Kernelized Principal Component Analysis of a Linear Parameter Varying Model of the Gyroscope

Master's Thesis

Submitted in fulfillment of the requirements for the Joint Degree of

Master of Science in Mathematical Modelling in Engineering:

Theory, Numerics, Applications

Supervisor:

Prof. Dr. Herbert Werner

Prof. Dr. Timo Reis

Candidate:

Saeed Ghoorchian

Matriculation Number:

6872909

2016/2017

UNIVERSITÄT HAMBURG

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

UNIVERSITAT AUTÒNOMA DE BARCELONA

Erasmus Mundus Consortium "MathMods"

Joint Degree of Master of Science in

Mathematical Modelling in Engineering: Theory, Numerics, Applications

In the framework of the

Consortium Agreement and Award of a Joint/Multiple Degree 2013-2019

Master's Thesis

Kernelized Principal Component Analysis of a Linear Parameter Varying Model of the Gyroscope

Supervisors:

Prof. Herbert Werner

Prof. Timo Reis

Candidate:

Saeed Ghoorchian

Matriculation Number:

6872909

2016/2017

Laurea Magistrale in Ingegneria Matematica

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Università degli Studi dell'Aquila

Acknowledgements

I would first like to express my profound gratitude to my thesis supervisors, Prof. Dr. Herbert Werner and Prof. Dr. Timo Reis, for their valuable guidance, useful advice, and beneficial critiques during my research work. Their supervision helped me all the time of working on my thesis. I would like to send my special thanks to Mr. Adwait Datar for his constant support, endless help, fascinating patience and intense encouragement during this project. He truly has a big role in this thesis. I would like to acknowledge Prof. Bruno Rubino and Prof. Dr. Ingenuin Gasser for all their help and support during my studies as the MathMods student. Their cooperation provided me such a unique experience. Finally, I would like to thank my family for their encouragement and support throughout my studies. I am extremely grateful for all the confidence and kindness that I received from my mother, my sister and my brother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Principal Component Analysis (PCA) is a traditional mathematical method that extracts the main features of a given set of data points by finding the principal components of the data. To this end, PCA captures the principal axes in the space of the data points by maintaining the highest possible variance of the data points in that direction. This makes PCA a powerful algorithm when it comes to reducing the dimension of some given data points to be able to represent them with fewer number of components while keeping much useful information form the data.

If we are given a set of m data points $x_1, \dots, x_m \in \mathbb{R}^n$ (training data points) the goal of PCA is then to reduce the dimension of such data points to $\ell < n$. To do so, PCA tries to find a lower dimensional space and then project each data point onto this space. This space is constructed using principal axes (eigenvectors of corresponding covariance matrix) of the data points. PCA first sort these eigenvectors, measured by their corresponding eigenvalues, and keep the ones that contain the principal information of the data and finally project the data points onto the space constructed using these principal directions.

There are some cases, however, where finding the best principal axes are not easy, and ideally we cannot expect to maintain much useful information from the data points after reducing their dimension using PCA. In other words, we need more complex principal axes that can represent the reduced data points better. This is due to the fact that PCA algorithm gives us the principal directions in the space of data points (input space) as linear axes whereas in some situations we might need nonlinear ones. Thanks to kernel methods, we can tackle this problem by kernelizing the PCA algorithm. We call this kernelized version of PCA as Kernel Principal Component Analysis (KPCA) algorithm. In general, kernelized version of any algorithm involves mapping the data into a higher dimensional space (feature space) and then applying linear techniques in this space. Therefore KPCA maps the data points into feature space via a nonlinear map ϕ (feature map) to perform dimensionality reduction in that space using linear methods. The so-called kernel trick allows us to perform linear techniques in this higher dimensional space without the knowledge of the feature map ϕ.

The ability of PCA and KPCA to reduce the dimension of given data points makes it ideal for reducing the number of scheduling parameters in the Linear Parameter Varying (LPV) models. LPV systems are dynamical systems whose state-space matrices are function of time-varying parameters $\rho_i(t)$, $i = 1, \dots, n$, called the scheduling variables. One could then represent a general form of a nonlinear system as an LPV system (quasi-LPV) by hiding the non-linearity behind these scheduling variables. Our aim is to represent the LPV model with the less number of scheduling parameters to reduce the complexity of the model and consequently to reduce the computational cost of modelling and simulation as well as the cost of analysis and control synthesis.

The task of reducing the number of scheduling parameters can be seen as reducing the dimension of some data points if we stack up all these parameters in a vector $\rho(t) \in \mathbb{R}^n$ and treat them in different time samples as our data points. Therefore simplifying the task, we aim to reduce the number of scheduling parameters of the LPV model by reducing the dimension of the vector of scheduling parameters in each time sample. We call these vectors in different time samples as our patterns or simply data points.

Although many have exploited different methods to reduce the number of scheduling parameters of the LPV systems and represent the same inherent behaviour of the LPV model using reduced LPV model (e.g. [5], [6]) and there are general algorithms like PCA and KPCA to be implemented for this purpose, the performance of the reduced LPV model should be investigated case by case to make sure that the reduced LPV model is behaving close to the original full order LPV model. The is the inspiration that motivated us to start this thesis and investigate the application of these methods for a model of interest.

We have exploited the above mentioned algorithms to reduce the number of scheduling parameters for a LPV model of the gyroscope. In this thesis, we will show the corresponding results and conclude that how well the reduced LPV model is working in comparison to the full order LPV model. First, we start with a short summary of each chapter to familiarize the reader with the whole structure of this thesis. Meanwhile, additional references to the relevant topics are also given to address more details after studying each chapter.

1.1 Thesis Overview

Chapter 2: Mathematical Prerequisites

This chapter describes the mathematics required to understand the idea behind this work during reading this thesis. We try to get some intuition about what PCA and KPCA algorithms do. We will also present some basic theorems which help us to understand the idea behind these methods and how these algorithms work. We have mostly exploited [1], [7] and [8] to explain the theory behind these methods but as additional resources we refer to [9] and [10]. Together with these two algorithms, we will also introduce LPV systems in a more detailed way and explain the general form of such systems ([11]).

Chapter 3: Model Description

This chapter introduces the model of the gyroscope taken from [4] with which we will work throughout this book. First, we present the nonlinear model of the gyroscope together with all the details and parameters. For more details about the structure and function of gyroscope we refer to [12]. We will also propose a way of how to gain the LPV model out of the nonlinear model. We have derived this model and presented in this chapter.

Chapter 4: Preimage Problem

This chapter explains the preimage problem and how to achieve the reduced LPV model. First, it illustrates the problem and how to go back from feature space to input space. We will state the challenges regarding this work and what are other methods to do so. Following this, we will also introduce a proposed way in [6] of how to achieve the reduced LPV model with the desired number of reduced parameters.

Chapter 5: Performance

To know the performance of the reduced LPV model, this chapter introduces a measure proposed in [13] to see the accuracy of the reduced LPV model for different number of reduced LPV parameters and also different kernel function parameters. We have utilized this measure to be able to decide the number of reduced scheduling variables in reduced LPV model and choose the kernel functions and the value for their corresponding parameters.

Chapter 6: Implementation and Numerical Results

In this chapter we present our numerical results and compare the behaviour of the LPV model of the gyroscope with its reduced LPV model and also its nonlinear model. We will also talk about the way we have constructed our LPV and reduced LPV model. we will explain how we have trained our KPCA algorithm to find the principal directions and reduced training data points. We will show some interesting results depicting the behaviour of LPV and reduced LPV model to compare them for different choices of kernel functions, kernel parameters and different number of reduced scheduling variables.

Chapter 7: Conclusion

This chapter concludes the main results obtained during this work in a nutshell and summarizes the accomplishments gained in this thesis. As a guideline, we will also include an algorithmic procedure of how to perform the dimensionality reduction for the system of interest.

Chapter 2

Mathematical Prerequisites

This chapter describes the mathematical prerequisites related to Principal Component Analysis (PCA), Kernelized Principal Component Analysis (KPCA), and Linear Parameter Varying (LPV) systems that will be discussed in the present thesis. Its goal is to provide an overview of the basic concepts. Our aim is to keep this chapter as basic as possible. Consequently, we have presented the claims in this chapter without proofs. Nevertheless, we refer the reader to Appendix A to see a detailed proof of these claims. Extra references to the relevant papers or books will enable the interested reader to fill in the gaps. Moreover, throughout this chapter, we have assumed that the reader is familiar with the basic concepts from linear algebra ([14]), probability theory ([15]), and functional analysis ([16]).

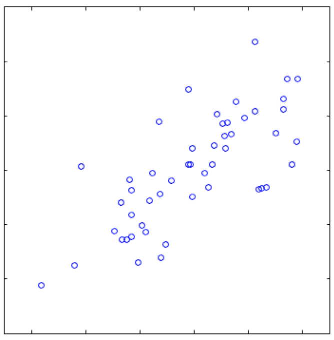
As explained at the beginning of this thesis, our aim simply is to reduce the dimension of some data points in a way that we can still keep useful information after dimensionality reduction. One way to do so is to find a lower dimensional space constructed by principal axes of the data points and project them onto this space. One might think of these axes as the directions in which variance of the data is as high as possible. This is the basic idea behind PCA algorithm (Section 2.1).

Section 2.2, however, describes the situation when the variance in different directions doesn't vary that much. In other words, they cannot represent the underlying information of the data points after projecting the points onto them. To bypass this problem we will introduce kernel functions and describe how to utilize them to gain a more powerful algorithm called KPCA. We note that most of the facts in the first two sections are coming from [1].

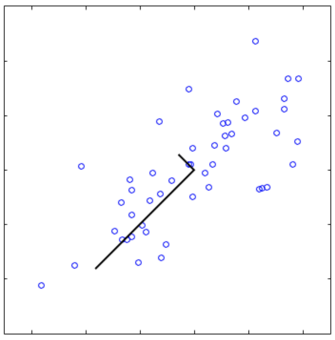
Finally, section 2.3, introduces the general form of LPV systems. Later we will aim to achieve a reduced LPV model which is in an affine form. Therefore we will also define the corresponding definition. As most of the definitions in this section are coming from [11], we refer the reader to this book for further studies.

2.1 Principal Component Analysis

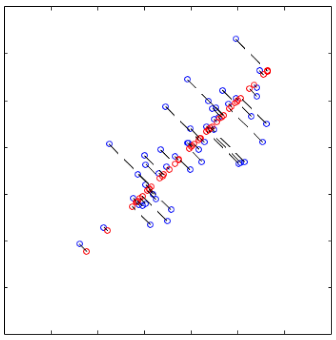
Let us start with a simple example to get some intuition about how PCA works. Assume we are given a set of m training data points $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ like the one shown in Fig. 2.1a. Throughout this thesis, we show the matrix of all training data points with $X \in \mathbb{R}^{m \times n}$ in which the training data are in the rows. Therefore the first row of X corresponds to the first training example x_1, the second row of X corresponds to the second training point x_2 and so on. We want to find the directions in which the variance of the data is highest after projecting them onto these directions. If we compute the covariance matrix $C = X^T X$ and solve the eigendecomposition problem for this matrix we will find the two eigenvectors depicted in Fig. 2.1b. We have drawn the eigenvector corresponding to the bigger eigenvalue with a longer line and the eigenvector corresponding to the smaller eigenvalue with the shorter line.



(a)



(b)



(c)

Figure 2.1: (a) Sketch of a sample set of training data points in $\mathbb{R}^2$ (b) Plot of eigenvectors of covariance matrix of the data points (c) Projection of the data points onto the first principal axis

Hence, we will project each single data point onto the first eigenvector, meaning the eigenvector corresponding to the bigger eigenvalue. In this way, we have projected them onto a single line which can be seen as the space $\mathbb{R}$. Therefore reducing the dimension of the data points from 2 to 1. The projected points onto this line which we call the principal axis or principal direction is shown in Fig. 2.1c The red circles represent the projection of data points.

Keeping in mind this intuition, we can now move on towards PCA algorithm. PCA is explained in algorithm 1. For a better understanding of how exactly this algorithm works we refer the reader to [1] or [7]. Note that in the following algorithm $\bar{x}$ denotes the average of the data points defined as: $\bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i$

To understand how the matrix $X_{Reduced}$ gives us the reduced points, note that to project the data points onto the eigenvectors we just need to calculate the inner product between them. This is obvious from the definition of inner product. Therefore denoting by $P_v(x_i)$ the function of projection which projects the points x_i onto the eigenvector v, the following gives us the projected point (reduced point corresponding to the data point x_i):

$$P_v(x_i) = v^T x_i \tag{2.1}$$

In the general case, if we aim to project on a subspace spanned by ℓ vectors $v_1, \ldots, v_\ell \in \mathbb{R}^n$, we utilize this formula to compute each of the ℓ coordinates of the reduced point. The reduced points correspond to the rows of matrix $X_{Reduced}$. Therefore, as an example, the first row of matrix $X_{Reduced}$ will correspond to projection of the data point x_1 onto the space constructed by first ℓ eigenvectors of covariance matrix (namely, the eigenvectors corresponding to first ℓ largest eigenvalues).

Algorithm 1: PCA

```
1 Given: Data points
      $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$
2 Center the data points:
      $\tilde{x}_i = x_i - \bar{x}$
  for all $i$
3 Compute the new matrix of data $\tilde{X}$ with the centered data points $\tilde{x}_i$
  and compute the convariance matrix    $C = \tilde{X}^T \tilde{X}$
4 Solve the eigendecomposition problem to achieve    $C = USU^T$
5 Define $U_\ell$ as the matrix containing the $\ell$ largest eigenvectors, i.e. the $\ell$
  eigenvectors corresponding to the first $\ell$ largest eigenvalues
6 Compute the new data points:
      $X_{Reduced} = \tilde{X} U_\ell \in \mathbb{R}^{m \times \ell}$
```

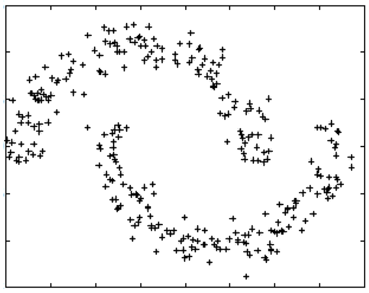


Figure 2.2: Plot of another sample set of training data in $\mathbb{R}^2$ [1]

2.2 Kernelized Principal Component Analysis

Now assume we are given another data set like the one shown in Fig. 2.2. PCA gives us the principal axes as lines, whereas in this case we would like to have nonlinear axes so that we can retain much more information carried by the data points after projection. In this situation, the linear algorithms such as PCA don't work and we need stronger mathematical tools to tackle this problem.

To bypass this problem one uses the KPCA method which uses kernel functions to improve PCA. Generally speaking, the goal of kernel methods is to introduce a nonlinear component to linear methods. The next two sections describe the basic definitions and theorems which help us understand the KPCA algorithm better.

2.2.1 Kernel Trick

The idea of kernel methods is to project the data points into a higher dimensional space (*feature space*) to be able to extract the principal features of the data points in that space by linear methods and then preimage them into the original space (*input space*). To this end, we need to define a map called the *feature map* to project the data into feature space.

Let us again have a look at the PCA algorithm. Observe that to train this algorithm we only need to know the inner products between the training data points $\langle x_i, x_j \rangle$ as the covariance matrix is constructed only using the inner products of the data points: $C_{ij} = x_i^T x_j = \langle x_i, x_j \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the inner product or scalar product. Also to evaluate the trained algorithm at the test point we only need to be able to compute scalar products between the test points and the training data points. Therefore if we want to replace the data points with their corresponding images under the feature map ϕ, we just need to know how to calculate the scalar products $\langle \phi(x_i), \phi(x_j) \rangle$ and we do not need to compute $\phi(x_i)$, $i = 1, \ldots, m$ explicitly. The idea of replacing the inner products $\langle x_i, x_j \rangle$ with $\langle \phi(x_i), \phi(x_j) \rangle$, is what is called the *kernel trick* which helps us move on towards KPCA algorithm.

2.2.2 Kernel PCA Algorithm

Definition 2.1. Kernel Function [1]
Let $\mathcal{X}$ be any space. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel function if for all $x_1, x_2, ..., x_m \in \mathcal{X}$ and $c_1, c_2, \ldots, c_m \in \mathbb{R}$ we have

$$\sum_{i,j=1}^{m} c_i c_j k(x_i, x_j) \geq 0$$

Given a set of points $x_1, x_2, \ldots, x_m$ we define the corresponding kernel matrix as the matrix K with entries $K_{ij} = k(x_i, x_j)$.

The next theorem shows us that kernel functions and feature maps lead to each other. This is an important theorem since it tells us about the structure of feature space and allows us to utilize kernel methods without knowing the feature map ϕ explicitly.

Theorem 2.1. ***Kernel implies embedding*** *[1]*
A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathcal{H}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

If we denote by Φ the $m \times n$ matrix that contains the data points $\phi(x_i)$ in the rows, then we can compute the corresponding kernel matrix as $K = \Phi\Phi^T \in \mathbb{R}^{m \times m}$ with the entries: $\quad K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$.

To understand the relation between covariance matrix C and kernel matrix K we can investigate the relation between their corresponding eigenvalues/eigenvectors (shortly denoted by Eigs). To this end, we present the two following two important theorems.

Theorem 2.2. ***Eigs of K via Eigs of C*** *[1]*
Consider a set of points $x_1, \ldots, x_m \in \mathbb{R}^n$. Assume that λ and a are an eigenvalue and corresponding eigenvector of the kernel matrix K (with respect to the standard scalar product). Assume that a is not orthogonal to span $\{x_1, \ldots, x_m\}$. Then $v := X^T a$ is an eigenvector of C with eigenvalue λ.

This theorems shows us that all pairs of eigenvalues and eigenvectors (λ, a) of K lead to the pairs (λ, v) of C, unless $v = X^T a = 0$.

Theorem 2.3. ***Eigs of C via eigs of K*** *[1]*
Assume that the points x_i are centered. Then to compute the ℓth eigenvector v of C we can proceed as follows:
- compute the matrix K
- compute its ℓth eigenvector $a = (a_1, ..., a_m)^T \in \mathbb{R}^m$

- make sure that $\|a\| = 1$.

- set $v = \frac{1}{\sqrt{\lambda}} \sum\limits_j a_j x_j$

Now we are in the position to kernelize the PCA algorithm. We have explained it in algorithm 2.

Algorithm 2: KPCA

```
1 Given: Data points
      $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$
2 Compute the kernel matrix $K$ with the entries $k_{ij} = k(x_i, x_j)$
3 Center the data in feature space using the following formula:
      $\tilde{K} = K - I_m K - K I_m + I_m K I_m$
4 Solve the eigendecomposition problem to gain $\tilde{K} = USU^T$ . Let $U_k$
    denotes the $k$th column of $U$ and $\lambda_k$ the corresponding eigenvalue.
5 Define $U_\ell$ as the matrix containing the $\ell$ largest eigenvectors, meaning
    $\ell$ eigenvectors corresponding to the first $\ell$ largest eigenvalues, which
    has the columns $\frac{U_k}{\lambda_k}, k = 1, 2, ..., \ell$
6 Compute the low dimensional representation points
      $Y_{Reduced} = \tilde{K} U_\ell \in \mathbb{R}^{m \times \ell}$
```

In this algorithm I_m denotes the $m \times m$ matrix with all the entries equal to $\frac{1}{m}$. To understand the last formula where we obtain the reduced points we should express the projection on eigenvectors of C using K. Assume we want to project on eigenvector v of C. We have already seen that we can write $v = \sum\limits_i a_i x_i$ (Theorem 2.2). Thus using equation 2.1:

$P_v(x_i) = v^T x_i = \langle \sum_j a_j x_j, x_i \rangle = \sum_j a_j \langle x_j, x_i \rangle$

We again see that to compute the projections we only need scalar products. Therefore all we have done is to replace the inner products in PCA algorithm with the corresponding entry of kernel matrix K.

2.3 Linear Parameter Varying Systems

We will start with the general form of nonlinear dynamical systems. Such systems can be represented by

$$\begin{cases} \dot{x}(t) = f(t, x(t), u(t)), \;\; x(0) = x_0 \\ y(t) = h(t, x(t), u(t)) \end{cases} \tag{2.2}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $y \in \mathbb{R}^{n_y}$ denote the vector of states, input, and output of the system. The model discussed in this thesis is in the shape of model 2.2.

As it is exclusively done in the literature on LPV systems (e.g. [17]) one can choose a set of parameters (the so-called *scheduling variables*) $\rho_i(t)$, $i = 1, \dots, n$, and hide the nonlinearity behind these parameters to write the dynamical system of such nonlinear systems as an LPV system (quasi-LPV). We usually collect these parameters in a vector and denote it by $\rho(t)$ and associate with this vector a compact set $\mathcal{P}$ and assume:

$$\rho(t) \in \mathcal{P} \subset \mathbb{R}^{n_\rho}, \quad \forall t \geq 0$$

The general form of LPV systems can be represented as follows:

$$\begin{cases} \dot{x}(t) = A(\rho(t))x + B(\rho(t))u, \;\; x(0) = x_0 \\ y(t) = C(\rho(t))x + D(\rho(t))u \end{cases} \tag{2.3}$$

where as we see the state space matrices of this system depends on time varying parameter vector $\rho(t)$. We can think of this vector of parameters $\rho(t)$ in the fixed time sample t as our data point in the space $\mathbb{R}^n$. Therefore by reducing the number of parameters we mean reducing the dimension of data point $\rho(t)$.

We shall also introduce the notion of affine LPV systems which we will use in this thesis. An LPV state-space representation is considered to be affine if

$$Q(\rho(t)) = Q_0 + \sum_{i=1}^{n} Q_i \rho_i(t)$$

where $\rho_i(t)$ is the ith entry of parameter vector $\rho(t)$ and $Q(\rho(t))$ is the compact representation of LPV model 2.3 defined as follows:

$$Q(\rho(t)) = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix}$$

Chapter 3

Model Description

This chapter introduces the model of interest which we shall work with throughout this thesis. We will begin by introducing the nonlinear dynamical system of the gyroscope together with all the parameters and details (section 3.1). We want to represent this model as an LPV model. To this end, we will propose a way of how to convert this model to the LPV form (section 3.2). The proposed way is based on partitioning the part of model which is not obviously linearly dependent on scheduling parameters to be able to write the whole model as an LPV system.

The control moment gyroscope can be described as a flywheel which is suspended in a ring. The rings are usually called gimbals. It has four degree of freedom, and each sector is linked to the previous part by a rotational joint perpendicular to the last joint axis. It can be mounted on a base or inside an instrument. The control moment gyroscope is shown in Fig. 3.1.

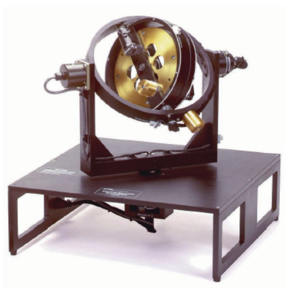


Figure 3.1: Control moment gyroscope [2]

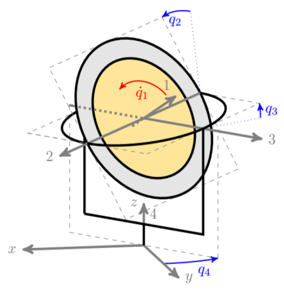


Figure 3.2: Kinematics of the control moment gyroscope [3]

According to the conservation of angular momentum, while the wheel is rotating, the direction of axis of rotation is unaffected even if the base is rotating or leaning. That is what makes them useful to be utilized for the purpose of maintaining orientation. For example they are used in compasses, the equipment that keeps large ships carriers from rolling on the waves, and automatic pilots on airplanes and ships.

The kinematic sketch of gyroscope is shown in Fig. 3.2. As we see the 4 different angles of disks or wheels are denoted by q_1, q_2, q_3, and q_4 and their corresponding angular velocities are shown by $\dot{q}_1, \dot{q}_2, \dot{q}_3$ and $\dot{q}_4$, respectively. Throughout this thesis we will denote the corresponding vector of angles and vector of angular velocities by

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$$

and

$$\dot{q} = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 \end{bmatrix}^T$$

, respectively.

3.1 Nonlinear Model of Gyroscope

Now we are in a position to present the nonlinear dynamical system of the gyroscope. For more details about this model and also the structure of the gyroscope we refer the reader to [12]. C. Hoffmann in [4] has derived the full nonlinear model of the gyroscope using Newton-Euler equations. The interested reader can refer to this paper for more details about the derivation of this model. Equation 3.1 represents the full nonlinear model of the gyroscope.

$$\begin{bmatrix} b_1 & 0 & b_1c_2 & b_1s_2c_3 \\ 0 & b_3 & 0 & -b_3s_3 \\ b_1c_2 & 0 & b_2s_2^2+b_4 & -b_2s_2c_2c_3 \\ b_1s_2c_3 & -b_3s_3 & -b_2s_2c_2c_3 & -b_2s_2^2c_3^2+b_5s_3^2+b_6 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \\ \ddot{q}_4 \end{bmatrix} + \begin{bmatrix} b_{13}\dot{q}_1 \\ b_{14}\dot{q}_2 \\ b_{15}\dot{q}_3 \\ b_{15}\dot{q}_4 \end{bmatrix}$$
$$+ \begin{bmatrix} k_1(\dot{q},q) \\ k_2(\dot{q},q) \\ k_3(\dot{q},q) \\ k_4(\dot{q},q) \end{bmatrix} = \begin{bmatrix} b_{16} & 0 \\ 0 & b_{17} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.1}$$

$$k_1(\dot{q},q) = b_1(c_2c_3\dot{q}_2\dot{q}_4 - s_2\dot{q}_2\dot{q}_3 - s_2s_3\dot{q}_3\dot{q}_4)$$

$$\begin{aligned} k_2(\dot{q},q) = {} & b_1(s_2\dot{q}_1\dot{q}_3 - c_2c_3\dot{q}_1\dot{q}_4) + b_2(c_2c_3^2s_2\dot{q}_4^2 - c_2s_2\dot{q}_3^2) \\ & - b_8c_3\dot{q}_3\dot{q}_4 + b_7(1-2s_2^2)c_3\dot{q}_3\dot{q}_4 + 2b_9c_2^2c_3\dot{q}_3\dot{q}_4 \end{aligned}$$

$$\begin{aligned} k_3(\dot{q},q) = {} & b_1(s_2s_3\dot{q}_1\dot{q}_4 - s_2\dot{q}_1\dot{q}_2) + (b_8+b_7)c_3\dot{q}_2\dot{q}_4 + b_{11}s_3c_3\dot{q}_4^2 \\ & + b_{10}(2c_2^2c_3\dot{q}_2\dot{q}_4 - 2s_2c_2\dot{q}_2\dot{q}_3 - s_3c_2^2c_3\dot{q}_4^2) \end{aligned}$$

$$\begin{aligned} k_4(\dot{q},q) = {} & b_1(c_2c_3\dot{q}_1\dot{q}_2 - s_2s_3\dot{q}_1\dot{q}_3) + b_2s_2s_3c_2\dot{q}_3^2 - 2b_{11}s_3c_3\dot{q}_3\dot{q}_4 \\ & + 2b_{10}(c_2^2c_3\dot{q}_2\dot{q}_3 + s_2c_2c_3^2\dot{q}_2\dot{q}_4 + s_3c_2^2c_3\dot{q}_3\dot{q}_4) + b_{12}c_3\dot{q}_2\dot{q}_3 \end{aligned}$$

In this model s_i and c_i are abreviations and denote:

$$s_i = \sin(q_i), c_i = \cos(q_i)$$

For simplicity we have dropped the index t which is usually used to represent the dependency of variables on the variable time. The physical parameters with their experimentally identified values are listed in the table 3.1. The vector of scheduling parameters of this model is also taken from [4] and is equal to

$$\rho = \begin{bmatrix} q_2 & q_3 & \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 \end{bmatrix}^T \tag{3.2}$$

Note that the vector of scheduling variables is in $\mathbb{R}^6$ and therefore we have 6 parameters as the scheduling variables. As we aimed at the beginning we want to reduce the number of scheduling parameters by reducing the dimension of this vector.

Constants

Parameter	Value $\times 10^{-2}$	Parameter	Value $\times 10^{-2}$
b_1	2.73 kgm^2	b_{10}	1.35 kgm^2
b_2	-1.35 kgm^2	b_{11}	4.41 kgm^2
b_3	2.40 kgm^2	b_{12}	-3.75 kgm^2
b_4	6.81 kgm^2	b_{13}	0.0187 Nms/rad
b_5	-3.06 kgm^2	b_{14}	1.18 Nms/rad
b_6	13.35 kgm^2	b_{15}	0.27 Nms/rad
b_7	-1.25 kgm^2	b_{16}	66.6 Nm
b_8	2.30 kgm^2	b_{17}	244.0 Nm
b_9	-0.10 kgm^2		

Table 3.1: Physical parameters of nonlinear model of gyroscope [4]

3.2 LPV Model of Gyroscope

As we aimed in the beginning we want to convert the nonlinear model to the general LPV model defined as:

$$\begin{cases} \dot{x}(t) = A(\rho(t))x + B(\rho(t))u \\ y(t) = C(\rho(t))x + D(\rho(t))u \end{cases} \tag{3.3}$$

where x is the vector of states, y is the output, and u is the input for the system defined as follows:

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T$$

As for the scheduling parameter vector ρ we take 3.2 as explained in the previous section. Again for simplicity we will drop the index t as a representative to show the dependency on time. Therefore again starting with the nonlinear model 3.1, this time we have labelled the different parts of the model to explain better the whole procedure of representing it as the LPV model.

$$\underbrace{\begin{bmatrix} b_1 & 0 & b_1c_2 & b_1s_2c_3 \\ 0 & b_3 & 0 & -b_3s_3 \\ b_1c_2 & 0 & b_2s_2^2 + b_4 & -b_2s_2c_2c_3 \\ b_1s_2c_3 & -b_3s_3 & -b_2s_2c_2c_3 & -b_2s_2^2c_3^2 + b_5s_3^2 + b_6 \end{bmatrix}}_{\text{N}} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \\ \ddot{q}_4 \end{bmatrix} + \underbrace{\begin{bmatrix} b_{13}\dot{q}_1 \\ b_{14}\dot{q}_2 \\ b_{15}\dot{q}_3 \\ b_{15}\dot{q}_4 \end{bmatrix}}_{\text{L}}$$

$$+ \underbrace{\begin{bmatrix} k_1(\dot{q}, q) \\ k_2(\dot{q}, q) \\ k_3(\dot{q}, q) \\ k_4(\dot{q}, q) \end{bmatrix}}_{\text{E}} = \underbrace{\begin{bmatrix} b_{16} & 0 \\ 0 & b_{17} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\text{F}} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Therefore we can rewrite the nonlinear model 3.1 in the block matrix form as follows:

$$N\ddot{q} + S\dot{q} + E = Fu$$

where matrix S is simply defined as the diagonal of the vector $\begin{bmatrix} b_{13} & b_{14} & b_{15} & b_{15} \end{bmatrix}^T$ coming from vector L as follows:

$$S = \begin{bmatrix} b_{13} & 0 & 0 & 0 \\ 0 & b_{14} & 0 & 0 \\ 0 & 0 & b_{15} & 0 \\ 0 & 0 & 0 & b_{15} \end{bmatrix}$$

The idea is to partition the vector E and try to rewrite it in the form of:

$$E = M(\rho)q + W(\rho)\dot{q}$$

Although there is not a unique way to do so, but we have tried two or three different partitioning and came up with the best possible match between the nonlinear model and the LPV model. The key point in this step is that while partitioning we have tried to make as many elements of $A(\rho)$ nonzero as possible. Here is our proposed way to partition the matrix E:

$$\begin{aligned} k_1(\dot{q}, q) = &-(b_1\dot{q}_2\dot{q}_3q_2)\frac{s_2}{q_2} \\ &- (b_1s_2\dot{q}_3\dot{q}_4q_3)\frac{s_3}{q_3} \\ &+ (b_1c_2c_3\dot{q}_2)\dot{q}_4 \end{aligned}$$

$$\begin{aligned}
k_2(\dot{q}, q) &= ((b_1\dot{q}_1\dot{q}_3 + b_2c_2c_3^2\dot{q}_4^2 - b_2c_2\dot{q}_3^2 - 2b_7s_2c_3\dot{q}_3\dot{q}_4)q_2)\frac{s_2}{q_2} \\
&- (b_1c_2c_3\dot{q}_4)\dot{q}_1 \\
&+ (-b_8c_3\dot{q}_4 + b_7c_3\dot{q}_4)\dot{q}_3 \\
&+ (2b_9c_2^2c_3\dot{q}_3)\dot{q}_4
\end{aligned}$$

$$\begin{aligned}
k_3(\dot{q}, q) &= ((-b_1\dot{q}_1\dot{q}_2 - 2b_{10}c_2\dot{q}_2\dot{q}_3)q_2)\frac{s_2}{q_2} \\
&+ ((b_1s_2\dot{q}_1\dot{q}_4 + b_{11}c_3\dot{q}_4^2 - b_{10}c_2^2c_3\dot{q}_4^2)q_3)\frac{s_3}{q_3} \\
&+ ((b_8 + b_7)c_3\dot{q}_4)\dot{q}_2 \\
&+ (2b_{10}c_2^2c_3\dot{q}_2)\dot{q}_4
\end{aligned}$$

$$\begin{aligned}
k_4(q, \dot{q}) &= ((b_2s_3c_2\dot{q}_3^2 + 2b_{10}c_2c_3^2\dot{q}_2\dot{q}_4)q_2)\frac{s_2}{q_2} \\
&+ ((-b_1s_2\dot{q}_1\dot{q}_3 - 2b_{11}c_3\dot{q}_3\dot{q}_4 + 2b_{10}c_2^2c_3\dot{q}_3\dot{q}_4)q_3)\frac{s_3}{q_3} \\
&+ (b_1c_2c_3\dot{q}_2)\dot{q}_1 \\
&+ (2b_{10}c_2^2c_3\dot{q}_3)\dot{q}_2 \\
&+ (b_{12}c_3\dot{q}_2)\dot{q}_3
\end{aligned}$$

Hence we obtain the matrices $M(\rho)$ and $W(\rho)$ as follows:

$$M(\rho) = \begin{bmatrix} 0 & M_{12} & M_{13} & 0 \\ 0 & M_{22} & 0 & 0 \\ 0 & M_{32} & M_{33} & 0 \\ 0 & M_{42} & M_{43} & 0 \end{bmatrix}$$

$$W(\rho) = \begin{bmatrix} 0 & 0 & 0 & W_{14} \\ W_{21} & 0 & W_{23} & W_{24} \\ 0 & W_{32} & 0 & W_{34} \\ W_{41} & W_{42} & W_{43} & 0 \end{bmatrix}$$

where the nonzero entries are defined as:

$$M_{12} = -(b_1\dot{q}_2\dot{q}_3)\frac{s_2}{q_2}$$

$$M_{13} = -(b_1 s_2\dot{q}_3\dot{q}_4)\frac{s_3}{q_3}$$

$$M_{22} = (b_1\dot{q}_1\dot{q}_3 + b_2c_2c_3^2\dot{q}_4^2 - b_2c_2\dot{q}_3^2 - 2b_7s_2c_3\dot{q}_3\dot{q}_4)\frac{s_2}{q_2}$$

$$M_{32} = (-b_1\dot{q}_1\dot{q}_2 - 2b_{10}c_2\dot{q}_2\dot{q}_3)\frac{s_2}{q_2}$$

$$M_{33} = (b_1s_2\dot{q}_1\dot{q}_4 + b_{11}c_3\dot{q}_4^2 - b_{10}c_2^2c_3\dot{q}_4^2)\frac{s_3}{q_3}$$

$$M_{42} = (b_2s_3c_2\dot{q}_3^2 + 2b_{10}c_2c_3^2\dot{q}_2\dot{q}_4)\frac{s_2}{q_2}$$

$$M_{43} = (-b_1s_2\dot{q}_1\dot{q}_3 - 2b_{11}c_3\dot{q}_3\dot{q}_4 + 2b_{10}c_2^2c_3\dot{q}_3\dot{q}_4)\frac{s_3}{q_3}$$

$$W_{14} = b_1c_2c_3\dot{q}_2$$

$$W_{21} = -b_1c_2c_3\dot{q}_4$$

$$W_{23} = -b_8c_3\dot{q}_4 + b_7c_3\dot{q}_4$$

$$W_{24} = 2b_9c_2^2c_3\dot{q}_3$$

$$W_{32} = (b_8 + b_7)c_3\dot{q}_4$$

$$W_{34} = 2b_{10}c_2^2c_3\dot{q}_2$$

$$W_{41} = b_1c_2c_3\dot{q}_2$$

$$W_{42} = 2b_{10}c_2^2c_3\dot{q}_3$$

$$W_{43} = b_{12}c_3\dot{q}_2$$

The curious reader should verify the way of separation of vector E. Again going back to the nonlinear model we can write equation 3.1 in the form of:

$$N\ddot{q} + (S + W)\dot{q} + Mq = Fu$$

where to make it more clear we have also dropped the vector ρ which shows the dependency on scheduling parameters (as we do it in the below also). Now if we define the vector of state x to be equal to

$$x = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & \dot{q}_1 & \dot{q}_2 & \dot{q}_3 & \dot{q}_4 \end{bmatrix}^T$$

we can finally write the nonlinear model in the form of LPV system 3.3 with the state-space matrices defined as follows. Note that we have chosen the matrices C and D arbitrarily to be equal to identity matrix and zero matrix, respectively.

$$A = \begin{bmatrix} I_{4\times 4} & 0 \\ 0 & N \end{bmatrix}^{-1} \begin{bmatrix} 0 & I_{4\times 4} \\ -M & -(S + W) \end{bmatrix}$$

$$B = \begin{bmatrix} I_{4\times 4} & 0 \\ 0 & N \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ F \end{bmatrix}$$

$$C = I_{8\times 8}$$

$$D = 0_8$$

where $I_{n\times n}$ denotes the $n \times n$ identity matrix and 0_8 denotes the zero vector of dimension 8.

Therefore we can achieve the LPV model of the gyroscope and show it in the compact form by

$$Q(\rho(t)) = \begin{bmatrix} A(\rho(t)) & B(\rho(t)) \\ C(\rho(t)) & D(\rho(t)) \end{bmatrix}$$

Chapter 4

Preimage Problem

This chapter illustrates the main idea of preimaging the data points in feature space into input space. This will also explain how to derive the reduced LPV model with the desired number of reduced parameters. Note that at this position we have already trained our KPCA algorithm. By this, we mean we have extracted the principal axes in the feature space and our algorithm has already learned what are the best principal axes to use them for projecting the data points. The KPCA algorithm then will give us as the result the reduced data points in the feature space. Now the question is how to go back from feature space to input space since we want to obtain the set of reduced points in the original space where the data points exist. Section 4.1 discusses about this problem.

Our goal, however, is to obtain a reduced LPV model which is in affine form. To this end we just need to be able to construct a reduced LPV model which gives a similar response as the full order LPV model given the same input to the system. Section 4.2 introduces an alternative way to solve this problem of finding the reduced LPV model by introducing an optimization problem which is a proposed way in [6]. This optimization tries to minimize the distance between state-space matrices of the full order LPV model and its reduced one. As we have already seen the full order LPV model has 6 scheduling parameters whereas the number of parameters in the reduced model can vary from 1 to 6 according to user's desire based on the magnitude of the eigenvalues of kernel matrix or considering the table of the accuracy of the reduced LPV model which we will also present in the next chapter.

4.1 Problem Statement

Using the KPCA algorithm we were able to reduce the dimension of mapped data points by projecting them onto the space constructed by principal axes coming form kernel matrix K. This projection takes place in the feature space and not in the input space where the data points originally exist. What is remained is to map this reduced points into the original input space. Indeed, we still don't know where are the positions of the reduced data points in the input space.

To solve the problem of preimaging we need to have a general understanding of the exact connection between input space and feature space. Therefore here we just want to get an intuition of what exactly we mean by preimaging problem. Suppose we have performed the KPCA algorithm and as a solution we have obtained the projected points in feature space. These points can be written as a linear combination of mapped input data points via the feature map ϕ since as we already know the points $\phi(x_i)$ construct an inner product space and make a basis for our Hilbert space. Let us consider only one of them and denote it by z. Therefore we can write it as:

$$z = \sum_{i=1}^{m} \alpha_i \phi(x_i) \tag{4.1}$$

where again $x_1, \ldots, x_m \in \mathbb{R}^n$ are data points and $\alpha_i \in \mathbb{R}$.

To find the exact preimage of such a point, we need to find a point $x \in \mathbb{R}^n$

such that we have $\phi(x) = z$. In other words, we require the point z in the equation 4.1 to have a preimage under map ϕ. But since the map ϕ is usually nonlinear we cannot generally assure the existence of such preimages. Schölkopf in [7] has suggested a way to find the *approximate preimage* $\tilde{x}$ instead of the exact one. The idea is to solve an optimization problem of the form:

$$\min_{x} \; \|z - \phi(x)\|$$

which tries to find an approximate preimage $\tilde{x}$ for the point z in the way that $\phi(\tilde{x})$ is in the minimal distance from z.

In our case, however, if we want to find the approximate preimage for each single projected point in every sampling time t in real time using this optimization problem, we might face some practical restrictions since this approach needs some hardware with heavy computational power while simulating. As an example of investigating this method we refer the reader to [18]. We will see in the next section a remedy for this problem which helps us towards our goal, namely finding the reduced LPV model for the gyroscope.

4.2 Reduced LPV Model of Gyroscope

To find the reduced LPV model of gyroscope we can try to solve an optimization problem proposed in [6] which minimizes the distance between the state space matrices of full order LPV model of the gyroscope and its reduced LPV model. Using this optimization we can actually avoid preimaging problem and directly obtain our reduced LPV model. We aim to solve an optimization problem of the form

$$\min_{R_i, i=0,1,\ldots,\ell} \frac{1}{m}\sum_{t=1}^{m} \|Q(\rho(t)) - R(\theta(t))\|_F^2 \tag{4.2}$$

where $Q(\rho(t))$ is as defined in section 3.2 and for an affine reduced model we have

$$R(\theta(t)) = R_0 + \sum_{i=1}^{n} R_i\theta_i(t)$$

We have denoted the vector of reduced parameters in the reduced LPV system with $\theta(t)$. Moreover, m is the total number of time samples (data points) and $\|\cdot\|_F$ represents the Frobenius norm defined as $\|A\|_F = \sqrt{Tr(AA^H)}$ where A^H is the conjugate transpose of matrix A.

The type of optimization problem which we are dealing with is nonlinear (quadratic) unconstrained optimization. According to the theorem mentioned below, if we prove the convexity of the cost function and can find a local minimum for this problem, we can deduce the existence of the global minimum for this optimization problem. A proof of this theorem can be found in any textbook on optimization (e.g [19]).

Theorem 4.1. *Let $\mathcal{X}$ be a normed space, $C \subseteq \mathcal{X}$ convex, and $f : C \to \mathbb{R}$ be a convex function. If x^0 is a local minimum of f on C, then x^0 is a global minimum of f on C.*

Therefore we just need to prove the convexity of the cost function and convexity of the set from which the matrices R_i are taken from. The set of matrices is trivially convex since we didn't put any constraint on them. It is the set of all matrices in $\mathbb{R}^{16\times 10}$. We only need to prove the convexity of our cost function

$$J(R_1, \ldots, R_\ell) = \frac{1}{m}\sum_{t=1}^{m} \|Q(\rho(t)) - R(\theta(t))\|_F^2 \tag{4.3}$$

First let us state some basic facts which help us prove the convexity of cost function.

- Sum of convex functions is convex, i.e if f_i, $i = 1, \dots, \ell$ are convex therefore $f = \sum_{i=1}^{\ell} f_i$ is convex.
- Square of non-negative convex function is convex, i.e. if f is non-negative and convex therefore f^2 is also convex

It turns out that all we need to do is to prove the convexity of the following function: (for simplicity we will drop the time index t)

$$f(R_1, \dots, R_\ell) = \|Q(\rho) - (R_0 + \sum_{i=1}^{\ell} R_i\theta_i)\|_F \tag{4.4}$$

To this end we define

$$\hat{\theta} = \begin{bmatrix} I & \theta_1 I & \dots & \theta_\ell I \end{bmatrix}$$

$$\hat{X} = \begin{bmatrix} R_0 & R_1 & \dots & R_\ell \end{bmatrix}^T$$

where I denotes the identity matrix. Therefore we can rewrite function 4.4 in the block matrix form as follows

$$f(\hat{X}) = \|Q(\rho) - \hat{\theta}\hat{X}\|_F \tag{4.5}$$

We want to prove: if f is given in the form of 4.5 then

$$f(\alpha\hat{X}^1 + (1-\alpha)\hat{X}^2) \leq \alpha f(\hat{X}^1) + (1-\alpha) f(\hat{X}^2)$$

for any $\hat{X}^1, \hat{X}^2 \in \mathbb{R}^{16\ell\times 10}$ and any $\alpha \in [0, 1]$

Proof.

$$\begin{aligned}
f(\alpha\hat{X}^1 + (1-\alpha)\hat{X}^2) &= \|Q(\rho) - (\alpha\hat{\theta}\hat{X}^1 + (1-\alpha)\hat{\theta}\hat{X}^2)\|_F \\
&= \|Q(\rho) + \alpha Q(\rho) - \alpha Q(\rho) - (\alpha\hat{\theta}\hat{X}^1 + (1-\alpha)\hat{\theta}\hat{X}^2)\|_F \\
&= \|\alpha Q(\rho) - \alpha\hat{\theta}\hat{X}^1 + (1-\alpha)Q(\rho) - (1-\alpha)\hat{\theta}\hat{X}^2)\|_F \\
&\leq \|\alpha(Q(\rho) - \hat{\theta}\hat{X}^1)\|_F + \|(1-\alpha)(Q(\rho) - \hat{\theta}\hat{X}^2)\|_F \\
&= \alpha f(\hat{X}^1) + (1-\alpha)f(\hat{X}^2)
\end{aligned}$$

where in the second line we have added and subtracted $\alpha Q(\rho)$ and in the second last line we have used the triangle inequality property of the norms. □

Putting it altogether we conclude that the cost function 4.3 is convex and if we can find a local minimum for the optimization problem 4.2, we can then assure the existence of the global minimum for this problem. Therefore by solving this optimization problem we can obtain the reduced LPV model of the gyroscope which represents the same underlying behaviour as LPV model but with fewer number of parameters.

Chapter 5

Performance

This chapter describes the performance of the reduced LPV model of the gyroscope discussed in the previous chapter. We start with introducing a proposed measure in [13] for the accuracy of such systems. We will use this measure to see the performance of our reduced LPV model while utilizing PCA algorithm and KPCA algorithm with the different kernel functions and also for different number of reduced parameters.

5.1 Accuracy of the Reduced LPV Model

Accuracy of the reduced LPV model can be measured from the fraction of total data variation defined as

$$a(\ell) = \frac{\sum\limits_{i=1}^{\ell} \lambda_i}{\sum\limits_{i=1}^{m} \lambda_i}$$

where again ℓ is the dimension of reduced scheduling parameters, λ_i is the ith eigenvalue of the centered kernel matrix $\tilde{K}$, and m is the total number of time samples (data points). As for the kernel function we have considered two different functions, namely a Gaussian function (radial basis function) and a polynomial function.

Note that according to our earlier definition of kernel functions these two are valid kernel functions. We just mention that using these functions, the entry K_{ij} of the kernel function can be computed as follows:

- For a Gaussian function: $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{\sigma})$
  with σ being the only kernel function parameter.
- For a polynomial function: $k(x_i, x_j) = \langle x_i, x_j \rangle^d = (x_i^T x_j)^d$
  with d as the kernel function parameter

We have plotted the accuracy for the reduced LPV model of the gyroscope achieved in the previous chapter as a function of the number of reduced parameters and kernel parameters. The results in the case of the two different kernel functions (Gaussian and polynomial) are shown in figures 5.1 and 5.2, respectively. In the case of Gaussian kernel function observe that if we reduce the number of scheduling parameters from $\ell = 6$ to $\ell = 3$ still we can have a high accuracy, say more than 90%, if we work with a quite large Gaussian parameter σ. In the case of polynomial kernel, however, for all the number of reduced parameters ℓ, the reduced LPV model preserves a high accuracy although for the polynomial of degree 2 it shows the least accuracy.

To see more details in the case of Gaussian kernel function we have also plotted the accuracy as a function of number of reduced parameters ℓ for different but fixed Gaussian parameters (Fig. 5.3). This explains if we use a small Gaussian parameter σ we cannot preserve the accuracy of the reduced LPV model very high. Therefore while using Gaussian to train and test our KPCA algorithm, we will use a large value for σ, say 100. This shows the importance of tuning the kernel parameters to achieve the desired results.

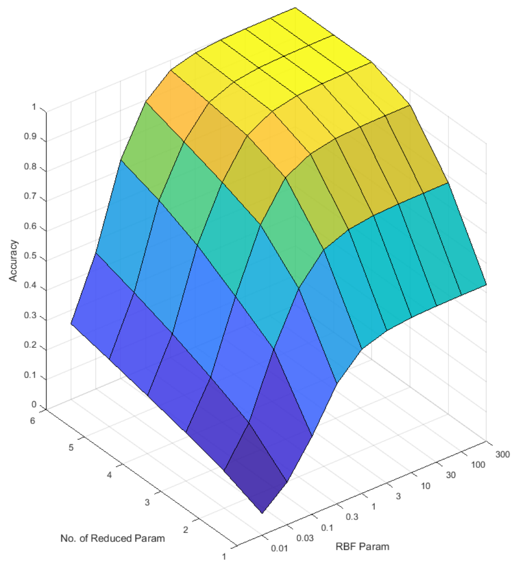


Figure 5.1: Accuracy with respect to the number of scheduling variables ℓ and Gaussian function parameter σ

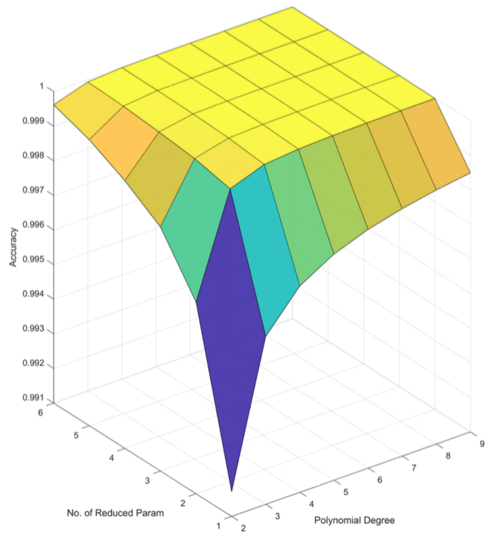


Figure 5.2: Accuracy with respect to the number of scheduling variables ℓ and polynomial kernel degree d

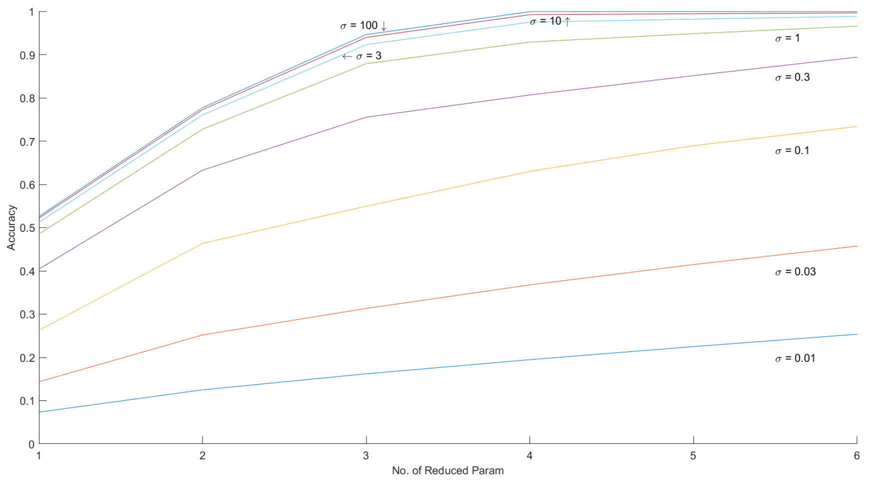


Figure 5.3: Accuracy with respect to the the number of reduced parameters ℓ for different choices of Gaussian parameter σ

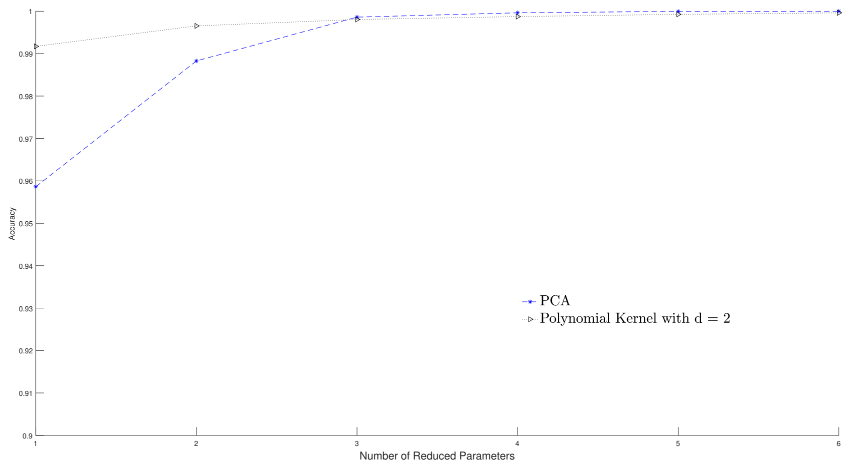


Figure 5.4: Plot of accuracy as function of number of reduced parameters ℓ for PCA and KPCA with polynomial kernel

As a comparison between the PCA and the KPCA methods, we have shown the accuracy of the reduced LPV model with respect to the number of reduced parameters for PCA and KPCA with polynomial kernel function (Fig 5.4). As we can see, dimensionality reduction using polynomial kernels results in a higher accuracy than using the PCA algorithm as we expected. This supports the idea that KPCA algorithms are a more powerful method than PCA while performing dimensionality reduction.

The detailed results for PCA and KPCA with two kernel functions Gaussian and polynomial are listed in table 5.1, in terms of percentage. It is important to note that the fraction given in the beginning of this section, is

Kernel Function	Parameters	Accuracy(%)					
		$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$
PCA		95.85	98.83	99.86	99.96	100	100
	$\sigma = 0.01$	07.35	12.51	16.22	19.48	22.51	25.37
	$\sigma = 0.03$	14.38	25.20	31.33	36.78	41.47	45.74
	$\sigma = 0.1$	26.28	46.38	54.96	63.01	68.96	73.39
Radial	$\sigma = 0.3$	40.43	63.31	75.54	80.68	85.14	89.42
Basis	$\sigma = 1$	48.54	72.78	87.90	92.90	94.84	96.59
Kernel	$\sigma = 3$	51.23	76.06	92.32	97.50	98.20	98.83
Function	$\sigma = 10$	52.22	77.28	93.98	99.24	99.45	99.65
(Gaussian)	$\sigma = 30$	52.50	77.64	94.47	99.74	99.82	99.88
	$\sigma = 100$	52.60	77.76	94.64	99.92	99.94	99.96
	$\sigma = 300$	52.63	77.80	94.69	99.97	99.98	99.99
	$d = 2$	99.17	99.65	99.81	99.87	99.93	99.96
	$d = 3$	99.59	99.95	99.97	99.98	99.99	99.99
	$d = 4$	99.70	99.98	99.99	100	100	100
Polynomial	$d = 5$	99.75	99.99	100	100	100	100
Kernel	$d = 6$	99.79	99.99	100	100	100	100
	$d = 7$	99.81	100	100	100	100	100
	$d = 8$	99.83	100	100	100	100	100
	$d = 9$	99.85	100	100	100	100	100

Table 5.1: Accuracy of the reduced LPV model of gyroscope in percentage

calculating the accuracy only based on training data examples. We never know what happens if we start testing the trained KPCA algorithm using unseen patterns. Therefore the numbers given in this section are only calculated based on the training data points, and we cannot expect the exact same results if we apply our trained PCA or KPCA algorithm to new previously unseen data points. However this helps us to decide the number of reduced scheduling parameters before constructing the reduced LPV model and tune the kernel function parameter beforehand so that we can expect the accuracy close to the accuracy given here.

Chapter 6

Implementation and Numerical Results

This chapter depicts a full image of all the results coming from our work. Note that at this point we are going to test our trained KPCA algorithm on the new previously unseen data points. Therefore we can actually see how well our trained algorithm is performing when it comes to reducing the dimension of test data points being produced under different inputs.

Section 6.1 describes the way of constructing the simulation model to gain the LPV model and reduced LPV model out of their state-space matrices which we discussed in previous chapters. First, the behaviour of the full order LPV model of the gyroscope is depicted in comparison with its nonlinear model to observe how well our proposed way to construct the LPV model is working (secion 6.2). Second, we observe the results of the simulation to see how well the reduced LPV model behaves compared to full order LPV model of the gyroscope. We will see the performance for the reduced LPV model of the gyroscope with different numbers of parameters for two kernel functions Gaussian and polynomial. (section 6.3)

6.1 Simulation

After obtaining the LPV model and reduced LPV model of the gyroscope, i.e the corresponding state-space matrices, now we want to simulate them to be able to compare their behaviours to see how well our proposed models are performing. To this end, first we need to run these two models over a fine grid across the range of the original and reduced parameters to get the different samples of the state-space matrix over different nodes of this grid.

The range of operation of original scheduling variables, i.e. the limits of the angles and angular velocities q_2, q_3, $\dot{q}_1$, $\dot{q}_2$, $\dot{q}_3$, and $\dot{q}_4$, for a 4 degree of freedom control moment gyroscope (4-DOF CMG) are taken from [4] and are listed in the table 6.1. However, to see the range of operation of reduced scheduling parameters we should first perform the dimensionality reduction to obtain these parameters and then observe the interval in which they are being produced to define the corresponding range.

Variables	Range
q_2	$[-25, \dots, 25]$ $[°]$
q_3	$[-75, \dots, 75]$ $[°]$
$\dot{q}_1$	$[30, \dots, 65]$ $[rad/s]$
$\dot{q}_2$	$[-2, \dots, 2]$ $[rad/s]$
$\dot{q}_3$	$[-2, \dots, 2]$ $[rad/s]$
$\dot{q}_4$	$[-2, \dots, 2]$ $[rad/s]$

Table 6.1: Operation interval of scheduling variables of a 4-DOF CMG [4]

What is crucial in the process of simulation is that we should try keeping the scheduling variables inside their range of operation by choosing the correct input. To this end, a lot of investigations were done in order to choose a zero mean input excitation signal such that the scheduling parameters remain inside the bounds defined by the compact set $\mathcal{P}$.

Everything we have done so far can be categorized in the process of training the KPCA algorithm. Now we want to see the performance of the trained KPCA algorithm on the new previously unseen set of data points (new vectors of scheduling variables in different time samples). We now explain exactly what we want to do in order to be able to test the performance of our trained KPCA algorithm.

Once we simulated the original full order LPV system, we could collect the training data points as the vectors of scheduling variables. Then we implemented the KPCA algorithm on this training data to gain the set of reduced data over a finite time. In this step itself, we obtained the kernel matrix and the corresponding eigenvalues and eigenvectors. Now that we have these tools, namely kernel matrix and the corresponding eigenvectors, the algorithm has already learned how to reduce the dimension of the new data points in the way that after reducing the dimension we will not lose much useful information. This means the KPCA algorithm has found the principal axes on which we can project our new data points. Therefore all we need to do is to simulate the reduced LPV model to gain the new set of original scheduling parameters (test data points) and then again run the KPCA with the learned eigenvectors to obtain the new reduced scheduling variables (reduced test data points) in each time sample.

6.2 Nonlinear Model vs Full Order LPV Model

The plot of the 8 states of the nonlinear model of the gyroscope in contrast to its LPV model is depicted in the Fig. 6.1, for the first 100 seconds. To make it clear, we have denoted the angular velocities with w_1, w_2, w_3, and w_4 instead of our previous notations $\dot{q}_1$, $\dot{q}_2$, $\dot{q}_3$ and $\dot{q}_4$, respectively. We see that the behaviour of the LPV model is following the path of nonlinear model reasonably good, and it is not diverging. This is a good sign for us since it assures us that our LPV model is behaving close to the nonlinear one and we can collect the set of training data points more accurately. Hence we can train the KPCA algorithm more precisely so that the reduced LPV model will also behave closely to the nonlinear model.

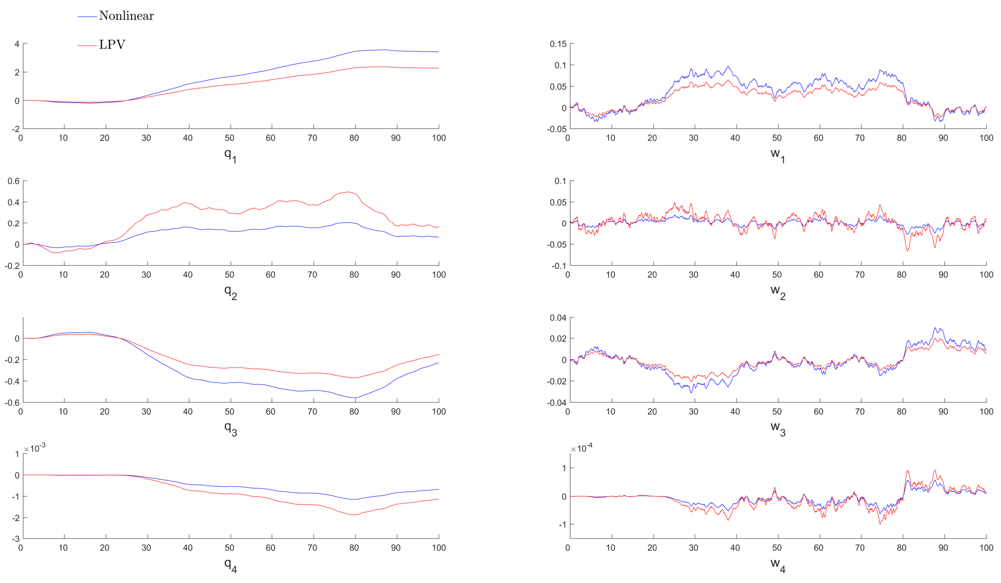


Figure 6.1: States of the Gyroscope full-order LPV model and nonlinear model

6.3 Full Order LPV Model vs Reduced LPV Model

The plot of the 8 states of the full order LPV model of the gyroscope as well as its reduced LPV model for two different kernel functions Gaussian and polynomial are shown in the figures 6.2 and 6.3, respectively, for the first 10 seconds. Again to make it more clear, we have denoted the angular velocities with w_1, w_2, w_3, and w_4 instead of our previous notations $\dot{q}_1$, $\dot{q}_2$, $\dot{q}_3$, and $\dot{q}_4$, respectively. For computational reasons, we used a subset of 7000 training examples (7000 time samples of vector of scheduling parameters) to compute the kernel matrix K and therefore to reduce the dimension of data points. In the case of Gaussian function we have plotted the behaviour of the reduced LPV model with $\ell = 1, 2, \ldots,$ and 6 reduced parameters. Whereas for polynomial kernel we have just shown the results for $\ell = 1$, 3, and 5.

As for the reduced LPV model obtained via utilizing Gaussian kernel we have set the Gaussian parameter σ equal to 100. It turns out that the dynamical behaviour for the reduced LPV system with $\ell = 6$ parameters (green line) is close to the full order LPV model (red line) of gyroscope as it was expected according to the table 5.1 presented in section 5.1. However note that the behaviour of the reduced LPV model of the gyroscope with $\ell = 1$ parameter (black line) is diverging in some states such as q_4 or w_4. Observe that the reduced LPV model with $\ell = 3$ parameters (blue line) is also following the path of the full order LPV model of gyroscope reasonably good for all states. This shows us that our proposed reduced LPV model of gyroscope is performing well even if we reduce the number of scheduling parameters from 6 to 3 (50%).

In the case of polynomial kernels we have chosen the degree of the polyno-

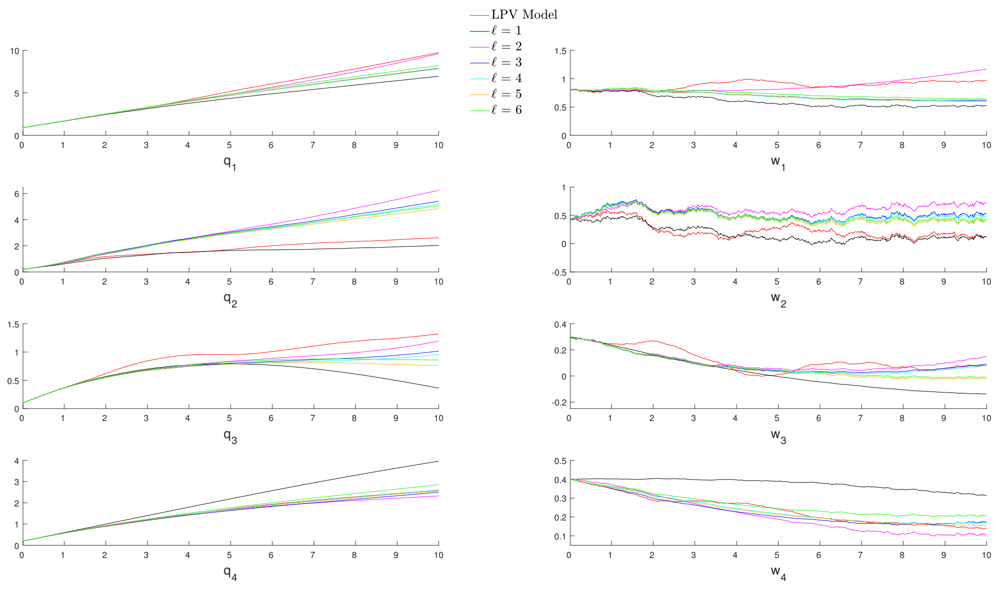


Figure 6.2: States of the gyroscope full-order LPV model and reduced LPV model based on Gaussian kernel for $\ell = 1, 2, \ldots, 6$

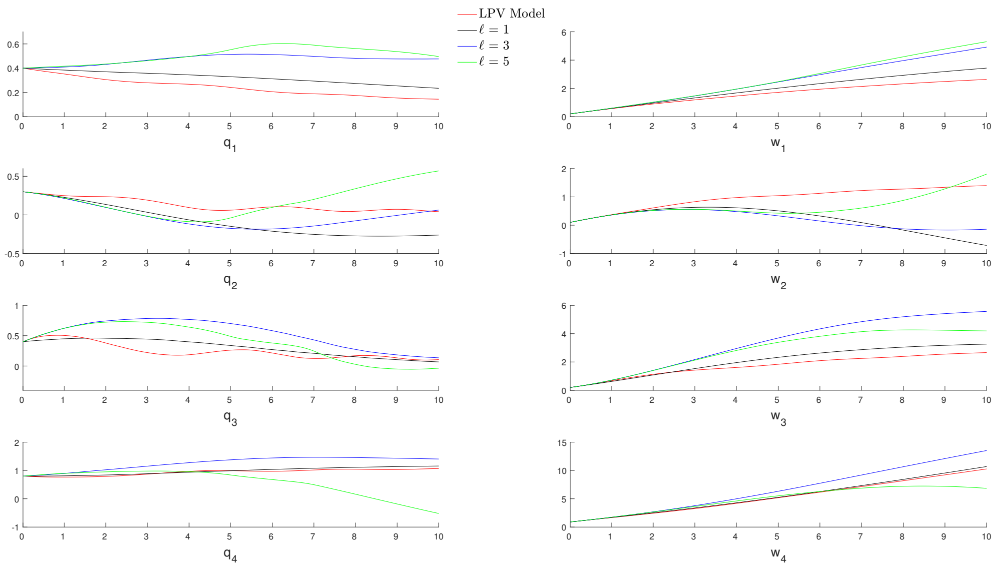


Figure 6.3: States of the gyroscope full-order LPV model and reduced LPV model based on polynomial kernel for $\ell = 1$, 3, and 5

mial d to be equal to 2. In this case we see that the reduced LPV model with 3 reduced parameters (blue line) is following the path of the full order LPV model (red line) with a high accuracy as we expected. Here the reduced LPV model with 1 parameter (black line) is also performing well and stays close to the LPV model almost in all the states. This is evidence for the accuracy explained before for the reduced LPV systems coming from polynomial kernels. This is very important since in this case we can reduce the number of scheduling parameters in the LPV model of gyroscope from 6 to 1 and still maintain high accuracy.

Chapter 7

Conclusion

In this thesis we applied the PCA algorithm and its kernelized version KPCA for the purpose of reducing the number of scheduling variables in a LPV system of the gyroscope. To obtain the LPV model of the gyroscope we proposed a way for the separation of its nonlinear model and converted the nonlinear model to a LPV model successfully using this approach.

Having the LPV model, we simulated this model to compare the dynamical behaviour of this model with the nonlinear one. We simulated the LPV model over a fine grid across the valid range of scheduling parameters. The results showed us the two system behave similarly. In this step we collected the data on which we want to perform the dimensionality reduction.

We accomplished the dimensionality reduction on the set of parameter vectors of the LPV model using the KPCA method. To train the KPCA algorithm with the collected training data points we utilized two different kernel functions, Gaussian and polynomial. We performed the dimensionality reduction on them and obtained the set of reduced parameter vectors.

In the next step, we tried to obtain the reduced LPV model of the gyroscope by solving a nonlinear, unconstrained optimization problem which minimizes the distance between state-space matrices of the LPV model and the reduced LPV model. We proved the convexity of this optimization problem. Therefore we concluded the existence of the global minimum in case of existence of the local minimum. The reduced LPV model was successfully made utilizing this optimization problem.

We also showed various plots of the accuracy of the reduced LPV model while considering different number of reduced parameters and different kernel functions. We compared the accuracy of this model for PCA-based reduction and KPCA-based reduction. These results helped us for tuning the kernel parameter and choosing the number of reduced parameters.

To check the dynamical behaviour of the reduced LPV model, we simulated this model and compared it with the LPV model. We tested our trained KPCA algorithm for two different kernel functions on the test data points (scheduling parameters). The results for different kernel functions and different number of reduced parameters were discussed case by case. For higher numbers of reduced parameters, the reduced LPV model was following the path of LPV model. Whereas, for lower numbers of the reduced parameters, in some cases, the results didn't show a high accuracy, as we expected.

All in all, the results coming from this thesis agrees with the success in the use of the KPCA algorithm for the purpose of reducing the number of scheduling variables in the LPV model of the gyroscope.

To summarize our work in this thesis, we will present an algorithmic way of how to perform the dimensionality reduction on the LPV model of the gyroscope.

Algorithm 3: Dimensionality reduction for the model of the gyroscope

```
1 Derive the LPV model of the gyroscope from its nonlinear model 3.1
    using the proposed way in section 3.2
2 Simulate the LPV model over a fine grid across the valid range of
    scheduling variables (Table 6.1) of the LPV model
3 Collect the data points (vectors of scheduling variables)
4 Choose the number of training data points $m$, the kernel function $k$
    with corresponding kernel parameter, and the desired number of
    reduced parameters $\ell$ based on significant eigenvalues or based on the
    accuracy table 5.1
5 Train KPCA algorithm (Alg. 2) and reduce the number of scheduling
    variables (dimension of the training data points).
6 Solve the optimization problem 4.2 to gain the reduced LPV model
7 Construct the reduced LPV model in different grids across the range
    of reduced parameters
8 Test the trained KPCA algorithm to see the performance of the
    reduced LPV model with respect to the LPV model
```

Appendix

A. Proofs

Proof of Theorem 2.1 [1]

Proof of $\Leftarrow$

This is clear by the definition of the kernel function

Proof of $\Rightarrow$

We have to prove the following:

Given $\mathcal{X}$ and k, there exists a vector space $\mathcal{H}$ with a scalar product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$ for all $x_i, x_j \in \mathcal{X}$.

To this end, we introduce the Reproducing Kernel Hilbert Space (RKHS), a scalar product on this space and a corresponding feature mapping ϕ. As vector space we are going to use a space of functions. Consider a mapping $\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ (to the space of all real-valued functions on $\mathcal{X}$), defined as

$$x_i \mapsto k_{x_i} := k(x_i, \cdot)$$

That is, the point $x_i \in \mathcal{X}$ is mapped to the function $k_{x_i} : \mathcal{X} \to \mathbb{R}$, $k_{x_i}(x_j) = k(x_i, x_j)$. Now consider the images $\{k_{x_i} \mid x_i \in \mathcal{X}\}$ as a spanning set of a vector space. That is, we define the space $\mathcal{G}$ that contains all finite linear combinations of such functions:

$$\mathcal{G} := \{\sum_{i=1}^{r} \alpha_i k(x_i, \cdot) \mid \alpha_i \in \mathbb{R}, r \in \mathbb{N}, x_i \in \mathcal{X}\}$$

We define a scalar product on $\mathcal{G}$ as follows. For the spanning functions we define

$$\langle k_{x_i}, k_{x_j} \rangle = \langle k(x_i, \cdot), k(x_j, \cdot) \rangle := k(x_i, x_j)$$

For general functions in $\mathcal{G}$ the scalar product is then given as follows. If

$$g = \sum_i \alpha_i k(x_i, \cdot) \text{ and } f = \sum_j \beta_j k(x_j, \cdot) \text{ then } \langle f, g \rangle_{\mathcal{G}} := \sum_{i,j} \alpha_i \beta_j k(x_i, x_j)$$

To make sure that this is really a scalar product, observe that this is well-defined (which is not so obvious because there might be several different linear combinations for the same functions) and it satisfies all properties of a scalar product. Finally, to make $\mathcal{G}$ a proper Hilbert space we need to take its topological completion $\bar{\mathcal{G}}$, that is we add all limits of Cauchy sequences. The resulting space $\mathcal{H} := \bar{\mathcal{G}}$ is called the reproducing kernel Hilbert space. By construction, it has the property that $k(x_i, y_j) = \langle \phi(x_i), \phi(x_j) \rangle$.
To see the reproducing property we need to prove the following: If

$$f = \sum_i \alpha_i k(x_i, \cdot)$$

Then

$$\langle f, k(x, \cdot) \rangle = f(x)$$

which is true since we have:

$$\begin{aligned} \langle k(x, \cdot), f \rangle &= \langle k(x, \cdot), \sum_i \alpha_i k(x_i, \cdot) \rangle \\ &= \sum_i \alpha_i \langle k(x_i, \cdot), k(x, \cdot) \rangle \\ &= \sum_i \alpha_i k(x_i, x) \\ &= f(x) \end{aligned} \tag{7.1}$$

□

Proof of Theorem 2.2 [1]

$$\begin{aligned} & Ka = \lambda a \\ \Longleftrightarrow\ & XX^T a = \lambda a \\ \Longleftrightarrow\ & X^T XX^T a = \lambda X^T a \\ \Longleftrightarrow\ & Cv = \lambda v \end{aligned}$$

If $v \neq 0$, then it is an eigenvector of C. $\square$

Proof of Theorem 2.3 [1]

Proof in several steps:

Step 1: non-zero eigenvectors of C are linear combinations of input points:

$\lambda v = Cv$ and $C = \sum_{j=1}^{m} x_j x_j^T$ implies

$v = \frac{1}{\lambda} Cv = \sum_{j=1}^{m} x_j x_j^T v = \frac{1}{\lambda} \sum_{j=1}^{m} x_j \langle x_j, v\rangle =: \sum_j a_j x_j$

Step 2: express eigvector of C as eigvector of K:

$$\begin{aligned} Cv = \lambda v \ &\Longleftrightarrow (\sum_{j=1}^{m} x_j x_j^T)((\sum_{i=1}^{m} a_i x_i) = \lambda \sum_{i=1}^{m} a_i x_i \\ &\Longleftrightarrow \sum_{i,j=1}^{m} x_j x_j^T a_i x_i = \lambda \sum_{i=1}^{m} a_i x_i \\ &\Longleftrightarrow x_s^T (\sum_{i,j} a_i x_j \langle x_j, x_i\rangle) = \lambda \sum_i a_i x_s^T x_i \\ &\Longleftrightarrow \sum_{i,j} a_i \langle x_s, x_j\rangle \langle x_j, x_i\rangle = \lambda \sum_i a_i \langle x_i, x_s\rangle \\ &\Longleftrightarrow (K^2 a)_s = \lambda (Ka)_s \end{aligned}$$

where in the third line we have multiplied by x_s^T for some s. Thus we obtain:

$v = \sum_i a_i x_i$ is the eigenvector of C if and only if

$$\begin{aligned} &\forall s : (K^2 a)_s = \lambda (Ka)_s \\ \iff &K^2 a = \lambda K a \\ \iff &Ka = \lambda a \end{aligned}$$

if and only if a is eigenvector of K with eigenvalue λ

Step 3: Normalization:

Let $a \in \mathbb{R}^m$ be an eigenvector of K with $\|a\| = 1$. Then the length of the corresponding eigenvector $v \in \mathbb{R}^n$ of C is given by

$$\begin{aligned} \|v\|^2 &= \|\sum_j a_j x_j\|^2 = \langle \sum_j a_j x_j, \sum_i a_i x_i \rangle = \sum_{i,j} a_i a_j \langle x_i, x_j \rangle = \sum_{i,j} a_i a_j k(x_i, x_j) \\ &= a^T K a = a^T \lambda a = \lambda \end{aligned}$$

To obtain a unit eigenvector v, we have to normalize the eigenvector a obtained from K with $\frac{1}{\sqrt{\lambda}}$. $\square$

References

[1] Ulrike von Luxburg. Lecture notes on machine learning, 2015.

[2] H. S. Abbas, A. Ali, S. M. Hashemi, and H. Werner. Lpv gain-scheduled control of a control moment gyroscope. In *2013 American Control Conference*, pages 6841–6846, June 2013.

[3] Julian Theis, Christian Radisch, and Herbert Werner. Self-scheduled control of a gyroscope. *IFAC Proceedings Volumes*, 47(3):6129 – 6134, 2014. 19th IFAC World Congress.

[4] C. Hoffmann and H. Werner. Lft-lpv modeling and control of a control moment gyroscope. In *2015 54th IEEE Conference on Decision and Control (CDC)*, June 2015.

[5] A. Kwiatkowski and H. Werner. Pca-based parameter set mappings for lpv models with fewer parameters and less overbounding. *IEEE Transactions on Control Systems Technology*, 16(4):781–788, July 2008.

[6] S. Z. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin. A kernel-based pca approach to model reduction of linear parameter-varying systems. *IEEE Transactions on Control Systems Technology*, 24(5):1883–1891, Sept 2016.

[7] B. Schölkopf and A. J. Smola. *Learning With Kernels.* W. H. Freeman, Cambridge, MA, USA: MIT Press, 2002.

[8] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Ratsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, Sep 1999.

[9] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[10] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[11] Herbert Werner. Lecture notes on advanced topics in control, 2016.

[12] T. R. Parks. *Manual for Model 750 Control Moment Gyroscope*. Educational Control Products, 1999.

[13] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[14] K. Hoffman and R.A. Kunze. *Linear algebra*. Prentice-Hall mathematics series. Prentice-Hall, 1971.

[15] S. Ghahramani. *Fundamentals of Probability with Stochastic Processes*. Pearson/Prentice Hall, 2005.

[16] *Introductory Functional Analysis with Applications*. Wiley classics library. Wiley India Pvt. Limited, 2007.

[17] Roland Tóth. *Modeling and identification of linear parameter-varying systems*, volume 403. Springer, 2010.

[18] S. Z. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin. Parameter set-mapping using kernel-based pca for linear parameter-varying systems*. *IEEE Transactions on Control Systems Technology*, pages 781–788, July 2014.

[19] David G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

Erklärung

Die vorliegende Arbeit habe ich selbständig verfasst und keine anderen als die angegebenen Hilfsmittel-insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen-benutzt. Die Arbeit habe ich vorher nicht in einem anderen Prüfungsverfahren eingereicht. Die eingereichte schriftliche Fassung entspricht genau der auf dem elektronischen Speichermedium.

Ort, Datum

Unterschrift