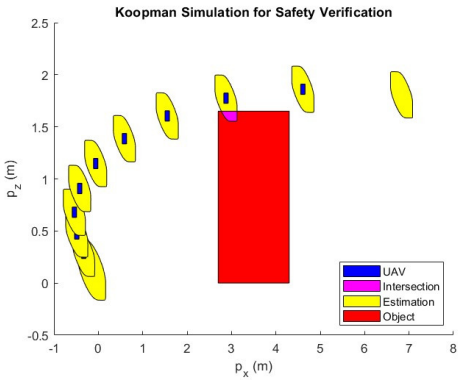
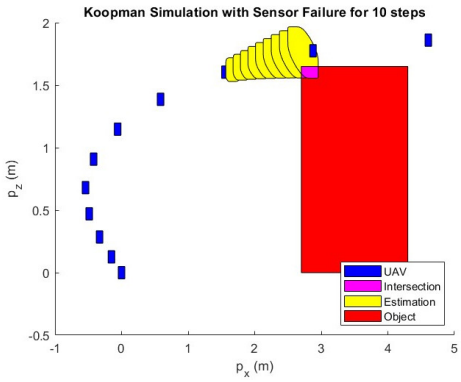
TÉCNICO LISBOA





Formal Safety Verification Using Reachability Analysis for the Koopman Operator

Beatriz Gonçalves Contente

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Daniel de Matos Silvestre
Prof. Rodrigo Martins de Matos Ventura

Examination Committee

Chairperson: Prof. Pedro Tiago Martins Batista
Supervisor: Prof. Daniel de Matos Silvestre
Member of the Committee: Prof. Rita Maria Mendes de Almeida Correia da Cunha

November 2024

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Firstly, I would like to thank my supervisors, Professor Daniel Silvestre and Professor Rodrigo Ventura for all the guidance, knowledge and patience provided throughout the development of this thesis.

Additionally, a thank you to my friends for being beside me all these years and whom without I would not have succeeded. I am the person I am today because of them.

To my Erasmus group, who is spread around Europe, thank you for our great time together and for continuously meeting up again.

A special thanks to my parents and brother, for encouraging me when I needed it the most and to my best friend, Inês Ventura, for going through everything I do.

Finally, I am forever grateful to Dário Bento for listening to me everyday, defending all my choices and whose love and support has been endless and free of judgment.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through project FirePuma (https://doi.org/10.54499/PCIF/MPG/0156/2019), through LARSyS FCT funding (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020, and 10.54499/UIDB/50009/2020) and through COPELABS, University Lusófona project 10.54499/UIDB/ 04111/2020.

Abstract

The increasing use of UAVs for civilian purposes has highlighted the critical need for reliable navigation systems that ensure safety, even in sensor faults. Securing this safety remains a key challenge for non-linear systems affected by bounded uncertainties. This work aims to develop an algorithm that predicts potential collisions between a UAV and obstacles close to its path, accounting for GPS sensor failure and state uncertainties due to measurement errors. The proposed solution applies set-based state estimation methods to reachability analysis. The developed work required an initial exploration of two methods for system identification, SINDy, and Koopman Theory, to obtain an approximated linearization of the dynamics through training data. The Koopman operator was adapted for collision prediction to support convex set operations, allowing state estimation as a constrained zonotope. This permitted the verification of the possibility of the intersection between the estimated state and the obstacle, allowing access to safety verification throughout the UAV's trajectory. As comparison, state estimation was also conducted using the Lagrange remainder error matrix, which over-approximates linearization errors during set propagation. Finally, the final algorithms also suggested the minimization of the over-approximation of the reachable sets for both methods through zonotope splitting.

Keywords

UAV; System Identification; Koopman Operator; Set-based Estimation; Constrained Zonotopes; Lagrange Remainder; Safety Verification.

Resumo

A crescente popularidade de veículos aéreos não tripulados (VANT) destacou a necessidade de sistemas de navegação fiáveis que garantam a segurança mesmo em casos de falhas nos sensores. Para sistemas não lineares afetados por incertezas limitadas, esta garantia continua a ser um desafio fundamental. Este trabalho tem como objetivo desenvolver um algoritmo que preveja potenciais colisões entre um VANT e obstáculos próximos da sua trajetória, considerando falhas no sensor GPS e incertezas de estado devido a erros de medição. A solução proposta aplica métodos de estimação de estado para conjuntos convexos usando análise de atingibilidade. O trabalho desenvolvido exigiu uma exploração inicial de dois métodos de identificação de sistemas, o SINDy e a Teoria de Koopman, para obter uma linearização aproximada da dinâmica através de dados do modelo. Para a previsão de colisões, o operador Koopman foi adaptado para suportar operações de conjuntos convexos, permitindo a estimação de estado como um zonótopo restrito. Isto permitiu verificar a possibilidade de intersecção entre o estado estimado e o obstáculo, permitindo aceder à verificação de segurança ao longo da trajetória do VANT. Como comparação, foi também realizada a estimação do estado utilizando a matriz de erros de Lagrange, que superaproxima os erros de linearização durante a propagação do conjunto. Por fim, os algoritmos finais sugeriram também a minimização da sobre-aproximação dos conjuntos alcançáveis para ambos os métodos através da divisão de zonótopos.

Palavras Chave

VANT; Identificação de Sistemas; Operador de Koopman; Estimação de conjuntos; Zonótopos; Restos de Lagrange; Verificação de segurança.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

CORA	Continuous Reachability Analyzer
DMD	Dynamic Mode Decomposition
GP	Genetic Programming
GPR	Gaussian Process Regression
ODE	Ordinary Differential Equation
PCE	Polynomial Chaos Expansion
SINDy	Sparse Identification of Nonlinear Dynamical Systems
SVD	Single Value Decomposition
UAV	Unmanned Aerial Vehicle

1

Introduction

Contents

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) have gathered a diverse range of applications throughout various fields as they offer the advantage of operating autonomously without the need for human intervention. Initially applied in military missions, its use has expanded to civilian purposes. In agriculture, UAVs allows for the monitoring of crop health, providing real-time high-resolution images In remote sensing, they are used for environmental assessments, forestry management, and resource monitoring By being equipped with multispectral sensors, they offer special information regarding wildlife tracking, mineral exploration and disaster assessment With many other applications regarding meteorological research, delivery services and media aiding, their versatility and low operational costs made them a valuable tool with a substantially growing market [1].

UAVs are widely used to navigate autonomously in both indoor and outdoor environments, which implies the importance of obstacle detection algorithms. In the autonomous navigation system, the sensor plays an important role. These sensors are used for environment perception, data sending from the base station to the hovering drone, mapping, surveying and photography, which can be used further for path planning and obstacle avoidance [2]. However, there are challenges related to accuracy in detecting and avoiding obstacles due to sensor faults, sensor noise, or data affected by outside disturbances.

Due to these challenges, UAVs have been studied and tested in various technologies, including fault diagnosis and fault tolerant control. In [3], the fault tolerant control problem is addressed, specifically the situation of inertial motion unit sensor failure The objective is to design a neural network-based estimator that allows to diagnose sensor faults and maintain acceptable control performance. Using this network, the angular rates are predicted based on previous states and control inputs, which allows for the recognition of fault and control based on state estimation.

Similarly, there is the need for the development of Fault Tolerant Control in case of other sensor failure or in more extreme cases like nonlinear systems subject to bounded uncertainties. The uncertainties stem from mismatched models (so-called process uncertainties) or from the inaccuracies of sensors (noises in the measurement output) [4] and can be represented by convex sets In [5], it is explored the reachable set estimation and safety verification problems for dynamical systems using these concepts. This safety verification is performed by examining the emptiness of the intersection between the over-approximation of reachable and unsafe sets.

The work developed in this thesis will apply the existing information regarding fault tolerant control under sensor failure and the set-based state estimation for uncertainties in the measurements to attempt to fill in the gaps regarding situations where both these concepts are needed. The development of this thesis focuses on small quadrotors, as they are optimized to fly in low-space environments. Still, the proposed methods can also be modified for other classes of UAVs.

1.2 Objectives and Outline of the Thesis

The main objective of this thesis is to develop a method that allows for the detection of an obstacle in the path of a UAV, even in the case of GPS sensor failure, with nonlinear dynamics and measurement errors in the sensors. The goal is for the UAV system to rely on state estimation for safety verification in these conditions.

There are a few challenges in developing the state estimation as the UAV's dynamics are nonlinear and, due to the GPS's sensor measurement errors, the exact value of the state at failure is within an interval of possible values. To aid this, there is a need to involve methods for system identification that allow for a linearization of the dynamics using training data. Afterward, it is also necessary to apply operations with convex sets to represent the state uncertainties and find an accurate approximation of the propagated state as a set. Considering this, there are intermediate goals that need to be met to achieve the final goal.

It is necessary to test system identification methods and apply one that allows a space in which the system's dynamics are known and linear. This is developed in Chapter 2 with an initial investigation of existing information on system identification and a later application of two methods, Sparse Identification of Nonlinear Dynamical Systems (SINDy) and Koopman Theory. In this chapter, the aforementioned UAVs system will also be presented. The final section consists of the presentation of the simulation results for both methods as well as an in-depth analysis concluding with the choice of which method to use in further work.

Chapter 3 is the core of the development of this thesis. It presents the problem statement at hand as well as background on obstacle detection and avoidance algorithms. It also provides the necessary mathematical background regarding convex sets Further along this chapter, there are proposed two possible solutions to this problem, the first one achieved by adapting the state of the art applicable to this thesis, and the second is the developed method consisting of combining the results from the previous chapter to the knowledge obtained regarding set operations The final part of this chapter is the presentation of the simulation results for these two methods along with an evaluation over the effectiveness in achieving the main objective.

In Chapter 4, there is the conclusion to this thesis, consisting in a recapitulation of the work developed and reached results, plus the discussion of the possible future related work.

1.3 Symbols and Notation

Throughout this dissertation, the symbols and notation used are represented as such:

Symbol	Description
x_i	element i of vector x
$\dot{x}$	time derivative of vector x
$x^\intercal$	transpose of vector x
$A^\intercal$	transpose of matrix A
$A_{m,n}$	matrix A with m rows and n columns
I_n	identity matrix with n rows and n columns
$\mathcal{Z}$	representation for zonotopes
$\mathbb{R}^n$	set of ordered n-tuples of real numbers
$\mathbb{R}^{n\times m}$	set of n by m matrices with real elements
$diag[x_1, x_2, \dots]$	diagonal matrix with elements x_i
$\|\| \cdot \|\|_{MSE}$	Mean Squared Error

Table 1.1: List of Symbols and Notation

2

System Identification

Contents

2.1 Introduction and Background

System identification is a procedure that approximates the mathematical models of dynamic systems from observed input-output data. It has applications in both cases where the equations of the system's model are unknown and, for specific methods, the creation of a linearized space for complex and high-dimensional nonlinear systems. These simplifications facilitate operations such as controller design, state estimation, and set-based propagation.

In the scope of system identification for nonlinear dynamic systems, various methodologies have already been developed to tackle the inherent complexities and challenges posed by nonlinearity each of them providing different advantages and capabilities. Some of these methods were explored during the progress of this thesis to confirm which is better inserted in the scope of the developed work.

2.1.1 Gaussian Process Regression

As explained in [6], while conventional regressions usually determine parameters directly, Gaussian Process Regression (GPR) establishes a covariance function to establish the relationship between data points. As a result, GPR can accurately estimate outputs and quantify uncertainty. Under the Bayesian framework, a prior Gaussian distribution must be specified for the regression parameter, leading to a Gaussian-distributed response variable. The covariance function determines the correlation between inputs and is crucial for making predictions. GPR uses maximum likelihood estimation to optimize the hyperparameters of this function. This method offers a non-parametric mean of managing nonlinear correlations in data, making it an effective instrument for system identification.

Even though GPR has been widely used for various modeling applications, there are still some unresolved issues with this method, such as the fact that most implementations model only a single response variable. This situation is approached in [6].

2.1.2 Genetic Programming

Genetic Programming (GP) is an evolutionary method for automatically generating nonlinear MISO (multiple input, single output) models [7]. It is undergoing rapid development, and many other variants have been developed, i.e., grammatical evolution, gene expression programming (GEP), Cartesian GP, linear GP, and semantic GP. GP can also be used to solve scientific and engineering problems like classification, time series prediction, and rule identification [7].

As stated in [8], GP mimics biological evolution to create models. It starts with an initial population of models by combining basis functions, inputs, and constants, structured in a tree-like fashion. In each iteration of the algorithm, every individual model in the population is fitted to the empirical data using nonlinear regression and then graded according to how well it matches the data. If the output data

is accurately predicted, the probability of a given model surviving into the next generation increases. Components of successful models are then recombined with others to form new models. These models can serve as a source for the next iteration, using operators like crossover, mutation, and permutations. Crossover combines parts of two models, mutation introduces random changes, and permutation switches parts within a model. The parameters for each new individual in the new generation are again determined by nonlinear regression and graded. This cycle continues until a stopping criterion is attained.

However, this method too presented disadvantages, such as the slow iterative evolution process when dealing with complex systems and the difficulty implementing and tuning the crossover and mutation operations, resulting in inaccurate solutions or increased computation expense [8].

2.1.3 Polynomial Chaos Expansion

As explained in [9] and based on the generalized Fourier expansion, Polynomial Chaos Expansion (PCE) represents a state of the art approach to parametric problems and uncertainty quantification. The performance and state of an engineering system rely on its constituent parameters, which in many cases are unknown. It is essential to find an explicit expression of the function relationship between these uncertain parameters and system outputs due to it being needed for system analysis and control. PCE has become a research focus of computational mathematics and has been applied to all kinds of static or dynamic problems in engineering fields such as electrical power systems, electric circuits, fluid dynamics, and control engineering [10].

The goal is to approximate the implicit parameter-output function with a globally optimal explicit polynomial function that maintains accuracy in the case of strong nonlinearity, contrary to the local Taylor expansion. The polynomial approximation used in PCE theory is the weighted and generalized Fourier expansion (GFE) with an orthogonal polynomial basis. This is optimal in the sense of probability measure and is globally optimal in the sense of weighted 2-norm error [9].

PCE's accuracy increases with the range of parameters and with the nonlinearity of the function, and the probability distributions of system outputs are calculated by combining this polynomial surrogate model with sampling methods. Since the surrogate model requires low computational cost while still being accurate and easy to sample, this method shows more efficiency than others alike [9].

2.1.4 Chosen Methods

While the previous method showed few disadvantages and could be applied to this thesis due to its management of nonlinear data, none of the presented methods actually allowed the linearization of complex nonlinear systems. Therefore, they do not solve the problem of the lack of a state space model

that permits a more straightforward design for controllers or state estimation that can be applied to obstacle detection. It also hampers the operations with convex steps, as these require a discretization of the system and the existence of state space matrices to allow linear mapping.

The chosen methods for system identification were the SINDy and the Koopman operator methods, explained in the further development. SINDy, similarly to the methods discussed, finds the ordinary differential equations of the model through data with low computational power and a simple algorithm. This will mainly be used to compare to the Koopman method. The Koopman operator is a way for system identification that, instead of finding the system's model, produces a space in which the system is linearized through a transformation of the states, which was the ideal approach.

2.2 System Dynamics

For the implementation of the aforementioned system identification methods, it was chosen two different system dynamics: a quadrotor UAV, which will be the system that is most used throughout the thesis, and a flexible two-link manipulator, to verify the identification algorithm's response to complex flexible structures.

2.2.1 Quadrotor System

The first system that is accounted for is a 12-dimensional nonlinear system of a 4-rotor UAV from [11]. When dealing with the control of a model of a multi-rotor UAV, it is important to note that the development and simulation of a dynamic UAV model is firstly necessary for the aspect of testing in the field of navigation and control in a closed room. This model has already been thoroughly developed and compared to the actual flight test results, and it is accurate enough to be used as a basis for simulating the UAV's dynamics. The reference system used is as such:

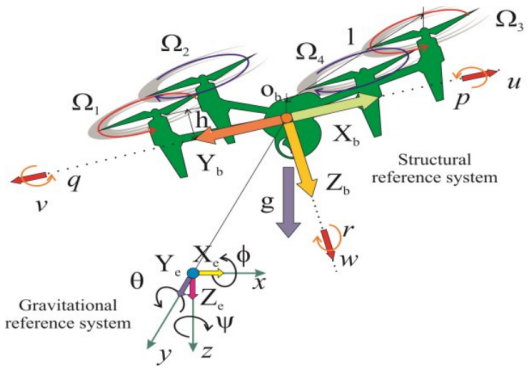


Figure 2.1: UAV System Reference

The original Ordinary Differential Equations (ODEs) for the dynamics of this system, based on [11],

have 9 state variables: the UAV's Euler angles (ψ, θ, ϕ), defined by a deviation $(-\pi < \psi < \pi)$, inclination $(-\frac{\pi}{2} < \theta < \frac{\pi}{2})$ and tilting $(-\frac{\pi}{2} < \phi < \frac{\pi}{2})$, its angular velocities $(v_\psi, v_\theta, v_\phi)$, which are the angles derivatives concerning time, and the current UAV speed (v_x, v_y, v_z), expressed in the reference system associated with the aircraft. Due to the necessity of further representing 3-dimensional obstacles in the same Koopman domain as the state variables, the system was augmented to include the positional variables (p_x, p_y, p_z) as a part of the state, which is now represented as $x = [p_x\ p_y\ p_z\ v_x\ v_y\ v_z\ \psi\ \theta\ \phi\ v_\psi\ v_\theta\ v_\phi]^\intercal$.

The chosen value for gravitational constant g was $9.81\ \mathrm{ms}^{-2}$, all moments of inertia (I_{xx}, I_{yy}, I_{zz}) were set as $1/6\ \mathrm{kg} \cdot \mathrm{m}^2$ since the UAV is symmetrical, it can theoretically be modeled with equal moments of inertia. Finally, the quadrotor's mass is $1\ \mathrm{kg}$ and length l is $1\ \mathrm{m}$. The ODEs of this dynamical system are:

$$\begin{aligned}
m\dot{v}_x &= -m(g\sin\theta - v_\theta v_z + v_\phi v_y) \\
m\dot{v}_y &= m(g\cos\theta\sin\phi - v_\phi v_x + v_\psi v_z) \\
m\dot{v}_z &= m(g\cos\theta\cos\phi - v_\psi v_y + v_\theta v_x) - u_1 \\
I_{xx}\dot{v}_\psi &= (I_{yy} - I_{zz})v_\theta v_\phi + lu_2 \\
I_{yy}\dot{v}_\theta &= (I_{zz} - I_{xx})v_\phi v_\psi + lu_3 \\
I_{zz}\dot{v}_\phi &= (I_{xx} - I_{yy})v_\psi v_\theta + lu_4,
\end{aligned} \tag{2.1}$$

where $u = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\intercal$ is the system's input. During hover, the thrust t_i and torque m_{Qi} for each rotor is estimated as:

$$t_i \approx b\Omega_i^2 \quad m_{Qi} \approx d\Omega_i^2 \tag{2.2}$$

where b is the thrust ratio in hover and d is the coefficient of resistance. The inputs can then be described by:

$$\begin{aligned}
u_1 &= \sum_{i=1}^{4} t_i = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
u_2 &= (-t_2 + t_4) = b(-\Omega_2^2 + \Omega_4^2) \\
u_3 &= (t_1 - t_3) = b(\Omega_1^2 - \Omega_3^2) \\
u_4 &= \sum_{i=1}^{4} (-1)^i m_{Qi} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2).
\end{aligned} \tag{2.3}$$

2.2.2 Flexible Bar System

A Finite Element Model for Flexible Two-link Manipulators was also considered, representing a complex system due to its intricate nonlinear dynamics and high dimensionality, with 12 states governing its behavior [12].

As shown in Figure 2.2, a two-link flexible link manipulator is modeled as an Euler-Bernoulli beam, which is developed in [13]. The manipulator has rotary joints moving in the horizontal plane (revolute

joints) in the open kinematics chain configuration and bending deformations that are restricted in the motion plane. The flexible manipulator is of type fixed-fixed, fixed-free. The two links are connected using motor 2 of mass M_1. Motor 1 is fixed at the origin of the fixed base frame XOY. The free tip of the second link has a payload attached of mass M_2. P_1 and P_2 are the points on the flexible manipulator. The flexural rigidity of the links is represented by EI_1 and EI_2.

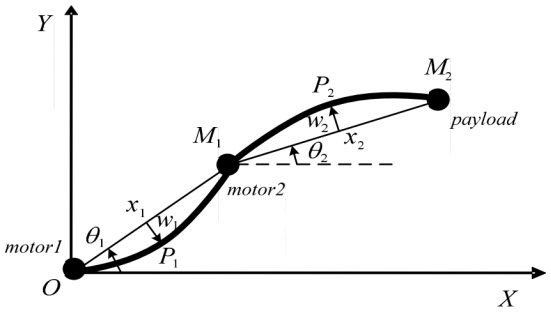


Figure 2.2: Two-link Flexible Manipulator Reference

The values chosen for the system's parameters are as such:

	Link 1	**Link 2**
Motor Mass/ Payload (kg)	0.8	0.5
Length (m)	1	1
Link Mass (kg)	2	2
EI $(\mathrm{N} \cdot \mathrm{m}^2)$	2	2

Table 2.1: Two-link Flexible Manipulator Parameters

Using Lagrange's dynamics formulation, the dynamic equations of motion are obtained from the finite element model applied to multi-link flexible manipulators [12]. The closed-form dynamic equations of motion for flexible link manipulators can be derived using symbolic manipulation software and result in the following equations of motion:

$$\begin{bmatrix} M_{rr} & M_{rf} \\ M_{rf}^{\mathsf{T}} & M_{ff} \end{bmatrix} \begin{bmatrix} \ddot{q_r} \\ \ddot{q_f} \end{bmatrix} + \begin{bmatrix} h_r(q,\dot{q}) \\ h_f(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} c_r(q) \\ c_f(q) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & N \end{bmatrix} \begin{bmatrix} q_r \\ q_f \end{bmatrix} = \begin{bmatrix} \Gamma \\ 0 \end{bmatrix}. \tag{2.4}$$

In this equation, the vector $q = \begin{bmatrix} q_r^{\mathsf{T}} & q_f^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$, is the vector of generalized joint, $q_r = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^{\mathsf{T}}$, and the vector of flexible deformation variables, $q_f = \begin{bmatrix} \delta_{11} & \delta_{12} & \delta_{21} & \delta_{22} \end{bmatrix}^{\mathsf{T}}$. This system regards any multi-link flexible manipulator. The system used is a two-link flexible manipulator so that the state vector will be $x = \begin{bmatrix} \theta^{\mathsf{T}} & \delta^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$ with

$$\theta = \begin{bmatrix} \theta_1 & \dot{\theta_1} & \theta_2 & \dot{\theta_2} \end{bmatrix}^{\mathsf{T}} \tag{2.5}$$

$$\delta = \begin{bmatrix} \delta_{11} & \dot{\delta_{11}} & \delta_{12} & \dot{\delta_{12}} & \delta_{21} & \dot{\delta_{21}} & \delta_{22} & \dot{\delta_{22}} \end{bmatrix}^{\mathsf{T}}. \tag{2.6}$$

The vector $h = \begin{bmatrix} h_r(q,\dot{q})^\intercal & h_f(q,\dot{q})^\intercal \end{bmatrix}^\intercal$ represents the Coriolis and centrifugal terms, while the vector c represents the gravitational terms. The matrix N is the flexural structure stiffness matrix of the system, and the matrix M is the configuration mass. The expressions for M, h and N [13], are in the Attachments. The resulting ODEs were calculated in *Python* by solving the matrix equation.

2.3 Sparse Identification of Nonlinear Dynamical Systems

According to [14], SINDy has emerged as a powerful tool in the realm of data-driven modeling, offering a systematic approach to uncovering governing equations from observed data. Rooted in the fields of applied mathematics and machine learning, SINDy addresses the fundamental challenge of extracting concise yet accurate representations of complex dynamical systems directly from empirical data. At its core, SINDy leverages the principles of sparsity and model parsimony to identify the governing dynamics of a system, thereby enabling concise models that capture the underlying physics or dynamics driving observed behaviors. With its ability to distill intricate dynamics into mathematical expressions, SINDy has found applications across various domains, offering a data-centric approach to uncovering the governing laws that shape natural and engineered systems alike.

2.3.1 Mathematical Background

The dynamical system considered is as such:

$$\dot{x} = f(x(t)), \tag{2.7}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system at time t and $f(x(t))$ is the function that represents the dynamics of the system [15]. The focus lies in exploiting the fact that many physical systems can be described by only a few relevant terms, resulting in sparse governing equations in a high-dimensional nonlinear function space. The dynamical system discovery is done through sparse regression and compressed sensing [14].

To determine the function f from data, it is necessary to gather a time series of the system's state and to obtain the state's derivative, either through direct measurement or numerical computation. The collected data is sampled at various time points and organized into two matrices [16].

$$X = \begin{bmatrix} x^\intercal(t_1) \\ x^\intercal(t_2) \\ \vdots \\ x^\intercal(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{2.8}$$

$$\dot{X} = \begin{bmatrix} \dot{x}^\intercal(t_1) \\ \dot{x}^\intercal(t_2) \\ \vdots \\ \dot{x}^\intercal(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix} \tag{2.9}$$

The next step would be to construct a library ($\Theta(\mathsf{X})$), whose columns would be candidate nonlinear functions that might describe the system's dynamics [14].

$$\Theta(\mathsf{X}) = \begin{bmatrix} 1 & \mathsf{X} & \mathsf{X}^{P_2} & \mathsf{X}^{P_3} & \cdots & \sin(\mathsf{X}) & \cos(\mathsf{X}) & \cdots \end{bmatrix} \tag{2.10}$$

In the equation above, polynomial terms of degree n are represented by X^{P_n}, which denotes the n-degree nonlinearities in the state.

$$\mathsf{X}^{P_2} = \begin{bmatrix} x_1^2 & x_1x_2 & \cdots & x_2^2 & x_1x_3 & \cdots & x_n^2 \end{bmatrix} \tag{2.11}$$

Only a few of these terms are probably active in each row of the function f [14]. The matrix $\Theta(\mathsf{X})$ is of size $m \times p$, with $m \times p$, since there will be more date samples than functions. When simulating the SINDy algorithm, it is practical to experiment with different function bases to determine the correct one to represent the dynamics in. From this, a sparse regression problem is set up to determine the sparse vectors of coefficients (ξ_k) that decide which terms are active [16].

$$\dot{\mathsf{X}} = \Theta(\mathsf{X})\boldsymbol{\Xi}, \quad \boldsymbol{\Xi} = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_n \end{bmatrix} \tag{2.12}$$

Performing a standard regression for each ξ_k typically yields solutions with non-zero contributions in each element. However, if one aims for sparsity in ξ, with most entries being zero, a solution would be to introduce an L^1 regularization term into the regression. This leads to the LASSO technique, where the parameter μ weighs the sparsity constraint, which will further correspond to the threshold of the optimizer [15].

$$\xi_k = \arg\min_{\xi_k'} ||\Theta\xi_k' - \dot{x}_k||_2 + \mu||\xi_k'||_1. \tag{2.13}$$

Once each of the objects X, $\dot{\mathsf{X}}$, $\Theta(\mathsf{X})$ and $\boldsymbol{\Xi}$ is determined, it is possible to write the approximation problem underlying SINDy:

$$\dot{x} = f(x) \approx \boldsymbol{\Xi}^\intercal \left(\Theta\left(x^\intercal\right)\right)^\intercal \tag{2.14}$$

However, the equation (2.14) is not exact due to the fact that, for most cases, the derivative of the state $\dot{\mathsf{X}}$ is not available and must be obtained through a differentiation method. This implies that the matrices X and $\dot{\mathsf{X}}$ are contaminated with noise. The equation (2.12) could be adapted to add a matrix Z consisting of independent identically distributed Gaussian entries with zero mean and noise magnitude

η [14].

$$\dot{\mathrm{X}} = \Theta(\mathrm{X})\boldsymbol{\Xi} + \eta\mathbf{Z} \tag{2.15}$$

To counteract the differentiation error derived from the computation of $\dot{\mathrm{X}}$, it is possible to use the total variation regularized derivative to remove the noise from the derivative, based on the total variation regularization [14].

As [14], the next objective is to find a sparse solution to an overdetermined system with noise. A possible solution is to implement the sequential thresholded least-squares algorithm. This algorithm derives a least-squares solution for $\boldsymbol{\Xi}$ and subsequently applies a threshold (μ) to all coefficients below a specified cutoff value. After identifying the indices of the remaining non-zero coefficients, another least-squares solution is computed onto the leftover indices. The resulting coefficients undergo another thresholding step using μ, and this process continues until convergence is achieved for the non-zero coefficients. This algorithm exhibits computational efficiency and quickly converges to a sparse solution within a limited number of iterations while also having the advantage of only depending on one parameter μ [16].

2.3.2 PySINDy Algorithm

In this section, the transition from the SINDy theoretical discourse to practical implementation is undertaken, building upon the foundational principles elucidated in the preceding sections. The objective is to present the overall SINDy model for *Python* and the choices made to adapt to the system being used.

The PySINDy package revolves around the SINDy class which consists of three primary components, one for each term in the above approximation problem (2.15). As explained in [17], to instantiate a SINDy object, one invokes *ps.SINDy*, in which it is possible to set the different modules. The first one is *pysindy.differentiate*, which performs numerical differentiation to compute the derivative of the states ($\dot{\mathrm{X}}$) in the case where its measurement is not possible. In the further simulations, this will always be the case. The next module, *pysindy.featurelibrary*, regards the library of candidate functions $\Theta(\mathrm{X})$. The latter, *pysindy.optimizers*, provides a set of solutions for the sparse regression problem for determining $\boldsymbol{\Xi}$. Finally, it is necessary to set the *feature.names*, which are the names of the state variables, and do a model fit to the X matrix with the state data [16].

In the algorithm below is the pySINDY model for a generic 2-state system [16].

Algorithm 1 PySINDy Algorithm

```
differentiation method = ps.FiniteDifference(order=1)
fourier library = ps.FourierLibrary(n frequencies=1)
polynomial library = ps.PolynomialLibrary(degree=1)
lib = polynomial library + fourier library
optimizer = ps.STLSQ(threshold=0.2)
model = ps.SINDy(differentiation.method=differentiation.method, feature.library=lib, optimizer=optimizer, feature.names=["x1","x2"])
model.fit(X, t=t)
```

The explanation regarding the hyperparameters of the pySINDy algorithm derives from both [17] and [18], as they corroborate each other. The differentiation method regards the computation of the derivative of the states $\dot{X}$. For most cases, including this one, the method chosen is the finite difference method, and it allows to select the order of differentiation. If the data is smooth (at least twice differentiable), then finite difference methods give accurate derivative approximations. However, finite difference methods tend to amplify noise in data. In the case of noisier data, choosing a more advanced differentiation method might be necessary. One can use smoothed finite difference derivatives, which performs differentiation by smoothing input data and then applying a finite difference method.

The feature library module allows the users to specify the set of library functions. It is possible to choose between already existing libraries or to use a generic library where one can customize the set of functions to be included as features. For the systems used further, this latter will be used along with the identity library, which corresponds to the identity function, the Fourier library, which includes the trigonometric functions and requires an input of the number of frequencies to take into account and the polynomial library, corresponding to the previously mentioned X^{P_n} and that allows to set the maximum degree of the polynomials to include.

For the optimizer, the method chosen is the aforementioned sequential thresholded least-squares algorithm (STLSQ). While deciding which threshold value to use, it is important to note that if the value is too low, the cutoff value for the coefficients in Ξ will not be fit to achieve enough zero coefficients, and the approximated output equations of motion will present too many terms of Θ. On the other hand, if the value for the threshold is too high, the cutoff value can make it so that all the coefficients are zero, preventing the model from finding the correct equations. The threshold has to be chosen according to each model, as the optimal value varies. If the accurate equations of motion are available, this choice can be made by simply experimenting with different values and observing the output. If this is not the case, one can calculate the mean square error of the original data with the data from the approximated equations and use this to choose a value for μ.

2.4 Koopman Operator

According to [19], contrary to the previous system identification methods discussed, the Koopman operator is a tool in dealing with nonlinear systems that allow for a simplification of their complexity by creating a space in which they take a linear form. This is achieved by equivalently representing the nonlinear system by a higher-dimensional one. Consequentially, the Koopman operator then offers an advantage for the prediction, control, and verification of dynamical systems.

The transformation to the Koopman domain makes use of an infinite-dimensional operator, which implies approximations due to its computational expense. Despite the importance of these approximations, there is a lack of a standardized approach, making it likely to use less structured methods. On the other hand, while the Koopman operator shows promise in handling nonlinear systems, its connections to established theoretical and dynamical system concepts are frequently overlooked [19].

2.4.1 Mathematical Background

Considering a dynamical system with its equations following the same formula as (2.7), where $x \in \chi \subseteq \mathbb{R}^n$ is the state of the system and f the vector field that describes its dynamics [20]. This function might also depend on time, t, external actuation or control, u, or other parameters.

The system in question may be either nonlinear or of unknown dynamics, requiring data-driven techniques to identify and analyze dynamical systems. Regarding nonlinearity, operator-theoretic methods for dynamical systems are gaining traction. As demonstrated further, nonlinear dynamical systems can be effectively represented using infinite-dimensional yet linear operators, exemplified by the Koopman operator [21]. The objective would be to find a new set of coordinates z from the existing x [20].

$$z = g(x) \tag{2.16}$$

that allows for the simplification of dynamics and, ideally, linearization

$$\dot{z} = Tz. \tag{2.17}$$

The end goal is to find a linearizing coordinate transform so that $z_{k+1} = Kz_k$, where the matrix K is the discrete-time analog of the continuous-time matrix T [20]. This is given by the eigenfunctions of the discrete-time, infinite-dimensional Koopman operator, κ.

Ideally, one would have access to the data measurements of the states of the system, which are governed by the discrete-time model:

$$x_{k+1} = F(x_k) \tag{2.18}$$

with $x_k = x(k\Delta t)$ [21]. This is known as a flow map of the system, represented by F for discrete systems. For continuous systems, it is defined by F_t and is used to advance initial conditions $x(0)$ forward along the trajectory so that trajectories evolve according to

$$x(t) = F_t \cdot (x(0)). \tag{2.19}$$

When the system is discrete, $t \in \mathbb{N}$, and the dynamics are autonomous, then F_t is a repeated composition of $F \equiv F_1$ given by $F_t(x) = F(F(\cdots(F(x))))$. The discrete-time dynamics are more general than the continuous-time formulation, encompassing discontinuous and hybrid systems as well.

However, this flow map model is not possible for nonlinear systems, so an alternative is necessary. The Koopman operator, similar to the flow map, instead advances measurement functions of the state with the flow of the dynamics. This is done by considering real-valued measurement functions, $g : M \to \mathbb{R}$, known as observables and elements of an infinite-dimensional Hilbert space [21]. Generally, the Lebesgue square-integrable functions on M provide the Hilbert space. The Koopman operator acts on these functions as:

$$\kappa_t g = g \circ F_t, \tag{2.20}$$

where $\circ$ is the composition operator. For a discrete-time system with a time step of Δt, the equation above becomes:

$$\kappa_{\Delta t} g(x_k) = g(F_{\Delta t}(x_k)). \tag{2.21}$$

So the general formula for discontinuous systems becomes:

$$\kappa g(x_k) := g(F(x_k)) = g(x_{k+1}). \tag{2.22}$$

It is possible to conclude that the Koopman operator defines an infinite-dimensional linear dynamical system that advances the observation of the state $g(x_k)$ to the next step, for any observable function g and any state x_k [21], for when the flow map is not available to directly advance the measurements of the state.

The dynamics of the Koopman linearized system can then be formulated as

$$g(x_{k+1}) = A\, g(x_k) + B\, u_k, \quad A \in \mathbb{R}^{p\times p},\ B \in \mathbb{R}^{p\times m}, \tag{2.23}$$

where

$$K_{\Delta t} = [A, B].$$

2.4.1.A Koopman Eigenfunctions

As stated in [21], the linear nature of the Koopman operator is useful, yet its infinite dimensionality presents challenges for both representation and computation. Rather than encompassing the evolution of all measurement functions within a Hilbert space, applied Koopman analysis strives to isolate crucial measurement functions that exhibit linear evolution alongside the dynamics' flow.

Actually, eigenfunctions of the Koopman operator offer a collection of unique measurements that display linear behavior over time and a central incentive for embracing the Koopman framework lies in its capacity to streamline dynamics through the eigen-decomposition of the operator.

For an eigenfunction φ of κ, corresponding to an eigenvalue λ, the following equations apply, for discrete and continuous-time, respectably:

$$\varphi(x_{k+1}) = \kappa_{\Delta t}\varphi(x_k) = \lambda\varphi(x_k) \tag{2.24}$$

$$\dot{\varphi}(x) = \kappa_t\varphi(x) = \lambda_t\varphi(x). \tag{2.25}$$

Koopman eigenfunctions can be obtained either from data or analytical expressions and stand as a pivotal applied hurdle in contemporary dynamical systems. Uncovering these facilitates globally linear representations of highly nonlinear systems. By combining (2.25) with the chain rule to the time derivative of the Koopman eigenfunctions, it is possible to obtain the partial differential equation (2.26). This assumes that the dynamics are both continuous and differentiable and allows to approximate the eigenfunctions:

$$\Delta\varphi(x) \cdot f(x) = \lambda\varphi(x). \tag{2.26}$$

The objective with these is to transform the nonlinear dynamics into completely linear in eigenfunction coordinates, which are given by $\lambda\varphi(x)$. This can be obtained from the above nonlinear PDE, either by solving for the Laurent series or with data via regression [20] which is the case for the algorithm used further.

2.4.1.B Singular Value Decomposition

Per [22], singular value decomposition is based on the following linear algebra theorem from linear algebra: a matrix A can be broken down into the product of three matrices - an orthogonal matrix U, a diagonal matrix S, and the transpose of an orthogonal matrix V. The theorem is usually presented as such:

$$A = USV^{\intercal}, \tag{2.27}$$

where $U^{\intercal}U = I$, $V^{\intercal}V = I$; the columns of U are orthonormal eigenvectors of $AA^{\intercal}$, the columns of V are orthonormal eigenvectors of $A^{\intercal}A$, and S is a diagonal matrix containing the square roots of eigenvalues from U or V in descending order (singular values) [22].

While Single Value Decomposition (SVD) has many applications, it is also a method for identifying and ordering the dimensions along which data points exhibit the most variation [22]. This was useful to check whether the matrix A from the Koopman algorithm was going to enlarge the initial zonotope, and if so, what direction would most be affected and what was the scale factor. The objective was to obtain a low maximum value for S, as close as possible to 1, since this would mean that the uncertainties of the initial state would remain the same throughout the propagation in the Koopman domain, decreasing the propagation error and the chances of intersection with the obstacle zonotope.

2.4.2 AutoKoopman Toolbox

The application of Koopman operator linearization has led to significant advancements in predicting, controlling, and verifying dynamical systems [23]. However, the effectiveness of its outcomes is heavily reliant on properly tuning hyperparameters, such as the number of observables. The AutoKoopman toolbox [23], a Python package, addresses this challenge by automating the process of learning precise models within a Koopman linearized framework, requiring minimal effort. It offers various tuning strategies to automatically optimize the associated hyperparameters of Koopman operator techniques. Supporting both discrete and continuous-time models, AutoKoopman incorporates all major observable types, including polynomials, random Fourier features, and neural networks.

Note that, from the section above, it was concluded that for each nonlinear system, there exists a space which enables the Koopman linearization. The technique used to learn an infinite dimensional linear operator is explicit and the objective is then to find a finite space in which the dynamics can be approximated by a linear operator. This leads to the optimization problem:

$$\underset{g,\kappa}{\arg\min} \sum_{(x_k,u_k)\in\tau} ||g(x_{k+1},u_{k+1}) - \kappa_{\Delta t} g(x_k,u_k)||_{MSE}. \tag{2.28}$$

There is an alternative strategy to finding the optimal solution that is less impractical, which would be to pursue a solution that is close to optimal using heuristic methods. This typically involves fixing the observables and then finding the Koopman operator. It is a common practice to consider observables that only act on the states [23].

Below is the algorithm used for further simulations regarding the features and implementation of the AutoKoopman package [23].

Algorithm 2 AutoKoopman Algorithm

```
fromautokoopman import autokoopman
experimentresults = autokoopman(
trainingdata,
samplingperiod=dt,
obstype='rff',
opt='grid',
nobs=150,
normalize=True,
verbose=True)
```

The training data regards a matrix with the vectors for the time samples considered, the values for the states at those times and the inputs generated. The sampling period used for all the further simulations is 0.01 seconds. In order to input a different vector u for each time step, necessary to both randomly excite the system or to implement the LQR controller, the training data was obtained by taking the last data point from the output vector of the *odeint* function for each period.

The data is first transformed into a normalized space through a scale factor (s) and an offset (m), before applying the observables that transform the state into the Koopman domain. Both of these values are stored in a data file to allow the transformation of other data to the Koopman domain.

The observable type chosen was the aforementioned random Fourier feature, for 150 values of frequency. Random Fourier features approximate kernel Dynamic Mode Decomposition (DMD) using extended DMD. Kernel DMD belongs to a class of algorithms that utilize symmetric and positive definite kernel functions to encode the similarity between two observations. These kernels are especially beneficial in cases like the one considered, as they enable efficient similarity computations without the need for explicit data mapping into the high-dimensional space.

In this context, an explicit representation of the observable mapping g is required, so the relationship between a kernel and its Fourier transform is exploited, thereby maintaining the beneficial properties of kernels while providing an explicit observable mapping.

The function $g_x(x) = [g_1(x), \ldots, g_p(x)]$ approximately represents the kernel function and the scalar observables are defined as:

$$g_i(x) = \sqrt{2}\cos(\omega_i^{\mathsf{T}} x + \alpha_i), \quad i = 1, \ldots, p, \tag{2.29}$$

where $\alpha_i \in [0, 2\pi]$ is uniformly selected, and $\omega_i \in \mathbb{R}^n$ is drawn from a probability distribution, in this case, normal distribution, corresponding to the kernel function [23].

Per [23], the algorithm implements different methods for solving the optimization problem, such as DMD, where the Koopman operator is found using extended Dynamic Mode Decomposition (eDMD). In eDMD, matrices X, X', and U are constructed from trajectory data. The Koopman operator $K_{\Delta t}$ is optimized by minimizing the mean square error, resulting in $K_{\Delta t} = X'[X, U]^+$, where $[X, U]^+$ is the

Moore-Penrose inverse. Due to the large size of $[X, U]$, a low-rank approximation via singular value decomposition is used for computational efficiency. For continuous-time systems, X' is replaced by time-derivatives of the observables. Similarly to the SINDy algorithm, AutoKoopman also makes use of sparse regression to obtain dynamics with only few nonlinear terms, where the nonlinear dynamics is represented as a symbolic closed-form expression.

Finally, regarding hyperparameter tuning, the method used is the grid search, which consists of sampling values for said parameters by generating candidates from a grid. AutoKoopman provides default search ranges for all hyperparameters but it is possible to also manually specify ranges. Due to grid search being affected by dimensionality, it works best for low-dimensional spaces such as random Fourier feature observables, which require less hyperparameters [23].

2.4.3 Transformation to Koopman Domain

The algorithm above returns the discrete linearization matrices A and B from (2.23). The matrix A will be 162×162, which correspond to the 150 frequencies used plus the number of original states, 12, while the matrix B will be 4×162. These matrices allow for the propagation of the states in the Koopman Domain using $x_{k+1} = Ax_k + Bu_k$.

For this, it is necessary to transform the coordinates of the initial states in $\mathbb{R}^{12}$. The method used is similar to the theory. First, it is necessary to transform the initial state to a normalized space using the scale factor s and the offset m. Afterward, the equation (2.29) is applied to construct an initial state for the Koopman model by mapping through the observable functions. In this case, the multiplier $\sqrt{2}$ will be changed to $\sqrt{\frac{2}{D}}$, where D corresponds to the number of frequencies used. The process to transform the initial state x_0 into the initial Koopman domain state z is then:

$$x_{norm} = s \cdot x_0 + m \qquad z = \begin{bmatrix} x_{norm} \\ \sqrt{\frac{2}{D}} \cdot \cos(\omega \cdot x_0 + \alpha^\intercal) \end{bmatrix}. \tag{2.30}$$

Using the information from the sections above, it was now possible to implement both the SINDy algorithm and the Autokoopman toolbox for the presented systems of the UAV and flexible bar.

2.5 Results

2.5.1 PySINDy

Since the PySINDy algorithm heavily relied on the hyperparameter choices and the threshold value could easily be chosen in a way that would eliminate too many or too few ODE coefficients, it was initially done a test in the algorithm's response to differently sized systems. This was done for the following systems:

a 2-state simple pendulum [24], a 4-state coupled pendulum [25], a 6-state pendulum with a nonlinear damping term and a spring attached to it and an 8-state mass-spring-damper system with additional nonlinear elements inspired by systems in [26] but altered by the author, the equations for these are in the Attachments. The systems chosen were simple and similar in equations so as to verify the effects of the dimensional increase. The initial conditions for these simulations were random vectors. All the simulations had values for the order of differentiation and threshold individually adjusted for the best results.

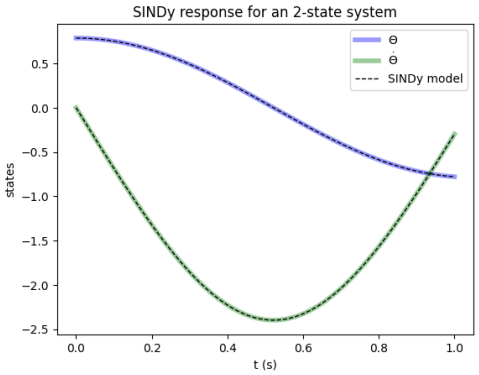


Figure 2.3: SINDy Estimation Model for a 2-State System

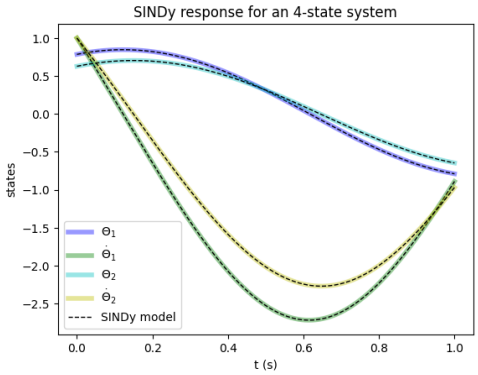


Figure 2.4: SINDy Estimation Model for a 4-State System

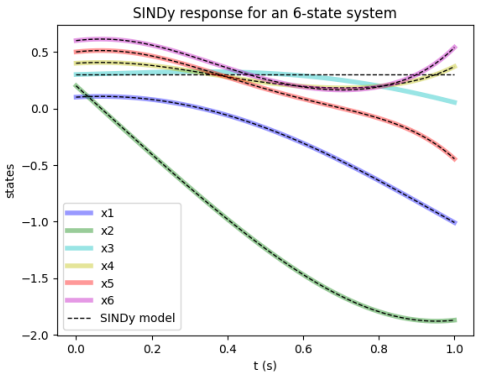


Figure 2.5: SINDy Estimation Model for a 6-State System

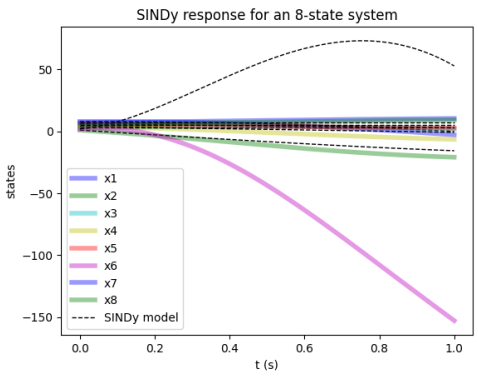


Figure 2.6: SINDy Estimation Model for an 8-State System

As it is possible to observe, the SINDy response becomes more inaccurate with an increased number of states. For the 2 and 4-state systems, all the ODEs were correctly obtained; while the 6-state has one of the ODEs set to zero, due to the threshold value, and the 8-state system has multiple mismatched trajectories. While the UAV system is less complex than the 8-state system used, it is of a higher dimension and likely to be affected by this threshold parameter problem.

Regarding the UAV model, the values for the hyperparameters were as such - an order of differentiation of 2, seeing as the data was smooth and noiseless, a Fourier library with 1 frequency and a polynomial library of degree 2, to match the nature of the original ODEs, and a threshold of 0.1, to

compromise as much as possible the elimination of extra coefficients while maintaining non-zero output ODEs. The inputs were random values between -1 and 1 and the initial states were set at zero. The results for the UAV's SINDy model are in the plot below.

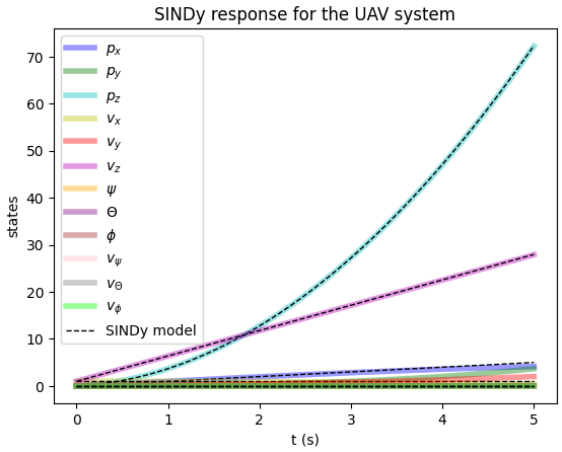


Figure 2.7: SINDy Estimation Model for the UAV system

As it is possible to observe, the SINDy model seemingly is able to approximate the original ODEs, despite the UAV's high dimensionality. However, this is not totally exact as the system was only able to find the ODEs for the state variables p_x, p_z and v_z, with the remaining ones set at zero due to the threshold value being too high. On the other hand, a smaller value for the threshold would make the correctly estimated ODEs have extra coefficients that would increase computational expense and inaccurately approximate the system.

Regarding the flexible bar model, the hyperparameter chosen were an order of differentiation of 2, a polynomial library of degree 1, and a Fourier library of 1 frequency. The results for the SINDy response are as such:

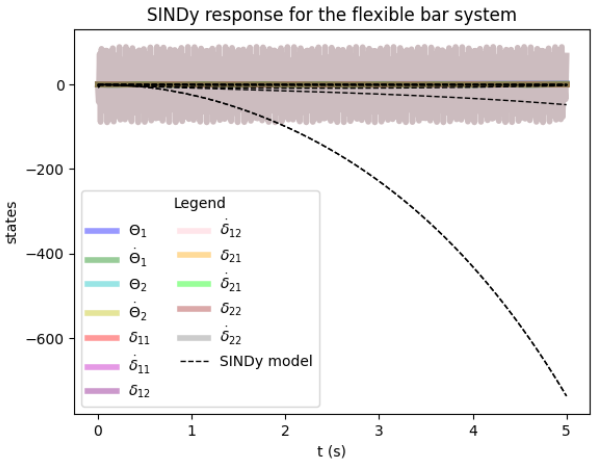


Figure 2.8: SINDy Estimation Model for the Flexible Bar System

Similarly to the UAV, the flexible bar's nonlinearity and high dimensionality of this system made the pySINDy algorithm unable to accurately simulate the SINDy model. Moreover, in this case the result is even more extreme since all but three ODEs were set at zero due to too many terms being truncated. It is important to note that the chosen threshold is a highly excessive cutoff value that is bound to eliminate essential coefficients, however, any value lower would deem the model nearly impossible to compute due the great amount of extra coefficients. It also seems that the real response for some state variables has too much noise to even accurately be approximated due to the nature of this system. To better fit the data, the algorithm would require interaction terms that are not included in the library of candidate functions and more considerable computational power.

Generally, it seems that high dimensional systems have a problem with data-driven identification. Beyond systems with a few variables, generally one will end up with small errors in the coefficients, from noise or finite sampling, that cause finite-time blow up of the model.

The solution to the problem found with data-driven identification is not within the scope of this thesis. This was explored as a background for the further development of the Koopman analysis and its advantages facing the SINDy algorithm. However, two solutions were thought of regarding the situation with the differentiation method and the threshold value.

Firstly, it would be relevant to experiment with more possible differentiation methods to test if this model blow-up occurs often or if it is just a particularity of the finite difference method. Notably, the forward Euler method would be interesting as it is a simple first-order method, in which the global error is proportional only to the step size and should not amplify with the number of states [27].

An algorithm developed to set different thresholds for different variables might also help solve the threshold compromise problem. However, this would either assume that the original ODEs are available, so to check whether the output equations are accurate enough or a different μ value is needed, or it would require a cycle in which several combinations of threshold values would be tested with the mean squared error, so as to pick the combination that outputs results closest to reality.

2.5.2 AutoKoopman

The AutoKoopman model output for the UAV's dynamic, using the parameters and algorithm expressed in the section above, was computed by appending six different cycles of 50 times steps with duration of 0.01 seconds. In each step a random value for the input vector u is introduced, within a range between -1 and 1 for each element. The objective was not to control the system but to randomly excite it so the Koopman algorithm could identify the system's dynamics more accurately. The presence of multiple trajectories allows for a better approximation of the real dynamics by the algorithm.

In the plot below, these six trajectories are represented in two dimensions for the state variables p_x and p_y, calculated using both the system's dynamics and the AutoKoopman algorithm.

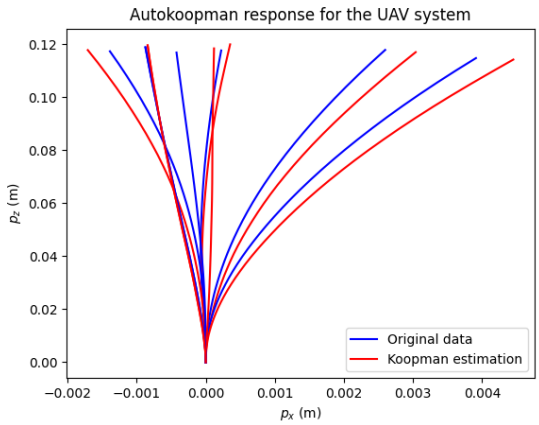


Figure 2.9: AutoKoopman Estimation Model for the UAV system

Despite some trajectories having a slight error, they still allowed the algorithm to accurately approximate the real system's dynamics, as it is possible to verify that most estimated trajectories are very similar to the original ones while the paths with a larger error still follow a similar curve .

Regarding the SVD analysis, the value for S_{max} in the chosen Koopman matrix A was 1.0649. This value is close to the desired value of 1 so the initial zonotope will not greatly expand for a small number of steps. However, the effect of this value on propagation for a longer horizon will inevitably impact the shape and size of the final state zonotope and will introduce errors in the intersection with the obstacle, due to its decimal points exponentially increasing with the number of steps.

For the flexible bar, the algorithm's parameters were set the same, with six cycles of 50 steps, but with an empty input vector, as it is not necessary in this model. Below is the output of this simulation:

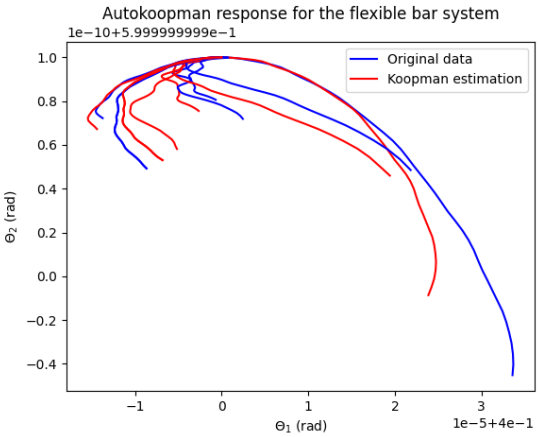


Figure 2.10: AutoKoopman Estimation Model for the Flexible Bar System

Similarly to the UAV system, the Autokoopman toolbox seems to approximate the system much more accurately than the PySINDy algorithm, as the estimation closely follows the original data's path, specifi-

cally in the central region of the response curve. However, due to the flexible bar's higher dimension and complexity, there are more considerable discrepancies between them near the end of some trajectories. There is also some noise in the Koopman estimation that was not present for the UAV system, which could indicate numerical instability or insufficient model order.

From the simulations above, it is trivial which of the methods provided the better outcome. The Autokoopman algorithm's approximation of the dynamics was much more accurate than for the PySINDy and seems to better deal with high-dimensional complex systems and flexible structures. Then, regarding the system used, due to its lesser complexity, the UAV obtained results that were slightly more exact than the ones obtained from the flexible bar, despite both these systems performing well with the Autokoopman algorithm. Seeing as this result matches this thesis scope of obstacle detection during sensor failure, the remaining work will be accounted for only the UAV dynamical system.

3

Reachability Analysis for Obstacle Detection

Contents

3.1 Background

3.1.1 Obstacle Avoidance Control

In the scope of this thesis, the problem at hand consists of obstacle detection using convex sets that represent the state's uncertainties. However, in a similar scope there are numerous methods to consider regarding obstacle avoidance algorithms for UAVs. Most methods consider variations of path re-planning or vision-based control. In [28], a vision-based approach to obstacle detection is used to design control laws to guide the UAV. In [29], three different path re-planning techniques are considered and selected through machine learning while [30] implements a sampling-based path planning. In [31], another solution is proposed regarding obstacle avoidance using a Nonlinear Model Predictive Control (NMPC) optimization technique.

3.1.1.A Vision-Based Control

As per [28], vision-based control is a different approach to obstacle detection and avoidance for an autonomous vehicle, such as a UAV, using visual information, i.e. video data, gathered through a camera or a motion sensor. By employing computer vision techniques, it is possible to estimate motion fields, redefine navigation parameters, and to determine object ranges. This technique allows to make decisions based on the captured images and the design of control and guidance laws to navigate the UAV between waypoints. This approach is particularly useful when there is insufficient data from other sensors.

Vision-based methods can be applied for obstacle avoidance as well as accurate motion estimation and reliable guidance for aircrafts. However, some of these processes can be computationally intensive, especially for real-time applications on micro-UAV platforms with limited onboard computation capability.

3.1.1.B Strategies for Path Re-planning

UAV path planning involves the pre-flight formulation of an optimal reference trajectory that satisfies constraints based on environmental information and mission requirements. During flight, in the detection of an obstacle, it might be necessary for the UAV to re-plan its original trajectory in order to avoid collision [32]. In the paper [29], three strategies for path re-planning regarding obstacle avoidance are presented,varying time requirements and plan quality.

The first one considers a full path re-planning from the next waypoint to the goal. This method allows for the most freedom but it is the slowest one to complete. Partial re-replanning begins only from the waypoing immediately preceding the new obstacle, it is the best balance between time and quality. Finally, the plan repair method is done only for colliding segments. While it is the fastest option, it is the

lowest plan quality.

Using machine learning techniques like Support Vector Machines (SVMs), it is possible to predict re-planning time and quality based on features like initial plan properties and obstacle information, in order to select one of the three methods.

This method explores the different ways to replan the UAV's path after obstacle detection, but it lacks development on the path planning implementation. In [30], the design and implementation of sampling-based path planning methods for a UAV to avoid collision are presented, based on the Closed-Loop Rapidly-Exploring Random Tree (RRT) algorithm.

This method consists in an initial waypoint sampling in the state space of the UAV and a following simulation of the trajectory using the UAV's closed-loop system, which consists of a trajectory-following controller and the vehicle's model. The obtained trajectories that induce no collision with obstacles are then added to a graph of candidate paths. Afterward, the selection of the path is based on the shortest path connecting the start and the goal in the graph.

In this paper, three variations of this method are explored: simplification of trajectory generation strategy, utilization of intermediate waypoints and collision prediction using reachable sets. The latter is an application within the scope of this thesis. In that case, instead of checking if a simulated trajectory will collide with an obstacle whose position is predicted by the linear motion assumption, the planner checks if a simulated trajectory intersects with the reachable set of a moving obstacle, which corresponds to the region that the obstacle could reach at any time.

3.1.2 Model Predictive Control

As stated in [33], Model Predictive Control (MPC) works as a family of controllers in which there is a direct use of an explicit and separately identifiable model. It is explored for both academic and industrial applications as its designed to obtain high performance control systems capable of operating without expert intervention for long periods of time. This method has branched out into numerous other techniques such as Dynamic Matrix Control and Model Algorithmic Control and has been adapted for obstacle avoidance.

MPC uses a mathematical model of a process, recording its behavior through past and present inputs and outputs. During each step, a sequence of future controls is calculated using an optimization of the performance criteria that includes the minimization of the difference between predicted and desired outputs. This optimization occurs over a moving time horizon since the first control action from the optimized sequence is implemented in each step. This process repeats over each step while the horizon moves forward [33].

MPC offers advantages over others like LQR controllers, since it optimizes a finite-time horizon, considering both the current and future timeslots and PID controllers, which lack predictive ability [34].

The paper [31] addresses the problem of motion planning regarding obstacles for aerial platforms under external disturbances and model uncertainties. The technique proposed is the aforementioned MPC but adapted for nonlinear systems with uncertainties on the planned path. It makes use of high-confidence ellipsoids in pose space to propagate the parameter's uncertainty and the initial state. It also uses an online model identification, which allows for adaptation to changing conditions. Results showed that the UAV presented a robust obstacle avoidance behaviour, even under external disturbances.

3.1.3 Differential flatness

Differential flatness is a property used in nonlinear control to describe the structure of a certain dynamic system. This occurs when a set of outputs (flat outputs) exist in a way that all the states and inputs of the system can be expressed as a function of these plus a finite number of their derivatives [35]. This makes it so that the system's dynamics can be captured and expressed in simpler terms.

One can verify if a system is differentially flat by testing out potential flat outputs that are functions of the state and input variables, in the case they can be expressed as algebraic functions, ensuring the number of flat outputs matches the number of inputs [35].

Identifying differential fatness is a way to simplify the control design of a system, due to the dynamics of the flat outputs being linear and decoupled, as well as trajectory generation since it makes it possible to plan the trajectory of a system by specifying the desired behavior of the flat outputs and computing the corresponding state and input trajectories [36]. This concept has been applied in both mechanical and robotic systems, including aerospace vehicles [36].

3.2 Constrained Zonotopes

This section aims to explore the theoretical framework of constrained zonotopes and their potential applications in set-based estimation and fault detection within dynamical systems, using the work done in [37]. Further, the aim is to investigate their utilization for propagating system dynamics within the Koopman domain. By delving into the theoretical underpinnings of constrained zonotopes, the goal is to establish a robust foundation for their practical implementation.

A zonotope is the result of a Minkowski sum of line segments defined by each generator vector so that it can be defined as:

$$\mathcal{Z} = \{G\xi + c : \|\xi\|_\infty \leq 1\} \tag{3.1}$$

The columns of the matrix G correspond to the vectors v_i (generators) and the vector c is called the center and is the starting point for these. Consequentially, zonotopes are convex polytopes that are

centrally symmetric, which means that every chord through c is bisected by c. This form corresponds to the generator-representation (G-rep) of zonotopes. There are other methods to represent a zonotope such as the halfspace-representation (H-rep), which is used for various bounded convex polytopes:

$$\mathcal{P} = \{z \in \mathbb{R}^n : Hz \leq q\}. \tag{3.2}$$

A constrained zonotope is a zonotope that allows linear equality constraints on ξ. Due to this, constrained zonotopes provide the computational advantages of zonotopes while enabling exact computations of a much wider class of sets [38]. A set $\mathcal{Z} \subset \mathbb{R}^n$ is a constrained zonotope if there exists $(G, c, E, b) \in \mathbb{R}^{n\times ng} \times \mathbb{R}^n \times \mathbb{R}^{nc\times ng} \times \mathbb{R}^{nc}$ such that

$$\mathcal{Z} = \{G\xi + c : \|\xi\|_\infty \leq 1, E\xi = b\}. \tag{3.3}$$

It is now referred to the constrained generator representation (CG-rep) and introduced the shorthand $\mathcal{Z} = \{G, c, E, b\} \subset \mathbb{R}^n$. For the further propagation in Koopman domain and set-based estimation, there are three essential operations:

$$R\mathcal{Z} + r = (RG_z, Rc_z + r, E_z, b_z), \tag{3.4}$$

$$\mathcal{Z} \oplus \mathcal{W} = \left(\begin{bmatrix} G_z & G_w \end{bmatrix}, c_z + c_w, \begin{bmatrix} E_z & 0 \\ 0 & E_w \end{bmatrix}, \begin{bmatrix} b_z \\ b_w \end{bmatrix} \right), \tag{3.5}$$

$$\mathcal{Z} \cap_R \mathcal{W} = \left(\begin{bmatrix} G_z & 0 \end{bmatrix}, c_z, \begin{bmatrix} E_z & 0 \\ 0 & E_w \\ RG_z & -G_w \end{bmatrix}, \begin{bmatrix} b_z \\ b_w \\ c_w - Rc - z \end{bmatrix} \right). \tag{3.6}$$

3.2.1 Constraint Reduction

It was also necessary to choose a strategy for constraint reduction, as the zonotope resulting from the transformation to Koopman domain does not have mathematically compatible matrices E and G. Using the proposition in [39], the strategy used was a heuristic called partial solve dualization. To solve one of the constraint equations, the first equation is solved for a single ξ_j:

$$\xi_j = e_{1j}^{-1} b_j - e_{1j}^{-1} \sum_{k\neq j} e_{1k}\xi_k. \tag{3.7}$$

This equation is then used to eliminate ξ_j from the remaining constraints and the equations $z = G\xi + c$. The first constraint is subsequently removed. By choosing

$$\Lambda_G \equiv G\zeta_{j1}e_{1j}^{-1}, \quad \Lambda_E \equiv E\zeta_{j1}e_{1j}^{-1}, \tag{3.8}$$

where $\zeta_{j1} \in \mathbb{R}^{ng \times nc}$ is zero except for a one in the $(j, 1)$ position, and evaluating, $\mathcal{Z} = \{\tilde{G}, \tilde{c}, \tilde{E}, \tilde{b}\}$, it follows that $\tilde{G}$ and $\tilde{E}$ have identically zero j-th columns due to the elimination of ξ_j, and $[\tilde{E}|\tilde{b}]$ has an identically zero first row, resulting from substituting the constraint $e_1^\intercal \xi = b_1$ with the trivial constraint $0^\intercal \xi = 0$. Removing these columns and rows, one constraint and one generator are eliminated.

3.2.2 Zonotope Splitting

After detecting the intersection of the propagated zonotope with the obstacle in the Koopman domain, it is possible to divide the zonotope into smaller sections to discover the section that causes the intersection and remove it from the initial zonotope. The goal was for the split section of the zonotope to also be a zonotope in the exact dimensions as the original. Below is the method considered based on [40].

A zonotope $\mathcal{Z}$ can be divided into two smaller zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$ by splitting the j-th generator of $\mathcal{Z}$.

Consider a zonotope $\mathcal{Z} = (c, g(1), g(2), \ldots, g(p))$. It can be split into two zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$ such that $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$ and $\mathcal{Z}_1 \cap \mathcal{Z}_2 = \mathcal{Z}^*$, where:

$$\mathcal{Z}_1 = \left(c - \frac{1}{2}g(j), g(1), g(2), \ldots, g(j-1), \frac{1}{2}g(j), g(j+1), \ldots, g(p)\right), \tag{3.9}$$

$$\mathcal{Z}_2 = \left(c + \frac{1}{2}g(j), g(1), g(2), \ldots, g(j-1), \frac{1}{2}g(j), g(j+1), \ldots, g(p)\right), \tag{3.10}$$

$$\mathcal{Z}^* = (c, g(1), g(2), \ldots, g(j-1), g(j+1), \ldots, g(p)). \tag{3.11}$$

Here, $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are formed by adjusting the center c by $\pm\frac{1}{2}g(j)$ and replacing the j-th generator $g(j)$ with $\frac{1}{2}g(j)$. The intersection $\mathcal{Z}^*$ is the zonotope without the j-th generator.

If the initial generator selected is $j = 1$ and then the resulting zonotopes are repeatedly split by the next generator $(j + 1)$, by the end, the initial zonotope will be divided into 2^n equally sized zonotopes.

3.2.3 Set-based State Estimation

In [39], the advantages of using constrained zonotope computations for set-based state estimation problems for discrete-time linear systems are presented, in comparison to existing classes. These are able to describe arbitrary convex polytopes, in the case of a non-limited complexity of the representation; they allow simple standard set operations like intersections and provide effective techniques to reduce the complexity of a set, which enables a tunable tradeoff between efficiency and accuracy.

Considering:

$$x_k = Ax_{k-1} + B_w w_{k-1},$$
$$y_k = Cx_k + D_v v_k,$$

with state $x_k \in \mathbb{R}^{n_x}$, output $y_k \in \mathbb{R}^{n_y}$, disturbance $w_{k-1} \in \mathbb{R}^{n_w}$, measurement error $v_k \in \mathbb{R}^{n_v}$ and omitted input u_k. Assuming that $x_0 \in \mathcal{X}_0$ and $(w_k, v_k) \in \mathcal{W} \times \mathcal{V}$ for all $k \in \mathbb{N}$, with $\mathcal{X}_0$, $\mathcal{W}$, and $\mathcal{V}$ compact, a set-valued estimate of x_k can be exactly computed using the recursive formula

$$\mathcal{X}_k = (A\mathcal{X}_{k-1} + B_w\mathcal{W}) \cap_C (y_k - D_v\mathcal{V}). \tag{3.12}$$

An enclosure of $\mathcal{X}_k$ can be computed through an outer approximation

$$\mathcal{O}_k \supseteq (A\mathcal{O}_{k-1} + B_w\mathcal{W}) \cap_C (y_k - D_v\mathcal{V}), \tag{3.13}$$

$$\mathcal{O}_0 \supseteq \mathcal{X}_0, \tag{3.14}$$

whenever the operations are not exact. This problem has been solved through many methods, starting with ellipsoidal computations and later developing more accurate but less efficient algorithms. In this dissertation, the algorithm primarily derives from the advantages of using constrained zonotope computations.

On the other hand, [41] provides a different set-based method that computes exact reachable sets for neural feedback systems for linear and nonlinear models. Similarly to the scope of this thesis, the goal is the avoidance of unsafe regions represented as constrained zonotopes, using linear program-based sufficient conditions. A neural feedback system is modeled as a discrete-time system with a feedback loop involving a feedforward neural network. The reachable set at each step is computed using the controller's nonlinearities and set-based operations. It also provides an over-approximated alternative since exact reachability can be computationally expensive, using a relaxation of constraints to reduce complexity.

3.3 Problem Statement

This dissertation addresses the issue of ensuring the safety of an UAV in the presence of uncertainties using reachability analysis. The primary goal is to verify if these uncertainties in the UAV's state, particularly due to sensor measurement errors, influence the accuracy of obstacle detection and whether set-based estimation methods allow for the prediction of the UAV's collision with an object in its path.

For this work, the UAV, following the dynamics $\dot{x} = f(x)$, is represented by a high-dimensional

zonotope, $\mathcal{Z}$, to reflect the UAV's full state vector, encapsulating said measurement uncertainties but excluding the UAV's physical dimensions.

The work is developed in order to assess if the UAV's trajectory can be predicted with sufficient accuracy to detect intersection between the estimation and obstacle sets in the face of sensor failure. This is done through calculating a state estimation zonotope, $\hat{\mathcal{Z}}$, from the UAV zonotope, using two different methods: one based on the existing method of applying the Lagrange remainder error matrix to set operations and another on the Koopman operator theory, previously explored. The obstacles to consider when performing the safety verification will also be represented by zonotopes, $\mathcal{O}$, with dimensions stipulated later.

The estimation set can then be calculated throughout the UAV's whole trajectory, making it possible to verify the existence of zonotope intersection between the estimation $\hat{\mathcal{Z}}$ and the obstacle $\mathcal{O}$. This enables proactive detection of potential collisions on the UAV's planned path, allowing to have access to the safety status of the UAV even during sensor fault.

3.4 State of the Art

A method has already been developed that can be applied to obstacle detection through constraint zonotopes, using the work done in [42] regarding the safety verification of dynamical systems using reachability analysis. This consists of initially approximating a nonlinear system with a linear model around a chosen operating point, using local linearization by a Taylor series. This linearization will inevitably be an approximation that inherently introduces errors. To account for these linearization errors, they are determined in an over-approximate manner and incorporated as additional uncertain inputs using the Lagrange remainder, ensuring that the resulting computations remain over-approximate and conservative. This allows for the calculation of the reachable set and the measurement of the probability of reaching an unsafe set.

By using constraint zonotopes as the state vector and the Lagrange remainder matrix, it is possible to use this method to estimate the state and further propagate the system. As mentioned, the initial zonotope will represent the uncertainties of the GPS position. In the case of sensor failure in the GPS, by applying the expressions developed in this method, one is able to obtain the state zonotope propagated in the near future using the state estimation from the linearization.

The zonotope will represent an over-approximation of the possible end positions of the UAV, since it takes into account the propagation of the initial uncertainties and the linearization errors. This helps to verify if there is the possibility of collision with an obstacle using zonotope intersection. During this thesis, this method will be referred to by the Lagrange remainder method.

3.4.1 Linearization

Regarding this method, the focus was on general nonlinear continuous systems characterized by uncertain parameters and exhibiting Lipschitz continuity. Similar to the approach taken for linear systems, the initial state $x(0)$ is assumed to belong to a set $\mathcal{X}_0 \subset \mathbb{R}^n$. The system dynamics are influenced by a set of uncertain but constant model parameters ρ_i, which are bounded within specified intervals ($\rho_i \in I$). Consequently, the parameter vector ρ is confined to a multidimensional interval $\mathcal{P} \subset I^p$, where p denotes the number of parameters. The system input u is drawn from a set $\mathcal{U} \subset \mathbb{R}^m$. All of the work regarding this linearization method is based on the work in [42].

The evolution of the state x is governed by the following differential equation:

$$\dot{x} = f(x, u, \rho), \tag{3.15}$$

$$x(0) \in \mathcal{X}_0 \subset \mathbb{R}^n, \quad \rho \in \mathcal{P} \subset I^p, \quad u \in \mathcal{U} \subset \mathbb{R}^m. \tag{3.16}$$

The initial local linearization requires an initial concatenation of the state and input vectors:

$$z = \begin{pmatrix} x \\ u \end{pmatrix}. \tag{3.17}$$

Through an infinite Taylor series, it is possible over-approximate the system by a first-order Taylor series and its Lagrange remainder:

$$\dot{x}_i \in f_i(z^*, \rho) + \underbrace{\left.\frac{\partial f_i(z,\rho)}{\partial z}\right|_{z=z^*} (z - z^*)}_{\text{1st order Taylor series}} + \underbrace{\frac{1}{2}(z - z^*)^\intercal \frac{\partial^2 f_i(\xi,\rho)}{\partial z^2}(z - z^*)}_{\text{Lagrange remainder } L_i}. \tag{3.18}$$

For fixed z and z^*, the Lagrange remainder L can take any value resulting from $\xi \in \{z^* + \beta(z - z^*) \mid \beta \in [0, 1]\}$. The variable z^* corresponds to the operating point used as a reference in the linearization. To revert to the standard notation of the linearized system, the vector z is decomposed back into the state vector x and the input vector u:

$$\begin{aligned} \dot{x} &\in f(z^*, \rho) + \left.\frac{\partial f(z,\rho)}{\partial z}\right|_{z=z^*} (z - z^*) + L \\ &= f(x^*, u^*, \rho) + \underbrace{\left.\frac{\partial f(x,u,\rho)}{\partial x}\right|_{x=x^*,u=u^*} (x - x^*)}_{A\Delta x} + \underbrace{\left.\frac{\partial f(x,u,\rho)}{\partial u}\right|_{x=x^*,u=u^*} (u - u^*)}_{B\Delta u} + L, \end{aligned} \tag{3.19}$$

where $\Delta x = x - x^*$ and $\Delta u = u - u^*$. In the absence of uncertain parameters ρ, matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ contain real numbers.

3.4.2 Over-approximation of the Set of Linearization Errors

When the system includes uncertain parameters, one can choose to over-approximate the uncertain system and input matrices using matrix zonotopes, which will be carried out in the development of this thesis. The Lagrange remainder is determined after defining:

$$J_i(\xi, \rho) := \frac{\partial^2 f_i(\xi, \rho)}{\partial z^2}, \tag{3.20}$$

where i refers to the i-th coordinate of f, one can write the Lagrange remainder in Equation as

$$L_i = \frac{1}{2}(z - z^*)^\intercal J_i(\xi, \rho)(z - z^*), \tag{3.21}$$

$$\xi(z, z^*) \in \{z^* + \beta(z - z^*) \mid \beta \in [0, 1]\}. \tag{3.22}$$

This can be over-approximated for $z \in \hat{\mathcal{Z}}$, where $\hat{\mathcal{Z}}$ is a zonotope defined as $\hat{\mathcal{Z}} = (c, g^{(1)}, \ldots, g^{(e)})$. The over-approximation is obtained by the following computations:

$$|L_i| \subseteq [0, l_i] \tag{3.23}$$

$$l_i = \frac{1}{2}\gamma^\intercal \max(|J_i(\xi(z, z^*), \rho)|)\gamma, \quad z \in \hat{\mathcal{Z}}, \quad \rho \in \mathcal{P}, \tag{3.24}$$

$$\gamma = |c - z^*| + \sum_{i=1}^{e} |g^{(i)}|, \tag{3.25}$$

where c is the center and $g^{(i)}$ are the generators of the zonotope $\hat{\mathcal{Z}}$. The value for $\max(|J_i(\xi(z, z^*), \rho)|)$ can be solved through an optimization problem.

This can be applied to a zonotope representing the uncertainty of the states ($\mathcal{Z}_k = (c, g^{(1)}, \ldots, g^{(e)})$) by linearizing around the center (c) of the initial zonotope and exchanging all the scalar operations for set operations, i.e. the sums for Minkowski sums. The bounding vector l of the absolute value of the Lagrange remainder is minimized by choosing $z^* = c$ as the linearization point.

3.4.3 Restriction of the Linearization Error

The Lagrange remainder presents a problem: its sensitivity to the size of the previously estimated state set. The larger the set becomes, the larger the linearization error set is bound to be, which in turn again enlarges the estimated state set.

This can be helped by restricting the linearization error R_{err} to a multidimensional interval $\bar{R}_{\text{err}}$, as done in [42]. The rate of growth of the admissible expansion of $\bar{R}_{\text{err}}$ is set by the user-chosen expansion

vector $\gamma \in \mathbb{R}^n$. This method also enables the discretization of the linearization error matrix L, which until now was only available in continuous form. The discretization would inevitably be necessary due to the operations with convex sets.

$$\bar{R}_{\text{err}}(\Delta t) := [-\gamma \cdot \Delta t, \gamma \cdot \Delta t], \tag{3.26}$$

where Δt is the step time interval. Assuming that the linearization error is constant for the time step $[k\Delta t, (k+1)\Delta t]$ and that the system matrix has no uncertainties, the following relation between the admissible reachable set and the admissible linearization error $\bar{L} = [-\bar{l}, \bar{l}]$ exists:

$$\bar{R}_{\text{err}}(\Delta t) = [-\gamma \cdot \Delta t, \gamma \cdot \Delta t] = A^{-1}(e^{A\Delta t} - I)[-\bar{l}, \bar{l}], \tag{3.27}$$

where the interval $[-\bar{l}, \bar{l}]$ will be represented by the zonotope L obtained from 3.21, and A is the system's discrete matrix.

3.4.4 Discretization

For the implementation of this method, it was necessary the linearization and later discretization of the real ODEs, using the work developed in 3.4.1 without the linearization error matrix L as a simple linearization method.

As mentioned above, for the training data in AutoKoopman, the discretization of the dynamics was obtained by solving the system of ordinary differential equations for each step. However, this method was not possible for the Lagrange remainder operation due to the work with constraint zonotopes requiring a discrete system. A simple discretization with the matrices A and B resulting from the linearization was also impossible due to the constant term $f(x^*, u^*, \rho)$.

A different method was employed by concatenating the variable $f(z^*, \rho)$ as an input. This way, the matrix B and the vector u are augmented as such:

$$u_{\text{aug}} = \begin{bmatrix} u \\ 1 \end{bmatrix} \quad B_{\text{aug}} = \begin{bmatrix} B & f(x^*, u^*, \rho) \end{bmatrix} \tag{3.28}$$

Through this, it was then possible to apply:

$$e^{\begin{bmatrix} A & B_{\text{aug}} \\ 0 & 0 \end{bmatrix} \Delta t} = \begin{bmatrix} A_d & B_d \\ 0 & I \end{bmatrix}, \tag{3.29}$$

where A_d and B_d are the discrete state space matrices and Δt is the time step used.

The initial zonotope is then propagated using a linear map with the matrices A_d and B_d. At each step, a new zonotope L is made, representing the linearization error matrix. This zonotope is a 12-dimension

zonotope with the center having every entry at zero, and the generator matrix being a diagonal matrix with l_i as its entries (3.24). This zonotope is then multiplied by $A^{-1}(e^{A\Delta t} - I)$ through another linear map. The next step's state zonotope will then be the result of the Minskowski sum of the current propagated zonotope and the matrix L.

3.4.5 Zonotope Splitting

The existing literature regarding the Lagrange remainder's method for safety verification using set-based forward propagation is rather conservative as it uses many over-approximations. Ideally, one could have an exact estimation of the propagated state by sampling the initial zonotope using a zero-spaced grid that would provide an infinite amount of points and propagating them using the UAV's known dynamics. In this case, the linearization would not be necessary as it is not a set-based operation, and no approximations would be made in the state estimation. The convex set uniting the end points would be a smaller set, inserted in the one resulting from the Lagrange remainder. However, this method would require a lot of computation power.

The proposed alternative to this problem consists of making use of zonotope splitting. While taking an infinite amount of points within the zonotope is computationally expensive, it is possible to split the initial zonotope into n subdivisions and simultaneously propagate those instead. The final result would then be n convex sets, which together would be smaller and inserted in the originally propagated set. The larger the value of n, the better the approximation is to the ideal method. Consequentially, the application of the Lagrange remainder method will initially consist of its original usage. Still, it will then evolve to the propagation of increasingly smaller subdivisions to achieve more accurate results.

3.5 Proposed Method

The proposed method is similar to the Lagrange remainder method in the way that it also makes use of a linearized version of the real system dynamics to propagate the UAV's initial uncertainty zonotope and get an estimation of the future state. This uncertainty again corresponds to the GPS measurement error. The end goal also consisted in using the intersection of two zonotopes for obstacle detection, one regarding the propagated zonotope and another for a hypothetical object in the quadrotor's path.

Unlike the Lagrange remainder method, the linearization is not done in the 12-dimensional space. Instead, the Autokoopman algorithm is applied, and a coordinate transformation is done on the initial uncertainty zonotope. Then, the matrices A and B resulting from the Autokoopman linearization are used to propagate this zonotope in the Koopman domain, allowing for an over-approximated estimation of the state, a few steps in advance.

The obstacle zonotope will also be represented in the Koopman Domain. The verification of intersection between these two zonotopes occurs in this high-dimensional space, using the theory in (3.6). If the intersection zonotope is null, then it is guaranteed that the UAV will not collide with the obstacle in the near future, even with sensor failure. As previously, if the intersection zonotope exists, it is possible to divide the initial zonotope into smaller subdivisions to verify if this detected collision is simply due to the over-approximations made during the zonotope's propagation or if it is actually a possible outcome due to the UAV's predicted trajectory.

For the development of this method, functions from the Continuous Reachability Analyzer (CORA) files [43] were used. This is a toolbox that integrates various vector and matrix set representations and operations on these set representations as well as reachability algorithms of various dynamic system classes. It is designed to exchange set representations without having to modify the code for reachability analysis.

3.5.1 Propagation in the Koopman Domain

The following development will require a verification of the accuracy of the linearization of the system so initially it was tested how would the system propagate in the Koopman domain. The first step was to have an initial unit hyper-cube constraint zonotope already in the Koopman domain, so as to avoid nonlinear transformations, and to propagate it using the previously obtained linear Koopman model. The initial zonotope's coordinates would be randomly sampled 500 times and transformed into the $\mathbb{R}^{12}$ domain to simultaneously advance them using the real system's dynamics for the same number of steps as the zonotope. Then these were to be transformed once more into the Koopman domain and checked if they remained inside the zonotope. This would ensure that the Koopman estimation of the state encompassed the state propagated using the real ODEs. It also provides information over the effect of the matrix A's S_{max} value in the expansion of the zonotope while propagating.

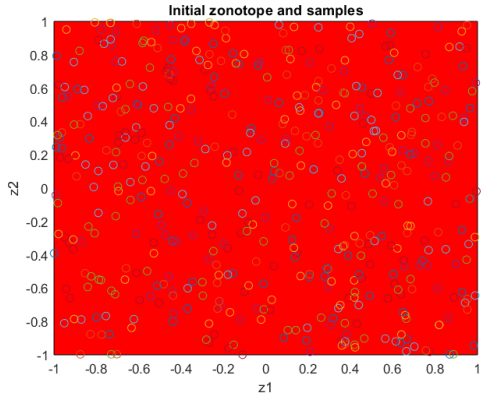


Figure 3.1: Sampled Initial Hyper-cube in the Koopman Domain

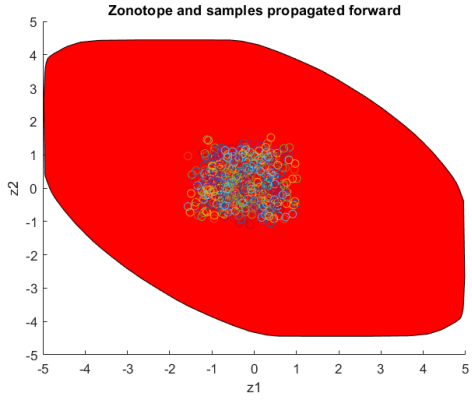


Figure 3.2: Sampled Propagated Hyper-cube in the Koopman Domain

As expected, the Koopman estimation does guarantee a semi-accurate propagation of the states compared to the real system's dynamics, as all the samples remained inside the initial zonotope. However, due to the matrix A's S_{max} value being larger than 1 and the approximations made when transforming the set to the Koopman domain, the zonotope propagated using the Autokoopman's matrices was greatly enlarged compared to where the samples are gathered. Consequentially, the Koopman method shows the same limitations as mentioned for the Lagrange remainder method regarding over-approximation errors.

While this allows for the continuation of this method's development, the remaining work must be done for a small number of steps to avoid a very large over-approximation, since this would defeat the purpose of the obstacle detection, and it reinforces the need to divide the initial zonotope into smaller sets to minimize these errors. The evaluation of the results should also consider that this estimation only encompasses the real values instead of providing an accurate representation.

3.5.2 Set-Based Coordinate Transformation

After the verification of the accuracy of Koopman domain propagation, it is necessary to consider the coordinate transformation using convex sets instead of vectors, since the operations in (2.30) are not compatible with the zonotope notation. The initial normalization of the states and the operation $\omega \cdot x_0 + \alpha^\intercal$ must now be done through a linear map (3.4). On the other hand, the cosine calculation cannot be performed exactly with convex sets and must be done through an approximation. An algorithm was developed that constrained the initial zonotope with cuts along the cosine function of each variable in the state.

The algorithm consists of employing an optimization problem for each variable in the state that finds how many minimum and maximum values there are in a certain interval. It then calculates their cosine and adds constraints to the zonotope that cut along these points. This is a considerable over-approximation, as it does not provide the exact cosine zonotope but rather an enclosure of it. This over-approximation is responsible for most of the errors in Koopman domain zonotopes.

This function returns a zonotope with a G matrix with double the columns of the original, with the zonotope corresponding to the cosine accounting for the first half of them. As mentioned, the obtained zonotope will not have constraint matrices E and b aligned with its dimension, so it is necessary to implement a constraint reduction. The method for constraint reduction explained in 3.2.1 was initially experimented but required a lot of computational power, due to removing constraints one at a time. A different method was then developed and is explained in the section below.

Finally, the state in (2.30) is achieved by appending the original state zonotope to this one. This makes the final constraint zonotope represented in a 162-dimension space, with the first 12 entries matching the zonotope in the original space. This transformation applies multiple approximations, more

specifically in the cosine calculation and the constraint reduction. This will affect the results of the set-based operations in the Koopman Domain. Since the zonotope won't exactly be in the same coordinates as the output of the AutoKoopman algorithm, the matrices A and B will not be tailored for the propagation of this zonotope.

3.5.3 Alternative Method for Constrain Reduction

To avoid the usage of the existing method for constraint reduction, another one was employed that consisted in solving an optimization problem. This method allowed to construct a polytope that approximated the real zonotope by taking random vectors and using them as the rows in the matrix H in the halfspace-representation for convex zonotopes, once these vectors intersected with the limits of the constraint zonotope. The resulting polytope would then be transformed back into a constrained zonotope, now with fewer constraints and with a matrix E that is mathematically compatible with the matrix G.

From [39], every constrained zonotope is a convex polytope. Let $\mathcal{P} = \{z \mid Hz \leq q\}$ be a convex polytope. By compactness, one may choose $\mathcal{Z}_0 = \{G, c\} \subset \mathbb{R}^n$ and $\sigma \in \mathbb{R}^n$ such that $\mathcal{P} \subseteq \mathcal{Z}_0$ and $Hz \in [\sigma, q]$, for all $z \in \mathcal{P}$ (the use of $\subseteq$ includes the possibility of equality throughout). Then $\mathcal{P} = \{z \in \mathcal{Z}_0 \mid Hz \in [\sigma, q]\}$. The interval $[\sigma, q]$ can be written in G-representation as

$$\left\{ \operatorname{diag}\left(\frac{q-\sigma}{2}\right), \frac{q+\sigma}{2} \right\}, \tag{3.30}$$

resulting in:

$$\mathcal{P} = \left\{ [G\ 0], c, \left[HG\ \operatorname{diag}\left(\frac{\sigma - q}{2}\right) \right], \frac{q+\sigma}{2} - Hc \right\}. \tag{3.31}$$

The chosen vector is a random unit vector within the 12-dimension sphere. The zonotope can be compiled into two outputs:F, which is a Yalmip constraint set and x, a sdpvar representing a point in $\mathcal{Z}$. The constraint set F is the constraint for the optimization problem, and the objective is to minimize the product of a vector v with a point in $\mathcal{Z}$ - x ($v^\intercal \cdot x$). The symmetric of this vector can then be set as the rows for H, while q is the symmetric of the value resulting from the optimization problem. The vector σ was chosen as the result of the optimization problem for the symmetric of v.

Solving this problem for n vectors, for a great value of n, the constraints obtained will eventually approximate the polytope to the original zonotope. Applying these methods to set-based operations allowed for the continuation of the development regarding sets in the Koopman domain without conflicting values of the constraints.

The algorithm for this method regarding said zonotope was applied for two different values of n, to show the effect this value has on the accuracy of the final zonotope. This method was tested for the following zonotope in the paper [39]:

$$\mathcal{Z} = \left\{ \begin{bmatrix} 1 & 0 & 1 \\ 1 & 2 & -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 & 1 & -1 \end{bmatrix}, 2 \right\}.$$

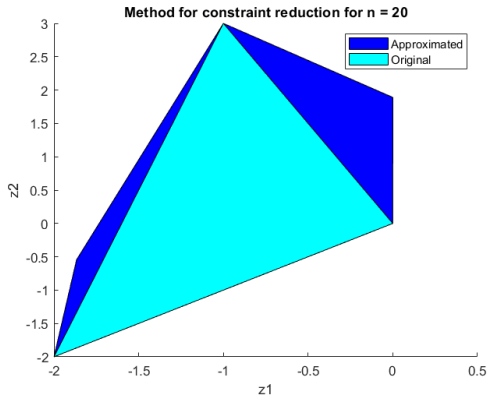


Figure 3.3: Constrain Reduction Method for n = 20

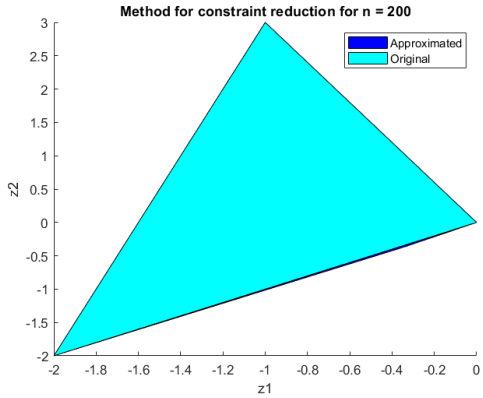


Figure 3.4: Constrain Reduction Method for $n = 200$

For the remaining work, the value chosen for n was 200, as it provided a reasonably exact result without requiring much computational power. However, this approximation should be taken into consideration when accessing this method's results.

3.6 Simulations

This section presents a series of simulations designed to evaluate the effectiveness of reachability analysis for obstacle detection using zonotopes. These simulations will also allow to compare the performance of both presented methods.

For the simulations, the UAV state zonotope is constructed in a 12-dimensional space to reflect the UAV's full state vector. It will have the UAV's GPS position at $t = 0$ s as the center. At this time, the UAV will be at the XYZ origin with all its velocities and Euler angles at zero. The first three generator vectors will correspond to the GPS measurement noise, representing the state variables p_x, p_y and p_z. In this case, it was established a value of ± 5 cm for all. Is is important to note that these values are mainly symbolic for the simulations and not an accurate representation of GPS measurement noise as this corresponds to a very accurate sensor in ideal conditions. The generators for the remaining states are set at 0 since the GPS does not measure those. For the Koopman method, this zonotope will then be transformed into the Koopman coordinate space of 162 dimensions.

The trajectory for the UAV is simple and consists in the path from position $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal$ m to $\begin{bmatrix} 3 & 0 & 3 \end{bmatrix}^\intercal$ m , using a LQR controller. In ideal conditions with sensor functionality, the positional uncertainties of

the UAV would not expand throughout the simulation, and the final zonotope would be equal in size to the initial one but with a translated center to the desired position.

There will be two 3-dimensional obstacle zonotopes to consider during the simulations. The first obstacle is an unknown $0.4\ \mathrm{m} \times 10\ \mathrm{m} \times 1.7\ \mathrm{m}$ object. Both obstacles have large dimensions in the y axis to serve as a representation of a wall. It is positioned near but outside the UAV's trajectory, with its center in the coordinates $\begin{bmatrix}1.2 & 0 & 0.85\end{bmatrix}^\intercal\ \mathrm{m}$, to be placed conveniently near the UAV's path. The center's p_z variable should always be half of the object's height, to place it on $p_z = 0\ \mathrm{m}$, i.e. the "ground". This way, without sensor failure and consequential estimation errors or disturbances in its path, the UAV should not intersect the obstacle. The second obstacle is of the same dimensions and center, except with a height of $1.1\ \mathrm{m}$, which goes through the UAV's planned path, so the intersection is guaranteed to happen even without sensor failure or disturbances. This serves as a comparison to the former. For the Koopman simulation, the obstacles need to be altered for a 12-dimensional representation, with the additional nine generators at zero, since this is a requirement to transform them into the same type of Koopman coordinates as the UAV's system. Without this, the intersection of the propagated zonotope and the obstacle zonotopes in the Koopman domain would not be possible.

For all the simulations regarding the Lagrange remainder, the LQR controller for the propagation will be designed for the path described. The matrices for Q and R, while varying in dimension for each simulation depending on the domain, were always similar to obtain fair results: $Q = diag\begin{bmatrix}10 & 1 & 10 & 1 \dots 1\end{bmatrix}$ and $R_{Lagrange} = I_{12}$ or $R_{Koopman} = I_{162}$.

In the Koopman domain, however, the LQR has to be adjusted to account for the uncontrollable states introduced by the Autokoopman toolbox. Therefore, only the controllable parts of matrices A and B can used to compute the gain vector, while the uncontrollable states decay at the rate of the slowest eigenvalue. Prefacing the simulations, an assessment of the controllers' design will be made to account for these differences.

All simulations are conducted over 100 time steps, with a $0.01\,\mathrm{s}$ interval between each step. Obstacles are represented by red convex sets, while the UAV's zonotope representing the positional uncertainties, based on its trajectory using the actual dynamics without state estimation, appears as a blue zonotope. This blue zonotope maintains a relatively consistent size, corresponding to the measurement error from GPS sensors. Yellow zonotopes represent state estimations, with any intersections between the UAV and obstacles (or their estimations) displayed in magenta. All zonotopes are visualized in a $2D$ projection on the XZ-plane to accommodate their high dimensionality.

The first two simulations represent an ideal scenario where the GPS sensor functions without failure. The UAV's trajectory passes near the first obstacle without intersecting it. In contrast, the second obstacle is set to intersect directly with the UAV's path, thereby validating the set operations' accuracy in both collision and non-collision conditions.

There will be a third simulation using the object for which there is no collision that allows for the calculation of an estimation of the state ten steps ahead throughout the UAV's path. While this does not yet represent the GPS sensor failure, it informs the user of the possible outcome if sensor failure does occur. If there is no intersection of the object with the state estimation, there is the knowledge that if sensor failure occurs, then the UAV will still be safe for at least ten steps. However, state estimation cannot be done regarding the exact last known state value since, in this case, due to the GPS's measurement noise, the last known state value is a zonotope representing its uncertainties, making it an over-approximation.

The fourth simulation considers the situation for which there is sensor failure for ten consecutive steps. In order to achieve this, at $t = 0.8$ s, the simulation will stop taking the GPS's measurements into consideration, representing the GPS sensor failing. This means that the real value of the states is not available to maintain the UAV in its path, and it is now necessary to rely on a state estimation Despite originally, without sensor failure, the UAV having no collision, in this case the controller will only have access to this estimation of the state based on a value that was already over-approximated, which can lead to the misdirection of the UAV. At $t = 0.9$ s, the GPS sensor will be functioning again, and the controller can take the now available information about the positional states to return the UAV to its desired path. Due to this failure, despite there being no intersection in the ideal situation, it is possible to have occurred in this one due to said misdirection.

A fifth simulation will be made, similar to the previous one but without the return of the GPS sensor functionality. This means that the estimation will be calculated for 20 steps, worsening the errors associated with the methods. While this has the disadvantage of possibly returning false positives regarding the intersection, it is over-reliable if it shows that no collision will happen.The final simulation consists of calculating the state estimation using sub-divisions of the initial uncertainty zonotope by splitting it to attempt to minimize said over-approximations in both methods.

These simulations enable a thorough assessment of the reachability analysis for collision prediction. The results indicate whether the proposed methods can reliably forecast collisions in the presence of uncertainties, sensor noise, and varying durations of GPS failure.

3.6.1 Real System Dynamics

To compare the Lagrange remainder and Koopman operator methods for state estimation, an initial simulation of the UAV trajectory was conducted. This simulation used a simplified linearization of the original ODEs, assuming no GPS sensor failure. The linearization was applied to design the LQR controller and propagate the zonotope at each step, and is defined by:

$$\dot{x} = f(x,u), \quad A = \left.\frac{\partial f_i}{\partial x_j}\right|_{(x_e,u_e)}, \quad B = \left.\frac{\partial f_i}{\partial u}\right|_{(x_e,u_e)}. \tag{3.32}$$

The initial zonotope centers on the UAV's state at $t = 0$, with a generator representing GPS measurement noise set to ± 5 cm. Generators for other states are set to zero, as the GPS does not measure them, and initially, the UAV is positioned at the origin (X, Y, Z) with all velocities and Euler angles at zero.

Similarly, an additional zonotope, identical in size to the initial one, represents the GPS measurement at each step. This zonotope remains constant in size throughout the simulation, with fixed generators, and its center reflects the current positional states (p_x, p_y, p_z) of the UAV.

Due to inherent approximations in the linearization of the real system, the state zonotope changes in size as it propagates. However, with the GPS sensor functioning, intersecting the propagated state zonotope with the GPS zonotope at each step simulates positional measurements, allowing the controller to update and proceed to the next step. This simulation was executed in two scenarios: one with an obstacle along the UAV's path resulting in collision, and another with an obstacle outside the trajectory. For computational efficiency, the simulation displays the state zonotope every ten steps.

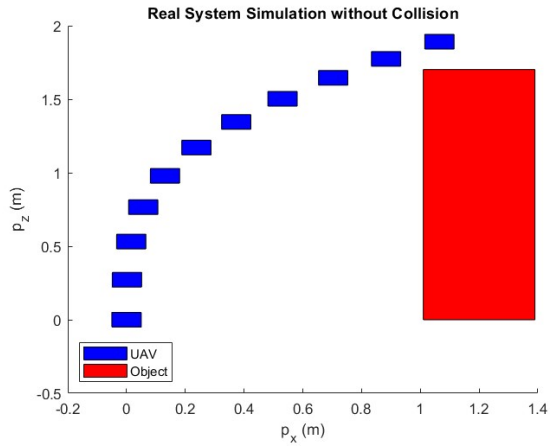


Figure 3.5: Real System Simulation with the Obstacle Outside the Path

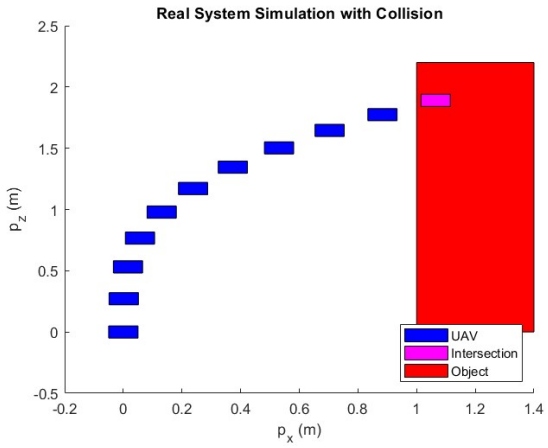


Figure 3.6: Real System Simulation with the Obstacle In the Path

As expected, the UAV followed the planned trajectory, matching predictions for obstacle interaction. The magenta area represents the intersection zonotope between the uncertainty zonotope and the obstacle, indicating a collision in the second simulation before reaching 100 steps. This collision reflects an actual intersection between the obstacle (in red) and the UAV positional representation (in blue). For the following sections, only the non-collision scenario will be analyzed, applying state estimation theory to evaluate safety even under potential sensor failures. The collision scenario is excluded, as the collision outcome is already established.

3.6.2 Controller Design

Before presenting the remaining simulations, it is important to explain the variations in the controller design, as these differences directly affect the results of each simulation. All controllers were implemented considering the zonotopes' center as the state vector, so as to not have the zonotopes' size affecting the response and to facilitate its design.

For the Lagrange remainder method, a simple LQR controller was used, implemented with the *dlqr* function. However, it was necessary to account for the concatenated constant input $f(x^*, u^*, \rho)$ applied during the linearization of the state. To handle this, the LQR controller's R matrix, which weights the input variables, had its last entry set to zero. This ensures the controller disregards this specific input when calculating the control matrix. While this controller may be slower in achieving a response similar to that of the Koopman-based method, it does stabilize the coordinates at the desired position with no overshoot or oscillations.

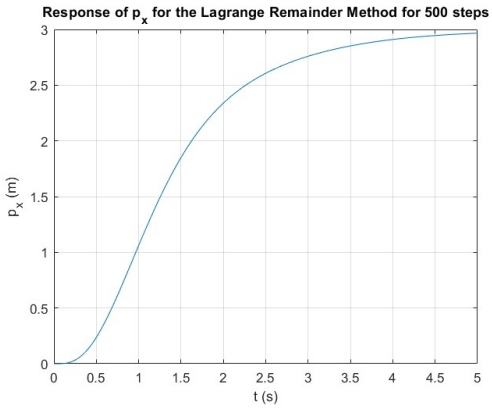


Figure 3.7: Lagrange System Response for p_x with LQR Controller for 500 Steps

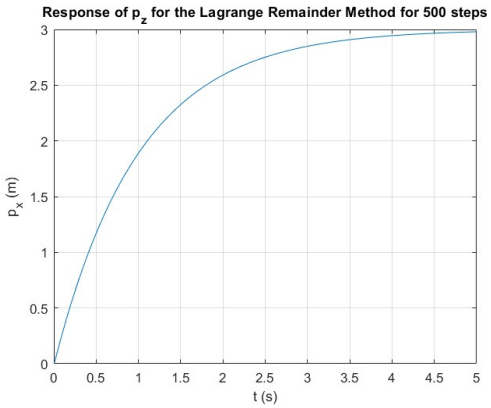


Figure 3.8: Lagrange System Response for p_z with LQR Controller for 500 Steps

In contrast, for the Koopman method, the LQR controller was designed solely for the controllable states, while for the uncontrollable states, the controller allows them to pass through without feedback. This configuration leads to a gain matrix K that is unitary for these states. This setup was necessary due to the instability and lack of controllability in the system matrices produced by the Autokoopman algorithm. This issue did not occur with the linearized system in the Lagrange method, simplifying the controller design in that case.

This difference in controller structure leads to variations in system response. The figures below show the response of p_x and p_z coordinates under both methods to the control law imposed over 100 steps, simulating the stipulated runtime. These simulations were conducted without set operations, and both controllers used the same disturbance matrix to represent actuator variations. Although the controller designs differ, the same proportional weighting was applied to p_x and p_z to maximize similarity.

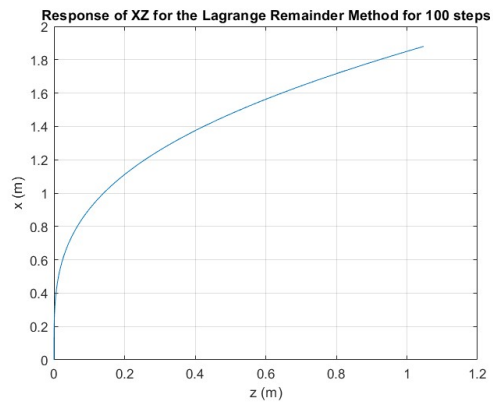


Figure 3.9: Lagrange System Response for p_x and p_z for 100 Steps

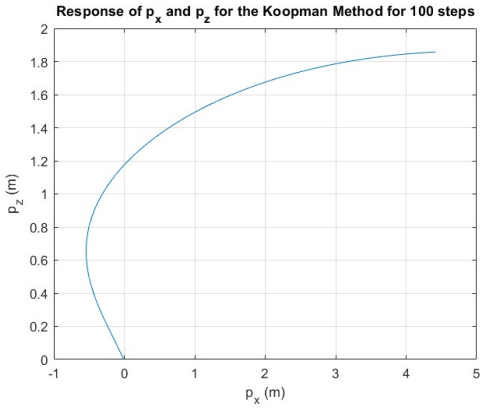


Figure 3.10: Koopman System Response for p_x and p_z for 100 Steps

As expected, the Koopman response shows certain disadvantages absent in the Lagrange response, such as significant overshoot in the p_x coordinate and the UAV failing to stabilize at the desired position, extending far past the 3-meter mark. Although the Koopman controller offers a faster response for the p_x state compared to the Lagrange method, it has a slower response for p_z, and the Lagrange controller ultimately stabilizes at the desired position over the longer runtime. Adjusting the weight matrix Q to decrease the p_x variable weight could mitigate the overshoot, yet the UAV would still not stabilize, rendering the result unsatisfactory.

Due to these response differences, the obstacle heights and positions in the UAV's path were adjusted, as the UAV follows a different trajectory in each case. Additionally, the remaining simulations must account for this instability. While not ideal, this outcome was anticipated due to domain differences and the significant approximations inherent in the Koopman method. Consequently, the object for which collision is guaranteed measures $0.8\,\text{m} \times 10\,\text{m} \times 1.65\,\text{m}$ and is positioned at $\begin{bmatrix}3.5 & 0 & 0.825\end{bmatrix}$ m, while the obstacle outside the UAV's path measures $0.8\,\text{m} \times 10\,\text{m} \times 1.8\,\text{m}$ and is located at $\begin{bmatrix}3.5 & 0 & 0.9\end{bmatrix}$ m.

A further limitation of the Koopman method is that estimation must be performed within the Koopman domain. Consequently, even without sensor faults, the UAV trajectory must be simulated using the Autokoopman-derived system to maintain compatibility with the estimation zonotope. This may produce results that appear more accurate than they would be in the real dynamics coordinate space, given the similarity between the estimation and propagation methods.

3.6.3 Lagrange Remainder

The Lagrange remainder method was simulated as described in Section 3.4.4. In this approach, since the coordinate space remains consistent with the original dynamics, the UAV propagates in fault-free conditions through a direct linearization of the original ODEs, as previously established. The Lagrange remainder linearization and error matrix are applied solely for the purpose of state estimation.

3.6.3.A Safety Verification

The next simulation considers the scenario with the first obstacle. In this case, we assume no sensor failure or UAV collision. However, this setup now introduces state estimation using the Lagrange remainder method. Every ten steps, when the state zonotope is displayed, the algorithm calculates a set representing the predicted state ten steps into the future.

This process provides a safety guarantee for the UAV even if a GPS sensor fault arises shortly. It also confirms the Lagrange remainder's accuracy in estimating the state, as the center of the estimation zonotope should align with the center of the next measured zonotope if the estimation is accurate.

Since the state estimation zonotope generated by this method is an over-approximation, its intersection with an obstacle does not necessarily mean a collision but rather suggests a possible risk if the sensor fails and no course correction occurs. Below is the plot of the simulation, demonstrating set-based state estimation every 10 steps to confirm the safety of the UAV.

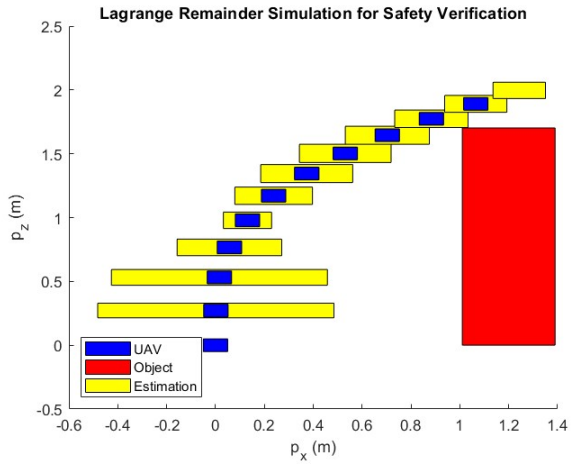


Figure 3.11: Safety Verification for the Lagrange Method

In this figure, the yellow set indicating state estimation does not intersect the object, as no intersection zonotope appears. Thus, under the Lagrange remainder method, the UAV avoids collision with the object throughout the next ten steps across the simulated trajectory, even with potential sensor failure. This outcome benefits users, as even in a worst-case scenario with the established parameters, the UAV design prevents collision in the near future.

Note that, as mentioned, disturbance vectors were randomly generated and kept constant across all simulations. However, larger disturbances or higher-than-anticipated GPS sensor inaccuracies could still occur, so using this method under different conditions might indicate collision risk.

Despite this safety verification, the Lagrange remainder estimation zonotope varied in size unexpectedly. This variability results from the l_i calculation in (3.4.1), which depends on a range of constantly shifting factors such as the zonotope's position and generator values. While the first three generators

remain constant due to GPS measurements, the other nine change with propagation. This can make the method overly conservative, as in the first ten steps, where the estimation set greatly over-approximates the state, or less reliable, as seen between steps 30 and 40, where the estimated set nearly matches the UAV state zonotope. Thus, while the method is effective for verifying safety, it lacks consistency for drawing conclusions about UAV behavior during sensor failure.

It is also important to note that the estimation of the zonotope center was accurate throughout the simulation, always aligning with the UAV's position center, despite the fact that the UAV propagation used a simple linearization of real dynamics while the estimation relied on the Lagrange method.

3.6.3.B Sensor Failure

The following simulations examine the scenario of an actual sensor failure. For the first 80 steps, the system will operate as usual, representing the functioning GPS sensor. After that, the sensor will fail, and the controller will no longer receive real-time position measurements. As a result, it will rely solely on the Lagrange remainder method for state estimation, using these estimates for controller feedback and obstacle detection.

Two situations are simulated: in one, the GPS sensor resumes functionality at $t = 0.9\,\mathrm{s}$, and in the other, the sensor fault persists until the end of the simulation, corresponding to 20 steps. It's expected that the longer fault will yield more conservative, over-reliable estimates as over-approximation errors accumulate throughout the state propagation. GPS sensor recovery is again indicated by the intersection of the estimated zonotope with the zonotope representing measurement errors.

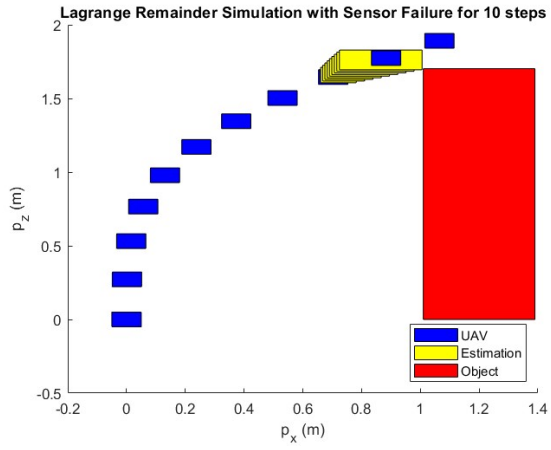


Figure 3.12: Sensor Failure for 10 Steps for Lagrange Method

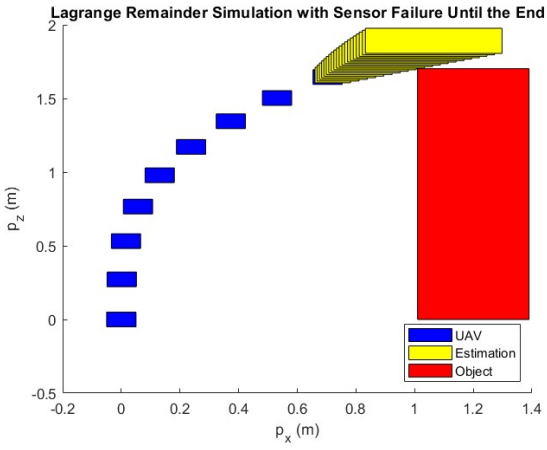


Figure 3.13: Sensor Failure Until the End for Lagrange Method

In the first simulation, with a GPS failure lasting ten steps starting at $t = 0.8\,\mathrm{s}$, the estimation zonotope does not intersect the object. While this is in line with prior safety verification, the Lagrange remainder

zonotope remains variable in size, meaning the intersection could have occurred at some point during the ten steps even if it didn't in the final step. However, as expected, the estimation zonotope increased over time, reflecting cumulative over-approximation errors. This increase is primarily influenced by the growing sum of the generator vectors, which scales with the size of the prior zonotope. Thus, while the initial estimate may fluctuate due to varying uncertainty factors, the zonotope's size and shape increase proportionally once it becomes dependent solely on previous estimates.

Upon the GPS sensor's return, the UAV realigns to its planned trajectory, enabled by the Lagrange method's accurate estimation of the zonotope's center. If this center estimate had been significantly off, the controller would have needed to correct the UAV back to its path, increasing the chance of a collision.

In the second simulation, where the sensor fault lasts through the remainder of the simulation, no intersection occurs between the estimated set and the obstacle. Although the estimation zonotope increases significantly in size due to propagation over-approximations, it follows a path similar to the UAV, avoiding the obstacle, mainly due to the zonotope center's accuracy. The expansion occurs most noticeably along the p_x direction, which contributes to obstacle avoidance; this effect aligns with the orientation of the obstacle, representing a wall. If the obstacle had been positioned differently, the increased estimation size in the x direction could have led to collision detection, given the larger over-approximation.

However, if the sensor fault were to extend beyond twenty steps, the risk of collision with the object would increase, as the controller would be working with an increasingly broad state estimate. In the absence of GPS measurements, initial errors may misdirect the UAV, potentially requiring a more advanced controller for sustained obstacle avoidance or human intervention to maintain safety. Nonetheless, since no intersection occurs here, this method offers reliable safety assurance for longer than demonstrated in the previous safety verification simulation.

3.6.3.C Zonotope Splitting

The final simulation for the Lagrange method involves splitting the initial uncertainty zonotope into two smaller subdivisions to evaluate if this approach mitigates over-approximations, particularly those related to the Lagrange remainder matrix.

Following the previously discussed zonotope splitting theory, the initial zonotope was divided along the x-axis and then the z-axis, each in separate simulations. These subdivisions were then propagated for 100 steps, with state estimation zonotopes calculated every ten steps as described in 3.6.3.A. In these simulations, the uncertainty zonotope is divided into blue and green colors corresponding to the subdivisions, represented by UAV_1 and UAV_2. The plots for these simulations are provided below:

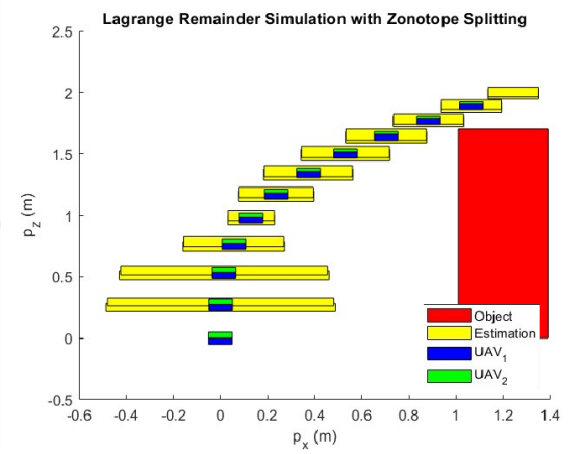


Figure 3.14: Simulation for the Zonotope Split by the z Axis with Lagrange Method

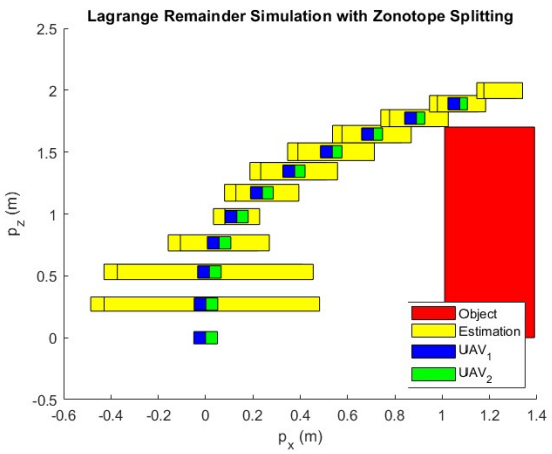


Figure 3.15: Simulation for the Zonotope Split by the x Axis with Lagrange Method

The results suggest that the estimation sets do not exhibit significant improvement in size or shape following zonotope splitting and remain unpredictable and often over-approximated. Nonetheless, there appears to be a slight directional improvement in the estimation set size in alignment with the chosen split axis. In the first figure, the estimation set is slightly smaller in the z direction. In contrast, in the second figure, a similar reduction is observed along the x direction—these being the respective axes selected for subdivision. This improvement arises primarily from the reduction in generator summation for l_i calculation, as smaller zonotopes have fewer generators, particularly in the split direction. Additionally, the propagation appears slightly affected by the division, as the two subdivisions show overlap near the end of the simulation, resulting in overlapping estimation sets.

On the other hand, the zonotope shows increased growth along the axis that was not split. This outcome is likely due to the nature of the J_i variable, which remains mostly sparse since second-degree terms are minimal in the ODEs. This sparsity reduces the impact of system dynamics on the Lagrange remainder error matrix, making it much more sensitive to the zonotope's position, which was altered. While the subdivisions maintain almost identical positioning to the initial zonotope, the differences are sufficient to cause slight impacts in this context.

Overall, zonotope splitting produced only minor, inconsistent changes and did not address this method's primary challenge — the unpredictability in estimation set size and shape. Although this approach may warrant further investigation, the current calculation of the Lagrange remainder matrix shows limitations when applied to high-dimensional systems with low-degree equations.

3.6.4 Koopman Operator

The following results illustrate the UAV trajectory within the Koopman domain, where both obstacles are present, no sensor failure occurs, and the stipulated controller is used:

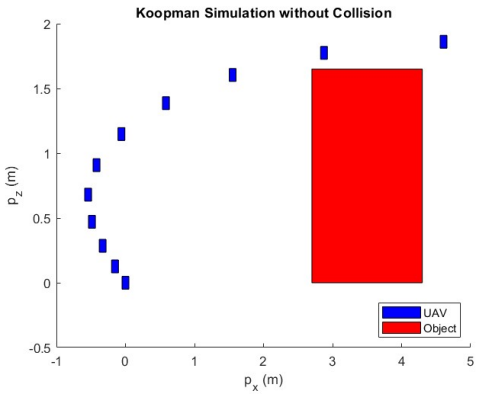


Figure 3.16: Koopman Simulation for Obstacle Outside UAV's Path

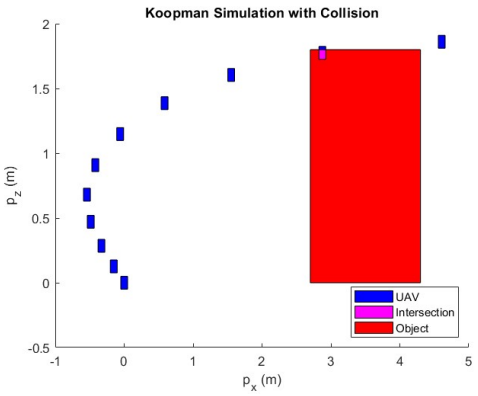


Figure 3.17: Koopman Simulation for Obstacle In UAV's Path

As anticipated, the UAV intersects with the second obstacle in one simulation, indicated by the presence of an intersection zonotope. The trajectory followed in both cases aligns with the system response observed earlier, without any zonotope enlargement linked to the GPS sensor. The second simulation, where the intersection occurs, demonstrates the UAV behavior in the event of a collision. Future simulations will use the first obstacle to assess whether safety is maintained in the Koopman domain during sensor failure. Note that the outcomes may differ from those using the Lagrange remainder method due to variations in trajectory, obstacles, and the estimation set calculation approach, resulting in differences in the zonotope's size and shape.

3.6.4.A Safety Verification

In parallel to the safety verification conducted previously, the first situation with no collision is considered here. Every 0.1 s, the algorithm calculates a state estimate ten steps into the future, this time using the Koopman method by propagating the state set with the Autokoopman matrices.

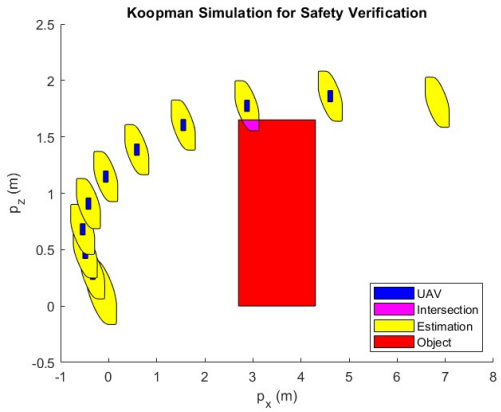


Figure 3.18: Safety Verification for the Koopman Method

In this scenario, the results suggest a potential collision if sensor failure were to occur at 0.9 s. This risk stems from the controller relying solely on an over-approximated state estimate, which could lead to minor UAV trajectory deviations.

Although this outcome diverges from the Lagrange remainder method's safety verification results, it does not imply that the Koopman method is less reliable. Multiple factors differ here due to propagation in the Koopman domain. The Koopman-based state estimation appears more consistent than the Lagrange remainder approach, with the estimation zonotope size remaining relatively stable. This consistency arises because the estimation set is derived by applying a linear map of the zonotope through the dynamics resulting from Autokoopman. Therefore, the estimation set's shape and size largely depend on the UAV size rather than its position, as seen in this simulation where the GPS sensor is assumed fully operational, maintaining a stable UAV uncertainty zonotope.

Regarding the over-approximation errors, these are present but manageable for a period of ten steps.The estimation set is somewhat larger than the UAV zonotope, providing a cautious margin without excessive enlargement. The estimation of the center of the state zonotope is also very exact, likely due to the propagation of the UAV happening in the Koopman domain.

3.6.4.B Sensor Failure

The results for the fourth and fifth simulations, where actual GPS sensor failure is considered, are presented below. As in the previous method, the GPS sensor fails for ten steps starting at $t = 0.8$ s. Normal sensor function resumes at $t = 0.9$ s, intersecting the estimation set with the GPS measurement zonotope. However, this intersection does not occur in the second simulation, requiring the UAV to rely solely on state estimation for collision prediction through to the simulation's end, which spans twenty steps.

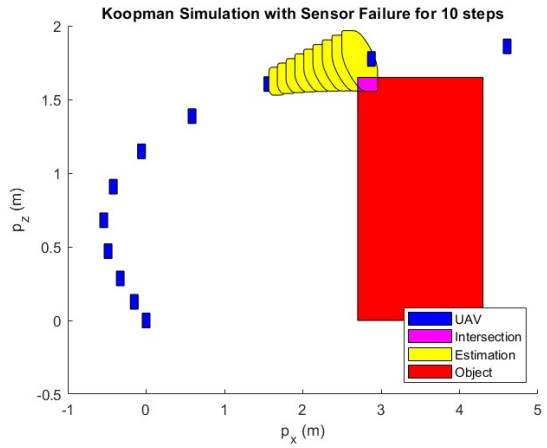


Figure 3.19: Sensor Failure for 10 Steps in the Koopman Method

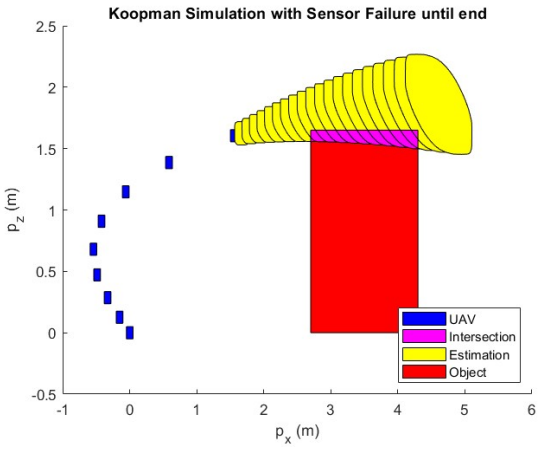


Figure 3.20: Sensor Failure Until End in the Koopman Method

In the first simulation, the GPS sensor's failure at $t = 0.8\,\mathrm{s}$ for ten steps results in the estimation zonotope intersecting with the object, as anticipated from the previous simulation. Similar to the Lagrange remainder estimation, the zonotope's size increases with each propagation step. However, it does not maintain its shape due to the Autokoopman matrix A's SVD S_{max} value, which expands certain variables in specific vector directions, creating an increasingly non-rectangular shape. This shape transformation, particularly along favored vectors, is a major contributor to over-approximation errors.

As with the Lagrange method simulation, the UAV returns to its planned trajectory once GPS functionality is restored, aligning the centers of the state estimation zonotope and the actual UAV zonotope. However, this realignment only prevents a collision if one has not already occurred. Suppose GPS measurement errors had been present at the failure's onset. In that case, the UAV might have been misdirected, potentially leading to a collision—an outcome not seen in the previous method.

In the second simulation, over-approximations noticeably increase with each step. The favored directions from the SVD value significantly influence the shape of the estimation zonotope. In scenarios where no collision is detected, this enlarged set provides a strong safety guarantee by accounting for a worst-case scenario. When collision is detected, however, it may be a false positive, as the UAV might avoid the obstacle despite the over-approximated set's indications. This outcome still signals a nonzero collision risk, advising caution even if an actual collision is unlikely.

3.6.4.C Zonotope Splitting

Previous sections concluded that zonotope splitting did not significantly alleviate the over-approximation issues with the Lagrange remainder matrix. However, based on the experiment shown in 3.5.1 and the over-approximations introduced when calculating the zonotopes' cosine, it is reasonable to test this approach with the Koopman simulations. An initial experiment was conducted to determine whether splitting the initial zonotope into sub-divisions during propagation could improve the estimation results.

To do this, the initial UAV state zonotope, in its original coordinate space at $t = 0\,\mathrm{s}$, was split into 2, 4, and 8 sub-divisions along different axes. These were then transformed into Koopman coordinates and propagated over 10 steps using only the Koopman matrices for estimation. It was necessary to divide the zonotope in the original coordinate space, as many of the over-approximations stemmed from the transformation between coordinate domains. Splitting the zonotope directly in the Koopman domain, as with the Lagrange method, would likely show minimal improvement. The results below display how the initial zonotope was divided and how the final estimation zonotopes compare to the original set, where the estimation set without zonotope splitting is shown in red, and the sub-divided regions appear as other colored areas, represented by UAV$_x$.

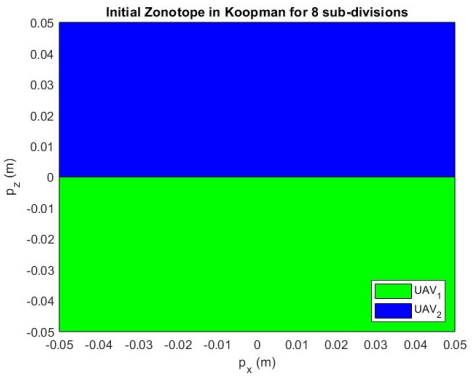


Figure 3.21: Original Koopman State Zonotope Split Into 2 Parts

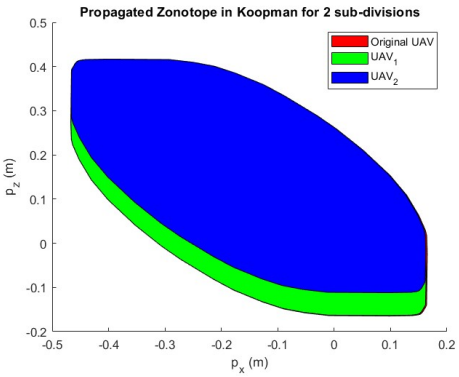


Figure 3.22: Koopman Propagation of Zonotope Split Into 2 Parts

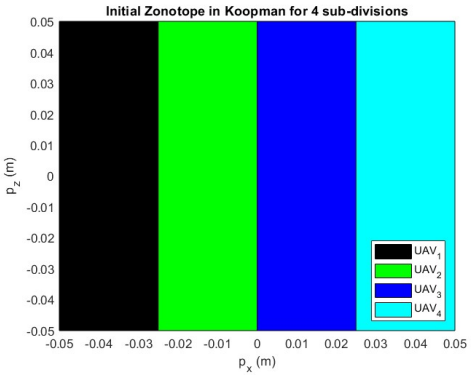


Figure 3.23: Original Koopman State Zonotope Split Into 4 Parts

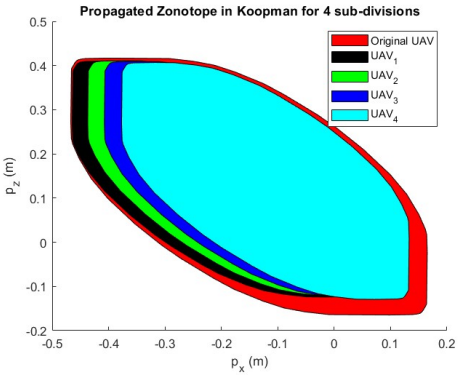


Figure 3.24: Koopman Propagation of Zonotope Split Into 4 Parts

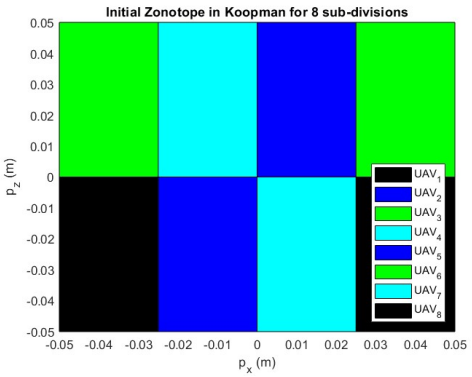


Figure 3.25: Original Koopman State Zonotope Split Into 8 Parts

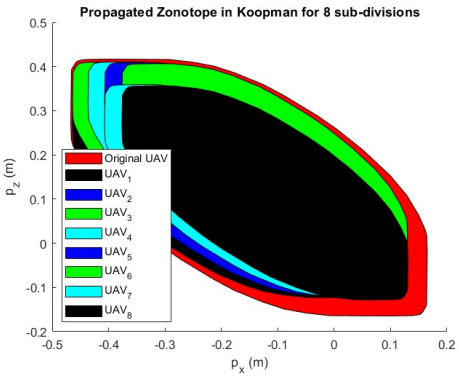


Figure 3.26: Koopman Propagation of Zonotope Split Into 8 Parts

In contrast to the Lagrange section experiments, this setup shows that over-approximation error decrease as the number of sub-divisions increases. The difference in area between the split sets and the original estimation set becomes more pronounced with further sub-division. In the sub-division into

2, the difference in sizing between the original propagated UAV state and the divided versions is barely noticeable. On the other hand, the improvement in the over-approximations becomes more prominent for the following cases. While theoretically, an infinite number of sub-divisions would yield the ideal result, implementing numerous subdivisions is computationally costly due to the optimization required when transforming zonotopes into the Koopman domain.

Consequentially, testing sub-divisions in the safety verification simulations remains relevant, as this approach may help prevent false collision detections due to over-approximations for a large number of sub-divisions. Shown below are the simulation results from the safety verification method outlined in 3.6.4.A, using initial zonotope splits of 2, 4, and 8 sub-divisions.

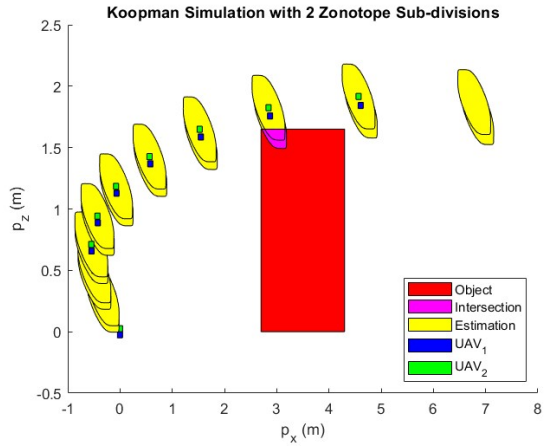


Figure 3.27: Koopman Safety Verification for 2 Zonotope Sub-divisions

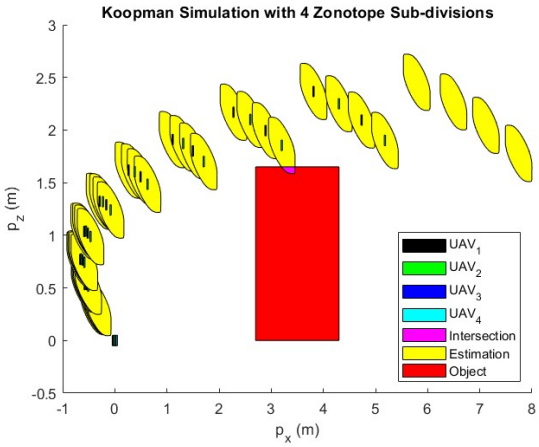


Figure 3.28: Koopman Safety Verification for 4 Zonotope Sub-divisions

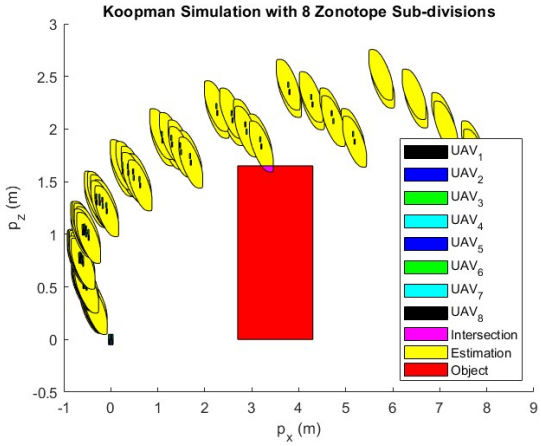


Figure 3.29: Koopman Safety Verification for 8 Zonotope Sub-divisions

The estimation sets produced by these sub-divisions differed significantly from those in the Lagrange method due to the nonlinear transformation into the Koopman domain and the associated propagation

errors. The propagation process, reliant on the matrix A and its SVD S_{max} values, significantly distorts and over-approximates zonotope shapes. Unlike the previous experiment, the combined area of these estimation sets is smaller than that of the original zonotope, with the error decreasing as the number of sub-divisions increases.

A relevant disadvantage, however, is the fact that the subdivided zonotopes propagate in a physically unrealistic manner, separating from each other even though the UAV should be represented by a single convex set. This behavior arises from two factors: the shift in zonotope centers upon splitting, which alters initial conditions, and the inaccuracies in the Koopman dynamics due to Autokoopman's unstable controller design. Consequently, the system, being partially uncontrolled, responds differently to minor changes in initial conditions.

There was also a difference in collision detection, occurring in one of the sub-divisions in the second simulation and in two sub-divisions for the others. It is clear that, for the situation stipulated, splitting by the x-axis provides an advantage in diminishing the number of intersected sub-sets, as it was the case for the zonotope split in 4. If not for the flawed propagation, this method could effectively identify which part of the uncertainty zonotope contributes to collision risk, aiding in further obstacle avoidance. By further sub-dividing, one could determine if collision detection is solely a result of over-approximation and, if not, devise strategies to avoid it.

While the benefits of zonotope splitting for set-based estimation are challenging to demonstrate due to the Koopman domain's intrinsic errors, such as coordinate transformation errors and flawed linearization matrices, minor improvements in over-approximation errors were observed in short-term simulations, such as 3.6.4.C. An enhanced Autokoopman algorithm producing stable, controllable matrices that more accurately approximate the real system could reduce these issues, leading to substantial improvements in the state estimation zonotope's accuracy.

4

Conclusion

Contents

4.1 Developed Work

Most existing algorithms for UAV obstacle detection and avoidance greatly rely on the state information gathered from its sensors or on prediction based on machine learning techniques. In this thesis, methods for safety verification regarding obstacles in the vicinity of the UAV's path under sensor failure conditions were developed by applying state estimation using convex set operations and a system identification algorithm.

Initial research regarding system identification led to the development of the SINDy and Koopman operator approaches. By evaluating results for both the UAV and a flexible structure, it was possible to conclude that the former, although effective in lower-dimensional systems, exhibited limitations regarding complex, high-dimensional systems caused by the choice of hyperparameters. On the other hand, the latter showed a more accurate approximation of the real system while also allowing for a coordinate space in which the system's dynamics were linear. This was directly applied to the state propagation necessary for collision prediction.

From this work, two solutions were proposed for the situation at hand. The first one consisted of applying existing literature about the Lagrange remainder matrix, which provides an over-approximation of the linearization error, to convex sets as a way to propagate the state and verify if there is the possibility of intersection with the obstacle. The second one, similarly to the latter, also predicts collision through zonotope intersection, but the propagation of the state is made through linear maps in the Koopman domain, with the system resulting from the Autokoopman algorithm. These methods allowed to perform simulations regarding safety verification throughout an UAV's path and also to check if obstacle detection is possible even during GPS sensor failure. Both of these methods heavily relied on over-approximations, therefore, it was proposed that the initial uncertainty zonotope was split and then propagated using these sub-divisions, to minimize the error.

The Lagrange remainder method was able to guarantee the safety of the UAV if the sensor fails for $0.1s$ and to detect the possibility of detection if the fault lasts more than that. The state estimation zonotope resulting from this method was unpredictable in size, being sensitive to many factors, such as the position and size of the uncertainty zonotope. This implies that some estimation sets were over-reliable, leading to possible false positives in detecting the chances of collision due to their unrealistic size, and some estimation sets that were almost the same size as the uncertainty zonotope, which is great to prevent the false detection of collision but is less reliable. While the results were able to provide a sense of security in guaranteeing that collision will not occur for a short period in the future, due to this constant change in the estimation zonotopes' sizes, this method is inherently flawed, making it difficult to predict its results for other systems or paths. Contrary to what was expected, the over-approximations of this method were not improved by zonotope splitting due to the nature of the original dynamics.

The Koopman operator method started at a disadvantage due to being developed in the Koopman

domain. The propagation was made using the matrices resulting from the Autokoopman, which were unstable and uncontrollable, in a space with 162 dimensions. This made for the difficulty in the controller design and an unstable and unpredictable behavior of the states. Due to the placement of the zonotope, the possibility of collision was detected if the sensor failed for $0.1s$, but this is not a setback of this method compared to the previous. However, for the same simulation time, the Koopman method showed a much more consistent sizing of the state zonotope, which made it more predictable, as it only proportionally depended on the size of the uncertainty zonotope, which was constant. It does have considerable over-approximations associated, specifically in the transformation of the sets to the Koopman domain and the propagation using matrices that disproportional expanded specific directions, so it is only reliable for a short period until the estimation zonotope becomes unrealistically over-sized. This situation was improved by the splitting of the original uncertainty zonotope. Still, due to the flaws in the propagation matrices and controller design, the sub-divisions of the zonotope separated when propagated, which does not make physical sense. However, it was proven that for a short running time, the sub-divisions' estimation sets do not separate and, in fact, together create a smaller area than the original estimation zonotope.

For an improved Autokoopman algorithm that better linearized and approximated the real dynamics, this method would prove to be useful in safety verification when sensor failure since it provided a state estimation set that was reliable without being unrealistic and detected intersection with the obstacle, even in a different coordinate space. Moreover, the main disadvantage of great over-approximations when propagating the estimation zonotope could be improved by splitting the initial uncertainty zonotope in ideal conditions.

4.2 Future Work

As mentioned, an essential part of continuing this line of work would be to improve the existing methods for system identification since many lack a space in which the dynamics are linearized, and the method that was experimented with was flawed and resulted in a nearly uncontrollable system to an otherwise controllable one. Moreover, the propagation in the space of the Koopman domain was inaccurate and prevented going into developments such as zonotope splitting.

Afterward, it would be important to apply this work to field experiments, trying the safety verification simulations in real-case scenarios, where a user would have access to the detection of the possibility of collision, so intervention would be possible. An alternative would be to develop obstacle avoidance algorithms that did not depend on sensors and could contour the obstacle even with existing state measurement errors. This work could then be expanded into other more complex systems, such as flexible structures, as it was attempted.

Bibliography

[1] G. N. Muchiri and S. Kimathi, “Practical experiences of uavs in precision agriculture and remote sensing,” *Proceedings of the 2016 Sustainable Research & Innovation (SRI) Conference*, 2022. [Online]. Available: https://sri.jkuat.ac.ke/jkuatsri/index.php/sri/article/view/325/301

[2] S. NT and P. Rajalakshmi, “Obstacle detection and collision avoidance on uav using rangefinder sensor with kalman filter technique,” in *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)*, 2022, pp. 1–6.

[3] S. Huang, F. Liao, and R. S. H. Teo, “Fault tolerant control of quadrotor based on sensor fault diagnosis and recovery information,” *Machines*, vol. 10, no. 11, p. 1088, 2022, special Issue: Dynamics and Motion Control of Unmanned Aerial/Underwater Vehicles. [Online]. Available: https://doi.org/10.3390/machines10111088

[4] C. E. Valero, M. E. Villanueva, B. Houska, and R. Paulen, “Set-based state estimation: A polytopic approach,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 277–11 282, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320306510

[5] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, “Reachable set estimation for neural network control systems: A simulation-guided approach,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2021.

[6] B. Wang and T. Chen, “Gaussian process regression with multiple response variables,” *Chemometrics and Intelligent Laboratory Systems*, vol. 142, pp. 159–165, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743915000180

[7] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, “Multifactorial genetic programming for symbolic regression problems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4492–4505, 2020.

[8] B. Grosman and D. R. Lewin, “Automated nonlinear model predictive control using genetic programming,” *Computers & Chemical Engineering*, vol. 26, no. 4, pp. 631–640, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135401007803

[9] D. Shen, H. Wu, B. Xia, and D. Gan, “Polynomial chaos expansion for parametric problems in engineering systems: A review,” *IEEE Systems Journal*, vol. 14, no. 3, pp. 4500–4514, 2020.

[10] N. Lüthen, S. Marelli, and B. Sudret, “Sparse polynomial chaos expansions: Literature survey and benchmark,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 9, no. 2, pp. 593–649, 2021. [Online]. Available: https://doi.org/10.1137/20M1315774

[11] Setlak, Lucjan and Kowalik, Rafał, “Studies of 4-rotor unmanned aerial vehicle uav in the field of control system,” *MATEC Web Conf.*, vol. 210, p. 05009, 2018. [Online]. Available: https://doi.org/10.1051/matecconf/201821005009

[12] R. J. Theodore and A. Ghosal, “Comparison of the assumed modes and finite element models for flexible multilink manipulators,” *Sage Journals*, vol. 14, no. 2, pp. 1–8, April 1995.

[13] X. Yang and Z. Zhong, “Dynamics and terminal sliding mode control of two-link flexible manipulators with noncollocated feedback,” *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 218–223, 2013, 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147466701631535X

[14] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Sage Journals*, vol. 113, no. 15, March 2016.

[15] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Sparse identification of nonlinear dynamics with control (sindyc),” *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016, 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316318298

[16] B. M. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, “Pysindy: A python package for the sparse identification of nonlinear dynamics from data,” 2020.

[17] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, “Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data,” *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020. [Online]. Available: https://doi.org/10.21105/joss.02104

[18] K. et al., “Pysindy: A comprehensive python package for robust sparse system identification,” *Journal of Open Source Software*, vol. 7, no. 69, p. 3994, 2022. [Online]. Available: https://doi.org/10.21105/joss.03994

[19] S. E. Otto and C. W. Rowley, “Koopman operators for estimation and control of dynamical systems,” *Annual Reviews*, vol. 4, pp. 59–87, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-071020-010108

[20] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern koopman theory for dynamical systems," 2021.

[21] S. L. Brunton, "Notes on koopman operator theory," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:229348032

[22] K. Baker, "Singular value decomposition tutorial," 01 2013.

[23] E. Lew, A. Hekal, K. Potomkin, S. Bogomolov, N. Kochdumper, S. Bak, and H. B., "Autokoopman: A toolbox for automated system identification via koopman operator linearization," 01 2023, pp. 237–250.

[24] S. Rajasekaran, "2 - free vibration of single-degree-of-freedom systems (undamped) in relation to structural dynamics during earthquakes," in *Structural Dynamics of Earthquake Engineering*, ser. Woodhead Publishing Series in Civil and Structural Engineering, S. Rajasekaran, Ed. Woodhead Publishing, 2009, pp. 9–42. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781845695187500026

[25] T. Idema, "8.04: Coupled oscillators," 2021, accessed: 2024-10-15. [Online]. Available: https://phys.libretexts.org/Bookshelves/University_Physics/Mechanics_and_Relativity_(Idema)/08:_Oscillations/8.04:_Coupled_Oscillators

[26] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, 2nd ed. Westview Press, 2015.

[27] L. Bröcker, "Euler integration and euler multiplication," *Advances in Geometry*, vol. 5, no. 1, pp. 145–169, 2005. [Online]. Available: https://doi.org/10.1515/advg.2005.5.1.145

[28] Z. He, R. Iyer, and P. Chandler, "Vision-based uav flight control and obstacle avoidance," in *2006 American Control Conference*, 2006, pp. 5 pp.–.

[29] M. Wzorek, J. Kvarnström, and P. Doherty, "Choosing path replanning strategies for unmanned aircraft systems," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 20, no. 1, pp. 193–200, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/13405

[30] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[31] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5876–5882.

[32] Y. He, T. Hou, and M. Wang, “A new method for unmanned aerial vehicle path planning in complex environments,” *Scientific Reports*, vol. 14, no. 1, p. 9257, Apr. 2024. [Online]. Available: https://doi.org/10.1038/s41598-024-60051-4

[33] C. E. García, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice—a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109889900022

[34] K. Holkar, K. Wagh, and L. Waghmare, “An overview of model predictive control,” *International Journal of Control and Automation International Journal of Control and Automation International Journal of Control and Automation International Journal of Control and Automation*, vol. 3, 01 2011.

[35] K. Sreenath, T. Lee, and V. Kumar, “Geometric control and differential flatness of a quadrotor uav with a cable-suspended load,” in *52nd IEEE Conference on Decision and Control*, 2013, pp. 2269–2274.

[36] M. Rathinam, R. M. Murray, and W. M. Sluis, “Differential flatness of mechanical control systems: A catalog of prototype systems,” 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:14167508

[37] R. Towle, “Polar zonohedra,” *The Mathematica Journal*, vol. 6, no. 2, pp. 8–12, 1996.

[38] V. Raghuraman and J. P. Koeln, “Set operations and order reductions for constrained zonotopes,” *Automatica*, vol. 139, p. 110204, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109822000498

[39] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, “Constrained zonotopes: A new tool for set-based estimation and fault detection,” *Automatica*, vol. 69, pp. 126–136, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109816300772

[40] M. Althoff, O. Stursberg, and M. Buss, “Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization,” in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 4042–4048.

[41] Y. Zhang and X. Xu, “Safety verification of neural feedback systems based on constrained zonotopes,” in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, Dec. 2022. [Online]. Available: http://dx.doi.org/10.1109/CDC51059.2022.9992655

[42] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” Ph.D. dissertation, Technische Universitat Munchen, 2010. [Online]. Available: https://mediatum.ub.tum.de/doc/1287517/document.pdf

[43] ——, “An introduction to cora 2015,” in *Proc. of the 1st and 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*, December 2015. [Online]. Available: https://easychair.org/publications/paper/xMm

A

Flexible Bar Matrices

The configuration matrix M is as such:

$$M = \begin{bmatrix} m_{11} & m_{12} & 0 & m_{13} & m_{14} & 0 \\ m_{21} & m_{22} & 0 & 0 & 0 & m_{26} \\ m_{31} & 0 & 0 & 0 & 0 & 0 \\ m_{41} & 0 & 0 & m_{44} & 0 & 0 \\ m_{51} & 0 & 0 & 0 & m_{55} & 0 \\ 0 & m_{62} & 0 & 0 & 0 & m_{66} \end{bmatrix}$$

with coefficients

$$\begin{aligned}
m_{11} &= \left(\frac{3}{4}m_1 + m_2 + M_1 + M_2\right) l_1^2 + m_1\left(\hat{\delta}_1^2 + \hat{\delta}_2^2\right), \\
m_{12} &= m_2 l_1 l_2 \cos(\theta_1 - \theta_2) + \frac{1}{2}m_1 l_1 \hat{\delta}_1 \sin(\theta_1 - \theta_2), \\
m_{13} &= m_1 \hat{l}_m l_1, \quad m_{14} = m_1 \hat{l}_m l_1, \\
m_{31} &= m_1 \hat{l}_m \cos(\theta_1 - \theta_2), \\
m_{51} &= \frac{1}{2}m_1 \hat{\delta}_1, \\
m_{22} &= \left(\frac{m_2}{2} + M_2\right) l_2^2 + m_1\left(\hat{\delta}_1^2 + \hat{\delta}_2^2\right), \quad m_{12} = m_2 l_2, \\
m_{26} &= -\frac{1}{2}m_2 \hat{l}_m l_2, \quad m_{33} = \frac{\pi^4}{2l_1^2}EI_1, \quad m_{44} = \frac{\pi^4}{2l_2^2}EI_2, \\
m_{55} &= \frac{1}{2}m_2, \quad m_{66} = m_2.
\end{aligned}$$

The Coriolis-centrifugal force can be described as:

$$h(q, \dot{q}) = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \end{bmatrix}$$

where:

$$\begin{aligned}
h_1 &= \left(\frac{1}{2}m_1 - m_1\right) l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + \frac{1}{2}(m_1 + M_2) l_2 \dot{q}_2 \sin(\theta_1 - \theta_2) \\
&\quad + m_2 \hat{l}_1 \dot{\theta}_1 \sin(\theta_1 - \theta_2) - m_2 l_1 \cos(\theta_1 - \theta_2) \\
&\quad + m_1 \hat{\delta}_1 \dot{\theta}_1 + m_1 \hat{\delta}_2 \dot{\theta}_2, \\
h_2 &= -\frac{1}{2}(m_1 + M_2) l_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + \frac{1}{2}m_1 l_1^2 \dot{\theta}_2^2 \cos(\theta_1 - \theta_2) \\
&\quad + m_2 \hat{\delta}_1 \dot{\theta}_1^2 + m_2 \hat{\delta}_2^2 \dot{\theta}_2^2,
\end{aligned}$$

$$
\begin{aligned}
h_3 &= -\frac{1}{2} m_1 \hat{\delta}_1 \dot{\theta}_1 \dot{\theta}_2, \\
h_4 &= -\frac{1}{2} m_1 \hat{\delta}_1 \dot{\theta}_1, \\
h_5 &= -\frac{1}{2} m_2 l_1^2 \dot{\theta}_1 \sin(\theta_1 - \theta_2) + \frac{1}{2} m_2 \hat{\delta}_2^2 \dot{\theta}_1, \\
h_6 &= -\frac{1}{2} m_2 \hat{\delta}_2 \dot{\theta}_2.
\end{aligned}
$$

The stiffness matrix is $N = \mathrm{diag}(N_{11}, N_{12}, N_{21}, N_{22})$, where:

$$
\begin{aligned}
N_{11} &= \frac{\pi^4}{2 l_1^2} E I_1, \quad N_{12} = \frac{8\pi^4}{l_1^3} E I_1, \\
N_{33} &= \frac{\pi^4}{2 l_1^2} E I_1, \quad N_{44} = \frac{8\pi^4}{l_2^3} E I_2.
\end{aligned}
$$

B

SINDy System Equations

2-State System

The state vector for this system is

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

, and the ordinary differential equation is:

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta)$$

4-State System

The state vector for this system is

$$x = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{bmatrix}$$

, and the ordinary differential equations are:

$$L\ddot{\theta}_1 = -g\sin(\theta_1) - \frac{k}{L}\left(\sin(\theta_1) - \sin(\theta_2)\right)$$

$$L\ddot{\theta}_2 = -g\sin(\theta_2) + \frac{k}{L}\left(\sin(\theta_1) - \sin(\theta_2)\right)$$

6-State System

The ordinary differential equations for the 6-state system are:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{g}{L}x_3 - cx_2 - ax_4$$

$$\dot{x}_3 = \sin(x_1)$$

$$\dot{x}_4 = 2x_2x_5$$

$$\dot{x}_5 = 3x_2x_6$$

$$\dot{x}_6 = 4x_2x_5$$

8-State System

The ordinary differential equations for the 8-state system are:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -2x_3 - x_3^2 - 0.5x_5 - 0.2x_5$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = x_1 - x_3^2 - 0.5x_5 - 0.2x_5$$

$$\dot{x}_5 = -0.5x_5 - 0.2x_5 - \sin(x_8)$$

$$\dot{x}_6 = 4x_2x_5$$

$$\dot{x}_7 = x_3$$

$$\dot{x}_8 = x_5$$