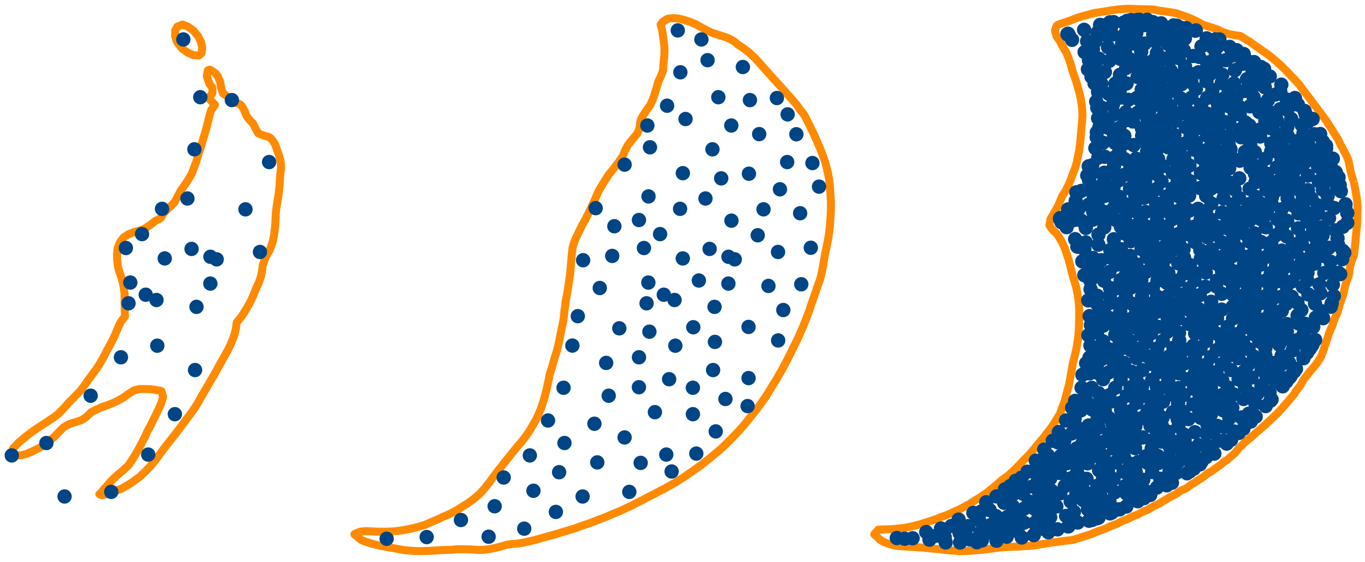
Master Thesis

A Machine Learning Approach to Control Musculoskeletal Systems



Submitted at the University of Stuttgart

by

Danny Drieß

March 2019

University of Stuttgart
Machine Learning and Robotics Lab
Institute for Parallel and Distributed Systems

A Machine Learning Approach to Control Musculoskeletal Systems

by

Danny Drieß

A thesis submitted in fulfillment of the requirements for the degree

Master of Science

in the course

Simulation Technology

First Examiner
Prof. Dr. Marc Toussaint
Machine Learning & Robotics Lab
University of Stuttgart

Second Examiner
Prof. Dr. Syn Schmitt
Biomechanics & Biorobotics Group
University of Stuttgart

Author: Danny Drieß from Tübingen
SimTech-ID: 56
Submission: March 2019

Contents

1. Introduction

One of the major current scientific endeavors is to understand and especially being able to synthesize what one would call intelligent behavior. Most species in our world that exhibit intelligent behavior have in common that they *move*. The ability to generate motion in our environment requires many different aspects of research in artificial intelligence, machine learning and robotics at the same time. First, the world and its properties have to be perceived via suitable sensors to gain an understanding of the world, including the state of the intelligent entity itself. In order to follow own objectives, decisions have to be taken, the necessary motions be planned and finally executed via suitable controllers. During the execution, adjustments to the plan in a feedback mechanism or complete replanning on a higher level are import to compensate for unforeseen events. A curious intelligent entity has to interact with the environment by trying things out, learning from failures and success to memorize experience for its own improvement. All this is implied by the need to generate motion. Therefore, a central path to understand intelligent behavior is to generate moving artificial systems that interact and manipulate our real environments.

As robotic research over many decades has shown, synthesizing movements for locomotion or manipulation of the environment is highly nontrivial. One could even say that the majority of robotic research deals in one way or another with the question how movements can be generated. In contrast, intelligent species found in nature seem to perform these tasks – perception, planning, control, movement, interaction with the environment – with ease. Furthermore, many animals (including humans) also show a high adaptability to new situations, implying their ability to learn from experience outside their predefined capabilities.

Since the motion apparatus and the nervous system of a biological organism has not evolved separately, the question arises whether the motion apparatus exhibit properties that are in some sense favorable for the nervous system to generate and learn motions in our environments.

When modeling the musculoskeletal system of vertebrates, it turns out that the dynamics from muscle stimulations to movements is highly complex [31], [17], [13], [3], [22]. Nonlinearities [17], delays [33], hidden states etc. make it difficult to design controllers for such systems with classical techniques. Since muscles can only actively produce forces in contraction direction, at least two muscles in an antagonistic setup are required to articulate one degree of freedom. Furthermore, musculoskeletal systems also include so-called biarticular muscles that drive more than one joint at the same time. Therefore, in addition to kinematic redundancy, musculoskeletal systems have a high actuator redundancy [42], further increasing the complexity and dimensionality of the system.

This redundant antagonistic setup together with the dynamical properties of the muscles has, however, also two important consequences. First of all, the passive and active elasticity of the muscle tendon units (MTUs) is favorable for interaction with the environment, since contact and impact forces are absorbed to a certain degree. Generating technical systems that are able to safely interact with uncertain environments through contacts are difficult to create [12], [11]. Secondly and more relevant for the present work, the antagonistic setup of the muscles with their characteristic force length/force velocity curves [17] imply a certain intrinsic stability [6] and small perturbations are compensated without the need of any control.

As shown in a previous work of the author of this document [13], this intrinsic stability is favorable for learning, since the learning algorithm does not have to learn a stabilizing controller first. Indeed, applying learning methodologies to real robotic systems is often problematic, since taking explorative actions during the learning process often lead to unstable and hence possibly dangerous system states. In contrast, as considered in [13], a musculoskeletal system can be controlled by applying constant muscle stimulations, which correspond to certain equilibrium states.

The goal of the present work is to learn a controller for a redundant musculoskeletal system that is able to predict muscle stimulations that lead to desired configurations of the system. Such a controller will be called an inverse model.

However, despite the advantages of a musculoskeletal system with respect to its intrinsic stability, the high actuator redundancy implies that there are infinitely many muscle stimulations leading to the same configuration in equilibrium. Therefore, direct learning of an inverse model with standard regression techniques is not possible or leads to undesirable effects. This non-uniqueness problem of learning inverse models has extensively been studied in robotics in the context of inverse kinematics, e.g. [14], [5], [40], [9], [24], [27]. However, an essential aspect missing in most existing approaches is to systematically and efficiently estimate the reachable set of the system, i.e. to know which parts of the workspace can be reached at all. The reachable set is essential for several reasons: First of all, an inverse model is well-defined only over the reachable set. In contrast, the function approximators used in most learning methods can be queried everywhere. Thus, a proper inverse model must represent where it even is reasonable to query it. Secondly, without knowing the domain of the inverse, it is not possible to define a suitable metric that measures the quality of an inverse model globally. In this respect, when defining an objective for learning an inverse model, points outside the reachable set should be neglected, which also requires to have an estimate of the reachable set. Finally, in an active learning setting, the aim should be to find a controller that is able to reach all possible points of the system, which is directly associated with estimating the reachable set.

1.1. Goals and Contributions

The goal of this work is to learn an inverse model to a redundant system, in particular motivated for control of musculoskeletal systems. The problem of learning an inverse model is inherently linked with the estimation of the reachable set. Hence, learning the inverse model and estimating its reachable set should be treated jointly and performed simultaneously within one framework. Due to the high dimensionality of the involved control input space, sampling the data from the real system has to be performed efficiently with the ambition of being able to both learn an accurate inverse model and estimate the reachable set. In the present work, a complete methodology, where bounds on the real performance error of the inverse are the central piece, is developed to achieve all three aims: inverse model learning, estimation of the reachable set and active exploration for data collection.

First, by formalizing more rigorously what it actually means to learn an inverse model, a methodology is derived where the inverse model, represented as a neural network, is learned by minimizing an upper bound on the real performance error. This upper bound is provided through a forward model (kernel regression), which is trained on the currently available data.

Determining the reachable set is a challenging problem that has been studied mainly for known analytical models. In many learning based methods, the problem of the reachable set is either ignored or it is assumed that the reachable set is known a priori. To overcome this, the present work secondly proposes a method to estimate the reachable set based on the derived error bound of the current learned forward and inverse model. This estimated reachable set is represented in a way that allows to decide whether a point is part of the estimated workspace easily. Therefore, the estimated reachable set is also efficient to compute.

Thirdly, an active exploration strategy is developed to efficiently generate the training set of the forward model with the goal of thereby improving the inverse model. This is realized by maximizing a lower bound on the true fill-distance of the real workspace (the domain of the inverse model), which is again provided through the error estimate. This strategy inherently trades-off exploration and exploitation. Since it explores in the low dimensional workspace instead of the high dimensional control input space, efficient data generation is possible.

A key feature of the proposed framework is that the learned inverse model provides guaranteed upper bounds on its performance when applied to the real system. These bounds are easy to compute and, as it is shown in the experiments, are also suitable in practice.

To summarize, the main methodological contributions of this work are

- Formalization of inverse model learning as minimization of an upper bound on the real performance of the inverse
- Inverse model learning framework with model quality estimation
- Estimation of the reachable set
- Active exploration strategy
- Error estimate for regularized kernel regression

and from an application point of view

- Control of a simulated musculoskeletal model of a human arm with 6 muscles and 2 joints to reach desired positions in its complete reachable workspace
- Lambda control learning for monosynaptic reflex
- Control of a real muscle driven robot with 2 joints and 5 pneumatic muscles mimicking the human arm model.

2. Related Work

2.1. Inverse Model Learning

Models play a central role in robotics and many other disciplines. In general, one can distinguish between so-called forward models, which describe how a system reacts to control inputs, and inverse models, which predict the necessary actions that lead to a desired change of the state of the system [34]. In most situations, forward models are unique in the sense that given the current state of the system and the control input, the next state is deterministically determined, if noise or other disturbances are neglected. Therefore, learning a forward model in continuous spaces in this situation is a supervised regression problem, which has been studied extensively in the machine learning community. In contrast, especially in robotics, there are infinitely many inverse models for the same system in nearly all relevant and interesting situations. Therefore, learning inverse models is much more challenging than learning forward models, since standard regression techniques cannot directly be applied.

This non-uniqueness of inverse models naturally arises in inverse kinematics. For redundant and even also for non-redundant kinematic chains, there are multiple (sometimes infinitely many) joint configurations that lead to the same task value. If a dataset contains such multiple solutions from sampling from the system, simple regression methods would average over those multiple configurations that correspond to the same task value. The average, however, is in most situations not a configuration that leads to the desired task value [14], [24], [34]. In some cases, the average would even be a completely invalid joint configuration that for example violates the joint limits. Since inverse kinematics is a basic and fundamental problem in robotics, many methods have been developed that address the non-uniqueness of inverse models specifically in the context of inverse kinematics.

One way to handle this averaging is to weight the data samples according to some objective [14], [40], [35]. If the joint configuration samples in the dataset that correspond to the same task value are weighted differently, a regression method would be driven towards those samples with higher weight. For example, the authors of [14] consider joint configurations as more important that are closer to a homing position of the robot. This allows to resolve the redundancy and also to influence which of the infinitely many inverses would be learned. A key insight of [35], where operational space control is learned, is to choose the weighting in accordance to the rigid body dynamics of the torque controlled robot. In contrast, for musculoskeletal systems, the choice of a suitable weighting objective is unclear and in [13], it was shown that such weightings therefore do not perform well for musculoskeletal systems.

Another method to resolve the non-uniqueness is to first learn a forward model, for which, as mentioned above, standard regression techniques can be used. The inverse itself is then obtained as a right inverse to the learned forward model [24], [34]. This way, the redundancy is resolved implicitly through the learned forward model. Such learning of the inverse through a forward model is also called distal teacher learning [24].

While the original distal teacher formulation in [24] considers one global inverse model, another approach is to learn multiple paired forward and inverse models [19], [27], [9]. The main idea of those approaches is that each forward-inverse pair represents different, smaller parts of the space. This is especially interesting in situations where the solution sets are not connected. Having multiple paired forward-inverse models, the question arises how from the multiple inverse models an actual prediction is performed, which is also called the module selection problem [19]. To this end, one way is to select the responsibility of the different inverse models depending on the current context. This context could for example be the state of the system or even a learned responsibility predictor. In an online setting, another way is to select the inverse models based on the current prediction errors of their corresponding forward models [19]. Instead of a hard selection, in those approaches typically multiple inverses contribute at the same time to the final prediction with different weightings based on their responsibility [19].

One disadvantage of these methods that use the forward model to guide the learning of the inverse model is that it is usually not taken into account that the learned forward model is imperfect. If the data for the forward model is sparse, as we show in the experiments, the forward model could guide the inverse unreasonably and therefore could lead to undesired results.

Speaking of data, most works mentioned so far also consider mainly given datasets or specify the data generation manually. Generating the data efficiently is, however, an important problem for the success of an inverse learning method. Due to the high dimensionality of the control input space, dense sampling is prohibitive. To address this issue, several authors propose to explore in the workspace/goal space instead of the control input space [40], [38], [2], [13]. By bootstrapping the iteratively learned inverse model, so-called goal babbling approaches [40], [38], [13] showed to generate the data to obtain an inverse model sample efficiently in high dimensional control input spaces. For example, in [40] inverse models for hyper-redundant kinematic chains are obtained. The idea behind bootstrapping the inverse model is that a target in the goal space is chosen. Then the control input is determined by the current learned inverse model, which is used to iteratively sample the next point.

The main limitation of those goal babbling methods is that either the exploration targets in the goal space are manually specified or chosen heuristically by knowing the true workspace. A more systematic approach to generating targets for the exploration in the goal space is intrinsic motivation, for example by estimating the competence progress [2], which also shows impressive results for challenging tasks.

These methods, however, also do not systematically estimate the reachable set. The only work the author of this document is aware of that deals with estimating the reachable set for learning inverse models is [39]. However, there is no representation of the reachable workspace which could be computed easily and the exploration in one direction stops if an heuristically chosen criteria indicates the end of the workspace. As it is discussed in section 5, where the main methodology of the present work is presented, estimating the reachable set is, however, an essential and inherent part of learning an inverse model. Therefore, the present work develops an approach to systematically estimate the reachable set in form of a representation that is also computationally feasible to determine simultaneously to learning the inverse model.

2.2. Active Learning in Robotics

So-called active learning deals with collecting data in a most informative manner. Especially in robotics the data collection procedure often involves real world experiments. Therefore, sampling necessary data while avoiding uninformative samples is of great importance for learning methodologies in robotics. One instance where active learning methodologies have successfully been applied in robotics is the use of Bayesian optimization [32] to tune controller parameters [12], [30], [4]. The main idea behind Bayesian optimization is to learn a surrogate cost function as a probabilistic model, most common a

Gaussian process [37]. Based on the information encoded in the learned Gaussian process, through a so-called acquisition function the next query location is selected to either find a better optimum or to explore unobserved parameter regions.

Most active learning methods are derived under the assumption that the unknown function can be queried at any chosen location. For example, in Bayesian optimization it is assumed that one can test the parameter the acquisition function recommends on the system. In the case of inverse model learning, when exploring in the workspace to circumvent the high dimensionality of the input space as described above, it is not possible to query an arbitrary desired location in the workspace, since this would require an already known perfect inverse. In [10] and [11] this problem of not being able to query desired targets directly has been addressed in the context of active tactile exploration.

In the present work, an active learning principle to generate the data for the forward model, while exploring in the low dimensional workspace and taking the imperfection of the learned forward model into account, is derived.

2.3. Learning to Control Musculoskeletal Systems

Several authors consider learning controllers for pneumatically driven robots [21], [44], [8], [13], [7], [18]. In [21] and [44] the redundancy problem is circumvented since only one control signal for an antagonistic muscle pair is learned. The other one is calculated such that both sum up to a constant. This is limiting, since not only the co-contraction can not be altered, depending on the chosen constant, it is even not possible to exhaust the complete possible motion range with this parameterization. Reinforcement learning is utilized in [8] to learn a controller for a finger with 2 joints driven by 4 muscles. However, the approach is limited to reach one single target position after learning, which means that no general controller is learned.

In [13] and [7] the non-uniqueness is resolved by only learning a forward model. The actual inverse query is then obtained by solving a non-convex optimization problem. Both [13] and [7] include a term in the optimization objective that ensures that the optimization problem stays close to the collected data. In the present work, a similar term is derived more rigorously, see section 5.2. With respect to the data generation in [13], the targets in the workspace are specified manually, but the current learned inverse model is then bootstrapped to sample the system to generate the data for the forward model. In contrast, the approach of [7] requires a preexisting controller to generate the data. Furthermore, the authors of [7] consider only one pair of muscles, which simplifies the problem due to the greatly reduced dimensionality. The present work and [13], in contrast, consider 6 muscles and a two dimensional workspace. However, [7] learns a full dynamics model, whereas [13] and the present work consider a static inverse only.

2.4. Control of Musculoskeletal Systems

Apart from learning based approaches, model based optimal control techniques have been studied to obtain a controller for musculoskeletal systems. In [29] a seven degree of freedom musculoskeletal arm model is controlled with a hierarchical optimal control methodology. Scaling up, in [28] 120 muscles for bipedal locomotion are controlled. While leading to impressive results, these works rely on an exact model of the musculoskeletal system. In addition, it is assumed that the algorithm has access to the full state information. As discussed in section 3, a biological system does not have access to all states.

3. Biomechanical Models of Musculoskeletal Systems

In its most fundamental form, a biological motion apparatus can be described by supporting passive structures and active elements that are influenced by the nervous system. In the following, the computational methods to model musculoskeletal systems for the purpose of this work are introduced. Furthermore, properties of such systems and their consequences are discussed.

3.1. Skeletal Model

As a first important part, the question arises how the supporting skeleton of a biological organism can be modeled. Due to the mechanical properties of the bones [3], [31], it is common to assume rigid body dynamics for the skeletal apparatus. Let $\mathbf{q} \in \mathbb{R}^{d_q}$ denote the joint configuration of a musculoskeletal system with d_q joints. The Euler-Lagrange equations

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \mathbf{R}(\mathbf{q})\mathbf{f}^{\mathrm{MTU}}, \tag{3.1}$$

relate the joint acceleration $\ddot{\mathbf{q}}$ with the exerted forces $\mathbf{f}^{\mathrm{MTU}} \in \mathbb{R}^m$ from the muscles. Here, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{d_q \times d_q}$ is the positive definite inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{d_q}$ and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{d_q}$ denotes the Coriolis/centripetal and gravity force vector, respectively.

The term $\mathbf{R}(\mathbf{q}) \in \mathbb{R}^{d_q \times m}$ is a nonlinearly on the current joint configuration depending matrix that represents the moment arms through which the linear force of the muscle actuators is translated to rotational torques in the joints. These moment arms are of great importance for the physiological accuracy. To model this nonlinear relationship, several methods have been proposed. This work uses a quadratic polynomial with parameters adjusted to fit experimental data [3].

Soft tissues like wobbling masses are also important to understand biological motion, especially for the correct modeling of impact dynamics [16]. However, since in this work mainly arm movements with low dynamics are considered, wobbling masses are not taken into account.

3.2. Muscle Models

One of the most essential aspects of computational methods for musculoskeletal systems is the muscle model itself. Experiments reveal that biological muscles exhibit a characteristic nonlinear force-length and force-velocity relation [17]. In order to recreate this behavior, vertebral muscles have been studied on various scales, ranging from atomistic over continuum mechanical models to macroscopic lumped models. Lumped muscle models not only give insights in the macroscopic behavior of muscles, they are also efficient to compute, which is important for the success of this work in finite time. The most common lumped muscle model is the so-called Hill-type model [22], which consists of a contractile element (CE), a parallel elastic element (PEE) and a serial elastic element (SEE). In [17] this model has been extended to include a serial damping element (SDE), which is used in this work. The four elements form a so-called muscle tendon unit (MTU). A visualization of the elements is shown in figure 3.1. The exerted force of this Hill-type MTU model with serial damping element is described by a differential equation

$$\dot{l}^{\mathrm{CE}} = f_{\mathrm{CE}}\left(l^{\mathrm{MTU}}, \dot{l}^{\mathrm{MTU}}, l^{\mathrm{CE}}, a\right) \tag{3.2}$$

and an algebraic equation

$$f^{\mathrm{MTU}} = f_F\left(l^{\mathrm{MTU}}, \dot{l}^{\mathrm{MTU}}, l^{\mathrm{CE}}, a\right). \tag{3.3}$$

Here, l^{CE} is an internal state that corresponds to the length of the contractile element, l^{MTU} and $\dot{l}^{\mathrm{MTU}}$ is the length of the complete MTU and its velocity, respectively. The quantity a denotes the muscle activity, which influences the contraction of the contractile element and the damping characteristics of the serial damping element. The forces exerted by each of the m MTUs are summarized in the vector

$$\mathbf{f}^{\mathrm{MTU}} = \left(f_1^{\mathrm{MTU}}, \dots, f_m^{\mathrm{MTU}}\right) \in \mathbb{R}^m. \tag{3.4}$$

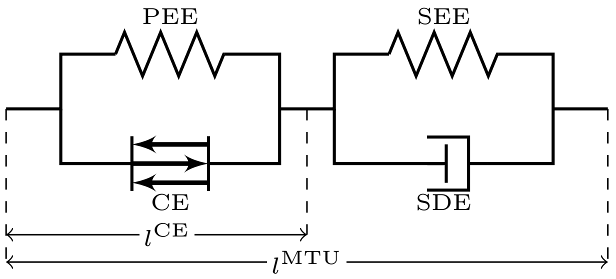


Figure 3.1: Hill-type model of a muscle tendon unit consisting of a contractile element (CE), a parallel elastic element (PEE), a serial elastic element (SEE) and a serial damping element (SDE).

3.3. Activation Dynamics

So far, the produced forces of the muscle tendon units depend on their activation state a. The so-called excitation-contraction coupling relates actual motor commands from the nervous system, the muscle *stimulations*, to the muscle *activity*. According to the Hatze approach [20], this excitation-contraction coupling is modeled with an additional, nonlinear first order dynamical system for each MTU, summarized in vector form

$$\dot{\mathbf{a}} = f_a(\mathbf{a}, \mathbf{l}^{\mathrm{CE}}, \mathbf{u}), \tag{3.5}$$

where $\mathbf{a} = (a_1, \dots, a_m)^T \in \mathbb{R}^m$ is the vector of all muscle activations, $\mathbf{l}^{\mathrm{CE}} \in \mathbb{R}^m$ the vector of the contractile element lengths and $\mathbf{u} \in \mathcal{U} = [0,1]^m$ denotes the normalized muscle stimulation of all muscles. This activation dynamics introduces additional delays [33], increases the state dimension for each muscle and acts as a low pass filter.

3.4. Antagonistic Setup – Redundancy

Since a biological muscle can only actively produce forces in contraction direction, at least two muscles are required to articulate a joint. A muscle pair that drives one joint is called a monoarticular muscle. In addition, biological systems found in nature also have so-called biarticular muscles that drive multiple joints at the same time. Therefore, a musculoskeletal system has significantly more actuators and hence control inputs than kinematic degrees of freedom [42]. One advantage of the antagonistic setup is that through different co-contraction levels the stiffness of the system can be adjusted. The consequences of this property are discussed in more detail in the next paragraph.

3.5. Attractor and Equilibrium Points

The rigid body, internal muscle and activation dynamics form a system of differential equations

$$\dot{\mathbf{y}} = f(\mathbf{y}, \mathbf{u}) \tag{3.6}$$

that is driven by the muscle stimulation input $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$. The state vector is

$$\mathbf{y} = \begin{pmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{l}^{\mathrm{CE}} \\ \mathbf{a} \end{pmatrix} \in \mathbb{R}^{2m+2d_q}, \tag{3.7}$$

where d_q is the number of kinematic degrees of freedom and m the number of muscles. Algebraic equations complete (3.6) to a complete description of the system. The dynamical system (3.6) is in an equilibrium state $(\mathbf{y}^*, \mathbf{u}^*)$ if

$$f(\mathbf{y}^*, \mathbf{u}^*) = 0. \tag{3.8}$$

Such equilibria can have different kind of stability properties. The redundant, antagonistic muscle setup, together with their characteristic force-length and force-velocity relations now have an important consequence to the stability properties of equilibrium points of the system. In figure 3.2, which is taken from [13], the resulting torque generated from two antagonistic muscles for one joint is shown for different static activation levels. As one can see, the resulting joint torque is zero at different joint angles, depending on the activation level. In case of an external perturbation, for example an increase in the joint angle, the passive properties of the MTUs generate a torque opposing the perturbation [13]. If the activation is increased in both muscles at the same time, which is called co-contraction, the joint stiffness, as seen by the slope of the violet line in figure 3.2, increases, while the equilibrium angle remains the same. Therefore, through the dynamics and the antagonistic setup of the MTUs, the joint behaves like a spring-damper system with tunable equilibrium angle and stiffness/damping properties [13].

This behavior implies that for a constant muscle stimulation, the musculoskeletal system reaches, under some assumptions, a certain stable equilibrium configuration. The question remains how large the attractor region for such an equilibrium point is. To formalize this, for an equilibrium point $(\mathbf{y}^*, \mathbf{u}^*)$, i.e. $f(\mathbf{y}^*, \mathbf{u}^*) = 0$, the set

$$\mathcal{A}(\mathbf{y}^*, \mathbf{u}^*) = \left\{ \boldsymbol{\xi} \in \mathbb{R}^{2m+2d_q} : \lim_{t \to \infty} \mathbf{y}(t) = \mathbf{y}^* \text{ for } \mathbf{y}(\cdot) \text{ solution of } \dot{\mathbf{y}} = f(\mathbf{y}, \mathbf{u}^*), \mathbf{y}(0) = \boldsymbol{\xi} \right\} \tag{3.9}$$

defines all initial conditions of the musculoskeletal system that converge to $\mathbf{y}^*$ for the constant control input $\mathbf{u}^*$. In this work, for the considered systems, it is assumed that for all equilibrium points $(\mathbf{y}^*, \mathbf{u}^*)$ it holds

$$\mathcal{A}(\mathbf{y}^*, \mathbf{u}^*)\Big|_{\mathcal{X}} = \mathcal{X}, \tag{3.10}$$

where $\mathcal{X} \subset \mathbb{R}^{d_q}$ denotes the set of all possible joint configurations. This means that independent from the start configuration, the musculoskeletal system can be controlled with a static muscle stimulation to a desired target configuration.

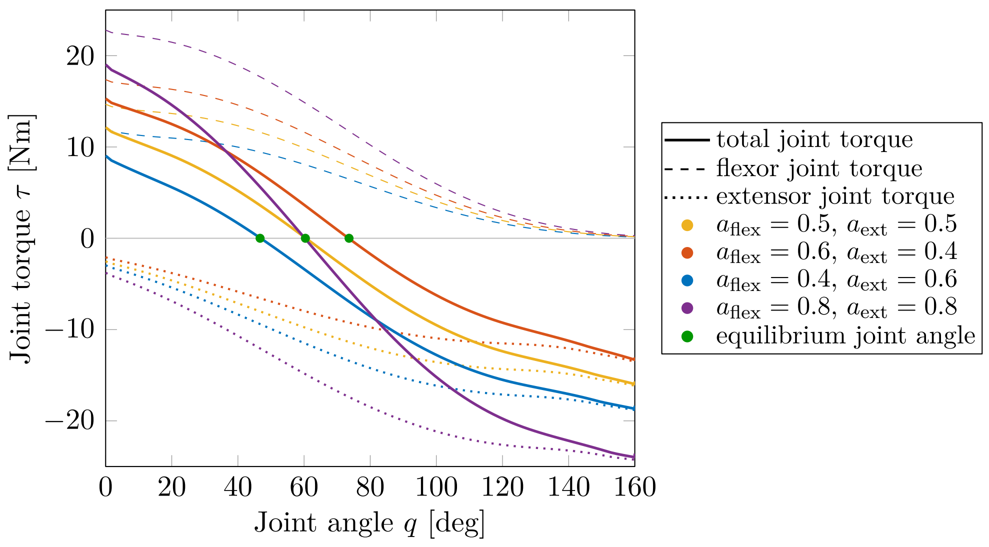


Figure 3.2: Relation between static muscle activity a and equilibrium states (green points) for an exemplary musculoskeletal model with one joint and two antagonistic muscles. Plot and data from [13].

3.6. Monosynaptic Reflex – Lambda Control and Hybrid Control

One hypothesis in animal motor control is that muscle stimulations are generated through a monosynaptic feedback loop for desired lengths $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_m)^T \in \mathbb{R}^m$ of the contractile elements, so-called lambda control [15]. The actual measured lengths $\mathbf{l}^{\text{CE}}$ are provided through the muscle spindels [23].

A basic feedback loop to realize this is a PD control scheme

$$\mathbf{u}_{\text{cl}} = \mathbf{K}_p \left(\mathbf{L}^{\text{CE}}_{\text{opt}}\right)^{-1} \left(\mathbf{l}^{\text{CE}} - \boldsymbol{\lambda}\right) - \mathbf{K}_d \dot{\mathbf{l}}^{\text{CE}}, \tag{3.11}$$

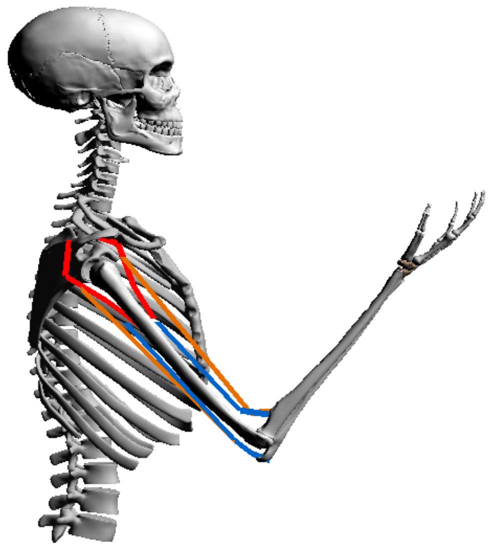
where $\mathbf{K}_p \in \mathbb{R}^{m \times m}$ and $\mathbf{K}_d \in \mathbb{R}^{m \times m}$ are the (diagonal) gain matrices. The matrix $\mathbf{L}^{\text{CE}}_{\text{opt}}$ acts as a normalization. Note that this is not a feedback on a desired configuration like position or joint angles of the system. It is a feedback on an internal state of the system.

Combining static open loop muscle stimulations $\mathbf{u}$ with this feedback on the length of the contractile elements is known as hybrid control. In this case, the desired lengths $\boldsymbol{\lambda}$ correspond to the lengths of the contractile elements in the equilibrium state that is reached through $\mathbf{u}$ alone. This way, the static open loop muscle stimulation defines the final configuration, the feedback on $\mathbf{l}^{\text{CE}}$ is used for both disturbance rejection and the dynamic behavior towards the equilibrium configuration.

3.7. Simulated Model of a Human Arm with Two Joints and Six Muscles

The actual simulated musculoskeletal system that is used throughout this work is a model of the human arm consisting of the upper and lower arm segment that are connected by the elbow joint. The upper arm segment is attached to the static torso by a revolute shoulder joint. Thus, the model is able to perform movements in the sagittal plane. The two joints are articulated by 6 MTUs in total. For each joint there is a pair of monoarticular MTUs. The two elbow muscles perform elbow flexion and extension, the two monoarticular shoulder muscles anteversion and retroversion. Additionally, two biarticular MTUs articulate both joints simultaneously. For simplicity and since only movements are possible in a 2 dimensional plane, the shoulder and biarticular muscles are also called flexors and extensors in this work. The model and its parameters have been developed in [43], which itself is based on [3].

Figure 3.3 left visualizes the simulation model, including the muscles. In figure 3.3 right, the true reachable set $\mathcal{X}$ of the arm model is shown. Two aspects have to be mentioned here. First, the true workspace is also only an estimation based on the kinematic data of the simulation model. This estimation does not take into account that the joint limits are modeled as linear damped springs. Secondly, the reachable space of a real human arm is larger. In the simulation model, the motion range has been reduced to ensure that the the attractor property holds everywhere in $\mathcal{X}$ and that the model stays within its validated range.



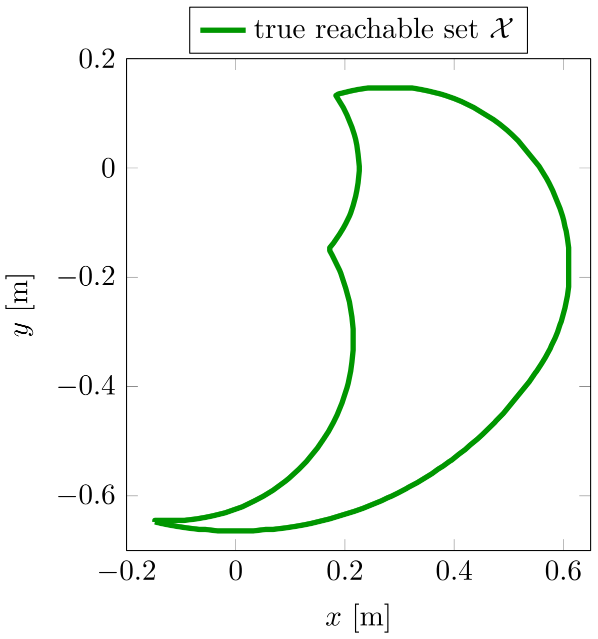


Figure 3.3: Left: Simulation model of human arm with elbow and shoulder joint to perform movements in the sagittal plane. Red lines are the two monoarticular shoulder muscle tendon units (MTUs), orange the two biarticular ones and blue depicts the two monoarticular elbow MTUs. The $\mathcal{X}$-space is the position of the hand of the arm in the sagittal plane. Right: True reachable set/workspace of the simulated arm model. The term “true” means that it is estimated based on the kinematics of the simulation model, not taking the way the joint limits are modeled as linear damped springs into account.

3.8. Real Bio-Inspired Muscle-Driven Robot Mimicking a Human Arm

In addition to the simulation model, the whole methodology is also evaluated on a real muscle-driven robot that mimics the simulation model of a human arm. Recreating the properties of biological muscles in a technical actuator is challenging. So-called pneumatic artificial muscles (PAMs) exhibit a similar nonlinear force-length and force-velocity relation as found in biology [26]. Figure 3.4 shows the real robot that is driven by 5 pneumatic muscles. Similar to the simulation model, the robot consists of the shoulder and elbow joint to allow movements in the sagittal plane. To further enhance the biological characteristics of the technical system, springs are added in series to the PAM actuators to form so-called muscle spring units (MSUs). As in the simulation model, two pairs of monoarticular MSUs drive each joint separately. A fifths MSU acts as a biarticular drive. The control input $\mathbf{u} \in [0,1]^5$ corresponds to pressures for the pneumatic muscles of 0 to 5 bar.

One of the biggest differences that is relevant for the present work between the biological MTU and the technical MSU is that the pneumatic muscles have very limited stretch and contract capabilities. More specifically, PAM actuators can only contract to 75% of their rest length, compared to the biological muscle, which is able to contract to 40% and be stretched to 170% of its rest lengths. Therefore, the possible motion range of the real robot is much smaller than of the simulated model. For further details about this robot refer to [13].

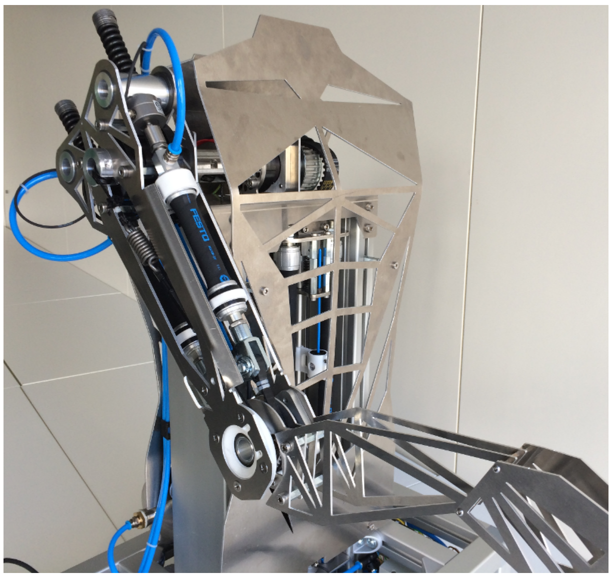


Figure 3.4: Real muscle-driven robot model mimicking the human arm model. The two joints (shoulder and elbow) are articulated by 5 pneumatic muscle spring units (MSUs), allowing movements in the sagittal plane. Two monoarticular MSUs drive each joint separately. The fifth MSU articulates both joints at the same time as a biarticular muscle to perform shoulder anteversion and elbow flexion [13].

4. Methodological Foundations

One purpose of this section is to introduce the necessary background about the main function approximation methods utilized in this work as well as to introduce the corresponding notation. More importantly, the central error estimate for regularized kernel regression is derived here in section 4.1.1.

4.1. Kernel Regression

A central part of this work is to learn a function $\phi : \mathcal{U} \to \mathcal{X}$, $\mathcal{U} \subset \mathbb{R}^m$, $\mathcal{X} \subset \mathbb{R}^d$ from data describing the input-output mapping, while additionally, under some assumptions, being able to estimate the error between the true function and the learned approximate. Kernel based methods are a suitable choice for this task, since the underlying theory of reproducing kernel Hilbert spaces enables to derive such error estimates elegantly. However, kernel methods also have disadvantages, which are discussed in sections 7.3 and 7.4. First, basic definitions and properties are introduced.

Definition 4.1 (Kernel)**.** Let $\mathcal{U}$ be an arbitrary, non-empty set. Then a symmetric function $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is called a kernel. Symmetry means $\forall_{\mathbf{u},\mathbf{u}' \in \mathcal{U}} : k(\mathbf{u}, \mathbf{u}') = k(\mathbf{u}', \mathbf{u})$.

Definition 4.2 (Positive Definite Kernel)**.** A kernel $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is called positive definite, if for all $w \in \mathbb{N}$ and all $\{\mathbf{u}_i\}_{i=1}^w \subset \mathcal{U}$, the kernel matrix $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^w \in \mathbb{R}^{w \times w}$ is positive semi definite.

Definition 4.3 (Strictly Positive Definite Kernel)**.** A kernel $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is called strictly positive definite, if for all $w \in \mathbb{N}$ and all $\{\mathbf{u}_i\}_{i=1}^w \subset \mathcal{U}$, $\mathbf{u}_i \neq \mathbf{u}_j$, $i \neq j$ the kernel matrix $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^w \in \mathbb{R}^{w \times w}$ is positive definite.

A very popular choice of a kernel is the Gaussian, which is sometimes also called squared exponential kernel.

Proposition 4.4 (Gaussian Kernel)**.** For a symmetric positive definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$, the Gaussian or squared exponential kernel is defined as

$$k(\mathbf{u}, \mathbf{u}') = \exp\left(-\frac{1}{2}\left(\mathbf{u} - \mathbf{u}'\right)^T \mathbf{\Sigma}^{-1} \left(\mathbf{u} - \mathbf{u}'\right)\right). \tag{4.1}$$

This kernel is strictly positive definite. Often $\mathbf{\Sigma} = \operatorname{diag}(l_1^2, \ldots, l_m^2)$, $l_m > 0$.

Proof. See [45]. □

Definition 4.5 (Reproducing Kernel Hilbert Space (RKHS))**.** Let $\mathcal{U} \neq \emptyset$. The Hilbert space H_k of functions $\phi : \mathcal{U} \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{H_k}$ is called a Reproducing Kernel Hilbert Space (RKHS) if and only if there exists a function $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ such that

$$\forall_{\mathbf{u} \in \mathcal{U}} \; : \; k(\mathbf{u}, \cdot) \in H_k \tag{4.2}$$

$$\forall_{\phi \in H_k} \forall_{\mathbf{u} \in \mathcal{U}} \; : \; \langle \phi, k(\mathbf{u}, \cdot) \rangle_{H_k} = \phi(\mathbf{u}). \tag{4.3}$$

The latter is called *reproducing property.*

Theorem 4.6 (Kernel from RKHS)**.** The function $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ from the definition 4.5 of the RKHS is a positive definite kernel.

Proof. Symmetry: Using the reproducing property (4.3) and the symmetry of an inner product

$$k(\mathbf{u}, \mathbf{u}') = \langle k(\mathbf{u}, \cdot), k(\mathbf{u}', \cdot) \rangle_{H_k} = \langle k(\mathbf{u}', \cdot), k(\mathbf{u}, \cdot) \rangle_{H_k} = k(\mathbf{u}', \mathbf{u}). \tag{4.4}$$

Positive definiteness: Choose $\varphi : \mathcal{U} \to H_k$, $\varphi(\mathbf{u}) = k(\mathbf{u}, \cdot)$ (so-called feature space representation). Then

$$k(\mathbf{u}, \mathbf{u}') = \langle k(\mathbf{u}, \cdot), k(\mathbf{u}', \cdot) \rangle_{H_k} = \langle \varphi(\mathbf{u}), \varphi(\mathbf{u}') \rangle_{H_k}, \tag{4.5}$$

which is positive definite. □

Theorem 4.7 (RKHS from Kernel)**.** Let k be a (strictly) positive definite kernel. Then there exists an RKHS H_k with reproducing kernel k.

Proof. See [45] □

An important class of functions $\phi \in H_k$ are finite linear combinations of kernels. In this case, the RKHS inner product can easily be calculated as a quadratic form.

Proposition 4.8 (RKHS Inner Product)**.** Let $\phi_1 = \sum_{i=1}^{n_1} b_i^1 k(\mathbf{u}_i^1, \cdot)$ and $\phi_2 = \sum_{i=1}^{n_2} b_i^2 k(\mathbf{u}_i^2, \cdot)$ be two functions $\phi_1 \in H_k$, $\phi_2 \in H_k$. Then

$$\langle \phi_1, \phi_2 \rangle = \mathbf{b}_1^T \mathbf{K}^{1,2} \mathbf{b}_2 \tag{4.6}$$

with $\mathbf{K}^{1,2} = \left(k(\mathbf{u}_i^1, \mathbf{u}_j^2)\right)_{i,j=1}^{n_1,n_2} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{b}_1 = (b_i^1)_{i=1}^{n_1} \in \mathbb{R}^{n_1}$, $\mathbf{b}_2 = (b_i^2)_{i=1}^{n_2} \in \mathbb{R}^{n_2}$.

Proof. This relies on the reproducing property (4.3) and bilinearity of a scalar product.

$$\langle \phi_1, \phi_2 \rangle = \left\langle \sum_{i=i}^{n_1} b_i^1 k(\mathbf{u}_i^1, \cdot), \sum_{j=1}^{n_2} b_j^2 k(\mathbf{u}_j^2, \cdot) \right\rangle = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} b_i^1 b_j^2 \langle k(\mathbf{u}_i^1, \cdot), k(\mathbf{u}_j^2, \cdot) \rangle \tag{4.7}$$

$$= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} b_i^1 b_j^2 k(\mathbf{u}_i^1, \mathbf{u}_j^2) = \mathbf{b}_1^T \mathbf{K}^{1,2} \mathbf{b}_2. \tag{4.8}$$

□

Proposition 4.9 (RKHS Norm)**.** Let $\phi = \sum_{i=1}^{n} b_i k(\mathbf{u}_i, \cdot)$. Then the RKHS norm of ϕ admits the form

$$\|\phi\|_{H_k} = \sqrt{\mathbf{b}^T \mathbf{K} \mathbf{b}} \tag{4.9}$$

with $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^{n} \in \mathbb{R}^{n \times n}$, $\mathbf{b} = (b_i)_{i=1}^{n} \in \mathbb{R}^n$.

Proof. Clear from proposition 4.8. □

The next theorem is of great importance for applications in machine learning and explains why finite dimensional linear combinations of kernels are an important class of functions.

Theorem 4.10 (Representer)**.** Let $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a positive definite kernel. $J : [0, \infty) \to \mathbb{R}$ a monotonically increasing function and $C : (\mathcal{U} \times \mathbb{R} \times \mathbb{R})^n \to \mathbb{R} \cup \{\infty\}$. For the set $\mathcal{D}_1 = \{(\mathbf{u}_i, x_i)\}_{i=1}^n$ (which will be called dataset later), if the (infinite dimensional) optimization problem

$$\min_{\phi \in H_k} C\left((\mathbf{u}_1, x_1, \phi(\mathbf{u}_1)), \dots, (\mathbf{u}_n, x_n, \phi(\mathbf{u}_n))\right) + J\left(\|\phi\|_{H_k}^2\right) \tag{4.10}$$

has a solution, then there exists a solution which can be written as

$$\phi = \sum_{i=1}^n b_i k(\mathbf{u}_i, \cdot) \tag{4.11}$$

with $b_i \in \mathbb{R}$, $i = 1, \dots, n$.

Proof. An elegant proof is due to [41]. □

The representer theorem implies that the solution of the infinite dimensional optimization problem can be expressed in terms of a finite combination of kernels located at the data inputs. This property is the key to use many kernel methods in practice. In the following, finally, based on the representer theorem, a kernel regression method is derived to learn a function from data.

Proposition 4.11 (Kernel Regression with RKHS Norm Regularization)**.** Let a dataset $\mathcal{D}_1 = \{(\mathbf{u}_i, x_i)\}_{i=1}^n$ be given, where $\mathbf{u}_i \in \mathcal{U}$ are the inputs and $x_i \in \mathbb{R}$ are the outputs. For the positive definite kernel $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ with associated RKHS H_k and a regularization parameter $\sigma \geq 0$, the kernel regression problem is defined as the optimization problem

$$\min_{\phi \in H_k} \sum_{i=1}^n |x_i - \phi(\mathbf{u}_i)|^2 + \sigma^2 \|\phi\|_{H_k}^2 . \tag{4.12}$$

If the kernel k is positive definite and $\sigma > 0$ or if the kernel is strictly positive definite and $\sigma \geq 0$, the unique solution of the kernel regression problem (4.12) is given by

$$\phi(\mathbf{u}) = \mathbf{x}^T \mathbf{G}^{-1} \boldsymbol{\kappa}(\mathbf{u}), \tag{4.13}$$

where $\mathbf{G} = \mathbf{K} + \sigma^2 \mathbf{I} \in \mathbb{R}^{n \times n}$, $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^{n \times n} \in \mathbb{R}^{n \times n}$, $\boldsymbol{\kappa}(\mathbf{u}) = (k(\mathbf{u}_i, \mathbf{u}))_{i=1}^n \in \mathbb{R}^n$, $\mathbf{x} = (x_i)_{i=1}^n \in \mathbb{R}^n$.

Proof. According to the representer theorem 4.10, the solution of (4.12) can be written as $\phi(\mathbf{u}) = \mathbf{b}^T \boldsymbol{\kappa}(\mathbf{u})$ for some $\mathbf{b} \in \mathbb{R}^n$. Therefore, (4.12) is equivalent to the finite dimensional, convex optimization problem

$$\min_{\mathbf{b} \in \mathbb{R}^n} \|\mathbf{x} - \mathbf{K}\mathbf{b}\|_2^2 + \sigma^2 \left\|\mathbf{b}^T \mathbf{K} \mathbf{b}\right\|_2^2 \tag{4.14}$$

which has the unique solution

$$\mathbf{b} = \mathbf{G}^{-1} \mathbf{x}, \tag{4.15}$$

since if k is positive definite and $\sigma > 0$, $\mathbf{G}$ is positive definite or if k is strictly positive definite (and only $\sigma \geq 0$), $\mathbf{G}$ is positive definite as well. □

Proposition 4.12 (Kernel Interpolation)**.** If $\sigma = 0$ and k is strictly positive definite, then $\phi(\mathbf{u}_i) = x_i$, i.e. the estimated function ϕ interpolates at the training input points. This implies that the optimal value of the objective 4.12 is zero.

Proof. For $\sigma = 0$, $\mathbf{G} = \mathbf{K}$. Furthermore, $\boldsymbol{\kappa}(\mathbf{u}_i) = \mathbf{K}_{:,i}$, which implies $\mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}_i) = \mathbf{e}_i$. □

Proposition 4.13 (RKHS Norm for Kernel Regression)**.** The RKHS norm of the kernel regression solution from proposition 4.11 can be calculated as

$$\|\phi\|_{H_k} = \sqrt{\mathbf{b}^T\mathbf{K}\mathbf{b}} = \sqrt{\mathbf{x}^T\mathbf{G}^{-1}\mathbf{K}\mathbf{G}^{-1}\mathbf{x}}. \tag{4.16}$$

Remark 4.14. The RKHS norm of a kernel regression solution is a function of both the observed support points $\mathbf{u}_i$ and their function values x_i.

Especially if using Gaussian kernels or other radial kernels, the value of the learned kernel regression function ϕ approaches zero where there is no data. Sometimes this would lead to undesired results. Therefore, one can introduce a mean prior function. For simplicity, only constant mean priors are considered here, although the extension to mean functions is straightforward.

Proposition 4.15 (Mean Prior)**.** Under the assumptions of proposition 4.11, the kernel regression problem with mean prior $m \in \mathbb{R}$ is defined es

$$\min_{\phi \in H_k} \sum_{i=1}^{n} |x_i - (\phi(\mathbf{u}_i) + m)|^2 + \sigma^2 \|\phi\|_{H_k}^2 \tag{4.17}$$

with the unique solution for a strictly positive definite kernel

$$\phi(\mathbf{u}) = (\mathbf{x} - m)^T \mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}) + m, \tag{4.18}$$

where $\mathbf{G} = \mathbf{K} + \sigma^2\mathbf{I} \in \mathbb{R}^{n\times n}$, $\mathbf{K} = (k(\mathbf{u}_i, \mathbf{u}_j))_{i,j=1}^{n\times n} \in \mathbb{R}^{n\times n}$, $\boldsymbol{\kappa}(\mathbf{u}) = (k(\mathbf{u}_i, \mathbf{u}))_{i=1}^{n} \in \mathbb{R}^n$, $\mathbf{x} = (x_i)_{i=1}^{n} \in \mathbb{R}^n$.

Proof. Analog to the proof of proposition 4.11. □

These were the main important definitions and properties of kernel based methods as required in this work. For more details about kernel theory, refer to [41], [45].

4.1.1. Error Estimate

As mentioned above and as will become clear later in section 5, a crucial aspect of this work is to estimate how well a learned function approximates the true one from which the data is generated. There are different kinds of error estimates. One type are asymptotic error bounds that measure the difference between the true and the learned function globally in terms of a specific function norm. These are typically used to investigate convergence rates. More useful for the purpose of this work are point-wise error estimates, which try to estimate the error between the true and learned function in a norm of their co-domains at a specific location in the input domain. If it is assumed that the true function comes from the same RKHS as the learned approximate, a point-wise error estimate can be derived elegantly. In the following such a point-wise error estimate for a regularized kernel regression (proposition 4.11) is stated. To the knowledge of the author of this document, this specific result has not been reported in the literature yet. However, both the structure of the error estimate and its proof are very similar to a well-known result from [45], where the unregularized case has been studied.

Theorem 4.16 (Kernel Regression Error Estimate)**.** If the true function $\hat{\phi} : \mathcal{U} \to \mathbb{R}$ comes from the same RKHS H_k as the kernel approximator ϕ from proposition 4.11, i.e. $\hat{\phi} \in H_k$ and fulfills $\left\|\hat{\phi}\right\|_{H_k} < \infty$, then for all $\mathbf{u} \in \mathcal{U}$ it holds

$$\left|\hat{\phi}(\mathbf{u}) - \phi(\mathbf{u})\right| \leq \left\|\hat{\phi}\right\|_{H_k} s(\mathbf{u}), \tag{4.19}$$

where

$$s(\mathbf{u}) = \sqrt{k(\mathbf{u},\mathbf{u}) - \boldsymbol{\kappa}(\mathbf{u})^T \mathbf{G}^{-1} \boldsymbol{\kappa}(\mathbf{u}) - \sigma^2 \boldsymbol{\kappa}(\mathbf{u})^T \mathbf{G}^{-2} \boldsymbol{\kappa}(\mathbf{u})}. \tag{4.20}$$

Before the proof can be given, two little lemmata are needed. First, the often helpful Woodbury identity.

Lemma 4.17 (Woodbury Matrix Inversion Formula)**.** Let $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{U} \in \mathbb{R}^{n\times k}, \mathbf{C} \in \mathbb{R}^{k\times k}, \mathbf{V} \in \mathbb{R}^{k\times n}$, $\det(\mathbf{A}) \neq 0$, $\det(\mathbf{A} + \mathbf{UCV}) \neq 0$, then

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{A}^{-1}. \tag{4.21}$$

Proof. See [36]. □

Next a compact representation of the solution of the kernel regression problem is stated.

Lemma 4.18. Define $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_n) : \mathcal{U} \to \mathbb{R}^n$ as

$$\mathbf{p}(\mathbf{u}) = \mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}), \tag{4.22}$$

where $\mathbf{G}$ and $\boldsymbol{\kappa}$ are defined as in proposition 4.11. Then the kernel regression solution (4.13) can be written as

$$\phi(\mathbf{u}) = \left(\hat{\phi}(\mathbf{u}_1), \ldots, \hat{\phi}(\mathbf{u}_n)\right)\mathbf{p}(\mathbf{u}). \tag{4.23}$$

For $\sigma = 0$, the elements of $\mathbf{p}$ are a Lagrange basis of the corresponding interpolation problem.

Proof. The representation (4.23) is clear. For $\sigma = 0$, it holds

$$\mathbf{p}(\mathbf{u}) = \mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}) = \mathbf{I}\mathbf{K}^{-1}\boldsymbol{\kappa}(\mathbf{u}) \tag{4.24}$$

and hence with proposition 4.12

$$\mathbf{p}(\mathbf{u}_i)_j = \mathbf{e}_j^T \mathbf{K}^{-1}\boldsymbol{\kappa}(\mathbf{u}_i) = \mathbf{e}_j^T \mathbf{e}_i = \delta_{ij}. \tag{4.25}$$

□

Now finally the error estimate can be proven.

Proof of theorem 4.16. For the error between the true function $\hat{\phi}$ and the learned approximate ϕ at an arbitrary $\mathbf{u} \in \mathcal{U}$ it holds

$$\begin{aligned}
\left|\hat{\phi}(\mathbf{u}) - \phi(\mathbf{u})\right|^2 &= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot)\right\rangle_{H_k} - \left\langle \phi, k(\mathbf{u},\cdot)\right\rangle_{H_k}\right|^2 && (4.26)\\
&= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot)\right\rangle_{H_k} - \left\langle \sum_{i=1}^n \hat{\phi}(\mathbf{u}_i)\mathbf{p}_i, k(\mathbf{u},\cdot)\right\rangle_{H_k}\right|^2 && (4.27)\\
&= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot)\right\rangle_{H_k} - \sum_{i=1}^n \left\langle \left\langle \hat{\phi}, k(\mathbf{u}_i,\cdot)\right\rangle_{H_k} \mathbf{p}_i, k(\mathbf{u},\cdot)\right\rangle_{H_k}\right|^2 && (4.28)\\
&= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot)\right\rangle_{H_k} - \sum_{i=1}^n \left\langle \hat{\phi}, k(\mathbf{u}_i,\cdot)\right\rangle_{H_k} \left\langle \mathbf{p}_i, k(\mathbf{u},\cdot)\right\rangle_{H_k}\right|^2 && (4.29)\\
&= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot)\right\rangle_{H_k} - \sum_{i=1}^n \left\langle \hat{\phi}, k(\mathbf{u}_i,\cdot)\right\rangle_{H_k} \mathbf{p}(\mathbf{u})_i\right|^2 && (4.30)\\
&= \left|\left\langle \hat{\phi}, k(\mathbf{u},\cdot) - \sum_{i=1}^n k(\mathbf{u}_i,\cdot)\mathbf{p}(\mathbf{u})_i\right\rangle_{H_k}\right|^2 && (4.31)\\
&\leq \left\|\hat{\phi}\right\|_{H_k}^2 \left\|k(\mathbf{u},\cdot) - \sum_{i=1}^n k(\mathbf{u}_i,\cdot)\mathbf{p}(\mathbf{u})_i\right\|_{H_k}^2 && (4.32)\\
&= \left\|\hat{\phi}\right\|_{H_k}^2 \left(k(\mathbf{u},\mathbf{u}) - 2\sum_{i=1}^n k(\mathbf{u}_\mathrm{i},\mathbf{u})\mathbf{p}(\mathbf{u})_i + \sum_{i=1}^n\sum_{j=1}^n k(\mathbf{u}_i,\mathbf{u}_j)\mathbf{p}(\mathbf{u})_i\mathbf{p}(\mathbf{u})_j\right) && (4.33)\\
&= \left\|\hat{\phi}\right\|_{H_k}^2 \left(k(\mathbf{u},\mathbf{u}) - 2\boldsymbol{\kappa}(\mathbf{u})^T\mathbf{p}(\mathbf{u}) + \mathbf{p}(\mathbf{u})^T\mathbf{K}\mathbf{p}(\mathbf{u})\right) && (4.34)\\
&= \left\|\hat{\phi}\right\|_{H_k}^2 \left(k(\mathbf{u},\mathbf{u}) - 2\boldsymbol{\kappa}(\mathbf{u})^T\mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}) + \boldsymbol{\kappa}(\mathbf{u})^T\mathbf{G}^{-1}\mathbf{K}\mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u})\right). && (4.35)
\end{aligned}$$

The right term on the right of (4.35) can further be simplified. To do so, the Woodbury identity (4.21) is applied to the term $\mathbf{G}^{-1}\mathbf{K}\mathbf{G}^{-1}$, which yields

$$\begin{aligned}
\mathbf{G}^{-1}\mathbf{K}\mathbf{G}^{-1} &= \mathbf{G}^{-1}\mathbf{K}\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} = \mathbf{G}^{-1}\mathbf{K}\left(\mathbf{K}^{-1} - \mathbf{K}^{-1}\left(\frac{1}{\sigma^2}\mathbf{I} + \mathbf{K}^{-1}\right)^{-1}\mathbf{K}^{-1}\right) && (4.36)\\
&= \mathbf{G}^{-1} - \mathbf{G}^{-1}\left(\mathbf{K}\left(\frac{1}{\sigma^2}\mathbf{I} + \mathbf{K}^{-1}\right)\right)^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\left(\frac{\sigma^2}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbf{K} + \mathbf{I}\right)\right)^{-1} && (4.37)\\
&= \mathbf{G}^{-1} - \sigma^2\mathbf{G}^{-1}\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} && (4.38)\\
&= \mathbf{G}^{-1} - \sigma^2\mathbf{G}^{-2}. && (4.39)
\end{aligned}$$

Inserting this into (4.35) gives

$$\left|\hat{\phi}(\mathbf{u}) - \phi(\mathbf{u})\right|^2 \leq \left\|\hat{\phi}\right\|_{H_k}^2 \left(k(\mathbf{u},\mathbf{u}) - \boldsymbol{\kappa}(\mathbf{u})^T\mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}) - \sigma^2\boldsymbol{\kappa}(\mathbf{u})^T\mathbf{G}^{-2}\boldsymbol{\kappa}(\mathbf{u})\right). \tag{4.40}$$

By taking the square root, the proposition follows. The first part of this proof followed a similar idea as in [45]. □

Readers familiar with Gaussian processes might recognize a similarity between the term (4.20) in the error estimate and the variance of a Gaussian process regression model. To show where this similarity comes from, we need the following lemma.

Lemma 4.19. $\mathbf{G}^{-2} = \mathbf{G}^{-1}\mathbf{G}^{-1}$ is positive definite. This also holds for $\sigma = 0$, if the kernel is strictly positive definite.

Proof. Let $\mathbf{a} \in \mathbb{R}^n \backslash \{0\}$, then

$$\mathbf{a}^T\mathbf{G}^{-2}\mathbf{a} = \mathbf{a}^T\mathbf{G}^{-1}\mathbf{G}^{-1}\mathbf{a} = \left\|\mathbf{G}^{-1}\mathbf{a}\right\|_2^2 > 0. \tag{4.41}$$

□

Proposition 4.20 (Relation to Gaussian Process Variance Estimate)**.** Let

$$\mathbb{V}(\mathbf{u}) = k(\mathbf{u}, \mathbf{u}) - \boldsymbol{\kappa}(\mathbf{u})^T\mathbf{G}^{-1}\boldsymbol{\kappa}(\mathbf{u}) \tag{4.42}$$

be the variance estimate of a Gaussian process model [37] for the regression problem defined in proposition 4.11. Then it holds

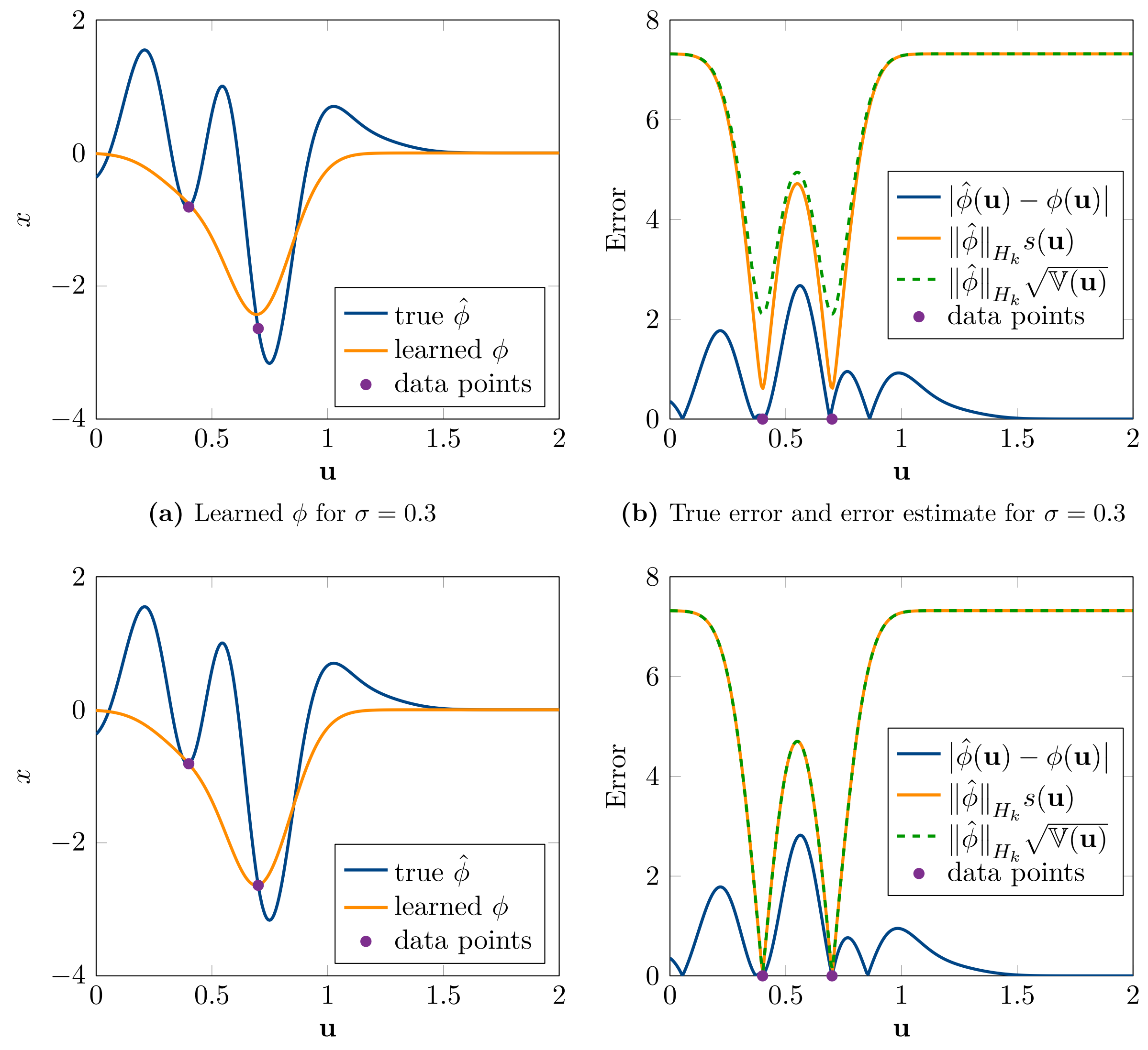
$$\left|\hat{\phi}(\mathbf{u}) - \phi(\mathbf{u})\right| \leq \left\|\hat{\phi}\right\|_{H_k} s(\mathbf{u}) \leq \left\|\hat{\phi}\right\|_{H_k} \sqrt{\mathbb{V}(\mathbf{u})}. \tag{4.43}$$

Proof. Follows directly from lemma 4.19 and theorem 4.16. □

Remark 4.21. Proposition 4.20 means that the bound derived in theorem 4.16 is tighter than an error estimate that is based on the variance of the Gaussian process.

Example 4.22 (Error Estimates)**.** To demonstrate the derived error estimate, figure 4.1 shows the true error between the true function and the learned kernel approximate as well as the estimated error for two different regularization parameters σ. In order to ensure that $\hat{\phi} \in H_k$, $\hat{\phi}$ is constructed as a linear combination of 7 Gaussian kernels at different locations with length scale $l = 0.2$. The true function has a RKHS norm of $\left\|\hat{\phi}\right\|_{H_k} = 7.32$, the learned approximate $\|\phi\|_{H_k} = 2.48$ for $\sigma = 0.3$ and $\|\phi\|_{H_k} = 2.69$ for $\sigma = 0.01$. One can see the regularizing effect on the RKHS norm of the parameter σ.

For $\sigma = 0.3$ (figure 4.1b) the derived bound from theorem 4.16 is more tight than the bound based on the variance of a Gaussian process. For lower regularization ($\sigma = 0.01$, figure 4.1d), the difference is not visible.



(a) Learned ϕ for $\sigma = 0.3$

(b) True error and error estimate for $\sigma = 0.3$

(c) Learned ϕ for $\sigma = 0.01$

(d) True error and error estimate for $\sigma = 0.01$

Figure 4.1: Visualization of the proposed error estimate for two different regularization parameters σ as well as comparison to the error estimate based on the variance of a Gaussian process.

4.2. Neural Networks

For representing the inverse model, this work utilizes feedforward neural networks. This section provides the basic notational background about neural networks necessary for the understanding of the rest of this work.

In general, one can think of a standard feedforward neural network as a parameterized function $\boldsymbol{\pi} : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}^m$. These parameters are also called the weights.

Definition 4.23 (Notation of Weights/Parameters)**.** Let $\mathcal{W} \subset \mathbb{R}^w$ be the space of all admissible parameters/weights of the neural network $\boldsymbol{\pi}$. This is denoted by

$$\boldsymbol{\pi}(\mathbf{x}) = \boldsymbol{\pi}(\mathbf{x}; \mathbf{w}). \tag{4.44}$$

A standard feedforward neural network now consists of multiple layers. In each layer, the input $\mathbf{y}_l \in \mathbb{R}^{n_l}$ is first affinely transformed $\mathbf{W}_l \mathbf{y}_l + \mathbf{b}_l$ with $\mathbf{W} \in \mathbb{R}^{n_l \times n_{l+1}}$, $\mathbf{b} \in \mathbb{R}^{n_{l+1}}$. The result is then processed element wise with a non-linear activation function a

$$\mathbf{y}_{l+1} = a(\mathbf{W}_l \mathbf{y}_l + \mathbf{b}_l), \tag{4.45}$$

leading to the output of this layer l. The dimensionality of the internal vectors of a specific layer is called the number of units of this layer. The matrices $\mathbf{W}_l$ and bias terms $\mathbf{b}_l$ of each layer are collected in the parameter $\mathbf{w} \in \mathcal{W}$. With respect to the activation functions, this work uses so-called ReLU functions.

Definition 4.24 (ReLU Activation Function)**.** The Rectified-Linear-Unit (ReLU) is defined as

$$a_{\text{ReLU}}(y) = \max(0, y). \tag{4.46}$$

Since $\boldsymbol{\pi}$ is used to predict muscle stimulations, it has to be ensured that $\boldsymbol{\pi}$ maps into the normalized muscle stimulation space $[0,1]^m$, according to section 3. To this end, a so-called sigmoid activation function is used.

Definition 4.25 (Sigmoid Activation Function)**.** The sigmoid is defined as

$$a_{\text{sig}}(y) = \frac{1}{1 + e^{-y}}. \tag{4.47}$$

Since $a_{\text{sig}}(\mathbb{R}) = (0,1)$, when using the sigmoid activation function for the output layer of $\boldsymbol{\pi}$, it holds $\boldsymbol{\pi}(\mathbb{R}^d) \subset (0,1)^m \subset [0,1]^m$.

A complete neural network with 2 hidden ReLU layers and a sigmoid output layer would, for example, look like

$$\boldsymbol{\pi}(\mathbf{x}) = a_{\text{sig}}\Big(\mathbf{W}_3 a_{\text{ReLU}}\big(\mathbf{W}_2 a_{\text{ReLU}}\left(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2\big) + \mathbf{b}_3\Big). \tag{4.48}$$

This structure is used for the experiments in this work to represent the inverse model $\boldsymbol{\pi}$, as described in section 5. The extension to arbitrary many layers is straightforward.

Now the major question remains how the weights of such a network can be optimized to solve a desired task. Assume that the objective of this task is defined in terms of a sum of the same objective L evaluated at a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, i.e. the weights of the neural network should optimize

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_{i=1}^{n} L(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\pi}(\mathbf{x}_i; \mathbf{w})). \tag{4.49}$$

Due to the often extremely high number of parameters, it is usually impossible to use second order optimization methods. Furthermore, there are many local minima, which makes (4.49) a challenging optimization problem. However, it turns out that gradient based methods like stochastic gradient descent work for many difficult problems. The main idea behind stochastic gradient descent is do gradient steps not on the complete dataset, but only on mini batches, i.e.

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \sum_{i=j}^{n_b} \nabla_{\mathbf{w}} L(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\pi}(\mathbf{x}_i; \mathbf{w})), \tag{4.50}$$

where $1 \leq j \leq n - n_b + 1$ with a batch size of $1 \leq n_b \leq n$ and the so-called learning rate $\alpha > 0$. A popular extension to stochastic gradient descent is Adam [25], which empirically can solve many difficult problems. There the idea is to maintain running averages of the gradients and their second moments for future gradient steps. For details, refer to [25].

5. Active Inverse Model Learning

From an abstract point of view, the goal of this work is to learn an inverse of a non-injective function. This function will be called the forward model and its inverse the inverse model. The forward model, in general, encodes how a system reacts to control inputs. More specifically, within the context of the control of musculoskeletal systems, the forward model describes the equilibrium configuration of the musculoskeletal system for a specific choice of muscle stimulations, the control input. The inverse model would then predict the necessary control inputs/muscle stimulations that leads to the desired state/configuration of the system.

In the following, it is first stated precisely what this work actually understands as learning an inverse model (section 5.1.2) and the involved challenges are identified. Based on the insights of formalizing the inverse model learning problem, concrete algorithms are derived that address these challenges. This includes learning the inverse model itself (section 5.2), a way to estimate the reachable set (section 5.3) and the active exploration strategy (section 5.4). The complete framework is then summarized in section 5.5, together with practical remarks. After the main methodology has been established, an extension to parameter dependent inverse models is developed in section 5.6. In section 5.7, it is shown how lambda control can be learned within this framework.

5.1. Formalization of Inverse Model Learning

5.1.1. Motivation

Before the actual inverse model learning problem is defined, it is first motivated why simply learning a function from data directly is not the answer to all questions.

A function is a mathematical object that is defined by its domain, which is a set, and the way it uniquely associates elements of the domain to another set, which is called the codomain. This association is usually termed as the mapping of the function. Therefore, the two functions f and g, which are defined by $f : [0, 1] \to \mathbb{R}$, $f(x) = x^2$ and $g : [-1, 1] \to \mathbb{R}$, $g(x) = x^2$ are *not* the same mathematical object.

Machine learning, approximation theory etc. study how to infer a function from collected data. This data, however, only describes the *mapping* itself. Therefore, strictly speaking, most machine learning algorithms do not learn a function, they only learn a part of it. While this distinction seems to be artificial for the two functions f and g mentioned a few lines above, consider the arcsin function. When sampling from this function and using for example a standard feedforward neural network to explain the data, one can also query the neural network at a value of 2. Here, the arcsin is not defined (real numbers), but the neural network would still predict a value, which could lead to many undesirable effects. Therefore, a neural network alone will never be able to represent the function as a whole properly. The same holds true for most function approximation techniques, since they are usually defined on the complete real hyper axis.

Apart from the problem of the unknown domain of a function, collecting samples from a data source does not necessarily mean that the underlying data generator is a function at all. As has been discussed in the related work section 2, this problem naturally occurs for example in inverse kinematics of robot arms. A dataset from such a robot arm could contain multiple joint configurations that lead to the same task value. If a regression method is utilized that is based on the squared error, the averaging over those multiple configurations that correspond to the same task value leads invalid results.

5.1.2. Problem Definition

First, the true forward model is formalized.

Definition 5.1 (Control Inputs)**.** The set

$$\mathcal{U} \subset \mathbb{R}^m \tag{5.1}$$

denotes the set of admissible control inputs. It is assumed that $\mathcal{U}$ is a compact set.

Definition 5.2 (True Reachable Set/Workspace)**.** The true reachable set or the workspace is denoted by

$$\mathcal{X} \subset \mathbb{R}^d. \tag{5.2}$$

It is assumed that $\mathcal{X}$ is compact or at least bounded as well.

Definition 5.3 (Redundancy)**.** This work assumes that $d \leq m$, which corresponds to redundancy in the sense that there are more control inputs than state dimensions ($d < m$).

Definition 5.4 (True Forward Model)**.** The *true* system

$$\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X} \tag{5.3}$$

is a function that statically and uniquely maps a control input $\mathbf{u} \in \mathcal{U}$ to a point $\mathbf{x} \in \mathcal{X}$ of the reachable set. More specifically, it describes how the real system uniquely, in absence of external disturbances, reacts to control inputs in the sense of a *static* mapping.

Remark 5.5. General forward models are usually functions from both the control input and the current state of the system (including external influences), either as state evolution equations or as differential equations. In contrast, this work only considers *static* and *state-independent* forward models.

Definition 5.6 (True Forward Model and Sets for Musculoskeletal Systems)**.** In the context of musculoskeletal systems as considered in this work, the control inputs are the normalized muscle stimulations, i.e.

$$\mathcal{U} = [0, 1]^m, \tag{5.4}$$

where $m \in \mathbb{N}$ is the total number of muscles of the system that can be stimulated independently. The reachable set $\mathcal{X} \subset \mathbb{R}^d$ is the set of all possible equilibrium configurations the system can reach, which could be, for example, the position of the hand of the arm model from section 3 or also its joint configuration. As has been discussed in section 3, it is assumed that each muscle stimulation $\mathbf{u} \in \mathcal{U}$ uniquely leads to a configuration $\mathbf{x} \in \mathcal{X}$ in equilibrium, independent from the start configuration. This is what is expressed by the true forward model $\widehat{\boldsymbol{\phi}}$. Therefore, the reachable set as $\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U})$ is also the set of all possible equilibrium positions.

Remark 5.7. Since the true forward model $\widehat{\boldsymbol{\phi}}$ is a static mapping, the dynamical behavior between equilibrium states is not represented by $\widehat{\boldsymbol{\phi}}$.

Remark 5.8 (Surjectiveness)**.** To simplify the discussion,

$$\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U}), \tag{5.5}$$

which means that the true forward model is surjective. This is not a limiting assumption, it just removes the need for technical details mentioned repeatedly.

After the true forward model has been formalized, the central description what this work considers as learning an inverse model can be stated.

Definition 5.9 (Inverse Model Learning)**.** For a true forward model $\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X}, \mathcal{U} \subset \mathbb{R}^m$, $\mathcal{X} \subset \mathbb{R}^d$, $d \leq m$, the inverse model learning problem is defined as learning both

1. the **reachable set** $\mathcal{X}$
2. the **inverse model** $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ such that

$$\forall_{\mathbf{x} \in \mathcal{X}} \ : \ \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) = \mathbf{x}. \tag{5.6}$$

While $\mathcal{X}$ is unknown, it is assumed that the set of admissible control inputs $\mathcal{U}$ is known. Further, the only information about the true forward model $\widehat{\boldsymbol{\phi}}$ available is the possibility to query it at a control input, which could involve a (costly) simulation or even a real robot experiment. In particular, no derivatives of $\widehat{\boldsymbol{\phi}}$ are available.

Proposition 5.10 (Existence of Inverse Model)**.** Under the assumptions of definition 5.4 and remark 5.8, such a (right) inverse $\boldsymbol{\pi}$ from definition 5.9 exists.

Proof. By defining $\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U})$ according to remark 5.8, $\widehat{\boldsymbol{\phi}}$ is trivially surjective by definition, which is sufficient for the existence of a right inverse. □

Proposition 5.11 (Inverse Model as Optimization Problem with Known True Forward Model)**.** Under the assumptions of definition 5.4 and remark 5.8, an inverse model $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ can be defined as the solution of

$$\boldsymbol{\pi}(\mathbf{x}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmin}} \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x} \right\| \tag{5.7}$$

for each $\mathbf{x} \in \mathcal{X}$, which always exists and has optimal value zero.

Proof. From proposition 5.10 it follows that $\forall_{\mathbf{x} \in \mathcal{X}} \exists_{\mathbf{u} \in \mathcal{U}} : \widehat{\boldsymbol{\phi}}(\mathbf{u}) = \mathbf{x}$. □

Defining an inverse this way has two main disadvantages. First, as stated in definition 5.9 of the inverse model learning problem, the true forward model is not available in terms of a mathematical expression. Therefore, solving (5.7) with derivative free methods is not only challenging methodologically, but also querying $\widehat{\boldsymbol{\phi}}(\mathbf{u})$ is very time consuming. The second problem, even if one would have $\widehat{\boldsymbol{\phi}}$ as a mathematical expression, is that (5.7) is a potentially non-convex optimization problem which would have to be solved for each desired target $\mathbf{x} \in \mathcal{X}$ again.

Instead, the goal of the present work is to represent the inverse model in a form that can be calculated efficiently for each desired target $\mathbf{x} \in \mathcal{X}$. In order to do so, the inverse model $\boldsymbol{\pi}$ is represented as a standard feedforward neural network, as described in section 4.2. In principle, other function approximators that are defined with a finite number of parameters could also be used. By using an output layer such that $\boldsymbol{\pi}$ maps into $\mathcal{U}$ and continuous activation functions, it holds $\boldsymbol{\pi} \in C(\mathcal{X}, \mathcal{U})$. The condition (5.6) is not yet in a manageable form for consideration of learning an inverse represented as a neural network, since one could not train the neural network on all $\mathbf{x} \in \mathcal{X}$.

If it is further assumed that the true forward model is continuous as well, the condition (5.6) can be reformulated in a way that is more suitable for learning.

Proposition 5.12. Let $\widehat{\boldsymbol{\phi}} \in C(\mathcal{U}, \mathcal{X})$ and $\boldsymbol{\pi} \in C(\mathcal{X}, \mathcal{U})$. Further assume that $\mathcal{X}$ does not contain isolated points. Then

$$\forall_{\mathbf{x} \in \mathcal{X}} \; : \; \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) = \mathbf{x} \tag{5.8}$$

$$\Leftrightarrow$$

$$\int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\| \mathrm{d}\mathbf{x} = 0. \tag{5.9}$$

Proof. $\Rightarrow$ is clear. To see $\Leftarrow$, assume that there exists a $\mathbf{x} \in \mathcal{X}$ such that

$$\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) \neq \mathbf{x}, \tag{5.10}$$

which implies

$$\left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\| > 0. \tag{5.11}$$

Since $\mathcal{X}$ does not contain isolated points and due to the fact that $\left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|$ is continuous, since $\widehat{\boldsymbol{\phi}} \circ \boldsymbol{\pi} \in C(\mathcal{X}, \mathcal{X})$ and norms are continuous, there exists a neighborhood $\mathcal{U}_\varepsilon(\mathbf{x})$ of $\mathbf{x}$ such that $\mathcal{U}_\varepsilon(\mathbf{x}) \cap \mathcal{X} \neq \emptyset$ and

$$\forall_{\tilde{\mathbf{x}} \in \mathcal{U}_\varepsilon(\mathbf{x}) \cap \mathcal{X}} \; : \; \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\tilde{\mathbf{x}})) - \tilde{\mathbf{x}} \right\| > 0. \tag{5.12}$$

Therefore, it follows

$$\int_{\mathcal{U}_\varepsilon(\mathbf{x}) \cap \mathcal{X}} \underbrace{\left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\tilde{\mathbf{x}})) - \tilde{\mathbf{x}} \right\|}_{>0} \mathrm{d}\tilde{\mathbf{x}} > 0, \tag{5.13}$$

which is a contradiction. Therefore

$$\forall_{\mathbf{x} \in \mathcal{X}} \; : \; \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\| = 0, \tag{5.14}$$

from which the proposition follows. □

Remark 5.13. Due to $\mathcal{U} = [0, 1]^m$ for a musculoskeletal system (definition 5.6), $\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U})$ (remark 5.8) and $\widehat{\boldsymbol{\phi}} \in C(\mathcal{U}, \mathcal{X})$, the assumption that $\mathcal{X}$ does not contain isolated points is automatically fulfilled.

However, inverses are, even for continuous true forward models, in general not continuous. Therefore, one cannot expect that the integral condition (5.9) can be fulfilled exactly for a continuous class of inverse models $\boldsymbol{\pi}$ such as neural networks. Therefore, the parameters $\mathbf{w} \in \mathcal{W}$ of the inverse model $\boldsymbol{\pi}$, i.e. the weights of the neural network in this case (section 4.2), would be chosen to minimize (5.9). Neural networks are also known for approximating discontinuities reasonably well, which justifies to use neural networks for representing the inverse model. This leads to the central (ideal) inverse model learning problem formulation.

Definition 5.14 (Ideal Inverse Model Learning Problem)**.** For the true forward model $\widehat{\boldsymbol{\phi}} \in C(\mathcal{U}, \mathcal{X})$, the parameters (weights) of the inverse model $\boldsymbol{\pi} \in C(\mathcal{X}, \mathcal{U})$ are determined as the solution of the optimization problem

$$\min_{\mathbf{w} \in \mathcal{W}} \int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) - \mathbf{x} \right\| \mathrm{d}\mathbf{x}. \tag{5.15}$$

Here one can also see that the reachable set $\mathcal{X}$ is essential to even *define* a proper metric to measure the quality of an inverse model.

Unfortunately, in this work, it is neither assumed to know the true reachable set $\mathcal{X}$, nor the true forward model $\widehat{\boldsymbol{\phi}}$ is available, as mentioned before, in terms of a mathematical expression. The only way to acquire knowledge about the true system is to evaluate $\widehat{\boldsymbol{\phi}}(\mathbf{u})$ for a specific control input $\mathbf{u} \in \mathcal{U}$. Therefore, the ideal inverse model learning formulation (5.15) is of little use.

To overcome this, the first idea would be to collect data from $\widehat{\boldsymbol{\phi}}$ and then learn a surrogate model $\boldsymbol{\phi}$. If the dataset is rich enough, one could set $\mathcal{X} \approx \boldsymbol{\phi}(\mathcal{U})$ and replace the true forward model $\widehat{\boldsymbol{\phi}}$ in (5.15) with the learned $\boldsymbol{\phi}$. Since a function approximator used for $\boldsymbol{\phi}$ provides also derivatives, (5.15) could be solved with gradient based training methods for neural networks as described in section 4.2. While seeming reasonable first, this approach has several shortcomings. First of all, as mentioned before, evaluating $\widehat{\boldsymbol{\phi}}(\mathbf{u})$ involves a costly numerical simulation or even a real robot experiment, which, together with the high dimensionality of the control input space $\mathcal{U}$, prohibits dense sampling in $\mathcal{U}$ to build a rich dataset. Especially if $\boldsymbol{\phi}$ is learned based on little data, one cannot expect that $\boldsymbol{\phi}$ approximates $\widehat{\boldsymbol{\phi}}$ everywhere in $\mathcal{U}$. At the same time, the optimization problem (5.15) would treat the forward model as exact and hence an unreasonable inverse could be learned.

Indeed, as it is shown in the experiments in section 6.8.1, simply using (5.15) with an iteratively learned $\boldsymbol{\phi}$ from little data compared to the size of the control input space is not sufficient and leads to bad performance. Furthermore, in addition to the fact that calculating $\boldsymbol{\phi}(\mathcal{U})$ is not only computationally demanding, for such a forward model, estimating the reachable set as $\mathcal{X} \approx \boldsymbol{\phi}$ also has misleading effects (section 6.7).

To summarize, learning an inverse model requires

- A surrogate objective to formulate the inverse model learning problem in a way that takes into account that the forward model is learned from little data
- A way to estimate the reachable set $\mathcal{X}$ that is also efficient to compute
- An exploration strategy to efficiently generate the data in the high dimensional input space in order to learn the forward model with the aim of reducing the error of the inverse model.

Solutions for each of these three requirements are derived in one unifying methodology in the following.

5.2. Learning the Inverse Model $\boldsymbol{\pi}$

During the learning process, in each iteration, a control input $\mathbf{u} \in \mathcal{U}$ is applied to the real system, which leads to a configuration $\mathbf{x} \in \mathcal{X}$. This data is added to a dataset as defined as follows.

Definition 5.15 (Dataset)**.** For $\mathbf{u}_i \in \mathcal{U}$ the value of the true forward model $\mathbf{x}_i = \widehat{\boldsymbol{\phi}}(\mathbf{u}_i) \in \mathcal{X}$ is stored in the dataset

$$\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^{n} . \tag{5.16}$$

The control input part is denoted by

$$\mathcal{D}_{\mathcal{U}} = \{\mathbf{u}_i\}_{i=1}^{n} \tag{5.17}$$

and the corresponding configurations as

$$\mathcal{D}_{\mathcal{X}} = \{\mathbf{x}_i\}_{i=1}^{n} = \widehat{\boldsymbol{\phi}}(\mathcal{D}_{\mathcal{U}}). \tag{5.18}$$

Remark 5.16 (Ordered Set and Indexing Notation)**.** For technical reasons these datasets $\mathcal{D}$, $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{X}}$ are ordered, i.e. one can think of

$$\mathcal{D} \subset \left(\mathbb{R}^m \times \mathbb{R}^d\right)^n, \qquad \mathcal{D}_{\mathcal{U}} \subset \left(\mathbb{R}^m\right)^n, \qquad \mathcal{D}_{\mathcal{X}} \subset \left(\mathbb{R}^d\right)^n. \tag{5.19}$$

Furthermore, to access the elements of these datasets, the following notation is used

$$\mathcal{D}(i) = (\mathbf{u}_i, \mathbf{x}_i), \ i = 1, \dots, n. \tag{5.20}$$

Similar to many software packages, indexing $(i : j)$, $1 \leq i < j \leq n$ means

$$\mathcal{D}(i : j) = \left((\mathbf{u}_o, \mathbf{x}_o)\right)_{o=i}^{j} . \tag{5.21}$$

For $\mathcal{D}_{\mathcal{U}}$ and $\mathcal{D}_{\mathcal{X}}$ the access is defined analogously. With this notation, one can also write

$$\mathcal{D}(i) = (\mathcal{D}_{\mathcal{U}}(i), \mathcal{D}_{\mathcal{X}}(i)) \tag{5.22}$$

to make the correspondence and the ordering more explicit.

Based on this currently available data, the forward model is learned.

Definition 5.17 (Learned Forward Model)**.** Let $k_j : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, $j = 1, \dots, d$ be d many, potentially different, (strictly) positive definite kernels with associated RKHSs H_{k_j}. Further, assume a mean prior $m_j \in \mathbb{R}$ and regularization parameter $(\sigma_j > 0)$ $\sigma_j \geq 0$ for each H_{k_j}. Given the dataset $\mathcal{D}$ from definition 5.15, the forward model $\boldsymbol{\phi} : \mathcal{U} \to \mathbb{R}^d$

$$\boldsymbol{\phi}(\mathbf{u}) = \left(\phi_1(\mathbf{u}), \dots, \phi_d(\mathbf{u})\right)^T \in \mathbb{R}^d \tag{5.23}$$

is learned, where each component ϕ_j is a solution of the kernel regression problem from proposition 4.11, 4.15 in H_{k_j} on the data $\mathcal{D}_j = \left\{(\mathbf{u}_i, (\mathbf{x}_i)_j)\right\}_{i=1}^{n}$.

Definition 5.18 (Parameterized Inverse Model)**.** The inverse model $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ that should be learned is represented as a parameterized function $\boldsymbol{\pi}(\,\cdot\,; \mathbf{w})$, where $\mathbf{w} \in \mathcal{W}$ are the parameters. More specifically, in this work, $\boldsymbol{\pi}$ is a feedforward neural, i.e. $\mathbf{w} \in \mathcal{W}$ are its weights (refer to section 4.2). Furthermore, it is assumed that $\boldsymbol{\pi}$ can also be evaluated on the set $\mathbb{X} \subset \mathbb{R}^d$ with $\mathcal{X} \subset \mathbb{X}$, which is automatically fulfilled for feedforward neural networks where $\mathbb{X} = \mathbb{R}^d$.

As discussed above, simply replacing $\widehat{\boldsymbol{\phi}}$ in (5.15) with this learned $\boldsymbol{\phi}$ and then optimizing the parameters $\mathbf{w}$ of $\boldsymbol{\pi}$ is not reasonable. Therefore, the goal is to derive an upper bound on the quality of the inverse model, when only a learned forward model is given. As a first step, the central error estimate for an arbitrary inverse model is derived under the assumption that the true forward model is an element of the RKHS.

Theorem 5.19 (Upper Bound on True Reaching Error / Error Estimate)**.** Assume that for the true forward model $\widehat{\boldsymbol{\phi}} = \left(\hat{\phi}_1, \ldots, \hat{\phi}_d\right)^T$ it holds $\hat{\phi}_i \in H_{k_i}$ with $\left\|\hat{\phi}_i\right\|_{H_{k_i}} < \infty$, $i = 1, \ldots, d$. Let $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ be an arbitrary inverse model according to definition 5.18, i.e. it can also be evaluated on $\mathbb{X}$ with $\mathcal{X} \subset \mathbb{X}$ meaning $\boldsymbol{\pi} : \mathbb{X} \subset \mathbb{R}^d \to \mathcal{U}$. Then the true reaching error in the $\mathcal{X}$-space for this arbitrary $\boldsymbol{\pi}$ can be bound by

$$\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| \le \varepsilon(\mathbf{x}), \tag{5.24}$$

where the error estimate is defined as

$$\varepsilon(\mathbf{x}) = \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\| + \sqrt{\sum_{i=1}^{d} \left\|\hat{\phi}_i\right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2}. \tag{5.25}$$

This bound holds for all $\mathbf{x} \in \mathbb{X}$, i.e. in particular for all $\mathbf{x} \in \mathcal{X}$. In the case of the present work where $\boldsymbol{\pi}$ is a standard feedforward neural network, $\mathbb{X} = \mathbb{R}^d$ (refer to definition 5.18), hence this bound is also true on complete $\mathbb{R}^d$.

Proof. By inserting a zero and using the triangle inequality, the true error of the inverse model at $\mathbf{x} \in \mathbb{X}$ can be estimated as

$$\begin{aligned} \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| &= \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) + \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| &(5.26)\\ &\le \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\| + \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))\right\|. &(5.27)\end{aligned}$$

The first term describes how well the inverse model is a right inverse to the learned forward model and can be computed, since all involved quantities are available as mathematical expressions. The second term is the error between the true and the learned forward model at $\boldsymbol{\pi}(\mathbf{x}) \in \mathcal{U}$. Using the kernel ridge regression error estimate (4.16) derived in section 4.1.1, the error between each component of the true forward model and the learned one can be upper bounded as

$$\left|\hat{\phi}_i(\boldsymbol{\pi}(\mathbf{x})) - \phi_i(\boldsymbol{\pi}(\mathbf{x}))\right| \le \left\|\hat{\phi}_i\right\|_{H_{k_i}} s_i(\boldsymbol{\pi}(\mathbf{x})) \tag{5.28}$$

for $i = 1, \ldots, d$. Plugging this into the definition of the Euclidean norm gives

$$\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}))\right\|^2 = \sum_{i=1}^{d} \left|\hat{\phi}_i(\boldsymbol{\pi}(\mathbf{x})) - \phi_i(\boldsymbol{\pi}(\mathbf{x}))\right|^2 \le \sum_{i=1}^{d} \left\|\hat{\phi}_i\right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2. \tag{5.29}$$

With this, the second term in (5.27) can be bound and the proposition follows. □

This theorem 5.19 is of crucial importance for the whole active inverse model learning framework and will come back several times in the following. As a first important part, this bound now allows to establish a connection between the quality of the inverse model, measured with a learned forward model, and the real quality, when measured with the true forward model. In order to do so, a small simple lemma is needed.

Lemma 5.20. For all $a, b \in \mathbb{R}$ it holds

$$(a+b)^2 \leq 2a^2 + 2b^2. \tag{5.30}$$

Proof.

$$(a+b)^2 = a^2 + 2ab + b^2 \leq 2a^2 + 2b^2 \Leftrightarrow 2ab \leq a^2 + b^2 \tag{5.31}$$
$$\Leftrightarrow (a-b)^2 \geq 0 \tag{5.32}$$

□

Using the derived error estimate (5.24) and lemma 5.20, it follows for the integral condition (5.9) of $\boldsymbol{\pi}$ being a right inverse of $\widehat{\boldsymbol{\phi}}$ that

$$\int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x} \leq \int_{\mathcal{X}} \varepsilon(\mathbf{x})^2 \, \mathrm{d}\mathbf{x} \tag{5.33}$$
$$\leq 2 \int_{\mathcal{X}} \left\| \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 + \sum_{i=1}^{d} \left\| \hat{\phi}_i \right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2 \, \mathrm{d}\mathbf{x}. \tag{5.34}$$

This upper bound does not require the true forward model in terms of a mathematical expression, only its RKHS norm. If further an upper bound on this norm is assumed, i.e. $\left\| \hat{\phi}_i \right\|_{H_{k_i}} \leq \beta_i$, $i = 1, \dots, d$, the quality of the inverse model can be estimated as

$$\int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x} \leq 2 \int_{\mathcal{X}} \left\| \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2 \, \mathrm{d}\mathbf{x}, \tag{5.35}$$

where no quantity of the true forward model involved. Of course the tightness of this bound depends on the tightness of $\left\| \hat{\phi}_i \right\|_{H_{k_i}} \leq \beta_i$. This derivation has established the desired connection in the sense that the quality of the inverse model for the true system is measured based on the learned forward model.

Theorem 5.21 (Upper Bound on Quality of Inverse Model)**.** Assume that $\left\| \hat{\phi}_i \right\|_{H_{k_i}} \leq \beta_i$ for $i = 1, \dots, d$. Then an upper bound on the quality (5.15) of the inverse model $\boldsymbol{\pi}$ is given by

$$\int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x} \leq 2 \int_{\mathcal{X}} \left\| \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2 \, \mathrm{d}\mathbf{x}. \tag{5.36}$$

As discussed before, there are potentially infinitely many inverse models. One might be interested in influencing which inverse would be learned. To this end, the following inverse regularizer is introduced.

Definition 5.22 (Inverse Regularizing Function)**.** A positive, monotone function $l : \mathcal{U} \to \mathbb{R}^+$ is called an inverse regularizing function.

Example 5.23. The choice

$$l(\mathbf{u}) = \left\| \mathbf{u} \right\|^2 \tag{5.37}$$

would encourage to learn inverses with low control inputs.

Finally, everything can be plugged together, which leads to the central inverse model learning optimization problem.

Definition 5.24 (Inverse Model Learning Optimization Problem Formulation)**.** The inverse model $\boldsymbol{\pi}$ is learned by optimizing

$$\min_{\mathbf{w}\in\mathcal{W}} \int_{\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))^2 + \eta\, l(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))\,\mathrm{d}\mathbf{x}. \tag{5.38}$$

Intuitively, the optimization problem tries to find an inverse to the learned forward model, while, which is expressed with the second term in this objective, staying close to the data.

5.3. Learning the Reachable Set $\mathcal{X}$

The integral formulation (5.38) in the central inverse model learning optimization problem (definition 5.24) assumed that the true reachable set $\mathcal{X}$ is known. In some situations it might be reasonable to assume this, for example if $\mathcal{X}$ is a hyperrectangle with a priori known limits in each dimension. However, the reachable set has, in general, a more complex shape, which is *not* known in terms of a mathematical description. As derived in proposition 5.12 and as can be seen in the integral formulation (5.38), the reachable set is essential to define the objective to learn the inverse model at all. Therefore, this work considers estimating the reachable set $\mathcal{X}$ as an inherent part of learning an inverse model, which should not be treated separately.
The requirements for an estimate of $\mathcal{X}$ are

- A computationally feasible estimation
- A clear representation which easily allows to decided whether a point belongs to the workspace or not
- The estimated set should not overestimate the real set unreasonably.

As a first idea, one could think of using the learned surrogate forward model $\boldsymbol{\phi}$ and then approximate the reachable set as

$$\mathcal{X} \approx \boldsymbol{\phi}(\mathcal{U}), \tag{5.39}$$

which could, in theory, be calculated, since it is assumed that the control input space $\mathcal{U}$ is known. This, however, has two main disadvantages. Similar to the discussion in section 5.1.2, (5.39) does not take into account that the learned forward model does not approximate $\widehat{\boldsymbol{\phi}}$ everywhere in $\mathcal{U}$ precisely, probably leading to a highly overestimated reachable set. But even if one could sample $\widehat{\boldsymbol{\phi}}$ densely enough to build a dataset for learning $\boldsymbol{\phi}$ everywhere in $\mathcal{U}$, it is not straightforward how the reachable set would then be represented based on $\boldsymbol{\phi}$ in a computationally efficient way, since for example asking whether a point $\mathbf{x}$ is part of the workspace would require to decided whether $\boldsymbol{\phi}(\mathbf{u}) - \mathbf{x} = 0$ has a solution $\mathbf{u} \in \mathcal{U}$, which is a non-trivial problem in its own. To overcome these issues, the main idea is to again utilize the error estimate from section 5.2 to estimate $\mathcal{X}$ based on the current learned forward and inverse model.

It is clear that one cannot expect to approximate $\mathcal{X}$ exactly by only learning from sampled data. Therefore, the following defines a notion of an approximate reachable set.

Definition 5.25 (True Reachable Set with Error Certainty)**.** For the true reachable set $\mathcal{X} \subset \mathbb{R}^d$, the set

$$\hat{\mathcal{X}}_c = \left\{ \mathbf{x} \in \mathbb{R}^d : \operatorname{dist}(\mathbf{x}, \mathcal{X}) < c \right\} \tag{5.40}$$

defines the true reachable set with error certainty $c > 0$.

Intuitively, $\hat{\mathcal{X}}_c$ is the set $\mathcal{X}$ enlarged by c, meaning that it overestimates $\mathcal{X}$ exactly by the value of c. Figure 5.1 visualizes $\hat{\mathcal{X}}_c$ in comparison to $\mathcal{X}$. With this, the present work defines the problem of estimating the workspace as approximating $\hat{\mathcal{X}}_c$ for a given c.

Definition 5.26 (Reachable Set Estimation Problem)**.** For a given error certainty $c > 0$, estimating the true reachable set $\mathcal{X} \subset \mathbb{R}^d$ is defined finding a set $\mathcal{X}_c \subset \mathbb{R}^d$, such that

$$\mathcal{X}_c \subset \hat{\mathcal{X}}_c. \tag{5.41}$$

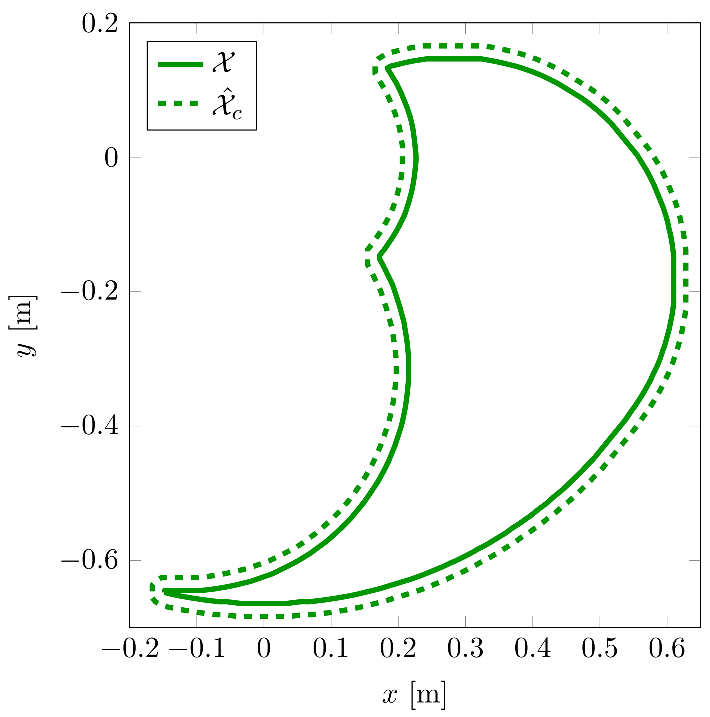


Figure 5.1: Visualization of the true reachable set with error certainty $\hat{\mathcal{X}}_c$ for $c = 0.02$ of the simulated arm model from section 3.7.

Utilizing the error estimate of the inverse model from theorem 5.19, the following theorem provides a solution to this estimation problem based on the current learned forward and inverse model.

Theorem 5.27 (Estimated Reachable Set)**.** The reachable set with error certainty $c > 0$ for the current learned forward and inverse model is determined by

$$\mathcal{X}_c = \{\mathbf{x} \in \mathbb{X} : \varepsilon(\mathbf{x}) < c\}\,, \tag{5.42}$$

for which it holds

$$\mathcal{X}_c \subset \hat{\mathcal{X}}_c. \tag{5.43}$$

In the case of $\boldsymbol{\pi}$ being a neural network, according to definition 5.18, $\mathbb{X} = \mathbb{R}^d$.

Proof. It is $\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) \in \mathcal{X}$ for every $\mathbf{x} \in \mathbb{X}$, since $\boldsymbol{\pi}(\mathbb{X}) \subset \mathcal{U}$. Therefore,

$$\operatorname{dist}(\mathbf{x}, \mathcal{X}) = \inf_{\mathbf{x}' \in \mathcal{X}} \|\mathbf{x}' - \mathbf{x}\| \leq \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| \tag{5.44}$$

by definition of dist and the infimum. By using the error estimate (5.24) from theorem 5.19, it follows

$$\operatorname{dist}(\mathbf{x}, \mathcal{X}) \leq \varepsilon(\mathbf{x}). \tag{5.45}$$

Therefore, $\varepsilon(\mathbf{x}) < c$ implies $\operatorname{dist}(\mathbf{x}, \mathcal{X}) < c$ and hence the proposition. □

Remark 5.28 (Interpretation)**.** Defining the domain of the inverse model as $\mathcal{X}_c$ has the interpretation that for every $\mathbf{x} \in \mathcal{X}_c$ it holds

$$\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| < c, \tag{5.46}$$

meaning that the true reached point has a guaranteed accuracy of c.

Remark 5.29 (Computational Feasibility and Representation)**.** Checking whether a point $\mathbf{x}$ belongs to the estimated workspace reduces to calculate $\varepsilon(\mathbf{x})$, which can be done efficiently (see section 5.5.2 for a discussion of the computational complexity of the proposed framework). Furthermore, since the $\mathcal{X}$-space is assumed to be low dimensional, $\mathcal{X}_c$ can also be represented by dense grid evaluation of ε.

Remark 5.30. Theorem 5.27 ensures that the estimated set never overestimates the true reachable set with the specified error certainty. Therefore, the third requirement for the estimation is fulfilled. However, it is not guaranteed that for a given c the set $\mathcal{X}_c$ is nonempty, which means that $\mathcal{X}_c$ could also *underestimate* the true reachable set significantly.

Remark 5.31 (Trade-Off)**.** It is clear that there is a trade-off for the parameter c. If c is large, then the true reachable set is typically overestimated. If c is small, the estimation is accurate, but probably too conservative and hence underestimated.

One problem with estimating the reachable set according to theorem 5.27 is that it relies on the quality of the current learned inverse model $\boldsymbol{\pi}$. Improving the quality of the inverse model, in turn, also depends on the quality of the estimated reachable set $\mathcal{X}_c$. An alternative to define the estimated reachable set without the inverse model would be the following.

Proposition 5.32 (Estimated Reachable Set Forward Model Based)**.** For $c > 0$ define

$$\mathcal{X}_c^{\mathcal{U}} = \left\{ \boldsymbol{\phi}(\mathbf{u}) : \mathbf{u} \in \mathcal{U} \wedge \sum_{i=1}^{d} \left\| \hat{\phi}_i \right\|_{H_{k_i}}^2 s_i(\mathbf{u})^2 < c^2 \right\}, \tag{5.47}$$

for which it holds

$$\mathcal{X}_c^{\mathcal{U}} \subset \hat{\mathcal{X}}_c. \tag{5.48}$$

Proof. This proof has a similar idea than the proof of theorem 5.27. For notational simplicity, abbreviate

$$\tilde{s}(\mathbf{u})^2 = \sum_{i=1}^{d} \left\| \hat{\phi}_i \right\|_{H_{k_i}}^2 s_i(\mathbf{u})^2. \tag{5.49}$$

Assume that $\mathbf{x} \in \mathcal{X}_c^{\mathcal{U}}$. This implies $\exists_{\mathbf{u}^* \in \mathcal{U}} : \boldsymbol{\phi}(\mathbf{u}^*) = \mathbf{x}$ and $\tilde{s}(\mathbf{u}^*) < c$. Since $\widehat{\boldsymbol{\phi}}(\mathbf{u}) \in \mathcal{X}$ for all $\mathbf{u} \in \mathcal{U}$, it holds for this specific $\mathbf{u}^*$ that

$$\operatorname{dist}(\mathbf{x}, \mathcal{X}) = \inf_{\mathbf{x}' \in \mathcal{X}} \|\mathbf{x}' - \mathbf{x}\| \leq \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}^*) - \mathbf{x} \right\| = \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}^*) - \boldsymbol{\phi}(\mathbf{u}^*) \right\| \leq \tilde{s}(\mathbf{u}^*) < c, \tag{5.50}$$

which means that $\mathbf{x} \in \hat{\mathcal{X}}_c$. □

Approximating $\hat{\mathcal{X}}_c$ with $\mathcal{X}_c^{\mathcal{U}}$ has the advantage that it only relies on the forward model and not the quality of the inverse model. Furthermore, the definition of $\mathcal{X}_c^{\mathcal{U}}$ takes into account that the forward model is learned on potentially little data. As shown in the experiments (see section 6.7), $\mathcal{X}_c^{\mathcal{U}}$ leads indeed to good results for estimating the reachable set. However, one main disadvantage remains, namely that it is not clear how one could check efficiently whether a point $\mathbf{x}$ is an element of $\mathcal{X}_c^{\mathcal{U}}$. While it is for example possible to calculate $\boldsymbol{\phi}\left(\tilde{\mathcal{U}}\right)$ on a discretized version $\tilde{\mathcal{U}}$ of $\mathcal{U}$ with modern hardware in finite time, the computation time is far away from being practical. In comparison, checking if a point is an element of $\mathcal{X}_c$ is, as discussed above, easily realizable.

5.4. Active Exploration Strategy

So far, both the inverse model learning optimization problem (5.38) and the estimation of the reachable set (5.42) assumed that the data $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^n$ (definition 5.15) for learning the forward model $\boldsymbol{\phi}$ is already collected. As mentioned several times in this work before, the high dimensionality of $\mathcal{U}$ prohibits dense sampling of the true system to generate this dataset. Therefore, as a final step, a methodology is developed to collect the data for the forward model sample efficiently. However, the goal of the active inverse model learning framework is not to learn a particularly good forward model, instead, the interest is in learning a good inverse model. This means that the data should be collected in a way that is particularly useful for determining a better inverse model. As discussed in section 5.3, knowing the reachable set is essential to even define the quality of an inverse model. Therefore, the exploration strategy should aim at selecting points such that they become dense in the complete unknown reachable set $\mathcal{X}$. This can be expressed in terms of the so-called fill-distance.

Definition 5.33 (Fill Distance)**.** For compact $\mathcal{X}$ the fill distance is defined as

$$\max_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\|. \tag{5.51}$$

Intuitively, the fill distance is the radius of the largest ball with center in $\mathcal{X}$ that does not contain any already sampled data points from $\mathcal{D}_\mathcal{X}$. An ideal active exploration strategy would therefore select a new $\mathbf{x}^*$ that maximizes the fill distance. In most active learning methodologies one could maximize the so-called acquisition function, which in this case would be the fill distance, and then sample the true system at the location $\mathbf{x}^*$ that maximizes the acquisition function. However, in this work, one can only query control inputs $\mathbf{u}$ and not directly points in the $\mathcal{X}$-space, in which the acquisition function is defined, since finding a $\mathbf{u}$ such that $\widehat{\boldsymbol{\phi}}(\mathbf{u}) = \mathbf{x}^*$ would require an already perfectly known inverse. This fact is made more explicit with the following, where a notion of the true fill distance of the system, as considered in this work, is defined.

Proposition 5.34 (True Fill Distance)**.** For the true forward model $\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X}$ the true fill distance

$$\max_{\mathbf{u}\in\mathcal{U}} \left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i \right\| \right) \tag{5.52}$$

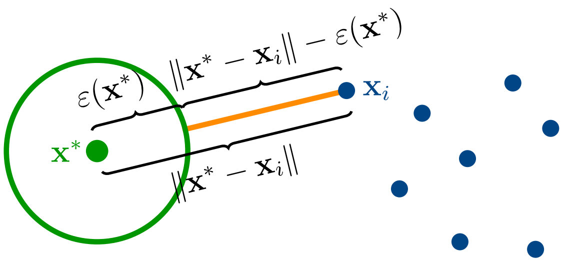
is well-defined.

Proof. Since $\mathcal{U}$ is assumed to be compact (definition 5.1) and $\widehat{\boldsymbol{\phi}}$ be continuous (proposition 5.12), the maximum over $\mathbf{u} \in \mathcal{U}$ is well-defined. □

The true fill distance (5.52) measures how large unexplored parts in the true $\mathcal{X}$ are. Since this is defined in terms of the (high dimensional) control input space and the true forward model, it is not possible to use (5.52) as an exploration criteria. Therefore, the goal of this section is to derive a computationally feasible lower bound on the true fill distance which is based on the current learned forward and inverse model, while taking into account that the learned models are not perfect.

Proposition 5.35 (Lower Bound on Real Distance to Data)**.** Given an inverse model $\boldsymbol{\pi} : \mathbb{X} \to \mathcal{U}$ with $\mathcal{X} \subset \mathbb{X} \subset \mathbb{R}^d$, the real distance of a desired target $\mathbf{x}^* \in \mathbb{X}$ reached with the inverse model $\boldsymbol{\pi}$ to the data points is lower bounded by

$$\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x}^* - \mathbf{x}_i\| - \varepsilon(\mathbf{x}^*) \leq \min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*)) - \mathbf{x}_i \right\|. \tag{5.53}$$



$\|\mathbf{x}^* - \mathbf{x}_i\| - \varepsilon(\mathbf{x}^*)$
$\varepsilon(\mathbf{x}^*)$
$\mathbf{x}_i$
$\mathbf{x}^*$
$\|\mathbf{x}^* - \mathbf{x}_i\|$

Figure 5.2: Visualization of the lower bound on the real distance to the data from proposition 5.35.

Proof. By inserting a zero and using the triangle inequality, it follows for all $\mathbf{x}^* \in \mathbb{X}$

$$\begin{aligned} \|\mathbf{x}^* - \mathbf{x}_i\| &= \left\|\mathbf{x}^* - \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*)) + \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*)) - \mathbf{x}_i\right\| && (5.54) \\ &\leq \left\|\mathbf{x}^* - \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*))\right\| + \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*)) - \mathbf{x}_i\right\| && (5.55) \\ &\leq \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*)) - \mathbf{x}_i\right\| + \varepsilon(\mathbf{x}^*), && (5.56) \end{aligned}$$

where the error estimate (5.24) of theorem 5.19 is used. Bringing $\varepsilon(\mathbf{x}^*)$ to the other side and taking the minimum over $\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}$, the proposition follows. □

This lower bound (5.53), which is visualized in figure 5.2, holds on $\mathbb{X}$, i.e. in the case of the present work where $\boldsymbol{\pi}$ is a neural network it holds on complete $\mathbb{R}^d$. Therefore, since additionally

$$\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbb{X})) \subset \widehat{\boldsymbol{\phi}}(\mathcal{U}) \tag{5.57}$$

it follows

$$\begin{aligned} \sup_{\mathbf{x}\in\mathbb{X}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) &\leq \sup_{\mathbf{x}\in\mathbb{X}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}_i\right\| \right) && (5.58) \\ &\leq \max_{\mathbf{u}\in\mathcal{U}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \left\|\widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i\right\| \right). && (5.59) \end{aligned}$$

This has proven the following important result.

Theorem 5.36 (Lower Bound on Real Fill Distance)**.**

$$\sup_{\mathbf{x}\in\mathbb{X}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \leq \max_{\mathbf{u}\in\mathcal{U}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \left\|\widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i\right\| \right) \tag{5.60}$$

Looking at figure 5.2, maximizing this lower bound on the real fill distance has the intuitive interpretation of finding a target point in the $\mathcal{X}$-space, whose error ball has the largest distance to all already sampled data points. This way, it is explicitly taken into account that it is not expected to reach a desired target with the current learned inverse model exactly.

Although, as mentioned above, this bound (5.60) is true on complete $\mathbb{X}$ or $\mathbb{R}^d$, it is not guaranteed that the maximum in 5.60 is attained on $\mathbb{X}$. There are several possibilities, which will be presented in the following, to modify $\mathbb{X}$ such that the maximum in (5.60) is attained and can therefore be used as an exploration strategy.

Ideally, one would use the exploration criteria

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \tag{5.61}$$

obtained by maximizing over the true reachable set $\mathcal{X}$, which would be well-defined, since $\mathcal{X}$ is assumed to be compact. Since, again, the true reachable set is not known a priori, one approach is to replace $\mathcal{X}$ in (5.61) with the closure of the current estimated reachable set, leading to

$$\mathbf{x}^* = \underset{\mathbf{x}\in\overline{\mathcal{X}_c}}{\operatorname{argmax}} \left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x}-\mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right), \tag{5.62}$$

which also exists if $\mathcal{X}_c$ is bounded, which is, except in pathological cases, always fulfilled.

However, exploring only inside the current estimated reachable set $\mathcal{X}_c$ can be too conservative, since the exploration strategy should also try to explore outside of $\mathcal{X}_c$ to expand the reachable set. In order to overcome this, two further possibilities are considered. First, the idea is to find a compact subset of $\mathbb{X} \subset \mathbb{R}^d$. A simple choice for such a set would be a hyperinterval.

Definition 5.37 (Bounding Box of True Reachable Set)**.** The set

$$\boxed{\mathcal{X}} = [l_1, u_1] \times \cdots \times [l_d, u_d] \tag{5.63}$$

with $l_i, u_i \in \mathbb{R}$, $l_i < u_i$, $i = 1, \ldots, d$ such that

$$\mathcal{X} \subset \boxed{\mathcal{X}} \tag{5.64}$$

is called the bounding box of the true reachable set $\mathcal{X}$, which is a compact set.

Although it corresponds to some kind of prior knowledge, finding such a bounding box is usually possible. With this bounding box, the exploration criteria

$$\mathbf{x}^* = \underset{\mathbf{x}\in\boxed{\mathcal{X}}}{\operatorname{argmax}} \left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x}-\mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \tag{5.65}$$

is well-defined. An alternative idea to $\boxed{\mathcal{X}}$ is to introduce a trust region around the already sampled data points in the $\mathcal{X}$-space. This leads to the criteria

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmax}} \left(\min_{\mathbf{x}_i\in\mathcal{D}_\mathcal{X}} \|\mathbf{x}-\mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \tag{5.66a}$$

$$\text{s.t.} \quad \operatorname{dist}(\mathbf{x}, \mathcal{D}_\mathcal{X}) \le d_\mathcal{X}, \tag{5.66b}$$

where $d_\mathcal{X} > 0$ denotes the maximum distance of the candidate points to the data points in $\mathcal{D}_\mathcal{X}$. Usually, one chooses $d_\mathcal{X} > c$ to enable the exploration criteria to choose target points outside the current estimated reachable set $\mathcal{X}_c$ (otherwise one could use (5.62)). Since $\operatorname{dist}(\mathbf{x}, \mathcal{D}_\mathcal{X}) \le d_\mathcal{X}$ defines a compact feasible set, the criteria (5.66) is also well-defined.

5.5. Complete Active Inverse Model Learning Methodology

In this section, the complete active inverse model learning framework is summarized as well as all assumptions are stated again. Furthermore, there are still some details missing before the methodology can be implemented in concrete, practical algorithms, which is the goal of this section as well.

Theorem 5.38 (Active Inverse Model Learning)**.** *Let $\mathcal{U} \subset \mathbb{R}^m$ be the compact set of control inputs without any isolated points. The true forward model $\widehat{\boldsymbol{\phi}} : \mathcal{U} \to \mathcal{X}$ is a continuous, surjective mapping from the control input space to the reachable set $\mathcal{X} = \widehat{\boldsymbol{\phi}}(\mathcal{U}) \subset \mathbb{R}^d$. Further assume that the true forward model $\widehat{\boldsymbol{\phi}} = \left(\hat{\phi}_1, \dots, \hat{\phi}_d\right)^T$ fulfills $\hat{\phi}_i \in H_{k_i}$ with $\left\|\hat{\phi}_i\right\|_{H_{k_i}} < \infty$, $i = 1, \dots, d$ for the reproducing kernel Hilbert spaces H_{k_i} corresponding to the strictly positive definite kernels $k_i : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$. Let $\beta_i > 0$ be given such that $\left\|\hat{\phi}_i\right\|_{H_{k_i}} \leq \beta_i$, $i = 1, \dots, d$. The inverse model $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{U}$ is assumed to be a parameterized class of continuous functions, e.g. standard feedforward neural networks with continuous activation functions, which can also be evaluated on the set $\mathbb{X}$ with $\mathcal{X} \subset \mathbb{X} \subset \mathbb{R}^d$. Based on the dataset $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^n$, consisting of control inputs $\mathbf{u}_i \in \mathcal{U}$ that have been applied to the true forward model leading to the configuration $\mathbf{x}_i = \widehat{\boldsymbol{\phi}}(\mathbf{u}_i) \in \mathcal{X}$, the forward model $\boldsymbol{\phi} : \mathcal{U} \to \mathbb{R}^d$, $\boldsymbol{\phi} = (\phi_1, \dots, \phi_d)^T$ is learned, where each component ϕ_j, $j = 1, \dots, d$ is a solution of the kernel regression problem in the RKHS H_{k_j} with mean prior $m_j \in \mathbb{R}$ as well as regularization parameter $\sigma_j \geq 0$ on the dataset $\mathcal{D}_j = \{(\mathbf{u}_i, (\mathbf{x}_i)_j)\}_{i=1}^n$.*

The parameters of the inverse model, denoted by $\mathbf{w} \in \mathcal{W}$ are then trained to minimize the inverse model learning optimization problem

$$\min_{\mathbf{w} \in \mathcal{W}} \int_{\mathcal{X}_c} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^d \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w}))^2 + \eta \, l(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) \, \mathrm{d}\mathbf{x}, \tag{5.67}$$

where $\mathcal{X}_c$ is the estimated reachable set with error certainty $c > 0$, given by

$$\mathcal{X}_c = \{\mathbf{x} \in \mathbb{X} : \varepsilon(\mathbf{x}) < c\}. \tag{5.68}$$

This estimated reachable set fulfills

$$\mathcal{X}_c \subset \left\{\mathbf{x} \in \mathbb{R}^d : \operatorname{dist}(\mathbf{x}, \mathcal{X}) < c\right\} = \hat{\mathcal{X}}_c, \tag{5.69}$$

i.e. it is guaranteed that the true reachable set is never overestimated more than specified by the error certainty c. The ε term is the error estimate, defined as

$$\varepsilon(\mathbf{x}) = \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\| + \sqrt{\sum_{i=1}^d \left\|\hat{\phi}_i\right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2} \tag{5.70}$$

as well as

$$\varepsilon_\beta(\mathbf{x}) = \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\| + \sqrt{\sum_{i=1}^d \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}))^2}, \tag{5.71}$$

which allows to upper bound the true reaching error

$$\left\|\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x}\right\| \leq \varepsilon(\mathbf{x}) \leq \varepsilon_\beta(\mathbf{x}), \tag{5.72}$$

when the learned inverse $\boldsymbol{\pi}$ is used to predict the control input $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ for a desired target $\mathbf{x} \in \mathbb{X}$. With this error estimate, the quality of the inverse model, i.e. how well the learned $\boldsymbol{\pi}$ is a right inverse to the true forward model $\widehat{\boldsymbol{\phi}}$ can be upper bounded as

$$\int_{\mathcal{X}_c} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x})) - \mathbf{x} \right\|^2 \mathrm{d}\mathbf{x} \leq \int_{\mathcal{X}_c} \varepsilon(\mathbf{x})^2 \,\mathrm{d}\mathbf{x} \leq \int_{\mathcal{X}_c} \varepsilon_\beta(\mathbf{x})^2 \,\mathrm{d}\mathbf{x}, \tag{5.73}$$

which does not require $\widehat{\boldsymbol{\phi}}$.

To generate the data $\mathcal{D}$ for the forward model, the whole framework is an iterative process. Starting with a manually specified initial control input and the corresponding resulting configuration when applied to the real forward model, the inverse model is trained on this single data point only. All subsequent data points in the $n+1$-th iteration are generated by selecting a target $\mathbf{x}^*_{n+1}$ in the $\mathcal{X}$-space, the current learned inverse model is used to predict a control input $\mathbf{u}_{n+1} = \boldsymbol{\pi}(\mathbf{x}^*_{n+1})$, which is applied to the true forward model, leading to $\mathbf{x}_{n+1} = \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*_{n+1}))$. This new data point is added to the dataset

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{u}_{n+1}, \mathbf{x}_{n+1})\}\,. \tag{5.74}$$

Then the forward model $\boldsymbol{\phi}$ is learned on this new dataset, the new reachable set $\mathcal{X}_c$ is estimated and finally the parameters of the inverse model $\boldsymbol{\pi}$ are adjusted according to (5.67) to take the new information about the system into account.

To select the exploration targets $\mathbf{x}^*$, the exploration strategies

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x} \in \overline{\mathcal{X}_c}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \tag{5.75}$$

or

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x} \in \mathbb{R}^d} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \tag{5.76a}$$

$$\text{s.t.} \quad \operatorname{dist}(\mathbf{x}, \mathcal{D}_\mathcal{X}) \leq d_\mathcal{X}, \tag{5.76b}$$

are well-defined. Both try to maximize a lower bound on the true fill distance of the system, i.e. it holds for the exploration target $\mathbf{x}^*$ determined by (5.75)

$$\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x}^* - \mathbf{x}_i\| - \varepsilon(\mathbf{x}^*) = \max_{\mathbf{x} \in \overline{\mathcal{X}_c}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \leq \max_{\mathbf{u} \in \mathcal{U}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i \right\| \right) \tag{5.77}$$

or by (5.76)

$$\begin{aligned} \min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x}^* - \mathbf{x}_i\| - \varepsilon(\mathbf{x}^*) = & \max_{\mathbf{x} \in \mathbb{R}^d} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon(\mathbf{x}) \right) \leq \max_{\mathbf{u} \in \mathcal{U}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\mathbf{u}) - \mathbf{x}_i \right\| \right). \\ & \text{s.t.} \quad \operatorname{dist}(\mathbf{x}, \mathcal{D}_\mathcal{X}) \leq d_\mathcal{X} \end{aligned} \tag{5.78}$$

5.5.1. Practical Considerations

In practice, calculating 5.67 with the estimated workspace is computationally infeasible, since the integral over $\mathcal{X}_c$ cannot be expressed in closed form. However, since it is assumed that the workspace is low dimensional, the integral can efficiently be approximated by a rectangular rule. In order to do so, a discretized version of the bounding box from definition 5.37 is defined.

Definition 5.39 (Discretized Bounding Box of True Reachable Set)**.** The set

$$\widetilde{\boxed{\mathcal{X}}} = \left\{l_{11}, \ldots, u_{1n_{h_1}}\right\} \times \cdots \times \left\{l_{d1}, \ldots, u_{dn_{h_d}}\right\} \tag{5.79}$$

is the discretized version of the bounding box $\boxed{\mathcal{X}}$ of the true reachable set $\mathcal{X}$ with n_{h_i} many equidistant points per dimension $i = 1, \ldots, d$ and $l_{i1} = l_i$, $u_{in_{h_1}} = u_i$ for l_i, u_i, $i = 1, \ldots, d$ from definition 5.37.

With this, the discretized version of the estimated reachable set can be defined as

$$\tilde{\mathcal{X}}_c = \left\{\mathbf{x} \in \widetilde{\boxed{\mathcal{X}}} : \varepsilon(\mathbf{x}) < c\right\}, \tag{5.80}$$

which can also be calculated by evaluating ε on the finite $\widetilde{\boxed{\mathcal{X}}}$. Especially in the beginning of the exploration, $\mathcal{X}_c$ and hence also $\tilde{\mathcal{X}}_c$ is often empty. Therefore, the already sampled data points $\mathcal{D}_\mathcal{X}$ are included to the estimated workspace. This is also reasonable, since actual reached data points are guaranteed to be part of the true workspace. These considerations lead to the final, practical optimization problem for learning the inverse model.

Definition 5.40 (Practical Inverse Model Learning Optimization Problem)**.** The weights of the inverse model are obtained through minimization of

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_{\mathbf{x} \in \tilde{\mathcal{X}}_c \cup \mathcal{D}_\mathcal{X}} \left(\|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w}))^2 + \eta\, l(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) \right). \tag{5.81}$$

All involved quantities are finite and known in terms of mathematical expressions including their derivatives. Thus, the objective (5.81) and its gradient can be calculated.

In practice, it turned out that sometimes the exploration strategy suffers from the problem that it tries to reach the same target point repeatedly. Assume that this target is denoted as $\mathbf{x}_T^*$. Since the corresponding muscle stimulations are generated through the current learned inverse model, it sometimes happens that the two predicted muscle stimulations $\boldsymbol{\pi}^{(i)}(\mathbf{x}_T^*) = \mathbf{u}_1$, $\boldsymbol{\pi}^{(i+1)}(\mathbf{x}_T^*) = \mathbf{u}_2$ for this same target are also nearly identical. The superscript for $\boldsymbol{\pi}$ indicates that the network is updated between the two predictions. Having two nearly identical muscle stimulations in the dataset for learning the forward model deteriorates the forward model at this location, since the condition number of the kernel matrix mainly depends on the separation distance of the data points [45]. To address this problem, the robustness of the exploration strategy can be further improved by restricting the exploration strategy to only search for target points that have a minimum distance to a certain number of old selected target points. More formally, this leads to the two practical exploration strategies

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x} \in \tilde{\mathcal{X}}_c} \left(\min_{\mathbf{x}_i \in \mathcal{D}_\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon_\beta(\mathbf{x}) \right) \tag{5.82a}$$

$$\text{s.t.} \quad \operatorname{dist}(\mathbf{x}, \mathcal{D}_{\mathrm{X}_T}(n - n_{\mathcal{X}_T} : n)) \geq d_{\mathcal{X}_T} \tag{5.82b}$$

in the estimated reachable set $\tilde{\mathcal{X}}_c$ (discretized version) or with trust region around the collected data points

$$\mathbf{x}^* = \underset{\mathbf{x} \in \boxed{\widetilde{\mathcal{X}}}}{\operatorname{argmax}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon_\beta(\mathbf{x}) \right) \tag{5.83a}$$

$$\text{s.t.} \quad \operatorname{dist}(\mathbf{x}, \mathcal{D}_{\mathcal{X}}) \leq d_{\mathcal{X}} \tag{5.83b}$$

$$\operatorname{dist}(\mathbf{x}, \mathcal{D}_{\mathrm{X}_T}(n - n_{\mathcal{X}_T} : n)) \geq d_{\mathcal{X}_T}, \tag{5.83c}$$

where $d_{\mathcal{X}_T} > 0$ denotes the minimum distance to a number of $n_{\mathcal{X}_T}$ old target points and

$$\mathcal{D}_{\mathcal{X}_T} = \{\mathbf{x}_i^*\}_{i=1}^n \tag{5.84}$$

denotes the set of all chosen targets by the exploration strategy. Please note that this modification is a condition on the exploration targets and not the observed values, which are still all included in the dataset. Therefore, the actual reached points can still be closer than $d_{\mathcal{X}_T}$. It has also to be stated that in most situations the methodology is successful without this modification, but not as reliable. The error estimate ε in (5.82) and (5.83) has been replaced by ε_β, since this only requires an upper bound on the RKHS norm of the true forward model and not the correct actual value, which is more practical. Of course, with ε_β all derived properties of the exploration strategy still hold.

Such max-min optimization problems as those of the exploration strategies are usually difficult to solve. However, since the $\mathcal{X}$-space is assumed to be low dimensional and the optimization problems are already defined on a discretized version of it (the set $\boxed{\widetilde{\mathcal{X}}}$), both optimization problems (5.82) and (5.83) can be solved easily by grid evaluation, utilizing additionally nearest neighbor trees for the sets $\mathcal{D}_{\mathcal{X}}$ and $\mathcal{D}_{\mathcal{X}_T}$. Generally, compared to the time required to run the simulations, i.e. querying $\widehat{\boldsymbol{\phi}}$ and training the neural network, the computation time for solving these exploration strategy optimization problems by the grid evaluation method combined with the nearest neighbor trees is neglectable in the experiments considered in this work.

From a software point of view, the whole framework is implemented in tensorflow [1].

5.5.2. Computational complexity

If the dataset contains n samples, learning the forward model has $\mathcal{O}(n^3)$ complexity. Calculating the objective (5.81) and its derivative for one single $\mathbf{x}_i$ has $\mathcal{O}(n^2)$ complexity. If the discretization contains n_h many points per dimension, calculating the complete objective (5.81) and its derivative, i.e. also one gradient step for optimizing the neural network, requires

$$\mathcal{O}\left(n_h^d n^2 + n^3\right) \tag{5.85}$$

many operations.

5.6. Parameter Dependent Inverse Model

So far, the result of the developed active inverse model learning framework is *one* learned inverse function $\boldsymbol{\pi}$. However, for a redundant system, it might be beneficial to utilize the redundancy. For example, a kinematically redundant robot arm could use its additional degrees of freedom to realize secondary tasks. In the context of musculoskeletal systems, as discussed in section 3.4, a musculoskeletal system uses the redundancy to alter its co-contraction to realize different passive stiffnesses of the system. In the following, first a general method for parameter dependent inverse models is presented. Then a concrete realization for controlling the co-contraction and hence the stiffness of the musculoskeletal system is developed.

Assume that there is a parameter $\mathbf{p} \in \mathcal{P} \subset \mathbb{R}^r$, $\mathcal{P}$ compact, which in some sense specifies further desired properties of the inverse model in addition to be a right inverse of the true forward model. This way, the inverse model becomes a function of both the desired configuration $\mathbf{x}$ and the parameter $\mathbf{p}$

$$\boldsymbol{\pi} : \mathcal{X} \times \mathcal{P} \to \mathcal{U}. \tag{5.86}$$

The inverse condition then is

$$\forall_{\mathbf{p}\in\mathcal{P}} \forall_{\mathbf{x}\in\mathcal{X}} : \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p})) = \mathbf{x}. \tag{5.87}$$

Analogously to proposition 5.12, if $\boldsymbol{\pi} \in C(\mathcal{X} \times \mathcal{P}, \mathcal{U})$, $\mathcal{P}$ does not contain isolated points, and $\widehat{\boldsymbol{\phi}}$ is continuous as well, the condition (5.87) is equivalent to

$$\int_{\mathcal{P}} \int_{\mathcal{X}} \left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w})) - \mathbf{x} \right\|^2 = 0. \tag{5.88}$$

The forward model is learned exactly as before, according to definition 5.17, on the dataset $\mathcal{D} = \{(\mathbf{u}_i, \mathbf{x}_i)\}_{i=1}^n$ from definition 5.15. Therefore, all other parts of the active inverse learning methodology can be derived completely analogously to the last sections with the only difference that everything is defined on the extended set $\mathcal{X} \times \mathcal{P}$. For technical reasons, assume that $\boldsymbol{\pi}$ as defined in (5.86) can be evaluated on the set $\mathbb{XP} \subset \mathbb{R}^d \times \mathbb{R}^r$ with $\mathcal{X} \times \mathcal{P} \subset \mathbb{XP}$, which, if $\boldsymbol{\pi}$ is a standard neural network as defined in section 4.2, is automatically fulfilled. Now starting with the error estimate, it holds for all $(\mathbf{x}, \mathbf{p}) \in \mathbb{XP}$

$$\left\| \widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p})) - \mathbf{x} \right\| \leq \varepsilon(\mathbf{x}, \mathbf{p}) \tag{5.89}$$

with

$$\varepsilon(\mathbf{x}, \mathbf{p}) = \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p})) - \mathbf{x}\| + \sqrt{\sum_{i=1}^d \left\| \hat{\phi}_i \right\|_{H_{k_i}}^2 s_i(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}))^2}, \tag{5.90}$$

which is proven exactly the same as theorem 5.19. This error estimate allows to define the inverse model learning optimization problem for a parameter dependent inverse model similar to the derivation in section 5.2 as

$$\min_{\mathbf{w}\in\mathcal{W}} \int_{\mathcal{P}} \int_{\mathcal{X}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^d \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w}))^2 + \eta \; l(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w}), \mathbf{p}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{p}. \tag{5.91}$$

Here, the function $l : \mathcal{U} \times \mathcal{P} \to \mathbb{R}^+$ is the central part that distinguishes the different inverses according to their parameter $\mathbf{p}$.

As a next step, the reachable set with error certainty $c > 0$ can be estimated as

$$\mathcal{XP}_c = \{(\mathbf{x}, \mathbf{p}) \in \mathbb{XP} : \varepsilon(\mathbf{x}, \mathbf{p}) < c\} . \tag{5.92}$$

Plugging this into the inverse model learning objective (5.91) leads to the final objective to learn the parameter dependent inverse model

$$\min_{\mathbf{w}\in\mathcal{W}} \iint_{\mathcal{XP}_c} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w}))^2 + \eta\, l(\boldsymbol{\pi}(\mathbf{x}, \mathbf{p}; \mathbf{w}), \mathbf{p})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{p}. \tag{5.93}$$

For the exploration strategy, a similar reasoning as in section 5.4 can be made to find a lower bound on the real distance to the data $\mathcal{D}_{\mathcal{XP}} = \{(\mathbf{x}_i, \mathbf{p}_i)\}_{i=1}^n$

$$\left\|\begin{pmatrix}\mathbf{x}^*\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\mathbf{x}_i\\ \mathbf{p}_i\end{pmatrix}\right\| = \left\|\begin{pmatrix}\mathbf{x}^*\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*, \mathbf{p}^*))\\ \mathbf{p}^*\end{pmatrix} + \begin{pmatrix}\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*, \mathbf{p}^*))\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\mathbf{x}_i\\ \mathbf{p}_i\end{pmatrix}\right\| \tag{5.94}$$

$$\leq \left\|\begin{pmatrix}\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*, \mathbf{p}^*))\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\mathbf{x}_i\\ \mathbf{p}_i\end{pmatrix}\right\| + \left\|\begin{pmatrix}\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*, \mathbf{p}^*))\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\mathbf{x}^*\\ \mathbf{p}^*\end{pmatrix}\right\| \tag{5.95}$$

$$\leq \left\|\begin{pmatrix}\widehat{\boldsymbol{\phi}}(\boldsymbol{\pi}(\mathbf{x}^*, \mathbf{p}^*))\\ \mathbf{p}^*\end{pmatrix} - \begin{pmatrix}\mathbf{x}_i\\ \mathbf{p}_i\end{pmatrix}\right\| + \varepsilon(\mathbf{x}^*, \mathbf{p}^*) \tag{5.96}$$

which holds for all $(\mathbf{x}^*, \mathbf{p}^*) \in \mathbb{XP}$. This leads to the exploration criteria for compact $\mathbb{XP}$

$$(\mathbf{x}^*, \mathbf{p}^*) = \underset{(\mathbf{x},\mathbf{p})\in\mathbb{XP}}{\operatorname{argmax}} \left(\min_{(\mathbf{x}_i,\mathbf{p}_i)\in\mathcal{D}_{\mathcal{XP}}} \left\|\begin{pmatrix}\mathbf{x}\\ \mathbf{p}\end{pmatrix} - \begin{pmatrix}\mathbf{x}_i\\ \mathbf{p}_i\end{pmatrix}\right\| - \varepsilon(\mathbf{x}, \mathbf{p}) \right). \tag{5.97}$$

The other variants from section 5.4 can be defined analogously. This exploration strategy explores in both the reachable set space $\mathcal{X}$ and the parameter space $\mathcal{P}$ simultaneously.

5.6.1. Controlling the Stiffness via Co-Contraction

The passive stiffness of a musculoskeletal system can be influenced by altering the co-contraction. The goal is to learn an inverse model which not only achieves a desired configuration in equilibrium, but additionally also has a parameter to control the co-contraction level. The main idea to achieve this is that there is a scalar parameter $p \in \mathcal{P} = [0, p_{\max}]$, which defines a certain ground level for the muscle stimulations of each muscle. The function $\boldsymbol{\pi}(\mathbf{x}, p) : \mathcal{X} \times \mathcal{P} \to \mathcal{U}$, which is a neural network, should then predict the delta to this ground level to actually reach the desired configuration. More formally, the inverse model is

$$\tilde{\boldsymbol{\pi}}(\mathbf{x}, p) = \boldsymbol{\pi}(\mathbf{x}, p) + p. \tag{5.98}$$

This function $\tilde{\boldsymbol{\pi}}$ should not only be a right inverse to the true forward model for every $p \in \mathcal{P}$, the predicted muscle stimulations should in addition be as close to the ground level on each muscle as possible, i.e.

$$\|\tilde{\boldsymbol{\pi}}(\mathbf{x}, p) - p\|^2 \tag{5.99}$$

should be minimal, which can be realized by

$$l(\mathbf{u}, p) = \|\mathbf{u} - p\|^2 \tag{5.100}$$

in (5.93). Plugging everything together leads to the optimization problem for the weights of $\boldsymbol{\pi}$

$$\min_{\mathbf{w}\in\mathcal{W}} \int_{\mathcal{P}} \int_{\mathcal{X}} \left\| \boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}, p; \mathbf{w}) + p) - \mathbf{x} \right\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}, p; \mathbf{w}) + p)^2 + \eta \left\| \boldsymbol{\pi}(\mathbf{x}, p; \mathbf{w}) \right\|^2 \mathrm{d}\mathbf{x}\, \mathrm{d}p, \tag{5.101}$$

which tries to find a right inverse to the forward model for each $p \in \mathcal{P}$ and staying close to the muscle stimulation ground level. Having learned this $\boldsymbol{\pi}$, the actual inverse is then given by (5.98).

5.7. Learning Hybrid Control - Lengths of Contractile Elements

In this section, a method to predict the lengths $\mathbf{l}^{\mathrm{CE}} \in \mathbb{R}^m$ of the contractile elements of the muscles for an equilibrium configuration is presented. This can then be used for realizing a monosynaptic reflex as described in section 3.6.

The goal is to find a model $\boldsymbol{\Omega} : \mathcal{X} \to \mathbb{R}^m$ that for a desired configuration $\mathbf{x} \in \mathcal{X}$ in equilibrium predicts the corresponding targets lengths $\boldsymbol{\lambda} = \boldsymbol{\Omega}(\mathbf{x})$. The difficulty of this is two fold. First, also in this case there are infinitely many lengths of the contractile element that correspond to the same configuration in equilibrium. Secondly, the length of the contractile element must correspond to an equilibrium state for a static muscle stimulation, as discussed in section 3.6.

However, since the redundancy problem has already been approached by learning the inverse model, obtaining a model to predict the necessary targets lengths of the contractile elements can be framed as a standard supervised regression problem, which addresses the first problem. For the second problem, the data collection process does not involve hybrid control. Instead, the data is collected with the active inverse model learning methodology by applying constant muscle stimulations. If the equilibrium is reached, the lengths of the contractile elements $\mathbf{l}^{\mathrm{CE}}$ for the muscle stimulation $\mathbf{u}$ are stored in a dataset

$$\mathcal{D}_{\boldsymbol{\lambda}} = \left\{ \left(\mathbf{u}_i, \mathbf{l}_i^{\mathrm{CE}} \right) \right\}_{i=1}^{n} . \tag{5.102}$$

Based on this data, a function $\boldsymbol{\Lambda} : \mathcal{U} \to \mathbb{R}^m$, represented as a neural network, is learned to minimize the standard squared error regression objective

$$\min_{\mathbf{w} \in \mathcal{W}} \sum_{i=1}^{n} \left\| \mathbf{l}_i^{\mathrm{CE}} - \boldsymbol{\Lambda}(\mathbf{u}_i; \mathbf{w}) \right\|^2 \tag{5.103}$$

with common stochastic gradient neural network training techniques (refer to section 4.2). The final predictor of the lengths of the contractile elements for a desired position $\mathbf{x}$ is then

$$\boldsymbol{\lambda} = \boldsymbol{\Omega}(\mathbf{x}) = \boldsymbol{\Lambda}(\boldsymbol{\pi}(\mathbf{x})) \tag{5.104}$$

with the learned inverse model $\boldsymbol{\pi}$ from section 5.2. Since the data for $\boldsymbol{\Lambda}$ is generated through the inverse model $\boldsymbol{\pi}$, one can expect that $\boldsymbol{\Omega}(\mathbf{x})$ shows good approximation capabilities.

5.8. Alternative: Represent Reachable Set with GP Classifier

During the development of the active inverse model learning framework, an alternative to present the reachable set has been explored. The idea is to learn a discriminative function that indicates

$$g(\mathbf{x}) \begin{cases} > 0 & \text{if } \mathbf{x} \in \mathcal{X} \\ \leq 0 & \text{if } \mathbf{x} \notin \mathcal{X} \end{cases} \tag{5.105}$$

as a Gaussian process classifier (for details about GP classification refer to [37]). The reachable set is then estimated as

$$\mathcal{X}_g = \left\{ \tilde{\mathbf{x}} \in \mathbb{R}^d : g(\tilde{\mathbf{x}}) > 0 \right\}. \tag{5.106}$$

With this, the active inverse model learning optimization problem would be

$$\min_{\mathbf{w} \in \mathcal{W}} \int_{\left\{ \tilde{\mathbf{x}} \in \mathbb{R}^d : g(\tilde{\mathbf{x}}) > 0 \right\}} \|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w}))^2 + \eta \; l\left(\boldsymbol{\pi}(\mathbf{x}; \mathbf{w})\right) \mathrm{d}\mathbf{x}. \tag{5.107}$$

Based on the uncertainty of the classifier, one can also define an active exploration strategy

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x} \in \mathbb{R}^d} \nu(\mathbf{x}) \quad \text{s.t.} \quad g(\mathbf{x}) = 0 \tag{5.108}$$

that explores at the boundary of the classifier where the uncertainty ν provided by the GP of the discriminative function is maximal.

The main question is, however, how the data for the classifier can be generated. It is clear that every reached position $\mathbf{x}$ during the exploration process is added to the dataset with a positive value for the discriminative function. How negative samples, which indicate certain parts that are not reachable, can be added to the dataset is not straightforward. Two different approaches have been tested. First of all, the mean prior of the discriminative function is set to a negative value. This indicates that in distance to the data points the classifier predicts unreachability. The reachability margin around the data points mainly depends on the kernel for the classifier and its hyperparameters. Without any method to generate negative samples, this method turned out to have a good performance in the beginning, since it explores rapidly. However, since never negative samples are added, the exploration strategy always tries to expand the reachable set, without knowing the boundary of the reachable set. Therefore, the performance at some point does not increase anymore. In contrast, the proposed active exploration strategy from section 5.4 and the reachable set estimation from section 5.3 automatically trade-off expanding the reachable set and improving the inverse inside the already estimated set. The second approach is to assume that there are sensors that indicate if the boundary of the workspace is reached. With this information, negative samples could be added to the dataset of the classifier, enabling to stop trying to expand the reachable set at the boundary of the true reachable set. Please note that this sensor information does not correspond to knowing the true $\mathcal{X}$ a priori, since the sensor only gives information during sampling of the true system. Having such sensors, however, corresponds to further knowledge this work would like to avoid.

An other disadvantage of using a separate classifier for representing the reachable set is that there is no coupling between the learned inverse model and the estimated reachable set. In contrast, the reachable set estimate $\mathcal{X}_c$ from theorem 5.27 is directly coupled to the inverse and tells also whether the current learned inverse is actually able to reach the points at all.

6. Experiments

The main aim of this section is to show experimentally that the developed active inverse model learning framework is not only able to accurately learn an inverse model of a highly redundant biomechanical system (section 6.4, 6.2, 6.3), but also to estimate its reachable set (section 6.7, 6.2). On the other hand, it is investigated in section 6.8 and 6.9 which parts of the developed methodology have which influence on the learning performance. Furthermore, the behavior of the musculoskeletal system, e.g. in terms of trajectories, when controlled with the learned inverse model, is studied in various experiments. For example, in section 6.5, out-of-center reaching experiments are performed. Section 6.6 investigates the attractor property of the musculoskeletal system discussed in section 3.5.

The experiments of sections 6.2 – 6.11 are all performed using the simulated human arm model, consisting of two joints (shoulder and elbow) that are driven by 6 muscles. Refer to section 3.7 for details about this model. The goal of the inverse model is to predict the 6 necessary muscle stimulations $\mathbf{u} \in \mathcal{U} = [0,1]^6$ such that the arm reaches a desired *hand position* $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^2$ in equilibrium.

In section 6.12, the same methodology is applied to a real bio-inspired robot that is driven by five pneumatic muscles, mimicking the simulation model of a human arm with two joints, see section 3.8 for more details about this robot. Here, the goal of the inverse model is to predict the 5 necessary pressure values for the pneumatic muscles such that the arm reaches a desired *joint angle* configuration $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^2$ in equilibrium.

6.1. Hyperparameters

If not stated differently for a specific experiment, the hyperparameters are as follows: The learning rate of the inverse model $\boldsymbol{\pi}$ is $\alpha_{\boldsymbol{\pi}} = 0.0005$, the learning rate of the lambda-controller is $\alpha_{\boldsymbol{\Lambda}} = 0.001$. The mean prior of the forward model is $\mathbf{m} = (m_1, m_2) = (0.4, -0.3)$, which has been chosen such that it is roughly in the middle of $\mathcal{X}$. The regularization parameter of each component of the forward model is $\sigma_i = 0.001$. The length-scale of the kernels of the forward model for each dimension and each component is $l = 0.2$. The initial muscle stimulation is $\mathbf{u}_0 = (0.22, 0.05, 0.12, 0.05, 0.15, 0.12)^T \in \mathbb{R}^6$. The inverse model and the lambda-controller are trained for 30 episodes every time a new data point has been added. The first 22 data points are sampled with noise according to the antagonistic noise generation proposed in [13] with parameters $\sigma_1 = \frac{1}{15}$ and $\sigma_2 = \frac{1}{30}$. A total of 1200 data points are collected. The error certainty to estimate the reachable set is $c = 0.02$. The bound on the RKHS norm of each component of the true forward model is set to $\beta_i = 2$, $i = 1, 2$. This has been tuned by testing a value of 1, $\sqrt{2}$ and 2. With exception to one experiment in section 6.9, the exploration strategy (5.83) with the trust region is used. The parameters of the exploration strategy are $d_{\mathcal{X}} = 0.1$, $d_{\mathcal{X}_T} = 0.02$, $n_{\mathcal{X}_T} = 200$. To solve the exploration optimization problem, $\widetilde{\boxed{\mathcal{X}}}$ from definition 5.39 is a 100×100 equidistant grid. The estimated reachable set $\mathcal{X}_c$ for the integral is calculated on a 30×30 grid for $\widetilde{\boxed{\mathcal{X}}}$. The corresponding bounding box is $\boxed{\mathcal{X}} = [-0.2, 0.65] \times [-0.7, 0.2]$. The muscle-stimulation trade-off parameter is $\eta = 10^{-5}$. This first seems small, but has the same magnitude than the squared desired accuracy of the inverse model.

Each simulated experiment is repeated 5 times with the same parameters but different random seeds. If a mean value with shaded error bars is drawn in the plots of this section, the shaded error bars depict one standard deviation.

6.2. Exploration Behavior

Figure 6.1 shows exploration result after a total of 1200 collected data points $\mathcal{D}_{\mathcal{X}}$ (blue) in the $\mathcal{X}$ space, i.e. the reached hand positions of the arm model, as well as the estimated $\mathcal{X}_c$ (orange) and true $\mathcal{X}$ (green), $\hat{\mathcal{X}}_c$ (green dashed) workspace. These workspaces are visualized by drawing their outer boundaries as lines. There is a close match between the dashed green line and the orange one, indicating that the developed methodology has estimated the reachable set with error certainty c accurately. This error certainty is set to $c = 0.02$, which means that for all points inside the learned $\mathcal{X}_c$, the inverse model guarantees to reach the desired point with an error of less than 2 cm. One can also see that the true reachable set in solid green is also only an estimation based on the analytical model (refer to section 3.7, figure 3.3) and therefore not completely correct, since there are data points outside $\mathcal{X}$. This is due to the fact that joint limits are modeled as damped springs (refer to section 3.1), allowing a slight violation of the joint limits depending on the exerted torques from the muscles. Furthermore, it seems that the inverse model is not able to reach the lower left part of the true workspace. The reason for this is that in this area, again due to modeling the joint limits as springs, the simulator starts to

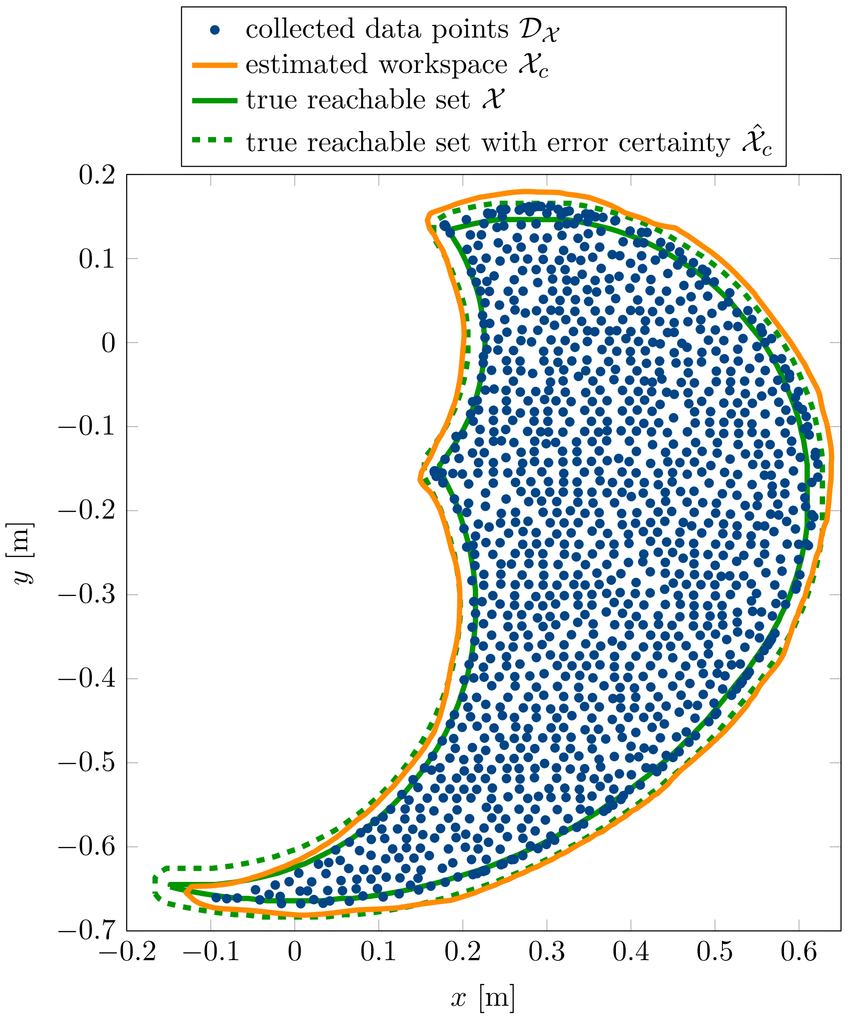


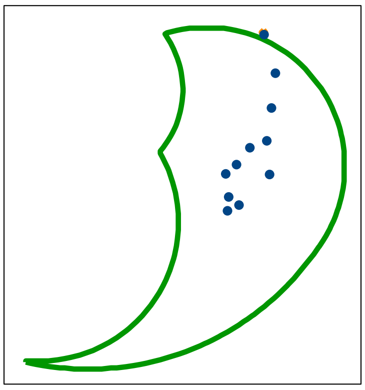
Figure 6.1: Exploration result after a total of 1200 collected data points. Blue points denote the reached hand positions of the arm. The dashed green line visualizes the boundary of set $\hat{\mathcal{X}}_c$ for $c = 0.02$, the orange line the corresponding learned estimate $\mathcal{X}_c$ of the reachable set.

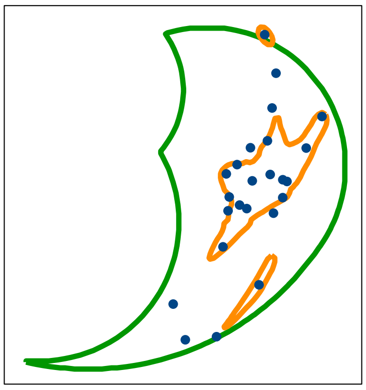
oscillate, not reaching an equilibrium. Hence, this is not a deficiency of the inverse model, in contrary, the inverse model recognizes this problem. For a more detailed discussion about the estimation of the reachable set refer to section 6.7.

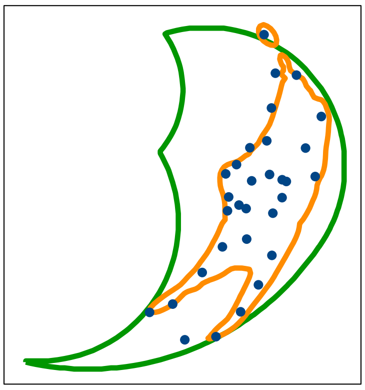
Next, figures 6.2 – 6.6 have to purpose to visualize the exploration behavior. Figure 6.2 shows the exploration process for a growing number n of collected data points in terms of both the distribution of the collected data points as well as the evolution of the learned reachable set. The first 22 data points are sampled with noise. At 10 collected data points (figure 6.2a), the estimated reachable set is still nearly empty. Starting from figure 6.2b, where 23 data point have been collected, the data is generated only using the exploration strategy and the current learned $\boldsymbol{\pi}$, without noise. One can also see in figure 6.2b that the estimated reachable set is not just an alpha shape of the data points, since there are some data points that are not included in $\mathcal{X}_c$ yet. After only 150 collected data points (figure 6.2j), nearly the complete reachable set is explored. With respect to a 6 dimensional control input space, 150 data points are very sparse. Many active learning methodologies have to deal with the so-called exploration/exploitation trade-off. The proposed active inverse model learning framework automatically deals with this: Whereas in the beginning of the procedure, up to about 150 data points (figure 6.2a – 6.2j), most collected data points *expand* the reachable set and therefore focus on fast exploration. From there on (figure 6.2k – 6.2x), the algorithm focuses on improving inside the estimated reachable set. As one can see, even in the beginning of the exploration, the distribution of the collected data points is relatively uniform in the sense that there is a comparably large and equal separation distance between the reached hand positions. This also indicates that the developed exploration strategy (section 5.4) empirically leads to good lower bounds on the true fill distance. Furthermore, the fact that the method can explore so quickly shows that the learned inverse model is able to extrapolate. Note that in all subfigures of figure 6.2 the number n does not count the first collected data point.

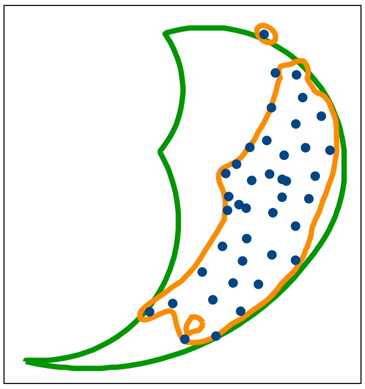
To further investigate the exploration behavior, figure 6.3, 6.4 and 6.5 show the way the exploration strategy (section 5.4) works. In figure 6.3a, three possible, exemplary exploration targets in the $\mathcal{X}$-space are shown. The error estimate circles of the red and violet target have a positive distance to the nearest data points. This distance of the circle to the data points is the lower bound on the true fill distance. The error estimate circle of the light blue target contains data points, therefore, the model believes that this target does not improve the true fill distance. The red target is actually the one that has the highest lower bound on the real fill distance of all possible targets. The actual reached position for this target is shown in figure 6.3b. One can also see that the reached position is inside the error estimate. Figure 6.4 and 6.5 shows a similar situation at an earlier stage of the exploration. Furthermore, between figure 6.4a and 6.4b, one can see how the reachable set updates after the new data point has been added and the model is retrained.

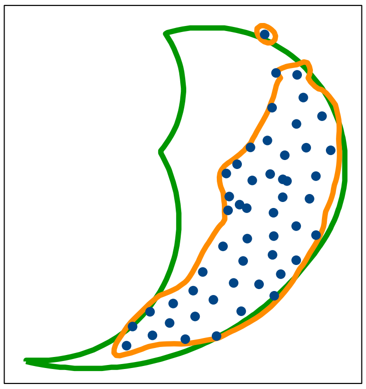
To further support the claim that the exploration strategy automatically and inherently trades of exploration and exploitation, figure 6.6 visualized how many target points the exploration strategy has chosen inside and outside the current estimated reachable set $\mathcal{X}_c$ over the number of collected data points. One can see that in the beginning more data points are chosen outside of $\mathcal{X}_c$, indicating that the exploration strategy focuses on expanding the reachable set. After 183 data points, the great majority of targets are chosen inside of $\mathcal{X}_c$, which means that from now on the focus is on improving the model inside $\mathcal{X}_c$. Indeed, as seen in figure 6.2, after 150 data points the workspace has nearly completely been explored, which the exploration strategy recognizes.

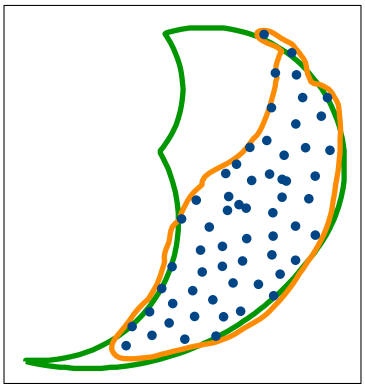

(a) $n = 10$ (still including noise)

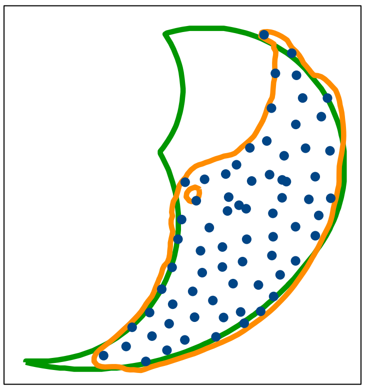

(b) $n = 23$ (no noise anymore)

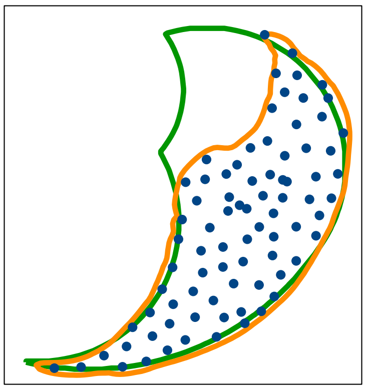

(c) $n = 30$

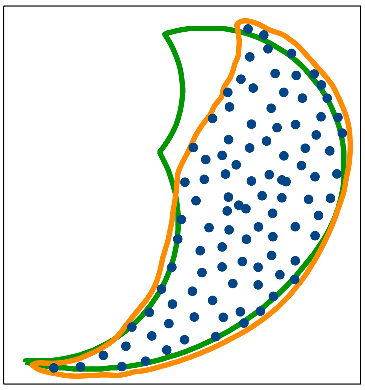

(d) $n = 40$

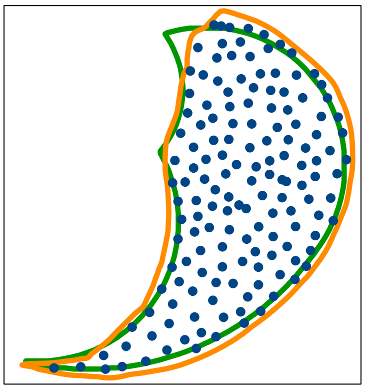

(e) $n = 50$

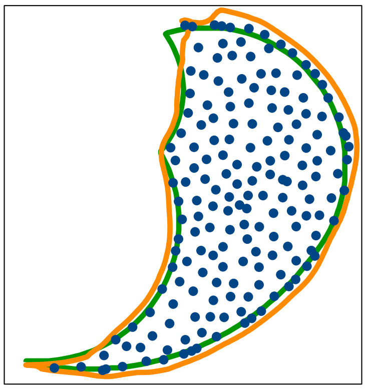

(f) $n = 60$


(g) $n = 70$


(h) $n = 80$


(i) $n = 100$


(j) $n = 150$


(k) $n = 175$

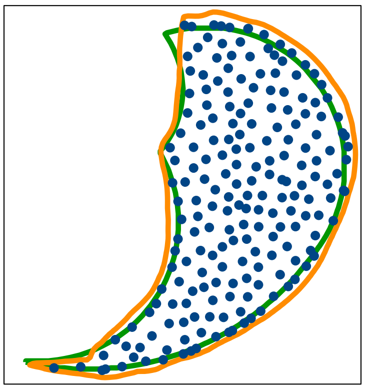
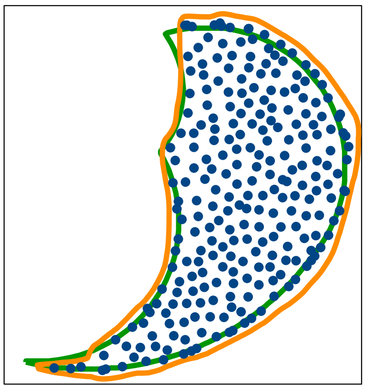
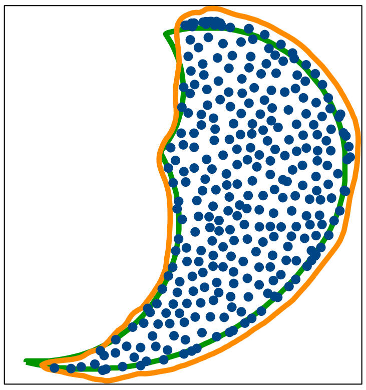
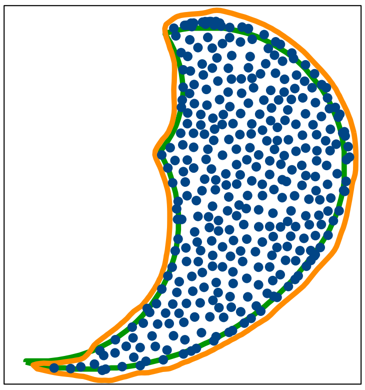
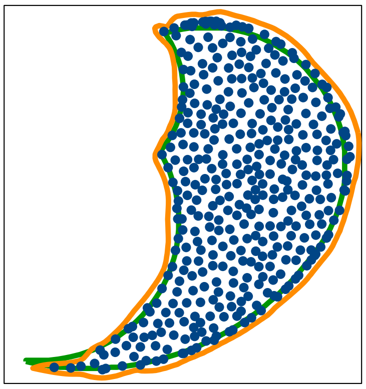

(l) $n = 200$

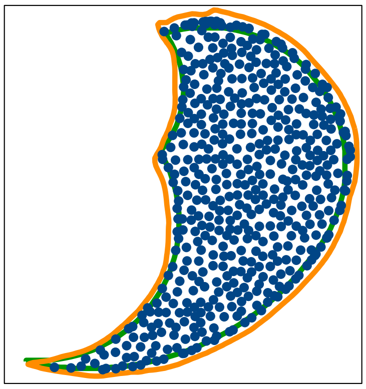
Figure 6.2: Exploration behavior. Subplots show different number n of collected data points so far. Same axis limits and legend as in figure 6.1.

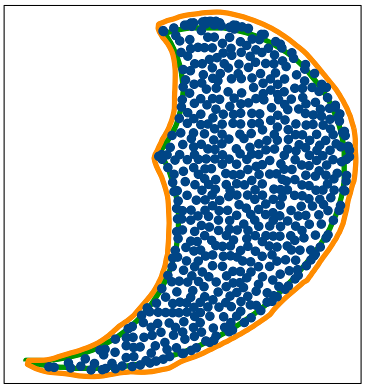

(m) $n = 250$

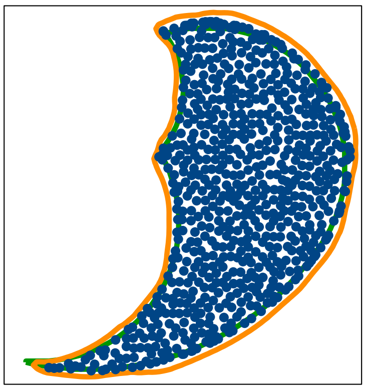

(n) $n = 300$

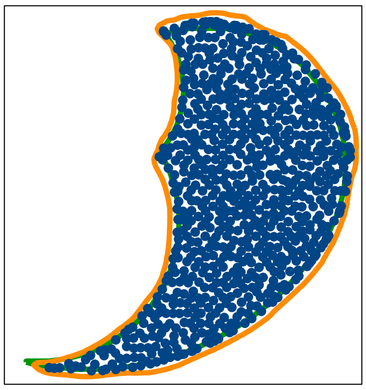

(o) $n = 350$

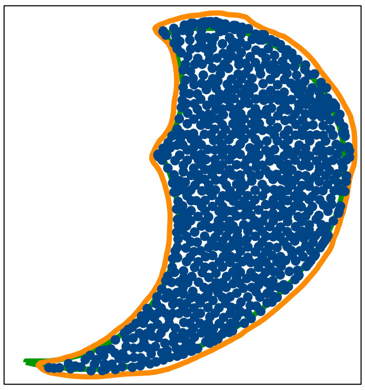

(p) $n = 400$

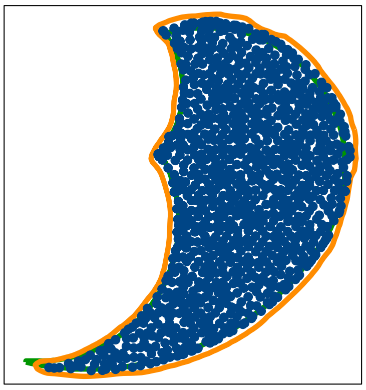

(q) $n = 500$

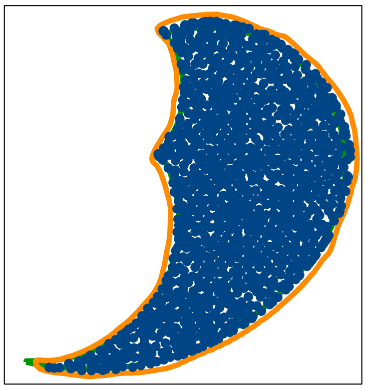

(r) $n = 600$


(s) $n = 700$


(t) $n = 800$


(u) $n = 900$


(v) $n = 1000$


(w) $n = 1100$

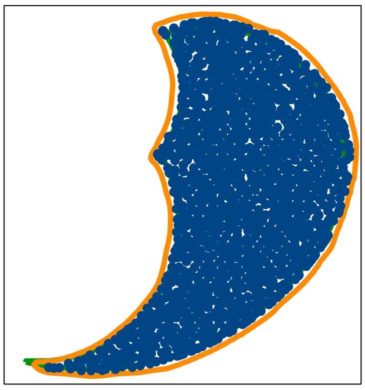
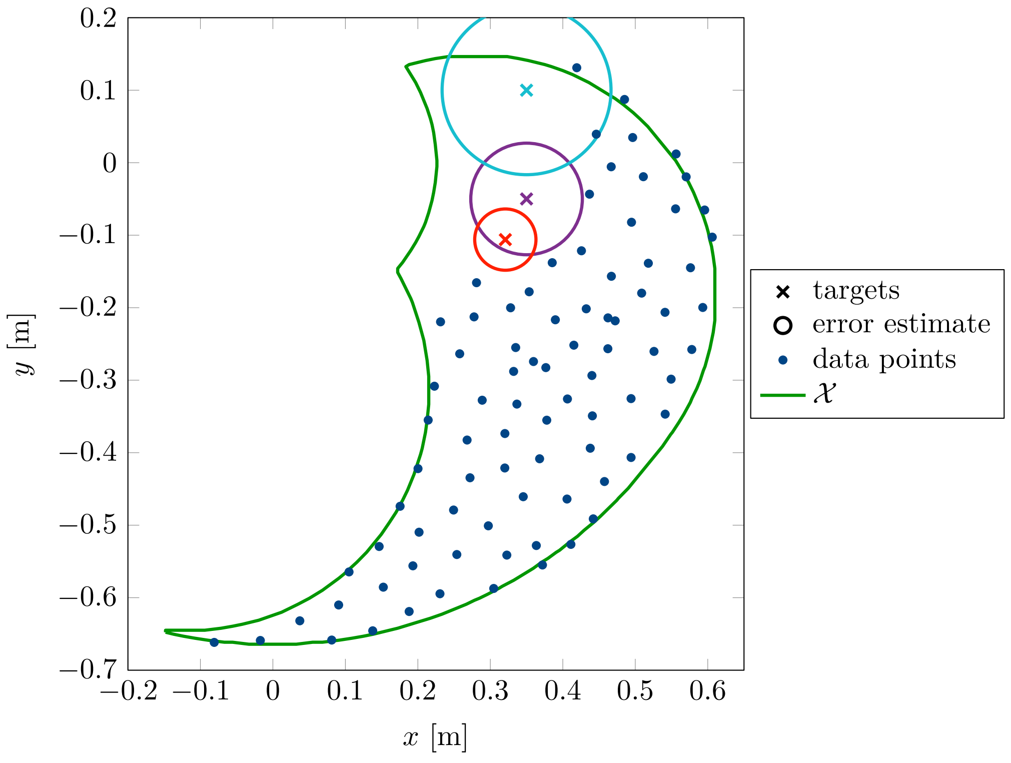
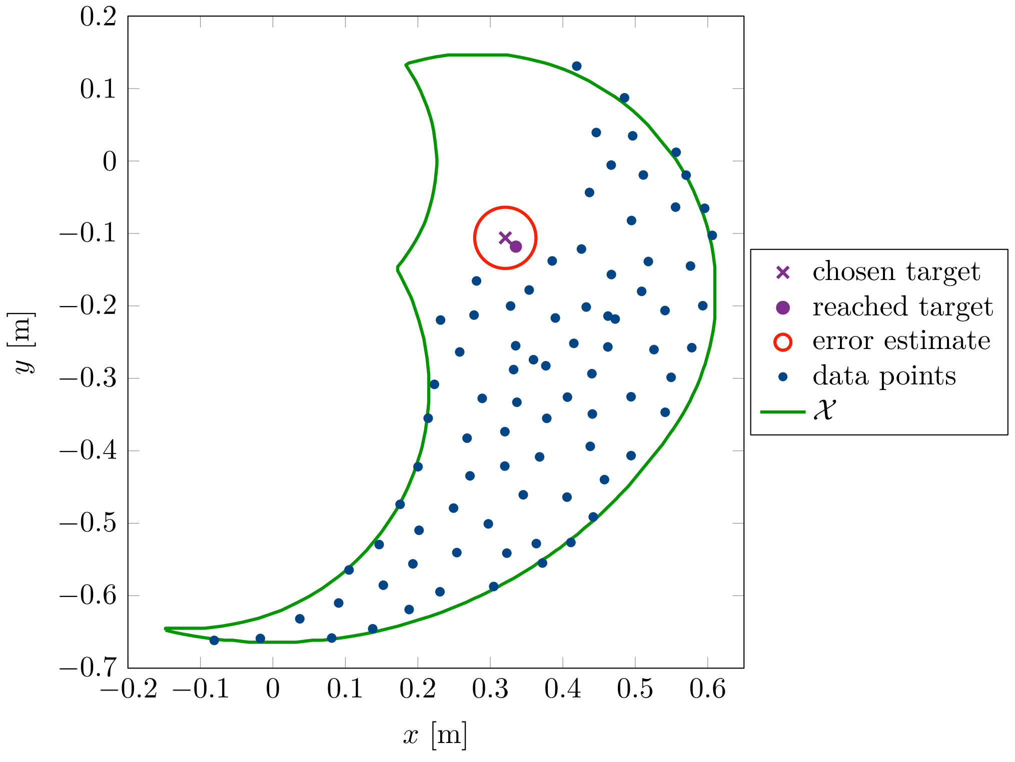

(x) $n = 1199$

Figure 6.2: (continued) Exploration behavior. Subplots show different number n of collected data points so far. Same axis limits and legend as in figure 6.1.

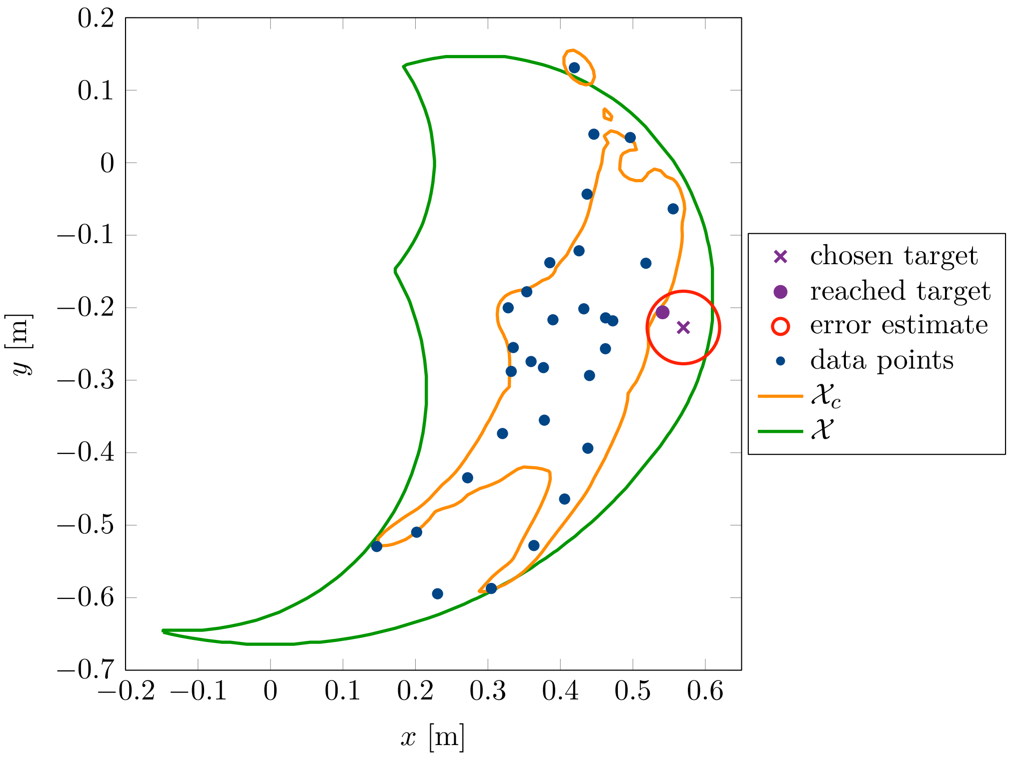


(a) Multiple possible targets with their error estimates

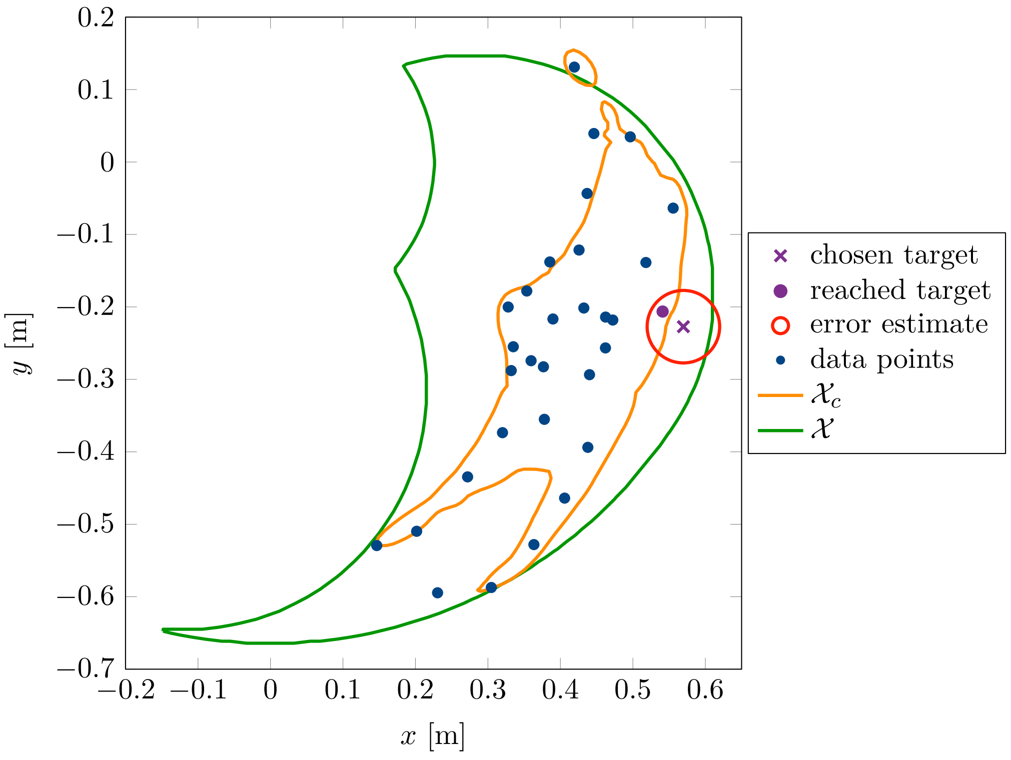


(b) Chosen target that maximizes the lower bound on the real fill distance

Figure 6.3: Visualization of the exploration strategy as lower bound on the real fill distance. 84 data points in data set before the new one is added.



(a) Reachable set before new data point is added



(b) Reachable set after new data point is added

Figure 6.4: Visualization of the exploration strategy as lower bound on the real fill distance. 29 data points in data set before the new one is added.

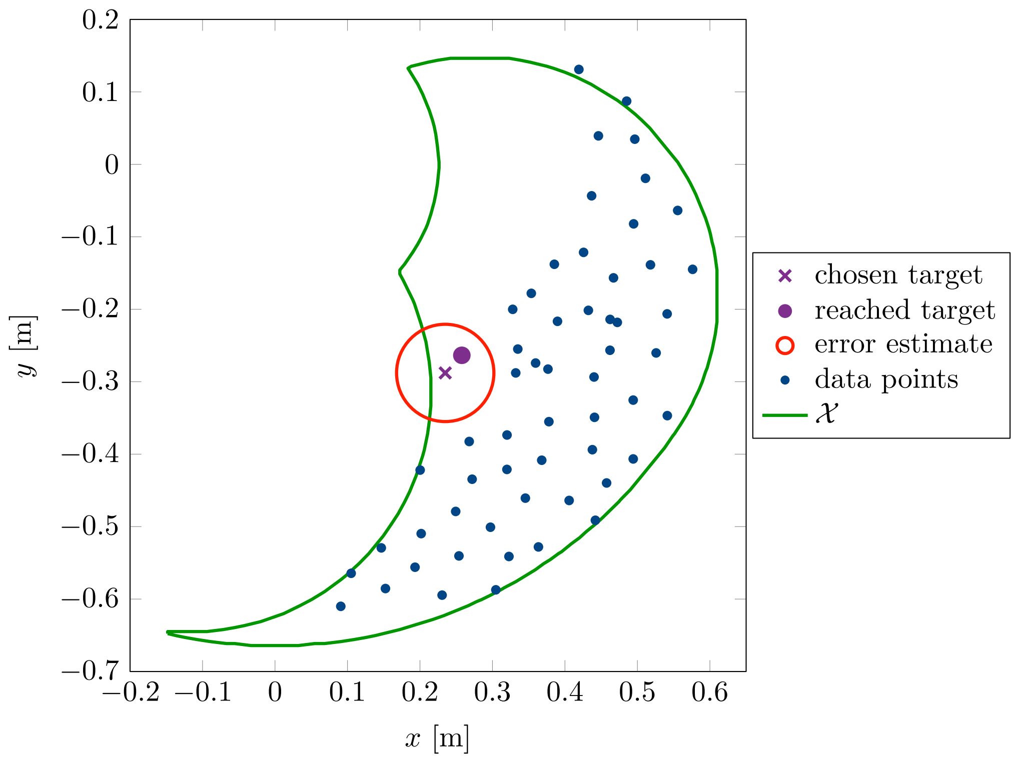


Figure 6.5: Visualization of the exploration strategy lower bound on the real fill distance. 55 data points in the data set before the new one is added.

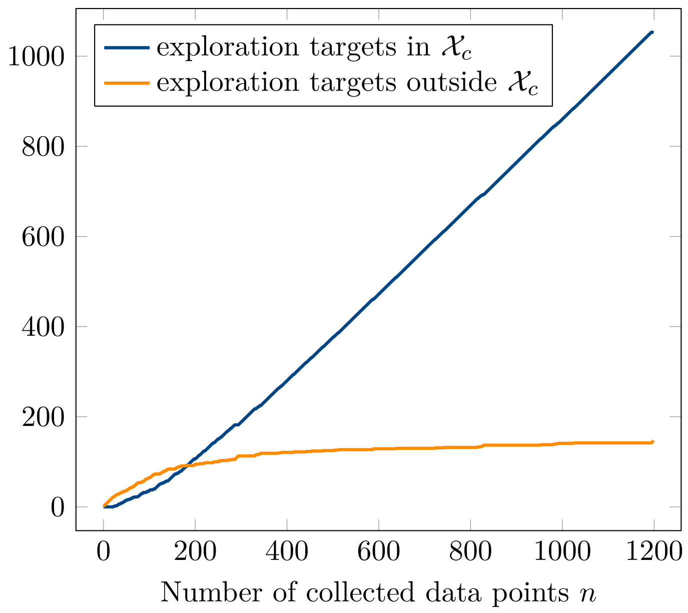


Figure 6.6: Visualization of the exploitation/exploration behavior of the proposed active exploration strategy. If an exploration target is chosen outside of $\mathcal{X}_c$, the orange line is increased by one, if an exploration target is chosen inside of $\mathcal{X}_c$, the blue line is increased by one.

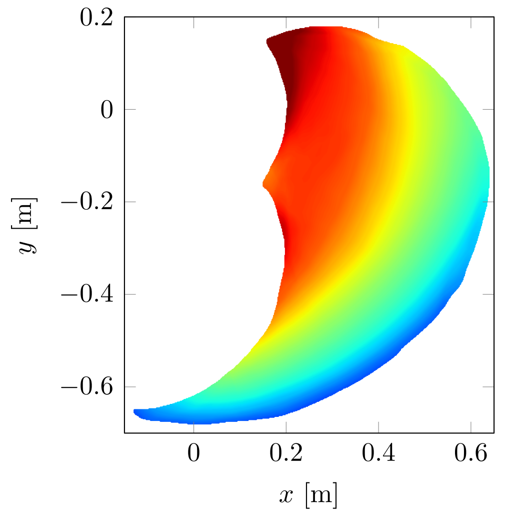
6.3. Learned $\boldsymbol{\pi}$

The goal of this section is to visualize the learned inverse model $\boldsymbol{\pi} : \mathcal{X}_c \to \mathcal{U}$. Figure 6.7 shows each component of $\boldsymbol{\pi}(\mathcal{X}_c)$, i.e. the muscle stimulations for each of the six muscles, as a function of the desired $\mathbf{x} = (x, y)$ hand position of the arm at the end of the exploration with 1200 collected data points. The color depicts the actual muscle stimulation, blue low, red high. One can see that the learned inverse is smooth. The estimated workspace $\mathcal{X}_c$ is also implicitly visualized in figure 6.7, since, as discussed in section 5.1.2, the domain of the inverse model is part of its mathematical definition. One can already see an intuitive, antagonistic and gravity compensating distribution of the muscle stimulations to reach the different parts of the workspace.

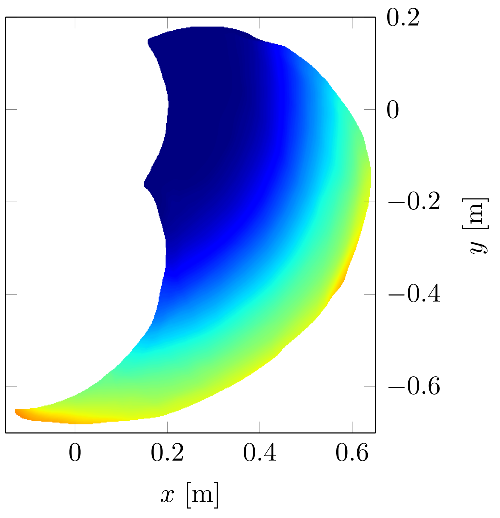
To further visualize this antagonistic behavior, figure 6.8 shows the stimulations $\boldsymbol{\pi}(\mathcal{X}_c)$ for each antagonistic muscle pair in a separate plot. For example, one can see the anti-correlation between the muscle stimulations of the elbow flexor and extensor (figure 6.8a) as well as of the shoulder flexor and extensor (figure 6.8b). As seen both in figure 6.7d and 6.8c, the biarticular extensor muscle is not used much, the biarticular flexor 6.7c, however, is important to reach the top left part of $\mathcal{X}$ and also has the highest predicted muscle stimulation at a certain location among all others. In addition, figure 6.8d, 6.8e and 6.8f show the collected muscle stimulations $\mathcal{D}_{\mathcal{U}}$ on top of the predicted stimulations by $\boldsymbol{\pi}(\mathcal{X}_c)$. Since the data is collected by bootstrapping $\boldsymbol{\pi}$, there is a relatively close match between the sampled and predicted stimulations.

Similarly to figure 6.8, figure 6.9 visualizes the antagonistic behavior of $\boldsymbol{\pi}$ after 150 collected data points. Figure 6.10 visualizes both learned inverse models after 1200 and 150 collected data points at the same time. Although 150 collected muscle stimulations $\mathcal{D}_{\mathcal{U}}$ are much more sparse, the predicted muscle stimulations $\boldsymbol{\pi}(\mathcal{X}_c)$ look already similar to the ones shown in figure 6.8, which can be explained by the fact that, as discussed in section 6.2, after 150 data points, the majority of the true reachable set is explored, which can be seen figure 6.2j.

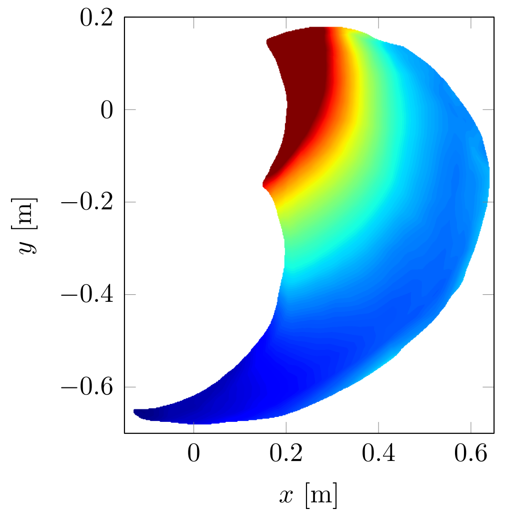
Finally, figure 6.11 visualizes the error estimate $\varepsilon_\beta(\mathbf{x})$ of the learned inverse model as a function of the desired $\mathbf{x} = (x, y)$ hand position after 1200 collected data points. One can see that for the majority of the estimated reachable set the model predicts a low error below 3 mm. Then beyond the boundary of the true reachable set, the error estimate increases approximately linearly. For an ideal error estimate, this behavior is totally expected, since the true error outside the true reachable set would at least linearly increase, since a point outside the true reachable set cannot be reached and hence would have an error that is at least as high as the closest distance of the target point to the boundary. According to (5.40) from definition 5.25, the true reachable set $\hat{\mathcal{X}}_c$ with error certainty c would also show this error distribution. Therefore, the derived error estimate is also in practice close to the ideal behavior.



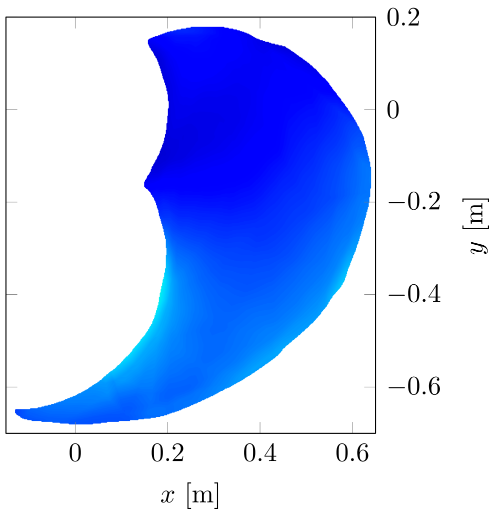
(a) Elbow flexor u_1



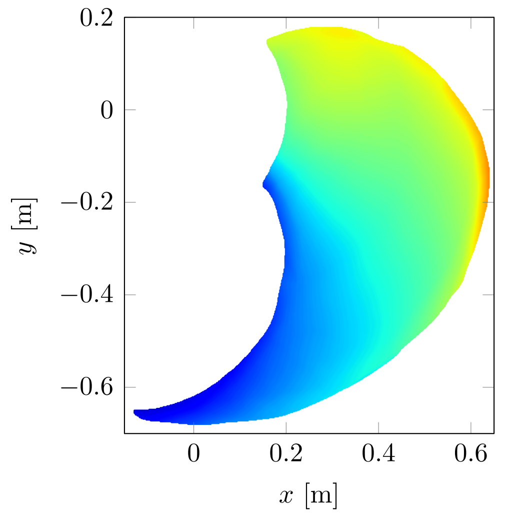
(b) Elbow extensor u_2



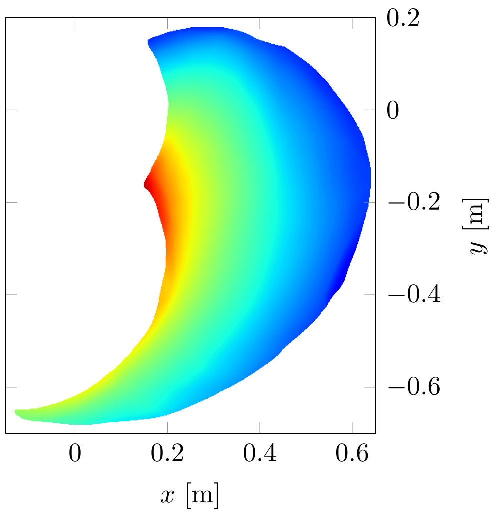
(c) Biarticular flexor u_3



(d) Biarticular extensor u_4



(e) Shoulder flexor u_5



(f) Shoulder extensor u_6

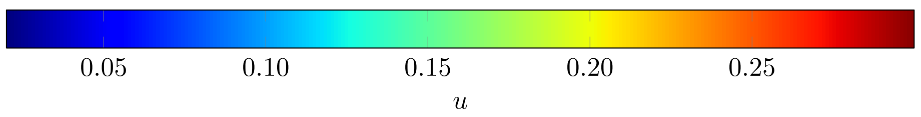
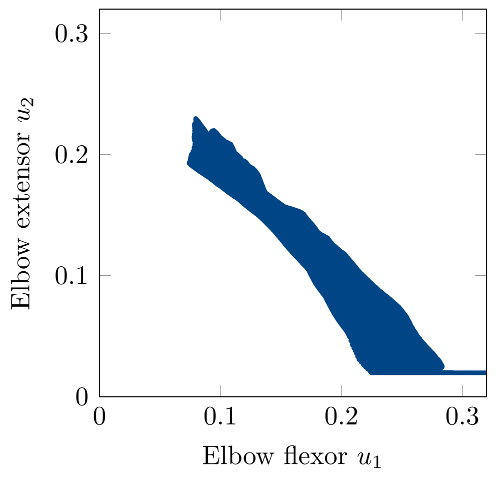
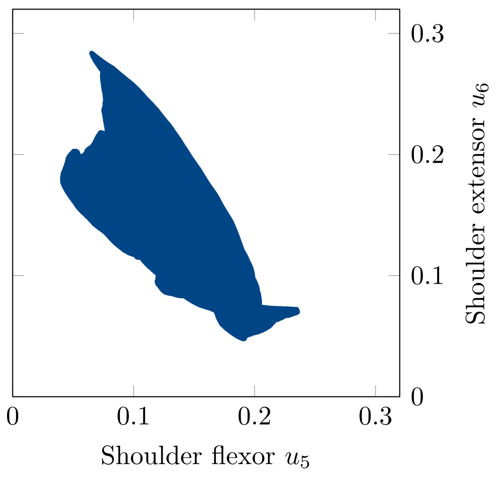


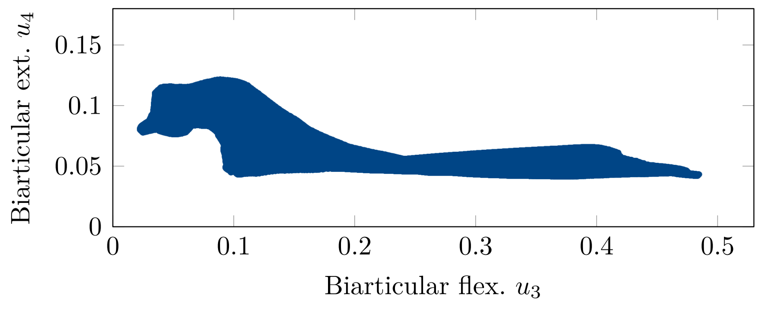
Figure 6.7: Learned $\boldsymbol{\pi} : \mathcal{X}_c \to \mathcal{U}$ with estimated workspace $\mathcal{X}_c$ for $c = 0.02$. Colors indicate muscle stimulations $\mathbf{u} = (u_1, \ldots, u_6)^T \in \mathbb{R}^6$. Dark red means a muscle stimulation larger than 0.3 (maximum 0.52).



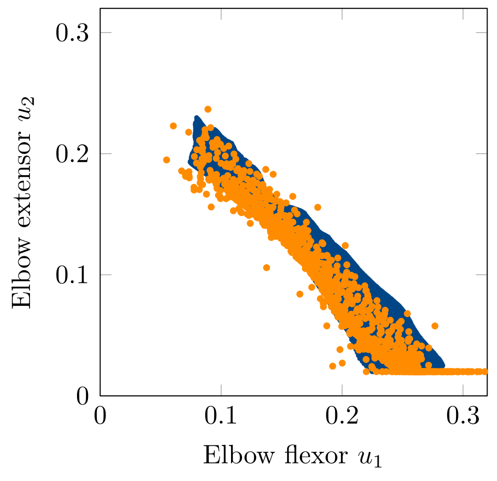
(a) Predicted elbow muscle stimulations



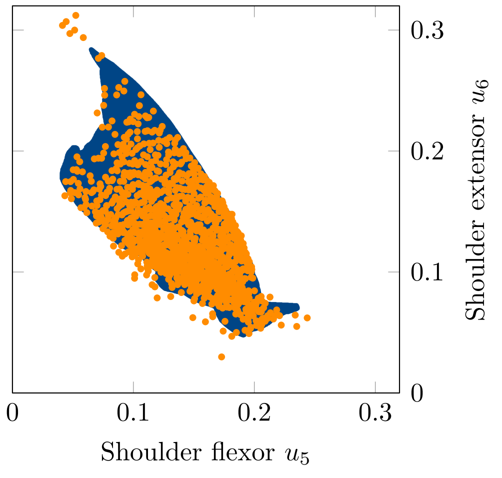
(b) Predicted shoulder muscle stimulations



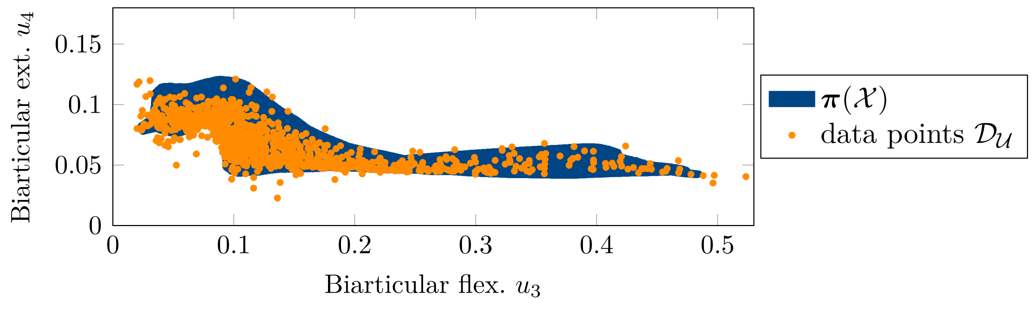
(c) Biarticular muscle stimulations



(d) Elbow muscle stimulations

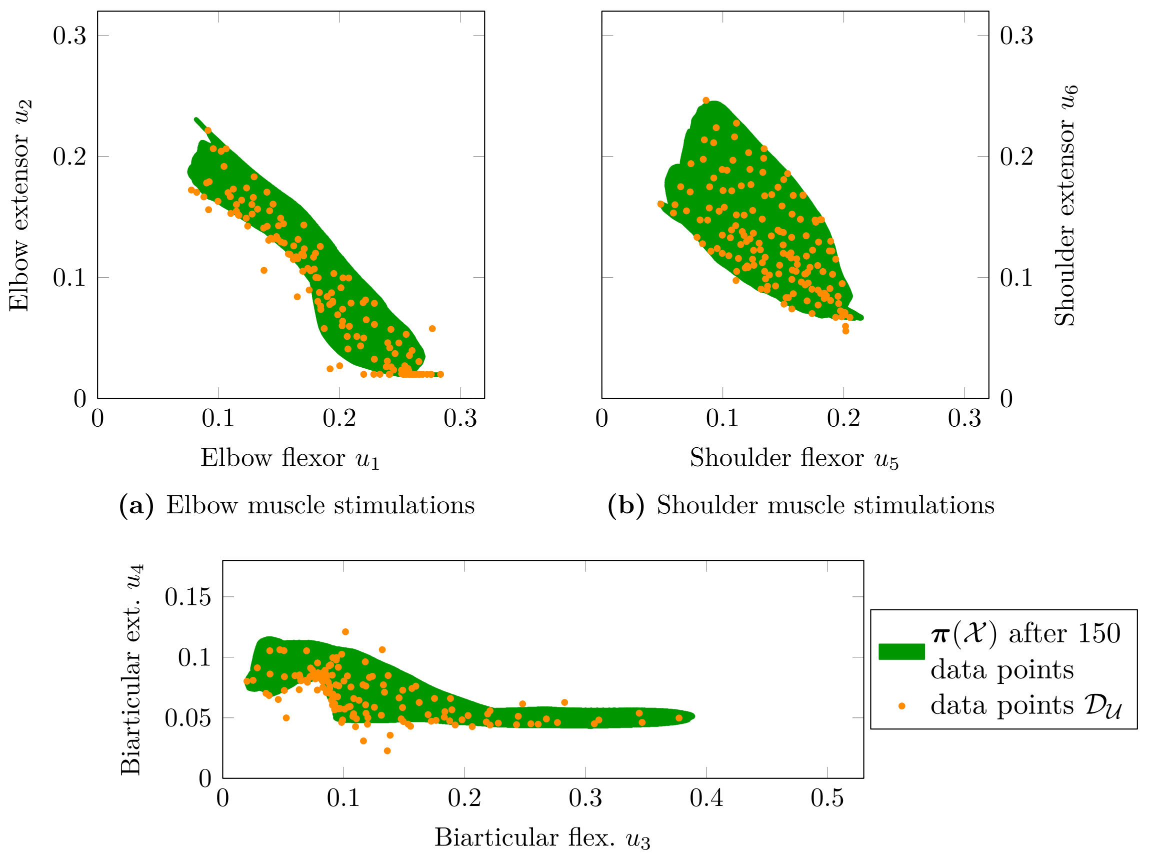


(e) Shoulder muscle stimulations



(f) Biarticular muscle stimulations

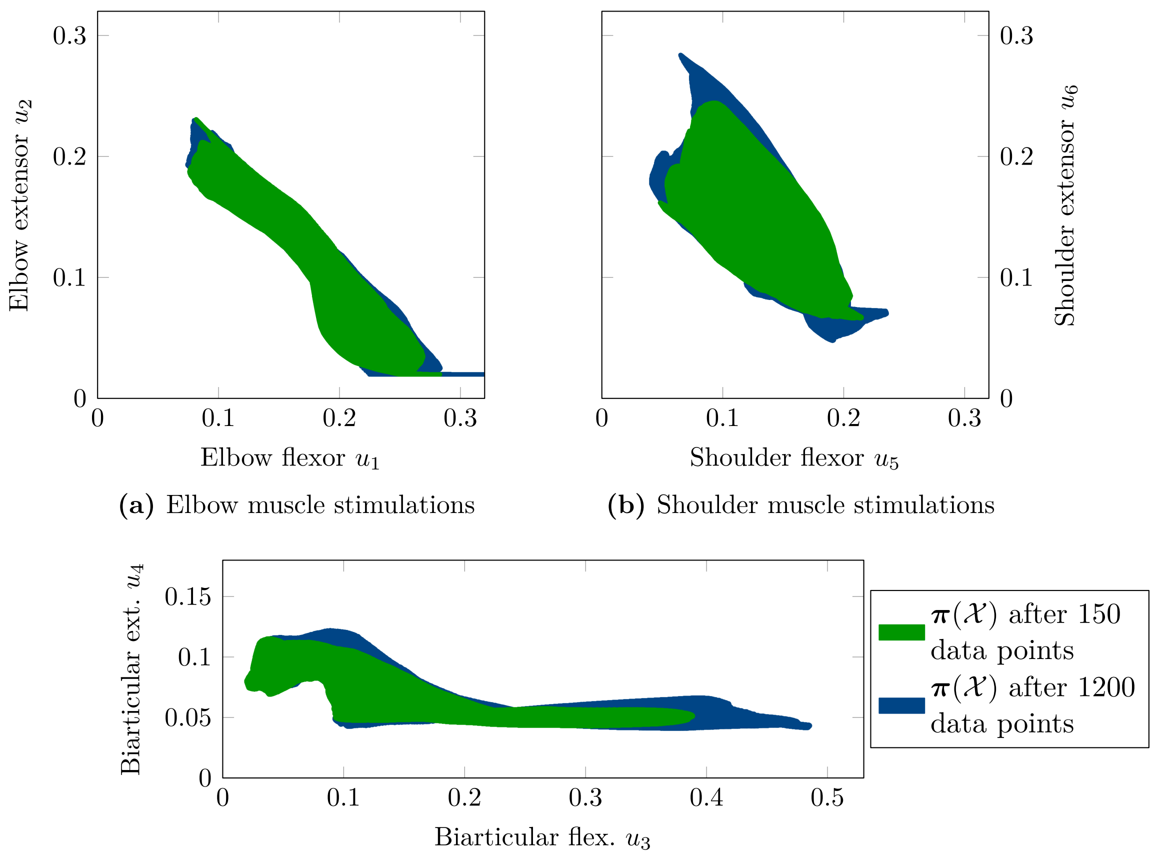
Figure 6.8: Visualization of antagonistic behavior of the learned inverse model $\boldsymbol{\pi}$ at the end of the exploration with 1200 collected data points.



(a) Elbow muscle stimulations **(b)** Shoulder muscle stimulations

(c) Biarticular muscle stimulations

Figure 6.9: Visualization of antagonistic behavior of the learned inverse model π after 150 collected data points.



(a) Elbow muscle stimulations **(b)** Shoulder muscle stimulations

(c) Biarticular muscle stimulations

Figure 6.10: Comparison of learned π after 150 and 1200 collected data points.

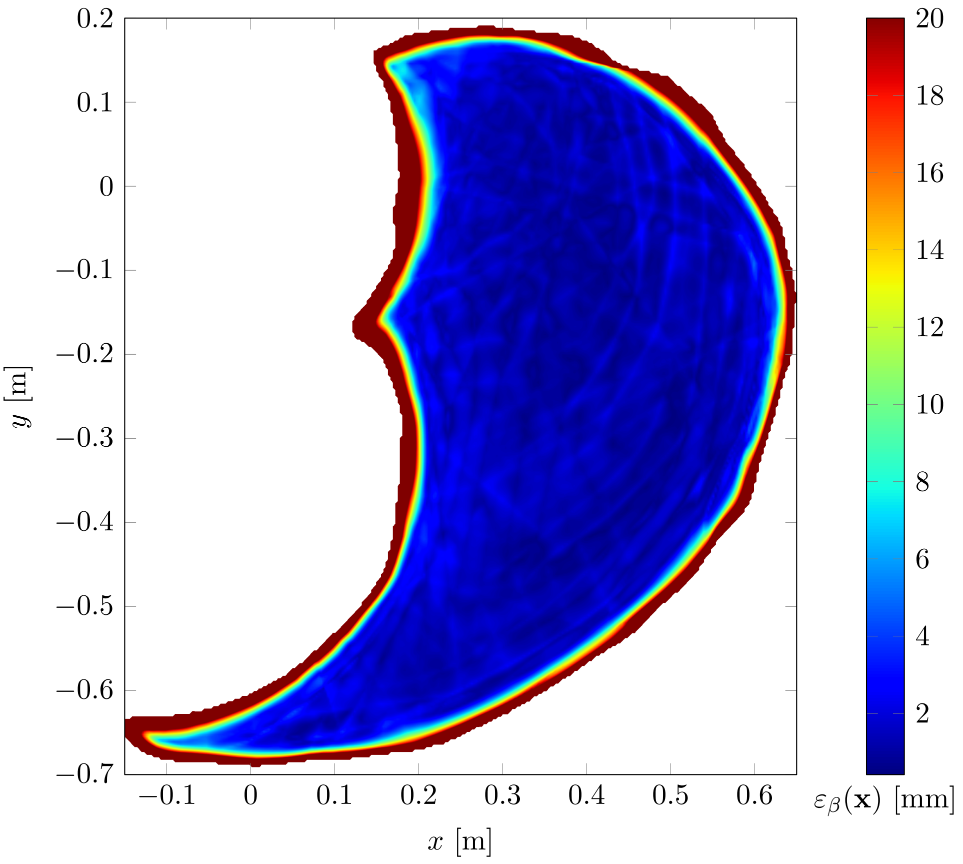


Figure 6.11: Visualization of the error estimate $\varepsilon_\beta(\mathbf{x})$ as a function of the desired $\mathbf{x} = (x, y)$ hand position. Dark red means 20 mm $< \varepsilon_\beta(\mathbf{x}) \leq 50$ mm, i.e. the boundary to the constant dark red value indicates the boundary of the estimated reachable set $\mathcal{X}_c$ with $c = 0.02 = 20$ mm. White is $\varepsilon_\beta(\mathbf{x}) > 50$ mm.

6.4. Point Reaching Evaluation

This experiment aims at testing the accuracy of the learned inverse model, when applied to the true system. The blue points in figure 6.12 show the desired target points. To achieve comparable results, these target points also serve as references for the ablation study in section 6.8 and the exploration strategy comparison in section 6.9 and 6.10.

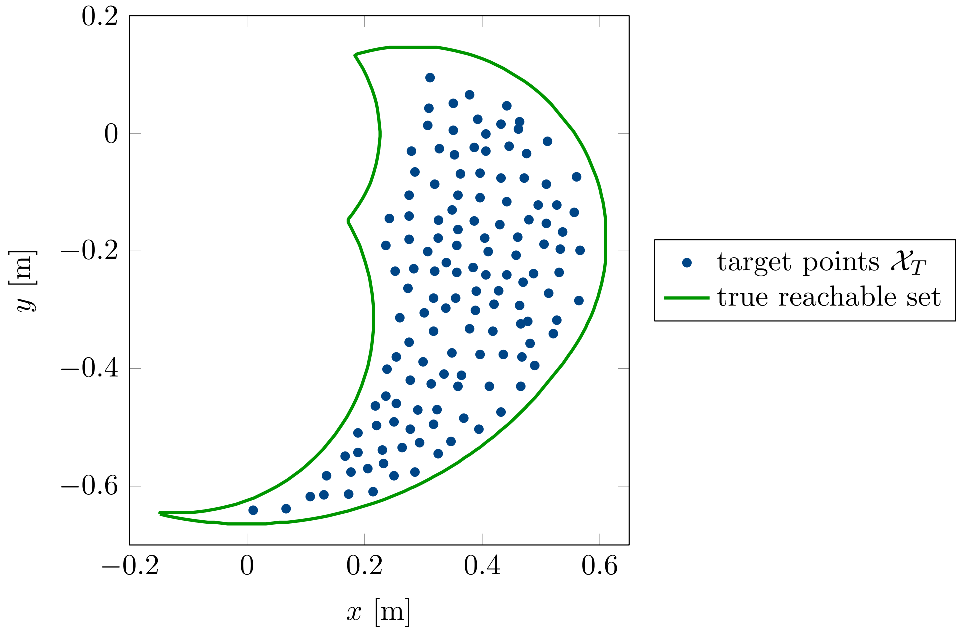


Figure 6.12: Target points for evaluating the point reaching accuracy, 137 in total.

Figure 6.13 shows the actual reached points (orange), when using the learned inverse model from section 6.3, 6.2 at the end of the exploration (1200 collected data points). As one can see, the orange points are visually hardly distinguishable from the targets in blue. Quantitatively, the mean error between the targets and the actual reached points is 1.63 ± 1.0 mm. Figure 6.14 shows a histogram of the reaching errors of this experiment. The evolution of the mean reaching error as well as the mean error estimate is shown in figure 6.15. The predicted mean error estimate after 1200 point is 1.79 ± 0.47 mm. One can see that in the beginning of the exploration, the error estimate is more conservative and the bound becomes tighter the more data is observed. Note that, since the inverse model is trained on 30 episodes each time a new data point is added, the total number of episodes the neural network is trained also increases over the number of collected data points.

Figures 6.13, 6.14, 6.15 all show the reaching error of one specific run of the experiment. To investigate the reproducibility of the point reaching accuracy, figure 6.16 visualizes the evolution of the mean and standard deviation of the mean reaching error of 5 repeated experiments with same parameters, but different random seeds over the number of collected data points. With an actual mean error after 1200 data points of 1.50 ± 0.42 mm and a mean predicted error estimate of 2.07 ± 0.40 mm, the point reaching accuracy is reproducibly low. In figure 6.16, one can also see that first, the error estimate is more conservative, both in terms of its mean and standard deviation. The more data is collected, the tighter the bound, again in terms of its mean and standard deviation. Therefore, the error estimate is also in practice a good measure for the performance of the learned inverse model.

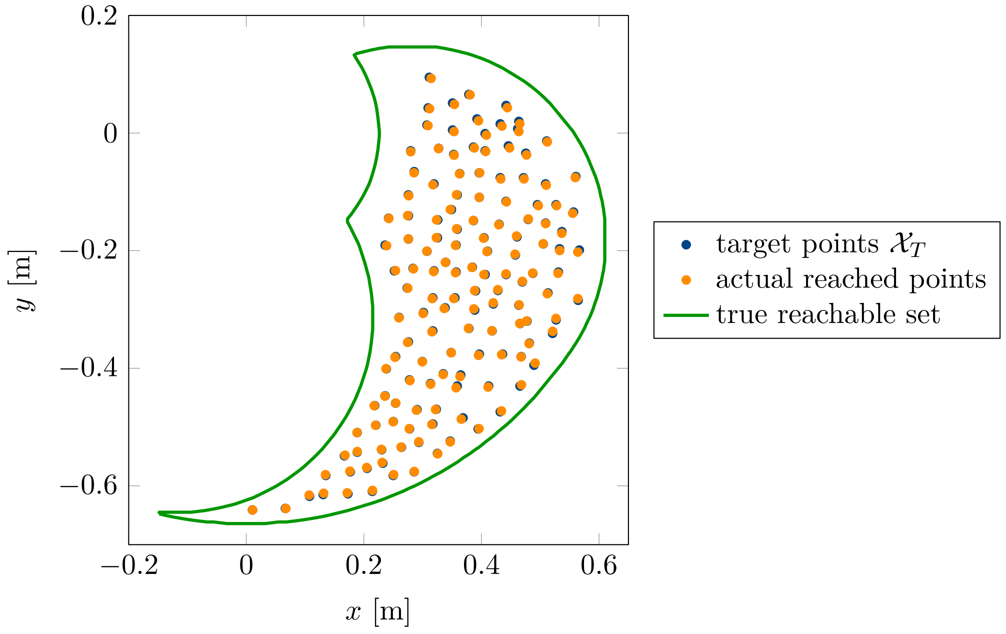


Figure 6.13: Point reaching evaluation with learned inverse $\boldsymbol{\pi}$ from section 6.3, 6.2. Actual reached points in orange with a mean error of 1.63 ± 1.0 mm.

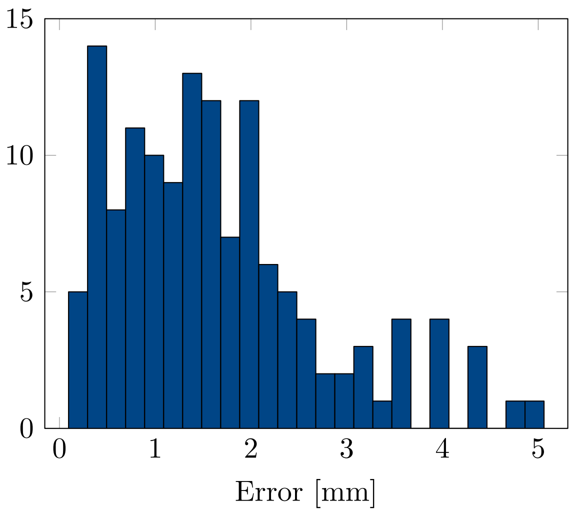


Figure 6.14: Histogram of reaching errors from figure 6.13.

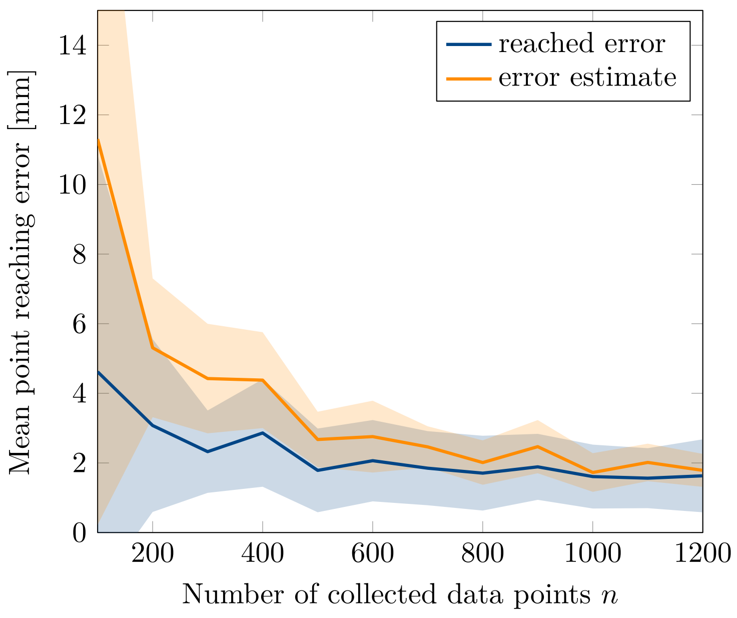


Figure 6.15: Evolution of the mean reaching error and the mean error estimate over the number of collected data points. The values correspond to one experiment shown in figure 6.13 with learned inverse $\boldsymbol{\pi}$ from section 6.3, 6.2. The final actual error after 1200 data points is 1.63 ± 1.0 mm, the error estimate predicts 1.79 ± 0.47 mm.

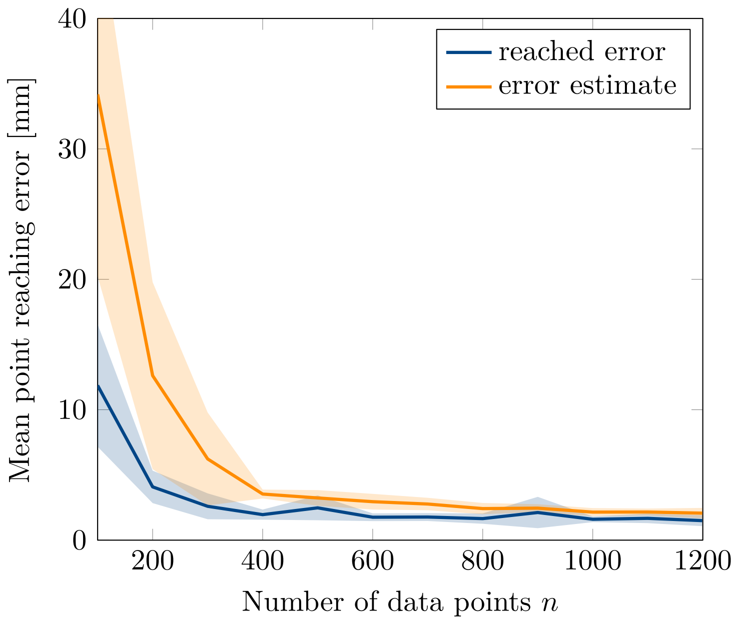


Figure 6.16: Evolution of the mean and standard deviation of the mean reaching error and the mean error estimate of 5 repeated experiments with the same parameters, but different random seeds over the number of collected data points. Final actual error after 1200 data points is 1.50 ± 0.42 mm, the error estimate predicts 2.07 ± 0.40 mm.

6.5. Out-Of-Center Reaching Trajectories

So-called out-of-center reaching motions are often studied in neuroscience when performing motor control experiments with humans and other animals. In this section, such out-of-center reaching experiments are performed with the learned inverse model from section 6.3, 6.2. It is investigated how the behavior changes the more data points are collected.

Figure 6.17 shows the starting position (blue), which is in the middle of the real workspace of the simulated arm model (green). The orange points denote the goal positions, which are spread around the starting position in a square with length of 20 cm. The arm is first moved to the start position by applying $\mathbf{u} = \boldsymbol{\pi}((0.4, -0.2))$ to the system until the equilibrium is reached. From this center position, a new muscle stimulated $\mathbf{u}_i = \boldsymbol{\pi}(\mathbf{x}_i)$ for each of the targets $\mathbf{x}_i$, $i = 1, \ldots, 8$ is applied.

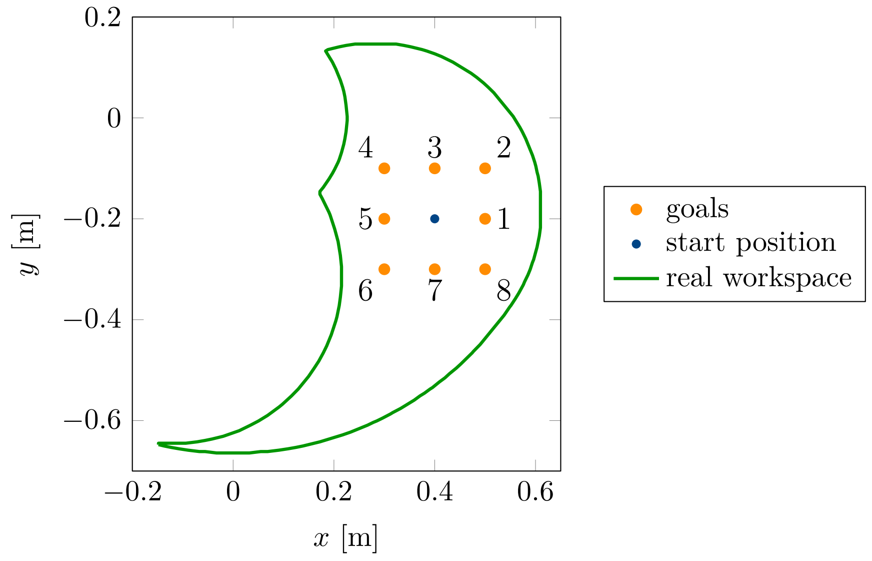


Figure 6.17: Out-of-center reaching experiment setup. The arm should reach the orange goal positions, starting from the blue position in the middle of the workspace each time.

In figures 6.18 – 6.25 the trajectories, actual reached positions and the corresponding error estimate are shown at different stages of the exploration, i.e. after a different number of collected data points. Furthermore, as a reference, these collected data points for training the forward and hence the inverse model are also visualized.

Starting with figure 6.18, where only 23 data points are collected, one can see that all trajectories move in the right direction. For targets 1, 2, 5, 7, 8, the actual error is already low. For target 4, the error and the error estimate is large, since there is no observed data in the vicinity of the target, but still the arm moves into the right direction, showing the extrapolation quality of the inverse model. For most targets, the error estimate is also large at this stage of the exploration. After 50 collected data points (figure 6.19), the error estimates for all targets are already much smaller. Target 3 and 6 also have a low error now. There is still the largest error on target 4, since in this area no new data has been observed. This trend continuous for 75 collected data points (figure 6.20). For 100 collected data points (figure 6.21), the error for target 4 is finally also small, since now there is observed data in its vicinity. From this stage of exploration on, mainly the error estimate gets smaller, see figures 6.22 – 6.25.

To further quantify these trends, figure 6.26 shows the reaching errors and their estimates for each target and each exploration stage. For example, one can see the decrease of both the error and error estimate of target 4. Generally, the decrease of the error estimate for all targets can be seen. Please note the different ordinate axis scales between figure 6.26a and 6.26b. With respect to the actual error, for most target points, one can also see a monotonic decrease in the actual error. However, looking at 6.26b, one can see that after 200 data points the reached errors for these 8 targets are lower then after 1200 data points. One has to take into account that the inverse model is a neural network, for which, since it is trained with stochastic gradient descent, one cannot expect monotonicity. The differences where this monotonicity is violated, compare 200, 300 and 1200 data points, are, however, not significant at all.

Furthermore, in figure 6.26 one can see that the error estimate is always an upper bound, which in the end, see figure 6.26b purple line, is even very tight. However, since the assumption that the true forward model, i.e. the musculoskeletal system, comes from the same RKHS as the kernel used for this experiment is extremely unrealistic, one cannot expect that the error estimate is always a true upper bound for the real error in practice. This out-of-center reaching experiment has been performed in addition to the shown results for 18 different stages of the exploration. In 5 of the $18 \cdot 8 = 144$ point reaches, the error estimate was slightly smaller than the true error, which corresponds to 3.5%, meaning that, although the assumption is unrealistic, in practice, the error estimate is a reasonable upper bound.

Looking at the trajectories of figures 6.18 – 6.25, one can see that the shape of the trajectories is nearly identical over the curse of the exploration procedure. This implies that the actual predicted muscle stimulations are only slightly changing over the exploration process, indicating a stable learning.

Finally, figure 6.27 shows the velocity profiles over time of the point reaching trajectories for the different targets with the inverse model after 1200 collected data points. One can see a characteristic accelerating and decelerating shape of the velocity.

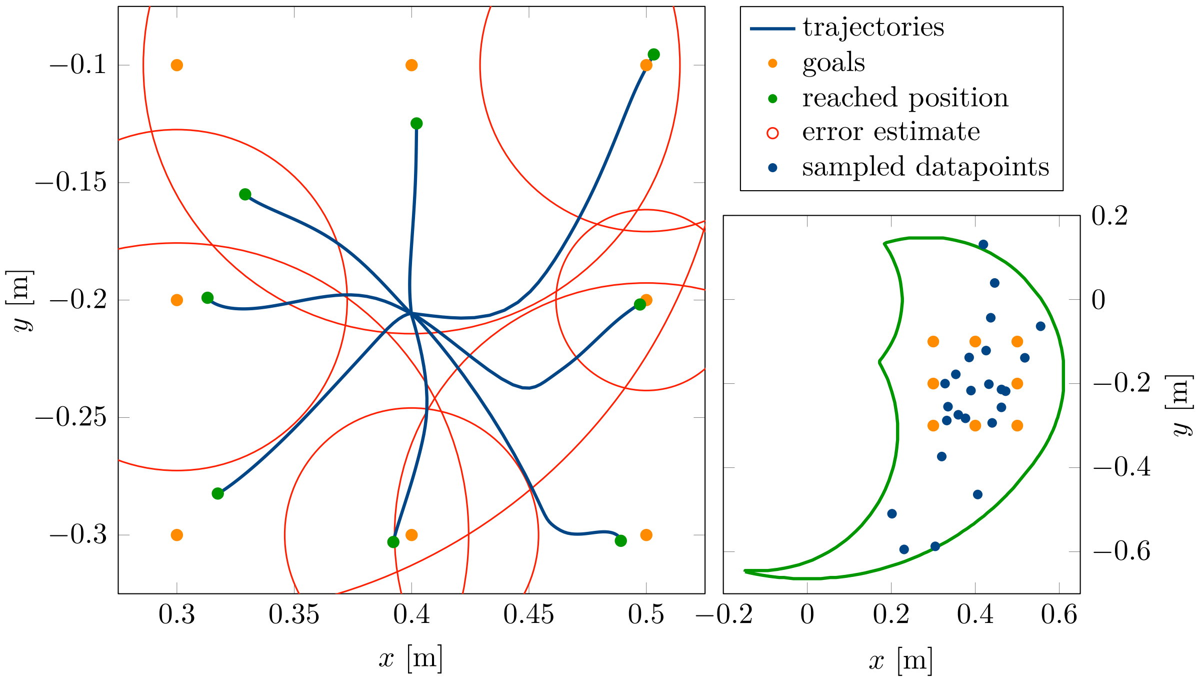


Figure 6.18: Out-of-center reaching experiment after 23 collected data points.

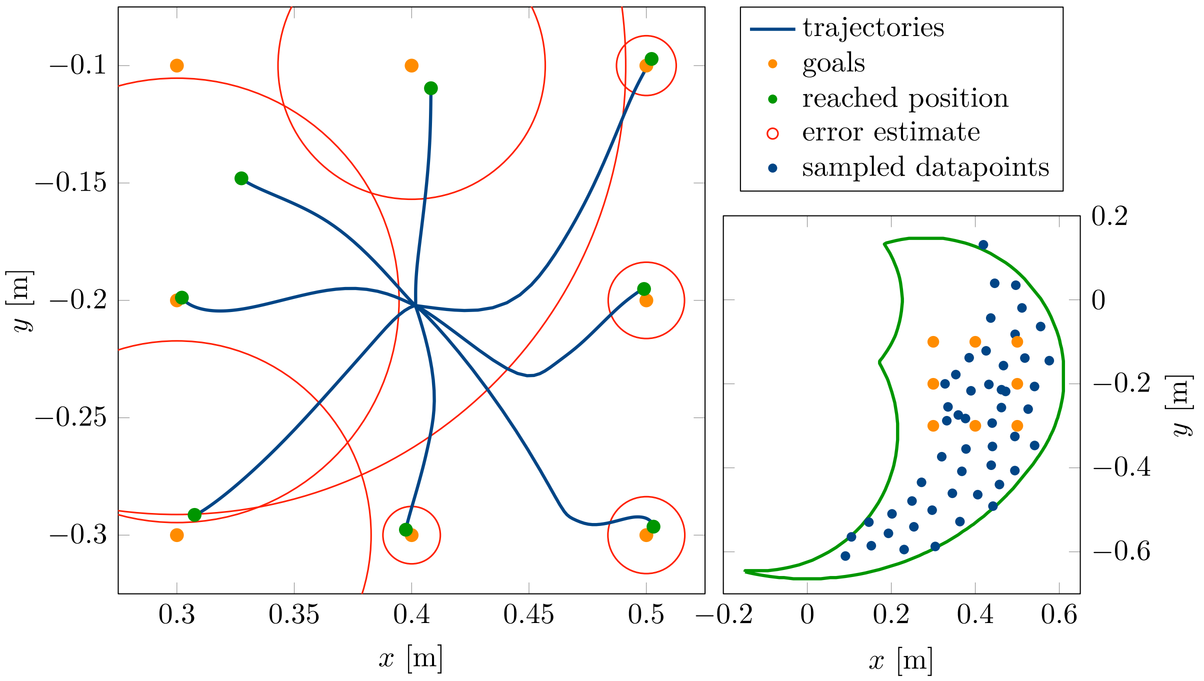


Figure 6.19: Out-of-center reaching experiment after 50 collected data points.

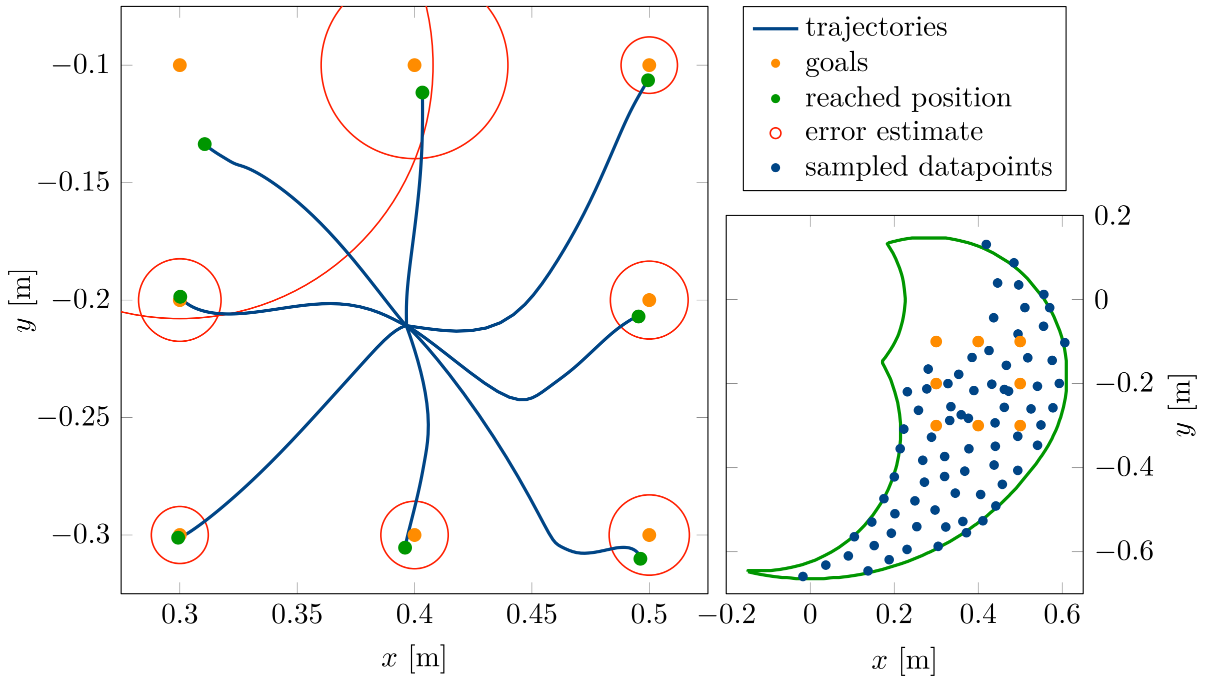


Figure 6.20: Out-of-center reaching experiment after 75 collected data points.

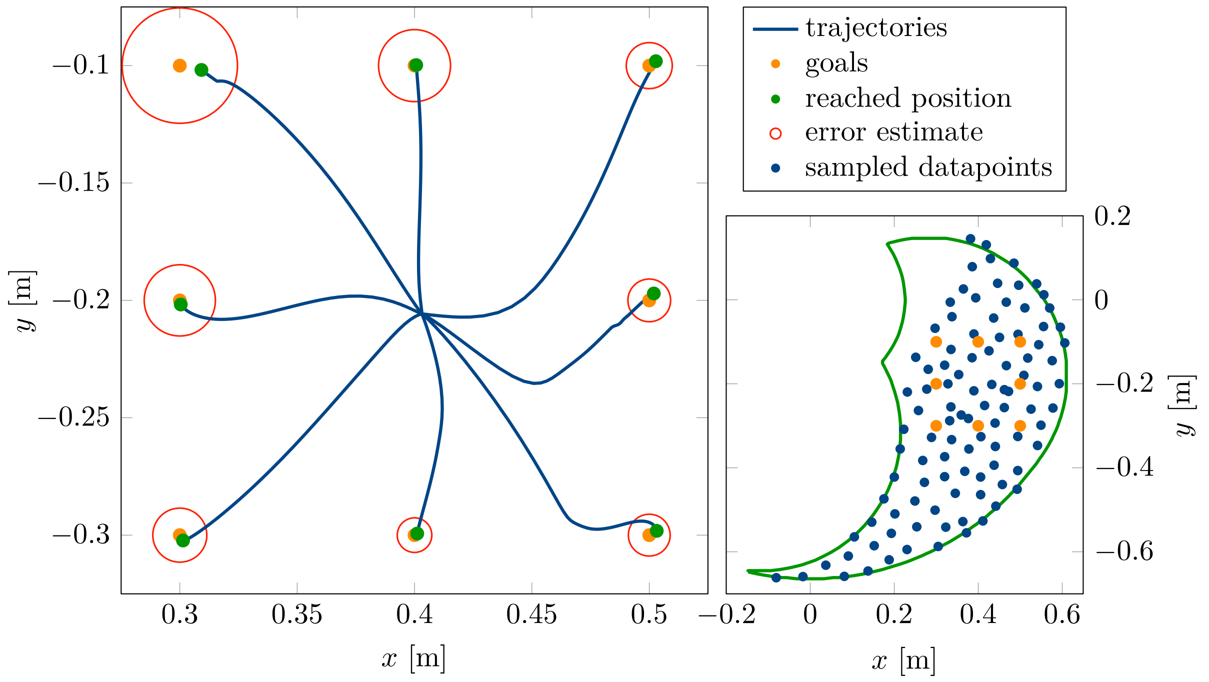


Figure 6.21: Out-of-center reaching experiment after 100 collected data points.

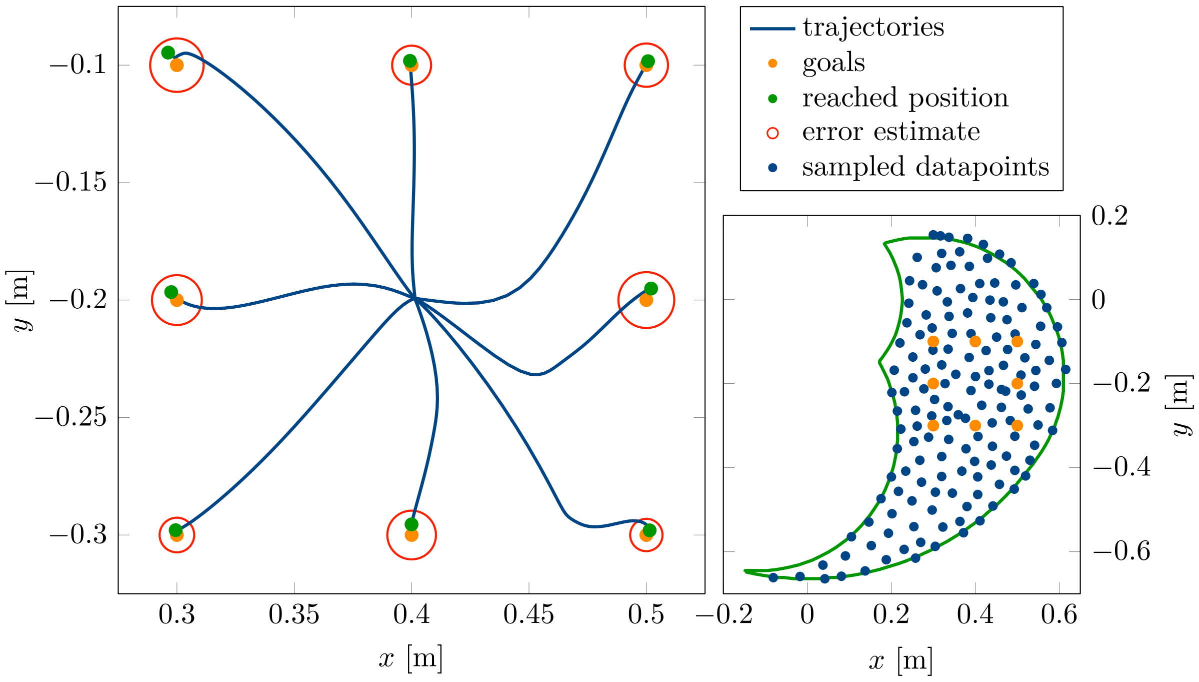


Figure 6.22: Out-of-center reaching experiment after 150 collected data points.

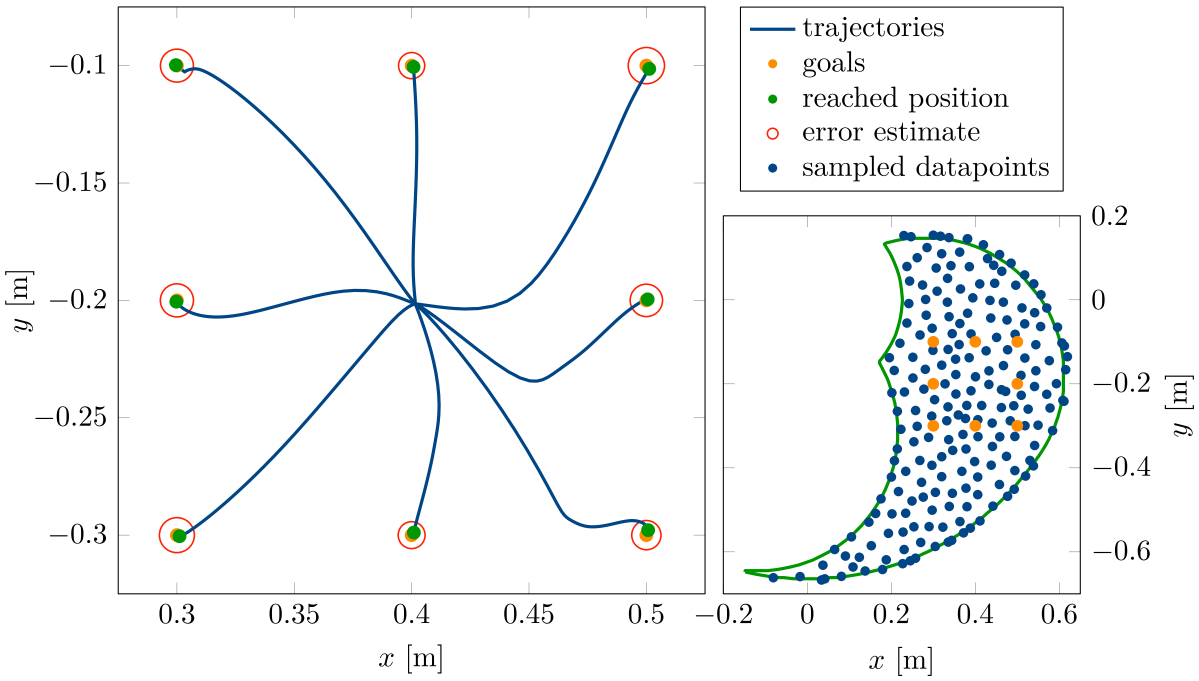


Figure 6.23: Out-of-center reaching experiment after 200 collected data points.

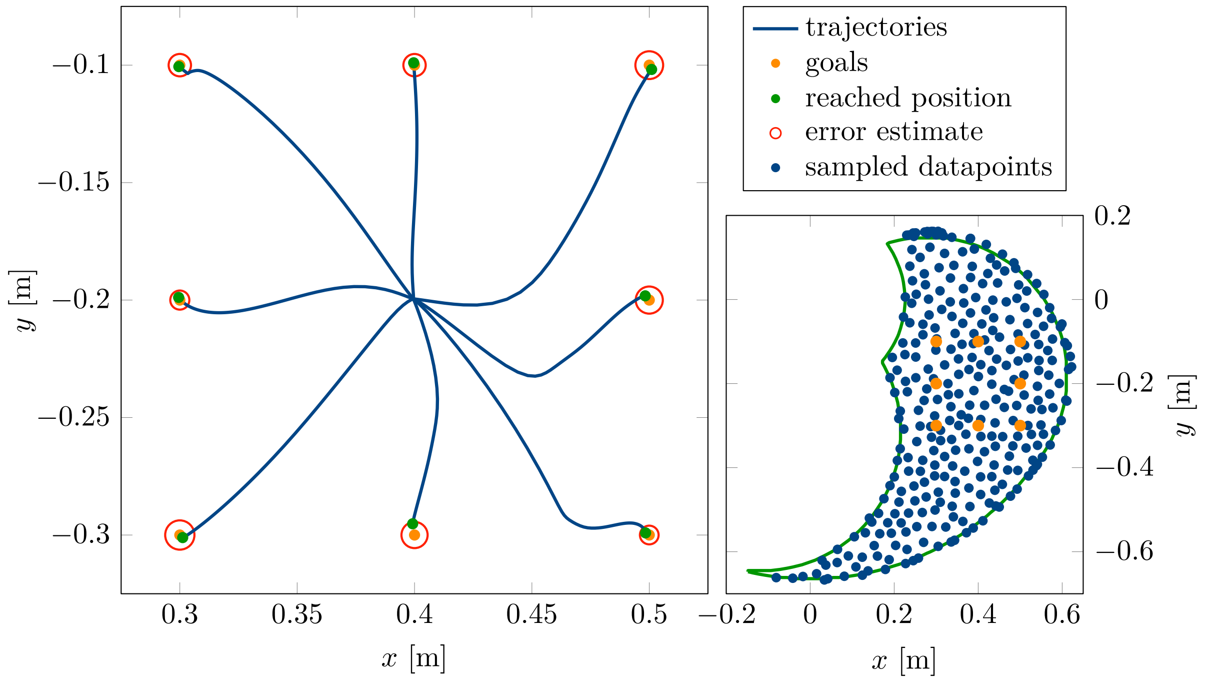


Figure 6.24: Out-of-center reaching experiment after 300 collected data points.

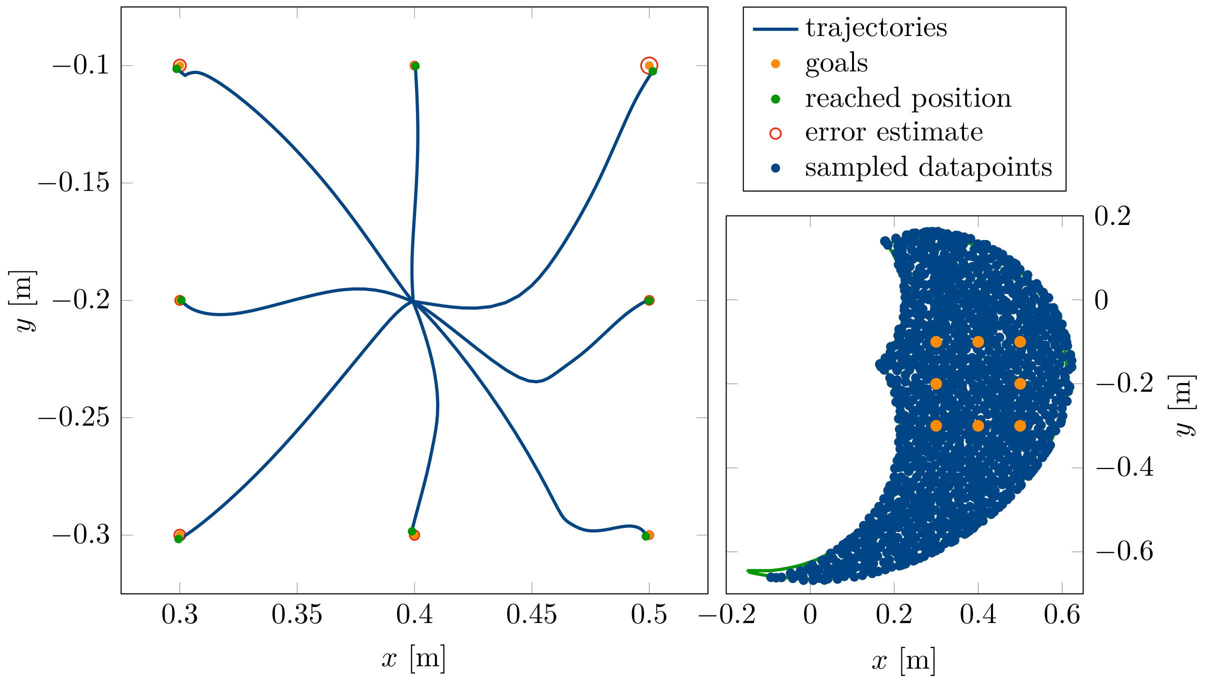
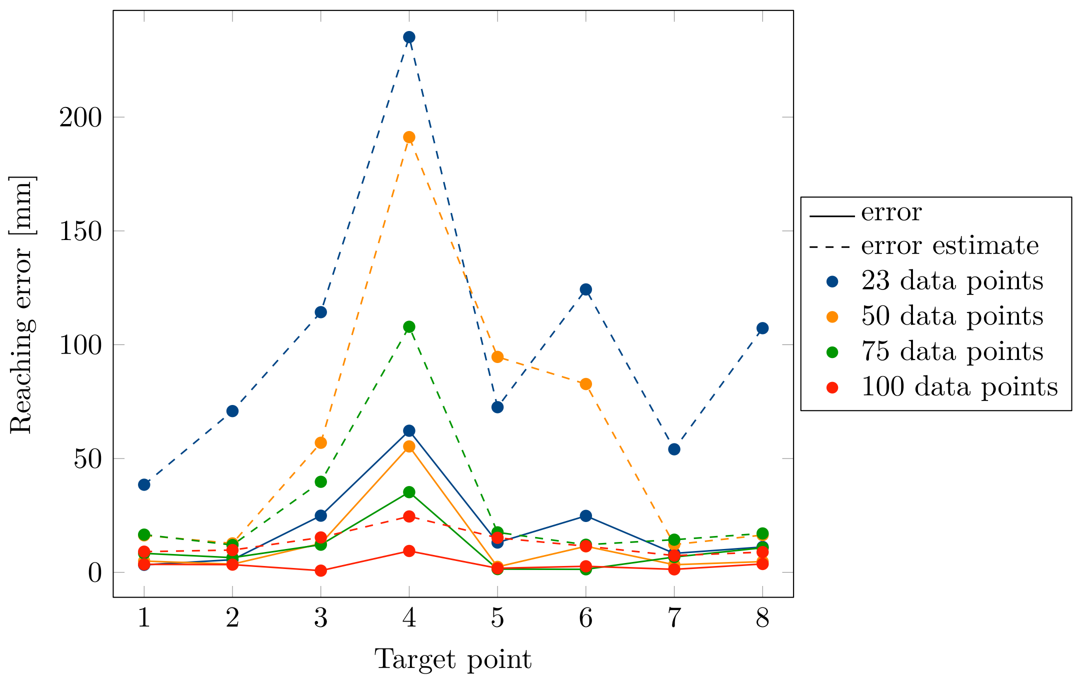
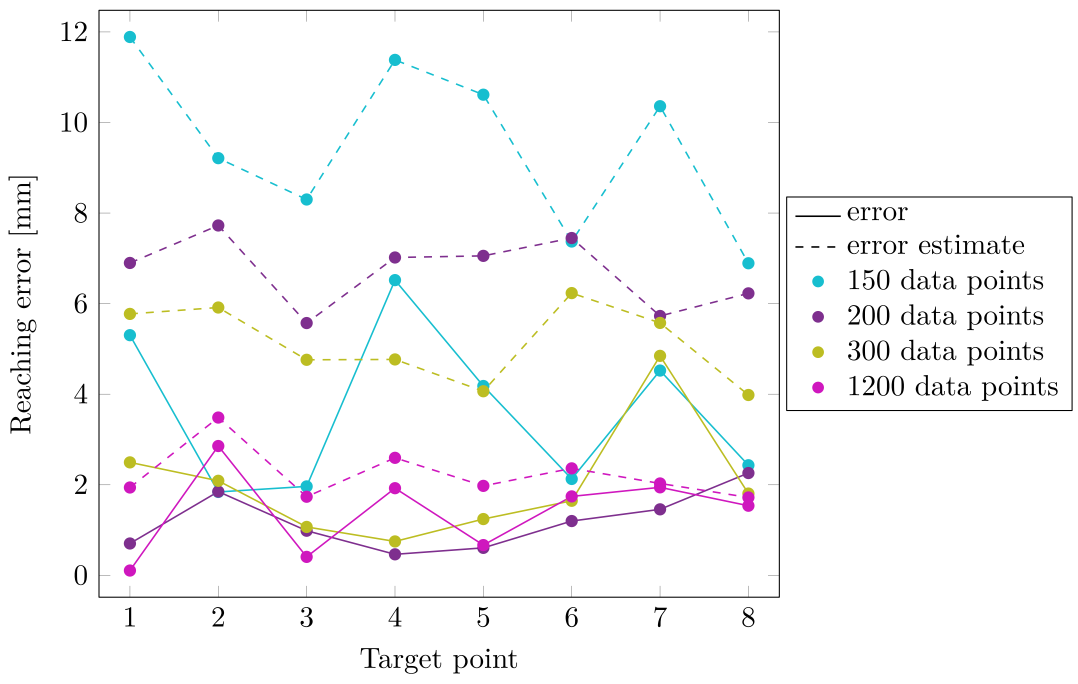


Figure 6.25: Out-of-center reaching experiment after 1200 collected data points.

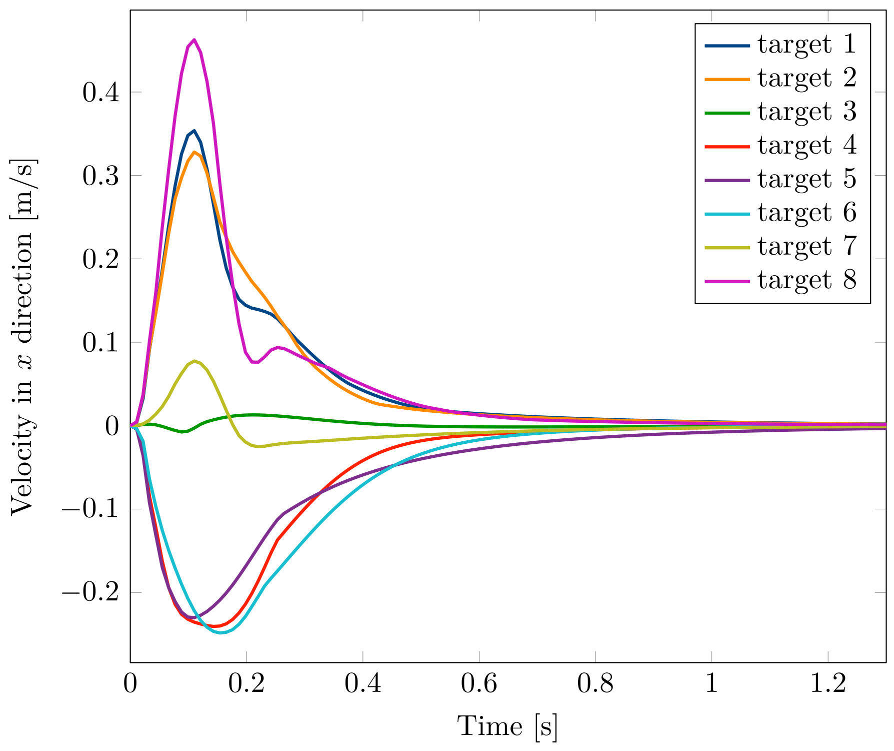


(a) Errors after 23, 50, 75 and 100 collected data points

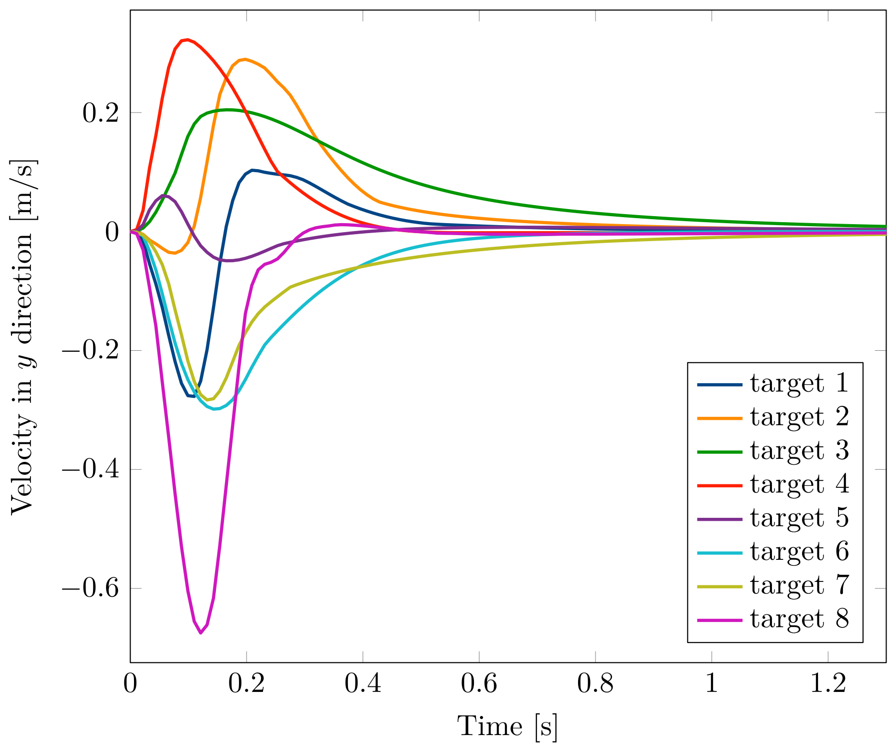


(b) Errors after 150, 200, 300, 1200 collected data points

Figure 6.26: Out-of-center reaching error and error estimate for the target points specified in figure 6.17. The different colors correspond to the out-of-center reaching experiment after a different number of data points are collected.



(a) Velocity in x-direction



(b) Velocity in y-direction

Figure 6.27: Velocity profiles of out-of-center reaching experiment for the different targets with the inverse model learned after 1200 collected data points.

6.6. Attractor Property

The whole methodology of this work was derived under the assumption that independent of the start configuration the musculoskeletal arm model always reaches a certain equilibrium configuration for a statically applied muscle stimulation. Abstractly, this was expressed by the fact that the true forward model $\widehat{\boldsymbol{\phi}}$ is a static, unique and state independent mapping from $\mathcal{U}$ to $\mathcal{X}$. In this experiment, it is shown that for the considered arm model of the present work, this property is indeed fulfilled. Furthermore, the behavior of the system, e.g. the concrete trajectory, between such switchings of static muscle stimulations is investigated.

Figure 6.28 shows 4 different attractor target positions as well as 6 different starting positions. The arm is first moved to the starting positions by applying the constant muscle stimulations predicted by the learned inverse model from section 6.3, 6.2. When the equilibrium is reached, the muscle stimulations are switched to a new constant one as predicted by learned inverse model to reach an attractor target position.

In figure 6.29, the resulting trajectories in the x, y-space of this experiment are visualized. As one can see, independent from the starting position, the attractor targets are reached. For attractor point 1 (figure 6.29a), which is in the middle of the workspace, there is little to no overshot. For attractor point 2 (figure 6.29b), which is at the top of the workspace, there is again little to no overshoot for all starting positions, except for starting position 1, which starts from the bottom left of the workspace and therefore has the highest distance to the target. A similar observation holds true for attractor point 3 (figure 6.29c). Looking at the trajectories for attractor point 4 (figure 6.29d), which is at the lower part of the workspace, especially if starting from the top of the workspace (starting position 4 and 5), one can see a larger overshoot. This can be explained by looking at the muscle stimulation distribution predicted by the learned inverse model as a function of the desired position as shown in figure 6.7. When switching from a position at the top of the workspace to the attractor target 4, the stimulations in the flexor muscles change from high to low, while in the extensor muscles the stimulation is increased (from a low value). Together with gravity, this leads to an acceleration towards the bottom of the workspace. However, the visco-elastic properties of the MTUs, as discussed in section 3, will then counteract this acceleration such that, despite the overshoot, the attractor point is reached without any further control action.

To further quantify also the temporal behavior of these trajectories, figure 6.30 plots the trajectories of figure 6.29 over time. Here the behavior described above can be seen more clearly. For attractor target 1, 2 and 3, it takes between 0.5 s and 1.0 s to reach the equilibrium. For attractor target 4, one can see the larger overshoot when starting from start position 3, 4 and 5. For those it takes about 1.5 s to reach the equilibrium.

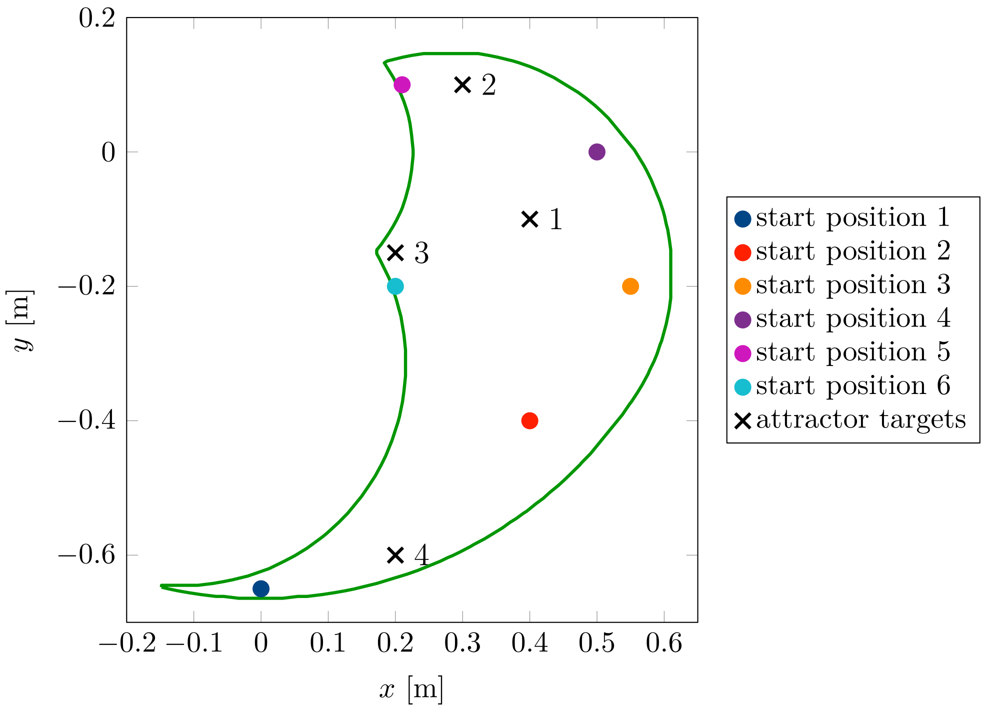
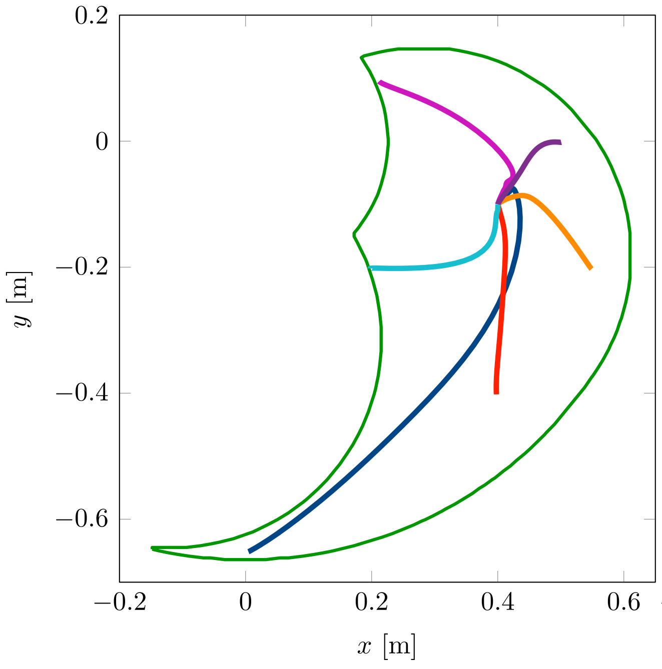
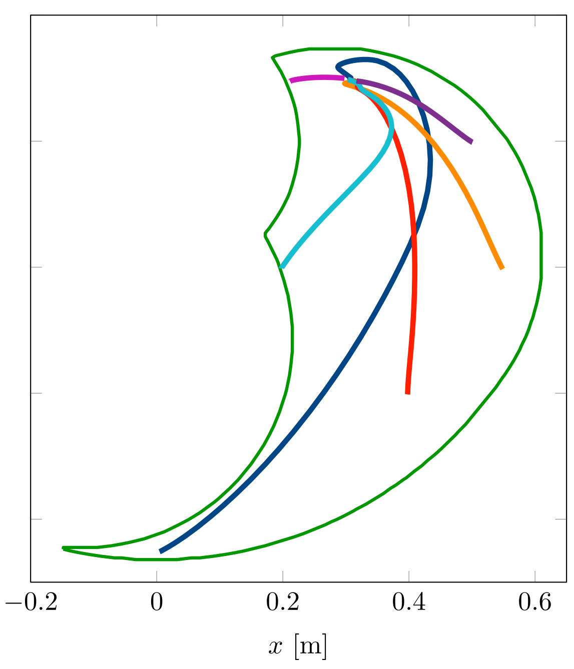


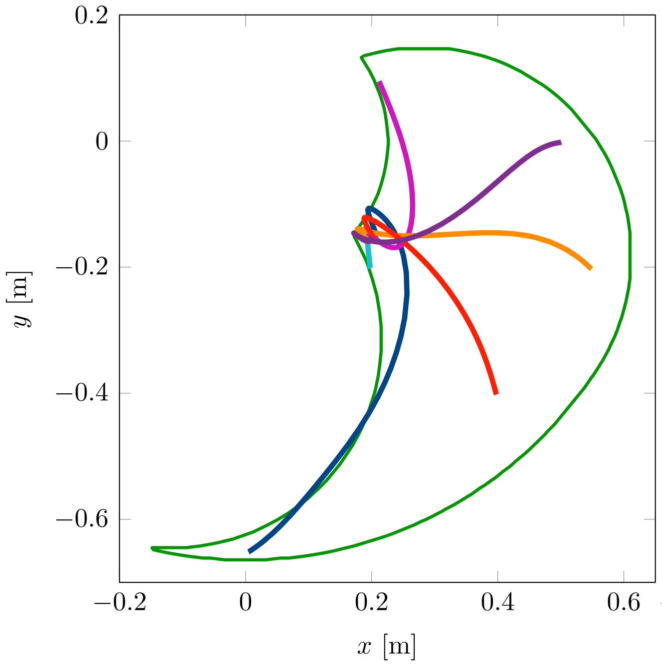
Figure 6.28: Attractor property experiment. Black crosses are target points which should be reached from the different start positions (colored points).



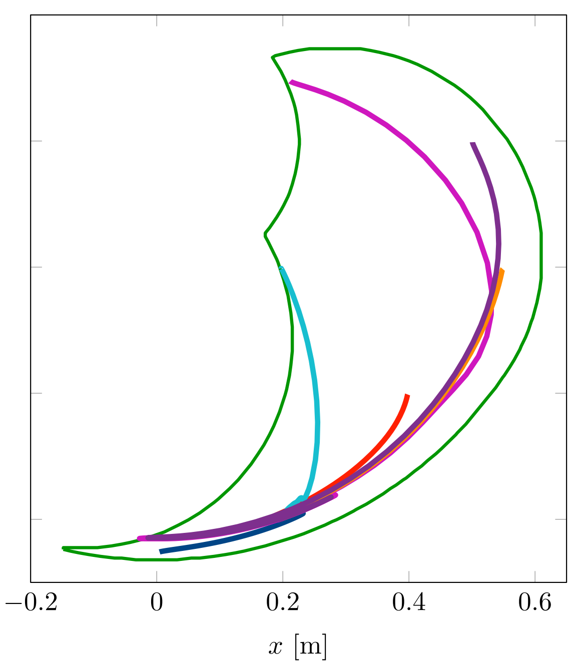
(a) Target attractor 1



(b) Target attractor 2

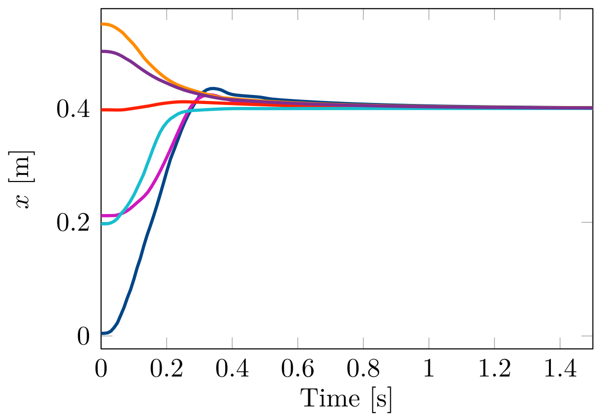


(c) Target attractor 3

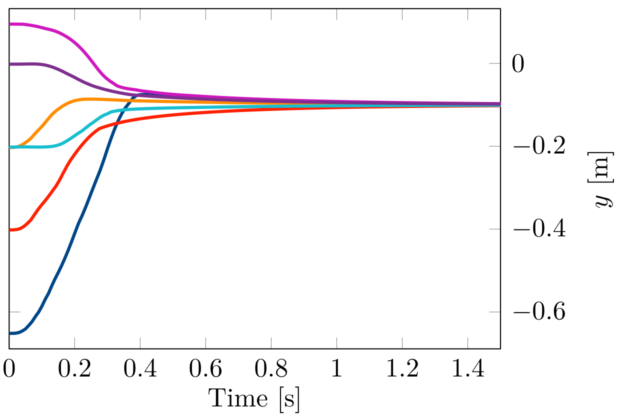


(d) Target attractor 4

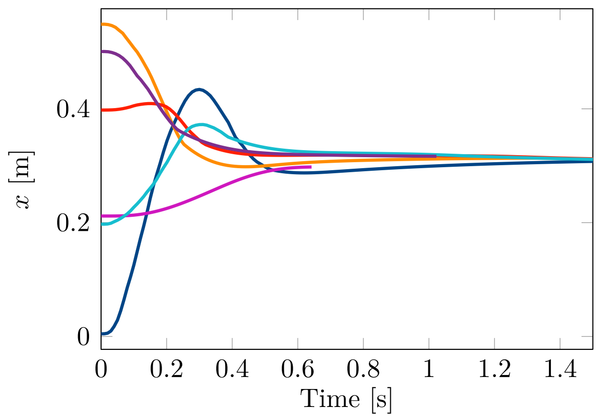
Figure 6.29: Observed trajectories from the starting positions to the attractor targets. Learned inverse model $\boldsymbol{\pi}$ from section 6.3, 6.2.



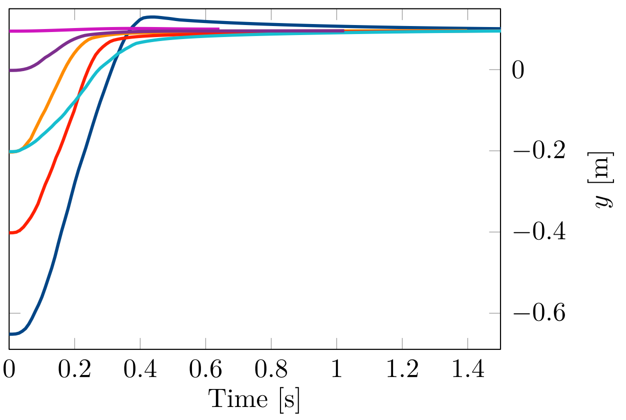
(a) Attractor target 1, x-direction



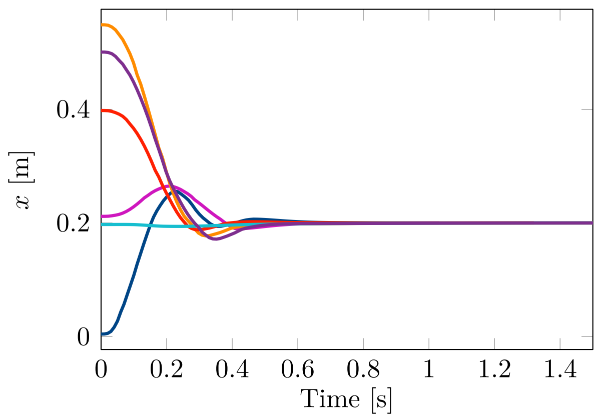
(b) Attractor target 1, y-direction



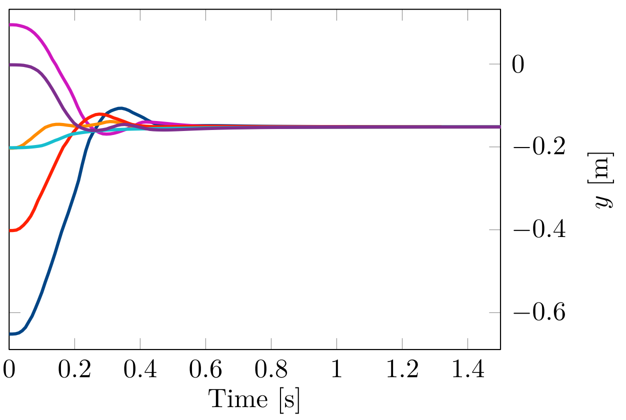
(c) Attractor target 2, x-direction



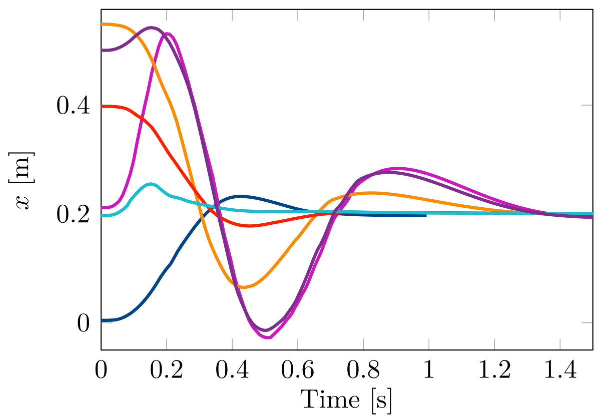
(d) Attractor target 2, y-direction



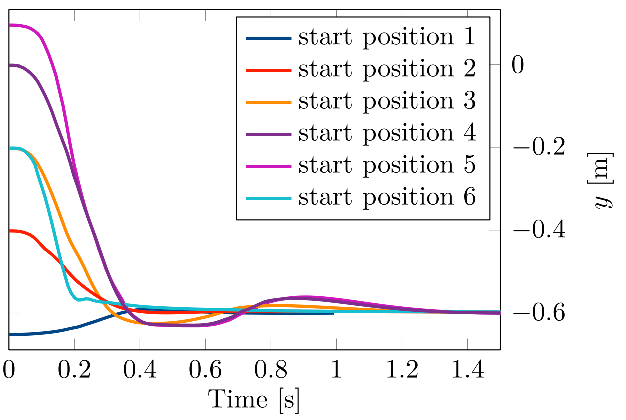
(e) Attractor target 3, x-direction



(f) Attractor target 3, y-direction



(g) Attractor target 4, x-direction



(h) Attractor target 4, y-direction

Figure 6.30: Trajectories of figure 6.29 plotted over time.

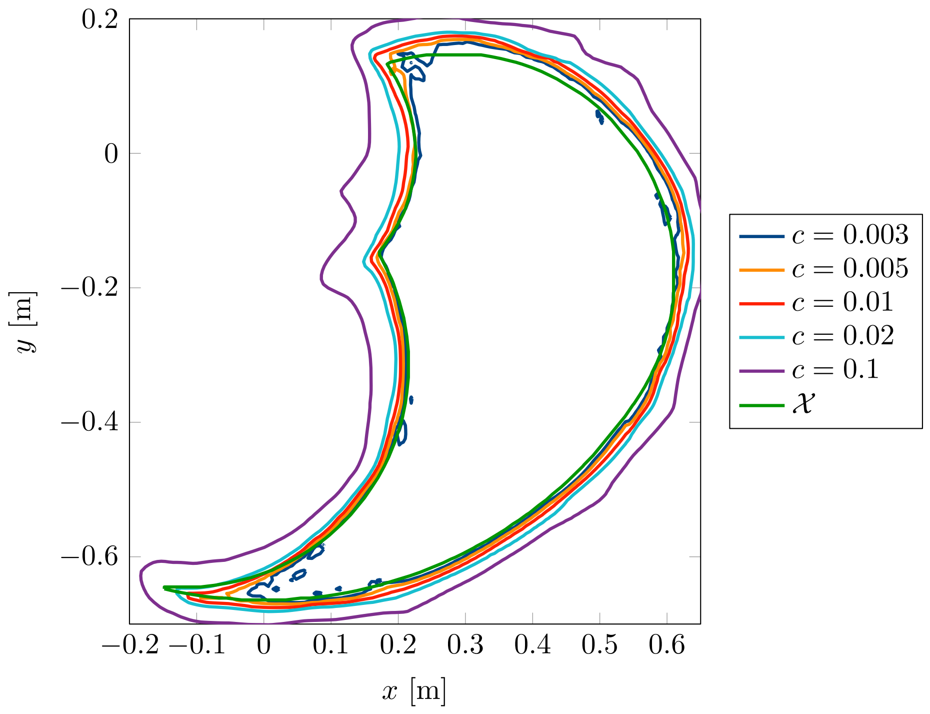
6.7. Reachable Set

This experiment aims at investigating the estimation of the reachable set in more depth. The evolution of the estimated reachable set over the number of collected data points has already been visualized in figure 6.2, where the error certainty is $c = 0.02$, meaning that for each desired target position $\mathbf{x} \in \mathcal{X}_c$, the learned inverse model is able to predict a muscle stimulation $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ such that when applied to the system, the arm will reach a position that has a maximum distance to the desired target of 20 mm. As has been evaluated in section 6.4, the point reaching accuracy for the target points of figure 6.12 with a mean of 1.50 ± 0.42 mm is much better than 20 mm. Indeed, as shown in figure 6.11 and as discussed in section 6.3, the error estimate for the majority of the reachable set is below 3 mm. More explicitly, figure 6.31 shows the boundary of the estimated reachable set $\mathcal{X}_c$ for different values of the error certainty c. One can see that up to $c = 0.003$, i.e. 3 mm error certainty, the estimated reachable set is nearly the complete true $\mathcal{X}$, except for the lower left corner, which corresponds to the observation of figure 6.11. For even smaller values of c like $c = 0.0015$ or $c = 0.002$, i.e. 2 mm and 1.5 mm error certainty, the estimated reachable set $\mathcal{X}_c$ starts to get smaller and contain holes. But still for these low values, $\mathcal{X}_c$ covers a large part of $\mathcal{X}$. This indicates that the learned inverse model is able to minimize the upper bound on the quality of the inverse model 5.67 tightly and uniformly over the complete reachable set.

Note that according to the theory about the estimated workspace (section 5.3), it must hold $\mathcal{X}_c \subset \hat{\mathcal{X}}_c$. However, it is not realistic to assume that the true musculoskeletal system as $\widehat{\boldsymbol{\phi}}$ is an element of the RKHS with a Gaussian kernel and its hyperparameters. Therefore, one cannot expect that the subset property strictly holds. Indeed, in figure 6.1 one can see that in the upper right part there is a slight violation of the subset property. However, this violation is very tiny, which is another indicator for the fact that the made assumptions can reasonably well be fulfilled in practice without extensive parameter tuning of for example β_i. Furthermore, similar to the discussion in section 6.4, the error estimate is also tight and hence the estimated reachable set $\mathcal{X}_c$ is close to $\hat{\mathcal{X}}_c$.

As has been discussed in section 5.3, proposition 5.32 provides a different way to obtain the reachable set based on the available information of the learned forward model only. The idea is to directly evaluate the learned forward model in the complete muscle stimulation space $\mathcal{U}$ and then use the error estimate of the forward model (theorem 4.16) to define the set $\mathcal{X}_c^{\mathcal{U}}$ for the error certainty c. To calculate this set, $\mathcal{U} = [0, 1]^6$ is discretized with 25 points per dimension, meaning that $\boldsymbol{\phi}$ is evaluated for a total of $244140625 \approx 244 \cdot 10^6$ muscle stimulations. The evaluation of the learned $\boldsymbol{\phi}$ on this discretized $\mathcal{U}$ can be done on modern hardware in a few hours. To put this into perspective, doing these many simulations to evaluate the true forward model $\widehat{\boldsymbol{\phi}}$ on this muscle stimulation grid on the computer of the author of this document would take about 8 years.

Figure 6.32, 6.33 and 6.34 show $\mathcal{X}_c^{\mathcal{U}}$ after 1200, 100 and 150 collected data points for different values of c as well as $\boldsymbol{\phi}(\mathcal{U})$. One can see three aspects. First, looking at figure 6.32a, 6.33a and 6.34a, only using $\mathcal{X} \approx \boldsymbol{\phi}(\mathcal{U})$ without taking into account that $\boldsymbol{\phi}$ is learned on very little data compared to the dimensionality of $\mathcal{U}$ leads to a heavily overestimated and hence completely wrong estimation of the reachable set. One reason for this is that at the boundary of the true reachable set some data points in the $\mathcal{U}$-space are close to each other. If the separation distance of the data points is small, the coefficients of the kernel linear combination (see section 4.1) become very large, leading to oscillations outside the observed data points. Secondly, using $\mathcal{X}_c^{\mathcal{U}}$ leads to a reasonable estimation of the reachable set which is very similar to $\mathcal{X}_c$, as can be seen in figure 6.32d for $c = 0.02$. In principle, $\mathcal{X}_c^{\mathcal{U}}$ has the chance of being a better estimate than $\mathcal{X}_c$, since it uses the complete information encoded in the learned forward model and not only the predictions through the current learned inverse model, since $\boldsymbol{\pi}(\mathbb{X}) \subset \mathcal{U}$. However, thirdly, the representation of $\mathcal{X}_c$ is not only much more useful, since it can be computed very quickly, compared to the computational demand of $\mathcal{X}_c^{\mathcal{U}}$, the estimation quality of $\mathcal{X}_c$ is at least the same as of $\mathcal{X}_c^{\mathcal{U}}$. Furthermore, as can be seen for example in figure 6.34d or 6.33d, $\mathcal{X}_c^{\mathcal{U}}$ on this already relatively dense gird of $\mathcal{U}$ still contains many empty spots, compared to $\mathcal{X}_c$, which could be evaluated on a much finer scale and hence does not suffer from this.



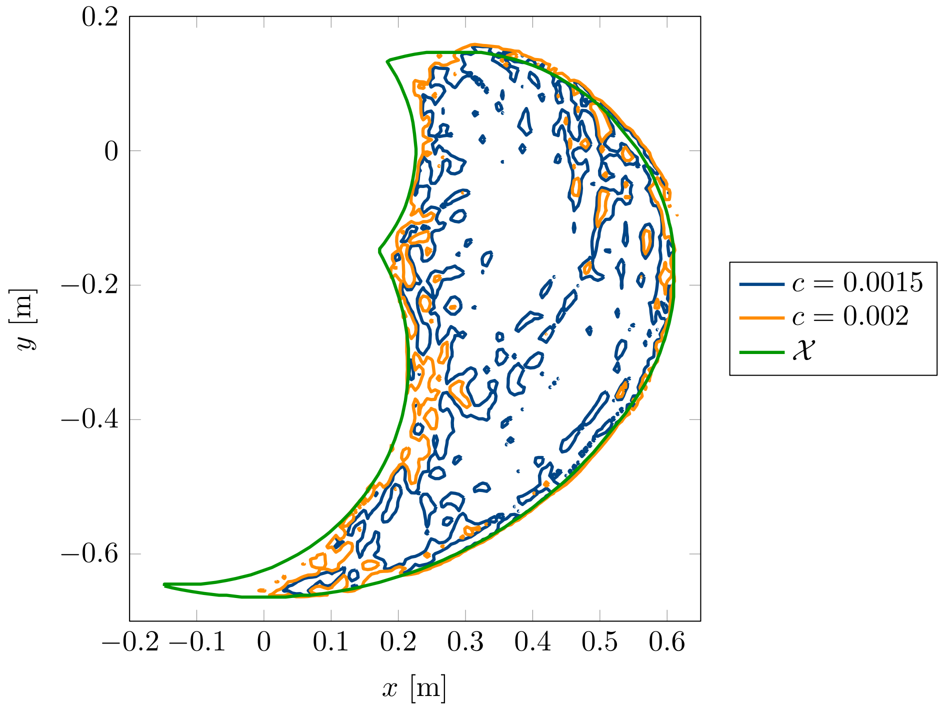
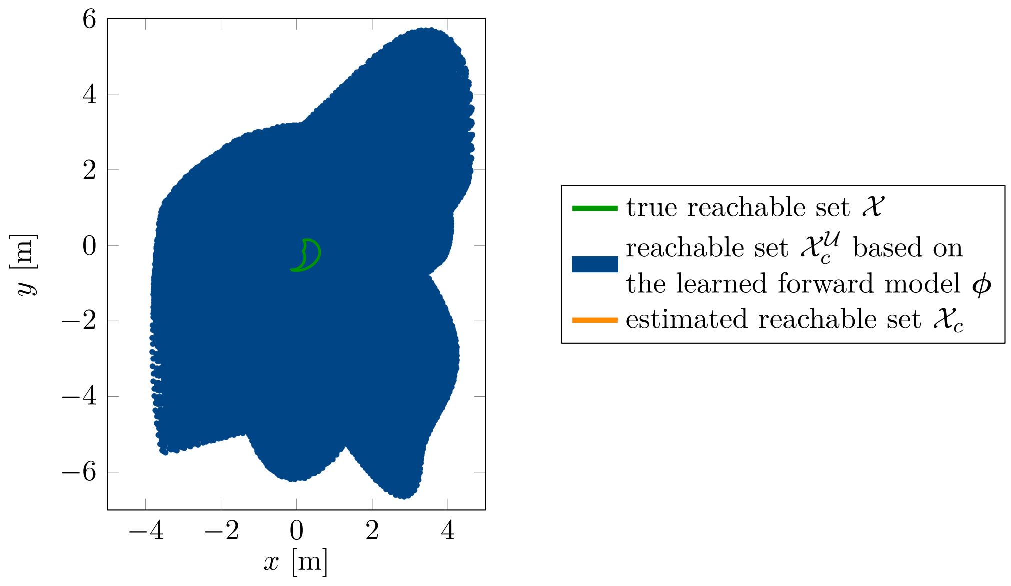
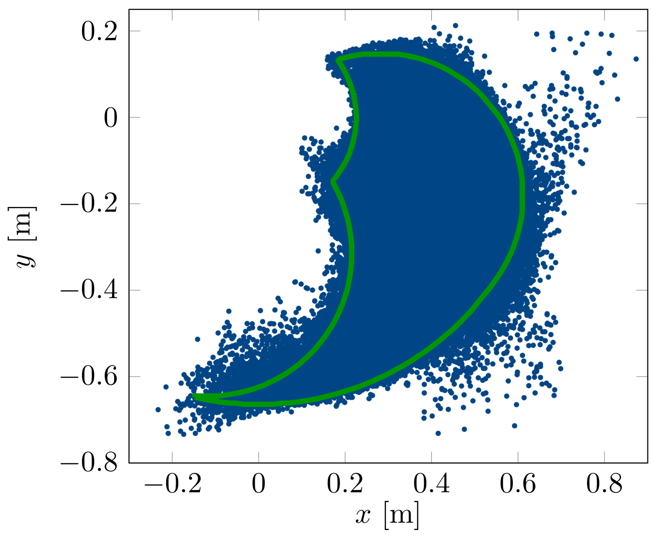


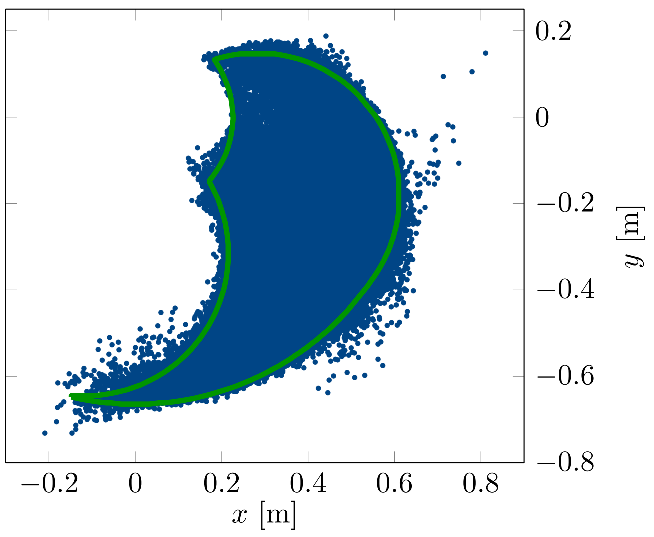
Figure 6.31: Estimated reachable set $\mathcal{X}_c$ for different values of the error certainty c after 1200 collected data points.



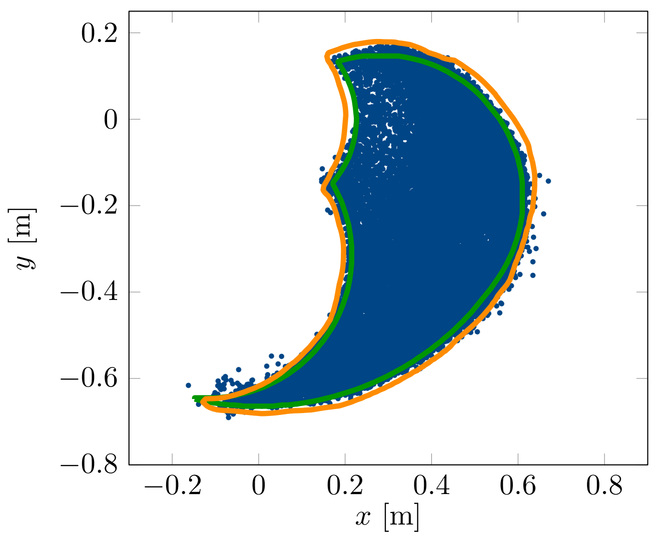
(a) $\phi(\mathcal{U})$, i.e. $c = \infty$



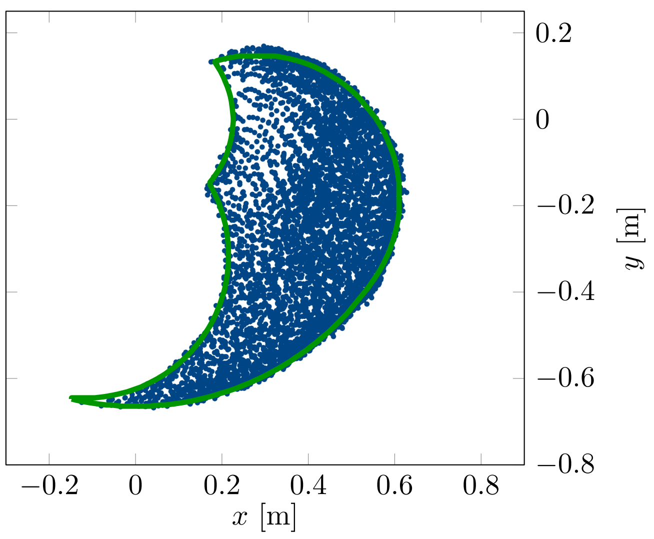
(b) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.04$



(c) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.03$

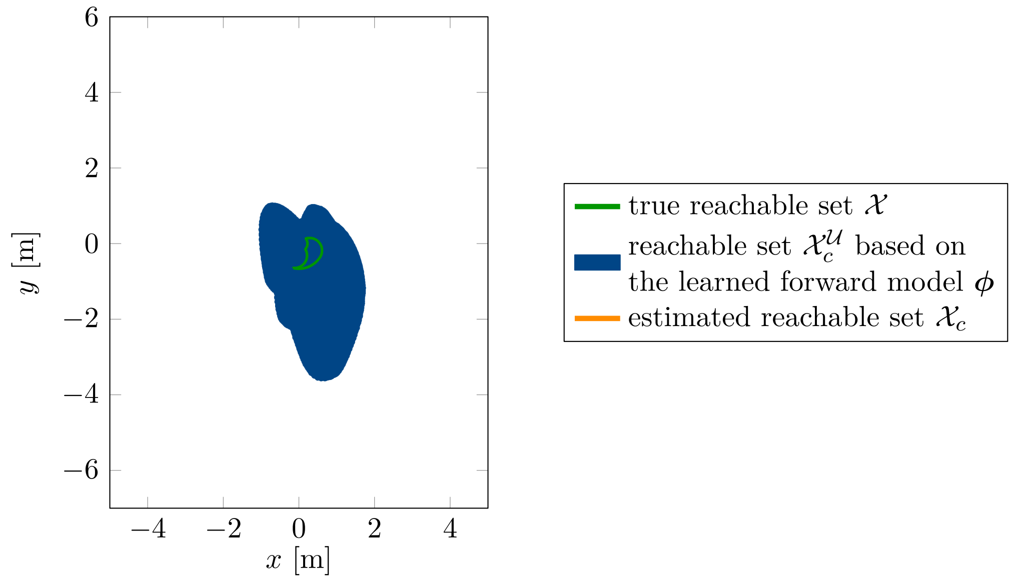


(d) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.02$

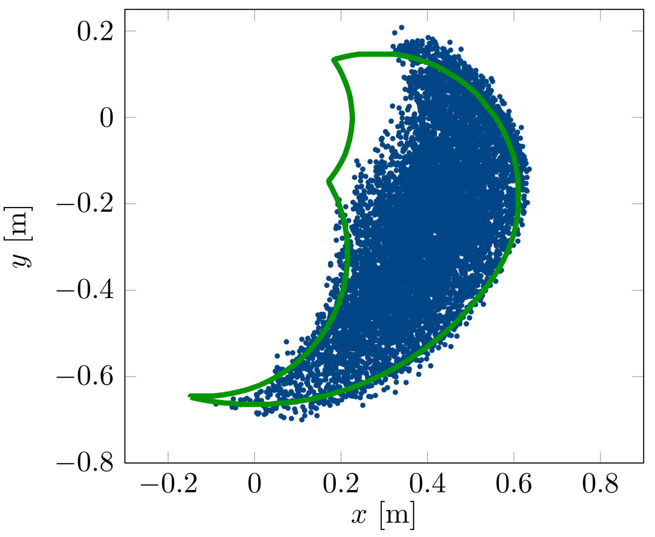


(e) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.01$

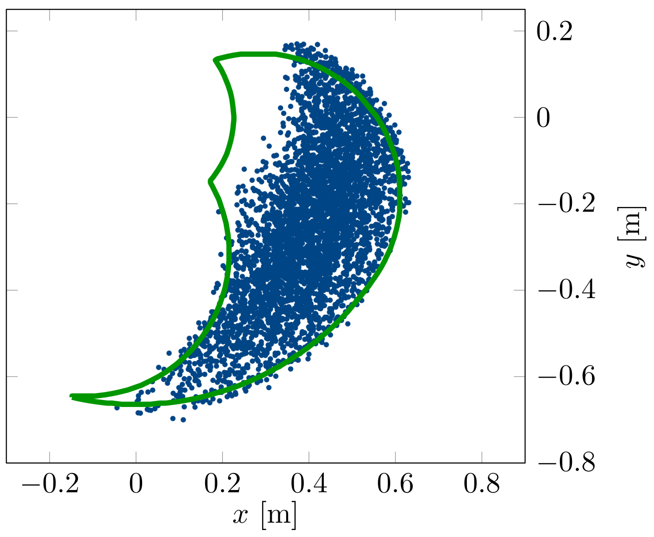
Figure 6.32: Reachable set $\mathcal{X}_c^{\mathcal{U}}$, calculated by evaluating $\phi(\mathcal{U})$ on a grid in $\mathcal{U}$, where ϕ is trained on 1200 collected data points.



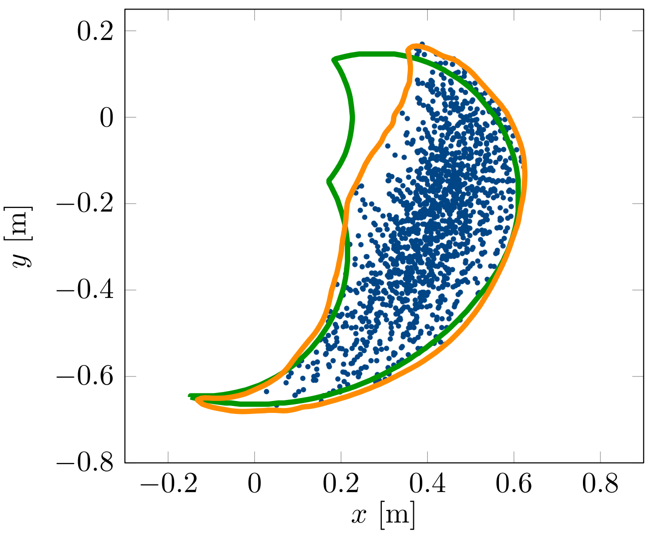
(a) $\phi(\mathcal{U})$, i.e. $c = \infty$



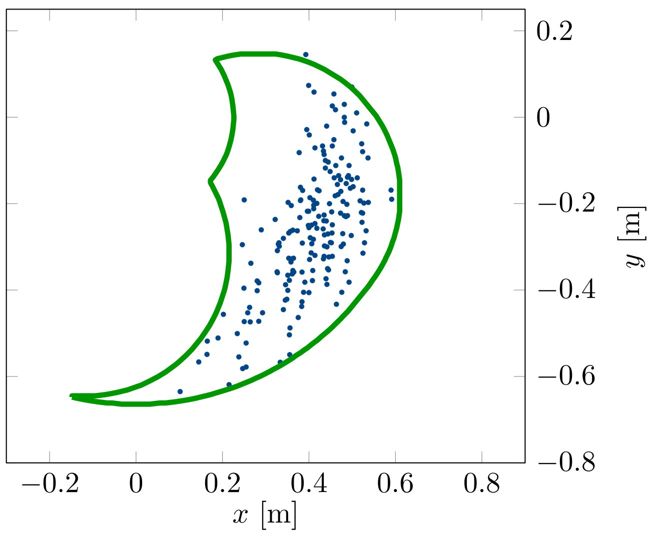
(b) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.04$



(c) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.03$

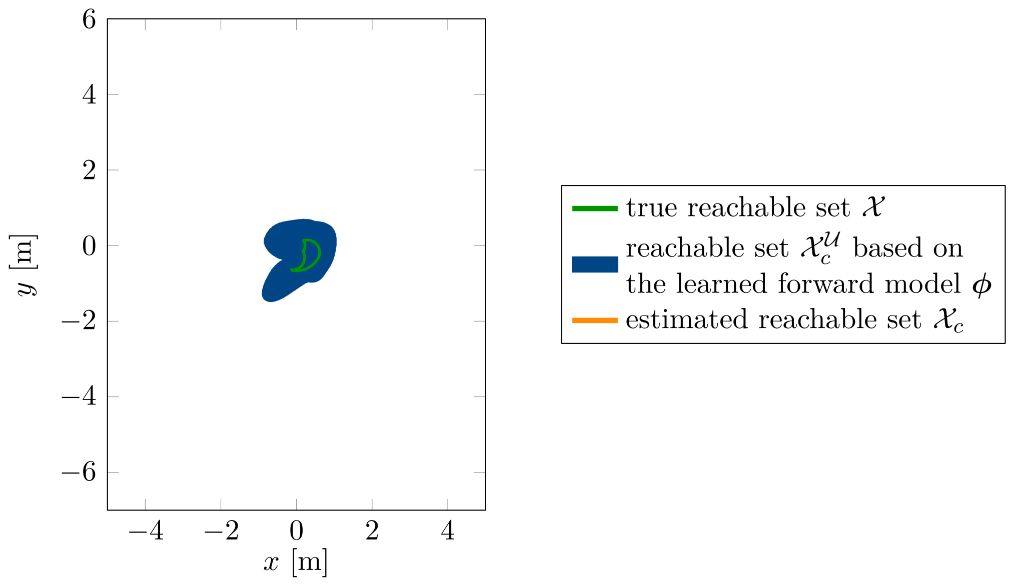


(d) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.02$

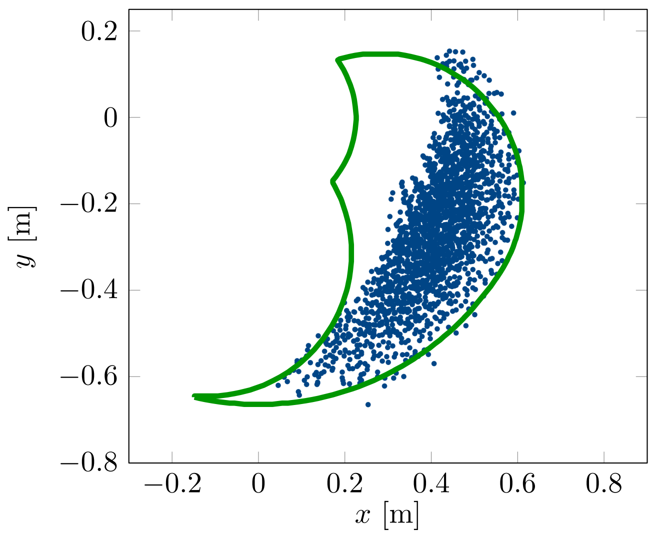


(e) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.01$

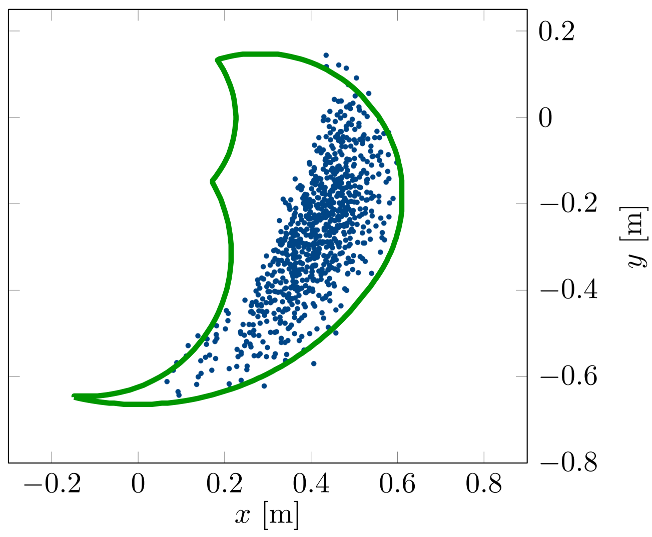
Figure 6.33: Reachable set $\mathcal{X}_c^{\mathcal{U}}$, calculated by evaluating $\phi(\mathcal{U})$ on a grid in $\mathcal{U}$, where ϕ is trained on 100 collected data points.



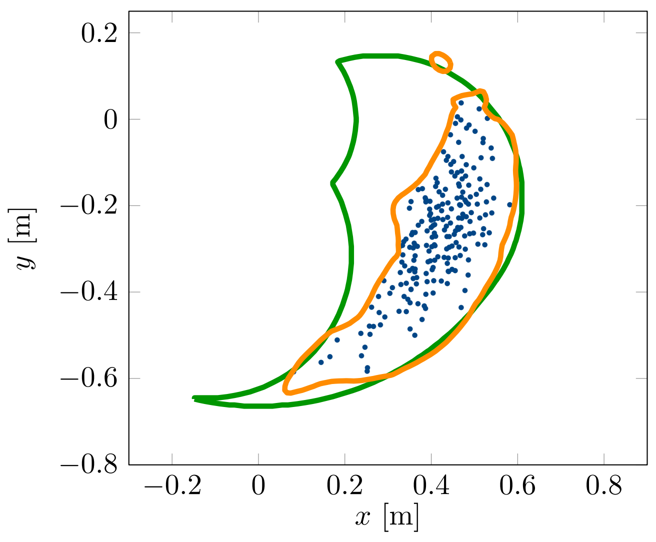
(a) $\phi(\mathcal{U})$, i.e. $c = \infty$



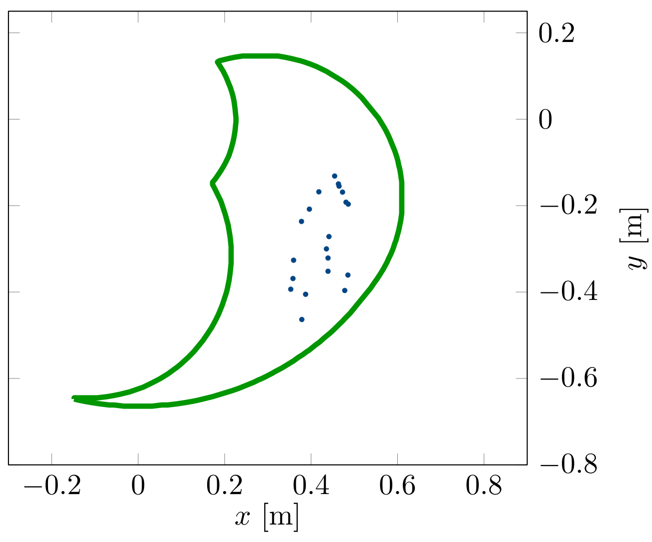
(b) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.04$



(c) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.03$



(d) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.02$



(e) $\mathcal{X}_c^{\mathcal{U}}$ for $c = 0.01$

Figure 6.34: Reachable set $\mathcal{X}_c^{\mathcal{U}}$, calculated by evaluating $\phi(\mathcal{U})$ on a grid in $\mathcal{U}$, where ϕ is trained on 50 collected data points.

6.8. Ablation Study

In this section, the importance and influence of the various parts of the proposed active inverse model learning framework on the point reaching performance is investigated. The point reaching targets are the same as for the point reaching evaluation experiment (section 6.4), see figure 6.12. Except for the part that is changed or left out, the rest of the methodology stays the same with the same parameters.

6.8.1. Neglecting the Upper Bound in the Inverse Learning Objective

Here the claim from section 5.2 is supported that a learned forward model can be misleading for learning an inverse, if it is not taken into account that a learned forward model does not approximate the true forward model everywhere in the control input space. In order to do so, it is considered what happens if the inverse model is not learned by optimizing the upper bound (5.38) on the real performance, but only as an inverse to the learned forward model. Formally, this means that the optimization problem for learning the inverse $\boldsymbol{\pi}$ is

$$\min_{\mathbf{w}\in\mathcal{W}} \int_{\mathcal{X}_c} \left\|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w})) - \mathbf{x}\right\|^2 + \eta \left\|\boldsymbol{\pi}(\mathbf{x};\mathbf{w})\right\|^2 \mathrm{d}\mathbf{x} \tag{6.1}$$

or in its actual implementation

$$\min_{\mathbf{w}\in\mathcal{W}} \sum_{\mathbf{x}\in\tilde{\mathcal{X}}_c\cup\mathcal{D}_{\mathcal{X}}} \left\|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w})) - \mathbf{x}\right\|^2 + \eta \left\|\boldsymbol{\pi}(\mathbf{x};\mathbf{w})\right\|^2, \tag{6.2}$$

i.e. the inverse model $\boldsymbol{\pi}$ is optimized such that it is a right inverse to the current learned forward model $\boldsymbol{\phi}$. In the objectives (6.1) and (6.2) the integral over the estimated set $\mathcal{X}_c$ is still included, where $\mathcal{X}_c$ is estimated based on the error estimate as developed in section 5.3. The exploration strategy as the lower bound on the real fill distance with trust region (refer to section 5.4) still uses the error estimate. All parameters are the same as stated in section 6.1.

Figure 6.35 shows the mean reaching error and its standard deviation for the same test points as used for the point reaching evaluation experiment of section 6.4, see figure 6.12, over the number of sampled data points. Furthermore, two different learning rates α for optimizing the inverse model are considered. As one can see, there is nearly no learning progress at all and the error is very high. With an error of 72 ± 60 mm for $\alpha = 0.0005$ after 1200 sampled data points, the performance is about 47 times worse than if the inverse is learned based on minimizing the upper bound on the real performance. If the learning rate is twice as high ($\alpha = 0.001$), then the reaching error of 245 ± 145 mm is even about 163 times worse after 1200 data points. One can also see that more data points do not necessarily imply a monotonic decrease in the reaching error, indeed, for $\alpha = 0.001$, the error after 1200 data points was even higher than after only 100 data points. The main reason for this is that there are large parts of the reachable set which remain unexplored, since, when neglecting the upper bound, the inverse model does not have good extrapolation and hence exploration capabilities. But also in explored parts of the workspace, the error is considerably higher than if the upper bound is included. Therefore, including the error estimate in the objective is crucially important.

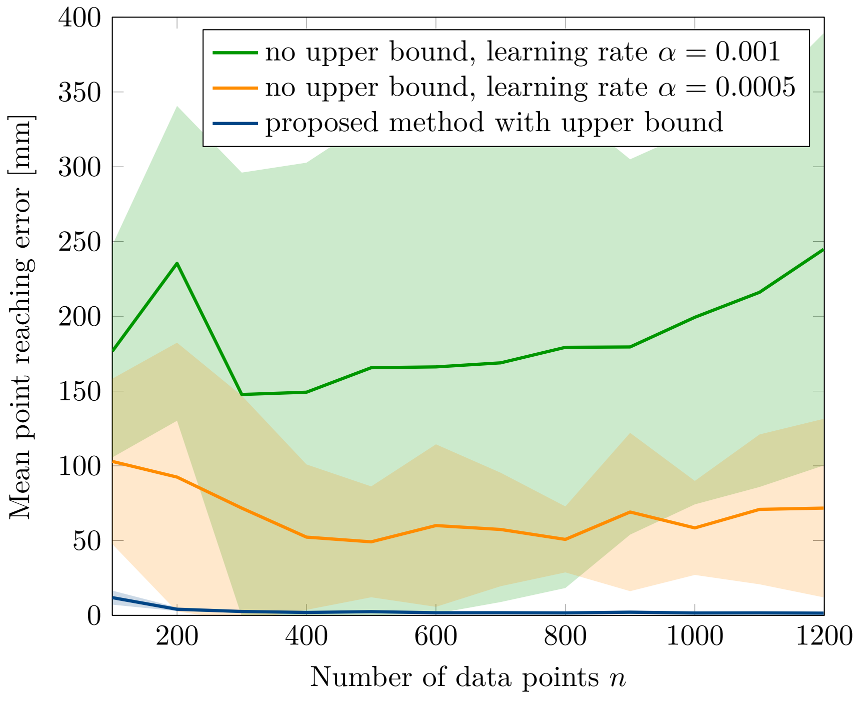


Figure 6.35: Neglecting the upper bound in the inverse learning objective. Therefore, the inverse model is only trained to be a right inverse of the current learned forward model without taking the imperfection of the forward model into account. Point reaching evaluation error over number of data points for two different learning rates of $\boldsymbol{\pi}$.

6.8.2. Integral

Next the influence of the value of c for estimating the reachable set for the integral over $\mathcal{X}_c$ in the inverse model learning objective (5.38) is investigated. One could argue that the objective (5.38) would also be reasonable when the integral is replaced by the sum over the observed data points only. Therefore, the case where there is no integral at all is additionally considered. In this case, the objective would be

$$\min_{\mathbf{w}\in\mathcal{W}} \sum_{\mathbf{x}\in\mathcal{D}_\mathcal{X}} \left(\|\boldsymbol{\phi}(\boldsymbol{\pi}(\mathbf{x};\mathbf{w})) - \mathbf{x}\|^2 + \sum_{i=1}^{d} \beta_i^2 s_i(\boldsymbol{\pi}(\mathbf{x};\mathbf{w}))^2 + \eta \, \|\boldsymbol{\pi}(\mathbf{x};\mathbf{w})\|^2 \right). \tag{6.3}$$

Figure 6.36 visualizes the evolution of the point reaching error over the number of collected data points. As one can see, if c is decreased from 0.02 to 0.01, the error is higher in the beginning. The reason for this is that if c is larger, the model extrapolates better. After about 500 collected data points, there is no statistically relevant difference, since then mainly the exploration focuses on the inside of $\mathcal{X}_c$. However, if c is made too large (in this case 0.05), the reaching error increases. This is caused by the fact that for large c, $\mathcal{X}_c$ contains points that are too far away from $\mathcal{X}$ and hence irritate the inverse learning, since the objective (5.38) would still try to find an inverse where no inverse exists.

In case the integral is neglected completely, i.e. the inverse is trained according to 6.3, the performance significantly drops and the final point reaching error after 1200 collected data points is about 3 times higher than if the integral is included (and $c = 0.02$).

To summarize, the integral is of great importance for high accuracy, but also requires some care for the choice of c. The influence of the presence of the integral is also further investigated in section 6.9 and 6.10 for different exploration strategies.

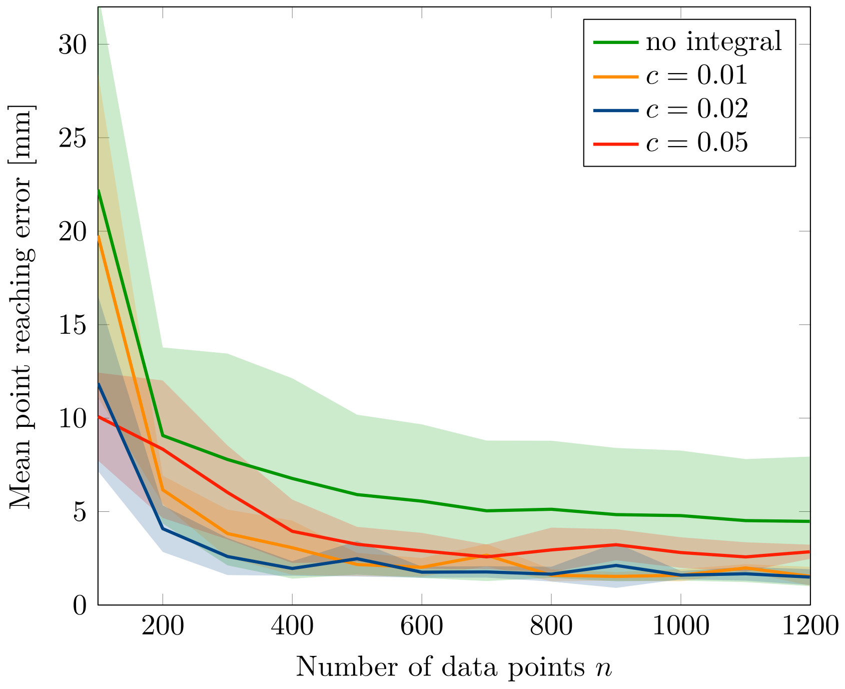


Figure 6.36: Influence of the presence of the integral as well as different error certainty values c to estimate the reachable set $\mathcal{X}_c$ for the integral in the inverse model learning optimization objective (5.38).

6.9. Exploration Strategy Comparison

In figure 6.37 different exploration strategies, with and without the integral in the learning objective, are compared to the proposed exploration strategy (in the variant with trust region 5.83) in terms of the point reaching performance over the number of collected data points. Random sampling in $\boxed{\mathcal{X}}$ (orange and green line) means uniform sampling in the bounding box of $\mathcal{X}$ (definition 5.37). Random in $\mathcal{X}_c$ denotes uniform sampling in the current estimated reachable set. Table 6.1 summarizes the point reaching errors after 1200 collected data points.

As one can see, the simple random sampling method in $\boxed{\mathcal{X}}$ has an about 4.1 times higher error than the proposed method. This is due to the fact that many targets are chosen outside of the true reachable set, which means that those points do not contribute to increasing the performance of the inverse model where it is defined, namely inside the reachable set.

With a final reached accuracy of 1.86 ± 0.54 mm, random sampling inside the current estimated reachable set $\mathcal{X}_c$ is competitive at the end of the exploration to the proposed strategy, which has an error of 1.50 ± 0.42 mm. One has to note that through the estimated reachable set a similar effect as the lower bound on the real fill distance is achieved, since both are derived based on the error estimate, which explains why random sampling in $\mathcal{X}_c$ also works well. The proposed exploration strategy is, however, in the beginning significantly faster than random sampling in $\mathcal{X}_c$.

For both random sampling methods, the performance significantly drops if no integral is considered (refer also to the discussion of section 6.8.2). This especially holds true for random sampling in $\boxed{\mathcal{X}}$, which has an 7.1 times higher error than the proposed method, since the integral helps for compensating areas inside $\mathcal{X}_c$ where no or little data has been observed.

In figure 6.38 the two variants of the proposed active exploration strategy are compared with each other. The one variant (5.83), which is used in all experiments so far and in all following experiments, is based on maximizing the lower bound on the real fill distance with a trust region around the already sampled data points. The other variant (5.82) maximizes the lower bound on the real fill distance inside the current estimated reachable set $\mathcal{X}_c$. As one can see, in the beginning, the first variant with the trust region is slightly better. This can be explained by the fact that with the chosen parameters of $d_{\mathcal{X}_T} = 0.1$ and $c = 0.02$, the trust region variant is allowed to sample outside of the current $\mathcal{X}_c$ and is therefore more explorative. However, the difference between those variants is not significant.

Table 6.1: Point reaching error after 1200 data points for different exploration strategies.

Exploration Strategy	**Mean reaching error** [mm]
proposed exploration strategy with trust region	$\mathbf{1.50 \pm 0.42}$
proposed exploration strategy in $\mathcal{X}_c$	$\mathbf{1.66 \pm 0.47}$
random in $\boxed{\mathcal{X}}$	6.19 ± 2.63
random in $\boxed{\mathcal{X}}$ without integral	10.72 ± 4.97
random in $\mathcal{X}_c$	1.86 ± 0.54
random in $\mathcal{X}_c$ without integral	6.90 ± 4.87

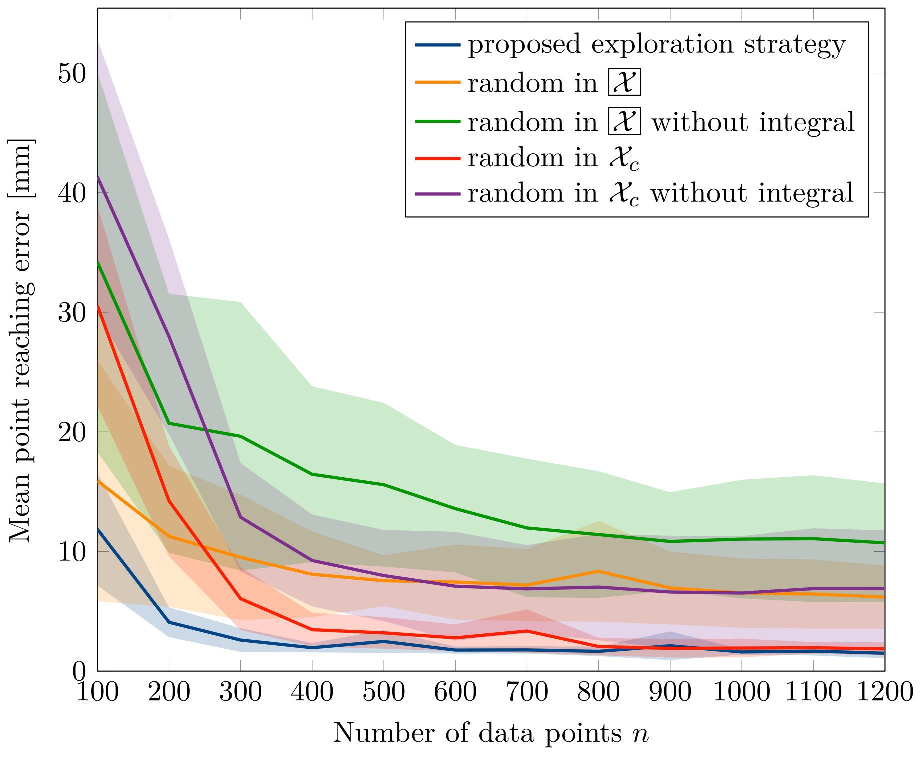


Figure 6.37: Exploration strategy comparison in terms of point reaching accuracy.

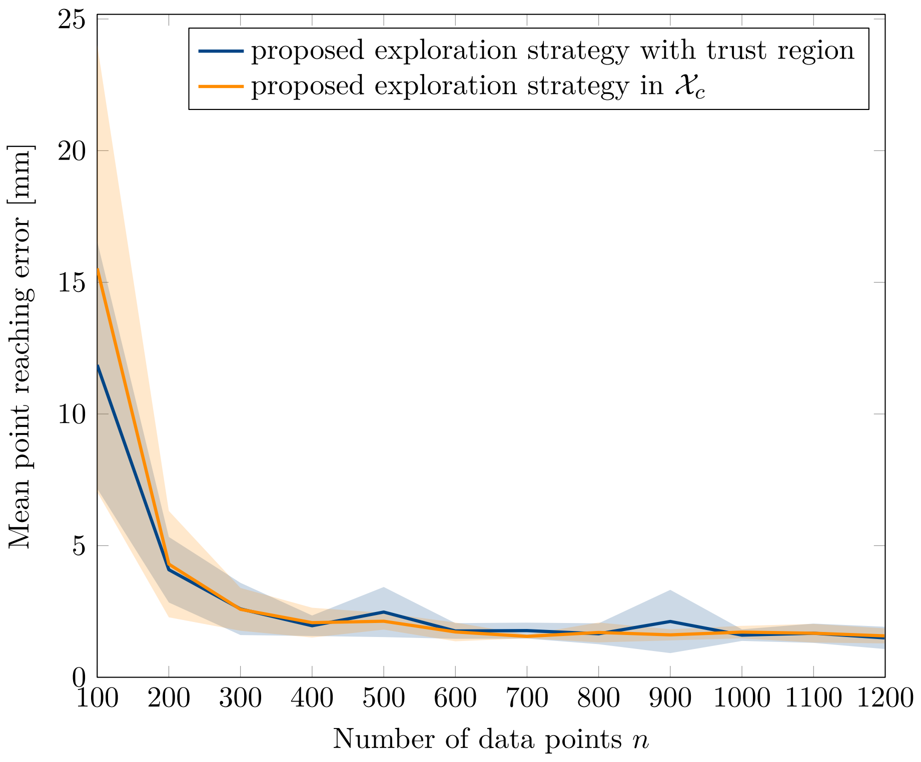


Figure 6.38: Exploration strategy comparison between (5.83) and (5.82) in terms of point reaching accuracy.

6.10. Exploration with Known True Reachable Set $\mathcal{X}$

In this experiment, it is investigated what happens if the true reachable set $\mathcal{X}$ is known for an exploration strategy. In one case, the lower bound on the real fill distance is maximized over $\mathcal{X}$, i.e. the exploration targets are chosen via

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}} \left(\min_{\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}} \|\mathbf{x} - \mathbf{x}_i\| - \varepsilon_\beta(\mathbf{x}) \right). \tag{6.4}$$

In the other case, the exploration strategy samples targets uniformly in the true reachable set $\mathcal{X}$. In both cases, the integral in the inverse model learning objective (5.38) is still computed over the current estimated reachable set $\mathcal{X}_c$.

Figure 6.39 shows the evolution of the point reaching error over the number of collected data points. As one can see, if the true reachable set is known, both the random and fill distance (6.4) based exploration strategies are able to achieve a better performance at the end of the exploration than if $\mathcal{X}$ is unknown. In the beginning, the random strategy is faster, while in the end the strategy based on the lower bound of the real fill distance leads to the lowest error overall. With a final reached error of 0.78 ± 0.07 mm, the fill distance based exploration strategy (6.4) has an extremely low error. However, the proposed method that does not rely on knowing the true $\mathcal{X}$ is very competitive both in terms of exploration speed and final reached error, indicating that the derived bounds are useful in practice for compensating the lack of knowing the true reachable set $\mathcal{X}$ a priori.

In figure 6.40 the evolution of the reaching error for the random and fill distance based strategy with known true $\mathcal{X}$ but without the integral in the inverse model learning objective (5.38) is visualized. If the integral is neglected, both the final reached error and the exploration speed in the beginning is worse. In this case, the final reached error is even higher than with the proposed method, which means that the integral is also important even if $\mathcal{X}$ is known for the exploration. However, compared to the results of section 6.8.2 and 6.9, the error increase if the integral is neglected for a known $\mathcal{X}$ for the exploration strategy is less pronounced than if the true reachable set is not available for the exploration. Interestingly, similar to the findings in section 6.9, the integral term especially seems to be important for random based strategies also in this case where $\mathcal{X}$ is known.

Table 6.2 summarizes the point reaching errors at the end of the exploration after 1200 collected data points.

Table 6.2: Point reaching error after 1200 data points.

Exploration Strategy	**Mean reaching error** [mm]
proposed exploration strategy	$\mathbf{1.50 \pm 0.42}$
lower bound on real fill distance in true $\mathcal{X}$	0.78 ± 0.07
lower bound on real fill distance in true $\mathcal{X}$ without integral	1.70 ± 1.14
random in true $\mathcal{X}$	1.28 ± 0.32
random in true $\mathcal{X}$ without integral	2.35 ± 0.96

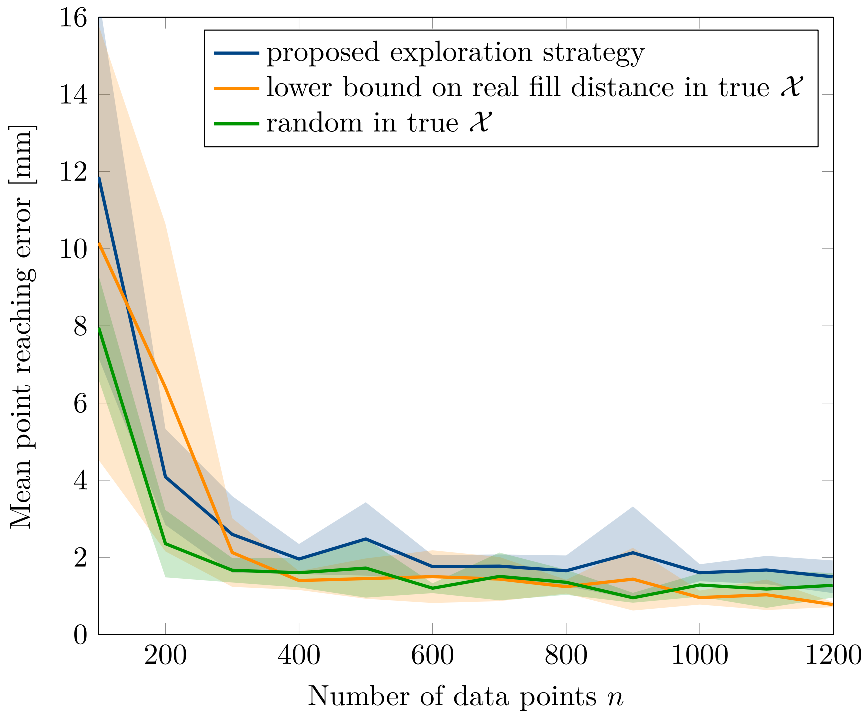


Figure 6.39: Comparison of the evolution of the mean reaching error over the number of collected data points in case the true reachable set $\mathcal{X}$ is known for the exploration strategy.

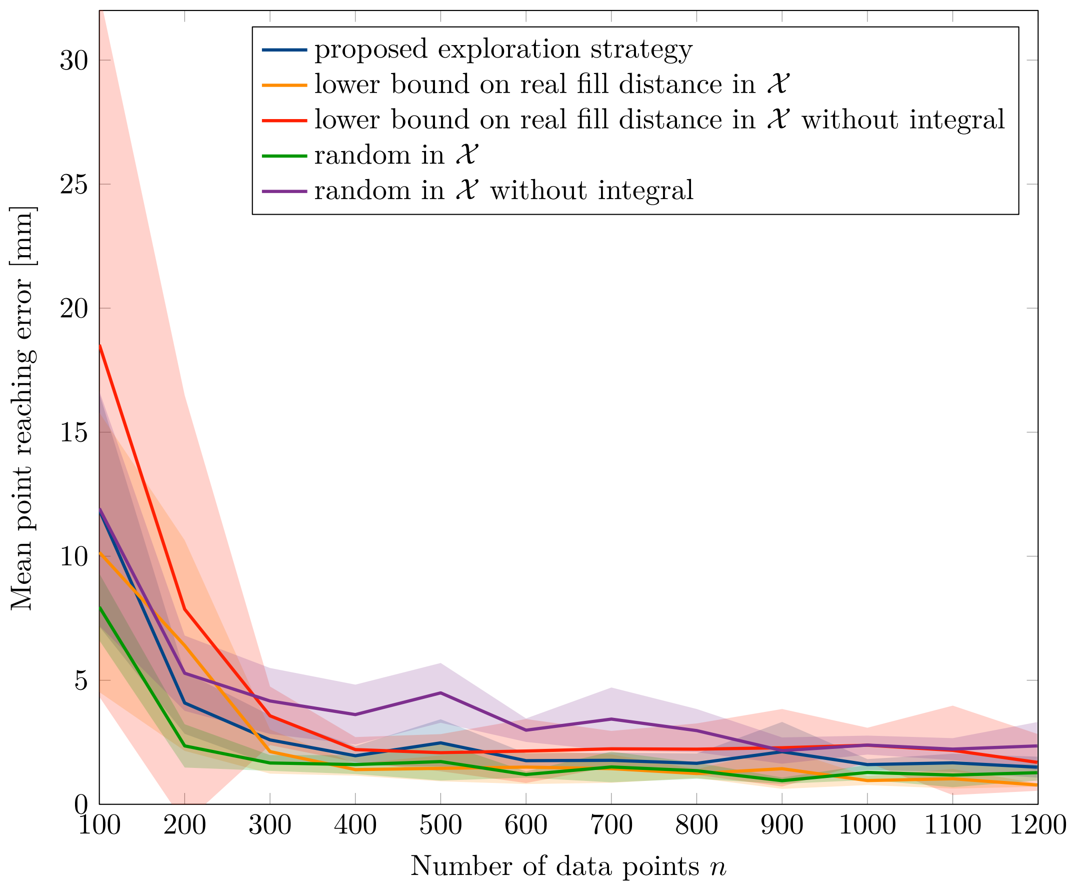


Figure 6.40: Comparison of the evolution of the mean reaching error over the number of collected data points in case the true reachable set $\mathcal{X}$ is known for the exploration strategy. Same plot as in figure 6.39, but with two additional situations where no integral in the inverse learning objective is included.

6.11. Hybrid Control

In this experiment, the behavior of the system if the learned lambda controller developed in section 5.7 is utilized together with the learned inverse model $\boldsymbol{\pi}$ to generate the muscle stimulations, which is known as hybrid control as described in section 3.6. For a desired target $\mathbf{x} \in \mathcal{X}_c$ the muscle stimulations are calculated by

$$\mathbf{u} = \boldsymbol{\pi}(\mathbf{x}) + \mathbf{K}_p \left(\mathbf{L}_{\text{opt}}^{\text{CE}}\right)^{-1} \left(\mathbf{l}^{\text{CE}} - \boldsymbol{\Omega}(\mathbf{x})\right) - \mathbf{K}_d \dot{\mathbf{l}}^{\text{CE}}, \tag{6.5}$$

consisting of the static open-loop part $\boldsymbol{\pi}(\mathbf{x})$ and the monosynaptic feedback on the current measured length of the contractile element $\mathbf{l}^{\text{CE}}$ with predicted target $\boldsymbol{\Omega}(\mathbf{x}) = \boldsymbol{\Lambda}(\boldsymbol{\pi}(\mathbf{x}))$ by the learned $\boldsymbol{\Lambda}$.

As discussed in section 3.6, the predicted lengths $\boldsymbol{\Omega}(\mathbf{x})$ of the contractile element must correspond to the equilibrium lengths determined by the static muscle stimulation $\boldsymbol{\pi}(\mathbf{x})$. Since $\boldsymbol{\Lambda}$ is learned, one cannot expect that this is perfectly fulfilled. In figure 6.41 the reached points with the hybrid controller after 1200 collected data points (the inverse $\boldsymbol{\pi}$ is the same of section 6.4) for the the same targets as in figure 6.12 are shown. As one can see, the difference between the targets and the actual reached points is hardly visible. Quantitatively, the mean error of 1.90 ± 1.11 mm is slightly higher than without the lambda control part, where the error was 1.63 ± 1.0 mm (see figure 6.13). In this experiment, the gains of the monosynaptic feedback are $\mathbf{K}_p = 0.1 \cdot \mathbf{I} \in \mathbb{R}^{6\times6}$ and $\mathbf{K}_d = 0.05 \cdot \mathbf{I} \in \mathbb{R}^{6\times6}$.

To study the influence of the additional feedback on the system, figure 6.42 shows a point-to-point reaching trajectory for different gains $\mathbf{K}_p = 0.1 \cdot \mathbf{I}$, $\mathbf{K}_p = 0.2 \cdot \mathbf{I}$, $\mathbf{K}_p = 0.3 \cdot \mathbf{I}$ and the case without lambda control. As one can see in figure 6.42c, if the monosynaptic reflex is included, the target position is reached more quickly than with $\boldsymbol{\pi}$ alone. However, for $\mathbf{K}_p = 0.2 \cdot \mathbf{I}$ a slight overshoot and for $\mathbf{K}_p = 0.3 \cdot \mathbf{I}$ a greater overshoot is observed. Furthermore, with the monosynaptic reflex the velocity profiles (see figure 6.42e) have a shape that is more similar to ones observed in human experiments [3].

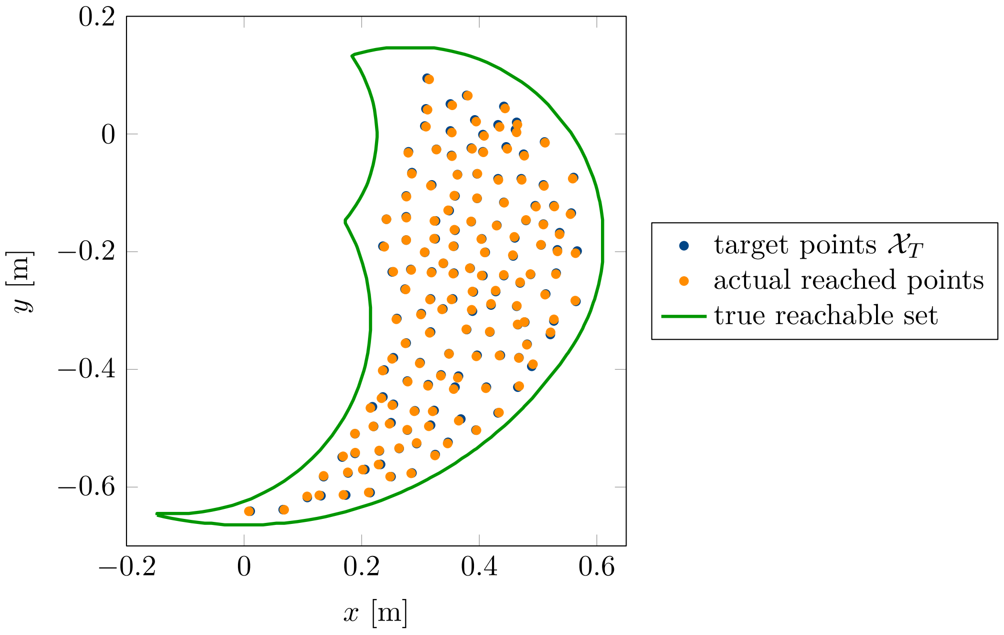
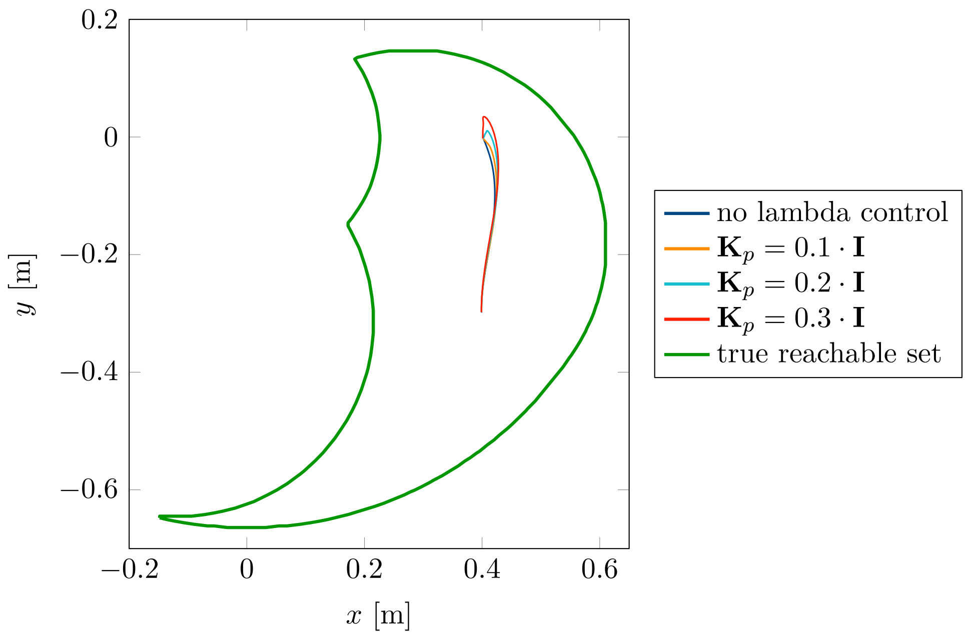
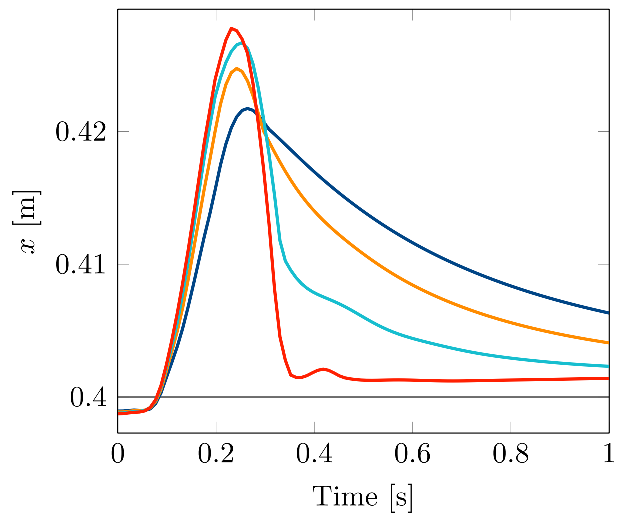


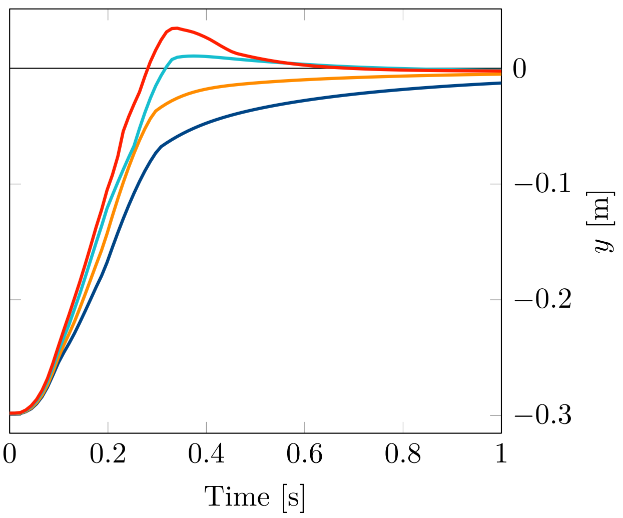
Figure 6.41: Point reaching evaluation with hybrid control. Mean reaching error 1.90 ± 1.11 mm.



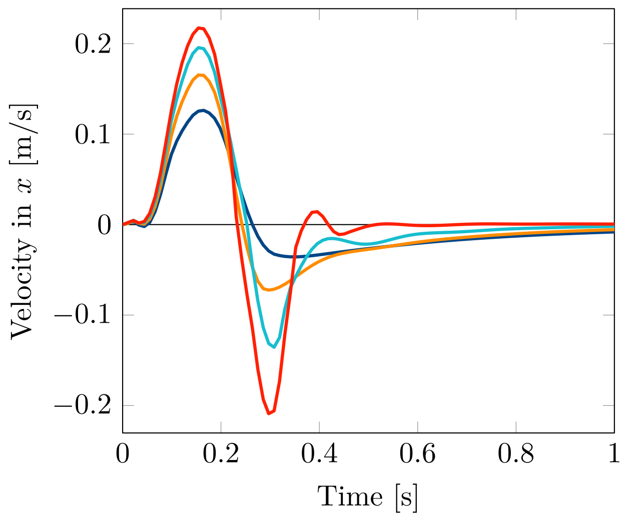
(a) Point-to-point trajectory



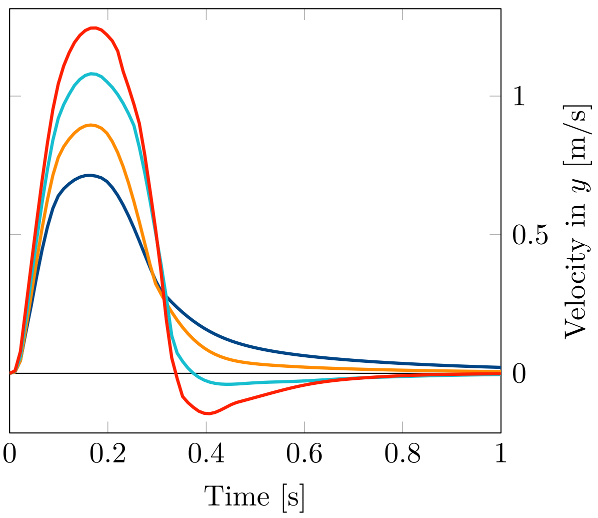
(b) Hand position in x-direction



(c) Hand position in y-direction



(d) Hand velocity in x-direction



(e) Hand velocity in y-direction

Figure 6.42: Hybrid control trajectories for point-to-point reaching experiment.

6.12. Real Pneumatic Muscle-Driven Robot

The goal of this experiment is to show that the proposed active inverse model learning framework also works with a real system, the muscle-driven robot, which mimics the simulated human arm. For details about the robot refer to section 3.8.

In comparison to the simulated experiments, the $\mathcal{X}$-space in this experiment is not the position of the hand of the arm, but its joint configuration. The reason for this is mainly since the forward kinematics of the real robot significantly differs from the CAD construction specification, implying that a calibration would be required first, which is not the scope of this work. Furthermore, the experiment has been tried once in the end-effector space with a roughly calibrated forward kinematics. Here it turned out that the performance was inferior compared to the joint configuration space as presented in this section. Since this was tried only once, it is expected that with tuning the parameters and calibrating the forward kinematics, good performance can also be achieved in the end-effector space.

Speaking of parameters, most of the hyperparameters (even the kernel parameters) of the real robot experiments are the same as described in section 6 with the following exceptions: Since the real robot only has five muscles, $m = 5$. The initial muscle stimulation is $\mathbf{u}_0 = \frac{1}{5}(1, 2, 1, 2, 0.5)^T \in \mathbb{R}^5$, corresponding to a pressure of $(1, 2, 1, 2, 0.5)$ bar. The upper bound on RKHS norm of the true forward model, i.e. the robot, for the inverse learning objective and the reachable set estimate is $\beta_i = \sqrt{3}$. With this value, however, the exploration turned out to be a bit too conservative. Therefore, for the exploration strategy this bound on the RKHS norm is set to a smaller value of $\beta_i = 1$.

Compared to the simulated human arm, a much wider range of the muscle stimulation space has to be utilized in order to move the arm through its complete possible joint space. To account for this fact, it turned out to be favorable to increase the number of data points that are sampled with noise to 50 and to also increase the amount of noise by a factor of 10.

All data points are collected by applying the muscle stimulation from the exploration strategy from the state of the robot where it has zero muscle stimulation. In total, collecting one data point takes 10 seconds.

6.12.1. Exploration Behavior

Figure 6.43 shows the exploration result after a total of 651 collected data points $\mathcal{D}_\mathcal{X}$ (blue) in the $\mathcal{X}$ space, i.e. the reached joint angle configurations of the real arm robot, as well as the estimated reachable set $\mathcal{X}_c$ (orange) for $c = 0.02$. This value for the error certainty means that for each desired joint configuration $\mathbf{x} \in \mathcal{X}_c$, the real robot should reach these angles within a accuracy of 0.02 rad or 1.15°, when using the learned inverse model. Refer to the next section for an evaluation of the actual accuracy. Looking at the shape of the estimated reachable set and the collected data points, one can see that for this muscle-driven system, the reachable set is not just a rectangle, as one may expect from an $\mathcal{X}$ space as a joint space. This is partly caused by the one biarticular muscle. Due to the limitations of the pneumatic muscles as discussed in section 3.8, the reachable set is smaller than of the one of the simulated human arm.

In figure 6.44, the exploration process for a growing number n of collected data points in therms of both the distribution of the collected data points as well as the evolution of the learned reachable set is visualized. After 60 collected data points (figure 6.44b), the estimated reachable set is still nearly empty. When 120 data points have been collected (figure 6.44c), the reachable set is mainly spread along the shoulder angle. Generally, it seems that the exploration in the shoulder angle is much faster than in the elbow angle. One reason to explain this is that the elbow pneumatic muscles have a significantly lower actuation range than the shoulder muscles and therefore also higher pressures are required to move the elbow the same amount as the shoulder. Despite this difficulty, the methodology is able to explore in the elbow angle direction successfully. Similar to the simulated experiments (see section 6.2), the exploration with the real robot also shows a uniform distribution of the collected data points with a comparably good separation distance.

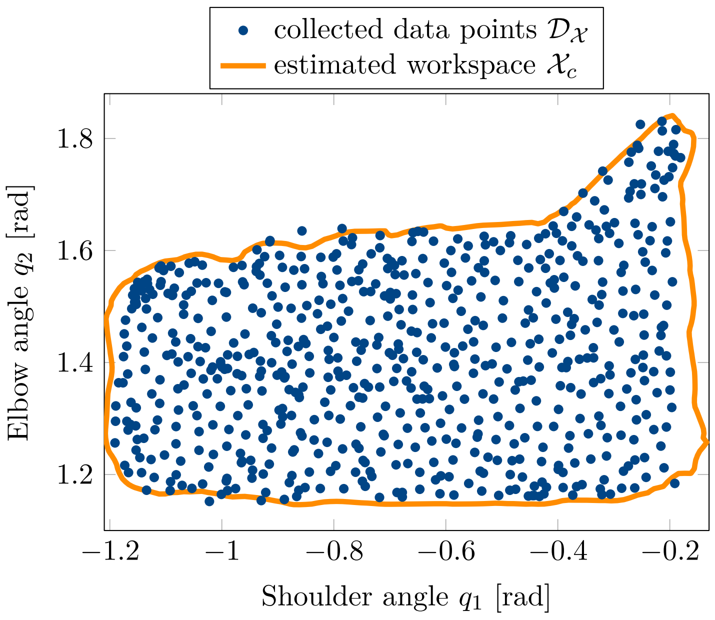


Figure 6.43: Exploration result after a total of 651 collected data points with the real muscle driven robot. Blue points denote the reached joint angle configurations of the robot. The orange line visualizes the boundary of the estimated reachable set $\mathcal{X}_c$ for $c = 0.02$.

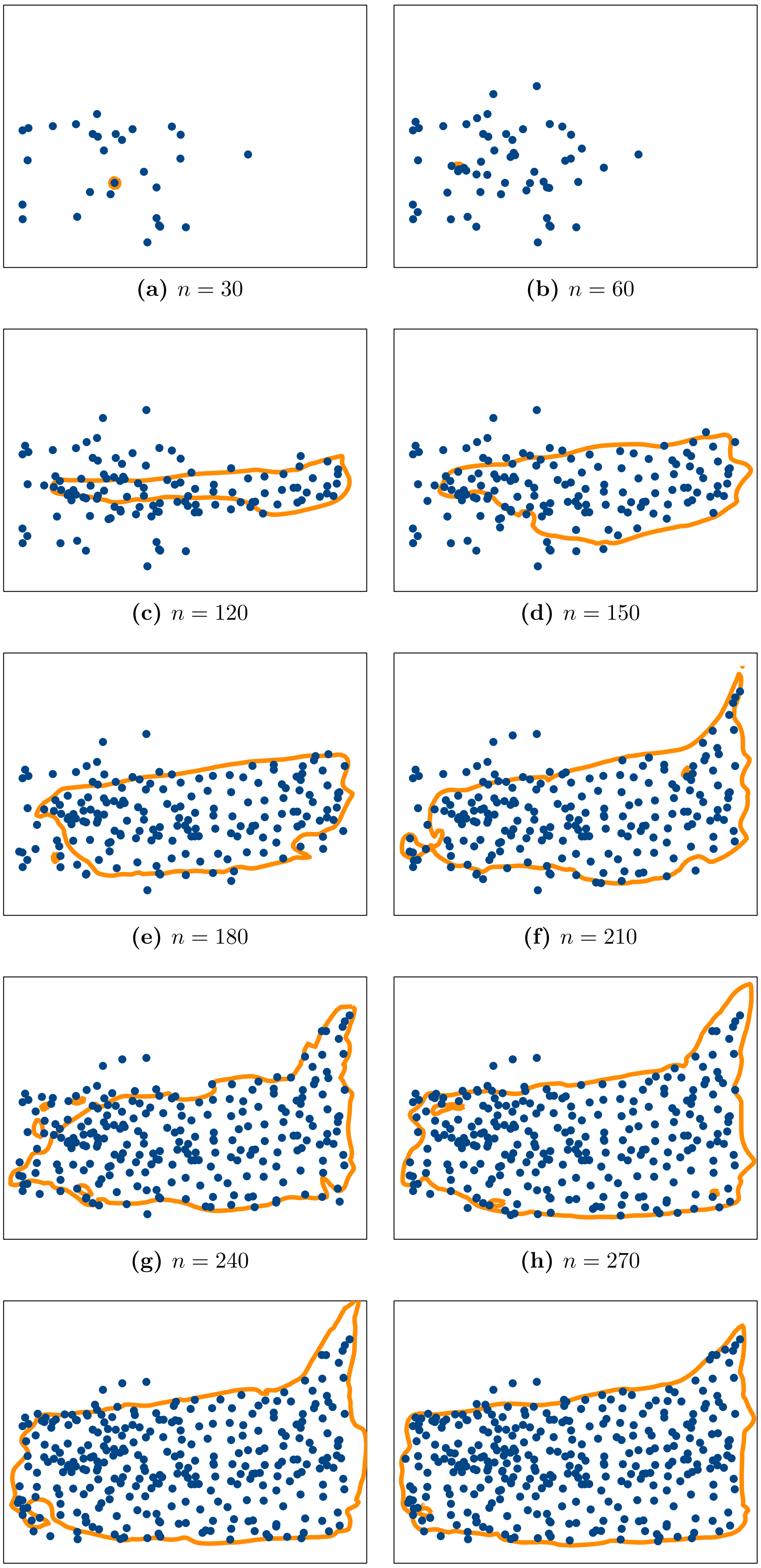
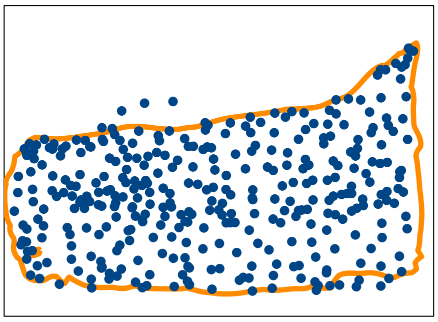
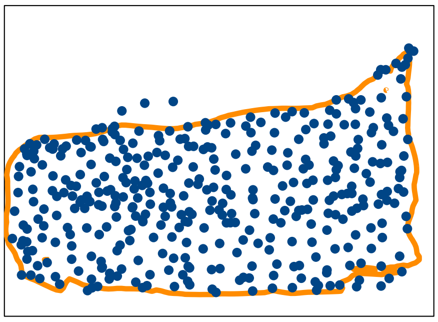
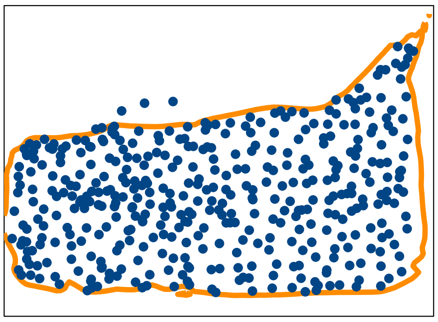
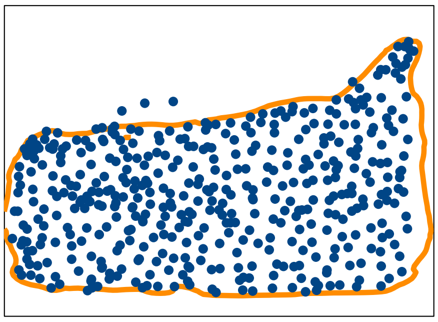

(a) $n = 30$ (b) $n = 60$ (c) $n = 120$ (d) $n = 150$ (e) $n = 180$ (f) $n = 210$ (g) $n = 240$ (h) $n = 270$ (i) $n = 300$ (j) $n = 330$

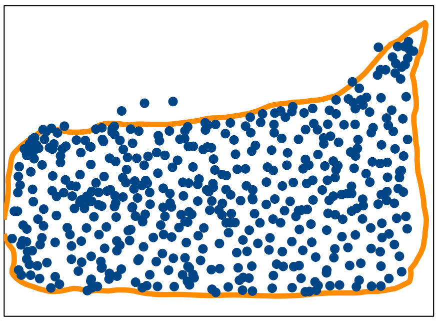
Figure 6.44: Evolution of the collected data points and estimated reachable set. Same legend and axis limits as in figure 6.43.

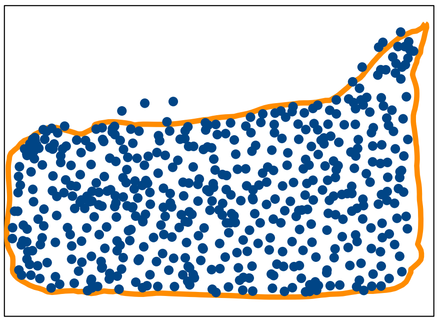

(k) $n = 360$

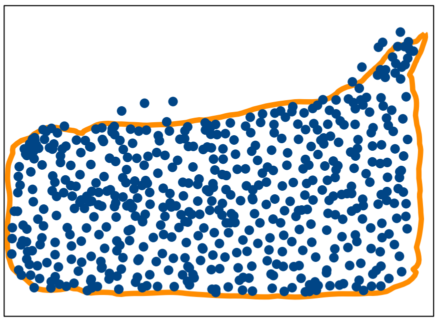

(l) $n = 390$

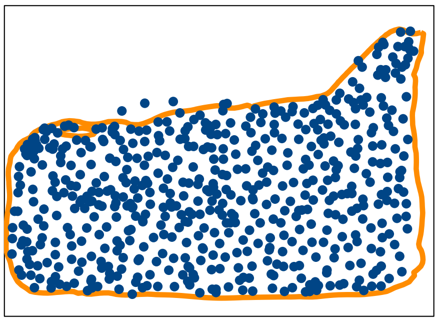

(m) $n = 420$

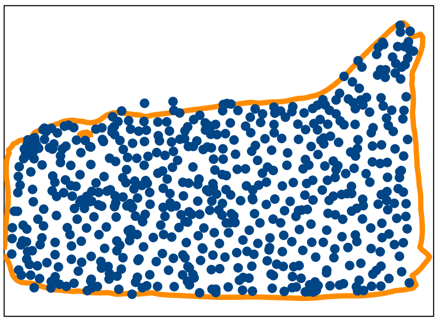

(n) $n = 450$


(o) $n = 480$


(p) $n = 510$


(q) $n = 540$


(r) $n = 570$


(s) $n = 600$

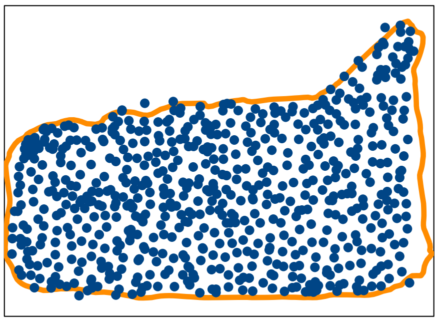

(t) $n = 651$

Figure 6.44: (continued) Evolution of the collected data points and estimated reachable set. Same legend and axis limits as in figure 6.43.

6.12.2. Point Reaching Evaluation

Figure 6.45 shows the 100 target points, i.e. the target angles of the shoulder and elbow joint, to test the accuracy of the learned inverse model. This point reaching evaluation

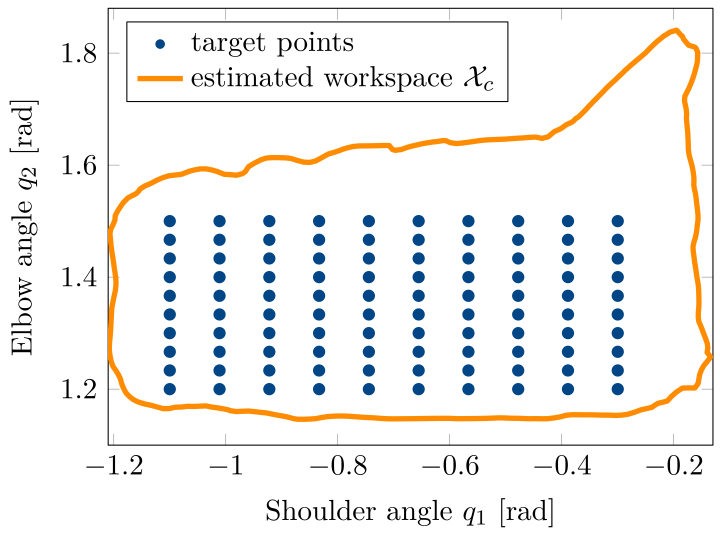
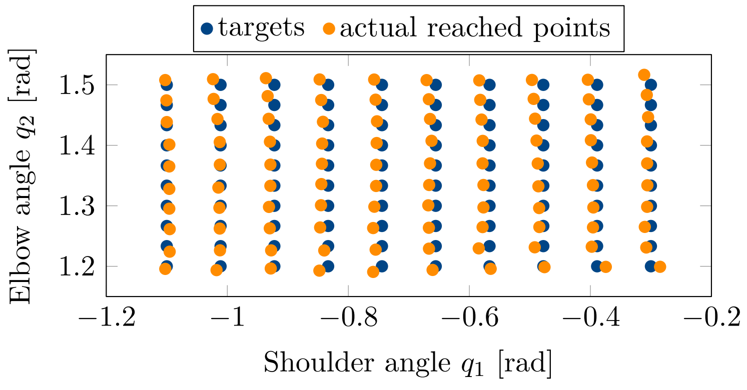


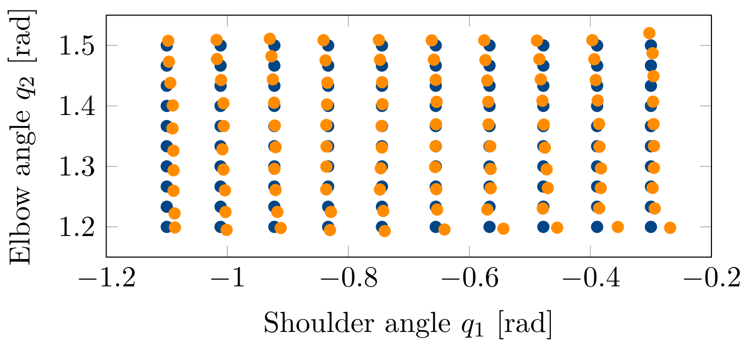
Figure 6.45: Target joint angles for evaluating the point reaching accuracy, 100 in total.

experiment is performed two times. In experiment 1, the active inverse model learning framework has collected 651 data points. The learned $\boldsymbol{\pi}$ is then used twice to reach the target points of figure 6.45. The reached points for both evaluations are shown in figure 6.46. For experiment 2, the learning framework is started from scratch with the same parameters. This time, only 435 data points are collected and the evaluation of the 100 target points is performed only once. The point reaching results are visualized in figure 6.47. Table 6.3 summarizes the accomplished accuracies for both experiments with the three evaluations in total. The error in the elbow joint is very consistent between the three different evaluation runs, in the shoulder joint, it shows more variation. Taking into account that this is a real system with friction, which is controlled open-loop, the accuracy between 0.3° and 0.5° in the shoulder and about 0.3° in the elbow is better than the expectation of the author of this work.

In contrast to the simulation where the true forward model, i.e. the system itself, is independent of its current state, the pneumatic muscles of the real robot as well as other friction effects imply that the true forward model of the real robot is *not* state independent [13]. Therefore, the evaluation accuracy reported in figure 6.46, 6.47 and table 6.3 is only valid when starting the motion from the resting state of the robot corresponding to zero muscle stimulations. To investigate the impact of these effects, figure 6.48 shows an experiment where 6 joint configurations are reached both starting from the resting position as in training and starting from the last reached configuration. As one can see, there is a difference in the accuracy, but except for the joint configuration in the lower right corner of figure 6.48, the difference is small. Table 6.4 quantifies the concrete errors of this experiment. The mean error in the shoulder joint increases by 34 %, in the elbow by 84 %, but are still low. In figure 6.49, the goal is to only move the elbow joint back and forth while keeping the shoulder angle constant. Table 6.5 reports the errors of each configuration. As one can see, the configuration where the elbow is flexed has a low error, independent from the reaching order, while the error in the extended position depends on the configuration history. This can again be explained by the way the robot is constructed and the deficiencies of pneumatic muscles with respect to their contraction range.



(a) Evaluation run 1



(b) Evaluation run 2

Figure 6.46: Actual reached joint angles for point reaching evaluation with real robot. Experiment 1 with 651 collected data points. Subplots (a), (b) show two evaluation runs for the same learned $\boldsymbol{\pi}$.

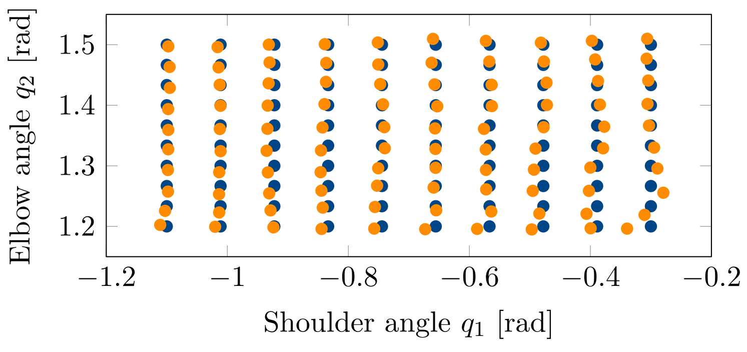


Figure 6.47: Actual reached joint angles for point reaching evaluation with real robot. Experiment 2 with 435 collected data points.

Table 6.3: Mean errors and standard deviations of point reaching evaluation experiment.

Experiment[a]	**Evaluation**[b]	**Mean shoulder angle error** [°]	**Mean elbow angle error** [°]
1	1	0.51 ± 0.24	0.31 ± 0.21
1	2	0.30 ± 0.33	0.32 ± 0.23
2		0.40 ± 0.33	0.29 ± 0.19

[a] Experiment means complete run of learning methodology
[b] Evaluation means same $\boldsymbol{\pi}$, but different evaluation run

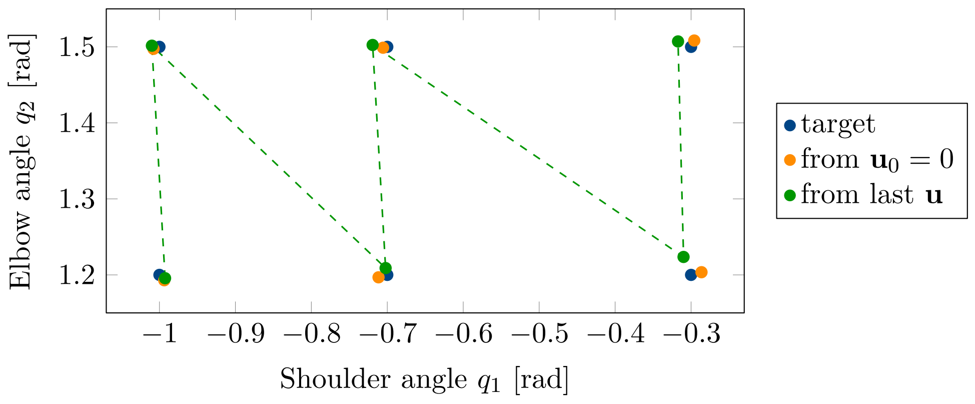


Figure 6.48: Comparison between point reaching accuracy if each configuration is approached from the starting configuration with zero muscle stimulation as in training (orange) versus starting from the last reached configuration (green), where the movement order starts from the lower left.

Table 6.4: Mean errors and standard deviations for experiment of figure 6.48.

Reaching mode	**Mean shoulder angle error** [°]	**Mean elbow angle error** [°]
from $\mathbf{u}_0 = 0$	0.47 ± 0.19	0.25 ± 0.14
from last $\mathbf{u}$	0.63 ± 0.33	0.46 ± 0.43

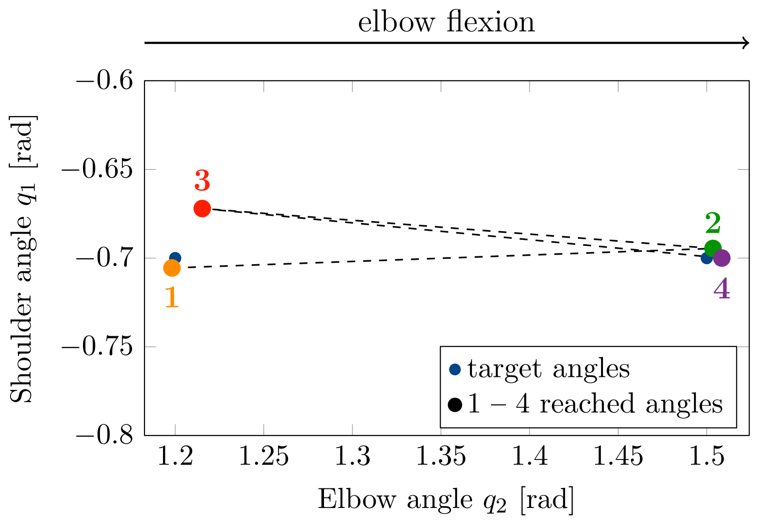


Figure 6.49: Accuracy if elbow joint should move back and forth while the shoulder angle should be kept constant. Numbers 1 – 4 indicate the movement order. Please note the switched axis.

Table 6.5: Mean errors and standard deviations for experiment of figure 6.49.

Configuration	**Shoulder angle error** [°]	**Elbow angle error** [°]
1	0.32	0.10
2	0.31	0.20
3	1.60	0.88
4	0.002	0.49

7. Discussion

7.1. Performance

In general, the resulting performance of the learned inverse model with a mean point reaching error of 1.50 ± 0.42 mm is more than sufficient, especially for a musculoskeletal system that is controlled open-loop. The proposed framework is able to learn an accurate inverse model data efficiently and to cover the true reachable set quickly.

7.2. Assumptions in Practice

The active inverse model learning framework is derived under the assumption that the true forward model comes from the same RKHS as used for the learned forward model. This allows to derive the error estimate which is central to define the inverse model learning optimization problem, the estimation of the reachable set and finally the exploration strategy. It is, however, completely unrealistic that the true forward model, i.e. the musculoskeletal system, is an element of the RKHS for a Gaussian kernel with finite RKHS norm. Even if this would be true, one still cannot expect that the correct hyperparameters would be chosen. Therefore, the question arises whether the assumptions and hence the derived bounds are useful in practice at all. As shown in the experiments, without extensive tuning of the hyperparameters, the derived bounds not only hold true (with minor exceptions), they are also tight enough to be useful in practice, for example to estimate the real point reaching error. Furthermore, if neglecting the upper bound in the exploration objective, the performance drops significantly, which is another indication for the usefulness of the bounds and the underlying theory for practical applications.

The main limitation of the present work is that the learned inverse models are state-independent, static mappings. While for the investigated systems, the attractor assumption holds in the complete workspace, this is not necessarily true for all musculoskeletal systems like a complete human body.

7.3. Scalability

This work considered problems where the input space is 6 dimensional and the workspace is 2 dimensional. When scaling to higher dimensional workspaces, three problems arise. First, as mentioned in section 5.5.2, the computational complexity of performing one gradient step on the complete data set with size n is $\mathcal{O}\left(n_h^d n^2 + n^3\right)$. Therefore, training the neural network becomes computationally demanding both for a growing dataset size and also in the dimensionality of the workspace, which causes the term n_h^d to grow exponentially. Indeed, it turned out that training the neural network through the kernel method and considering the integral formulation takes by far the most computational time. Secondly, increasing the workspace dimension also requires exponentially more data points to cover the complete workspace. Thirdly, the exploration optimization problem is solved via grid evaluation, which becomes prohibitive in higher dimensions as well.

However, scaling to higher dimensional input spaces does not seem to be a problem, although this has not been evaluated thoroughly in this work yet.

Considering the fact that in most experiments after about 200 data points an already accurate inverse model is learned that covers the complete reachable set is impressive when taking into account that 200 data points in a 6 dimensional input space is extremely sparse.

7.4. Forward Model

In the present work, the forward model is represented as a linear combination of Gaussian kernels. In distance to any observed data points, a Gaussian kernel approaches a zero value rapidly, hence the forward model approaches its constant prior mean value. This, however, is not realistic for a musculoskeletal system (and many other systems as well). Therefore, it is expected that a slightly more informed prior mean than a constant could increase the extrapolation capabilities of the forward model and hence also of the inverse model. With respect to the scalability as discussed in the last paragraph, better extrapolation capabilities are important if one would like to increase the workspace dimension.

The by far biggest problem of the forward model is, however, that a kernel method deteriorates if two data points are too close to each other in relation to the kernel width (for many radial kernels). Indeed, the condition number of the kernel matrix mainly depends on the separation distance of the data [45]. This can be counteracted to some extend by using a higher regularization parameter, which, if it is too high, also deteriorates the quality of the forward model, which is then less useful for learning the inverse.

In this regard, one can think of using other regression methods for learning the forward model. However, a crucial aspect to derive the whole methodology is the ability to estimate the error between a learned and the true function for the forward model. Thus, an alternative regression method would have to provide similar estimates, not at least to be able to state theoretical guarantees of the method.

7.5. Estimation of the Reachable Set

Estimating the reachable set turns out to be an essential aspect of learning an inverse model, not only from a theoretical side in order to be able to even properly define the inverse model and to evaluate its the quality, but also from a practical perspective. An estimation of the reachable set enables to define what it means to learn an inverse model in terms of an integral formulation. This integral term is important to not only achieve the low point reaching error of the learned inverse model, but also for the extrapolation and hence exploration capabilities of the model.

The way the reachable set is estimated turns out to be close to the true one and especially, its representation through the error estimate is fast to compute, since one can check efficiently whether a point belongs to the estimated reachable set or not.

In contrast, simply setting $\mathcal{X} \approx \phi(\mathcal{X})$ without taking the imperfection of the learned forward model into account leads to completely unreasonable results and is also computationally demanding.

From a theoretical point of view, under the stated assumptions, it is guaranteed that the estimated reachable set $\mathcal{X}_c$ does not overestimate the true reachable set $\hat{\mathcal{X}}_c$ with error certainty c. Two things have to be taken into account here. First, the true $\hat{\mathcal{X}}_c$ overestimates the true reachable set $\mathcal{X}$ exactly by the value of c. Therefore, since $\mathcal{X}_c$ tries to approximate $\hat{\mathcal{X}}_c$ and it is only guaranteed that $\mathcal{X}_c \subset \hat{\mathcal{X}}_c$, $\mathcal{X}_c$ is allowed to potentially also overestimate $\mathcal{X}$ by c. If the value of c is now too large, the inverse learning optimization problem through the integral term would try to find an inverse for unreachable parts of the true workspace, i.e. where no inverse exists. Therefore, the parameter c should be chosen carefully, since making it too small also leads to little advantage from the integral term. Actually, choosing this parameter c turns out to be not too difficult for the situations considered in this work, not at least since it has a clear interpretation.

7.6. Exploration Strategy

In order to generate useful data for the forward model efficiently in the high dimensional control input space, the current learned inverse model is bootstrapped by choosing exploration targets in the low dimensional target space.

Overall, the proposed active exploration strategy that is based on maximizing a lower bound on the real fill distance of the system leads to the lowest point reaching error and the fastest error decrease compared to different random strategies. The random strategy inside the current estimated reachable set is competitive at the end in terms of the point reaching error, but is slower in the beginning. Through constraining the exploration inside the current estimated reachable set, a similar effect as with the lower bound on the real fill distance is achieved. For all exploration strategies, the integral formulation is important, since it increases the extrapolation quality of the inverse model and hence the inverse model is better suited for exploration. In particular, the random based strategies rely on the presence of the integral term. The proposed exploration strategy is even competitive to a strategy that knows the true reachable set.

An interesting aspect of the proposed exploration strategy is that it inherently trades-off exploration and exploitation. In the beginning, most exploration target points are chosen outside of the current estimated reachable set for fast expansion of the estimated reachable set. At a later stage of the procedure, the majority of the targets are selected inside the estimated reachable set, which means that the exploration strategy recognizes that the reachable set has been explored and then focuses on improving the quality of the inverse model inside the reachable set.

7.7. Future Work

From an application point of view, in addition to the static predicted muscle stimulation from the inverse model, the developed method to learn the monosynaptic reflex could be applied to the real robot as well. It is assumed that by integrating feedback on the lengths of the pneumatic muscles, the impact of the hysteresis and friction of the real robot could be reduced. Furthermore, another interesting direction with respect to applications would be to increase the dimensionality both of the control input space and the workspace. As discussed above, it is expected that the method has no difficulties with an increased control input space size.

The active inverse model learning framework developed in this work has been evaluated on musculoskeletal systems only. Since it is a general method, future work could include applications to other inverse learning scenarios. Since the proposed methodology provides a guaranteed (under the mentioned assumptions) bound on the real performance error of the inverse model, one could also embed the framework into approaches that learn multiple paired forward-inverse models.

One limitation of the present work with respect to application for musculoskeletal systems is that the learned inverse is a static mapping and therefore is not able to directly influence the dynamics between the equilibrium states. Although with the monosynaptic reflex, as shown, this dynamics can already be influenced for example to reach the desired target more quickly, further extensions to this end are possible. For example, one can think of using the learned inverse within a trajectory planning framework. Furthermore, as mentioned before, the assumption that every point in the reachable set is an attractor point is also not fulfilled for every musculoskeletal system. To address this issue, one way would be to learn a state dependent inverse model.

8. Conclusion

In the present work, a novel methodology to learn an inverse model for redundant systems was developed. By formalizing what it actually means to learn an inverse model, a method was derived where the inverse model, represented as a neural network, is learned by minimizing an upper bound on the real performance error, which is provided through a learned forward model. Including this upper bound in the learning objective turned out to be very important for the performance. An essential aspect of the whole framework was to estimate the reachable set of the system. This allows to properly formulate the inverse learning objective in integral formulation, which showed to further increase the accuracy. A key feature of the proposed framework is that the learned models provide an estimate of the true reaching error. With the proposed active exploration strategy, which is based on maximizing an lower bound on real fill-distance, the necessary data to learn a forward model that is useful to learn the inverse could efficiently be generated in the high dimensional control input space.

The developed framework was evaluated both on a simulated musculoskeletal model of a human arm with 2 joints and 6 muscles as well as on a real muscle-driven robot which mimics the human arm model. The real robot has 2 joints and is articulated by 5 pneumatic muscle spring units. In both situations, the proposed method could efficiently learn an accurate inverse model and is able to estimate the reachable set to control the systems to a desired configuration in the complete possible workspace.

Despite the fact that it is unrealistic to assume that the true forward model comes from the same RKHS as induced by the chosen kernel and its hyperparameters, it turned out in the experiments that also in practice the assumptions of the theoretical derivations can be fulfilled reasonably well, without extensive tuning of the parameters. Therefore, the derived bounds are actually useful in practice, can be computed easily and give tight predictions.

It is fascinating that in most experiments after about 200 data points an already accurate inverse model was learned that covers the complete reachable set, considering the fact that 200 data points in a 6 dimensional input space are extremely sparse.

A. References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[2] A. Baranes and P.-Y. Oudeyer. “Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots”. In: *CoRR* (2013).

[3] A. Bayer, S. Schmitt, M. Günther, and D. Haeufle. “The influence of biophysical muscle properties on simulating fast human arm movements”. In: *Computer Methods in Biomechanics and Biomedical Engineering* 20.8 (2017).

[4] F. Berkenkamp, A. P. Schoellig, and A. Krause. “Safe controller optimization for quadrotors with Gaussian processes”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2016.

[5] B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters. “Learning inverse kinematics with structured prediction”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[6] I. Brown, S. Scott, and G. Loeb. “Preflexes — programmable high-gain zero-delay intrinsic responses of perturbed musculoskeletal systems”. In: *Society of Neuroscience, Abstracts* 21 (1995).

[7] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters. “Control of Musculoskeletal Systems using Learned Dynamics Models”. In: *Robotics and Automation Letters* (2018).

[8] Y. Cui, T. Matsubara, and K. Sugimoto. “Pneumatic artificial muscle-driven robot control using local update reinforcement learning”. In: *Advanced Robotics* (2017).

[9] B. Damas and J. Santos-Victor. “An online algorithm for simultaneously learning forward and inverse kinematics”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[10] D. Driess, P. Englert, and M. Toussaint. “Active learning with query paths for tactile object shape exploration”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[11] D. Driess, D. Hennes, and M. Toussaint. “Active Multi-Contact Continuous Tactile Exploration with Gaussian Process Differential Entropy”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2019.

[12] D. Drieß, P. Englert, and M. Toussaint. “Constrained Bayesian Optimization of Combined Interaction Force/Task Space Controllers for Manipulations”. In: *Proceedings of the Int. Conf. on Robotics and Automation (ICRA)*. 2017.

[13] D. Driess, H. Zimmermann, S. Wolfen, D. Suissa, D. Haeufle, D. Hennes, M. Toussaint, and S. Schmitt. “Learning to Control Redundant Musculoskeletal Systems with Neural Networks and SQP: Exploiting Muscle Properties”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2018.

[14] A. D'Souza, S. Vijayakumar, and S. Schaal. "Learning inverse kinematics". In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. 2001.

[15] A. Feldman. "Once more on the equilibrium-point hypothesis for motor control". In: *Journal of Motor Behavior* (1986).

[16] K. Gruber, J. Denoth, E. Stüssi, and H. Ruder. "The Wobbling Mass Model". In: *International Series on Biomechanics* (1987).

[17] D. Haeufle, M. Günther, A. Bayer, and S. Schmitt. "Hill-type muscle model with serial damping and eccentric force–velocity relation". In: *Journal of biomechanics* 47.6 (2014), pp. 1531–1536.

[18] C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, and M. Asada. "Real-Time Inverse Dynamics Learning for Musculoskeletal Robots based on Echo State Gaussian Process Regression". In: *Proceedings of Robotics: Science and Systems*. 2012.

[19] M. Haruno, D. Wolpert, and M. Kawato. "MOSAIC Model for sensorimotor learning and control". In: *Neural Computation* (2001).

[20] H. Hatze. "A general myocybernetic control model of skeletal muscle". In: *Biological Cybernetics* (1978).

[21] T. Hesselroth, K. Sarkar, P. P. Van Der Smagt, and K. Schulten. "Neural network control of a pneumatic robot arm". In: *IEEE Transactions on Systems, Man, and Cybernetics* (1994).

[22] A. V. Hill. "The heat of shortening and the dynamic constants of muscle". In: *Proceedings of the Royal Society of London. Series B - Biological Sciences* (1938).

[23] M. Hulliger. "The mammalian muscle spindle and its central control". In: *Reviews of Physiology, Biochemistry and Pharmacology*. Springer Berlin Heidelberg, 1984.

[24] M. Jordan and D. E. Rumelhart. "Forward models: Supervised learning with a distal teacher". In: *Cognitive Science* (1992).

[25] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the International Conference on Learning Representations* (2014).

[26] G. Klute, J. Czerniecki, and B. Hannaford. "McKibben artificial muscles: pneumatic actuators with biomechanical intelligence". In: *in Proceedings of the International Conference on Advanced Intelligent Mechatronics*. 1999.

[27] D. Koert, G. Maeda, G. Neumann, and J. Peters. "Learning Coupled Forward-Inverse Models with Combined Prediction Errors". In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2018.

[28] Y. Lee, M. S. Park, T. Kwon, and J. Lee. "Locomotion Control for Many-muscle Humanoids". In: *ACM Trans. Graph.* 33.6 (2014). ISSN: 0730-0301.

[29] D. Liu and E. Todorov. "Hierarchical optimal control of a 7-DOF arm model". In: *Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. 2009.

[30] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. "Automatic LQR Tuning Based on Gaussian Process Global Optimization". In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 2016.

[31] A. R. McNeill. *Principles of animal locomotion*. Princeton University Press, 2003.

[32] J. Mockus. “On Bayesian Methods for Seeking the Extremum”. In: *Proceedings of the IFIP Technical Conference*. 1974.

[33] F. Mörl, T. Siebert, S. Schmitt, R. Blickhan, and M. Günther. “Electro-mechanical delay in Hill-type muscle models”. In: *Journal of Mechanics in Medicine and Biology* 12.5 (2012).

[34] D. Nguyen-Tuong and J. Peters. “Model learning for robot control: A survey”. In: *Cognitive processing* 12 (2011).

[35] J. Peters and S. Schaal. “Learning Operational Space Control.” In: *Robotics: Science and Systems*. 2006.

[36] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. 2012.

[37] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[38] R. Rayyes, D. Kubus, and J. Steil. “Learning Inverse Statics Models Efficiently With Symmetry-Based Exploration”. In: *Frontiers in Neurorobotics* (2018).

[39] M. Rolf. “Goal babbling with unknown ranges: A direction-sampling approach”. In: *Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL)*. 2013.

[40] M. Rolf, J. J. Steil, and M. Gienger. “Online Goal Babbling for rapid bootstrapping of inverse models in high dimensions”. In: *Proceedings of the International Conference on Development, Learning and Epigenetic Robotics*. 2011.

[41] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 2002.

[42] R. Shadmehr. “Actuator and Kinematic Redundancy in Biological Motor Control”. In: *Visual Structures and Integrated Functions*. Ed. by M. A. Arbib and J.-P. Ewert. Springer Berlin Heidelberg, 1991.

[43] D. Suissa. “Modeling, Control and Optimization in Human Motor Control: A Simulation Study of a Physiological Human Arm”. MA thesis. University of Stuttgart, 2017.

[44] P. Van der Smagt and K. Schulten. “Control of pneumatic robot arm dynamics by a neural network”. In: *Proceedings of the World Congress on Neural Networks*. 1993.

[45] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004.

B. Declaration of Academic Integrity

I, Danny Drieß, hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Date, Place	Danny Drieß