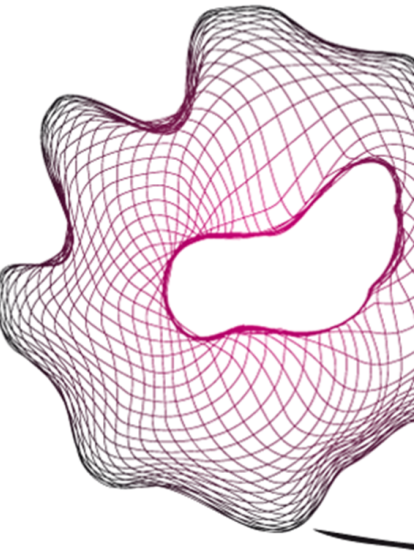
UNIVERSITY OF TWENTE.

Faculty of Engineering Technology



Generalization capabilities of a learning from demonstration framework capturing a human controller

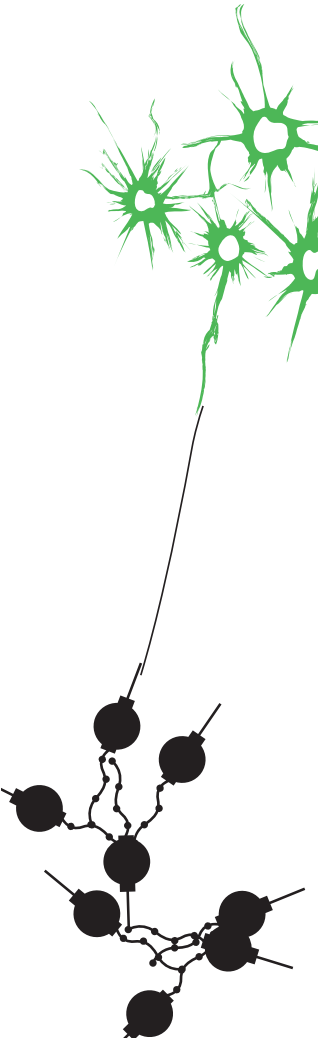
S.L.J.O. (Stephan) Jaspar

M.Sc. Thesis

April 2023

Committee:
dr. E.H.F. van Asseldonk
dr. ir. D. Dresscher
ir. A.H.G. Overbeek
dr. ir. M. Vlutters

BE-905,
Department of Biomechanical Engineering
Faculty of Engineering Technology,
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands



RAM ROBOTICS AND MECHATRONICS

GENERALIZATION CAPABILITIES OF A LEARNING FROM DEMONSTRATION FRAMEWORK CAPTURING A HUMAN CONTROLLER

S.L.J.O. (Stephan) Jaspar

MSC ASSIGNMENT

Committee:
dr. ir. D. Dresscher
dr. E.H.F. van Asseldonk
ir. A.H.G. Overbeek
dr. ir. M. Vlutters

April, 2023

009RaM2023
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Abstract

In 2013, the World Health Organisation (WHO) estimated the shortage of healthcare workers at approximately 17.4 million worldwide. A possible solution to alleviate the health care worker shortage could be found in robotics system performing basic routines task. However, these systems are still limited in their use, especially in dynamically changing environments. Learning from Demonstration (LfD) has the potential to eliminate these limitations. In robotic LfD, robots learns a new task based on only a few demonstrations from an expert. Currently, the non-reinforcement LfD frameworks are capable of learning the motion of simple tasks, but it remains unclear how well these frameworks generalize. Generalization is an important aspect, it eliminates the need to relearn the framework when something changes within a task. This exploratory study investigates whether it is possible to capture a human controller with a non-reinforcement (LfD) framework for closing a valve task. Based on literature, GMM/GMR was deemed to be most suitable framework to capture a human controller. The human controller for the valve task was modelled as P(D)-controller, admittance systems and corresponding feedback loops.
The GMM is learned with five demonstrations with random initial conditions in a range between 0 and 2π rad. The interpolation and extrapolation capabilities were tested by reproducing the respective controller over a range of initial conditions both inside and outside the range of 0 and 2π rad. The reproduction performance of the GMM/GMR was evaluated by visual inspection of the error and the RMSE between the demonstration of the model(ground truth) and reproduction. It lead to the result that GMM/GMR is capable of learning and reproducing the PD-controller, but with a limited extrapolation capability. The extrapolation is limited by the spread of the learning data. On the contrary, the GMM/GMR was not capable to capture the admittance model due to a singularity in the input-output relation. To conclude: GMM is not suitable to capture a human controller and different methods should be used to capture a human controller such as reinforcement learning or system identification.

Contents

Chapter 1

Introduction

In 2013, the World Health Organization (WHO) estimated the shortage of healthcare workers at approximately 17.4 million worldwide. The estimated shortage of healthcare workers in 2030 still exceeds 14 million [1]. The implications of these predicted shortages were particularly evident during the corona pandemic. These circumstances were further strengthen due to the shortage of Personal Protective Equipment (PPE) [2], leading to an increased risk of corona infections which could prohibit healthcare providers from working [3]. Furthermore, the PPE shortage imposed a moral and ethical dilemma for healthcare workers [4], as they had to choose between helping their patients or reduce the risk of infection by not helping patients. Therefore, *Yang et al.* [5] designed a robot which is controlled via teleoperation. The goal of the robot was to perform basic routines such as check-ups on patients, delivery of food and/or medicine, operation of medical equipment and disinfection of surfaces [5]. By fulfilling these objectives, the robot showed the potential to reduce person-to-person contact, lower the risk of infection, and decrease the consumption of PPE.
Although this solution is promising, it does not alleviate the shortage in healthcare workers. The robot still has to be controlled by a team of healthcare workers. Hence, it would be beneficial if the robot also could learn from the worker during the teleoperation period such that after a few task demonstrations the robot works autonomously.

1.0.1 Learning from demonstration

Learning from Demonstration (LfD) can bring the above mentioned scenario a step closer to reality. In LfD, robots (and computers) learn new skills/tasks via a demonstration of an expert (demonstrator). The demonstrator could be a human or another machine. The aim of LfD is that the end-user can learn new tasks/skills to the system without preliminary programming knowledge. However, it is important to

mention that LfD is not limited to solely reproduce tasks. A LfD framework should be able to generalize over the learned demonstrations to perform tasks which are new to the robot. Generalization deals with the question how the system should respond when it encounters an unknown state [6]. Thus, LfD eliminates the need for a robotics expert each time the robot has to be (re)programmed. It is therefore suitable in a fast changing setting, because the robot can for example indicate that it does not know how to perform a task. A nurse can in that case directly teach the task to the robot.

1.0.2 Stages of learning from demonstration

LfD consists of five stages: the demonstrator, data acquisition, data modelling, execution and refinement [7]. The demonstrator stage is concerned with the question ”Who will be performing the demonstration?” Typically, a human is used as demonstrator, but a different (robotic) system is also be suitable as demonstrator. The quality of reproduction is limited by the demonstration, because an incorrect demonstration leads to incorrect reproduction.
In the data acquisition stage the learning data is collected. The goal is to extract crucial information about the task during the demonstration. However, the demonstrator and robot interact and perceive the world in a different manner. The issue on how to convert the data from the demonstrator 'world' into the robot 'world' is known as the correspondence problem [8]. A solution to the correspondence problem can already be found in the interface/methodology used to record the data, which is known as the demonstration interface. An example of interface without correspondence problem is when the demonstrator physically guides the robot during the demonstration. If the interface does not solve the correspondence problem a separate mapping function has to be found by for example inverse kinematics [8].
The data modelling stage consists of deriving and learning a policy of the demonstrated task. The learned policy should not only be able to reproduce the learned data, but also generalize over new scenario's. Generalization allows to perform new tasks without teaching them.
The learning methods are classified in a low- or high-level tasks based on the complexity of the task [7] [9]. A low-level task is defined as primitive motion such as grasping, pushing and placing [7]. The low-level tasks are trajectories that could be described with one policy. The high-level task is composed of the low-level tasks and deals with the question how low-level tasks can be used to perform a more (new) complex task.
The execution and refinement stages are the last stages of LfD. In the execution stage the learned policy is reproduced/executed. It is important the system is able

to refine the task during reproduction, because this allows for adaptability the new environments. The system should be able to the relearn the new data while maintaining the old data (the refinement stage), which is called incremental learning [7], [9]. The incremental learning refines the task execution.

1.0.3 Related literature

The quality of high-level tasks dependent on the quality of low-level tasks. As high-level tasks are composed of low-level tasks. The focus in this research will therefore be low-level tasks. Currently, it is possible to learn low-level tasks. For example in the study by *Kormushev et al.* [10] an ironing task and door closing task were learned with multiple demonstrations. It captures the variability (differences) in the demonstrations. A two-step procedure was introduced in which first the position is encoded and later the force profile, which allowed for a better encoding of the force profile.
In the research conducted by *Zeng et al.* [11] a pushing task was learned. The task is captured by learning the joint probabilities between the position, velocity, force and stiffness variable. The position is used as input in the LfD framework to obtain the velocity, stiffness profile and force. The last example can be found in *Lin et al.* [12], where a grasping and manipulation framework is learned via the usage of LfD. The grasping force was estimated based on a video. The framework can successfully pick-up a bell pepper and other objects.

1.0.4 Problem statement

An important aspect of the data modelling stage is being able to generalize to new situation. It allows robots to adapt better in dynamically changing surroundings, which is required in for example healthcare. However, none of the above mentioned frameworks have been tested with respect to their generalization capabilities. Only variability (the small differences in execution) over the different demonstrations has been taken into account.
A benchmark is lacking to test if frameworks can adapt to new situations, such as a different shape during the ironing task or a different initial position in the grasping and manipulation task. In the study by *Wang et al.* [13] it is learned how to wipe a table clean. A generalizability test was implemented by changing the position of the dirt. The parameters of the framework were altered to work with the new goal position. It means that for each new goal position the framework has to be altered, which is not feasible in a fast dynamic changing environment where the goal changes.
Most of the current literature only takes into account one specific task without gen-

eralizing. The trajectory of the task is learned which limits the generalization capabilities of the system. A step towards better generalization would be learning the mechanism behind the execution of the task, the controller/controller scheme a human applies to execute the task. A simple example to support this statement is a P(D)-controller, which can generalize over different initial conditions without changing the parameters of the controller. Generalization eliminates the need to train the robot on every single movement it is supposed to know. This therefore raises the question: *"Which currently existing learning from demonstration framework is suitable for encoding a human (controller) executing an arbitrary low-level task?"* To the best of our knowledge this is the first study which address the research question and is therefore of exploratory nature.

1.0.5 Outline

The research question is answered by first presenting an overview from literature identified LfD frameworks in Chapter 2. After which in the methods (Chapter 3) a demonstration interface is selected (Section 3.2.1). An encoding method is selected (Section 3.2) based on a theoretical comparison. A task is selected in Section 3.3). A model of the task combined with the selected LfD framework is presented in Section 3.4 of the methods. In Section 3.5 of the methods it is described how simulations are performed and evaluated. The results are presented in Chapter 4 and discussed in Chapter 5. The research is concluded by summarizing and concluding with respect to the main findings in the conclusion (Chapter 6).

Chapter 2

Background

The background gives an overview of the learning from demonstration (LfD) frameworks that can be used for the data acquisition stage. The correspondence problem (how to map from demonstrator space to robot space) and high-level tasks framework are beyond the scope of this background. The frameworks are limited to the non-reinforcement learning frameworks. They require less demonstrations compared to reinforcement learning [14], which is better suited for quickly learning new tasks.
A demonstration framework is split in a demonstration interface and encoding method. A demonstration interface is the device/method on which demonstration is captured. The encoding method is the algorithm used to learn the demonstration. The process of LfD is shown in Figure 2.1. A concise overview of demonstration interfaces including the main limitations of the interfaces is described first. Next a description of commonly used learning from demonstration frameworks is treated.

2.1 Demonstration interface

The purpose of this section is to get the reader acquainted with the main types of demonstration interfaces. A demonstration interface is the device/method used by demonstrator to execute the demonstrations. The demonstration interface can be divided into three subcategories kinesthetic teaching, teleoperation and passive observation [9], [15].

2.1.1 Kinesthetic teaching

Kinesthetic teaching is an interface in which the demonstrator physically guides the robotic system to perform a task [9], [15], [16]. The motion is captured by the onboard sensors of the robot, the demonstration is recorded in the frame of the robot. Capturing the demonstration in the world of the robot eliminates the correspondence

problem. The problem is in this case solved by the demonstrator during the demonstration.
However, the demonstrator is limited by the dynamics of the movement during the demonstrations. For example, the goal is to pick up a glass of water. The demonstrator is limited in his/her movements, because it has to physically guide the robotic arm to the glass of water. Once the demonstrator reaches the glass both hands are required to close the fingers of the robotic arm for grasping the glass. The demonstration quality therefore depends on the how smooth the demonstrator executes the demonstration within their limited movement freedom.

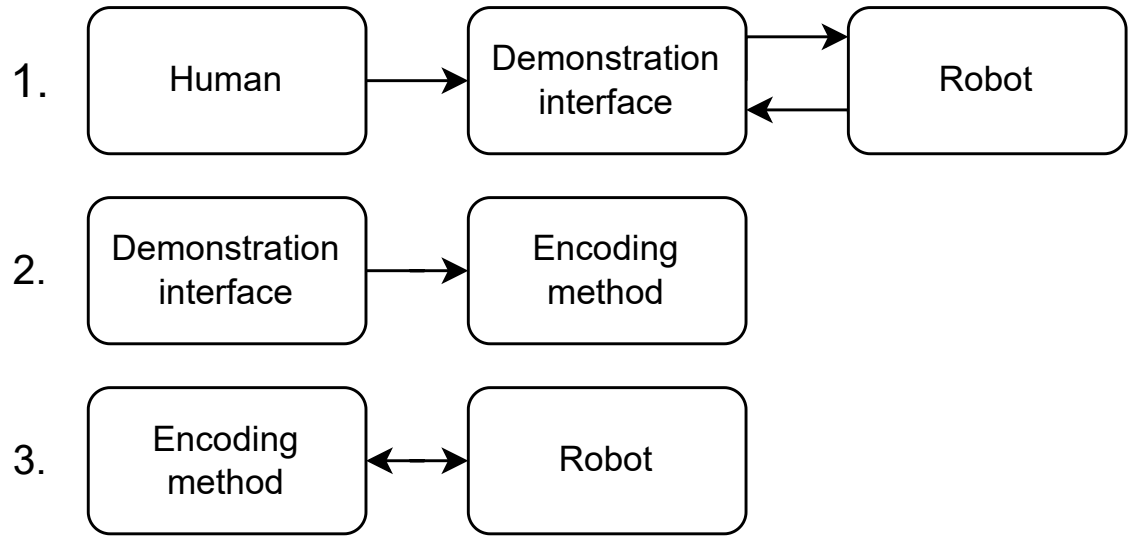


Figure 2.1: First, the human applies the demonstration via a demonstration interface on the robot. Next, the gathered data is used to learn the encoding method. In the last step the encoding method should be able to control the robot.

2.1.2 Teleoperation

In teleoperation, the robot is controlled remotely by the operator. The operator controls the robot via a teleoperation device, which can range from a game console controller to a haptic device. The correspondence problem now shifts from the demonstrator towards the teleoperation interface. Therefore, the correspondence problem is solved when framework is correctly implemented.
To obtain a natural feeling of controlling the robot practices is required by the operator [9]. Especially, if the morphology of the robot is completely different compared to the human. Additionally, a teleoperation device and software is required to control the robot.

2.1.3 Passive observation

In this method, the demonstration is recorded via a motion capture system, wearables or via the robot's camera [9], [15], [16]. The interface is more user friendly compared to other two, because the demonstrator can move/perform the task without constraint. The main limitation is that the correspondence problem has to be solved, a mapping has to be found between the demonstration space and the robot space. Furthermore, additional hardware is often required in the form of a motion capture system or wearables.

2.2 Learning framework

The learning frameworks can be divided into two groups, the probabilistic and deterministic frameworks. These are distinguished based on the type of data encoding. The probabilistic category uses multiple demonstrations for learning, which implements variability in the data. This group consists of the Gaussian Mixture Model (GMM), Hidden Markov Models (HMM) and Probabilistic Movement Primitives (ProMP). On the contrary, the deterministic category learns from one demonstration. It does therefore not include variability in the reproduction. The deterministic category consists of the Dynamic Motion Primitive (DMP). The frameworks are discussed in the above mentioned order, but between the HMM and ProMP the regression method used for GMM and HMM is discussed.

2.2.1 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is clustering technique that uses a superposition of Gaussians distributions. A superposition of Gaussians is a sum of multiple Gaussians, as seen in Figure 2.2. The GMM is the sum of the first (blue) and second (red) Gaussian. The GMM models the joint probability between the input and output spaces [17]. The clustering type used by GMM is soft clustering in which a data point is classified based on probabilities. Each classified data point has a probability that it belongs to a class. On contrary to hard clustering, where the data point is classified into one class. It is important to note that GMM can have multiple dimensional input and output variables. An example of multiple dimensional encoding can be found in Figure 2.3, where position over time is plotted and encoded with 5 components/Gaussians. It can be seen that the Gaussians are two dimensional covering the time and position variable.

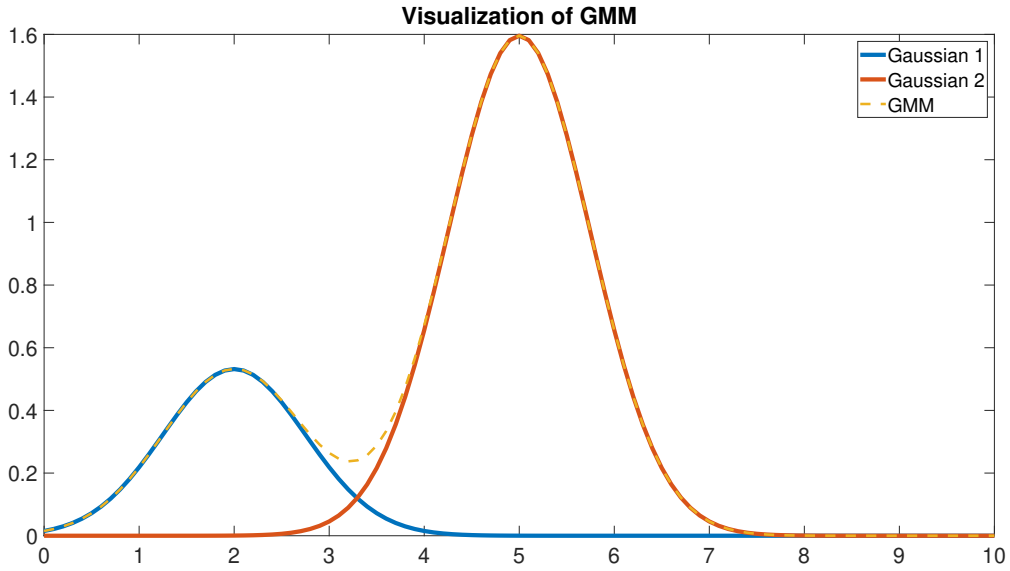


Figure 2.2: Two Gaussians (continuous line) and the sum of both Gaussians (dashed line) are plotted. It aids with understanding that GMM is a superposition of Gaussians.

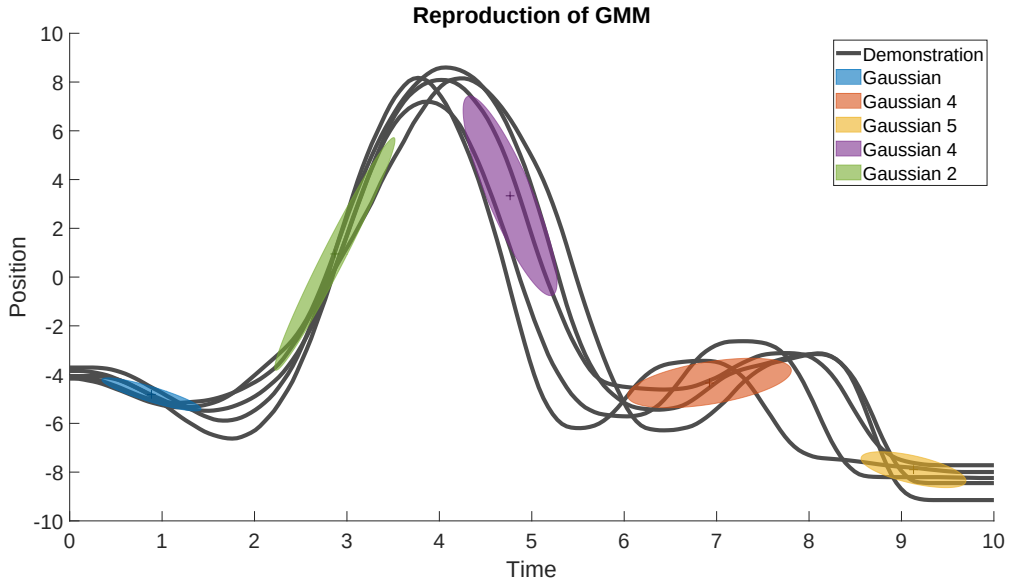


Figure 2.3: An example of how GMM can be used to encode a demonstration. The coloured ellipses represent the standard deviation of the different Gaussians components. The dots correspond to the mean of the respective Gaussian. The dashed lines are the different demonstrations which have a comparable shape.

2.2.1.1 Mathematical description

The mathematical notation of a GMM is found in Equation 2.1.

$$P(\mathbf{q}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{q}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) \tag{2.1}$$

where $P(\mathbf{q})$ is the likelihood of q belonging to the GMM, K is the number of Gaussians, k is one Gaussian component, $\mathbf{q}$ is the point that will be classified, $\boldsymbol{\mu}_k$ the mean of Gaussian k, $\boldsymbol{\Sigma}_k$ is the covariance of Gaussian k and π_k is the prior of Gaussian k. Sometimes it is more likely that data points belong to a specific class/-component, a cluster is biased. This bias is captured with the prior. The prior can be freely chosen or learned, provided that the sum of the mixing coefficients equals one.
After pre-processing the data, such as filtering and removing outliers, the model parameters are learned via the Expectation-Maximization algorithm, which is a maximum likelihood estimation of the mean, covariance and mixing coefficient. The result of E-M depends on the initial guess of the parameters as it is susceptible to local optimum [18].

2.2.1.2 Time dependency

A property of GMMs is that they are time-independent. The GMM classifies a data point independent with respect to a previous classified point. Hence, it does not take into account time and will only do so if there is an explicit state that includes time (dependency), such as time or velocity. Explicitly including time as a state introduces the problem that demonstrations for the same task have to be aligned in terms of time [19]. Spatial relations between demonstrations are incorrectly modelled if the executed task remains the same, but shifted over time. Therefore, the time alignment is required to correctly model the joint distributions over different demonstrations. It does allow for spatial variations between demonstrations.
A method to align data is Dynamic Time Warping (DTW). It uses spatial similarities to align the data, which can be seen in Figure 2.4. However, DTW limits the demonstration by requiring the same initial and final state of the signal [20]. Furthermore, the lack of time dependency in GMMS results in a classification problem if an input is reached twice in the same movement. For example, the position has a value of two and one wants the classify that point in Figure 2.3. It is impossible to classify the point to the orange or yellow Gaussian component with only the position information. A solution could be including additional dimension to the GMM in which this singularity does not exist. In the case of the example, an additional input signal should be used with a unique value when the the position reaches two for the first time and a different value when the position reaches two for the second time.

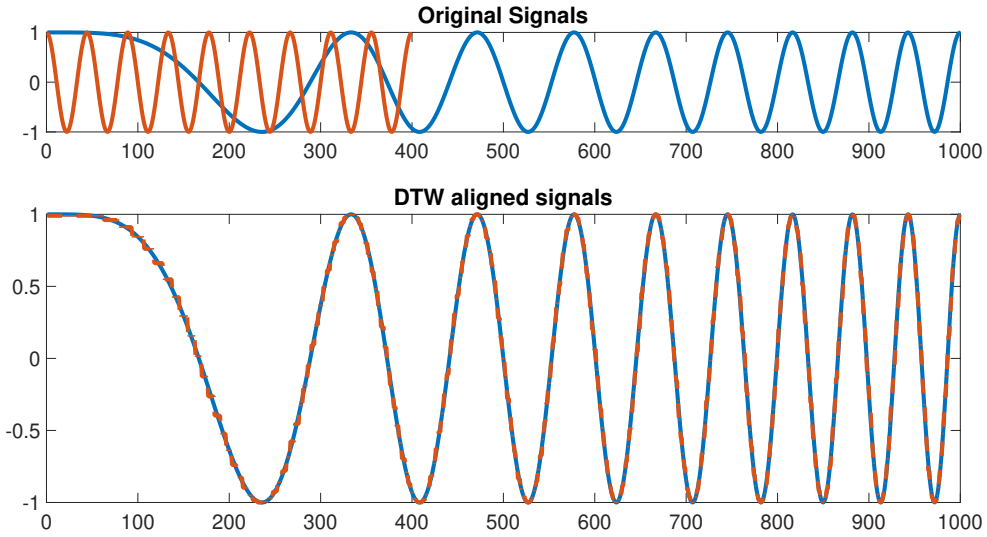


Figure 2.4: Example in which DTW was used to align the red data with the blue data. In the upper plot the blue line is an approximately stretched out version of the red line. The lower part shows how the red line is aligned with the blue line after applying DTW.

2.2.1.3 Curse of dimensionality

A disadvantage of the E-M algorithm is that the number of Gaussians/components are fixed before learning. An incorrect number of components results in under- or overfitting of the demonstrations. In addition, the curse of dimensionality is also prevalent for the GMM. An increase in components adds for each Gaussian an N-dimensional vector of means and N by N covariance matrix, where N is the number of variables that should be encoded. Therefore, GMMs are often in high-dimensional space and relearning is unfavourable as this is a time-consuming process [21]. A method to find an optimum between the number of components and informational loss is via the use of information criterion. The information criterion determines if the cost of adding another state weighs against the goodness of fit.
The curse of dimensionality is also present in feature selection (selecting what variables should be used for encoding), more features results in a higher dimensional space. There are two solutions to this problem, feature selection and feature transformation. In feature selection, it is determined which variables are useful by looking at the relation of input and output data. For example, the correlations between a potential input variable and the output variable. In feature transformation a dimension reduction is performed by transforming the original variables to a lower dimensional space [19]. An example of feature transformation is principal component analysis.

2.2.2 Hidden Markov Models

A Hidden Markov Model (HMM) is an encoding technique used in time series or sequence analysis where the goal is to recover a data sequence that is not directly observable [22]. As stated in *Guan et al.* [23] "*One can think of HMMs as GMMs with latent variables changing over time*". A HMM consists of hidden states (latent variables) whose outputs are the observations/demonstrations. The states are linked with each other and one can stay in the same state or jump to a different one. However, the next action only depends on the current state, which means it satisfies the Markov property.
An example of HMM encoding can be seen in Figure 2.5. It encodes the same movement as in Figure 2.3. The difference is the state transition probabilities. State transition probability describes the probability to jump from the current state to a different state or remain in the same state. The state transition diagram can be found on the right in Figure 2.5. It shows that HMM starts in state 1 and can only go to state 2 or stay within the state. The only way to arrive at the final state is by passing the other four states.

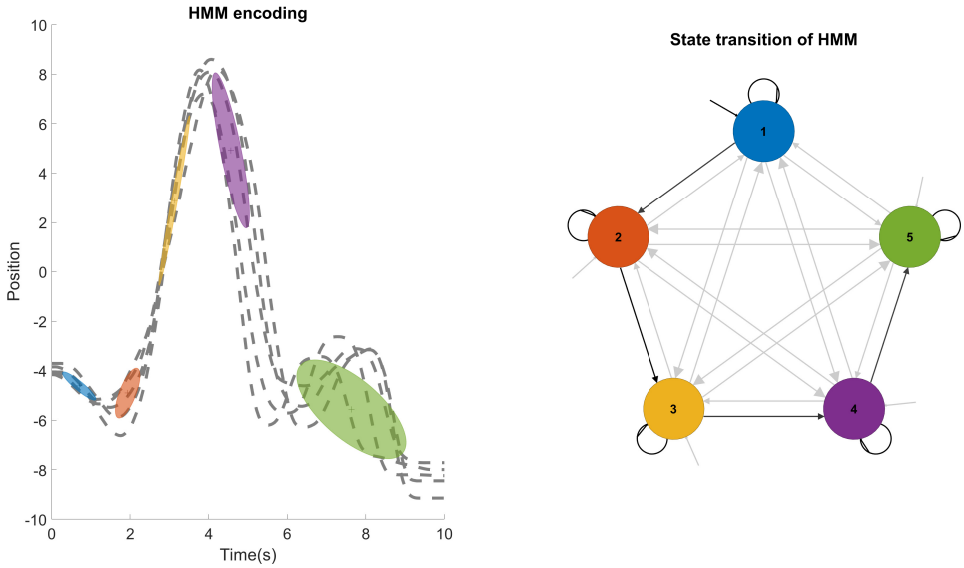


Figure 2.5: An example of how HMM encodes multiple demonstrations. On the left, the demonstrations are plotted including the observation probabilities. The observation probabilities are modelled as Guassians and are the colored areas. On the right, the transitions between states can be found. The color of state corresponds to the same coloured Gaussian of observation probability on the left diagram.

2.2.2.1 Mathematical description

The probability that the model with parameters ($\boldsymbol{\theta}$) explains the observed data $(\mathbf{X})$ can be found with Equation 2.2, where $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is the joint probability between the observed data and latent variables $(\mathbf{Z})$. The joint probability is defined as in Equation 2.3.

$$P(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{2.2}$$

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \pi \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A}) \prod_{m=1}^{N} p(\mathbf{x}_m|\mathbf{z}_m, \boldsymbol{\phi}) \tag{2.3}$$

To increase readability $|\theta$ (given the model parameters) is omitted. In Equation 2.3, π is the initial state probability. It is a vector containing the probability for each state that it will be the starting state.
The transition probability ($p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A})$ is the probability to jump from one state to another or stay within the state. Matrix $\mathbf{A}$ stores the transition probabilities. Each state has an observation probability $(p(\mathbf{x}_m|\mathbf{z}_m, \boldsymbol{\phi}))$, which indicates the probability that an observation belongs to a state given the parameters of the observation probability density function $(\boldsymbol{\phi})$. The observation probability can be encoded by a single distribution, for example Gaussian or multiples Gaussians, like a GMM. The HMM parameters are learned iterative via the Baum-Welch algorithm, which is a type of the E-M algorithm.

2.2.2.2 Comparison with GMM

HMM shares the same advantage as GMM such as encoding the probabilistic relation between input and output variables, but also similar disadvantage such as the curse of dimensionality.
However, HMM encodes time dependence indirectly, the next state depends on the current state. It exploits the sequentially in the data [19]. The HMM can align data due to exploitation of the sequentially in the data. Therefore, the system is more robust against scaling time and singularities.
For example, if a certain observation is reached twice at the beginning and during a movement. The GMM might not correctly classify the observation, because it has two options for the same value. The HMM could use the temporal information to correctly classify the point, because it takes into account the previous state. In addition, HMMs can encode several motion alternatives in the same model and partial movements can be trained [24]. This results in a better generalization, because partial movements can be combined in a different order which leads to new movements.

2.2.3 Regression method

The above mentioned encoding methods store data in discrete states/components. Therefore, a regression method is required to translate the components into a smooth trajectory, which can be fed as input to a robot controller. Gaussian Mixture Regression (GMR) is a method to create a smooth trajectory from states with latent variables.

GMR does not directly derive a regression function, but relies on the learned joint distribution via Gaussian conditioning [25]. It uses the joint distribution to estimate the conditional expectation of the output given the input [12]. An example of this can be seen in Figure 2.6, where a Gaussian is plotted (red area). The Gaussian models the joint distribution between the input (x) and output (y). The goal is find the output value y_1 corresponding to the input point x_1 with the aid of GMR. This is done by calculating the difference between x_1 and the mean μ_x (the pink line). This difference is multiplied with the slope (the blue line). The slope is calculated by normalizing the covariance with the variance. The last step is to add the output mean (μ_y). Thus, the GMR computes an output based on a scaled difference between the input mean (μ_x) and input (x_1). In case of multiple Gaussians, the aforementioned calculation is done for each Gaussians and scaled by the normalized likelihood of the input belonging to respective Gaussian (responsibility).

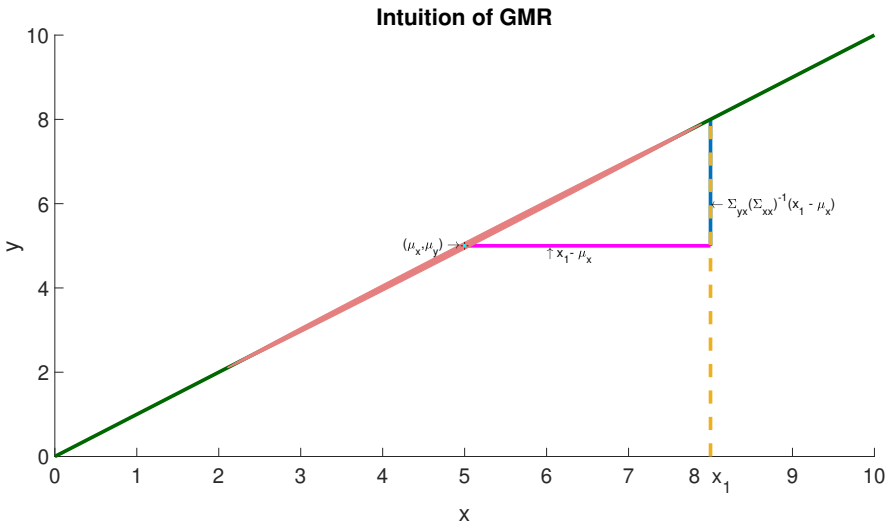


Figure 2.6: An example of a Gaussian (red area) and its mean (cyan dot). The green line is the GMR retrieved trajectory. The yellow striped line shows the desired input. The pink and blue line show how from the input mean an output value is found.

The biggest advantage compared to data driven regression is that it tries to model the conditional distribution, which allows to reproduce data even if there is

data missing [24]. The usage of joint probability density function allows for regression with multiple inputs and outputs. On top of that, the in- and output variables can be decided upon during reproduction, due to the fact that the joint probability is modelled [26]. For example, position, velocity and force are modelled, one of these can be input and the other two output or vice versa. The learning process is also fast and distinct from the retrieval process [18], as the computation time does not depend on the number of data points used to train the model, but on the the number of Gaussian components [25].
For HMM an adaption can be made such that regression model also takes the temporal behavior into account, which should prevent the explicit usage of time as input for the regression [24]. The GMR for HMM is called adapted Gaussian Mixture Regression (aGMR).

2.2.3.1 Mathematical formulation of GMR

The mathematical formulation of GMR is first discussed for GMM, because it is more intuitive compared to HMM. On top of that, HMM can be considered as time sequence expansion of GMM, which means that adapted GMR is an expansion of GMR. The full mathematical derivation of GMR can be found in the appendix (Section A).
As mentioned before GMR is the expectation of the conditional probability of the output (y) given the input (x). The corresponding formulation can be found in Equation 2.4

$$\mathbf{y}_{est} = E(\mathbf{y}|\mathbf{x}) = \sum_{j}^{K} w_j(\mathbf{x})\mathbf{m}_j(\mathbf{x}) \tag{2.4}$$

where y_{est} is the regressed output, K is the number of Gaussians, j is one Gaussian, $w_j(\mathbf{x})$ is the responsibility/weighting. The responsibility shows how well component j explains the data. The last parameter m_j is the conditional expectation for Gaussian j. It is defined as in Equation 2.5.

$$\mathbf{m}_j(\mathbf{x}) = \boldsymbol{\mu}_{\boldsymbol{jy}} + \boldsymbol{\Sigma}_{\boldsymbol{jyx}}\boldsymbol{\Sigma}_{jx}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{jx}) \tag{2.5}$$

Where $\boldsymbol{\mu}_{\boldsymbol{jy}}$ is the mean value for the output of Gaussian j, $\boldsymbol{\Sigma}_{\boldsymbol{jyx}}$ is the covariance between the input and output variables for Gaussian j. The inverse of the variance matrix is called the precision matrix. Variable $\boldsymbol{\Sigma}_{\boldsymbol{jx}}$ is the variance of the input data for Gaussians j and $\boldsymbol{\mu}_{jx}$ is the mean value for the input of Gaussian j.

2.2.3.2 Adapted GMR

The mathematical formulation for the adapted GMR is the same as GMR. However, a forward variable is now used instead of responsibility [24]. The forward variable is

the likelihood of observing a sequence of inputs and being in state j at time t. Due to this the adapted GMR takes into account the sequentially of HMM. Therefore, HMM should be used with aGMR and GMM with GMR. The usage of GMM with aGMR is not possible, because there is no forward variable for GMM. In addition, using HMM with GMR is also not possible, because the HMM does not have a prior.

2.2.4 Probabilistic movement primitives

Probabilistic movement primitives (ProMP) encode demonstrations as as distribution over the trajectories. The distribution is found by applying linear regression to each demonstration. The regression is performed by finding for each demonstration the weights which are multiplied with basis functions. The multiple demonstrations are combined by calculating a Gaussian distribution over the weights. The weight distribution is used to obtain the trajectory distribution. ProMP is not only limited to motion variables (position, velocity, acceleration) it can also include for example sensory variables such as force [27]. An example of ProMP can be found in Figure 2.7. It can be seen that there is a relatively large standard deviation, which is explained by the fact that the variance is calculated based on the variance of the weights. In this case, the weights have a wide spread which results in a large variance of the trajectory distribution.

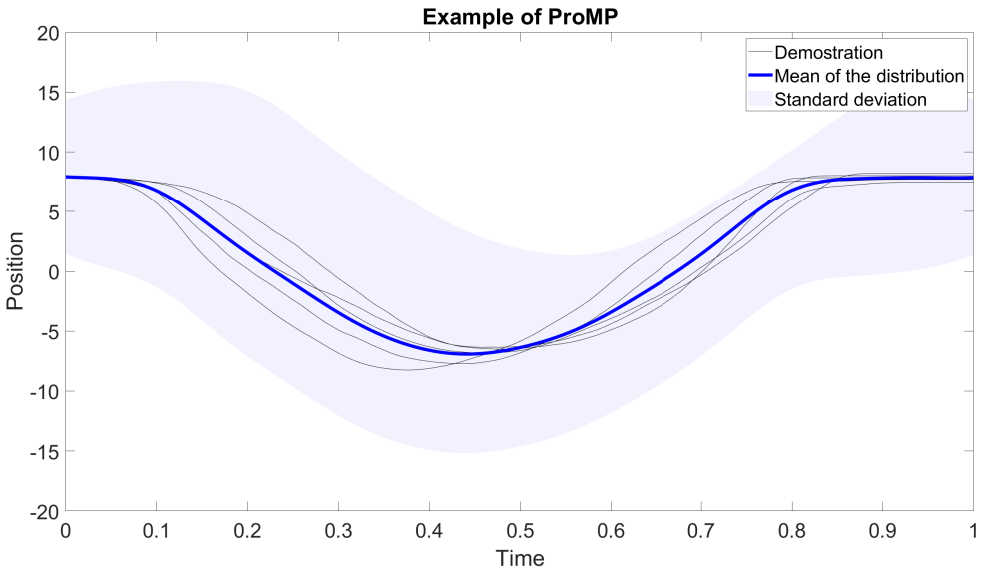


Figure 2.7: An example of how ProMP encodes the demonstrations. The black lines are the demonstrations to learn the distribution. The mean and standard deviation of the distribution is plot with blue. The standard deviation is large, because of the spread in weights between the demonstrations.

2.2.4.1 Mathematical description

The linear regression function for one demonstration can be found in Equation 2.6, where $\mathbf{\Phi_t} = \left[\boldsymbol{\phi}_\mathbf{t}, \dot{\boldsymbol{\phi}}_\mathbf{t}\right]$ are the basis functions for the positions and velocities respectively. The basis functions are a function of the phase variable. A phase variable is a time-based variable that is used to normalize the time of different demonstrations. The vector $\mathbf{w}$ is the weight vector, ϵ is noise/residual and t is the corresponding time step [28]. The basis functions representation reduces the number of parameters compared to the joint distribution for each time step [29].

$$\mathbf{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \mathbf{\Phi}_t^T \mathbf{w} + \epsilon \tag{2.6}$$

The probability of a trajectory (τ) consisting of T time steps given the weight vector $(\mathbf{w})$ is calculated via Equation 2.7 [29].

$$p(\boldsymbol{\tau}|\mathbf{w}) = \prod_t \mathcal{N}(\mathbf{y}_t|\Phi_t\mathbf{w}, \mathbf{\Sigma}_y) \tag{2.7}$$

In order to generalize over the demonstrations, a distribution over the weight vectors is learned via maximum likelihood estimation [28]. The basis functions however remain the same per demonstration. The marginal distribution of the trajectory can now be calculated, because the conditional probability of the trajectory and the distribution of the weight is known. The resulting distribution is found via Equation 2.8.

$$p(\tau;\theta) = \int p(\boldsymbol{\tau}|\mathbf{w})p(\mathbf{w},\theta)d\mathbf{w} \tag{2.8}$$

Where $p(\mathbf{w},\theta)$ is the distribution of the weight with the distribution parameters θ. Assuming that the weight distribution is Gaussian, the probability defined in Equation 2.8 can be learned by rewriting to Equation 2.9. The distribution is now learned with only the mean and covariance of the weights.

$$p(\mathbf{y};\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{y}_t|\mathbf{\Psi}\mathbf{w}, \mathbf{\Sigma}_y)\mathcal{N}(\mathbf{w}|\mu_w, \sigma_w)d\mathbf{w} = \mathcal{N}(\mathbf{y}_t|\mathbf{\Psi}^T\boldsymbol{\mu}_w, \mathbf{\Psi}\Sigma_w\mathbf{\Psi} + \Sigma_y) \tag{2.9}$$

A stochastic linear feedback controller is used to reproduce the mean from this distribution for each time step [27], [29].

2.2.4.2 Phase variable

Applying ProMP on multiple demonstrations requires to have a time domain of equal length for all demonstrations [30], which can be achieved by introducing a artificial clock. The phase variable (artificial clock) normalizes the time domain, for example between 0 and 1. It can be seen as replacement for time such that the location of the basis functions can be controlled [31]. The phase variable can be chosen freely, but

it should be a monotonically increasing function [28]. The different demonstrations should be approximately temporal aligned with respect to the phase variable [31]. Thus, time can be used as phase variable, but only in case the data has the same temporal phase [31]. If that does not hold DTW can be used to temporally align the data. Furthermore, the phase variable allows for temporal modulation, which means that the execution speed can be changed. This is done by changing the decay rate of the phase variable. This allows for a change in execution speed [29].

2.2.4.3 Curse of dimensionality

The ProMP is limited by high dimensional input, because it requires a large number of basis functions [21] and a large covariance matrix. In addition, a longer data sequence requires more basis functions to correctly encode the demonstrations. Therefore, the weight vector is longer and more data is required for regression. The covariance matrix can be restricted in shape by only allowing diagonal components, but this results in losing correlations between variables [32]. On top of that, more parameters such as the number of basis functions, have to be tuned in order to guarantee acceptable results.

2.2.5 Stability of probabilistic types

It is important to mention that it cannot be guaranteed for GMM and HMM that the desired goal will be reached, because the HMM and GMM alone have no means to converge to the desired end-position. A possible solution is adding a spring-damper mechanism, which pulls the system towards the end-position.
The linear feedback controller prevents the ProMP to drift away from the trajectory distribution for small disturbances. However, if the ProMP is not within the vicinity of the linear controller a non-linear controller can be used to bring the robot back to the vicinity of the linear controller [27].

2.2.6 Dynamic Motion Primitive

A Dynamic Motion Primitive (DMP) is a deterministic encoding and reproduction method. "*Dynamic motion primitives (DMPs) is a motion planning method based on the concept of teaching a robot how to move based on human demonstration. To this end, DMPs use a machine learning framework that tunes stable non-linear differential equations according to data sets from demonstrated motions. Consequently, the numerical solution of these differential equations represent the desired motion*" [33].
A DMP consists of a spring-damper system and a non-linear part. The non-linear part, which is also known as the forcing function, is used to encode motion. The forcing function is scaled with a phase variable. The phase variable depends on time,

which means the forcing function also depends on time. The phase variable decays over time such that near the end-time of the reproduction the system behaves as a stable spring-damper system.

2.2.6.1 Mathematical description

The spring-damper ensures (if parameters are properly chosen) that the system converges to its goal position within some steady state error, while the non-linear term encodes the motion. The DMP is formulated as in Equation 2.10 and 2.11. It is important to note that the DMP can only model one dimension, for example velocity in the y-direction.

$$\tau\ddot{y} = \alpha_z(\beta_z(g - y) - \dot{y}) + f(s) \tag{2.10}$$

This is often rewritten into the first-order notation:

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(g - y) - z) + f(s) \\ \tau\dot{y} &= z \end{aligned} \tag{2.11}$$

where y is the demonstrated state while learning or in case of reproduction the desired state, α_z and β_z are positive constants, which can be freely chosen and affect the behaviour of spring-damper-model and g is the goal position of the motion. The system is critically damped by picking constants that obey $\beta_z = \frac{\alpha_z}{4}$. Variable s is the phase variable, which is in the standard case defined as Equation 2.12.

$$\dot{s} = \alpha_s s \tag{2.12}$$

where $\dot{s}$ is the derivative of the phase variable and α_s is the time constant/temporal constant. Function $f(s)$ is the forcing function, the non-linear function that encodes the demonstrated movement. If the forcing term equals zero the system is a globally stable second-order system with $(z, y) = (0, g)$ as attractor [34], which is favourable as this guarantees a stable behaviour. The forcing function is defined as in Equation 2.13, where $\psi(s)$ is a basis function and w_i are the weights. The forcing functions determines the shape of the trajectory.

$$f(s) = \frac{\sum_{i=1}^{N} \psi_i(s) w_i}{\sum_{i=1}^{N} \psi_i(s)} s \tag{2.13}$$

The weights are used to scale the basis functions such that the demonstration trajectory is encoded. They are learned via regression, which is typically performed with only one demonstration.

2.2.6.2 Generalizability

The DMP learns the shape of the trajectory which allows for temporal scaling, scaling in the magnitude of the movement, different initial and goal position during reproduction [35] [36]. However, it requires to adjust the phase variable. The phase variable has to be tuned for a longer/shorter reproduction. The implicit time dependency results in the problem that the shape of the motion is dependent on the phase variable [35]. If the phase variable decays to quickly, the non-linear term will fade before the desired task is finished. Due to the spring-damper the DMP will reach the end-goal, but with a sped-up version of the the desired non-linear trajectory.

2.2.6.3 Multiple demonstrations

A main limitation of DMP is the lack of variability, because DMP learns from one demonstration. However, it is possible to encode multiple demonstrations by using a GMM and GMR as non-linear function [20] [37]. The GMM is able to encode multiple demonstrations via Gaussians components, while GMR is used to retrieve a continuous path from GMM. However, this also introduces the disadvantages of the GMM, such as high dimensionality. In addition, it makes the system more complex while the limitation of the phase variables are still present.

Chapter 3

Method

This chapter discusses how the research question *"Which currently existing learning from demonstration framework is suitable for encoding a human (controller) executing an arbitrary low-level task?"* was answered. Eventually, the suitable LfD framework will be applied on a humanoid robot named Eve. Hence, the demonstration interface was selected such that the findings of this preliminary study could be transferred to the Eve. However, implementation of the learning framework on the robot is beyond the scope of this thesis.
The selection of the LfD framework and task is covered first. The framework is selected in Section 3.2. The task selection is covered in Section 3.3. A model of the selected system and task is presented in Section 3.4. Lastly, in Section 3.5 it is explained how the selected frameworks will be evaluated on their performance with respect to their generalizability.

3.1 Robot

A picture of the humanoid Eve can be seen in Figure 3.1. The robot is controlled via a whole body controller, which accepts position, acceleration and velocity in either task space or joint space as input. However, the Eve does not have force sensors or a vision framework that can be used to process information of the task.

3.2 Framework selection

This section addresses the selection of demonstration interface and learning framework. Subsection 3.2.1 motivates the selection of the demonstration interface. Subsection 3.2.2 evaluates the learning framework based on requirements for encoding a human controller.

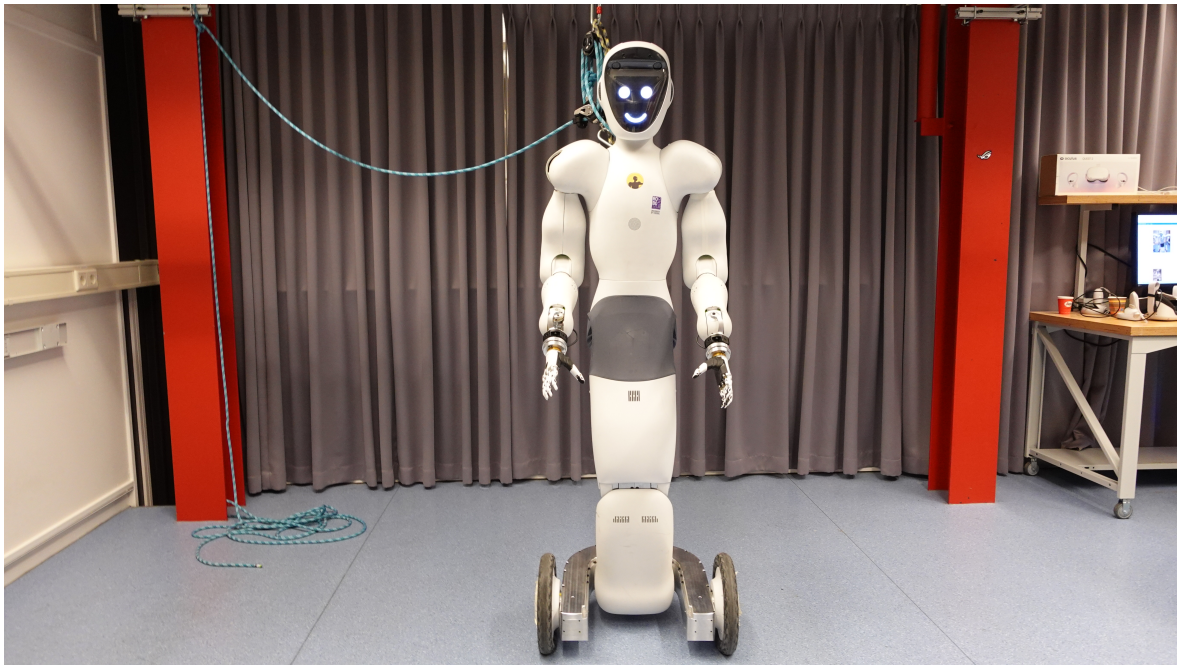


Figure 3.1: The wheeled humanoid robot Eve

3.2.1 Demonstration interface

Several issues should be considered upon selecting a demonstration interface such as the absence of force sensors, spatial limitations during the demonstrations and the correspondence problem. The teleoperation framework deemed to be the most suitable for the Eve based on the following three points:

1. The correspondence problem is solved by the interface. Thus, the mapping between the operator space and the robot space is already found.

2. Teleoperation provides force information during demonstrations and reproduction without the usage of force sensors. Teleoperation eliminates the need for physical modifications on the robot for force information during reproduction and learning. These physical modifications are outside the scope of the research.
   Instead, a virtual spring is placed between the desired position of the operator and the actual position of the robot (Figure 3.2). In the first step, the spring between Eve and the desired position is in rest. In the second step the desired position moves to a new position, the spring elongates and pulls the Eve position towards the new desired position. In the final step the spring is in rest again, but the Eve and desired position have moved to a different position.

3. A proven teleoperation framework exists, It is an adaptation of the by *Franken et al* [38] introduced two-layered passivity framework. The framework will be

used combined with an Omega, which is a haptic feedback device. It can be seen as a joystick which provide force feedback to the operator.

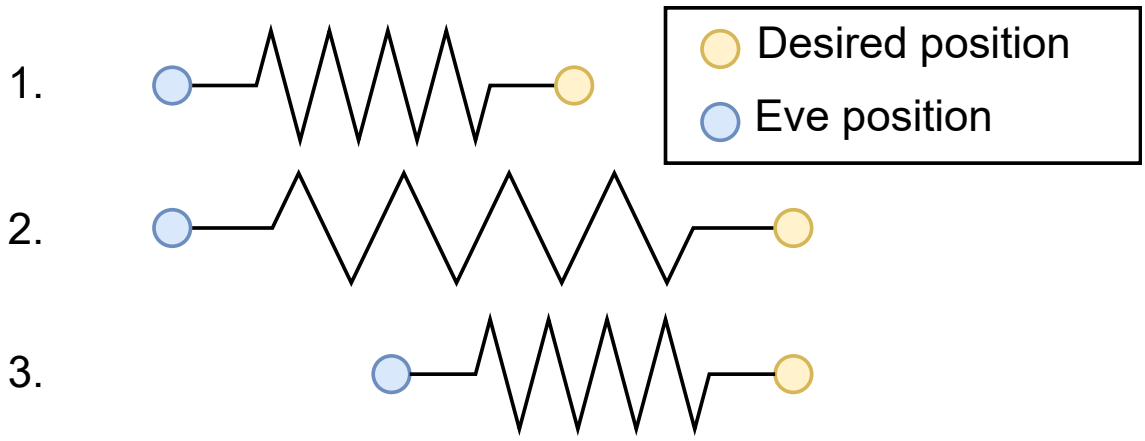


Figure 3.2: An example of the working of the virtual force. It shows how the desired position pulls the Eve towards the desired position.

3.2.2 Choice of encoding method

In order to evaluate if an encoding method is suitable to capture a controller the following three requirements were defined:

1. Encode multiple dimensions. Typically, a task consists of multiple dimensions and degrees of freedom. An example is a pick and place task in which force and position information are required. In this case, the position is used to locate the object and the force determines how the object should be picked up. Therefore, the desired framework should be able to capture the different degrees of freedom and dimensions.

2. Capture the relation between variables and how variables correlate with each other. In a P-controller, for example, it is known that the relation between the input and output is a constant parameter. It allows to capture underlying mechanism of the demonstrations.

3. Robust against noise, because the recorded data will contain noise. The framework should be able to extract the relevant information preferably without signal processing. Signal processing does not generalizes to different tasks. As an illustration, for some task high frequency information is important

while for other task lower frequency information is more relevant. In addition, the data obtained during reproduction will also contain noise. Therefore, the framework should be robust against noise.

Four possible encoding methods were identified from literature and explained in the background(*Section 2.2*): DMPs, ProMPs, HMM, and GMM. These methods were evaluated against the requirements (Table 3.1). In addition, two methods were evaluated through preliminary simulations, which lead to GMM being the most suitable for encoding a controller.

Table 3.1: Comparison of the frameworks against the requirements

Encoding framework	Encode multiple dimension	Underlying relation between variables	Robust against noise
GMM	x	x	x
HMM	x	x	x
DMP	x		
PromP	x		x

DMPs and ProMPs were eliminated as they solely reproduce the trajectory without modelling the underlying relationship between the input and output data (requirement 2). The two remaining candidates (HMMs and GMMs) both deemed to be viable based on theoretical grounds. These methods utilize the spatial pattern to encode trajectories by encoding the joint probability of the in- and output signals (requirement 2), which increases robustness against noise [39] (requirement 3).
However, based on preliminary simulation, HMMs were found to be unsuitable. The details of these preliminary simulation (including a derivation of (a)GMR) are found in Appendix A. The exploratory simulation consists of a reproducing a one dimensional trajectory. The HMM incorrectly reproduced the demonstrated trajectory, which is likely related to the forward variable. In addition, HMM can be seen as an expansion of GMM [23] [40]. A HMM has implicit time dependency due to Markov assumption, which increases the complexity of the method.
The incorrect reproduction in the exploratory simulations combined with the increased complexity resulted that GMM is the most suitable framework to capture the human controller.

3.3 Task selection

The performance of the LfD framework is evaluated using a task. The task that has to be learned and reproduced is closing a valve. The closing valve motion can also

be used in a healthcare setting, for example closing a bottle or a tap.
The task consists of two parts to model: linear dynamics and the stopping condition of the valve. The first part is the linear dynamics, which ensures that the valve rotates until the desired angle is reached. The linear dynamics are modeled with a mass-damper system. The final angle is the stopping condition of the valve, which results in zero acceleration and velocity of the valve. The stopping condition is modelled with a function that gradually saturates at the final angle. In addition, a model of when the task is closed for the demonstrator is required to the perform the simulations, the force threshold.
For a human it is clear when a valve is closed, but a quantitative description is tedious. Therefore, the following definition is introduced for a closed valve from the point of view of the demonstrator: the torque exerted by the operator on the valve exceeds a torque threshold and the angular velocity imposed by the operator is (near) zero. After the threshold has been reached the operator stops rotating the valve.
The threshold represents the minimum torque required to close the valve. The minimum torque is induced by the resistance of the valve. The resistance is a non-linear damping, which depends on the position of the valve. When the Eve is near the end-position the resistive torque increases due to high non-linear damping.
The threshold is defined in Equation 3.1, where τ_e is the torque measured/felt by the end-effector/arm, τ_r is the resistive torque, $\dot{\theta}_e$ is the rotational velocity measured/felt by the end-effector/arm and $\dot{\theta}_{human}$ is the velocity that the operator uses to open the valve.

$$\begin{aligned} &\textbf{If } \tau_e \geq \tau_r \text{ AND } \dot{\theta}_e \approx 0 \\ &\quad \dot{\theta}_{human} = 0 \\ &\textbf{end} \end{aligned} \tag{3.1}$$

Although the use of a force threshold would allow for more realistic simulation results, the report focuses only the valve dynamics. The force threshold increases the complexity of the controller captured by the GMM, while it is unclear if the GMM can capture a linear controller. The valve dynamics also include a non-linear threshold, but the GMM only captures the human and Omega dynamics.
The valve task allows to generalize over different initial conditions for the same controller, because the initial condition does not affect the force threshold in Equation 3.1. The threshold torque depends on the valve its parameters such as the damping constant or the radius of the valve. Additionally, generalization over different initial conditions is typically used in daily scenarios. The position at which a valve/tap is open is often unknown to an operator. Generalizing over different initial conditions is therefore considered as a benchmark.

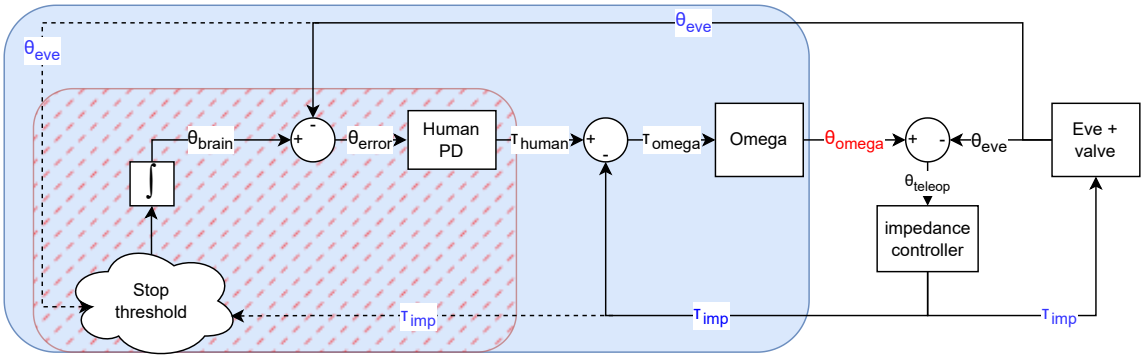
3.4 Framework model

The model used in this study simplifies the task to linear dynamics in one dimension. This section provides an overview of the model including the subsystems. A schematic drawing of the corresponding model can be found in Figure 3.3A. The goal is to capture the blue area, the Human-Omega, a combination of linear controllers in a feedback loop. The GMM/GMR should reproduce the Omega angle (red colored text in Figure 3.3A) based on the states of the Eve and impedance torque signals (the colored text in Figure 3.3A). The system with GMM/GMR in the results in the schematic of Figure 3.3B.

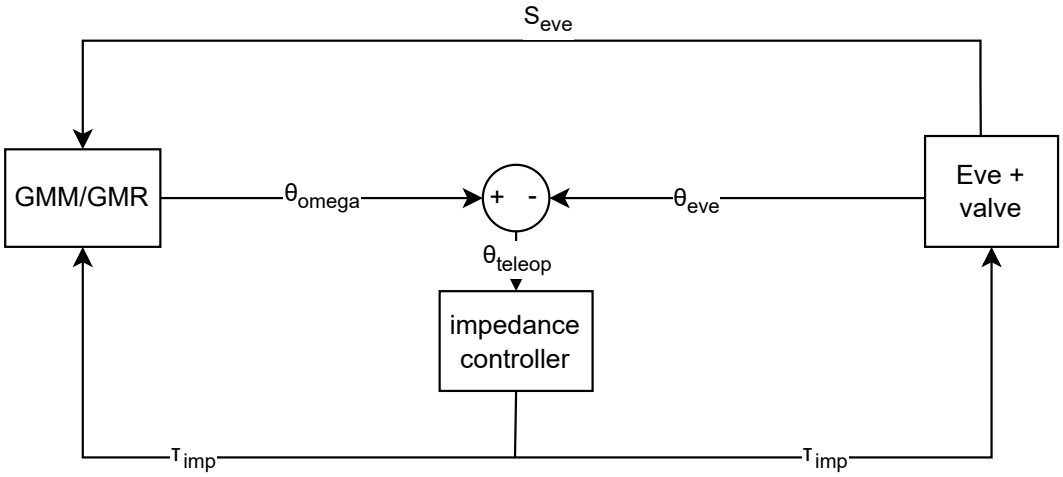
The human has a desired end-position in mind, which could be seen as the current visible angle of the omega/EVE plus some offset (θ_{brain}). The brain angle increases over time until the valve is closed. The human exerts a torque on the Omega based on the error (θ_{error}) between the brain angle (θ_{brain}) and the angle of Eve (θ_{eve}). The human torque (τ_{human}) combined with the impedance torque (τ_{imp}) is applied on the Omega. The impedance torque is the virtual torque exerted on the Eve and fed back to the human. The human torque and impedance torque have opposite sign, because the impedance torque acts against the human movement. It allows the user to feel for example the weight of the robot arm while moving.

The Omega maps the Omega torque (τ_{omega}) to a desired Eve angle (θ_{omega}) via an admittance model, which is a mass-damper system. The impedance controller (which is in this case a PD-controller) minimizes the error between the Eve and Omega (θ_{teleop}). The angle set by the Omega is used as input for the impedance controller and is the angle to which the Eve converges.

The resulting torque is the impedance torque (τ_{imp}), which is used as feedback torque to the human and input for the Eve. The Eve and valve are modelled in series and therefore the dynamics are concatenated. If the maximum angle of the valve is reached the velocity and acceleration of the Eve converge to zero such that the Eve+valve does not move. That results in an impedance torque which increases until the saturation torque of Equation 3.1 is achieved.



(3.3A)



(3.3B)

Figure 3.3: Figure to clarify what should be encoded by the GMM/GMR. The original model can be found in part 3.3A. The blue area is the part learned by the GMM. The red striped area is the human. The threshold set for the human is described by Equation 3.1. The blue colored part is removed and substituted by the GMM/GMR, which is visible in part 3.3B. It is important to note that multiple states of the Eve are used as input (S_{eve}), because it is not defined yet what states will be used for learning.

3.5 Experimental plan for the simulations

The goal of the simulation is to assess if GMM/GMR can learn, reproduce and generalize the Human-Omega controller (blue box in Figure 3.3A). To achieve this goal the GMM/GMR should encode a PD-controller, admittance system and corresponding feedback loops. It is unclear if the GMM/GMR can capture all these subsystems in once. Therefore, the complexity of the learning task was reduced by learning P(D)-controller and admittance separately. The controllers are split in different system, because the input-output relation of the linear controller is not affected by the dynamics of the system. The steps in which the simulations are performed are found in Figure 3.4. The first simulation performed was learning a P-controller, which is

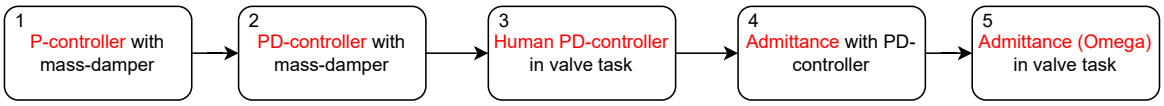


Figure 3.4: Flow diagram of how the simulations are performed and in what system. It shows the sequence of how simulations are performed. The red text indicates what was learned by the GMM.

placed in a loop with a mass-damper. The goal is to bring the mass to a desired goal position while changing the initial conditions. It is considered a baseline task, because the mapping between input and output is a linear line. The second simulation is the same as the P-controller simulation, but now a PD-controller is used. A PD-controller is more complex with respect to a P-controller, because of the differentiation and additional dimension. The simulations of step one and two are used to gain insight into GMM. It does not have a quantitative metric which defines it as successful. The third step is learning the PD-controller in the valve task, because the simulations of step one and two should give insight on how to capture the PD-controller in the valve task. The fourth step is learning a mass-damper placed in a loop with a PD-controller. The admittance with a P-controller has not been tested, because the input output relation of the admittance is the same when in a loop with a PD-controller. Lastly, the admittance model in the valve task was learned.

3.5.1 Initialization of GMM

In all steps the GMMs were learned with five demonstrations and initialized with k-means algorithm [41]. The demonstrations were learned by E-M algorithm. The

algorithm stops when the number of iteration exceeds 100 or if the likelihood increase is less than 0.0001. The simulations were performed with Matlab R2021b and Simulink. The code used for GMM, GMR, the Gaussian probability density function and E-M was based on the code written by [41]. The initial conditions for these demonstrations were randomly chosen between 0 and 2π rad with a uniform distribution. The initial conditions used to learn the GMM are found in Table 3.2.

Table 3.2: The randomized initial conditions of the five demonstrations used to learn the GMM.

Learning IC [rad]	0.41π	0.3553π	0.3664π	1.9145π	1.1205π

3.5.2 Evaluation of reproduction

The performance of reproduction is evaluated by the root mean square error and visual inspection of the error between demonstration and reproduction. A combination of these gave insight on the quality of reproduction. The root mean square error (RMSE) between the reproduction and ground truth combined with visual inspection was used to evaluate the reproduction [42]. The ground truth is defined as the simulation result of the system with the corresponding controller, thus the model from which the GMM/GMR learns. The following two properties were assessed by visual inspection of the error for all the simulations:

1. Interpolation is tested by reproducing with five initial conditions equidistant in a range from 0 rad to 2π rad.

2. Extrapolation is evaluated by reproducing initial conditions outside the range of 0 rad to 2π rad.

For the P(D)-controller and admittance simulations in separate systems two additional properties were evaluated to obtain insight in the quality of reproduction. The first additional property is the number of Gaussians. Even though the number of Gaussians can be deduced from the input-output relation, it is not guaranteed to result in a reproduction that follows the demonstration. The second additional property is the length of steady-state relative to the transient behaviour, because the steady-state does not add any new information to the reproduction. In the case of only one Gaussian it can therefore lead to a bias while learning the input and output relation. Therefore, this property is only tested on GMMs with one Gaussian.
Only for the valve-task a quantitative metric is used to define a reproduction as successful, because it will eventually be reproduced on the Eve. The metric is based on the accuracy of the Eve. An absolute error of less than 0.03 rad over the whole

trajectory defines the task as successful. This metric was determined by the measurement error of the end-effector of eve, which is 1.5 mm and the valve is assumed to have a radius of 5 cm, which results in 0.03 rad error. If a reproduction is below this value it can be seen as a perfect reproduction, because it is smaller than the measurement error of the Eve.
Details on the simulations for the specific systems are described in the following sections: Subsection 3.5.3 the mass-damper model includes the (evaluation) parameters used for the simulations. Subsection 3.5.4 describes the parameters of the valve task and includes more details about the human PD-controller simulations. Lastly, Subsection 3.5.5 describes the detailed procedure of the admittance system.

3.5.3 P(D) with a mass-damper

The first simulation performed was a P(D)-controller in a feedback loop with a rotational mass-damper. The goal of the P(D)-controller is to bring the mass to an arbitrary position of approximately π rad. A schematic of the system can be found in Figure 3.5. The standard equation of motion for a rotational mass-damper is used. A parallel unfiltered discrete P(D) controller was used to model the P(D) controller.

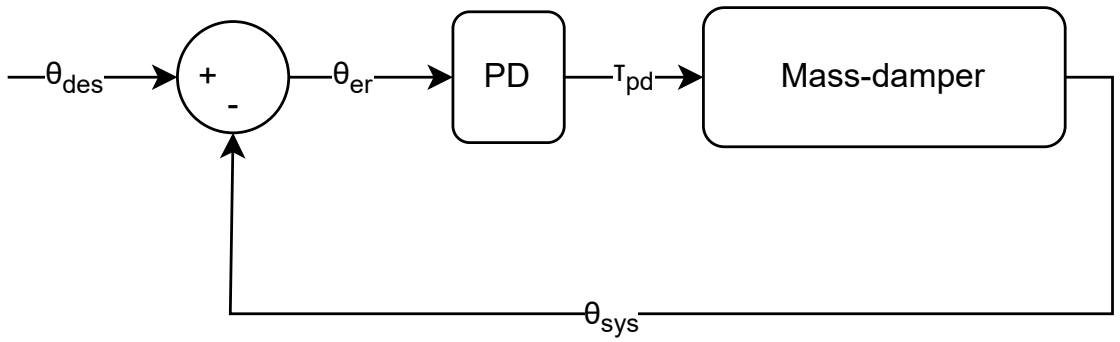


Figure 3.5: A simple schematic of the encoding of P(D)-controller in a loop with the mass-damper system. The input of the system is a constant desired angle (θ_{des}). This desired angle is subtracted by the angle of the mass-spring-damper (θ_{sys}), which results in the angle error (θ_{er}). The P(D)-controller calculates the resulting force (τ_{PD}) for the mass-damper.

The parameters for the rotational mass-damper system can be found in Table 3.3. In which T_1 is the demonstration time used to learn a GMM, while T_2 is the shorter demonstration time used to learn a GMM to explore the influence of steady state on reproduction. The parameters were tuned such that a stable non-oscillating steady state position was found.

Table 3.3: Mass-damper parameters

I_{rot} [kg]	0.05
D_{rot} [$\frac{\text{Ns}}{\text{m}}$]	5
T_1 [s]	5
T_2 [s]	1
dt [s]	0.002

The exact parameters used for evaluating the interpolation and extrapolation are found in Table 3.4. These are only used for the P(D)-controller combined with the mass-damper.

Table 3.4: Initial conditions used to evaluate the interpolation and extrapolation capabilities of GMM/GMR combined with a mass-damper.

Interpolation IC [rad]	0.00	0.50π	1.00π	1.50π	2.00π
Extrapolation IC [rad]	0.00	2.50π	5.00π	7.50π	10.00π

3.5.3.1 P-controller

As mentioned before the P-controller is considered as baseline, because its controller mechanism is trivial. It is a linear mapping between the angle error θ_{er} and torque τ_{pd}, which are the variables used to learn the P-controller.
The GMM/GMR should be able to generalize over a line with only one Gaussian. An example of how GMM captures a line can be found in Figure 3.6. The Gaussian has the same direction as the line. The GMR assumes a linear relation between the input and output while the slope is determined by the direction of the main component variance (which is in this case aligned with the line). The linear regression function is multiplied with the responsibility (weighting), which is based on the normalized probability that a point belongs to a Gaussian (it should always be one due to the fact that there is only one Gaussian).
Despite the fact that one Gaussian should be sufficient, simulations with five Gaussians were also performed to obtain an intuition on how the number of Gaussians affects the learning and reproduction. The Gaussians could for example be evenly spread over the angle error-force line and cover a larger area compared to one Gaussian, which should result in a bigger extrapolation range compared to one Gaussian. The corresponding parameters of the P-controller used for simulations can be found in Table 3.5. These were manually tuned such that the system converged to an angle of approximately π rad.

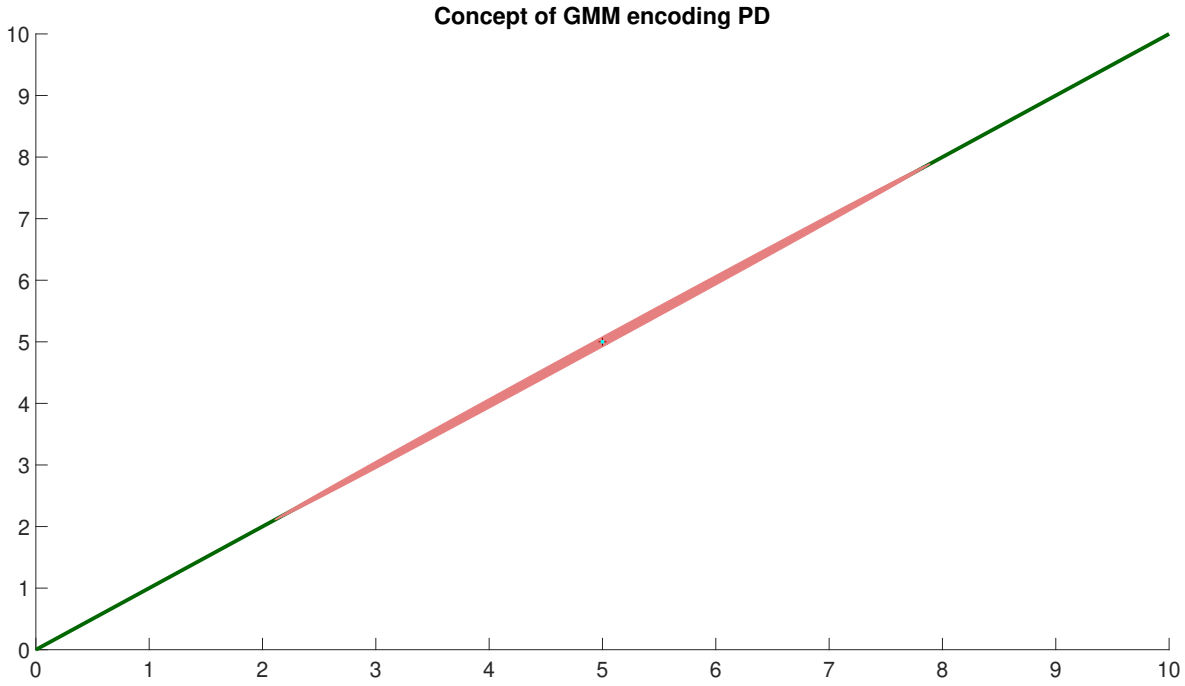


Figure 3.6: An example of how GMM with one state captures a linear line. It is used to motivate why one Gaussian should be sufficient to inter- and extrapolate a P-controller.

Table 3.5: Parameters used for the P-controller simulations.

K_p [$\frac{\text{Nm}}{\text{rad}}$]	25
K_i	0
K_d [$\frac{\text{Nms}}{\text{rad}}$]	0

3.5.3.2 PD-controller

The PD-controller is more complex compared to the P-controller, because of the differentiation. This complexity is evident in variable selection: if one would retain the P-controller combination of input and output for a PD-controller the GMM has to differentiate. GMMs cannot differentiate, because of their time independence.
In this specific scenario differentiation can be prevented via the variable selection. The velocity of the mass is the same as the derivative of the error, because the goal position remains constant over time. Therefore, the angle error (θ_{er}), angular velocity ($\dot{\theta}_{er}$) of the mass and torque (τ_{pd}) will be learned by the GMM.
The relation between the input and output variables should result in a plane within the velocity, error and torque space, which can be seen in Equation 3.2.

$$\tau_{pd} = k_p \underbrace{\theta_{error}}_{x} + k_d \underbrace{\dot{\theta}_{error}}_{y} \tag{3.2}$$

A plane is favourable for the encoding process of the GMM, due to the simple linear structure. Therefore, the reasoning as for the P-controller can be expanded to three dimensions (a plane). One Gaussian should be sufficient to encode the plane, because one Gaussian can capture the direction of the plane. In addition, the direction of the GMR is determined by the direction of the Gaussian, which was clearly visible in Figure 3.6.
Similar to the P-controller simulations with one and five Gaussians were performed. The corresponding parameters can be found in Table 3.5. These were manually tuned such that the system converged to an angle of approximately π rad.

Table 3.6: Parameters used for the PD controller during simulations.

K_p [$\frac{\text{Nm}}{\text{rad}}$]	25
K_i	0
K_d [$\frac{\text{Nms}}{\text{rad}}$]	2

3.5.4 Parameters of the Human-Omega model

The human PD-controller as seen in Figure 3.3A will be learned, but as already mentioned in the task selection (Section 3.3) the stop reaction of the human is not modeled. The reason for this is that it increases the complexity of what GMM/GMR should learn, while it is not certain if the GMM/GMR can learn a linear controller. The exact parameter specifications can be found in Table 3.7. These parameters are obtained by tuning the system such that the behaviour of Figure 3.7 is observed.

Table 3.7: Parameters used within the valve task model. The dynamics of the Eve and valve are concatenated, because they are modelled in series.

Damping Eve and valve [$\frac{\text{Nms}}{\text{rad}}$]	2
Inertia Eve and valve [kgm^2]	0.25
Damping Omega [$\frac{\text{Nms}}{\text{rad}}$]	0.75
Inertia Omega [kgm^2]	0.01
Stiffness impedance controller [$\frac{\text{Nm}}{\text{rad}}$]	35
Damping impedance controller [$\frac{\text{Nms}}{\text{rad}}$]	2
Stiffness human [$\frac{\text{Nm}}{\text{rad}}$]	10
Damping human [$\frac{\text{Nms}}{\text{rad}}$]	1
Estimated rotation velocity[$\frac{\text{rad}}{\text{s}}$]	2.5
Max valve angle [rad]	5π
Length of demonstration [s]	10
Sample time [dt]	0.002

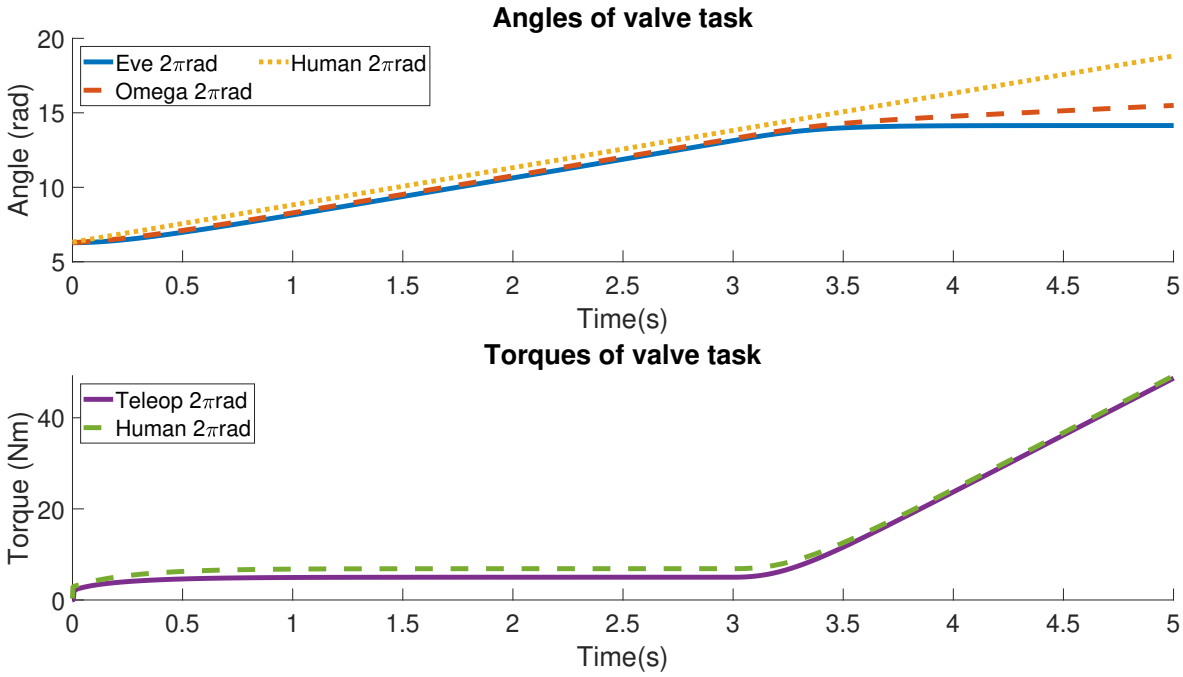


Figure 3.7: The Eve angle, which is the same as valve angle, and the Omega angle starts at 2π rad, while the human angle starts at 2.035π rad and rotates until 4.5π rad, where the angle of the Eve saturates. The Human and Omega continue rotating, because the stop condition of the human was not implemented. This results in an increasing error between the Eve and Human which lead to an increasing torque.

The initial conditions of the system are specified in the following manner: The Omega and Eve start at the same position with the initial velocity and acceleration set to 0. The initial value of the human angle is the initial Eve/ Omega angle with an 0.035 rad offset as it is not realistic that the human starts at exactly the same position of the Eve. Furthermore, it results in a zero error which leads to a system that does not move due to the multiplication with zero.
The number of Gaussians used for learning the valve task are based on the findings of learning the PD-controller in the separate system. The same random initial conditions (Table 3.2) were used to learn the task, but the initial condition of the extrapolation behavior is limited to 4π. Therefore, extrapolation will be performed over a smaller range which can be found in Table 3.8.

Table 3.8: Reproduction initial conditions for the start position of the valve to test the generalization capabilities.

Valve IC [rad]	0.00	1.00π	2.00π	3.00π	4.00π

3.5.4.1 Variable selection to learn the human PD-controller

The mathematical description of the human PD-controller is found in Equation 3.3

$$\tau_{human} = k_p\theta_{error} + k_d\dot{\theta}_{error} \tag{3.3}$$

where (τ_{human}) is the human torque, (θ_{error}) the error between brain angle and Eve angle and $(\dot{\theta}_{error})$ is the velocity of the error. Similar to the PD-controller with mass-damper differentiation was prevented by taking the error and velocity of the error as input. However, the velocity of the error $(\dot{\theta}_{error})$ cannot be obtained directly from the valve task model (see Figure 3.3A). Therefore, a related variable to the error is required, which is found by expanding and differentiating the error (θ_{error}). The velocity of the θ_{brain} is constant, because it was assumed that the operator closes the valve with a constant velocity. The Eve velocity ($\dot{\theta}_{Eve}$) therefore captures the same dynamics as the error velocity ($\dot{\theta}_{error}$).

$$\begin{aligned} \theta_{error} &= \theta_{brain} - \theta_{Eve} \\ \dot{\theta}_{error} &= \dot{\theta}_{brain} - \dot{\theta}_{Eve} \end{aligned} \tag{3.4}$$

3.5.5 Admittance model

An admittance controller uses a different mapping compared to a PD-controller. The output angle depends on the double integral of the angular velocity (input). It makes use of integration to obtain the desired output, which can be seen in Equation 3.5. The GMM is not capable of integration, because it requires memory. The integral action is circumvented by taking the double integral of torque an double integral of the velocity as input variable. Furthermore, taking the double integral of the variables as input results that the input-output path should be on a plane, which can be seen in Equation 3.5. The similar reasoning as for PD-controller about the plane holds here. Therefore, only on Gaussian was sufficient to encode the demonstration, but simulations were performed with one and five Gaussians.

$$\begin{aligned} I\ddot{\theta}_{omega} &= \tau_{humanerror} - D_{omega}\dot{\theta}_{omega} \\ I\dot{\theta}_{omega} &= \int (\tau_{humanerror} - D_{omega}\dot{\theta}_{omega})dt \\ I\theta_{omega} &= \underbrace{\int\int \tau_{humanerror} dt\, dt}_{z} - D_{omega}\underbrace{\int\int \dot{\theta}_{omega}\, dt\, dt}_{y} \end{aligned} \tag{3.5}$$

However, the initial conditions of these double integral have to be set to zero (or at least remain constant) while learning and reproducing the admittance system. If this is not the case the demonstrations will not lie on a plane. The integral of the angle cannot be used as input, because the initial condition changes per demonstration.

Therefore, the velocity was also used as output and fed back after integrating twice as input. This is visible in Figure 3.8.

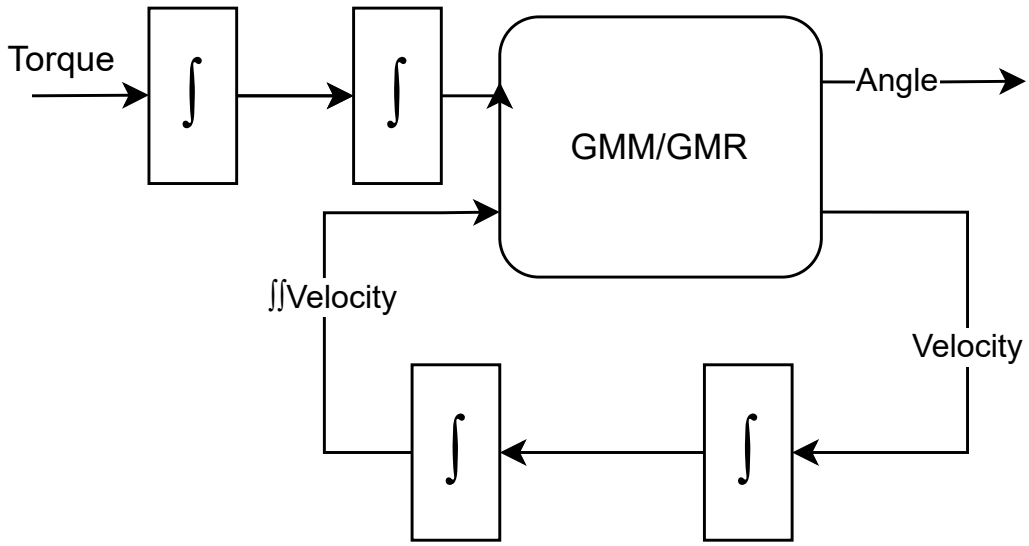


Figure 3.8: Illustrations of how the input and output variables work the GMM/GMR. The output is in a direct loop also input.

3.5.5.1 Evaluation

The admittance model was tested with the same procedure and criteria as for the PD-controller (Section 3.5.2). Similar simulations as with the PD-controller was performed (Section 3.5.3). First the admittance was learned in a separate system as seen in Figure 3.5, but the mass-damper is learned instead of the PD-controller. The next step was learning the Omega in the valve task (Figure 3.3A).
The performance of reproduction for the separate system was evaluated based on the interpolation and extrapolation capabilities of the GMM, which is done by reproducing the initial conditions mentioned in Table 3.4 while the GMM was learned with the initial conditions mentioned in Table 3.2. In addition, for the separate system the influence of steady state duration and number of Gaussians is evaluated. As mentioned before, the steady state duration can lead to a bias for a one Gaussian GMM. It can shift the mean towards the steady state, which might lead to an inaccurate representation of the task.
Similarly to the human PD-controller reproduction the Omega was learned with the number of Gaussians based on the results of the separate system. In addition, the system was evaluated with the same parameters (Table 3.7) and initial conditions

(Tab 3.8) as the human PD-controller. The variables that will be used to capture the Omega in the valve task are the angle of the Omega (θ_{Omega}) and the torque applied on the Omega (τ_{Omega}).

Chapter 4

Results

In this chapter, the simulations results are presented in the order as described in Figure 3.4. First, the results of the P-controller (Section 4.1) are presented. After which the results of the PD-controller are presented (Section 4.2). The simulations results of the human PD-controller are presented in Section 4.3 and the admittance results are presented in Section 4.4.

4.1 P-controller

The interpolation capabilities of GMM were evaluated by reproducing the P-controller with equidistant initial conditions, ranging from 0 rad to 2π rad. Figure 4.1 is an example of the P-controller reproduction by the one Gaussian GMM, where the mass has an initial condition of 0.5π rad. Based on visual inspection, the reproduction seems to exactly follow the demonstration, but the error (Figure 4.2A) indicates the reproduction is not the same.
The error at 0 s is the largest of the whole trajectory. This results from the initial conditions being at the largest distance to the mean of the One Gaussian GMM. The mean of one Gaussian GMM(0.0209 rad)is the same as the mean of all demonstration combined and the steady-state value has the largest contribution to the mean. This results from the fact that the different initial conditions converges to approximately the same steady-state value. Since the mean of the GMM is close to the steady state value, the error converges to 0 rad.
In contrast to the one Gaussian GMM, the error of the five Gaussians GMM starts approximately at zero (Figure 4.2B. The error peaks at around one second for all initial conditions except π rad. The peak is a result of GMR sampling from the incorrect Gaussian. However, the input eventually is correctly classified which results in a decrease of error, which result in the converging to approximately zero error. The misclassification is evident in Figure 4.3, where the third Gaussian (fifth subplot) has the highest probability, while the mean of the second Gaussian is closer to the value

of the angle error (marker in second plot). The second Gaussian should have the highest probability instead of the third Gaussian.

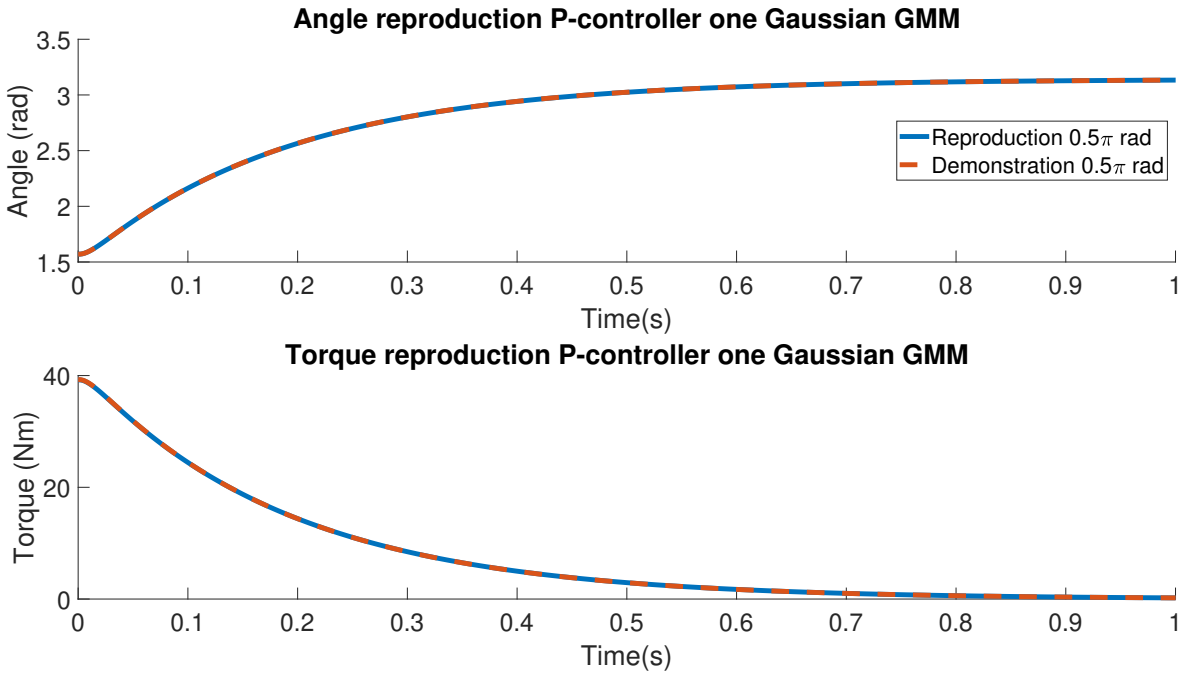


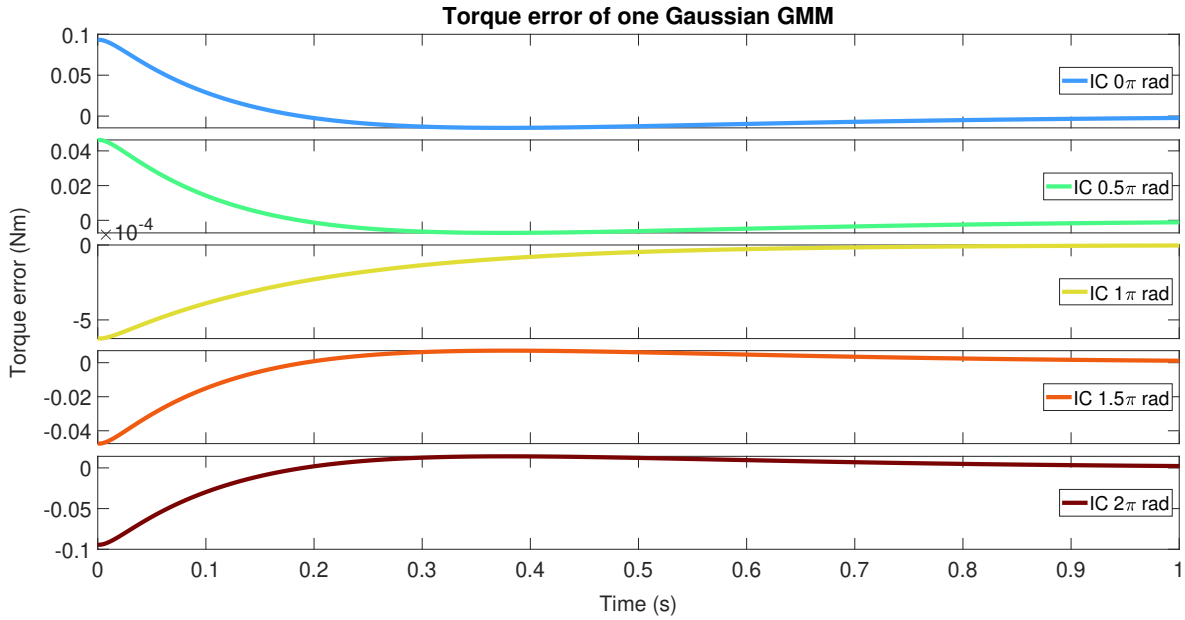
Figure 4.1: An example of the one Gaussian GMM reproducing the P-controller with the mass having an initial angle of 0.5π rad, where the dashed red line shows the ground truth. The solid blue line (plotted underneath the red line) is the reproduction of the one Gaussian GMM. The top graph shows the angle of the mass-damper. The bottom graph the torque of the P-controller, which is the output of the GMM/GMR. The plot is trimmed from 5 s to 1 s

.

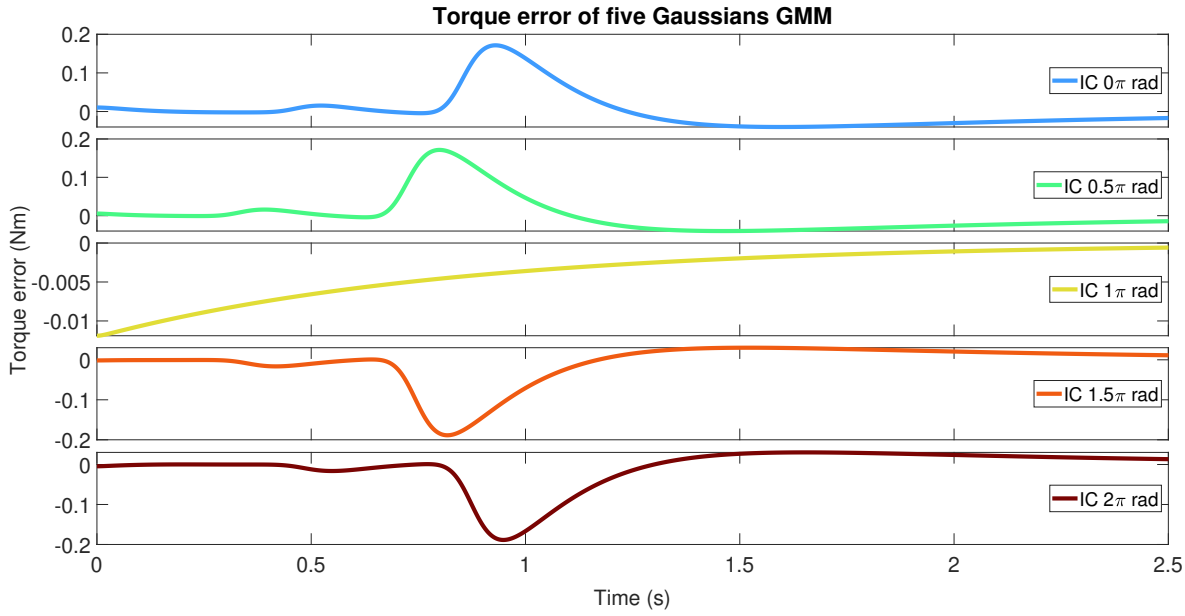
The RMSE between the demonstration and reproduction for the five Gaussians GMM is higher compared to the GMM with one Gaussian (Table 4.1). The initial conditions increases for both systems while for the one Gaussian GMM the RMSE increases, decreases and increases, which is due to the fact that the GMM encodes the data centered around the steady state value. Therefore, reproduction which starts close to the steady state value results in a lower RMSE.

Table 4.1: The RMSE value of the reproduction of P-controller with GMM for one and five Gaussians for the interpolation condition. The P-controller was used as ground truth to calculate the RMSE.

Initial condition [rad]	0	0.5π	π	1.5π	2π
One Gaussian [Nm]	0.0101	0.0050	0.0001	0.0051	0.0102
Five Gaussians [Nm]	0.0347	0.0347	0.0035	0.0365	0.0365



(4.2A)



(4.2B)

Figure 4.2: The torque error [Nm] between the P-controller and GMM for the interpolation initial conditions of the mass angle. Figure 4.2A: the error of the one Gaussian GMM. Figure 4.2B: the error of five Gaussians GMM. The plots are trimmed from 5 s to 1 s.

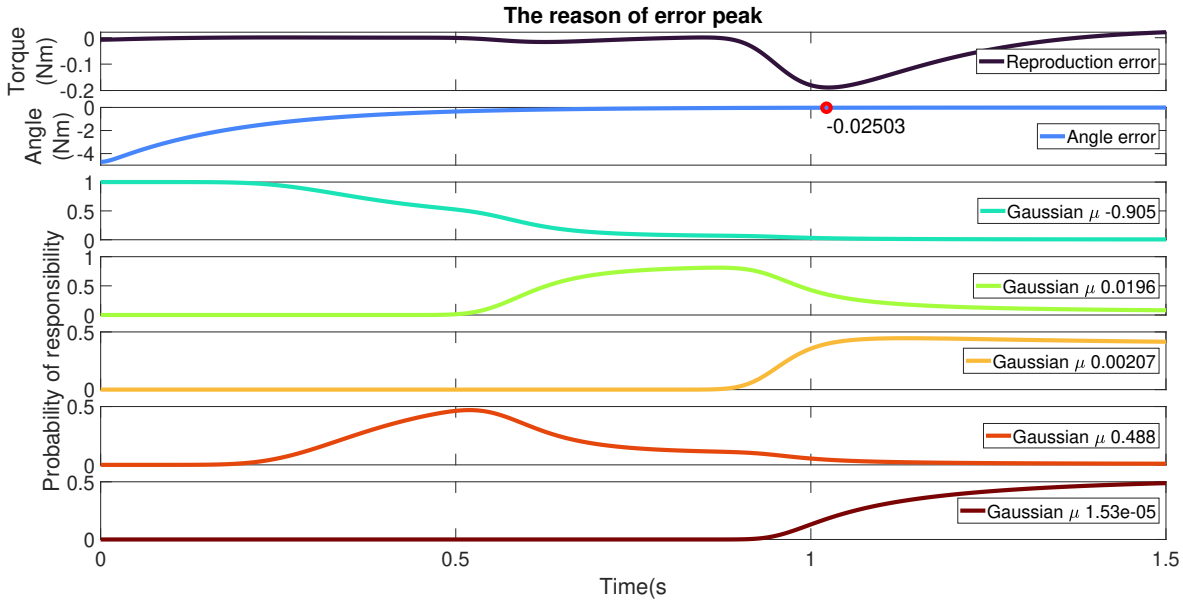
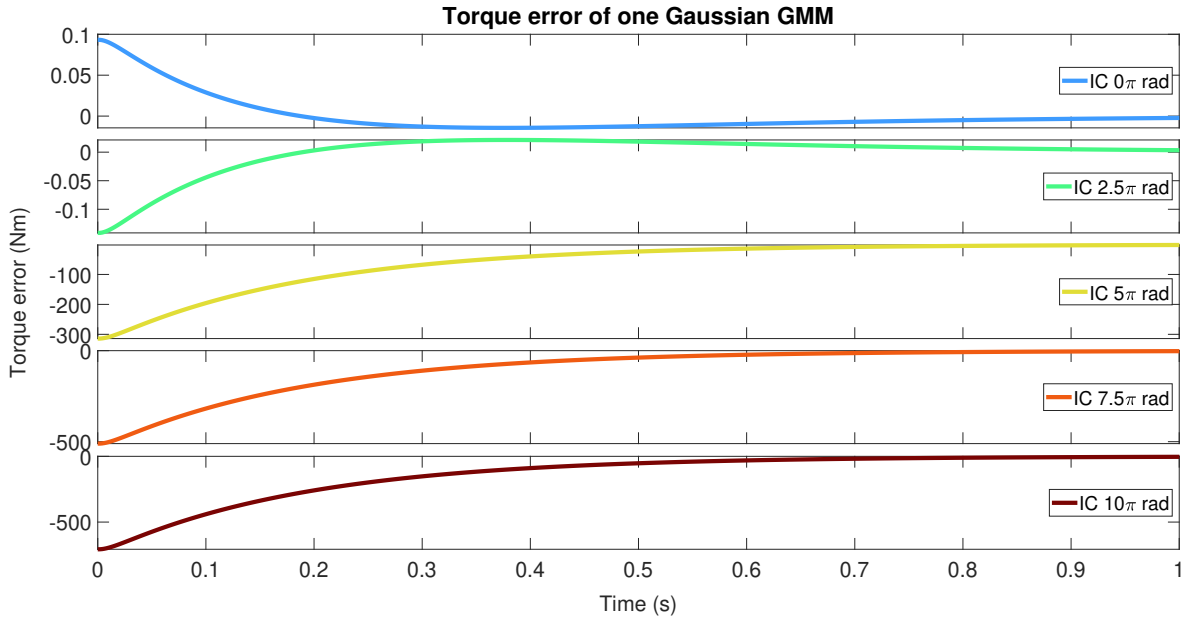


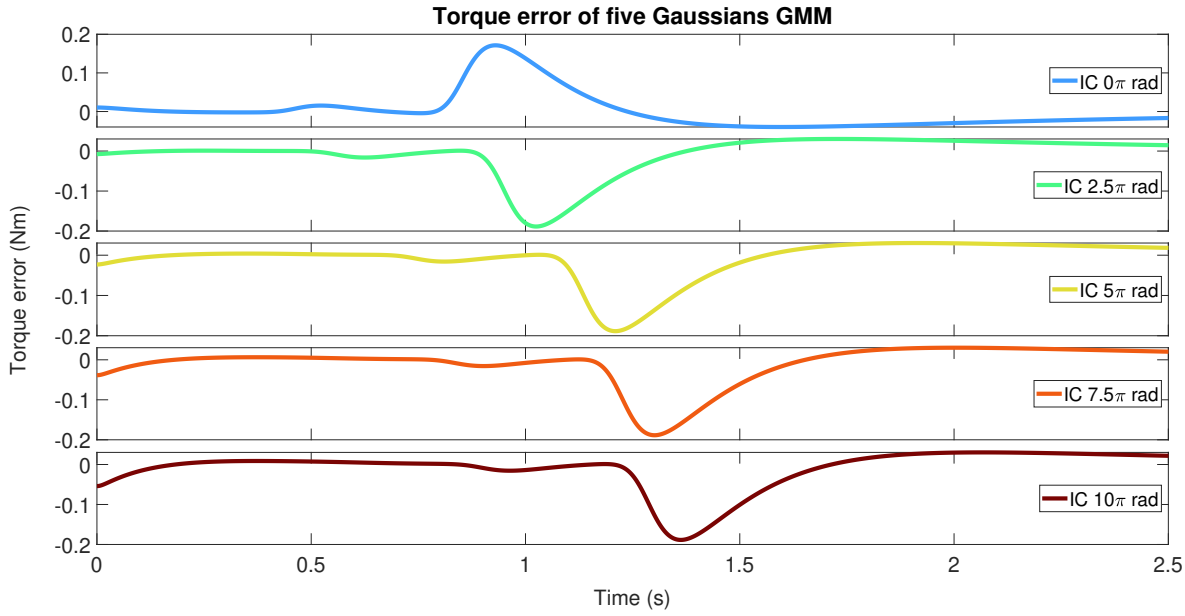
Figure 4.3: The top plot shows the reproduction error of the five Gaussians GMM for the initial condition of 2π rad. The second plot is the angle error (input of the GMM) where the red marker indicates the error value at the time instance of the error peak. The five lowest plots show the normalized responsibility (probability) of the five Gaussians used in the GMM. The corresponding Gaussian means are shown in the legend. The plot has been trimmed from 5 s to 1.5 s.

4.1.1 Extrapolation

The extrapolation capabilities of the P-controller reproduction were evaluated with equidistant initial conditions ranging from 0 rad to 10π rad. For initial condition 7.5π rad the reproduction of one Gaussian does not follow the dynamics of the P-controller (Figure 4.5). The initial condition is in the tail of the responsibility Gaussian. The responsibility shows how well Gaussian j explains the data (w_j in Equation 2.4). Being in the tail of the responsibility Gaussian results in a zero likelihood, which is multiplied with the regression value resulting in a zero torque output (Equation 2.4). The zero torque is also evident in the error of one Gaussian GMM (from third plot onwards in Figure 4.4A). The error of the one Gaussian GMM converges to zero, because the output of the P-controller in steady state equals zero.
The five Gaussian GMM on the contrary does follow the demonstration (see Figure 4.4B). The error between reproduction and demonstration does not exceed 0.2 Nm. The same error peaks as for the interpolation are visible, which is also due to incorrect classification of the Gaussian. The RMSE of the five Gaussians GMM remains constant for the different initial conditions, while the RMSE of the one Gaussian GMM increases over the initial conditions (see Table 4.2).



(4.4A)



(4.4B)

Figure 4.4: The torque error [Nm] between the P-controller and GMM for the extrapolation initial conditions of the mass angle. Figure 4.4A: the error of the one Gaussian GMM. The plot is trimmed from 5 s to 1 s. Figure 4.4B: the error of the five Gaussians GMM. The plot is trimmed from 5 s to 2.5s.

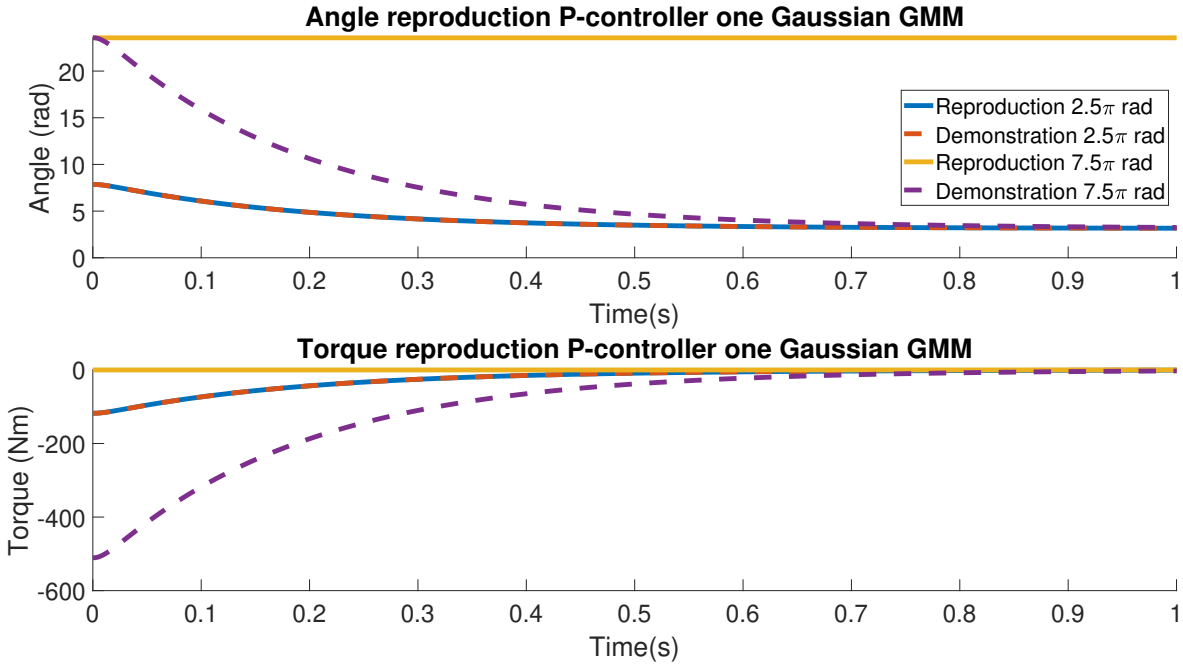


Figure 4.5: The one Gaussian GMM reproduction of a P-controller for an initial condition of the mass at 2.5π rad and 7.5π rad. The dashed line represents the ground truth. The solid line is the reproduction of the one Gaussian GMM. The top graph shows the angle of the mass-damper. The bottom graph the torque of the P-controller, which is the output of the GMM/GMR. The plot is trimmed from 5 s to 1 s.

Table 4.2: The RMSE value of the P-controller reproduction for the extrapolation initial condtions.

Initial condition [rad]	0	2.5π	5π	7.5π	10π
One Gaussian [Nm]	0.0101	0.0153	45.637	74.1595	102.68
Five Gaussians [Nm]	0.0347	0.0365	0.0365	0.0367	0.0369

4.1.2 Time

The reproduction of the GMM with one state and shorter learning time can be found in Figure 4.6. The reproductions are improved compared to the GMM with a normal simulation time system (Figure 4.4A). The shorter time GMM only fails in reproducing for initial condition of 10π rad, which is again a result of the responsibility. The initial condition is in the tail of the responsibility resulting in a zero value. The RMSE is smaller compared to the RMSE of the one Gaussian GMM (see Table 4.3), except for the initial condition of 10π rad. The RMSE is the same because both GMMs are in the extrapolation limit and produce a zero torque.

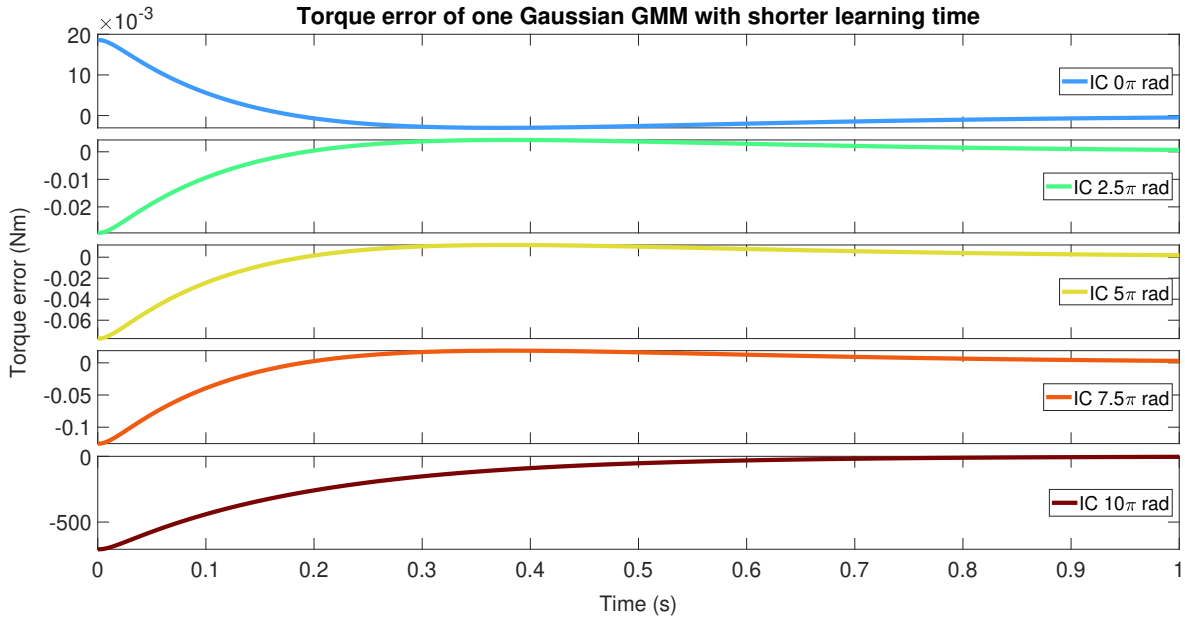


Figure 4.6: The torque error [Nm] between the reproduction of one Gaussian GMM with shorter learning time and the P-controller for the extrapolation initial conditions of the mass angle.

Table 4.3: The RMSE value of the reproduction for the one Gaussian GMM with a short learning time compared with the one Gaussian GMM.

Initial condition [rad]	0	2.5π	5π	7.5π	10π
One Gaussian [Nm] (shorter time)	0.0020	0.0032	0.0084	0.0136	102.68
One Gaussian [Nm] (normal time)	0.0101	0.0153	45.637	74.1595	102.68

4.2 PD - controller

An example of reproduction of the PD-controller with the one Gaussian GMM can be found in Figure 4.7, the one Gaussian GMM does not follow the desired trajectory. The error (see Figure 4.8A) indicates that the one Gaussian GMM is not able to reproduce the PD-controller for the interpolation initial conditions. All of the initial conditions result in the extrapolation limit which was also countered in the extrapolation of the P-controller by the one Gaussian GMM (Subsection 4.1.1). The five Gaussians GMM on the contrary does follow the desired behavior (see Figure 4.8B). The first peak in the error is due to the ideal differentiator, which results in aggressive behavior at the beginning of the simulations (visible at around 0.01 seconds in Figure 4.7). The GMM does not reproduce this aggressive peak, because it gener-

alizes over multiple peaks and combines them in a smooth distribution in which the peak is less prominent.
Similar to the P-controller the RMSE shows (Table 4.4) for both systems the pattern of increasing, decreasing and increasing RMSE. However, the RMSE error for the P-controller is a magnitude smaller.

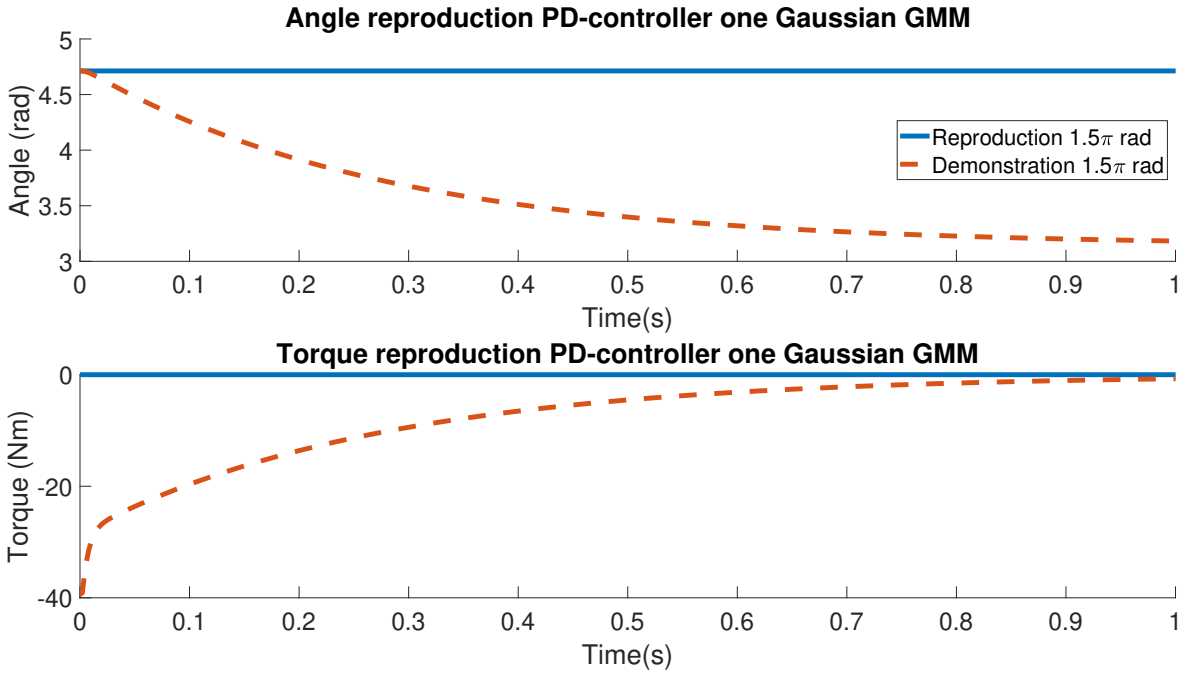
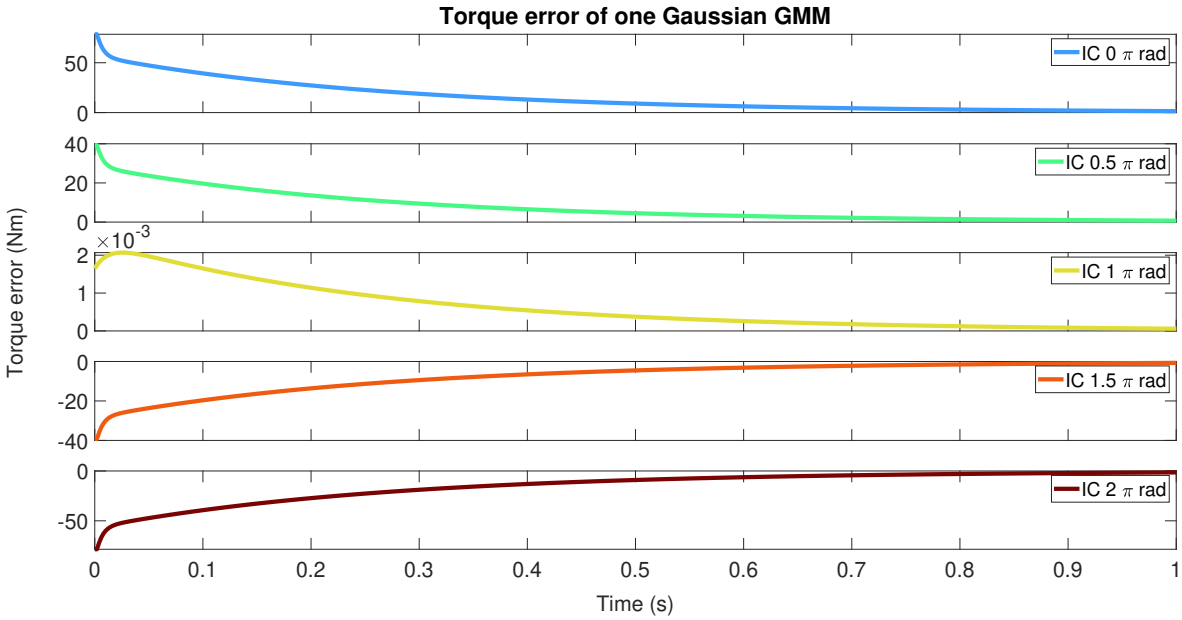


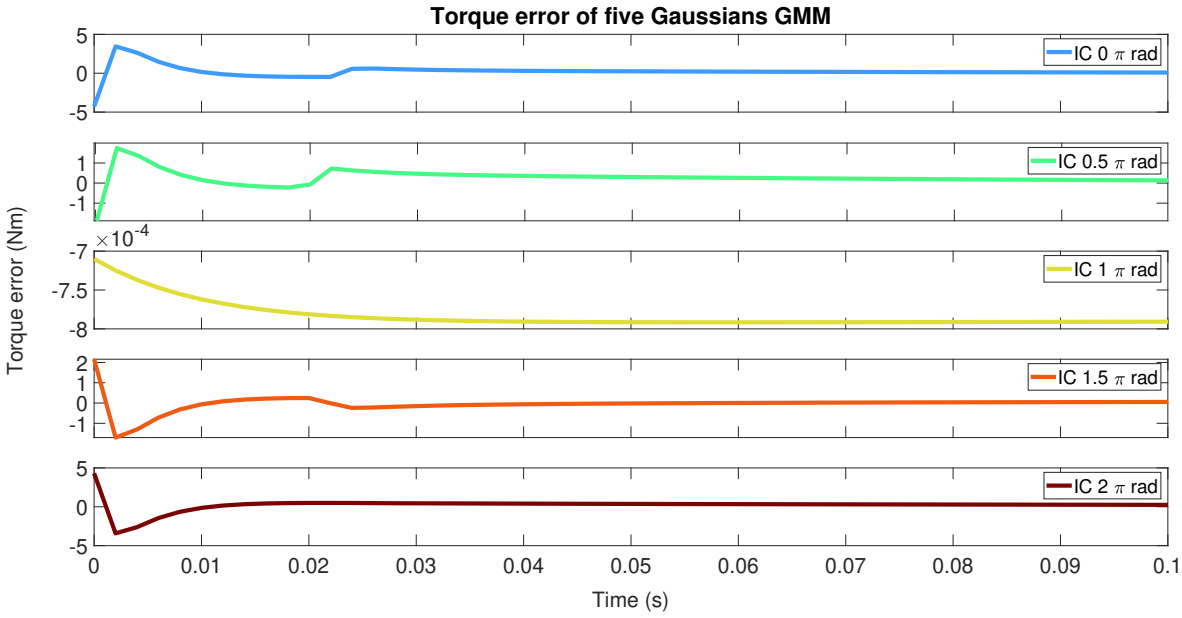
Figure 4.7: An example of the one Gaussian GMM reproducing a PD-controller with mass starting at 1.5π rad. The dashed red line shows the ground truth. The solid blue line (plotted underneath the red line) is the reproduction of the one Gaussian GMM. The top graph shows the angle of the mass-damper. The bottom graph depicts the torque of the PD-controller, which is the output of the GMM/GMR. The plot is trimmed from 5 s to 1 s.

Table 4.4: The RMSE value of PD-controller reproduction for one and five Gaussians with the interpolation initial conditions.

Initial condition [rad]	0	0.5π	π	1.5π	2π
One Gaussian [Nm]	9.6570	4.8290	0.0004	4.8290	9.6570
Five Gaussians [Nm]	0.1401	0.0813	0.0007	0.0704	0.1408



(4.8A)



(4.8B)

Figure 4.8: The torque error [Nm] between the reproduction PD-controller and GMM for the interpolation conditions. Figure 4.8A: the error for the GMM with one Gaussian is plotted. The plot is trimmed from 5 s to 1 s. Figure 4.8B: the error for the GMM with five Gaussians is plotted. The plot is trimmed from 5 s to 0.1s.

4.2.1 Extrapolation

In line with the P-controller simulation, the five Gaussian GMM result in a better reproduction for the extrapolation compared to the one Gaussian GMM. The results of

the one Gaussian GMM remain zero torque for all initial conditions. Therefore, it has been left out (Appendix C.1). On contrary, the five Gaussian GMM cannot reproduce the PD-controller from the initial condition 10π rad (see Figure C.3). The increased complexity of the PD-controller limits the extrapolation. In addition, the RMSE(Table 4.5) increases, because the initial conditions are going further in the tail of the Gaussian resulting in a lower responsibility. In other words, the initial conditions are further away from the known Gaussians and entering a space with less certainty.

Table 4.5: The RMSE value of the reproduction of PD-controller with GMM/GMR for five Gaussians for the extrapolation.

Initial condition [rad]	0	2.5π	5π	7.5π	10π
Five Gaussians [Nm]	0.1401	0.2089	0.5518	0.8955	86.915

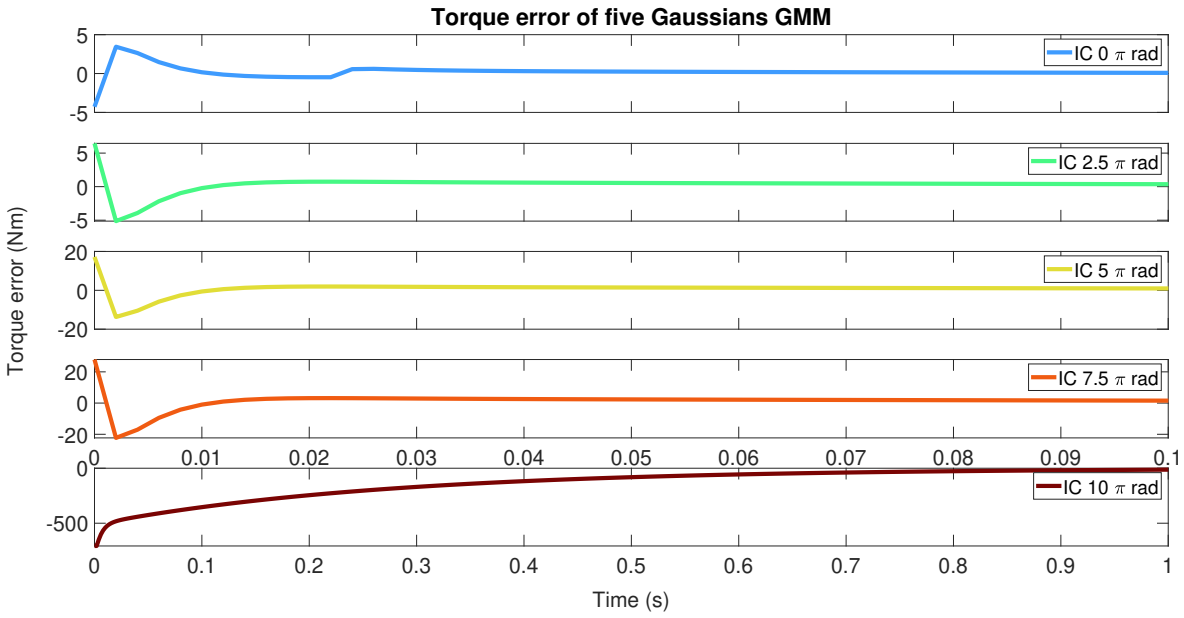


Figure 4.9: The torque error [Nm] between the reproduction of five Gaussians GMM and PD-controller for the extrapolation initial conditions of the mass angle. The time limit for the first until the fourth plot are the same for all the graphs. These have been trimmed from 5 s to 0.1 s. The fifth is trimmed from 5 s to 1 s.

4.2.2 Time

Based on the error in Figure 4.10 a shorter learning time results in a better production compared to the one Gaussian GMM. The steady state does influences the quality of learning for the one Gaussian GMM. This is also evident in the RMSE

(Table 4.6). The shorter learning time RMSE has a similar magnitude as the five Gaussian GMM, which indicates a similar performance. In addition the returning RMSE pattern can be found.

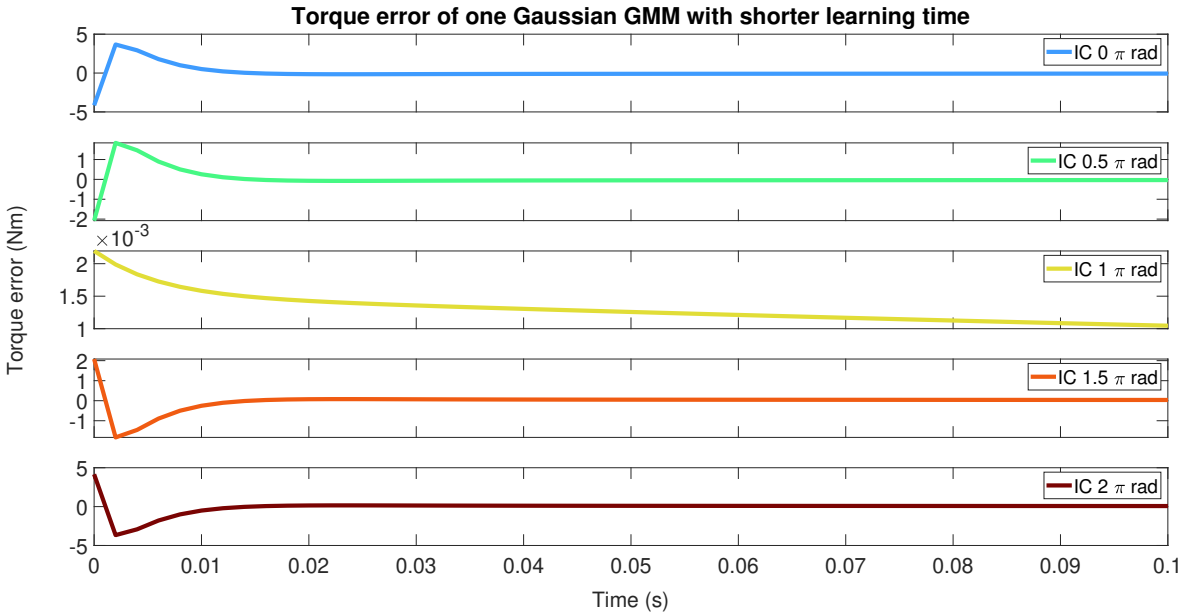


Figure 4.10: The torque error [Nm] between the reproduction PD-controller and the one Gaussian GMM with shorter learning time for the interpolation initial conditions. The plots are trimmed from 1 s to 0.1 s.

Table 4.6: The RMSE value of the PD-controller reproduction for the one and one Gaussians with shorter learning time GMM.

Initial condition [rad]	0	0.5π	1π	1.5π	2π
One Gaussian [Nm] (shorter time)	0.1333	0.0667	0.0003	0.0666	0.1333
One Gaussian [Nm] (normal time)	9.6570	4.8290	0.0004	4.8290	9.6570

4.3 Human PD-controller

Based on the aforementioned results five Gaussians were used to encode the human PD-controller of the valve task. An example of reproduction is seen in Figure 4.11. The angle error increases until a steady state error is reached between the Eve/valve and Omega and thus also between the human and Eve. When the Eve/-valve encounters the end point the angular velocity of the Eve/valve is zero, but the human angle continues rotating (the human-threshold is not implemented).

The torque error between the reproduction and demonstration for the human PD-controller is found Figure 4.12. The error at the beginning of the simulation is the result of the initial condition of the unfiltered PD-controller, which reacts aggressively on the initial error (This is also visible at torque plot in Figure 4.11). The second disturbance at around 2 seconds for 3π rad is the result of an incorrectly classified Gaussian, similar disturbance is evident for the other initial conditions. An example of the disturbance for initial condition π rad is plotted in Figure 4.13, at the time instance of the red marker the responsibility of the fifth Gaussian peaks, while the mean of the third Gaussian is closer towards the error. The responsibility is wrongly classified.

The absolute error between the reproduction and demonstration angle of the Eve is plotted over time for different conditions, see Figure 4.14. The black line represent the maximum tolerance after which the reproduction is not perfect. The reproduction for each initial condition is below this tolerance. Based on this result the GMM/GMR successfully reproduces the human PD-controller. The RMSE value can be found in Appendix C.3.

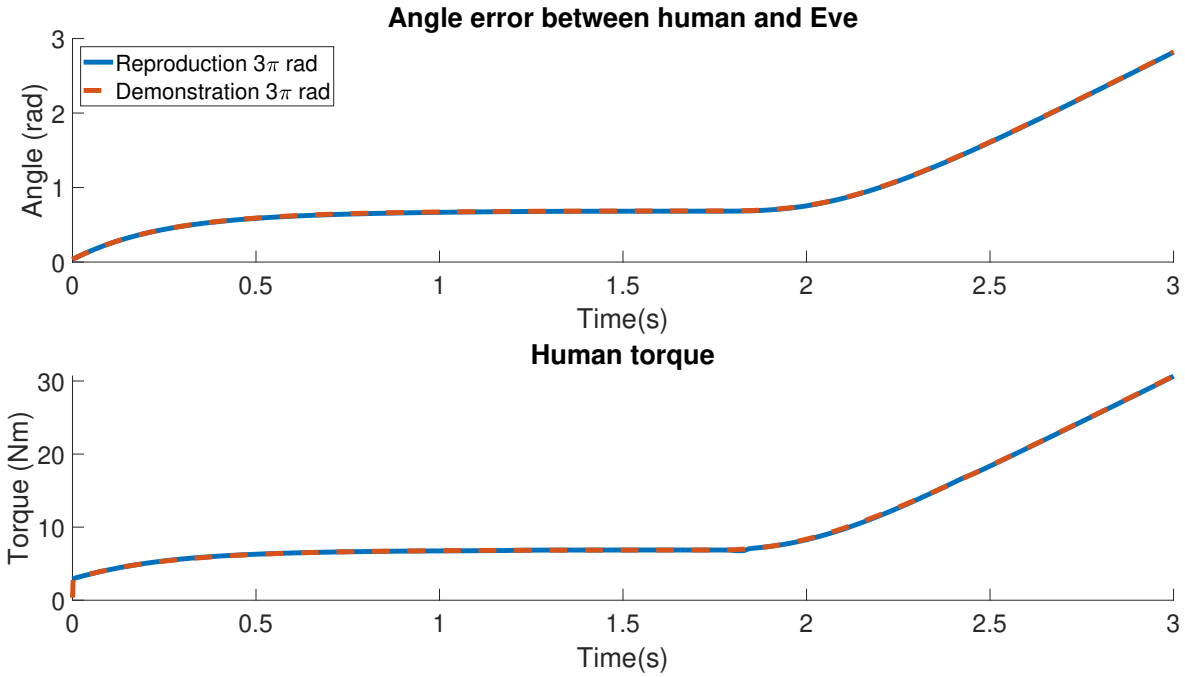


Figure 4.11: An example of GMM reproducing the human PD-controller with mass starting at 3π rad. The dashed red line shows the ground truth. The solid blue line (plotted below the red line) is the reproduction of the one Gaussian GMM. The top graph shows the angle of the Omega. The bottom graph depicts the torque of the human PD-controller, which is the output of the GMM/GMR. The plots are trimmed from 10 s to 3 s.

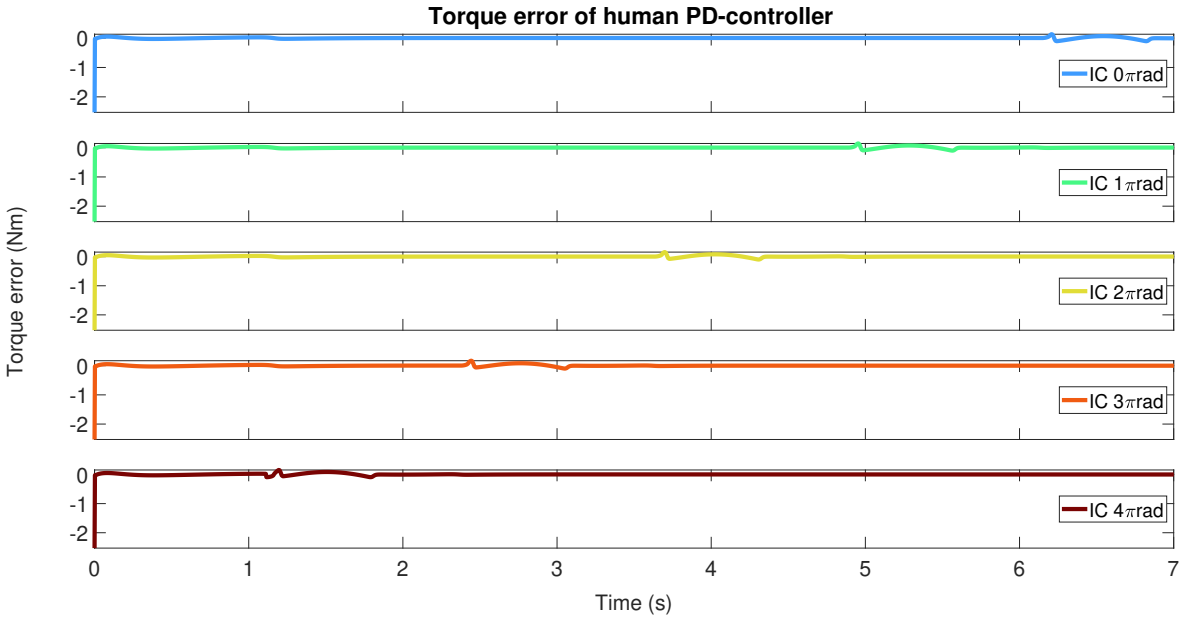


Figure 4.12: The torque error [Nm] between the reproduction of the human PD-controller and the GMM in the valve task. The plots are trimmed from 10 s to 7 s.

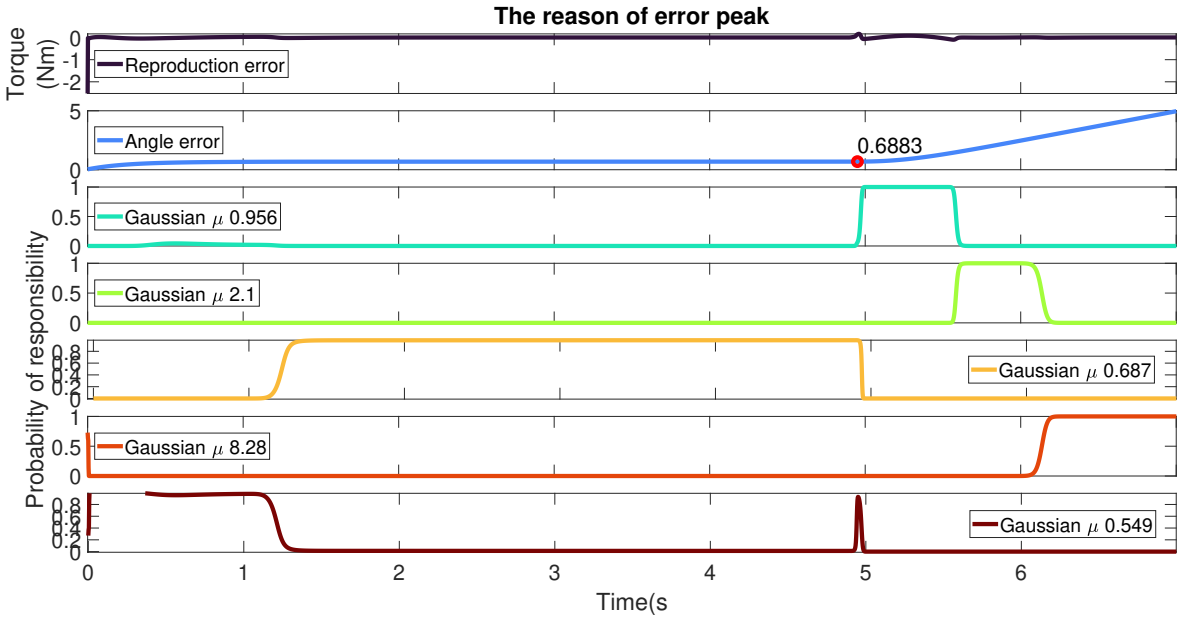


Figure 4.13: The top plot shows the reproduction error of the GMM capturing the human PD-controller for the initial condition of 1π rad. The second plot is the angle error (input of the GMM) where the red marker indicates the error value at the time instance of the error peak. The five lowest plots show the normalized responsibility (probability) of the five Gaussians used in the GMM. The corresponding Gaussian means are shown in the legend. The plot has been trimmed from 10 s to 7 s.

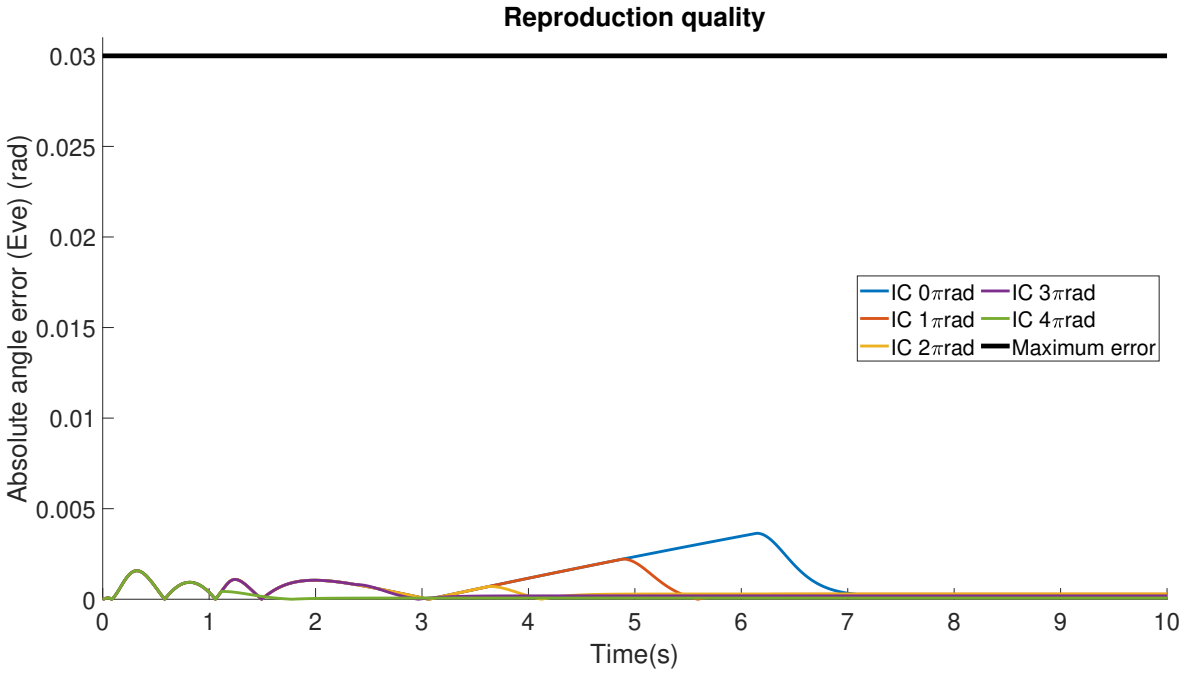


Figure 4.14: The absolute error of the reproduction and demonstration angle of the Eve for different initial conditions of the valve.

4.4 Admittance

Against expectations the GMM does not capture the admittance model, therefore this section contains both inter- and extrapolation results. Figure 4.15 is example of the one Gaussian GMM reproduction with the mass initial condition of 0.5π rad. The reproduction starts at the incorrect angle and at approximately 0.4 seconds the system starts oscillating. Additionally, the reproduced velocity diverges from the demonstrated one. The incorrect reproduction is not an exception, based on the error (Figure 4.16 for interpolation) and RMSE for the interpolation (Table 4.7) and extrapolation (Table 4.8) it can concluded that both one and five Gaussian(s) GMM cannot reproduce an admittance model.T The error plot for extrapolation initial conditions can be found in Appendix C.2.1.

The incorrect reproduction for both output variables is a result of a singularity when mapping the input data to output data. There does not exist an unique input pair for each output value. In other words, for one input pair multiple solutions exist, which is visible in Figure 4.17 . In the left graph of figure an input of 0 $\int \int Velocity$ and 0 $\int \int Torque$ results in an output value ranging from 0 to 2π rad at. It is not possible to reproduce the correct angle/velocity on solely this information.

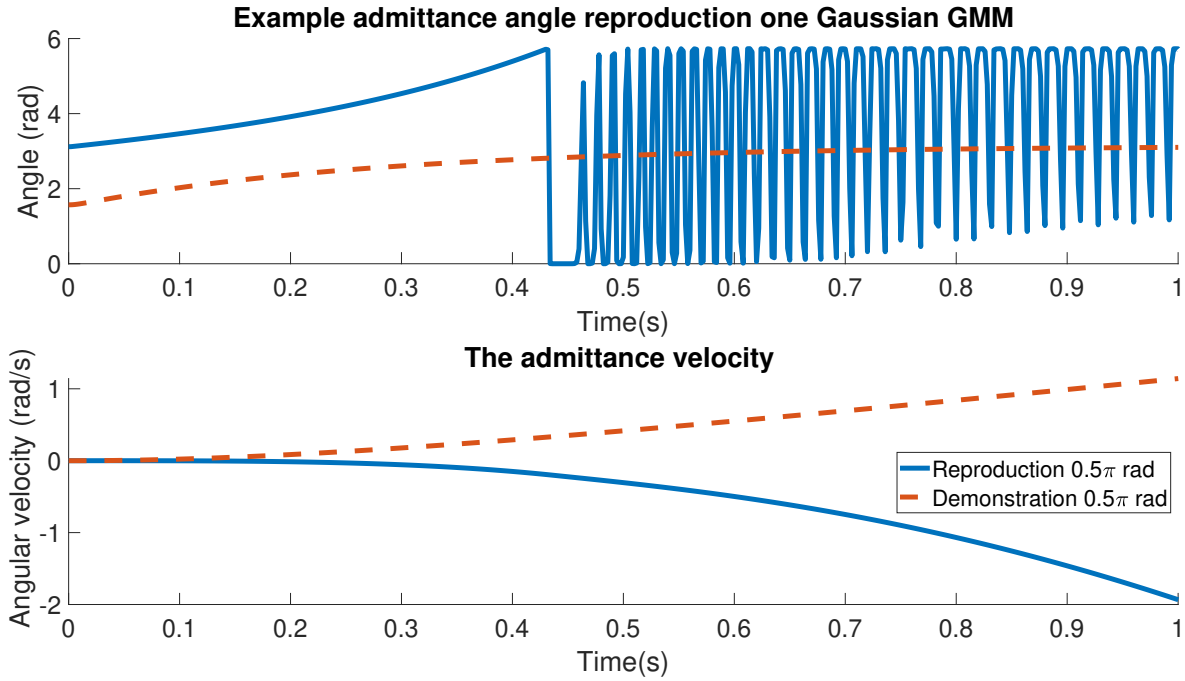


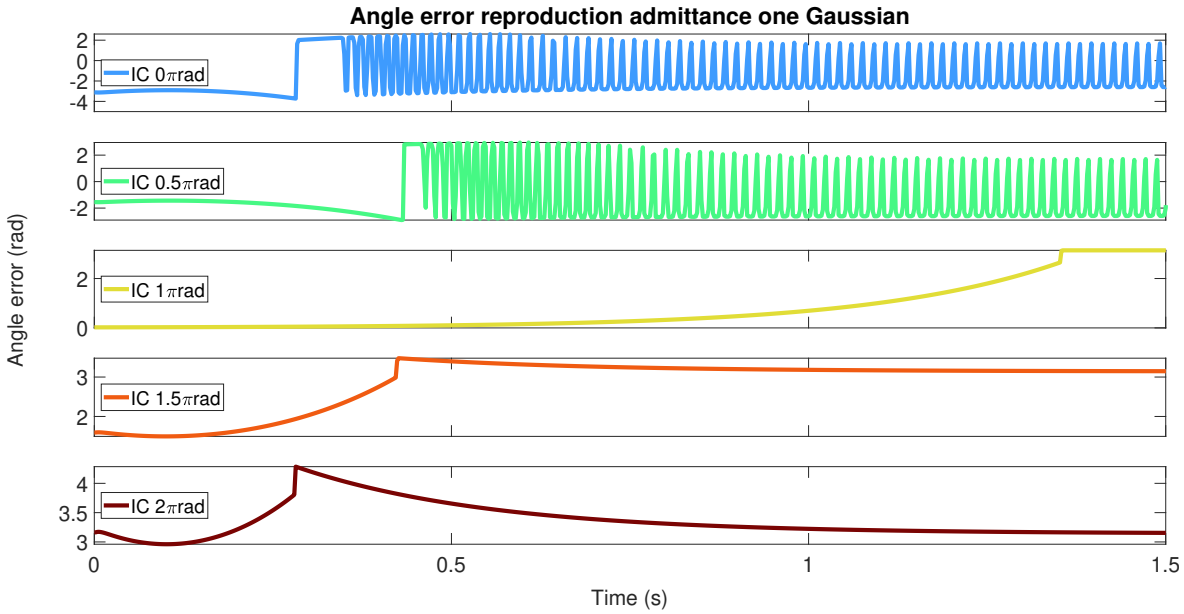
Figure 4.15: An example of initial condition 0.5π rad reproduced by the one Gaussian GMM to test the interpolation capabilities reproducing an admittance in loop with a PD-controller. The dashed red line shows the ground truth. The solid blue line is the reproduction of the one Gaussian GMM. The top graph shows the angle of the admittance. The bottom graph the velocity of the admittance, which is the output of the GMM/GMR. The plot is trimmed from 5 s to 1s.

Table 4.7: The RMSE value of the admittance reproduction for the one and five Gaussians GMM in case of interpolation.

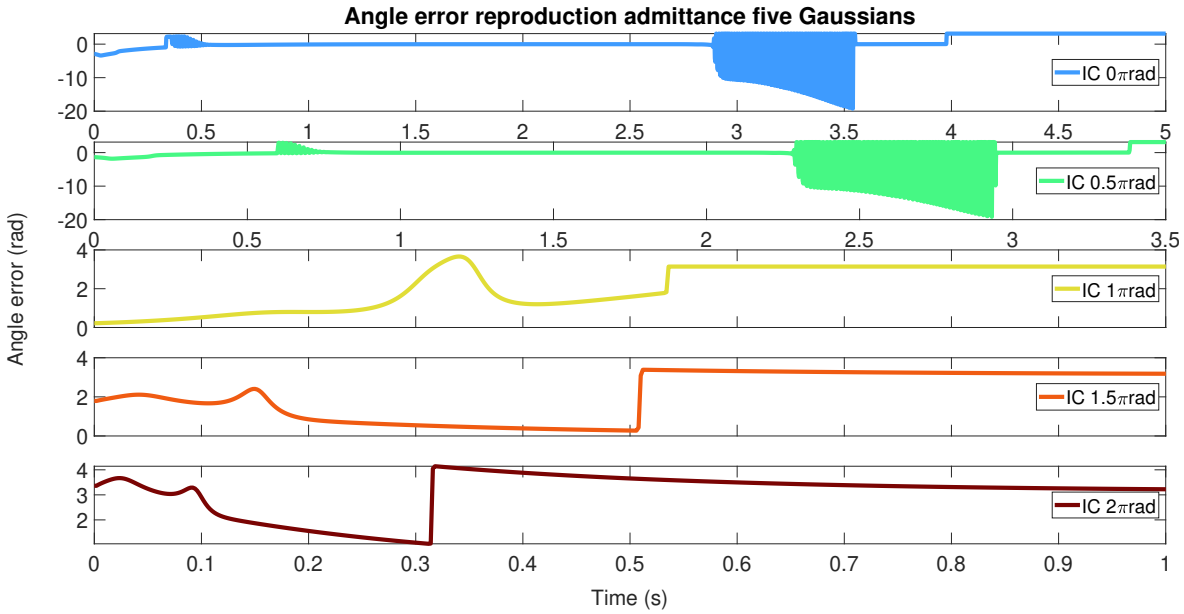
Initial condition [rad]	0	0.5π	1π	2π	3π
one Gaussian [rad]	2.2203	2.1240	2.7173	3.0764	3.2111
five Gaussians [rad]	3.4602	3.5786	3.0094	3.0166	3.1556

Table 4.8: The RMSE value of the admittance reproduction for the one and five Gaussians GMM in case of extrapolation.

Initial condition [rad]	0	2.5π	5π	7.5π	10π
one Gaussian [rad]	2.2203	3.3516	4.2267	5.2954	6.4627
five Gaussians [rad]	3.4602	3.2987	4.1839	5.2624	6.4344



(4.16A)



(4.16B)

Figure 4.16: The angle error [rad] between the reproduction and admittance system for the interpolation initial conditions of the mass angle. Figure 4.16A: the error of the one Gaussian GMM. The plot is trimmed from 5 s to 1 s. Figure 4.16B: the error of five Gaussians GMM. The plot is trimmed from 5 s to 3.5s for the second panel counted from the top. The last three panels are trimmed from 5 s to 1 s.

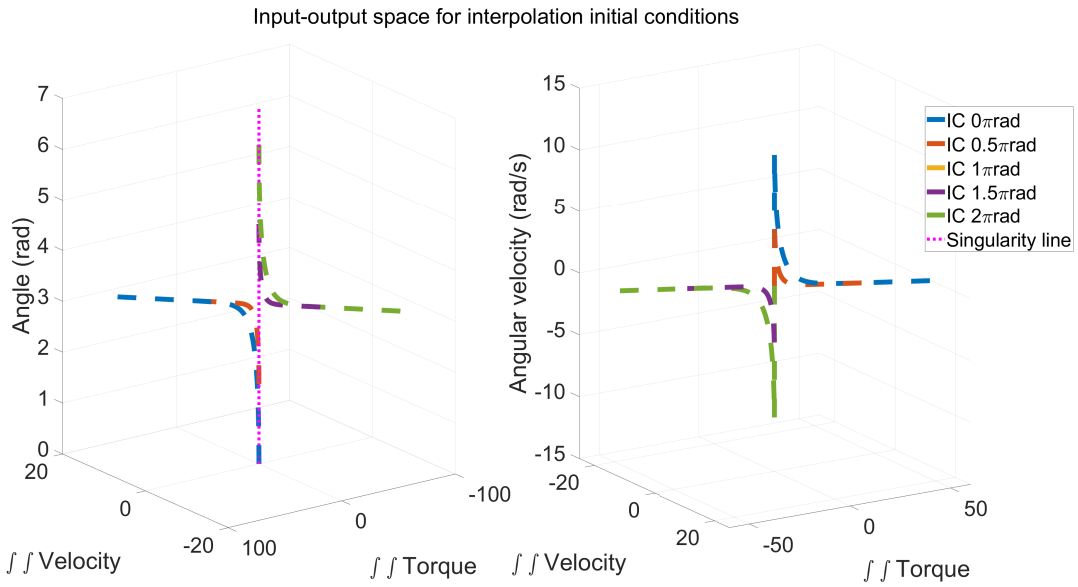


Figure 4.17: The input-output space of the GMM plotted with the interpolation initial conditions. It consists of the angle (left plot) and velocity (right plot) as output and the double integral of the torque and velocity are inputs. The singularity line shows that along that line a singularity exists, which is less evident for the angle plot compared to the velocity plot.

4.5 Omega

The last simulations performed were the reproduction of the Omega. Five states were used to learn the admittance mode, but the reproduction are in line with the reproduction of the admittance system in loop with a PD-controller, which is confirmed by the RMSE in Table 4.9. An example of reproduction can be found in Figure 4.18, where the system does follow the demonstration for the first 4.5 s after which it starts oscillating. However, as visible in Figure 4.19 only the initial condition 1π and 2π rad show a relative small error at the beginning. This is a consequence of the GMM encoding, which can be seen on the right of Figure 4.20. The first Gaussian follows approximately the slope of the trajectory of 1π and 2π rad and therefore the resulting reproduction also follows this slope yielding in a relative small error.
The incorrect reproduction are also the result of the singularity similar to the simulation results of the admittance system in loop with a PD-controller. However, the singularity is also visible in the right panel of Figure 4.20. The inputs have a linear relations, which means in this case that for one pair of input multiple output values are possible (visible in the left panel of Figure 4.20).

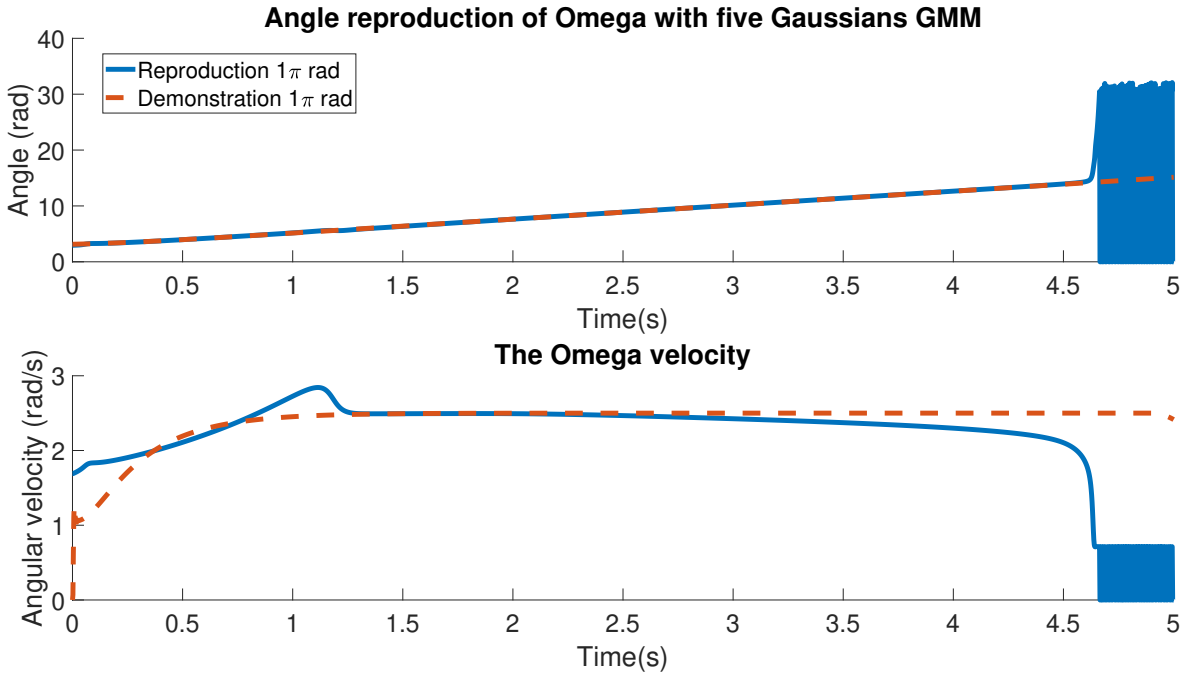


Figure 4.18: An example of one initial condition π rad reproduced by the five Gaussian GMM to reproduce the Omega. The dashed red line shows the ground truth. The solid blue line is the reproduction of the one Gaussian GMM. The top graph shows the angle of the admittance. The bottom graph the velocity of the admittance, which is the output of the GMM/GMR. The plot is trimmed from 10 s to 5s

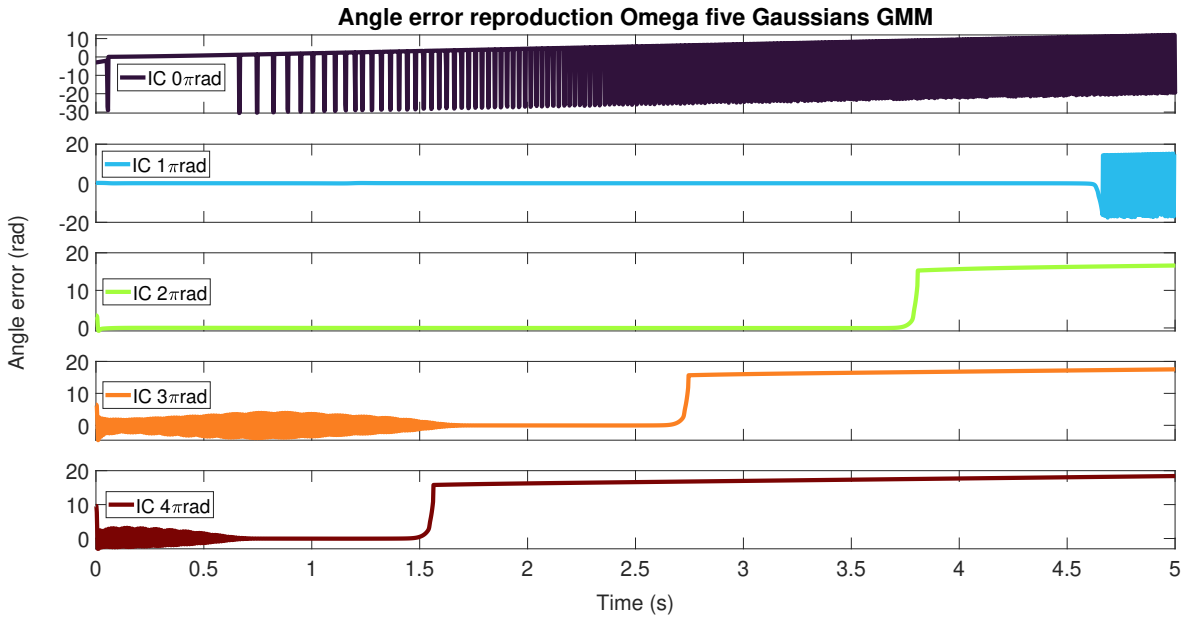


Figure 4.19: The angle error [rad] between the reproduction of five Gaussians GMM and the Omega. The plot is trimmed from 10 s to 5 s.

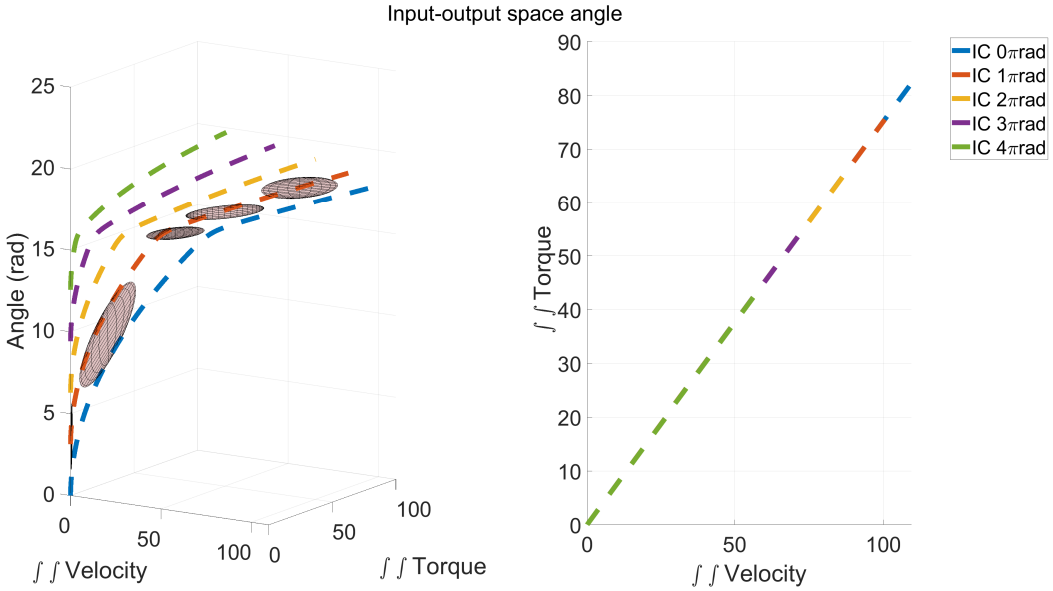


Figure 4.20: The input-output space of the angle for the admittance model. The left panel shows the trajectory of demonstrations in this space. The meshed ellipses are the Gaussians ellipse in three dimensions, it shows the variances of the Gaussians. The right panel shows the linear relationship between the input variables.

Table 4.9: The RMSE value of the Omega reproduction.

Initial condition [rad]	0	0.5π	1π	1.5π	2π
GMM [rad]	13.447	11.159	14.173	15.829	17.502

Chapter 5

Discussion

The purpose of the research was to investigate whether a state-of-the-art imitation learning framework can capture a (human) controller closing a valve. Based on a theoretical comparison, GMM/GMR was deemed to be most suitable for capturing the human controller. The human controller was split in sub controllers, which allowed to obtain insight on the limitations of GMM/GMR. These results indicate the ability of GMM/GMR to reproduce and generalize to a limited extent a system in which there exists a linear relation between the input and output variable without a singularity. A singularity arises when an input corresponds to multiple output values. Limitations include difficulties in selection of hyperparameters and features, and limited inter- and extrapolation capabilities. These limitations indicate that, while GMM can model linear systems, they are not suitable to capture human controllers. It imposes the restriction on the system to have a linear controller without singular input output variables. However, if the linearity of the controller is already known,system identification might be a better solution instead of GMM, a correctly identified controller/system does not impose limitations on the extrapolation capability.
Additionally, these restrictions imply that GMM are more suitable to capture motions instead of the underlying mechanism. It is a method which allows you to learn one type of task with limited spatial variance without prior knowledge of the task [18], [43], [44], which comes at the cost of limited generalizability. The generalizability of the GMM/GMR could be expanded by combing with another encoding method [44] and/or task parameters [45]. These expansions however solely capture the motion of a specific task and not the underlying mechanism. Inverse Reinforcement Learning (IRL) for LfD on the contrary shows, despite the demand for large number of demonstrations, more potential in generalization [46] and thus in capturing a human controller. The generalization capabilities are a result of extracting an environment independent reward function [46]–[48]. However, a drawback is that there might exist multiple reward functions for one task, which makes it an ill-posed problem. This makes it difficult or even impossible to find the correct solution [47], because the

same reward function might be optimized by multiple policies [48]. In order to deduce the reward function, large amount of data is required [14]. Additionally, if the environment changes, a new policy has to be learned based on the found reward function [47]. Lastly, the method remains a black box method, which limits the adaptability of the reward function. It might be unclear how to adapt the reward function without completely relearning to different tasks.
Even though IRL might have the potential to capture the underlying mechanisms of a human controller via environment independent reward functions, it suffers from limitations such as requiring a large number of demonstrations, the inability to always find the correct reward function and the requirement to relearn a policy when the environment changes.
A different potential framework for capturing a controller is combining Lfd with machine learning. For example, *Hadar et al.* [49] eliminates the requirement of large number of demonstrations by combining LfD with reinforcement learning. This requires at most twenty minutes of interactive learning after which it can generalize to new object configurations. It is a promising direction. However, it uses a visual framework and it is also a black box method. Additionally, the goal of the paper was not to model a human controller, but to find a policy to execute the desired tasks.
As mentioned before, the GMM is limited in reproducing controllers. This problem is most evident in the incorrect reproduction of the admittance system, which is a result of the singularity. Additionally, the self-dependency of the GMM is also a potential cause for incorrect reproductions. The input of the GMM depends on the double integral of the velocity, which is an output of the GMM. The output will be integrated twice and used as input in the next sample, which is thus also an incorrect input. The incorrect input results in an incorrect the output. It is a circular error. Thus, the admittance model cannot be reproduced for this input output pair.
A different selection of input might solve this problem, but it does not guarantee a successful reproduction. A solution to the feature selection problem could be by using the correlation variables together with the physics of the system [50]. If two input variables are highly correlated one of them might be redundant for learning the correspondent task.
The input output selection is not the only factor that determines the quality of reproduction. The (spatial) spread of the demonstration data influences the range of the responsibility. Based on the P- and PD-controller reproduction, the responsibility seems to be a narrow tailed Gaussian. It leads a zero value reproduction, because the likelihood that the GMM explains the data is zero. The spread of the demonstration data determines the variance of the responsibility and it therefore influences the generalizability. This is evident while taking into account the demonstration time when learning the P(D)-controller. Decreasing the demonstration time resulted in a

smaller contribution of the steady state on the mean and variance, which resulted in a better reproduction compared to the original length. The variance widens the Gaussian, resulting in a larger range of possible encoding.
Lastly, the desired reproduction depends on choosing an adequate number of components. The quality of the reproduction and capabilities of a GMM are depends on the hyperparameters selected within the GMM. Especially the number of components influences the quality of the reproduction, which is evident in the P- and PD-control cases. The results might suggest that more Gaussian components would result in a better reproduction. However, it might lead to overfitting of the data, which results in poor generalization. Therefore, determining the optimal number of components remains a key problem in LfD. Future studies could further optimise this procedure by relying on methods based on an Bayesian information criterion (BIC) which success has been demonstrated [51]–[53]. However, the BIC only selects the parameters that are optimal for the GMM, but not for regression [54]. Therefore, a combination of information criterion and comparing RMSE ([55]) was recommended for future research.

Chapter 6

Conclusion

Although the proposed LfD framework was not capable of reproducing the valve task, this study shed more light on the limitations of GMM/GMR in capturing a human controller. In conclusion, a GMM/GMR showed potential to capture human controller. GMM/GMR was capable of learning and reproducing a P(D)-controller, but with limited generalization capabilities. These limitations are a result of the responsibility used in the GMR and the spread of the data. Additionally, the reproduction of an admittance system was unsuccessful. The GMM performance was limited by a singularity in input-output variable relation. Furthermore, if it is already known that a system can be controlled with a linear controller it might be more beneficial to use system identification in contrast to using GMM. It lead to the conclusion that the GMM are not suitable to capture a human controller. The method is better suited to capture a motion over time with limited spatial variance in which there is no prior information about the motion/system. A promising field for capturing a human controller could be reinforcement learning, but these introduces other limitations such as the requirement for a large set of demonstrations and less intuitive understanding of the control method.

Bibliography

[1] "Global strategy on human resources for health: Workforce 2030."

[2] M. L. Ranney, V. Griffeth, and A. K. Jha, "Critical Supply Shortages — The Need for Ventilators and Personal Protective Equipment during the Covid-19 Pandemic," *New England Journal of Medicine*, vol. 382, no. 18, p. e41, 4 2020. [Online]. Available: https://www.nejm.org/doi/full/10.1056/NEJMp2006141

[3] The Lancet, "COVID-19: protecting health-care workers," *The Lancet*, vol. 395, no. 10228, p. 922, 3 2020. [Online]. Available: http://www.thelancet.com/article/S0140673620306449/fulltexthttp://www.thelancet.com/article/S0140673620306449/abstracthttps://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)30644-9/abstract

[4] J. B. Herron, A. G. Hay-David, A. D. Gilliam, and P. A. Brennan, "Personal protective equipment and Covid 19- a risk to healthcare staff?" *British Journal of Oral and Maxillofacial Surgery*, vol. 58, no. 5, pp. 500–502, 6 2020. [Online]. Available: http://www.bjoms.com/article/S0266435620301650/fulltexthttp://www.bjoms.com/article/S0266435620301650/abstracthttps://www.bjoms.com/article/S0266-4356(20)30165-0/abstract

[5] G. Yang, H. Lv, Z. Zhang, L. Yang, J. Deng, S. You, J. Du, and H. Yang, "Keep Healthcare Workers Safe: Application of Teleoperated Robot in Isolation Ward for COVID-19 Prevention and Control," *Chinese Journal of Mechanical Engineering (English Edition)*, vol. 33, no. 1, pp. 1–4, 12 2020. [Online]. Available: https://cjme.springeropen.com/articles/10.1186/s10033-020-00464-0

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, pp. 469–483, 2009. [Online]. Available: www.elsevier.com/locate/robot

[7] A. D. Sosa-Ceron, H. G. Gonzalez-Hernandez, and J. A. Reyes-Avendaño, "Learning from Demonstrations in Human–Robot Collaborative Scenarios: A Survey," *Robotics 2022, Vol. 11, Page 126*, vol. 11, no. 6, p. 126, 11

2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/6/126/htmhttps://www.mdpi.com/2218-6581/11/6/126

[8] M. Arduengo, A. Arduengo, A. Colomé, J. Lobo-Prat, and C. Torras, "Human to Robot Whole-Body Motion Transfer," 9 2019. [Online]. Available: http://arxiv.org/abs/1909.06278

[9] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from Humans," *Springer Handbook of Robotics*, pp. 1995–2014, 1 2016. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-32552-1_74

[10] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input," *https://doi.org/10.1163/016918611X558261*, vol. 25, no. 5, pp. 581–603, 3 2012. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1163/016918611X558261

[11] C. Zeng, C. Yang, J. Zhong, and J. Zhang, "Encoding Multiple Sensor Data for Robotic Learning Skills from Multimodal Demonstration," *IEEE Access*, vol. 7, pp. 145 604–145 613, 2019.

[12] Y. Lin, S. Ren, M. Clevenger, and Y. Sun, "Learning grasping force from demonstration," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1526–1531, 2012.

[13] N. Wang, C. Chen, and A. D. Nuovo, "A Framework of Hybrid Force/Motion Skills Learning for Robots," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 1, pp. 162–170, 3 2021.

[14] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[15] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *https://doi.org/10.1146/annurev-control-100819-063206*, vol. 3, no. 1, pp. 297–330, 5 2020. [Online]. Available: https://www.annualreviews.org/doi/abs/10.1146/annurev-control-100819-063206

[16] Y. Liu, Z. Li, H. Liu, and Z. Kan, "Skill transfer learning for autonomous robots and human-robot cooperation: A survey ," *Robotics and Autonomous Systems*, vol. 128, p. 103515, 2020. [Online]. Available: https://doi.org/10.1016/j.robot.2020.103515

[17] A. Paraschos, E. Rueckert, J. Peters, and G. Neumann, “Model-free Probabilistic Movement Primitives for physical interaction,” *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 2860–2866, 12 2015.

[18] S. Calinon, F. Guenter, and A. Billard, “On learning the statistical representation of a task and generalizing it to various contexts,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 2978–2983, 2006.

[19] L. Rozo, P. Jiménez, and C. Torras, “A robot learning from demonstration framework to perform force-based manipulation tasks,” *Intelligent Service Robotics 2013 6:1*, vol. 6, no. 1, pp. 33–51, 1 2013. [Online]. Available: https://link.springer.com/article/10.1007/s11370-012-0128-9

[20] A. Pervez, A. Ali, J. H. Ryu, and D. Lee, “Novel learning from demonstration approach for repetitive teleoperation tasks,” *2017 IEEE World Haptics Conference, WHC 2017*, pp. 60–65, 7 2017.

[21] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, “Non-parametric Imitation Learning of Robot Motor Skills,” 2019.

[22] S. Calinon and D. Lee, “Learning Control.” [Online]. Available: https://doi.org/10.1007/978-94-007-7194-9_68-1

[23] Y. Guan, N. Wang, and C. Yang, “Review of the techniques used in motor-cognitive human-robot skill transfer,” *Cognitive Computation and Systems*, vol. 3, no. 3, pp. 229–252, 9 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1049/ccs2.12025https://onlinelibrary.wiley.com/doi/abs/10.1049/ccs2.12025https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ccs2.12025

[24] S. Calinon, F. D’Halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, “Learning and reproduction of gestures by imitation,” *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 6 2010.

[25] N. Jaquier, D. Ginsbourger, and S. Calinon, “Learning from demonstration with model-based Gaussian process,” 10 2019. [Online]. Available: https://arxiv.org/abs/1910.05005v1

[26] S. Calinon, F. D’Halluin, D. G. Caldwell, and A. G. Billard, “Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework,” *9th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS09*, pp. 582–588, 2009.

[27] A. Paraschos, E. Rueckert, J. Peters, and G. Neumann, "Probabilistic movement primitives under unknown system dynamics," *https://doi-org.ezproxy2.utwente.nl/10.1080/01691864.2018.1437674*, vol. 32, no. 6, pp. 297–310, 3 2018. [Online]. Available: https://www-tandfonline-com.ezproxy2.utwente.nl/doi/abs/10.1080/01691864.2018.1437674

[28] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic Movement Primitives," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[29] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 3 2018. [Online]. Available: https://link.springer.com/article/10.1007/s10514-017-9648-7

[30] S. de Zwart, "Impact-Aware Learning from Demonstration," 2019. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Ac6f91fb2-2544-4802-bcda-4ee70ab0e2be

[31] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks," *Autonomous Robots*, vol. 41, no. 3, pp. 593–612, 3 2017. [Online]. Available: https://link.springer.com/article/10.1007/s10514-016-9556-2

[32] S. Gomez-Gonzalez, G. Neumann, B. Scholkopf, and J. Peters, "Adaptation and Robust Learning of Probabilistic Movement Primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 4 2020.

[33] P. L. Hera, D. O. Morales, and O. Mendoza-Trejo, "A study case of Dynamic Motion Primitives as a motion planning method to automate the work of forestry cranes," *Computers and Electronics in Agriculture*, vol. 183, p. 106037, 4 2021.

[34] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/23148415/

[35] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic Movement Primitives in Robotics: A Tutorial Survey," *undefined*, 2021. [Online]. Available: www.sagepub.com/

[36] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A DMPs-Based Framework for Robot Learning and Generalization of Humanlike Variable Impedance Skills," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1193–1203, 6 2018.

[37] C. Li, A. Fahmy, S. Li, and J. Sienz, "An Enhanced Robot Massage System in Smart Homes Using Force Sensing and a Dynamic Movement Primitive," *Frontiers in Neurorobotics*, vol. 14, p. 30, 6 2020.

[38] M. Franken, S. Stramigioli, S. Misra, C. Secchi, and A. MacChelli, "Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 741–756, aug 2011.

[39] Z. Zhu and H. Hu, "Robot Learning from Demonstration in Robotic Assembly: A Survey," *Robotics 2018, Vol. 7, Page 17*, vol. 7, no. 2, p. 17, 4 2018. [Online]. Available: https://www.mdpi.com/2218-6581/7/2/17/htmhttps://www.mdpi.com/2218-6581/7/2/17

[40] L. Rozo, P. Jiménez, and C. Torras, "Force-based robot learning of pouring skills using parametric hidden Markov models," *9th International Workshop on Robot Motion and Control, RoMoCo 2013 - Workshop Proceedings*, pp. 227–232, 2013.

[41] S. Calinon, "Mixture Models for the Analysis, Edition, and Synthesis of Continuous Time Series," pp. 39–57, 4 2021. [Online]. Available: http://arxiv.org/abs/2104.10731http://dx.doi.org/10.1007/978-3-030-23876-6_3

[42] S. Calinon, E. L. Sauser, A. G. Billard, and D. G. Caldwell, "Evaluation of a probabilistic approach to learn and reproduce gestures by imitation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2671–2676, 2010.

[43] N. Ettehadi and A. Behal, "Implementation of feeding task via learning from demonstration," *Proceedings - 2nd IEEE International Conference on Robotic Computing, IRC 2018*, vol. 2018-Janua, pp. 274–277, 4 2018.

[44] B. Ti, Y. Gao, Q. Li, and J. Zhao, "Dynamic movement primitives for movement generation using gmm-gmr analytical method," *2019 IEEE 2nd International Conference on Information and Computer Technologies, ICICT 2019*, pp. 250–254, 5 2019.

[45] S. Calinon, T. Alizadeh, and D. G. Caldwell, “On improving the extrapolation capability of task-parameterized movement models,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 610–616, 2013.

[46] J. Hua, L. Zeng, G. Li, and Z. Ju, “Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning,” *Sensors 2021, Vol. 21, Page 1278*, vol. 21, no. 4, p. 1278, 2 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1278/htmhttps://www.mdpi.com/1424-8220/21/4/1278

[47] S. Adams, T. Cody, and P. A. Beling, “A survey of inverse reinforcement learning,” *Artificial Intelligence Review*, vol. 55, no. 6, pp. 4307–4346, 8 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10462-021-10108-x

[48] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, “Imitation Learning,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, 4 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3054912

[49] S. Haldar, J. Pari, A. Rai, and L. Pinto, “Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations,” 3 2023. [Online]. Available: https://arxiv.org/abs/2303.01497v1

[50] L. Rozo, P. Jiménez, and C. Torras, “Robot Learning from Demonstration in the Force Domain,” 2011.

[51] L. Rozo, S. Calinon, D. G. Caldwell, P. Jiménez, and C. Torras, “Learning Physical Collaborative Robot Behaviors From Human Demonstrations,” *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 513–527, 6 2016.

[52] J. G. Eljaik, R. Lober, A. Hoarau, and V. Padois, “Optimization-based controllers for robotics applications (ocra): The case of icub’s whole-body control,” *Frontiers in Robotics and AI*, vol. 5, p. 24, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2018.00024

[53] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, “Robot Learning System Based on Adaptive Neural Control and Dynamic Movement Primitives,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 777–787, 3 2019.

[54] Y. Q. Wang, Y. D. Hu, S. E. Zaatari, W. D. Li, and Y. Zhou, “Optimised Learning from Demonstrations for Collaborative Robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102169, 10 2021.

[55] C. J. Perez-Del-Pulgar, J. Smisek, V. F. Munoz, and A. Schiele, “Using learning from demonstration to generate real-time guidance for haptic shared control,” *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pp. 3205–3210, 2 2017.

[56] H. G. Sung, “Gaussian mixture regression and classification,” Ph.D. dissertation, Rice University, Houston, Texas, 5 2004. [Online]. Available: https://scholarship.rice.edu/handle/1911/18710

[57] K. Mardia, J. Kent, and J. Bibby, *Multivariate Analysis*, 1st ed. London: Academic Press, 12 1979.

[58] C. M. Bishop, “Pattern recognition and machine learning,” 2013. [Online]. Available: https://books.google.com/books/about/Pattern_Recognition_and_Machine_Learning.html?id=HL4HrgEACAAJ

[59] S. Calinon, “pbdlib-matlab,” 2023. [Online]. Available: https://academia.stackexchange.com/questions/14010/how-do-you-cite-a-github-repository

Appendix A

Theoretical derivation of GMR

The goal of this chapter is to show how GMR and the adapted GMR are derived and how initial intuition of these methods was obtained by exploratory simulations. A mathematical derivation of the GMR can also be found in *Sung* [56]. The derivation of GMR is mentioned in Section A.1.1 for completeness. The derivation of the aGMR is found in Section A.1.1. Additionally, the results and discussion of two exploratory simulations are shown in Section A.2.1.

A.1 Theoretical derivation of GMR

The derivation of GMR is more intuitive compared to HMM, because GMR does not include sequentiality in the regression. The goal of GMR is to convert discrete states into a continuous trajectory. GMR can only be used with encoding methods that make use of latent variables to encode the data. Regression is performed via the expectation of the output data given the input.
The goal of the derivation is obtain an expectation function from the conditional probability of the output (y) given the input (x) such that a value can be sampled from the GMM with only the known input.
The derivation of GMR starts by the marginal joint distribution of GMM of two variables, input (x) and output variable (y). The GMM is found in Equation A.1), where π_j is the prior of Gaussian j and ψ a multivariate Gaussian with mean $\boldsymbol{\mu}_j$ and variance $\boldsymbol{\Sigma}_j$. It is obtained from the GMM by computing the sum over all K Gaussian components, weighted by their respective priors.

$$p(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{K} \pi_j \phi(\mathbf{x}, \mathbf{y} | \boldsymbol{\mu}_{\mathbf{xy}j}, \boldsymbol{\Sigma}_{\mathbf{xy}j}) \tag{A.1}$$

The conditional probability of the output given the input is derived via Bayes rule, which can be seen in Equation A.2.

$$\begin{aligned} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) &= p(\mathbf{x},\mathbf{y}) \\ p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})} \end{aligned} \tag{A.2}$$

The joint probability is partitioned into Equation A.3, according to *Mardia et al.* [57]. Rewriting the joint probability eases finding the marginal probability, and therefore also obtaining the likelihood.

$$p(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{K} \pi_j \phi(\mathbf{y}|\mathbf{x}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\sigma}_j^2)\phi(\mathbf{x}, \boldsymbol{\mu}_{j\mathbf{x}}, \boldsymbol{\Sigma}_{j\mathbf{x}}) \tag{A.3}$$

Where $\phi(\mathbf{y}|\mathbf{x}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\sigma}_j^2)$ describes the conditional probability density function (pdf) for Gaussian j with mean $\mathbf{m}_j$ and variance $\boldsymbol{\sigma}_j^2$, $\phi(\mathbf{x}, \boldsymbol{\mu}_{j\mathbf{x}}, \boldsymbol{\Sigma}_{j\mathbf{x}})$ describes the pdf for Gaussian j with mean ($\boldsymbol{\mu}_{jx}$) and the variance of x ($\boldsymbol{\Sigma}_{jx}$). The mean ($\mathbf{m}_j$) is defined as in Equation A.4, which is known as the regression function or conditional expectation for Gaussian j. The variance is defined as in Equation A.5 which is found by the variance of $\mathbf{x}$ ($\boldsymbol{\Sigma}_x$) and covariance between $\mathbf{x}$ and $\mathbf{y}$ ($\boldsymbol{\Sigma}_{xy}$)

$$\mathbf{m}_j(\mathbf{x}) = \boldsymbol{\mu}_{j\boldsymbol{y}} + \boldsymbol{\Sigma}_{j\boldsymbol{y}\boldsymbol{x}}\boldsymbol{\Sigma}_{j\boldsymbol{x}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{jx}) \tag{A.4}$$

$$\boldsymbol{\sigma}_j^2 = \boldsymbol{\Sigma}_{j_{YY}} - \boldsymbol{\Sigma}_{j_{YX}}\boldsymbol{\Sigma}_{j_X}^{-1}\boldsymbol{\Sigma}_{j_{XY}} \tag{A.5}$$

The marginal density function of the input variable is found by integrating over the output variables (Equation A.6). It results in pdf that only depends on the observation, because taking into account all probabilities of the whole space should result in a probability of one.

$$p(\mathbf{x}) = \int p(\mathbf{x},\mathbf{y})dy = \sum_{j}^{K} \pi_j \phi(\mathbf{x}, \mu_{jx}, \boldsymbol{\Sigma}_{jx}) \tag{A.6}$$

The conditional probability ($\mathbf{y}$ given $\mathbf{x}$) is obtained using Bayes rule, by substituting equations A.3 and A.6 into equation A.2

$$p(\mathbf{y}|\mathbf{x}) \sum_{j}^{K} \pi_j \phi(\mathbf{x}, \boldsymbol{\mu}_{jx}, \boldsymbol{\Sigma}_{jx}) = \sum_{j=1}^{K} \pi_j \phi(\mathbf{x}, \boldsymbol{\mu}_{jx}, \boldsymbol{\Sigma}_{jx})\phi(\mathbf{x}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\Sigma}_j) \tag{A.7}$$

$$p(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{K} \frac{\pi_j \phi(\mathbf{x}, \boldsymbol{\mu}_{jx}, \boldsymbol{\Sigma}_{jx})}{\sum_j^K \pi_j \phi(\mathbf{x}, \boldsymbol{\mu}_{jx}, \boldsymbol{\Sigma}_{jx})} \phi(\mathbf{x}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\Sigma}_j) \tag{A.8}$$

$$p(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{K} w_j(\mathbf{x})\phi(\mathbf{x}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\Sigma}_j) \tag{A.9}$$

Equation A.9 can also be rewritten into a more intuitive notation. This is done by introducing a new variable, the responsibility. This is the same as the weight $(w_j(\mathbf{x}))$ in Equation A.9. The responsibility shows how well component j explains the data. The second part of the equation is the probability that an output point belongs to the component j. This results into the following notation:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{K} \underbrace{w_j(\mathbf{x})}_{P(\mathbf{z}_j|\mathbf{x})} \underbrace{\phi(\mathbf{y}, \mathbf{m}_j(\mathbf{x}), \boldsymbol{\Sigma}_{\boldsymbol{j}})}_{P(\mathbf{y}|\mathbf{z}_j)} \tag{A.10}$$

Where z is considered a latent variable (in this case a Gaussian j) However, we are only interested in the expectation of the posterior probability (Equation A.11). There is no need to calculate the likelihood of the conditional probability each iteration. Only the mean (A.4) and the weight are of importance to calculate the expectation/regression function.

$$\mathbf{y}_{est} = E(\mathbf{y}|\mathbf{x}) = \sum_{j}^{K} w_j(\mathbf{x})\mathbf{m}_j(\mathbf{x}) \tag{A.11}$$

The regression function is a summation of the means based on the weights. Therefore, the new data point is found by the weighted sum of the conditional means. The means are scaled by how well the corresponding component explains the data. It considers how likely it is that a certain observation belongs to a component and thus how much influence a component should have on the resulting output.

A.1.1 Adapted GMR

Similar to GMR the goal is to find an expression for the conditional expectation of the input given the output for the HMM instead of GMM. In order to so, the probability relation in Equation A.10 is used as start of the GMR derivation, which is for clarity written into Equation A.12. The $P(\mathbf{y}|\mathbf{z}_j)$ shows the likeliness of an output value based on the state z. The $P(\mathbf{z}_j|\mathbf{x})$ shows the likeliness of the state based on the input. The input variable is classified to a state and based on this state an output variable is sampled. The terms in Equation A.12 have to be expressed with variables/probabilities of HMMs.

$$P(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{K} P(\mathbf{z}_j|\mathbf{x})P(\mathbf{y}|\mathbf{z}_j) \tag{A.12}$$

The observation probability $P(\mathbf{y}|\mathbf{z}_j)$ of Equation A.12 equals the emission probability of HMM.

Next the responsibility for HMM is required to obtain the posterior distribution as in Equation A.12. The sequentiality in the HMM requires to take into account the whole

sequence of data points, or at least have some memory. Bayes rule is applied to find the responsibility [58] which can be seen for state j at time instance n ($\mathbf{z}_{jn}$) in Equation A.13.

$$\gamma_j(\mathbf{x}_n) = p(\mathbf{z}_{jn}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{z}_{jn})P(\mathbf{z}_{jn})}{P(\mathbf{X})} \tag{A.13}$$

Equation A.13 can be rewritten into Equation A.14 by introducing the forward variable α and the backward variable β. It is important to note that in Equation A.14, the subscript $_n$ is used to indicate time instance of the responsibility.

$$\gamma_j(\mathbf{x}_n) = \frac{p(\mathbf{x}_1 ... \mathbf{x}_n, \mathbf{z}_{jn}) p(\mathbf{x}_{n+1} .. \mathbf{x}_N | \mathbf{z}_{jn})}{p(\mathbf{X})} = \frac{\alpha_j(\mathbf{x}_n)\beta_j(\mathbf{x}_{jn})}{p(\mathbf{X})} \tag{A.14}$$

The forward variable is defined as in Equation A.16, a derivation can be found in [58]. The forward variable is the probability that a sequence of data happens at the same time with state $\mathbf{x}$.

$$\alpha_j(\mathbf{x}_n) = p(\mathbf{x}_1 ... \mathbf{x}_n, \mathbf{z}_{jn}) \tag{A.15}$$

$$\alpha_j(\mathbf{x}_n) = p(\mathbf{x}_n|\mathbf{z}_{jn}) \sum_{\mathbf{z}_n - 1} \alpha_j(\mathbf{x}_{n-1}) p(\mathbf{z}_{jn}|\mathbf{z}_{n-1}) \tag{A.16}$$

The backward variable is the conditional probability that a sequence will happen given the current state and the next state.

$$\beta_j(\mathbf{x}_n) = p(\mathbf{x}_{n+1}, .., \mathbf{x}_N | \mathbf{z}_{jn}) \tag{A.17}$$

$$\beta_j\left(\mathbf{x}_n\right) = \sum_{\mathbf{z}_{n+1}} \beta_j\left(\mathbf{z}_{n+1}\right) p\left(\mathbf{x}_{n+1} \mid \mathbf{z}_{n+1}\right) p\left(\mathbf{z}_{n+1} \mid \mathbf{z}_{jn}\right) \tag{A.18}$$

One can already conclude that it is not possible to calculate the backward variable in real time. The backward variable depends on observations of the future, which are not measurable at time step n. However, it is not relevant at time step n to know the responsibility over the whole time sequence. Only the responsibility at up to time instance n is relevant, because the goal is to sample the corresponding output y at time instance n and not at time instance $n + 1$. Therefore, only the forward variable is used to calculate the responsibility, which results into Equation A.19.

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \sum_{j=1}^{K} P(\mathbf{z}_j|\mathbf{x}) P(\mathbf{y}|\mathbf{z}_j) \\ &= \sum_{j=1}^{K} \frac{\alpha_j(\mathbf{x})}{P(\mathbf{x})} P(\mathbf{y}|\mathbf{z}_j) \\ &= \sum_{j=1}^{K} \frac{\alpha_j(\mathbf{x})}{\sum_{q=1}^{K} \alpha_q(\mathbf{x})} P(\mathbf{y}|\mathbf{z}_j) \end{aligned} \tag{A.19}$$

The expectation of the conditional probability (Equation A.19) is notated as in Equation A.20, which is the aGMR function. This is in line by the less mathematical derivation proposed by *Calinon et al.* [24].

$$\mathbf{y}_{est} = E(\mathbf{y}|\mathbf{x}) = \sum_{j}^{K} \frac{\alpha_j(\mathbf{x})}{\sum_{q=1}^{K} \alpha_q(\mathbf{x})}(\mathbf{x})\mathbf{m}_j(\mathbf{x}) \tag{A.20}$$

A.2 Exploratory simulations

The simulations were performed with code and a dataset provided by *Calinon* [59]. The GMM, HMM and GMR were already included in this code base. The GMR code was adapted to obtain the aGMR.
The dataset are demonstrations of letters written in 2 dimensions. An example of the letter Q is visible in Figure A.1. The demonstrations include position, velocity

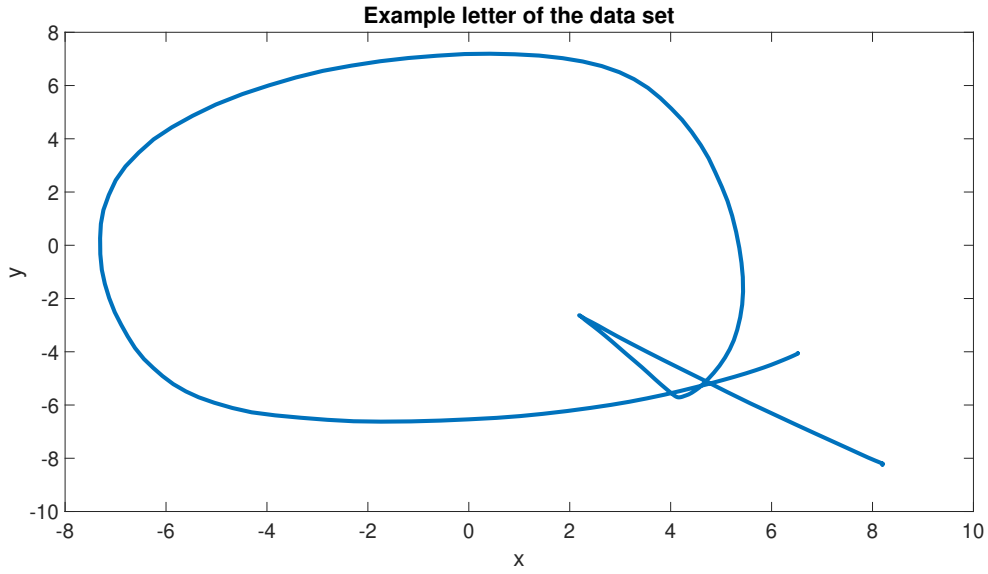


Figure A.1: An example of the dataset which is used. The letter Q is plotted, but the data set consists of all the letters of the alphabet.

and acceleration information in two dimensions and time with only one dimensions. For the reproduction test, the goal was to reproduce only the y component of the letter Q (see Figure A.1). The time is used as input, because it ensures that there is a relation between the data and position. The data is time dependent and there are no singularities. In addition, it allowed to use a left-to-right topology for the HMM. The first state is on the left and the final state is on the right without feedback between the states. The parameters used are the same for both frameworks and can be found in Table A.1. Both algorithms were initialized with K-means and afterwards learned with E-M for GMM and the Baum-Welch algorithm for HMM.

The quality of reproduction is based on visual inspection, which means there is no quantitative metric. The comparison was evaluated by inspecting which of the reproduction more closely represents the demonstrations.

Table A.1: Parameters used in to reproduce the y-direction of the letter q, including the time step dt and duration T.

Demonstrations	$5\frac{Nm}{rad}$
States/Gaussians	5
dt	$0.0001s$
T	$10s$

A.2.1 Results & Discussion

The HMM shows a more linear reproduction behaviour compared to the GMM as seen in Figure A.2. The states however are captured at around the same time. The linearity has to be related with the responsibility, because it is the only difference between the GMR and aGMR. This is also evident in Figure A.3 and A.4. Here the responsibility and reproduction are plotted. The responsibility of the GMM is more Gaussian shaped while transitioning from one state to another, while the responsibility of HMM results in a faster saturation to a probability of one. That results is evident in for example Gaussian 5, all the data points in the vicinity of the state are a linear line with direction of that Gaussian. A solution to this could be the use of more states, however that also requires more data and increases computational time. While, the GMM does not require this. Therefore, based on these results GMM was deemed to be more suitable.

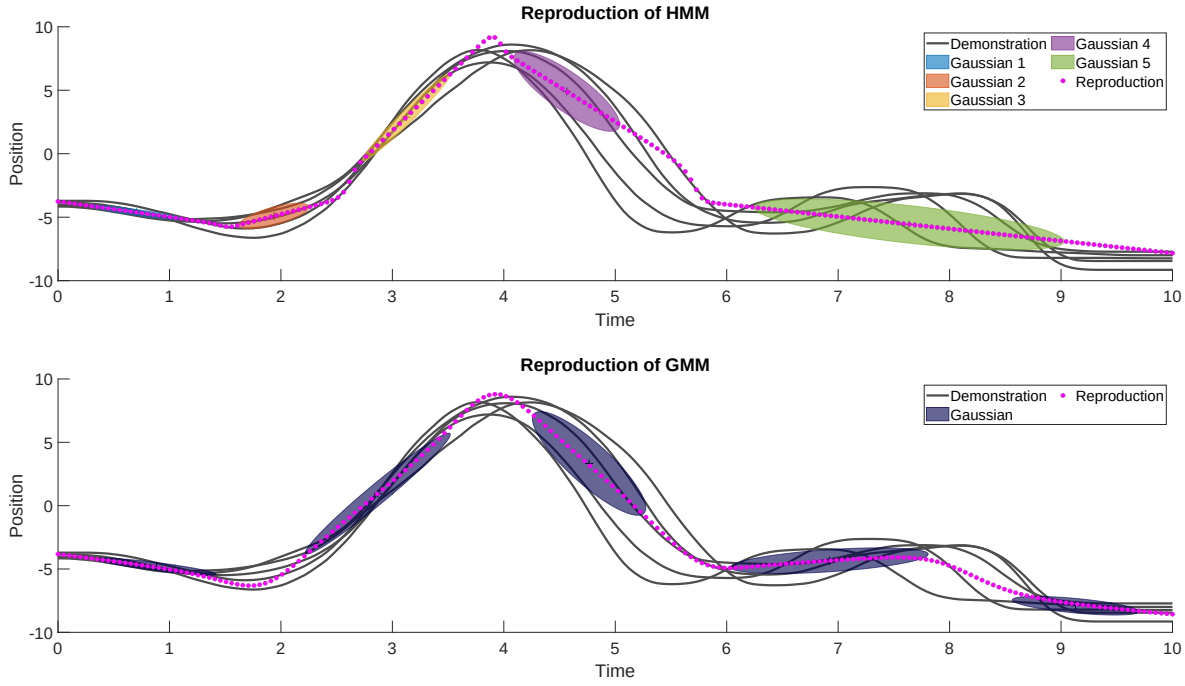


Figure A.2: Exploratory results which shows that the HMM (top graph) result in a less desired reproduction compared to the GMM (bottom graph). The dotted magenta line shows the reproduction, the grey lines are the demonstrations and the colored areas are the states/Gaussians.

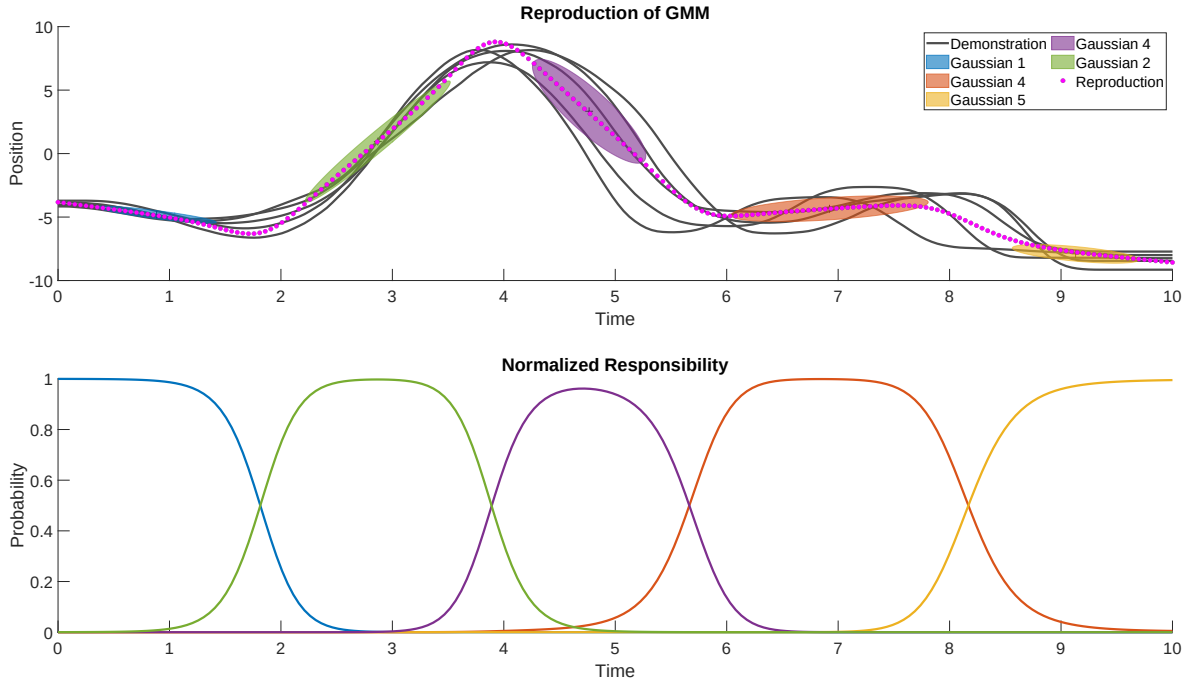


Figure A.3: The top plot is the reproduction of the y component of the letter Q by the GMM. The bottom plot shows the responsibility.

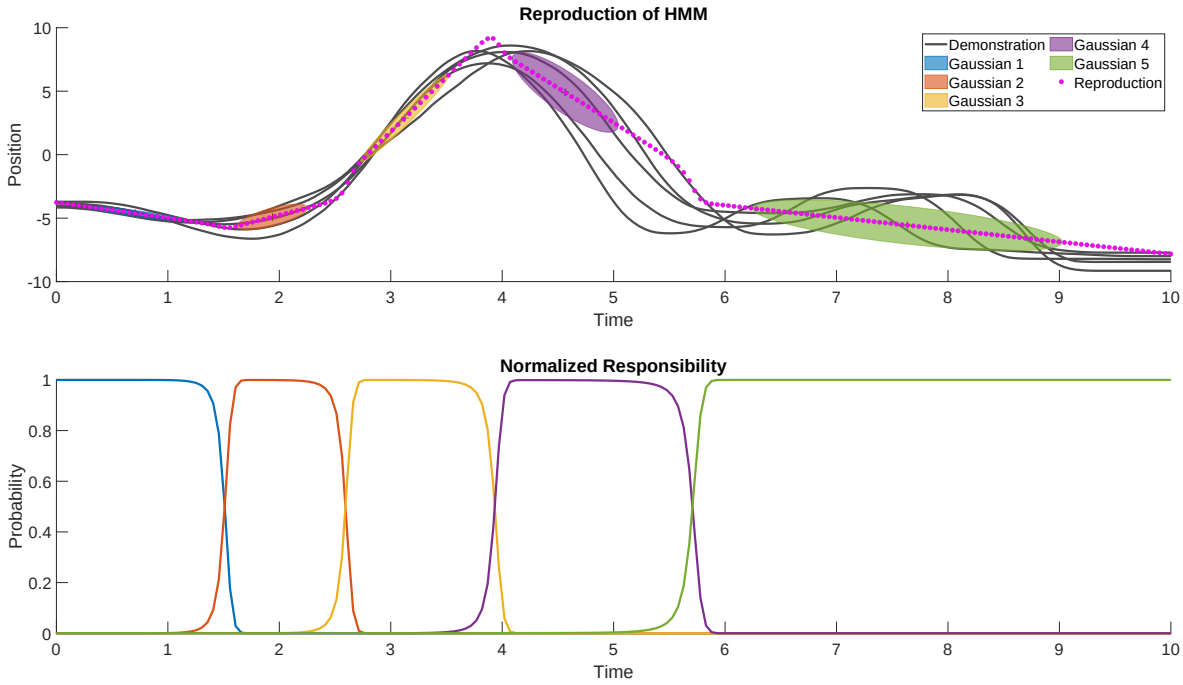


Figure A.4: The top plot is the reproduction of the y component of the letter Q by the HMM. The bottom plot shows the responsibility.

Appendix B

Valve task simulations

To ease the modelling of the valve task, the task was split in three complexity levels. The first level is the linear dynamics of the system, only the linear system is modelled without thresholds/ function. This model consists of PD-controller, admittance controller and feedback loops. The second level is the same as the first level, but it includes a saturation function to model the end position of the valve. The last level includes the force threshold combined with the saturation of the valve.

B.1 First level of complexity

The valve-task has different levels of complexity while modelling. The schematic of how the task is modelled can be seen in Figure B.1.

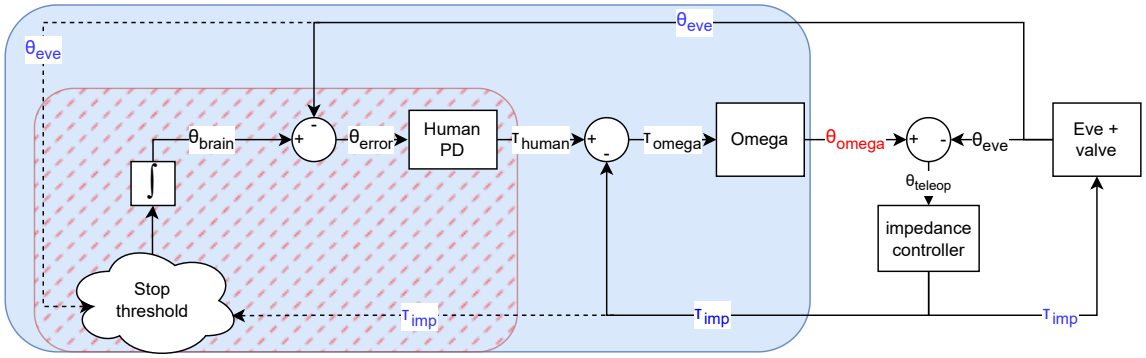


Figure B.1: A schematic overview of the valve closing task.

If the stop threshold has not been exceed, a constant velocity is produced to ensure the human continues rotating until the valve is closed. The PD controller and impedance controller are modelled with the standard notation for a PD-controller

as seen in Equation B.1 and B.2 respectively. It is important to note that the PD-controller is ideal and the derivative is unfiltered, which results in an aggressive PD-controller.

$$K_p\theta_{error} + K_d\dot{\theta}_{error} = \tau_{human} \tag{B.1}$$

$$K_{pimp}\theta_{teleop} + K_{dimp}\dot{\theta}_{error} = \tau_{teleop} \tag{B.2}$$

The Omega and Eve+damper are modelled as an admittance system as seen in Equation B.3 and B.4.

$$I\ddot{\theta}_{omega} = \tau_{omega} + D\dot{\theta}_{omega} \tag{B.3}$$

$$I\ddot{\theta}_{eve} = \tau_{imp} + D\dot{\theta}_{eve} \tag{B.4}$$

The parameters used to encode the system can be found in Table B.1.

Table B.1: Parameters used within the valve task model. The dynamics of the Eve and valve are concatenated, because they are modelled in series.

Damping Eve and valve [$\frac{\text{Nms}}{\text{rad}}$]	2
Inertia Eve and valve [kgm^2]	0.25
Damping Omega [$\frac{\text{Nms}}{\text{rad}}$]	0.75
Inertia Omega [kgm^2]	0.01
Stiffness impedance controller [$\frac{\text{Nm}}{\text{rad}}$]	35
Damping impedance controller [$\frac{\text{Nms}}{\text{rad}}$]	2
Stiffness human [$\frac{\text{Nm}}{\text{rad}}$]	10
Damping human [$\frac{\text{Nms}}{\text{rad}}$]	1
Estimated rotation velocity[$\frac{\text{rad}}{\text{s}}$]	2.5
Max valve angle [rad]	5π
Length of demonstration [s]	10
Sample time [dt]	0.002

The initial conditions of the system are specified in the following manner: The Omega and Eve start at the same position with the initial velocity and acceleration set to 0. The initial value of the human angle is the initial Eve/ Omega angle with an 0.035 rad offset as it is not realistic that the human starts at exactly the same position of the Eve. Furthermore, it results in a zero error which leads to a system that does not move due to the multiplication with zero. It is expected that the Omega and Eve start at the same angle and diverge, because of how the PD-controller is tuned. The damping induces some saturation time to steady state, which has as a result that the Eve will not reach the same position as the Omega/input. In addition, the torque should be the same for the different initial conditions. The torque should converge

to a steady state value. This is accordance with the result as seen in Figure B.2 for the angle and Figure B.3 for the torque.

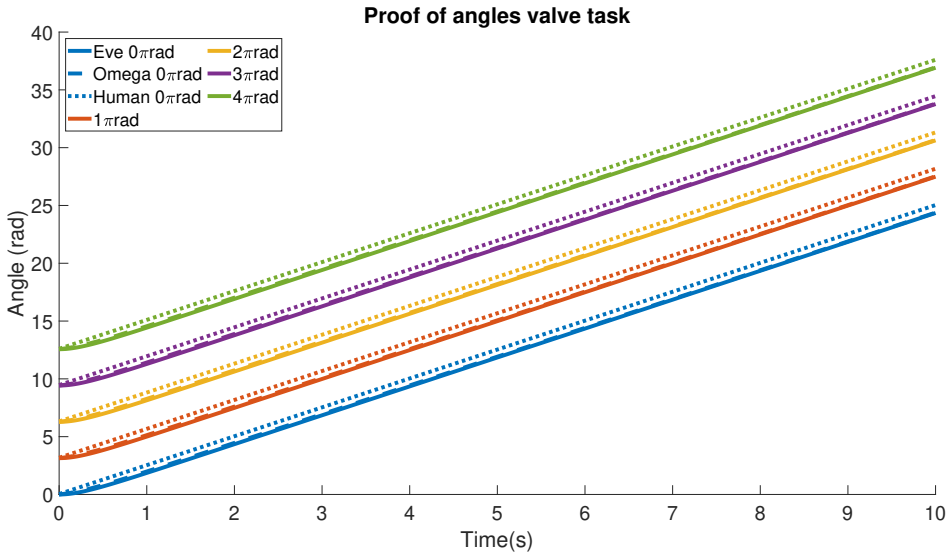


Figure B.2: The angles within the valve task without limits modelled. It is a linear system. The solid line represents the angle of the Eve/valve, the striped line the angle of the Omega and the dotted line the angle of the human.

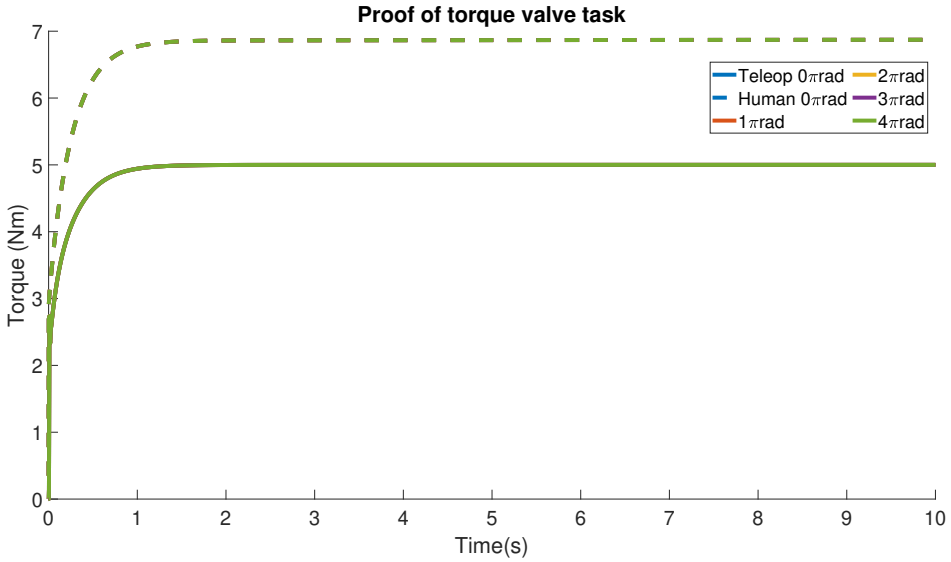


Figure B.3: The torque results within the valve task without limits modelled. It is a linear system. The solid line represents the torque of the impedance controller, the striped line represents the human torque.

B.2 Second level of complexity

The model for the second level of complexity is the same as the first level, but a saturation function is added to model the end-position of the valve. The most trivial method is would be a clipping function, but that results in a sudden stop of the valve. A more representative situation is a gradual saturation towards the end-position, which is realised with a hyperbolic tangent. The transition between the constant rotation and hyperbolic behaviour should be tuned to a level in which the transition is smooth. If it is not properly tuned the transitions result in a peak in the PD-controller due to the aggressive controller. The exact implementation can be seen in the Pseudocode 1. The buffer position is used to ensure that the saturation function starts at the correct position and an artificial clock is used to ensure that hyperbolic tangent starts at a zero input. The resulting behavior of the simulations can be seen in Figure B.4, where all angles are plotted. It can be clearly seen that the human and Omega angle linearly keeps increasing, until the Eve encounters the end position to the valve at this point the system smoothly saturates. The torque (Figure B.5) increases until the error between the Eve and Omega and thus the human is saturated. As the end position of the valve is reached the torque increases until infinity, because the error between the Eve and Omega keeps increasing.

Algorithm 1 Saturation of valve angle

1: The algorithm uses n as time instance. It is a realtime algorithm, which means that $\mathbf{n-1}$ is the sample of previous time instance.
2: **Variables:** $maxValveAngle$, $humanAngleVelocity$
3: **Input:** Torque $\mathbf{T}$, Velocity $\mathbf{v}$, Artificial clock $\mathbf{c}$, Position $\mathbf{p}$, Buffer position $\mathbf{b}$, sample time dt
4: **Output:** Position $\mathbf{p}$
5: **Initialization:** $\mathbf{c}(0) = 0$, $\mathbf{b}(0) = 0$
6: **for each** time instance $\mathbf{n}$ **do**
7: $\quad \mathbf{a} = \frac{\mathbf{T}(n) - eveD \cdot \mathbf{v}(n-1)}{I}$
8: $\quad$ **if** $(\mathbf{p}(n-1) >= (maxValveAngle) - 1$ **then**
9: $\quad\quad$ **if** $\mathbf{b}(n) == 0$ **then**
10: $\quad\quad\quad \mathbf{v}(n) = \mathbf{v}(n-1) + \mathbf{a}(n) \cdot dt$
11: $\quad\quad\quad \mathbf{b}(n) = \mathbf{p}(n-1) + \mathbf{v}(n) \cdot dt$
12: $\quad\quad$ **end if**
13: $\quad\quad \mathbf{a}(n) = 0$
14: $\quad\quad \mathbf{p}(n) = \mathbf{b}(n) + \mathbf{tanh}(humanAngleVelocity \cdot \mathbf{c}(n))$
15: $\quad\quad \mathbf{v} = \frac{\mathbf{p}(n) - \mathbf{p}(n-1)}{dt}$
16: $\quad\quad \mathbf{c}(n) = \mathbf{c}(n-1) + dt$
17: $\quad$ **else**
18: $\quad\quad \mathbf{v}(n) = \mathbf{v}(n-1) + \mathbf{a} \cdot dt$
19: $\quad\quad \mathbf{p}(n) = \mathbf{p}(n-1) + \mathbf{v}(n) \cdot dt$
20: $\quad\quad \mathbf{c}(n) = 0$
21: $\quad\quad \mathbf{b}(n) == 0$
22: $\quad$ **end if**
23: **end for**

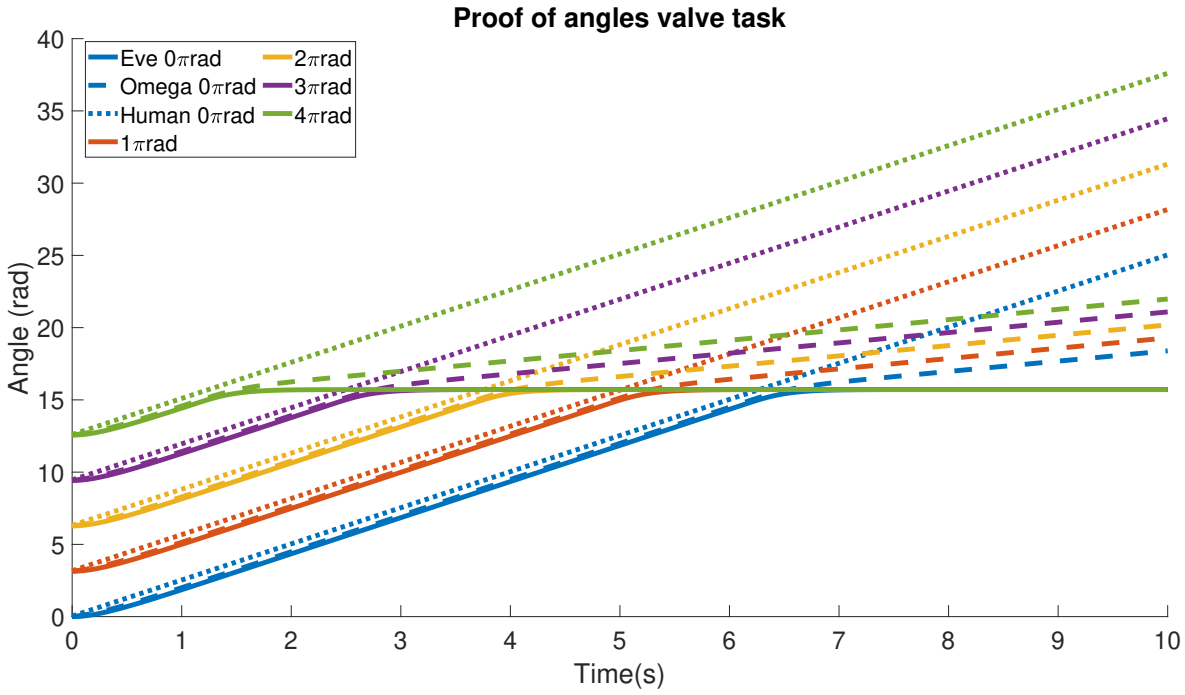


Figure B.4: The angles within the valve task after implementing the valve task saturation function. The continuous colored line represent the behavior of the Eve, the dotted line the human and the striped line the Omega.

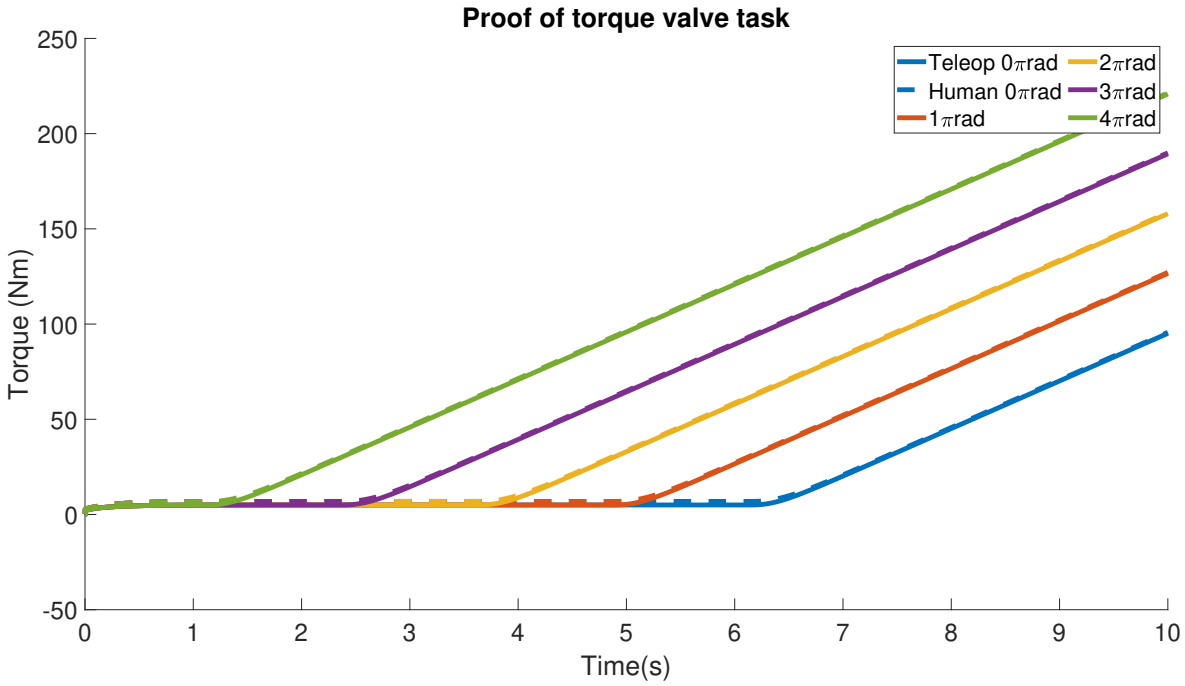


Figure B.5: The torques within the valve task after implementing the valve task saturation function. The continuous colored line represent the behavior of the Eve, the dotted line the human and the striped line the Omega.

B.3 Human threshold

Lastly, the behaviour of the human threshold is modelled combined with the saturation function. The threshold of the human stopping condition is implemented by Equation B.5.

$$\begin{aligned} &\textbf{If } \tau_{teleoperation} \geq 10 \text{ AND } \dot{\theta}_{eve} == 0 \\ &\quad \dot{\theta}_{human} = 0 \\ &\textbf{end} \end{aligned} \tag{B.5}$$

The Omega and human angle saturates a few seconds after the valve max angle is reached (Figure B.6). This behaviour can be improved by changing the velocity threshold to using a less than/or equal to a constant velocity. That would result in quicker saturation and more realistic scenario. However, the torque does not show behavior which is expected (Figure B.7). There are oscillations and it does not converges to zero. The oscillations are a result of the threshold, the sudden zero velocity result in some oscillations. These are amplified by the aggressive PD-controller. The problem can be solved by gradually decreasing the human velocity to zero. The constant torque is a result of the difference between the Eve and Omega angle, which is constant after saturation.

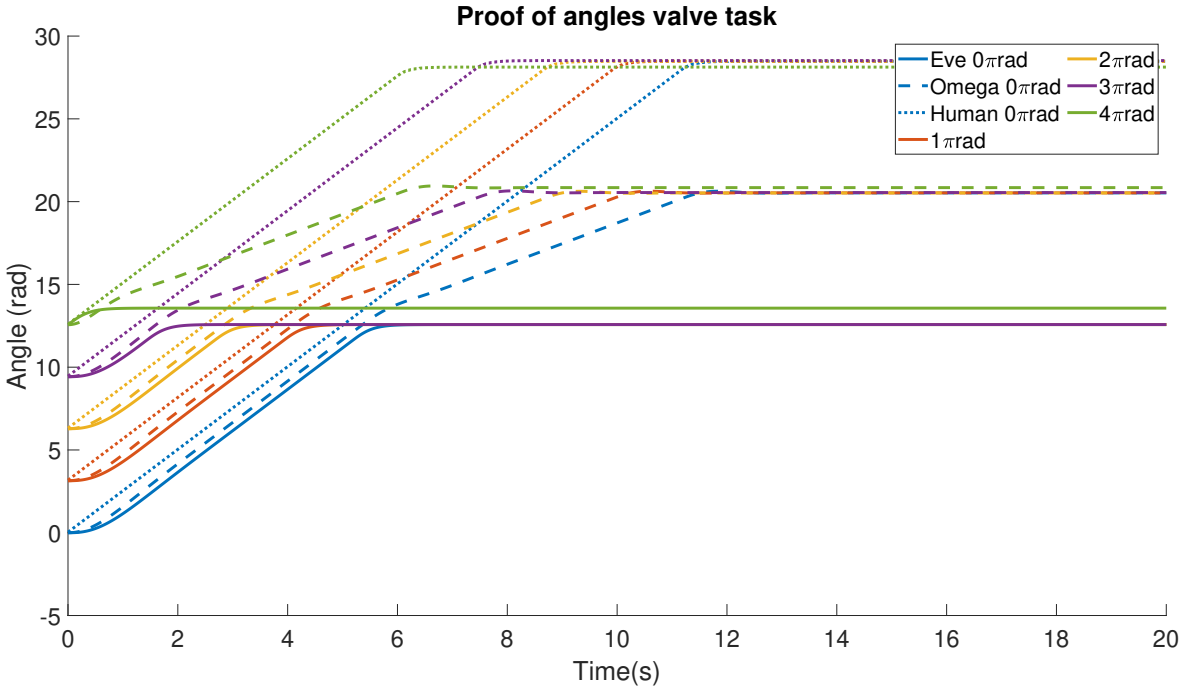


Figure B.6: The angles within the valve task after implementing human threshold. The continuous colored line represent the behavior of the Eve, the dotted line the human and the striped line the Omega.

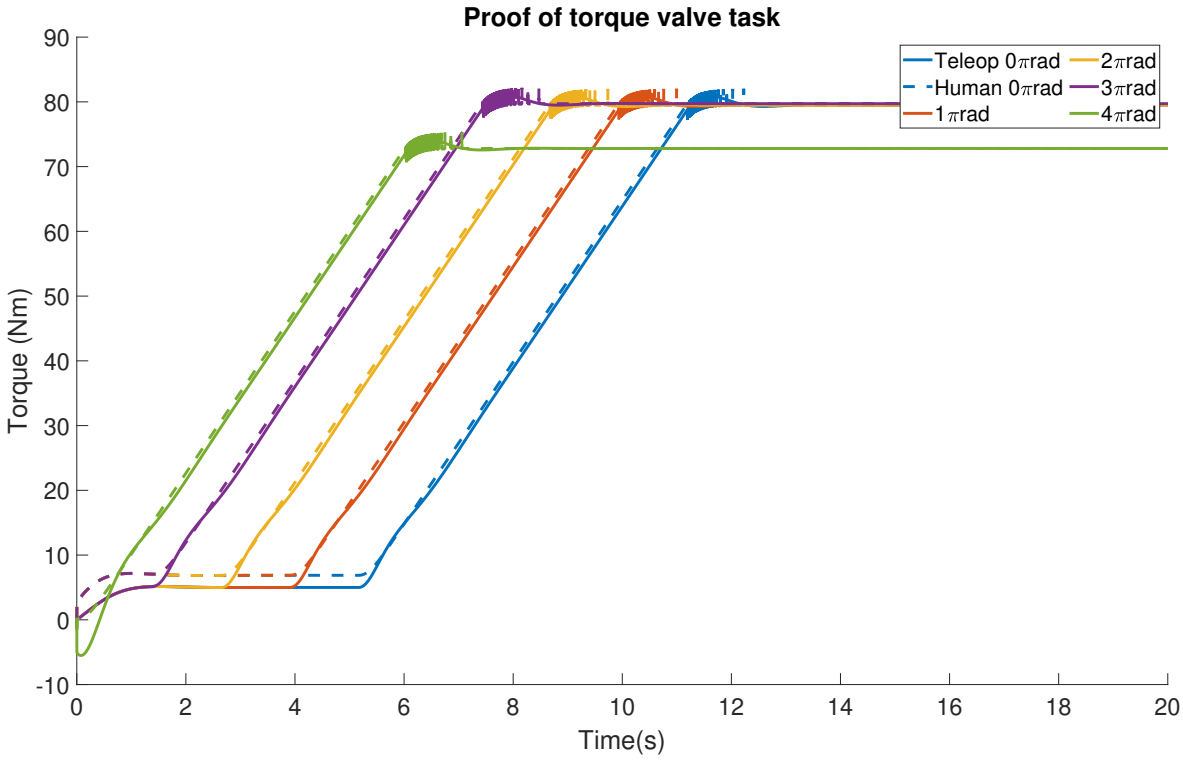


Figure B.7: The torques within the valve task after implementing the human threshold. The continuous colored line represent the behavior of the Eve, the dotted line the human and the striped line the Omega.

Appendix C

Additional results

C.1 Extrapolation result oen Gaussian GMM for the PD-controller

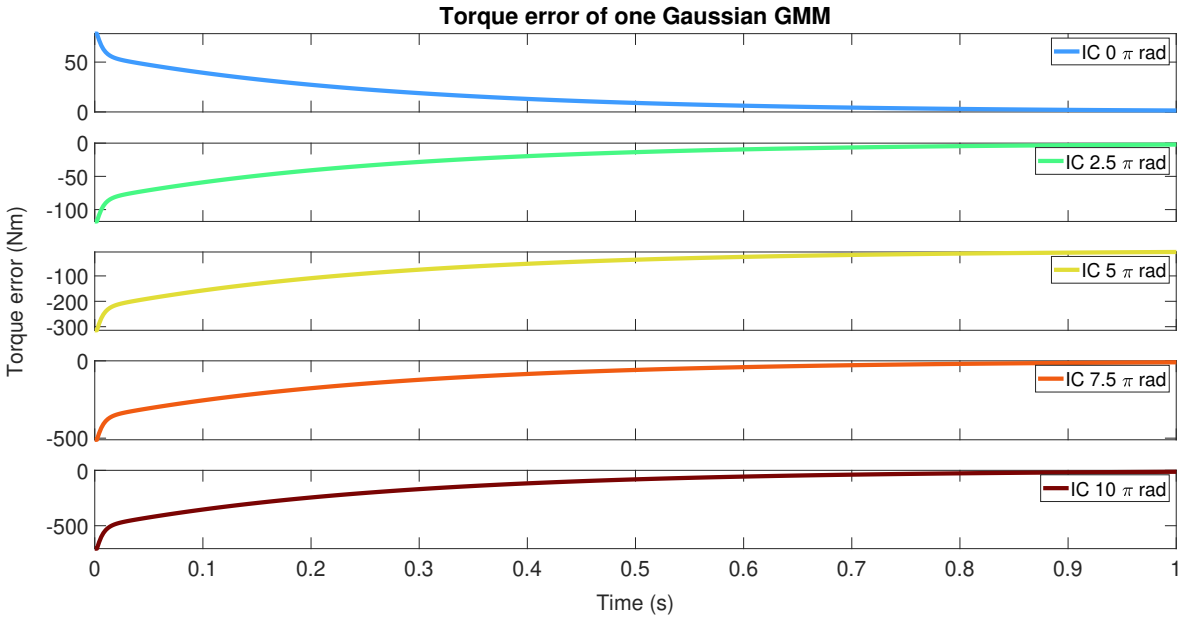


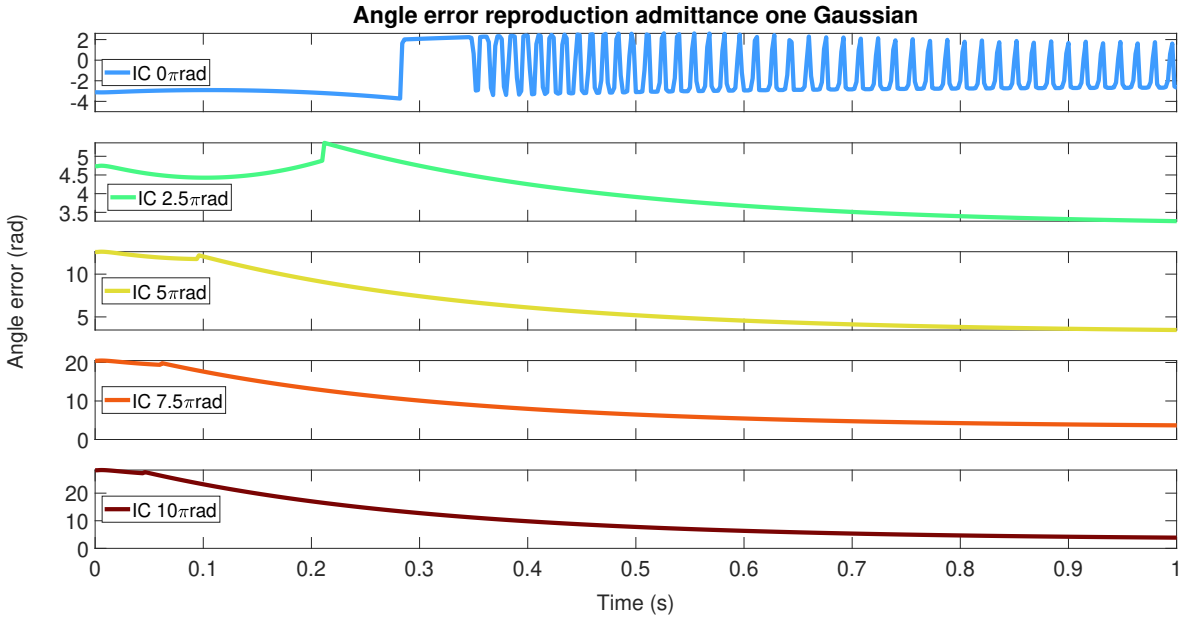
Figure C.1: The torque error [Nm] between the reproduction of one Gaussians GMM and PD-controller for the extrapolation initial conditions of the mass angle.

Table C.1: The RMSE value of the one Gaussian GMM for PD-controller while extrapolating.

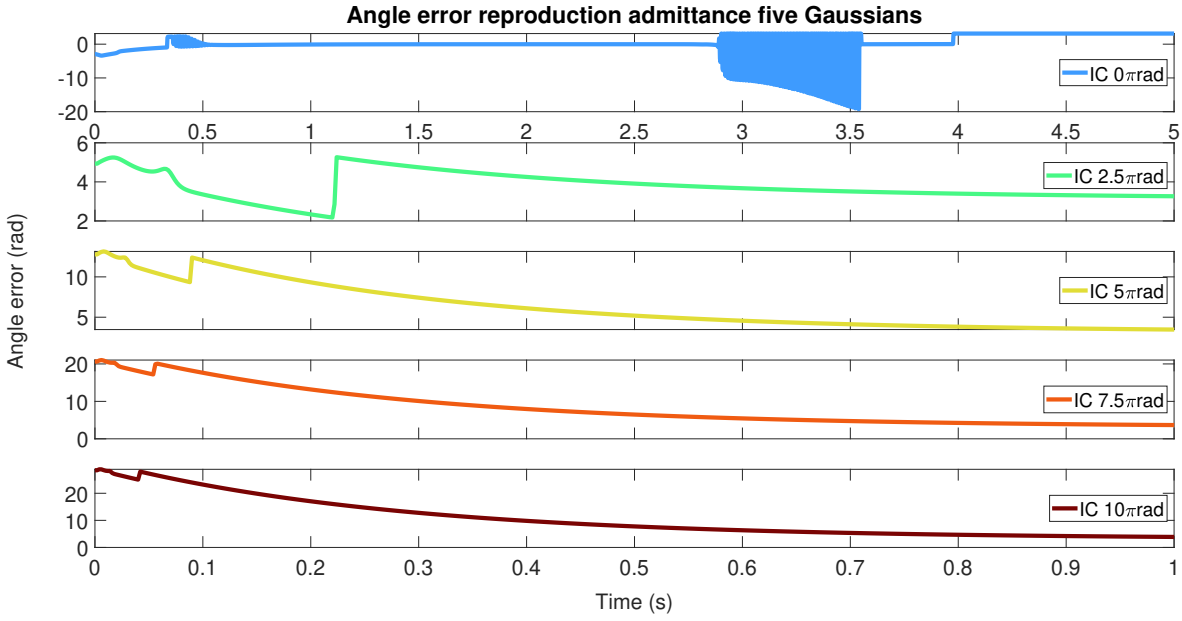
Initial condition [rad]	0	2.5π	5π	7.5π	10π
Five Gaussians [Nm]	9.657	14.485	38.6291	62.772	86.915

C.2 Admittance

C.2.1 Extrapolation



(C.2A) .



(C.2B)

Figure C.2: The angle error [rad] between the reproduction and admittance system for the extrapolation initial conditions of the mass angle. Figure C.2A: the error of the one Gaussian GMM. The plot is trimmed from 5 s to 1 s. Figure C.2B: the error of five Gaussians GMM. The bottom four panels are trimmed from 5 s to 1 s.

C.2.2 Time

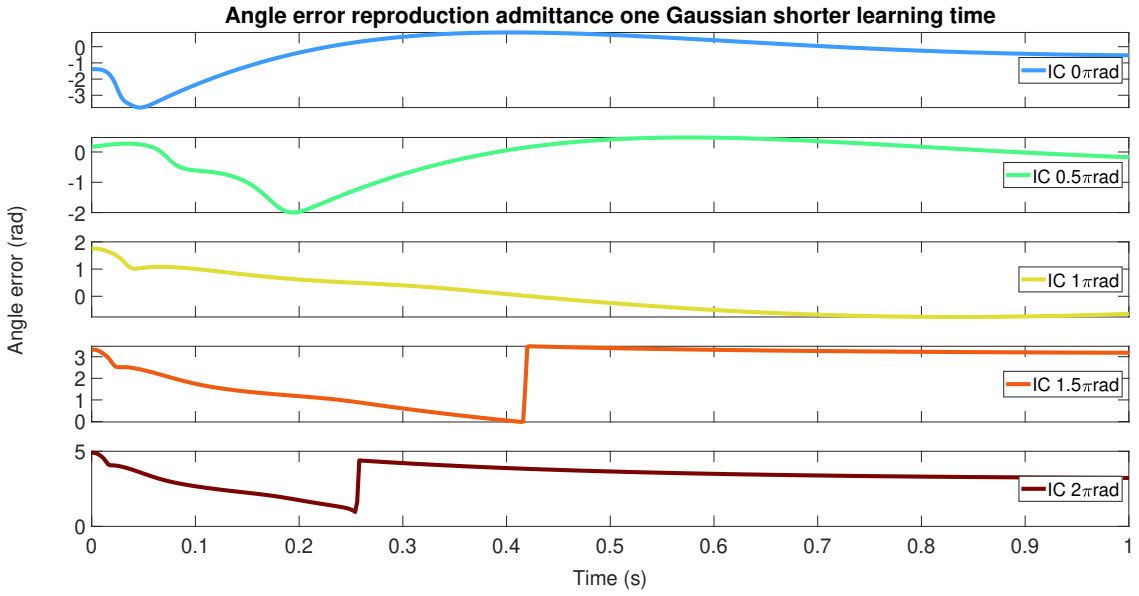


Figure C.3: The angle error [rad] between the reproduction of the one Gaussian GMM with shorter learning time and admittance controller for the interpolation initial conditions of the mass angle.

Table C.2: The RMSE value of the one Gaussian GMM with shorter learning time for admittance reproduction.

Initial condition [rad]	0	0.5π	1π	1.5π	2π
One Gaussian (shorter time)[rad]	0.5649	0.3384	0.3567	3.0569	3.1976

C.3 Human PD-controller

Table C.3: The RMSE value of the human PD-controller reproduction.

Initial condition [rad]	0	1π	2π	3π	4π
Five Gaussians [Nm]	0.0211	0.0213	0.0215	0.0218	0.0218