Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Human-Robot Interaction Force Estimation for Transparency Control on Wearable Robots

Master Thesis

April 4, 2016

Supervised by
Dr. Thiago Boaventura

Author
Lisa Hammer

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Human-Robot Interaction Force Estimation for Transparency Control on Wearable Robots

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Hammer	Lisa

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 04.04.2016	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

Abstract

Exoskeletons have become one of the most popular research topics in robotics and rehabilitation within recent years. Thus, the EU Balance Project [3] was brought into life. It aims at developing an exoskeleton for postural balance control. This means that the exoskeleton should support the user whenever balance issues arise but be imperceptible otherwise. This type of exoskeleton could be used as support for healthy people as well as patients in everyday life or rehabilitation. Especially rehabilitation patients who can walk independently but have trouble retaining postural balance could benefit from this project.
The main role of ETH or more specifically ADRL in the project is the implementation of robust and flexible interaction control. The controller consists of two overall parts - the balance and the transparency controller. The transparency controller hereby takes care of making the exoskeleton imperceptible to the user, i.e. reducing all interaction forces due to joint friction, damping etc. down to zero. This part needs to work exceptionally well since it is active most of the time unless there are balance issues. To this end, a new transparency control paradigm was proposed in [7] specifically for this application. It uses an inverse dynamics feedforward as well as one acceleration and one force feedback term for interaction force reduction. Additionally it uses a Kalman filter to estimate interaction force, which cannot be directly measured. This thesis describes the first step of implementing this new control paradigm in both simulation and hardware.
First, the system was modelled in a linear fashion as two spring-coupled masses. This system was then simulated using MATLAB Simulink to get first information on the most important system parameters. This resulted in the notion that, for Kalman filter estimation, the crucial parameters were related to the attachment spring dynamics, which need to be known quite reliably in both value and overall dynamics to achieve estimation errors below 20%. For the controller itself, actuator saturation can become an issue when either the robot and human masses are very different or very fast movements are being performed. Furthermore, the overall quality of the Kalman filter estimate also influences controller performance significantly since the estimate is being used directly in the force feedback controller. Other tested parameters like noise and delays were negligible for both Kalman filter and controller performance as long as they were not too large. Human force models also seemed to have a smaller influence on Kalman filter performance than the correct spring dynamics model.
With these simulation results we started our hardware experiments. Some hardware adjustments were necessary due to limited sampling frequency capabilities in the previous setup. Thus, a new PCB interface between microcontroller and mechanical hardware was introduced to enable the usage of a new, faster microcontroller. Additionally,two electric motors were

installed to replace the hydraulic actuators which were previously used.
With this setup, first experiments on controller and Kalman filter performance were conducted. Both perform similar to what was seen in simulation as far as conclusions can be drawn from the available data.
Future work will have to show more detailed experimental results on the full control paradigm as well as a more sophisticated Kalman filter approach. Furthermore, the results will have to be adapted to the articulated case to draw final conclusions on performance of the real exoskeleton.

Acknowledgements

This thesis was made possible with the help of many people, some actively advising and some in the background.
Firstly I want to thank Thiago Boaventura for his continuous support and advice as well as always bringing in new ideas on what to try when everything else failed. Secondly I want to thank Jonas Buchli for the opportunity to participate in this project. Last but not least, a thank you is due to my parents as well as my friends who listened to my frustrations and excitement for 6 months and kept me sane.

Contents

List of Figures

Glossary

K	Spring Stiffness.
f_i	Interaction Force.
f_r	Robot Force.
f_h	Human Force.
f_{FF}	Feedforward Controller Force.
$f_{FB,a}$	Acceleration Feedback Controller Force.
$f_{FB,f}$	Force Feedback Controller Force.
λ	Characteristic Equation Root.
x_r	Robot Position.
$\dot{x_r}$	Robot Velocity.
$\ddot{x_r}$	Robot Acceleration.
x_h	Human Position.
$\dot{x_h}$	Human Velocity.
$\ddot{x_h}$	Human Acceleration.
m_r	Robot Mass.
m_h	Human Mass.

Acronyms

ADRL Agile & Dexterous Robotics Lab.

ETH Eidgenössische Technische Hochschule (Swiss Federal Institute of Technology) Zürich.

FC2D Force Control in Two Degrees Experimental Setup.

IIT Istituto Italiano di Tecnologia (Italian Institute of Technology).

IMU Inertial Measurement Unit.

MC Microcontroller.

RMSE Root Mean Square Error.

1 Introduction

With both robotics technology and the knowledge of human motion advancing during recent years, exoskeletons have become widely popular as a research topic combining the two fields. There are many different types of exoskeletons, ranging from fully trajectory controlled robots to purely supportive versions. The applications range from rehabilitation, e.g. Lokomat [6] by Swiss company Hocoma, over support functionality to human performance enhancement, both of which can be found in the function range of Japanese venture firm Cyberdyne's Hybrid Assistive Limb (HAL) [5]. As wide as the application range, as varied are the types of exoskeletons. While there are many projects on lower-limb exoskeletons, i.e. walking assistance, there also are projects on arm [1], hand [4] and full-body robots.
In the light of these developments, the EU Balance Project [3] was started in 2013 to develop a "platform-independent control strategy and architecture" [2] for exoskeletons. The main focus of the project is to improve postural balance of the user, thereby supporting them in everyday-life tasks in both clinical and home settings. This type of development is necessary since state of the art exoskeletons are often controlled quite rigidly and cannot account for disturbances, i.e. movement of the human, very well. This becomes an issue when the exoskeleton is worn by a healthy person or a person with at least some extent of motor control in the respective limb. These disturbances then induce significant balance problems, which in the worst case lead to a fall. Additionally, tasks like walking uphill and rapid direction or speed changes are difficult to do, even for healthy users, when working against the robot.
Therefore, a new control paradigm needs to be developed to follow human motion instead of inhibiting it. That means that the exoskeleton should generally be imperceptible and only support when needed, i.e. when postural balance assistance is required. This could be used for support of elderly people or people with motor impairment who tend to trip or stumble more often than usual. The exoskeleton would then prevent them from falling whenever a dangerous situation occurs but allow independent movement of the user while they can maintain postural balance on their own.
A basic function that an exoskeleton needs to keep from disturbing the user's motions is transparency control. Transparency in human-robot interaction is defined as the absence of all interaction forces between human and robot. To achieve transparency, all friction, damping, inertia etc. in the robot joints need to actively be compensated for. Ideally, the robot should then follow the user's motion without applying any force onto them.
State of the art transparency controllers use "interaction force feedback, impedance and admittance controllers, and/or electromyography" [7]. However, all of these methods either have stability or calibration issues, which decrease their performance outside of theoretical work

significantly. Therefore, a new acceleration-based transparency control concept was proposed in [7]. It uses acceleration measurements from IMUs which are attached to both robot and human to do feedforward and acceleration control. Additionally, the measurements are fed into a Kalman filter to estimate the interaction force, which is then used in an additional force controller resulting in three controllers in total.
Chapter 2 explains the system model and controller concept in more detail. Chapter 3 contains results and discussion of the system simulation. Chapter 4 describes the general hardware of the test setup. In Chapter 5, the development of new electronics for the test setup is described. Chapter 6 explains the experiments conducted on the test setup and the respective results. Finally, Chapter 7 concludes the work and gives an outlook on future research to be done.

2 System Model

2.1 Mechanical Model

We model our system as two spring-coupled masses. The masses correspond to the human and the exoskeleton, from here on also referred to as the robot. The spring represents the attachment between the exoskeleton and the human body with e.g. an elastic band. Both parts can apply a force or movement to the system, which leads to the spring producing an interaction force. In a first stage we model everything linearly for model simplicity reasons. This model can then later be adapted to an articulated case. Figure 2.1 shows a sketch of the system. The relationships between the three forces and the mass trajectories are as follows.

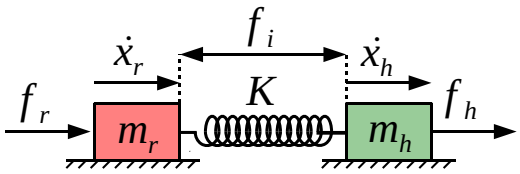


Figure 2.1: Spring-mass model of the exoskeleton. Adapted from [7].

$$\dot{f}_i = K(\dot{x}_r - \dot{x}_h) \tag{2.1a}$$
$$f_r - f_i = m_r \ddot{x}_r \tag{2.1b}$$
$$f_h + f_i = m_h \ddot{x}_h \tag{2.1c}$$

Combining them leads to

$$f_h + f_r = m_r \ddot{x}_r + m_h \ddot{x}_h \tag{2.2a}$$
$$= \frac{m_r \ddot{f}_i}{K} + m_r \ddot{x}_h + m_h \ddot{x}_h \tag{2.2b}$$

which then results in

$$m_r \ddot{f}_i + f_i K = K(f_r - m_r \ddot{x}_h) \tag{2.2c}$$

2.2 Controller Concept

2.2.1 Schematic

As opposed to state of the art impedance or admittance control schemes for doing transparency control on the system, a new acceleration-based controller framework was proposed in [7]. A block diagram of the system including controllers can be seen in Figure 2.2, where the three blocks on the right represent the mechanical model descibed in Section 2.1 and the four blocks on the left form the controller.

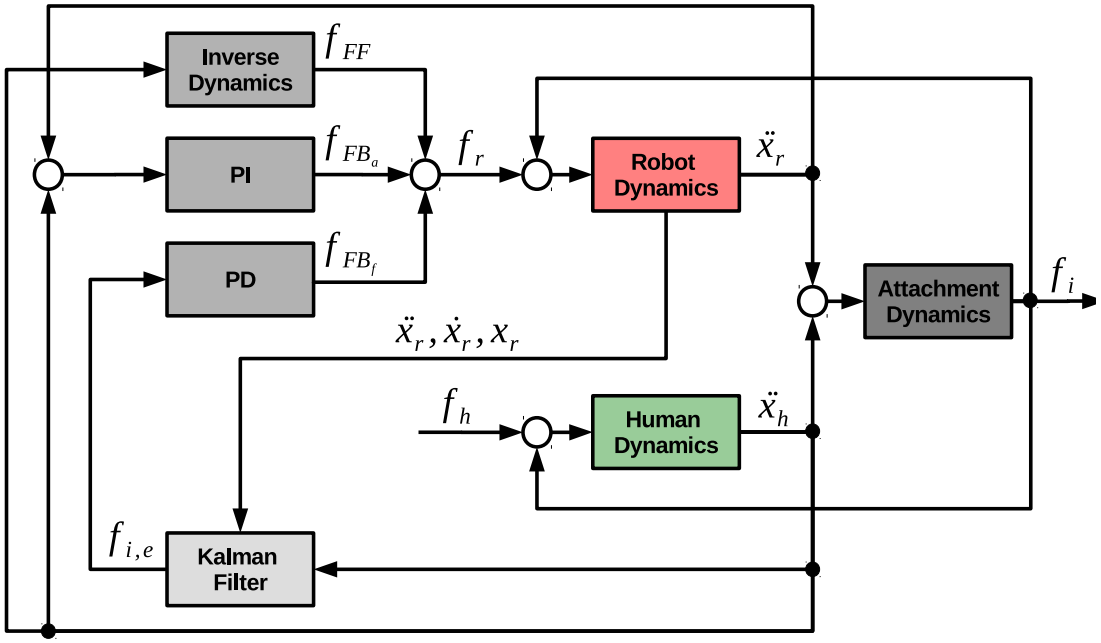


Figure 2.2: Controller schematic of the whole system. Adapted from [7]. Robot Dynamics: Inertial and actuator terms. Human Dynamics: Inertial terms. Attachment Dynamics: Spring term.

2.2.2 Feedforward

The first block of the controller part is the "Inverse Dynamics" block. The inverse dynamics term is given by

$$f_{FF} = m_r \ddot{x}_h \tag{2.3}$$

which, in the ideal case, exactly compensates for the force induced on the robot by a human acceleration. With pure feedforward, the system dynamics become

$$m_r \ddot{f}_i + f_i K = K(m_r \ddot{x}_h - m_r \ddot{x}_h) = 0 \tag{2.4a}$$

resulting in a characteristic equation of

$$m_r\lambda^2 + K = 0 \tag{2.4b}$$

with roots

$$\lambda_{1,2} = \pm\sqrt{-\frac{K}{m_r}} \tag{2.4c}$$

This corresponds to a purely oscillatory response with a frequency of $f = \sqrt{\frac{K}{m_r}}$.

2.2.3 Feedback

The next part is the feedback controller, consisting of a PI acceleration and a PD force controller. The controller terms are given as acceleration feedback

$$f_{FB,a} = K_{p,a}(\ddot{x}_h - \ddot{x}_r) + K_{i,a}(\dot{x}_h - \dot{x}_r) \tag{2.5}$$

and force feedback

$$f_{FB,f} = K_{p,f}(f_{des} - f_i) + K_{d,f}(\dot{f}_{des} - \dot{f}_i) \tag{2.6a}$$

which, with our transparency goal of the desired force $f_{des} = 0$, becomes

$$f_{FB,f} = -K_{p,f}f_i - K_{d,f}\dot{f}_i \tag{2.6b}$$

In the ideal case, these two feedback controllers are redundant. However, since we do not have perfect information on the interaction force, we use the acceleration feedback as support. On the other hand, using acceleration feedback only is not sufficient to reduce the interaction force to zero. An example for which the pure acceleration feedback does not work is a movement starting from rest with an initial spring compression, i.e. an initial f_i. Since the acceleration controller will not see the acceleration leading to that force, it will not counteract it, resulting in a fixed force offset. Therefore we need both controllers to achieve the most transparent behaviour possible.
With the feedback controllers we always use the feedforward as well, which leads to the following system dynamics.

$$\begin{aligned} m_r\ddot{f}_i + f_iK &= K(m_r\ddot{x}_h + K_{p,a}(\ddot{x}_h - \ddot{x}_r) + K_{i,a}(\dot{x}_h - \dot{x}_r) - K_{p,f}f_i - K_{d,f}\dot{f}_i - m_r\ddot{x}_h) \\ &= K(K_{p,a}(\ddot{x}_h - \ddot{x}_r) + K_{i,a}(\dot{x}_h - \dot{x}_r) - K_{p,f}f_i - K_{d,f}\dot{f}_i) \end{aligned} \tag{2.7a}$$

This results in a characteristic equation of

$$(m_r + K_{p,a})\lambda^2 i + (KK_{d,f} + K_{i,a})\lambda + K(1 + K_{p,f}) = 0 \tag{2.7b}$$

with roots

$$\lambda_{1,2} = \frac{-(KK_{d,f} + K_{i,a}) \pm \sqrt{(KK_{d,f} + K_{i,a})^2 - 4K(m_r + K_{p,a})(1 + K_{p,f})}}{2(m_r + K_{p,a})} \tag{2.7c}$$

In the ideal case, we would tune both feedback controllers to the same critically damped behaviour to then compare their performance. To find these controllers, we examine the roots for each feedback controller separately. For critical damping, we need the radicand to be zero, which gives us the first two conditions for the controller parameters.

$$KK_{d,f}^2 - 4Km_r(1 + K_{p,f}) \stackrel{!}{=} 0 \tag{2.8a}$$

$$K_{i,a}^2 - 4K(m_r + K_{p,a}) \stackrel{!}{=} 0 \tag{2.8b}$$

The third condition arises from the fact that we want both controllers to behave in the same way to be able to compare them. This, together with the radicand of zero gives us

$$\frac{KK_{d,f}}{2m_r} \stackrel{!}{=} \frac{K_{i,a}}{2(m_r + K_{p,a})} \tag{2.8c}$$

By choosing one parameter freely, we can then determine the other three according to these conditions.

2.2.4 Lyapunov Stability

For the situation of both feedback controllers activated at the same time we conducted a Lyapunov stability analysis to get stability conditions for this general case. To this end, we first write the combined controller forces from Equations 2.3, 2.5 and 2.6b as an overall robot force in terms of f_i with the help of the system dynamics from Equations 2.1a-2.1c.

$$f_r = m_r\ddot{x}_h - K_{p,f}f_i - K_{d,f}\dot{f}_i - \frac{K_{p,a}}{k_a}\ddot{f}_i - \frac{K_{i,a}}{k_a}\dot{f}_i \tag{2.9}$$

The whole system dynamics from Equation 2.2c then becomes

$$m_r\ddot{f}_i + k_a f_i = k_a(m_r\ddot{x}_h - K_{p,f}f_i - K_{d,f}\dot{f}_i - \frac{K_{p,a}}{k_a}\ddot{f}_i - \frac{K_{i,a}}{k_a}\dot{f}_i - m_r\ddot{x}_h) \tag{2.10a}$$

which leads to

$$\ddot{f}_i(m_r + K_{p,a}) + \dot{f}_i K_{i,a} + f_i k_a(1 + K_{p,f} + K_{d,f}) = 0 \tag{2.10b}$$

Separating this into two first-order equations yields

$$\begin{pmatrix} \ddot{f}_i \\ \dot{f}_i \end{pmatrix} = \begin{pmatrix} \frac{-K_{i,a}}{(m_r+K_{p,a})} & \frac{-K_a(1+K_{p,f}+K_{d,f})}{(m_r+K_{p,a})} \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \dot{f}_i \\ f_i \end{pmatrix} \tag{2.11}$$

for which we define

$$F = \begin{pmatrix} \frac{-K_{i,a}}{(m_r+K_{p,a})} & \frac{-K_a(1+K_{p,f}+K_{d,f})}{(m_r+K_{p,a})} \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ 1 & 0 \end{pmatrix} \tag{2.12}$$

According to [8] we want to find a Lyapunov function of the form

$$V(x) = x^T P x, \text{ where x} = \begin{pmatrix} \dot{f}_i \\ f_i \end{pmatrix} \tag{2.13}$$

From the derivative

$$\dot{V}(x) = \dot{x}^T P x + x^T P \dot{x} \tag{2.14}$$

plugging in our system dynamics of the form $\dot{x} = Fx$ yields the Lyapunov equation

$$F^T P + PF = -Q \tag{2.15}$$

where, if Q is any positive definite matrix and all eigenvalues of F are in the left half plane, the solution P will be positive as well. For better readability we solve the equation in terms of α and β. We choose the most simple positive definite Q-matrix possible, which is the 2x2 identity matrix.

$$\begin{pmatrix} \alpha & 1 \\ \beta & 0 \end{pmatrix} \begin{pmatrix} p & q \\ q & r \end{pmatrix} + \begin{pmatrix} p & q \\ q & r \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.16}$$

This results in three equations

$$2\alpha p + 2q = -1 \tag{2.17a}$$

$$\alpha q + r + p\beta = 0 \tag{2.17b}$$

$$2\beta q = 1 \tag{2.17c}$$

Solving them gives us p, q and r, which we can expand again with our real α and β values.

$$\begin{aligned} p &= \frac{1}{2\alpha}(\frac{1}{\beta} - 1) \\ &= \frac{(m_r + K_{p,a})}{2K_{i,a}}(\frac{(m_r + K_{p,a})}{k_a(1 + K_{p,f} + K_{d,f})} + 1) \end{aligned} \tag{2.18}$$

$$\begin{aligned} q &= -\frac{1}{2\beta} \\ &= \frac{(m_r + K_{p,a})}{2k_a(1 + K_{p,f} + K_{d,f})} \end{aligned} \tag{2.19}$$

(2.20)

$$r = \frac{\alpha}{2\beta} - \frac{1}{2\alpha}(1-\beta) \tag{2.21}$$
$$= \frac{K_{i,a}}{2(1+K_{p,f}+K_{d,f})k_a} - \frac{(m_r+K_{p,a})}{2K_{i,a}} - \frac{k_a(1+K_{p,f}+K_{d,f})}{2K_{i,a}}$$

We want to make P positive definite. Therefore we determine the leading principal minors μ_i of P, which for a positive definite matrix all have to be greater than zero.

$$\mu_1 = p \tag{2.22a}$$
$$\mu_2 = pr - q^2 \tag{2.22b}$$

Setting both μ_i greater than zero leads us to the following conditions for Lyapunov stability or our system.

$$m_r + K_{p,a} + k_a(1+K_{p,f}+K_{d,f}) > 0 \tag{2.23a}$$
$$K_{i,a}^2 + k_a(1+K_{p,f}+K_{d,f})^2 - (m_r+K_{p,a})^2 > 0 \tag{2.23b}$$

The first condition will always be fulfilled as long as we set our control parameters to positive values. The second condition then gives us a relationship between the parameters to keep the system Lyapunov stable.

2.2.5 Kalman Filter

The final part of the control system is a Kalman filter. It is not straightforward to attach force sensors to an elastic attachment like the one that will be used in the exoskeleton and get an accurate readout. Thus, we might not be able to measure force directly. Since we want to control interaction force, we need a means to determine interaction force, ideally, without measuring it. This is achieved by using a Kalman filter, which uses a model of the system dynamics to estimate interaction force from related measurements.
Generally, a Kalman filter consists of two stages. The first one is the prediction stage, where the new system state is estimated according to the system model. The second stage, i.e. the correction stage, takes the prediction and corrects it with respect to the available measurements. The equations for this are as follows.

$$x_k = Ax_{k-1} + Bu_{k-1} \qquad \text{(predicts the new state)} \tag{2.24a}$$
$$P_k = AP_{k-1}A^T + Q \qquad \text{(predicts the new error covariance)} \tag{2.24b}$$

$$K_k = P_k H^T (H P_k H^T + R)^{-1} \quad \text{(determines the Kalman gain)} \tag{2.25a}$$

$$x_k = x_k + K_k (y_k - H x_k) \quad \text{(corrects the state estimate)} \tag{2.25b}$$

$$P_k = (I - K_k H) P_k \quad \text{(corrects the error covariance)} \tag{2.25c}$$

Hereby, x and y correspond to the system state and the measurement vector. They are updated at each timestep just like the control input u. The A and B matrices are the state matrices, which describe the propagation of the state from one timestep to the next, i.e. $x_{k+1} = Ax_k + Bu_k$ where x_k and u_k are the respective state and control input at timestep k. H is the measurement matrix, which relates the measurements to the system states. Q and R are covariance matrices describing the uncertainty in the process itself, i.e. state propagation, and the measurements, i.e. noise, delay, etc., respectively. P is the state covariance matrix. At each step it is updated to the current uncertainty in the estimated states. Finally, K is the Kalman gain. It is iteratively updated and corresponds to the correction factor which relates the propagated state and the current measurement.
The states and the state space matrices that we use in our Kalman filter are the following. P and K are not listed since they are updated at every step. These matrices are untested on the microcontroller as of now as explained in Chapter 6.

$$x = \begin{pmatrix} x_r & \ddot{x}_r & x_h & \ddot{x}_h \end{pmatrix} \quad \text{(state)} \tag{2.26}$$

$$y = \begin{pmatrix} x_r & \ddot{x}_r & x_h & \ddot{x}_h \end{pmatrix} \quad \text{(measurement)} \tag{2.27}$$

$$u = \begin{pmatrix} \dot{f}_r & 0 \end{pmatrix} \quad \text{(control input)} \tag{2.28}$$

$$A = \begin{pmatrix} 1 & \frac{dt^2}{2} & 0 & 0 \\ 0 & 1 & \frac{dt^2}{2} & 0 \\ 0 & -\frac{K}{m_r} & 0 & \frac{K}{m_r} \\ 0 & \frac{K}{m_h} & 0 & -\frac{K}{m_h} \end{pmatrix} \quad \text{(system matrix)} \tag{2.29}$$

$$B = \begin{pmatrix} 0 & 0 \\ \frac{1}{m_r} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_h} \end{pmatrix} \quad \text{(input matrix)} \tag{2.30}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(measurement matrix)} \tag{2.31}$$

$$Q = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \quad \text{(process noise covariance matrix)} \tag{2.32}$$

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \quad \text{(measurement noise covariance matrix)} \tag{2.33}$$

3 Simulation

3.1 Simulink Model

Before we went over to the experimental setup we simulated our model extensively to get an overview of the relevant parameters for a good performance of our system. Figures 3.1 to 3.4 show the top level of the Simulink model used to simulate the system. The following section shows the simulation results. A more detailed reasoning and discussion of all of the cases can be found in the Discussion section.

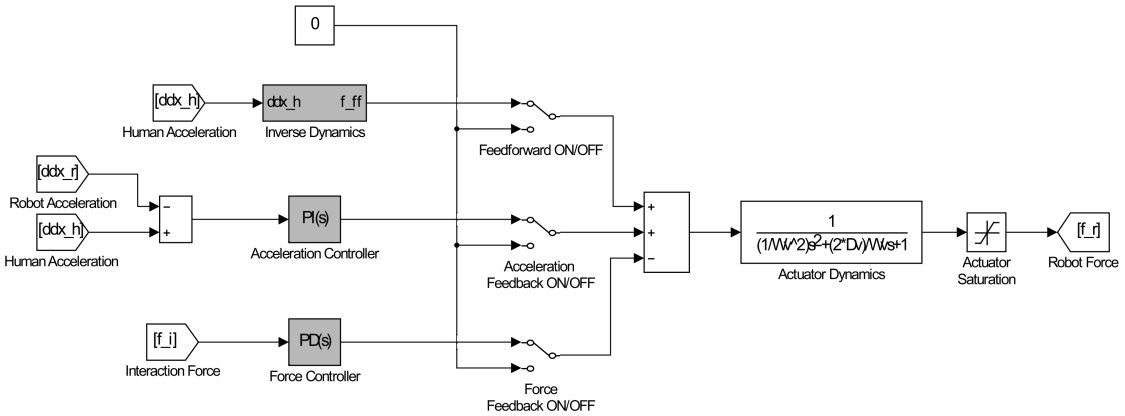


Figure 3.1: Controller model with possibility to switch all controllers on separately. The generated robot force is run through a second order system and a saturation block to emulate real actuator behaviour. The latter two blocks could also be considered part of the robot dynamics block in the Figure 3.2 as previously mentioned in Section 2.2.1.

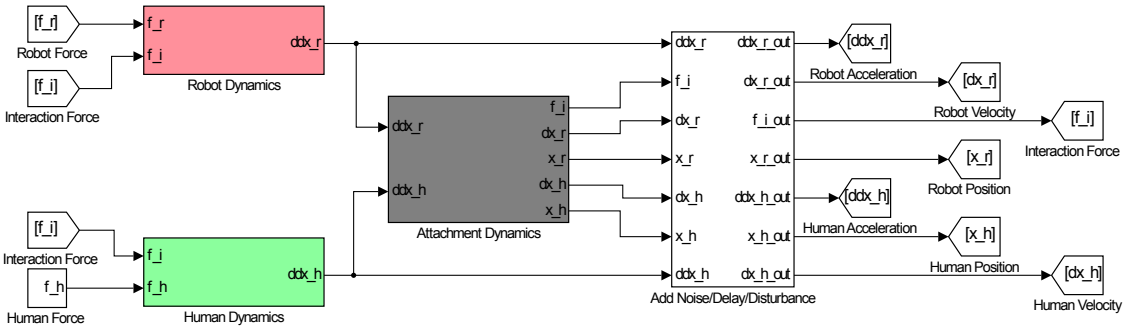


Figure 3.2: Model of the mechanical system. The three coloured blocks correspond to the respective blocks in Section 2.2.1 and contain the mechanical equations from Section 2.1. The white block adds noise and delay onto all signals and a pulse disturbance onto the interaction force to emulate real measurements.

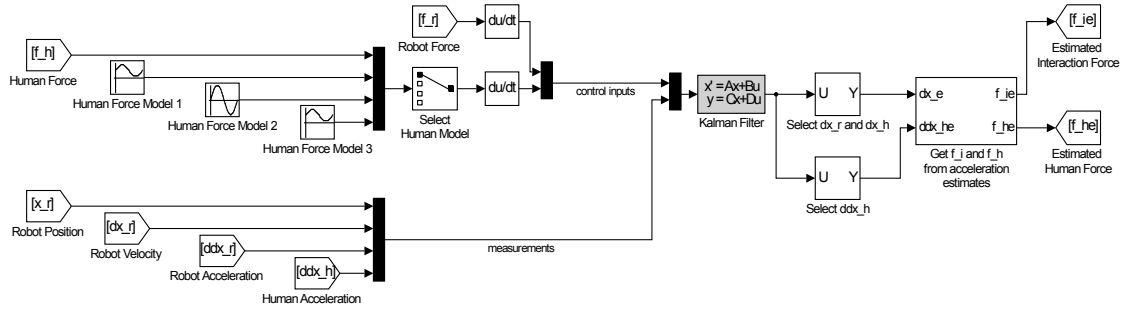


Figure 3.3: Kalman filter model. The Kalman filter itself is generated with the "kalman" command in MATLAB and imported into Simulink as state space matrices. The robot trajectory and human acceleration are fed into the filter as measurements. The robot force and a human force model are used as system inputs. Different human force models can be selected. After the Kalman filter, the estimates for the interaction force and the human force are calculated from the Kalman filter state estimates.

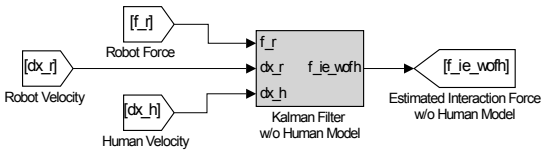


Figure 3.4: Kalman filter model without a human force input. This filter generally works the same way as the one in Figure 3.3 but with different system matrices which do not use a human force model to estimate the states.

3.2 Simulation Results

3.2.1 Kalman Filter Performance

We began our simulation with the investigation of the Kalman filter performance. We tested the influence of different separate factors which could possibly decrease force estimation accuracy. For this we switched off all controllers, noise and delays and examined the resulting interaction force estimate produced by the Kalman filter, depending on different settings.
We started out by investigating the influence of different human force models on Kalman filter performance. To get detailed information on the influence of the human model quality we systematically changed our model in terms of offset, amplification etc. and simulated them one after the other. First, we analyzed the influence of a human force model offset. The results can be seen in Figure 3.5. Next, we performed the same experiment for an amplitude change, a delay and a frequency change in the human force model as pictured in Figures 3.6 to 3.8. Lastly, we investigated the influence of human force model complexity on the Kalman filter estimate. The results can be seen in Figure 3.9.
After this analysis, we investigated the influence of delays on Kalman filter performance. We performed this test for both the case of a damped and a pure spring attachment to see if the incorporation of a damper, which uses the velocity term, changes the influence of delays in the separate measurements. The results of this simulation are shown in Figures 3.10 and 3.11.
Figures 3.12 to 3.17 show the influence of an unknown spring stiffness on the Kalman filter estimation accuracy. We tried this for both variations in the linear spring stiffness and nonlinear springs while still assuming a linear spring for estimation.
The final two simulation experiments investigated uncertainties in the human and robot masses as seen in Figures 3.18 and 3.19 and an unknown attachment damping, shown in Figure 3.20.

3.2.2 Controller Performance

After having evaluated the performance of the Kalman filter, we now needed to know which parameter most influences controller performance. Thus, we conducted similar simulation experiments for the controllers as we had done for the Kalman filter before. For this, we first used the real interaction force for the force controller to avoid our controller evaluation being influenced by estimation errors. We also switched all non-ideal behaviour, e.g. noise, off at the start to investigate the parameters separately. All controllers were retuned to similar dynamics where possible to receive a comparable result. The tuning goals were the following.

- phase margin 70°
- settling time < 0.5s
- rise time < 0.1s
- overshoot $< 15\%$

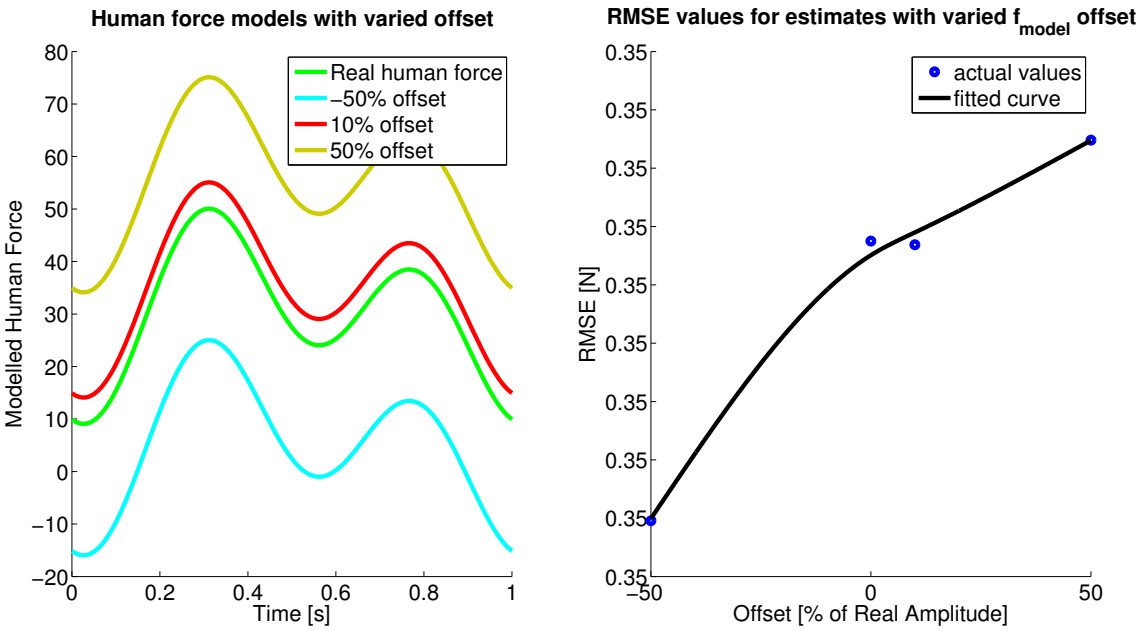


Figure 3.5: Kalman filter performance with an offset human model. Left: Estimated interaction force. Right: RMSE of the estimate. There are hardly any changes in the estimation error, which stays around 0.35N.

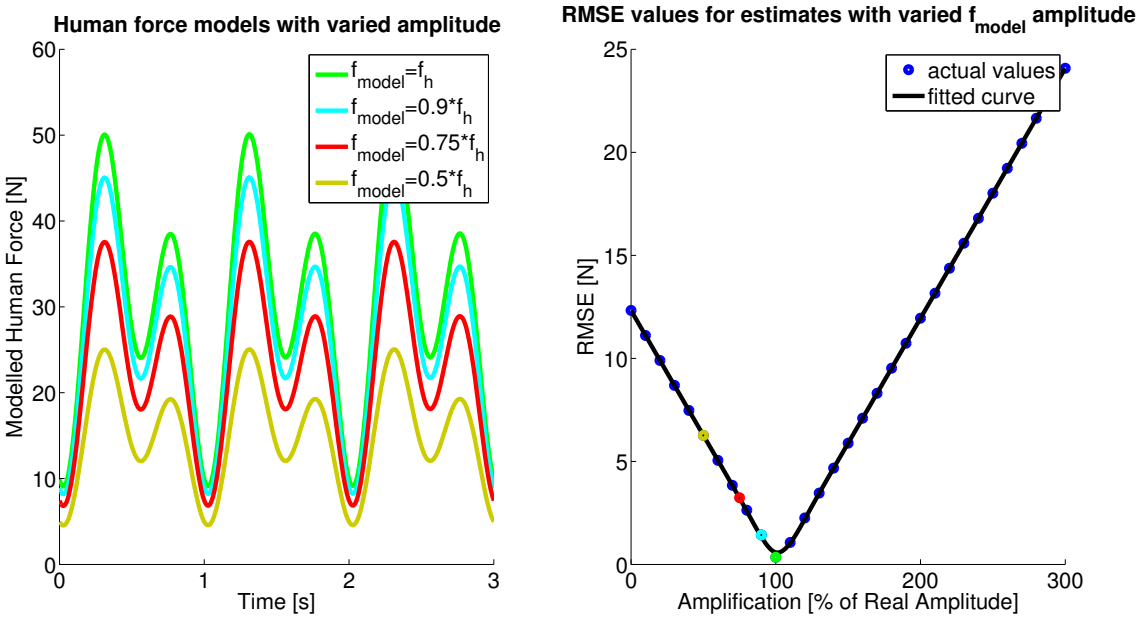


Figure 3.6: Kalman filter performance with a varied human model amplitude. Left: Human models used for estimation. Real human force and models with 90%, 75% and 50% of the real amplitude. Right: RMSE of the estimate for 10% increments in amplification. The estimation error varies linearly with the human force model amplitude in both de- and increase.

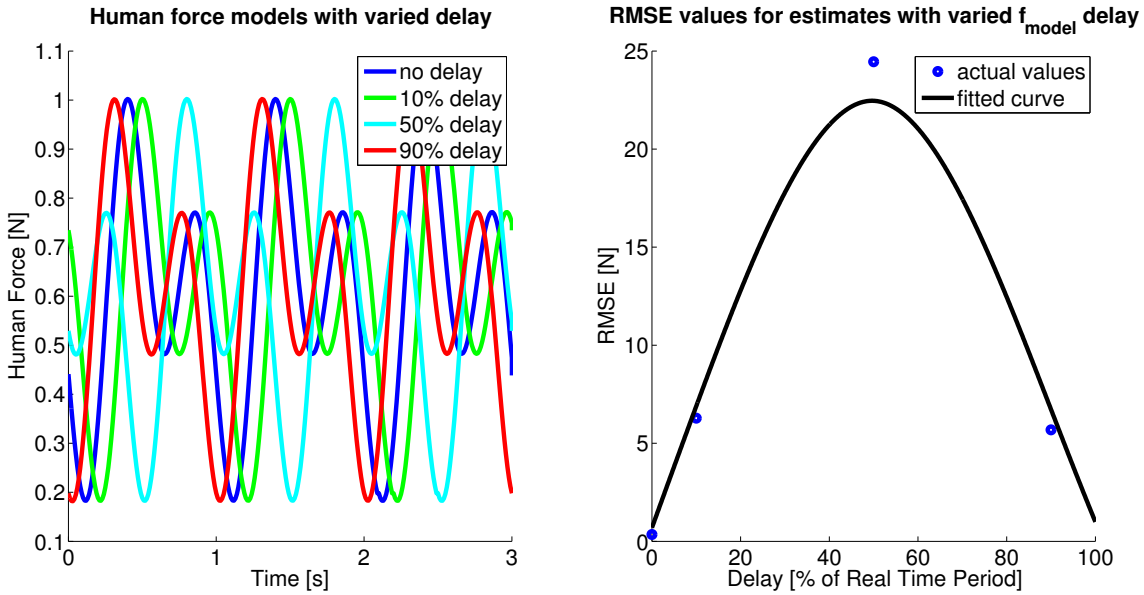


Figure 3.7: Kalman filter performance with a varied human model delay. Left: Human models used for estimation. Real human force and models with delays of 10%, 50% and 90% of the real force period. Right: RMSE of the estimate for the models shown on the left. The estimation error rises up to around 25N for a delay of 50% of the period, then it decreases symmetrically to the increase.

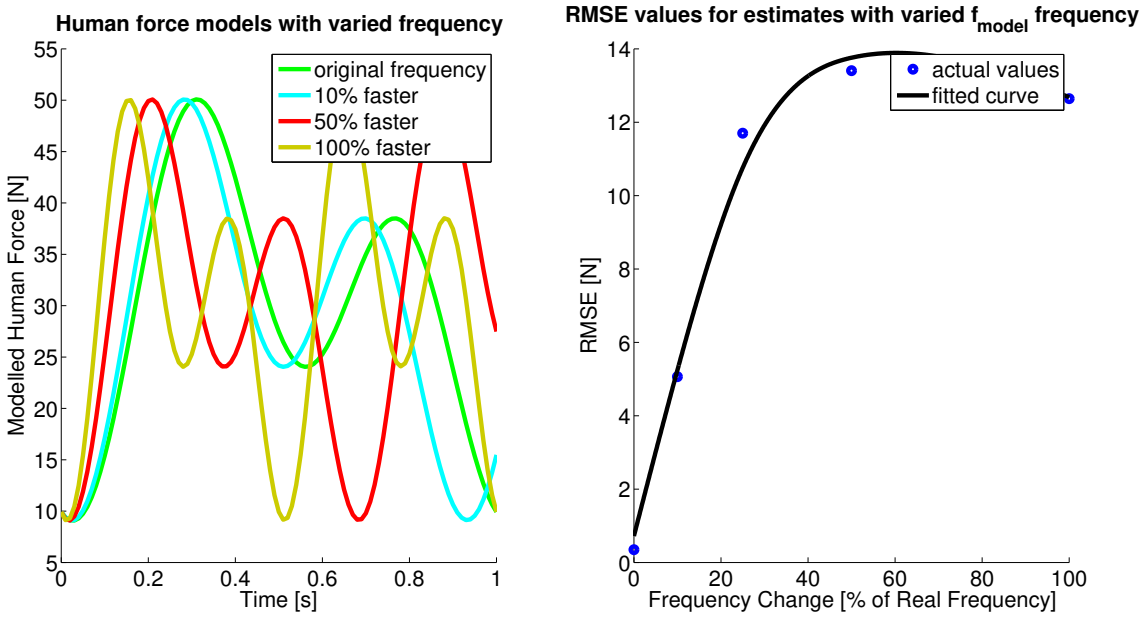


Figure 3.8: Kalman filter performance with a varied human model frequency. Left: Human models used for estimation. Real human force and models with 10%, 50% and 100% increase in frequency. Right: RMSE of the estimate for selected frequency increments in the human force model. The RMSE rises quickly for small frequency increments but then flattens towards a doubling of the frequency.

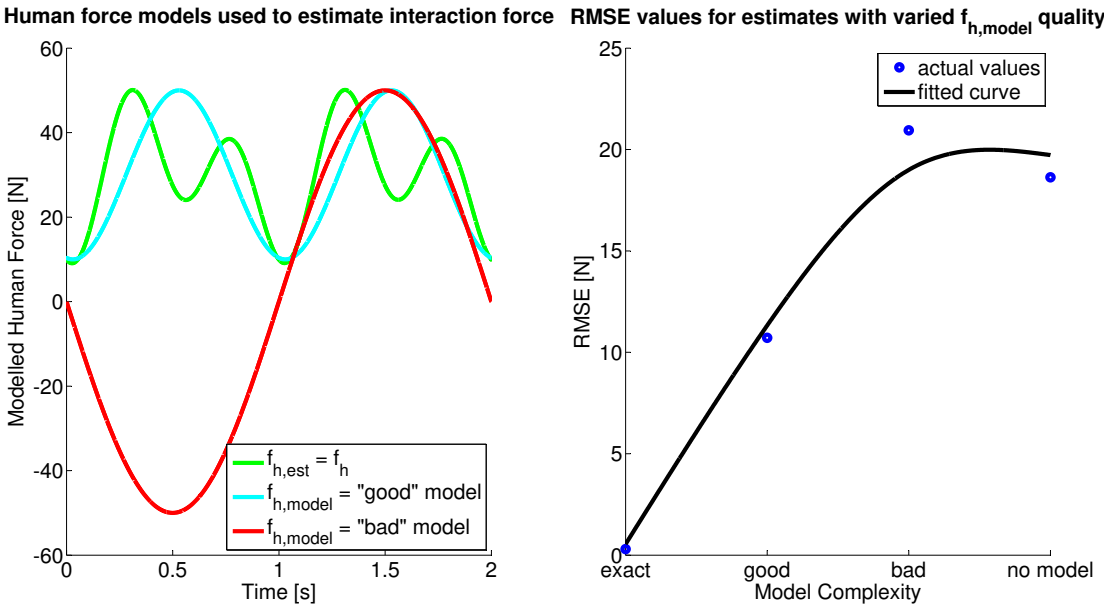


Figure 3.9: Kalman filter performance with a varied human model complexity. Left: Human models used for estimation. Real human force, one model closely, one only partly following the real force. Right: RMSE of the estimate for the human force models on the left as well as a separate Kalman filter without any human force model. The estimation error decreases with model complexity. An overly simplified model even performs worse than no model at all.

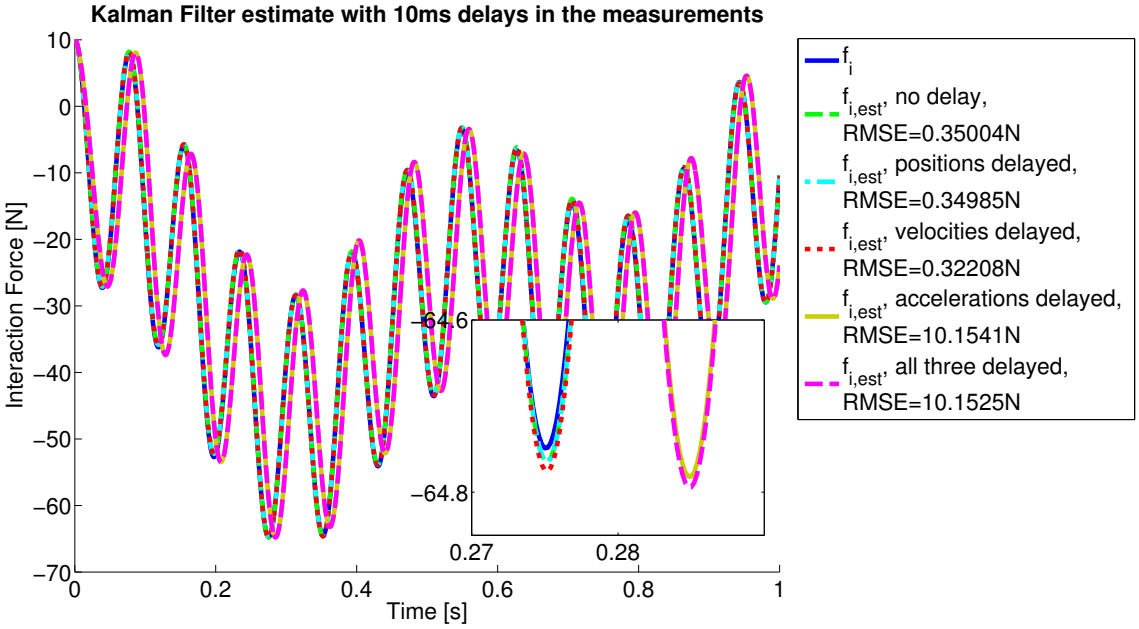


Figure 3.10: Kalman filter performance with delayed measurements. First the measurements were delayed one after the other, then all of them at once. Position and velocity measurement delays show no significant effect on estimation quality. Acceleration measurement delay shifts the whole estimate, increasing the RMSE to around 10N.

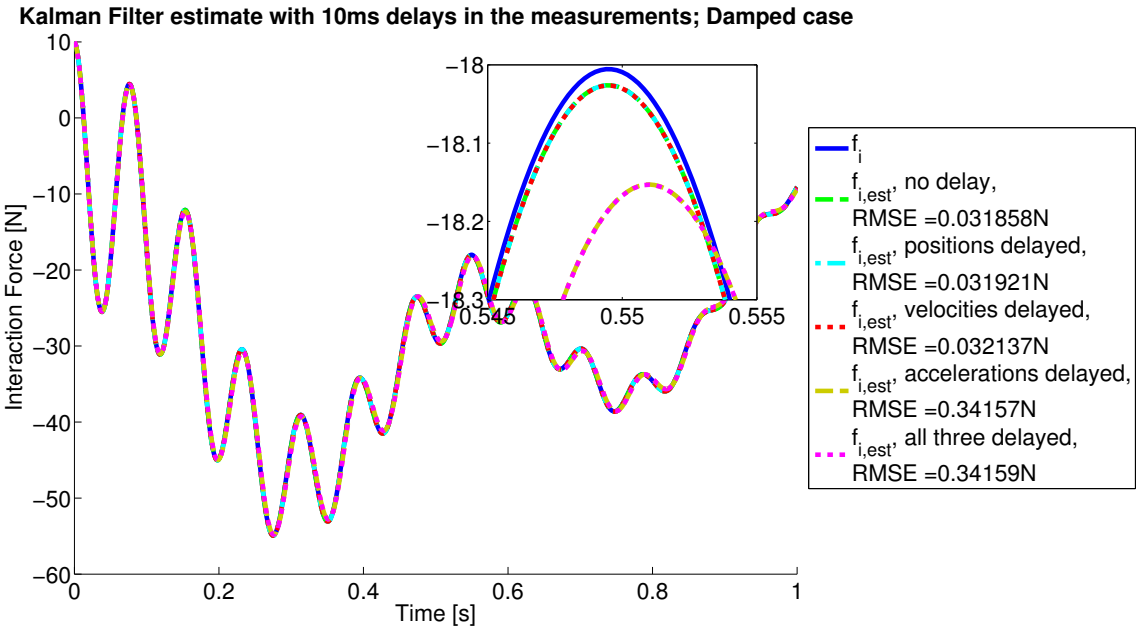


Figure 3.11: Kalman filter performance with delayed measurements and a damped attachment. First the measurements were delayed one after the other, then all of them at once. Again, position and velocity measurement delays show no significant effect on estimation quality. Acceleration measurement delay shifts the estimate but only raises the RMSE to about 0.34N as opposed to 10N in the undamped case.

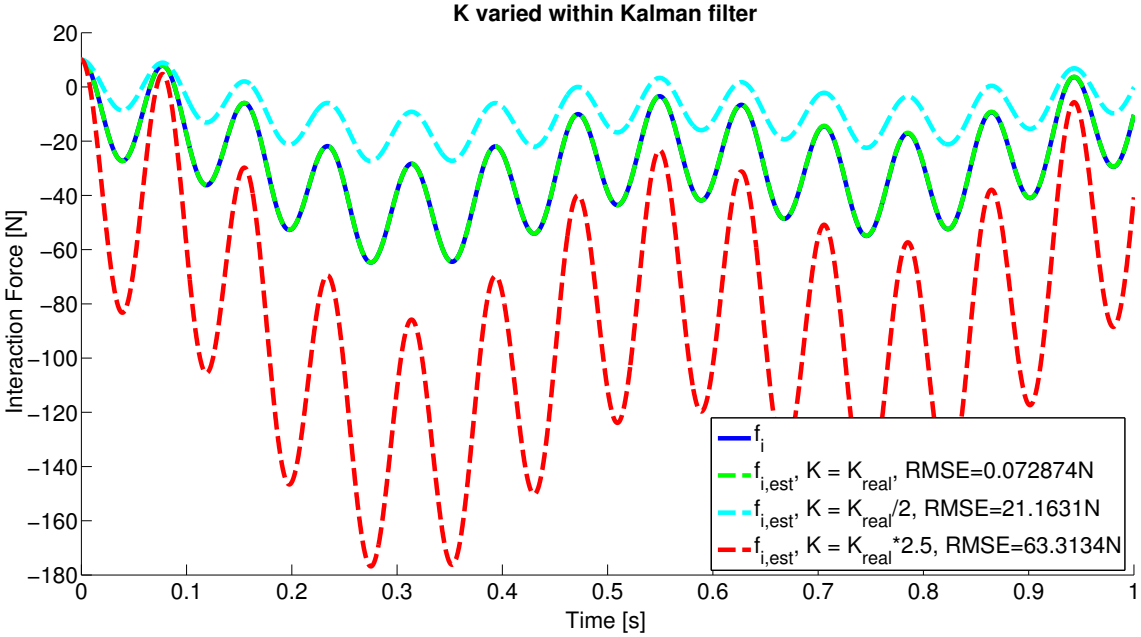


Figure 3.12: Kalman filter performance with unknown spring stiffness. The attachment spring stiffness was held constant while the spring stiffness used for estimation in the Kalman filter was varied by factors of 0.5 and 2.5. The estimate amplitude varies with the modelled spring stiffness. The higher the variation, the extremer the amplification.

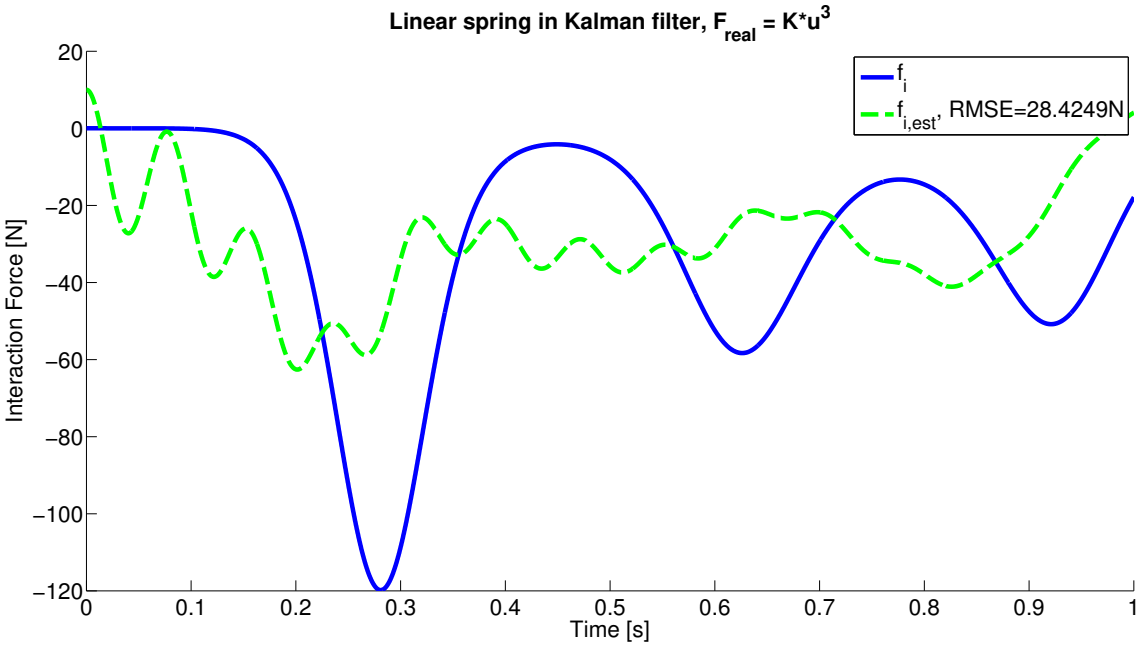


Figure 3.13: Kalman filter performance with unknown spring stiffness. The attachment spring was changed to an exponential nonlinear behaviour with $f_i = Ku^3$ while the spring model in the Kalman filter stayed linear with $f_i = Ku$. The linear Kalman filter model is unable to account for the nonlinear spring behaviour, which becomes especially apparent when the force changes rapidly.

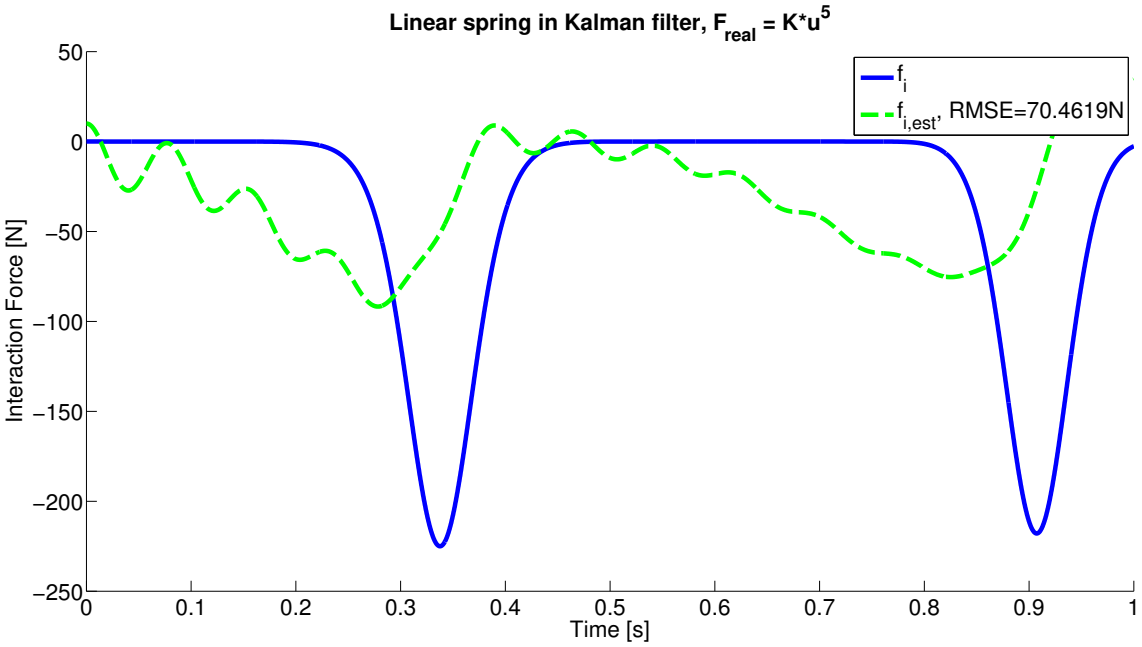


Figure 3.14: Kalman filter performance with unknown spring stiffness. The attachment spring was changed to a higher order exponential nonlinear behaviour with $f_i = Ku^5$ while the spring model in the Kalman filter stayed linear with $f_i = Ku$. Again, the Kalman filter is unable to estimate the nonlinear behaviour correctly.

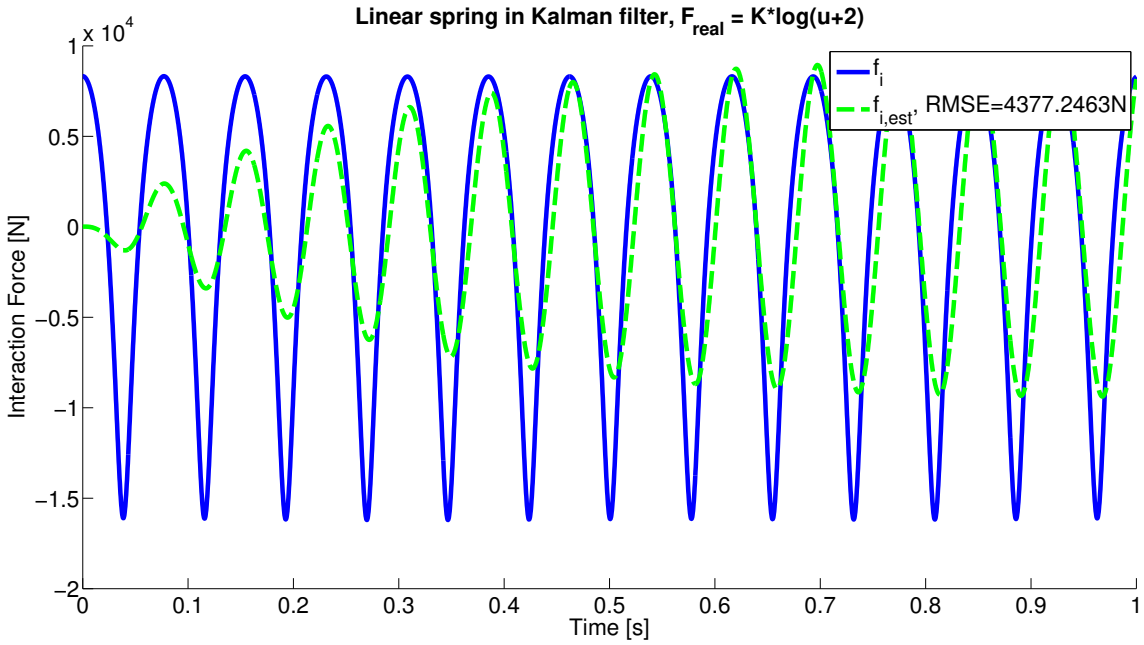


Figure 3.15: Kalman filter performance with unknown spring stiffness.The attachment spring was changed to a logarithmic nonlinear behaviour with $f_i = Klog(u+2)$ while the spring model in the Kalman filter stayed linear with $f_i = Ku$. In this case the Kalman filter can estimate the general periodic motion but can never reach the high amplitudes of the nonlinear spring, resulting in the highest RMSE of all tested cases.

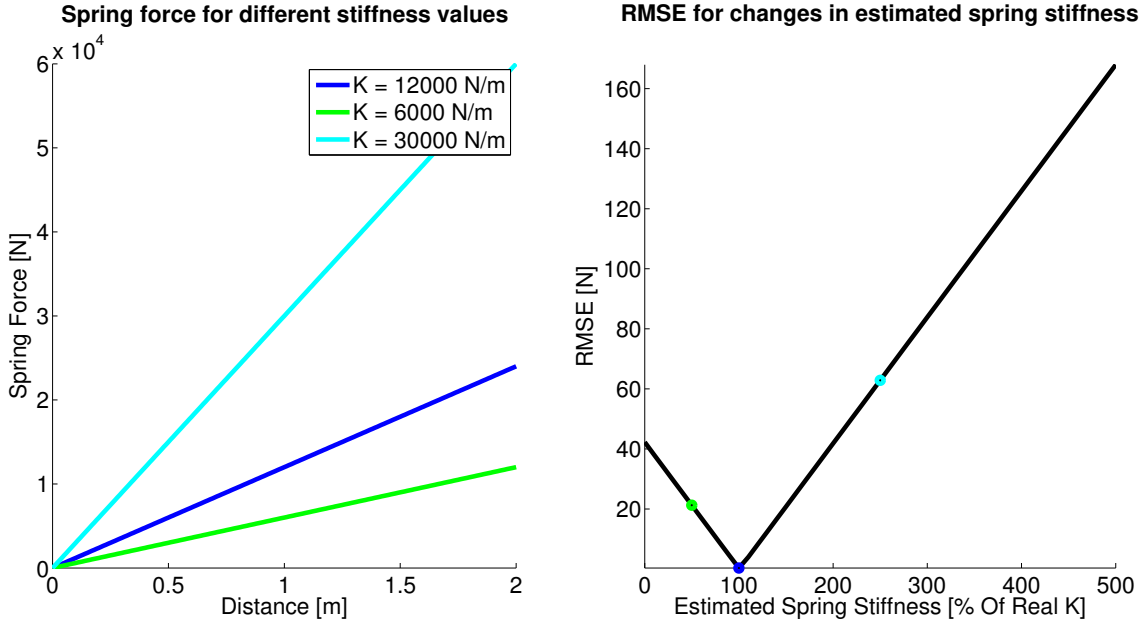


Figure 3.16: Kalman filter performance with unknown spring stiffness as in Figure 3.12. Left: Spring force for varied spring stiffness values, where the green line corresponds to the real spring stiffness. Right: RMSE of the corresponding estimated interaction force for 1% increments in the modelled linear spring stiffness. The estimation error varies linearly with the change in linear spring stiffness.

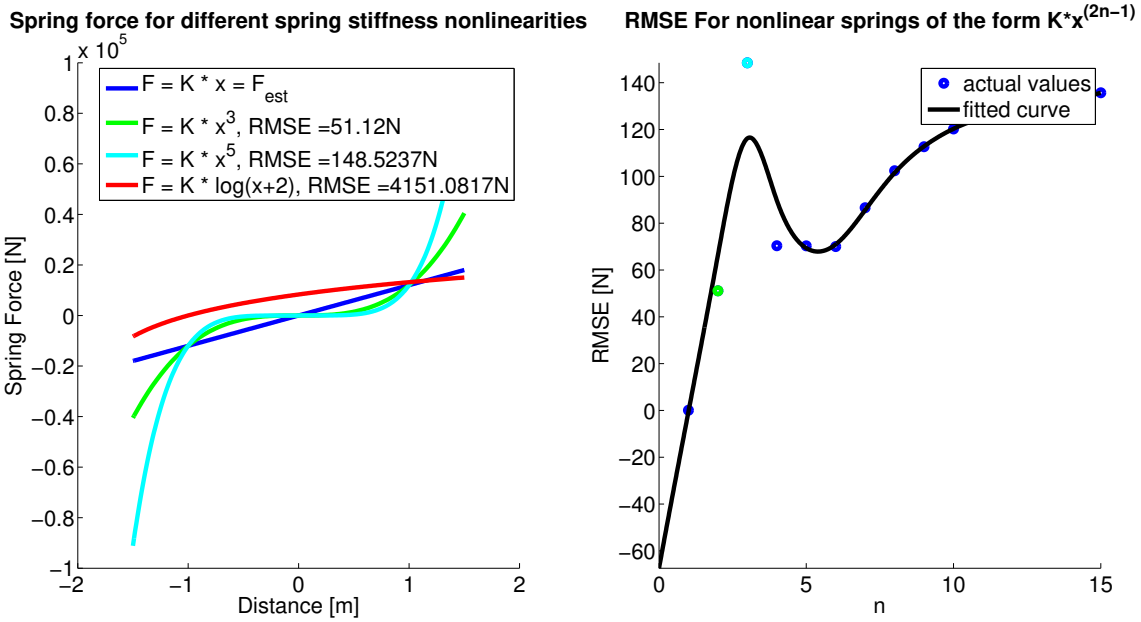


Figure 3.17: Kalman filter performance with unknown spring stiffness as seen in Figures 3.13 to 3.15. Left: Spring force for varied nonlinear springs, where the blue line corresponds to the spring used for estimation. Right: RMSE of the estimated interaction force for springs of the form $Kx^{(2n-1)}$ for integer increments of n. The RMSE increases rapidly for smaller exponents but then seems to flatten out around 120N.

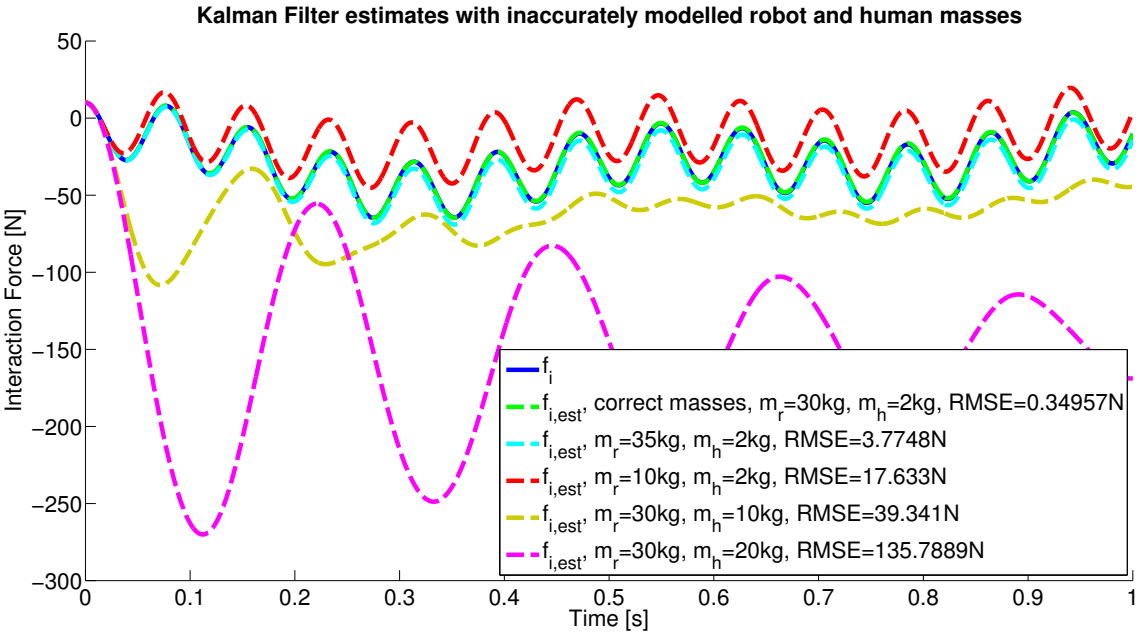


Figure 3.18: Kalman filter performance for unknown masses. Both human and robot masses were varied separately. For small variations in both masses, the RMSE is negligible. If the uncertainties become large, the estimation error rises rapidly.

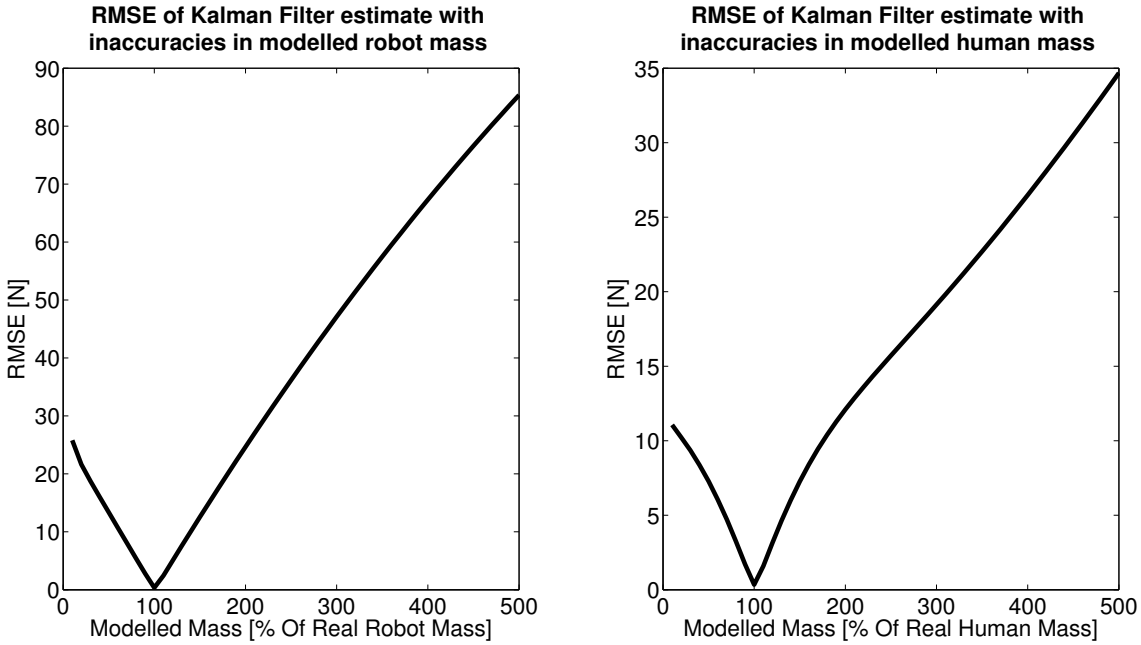


Figure 3.19: Kalman filter performance for unknown masses. Both human and robot masses were varied separately in 1% increments. Left: Estimation RMSE for varied robot mass. Right: Estimation RMSE for varied human mass. The estimation error rises almost linearly for variations in both human and robot mass.

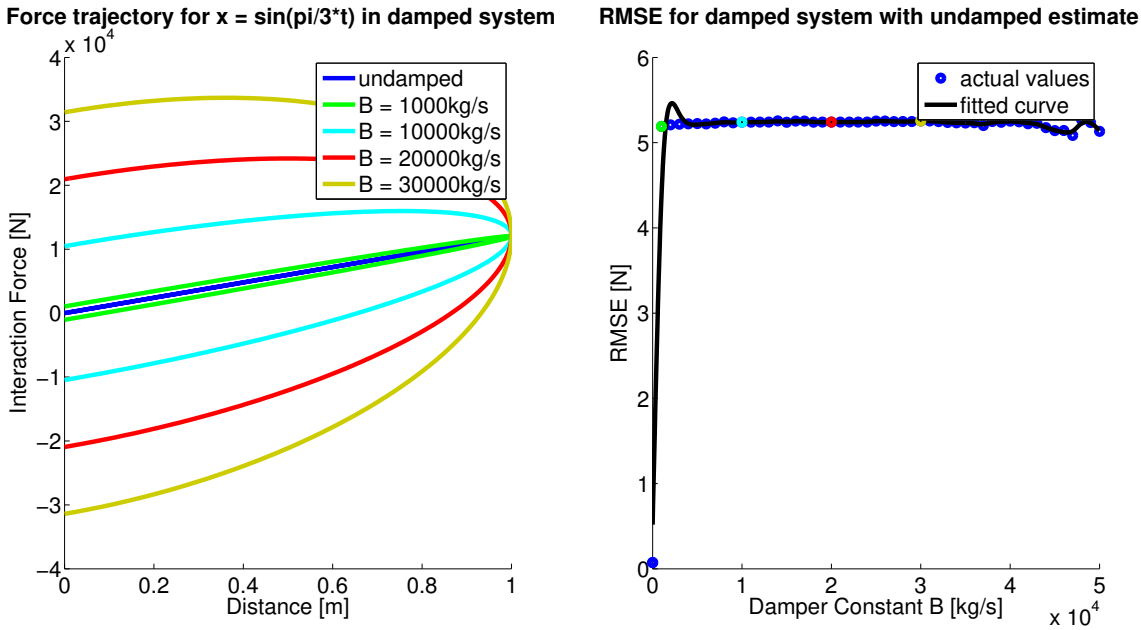


Figure 3.20: Kalman filter performance with unknown damping in the attachment. Left: Force trajectory for a sinusoidal position change. The blue line corresponds to the attachment used for estimation. Right: Estimation RMSE for increasingly damped attachment with undamped estimate. The RMSE rises quickly as soon as damping is present but then stays almost constant around 5N independent of the damper strength.

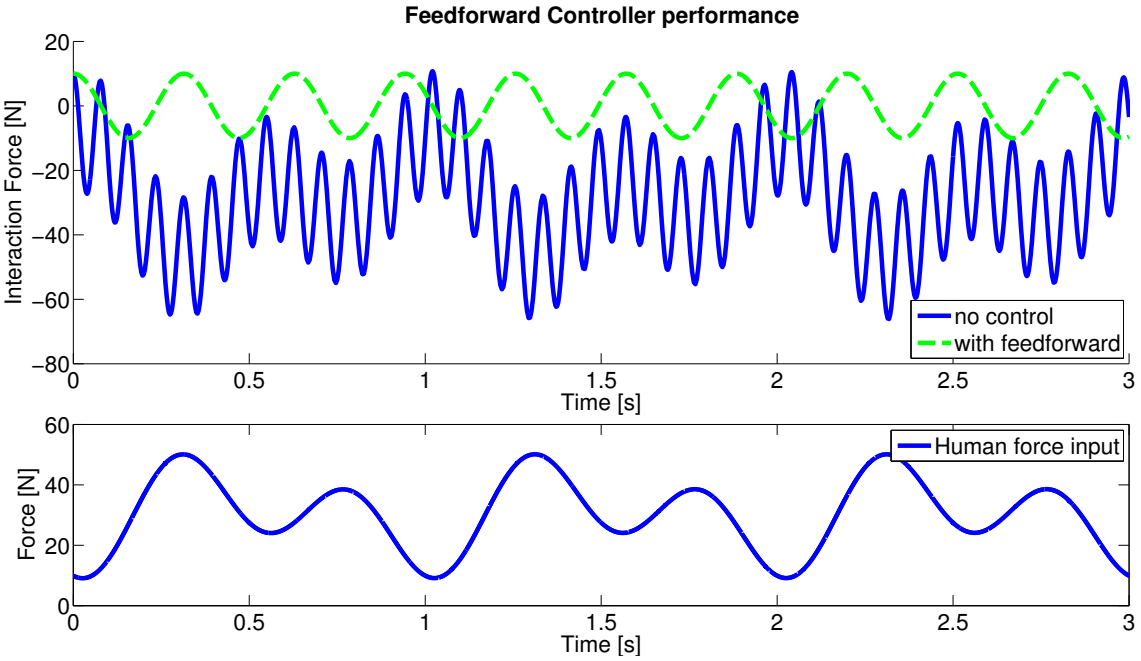


Figure 3.21: Ideal case feedforward controller performance. Bottom: Human force input to the system. Top: System response without any control and pure feedforward in terms of interaction force. The feedforward controller removes the slow oscillations caused by the human input force but retains the fast oscillations from spring deflection.

To start our experiment, we examined the controllers separately. First, we switched off both feedback controllers and evaluated the feedforward controller in the ideal case. The results can be seen in Figure 3.21.
Next, we investigated the influence of non-ideal parts. We started with the actuator, which we modelled with a second order dynamics term to emulate actuator dynamics as well as force saturation as seen in Figures 3.22 and 3.23 respectively. In Figure 3.24 we repeated the saturation experiment but with equal masses and fast force pulses to show that both large mass differences and fast changes in force lead to actuator saturation issues.
For the next two experiments we looked at non-ideal sensors which produce noisy or delayed measurements. The results from this can be seen in Figures 3.25 and 3.26.
Finally, we tried to emulate the most realistic case that we could produce in simulation. We used the Kalman filter estimate for force control and switched all previously discussed issues on. After retuning the Kalman filter and controllers to adjust for this non-ideal case, we obtained the results shown in Figure 3.27.

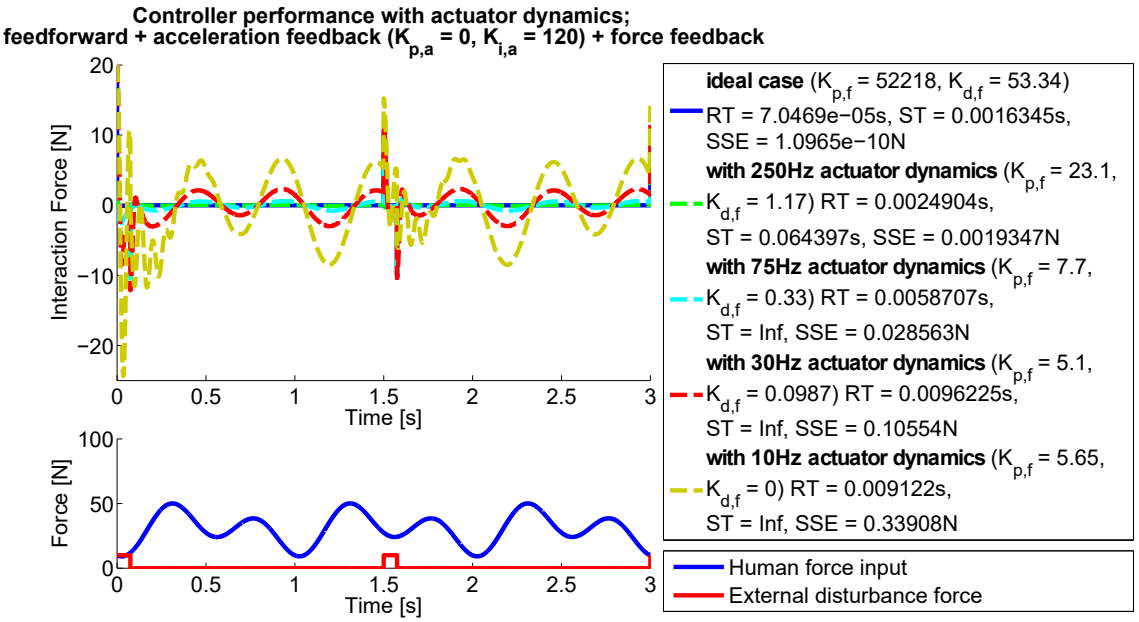


Figure 3.22: Controller performance depending on actuator dynamics. Bottom: Human force input to the system and external disturbance to that force. Top: System response for different actuator bandwidths. Listed in the legend are the respective controller parameters as well as rise time (RT), settling time (ST) and steady-state error (SSE). A fast actuator controls the system almost perfectly. Actuator bandwidths under 75Hz start to lead to some oscillation at the frequency of the human movement.

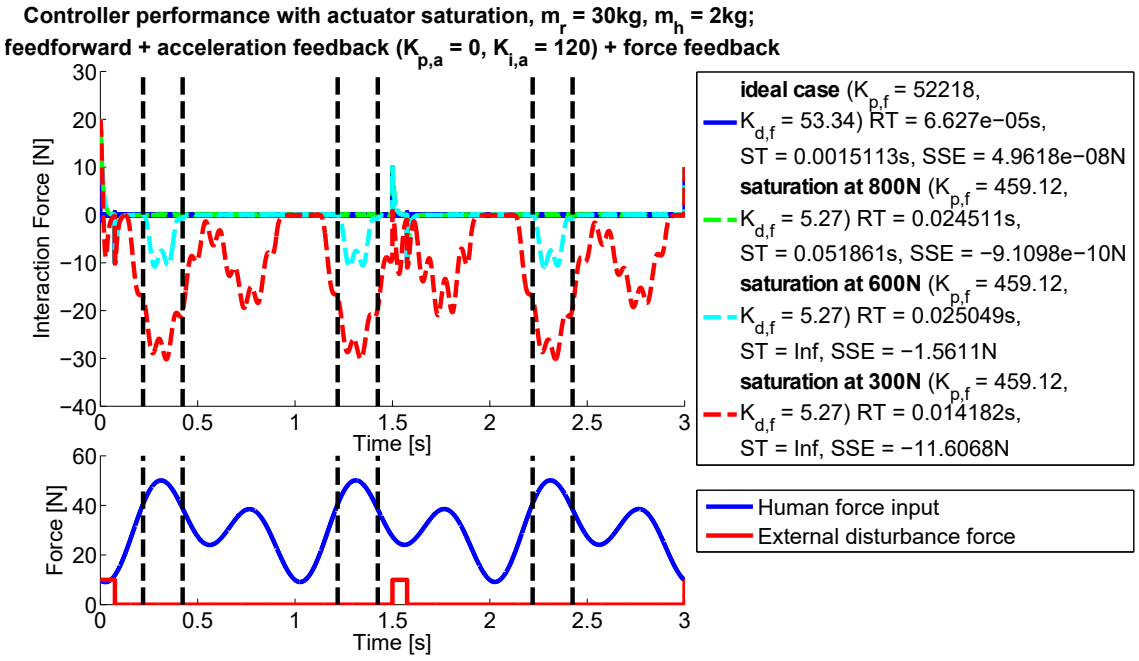


Figure 3.23: Controller performance depending on actuator saturation. Bottom: Human force input to the system and external disturbance to that force. Top: System response for differently saturated actuators. Listed in the legend are the respective controller parameters as well as rise time, settling time and steady-state error. The black lines show the entering and leaving of 600N saturation. Depending on the saturation force, the actuator is unable to counteract the human force at either its higher or both peaks.

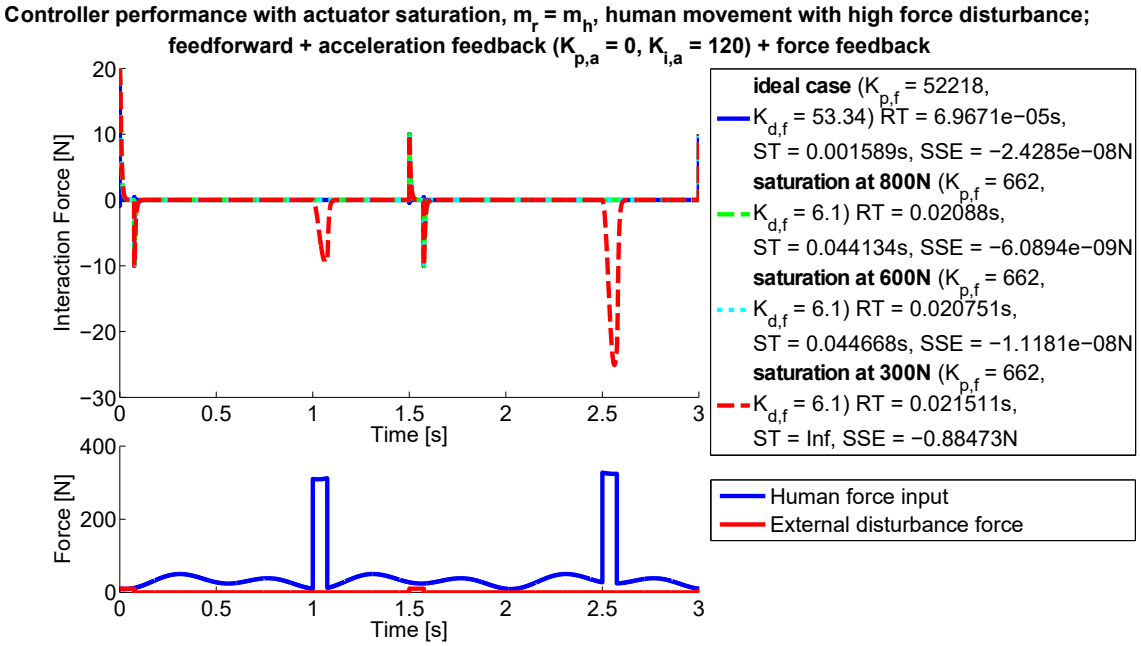


Figure 3.24: Controller performance depending on actuator saturation with fast human force pulses. Bottom: Human force input to the system and external disturbance to that force. Top: System response for differently saturated actuators. Listed in the legend are the respective controller parameters as well as rise time, settling time and steady-state error. With the masses equal, only very fast changes in force lead to actuator saturation. Apart from the lowest saturated actuator, behaviour is almost perfect.

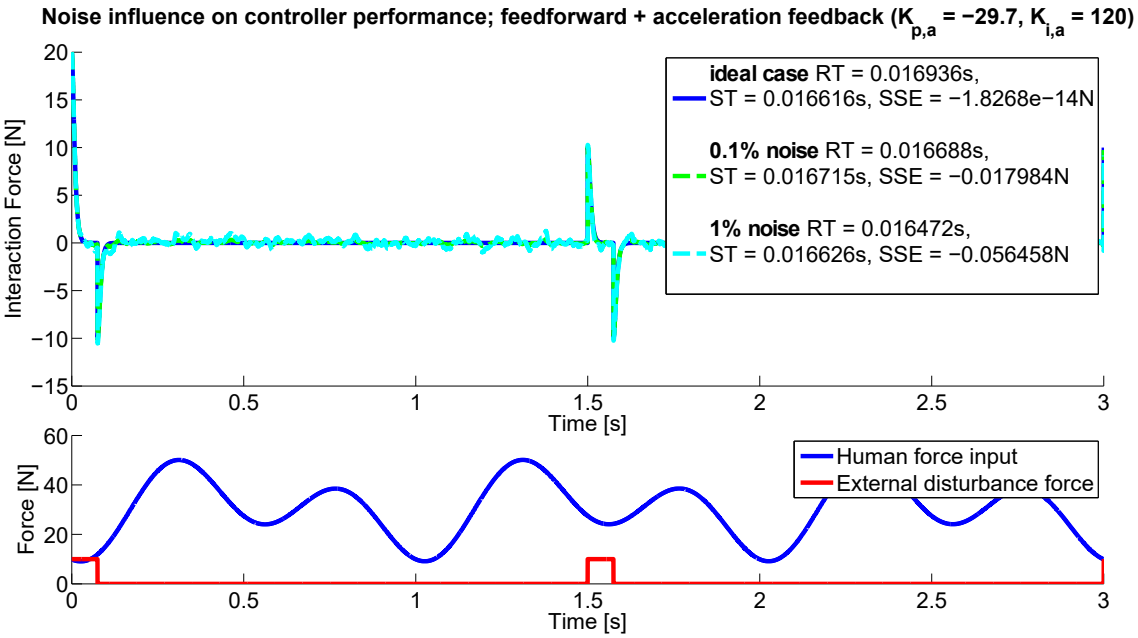


Figure 3.25: Controller performance depending on measurement noise. Bottom: Human force input to the system and external disturbance to that force. Top: System response for different amounts of measurement noise. Listed in the legend are the respective controller parameters as well as rise time, settling time and steady-state error. Noise was added onto both human and robot acceleration measurements. The controller performance is hardly being influenced by adding noise into the system.

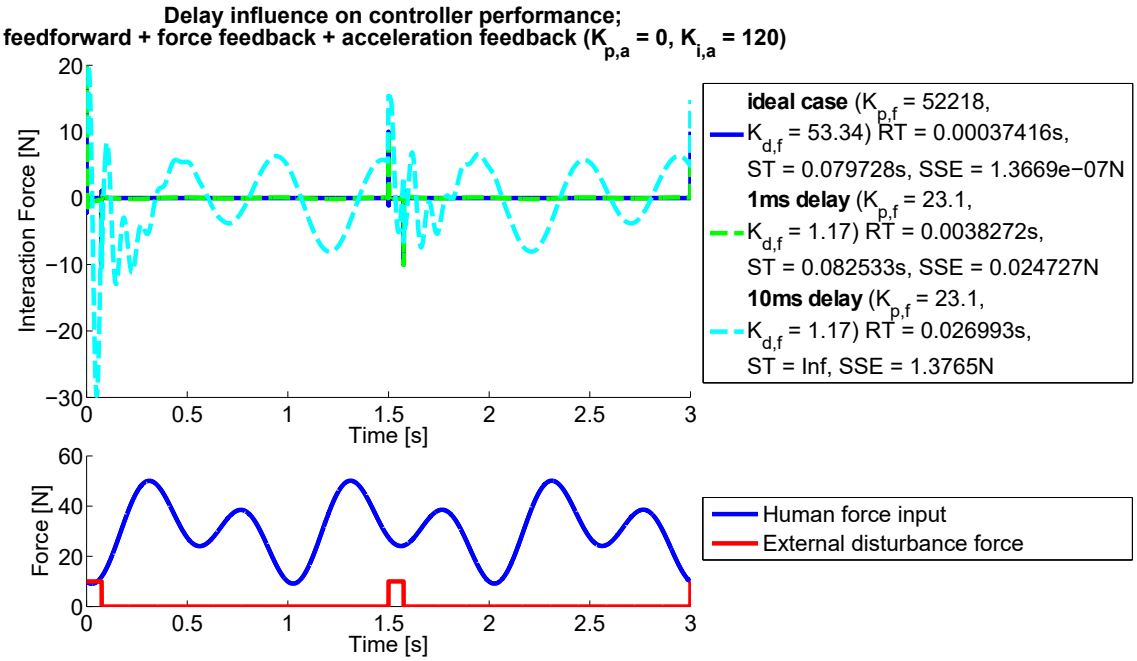


Figure 3.26: Controller performance depending on measurement delay. Bottom: Human force input to the system and external disturbance to that force. Top: System response for different measurement delays. Listed in the legend are the respective controller parameters as well as rise time, settling time and steady-state error. For simplicity reasons, all measurements were delayed by the same amount. Up to a 1ms delay, controller performance is similar to the ideal case. At 10ms delay, significant oscillations arise, similar to the actuator dynamics case.

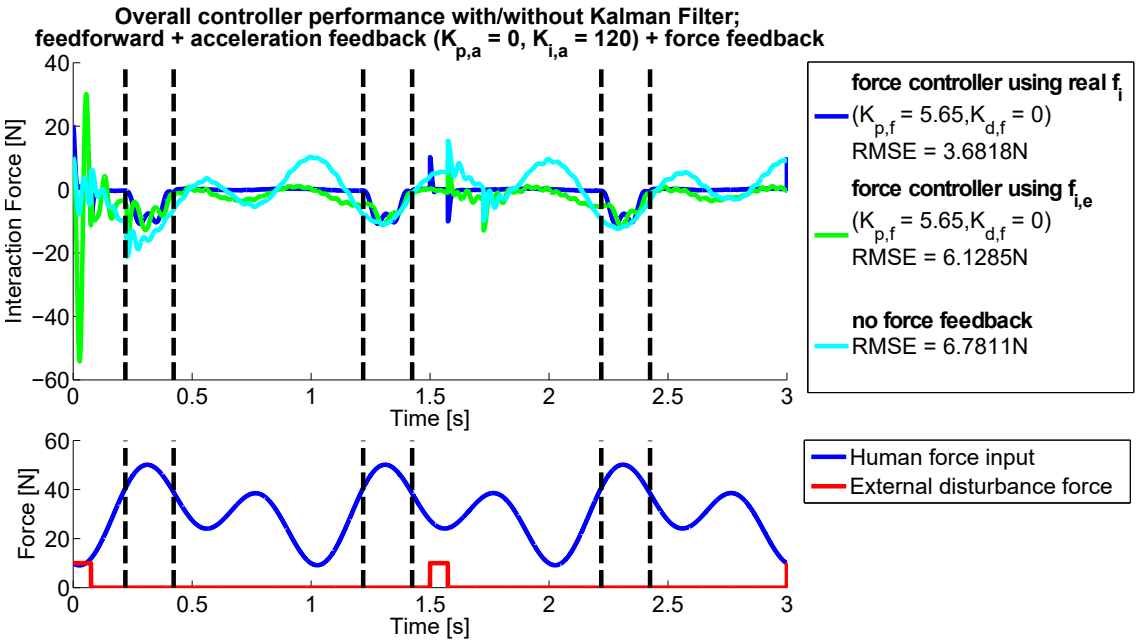


Figure 3.27: Controller performance depending on use of Kalman filter or measuring interaction force. Bottom: Human force input to the system and external disturbance to that force. Top: System response with actuator dynamics, saturation and combinations of using the Kalman filter estimate for control vs. not using any force control at all. Listed in the legend are the respective controller parameters as well as the RMSE of the controlled interaction force. The black lines show the entering and leaving of 600N saturation. The Kalman filter causes high frequency and amplitude oscillations whenever quick changes in input force arise, as seen at the initial decrease. Apart from some minor oscillations, the controller works well if no fast changes in force arise. Compared to this case, the pure acceleration feedback/feedforward controller seems to oscillate significantly more in the long term. The RMSE comparison hardly shows any difference between the latter two controllers since the high amplitude and the long term oscillations of the Kalman filtered and the pure acceleration controller respectively are similar in overall error sum.

3.3 Discussion

The Kalman filter performance was evaluated in terms of the estimation RMSE as indicated in the plots in Section 3.2.1.
The first aspect we investigated was the influence of different human force models on the Kalman filter estimate. This was necessary because, ideally, we want to run the whole system without any information on the human force input whatsoever. To be able to compare this case to the one where information on the human force is available, we need to know how different human models influence the quality for our estimate.
We started out by applying an offset to the human force model. This did not change the estimation results in a significant way. This might be due to the fact that the offset of the Kalman filter estimate is more influenced by the initial conditions than the measurement or control inputs. Therefore, with the right initial conditions, the offset influence becomes negligible.
As expected, the influence of human force model amplification rises linearly with the amplification value as seen in Figure 3.6.
Similarly, the estimate for human model delay behaves exactly as expected. Up until a delay of 50% of the period, i.e. the maximum shift with respect to the real input, the error increases, then it decreases until, at 100% real and modelled input are the same again.
For a change in frequency of up to 50%, the error increases just like in the delayed case but then stays approximately the same instead of falling again. This is because, as opposed to the delay case, a doubled frequency does not mean that the signal returns to its original shape and value. However, since the signals are periodic and have common points whenever their frequencies are an integer multiple of each other, they become more similar and therefore the error does not rise any more.
As a last point on human model evaluation we investigated the influence of model complexity on the estimation result. Figure 3.9 shows the estimation results for human models of different complexity. One models the real force closely, one uses a significantly lower frequency. Lastly we also included the case where there is no human model at all. We can see that, if we have a good human model, which models the higher frequencies of the system, it is an advantage to use it for estimation. However, if we cannot provide a good human model, it is better to not use it at all, which intuitively makes sense.
Next, we analyzed the influence of measurement delay on the Kalman filter estimate. All measurements were delayed by 10ms, which corresponds to approximately 1% of the input force period. Figures 3.10 and 3.11 show that, in both the undamped and the damped case, acceleration delay is the only measurement influencing the Kalman filter estimate. We believe that this is due to acceleration being most directly related to force and therefore the force estimate being more susceptible to acceleration than position or velocity delays.
The next overall experiment we performed was on uncertain attachment behaviour. Figures 3.12 to 3.15 show the estimate for different uncertainties in the linear spring stiffness as

well as nonlinearities. We see that, while an unknown linear spring can be problematic when the uncertainty is large, the more significant problem lies in unmodelled nonlinearities. This is a major issue since the traditional Kalman filter can only estimate linear behaviour. It is therefore important to make sure that the attachment behaves linearly. Otherwise, a change towards an extended or unscented Kalman filter would become necessary. Figures 3.16 and 3.17 sum up the RMSE values for the tested attachment uncertainties. This shows us that, while the nonlinearities result in a higher initial error, the uncertain linear spring error rises much faster. Therefore we conclude that it is also important to know the spring stiffness as exactly as possible. This high dependency of the estimation quality on the attachment model is probably due to the fact that the spring model is the defining factor of the interaction force dynamics. If we do not model this parameter correctly, the whole of our estimated dynamics will be affected.

The next aspect we investigated were changes in robot and human mass while the Kalman filter retains its nominal parameters. We can see from Figures 3.18 and 3.19 that, as expected, the estimation error changes linearly with mass uncertainties. The robot mass will be known exactly in the exoskeleton. The mass of human limb segments can be estimated reasonably accurately with techniques like described in [9]. With those two facts in mind we can safely assume not to run into any issues due to mass uncertainties, since simulation shows us that mass only becomes a problem when the modelled values significantly differ from the real ones.

Lastly, we investigated an attachment damping which is not modelled in the Kalman filter. Figure 3.20 shows that, while unknown damping leads to an immediate error in the estimate, increasing the damping does not further increase the error. We assume that this happens because the damping is not modelled. Thus, while the sudden existence of a damping where none is modelled will influence the estimate, increasing this damping does not have any further worsening effect on estimation quality.

Figure 3.28 shows a summary of the Kalman filter estimation errors. We can see clearly that the most prominently contributing factor in estimation errors is the modelled spring stiffness. Especially unmodelled spring nonlinearities make the estimation error skyrocket. Uncertainties in robot and human masses can also become an issue to a certain extent, especially when one mass is significantly smaller than the other as was the case in our simulation.

While the other influences are small compared to the previously mentioned ones, they might not be negligible in the real setup. Particularly delays and amplitude changes in the human model, whose estimation error ranges around 20% of the input force might create a significant interaction force when combined. Therefore, while focussing on spring behavior estimation, we also need to keep the remaining issues in mind.

Controller performance was harder to evaluate quantitatively than Kalman filter performance since we specifically tuned the controllers to meet certain performance criteria whenever possible. However, some criteria were harder to meet than others. Especially settling time was an

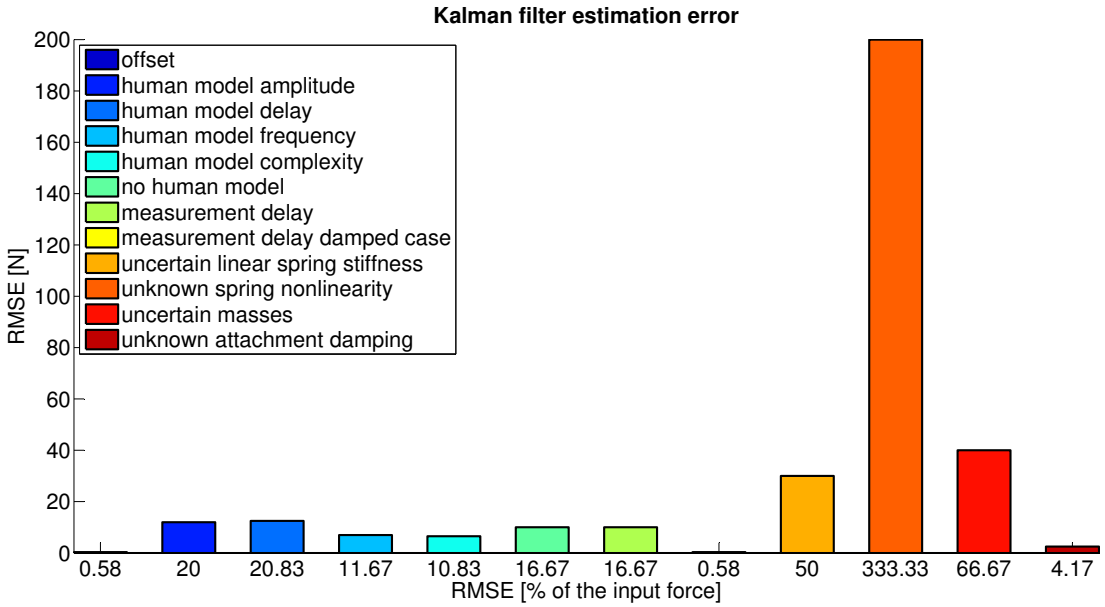


Figure 3.28: RMSE of Kalman filter estimates under different conditions. The y-axis shows the actual error force values, the x-axis shows them as a percentage of the human input force. The values shown are average values for each tested condition. While we show a mean value, the variance was not evaluated since we do not have enough data from different experiments for each case to result in a statistically meaningful statement.

issue for the cases where some oscillation remained in the system. These cases are indicated with an infinite (Inf) settling time in Section 3.2.2.

The first tested case, i.e. pure feedforward control shows us that, while the feedforward command accounts for the average force resulting from a nonzero human input, it cannot compensate for the oscillations due to a nonzero initial force. This is consistent with the notion that a feedforward controller only applies a command according to another input, in our case the human acceleration. It does not take into account any past states of the system and thus cannot correct for the nonzero initial condition. Therefore we know that we need at least one feedback controller to reduce the interaction force to zero for arbitrary initial conditions.

We did not perform separate experiments on the two feedback controllers since we deemed it unnecessary. On one hand we know that we need the acceleration controller because our force estimate and thus our force controller performance will never be ideal. On the other hand we can think of the case where we start our experiment at zero acceleration but a nonzero interaction force. Then the acceleration controller would not be able to reduce this initial force down to zero. Therefore we use both controllers in any case. Figure 3.27 also shows a case where pure acceleration feedback performs worse than the combination with force control, thereby supporting our theory.

The next investigated issue was the influence of internal actuator behaviour on controller performance. While we assume our desired robot force to be directly translated to a real force

in the setup, this real force can be influenced by actuator dynamics or saturation. The results from Figures 3.23, 3.24 and 3.25 show that this can have a significant influence on controller performance. The actuator dynamics plot shows some oscillations, which the controller cannot reduce. This becomes more prominent the closer the actuator bandwidth gets to the human input force frequency. This behaviour is reasonable considering that the actuator has no way of following a movement at a specific frequency if its bandwidth does not allow for movement at that frequency. Even while the frequency is still lower but close to the actuator dynamics, there will be some extent of oscillation since actuator performance will decrease close to its bandwidth limits. Thus, we should always keep our actuators fast enough to be far outside of the human motion bandwidth.
Actuator saturation becomes an issue especially when human and robot mass are very different, as seen in Figure 3.24. The same goes for high acceleration movements as seen in Figure 3.25. In both of those cases a small input force results in a large desired robot force, which may not be produced by the actuator. Therefore we should keep the desired motions and moved masses in mind when choosing the actuators to ensure that saturation will not be reached even with a large robot mass or fast movements.
Concerning our simulation results, noise does not seem to have a significant influence on controller performance. This is mostly due to the fact that we have low noise acceleration sensors. This experiment was conducted without the force controller since it would either change the result to the positive or negative depending on the usage of the Kalman filter estimate or the perfect interaction force value for force control.
Measurement delay however seems to influence controller performance quite significantly. The results are similar to the actuator dynamics results for lower frequencies. This might have to do with the fact that, in both cases, the controller cannot follow the human force as fast as necessary - in one case because it is inherently too slow, in the other case because the information it gets arrives too late to be used in the corresponding time step. Both cases lead to high amplitude oscillations which cannot be reduced by retuning.
Lastly, the influence of the Kalman filter was investigated. This was done including some of the previously tested issues, like actuator dynamics and saturation to generate a slightly more complicated signal for the Kalman filter to estimate. We see that the Kalman filter seems to have difficulties with estimation at high frequencies. This leads to high frequency oscillations at every disturbance peak. This makes sense since the Kalman filter is a linear estimator and there are nonlinear terms like actuator saturation in the system. For small accelerations, the desired forces will be small and thus saturation will not be an issue. In that case the Kalman filter will return a good estimate. As soon as accelerations become higher, the system behaviour differs significantly from a linear one and the estimation quality decreases. Other than that, the estimate stays reasonably close to the ideal case. For comparison we repeated this experiment without any force feedback, which showed us that, even with a non-ideal interaction force estimate our controller performs better with both feedback controllers than with acceleration feedback only. This is exactly what we expected.

Overall it can be said that we can expect good controller performance when we do not run into any issues with actuator saturation, actuator dynamics and delays, which is all feasible by choosing the correct hardware.

4 Mechanical Setup

4.1 Original Setup

This section will describe the general mechanical concept and wiring of the setup. For details on the electronics, refer to Chapter 5.
Our test setup for the linear model described in 2.1 is the FC2D (force control in two degrees) setup, which can be seen in Figure 4.1.S The system consists of the following components,

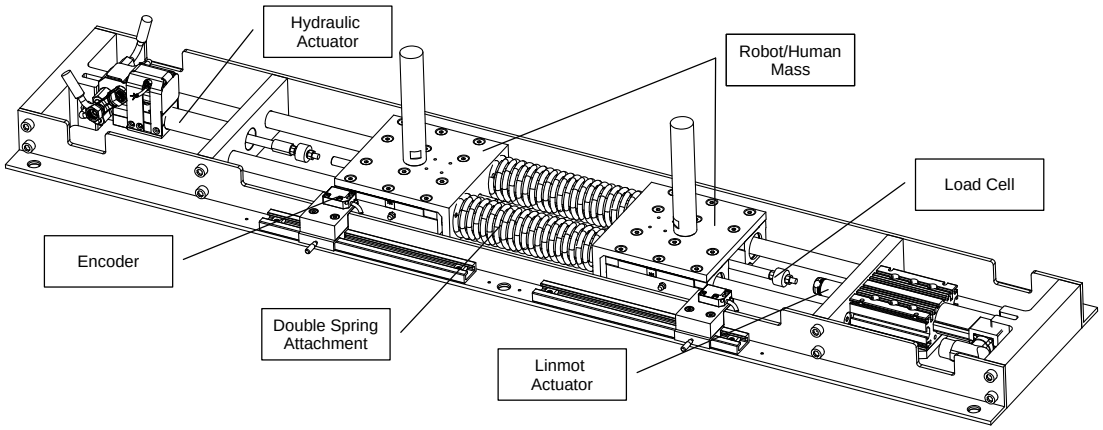


Figure 4.1: FC2D mechanical setup consisting of two masses connected by two springs, two actuators, two load cells, two encoders and stiff connecting parts. Not pictured are the IMUs, which sit on top of the mass carts.

where the range either refers to the measuring or the actuator force range and the output corresponds to the analog sensor output, if applicable. Listed are all of the active and sensing parts. The remainder of the setup consists of Misumi slide bearings, attachments and rods as well as custom parts.

Qty	Manufacturer	Part #	Part	Range	Output
2	Hoerbiger	LB6-1610-0080-4M	hydraulic actuator	$0.1 - 3.2kN$	
2	Moog	E024-177LA	servo valve		
2	Parker	PTD.VB2501B1C1	pressure sensor	$0 - 250bar$	$0 - 5V$
4	RS	797-4986	pressure sensor	$0 - 250bar$	$0 - 5V$
2	Burster	8417-6005-V217	load cell	$0 - 5kN$	$1\frac{mV}{V}$
2	XSENS	MTI-100-2A504	IMU	$0 - 50\frac{m}{s^2}$	serial bus
2	RLS	LM10IC001AB10F00	incremental encoder		serial bus
2	Spiral AG		spring		

The original setup, which was an adapted version of the HyQ robot using the same actuators and load cells, used two hydraulic cylinders for actuation. In experimentation the whole setup turned out to be inadequate for stable and robust force control. While the system was running, noise and delays made it unfit for high performance force reference tracking, especially at low force magnitudes. Therefore we decided to make adjustments to the setup described in the following section.

4.2 LinMot Adjustments

4.2.1 Components and wiring

We decided to incorporate two electric linear motors in the system as a replacement for the hydraulic actuators. The actuators we chose were the LinMot PS01-37x120F-HP-C stators combined with PL01-20x240/180-HP sliders. To power and control the system we also needed a driver and a power supply for each motor. We used the LinMot B1100-GP-HC drive and the S01-72/1000 power supply. Figure 4.2 shows how the components were wired. additionally, we had to adjust some of the mechanical parts to account for the different stator shape, which can be seen in Figure 4.1.
After initial testing we decided to replace the load cell as well, since our new actuators only provide up to 250N of force. With our previously used load cell range of 0-5000N, this put us at the lowest part of our measurement range, resulting in a low signal to noise ratio. We used an Interface SMT1-250N load cell as a replacement, which has a 250N force range and significantly better accuracy. With this load cell we managed to achieve very good force control performance which can be seen in Section 6.2.

4.2.2 Configuring the motors

The configuration of the motors is done in LinMot-Talk, a software which can be downloaded from the LinMot homepage. We used the current version of the software, i. e. LinMot-Talk 6.4. Figure 4.3 shows a screenshot of the main control window. Starting the software for the first time, the first thing to be done is to choose the correct drive via "File $\rightarrow$ Login/Open Offline". This opens the RS-232 connection with the driver and shows the window from Figure 4.3. Next, the correct motor type is chosen. This is done via the motor wizard, which is symbolized by a sparking wand in the toolbar. In this application, stator and slider type, mounting, cables controller parameters and any additional information that can help with motor control can be selected. After finishing this wizard, the motor is ready to operate.
The Control Panel in the main window can be used to operate the motor manually. Since we wanted to control it by analog inputs however, we needed to configure it accordingly.
First, we had to set the Homing Procedure. Homing is necessary for the motor to go into

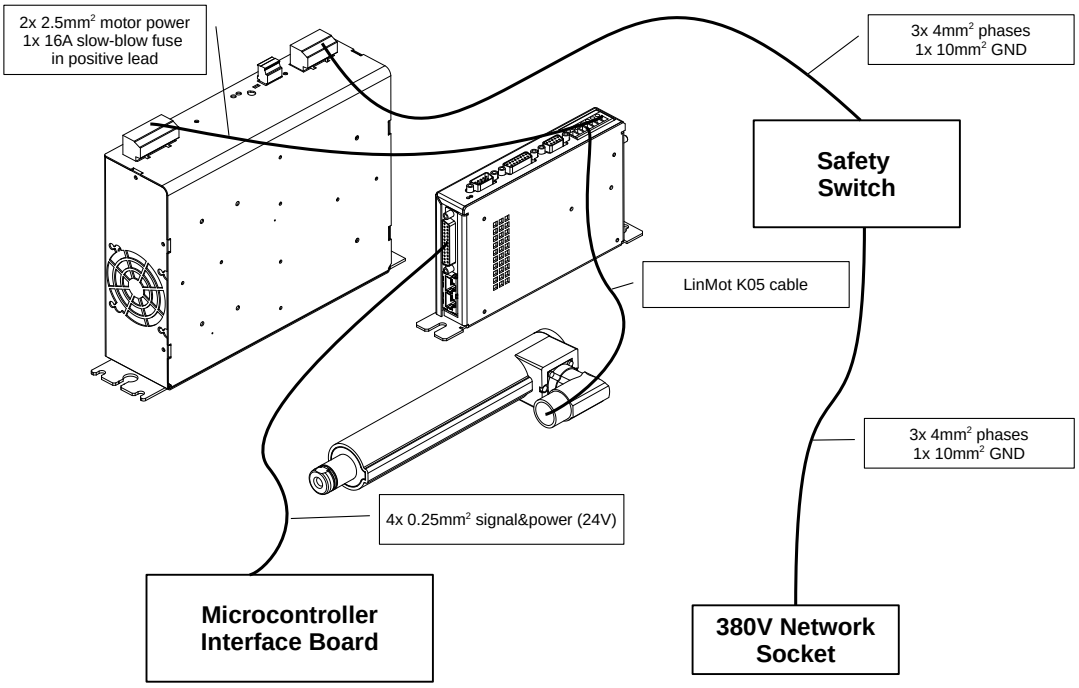


Figure 4.2: Wiring of the LinMot linear motor with its drive, power supply and the rest of the setup.

operational mode since it is position controlled by default and therefore needs to know its start position. We navigated to the "Parameters → Easy Steps → Smart Control Word Behavior" tab and set the "Intf Switch On Flag Behavior" and "Intf Home Flag Behavior" parameters to "Autostart" and "Autohome" respectively. This leads to the motor automatically starting the firmware and then beginning homing as soon as the analog input is powered up with 24V. Since our only hard mechanical stop in the system is the ring on the motor slider we needed to use that as our home position. This position is set via the "Homing Mode" parameter in the motor wizard.

Both of our motors' analog inputs use the same power supply. Thus, they are both powered up and start homing at the same time. Since they are not strong enough to stretch the spring and go back to the mechanical stop on the slider at the same time, we set one actuator to the "Mechanical Stop Negative Search" and one to "Mechanical Stop Positive Search" homing mode. This means that, on power up, one motor will be homed correctly at the mechanical stop while the other one will compress the spring until it reaches the required current to identify a mechanical stop. After this homing procedure, both motors will be operational.

Since the position is also used for counteracting the cogging forces and adjusting for the percentage of the slider outside if the stator, i.e. the slider percentage not reachable by the magnets, we wanted to re-home the second actuator. To this end, we set the "Parameters

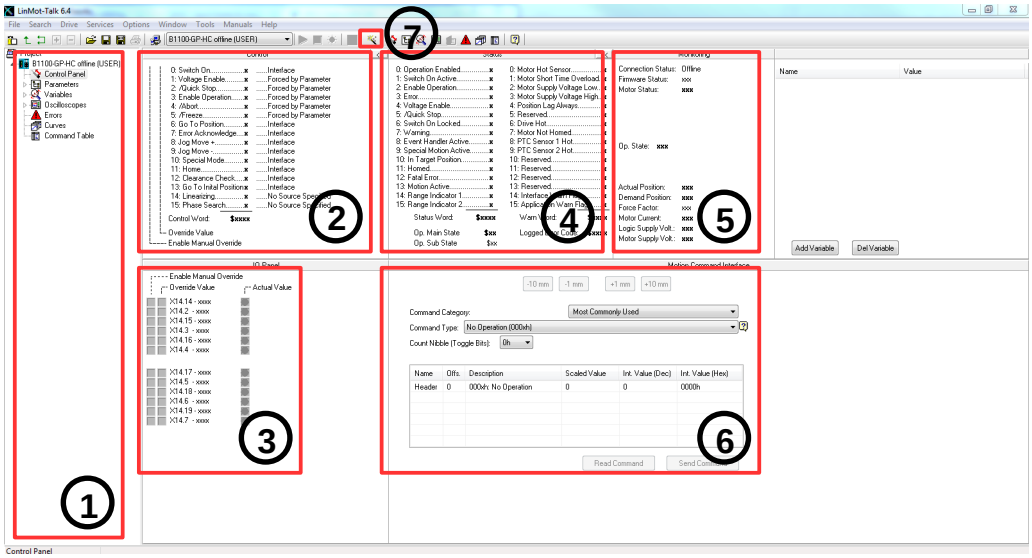


Figure 4.3: LinMot-Talk 6.4 Main Window. 1: Software navigation tree. 2: Current control settings. 3&6: Manually set parameters. 4: Current actuator status. 5: Actuator monitoring. 7: Motor wizard.

→ Motion Control SW → Motion Interface → Run Mode Settings → Run Mode Selection" parameter to "Command Table Mode". This leads to the motor automatically executing the Command Table entry specified in "Parameters → Motion Control SW → Motion Interface → Run Mode Settings → Command Table Settings → Command Table Entry ID" after homing. We set this parameter to 1. From there we navigated to the command table and created a new entry at ID 1 containing the command "Variable/Parameter Access → Write Live Parameter → 61E8h → 00000001h", which corresponds to setting the homing mode to "Mechanical Stop Negative Search". For this step we also checked the "Auto execute new command on next cycle" tick box and set the sequenced entry to 2. On ID 2 we then generated the command "Variable/Parameter Access → Write Interface Control Word → 0043Fh", which has the same effect as manually enabling homing on the Control Panel. With these settings the second motor will first be homed with the other motor as described before and then re-home to the correct position.

The next issue was that, after this second homing procedure, the driver automatically started the command table at ID 1 again. Thus we shifted the previously created entries down one line and added a "Conditions → IF Current Greater Than → 5A → 2 → 4" command. This means that, on first entry into the command table, i.e. at the false home position with a compressed spring and therefore high current, the next command that is executed will be the command table entry at ID 2. At second entry into the command table, the motor will have been homed correctly and therefore be situated in a neutral position with hardly any current.

Thus, the execution will jump to entry ID 4. Now the last task was to set an entry for that case.
Since we want to operate the actuators via analog inputs we had to set the driver up for that. This was done by generating a "Variable/Parameter Access → Write Interface Control Word → 0083Fh" command on ID 4, which sets the "Parameters → Motion Control SW → State Machine Setup → Special Mode → Mode" parameter to "+/-10V Current Command Mode". This leads to the motor generating a current proportional to the differential analog input voltage. The maximum output current is set via the "Parameters → Motion Control SW → State Machine Setup → Special Mode → Current Command Cinfig → 10V Current" parameter. We set this to be 15A, i.e. the maximum allowed current for our actuator type.
For the correctly homed motor at power up we set the Run Mode to Command Table Mode as well but entered the Special Mode Command at ID 1 so that it immediately goes into Current Command Mode after initial homing.
With that our drivers and actuators were setup for analog control via the microcontroller and interface boards. The boards are described in more detail in Chapter 5.

5 Electronics

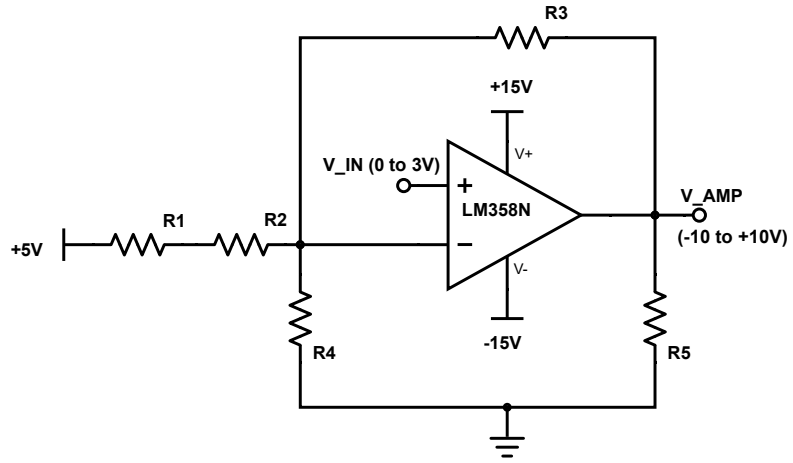
5.1 Motivation

Starting into the project, there was a previous version of the FC2D setup, which first experiments were being performed on. However, this setup had a major drawback, namely the slow microcontroller (MC), sampling at a maximum of 800Hz. While this might have been fast enough to achieve a working transparency controller, it would have been impossible to investigate the influence of sampling rates on performance since we would have had to always use the highest possible sampling frequency. Thus, a new MC was included in the setup. The new MC is the STM32F4 with a maximum bandwidth of 30kHz, enabling us to oversample for a resulting sampling and control frequency of 5kHz with enough range in both directions to vary this frequency if necessary.
While this majorly increased frequency provided us with a much better ability to track small and fast movements, the new MC presented us with a new problem. The old MC had a reference voltage of 4.1V, whereas the new one only used 3.3V. Therefore we had to adjust the remaining electronics to account for this new voltage. The board was originally created for the hydraulic actuators and then adjusted for the electric motors by leaving out the valve current and pressure sensor signal conversion as described in Sections 5.2 and 5.4. We decided to create a new PCB including all electronics except for the MC itself on one board, making wiring easy and convenient. We created one board for each actuator to retain the possibility of exchanging the actuators separately, e.g. for an electric motor. The following sections describe the different parts of the circuit and the final evaluation of the board.

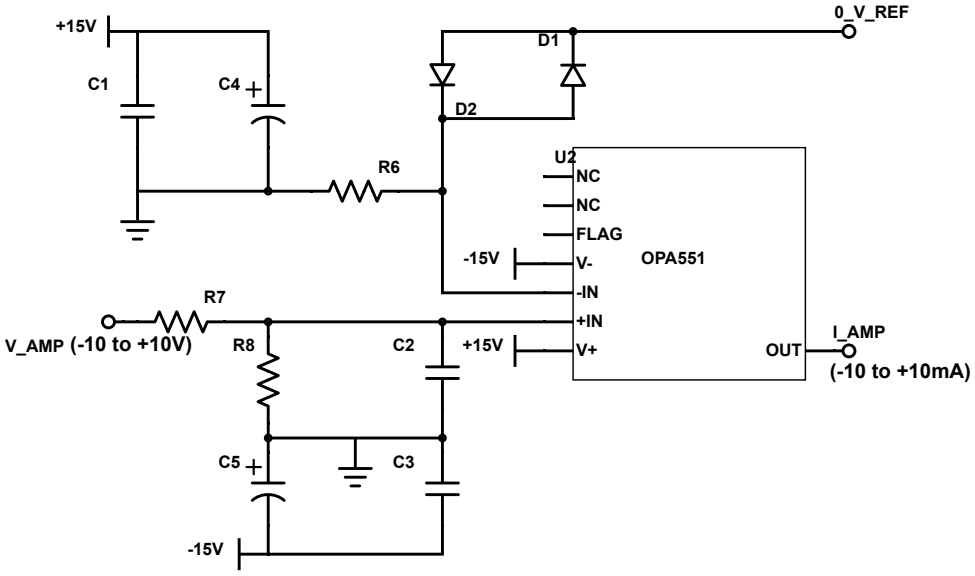
5.2 Valve Signal Conversion

The hydraulic actuators used in FC2D are controlled by two Moog E024 valves, which take a current input between -10 and 10mA and proportionally translate it to opening ratios. Therefore, the signal from the microcontroller has to be converted into the appropriate current range. This is done in two steps. Firstly, the voltage range is adjusted to -10 to $+$10V as seen in Figure 5.1a. This voltage is then put through a second stage which transforms it into current. This can be seen in 5.1. Since there is no current into an op-amp per definition, we can infer the following relationship between the currents through the respective resistors

$$I_1 + I_2 + I_3 = I_4 \tag{5.1a}$$



(a) Valve voltage amplification. R_1=18kΩ, R_2=1.5kΩ, R_3=39kΩ, R_4=10kΩ, R_5=100kΩ



(b) Voltage to current conversion. R_6=500Ω, R_7=47kΩ, R_8=47kΩ, C_1=100nF, C_2=1.8nF, C_3=100nF, C_4=4.7μF, C_5=4.7μF, D_1=yellow, D_2=green

Figure 5.1: Circuits for valve signal conversion.

which, in terms of voltages and resistors with U_{ref} as the 5V input to R_1 and U_{out} as op-amp output, then becomes

$$\frac{U_{ref} - U_-}{R_1 + R_2} + \frac{U_{out} - U_-}{R_3} = \frac{U_-}{R_4} \tag{5.1b}$$

The second thing which we know about op-amps is that they keep both of their inputs at the same voltage. Therefore,

$$\frac{U_{ref} - U_+}{R_1 + R_2} + \frac{U_{out} - U_+}{R_3} = \frac{U_+}{R_4} \tag{5.2a}$$

which results in

$$U_{out} = (\frac{U_+}{R_4} - \frac{U_{ref} - U_+}{R_1 + R_2})R_3 + U_+ = 6.9U_+ - 10V \tag{5.2b}$$

where U_+ refers to the MC signal. Since this part of the circuit is actually adjusted to a MC range of 0 to 3V, this exactly corresponds to a transformed range of -10 to +10.7V. The second part seen in Figure 5.1b is again adapted from a previous IIT design. Hereby the capacitor pairs C_1 and C_4 and C_3 and C_5 serve as decoupling capacitors between the ± 15 voltage rails and ground. This leads to a more stable supply voltage. R_7 and C_2 form a low-pass filter with cutoff frequency

$$f_c = \frac{1}{2\pi R_7 C_2} \approx 1.9kHz \tag{5.3}$$

R_8 is connected to ground, thereby halving the voltage at the positive op-amp input. The voltage gain of the op-amp is given by

$$G = 1 + \frac{R_L}{R_6} = \frac{R_6 + R_L}{R_6} \tag{5.4a}$$

and thus

$$U_{out} = \frac{R_6 + R_L}{R_6} U_{in} \tag{5.4b}$$

where R_L is the load resistance corresponding to the inner resistance of the valve.
We know that, internally, the valve current input (I_{AMP} in Figure 5.1b) and the current feedback ($0V_{REF}$ in Figure 5.1b) are connected through the load resistance. Therefore the output current is equal to

$$I_{out} = \frac{V_{out}}{R_6 + R_L} \tag{5.5a}$$

which, with Equation 5.4b, becomes

$$I_{out} = \frac{\frac{R_6 + R_L}{R_6} U_{in}}{R_6 + R_L} = \frac{U_{in}}{R_6} = 0.002U_{in} \tag{5.5b}$$

Remembering that our -10 to +10V range gets halved by R_8, this leads to an exact transformation to a current range of -10 to +10mA. The two diodes indicate if the output current is positive (green) or negative (yellow).
Since for the LinMots we needed a voltage instead of a current input we used the same PCB layout but left out all parts from Figure 5.1b and instead fed the $\pm 10V$ from Figure 5.1a directly back to the output pins.

5.3 Load Cell Amplification

This part was taken directly from a previous design done at IIT. It amplifies the load cell voltage to the full MC 0 to 3.3V range. The only adjustment we had to do was to change the reference voltage to 3.3V instead of 4.1V. Figure 5.2 shows all of the components. The gain

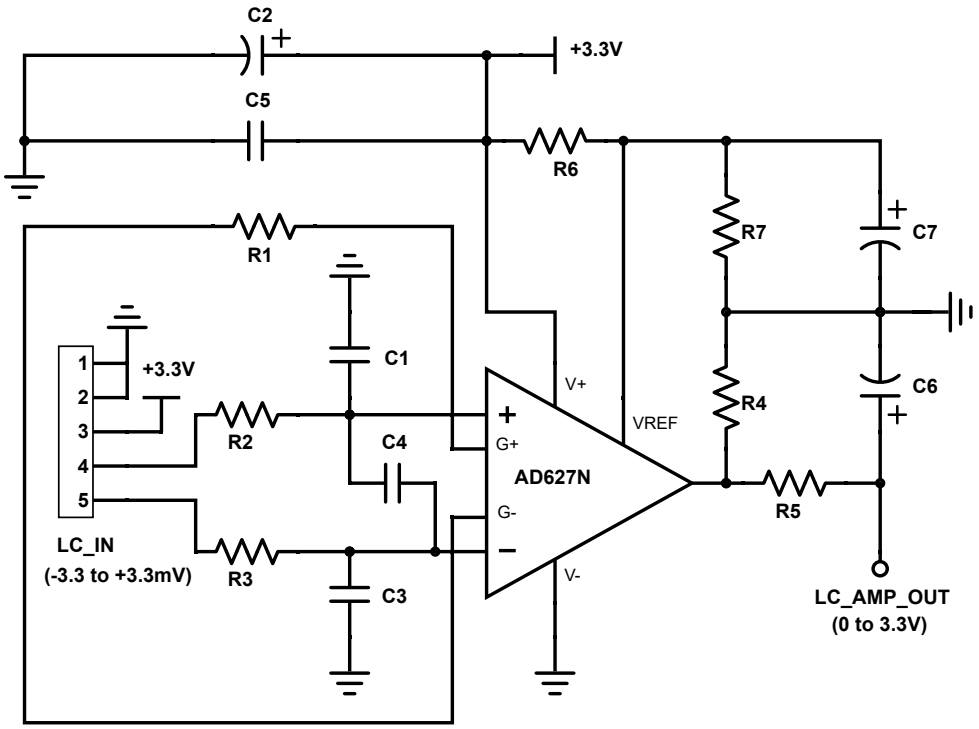


Figure 5.2: Circuit diagram for load cell amplification. R_1=402Ω, R_2=20kΩ, R_3=20kΩ, R_4=100kΩ, R_5=200Ω, R_6=47kΩ, R_7=47kΩ, C_1=1nF, C_2=0.33μF, C_3=1nF, C_4=22nF, C_5=10nF, C_6=0.33μF, C_7=1μF

of the AD627 op-amp depends purely on resistor R_1, which determines it to be

$$G = 5 + (\frac{200k\Omega}{R_1}) = 502.5 \tag{5.6}$$

The RC pairs R_2 and C_1 and R_3 and C_3 are low-pass filters for the input signal. The cutoff frequency for both filters is

$$f_c = \frac{1}{2\pi R_2 C_1} = \frac{1}{2\pi R_3 C_3} \approx 8kHz \tag{5.7}$$

C_4 serves as decoupling capacitor between the two op-amp inputs, making the input voltages more stable. C_2 and C_5 do the same for the supply rail. R_5 and C_6 form an additional low-pass filter at the output with a cutoff of

$$f_c = \frac{1}{2\pi R_5 C_6} \approx 2.4kHz \tag{5.8}$$

R_4 serves as a pull-down resistor to ground, stabilizing the op-amp output. R_6 and R_7 set the local reference voltage (local ground) for the op-amp to half the supply voltage with their equal resistance values. C_7 again acts as a decoupling capacitor between the actual and the local ground. This leads to an output voltage of

$$U_{out} = GU_{in} + U_{ref} = 502.5U_{in} + 1.67V \tag{5.9}$$

with the previously calculated gain and the local ground U_{ref}. This produces an output range of approximately 0 to 3.3V. The approximate value is due to the fact that we rely on standard resistor values for R_1. The deviation from the desired values is in the millivolt range.

5.4 Pressure Sensor Signal Conversion

The most straightforward part of the board was the conversion of the pressure sensor signals from their output range of 0 to 5V to the MC input range of 0 to 3.3V. Figure 5.3 shows the circuit diagram for one sensor. This circuit was included four times per board - twice for the two pressure sensors needed per actuator, once for the tank and once for the supply pressure. The first stage of the circuit consists of a basic voltage divider formed by resistors R_1 and R_3.

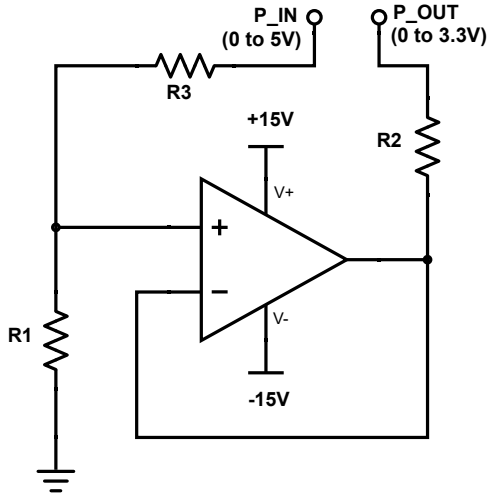


Figure 5.3: Circuit diagram for pressure sensor signal conversion. R_1=39kΩ, R_2=39kΩ, R_3=20kΩ

The voltage at the positive input of the op-amp then becomes

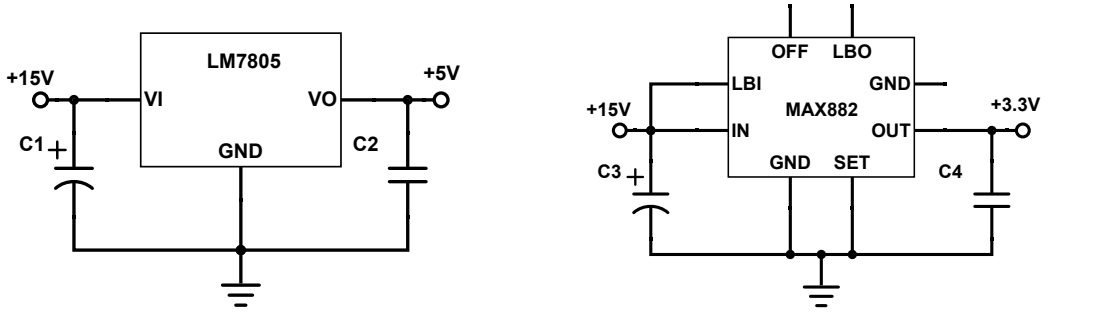
$$U_+ = U_{in}\frac{R_1}{R_1 + R_3} \approx 0.66U_{in} \tag{5.10}$$

which exactly transforms the pressure sensor range (0 to 5V) to the microcontroller range (0 to 3.3V). The op-amp itself acts as a buffer to make the voltage conversion more stable and independent of resistive loads. With no parallel resistor it has a gain of one, which leads to the voltage at the output being equal to the positve input voltage. Since, per definition, the input

current into an op-amp is zero, the current through R_2 is zero as well resulting in no voltage drop between the op-amp output and the output pin. The purpose of R_2 is the attenuation of very high frequency oscillations from the op-amp output, also resulting in a more stable output signal.
Again, this part was left out on the boards for the electric motors since it is only necessary for the hydraulic pressures.

5.5 Auxiliary parts

Finally, there are three auxiliary circuit parts needed for the functional parts described in the previous sections. First of all, there are two voltage regulator circuits. One produces 3.3V and is used as a reference voltage for the load cell amplifier. The other one produces 5V and is used for the same purpose in the first stage of the valve amplifier. Both circuits can be seen in Figure 5.4.
Figure 5.5 shows the third auxiliary circuit part. This part splits the applied 30V into two voltage rails of ± 15V. We need this for the valve amplifier since the voltage range is shifted down to -10V, which requires a dual power supply for the op-amp. The LED is added to indicate whenever the board is powered.



(a) 5V voltage regulator. C_1=0.3μF, C_2=0.1μF (b) 3.3V voltage regulator. C_3=0.1μF, C_4=2.2μF

Figure 5.4: Auxiliary voltage regulator circuits.

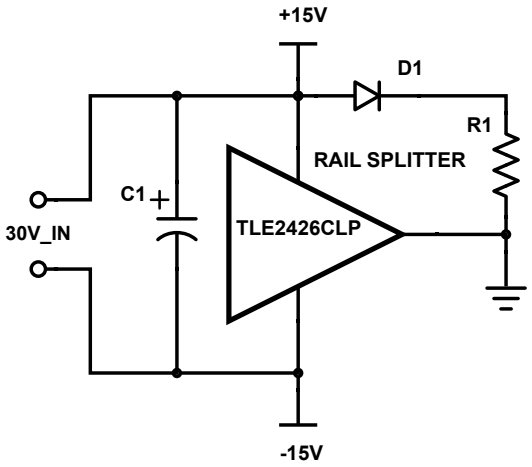


Figure 5.5: Rail splitter for transforming 30V to ±15V. R_1=1kΩ, C_1=220μF, D_1=red

5.6 Evaluation

To verify that our new board worked the way it was supposed to and was fast enough to drive the required signals, we performed some evaluation experiments. For both pressure sensor and load cell parts, we just put in a constant voltage from a DC power supply and directly measured if the output range was correct.

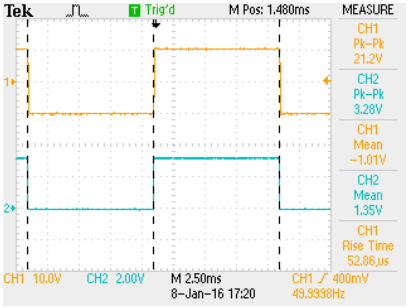
The valve amplification part however needed some further investigation. Since the valves need to react quite quickly during a control task, we wanted to additionally evaluate the dynamic behaviour of the circuit. To this end, we applied a square wave to the circuit and examined the response with an oscilloscope. The following figures show the results for different frequencies and amplitudes. The full amplitude range always refers to a square wave input between 0 to 3.3V, ideally resulting in an output of -10 to +10mA.

From Figure 5.6a we can see that the first stage of the valve amplifier works almost perfectly, transforming the 0 to 3.3V square wave to -10 to +10V. Thus we concentrated on the current conversion stage, as seen in Figures 5.6b-5.9b. There we can see that while at 50Hz the response is still reasonably fast for all amplitude ranges, we run into issues when driving the signal at higher frequencies and full amplitude range. Especially the 500Hz wave shows that, at full amplitude range, we do not even reach the maximum value within the step time. At quarter amplitude range however - which we will mostly be staying in - the maximum value is still reached fairly quickly, still enabling us to experiment at high frequencies.

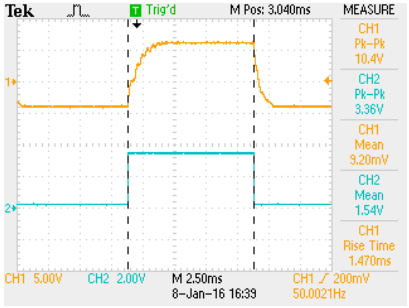
As a means to quantify the performance we estimated the circuit bandwidth for full, half and quarter according to

$$BW \approx \frac{0.35}{T_r} \tag{5.11}$$

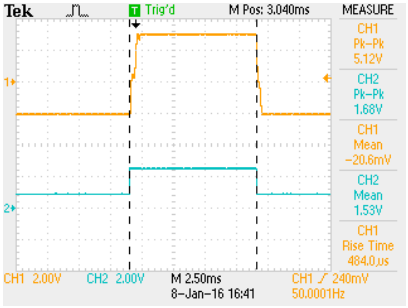
where T_r is the current rise time. We had the oscilloscope display rise times, giving us values of $T_{r,100} = 1.47ms$, $T_{r,50} = 484\mu s$, $T_{r,25} = 197.8\mu s$ for full, half and quarter range respectively.



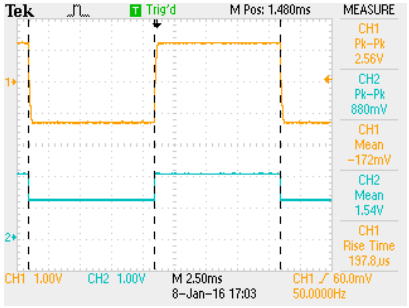
(a) Voltage response at full amplitude range.



(b) Current response at full amplitude range.



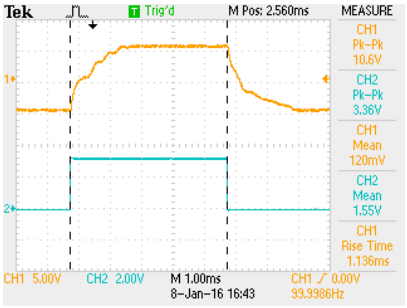
(c) Current response at half amplitude range.



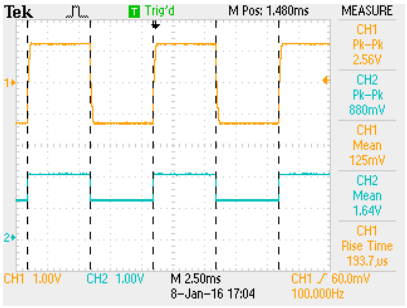
(d) Current response at quarter amplitude range.

Figure 5.6: Board response to a 50Hz square wave. Blue: square wave reference. Orange: board response. The voltage response follows the reference almost perfectly without any significant delay. For the current response, the delay gets smaller with a decrease in amplitude, making the lower amplitude range responses almost immediate.. The full reference value is reached for all amplitude ranges.

This results in approximate bandwidth values of $BW_{100} \approx 240Hz$, $BW_{50} \approx 720Hz$ and $BW_{25} \approx 1.77kHz$, which is consistent with our graphical evaluation at higher frequencies.
To make sure that we would not decrease performance with our new board compared to the old setup, we did the same experiments with the Moog G123-815 Buffer Amplifier, which was previously used. Figures 5.10-5.13 show that, while the Moog amplifier performs faster at higher frequencies, it has a consistent large overshoot over the whole frequency range, resulting in hard to distinguish high and low values. Clearly, the focus of this amplifier lies on speed as opposed to a critically damped step response. This is apparent from the bandwidth evaluation as well. Rise time varies between $30 - 40\mu s$ for quarter and full range, resulting in approximate bandwidths of $BW_{100} \approx 8.75kHz$ and $BW_{25} \approx 11.67kHz$. Despite those higher bandwidths however, the large overshoot leads to similar settling times for the Moog amplifier and our board.
Since we were not aiming at control frequencies higher than 1.5kHz and we mostly stay within smaller amplitude ranges, we concluded that the performance of our new board was good enough to work for our setup in spite of the lower bandwidth.

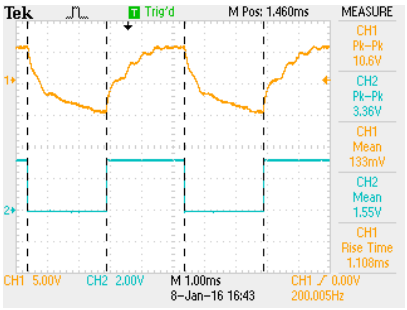


(a) Current response at full amplitude range.

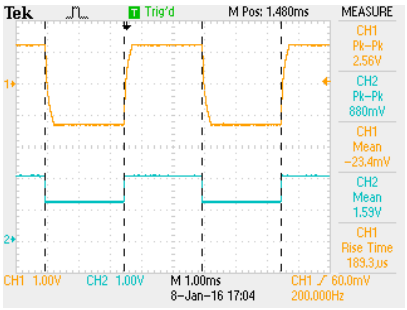


(b) Current response at quarter amplitude range.

Figure 5.7: Board response to a 100Hz square wave. Blue: square wave reference. Orange: board response. The full reference value is reached for all amplitude ranges. The full amplitude response takes almost 50% of the pulse time to reach the reference.

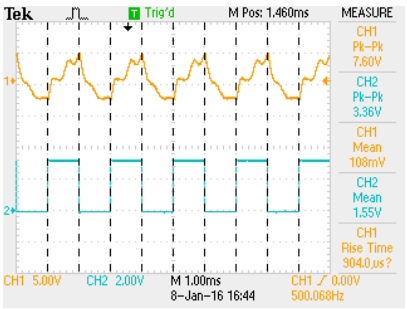


(a) Current response at full amplitude range.

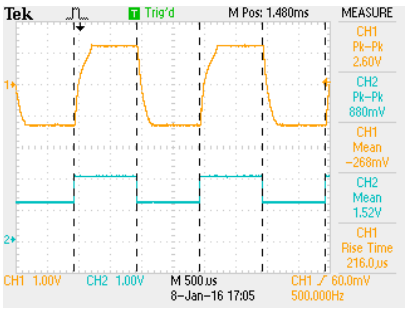


(b) Current response at quarter amplitude range.

Figure 5.8: Board response to a 200Hz square wave. Blue: square wave reference. Orange: board response. The full reference force is reached for all amplitude ranges. The full amplitude response takes over 50% of the pulse time to reach the reference.

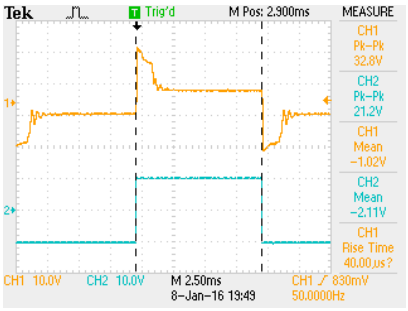


(a) Current response at full amplitude range.

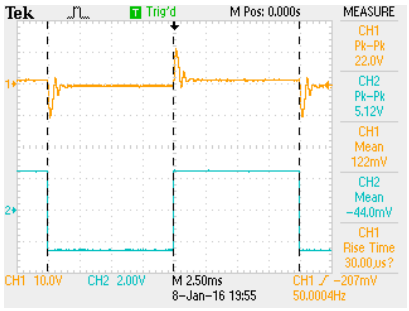


(b) Current response at quarter amplitude range.

Figure 5.9: Board response to a 500Hz square wave. Blue: square wave reference. Orange: board response. The full amplitude response does not reach the reference within the pulse time anymore. The quarter amplitude range response still reaches the full reference with a small delay.

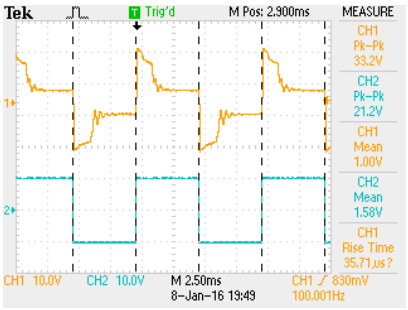


(a) Current response at full amplitude range.

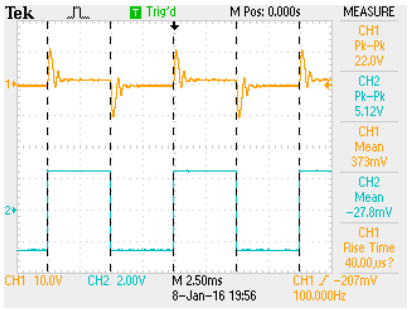


(b) Current response at quarter amplitude range.

Figure 5.10: Moog amplifier response to a 50Hz square wave. Blue: square wave reference. Orange: board response. The full value is reached almost immediately with a large overshoot. A steady state is reached after approximately 10-30% of the pulse time.

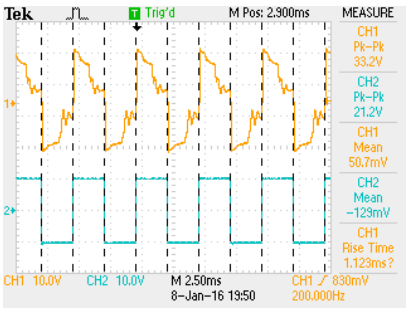


(a) Current response at full amplitude range.

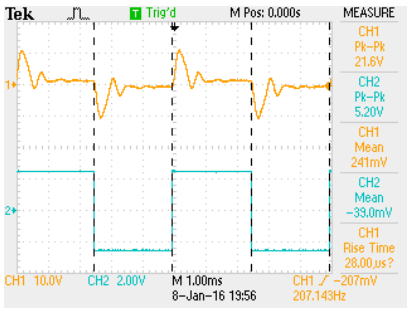


(b) Current response at quarter amplitude range.

Figure 5.11: Moog amplifier response to a 100Hz square wave. Blue: square wave reference. Orange: board response. The full value is reached almost immediately with a large overshoot. A steady state is reached after approximately 20-50% of the pulse time.

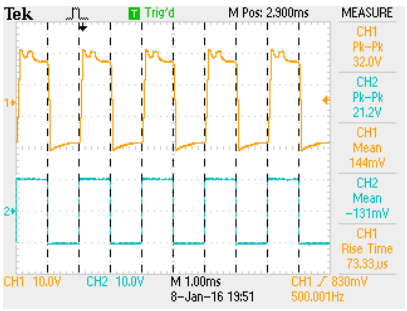


(a) Current response at full amplitude range.

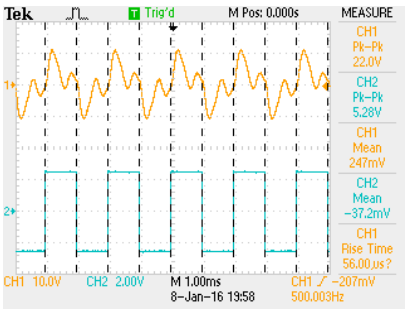


(b) Current response at quarter amplitude range.

Figure 5.12: Moog amplifier response to a 200Hz square wave. Blue: square wave reference. Orange: board response. The full value is reached almost immediately with a large overshoot. A steady state is reached after approximately 40-60% of the pulse time.



(a) Current response at full amplitude range.



(b) Current response at quarter amplitude range.

Figure 5.13: Moog amplifier response to a 500Hz square wave. Blue: square wave reference. Orange: board response. The full value is reached almost immediately with a large overshoot. No steady state is reached.

6 Experiments

6.1 Code Adjustments

Since most of the needed functions like the operating system and basic controllers were already implemented, the main addition to the code was the Kalman filter. The biggest challenge in implementing a Kalman filter on a microcontroller is speed since it inherently does not have the computation power of a full computer. A Kalman filter consists of several matrix operations, which require quite a lot of computational power, even if programmed very efficiently. We want to use our Kalman filter results for the controllers in real time. Thus, while the filter does not necessarily have to run in real time due to slower changes in the measurements, it still has to run at a frequency similar to the sensor sampling frequencies, e.g. 400Hz for the IMUs.
We first experimented with the Meschach library and self-written matrix functions but finally settled on the CMSIS DSP library, which is specifically tailored to Cortex-M processors. Since our MC uses a Cortex-M4, this seemed to be the most efficient solution.
The Kalman filter code consists of two main parts, both of which can be seen in Appendix A. Appendix A also lists the structures used in the Kalman functions.
The initialization is called in the main function. It sets all of the initial values and system matrices and allocates memory for several auxiliary matrices. Those auxiliary matrices are needed since we can only perform one matrix operation at a time and therefore need a place to store our result until it is used in the next step.
The Kalman filter itself is run in the real time tasks function. Therefore, as of now, the filter is running at the real time frequency of 5kHz. This function executes one matrix operation after the other according to the Kalman filter equations shown in Section 2.2.5. The following sections show and discuss our preliminary experimental results.

6.2 Results

6.2.1 Controller

Firstly, we performed an experiment with the existing controller code to verify that our hardware adjustments had led to good controller performance. As of now, only the feedforward controller is implemented. The results of this experiment can be seen in Figure 6.1. It shows force tracking of the desired robot force as well as the resulting interaction force.

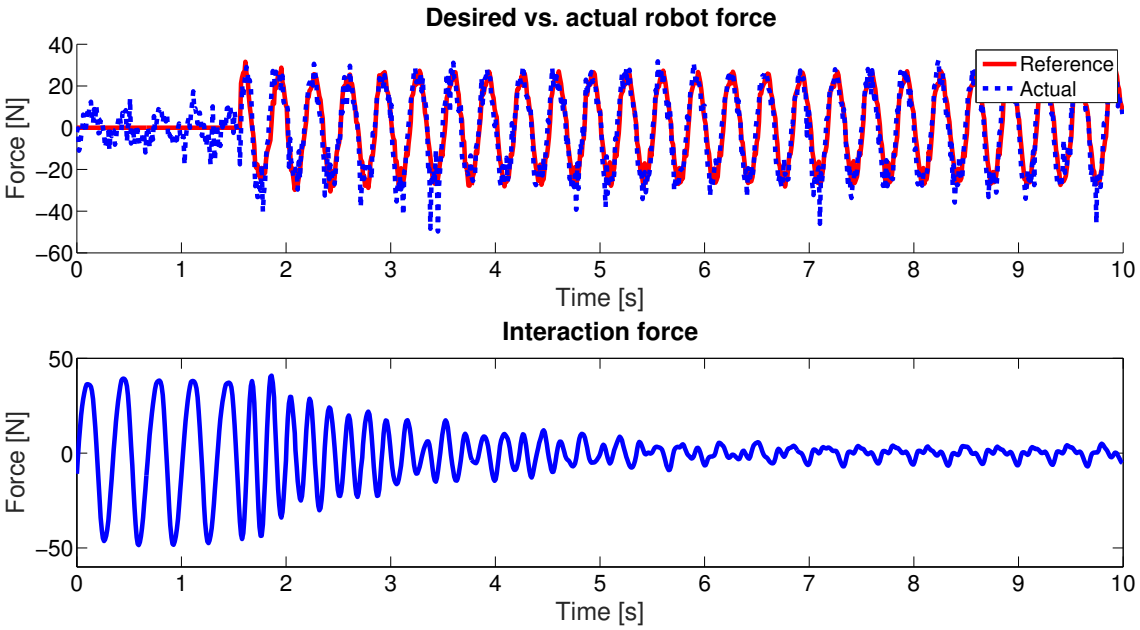


Figure 6.1: First transparency control experiment. This experiment was performed with feed-forward control only. The controller is switched on at t=1.5s. Top: Robot force reference versus actual applied force. Bottom: Resulting interaction force. The force tracking is very accurate in both phase and frequency. Some noise peaks can be seen in the amplitude. The controlled interaction force retains some oscillations stemming from attachment dynamics.

6.2.2 Offline Kalman Filter

As a first evaluation of our Kalman filter, we ran the data from the force control experiment through an offline Kalman filter in MATLAB. We used the exact filter algorithm described in Section 2.2.5 instead of MATLAB's own 'kalman' function to ascertain its functionality for later implementation on the microcontroller. The results that we obtained can be seen in Figure 6.2.

6.2.3 Kalman Filter

We performed four different experiments on our Kalman filter. For technical reasons we could only perform these experiments on a setup with the masses detached and position and force measurements only. We thus decided to adjust the Kalman filter to a simpler version estimating robot force with the available measurements to prove that our system was working as expected. This simpler filter is also the one shown in the code in Appendix A since the other version is untested on the experimental setup.

For this filter we use robot position, velocity, acceleration and force as our four states. We performed two different types of experiments, one with only position and one with both position and force as measurements. For both of these cases we examined both the case where we moved the mass by hand and the case where the mass was moved by the electric motor. The

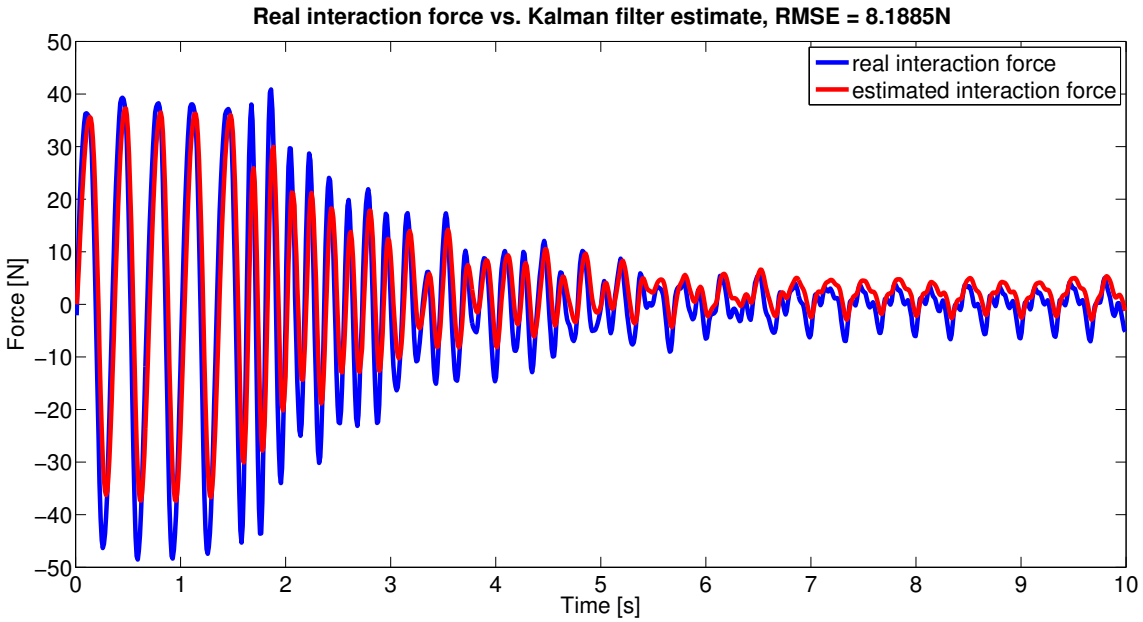


Figure 6.2: Offline Kalman filter experiment. The estimate follows the real force closely in both shape and phase. Only the amplitude is slightly off.

adjusted Kalman matrices, which vary from the ones shown in Section 2.2.5, are listed in Appendix B.

The results of these four experiments are shown in Figures 6.3 to 6.6, where the first two correspond to the case with one measurement and the second two use both measurements. A detailed discussion of the results can be found in the Section 6.3.

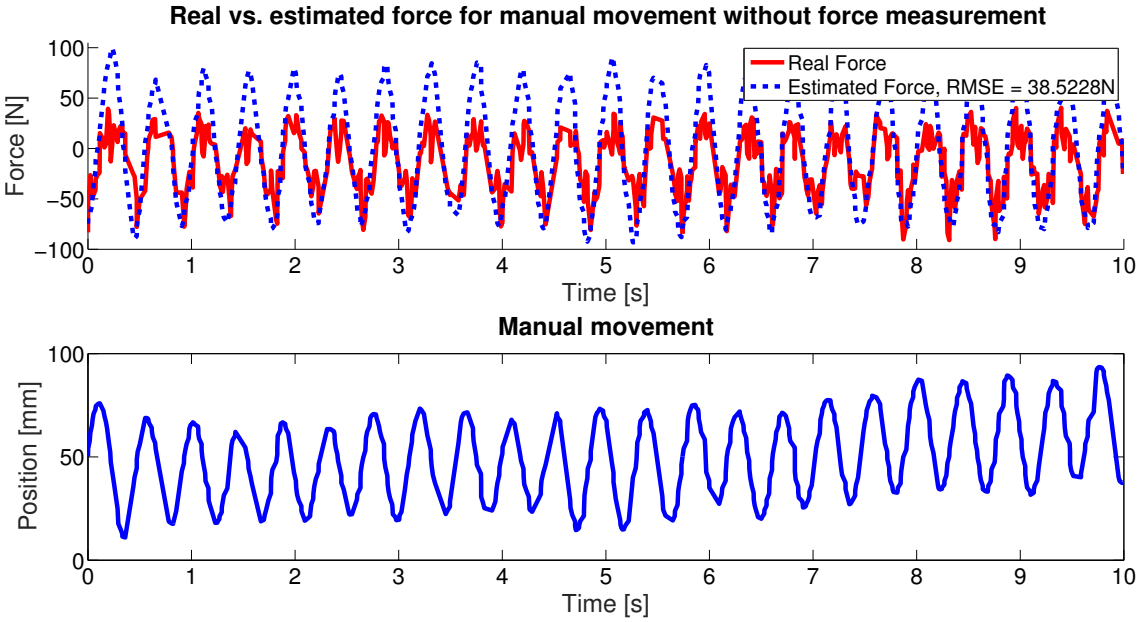


Figure 6.3: Kalman filter experiment with manual movement and one measurement. The mass was moved back and forth by hand while measuring position. From this, applied force was estimated. Top: Actual versus estimated force applied onto the mass. Bottom: Position of the mass. The estimate is accurate regarding frequency and phase. The estimated amplitude is slightly too large.

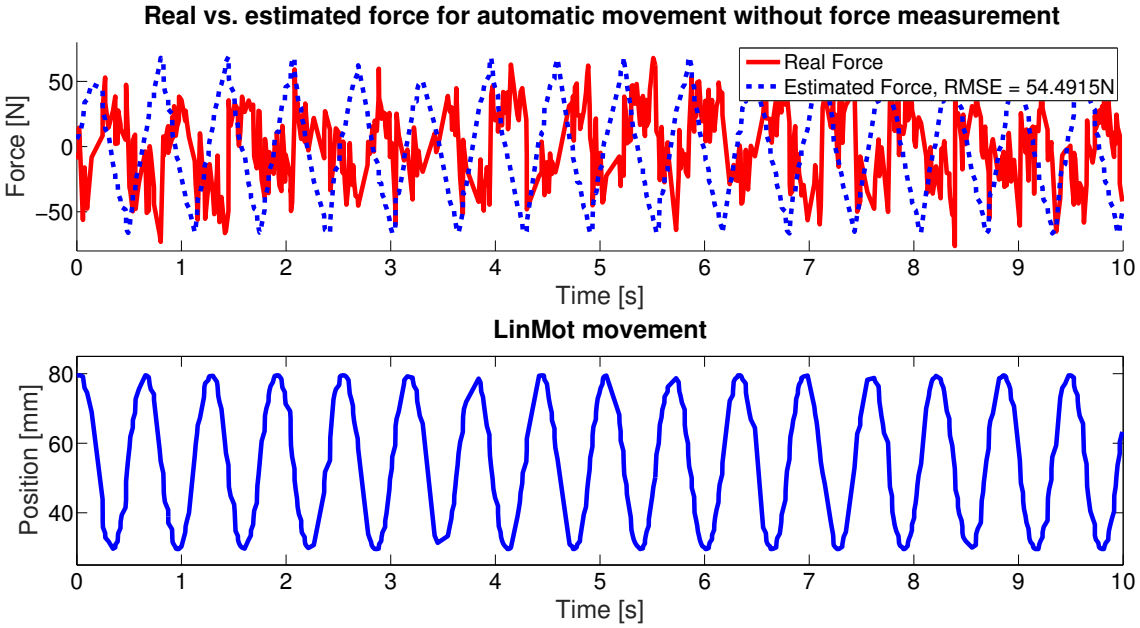


Figure 6.4: Kalman filter experiment with automatic movement and one measurement. The mass was moved back and forth by the electric motor while measuring position. From this, applied force was estimated. Top: Actual versus estimated force applied onto the mass. Bottom: Position of the mass. The estimate is accurate in phase and amplitude but shifted by 30-40% of a period.

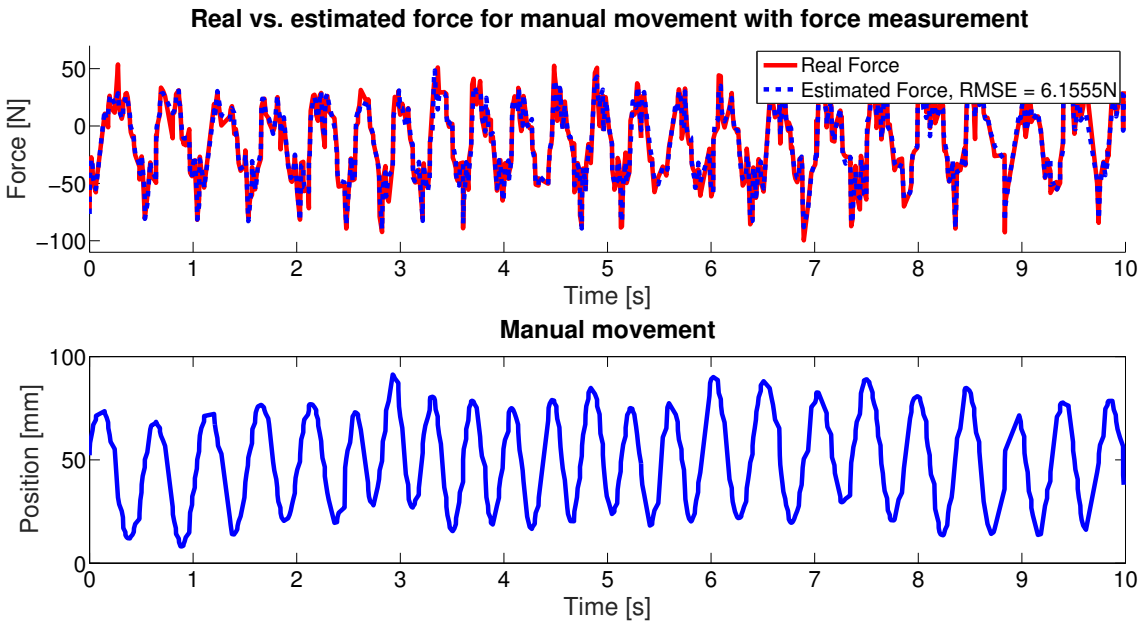


Figure 6.5: Kalman filter experiment with manual movement and two measurements. The mass was moved back and forth by hand while measuring position and force. From this, applied force was estimated. Top: Actual versus estimated force applied onto the mass. Bottom: Position of the mass. The estimate is very accurate in all regards.

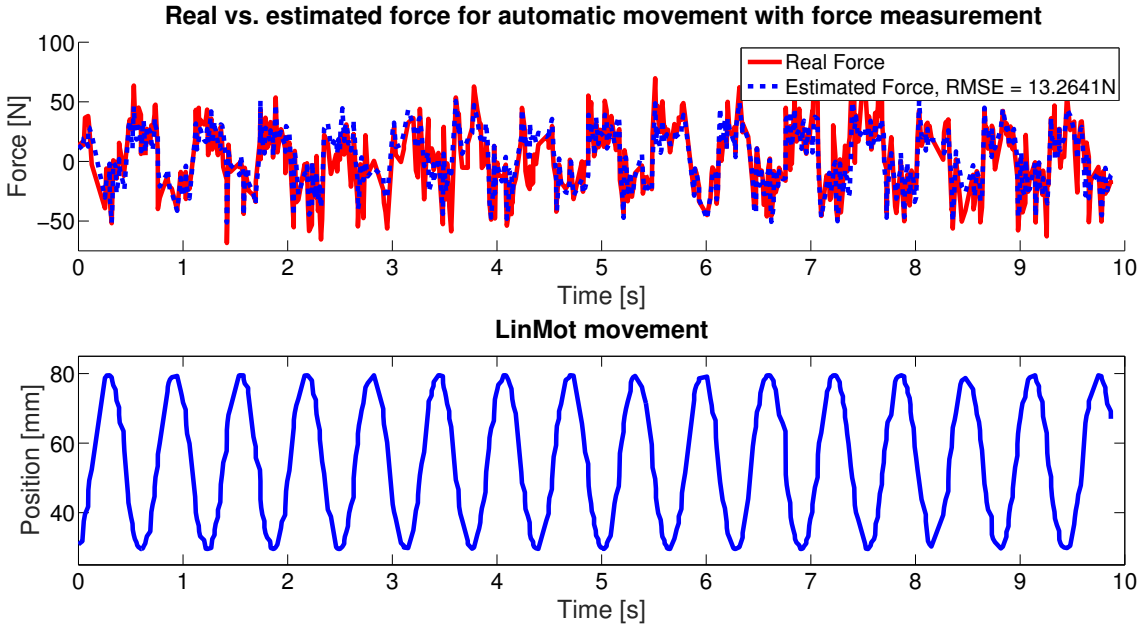


Figure 6.6: Kalman filter experiment with automatic movement and two measurements. The mass was moved back and forth by the electric motor while measuring position and force. From this, applied force was estimated. Top: Actual versus estimated force applied onto the mass. Bottom: Position of the mass. The estimate is very accurate in all regards, additionally filtering out some of the measurement noise in the interaction force.

6.3 Discussion

The controller experiment shows us that, even when using feedforward only, the controller can reduce interaction force significantly. As expected , some oscillations remain in the system. We experienced the same phenomenon in simulation as seen in Figure 3.21. This might be due to nonzero initial forces or unmodelled nonlinearities in the system or actuator dynamics. Looking at reference force tracking, we see that the desired force is being tracked very closely apart from minor noise issues.
If we were aiming at controlling a sinusoidally moving system such as this one, the feedforward controller might even be enough to achieve a transparent feeling for the user, assuming that the residual oscillations are small enough. We do however see a gradient decrease in interaction force when the controller is switched on. This leads us to the assumption that, if the motion was more complicated than a simple sine, we would see this transient behaviour for every change, forcing us to add the feedback controllers. This will have to be investigated further in future experiments with and without feedback control.

The offline Kalman filter shows us that our results from simulation are reproducible on hardware. The estimate follows the real force closely, especially regarding phase, which according to simulation is one of the most critical factors. The inaccuracies in amplitude are most likely due to insufficient tuning of the filter or inaccurate initial conditions. This might be improved in future versions or even neglected since amplitude variations in this degree might not influence controller performance in a significant way.

The first two Kalman experiments are the closest that we could get to imitate the real estimation task without actually being able to investigate the interaction force. Both experiments aim at estimating a state which is not being directly measured. The first experiment shows us that , especially regarding phase, we can estimate our robot force very accurately. There seems to be some constant error in the estimation amplitude which might be due to unsufficient tuning of the filter. This offset explains the large RMSE value. Comparing these results to the second experiment, we see that the Kalman filter performed better in the first case. While the RMSE is large in both values, there is a significant phase shift of the estimate with respect to the measured force in the second case. However, we are unsure as of now if this is an actual decrease in estimation quality since the Kalman filter is the exact same for both cases. The force measurement on the other hand seems to be significantly more noisy when the mass is moved by the actuator. This is probably caused by vibrations of the motor itself or the cooling fan. Future investigations will have to show if this issue persists when more measurements are available, making an estimation fault less likely.
The second two Kalman experiments are less related to our final goal since, for now, we assume that we cannot measure the estimated parameter. Still, these experiments can give us information on our system behaviour combined with a Kalman filter and thus are included

nonetheless. Again, in the first experiment, the estimate is close to the actual force. In this case the RMSE is small as well, which is to be expected considering that we are using the measured force for estimation. Comparing the result to the second experiment we can see a decrease in performance just like previously, yet the decrease is not as major this time. This conforms to visual inspection showing us significantly less phase shift in the estimate. As previously mentioned, this improvement has to be taken with caution however since we are not certain if there is an issue with the force measurement when the motor is active. In any case, these last two experiments show us that the inherent sensor fusion of the Kalman filter improves the estimation quality the more sensors are available.

7 Conclusion and Outlook

We began this project with the aim to get one step closer to implementing a new transparency control paradigm on an exoskeleton for balance support. The main goal was to simulate and experiment on the linear case as a simpler model for the articulated robot.
In the course of this work we achieved a fully working linear experimental setup with significantly better controller performance than the previous version. We also implemented a Kalman filter for interaction force estimation and showed its functionality in a simpler case. In addition to this we showed that our full Kalman filter works in an offline implementation. Furthermore, we extensively simulated the linear case and evaluated it in terms of control and Kalman filter performance.

Overall, our simulation results show that the proposed control paradigm is a good approach for achieving transparency under many different conditions. We tried to design our simulated cases as general and versatile as possible to obtain results which would hold in linear experiments and be translatable to the articulated case. With our separate investigation of possible performance influencing factors we are confident that we achieved this goal.
Our evaluation of the simulated Kalman filter revealed precise knowledge of the attachment between human and robot to be the crucial factor in interaction force estimation. This has to be kept in mind for future experiments to see how much uncertainty we can tolerate. This result is significant to us since the final exoskeleton will have an elastic band fixation whose force dynamics we will only be able to model approximately. Other factors seemed to have a smaller or even insignificant influence on Kalman filter performance which leaves us optimistic that we will be able to achieve high estimation accuracy.
Controller simulation showed timing issues like low actuator bandwidth or measurement delays to be the highest contributing factor to control issues. We were able to reduce those problems in the experimental setup by installing the much faster electric motors in place of the previous hydraulic actuators. Thus, and because of the high sampling rate of our microcontroller, we believe that we can achieve robust and stable transparency control as seen in simulation on our setup.

The hardware adjustments performed in the course of this project enable us to conduct detailed experiments to validate the results obtained in simulation. As opposed to the previous setup, the force controller can now perform in a stable manner even when we use a pure feedforward command. This reassures us in our conclusions from simulation, since the setup behaves exactly as expected. Similarly, we have shown in experiments that we are able to use a Kalman

filter in real time to estimate parameters for which no measurement is available. While we have not performed this experiment with the spring interaction yet, we see no reason why this would change since the principle is the same with the only difference being other system and Kalman matrices.

Future work on this project first and foremost relates to more extensive experiments. While the general concept of both the controller and Kalman filter has been proven to work in practice as well as simulation, more detailed experimental analysis has to be performed relating to all of the simulated cases.
First of all, the controller up until now only consists of a feedforward command. While, as shown in Section 6.2.1, this works well for specific cases, simulation has shown us that to cover arbitrary initial conditions and motions we need both feedback controllers. Thus, these controllers need to be implemented and tested similarly to the feedforward controller.
Furthermore, the Kalman filter experiments have to be extended to the real interaction force estimation. Notwithstanding our promising experimental results so far, we can only make valid statements on Kalman filter performance when we have tested it in its real application. Additionally, the Kalman filter also has an influence on controller performance, which has to be investigated in a separate experiment.

Subsequently, the results from linear analysis have to be translated to the articulated case before they can be tested on the exoskeleton. We expect a simple articulated joint with two links to behave very similar to the linear spring-mass model we examined here. Verifying this assumption in simulation is the next step after conclusion of the linear case experiments. If the results hold true for the articulated case, the successive step is then to implement the complete control paradigm on the exoskeleton and test it.
Finally, the last step will be to combine the transparency with the balance controller to achieve the full desired exoskeleton functionality.

A Code

A.1 Structures

```
// system matrices (constant)
typedef struct {
        // state space matrices A & B (C & D are not needed since we only care
            about states, not output)
        arm_matrix_instance_f32  A;
        arm_matrix_instance_f32  B;

        // measurement noise matrix
        arm_matrix_instance_f32  H;
        // transposed measurement noise matrix
        arm_matrix_instance_f32  HT;
} system_mat;

// covariance matrices (constant)
typedef struct {
        // error covariance matrices accounting for disturbance and noise
        arm_matrix_instance_f32  Q;
        arm_matrix_instance_f32  R;
} kalman_mat;


// states etc (variable)
typedef struct {
        arm_matrix_instance_f32  currstate;
        arm_matrix_instance_f32  currmeas;
        arm_matrix_instance_f32  currcontrolin;
        arm_matrix_instance_f32  Pprev;
        arm_matrix_instance_f32  Kprev;
} kalman_in;

// complete matrices needed for kalman
typedef struct {
        kalman_in in;
        system_mat system;
        kalman_mat co;
} kalman;
```

A.2 Initialization

```
void initKalman(kalman* KF) //initialize Kalman matrices, input, output etc.
{
    /**********************/
    /*** const matrices ***/
    /**********************/

    // system matrix
    arm_mat_init_f32(&KF->system.A, nstates, nstates, (float32_t *)A_matrix);
    // input matrix
    arm_mat_init_f32(&KF->system.B, nstates, ncontrols, (float32_t *)B_matrix);
    // measurement matrix
    arm_mat_init_f32(&KF->system.H, nmeasures, nstates, (float32_t *)H_matrix);
    // transposed measurement matrix
    arm_mat_init_f32(&KF->system.HT, nstates, nmeasures, (float32_t *)HT_matrix);

    // process noise covariance matrix
    arm_mat_init_f32(&KF->co.Q, nstates, nstates, (float32_t *)Qinit);
    // measurement noise covariance matrix
    arm_mat_init_f32(&KF->co.R, nmeasures, nmeasures, (float32_t *)Rinit);

    // identity matrix
    arm_mat_init_f32(&In, nstates, nstates,(float32_t *)I);

    /*************************/
    /*** variable matrices ***/
    /*************************/

    // state covariance matrix
    arm_mat_init_f32(&KF->in.Pprev, nstates, nstates, Pinit);
    // Kalman gain
    arm_mat_init_f32(&KF->in.Kprev, nstates, nmeasures, Kinit);

    // initial state
    arm_mat_init_f32(&KF->in.currstate, nstates, 1, init_state);
    // initial measurement
    arm_mat_init_f32(&KF->in.currmeas, nmeasures, 1, init_meas);
    // initial control input
    arm_mat_init_f32(&KF->in.currcontrolin, ncontrols, 1, init_control);

    // auxiliary matrices
    arm_mat_init_f32(&temp_mat_n1a, nstates, 1,n1a_init);
    arm_mat_init_f32(&temp_mat_n1b, nstates, 1,n1b_init);
    arm_mat_init_f32(&temp_mat_nna, nstates, nstates,nna_init);
    arm_mat_init_f32(&temp_mat_nnb, nstates, nstates,nnb_init);
    arm_mat_init_f32(&temp_mat_nm,nstates,nmeasures,nm_init);
    arm_mat_init_f32(&temp_mat_mma,nmeasures,nmeasures,mma_init);
    arm_mat_init_f32(&temp_mat_mmb,nmeasures,nmeasures,mmb_init);
    arm_mat_init_f32(&temp_mat_m1a,nmeasures,1,m1a_init);
    arm_mat_init_f32(&temp_mat_m1b,nmeasures,1,m1b_init);
}
```

A.3 Kalman Filter

```
void KalmanFilter(kalman* KF)
{
     /***********************/
     /*** prediction step ***/
     /***********************/

     // x_k+1 = A * x_k + B * u_k

     // x_k+1 = A * x_k
     arm_mat_mult_f32(&KF->system.A, &KF->in.currstate, &KF->in.currstate);
     if(ncontrols>0)
     {
        // B * u_k
        arm_mat_mult_f32(&KF->system.B, &KF->in.currcontrolin, &temp_mat_n1b);
        // x_k+1 = x_k+1 + ans
        arm_mat_add_f32(&KF->in.currstate, &temp_mat_n1b, &KF->in.currstate);
     }

     // P_k+1 = A * P_k * A^T +Q

     // A^T
     arm_mat_trans_f32(&KF->system.A,&temp_mat_nna);
     // P_k * A^T
     arm_mat_mult_f32(&KF->in.Pprev, &temp_mat_nna, &temp_mat_nnb);
     // A * ans
     arm_mat_mult_f32(&KF->system.A, &temp_mat_nnb, &temp_mat_nna);
     // P_k+1 = ans + Q
     arm_mat_add_f32(&temp_mat_nna, &KF->co.Q, &KF->in.Pprev);

     /***********************/
     /*** correction step ***/
     /***********************/

     // K_k = P_k * H^T * ( H * P_k * H^T + R )^-1

     // P_k * H^T
     arm_mat_mult_f32(&KF->in.Pprev,&KF->system.HT,&temp_mat_nm);
     // H * ans
     arm_mat_mult_f32(&KF->system.H, &temp_mat_nm, &temp_mat_mma);
     // ans + R
     arm_mat_add_f32(&temp_mat_mma, &KF->co.R, &temp_mat_mmb);
     // ans^-1
     arm_mat_inverse_f32(&temp_mat_mmb,&temp_mat_mma);
     // H^T * ans
     arm_mat_mult_f32(&KF->system.HT,&temp_mat_mma,&temp_mat_nm);
     // K_k = P_k * ans
     arm_mat_mult_f32(&KF->in.Pprev,&temp_mat_nm,&KF->in.Kprev);

     // x_k = x_k + K_k * ( z_k - H * x_k )

     // H * x_k
     arm_mat_mult_f32(&KF->system.H,&KF->in.currstate,&temp_mat_m1a);
     // z_k - ans
     arm_mat_sub_f32(&KF->in.currmeas,&temp_mat_m1a,&temp_mat_m1b);
```

```
    // K_k * ans
    arm_mat_mult_f32(&KF->in.Kprev, &temp_mat_m1b, &temp_mat_n1a);
    // x_k = x_k + ans
    arm_mat_add_f32(&KF->in.currstate, &temp_mat_n1a, &KF->in.currstate);

    // P_k = ( I - K_k * H ) * P_k

    // K_k * H
    arm_mat_mult_f32(&KF->in.Kprev, &KF->system.H, &temp_mat_nna);
    // I - ans
    arm_mat_sub_f32(&In, &temp_mat_nna, &temp_mat_nnb);
    // P_k = ans * P_k
    arm_mat_mult_f32(&temp_mat_nnb, &KF->in.Pprev, &KF->in.Pprev);
}
```

B Adjusted Kalman Matrices

The following matrices describe the Kalman filter with position and force measurement. For the pure position measurement filter the second measurement is simply left out and the y, H and R matrices are cut down accordingly.

$$x = \begin{pmatrix} x_r & \dot{x}_r & \ddot{x}_r & f_r \end{pmatrix} \quad \text{(state)} \tag{B.1}$$

$$y = \begin{pmatrix} x_r & f_r \end{pmatrix} \quad \text{(measurement)} \tag{B.2}$$

$$A = \begin{pmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & dt & 0 \\ 0 & 0 & 0 & -\frac{1}{m_r} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(system matrix)} \tag{B.3}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(measurement matrix)} \tag{B.4}$$

$$Q = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \quad \text{(process noise covariance matrix)} \tag{B.5}$$

$$R = \begin{pmatrix} 0.001 & 0 \\ 0 & 10 \end{pmatrix} \quad \text{(measurement noise covariance matrix)} \tag{B.6}$$

Bibliography

[1] ARMin - Arm Rehabilitation. http://www.sms.hest.ethz.ch/research/arm_rehab.

[2] Balance Project - Objectives. http://balance-fp7.eu/objectives.php.

[3] EU BALANCE Project. http://balance-fp7.eu/index.php.

[4] Festo ExoHand. https://www.festo.com/group/de/cms/10233.htm.

[5] HAL - Hybrid Assistive Limb. http://www.cyberdyne.jp/english/products/HAL/index.html.

[6] Lokomat - Functional Robotic Gait Therapy. https://www.hocoma.com/world/en/products/lokomat/.

[7] Thiago Boaventura; Jonas Buchli. Acceleration-based Transparency Control Framework for Exoskeletons.

[8] Gene F. Franklin, David J. Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition, 2001.

[9] David A. Winter. *Biomechanics and Motor Control of Human Movement, 4th Edition*. John Wiley & Sons, Inc., 2009.