Technische Universität München

Department of Mathematics

Bachelor's Thesis

Analysing Kernel Flows through the Koopman Operator

Daniel Kowatsch

Supervisor: Prof. Dr. Hans-Joachim Bungartz

Advisor: Dr. Felix Dietrich

Submission Date: 15.02.2022

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

Garching,

Zusammenfassung

Kernel-Methoden spielen eine wichtige Rolle bei der Lösung von Klassifizierungs- und Regressionsproblemen. Während ein Kernel in der Regel von Hand entworfen wird, haben Owhadi und Yoo vor kurzem Kernel Flows als datengesteuerten Ansatz zum Finden geeigneter Kernel vorgeschlagen. Eine erste Analyse von Darcy hat gezeigt, dass Konvergenz und Stabilität der Konvergenz von der Initialisierung der Parameter abhängen. Wir analysieren Kernel Flows unter Verwendung des von Dietrich et al. vorgeschlagenen Koopman-Operator-Frameworks, um das Verhalten des Algorithmus besser zu verstehen. Zu diesem Zweck formalisieren wir Kernel Flows als dynamisches System und verwenden es, um Kernel für Spielzeugdatensätze zu erlernen und das Punktspektrum des Koopman-Operators unter Verwendung von EDMD anzunähern. Auf der Grundlage des Spektrums suchen wir nach Eigenwerten, die nahe bei eins liegen, und bestimmen die Anziehungsbereiche auf der Grundlage einer Clusterung der Auswertung der Eigenfunktionen zu Eigenwerten, die nahe bei eins liegen. Wir zeigen, dass die Analyse durch den Koopman-Operator von einer guten Wahl der Anzahl der Cluster abhängt, die beim k-means-Clustering verwendet werden. Dies stellt ein Problem dar, da unseres Wissens nach kein Ansatz zur zuverlässigen Bestimmung der Anzahl der Anziehungsbereiche existiert. In unserer Analyse der Kernel Flows durch den Koopman-Operator haben wir gezeigt, dass es einen signifikanten Unterschied in der Form der dargestellten Anziehungsbereiche der verwendeten synthetischen Datensätze im Vergleich zu den Anziehungsbereichen auf echten Datensätzen gibt. Dies könnte auf ein Problem mit den synthetischen Datensätzen und damit mit der darauf basierenden Analyse hindeuten.

Abstract

Kernel methods play an important role in solving classification and regression problems. While usually a kernel is designed by hand, recently, Owhadi and Yoo proposed Kernel Flows as a data-driven approach to obtaining suitable kernels. A first analysis by Darcy has shown that convergence and stability of convergence depend on the initialisation of parameters. We further analyse Kernel Flows utilising the Koopman operator framework proposed by Dietrich et al. in order to better understand the behaviour of the algorithm. To this end, we formalize Kernel Flows as a dynamical system and use it to learn kernels for toy data sets and estimate the point spectrum of the Koopman operator using extended dynamic mode decomposition. Based on the spectrum, we search for eigenvalues close to one and determine the basins of attraction based on a clustering of the evaluation of the eigenfunctions to eigenvalues close to one. We show that the analysis through the Koopman operator relies on a good choice of the number of clusters used in k-means clustering which poses a problem since, to the best of our knowledge, no approach to reliably determine the number of basins of attraction exists. In our analysis of Kernel Flows through the Koopman operator, we have shown that there is a significant difference in the form of the depicted basins of attraction of the used synthetic datasets compared to real basins of attraction on real datasets. This might indicate a problem with the synthetic datasets and, thus, the analysis based on it.

Contents

1 Introduction

Many problems in the real world can be formalized as some sort of prediction by a function. That means, we have some known input variables and want to predict one or more target variables. Even if the relationship between input variables and target variables is not completely deterministic, it is still possible to use such an approach by extending it with the assumption of a noisy target variable. Common examples for such problems include the prediction of the correct drug dose based on medical features, the prediction of stock market prices based on a history of previous values, the classification of pictures into classes describing their content, the prediction of interesting video content based on a users history of previously watched content, or the prediction of the progression of diseases like diabetes based on medical information of a patient.

One important question when modeling a given real world problem is, what function should be used to model the relationship between input features and output variables. One option is Linear Regression, where the target variable is just a weighted sum of the input parameters. Another common choice for classification are SVMs, which essentially attempt to find a line separating two classes. An approach based on the distance between datapoints in feature space is k-nearest neighbors. Here, in order to find the correct label to a new datapoint, we calculate the distance between the new datapoint and a set of existing ones with known label and, then, label the new datapoint according to the majority class of the k nearest neighbors, where k is determined by the person developing the model.

Unfortunately, all these approaches have significant limitations. For example, there just simply might not be a linear relationship between measured medical features and the progression of diabetes. Also, there is not necessarily a clear line one can draw between pictures of cats and pictures of dogs. Or, for k-nearest neighbors, the natural distance between two sets of features describing videos might not necessarily mean, that they are similarly interesting to a watcher. One solution to these problems are kernels, which essentially project the features into a different space, in which e.g. exists a linear relationship between projected medical features and the progression of diabetes. As a result of this, kernel methods are one of the go-to approaches when modelling the relationship between feature variables and target variables. While the introduction of kernels significantly increases the expressive power of existing methods, it also comes with the problem of selecting the best kernel for a given problem. Since the solution to the kernel selection problem is not trivial, Owhadi&Yoo introduced Kernel Flows, a data-driven approach to finding good kernels.

The relevance of Kernel Flows to solving the problem of finding good kernels to a given problem motivates us to analyse the algorithm in this thesis. In order to analyse Kernel Flows, we use the Koopman Operator Framework which has recently been proposed for analysis of algorithm by Dietrich et al. [8]. For this purpose, we first introduce the state of the art in chapter 2. This noticeably includes a brief introduction in the basic idea of kernel regression in section 2.1.1. In section 2.1.2 we provide some examples of existing families of kernels, which will later on represent the set of potential kernels in which we search for the optimal one with Kernel Flows. But before we can formally introduce Kernel Flows, we first present the notion of a 'good' kernel used by Owhadi&Yoo [24] in section 2.1.3. After introducing sufficient background knowledge, we can finally introduce

the parametric version of Kernel Flows, the algorithm under investigation in this thesis, in section 2.1.4.
After explaining the necessary background for Kernel Flows in the first half of the state of the art, the second half explains the required basics for applying the Koopman operator in section 2.2 Since the Koopman operator by composition with the flow map of a dynamical system, we first have to provide a quick introduction of the basics of dynamical systems in section 2.2.1. Then, section 2.2.2 briefly defines operators and their spectrum, so we can formally introduce the Koopman operator, which forms the basis of our analysis approach. As previously mentioned, the Koopman operator requires a dynamical system. Because of this we present the dynamical system view on algorithms originally introduced by Dietrich et al. [8] in section 2.2.3. Then, the only thing remaining for section 2.2.4 is the introduction of EDMD, an algorithm for finding a finite dimensional approximation of the Koopman operator, its eigenvalues and eigenfunctions. This allows the practical application of Koopman Operator Theory without the requirement of explicit derivation of the Koopman operator.
In chapter 3 we present out experiments and results. Our first contribution comes in section 3.1, where we properly formalize Kernel Flows as a dynamical system, which allows for the analysis of Kernel Flows through the Koopman operator. Then, we explain in detail our approach to analysing Kernel Flows through the Koopman operator in section 3.2. To this end, we first provide a detailed explanation of the variables and design choices for the approximation of the Koopman operator in section 3.2.1. Next, in section 3.2.2 we exhaustively describe our approach to detecting the basins of attraction using the Koopman operator and illustrate everything with the example provided by Dietrich et al. [8]. The result are two major contributions of our own: First, we verify the results of Dietrich et al. [8]. Second, we show that the fact that the correct number of basins of attraction is not known, can cause noticeable problems. In section 3.3 we further investigate the convergence of Kernel Flows with multiple parameters and, thus, extend the analysis of Darcy [7]. Additionally, we compare the influence of different optimizers on the basins of attraction. After the analysis on synthetic data, we investigate Kernel Flows on real two real datasets in section 3.4.
Chapter 4 closes this thesis by discussing the results and possible implications in section 4.1 and providing an outlook on future work in section 4.2.

2 State of the Art

This chapter provides the mathematical background for the analysis of the Kernel Flows algorithm through the Koopman operator. The algorithm under analysis in this thesis is Kernel Flows which aims at estimating an ideal kernel to a given dataset. Section 2.1.1 will provide relevant background knowledge for kernels and provide a brief explanation on why kernels are relevant. Then, section 2.1 will introduce Kernel Flows. Next, we explain the background for our analysis through the Koopman operator. To this end, section 2.2 will provide a short introduction into dynamical systems, the Koopman operator, and how methods of dynamical system analysis can be applied for the analysis of algorithms. Section 2.2.4 explains Extended Dynamic Mode Decomposition (EDMD), a data-driven approach to estimating the Koopman operator, which allows for the usage of approximate models of algorithms instead of requiring manually derived analytical ones.

2.1 The Kernel Flow Algorithm

This section introduces the Kernel Flow algorithm [24], which this thesis analyses using the Koopman operator. This algorithm is a data-driven approach to finding a 'good' kernel for a problem at hand. So, section 2.1.1 will first formally introduce kernels and explain their use in practical application. Then, section 2.1.2 gives examples of kernels, we will use later on in this thesis. In section 2.1.3 we present the notion of what a 'good' kernel as defined by Owhadi&Yoo [24] and follow up with an explanation of Kernel Flows in section 2.1.4.

2.1.1 Kernel Regression

The field of kernel regression is well studied and a comprehensive summary would exceed the scope of this thesis. Thus, we limit ourselves to the basics required for understanding kernels and their utility in regression. For more details we refer to Bishop [2] and Schölkopf [28].
Let $\mathcal{X}, \mathcal{Y}$ be metric spaces and $\mathcal{D}_{kernel} := \{(x_i, y_i) | (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, ..., N\}$ be a dataset with $N \in \mathbb{N}$ datapoints. Typically, we have $\mathcal{Y} \subset \mathbb{R}$ with standard topology and induced norm and metric. The goal of regression tasks is to find a function $f : \mathcal{X} \to \mathcal{Y}$ mapping observations $x \in \mathcal{X}$ to target values $y \in \mathcal{Y}$. It is also commonly assumed that the target value contains some Gaussian noise $\epsilon \sim N(0, \sigma^2)$, where σ is the standard deviation of the noise. So, for $i = 1, ..., N$ the data y_i follows

$$y_i = f(x_i) + \epsilon. \tag{1}$$

The simplest models for f would be linear, meaning y_i would just be a linear combination of the elements contained in the vector x_i. Unfortunately, this simple approach imposes limitations on the expressive power of the model. To solve this, a set of basis functions $\phi_j : \mathcal{X} \to \mathcal{Y}$ can be introduced. Since the basis functions are not necessarily linear, this allows to first apply a non-linear function to map the input in a feature space, and then consider the target value as a function of elements in the feature space. For the set of basis function we introduce the vector notation ϕ with

$$\phi(x)_j := \phi_j(x). \tag{2}$$

Using the basis functions we can also define the Gram matrix $\Theta \in \mathbb{R}^{N \times N}$ by

$$\Theta_{i,j} := \phi(x_i)^T \phi(x_j). \tag{3}$$

It should be noted that if points x_i, x_j never coincide for $i \neq j$, then Θ is a symmetric and positive-definite matrix. Next, we define a kernel as a similarity measure

$$k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}. \tag{4}$$

If

$$k(x, x') = k(x', x) \tag{5}$$

holds for arbitrary $x, x' \in \mathcal{X}$, the kernel is considered symmetric. A kernel k is considered positive-definite if the matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ is positive-definite, where $\mathbf{K}$ is defined by

$$\mathbf{K}_{i,j} := k(x_i, x_j). \tag{6}$$

This means, that the Gram matrix Θ defined by the basis functions ϕ is a kernel and already provides the first use case for kernels: If in a model the input $x \in \mathcal{X}$ is only used in an inner product with itself, this inner product can be replaced by a kernel which represents the usage of some base functions $k(x_i, y_i) = \langle \phi(x_i), \phi(x_j) \rangle$. This approach is commonly referred to as the Kernel trick.

2.1.2 Examples of Kernels

While section 2.1.1 provides a mathematical definition and explanation of kernels in general, this section gives some examples of commonly used kernels. The examples we will consider are the Gaussian kernel, the n-linear Gaussian kernel, and the rational quadratic kernel which are also used in the analysis of Kernel Flows by Darcy [7]. Before formally introducing the aforementioned kernels, it is also worth noting that Darcy selected these as examples, since they are universal kernels. Simplified, universal kernels are continuous kernels capable of approximating an arbitrary continuous function f on a compact subset of the domain of f, but since for this thesis the details of universal kernels are not of importance, we refer to Micchelli et al. [21] for a proper mathematical description and associated proofs.
The first kernel we introduce is the Gaussian kernel. For arbitrary x, x' in some normed space $\mathcal{X}$ the Gaussian kernel is given by

$$k(x, x') := \exp(-\frac{\|x - x'\|^2}{2\sigma^2}), \tag{7}$$

where $\sigma > 0$ is some parameter, which has to be selected.
Since finite sums of kernels and multiplications of kernels with a scalar $c > 0$ result in new valid kernels [2], it is possible to define the n-linear Gaussian kernel based on the Gaussian kernel by

$$k(x, x') := \sum_{i=1}^{n} \beta_i^2 \exp(-\frac{\|x - x'\|^2}{2\sigma_i^2}). \tag{8}$$

Here, $\forall i = 1, ..., n : \sigma_i > 0$ as for Gaussian kernels and $\beta_i \in \mathbb{R}$.
The third and last example of a kernel is the rational quadratic kernel. For parameters $\alpha, \beta, \gamma > 0$ it is defined by

$$k(x, x') := (\beta^2 + \gamma \|x - x'\|)^{-\alpha}. \tag{9}$$

2.1.3 Measuring Kernel Quality

Kernel Flows focuses on finding a 'good' kernel for a problem at hand, so in order to compare different candidates of kernels Owhadi&Yoo [24] introduced a function to measure the quality of kernels. This section will present this function and explain its rational.
The base idea of the approach introduced by Owhadi&Yoo is, that a kernel is 'good' if a reduction in the number of data points used to fit a kernel from N_{kernel} to $\frac{N_{kernel}}{2}$ does not result in a large change in the optimal recovery. Colloquially, this would mean that 'the kernel generalizes well' when using $\frac{N_{kernel}}{2}$ points is sufficient. Since the computation of the optimal recovery is essential to this approach, we will cover the subject first. So, let

$$\mathcal{D}_{kernel} := \{(x_i, y_i) | i = 1, ..., N_{kernel}\} \subset \mathcal{X} \times \mathcal{Y} \tag{10}$$

be a dataset, where $\mathcal{X}, \mathcal{Y}$ are some metric spaces and $N_{kernel} \in \mathbb{N}$ is the number of datapoints in the dataset. Then, the problem of finding an unknown function u^+ interpolating the dataset can be formulated as the minimizer of the relative error

$$\min_{\substack{v \in \mathcal{B}, \\ \forall 1 \le i \le N_{kernel} : v(x_i) = u(x_i)}} \max_{u \in \mathcal{B}} \frac{\|u - v\|^2}{\|u\|^2}, \tag{11}$$

where $\mathcal{B}$ is a set of functions to which we are restricted. Let $\mathcal{B}^*$ be the dual space of $\mathcal{B}$ and $\phi_i(\cdot) := \delta(\cdot - x_i) \in \mathcal{B}^*$ be delta Dirac functions. If the norm $\|\cdot\|$ is quadratic, then for $u \in \mathcal{B}$

$$\|u\|^2 = \langle Q^{-1} u, u \rangle, \tag{12}$$

where $Q : \mathcal{B}^* \to \mathcal{B}$ is a positive symmetric linear bijection and $\langle \phi, u \rangle$ denotes the duality product for $\phi \in \mathcal{B}^*, u \in \mathcal{B}$. Further, let Θ be the $N_{kernel} \times N_{kernel}$ Gram matrix with entries $\Theta_{i,j} = \langle \phi_i, Q\phi_j \rangle$ and $A = \Theta^{-1}$ be its inverse. Then, for the special case of a quadratic norm, the minimizer for (11) is given by

$$v^+ = \sum_{i,j=1}^{N_{kernel}} y_i A_{i,j} Q\phi_j. \tag{13}$$

With Eq. 13 we have a solution for the case that the norm $\|\cdot\|$ is quadratic, but since the intention is to apply the results to finding an optimal kernel, we will explain the connection between these results and kernels next. To this end, consider the kernel

$$K(x, x') := \langle \delta(\cdot - x), Q\delta(\cdot - x') \rangle. \tag{14}$$

Then, with the norm $\|\cdot\|$ defined by

$$\|u\|^2 = \sup_{\phi \in \mathcal{B}^*} \frac{\int \phi(x) u(x) dx)^2}{\int \phi(x) K(x, y) \phi(y) dx dy} \tag{15}$$

for $u \in \mathcal{B}$ the normed space $(\mathcal{B}, \|\cdot\|)$ can be considered a Reproducing Kernel Hilbert Space. By inserting the definition of the kernel K given in Eq. 14 into Eq. 13 we have

$$v^+(\cdot) = y^T AK(x, \cdot), \tag{16}$$

where $y^T AK(x, \cdot) = \sum_{i,j=1}^{N_{kernel}} y_i A_{i,j} K(x_j, \cdot)$, $A = \Theta^{-1}$, and $\Theta_{i,j} = K(x_i, x_j)$. Note that this is precisely the definition of the prediction formula of expectation of a Gaussian Process Model.
The next step for evaluating the quality of a kernel is to compare the optimal recovery for the entire dataset to the optimal one on a subset. Thus, let $m \in \mathbb{N}_{\neg\setminus\ltimes\lessdot}, 1 \leq m \leq N_{kernel}$ be the number of datapoints to sample for an estimation and $\{s_1, s_2, ..., s_m\} \subset \{1, ..., N_{kernel}\}$ a selection of m distinct elements. Then, we define the sample dataset $\mathcal{D}_s \subset \mathcal{D}_{kernel}$ as

$$\mathcal{D}_s := \{(x_{s_i}, y_{s_i}) | i = 1, ..., m\}. \tag{17}$$

On this sample dataset $\mathcal{D}_s$ we can determine another optimal recovery v^s given by

$$v^s(\cdot) = \sum_{i=1,j}^{m} y_{s_i} \bar{A}_{i,j} K(x_{s_j}, \cdot), \tag{18}$$

where $\bar{A} = \bar{\Theta}^{-1}$ and $\bar{\Theta}_{i,j} = \Theta_{s_i, s_j}$. Then, the quality of a kernel K is measured by the ratio

$$\rho := \frac{\|v^+ - v^s\|^2}{\|v^+\|^2}. \tag{19}$$

As shown by Owhadi&Yoo [24], the ratio ρ can also be represented by

$$\rho = 1 - \frac{y^T \bar{A} y}{y^T A y}. \tag{20}$$

From this representation it is clear, that $\rho \in [0, 1]$ and ρ close to zero indicates that v^s and v^+ are similar and, therefore, we the kernel K generalizes well and can be considered good.

2.1.4 Kernel Flows

With the quality measure presented in section 2.1.3 we now introduce Kernel Flows with a parametric family of kernels. For this we will follow closely the work of Owhadi&Yoo [24]. It should be noted that there also exists a non-parametric variant of the Kernel Flows algorithm, which will not be covered in this section.
The parametric Kernel Flows algorithm considers ρ as a function of the kernel parameters and then minimizes it using gradient descent methods in order to find an optimal kernel in a parameterized family of kernels. So, first we denote the finite dimensional linear space of parameters by $\mathcal{W}$. Then, let $K(x, x', W)$ be a family of kernels parameterized by $W \in \mathcal{W}$. As previously, let $\mathcal{X}, \mathcal{Y}$ be metric spaces and let $N_{kernel} \in \mathbb{N}$ be the number of datapoints in a dataset

$$\mathcal{D}_{kernel} := \{(x_i, y_i) | \forall i = 1, ..., N_{kernel} : (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}. \tag{21}$$

Let $N_b \leq N_{kernel}$ be the batch size used for gradient estimation and $N_c := round(\frac{N_b}{2})$ be the number of datapoints used for estimating the quality of the kernel using ρ. By $\{s_{b,1}, ..., s_{b,N_b}\} \subset \{1, ..., N_{kernel}\}$ we denote the selection of N_b distinct datapoints in the current batch, by $\{s_{c,1}, ..., s_{c,N_c}\} \subset \{1, ..., N_b\}$ we denote the selection of N_c datapoints used to estimate the quality of the kernel in ρ. We define the $N_c \times N_b$ sub-sampling matrix π by $\pi_{i,j} := \delta_{s_{c,i},j}$ and the sub-sampled target vectors $y_b \in \mathbb{R}^{N_b}, y_c \in \mathbb{R}^{N_c}$ by $y_{b,i} := y_{s_{b,i}}, y_{c,i} := y_{s_{c,i}}$, respectively. In the formulation of ρ in Eq. 20 the kernel is implicitly contained since $A = \Theta^{-1}$, $\bar{A} = (\pi\Theta\pi^T)^{-1}$, and $\Theta_{i,j} = K(x_i, x_j)$. So, when we consider the function $\Theta(W) : \mathcal{W} \to \mathbb{R}^{N_b \times N_b}$ with $\Theta_{i,j}(W) = K(x_{s_{b,i}}, x_{s_{b,j}}, W)$ and insert it in the definition of ρ, we have

$$\rho(W, s_b, s_c) = 1 - \frac{\|y_c^T(\pi\Theta(W)\pi^T)^{-1}y_c\|^2}{\|y_b^T\Theta(W)^{-1}y_b\|^2}. \tag{22}$$

Thus, ρ can be considered a function of W, the parameter of the kernel. This allows to use gradient descent methods to minimize ρ and, therefore, find an optimal kernel in the parameterized family of kernels. The parametric variant of Kernel Flows is summarized in algorithm 1.

Algorithm 1 Pseudo-code of parametric Kernel Flows

```
Initialize $W$
while Not end criterion do
    Select $s_{b,1}, .., s_{b,N_b}$ out of $\{1, ..., N_{kernel}\}$
    Select $s_{c,1}, .., s_{c,N_c}$ out of $\{1, ..., N_b\}$
    $W \leftarrow W - \epsilon\nabla_W\rho(W, s_b, s_c)$
end while
```

2.2 The Koopman Operator of Algorithms

The Koopman operator has been initially introduced by Koopman [16] in 1931 in the context of hamiltonian systems and Hilbert spaces. It allows for analysis of non-linear dynamical systems using a linear operator. In the decades since, noticeable progress in the theory of the Koopman operator has been made.

The Renaissance of the Koopman operator has been brought about by Igor Mezić [18, 19, 20] by introducing majorly to extending the theoretical background and making the Koopman Operator Theory practically viable.

Črnjarić-Žic et al. [6] analysed the properties of the Koopman operator for different types of random dynamical systems. Further, they defined a stochastic Hankel DMD allowing for numerical approximation of eigenvalues and eigenfunctions of the stochastic Koopman operator. In addition to proving the convergence of their algorithm, they utilise it on examples for model reduction.

Nüske et al. [23] showed probabilistic error bounds for prediction error and approximation error depending on the number of training points.

Alexander&Giannakis [1] utilise the Koopman operator framework to reformulate kernel analog forecasting techniques.

Dogra&Redman [9] consider the optimisation of weights during the training of a neural network as a dynamical system and use Koopman operator theory to develop a new training technique for faster training than existing gradient-based approaches like ADAM or SGD.
For a more detailed overview on existing literature, we refer to Budišić et al. [5].
Gonzalez et al. [11] disprove common misconceptions regarding the Koopman operator. This includes attributes of the Koopman operator like boundedness or compactness. Further, they show that, in general, there does not exist a lattice of eigenfunctions and eigenfunctions are not necessarily shared among Koopman operators.

2.2.1 An Introduction to Dynamical Systems

The fundamental motivation behind dynamical systems is to model and study the behaviour of evolving systems over a long time period. Due to its prevalence in, for example, physics, this topic is well studied in literature. Nevertheless, for this thesis the basic concepts are sufficient and will be introduced here. For a more comprehensive introduction we refer to Brin&Stuck [3].
First, let $\mathcal{M} \neq \emptyset$ be a non-empty set. We refer to $\mathcal{M}$ as the state space and every element $x \in \mathcal{M}$ is called a state. Second, we define a one-parameter family of maps $\mathcal{F}_t := \{f^t : \mathcal{M} \to \mathcal{M}\}$, which forms a one-parameter group or semigroup. This means that $f^0 = I$, where I is the identity function, and

$$\forall t, s : f^{t+s} = f^t \circ f^s. \tag{23}$$

If t ranges over $\mathbb{R}$, then the map f associated with a dynamical system is called a flow. If t ranges over $\mathbb{R}_0^+$, then it is called a semiflow.
While this is the most general definition of dynamical system, for the purpose of this thesis, we will limit ourselves to dynamical systems based on differential equations and follow the terminology of Dietrich et al. [8]. So, here the state space $\mathcal{M}$ will be a smooth, k-dimensional Riemannian submanifold embedded in a d-dimensional Euclidean space. Further, we define a differentiable vector field $v : \mathcal{M} \to \mathbb{R}^k$. The flow induced by this vector field is given by a map $S : \mathbb{R}^+ \times \mathcal{M} \to \mathcal{M}$, for which holds

$$\forall s, t \in \mathbb{R}^+, \forall x \in \mathcal{M} : S(t+s, x) = S(t, S(s, x)), \left. \frac{d}{dt} S(t, x) \right|_{t=0} = v(x). \tag{24}$$

Using the flow map S and defining for each $t \in \mathbb{R}^+$ a new map S_t by

$$\forall x \in \mathcal{M} : S_t(x) = S(t, x), \tag{25}$$

we have a family of flow maps $\mathcal{F}_t$. From the conditions in Eq. 24 it follows that $\mathcal{F}_t$ is a semigroup. Thus, this approach is a special case of the original definition of dynamical systems. For the remainder of this thesis, we will work with dynamical systems based on vector fields.
In section 2.2.3 we will show how algorithms can be considered dynamical systems. While there are continuous algorithms, the main focus in this thesis is on iterative algorithms which work in discrete time. This can be achieved easily by discretizing the flow map S. To this end, we fix a $\Delta t \in \mathbb{R}^+$ and define $S_{\Delta t}(x) = S(\Delta t, x)$ for arbitrary $x \in \mathcal{M}$.

Then, $S_{\Delta t}$ generates a dynamical system which only allows for time steps of size Δt and is, therefore, a time-discrete and time-homogeneous dynamical system. Note that the inverse direction is not possible in general: some iterative algorithms do not have a continuous counterpart. A good example of this is the logistic map. Its iterate produce a chaotic sequence in one dimension, but there cannot be a dynamical system in continuous time that is chaotic in just one variable.

2.2.2 The Koopman Operator

The fundamental basis for the analysis of Kernel Flows presented in this thesis is the Koopman operator. The following introduces the mathematical setting.
In functional analysis a continuous linear operator $T : \mathcal{X} \to \mathcal{Y}$ is a linear and continuous map between two normed spaces $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$. In the context of this thesis, we will further assume that the normed spaces $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ are Banach spaces. Due to the linearity of operators the study of operators in functional analysis can be considered a generalisation of linear algebra, but unlike linear algebra functional analysis is not limited to finite dimensional spaces.
One important similarity to linear algebra is the study of the spectrum of operators. In linear algebra, the study of the spectrum refers to the analysis of the eigenvalues and eigenvectors of a matrix. For a given field $\mathbb{F}$ a value $\lambda \in \mathbb{F}$ is an eigenvalue of a matrix $T \in \mathbb{F}^{n \times n}$, if there exists a vector $x \in \mathbb{F}^n \setminus \{0\}$ with

$$Tx = \lambda x, \tag{26}$$

which holds if and only if

$$\lambda I - T \text{ is not injective,} \tag{27}$$

where I is the identity matrix. In finite dimensional spaces $\lambda I - T$ not injective holds if and only if $\lambda I - T$ is not surjective. In infinite dimensional spaces this equivalence does not hold, so the spectrum is partitioned into three parts, the point spectrum, the continuous spectrum, and the residual spectrum. So, let $\mathbb{F}$ be a field and $\mathcal{X}$ be a - possible infinite dimensional - vector space over $\mathbb{F}$. For an operator $T : \mathcal{X} \to \mathcal{X}$, the point spectrum is defined by

$$\sigma_p(T) := \{\lambda \in \mathbb{F} : \lambda I - T \text{ is not injective}\}. \tag{28}$$

The continuous spectrum is given by

$$\sigma_c(T) := \{\lambda \in \mathbb{F} : \lambda I - T \text{ is injective but not surjective with dense range}\} \tag{29}$$

and the residual spectrum by

$$\sigma_r(T) := \{\lambda \in \mathbb{F} : \lambda I - t \text{ is injective but not surjective without dense range}\}. \tag{30}$$

The spectrum of the operator $T : \mathcal{X} \to \mathcal{X}$ is given by the disjoint union

$$\sigma(T) := \sigma_p(T) \cup \sigma_c(T) \cup \sigma_r(T). \tag{31}$$

For the analysis in this thesis we will focus on the point spectrum. Usually, the point spectrum $\sigma_p(T)$ of an operator $T : \mathcal{X} \to \mathcal{X}$ consists of a union of discrete points. It should also be noted, that only $\lambda \in \sigma_p(T)$ is considered an eigenvalue of T and an vector $x \in \mathcal{X} \setminus \{0\}$ is called an eigenvector. If $\mathcal{X}$ is a function space it is common to use the term eigenfunction instead of eigenvector for $x \in \mathcal{X}$.
In the context of dynamical systems the definition of a family of operators by composition with flow maps is possible and commonly referred to as Koopman operator. The function space on which the Koopman operator operates on is referred to as the space of observables and every element in this function space is called an observable. In this thesis, we limit ourselves to a common choice, we will now formally introduce. So, let $L^2(\mathcal{X}, \mathbb{C}, \mu)$ be the function space of complex-valued functions which are square integrable with respect to a measure μ. For simplicity, we will denote this space by

$$\mathcal{F} = L^2(\mathcal{X}, \mathbb{C}, \mu) = \{g : \mathcal{X} \to \mathbb{C}, s.t. \int_{\mathcal{X}} |g(x)|^2 d\mu(x) < \infty\}. \tag{32}$$

The square integrability of this function space provides an intrinsic inner product defined by

$$\langle g_1, g_2 \rangle := \int_{\mathcal{X}} g_1(x)\overline{g_2(x)} d\mu(x), \tag{33}$$

where the bar over g_2 denotes the complex conjugation of the function evaluation. Since $\mathcal{F}$ is technically a space of equivalence classes with respect to μ, meaning every function in an equivalence class only differs in a μ-null set, this inner product induces a norm on $\mathcal{F}$, here denoted by $\|\cdot\|_{\mathcal{F}}$. So, $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ forms a normed space.
For the definition of the Koopman operator $\mathcal{K}^t$ for $t \in \mathbb{R}^+$ we require a flow map. So, let $\mathcal{F}_t = \{S_t : t \in \mathbb{R}\}$ be a flow as defined in section 2.2.1. Further, let $g : \mathcal{X} \to \mathbb{C} \in \mathcal{F}$ be an observable. Then the Koopman operator $\mathcal{K}^t : \mathcal{F} \to \mathcal{F}$ is defined by

$$[\mathcal{K}^t g](x) := (g \circ S_t)(x). \tag{34}$$

Since the Koopman operator $\mathcal{K}^t$ is parameterised by t, it is clear, that we actually have a family of operators and not just a single one. It is common in literature to choose one parameter t_0 and then to only consider $\mathcal{K}^{t_0}$ and analyse this operator. If not specified otherwise, we will work with $t_0 = 1$ and, for notational convenience, drop the superscript and only write $\mathcal{K}$.
This covers the formal introduction of the Koopman operator and, as can be seen from the definition, intuitively, the Koopman operator $\mathcal{K}^t$ can be interpreted as shifting the observable through the dynamical system forward in time by t. This means, instead of evaluating an observable $g : \mathcal{X} \to \mathbb{C} \in \mathcal{F}$ at $x \in \mathcal{X}$, it will be evaluated at the state reached after t time has passed in the dynamical system, when we start at state $x \in \mathcal{X}$. It should be noted, that there also exists a notion of a stochastic Koopman operator which takes

2.2.3 A Dynamical System Perspective on Algorithms

The idea to consider algorithms as dynamical systems and apply the Koopman operator framework for analysis was first suggested by Dietrich et al. [8] and stems from the observation that many iterative algorithms act upon a finite-dimensional state space. Further, future states only depend on the current state and time passed. Thus, they can be considered as a variant of a time-discrete dynamical system. We formalise this by first defining a state space $\mathcal{M} \subseteq \mathbb{R}^d$, where $d \in \mathbb{N}$. Further, we assume that $\mathcal{M}$ is a smooth, k-dimensional Riemannian submanifold embedded in an d-dimensional Euclidean space, where $d \in \mathbb{N}, d \geq k$ holds. The embedding induces the metric on $\mathcal{M}$. Then, an iterative algorithm is defined by a differentiable map $u : \mathcal{M} \to \mathcal{M}$, which steps forward in iteration number. For a given iteration number $n \in \mathbb{N}$ and corresponding state $x_n \in \mathcal{M}$, a single iteration is defined by

$$x_{n+1} = u(x_n). \tag{35}$$

By considering the map $u : \mathcal{M} \to \mathcal{M}$ as a flow map, we can use the definition of dynamical systems induced by vector fields as elaborated on in section 2.2.1. This approach provides us with a flow map $S_{\Delta t} : \mathcal{M} \to \mathcal{M}$. While we could assign an arbitrary time interval Δt in the discrete time dynamical system to a single update step in the iterative algorithm, for simplicity, we will use $\Delta t = 1$.
Now, that we have reformulated an iterative algorithm as a dynamical system, we can deploy methods for dynamical system analysis. In this thesis, we will mainly focus on the Koopman operator introduced in section 2.2.2.

2.2.4 Data-driven Estimation of the Koopman Operator

Section 2.2 has introduced the Koopman operator for a dynamical system, but so far this is a strictly formal definition and the application of the Koopman algorithm would require explicit calculation of the Koopman operator for a given dynamical system. This is only possible for a very limited number of systems and not applicable in practice. In order to solve this problem several approximative algorithms have been proposed over time. One example is the stochastic Hankel DMD introduced by Črnjarić-Žic et al. [6] which allows for an approximation of a stochastic Koopman operator and its spectral elements. In this thesis we will use extended dynamic mode decomposition (EDMD) for a data driven approximation of the Koopman operator. It was first proposed by Williams et al. [29] in order to approximate the leading eigenvalues and eigenfunctions of the Koopman operator given a dataset of snapshot pairs and a dictionary of scalar observables. Williams et al. further showed that EDMD can be considered an extension of dynamic mode decomposition [27], a previously existing algorithm for approximating the eigenvalues of the Koopman operator, and that for Markov processes the algorithm approximates the eigenfunctions of the stochastic variant of the Koopman operator. Other approximation algorithms are also available. For example the stochastic Hankel DMD introduced by Črnjarić-Žic et al. [6] which allows for an approximation of a stochastic variant of the Koopman operator and its spectral elements.
The fundamental idea of EDMD is to find a finite dimensional approximation $\mathbf{K}$ of the Koopman operator $\mathcal{K}$ using a dataset of snapshots of the dynamical system $\mathcal{D}_{snap}$ and a dictionary of observables $\mathcal{D}_{obs}$. So, let $\mathcal{M}$ be a state space, $S_t : \mathcal{M} \to \mathcal{M}$ be the flow map of a dynamical system with step size t, $\mathcal{F}$ be a space of observables, and $\mathcal{K}$ be the Koopman operator defined by composition with S_t. Further, let

$$\mathcal{D}_{snap} = \{(x_i, y_i) | x_i, y_i \in \mathcal{M}, y_i = S_t(x_i), \forall i = 1, ..., N_{snap}\} \tag{36}$$

be a dataset of snapshots of the dynamical system and

$$\mathcal{D}_{obs} = \{\psi_i | \psi_i \in \mathcal{F} \forall i = 1, ..., N_{obs}\} \tag{37}$$

be a dictionary of observables. Here, $N_{snap} \in \mathbb{N}$ is the number of snapshots in the dataset and $N_{obs} \in \mathbb{N}$ is the number of observables in the dictionary. For brevity, we use $\mathcal{F}_{\mathcal{D}} \subset \mathcal{F}$ to refer to the span $\langle \mathcal{D}_{obs} \rangle$ of the dictionary of observables $\mathcal{D}_{obs}$. It should be noted that the span $\mathcal{F}_{\mathcal{D}}$ of the dictionary of observables $\mathcal{D}_{obs}$ does not necessarily span the entire space of observables $\mathcal{F}$, but for EDMD we will assume that $\mathcal{F}_{\mathcal{D}}$ contains a sufficiently good approximation of the leading eigenfunctions of the Koopman operator $\mathcal{K}$. Additionally, we introduce

$$\Psi : \mathcal{M} \to \mathbb{C}^{1 \times N_{obs}}, x \mapsto [\psi_1(x), \psi_2(x), .., \psi_{N_{obs}}(x)] \tag{38}$$

as a helper function to simplify notation.
First, we explain the approximation of the eigenfunctions. To this end, we first note that by definition an element $\phi \in \mathcal{F}_{\mathcal{D}}$ can be represented as

$$\phi = \sum_{i=1}^{N_{obs}} a_i \psi_i = \Psi \mathbf{a}, \tag{39}$$

where $\mathbf{a} \in \mathbb{C}^{N_{obs}}$ is a vector of coefficients. Then, since $\mathcal{F}_{\mathcal{D}}$ is in general not an invariant subspace of the Koopman operator $\mathcal{K}$, we can represent the effect of applying the Koopman operator $\mathcal{K}$ on an observable ϕ by

$$\mathcal{K}\phi = (\Psi \circ S_t)\mathbf{a} = \Psi \cdot \mathbf{K}\mathbf{a} + r \tag{40}$$

for some $\mathbf{K} \in \mathbb{C}^{N_{obs} \times N_{obs}}$ and a residual term $r \in \mathcal{F}$. In Eq. 40 we separated the application of the Koopman operator into two parts. Firstly, $\Psi(\mathbf{K}\mathbf{a})$, which lies within $\mathcal{F}_{\mathcal{D}}$ and, secondly, $r \in \mathcal{F}$ which isn't contained in $\mathcal{F}_{\mathcal{D}}$. So, if we choose $\mathbf{K}$ in a way that minimizes the residual term r, we can consider $\Psi(\mathbf{K}\mathbf{a})$ to be the best approximation of the Koopman operator. To this end we consider the objective function

$$J := \frac{1}{2} \sum_{i=1}^{N_{snap}} |r(x_i)|^2 \tag{41}$$

$$= \frac{1}{2} \sum_{i=1}^{N_{snap}} |\mathcal{K}\phi(x_i) - \Psi(x_i)(\mathbf{K}\mathbf{a})|^2 \tag{42}$$

$$= \frac{1}{2} \sum_{i=1}^{N_{snap}} |((\Psi \circ S_t)(x_i) - \Psi(x_i)\mathbf{K})\mathbf{a}|^2 \tag{43}$$

$$= \frac{1}{2} \sum_{i=1}^{N_{snap}} |(\Psi(y_i) - \Psi(x_i)\mathbf{K})\mathbf{a}|^2. \tag{44}$$

For the first and second equality we resolved Eq. 40 for r. For the third one we used that in our dataset $\mathcal{D}_{snap}$ by definition $y_i = S_t(x_i)$ holds for every $i = 1, ..., N_{snap}$. Now, let

$$\mathbf{G} = \frac{1}{N_{snap}} \sum_{i=1}^{N_{snap}} \Psi(x_i)^* \Psi(x_i), \tag{45}$$

$$\mathbf{A} = \frac{1}{N_{snap}} \sum_{i=1}^{N_{snap}} \Psi(x_i)^* \Psi(y_i), \tag{46}$$

where $\Psi(x_i)^*$ denotes the conjugate transpose of $\Psi(x_i)$. Further, we denote the pseudo-inverse of $\mathbf{G}$ by $\mathbf{G}^+$. Then, since J is a least squares problem, the optimal solution for $\mathbf{K}$ is given by

$$\mathbf{K} = \mathbf{G}^+\mathbf{A}. \tag{47}$$

Now, with an optimal finite dimensional approximation $\mathbf{K} \in \mathbb{C}^{N_{obs} \times N_{obs}}$ of the Koopman operator $\mathcal{K}$, we can consider consider the approximation of eigenvalues and eigenfunctions. So, let ξ_i be the eigenvector of $\mathbf{K}$ to the i-th eigenvalue λ_i, then the approximation of the i-th eigenfunction φ_i of $\mathcal{K}$ is given by

$$\varphi_i = \Psi \xi_i. \tag{48}$$

While an approximation of the eigenvalues and the eigenfunctions of the Koopman operator are useful for spectral analysis, they are not sufficient to predict future states of the dynamical system. To this end, we consider the full state observable to capture the effect of the dynamical system on the state space using the Koopman operator. The full state observable is defined as

$$g : \mathcal{M} \to \mathcal{M} \tag{49}$$

$$g(x) = x. \tag{50}$$

Since the space of observables contains functions mapping from $\mathcal{M}$ to $\mathbb{C}$, we have to express the full state observable using component functions

$$g_i : \mathcal{M} \to \mathbb{C}, g_i(x) = e_i^* x, \tag{51}$$

where e_i denotes the i-th unit vector in $\mathbb{R}^N$. Then, each component function can be approximated using the observables in the dictionary $\mathcal{D}_{obs}$ and the weights be determined. For practical application it is common to first represent the eigenfunctions using the dictionary $\mathcal{D}_{obs}$ and then represent the components of the full state observable as a weighted sum of eigenfunctions. In this case, the weights are referred to as Koopman modes, but since we will not use the Koopman modes for our analysis, we refer to Williams et al. [29] for a detailed explanation.

3 Analysis of Kernel Flows through the Koopman Operator

This chapter details the analysis of Kernel Flows through the Koopman operator and, thus, forms the main contribution of this thesis. First, we properly formalise Kernel Flows as a dynamical system in section 3.1 by applying the general idea of viewing algorithms as dynamical systems presented in section 2.2.3 to Kernel Flows presented in section 2.1.4. Then, in section 3.2 we provide the details on the approximation of the Koopman operator, methods to analyse a dynamical system through this approximation of the Koopman operator, and briefly detail the implementation. After this explanation of the analysis in general, in section 3.3 we deploy it on synthetic data in order to investigate the influence of hyper-parameters and the behaviour of Kernel Flows. Section 3.4 further analysis the behaviour of Kernel Flows, when applied on real datasets.

3.1 Kernel Flows as a Dynamical System

Before we can analyse Kernel Flows through the Koopman operator, we have to specify how we can view them as dynamical systems. While section 2.2.3 presents the fundamental idea for viewing algorithms as dynamical systems, this section will provide the specifics for Kernel Flows with a parametric family of kernels. It should be noted, that some variables will still be defined somewhat vague, since they are dependent on the specific scenario in which we analyse Kernel Flows through the Koopman operator.
First, we define two metric spaces $\mathcal{X}, \mathcal{Y}$ and a dataset

$$\mathcal{D}_{kernel} = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y}, 1 \leq i \leq N\} \tag{52}$$

with $N \in \mathbb{N}$ datapoints. The metric space $\mathcal{X}$ is the feature space which will be used by the kernel to predict a target in the target space $\mathcal{Y}$. The dataset $\mathcal{D}_{kernel}$ will be used for Kernel Flows and, thus, corresponds to the dataset defined in Eq. 21. The index *kernel* is used to clearly mark the dataset as use for Kernel Flows and distinguish it from the dataset used to estimate the Koopman operator $\mathcal{K}$ using EDMD as presented in section 2.2.4. This dataset will be introduced later on. Both metric spaces $\mathcal{X}$ and $\mathcal{Y}$, the dataset $\mathcal{D}_{kernel}$, and the number of datapoints N depend on the specific experiment and, therefore, can not be specified any further at this place, but will be briefly explained for each experiment.
Next, we need to determine the family of kernels and the corresponding parameter space $\mathcal{W}$. The choice of the family of kernels is a research question in itself and not the topic of this thesis, so we use - depending on the experiment - either a family of kernels based on Gaussian kernels or rational quadratic kernels. This choice is mainly motivated in order to allow for comparison of the results to the previous analysis performed by Darcy [7]. Since the family of kernels depends on the experiment, the specifics will be explained for each experiment individually. Important for our formalisation of Kernel Flows as a dynamical system is that the algorithm updates the parameter space $\mathcal{W}$ with each iteration step. Therefore, the parameter space $\mathcal{W}$ will be considered the state space $\mathcal{M}$ of the dynamical system.
The last remaining step is to formally define the flow map $S : \mathcal{M} \rightarrow \mathcal{M}$ of the dynamical system. As explained in section 2.2.3 we use the update function $u : \mathcal{M} \rightarrow \mathcal{M}$ of the

algorithm to do so. In order to determine the update function u, we consider Kernel Flows as summarized in algorithm 1. In the first step a variable W for the current parameters of the kernel is initialized. Then, in a while loop this variable W is repeatedly updated by first selecting $1 < N_b \leq N$ datapoints with an index vector s_b from the dataset as a batch, then from these N_b datapoints $1 \leq N_c < N_b$ datapoints with index vector s_c are selected for determining the quality of the kernel. The last step is updating the variable W containing the kernel parameters by assigning the new value

$$W_{n+1} = W_n - \epsilon \nabla_{W_n} \rho(W_n, s_b, s_c). \tag{53}$$

Since the update function $u : \mathcal{M} \to \mathcal{M}$ we use has to be a function only in $\mathcal{M} = \mathcal{W}$, but the current update also depends on s_b and s_c an approach for removing the variables s_b and s_c from the domain of the update function is required. The simplest approach to this, would be to consider the index vectors s_b and s_c to be fixed. While simple, this approach would also result in Kernel Flows finding a kernel where optimal recovery for the sub-datasets of $\mathcal{D}_{kernel}$ selected by s_b and s_c, respectively, is similar. Each datapoint $(x_i, y_i) \in \mathcal{D}_{kernel}$ not selected by the index vector s_b would therefore be completely ignored. Since such a reduction of the dataset is not acceptable in general, this approach is unfit. Another potential approach is to define Kernel Flows as a random dynamical system. This requires a formal introduction of a probability space according to which the indices in s_b get selected from all indices in the dataset. A reasonable probability distribution would be uniform distribution on the set of indices in the dataset $\mathcal{D}_{kernel}$. Then, another probability space for the selection of indices s_c from the indices in s_b is required. Once again a uniform distribution on the set of previously selected indices would be reasonable. Then, these two probability spaces induce a conditional probability space on $\mathcal{W}$ which determines the distribution of the next value of W given its current one. The associated conditional probability distribution would be the replacement of the update function $u\mathcal{W} \to \mathcal{W}$ in a stochastic setting. Further, due to the introduction of the probability space related to a random dynamical system, the Koopman operator as defined in section 2.2.2 can not be used directly, but instead has to be adapted to a stochastic Koopman operator.
Since we do not see a significant benefit from the introduction of probability spaces, we instead model the update function $u : \mathcal{W} \to \mathcal{W}$ without any probability spaces on $\mathcal{W}$ and assume that s_b and s_c are the images of pseudo-random functions. This allows us to still model Kernel Flows as a deterministic dynamical system as opposed to a random one, while not fixing the selection of datapoints to only a few ones. In order to further justify this approach, we want to point out, that due to the selection of the state space $\mathcal{W}$ for our experiments a repeated visit of exactly the same state during a run of Kernel Flows is unlikely and as long as a state $W \in \mathcal{W}$ does not get revisited a random selection of s_b and s_c is indistinguishable from a pseudo-random one depending on W. We now formally introduce our approach. So, let

$$\mathcal{A}_b := \{s \in \{1, ..., N\}^{N_b} | \forall 1 \leq i, j \leq N_b : i \neq j \implies s_i \neq s_j\} \tag{54}$$

be the set of valid selections of N_b different indices from the dataset $\mathcal{D}_{kernel}$. Then, the selected indices s_b are the image of the pseudo-random function

$$r_b : \mathcal{W} \to \mathcal{A}_b. \tag{55}$$

Analogously, let

$$\mathcal{A}_c := \{s \in \{1, ..., N_b\}^{N_c} | \forall 1 \leq i, j \leq N_c : i \neq j \implies s_i \neq s_j\} \tag{56}$$

be the set of valid selections of N_c indices from an vector of dimension $N_b > N_c$ and

$$r_c : \mathcal{W} \to \mathcal{A}_c \tag{57}$$

be another pseudo-random function. Then, the index vector s_c selected for estimating the quality of the kernel with parameters W is given by

$$\begin{aligned} s_c &= \pi(s_b, r_c(W)) && (58) \\ &= \pi(r_b(W), r_c(W)), && (59) \end{aligned}$$

where

$$\begin{aligned} \pi : \mathcal{A}_b \times \mathcal{A}_c &\to \{1, ..., N\}^{N_c}, && (60) \\ \pi_i(x, y) &= x_{y_i} && (61) \end{aligned}$$

selects the elements of the first variable according to the indices provided by the second variable. As a result the parameter update assignment of Kernel Flows described in Eq. 53 can be reformulated as

$$\begin{aligned} W_{n+1} &= W_n - \epsilon \nabla_{W_n} \rho(W_n, s_b, s_c) && (62) \\ &= W_n - \epsilon \nabla_{W_n} \rho(W_n, r_b(W_n), \pi(r_b(W_n), r_c(W_n))). && (63) \end{aligned}$$

Here, it should be noted that we slightly abuse notation of the nabla operator. In our case $\nabla_{W_n} \rho(W_n, r_b(W_n), \pi(r_b(W_n), r_c(W_n)))$ still only refers to the Frechét derivative of ρ with respect to the first variable W_n and not the second and third one, which also depend on W_n now. As can be seen in Eq. 63, now the new set of parameters depends deterministically on the previous ones and we can define a deterministic update function u by

$$\begin{aligned} u : \mathcal{W} &\to \mathcal{W}, && (64) \\ W &\mapsto W - \epsilon \nabla_W \rho(W, r_b(W), \pi(r_b(W), r_c(W))). && (65) \end{aligned}$$

As explained in section 2.2.3 we now define the flow map S of the deterministic dynamical system by setting

$$S = u. \tag{66}$$

So, to summarize, we model Kernel Flows as a deterministic dynamical system where the state space is given by the parameter space $\mathcal{W}$ of the family of kernels used in Kernel Flows. The flow map of the dynamical system is the update function used in Kernel Flows to assign a new value to the current estimate of the best parameters for the kernel. The random selection of datapoints involved in the parameter update of Kernel Flows is modelled by using pseudo-random functions.

3.2 Analysis Methodology

This section explains the general approach for analysing Kernel Flows through the Koopman operator and uses gradient descent on Himmelblau's function [14] as an illustrative example. For the selection of the example, we have several reasons. First, it is the example provided by Dietrich et al.[8], which allows us to verify their results. Further, due to the fact that Himmelblau's function is well-known and well-established in numerical analysis, it allows for easier understanding of results like detection of basins of attraction than when performed on Kernel Flows. Third, since Dietrich et al.[8] already made use of this example, we do not have to properly formulate gradient descent as a dynamical system, but can refer interested readers to the work of Dietrich et al.[8].

3.2.1 Approximation of the Koopman Operator

Before we can explain the analysis of Kernel Flows through the Koopman operator, we first have to provide the specifics on how we use EDMD presented in section 2.2.4 to find a finite-dimensional approximation $\mathbf{K}$ of the Koopman operator $\mathcal{K}$. To this end, we first repeat some notation introduced in previous sections. So, as described in section 3.1, we consider Kernel Flows as a dynamical system. Therefore, $\mathcal{W}$ denotes the state space of the dynamical system and S is the flow map. Of this dynamical system we estimate a finite-dimensional approximation $\mathbf{K}$ of the Koopman operator $\mathcal{K}$ by using EDMD described in section 2.2.4. To this end, we require a dataset of snapshots of the dynamical system

$$\mathcal{D}_{snap} = \{(x_i, y_i) \in \mathcal{W} \times \mathcal{W} | 1 \leq i \leq N_s, S(x_i) = y_i\}, \tag{67}$$

where N_s is the number of snapshots we have. Here, we briefly want to remind that this dataset for approximating the Koopman operator $\mathcal{K}$ is ***not*** to be confused with the dataset $\mathcal{D}_{kernel}$ which is used in Kernel Flows. In order to generate the dataset $\mathcal{D}_{snap}$, we first select a connected, compact subset $\mathcal{W}_c$ of the parameter space $\mathcal{W}$, which is the subset of interest within we want to analyse our dynamical system. Since this subset of interest $\mathcal{W}_c$ depends on the parameter space $\mathcal{W}$, we provide the specifics for each experiment in the corresponding section. For the illustrative example we use the compact subset $[-4, 4]^2$ like Dietrich et al. [8]. After deciding on a subset of interest, we select N_s datapoints from $\mathcal{W}_c$, which are equidistant with respect to the maximum norm. On these datapoints, we evaluate the flow map S of our dynamical system. So, for our experiments this means we initialize Kernel Flows with a selected datapoint x_i as parameters and then calculate one update step of Kernel Flows to get the corresponding next state y_i. In the illustrative example the computation of a datapoint corresponds to one gradient descent step on Himmelblau's function.

Besides the dataset of snapshots $\mathcal{D}_{snap}$, we also require a dictionary of observables $\mathcal{D}_{obs}$ which serves as the basis to approximate the space of observables $\mathcal{F}$. While the choice of the dictionary $\mathcal{D}_{obs}$ is important, we simply follow Dietrich et al. [8] in order to keep the illustrative example comparable. Thus, the dictionary of observables $\mathcal{D}_{obs}$ contains three different groups of observables. The first one only contains the one function

$$\mathbf{1} : \mathcal{W} \to \mathbb{R}, \tag{68}$$

$$W \mapsto 1. \tag{69}$$

The second one is the set of axis functions π_j

$$\pi_j : \mathcal{W} \to \mathbb{R}, \tag{70}$$
$$W \mapsto W_j, \tag{71}$$

where $1 \leq j \leq dim(\mathcal{W})$ and $dim(\mathcal{W})$ is the dimension of the finite-dimensional state space $\mathcal{W}$. The third one is a set of 500 thin-plate radial basis functions [10]. For these, we set the parameter $\delta = 10^{-3}$ and select 500 function centers μ_i uniformly at random from the set of interest $\mathcal{W}_c$. Then, the thin-plate radial basis functions are given by

$$d_i : \mathcal{W} \to \mathbb{R}, \tag{72}$$
$$d_i(W) = \|W - \mu_i\|^2 \ln(\|W - \mu_i\| + \delta). \tag{73}$$

So, in total the dictionary of observables $\mathcal{D}_{obs}$ contains $N_{obs} = 501 + dim(\mathcal{W})$ observables to approximate the space of observables $\mathcal{F}$. For our illustrative example, the number of observables in the dictionary of observables $\mathcal{D}_{obs}$ is $N_{obs} = 503$ since Himmelblau's function is a function mapping from $\mathbb{R}^2$ into $\mathbb{R}$ and, therefore the state space of gradient descent is also two-dimensional.
Now that we have explained the generation of the dataset of snapshots $\mathcal{D}_{snap}$ and the dictionary of observables $\mathcal{D}_{obs}$, we can use EDMD as described in section 2.2.4 to find a finite-dimensional approximation $\mathbf{K} \in \mathbb{C}^{N_{obs} \times N_{obs}}$ of the Koopman operator $\mathcal{K}$ of the dynamical system under analysis. This approximation forms the basis of our analysis. Since eigenvalues and eigenfunctions play an important role, we want to repeat that we use λ_i to denote the i-th eigenvalue of the Koopman operator and φ_i for the corresponding eigenfunction.

3.2.2 Basins of Attraction

One of the main tools in the analysis of algorithms through the Koopman operator is the determination of basins of attraction. Determining and visualising the basins of attraction provides a rough overview on which regions the dynamical system is expected to converge to some value. This section provides an abbreviated explanation of the idea behind this analysis technique and explains the approach used in this thesis. For a large part, the approach is based on the work of Dietrich et al. [8]. For a more extensive explanation and the mathematical foundation of this approach, we refer to Dietrich et al. [8], Budišić&Mezić [4], and Budišić et al. [5].
First, we start with the basic idea. To this end, we remind the reader that, as explained in section 3.2.1, we acquired a finite-dimensional approximation $\mathbf{K}$ of the Koopman operator $\mathcal{K}$ through the application of EDMD. Further, this resulted in N_{obs} approximations λ_i of the eigenvalues of the Koopman operator $\mathcal{K}$ and corresponding approximative eigenfunctions φ_i. As mentioned in section 2.2.2, the pure-point spectrum $\sigma_p(\mathcal{K})$ of a continuous linear operator like the Koopman operator can be considered a generalisation of eigenvalues and eigenvectors of matrices in linear algebra. This can also be seen in the definition

$$\sigma_p(\mathcal{K}) = \{\lambda \in \mathbb{C} | \lambda I - \mathcal{K} \text{ is not injective}\}. \tag{74}$$

So, for a pair of eigenvalue λ_i and eigenfunction φ_i of the Koopman operator $\mathcal{K}$ it holds

$$\mathcal{K}\varphi_i = \lambda_i \varphi_i. \tag{75}$$

Further, we remember that the eigenfunctions φ_i can be used to reconstruct the full-state observable

$$g : \mathcal{W} \to \mathcal{W}, \tag{76}$$

$$g(W) = W, \tag{77}$$

and, thus, allow to mimic the behaviour of the dynamical system on the state space. As a result of these two facts, we can split eigenvalues into different sets, which allow for analysis of long term behaviour of the associated eigenfunctions and, therefore, to some extend the dynamical system. One set, one can consider is the set of eigenvalues within the open unit circle $B_1(0)$. By repeatedly applying Eq. 75, we can see that the influence of associated eigenfunctions on the state of the dynamical system vanishes. Because of this observation we will not further consider eigenvalues in this set.
Another interesting set to consider is the set of absolutely large eigenvalues, meaning eigenvalues λ_i for which holds $|\lambda_i| \gg 1$. Here, when applying Eq. 75 n times, we have that the effect of the eigenfunctions φ_i on the state is increased by a factor $\lambda_i^n \gg 1$. Therefore, their associated eigenfunctions describe regions in which it can be assumed that the dynamical system accelerates over time.
In this analysis we mainly focus on eigenvalues $\lambda_i \approx 1$. By considering Eq. 75, we see that eigenfunctions associated with these eigenvalues are preserved under the flow of the dynamical system. This allows to construct an ergodic decomposition [4, 5] of the state space which separates basins of attraction. To this end, we first select the eigenvalues $\lambda_i \approx 1$ and their corresponding eigenfunctions φ_i. In the illustrative example, we can assume that we will have at least four such eigenvalues, since Himmelblau's function is known to have four local minima which together with some environment form the basins of attraction. In general, the number of basins of attraction or the number of eigenvalues $\lambda_i \approx 1$ is not known. So, we chose an approximation tolerance $\epsilon > 0$ so that if $|\lambda_i - 1| < \epsilon$ we consider the eigenvalue λ_i to be one. Since the choice of ϵ also depends on the dynamical system, we further limit ourselves to at most ten clusters. Our selection process for eigenvalues leaves us with $N_1 \in \mathbb{N}$ eigenvalues $\lambda_i \approx 1$ and corresponding eigenfunctions $\varphi_i : \mathcal{W} \to \mathbb{C}$. The exact value of N_1 depends on the dynamical system under analysis. Without loss of generality, we assume that these eigenvalues λ_i and eigenfunctions φ_i have indices $i \in \{1, ..., N_1\}$, since otherwise these indices can be enforced by relabeling them with a suitable permutation. In order to approximate the ergodic decomposition, we use a helper function

$$\Psi : \mathcal{W} \to \mathbb{C}^{N_1}, \tag{78}$$

$$W \mapsto [\varphi_1(W), ..., \varphi_{N_1}(W)], \tag{79}$$

which evaluates the eigenfunctions associated with eigenvalues $\lambda_i \approx 1$ at a given point W in the state space $\mathcal{W}$ and returns a N_1-dimensional vector thereof. Next, we evaluate the helper function Ψ on the first element of each datapoint in the dataset of snapshots $\mathcal{D}_{snap}$. To be more specific, for each $(x_i, y_i) \in \mathcal{D}_{snap}$ we evaluate $z_i := \Psi(x_i) \in \mathbb{C}^{N_1}$ and store it in a dataset $\mathcal{D}_{basins}$. To recall the first elements x_i of each datapoint (x_i, y_i) in the dataset of snapshots $\mathcal{D}_{snap}$ are chosen so they create a mesh on the set of interest $\mathcal{W}_c$ on which we want to analyse the dynamical system. Therefore, the dataset $\mathcal{D}_{basins}$ contains evaluations of relevant eigenfunctions on the entire compact set of interest $\mathcal{W}_c$.

The next step to finding the basins of attraction for our dynamical system is to cluster the dataset $\mathcal{D}_{basins}$ in the space $\mathbb{C}^{N_1}$. Then, each cluster should correspond to a basin of attraction. For the illustrative example with Himmelblau's function, we know that we have four basins of attraction and, thus, require four clusters. Unfortunately, the number of basins of attraction of a dynamical system is not known, in general. As a result, the number of clusters we have to find is also not known, in general. While there are several approaches to solve this problem[22] we instead use an upper bound of the number of clusters we need. Then, we just cluster according to this upper bound. We note, that this approach is not perfect and might very well result in datapoints which should belong to a single cluster to be distributed among several clusters, but it saves us the development of a more elaborate solution to this nontrivial problem. In order to find an upper bound of the number of clusters required, we exploit our knowledge on how the number of clusters, the number of basins of attraction, and the number of eigenvalues $\lambda_i \approx 1$ relate to each other. We know that N_1, the number of eigenvalues $\lambda_i \approx 1$, is an upper bound of the number of basins of attraction. Therefore, N_1 is also an upper bound of the number of clusters since ideally each cluster corresponds to one basin of attraction,. Even when the number of cluster is known (or estimated as in our case) there are still several algorithms to chose from. In this thesis we use k-means, mainly because it is the algorithm use by Dietrich et al. [8], but its speed is also a significant benefit.
After clustering the datapoints in $\mathcal{D}_{basin}$ we plot them in the set of interest $\mathcal{W}_c$, where the color represents the cluster. It should be noted, that the label associated with a cluster is arbitrary and we are only interested in which datapoints belong to the same cluster. Thus, a legend for the colors is not required.

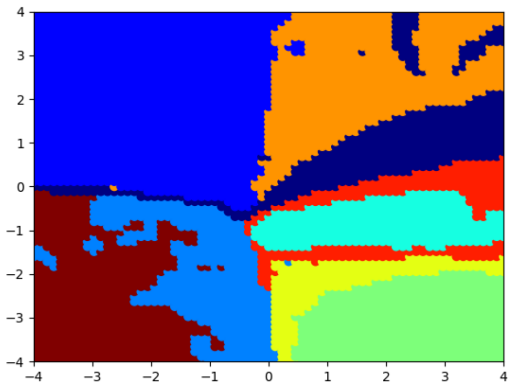

Figure 1: Plot of clustering results on gradient descent on Himmelblau's function. Each color represents one cluster. In practice, one basin of attraction can correspond to multiple clusters. Here, we can see nine different clusters, but still make out the contour of the actual basins of attraction.

An example of a plot visualising the results of this analysis for gradient descent on Himmelblau's function is provided in Fig. 1. As can be seen the area is split into nine different clusters, even though four would be sufficient. If the regions of the basins of attraction of the gradient descent algorithm on Himmelblau's function are known, it is still possible to see which clusters should be considered together in order to form a basin of attraction. In order to investigate, why we see nine clusters instead of four, as we expected, we first reduce the number of clusters to four, meaning N_1, the number of eigenvalues $\lambda_i \approx 1$, is set to four, instead of estimating it with a tolerance value $\epsilon > 0$. The results of this can be seen in Fig. 2.

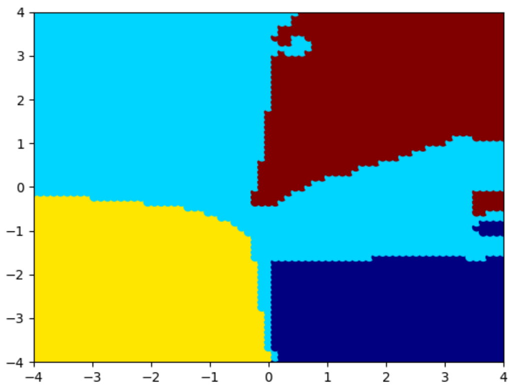

Figure 2: Plot of clustering results on gradient descent on Himmelblau's function. Each color represents one cluster. In practice, one basin of attraction can correspond to multiple clusters. Here, we have restricted the number of clusters to four artificially. The clusters roughly correspond to the basins of attraction.

While the results do not depict the basins of attraction completely correct, each cluster still roughly corresponds to a basin of attraction. Thus, the discrepancy of our results to the results of Dietrich et al. [8] are mainly due to the varying number of clusters, which again is the result of us not assuming to know the correct number of basins of attraction and, thus, of clusters when running the analysis. The remaining difference is likely due to differing centres for the thin-plate radial basis functions used in EDMD and a varying number of datapoints used to approximate the Koopman operator.

0.00049	0.00049	0.00264	0.00352	0.00352	0.00632	0.00776	0.00808	0.00808

Table 1: This table provides the distance of the nine closest eigenvalues to one in the example with Himmelblau's function. As can be seen, the table contains duplicate values, which might indicate duplicate eigenvalues. Further, no significant jump in the distance of the eigenvalues can be noticed.

In order to see, if there would be a better natural value for the approximation tolerance ϵ, we have a look at the distance of the nine eigenvalues λ_i closest to one. The distances are shown in Table 1. As can be seen, there are several duplicate values, meaning there are multiple eigenvalues which have equal distant to one. Equal distance might indicate identical eigenvalues, which might lead to the idea of removing duplicates in order to reduce the number of clusters and get a better depiction of the basins of attraction. we want to note that in the ideal case the approximation of the eigenvalues provided by

EDMD converges to the true eigenvalues, which would be one in our scenario. Therefore, this would result in a reduction of the estimated number of clusters and, thus, basins of attraction to one which would be incorrect. Because of this consideration we do not remove duplicate eigenvalues from the list. Next, we can also consider if we can select a better value for ϵ by hand. As can be seen in Table 1, there are no noticeable jumps in distance anywhere. As a result a simple consideration of the distances of eigenvalues λ_i to one is not sufficient.

3.2.3 Implementation

In this section we briefly note the software and tools used for our implementation of the analysis. The code is written in Python 3.9.7. Basic numerical operations and data processing is done using the numpy library [12]. This noticeably also includes the random sampling for the generation of synthetic data and the calculation of the corresponding target values. Existing datasets are imported used functionality provided by scikit-learn [26]. Further processing of data requires pandas [25] as an intermediate representation, which is then used by datafold [17] in order to generate time series datasets. These time series datasets are used by datafold implementation of EDMD. Here, we want to note that the thin-plate kernels we use in this thesis do currently not exist in datafold. In order to solve this, we developed our own implementation compatible with datafold. In this thesis, we use k-means for clustering. The implementation thereof is provided by scikit-learn. For plotting we use matplotlib [15].

3.3 Kernel Flows on Synthetic Data

In this section, we generate synthetic datasets in order to analyse Kernel Flows in a controlled environment. First, in section 3.3.1 we analyse the behaviour of Kernel Flows in a scenario with two parameters to optimize in order to be able to set the results of Darcy [7] in context. Then, in section 3.3.2 we investigate the influence of the optimizer on our analysis. Under investigation are stochastic gradient descent (SGD) and Nesterov Momentum.

Before we explain the details of the analysis, we explain the data generation process. Some variables depend on the experiment and, therefore, the specific values will be given in the corresponding section. So without further ado, let $\mathcal{X} := \mathbb{R}, \mathcal{Y} := \mathbb{R}$ be normed spaces with the euclidean norm. In order to be able to train Kernel Flows, for each experiment we generate a synthetic dataset

$$\mathcal{D}_{kernel} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} | 1 \leq i \leq N_{kernel}\}, \tag{80}$$

where $N_{kernel} = 100$. For the generation of each datapoint $(x_i, y_i), i \in \{1, ..., N_{kernel}\}$, we first sample $x_i \sim Uni(-10, 10)$. Here, we use $Uni(-10, 10)$ to denote the uniform distribution on the interval $(-10, 10)$. The target value y_i will be calculated according to the sum of two Gaussian kernels. So, let $\sigma_1, \sigma_2 > 0$ be two standard deviations for Gaussian kernels, then based on the sample value $x_i \in \mathcal{X}$, we calculate the target value $y_i \in \mathcal{Y}$ according to

$$y_i := \exp(-\frac{\|x_i\|^2}{2\sigma_1^2}) + \exp(-\frac{\|x_i\|^2}{2\sigma_2^2}). \tag{81}$$

The exact values for σ_1 and σ_2 depend on the experiment and will be provided in the respective section. So to summarize, the data generation process consists of first sampling random values x_i from a pre-defined interval and then calculating the corresponding value of a sum of two Gaussian kernels as target value y_i.
Another hyper-parameter we have to specify for our analysis is the kernel we use for Kernel Flows. Here, we select a two-linear Gaussian kernel, where we set $\beta_1 = \beta_2 = 1$. The parameters $\sigma_1, \sigma_2 > 0$ are the parameters Kernel Flows will optimise. Therefore, out kernel $k : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$ is given by

$$k(x, x') = \exp(-\frac{\|x - x'\|^2}{2\sigma_1^2}) + \exp(-\frac{\|x - x'\|^2}{2\sigma_2^2}), \tag{82}$$

where $x, x' \in \mathcal{X}$.
For the selection of this kernel we have two main reasons. First, in section 3.3.1 it allows us to be consistent with Darcy [7] for the convergence analysis in the multi-parameter scenario, which enables comparison of results. Second, the depiction of the parameter space is easiest when it is two dimensional. For a higher number of dimensions, it would be necessary to use some approach for dimensionality reduction like PCA. This in turn would further complicate the interpretation of the results.

3.3.1 Multiple Parameters

In this section, we first analyse the behaviour of Kernel Flows for multiple parameters. The main reason for this analysis are the results of Darcy [7]. He showed that Kernel Flows reliably converges to the true value when we only have to optimise one parameter, but when two parameters are optimised simultaneously Kernel Flows does not necessarily converge to the true parameters. It should be noted, that Kernel Flows still converges and yields good results, just not with the true parameters. In order to investigate this behaviour further, we analyse the scenario using the Koopman operator and attempt to detect basins of attraction.
For this experiment the parameter space is $\mathcal{W} := \mathbb{R}^+ \times \mathbb{R}^+$, the squared set of positive real numbers. In order to be consistent with Darcy [7], we select the values $\sigma_1 = 0.9, \sigma_2 = 1.5$ for the generation of the dataset $\mathcal{D}_{kernel}$ for training Kernel Flows. Accordingly, we define the set of interest $\mathcal{W}_c$ to be the cube $[0.1, 3] \times [0.1, 3]$ and use 71 sample points per dimension which leaves us with a total $N_s = 5041$ datapoints in $\mathcal{D}_{snap}$.
The results of our search for basins of attraction are shown in Fig 3. As can be seen, the with our approach we find ten different clusters, but due to the arrangement of the set of clusters in the top-right corner and the fact that some clusters like the light orange one are disconnected, it can be assumed that the number of clusters our method yields is larger than the number of basins of attraction. Therefore, some clusters probably have to be combined. Which ones these are cannot be determined reliably by inspection, thus the remainder of this analysis is somewhat speculative.
First, we note that the line $\sigma_1 = \sigma_2$ can be clearly noticed in the plot. Further, the plot can be somewhat mirrored around this line. These facts can easily be explained by the fact that our kernel is the sum of two Gaussian kernels with equal weights $\beta_1 = \beta_2 = 1$ and addition is commutative. We notice that the both solutions for the true values for the parameters σ_1, σ_2 are contained in the same cluster. In Darcy's work, the parameters

σ_1, σ_2 converged to 1.00 and 1.25 respectively. This point is also in the same cluster as the true values $0.9, 1.5$. Thus, assuming the clusters are sufficiently correct, this indicates that the converged point and the true value of the parameters are in the same basin of attraction. This would implicate that either convergence to the true values is given, but slows significantly over time, or there truly is non-convergence which is then likely introduced by the random nature of the algorithm or an insufficient approximation due to the dataset being to small.

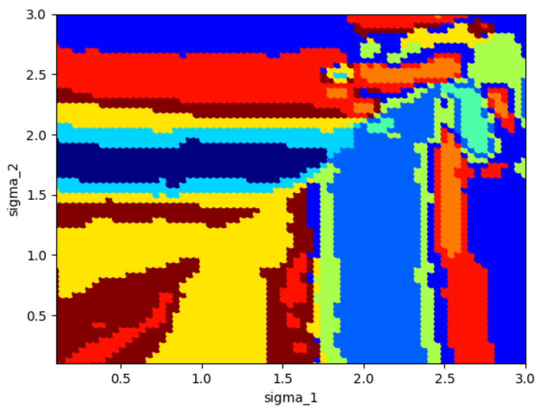

Figure 3: Plot of basins of attraction for analysis of convergence in a two-parameter space. Each color represents one cluster. In practice one basin of attraction can correspond to multiple clusters. The axes represent the different dimensions in parameter space.

3.3.2 Comparison of Optimizers

In this section we investigate the effect of the optimizer used in Kernel Flows. The optimizers under investigation are stochastic gradient descent (SGD) and Nesterov Momentum. The reason for this analysis is that Nesterov Momentum is capable of using information of past gradients in order to avoid getting trapped in local minima. Thus, we want to analyse if this difference in behaviour can be detected by locating the basins of attraction using the Koopman operator.
Like in the previous section, the parameter space is $\mathcal{W} := \mathbb{R}^+ \times \mathbb{R}^+$. For the generation of the dataset $\mathcal{D}_{kernel}$ we select the true values $2.0, 3.0$ for σ_1, σ_2 respectively. The set of interest is the cube $[0.1, 6.0] \times [0.1, 6.0]$ and we use 71 sample points per dimension. Therefore, the total number of datapoints N_s in the dataset $\mathcal{D}_{snap}$ is 5041.
The results of the analysis with stochastic gradient descent are depicted in Fig. 4. Fig. 5 shows the clustering results for Nesterov Momentum. Once again, we note that the number of clusters can not reliably be selected to be the correct number of basins of attraction. Therefore, we have to consider that some clusters should be united in order to get the correct basins of attraction. The fact that in both images some clusters are disconnected suggests, that the selection of the number of clusters is not optimal.
By comparing Fig. 4 and Fig.5, we notice that the rough layout of the clusters is comparable. In both figures, we can see a set of clusters which roughly covers the area of the

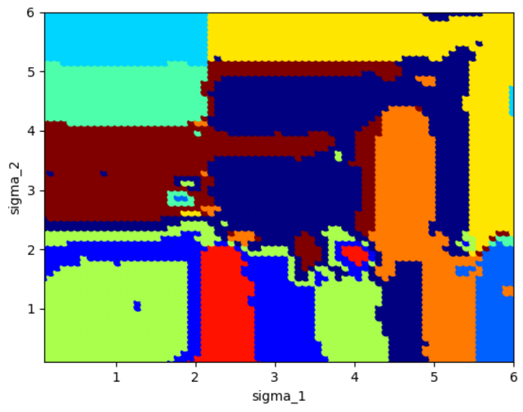

Figure 4: Plot of basins of attraction for the comparison of optimizers. This plot depicts the results for SGD. Each color represents one cluster. In practice one basin of attraction can correspond to multiple clusters. The axes represent the different dimensions in parameter space.

cube $[2,6] \times [2,6]$. Further, we can detect a set of clusters covering the area of the cube $[0,2] \times [0,2]$ in both images. The main difference is that for Nesterov Momentum the bottom left corner is split in noticeably less clusters than for SGD, but this might just be a result of too many clusters in the analysis.
To summarize, our analysis did not find noticeable difference in the basins of attraction in the parameter space. This indicates that both algorithm converge in roughly the same regions to similar values, but it should also be noticed that an overestimation of the number of clusters might have significant effect on the results of the analysis.

3.4 Kernel Flows on Real Data

While the analysis of Kernel Flows on synthetic data is useful to make observations, form hypothesis, and test or prove them, it is also important to analyse the behaviour of Kernel Flows on real datasets in order to check whether or not the process for generating synthetic data introduces bias, which might mask challenges faced in real data. To this end, we analyse Kernel Flows on three real datasets. Section 3.4.1 works with the Boston Housing dataset, while we take a better look on the behaviour of Kernel Flows when trained on the Diabetes dataset in section 3.4.2.
Like in section 3.3 we use a two-linear Gaussian kernel for Kernel flows in the experiments in this section. As previously, the weights of the kernel are equal and set to $\beta_1 = \beta_2 = 1$. The remaining parameters which build our parameter space $\mathcal{W}$ are the two standard deviations σ_1, σ_2. Therefore, we have $\mathcal{W} = \mathbb{R}^+ \times \mathbb{R}^+$. The input space $\mathcal{X}$ and the target space $\mathcal{Y}$ depend on the used dataset and cannot be further specified here.
For the set of interest $\mathcal{W}_c$ we select $\mathcal{W}_c = [1, 1000] \times [1, 1000]$ in order to cover the selection of initial values by Darcy [7]. From the set of interest $\mathcal{W}_c$ we selected equidistant points so we have 71 different values per dimensions. Thus, the total number of datapoints N_s in the dataset $\mathcal{D}_{snap}$ for approximating the Koopman operator $\mathcal{K}$ is 5041.

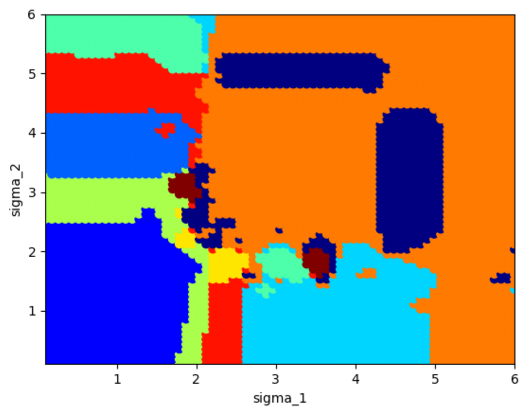

Figure 5: Plot of basins of attraction for analysis for comparison of optimizers. This plot depicts the results for Nesterov Momentum. Each color represents one cluster. In practice one basin of attraction can correspond to multiple clusters. The axes represent the different dimensions in parameter space.

3.4.1 Boston Housing Dataset

The Boston Housing Dataset [13] is a frequently used example dataset in machine learning. It contains data related to the Boston housing market and contains information of the U.S. Census Service. Further, it consists of 506 datapoints. The target when using this dataset is usually to train a model for predicting the housing price given a set of thirteen features. The feature list includes, but is not limited to:

- the crime rate per capita by town,
- concentration of nitric oxides in parts per million,
- average number of rooms per dwelling,
- the pupil-teacher ratio by town.

We provide a brief summary of important statistics of the dataset in Table 2. As can be seen, the input space for Kernel Flows is the metric space $\mathcal{X} := \mathbb{R}^{13}$, the output space is $\mathcal{Y} := \mathbb{R}$. Since we consider the Boston Housing dataset as the training dataset for Kernel Flows, the dataset corresponds to $\mathcal{D}_{kernel}$ and the number of datapoints in the dataset is $N_{kernel} := 506$.
The results of the analysis are depicted in Fig. 6. For this analysis we want to note that there are still some disconnected clusters. This might indicate that the number of clusters is not chosen correctly, but unlike in the experiments with synthetic data, most clusters seem more natural. Therefore, it is also reasonable to consider, that the restriction of the number of clusters to ten might be to strict, since there might be a larger number of basins of attraction in the considered set of interest $\mathcal{W}_c$.

Samples Total	Number Features	Type Features	Target Range
506	13	real, positive	5. - 50.

Table 2: Summary of important statistics of the Boston Housing dataset. The column Samples Total contains the total number of samples in the dataset, Number Features is the dimension of the input space for a model. The column Type Features describes what data types can be expected from the 13-dimensional feature space. In this case, we have positive real-valued features. The Target Range gives the set of possible values in the target variable. In this case the values range from 5. to 50., where the target value is given in 1000$.

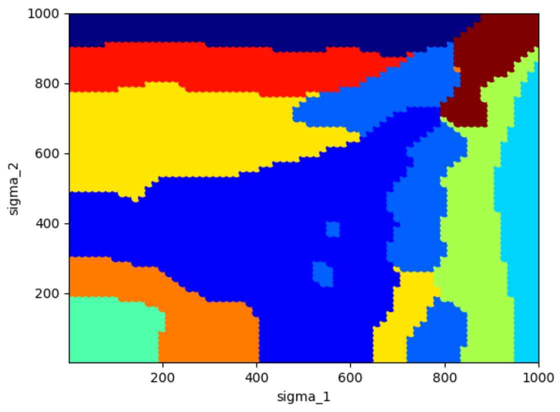

Figure 6: Plot of basins of attraction for the Boston Housing dataset. Each color represents one cluster. In practice one basin of attraction can correspond to multiple clusters. The axes depict the two parameters under optimisation, σ_1 and σ_2.

3.4.2 Diabetes Dataset

The diabetes dataset is another commonly used toy dataset in machine learning. The goal is to predict the progression of diabetes one year after taking the initial measurements based on a set of features which might be of medical relevance for the prediction. The features include information like the age of a subject in years, a subject's sex, the body mass index, and additional medical measurements like

- average blood pressure,
- total serum cholesterol,
- low- and high-density lipoproteins,
- total cholesterol,
- triglycerides levels,
- and blood sugar levels.

A short summary of important characteristics of the Diabetes dataset is provided in Table 3. The input space for Kernel Flows is modelled as metric space $\mathcal{X} := \mathbb{R}^{10}$, the target space as $\mathcal{Y} := \mathbb{R}$. The Diabetes dataset fills the role of the training dataset $\mathcal{D}_{kernel}$ for Kernel Flows, where the number of datapoints is $N_{kernel} := 442$.

Samples Total	Number Features	Type Features	Target Range
442	10	real, (-.2, .2), categorical	integer, 25-346.

Table 3: Summary of important statistics of the Diabetes dataset. The column Samples Total contains the total number of samples in the dataset, Number Features is the dimension of the input space for a model. The column Type Features describes what data types can be expected from the 10-dimensional feature space. In this case, we have real-valued features and features in range $(-.2, .2)$. The Target Range gives the set of possible values in the target variable. In this case the values range from 25-346 and only take on integer values.

The results of the analysis are shown in Fig. 7. The first thing to notice is that most clusters seem to touch in a single point. Based on the fact that this also happens in the analysis of Himmelblau's function at the local maximum of the function, this might indicate a local maximum of the loss function ρ in the parameter space. Further, we notice that similar to the analysis in section 3.4.1, the clusters seem to have a more natural structure than the one in section 3.3.

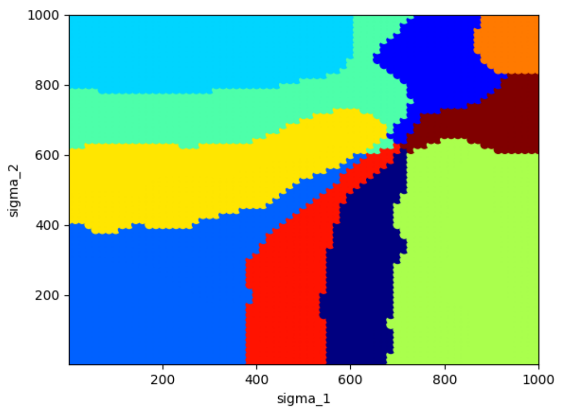

Figure 7: Plot of basins of attraction for the Diabetes dataset. Each color represents one cluster. In practice one basin of attraction can correspond to multiple clusters. The axes represent the dimensions in parameter space.

4 Conclusion

In this thesis we used the Koopman operator to analyse Kernel Flows, more specifically the parametric version of Kernel Flows. To this end, we first highlighted in chapter 1 the importance of kernel methods in general and, as a result of the requirement to determine a good kernel for a given problem, the importance of Kernel Flows, a data-driven algorithm to finding good kernels, in particular. This motivates and justifies the analysis of Kernel Flows. While there already exists some analysis of the algorithm in the original work of Owhadi&Yoo [24] and by Darcy [7], a new approach can discover new information and result in additional insights which is the reason why we use the Koopman algorithm for analysis.
After providing the motivation for the thesis, we cover the basics in chapter 2. This mainly includes an introduction into the concept of kernels and an explanation of parametric Kernel Flows in section 2.1, as well as an introduction of the Koopman operator in section 2.2. While there exists a vast amount of literature on the topic of Koopman operator theory and related fields, we limit ourselves to a brief introduction to dynamical systems followed by a short summary of spectral theory in functional analysis and the introduction of the Koopman operator. We also briefly explain the dynamical system perspective on algorithms introduced by Dietrich et al. [8] and EDMD for approximating the Koopman algorithm in a fashion suitable for practical application.
Our main contributions are presented in chapter 3. We start out by properly formalizing Kernel Flows as a dynamical system in section 3.1. The formalization of Kernel Flows as dynamical system allows us to approximate the Koopman operator of the system as detailed in section 3.2. The approximated Koopman operator then serves as the basis to our approach to finding the basins of attraction in the parameter space. Here, we find based on the exemplary analysis of gradient descent on Himmelblau's function, that the estimation of the number of basins of attraction has a noticeable influence on the results of the analysis. Next, we apply our method to Kernel Flows. In section 3.3 we first consider Kernel Flows on synthetic data, so we can control for variables and know the optimal parameters. To finish our analysis, we consider Kernel Flows on real datasets in section 3.4.
We end this thesis by discussing our results in section 4.1 and providing an overview on future work in section 4.2.

4.1 Discussion

In this thesis, the major observation we contribute regarding the analysis of algorithm through the Koopman operator is that the selection of the number of clusters plays a significant role for the results and the visibility of basins of attraction. Since the correct number is not known in general, some method for estimating the correct number is required. In this thesis we tried the usage of a simple cut-off value $\epsilon > 0$ for estimating if an eigenvalue is close enough to one. Unfortunately, this simple approach is not successful since the optimal value for ϵ varies significantly from case to case. Therefore, another more elaborate approach is required.
Besides the problem of estimating the number of clusters, we were able to verify the results of Dietrich et al. [8], further showing their correctness and the utility of the analysis of

algorithms through the Koopman operator.
Regarding Kernel Flows the results of the analysis are somewhat limited due to the aforementioned problems with estimating the number of clusters, but the results are sufficiently good to indicate that in the multi-parameter scenario the parameters should converge to the true values. The exact reason why convergence does not happen could not found.
When comparing SGD and Nesterov Momentum, we could not find any noticeable differences in the results of the analysis. This indicates that both optimizers converge in similar regions of the parameter space to similar values, but it should be noted that according to Darcy [7] Nesterov Momentum converges noticeably faster than SGD.

4.2 Future Work

Based on our results, the most important next step would be the development of a method to reliably cluster the evaluations of the eigenfunctions with the correct amount of clusters. To this end, several approaches could be taken. One would be to switch from k-means to a clustering algorithm which does not require knowledge about the number of clusters a priori. Another option would be an iterative approach. This means first clustering with k-means, then some evaluation on the quality of the clustering, e.g. with some version of linkage, and the repeating the process until (hopefully) convergence. We want to not, that before attempting the second approach the vast amount of literature on the topic of clustering should be consulted since there might very well already exist an algorithm which essentially performs this task, but probably more efficient.
In the analysis of Kernel Flows, we noticed that the clusters on real data seem more natural than on the synthetic dataset. A good next step would be to analyse the cause of this behaviour, since it might be due to some characteristics of the synthetic dataset which render the entire analysis on these useless. Another straight forward option would be the analysis of more elements of Kernel Flows. In this thesis we only analysed Kernel Flows with a two-linear Gaussian Kernel, other kernel options have not been considered. Other aspects which have not been investigated are the influence of the batch size or regularization on Kernel Flows.
Further, in this thesis we only considered parameterized Kernel Flows. An extension of our approach to the non-parameterized version of Kernel Flows would also be interesting.

References

[1] Romeo Alexander and Dimitrios Giannakis. Operator-theoretic framework for forecasting nonlinear time series with kernel analog techniques. *Physica D: Nonlinear Phenomena*, 409:132520, Aug 2020.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Michael Brin and Garrett Stuck. *Introduction to Dynamical Systems.* Cambridge University Press, 2002.

[4] Marko Budišić and Igor Mezić. An approximate parametrization of the ergodic partition using time averaged observables. pages 3162 – 3168, 01 2010.

[5] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22:047510, 2012.

[6] Nelida Črnjarić-Žic, Senka Maćešić, and Igor Mezić. Koopman operator spectrum for random dynamical systems. *Journal of Nonlinear Science*, September 2019.

[7] Matthieu Darcy. Kernel flows demystified: Applications to regression, 2020.

[8] Felix Dietrich, Thomas N. Thiem, and Ioannis G. Kevrekidis. On the koopman operator of algorithms. *SIAM Journal on Applied Dynamical Systems*, 19(2):860–885, January 2020.

[9] Akshunna S. Dogra and William T. Redman. Optimizing neural networks via koopman operator theory. June 2020.

[10] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In Walter Schempp and Karl Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100, Berlin, Heidelberg, 1977. Springer Berlin Heidelberg.

[11] Efrain Gonzalez, Moad Abudia, Michael Jury, Rushikesh Kamalapurkar, and Joel A. Rosenfeld. Anti-koopmanism. May 2021.

[12] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[13] D. Harrison and D.L. Rubinfeld. Hedonic Housing Prices and the Demand for Clean Air. *Journal of Environmental Economics and Management*, 5:81–102, 1978.

[14] David Mautner Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill, 1972.

[15] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[16] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, 1931.

[17] Daniel Lehmberg, Felix Dietrich, Gerta Köster, and Hans-Joachim Bungartz. datafold: data-driven models for point clouds and time series on manifolds. *Journal of Open Source Software*, 5(51):2283, 2020.

[18] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, Aug 2005.

[19] Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45(1):357–378, jan 2013.

[20] Igor Mezic. Koopman operator spectrum and data analysis. *arXiv*, (1702.07597), February 2017.

[21] Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(95):2651–2667, 2006.

[22] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[23] Feliks Nüske, Sebastian Peitz, Friedrich Philipp, Manuel Schaller, and Karl Worthmann. Finite-data error bounds for koopman-based prediction and control. August 2021.

[24] Houman Owhadi and Gene Ryan Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019.

[25] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[27] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656(August):5–28, 2010.

[28] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Learning Theory*, 2001.

[29] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, June 2015.