



Technische Universität München

Department of Mathematics



Bachelor's Thesis

# Feature Extraction for Nonlinear System Control Through Jointly Smooth Functions

Robert Schmidt

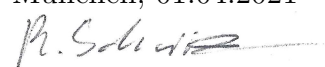
Supervisor: Prof. Dr. Hans-Joachim Bungartz

Advisor: Dr. Felix Dietrich

Submission Date: 01.04.2021

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

München, 01.04.2021

A handwritten signature in black ink, appearing to read 'R. Schmidt', with a horizontal line extending to the right.

Robert Schmidt

# Zusammenfassung

In dieser Arbeit werden wir einen aktuellen Ansatz der Merkmalfindung in dynamischen Systemen diskutieren. Er involviert eine Kernel Methode um glatte Funktionen auf beobachteten Mannigfaltigkeiten zu generieren. Diese Funktionen können dann angewendet werden um effektive Parameter in nichtlinearen dynamischen Systemen zu identifizieren, ohne dabei Wissen über das System zu benötigen. Desweiteren können sie verwendet werden um den Systemverlauf vorrauszusagen. Wir werden zwei häufig verwendete Steuermechanismen PID und MPC dazu verwenden um Daten von einem invertierten Pendel zu erhalten, auf welchen wir dann die Robustheit von unseren glatten Funktionen gegenüber Sensorrauschen und hochdimensionalen Inputdaten demonstrieren. Abschließend werden dann noch Daten von einer landenden simulierten Rakete analysiert um zu zeigen dass unsere Methoden auch auf komplexe Systeme anwendbar sind.

## Abstract

In this thesis we are going to discuss a recent approach to extract features from dynamic systems. It involves a kernel method to generate smooth functions on observed manifolds. Such functions can be used to identify effective parameters of nonlinear systems without knowledge of the dynamics and can be further combined to form a predictor for the system. We are going to generate data implementing PID and MPC controllers for an inverted pendulum, on which we will test the rigidity of our jointly smooth functions concerning noise and high dimensional input. In the end we will demonstrate the usability of our methods for complex dynamic systems by applying them to data from a landing spacecraft simulation.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>State of the Art</b>	<b>2</b>
2.1	Basic Controllers . . . . .	2
2.1.1	Proportional-Integral-Derivative Controller . . . . .	2
2.1.2	Model Predictive Control . . . . .	4
2.2	Spectral discovery of jointly smooth functions . . . . .	5
2.2.1	Kernel method . . . . .	6
2.2.2	Discovery of Smooth Functions over Manifolds . . . . .	7
<b>3</b>	<b>Feature Extraction Through Jointly Smooth Functions</b>	<b>9</b>
3.1	Toy Problem . . . . .	9
3.2	Control of an inverted pendulum . . . . .	11
3.2.1	Modelling of the inverted pendulum . . . . .	11
3.2.2	Controller Realization and Choice of Hyperparameters . . . . .	12
3.2.3	Jointly smooth functions over pendulum data . . . . .	15
3.3	Landing rockets . . . . .	18
3.3.1	The Simulation environment and obtaining data . . . . .	18
3.3.2	Extraction of jointly smooth functions . . . . .	19
<b>4</b>	<b>Conclusion</b>	<b>21</b>
	<b>References</b>	<b>22</b>

# 1 Introduction

As technology advances sensors get more and more compact, we can incorporate an increasing amount of them in confined dynamic systems in hope of achieving a higher quality of controllability. Therefore the acquisition of meaningful features from multi-modal data became a central problem in data analysis. In this thesis we will view the data as high-dimensional points residing on (multiple) low-dimensional manifolds and we will then operate in the space of real functions on these manifolds. In particular we will define functions in the span of the top eigenfunctions of the Laplacian, as smooth on said manifold. We will obtain jointly smooth functions by calculating the SVD of the union of spaces of real functions on the manifold. These functions are of interest as they represent the commonality of the observed manifolds and form a relationship between data sets of possibly different modalities.

In this thesis the use cases of jointly smooth functions (JSFs) in combination with dynamic systems will be discussed. We will review two commonly used control loops PID and MPC, so own data can be gathered from the highly unstable dynamic system of an inverted pendulum. We will present a method on how you can use JSFs in order to identify and obtain effective parameters of dynamic systems in a model-free manner. Further we will test the rigidity of JSFs to noise and high dimensional redundant data. Also a theoretical approach to obtain a predictor from the obtained functions will be presented. In the end we will demonstrate the real world feasibility by gathering meaningful JSFs from the complex dynamic system of a landing spacecraft simulation.

## 2 State of the Art

We start of by introducing commonly used controllers and also the algorithm we will use to analyse data from such controllers on the dynamic system of an inverted pendulum in the next chapter.

### 2.1 Basic Controllers

#### 2.1.1 Proportional-Integral-Derivative Controller

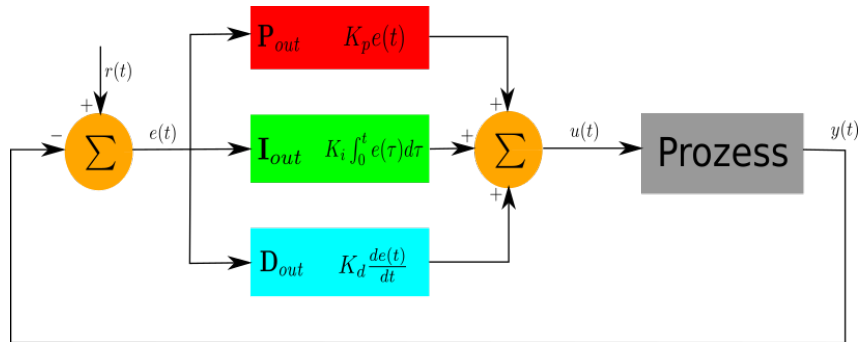


Figure 1: Visualisation of the PID control loop. The error  $e(t)$  is calculated as the difference between a setpoint  $r(t)$  and the system state  $y(t)$ . The control output  $u(t)$  is then calculated as a sum of the three terms  $P_{out}$ ,  $I_{out}$  and  $D_{out}$ .

The Proportional-Integral-Derivative Controller is a control loop mechanism using feedback. The goal is to stabilize a process variable  $x(t)$  at a target value of  $r(t)$ . The controller output  $u(t)$  is calculated by taking the error function  $e(t) := r(t) - x(t)$  in the three control terms into consideration. It is tuned by setting the parameters  $K_p$ ,  $K_i$  and  $K_d$ , hereby you can choose to not use one or more of the control terms by setting the corresponding parameter to zero, resulting in so called PI/PD/P/...-controllers. The control  $u$  of the system is then simply defined as:

$$u(t) := P_{out} + I_{out} + D_{out} = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (2.1)$$

with the tuning parameters  $K_p, K_i, K_d \in \mathbb{R}$ :

- $K_p$  proportional gain
- $K_i$  integral gain
- $K_d$  derivative gain

#### Proportional term

This term produces an output directly proportional to the error  $e(t)$ . It is adjusted by tuning the proportional gain  $K_p$ . If  $K_p$  is too large, the system might become unstable as the controller is overreacting. Contrary if  $K_p$  is too small the system is not responsive enough and the output might be too low to counteract the error inducing forces. The proportional term is usually the most dominant one of the three as it is directly connected

to the error function.

$$P_{out} := K_p e(t)$$

### Integral Term

The integral term increases in proportion the error and also the duration it has already persisted. So if initially the output is too small to reduce the error down to zero, then  $I_{out}$  will slowly rise to compensate. It is used to correct for any constant offset to the target. Because the integral term depends on accumulated errors and not the current error it can cause the system to overshoot the target initially.

$$I_{out} := K_i \int_0^t e(\tau) d\tau$$

### Derivative Term

The derivative term incorporates the slope of the error function. As such it aims to predict the trajectory of the system and therefore reducing the amount of overshooting, which results in damping and less oscillation of the error curve. As this term does not include the error itself, a pure D-controller can not bring the system to its target.

$$D_{out} := K_d \frac{de(t)}{dt}$$

### In practice

Because in practice we obtain discrete sensor data, the controller changes its output in set intervals (e.g. every 0.2sec). Hence the above formula can be simplified to:

$$u(t_n) = K_p e(t_n) + K_i \sum_{k=0}^n e(t_k) + K_d [e(t_n) - e(t_{n-1})] \quad (2.2)$$

Designing and tuning a proportional-integral-derivative (PID) controller appears to be conceptually intuitive, but can be hard in practice, if multiple (and often conflicting) objectives such as short transient and high stability are to be achieved. Usually, initial designs obtained by all means need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired[1].

### 2.1.2 Model Predictive Control

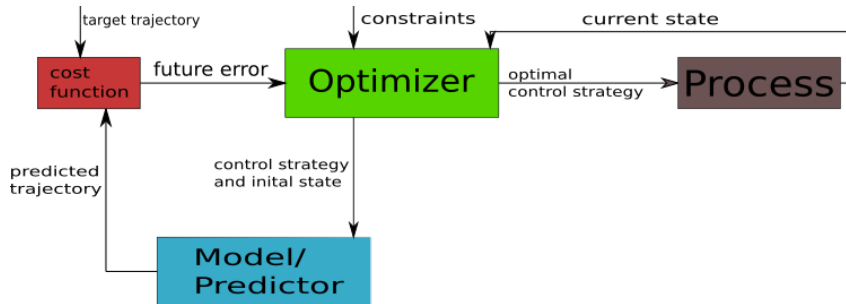


Figure 2: Basic MPC control illustration. The optimizer calculates an optimal control strategy, utilizing a model of the system.

Model Predictive Control (MPC) incorporates optimization of a process model in a finite time horizon  $h \in \mathbb{N}$ . At a given time  $t$  and sampling time  $T$  the controller calculates an optimal control strategy for the time interval  $[t, t + hT]$ , by numerically minimizing a cost function over the predicted trajectory of the system, which is estimated by a model/predictor. Then only the first step of the control strategy is implemented and the procedure repeated at the next iteration. This results in a shifting time frame over which the cost function is minimized in every step.

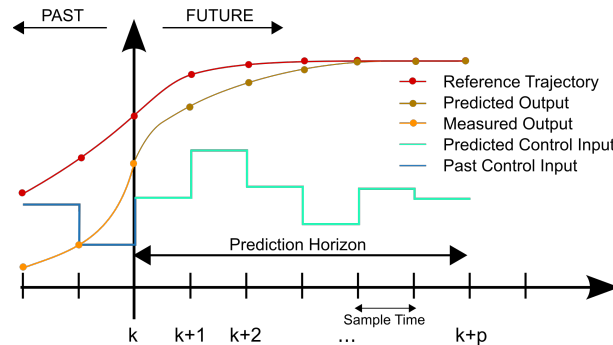


Figure 3: Illustration of the MPC optimization task. The optimizer tries to fit the trajectory of the system to a reference trajectory based on predictions made by the model up to a horizon. Figure adapted from [2].

Assume we have a state space  $\Omega \subset \mathbb{R}^{n_1}$  and a control space  $\mathcal{U} \subset \mathbb{R}^{n_2}$  of variables we can freely manipulate, then a predictor needed for MPC is a function

$$M : \Omega \times \mathcal{U}^h \rightarrow \Omega^h, (x^{(0)}, u^{(1)}, \dots, u^{(h)}) \mapsto (x^{(1)}, \dots, x^{(h)})$$

such that it predicts the trajectory  $x^{(0)} \rightarrow x^{(1)} \rightarrow \dots \rightarrow x^{(h)}$  of future states, given the control  $u_i$  at between time steps.

Incorporating a basic quadratic cost function:

$$e(x, u) = \sum_{i=1}^{n_1} w_{x_i} (r_i - x_i)^2 + \sum_{i=1}^{n_2} w_{u_i} \Delta u_i^2 \quad (2.3)$$



where:

- $x \in \Omega$  is the vector of process variables
- $r \in \Omega$  is the vector of reference variables
- $u \in \mathcal{U}$  is the vector of controlled variables
- $w_{x_i} \in \mathbb{R}$   $i^{\text{th}}$  weighting coefficient reflecting importance of  $x_i$
- $u_{u_i} \in \mathbb{R}$   $i^{\text{th}}$  weight penalizing big changes in  $u_i$

The optimization problem is then given by:

$$\hat{\mathbf{u}}(\hat{x}) = \arg \min_{u^{(1)}, \dots, u^{(h)} \in \mathbb{R}} \sum_{i=1}^h w_i e(x^{(i)}, u^{(i)}) \quad (2.4)$$

with constraints:

$$\alpha \leq \mathbf{u}^{(i)} \leq \beta, \forall i \in \{1, \dots, h\}$$

where the states  $x^{(i)}$  are obtained via the model  $M(\hat{x}, u^{(1)}, \dots, u^{(h)}) = (x^{(1)}, \dots, x^{(h)})$

MPC is a powerful tool which offers a lot of flexibility and is highly customizable as it has many weights which can be tweaked, to e.g. sacrifice immediate gain for future gain ( $w_i$ ), weighting the importance of variables ( $w_{x_i}$ ), or smoothing out the control ( $w_{u_i}$ ). But all this requires a good model of the system which can be hard to acquire, and might be computationally expensive to calculate.

## 2.2 Spectral discovery of jointly smooth functions

As mentioned above, a good model is often crucial for the control of dynamic systems. Such a model needs to be reliable, resistant to disturbance in the input and ideally redundant in the sense that if one or a few input-sensors are lost the model still functions somewhat reasonable. Furthermore finding minimal parameter space realizations in an ongoing challenge[3]. For the purpose of creating such a model which predicts variables from sensor data without knowledge of the dynamic system. We will discover smooth functions  $\{f^{(m)}\}_{m \in \mathbb{N}}$  defined on both the state space  $X \subset \mathbb{R}^{n_x}$  and output space of our predicted variable  $Y \subset \mathbb{R}^{n_y}$ , namely  $f_X^{(m)} : X \rightarrow \mathbb{R}$  and  $f_Y^{(m)} : Y \rightarrow \mathbb{R}$  respectively. Such that for pairs of points in state and corresponding output space  $\{(x_i \in X, y_i \in Y)\}_{i=1}^N$ , the functions satisfy  $f_X^{(m)}(x_i) = f_Y^{(m)}(y_i) \in \mathbb{R}, \forall i, m$ . Now assume we find a selection  $\{i_1, i_2, \dots, i_k\} \subset \mathbb{N}$  such that  $(f_X^{(i_1)}, \dots, f_X^{(i_k)})$  and  $(f_Y^{(i_1)}, \dots, f_Y^{(i_k)})$  are a discrete realisation of smooth and injective functions  $F_X : X \rightarrow \mathbb{R}^k, F_Y : Y \rightarrow \mathbb{R}^k$  such that:

$$F_X(x_i) := (f_X^{(i_1)}(x_i), \dots, f_X^{(i_k)}(x_i)) = (f_Y^{(i_1)}(y_i), \dots, f_Y^{(i_k)}(y_i)) =: F_Y(y_i) \quad (2.5)$$

Then for the points  $x_i$  it follows that  $(F_Y^{-1} \circ F_X)(x_i) = y_i$ . It turns out that this method can be expanded for unseen states  $\hat{x}$  and  $\hat{y} := (F_Y^{-1} \circ F_X)(\hat{x})$  will a good output approximation. This will result in a model that is resistant to disturbance, as demonstrated in the next chapter. The use of such model construction in MPC will be subject of future work. We use a kernel based spectral method to find such function  $f^{(m)}$ . In this section we will develop the tools, required to discover such functions.

### 2.2.1 Kernel method

Kernel machines are instance-based learners, which means rather than learning parameters corresponding to features of the input space, they keep all training data  $(x_i, y_i)$  in memory and learn weights  $w_i$  for each one of them. The output for a new input  $x^*$  is then calculated by using a symmetric similarity function  $k$  called kernel to compare the new input data to the training data weighted with  $w_i$ . For instance for a binary classifier you can calculate the weighted sum:

$$\hat{y} = \text{sgn} \sum_{i=1}^n w_i y_i \mathbf{k}(x_i, x^*)$$

where:

- $\hat{y}$  is the classifier's predicted label for input  $x^*$
- $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is the kernel function measuring similarity
- $\{(x_i, y_i)\}_{i=1}^N$  is the labelled training set with input  $x \in \mathcal{X}$  and output  $y \in \{-1, 1\}$
- $w_i \in \mathbb{R}$  are the weights determined by the learning algorithm

Furthermore a kernel function is a way to calculate a dot product of data in some higher dimensional feature space  $\mathcal{V}$ , without knowledge of this space or how to compute such mapping  $\varphi : \mathcal{X} \rightarrow \mathcal{V}$ . A kernel  $k$  therefore corresponds to a dot product[4] such that

$$\mathbf{k}(x, y') = \langle \varphi(x), \varphi(y) \rangle$$

Many data analysis methods use kernels to efficiently represent data. One of the most frequently used kernels is the Gaussian kernel (RBF Kernel), given by following matrix to represent input data,  $\sigma_x \in \mathbb{R}$  is a parameter controlling the 'strictness' of the kernel as similarity function:

$$\mathbf{K}_x[i, j] = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma_x^2}\right), \mathbf{K}_x \in \mathbb{R}^{N \times N}$$

This matrix approximates the operator  $\exp(-\sigma_x L_x)$  which has the same eigenfunctions as the Laplace-Beltrami operator  $L_x$ . These eigenfunctions span a dense subspace of the function space and correspond to non-negative real eigenvalues[5]. The higher the eigenvalue, the more oscillatory the eigenfunction is. This is demonstrated in figure 4 by plotting the eigenfunctions of the kernel matrix of random points on the unit circle.

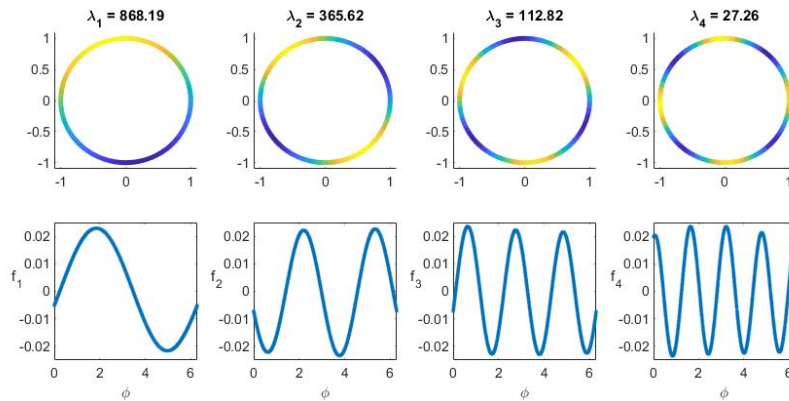


Figure 4: Four different eigenfunctions of the kernel matrix of random points on the unit circle. As scatter-plot in the plane and also graphed in relation to the radial angle  $\phi$ . Note that higher eigenvalues  $\lambda_i$  correspond to a higher oscillation.

### 2.2.2 Discovery of Smooth Functions over Manifolds

This section is a summary of a recent paper from O. Yair et al.[6] which builds foundation for the next chapter.

#### Smooth Functions on Data

We consider a set of points  $\{x_i \in \mathcal{M}_x\}_{i=1}^N$  on a manifold  $\mathcal{M}_x \subset \mathbb{R}^d$  embedded in  $d$ -dimensional Euclidean space. Our following definition of smoothness is inspired by the Dirichlet energy  $\frac{1}{2} \int_{\Omega} \|\nabla f(x)\|_2^2 dx$  of a function  $f : \Omega \rightarrow \mathbb{R}$ , which is a measure of how variable the function is, and further adapted to Kernel matrices, by calling functions smooth if they lie in the span of top eigenvectors of the kernel matrix.

Note that since the functions  $f$  we will work with are discrete, we will use a vector representation through their evaluation on data. They can be viewed as sampled versions of functions  $\hat{f} : \mathcal{M}_x \rightarrow \mathbb{R}$ . Namely the  $i$ -th coordinate corresponds to the evaluation of  $\hat{f}$  at the  $i$ -th data point:  $f[i] := \hat{f}(x_i)$

**Definition 2.1** (Truncated energy). For a number  $d < N < \infty$  and a given function  $\mathbf{f} \in \mathbb{R}^N$ , define its  $d$ -truncated energy  $E_x^d(\mathbf{f})$  with respect to a kernel  $K_x$  by

$$E_x^d(\mathbf{f}) := \|\mathbf{W}_x^T \mathbf{f}\|_2^2 \quad (2.6)$$

where  $\mathbf{W}_x := [w_1 w_2 \dots w_d] \in \mathbb{R}^{N \times d}$  the matrix of normalized eigenvectors corresponding to the  $d$ -highest eigenvalues of  $K_x$

**Definition 2.2** (smooth function). A function  $\mathbf{f} \in \mathbb{R}^N$  with  $\|\mathbf{f}\|_2 = 1$  and with  $E_x^d(\mathbf{f}) = 1$  is called smooth. For two functions  $\mathbf{f}_1, \mathbf{f}_2 \in \mathbb{R}^N$  with  $\|\mathbf{W}_x^T \mathbf{f}_1\|_2 > \|\mathbf{W}_x^T \mathbf{f}_2\|_2$  we call  $\mathbf{f}_1$  smoother than  $\mathbf{f}_2$ .

*Remark:* For all  $\mathbf{f} \in \mathbb{R}^N$  with  $\|\mathbf{f}\|_2 = 1$  it follows from the definition:

$$E_x^d(\mathbf{f}) = 1 \iff \mathbf{f} \in \text{span}(\mathbf{W}_x)$$

**Definition 2.3** (Jointly smooth function). a function  $\mathbf{f} \in \mathbb{R}^N$  with  $\|\mathbf{f}\|_2 = 1$  and  $d$ -truncated Energy  $E_x^d(\mathbf{f}) = E_y^d(\mathbf{f}) = 1$ , with respect to  $\mathbf{K}_x$  and  $\mathbf{K}_y$ , is called jointly smooth on  $\mathcal{M}_x$  and  $\mathcal{M}_y$ .

### Proposed Algorithm

Now we consider two sets of observations  $\{x_i \in \mathcal{M}_x\}_{i=1}^N$  and  $\{y_i \in \mathcal{M}_y\}_{i=1}^N$  such that they build corresponding pairs  $\{(x_i, y_i)\}_{i=1}^N$ . O. Yair et al. now propose Algorithms to find orthogonal functions  $f_m$  which are close to jointly smooth (ordered in decreasing smoothness) on  $\mathcal{M}_x$  and  $\mathcal{M}_y$ . Namely the functions  $f_m$  can be approximated by linear combinations of eigenfunctions of both kernels. These functions are of interest as they represent the common part between the two manifolds. In the next chapter of this thesis we will present applications of such functions. Algorithm 1 is slightly modified from the original to fit our needs.

---

#### Algorithm 1: Jointly smooth functions from 2 observations

---

**Input:** 2 observations  $\{x_i, y_i\}_{i=1}^N$  where  $x_i \in \mathbb{R}^{n_x}$  and  $y_i \in \mathbb{R}^{n_y}$ .

**Output:**  $M$  jointly smooth functions  $\{f_m \in \mathbb{R}^N\}_{m=1}^M$ .

1. Compute the kernels  $\mathbf{K}_x, \mathbf{K}_y \in \mathbb{R}^{N \times N}$ :

$$\mathbf{K}_x [i, j] = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma_x^2}\right), \quad \mathbf{K}_y [i, j] = \exp\left(-\frac{\|y_i - y_j\|_2^2}{2\sigma_y^2}\right)$$

2. Compute  $\mathbf{W}_x \in \mathbb{R}^{N \times d_x}$ ,  $\mathbf{W}_y \in \mathbb{R}^{N \times d_y}$  the first  $d$  normalized eigenvectors of  $\mathbf{K}_x$  and  $\mathbf{K}_y$  associated with the largest eigenvalues, such that all eigenvalues are still greater than  $\varepsilon$ .
  3. Set  $\mathbf{W} := [\mathbf{W}_x, \mathbf{W}_y] \in \mathbb{R}^{N \times (d_x + d_y)}$
  4. Compute a SVD decomposition of  $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
  5. Set  $f_m$  to be the  $m$ -th column of  $\mathbf{U}$ .
- 

*Remark:* one flaw of the Gaussian kernel, used in this method, is that it directly depends on the euclidean distance of data points.  $\|\cdot\|_2$  is highly influenced by the largest vector component. This results in the kernel being biased towards the variable with the biggest changes. So algorithm 1 will produce functions mostly depending on those variables. This means we might need to select or scale our input data accordingly. Further in this thesis we will use  $\sigma_x := 0.3 \cdot \text{median}(\{\|x_i - x_j\|_2^2\}_{i,j})$  and  $\sigma_y$  analogously, as it has shown experimental success. It has also shown practical to restrict  $W_x$  and  $W_y$  to only use eigenvectors corresponding to large enough eigenvalues, as it excludes noise by ignoring small features represented by those vectors, this was not mentioned in the original paper. For this thesis we will utilize a lower bound of  $\varepsilon = 10^{-8}$ .

The main paper also gives algorithms on how to calculate jointly smooth functions from more than two observations and how to estimate  $f_m(x^*)$  for some unseen data  $x^*$  based on the Nyström method. Especially the extension can be used to build a predictor as mentioned in the beginning. Furthermore the paper gives a reasonable value of the parameter  $M$ , namely a lower bound on the d-truncated Energy of the functions  $f_m$  for which those become 'not smooth enough'. We will not go into much depth for those topics and therefore exclude further information in this thesis. Please regard the original paper [6] for

more detailed information, such as the prove that  $f_m$  are indeed jointly smooth on  $\mathcal{M}_x$  and  $\mathcal{M}_y$ .

### 3 Feature Extraction Through Jointly Smooth Functions

In this chapter we will use jointly smooth functions (JSF) in order to obtain features of various dynamic systems. Firstly we construct a toy problem on which the JSFs' ability, to identify and extract effective parameters, is demonstrated. Following that, the classic example of a nonlinear system, the inverted pendulum, is covered. Here we derive the necessary information from basic physics and Lagrange equations, so that we can simulate the system and implement PID/MPC controllers. Those are then used to extract JSFs on data gathered from a controlled inverted pendulum. And finally we illustrate feasibility of our JSF-methods for complex systems by taking a look at data from a simulated spacecraft. The goal is to interpret jointly smooth functions as features of the dynamic systems and utilize them to implement a control loop for said system.

#### 3.1 Toy Problem

Finding minimal parameter spaces for nonlinear dynamical systems from observations is an ongoing challenge[3]. In this example we will use jointly smooth functions to identify an effective parameter from the observation of an aircraft. The planes altitude  $x_1$  and velocity  $x_2$  is modelled by the nonlinear dynamic system described by following differential equation, depending on parameters  $p_1, p_2, p_3$ .

$$\dot{x} = J(s(x)) \cdot g_{p_1, p_2, p_3}(s(x)) \in \mathbb{R}^2 \quad (3.1)$$

with nonlinear transformation  $J$  and linear oscillator  $g$  defined by:

$$J(x) = \begin{bmatrix} 1 & -2x_2 \\ -2x_1 + 2x_2^2 & 1 + 4x_1x_2 - 4x_2^3 \end{bmatrix}, \quad s(x) = \begin{bmatrix} x_1 + x_1^4 + 2x_1^2x_2 + x_2^2 \\ x_1^2 + x_2 \end{bmatrix}$$

and

$$g_{p_1, p_2, p_3}(x) = \begin{bmatrix} -2 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 - (p_1 + p_2^3) \\ x_2 - p_3 \end{bmatrix} \quad (3.2)$$

the system models the planes as its speed and velocity oscillates towards a steady state given by  $\hat{x} := s(p_1 + p_2^3, p_3)$ . Equation 3.1 describes a spiralling contraction towards  $\hat{x}$ , which is illustrated in figure 5(a) by its flow. Note that the systems dynamics only depend on the parameters  $p_{\text{eff}} := p_1 + p_2^3$  and  $p_3$ , despite having a three dimensional parameter space.

Now suppose we have no knowledge of the dynamics of our system and only  $p_1$  and  $p_2$  are controllable while  $p_3$  is unknown. We can simulate the aircraft and therefore observe the steady state for different parameters  $p_1$  and  $p_2$ . From such observations we will use algorithm 1 to identify the effective parameter  $p_{\text{eff}}$ . For this purpose we generate

$N = 4000$  parameter triplets i.e.  $\{(p_1, p_2, p_3)_i\}_{i=1}^N$  uniformly distributed on  $[-1, 1]^3$  and then observe the steady state  $\{(\hat{x}_1, \hat{x}_2)_i\}_{i=1}^N$  for each set of parameters through simulation until convergence. We then apply algorithm 1 on  $\{(p_1, p_2)_i, (\hat{x}_1, \hat{x}_2)_i\}_{i=1}^N$  with  $d = 500$  in order to get jointly smooth function  $f_m$  on the observable parameters  $(p_1, p_2)$  and steady state  $\hat{x}$ . In the top row of figure 5(b) we plot the four most jointly smooth functions as functions of  $p_1 + p_2^3$  and observe that there is a obvious correspondence with the unknown parameter combination. In order to find such combination we can take a look at the scatter of  $p_1$  against  $p_2$ , coloured by the jointly smooth functions  $f_m$ , as depicted in the second row of figure 5(b). The level sets coincide with the graphs of  $p_1 + p_2^3 = C$ ,  $C \in \mathbb{R}$ , which is expected because along these graphs the effective parameter is constant and therefore the dynamical system behaves the same. Going back to our plane, this implies that we can reduce the parameter space for the aircrafts' steady state control by utilizing the parameter combination  $p_1 + p_2^3$ .

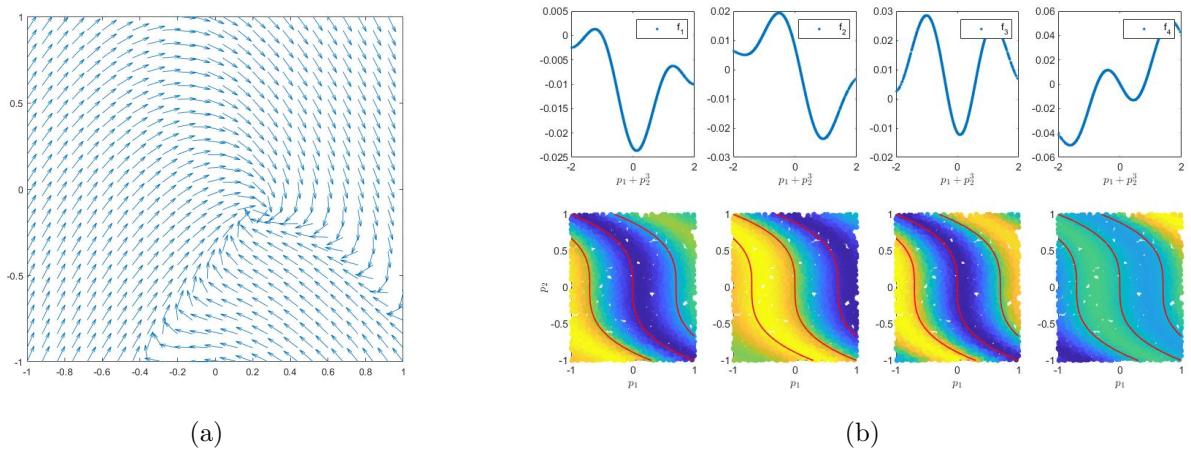


Figure 5: On the left we have the flow of the system for parameters  $(p_1, p_2, p_3) = (0.2, 0.3, -0.1)$ ; on the right the top four jointly smooth functions, obtained by algorithm 1, are depicted. The top row plots the functions against the hidden effective parameter  $p_{\text{eff}} := p_1 + p_2^3$  and observe that all  $f_i$  are smooth on  $p_{\text{eff}}$ . The bottom row illustrates how to find such effective parameter, as  $p_{\text{eff}}$  coincides with the level sets of the scatter plot of  $p_1$  to  $p_2$  coloured by  $f_m$ . The red lines are the graphs of  $p_1 + p_2^3 = C \in \{-0.7, 0, 0.7\}$

## 3.2 Control of an inverted pendulum

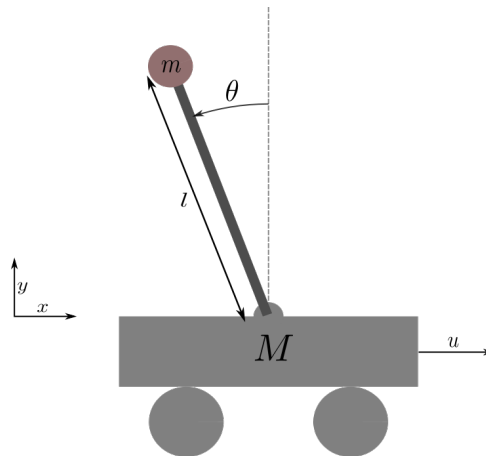


Figure 6: Illustration of our inverted pendulum on a cart, controlled by a force  $u$ .

The inverted pendulum is a classic problem in dynamics and control theory which can be used to benchmark control strategies[7][8]. Here we will use this dynamic system to expand on the use cases and demonstrate the feature defining ability of jointly smooth functions. We will gather data from the control of such pendulum, use it to find jointly smooth functions and further illustrate how to utilize those JSFs to develop a control system from the data.

### 3.2.1 Modelling of the inverted pendulum

#### System Description

We consider a cart pendulum system as depicted in figure 6. The cart of mass  $M$  can move on the track with a ball (with mass  $m$ ) attached via a massless rod of length  $l$ .  $\theta$  is the angle in which the rod deviates counter-clockwise from vertical, while the gravitational acceleration acting on the arm is  $g$ . The system is controlled by a force  $u(t)$  on the cart, in the positive  $x$  direction. The masses are considered point-masses attached to the rod and the system is assumed to be frictionless.

#### Deriving the equations of motion

By defining the potential energy of the cart as zero we can write the total potential energy  $V$  of the system as:

$$V = mgl \sin(\theta) \quad (3.3)$$

In order to calculate the kinetic energy  $T$  we derive the position  $x_b$  and velocity ( $\dot{x}_b$ ) of the ball, from the carts horizontal position  $x$  and angle  $\theta$ :

$$x_b = \begin{bmatrix} x - l \sin(\theta) \\ l \cos(\theta) \end{bmatrix} \quad (3.4)$$

$$\dot{x}_b = \begin{bmatrix} \dot{x} - l\dot{\theta} \cos(\theta) \\ -l\dot{\theta} \sin(\theta) \end{bmatrix} \quad (3.5)$$

and then write  $T$  as the sum of kinetic energy of cart and ball:

$$T = \frac{1}{2}M\|\dot{x}\|_2^2 + \frac{1}{2}m\|\dot{x}_b\|_2^2 = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 - ml\dot{\theta}\dot{x}\cos(\theta) \quad (3.6)$$

using the Lagrangian  $\mathcal{L} = T - V$  we apply the Euler-Lagrange equations.

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{\theta}} \right) - \frac{\partial \mathcal{L}}{\partial \theta} = Q_\theta \quad (3.7)$$

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{x}} \right) - \frac{\partial \mathcal{L}}{\partial x} = Q_x \quad (3.8)$$

$$\mathcal{L} = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 - ml\dot{\theta}\dot{x}\cos(\theta) - mgl\sin(\theta) \quad (3.9)$$

with the external moment  $Q_\theta = 0$  and  $Q_x = u$  we get:

$$l\ddot{\theta} = \ddot{x}\cos(\theta) + g\sin(\theta) \quad (3.10)$$

$$(M+m)\ddot{x} - ml\cos(\theta)\ddot{\theta} + ml\dot{\theta}^2\sin(\theta) = u \quad (3.11)$$

by substitution of  $\ddot{\theta}$  and  $\ddot{x}$  respectively, the final equations of motion follow:

$$\ddot{\theta} = \frac{u\cos(\theta) + ml\dot{\theta}^2\sin(\theta)\cos(\theta) + (M+m)g\sin(\theta)}{Ml + ml - ml\cos^2(\theta)} \quad (3.12)$$

$$\ddot{x} = \frac{u + mg\cos(\theta)\sin(\theta) + ml\dot{\theta}^2\sin(\theta)}{M + m - m\cos^2(\theta)} \quad (3.13)$$

These can be used to describe the dynamic system with a time independent ordinary differential equation of first order in a four dimensional state space. This allows us to numerically simulate the nonlinear inverted pendulum on a cart and further control it via the parameter  $u$ .

$$\frac{d}{dt}\mathbf{x} = \frac{d}{dt} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{u + mg\cos(\theta)\sin(\theta) + ml\dot{\theta}^2\sin(\theta)}{M + m - m\cos^2(\theta)} \\ \dot{\theta} \\ \frac{u\cos(\theta) + ml\dot{\theta}^2\sin(\theta)\cos(\theta) + (M+m)g\sin(\theta)}{Ml + ml - ml\cos^2(\theta)} \end{bmatrix} \quad (3.14)$$

### 3.2.2 Controller Realization and Choice of Hyperparameters

Here we demonstrate two Controller realizations, one PID and MPC-Controller. For this we will use an inverted pendulum on a cart with mass  $M = 5kg$ . The ball ( $m = 1kg$ ) is connected with a massless rod of length  $l = 1.5m$ . The gravitational acceleration is  $g = 9.81\frac{m}{s^2}$  and we will ignore any friction so we can use the equations of motion from the previous section.



### System simulation

The simulation on which we test our following controllers is based on numerical integration of the equations of motion (3.12) using an explicit Runge-Kutta (4,5) formula[9]. With a sample rate  $T = 0.2s$  we can, in this manner calculate the trajectory of the system originating from a state  $x_i$  at time  $t_i$  up to a state  $x_{i+1}$  at time  $t_{i+1} := t_i + T$ . We iterate this procedure until we reach the time limit of  $t_{max} = 5s$ . While we update the control force  $u$  according to the controller at each iteration, taking  $x_i$  into account.

### PID-Controller

Since the PID-Controller only works with one process variable, we will use the angle deviation of the rod from vertical  $\theta$ , resulting in a error function  $e(t) := -\theta(t)$ . As for parameters  $K_p$ ,  $K_i$  and  $K_d$ , we obviously need to make the control force  $u$  depended on the angle  $\theta$  and angular velocity  $\dot{\theta}$ , but since we did not incorporate  $\dot{\theta}$  into the error function we will need a PD-Controller or PID-Controller to account for the change in angle. Because we already have a understanding of the dynamic system we can derive a lower bound on  $K_p$  from the equation of motion (3.12) using the low angle approximations,  $\cos(x) \approx 1$  and  $\sin(x) \approx x$ . So for a angular stationary ( $\dot{\theta} = 0$ ) pendulum we get:

$$0 = \ddot{\theta} \approx \frac{u + (M + m)g\theta}{Ml} \Rightarrow u \approx -(M + m)g\theta = -58.86 \theta$$

In conclusion we need  $K_p > 58.81$  for the controller to overcome the gravitational pull on the ball. In expectation that the angular velocity is equally influential we can start off with parameters  $K_p = K_d = 100$  and see how the system behaves. The controller can then be further adjusted for its needs, depending on whether you want to minimize overshoot, time to reach a stable state, etc. Increases in  $K_p$  will result in stronger corrections and  $K_d$  can decrease oscillations and overshoot.

The result of such PD-Controller on the system is demonstrated in figure 6(a). As you can see we only regulate the angle of the pendulum. Often you want to also control more than one process variable, this is achieved by creating a separate PID-controller for each one. So in our case we could implement another controller which affects the position[10].

### MPC-Controller

One of the many advantages of MPC, in contrast to PID, is that you can control more than one process variable at once. So we will incorporate travelling of the cart to a given point and stabilizing it there. Additionally to not utilize the exact same equations as model as used by the simulation, we instead make use of a linearised version obtained via small angle approximation:

$$\frac{d}{dt} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{u+mg\theta+ml\dot{\theta}^2\theta}{M} \\ \dot{\theta} \\ \frac{u+ml\dot{\theta}^2\theta+(M+m)g\theta}{Ml} \end{bmatrix} \quad (3.15)$$

The model incorporates a numerical solution of this differential equation to simulate the

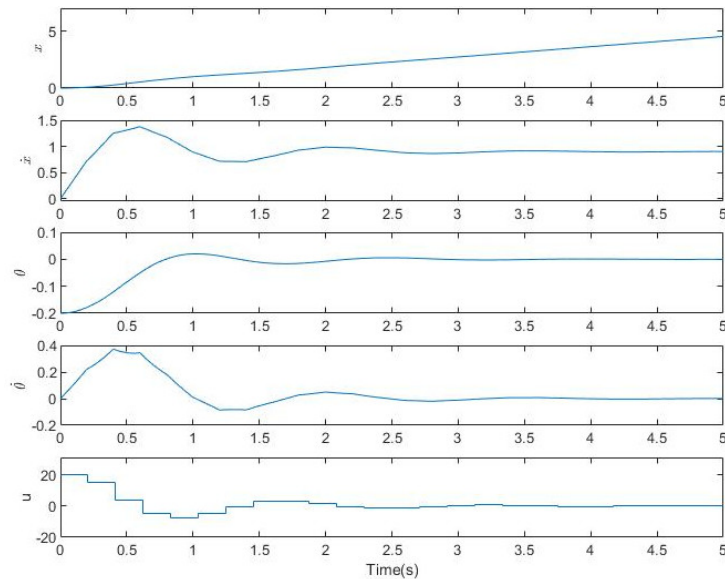
trajectory. Further we will use following quadratic cost function:

$$\mathbf{e}(x, u) = \sum_{i=1}^4 w_{x_i} (r_i - x_i)^2 + w_u \Delta u^2 \quad (3.16)$$

Further apply following weights:  $w := (0.25, 0.1, 1, 0)^T$ ,  $w_x := (1, 0, 1, 0)^T$  (only taking position and angle into account) and  $w_u := 0.001$ , forcing more subtle changes in the control force.

$$\hat{\mathbf{u}}(\hat{x}) = \arg \min_{u^{(1)}, \dots, u^{(h)} \in \mathbb{R}} \sum_{i=1}^h \frac{i}{2} \mathbf{e}(x^{(i)}, u^{(i)}) \quad (3.17)$$

The solution of this optimization problem is numerically obtained with the constraints  $-20 < u^{(i)} < 20$ . So that the controller will then use the first component of  $\hat{u}$ , implementing only the first control step of the obtained optimal control strategy. The trajectory endorsed by this controller is depicted in figure 6(b,c).



(a) PID-Controller  $(K_p, P_i, P_d) = (100, 0, 140)$

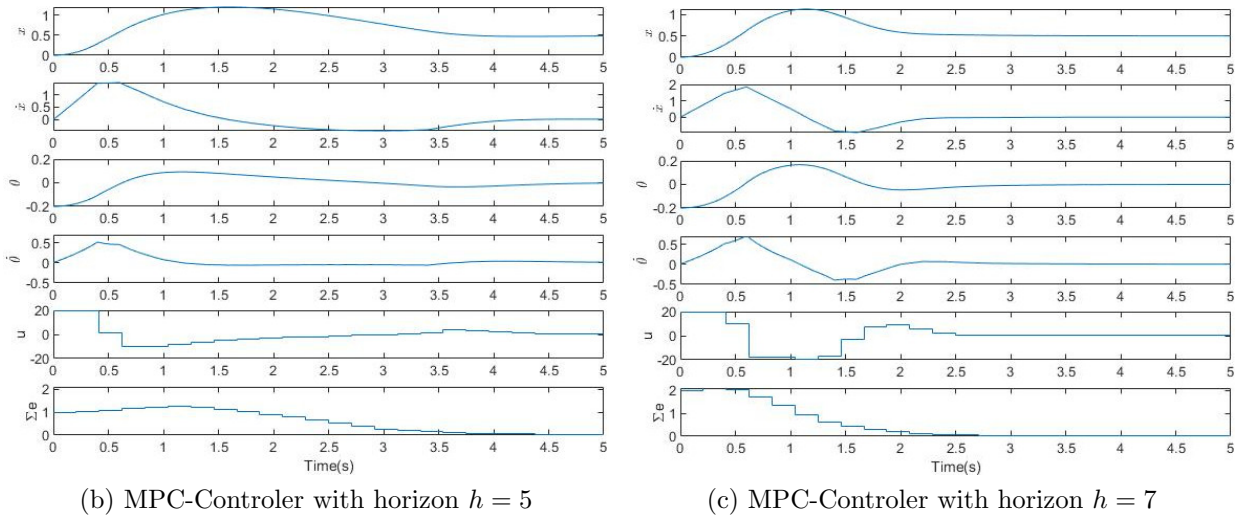


Figure 6: (a) illustrates a PD controller with parameters  $K_p = K_d = 100$ . You can see the angle  $\theta$  slowly oscillating towards the target  $r(t) = 0$ , meanwhile the uncontrolled cart-position  $x$  steadily rises; (b) and (c) show the systems trajectory under MPC-Controllers. They both have the same weights in the cost function as defined above, the same target  $r(t) = (0.5, 0, 0, 0)^T$ , but different horizons. The last row is the diagram, is the cost function the optimizer tries to minimize. Note that they both overshoot their target and then slowly approach position and angle from above. The one with greater horizon acts more aggressive as it sees immediate gain in the future and therefore achieves the target state sooner. But the higher horizon resulted in doubling of the average computation time per time step.

### 3.2.3 Jointly smooth functions over pendulum data

We will now apply the methods of section 2.2.2 to find jointly smooth functions on data gathered from pendulum simulations. It is generated by letting the PD-Controller ( $K_p = K_d = 100$ ) stabilize a pendulum for 5 seconds at 50 equidistant starting angles  $\theta \in [-0.5, 0.5]$ . The simulation has a sampling time  $T = 0.1s$ , meaning that every  $0.1s$  a new control output is calculated and applied until the next one is requested. The states  $x_i := (\theta_i, \dot{\theta}_i)$  on which the controller acts (containing angle  $\theta$  and angular velocity  $\dot{\theta}$ ), are then collected with their corresponding output  $u_i$ , such that we get observations  $\{(x_i, u_i)\}_{i=1}^N$  where  $x_i \in \mathbb{R}^2$  and  $u_i \in \mathbb{R}$ . In this manner we get  $N = 2500$  such points, on which Algorithm 1 can be applied with  $d = 250$ . This generates jointly smooth functions  $f_m$  on the manifolds underlying state and output space. Since the control output is one dimensional the functions are expected to be smooth on the output data. Figure 7(top) shows the 4 smoothest functions as their evaluation on the control output.

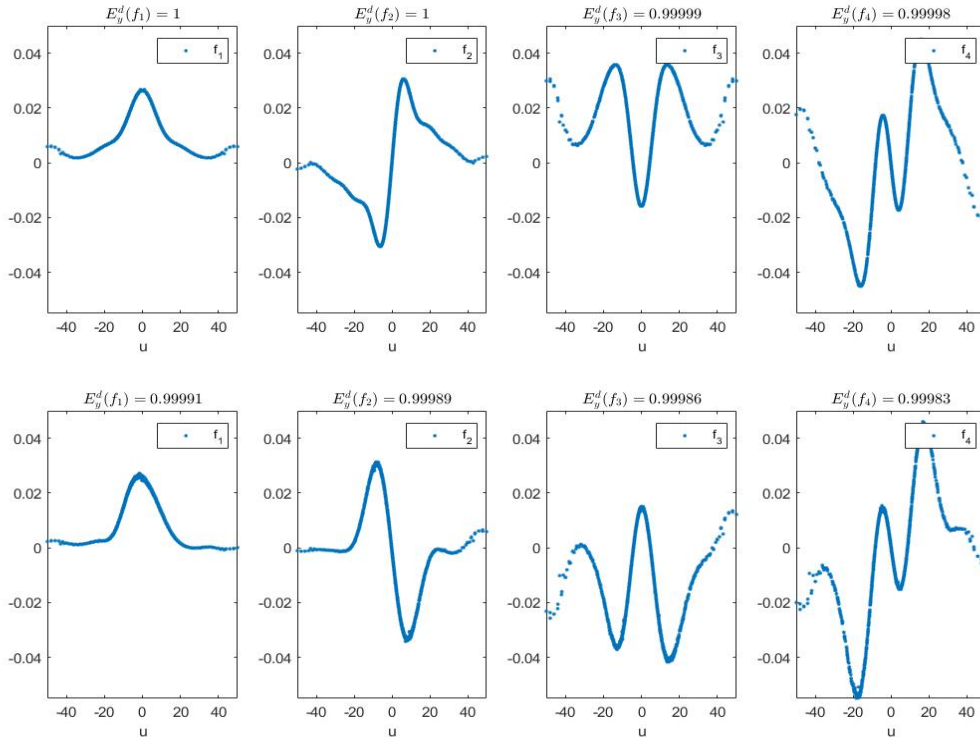


Figure 7: Top 4 jointly smooth functions obtained by using algorithm 1 on data gathered by 50 simulations of inverted pendulum under PID-control; The top row uses the angel and angular velocity in its data, while the bottom row has 5 dimensions of noise included in the sensor data. Both still deliver roughly the same functions, up to their sign and mild boundary effects.

### Adding noise

We will now add further sensors, to the existing ones ( $\theta$  and  $\dot{\theta}$ ), which only return noise. This simulates sensors which either measure completely unrelated quantities or broken ones only returning noise. We expand  $x_i$  from previously to  $x_i^* := (\theta_i, \dot{\theta}_i, z_i^{(1)}, \dots, z_i^{(5)})$  where  $z_i^{(1)}, \dots, z_i^{(5)}$  are uniformly distributed on  $[0, 0.5]^5$ . We should not use too big noise, because the used Gaussian kernel is biased towards the vector components of greatest value changes. This noise is still considerable big enough, regarding the largest starting angle is  $\pm 0.5$ . The usage of other kernels to counteract this issue is subject of future work. We again apply algorithm 1 to  $\{(x_i^*, u_i)\}_{i=1}^N$  with  $d = 250$ . The obtained jointly smooth functions  $f_m$ , depicted in figure 7(bottom) are approximately the same as the ones acquired from noiseless data, up to mainly boundary effects and the sign. This concludes that obtaining jointly smooth functions also works reasonably well under the influence of unrelated or noised sensors.

### Adding shifted sensor data

Now we will briefly test if the obtained jointly smooth functions are dependent on the embedding of the manifold. We will do this by expanding the input space by shifted data residing on the same manifold. For this we will use  $x_i^+ := (x_i, x_i^{0.3T}, x_i^{0.6T}, x_i^{0.9T}) \in \mathbb{R}^8$  whereas  $x_i$  is a observed state at time  $t_i$  as before and  $x_i^{0.3T}$  is the state observed at time

$t_i + 0.3T$ .  $x_i^{0.6T}, x_i^{0.9T}$  are defined respectively. This means we add states observed between time-steps. This is a simple method to lift the data into 6 dimensional space. As usual algorithm 1 is then applied to  $\{(x_i^+, u_i)\}_{i=1}^N$  with  $d = 250$ . The resulting jointly smooth functions do not noticeably differ from the ones obtained from the pure data. This implies that we can also add further sensors, related to the output, resulting in redundant information and still obtain similar JSFs because those depend on the underlying manifold, not the concrete realization.

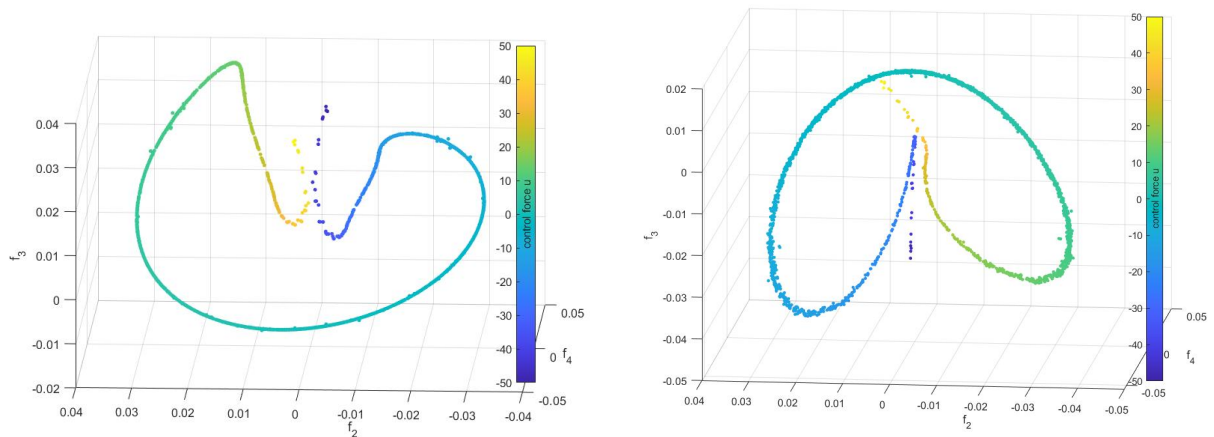
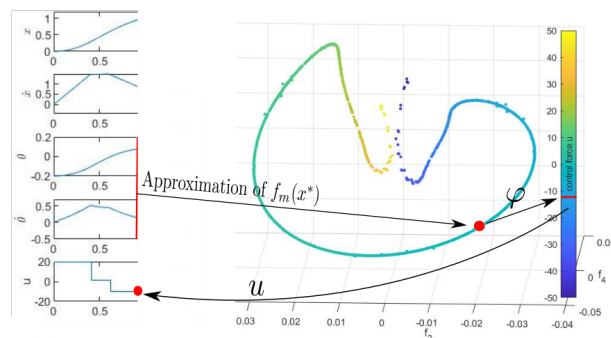


Figure 8:  $f_2, f_3, f_4$  plotted against each other and coloured according to the control output  $u$ ; the right plot included used 5 dimensions of noise in the generation of  $f_m$

### Output prediction for unseen states

Since each jointly smooth function represents something common between the sensor space and control output, we can interpret such a function as features of the dynamic system. We can now try to make a selection  $\{f_{i_1}, f_{i_2}, \dots, f_{i_k}\}$  of JSF's, such that the curve  $F := \{(f_{i_1}(x_i), \dots, f_{i_k}(x_i)) \in \mathbb{R}^k \mid i \in \{1, \dots, N\}\}$  is a discrete sampling of a smooth and injective curve  $\hat{F}$ . Furthermore  $x_i$  corresponds to a control  $y_i$  which induces a continuous mapping  $\varphi : \hat{F} \rightarrow \mathbb{R}$  from the feature space to the control space. Such a selection is represented in figure 8. The motivation is, that if we find such a curve  $\hat{F}$  we can predict a control output from this feature space  $\mathbb{R}^k$  using  $\varphi$ . For new unseen sensor data  $x^*$  and a good approximation of  $f_m(x^*)$  (presented in [6]) we can then make a prediction for the control  $y^* = \varphi(f_{i_1}(x^*), \dots, f_{i_k}(x^*))$ .



### 3.3 Landing rockets

Considering how expensive space travel is, re-usability of spacecraft is an important topic in the advancements of the field. After making history by the successful landing of the Falcon 9 rocket by the private company SpaceX[11], this subject gained increasing public attention. In the following section we will apply our methods of extracting JSFs to data gathered from a simulated rocket as it performs vertical landings.



#### 3.3.1 The Simulation environment and obtaining data

The dynamic system is given by Kerbal Space Program (KSP), a space flight simulation video game published in 2015. It offers an environment for realistic simulations of space travel. We will use data obtained by Kaan Atukalp, where he demonstrates a MPC-controller utilizing machine learning in his predictor to land a rocket[12]. The controller is based on the design Ali Ganbarov developed in his thesis which uses MPC for altitude control and PID to keep the spacecraft vertical[13]. Said data includes 7 landings from an altitude of 3000m. The MPC-controller minimizes towards a target altitude  $y_{\text{target}}$  and target velocity  $v_{\text{target}}$  depending on the current altitude  $y$ :

$$\mathbf{y}_{\text{target}} := \begin{cases} y - 500 & 5000 < y \\ y - 800 & 5000 \geq y > 1000 \\ -2 & 1000 \geq y \end{cases} \text{ and } \mathbf{v}_{\text{target}} := \begin{cases} -200 & 1000 < y \\ 0 & 1000 \geq y \end{cases} \quad (3.18)$$

We will only use the vertical speed, altitude and thrust in the upcoming, even though the data also includes orientation, drag, etc. In Figure 9 we visualize the data used in this experiment. You can see, the controller does not counteract the fall for approximately the first 14s where it starts to stabilize the vessel to a constant velocity of around  $-150 \frac{m}{s}$ . Note that the target speed  $v_{\text{target}} = -200 \frac{m}{s}$  is not approached because MPC also tries to minimize the distance towards the target altitude over the timespan of the horizon.

Particularly a stable altitude on the horizon would imply a speed of 0, all this results in achieving neither the target altitude nor speed. Similarly to the time interval  $[0s, 14s]$  there is no throttle around  $17s$  as the vessel drops below  $1000m$  and the target values change, resulting in the altitude objective dominating the target speed. Then in the final stage, the MPC-controller does its main work, balancing both objectives aiming for a gentle touch down.

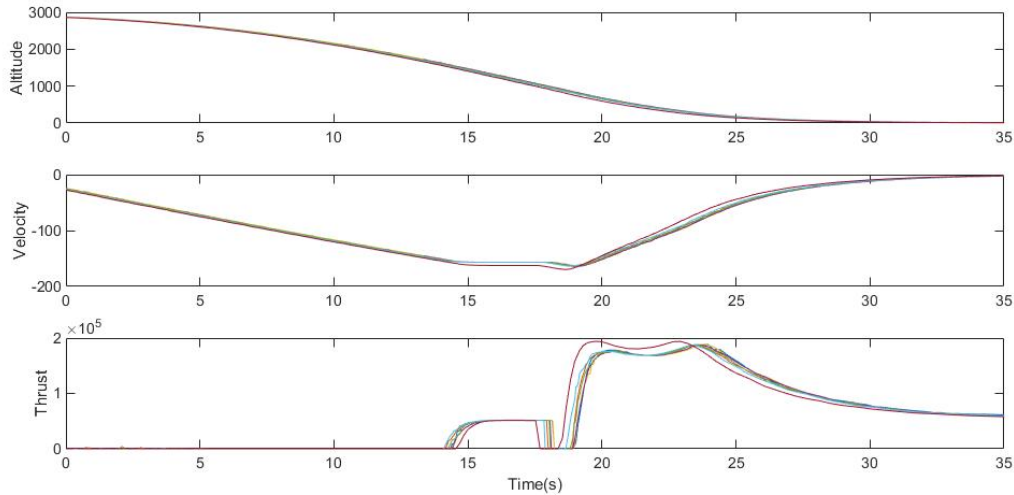


Figure 9: Visualization of the 7 landings used in the data set

### 3.3.2 Extraction of jointly smooth functions

We collect the observations from all 7 landings and filter for altitude  $x_i$ , speed  $v_i$  and thrust  $u_i$  resulting in  $N = 2919$  data points  $\{((x_i, v_i), u_i)\}_{i=1}^N$ . On this we can apply algorithm 1 with  $d = 300$  and get jointly smooth functions  $f_m$  on the manifolds of sensor data (altitude/speed) and the control output (thrust).

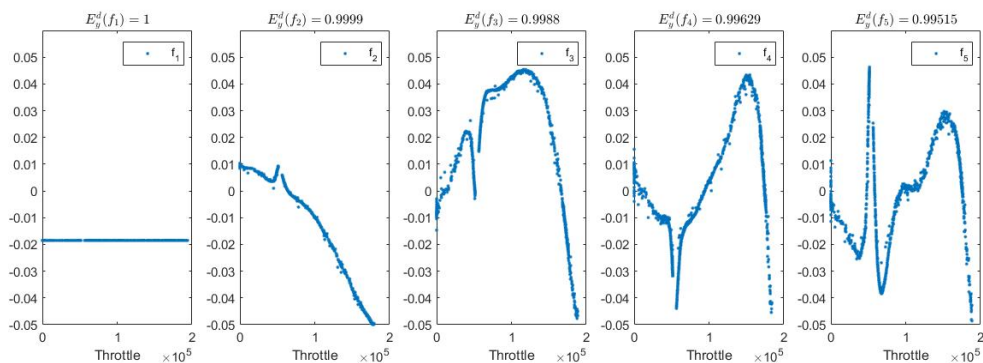


Figure 10: Top five jointly smooth functions.

In figure 10 we depict the 5 most jointly smooth functions on the observations.  $f_1$  is the trivial constant function and therefore not of great interest. On the rest you can see a flaw of this method as all functions have a big sudden spike around  $5 \cdot 10^4$  throttle. This

can occur if the data is not dense enough distributed. On close inspection there is no observation with a throttle of  $(5.3 \pm 0.3) \cdot 10^4$ , which causes a discontinuity there. Even though some boundary effects appear around this spot, we can still continue with our method and create a selection of jointly smooth functions such that we get an injective mapping into some feature space.  $\{f_2\}$  might be a sufficient choice, but  $\{f_2, f_4\}$  probably result in more accurate predictions around the discontinuity. This is depicted in figure 11. The result is obviously a segmented curve.

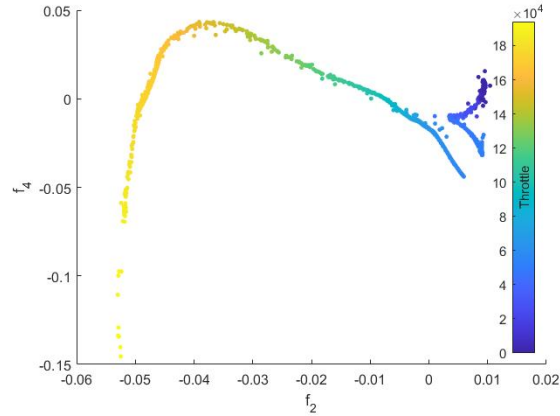


Figure 11:  $f_2$  and  $f_4$  form an injective mapping into the feature space  $\mathbb{R}^2$ , which can therefore be used for our system control loop.



## 4 Conclusion

In this thesis we reviewed two popular control mechanisms PID and MPC, such that we can stabilize an inverted pendulum, in order to generate sensor data. This data is then used to discuss the flaws, advantages and mainly applications of jointly smooth functions, a recent approach employing a spectral method on kernels. Said function are then utilized to discover common parts between data sets gartered from dynamic systems simulations. Therefore they represent features of the underlying system. We showed in our toy example how it can be possible to extract effective parameters from JSFs, reducing the parameter space for system control. Furthermore we demonstrate the resistance of JSFs to noised and redundant data. Ultimately we show how JSFs can theoretically be utilized as a predictor based on a very small amount of training data. Further we demonstrate that our approach can also be used in complex dynamic systems, by applying it to rocket landing simulations.

In this thesis we used the Gaussian kernel, but noted this also induces a few limitations on our data. Experimenting with different kernels to overcome this issue will be subject of future work. Further it is suggested that we can modify our predictor, to work under malfunction of some sensors by looking at more than 2 observation. If this turns out to be feasible or not remains to be seen.

## References

- [1] Kiam Heong Ang, G. Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.
- [2] Behrendt Martin. A discrete mpc scheme. [https://commons.wikimedia.org/wiki/File:MPC\\_scheme\\_basic.svg](https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg).
- [3] Alexander Holiday, Mahdi Kooshkbaghi, Juan M. Bello-Rivas, C. William Gear, Antonios Zagaris, and Ioannis G. Kevrekidis. Manifold learning for parameter reduction. *Journal of Computational Physics*, 392:419431, Sep 2019.
- [4] Thomas Hofmann, Bernhard Schölkopf, and Alexander Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36, 01 2007.
- [5] Yoshida Kosaku. *Functional analysis*. Springer, 1995.
- [6] Or Yair, Felix Dietrich, Rotem Mulayoff, Ronen Talmon, and Ioannis G. Kevrekidis. Spectral discovery of jointly smooth features for multimodal data, 2020.
- [7] Yoshida Kosaku. *Bioinspired legged locomotion: models, concepts, control and applications*. M. Sharbafi and A. Seyfarth, 2017.
- [8] Maria Landry, Sue Campbell, Kirsten Morris, and Cesar Aguilar. Dynamics of an inverted pendulum with delayed feedback control. *Siam Journal on Applied Dynamical Systems - SIADS*, 4, 01 2005.
- [9] L.F. and M.W. Reichelt. The matlab ode suite. *SIAM Journal on Scientific Computing*, 18:1–22, 1997.
- [10] Elisa Sara Varghese, Anju K Vincent, and V Bagyaveereswaran. Optimal control of inverted pendulum system using PID controller, LQR and MPC. *IOP Conference Series: Materials Science and Engineering*, 263:052007, nov 2017.
- [11] Loren Grush. SpaceX successfully landed its falcon 9 rocket after launching it to space. <https://www.theverge.com/2015/12/21/10640306/spacex-elon-musk-rocket-landing-success>, Dec 2015.
- [12] Kaan Atukalp. Automated feature selection and learning of spaceship model for model predictive control, 2021.
- [13] Ali Ganbarov. Autonomous spaceship navigation and landing using model predictive control, 2020.