Interner Bericht

DLR-IB-RM-OP-2024-174

GMM-based Learning from Demonstration of Dynamic Reaching Motions for Elastic Joint Robots

Masterarbeit

Alexander Busch

Deutsches Zentrum
DLR für Luft- und Raumfahrt

SAPIENZA
UNIVERSITÀ DI ROMA

GMM-based Learning from Demonstration of Dynamic Reaching Motions for Elastic Joint Robots

Dipartimento di Ingegneria informatica, automatica e gestionale
Laurea Magistrale in Intelligenza Artificiale e Robotica

Alexander Busch
ID number 2018082

Advisor
Prof. Alessandro De Luca

Co-Advisors
M.Sc. Xuming Meng
Dr.-Ing. Manuel Keppler

Academic Year 2023/2024

Thesis defended on 22. October 2024
in front of a Board of Examiners composed by:

Prof. Alessandro De Luca (chairman)
Prof. Luca Becchetti
Prof. Roberto Beraldi
Prof. Gabriele Proietti Mattia
Prof. Giulia Fiscon
Prof. Andrea Cristofaro
Prof. Michele Scarpiniti

GMM-based Learning from Demonstration of Dynamic Reaching Motions for Elastic Joint Robots
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: busch.2018082@studenti.uniroma1.it

Abstract

Reaching for objects is a fundamental aspect of human motion and daily functioning. In robotics, conventional approaches lag in the ability to perform these reaching-and-grasping motions smoothly and continuously. Furthermore, traditional approaches are based on executing pre-planning motions. Perturbations of the target or the robot during the motion require re-planning, which is usually time-consuming and thus limits the reactiveness of the motion. In recent years, Learning from Demonstration (LfD) has gathered research efforts to overcome some of these challenges. LfD learns to take optimal actions even in unstructured environments based on the current state of the system. However, the LfD community mainly focuses on developing approaches for the well-established rigid joint robots. Compared to rigid joint robots elastic joint robots offer increased safety and performance and are thus well-suited for the fast reach-and-grasp motion. Due to their mechanical structure, elastic joint robot controllers require higher-order reference signals which is why state-of-the-art approaches for rigid joint robots cannot be transferred directly to elastic joint robots. This thesis extends a Gaussian Mixture Model (GMM) based LfD approach, the Stable Estimator of Dynamical Sytems (SEDS), to learn the object reaching motions. The proposed method provides not only the typical velocity reference signal but additional derivatives up to the jerk level for the elastic joint robots. Furthermore, additional constraints in the optimization are developed to prevent discontinuities of the reference signals. The method is successfully tested on the DLR robot *neoDavid*, an anthropomorphic robot with variable stiffness actuators. It is shown that the method successfully learned the characteristics of the demonstration data for fast-reaching motions. Additionally, the experiments show that providing additional control signals prevents an overshoot for highly dynamic motions.

Contents

List of Figures

Chapter 1

Introduction

Humans reach and grasp for objects in one smooth and continuous motion, adjusting arm and finger positions simultaneously without significant de-acceleration. While reach-and-grasp motions are essential for humans in their day-to-day lives, they also constitute a fundamental paradigm in robotic manipulation, with significant implications across household, service-oriented, and industrial robotics domains. Optimizing the efficiency and speed of this motion directly impacts the overall performance and utility of robotic systems in various applications. The ability to execute rapid and precise reach-and-grasp maneuvers is essential for enhancing the capabilities of robots in complex, dynamic environments.

However, conventional reach-and-grasp motions using collaborative robots are made up of a discontinuous sequence of actions. To simplify grasping, the end-effector is first positioned to a so-called pre-grasp pose, halts, and then approaches the target with almost zero velocity to create the quasi-static conditions of force-closure grasping. Both the pre-grasp pose and the gripper/finger contact locations are pre-planned and therefore need to be computed in advance. In case of a target or robot perturbation, a new plan has to be generated, making real-time applications challenging.

In recent years, Learning from Demonstration (LfD) has been used as a way to program mostly collaborative robots based on human demonstration data. Instead of being explicitly programmed with detailed instructions for every possible scenario, the robot learns to perform tasks by generalizing from the provided demonstrations. Due to their widespread usage in academia and industry, these methods are mainly developed for collaborative robots.

Articulated soft robots (ASR) are able to move faster than collaborative robots due to their intrinsic elasticity [7]. Furthermore, they are less likely to be damaged in case of a collision and are therefore well suited for fast reach-and-grasp motions. The elasticity comes at the price of major control challenges in set point regulation

and trajectory tracking as the lower stiffness results in lower control bandwidth and unwanted oscillation [1]. Higher-order reference signals are required to achieve precise control of the robot. However, most research on LfD focuses on developing methodologies for collaborative robots, where providing velocity commands is sufficient for most of the use cases. To achieve fast and continuous reach-and-grasp motions analogous to those of humans, it is necessary to extend state-of-the-art LfD approaches to meet the control requirements of elastic joint robots. Therefore, this work implements a motion generator and validates its capabilities on the elastic joint DLR robot *neoDavid* [4].

1.1 Goals of this Thesis

The thesis sets out to achieve the following characteristics of the developed motion generator:

(**G1**) It should provide a fast and continuous reaching trajectory.

(**G2**) It should be guaranteed to converge to the target position.

(**G3**) It should capture the characteristics of the demonstration data and be able to extrapolate from it.

(**G4**) It should successfully run on the *neoDavid* system, therefore providing higher order reference signals.

These goals will be revisited and discussed at the end of this work.

1.2 Contribution

In this work, I leverage the Elastic Structure Preserving (ESP) controller [8] and implement a motion generator that enables *neoDavid* to naturally execute a reaching motion. To this end, the thesis makes the following contributions:

1) The Stable Estimator of Dynamical Systems (SEDS) [5] of learning a stable dynamical system (DS) from demonstration data utilizing a Gaussian Mixture Model (GMM) is extended by deriving its expression analytically w.r.t. time to achieve higher-order reference trajectories of the reaching motion, which is a requirement for the controller of elastic joint robots.

2) The mixing terms of the GMM cause discontinuities of the higher-order reference signals, resulting in small regions (1 mm^3) where jerk signals vary in magnitude of 10^4. To prevent this, both upper and lower bounds on

the Gaussian covariance eigenvalues are introduced as constraints during the optimization.

3) The motion generator was successfully tested in simulation and validated on the *neoDavid* system. During the experiments, the higher-order time derivative reference signals eliminated overshoot in fast-reaching motions, and the eigenvalue constraints eliminated the strong variation in cartesian jerk (factor 10^4) signal magnitude within small state space regions (1 mm^3) otherwise encountered.

Chapter 2

Fundamentals

This chapter briefly introduces the most important concepts used in this thesis.

2.1 Background of Elastic Joint Robots

Since their invention, robots could primarily be found on factory floors, lifting heavy loads. Due to a lack of safety mechanisms that could avoid or detect an unintentional human-robot collision, they posed an intense risk and had to be separated from the human workforce. With advancements in technology, fields of application other than mundane lifting and moving heavy objects on an assembly line became possible. Most notable was the introduction of the so-called lightweight arms, specifically the LWR III developed by the DLR (Deutsches Zentrum für Luft- und Raumfahrt) in 2002, equipped with harmonic drives in the joints [9]. Measuring both link and motor position, the known stiffness of the harmonic drives could be used as a virtual torque sensor. The joint torque sensing enabled compliant environment interaction and safety due to possible collision detection and torque limiting. New applications in industry, medicine, service, and household emerged that leveraged this innovation to realize more complex tasks, enabling the coexistence and even cooperation of humans and robots [10], [11]. However, the harmonic drive-induced elasticity, where elasticity refers to the ratio of deflection and associated torque evaluated after the reduction gears, of an LWR joint with around $6000 - 15000$ Nm/rad [12] isn't enough to protect the drive trains in case of a high-speed collision with a stiff object. This was partly by design though, as increased elasticity comes at the price of major control challenges.

ASR embrace elasticity by deliberately introducing springs between the motors and the links. ASR springs are less stiff than the harmonic drive of lightweight robots. With only $200 - 800$ Nm/rad [2] joint stiffness they can absorb shocks and impacts without damaging the drive train. Furthermore, elastic joints offer the

Figure 2.1. The elastic joint robot *neoDavid* [1].

possibility to exploit their passive dynamics and enable more efficient and faster motions [13]. On the other hand, the elasticity introduces major control challenges, see Section 2.2.

2.1.1 DLR Robot *neoDavid*

The *neoDavid* robot [4] or *neoDavid* system, formerly the DLR hand arm system [3], is a wheeled, anthropomorphic, passively compliant robot designed to have human characteristics in regards to size, weight, and performance. It has a 7 degrees of freedom (DoF) right arm and a 20 DoF hand [3]. In recent years, a 5 DoF left arm with an 8 DoF two-finger hand, a 3 DoF neck, a 3 DoF torso, and a holonomic wheelbase were added. See Fig. 2.4 for a history of the development and the current setup.

Except for its holonomic base, its main feature is the variable stiffness actuation in all its joints. Variable stiffness actuators (VSA) have three main advantages:

1. The ability to absorb impact energy without transferring it directly onto the drive train, reducing harm to the colliding object and the robot.

2. In cyclic tasks, the motor side can excite the link side depending on its stiffness in a way that the natural vibration behavior of the system is used. Most of the movement takes place in the springs rather than the motors and gearboxes, causing the system to save significant energy [3].

3. The storage and release of potential energy in the springs enables the link side to achieve higher speeds than the maximum motor speed [14], yielding highly dynamic motions. VSA designs try to maximize stiffness range, power output, potential energy storage, efficiency, and accuracy while minimizing cost, weight,

and size. For a comprehensive review of the design and the components of VSA, the interested reader is referred to [7].

The following briefly introduces the types of VSA used in the *neoDavid* system. Specifically, the actuators of *neoDavid*'s arm are discussed as those are predominantly used in this thesis. The arm is made up of four DLR Floating Spring Joints (FSJ) [2] in the shoulder up to the elbow, followed by 3 bidirectional antagonistic variable stiffness Joints (BAVS) [15] in the forearm. FSJs are made up of two cam rollers, which are connected to the gear output shaft. The two cam disks are fixed to the link side and stiffness actuator. This design enables the storage of a lot of potential energy in a compact setup. Due to the exponential cam path design, the output stiffness varies w.r.t. the spring deflection. A small motor is used to set the minimal intrinsic stiffness by moving the cam disks, thus changing the pretension of the spring. See case *c)* in Fig. 2.2 where the stiffness actuator moved one cam disk and note the smaller deflection in *d)* compared to *b)* when the same external force is applied on the link side.

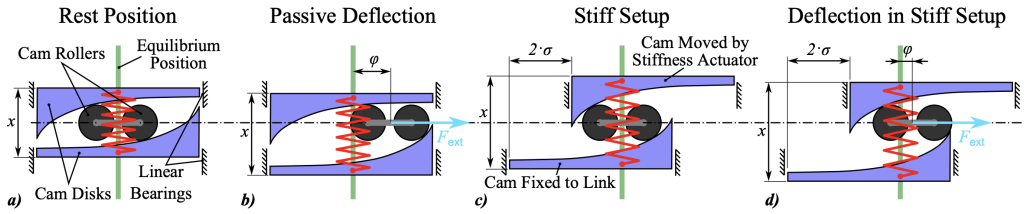


Figure 2.2. Comparison of joint deflection φ under an external force on the link side for different stiffness settings. ***a)*** resting position ***b)*** external force on link side ***c)*** moved cam disk to stiffen joint ***d)*** external force on the link side during stiff setup [2].

Since the forearm has to contain the drives for both the remaining three forearm joints as well as the drives of the hand's 20 DoF, using FSJ for the forearm wasn't an option. On the other hand, the compact, human-inspired, tendon-driven antagonistic joint type isn't enough to carry the necessary load of the hand. Thus, the BAVS was developed. Other than the antagonistic joint, the BAVS joint can use both motors to exert torque, making it able to carry a heavier load while having the same size as the antagonistic joint [15]. This comes at the cost of additional complexity due to the need for twice the amount of tendons and spring elements. Unlike the FSJ, varying the stiffness of the BASV joint doesn't require additional motors. The link side torque is the sum of the exerted torque of both motors. Stiffening the joint can be achieved by both motors exerting the same torque with opposite signs. The link will not move, but it will appear stiffer when an external force is applied.

As mentioned, the drives for the hand had to be contained in the forearm. This makes the hand light while still being able to execute dynamic motions. To meet the

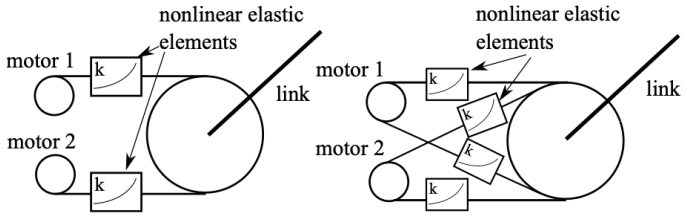


Figure 2.3. Comparison of antagonistic joint and bi-directional antagonistic joint (BAVS) [3].

requirements of small size, human-like power output, and compliance upon collision, the flexible antagonistic spring (FAS) mechanism [16] was designed with a magnetic sensor for tendon force measurement. In total, there are 36 FAS units for the 20 DoF-hand.

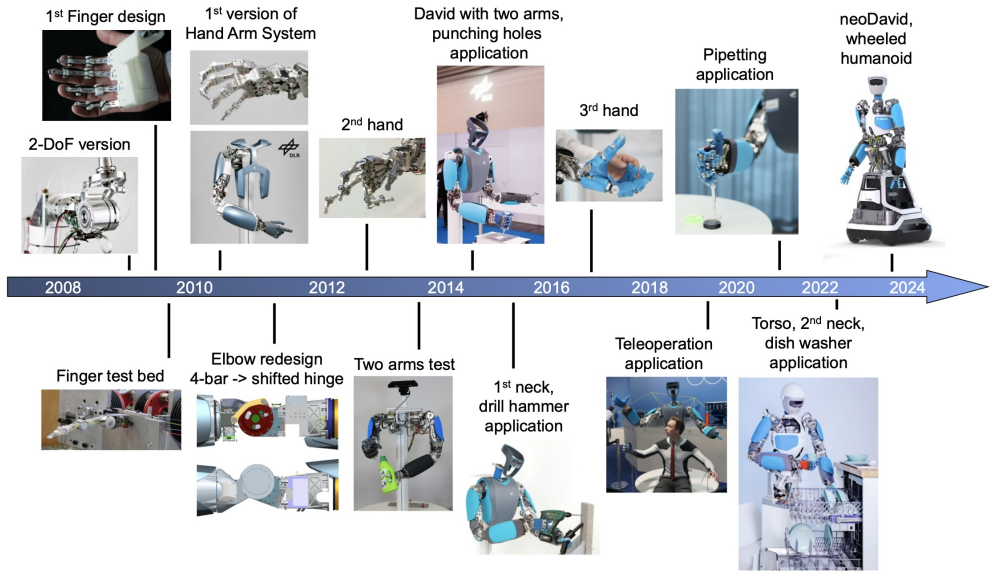
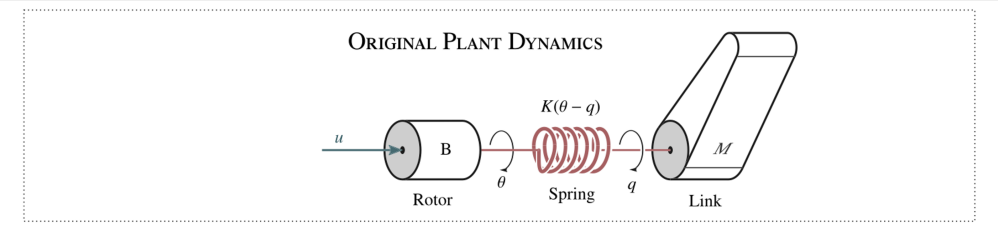


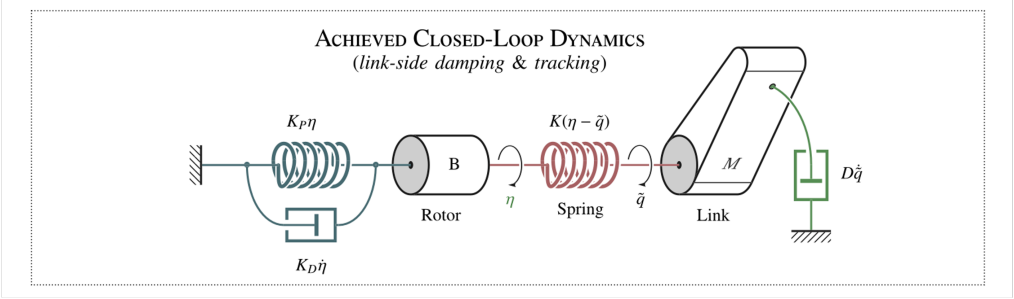
Figure 2.4. Development history of *neoDavid* [4].

2.2 Elastic Structure Preserving (ESP) Controller

One of the challenges in controlling elastic joint robots is that there is no direct control input to the link side, see Fig. 2.5a. The input u is given to the motor which, through a spring, drives the link. This renders the system underactuated due to it having fewer commands than generalized coordinates. Furthermore, the spring dynamics lead to undesirable oscillation at the link side and lower control bandwidth. The ESP controller [17] [8] is a state-of-the-art controller for link-



(a) Original Dynamics for linear stiffness elastic joint [8].



(b) Transformed Closed-Loop Dynamics with ESP controller [8].

side tracking of FSJ joints and is in use on the first four links on the *neoDavid* system. The FAS and the BASV forearm and wrist joints are controlled by the ESPi approach [18] and the work presented in [19] respectively. ESP leverages a coordinate transformation to obtain a system with link-side dynamics similar to the original one but with an additional damping term to counter the oscillation. The motor-side terms are minimally changed, most notably without an extensive inertia matrix shaping, without Coriolis compensation, and with the elastic properties of the system still intact. The tracking is achieved by a PD controller of the transformed motor coordinates. In the following, the controller is shortly introduced to give motivation for the additional reference signals developed in the thesis.

Considering the simplified sponge model [20] which assumes that the angular part of the kinetic energy of each rotor is due only to its own rotation, the robot dynamics are given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = -g(q) + \psi(\theta - q) + \tau_{ext}, \tag{2.1a}$$

$$B\ddot{\theta} + \psi(\theta - q) = u, \tag{2.1b}$$

where q and θ is the link and motor coordinates respectively, $\psi(\theta - q)$ is the nonlinear spring torque connecting link and motor. $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix of the rigid links, $B \in \mathbb{R}^{n\times n}$ the diagonal reflected inertia matrix of the actuators, $C(q,\dot{q})\dot{q} \in \mathbb{R}^{n\times n}$ the matrix of Coriolis and centrifugal forces, $g(q) \in \mathbb{R}^n$ the gravity vector.

To have link-side damping and the ability to track link-side trajectories, the goal

is to transform the current system into a system of the form of:

$$M(\tilde{q})\ddot{\tilde{q}} + C(\tilde{q},\dot{\tilde{q}})\dot{\tilde{q}} = -D\dot{\tilde{q}} + \psi(\eta - \tilde{q}) + \tau_{\text{ext}}, \tag{2.2a}$$

$$B\ddot{\eta} + \psi(\eta - \tilde{q}) = \bar{u}, \tag{2.2b}$$

where the new coordinates $\tilde{q}$ and η of the new system are introduced. This system has the additional term $-D\dot{\tilde{q}}$ to counter the intrinsic oscillation as well as the gravity vector. If it is possible to find a mapping between (2.1) and (2.2), namely $[q,\ \theta] \to [\tilde{q},\ \eta]$, a controller design would be rather straightforward. The mapping between q and $\tilde{q}$ is chosen to be $\tilde{q}(t) := q(t) - q_d(t)$. This helps in the later control design, as the controller needs to stabilize the system at the origin. Proceeding to find η w.r.t. θ it is helpful to notice that system (2.2a) and (2.1a) are only equal if

$$\psi(\theta - q) = \psi(\eta - \tilde{q}) + n(\tilde{q},\dot{\tilde{q}}), \tag{2.3}$$

$$\text{with} \qquad n(\tilde{q},\dot{\tilde{q}}) = g(\tilde{q}) - D\dot{\tilde{q}} + M(\tilde{q})\ddot{q}_d + C(\tilde{q},\dot{\tilde{q}})\dot{q}_d. \tag{2.4}$$

In general, the analytical inverse of $\psi(\theta - q)$ is not available, so it cannot be used to find the transformation. Instead, one can differentiate (2.3) w.r.t. time, then isolate $\dot{\theta}$ and differentiate again w.r.t. time to obtain

$$\ddot{\theta} = A(t,\eta,\tilde{q},\dot{\tilde{q}})\ddot{\eta} + \dot{A}(t,\eta,\tilde{q},\dot{\tilde{q}})\dot{\eta} + \dot{a}(t,\eta,\tilde{q},\dot{\tilde{q}},\ddot{\tilde{q}}), \tag{2.5}$$

which now only depends on the new coordinates and thus completes the coordinate transformation. See [8] for details on A and a. It is now possible to map the new system to the original one. Substituting (2.5) back into (2.1b) gives:

$$BA\ddot{\eta} + B\dot{A}\dot{\eta} + B\dot{a} + \psi(\theta - q) = u, \tag{2.6}$$

for the transformed motor dynamics. For easier controller design one can premultiply (2.6) by $A^\top$ and group $B_\eta = A^\top BA$ and $C_\eta = A^\top B\dot{A}$ to interpret them as the mass and Coriolis/centrifugal matrices of the transformed system to obtain

$$B_\eta\ddot{\eta} + C_\eta\dot{\eta} + A^\top\psi(\theta - q) + A^\top B\dot{a} = A^\top u. \tag{2.7}$$

Now, a controller in the transformed coordinates can be designed and easily applied to the original system by the developed transformation. The control signal is made up of three components:

$$u_{\text{esp+}} = \check{u}_{\text{esp+}} + \hat{u}_{\text{esp+}} + \bar{u}_{\text{esp+}}. \tag{2.8}$$

$\check{u}_{\text{esp+}} = B\dot{a}$ is used to eliminate the non-linear terms, $\hat{u}_{\text{esp+}} = \psi(\theta - q) - A^{-\top}\psi(\eta - \tilde{q})$

introduces the spring dynamics to the motor side. Lastly, $\bar{u}_{\text{esp+}}$ is the PD term

$$\bar{u}_{\text{esp+}} = -A^{-\top}(K_D\dot{\eta} + K_P\eta). \tag{2.9}$$

These control signals shape the system as little as possible since inertia matrix B_η and Coriolis/centrifugal matrix C_η are not compensated for. The simple PD controller brings η to the origin. It thus also brings $\tilde{q}$ to the origin and therefore achieves link-side tracking. Under the assumption of properly chosen K_P and K_D [8], the resulting closed-loop dynamics of the system are

$$M(\tilde{q})\ddot{\tilde{q}} + C(\tilde{q},\dot{\tilde{q}})\dot{\tilde{q}} = -D\dot{\tilde{q}} + \psi(\eta - \tilde{q}) + \tau_{\text{ext}}, \tag{2.10a}$$

$$B_\eta\ddot{\eta} + C_\eta\dot{\eta} + \psi(\eta - \tilde{q}) = -K_P\eta - K_D\dot{\eta}. \tag{2.10b}$$

One significant observation for this thesis is that $n(\tilde{q},\dot{\tilde{q}})$ in (2.4) which depends on $\ddot{q}_d$ is differentiated twice w.r.t. time to obtain the expression (2.5) for $\ddot{\theta}$. This means that the coordinate transformation depends on the fourth derivative of the desired joint angles, i.e. the snap $q_d^{(4)}$[1]. In the controller input, this manifests when using $\check{u}_{\text{esp+}}$ to compensate for the non-linear terms as for the computation of $\dot{a}$, $q_d^{(3)}$ and $q_d^{(4)}$ are required.

2.2.1 Lyapunov Stability

A Lyapunov function $V(\xi)$ is a scalar function that can be used to prove the stability of a dynamical system without solving the system differential equation explicitly. It is thus a useful tool to verify the characteristics of a controller or motion planner to ensure that the robot will always reach the desired reference signals. For a Lyapunov function (LF) to be valid, it must have a continuous 1^{st} time derivative $\dot{V}(\xi)$ and should be positive definite, with a negative definite 1^{st} time derivative

$$\begin{cases} (\mathbf{a})\ V(\xi) > 0 & \forall \xi \in \mathbb{R}^d,\ \xi \neq \xi^*, \\ (\mathbf{b})\ \dot{V}(\xi) < 0, & \forall \xi \in \mathbb{R}^d,\ \xi \neq \xi^*, \\ (\mathbf{c})\ V(\xi^*) = 0, & \dot{V}(\xi^*) = 0. \end{cases} \tag{2.11}$$

If such a function exists, the DS will asymptotically converge to the equilibrium ξ^*, and the system has asymptotic stability. If further $V(\xi)$ is radially unbounded, meaning $||\xi|| \to \infty \implies V(\xi) \to \infty$, then the system is globally asymptotically stable (GAS).

[1] In this thesis, the i-th time derivative of the position q is denoted as $q_d^{(i)}$

2.3 Learning from Demonstration

Learning from Demonstration is a way to teach robots tasks without explicitly hand-crafting a trajectory. The goal is to learn how to map from state to action by demonstrating a specific behavior. The word behavior or task reflects the fact that not a single trajectory is learned, but a type of movement, i.e., the reaching and grasping motion of an object from a set of initial configurations. There are different ways to acquire demonstration data for robot learning. The most convenient choice is often kinesthetic teaching, where the robot is guided by a human, and the trajectory data is recorded. Increasingly popular with more advanced computer vision algorithms is a demonstration from raw observations, where the learning framework takes video or image data as input. Tracking with a dedicated tracking system is also possible although these systems are costly and require careful calibration. Lastly, teleoperation can be used where the robot is controlled via an interface and the executed trajectory stored. However, these interfaces are often complex and costly systems.

Given a set of demonstrations, a trained policy maps the input information to a desired action. Usual action spaces are Cartesian/ joint space positions, velocities, and accelerations, or reference motor torques directly. There are different embodiments for a policy; neural networks (NN), dynamical motion primitives (DMP), and dynamical systems. They have in common that no conventional planning problem like a rapidly exploring random tree (RRT) search [21] or an optimization problem as in [22] has to be solved. These approaches require a significant computation time to come up with a full trajectory towards the target. They are thus extremely challenged by changing environments as constant re-planning would be required to adapt the plan to new situations.

A possible closed-loop system with a DS learned from data is depicted in Fig. 2.6. At every time step, the DS is queried with the current state ξ and outputs a reference signal by using the velocity of the DS given the current state as a desired velocity reference signal for the robot. Iteratively passing these desired velocities to a low-level control will guide the robot to the equilibrium of the system (usually the zero vector). This system architecture adds the motion generator to the control loop. To ensure convergence to arbitrary targets the DS isn't queried with the current position, but rather the position error w.r.t. the target. This has the advantage that perturbations of the target can be compensated for. If the target changes, not the system's attractor is changed, but the robot's position relative to it. This enables seamless reactive behavior as long as the position of the target is known. See Fig. 2.7 for a 2D example of a learned DS from data.

At first sight, it might seem as if LfD can be purely formalized as a machine

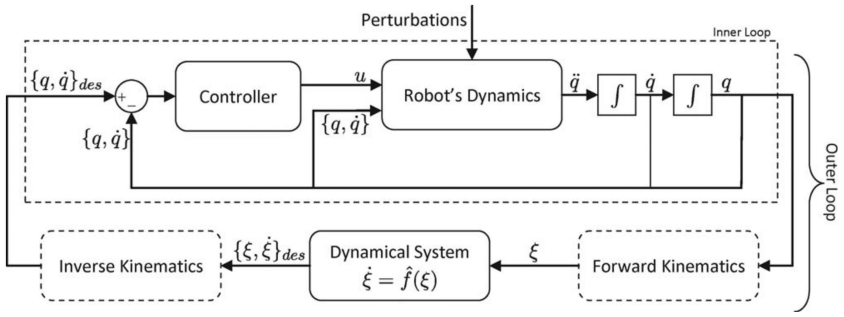


Figure 2.6. Possible system architecture using DS as motion generator for rigid joint robots [5].

learning problem with all its challenges and solutions. However, since the output will guide a robot, classical control theory aspects like stability, convergence, and response under disturbance of the motion generator need to be considered. These don't refer to the lower-level controller but to the learned model of the motion generator and its reference signal output. These are requirements that can be a deal breaker when it comes to safety-critical applications: A black box LfD policy cannot yield mathematical proof for the motion generator's stability or any other characteristic.

All these facets, type of demonstration, choice of state, action, and control-related requirements, must be considered when choosing from the broad set of tools that the machine learning community has to offer.

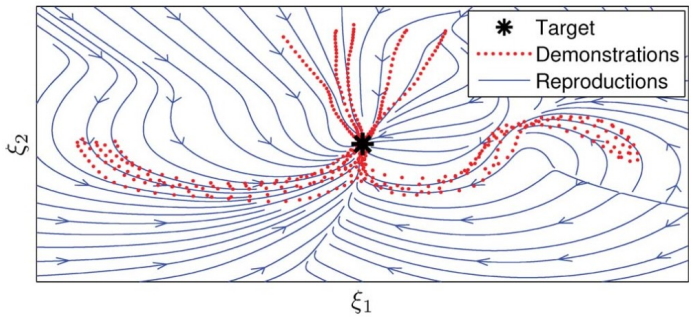


Figure 2.7. 2D visualization of a learned DS [5].

One way to group LfD works is whether or not the systems depend on time. A major advantage of time-invariant (autonomous) approaches is the ability to adapt to temporal perturbation. As the models are queried only with the current state, perturbations of the robot position due to interactions or errors in state estimation don't require re-planning. Non-autonomous systems, on the other hand, are not so robust to interruptions in their motions because of their time dependency, they require a heuristic on how to proceed after an interruption or perturbation.

As outlined in the previous subsections, higher-order derivative reference signals other than velocity and acceleration are useful when controlling elastic joint robots. State-of-the-art approaches can be extended to obtain higher-order reference signals by (1) finding the analytical time derivative of the model, or (2) extending the input state to contain not only the position but also velocity, acceleration, and jerk. Computing the analytical time derivative of a model is only possible if the model's structure allows for it. NN models for example cannot be derived w.r.t. time. Extending the state to e.g. velocity $\xi = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ results in an output vector of $\dot{\xi} = \begin{bmatrix} \dot{x}_d \\ \ddot{x}_d \end{bmatrix}$. This approach can be challenging as it must be ensured that $\ddot{x}_d$ really is a time derivative of $\dot{x}_d$.

2.3.1 Gaussian Mixture Model (GMM)

A GMM is a probabilistic model that represents a distribution as a weighted sum of multiple Gaussian distributions. It's powerful due to being both explainable and expressive while being relatively simplistic. A Gaussian distribution is given by

$$\mathcal{N}(x \mid \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)\right). \tag{2.12}$$

The density function of a GMM is given by

$$p(\xi) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\xi \mid \mu_k, \Sigma_k). \tag{2.13}$$

To output a valid probability function, π_k must satisfy

$$\sum_{k=1}^{K} \pi_k = 1, \quad \text{and} \quad 0 \leq \pi_k \leq 1. \tag{2.14}$$

The optimization parameters of the model are the mixing terms π_k, the means of each Gaussian μ_k, and the covariance matrices Σ_k. The expectation maximization (EM) algorithm is most commonly used to find the maximum likelihood estimate of the parameters. It consists of two steps that are executed iteratively until convergence or a maximum iteration threshold is reached:

Expectation Step: Each data point x_i is assigned a probability of belonging to the k^{th} Gaussian based on the current estimates of π_k, μ_k, Σ_k

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}. \tag{2.15}$$

Maximization Step: Given the updated soft clustering, the parameters are updated

$$\pi_k = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ik}, \tag{2.16}$$

$$\mu_k = \frac{\sum_{i=1}^{N}\gamma_{ik}x_i}{\sum_{i=1}^{N}\gamma_{ik}}, \tag{2.17}$$

$$\Sigma_k = \frac{\sum_{i=1}^{N}\gamma_{ik}(x_i-\mu_k)(x_i-\mu_k)^\top}{\sum_{i=1}^{N}\gamma_{ik}}. \tag{2.18}$$

2.4 Orientation Parametrization

The orientation of an object in the 3D Cartesian space belongs to the Special Orthogonal group called SO(3). There are four main ways to parameterize SO(3): rotation matrices, Euler angles, axis angles, and quaternions. Rotation matrices are 3x3 orthonormal matrices. They are singularity-free, not intuitive to understand, and over-parametrized as they describe 3 DoF with 9 parameters. Euler angles represent 3D rotations as a sequence of three rotations around principal axes. There are different conventions on the order of rotations. They are intuitive to understand and a minimal representation, but suffer from the gimbal lock phenomenon. A gimbal lock occurs when two of the three rotation axes align, causing a loss of one DoF in the rotation system, resulting in the inability to rotate around a certain axis. Furthermore, they are not unique, different sets of Euler angles can represent the same orientation. Switching between these representations causes discontinuities. The Axis-angles representation is the pair of a unit vector e, indicating the axis around which the rotation occurs, and an angle, determining how much the object is rotated around that axis ψ. It has no gimbal lock issue but has singularities around 0 and 360 degrees. Quaternions are 4D complex numbers that can efficiently represent 3D rotation when normalized. Even though quaternions aren't intuitive, they are widely used in robotics and computer graphics. Normalized quaternions (magnitude = 1) are called unit quaternions (UQ) and take the form

$$\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}, \tag{2.19}$$

with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$ and $a, b, c, d \in \mathbb{R}$. They represent a double cover of SO(3) ($\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation). UQ are

- numerically stable, as renormalization is a simple operation that doesn't introduce significant new errors
- smoothly interpolable (e.g., Slerp [23])

- a compact representation (4 parameters)
- singularity free
- unique (with the exception that $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation as quaternions represent a double cover of SO(3))
- computationally efficient

The compact representation is especially useful for machine learning applications as fewer parameters generally ease learning. The product of two quaternions $\mathbf{q}_1 = a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}$ and $\mathbf{q}_2 = a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}$ is given by

$$\begin{aligned}\mathbf{q}_1 \circ \mathbf{q}_2 = &(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) \\ &+ (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\mathbf{i} \\ &+ (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\mathbf{j} \\ &+ (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\mathbf{k}.\end{aligned} \tag{2.20}$$

The inverse of a UQ is given by flipping the signs of its vector part

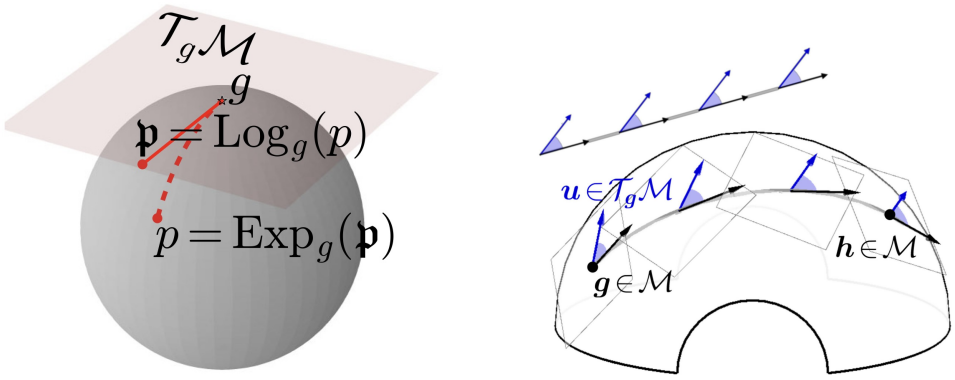
$$\mathbf{q}^{-1} = \mathbf{q}^* = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k}. \tag{2.21}$$

Rotations can be concatenated by $\mathbf{q}'_s = \mathbf{q}_s\mathbf{q}_r$, where $\mathbf{q}_r$ is expressed in the frame of $\mathbf{q}_s$. $\Delta\mathbf{q}^b_{12} = \mathbf{q}_1^*\mathbf{q}_2$ is the rotation needed to rotate $\mathbf{q}_1$ onto $\mathbf{q}_2$ with respect to the body frame. $\Delta\mathbf{q}^w_{12} = \mathbf{q}_2\mathbf{q}_1^*$ is the rotation needed to rotate $\mathbf{q}_1$ onto $\mathbf{q}_2$ with respect to the world frame.

2.5 Riemannian Manifolds

In general, a manifold is a topological space that locally resembles the Euclidean space but may have a more complex global structure. A Riemannian manifold $\mathcal{M}$ is a real, smoothly varying manifold with a positive definite metric tensor that defines the intrinsic curvature and torsion of the manifold. Many spaces that are important in robotics can be interpreted as Riemannian manifolds like the Euclidean space, joint configuration space (torus), the special orthogonal group SO(3), the special Euclidean group SE(3), and the symmetric positive definite matrices S^d_{++}.

Most important for this thesis are the concepts of distance and derivatives on the Riemannian Manifold. Similar to straight lines in Euclidean space, manifolds have the notion of geodesics which are smooth curves defining the minimum distance between two points on the manifold. They have the property that their acceleration



(a) Manifold mappings [24]. **(b)** Parallel Transport [25].

Figure 2.8. Visualization of parallel transport and mapping between Manifold and TS.

vector is always perpendicular to the manifold's tangent space at each point and that their second covariant derivative along the curve vanishes.

Derivatives on Riemannian manifolds are defined by a vector space, called tangent space (TS). The TS $T_{\mathbf{p}}\mathcal{M}$ is spanned by all possible tangent vectors passing through $\mathbf{p} \in \mathcal{M}$, e.g. for the SO(3) sphere, the TS at $T_{\mathbf{p}}\mathcal{M}$ would be a plane, touching the $\mathcal{M}$ at $\mathbf{p}$. The exponential map $\mathrm{Exp}_{\mathbf{p}_0} T_{\mathbf{p}_0}\mathcal{M} \rightarrow \mathcal{M}$ maps a point $\mathfrak{p}$ in the TS of $\mathbf{p}_0$ back to a point $\mathbf{p}$ on the manifold. $\mathbf{p}$ lies along the direction between $\mathbf{p}_0$ and $\mathfrak{p}$ and has a geodesic distance to $\mathbf{p}_0$ equal to the norm of $\mathfrak{p}$. The inverse of that mapping is called the logarithmic map $\log_{\mathbf{p}_0} : \mathcal{M} \rightarrow T_{\mathbf{p}_0}\mathcal{M}$, see Fig. 2.8b for a visualization. Parallel transport moves a vector between tangent spaces in a way that the angle between the vector and the geodesic, spanned between the origins of the two tangent spaces, stays constant $\Gamma_{\mathbf{p}\rightarrow\mathbf{h}} : T_{\mathbf{p}}\mathcal{M} \rightarrow T_{\mathbf{h}}\mathcal{M}$, see Fig. 2.8a.

Chapter 3

Related Work

This chapter presents some of the existing work related to LfD and determines if they are suitable for the reaching motion application. Given the extent of LfD-related works, this chapter focuses mainly on DS-based LfD. For a general overview, the interested reader is referred to [26] and [27]. In the following, particular emphasis is placed on the ability of each method to be extended to higher-order derivative signals.

3.1 Nonlinear Mixture of Linear Systems

One of the most recognized approaches in learning an autonomous DS from demonstration is the SEDS [5], which uses a GMM to model the joint probability of state and velocity demonstration data. The authors write the conditional probability of the velocity given the state as a non-linear combination of linear systems

$$\dot{\xi} = \sum_{k=1}^{K} h_k(\xi)(A_k\xi + b_k). \tag{3.1}$$

For this, stability constraints utilizing the LF $V(x) = (\xi - \xi^*)^\top(\xi - \xi^*)^\top$ were derived. However, due to the simplistic LF, the expressiveness of the model is limited as discussed in Section 4.1.3.

To overcome the limited expressiveness of SEDS, the authors in [28] fit the Gaussians of the GMM in a way that the largest eigenvalue matches the direction of the trajectory and that data points close to each other in space, but with very different velocity values, are represented by different Gaussians. This enables a two-stage optimization with a more expressive LF. While this work has the advantage of choosing the number of Gaussians automatically and exhibiting very good expressiveness, it tends to use a lot of Gaussians which becomes a problem when having demonstration data covering the whole workspace. In this case, the number

of Gaussians can grow so large that real-time inference isn't possible anymore.

The work in [29] extends [28] to accommodate SO(3) with orientation data represented by UQ. It projects all trajectories into the TS of the attractor and models the demonstration data with a GMM using EM. Then, using the fixed mixing term of the GMM, the A_k matrices are optimized as in [28]. The model outputs a vector in the TS of the attractor that needs to be mapped to an angular velocity. Due to its simplicity and similarity to SEDS, the method has been implemented in MATLAB and tested in both a MATLAB and a SIMULINK simulation. See Section 4.5 for details.

3.1.1 Control Lyapunov Functions

In [30] the authors surpass the expressiveness of SEDS by introducing Control Lyapunov Function-based Dynamic Movements (CLF-DM). The approach uses a weighted sum of asymmetric quadratic functions (WSAQF) to learn an LF that later helps to ensure the stability of a DS learned with an arbitrary regression technique. To take advantage of the expressive arbitrary regression technique and still guarantee stability, a correction signal is computed online and added to the model output, in case the model doesn't comply with the previously learned LF. However, the learning of the more complex LF and online computation of the correction signal introduces additional complexity and hyperparameter selection which do not justify the expected increase in performance as the reaching motion doesn't benefit much from a more expressive LF.

3.2 Diffeomorphic Functions

Instead of making the LF more expressive or computing an online correction signal, τ-SEDS [31] tackles the reproducibility accuracy problem of SEDS by using a diffeomorphic transformation. A diffeomorphic transformation is a continuously differentiable, bijective function guaranteed to have an inverse. The task space is transformed in such a way that in the new space, the transformed demonstration data can be captured with a simple LF and hence SEDS can be used to learn the dynamics in the transformed space. The learned system is then back-transformed to the original task space for inference. While this approach is promising, the increased expressiveness does not justify the additional computational complexity of the reaching motion.

Similarly, [32] uses diffeomorphic transformations to map a nominal DS ($-\dot{\xi} = \xi$) onto the training trajectory. The nominal DS is constructed with as many terms as there are data points in the demonstration. Although the approach's expressiveness

is high, scaling to multiple demonstrations throughout the whole workspace for real-time inference is infeasible.

3.2.1 Riemmanian Manifolds

In contrast to [31] and [32] that learn the parameters of a diffeomorphic transformation between Euclidean spaces, [33] learns a stable dynamical system on a Riemannian Manifold (SDS-RM) by mapping of a basic stable DS ($\dot{x} = -x$) to the demonstration data on a Riemannian Manifold. Besides Euclidean demonstration data, this approach lends itself to the $SO(3)$ space of UQ and SPD matrices.

Leveraging Riemannian Manifolds as well is the work [34], which uses NNs to model the diffeomorphism. The authors propose a NN as a combination of bijective and continuously differentiable functions. In [35], the authors leverage differential geometry to learn an embedding with a simple 2-layer feed-forward model on a Riemannian manifold. This results in a second-order system, already a step towards higher-order reference signals. However, as for many NN approaches, the extrapolation capabilities of the models are limited; the behavior in regions away from demonstration data is unpredictable and needs to be treated explicitly. This is a major disadvantage of these systems.

3.3 Gaussian Processes

Both works in [36] and [37] are based on the idea that regions of the state space without demonstration data bear a risk of undesired behavior. The authors thus propose an uncertainty-aware Gaussian Process state space model (GP-SSM) in combination with a CLF that guides the robot to regions with available training data. Even though the method can perfectly reproduce the data, which is beneficial when trying to extend the state space to accommodate higher-order derivatives, the computation of the correction signal requires the computation of a trajectory roll-out. This is seen as too much of a computational burden for the real-time application.

3.4 Neural Networks

As one of the first works [38] learns a LF parametrized by a NN from data. The simplicity of the network however limits the expressiveness. A more recent work [39] proposes a valid LF encoded by a NN and then trains a GP to create a stable DS. The proposed LF is modeled with a radial basis function neural network (RBFNN), which is implemented as a weight matrix multiplied by a vector of basis functions. Its main advantage is that this NN is differentiable. In [40], a combination of attention and diffeomorphic layers is designed to train an expressive LF. These works, although

showing good demonstration reproducibility, don't fulfill the requirements of the thesis as the optimization significantly depends on the selection of hyperparameters, and an extension to higher-order derivatives is difficult as the architectures of the models only guarantee a first-order analytical derivative. For the control of the elastic joint robot, however, higher-order derivatives are needed.

3.5 Dynamic Motion Primitives (DMP)

Dynamic motion primitives [41] are different than a DS approach in that they superpose a linear, stable spring damper system with a forcing term to reproduce the desired behavior. Many extensions and different formulations have been presented over the years, see [42] for a comprehensive overview. Even reference signals up to the snap level have been developed in [43]. However, for all these approaches, the forcing term is always time-dependent, making the model non-autonomous and thus not robust to an unstructured environment.

3.6 Probabilistic Approaches

The probabilistic motion primitive [44] encodes a distribution of trajectories via a weight vector and basis functions. A Gaussian distribution is placed over the weights resulting in a hierarchical Bayesian model. While this approach enables useful features like via-points, the basis functions scale poorly with the dimension of the state space which would be a problem for an extended state that is needed to obtain higher-order reference signals. Furthermore, the implicit time dependency of the basis functions renders the model non-autonomous.

The work in [45] improves the probabilistic motion primitives in that it is autonomous and replaces the basis functions with kernels, reducing the computational burden of higher dimensional state spaces. However, an extended state for this particular model is difficult as the structure of the model doesn't allow for additional constraints that ensure the consistency of the output: the desired acceleration should in fact be a time derivative of the desired velocity.

3.7 Conclusion

To the best of my knowledge, there is no time-invariant LfD method providing reference signals of higher order other than acceleration. This highlights the necessity of this work, as the ESP controller benefits from higher-order derivatives to compensate for non-linear terms when tracking a reference joint trajectory.

Chapter 4

Method

This chapter outlines the methodology of the designed motion generator. First, the SEDS approach is introduced in detail and the time derivatives to obtain acceleration and jerk references are presented. Second, the motivation behind the covariance eigenvalue constraint, developed to prevent model output discontinuities, is discussed. Additionally, the adopted orientation motion generator is described. The chapter concludes with the simulation results of both models and an ablation study of the different hyperparameters.

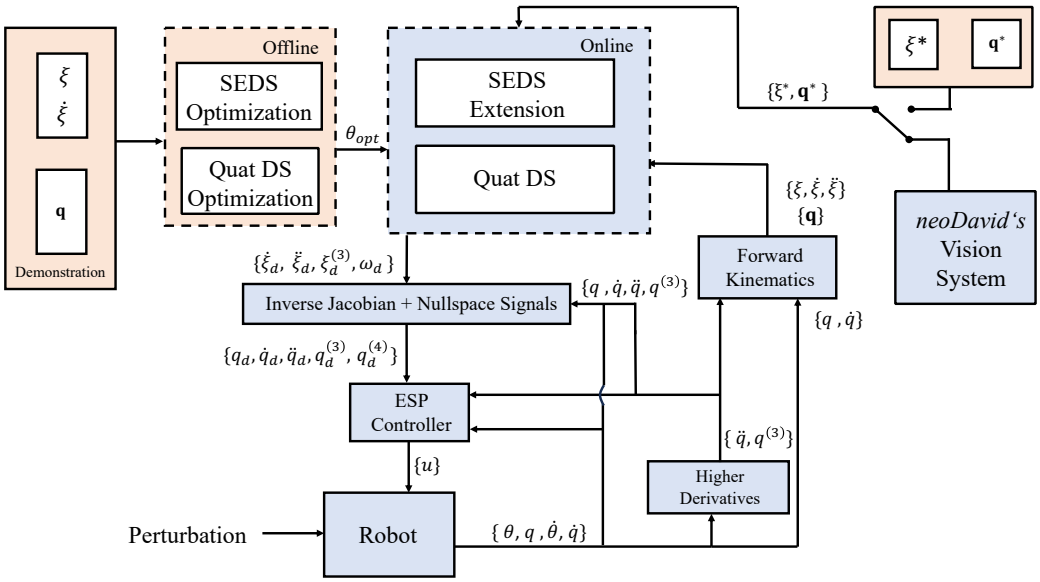


Figure 4.1. Visualization of the motion generator learning and control system with $\dot{\xi}_d, \ddot{\xi}_d, \xi_d^{(3)}$ as model outputs, ω_d as desired angular velocity of the orientation motion generator QuatDs, $\mathbf{q}$ as UQ, θ_{opt} as optimization parameters, $\mathbf{q}^*$ and ξ^* as targets, u as the ESP control input to the robot and q and θ as joint and motor coordinates respectively.

The workflow of the approach is described by 1) collecting demonstration data

in the form of $D = \{\xi_n, \dot{\xi}_n\}_{n=1}^N$, with N as the number of data points, $\xi = x - x^*$, $x \in \mathbb{R}^3$ as the end-effector position and $Q = \{\mathbf{q}^n\}_{n=1}^N$ with $\mathbf{q}$ as the world frame orientation in UQ. Details on demonstration data are given in Section 5.1. 2) Offline training of the motion generators for orientation (Quat DS), see Section 4.5 and SEDS Section 4.1. 3) Given a target ξ^*, $\mathbf{q}_{att}$, querying the motion generator with the current state to obtain cartesian reference signals and 4) mapping the cartesian signals to the joint space and adding nullspace control signals, and 5) passing the joint space references to the ESP Controller. See Fig. 4.1 for an overview.

4.1 Stable Estimator of Dynamical Systems (SEDS)

SEDS [5] approximates the demonstration data with a multivariate GMM

$$P(\xi, \dot{\xi}) = \sum_{k=1}^{K} P(k) P(\xi, \dot{\xi} \mid k), \tag{4.1}$$

where K is the number of Gaussians used in the GMM, $0 < P(k) < 1$ is the prior, and $P(\xi, \dot{\xi} \mid k) = \mathcal{N}(\xi, \dot{\xi} \mid \mu^k, \Sigma^k)$ is the multivariate Gaussian with

$$\mu^k = \begin{pmatrix} \mu_\xi^k \\ \mu_{\dot{\xi}}^k \end{pmatrix}, \quad \Sigma^k = \begin{pmatrix} \Sigma_{\xi\xi}^k & \Sigma_{\xi\dot{\xi}}^k \\ \Sigma_{\dot{\xi}\xi}^k & \Sigma_{\dot{\xi}\dot{\xi}}^k \end{pmatrix} \in \mathbb{R}^{6\times 6}. \tag{4.2}$$

In [46], the authors introduced the Gaussian Mixture Regression for imitation learning, where the distribution in (4.1) is conditioned on the state ξ and results in:

$$\dot{\xi} = \sum_{k=1}^{K} \frac{P(k) P(\xi \mid k)}{\sum_{i=1}^{K} P(i) P(\xi \mid i)} \left(\mu_{\dot{\xi}}^k + \Sigma_{\dot{\xi}\xi}^k (\Sigma_\xi^k)^{-1} (\xi - \mu_\xi^k) \right). \tag{4.3}$$

Through a change of variables, this can be simplifyed to

$$\dot{\xi} = \sum_{k=1}^{K} h^k(\xi)(A^k \xi + b^k), \tag{4.4}$$

where

$$\begin{cases} A^k = \Sigma_{\dot{\xi}\xi}^k (\Sigma_{\xi\xi}^k)^{-1}, \\ b^k = \mu_{\dot{\xi}}^k - A^k \mu_\xi^k, \\ h^k(\xi) = \frac{P(k)P(\xi|k)}{\sum_{i=1}^{K} P(i)P(\xi|i)}. \end{cases} \tag{4.5}$$

This allows for on the model, as the current state can now be used as input, with the output providing the desired velocity. The nonlinear weighting terms $h^k(\xi)$ control the relative influence of each Gaussian on the final prediction and act as an

activation function for each Gaussian. Intuitively, $h^k(\xi)$ increases as ξ approaches μ^k, with the activation profile of $h^k(\xi)$ depending on $\Sigma^k_{\xi\xi}$. Due to this weighting, it is possible to create nonlinear behaviors from the otherwise linear Gaussians.

4.1.1 Stability of SEDS

Two conditions on A^k and b^k to render the system in (4.4) stable:

$$\begin{cases} (\mathbf{a})\ b^k = -A^k\xi^* & \forall k = 1...K, \\ (\mathbf{b})\ A^k + (A^k)^\top \prec 0 & \forall k = 1...K. \end{cases} \tag{4.6}$$

For the LF of $V(\xi)$ proposed by the authors

$$V(\xi) = \frac{1}{2}(\xi - \xi^*)^\top(\xi - \xi^*) \quad \forall \xi \in \mathbb{R}^d, \tag{4.7}$$

it is clear that due to its quadratic nature, condition (2.11)(**a**) of the theorem is satisfied. Condition (2.11)(**b**) is satisfied due to

$$\begin{aligned}
\dot{V}(\xi) &= \frac{dV}{dt} = \frac{dV}{d\xi}\frac{d\xi}{dt} && (4.8)\\
&= \frac{1}{2}\frac{d}{d\xi}\left((\xi - \xi^*)^\top(\xi - \xi^*)\right)\dot{\xi} && (4.9)\\
&= (\xi - \xi^*)^\top\dot{\xi} && (4.10)\\
&= (\xi - \xi^*)^\top \underbrace{\sum_{k=1}^{K} h^k(\xi)(A^k\xi + b^k)}_{=\dot{\xi}\ \text{(see (4.4))}} && (4.11)\\
&= (\xi - \xi^*)^\top \sum_{k=1}^{K} h^k(\xi)(A^k(\xi - \xi^*) + \underbrace{A^k\xi^* + b^k}_{=0\ \text{(see (4.6))}} && (4.12)\\
&= (\xi - \xi^*)^\top \sum_{k=1}^{K} h^k(\xi)(A^k(\xi - \xi^*) && (4.13)\\
&= \sum_{k=1}^{K} \underbrace{h^k(\xi)}_{h^k(\xi)>0}\ \underbrace{(\xi - \xi^*)^\top A^k(\xi - \xi^*)}_{<0\text{(see (4.6))}} && (4.14)\\
&< 0 \quad \forall \xi \in \mathbb{R}^d, \quad \xi \neq \xi^*. && (4.15)
\end{aligned}$$

Therefore enforcing the stability conditions in (4.6) guarantees the stability of the system, meaning that if the reference signals are followed, the end-effector always converges to the target.

4.1.2 SEDS optimization

With the stability conditions developed, the question remains on how to choose the parameters of the GMM, so that the resulting system in (4.4) can best reproduce the demonstration data. As a reminder, the GMM is given by

$$P(\xi^{t,n}, \dot{\xi}^{t,n} \mid \theta) = \sum_{k=1}^{K} P(k) P(\xi^{t,n}, \dot{\xi}^{t,n} \mid k), \tag{4.16}$$

where θ is the optimization parameter vector given by $\{\pi^1 \dots \pi^K;\ \mu_\xi^1 \dots \mu_\xi^K;\ \Sigma^1 \dots \Sigma^K\}$, the vector that holds the parameters of the GMM. In the original work, the authors proposed two cost functions: the log-likelihood and the mean square error (MSE). Similar to the author's findings, the log-likelihood loss (4.17a) yielded better results for the data collected during the thesis, due to it emphasizing the direction of the output velocity vector more than the MSE cost function. Therefore, the log-likelihood function was chosen to train the model. The loss function is optimized under the constraint for stability given in (4.6) as well as the constraints to guarantee a valid GMM (2.14). The optimization problem thus becomes

$$\min_{\boldsymbol{\theta}} \qquad J(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T^n} \log P(\xi^{t,n}, \dot{\xi}^{t,n} \mid \theta), \tag{4.17a}$$

$$\text{subject to} \quad \begin{cases} (\mathbf{a})\ b^k = -A^k \xi^*, \\ (\mathbf{b})\ A^k + (A^k)^\top \prec 0, \\ (\mathbf{c})\ \Sigma^k \succ 0, \quad \Sigma^k = (\Sigma^k)^\top, & \forall k = 1 \dots K \\ (\mathbf{d})\ 0 < \pi^k \leq 1, \\ (\mathbf{e})\ \sum_{k=1}^{K} \pi^k = 1, \end{cases} \tag{4.17b}$$

with the last three constraints ensuring that the resulting GMM is valid, namely, that Σ^k is a proper covariance matrix and $P(\xi^{t,n}, \dot{\xi}^{t,n} \mid \theta) < 1$ in (4.16). To increase efficiency and optimization speed, many constraints can be made implicit by a change of variables. First, (**a**) in (4.17b) is made implicit by not optimizing for b^k and substituting $-A^K \xi^*$ after the training. By changing π^k to $\tilde{\pi}^k = \ln\left(\pi^k\right)$ and recovering it after the optimization with $\pi^k = e^{\tilde{\pi}^k} / (\sum_{i=1}^{K} e^{\tilde{\pi}^i})$, constraint (**d**) and (**e**) are implicitly enforced as the exponential function is always positive and the division ensures the sum of all π^k is equal to 1. Lastly, by defining $L^k = Chol(\Sigma^k)$ as the Cholesky decomposition which always exists since Σ^k is symmetric positive definite and recovering Σ^k after the optimization like this: $\Sigma^k = L^k (L^k)^\top$, the constraint (**c**) is also satisfied since this multiplication will always give a symmetric positive definite

matrix. The new optimization parameters are $\boldsymbol{\theta} = \{\tilde{\pi}^1 \dots \tilde{\pi}^k, \mu_\xi^1 \dots \mu_\xi^k, L^1 \dots L^k\}$, where now only constraint (**b**) needs to be enforced by the optimizer.

The resulting optimization problem is non-convex and is solved with non-linear programming. The existence of a solution is always guaranteed as one can run EM (2.15) over the demonstration data, enforce (**b**) by first taking the absolute values of Σ^k and then negating $\Sigma^k_{\dot{\xi}\xi} = -\Sigma^k_{\dot{\xi}\xi}$ and $\Sigma^k_{\xi\dot{\xi}} = -\Sigma^k_{\xi\dot{\xi}}$ and enforce (**a**) by choosing $\mu^k_{\dot{\xi}} = \Sigma^k_{\dot{\xi}\xi}(\Sigma^k_{\xi\xi})^{-1}(\mu^k_\xi - \xi^*)$. Given that the problem is non-convex, a guarantee for an optimal solution doesn't exist and depends on the initialization. To initialize the problem, k-means is run and based on the resulting means, the Gaussian Covariances are computed as an intialization for EM. Initialized by k-means, EM is run to obtain an initial guess for the optimizer. K is empirically chosen based on the results discussed Section 4.5. The solver is provided with the analytical derivatives of the cost function and the constraints w.r.t. the optimization parameter θ to decrease training time. See the corresponding technical report [47] for details.

4.1.3 Limitations

Due to the presented quadratic LF in (4.7), the SEDS output $\dot{\xi}$ has to full-fill $\dot{V}(\xi) = \frac{\partial V}{\partial \xi}\frac{\mathrm{d}\xi}{\mathrm{dt}} = \frac{\partial V}{\partial \xi}\dot{\xi} < 0$, except for $\xi = \xi^*$. The dot product of two vectors is only negative, if their angle is obtuse, meaning that their absolute angle is greater than 90 degrees. Since $V(\xi)$ is a perfect bowl with its lowest point at the attractor, $-\frac{\partial V}{\partial \xi}$ always points directly towards the target. This implies that for the angle between $\frac{\partial V}{\partial \xi}$ and $\dot{\xi}$ to be obtuse, the direction of $\dot{\xi}$ is restricted to result in a decrease in the distance to the target. This hinders the model from representing highly non-linear motions because, because it can't ever move away from the target.

4.2 Extended State

One possible way to achieve acceleration or even higher-order signals with SEDS is to extend the input state to include both position and velocity

$$\xi = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}. \tag{4.18}$$

This would result in an output of both desired velocity and desired acceleration. However, when naively extending the state without additional constraints in the optimization, it is not guaranteed that the output velocity and the output acceleration are physically consistent, meaning that the desired acceleration is the time derivative of the desired velocity. To develop the constraints that enforce the physical consistency, one can impose a matrix structure for A_k and b_k to guarantee a valid

second-order dynamical system

$$\underbrace{\begin{pmatrix}\dot{x}\\ \ddot{x}\end{pmatrix}}_{\dot{\xi}} = \sum_{k=1}^{K} h^k(\xi,\dot{\xi}) \left[\begin{pmatrix} 0 & 1 \\ A_1^k & A_2^k \end{pmatrix} \underbrace{\begin{pmatrix} x \\ \dot{x} \end{pmatrix}}_{\xi} + \begin{pmatrix} 0 \\ b_1^k \end{pmatrix} \right]. \tag{4.19}$$

Constraining the top-left submatrix of A_k to be zero and the top-right submatrix of A_k to be the identity, along with the first half of b_k to be zero as shown in (4.19) would guarantee a physically consistent output. This idea was also followed in [48] and [6] in the context of a robotic catching application. Even though this method can

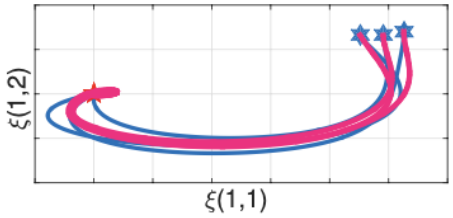


Figure 4.2. Trajectory rollouts presented in [6] for the extended state approach.

reproduce demonstration data reasonably well, it exhibits rapid direction changes [6]. Furthermore, the models often overshoot, as can be seen in Fig. 4.2. Additionally, the models are trained on acceleration data, and imperfections in reproducing the acceleration data will propagate to the velocity level and cause unexpected behaviors. This effect is likely to increase for higher orders, therefore the approach is not further investigated.

4.3 Time Derivatives

The SEDS itself is time-independent, as its output is solely dependent on the state. However, while the model is time-independent, the underlying DS is time-dependent and can be derived w.r.t. time. One can therefore train the model on velocity data only and derive its expression to obtain the additional acceleration and jerk reference signals. This further guarantees that the signals are physically consistent.

In the following, the time derivatives of the 1st and 2nd order are presented. Since $\dot{\xi}$ depends on ξ and implicitly on time, derivatives w.r.t. ξ are partial derivatives and derivatives w.r.t. time are total derivatives and therefore denoted by $\frac{\partial}{\partial \xi}$ and $\frac{\mathrm{d}}{\mathrm{dt}}$ respectively.

4.3.1 Acceleration

Using the product rule to derive the SEDS expression in (4.4) one obtains

$$\ddot{\xi} = \sum_{k=1}^{K} \frac{\mathrm{d}}{\mathrm{dt}} \left(h^k(\xi) \right) \; (A^k \xi + b^k) + h^k(\xi) \; \frac{\mathrm{d}}{\mathrm{dt}} \left(A^k \xi + b^k \right). \tag{4.20}$$

Continuing using the chain rule for $\frac{\mathrm{d}}{\mathrm{dt}} \left(h^k(\xi) \right)$ and the time derivative for $\left(A^k \xi + b^k \right)$ one obtains

$$\ddot{\xi} = \sum_{k=1}^{K} \frac{\partial \, h^k(\xi)}{\partial \, \xi} \; \dot{\xi} \; (A^k \xi + b^k) + h^k(\xi) \; A^k \; \dot{\xi}. \tag{4.21}$$

The expression for $h^k(\xi)$ is given in (4.5). $P(\xi \mid k)$ is the conditional probability distribution function corresponding to the k-th Gaussian function. Hence, $h^k(\xi)$ is a continuous and continuously differentiable function. Applying the quotient rule gives

$$\frac{\partial \, h^k(\xi)}{\partial \, \xi} = \frac{\frac{\partial}{\partial \, \xi} \left(P(k) P(\xi \mid k) \right) \; \sum_{i=1}^{K} P(i) P(\xi \mid i) - P(k) P(\xi \mid k) \; \frac{\partial \, \sum_{i=1}^{K} P(i) P(\xi \mid i)}{\partial \, \xi}}{\left(\sum_{i=1}^{K} P(i) P(\xi \mid i) \right)^2}. \tag{4.22}$$

To compute $\frac{\partial}{\partial \, \xi} \left(P(k) P(\xi \mid k) \right)$, one considers $P(k) \in \mathbb{R}$ that doesn't depend on ξ and thus is a constant in the derivative and $P(\xi \mid k)$, which is the multivariate Gaussian distribution

$$P(\xi \,|\, k) = \frac{1}{(2\pi)^{d/2} |\Sigma_{\xi\xi}^k|^{1/2}} \exp \left[-\frac{1}{2} (\xi - \mu_\xi^k)^\top (\Sigma_{\xi\xi}^k)^{-1} (\xi - \mu_\xi^k) \right]. \tag{4.23}$$

Finally, using the symmetric property of $(\Sigma_{\xi\xi}^k)^{-1}$ for the derivative $\left(\frac{\partial \, x^\top A \, x}{\partial x} = 2 x^\top A \right)$ one obtains

$$\frac{\partial \, P(k) P(\xi \mid k)}{\partial \, \xi} = -P(k) P(\xi \mid k) (\xi - \mu_k)^\top (\Sigma_{\xi\xi}^k)^{-1}. \tag{4.24}$$

4.3.2 Jerk

Considering the expressions for $\ddot{\xi}$ (4.21), $h^k(\xi)$ (4.22) and $P(\xi \,|\, k)$ (4.24) developed in the previous section, the jerk signal is given by

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{dt}} \ddot{\xi} \;=\; \xi^{(3)} \;=& \sum_{i=1}^{K} (A^k \xi + b^k) \left[\dot{\xi}^\top \left(\frac{\partial^2}{\partial \xi^2} h^k(\xi) \dot{\xi} \right) + \frac{\partial}{\partial \xi} h^k(\xi) \; \ddot{\xi} \right] \\ &+ 2 \left(\frac{\partial}{\partial \xi} h^k(\xi) \dot{\xi} \right) A^k \dot{\xi} + h^k(\xi) A^k \ddot{\xi}, \end{aligned} \tag{4.25}$$

where $\frac{\partial u^\top v}{\partial x} = u^\top \frac{\partial v}{\partial x} + v^\top \frac{\partial u}{\partial x}$ was used. The second derivative of $h(\xi)$ w.r.t. ξ is given by:

$$\begin{aligned}
\frac{\partial^2}{\partial \xi^2} h^k(\xi) \;=\; & \frac{\frac{\partial^2}{\partial \xi^2} P(k)P(\xi \mid k)}{\sum_{i=1}^{K} P(i)P(\xi \mid i)} - \frac{\frac{\partial}{\partial \xi}\left(P(k)P(\xi \mid k)\right) \frac{\partial}{\partial \xi}\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^\top}{\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^2} \\
& + 2\, \frac{P(k)P(\xi \mid k) \frac{\partial}{\partial \xi}\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right) \frac{\partial}{\partial \xi}\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^\top}{\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^3} \\
& - \frac{\frac{\partial}{\partial \xi}\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right) \frac{\partial}{\partial \xi}\left(P(k)P(\xi \mid k)\right)^\top}{\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^2} \\
& - \frac{P(k)P(\xi \mid k) \frac{\partial^2}{\partial \xi^2}\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)}{\left(\sum_{i=1}^{K} P(i)P(\xi \mid i)\right)^2}.
\end{aligned} \tag{4.26}$$

Finally, deriving $P(k)P(\xi \mid k)$ twice w.r.t. ξ gives

$$\frac{\partial^2}{\partial \xi^2} P(k)P(\xi \mid k) = P(k)P(\xi \mid k) \left[(\Sigma^k_{\xi\xi})^{-1}(\xi - \mu^k_\xi)(\xi - \mu^k_\xi)^\top (\Sigma^k_{\xi\xi})^{-1} - (\Sigma^k_{\xi\xi})^{-1} \right]. \tag{4.27}$$

4.3.3 Snap

The analytical 3rd-order derivation of $h_k(\xi)$, necessary to compute the desired snap, is a three-dimensional tensor. The necessary online computational complexity isn't justified for the reaching motion since control signals up to the Jerk level are enough to smoothly execute very fast motions without overshooting as shown in Chapter 5. Nonetheless, the snap, except for $h_k(\xi)^{(4)}$, is presented below. Future work might investigate the influence of the snap-signal by numerically approximating $h_k(\xi)^{(4)}$.

Starting from the Jerk expression (4.25) the single terms are derived one by one:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\left[2\left(\frac{\partial}{\partial\xi}\,h^k(\xi)\dot{\xi}\right)A^k\dot{\xi}\right] &= 2\left(\frac{\partial}{\partial\xi}\,h^k(\xi)\dot{\xi}\right)A^k\ddot{\xi} \\ &\quad + 2A^k\dot{\xi}\left[\frac{\partial}{\partial\xi}\,h^k(\xi)\ddot{\xi} + \dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\ddot{\xi}\right)\right]\end{aligned} \tag{4.28}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}(A^k\xi+b)\left(\frac{\partial}{\partial\xi}\,h^k(\xi)\ddot{\xi}\right) &= \left(\frac{\partial}{\partial\xi}\,h^k(\xi)\ddot{\xi}\right)A^k\dot{\xi} \\ &\quad + (A^k\xi+b)\left[\frac{\partial}{\partial\xi}\,h^k(\xi)\xi^{(3)} + \ddot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\dot{\xi}\right)\right]\end{aligned} \tag{4.29}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}(A^k\xi+b)\left[\dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\dot{\xi}\right)\right] &= \dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\dot{\xi}\right)A^k\dot{\xi} \\ &\quad + (A^k\xi+b)\left[\dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\ddot{\xi}\right)\right] \\ &\quad + (A^k\xi+b)\left[\left(\frac{\partial^3}{\partial\xi^3}\,h^k(\xi)\dot{\xi}\right)\dot{\xi}\right] \\ &\quad + (A^k\xi+b)\left[\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\ddot{\xi}\right)^\top\ddot{\xi}\right]\end{aligned} \tag{4.30}$$

$$\frac{\mathrm{d}}{\mathrm{dt}}h^k(\xi)\ddot{\xi} = \left(\frac{\partial}{\partial\xi}\,h^k(\xi)\dot{\xi}\right)A^k\ddot{\xi} + h^k(\xi)A^k\xi^{(3)}. \tag{4.31}$$

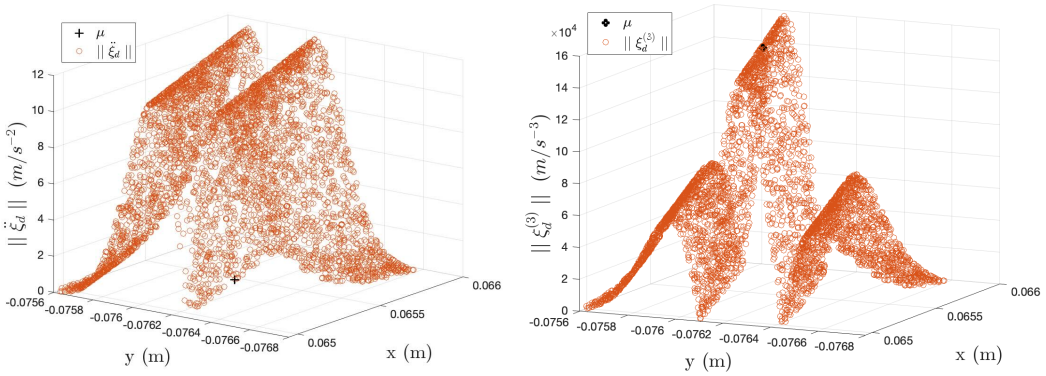
Grouping these terms gives the expression for the Snap signal:

$$\begin{aligned}\xi^{(4)} = 3\;&A^k\dot{\xi}\left[\left(\frac{\partial}{\partial\xi}\,h^k(\xi)\ddot{\xi}\right) + \dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\dot{\xi}\right)\right] \\ &+ 3A^k\ddot{\xi}\left(\frac{\partial}{\partial\xi}\,h^k(\xi)\dot{\xi}\right) \\ &+ (A^k\xi+b)\left\{\dot{\xi}^\top\left[\left(\frac{\partial^3}{\partial\xi^3}\,h^k(\xi)\dot{\xi}\right)\dot{\xi}\right] + \frac{\partial}{\partial\xi}\,h^k(\xi)\xi^{(3)}\right\} \\ &+ (A^k\xi+b)\left[3\;\dot{\xi}^\top\left(\frac{\partial^2}{\partial\xi^2}\,h^k(\xi)\ddot{\xi}\right)\right] \\ &+ h^k(\xi)A^k\xi^{(3)}.\end{aligned} \tag{4.32}$$

4.4 Upper and Lower Bounds of Covariance Eigenvalues

When providing the controller with the higher-order reference signals, the model outputs rapid acceleration (factor 12) and jerk (factor 10^4) with small changes of the state (within 1 mm^3), see Fig. 4.3a and Fig. 4.3b. Running such a model on

the robot results in seemingly unpredictable behavior and therefore needs to be addressed. This section will first discuss the cause of this irregular model behavior, explain why and where it occurs and lastly propose a solution to prevent it from happening, all without impacting model performance.



(a) Unbounded acceleration signal from the model derivative.

(b) Unbounded jerk signal from the model derivative.

Figure 4.3. Example for increase and decrease in acceleration and jerk for covariance with small eigenvalues. Z-axis has been replaced with model output magnitude.

4.4.1 Cause of Volatile Acceleration and Jerk Signal

The model output is the nonlinear sum of K linear systems, each created by a Gaussian distribution (4.4). Analysis of each Gaussian shows that only one Gaussian is responsible for the irregular behavior. In the term for that Gaussian, the inverse of the covariance $(\Sigma^k_{\xi\xi})^{-1}$ is ill-conditioned (Frobenius norm of $\approx 10^{10}$ versus the Frobenius norm of other covariances with ≈ 500). The inverse of the covariance $\Sigma^k_{\xi\xi}$ is proportionally present in the acceleration and jerk signal through the derivatives of $h^k(\xi)$ (4.22), (4.26) and the constant matrices A^k (4.5) and vectors b^k (4.5). If the inverse increases, so do these terms. With regard to the velocity the inverse is only present in A^k and b^k. However, it is balanced by the corresponding prior $P(k)$ being very small so that even for an ill-conditioned inverse, there is no significant increase in velocity, see Fig. 4.4. Other than the velocity, the extension to higher order derivatives doesn't have such a balancing mechanism because the model is trained solely on velocity and not acceleration and jerk data so that the contribution of $(\Sigma^k_{\xi\xi})^{-1}$ for the derivatives of $h^k(\xi)$ (4.22), (4.26) cannot be considered by the optimizer in the same way. Furthermore, $\dot{h}^k(\xi)$ and $\ddot{h}^k(\xi)$ are not bounded as $h^k(\xi)$, and can therefore increase uncontrollably, see Fig. 4.5.

Given that $h^k(\xi)$ is the activation function of the k^{th} Gaussian it is influenced by $\Sigma^k_{\xi\xi}$ and μ^k. The closer ξ is to μ^k, the larger $h^k(\xi)$. If the variance of a Gaussian

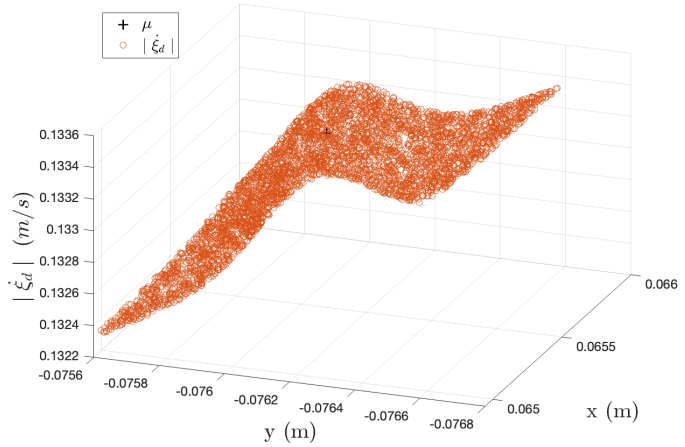


Figure 4.4. Example of the increase in velocity around the critical point.

decreases, the activation occurs within a smaller radius around μ^k. Furthermore, if the covariance matrix is ill-conditioned, $h^k(\xi)$ is activated only very close to μ^k and reaches its maximum when ξ coincides with μ^k (Fig. 4.5). Therefore the irregular behavior occurs around the mean of the Gaussian with the ill-conditioned covariance.

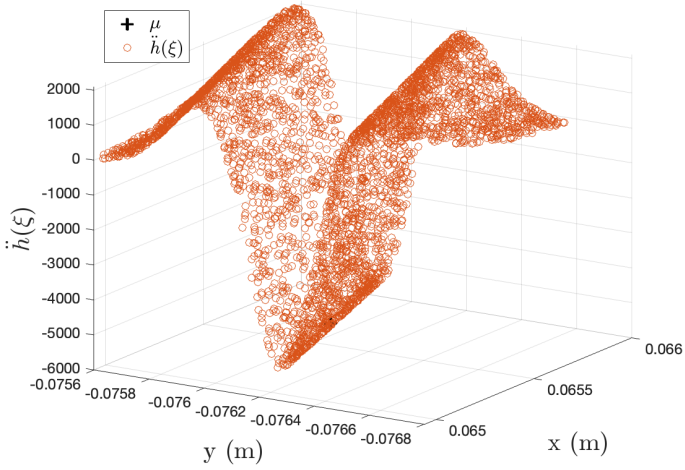


Figure 4.5. Example of rapid changes in $\ddot{h}^k(\xi)$ given small changes of ξ along the x and y axis for ill-conditioned gaussian.

4.4.2 Cause of Ill-Conditioned Covariance Matrices

Already the k-means initialization yields ill-conditioned covariances, due to the fact that trajectory data is very biased towards one of three spatial directions, resulting in a large eigenvalue along one direction and small eigenvalues along the other two. This is also true for some of the covariances after the EM step. However, even if the solver starts with an initial guess that doesn't contain an ill-conditioned

covariance, it might still converge to one. There is no optimization constraint to prevent this as the velocity data is still well reproduced even with an ill-conditioned covariance and the model's output of acceleration and jerk is not included in the cost function.

4.4.3 Preventing Ill-Conditioned Covariance Matrices

There are two ways for a symmetric positive definite matrix to be ill-conditioned which results in large elements in magnitude of its inverse: 1. near-singular matrices, caused by small eigenvalues of that matrix, 2. large conditioning number, caused by large ratio between largest and smallest eigenvalue.

The condition number for a positive definite matrix C is given as $\mathcal{K}(C) = \frac{\lambda_{max}}{\lambda_{min}}$. To prevent a large conditioning number an upper bound on the eigenvalues can be implemented. To prevent near-singular eigenvalues, a lower bound during the optimization is enforced that guarantees a minimum value for all eigenvalues. The optimizer uses the constraint formulation $c(x) \leq 0$, therefore the resulting optimization constraint for the i^{th} eigenvalue λ of the k^{th} Gaussian is

$$c_{\lambda k,i,lower} = -\lambda_{k,i} + \alpha \leq 0, \tag{4.33}$$

$$c_{\lambda k,i,upper} = \lambda_{k,i} - \beta \leq 0, \tag{4.34}$$

with α and β being the lower and upper bound, respectively.

Analytical derivative of constraints w.r.t. optimization parameters

The optimizer uses analytical derivatives of the cost function and the constraints w.r.t. the optimization parameters θ to speed up the optimization. The derivatives of the upper and lower bound eigenvalue constraints are equal and opposite, therefore only the derivatives for the upper bound $\frac{\partial c_{\lambda k,i,lower}}{\partial \theta}$ will be developed. The upper bound β is constant, thus zero in the derivative which results in $\frac{\partial c_{\lambda k,i,lower}}{\partial \theta} = \frac{\lambda_{k,i}}{\partial \theta}$. Out of all the parameters in θ, $\lambda_{k,i}$ only depends on the entries of $\Sigma^k_{\xi\xi}$, the derivative for all other parameters is zero. Furthermore, the parameters of Σ^k are stored in the cholesky decomposition L^k of Σ^k, see Section 4.1.2. As $\Sigma^k_{\xi\xi}$ is the upper left submatrix of Σ^k, the matrix $B^k(B^k)^\top = \Sigma^k_{\xi\xi}$, with B^k lower triangular is introduced to ease notation.

Starting from the general result in [49][1], the derivative of an eigenvalue w.r.t. the entries of a matrix, given that the matrix is symmetric and eigenvectors and

[1] $\partial\lambda_i = v_i^\top (\partial G)\, v_i$, with G being a symmetric matrix and v_i, λ_i the i^{th} eigenvector, eigenvalue respectively.

eigenvalues are smoothly varying is:

$$\frac{\partial \lambda_{k,i}}{\partial \Sigma^k_{\xi\xi,m,n}} = v_i^\top \left(\frac{\partial \Sigma^k_{\xi\xi}}{\partial \Sigma^k_{\xi\xi,m,n}}\right) v_i. \tag{4.35}$$

The derivative of $\Sigma^k_{\xi\xi} = B^k (B^k)^\top$, w.r.t. the optimization parameters is given by

$$\frac{\partial \Sigma_{\xi\xi}}{\partial B_{m,n}} = \frac{\partial B B^\top}{\partial B_{m,n}} = \frac{\partial B}{\partial B_{m,n}} B^\top + B \frac{\partial B^\top}{\partial B_{m,n}} = J_{m,n} B^\top + B J_{m,n}^\top, \tag{4.36}$$

where $\partial(X^\top) = \partial(X)^\top$ and $\frac{\partial B}{\partial B_{m,n}} = J_{m,n}$, with $J_{m,n}$ as the single-entry matrix, with a one at (m, n) and zero elsewhere. The k superscript is omitted for readability. Substituting (4.36) into (4.35), the derivative of the i^{th} eigenvalue λ of the k^{th} Gaussian w.r.t. the relevant optimization parameters given by $B_{m,n}$ is:

$$\frac{\partial \lambda_i^k}{\partial B^k_{m,n}} = (v_i^k)^\top \left(J_{m,n} (B^k)^\top + B^k J_{m,n}^\top \right) v_i^k, \tag{4.37}$$

where v_i^k is the corresponding eigenvector. This expression is passed to the solver.

4.5 Orientation Learning

So far only the position, but not the orientation of the robot has been considered. To enable orientation learning, this section illustrates the adopted approach presented in [29] for learning from demonstration in SO(3) via compact orientation representation of UQ.

4.5.1 Optimization

The approach [29] bears similarities to SEDS but in contrast to it, the GMM is not used to learn A_k and b_k, but only used for the optimization of the state-dependent mixing term $\gamma_k(\mathbf{q}^i)$ ($h^k(\xi)$ in SEDS), similar to LPV-DS [28]. A_k is optimized separately along with an additional matrix P to influence the gradient of the LF, while b_k is fixed to zero. The model is given by

$$(\mathfrak{q}_{att})^{des} = \sum_{i=1}^{M} \gamma_k(\mathbf{q}) A_k \log_{\mathbf{q}_{att}}(\mathbf{q}), \tag{4.38}$$

with $\log_{\mathbf{q}_{att}} : \mathcal{M} \to T_{\mathbf{q}_{att}}\mathcal{M}$ being the logarithmic mapping from the manifold to the TS spanned at $\mathbf{q}_{att}$, and $(\mathfrak{q}_{att})^{des}$ the desired orientation in $T_{\mathbf{q}_{att}}$.

The optimization process has two steps. First, the GMM is trained. However, it is only used to compute the mixing term $\gamma_k(\mathbf{q})$. Secondly, the A_k and P matrices

are optimized separately under the constraint (4.43), resulting in a non-convex optimization problem. The problem can be solved by non-linear constraint semi-definite programming. To optimize the GMM, EM is used. However, since EM assumes linearity of the space it operates on, it cannot be used directly as quaternions are naturally defined in the Hilbert Space. Given that UQ form a Riemannian manifold and that the TS of a Riemannian manifold is Euclidean, EM can be applied in the TS of a Riemannian manifold. All trajectories are therefore shifted to end at the same attractor. All trajectories are then mapped from the manifold into the TS of the attractor $\mathbf{q}_{att}$. With the data being in a Euclidean space represented by the TS of the attractor, EM can be applied which yields a GMM in $T_{\mathbf{q}_{att}}$. To compute the GMM on the manifold, each data point in the TS is assigned to a Gaussian based on the maximum likelihood. With this hard assignment, the GMM of the TS can be projected to the Riemannian manifold. To do this, each Gaussian mean on the manifold is computed according to

$$\tilde{\mu} = \underset{\mathbf{p}\in\mathbb{S}^3}{\operatorname{argmin}} \sum_{i=1}^{N} d(\mathbf{q}_i, \mathbf{p})^2, \tag{4.39}$$

with $\mathbf{p}, \mathbf{q}, \tilde{\mu} \in \mathcal{M}$ and $d(\mathbf{p}, \mathbf{q}) = \arccos\left(\mathbf{p}^\top \mathbf{q}\right)$, representing the shortest path between two points on the manifold, known as a geodesic. $\tilde{\mu}$ is found with the Frechet mean [50], which extends the mean from Euclidean Space to Riemannian manifolds. In practice, $\tilde{\mu}$ is efficiently computed with the iterative approach in [51]. Once the means are found, the covariance is computed, similar to the Euclidean space

$$\Sigma = \frac{1}{(N-1)} \sum_{i=1}^{N} \log_{\tilde{\mu}} \mathbf{p}_i \, \log_{\tilde{\mu}} \mathbf{p}_i^\top. \tag{4.40}$$

Having defined mean and covariance, the probability of observing a UQ on a Gaussian is given by

$$\mathcal{N}(\log_{\tilde{\mu}} \mid \mu = 0, \Sigma), \tag{4.41}$$

where the UQ is projected in $T_{\tilde{\mu}}$ and $\mu = 0$ since the Logarithmic mean w.r.t. itself is the origin of the TS. With this, $\gamma_k(\mathbf{q})$ is defined as the posterior probability of a quaternion $\mathbf{q}$ in the GMM

$$\gamma_k(\mathbf{q}) = \frac{\pi_k \mathcal{N}(\mathfrak{q}_{\tilde{\mu}_k} \mid \theta_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathfrak{q}_{\tilde{\mu}_k} \mid \theta_j)}, \tag{4.42}$$

where $\mathcal{N}(\tilde{\mu} \mid \theta_k)$ is the probability of observing $\mathbf{q}$ from the k-th Gaussian component parametrized by the mean and covariance matrix $\theta_k = \{\tilde{\mu}_k, \Sigma_k\}$.

It has to be noted that this approach can introduce errors in the learned model

because even though the TS is linear, the underlying geometry and the mapping between the manifold and TS aren't. This creates distortions if the optimization occurs in a single TS [25]. Future work might address this by deploying the EM for Riemannian manifolds similar to [52].

The learned $\gamma_k(\mathbf{q}^i)$ can now be used in the optimization of A_k and P where the following optimization problem is defined

$$\begin{aligned} &\underset{\Theta}{\text{argmin}} \sum_{i=1}^{M} \left\| (\hat{\mathfrak{q}}_{att}^i)^{des} - (\mathfrak{q}_{att}^i)^{des} \right\|^2, \\ &\text{s.t.} \left\{ \sum_{j=1}^{K} \sum_{k=1}^{K} \gamma_j(\mathbf{q}^i)\, \gamma_k(\mathbf{q}^i)\; A_j P A_k^\top - P \prec 0, \right. \end{aligned} \tag{4.43}$$

with $(\hat{\mathfrak{q}}_{att}^i)^{des}$ as the model output given by (4.38) and $P = P^\top \succ 0$. With the additional matrix P in the LF, one can flatten the gradient of the LF along one direction of the state space, thus increasing the range of possible model outputs. This increase in flexibility is indeed needed as the problem is more complex. See [29] for the stability analysis of this discrete system.

4.5.2 Model Output to Reference Signal

Given that neither angular velocities nor quaternion derivatives live in quaternion space, it is impossible to use the usual affine mapping from state ξ to $\dot{\xi}$ known from SEDS and LPV-DS. Instead, the discrete system that, given the current orientation $\mathbf{q}$, predicts the desired orientation represented in the TS of the attractor was designed (4.38). To convert the desired orientation $\mathfrak{q}_{att}^{des}$ into a reference angular velocity, the vector is parallel transported to the TS of the current orientation $\Gamma_{\mathbf{q}_{att} \to \mathbf{q}} : T_{\mathbf{q}_{att}}\mathcal{M} \to T_{\mathbf{q}}\mathcal{M}$, where the exponential map is used to map it onto the manifold which then represents the desired orientation in the world frame. The desired angular velocity is obtained by first computing the error between the current and the desired orientation $\mathbf{q}_e = \bar{\mathbf{q}} \circ (\hat{\mathbf{q}})^{des}$, converting it to the axis angle representation and dividing by the demonstration sampling time dt: $\omega^{des} =$`quat2rotvec`$(\mathbf{q}_e)/\text{dt}$, with `quat2rotvec` as the conversion from UQ to axis-angle representation [53]. For an overview see Algorithm 1.

The labels $(\mathfrak{q}_{att}^i)^{des}$ for the optimization are generated according to Algorithm (2).

4.6 Simulation Results

This section presents to which extent the demonstrated data characteristics have been learned by the model and quantifies the model's performance with the error metrics MSE, dynamic time warping (DTW), and direction error. The performance

Algorithm 1 Model Output to Reference Signal

Input $\hat{\mathfrak{q}}_{att}^{des}, \mathbf{q}$, dt ▷ model output, current orientation, and data sampling rate respectively

Output ω^{des}

$\hat{\mathfrak{q}}_{body}^{des} = \Gamma_{\mathbf{q}_{att} \to \mathbf{q}} \left(\hat{\mathfrak{q}}_{att}^{des} \right)$ ▷ note that *body* only refers to the TS origin

$\hat{\mathbf{q}}^{des} = \exp_{\mathbf{q}} \left(\hat{\mathfrak{q}}^{des} \right)$

$\mathbf{q}_e = \bar{\mathbf{q}} \circ \hat{\mathbf{q}}^{des}$ ▷ body frame rotation to align $\mathbf{q}$ with $\hat{\mathbf{q}}^{des}$

$\omega^{des} = \texttt{quat2rotvec}\ (\mathbf{q}_e)\ /\text{dt}$ ▷ desired angular vel in body frame

Algorithm 2 Creating Labels from Data

for each trajectory **do**
  Input $\mathcal{Q} := \{\mathbf{q}^i\}_{i=1}^N$
  for $i < N$ **do**
    $(\mathbf{q}^i)^{des} = \mathbf{q}^{i+1}$
    $(\mathfrak{q}_{body}^i)^{des} = \log_{\mathbf{q}^i}(\mathbf{q}^i)^{des}$ ▷ map desired orientation in local TS
    $(\mathfrak{q}_{att}^i)^{des} = \Gamma_{\mathbf{q}_{att} \to \mathbf{q}^i}(\mathfrak{q}_{body}^i)^{des}$ ▷ Move TS vector into TS at attractor
  end for
  $(\mathbf{q}^N)^{des} = \mathbf{q}^N$
  $(\mathfrak{q}_{body}^N)^{des} = \log_{\mathbf{q}^N}(\mathbf{q}^N)^{des}$
  $(\mathfrak{q}_{att}^N)^{des} = \Gamma_{\mathbf{q}_{att} \to \mathbf{q}^N}(\mathfrak{q}_{body}^N)^{des}$
end for

is given w.r.t. the hyperparameters K, α, and β, where K is the total number of Gaussians and α and β the value for the lower and upper bound on the covariance eigenvalues respectively. Models of both the position, see Section 4.1, and orientation, see Section 4.5, motion generators are discussed. The demonstration data was captured by physically guiding the right arm of *neoDavid* in gravity compensation mode. Details on the motion demonstration data can be found in Section 5.1.

For this section, trajectory roll-outs in MATLAB are used. Given an initial position, the model is queried and then integrated with the Euler method to obtain the position of the next time step: $\xi_{i+1} = \xi_i + \dot{\xi}_d \, dt$. The step size for the simulations was set to $0.001s$.

4.6.1 Hyperparameter Design

To compare the influence of the number of Gaussians K and the lower and upper bound α and β for the covariance eigenvalues, different error metrics are used. The common MSE has the downside of not explicitly considering the direction discrepancy between the model output and the demonstration data. The ability to reproduce the desired direction is important and not properly captured by the MSE. Therefore a direction error is used as well. MSE and direction error are given

respectively by

$$e_{\text{mse}} = \frac{1}{\mathcal{T}} \sum_{n=1}^{N} \sum_{t=0}^{T^n} (\dot{\xi}^{t,n} - \hat{\dot{\xi}}^{t,n})^\top (\dot{\xi}^{t,n} - \hat{\dot{\xi}}^{t,n}), \tag{4.44}$$

$$e_{\text{dir}} = \left(1 - \frac{(\dot{\xi}^{t,n})^\top \hat{\dot{\xi}}^{t,n}}{\left\|\dot{\xi}^{t,n}\right\| \left\|\hat{\dot{\xi}}^{t,n}\right\| + \epsilon} \right)^2 . \tag{4.45}$$

Lastly, DTW is used as a metric to evaluate the models. DTW measures similarities between temporal sequences which may vary in speed. It produces optimal discrete matching between one series to another and from these matches an Euclidean-based distance. Provided two time signals, DTW is given by:

$$DTW(A, B) = \min \sqrt{\sum_{i,j} d(a_i, b_j)^2}. \tag{4.46}$$

First introduced in [54] it was adopted in [55] and used in many LfD works as an error metric [56], [57].

Two main hyperparameters influence the performance of a model: the number of Gaussians K and whether the eigenvalues of the Gaussian covariance matrices are constrained according to section 4.4. Given that the optimization problem is not convex, there is no guarantee that the solver finds the global minimum of the cost function. Different initial points in the optimization can make for different points of convergence. Each hyperparameter set is therefore trained four times with random initialization and the best model is kept. The model performances for different combinations of these hyperparameters are presented in Table 4.1. The upper bound is chosen to be $\beta = 100\alpha$ as it results in a conditioning number of 100 which ensures that the matrix won't be ill-conditioned due to a difference in minimum and maximum eigenvalues. The lower bound values $\alpha = 10^{-3}$ and 10^{-2} have been chosen as both promote significant smoothness of the models' output. For $\alpha = 10^{-2}$, the model performance decreased significantly w.r.t to the error metrics. The constraints are too restrictive and hinder the model from correctly learning the data. On the other hand, $\alpha = 10^{-3}$ exhibits a similar performance level to the unconstrained models and was therefore chosen for the final model. With a growing number of K, the computation time for training and inference increases proportionally. Still, the biggest model with $K = 35$ was trained on a MacBook Air with the ARM Apple M1 chip and 16 GB of RAM in 57 minutes. This is significantly faster than the training of deep learning motion generator approaches which can require hours or days of training on high-end GPUs.

Table 4.1. Error metrics by giving different hyperparameters.

MSE	DTW	Direction Error	Number of Gaussians	No Const.	$\alpha = 10^{-3}$	$\alpha = 10^{-2}$
0.0688	25.321	0.1056	5	1	0	0
0.0676	24.991	0.1017	5	0	1	0
0.0654	23.479	0.1127	10	1	0	0
0.0651	24.030	0.0989	10	0	1	0
0.0674	24.017	0.1291	10	0	0	1
0.0700	26.949	0.1216	15	1	0	0
0.0583	19.354	0.0846	15	0	1	0
0.0709	25.512	0.1243	15	0	0	1
0.0673	26.769	0.1025	25	1	0	0
0.0577	17.2983	0.0823	25	0	1	0
0.0671	26.038	0.1149	25	0	0	1
0.0522	17.0466	**0.0772**	35	0	1	0
0.0514	**16.662**	0.0788	45	0	1	0

4.6.2 Qualitative Assessment of Motion Generator

The best model obtained from the numerical evaluation is used to qualitatively evaluate the motion generator's ability to capture the demonstration data characteristics. In the following, the origin represents the attractor of the system. Furthermore, from *neoDavid's* point of view, the orientation of the coordinate system is such that the x-axis points straight ahead, the y-axis left, and the z-axis up. Therefore, positions relative to the target are labeled as in front, if the x-axis is negative, and left/right if the y-axis is positive/negative respectively. Up or below the target refers to a positive or negative z-axis, respectively.

"Bottom-to-top" motion

As seen in Fig. 4.6, the bottom-to-top motion is encoded successfully to avoid a potential shelf or table for motions that start below and from the right side of the target. Due to the workspace of *neoDavid* and the limited possibilities to reach behind and below a target, these are the most likely cases in which a reaching motion has to avoid a potential shelf or table. Initial positions where the model wasn't able to capture this behavior can be seen in some trajectories of Fig. 4.7.

"Reach-around" motion

The second characteristic set out to be captured in the motion generator was the reach around for starting positions left of the target, to avoid knocking over the object. For the majority of the cases where the initial position is left and in

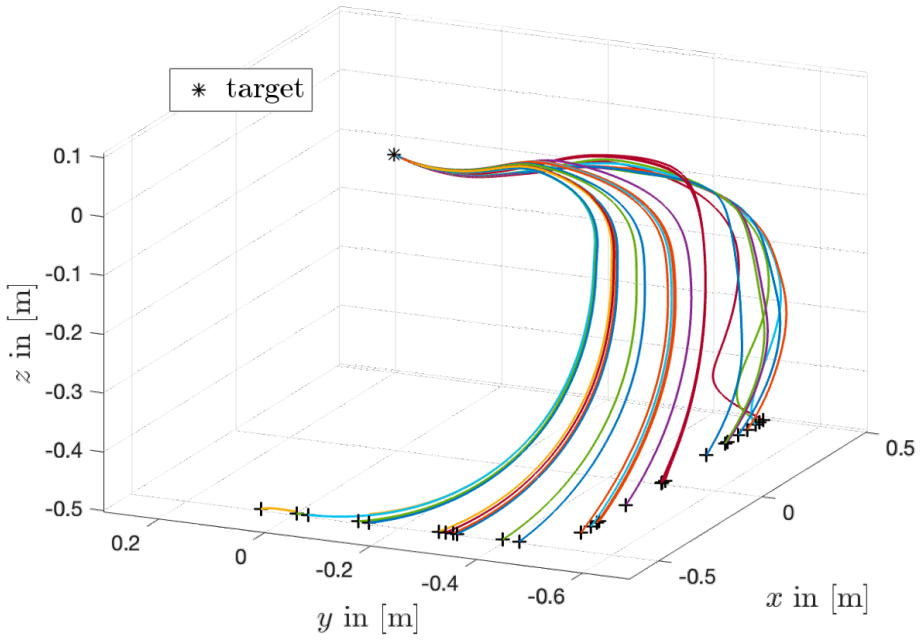


Figure 4.6. A qualitative assessment of the bottom-to-top reaching motion via different trajectory roll-outs.

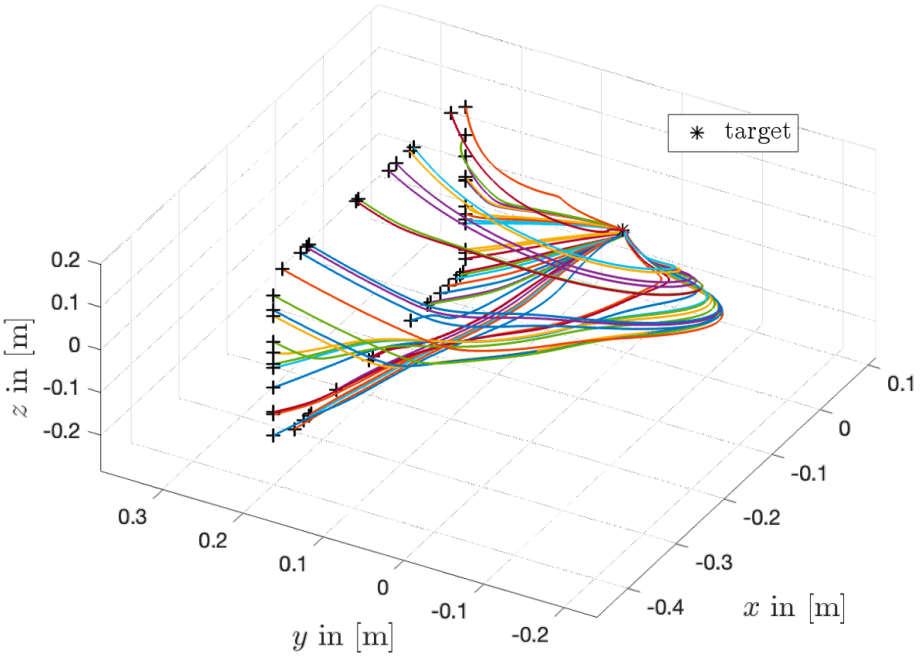


Figure 4.7. A qualitative assessment of the implicit reach around capability of the motion generator.

front of the target the reach around works. However, when for the same x and y coordinates the z-axis coordinate decreases, the reach-around motion fails. This is

caused by a lack of demonstration data: As can be seen in Fig. 5.3 in the bottom-left, demonstrations are lacking for this particular part of the region. Furthermore, some demonstrations violate the quadratic LF (4.7) because the collected data increases its distance to the target temporarily Fig. 5.2. As discussed, this cannot be captured by the model. With additional demonstration data and a more expressive LF, these shortcomings are expected to disappear.

4.6.3 Simulation of Orientation Motion Generator

Other than the positional motion generator, the evolution of the orientation trajectory is obtained by converting the model output to the desired orientation $\mathbf{q}_d$ and using this as the model input for the next time step. A comparison of a demonstration versus the motion generator's reproduction given the initial demonstration orientation is shown in Fig. 4.8. Trajectories from different initial orientations behave similarly.

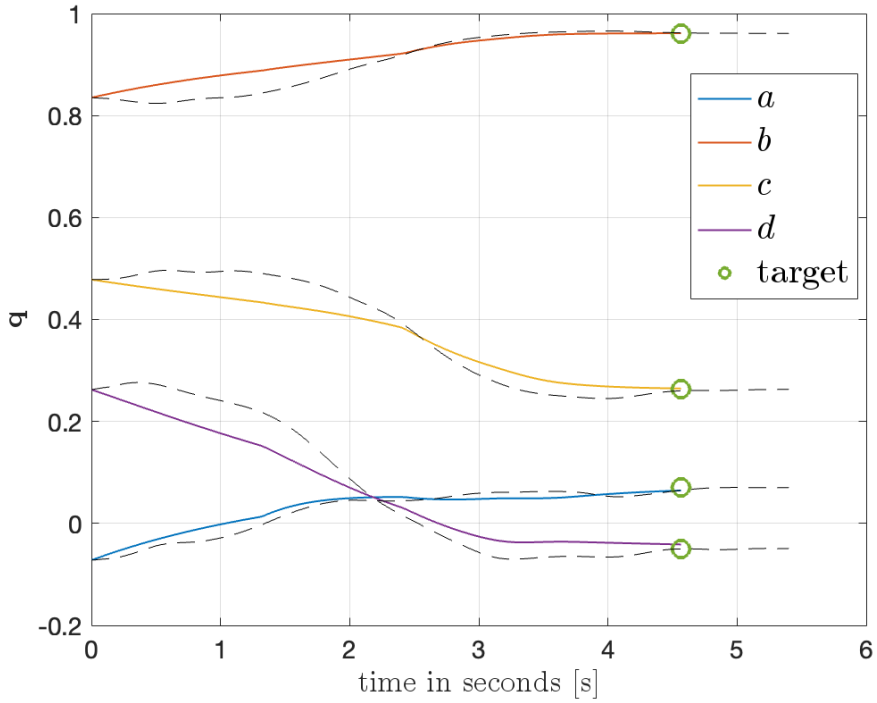


Figure 4.8. Quaternion trajectory of motion generator output versus provided training demonstration.The black dashed lines stand for demonstration data.

Chapter 5

Experimental Validations

This chapter outlines the practical steps taken to realize the theoretical concepts discussed in the previous chapter and presents the outcomes of the conducted experiments. The chapter begins with an overview of the demonstration setup, followed by a discussion about the desired motion characteristics and the demonstration data pre-processing. Additionally, it covers the available hardware setup and the development of control signals based on the motion generators' reference signals, followed by the presentation of the conducted experiments. Lastly, it discusses the chosen approach in relation to the initial goals formulated in Section 1.1.

Figure 5.1. Kinesthetic teaching of the robot in gravity compensation mode. The ketchup bottle was used as a reference for the demonstrator. Reaching motions for different initial and target positions were captured to cover the whole workspace of the robot.

5.1 Motion Demonstration

This section introduces the capture and pre-processing of the demonstrated motions.

5.1.1 Demonstration Setup

In this work, the gravity compensation mode of *neoDavid* was used to collect 17 demonstrations that were believed to sufficiently cover the relevant state space. Kinesthetic teaching was chosen as a way to collect the demonstrations since it doesn't require additional hardware and no computer vision, making it the quickest and most convenient method. The right arm of *neoDavid* was guided to execute reaching motions from different starting positions, see Fig. 5.1. During the demonstrations, the SE(3) pose of the hand TCP was recorded in the form of transformation matrices.

5.1.2 Demonstration Characteristics

The demonstration data was collected to best represent the reaching motion of a right arm as the hand of *neoDavid* is attached to its right arm. Two characteristics were chosen to be most important for this: 1) all trajectories approach the target from the right, even if they start left of it, see Fig. 5.2 and 2) a reaching motion starting below the target should first rise to the target height without decreasing the distance along the x- and y-axis. Only if it is leveled with the target it should approach it,see Fig. 5.3. Humans approach an object with their right hand from the right side instead of the left due to the wrist's natural range of motion. Even though the *neoDavid*'s wrist joints would allow for left-side grasping, the goal of human-like execution of the task justifies the additional complexity introduced in the demonstration data.

Assuming that an object is positioned on a shelf or table, adjusting the height to the correct level before approaching the target on the x-axis and y-axis when starting from below the target height reduces the risk of a collision with the table or shelf.

5.1.3 Pre-Processing

The raw demonstration data is pre-processed to obtain a suitable dataset for the optimization. First, the demonstration data is shifted so that all trajectories are aligned to terminate at the origin. This is essential for the model to learn a consistent motion. The obtained positions are smoothed with a moving average and a window size of 150 milliseconds, centered around the current data point. The position data is numerically derived and then truncated in the beginning when the

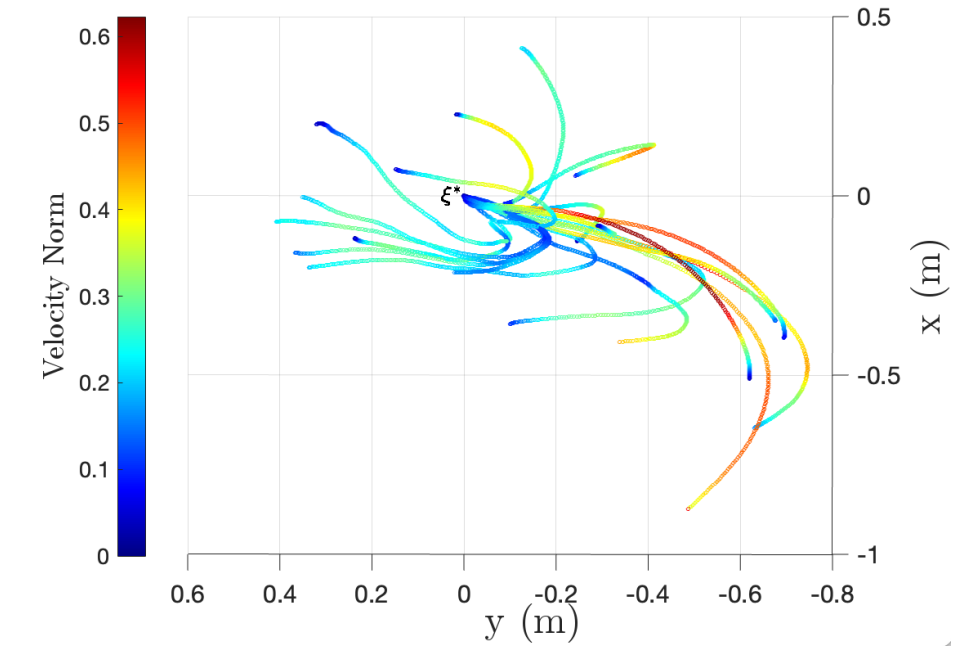


Figure 5.2. Top view of shifted translation demonstration data to end at the origin, velocity scaled, showing approach direction.

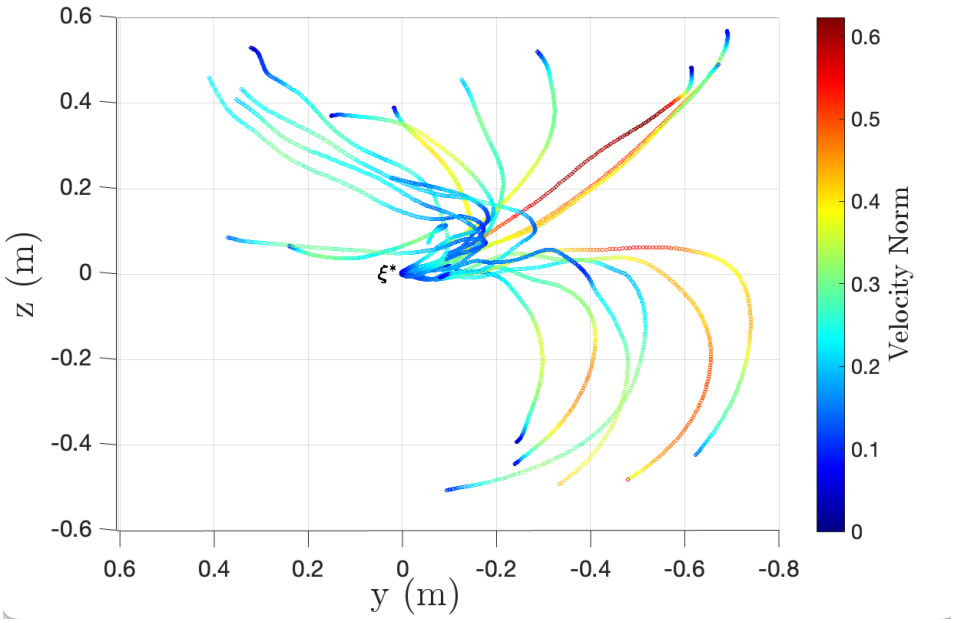


Figure 5.3. Side view of translation demonstration data, shifted to the origin, velocity scaled, showing avoidance of a shelf or table for "bottom-to-top" motions.

velocity norm first exceeds 0.1 m/s and in the end when it drops below 0.01 m/s. Next, each demonstration is downsampled to roughly 300 data points to first, reduce the computational burden of the training, as one optimization step involves the computation of as many predictions as there are data points, and second, prevent

longer demonstrations from influencing the training disproportionaly. Close to the target, all demonstrations are overlapping, thus holding the same information, resulting in an overrepresentation of the origin region during the optimization. Some demonstrations are therefore truncated before entering a 10 cm radius around the attractor.

The pre-processing results in a dataset $D = \{\xi^{t,n}, \dot{\xi}^{t,n}\}_{t=0,n=1}^{T^n,N}$ of 5492 tuples with $\xi = x - x^*$ as the position error, $\dot{\xi}$ its time derivative, $N = 17$ the total number of demonstrations and T^n the number of tuples per demonstration. x^* is the final position of each demonstration that is used to shift the whole trajectory to end at zero, resulting in $\xi^* = 0 \in \mathbb{R}^3$.

5.1.4 Solver Settings

For the optimization of the problem in (4.17a), MATLAB's [58] `fmincon` [59] solver was used. `fmincon` is a solver for constrained non-linear optimization problems. This solver is free to use and is generally considered to perform well for this type of constrained non-convex optimization problem. Besides the main hyperparameters of the method, the number of Gaussian and the consideration of eigenvalue constraints, the solver has been used with the following settings:

- the `interior-point` optimization algorithm was used, as it is generally the fastest and most reliable algorithm for this kind of problem.
- The `StepTolerance` was set to 10^{-15}. If the parameters are updated by a smaller value than this tolerance, the optimization terminates.
- The `FunctionTolerance` was set to 10^{-12} and is the minimum that the cost function has to change for each optimization step without the solver terminating.
- `TolCon`, the tolerance by which the constraints are allowed to be violated was set to 10^{-12}.
- `MaxIter`, the maximum number of iterations was set to 2000, and chosen in a way that it is not a significant limiting factor. Most of the time, the solution converges before reaching the maximum iteration and doesn't terminate only because the `StepTolerance` and `FunctionTolerance` are set to very small values.

5.2 Experimental Setup

To validate the motion generator on a real system, the *neoDavid* robot introduced in Section 2.1.1 was used to execute a reaching-and-grasping motion. To show the

reactiveness to target perturbations, the RGB-D camera inside *neoDavid's* head together with the existing object detection and tracking pipeline [60] and [61] was leveraged to track a ketchup bottle. Fig. 5.4 shows the experimental setup. The pipeline gives the current position of the tracked object which was fed back as the attractor of the motion generator. To execute the developed method the reference signal was converted to the joint space and provided as input to the ESP controller which generated the control signals. A SIMULINK model was connected to a real-time kernel that was responsible for receiving sensor data and relaying reference control signals to the robot. The unused torso joints were fixed. Due to technical difficulties, the last two joints, joints 6 and 7, had to be kept constant. Since at least 6 DoF are necessary for full pose control, the experiments focus on the positional motion generator without the orientation signals.

Figure 5.4. Setup for the target perturbation experiments.

5.2.1 Control Signals

To control *neoDavid*, the available ESP controller requires reference signals in the joint space. However, SEDS and its analytical derivatives provide a desired Cartesian velocity $\dot{x}_d$, acceleration $\ddot{x}_d$, and jerk $x_d^{(3)}$ signal with $x \in \mathbb{R}^3$ as the translation along x, y and z axis. A mapping from Cartesian space to joint space is needed. This mapping is given by the Jacobian $J(q)$ and its derivatives $\dot{J}(q, \dot{q})$ and $\ddot{J}(q, \dot{q}, \ddot{q})$. Usually $J(q) \in \mathbb{R}^{6\times7}$ but given that $\dot{x}_d \in \mathbb{R}^{3\times1}$, and the last two joints are disfunctional, $J(q)$ is truncated to $J(q) \in \mathbb{R}^{3\times5}$. The geometric Jacobian has been

used as this Jacobian also maps to the end-effector angular velocity, which is needed for future orientation control. Together with its time derivatives and the unique right side pseudo inverse $J(q)^{\#}$ they give the mappings between the desired cartesian and the desired joint signals. For computation efficiency, the static consistent null space projector has been used instead of the dynamically consistent projector

$$\dot{q}_d = J^{\#}\dot{x}_d, \tag{5.1a}$$

$$\ddot{q}_d = J^{\#}(\ddot{x}_d - \dot{J}\dot{q}_d), \tag{5.1b}$$

$$q_d^{(3)} = J^{\#}(x_d^{(3)} - 2\dot{J}\ddot{q}_d - \ddot{J}\dot{q}_d), \tag{5.1c}$$

where $J^{\#}$[1] is given by

$$J^{\#} = J^{\top}(JJ^{\top})^{-1}. \tag{5.2}$$

$J^{\#}$ is only valid for non-singular J as the joint reference signals grow unbounded when J approaches a singularity. In general, one uses the QR-Factorization to find the pseudo-inverse for rank-deficient Jacobians [62]. Dependencies on the joint configuration of $J(q)$ have been omitted for readability.

Furthermore, since the available joints result in 5 DoF, and the cartesian reference signal has three dimensions, two additional DoF need to be accounted for to prevent null space joint drift. Therefore, a null space controller to stabilize the null space dynamics is deployed. The following signals are given to the controller

$$\dot{q}_{ref} = \dot{q}_d + (I - J^{\#}J)\,\dot{q}_{\text{null}}, \tag{5.3a}$$

$$\ddot{q}_{ref} = \ddot{q}_d + (I - J^{\#}J)\,\ddot{q}_{\text{null}} - (J^{\#}\dot{J} + \dot{J}^{\#}J)\,\dot{q}_{\text{null}}, \tag{5.3b}$$

$$\begin{aligned} q_{ref}^{(3)} = {} & q_d^{(3)} + (I - J^{\#}J)q_{\text{null}}^{(3)} - 2(J^{\#}\dot{J} + \dot{J}^{\#}J)\ddot{q}_{\text{null}} \\ & - (\dot{J}^{\#}\dot{J} + J^{\#}\ddot{J} + \ddot{J}^{\#}J + \dot{J}^{\#}\dot{J})\dot{q}_{\text{null}}. \end{aligned} \tag{5.3c}$$

To develop the time derivatives of the pseudo inverse, the auxiliary variable $Z = JJ^{\top}$ is introduced and its inverse is derived twice to yield $\mathrm{d}(Z^{-1})$ and $\mathrm{dd}(Z^{-1})$.

[1] In practice, a damped least square (DLS) Jacobian given by $J_{DLS} = J^{\top}(JJ^{\top} + I\epsilon)^{-1}$ with $\epsilon = 0.0001$ is used to approximate the pseudo-inverse Jacobian $J^{\#}$. $I\epsilon$ ensures that the expression remains well-conditioned when J is singular or near-singular. Because J_{DLS} isn't a mathematically exact pseudo-inverse, $(I - J_{DLS}J)$ also does not perform an exact orthogonal projection into the nullspace of J, but prioritizes computational efficiency and ease of implementation over the accuracy of the introduced, reasonably small, error.

$$\mathrm{d}(Z^{-1}) = -Z^{-1}(\dot{J}J^\top + J\dot{J}^\top)\ Z^{-1}, \tag{5.4a}$$

$$\begin{aligned}\mathrm{dd}(Z^{-1}) = &-\mathrm{d}(Z^{-1})\ (\dot{J}J^\top + J\dot{J}^\top)\ Z^{-1},\\ &- Z^{-1}\ (\ddot{J}J^\top + 2\dot{J}\dot{J}^\top + J\ddot{J}^\top)\ Z^{-1},\\ &- Z^{-1}\ (\dot{J}J^\top + J\dot{J}^\top)\ \mathrm{d}(Z^{-1}).\end{aligned} \tag{5.4b}$$

With these terms, the pseudo inverse derivatives can compactly be given by

$$\dot{J}^{\#} = \dot{J}^\top Z^{-1} + J^\top\ \mathrm{d}(Z^{-1}), \tag{5.5a}$$

$$\ddot{J}^{\#} = \ddot{J}^\top\ Z^{-1} + 2\dot{J}^\top\ \mathrm{d}(Z^{-1}) + J^\top\ \mathrm{dd}(Z^{-1}). \tag{5.5b}$$

The null space command is often used to avoid singularity configurations based on the derivative of a manipulability measure w.r.t. q [63]. However, in the experimental setup, singularities only occur when the robot is fully stretched out, a situation not to be expected to arise. Therefore, further considerations to prevent this are unnecessary. To streamline computational efforts, a constant configuration is used as a reference resting position. A PD-controller is deployed that draws the arm to the desired configuration q_{nominal} (5.9). (5.6a) is derived twice to provide the null space signal on the acceleration and jerk level

$$\dot{q}_{\text{null}} = K_{\text{null}}(q_{\text{nominal}} - q) - D_{\text{null}}\,\dot{q}, \tag{5.6a}$$

$$\ddot{q}_{\text{null}} = -K_{\text{null}}\dot{q} - D_{\text{null}}\ddot{q}, \tag{5.6b}$$

$$q^{(3)}_{\text{null}} = -K_{\text{null}}\ddot{q} - D_{\text{null}}q^{(3)}. \tag{5.6c}$$

Diagonal matrices with higher values give the gains for the first 4 joints as their motors are stronger in comparison to the last 3 joints

$$K_{\text{null}} = \mathrm{diag}([5, 5, 5, 5, 2]), \tag{5.7}$$

$$D_{\text{null}} = \mathrm{diag}([1, 1, 1, 1, 0.1]), \tag{5.8}$$

$$q_{\text{nominal}} = [0, 30°, 0, 30°, 0]. \tag{5.9}$$

Note, that the motion generator doesn't yield a desired position. Therefore the proportional K_P term in the ESP controller is not used by setting $q_d = q$. Also, as the motion generator is purely state-dependent, there is no initial transition phase to gently accelerate the robot to the desired speed. To avoid saturation of the motors at the beginning of the motion, a ramp-up phase was implemented that is used to

scale the reference signals

$$\psi(t) = \min\left(1, \rho \int \mathrm{d}t\right), \tag{5.10}$$

where t is the current time starting at zero, $\psi(0) = 0$, and ρ is the scaling factor of the ramp-up time, chosen to be 6, 20, and 50 for velocity, acceleration, and jerk respectively.

5.3 Experimental Results

This section presents the results obtained from the implementation and testing of the proposed positional motion generator on the *neoDavid*.

5.3.1 Unscaled Motion Generator

When deploying the motion generator, the reaching motion was successfully executed. However, the acceleration and jerk reference signals both have a magnitude smaller than 1 and they therefore don't contribute significantly to the control signals, see Fig. 5.5. This is due to the demonstration data that doesn't contain many high acceleration and high jerk motions and is further caused by the covariance eigenvalue constraints that promote model output smoothness. Additionally, with a maximum end-effector velocity of 0.25 m/s, the motions are relatively slow. This is in line with the speed of the collected demonstrations. To evaluate the significance of the additional reference signals in the tracking performance and to achieve dynamic motions, the motion generator is scaled by a constant. Since the motion generator represents a DS and the additional constant doesn't violate the stability conditions, the system is still GAS, but evolves faster and consequently shows stronger acceleration and jerk.

5.3.2 Incomplete Reference Signal and Scaled Motion Generator

To establish a tracking performance baseline, the output of the motion generator is scaled by a factor of 5 and only the reference velocity signal is provided to the ESP controller with acceleration and jerk signal set to zero. With 1.43 m/s this results in a significantly faster maximum end-effector velocity of the hand's TCP compared to the unscaled motion generator, see Fig. 5.6b. However, due to the increased velocity reference signals, there is a significant overshoot of 10 centimeters along both the x and the y axis in the trajectory (Fig. 5.6a). See Fig. 5.7 for an overlayed image sequence of the overshooting trajectory. It is concluded that the reference signal cannot be tracked accurately with only the velocity reference signals.

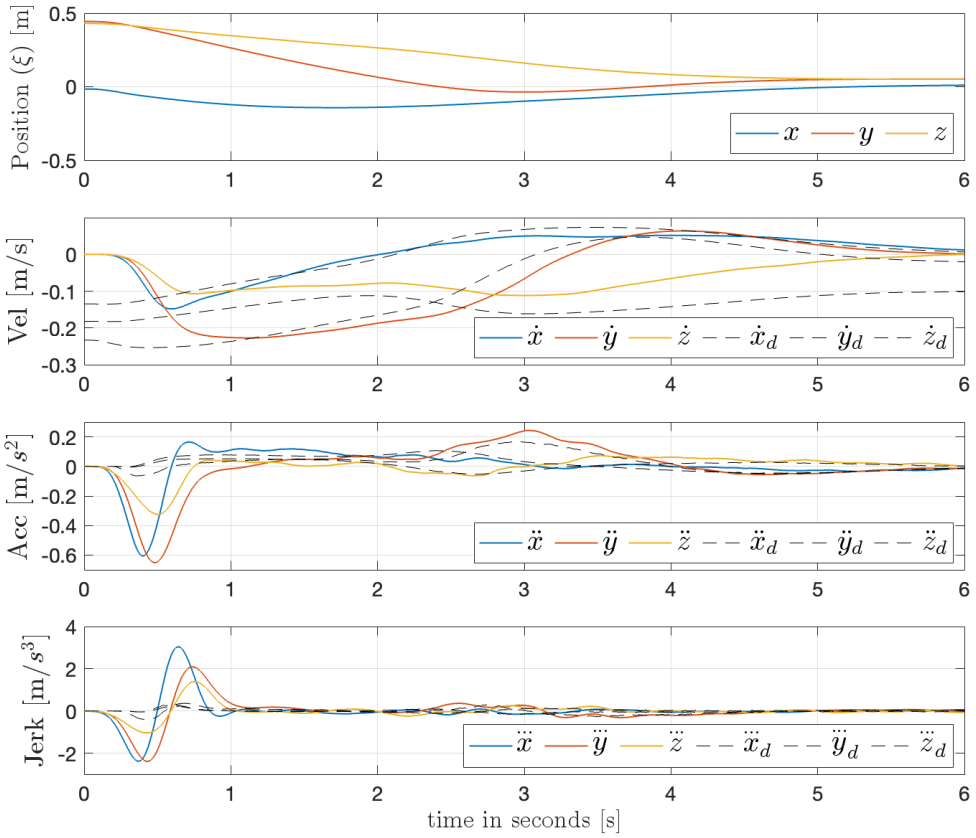
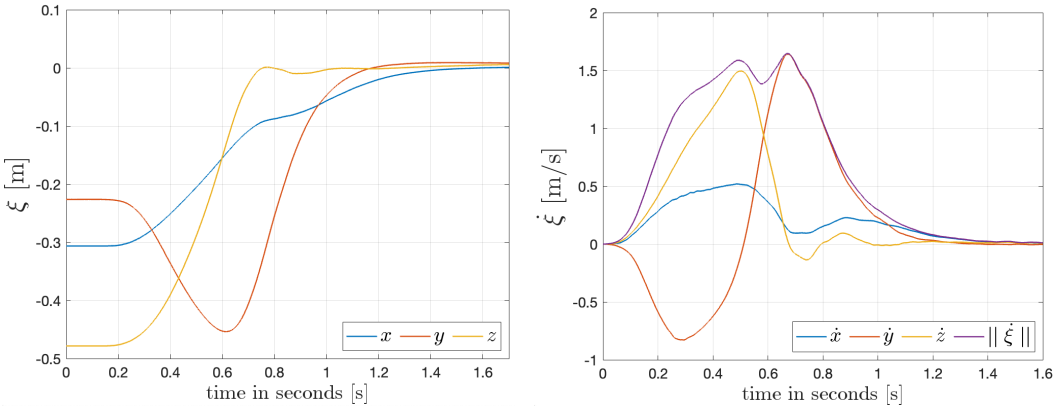


Figure 5.5. Cartesian signals of a trajectory based on an unscaled motion generator. Note the small reference values of acceleration and jerk.



(a) Cartesian position trajectory **without** all reference signals, but only velocity reference. The motion generator output is scaled by a factor of 5. Note the overshoot.

(b) Cartesian velocity trajectory **without** all reference signals, but only velocity reference. The motion generator output is scaled by a factor of 5. Note the smaller velocity norm compared to Fig. 5.8b.

Figure 5.6. Cartesian position and velocity trajectory using a scaled motion generator by the factor 5 and only providing velocity reference signals to the controller.

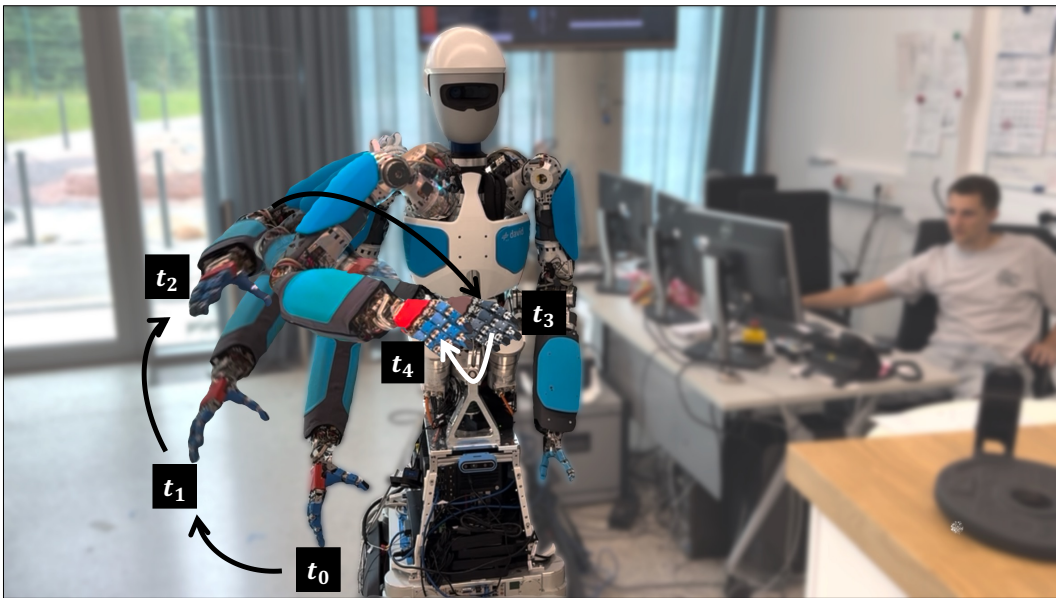
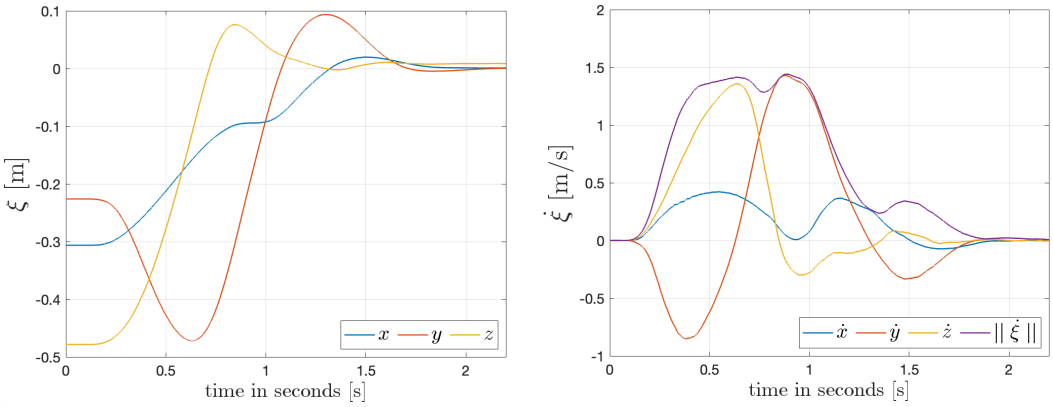


Figure 5.7. Trajectory of scaled motion generator by factor 5 **without** acceleration and jerk signal. Note the overshoot at t_3. Trajectory with acceleration and jerk didn't exhibit overshoot (Fig. 5.8a).

5.3.3 Entire Reference Signal and Scaled Motion Generator

To see the impact of the additional reference signal, the motion generator is again scaled by the factor 5, but now all reference signals, namely velocity, acceleration, and jerk are given to the ESP controller. As a result, the additional reference signals

successfully eliminated the overshoot (Fig. 5.8a), while at the same time increasing the maximum cartesian end-effector velocity by a margin of 0.2 m/s to 1.63 m/s. This proves the effectiveness of the developed approach as it increased speed while at the same time eliminating overshoot.



(a) Cartesian space position trajectory **with** all reference signals. The motion generator output is scaled by a factor of 5.

(b) Cartesian velocity trajectory **with** all reference signals. The motion generator output is scaled by a factor of 5. Note the larger velocity norm compared to Fig. 5.6b.

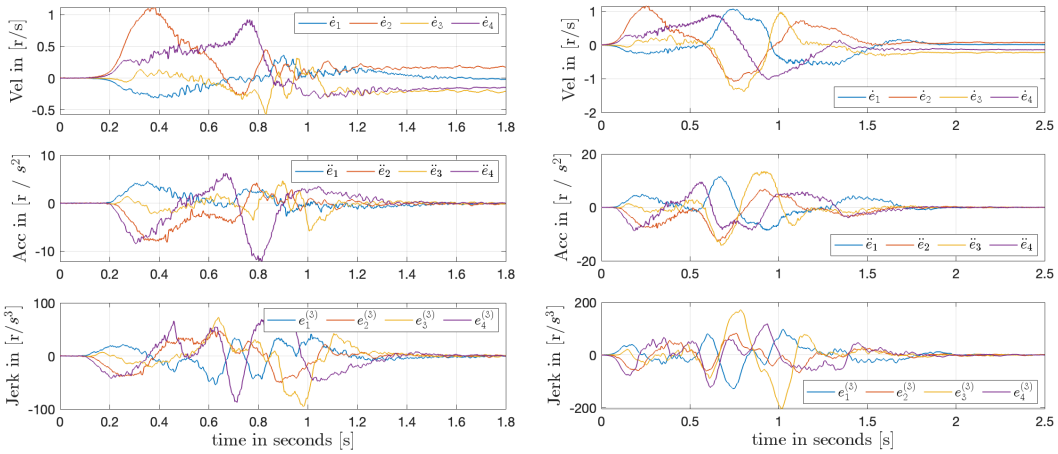
Figure 5.8. Cartesian position and velocity trajectory using a scaled motion generator by the factor 5 and providing all reference signals to the controller.

5.3.4 Effects of Higher-Order Reference Signals in Joint Space

In the joint space, the effects of providing the higher-order control signals are also evident. Scaling the motion generator by a factor of 5 and then comparing the results for the velocity-only Fig. 5.11 signal and the full reference signal stack Fig. 5.10, one can see the improved tracking of the reference velocity, acceleration, and jerk, when comparing their errors Fig. 5.9.

5.3.5 Target Perturbation and Physical Human-Robot Interaction

The experiments for the target perturbation were conducted by moving the ketchup bottle to different positions along all axes (Fig. 5.13). Since the orientation control couldn't be evaluated due to the dysfunctional last two joints, the bottle was kept at a constant orientation throughout the experiments. The motion generator was able to adapt seamlessly to the changes along all axes. Furthermore, the robot avoided the table regardless of the attractor's height without explicit obstacle avoidance. The human-robot interaction was smooth and the robot reacted seamlessly to the target perturbations.



(a) Velocity, acceleration, and jerk joint space error **with** all reference signals. The motion generator output is scaled by a factor of 5. Note the smaller error compared to Fig. 5.9b.

(b) Velocity, acceleration, and jerk joint space error **without** all reference signals, but only velocity reference. The motion generator output is scaled by a factor of 5. Note the larger error compared to Fig. 5.9a.

Figure 5.9. Velocity, acceleration, and jerk joint space error **with** and **without** all reference signals. The motion generator output is scaled by a factor of 5.

5.3.6 Reaching-and-Grasping Motions

Lastly, a distance-dependent finger closure approach was tested together with the motion generator to validate its ability to support the fast reach-and-grasp motions the thesis set out to achieve. Even for fast motions, the method was able to facilitate a successful and smooth reach-and-grasp motion, see Fig. 5.12.

5.3.7 Steady State Error

With $\|e_{ss}\| = 7.5$ cm the steady-state error for the unscaled DS is quite large. The steady-state error is not caused by the model, as it is GAS. In practice, this error may be a result of an inaccurate model or unmodelled friction forces.

Scaling the system by a constant factor is approximately inversely proportional to the steady-state error as Table 5.1 shows. The steady-state error probably decreases due to the stronger control signals overpowering possible model uncertainties or unmodelled forces.

Even though no proportional gain on the position error exists in the closed loop system, multiplying the DS causes the system to behave as if the stiffness gain had been increased. Future work might investigate the effect of a scaled motion generator on the closed-loop stiffness of the ESP controller.

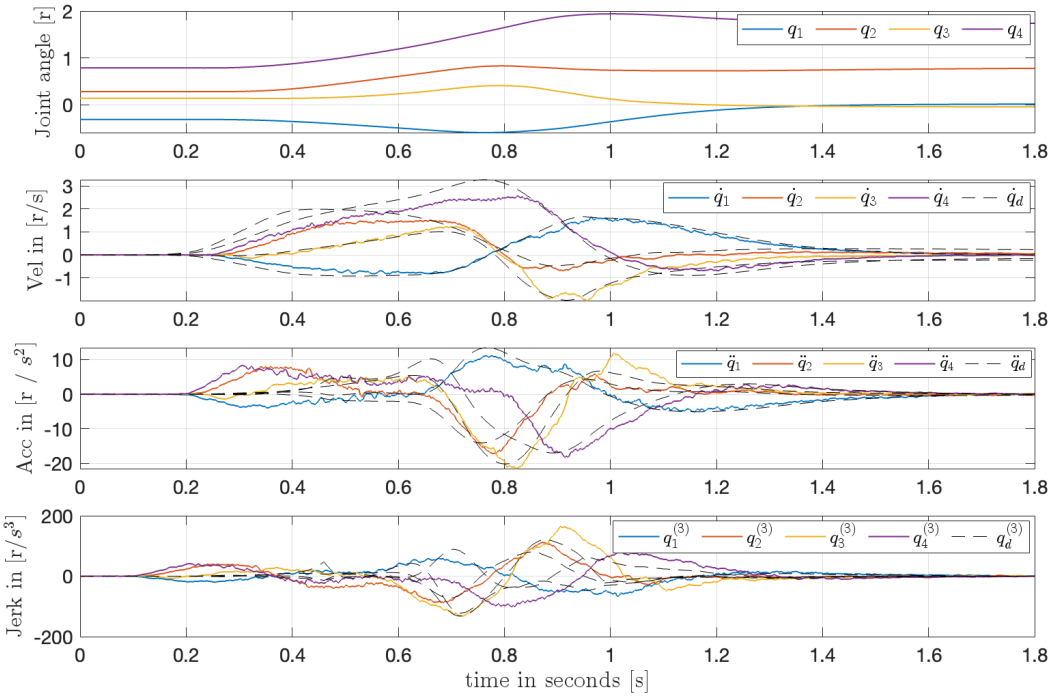


Figure 5.10. Joint space trajectory with all reference signals. The motion generator output is scaled by a factor of 5 to achieve highly dynamic motions and demonstrate the impact of providing additional reference signals.

Table 5.1. Steady state error depending on scaling, provided signal and starting position.

Scaling	Starting Position	Reference Signal	Steady State Error in cm
1	top left	vel	10.33
1	top left	vel, acc	10.95
1	bottom right	vel, acc	10.25
1	middle right	vel, acc	8.63
2	top left	vel	4.93
2	top left	vel, acc	5.32, 5.47, 5.70
2	top left	vel, acc, jerk	5.22, 5.77, 5.56
2	bottom right	vel, acc	5.01
3	top left	vel, acc	4.38, 4.52, 4.48
3	bottom right	vel, acc	3.14
3	bottom right	vel, acc, jerk	2.76
4	bottom right	vel, acc, jerk	2.00
5	top left	vel, acc, jerk	1.63
5	bottom right	vel, acc, jerk	1.42, 1.31, 1.31
5	bottom right	vel	1.51

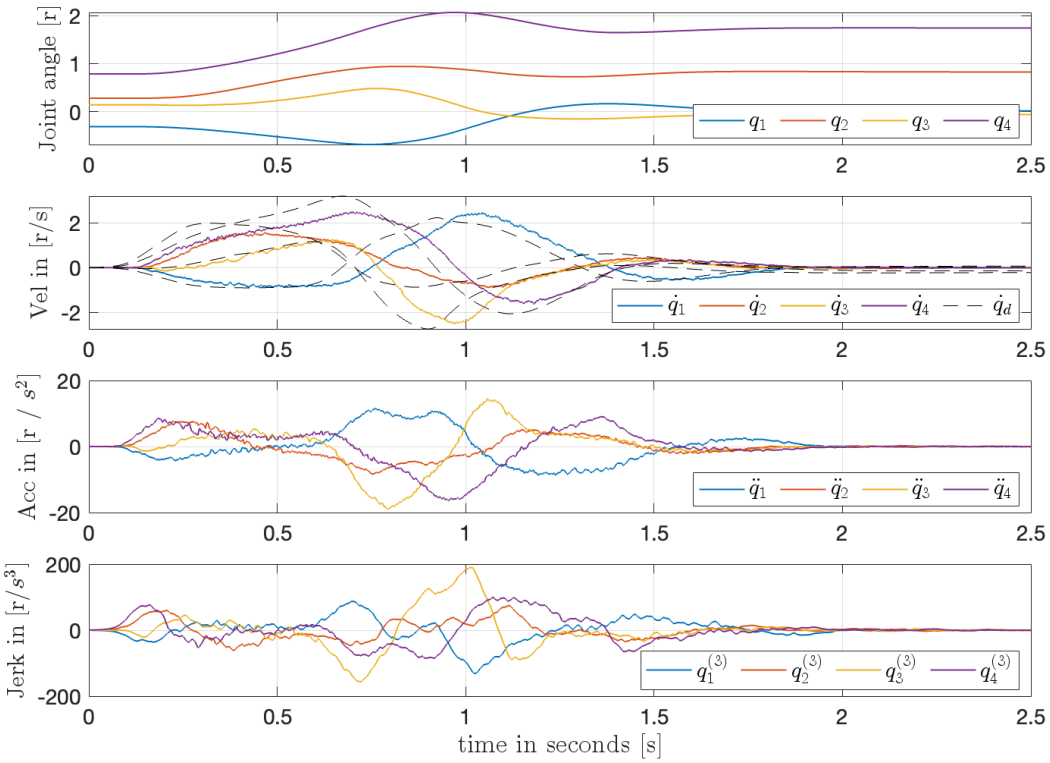
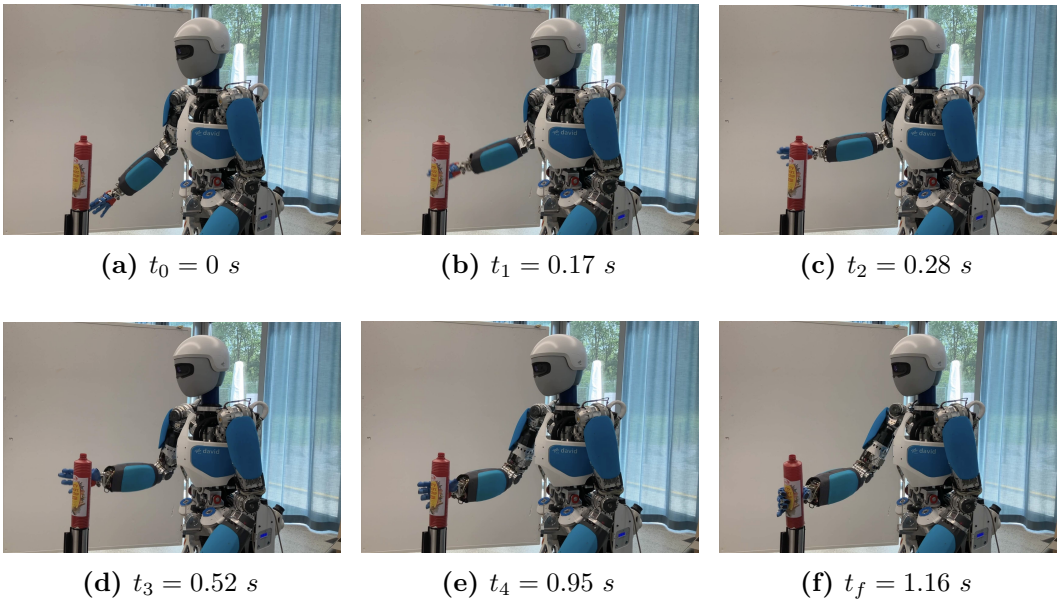


Figure 5.11. Joint space trajectory with only velocity reference. The motion generator output is scaled by a factor of 5 to achieve highly dynamic motions and demonstrate the impact of **not** providing additional reference signals.



(a) $t_0 = 0\ s$ **(b)** $t_1 = 0.17\ s$ **(c)** $t_2 = 0.28\ s$

(d) $t_3 = 0.52\ s$ **(e)** $t_4 = 0.95\ s$ **(f)** $t_f = 1.16\ s$

Figure 5.12. Image sequence of a reach-and-grasping motion, completed in 1.16 seconds. For target localization, the vision pipeline of *neoDavid* is used.

5.4 Discussion

At the beginning of the thesis, 4 goals have been presented. In the following, each of them will be revisited and discussed.

(**G1**) was to design a motion generator that provides fast and continuous reaching motions. As the results show, the motion generator has been successfully validated, especially for fast-reaching motions. The introduced constraints on the eigenvalues eliminated spikes in the control signal of acceleration and jerk and thus successfully contributed towards the smoothness of the motion.

(**G2**) was to guarantee convergence to the target even in the case of physical human-robot interaction and target perturbation. This has been successfully validated. However, on the real system, the steady-state error offers an opportunity for future work. The reasons for this have been discussed. A possible solution is to (1) scale the motion generator by a constant or (2) add specific control signals. (1) has shown to be an effective measure in decreasing the steady-state error but comes with the time-scaling of the whole motion which is undesirable depending on the application. (2) might involve the addition of a velocity error-dependent integral term to the reference signal. Activated only when the end-effector enters a certain radius around the target, it would ensure that the reaching motion is altered only in the vicinity of the target.

(**G3**): Learning the characteristics of the demonstration data and extrapolating from it was mostly successful. The motion generator couldn't avoid potential shelves with initial positions that had no demonstration data. Additionally, the reach-around motion for initial positions behind the target couldn't be reproduced. This is due to the mentioned drawback of SEDS and can potentially be resolved by using a more expressive Lyapunov, similar to the one presented in [28].

(**G4**): As the experimental validations show, the method was successfully run on the *neoDavid* elastic joint robot. The reaching motion, especially very fast ones, have been realized. The anticipated technical requirement for the first goal was higher-order derivatives of the desired joint trajectory. Everything but the snap has been provided, which was sufficient to achieve a smooth reaching motion without overshoot, even at very high speeds. The introduced constraints on the eigenvalues eliminated spikes in the control signal of acceleration and jerk and thus successfully contributed towards the smoothness of the control signals.

All of the goals have therefore been achieved.

Figure 5.13. Experiment with online feedback of *neoDavid's* vision system. The motion generator is turned on during the whole experiment. (a) - (c): bottom-top-reaching motion. (c) - (d): human-robot interaction. (e) - (f): bottom-top-reaching motion. (g)-(h): target perturbation along z-axis. (i) - (j): human-robot interaction. (k): target perturbation during reaching motion. (l): seamless convergence to target despite perturbation.

Chapter 6

Conclusion and Future Work

This work explored the efficacy of LfD techniques in enabling elastic joint robots to perform fast and smooth reaching motions. Based on the demonstrations collected via kinesthetic teaching, a motion generator for position and orientation was learned. The well-known SEDS approach was extended by developing its analytical time derivatives to provide higher-order reference signals to the elastic joint robot. As the validation experiments on the elastic joint robot *neoDavid* show, the additional reference signals increased tracking performance significantly and eliminated overshoot for highly dynamic reaching motions. Furthermore, optimization constraints on the eigenvalues of the model's covariance matrices were introduced. They were effective in smoothing the reference signals and preventing reference signal discontinuities otherwise encountered. The results further demonstrated the robustness and real-time applicability of the motion generator, as it was able to seamlessly adapt its trajectory to a new target in response to perturbations or physical human-robot interactions.

A possible avenue for future work is the extension of the orientation motion generator to be able to converge to arbitrary orientations. At present, convergence is limited to a specific orientation, which poses a limitation in unstructured environments where the target's orientation is unknown before model training and can change rapidly. A potential extension could involve designing a co-dependency between the position and orientation reference signals.

Even though the motion generator was able to learn the characteristics of a shelf or table avoidance when reaching for the target in the most important state space regions, there is an opportunity for future work to model and avoid obstacles explicitly. This would enable the robot motion to have guaranteed obstacle avoidance properties and contribute to safer motions in unstructured environments.

The regularization of introducing constraints on the covariance eigenvalues

smoothed the model output for acceleration and jerk, resulting in smooth motions. However, future work might address the need for guaranteed upper bounds on velocity, acceleration, and jerk.

Future work might also investigate the impact of the snap reference signal on the tracking performance. Lastly, a non-zero final velocity motion could be implemented for fast pick-up or catching tasks.

Building on this work, these research directions are promising in developing a reliable and reactive motion generator for elastic joint robots in virtually any environment.

Bibliography

[1] M. Keppler, C. Ott, and A. Albu-Schäffer, "From underactuation to quasi-full actuation: Aiming at a unifying control framework for articulated soft robots," *International Journal of Robust and Nonlinear Control*, vol. 32, 03 2022.

[2] S. Wolf, O. Eiberger, and G. Hirzinger, "The DLR FSJ: Energy based design of a variable stiffness joint," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 5082–5089.

[3] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger *et al.*, "The DLR hand arm system," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3175–3182.

[4] S. Wolf, C. Hofmann, T. Bahls, H. Maurenbrecher, and B. Pleintinger, "Modularity in humanoid robot design for flexibility in system structure and application," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots*, 2023, pp. 1–7.

[5] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Transactions on Robotics*, no. 5, pp. 943–957, 2011.

[6] S. S. Mirrazavi Salehian, "Compliant control of uni/multi-robotic arms with dynamical systems," *Ecole Polytechnique Federale de Lausanne, Tech. Rep.*, 2018.

[7] S. Wolf, G. Grioli, O. Eiberger, W. Friedl, M. Grebenstein, H. Höppner, E. Burdet, D. G. Caldwell, R. Carloni, M. G. Catalano, D. Lefeber, S. Stramigioli, N. Tsagarakis, M. Van Damme, R. Van Ham, B. Vanderborght, L. C. Visser, A. Bicchi, and A. Albu-Schäffer, "Variable stiffness actuators: Review on design and components," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 5, pp. 2418–2430, 2016.

[8] M. Keppler, D. Lakatos, C. Ott, and A. Albu-Schäffer, "Elastic structure preserving (ESP) control for compliantly actuated robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 317–335, 2018.

[9] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl, "DLR's torque-controlled light weight robot III: Are we reaching the technological limits now?" in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1710–1716 vol.2.

[10] C. Taesi, F. Aggogeri, and N. Pellegrini, "Cobot applications—recent advances and challenges," *Robotics*, vol. 12, no. 3, p. 79, 2023.

[11] L. Liu, F. Guo, Z. Zou, and V. G. Duffy, "Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review," *International Journal of Human–Computer Interaction*, vol. 40, no. 4, pp. 915–932, 2024.

[12] A. De Luca and W. J. Book, "Robots with flexible elements," *Springer handbook of robotics*, pp. 243–282, 2016.

[13] A. Bicchi and G. Tonietti, "Fast and "soft-arm" tactics [robot arm design]," *IEEE Robotics Automation Magazine*, vol. 11, no. 2, pp. 22–33, 2004.

[14] S. Wolf and G. Hirzinger, "A new variable stiffness design: Matching requirements of the next robot generation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1741–1746.

[15] F. Petit, M. Chalon, W. Friedl, M. Grebenstein, A. Albu-Schäffer, and G. Hirzinger, "Bidirectional antagonistic variable stiffness actuation: Analysis, design & implementation," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4189–4196.

[16] W. Friedl, M. Chalon, J. Reinecke, and M. Grebenstein, "FAS: A flexible antagonistic spring element for a high performance over actuated hand," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 1366–1372.

[17] M. Keppler, C. Raschel, D. Wandinger, A. Stemmer, and C. Ott, "Robust stabilization of elastic joint robots by ESP and PID control: Theory and experiments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8283–8290, 2022.

[18] G. J. Pollayil, X. Meng, M. Keppler, M. Pfanne, A. Bicchi, and C. Ott, "Elastic structure preserving impedance control for nonlinearly coupled tendon-driven systems," *IEEE Control Systems Letters*, vol. 6, pp. 1982–1987, 2021.

[19] X. Meng, M. Keppler, and C. Ott, "Elastic structure preserving impedance control of bidirectional antagonistic variable stiffness actuation," in *2021 European Control Conference*. IEEE, 2021, pp. 263–269.

[20] M. SPONG, "Modeling and control of elastic joint robots," *Journal of dynamic systems, measurement, and control*, vol. 109, no. 4, pp. 310–319, 1987.

[21] O. Adiyatov and H. A. Varol, "A novel RRT*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation*. IEEE, 2017, pp. 1416–1421.

[22] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[23] K. Shoemake, "Animating rotation with quaternion curves," in *12th Annual Conference on Computer Graphics and Interactive Techniques*, 1985, pp. 245–254.

[24] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1240–1247, 2017.

[25] S. Calinon and N. Jaquier, "Gaussians on Riemannian manifolds for robot learning and adaptive control," *CoRR*, vol. abs/1909.05946, 2019. [Online]. Available: http://arxiv.org/abs/1909.05946

[26] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.

[27] Y. Hu, F. J. Abu-Dakka, F. Chen, X. Luo, Z. Li, A. Knoll, and W. Ding, "Fusion dynamical systems with machine learning in imitation learning: A comprehensive overview," *Information Fusion*, vol. 108, p. 102379, Aug. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.inffus.2024.102379

[28] N. B. Figueroa Fernandez and A. Billard, "A physically-consistent Bayesian non-parametric mixture model for dynamical system learning," in *CoRL*, 2018, pp. 927–946.

[29] S. Sun and N. Figueroa, "SE(3) linear parameter varying dynamical systems for globally asymptotically stable end-effector control," *arXiv preprint arXiv:2403.16366*, 2024.

[30] S. Mohammad Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014000372

[31] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems*, vol. 70, pp. 1–15, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015000883

[32] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, pp. 51–59, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167691116300846

[33] M. Saveriano, F. J. Abu-Dakka, and V. Kyrki, "Learning stable robotic skills on Riemannian manifolds," *Robotics and Autonomous Systems*, vol. 169, p. 104510, nov 2023. [Online]. Available: https://doi.org/10.1016%2Fj.robot.2023.104510

[34] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 630–639.

[35] B. Fichera and A. Billard, "Learning dynamical systems encoding non-linearity within space curvature," *arXiv preprint arXiv:2403.11948*, 2024.

[36] J. Umlauft, A. Lederer, and S. Hirche, "Learning stable Gaussian process state space models," in *2017 American Control Conference*. IEEE, 2017, pp. 1499–1504.

[37] L. Pöhler, J. Umlauft, and S. Hirche, "Uncertainty-based human motion tracking with stable Gaussian process state space models," *IFAC*, vol. 51, no. 34, pp. 8–14, 2019.

[38] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven Lyapunov candidates," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1216–1222.

[39] Z. Jin, D. Qin, A. Liu, W.-A. Zhang, and L. Yu, "Learning neural-shaped quadratic Lyapunov function for stable, accurate and generalizable human–robot skills transfer," *Robotics and Computer-Integrated Manufacturing*, vol. 82, p. 102526, 2023.

[40] Y. Zhang, L. Cheng, H. Li, and R. Cao, "Learning accurate and stable point-to-point motions: A dynamic system approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1510–1517, 2022.

[41] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1398–1403 vol.2.

[42] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *CoRR*, vol. abs/2102.03861, 2021. [Online]. Available: https://arxiv.org/abs/2102.03861

[43] A. Wahrburg, S. Guida, N. Enayati, A. M. Zanchettin, and P. Rocco, "FlexDMP – extending dynamic movement primitives towards flexible joint robots," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 7592–7598.

[44] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf

[45] Y. Huang, L. D. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *CoRR*, vol. abs/1708.08638, 2017. [Online]. Available: http://arxiv.org/abs/1708.08638

[46] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems*, vol. 37, no. 2, pp. 286–298, 2007.

[47] S. M. Khansari-Zadeh and A. Billard, "The derivatives of the SEDS optimization cost function and constraints with respect to the learning parameters," 2011. [Online]. Available: https://infoscience.epfl.ch/handle/20.500.14299/66860

[48] S. S. Mirrazavi Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for catching softly a flying object: Theory and experiment," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.

[49] L. C. Rogers, "Derivatives of eigenvalues and eigenvectors," *AIAA Journal*, vol. 8, no. 5, pp. 943–944, 1970.

[50] M. Arnaudon, F. Barbaresco, and L. Yang, "Medians and means in Riemannian geometry: existence, uniqueness and computation," 2011. [Online]. Available: https://arxiv.org/abs/1111.3120

[51] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, pp. 127–154, 2006.

[52] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3DoF orientations," *Robotics and Autonomous Systems*, vol. 87, pp. 28–37, 2017.

[53] S. Kim and M. Kim, "Rotation representations and their conversions," *IEEE*, vol. 11, pp. 6682–6699, 2023.

[54] R. Bellman and R. Kalaba, "A mathematical theory of adaptive control processes," *National Academy of Sciences*, vol. 45, no. 8, pp. 1288–1290, 1959.

[55] C. F. Jekel, G. Venter, M. P. Venter, N. Stander, and R. T. Haftka, "Similarity measures for identifying material parameters from hysteresis loops using inverse analysis," *International Journal of Material Forming*, vol. 12, pp. 355–378, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:139768153

[56] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "ImitationFlow: Learning deep stable stochastic dynamic systems by normalizing flows," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5231–5237.

[57] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, "Continual learning from demonstration of robotics skills," *Robotics and Autonomous Systems*, vol. 165, p. 104427, 2023.

[58] T. M. Inc., "Matlab version: 9.13.0 (r2022b)," in *MATLAB*. Natick, Massachusetts, United States: The MathWorks Inc., 2022. [Online]. Available: https://www.mathworks.com

[59] ——, "Optimization toolbox version: 9.4 (r2022b)," in *OptimizationToolbox*. Natick, Massachusetts, United States: The MathWorks Inc., 2022. [Online]. Available: https://www.mathworks.com

[60] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3D orientation learning for 6D object detection," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 714–729, 2020.

[61] M. Stoiber, M. Pfanne, K. H. Strobl, R. Triebel, and A. Albu-Schäffer, "A sparse Gaussian approach to region-based 6DoF object tracking," in *Asian Conference on Computer Vision*, 2020, pp. pp. 666—682.

[62] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3733–3738.

[63] A. Dietrich, T. Wimböck, and A. Albu-Schäffer, "Dynamic whole-body mobile manipulation with a torque controlled humanoid robot via impedance control laws," in *International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3199–3206.