TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ANDREI AHONEN

LEARNING AUTONOMOUS MOTION GENERATING DYNAMICAL SYSTEMS FROM DEMONSTRATION

Master's thesis

Examiner: Associate Professor Reza Ghabcheloo

The examiner and topic of the thesis were approved on 30 May 2018

ABSTRACT

ANDREI AHONEN: Learning autonomous motion generating dynamical systems from demonstration
Tampere University of Technology
Master of Science Thesis, 40 pages
November 2018
Master's Degree Programme in Automation Engineering
Major: Robotics
Examiner: Associate Professor Reza Ghabcheloo

Keywords: Dynamical systems, Learning from Demonstration, Behavioral Cloning, Robotics, Machine Learning

This thesis studies dynamical systems based learning methods at a proof-of-concept level. The purpose of dynamical systems is to generate motion. In particular, three different methods are studied in detail and implemented in software to judge their applicability for a real robotic system. These methods were chosen for the stability they guarantee for the dynamical system. The software was used with a real manipulator arm to reproduce taught motions and the reproduction data was recorded and studied. The results indicate that the methods are viable for learning robotic motions but caution should be exercised when using the dynamical system for motion generation.

TIIVISTELMÄ

ANDREI AHONEN: Liikkeen tuottavan dynaamisen järjestelmän oppiminen esimerkistä
Tampereen teknillinen yliopisto
Diplomityö, 40 sivua
Marraskuu 2018
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Robotiikka
Tarkastaja: apulaisprofessori Reza Ghabcheloo

Avainsanat: Dynaamiset järjestelmät, Koneoppiminen, Robotiikka

Tässä diplomityössä tarkastellaan dynaamisiin järjestelmiin perustuvan oppimisen toteutuskelpoisuutta. Dynaamisten järjestelmien tarkoitus on tuottaa liikettä. Työssä tutkitaan erityisesti kolmea erilaista lähestymistapaa jotka toteutetaan ohjelmistona sovelluskelpoisuuden arviointia varten. Lähestymistavat on valittu niiden tuottamien dynaamisten järjestelmien stabiilisuustakeiden vuoksi. Robottikättä ohjaavalla ohjelmistolla tuotettiin opetettuja liikkeitä ja tuotettu liike tallennettiin ja käytiin läpi. Tulokset antavat syytä olettaa, että tutkitut menetelmät ovat varteenotettavia tapoja oppia robotin liikkeitä, mutta dynaamisen järjestelmän soveltaminen liikkeentuotossa tulee tehdä huolella.

PREFACE

This thesis is partly funded by, and is part of, the MIDAS-project.

In Tampere, Finland, on 30 November 2018

Andrei Ahonen

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

DS	dynamical system
SEA	swept error area
SEDS	Stable estimator of dynamical systems
LMDS	Local modulation of dynamical systems
MDP	Markov Decision Process
LASA	Learning algorithms and systems laboratory
DMP	Dynamical movement primitive
DOF	Degrees of freedom
GmbH	Gesellschaft mit beschränkter Haftung
FCI	Franka Control Interface
UDP	User Datagram Protocol
ROS	Robot Operating System
DH	Denavit-Hartenberg

t	time, indexing variable
d	number of dimensions
x	a real number
$\dot{x}$	time derivative of x
$f(x)$	a function of x
$f'(x)$	Derivative of f along x
$f(x, y)$	a function of x and y
$\mathbb{R}$	Space of real numbers
$\mathbb{R}^d$	Real coordinate space of d dimensions
$\mathbf{x}$	Vector- or matrix-valued variable
$\in$	" is an element of "
$\iff$	" if and only if "
$\mapsto$	" maps to "
$\frac{d}{dx}$	Derivative along x
$\cdot$	Dot product of vectors, scalar multiplication of a vector,product of real variables(Context dependent)
$\|x\|$	Euclidean norm of x
N	Number of demonstrations in a dataset
T	Number of points in a trajectory
τ	time, a generic symbol
i	indexing variable
$\leftarrow$	Assignment
ϵ	A small, but positive real number
∇f	Gradient of f
μ	Modulation function, expected value of probability distribution
$\mathbb{R}^+$	Non-negative real numbers
$\hat{x}$	Estimate of x
$P(x\|y)$	probability(distribution) of x given y
$\mathbf{A}^T$	Transpose of $\mathbf{A}$
$\boldsymbol{\Sigma}$	Covariance matrix, sum
$\boldsymbol{\Theta}$	Set of parameters of a probability distribution, a vector
$\|\mathbf{A}\|$	Determinant of $\mathbf{A}$
#	" number of "
$\begin{bmatrix}\mathbf{x}\\\mathbf{y}\end{bmatrix}, \begin{pmatrix}\mathbf{x}\\\mathbf{y}\end{pmatrix}$	Vector concatenation of $\mathbf{x}$ and $\mathbf{y}$
$\angle(\mathbf{x}, \mathbf{y})$	angle between $\mathbf{x}$ and $\mathbf{y}$
$\boldsymbol{\Phi}$	A function
$\boldsymbol{\Phi}^{-1}$	Inverse function of $\boldsymbol{\Phi}$
ψ	A function
ρ	a real number
$\otimes$	Outer product
ξ	A function
$\times$	Cross product
γ	a real number
$[a, b)$	Half-open interval between a and b
(a, b)	Open interval between a and b

1. INTRODUCTION

Dynamical systems based learning is an emerging approach to learning movement. This thesis focuses in particular on studying autonomous dynamical systems as an approach to learning movement from demonstrations. The research is done on a proof-of-concept level as the approach is relatively recent in the literature.

Chapter 2 introduces the basic theoretical concepts related to dynamical systems and learning them. It should be noted that currently there is no consensus in the literature about the mathematical notation of the topic.

Chapter 3 introduces and analyses the three methods chosen for this study. The analysis is not the same for all three methods, but the main focus in all of the analyses is the form of the dynamical system, and how to evaluate and learn it. The limitations and computational complexity of each method is discussed and the chapter concludes with a brief description of other related methods.

Chapter 4 describes the experimental set-up and introduces the equipment used in the real life implementation. This chapter also explains some of the choices that were made for the software implementation and the describes the general software architecture of the final implementation. Finally, a networking issue affecting the reproduction of movement is briefly discussed.

Chapter 5 presents the recorded data and the learning results. The reproduction movement gained from the actual robotic arm movement is shown visually and the reproduction results based on computations are gathered into tables. Finally, the findings are discussed and a subjective comparison of the three methods is made.

2. THEORY

This thesis focuses on applying the introduced methods for generating motion in robotics. Although the subject matter introduced in this chapter is quite general and could be applied elsewhere, it should be remembered that this thesis assumes the application case to be in robotics.

2.1 Dynamical systems

In this thesis, a dynamical system means something that is represented by coupled ordinary differential equations where the derivatives of the variables are with respect to time t. In particular, the equations are real valued, first order and autonomous. A fixed collection of variables is called a state. If we have d variables, the system has the form of the following equation.

Definition 2.1 (**Autonomous dynamical system**). *Let $f_1, f_2, \dots, f_d : \mathbb{R}^d \mapsto \mathbb{R}$ be continuous. Then, an autonomous dynamical system is defined as:*

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_1 \\ \vdots \\ \dot{x}_d \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \dots, x_d) \\ f_2(x_1, x_2, \dots, x_d) \\ \vdots \\ f_d(x_1, x_2, \dots, x_d) \end{pmatrix}.$$

For more compact notation, a collection of variables is referred to with a single, bold-faced symbol. This collection is thought of as a vector- or matrix-valued variable. If, in addition, we collect the functions on the right-hand side of the equation (2.1) as a vector-valued function, then the equation then can be shortened in the following way:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} = \mathbf{x} \tag{2.2}$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_1 \\ \vdots \\ \dot{x}_d \end{pmatrix} = \dot{\mathbf{x}}, \tag{2.3}$$

$$\begin{pmatrix} f_1(x_1, x_2, \dots, x_d) \\ f_2(x_1, x_2, \dots, x_d) \\ \vdots \\ f_d(x_1, x_2, \dots, x_d) \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_d(\mathbf{x}) \end{pmatrix} = f(\mathbf{x}), \tag{2.4}$$

Definition 2.5 (**Vector form of an autonomous dynamical system**).

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}). \tag{2.6}$$

A discrete time forward solution for equation (2.6), given the initial point $\mathbf{x}_0$, is called a trajectory, or an evolution of state. A path is the trajectory without the time, meaning it is an ordered set of points. In this thesis however, the terms path and trajectory can be used interchangeably since it is assumed that the time information is known.

In the examples and illustrations, a state is interpreted as a position in 2 or 3 dimensions. In these cases, the motion of a point can be generated if the function of eq. (2.6) is known. Additionally, for an intuitive understanding it is often helpful to look upon the system as a vector field.

2.2 Stability of dynamical systems

Although there are many definitions of stability [16][9], this thesis uses just 2 of them: globally asymptotically stable and locally asymptotically stable. In general, the equilibrium point $\mathbf{x}^* : f(\mathbf{x}^*) = \mathbf{0}$ is said to have one of the 2 (or more) types of stability, but in this thesis the DS itself is said to to have one of them. This means that the the equilibrium is assumed to exist and the mentioned type of stability is applied to it.

Informally speaking, when a dynamical system is said to be globally, asymptotically stable, it means that when given an arbitrary initial position $\mathbf{x}_0 \in \mathbb{R}^d$ the solution settles to an equilibrium $\mathbf{x}^* \in \mathbb{R}^d$. Local asymptotic stability means that there exists a subset of the state space from which the DS settles to the equilibrium. It should be noted that $\mathbf{x}^*$ can be used interchangeably with $\mathbf{0}$. [9]

It is generally known that globally asymptotic stability can be defined via the Lyapunov function, which means that there is no need to know the solution to the system of ODEs. The dynamical systems discussed here are assumed to be globally, asymptotically stable, if not stated otherwise.

Asymptotic, global stability can be defined through a famous result of Lyapunov as follows.
Theorem 2.7 (**Lyapunov stability**). *A dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is globally, asymptotically stable with equilibrium at $\boldsymbol{x}^*$ when there exists a radially unbounded, sufficiently smooth Lyapunov-function $V(\boldsymbol{x})$ for which [9]:*

$$V(\boldsymbol{x}) > 0 \iff \boldsymbol{x} \neq \boldsymbol{x}^*, \tag{2.8}$$

$$\dot{V}(\boldsymbol{x}) < 0 \iff \boldsymbol{x} \neq \boldsymbol{x}^*, \tag{2.9}$$

$$V(\boldsymbol{x}) = 0 \iff \boldsymbol{x} = \boldsymbol{x}^*, \tag{2.10}$$

$$\dot{V}(\boldsymbol{x}) = 0 \iff \boldsymbol{x} = \boldsymbol{x}^*. \tag{2.11}$$

Proof. The proof is shown in [9]. □

The following remark is a useful identity, which will be used later.

Remark 2.12 (**Derivative of Lyapunov function in the direction of the system solutions**).

$$\dot{V}(\boldsymbol{x}) = \frac{dV}{dt} = \frac{dV}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt} = \nabla V \cdot \dot{\boldsymbol{x}} = \nabla V \cdot f(\boldsymbol{x}), \tag{2.13}$$

Using this identity in Lyapunov stability results in

$$\nabla V \cdot f(\boldsymbol{x}) < 0. \tag{2.14}$$

Essentially, the inequality (2.14) says that the velocity vector given by $\dot{\mathbf{x}} = f(\mathbf{x})$ can never point in a direction where the Lyapunov-function would decrease. For a quadratic Lyapunov-function $V(\mathbf{x}) = \|x\|^2$ for example, this would mean the trajectories can never move away from equilibrium.

2.3 Problem definition

In the broader context of learning methods, this thesis focuses on the methods that are called model-free behavioral cloning, as discussed in [19]. In this thesis, a model-free method means that the chosen methods do not explicitly take into account any actuator dynamics. The model and the limitations are assumed to underlie the demonstration data. Behavioral cloning means that the target of learning is the direct mapping between states and commands, a mapping between positions and velocities in this thesis' case, rather than any value function or suchlike.

Learning dynamical systems means learning the function of equation (2.6). In practice, this amounts to optimizing the parameters of a particular form of a function in terms of some suitable similarity or error metric. Since there is no established metric for measuring the success of DS-based learning, different methods might have varying definitions for it.

The demonstration data typically consists of sampled trajectories. The dynamics, such as velocities and accelerations, can be inferred or estimated from the trajectories if needed. One source for benchmark data is from Khansari-Zadeh [11]. It contains 2-dimensional trajectories depicting movement patterns for handwriting.

For example, the dataset could consist of N different demonstrations with T states in each. Then a datapoint would be a double (τ is the actual recorded time, t is index):

$$(\tau_{n,t},\ \mathbf{x}_{n,t}),\quad n = 1, \dots, N,\quad t = 1, \dots, T \tag{2.15}$$

For autonomous DS, the dataset is often thought of as a pair of positions, $\mathbf{x}$, and velocities $\dot{\mathbf{x}}$. Then each datapoint is thought of as:

$$(\mathbf{x}_{n,t},\ \dot{\mathbf{x}}_{n,t}) \quad n = 1, \dots, N, \quad t = 1, \dots, T. \tag{2.16}$$

It should be noted that each demonstration might have a different number of points. This possibility is ignored in the notation for the sake of clarity. One important notion of the presented form of a dynamical system is that no autonomous dynamical system can produce a self-intersecting trajectory, namely a loop. That would imply that one point in $\mathbb{R}^d$ is associated with two different velocity vectors, a property that cannot exist in the solution trajectory of such a system. Finally, the learning is intentionally defined with ambiguous terms.

Definition 2.17 (Problem statement). *Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ be an autonomous dynamical system and let $\boldsymbol{\theta}$ denote the set of parameters of the dynamical system function $f(\boldsymbol{x})$. Then the learning means finding the parameters that maximize a similarity between dynamical systems defined by $f(\boldsymbol{x})$ and the dynamical system responsible for the demonstrations, and which keep the dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ stable.*

This definition doesn't state what the similarity between dynamical systems means. This is discussed in the next section.

2.4 Similarity measures

It is not immediately obvious how similarity is defined when discussing, comparing, or learning dynamical systems or their trajectories.

One problem comes from the fact that there are finitely many points in the demonstration data and the DS is defined everywhere in the state space. However one of the aims of the DS-based approach is that it should generalize in the sense that it produces satisfactory results even if the initial positions and trajectories differ from the demonstrations. However, since we get the information of the DS to be learned from the data points a measure has to use those.

Another problem is that there is no single established and widely accepted measure for similarity. This results in using subjective intuition to judge the results, gained from visualizing the DS and the trajectories. As such, no methods can be compared to each other objectively.

Yet another problem comes from the issue of shape and speed. As discussed below, they can be separated and researchers are often only interested in learning the shape, as the speed can be modulated. But there is no exact definition for shape, which also adds to the problem. One method that tries to capture the intuition behind learning the shape was introduced in [12] and is called the swept error area, or SEA. It is well-suited for 2-dimensional cases but insufficient for higher dimensionalities. One way to generalize SEA could be to introduce

some sort of triangulation method in order to estimate the area of the surface exemplified in figure 14. of [12].

Ideally, there would be only one measure quantifying this similarity and this would be used both in learning and validating the results. However, an approach like SEA relies on solving the trajectory in order to evaluate the measure. As such, it is a computationally demanding approach and is not a realistic choice as a similarity measure to be used in optimization. A more computationally realistic choice is to compare the individual velocity vectors of the demonstrations and the DS. The downside of this measure is that it gives a rather limited view of the similarity.

In this thesis two measures are used to verify the results, but not to train the models. For reasons associated with the separation of speed and shape, they are intended to downplay the significance of speed in order to measure the shape more accurately. Algorithm 1 is a cumulative error-based measure which solves the DS as it is being calculated. This is done in a time-agnostic manner and gives a measure of dissimilarity rather than similarity. Algorithm 2 ignores time altogether and measures the similarity by the scaled cosine of their angle. Scaled means that the cosines are mapped to $[0, 1]$ instead of $[-1, 1]$. If the vectors are pointing in the same direction it gives out the value 1. The scaling is not necessary and it is done here only to emphasize it as a similarity measure. It should be noticed, that the normalized cosine similarity means scaled cosine similarity, later in the thesis.

Algorithm 1 Cumulative error measure

$N \leftarrow$ the number of points in demonstration trajectory
$\mathbf{x} \leftarrow$ first point in demonstration trajectory
$Error \leftarrow 0$
for $i = 1 \dots N - 1$ **do**
  $\mathbf{p} \leftarrow i$th point in demonstration path
  $\mathbf{q} \leftarrow (i + 1)$th point in demonstration path
  $StepSize \leftarrow \|\mathbf{q} - \mathbf{p}\|$
  $\dot{\mathbf{x}} \leftarrow f(\mathbf{x})$
  $\dot{\mathbf{x}} \leftarrow \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}$ ▷ Assuming $\|\dot{\mathbf{x}}\| \neq 0$
  $\mathbf{x} \leftarrow \mathbf{x} + StepSize \cdot \dot{\mathbf{x}}$
  $Error \leftarrow Error + \|\mathbf{x} - \mathbf{q}\|$
end for

Algorithm 2 Normalized cosine similarity

$\mathbf{p} \leftarrow$ a point in demonstration path
$\mathbf{q} \leftarrow$ the next point in demonstration path
$\dot{\mathbf{x}} \leftarrow f(\mathbf{p})$
$Similarity \leftarrow \frac{\dot{\mathbf{x}} \cdot (\mathbf{q} - \mathbf{p})}{\|\dot{\mathbf{x}}\| \|\mathbf{p} - \mathbf{q}\|}$ ▷ Assuming $\|\dot{\mathbf{p}}\| \neq 0$ or $\|\mathbf{q} - \mathbf{p}\| \neq 0$
$Similarity \leftarrow \frac{Similarity + 1}{2}$

2.5 Separating shape and speed in a dynamical system

This thesis focuses only on the shape of the dynamical system rather than its speed. This is due to the limitations of the actuators. They require sufficiently smooth and bounded signals and the implementation details complicate their delivery.

In this thesis, the shape will be defined by the direction of the velocity vectors while the speed will be defined by their magnitude. Therefore, the learning of the dynamical system of the demonstration can be separated into two phases. In the first phase only the shape is taken into account when applying a similarity metric. This allows more freedom in designing or modifying the algorithms for learning. After the shape is deemed fit, the magnitude of the velocity vectors can be transformed to be nearly uniform and then modulated with a suitable function. This is shown more precisely in the following definition.

Definition 2.18 (**Modulating the speed of a dynamical system**). *Let $\epsilon > 0$ be a small constant and $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ globally, asymptotically stable. Then a new similarly shaped DS can be defined as follows. Let*

$$\mu : \mathbb{R}^d \mapsto [0, \infty) \tag{2.19}$$

be smooth. Then the new DS is

$$\dot{\boldsymbol{x}} = g(\boldsymbol{x}) = \frac{\mu(\boldsymbol{x}) f(\boldsymbol{x})}{\|f(\boldsymbol{x})\| + \epsilon}. \tag{2.20}$$

In definition 2.18 the $\epsilon > 0$ is a small constant to avoid being undefined at the equilibrium point. The error resulting from its use is negligible.

After this transformation, the learning of the speed profile can be performed. The speed profile is essentially the modulating function $\mu : \mathbb{R}^d \mapsto (0, \infty)$, which is a much simpler case to learn than the shape. The modulation does not change the stability, as is shown by the following theorem.

Theorem 2.21. *The dynamical system of equation* (2.20) *is globally, asymptotically stable if the original dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is globally asymptotically stable.*

Proof. Since the original dynamical system is globally, asymptotically stable there exists a Lyapunov-function as defined in theorem 2.7. Since $\mu(\mathbf{x}), \epsilon$ are both positive, it follows that

$$\nabla V \cdot f(\mathbf{x}) < 0 \iff \frac{\nabla V \cdot f(\mathbf{x}) \mu(\mathbf{x})}{\|f(\mathbf{x})\| + \epsilon} < 0 \tag{2.22}$$

and

$$\nabla V \cdot f(\mathbf{x}) = 0 \iff \frac{\nabla V \cdot f(\mathbf{x}) \mu(\mathbf{x})}{\|f(\mathbf{x})\| + \epsilon} = 0. \tag{2.23}$$

Therefore $V(\mathbf{x})$ is a Lyapunov-function for $\dot{\mathbf{x}} = \frac{\mu(\mathbf{x}) f(\mathbf{x})}{\|f(\mathbf{x})\| + \epsilon}$ and the system is globally, asymptotically stable. □

2.6 Single and multiple demonstration patterns and using the dynamical system

The demonstrations usually contain trajectories starting from multiple different positions. When there is only one cluster (in the statistical sense of the word) of starting points the term demonstration pattern is used. When there are multiple clusters the demonstration contains multiple demonstration patterns. This difference is best exemplified by first looking at figure 2.1 in which there is only one cluster of starting points in the upper right corner with all the trajectories ending at the origin. Then it can be compared to figure 2.2, which has 3 clusters.

Figure 2.1. *Single demonstration pattern (from [11]). The trajectories start from upper right hand corner*

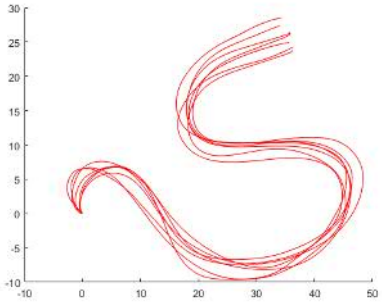
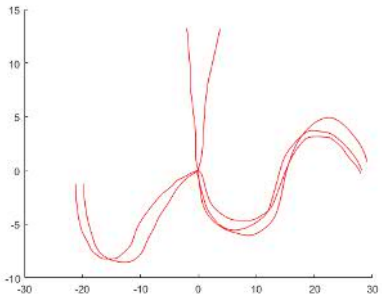


Figure 2.2. *Multiple demonstration patterns(from [11]). The trajectories start from the outer areas and all end in the same point in the center.*



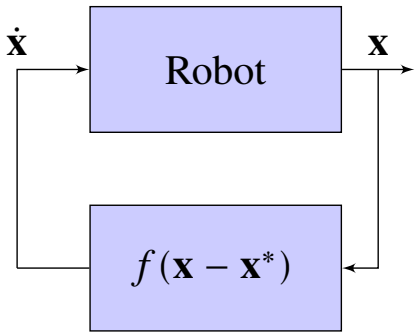
The learned dynamical system is used with a robot by giving it velocity commands in real time. Figure 2.3 shows a simplified architecture for using the function. The robot is assumed to have internal velocity controller set up to execute the velocity commands. It should be noted that the goal point $\mathbf{x}^*$ is used to transform the dynamical system coordinates to the origin. This transformation does not affect the stability [9]. The background assumption for the function in figure 2.3, is that the equilibrium point for the DS defined by the f is at $\mathbf{0}$.

One practical issue with using the DS in a real system is getting it started. Often the system starts from rest, and due to inertia the accelerations are bounded. In the implementation part of this thesis a smooth step signal, the logistic function, was used to start the DS with parameters found by simple trial and error.

$$g(t, \mathbf{x}) = \frac{m}{1 + e^{-k(t-t_0)}} f(\mathbf{x}) \tag{2.24}$$

Although using this renders the system time-dependent in principle, it does not have any effect on the theory introduced here. In equation (2.24), $m \in \mathbb{R}^+$ is the maximum value of the step modulating the DS, and $k \in \mathbb{R}^+$ is the steepness of the curve and $t_0 \in \mathbb{R}^+$ is the value that dictates when the smooth signal reaches the median value of the maximum.

Figure 2.3. *Simplified control architecture for using the dynamical system in a robot*



A final aspect of using the DS is the stability of the whole system. The robot with a velocity controller is its own dynamical system, but this thesis is not concerned of its details. It is only assumed that the real robot follows velocity commands fast enough and is stable. More details about robot dynamics and control can be found, for example, in [3] or [2].

The stability of the combination of the robot and the DS is discussed in terms of the figure 2.3 in the next remark.

Remark 2.25. *The dynamical system "Robot" is assumed to stable. Also, the dynamics of this controlled system is assumed to be fast enough, so that the velocity command seems instantaneously executed, when compared to the timescale of the loop in figure 2.3.*
Under these assumptions the system of figure 2.3 is globally and asymptotically stable, because the dynamical system defined by $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ *is globally and asymptotically stable.*

3. METHODS

3.1 SEDS

3.1.1 Definition

SEDS stands for Stable Estimator of Dynamical Systems. It is derived from the viewpoint that the state $\mathbf{x}$ and velocity $\dot{\mathbf{x}}$ are jointly distributed random variables. The probability distribution function is composed from K gaussian distributions forming a mixed gaussian model. The estimate for velocity defining the dynamical system is derived as being the expected value of velocities posterior distribution. For further details of this, see [4], [10], [13].

Before SEDS can be fully defined, it is necessary to introduce a result shown in [10] to give some background knowledge.

Theorem 3.1 (**Special case of velocity posterior mean**). *Let $\boldsymbol{X}$ and $\dot{\boldsymbol{X}}$ both be d-dimensional, jointly distributed random vectors and let $\hat{K}$ be a random element of* $\{1, 2, ..., K\}$.
Also, let the joint probability density function be defined by

$$P\begin{pmatrix}\boldsymbol{X}=\boldsymbol{x}\\ \dot{\boldsymbol{X}}=\dot{\boldsymbol{x}}\end{pmatrix} = \sum_{k=1}^{K} \frac{P(\hat{K}=k)}{\sqrt{(2\pi)^{2d}|\boldsymbol{\Sigma}^k|}} e^{-0.5\left(\begin{pmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\end{pmatrix}-\mu^k\right)^T \Sigma_{\boldsymbol{x}}^{k^{-1}} \left(\begin{pmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\end{pmatrix}-\mu^k\right)} \tag{3.2}$$

Additionally, restrict the case by

$$\mu_{\dot{\boldsymbol{x}}}^k = \boldsymbol{A}_k \mu_{\boldsymbol{x}}^k. \tag{3.3}$$

where

$$\boldsymbol{A}_k = \boldsymbol{\Sigma}_{\dot{\boldsymbol{x}}\boldsymbol{x}}^k (\boldsymbol{\Sigma}_{\boldsymbol{x}}^k)^{-1}. \tag{3.4}$$

Then a posterior distribution mean of $\dot{\boldsymbol{X}}$ is given by:

$$P(\dot{\boldsymbol{X}}=\dot{\boldsymbol{x}}|\boldsymbol{X}=\boldsymbol{x}) = \sum_{k=1}^{K} h_k(\boldsymbol{x})\boldsymbol{A}_k(\boldsymbol{x}-\boldsymbol{x}^*). \tag{3.5}$$

where

$$h_k(\boldsymbol{x}) = P(\hat{K}=k \mid \boldsymbol{X}=x) = \frac{P(\hat{K}=k)P(\boldsymbol{X}=\boldsymbol{x}|\hat{K}=k)}{\sum_{i=1}^{K} P(\hat{K}=i)P(\boldsymbol{X}=\boldsymbol{x}|\hat{K}=i)}. \tag{3.6}$$

Proof. The proof can be found by first considering the general case proved in ([4]) and then restricting the case for stability as shown in [10]). □

Because SEDS stands for *Stable* Estimator of Dynamical Systems, it has to be shown when the DS defined by (3.5) is stable.

Theorem 3.7. *The dynamical system*

$$\dot{\boldsymbol{x}} = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \boldsymbol{A}_k (\boldsymbol{x} - \boldsymbol{x}^*)$$

is globally, asymptotically stable, when each $\boldsymbol{A}_k$ is negative definite.

Proof. Let $V(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}^*\|^2$ and each $\mathbf{A}_k$ be negative definite. Clearly $V(\mathbf{x}) > 0 \iff \mathbf{x} \neq \mathbf{x}^*$ and $V(\mathbf{x}) = 0 \iff \mathbf{x} = \mathbf{x}^*$.

$$\dot{V}(\mathbf{x}) = \nabla V(\mathbf{x}) \cdot \dot{\mathbf{x}} = (\mathbf{x} - \mathbf{x}^*)^T \sum_{k=1}^{K} h_k(\mathbf{x}) \mathbf{A}_k (\mathbf{x} - \mathbf{x}^*).$$

$$= \sum_{k=1}^{K} h_k(\mathbf{x}) (\mathbf{x} - \mathbf{x}^*)^T \mathbf{A}_k (\mathbf{x} - \mathbf{x}^*) < 0 \tag{3.8}$$

since each $h_k(\mathbf{x}) > 0$ and $\mathbf{A}_k$ is negative definite. Additionally

$$\dot{V}(\mathbf{x}) = \sum_{k=1}^{K} h_k(\mathbf{x}) (\mathbf{x} - \mathbf{x}^*)^T \mathbf{A}_k (\mathbf{x} - \mathbf{x}^*) = 0 \iff \mathbf{x} = \mathbf{x}^*, \tag{3.9}$$

so $V(\mathbf{x})$ is a Lyapunov-function for $\dot{\mathbf{x}} = \sum_{k=1}^{K} h_k(\mathbf{x}) \mathbf{A}_k (\mathbf{x} - \mathbf{x}^*)$ and is thus globally, asymptotically stable. □

Finally, the definition follows from the previous results.

Definition 3.10 (SEDS). *Stable estimator for dynamical systems (SEDS) is the dynamical system gained, when the velocity is interpreted as the posterior mean estimate shown in equation* (3.5) *and restricting the case to stable dynamics:*

$$\dot{\boldsymbol{x}} = \sum_{k=1}^{K} h_k(\boldsymbol{x}) \boldsymbol{A}_k (\boldsymbol{x} - \boldsymbol{x}^*), \text{ each } \boldsymbol{A}_k \text{ is negative definite} \tag{3.11}$$

Remark 3.12. *It should be noticed, that the $\boldsymbol{A}_k$ might not be symmetric. a simple result from linear algebra states that:*

$$\textbf{\textit{A nonsymmetric matrix }}\boldsymbol{M} \textit{ is negative definite} \iff \boldsymbol{M}+\boldsymbol{M}^T \textit{ is negative definite}. \tag{3.13}$$

This result is used to verify the negative definiteness of $\boldsymbol{A}_k$, when training the model.

3.1.2 Training SEDS

Training the SEDS starts by interpreting the data, which are samples from the joint distribution defined by the probability density function in equation (3.2). To be more specific, the data should be in a similar form as is shown in equation (2.16). This allows the natural use of the similarity measure of log-likelihood to be used for training the SEDS. Also, the mean square error can be used as a dissimilarity measure as shown in the original article. In

the case of this thesis, only the log-likelihood version was used for the reasons mentioned in [10].

If the learning is thought of only as an optimization for maximal likelihood, the resulting DS might not be stable. This is avoided by posing the problem as a constrained optimization problem with the constraints ensuring that the parameters define the probability distributions and that the resulting system is stable. For convenience, let

$$\boldsymbol{\theta} = \left\{P(\hat{K} = 1), \mu_{\mathbf{x}}^1, \Sigma^1, ..., P(\hat{K} = K), \mu_{\mathbf{x}}^k, \Sigma^K\right\}. \tag{3.14}$$

denote the parameters of the probability density function shown in equation (3.2). Notice that the $\mu_{\dot{\mathbf{x}}}^k$ is not included as it assumed to be constrained by equation (3.3).

Definition 3.15 (Training as constrained optimization). *Let $\boldsymbol{\theta}$ denote the parameters of SEDS. A measure for dissimilarity or error is given by the log-likelihood of the joint probability distribution:*

$$J(\boldsymbol{\theta}) = -\frac{1}{TN}\sum_{n=1}^{N}\sum_{t=1}^{T} log(P\begin{pmatrix}\boldsymbol{X} = \boldsymbol{x}_{n,t}\\ \dot{\boldsymbol{X}} = \dot{\boldsymbol{x}}_{n,t}\end{pmatrix}). \tag{3.16}$$

The parameters for SEDS to reproduce the demonstrative motion can be found by solving this constrained optimization problem for $\boldsymbol{\theta}$:

$$\underset{\theta}{\textbf{minimize}}\, J(\boldsymbol{\theta}) \tag{3.17}$$

subject to

$$\begin{cases} \boldsymbol{A}_k + \boldsymbol{A}_k^T \text{ is negative definite for each } k \\ \text{each } \Sigma^k \text{ is positive definite and symmetric} \\ \sum_{k=1}^{K} P(\hat{K} = k) = 1 \\ P(\hat{K} = i) \in (0, 1],\ i = \{1, \dots, K\}. \end{cases} \tag{3.18}$$

One important feature of implementing the training algorithm is how to choose the initial values. The authors of SEDS have suggested one such method, but it was noticed in this implementation that sometimes their method leads to numerical problems such as covariance of state $\Sigma_{\mathbf{x}}^k$) losing rank. As this was mostly the case when the SEDS had more than two components ($k > 2$), a better method for obtaining the initial values was to split the data into k parts and fit an initial SEDS to each part with just one gaussian component.

For practicality, the covariance matrices Σ^k can be stored as their Cholesky decompositions L_k, $\Sigma^k = L_k L_k^T$. This way the number of parameters to be stored in the memory can be reduced and the covariance be kept as a positive definite symmetric matrix. Also the $\mu_{\dot{\mathbf{x}}}$ does not to be stored during optimization as it is derived from other parameters. Lastly, the negative definiteness of A_k can be evaluated with Sylvester's criterion for $A_k + A_k^T$ [13].

3.1.3 Computational complexity of training SEDS

The number of parameters to be optimized when training SEDS is [10]:

$$\#parameters = K(1 + 2d + 2d^2). \tag{3.19}$$

The rate of training is of course dependent on the constrained nonlinear optimization algorithm. The speed can be increased using the analytical form of gradient for the cost function $J(\boldsymbol{\theta})$. However the implemented algorithm for the following results did not use those analytical derivations, and as such the gradient evaluation was done numerically by the used software.

3.1.4 Limitation of SEDS

The main limitation can be derived from the Lyapunov-function that underlies the optimization constraints for training SEDS. Since the quadratic Lyapunov function is symmetric, its contour lines are circles and therefore the negative gradient vector always points to goal $\mathbf{x}^*$. Because of this the trajectories cannot move away from the goal. The following remark discusses this more accurately.

Remark 3.20 (SEDS can not move away from goal). *Assuming that $V(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}^*\|^2$ is the Lyapunov function, the stability requirements can be rephrased as:*

$$\frac{dV}{dt} = \nabla V(\boldsymbol{x}) \cdot f(\boldsymbol{x}) \tag{3.21}$$

$$= \|\nabla V(\boldsymbol{x})\| \cdot \|f(\boldsymbol{x})\| \cos(\angle(\nabla V(\boldsymbol{x}), f(\boldsymbol{x}))) < 0, \tag{3.22}$$

$$\iff \cos\Big(\angle\Big(-(\boldsymbol{x} - \boldsymbol{x}^*), f(\boldsymbol{x})\Big)\Big) > 0. \tag{3.23}$$

Using the rephrasing it can be concluded that the angle between the DS velocity vector and the vector pointing to the $\boldsymbol{x}^*$ can not reach an angle that would result in the velocity pointing away from the goal:

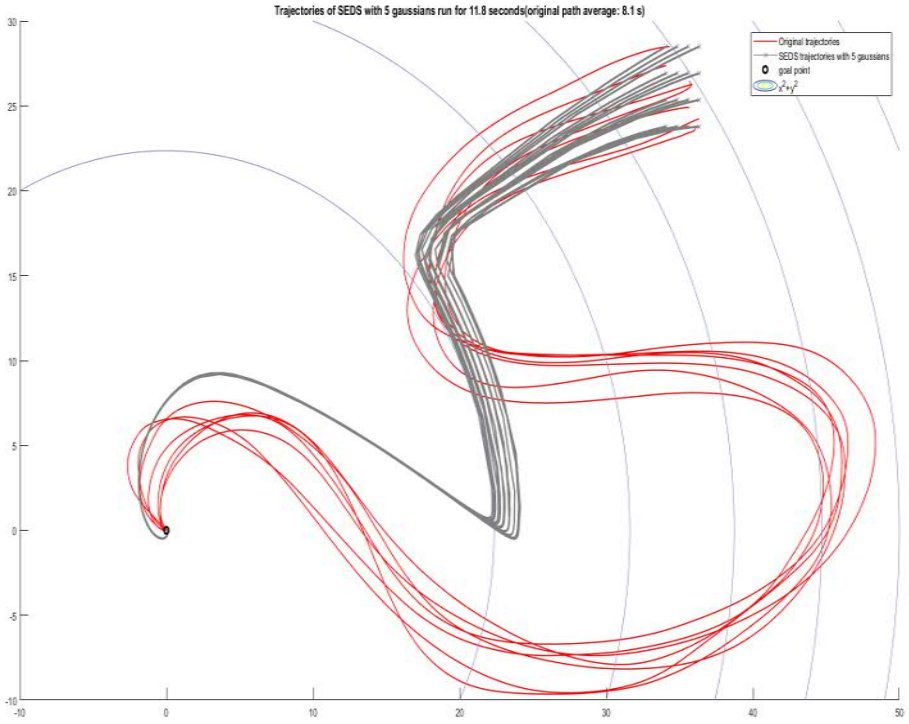
$$\angle\Big(-(\boldsymbol{x} - \boldsymbol{x}^*),\, f(\boldsymbol{x})\Big) \in (-\frac{\pi}{2}, \frac{\pi}{2}). \tag{3.24}$$

In figure 3.1 the circular lines are the contours of the quadratic Lyapunov function, the red curves are the demonstrations and the grey curves are SEDS trajectories with starting points denoted by small crosses. Origin is used as the goal point. The figure demonstrates how the trajectory can parallel the contour but never move away from the goal.

3.2 Diffeomorphic method

In this thesis, a diffeomorphic method means any learning method that uses diffeomorphic dynamical systems to reproduce the demonstrations. For contrast, in [17] the diffeomorphisms were used to compensate for the main limitation of SEDS. However, this thesis focuses on type of method introduced in [21]. It uses diffeomorphisms more directly than in [17]. The authors of [21] provided a theoretical background for the method, which will be discussed in this section.

Figure 3.1. *Two dimensional illustration of the limitation of SEDS. Red lines are the demonstration trajectories (from [11]). The computational reproductions are the grey lines. Circular lines depict the contours of Lyapunov-function*



3.2.1 Diffeomorphic dynamical systems

Diffeomorphic dynamical systems are based on the notion of diffeomorphism. Although it can be defined more generally ([15]), in this thesis the diffeomorphism will defined as follows.

Definition 3.25 (**Diffeomorphism**)**.**

$$\boldsymbol{\Phi} : \mathbb{R}^d \mapsto \mathbb{R}^d, \tag{3.26}$$

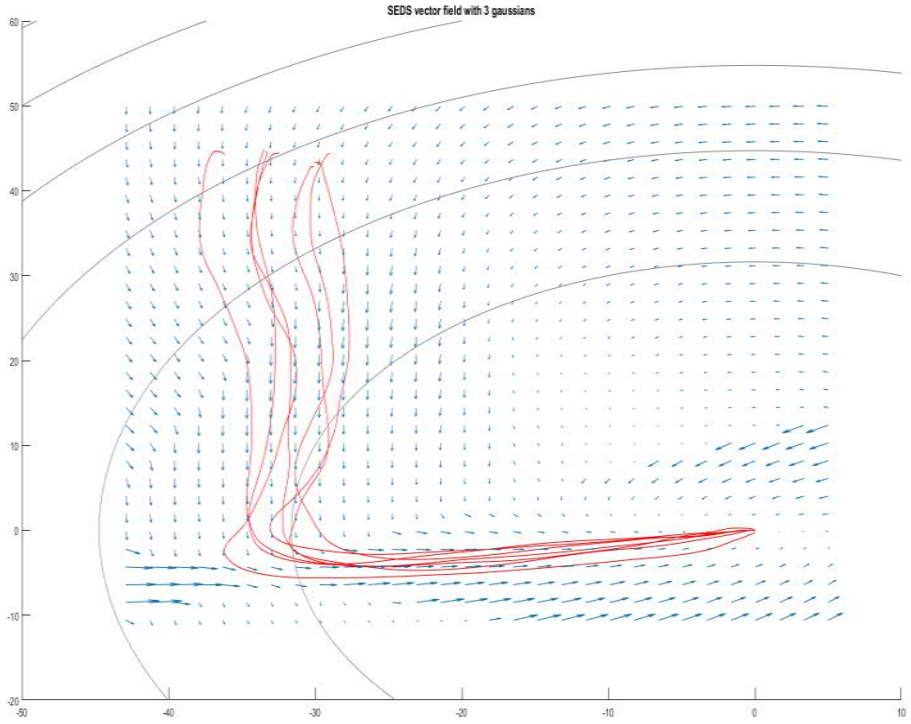
$$\boldsymbol{\Phi} \text{ is smooth and has a smooth inverse, } \boldsymbol{\Phi}^{-1}. \tag{3.27}$$

The diffeomorphism is a function that moves points of $\mathbb{R}^d$ in a smooth and invertible manner. To make them applicable to dynamical systems, the following definition is used.

Definition 3.28 (**Diffeomorphic dynamical systems**)**.** *Let $J_{\boldsymbol{\Phi}}$ denote the Jacobian of the diffeomorphism $\boldsymbol{\Phi}$. When two dynamical systems, $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and $\dot{\boldsymbol{x}} = g(\boldsymbol{x})$ are diffeomorphic, it means that there exists a diffeomorphism $\boldsymbol{\Phi}$ for which [21]:*

$$f(\boldsymbol{\Phi}(\boldsymbol{x})) = J_{\boldsymbol{\Phi}}(\boldsymbol{x}) g(\boldsymbol{x}). \tag{3.29}$$

Figure 3.2. *Example SEDS vector field for simpler shape. Data is from [11]. The demonstration trajectory are the continuous red lines, the velocity vectors of SEDS are the blue arrows and the circular gray lines are the contours of Lyapunov-function*



In this thesis the diffeomorphic DS defined by $g(\mathbf{x})$ will be $\dot{\mathbf{x}} = g(\mathbf{x}) = -\mathbf{x}$. This choice has two benefits. The first is that it provides an intuitive understanding of what the diffeomorphisms do in that case. Because all the time forward solutions of $\dot{\mathbf{x}} = -\mathbf{x}$ are straight lines, the diffeomorphisms transform these straight lines into more complex curves. The second and more important benefit is the result stated in the next theorem. Also, for this reason the DS $\dot{\mathbf{x}} = -\mathbf{x}$ is used, as it is globally asymptotically stable.

Theorem 3.30. *If one of two diffeomorphic dynamical systems is globally, asymptotically stable, then they both are.*

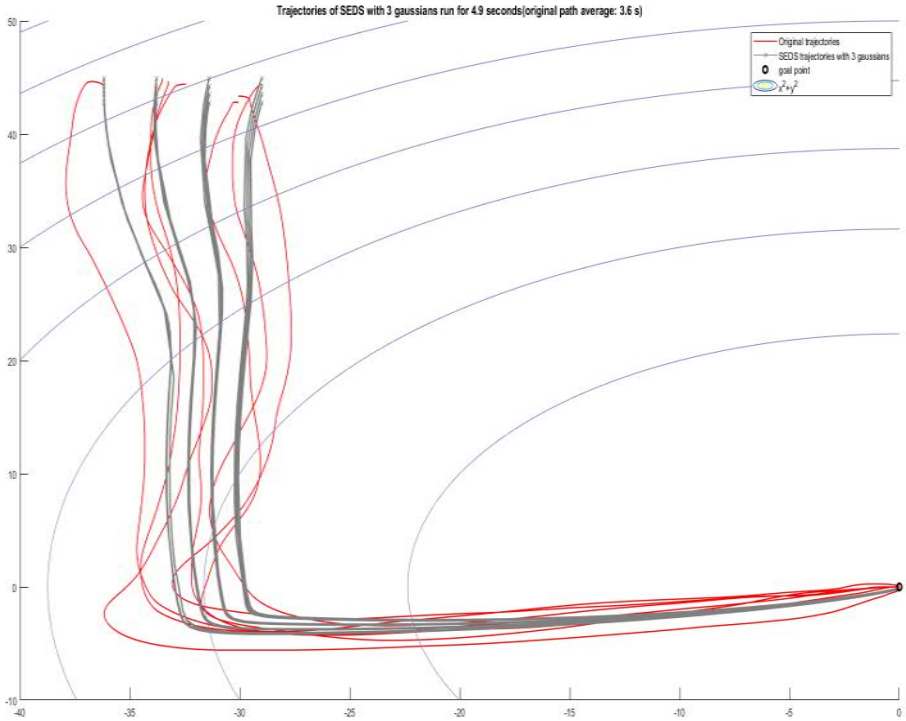
Proof. The proof is shown in [21]. □

How to get the necessary diffeomorphism is yet to be defined, and will be dealt with later. Assuming that $\boldsymbol{\Phi}(\mathbf{x})$ is somehow gained from the data, the DS to reproduce them can be derived as follows.

Remark 3.31. *Let $\boldsymbol{\Phi}$ be a diffeomorphism learned from the demonstrative data. Then, the following DS reproduces the demonstrations in a stable manner([21]):*

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) = -J_{\boldsymbol{\Phi}}(\boldsymbol{\Phi}^{-1}(\boldsymbol{x}))\boldsymbol{\Phi}^{-1}(\boldsymbol{x}). \tag{3.32}$$

Figure 3.3. *Example SEDS reproduction trajectories for simpler shape (from [11]). The demonstration trajectories are the continuous red lines, the reproduction trajectories are the gray lines and circular lines are the contours of Lyapunov-function*



3.2.2 Diffeomorphisms from local translations

The diffeomorphisms in this thesis are a sequential combination of local translations. The local translations must be diffeomorphisms themselves, which has been proved by the authors of [21]. Most of the evaluations of the main diffeomorphism stem from evaluating the inverse of a local translation. The definition for local translations will be introduced later.

Definition 3.33 (**Diffeomorphism from local translations**). *Let K be the number of local translations used. Then the diffeomorphism from the local translations comes from*

$$\Phi(\boldsymbol{x}) = \psi_K \circ \psi_{K-1} \circ \ldots \circ \psi_1(\boldsymbol{x}), \tag{3.34}$$

for which the inverse is found by

$$\Phi^{-1}(\boldsymbol{x}) = \psi_1^{-1} \circ \psi_2^{-1} \circ \ldots \circ \psi_K^{-1}(\boldsymbol{x}). \tag{3.35}$$

To evaluate the diffeomorphic dynamical system function, it is necessary to find the Jacobian J_Φ of the diffeomorphism Φ. Finding the Jacobian of the diffeomorphism is based on finding the Jacobians of the local translations. By using some calculus, a result comes in the form of following remark.

Remark 3.36. *the Jacobian of the diffeomorphism from sequential local translations is found by:*

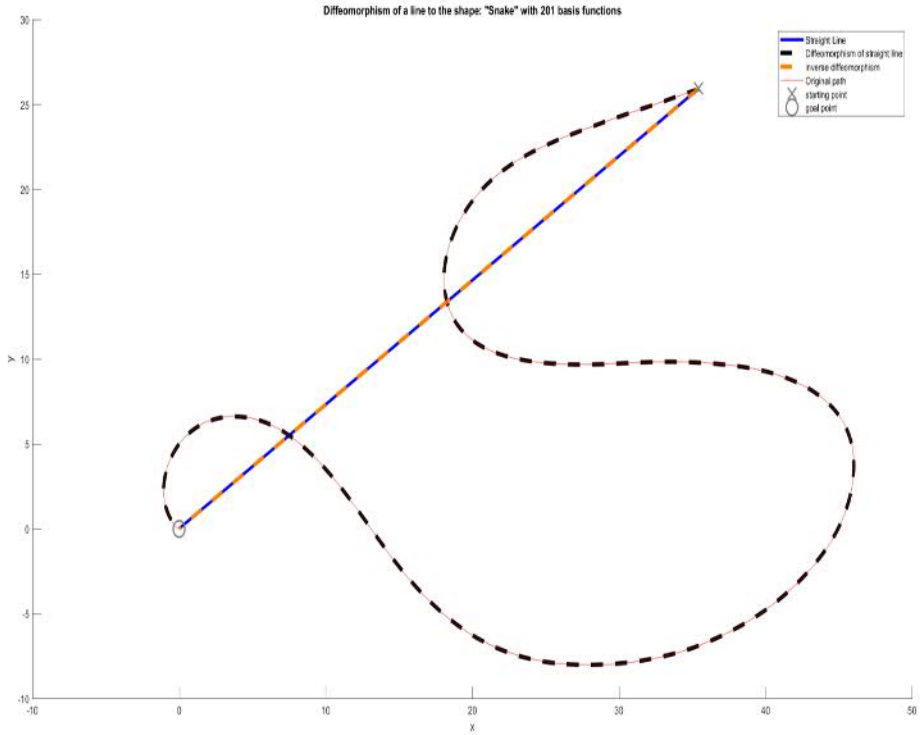
$$J_{\Phi}(\boldsymbol{x}) = \frac{d\psi_K}{d\psi_{K-1}} \frac{d\psi_{K-1}}{d\psi_{K-2}} \dots \frac{d\psi_1}{d\boldsymbol{x}}. \tag{3.37}$$

This can be put into a different form of:

$$J_{\Phi} = J_{\psi_K}(\psi_{K-1} \circ \psi_{K-2} \circ \dots \psi_1(\boldsymbol{x})) J_{\psi_{K-1}}(\psi_{K-2} \circ \psi_{K-3} \circ \dots \psi_1(\boldsymbol{x})) \dots$$
$$\dots J_{\psi_2}(\psi_1(\boldsymbol{x})) J_{\psi_1}(\boldsymbol{x}). \tag{3.38}$$

from which we see how the Jacobians of the local translations become necessary. The Jacobian of a local translation can be found analytically and the formula for it is shown in the next subsection.

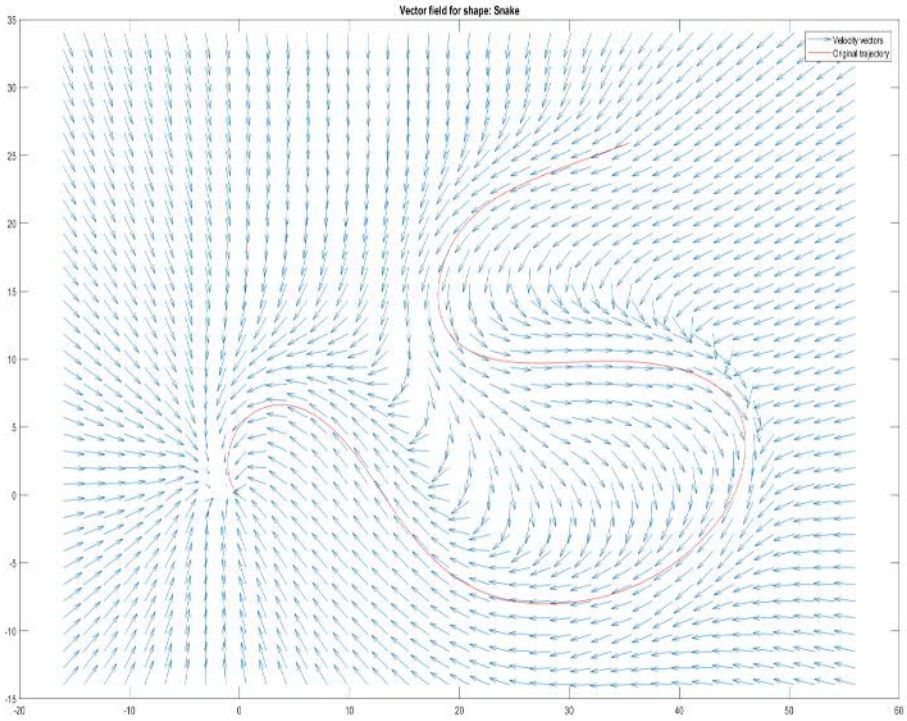
***Figure 3.4**. Diffeomorphism of a straight line into a more complex curve. The narrow continuous red line is the average trajectory of demonstrations of the same dataset as in figure 2.1. The dashed black line is the straight line of blue dashed line gone through the diffeomorphism. The dashed orange line is the dashed black line gone through the inverse of the diffeomorphism. This diffeomorphism was used to derive the DS in figure 3.5*



3.2.3 Local translations

So far, the local translations have been used but not defined. An intuitive understanding of a local translation is that it translates a point of $\mathbb{R}^d$ along vector $\mathbf{v}$. The translation is forced to act locally by the kernel function.

Figure 3.5. *Vector field of diffeomorphic dynamical system. The continuous red line is the average trajectory of demonstrations of the same dataset as in figure 2.1. The blue arrows are modulated velocity vectors of the DS*



Definition 3.39 (**Local translation and its Jacobian**). *Let* $\rho \in \mathbb{R}$ *and* $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^d$. *The local translation is a function* $\psi : \mathbb{R}^d \mapsto \mathbb{R}^d$ *defined by*

$$\psi_{\rho,\boldsymbol{p},\boldsymbol{v}}(\boldsymbol{x}) = \boldsymbol{x} + k_{\rho,\boldsymbol{p}}(\boldsymbol{x}) \cdot \boldsymbol{v}. \tag{3.40}$$

The function $k_{\rho,\boldsymbol{p}} : \mathbb{R}^d \mapsto (0, 1]$ *is a smooth radial basis function defined by:*

$$k_{\rho,\boldsymbol{p}}(\boldsymbol{x}) = e^{-\rho^2 \|\boldsymbol{x}-\boldsymbol{p}\|^2}. \tag{3.41}$$

Finally, the Jacobian of a local translation is found by

$$J_\psi(\boldsymbol{x}) = I - 2\rho^2 k(\rho, \boldsymbol{x}, \boldsymbol{p}) \cdot \boldsymbol{v} \otimes (\boldsymbol{x} - \boldsymbol{p}) = I - 2\rho^2 k(\rho, \boldsymbol{x}, \boldsymbol{p}) \cdot \boldsymbol{v}(\boldsymbol{x} - \boldsymbol{p})^T. \tag{3.42}$$

The main idea of the kernel function $k_{\rho,\mathbf{p}}(\mathbf{x})$ is to approach the zero value when still far away from the chosen point $\mathbf{p}$ and to allow the translation to happen gradually in circular area with a radius defined by the parameter ρ. An illustration of this is shown in figure 3.7, in which the area of effect for the local translation is indicated by the blue hue and an arrow shows the vector $\mathbf{v}$ of the translation. The parameters ρ defines the size of the area and $\mathbf{p}$ is its center.

The local translation has to be invertible and that requirement was shown in [21] to be defined by the parameters ρ and $\mathbf{v}$:

$$\rho\|\mathbf{v}\| < \frac{1}{\sqrt{(2)}}e^{\frac{1}{2}}. \tag{3.43}$$

In other words, in order for the local translation to be invertible, the area of effect must be greater than an area defined by a radius of $\frac{1}{\sqrt{(2)}\|\mathbf{v}\|}e^{\frac{1}{2}}$.
For the local translation to be of any practical interest, its inverse has to be evaluable. The authors of [21] showed that the inverse is of the form:

$$\psi^{-1}(\mathbf{x}) = \mathbf{x} + r \cdot \mathbf{v}, \tag{3.44}$$

where the r is defined by position $\mathbf{x}$ and the parameters $\rho, \mathbf{p}, \mathbf{v}$. Evaluating the inverse of the local translations then results in finding the root of a function $h : \mathbb{R} \mapsto \mathbb{R}$ which means finding r for $h(r) = 0$. The function h itself is defined as

$$h(r) = r + k_{\rho,\mathbf{p}}(\mathbf{x} + r \cdot \mathbf{v}). \tag{3.45}$$

As the root can not be found analytically, it has to be found by applying a numerical method. For this, the straightforward use of Newton's method suffices with the iterations being defined by:

$$r_{n+1} = r_n - \frac{h(r_n)}{h'(r_n)}. \tag{3.46}$$

In this, the $h'(r)$ can be found analytically:

$$h'(r) = 1 - 2\rho^2((\mathbf{x} + r \cdot \mathbf{v} - \mathbf{p}) \bullet \mathbf{v})k(\rho, \mathbf{x} + r \cdot \mathbf{v}, \mathbf{p}). \tag{3.47}$$

Inverting the local translation is summarized in the next remark.

Remark 3.48 (**Evaluating the inverse of a local translation**). *Find value for r by iterating*

$$r_{n+1} = r_n - \frac{h(r_n)}{h'(r_n)}. \tag{3.49}$$

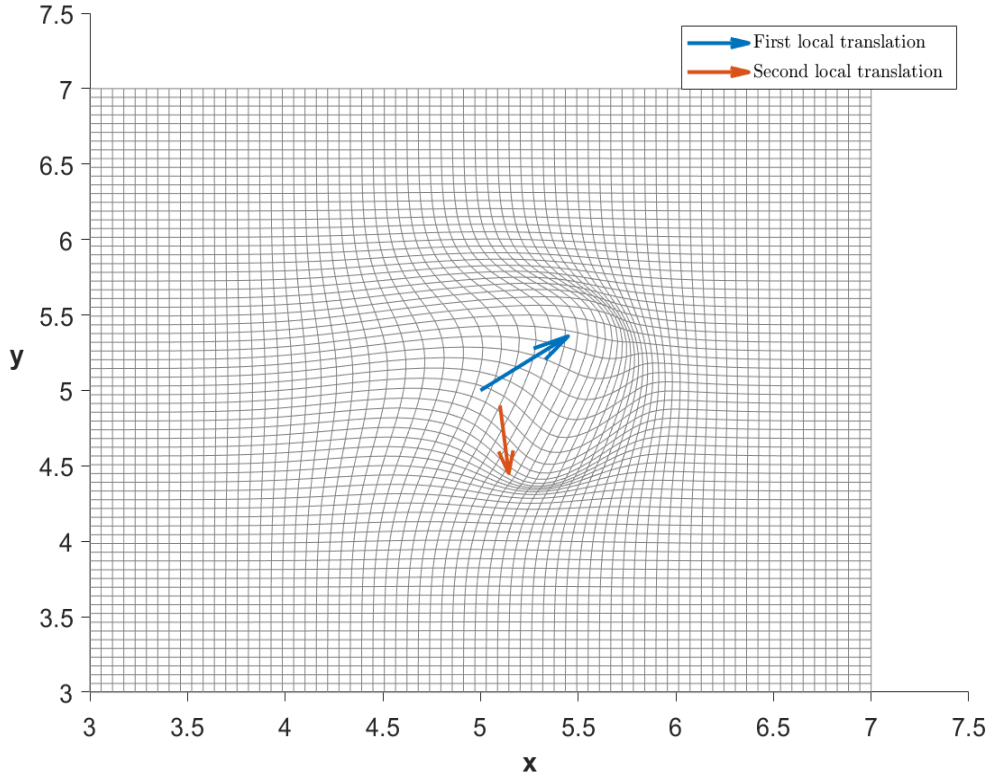
Then the inverse of a local translation is found simply by

$$\phi^{-1}(\boldsymbol{x}) = \boldsymbol{x} + r \cdot \boldsymbol{v}. \tag{3.50}$$

3.2.4 The algorithm for finding the local translations

In [21] the authors provided a computationally fast heuristic to derive a diffeomorphism between straight line and a curve. A 2-dimensional example case is shown in figure 3.4. In it, the diffeomorphism is between a straight line and an averaged curve from one demonstration pattern in the LASA handwriting dataset [11]. The heuristic is fast and accurate for most

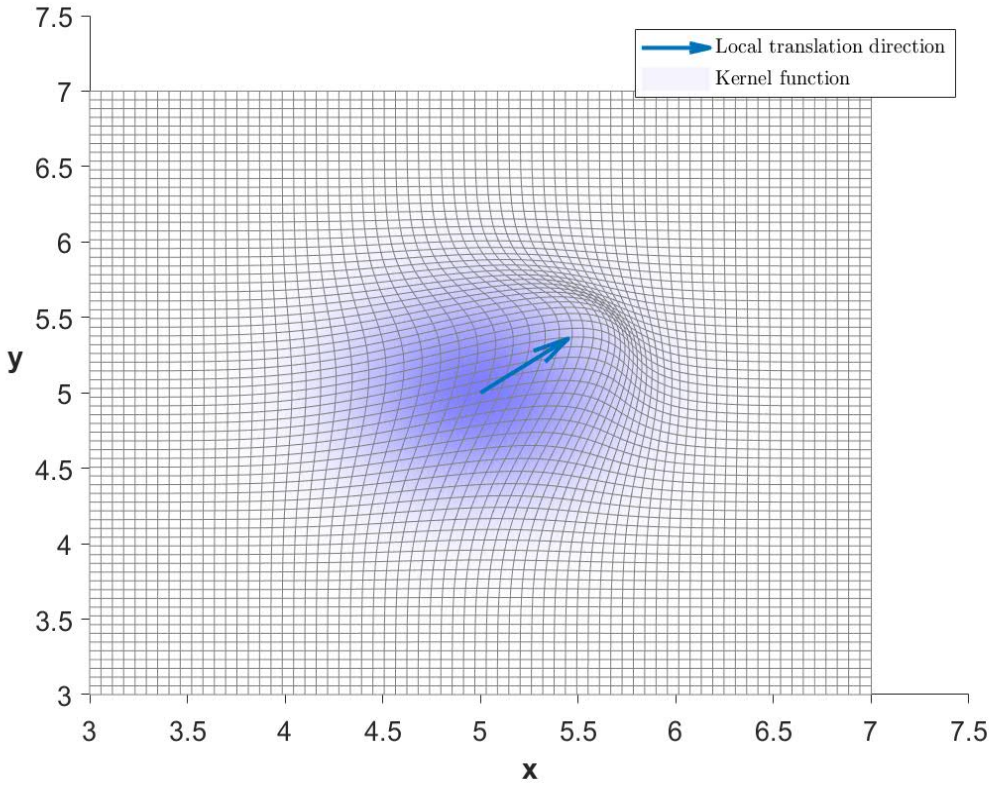
Figure 3.6. *Two sequential local translations in 2-dimensions depicting how they bend the surrounding space. The blue and orange arrows are the parameter $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ of the local translations.*



cases, but it restricts the learning to only a single demonstration pattern and relies on the average demonstration pattern being smooth enough. In one sense, this way of learning does not use the dynamical system similarity measure discussed earlier, but a surrogate similarity measure which only compares points assumed to lie on the curves.

Considering that the final diffeomorphism has K local translation and each local translation has parameters $\rho, \mathbf{p}, \mathbf{v}$, the diffeomorphic dynamical system then needs $K(1 + 2d)$ parameters. The learning of the diffeomorphic dynamical system could then be posed as a constrained optimization problem, if a satisfactory similarity measure can be used. The constraints ensure the diffeomorphism is invertible and possibly have the fixed point of at $\mathbf{0}$.

***Figure 3.7**. One 2 dimensional local translation showing the area of effect with the blue hue. The blue arrow is the parameter **v** of the local translation. The position of the area is defined by the parameter **p** of the local translation and the size of it is defined by ρ.*



It should be possible to pose the learning problem in the same way as for SEDS in definition 3.15, but using the cosine similarity measure and constricting the case by the invertibility criteria. The idea is formalized in following remark.

Remark 3.51 (Using cosine similarity to find diffeomorphic DS).

$$\underset{\rho_{1,\dots,K},\boldsymbol{p}_{1,\dots,K},\boldsymbol{v}_{1,\dots,K}}{\textbf{minimize}} \; J(\rho_{1,\dots,K},\boldsymbol{p}_{1,\dots,K},\boldsymbol{v}_{1,\dots,K}) = -\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\frac{\dot{\boldsymbol{x}}_{n,t}\cdot f(\boldsymbol{x}_{n,t})}{\|\dot{\boldsymbol{x}}_{n,t}\|\|f(\boldsymbol{x}_{n,t})\|}, \tag{3.52}$$

with subject to

$$\rho_k\|\boldsymbol{v}_k\| < \frac{e^{\frac{1}{2}}}{\sqrt{(2)}}, \quad k = 1,\dots,K, \tag{3.53}$$

$$f(\boldsymbol{0}) = \boldsymbol{0}. \tag{3.54}$$

3.3 LMDS

LMDS stands for local modulation of dynamical systems. It differs from the previous methods in that it does not create a dynamical system itself, but modifies a pre-existing one. As stated in the article introducing it, [14], LMDS implements incremental learning, improving a learned model while preserving some stability.

3.3.1 Local modulations and stability

The locally modulated dynamical system means that for every state $\mathbf{x} \in \mathbb{R}^d$ there exists a fully ranked linear mapping $M(\mathbf{x}) \in \mathbb{R}^{d \times d}$, which is the identity map outside a certain region in the state space. This region should not include the equilibrium $\mathbf{x}^*$, otherwise the stability of the DS is no longer guaranteed. The strongest proof for stability guarantees local asymptotic stability under certain criteria. [14]

The method can include learning the speed profile of the demonstrations, but in this thesis that part of learning is omitted and instead focuses on learning the direction of the modulation matrix. In this approach the modulation matrix is reduced to a rotation matrix. Additionally, this thesis focuses mainly on 3-dimensional DS, and as such the rotations considered are the rotations of $\mathbb{R}^3$.

3.3.2 Learning the local modulations

The LMDS learning has two stages. In the first stage, the modulation data from the original demonstration data is calculated, while the second involves the application of a suitable local regression method to learn the function

$$\xi : \mathbb{R}^d \mapsto \mathbb{R}^s. \tag{3.55}$$

It takes in a position and gives out an s-dimensional modulation vector $\theta \in \mathbb{R}^s$. The modulation vector in the 3-dimensional case is

$$\theta = \begin{pmatrix} \mathbf{v} \\ \gamma \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \gamma \end{pmatrix}. \tag{3.56}$$

Although it is stated that the learning of this method is achievable by any local regression method, the original article used gaussian process regression [26] for the learning [14]. Hence, the actual learning part is then reduced to applying the gaussian process regression to each entry of the modulation vector θ. The chosen kernel function is the squared exponential.

For a 3-dimensional case, the modulation vector can be thought to include the axis of rotation $\mathbf{v}$ and the amount of rotation γ. The following 2 equations (3.58, 3.59) describe how to calculate the modulation vector from each data point $(\mathbf{x}_{n,t}, \dot{\mathbf{x}}_{n,t})$.

Remark 3.57 (Calculating the modulation data for 3 dimensions). *Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ be a pre-existing DS. For each non-zero datapoint $(\boldsymbol{x}_{n,t}, \dot{\boldsymbol{x}}_{n,t})$ calculate the axis of rotation*

$$\boldsymbol{v}_{n,t} = \frac{f(\boldsymbol{x}_{n,t}) \times \dot{\boldsymbol{x}}_{n,t}}{\|f(\boldsymbol{x}_{n,t}) \times \dot{\boldsymbol{x}}_{n,t}\|}, \tag{3.58}$$

and the amount of rotation

$$\gamma_{n,t} = \arccos\Big(\frac{f(\boldsymbol{x}_{n,t}) \cdot \dot{\boldsymbol{x}}_{n,t}}{\|f(\boldsymbol{x}_{n,t})\|\|\dot{\boldsymbol{x}}_{n,t}\|}\Big). \tag{3.59}$$

After the data has been transformed into modulation data, the gaussian process regression can be applied for each modulation variable. In this thesis, the gaussian process regression and prediction was carried out with commercial software.

Remark 3.60 (Learning and using the local modulation in 3 dimensions). *For each element in θ treat it as a single gaussian process and let GP_predict stand for gaussian process prediction. Then apply gaussian process regression to each element in order to find the prediction the function*

$$\xi(\boldsymbol{x}) = \begin{pmatrix} GP\_predict\_v_1(\boldsymbol{x}) \\ GP\_predict\_v_2(\boldsymbol{x}) \\ GP\_predict\_v_3(\boldsymbol{x}) \\ GP\_predict\_\gamma(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} v_1(\boldsymbol{x}) \\ v_2(\boldsymbol{x}) \\ v_3(\boldsymbol{x}) \\ \gamma(\boldsymbol{x}) \end{pmatrix}. \tag{3.61}$$

Then calculate the modulation by using the exponential form of Rodrigues' rotation formula:

$$R(\xi(\boldsymbol{x})) = expm\Bigg(\gamma \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}\Bigg). \tag{3.62}$$

Finally, the modulation of the DS can then be simply achieved by:

$$\dot{\boldsymbol{x}} = g(\boldsymbol{x}) = R(\xi(\boldsymbol{x}))f(\boldsymbol{x}). \tag{3.63}$$

3.3.3 Data selection for LMDS

Since gaussian process regression is a nonparametric method, the data used for training is actually part of the model. This imposes practical limits on how much training data can be utilised when using gaussian processes in LMDS. This question of choosing the appropriate data set for LMDS is the main challenge in its application for real systems. The original paper [14] discusses sparsity and introduces a heuristic method to incrementally build a training set from a larger body of data. In this thesis, however, a simpler heuristic was used instead. In this simpler heuristic, an average trajectory was calculated from the demonstration data set and a fixed number of points from that average trajectory were chosen to be included in the training set.

One additional concern when choosing the data points is to consider the region of influence for the local modulations. The region of influence is that part of the state space where the modulation differs substantially from the identity map. This region is defined by the gaussian process hyperparameters and the data points chosen for it. It is not immediately obvious how these choices modify the region of influence. This is important, because as discussed earlier, it has an immediate effect on the stability of the resulting modulated DS.

3.4 Other dynamical system based methods

There are many ways for an agent to learn movement from demonstrations but this section deals with only a handful of these methods. One very popular approach is based on Markov decision processes (MDP), in which all actions and states have a value or cost attached to them. This value or cost function encodes the behavior the agent is supposed to realize. These approaches are often called reinforcement learning or inverse reinforcement learning, but this thesis does not deal with them.

Other stable DS-based methods can be divided into autonomous DS or non-autonomous dynamical systems. Since the SEDS has a sound theoretical basis, there is ongoing work to mitigate its limitations. Since the main limitation of SEDS stems from the form of Lyapunov-function used to ensure the stability, these methods often focus on modifying the Lyapunov-function to allow the DS more freedom to reproduce demonstrations.

One example was introduced in [12], in which the DS and Lyapunov-function candidate are learned separately. The DS can be learned with any relevant method, meaning it can be unstable. The stability is then forced by online-calculations based on the Lyapunov function, which is learned from the demonstrations. The enforcement happens by adding the stabilizing command $u(\mathbf{x})$ to the original DS $\dot{\mathbf{x}} = f(\mathbf{x})$ resulting in a stable DS $\dot{\mathbf{x}} = f(\mathbf{x}) + u(\mathbf{x})$.

Another example is the method called τ-SEDS, which was introduced in [17]. It alleviates the restrictive quadratic Lyapunov-function by the following steps. First, a Lyapunov-function is learned from the data. Secondly, a diffeomorphism is calculated which transforms the learned Lyapunov-function into a quadratic Lyapunov-function. Finally, this diffeomorphism is used to transform the data so that it is compatible with a quadratic Lyapunov-function, which means that the SEDS can be used. After the DS has been found by SEDS, the diffeomorphism can be applied to the resulting DS which then reproduces the demonstration better than the original SEDS.

One very popular approach based on non-autonomous dynamical systems is called dynamical motion primitives (DMP)[23], [8], [20], [25], [24]. The dynamical systems used by DMPs are a combination of nonlinear and stable linear systems. The DS first follows nonlinear dynamics but is then smoothly switched to stable linear dynamics by using a phasing variable. This phasing variable is essentially a dynamical system of its own and during the execution of a DMP it is the part that makes the DMP time-dependent.

The argument for preferring autonomous methods to non-autonomous ones is that they are more robust against spatial and temporal disturbances [10] [17].

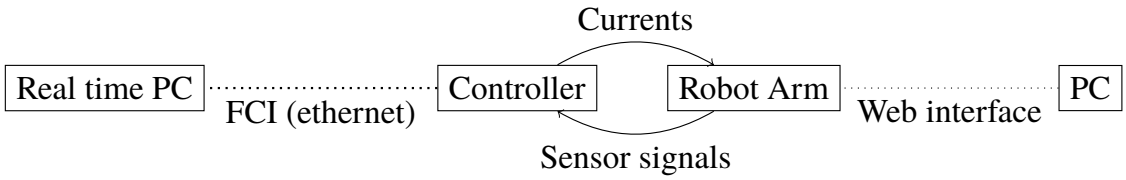
4. TESTING ALGORITHMS ON A ROBOTIC ARM SYSTEM

4.1 Physical equipment and set-up

The aim of the tests was to explore the applicability of a DS-based approach to learning motion on a proof-of-concept level. The methods were tested on a 7-DOF manipulator arm made by Franka Emika GmbH. Figure 4.1 is a conceptual depiction of the physical system.

The physical set-up consists of a real time PC, the arm controller, the arm and a PC. The real-time PC is connected to the controller via an Ethernet cable, while the arm and the controller are connected through a customised cable. Presumably, this customised cable transfers power to the arm's motors and transmits information about the status of the arm's joints to the controller. The robot arm also has an Ethernet interface for connection to a PC. This Ethernet connection allows the use of a web-interface. Among its many purposes, this web-interface is used to set up the robot arm's internal networking parameters and its physical parameters, and to unlock its joints.

Figure 4.1. *Systems physical setup. A block represents a single physical item and dashed line an Ethernet connection.*



4.2 Software set-up for reproducing learned motions

Most of the functionality is defined by the software on the real-time PC. This PC has a real-time Linux [27] installed in it, as required by the franka control interface, FCI [6]. The client part of the FCI is implemented as a C++ library, known as libfranka in the real-time Linux. Through the FCI, the real time PC can establish a motion generator for the robot arm. As described in the FCI documentation [6], a motion generator means a functionality that gives out movement-related commands for the system to follow, such as points or velocities. These commands are in joint coordinates or in Cartesian coordinates. In this thesis, the movement generator is the DS learned from the demonstrations.

In addition to the FCI, a popular middleware library known as ROS, short for Robot Operating System [18], was used. ROS provides many helpful functionalities for robotics,

such as communication between running programs. The FCI provides an interface for the ROS known as franka_ros.

The franka_ros includes a group of examples written in C++ demonstrating the usage of its components. The most immediately useful subset of examples is called franka_example_controllers [7]. This provided a template for a cartesian space motion generator in the form of a ROS controller. ROS control is a ROS package for implementing real-time control loops in a ROS node [1]. There is a useful diagram in figure 4.2 from the original article introducing ros_control that illustrates clearly how a ROS controller works.

In terms of figure 4.2, the implemented DS was part of a "Controller", indicated by a yellow box in the figure. As can be seen, the "Controller" is part of "Controller Manager" which is a ROS-node. The "Controller Manager" runs inside the real-time Linux and can communicate with other ROS-nodes via ROS. Figure 4.2 illustrates the hierarchy.

The initial implementation and investigation of the methods was done in MATLAB® [1]. Since the example ROS controllers were written in C++, a MATLAB toolbox called MATLAB Coder™ was used. The toolbox can convert functions written in MATLAB to C/C++ with some restrictions. A MATLAB-function was first written for each form of the dynamical systems (SEDS, LMDS, diffeomorphic DS) and then converted to a C++ source code.

After this conversion, a simple wrapper class was written for each DS to be used in a modified form of cartesian_velocity_example_controller from franka_example_controllers. The SEDS and diffeomorphic DS take in their parameters from an external .mat file, while the LMDS uses a MATLAB Statistics and Machine Learning Toolbox™ for the gaussian process prediction. This forces the generated C++ code to contain all the parameters in it as code. This stems from the fact that in the initial implementation of LMDS for this thesis, the part used for learning the gaussian processes was a Statistics and Machine Learning Toolbox-function[2]. Since it had already provided satisfactory results there was no need to write custom code.

4.3 Software set-up for learning the motions

As mentioned in the earlier section, the learning methods were implemented in MATLAB. The teaching data was gathered using an example controller from the franka_example_controllers, ROS and a MATLAB toolbox called Robotics Systems Toolbox™. The modified example controller recorded the time, the robot joint values and published these values in ROS topics. A Simulink® [1] node was used to listen to these topics and record the values in MATLAB workspace.

[1]MATLAB and Simulink are registered trademarks of Mathworks, Inc

[2]URL: https://se.mathworks.com/help/stats/fitrgp.html , accessed 2018-10-29.

Figure 4.2. *Overview of ROS Control software architecture (from [1]). It shows how the controller is a part of Controller Manager node and how the controller interfaces the physical equipment.*

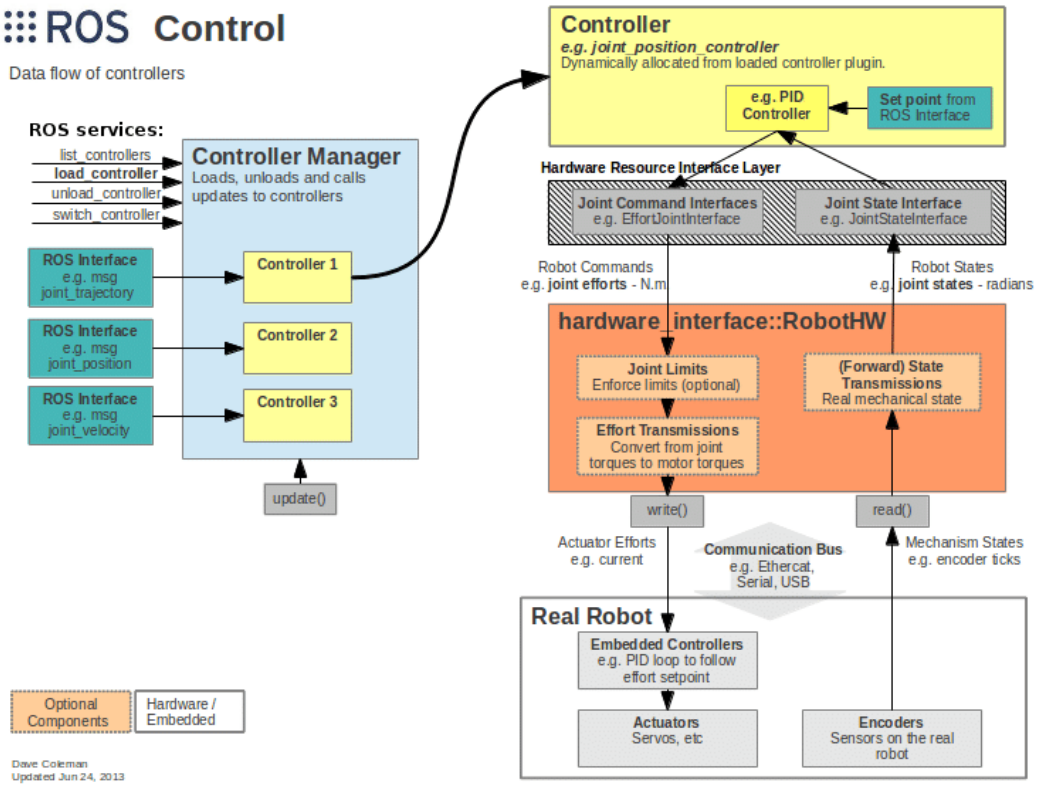
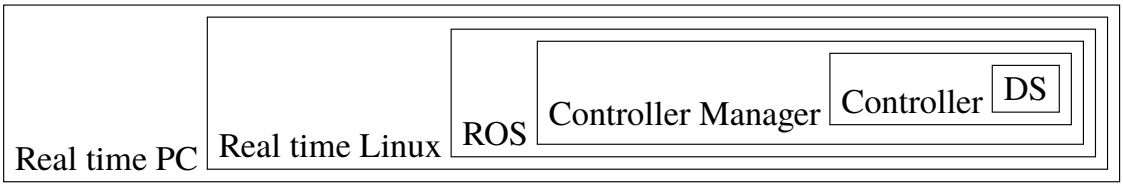


Figure 4.3. *Conceptual diagram showing the software hierarchy*



4.4 Motion reproduction limitations imposed by the set-up

The FCI documentation explains that the command signal given to the arm has to be sufficiently smooth. Essentially, this means that the forces resulting from the velocity signals should not exceed the limits dictated by the FCI. If there is a violation of this the robot arm stops all movement and goes into a reflex error mode which has to be manually resolved. The FCI provides a software implementation of a signal rate limiter and a low-pass filter, but recommends that the end-user should nevertheless provide a sufficiently smooth signal without relying on the rate limiter and filter.

Assuming that the signal adheres to the limitations, there are 2 additional sources for signal non-smoothness. The first comes from the fact that the control loop might not execute in

time, although this happens very rarely. However, the second and much more important source for error arises from the fact that the physical controller of the arm and the real-time PC communication is UDP[22]-based. Since UDP is not deterministic, there is always some packet loss when it is used. This is a major drawback for a real time application such as movement control. The rate of packet loss is dependent on the physical set-up of the network, so the FCI must be provided sufficient networking capability in order to use it.

This is the reason for the separation of the speed and shape discussed in section 2.5. When trying to run the demonstrations at the original speed, the networking was a major obstacle. Often the test run could not be run at the original speed without the system stopping prematurely. The stops come from internal safety implementations that prevent the arm from excessive torque commands. Because the signal was not smooth enough, the velocity commands resulted in excessive angular jerks and accelerations.

A solution for this could come from implementing the DS part so that it communicates with the velocity controller in a deterministic manner. A computer bus or something more sophisticated should remove the problem of inconsistent communication frequency.

It has to be noted that the system can also stop when reaching joint or workspace limits. This however, is not a limitation of the implementation but rather of the methods. The methods do not take any actuator limitations into account so it is possible that the resulting trajectories exceed the actuator's limits.

5. RESULTS AND CONCLUSIONS

5.1 Recorded data

A collection of 3 different 3-dimensional demonstration patterns was recorded to test the methods. The patterns depict an arbitrary movement resembling an equally arbitrary shape and do not necessarily model any actual work-related movements, but it might be helpful to interpret them as pick-and-place sort of movements. Figures 5.1, 5.2, 5.3 show the recorded patterns. The demonstration pattern 1 shows a movement that starts from the left side, negative y-axis, and ends on the right side. Demonstration pattern 2 shows a movement that starts from the lower left corner and ends at the upper right corner. Finally the demonstration pattern 3 shows a movement starting from the upper left corner and ending at the lower center.

The data was collected by implementing a Simulink-node listening to a particular topic in ROS and saving the results in the MATLAB-workspace. The recording was done in the joint space meaning that 7 joint values for each time instant were recorded with the time. The conversion to Cartesian space without the pose was done by using the modified DH-parameters([5]) provided in the FCI documentation.

Figure 5.1. *Demonstration pattern 1. The trajectories start from the lower left-hand side*

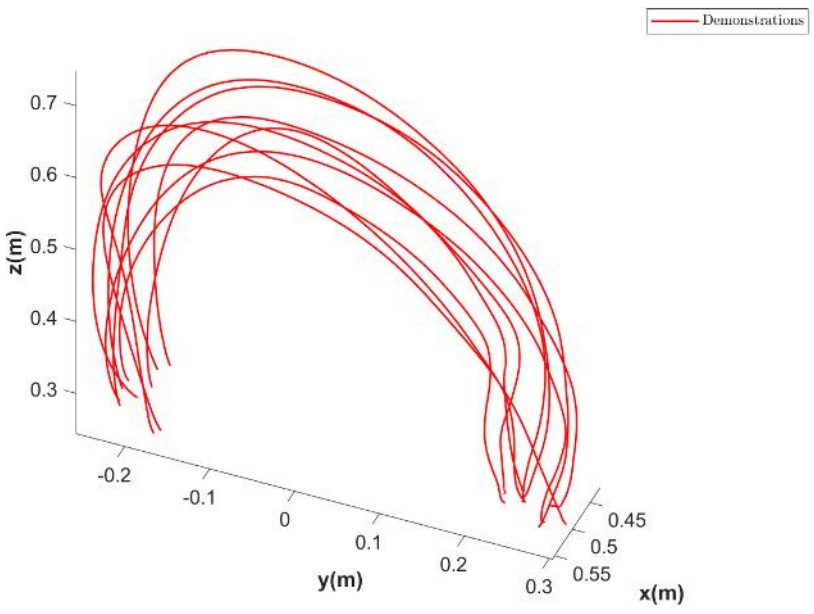


Figure 5.2. *Demonstration pattern 2. The trajectories start from the left hand side*

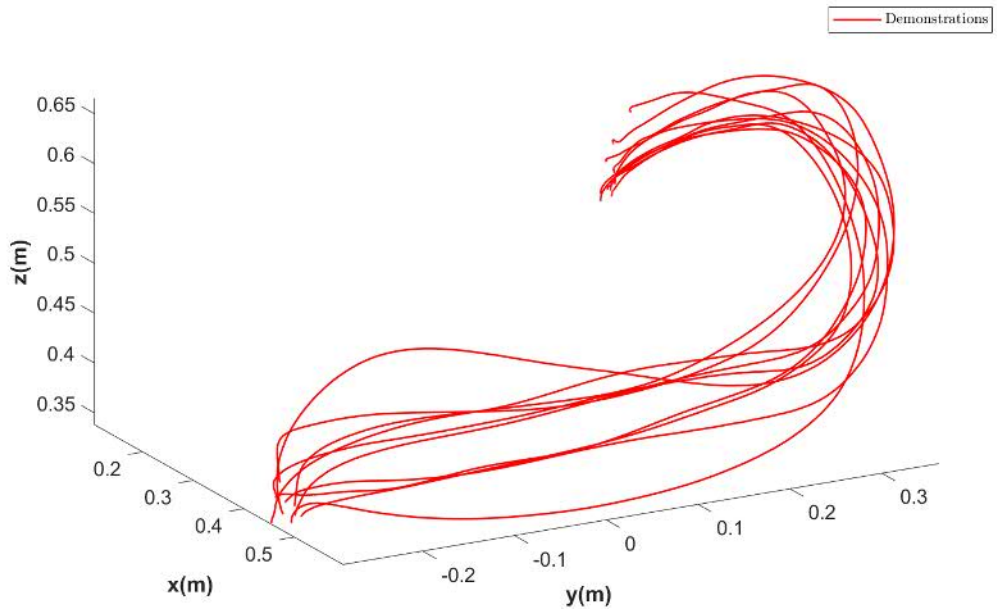
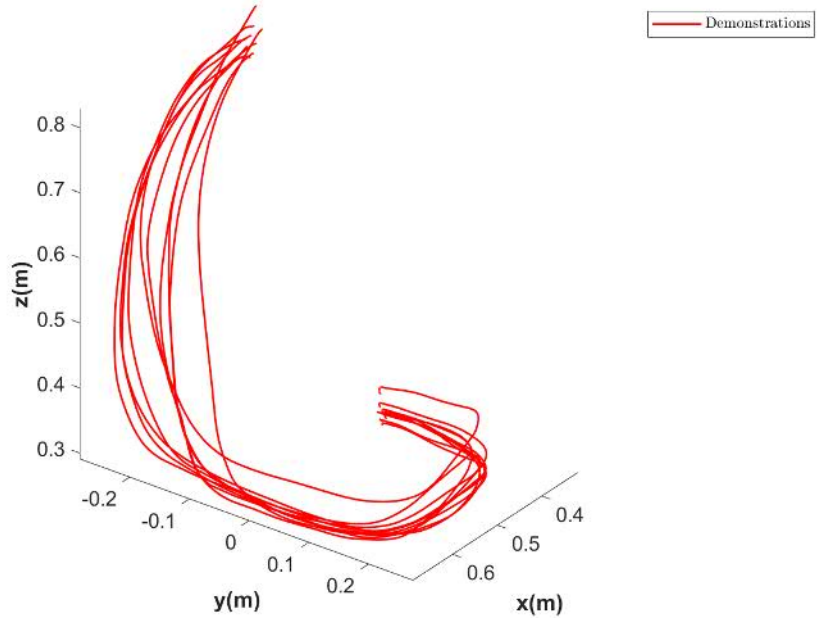


Figure 5.3. *Demonstration pattern 3. The trajectories start from the upper left-hand side*



5.2 Reproduction results

Reproduction was done in two ways. The first way was with the real robot arm and the second way was to solve the ODE:s with the same initial positions as in the demonstration data. The motivation for reproducing the demonstrations on a real robot was to try out the methods' applicability in a real-world situation. It highlighted the infrastructure needed for applying the methods. One such piece of infrastructure that is clearly needed is a velocity controller with smooth communication between the DS and the controller. The motivation for reproducing the motions by solving the ODE:s is that it is more convenient not to need

the physical set-up required for running the motions. This allows a faster and perhaps more general way of comparing the methods.

In the real-robot reproductions, 5 different trajectories were recorded for each method and demonstration pattern. These reproductions are intended for visualization and subjective evaluation of the methods' efficacy. One immediate effect of the model-free approach is that some reproductions end prematurely due to exceeding the movement limit or exceeding the reach of the arm. This can be seen in figures 5.4, 5.8 and 5.11.

Although these reproductions can also shed some light on the question of a method's robustness and generalization, these qualities are not discussed in this thesis and have not yet been formalized in the context of DS-based learning. However, there are some indications as to how the methods compare with each other in that the initial positions varied greatly from those in the demonstrations. Figures 5.5 and 5.6 show the significant differences in this regard.

The second way of reproduction will not be visualized, but will act as a basis for calculating similarity measures with the algorithms 1 and 2 introduced in section 2.4. Table 5.1 shows the numbers calculated with algorithm 1 and adding the results together for each demonstration in a pattern. These results indicate that the most accurate method in this regard was LMDS.

Table 5.1. *Cumulative error measure (Higher number means less similar)*

	SEDS	LMDS	Diffeomorphic DS
Demonstration pattern 1	74.59	35.11	78.61
Demonstration pattern 2	94.76	63.49	37.26
Demonstration pattern 3	207.0	40.63	46.25
average	125.4	46.41	53.71

Since the previous table, 5.1, shows a measure of dissimilarity rather than similarity, it should be transformed into a measure of similarity comparable to results given by the algorithm 2. Assuming that the $Error \geq 0$, this can simply be achieved by:

$$Similarity = \frac{100}{1 + Error}. \tag{5.1}$$

In order to make table 5.1 more comparable to table 5.3, the latter is transformed with the previous equation, except for the averages.

Now the methods can be compared with regard to both similarity measures. Judging from the result, it can be seen that LMDS provides better cumulative similarity while the diffeomorphic DS gives better reproduction in terms of velocity direction.

It should be noted however, that the LMDS was SEDS-based and the sparse data selection was based on a very simple heuristic. Also, the diffeomorphism for the diffeomorphic DS was gained by the original author's fast heuristic algorithm. There is still much room

Table 5.2. *Transformed cumulative similarity measure of table 5.1 (Higher number means more similar)*

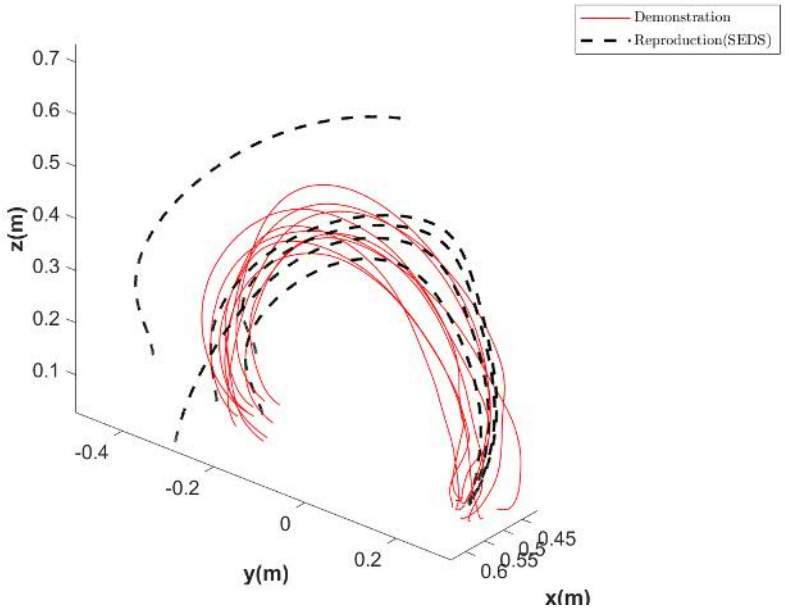
	SEDS	LMDS	Diffeomorphic DS
Demonstration pattern 1	1.32	2.77	1.26
Demonstration pattern 2	1.04	1.55	2.61
Demonstration pattern 3	0.48	2.40	2.16
average	0.95	2.24	2.01

Table 5.3. *Normalized cosine similarity results (Higher number means more similar)*

	SEDS	LMDS	Diffeomorphic DS
Demonstration pattern 1	0.9124	0.9655	0.9612
Demonstration pattern 2	0.8636	0.9588	0.9855
Demonstration pattern 3	0.8833	0.9715	0.9753
average	0.8865	0.9653	0.9740

for improvement for both of these methods, and if restricted to the three methods chosen, probably the best result would be gained from applying the LMDS with good sparsity criteria to a diffeomorphic DS gained from a more general optimization. Finally, it should be remembered that so far the methods do not take actuator limitations into account.

Figure 5.4. *SEDS reproductions for demonstration pattern 1 on Franka Panda. The demonstrations and reproductions begin from the left-hand side. The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*



5.3 Conclusion

In table 5.4 the comparison is made subjectively by giving each method a value between 1 and 5 indicated with asterisks. The methods were compared in 5 aspects: implementation difficulty, runtime complexity, reproduction accuracy, scalability and stability. What these

Figure 5.5. *LMDS reproductions for demonstration pattern 1 on Franka Panda. Demonstrations and reproductions begin from left hand side. The demonstration curves are the red continuous lines and reproductions are the black dashed lines.*

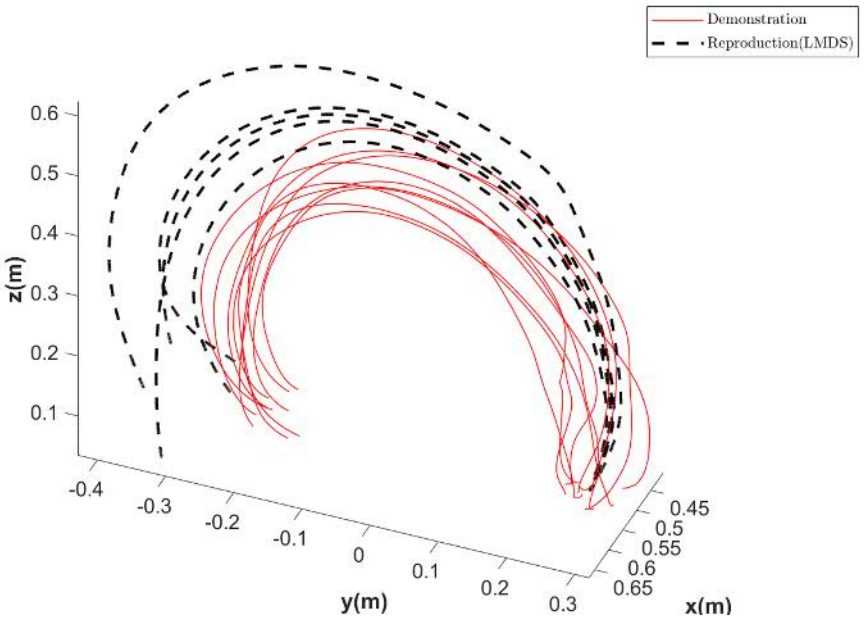
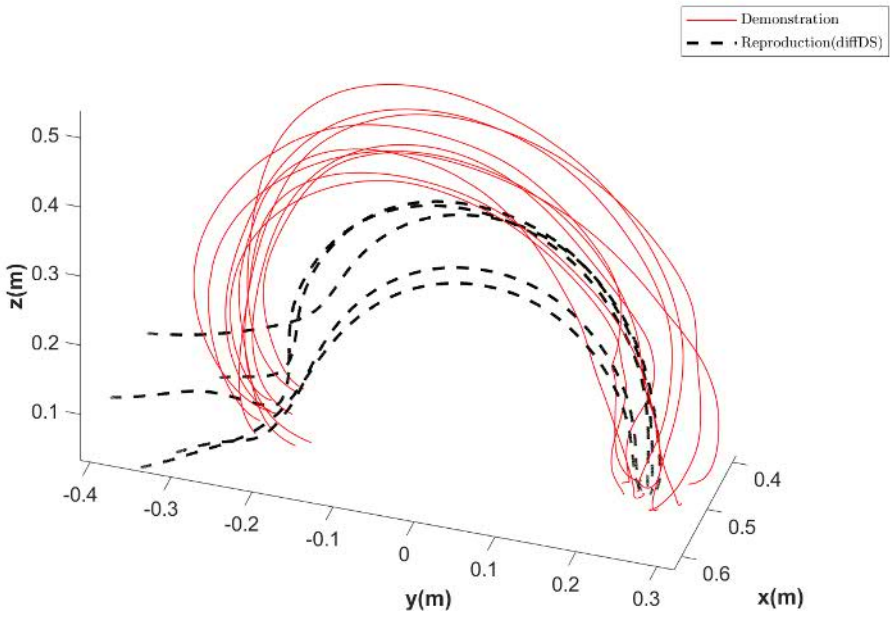


Figure 5.6. *Diffeomorphic DS reproductions for demonstration pattern 1 on Franka Panda. The demonstrations and reproductions begin from the left-hand side. The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*



aspects mean and the reasoning behind the scores given for them is explained below. The underlying assumption for the diffeomorphic DS is that it uses the heuristic algorithm to gain it as introduced by the authors of [21].

Implementation difficulty is the subjective assessment of the effort needed to gain a dynamical system function with the specified method.The diffeomorphic DS is given the

Figure 5.7. *SEDS reproductions for demonstration pattern 2 on Franka Panda. The demonstrations and reproductions begin from the lower left-hand side. The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*

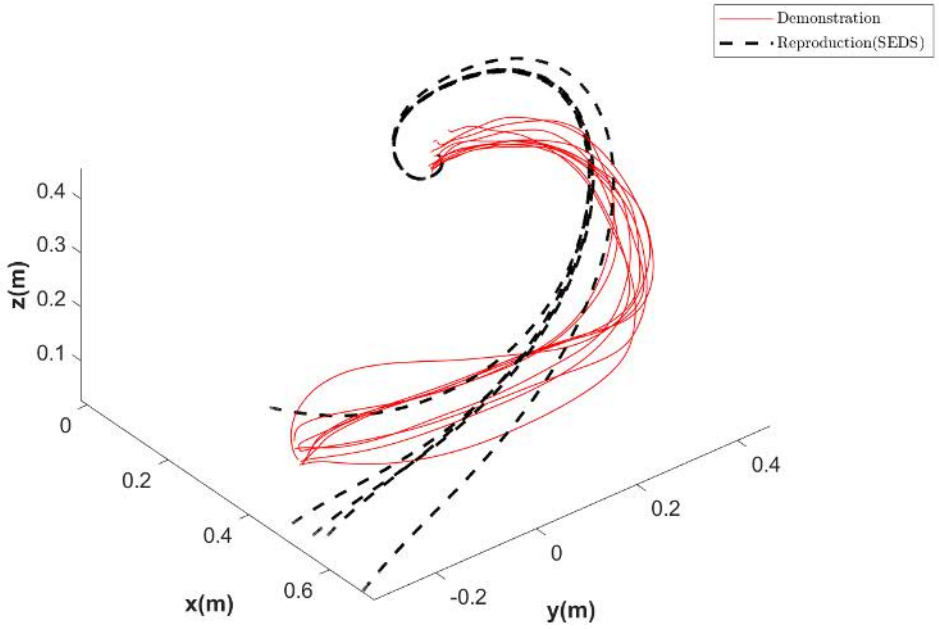
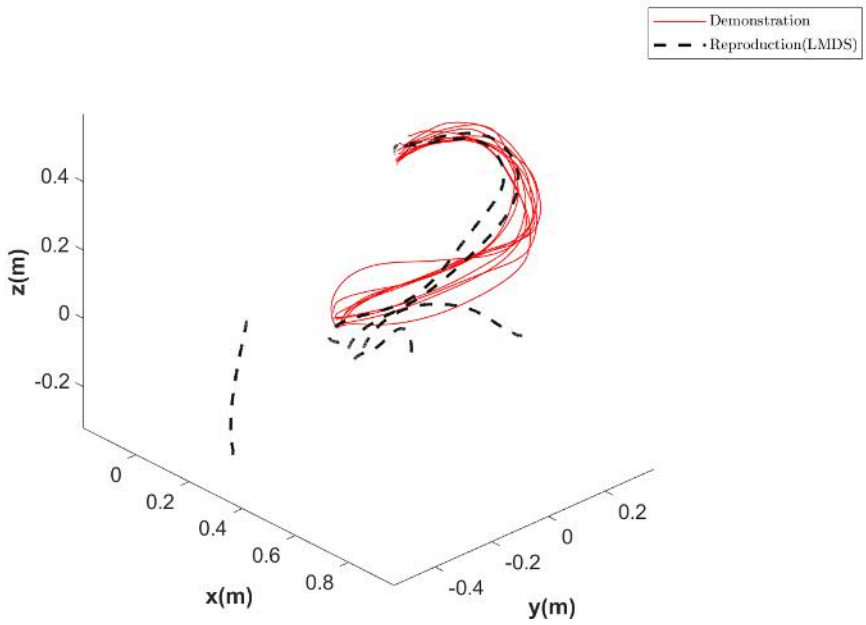


Figure 5.8. *LMDS reproductions for demonstration pattern 2 on Franka Panda. The demonstrations and reproductions begin from the lower left-hand side. The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*



lowest value as it is relatively straightforward to implement. The high value for LMDS reflects the fact that it always requires a pre-existing DS in order to use it. Also, much of its implementation complexity can be put down to the implementation of gaussian process regression.

Runtime complexity could be analyzed more rigorously, but for practical purposes this

Figure 5.9. *Diffeomorphic DS reproductions for demonstration pattern 2 on Franka Panda. The demonstrations and reproductions begin from the lower left-hand side. The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*

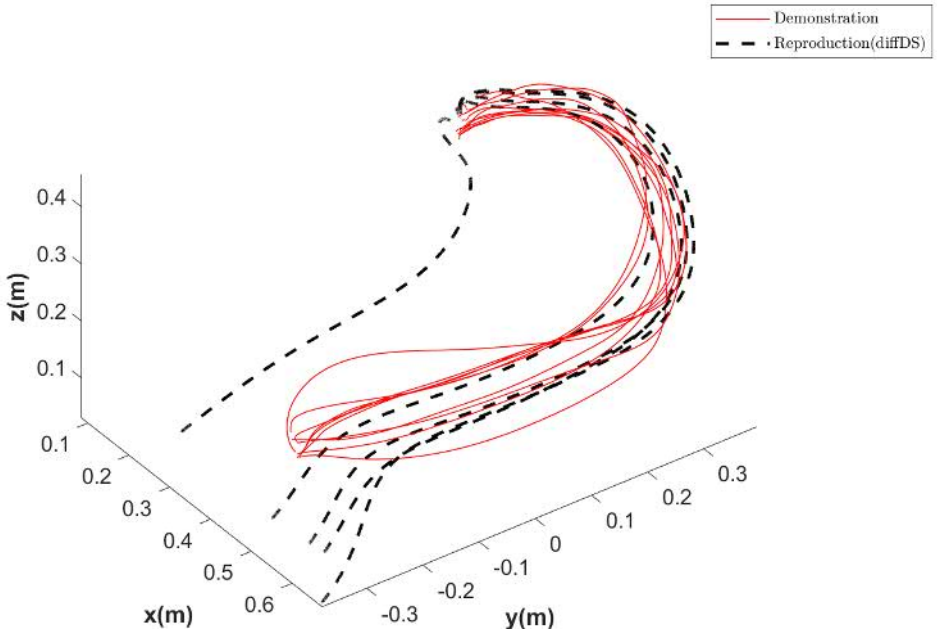
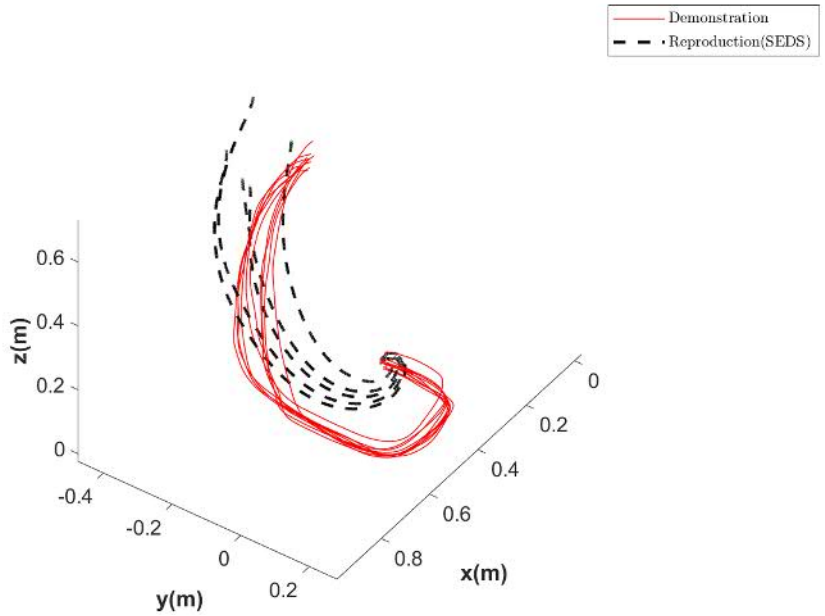


Figure 5.10. *SEDS reproductions for demonstration pattern 3 on Franka Panda. The demonstrations and reproductions begin from the upper left-hand side.The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*



simple value is assumed to be sufficient. Also the runtime complexity is highly correlated with the scalability that is discussed later in this chapter. For this criterion, SEDS is given the lowest value as evaluating the dynamical system function mainly requires only elementary functions and the inversion of fixed-size matrices. The diffeomorphic DS is

Figure 5.11. *LMDS reproductions for demonstration pattern 3 on Franka Panda. The demonstrations and reproductions begin from the upper left-hand side.The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*

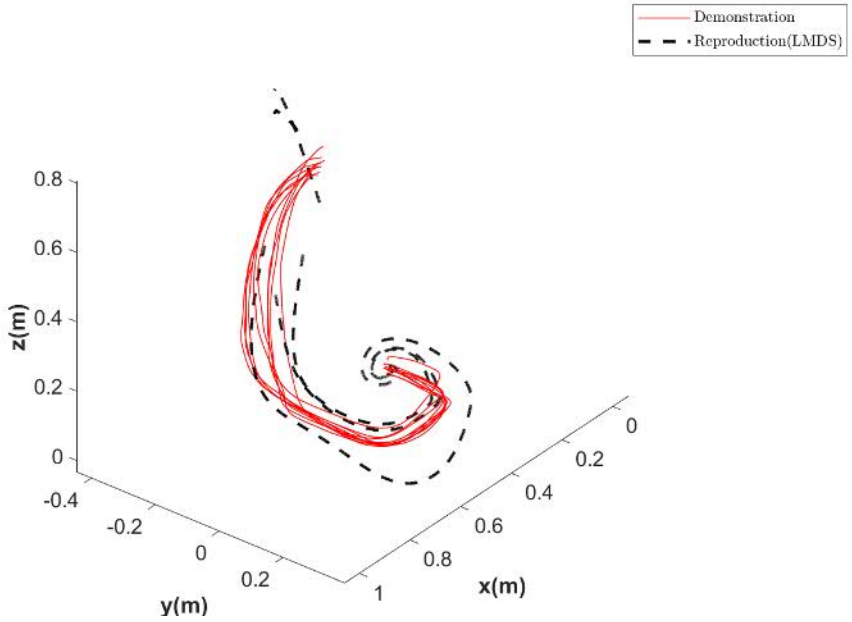
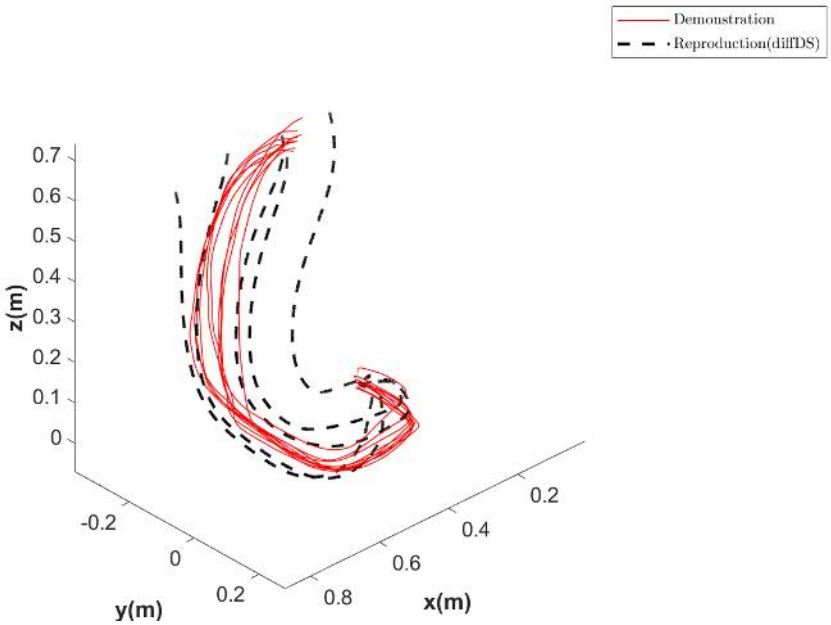


Figure 5.12. *Diffeomorphic DS reproductions for demonstration pattern 3 on Franka Panda. The demonstrations and the reproductions begin from the upper left-hand side.The demonstration curves are the red continuous lines and the reproductions are the black dashed lines.*



next in this category, because the evaluation of inverse diffeomorphism requires an iterative method. Finally, the LMDS is deemed most complex at runtime because again it requires a pre-existing DS and uses the gaussian process prediction for the evaluation.

Reproduction accuracy is based on the similarity results shown above, taking the limited

***Table 5.4**. Subjective comparison of the methods. The amount of asterisks is directly proportional to the quality mentioned in the columns. For example, more asterisks in computational complexity means it is more complex (and as such, slower) to run.*

	Implementation difficulty	Runtime complexity	Reproduction accuracy	Scalability	Stability
SEDS	***	*	**	***	*****
LMDS	*****	*****	*****	*	***
Diffeomorphic DS	**	***	****	****	****

form of the implementation into account. In this category, LMDS has most potential because it can further improve on the already very good result. SEDS is deemed lowest here, because of its fundamental limitation arising from the Lyapunov-function.

Scalability is the subjective assessment of the effort required to apply the method for higher than 3-dimensional cases. The diffeomorphic DS with its heuristic way of training is easily scalable for almost any dimension. If the training is done with a more general optimization, it would still be less complex, or at worst comparable to, SEDS as an optimization problem. The LMDS is based mostly on rotations, so its generalization for higher than 3 dimensions is possible, but requires further analysis.

The stability is also a subjective comparison. The SEDS and diffeomorphic DS have both been proved to be asymptotically and globally stable. The differences in their scores come from the details of the proofs shown in the articles in which they were introduced. The SEDS proof is easily followable and can stand very rigorous examination but the diffeomorphic DS stability proof is slightly more obscure. The LMDS has the weakest notion of stability, mostly because choosing the data points to be included in the gaussian processes can affect the stability negatively.

Finally, using the DS places high demands on the computational and networking equipment. Because of this it is highly recommended that the velocity commands should be transmitted in a deterministic manner, such as via a computer bus. The UDP-based communication is simply not sufficient for highly dynamic applications.

REFERENCES

[1] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, E. Fernandez Perdomo, ros_control: A generic and simple control framework for ROS, The Journal of Open Source Software, Vol. 2, Iss. 20, Dec. 2017, pp. 456 – 456.

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, May 2005.

[3] W. Chung, L.C. Fu, S.H. Hsu, Motion Control, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 133–159. Available: https://doi.org/10.1007/978-3-540-30301-5_7

[4] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, Journal of Artificial Intelligence Research, Vol. 4, Iss. 1, Mar. 1996, pp. 129–145.

[5] J. Craig, Introduction to Robotics: Mechanics and Control, Pearson/Prentice Hall, Addison-Wesley series in electrical and computer engineering: control engineering, 2005. Available: https://books.google.fi/books?id=MqMeAQAAIAAJ

[6] Franka Emika GmbH, Franka control interface documentation. Available (accessed on 19.10.2018): https://frankaemika.github.io/docs/index.html

[7] Franka Emika GmbH, franka_example_controllers source code. Available (accessed on 29.10.2018): https://github.com/frankaemika/franka_ros

[8] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: Learning attractor models for motor behaviors, Neural Computation, Vol. 25, Iss. 2, PMID: 23148415, 2013, pp. 328–373.

[9] H. Khalil, Nonlinear Systems, Prentice Hall, Pearson Education, 2002. Available: https://books.google.fi/books?id=t_d1QgAACAAJ

[10] S. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with gaussian mixture models, IEEE Transactions on Robotics, Vol. 27, Iss. 5, 2011, pp. 943 – 957.

[11] S.M. Khansari-Zadeh, LASA Handwriting Dataset, version 2.0, https://bitbucket.org/khansari/lasahandwritingdataset, 2010. [Online; accessed 15-November-2017].

[12] S.M. Khansari-Zadeh, A. Billard, Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions, Robotics and Autonomous Systems, Vol. 62, Iss. 6, 2014, pp. 752 – 765.

[13] S. Khansari-Zadeh, A. Billard, The derivatives of the SEDS optimization cost function and constraints with respect to its optimization parameters, Learning algorithms and systems laboratory, École Polytechnique Fédérale de Lausanne, Techn. rep., 2011.

[14] K. Kronander, M. Khansari, A. Billard, Incremental motion learning with locally modulated dynamical systems, Robot. Auton. Syst., Vol. 70, Iss. C, Aug. 2015, pp. 52–62.

[15] J. Lee, Introduction to Smooth Manifolds, Springer, Graduate Texts in Mathematics, 2003. Available: https://books.google.fi/books?id=eqfgZtjQceYC

[16] A.N. Michel, L. Hou, D. Liu, Fundamental Theory: The Principal Stability and Boundedness Results on Metric Spaces, Birkhäuser Boston, Boston, MA, 2008, pp. 71–148. Available: https://doi.org/10.1007/978-0-8176-4649-3_3

[17] K. Neumann, J.J. Steil, Learning robot motions with stable dynamical systems under diffeomorphic transformations, Robotics and Autonomous Systems, Vol. 70, 2015, pp. 1 – 15.

[18] Open Source Robotics Foundation. Available (accessed on 27.10.2018): http://www.ros.org/

[19] T. Osa, J. Pajarinen, G. Neumann, J.A. Bagnell, P. Abbeel, J. Peters, An algorithmic perspective on imitation learning, Foundations and Trends® in Robotics, Vol. 7, Iss. 1-2, 2018, pp. 1–179.

[20] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: 2009 IEEE International Conference on Robotics and Automation, May, 2009, pp. 763–768.

[21] N. Perrin, P. Schlehuber-Caissier, Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems, Systems & Control Letters, Vol. 96, 2016, pp. 51 – 59.

[22] J.B. Postel, Rfc 768: User datagram protocol, Internet Engineering Task Force, 1980. DOI: 10.17487/RFC0768.

[23] S. Schaal, Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics, Springer Tokyo, Tokyo, 2006, pp. 261–280. Available: https://doi.org/10.1007/4-431-31381-8_23

[24] S. Schaal, S. Kotosaka, D. Sternad, Nonlinear dynamical systems as movement primitives, in: Humanoids2000, First IEEE-RAS International Conference on Humanoid Robots, Cambridge, MA, Sept. 2000, CD-Proceedings. clmc.

[25] S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert, Learning movement primitives, in: Dario, P., Chatila, R. (eds.), Robotics Research. The Eleventh International Symposium, Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 561–572.

[26] M. Seeger, Gaussian processes for machine learning, International Journal of Neural Systems, Vol. 14, 2004, pp. 69–106.

[27] The Linux Foundation, Real time linux documentation. Available (accessed on 27.10.2018): https://wiki.linuxfoundation.org/realtime/documentation/start