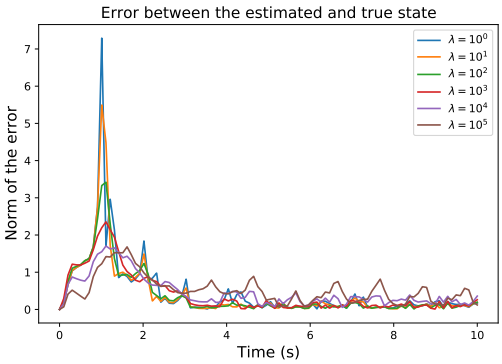
(b) $\sigma^2 = 0.03^2$



(c) $\sigma^2 = 0.06^2$



(d) $\sigma^2 = 0.10^2$



(e) $\sigma^2 = 0.60^2$

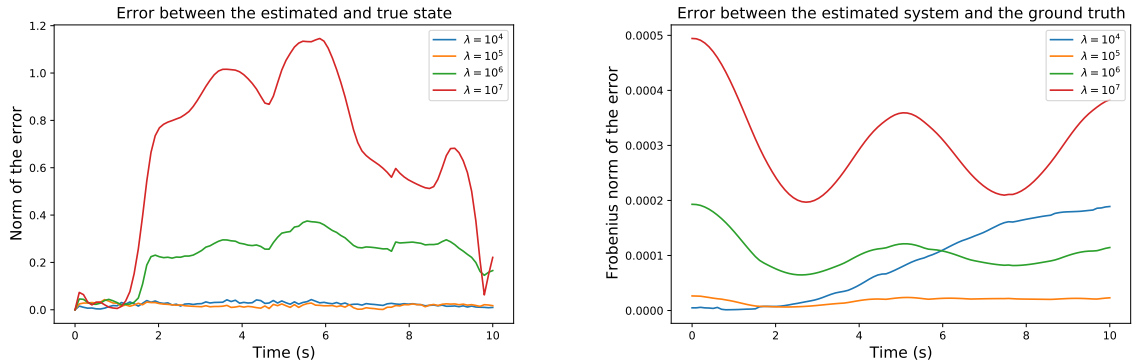
Figure 3.6: Prediction error variation with noise variance, for different variances, with λ, considering the LTV system.

we have about the simulated system, it can be inferred that these peaks appear around the point the transition between the transient state and the steady state occurs. Increasing the variance of the noise, the value for said peaks increases significantly.

Taking the previous conclusions into account, the following evaluations will be performed considering that the measurements noise follows a Gaussian distribution with zero mean and variance $\sigma^2 = 0.06^2$, representing about 1% of the initial values for the state, which is a fair assumption.

As part of the validation process, we can also evaluate what is the λ that best adapts the estimated model to the true system characteristics. To do so, we evaluate the system in the previous conditions,

with the chosen value for the noise variance. Once again the estimation error and the prediction power were evaluated and the statistical measures were used to make a choice of most fitting λ.



(a) Estimation error.

(b) Prediction error.

Figure 3.7: Evaluation metrics evolution with time to infer the optimal $\lambda = 10^5$.

From a simple analysis of the plots in Figure 3.7, we can say that the λ that best fits the system we are trying to estimate is $\lambda = 10^5$, as the lowest errors occur at this value.

3.5 Benchmark

The novelty of the algorithm presented comes from its efficiency, mainly when compared to other approaches to the same problem.

As we are solving a convex optimization problem, we can obtain a solution through `cvxpy`, a domain specific language that allows for easy problem syntax to express convex optimization problems in Python and is widely used [31]. For this specific use case, the ECOS solver [32], an interior-point solver for second-order cone programming, is automatically invoked. Hence, we evaluate the problem agnostic approach with this strategy.

In addition, we are also going to be evaluating the solution of (3.21) through a SBCD algorithm, as presented in Algorithm 1. Finally, we run again the COSMIC algorithm. These simulations, as all the ones run in a Python environment, were made in Google Colab development setting, with 13GB of RAM available and the Intel(R) Xeon(R) CPU @ 2.20GHz processor.

Table 3.3: Performance comparison of different system identification solutions.

Instances per trajectory	`cvxpy`		***SBCD***		***closed-form***	
	Cost	**Time (s)**	**Cost**	**Time (s)**	**Cost**	**Time (s)**
100	100.960	2.929	100.960	0.031	100.960	0.014
1000	971.070	42.588	971.070	0.273	971.070	0.106
10000	*	*	9710.431	3.531	9710.431	1.017
100000	*	*	97303.028	21.055	97303.028	9.696

*Session crashed after using all available RAM.

The benchmarking metrics are the cost function and the computation time. The cost function represents how close the solution found by an algorithm is to the information collected from the data and the time it takes to achieve a solution is a direct indicator of the computational power needed to make the calculations. To perform the comparison in Table 3.3, as we are studying the performance of the methods against cost and time, we simulated a 1/10 sampling rate trajectory and varied its length between 100 s, 1000 s, 10000 s and 100000 s with $\lambda = 0.001$ and stopping the SBCD at 1 million iterations, while using the setting presented previously.

The `cvxpy` approach tries to solve the problem for all time steps simultaneously, leading to a solution that requires the machine to invert a $(N-1)(p+q) \times (N-1)(p+q)$ matrix, as stated in (3.7). As the number of instances per trajectory increases, the complexity also increases and the `cvxpy` approach can no longer compute a solution in due course, while both SBCD and closed-form approaches continue to perform as expected.

Moreover, for more disturbed data, the computation needed to reach this solution becomes more difficult. Regarding the cost function from Table 3.3, no significant changes are observed from one method to another, indicating that all are reaching the desired solution, different only on the time that it takes to reach it.

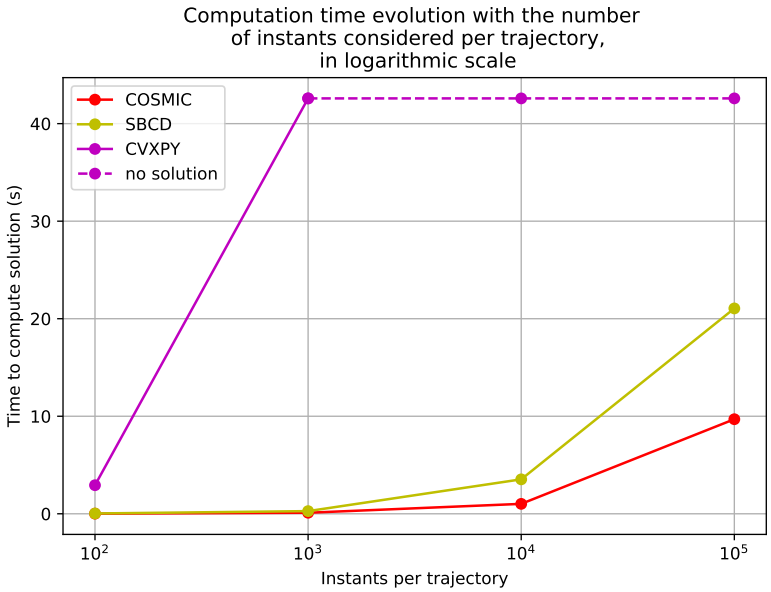


Figure 3.8: Computation time comparison of different system identification solutions.

This leads to the conclusion that the closed-form algorithm is the best option to solve (3.21), as it can achieve minimal cost values with an efficient use of computational resources, reaching a solution for all cases tested, performing significantly better than the other approaches. Moreover, from Figure 3.8, which takes into account a logarithmic scale in the x axis to ease the plot interpretation we can experimentally observe the linearity with the number of instances per trajectory that is stated in Theorem 3.

Furthering the analysis, to account for the optimal λ, we performed the previous analysis for $\lambda = 10^5$, which led to new findings regarding the high dependence of the convergence of the SCBD on the value of λ. As the results for the problem agnostic approach are the same, we opt to analyze the SBCD in

detail, in comparison to our proposed closed-form solution.

Table 3.4: Performance comparison to make evident SBCD dependence on λ.

Instances per trajectory	***SBCD limited***		***closed-form***	
	Cost	**Time (s)**	**Cost**	**Time (s)**
100	170.094	90.626	150.403	0.011
1000	2663.71690	61.905	1521.0506	0.120

As presented in Table 3.4, when the λ is too high, the SBCD does not compare to the performance of the closed-form algorithm, as it is to be expected, given the dependence on the derivative calculation, which ends up always being too big to end the loop. To overcome this interference, we could adjust accordingly the stopping criterion. However, this would not guarantee an optimal solution, as the derivative would not be equal to zero.

To better understand this behavior, we can notice than on 3.27, if the value of λ is big enough, the data dependent terms can become very small and the update of the optimization variable will just be the propagation of the neighboring values. Thus, the solution will be heavily dependent on the initialization of the optimization variable. In conclusion, we can state that the SBCD is too dependent on previous knowledge of the behavior of the data, given that we need to adjust the value of ϵ and the initialization of $\mathbf{C}$ accordingly to have a viable solution.

Chapter 4

Controller design for estimated model

To follow up on the system identification from data, we derive an optimal control law for the LTV estimated model based on a dynamic programming setup. We start with the presentation of the dynamic programming algorithm we chose to apply. Then, regarding the evaluation of its application, we study the design parameters that best fit the system and address the advantages of the controller developed against the ground truth of the spring-mass-damper system and the previously used controller, built from an LTI system. We regard the N time instants trajectory from the previous chapter and all relevant variables are to be considered as defined before.

4.1 Proposed solution

The optimal controller design for the LTV system estimated in Chapter 3 can be addressed as a dynamic programming problem. As an LTV system, we need to consider different optimal control laws for all instants, reaching an optimal control path. This requirement fits perfectly with the Principle of Optimality, which states that, given a set of initial conditions, if the optimal solution for a problem passes through a point, the solution with the said point as the initial condition must also be the optimal one [33]. The Principle of Optimality is the basis for the dynamic programming setting and the work presented in this Chapter.

With the goal of minimizing the cost function (2.12), we start by addressing the issue from the last point of the optimal solution and state that the minimal cost for this time step is $J^*_{N-1} = \frac{1}{2}\mathbf{x}^T_{N-1}\mathbf{H}\mathbf{x}_{N-1}$. Then, we go backwards by addressing one more step of the optimal path each time step.

Once again considering a convex optimization problem, the optimal control input for a trajectory at $N-2$ is the one that leads to a solution to the problem

Problem 4.

$$\underset{\mathbf{u(N-2)}}{\textit{minimize}} \quad J_{N-2} = c(\mathbf{x}(N-2), \mathbf{u}(N-2)) + J^*_{N-1}. \tag{4.1}$$

It is then derived directly from (3.6) and the knowledge about the system, i.e.

$$\mathbf{x}(N-1) = \mathbf{A}(N-2)\mathbf{x}(N-2) + \mathbf{B}(N-2)\mathbf{u}(N-2)$$

that the solution is

$$\mathbf{u}^*(N-2) = -\left(\mathbf{R}_{N-2} + \mathbf{B}^T(N-2)\mathbf{H}\mathbf{B}(N-2)\right)^{-1}\mathbf{B}_{N-2}^T\mathbf{H}\mathbf{A}(N-2)\mathbf{x}(N-2). \tag{4.2}$$

From (4.2), we can define a new variable that will be referred to as the optimal gain, in this particular statement for instant $N-2$

$$\mathbf{K}_{N-2} = \left(\mathbf{R}_{N-2} + \mathbf{B}^T(N-2)\mathbf{H}\mathbf{B}(N-2)\right)^{-1}\mathbf{B}^T(N-2)\mathbf{H}\mathbf{A}(N-2). \tag{4.3}$$

$\mathbf{u}_{N-2}^*$ is the best control action at time $N-2$ and the optimal cost associated with it is

$$\begin{aligned} J_{N-2}^*[\mathbf{x}(N-2)] = \frac{1}{2}\mathbf{x}^T(N-2)[&\mathbf{Q}_{N-2} + \mathbf{K}_{N-2}^T\mathbf{R}_{N-2}\mathbf{K}_{N-2} + \\ &\left(\mathbf{A}(N-2) - \mathbf{B}(N-2)\mathbf{K}_{N-2}\right)^T\mathbf{P}_{N-1}\left(\mathbf{A}(N-2) - \mathbf{B}(N-2)\mathbf{K}_{N-2}\right)]\mathbf{x}(N-2). \\ = \frac{1}{2}&\mathbf{x}^T(N-2)\mathbf{P}_{N-2}\mathbf{x}(N-2), \end{aligned} \tag{4.4}$$

From (4.4), we can additionally outline

$$\begin{aligned} \mathbf{P}_{N-2} =&\mathbf{Q}_{N-2} + \mathbf{K}_{N-2}^T\mathbf{R}_{N-2}\mathbf{K}_{N-2} + \\ &\left(\mathbf{A}(N-2) - \mathbf{B}(N-2)\mathbf{K}_{N-2}\right)^T\mathbf{P}_{N-1}\left(\mathbf{A}(N-2) - \mathbf{B}(N-2)\mathbf{K}_{N-2}\right). \end{aligned} \tag{4.5}$$

If we represent $\mathbf{P}_{N-1} = \mathbf{H}$, we can then find the link between $\mathbf{K}$ and $\mathbf{P}$ and, by induction, solve for the whole length of the problem. We opt not to expand on the induction formulation, as it can be found on [33].

The dynamic programming problem is initialized with $\mathbf{P}_{N-1} = \mathbf{H}$ and the following statements hold true for $k \in [0, ..., N-2]$, starting from the last point of the trajectory

$$\begin{cases} \mathbf{K}_k = \left(\mathbf{R}_k + \mathbf{B}^T(k)\mathbf{P}_{k+1}\mathbf{B}(k)\right)^{-1}\mathbf{B}^T(k)\mathbf{P}_{k+1}\mathbf{A}(k) \\ \mathbf{P}_k = \mathbf{Q}_k + \mathbf{K}_k^T\mathbf{R}_k\mathbf{K}_k + \left(\mathbf{A}(k) - \mathbf{B}(k)\mathbf{K}_k\right)^T\mathbf{P}_{k+1}\left(\mathbf{A}(k) - \mathbf{B}(k)\mathbf{K}_k\right) \end{cases}. \tag{4.6}$$

Algorithm 3 Dynamic Programming for LTV controller design

Input: $\mathbf{H}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{x}$
Output: $\mathbf{K}$

$\mathbf{P}_{N-1} \leftarrow \mathbf{H}$
for $k \in [N-2, ..., 0]$ **do**
  $\mathbf{K}_k \leftarrow \left(\mathbf{R}_k + \mathbf{B}^T(k)\mathbf{P}_{k+1}\mathbf{B}(k)\right)^{-1}\mathbf{B}^T(k)\mathbf{P}_{k+1}\mathbf{A}(k)$
  $\mathbf{P}_k \leftarrow \mathbf{Q}_k + \mathbf{K}_k^T\mathbf{R}_k\mathbf{K}_k + \left(\mathbf{A}(k) - \mathbf{B}(k)\mathbf{K}_k\right)^T\mathbf{P}_{k+1}\left(\mathbf{A}(k) - \mathbf{B}(k)\mathbf{K}_k\right)$
end for
return $\mathbf{K}$

For the purpose of this work, we opt to output the optimal gain sequence and not the optimal control input path, to allow for more flexibility to test the possible applications of the controller designed and

to ease the input in our previously designed simulation environment. Algorithm 3 derives from the adaptation of the work developed in [33] to our scenario.

4.2 Performance evaluation

With the result from Algorithm 3, we can modify the simulation environment described before to account for the new controller at each instant. The main difference stands in the application of the model estimated by COSMIC. While we used it in the control loop to validate throughout Chapter 3, in this setting we input it in the dynamic programming algorithm and use the results in a new control loop with the ground truth system, to validate the controller performance. Figure 4.1 represents the new version of the workflow of the simulations.

Regarding the estimated model, we used the parameters found to be the most suiting in Chapter 3 to develop and save a model, i.e. $\lambda = 10^5$, and the measurements were disturbed by noise with standard deviation $\sigma = 0.06$. Then, we load the new model into another Python environment to allow for it to be used in the dynamic programming algorithm that results in an optimal control path. Afterwards, the solution is returned to Simulink and tested in the original system.

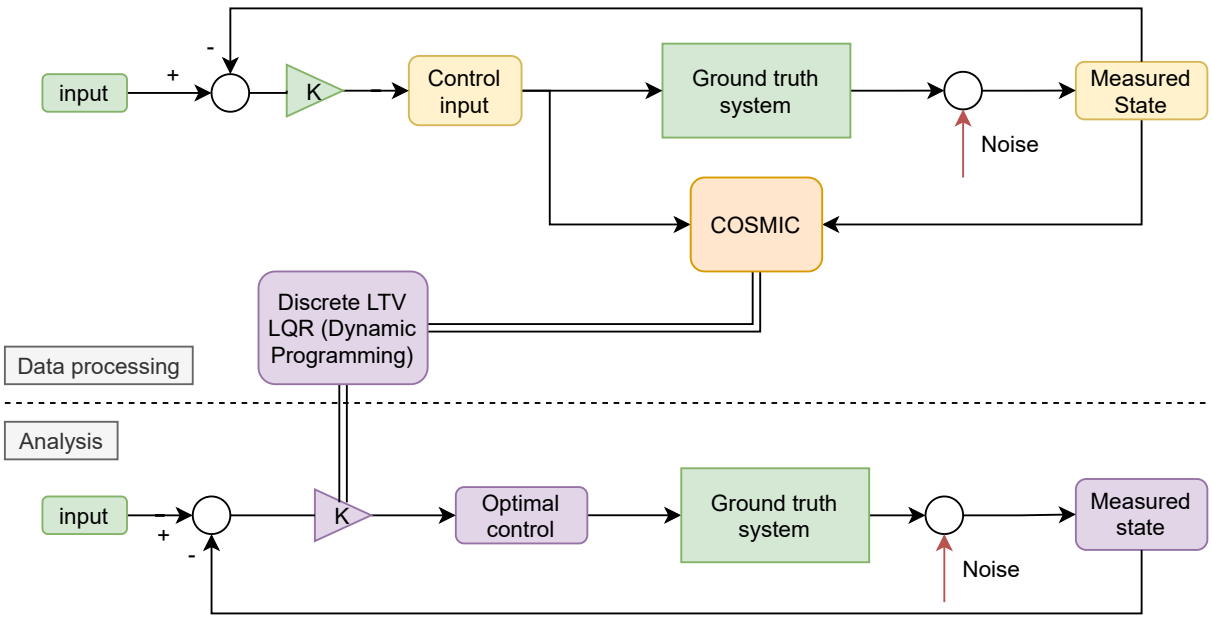


Figure 4.1: Schematic representing the validation procedure for the controller designed.

4.2.1 Design parameters

The design of the controller resides in finding matrices $\mathbf{Q_k}$ and $\mathbf{R_k}$ that best suit the system modeled and result in a better fit concerning the metrics previously defined in Section 2.2.5. Regarding these matrices and to ease controller analysis, we are assuming, without severe loss of performance, that

they are constant throughout the trajectory, from now on just referred to as $\mathbf{Q}$ and $\mathbf{R}$, which significantly reduces the number of design parameters necessary to solve for the optimal control path.

Additionally, we are also going to define $\mathbf{Q}$ and $\mathbf{R}$ as diagonal matrices, which translates in independent weighting for each state variable and control input. Thus, we have, for the problem addressed in this Chapter,

$$\begin{cases} \mathbf{Q} = \begin{bmatrix} q_x & 0 \\ 0 & q_v \end{bmatrix} \\ \mathbf{R} = r \\ \mathbf{H} = \mathbf{Q} \end{cases} . \tag{4.7}$$

Leaving the definition of the design parameters $\mathbf{Q}$ and $\mathbf{R}$ like this, we can rewrite the cost function as

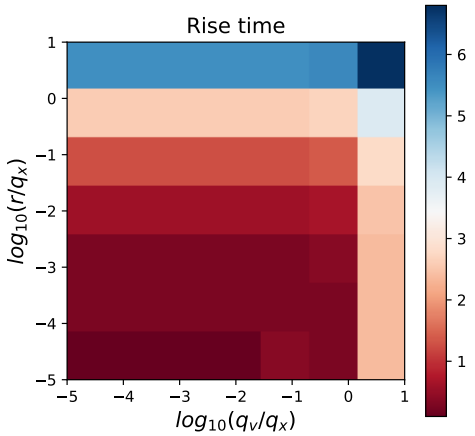
$$\begin{aligned} J &= \frac{1}{2} \sum_{k=0}^{N-1} \mathbf{x}_k^T \begin{bmatrix} q_x & 0 \\ 0 & q_v \end{bmatrix} \mathbf{x}_k + \mathbf{u}_k^T r \mathbf{u}_k \\ &= \frac{1}{2} \sum_{k=0}^{N-2} z_k q_x z_k + \dot{z}_k q_v \dot{z}_k + u_k r u_k \, . \\ &= \frac{1}{2} \sum_{k=0}^{N-2} q_x \left(z_k^2 + \frac{q_v}{q_x} \dot{z}_k^2 + \frac{r}{q_x} u_k^2 \right) \end{aligned} \tag{4.8}$$

From (4.8), we conclude that the cost function minimum is independent from the value of q_x, which for the purpose of the design choices can be assigned as a constant, and the design parameters are reduced to the values of $\frac{q_v}{q_x}$ and $\frac{r}{q_x}$. With this setup, q_x was left as $q_x = 1$ and the ratios were looped through with values from 10^{-6} to 1 to compute multiple control laws, that were then fed to the real time phase of the schematic in 4.1. The results for each pair $\left\{ \frac{q_v}{q_x}, \frac{r}{q_x} \right\}$ are then studied to evaluate the controller performance metrics and the results of this exercise are represented in Figure 4.2.

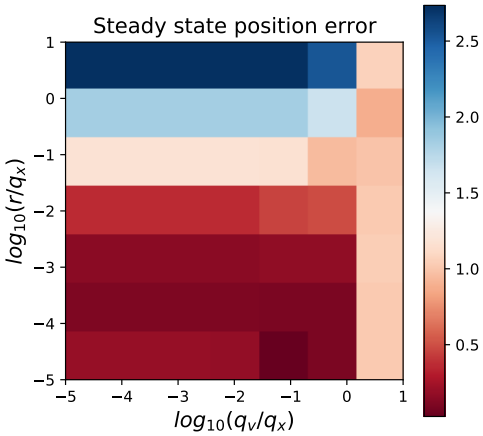
For this part of the work, we study the position tracking effectiveness of the system, while feeding it a null velocity reference. From the previous knowledge of the system, as presented in Section 2.3, we have $z = 1$ m and $\dot{z} = 0$ m/s, such that $\mathbf{x_{ref}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ entailing the following state equation, regarding $\mathbf{K}_k$ as the output of Algorithm 3,

$$\mathbf{x}(k+1) = \mathbf{A_{GT}}(k)\mathbf{x}(k) + \mathbf{B_{GT}}(k)\mathbf{K_k}(\mathbf{x_{ref}} - \mathbf{x}(k)). \tag{4.9}$$

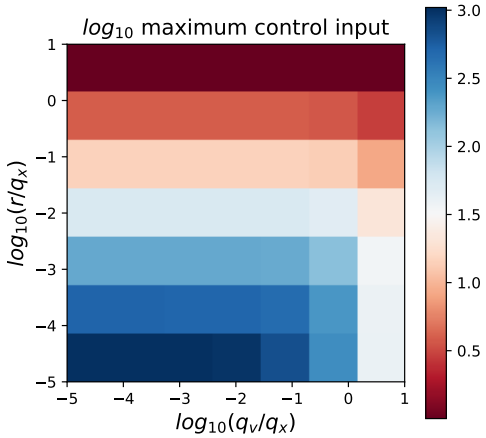
For the purpose of evaluating the best design parameters, initial conditions were kept constant and equal to zero.



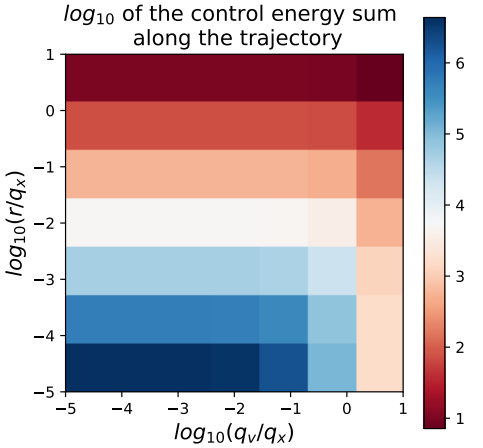
(a) Rise time.



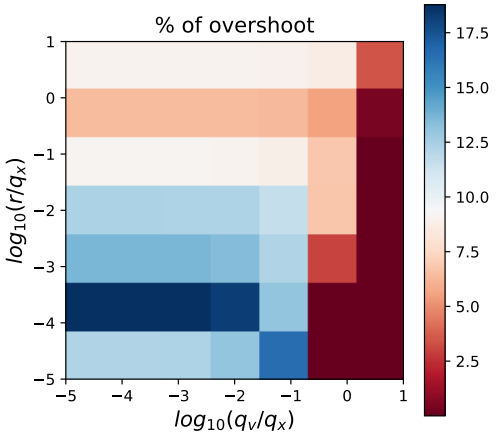
(b) Steady state error.



(c) Maximum control input.



(d) Sum of the control energy.



(e) Overshoot.

Figure 4.2: Controller evaluation metrics for different $\frac{q_v}{q_x}$ and $\frac{r}{q_x}$.

To find the controller that maximizes the performance, we need to minimize all five metrics. However, from the visual inspection of 4.2 (red and blue representing low and high values, respectively) it comes up that this is a contradictory objective for some of the metrics. Rise time and steady state error present lower values when the maximum control input, sum of the control energy and overshoot percentage have higher values.

The rise time presents optimal behavior for small values of q_v and r, increasing as both quantities increase, while the steady state error does not present such a linear conduct, showing its minimum for the lowest value of r but for an increased q_v. Contrarily, the maximum control input and the sum of the control energy have the same evolution with the evolution of q_v and r, having its lowest values for high ratios of $\frac{q_v}{q_x}$ and $\frac{r}{q_x}$, as expected as more focus is given to the control in such setting. Regarding the percentage of overshoot, it is clear that its behavior differs from all the other metrics, having its lowest values for high values of q_v and low values of r, due to the extra focus we give to the velocity in this case, while maintaining a smaller ratio control and output, which results in smaller outputs.

As all overshoot values are below the desired margin, we can establish this metric as less significant for this particular case, given that all values present acceptable margins, as any pair $\left\{\frac{q_v}{q_x}, \frac{r}{q_x}\right\}$ could work. Thus, we wish to find a compromise between the rise time and steady state error optimal design parameters and the maximum input control and sum of control energy ones. To have such a result, we opt to follow the analysis and, as such, design the optimal controller for the spring-mass-damper system with $\frac{q_v}{q_x} = 10^{-1}$, $\frac{r}{q_x} = 10^{-3}$ and $q_x = 1$.

4.2.2 Results

Having found the optimal control law by solving the discrete LTVLQR with dynamic programming, the results are now analyzed and compared to previously used controller design techniques. To perform these tests, random initial conditions are set and vary between 0 and 6, both for position and velocity and the measured state is affected by noise.

For a first approach to the new controller, we test the system ability to do reference tracking, by feeding the system with two different types of inputs:

$$\begin{cases} \mathbf{x_{ref}^{sin}} = \begin{bmatrix} 5\sin(0.5 \times k\Delta t) \\ \cos(0.5 \times k\Delta t) \end{bmatrix} \\ \mathbf{x_{ref}^{cte}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{cases} . \tag{4.10}$$

Figure 4.3 presents the results of this experiment and it can be stated that the controller designed taking into account the estimated model performs as expected and is a good option to control this system, being able to track references easily, converging in less than 10% of the trajectory time and presenting smaller oscillations. Moreover, the changing initial conditions do not affect the final result and the system is able to perform regardless. Furthermore, we can evaluate the system performance comparing it to

two other controllers, to infer about the quality of the model and the quality of the controller.

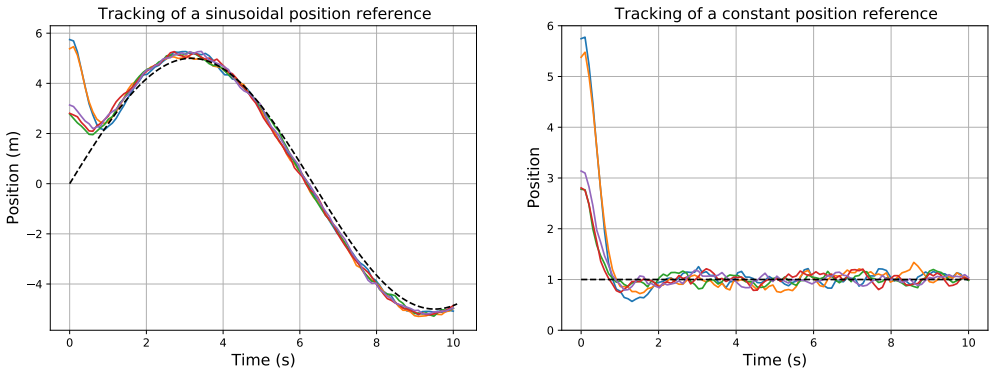


Figure 4.3: System position response from different initial conditions.

At first, we generate one additional controller from the ground truth system, which is then fed to the system and its output is evaluated in contrast with the controller designed with the estimated model, having the same initial conditions. A qualitative assessment from Figure 4.4 shows that the system responds similarly to both controllers and the optimal control path has the same behavior for both options. Hence, the estimated system is able to display the system behavior as well as the ground truth, which once again proves that the COSMIC algorithm is a good option for system identification leading to controller design.

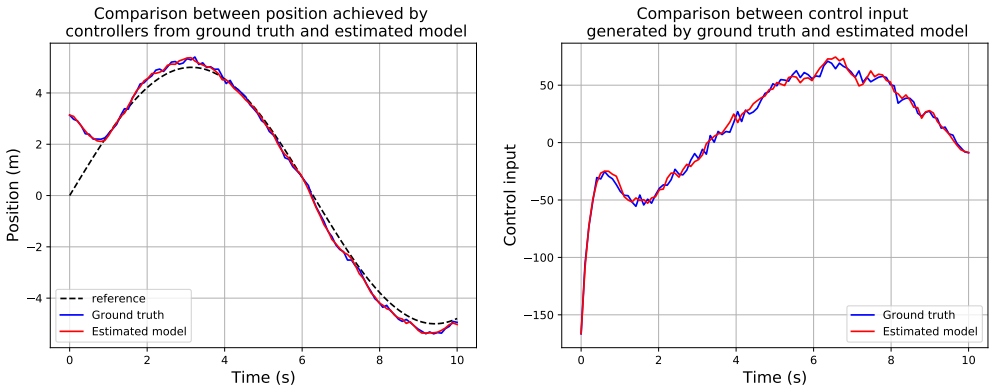


Figure 4.4: System position response to controller generated by the ground truth system and the estimated model.

The last comparison is performed against the controller used for the data collection phase, that is constant throughout the trajectory and does not consider the system specific characteristics, in contrast with the system model estimated by COSMIC or the ground truth. Although presenting a similar system response, as observed in Figure 4.5, we can observe that the controller was not optimal for this system, presenting a considerable difference to the optimal control path that should have been followed. The

control input plot indicates that there might be a compensation from the over performance at the beginning of the trajectory, with smaller than expected inputs from 3.5 s onwards, even if this is not visible in the system reference tracking. We consider that the controller derived from the COSMIC estimated model is a better option for the control of the spring-mass-damper system than the LTILQR.

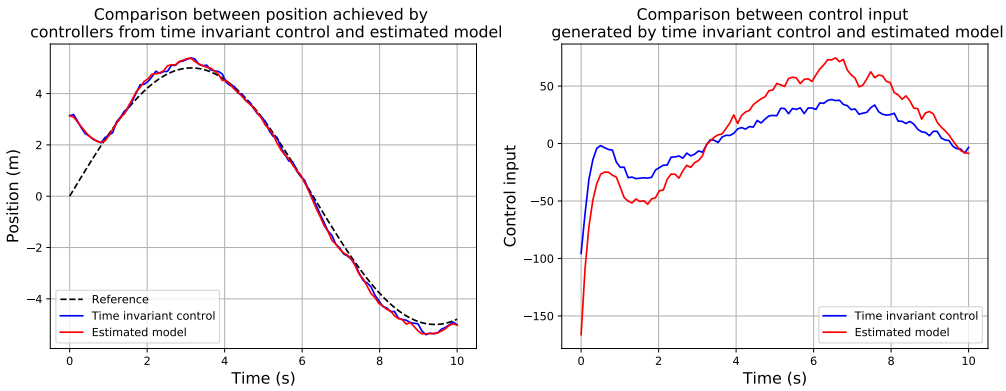


Figure 4.5: System position response to controller generated by the LTILQR and the estimated model.

Finally, after performing a statistical analysis of the tracking error of the system when controlled by controllers designed from the three different strategies formerly presented, we can confidently say that the estimated model is a good approximation of the system, as shown by the close values of the statistical metrics in Table 4.1, which refer to the error between the true trajectory and the trajectories resulting from the application of all three controllers. Moreover, it allows for a better controller synthesis than previously used techniques, performing better than the LTI LQR, presenting a smaller mean and sum of squared error, once again from Table 4.1, which was expected given that the LTI is an approximation for the slowly varying LTV.

Table 4.1: Statistical comparison of the controller design methods by addressing the mean, standard deviation and sum of squared error of the prediction error.

Control	**Mean**	**Standard deviation**	**Sum of squared error**
Estimated model	0.1050	0.7346	0.5506
Ground truth	0.0994	0.7441	0.5635
Time invariant	0.1218	0.7425	0.5661

To conclude this Chapter, it can be stated that the estimated model from COSMIC is a good approximation of the spring-mass-damper system used throughout this work and the dynamic programming strategy worked well for the problem posed. Hence, this system identification and controller design framework is validated and we can further test it in more complex environments.

Chapter 5

Comet Interceptor as a Case Study

This Chapter presents the application of previously discussed solutions to a Space mission environment, namely the Comet Interceptor, in the context it was introduced in Chapter 1. We start by describing the Low Fidelity Simulator where the mission is simulated, from the theoretical properties of the system representation to the functionalities it has. Then, the COSMIC algorithm is used to identify the attitude error dynamics with respect to the nominal mission trajectory and we apply the proceedings from Chapter 4 to develop a new attitude controller and test it in the simulation environment.

Throughout this Chapter, we represent the inertial frame centered in the target as $\{I\}$ and the body fixed frame, centered in the spacecraft, as $\{B\}$. This indication is given as a superscript before the variable and when omitted we consider it to be represented in the body frame $\{B\}$.

5.1 Background

5.1.1 Attitude representation

The representation of the spacecraft attitude is done using unit quaternions, usually represented as

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{q_v} \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}, \tag{5.1}$$

allowing for the use of well-known algebraic manipulation to be used with this quantity. q_w is the scalar part and $\mathbf{q_v}$ is the vector one.

In order to understand the importance of this representation, we first need to address the definition of some basic quaternion operations. The statements made in this section are based on Solá et al [34] and we are only going to present the operations important for the understanding of the work done here and are not straightforward from common mathematical background.

Let us have two unit quaternions

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{q_v} \end{bmatrix} \qquad \text{and} \qquad \mathbf{p} = \begin{bmatrix} p_w \\ \mathbf{p_v} \end{bmatrix}.$$

The quaternion product is

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix} = \begin{bmatrix} p_w q_w - \mathbf{p}_v^T \mathbf{q}_v \\ p_w \mathbf{q}_v + q_w \mathbf{p}_v + [\mathbf{p}_v]_\times \mathbf{q}_v \end{bmatrix}.$$

The skew operator is defined as

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix},$$

representing a skew-symmetric matrix. This encodes the cross-product as in $[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}$ and can be particularly useful for multiplication operations.

The quaternion conjugate is

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ -\mathbf{q_v} \end{bmatrix},$$

the identity is

$$\mathbf{q_1} = 1 = \begin{bmatrix} 1 \\ \mathbf{0_v} \end{bmatrix}$$

and the unit quaternion presents a norm equal to 1, calculated as for a usual vector.

5.1.2 Attitude dynamics and kinematics

The attitude dynamics and kinematics of the spacecraft are described by the general representation of these phenomenons, i.e.

$$\begin{cases} \dot{\mathbf{q}}(t) = \frac{1}{2}\mathbf{q}(t) \otimes \begin{bmatrix} 0 & \boldsymbol{\omega}(t) \end{bmatrix}^T \\ \mathbf{J}\dot{\boldsymbol{\omega}}(t) = [\mathbf{J}\boldsymbol{\omega}(t)] \times \boldsymbol{\omega}(t) - \mathbf{u}(t) + \mathbf{T}(t) \end{cases} \tag{5.2}$$

The attitude is represented by the quaternion $\mathbf{q} = \begin{bmatrix} q_0 & q_x & q_y & q_z \end{bmatrix}^T$, with norm 1 and $\{q_0, q_x, q_y, q_z\} \in \mathbb{R}$, and it transforms a vector represented in the the body fixed frame, centered in the spacecraft, in a vector in the inertial frame centered in the target. The angular velocity is $\boldsymbol{\omega} \in \mathbb{R}^3$. Then, to complete the attitude dynamics, we also define torque input as $\mathbf{u} \in \mathbb{R}^3$ and the external disturbances $\mathbf{T} \in \mathbb{R}^3$, which can be caused by multiple factors but are not going to be considered for the first stage of this application.

$\mathbf{J}$ represents the inertia in the spacecraft body frame. This system is nonlinear.

The pointing error is defined as the angle between the direction that the camera is truly pointing at, $\mathbf{d}_{LOS}$, and the direction of the comet, $\mathbf{d}_c$, both in the body fixed frame. Thus, the cosine of the pointing error is

$$\cos(\theta_{PE}) = \mathbf{d}_{LOS}^T \mathbf{d}_c. \tag{5.3}$$

The latter direction can be obtained from the position of the spacecraft represented in the inertial frame, using the attitude information that can be retrieved from the attitude kinematics and the translation information, which entails

$$\begin{bmatrix} 0 \\ \mathbf{d}_c \end{bmatrix} = \mathbf{q}^* \otimes \begin{bmatrix} 0 \\ {}^I\mathbf{r}_c \end{bmatrix} \otimes \mathbf{q}, \tag{5.4}$$

where ${}^I\mathbf{r}_c$ is a unitary vector [35].

5.1.3 Control engineering approach

As stated before, the work of this thesis intends to focus on the system identification to aid the controller design of a Space mission. The main objective of the Comet Interceptor is to keep the target within the cameras Field of View (FoV), in order to obtain high quality images of the comet. Thus, the GNC and AOCS are to be designed with this objective in mind, which requires an intense process of design, validation and verification. In a control engineering project, the build up to the most detailed simulator of the mission is performed in multiple steps, each requiring proof of work and guarantees of safety.

In this sense, Low Fidelity Simulators are of extreme importance and generate multiple findings that are then incorporated in High Fidelity Simulators, such as Functional Engineering Simulators. The main differences between Low and High Fidelity Simulators lie in the imposition and design of external perturbations, which are much more detailed for the more complex version and represent more accurately the environment where the body we are studying is going to be integrated and perform its mission. Additionally, High Fidelity Simulators include a comprehensive design of the interaction between different parts of the on board software for guidance, navigation and control purposes, which is critical for mission success.

The main take away from the Low Fidelity Simulator is its approach to separate dynamical properties of the system, which result in first solutions for the problems encountered that can then be tuned and adapted to the full overview of the system. Thus, the work presented here is an approach to the first step of the design process, where the nonlinear system is represented, without external perturbations, to allow for a deeper knowledge of the system behavior to the control inputs that may be given to it.

Figure 5.1 represents the general structure that we work with and the data flow between the different blocks. Generally, a mission simulator is composed by a representation of the real world, which can contain only a representation of the Dynamics and Kinematics Environment (DKE), with or without external

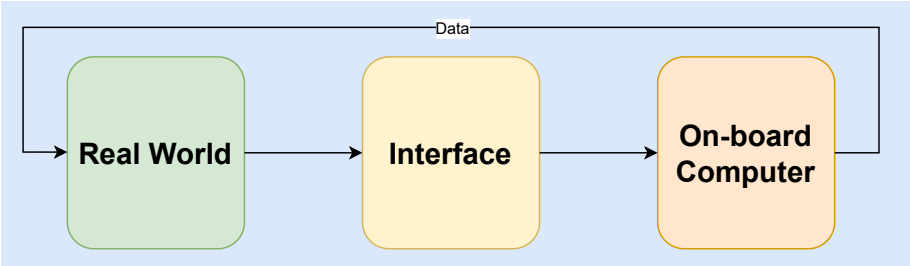


Figure 5.1: Low Fidelity Simulator structure.

disturbances, or go even further and address the actuators and sensors that are going to be present in the physical system, simulating its limitations (e.g. the reaction wheel saturation) and the perturbations they may introduce (e.g. sensor noise and imperfections).

Figure 5.2 represents the usual structure of the real world block. For the purpose of this work, in the Low Fidelity Simulator, the actuators and sensors are not going to be implemented, with the data available for the on board computer being available from a perfect DKE.

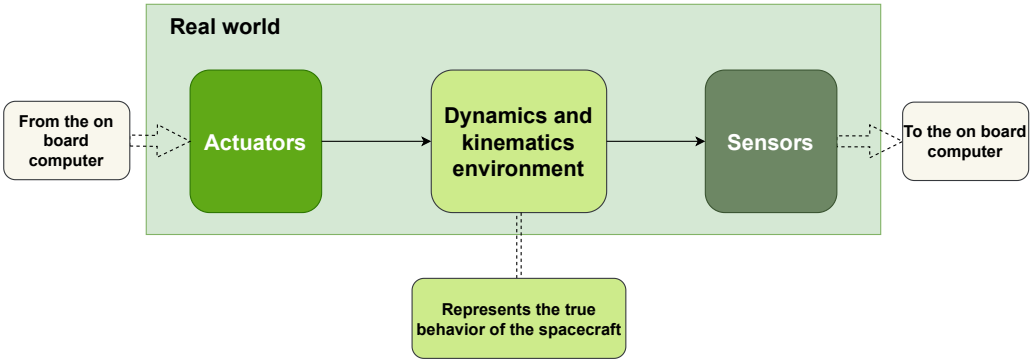


Figure 5.2: Details of the real world implementation.

The on board computer has the implementation of all the computation that is needed for the guidance, navigation and control of the mission. In here, the estimation of the true state of the spacecraft is computed, along with the reference path it might follow and the commands it should receive to achieve its objectives, with the links between each part being explicitly described in the diagram 5.3. We assume, for this trial, that the navigation and guidance are perfect. The output of this block results from the coordination of all parts and will be then input into the real world component of the Low Fidelity Simulator as it would occur in the true spacecraft.

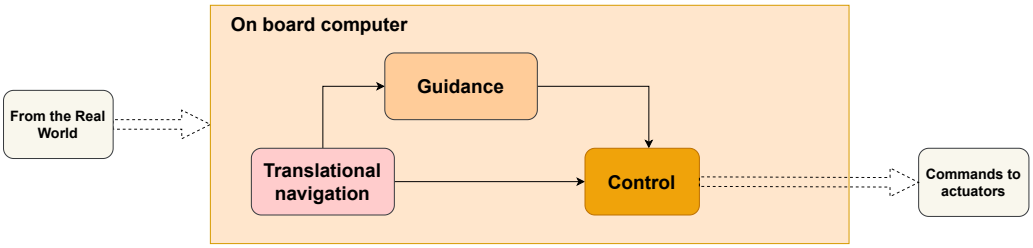


Figure 5.3: Details of the software implementation.

To be performed correctly, the connection between all systems must be seamless and as close to

reality as possible. This simulator is the basis of the work from here on, allowing for the data collection and validation of the results.

5.2 Design approach to the nonlinear attitude control problem

As the simulator is working in an approximate continuous time, we will be considering a continuous linearization and then performing the discretization of the achieved system to have a basis to be compared with the expected COSMIC result. The controller present in the data collection phase is based on a LTI LQR developed for continuous-time.

As stated previously, we expect the spacecraft to perform a well established trajectory for the fly-by mission, here considered in a linear trajectory that passes right in front of the target, perpendicular to it. The reference is computed in the simulator and the control objective is to achieve a zero error between the reference and the true attitude of the spacecraft. As such, we intend to control the attitude, the angular velocity, i.e. the derivative of the attitude, and the attitude integral, which is a proportional, derivative and integral setting, allowing for a more precise control and a fair comparison to the new controller we wish to synthesize, through the dynamic programming approach presented in Chapter 4.

5.2.1 Linearization

The nonlinear attitude dynamics and kinematic system can be linearized around a nominal trajectory, when assuming that the states can be represented as its nominal value, here the reference, disturbed by an error, i.e. the variable that we intend to control and drive to zero. Thus, we can take into account that the angular velocity, the dynamics component we intend to control, can be represented as

$$\boldsymbol{\omega}(t) = \bar{\boldsymbol{\omega}}(t) + \boldsymbol{\delta\omega}(t). \tag{5.5}$$

The nominal angular velocity is obtained by imposing a nominal input profile $\mathbf{u}(t)$, such that the total input torque is

$$\mathbf{u}(t) = \bar{\mathbf{u}}(t) + \boldsymbol{\delta}\mathbf{u}(t). \tag{5.6}$$

If we disregard the terms of second and higher order, we can compute the linearized dynamics equations as

$$\mathbf{J}\dot{\boldsymbol{\delta\omega}}(t) = \underbrace{([\mathbf{J}\bar{\boldsymbol{\omega}}(t)]_\times - [\bar{\boldsymbol{\omega}}(t)]_\times \mathbf{J})}_{\mathbf{A}_{\boldsymbol{\omega}}(t)} \boldsymbol{\delta\omega}(t) - \boldsymbol{\delta}\mathbf{u}(t). \tag{5.7}$$

Due to its properties, writing the true quaternion as the sum of the nominal one and an error quaternion is not feasible as this does not hold true. However, following the same reasoning and using the properties of unit quaternions, the quaternion error can be written as the composition of two rotations, given that

$$\bar{\mathbf{q}}(t) = \boldsymbol{\delta}\mathbf{q}(t) \otimes \mathbf{q}(t) \tag{5.8}$$

$\boldsymbol{\delta}\mathbf{q}$ can be seen as enconding a small rotation, and hence we can write

$$\begin{bmatrix} 0 \\ \boldsymbol{\delta\theta} \end{bmatrix} = 2\boldsymbol{\delta}\mathbf{q}, \tag{5.9}$$

where $\boldsymbol{\delta\theta}$ represents the angular displacement, which entails that we can calculate it from the rotational kinematics. From [34], we know that the linearized dynamics are

$$\dot{\boldsymbol{\delta\theta}}(t) = -[\bar{\boldsymbol{\omega}}(t)]_\times \boldsymbol{\delta\theta}(t) + \boldsymbol{\delta\omega}(t). \tag{5.10}$$

Furthermore, we will add an integral term to allow for a more robust control, helping the system reject low frequency disturbances and mitigate the steady state error, where we integrate the angular variation over time.

Hence, representing the integral of the attitude error as $\int \boldsymbol{\delta\theta} = \boldsymbol{\delta\iota}$, we define the system used for the controller design as

$$\begin{cases} \dot{\boldsymbol{\delta\theta}}(t) = -[\bar{\boldsymbol{\omega}}(t)]_\times \boldsymbol{\delta\theta}(t) + \boldsymbol{\delta\omega}(t) \\ \dot{\boldsymbol{\delta\omega}}(t) = \mathbf{J}^{-1}\mathbf{A}_{\boldsymbol{\omega}}(t)\boldsymbol{\delta\omega}(t) - \mathbf{J}^{-1}\boldsymbol{\delta}\mathbf{u}(t) \\ \dot{\boldsymbol{\delta\iota}}(t) = \boldsymbol{\delta\theta}(t), \end{cases} \tag{5.11}$$

which can be simplified to the linear time-variant system

$$\begin{bmatrix} \dot{\boldsymbol{\delta\theta}}(t) \\ \dot{\boldsymbol{\delta\omega}}(t) \\ \dot{\boldsymbol{\delta\iota}}(t) \end{bmatrix} = \mathbf{A}(t) \begin{bmatrix} \boldsymbol{\delta\theta}(t) \\ \boldsymbol{\delta\omega}(t) \\ \boldsymbol{\delta\iota}(t) \end{bmatrix} + \mathbf{B}(t)\boldsymbol{\delta}\mathbf{u}(t). \tag{5.12}$$

Matrices $\mathbf{A}$ and $\mathbf{B}$ are then given by

$$\mathbf{A}(t) = \begin{bmatrix} -[\bar{\boldsymbol{\omega}}(t)]_\times & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}^{-1}\mathbf{A}_{\boldsymbol{\omega}}(t) & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{B}(t) = \begin{bmatrix} \mathbf{0} \\ -\mathbf{J}^{-1} \\ \mathbf{0} \end{bmatrix}$$

with $\mathbf{A} \in \mathbb{R}^{(3+3+3)\times(3+3+3)}$ and $\mathbf{B} \in \mathbb{R}^{(3+3+3)\times 3}$ and the torque input considered in the three directions of the body frame and not divided by the reaction wheels. The nominal values of the states, the reference, are obtained by an analytical solution that will be presented in a section further ahead.

5.2.2 Discretization

In order to fairly compare the system we intend to estimate with the previously described one, we need to discretize (5.11). From the linear systems theory, we know that the continuous time-variant system can be discretized as

$$\boldsymbol{\delta}\mathbf{x}(t_{k+1}) = \boldsymbol{\Phi}_{k+1|k}\boldsymbol{\delta}\mathbf{x}(t_k) + \mathbf{G}_{k+1|k}\boldsymbol{\delta}\mathbf{u}(t_k), \tag{5.13}$$

where the state transition matrix $\mathbf{\Phi}_{k+1|k}$, and the input propagation matrix, $\mathbf{G}_{k+1|k}$ solve the dynamical system, allowing the determination of the state $\boldsymbol{\delta}\mathbf{x}$ at time step $k+1$ as a linear combination of the state and the input from the previous time step k. We do not compare directly these results with the ones from COSMIC, as the nonlinearities are not necessarily dealt with in the same way, but compare instead the ability to predict the state and compare both performances with previously collected data from a test trajectory.

With respect to the solution of $\boldsymbol{\delta\theta}(t)$, the variation of constants method [36] provides

$$\boldsymbol{\delta\theta}(t_{k+1}) = \mathbf{\Phi}_{\boldsymbol{\theta}}(t_{k+1}, t_k)\boldsymbol{\delta\theta}(t_k) + \int_{t_k}^{t_{k+1}} \boldsymbol{\delta\omega}(t_k), \tag{5.14}$$

where $\mathbf{\Phi}_{\boldsymbol{\theta},k+1|k} := \mathbf{\Phi}_{\boldsymbol{\theta}}(t_{k+1}, t_k)$.

For the angular velocity, we can also write, assuming that the input is constant within each $[t_k, t_{k+1}]$ interval,

$$\boldsymbol{\delta\omega}(t_{k+1}) = \mathbf{\Phi}_{\boldsymbol{\omega}}(t_{k+1}, t_k)\boldsymbol{\delta\omega}(t_k) + \underbrace{\int_{t_k}^{t_{k+1}} \mathbf{\Phi}_{\boldsymbol{\omega}}(t_{k+1}, \tau)\mathbf{B}_{\boldsymbol{\omega}}\mathrm{d}\tau}_{\mathbf{G}_{\boldsymbol{\omega}}(t_{k+1}, t_k)} \boldsymbol{\delta}\mathbf{u}(t_k), \tag{5.15}$$

where $\mathbf{\Phi}_{\boldsymbol{\omega},k+1|k} := \mathbf{\Phi}_{\boldsymbol{\omega}}(t_{k+1}, t_k)$.

Lastly, for the integral term discretization, the statement is different, directly from the integral definition, such that

$$\boldsymbol{\delta\iota}(t) = \boldsymbol{\delta\iota}(t_k) + \int_{t_k}^{t} \boldsymbol{\delta\theta}(t_k). \tag{5.16}$$

Noting that the input is constant within each interval, we can write, for $t_k \leq t < t_{k+1}$,

$$\begin{bmatrix} \dot{\boldsymbol{\delta\theta}}(t) \\ \dot{\boldsymbol{\delta\omega}}(t) \\ \dot{\boldsymbol{\delta\iota}}(t) \\ \dot{\boldsymbol{\delta}}\mathbf{u}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}(t) & \mathbf{B}(t) \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\bar{\mathbf{A}}} \begin{bmatrix} \boldsymbol{\delta\theta}(t) \\ \boldsymbol{\delta\omega}(t) \\ \boldsymbol{\delta\iota}(t) \\ \boldsymbol{\delta}\mathbf{u}(t) \end{bmatrix}. \tag{5.17}$$

Considering that the solution for a linear system is given by the state transition matrix $\bar{\mathbf{\Phi}}(t, t_k)$ that respects the ordinary differential equation

$$\dot{\bar{\mathbf{\Phi}}}(t, t_k) = \bar{\mathbf{A}}(t)\bar{\mathbf{\Phi}}(t, t_k), \tag{5.18}$$

with $\bar{\mathbf{\Phi}}(t_k, t_k) = \mathbf{I}$, we can extract $\mathbf{\Phi}_{k+1|k}$ and $\mathbf{G}_{k+1|k}$ by integrating (5.18) numerically using the nominal angular velocity reference. Then,

$$\bar{\mathbf{\Phi}}(t_{k+1}, t_k) = \begin{bmatrix} \mathbf{\Phi}_{k+1|k} & \mathbf{G}_{k+1|k} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_{\boldsymbol{\theta},k+1|k} & \mathbf{\Phi}_{\boldsymbol{\theta\omega},k+1|k} & \mathbf{0} & \mathbf{G}_{\theta,k+1|k} \\ \mathbf{0} & \mathbf{\Phi}_{\boldsymbol{\omega},k+1|k} & \mathbf{0} & \mathbf{G}_{\omega,k+1|k} \\ \mathbf{\Phi}_{\boldsymbol{\iota\theta},k+1|k} & \mathbf{\Phi}_{\boldsymbol{\iota\omega},k+1|k} & \mathbf{I} & \mathbf{G}_{\iota,k+1|k} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{5.19}$$

5.2.3 Reference trajectory computation

Although not the focus of this work, the reference trajectory is needed for the system implementation in the simulator. Thus, we present the study of the desired attitude through the rotation matrix that defines it, being represented by

$$\mathbf{R}_{ref} = \begin{bmatrix} \frac{{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|} & \frac{{}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{h}(t)\|} & -\frac{{}^{I}\mathbf{r}(t)}{\|{}^{I}\mathbf{r}(t)\|} \end{bmatrix} \tag{5.20}$$

where ${}^{I}\mathbf{r}(t) \in \mathbb{R}^3$ is the position of the spacecraft with respect to the inertial frame $\{I\}$ centered on the comet and ${}^{I}\mathbf{h}(t) \in \mathbb{R}^3$ is the angular momentum of the orbit, defined as ${}^{I}\mathbf{h}(t) = {}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{v}(t)$ with ${}^{I}\mathbf{v}(t) \in \mathbb{R}^3$ being the linear velocity of the spacecraft with respect to the comet. With this definition, the z-axis of the spacecraft body frame will point towards the comet, the y-axis will be parallel to the angular momentum pointing downwards, and the x-axis will complete the right-handed frame.

From [34], we know that the following holds true, in particular for the reference attitude,

$$\dot{\mathbf{R}}_{ref}(t) = \mathbf{R}_{ref}(t) \left[\boldsymbol{\omega}_{ref}(t)\right]_{\times} \quad \Leftrightarrow \quad \mathbf{R}_{ref}^{T}(t)\dot{\mathbf{R}}_{ref}(t) = \left[\boldsymbol{\omega}_{ref}(t)\right]_{\times}. \tag{5.21}$$

As such, let us consider the derivative of each column of the rotation matrix.

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|}\right) = \frac{{}^{I}\mathbf{v}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|} - \frac{{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|^3}\left({}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\right)^{T}\left({}^{I}\mathbf{v}(t) \times {}^{I}\mathbf{h}(t)\right) \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{{}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{h}(t)\|}\right) = \mathbf{0} \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{{}^{I}\mathbf{r}(t)}{\|{}^{I}\mathbf{r}(t)\|}\right) = \frac{{}^{I}\mathbf{v}(t)}{\|{}^{I}\mathbf{r}(t)\|} - \frac{{}^{I}\mathbf{r}(t)}{\|{}^{I}\mathbf{r}(t)\|^3}{}^{I}\mathbf{r}(t)^{T}{}^{I}\mathbf{v}(t) \end{cases}. \tag{5.22}$$

The angular momentum of the orbit is considered to be constant, either through the assumption of a Keplerian orbit or the straight-line fly-by to the comet.

Thus, using (5.20) and (5.22), we can solve (5.21) can lead to

$$\left[\boldsymbol{\omega}_{ref}(t)\right]_{\times} = \begin{bmatrix} 0 & 0 & -\left(\frac{{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|}\right)^{T}\frac{{}^{I}\mathbf{v}(t)}{\|{}^{I}\mathbf{r}(t)\|} \\ 0 & 0 & 0 \\ -\frac{{}^{I}\mathbf{r}^{T}(t)}{\|{}^{I}\mathbf{r}(t)\|}\frac{{}^{I}\mathbf{v}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|} & 0 & 0 \end{bmatrix} \tag{5.23}$$

where the fact that both ${}^{I}\mathbf{r}(t)$ and ${}^{I}\mathbf{v}(t)$ are perpendicular to the angular momentum vector was used to remove identically zero elements of the matrix. This shows that the spacecraft only needs to rotate along its y-axis to maintain pointing to the comet, as, from (5.23), it is possible to write

$$\boldsymbol{\omega}_{ref}(t) = \begin{bmatrix} 0 \\ -\left(\frac{{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|}\right)^{T}\frac{{}^{I}\mathbf{v}(t)}{\|{}^{I}\mathbf{r}(t)\|} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{{}^{I}\mathbf{r}^{T}(t)}{\|{}^{I}\mathbf{r}(t)\|}\frac{{}^{I}\mathbf{v}(t) \times {}^{I}\mathbf{h}(t)}{\|{}^{I}\mathbf{r}(t) \times {}^{I}\mathbf{h}(t)\|} \\ 0 \end{bmatrix}. \tag{5.24}$$

Having the reference angular velocity, the reference attitude quaternion is obtained through the conversion of (5.20) to quaternions, with the reference quaternion as $\mathbf{q}_{ref} = \begin{bmatrix} q_{w_{ref}} & \mathbf{q}_{\mathbf{v_{ref}}}^{\mathbf{T}} \end{bmatrix}^T$ regarding each element of $\mathbf{R}_{ref}$ as R_{ij}, with $i, j = \{x, y, x\}$, and from [37],

$$\begin{cases} q_{w_{ref}} = \sqrt{1 + \operatorname{Tr} \mathbf{R}_{ref}} \\ q_{x_{ref}} = \sqrt{\frac{R_{xx}}{2} + \frac{1 - \operatorname{Tr} \mathbf{R}_{ref}}{4}} \\ q_{y_{ref}} = \sqrt{\frac{R_{yy}}{2} + \frac{1 - \operatorname{Tr} \mathbf{R}_{ref}}{4}} \\ q_{z_{ref}} = \sqrt{\frac{R_{zz}}{2} + \frac{1 - \operatorname{Tr} \mathbf{R}_{ref}}{4}} \end{cases} \tag{5.25}$$

Advancing the analysis to compute the reference torque, we can start by simplifying (5.24), considering that the relative velocity can be decomposed in a tangential and radial component as shown in Figure 5.4. In that case, noting that $\left\| {}^I\mathbf{r}(t) \times {}^I\mathbf{h}(t) \right\| = \left\| {}^I\mathbf{r}(t) \right\| \left\| {}^I\mathbf{h}(t) \right\|$ as they are perpendicular, the angular speed is

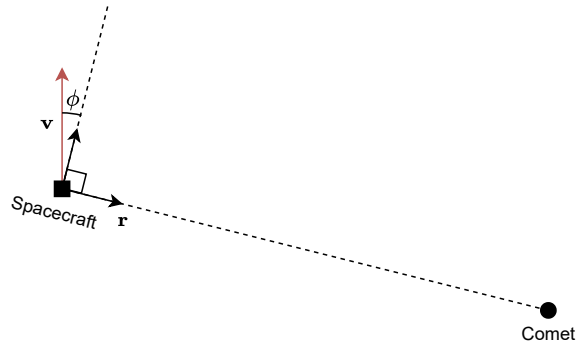


Figure 5.4: Components of the relative velocity.

$$\begin{aligned} \omega_{y_{ref}}(t) &= -\left(\frac{{}^I\mathbf{r}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} \times \frac{{}^I\mathbf{h}(t)}{\left\| {}^I\mathbf{h}(t) \right\|} \right)^T \left(\frac{v_{radial}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} \frac{{}^I\mathbf{r}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} - \frac{v_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} \frac{{}^I\mathbf{r}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} \times \frac{{}^I\mathbf{h}(t)}{\left\| {}^I\mathbf{h}(t) \right\|} \right) \\ &= \frac{v_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} = \frac{\left\| {}^I\mathbf{v}(t) \right\|}{\left\| {}^I\mathbf{r}(t) \right\|} \cos\phi(t) = \frac{\left\| {}^I\mathbf{h}(t) \right\|}{\left\| {}^I\mathbf{r}(t) \right\|^2} \end{aligned} \tag{5.26}$$

where $\phi(t)$ is the flight path angle, as shown in Figure 5.4. From close inspection of the results of this expression, it is clear that this is simply the result of well-known relation for rotational motion, ${}^I\mathbf{v}(t) = {}^I\boldsymbol{\omega}(t) \times {}^I\mathbf{r}(t)$, thus validating the approach.

Finally, to obtain the reference angular acceleration (and the associated torques), consider the time derivative of (5.26), given by

$$\begin{aligned} \dot{\omega}_{y_{ref}}(t) &= \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{v_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} \right) = \frac{\dot{v}_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|} - \frac{v_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|^3} {}^{I}\mathbf{r}^T(t) {}^I\mathbf{v}(t) \\ &= -\frac{1}{\left\| {}^I\mathbf{r}(t) \right\|} \left({}^I\dot{\mathbf{v}}^T(t) \frac{{}^I\mathbf{r}(t) \times {}^I\mathbf{h}(t)}{\left\| {}^I\mathbf{r}(t) \times {}^I\mathbf{h}(t) \right\|} - {}^I\mathbf{v}^T(t) \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{{}^I\mathbf{r}(t) \times {}^I\mathbf{h}(t)}{\left\| {}^I\mathbf{r}(t) \times {}^I\mathbf{h}(t) \right\|} \right) \right) - \frac{v_{tangential}(t)}{\left\| {}^I\mathbf{r}(t) \right\|^2} \frac{{}^I\mathbf{r}^T(t)}{\left\| {}^I\mathbf{r}(t) \right\|} {}^I\mathbf{v}(t) \end{aligned} \tag{5.27}$$

where it can be noticed that the last parcel contains the radial velocity. Substituting the first line of (5.22)

in the above expression and again considering the decomposition of the velocity gives

$$\begin{aligned}
\dot{\omega}_{y_{ref}}(t) = & -\frac{1}{\|{}^I\mathbf{r}(t)\|}\left({}^I\dot{\mathbf{v}}^T(t)\frac{{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)}{\|{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\|} - {}^I\mathbf{v}^T(t)\frac{{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)}{\|{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\|^3}\left({}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\right)^T\left({}^I\mathbf{v}(t)\times{}^I\mathbf{h}(t)\right)\right) \\
& -\frac{v_{tangential}(t)}{\|{}^I\mathbf{r}(t)\|}\frac{v_{radial}(t)}{\|{}^I\mathbf{r}(t)\|} \\
= & -\frac{{}^I\dot{\mathbf{v}}^T(t)}{\|{}^I\mathbf{r}(t)\|}\frac{{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)}{\|{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\|} - \frac{v_{tangential}(t)}{\|{}^I\mathbf{r}(t)\|}\left(\frac{{}^I\mathbf{v}(t)}{\|{}^I\mathbf{r}(t)\|}\times\frac{{}^I\mathbf{h}(t)}{\|{}^I\mathbf{h}(t)\|}\right)^T\frac{{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)}{\|{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\|} \\
& -\frac{v_{tangential}(t)}{\|{}^I\mathbf{r}(t)\|}\frac{v_{radial}(t)}{\|{}^I\mathbf{r}(t)\|} \\
= & -\frac{1}{\|{}^I\mathbf{r}(t)\|}\underbrace{{}^I\dot{\mathbf{v}}^T(t)\frac{{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)}{\|{}^I\mathbf{r}(t)\times{}^I\mathbf{h}(t)\|}}_{\text{tangential linear acceleration}} - 2\frac{v_{tangential}(t)}{\|{}^I\mathbf{r}(t)\|}\frac{v_{radial}(t)}{\|{}^I\mathbf{r}(t)\|}
\end{aligned} \tag{5.28}$$

The value of the first parcel will depend on the actual dynamic model considered. In both the two body problem and the straight-line flyby model, the tangential acceleration is zero, and, as such, the angular acceleration is merely dependent on the distance to the central body and the tangential and radial velocities.

In order to obtain the reference input profile that generates the reference angular acceleration and velocity, it is necessary to solve the inverse dynamics problem. Generically, not considering reaction wheel momenta $\mathbf{h}(t)$, this becomes the simple algebraic problem

$$\mathbf{J}\dot{\boldsymbol{\omega}}_{ref}(t) + \boldsymbol{\omega}_{ref}(t)\times\mathbf{J}\boldsymbol{\omega}_{ref}(t) = \boldsymbol{\tau}_{ref}(t). \tag{5.29}$$

5.3 COSMIC use case and performance assessment

Similarly to what was done for the validation scenario, we had a setup for data collection and a setup for control validation. The schematic of the simulation is presented in Figure 5.5.

Having established the Low Fidelity Simulator, the data collection procedure can be optimized to account for the result of Theorem 1, by calculating the sum of the covariances of each run of the simulator, up until the minimum singular value of the sum is above a certain threshold, a fair amount above zero, to guarantee that the data is sufficient to characterize the system. For this particular case, we defined the threshold as 10 and we were able to stop the simulator after 222 runs. Thus, at a sample time of $1/5$ s and a total of 200 s, each trajectory had 1001 instants being considered, $N = 1001$.

5.3.1 Model estimation

For the purpose of this proof of concept, and considering the motivation of this work, we opt to address the problem with varying λ_k. Taking into account the previously knowledge about the system, i.e. the angular velocity over the y axis that is desired, we establish that the work can be performed considering two different values for λ_k and three zones of the trajectory where this value will be maintained constant. Figure 5.6 makes explicit this decision. To better adjust these values to the systems needs,

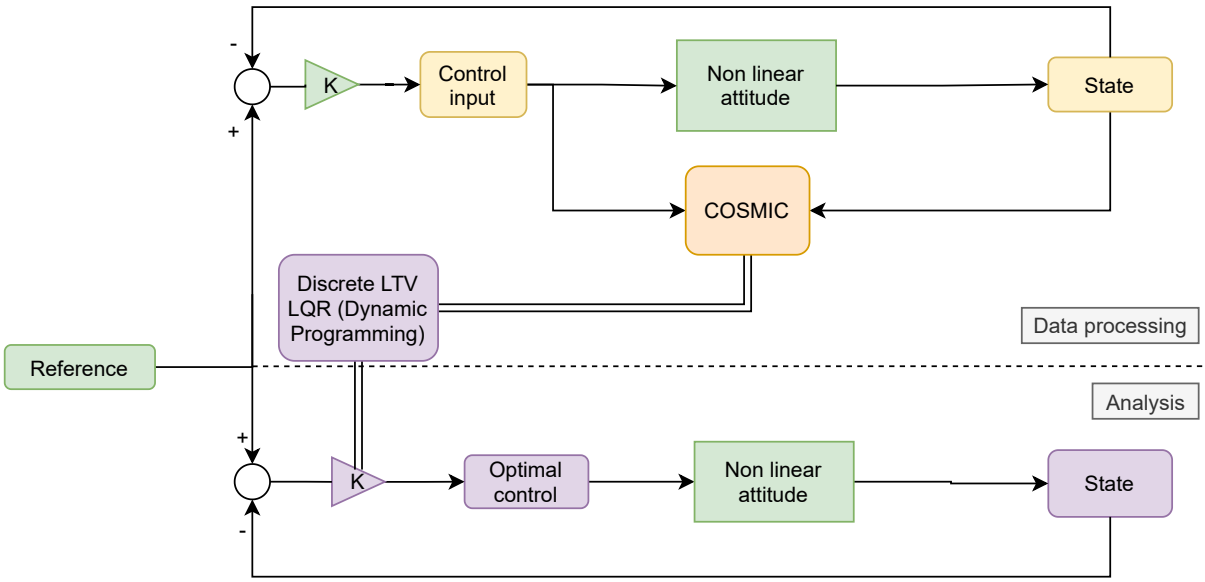


Figure 5.5: Data collection and controller validation in the Low Fidelity Simulator.

we perform a parametric study.

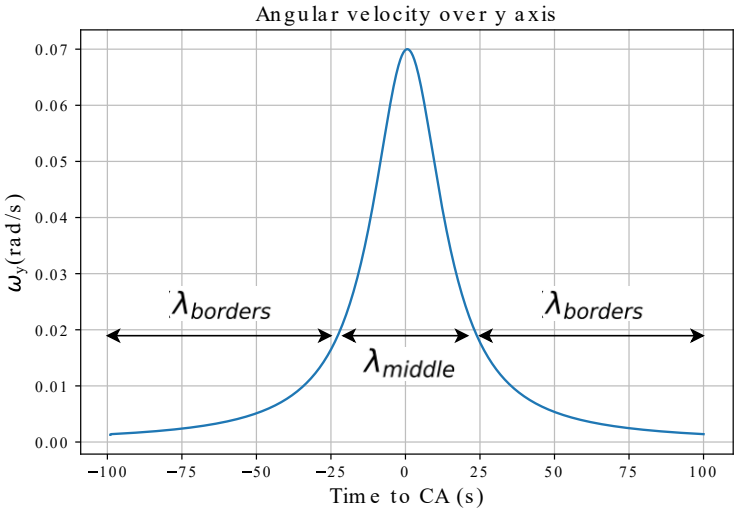
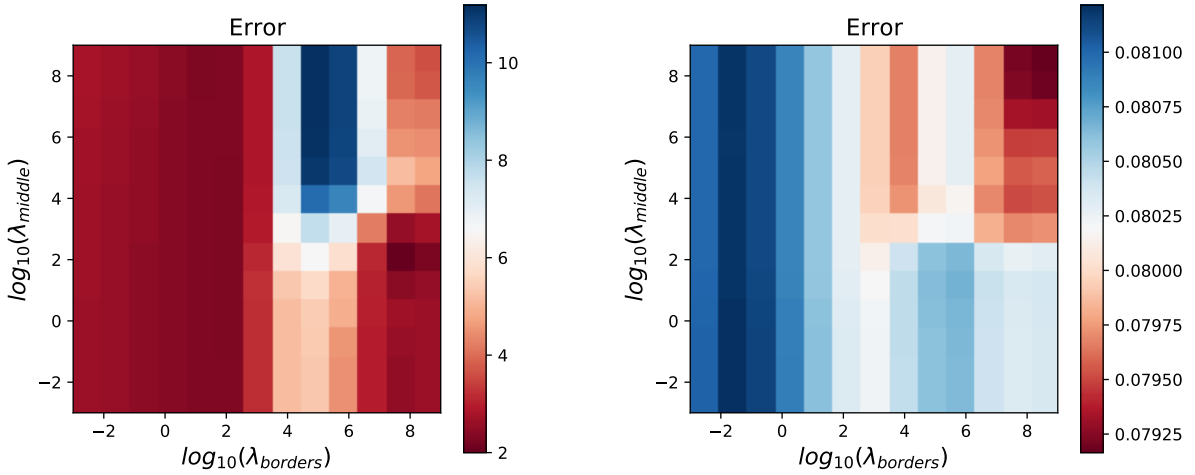


Figure 5.6: Different λ_k for different trajectory zones.

With the data collected, we input it in COSMIC and we study the model estimated and run the algorithm multiple times, considering each iteration the model achieved and comparing the estimated state, using previously collected control inputs, with the true trajectory. The state used to find $x(k+1)$ was the state estimated in k or the true state in k, which allowed us to identify two different performance metrics, the estimation power that amplifies the linearization error and the estimation power in each instant, respectively.

Analyzing Figure 5.7 (a), we can infer that the lowest estimation error from the previous estimated state occurs for $\lambda_{middle} = 10^2$ and $\lambda_{borders} = 10^8$. For the instantaneous estimation error, the error



(a) Mean estimation error, from estimated state.

(b) Mean estimation error, from true state.

Figure 5.7: Parametric results for the variation of λ_k along the trajectory.

decreases as the λ_k increases. We choose the best pair to be the one with the lowest mean estimation error, from estimated state, as it also appears to have a low error when calculated with the true state.

This result is in line with the analysis of the system and its expected behavior. For the zones further away from the closest approach point, the system dynamics is much slower and the variations between instants are less disruptive, which is in line with a higher value of λ_k, that imposes a narrower difference between the optimal variable variation in consecutive instants. In the middle of the trajectory, the spacecraft is at its closest point to the target and its attitude needs to change much faster to maintain a pointing error small enough, thus the system changes much faster and the difference between to instants needs to be larger, which is allowed by the smaller value of λ_k.

Additionally, to verify if the estimation is an upgrade from the linearization, and thus prove that the COSMIC algorithm can be an useful tool for the dynamical systems analysis, we plot the evolution of the estimated states for each case, when calculated with the true state at k, and also the true evolution of the state. The comparison is available in 5.8 and it the estimated state proves to be an upgrade from the linearization, although very close to this theoretical approach. The main difference can be observed for the evolution of $\delta\theta_z$, where the linearized system is unable to follow as expected.

This result provides a good basis to follow up our analysis, as it demonstrates the advantage of using the estimated model in detriment of the linearization output and establishes COSMIC as a step forward to allow for faster deployment of the first phase analysis of a dynamical system.

Before proceeding to the next phase of the implementation, a reachability and controllability analysis of the estimated model is performed. Firstly, from [26], we define the reachability Gramian as

$$\mathbf{W}_R(k_0, k_1) := \sum_{\tau=k_0}^{k_1-1} \mathbf{\Phi}(k_1, \tau+1)\mathbf{B}(\tau)\mathbf{B}^T(\tau)\mathbf{\Phi}^T(k_1, \tau+1) \tag{5.30}$$

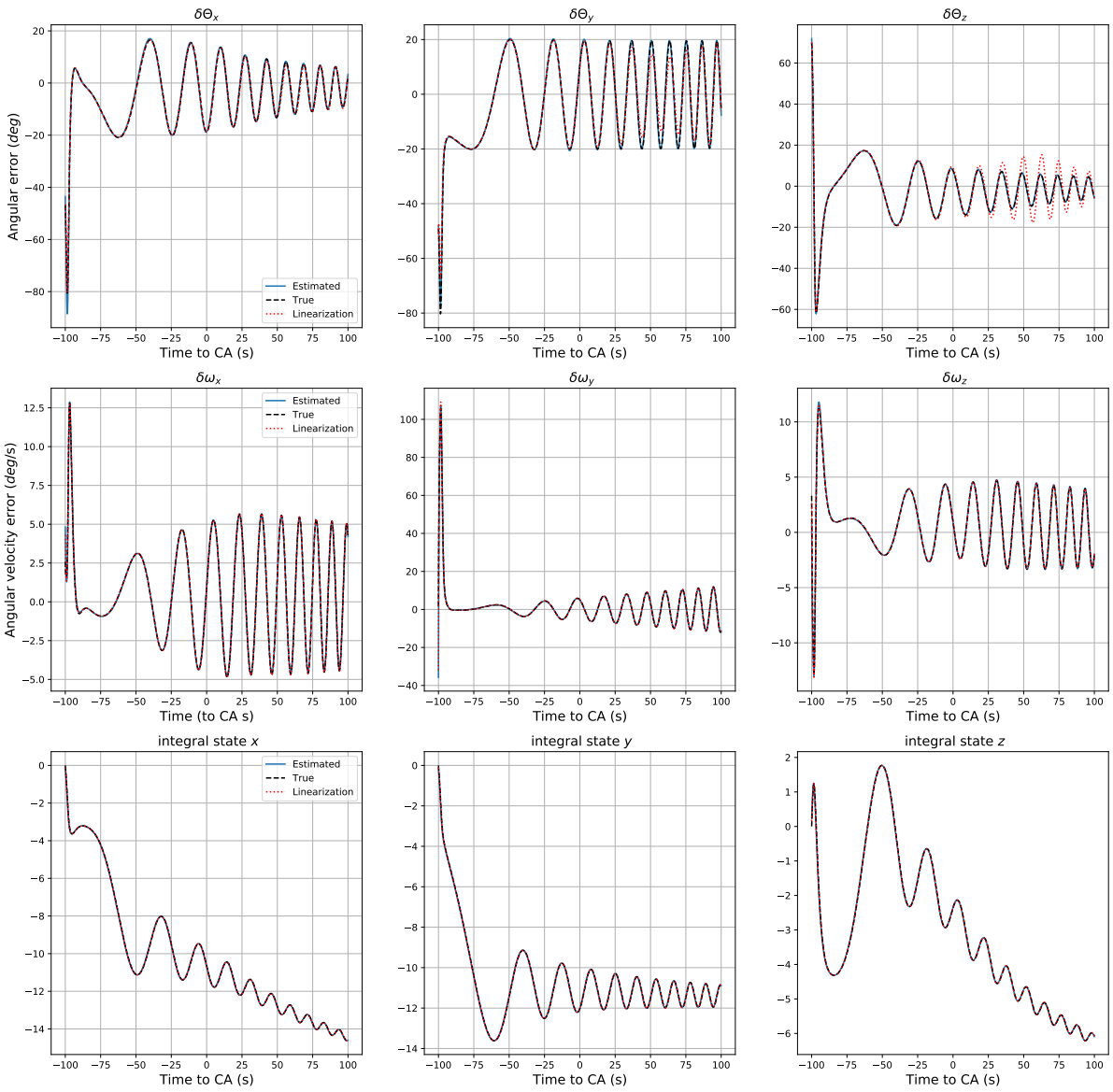


Figure 5.8: Comparison between the performance of the estimated model and the linearization.

and the controllability counterpart as

$$\mathbf{W}_C(k_0, k_1) := \sum_{\tau=k_0}^{k_1-1} \mathbf{\Phi}(k_0, \tau+1)\mathbf{B}(\tau)\mathbf{B}^T(\tau)\mathbf{\Phi}^T(k_0, \tau+1). \tag{5.31}$$

The backwards transitions are defined as

$$\mathbf{\Phi}(k_0, \tau+1) = \mathbf{\Phi}(\tau+1, k_0)^{-1} = \{\mathbf{A}(\tau-1)\mathbf{A}(\tau-2)...\mathbf{A}(k_0)\}$$

As such, we calculated the minimum singular values of the controllability and reachability Gramians, to verify if the learned dynamics were in fact retaining these properties necessary for the optimal control design or if the linearization performed by the COSMIC application was not good enough and was

unable to characterize certain aspects of the model. The study performed was, however, positive, having achieved all positive singular values for the Gramians, thus entailing that we can control the estimated system and proceed with our analysis. If the outcome was not positive, it would be constructive to take a step back and address the value of λ_k or even the problem formulation as linear time-variant system.

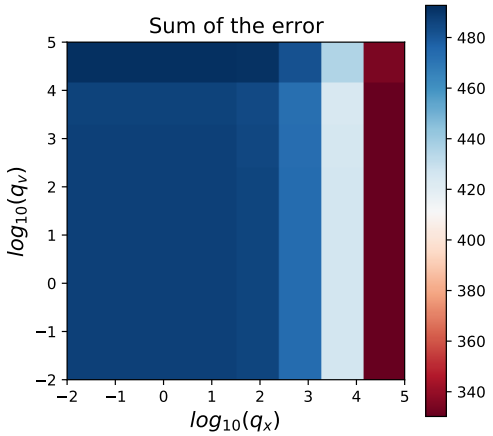
5.3.2 Controller design

Moving forward to the second phase of the COSMIC applicability to Space missions, we apply the controller design framework described in Chapter 4 to the estimated model. Thus, we start by describing the cost function for the LQR for the state and input system we are currently addressing. Once again, the design matrices are considered constant along the trajectory and $\mathbf{H} = \mathbf{Q}$. First, we define $\mathbf{Q_x} = q_x\mathbf{I}$, $\mathbf{Q_v} = q_v\mathbf{I}$, $\mathbf{Q_i} = q_i\mathbf{I}$ and $\mathbf{R} = r\mathbf{I}$, yielding

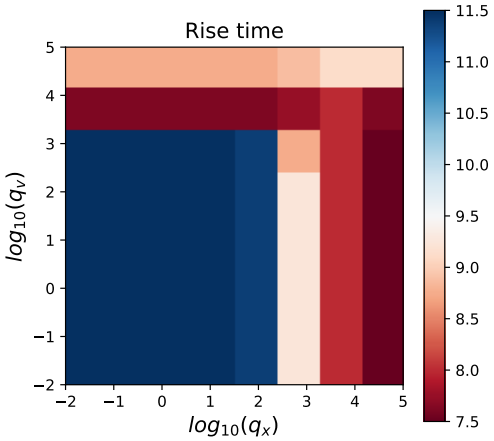
$$\begin{aligned} J &= \frac{1}{2}\sum_{k=0}^{N-1} \mathbf{x}_k^T \begin{bmatrix} \mathbf{Q_x} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q_v} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q_i} \end{bmatrix} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k \\ &= \frac{1}{2}\sum_{k=0}^{N-1} q_x \boldsymbol{\delta\theta}_k^T \boldsymbol{\delta\theta} + q_v \boldsymbol{\delta\omega}_k^T \mathbf{x}_k + q_i \boldsymbol{\delta\theta_i}_k^T \mathbf{I} \boldsymbol{\delta\theta_i}_k + r\mathbf{u}_k^T \mathbf{u}_k \\ &= \frac{1}{2}\sum_{k=0}^{N-1} q_i \left(\frac{q_x}{q_i} \boldsymbol{\delta\theta}_k^T \boldsymbol{\delta\theta} + \frac{q_v}{q_i} \boldsymbol{\delta\omega}_k^T \mathbf{x}_k + \boldsymbol{\delta\theta_i}_k^T \boldsymbol{\delta\theta_i} + \frac{r}{q_i} \mathbf{u}_k^T \mathbf{u}_k \right). \end{aligned} \tag{5.32}$$

Similarly to the spring-mass-damper scenario, we can single out a design parameter from which the tuning will be independent. However, there are now three tunable parameters. To start our analysis, we define q_i and settle it as $q_i = 10^2$. Regarding the physical limitations of our system, namely the maximum commanded torque that the actuators can react to, we additionally study a range of values for r that are within the acceptable range. For the purpose of this study, we admit that the spacecraft can act accordingly to no more than 2 $N.m$ commanded torque for each frame axis. Thus, after evaluating the possibilities, the chosen value for r was $r = 10$. From here on, we can proceed as implemented for the validation model.

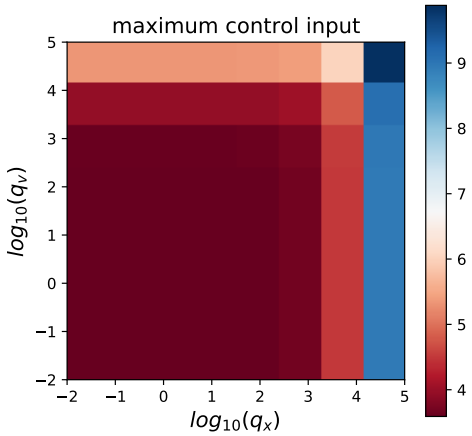
In these conditions, we define the most important metrics as the total sum of the error norm for the whole trajectory and the percentage of overshoot in the first 50 seconds of the approach, that we wish to minimize. The rise time was not taken into account. Moreover, as for the choice of r, the most important metric was the maximum control input, that should not exceed the threshold acceptable for the physical system, risking the saturation of the reaction wheels and the loss of the mission objective. The additional metric of the sum of the control energy solidifies the choice that minimizes the control input, which we also want to reduce.



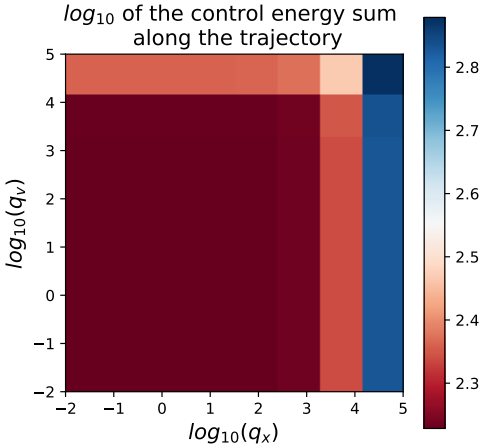
(a) Sum of the error.



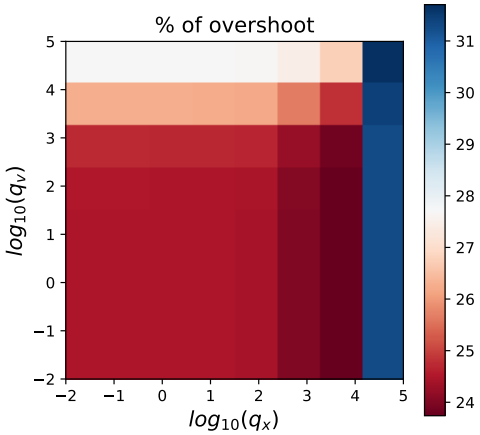
(b) Rise time.



(c) Maximum control input.



(d) Sum of the control energy.



(e) Overshoot.

Figure 5.9: Attitude controller evaluation metrics for different q_v and q_x.

Thus, having analyzed the results from Figure 5.9, the resulting controller was designed with the previously stated design parameters and $q_x = 10^4$ and $q_v = 10^5$. The resulting system is fairly slow and we could tune it to become faster, however that would come with a control input cost that, for the specifics of the Comet Interceptor mission, we are willing to accept. The time to stabilize the system to a reference is around 50 s.

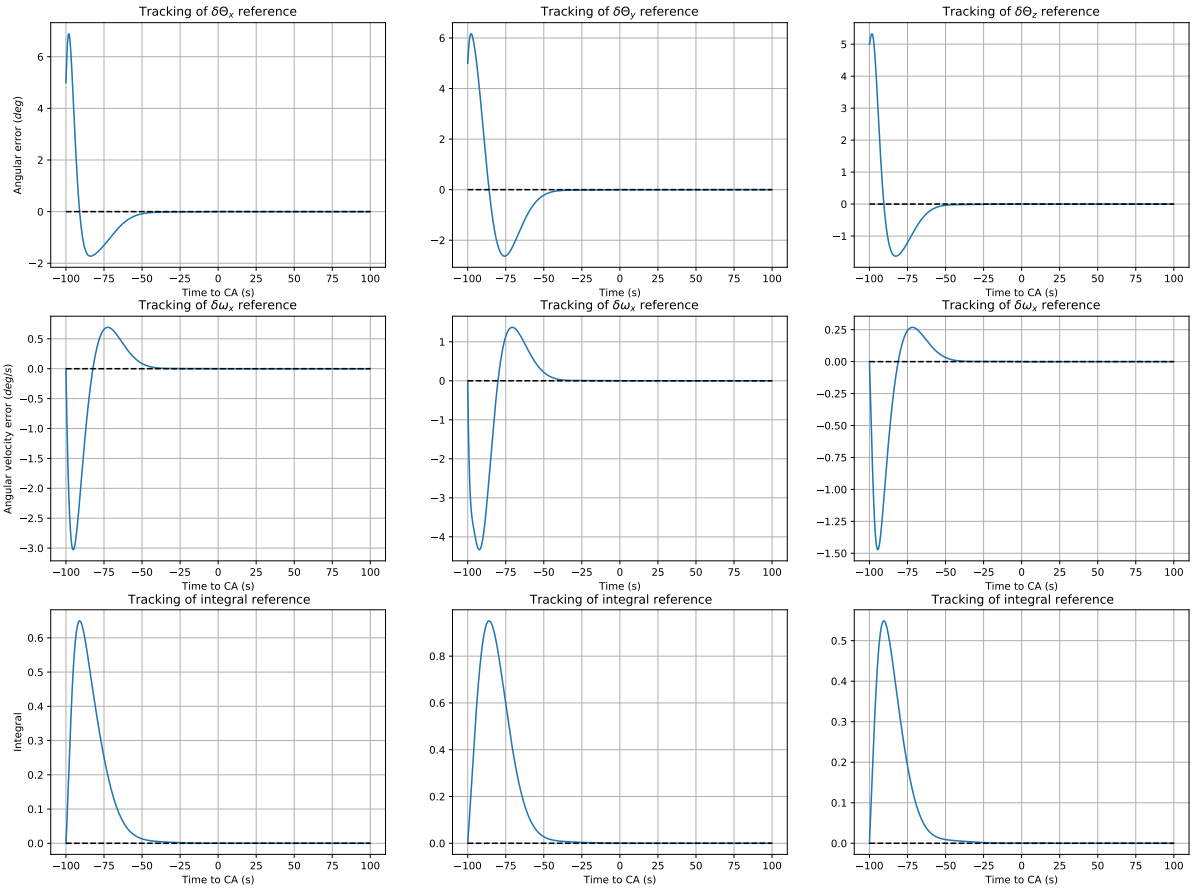
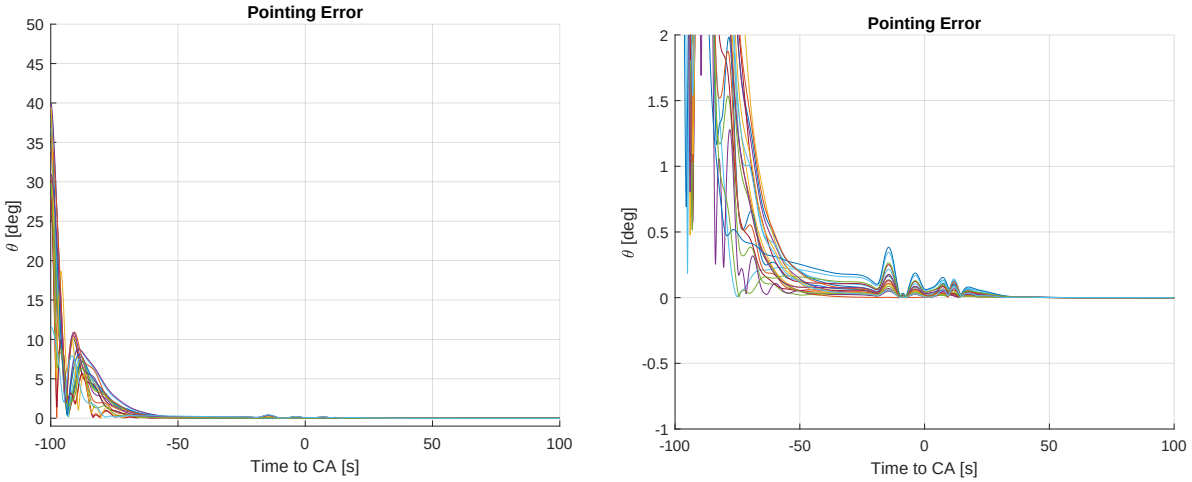


Figure 5.10: Controller design with the estimated model.

5.3.3 Low fidelity simulator

With the new controller, we are now able to input the optimal gain calculated for each distinct instant into the Low Fidelity Simulator, instead of the constant gain that had been used to collect the data. To do so, we once again use a framework corresponding to the analysis phase represented in Figure 5.5. To achieve a representative variety of results, we modify the initial attitude between each run, while maintaining the initial angular velocity null. We wish to achieve a pointing error, in relation to the optimal trajectory given by the reference, as close to zero as possible.

Figure 5.11(a) shows the results from varying initial conditions and it is clear that the controller from the linear system identification performed by COSMIC is able to control the nonlinear system and is a good option for the first approach to the GNC framework of the mission. Additionally, in Figure 5.11(b)



(a) Pointing error from different initial conditions.

(b) Pointing error in the closest approach.

Figure 5.11: Pointing error resulting from the designed controller application.

we find it interesting to observe pointing error for the faster dynamics, where there are clearly more oscillations but the error stays well below the maximum allowed threshold for the spacecraft to be able to perform its mission.

The drawbacks from this implementation come from the lack of disturbances, present in the real world environment, such as the gravity gradient induced torque, the solar radiation pressure effects, the comet coma interference, mainly in the closest approach phase, which can also cause random impacts of large particles, and the solar panel flexible modes perturbations.

The next step from this approach is then to collect data from a High Fidelity Simulator, such as the Functional Engineering Simulator, and perform the previously described procedure to collect data, design a controller and validate its use in the real mission software.

Chapter 6

Conclusions

The main contribution of this work was the development of COSMIC, a closed-form system identification algorithm from data for linear time-variant systems, formulating the identification problem as a regularized least squares, with a regularization term that imposes a constrained variation between the solution consecutive instances. This approach is a result of the collaboration between control systems theory and theory guided ML.

We started assessing the uniqueness of the solution and derived a necessary and sufficient condition for the data collection in order for the regularized least squares to output a valid solution, finding that the sum of the covariances of each run must be positive definite in order to achieve a solution.

The closed-form solution is achieved by cautious mathematical interpretation of the problem, having been driven by a study from the general purpose solution to a detailed approach to the problem specificities. We prove that COSMIC can be derived through LU factorization and that the finite time solution has a complexity that varies linearly with the number of time steps, i.e. the number of multiplications performed by the algorithm is

$$c = (N-1)\left((p+q)^3 + (2p+3)(p+q)^2\right).$$

Moreover, when data matrices are ill conditioned, we also proposed a preconditioning step in order to obtain a stable algorithm.

The validation procedure was performed using a spring-mass-damper problem and, through a parametric study, we found that COSMIC performed well in the presence of sensor noise, for different values of λ_k, the regularization parameter for each time step k. We found the most fitting λ_k for this system and confirmed theoretical predictions that larger values of the tuning parameter derive less varying systems, as the one we proposed for validation. We achieved a solution that performs significantly better than more established approaches to the optimization problem, for instance the direct derivation for the whole trajectory, using `cvxpy`, and a Stochastic Block Coordinate Descent algorithm designed specifically for this problem.

Having performed the assessment of COSMIC, we proposed and implemented a dynamic programming approach for the controller design from the estimated model. The proposed setup leverages from

the Principle of Optimality to reach an optimal solution for the LTV LQR, followed by a tuning phase of the design parameters, ending with a controller. To validate the results, we tested different settings, achieving nominal performance for all cases and concluding that the controller design from COSMIC estimated systems is feasible and can represent a better solution than well-established techniques, that leverage for example from the approximation of the system to more well behaved counterparts.

Finally, and fulfilling the motivation that brought us to this problem, COSMIC was tested in a Low Fidelity Simulator of the Comet Interceptor mission and was used to identify the nonlinear attitude error dynamics and kinematics of the spacecraft. Then, to demonstrate the practical applicability of the system model and its usefulness in an engineering project, we developed a controller from the identified system, regarding the same dynamical programming framework, and testing the results in the original Low Fidelity Simulator. We found experimental evidence that the COSMIC method is a good option for facilitating the system identification phase of the design process and can speed up the controller synthesis, while automating the design process without the need of highly specialized system engineering.

We are confident that the results from this work will help develop the system identification procedure for LTV systems and the representation as a regularized least squares problem can be an extremely constructive for multiple settings. Moreover, the application in a real Space mission is a great step towards an industry that takes advantage of all the data available and we hope it is used in the future as a tool to expedite the early phases of development of the GNC and AOCS frameworks for different missions.

6.1 Future work

Regarding the possible adaptations of this work in the future and further developments, we state the obvious next step as the implementation in a Functional Engineering Simulator, that accounts for the environment characteristics and possible, difficult to model, disturbances. This would allow for a more compelling result and provide further guarantees that the nonlinearities of the system can be well approximated by the linear time-variant dynamics.

We can think about leveraging from the linear complexity with the number of time steps and fast computation times to possibly apply COSMIC in an online setting, which would entail a more judicious analysis of robustness and guarantees and an intense testing phase to be able to deploy the solution. COSMIC can also be used as a first approach to multiple control problem, such as adaptive control and Model Predictive Control.

Finally, it would be interesting to approach this formulation of the LTV system for more applications, as the novelty of the variation between instants restriction can be useful in other settings.

Bibliography

[1] ESA. 11th international ESA conference on guidance, navigation & control systems. `https://atpi.eventsair.com/QuickEventWebsitePortal/20a05-gnc-2020/website`, 2021. Accessed: 08-06-2021.

[2] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[3] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof. Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annual Reviews in Control*, 43:1–64, 2017.

[4] ESA. Assessment of Mission to Intercept a Long Period Comet or Interplanetary Object. CDF Study Report, ESA, Dec. 2019.

[5] S. Dean, N. Matni, B. Recht, and V. Ye. Robust guarantees for perception-based control. In *Learning for Dynamics and Control*, pages 350–360. PMLR, 2020.

[6] L. Ljung. System identification. In *Signal Analysis and Prediction*, pages 163–173. Birkhäuser Boston, 1998.

[7] S. N. Kumpati, P. Kannan, et al. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[8] M. Hardt, T. Ma, and B. Recht. Gradient descent learns linear dynamical systems. *Journal of Machine Learning Research*, 2016.

[9] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics.*, 2018.

[10] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni. System level synthesis. *Annual Reviews in Control*, 47:364–393, Jan. 2019. ISSN 1367-5788. doi: 10/gm3tnr.

[11] S. Oymak and N. Ozay. Non-asymptotic identification of lti systems from a single trajectory. In *2019 American Control Conference (ACC)*, pages 5655–5661. IEEE, 2019.

[12] T. Sarkar and A. Rakhlin. Near optimal finite time identification of arbitrary linear dynamical systems. In *International Conference on Machine Learning*, pages 5610–5618. PMLR, 2019.

[13] S. Dudul and A. Ghatol. Identification of linear dynamical time-variant systems using feedforward neural network. *IE (I) Journal*, 2004.

[14] S. Lin, H. Wang, and J. Zhang. System identification via meta-learning in linear time-varying environments. *arXiv:2010.14664*, 2020.

[15] S. Formentin and A. Chiuso. Control-oriented regularization for linear system identification. *Automatica*, 127:109539, 2021.

[16] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

[17] H. Mania, M. I. Jordan, and B. Recht. Active learning for nonlinear system identification with guarantees. *arXiv:2006.10277*, 2020.

[18] L. Vanbeylen, E. Louarroudi, and R. Pintelon. How nonlinear system identification can benefit from recent time-varying tools: the time-varying best linear approximation. In *52nd IEEE Conference on Decision and Control*, pages 4913–4918. IEEE, 2013.

[19] R. Dobbe, S. Liu, Y. Yuan, and C. Tomlin. Blind identification of fully observed linear time-varying systems via sparse recovery. *Automatica*, 100:330–335, 2019.

[20] C. Sánchez-Sánchez and D. Izzo. Real-time optimal control via deep neural networks: study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.

[21] D. Izzo and E. Öztürk. Real-time optimal guidance and control for interplanetary transfers using deep networks. *arXiv:2002.09063*, 2020.

[22] D. Izzo, D. Tailor, and T. Vasileiou. On the stability analysis of deep neural network representations of an optimal state feedback. *IEEE Transactions on Aerospace and Electronic Systems*, 57(1): 145–154, 2020.

[23] J. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 1999.

[24] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[25] K. J. Åström and R. M. Murray. *Feedback systems*. Princeton University Press, 2010.

[26] J. P. Hespanha. *Linear systems theory*. Princeton University Press, 2018.

[27] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2005. ISBN 0-470-01168-8.

[28] S. Rabanser, L. Neumann, and M. Haltmeier. Analysis of the block coordinate descent method for linear ill-posed problems. *SIAM Journal on Imaging Sciences.*, 2019.

[29] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. Dover Publications, New York, 06 1994. ISBN 9780486680293.

[30] O. Axelsson. *Iterative solution methods*. Cambridge University Press, 1996.

[31] S. Diamond and S. Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[32] A. Domahidi, E. Chu, and S. Boyd. Ecos: An SOCP solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[33] S. H. Zak. *Systems and control*, volume 198. Oxford University Press, 2003.

[34] J. Sola. Quaternion kinematics for the error-state Kalman filter. *arXiv:1711.02508*, 2017.

[35] P. Lourenço, J. Franco, T. Milhano, and J. Branco. Model Predictive Control for On-Board-Optimized Attitude Guidance for Cometary Fly-By - Problem Statement, Solutions and V&V Process. In *Proceedings of the 8th International Conference on Astrodynamics Tools and Techniques*, Sopot, Poland (Virtual), June 2021. ESA.

[36] E. A. Coddington and N. Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[37] M. Ben-Ari. A tutorial on Euler angles and quaternions, 2018. URL `http://www.weizmann.ac.il/sci-tea/benari/mathematics#rotations`.