Estimate Human Stiffness, Damping, and Intention in a Collaborative Carrying Scenario

Jens Otto Hee Iversen

Masters Project in Engineering (Robot Systems)
Supervisor: Christoffer Sloth
Co-supervisor: Inigo Iturrate San Juan

June 2022

The Maersk Mc-Kinney Moeller Institute
University of Southern Denmark

Abstract

This thesis explores a collaborative carrying scenario between a human and a robot. Estimation of human motion intention and human impedance is presented. The motion intention is estimated with two methods; ARX Model and NARX model with a neural network as its function approximator. Impedance estimation has been attempted with a variable damping method and in the frequency domain using a transfer function of a mass-spring-damper system. Furthermore, human-like motion is generated using the minimum jerk model. An admittance controller is introduced to enable proper human-robot interaction. All experiments is done in MATLAB and Simulink. Therefore, a PD Controller with online gravity compensation was introduced to bring in dynamics. Experiments show that the ARX Model and variable damping methods work whereas the NARX Network worked partly. The impedance estimation in the frequency domain unfortunately could not produce reliable results. The admittance controller works as intended and the PD controller with online gravity compensation work reasonably well, but needed more fine-tuning.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Robots have the ability to assist humans in a vast variety of tasks. This can often increase the productivity of a particular task or decrease the burden of human labor. Many tasks are difficult to fully automate and still need the assistance and guidance of human operators. This creates an essential interaction between the human and robot called human-robot interaction (HRI). The interaction should convey the goal and intentions of the human to the robot. This is quite often a difficult task to do depending on the objective and can be hard to do accurately. Constructing the correct HRI can be conducted in many ways depending on the different interaction that is desired. The HRI is often the human showing the robot how to assist or execute a specific task.
Impedance [1] and admittance control is common methods to properly construct HRI and enable ease of guidance for the human to the robot. It is also a well researched area and have been studied for decades [2, 3, 4].
Programming by demonstration, exoskeleton or co-collaborative tasks where the human and robot do a task in unity are scenarios where impedance or admittance control could be used to enable humans to guide the robot. A co-collaborative task between a human and a robot has been tried before with a variety of different application such transferring objects between a human and a robot [5], manipulation tasks [6], assembly tasks [7], and carrying tasks [8, 9, 10, 11, 12]. In the co-collaborative carrying task, a likely scenario could be that the movement and placement of the object are decided by the human and it is the robot's job to assist the human in carrying the object, so the physical burden is minimized as much as possible. Some of the more notable and successfully are using visual sensing [10, 11], force-sensing [13], sensing of muscle activity [12], or verbal communication. Most of the works attempt to estimate the impedance of the human operator, by modeling the human arm/hand after a variety of a mass-spring-damper system, so the stiffness, damping, or motion intention can be estimated.

1.1 Problem Statement

The work in this master thesis will attempt to propose a method for making a collaborative carrying scenario possible. The thesis will propose methods for estimating the motion intention of the human operator and the human impedance. It will do the estimation based only on modeling and the force sensors attached to the robot. The idea is that the human does not require any equipment to use the system. Sub-goals for the thesis has been set:

- Estimate the motion intention of the human operator online.
- Estimate the impedance of the human operator online, such as the stiffness and damping.
- Implement an admittance controller to ease HRI.
- Combine all subparts into a full system.
- Ensure stability of the full system.
- Implement the full system in simulation (MATLAB/Simulink).
- Implement the full system on a physical robot (UR10e).

1.2 Thesis Content

Here an overview of the whole report will be given and the content of each chapter:

Chapter 1: is the introduction, problem statement, and explanation of the content of each chapter in the thesis.

Chapter 2: is the related works relevant to the thesis.

Chapter 3: is about the modelling and estimation. The chapter presents models for generating human motion, predicting human motion, and human impedance.

Chapter 4: is about the control of the thesis. This contains the admittance controller, the PD Controller with online gravity compensation and the stability of these.

Chapter 5: shows the full system with all the parts combined together.

Chapter 6: shows the test and results for the full system.

Chapter 7: is the overall discussion of the whole project together with potential future work that could be conducted.

Chapter 8: is the overall conclusion to the thesis.

Chapter 9: is the acknowledgement of the people that have helped make this thesis a reality.

2 Related works

This section will introduce research relevant to the topics talked about in chapter 1 and the problem statement
J. DelPreto et al. [12] present a human-robot team lifting task using the muscle activity of a human recorded using an EMG sensor. The sensor measures two different muscle activities: bicep and tricep. The bicep muscle activity is used for their setpoint algorithm which estimates changes in the operator's hand height. This is used to make the robot match the same lifting height as the operator's hand. The setpoint algorithm has eight parameters which are used for the proposed algorithm. Some of the parameters were estimated using a genetic algorithm and some of them are based on the individual human. The bicep and tricep are used for their rolling gesture classification, which is detecting up and down gestures from the two muscle activities. This allows the user to control fine adjustments or move the robot to targets further from the user's hand height.

X. Yu et al. [10] propose a hybrid framework for human-robot co-carrying tasks, that uses visual and force sensing. They use visual sensing to obtain the human motion, by having a calibration board at the grasp pose of the human operator. The force sensing is done at the end-effector of the robot and is used as an input to an observer whose purpose is to estimate the intention of human motion. All of this is happening online, so it can be used in practice. For both the visual sensing and the force-sensing a moving average filter ought to be used for practical purposes to obtain smooth data. At last, they used a neural network to compensate for uncertainties in the dynamics of the robot.

F. Müller et al. [14] present a user force-dependent variable impedance controller (VIC), where the force is proportional to the velocity in the steady-state and the force is nearly proportional to the acceleration in the acceleration and deceleration phase. This yields an force-dependent VIC that is in the form of a velocity- and acceleration-based VIC. They also present a stability observer so they can guarantee stable behavior. For estimating the stiffness and damping of the human, they set up a model for the human. The model consists of a passive part, an active part, and a response time. The passive part considers models the human arm using a mass-damping system. The active part describes all the active reactions of a human, such as acceleration, velocity, position, and force. For estimation the stiffness they use the ratio of the derivative of the user force and the derivative of the velocity. For the damping a similar approach is applied where it is the derivative of the velocity instead of position.
X. Yu et al. [13] present a method for estimating human stiffness and human

intention. They use a spring model for modeling the human and then apply the statistical Bayesian parameter estimation method to estimate the stiffness and human intention. They do this by using the dynamic relationship between human stiffness and motion. Furthermore, they also use a neural network to estimate uncertainties in the robot model so it can be compensated for.
K. Li et al. [15] proposes a compliant dual loop controller. They present an inner position loop and an outer force loop. The outer loop is designed to generate a force for assisting the human that is interacting with the robot. The outer loop consists of a human arm model based on a spring-damper system, an MRAC based force compensation method that is proposed to eliminate time delay during HRI, and a Sparse Bayesian Learning (SBL) based human intention predictor. The inner loop generates a Cartesian drive torque for the robot based on second-order nonlinear differential function and a PD controller with online gravity compensation to compensate for the unknowns in the nonlinear differential function. The other loop SBL-based human intention predictor is modeled after a combination of a nonlinear autoregressive model with exogenous inputs (NARX) and a relevance vector machine.
T. Bitz et al. [16] proposes a variable damping control to improve agility, stability, and reduction of user effort. A dual-sided logistic function (DSLF) is used to vary the damping of the controller. The function is used in real-time, so the varying damping can happen online. Furthermore, the function is also designed to have no sudden jumps in damping making it have smooth curves. The human intention is estimated using the product of the velocity and acceleration and they called it user intent. The user intent gives a measure of the change in kinetic energy of the system. The DSLF has two cases; one where the user intent is positive and one where it is negative. The reason for this is that then the user intent is positive it indicates that the human intends to initiate and accelerate the robot. Therefore, the damping should be decreased to improve agility. It is vice versa then the user intent is negative, then the human intends to decelerate the robot and therefore the damping should increase. When the user intent is zero, meaning no movement of the robot is occurring the damping converges to its resting point. Furthermore, there are user set limits to the damping, so it is possible to restrict it within a certain range. The reason for this is to ensure stability. The proposed work is tested and improves the trade-off between agility and stability and reduces human effort compared to a fixed damping controller.

X. Yu et al. [17] present a method for estimating human motion intention and human impedance parameters. For the human motion intention estimation, they use a Radial basis function neural network (RBFNN) and use an adaptive method to properly estimate the weight of the RBFNN. Furthermore, they use a cost function and adjust the radial basis function based on the steepest descent method since the interaction force becomes small when the motion prediction is close to the ground truth. For estimating the impedance they have an impedance model and an impedance learner. The impedance model

is a damping-stiffness model and the impedance learner uses an impedance tuning rule for lowering the impedance parameter when the human is leading the task, using the Least Square method to estimate the impedance parameter using a cost function, and at last using a moving average filter to improve the estimation accuracy. Furthermore, they also use an RBFNN to compensate for uncertainties in the robot dynamics together with barrier Lyapunov functions to ensure position and velocity constraints are not violated.

3 Modelling

This chapter contains three parts. The first part is generating human like-motion which is essential since it is the foundation for all the tests done in this project. This is explained in-depth in section 3.1. The second part is about predicting the motion of the human operator and is explained in section 3.2. The third and final part is estimating human impedance and is explained in section 3.3. For each of the three sections there is a *methods* section for explaining the different methods that could be utilized, a *method choice* section for explaining which method is used, an *implementation* section that explains how the chosen methods are implemented, a *results* section that shows the results of the tested methods, a *discussion* section that discusses the results, and at last a *conclusion* section that concludes on the results.

3.1 Generate Human Motion

Since the whole project revolves around a human operator guiding a robot it would be best if the trajectory used for testing resembled that of human motion the most. This would give better and more accurate insight if the chosen methods works as intended.

Methods

There are two main ways to describe and generate human arm movement [18]. The first is with descriptive models and the second is with complete models. Descriptive models have the goal of describing the apparent behavior of human motion but do not take into account the underlying biological principles. However, this is exactly what complete models do. It takes into account the dynamics of the human arm and focuses on joint torques, external forces, and motor commands. As a third option, there is also the possibility of recording human motion using a physical robot.

Descriptive Models
There exist quite a few methods for descriptive models such as Fitts law, the 2/3 power law, and minimum jerk [18][19][20]. Fitts law has rapid goal-directed movements and quantifies the intuitive relationship that is between the duration of the movement with the distance and dimension of the target. The relationship can be expressed mathematically[21] and is shown in equation (3.1).

$$\mathrm{MT} = a + b \log_2 (2A/W) \tag{3.1}$$

where MT is the movement time, a and b is regression coefficients, A is the distance of movement from start to target center and W is the width of the target.
The 2/3 power law states that in curved movements the velocity decreases with increasing curvature. This is expressed with the velocity being related to the inverse of the curvature radius by $v(t) = \gamma k(t)^{-1/3}$. Where γ is a constant factor and $k(t)$ is the curvature. There is some limitations to the original approach and it is that the velocity at straight segments and inflection points goes to infinity. Furthermore, it is inaccurate at low velocity and cannot predict the velocity at the end of the path.
Minimum jerk used the derivative of the acceleration called jerk. This is done in order to optimize the smoothness and suppress vibrations[19]. minimum jerk uses a bell-shaped velocity profile since it is one of the most consistent features of human arm behavior in straight movements. The magnitude of jerk (J) is defined in equation (3.2) and is to address planar movements.

$$J = \sqrt{\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2} \tag{3.2}$$

x and y are the time-varying hand position coordinates. Optimal control methodology often requires a cost function that can be minimized to achieve the best performance. This also exists for the minimum jerk, see equation (3.3).

$$C = \frac{1}{2}\int_0^{t_f} \left(\left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2\right) dt \tag{3.3}$$

t_f is the time duration. The mathematically expression for the hand trajectory assuming zero velocity and acceleration at the start and end of the trajectory can be seen in equation (3.4).

$$\begin{aligned} x(t) &= x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \\ y(t) &= x_0 + (y_0 - y_f)(15\tau^4 - 6\tau^5 - 10\tau^3) \end{aligned} \tag{3.4}$$

where $\tau = t/t_f$, x_0 and y_0 is the initial hand position coordinates at $t = 0$, and x_f and y_f is the final hand coordinates at $t = t_f$. This is only for straight point to point movements and the shape of the trajectory does not change with amplitude or duration of movement, but only scales the position and time axes. Also note that the trajectory is described with a single fifth order polynomial which is quite common to use for minimum jerk trajectory. Having curved point to point movements where the trajectory needs to pass through a

waypoint (x_1, y_1) at some intermediate time t_1 introduces equality constraints which look like $x(t_1) = x_1$, $y(t_1) = y_1$. It is required that there is continuity of the velocity and acceleration at t_1, since discontinuity would require infinite accelerations and jerks. The expression for the hand trajectory is now split into two expression one for $t \leq t_1$ called $x^-(\tau)$ and one for $t \geq t_1$ called $x^+(\tau)$. The expression will only be showed for x, but the expression for y is similar to that of x. See equation (3.5) and (3.6) for the expressions.

$$x^-(\tau) = \frac{t_f^5}{720}(\pi_1(\tau_1^4(15\tau^4 - 30\tau^3) + \tau_1^3(80\tau^3 - 30\tau^4) - 60\tau^3\tau_1^2 + 30\tau^4\tau_1 - 6\tau^5) + c_1(15\tau^4 - 10\tau^3 - 6\tau^5)) + x_0 \tag{3.5}$$

$$\begin{aligned} x^+(\tau) &= \frac{t_f^5}{720}(\pi_1(\tau_1^4(15\tau^4 - 30\tau^3 + 30\tau - 15) + \tau_1^3(-30\tau^4 + 80\tau^3 - 60\tau^2 + 10)) \\ &\quad + c_1(-6\tau^5 + 15\tau^4 - 10\tau^3 + 1)) + x_f \\ &= x^-(t) + \pi_1 \frac{t_f^5(\tau - \tau_1)^5}{120} \end{aligned} \tag{3.6}$$

where $\tau_1 = t_1/t_f$, c_1 and π_1 is constant coefficients and will be c_2 and π_2 for $y^-(\tau)$ and $y^+(\tau)$. For more detail on these constants see [20].

Complete Model

The models presented under descriptive models focus only on the generation of trajectory in Cartesian space and do not consider how the joint space redundancy is solved. However, this is what complete models do using theories that consider arm dynamics and including torque and external forces into their cost function. Since the arm dynamics is non-linear and depends on joint kinematics there is a consequence that the minimization of a cost function affects the resolution of all levels on redundancy. There are two forms of complete models; dynamic models and stochastic models. Dynamic models take the cost function from the minimum jerk and introduce dynamic variables. Three major models exist for dynamic models and are named after their variable of interest; *minimum torque change*, *minimum motor command change*, and *minimum commanded torque change*. The minimum torque change model was the most influential because different dynamic variables have been tested for the cost function and the torque variable was the one that yielded the best agreement with observed behavior. The cost function can be seen in equation (3.7).

$$C = \frac{1}{2}\int_0^{t_f} \sum_i \left(\frac{dz_i}{dt}\right)^2 dt \tag{3.7}$$

where z_i is the torque generated at joint i. This function is then minimized under the constraints of the musculoskeletal dynamics. For the purpose of

addressing planar movements the arm dynamics are approximated by a two-joint planar robot dynamics, with inertial, geometric, and viscosity parameters representative of the human arm characteristics. The final output of the movement planning which are all produced at the same time is hand trajectory, joint trajectory, and torque.

Stochastic models suggest that human movement can be explained more consistently within a framework where the noise in muscle commands is considered. There is still things the dynamic models does not take into account such as then humans are provided with visual feedback which is changed in the area between the initial and final position, they adapt the hand path in order to correct the trajectory. Also the dynamic model mentioned here is not able to predict certain movement experiments. This is all something stochastic models are trying to correct for. However, there exists quite a lot of method that fall under the category of stochastic models and since stochastic models are not going to be the focus here, it will not be elaborated on in more detail. If more detail is desired see [18].

Recording Trajectory on Physical Robot

An option to generate human-like motion is to record human motion on a physical system. This will naturally give the most accurate human like-motion, but can also be a very time-consuming task. A way to record human-like motion on a physical system could be to use a UR robot from their e-series [22] since it has a force torque sensor (FT-sensor) in wrist 3. Then it would be possible to use the free drive to move the robot around and record the necessary information such as TCP pose, joint angles, FT-sensor readings, etc.

Method choice

It is chosen to use a descriptive model instead of a complete model since the only interest for this project is the movement of the human hand touching the robot and it is therefore deemed unnecessary to have a complete model since it would add extra complexity that would not be necessary. For the descriptive model it is chosen to use the minimum jerk model since it is model that describe the movement the most complete and it works the best in most scenarios. Also this model is one of the most used model of the descriptive ones and is the one most other models take inspiration from. Furthermore, the minimum jerk model can generate trajectories for straight and curved movements and tries to smooth them as much as possible.

Furthermore, it has also been chosen to record trajectories on a physical UR10e robot, so it is possible to compare this with the minimum jerk model, but also to have a more realistic trajectory.

Implementation

For full implementation of the minimum jerk model and the trajectory recordings on the UR10e, see Appendix A.1.

Minimum Jerk Model

There is no reason to implement a minimum jerk model from scratch, since there exist numerous implementations publicly available online and some of them have also more advanced functionality than the one described in this thesis. Therefore, an implementation of a minimum jerk model in MATLAB has been found [23] and it has been modified to fit the use case of the thesis. A simple diagram of the implementation can be seen in figure 3.1.

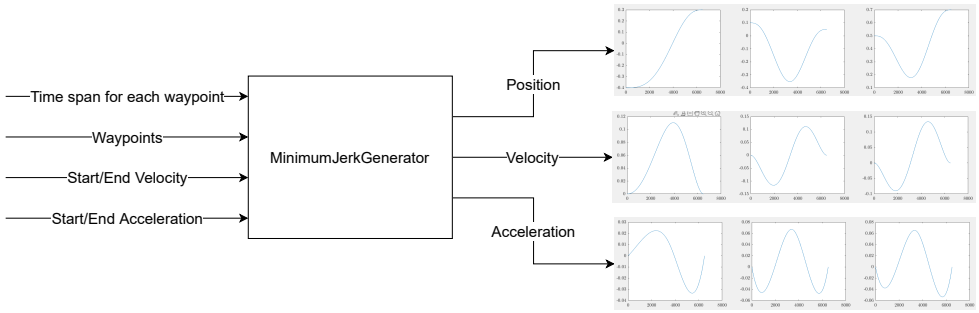


Figure 3.1: Diagram of the implementation of the minimum jerk model.

The *MinimumJerkGenerator* is the implementation from [23] there the sampling frequency has been changed to match that of the simulation used in this project (see chapter 5), together with the option of choosing to visualize the position, velocity, and acceleration. The method have the following inputs; a list of waypoints to visit, the timestamp for each of those waypoints, the start and end velocity and acceleration of the trajectory. The method then outputs a positional trajectory, the velocity profile and acceleration profile. There is not a force output of the minimum jerk model which there needs to be in order for the other sections of the project to use the trajectories. However, a force can easily be generated using the acceleration output from the minimum jerk model by applying Newtons 2. law, see equation (3.8), if a mass is assumed.

$$F = ma \tag{3.8}$$

where F is the force, m is the mass, and a is the acceleration. Then applying Newtons 2. law to each data point of the acceleration would result in a force output with the same behavior as the acceleration but the amplitude guided by the assumed mass.

Recording Trajectory on Physical Robot

A python script was develop to record trajectories on the physical UR10e. The script been made with a small interface that makes it possible to toggle free

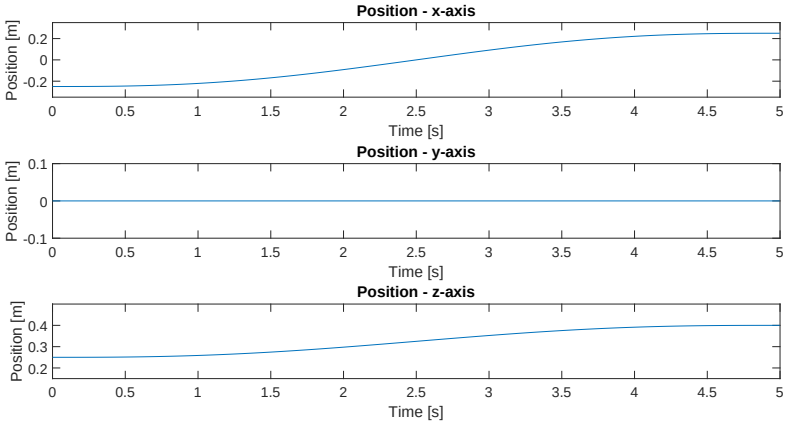
drive on the robot, start and stop recording of trajectories, and save the last recorded trajectory to a csv file. For each recording the pose of the end-effector, the joint positions, and the force sensor readings are saved.

Results

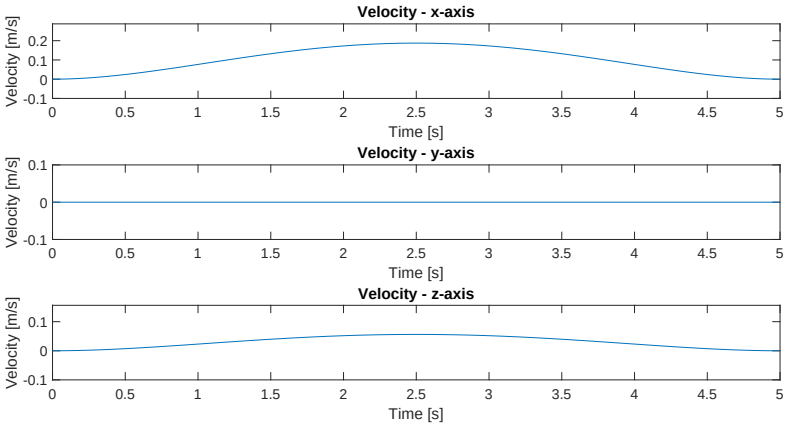
Minimum Jerk Model
Four different minimum jerk trajectories have been generated. The first trajectory can be seen in figure 3.2, the rest can be seen in Appendix A.2 (figures A.1, A.2, A.3). For each trajectory, the position, the velocity, and the acceleration has been generated for each of the three axis. Table 3.1 shows the waypoints used for generating the trajectories together with the timestamps for each waypoint. All the trajectories are generated with zero velocity and acceleration at the start and endpoint.

	Axis	**Start Point**	**Waypoint 1**	**Waypoint 2**	**End Point**
MJT 1	**x-axis**	-0.25	-	-	0.25
	y-axis	0	-	-	0
	z-axis	0.25	-	-	0.4
	Time	0	-	-	5
MJT 2	**x-axis**	-0.25	-	-	0.25
	y-axis	0	-	-	0
	z-axis	0.25	-	-	0.4
	Time	0	-	-	2
MJT 3	**x-axis**	0	2	1	0
	y-axis	0	0	$\sqrt{3}$	0
	z-axis	0	0	0	0
	Time	0	4	8	12
MJT 4	**x-axis**	-0.4	-0.1	-	0.3
	y-axis	0.1	-0.35	-	0.05
	z-axis	0.5	0.2	-	0.7
	Time	0	7	-	13

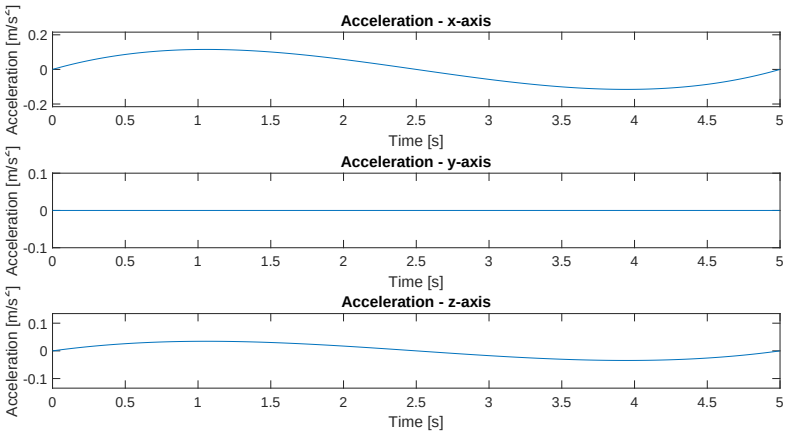
Table 3.1: Waypoints and timestamps for generating the four minimum jerk trajectories. MJT stands for minimum jerk trajectory.



(a) Position



(b) Velocity



(c) Acceleration

Figure 3.2: The first trajectory generated using the minimum jerk model. Will be referred to as minimum jerk trajectory 1.

Recording Trajectory on Physical Robot

There has been record four trajectories on a physical UR10e. The trajectory was record with the principal of minimum jerk in mind. The first trajectory is shown in figure 3.3 and the rest can be found in Appendix A.2 (figure A.4, A.5, A.6). For each trajectory the position and force will be shown.

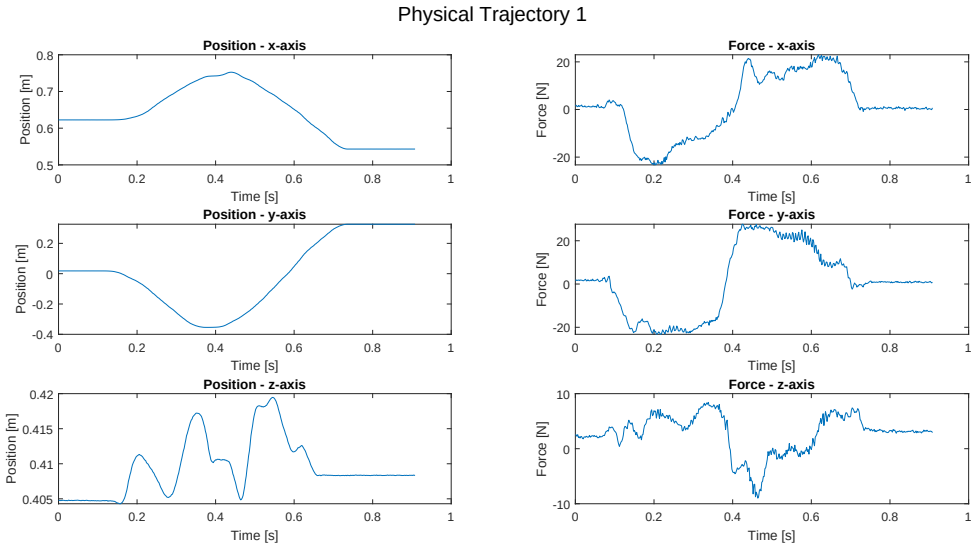


Figure 3.3: The first trajectory recorded on a physical UR10e. Will be called physical trajectory 1.

Discussion and Conclusion

From figure 3.2, A.1, A.2, and A.3 it can be seen there has been generated four trajectories following the minimum jerk model and that their positional trajectory is was would be expected from the model. Furthermore, it can be seen that the velocity profile follows the bell shape, which was desired. Trajectory 1 and 2 are the same positionally just with different timestamps. This has been done to have tests with the same postional trajectory, but different time lengths and amplitude in velocity. The four trajectories have different amounts of waypoints to visit. This was done to have more variety in trajectories.

From figure 3.3, A.4, A.5, and A.6 the four trajectory recorded on the physical UR10e robot can be seen. The only thing to notice here is that the force measurements have a fair bit of noise and this might cause challenges for other parts of the project.

3.2 Motion Prediction

Predicting the motion of the human user is the essential part of this thesis. Three methods will be elaborated on in-depth. The methods all have in common that they have force data as input and position as output, as this is desired. The first method is the use of an autoregressive model with exogenous inputs (ARX) [24] and will be referred to as ARX Model. The second method is the work by K. Li et al. [15] that uses a nonlinear autoregressive model with exogenous inputs (NARX) and a Sparse Bayesian Learning (SBL) human intention predictor and will be called SBL Human Intention Predictor from now on. The third and last method is the use of another NARX using a neural network as its function approximator [25] and will be refereed to as NARX Network from here on out.

Methods

ARX Model
The ARX Model is an extension of the autoregressive (AR) model, but unlike AR it includes an input term [24]. The main idea of the ARX Model is that the current and future outputs depend on past inputs and outputs. The ARX Model can be expressed mathematically as shown in equation (3.9).

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) = b_1 u(t-n_k) + \ldots \\ + b_{n_b} u(t-n_b-n_k+1) + e(t) \end{aligned} \tag{3.9}$$

where $y(t)$ is the output at time t, n_a and n_b is the order of the ARX Model with the number of poles and zeros respectively, n_k is the number of samples that occur before the input has an effect on the output (also called delay) and $e(t)$ is a white-noise disturbance value. Equation (3.9) can be written in matrix form for compactness with q being the delay operator, see equation (3.10).

$$\begin{aligned} A(q)y(t) &= B(q)u(t-n_k) + e(t) \\ A(q) &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \\ B(q) &= b_1 + b_2 q^{-1} + \ldots + b_{n_b} q^{-n_b+1} \end{aligned} \tag{3.10}$$

NARX - SBL Human Intention Predictor
K. Li et al. [15] express the human desired position in the form of a nonlinear time-varying function or closely related to a variation of a mass-spring-damper system. See equation (3.11).

$$\boldsymbol{M}_H \ddot{\boldsymbol{p}} + \boldsymbol{D}_H \dot{\boldsymbol{p}} + \boldsymbol{K}_H (\boldsymbol{p}_d - \boldsymbol{p}) = \boldsymbol{f}_H \tag{3.11}$$

where $\boldsymbol{M}_H$, $\boldsymbol{D}_H$, $\boldsymbol{K}_H \in \mathbb{R}^{n \times n}$ is positive definite matrices, where they are the mass, damping, and stiffness of the human arm respectively. $p \in \mathbb{R}^{n \times 1}$ is

the position, $p_d \in \mathbb{R}^{n\times 1}$ is the desired position and $f_H \in \mathbb{R}^{n\times 1}$ is the external force exerted by the human. For simplicity of explanation, only the x-axis is looked at. Then a single function $G(\cdot)$ can express the desired position, see equation (3.12).

$$x_d = G\left(f_H, \dot{x}, x\right) \tag{3.12}$$

where $x, \dot{x}, x_d$ represent the robot manipulator position, robot manipulator velocity, and human desired position along the x-axis. $G(\cdot)$ can then be transformed into a NARX model, so the human desired position can be expressed in a time series. See equation (3.13).

$$\begin{aligned} \boldsymbol{A}\left(z^{-1}\right) x_d(k) &= \boldsymbol{B}^T\left(z^{-1}\right) \boldsymbol{F}^T\left(\boldsymbol{u}(k)\right) + \zeta(k) \\ \boldsymbol{A}\left(z^{-1}\right) &= 1 + a_1 z^{-1} + \ldots + a_n z^{-n} \\ \boldsymbol{B}\left(z^{-1}\right) &= \begin{bmatrix} b_1^1 & b_2^1 & \ldots & b_q^1 \\ b_1^2 & b_2^2 & \ldots & b_q^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^m & b_2^m & \ldots & b_q^m \end{bmatrix} \begin{bmatrix} z^{-1} \\ z^{-2} \\ \vdots \\ z^{-q} \end{bmatrix} \end{aligned} \tag{3.13}$$

where $\mathbf{u}(k) = \left[f_{H,x}(k), \dot{x}(k), x(k)\right]^T$ is an input vector that contains the force, velocity and position. $\boldsymbol{F}(\boldsymbol{u}(k)) = \left[\tilde{f}_1(\boldsymbol{u}(k)), \tilde{f}_2(\boldsymbol{u}(k)), \ldots, \tilde{f}_m(\boldsymbol{u}(k))\right]$ is a vector of directional functions, which is determined by the user. $\boldsymbol{\zeta}(k)$ is additive noise. To simplify equation (3.13) is can be rewritten into the form shown in equation (3.14).

$$\begin{aligned} \boldsymbol{x}_d(k) &= \boldsymbol{\omega}\boldsymbol{\Phi}(k) + \boldsymbol{\zeta}(k) \\ \boldsymbol{x}_d(k) &= [\hat{x}(k|k-1), \hat{x}(k-1|k-2), \ldots, \hat{x}(k-N|k-N-1)]^T \\ \boldsymbol{\Phi}(k) &= [\phi(k), \phi(k-1), \ldots, \phi(k-N-1)] \\ \boldsymbol{\zeta}(k) &= \left[\hat{\zeta}(k|k-1), \hat{\zeta}(k-1|k-2), \ldots, \hat{\zeta}(k-N|k-N-1)\right]^T \end{aligned} \tag{3.14}$$

where $\phi(k) = [y(k-1), y(k-2), \ldots, y(k-n), \boldsymbol{F}^T(\boldsymbol{u}(k-1)), \boldsymbol{F}^T(\boldsymbol{u}(k-2)), \ldots, \boldsymbol{F}^T(\boldsymbol{u}(k-q))]^T$ is a directional function vector with $q \leq n$. $\omega = \left[a_1, a_2, \ldots, a_n, b_1^1, b_1^2, \ldots, b_1^m, b_2^1, b_2^2, \ldots, b_2^m, \ldots, b_q^1, b_q^2, \ldots, b_q^m\right]$ is the weights of vector $\phi(k)$. $\boldsymbol{\zeta}(k)$ is the noise vector and is subjected to a Gaussian distribution. $\boldsymbol{\omega}$ can become quite large which would require a lot of weights to be identified and this could lead to a drastic amount of over fitting [26]. To avoid this problem hyperparameters are introduced. Then the weights can be estimated as in equation (3.15).

$$\begin{aligned} \boldsymbol{\omega}_{est} &= \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{x}_I \\ \boldsymbol{\Sigma} &= \left(\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \tilde{\boldsymbol{A}}\right)^{-1} \end{aligned} \tag{3.15}$$

where $\tilde{\boldsymbol{A}} = diag(\alpha_1, \ldots, \alpha_N)$ and β are the hyperparameters and $\mathbf{x}_I$ assumed to be $\phi(k+1)$[1]. This should make it possible to predict the human desired position based and the general equation is shown in equation (3.16).

$$\tilde{\boldsymbol{X}}_d(k+1) = \boldsymbol{S}_x\tilde{\boldsymbol{x}}(k) + \boldsymbol{S}_u\boldsymbol{U}(k) \tag{3.16}$$

where $\tilde{\boldsymbol{X}}_d(k+1) = [x_d(k+1|k), x_d(k+2|k), \ldots, x_d(k+H_p|k)]^T$ is the vector of the next H_p prediction outputs, $\boldsymbol{U}(k) = [\boldsymbol{F}^T(\boldsymbol{u}(k|k)), \boldsymbol{F}^T(\boldsymbol{u}(k+1|k)), \ldots, \boldsymbol{F}^T(\boldsymbol{u}(k+H_u-1|k))\ ]^T$ is an input vector of the next H_u steps. Matrices $\boldsymbol{S}_x$ and $\boldsymbol{S}_u$ consist of mathematical terms of the estimated $\boldsymbol{\omega}$ and are explain in more detail in [27].

NARX - NARX Network

The NARX model is an extension of the ARX Model and adds a nonlinearity component. This makes it possible to be more complex. NARX model can be described mathematically[28][25], as in equation (3.17).

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n_y), u(t-1), u(t-2), \ldots, u(t-n_u)) \tag{3.17}$$

where $y(t)$ is regressed on previous values of output signals $(y(t-1), y(t-2), \ldots, y(t-n_y))$ and previous values of independent input signals $(u(t-1), u(t-2), \ldots, u(t-n_u))$. f is a function to calculate $y(t)$. The function is an important part of the NARX model since it is the one controlling the behavior of the output. So it is important to model this correctly to get the desired behavior. This can be done in varies ways. One of the ways that it can be done is by using a feedforward neural network to approximate it, which will extend it from a NARX model to a NARX Network. An example of a two-layer feedforward network can be seen in figure 3.4.

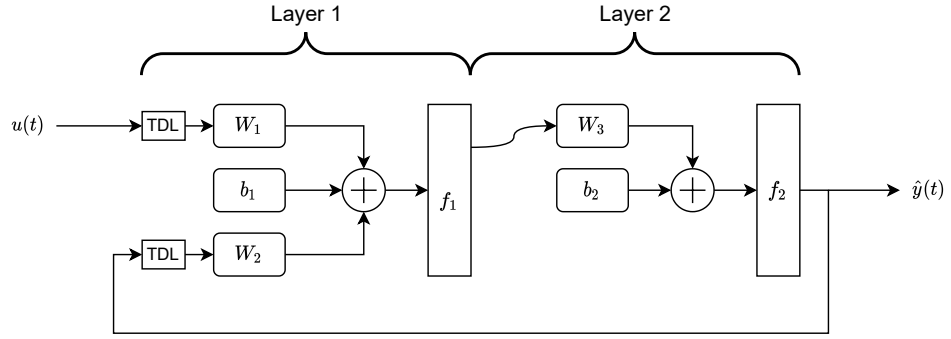


Figure 3.4: Example of a two-layer feedforward network for a NARX Network. The TDL block is a time-delay block, b_1 and b_2 is bias terms, W_1 and W_2 are weights, $u(t)$ is the input, and $\hat{y}(t)$ is the estimated output.

[1]This is not clear in the paper and is therefore an assumption.

The number of layers in f_1 can be chosen by the designer depending on the desired behavior. The NARX Network can also be used for a variety of things such as predicting the next value of input signal, nonlinear filtering, modeling of nonlinear dynamic systems, and predicting future outputs of time series values. The latter part is the focus of the NARX Network used in this thesis.

Since a neural network can be used to approximate f it has to be trained in order to work properly. This gives the option of choosing a parallel- or series-parallel architecture when training the network since both the true input and output data will be available. In a parallel architecture, the output of the network is fed back as input whereas in a series-parallel architecture it is the true output that is fed in instead. See figure 3.5 for visualization.

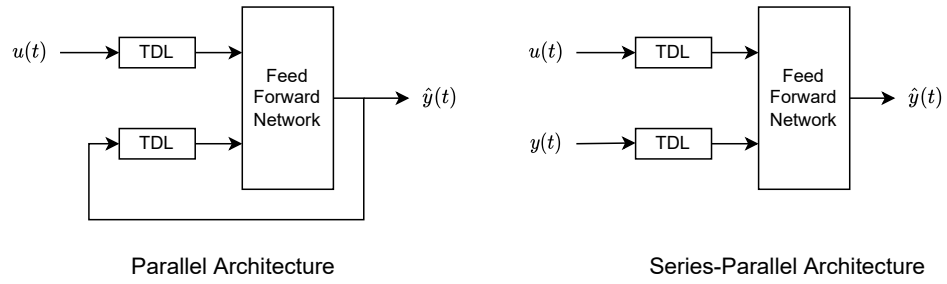


Figure 3.5: Parallel and Series-Parallel Architecture. The TDL block is a time-delay block.

It is often preferred to used the series-parallel architecture under training since the input that is fed forward is more accurate and it allows for use of static backpropagation in training. After training the real output is of course fed back as a closed-loop system.

Method choice

Choosing which method to implement is not easy since each method has its advantages and disadvantages. Some advantages of the SBL Human Intention Predictor and NARX Network is that they have nonlinearity which could be beneficial for the prediction of the human motion since human motion is not always simple or linear, as the reason, there is used a fifth-order polynomial for generating the minimum jerk trajectories (see section 3.1). This is also one of their disadvantages since it makes their models more complex which makes implementation harder and requires more fine-tuning. In the case of the NARX Network it has to be trained before it can be used, whereas the other two models do not require this. Training can be time-consuming and there needs to be sufficient data for it to be successful. Furthermore, it takes time to fine-tune all the parameters so it fits the correct purpose. The estimation of the ARX is the most efficient of the polynomial estimation methods since its results of solving linear regression equations in analytical form [29]. The

solution from the ARX Model is also unique and satisfies the global minimum of once loss function. The disadvantage of the ARX Model is that there is no disturbance part in the model. Therefore, when the disturbance is not white noise the estimation can be biased.

Since each of the methods has its advantages and disadvantages they are all chosen to be implemented and tested. Another reason for doing this is that it could be interesting to see how the different methods perform compared to each other. The implementation of the SBL Human Intention Predictor was not successful in predicting useful results. The failure is believed to be lack of information, so it is not possible to implement the full model as intended. Also, the paper had a few mathematical errors which resulted in different assumptions that had to be made in order for the model to work. Furthermore, time constraints for the project did not allow for further investigation into the mistakes made and therefore no time remained to fix them. For these reasons, the method will not be discussed any further and only the other two methods were implemented and tested.

Implementation

The ARX Model and the NARX Network have both been implemented in MATLAB/Simulink.

ARX Model

The ARX Model is implemented for each positional axis, meaning that there is one for the x-,y-, and z-axis respectively. The implementation is straight forward and uses two inbuilt MATLAB methods which are called *arx*[2] and *forecast*[3]. Since it is desired to do the estimation online the whole force input cannot be used at once and therefore needs to be used iteratively. The implementation of this can be seen in algorithm 1.

[2]arx - System Identification Toolbox - `https://se.mathworks.com/help/ident/ref/arx.html`

[3]forecast - System Identification Toolbox - `https://se.mathworks.com/help/ident/ref/idmodel.forecast.html`

Algorithm 1 ARX Model

Input: $f \leftarrow$ Force Input, $p \leftarrow$ Start Predictions, min_f, min_p
Output: $y \leftarrow$ Predictions
  $f_{list} \leftarrow$ empty list
  $f_{list,future} \leftarrow$ empty list
  $p_{list} \leftarrow$ empty list
  loop
    if $Length(p_{list}) \leq min_p$ **then**
      $p_{list}.append(p)$
      $f_{list}.append(f)$
    end if
    if $Length(f_{list,future}) = min_f$ **then**
      $system \leftarrow arx(f_{list}, p_{list})$
      $y \leftarrow$ forecast$(system, f_{list}, p_{list}, f_{list,future})$
      $p_{list}.extend(y)$
      $f_{list}.extend(f_{list,future})$
      $f_{list,future} \leftarrow$ empty list
    else
      $f_{list,future}.append(f)$
    end if
  end loop

In algorithm 1 there is a constant flow of force inputs, f, coming from a force sensor or data stream. There is a start prediction, p, which is required for starting the algorithm and it is the current start position of the robot position. min_f is the minimum number of force readings needed to predict, meaning they are how many samples are desired to predict min_f into the future. min_p is the minimum number of start predictions needed before the first true prediction can happen.

NARX Network

The implementation of the NARX Network is straight forward since MATLAB has inbuilt functionality for it. Using the MATLAB function *narxnet*[4] it is possible to setup the network. Then using the function *preparets*[5] it was possible to prepare the training data for training the network. Training the network was done using the function *train*[6]. When the network has been trained the function *closedloop*[7] was used to make a feedback loop in the

[4]narxnet - Deep Learning Toolbox - https://se.mathworks.com/help/deeplearning/ref/narxnet.html

[5]preparets - Deep Learning Toolbox - https://se.mathworks.com/help/deeplearning/ref/preparets.html

[6]train - Deep Learning Toolbox - https://se.mathworks.com/help/deeplearning/ref/network.train.html

[7]closeloop - Deep Learning Toolbox - https://se.mathworks.com/help/deeplearning/ref/closeloop.html

network, so it was able to predict future outputs.

The implementation and training of the NARX Network was done so it only predictions on one axis just like the ARX Model. So for prediction on all three axis it needs to be executed separately on each axis.

Test and Results

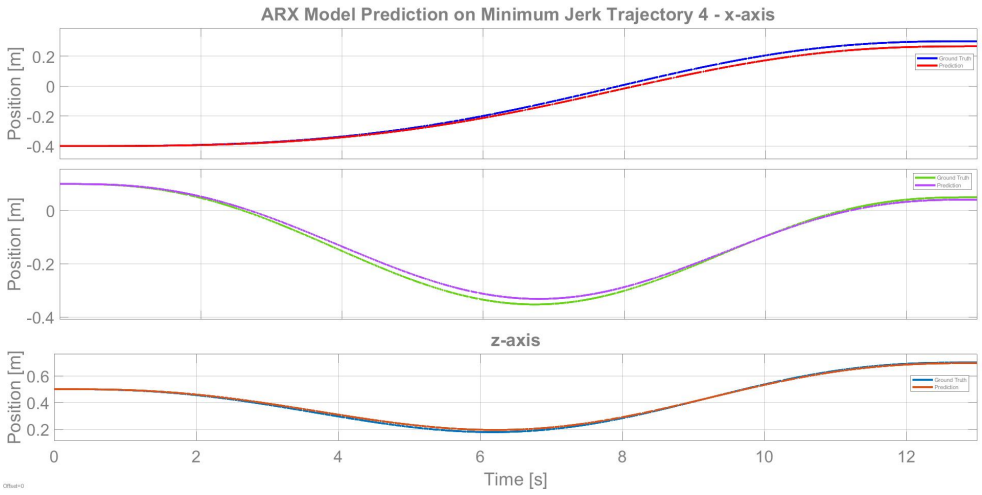
Each of the methods implemented has been tested on the four minimum jerk trajectories generated in section 3.1 and on the trajectories recorded on the physical UR10e. Each trajectory is compared to the ground truth to see the accuracy. The test has only been completed in simulation and not on the physical robot because of time constraints.

The mass used for calculate the force for the minimum jerk trajectories is 1kg for the ARX Model and 5kg for the NARX Network.

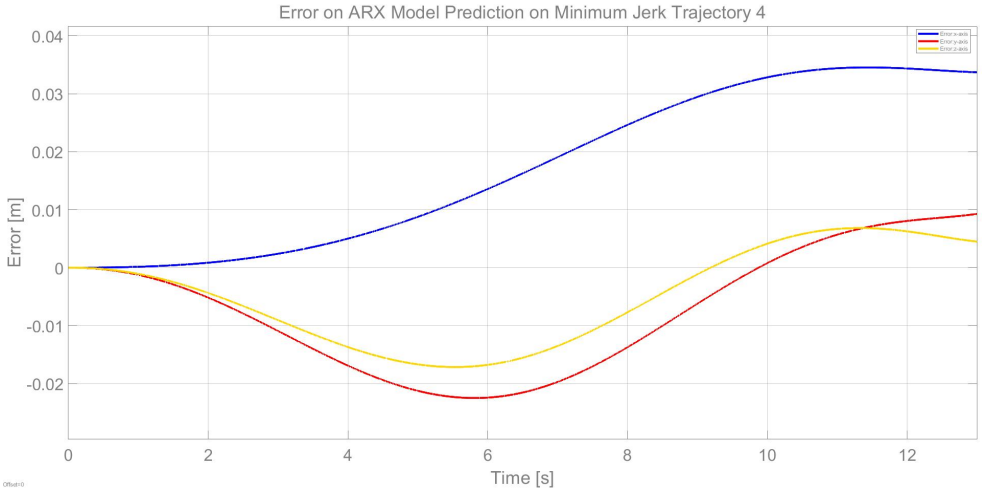
ARX Model

The ARX Model has been tested on the minimum jerk trajectories and the physical trajectories. For each test the prediction for each axis is plotted with the ground truth and a plot of the error between the two is also shown. Only the test for the minimum jerk trajectory 4 and the physical trajectory 1 will be shown, see figure 3.6 and 3.7 respectively. The rest of the trajectories can be found in Appendix A.3 (figures A.7, A.8, A.9, A.10, A.11,A.12).

Besides this different prediction length (min_f from algorithm 1) has also been tested so it is possible to see how few data points are needed to have accurate prediction. Since a lower number of data points needed would results in less delay causing the system to be more responsive. This has only been test on the minimum jerk trajectories since they performed better than the physical ones. Prediction lengths 20, 50, 75, 100, and 150 has been tested. Table 3.2 shows the max error for each trajectory and their axis. Table 3.3 shows the error at the last prediction step, so at the end of the trajectory. Table 3.4 shows the mean error over the whole trajectory.

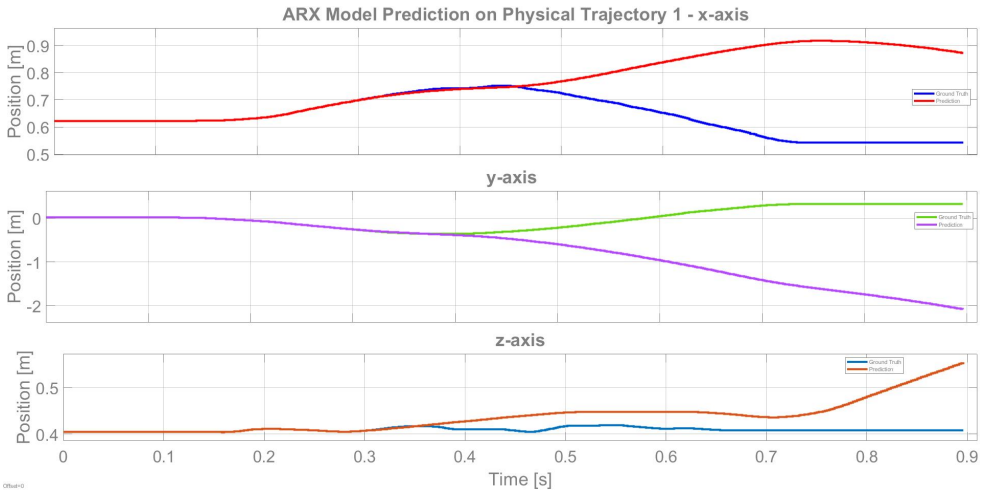


(a) Prediction and Ground truth

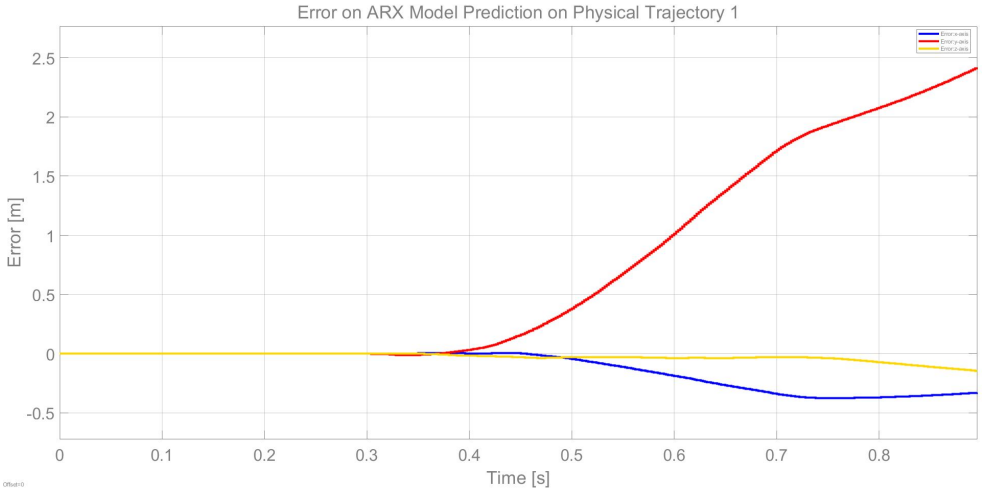


(b) Error between prediction and ground truth

Figure 3.6: ARX Model prediction of minimum jerk trajectory 4.



(a) Prediction and Ground truth



(b) Error between prediction and ground truth

Figure 3.7: ARX Model prediction of physical trajectory 1.

	Max Error					
	Axis	**20**	**50**	**75**	**100**	**150**
MJT 1	**x-axis**	0.0260	0.0214	0.0254	0.0294	-0.0363
	y-axis	0	0	0	0	0
	z-axis	0.0083	0.0065	0.0076	0.0088	-0.0109
MJT 2	**x-axis**	0.0315	0.0360	-0.0549	-0.1030	-0.1431
	y-axis	0	0	0	0	0
	z-axis	0.0095	0.0108	-0.0165	-0.0309	-0.04291
MJT 3	**x-axis**	0.1202	0.0941	-0.1426	-0.2252	-0.4236
	y-axis	0.0996	0.0986	0.1355	0.1799	0.2807
	z-axis	0	0	0	0	0
MJT 4	**x-axis**	0.0345	0.0200	0.0217	0.0253	0.0338
	y-axis	-0.0225	0.0163	0.0196	0.0305	0.0551
	z-axis	-0.0171	0.0127	0.0223	0.0329	0.0594

Table 3.2: Maximum errors on different predictions lengths (20, 50, 75, 100, 150). MJT stands for minimum jerk trajectory.

	End Error					
	Axis	**20**	**50**	**75**	**100**	**150**
MJT 1	**x-axis**	0.0236	0.0045	-0.0052	-0.0160	-0.0363
	y-axis	0	0	0	0	0
	z-axis	0.0083	0.0024	-0.0016	-0.0048	-0.0109
MJT 2	**x-axis**	0.0196	-0.0215	-0.0549	-0.1030	-0.1431
	y-axis	0	0	0	0	0
	z-axis	0.0059	-0.0064	-0.0165	-0.0309	-0.0429
MJT 3	**x-axis**	-0.0071	-0.0446	-0.0988	-0.1726	-0.3735
	y-axis	0.0027	0.0171	0.0374	0.0645	0.1358
	z-axis	0	0	0	0	0
MJT 4	**x-axis**	0.0337	0.0126	0.0076	0.0046	0.0010
	y-axis	0.0093	0.0144	0.0076	0.0139	0.0314
	z-axis	0.0045	0.0013	0.0042	0.0088	0.0289

Table 3.3: End errors (error at last prediction) on different predictions lengths (20, 50, 75, 100, 150). MJT stands for minimum jerk trajectory.

	Mean Error					
	Axis	**20**	**50**	**75**	**100**	**150**
MJT 1	**x-axis**	0.0160	0.0122	0.0125	0.0121	0.0118
	y-axis	0	0	0	0	0
	z-axis	0.0051	0.0039	0.0037	0.0036	0.0035
MJT 2	**x-axis**	0.0198	0.0144	0.0102	-0.0003	-0.0064
	y-axis	0	0	0	0	0
	z-axis	0.0059	0.0043	0.0031	-0.0001	-0.0019
MJT 3	**x-axis**	0.0451	-0.0027	-0.0360	-0.0755	-0.1769
	y-axis	0.0295	0.0198	0.0263	0.0381	0.0723
	z-axis	0	0	0	0	0
MJT 4	**x-axis**	0.0171	0.0111	0.0116	0.0129	0.0166
	y-axis	-0.0076	0.0003	0.0006	0.0042	0.0127
	z-axis	-0.0050	-0.0001	0.0032	0.0069	0.0173

Table 3.4: Mean error on different predictions lengths (20, 50, 75, 100, 150). MJT stands for minimum jerk trajectory.

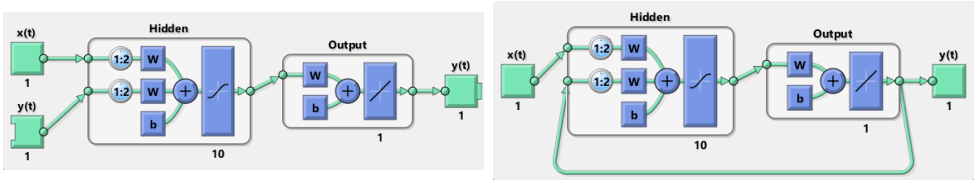
For the ARX Model the three parameters n_a, n_b, and n_k has been chosen to $n_a = 5$, $n_b = 1$, and $n_k = 1$. The reason for choosing these numbers is that a fifth-order polynomial is used to generate the minimum jerk trajectories, so a fifth-order polynomial is chosen for the ARX Model estimation. Also, a delay (n_k) of 1 is because the input has an immediate effect on the output and is not delayed.

NARX Network

For training of the NARX Network 1000 minimum jerk trajectories were generated, where each trajectory has three-axis, resulting in a true count of 3000 minimum jerk trajectories. For each trajectory, a random number of waypoints in the interval $[2; 5]$ was picked and for each waypoint, its position was randomly generated in the range $[-0.5; 0.5]$. For each waypoint, its timestamp was randomly generated as an integer with a minimum margin to the last timestamp of 1 second and a maximum margin to the last timestamp of 7 seconds. This resulted in the 3000 randomly generated minimum jerk trajectories. Then the NARX Network was trained on all of these and the training parameters can be seen in table 3.5. See figure 3.8 for visualization of network.

Epochs for training	~ 5000
inputDelays	1:2 (default)
feedbackDelays	1:2 (default)
hiddenSizes	10 (default)
feedbackMode	open (default)
trainFcn	trainlm - Levenberg-Marquardt Algorithm (default)
train/val/test ratio	0.7/0.15/0.15 (default)

Table 3.5: Narx Network training parameters.



(a) Open Loop Network.

(b) Close Loop Network.

Figure 3.8: Diagram of the NARX Network used with its open loop in training and closed-loop after training.

The NARX Network was tested on all of the four minimum jerk trajectories and on all of the four physical trajectories. For each test the prediction for each axis is plotted together with its ground truth and the error between the two is also plotted. Only the test from the minimum jerk trajectory 3 and the physical trajectory 1 is shown, see figure 3.9 and figure 3.10 respectively. The remaning tests can be seen in Appendix A.3 (figures A.13, A.14, A.15, A.16, A.17, A.18).

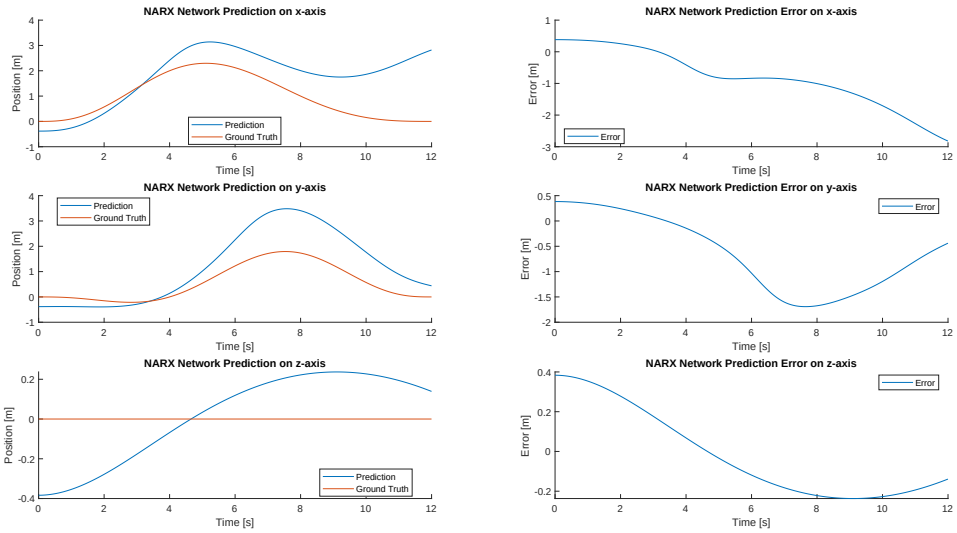


Figure 3.9: NARX Network prediction on minimum jerk trajectory 3.

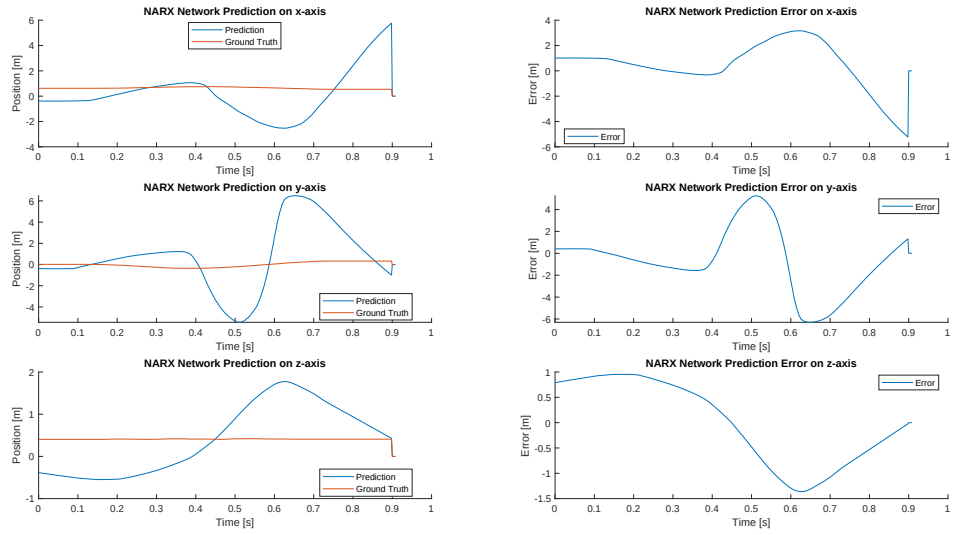


Figure 3.10: NARX Network prediction on physical trajectory 1.

Discussion

The ARX Model predictions works fairly well on the generated minimum jerk trajectories with only a few centimeters as error (figures A.7, A.8, A.9, 3.6). Where the maximum error seen on all the minimum jerk trajectories is -42.36cm and its for trajectory 3 with 150 prediction length. This seems

quite larger, but appears to be an extreme outlier and the other maximum errors are substantially smaller. It can be seen that the estimation is a little delayed compared to the ground truth, but this is to be expected since it has to use previous data to predict into the future. So a true real-time estimation will not be possible unless a single step is predicted into the future each time. But this is not applicable in this case since the implementation methods used requires at least a few data points depending on the the polynomial order used. From table 3.2 it can be seen that generally a larger prediction length such as 100 and 150 result in a larger maximum error. However, a smaller one does not always result in the lowest maximum error. So from the maximum error, it seems that the best prediction length is somewhere in the middle values. From table 3.3 it is hard to say if a smaller or larger prediction length results in a better end error since it appears to be dependable on which trajectory the ARX Model is predicting on. However, it seems that the worst end errors either happen at prediction length 20 or 150, so here the middle values appear the best again. It can also be seen that depending on the prediction length the end error can switch signs. This makes sense since the longer into the future the model has to predict the more inaccurate it gets. From table 3.4 it can be seen that for most of the trajectories the mean error does not change much depending on the prediction length, but only does for a few of them. But here it is hard to see if there is a pattern for it. From these three tables, it seems that the middle values, primarily 50, are the best values.

The ARX Model predictions on the physical trajectories does not work well and the prediction length had to be turned up to at least 150 to have some response from the model (figures 3.7, A.10, A.11, A.12). This is quite unfortunate but could be because of the noise in the force sensor data. As said earlier the ARX Model does not contain a disturbance part in its model and therefore the noise and drift in the force sensor reading could cause the model to be unstable. It can be seen that the model predicts accurately at the start, but after some time it diverges too far away from the ground truth.

The NARX Network work on some of the minimum jerk trajectories and not at all on the other (figures A.13, A.14, 3.9, A.15). It can be seen that there is potential for the network to be able to predict the correct output with potentially more training and a better data set. Also from the figures, it can be seen that the network struggles with finding the correct start values, but the motion is fairly close, but this just offsets the whole prediction. It can be seen that the NARX Network is not good at predicting no change in motion since it seems to always expect a motion to occur. This is likely due to the construction of the data set used to train the network since it does not contain any trajectories with no change. This is an improvement that could be made, but due to time constraints is not possible for this project.

The NARX Network does not work on the physical trajectories (figures 3.10, A.16, A.17, A.18). It can be seen that the network is unstable in its prediction and can not converge to a stable point. This can be due to several

factors such as the network not being trained thoroughly enough, the training data not being inclusive enough, training the network to be overfitted to the training data, or that the force data from the physical trajectories has a lot more noise and are not as smooth as the once from the minimum jerk trajectories. Since there is a difference in the forces from the minimum jerk model and the physical recorded ones it could be hard for the network to learn the behavior of the physical one. Also because of the difference, it is likely that the network was overfitted to the training data. An improvement that could be done is to also train the network on physical recording or add noise to the minimum jerk trajectories forces, so it mimics the real world more.

Conclusion

It can be concluded that the ARX Model prediction works well on the generated minimum jerk trajectories, but does not work well on the physical trajectories. Furthermore, the best prediction length appear to be around 50, for a more precise number further experiments has to be conducted.

The NARX Network's predictions works well on some of the minimum jerk trajectories, but not all of them and it can be concluded that there is potential for the method, but more training and experiments needs to be completed in order to confirm. It can also be concluded that the network currently does not work on the physical trajectories, likely due to overfitting.

3.3 Human Impedance

Estimating the impedance of the human is an important part of the project, because if done correctly it can greatly increase the assistance that the robot is doing for the human. Three methods will be explained in-depth in this chapter. The first is estimation the stiffness using the statistical Bayesian parameter estimation method [13], this will be referred to as the Bayesian method. The second is setting up a mathematical form for damping depending on the kinetic energy of the robot's end-effector [16] and will be referred to as variable damping. The third and last is estimating both the stiffness and damping using the frequency domain [30] and will be referred to as impedance estimation in the frequency domain (IEFD).

Methods

Bayesian method

X. Yu et al. [13] simplify the human dynamic model in h dimensional Cartesian space to a simple spring model shown in equation (3.18).

$$f_h = K_h(x_h - x) \tag{3.18}$$

where $K_h \in \mathbb{R}^{h\times h}$ is the human's stiffness matrix, $x_h \in \mathbb{R}^h$ is the human's motion intention, $x \in \mathbb{R}^h$ is the actual position of the human, and $f_h \in \mathbb{R}^h$ is the interaction force between the human and the carried object. Then the estimation of equation (3.18) in one dimension is shown in equation (3.19).

$$\hat{f}_{h1} = \hat{K}_{h1}(\hat{x}_{h1} - x) \tag{3.19}$$

where $\hat{K}_{h1}$, $\hat{x}_{h1}$, and $\hat{f}_{h1}$ denotes the estimation of the human stiffness, human motion intention, and the interaction force in one dimension.

Then they use the Bayesian parameter estimation method to estimate K_{h1} and x_{h1} by establishing a quadratic cost function shown in equation (3.20).

$$\lambda = \left(\frac{\hat{f}_{h1}(t-1) - \hat{f}_{h1}(t)}{\dot{x}_1(t)} - \frac{f_{h1}(t-1) - f_{h1}(t)}{\dot{x}_1(t)} \right)^2 \tag{3.20}$$

Here $(f_{h1}(t-1) - f_{h1}(t))/\dot{x}_1(t)$ can be regarded as K_h from equation (3.18). Then equation (3.20) can be used to get the estimate of K_h. f_{h1} and $\dot{x}_1$ can be obtained from force and velocity sensors, respectively.

It is then assumed that $(f_{h1}(t-1) - f_{h1}(t))/\dot{x}_1(t)$ follows the Gaussian distribution, so the random variable set k_1 of $(f_{h1}(t-1) - f_{h1}(t))/\dot{x}_1(t)$ obyes $k_1 \sim N\left(\mu, \sigma^2\right)$ where $N(\cdot)$ is the Gaussian distribution, μ is the mathematical expectation, and σ^2 is the variance of k_1. Then if the human stiffness parameter K_{h1} can be deemed as μ, then it is possible to estimate K_{h1} accordingly to the Bayesian parameter estimation method if σ^2 is know before hand. Then the cost funtion from equation (3.20) can be rewritten into equation (3.21).

$$\lambda = (\hat{\mu} - \mu)^2 \tag{3.21}$$

where $\hat{\mu}$ is the estimate of μ. Then the predictor probability distribution of stiffness parameter $p(\mu)$ is shown in equation (3.22).

$$p(\mu) \sim N(\mu_0, \sigma_0^2) \tag{3.22}$$

where μ_0 and σ_0^2 denotes the predictor expectation and variance of μ. Then the updater probability distribution $p(\mu|k)$ is shown in equation (3.23).

$$p(\mu|k) = \frac{p(k|\mu)p(\mu)}{\int p(k|\mu)p(\mu)d\mu} \tag{3.23}$$

where $p(k|\mu)$ is the joint probability distribution and is calculated like shown in equation (3.24).

$$p(k|\mu) = \prod_{i=1}^{n} p\left(\frac{f_{h1i}(t-1) - f_{h1i}(t)}{\dot{x}_{1i}(t)} |\mu \right) \tag{3.24}$$

where $(f_{h1i}(t-1) - f_{h1i}(t))/\dot{x}_{1i}$ is the ith element of a set k. Substituting equation (3.22) and (3.24) into (3.23) and remember that $p(k|\mu)$ and $p(\mu)$

follows the Gaussian distribution the updater probability distribution can be obtained and is shown in equation (3.25).

$$p(\mu|k) = \frac{1}{\sqrt{2\pi}\sigma_n}\exp\left(-\frac{1}{2}\frac{(\mu-\mu_n)^2}{\sigma_n^2}\right) \sim N\left(\mu_n, \sigma_n^2\right) \tag{3.25}$$

where μ_n and σ_n^2 is shown in equation (3.26).

$$\begin{aligned}
\mu_n &= \frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0 \\
\hat{\mu}_n &= \frac{1}{n}\sum_{i=1}^{n}\frac{f_{h1i}(t-1) - f_{h1i}(t)}{\dot{x}_{1i}(t)} \\
\sigma_n^2 &= \frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2}
\end{aligned} \tag{3.26}$$

Then the use of the quadratic cost function from equation (3.20) makes it possible to describe the Bayesian parameter estimation $\hat{\mu}$ as the conditional expectation when k is given and μ can be estimated as shown in equation (3.27).

$$\hat{\mu} = \int \mu p(\mu|k)d\mu = \int \mu\frac{1}{\sqrt{2\pi}\sigma_n}\exp\left(-\frac{1}{2}\frac{(\mu-\mu_n)^2}{\sigma_n^2}\right)d\mu = \mu_n \tag{3.27}$$

Then the Bayesian estimation of μ can be rewritten in to equation (3.28).

$$\begin{aligned}
\hat{\mu} &= \frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0 \\
\hat{\mu}_n &= \frac{1}{n}\sum_{i=1}^{n}\frac{f_{h1i}(t-1) - f_{h1i}(t)}{\dot{x}_{1i}(t)} \\
\hat{\sigma}^2 &= \sigma_n^2 = \frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2}
\end{aligned} \tag{3.28}$$

The estimate of the human stiffness parameter $\hat{K}_{h1}$ remains in the interval from $(\hat{\mu}-\hat{\sigma})$ to $(\hat{\mu}+\hat{\sigma})$. This will estimate the stiffness in one dimension and can easily be scaled up to be done on h dimensions.

Impedance estimation in the frequency domain

M. J. Fu et al. [30] focus on modeling the 3D dynamics of the human arm and the inter and intrasubject variability due to human variation and grip force strength respectively, as a function of frequency. This is all done with experiments done on human subjects. The relevant work for this thesis is their arm dynamic model structure and how they estimate its parameters.

They conceptualize the human arm after a mass-spring-damper (MSD) system with five parameters; one mass (M), two dampers (b_1, b_2), and two springs (k_1, k_2). Their reason for this is because other studies have used similar methods, MSD systems can easily be simulated, and it is appropriate for their work. The mass represents the inertia of the arm, damper b_1 and spring k_1 represent the grasp stiffness of the human hand, and damper b_2 and spring k_2 represent the stiffness of the human arm. They do all of their work in the frequency domain therefore they present a transfer function for the human arm (H_{arm}) with force measurement (F_{sensor}) as input and the position of the human hand (X_{arm}) as output all based on the five parameters from the MSD system. See equation (3.29) and for full details of how the transfer function was derived see [30].

$$H_{arm}(s) = \frac{X_{arm}(s)}{F_{sensor}(s)} = \frac{Ms^2 + (b_1 + b_2)s + k_1 + k_2}{b_1 Ms^3 + (b_1 b_2 + k_1 M)s^2 + (b_2 k_1 + b_1 k_2)s + k_1 k_2} \tag{3.29}$$

The transfer function is fitted to measure human experiment frequency response in each positional axis, so it is possible to identify the five parameters of the MSD system. They use Welch's transfer function to compute the frequency response with a 32 Hamming-windowed segments and 50% overlap to minimize fast Fourier transform artifacts. Each fit is performed using nonlinear constrained optimization using the cost function in equation (3.30).

$$\sum_{n=1}^{p} W_t(n) \left(H_{exp}\left(j2\pi \frac{n}{N}\right) - H_{arm}\left(j2\pi \frac{n}{N}\right)\right)^2 \tag{3.30}$$

where $W_t(n)$ is a weighting function, H_{exp} is the frequency response from the experiments, H_{arm} is the output of equation (3.29) with the identified parameters, p is the number of data points, and N is the total number of frequency response points from the 32-segment Welch frequency response estimation. The weighting function is a mean-squared coherence of the force input and position output.

Variable Damping

T. Bitz et al. [16] presents a variable damping controller that uses user intent which they define as the product of the robots end-effector velocity ($\dot{x}$) and its acceleration ($\ddot{x}$), $\dot{x}\ddot{x}$. This can be seen as a measure of the change in kinetic energy of the system. When $\dot{x}\ddot{x}$ is positive ($\dot{x}\ddot{x} > 0$) the user intends to initiate and accelerate the robot. Therefore, it is desired to lower the damping (b_r) since this will aid enchanted agility and yield a faster response by the system. Then on the other hand when $\dot{x}\ddot{x}$ is negative ($\dot{x}\ddot{x} < 0$) the user intends to decelerate the robot and it is therefore desired to increase the damping to assist with stability. To accommodate these desires and restrict the damping

within a certain range, a dual-sided logistic function is designed. The function can be seen in equation (3.31).

$$b_r = \begin{cases} -b_{lb} + \frac{2b_{lb}}{1+e^{-k_p \dot{x}\ddot{x}}} + b_c, & \dot{x}\ddot{x} \geq 0 \\ b_{ub} - \frac{2b_{ub}}{1+e^{-k_n \dot{x}\ddot{x}}} + b_c, & \dot{x}\ddot{x} < 0 \end{cases} \tag{3.31}$$

$b_{lb} < 0$ and $b_{ub} > 0$ is the lower and upper bound of the damping, respectively. b_c is the center damping value, which is accomplish when $\dot{x}\ddot{x} = 0$. k_p is a tuning parameters which controls how quickly the damping varies from the lower bound to the center value, $[b_{lb} + b_c, b_c]$. k_n has the same functionality just from the center value to the upper bound, $[b_c, b_{ub} + b_c]$. Equation (3.32) shows how to determined k_p and k_n.

$$\begin{aligned} k_p &= \frac{\ln\left(\frac{1-s}{1+s}\right)}{(\dot{x}\ddot{x})_{\max}} \\ k_n &= \frac{\ln\left(\frac{1+s}{1-s}\right)}{(\dot{x}\ddot{x})_{\min}} \end{aligned} \tag{3.32}$$

s is a constant for sensitivity measure. $(\dot{x}\ddot{x})_{\max}$ and $(\dot{x}\ddot{x})_{\min}$ is the maximum and minimum value of $\dot{x}\ddot{x}$ during normal movements. Choosing k_p and k_n using this method, assist in speed variability over different subject using the system. With the sensitivity measure introduced the damping b_r will be $sb_{lb} + b_c$ at $\dot{x}\ddot{x} = (\dot{x}\ddot{x})_{\max}$ and $sb_{ub} + b_c$ at $\dot{x}\ddot{x} = (\dot{x}\ddot{x})_{\min}$.

Method choice

The Bayesian method and IEFD is both method that are not applicable for online use. Especially the IEFD method since it is done in the frequency domain it requires all of the inputs and outputs to be know before estimation can occur. This is not ideal for this project since it is desired to estimate impedance online. However, this method could still be used if it is modified so the estimation is conducted over a smaller size of the demonstration. This would enable the method to be able to estimate online. This would of course not give a real time estimation, but it could give an estimation often enough to potentially be useful. This concept is interesting and is therefore chosen to be implemented and tested. The Bayesian method is not chosen since it is not a method useful for online use and only estimates the stiffness of the impedance and it is therefore preferable to use the IEFD method as this has the possibility to estimate all the desired parameters for impedance.

The variable damping method is also an interesting method since its idea is to enchanted the agility of the system and yield a faster response time. However this is not an estimation method, but rather an heuristic method for changing the damping accordingly to the robots movement. Also the method

only gives a behavior for the damping, but a behavior for the stiffness could also be derived from this by ensuring stability in the admittance controller (see section 4.1). Therefore, is this method chosen as well and the stiffness part will be elaborated on under implementation.

Implementation

Impedance estimation in the frequency domain
When implementing IEFD a number of inbuilt MATLAB functions are used. When computing the Welch's transfer function the MATLAB function *tfestimate*[8] is used. The nonlinear constrained optimization is done using the MATLAB function *fmincon*[9]. To get the weight function and for it to be mean-squared coherence the MATLAB function *mscohere*[10] is used. The algorithm for estimation the parameters for a sequency of forces and positions can be seen in Algorithm 2.

Algorithm 2 Impedance estimation in the frequency domain

Input: $f \leftarrow$ Force Input, $p \leftarrow$ Position Input
Output: M, b_1, k_1
$w_{segment} \leftarrow$ 32Hamming-windowed
$x0 \leftarrow$ Start Guess
$H_{exp} \leftarrow tfestimate(f, p, w_{segment}, ...)$
$W_t \leftarrow mscohere(f, p, w_{segment}, ...)$
$func \leftarrow cost_{function}(H_{exp}, W_t, ...)$
M, b_1, $k_1, \leftarrow fmincon(func, x0, ...)$

Instead of the five parameters MSD system used originally for this work, it is chosen to simplify it and use a standard three parameters MSD system with one mass (M), one spring (k_1), and one damper (b_1) instead just as K. Kosuge et al. [31]. The transfer function is shown in equation (3.33).

$$H_{arm}(s) = \frac{X_{arm}(s)}{F_{sensor}(s)} = \frac{1}{Ms^2 + bs + k} \tag{3.33}$$

For validating that the estimation is accurate a MSD system is simulated with the position as input and the force as output, using the estimated parameters. When a comparison between the input force used for estimation and the expected output force (f_{exp}) from the simulated MSD system can be done. To simulate the MSD system is quite simple and equation (3.34) is used.

[8] tfestimate - Signal Processing Toolbox - https://se.mathworks.com/help/signal/ref/tfestimate.html
[9] fmincon - Optimization Toolbox - https://se.mathworks.com/help/optim/ug/fmincon.html
[10] mscohere - Signal Processing Toolbox - https://se.mathworks.com/help/signal/ref/mscohere.html

$$f_{exp} = k_{est}x + b_{est}\dot{x} \tag{3.34}$$

where k_{est} is the estimated spring stiffness, b_{est} is the estimated damping, and x and $\dot{x}$ is the position and velocity respectively. The reason the mass is not present is that essentially the position is acting on the mass and it therefore does not have an effect.

Variable Damping
Implementing the variable damping is straightforward since it is just feeding the outputs of the robot end-effector to the mathematical equation. The implementation has been done in MATLAB/Simulink like the other implementation in this project.

However, to ensure stability in the admittance controller the mass and stiffness also need to be chosen. The mass will be the same used to calculate the force for the minimum jerk trajectory since the assumption was made there (see section 3.1). Then equation (3.35) can be used to calculate the stiffness for each axis with a damping ratio of 1. This equation is derived from the stability of the admittance controller (see equation (4.5)) that is characterized after a second-order system. This will be explained in more detail in section 4.1.

$$k_{vp} = \frac{d_{vp}^2}{4m_{vp}} \tag{3.35}$$

where m_{vp}, k_{vp}, and d_{vp} is the mass, spring, and damping respectively.

Test and Results

Impedance estimation in the frequency domain
The IEFD method is test on all four minimum jerk trajectories and the four physical trajectories. The method is first tested on the trajectories with the full data and then it is tested on parts of the data in sequence. For each test the expected force is plotted against the original force used. This will only be showed for the minimum jerk trajectory 3 and the physical trajectory 2. This can be seen in figure 3.11 and figure 3.12 respectively. The remaining figures (A.19, A.20,A.21, A.22, A.23, A.24) can be seen in Appendix A.4. The parameter estimation for each axis for all the trajectories can be seen in table 3.6 for the minimum jerk trajectories and table 3.7 for the physical trajectories.

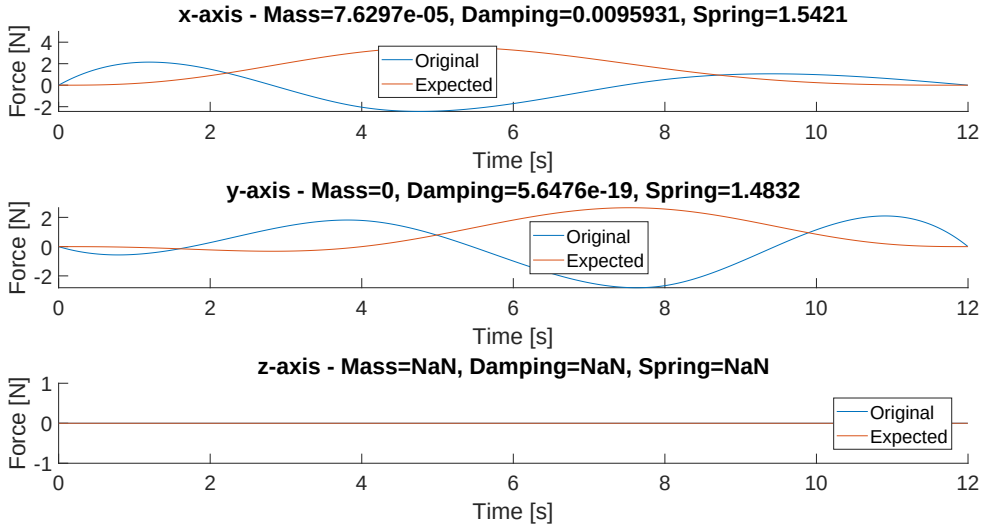


Figure 3.11: Impedance estimation of minimum jerk trajectory 3 conducted on the whole trajectory using the IEFD method.

Axis	**Mass**	**Damping**	**Spring**
Minimum Jerk Trajectory 1			
x	0	0	3.1688
y	NaN	NaN	NaN
z	1.8023e-05	0.0053736	1.931
Minimum Jerk Trajectory 2			
x	0	0.071518	20.1502
y	NaN	NaN	NaN
z	0	0	16.572
Minimum Jerk Trajectory 3			
x	7.6297e-05	0.0095931	1.5421
y	0	5.6476e-19	1.4832
z	NaN	NaN	NaN
Minimum Jerk Trajectory 4			
x	0	0.0057474	0.4760
y	0.0001895	0.010879	1.4632
z	0	0.019424	2.8974

Table 3.6: Estimated parameters using the IEFD method for each of the four minimum jerk trajectories and their respective axis. NaN means that the test was not run on this trajectory since it is not changing.

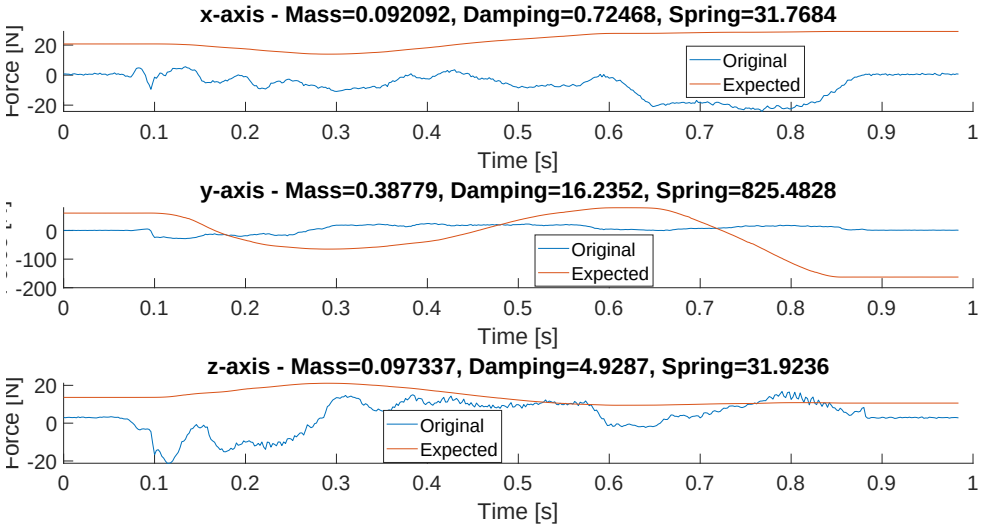


Figure 3.12: Expected force for impedance estimation of physical trajectory 2 conducted on the whole trajectory using the IEFD method.

Axis	Mass	Damping	Spring
Minimum Jerk Trajectory 1			
x	0.11193	7.4415	91.5455
y	4.4172e-16	8.8828	181.1203
z	0.092523	0.0053736	30.023
Minimum Jerk Trajectory 2			
x	0.092092	0.72468	31.7684
y	0.38779	16.2352	825.4828
z	0.097337	4.9287	31.9236
Minimum Jerk Trajectory 3			
x	0.24703	5.9004	312.91
y	0.16008	17.272	324.6022
z	0.076675	3.6656	38.252
Minimum Jerk Trajectory 4			
x	5.2638e-15	12.4423	100.4423
y	0.1015	48.7779	153.5182
z	2.7527e-05	12.5407	181.1513

Table 3.7: Estimated parameters using the IEFD method for each of the four physical trajectories and their respective axis.

The test on the sub parts of the data is only showed for minimum jerk trajectory 3 and physical trajectory 4. See figure 3.13 and figure 3.14 respectively. The remaining test of the other trajectories can be seen in the figures (A.25,

A.26, A.27, A.28, A.29, A.30) in Appendix A.4. The estimated parameters is visibly in the figures.

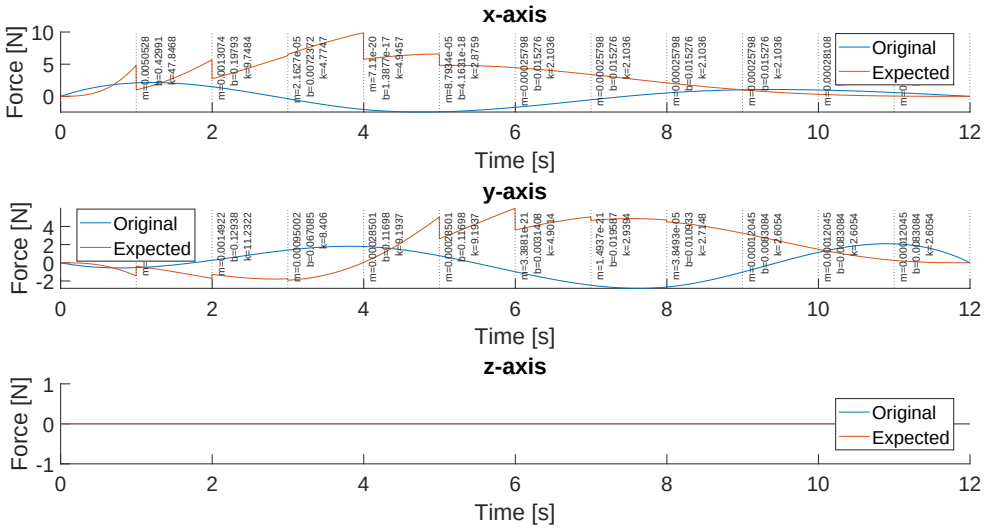


Figure 3.13: Impedance estimation of minimum jerk trajectory 3, done partly using the IEFD method.

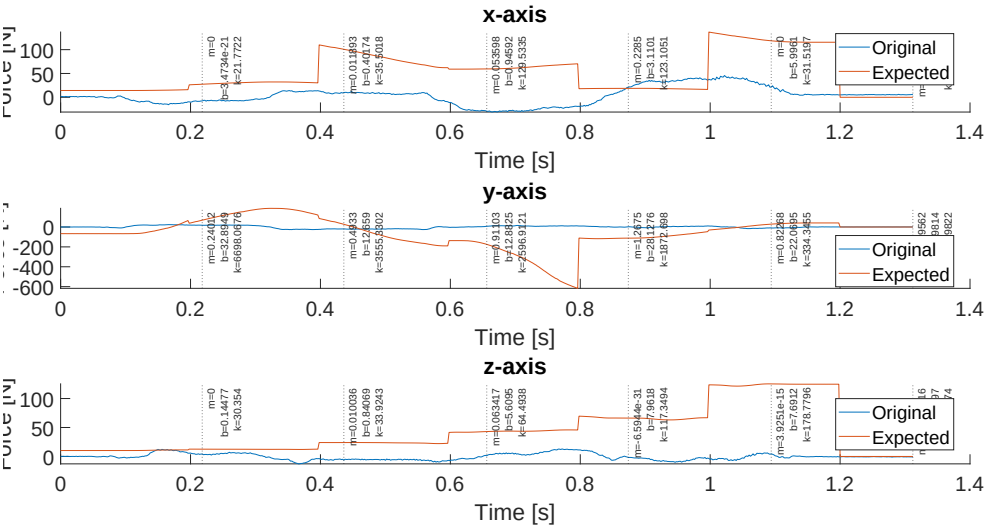


Figure 3.14: Expected force for impedance estimation of physical trajectory 4, done partly using the IEFD method.

Variable Damping
The variable damping has been tested on all four of the generated minimum jerk trajectories. For each of the trajectories the damping, stiffness, and user intent is plotted. Only the results for minimum jerk trajectory 4 will be shown, see figure 3.15. The plots from the other three trajectories can be seen in Appendix A.4 (figures A.31, A.32, A.33). The parameters used can be seen in table 3.8.

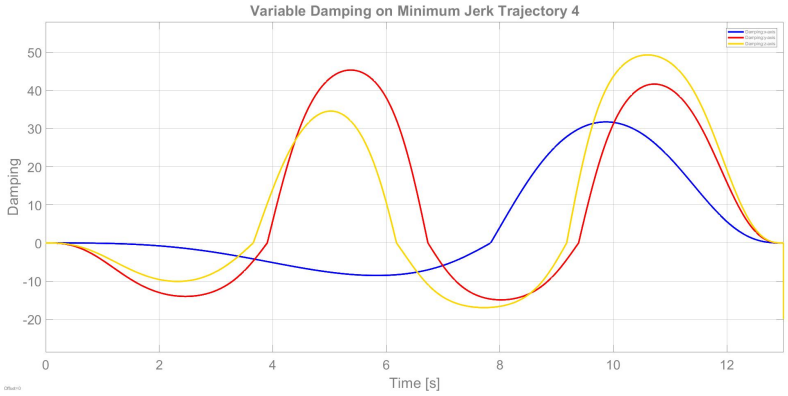
b_{ub}	b_{lb}	b_c	s
60	-20	0	0.95

Table 3.8: Parameters used for the variable damping method.

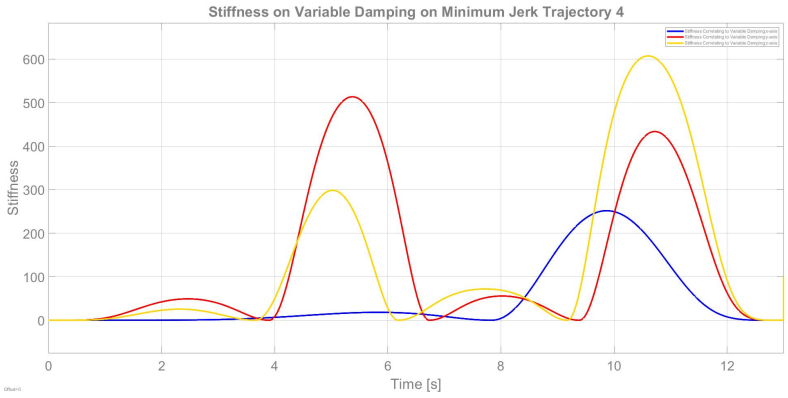
Discussion

The IEFD method managed to produce results when tested on the minimum jerk trajectories, but the validation process (figures A.19, A.20, 3.11,A.21) was not successful. From table 3.6 it can be seen that the estimated parameters on the whole trajectories was very small expect for the stiffness which seems like the only parameters the estimation was successful in estimation. Unfortunatly this alone cannot be validated. The IEFD method managed to work better on the physical trajectories even tho the validation method (figures A.22, 3.12, A.23, A.24) was not much better. But from table 3.7 the parameters seems more promising and most of them fall in line with the original work by M. J. Fu et al. [30]. The switch to using only a three parameter MSD system instead of a five parameter MSD system could might have lead to these outcomes, but when the same test was performed on the five parameter MSD system many of the estimated parameters seemed to be neglected, meaning they where often estimation to be zero or extremely small. Doing the test partly one the force and position data (figures A.25, A.26, 3.13, A.27, A.28, A.29, A.30, 3.14) did not validate successfully and the parameter estimation results was the same as behavior as when done on the whole trajectories.

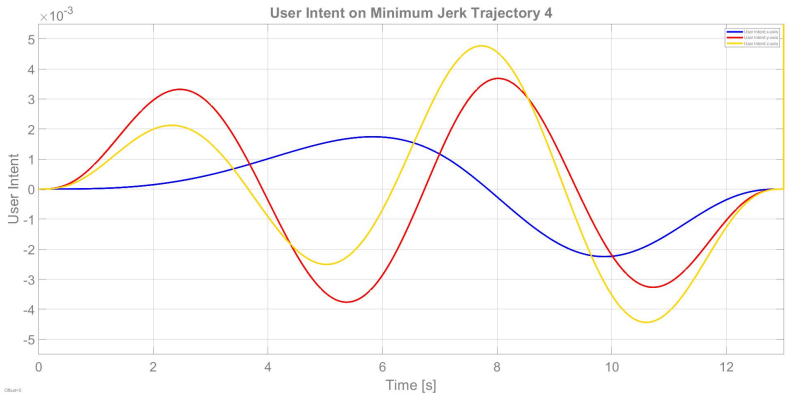
The variable damping methods works as intended on the minimum jerk trajectories (figures A.31, A.32, A.33, 3.15). But this is well with expectation since it is not an estimation method, but more a heuristic of the end-effectors' movements. It can be seen that the method works as intended since the damping decreases when $\dot{x}\ddot{x} > 0$ and increases when $\dot{x}\ddot{x} < 0$. It can also be seen that the bounds of the damping work well, which is a nice feature to restrict its limits. It can also be seen that the damping changes in accordance with the user intents magnitude and not just its behavior. This can be seen in figure A.31 and A.32 since they are the same trajectory, but the latter is done in 2 seconds instead of 5, making the movements faster (minimum jerk trajectory 1 and 2). The only thing that is changeable is the limits of the upper (b_{ub}) and lower bound (b_{lb}), but since the system, unfortunately, has not been implemented



(a) Variable damping



(b) Stiffness



(c) User intent

Figure 3.15: Variable damping, stiffness, and user intent for minimum jerk trajectory 4.

fully on a physical robot it is hard to get a fell for these parameters, since they are very dependable on what task is desired, what behavior is wanted, and how the feeling is. So there is no reason to test these further since the behavior would be the same, just the magnitude of the output will change. So that's why the parameters used in this project are the same as T. Bitz et al. [16]. The method has not been tested on the physical trajectories as it has already been confirmed that the method works as intended on the minimum jerk trajectories and since it is not an estimation method the behavior will just change with the behavior of the trajectory. However, since there often is noise on the physical trajectories this could translate into the damping.

Conclusion

It can be concluded that the IEFD does not work as intended as it cannot be validated properly. However, the method seems to work better on the physical trajectories than the minimum jerk trajectories. It can also be concluded that the variable damping works as intended.

4 Control

This chapter contains two parts. One for the admittance controller of the project, see section, 4.1 and one for the PD controller of the project, see section 4.2.

4.1 Admittance controller

The admittance controller is a key part of the project since it enables an easy way to have HRI. An admittance controller is based on the concept of a compliant frame which is a suitable reference frame for describing the ideal behavior of the robot's end-effector under impedance control [32]. A simplified block scheme of an admittance controller can be seen in figure 4.1.

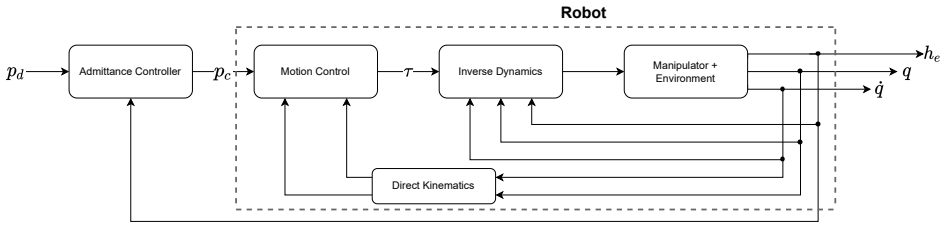


Figure 4.1: Simplified block scheme of an admittance controller.

The gray dotted box from figure 4.1 is often what is contained inside a physical robot and there usually is not an option to control these. This is one of the main reasons that an admittance controller is a great tool since it allows for controlling the robot with compliance. The admittance control can be expressed mathematically and is split into translational and rotational parts. The translational part can be seen in equation (4.1).

$$M_p \Delta \ddot{p}_{cd} + D_p \Delta \dot{p}_{cd} + K_p \Delta p_{cd} = f \tag{4.1}$$

where M_p, D_p, $K_p \in \mathbb{R}^{3\times3}$ are positive definite matrices that is the parameters of a mechanical impedance and is the mass, damping, and stiffness respectively. $\Delta p_{cd} \in \mathbb{R}^3$ is the error between the desired frame and complaint frame, $\Delta p_{cd} = p_c - p_d$. $f \in \mathbb{R}^3$ is the force. The rotational part of the admittance controller looks quite similar. See equation (4.2).

$$M_o \Delta \dot{\omega}_{cd} + D_o \Delta \omega_{cd} + K_o^* \epsilon_{cd} = \tau \tag{4.2}$$

where M_o, $D_o \in \mathbb{R}^{3\times3}$ are positive definite matrices that is the equivalent to the same three parameters for the translational part (M_p, D_p). $\tau \in \mathbb{R}^3$ is the torque. $\Delta\omega_{cd} \in \mathbb{R}^3$ is the anglur velocity between the complaint frame and desired frame, $\Delta\omega_{cd} = \omega_c - \omega_d$. $\epsilon_{cd} = \eta_d\epsilon_c - \eta_c\epsilon_d - S(\epsilon_c)\epsilon_d$ is the orientation difference expressed in quaternions. K_o^* is the rotational stiffness matrix and where K_o is the stiffness in Euler angles. See equation 4.3.

$$\begin{aligned} K_o^* &= 2E(\eta_{cd}, \epsilon_{cd})K_o \\ E(\eta, \epsilon) &= \eta I - S(\epsilon) \end{aligned} \tag{4.3}$$

Implementation

When implementing the rotational part of the admittance controller it is not possible to simply integrate the angular velocity to get the joint configuration, just as it would be done for the translational part. Instead the angular velocity can be integrated in a certain way to convert it into a quaternion, $q = (\eta_{cd}, \epsilon_{cd})$. The method for doing this is using a exponential map as shown in equation 4.4.

$$\begin{aligned} q(t+dt) &= \exp\left(\frac{dt}{2}\omega_{cd}(t+dt)\right) * q(t) \\ \exp(t) = (\eta, \epsilon) &= \left(\cos(||r||), \frac{r}{||r||}\sin(||r||)\right) \end{aligned} \tag{4.4}$$

A block diagram of the translational and rotational part can be seen in figure 4.2. There are ten inputs. Two for the force (f) and torque (τ) applied to the system, two for the desired position (p_d) and desired rotation (q_d), and six for the mass, damping and stiffness matrices for the positional (M_p, D_p, K_p) and rotational (M_o, D_o, K_o) part respectively. The controller has two outputs, one for the compliant position (p_c) and one for the compliant rotation (q_c).
Choosing M_p, D_p, and K_p or M_o, D_o, and K_o to any arbitrary values is not possible if stability of the system needs to be assured. The admittance controller is modeled after a MSD system and the dynamics are given by a second-order system and is therefore characterized by a natural frequency (ω_n) and damping ratio ζ. So the goal would be to have a damping ratio of 1 ($\zeta = 1$) so the controller would be critical stable. This is done for each axis, also making it possible to have stability in each axis with different dynamics. The equation for the natural frequency (ω_{nx}) and damping ratio (ζ_x) for the x-axis can be seen in equation 4.5. This is the same equation for the y- and z-axis, just with their respective parameters.

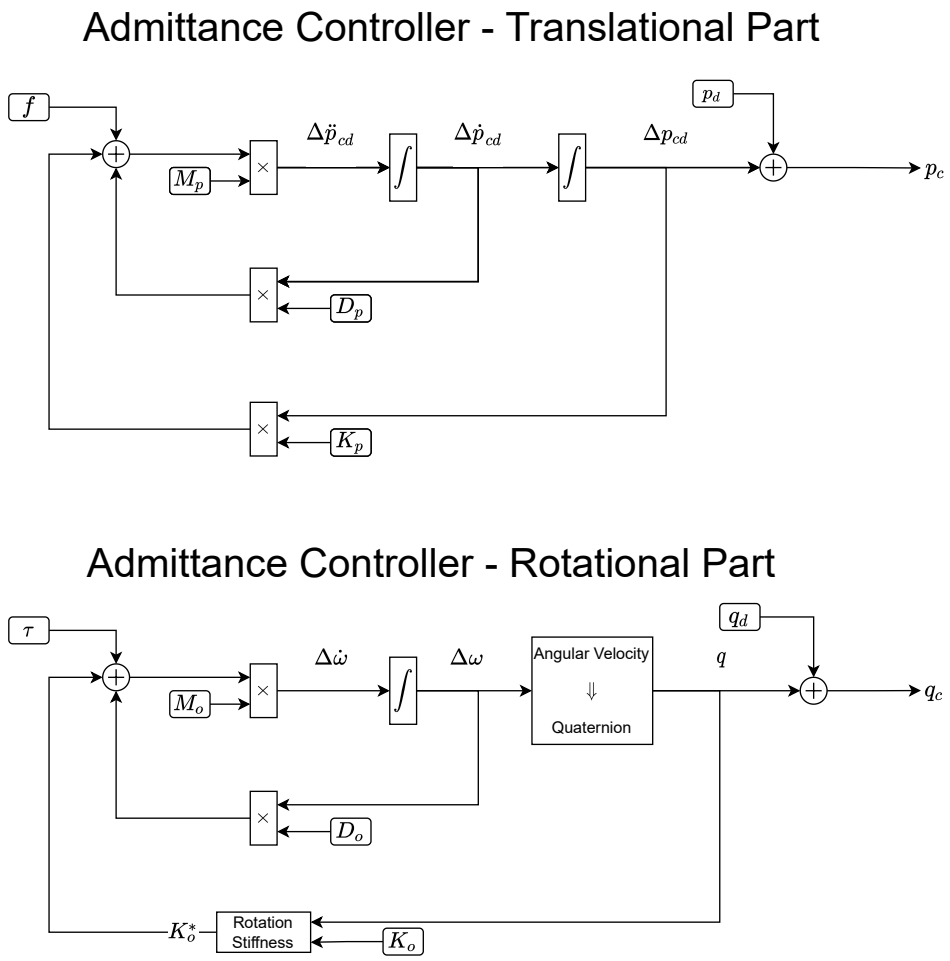


Figure 4.2: Block diagram of implementation of admittance controller.

$$\begin{aligned} \omega_{nx} &= \sqrt{\frac{k_{p_x}}{m_{d_x}}} \\ \zeta_x &= \frac{k_{d_x}}{2\sqrt{m_{d_x} k_{p_x}}} \end{aligned} \tag{4.5}$$

where m_{d_x}, k_{d_x}, and k_{p_x} is the x-element of the mass, damping, and spring matrices respectively. So it is possible to have a critical stable axis if two of the values of m_{d_x}, k_{d_x}, and k_{p_x} is know, then the last can be calculated by solving for it.

This is a fine method if the mass, damping and spring matrices are static, but this is not the case for this project since the goal is to estimate these parameters. Therefore, it is important that if the parameters are all estimated

that they have some form of relation to each other and it would properly be best that one of them dominates and the other follows it as in the variable damping method (see section 3.3).

Test

The implemented of the admittance controller has been tested with constant desired position (p_d), desired orientation (p_o), mass (M_p, M_o), damping (D_p, D_o), and stiffness (K_p, K_o). See equation (4.6) for parameters used. Then a force and torque is applied after 1 seconds with a step response and removed again at 2 second. The force and torque applied can be seen in figure 4.3. The compliant position and rotation in euler angles can be seen in figure 4.4. The joint configurations of the simulated UR10 can be seen in figure 4.5.

$$
\begin{aligned}
p_d &= \begin{bmatrix} -0.25 & 0 & 0.25 \end{bmatrix} & p_o &= \begin{bmatrix} \frac{\pi}{2} & 0 & 0 \end{bmatrix} \\
M_p &= \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} & M_o &= \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix} \\
D_p &= \begin{bmatrix} 13.4164 & 0 & 0 \\ 0 & 13.4164 & 0 \\ 0 & 0 & 13.4164 \end{bmatrix} & D_o &= \begin{bmatrix} 6.4807 & 0 & 0 \\ 0 & 6.4807 & 0 \\ 0 & 0 & 6.4807 \end{bmatrix} \\
K_p &= \begin{bmatrix} 15 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 15 \end{bmatrix} & K_o &= \begin{bmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{bmatrix}
\end{aligned}
\tag{4.6}
$$

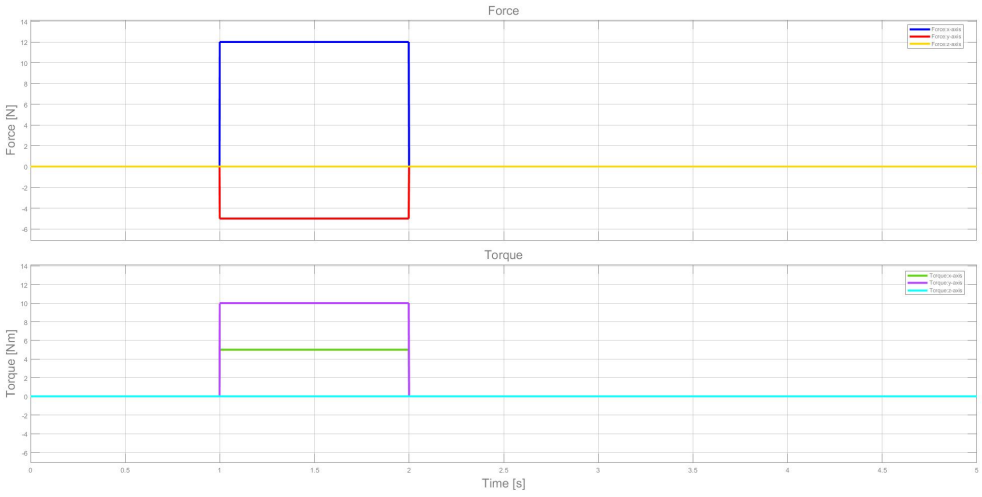


Figure 4.3: Force and torque applied at 1 second and removed at 2 second for the test of the admittance controller with constant desired pose, mass, damping, and stiffness.

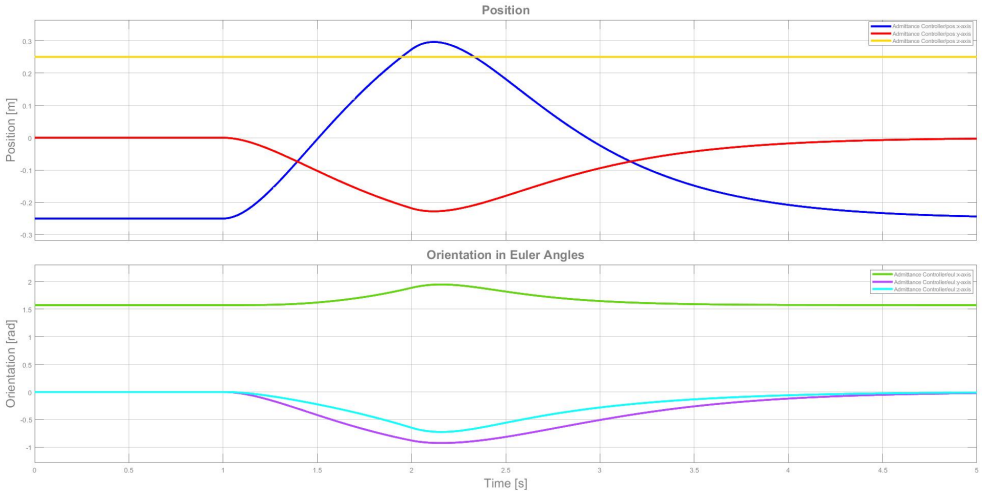


Figure 4.4: The compliant position and orientation for the test of the admittance controller with constant desired pose, mass, damping, and stiffness.

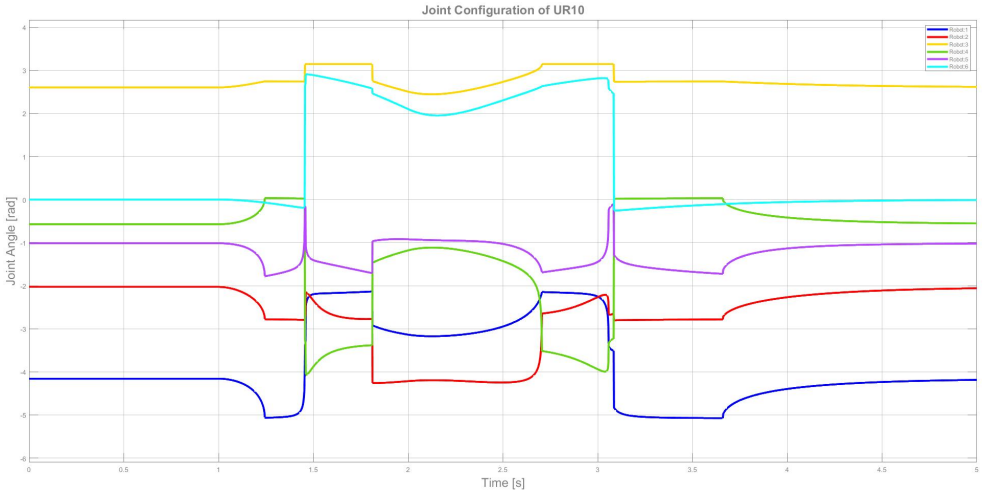


Figure 4.5: The joint configurations for the test of the admittance controller with constant desired pose, mass, damping, and stiffness.

Further test with the admittance controller having non-constant variables will be done in chapter 6.

Discussion and Conclusion

From the test done on the admittance controller (figure 4.4) it can be seen that it works as intended and the compliant frame moves accordingly to the applied force and torques. After the force is no longer applied the compliant frame moves back to its desired pose without any oscillation since the parameters

(mass, spring, and damping) was chosen so the controller would be critically damped.

From figure 4.5 it can be seen that when there is a change in the desired pose for the simulated UR10 the joint configuration can jump quite drastically. It would properly be best to introduce some dynamics to reduce this. This will be done in section 4.2.

4.2 PD Controller

To avoid the simulation environment having almost no dynamics a PD controller with gravity compensation has been added so dynamics will be incorporated. A block diagram of PD controller with gravity compensation controlled in joint space can be seen in figure 4.6.

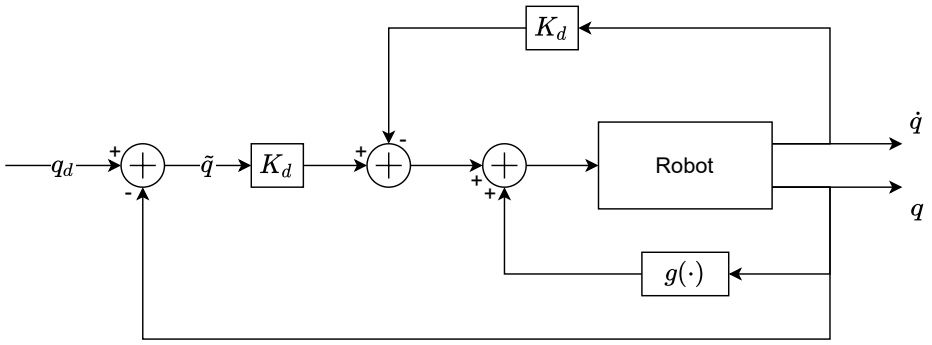


Figure 4.6: Block diagram of PD controller with gravity compensation.

$g(\cdot)$ is the gravity compensation. q_d is the desired joint configurations and $\tilde{q}$ is the error between the actual joint configuration (q) and the desired joint configuration. $K_p \in \mathbb{R}^{n\times n}$ and $K_d \in \mathbb{R}^{n\times n}$ is the proportional and derivative gain respectively.

The hard part of a PD controller is choosing the correct gains for K_p and K_d so that the controller is stable. It is also important to have a good gravity compensation that is not imperfect. But according to B. Siciliano et al. [32] stability is ensured as long as K_p and K_d is positive definite matrices. It is also an requirement that the gravity term $g(\cdot)$ is computed online and is not imperfect. If these requirements are assured then stability is as well. Therefore, is it chosen to have the gravity term computed online. Normally when tuning a PD controller a method like Ziegler-Nichols can be used, but since there is a gravity term incorporated it is not as simple. But then instead when choosing the values of K_p and K_d it should be done by how fast a system response is desired. But since it is almost always wanted to have the fastest system response possible, so the robot converges the fastest to its desired position, the gains is chosen as high as possible while the response looks acceptable. So

it is fine-tuned by trail and error. This is also justifiable to do since the desired configuration to the PD controller is going to be smooth paths and not just a step response.

Implementation and results

The PD controller with online gravity compensation has been implemented in Simulink. The online gravity compensation was made possible with the Gravity Torque block[1]. Finding the values for the PD controller was done by first using the values from A. De Luca et al. [33] and then modifying the values slowly and comparing the joint configurations and end-effector pose until a satisfying behavior was attained. This lead K_p to be $K_p = diag(\begin{bmatrix}400 & 400 & 150 & 100 & 80 & 40\end{bmatrix})$ and K_d to be $K_d = diag(\begin{bmatrix}23.1250 & 23.1250 & 20.1250 & 7.6250 & 4.0625 & 0.5000\end{bmatrix})$. The same test as the one used for the admittance controller (see section 4.1) is the one used here, expect no torque is applied. A graph of the desired joint configuration coming from the inverse kinematic solver (see chapter 5 for explaination of the inverse kinematic solver is shown in figure 4.7 and the joint configurations from the PD controller with online gravity compensation can be seen in figure 4.8. The end-effectors position and orientation can be seen in figure 4.9.

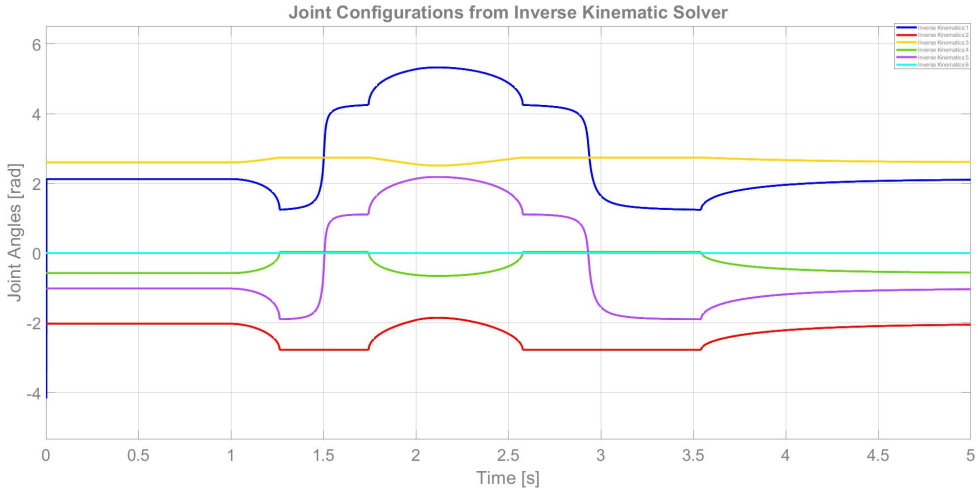


Figure 4.7: The joint configurations of the inverse kinematic solver for the test of the PD controller with online gravity compensation.

[1]Gravity Torque - Robotics System Toolbox - `https://se.mathworks.com/help/robotics/ref/gravitytorque.html`

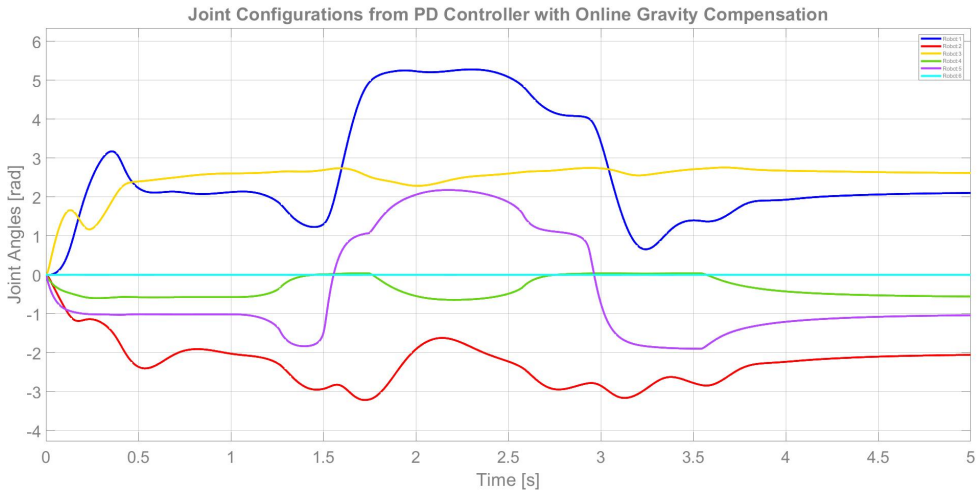


Figure 4.8: The joint configurations of the PD controller solver for the test of the PD controller with online gravity compensation.

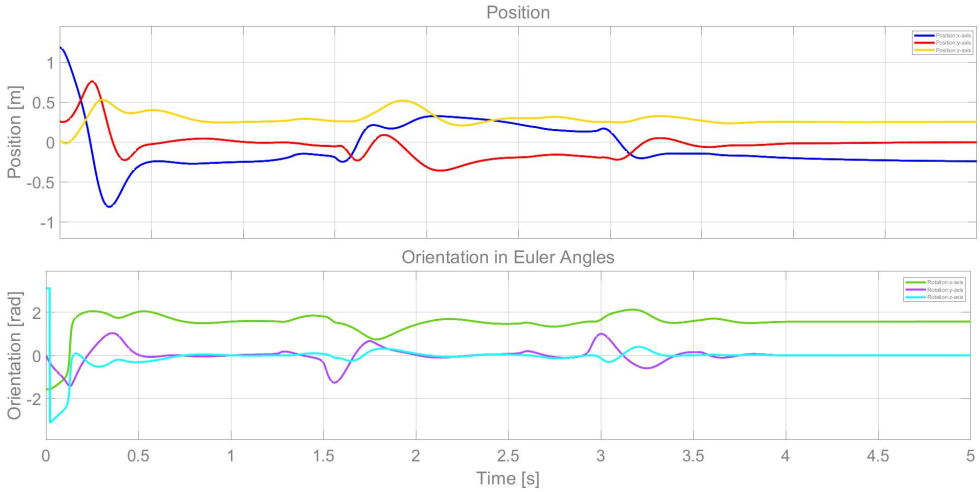


Figure 4.9: The end-effectors position and orientation for the test of the PD controller with online gravity compensation.

Discussion

From figure 4.7 and figure 4.8 it can be seen that the PD controller works fairly good at mimicking the desired joint configuration from the IK solver. It is better at doing it for the last three joints rather than the first three. So the first three could probably be fine-tuned more precisely if better behavior was desired, but was not done due to time constraints. The problem with the first three joints being less precise than the last three is that the first three have a bigger influence on the error of the end-effector if they themselves have errors.

This could also explain why the plot in figure 4.9 does not mimic the plot in figure 4.4 as well. It can be seen that the PD controller still manages to get to the same waypoint at some of the timestamps as the compliant frame. Such timestamp is around the ~ 2.2 second mark for both the x- and y-axis for the positional part.

Conclusion

It can be concluded that the PD controller with online gravity compensation manages to introduce dynamics into the simulation and it works fairly well, but still needs more fine-tuning to work properly.

5 System

Here the full system for the simulation with all the subparts combined together will be shown. A visualization over the whole system can be seen in figure 5.1.

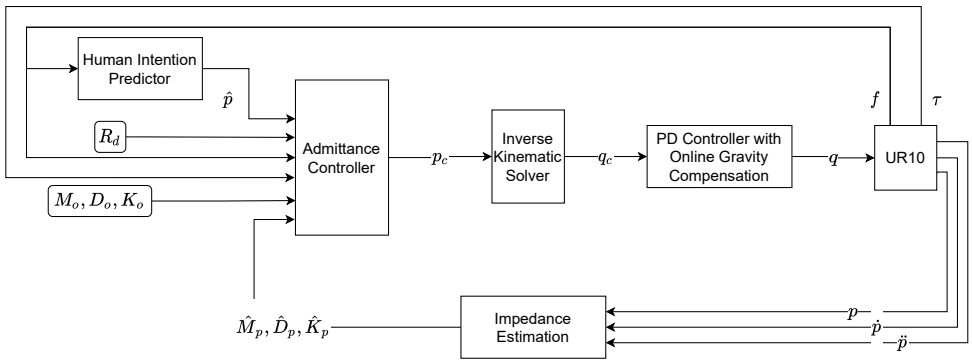


Figure 5.1: Illustration of the full system. p, $\dot{p}$, and $\ddot{p}$ is the position, velocity, and acceleration of the robots end-effector respectively. f and τ is the force and torque from the robot. $\hat{M}_p$, $\hat{D}_p$, and $\hat{K}_p$ is the estimated mass, damping, and stiffness values from the impedance estimation for the positional part of the admittance controller. M_o, D_o, and K_o is the constant mass, damping, and stiffness for the rotational part of the admittance controller. $\hat{p}$ is the estimated position of the human intention predictor. R_d is the desired rotation. p_c is the complaint pose of the admittance controller. q_c is the joint configuration to p_c. q is the joint configuration from the PD controller with online gravity compensation.

The human intention predictor block can be either of the methods from section 3.2. So it can be the ARX Model predictor or the NARX Network predictor. The same concept is for the impedance Estimation block. This can either be the IEFD estimation or the variable damping method.
Since the project was only done in simulation there is no illustration of what the system would look like if done on a physical robot. However, the only difference would be that the three blocks Inverse Kinematic Solver, PD Controller with Online Gravity Compensation, and UR10 would instead be one block with the physical robot.

The simulation used for the whole project was done in MATLAB and Simulink with a sampling frequency of 500Hz. For each method used for the system, the explanation of its implementation has been elaborated in their respective sections. However, the simulated UR10 robot and the inverse kinematic solver have not been explained. Simulating the UR10 robot was done by

loading the pre-existing UR10 model (called *universalUR10*) from the MATLAB library in the form of a rigidBodyTree[1]. Then the Simulink block VR RigidBodyTree[2] was used to simulate the loaded UR10 model. The inverse kinematic solver was introduced since it is only possible to control the simulated UR10 model with joint configuration. So it was necessary to convert the compliance frame given by the admittance controller to joint configuration. The Simulink block Inverse Kinematics[3] block was used for this. For the full implementation see Appendix A.1.

[1]rigidBodyTree - Robotics System Toolbox - `https://se.mathworks.com/help/robotics/ref/rigidbodytree.html`

[2]VR RigidBodyTree - Simulink 3D Animation - `https://se.mathworks.com/help/sl3d/vrrigidbodytree.html`

[3]Inverse Kinematics - Robotics System Toolbox - `https://se.mathworks.com/help/robotics/ref/inversekinematics.html`

6 Test and Results

Here the tests of the full system will be shown. The test that is conducted is done on the minimum jerk trajectory using the ARX Model prediction with a 50 prediction length. The variable damping method is used for human impedance. Here the prediction of the ARX Model is fed to the admittance controller together with the mass, damping, and stiffness from the variable damping method, see chapter 5 for more detailed explanation. The prediction of the ARX Model is identical to figure A.7 and the damping and stiffness is same as the one in figure A.31. The complaint position and orientation from the admittance controller is shown in figure 6.1, the joint configuration from the inverse kinematic solver is shown in figure 6.2, the output of the PD controller with online gravity compensation is shown in figure 6.3, and the position and orientation of the end-effector of the simulated UR10 is shown in figure 6.4.

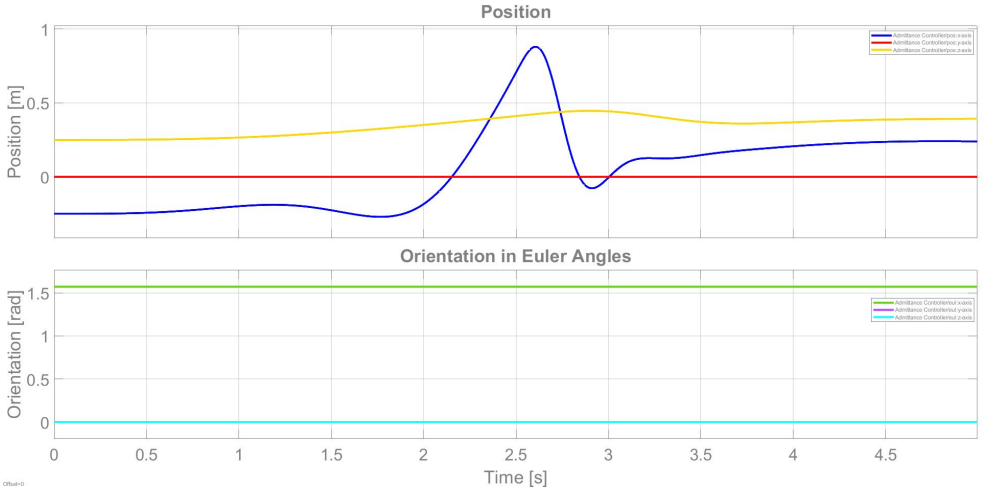


Figure 6.1: Complaint position and orientation of tests on the full system using the minimum jerk trajectory 1.

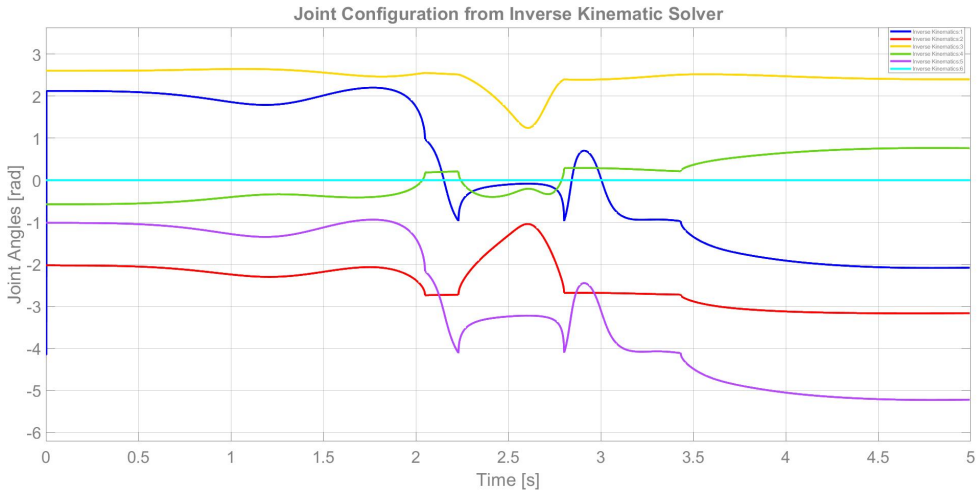


Figure 6.2: Inverse kinematic solver of tests on the full system using the minimum jerk trajectory 1.

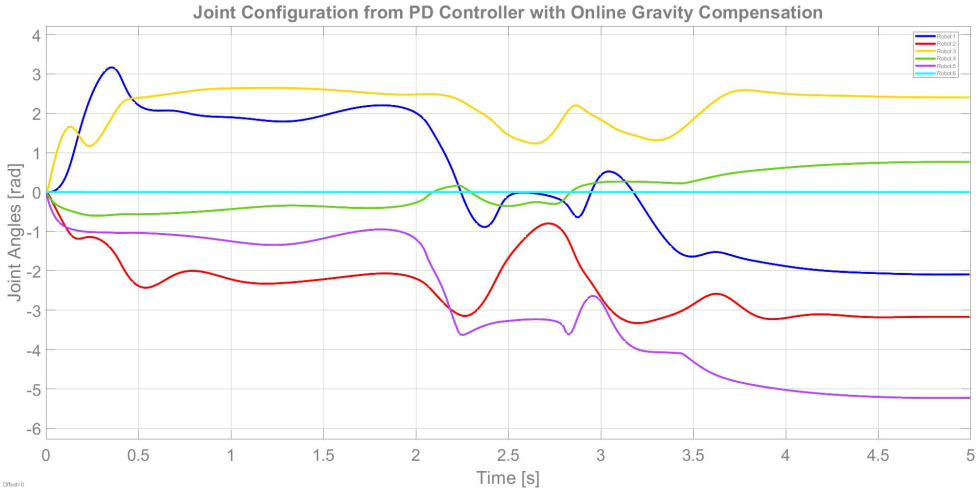


Figure 6.3: PD Controller with online gravity compensation of tests on the full system using the minimum jerk trajectory 1.

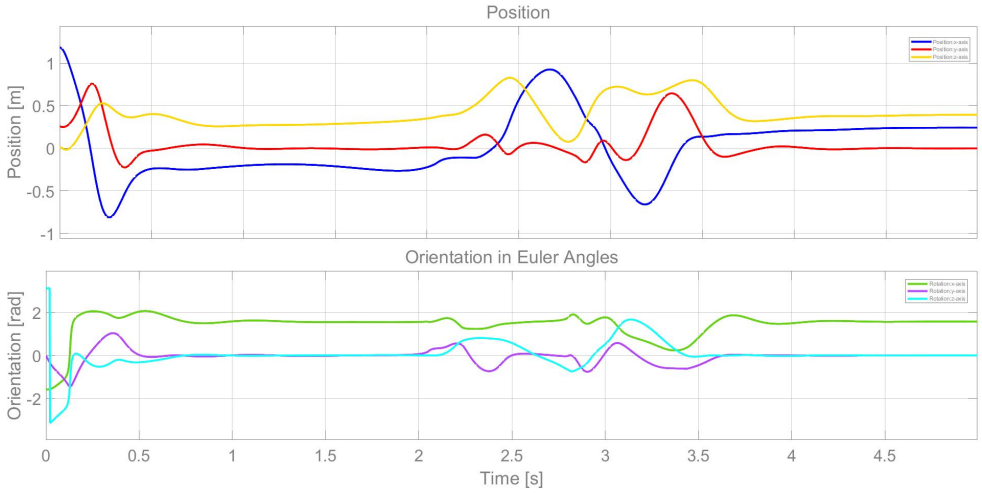


Figure 6.4: End-effector of simulated UR10 of tests on the full system using the minimum jerk trajectory 1.

7 Discussion

The test completed on the full system done in chapter 6 was only completed on the minimum jerk trajectories since the performance of human motion prediction and the human impedance estimation did not perform as intended for the physical trajectories. From figure 6.2 and figure 6.3 the performance is the same as then the PD controller was tested separately. This is within expectation. From figure 6.1 and figure 6.4 it can be seen that the end-effector of the robot follows that of the compliant frame fairly well. However, it can be seen that it takes time for the end-effector to stabilize to the correct position. This is due to the PD controller not converging fast enough. But still, it is not a perfect match.

It is obvious to see that there is room for improvement in different parts of the project. The ARX Model is not great at predicting the physical trajectories that might be caused by the noise on the force readings. This also needs improvement and could potentially be done if the force readings were filtered. However, this would delay the force readings and cause a larger delay for the prediction. The NARX Network needs to be trained to work properly on all minimum jerk trajectories and then work on the physical ones as well. Here a change to the data set is a must, as mentioned in the discussion of section 3.2. The IEFD method for estimating human impedance parameters was not successful, only the estimated parameters from table 3.7 is similar to the ones present in authors work [30] but is still not acceptable. The cause for the method not being successful could be that not the right MSD model is used, the right validation method, or that the method is not applicable to this use case. Further investigation is needed to find the exact cause. The variable damping method works as intended and produces reliable results.

Looking at the sub-goals presented in the problem statement section 1.1 it can be seen that most of the goals have been met. Estimating the motion intention of the human operator online has been successfully completed with the use of the ARX Model and partly the NARX Network. Estimating the impedance of the human operator has not been completely successful since the IEFD didn't work as intended but the variable damping method did, but it is technically not an estimation method. Ensuring the stability of the full system has been done since the stability of the PD controller with online gravity compensation and the admittance controller stability has been ensured. However, noise could potentially make some of the methods like the variable damping and the ARX model oscillate quite a lot. The admittance controller has also been successfully been implemented to ease the HRI. Combining all subparts of the project has been done into a full system with the method that

succeded, but unfortunately not the ones that did not. The full system has been implemented in MATLAB and Simulink but has not been implemented onto a physical robot because of time constraints.

For the full system described in chapter 5 there is not an indication of the object that is carried between the human and robot. The reason for that is that in the simulation adding an object to carry on the robot would be quite simple since it would be possible to just add a body to the rigidBodyTree that the simulated robot was expressed with. However, since it is assumed that the object is rigidly attached to the end-effector of the robot it is equivalent to just changing the mass and center of mass of the end-effector instead. That would depend on the object that is carried and since that is not specified such an object it is therefore assumed that the object is a mass point of which the tiny mass and volume can be ignored. Therefore, the interaction force between the human and object is the same as the interaction force between the object and the robot. X. Yu et al. [13] uses a similar approach.

7.1 Future Work

For future work that could be done and how not already been mentioned, there are a few things that would be suitable to add to the project. The reason for these things not being added is due to time constraints on the project. The first future improvement is increasing the stiffness when the robot's end-effector is nearing it end of its reachability space. Figure 7.1 shows a illustration of the concept.

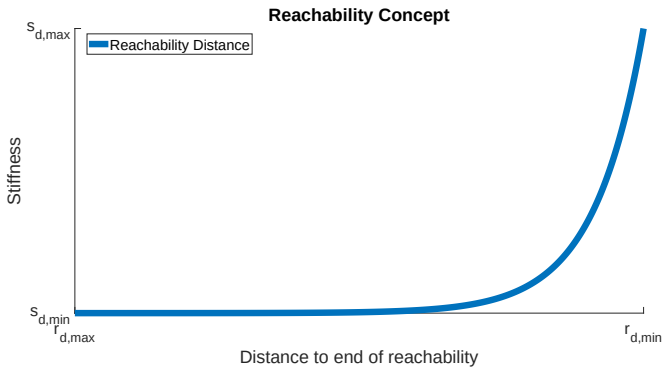


Figure 7.1: Reachability Concept. $r_{d,max}$ and $r_{d,min}$ is the maximum and minimum reachability distance respectively. The $r_{d,min}$ would likely be zero. $s_{d,min}$ and $s_{d,max}$ is the minimum and maximum stiffness respectively.

The idea is that when the end-effector is close to the edge of its reachability space the stiffness increase drastically so it is not possible for the human

operator to move the end-effector to the edge of reachability. This prevents it from getting into a singularity at the edge and also prevents the human from damaging the robot if they are trying to move it outside its reachability space. The idea originated from A. Dahlin et al. [34] where they use a similar idea for their potential term on their dynamic movement primitives. It was then thought to be used as the concept explain here. This could have been put into practice by using a simple exponential function and calculating the reachability distance for the robot used. This was however not done, since it was chosen to focus on the more essential part of the projects.

The second future improvement that could be suitable for the project is to add singularity avoidance [35, 36]. This would ensure that the robot would never end up in a singularity and the robot would remain stable. This could be achieved by detecting when coming close to a singularity and then providing a form of feedback back to the human operator. This can be done in different ways like using haptic feedback to alarm the human or adding a virtual force on the end-effector of the robot so it pushes the end-effector away from the singularity if approaching it.

8 Conclusion

The method minimum jerk model for generating human-like motion was successfully implemented and produced positional trajectory with velocity and acceleration profile, where it was possible to generate a force profile using Newtons 2. law and the acceleration profile. Physical trajectories was also recorded on a physical UR10e robot.

Three methods for human motion prediction were presented and two were successfully implemented. The two methods were using the ARX Model and NARX Network structure. The ARX Model worked great on the minimum jerk trajectories with a maximum of -42.36cm in error on the worst trajectory, but this was an extreme outlier compared to the rest. Different prediction length was experimented with and a prediction length around 50 was found to be the best and most reliable. The model did not perform as well on the physical trajectories and was only able to predict some of the trajectories before drifting too far. So more work is needed there. The NARX Network was able to predict fairly well on some of the minimum jerk trajectories and not at all on others. The NARX Network did not work on the physical trajectories and it was concluded that the cause properly was overfitting and that there needed modifications to the training and the data set used for training.

Two methods was used for estimating human impedance. The IEFD method was not successful in estimating the parameters correctly according to the validation method on either of the minimum jerk or physical trajectories. However, the parameters for the physical trajectories look moderately similar to the experiments of the authors of the method. The variable damping method worked as intended and was successful. The stiffness was calculated accordingly to the damping.

An admittance controller was applied to allow for HRI. The admittance controller was the only part of the project that was implemented with a rotational component, whereas the other parts, such as the generation of human motion, motion prediction, and impedance estimation were only done positionally. The admittance controller was successfully implemented and worked as intended.

A PD controller with online gravity compensation was applied to introduce dynamics into the simulation. This was a success, but it is concluded that the tuning of its gains needed a bit more work.

All the successfully separated parts were put together as one system and tested. Here the results were satisfactory and the system performed as intended. The system was also stable as this was ensured when implementing the methods.

9 Acknowledgement

I want to thank my two supervisors Christoffer Sloth and Inigo Iturrate San Juan for guiding me through my master thesis, as well as Emil Lykke Diget for helping with implementation of certain parts of the thesis. I also want to thank the University of Southern Denmark for letting me write my masters thesis at their establishment. Furthermore, I want to thank my colleage Jacob Falgren Christensen for helping with any questions and problems I encountered throughout the project. Last but not least, I wanna thank me.

Bibliography

[1] F. J. Abu-Dakka and M. Saveriano, "Variable impedance control and learning—a review," *Frontiers in Robotics and AI*, vol. 7, 2020.

[2] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American Control Conference*, pp. 304–313, 1984.

[3] R. Ikeura and H. Inooka, "Variable impedance control of a robot for cooperation with a human," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3097–3102 vol.3, 1995.

[4] T. Lasky and T. Hsia, "On force-tracking impedance control of robot manipulators," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pp. 274–280 vol.1, 1991.

[5] A. Edsinger and C. C. Kemp, "Human-robot interaction for cooperative manipulation: Handing objects to one another," in *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 1167–1172, 2007.

[6] L. Peternel, T. Petrič, E. Oztop, and J. Babič, "Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach," *Autonomous robots*, vol. 36, no. 1, pp. 123–136, 2014.

[7] C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rosel, J. Gast, A. Bannat, and F. Wallhoff, "Joint-action for humans and industrial robots for assembly tasks," in *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 130–135, IEEE, 2008.

[8] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative human-humanoid carrying using vision and haptic sensing," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 607–612, 2014.

[9] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 399–404, 2009.

[10] X. Yu, W. He, Q. Li, Y. Li, and B. Li, "Human-robot co-carrying using visual and force sensing," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 9, pp. 8657–8666, 2021.

[11] X. Yu, S. Zhang, Y. Liu, B. Li, Y. Ma, and G. Min, "Co-carrying an object by robot in cooperation with humans using visual and force sensing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2207, p. 20200373, 2021.

[12] J. DelPreto and D. Rus, "Sharing the load: Human-robot team lifting using muscle activity," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7906–7912, 2019.

[13] X. Yu, W. He, Y. Li, C. Xue, J. Li, J. Zou, and C. Yang, "Bayesian estimation of human impedance and motion intention for human–robot collaboration," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1822–1834, 2021.

[14] F. Müller, J. Janetzky, U. Behrnd, J. Jäkel, and U. Thomas, "User force-dependent variable impedance control in human-robot interaction," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1328–1335, 2018.

[15] K. Li, R. Chen, T. Nuchkrua, and S. Boonto, "Dual loop compliant control based on human prediction for physical human-robot interaction," in *2019 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 459–464, 2019.

[16] T. Bitz, F. Zahedi, and H. Lee, "Variable damping control of a robotic arm to improve trade-off between agility and stability and reduce user effort," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11259–11265, 2020.

[17] X. Yu, Y. Li, S. Zhang, C. Xue, and Y. Wang, "Estimation of human impedance and motion intention for constrained human–robot interaction," *Neurocomputing*, vol. 390, pp. 268–279, 2020.

[18] F. Campos and J. Calado, "Approaches to human arm movement control—a review," *Annual reviews in control*, vol. 33, no. 1, pp. 69–77, 2009.

[19] A.-N. Sharkawy1, "Chapter 5 minimum jerk trajectory generation for straight and curved movements: Mathematical analysis,"

[20] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[21] I. S. MacKenzie, "Fitts' law as a performance model in human-computer interaction," *Unpublished Doctoral Dissertation, University of Toronto.*

[22] UR, "e-series cobots | collaborative robots by universal robots." https://www.universal-robots.com/e-series/.

[23] H. A. (2022), "A general multi-segment minimum jerk trajectory toolbox (v1)." https://www.mathworks.com/matlabcentral/fileexchange/75384-a-general-multi-segment-minimum-jerk-trajectory-toolbox-v1.

[24] MATLAB, "Estimate parameters of ARX, ARIX, AR, or ARI model - MATLAB arx - MathWorks nordic." https://se.mathworks.com/help/ident/ref/arx.html.

[25] MATLAB, "Design time series NARX feedback neural networks - MATLAB & simulink - MathWorks nordic." https://se.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html.

[26] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[27] K. Li, T. Nuchkrua, H. Zhao, Y. Yuan, and S. Boonto, "Learning-based adaptive robust control of manipulated pneumatic artificial muscle driven by h2-based metal hydride," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1284–1289, 2018.

[28] Q. Liu, W. Chen, H. Hu, Q. Zhu, and Z. Xie, "An optimal narx neural network identification model for a magnetorheological damper with force-distortion behavior," *Frontiers in Materials*, vol. 7, 2020.

[29] MATLAB, "Selecting a model structure in the system identification process." https://www.ni.com/da-dk/support/documentation/supplemental/06/selecting-a-model-structure-in-the-system-identification-process.html.

[30] M. J. Fu and M. C. Cavusoglu, "Human-arm-and-hand-dynamic model with variability analyses for a stylus-based haptic interface," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 6, pp. 1633–1644, 2012.

[31] K. Kosuge, Y. Fujisawa, and T. Fukuda, "Control of mechanical system with man-machine interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 87–92, 1992.

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010.

[33] A. De Luca, B. Siciliano, and L. Zollo, "Pd control with on-line gravity compensation for robots with elastic joints: Theory and experiments," *automatica*, vol. 41, no. 10, pp. 1809–1819, 2005.

[34] A. Dahlin and Y. Karayiannidis, "Adaptive trajectory generation under velocity constraints using dynamical movement primitives," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 438–443, 2020.

[35] M. Weyrer, M. Brandstötter, and M. Husty, "Singularity avoidance control of a non-holonomic mobile manipulator for intuitive hand guidance," *Robotics*, vol. 8, no. 1, p. 14, 2019.

[36] T. DeWolf, "Robot control part 6: Handling singularities." https://studywolf.wordpress.com/2013/10/10/robot-control-part-6-handling-singularities/.

A Appendix

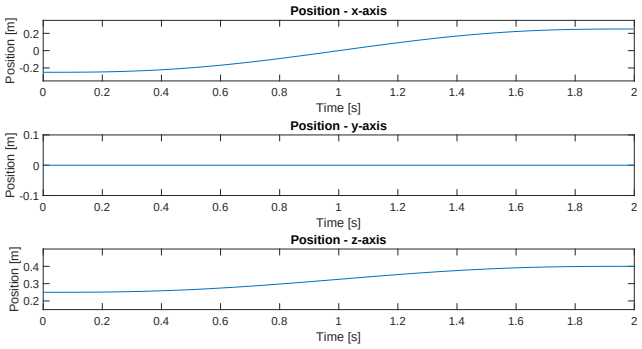
A.1 Implementation

- GitHub Profile : `https://github.com/JensOHI`
- GitHub Project Name: Master_Thesis
- GitHub Project Link: `https://github.com/JensOHI/Master_Thesis`

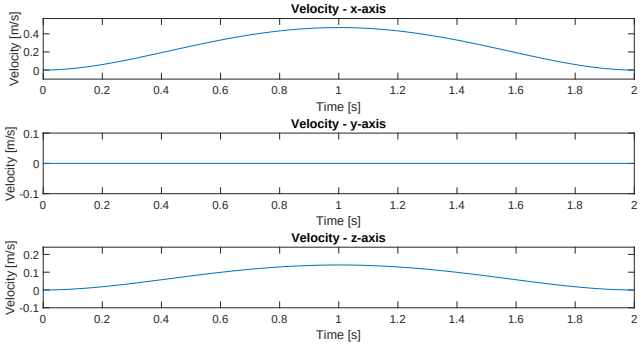
If the GitHub repository is unavailable please contact Jens Otto Hee Iversen at jens@pilbro.dk.

A.2 Generate Human Motion Additional Graphs

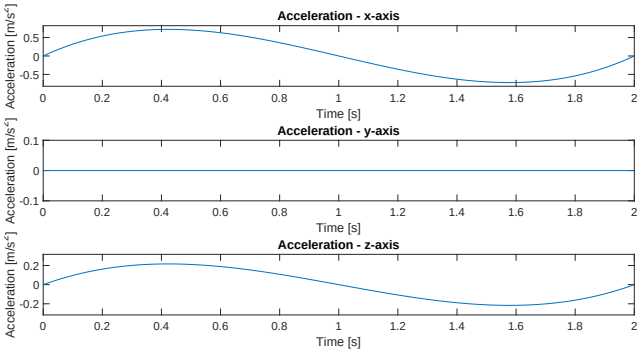
Minimum Jerk Model Additional Graphs



(a) Position

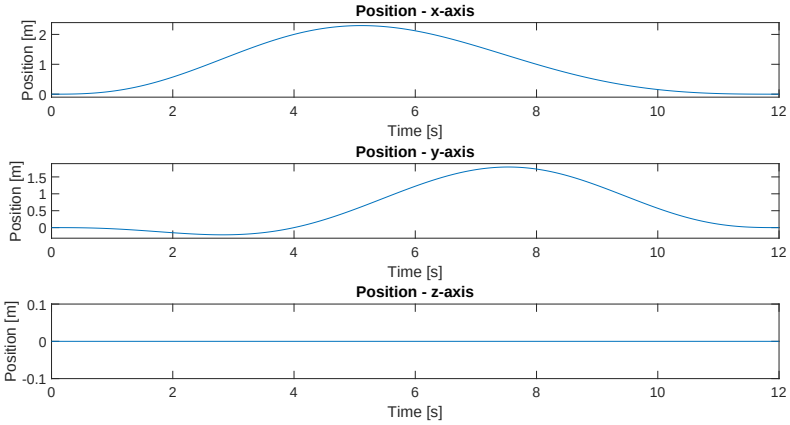


(b) Velocity

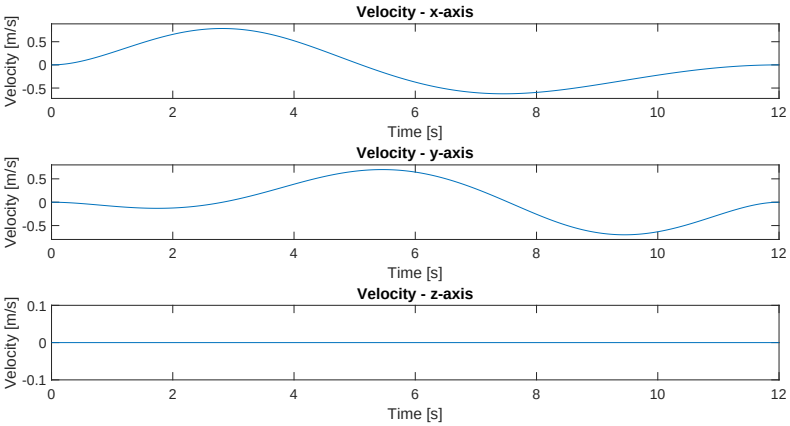


(c) Acceleration

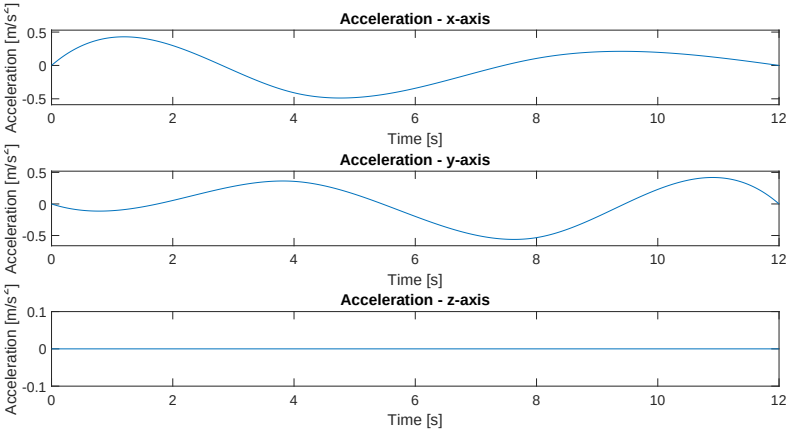
Figure A.1: The second trajectory generated using the minimum jerk model. Will be referred to as minimum jerk trajectory 2.



(a) Position

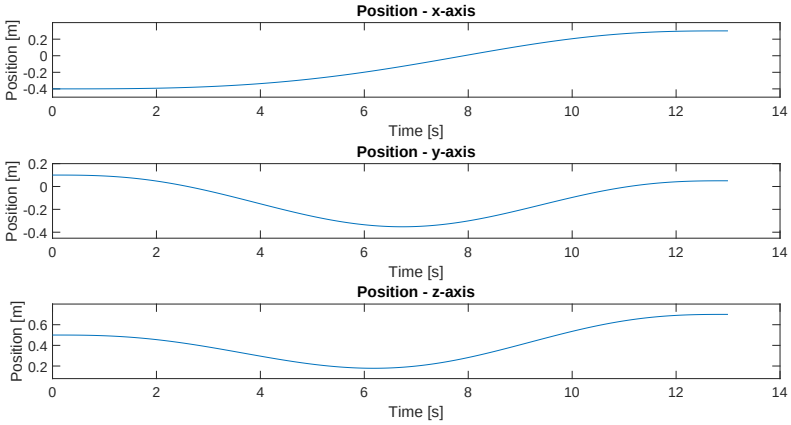


(b) Velocity

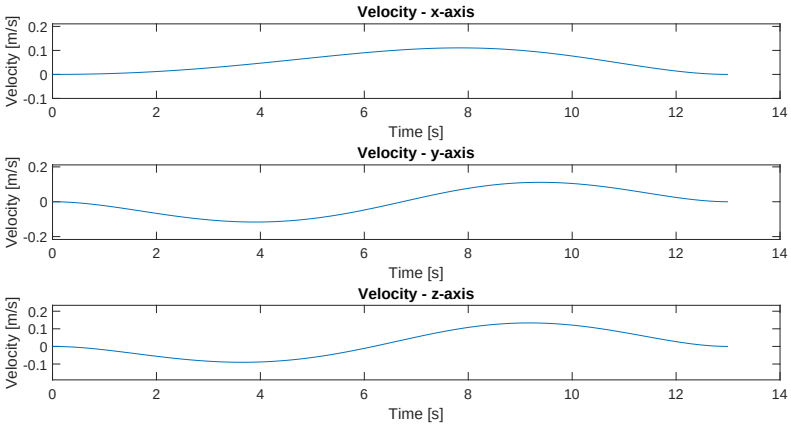


(c) Acceleration

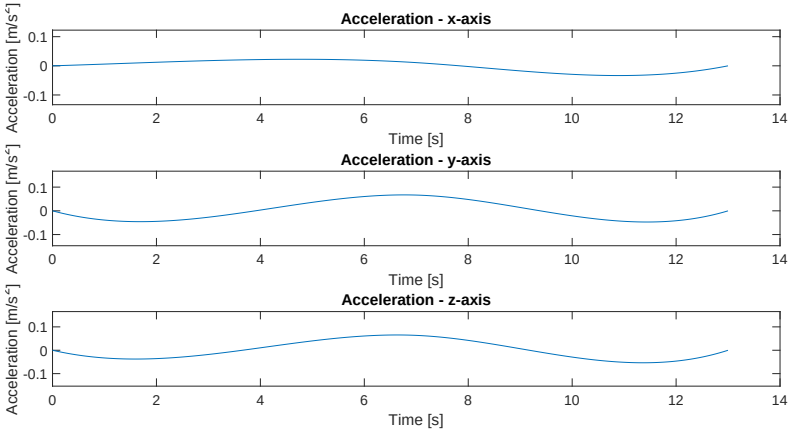
Figure A.2: The third trajectory generated using the minimum jerk model. Will be referred to as minimum jerk trajectory 3.



(a) Position



(b) Velocity



(c) Acceleration

Figure A.3: The fourth trajectory generated using the minimum jerk model. Will be referred to as minimum jerk trajectory 4.

Recording Trajectory on Real Robot Additional Graphs

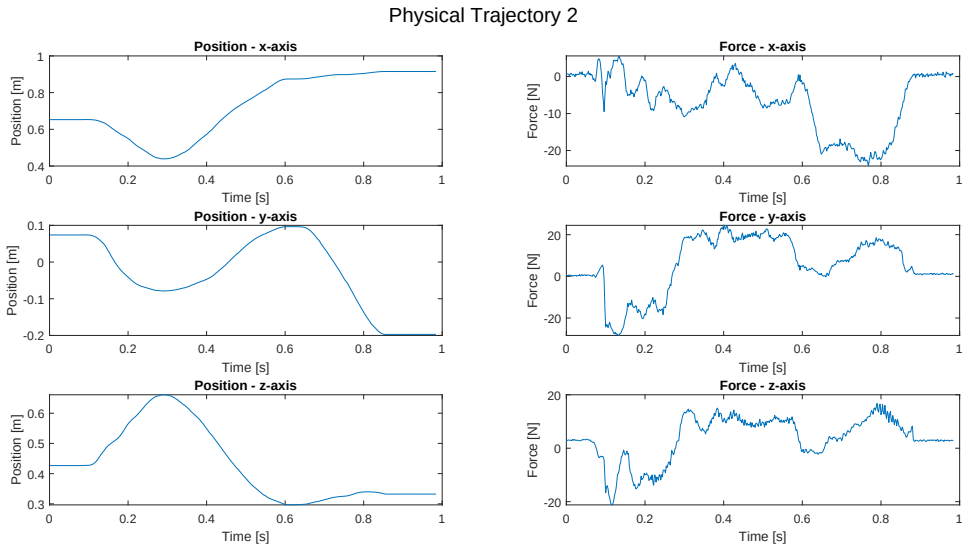


Figure A.4: The second trajectory recorded on a physical UR10e. Will be called physical trajectory 2.

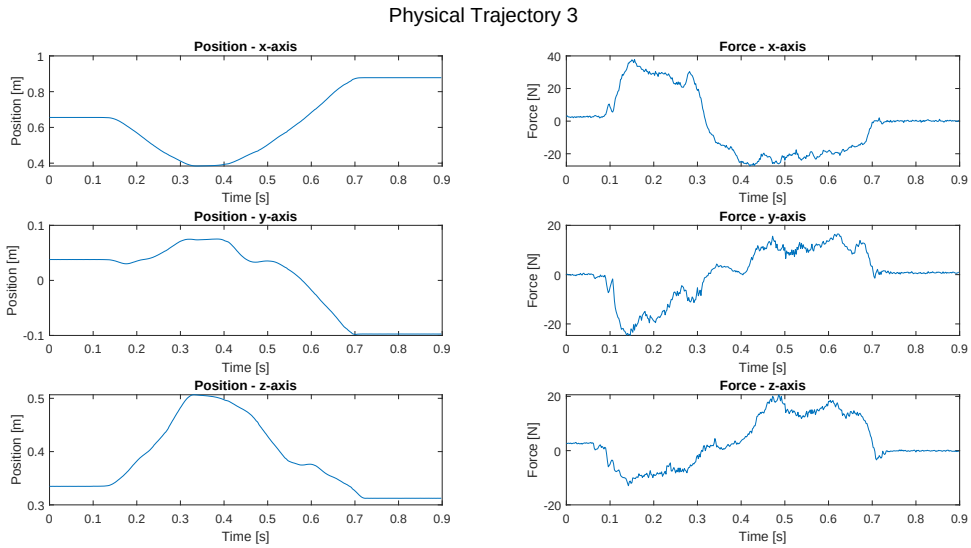


Figure A.5: The third trajectory recorded on a physical UR10e. Will be called physical trajectory 3.

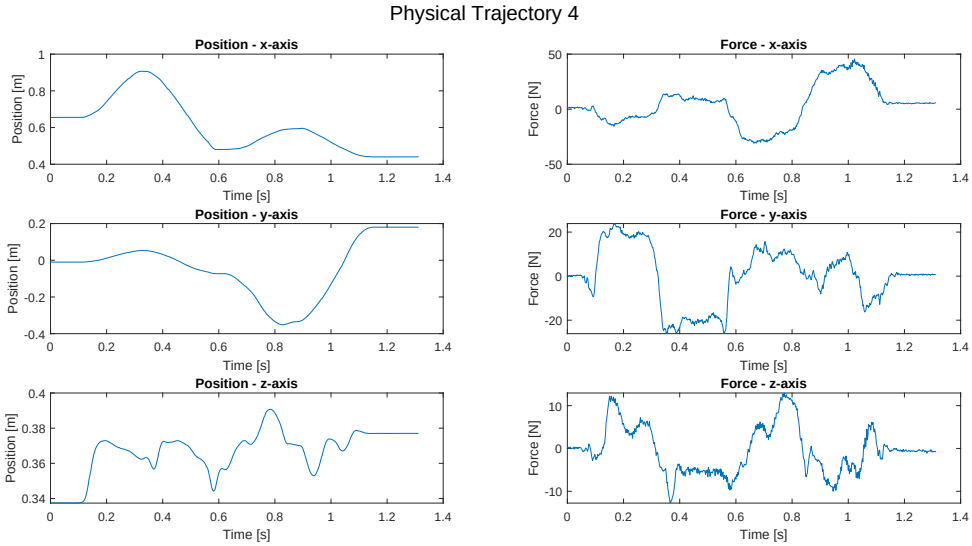
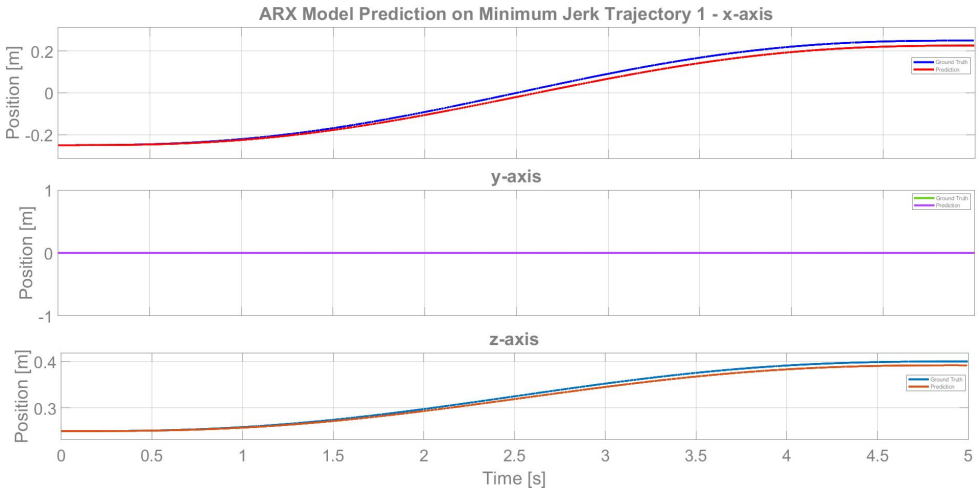


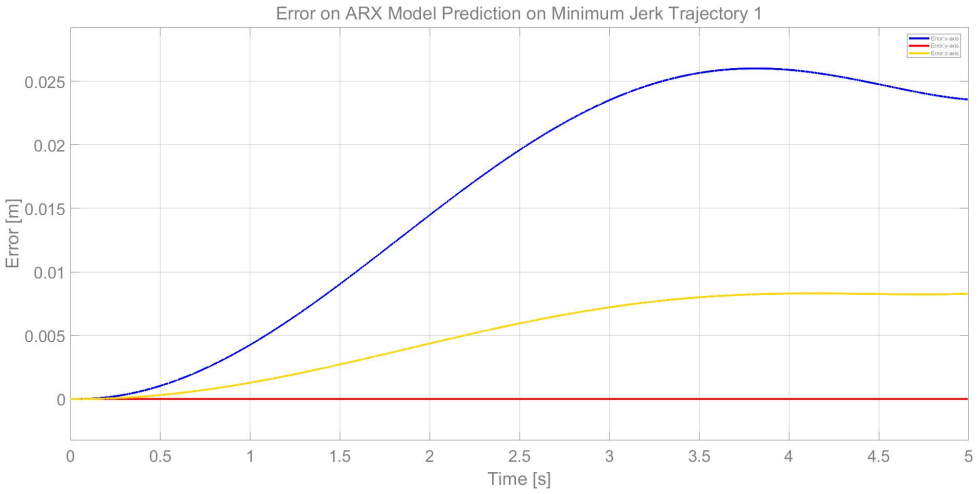
Figure A.6: The fourth trajectory recorded on a physical UR10e. Will be called physical trajectory 4.

A.3 Motion Prediction Additional Tests

ARX Model

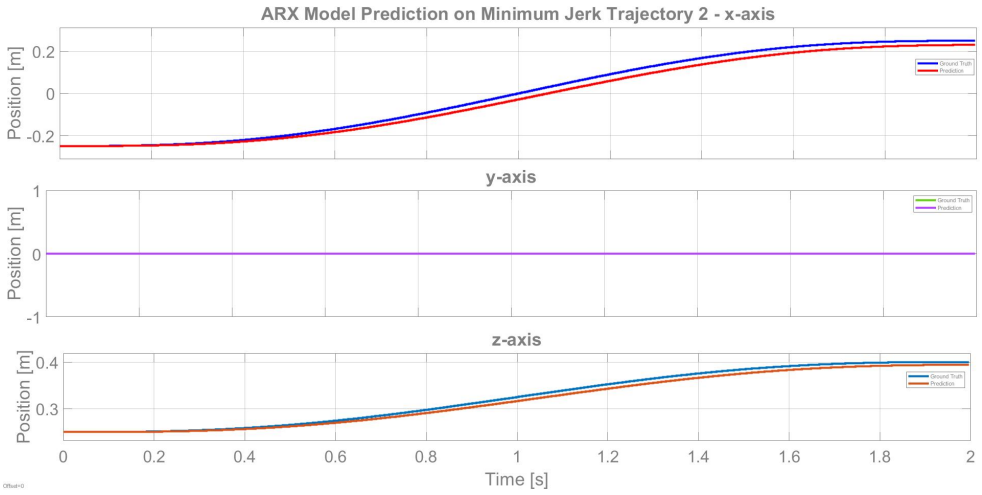


(a) Prediction and Ground truth

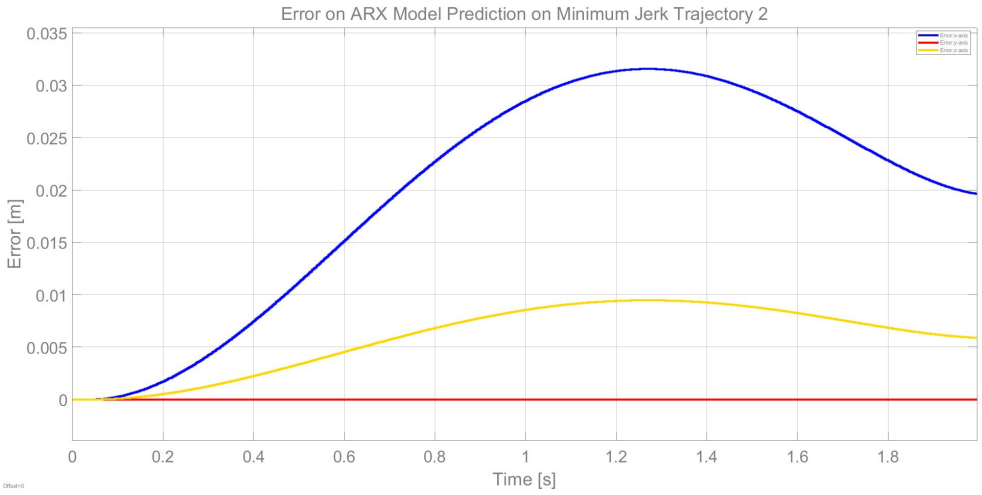


(b) Error between prediction and ground truth

Figure A.7: ARX Model prediction of minimum jerk trajectory 1.

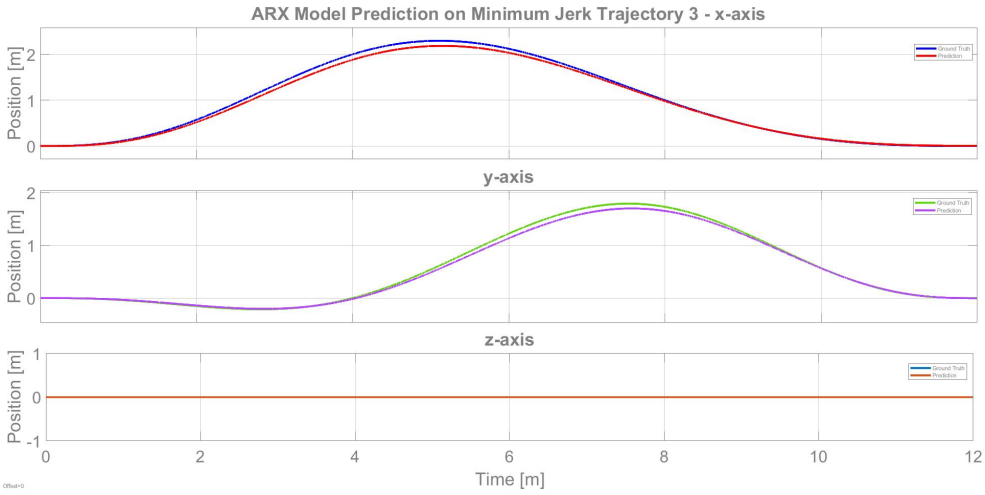


(a) Prediction and Ground truth

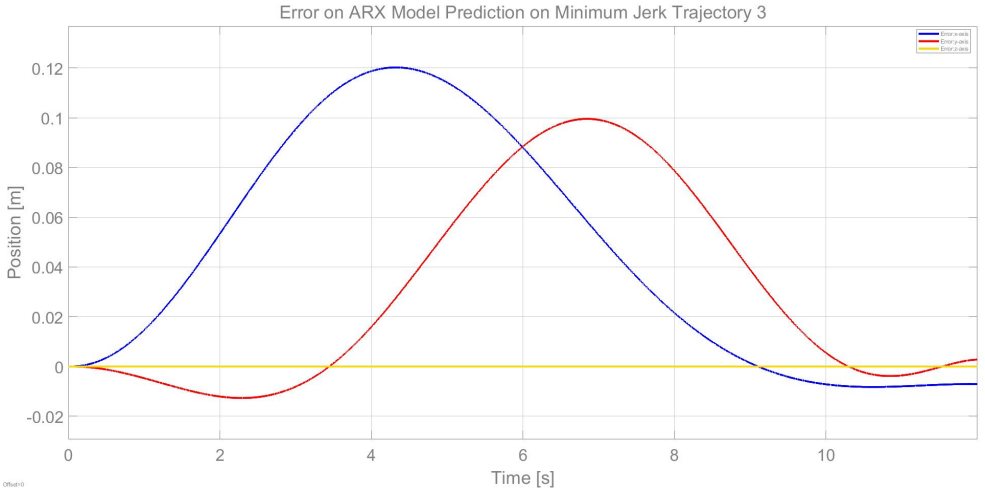


(b) Error between prediction and ground truth

Figure A.8: ARX Model prediction of minimum jerk trajectory 2.

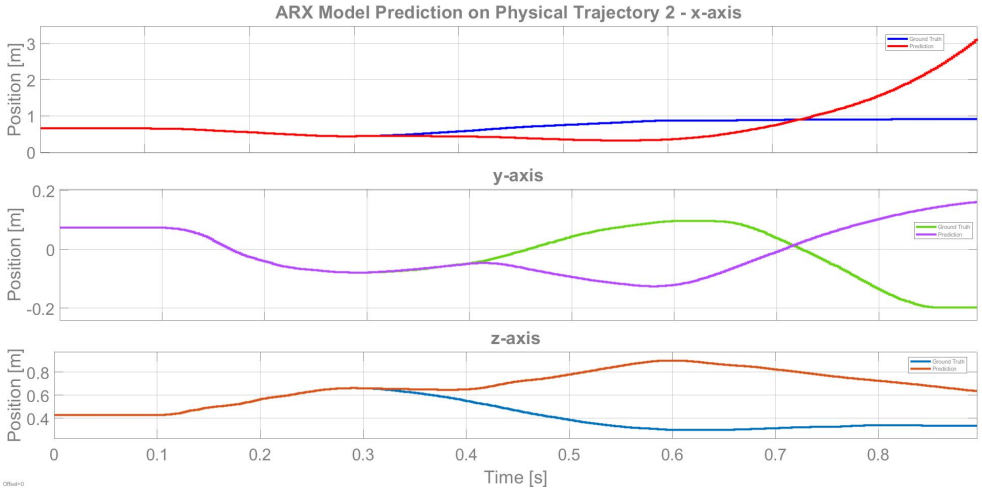


(a) Prediction and Ground truth

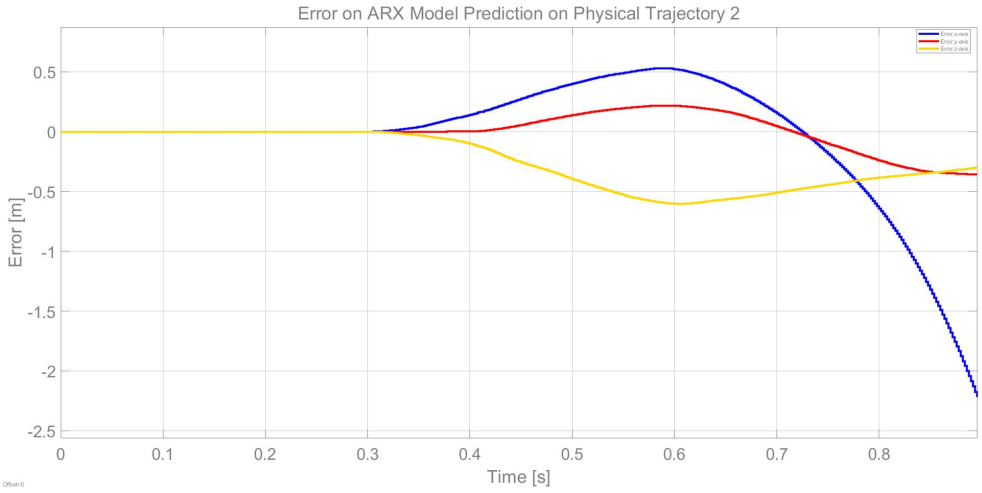


(b) Error between prediction and ground truth

Figure A.9: ARX Model prediction of minimum jerk trajectory 3.

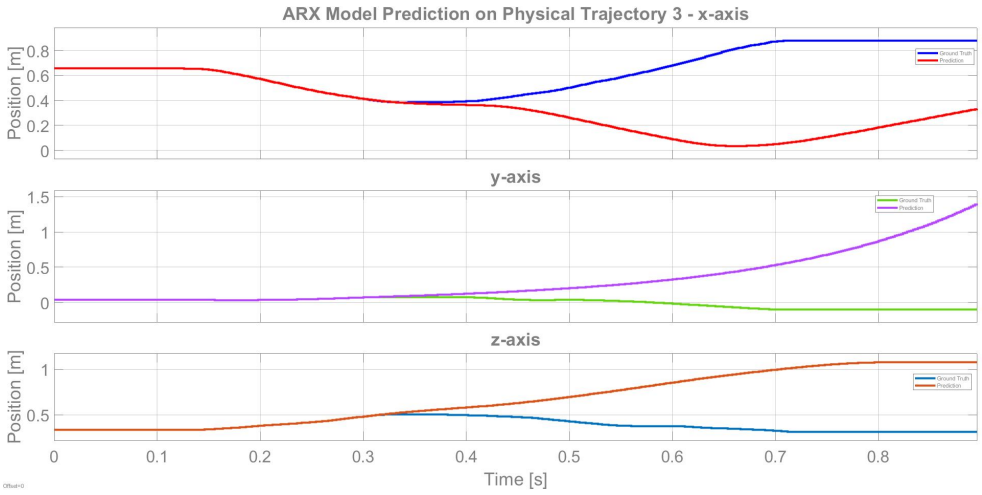


(a) Prediction and Ground truth

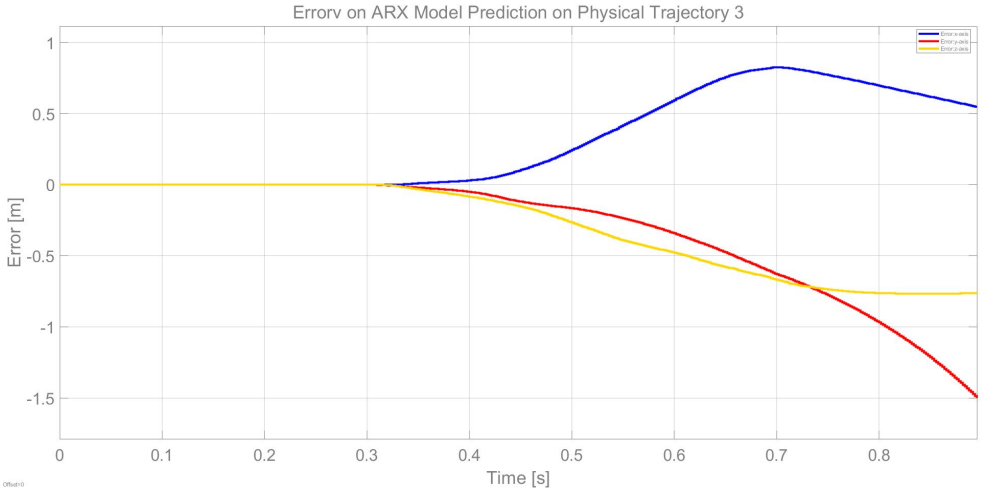


(b) Error between prediction and ground truth

Figure A.10: ARX Model prediction of physical trajectory 2.

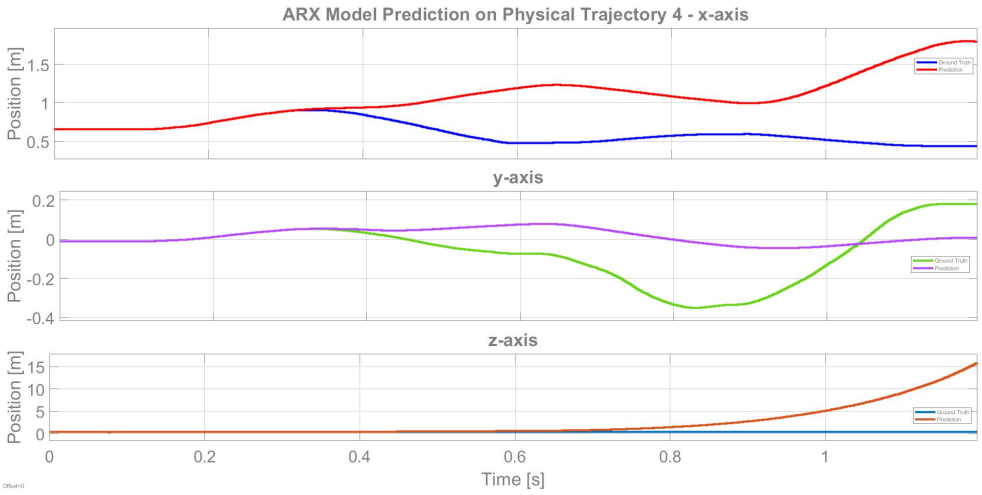


(a) Prediction and Ground truth

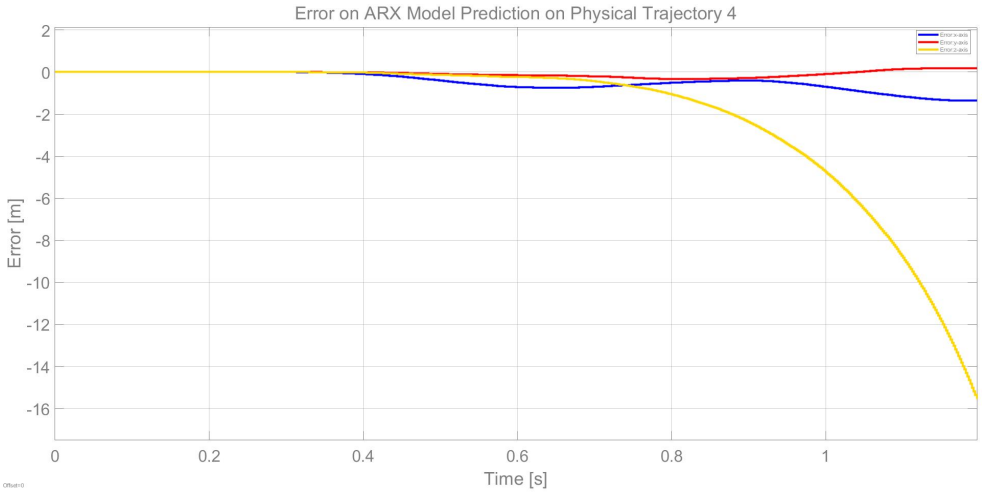


(b) Error between prediction and ground truth

Figure A.11: ARX Model prediction of physical trajectory 3.



(a) Prediction and Ground truth



(b) Error between prediction and ground truth

Figure A.12: ARX Model prediction of physical trajectory 4.

NARX Network

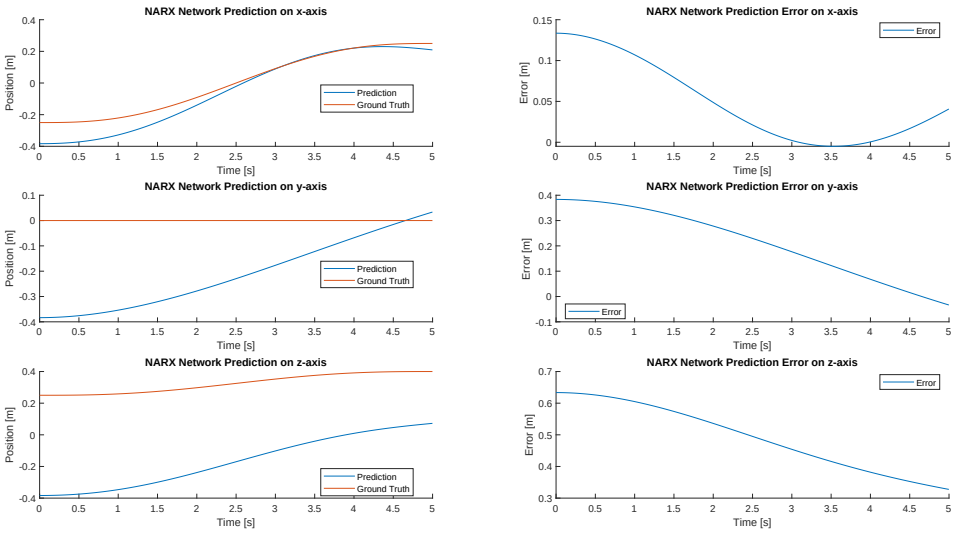


Figure A.13: NARX Network prediction on minimum jerk trajectory 1.

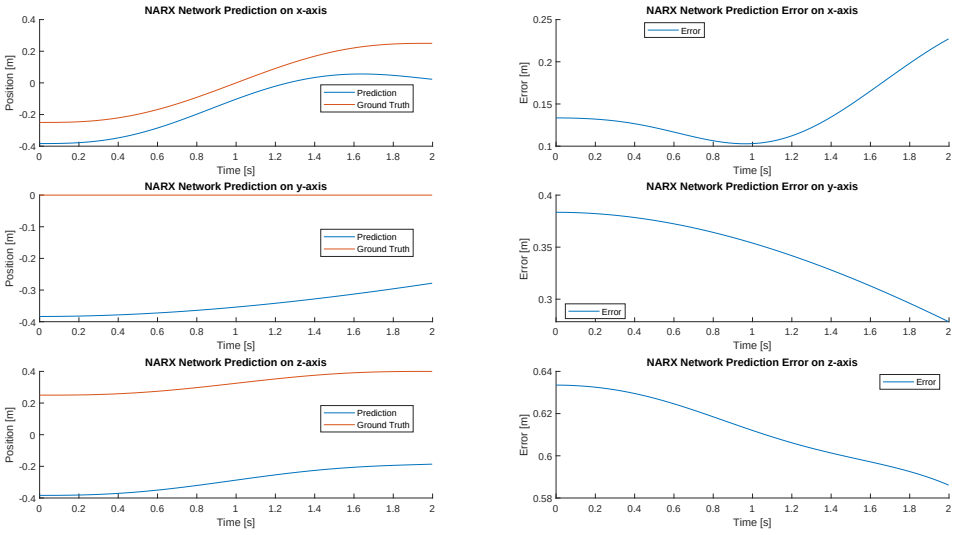


Figure A.14: NARX Network prediction on minimum jerk trajectory 2.

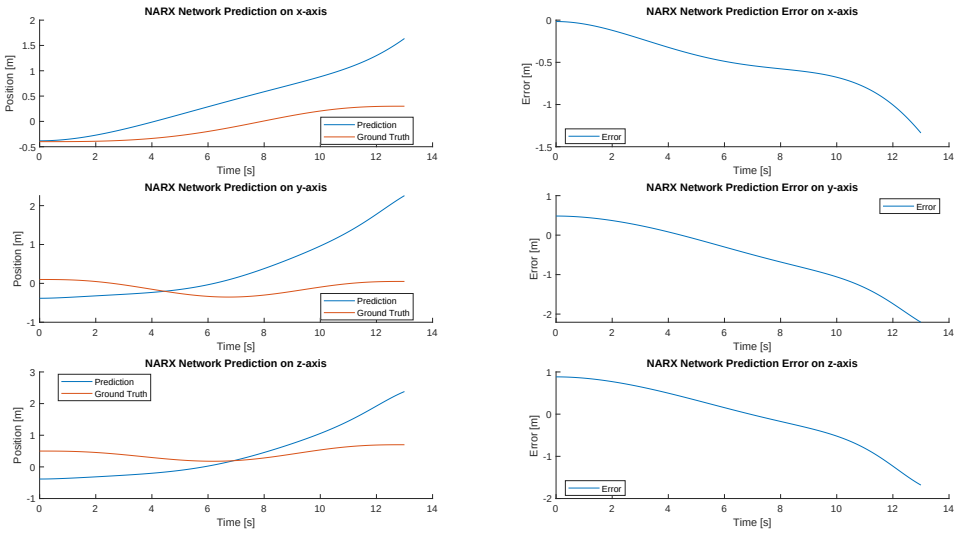


Figure A.15: NARX Network prediction on minimum jerk trajectory 4.

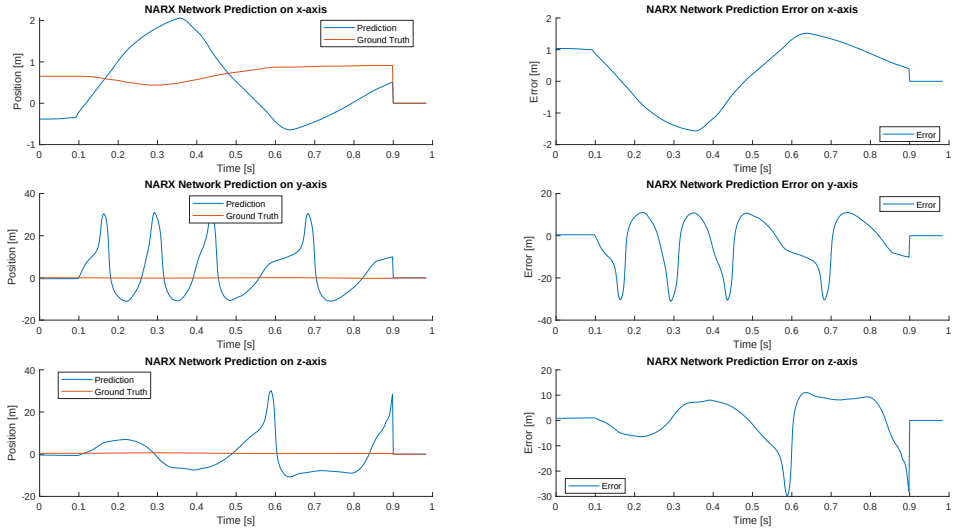


Figure A.16: NARX Network prediction on physical trajectory 2.

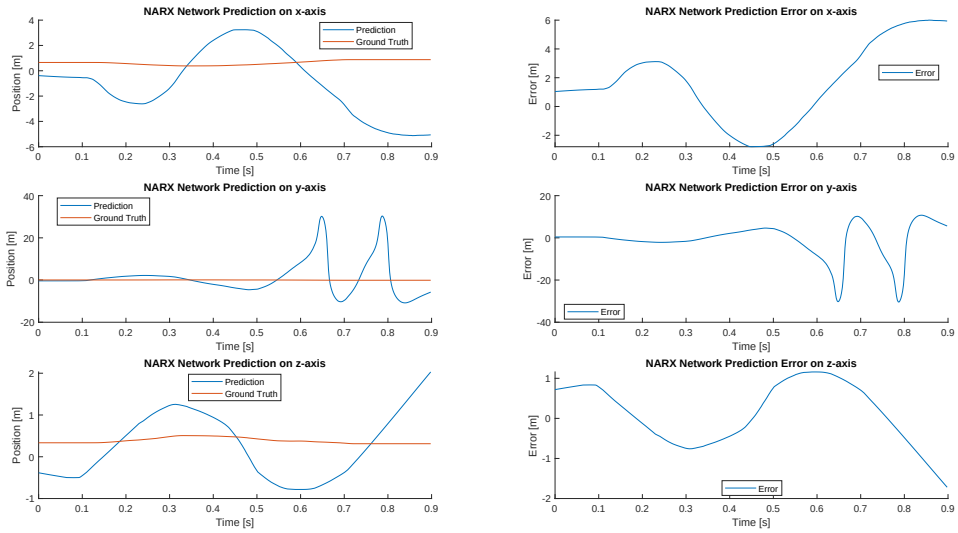


Figure A.17: NARX Network prediction on physical trajectory 3.

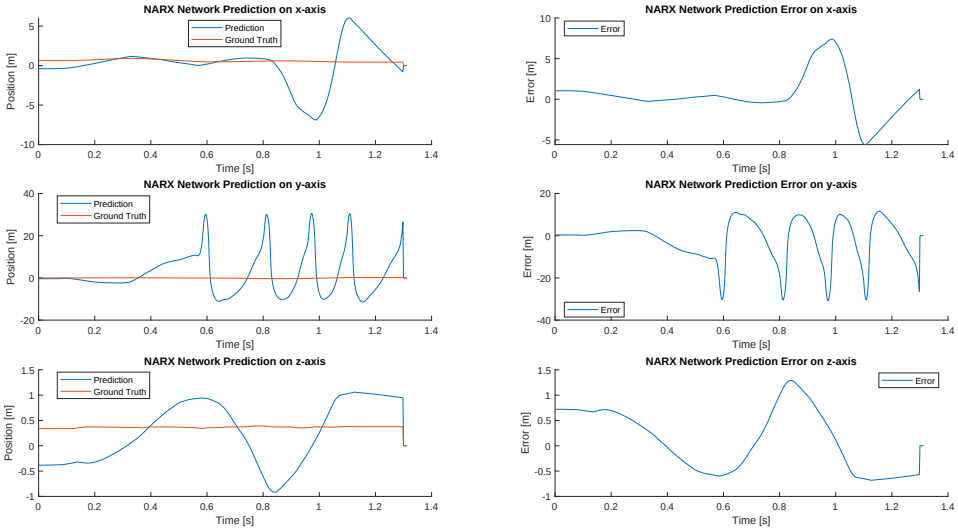


Figure A.18: NARX Network prediction on physical trajectory 4.

A.4 Human Impedance Additional Tests

IEFD

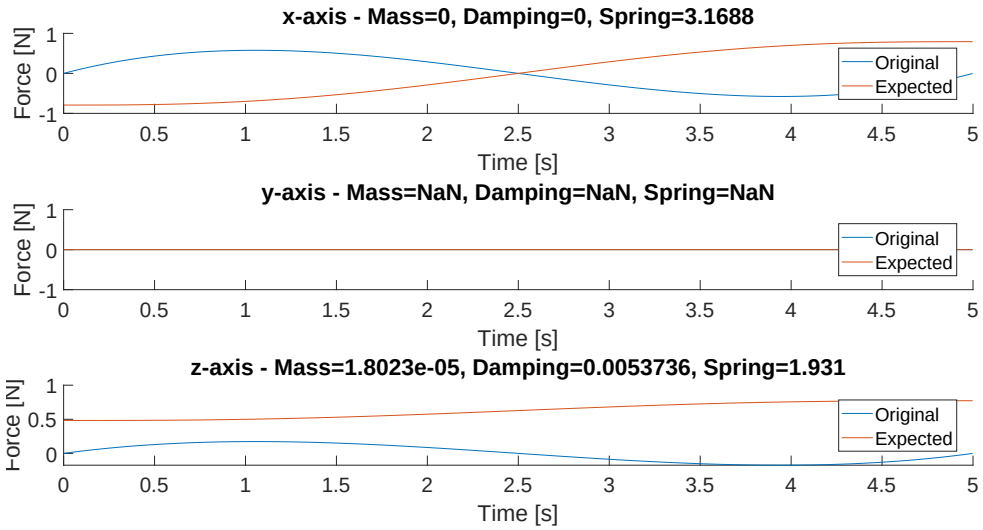


Figure A.19: Impedance estimation of minimum jerk trajectory 1 conducted on the whole trajectory using the IEFD method.

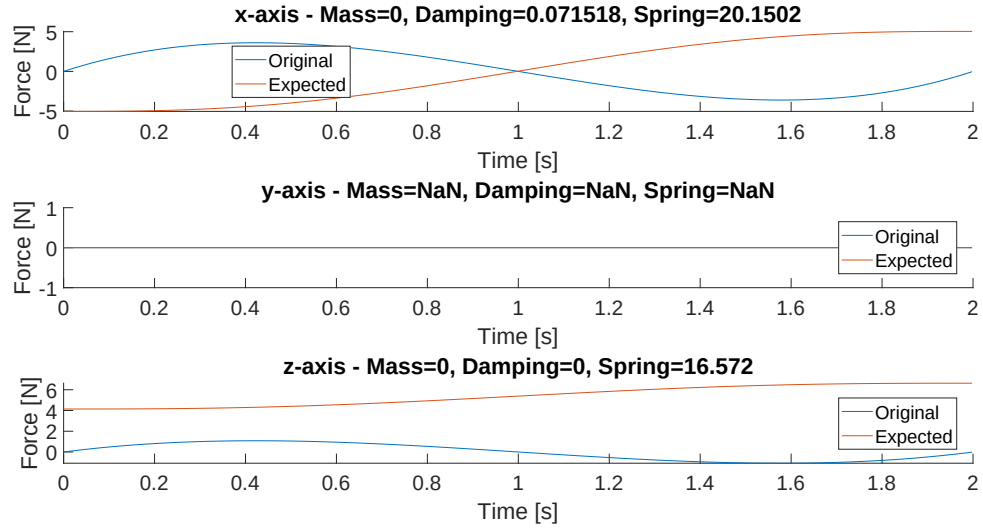


Figure A.20: Impedance estimation of minimum jerk trajectory 2 conducted on the whole trajectory using the IEFD method.

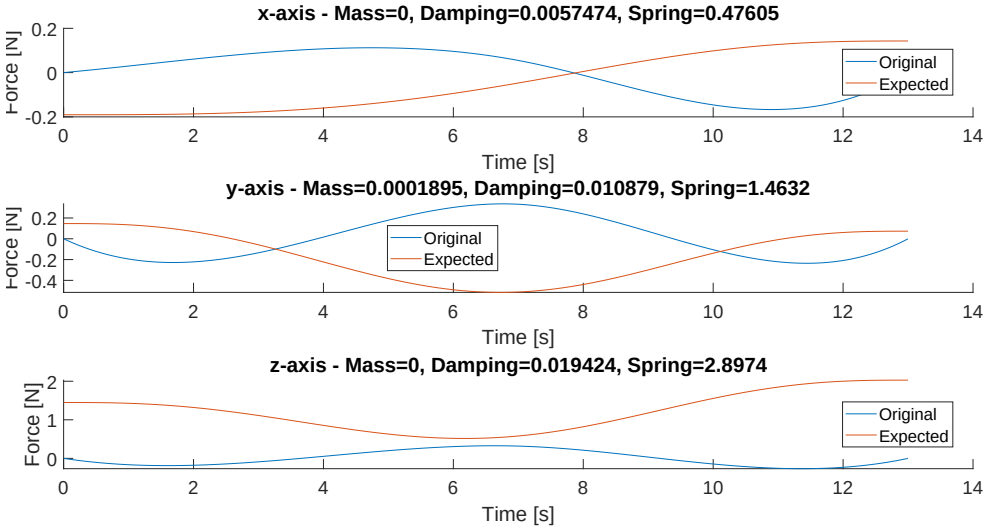


Figure A.21: Impedance estimation of minimum jerk trajectory 4 conducted on the whole trajectory using the IEFD method.

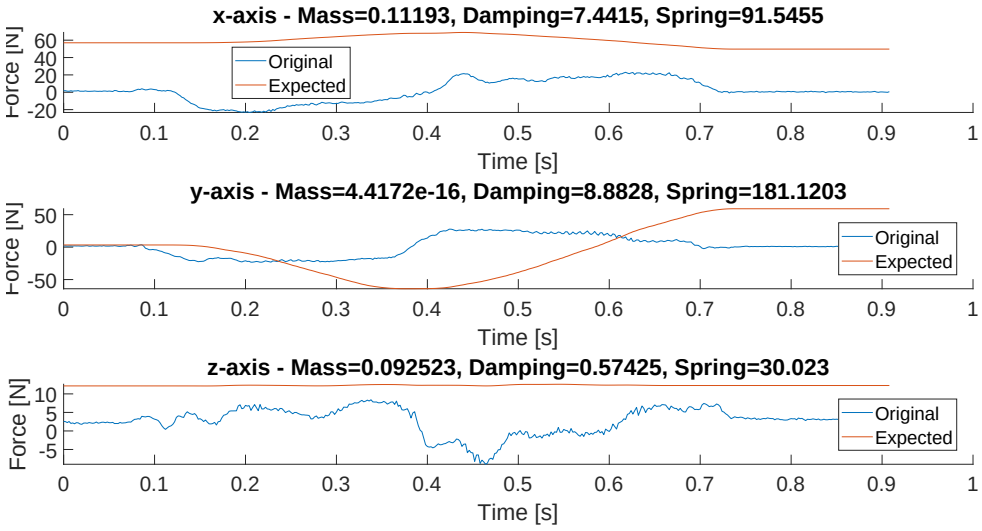


Figure A.22: Expected force for impedance estimation of physical trajectory 1 conducted on the whole trajectory using the IEFD method.

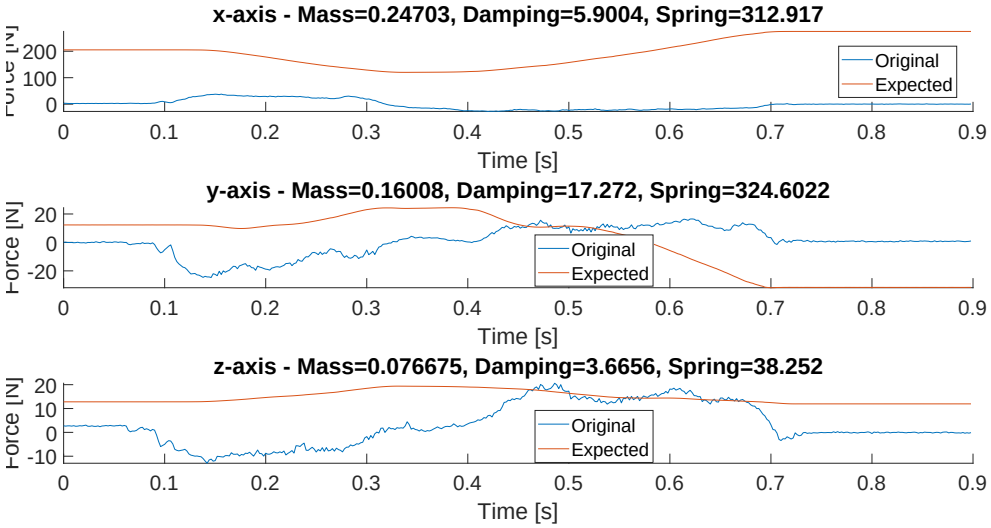


Figure A.23: Expected force for impedance estimation of physical trajectory 3 conducted on the whole trajectory using the IEFD method.

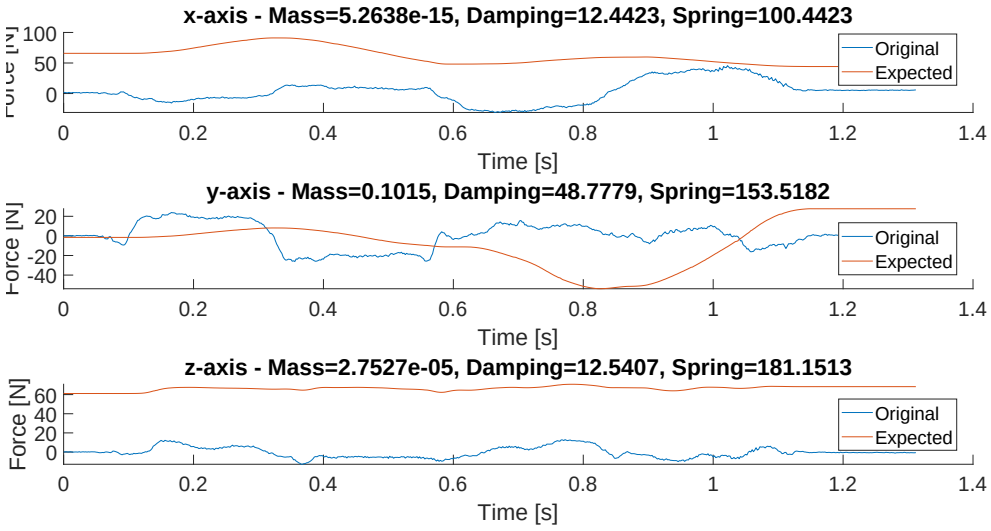


Figure A.24: Expected force for impedance estimation of physical trajectory 4 conducted on the whole trajectory using the IEFD method.

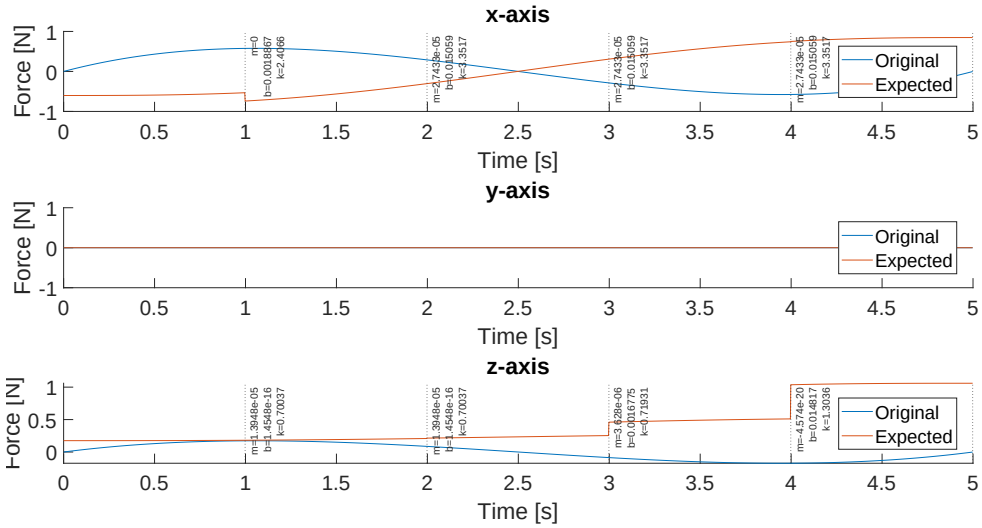


Figure A.25: Impedance estimation of minimum jerk trajectory 1, done partly using the IEFD method.

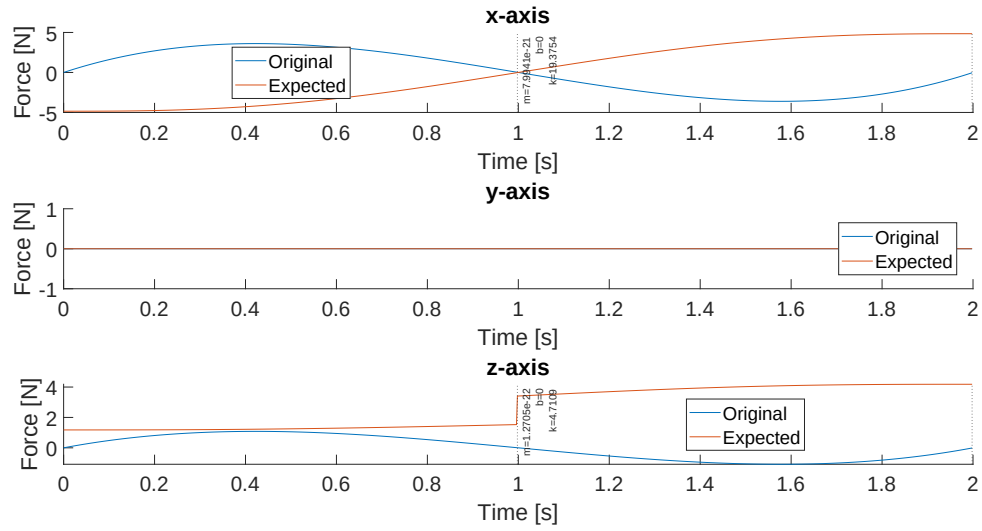


Figure A.26: Impedance estimation of minimum jerk trajectory 2, done partly using the IEFD method.

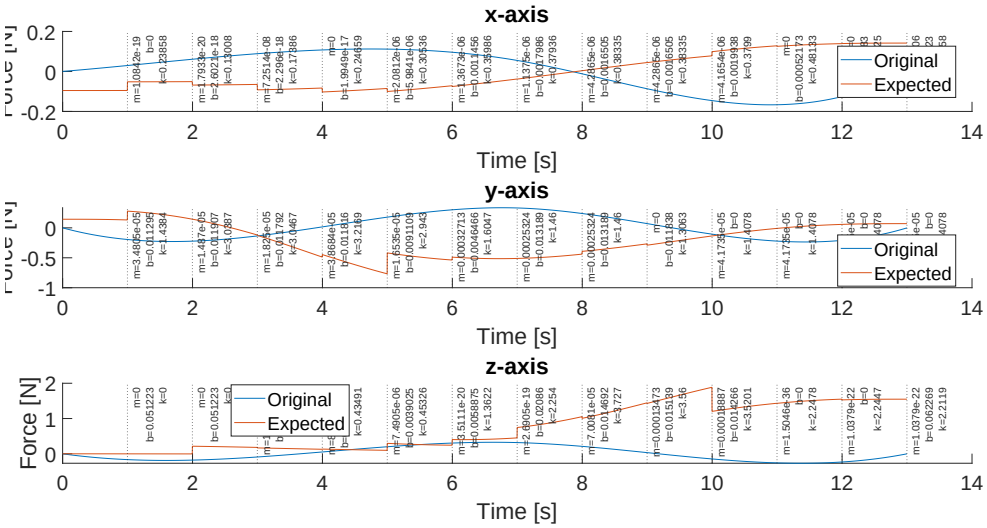


Figure A.27: Impedance estimation of minimum jerk trajectory 4, done partly using the IEFD method.

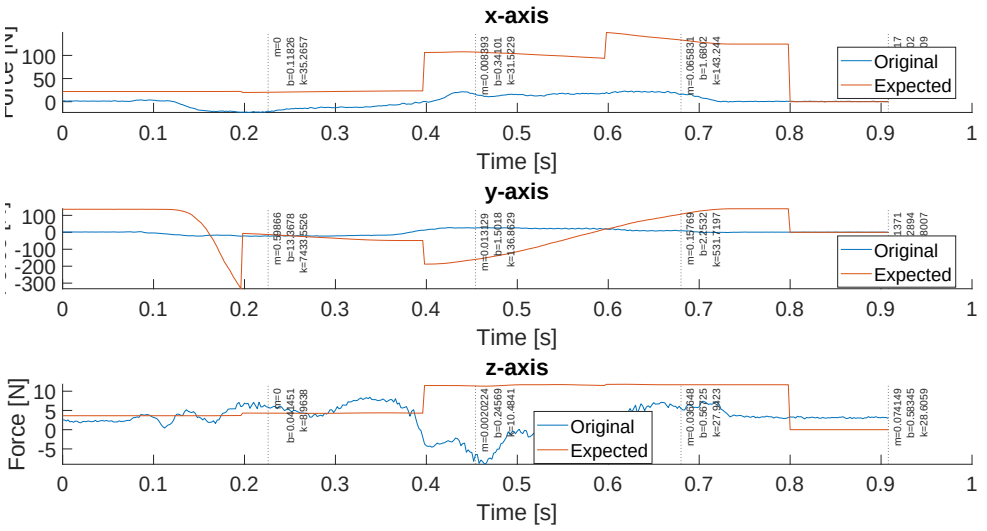


Figure A.28: Expected force for impedance estimation of physical trajectory 1, done partly using the IEFD method.

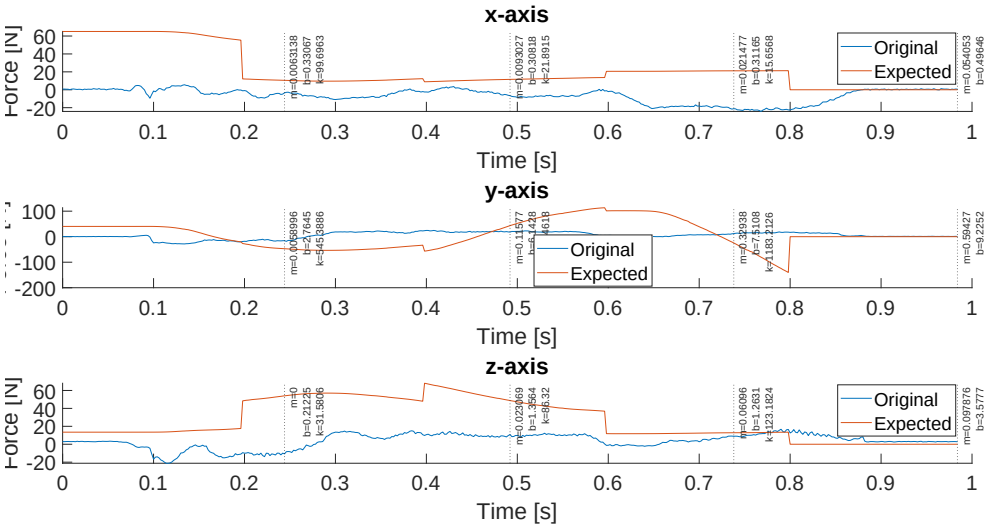


Figure A.29: Expected force for impedance estimation of physical trajectory 2, done partly using the IEFD method.

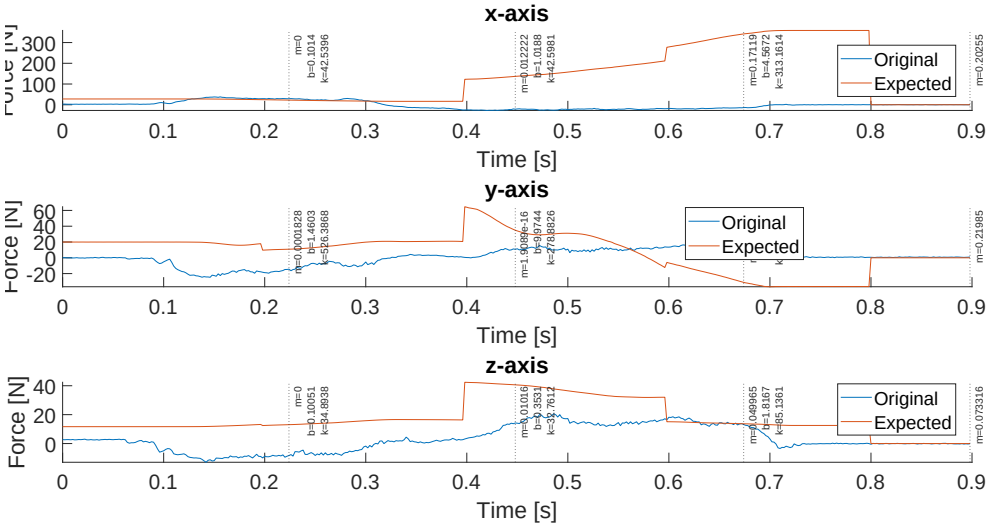
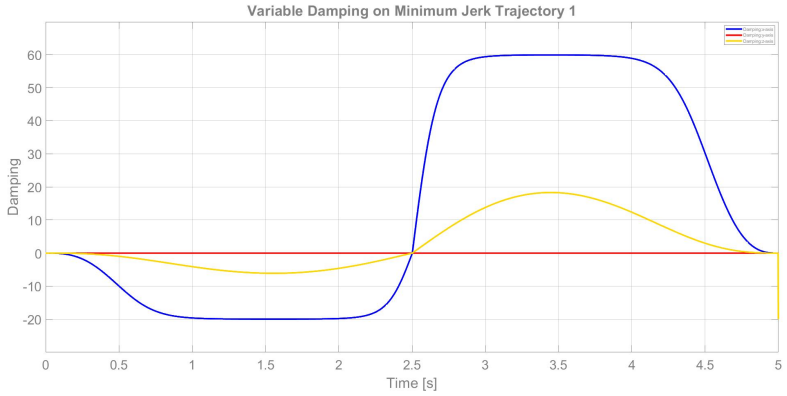
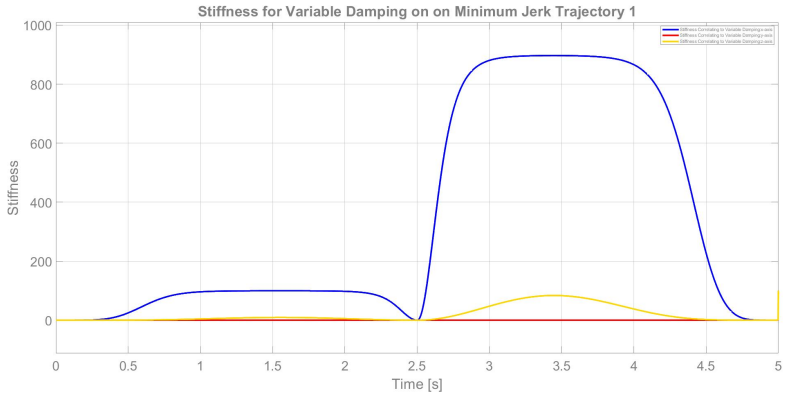


Figure A.30: Expected force for impedance estimation of physical trajectory 3, done partly using the IEFD method.

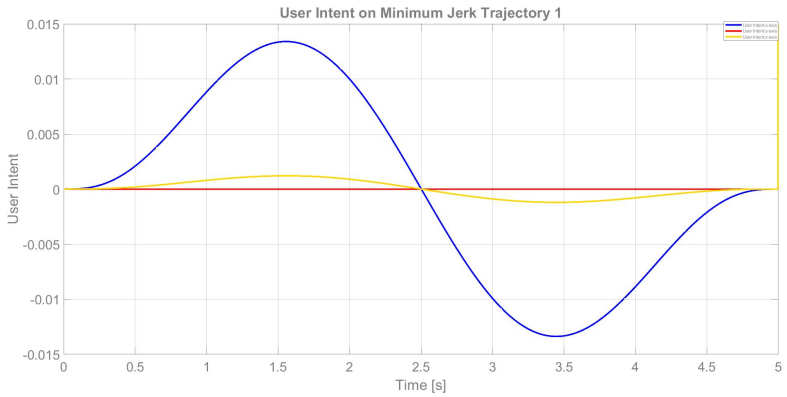
Variable Damping



(a) Variable damping

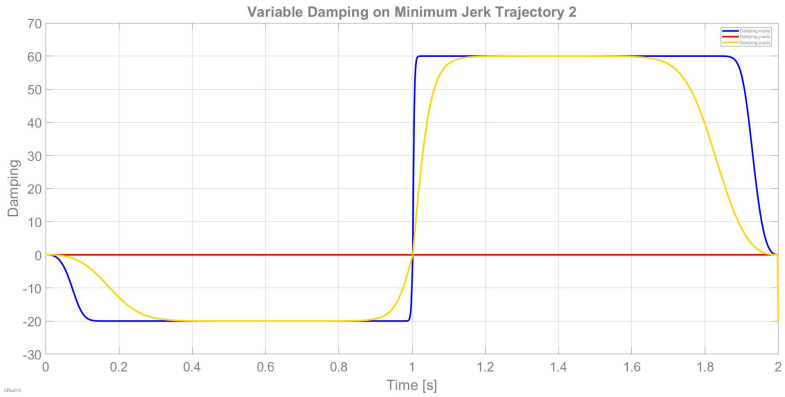


(b) Stiffness

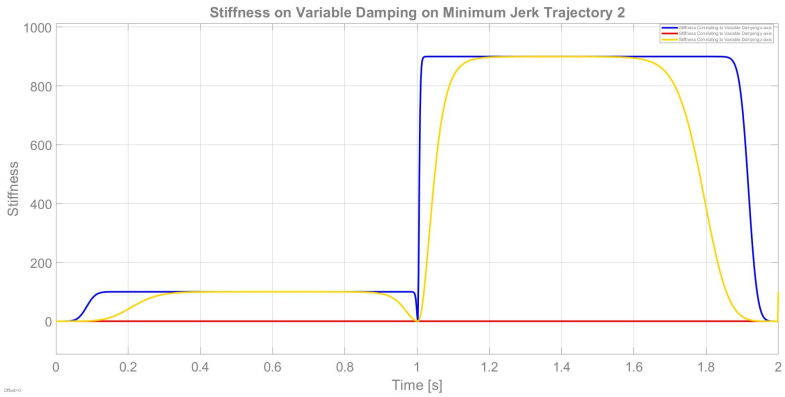


(c) User intent

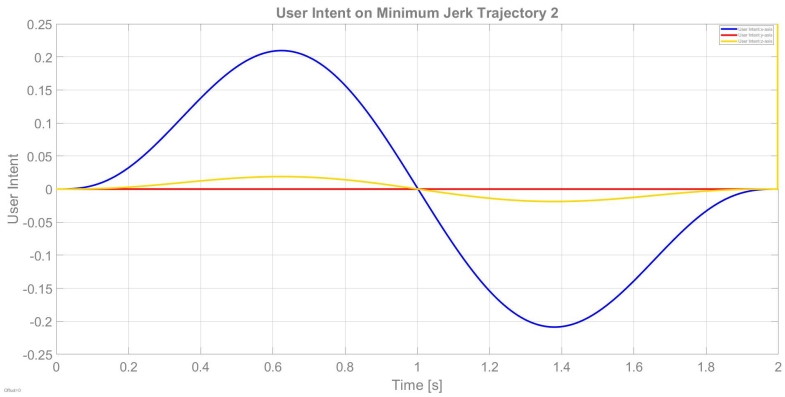
Figure A.31: Variable damping, stiffness, and user intent for minimum jerk trajectory 1.



(a) Variable damping

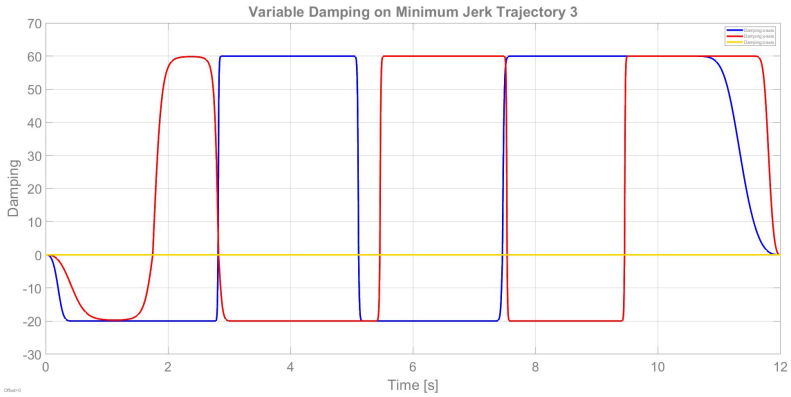


(b) Stiffness

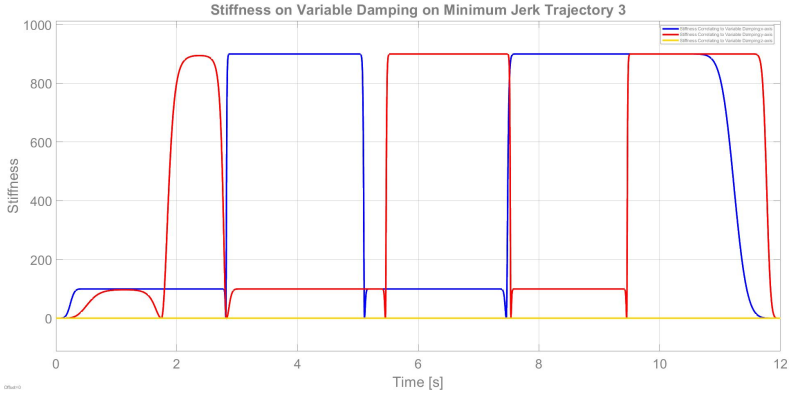


(c) User intent

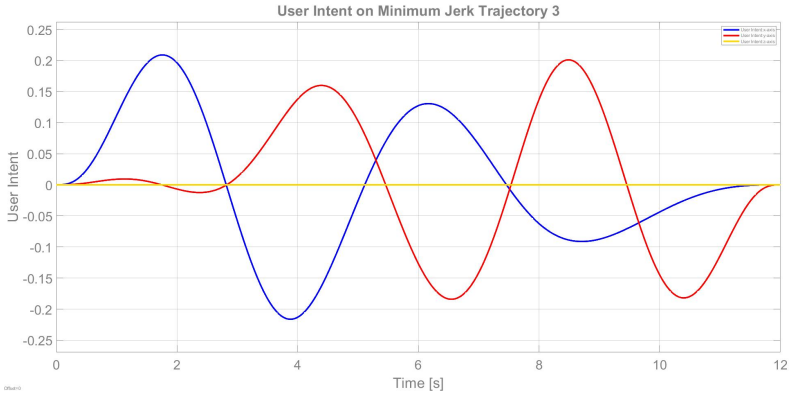
Figure A.32: Variable damping and user intent for minimum jerk trajectory 2.



(a) Variable damping



(b) Stiffness



(c) User intent

Figure A.33: Variable damping and user intent for minimum jerk trajectory 3.