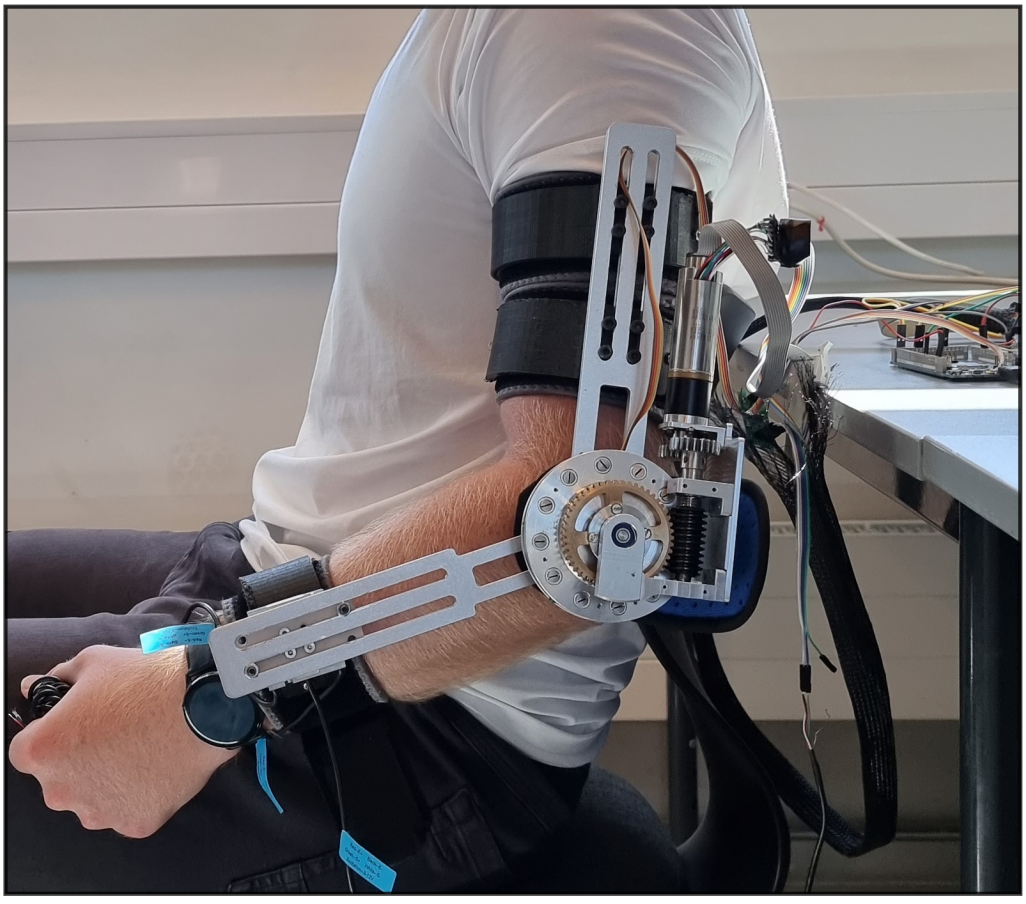
Physical human-robot interaction control of an assistive arm exoskeleton



Master Thesis

Andreas Pedersen
Marc Murcia I Matute
Electro-Mechanical System Design
Aalborg University

3rd of June 2021

AALBORG UNIVERSITY
STUDENT REPORT

Faculty of Engineering and Science
Electro-Mechanical System Design
Fibigerstræde 16
9000 Aalborg
http://www.tnb.aau.dk

Title:

Physical human-robot interaction control of an assistive arm exoskeleton

Project:

Master Thesis

Project period:

01-02-2021 to 03-6-2021

Project group:

EMSD4-grp6

Authors:

Andreas Pedersen
Marc Murcia i Matute

Supervisor:

Shaoping Bai
Per Johansen

Edition: 1st
Pages: 91
Appendices: 3
Completed: 02-06-2021

Synopsis:

Soft exoskeleton solutions are increasingly gaining importance due to their safety. In this project, two control strategies for a compliant exoskeleton with a variable stiffness actuator have been proposed: one regarding a payload attached to the exoskeleton and the other when the user grabs the payload. These control strategies are based on a developed human-robot interaction (HRI) model and can be split into low-level and high-level control. For the low-level control, a gain scheduling-based LQR controller is proposed. For the high-level control, an admittance filter-based method is used. The stability of the exoskeleton has been numerically studied based on the HRI model. Both control strategies have been implemented, and real test results are presented, which support the control strategies and can be considered a proof of concept. Moreover, a new 'sensorless' control strategy has been proposed in simulations.

Preface

This project is carried out by the 4^{th} semester Electro-Mechanical Systems Design master students in group EMSD4-grp6 at Aalborg University. The project was supervised by Professor Shaoping Bai and Associate Professor Per Johansen. The project spanned from the 1^{st} of February 2021 to the 3^{rd} of June 2021. Software used to complete the project includes MATLAB, Simulink, Arduino IDE, RealTerm, and SolidWorks. The report has been written in LaTeX.

The authors give thanks to the two supervisors for the guidance and assistance. A special thanks to Zhongyi Li, Ph.D., for designing and assembling the majority of the exoskeleton and his continued availability to the group. Also, a special thanks to Muhammad Raza Ul Islam, co-founder, BIOX Aps, for his assistance on the load cell setup, introduction to and instruction of the BIOX AAL band, and presentation of the payload estimation algorithm.

Reading guide

The report is written at the master level in engineering; therefore, familiarity with control theory, dynamics, mechanical modeling, and general engineering terminology is necessary. The main report can be read independently of the appendix, which solely provides additional material.

Harvard-style referencing is used in the report, where the author's surname and publication year are displayed in parenthesis as; (surname, year), for passive references, and as; surname (year), for active references. All figures and tables are numbered according to the chapter and the order in which they appear.

Time derivatives are written with the dot notation, i.e., the time derivative of the angle θ is written as $\dot{\theta}$. Matrices and vectors are written in bold, i.e., $\mathbf{M}$.

Resumé

Dette projekt er omhandlende kontrol af menneske-robot interaktion gennem et exoskelet monteret med en variabel stivheds mekanisme (VSM). Denne mekanisme er designet på Aalborg Universitet og har den karakteristika at dens stivhed er ikke-lineær. Dette komplicerer det såkaldt 'lavniveau' positionskontrol designet i og med system kan ændre sig så meget at kontrolleren ikke virker mere. Derfor er en ikke-lineær forstærkningsplanlægning LQR-kontroller blevet designet. Denne kontroller er afhængig af parameterestimering, hvilket er blevet gjort på baggrund af den mekaniske og dynamiske model af systemet. Denne parameterestimering er ikke optimal, da estimeringerne ikke er perfekte, men kan dog anvendes. Responset på den ikke-lineær kontroller er blevet implementeret i en Arduino DUE og er blevet testet. Det kan konkluderes at kontrolleren er overordnet set god, med en gennemsnitlig procentvis fejl på omkring 6%.

Positionskontrollen er dog kun halvdelen af projektet. For at kontrollere menneske-robot interaktion er en såkaldt 'højniveau' kontroller blevet designet. Denne kontrol tager et input fra in kraft sensor, der måler interaktionskraften mellem menneske og robot, og omdanner det – gennem den ønskede dynamik som mennesket skal føle – til en positionsreference som den ikke-lineær forstærkningsplanlægning LQR-kontroller har til opgave at tracke. Efter implementeringen af både højniveau og lavniveau kontrollerne, blev tre tests udført. I første test skal kontrolsystemet kompensere for exoskeletet selv, under forskellige belastnings niveauer, hvor mennesket gerne skulle føle ens belastning for begge tests. Det kan konkluderes at kontrolsystem kan håndtere denne opgave. I anden test skal robotten kun følge menneskets arm, uden at generere nogen form for hjælp - $\tau_{vsm} = 0$. Det kan konkluderes at kontrolsystemet er i stand til at operere i lavimpedans-mode. I den sidste test skal kontrol robotten hjælpe med en belasting placeret i menneskets hånd. Denne belastning er meget kompliceret at estimere, derfor er belastning antaget at være kendt og givet direkte. Et ønsket hjælpeniveau er givet til at være 0.73% af den totale belastning. Belastningen i test er vægten af menneskets arm, hvilket er estimeret. Grundet at denne estimering ikke er god – da vægten af armen ikke er kendt – er den reelle hjælp kun 0.5%, hvilket er en procentvis fejl på 28%. Ved at åbne en bedre estimering og en bedre positions sporing, det er postuleret at et bedre resultat kan opnås.

Det kan derfor konkluderes at den samlede kontrolalgoritme og indgående høj- og lavniveau kontroller, kan assistere et menneske gennem menneske-robot interaktion gennem det designet exoskelet og tilhørende VSM.

Nomenclature

Symbol	Description	Unit
b_h	Human damping	$[\frac{\text{Nms}}{\text{rad}}]$
c	Geometric variable	$[-]$
k_h	Human stiffness	$[\frac{\text{Nm}}{\text{rad}}]$
k_s	Spring stiffness	$[\frac{\text{N}}{\text{m}}]$
k_t	Motor torque constant	$[\frac{\text{Nm}}{\text{A}}]$
m_h	Human inertia	$[\text{kgm}^2]$
B_a	Admittance filter damping	$[\frac{\text{Nms}}{\text{rad}}]$
B_x	Viscous damping coefficient	$[\frac{\text{Nms}}{\text{rad}}]$
G_A	Admittance filter potential	$[\frac{\text{Nm}}{\text{rad}}]$
I	Motor current	[A]
J_1	Total Jacobian of the VSM	$[-]$
J_x	Moment of inertia	$[\text{kgm}^2]$
K_{vsm}	Equivalent stiffness of the VSM	$[\frac{\text{Nm}}{\text{rad}}]$
k_L	Linear stiffness	$[\frac{\text{Nm}}{\text{rad}}]$
M	Payload mass	[kg]
M_a	Admittance filter inertia	$[\text{kgm}^2]$
N	Configuration number	$[-]$
$P_{0,\,i}$	Initial or perimeter length section i	[m]
Y_v	Virtual dynamics	$[-]$
Z_h	Human impedance	$[-]$
α, β , ρ	Contact angles	[rad]
θ_d	Deflection angle	[rad]
$\theta_m^{(i)}$	Motor angle and its derivatives	$[\frac{\text{rad}}{\text{s}^{(\text{i})}}]$
$\theta_p^{(i)}$	Exoskeleton arm angle and its derivatives	$[\frac{\text{rad}}{\text{s}^{(\text{i})}}]$
θ_{ref}	Trajectory reference	[rad]
τ_{act}	Interaction torque	[Nm]
τ_h	Human torque	[Nm]
τ_{int}	Intrinsic torque	[Nm]
τ_m	Motor torque	[Nm]
τ_{pay}	Payload torque	[Nm]
τ_{vsm}	VSM torque	$[\frac{\text{Nm}}{\text{rad}}]$

Contents

1 Introduction

With the rapid developments in technology and increasing knowledge about health and the human body, the average lifespan of the human being is ever increasing. This means that societies around the world will contain an increasing number of older people. Age plays a significant factor in the overall health of the body. The older the body, the weaker and more susceptible the body is to diseases, strokes, and other health concerns. A stroke can significantly reduce the mobility and movement of the body and often includes a lengthy rehabilitation process (Medicine, 2021). Many developments in technologies such as exoskeletons have been made. These exoskeletons are designed to assist the user in many different ways depending on the user's issue. For example, the exoskeleton can provide an extra torque such that lifting things becomes more manageable. It can also help guide the user through motions to assist with a rehabilitation process or restrict a non-desired motion.

1.1 Application of Exoskeletons

The field of exoskeletons was first studied in the 1960s (Cloud, 1965). Since then, many developments have been made. There are many exoskeletons, from which three are outlined here: assistive, haptic, and rehabilitation.

1.1.1 Assistive

Assistive exoskeletons are used for persons with impaired physical abilities, such as decreased strength or tremor, or increasing strength and endurance of healthy persons. From this, assistive exoskeletons can be further subcategorized into two types: performance-enhancing and movement correction.

Performance-enhancing

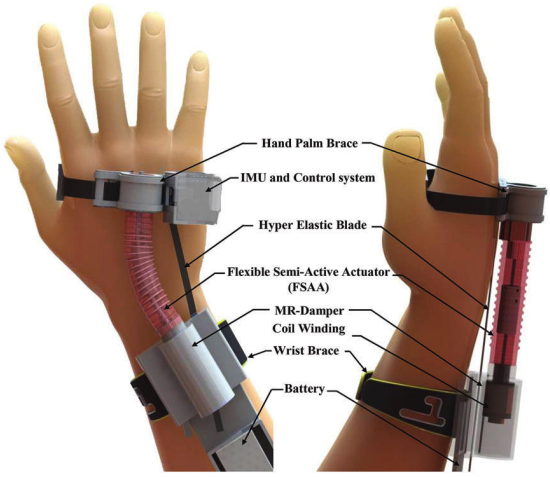
Performance-enhancing exoskeletons are developed to increase the total strength of the user. The user could either be an old or sick person, or a healthy person, whose job is very physically demanding. An example of the latter case is an exoskeleton developed to help workers who are exposed to heavy workloads daily (e.g., Ekso EVO (Ekso Bionics, 2021)). The robot enhances strength and endurance for carrying large loads, which eases the strain on the user. Another benefit is the distribution of load, which ensures better overall physical health of the joints and skeletal system.

Movement correction

Diseases or injuries can cause the movement control of an arm to be impaired. Loss or reduction in a person's control over their arm movements can often be in the form of tremors. Exoskeletons specifically designed for this can allow arm movement within a specific frequency range. If movements are made outside of the range, they will be dampened. This dampens the tremor frequencies - which are usually higher than typical movement frequencies - and the user can experience a more normal behavior of the arm. In Zahedi et al. (2020) a soft exoskeleton was designed to dampen the oscillations during a tremor. An illustration of the exoskeleton can be seen in Figure 1.1a.

1.1.2 Haptic

Haptic exoskeletons are used for many different things, such as VR for either games or rehabilitation or for controlling an off-site robot (teleoperation). The point is to give the user feedback of what the forces the robot would feel. For teleoperated exoskeletons, a master-slave configuration is usually used as in Letier et al. (2010). Here, the user is connected to an exoskeleton, and when a movement is made, the off-site robot makes the same movement. If the off-site robot hits an obstacle or experiences a force, the force is transferred back to the exoskeleton - in some scaled form - so the user can react. This gives a more human/natural movement of the robot. Haptics can also be utilized in a virtual environment; an example is Gupta and O'Malley (2006) in which an exoskeleton was designed to train the user in virtual tasks. The exoskeleton then guides the user through the movement and gives feedback based on the task. The exoskeleton can be seen in Figure 1.1b.



(a) Tremor damping exoskeleton (Zahedi et al., 2020)

(b) Haptic exoskeleton designed for virtual tasks (Gupta and O'Malley, 2006)

1.1.3 Rehabilitation

Rehabilitation exoskeletons are often used for individuals who have been in an accident, had a stroke (or similar), or had surgery. This individual might struggle to perform specific tasks. The robot would then act as a guide and help the user through the movements, helping them to relearn the movements and, e.g., give the means to walk around and gaining independence from a wheelchair (Gorgey, 2018).

1.1.4 Exoskeleton Developments at Aalborg University

Exoskeletons have been a focus at Aalborg for a couple of years, and recently this focus has been put into the development of soft exoskeletons - exoskeletons with compliant actuators. In 2019a, novel variable stiffness mechanisms (VSM) were developed by Li and Bai (2019). This VSM has the advantage that it can be made compact, as only one actuator is needed. This is due to the variability characteristics of the mechanisms; the stiffness is nonlinear, meaning that at small deflection angles, the stiffness is low and increases nonlinearly with the angle. The system is also reconfigurable such that it can be tuned to the use case. These reconfigurations are done while the power is off to increase compactness and decrease mass. The schematics of the mechanism can be seen in Figure 1.2.

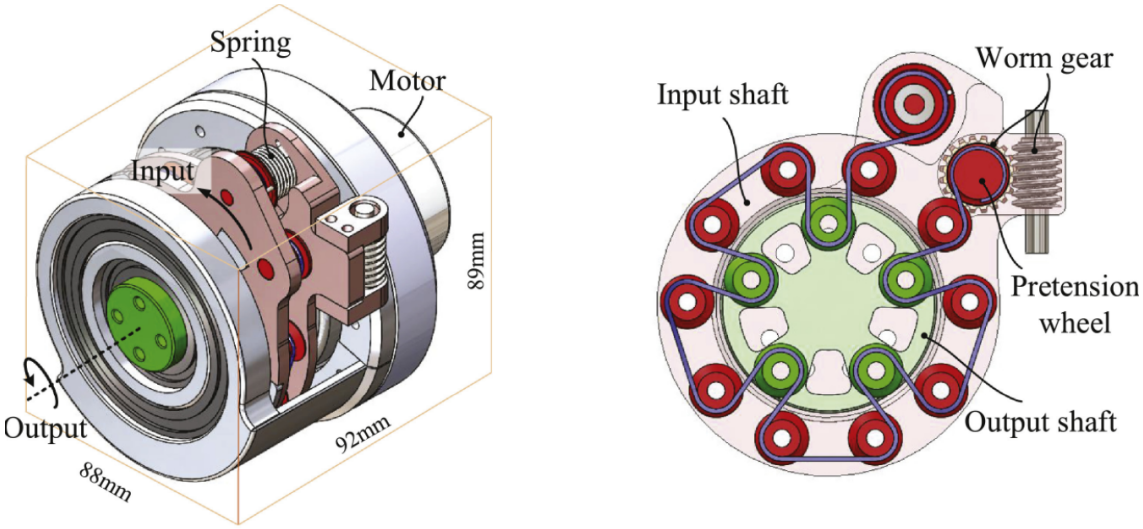


Figure 1.2: First iteration of the variable stiffness mechanism developed at Aalborg University (Li and Bai, 2019)

This VSM has since been developed further by Li et al. (2020) and a position control was designed in Pedersen and Murcia i Matute (2020). Latest, the design has been made more compact, where some of the reconfigurability has been removed - specifically the pretension wheel, and the linear spring has been replaced by polyurethane rubber bands. The newest design can be seen in Figure 1.3.

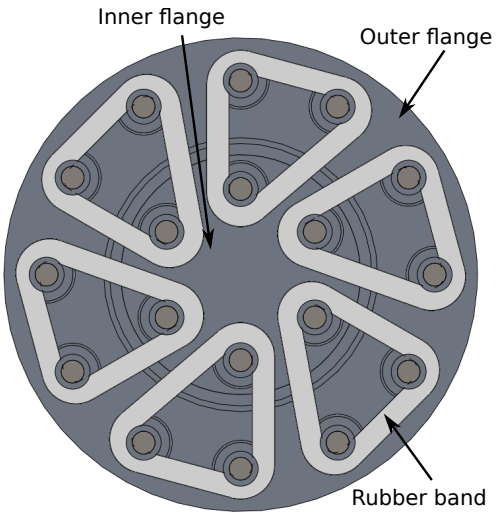


Figure 1.3: CAD model of the latest iteration of the variable stiffness actuator

This VSM has since been implemented in an exoskeleton, which is to be used in this project. The exoskeleton can be seen worn in Figure 1.4

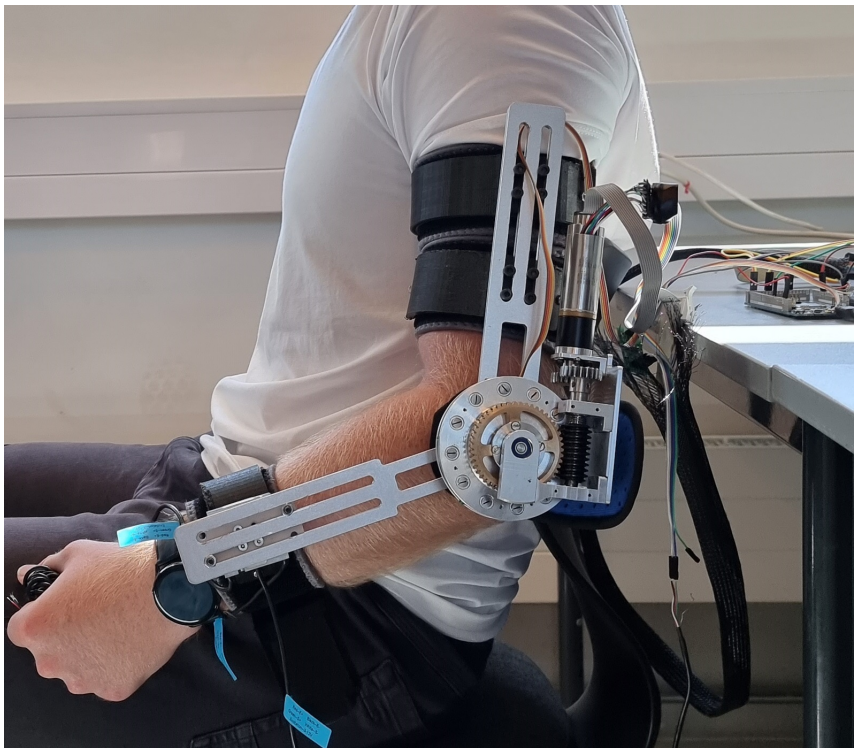


Figure 1.4: Exoskeleton worn by the user

The construction and modeling of the VSM will be described in more detail in Chapter 3.

1.2 Control of Exoskeleton: State of the Art

As this project focuses on the control aspect of exoskeletons, an analysis of the existing solutions to the control problem is made. Generally, the control problem of the exoskeleton can be broken down into a high-level and low-level control - hierarchical control. The high-level controller handles the interaction between the human and the exoskeleton, e.g., makes sure that the exoskeleton is not working against the human if that is not desired. The low-level controller handles the specific actuator and makes sure that the actuator produces the correct torque or speed, according to what the high-level controller is demanding.

1.2.1 Control Algorithm Classifications

There are many different ways to classify the control algorithms. One of these is dependent on the signal type used to control the exoskeleton. Three types are presented here: biological signals, non-biological signals, and platform-independent.

Biological signal

Biological signals are signals that originate directly from the human. Commonly, these signals are electromyography (EMG), which are electrical signals transmitted by motor neurons - neurons that control the muscle activity (Mayo Clinic, 2019). Based on these signals, a controller can be designed. However, due to a delay between the neurons demanding muscle activity and the actual production of force, the signal can be used to predict movements of the arm (Gopura and Kiguchi, 2009). This is called motion intention detection and can increase the performance of the controller. The most significant issues with an EMG-signal-based control are that the signals are user-dependent and can even change for the same user on a day-to-day basis. It is also sensitive skin conditions and very noisy, which can be an issue for low muscle activity. EMG-based motion intention detection is used in Gopura et al. (2009) where an impedance control was used to control the force interaction between the human and the exoskeleton.

Non-biological signal

Non-biological signals are based on signals resulting from a biological signal, such as motion or force/torque. The advantages of using non-biological signals are that they are easier to work with - the signal-to-noise ratio is better - and they are not dependent on skin conditions or similar. Several HRI control algorithms using interaction cuff sensors have been developed. An example of those is admittance and impedance controllers. For example, Huo et al. (2011) uses force sensing resistors (FSRs) to design an intention reaching direction algorithm (IRD), and an admittance control is used for interaction force control.

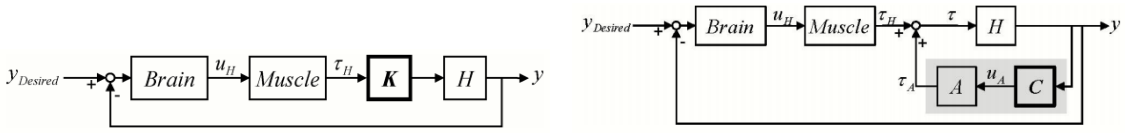
Platform independent

Platform independent controllers are independent on the type of signal - biological or non-biological. In Kong and Tomizuka (2009) a fictitious gain control was proposed. The goal is to change a gain parameter, K, such that:

- K = 1 $\rightarrow$ robot will just follow the arm

- K $>$ 1 $\rightarrow$ robot assist the movement
- K $<$ 1 $\rightarrow$ robot resists the movement

In Figure 1.5, a block diagram of the fictitious gain algorithm can be seen.



(a) Fictitious gain algorithm with gain, K (Kong and Tomizuka, 2009)

(b) Controller, C, and actuator, A (Kong and Tomizuka, 2009)

Figure 1.5: Block diagrams of the fictitious gain algorithm.

The fictitious gain, K, in Figure 1.5a is replaced with the controller, C, and actuator, A, in Figure 1.5b. The design goal is to set these two equal to each other and isolate for the controller. To do this, the inverse of the kinematics, H, is needed. This can be difficult to obtain. Therefore, a torque estimation is proposed. This torque estimation can be obtained from, e.g., EMG sensors, muscle fiber expansion sensors, or muscle hardness sensors. Hence, this controller is platform-independent.

1.2.2 Interaction Force Control

Interaction force control is concerned with the interaction force between the exoskeleton and the human. The main goal is to control the human-robot interaction (HRI) forces to obtain the desired assistance. A force sensing interaction cuff is often used to measure the interaction force - which is a non-biological signal. This interaction cuff is attached to the exoskeleton, and the user is wearing it around the forearm. If the force interaction is not zero, then this interaction cuff will be able to detect it. The interaction force controller is usually an admittance or impedance controller. These controllers act as high-level controllers that generate the reference for the low-level controller based on the user input.

Impedance control

The impedance controller relates the *movements* imposed by the user to a force - which is used to manipulate the exoskeleton through a low-level torque control. In Li et al. (2017) an adaptive impedance control is proposed, where the parameters of the controller are updated using a neural network.

Admittance control

The admittance controller is opposite to the impedance controller, as it relates the *force* produced by the user to an equivalent movement - which is used to manipulate the exoskeleton through a low-level position/velocity control. An admittance control is proposed in Huo et al. (2011), which is improved by an intentional reaching direction algorithm.

1.2.3 Motion Intention Detection

Motion intention detection is commonly used to predict the movement of the arm. This can improve the performance of the high-level controller as more information on how the user will behave is known. This can take a lot of different shapes, but machine learning is commonly used to treat the signals and predict the user motion intention (Huo et al., 2011).

1.2.4 Low-level Control

The low-level controller is not the main focus of this project, but it is still essential. In Pedersen and Murcia i Matute (2020) low-level controllers for a novel variable stiffness actuator - to be used in this project - were proposed. Both linear and nonlinear controllers were proposed in the paper, where the nonlinear controllers showed to be a viable solution. The controllers were designed for position control, which means that the high-level controller has to convert a desired force/torque into a position, which will be used as input for the low-level controller. The low-level controller can also be a torque controller, in which case the high-level controller has to provide a desired force/torque - calculated from motion data - which will be the low-level controller's input.

1.3 Simple Anatomy of the Mechanics of the Human Arm

As this project will be concerned with controlling the arm, only the mechanics of the upper body extremities will be described. According to Gopura and Kiguchi (2009) the arm of a human, containing the shoulder, elbow, and wrist (excluding the fingers), can be described as a seven degrees of freedom system. These can be seen in Figure 1.6.

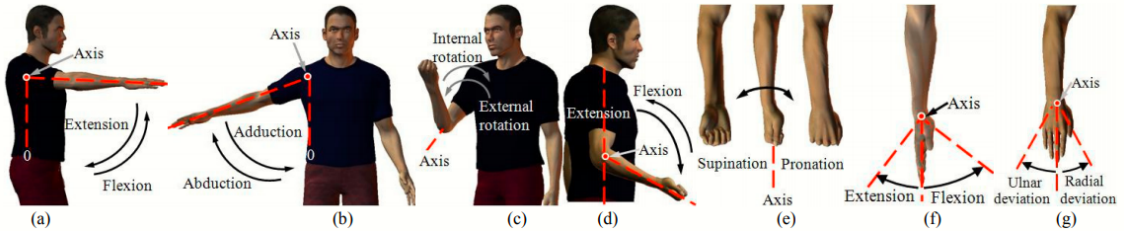


Figure 1.6: Illustration of the degrees of freedom of a human arm (Gopura and Kiguchi, 2009)

The shoulder is the most complex joint as it contributes 3 DoFs - extension/flexion (move the arm behind/in front of the body), adduction/abduction (move the arm from side to side, across the body), and internal/external rotation. The elbow has 2 DoFs - extension/flexion and supination/pronation - and the wrist also has 2 DoFs - extension/flexion and ulnar-/radial deviation.

1.4 Use Case Definition

Since there are many problems to solve - assistive, rehabilitation, haptics, etc. - a use case will be defined, which will be the project's focus.

The project will attempt to solve an assistive, performance-enhancing problem, where the user would need help to lift an object. Therefore, the control will be designed to remove some of the torque from the payload to ease the load on the user. This naturally limits the project, e.g., no trajectory will be provided, nor will any movements - such as tremor - be canceled. Interaction force controllers such as admittance or impedance could prove viable solutions, where the performance of both can be enhanced by utilizing motion intention detection (Huo et al., 2011).

1.5 Problem Formulation

From the foregone analysis, three types of exoskeletons were presented: assistive, rehabilitation, and haptic. Here assistive types can be subcategorized into performance-enhancing and movement correction. This project will deal with an assistive, performance-enhancing use case, where the user will be loaded with a payload. The project's goal is to design a controller that can solve this problem; hence the following problem statement is formulated:

"How can human-robot interaction using an exoskeleton arm with a variable stiffness mechanism be controlled such that the effect of a payload is reduced?"

System Requirements 2

In this chapter, the requirements for the system will be presented. These requirements will be used as a guideline for the design process and used to evaluate the tests.

Low-level control requirements

The low-level control requirements are concerning the position control, which is responsible for ensuring that the position of the exoskeleton arm is the same as the reference calculated from the high-level control. The requirements will be concerning the performance of the position control. Stability will be proved for specific cases rather than generally. Hence no requirements for stability will be listed. The requirements can be seen in Table 2.1.

Requirement	**Value**
Bandwidth	$> 2\pi \frac{\text{rad}}{s}$
Average error	0.05 rad
Average % error	5 %

Table 2.1: Low-level controller requirements

The error requirements are chosen based on the results obtained in Pedersen and Murcia i Matute (2020). The bandwidth is based on the expected maximum arm movement speed of the user.

High-level control requirements

The high-level control requirements are concerning the admittance filter and the whole system. These requirements will describe how much the exoskeleton should be helping. The requirements are divided into the tests which will be performed. The tests are described in more detail in Chapter 7. The requirements can be seen in Table 2.2.

Test	**Requirement**	**Value**
Robot compensation	Average τ_{act} error between tests	0.25 Nm
Low impedance	Average τ_{vsm} error	0.3 Nm
Assistive	Assist level error	10%

Table 2.2: High-level controller requirements

For the robot compensation test, the requirement is set to average τ_{act} error between tests. This requirement compares a test without payload and the payload, where the interaction torque should be the same for both tests.

Modeling of the System 3

In Li et al. (2020) a new Variable Stiffness Mechanism (VSM) for exoskeletons was proposed and developed. It was designed to be compact, light, and reconfigurable to be a standalone for different applications. In Pedersen and Murcia i Matute (2020), different position control strategies for the Variable Stiffness Actuator (VSA) were studied. This project is developed based on this previous work. Nevertheless, some changes have been made to the VSM, and some other parts have to be considered to represent the Human and Robot Interaction (HRI) properly.

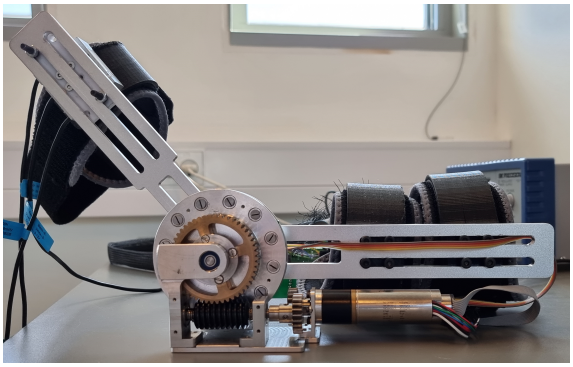


Figure 3.1: The exoskeleton with novel VSM attached

3.1 Human and Robot Interaction Model

Human and robot interactive systems, indeed exoskeletons, are not usually tested in numerical simulations. Instead, prototypes are built, and tests are carried out on the real system. This is due to the complexity of modeling the human body and its interaction with a robot. Nonetheless, if a model can be derived, different potential control algorithms can be numerically simulated without any danger, and stability analyses can be conducted.

The exoskeleton working in synergy with its user can be represented as in Figure 3.2. This idea has been used in Buerger and Hogan (2006) to evaluate the robot stability while interacting with an unknown environment, and by Keemink et al. (2018) for a stability study of HRI.

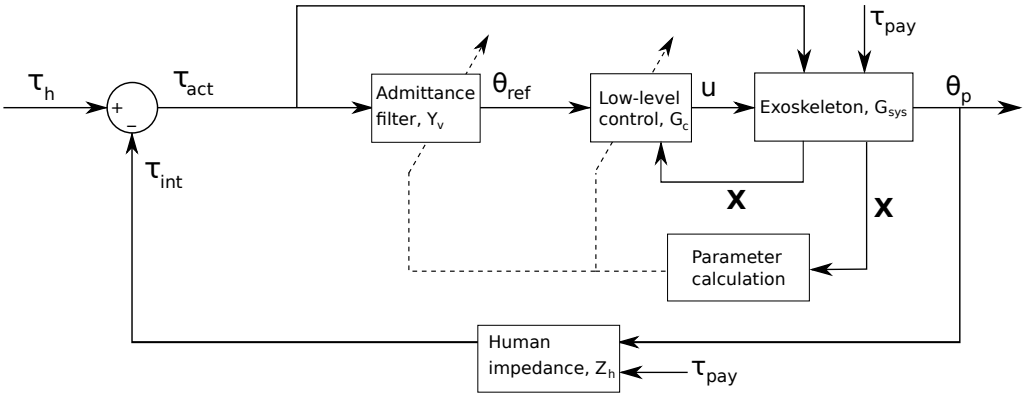


Figure 3.2: Schematic of the HRI model. Where τ_h is the voluntary torque applied by the human, τ_{int} is the torque generate by the human arm's intrinsic dynamics, τ_{act} is the interaction torque between the user and the exoskeleton measured by a sensor (which is transformed from force to torque), θ_{ref} is the position reference obtained from the admittance filter, u is the control input to the exoskeleton, θ_p is the exoskeleton, and arm position, $\mathbf{X}$ is the low-level control states, and τ_{pay} is the payload carried by the exoskeleton

The HRI model schematic comprises the exoskeleton model, the low-level control, the admittance filter, the parameters calculation, and the human impedance. The combination of the former two - exoskeleton model and low-level control - has a similar structure from the one used previously in Pedersen and Murcia i Matute (2020), except for the introduction of the interaction torque, τ_{act}.

The admittance filter is responsible for transforming the HRI torque, τ_{act}, to the reference for the low-level control, θ_{ref}. This filter, Y_v, describes how the user should feel the payload. The admittance filter can be either constant or variable, the former being simpler but having the disadvantage of not adapting to different conditions. The latter can adapt to different conditions. However, it needs parameter estimations - parameter calculation block.

These three different components describe the basic uncoupled standalone admittance-based control block diagram. By using this control strategy, the exoskeleton behaves as an admittance system, as its name implies. To obtain a closed-loop HRI interaction, an impedance human arm model, Z_h, is used in the feedback. The human impedance model represents the intrinsic dynamics of the human arm, τ_{int}, while being moved by the exoskeleton - passive forces due to the motion-induced on it and its applied payload. Additionally, the human can also produce a voluntary active torque, which cannot be easily modeled due to its high dependency on different factors as: desired trajectory, payload, and muscle anatomy. Instead, this is considered the input to the system. The subtraction of both is the interaction torque, τ_{act}, measured by a sensor, and represents the net human force exerted on the exoskeleton.

In the rest of this chapter, both the exoskeleton and human models are described in more detail.

3.2 Exoskeleton Model

This section concerns the exoskeleton model, which is marked in green in Figure 3.3.

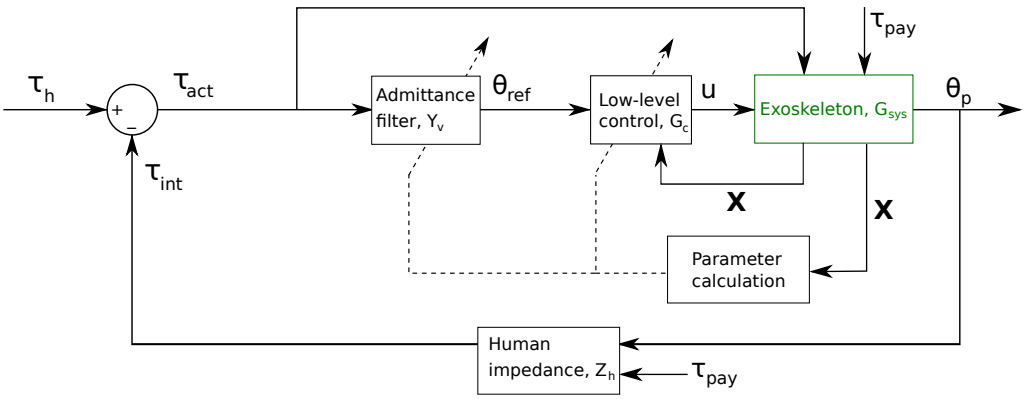
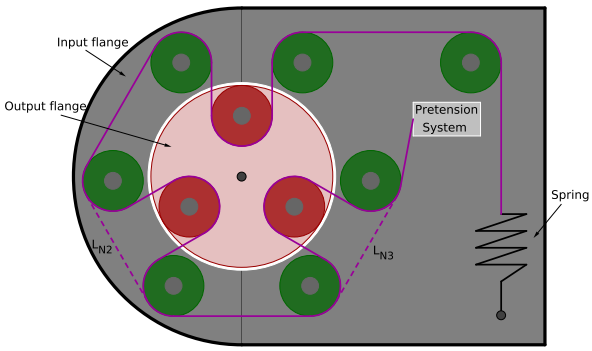


Figure 3.3: Entire system. The exoskeleton model is marked in green

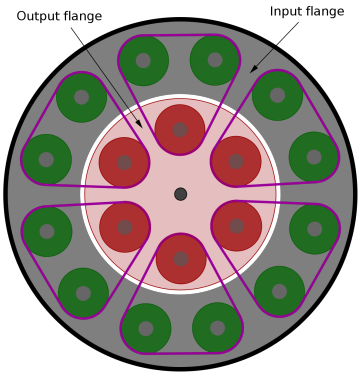
The exoskeleton model can be split into two sub-models: the Variable Stiffness Mechanism (VSM) stiffness model and the dynamic model. Both models have been previously considered in Li et al. (2020) and Pedersen and Murcia i Matute (2020). So only changes to its design and model will be considered.

3.2.1 Variable Stiffness Mechanism (VSM)

The Variable Stiffness Mechanism (VSM) design has been slightly changed, nevertheless the same concept has been exploited. In Figures 3.4a and 3.4b both design schemes can be seen.



(a) Previous VSM design (Pedersen and Murcia i Matute, 2020)



(b) New VSM design

Figure 3.4: Both VSM designs

Mechanical construction of the VSM

The main idea of the VSM is to make the connection between the human and the actuator compliant, as this gives the exoskeleton a more human feel. The VSM concept consists of an input/outer flange - connected to the actuator - and an output/inner flange - connected to the exoskeleton arm and the human, as seen in Figures 3.4a and 3.4b. The connection between these two flanges in the old VSM is achieved by using a linear spring

as a passive elastic element. It was connected to a wire, which went around pulleys on the inner and outer flange. This makes the connection between the link/human and the actuator compliant. Due to its geometrical configuration, the total stiffness of the VSM has a nonlinear behavior. As has been mentioned previously, the VSM was designed to be reconfigurable. The pretension of the system, the number of pulleys used, and the spring are used for this purpose and can be changed to change the torque and stiffness characteristics and adapt for different applications.

For the new design, the spring-wire system has been modified to use rubber bands between the pulleys. By doing this, the system loses one of its reconfiguration parameters, as the pretension of the system cannot be adjusted. Nonetheless, the design is more compact. Moreover, to measure the pretension in the previous VSM, a sensor must be used - not included in the design. Additionally, more pulleys have been added to the new design, yielding higher reconfigurability and a higher possible total VSM stiffness and torque.

By doing these modifications the system can be made more compact without losing that much of its reconfigurability. Though, its a bit harder to reconfigure.

Stiffness & Torque Mathematical Models

In Pedersen and Murcia i Matute (2020) and Li et al. (2020), the stiffness model of the previous VSM was derived. Using a similar procedure, the new VSM's stiffness model can be derived.

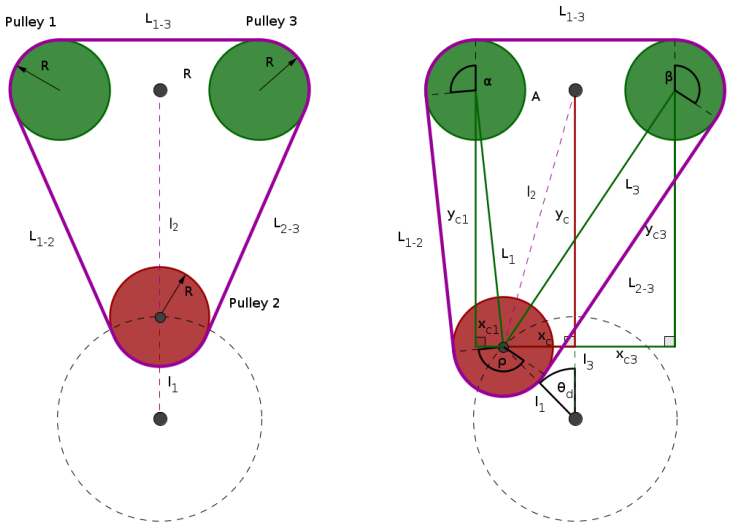


Figure 3.5: VSM section

The combination of two input flange pulleys (Pulley 1 - Pulley 3), an output flange pulley (Pulley 2), and the rubber band interconnecting the pulleys is called a section, see Figure 3.5. The total VSM torque τ_{vsm} can be described as the combination of the torque produced by the number of operative sections, N.

$$\tau_{vsm} = \tau_{section} N \tag{3.1}$$

Hence, reducing its derivation to one of the sections. The force procured by the rubber

band in a section depends on its elongation

$$F_{section} = k_s(P_i - P_0) + F_0 \tag{3.2}$$

where P_i is the length of the rubber band, P_0 is the length at zero elongation, F_0 is the pretension of the rubber band, and k_s is rubber band stiffness. From Figure 3.5 the band length can be determined

$$P_i = L_{1-3} + L_{1-2} + L_{2-3} + R(\alpha + \beta + \rho) \tag{3.3}$$

from which L_{1-3} is constant and the sum of the contact angles is $\alpha + \beta + \rho = 2\pi$. Making the rubber band elongation depending purely on L_{1-2} and L_{2-3}. Due to both input and output pulleys have the same radius, L_{1-2} and L_{2-3} are equal to L_1 and L_3 respectively, Figure 3.5.

$$\begin{aligned} L_{1-2} &= L_1 = \sqrt{{x_{c1}}^2 + {y_{c1}}^2} \\ L_{2-3} &= L_3 = \sqrt{{x_{c3}}^2 + {y_{c3}}^2} \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} x_{c1} &= \frac{L_{1-3}}{2} - l_1 \sin\theta_d & \qquad x_{c3} &= \frac{L_{1-3}}{2} + l_1 \sin\theta_d \\ y_{c1} &= l_3 - l_1 \cos\theta_d & \qquad y_{c3} &= l_3 - l_1 \cos\theta_d \end{aligned} \tag{3.5}$$

Then the section's torque can be given as

$$\tau_{section} = F_{section} J = F_{section} \frac{\partial P_i}{\partial \theta_d} \tag{3.6}$$

where J is the Jacobian and is the sum of distances that are perpendicular to the rubber band force. Given that only two components of the rubber band length elongate, due to deflection of the VSM, θ_d, Eq. (3.6) can be redefined as

$$\tau_{section} = F_{section} \frac{\partial L_{1-2}}{\partial \theta_d} + F_{section} \frac{\partial L_{2-3}}{\partial \theta_d} \tag{3.7}$$

where $\frac{\partial L_{1-2}}{\partial \theta_d}$, and $\frac{\partial L_{2-3}}{\partial \theta_d}$ are the perpendicular distances between the center of rotation of the output flange and the pulling force direction of each segment - L_{1-2} and L_{2-3}.

Combining Eqs. (3.1), (3.2) and (3.6) the total VSM torque equation is obtained.

$$\tau_{vsm} = k_s \Delta P J N + F_0 J N \tag{3.8}$$

where $\Delta P = P_i - P_0$.

The nonlinear stiffness of the VSM can then be determined using the principal of virtual work, differentiating the VSM torque, τ_{vsm}, over the deflection, θ_d

$$\begin{aligned} \delta\tau_{vsm} = K_{vsm}\delta\theta_d \quad \Rightarrow \quad \frac{\delta\tau_{vsm}}{\delta\theta_d} &= K_{vsm} \\ &= k_s \frac{\partial \Delta P}{\partial \theta_d} JN + k_s \Delta P \frac{\partial J}{\partial \theta_d} N + F_0 \frac{\partial J}{\partial \theta_d} N \\ &= k_s N (J^2 + \Delta P \frac{\partial J}{\partial \theta_d}) + F_0 \frac{\partial J}{\partial \theta_d} N \end{aligned} \tag{3.9}$$

The Jacobian, J, and its derivative, $\dot{J}$, can be seen in Appendix A.

Stiffness & Torque Characteristic Curves

Apart from the mathematical model, a test to determine the VSM torque and stiffness has been conducted by Zhongyi Li. The test consisted on measuring the exerted torque by the VSM at different deflections, θ_d. The VSM used in the exoskeleton has the following parameters seen in Table 3.1.

Parameter	Value	Unit
L_{1-3}	0.0176	[m]
l_1	0.015	[m]
l_3	0.034	[m]
R	0.004	[m]
k_s	4	$[\frac{\text{N}}{\text{mm}}]$
N	6	[-]
F_0	7	[N]

Table 3.1: VSM parameters

The rubber band pretension, F_0, in the real system was unknown and has been determined so that a good fit between the mathematical model and the test curve is obtained at zero deflection. At which only the pretension affects the VSM stiffness, Pedersen and Murcia i Matute (2020).

From the data collected in the test, the torque curve can be obtained, see Figure 3.6. This data has been fitted into a polynomial, describing the VSM torque, τ_{vsm}, as

$$\tau_{vsm} = 15.13\ {\theta_d}^3 + 2.319 \cdot 10^{-16}\ {\theta_d}^2 + 1.596\ \theta_d - 5.151 \cdot 10^{-17} \tag{3.10}$$

deriving over the deflection, as in Eq. (3.9), the stiffness of the experimental test is calculated to

$$K_{vsm} = 45.39\ {\theta_d}^2 + 4.6380 \cdot 10^{-16}\ \theta_d + 1.596 \tag{3.11}$$

In Figure 3.6 the torque and stiffness curves are displayed. Where the fitting polynomial, and the mathematical model are presented for both the torque and stiffness.

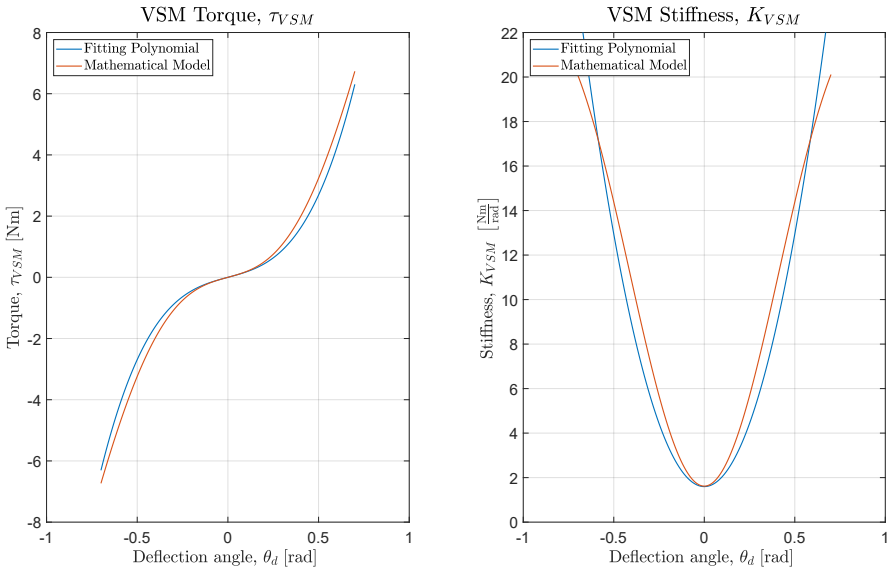


Figure 3.6: Characteristic curves for the VSM torque and stiffness

From this comparison, it can be concluded that the developed mathematical model is a fair approximation of the real system. Nevertheless, a slight difference between the mathematical model and real system is exposed. The mathematical model is always displaying a slightly higher stiffness and consequently higher torque. This is due to the worm gear, which at small deflections can move horizontally. This lowers the overall stiffness of the real system, which is also in the figures. Nevertheless, after a certain deflection, the worm gear reaches its horizontal limit and cannot move further. Then the system stiffness purely comes from the VSM, but with a small amount of deflection already present in the system, which is not modeled. For this reason, the experimental torque and stiffness are expected to be of a slightly lower magnitude for the same deflection angle.

In the rest of the report, the two fitting polynomials Eqs. (3.10) and (3.11) are used if it is not otherwise stated. This is due to it is a better representation of the real system. Although, the mathematical model provides opportunities to test for different configurations.

3.2.2 Dynamic Model

The dynamic model of the exoskeleton can be easily modeled as in Li et al. (2020) and Pedersen and Murcia i Matute (2020). With the small addition of the work produced by the human user on the exoskeleton, τ_{act}.

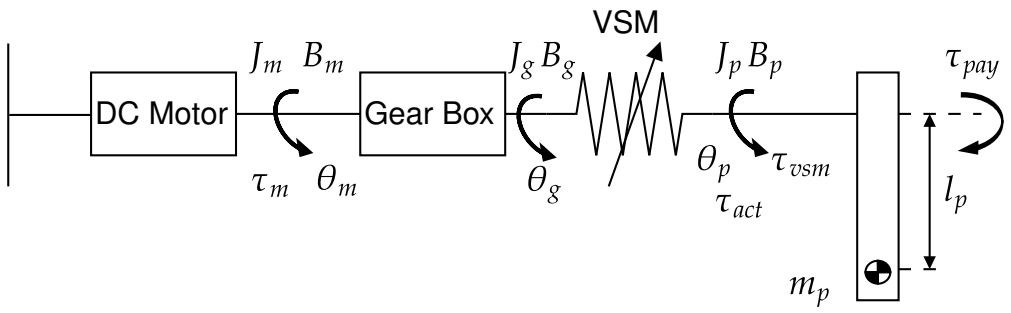


Figure 3.7: Exoskeleton diagram

The VSM links the motor to the exoskeleton forearm, transforming it to a variable stiffness actuator (VSA). Therefore, the dynamic model comprises two equations of motion, one for the motor side and one for the exoskeleton forearm.

The motor side equation of motion do not change compared to Pedersen and Murcia i Matute (2020), and is stated again here

$$\tau_m = J_{tot}\ddot{\theta}_m + (B_m + \beta B_{vsm})\dot{\theta}_m + T_{c,g}\text{sign}(\dot{\theta}_\text{m}) + n_{tot}\beta\tau_{vsm}(\theta_d) \tag{3.12}$$

where

$$J_{tot} = J_m + \beta J_{vsm} + \beta J_{mg} + \beta_1 J_{wg} + \beta J_{ww} \tag{3.13}$$

the different variables' descriptions can be seen in Table 3.2.

Variable	Description
τ_m	Torque output of the motor
$\tau_{vsm}(\theta_d)$	Torque output of the VSM
J_{tot}	Total inertia of the system
J_m	Inertia of the motor
J_{vsm}	Inertia of the VSM
J_{mg}	Inertia of the motor's gearbox
J_{wg}	Inertia of the worm gear's small gear
J_{ww}	Inertia of the worm gear's large gear
$\ddot{\theta}_m$, $\dot{\theta}_m$, θ_m	Acceleration, velocity, and position of the motor
β_1, β	Gear ratio and efficiency of motor gear box and worm gear
n_{tot}, n_1	Total gear ratio, and gear ratio of the motor gearbox
B_m, B_{vsm}	Damping of motor and VSM
$T_{c,g}$	Coulomb friction of the gearbox

Table 3.2: Description of variables in Eqs. (3.12) and (3.13) (Pedersen and Murcia i Matute, 2020)

The exoskeleton forearm is affected by force applied by its user, τ_{act}. And then the governing exoskeleton torque is the result of the sum of the HRI torque, τ_{act}, and the VSM torque, τ_{vsm}, in that way, its equation of motion can be defined as

$$\tau_{vsm}(\theta_d) + \tau_{act} = J_p\ddot{\theta}_p + (B_{vsm} + B_p)\dot{\theta}_p + T_{c,p}\text{sign}(\dot{\theta}_\text{p}) + \tau_{pay} \tag{3.14}$$

the different variables descriptions can be seen in Table 3.3.

Variable	Description
$\tau_{vsm}(\theta_d)$	Torque output of the VSM
τ_{pay}	Torque produced by a payload
τ_{act}	Torque produced by the user on the exoskeleton
J_p	Inertia of the exoskeleton forearm
$\ddot{\theta}_p$, $\dot{\theta}_p$, θ_p	Acceleration, velocity, and position of the exoskeleton forearm
B_{vsm}, B_p	Damping of VSM and exoskeleton forearm
$T_{c,p}$	Coulomb friction

Table 3.3: Description of variables in Eq. (3.14)

3.2.3 Model Verification

Although the dynamic model is almost the same as in Li et al. (2020) and Pedersen and Murcia i Matute (2020), the values of some of the parameters might differ. Therefore, a verification of the model and parameters will be made. To verify the model, a series of tests on the real system have been conducted. The tests are split into two; a set of free vibrations tests used to verify the exoskeleton arm dynamics and a set of motor input tests used to verify the motor side dynamics. In each set, two tests have been made; one without payload and one with payload. The payload is placed at the end of the exoskeleton forearm as a dead weight. The test setup is described in more detail in Chapter 6.

For the free vibrations tests, the exoskeleton arm was held such that it was perpendicular to the ground. Then the arm was moved to an arbitrary position and released. The dynamics are then measured using an absolute encoder. For the motor input tests the motor was given an input of: $u = 1$ A and $u = 0.25$ A for a payloads of $M = 1.1$ kg and $M = 0$ kg, respectively. Test conditions for all four tests were then reconstructed in simulation. The parameters are then altered using a trial-and-error method until a fit for all four tests is achieved simultaneously. The tests and simulations can be seen in Figure 3.8.

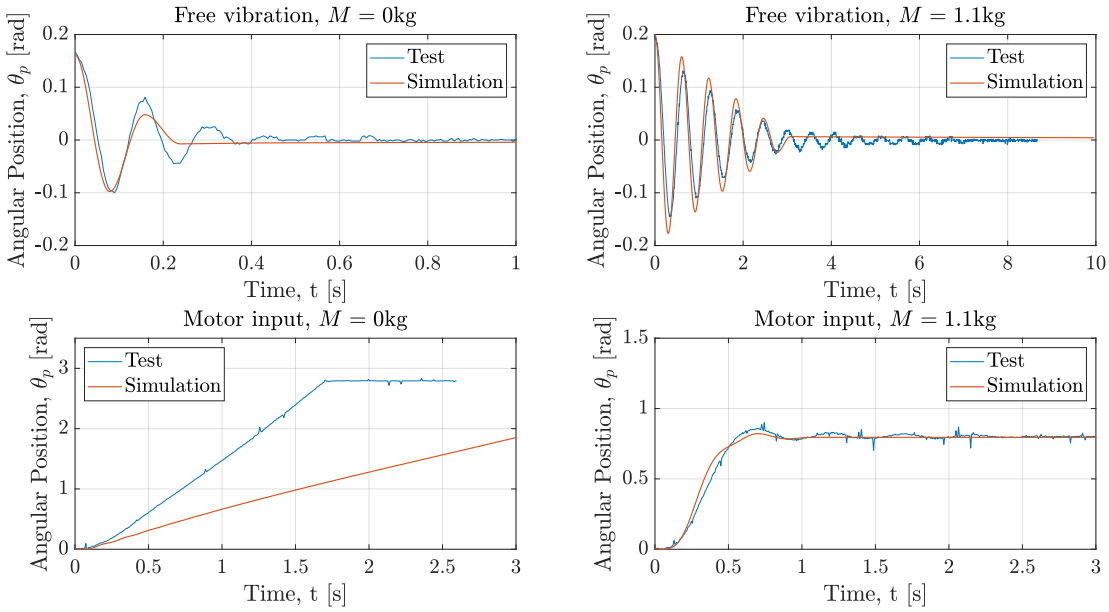


Figure 3.8: Tests and simulations used for verifying the model and parameters

From the figures, it can be seen that the fits are generally good. However, it can be seen that the friction seems too high for the motor input and free vibrations test for a payload of $M = 0$ kg. Decreasing the friction (or damping) results in much better fits for these two tests - which in turn worsens the fits for a payload of $M = 1.1$ kg. The reason for this is the friction behavior and construction of the worm gear. The worm gear consists of two parts, a spur gear, and a worm screw. The screw is attached to a rod, which connects to the frame through two bearings. These bearings sit loosely in the frame, so to stop the worm screw from hitting the frame, two small springs are used as spacers. This means that the worm screw is moving back and forth when torque is applied - as also mentioned in Section 3.2.1. This movement means that the friction behavior is changed, as not all torque is converted to friction through the self-locking characteristic of the worm gear. For low payloads, this has a fairly large effect, which is seen in Figure 3.8. This also explains why the final vibrations for the $M = 1.1$ kg test cannot be captured. However, as the effect diminishes already at $M = 1.1$ kg, the fits are deemed sufficient. The new parameters can be seen in Table 3.4.

Motor parameters	
Parameter	**Value**
J_{tot}	$1.5702 \cdot 10^{-6}$ kgm^2
J_m	$1.108 \cdot 10^{-6}$ kgm^2
J_{VSM}	$2.7 \cdot 10^{-5}$ kgm^2
J_{mg}	$4.8 \cdot 10^{-6}$ kgm^2
J_{wg}	$4.56 \cdot 10^{-6}$ kgm^2
J_{ww}	$2.4 \cdot 10^{-5}$ kgm^2
n_1	5.4
n_{tot}	270
η_{wg}	0.19
η_{mg}	0.7
$T_{c,g}$	0.00191 Nm
B_m	$1.86 \cdot 10^{-5} \frac{\text{Nms}}{\text{rad}}$
B_{VSM}	0.012 $\frac{\text{Nms}}{\text{rad}}$

Link side parameters	
Parameter	**Value**
l_M	0.185 m
l_p	0.2 m
m_p	0.07493 kg
J_p	0.0013 kgm^2
g	9.82 $\frac{\text{m}}{\text{s}^2}$
$T_{c,p}$	0.033 Nm
B_p	0 $\frac{\text{Nms}}{\text{rad}}$

Table 3.4: Parameters used in the model.

3.3 Human Model

To complete the HRI model the intrinsic dynamic of the human arm has to be derived as an impedance human model, Z_h. This is represented in Figure 3.9

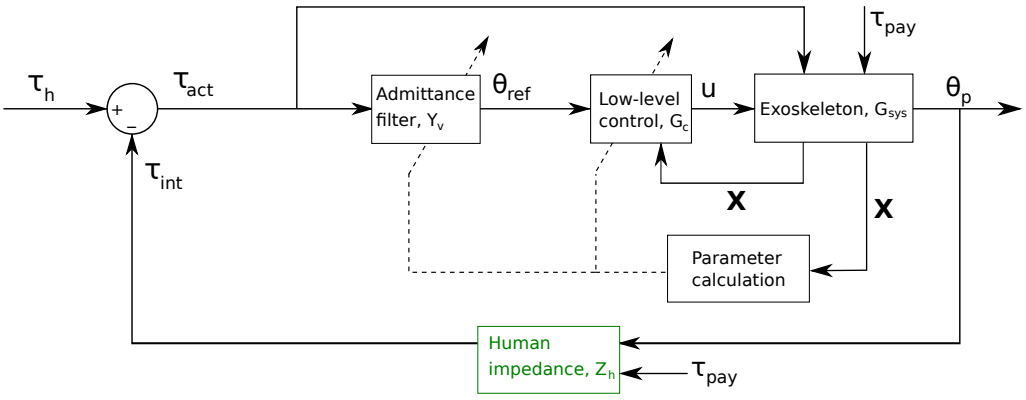


Figure 3.9: Entire system. The human impedance model is marked in green

There are multiple ways to obtain a human arm model, depending on the accuracy needed. For this use case, an accurate human model is not necessary, as discussed further in the

project. The human arm can then be modeled as a set of rigid bodies with 7-DoF, see Figure 3.10.

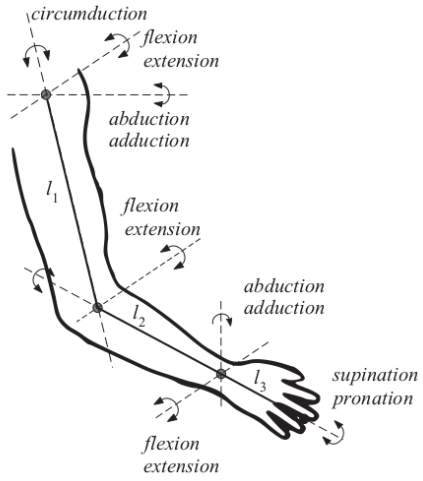


Figure 3.10: Human arm DoF (Pons, 2008)

Nonetheless, from this 7-DoF, only two are significant for the scope of this project. These are flexion and extension of the shoulder and elbow (the hand has been neglected). Thus, the model has been reduced to 2-DoF, from which the elbow equation of motion is desired. Because of assuming the human arm as a set of rigid bodies, tools to describe the kinematics of a rigid robot as the Denavit–Hartenberg convention can be used so that the Euler-Lagrange equation can be used to determine the equations of motion describing the human arm. However, first, the different human parameters needed will be presented.

Human Parameters

The human arm parameters depend on the user wearing the exoskeleton. For this reason, the whole impedance arm model will be user-dependent. This disadvantage is taken into consideration in Chapter 5 so that the exoskeleton can work for different users. The necessary parameters used in the development of the equations of motion are the forearm mass, FA_m, the upper arm mass, UA_m, the forearm length, FA, the upper arm length, UA, and their center of masses, which are obtained from Winter (2009). The inertias of the different bodies are determined using the radius of gyration, RoG, as in Krishnan et al. (2016), and are defined as

$$\begin{aligned} \mathbf{I_1} &= \begin{bmatrix} R_{UL}^2 & 0 & 0 \\ 0 & R_{US}^2 & 0 \\ 0 & 0 & R_{UF}^2 \end{bmatrix} UA_m \\ \mathbf{I_2} &= \begin{bmatrix} R_{FL}^2 & 0 & 0 \\ 0 & R_{FS}^2 & 0 \\ 0 & 0 & R_{FF}^2 \end{bmatrix} FA_m \end{aligned} \tag{3.15}$$

where R_{UL}, R_{US}, and R_{UF} are the upper arm longitudinal, sagittal, and frontal radius of gyration. R_{FL}, R_{FS}, and R_{FF} are the respective forearm radius of gyration.

	Mass $[kg]$	Length $[m]$	Center of Mass $[m]$		Radius of gyration $[m]$		
			Proximal	Distal	Sagittal	Frontal	Longitudinal
Forearm	$0.016B_m$	$0.146B_h$	$0.43FA$	$0.57FA$	$0.295FA$	$0.284FA$	$0.13FA$
Upper arm	$0.028B_m$	$0.186B_h$	$0.436UA$	$0.564UA$	$0.328UA$	$0.31UA$	$0.182UA$

Table 3.5: Human parameters, where B_m and B_h are the body mass and height respectively.

Denavit-Hartenberg Parameters

The coordinate frames of the different joints and end-effector can be seen in Figure 3.11

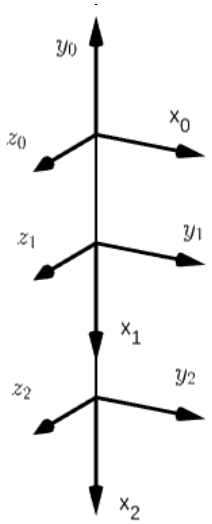


Figure 3.11: Rigid body & coordinate frames

The link and joint parameters of the defined solid rigid body are

Link	**Joint angle,** θ_i	**Joint distance,** d_i	**Link length,** a_i	**Link twist angle,** α_i
1	$\theta_1 - \frac{\pi}{2}$	0	UA	0
2	θ_2	0	FA	0

Table 3.6: D-H parameters

From these parameters, the different transformation matrices are determined, as well as the total transformation matrix between the origin and the end-effector - the wrist.

$$\mathbf{T_{01}} = \begin{bmatrix} \sin\theta_1 & \cos\theta_1 & 0 & UA\sin\theta_1 \\ -\cos\theta_1 & \sin\theta_1 & 0 & -UA\cos\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$\mathbf{T_{12}} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & FA\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & FA\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

$$\mathbf{T_{02}} = \mathbf{T_{01}}\mathbf{T_{12}} = \begin{bmatrix} \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & FA\sin(\theta_1+\theta_2)+UA\sin\theta_1 \\ -\cos\theta_1+\theta_2 & \sin\theta_1+\theta_2 & 0 & -FA\cos\theta_1+\theta_2-UA\cos(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

where UA and FA are the upper arm and forearm lengths, respectively.

The center of masses of the different rigid bodies are necessary to calculate the arm's generalized equation of motion. The forearm and upper arm center of masses were obtained from Winter (2009), in which these lengths are approximated relative to the human height. These CoMs must be determined with respect to the origin of the system, o_0.

$$\mathbf{cm_{01}} = \mathbf{T_{01}} \begin{bmatrix} -UA_{cm} \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta_1(\Delta UA) \\ -\cos\theta_1(\Delta UA) \\ 0 \\ 1 \end{bmatrix} \tag{3.19}$$

$$\mathbf{cm_{02}} = \mathbf{T_{02}} \begin{bmatrix} -FA_{cm} \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \Delta FA\sin(\theta_1+\theta_2)+UA\sin\theta_1 \\ -\Delta FA\cos(\theta_1+\theta_2)-UA\cos\theta_1 \\ 0 \\ 1 \end{bmatrix} \tag{3.20}$$

from which $\Delta UA = UA - UA_{cm}$ and $\Delta FA = FA - FA_{cm}$.

The full Jacobian of the different center of masses over the joint variables $\mathbf{q} = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T$ are defined as

$$\mathbf{J_{cm,1}} = \begin{bmatrix} \frac{\partial \mathbf{cm_{01}}}{\partial \theta_1} & \frac{\partial \mathbf{cm_{01}}}{\partial \theta_2} \\ \frac{\partial \omega_{\mathbf{01}}}{\partial \theta_1} & \frac{\partial \omega_{\mathbf{01}}}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} \Delta UA\cos\theta_1 & 0 \\ \Delta UA\sin\theta_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.21}$$

$$\mathbf{J_{cm,1}} = \begin{bmatrix} \Delta FA\cos(\theta_1+\theta_2) + UA\cos\theta_1 & \Delta FA\cos(\theta_1+\theta_2) \\ \Delta FA\sin(\theta_1+\theta_2) + UA\sin\theta_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{3.22}$$

where $\omega_{\mathbf{01}}$ and $\omega_{\mathbf{02}}$ are the angular velocities of the different bodies, being θ_1 and $\theta_1 + \theta_2$, respectively.

The last component needed to compute the Euler-Langrange equation and derive the human arm impedance model are the mass matrices

$$\mathbf{M_1} = \begin{bmatrix} UA_m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I_1} \end{bmatrix} \tag{3.23}$$

$$\mathbf{M_2} = \begin{bmatrix} FA_m\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I_2} \end{bmatrix} \tag{3.24}$$

where $\mathbf{I}$ is a 3×3 identity matrix, $\mathbf{0}$ is a 3×3 zero matrix, and I_1 and I_2 are the inertia matrices of each rigid body, respectively, which can be seen in Eq. (3.15).

Euler-Lagrange

With all the necessary information of the system, the kinetic and potential energy of the human arm can be established.

$$\kappa = \frac{1}{2}\dot{\mathbf{q}}^T(\underbrace{\mathbf{J_{cm1}}^T\mathbf{M_1}\mathbf{J_{cm1}} + \mathbf{J_{cm2}}^T\mathbf{M_2}\mathbf{J_{cm2}}}_{\mathbf{X}})\dot{\mathbf{q}} \tag{3.25}$$

$$P = \mathbf{g}^T(UA_m\mathbf{cm_{01}} + FA_m\mathbf{cm_{02}}) \tag{3.26}$$

where g is the gravity term and is given as: $\mathbf{g} = \begin{bmatrix} 0 & 9.81 & 0 \end{bmatrix}^T$.

Now the Euler-Lagrange equation can be established - in which the Lagrangian is defined as $L = \kappa - P$ - and the equations of motion derived

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} &= \frac{d}{dt}\left(\frac{\partial \kappa}{\partial \dot{\mathbf{q}}} + \cancelto{0}{\frac{\partial P}{\partial \dot{\mathbf{q}}}}\right) - \frac{\partial \kappa}{\partial \mathbf{q}} + \frac{\partial P}{\partial \mathbf{q}} \\ &= \frac{d}{dt}\mathbf{X}\dot{\mathbf{q}} - \frac{\partial \kappa}{\partial \mathbf{q}} + \frac{\partial P}{\partial \mathbf{q}} = \mathbf{X}\ddot{\mathbf{q}} + \underbrace{\dot{\mathbf{X}}\dot{\mathbf{q}} - \frac{\partial \kappa}{\partial \mathbf{q}}}_{\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})} + \underbrace{\frac{\partial P}{\partial \mathbf{q}}}_{\mathbf{G}(\mathbf{q})} \end{aligned} \tag{3.27}$$

Thus, the equation of motion can be split into three terms. The first one is the Mass-Inertia matrix, $\mathbf{X}$, which is defined as

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \tag{3.28}$$

and its components are

$$X_{11} = j_{1z} + j_{2z} + FA_m\Delta FA^2 + FA_mUA^2 + UA_m\Delta UA^2 + 2\Delta FAFA_mUA\cos\theta_2 \tag{3.29}$$

$$X_{12} = X_{21} = j_{2z} + FA_m\Delta FA^2 + \Delta FAFA_mUA\cos\theta_2 \tag{3.30}$$

$$X_{22} = j_{2z} + FA_m\Delta FA^2 \tag{3.31}$$

As can be seen in Figure 3.12, inertias are coupled between the different rigid bodies, where the shoulder acceleration, $\ddot{\theta}_1$, has a higher impact,than the elbow acceleration, $\ddot{\theta}_2$, on the elbow's equation of motion.

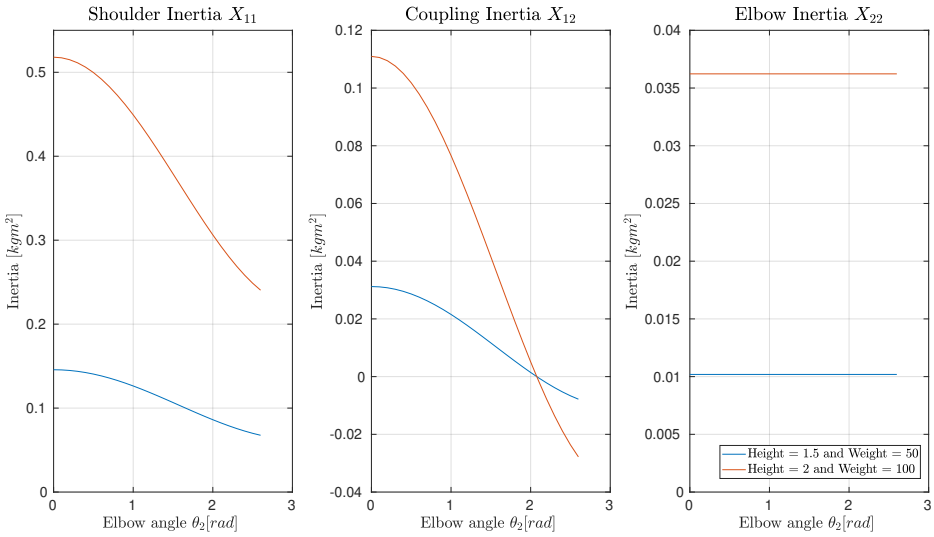


Figure 3.12: Mass-Inertia matrix, at different configurations and for two different humans

The second term is the the Coriolis and Centrifugal force, $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, which is

$$\mathbf{B}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} -FA_mUA\dot{\theta}_2\Delta FA\left(2\dot{\theta}_1 + \dot{\theta}_2\right)\sin\theta_2 \\ FA_mUA\dot{\theta}_1^2\Delta FA\sin\theta_2 \end{bmatrix} \tag{3.32}$$

From Figure 3.13 and Eq. (3.32) it can be seen that the elbow only experiences centrifugal forces, due to the shoulder rotation. Meanwhile, the shoulder has both Centrifugal and Coriolis forces. Once again that is due to the simplification of the human arm model. By adding the hand as a rigid body to the model, the elbow will also be affected by centrifugal forces. Moreover, the muscle has not been modeled, doing so would add some damping into the system.

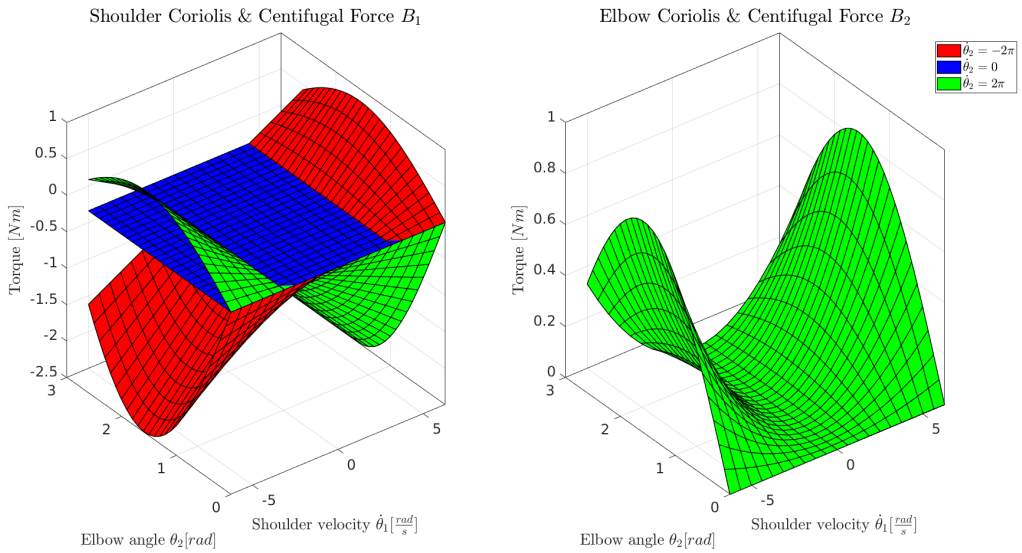


Figure 3.13: Centrifugal & Coriolis Force at different elbow velocities. In the right figure the three cases are exactly the same, since B_2 is not dependent on $\dot{\theta}_2$, as seen in Eq. (3.32)

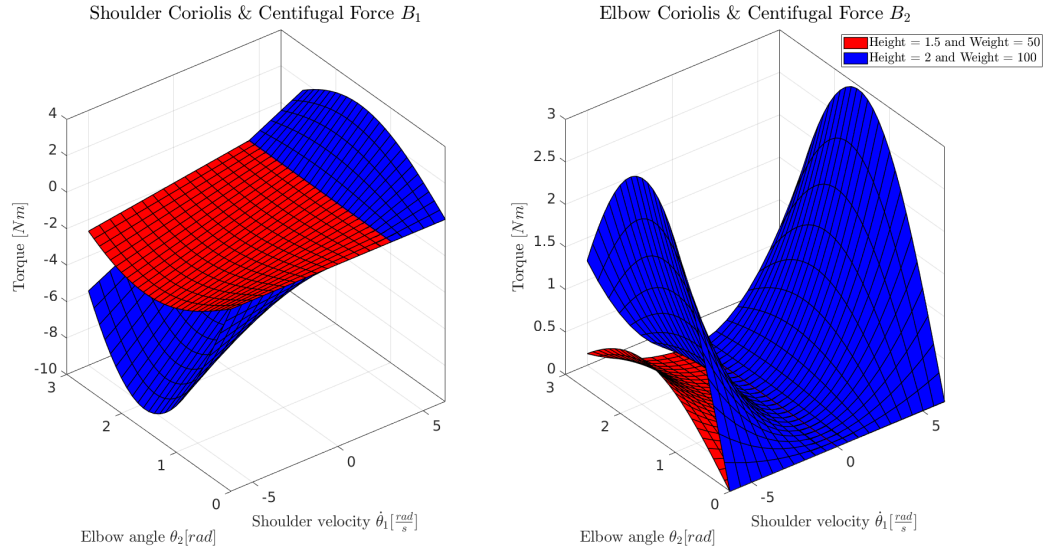


Figure 3.14: Centrifugal & Coriolis force for different users

The last term of the human arm equation of motion is the Gravitational force, $\mathbf{G}(\mathbf{q})$

$$\mathbf{G}(\mathbf{q}) = \begin{bmatrix} FA_m g\left(\Delta FA \sin\left(\theta_1 + \theta_2\right) + UA \sin\left(\theta_1\right)\right) + UA_m g \Delta UA \sin\theta_1 \\ FA_m g \Delta FA \sin\left(\theta_1 + \theta_2\right) \end{bmatrix} \tag{3.33}$$

The flexion and extension of the shoulder and elbow are the main contributors to the gravitational term, together with the abduction and adduction of the shoulder, which has not been modeled. Nonetheless, its impact on the system is similar to flexion and extension of the shoulder, considering it a reasonable model reduction. As for $\mathbf{B}(\mathbf{q},\dot{\mathbf{q}})$, the arm muscles can introduce some stiffness into the system, as those behave as a variable damping-spring mechanism.

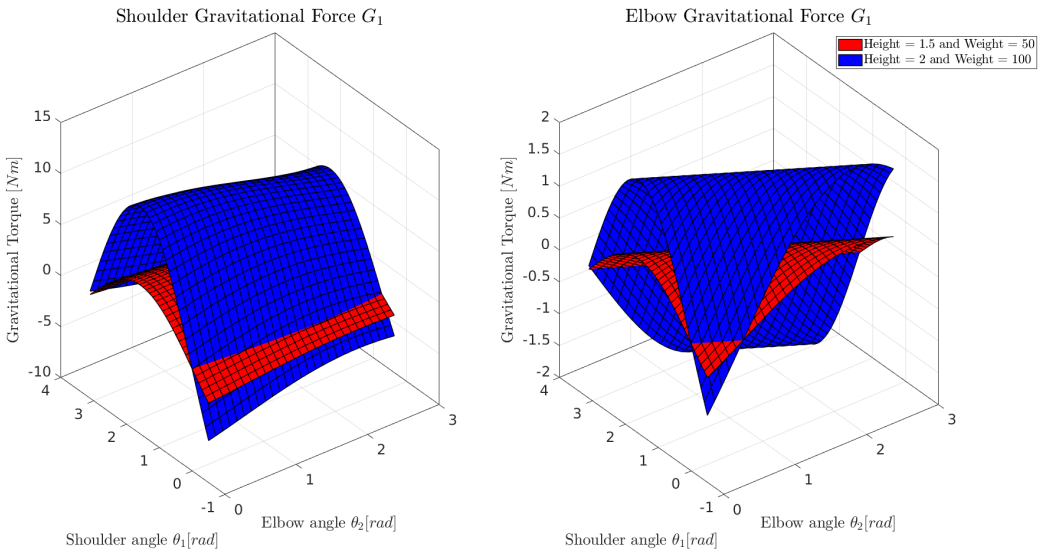


Figure 3.15: Gravitational force for different users

Simplified Model

Overall the human arm is highly nonlinear, user and usage dependent, as can be seen from Figures 3.12, 3.13, 3.14 and 3.15. It depends on the human characteristics, the damping, and stiffness produced by its muscles, and other parameters. Thus, making the obtention of a proper and simple human arm model a really difficult task. However, for this case the impedance human arm model - the elbow equation of motion - is mainly used for stability analysis of the whole system, apart than for simulations. For this purpose it is then approximated by a second order system, as done in Keemink et al. (2018),

$$\tau_{int} = \tau_{elbow} = m_h\ddot{\theta}_p + b_h\dot{\theta}_p + k_h\theta_p \tag{3.34}$$

where $\theta_2 = \theta_p$ as the user forearm is attached to the exoskeleton. Nevertheless, m_h, b_h, and k_h are not considered constant. Instead, they describe all possible configurations in a defined range.

$$0.001 < m_h < 0.1 \tag{3.35}$$

$$0.1 < b_h < 7.5 \tag{3.36}$$

$$0.1 < k_h < 75 \tag{3.37}$$

These ranges are obtained based on the previous human arm model and Buerger and Hogan (2006) so that the muscle characteristics can be introduced.

Low-Level Position Control 4

In this chapter, two new position control strategies are presented, which are based on the analysis by Pedersen and Murcia i Matute (2020), where many control types were tested. The proposed control strategies are a state-space control based on the LQR-control structure, with integral action, also called LQRI. And a nonlinear LQR gain scheduling controller. For this, parameter estimations are needed - marked in green in Figure 4.1 - and a new method is presented. In Figure 4.1 the entire system is represented. This chapter is concerned with the low-level control marked in green.

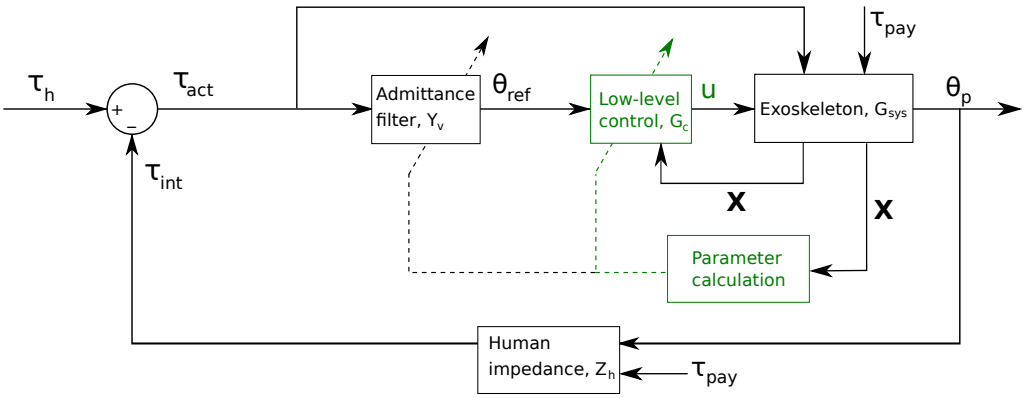


Figure 4.1: Schematic of the system. Low-level position control and parameter estimation are marked in green

In Figure 4.1 it can be seen that the interaction torque, τ_{act}, also affects the system directly. This is, however, not included in the control design process to simplify the design process.

Previous nonlinear control

In Pedersen and Murcia i Matute (2020) two nonlinear controllers were developed. These were a gain scheduling (GS) LQR control and a feedback linearization (FL) control. The controllers were only tested in simulation but were deemed potent solutions. In this project, the nonlinear controllers have been attempted implemented. However, it was concluded that this could not be done since the microcontroller - an Arduino DUE - is not fast enough to allow for a sufficiently low sampling time - see Section 6.4. In the simulation the necessary sampling time was $t_s = 0.001$s, and only a sampling time of $t_s = 0.005$s was achieved. This is due to the extended Kalman filter (EKF) - used for parameter estimations -, which contains multiple 6x6 matrices. Many of these matrices are multiplied multiple times, and an inverse of a 6x6 matrix is also needed. These calculations are cumbersome, so implementing the nonlinear control scheme with an EKF as a parameter observer in the

microcontroller was not feasible. Therefore, new possible solutions are needed.

4.1 Linearization

To linearize the system, a couple of assumptions are made. These assumptions are:

- Small angles are assumed, which means that $\sin(x) = x$. For increasing angles, this assumption becomes worse.
- Coulomb friction is neglected as it is discontinuous and makes the system non-differentiable.
- The stiffness is linear, meaning that the VSM behaves linearly.

For the last assumption the nonlinear stiffness needs to be linearized. This is done using a Taylor series expansion

$$\tau_{tot} \approx \tau_{tot}(\bar{\theta}_d) + \frac{\partial \tau_{tot}(\theta_d)}{\partial \theta_d}\Big|_{\theta_d=\bar{\theta}_d}(\theta_d - \bar{\theta}_d) = K_{eq,\,tot}\Big|_{\theta_d=\bar{\theta}_d}(\theta_d - \bar{\theta}_d) \tag{4.1}$$

Here the linearization point is $\theta_d = 0$. This yields

$$\tau_{tot} = k_L \theta_d \tag{4.2}$$

where k_L is the linear stiffness. For an overview of the system a state-space representation is presented:

$$\begin{bmatrix} \ddot{\theta}_p \\ \dot{\theta}_p \\ \ddot{\theta}_m \\ \dot{\theta}_m \end{bmatrix} = \begin{bmatrix} -\frac{B_{vsm}+B_p}{J_p} & -\frac{m_p l_p g + \tau_{pay} + k_L}{J_p} & 0 & \frac{k_L}{J_p n} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{n\beta k_L}{J_{tot}} & -\frac{B_m + \beta B_{vsm}}{J_{tot}} & -\frac{\beta k_L}{J_{tot}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_p \\ \theta_p \\ \dot{\theta}_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{J_{tot}} \\ 0 \end{bmatrix} \tau_m \tag{4.3}$$

Verification of the linear model

To verify the linear model, the nonlinear model is used. The linearized model is compared with the nonlinear model in a free vibrations simulation, which can be seen in Figure 4.2.

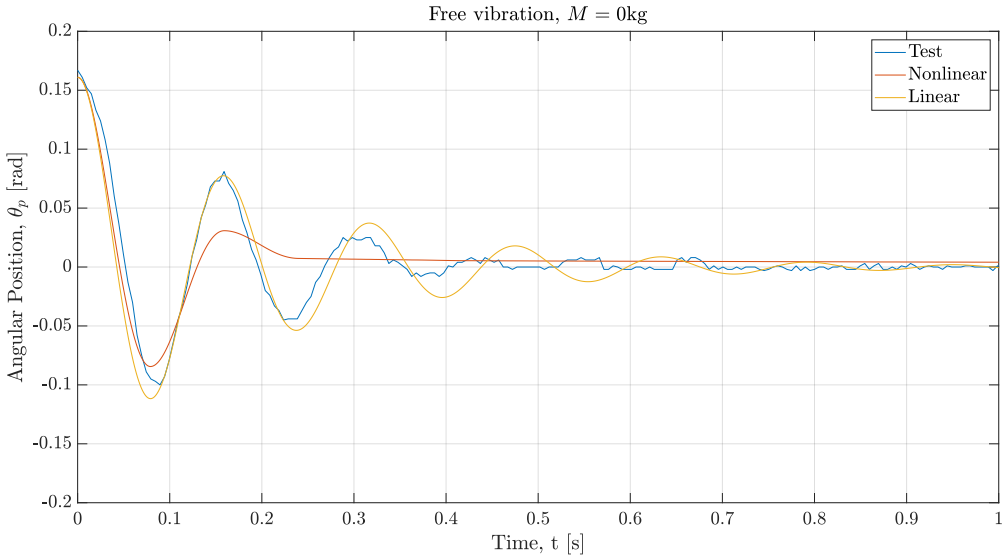


Figure 4.2: Verification of the linear model. $k_L = 2\frac{\text{Nm}}{\text{rad}}$ and $M = 0$ kg

From the figure it can be seen that the linear model approximates the nonlinear model. However, less damping is seen. The linear model does, however, approximated the real system better that the nonlinear model. The main reason for this is that the Coulomb friction has been assumed zero. This will only be the case for a payload of $M = 0$ kg, as explained in Section 3.2.3. However, the two simulation (and the test) have similar natural frequencies. The linear model is deemed sufficient for position control design.

4.2 State-Space Control

From Pedersen and Murcia i Matute (2020) it was clear that none of the linear controllers could effectively control the system for all payloads. Therefore, a new state-space control design is presented based on an LQR design with integral action. For the design of the gains, interaction torque - represented in Figure 4.3 as τ_{dist} - have not been included. However, the controller's disturbance rejection abilities will be analyzed using simulations when evaluating the designed control. In Figure 4.3 the block diagram of the control is presented.

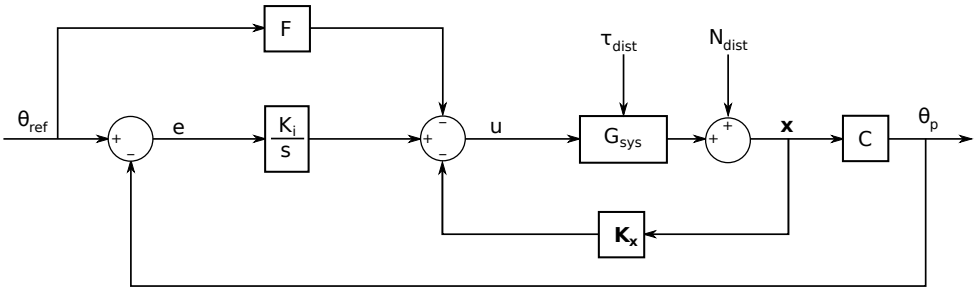


Figure 4.3: Block diagram of state-space control with feed-forward, G_c. θ_{ref} is the angular position reference, e is the error between the reference and the actual angular position, F is feed-forward gain, $\frac{K_i}{s}$ is a gain of the integral of the error, u is the current input to the motor, G_{sys} is the system, $\mathbf{x}$ is the states of the system, $\mathbf{K_x}$ is the gain of the states, and θ_p is the angular position of the exoskeleton. τ_{dist} and N_{dist} are the disturbance from the interaction torque and model inaccuracies and disturbance in terms of sensor noise, respectively

The control law is given by:

$$u = \mathbf{K_x X} - \mathrm{K_i E} + \mathrm{F}\theta_{ref} \tag{4.4}$$

To determine the gains, $\mathbf{K_x}$ and K_i, the LQI method is used. Here the gains are calculated by optimizing a cost function called the Riccati equation. The solution of this optimization problem is gains that satisfy that the system is stable - no RHP poles.

Calculation of feed-forward gain, F

To calculate the feed-forward gain, F, the relationship $\frac{\theta_p}{\theta_{ref}} = 1$ is used. This relationship is stating that the actual angular position is equal to the reference, which means that the error is zero. To calculate this, steady-state is assumed: $\ddot{\theta}_p = \dot{\theta}_p = \ddot{\theta}_m = \dot{\theta}_m = 0$. Looking at the dynamics of the system in Eqs. (3.12) and (3.14), these equations will reduce to:

$$\tau_{vsm} = G_p \theta_p \tag{4.5}$$

$$\tau_m = n\beta\tau_{vsm} \tag{4.6}$$

where $\tau_m = uk_t$, where u is the input from the controller and k_t is a constant used to transform the input, u, from current to torque. In steady-state u is given by:

$$u = -K_2\theta_p - K_4\theta_m + F\theta_{ref} \tag{4.7}$$

Assuming linear stiffness characteristics the VSM torque can be expressed as

$$\tau_{vsm} = k_L \left(\frac{\theta_m}{n} - \theta_p \right) \tag{4.8}$$

Inserting Eq. (4.8) into Eqs. (4.6) and (4.5) the following is obtained:

$$k_L \left(\frac{\theta_m}{n} - \theta_p \right) = G_p \theta_p \tag{4.9}$$

$$u = n\beta \frac{k_L}{k_t} \left(\frac{\theta_m}{n} - \theta_p \right) = -K_2\theta_p - K_4\theta_m + F\theta_{ref} \tag{4.10}$$

To obtain the relationship $\frac{\theta_p}{\theta_{ref}} = 1$, the dependency on θ_m in Eq. (4.10) needs to be removed. Therefore, θ_m in Eq. (4.9) is isolated and inserted into Eq. (4.10):

$$\frac{n}{k_t}\beta G_p\theta_p = \left(-K_2 - K_4\left(\frac{G_p}{k_L} + n\right)\right)\theta_p + F\theta_{ref} \tag{4.11}$$

The relationship can now be found:

$$\frac{\theta_p}{\theta_{ref}} = \frac{F}{\left(\frac{n}{k_t}\beta G_p + K_2 + K_4\left(\frac{G_p}{k_L} + n\right)\right)} \tag{4.12}$$

As mentioned, if the relationship is equal to 1, it is then ensured that the exoskeleton's actual angular position is equal to the reference at steady-state. Therefore, Eq. (4.12) is set equal to 1, and the feed-forward gain, F, is isolated.

$$F = \left(\frac{n}{k_t}\beta G_p + K_2 + K_4\left(\frac{G_p}{k_L} + n\right)\right) \tag{4.13}$$

The term G_p is the gravitational term of the exoskeleton arm dynamics in Eq. (3.14). This term contains the payload as well. However, this payload is not known - but will be estimated in Section 4.3 for the nonlinear control - therefore, it will here be used as a tuning parameter.

4.2.1 Stability Analysis of the State-Space Controller

For the stability analysis, the controller is added to the state-space system in Eq. (4.3). To do this, the system must be augmented such that the integral of the error, accumulated error, is a new state.

$$\begin{bmatrix} \ddot{\theta}_p \\ \dot{\theta}_p \\ \ddot{\theta}_m \\ \dot{\theta}_m \\ e \end{bmatrix} = \begin{bmatrix} -\frac{B_{vsm}+B_p}{J_p} & -\frac{m_p l_p g + F_{pay} + k_L}{J_p} & 0 & \frac{k_L}{J_p n} & 0 \\ 1 & 0 & 0 & 0 & 0 \\ -\frac{K_1 k_t}{J_{tot}} & \frac{-K_2 k_t + n\beta k_L}{J_{tot}} & -\frac{K_3 k_t + B_m + \beta B_{vsm}}{J_{tot}} & -\frac{K_4 k_t + \beta k_L}{J_{tot}} & -\frac{K_5 k_t}{J_{tot}} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_p \\ \theta_p \\ \dot{\theta}_m \\ \theta_m \\ \int e \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{F k_t}{J_{tot}} \\ 0 \\ 1 \end{bmatrix} \theta_{ref} \tag{4.14}$$

Given this new system - still with θ_p as the only output - the linear stability can be analyzed.

To determine the stability of the proposed controller, a pole-zero map is used. The poles and zeros of the system - for multiple configurations of stiffness and payload - are plotted. If all of the poles are located in the left-half plane (LHP) then the system is stable. The plots can be seen in Figures 4.4 and 4.5.

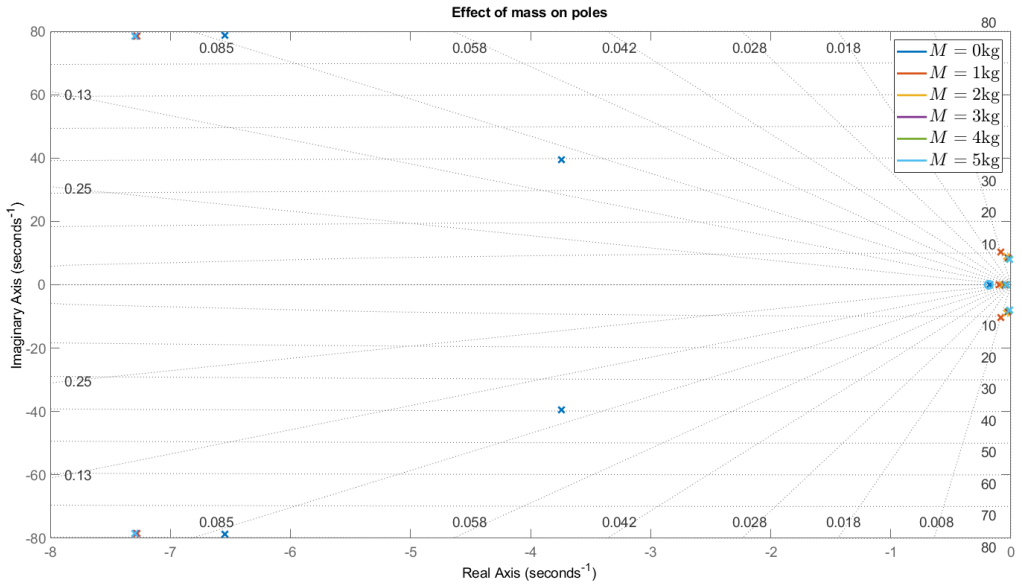


Figure 4.4: Illustration of how the poles and zeros are affected by a changing payload mass. $k_L = 2\left[\frac{Nm}{rad}\right]$

From Figure 4.4 it can be seen that increasing the payload mass moves the slowest poles closer to the RHP. This has the effect of decreasing the damping and the system response. This behavior makes sense when relating to the physics of adding a payload, as increasing the mass decreases the system's natural frequency and increases the energy required to change its velocity. Although the showed configurations are stable, the system likely becomes unstable for some larger payload masses. The relationships between the real pole and the complex poles seem to be similar for all payload masses, except for $M = 0$ kg. This means that the response is expected to be similar for all the payload masses. For $M = 0$ kg the response would seem to be mainly dominated by the real pole - hence a response approximating a first-order response is expected.

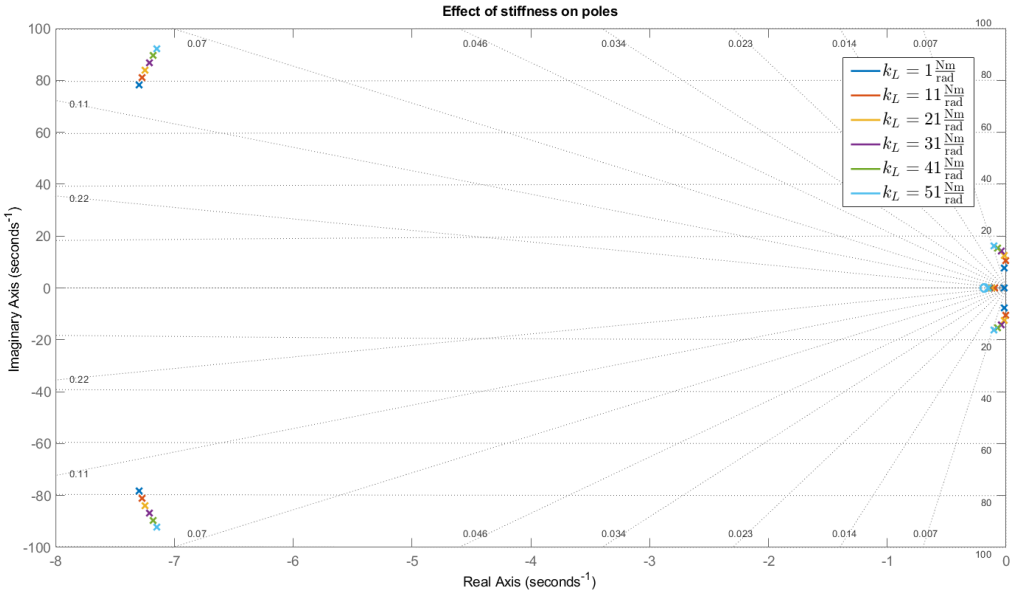


Figure 4.5: Illustration of how the poles and zeros are affected by a changing stiffness. M = 5 [kg]

From Figure 4.5 it can be seen that the fastest, left-most poles are much faster than the other poles. Hence the response will be dominated by the slowest, right-most poles. Looking at the behavior of the poles as the linear stiffness changes, it can be seen that the poles move slightly out and away from the RHP. This makes the system less oscillatory and increases the system response. This behavior is also expected, as increasing the stiffness increases the natural frequency of the system.

Overall, it can be seen from the figures that the system is stable for all analyzed configurations. However, care should be taken when increasing the payload significantly.

Robustness

As mentioned, the interaction torque, τ_{act}, also affects the VSM and should therefore be taken into account. However, for simplicity, it has been neglected in the control design. To see if the controller can handle τ_{act}, it is added in the simulations as a disturbance. Also, white noise is added to the simulation to simulate sensor noise.

For the simulations the τ_{act} is simulated as a sine wave with an amplitude of $A = 1$ Nm and a frequency of $f = \pi \frac{\text{rad}}{\text{s}}$. The white noise has a variance of $\sigma_{noise} = 0.02$ Nm, which is roughly 10% of the variance of the sum of all torques. The robustness simulation can be seen in Figure 4.6.

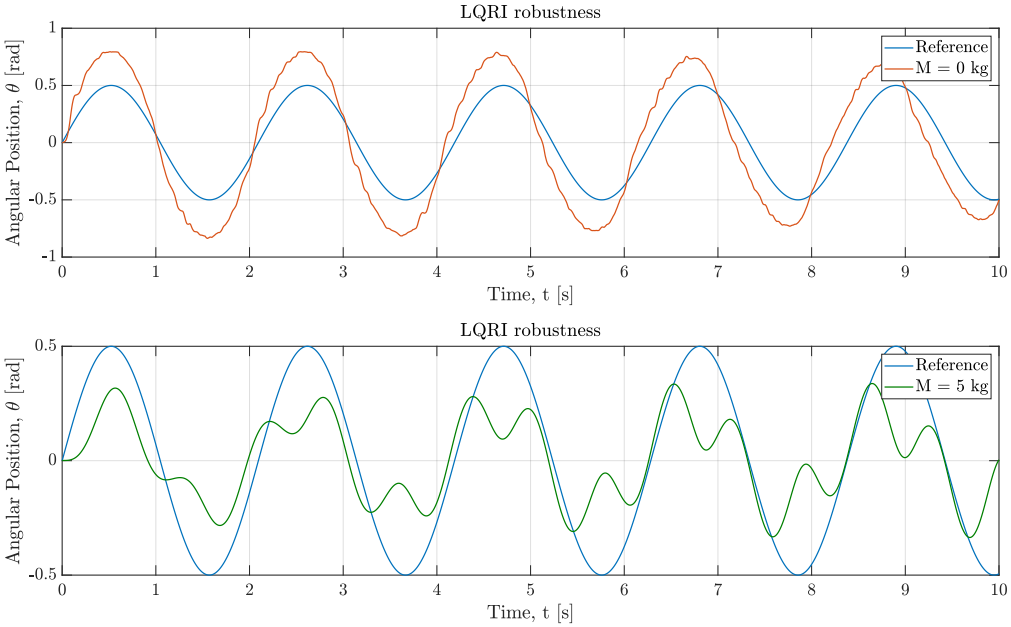


Figure 4.6: Robustness of the controller with payloads of $M = 0$ kg and $M = 5$ kg for a sinusoidal input

From the plots it can be seen that the controller can somewhat handle noise, but is largely affected by the interaction torque. The errors for the two tests can be seen in Figure 4.7.

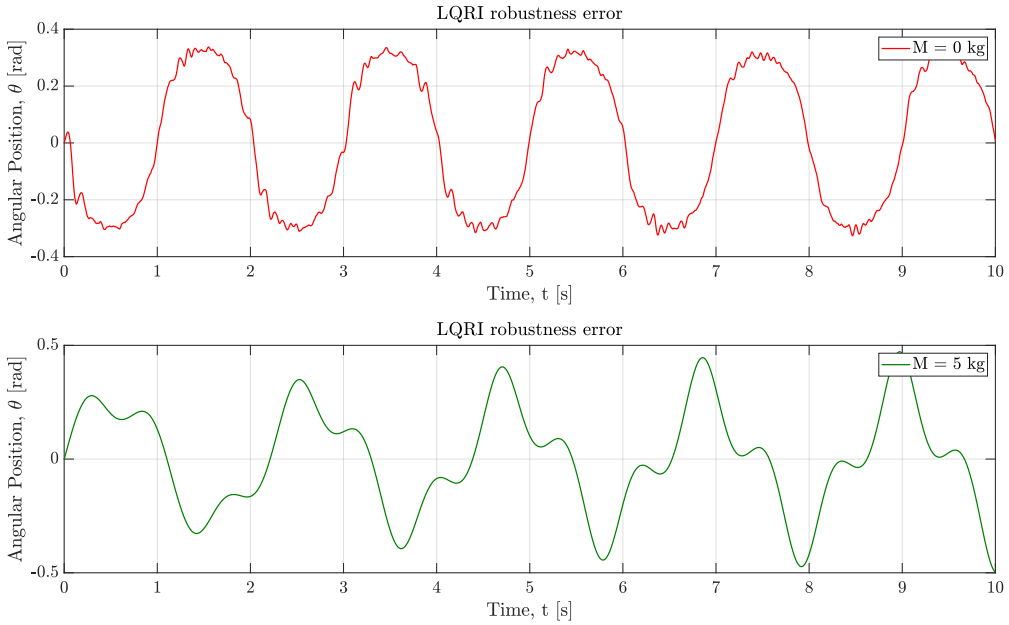


Figure 4.7: Errors for payloads of $M = 0$ kg and $M = 5$ kg for a sinusoidal input

The average absolute errors for the two simulations are given in Table 4.1.

Payload	Error
M = 0 kg	0.2268 rad
M = 5 kg	0.1829 rad

Table 4.1: Average error of robustness simulations

The average errors clearly show that the controller cannot reject disturbances. It is expected that the payload has a more significant effect on the system's performance than the disturbance. However, the robustness analysis suggests that the disturbance has a larger effect, given the scale of errors. The controller's performance is analyzed to study this, where the bandwidth is found and the disturbance is removed.

Performance - bandwidth

To evaluate the controller's performance a chirp signal is given to the system. A chirp signal gradually increases frequency. As the amplitude is 1 rad, bandwidth is the frequency at which the maximum position goes below -3 dB $\cdot$ 1 rad = 0.708 rad. The chirp signal has an initial frequency of 0.063 $\frac{\text{rad}}{\text{s}}$ and an end frequency of 2π $\frac{\text{rad}}{\text{s}}$, which is reached after 100 seconds. The chirp signal simulation can be seen in Figure 4.8.

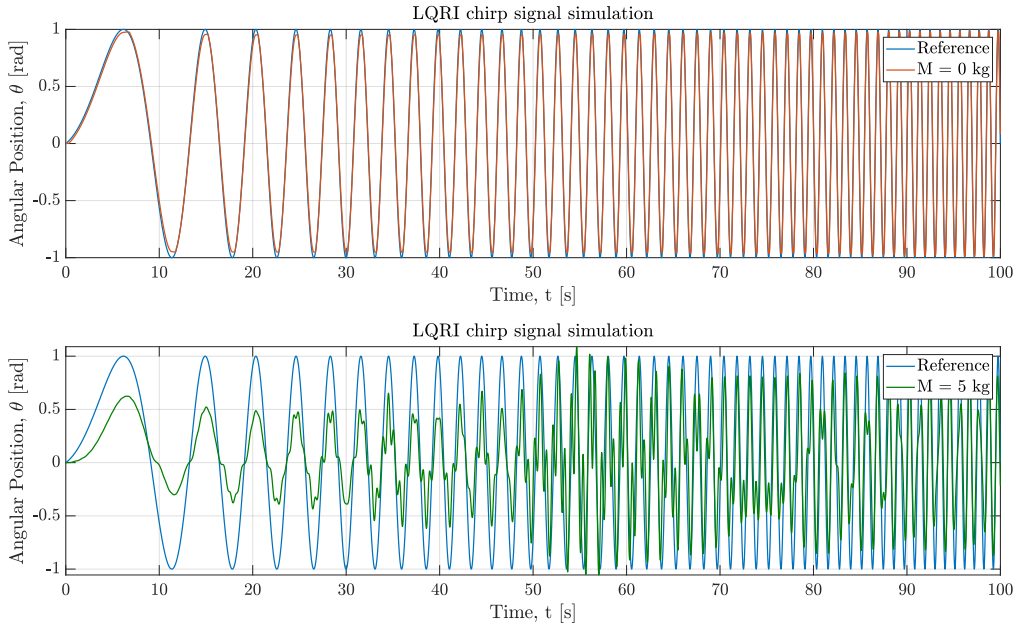


Figure 4.8: Chirp signal input for payloads $M = 0$ kg and $M = 5$ kg

Looking at the top graph in Figure 4.8 it can be seen that the controller has more than a bandwidth of 2π $\frac{\text{rad}}{\text{s}}$ when the payload is zero. This means that errors seen in the robustness simulation in Figure 4.7 are due to the poor disturbance rejection capabilities of the controller. From the bottom graph, however, it becomes clear that the controller loses a substantial amount of bandwidth for a payload of $M = 5$ kg. As it cannot follow the reference for 0.063 $\frac{\text{rad}}{\text{s}}$, the bandwidth is low than 0.063 $\frac{\text{rad}}{\text{s}}$. The exact value is not relevant as it is already more than 100 times lower than the requirements. This means that the error seen in robustness simulation in Figure 4.7 is due to the poor bandwidth and that the

payloads effect on the system is, as expected, more significant than the disturbance. The controller's performance is unacceptable when comparing to the requirements. Increasing the gains further will result in instability, as shown in Section 4.2.1. It seems, therefore, that the limit of the controller has been reached. Therefore, a nonlinear controller will be designed and analyzed to see if a sufficient low-level position control can be designed.

4.3 Gain Scheduling LQR

As seen in Section 4.8, the linear controller does not have a high enough bandwidth to control the system satisfactorily when the payload is added. Therefore, a new method for parameter estimation will be presented, and a gain scheduling LQR controller will be designed.

As mentioned at the beginning of Chapter 4, an EKF and two nonlinear controllers in Pedersen and Murcia i Matute (2020) were developed. However, due to hardware limitations, it has not been possible to implement the EKF, and the controllers were therefore disregarded. A new methodology - that uses the properties of the VSM to estimate the necessary parameters - has been developed, which makes the implementation of the nonlinear controllers possible. However, an EKF should be used if possible since it can provide much more accurate parameter estimations.

New methodology for model based parameter estimation

In the previous system in Pedersen and Murcia i Matute (2020), the variable stiffness mechanism was more reconfigurable than the updated system used in this project. This means fewer parameters vary over the deflection angle, θ_d, which enables using the model for estimating the two variable parameters left. These two parameters are stiffness and payload torque. For determining the stiffness, the formulation in Eq. (3.11) is used. As two encoders are measuring the motor position and the exoskeleton arm position, the deflection angle can be calculated from Eq. (4.15).

$$\theta_d = \frac{\theta_m}{n} - \theta_p \tag{4.15}$$

where θ_m is the motor position, n is the gear ratio, and θ_p is the exoskeleton angle.

To estimate the payload the link side dynamics equation in Eq. (3.14) is used - repeated here:

$$\tau_{vsm}(\theta_d) + \tau_{act} = J_p\ddot{\theta}_p + (B_{vsm} + B_p)\dot{\theta}_p + T_{c,p}\text{sign}(\dot{\theta}_\text{p}) + \tau_{pay} \tag{4.16}$$

Here steady-state is assumed, which reduces the equation to:

$$\tau_{vsm} + \tau_{act} = \tau_{pay} \tag{4.17}$$

In the real system both the payload and the interaction torque will be affecting the exoskeleton such that the effective payload that needs to be compensated for is given by:

$$\tau_{pay,\,effective} = \tau_{pay} - \tau_{act} \tag{4.18}$$

Comparing with Eq. (4.17) it is clear that the effective payload, which needs compensating is equal to the VSM torque - assuming steady-state.

$$\tau_{vsm} = \tau_{pay,\,effective} \tag{4.19}$$

From equation Eq. (3.10) the VSM torque can be calculated. To keep the payload definition as general as possible, it will be defined as a torque, such that the payload mass will not be calculated. The gain scheduling controller will therefore be designed using the stiffness of the VSM and the payload force. The definition is seen in Eq. (4.20).

$$\tau_{pay} = \frac{\tau_{vsm}}{\sin \theta_p} \tag{4.20}$$

Care should be taken when θ_p approaches 0. However, due to the definition of θ_p - when the arm is completely straight and parallel to the support link - an angle of $\theta_p = 0$ will rarely occur. In any case, a moving average filter can be used to mitigate this problem.

4.3.1 Control Design

The control structure for an LQR controller can be seen in Figure 4.9.

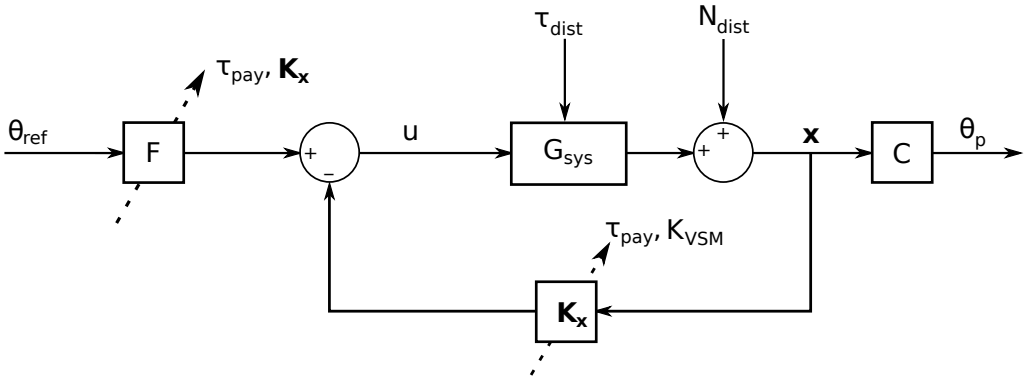


Figure 4.9: Schematic of the LQR gain scheduling

A gain scheduling controller has its gains updated based on either system states or parameters. In Pedersen and Murcia i Matute (2020) a gain scheduling LQR control was developed, but it was based on an EKF, which was not able to be implemented. However, using the new payload estimation, it should now be possible to implement. Therefore, a gain scheduling LQR (GS LQR) controller will be designed.

To design the gains, the 'lqr'-function in MATLAB is used. This function takes the parameters, and the system as input optimizes the gains for a minimization problem of a cost function and outputs the optimal gains. The cost function is seen in Eq. (4.21).

$$J(u) = \int_0^\infty \left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + 2\mathbf{x}^T\mathbf{N}\mathbf{u}\right) dt \tag{4.21}$$

where $\mathbf{x}$ is the states, $\mathbf{Q}$ contains the penalty of the importance of the states - high values equal high cost, which means more importance -, $\mathbf{u}$ is the input, $\mathbf{R}$ contains the penalty of the importance of the input - high values equal high cost, which means more importance. The last term describes penalty of the cross product of the states and input. This term is set to zero, i.e., $\mathbf{N} = \mathbf{0}$. The quadratic nature of the cost function is used to ensure that any negative values of the states and input are not decreasing the cost. MATLAB needs $\mathbf{Q}$ and $\mathbf{R}$ matrices, as well as the system matrix, $\mathbf{A}$, and input matrix, $\mathbf{B}$. The $\mathbf{Q}$ and $\mathbf{R}$

matrices can be seen in Eq. (4.22).

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{R} = 0.0331 \tag{4.22}$$

The cost parameters have been chosen on a trial-and-error basis, but the intent is to penalize the exoskeleton position highly since errors in the position have a large effect on the performance of the entire system - see Chapter 5. The values of the input are not as important, as long as saturation is not a problem.

Since some of the parameters - namely the payload and VSM stiffness - varies over the states, multiple gains will be calculated and stored in an array. To represent as many configurations of the parameters as possible, the gains are calculated for 280 configurations. A fitting polynomial is then found, dependent on two variables - the payload and the stiffness. For stability reasons, the polynomials are as high of an order as possible. The polynomials and gains for the polynomials can be seen in Appendix B.

Stability

For an LQR controller, stability is guaranteed for the gains found. This is also the case for the GS LQR, but only for the specific gains found through solving the optimization problem. Since polynomials are used to interpolate between the gains, stability cannot be guaranteed for all points, but only in the vicinity of the gains (CHEN, 2005). This region of stability can be challenging to determine. Therefore, stability will be inferred from simulations.

Robustness

To determine how robust the controller is, simulations will be made with different payloads, where a disturbance is added consisting of a noise signal - simulating sensor noise - and the interaction torque. The values of these are given in Table 4.2.

Payload	Disturbance, τ_{act}	Noise
M = 0 kg	$A = 0.2$ Nm, $f = \pi \frac{\text{rad}}{\text{s}}$	$\sigma = 0.02$ Nm
M = 5 kg	$A = 2$ Nm, $f = \pi \frac{\text{rad}}{\text{s}}$	$\sigma = 0.2$ Nm

Table 4.2: Disturbance values for the GS LQR robustness simulation

The simulations can be seen in Figures 4.10 and 4.11.

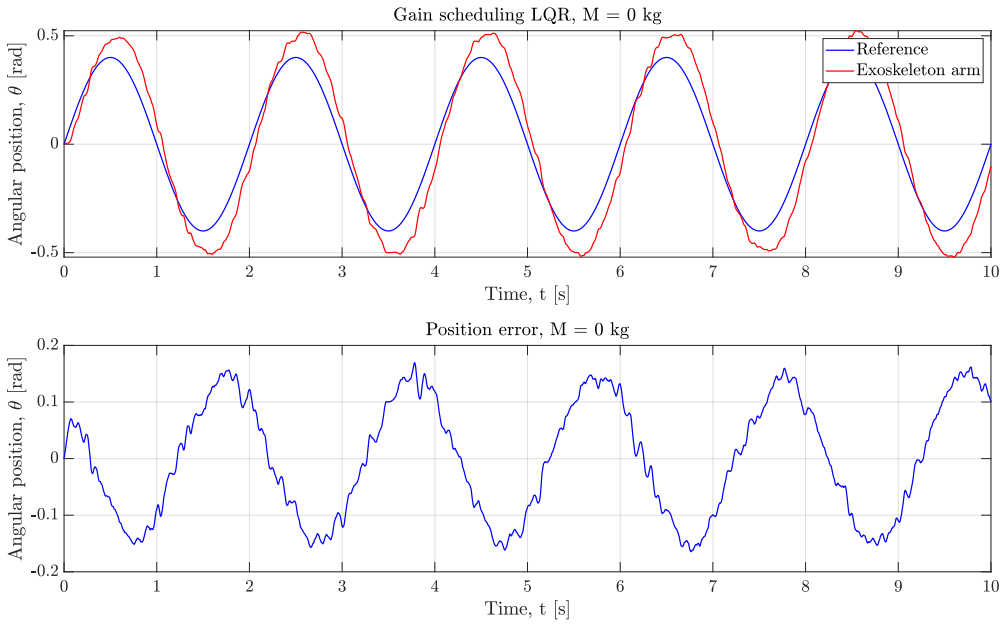


Figure 4.10: Robustness simulation of the gain scheduling LQR controller, with payload of $M = 0$ kg

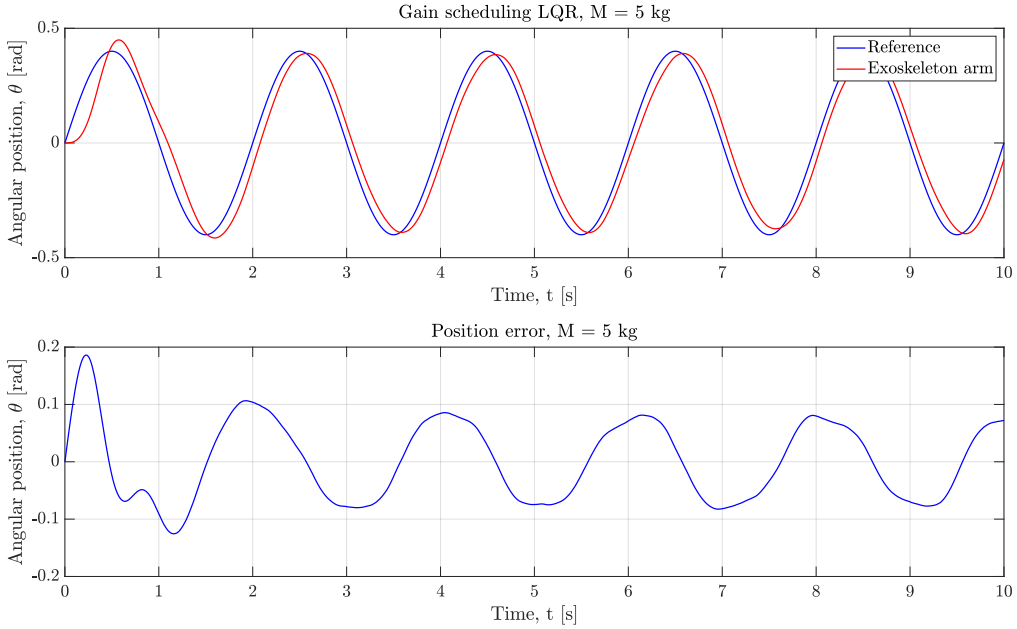


Figure 4.11: Robustness simulation of the gain scheduling LQR controller, with payload of $M = 5$ kg

From the figures, it can be seen that the controller can reject disturbances when a payload is acting on the system. However, when there is no payload, it struggles to reject disturbances. The main reason for this is that the payload's effect on the system is more significant than the disturbance, so the controller's ability to handle the payload size means that the disturbance can more easily be rejected. However, when there is no payload affecting the system, the influence of the disturbance becomes amplified. A

payload of $M = 0$ kg is, however, not very realistic as the interaction force sensor and cuff will be attached to the arm, as well as the human itself.

Looking at the errors, it is clear that the main error comes from the slight phase shift present in the system. The mean absolute error of payload $M = 5$ kg is $MAE_5 = 0.095$ rad, which around twice of the requirement. Mean absolute error of payload $M = 0$ kg is similar at $MAE_0 = 0.090$ rad. These errors can be decreased by increasing the gains, but this will not be done for stability reasons. Also, a phase shift will not affect the performance as much as a steady-state error, which does not seem to be present in the system.

Performance - bandwidth

To evaluate the controller's performance, the bandwidth and the errors of a sinusoidal input are studied. Bandwidth is usually defined as the frequency at -3 dB of gain. Since the GS LQR is a nonlinear controller, the bandwidth will be studied through simulations, specifically, using a chirp signal as input. The chirp signal's frequency gradually increases as time increases until a final, specified frequency is reached. The amplitude will be defined as 1 rad, meaning that when the controller cannot reach a maximum position of -3 dB $\cdot$ 1 rad $= 0.708$ rad the frequency at this point will be defined as the controller's bandwidth. The chirp signal settings can be seen in Table 4.3.

pay	Total time	Initial frequency	End frequency
M = 0 kg	50 s	0.063 $\frac{\text{rad}}{\text{s}}$	12.57 $\frac{\text{rad}}{\text{s}}$
M = 5 kg	100 s	0.063 $\frac{\text{rad}}{\text{s}}$	6.91 $\frac{\text{rad}}{\text{s}}$

Table 4.3: Chirp signal settings.

The simulation result can be seen in Figures 4.12 and 4.13.

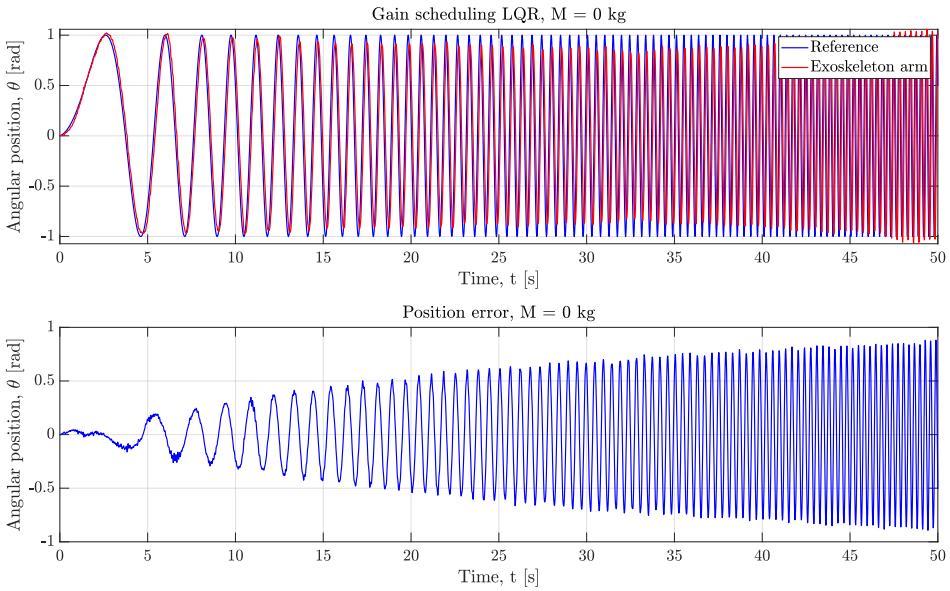


Figure 4.12: Chirp signal simulation of the gain scheduling LQR controller, with payload of $M = 0$ kg

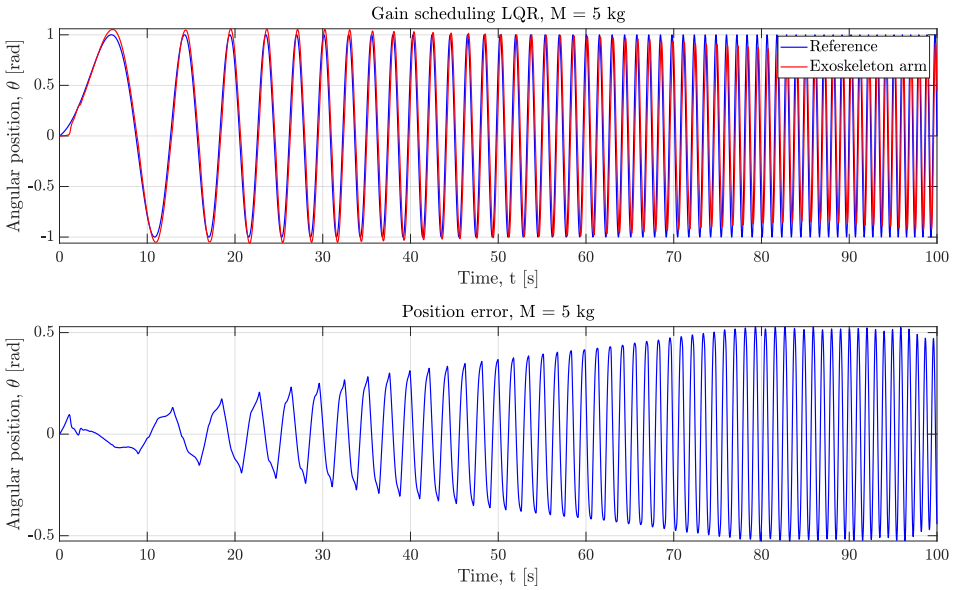


Figure 4.13: Chirp signal simulation of the gain scheduling LQR controller, with payload of $M = 5$ kg

For the simulation for no payload in Figure 4.12, the controller starts struggling to follow the reference at 10 seconds and reached the smallest maximum exoskeleton position at 33 seconds. The maximum position does not go below 0.708 rad, therefore, the bandwidth will be considered as the frequency at 33 seconds. This frequency is calculated to be: $f_{b,\,0} = \frac{33\text{ s}}{50\text{ s}}\ 12.57\ \frac{\text{rad}}{\text{s}} = 8.2962\ \frac{\text{rad}}{\text{s}}$. This bandwidth is 32% higher than the requirement of 2π. The controller can therefore successfully control the system for no payload.

For the simulation for a payload of $M = 5$ kg, in Figure 4.12, it can be seen that the controller struggles to follow the reference at around 65 seconds, which culminates around 85 seconds before increasing again. The maximum exoskeleton position does not go below 0.708 rad, therefore, the bandwidth will be considered to be the frequency at 85 seconds, which approximates to: $f_{b,\,5} = \frac{85\text{ s}}{100\text{ s}}\ 6.91\ \frac{\text{rad}}{\text{s}} = 5.8735\frac{\text{rad}}{\text{s}}$. This bandwidth is roughly 93% off the requirement, $2\pi\ \frac{\text{rad}}{\text{s}}$. This can likely be increased by increasing the gains. However, for the stability reasons mentioned previously a more conservative approach has been taken, and the control is deemed sufficient.

Stability-wise, it can be seen in all the simulations that the system can handle different disturbances, parameters, configurations, and inputs without becoming unstable. Therefore, the system is assumed to be stable for reasonable configurations.

High-Level Control 5

In this chapter, the high-level control is presented and discussed. The high-level control in an exoskeleton can take different forms, depending on, for example, the sensor's data. In this project, as can be seen in Figure 5.1, a force interaction sensor between the human arm and the exoskeleton is used, τ_{act}. Using this information, the high-level control must give a reference position signal to the low-level control, developed in Chapter 4. As can be seen in Figure 5.1 the human arm behaves as an impedance system, meaning it provides a torque after a motion. Thus an admittance filter - the inverse of impedance - is used as the high-level control. Moreover, two kinds of exoskeletons that interact with humans can be considered, an exoskeleton where the payload is attached to it or an exoskeleton where the payload is grabbed directly by the user.

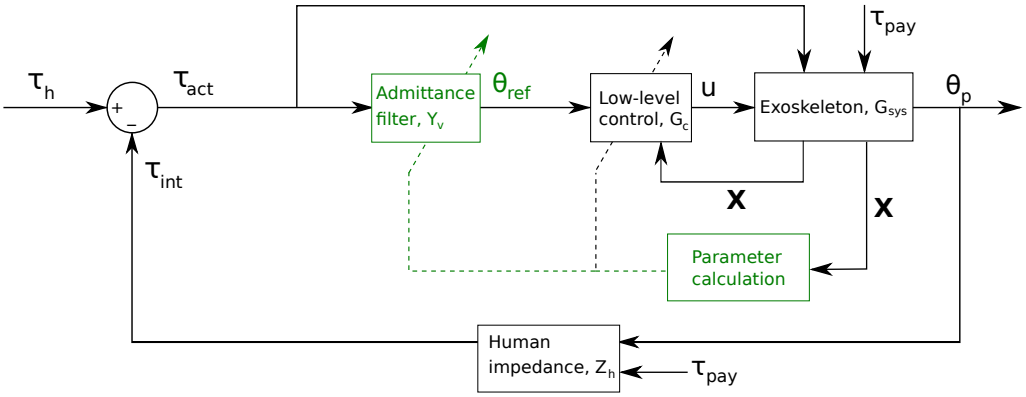


Figure 5.1: Schematic of the system. High-level control and parameter estimation for an adaptive admittance filter is marked in green

The voluntary torque applied by the user, τ_h, and the intrinsic human arm dynamics, τ_{int}, are unknown. Only the interaction force, τ_{act}, is known by the control algorithm. Also, the exoskeleton, VSA, is a MISO (Multiple Input, Single Output) system, having the interaction force, τ_{act}, and the position reference, θ_{ref} as inputs.

In this chapter, both high-level controls - for the two payload configurations - will be discussed.

5.1 Exoskeleton Payload

In this configuration, the payload is attached directly to the exoskeleton - increasing its total weight, see Figure 5.2. The goal then is to transform the exoskeleton's dynamic

behavior using a high-level control so that its user feels the desired dynamics. This could either make the system feel lighter or heavier from the user's point of view.

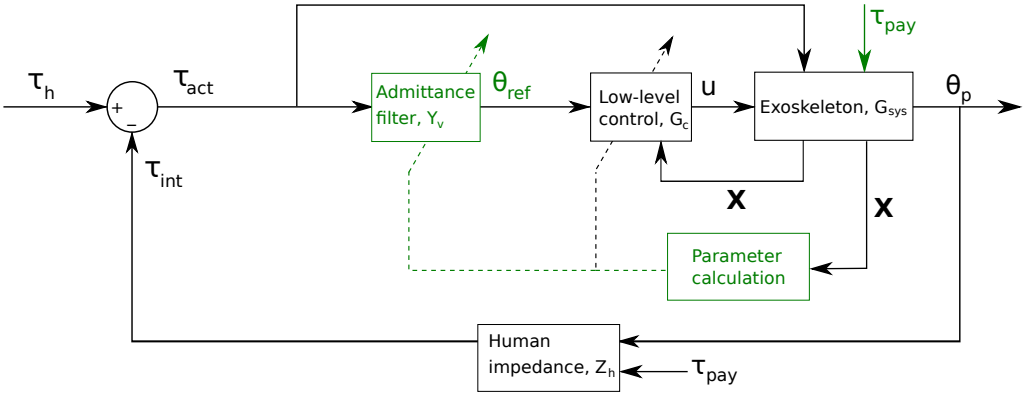


Figure 5.2: Schematic of the exoskeleton with the payload attached on it - marked in green

To achieve this goal, an admittance filter can be used. It works in the way that, when provided by an interaction force between the user and the exoskeleton, the admittance filter will then produce a reference based on its virtual dynamics, which the low-level control will track. If the low-level control is ideal, the whole system would respond as the admittance filter dynamics command. In other words, the admittance filter is responsible for determining the dynamics felt by the user. Furthermore, it is designed depending on the desired exoskeleton behavior.

The admittance filter then can be understood as a simple second order equation of motion

$$Y_v(s) = \frac{1}{M_a s^2 + B_a s + G_a} \tag{5.1}$$

where M_a, B_a, and G_a are the desired inertia, damping, and potential, to be felt by the user. Since the gain scheduling low-level control is a polynomial of LQR controllers for different payloads and stiffness, the system can be approximated as a state-space at different configurations - indeed a parameter varying state-space. From this approximation a linear control structure analysis can be done. Then the whole exoskeleton state-space formulation - from interaction torque to exoskeleton forearm position - can be defined as

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}\mathbf{X} + \mathbf{B}\tau_{act} \\ \theta_p &= \mathbf{C}\mathbf{X} \end{aligned} \tag{5.2}$$

where

$$\mathbf{X} = \begin{bmatrix} \dot{\theta}_p & \theta_p & \dot{\theta}_m & \theta_m & \dot{\theta}_{ref} & \theta_{ref} \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} -\frac{B_{vsm}+B_p}{J_p} & -\frac{\tau_{pay}+k_L}{J_p} & 0 & \frac{k_L}{nJ_p} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{K_1 k_t}{J_{tot}} & \frac{-K_2 k_t + n\beta k_L}{J_{tot}} & -\frac{K_3 k_t + B_m + \beta B_{vsm}}{J_{tot}} & -\frac{K_4 k_t + \beta k_L}{J_{tot}} & 0 & \frac{F k_t}{J_{tot}} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{B_a}{M_a} & -\frac{G_a}{M_a} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.3}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{J_p} & 0 & 0 & 0 & \frac{1}{M_a} & 0 \end{bmatrix}^T$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

from which the stiffness, k_L, payload, τ_{pay}, and exoskeleton forearm inertia, J_p are the varying parameters, and the gains K_1, K_2, K_3, and K_4 are dependent on those parameters. The transfer function is given by

$$Y_s(s) = \frac{\theta_p}{\tau_{act}} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \tag{5.4}$$

Using this approximated model the systems performance, without considering the coupling with its user, can be evaluated using a bode plot. Here the payload is considered to be M = 4 kg and it is desired to feel as M = 2 kg. Then the admittance filter variables are set to be $M_a = \frac{J_p}{2}$, $G_a = \frac{\tau_{pay}}{2}$, and the damping is arbitrarily decided to be $B_a = 1$.

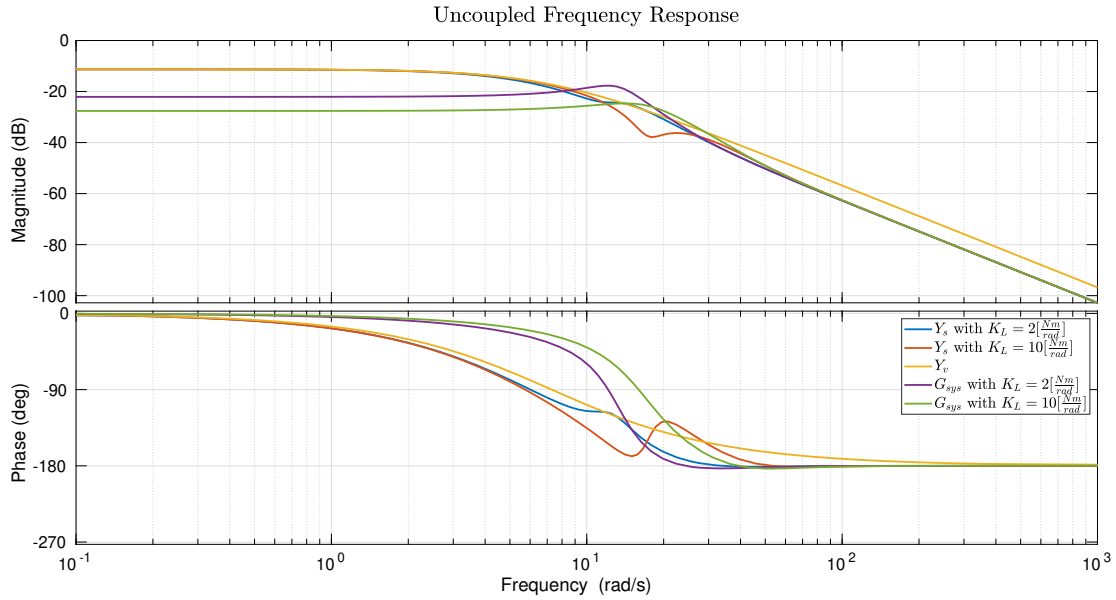


Figure 5.3: Uncoupled system frequency response. Y_s is the whole system dynamics, Y_v the virtual dynamics set by the admittance filter, and G_{sys} is the exoskeleton response due to an interaction force, τ_{act}

Figure 5.3 represents how the system will perform, based on its linear approximation. From this, it can be seen the desired dynamics set by the admittance filter, Y_v, and

that the low-level control can track the reference set by the admittance filter, Y_v at low frequencies (smaller than 10 $\frac{\text{rad}}{\text{s}}$), meaning that the whole system, Y_s, will behave as the admittance filter virtual dynamics commands. However, at higher frequencies (higher than 10 $\frac{\text{rad}}{\text{s}}$), the whole system Y_s switches to behave as the exoskeleton dynamics, G_{sys}, considering the interaction force, τ_{act}, as its input. When this change of behavior happens, an excess of phase lag is introduced into the system. The admittance filter can have different configurations depending on the parameters used. However, in this case, the used admittance filter is a mass-damper-spring combination, which is a good representation of the human forearm.

The exoskeleton actuator is a VSA, having a nonlinear stiffness. In Figure 5.3 two different linear stiffness are compared, and it can be seen that VSM stiffness does not highly affect the system's performance. Instead, the behavior at low frequencies while tracking the virtual dynamics is very similar. The only visible effect seen on the frequency response is that as the admittance gravitational term has been set to depend only on the payload, the stiffness factor is not taken into account, and the admittance filter will have the same virtual dynamics, disregarding the stiffness. This yields to help more at some positions than others but never restricting the user's motion.

Apart from the stiffness, the system's payload can vary as well. In Figure 5.4 it can be seen that if a heavier mass is placed on the exoskeleton, it will demand a higher assistive torque in order to still follow the desired dynamics. However, if the exoskeleton has a lower mass, it will work against the user in order to respond as the admittance filter, which demands dynamics with a higher payload.

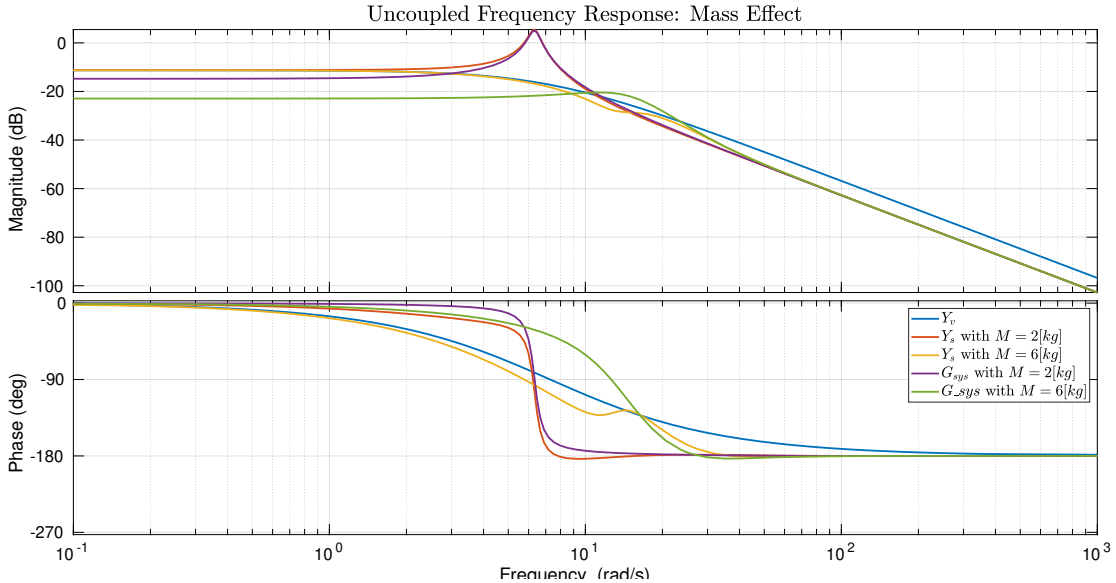


Figure 5.4: Uncoupled system frequency response with a mass

This analysis gives an overview of how an admittance-based control algorithm behaves. However, only the uncoupled system has been analyzed, in which the user is not in the feedback. So it has to be complementarily analyzed with the coupled system stability.

5.1.1 Stability

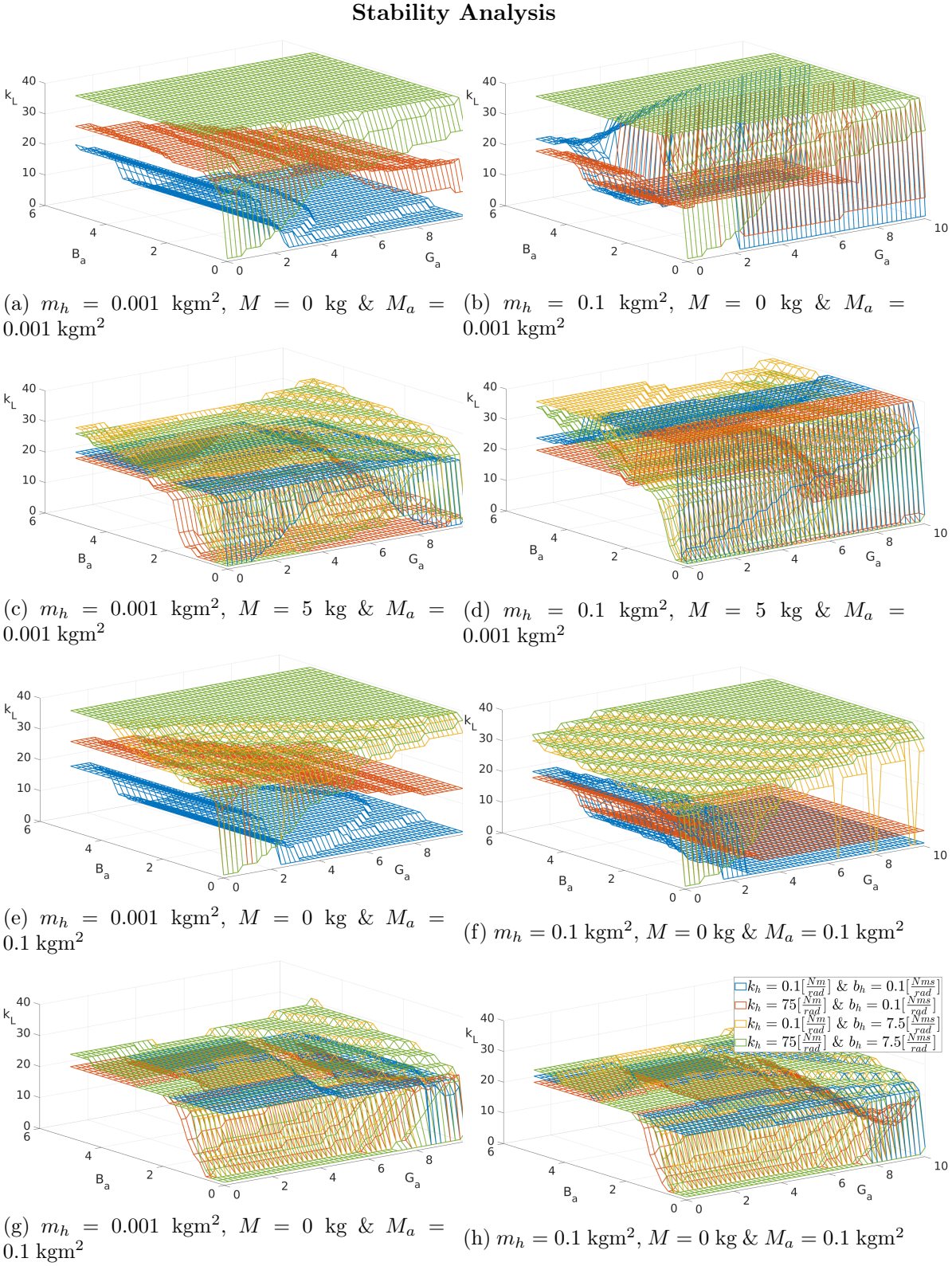
There are different ways to study an HRI exoskeleton based on admittance control. One of the most common ones is by ensuring 'passivity' of the system. A passive system is defined as a system that cannot deliver more energy than the energy given to it, and in Colgate (1988) has been shown that if the target inertia is below 50% of its real value, the system will not behave as a passive system. This method is based on everyday life experience, in which if an active system interacts with a passive system, the interaction will have stable dynamics. So then, if 'passivity' of admittance control based exoskeleton can be guaranteed, the HRI will be stable(Colgate, 1988). Nonetheless, this approach has some limitations. One of them is that time delays, actuators, and sensor limitations can affect the system's passivity, making it a complex task to implement a passive admittance-based control. Moreover, it restricts the admittance filter performance due to its conservative nature. When aiming for 'passivity,' it is assumed that the human arm can have infinite inertia, stiffness, and damping, while in reality, HRI is bounded (Keemink et al., 2018). The approach used here considers the bounded HRI dynamics, which considers any given time impedance dynamics of any user. These impedance parameter boundaries have been obtained by combining the ones used in Buerger and Hogan (2006) and the human arm model developed in Section 3.3. These boundaries are presented again here

$$\begin{aligned} 0.001 \leq m_h \leq 0.1 \\ 0.1 \leq b_h \leq 7.5 \\ 0.1 \leq k_h \leq 75 \end{aligned} \tag{5.5}$$

To study stability a numerical approach has been used, for this range of human arm parameters and a range of different possible admittance controllers, payloads, and stiffness of the system.

$$\begin{aligned} 0.001 \leq M_a \leq 0.1 \\ 0.1 \leq B_a \leq 5 \\ 0 \leq G_a \leq 10 \\ 0 \leq M \leq 5 \\ 1 \leq k_L \leq 39 \end{aligned} \tag{5.6}$$

Two main things must be mentioned: the stability is with respect to the VSM stiffness, k_L, meaning that the limit stiffness - that makes the system unstable - is used to represent the stability of the coupled exoskeleton. Due to the limit stiffness introduced is $k_L = 39\frac{\text{Nm}}{\text{rad}}$ being higher than the seen in the stiffness model - Section 3.2.1 - the coupled exoskeleton stability, that is stable for the whole stiffness range is represented as a plane at $k_L = 39\frac{\text{Nm}}{\text{rad}}$. Secondly, to reduce the complexity, the extremes of each parameter are used to study coupled stability and represent how the stability changes in between.



(a) $m_h = 0.001$ kgm^2, $M = 0$ kg & $M_a = 0.001$ kgm^2

(b) $m_h = 0.1$ kgm^2, $M = 0$ kg & $M_a = 0.001$ kgm^2

(c) $m_h = 0.001$ kgm^2, $M = 5$ kg & $M_a = 0.001$ kgm^2

(d) $m_h = 0.1$ kgm^2, $M = 5$ kg & $M_a = 0.001$ kgm^2

(e) $m_h = 0.001$ kgm^2, $M = 0$ kg & $M_a = 0.1$ kgm^2

(f) $m_h = 0.1$ kgm^2, $M = 0$ kg & $M_a = 0.1$ kgm^2

(g) $m_h = 0.001$ kgm^2, $M = 0$ kg & $M_a = 0.1$ kgm^2

(h) $m_h = 0.1$ kgm^2, $M = 0$ kg & $M_a = 0.1$ kgm^2

Figure 5.5: Numerical stability analysis

From this analysis, different characteristics of the coupled stability can be obtained. First, considering the different combinations of possible admittance filters, it can be seen that pure inertial admittance filter, M_a, would tend to be closer to unstable. Contrarily, a pure

damper admittance filter, B_a, has the best stability-wise response for all kinds of human arm configurations. Generally, it can be seen how the mass attached to the exoskeleton, M, dictates the maximum value of the admittance potential, G_a. This can be because if the payload is bigger, the control algorithm will set heavier virtual dynamics, and the exoskeleton will restrict instead of helping, potentially making the system unstable. An exception to this can be seen in Figure 5.5b.

Another detail that can be extracted is that as the payload increases, the stability impact of the different possible human arms converges to the same stability limit. It is also important that from this analysis, as mentioned, the extreme cases have been studied, some of them being fairly unlikely. However, a general idea of the system's stability can be achieved.

From this study a range of different stable admittance filters can be obtained, giving a range of possibilities when designing the desired virtual dynamics, see Table 5.1. In the case of not knowing the systems payload a good approach would be to assume the worse case scenario, being in this case for a payload of M = 0 kg.

$M = 0$	$M = 5$
$0.001 < M_a < 0.1$	$0.001 < M_a < 0.1$
$B_a > 0$	$B_a > 0$
$G_a < 2$	$G_a < 7$

Table 5.1: Stability ranges

These ranges are determined assuming that the VSM stiffness does not overpass $k_{vsm} < 20\ \frac{\text{Nm}}{\text{rad}}$. If the stiffness is higher, coupled stability cannot be guaranteed, and the low-level control algorithm would be adjusted to achieve a stable HRI.

In this stability analysis, the low-level controller designed in Chapter 4 has been used without checking how different low-levels controllers would affect the coupled exoskeleton stability. Since the low-level control algorithm has a significant impact on the HRI stability, a better approach would be to design the low-level control while checking the coupled stability and its performance, using, for example, an optimization algorithm weighting stability versus performance (Buerger and Hogan, 2007).

As a last remark, it has to be mentioned that a linear stability analysis approach has been used to study a highly nonlinear system. So even though a wide range of possible cases has been considered, the nonlinear stability cannot be completely guaranteed.

5.1.2 Adaptive Admittance Filter

One of the issues with the proposed admittance filter is that the admittance filter will always have the same virtual dynamics regardless of the payload. In some scenarios, this is not an issue, but if the payload is unknown or varies over time, the exoskeleton might stop assisting its user and instead restricts the motion, such that the user feels the designed virtual dynamics. To avoid this issue, an adaptive admittance filter could be implemented. A basic idea has been used to adapt the admittance filter.

This approach is based on the payload estimation, either from an extended Kalman filter

or from the model based estimation from Section 4.3. Basically, this estimation is used to calculate the new admittance stiffness and inertia - if information about its mass is obtained.

$$M_a = \frac{J_p}{k_0} \tag{5.7}$$

$$G_a = \frac{\tau_{pay}}{k_0} \tag{5.8}$$

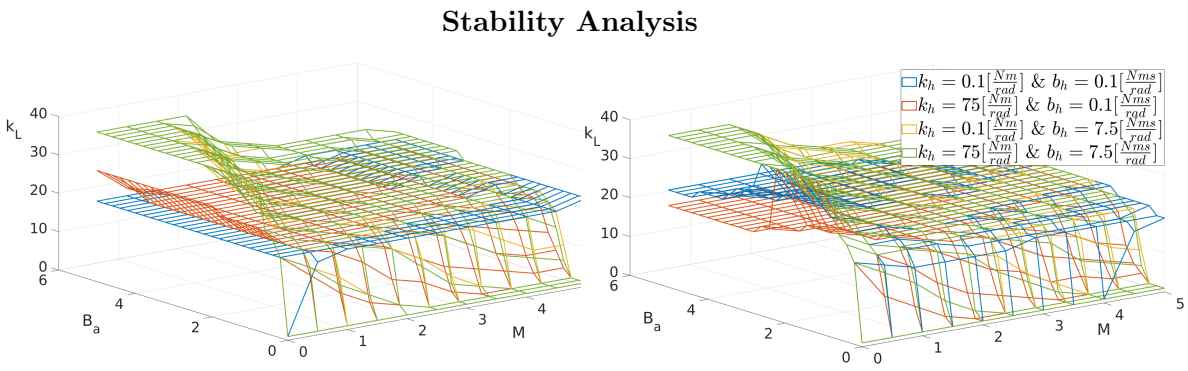
while keeping the admittance filter's dissipative parameter constant, B_a, k_0 is the desired advantage. Using this simple adaption of parameters based on the estimation already used by the low-level control, the admittance filter will try to track the desired advantage, which can be defined as

$$k_0 = \frac{\tau_{act}}{\tau_{pay}} \tag{5.9}$$

By using this approach, the desired advantage can be set, even for unknown or varying payloads. Besides, assuming a good payload and inertia estimation is procured, the system would have the benefit that for any attached payload, M, the admittance filter would never overpass its stiffness, G_a, limit seen in Figure 5.5. Making the whole control algorithm more robust to different configurations.

Stability

The stability of the coupled HRI exoskeleton with an adaptive admittance filter is then studied. This study has been done using the same approach as for the constant admittance filter. However, some of the design parameters existing for the normal admittance filter are adapted, meaning fewer possible configurations exist, as only the admittance damper is a tunable parameter. The human arm model has been described using the same boundaries as explained in Eq. (5.5).



(a) $m_h = 0.001$ kgm^2 (b) $m_h = 0.1$ kgm^2

As mentioned, the only real arbitrarily tuned parameter is the admittance damper, B_a, and from this analysis, it can be seen that if it is bigger than zero, the coupled system will have a stable behavior. However, in this analysis, no considerations of a possible error in the payload estimation are considered. Taking this into account, the adaptive admittance filter is expected to be more prompt to become unstable for bad payloads estimations and noisy signals.

5.1.3 Simulations

Using the developed model in Chapter 3, different simulations of the control algorithms have been done, showing the coupled exoskeleton performance with its user. This provides more information about the expected performance in real life, but still, due to all the approximations done and the complexity of the human arm, a discrepancy is expected, and adjustments will be necessary when implementing in the real system. For all simulations, the used low-level control has been developed and discussed previously in Chapter 4 and is the same used in the real system.

The scenario that has been simulated is for an exoskeleton payload equivalent to 4 kg located at its end-effector, and the user produces voluntary torque equivalent to a sinusoidal with an amplitude of 2 Nm, a frequency of 1 $\frac{\text{rad}}{\text{s}}$ and is biased by +2 Nm. The bias has been introduced so that the simulation takes place on a range of plausible human arm positions, as the elbow cannot have negative angles, with respect to the previous reference frames defined in Chapter 3. The coupled HRI has been computed for the two proposed control algorithms: a constant admittance filter and an adaptive admittance filter. The former has been designed such that the user feels an exoskeleton with a weight equivalent to 2 kg; basically, it halves the user's effort. Its parameters then are defined exactly as 5.1

$$\begin{aligned} M_a &= \frac{J_p}{2} \\ B_a &= 2 \\ G_a &= \frac{\tau_{pay}}{2} \end{aligned} \tag{5.10}$$

The adaptive admittance filter has been designed so that an advantage of 0.5 can be achieved in every given second. In other words, the admittance potential, G_a, is varied with respect to this goal.

For the first performance simulations the exoskeleton payload is changed after 20 seconds to a payload equivalent to 2 kg on the end-effector.

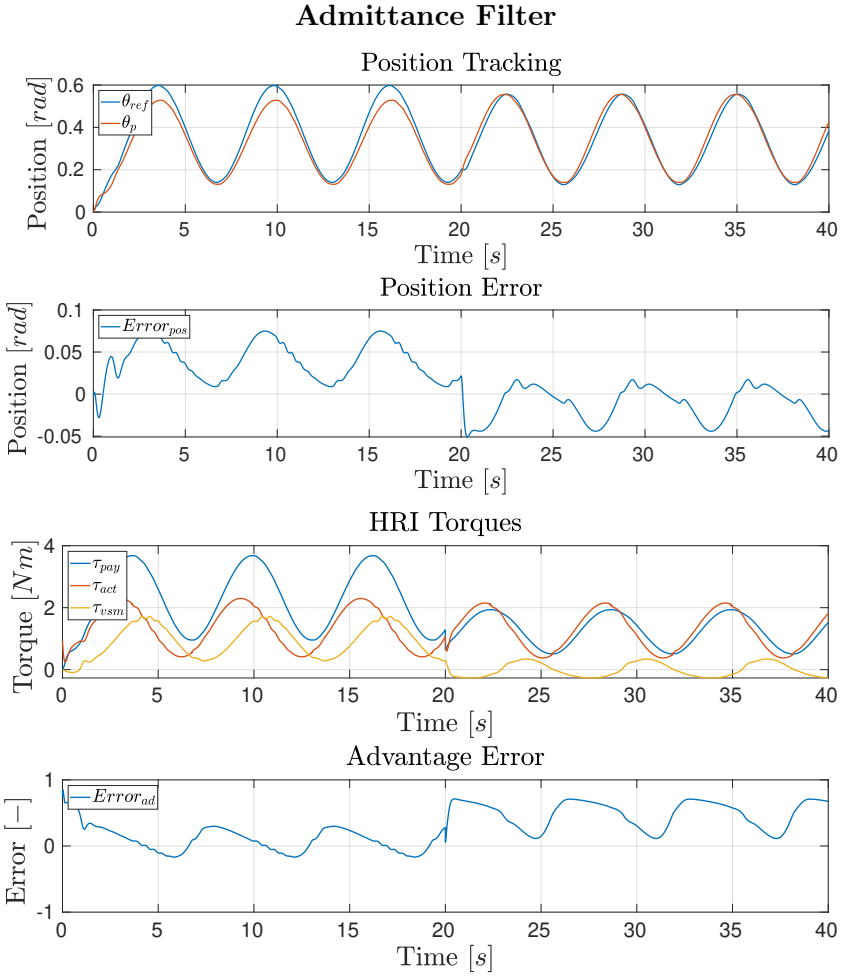


Figure 5.7: Admittance filter performance

The admittance filter can reshape the coupled HRI exoskeleton so that its user feels the designed virtual dynamics. The admittance filter reference represents these virtual dynamics, and fairly good tracking can be observed. The tracking has a better performance for a system in which the exoskeleton dynamics are already closer or equal to the virtual dynamics. In which case, the system behaves as a transparent exoskeleton - low impedance. This is because the possible error introduced by the low-level control is of a lower magnitude, given the lower impact necessary to make on the exoskeleton to achieve the desired behavior. Apart from analyzing the position tracking performance, the behavior of the HRI torques is of huge importance in assistive exoskeletons. For the first 20 seconds, as mentioned earlier, the exoskeleton payload is 4 kg. Therefore because of the desired virtual dynamics, the user must be able to move the payload to its desired position by just doing half of the payload torque, τ_{pay}. As can be seen from the graphs, the error advantage oscillates quite a bit around a mean error close to zero. This might be because not all the exoskeleton's dynamics are halved but just the inertia and the exoskeleton payload, while the admittance damper, B_a, has been manually tuned. If set as half of its real value, a better advantage error could be obtained. Nonetheless, the system might lose some stability margin. If having a lower frequency human active force, τ_h, the impact of the inertial and damper terms would be smaller, and a better advantage error would be

achieved. For the last 20 seconds, the exoskeleton dynamics are already close to the virtual dynamics. Therefore, the exoskeleton will try not to produce a torque on the system, so the user feels these dynamics; basically, the exoskeleton behaves in a transparent mode.

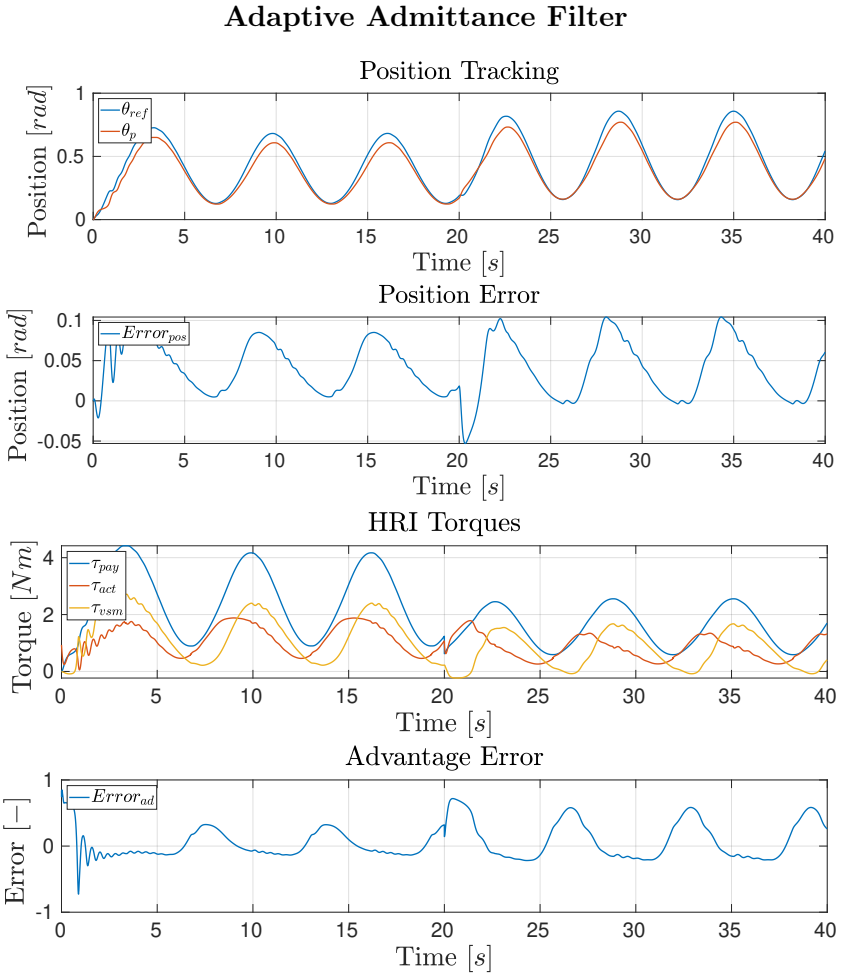


Figure 5.8: Adaptive admittance filter performance

The goal of the adaptive admittance filter is different from the constant admittance filter, being to obtain the desired advantage no matter what the exoskeleton payload is. As can be seen for the first 20 seconds, a better advantage tracking is achieved compared to the constant admittance filter. However, the difference in performance and goal of the two different control algorithms can be noticed in the last 20 seconds, for which the adaptive admittance filter reduces the user's feeling of the exoskeleton payload to the desired advantage, instead of turning the system into transparent mode.

Robustness

The proposed control algorithms are based on the derived models in Chapter 3. These models are not perfect. Therefore, a robustness analysis takes importance on the system. For this analysis, both disturbances and sensor noise of the five available sensors have been considered. The sensor noises have been added using white noise with different variances and limited maximum frequencies.

-	Limited Frequency	Variance
$\dot{\theta}_p$	$< 100[Hz]$	0.06
θ_p	$< 100[Hz]$	0.06
$\dot{\theta}_m$	$< 100[Hz]$	60
θ_m	$< 100[Hz]$	15
τ_{act}	$< 100[Hz]$	0.2
Disturbance	$< 20[Hz]$	1

Table 5.2: Noise and disturbance characteristics

The robustness analysis of the admittance filter can be seen in Figure 5.9. This performance suggests that the control algorithm can work regardless of some noise and disturbances applied to it. These disturbances represent discrepancies with the real system and other unknown influences on the exoskeleton. The noise and disturbances that can be handle are described in Table 5.2. Even if the controller can handle this noise and disturbances, obtaining a good position and advantage tracking, the effect on the user commodity has the same importance, and big oscillations exist on the exoskeleton torque τ_{vsm}.

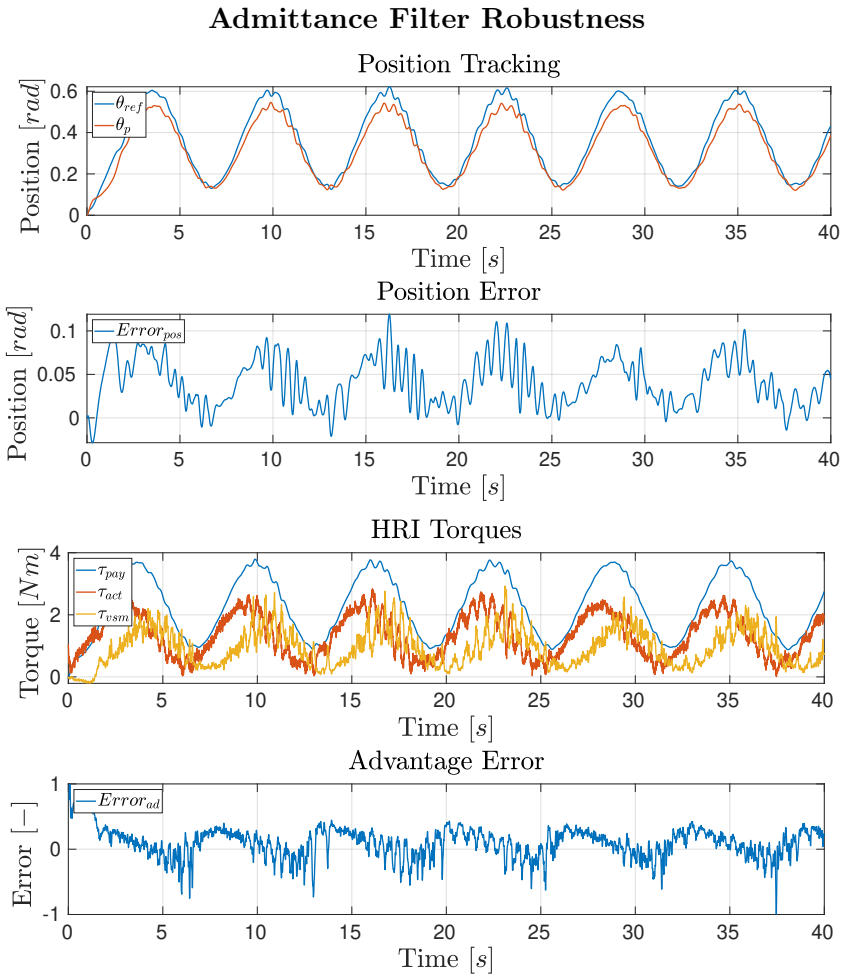


Figure 5.9: Admittance filter robustness on Matlab & Simulink simulation

The adaptive admittance-based controlled exoskeleton has a similar response behavior,

yielding the same conclusion of the issue concerning the user's commodity.

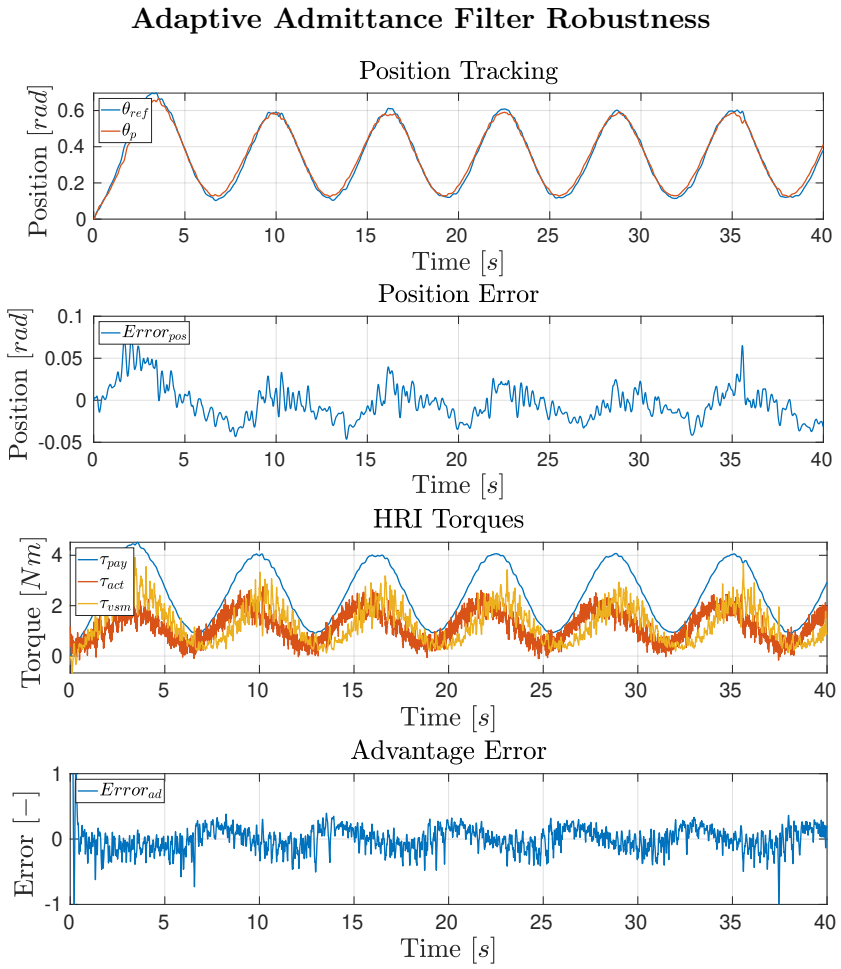


Figure 5.10: Adaptive admittance filter robustness performance

5.2 Human Payload

In the case that the human carries directly the payload, the control algorithm goal is still the same, achieving some assistance from the exoskeleton. However, the control algorithm slight differs from the previously mentioned.

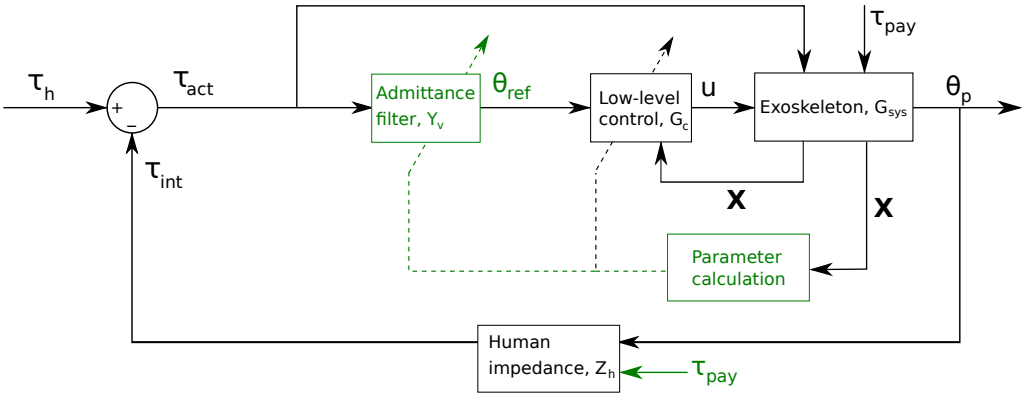


Figure 5.11: Schematic of the exoskeleton with the payload grabbed by the user - marked in green

As can be seen from Figure 5.11 now, the payload affects and reshapes the intrinsic torque produced by the human arm. It changes the human impedance, Z_h. Then by using the same approach, only the exoskeleton dynamics can be affected by the virtual dynamics. However, this will not provide any advantage to the user with the mass that is grabbed. This is due to, for the exoskeleton to help the user, it has to produce a torque, τ_{vsm}, in the admittance filter reference, θ_{ref}, direction. Moreover, by its definition, the reference is defined by the interaction torque, τ_{act}, indeed the reference set by the admittance filter follows the interaction torque, τ_{act}. So as for this configuration, the payload is grabbed by the user, where the interaction torque, τ_{act}, and the VSM torque, τ_{vsm}, approximately will be the same but of opposite sign, as can be seen in Figure 5.12.

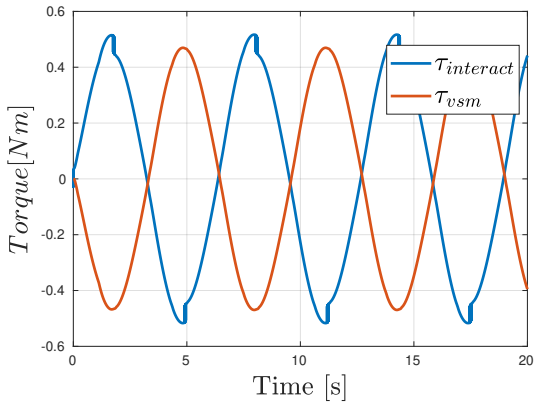


Figure 5.12: HRI using the previous control algorithm

This demonstrates that the exoskeleton is not able to assist the user by using the previously developed algorithm. To do so, the exoskeleton must take care of a part of the intrinsic arm-payload torque, τ_{int}; indeed τ_{vsm} must work on the direction set by the admittance filter θ_{ref}.

A similar approach to Islam and Bai (2019) has been used for this scenario. This approach needs payload estimation. However, in this project, the payload estimation has not been looked into. Instead, the goal has been to prove this concept for soft exoskeletons and extend the study and discuss its stability.

To achieve the assistive exoskeleton help the HRI torque, τ_{act}, must be redefined, this is accomplished by biasing the real interaction torque by a value, which is the assistive torque. This reshapes the input to the admittance filter, by defining

$$\tau_b = \tau_{act} + k\tau_{pay}\sin\theta_p \tag{5.11}$$

where the modified advantage, k, is defined as

$$k = k_0 + \frac{G_a}{\tau_{pay}} \tag{5.12}$$

this adapts the desired advantage, k_0, so that its achieved no matter the admittance filter stiffness G_a, assuming of an ideal low-level control, indeed $\theta_p = \theta_{ref}$, and steady-state has been used.

Introducing this bias of interaction torque, τ_b, the exoskeleton state-space from interaction torque to forearm position can be defined as

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}\mathbf{X} + \mathbf{B}\tau_{act} \\ \theta_p &= \mathbf{C}\mathbf{X} \end{aligned} \tag{5.13}$$

where

$$\mathbf{X} = \begin{bmatrix} \dot{\theta}_p & \theta_p & \dot{\theta}_m & \theta_m & \dot{\theta}_{ref} & \theta_{ref} \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} -\frac{B_{vsm}+B_p}{J_p} & -\frac{m_p l_p g + k_L}{J_p} & 0 & \frac{k_L}{nJ_p} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{K_1 k_t}{J_{tot}} & \frac{-K_2 k_t + n\beta k_L}{J_{tot}} & -\frac{K_3 k_t + B_m + \beta B_{vsm}}{J_{tot}} & -\frac{K_4 k_t + \beta k_L}{J_{tot}} & 0 & \frac{F k_t}{J_{tot}} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & k\frac{\tau_{pay}+k_h}{M_a} & 0 & 0 & -\frac{B_a}{M_a} & -\frac{G_a}{M_a} \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.14}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{J_p} & 0 & 0 & 0 & \frac{1}{M_a} & 0 \end{bmatrix}^T$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and by adding the payload on the human arm impedance, Z_h, the coupled HRI exoskeleton is completely defined

$$\tau_{int} = m_h\ddot{\theta}_p + b_h\dot{\theta}_p + (k_h + \tau_{pay})\theta_p \tag{5.15}$$

This approximated model of the real system is used to develop a stability analysis, and the nonlinear model develop in Chapter 3 with the addition of control law is used for simulations of performance.

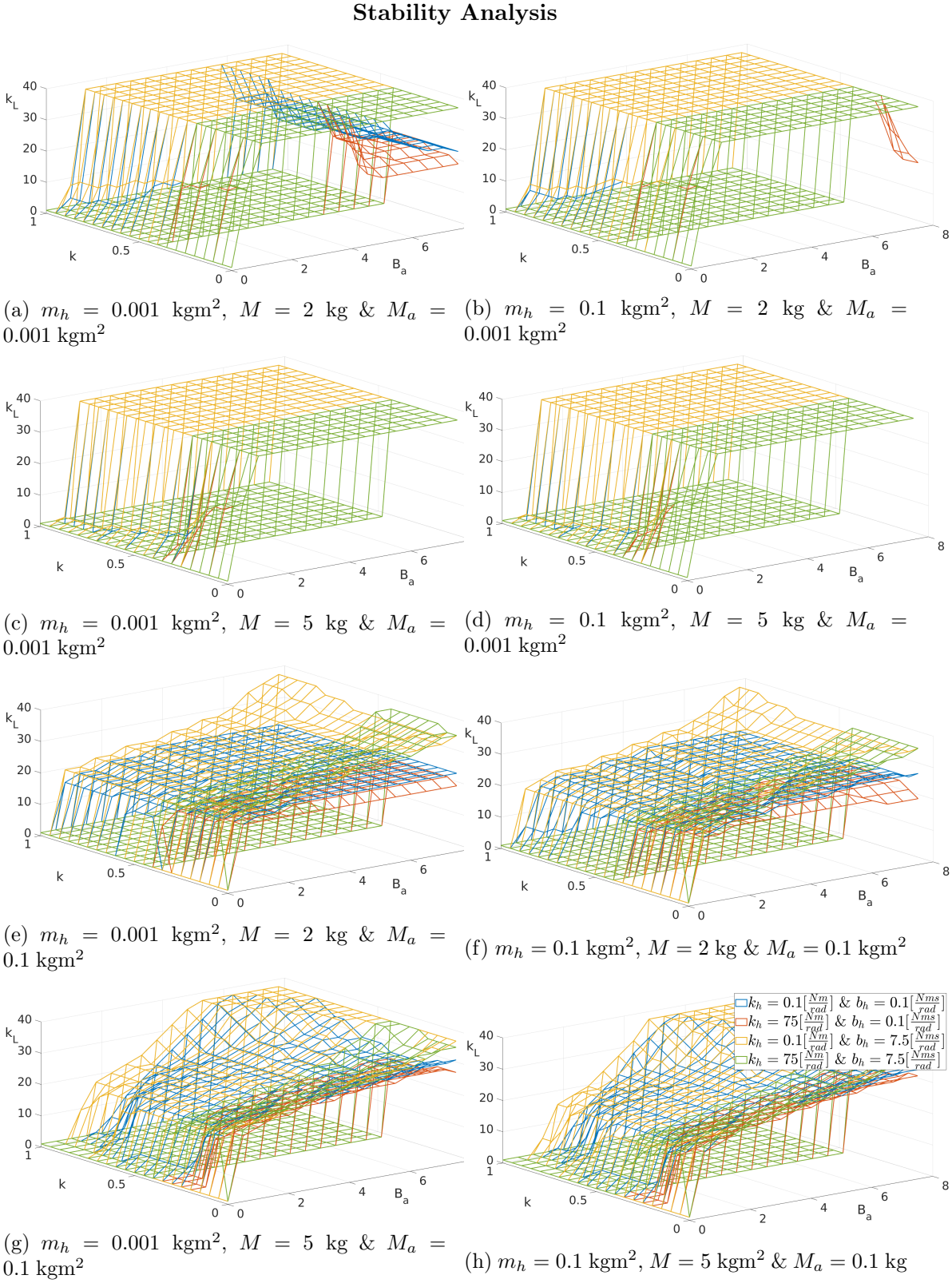
5.2.1 Stability

As previously, a numerical stability analysis for the different bounded exoskeleton and HRI dynamics has been used. For this case, another parameter is added to the computation: the desired advantage, k, which is considered to have a bigger impact on the system's stability and performance than the admittance potential, G_a.

$$0 \leq k \leq 1 \tag{5.16}$$

ranging from zero to full payload assistance. The rest of the parameters have the same boundaries as given in Eqs. (5.5) & (5.6).

As mentioned earlier an ideal payload estimation is assumed, as well as the extreme parameters is used to simplify the analysis while still having a good foundations.



(a) $m_h = 0.001$ kgm^2, $M = 2$ kg & $M_a = 0.001$ kgm^2

(b) $m_h = 0.1$ kgm^2, $M = 2$ kg & $M_a = 0.001$ kgm^2

(c) $m_h = 0.001$ kgm^2, $M = 5$ kg & $M_a = 0.001$ kgm^2

(d) $m_h = 0.1$ kgm^2, $M = 5$ kg & $M_a = 0.001$ kgm^2

(e) $m_h = 0.001$ kgm^2, $M = 2$ kg & $M_a = 0.1$ kgm^2

(f) $m_h = 0.1$ kgm^2, $M = 2$ kg & $M_a = 0.1$ kgm^2

(g) $m_h = 0.001$ kgm^2, $M = 5$ kg & $M_a = 0.1$ kgm^2

(h) $m_h = 0.1$ kgm^2, $M = 5$ kgm^2 & $M_a = 0.1$ kg

Figure 5.13: Numerical stability analysis

From this analysis, different directives to the design of the control algorithm can be obtained. First, as can be seen from the stability analysis, it is essential that the admittance filter's dissipative coefficient, B_a, is not zero as this yields to a coupled unstable system.

Secondly, the inertia of the admittance filter, M_a, is desired to be low. This will provide a broader range of stiffness for which the exoskeleton is stable. Nonetheless, higher admittance inertia is not prompt to make the system unstable, meaning that it can be tuned to the desired performance but keeping stability considerations in mind. Lastly, the admittance filter potential, G_a, does not considerably affect the system's stability due to the modified advantage, k, compensates for it. Apart from information about the admittance filter, the stability analysis shows for which advantages, k_0, the coupled system will still be stable.

As mentioned earlier, the extremes of the different parameters are used for the stability analysis. Some of these extremes are not the most realistic ones, as for example a human arm stiffness of $k_h = 75\ \frac{\text{Nm}}{\text{rad}}$ and $b_h = 0.1\ \frac{\text{Nms}}{\text{rad}}$. So a higher advantage could be achieved. However, a conservative approach for HRI is usually preferred as safety is a top priority. Taken this into considerations the different limits can be obtained, considering that the system must be stable for stiffness lower than $k_{vsm} = 20\ \frac{\text{Nm}}{\text{rad}}$

$M = 2$	$M = 5$
$0.001 < M_a < 0.1$	$0.001 < M_a < 0.1$
$B_a > 0$	$B_a > 0$
$k_0 < 0.4$	$k_0 < 0.4$

Table 5.3: Stability range

5.2.2 Simulations

Once again, the proposed modified interaction admittance-based control algorithm has been computed in numerical simulations, providing another source of information to its analysis before implementation to the real system. For this simulation, a payload equivalent to 2 kg is grabbed by the user. Its estimation is assumed to be ideal for the high-level control, and the voluntary active torque produced by the user can be described as a sine wave with a frequency of 1 $\frac{\text{rad}}{\text{s}}$, an amplitude of 2 Nm while being biased by +3 Nm. Then the high-level control goal is designed such that an assistance of $k_0 = 0.3$ can be achieved. The used admittance filter parameters are

$$\begin{aligned} M_a &= 0.01 \\ B_a &= 2 \\ G_a &= 0 \end{aligned} \tag{5.17}$$

In Figure 5.14 the coupled HRI exoskeleton performance is shown. Regarding the position tracking, a displacement between the reference and the actual user's forearm is noticeable for large angular positions. This is due to an error in the payload estimation used by the low-level control. By using another estimation method, for example, an extended Kalman filter (EKF), better position tracking could be obtained. Apart from controlling the forearm position, the modified interaction admittance filter also controls the torque applied by the exoskeleton such that the desired advantage can be accomplished. From the numerical simulations, different desired characteristics from the control algorithm can be noticed. First, how, by biasing the interaction torque, τ_{int}, it is achieved that exoskeleton torque, τ_{vsm}, can work in the same direction as the forearm position, θ_p, providing then

an assistance to the user. Nevertheless, as was shown previously, the HRI interaction torque, τ_{act}, will still be approximately equal but of opposite sign to the VSM torque, τ_{vsm}, suggesting a possible sensorless approach, which is discussed later in this section. Regarding the advantage, it is seen that the average value will be around the desired. However, for high-frequency interaction forces - like this one - a significant error will be present. Nonetheless, this error goes to zero for steady-state.

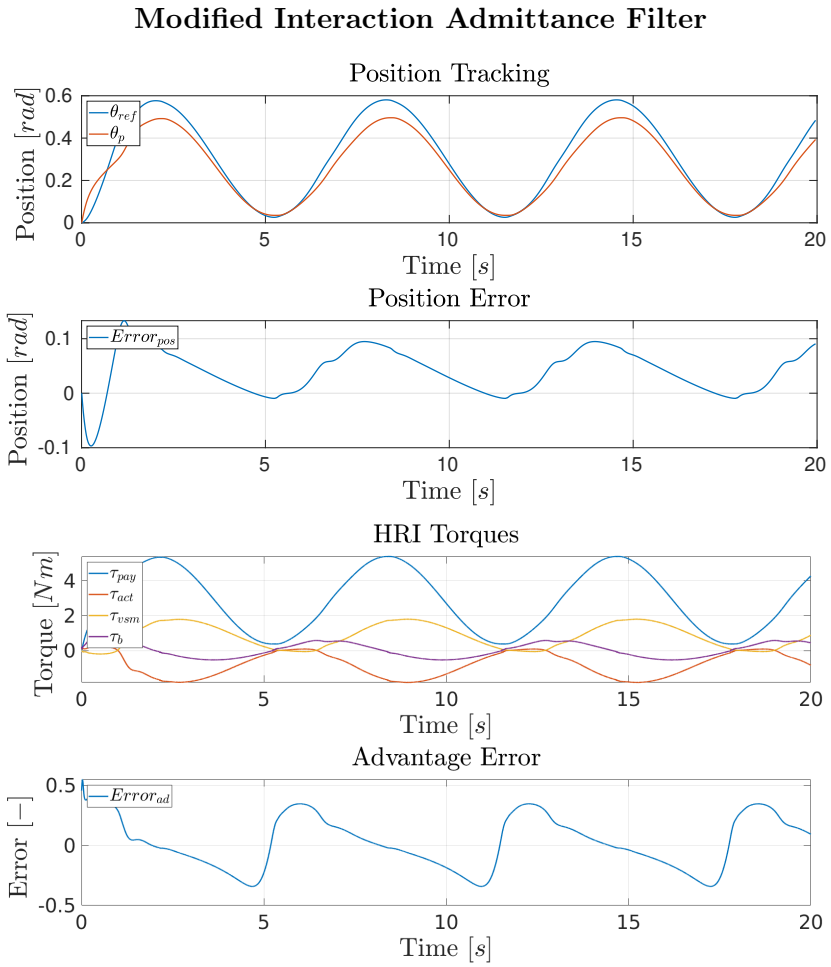


Figure 5.14: Modified interaction admittance filter performance

Robustness

The robustness of the modified interaction admittance filter is analyzed in the same manner than before for the exoskeleton payload scenario. The introduced noise and disturbance is described in Table 5.4.

-	Limited Frequency	Vriance
$\dot{\theta}_p$	$< 100[Hz]$	0.1
θ_p	$< 100[Hz]$	0.1
$\dot{\theta}_m$	$< 100[Hz]$	60
θ_m	$< 100[Hz]$	15
τ_{act}	$< 100[Hz]$	0.2
Disturbance	$> 20[Hz]$	0.8

Table 5.4: Noise and disturbance parameters used in the robustness simulations

As can be seen from this numerical simulation, the system has higher robustness than for the exoskeleton payload scenario. This is in part due to the user grabbing a mass the human feedback, τ_{int}, have slower dynamics than if this payload was zero. This would also mean that the robustness of the coupled HRI exoskeleton is dependent on the actual, τ_{pay}. Moreover, as mentioned for the previous numerical simulation, the estimation of τ_{pay} is assumed ideal, which in reality has to be calculated by some mean, and an error in this calculation could yield a less robust performance.

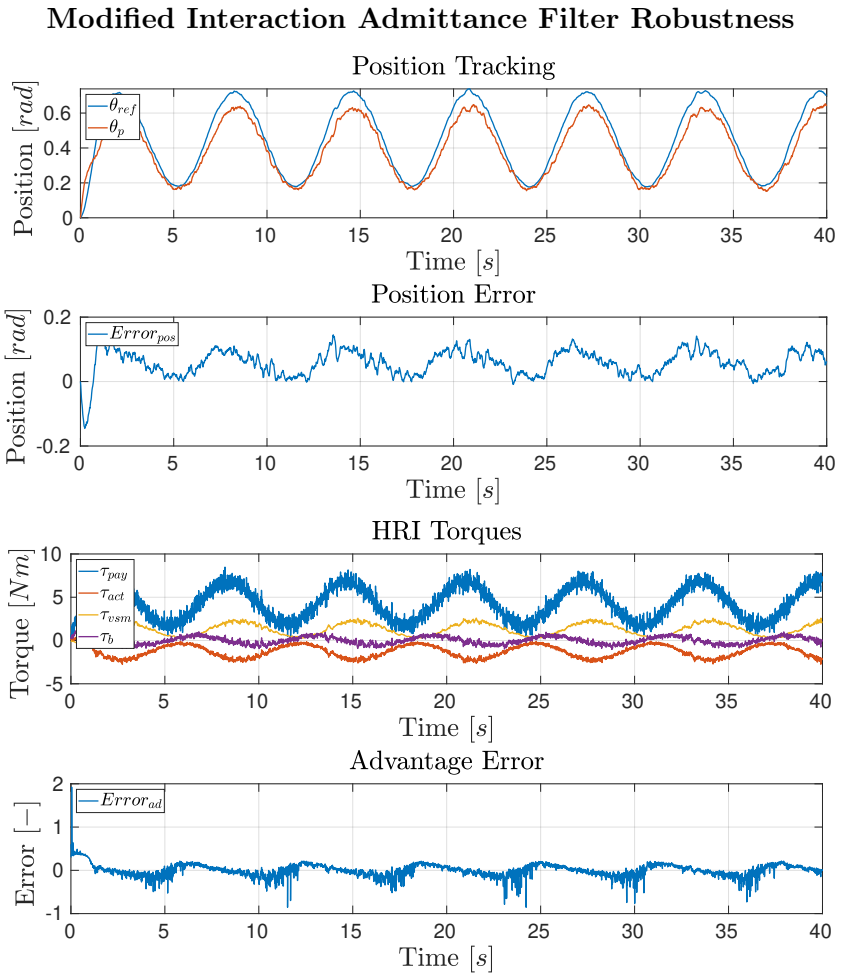


Figure 5.15: Admittance filter performance for the assistive robustness simulation

Sensorless modified interaction admittance filter

Based on the fact that the exoskeleton has a lightweight arm, the real interaction torque, τ_{int}, and the VSM torque, τ_{vsm}, are approximately equal but opposite. For this reason, a 'sensorless' approach could be used to control the HRI torque of the coupled system. By 'sensorless,' it is meant that the HRI torque is not sensed and is approximated by the VSM torque estimation, τ_{vsm}. Nonetheless, the low-level control states still need to be sensed, and the actual payload on the human arm must be estimated. This approach provides a sensor reduction, as well as weight-reducing and removal of its influence on the dynamics of the system.

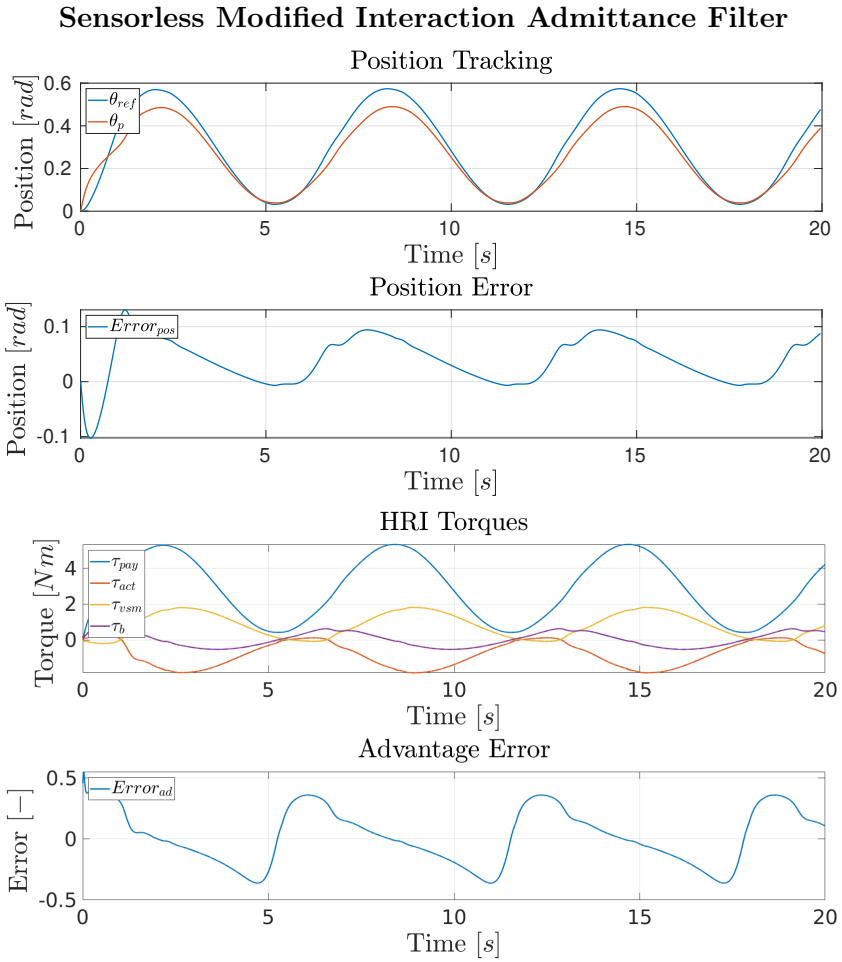


Figure 5.16: Sensorless control simulation

The performance of the sensorless approach can be seen to be practically the same as sensing the HRI torque.

Implementation 6

In this chapter, the implementation of the control algorithm is presented. Also, the test setup and components are explained. Important parts of the code associated with the microcontroller are also presented.

6.1 Test Setup

The test setup consists of the exoskeleton arm itself, an Arduino DUE that handles all of the calculation and control of the exoskeleton, and an ESCON 50/5 servo controller for converting the controller input to a usable signal and also to make sure that the motor settles at the given input. The test setup can be seen in Figure 6.1.

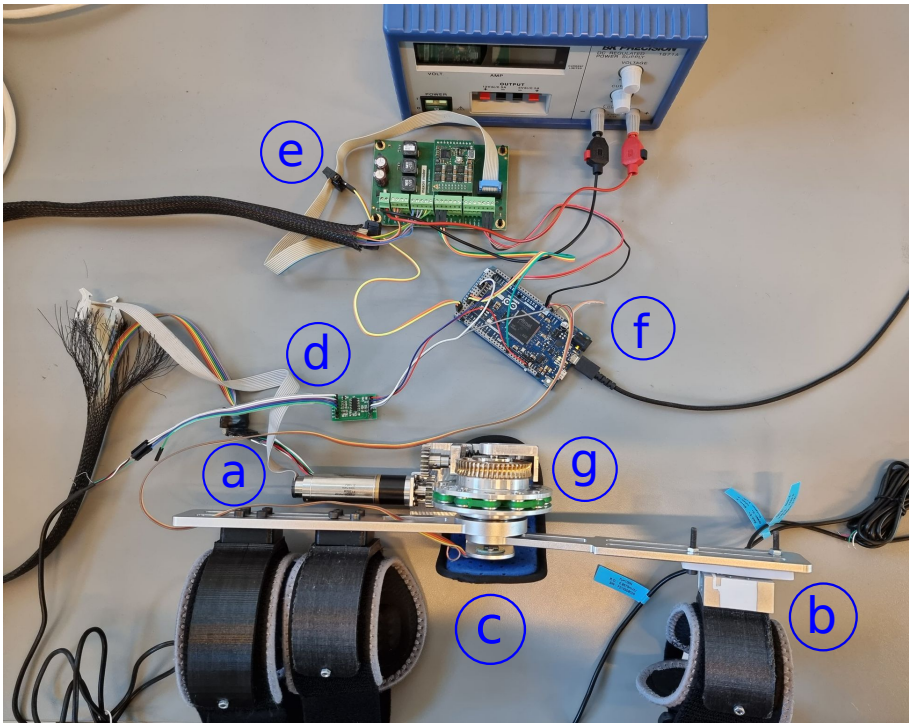


Figure 6.1: Overview of the test setup

The exoskeleton is made up of a Maxon 90 W EC-4 pole at ⓐ, which as mentioned is controlled by the ESCON 50/5 servo controller at ⓔ. The motor is equipped with an Encoder MR, Type ML, for measuring its position. The motor has an internal gearbox with a gear ratio of 5.4. The output shaft is connected to a spur gear with a gear ratio of 1. The motor is connected to the VSM at ⓖ through a worm gear with a gear ratio of

50. This worm gear has the characteristic that it makes the system self-locking, meaning that torque transferred from the VSM to the motor is pure friction. This implies that the exoskeleton cannot move if the motor is not moving - disregarding the deflection due to the stiffness characteristics of the VSM.

The VSM has six pulley sections - with three pulleys each, connected by a compliant element. The outer flange is connected to the worm gear, and the inner flange is connected to the exoskeleton arm, where the user will interact with the robot.

On the exoskeleton arm, a magnetic RLS RMB20 absolute encoder at ⓒ is attached, which measures the position of the exoskeleton arm. A 3-axis load cell 0-100N Multi-axis force sensor from Forsentek at ⓑ is positioned at the end of the exoskeleton arm, which is amplified by a SparkFun Load Cell Amplifier - HX711 at ⓓ. This load cell is also connected to the cuff, where the user wears the exoskeleton, and is used to measure the interaction force, F_{act}, which is feed back into the Arduino at ⓕ and used for calculating the reference to the low-level controller.

6.1.1 Electrical Wiring Setup

To give an overview of how the wires are connected a wiring diagram will be presented and the connections and signals will be explained. The wiring diagram can be seen in Figure 6.2.

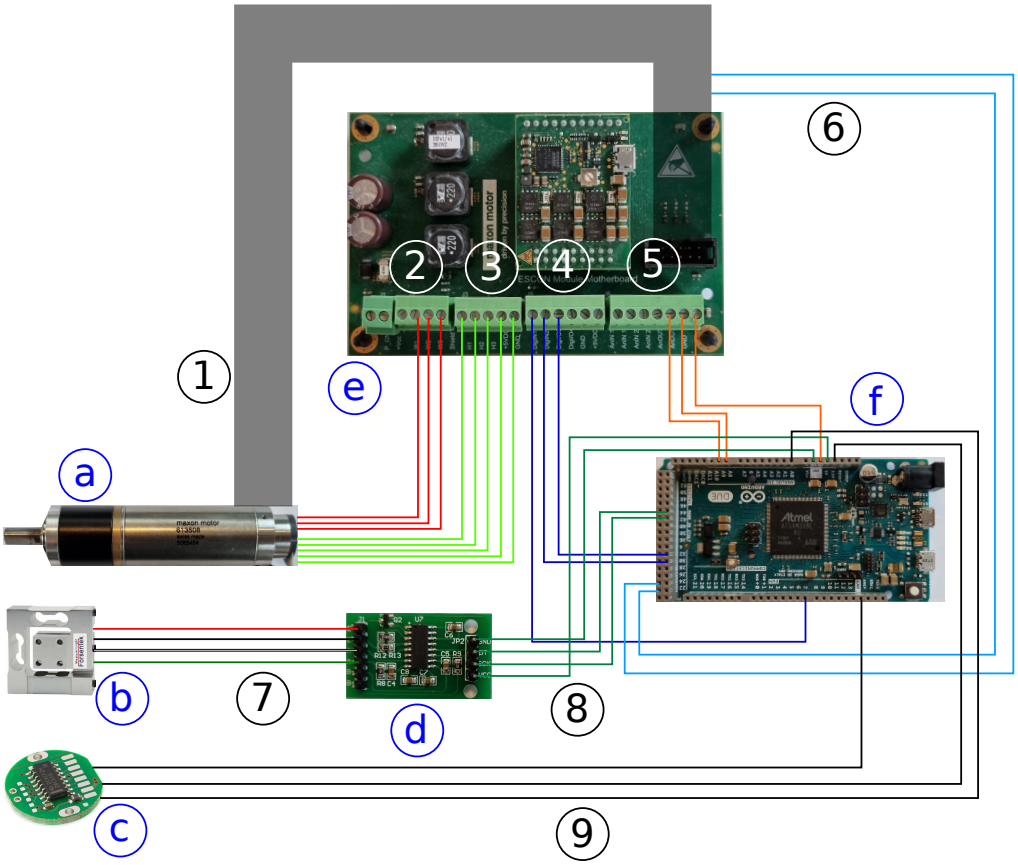


Figure 6.2: Wiring diagram of the setup. Blue letters represent a component - corresponding to Figure 6.1 -, and numbers represent a group of wires

The gray band at ① represents the ribbon cable that carries the position encoder data from the motor at ⓐ, to the ESCON 50/5 servo controller at ⓔ. These data are also used in the Arduino at ⓕ for motor position and velocity calculation - used as feedback for the low-level controller - and are represented by the light blue wires at ⑥. The red wires at ② are the currents for the three motor windings. The green wires at ③ represent the hall sensor data and the power for it. The blue wires at ④ are the digital input signals to the motor controller. These signal are calculated in the Arduino and include: motor enable, motor direction, and PWM motor input. These are also explained in the flow chart in Figure 6.3. The orange wires at ⑤ carries analog signals of the velocity and current, measured in the motor and send to the Arduino. These signal will not be used, since the velocity is already calculated from the motor position Encoder MR, Type ML, which is easier to calibrate - see Section 6.3. The Forsentek 3-axis load cell at ⓑ sends data to the SparkFun Load Cell Amplifier at ⓓ. The wires at ⑦ carries these signals, and are color coded after the real colors of the wires. The dark green wires at ⑧ sends the amplified signals to the Arduino where they are used in the high-level control to generate a position reference for the low-level controller. Lastly, the black wires at ⑨ carries the signals from the magnetic RLS RMB20 absolute encoder at ⓒ to the Arduino as feedback for the low-level controller.

6.1.2 Setup of ESCON 50/5 Servo Controller

The setup of the motor controller board is largely based on Pedersen and Murcia i Matute (2020) - using ESCON studio. Since the designed low-level control is a position control, it gives a motor torque input. The ESCON is, therefore, setup in current control mode, where the torque input from the low-level control is scaled by the factor $k_t = 0.0281\ \frac{\text{Nm}}{\text{A}}$. The relation between torque and current is given by:

$$\tau = k_t I \quad \longrightarrow \quad I = \frac{\tau}{k_t} \tag{6.1}$$

As mentioned in Pedersen and Murcia i Matute (2020), the ESCON board has a problem where inputs lower than 1.59% duty cycle are not registered by the microcontroller. A new solution is proposed in this project, where the calculated input is scaled by 10% in the Arduino before the signal is sent to the ESCON 50/5. To compensate, the duty cycle calibration in ESCON Studio is changed such that a 10% input is defined as: $I = 0$ A. This essentially defines a new zero at 10% duty cycle input, meaning that the calculated duty cycle will never reach the 1.59% duty cycle threshold. This has the consequence of reducing the maximum possible input by 10% as well. However, the input is expected to be in the lower range more often than near the maximum.

6.2 Arduino Implementation

As mentioned, an Arduino DUE is the microcontroller used. The algorithm and code are very similar to the algorithm presented in Pedersen and Murcia i Matute (2020). Therefore only a summary will be made. For further details, see Pedersen and Murcia i Matute (2020). For calculating the input to the motor, an interrupt has been used. This interrupt occurs every 200 Hz. In this interrupt, the latest available sensor data - position and velocity of the motor and the exoskeleton arm and interaction force - is read. The reference is then

calculated, and from this reference, a motor input is calculated. The input is transformed to a duty cycle by

$$duty = \left(\frac{u}{u_{max}} + 10\% \right) P_{res} \tag{6.2}$$

where u is the input, u_{max} is the maximum allowed input, and P_{res} is the PWM resolution which is set to 12 bit: $P_{res} = 2^{12} = 4096$. The 10% addition comes from the solution described in Section 6.1.2. The sign of the input is also determined, which sets the direction of the motor independently from the input value. The algorithm can be seen in Figure 6.3, which is based on Pedersen and Murcia i Matute (2020).

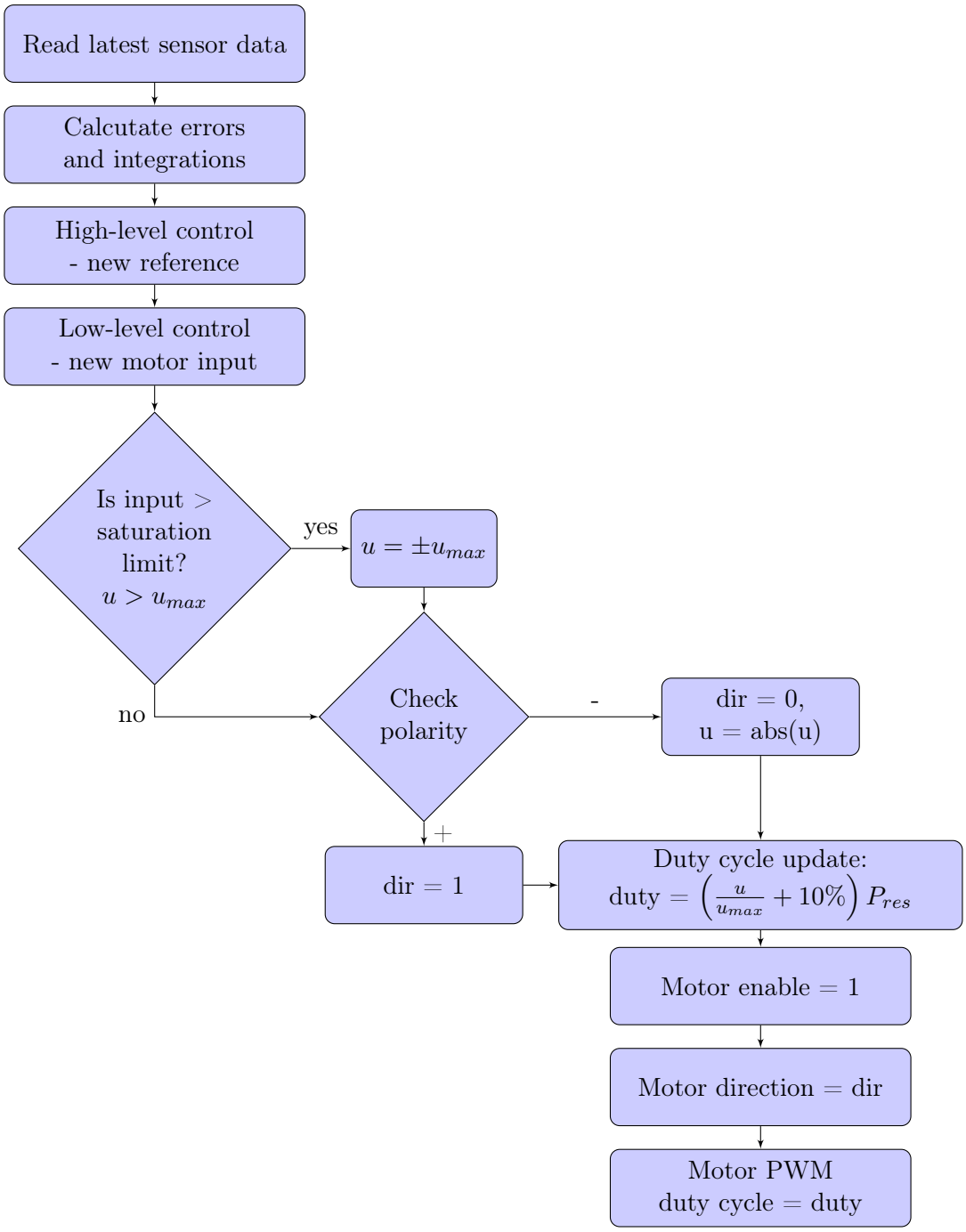


Figure 6.3: Flowchart of the algorithm used in the Arduino to control the exoskeleton (Pedersen and Murcia i Matute, 2020)

For evaluating the results, the sensor data and references are sent to a PC using serial communication, with a baud rate of 115200 Bd. The entire Arduino code can be seen in Appendix C.

6.3 Sensor Calibration

As mentioned, three sensors are used in the system: two encoders - one for motor and one for exoskeleton arm position - and a force sensor to measure the interaction force. These sensors send raw and unusable data, so they, therefore, have to be calibrated. The calibration process is similar to Pedersen and Murcia i Matute (2020) but will be briefly repeated here.

Exoskeleton position encoder

To measure the exoskeleton arm's position, a magnetic RLS RMB20 absolute encoder is used. Calibration is done using a digital protractor. The arm is moved to different positions, which are noted. At these positions, the raw encoder data is also noted. Linear regression is made on these two data sets, and the resulting polynomial is then the formula used to transform the raw data to usable position data in radians. The fit can be seen in Figure 6.4.

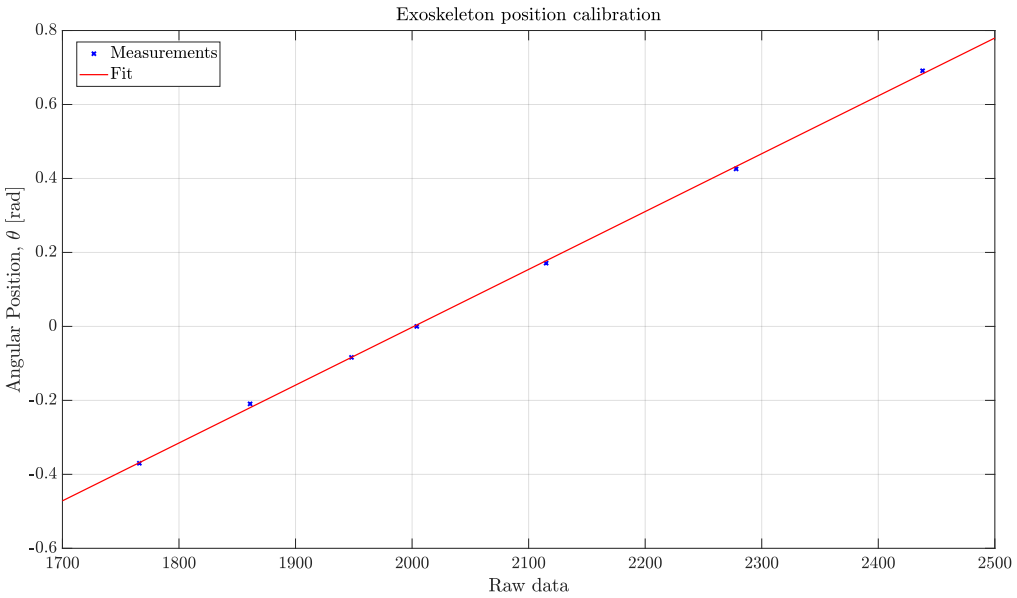


Figure 6.4: Exoskeleton arm position calibration

The resulting polynomial is then given by:

$$\theta_p = 0.001564 \cdot raw_p - 3.131 \tag{6.3}$$

Motor position encoder

Calibrating the motor position encoder is a little more tricky since it is difficult to measure the motor position - which spins 270 times faster than the exoskeleton arm. However, the relation between the motor position and the arm position can be used here. This relationship is defined as the deflection angle, and if the deflection angle is zero, then two positions are directly related through the approximation:

$$\theta_d = \frac{\theta_m}{n} - \theta_p \approx 0 \quad \Rightarrow \quad \theta_m \approx \theta_p n \tag{6.4}$$

where θ_d is the deflection angle, θ_m is the motor position, θ_p is the exoskeleton arm position, and n is the gear ratio. If the gravitational effect on the arm is kept to a minimum, then the approximation can be good enough to calibrate the motor position. The motor is then set in different positions, and the raw motor encoder data and the corresponding calibrated exoskeleton arm position data are noted. Linear regression is again used to fit the two data sets, where the resulting polynomial is used to transform from raw data to a position. The fitting can be seen in Figure 6.5.

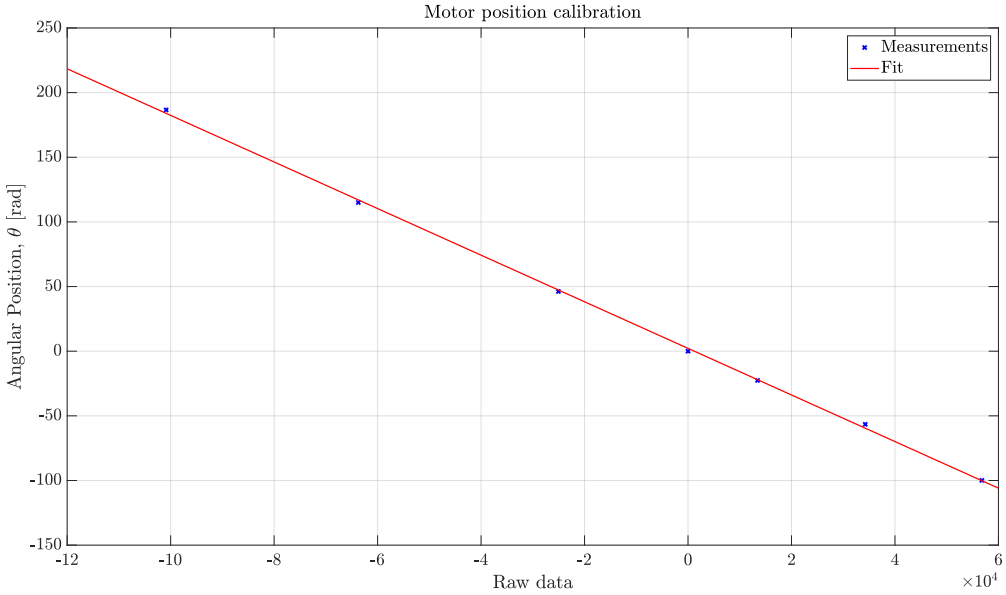


Figure 6.5: Motor position calibration

The calibration polynomial is given by:

$$\theta_m = -0.001802 \cdot raw_m + 2.179 \tag{6.5}$$

Force sensor

The force sensor is calibrated using a known weight attached to the load cell, which is attached to the exoskeleton. The exoskeleton arm position is set to $\frac{\pi}{2}$ - perpendicular to the ground - so the force sensor measures the total weight. The length from the axis of rotation to the center of mass of the weight is measured and is together with the standard gravity, g, multiplied with the mass of the weight. This gives the torque produced by the weight. The weight is then changed and the process repeated. A linear regression of raw and torque data is made, which gives the calibration polynomial. The fit can be seen in Figure 6.6.

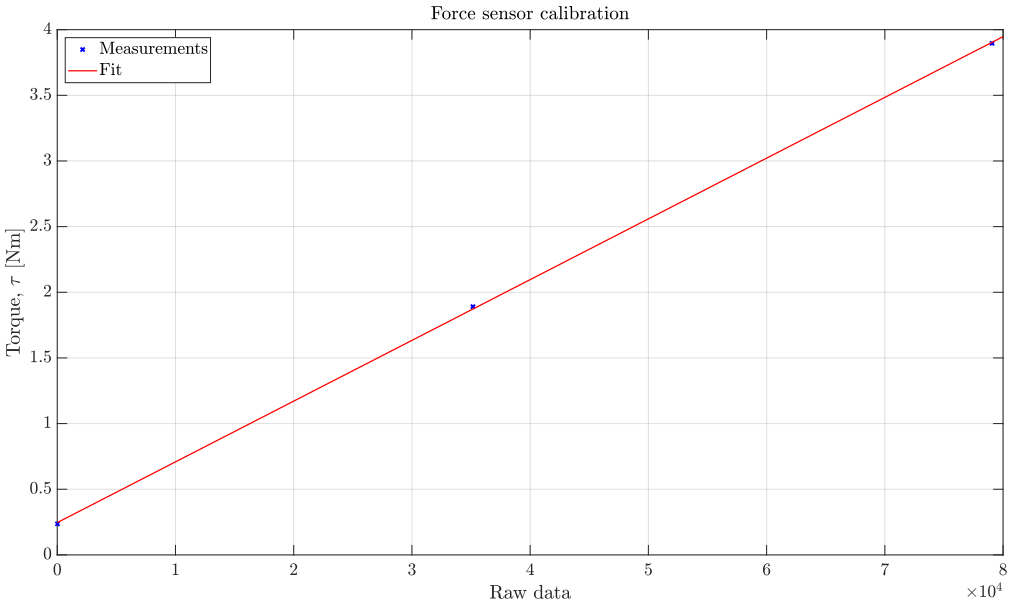


Figure 6.6: Force sensor calibration

The calibration polynomial is given by:

$$\tau_{act} = 4.626 \cdot 10^{-5} \cdot raw_{act} + 0.2458 \tag{6.6}$$

6.4 Implementation of EKF and Nonlinear Control

As mentioned in Section 4 the Nonlinear controllers and the EKF from Pedersen and Murcia i Matute (2020) have been attempted implemented. However, it was deemed not to be feasible due to the slow processing time of the Arduino DUE. In simulations, the EKF needs a sampling time of $t_s = 0.001$ s to estimate the states and the parameters satisfactorily. While in the Arduino, no lower than $t_s = 0.005$ s was obtained. This is because the microcontroller is not fast enough to carry out the calculations of multiple 6x6 matrices - including an inverse - in the sampling time given. After one sampling, the microcontroller will generate a new interrupt while all the calculations in the previous interrupt have not been completed, meaning that the motor will never run since a new input is never sent. This heavily limits the minimum possible sampling time to around $t_s = 0.005$ s, as mentioned.

Estimations from the EKF

To illustrate the problem with the high sampling time the estimation results will now be presented. A test was made with a sampling time of $t_s = 0.005$ s, a payload of $M = 1.1$ kg. The controller used is a tuned version of the LQI designed in Pedersen and Murcia i Matute (2020). The state estimations can be seen in Figure 6.7.

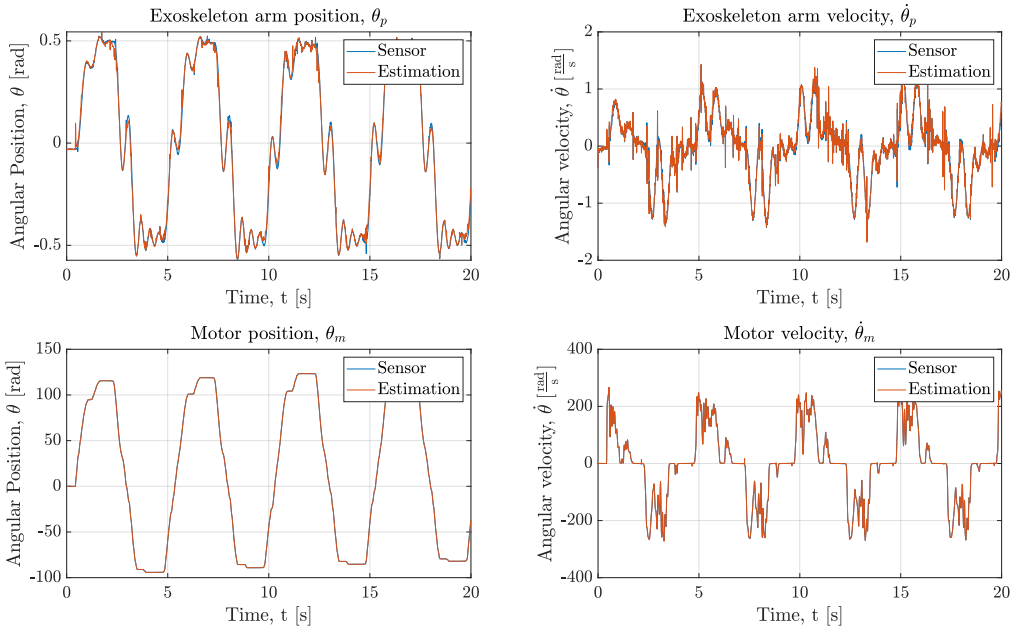


Figure 6.7: State estimation from the EKF

From the figure it can be seen that even though the the sampling time is 5 times large than the suggestion from the simulation, the EKF can estimate the state satisfactorily. However, when looking a the parameters - payload, and VSM stiffness and torque - the problem arises, see Figure 6.8.

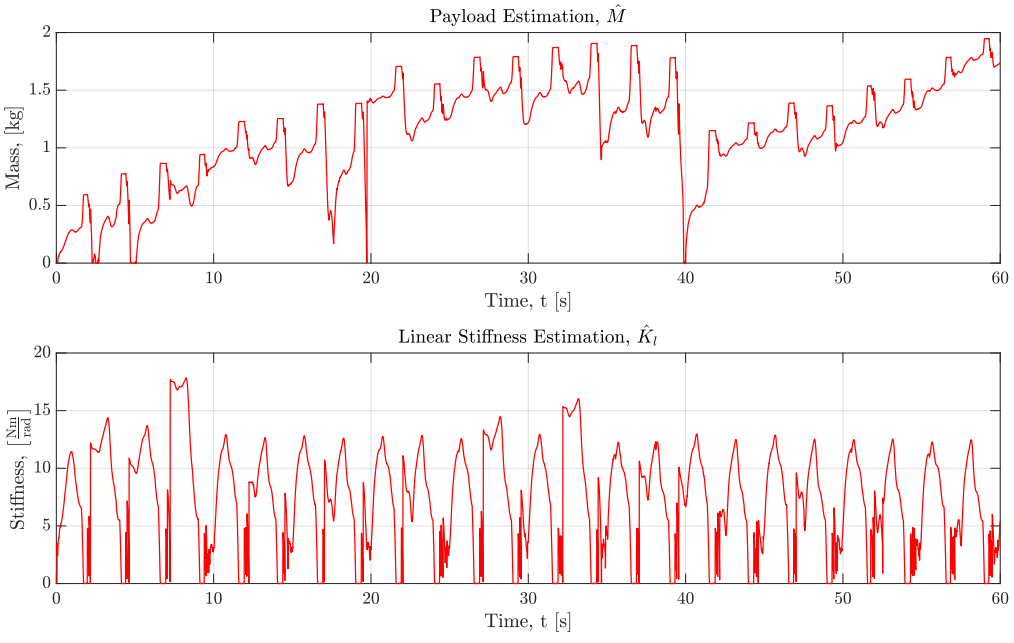


Figure 6.8: EKF's parameter estimation. Test with payload of $M = 1.1$ kg

Figure 6.8 shows that the parameter estimations are not perfect. The biggest problem is the payload estimation due to the variance. This variation will produce a variation in the gains for the gain scheduling control. This makes the controller unstable. It is

postulated that the nonlinear control algorithms can be a feasible solution in a more powerful microcontroller, where the sampling time can be sufficiently decreased is used. The test also shows that the estimations are not completely off. The EKF can therefore be a good solution if better - more stable - estimations can be obtained. However, after rigorous testing, better estimation could not be obtained.

Test Results 7

In this chapter, all the tests will be presented. Tests for both the low-level and high-level controllers have been conducted. For the low-level controller, tests with and without payload were made for a human-generated input. The high-level control tests include tests where the exoskeleton should compensate for itself regardless of its mass, an assistive test, and transparency mode - also called low impedance.

In Chapter 2, the requirements to the controllers were presented. These are repeated here in Tables 7.1 and 7.2.

Requirement	**Value**
Average error	0.05 rad
Average % error	5 %

Table 7.1: Low-level controller requirements

Test	**Requirement**	**Value**
Robot compensation	Average τ_{act} error between tests	0.25 Nm
Low impedance	Average τ_{vsm} error	0.3 Nm
Assistive	Assist level error	10%

Table 7.2: High-level controller requirements

7.1 Payload Estimation

As mentioned in Section 4.3 the payload can be estimated by using the VSM torque directly. This has been tested in Chapter 5, where simulations proved that this approximation is a feasible solution. However, to study if the approximation also can be used in a real-world scenario it will be analyzed in the tests. To make this analysis, two tests have been made; a test with no payload, and a test with payload. For the test with no payload, the interaction force sensor and the cuff are attached to the exoskeleton arm. The payloads are given by there gravitational effect, and for the two test these can be calculated to:

$$\begin{aligned} \tau_{pay,0} &= (m_{act} + m_{cuff})\ g\ l_p = 0.35\text{Nm} \\ \tau_{pay,1.1} &= (m_{act} + m_{cuff} + M)\ g\ l_p = 2.40\text{Nm} \end{aligned} \tag{7.1}$$

The results from the tests can be seen in Figure 7.1.

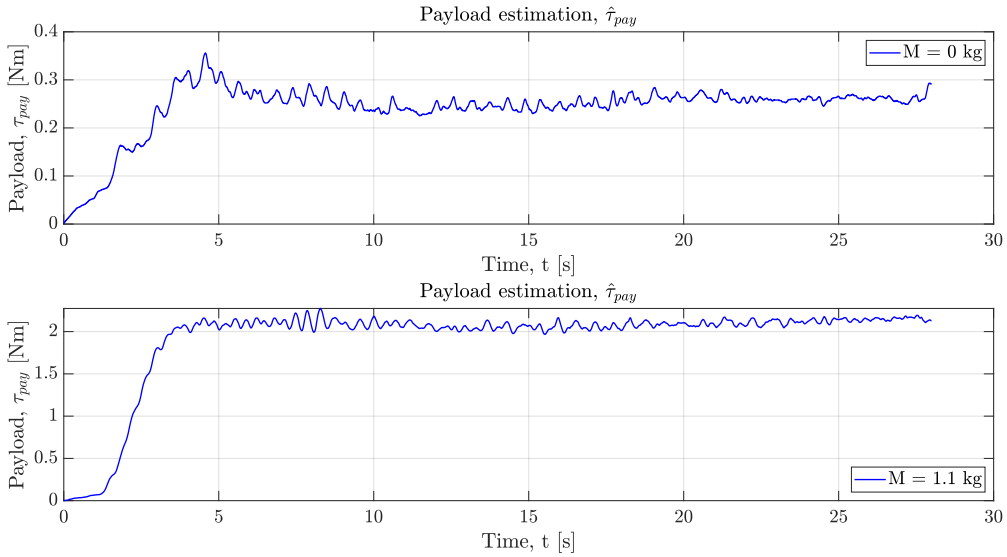


Figure 7.1: Payload estimation using the torque produced by the variable stiffness mechanism

From the figure, it can be seen that the estimation slightly underestimates the actual payload. This is, however, deemed sufficient since a lower payload estimation means that the control will be less aggressive. This means that the control will not be as good, but it is safer than overestimating the payload. Since the estimation has been deemed sufficient, it will be used in the rest of the tests.

7.2 Low-Level Position Control Tests Results

To test the GS LQR low-level position controller, two tests have been conducted. In the first test, no payload was added to the system, and in the second test, a payload was added. This payload was a 1.1 kg weight attached to the end of the arm. To get a more realistic evaluation of the controller, the input is generated by a human. To do this, an AAL-Band 2.0 from Biox is used. The band has an array of eight FSR sensors that measure the intensity of the muscle contraction (BioX, 2020). Using a neural network program in MATLAB, the input can be generated, recorded, and sent to the Arduino. The input and the results from the two tests can be seen in Figure 7.2.

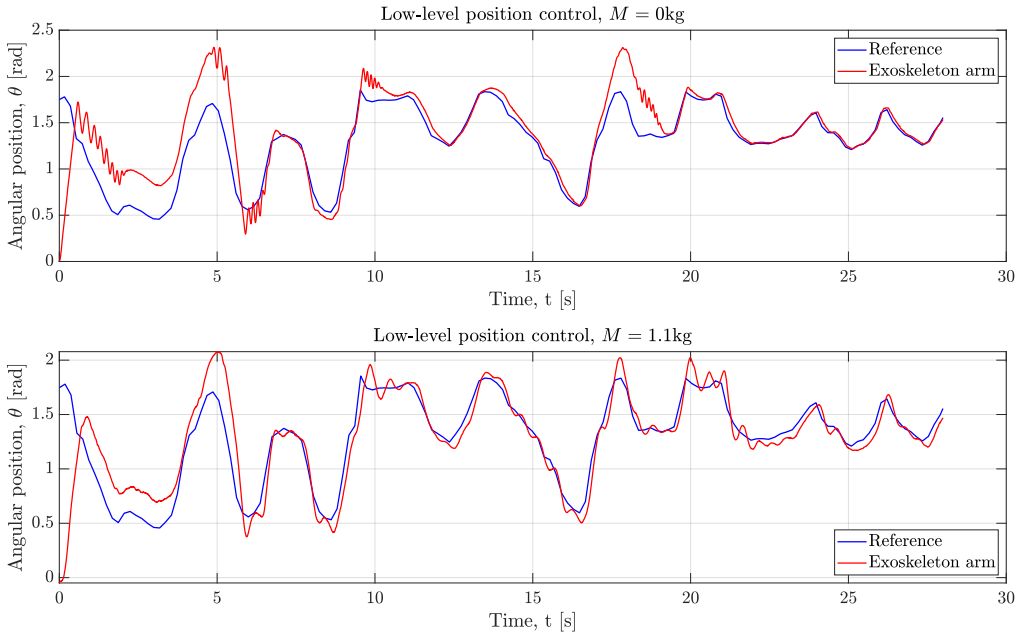


Figure 7.2: Test of the position controller for two load cases: $M = 0$ kg and $M = 1.1$ kg

From the figure it can be seen that the arm initially starts at $\theta_p = 0$ rad, where the reference starts at $\theta_{ref} = 1.75$ rad. This means that the controller gives a large input initially and takes around 6 seconds to eliminate the errors. Also seen is that when larger changes in the position are requested, the controller is too aggressive for $M = 0$ kg and is overshooting the reference, where also vibrations are induced. This is also the case for $M = 1.1$ kg. However, the overshoot is generally smaller. Overall, from the figures, the GS LQR seems to work as intended and can handle different load cases equally well.

Looking at the errors of the test - which can be seen in Figure 7.3 - it can be seen that the errors decrease ones the controller starts tracking the reference.

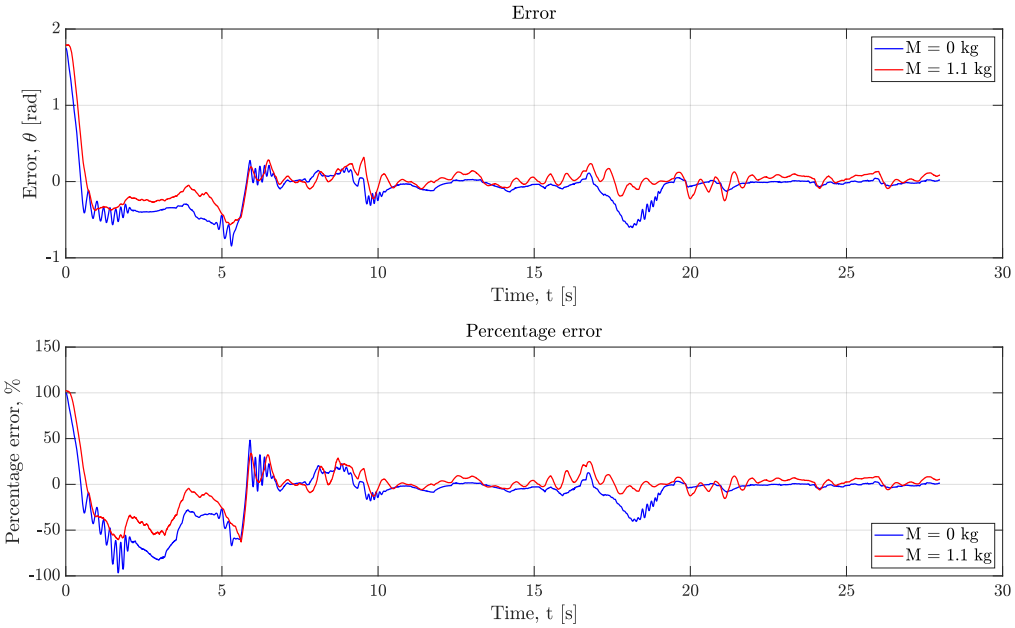


Figure 7.3: Errors for the position controller tests for two load cases: $M = 0$ kg and $M = 1.1$ kg

To obtain a more accurate representation of the controller performance, the initial response is excluded when calculating the average errors. The errors can be seen in Table 7.3.

Test	Error	Percentage error
M = 0 kg	0.078 rad	5.9 %
M = 1.1 kg	0.069 rad	5.7 %

Table 7.3: Average of the error and percentage error for two load cases: $M = 0$ kg and $M = 1.1$ kg

It can be seen that the controller is very close to meeting all of the requirements. Further tuning of the gains could likely give a better result.

7.3 High-Level Admittance Filter Test Results

To test the admittance filter, three experiments have been conducted. The first is a robot compensation test. The second is a low impedance test. Here the exoskeleton arm should follow the human without helping, meaning that the human produces all the force as if the exoskeleton was not attached to the user. The final test looks at the assistive capabilities of the exoskeleton.

7.3.1 Robot Compensation

As mentioned, this experiment investigates if the exoskeleton can compensate for itself and a payload attached to the exoskeleton. To conduct the test, the experiment was run twice, once without the payload to obtain a baseline and again with a payload. The payload used

is $\tau_{pay} = 2.40$ Nm. The user then did a series of moves. The results can be seen in Figure 7.4.

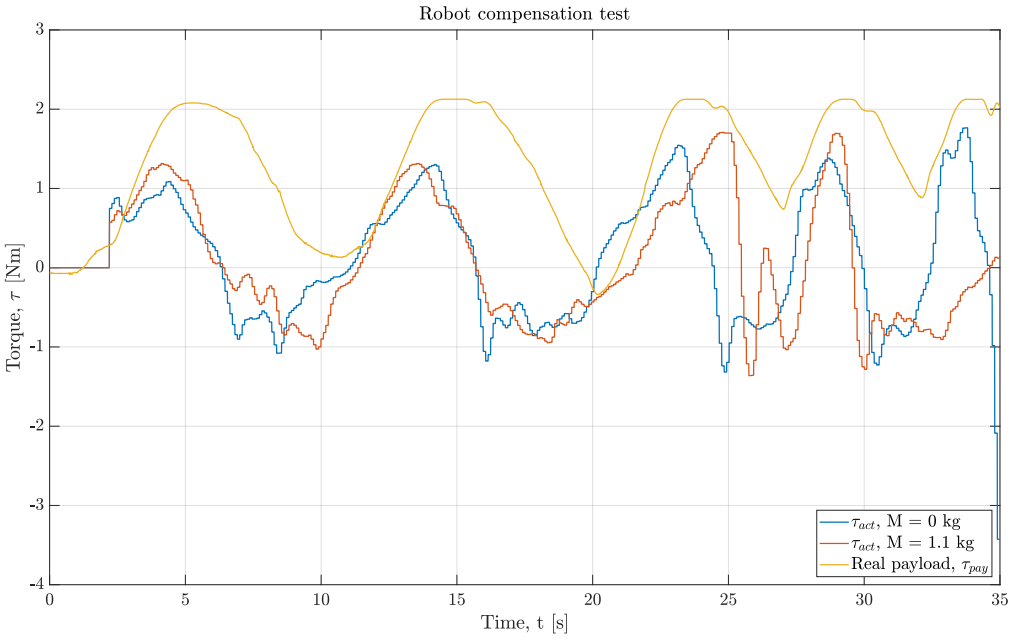


Figure 7.4: Interaction torque for $M = 0$ kg and $M = 1.1$ kg, and the actual payload for the $M = 1.1$ kg test

Two things can be concluded from the figure. Firstly, the system does compensate for the payload, as the interaction torque, τ_{act}, is lower than the actual payload. Secondly, the compensation works as intended, as the interaction torque, τ_{act}, is similar for both the test without a payload and with payload. The error between the two interaction torques can be seen in Figure 7.5.

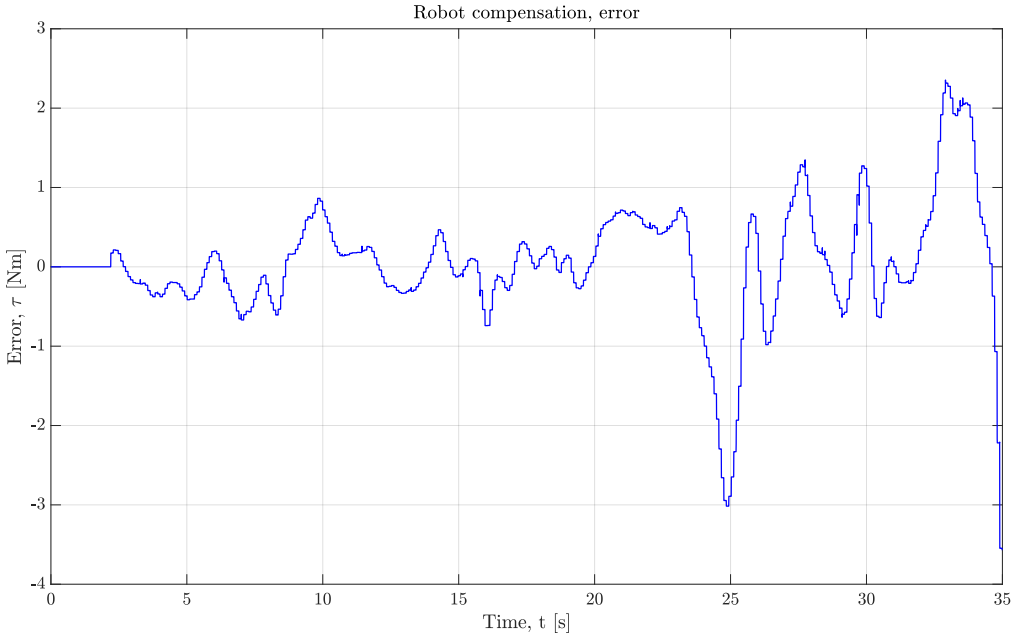


Figure 7.5: Error between the interaction torque of $M = 0$ kg and $M = 1.1$ kg

As the input is generated by a human it is somewhat difficult to compare all of it. Therefore, only the first part - up to about 20 seconds - will be included in the evaluation of the error between the two tests, which can be seen in Table 7.4.

Test	Error
Robot compensation	0.24 Nm

Table 7.4: Average of the error of the interaction torque between the test of $M = 0$ kg and $M = 1.1$ kg

It can be concluded that the robot compensation works as intended, and the average error is below the requirement.

7.3.2 Low Impedance

As mentioned, the low impedance experiment tests if the exoskeleton can follow the human without producing any force. The tests are conducted using the variable admittance filter, where the advantage - described in Section 5.1.2 - is set to 0. A payload of $\tau_{pay} = 2.40$ Nm is attached to the robot. For the low impedance to work correctly, the user should feel all of the 2.40 Nm produced by the payload, meaning that the interaction torque, τ_{act}, should be equal to the payload torque, τ_{pay}. The results can be seen in Figure 7.6.

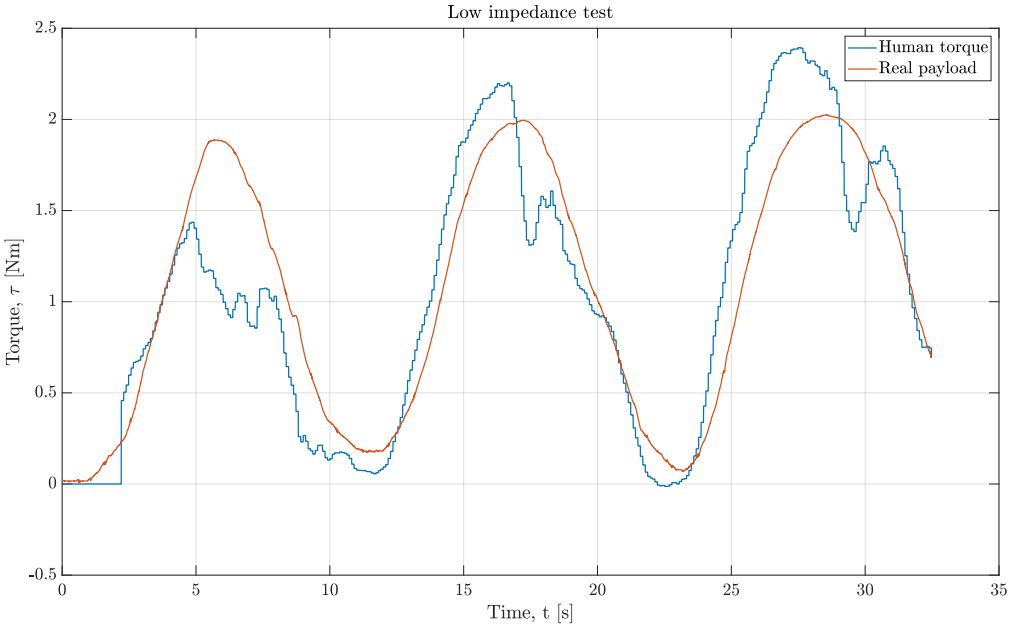


Figure 7.6: Low impedance test, where *human torque* $= \tau_{act}$

From the figure, it can be seen that the interaction is trying to follow the payload meaning that the VSM torque is going to zero. However, it does not seem good enough. Initially, it helps too much, and after the resistance keeps increasing. By analyzing the VSM torque, the low impedance capabilities of the system can be evaluated. The VSM torque is seen in Figure 7.7.

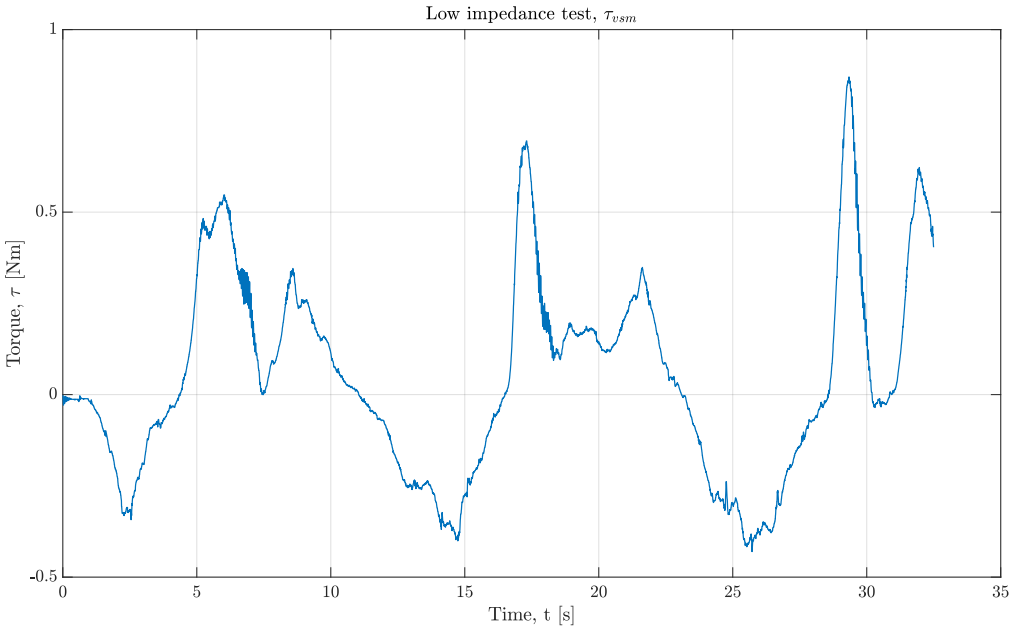


Figure 7.7: VSM torque from the low impedance test

From the figure, it can be seen that VSM torque does not go to zero. According to the requirements, the average error should be around 0.3 Nm and looking in Table 7.5 it can be seen that the average error is lower than the requirement. However, due to large peaks, the admittance filter needs more tuning to decrease the peaks sufficiently.

Test	**Error**
Robot compensation	0.21 Nm

Table 7.5: Average of the error of impedance test

7.3.3 Assistive

In this test, the exoskeleton has to reduce an assistive torque to help the user to lift a payload that the human is holding. The main difference from the robot compensation test is the placement of the payload. In the robot compensation test, the payload is placed *on* the robot - before the interaction force sensor - whereas in the assistive test, the payload is placed *in* the user's hand - after the interaction force sensor. This makes the payload much more challenging to detect. Islam and Bai (2019) developed a method to detect the payload in the user's arm by measuring the force that the biceps produce using an FSR sensor band similar to the one used to generate the human input for the low-level position control test in Section 7.2. However, due to time constraints, this methodology has not been implemented. Instead, the payload is assumed known and is hard-coded into the Arduino. The payload used was the pure weight of the arm of the user. The payload is then added to the measured interaction torque to give an altered interaction torque, which is fed to the control system, as explained in Section 5.2.

The evaluation will be of the percentage assistance the exoskeleton provides compared to the desired assistance. The desired assistance - or advantage - is given by k_0. For the test the advantage was set to be $k_0 = 0.73$. The measured interaction torque and the

torque produced by the exoskeleton can be seen in Figure 7.8 as well as the position of the exoskeleton and reference.

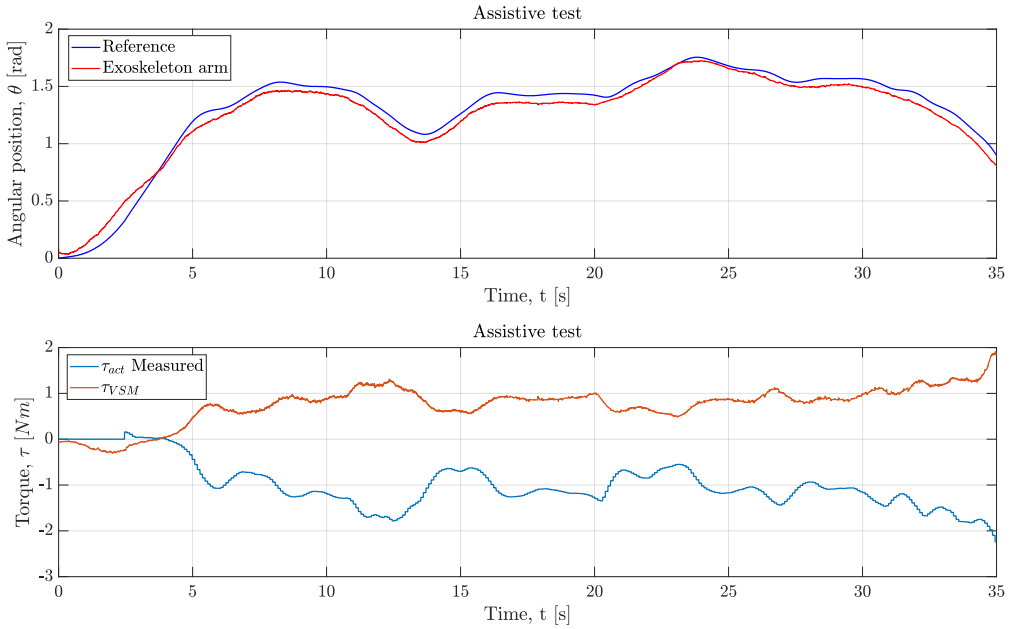


Figure 7.8: Reference, angular position, measured interaction torque, and VSM torque of the assistive test

If the VSM torque is equal to the absolute value of the interaction torque and is working in the same direction as the exoskeleton's movement, then the exoskeleton is helping. In Figure 7.8 it can be seen that the VSM torque does tend towards the interaction torque, in an absolute sense. The inverse behavior and working in the direction of movement means that it compensates for the load. The figure also shows that the interaction torque is slightly larger than the VSM torque, meaning the user still has to produce torque to lift the payload. To study how close the actual assistance is to the desired, the error between the actual and the desired is found. The error is given by

$$e_{k_0} = k_0 - \frac{\tau_{vsm}}{\tau_{pay}} \tag{7.2}$$

The average of this error is the found to be $e_{k_0} = 0.23\%$, which means that the exoskeleton has an effective desired advantage of $k_{0,\,eff} = 0.73\% - 0.23\% = 0.5\%$. This is an average percentage error of 27.4%. The biggest reason for this is the hard-coding of the payload of the human arm. In reality, this payload can vary greatly, which can make the assistance difficult. Estimating this payload could provide a better result.

7.4 Truly Sensorless Control

One of the main benefits of the VSM design is that it eliminates the need for a torque sensor to measure the actual torque affecting the user. This torque can be calculated from the τ_{vsm} formulation in Eq. (3.14). However, a force sensor is still needed to measure the interaction torque between the user and the exoskeleton. However, looking at the torques in the system, it becomes clear that the VSM torque necessarily has to be equal

to the interaction torque, as the interaction is defined as the torque that the exoskeleton is experiencing. Therefore, the interaction force sensor can be removed, and the calculated VSM torque - calculated from the encoder data - can be used as an interaction torque approximation. To test how good this assumption is, an experiment is conducted. Here the motor position is held constant meaning: $\tau_m = 0$. Therefore, the input flange - connected to the motor - will stay in place due to the worm gear. The exoskeleton arm position will be changed by applying a torque on the interaction cuff connected to the interaction force sensor. The deflection angle, θ_d, will change, and the VSM torque can be calculated and compared to the measured interaction torque. The results are seen in Figure 7.9.

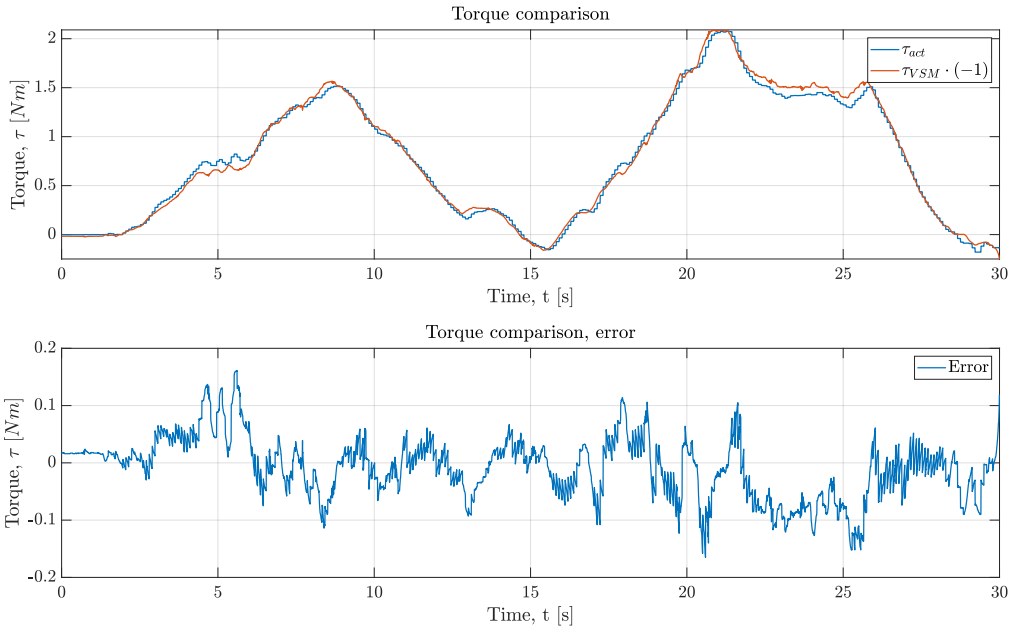


Figure 7.9: Interaction torque and VSM torque comparison and the error between them

From the figure, it can be seen that the VSM torque is a good approximation of the interaction torque. The average absolute error is lower than 0.05 Nm - see Table 7.6. This area should be investigated much more. For example, is how good the approximation is while running an actual test. Also, this approximation is based on the assumption that the payload is not attached to the exoskeleton, as this will drastically change the VSM torque. In other words, the approximation holds if torque produced by the exoskeleton arm's weight is very low - a lightweight arm. In general, it seems that the approximation can possibly be used, and the interaction sensor can be removed.

Test	**Error**
Torque comparison	0.04 Nm

Table 7.6: Average error between interaction torque and VSM torque

7.5 Test Result Conclusion

From the conducted tests, it can be seen that the low-level position controllers can track a human-generated position reference and comply with the requirements.

The two designed high-level control algorithms - for a payload on the exoskeleton and the payload on the human - can enforce the demanded dynamics. For the robot compensation test, it can be seen that the exoskeleton provides similar dynamics for two load cases. In the low impedance test, it is seen that the admittance filter still needs to be tuned to obtain a better response. For the assistive test, it can be seen that the demanded advantage is not obtained. An error of 27.4% exists. However, the exoskeleton does assist the user.

Lastly, the truly sensorless operation was studied and found that this is a potential for further decreasing the weight of the exoskeleton. However, this needs to be studied much more to reveal possible pitfalls.

Conclusion 8

As a conclusion to the problem analysis in Chapter 1, the following problem formulation was stated:

"How can human-robot interaction using an exoskeleton arm with a variable stiffness mechanism be controlled such that the effect of a payload is reduced?"

It can be concluded that the proposed control system can control the interaction between a human and an exoskeleton arm with a variable stiffness mechanism, such that the effect of a payload can be reduced.

The control system consists of a high-level admittance filter and a low-level gain scheduling LQR controller, which were designed in Chapter 5 and Chapter 4, respectively. These controllers were developed based on two models: an exoskeleton model and a human impedance model, both developed in Chapter 3. The exoskeleton model consists of a stiffness model and a dynamic model. The dynamic model consists of the dynamics of the motor and the dynamics of the variable stiffness mechanism. Both the stiffness and dynamic model were verified, and it can be concluded that they represent the system with minor discrepancies.

Both linear and nonlinear position controllers were designed. The linear LQRI controller was not good enough to track a position reference. However, the nonlinear gain scheduling LQR controller was good enough and met the requirements formulated in Chapter 2. The nonlinear controller depends on parameter estimations, which were developed by utilizing the system model and characteristics.

Two high-level controllers were designed: one for a payload on the robot and one for a payload on the human. During the tests in Chapter 7, both controllers could complete their tasks, which is to demand specific dynamics from the system. The high-level control for the payload on the robot should demand the same dynamics regardless of the payload. And the high-level control for the payload on the human should provide a given advantage. Both controllers completed these tasks successfully, which is a proof of concept about the usability of these control strategies for compliant exoskeletons. However, improvements to both can be made.

9 Discussion

In this chapter, some of the problems and findings will be discussed.

Modeling

One of the main reasons for the discrepancies between the model and the verification tests is friction. The VSA uses a worm gear to transfer the motor torque to the compliant elements and the user. A characteristic of the worm gear is that it is self-locking. This means that any forces transferred back through the system - say forces from the user - will be transformed into pure friction in the worm gear. Modeling friction is usually very complicated, although attempted using Coulomb friction. In reality, the friction could be much more complex, and determining values for the friction is not straightforward. Lastly, the friction characteristics seem to be highly dependent on the payload, as concluded in the model verification, where a good fit could only be obtained for one payload case at a time.

Even though the stiffness and torque models fit with real tests, there is still some discrepancy. This most like due to the worm screw's horizontal movement, as it has not been fastened to the frame. Instead, two small springs (or steel thread) have been used as spacers, adding some unknown compliance to the system. This also shows up when comparing the model and the real system, as the model predicts slightly higher stiffness.

The human model is derived based on the assumption that the human arm behaves like a rigid body. In reality, this is not the case. The human arm dynamics are highly nonlinear as the human can change the parameters of the arm independently. This can have a large effect on the system's stability, which is dependent on the human. However, since stability was analyzed for a wide range of parameters, this will likely not be a problem. A more advanced model could be derived, e.g., using a Hill model, if necessary.

Low-level control

The nonlinear gain scheduling LQR control is based on the new model-based parameter estimations. However, these parameter estimations are approximations and dependent on assumptions, such as steady-state and that the model accurately represents the system. If these estimations are off due to any disturbance in the system, such as the human interaction, the calculated gains will be wrong for the current load conditions, resulting in worse controller performance. In Section 7.1 the payload estimation was tested to be slightly too low. This is good for the stability of the system but does leave some performance on the table. A better estimation algorithm should be used, i.e., an extended Kalman filter, which is a well-documented tool and can take the system dynamics into

account. Using an EKF could result in better performance in both the simulations and in the tests. Also, as discussed in Section 4.3, the performance can be increased by redesigning the gains for new penalty values.

High-level control

The whole control algorithm has been divided into two separate controllers, which are afterward merged. These are the high-level and the low-level control. This introduces some limitations to the whole control algorithm and makes it harder to obtain an optimal response. Also, the stability analysis has been developed based on linear control theory, in which the stability has been computed for different cases. This does not ensure stability in the nonlinear controller, and tests have not been done for this system to prove the method. The different high-level control algorithms are highly based on payload estimation, which has not been considered here. This estimation significantly affects both the performance of the HRI exoskeleton and its stability. Lastly, a 'sensorless' control strategy has been proposed and simulated. Nonetheless, a study on the real system to determine its real potential is missing.

Implementation

A recurring problem from Pedersen and Murcia i Matute (2020) is the lower PWM input threshold of the ESCON 50/5 servo controller. The lowest possible input is 1.59%, which means that if minor corrections need to be made, the motor will not move, and an error occur, which turns it off. To mitigate this, 10% was added to the control input, which was then subtracted in the ESCON motor control setup by defining zero to be at 10% duty cycle. However, this does mean that the maximum possible input is limited to 90% of the theoretical maximum. This is a good trade-off since fine movements at small errors are more important. However, this limitation should be kept in mind as it also limits the controller for higher payloads, and thereby the overall performance of the controller.

In Section 6.4 an EKF was attempted implemented in the Arduino DUE. The implementation was deemed not possible as the filter's estimation did not work with the nonlinear gain scheduling controller used at the test. This could have been due to too large controller gains. Decreasing these could result in a usable response. However, this was not tested.

10 Further Work

In this chapter, related and additional tasks, which can be studied further, will be presented and discussed.

Low-level control - tuning and EKF

As seen from both the simulations and the real system tests, the position control is not perfect. The primary source of error comes from a slight phase shift from the reference to the actual position. A couple of solutions could be studied. First, the gains of the gain scheduling LQR control can be increased, which can better the response. Secondly, the control is dependent on estimations of parameters. If these estimations are off, the controller gains are incorrect for the specific load case. A better solution would be to implement an extended Kalman filter. However, as shown in Section 6.4, the implementation was not successful. It should, therefore, be studied if the EKF somehow can be implemented.

High-level control - parameter tuning

As for the low-level control, the parameters have not been tuned. This could be a focal point in the future; to obtain optimal parameter values for a given load and use case. Also, both the low-level control and the high-level control can be studies as one system. The different gains and parameters of the whole control algorithm can be optimized simultaneously for better performance and stability of the exoskeleton.

Sensorless control

In Section 7.4 an estimation of the interaction torque was presented. This estimation should be studied further since a sufficient estimation can remove the need for an expensive and heavy force sensor. The primary study area is, studying in which conditions the estimation holds. It remains to be examined if the estimation holds while running a test where a human uses the exoskeleton. Also, if the exoskeleton arm is too heavy or a payload is attached to it, deflection in the VSM will exist, meaning a non-zero VSM torque, while the interaction torque is zero, resulting in a wrong estimation.

Payload estimation for payloads on the human

As explained in Section 5.2, the approach to provide an assistive torque while the user is holding a payload presented in Islam and Bai (2019) is adapted. The payload estimation is not implemented but hard-coded instead. If the estimation algorithm is implemented, the payload estimation will depend on the assistive torque from the exoskeleton. Assisting

the user will mean the user is generating less force - tensing the muscle less - which would decrease the estimated payload, resulting in a potential wrong estimation. Therefore, it should be analyzed if the VSM's assistive torque can compensate the estimation by adding it to the estimated payload to regain the correct estimation. This could give more freedom when designing the desired advantage, increasing the potency of the exoskeleton.

Intention detection

In Chapter 1, it was described that the performance of the exoskeleton could be improved by designing an intentional reaching direction algorithm (IRD) or motion intention detection algorithm (MID). These algorithms have not been considered in this project. However, they could be studied, and their effect on a variable stiffness exoskeleton analyzed.

Bibliography

BioX, **2020**. BioX. *AAL-Band Data Sheet*. `https://www.bioxgroup.dk/wp-content/uploads/2020/10/BioX-AAL-Band-Data-Sheet.pdf`, 2020. Online; accessed 29 May 2021.

Buerger and Hogan, **2006**. Stephen P. Buerger and Neville Hogan. *Relaxing passivity for human-robot interaction*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006.

Buerger and Hogan, **2007**. Stephen P. Buerger and Neville Hogan. *Complementary Stability and Loop Shaping for Improved Human-Robot Interaction*. IEEE Transactions on Robotics, 2007.

CHEN, **2005**. WAI-KAI CHEN. *The Electrical Engineering Handbook*. Academic Press, 2005.

Cloud, **1965**. Wallace Cloud. *Man Amplifiers: Machines that Let You Carry a Ton*. Popular Science, 1965.

Colgate, **1988**. James Edward Colgate. *The Control of Dynamically Interacting Systems*. Massachusetts Institute of Technology, 1988.

Ekso Bionics, **2021**. Ekso Bionics. *Ekso EVO*. `https://eksobionics.com/ekso-evo/`, 2021. Online; accessed 15 February 2021.

Gopura and Kiguchi, **2009**. R. A. R. C. Gopura and Kazuo Kiguchi. *Mechanical Designs of Active Upper-Limb Exoskeleton Robots: State-of-the-Art and Design Difficulties*. IEEE 11th International Conference on Rehabilitation Robotics, 2009.

Gopura et al., **2009**. R. A. R. C. Gopura, Kazuo Kiguchi and Yang Li. *SUEFUL-7: A 7DOF Upper-Limb Exoskeleton Robot with Muscle-Model-Oriented EMG-Based Control*. The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

Gorgey, **2018**. Ashraf S Gorgey. *Robotic exoskeletons: The current pros and cons*, US National Library of Medicine; National Institutes of Health, 2018.

Gupta and O'Malley, **2006**. Abhishek Gupta and Marcia K. O'Malley. *Design of a Haptic Arm Exoskeleton for Training and Rehabilitation*. IEEE/ASME TRANSACTIONS ON MECHATRONICS, 2006.

Huo et al., **2011**. Weiguang Huo, Jian Huang, Yongji Wang, Jun Wu and Lei Cheng. *Control of Upper-Limb Power-Assist Exoskeleton Based on Motion Intention Recognition*. IEEE International Conference on Robotics and Automation, 2011.

Islam and Bai, **2019**. Muhammad R. U. Islam and Shaoping Bai. *Payload estimation using forcemyography sensors for control of upper-body exoskeleton in load carrying assistance*. Modeling, Identification and Control, Vol. 40, 2019.

Keemink et al., **2018**. Arvid QL Keemink, Herman van der Kooij and Arno HA Stienen. *Admittance control for physical human–robot interaction*. The International Journal of Robotic Research, 2018.

Kong and Tomizuka, **2009**. Kyoungchul Kong and Masayoshi Tomizuka. *Control of Exoskeletons Inspired by Fictitious Gain in Human Model*. IEEE/ASME TRANSACTIONS ON MECHATRONICS, 2009.

Krishnan et al., **2016**. R. Hari Krishnan, V. Devanandh, Aditya Kiran Brahma and S. Pugazhenthi. *Estimation of mass moment of inertia of a human body, when bending forward, for the design of a self-transfer robotic facility*. Journal of Engineering Science and Technology, 2016.

Letier et al., **2010**. Pierre Letier, Elvina Motard and Jean-Philippe Verschueren. *EXOSTATION : Haptic Exoskeleton Based Control Station*. 2010 IEEE International Conference on Robotics and Automation, 2010.

Li et al., **2017**. Zhijun Li, Zhicong Huang, Wei He and Chun-Yi Su. *Adaptive Impedance Control for an Upper Limb Robotic Exoskeleton Using Biological Signals*. IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, 2017.

Li and Bai, **2019**. Zhongyi Li and Shaoping Bai. *A novel revolute joint of variable stiffness with reconfigurability*. Mechanism and Machine Theory, Volume 133, Pages 720-736, 2019.

Li et al., **2020**. Zhongyi Li, Shaoping Bai, Ole Madsen, Weihai Chen and Jianbin Zhang. *Design, modeling and testing of a compact variable stiffness mechanism for exoskeletons*. Mechanism and Machine Theory, Volume 151, 2020.

Mayo Clinic, **2019**. Mayo Clinic. *Electromyography (EMG)*. `https://www.mayoclinic.org/tests-procedures/emg/about/pac-20393913`, 2019. Online; accessed 15 February 2021.

Medicine, **2021**. Johns Hopkins Medicine. *Stroke Recovery Timeline*. `https://www.hopkinsmedicine.org/health/conditions-and-diseases/stroke/stroke-recovery-timeline`, 2021. Online; accessed 25 February 2021.

Pedersen and Murcia i Matute, **2020**. Andreas Pedersen and Marc Murcia i Matute. *Control of a Novel Variable Stiffness Actuator*, Aalborg University; Faculty of Engineering and Science, 2020.

Pons, **2008**. Jose L. Pons. *Wearable Robots: Biomechatronic Exoskeletons*. John Wiley & Sons, Ltd, 2008.

Winter, **2009**. David A. Winter. *Biomechanics and Motor Control of Human Movement*. John Wiley & Sons, Ltd. Fourth Edition, 2009.

Zahedi et al., 2020. Ahmad Zahedi, Bin Zhang, Andong Yi and Dingguo Zhang. *A Soft Exoskeleton for Tremor Suppression Equipped with Flexible Semiactive Actuator*. Soft Robotics, 2020.

Jacobian and Jacobian Derivative A

The Jacobian and its derivative used in the VSM mathematical model are provided in this appendix:

$$J = X1X3 + X2X4 \tag{A.1}$$

where

$$\begin{aligned} X1 &= \frac{1}{\sqrt{\left(\frac{L_1}{2} - l_1 \sin\theta_d\right)^2 + \left(l_3 - l_1 \cos\theta_d\right)^2}} \\ X2 &= \frac{1}{\sqrt{\left(\frac{L_1}{2} + l_1 \sin\theta_d\right)^2 + \left(l_3 - l_1 \cos\theta_d\right)^2}} \\ X3 &= \left(l_1 l_3 \sin\theta_d - \frac{L_1 l_1 \cos\theta_d}{2}\right) \\ X4 &= \left(l_1 l_3 \sin\theta_d + \frac{L_1 l_1 \cos\theta_d}{2}\right) \end{aligned} \tag{A.2}$$

And its derivative is

$$\dot{J} = \dot{X}1X3 + X1\dot{X}3 + \dot{X}2X4 + X2\dot{X}4 \tag{A.3}$$

where

$$\begin{aligned} \dot{X}1 &= -\frac{l_1\left(L_1 \cos\theta_d - 2l_3 \sin\theta_d\right)}{2X1} \\ \dot{X}2 &= \frac{l_1\left(L_1 \cos\theta_d + 2l_3 \sin\theta_d\right)}{2X2} \\ \dot{X}3 &= \left(l_1 l_3 \cos\theta_d + \frac{L_1 l_1 \sin\theta_d}{2}\right) \\ \dot{X}4 &= \left(l_1 l_3 \cos\theta_d - \frac{L_1 l_1 \sin\theta_d}{2}\right) \end{aligned} \tag{A.4}$$

Gain Scheduling Polynomials B

Here the gain scheduling polynomials are presented. Where $x_1 = \hat{\tau}_{pay}$ and $y_1 = \hat{K}_{vsm}$.

Pendulum Speed Gain

Gain parameters array:

$$\begin{aligned} g_1[18] = \{&3.00955479863,\ 2.26948522723,\ -0.6046405500,\\ &-0.44608102976,\ -0.0378481374048,\ 0.050393557205,\\ &0.042720383761, 0.008379105155,\ -0.000551700986,\\ &-0.001505553850,\ -0.0019559224802, -0.00042004441978,\\ &-0.0001927435432,\ 5.6873468403e-05,\ 3.404150708e-05,\\ &7.768978178e-06,\ 3.925977765e-06,\ 2.1539784905e-06\} \end{aligned} \tag{B.1}$$

Gain calculation polynomial

$$\begin{aligned} K_1 = {} & g_1[0] + g_1[1]x_1 + g_1[2]y_1 + g_1[3]x_1^2 + g_1[4]x_1y_1 + g_1[5]y_1^2 + g_1[6]x_1^3 \\ & + g_1[7]x_1^2y_1 + g_1[8]x_1y_1^2 + g_1[9]y_1^3 + g_1[10]x_1^4 + g_1[11]x_1^3y_1 \\ & + g_1[12]x_1^2y_1^2 + g_1[13]x_1y_1^3 + g_1[14]x_1^5 + g_1[15]x_1^4y_1 + g_1[16]x_1^3y_1^2 \\ & + g_1[17]x_1^2y_1^3 \end{aligned} \tag{B.2}$$

Pendulum Position Gain:

Gain parameters array:

$$\begin{aligned} g_2[21] = \{&0.866660933975961,\ 16.6549030037834,\ -2.28615642443827,\\ &-6.26309227484011,\ 3.90072949202062,\ -1.14512070267785,\\ &0.758196752918755,\ -0.287306392844755,\ -0.296035813759342,\\ &0.209449922775070,\ -0.0393598394194426,\ 0.0110744916899031,\\ &0.0114616435068732,\ 0.0119396616266815,\ -0.0137439691664128,\\ &0.000741491312478,\ -0.000162036726504,\ -0.000199182000466,\\ &-0.000165152659197,\ -0.000207185193518,\ 0.000320148346465\} \end{aligned} \tag{B.3}$$

Gain calculation polynomial

$$\begin{aligned} K_2 = {} & g_2[0] + g_2[1]x_1 + g_2[2]y_1 + g_2[3]x_1^2 + g_2[4]x_1y_1 + g_2[5]y_1^2 + g_2[6]x_1^3 \\ & + g_2[7]x_1^2y_1 + g_2[8]x_1y_1^2 + g_2[9]y_1^3 + g_2[10]x_1^4 + g_2[11]x_1^3y_1 \\ & + g_2[12]x_1^2y_1^2 + g_2[13]x_1y_1^3 + +g_2[14]y_1^4 + g_2[15]x_1^5 + g_2[16]x_1^4y_1 \\ & + g_2[17]x_1^3y_1^2 + g_2[18]x_1^2y_1^3 + g_2[19]x_1y_1^4 + g_2[20]y_1^5; \end{aligned} \tag{B.4}$$

Motor Speed Gain:

Gain parameters array:

$$\begin{aligned} g_3[21] = \{&0.003269005751,\ -0.00149016998,\ 0.000621987586,\\ &0.00034080966757,\ -5.35407771882e-05,\ -8.5878385351e-05,\\ &-3.5938548104e-05,\ 3.01762836605e-06,\ 4.2690095203e-06,\\ &7.1016788301e-06,\ 1.7377035242e-06,\ -5.48962421e-08,\\ &-1.92130778515e-07,\ -1.51357275267e-07,\ -3.191406921e-07,\\ &-3.12953954467e-08,\ -3.8600114658e-10,\ 3.392442951e-09,\\ &2.70469618135e-09,\ 2.296794150e-09,\ 5.89433273531e-09\} \end{aligned} \tag{B.5}$$

Gain calculation polynomial

$$\begin{aligned} K_3 = {} & g_3[0] + g_3[1]x_1 + g_3[2]y_1 + g_3[3]x_1^2 + g_3[4]x_1y_1 + g_3[5]y_1^2 + g_3[6]x_1^3 \\ & + g_3[7]x_1^2y_1 + g_3[8]x_1y_1^2 + g_3[9]y_1^3 + g_3[10]x_1^4 + g_3[11]x_1^3y_1 \\ & + g_3[12]x_1^2y_1^2 + g_3[13]x_1y_1^3 + +g_3[14]y_1^4 + g_3[15]x_1^5 + g_3[16]x_1^4y_1 \\ & + g_3[17]x_1^3y_1^2 + g_3[18]x_1^2y_1^3 + g_3[19]x_1y_1^4 + g_3[20]y_1^5; \end{aligned} \tag{B.6}$$

Motor Position Gain:

Gain parameters array:

$$\begin{aligned} g_4[21] = \{&0.14706645545,\ -0.108697066299,\ 0.0422367250206,\\ &0.028603138492,\ -0.00781255170,\ -0.0045417378394,\\ &-0.00323156584,\ 0.0007039616055,\ 0.00047697548685,\\ &0.0003056414877,\ 0.00016304279230,\ -2.811529774e-05,\\ &-2.650301471523e-05,\ -1.4115251339e-05,\ -1.18512062955e-05,\\ &-3.023717266075-06,\ 4.00331077015e-07,\ 5.3178771826e-07,\\ &3.05969379498e-07,\ 1.8643189677e-07,\ 1.96492621202e-07\} \end{aligned} \tag{B.7}$$

Gain calculation polynomial

$$\begin{aligned} K_4 = {} & g_4[0] + g_4[1]x_1 + g_4[2]y_1 + g_4[3]x_1^2 + g_4[4]x_1y_1 + g_4[5]y_1^2 + g_4[6]x_1^3 \\ & + g_4[7]x_1^2y_1 + g_4[8]x_1y_1^2 + g_4[9]y_1^3 + g_4[10]x_1^4 + g_4[11]x_1^3y_1 \\ & + g_4[12]x_1^2y_1^2 + g_4[13]x_1y_1^3 + +g_4[14]y_1^4 + g_4[15]x_1^5 + g_4[16]x_1^4y_1 \\ & + g_4[17]x_1^3y_1^2 + g_4[18]x_1^2y_1^3 + g_4[19]x_1y_1^4 + g_4[20]y_1^5; \end{aligned} \tag{B.8}$$

Code C

```
1
//================================================================================
// INCLUDES
//================================================================================

#include <Encoder.h>
#include <math.h>
#include "constants_and_globals.h"
#include <FileIO.h>
#include <DueTimer.h>
#include <BasicLinearAlgebra.h>
#include <Ewma.h>
#include "HX711.h"



//================================================================================
// DEFINITIONS
//================================================================================

#define SIN_ELEMENTS 359                //Definition of sinus table for pwm, 360 degrees
using namespace BLA;

//================================================================================
//  CONSTANTS
//================================================================================

///////// Filter //////////////
Ewma adcFilter1(0.01);
Ewma pendulumfilter(0.1);
Ewma qwffilter(0.01);
////////////////////// CONTROLLER INITS//////////////////////////
// gains for LQRI controller
float K_lqri[5] = {0.0046, 0.0988, 0.0002, 0.0085+0.02*3, -60.6228/2.5};

unsigned int increment = 0;
int tsample =5000;          // mu s
float tsamples = (float)tsample * 0.000001;        // s
float invers_tsamples = 1 / tsamples; // inverse of tsamples frequency
const float fs = 0.5;
float Errorint = 0;

float dtheta_p = 0;
float theta_p = 0;
float dtheta_m = 0;
float theta_m = 0;
float T_vsm = 0;
float K_l = 0;
float payload = 0;
float M = 0;

// Admittance filter
float Ma = 0.06;
float Ba = 3;
float Ga = 2;
float advantage = 2;

// feed forward term
float g = 9.82;
float l_p_t = 0.21;
float l_p = 0.2165;
int MM = 0;
float mm_p = 0.2;
```

```
float eta_g = 0.65;
float n = 270;
float beta = 1 / (n*n) * eta_g;
float k_l = 35;
float kgg = MM * g * l_p_t + mm_p * g * l_p;
float k_t = 0.0281;
float k22 = K_lqri[1];
float k44 = K_lqri[3];
float FF = (beta*kgg*k_l * n + k44*kgg*k_t * n + k44*k_l*k_t * n + k22*k_l*k_t) / (k_l*k_t);

// Sensor inits
int intrcr = 0;
float sendat[7] = {};
float sendate[7] = {};
long Motorencoder;
float deg2rad = 0.01745329252;
float dd[8] = {0, 0, 0, 0, 0, 0, 0, 0};
float motorvelfilter;
float velfilter[8] = {0, 0, 0, 0, 0, 0, 0, 0};
float motordd[2] = {0, 0};
float errorpenpos[2] = {0, 0};
float curarray[3] = {0, 0, 0};
float refarray[3] = {0, 0, 0};
float u, motorpos, penpos, motorvel, penvel, penref;
unsigned int duty_update = 0;
float I = 0;
float Stiffness;
float force;
float Calibration_factor = 20650;
HX711 Interaction_sensor;
float dtheta_ref=0;
float theta_ref = 0;
float interaction_force=0;
float pendulum;
float t_int = 0.001;
float T_pay = 0;
////////////////////////////////////////////////


const float ANGLE[SIN_ELEMENTS]={0.000000,0.017453,0.034907,0.052360,0.069813,0.087266,
    0.104720,0.122173,0.139626,0.157080,0.174533,0.191986,0.209440,0.226893,0.244346,0.261799,
    0.279253,0.296706,0.314159,0.331613,0.349066,0.366519,0.383972,0.401426,0.418879,0.436332,
    0.453786,0.471239,0.488692,0.506145,0.523599,0.541052,0.558505,0.575959,0.593412,0.610865,
    0.628319,0.645772,0.663225,0.680678,0.698132,0.715585,0.733038,0.750492,0.767945,0.785398,
    0.802851,0.820305,0.837758,0.855211,0.872665,0.890118,0.907571,0.925025,0.942478,0.959931,
    0.977384,0.994838,1.012291,1.029744,1.047198,1.064651,1.082104,1.099557,1.117011,1.134464,
    1.151917,1.169371,1.186824,1.204277,1.221730,1.239184,1.256637,1.274090,1.291544,1.308997,
    1.326450,1.343904,1.361357,1.378810,1.396263,1.413717,1.431170,1.448623,1.466077,1.483530,
    1.500983,1.518436,1.535890,1.553343,1.570796,1.588250,1.605703,1.623156,1.640610,1.658063,
    1.675516,1.692969,1.710423,1.727876,1.745329,1.762783,1.780236,1.797689,1.815142,1.832596,
    1.850049,1.867502,1.884956,1.902409,1.919862,1.937315,1.954769,1.972222,1.989675,2.007129,
    2.024582,2.042035,2.059489,2.076942,2.094395,2.111848,2.129302,2.146755,2.164208,2.181662,
    2.199115,2.216568,2.234021,2.251475,2.268928,2.286381,2.303835,2.321288,2.338741,2.356194,
    2.373648,2.391101,2.408554,2.426008,2.443461,2.460914,2.478368,2.495821,2.513274,2.530727,
    2.548181,2.565634,2.583087,2.600541,2.617994,2.635447,2.652900,2.670354,2.687807,2.705260,
    2.722714,2.740167,2.757620,2.775074,2.792527,2.809980,2.827433,2.844887,2.862340,2.879793,
    2.897247,2.914700,2.932153,2.949606,2.967060,2.984513,3.001966,3.019420,3.036873,3.054326,
    3.071779,3.089233,3.106686,3.124139,3.141593,3.159046,3.176499,3.193953,3.211406,3.228859,
    3.246312,3.263766,3.281219,3.298672,3.316126,3.333579,3.351032,3.368485,3.385939,3.403392,
    3.420845,3.438299,3.455752,3.473205,3.490659,3.508112,3.525565,3.543018,3.560472,3.577925,
    3.595378,3.612832,3.630285,3.647738,3.665191,3.682645,3.700098,3.717551,3.735005,3.752458,
    3.769911,3.787364,3.804818,3.822271,3.839724,3.857178,3.874631,3.892084,3.909538,3.926991,
    3.944444,3.961897,3.979351,3.996804,4.014257,4.031711,4.049164,4.066617,4.084070,4.101524,
    4.118977,4.136430,4.153883,4.171337,4.188790,4.206244,4.223697,4.241150,4.258604,4.276057,
    4.293510,4.310963,4.328416,4.345870,4.363323,4.380776,4.398230,4.415683,4.433136,4.450590,
    4.468043,4.485496,4.502949,4.520403,4.537856,4.555309,4.572762,4.590216,4.607669,4.625123,
    4.642576,4.660029,4.677482,4.694936,4.712389,4.729842,4.747295,4.764749,4.782202,4.799655,
    4.817109,4.834562,4.852015,4.869469,4.886922,4.904375,4.921828,4.939282,4.956735,4.974188,
    4.991642,5.009095,5.026548,5.044002,5.061455,5.078908,5.096361,5.113815,5.131268,5.148721,
    5.166174,5.183628,5.201081,5.218534,5.235988,5.253441,5.270895,5.288348,5.305801,5.323254,
    5.340707,5.358161,5.375614,5.393067,5.410521,5.427974,5.445427,5.462881,5.480334,5.497787,
    5.515240,5.532694,5.550147,5.567600,5.585053,5.602507,5.619960,5.637414,5.654867,5.672320,
    5.689773,5.707227,5.724680,5.742133,5.759586,5.777040,5.794493,5.811946,5.829400,5.846853,
    5.864306,5.881760,5.899213,5.916666,5.934119,5.951573,5.969026,5.986479,6.003932,6.021386,
    6.038839,6.056293,6.073746,6.091199,6.108652,6.126106,6.143559,6.161012,6.178465,6.195919,
    6.213372,6.230825,6.248279};

const float SINTABLE[SIN_ELEMENTS]={0.000000,0.017452,0.034899,0.052336,0.069756,0.087156,
```

```
    0.104528,0.121869,0.139173,0.156434,0.173648,0.190809,0.207912,0.224951,0.241922,0.258819,
    0.275637,0.292372,0.309017,0.325568,0.342020,0.358368,0.374607,0.390731,0.406737,0.422618,
    0.438371,0.453990,0.469472,0.484810,0.500000,0.515038,0.529919,0.544639,0.559193,0.573576,
    0.587785,0.601815,0.615662,0.629320,0.642788,0.656059,0.669131,0.681998,0.694658,0.707107,
    0.719340,0.731354,0.743145,0.754710,0.766044,0.777146,0.788011,0.798636,0.809017,0.819152,
    0.829038,0.838671,0.848048,0.857167,0.866025,0.874620,0.882948,0.891007,0.898794,0.906308,
    0.913545,0.920505,0.927184,0.933580,0.939693,0.945519,0.951057,0.956305,0.961262,0.965926,
    0.970296,0.974370,0.978148,0.981627,0.984808,0.987688,0.990268,0.992546,0.994522,0.996195,
    0.997564,0.998630,0.999391,0.999848,1.000000,0.999848,0.999391,0.998630,0.997564,0.996195,
    0.994522,0.992546,0.990268,0.987688,0.984808,0.981627,0.978148,0.974370,0.970296,0.965926,
    0.961262,0.956305,0.951057,0.945519,0.939693,0.933580,0.927184,0.920505,0.913545,0.906308,
    0.898794,0.891007,0.882948,0.874620,0.866025,0.857167,0.848048,0.838671,0.829038,0.819152,
    0.809017,0.798635,0.788011,0.777146,0.766044,0.754710,0.743145,0.731354,0.719340,0.707107,
    0.694658,0.681998,0.669131,0.656059,0.642788,0.629321,0.615661,0.601815,0.587785,0.573576,
    0.559193,0.544639,0.529919,0.515038,0.500000,0.484810,0.469472,0.453991,0.438371,0.422618,
    0.406737,0.390731,0.374607,0.358368,0.342020,0.325568,0.309017,0.292372,0.275637,0.258819,
    0.241922,0.224951,0.207912,0.190809,0.173648,0.156434,0.139173,0.121869,0.104528,0.087156,
    0.069756,0.052336,0.034899,0.017452,-0.000000,-0.017452,-0.034899,-0.052336,-0.069756,
    -0.087156,-0.104528,-0.121869,-0.139173,-0.156434,-0.173648,-0.190809,-0.207912,-0.224951,
    -0.241922,-0.258819,-0.275637,-0.292372,-0.309017,-0.325568,-0.342020,-0.358368,-0.374607,
    -0.390731,-0.406737,-0.422618,-0.438371,-0.453991,-0.469472,-0.484810,-0.500000,-0.515038,
    -0.529919,-0.544639,-0.559193,-0.573576,-0.587785,-0.601815,-0.615661,-0.629320,-0.642788,
    -0.656059,-0.669131,-0.681998,-0.694658,-0.707107,-0.719340,-0.731354,-0.743145,-0.754710,
    -0.766045,-0.777146,-0.788011,-0.798635,-0.809017,-0.819152,-0.829038,-0.838671,-0.848048,
    -0.857167,-0.866025,-0.874620,-0.882948,-0.891006,-0.898794,-0.906308,-0.913545,-0.920505,
    -0.927184,-0.933581,-0.939693,-0.945519,-0.951056,-0.956305,-0.961262,-0.965926,-0.970296,
    -0.974370,-0.978148,-0.981627,-0.984808,-0.987688,-0.990268,-0.992546,-0.994522,-0.996195,
    -0.997564,-0.998630,-0.999391,-0.999848,-1.000000,-0.999848,-0.999391,-0.998630,-0.997564,
    -0.996195,-0.994522,-0.992546,-0.990268,-0.987688,-0.984808,-0.981627,-0.978148,-0.974370,
    -0.970296,-0.965926,-0.961262,-0.956305,-0.951056,-0.945519,-0.939693,-0.933580,-0.927184,
    -0.920505,-0.913545,-0.906308,-0.898794,-0.891007,-0.882948,-0.874620,-0.866025,-0.857167,
    -0.848048,-0.838670,-0.829038,-0.819152,-0.809017,-0.798635,-0.788011,-0.777146,-0.766044,
    -0.754710,-0.743145,-0.731354,-0.719340,-0.707107,-0.694659,-0.681998,-0.669131,-0.656059,
    -0.642788,-0.629321,-0.615661,-0.601815,-0.587785,-0.573577,-0.559193,-0.544639,-0.529919,
    -0.515038,-0.500000,-0.484809,-0.469471,-0.453991,-0.438371,-0.422618,-0.406736,-0.390731,
    -0.374607,-0.358368,-0.342020,-0.325568,-0.309017,-0.292372,-0.275638,-0.258819,-0.241922,
    -0.224951,-0.207912,-0.190809,-0.173648,-0.156434,-0.139173,-0.121869,-0.104529,-0.087156,
    -0.069756,-0.052336,-0.034900};
const float COSTABLE[SIN_ELEMENTS] = {1.000000, 0.999848, 0.999391, 0.998630, 0.997564,
      0.996195, 0.994522, 0.992546, 0.990268, 0.987688, 0.984808, 0.981627, 0.978148,
      0.974370, 0.970296, 0.965926, 0.961262, 0.956305, 0.951057, 0.945519, 0.939693,
      0.933580, 0.927184, 0.920505, 0.913545, 0.906308, 0.898794, 0.891007, 0.882948,
      0.874620, 0.866025, 0.857167, 0.848048, 0.838671, 0.829038, 0.819152, 0.809017,
      0.798636, 0.788011, 0.777146, 0.766044, 0.754710, 0.743145, 0.731354, 0.719340,
      0.707107, 0.694658, 0.681998, 0.669131, 0.656059, 0.642788, 0.629320, 0.615662,
      0.601815, 0.587785, 0.573576, 0.559193, 0.544639, 0.529919, 0.515038, 0.500000,
      0.484810, 0.469472, 0.453991, 0.438371, 0.422618, 0.406737, 0.390731, 0.374607,
      0.358368, 0.342020, 0.325568, 0.309017, 0.292372, 0.275637, 0.258819, 0.241922,
      0.224951, 0.207912, 0.190809, 0.173648, 0.156434, 0.139173, 0.121869, 0.104528,
      0.087156, 0.069757, 0.052336, 0.034899, 0.017452, -0.000000, -0.017452,
      -0.034899, -0.052336, -0.069756, -0.087156, -0.104529, -0.121869, -0.139173,
      -0.156434, -0.173648, -0.190809, -0.207912, -0.224951, -0.241922, -0.258819,
      -0.275637, -0.292372, -0.309017, -0.325568, -0.342020, -0.358368, -0.374607,
      -0.390731, -0.406737, -0.422618, -0.438371, -0.453990, -0.469472, -0.484810,
      -0.500000, -0.515038, -0.529919, -0.544639, -0.559193, -0.573576, -0.587785,
      -0.601815, -0.615661, -0.629320, -0.642788, -0.656059, -0.669131, -0.681998,
      -0.694658, -0.707107, -0.719340, -0.731354, -0.743145, -0.754710, -0.766044,
      -0.777146, -0.788011, -0.798635, -0.809017, -0.819152, -0.829037, -0.838671,
      -0.848048, -0.857167, -0.866025, -0.874620, -0.882948, -0.891006, -0.898794,
      -0.906308, -0.913545, -0.920505, -0.927184, -0.933580, -0.939693, -0.945519,
      -0.951056, -0.956305, -0.961262, -0.965926, -0.970296, -0.974370, -0.978148,
      -0.981627, -0.984808, -0.987688, -0.990268, -0.992546, -0.994522, -0.996195,
      -0.997564, -0.998630, -0.999391, -0.999848, -1.000000, -0.999848, -0.999391,
      -0.998630, -0.997564, -0.996195, -0.994522, -0.992546, -0.990268, -0.987688,
      -0.984808, -0.981627, -0.978148, -0.974370, -0.970296, -0.965926, -0.961262,
      -0.956305, -0.951057, -0.945519, -0.939693, -0.933580, -0.927184, -0.920505,
      -0.913545, -0.906308, -0.898794, -0.891007, -0.882948, -0.874620, -0.866025,
      -0.857167, -0.848048, -0.838671, -0.829038, -0.819152, -0.809017, -0.798635,
      -0.788011, -0.777146, -0.766044, -0.754710, -0.743145, -0.731354, -0.719340,
      -0.707107, -0.694658, -0.681998, -0.669131, -0.656059, -0.642788, -0.629320,
      -0.615662, -0.601815, -0.587785, -0.573576, -0.559193, -0.544639, -0.529919,
      -0.515038, -0.500000, -0.484810, -0.469472, -0.453991, -0.438371, -0.422618,
      -0.406737, -0.390731, -0.374607, -0.358368, -0.342020, -0.325568, -0.309017,
      -0.292372, -0.275637, -0.258819, -0.241922, -0.224951, -0.207912, -0.190809,
      -0.173648, -0.156435, -0.139173, -0.121870, -0.104528, -0.087156, -0.069757,
      -0.052336, -0.034900, -0.017452, 0.000000, 0.017452, 0.034899, 0.052336,
      0.069757, 0.087156, 0.104528, 0.121869, 0.139173, 0.156435, 0.173648, 0.190809,
      0.207911, 0.224951, 0.241922, 0.258819, 0.275637, 0.292371, 0.309017, 0.325568,
```

```
      0.342020, 0.358368, 0.374606, 0.390731, 0.406737, 0.422618, 0.438371, 0.453990,
221   0.469472, 0.484810, 0.500000, 0.515038, 0.529919, 0.544639, 0.559193, 0.573576,
      0.587785, 0.601815, 0.615662, 0.629320, 0.642788, 0.656059, 0.669131, 0.681998,
223   0.694658, 0.707107, 0.719340, 0.731354, 0.743145, 0.754710, 0.766044, 0.777146,
      0.788011, 0.798636, 0.809017, 0.819152, 0.829037, 0.838671, 0.848048, 0.857167,
225   0.866025, 0.874620, 0.882948, 0.891007, 0.898794, 0.906308, 0.913546, 0.920505,
      0.927184, 0.933580, 0.939693, 0.945519, 0.951057, 0.956305, 0.961262, 0.965926,
227   0.970296, 0.974370, 0.978148, 0.981627, 0.984808, 0.987688, 0.990268, 0.992546,
      0.994522, 0.996195, 0.997564, 0.998630, 0.999391};
229
    Encoder knobRight(enc_pinA, enc_pinB); //(27, 26) teensy (motor controller) // motor encoder
        positions
231
    //================================================================================
233 // INITIAL SETUP
    //================================================================================
235
    void setup() {
237   // serial communication setup
      analogWriteResolution(12);        // set the analog output resolution to 8 bit
239   analogReadResolution(12);         // set the analog input resolution to 8 bit
      Serial.begin(115200);            // start the serial communication with a baud rate of 250000
241
      // configure GPIO pins
243   pinMode(PWM_pin, OUTPUT);        // configuring the PWM pin 7 to be an output. It needs to be
        updated in the interrupt function
      analogWrite(PWM_pin, 10);        // setting the initial duty cycle to be 10/255*100=3.92%
245   pinMode(EN_pin, OUTPUT);         // set pin 30 to be an output. This sends an enable signal
        to the teensy.
      digitalWrite(EN_pin, HIGH);       // sets the enable pin to be 0
247   pinMode(DIR_pin, OUTPUT);        // sets pin 32 to be an output. This determines direction of
        the motor rotation. It is effectively a sign. It needs to be updated in the interrupt
        function
      digitalWrite(DIR_pin, HIGH);     // sets the pin to be high initially, which I suspect means
        positive rotation direction (CCW theta_p). it is!!
249
251   // initialize the interrupt function and call it as well
      Timer1.start(tsample);                          // timer freqeuency [mu s]. Determines the
        frequency of the timer generated interrupt. 5556 mu s = refencence freq of 0.5 Hz: [mu s]
        = (1/x[Hz])/360[sin values]*10e6
253   Timer1.attachInterrupt(control_isr);      // attaching the function to be executed everytime
        the interrupt occurs.
      //
255   //  Timer0.start(tsample_EKF);                        // timer freqeuency [mu s]. Determines
        the frequency of the timer generated interrupt. 5556 mu s = refencence freq of 0.5 Hz: [mu
        s] = (1/x[Hz])/360[sin values]*10e6
      //  Timer0.attachInterrupt(EKF_isr);      // attaching the function to be executed everytime
        the interrupt occurs.
257
      Interaction_sensor.begin(44, 42);
259   Interaction_sensor.set_scale();
      Interaction_sensor.tare(); //Reset the scale to 0+
261   float zero_factor = Interaction_sensor.read_average(); //Get a baseline reading

263   Interaction_sensor.set_scale(21.614*1000 - 5.3109); //Adjust to this calibration factor
    //14840/1.3
265 //  Serial.print(zero_factor);

267 }


269
    //================================================================================
271 // MAIN LOOP
    //================================================================================
273
    void loop() {
275
    interaction_force  = Interaction_sensor.get_units(1);
277
    }
279


281
    //================================================================================
283 // FUNCTIONS
    //================================================================================
285
```

```
/////////////// Sensor data readings //////////////
void DataReading(float * Readings) {
  Motorencoder = knobRight.read();      // read the data from the motor encoder

  Readings[0] = ((float)Motorencoder)*(-1802.1/1000000) + 2.1791;          //position after the motor gear box before the worm gear [rad]

  float pendulum = analogRead(A0)*1.5201/1000 - 3.0754;                  // pendulum position
  Readings[1] = pendulumfilter.filter(pendulum);                         // applying a fitler

  //  Readings[6] = analogRead(velocity_feedback) * 3.3 / 4096;     // velocity of the motor.

  // motor velocity calculation
  motordd[0] = (motordd[1] + Readings[0]) * 0.5;
  motorvelfilter = (motordd[0] - motordd[1]) * invers_tsamples;
  Readings[2] = motorvelfilter;
  motordd[1] = motordd[0];

  // pendulum velocity
  dd[0] = (dd[1] + dd[2] + dd[3] + dd[4] + dd[5] + dd[6] + dd[7] + Readings[1]) * 0.1250;
  Readings[5] = dd[0];
  velfilter[0] = (dd[0] - dd[1]) * invers_tsamples;
  Readings[3] = velfilter[0];
  dd[7] = dd[6];
  dd[6] = dd[5];
  dd[5] = dd[4];
  dd[4] = dd[3];
  dd[3] = dd[2];
  dd[2] = dd[1];
  dd[1] = dd[0];

  //  Readings[4] = analogRead(current_feedback)* 3.3 / 4096-1.5;       // current of the motor. Both the velocity and the current had a bias value of 1.5, hence why it is subtracted.
}


// interrupt function with the control algorithms
void control_isr() {
  // reference
  // float AAA = 0.2;
  // float freqqq = 0.5;
  // float raddd = 2 * PI * freqqq * increment * tsamples;

  //  penref =  AAA * sinus2(raddd);          // sine ref
  //  penref = 0.4;                            // step input
  // penref = REF[increment];                 // human generated input
  ++increment;


  // sensor data
  DataReading(sendat);
  motorpos = sendat[0];
  penpos = sendat[1];
  motorvel = sendat[2];
  penvel = sendat[3];
  pendulum = sendat[4];
  float  penvel2 = qwffilter.filter(penvel);
  float theta_deflection = motorpos/n - penpos;
// Stiffness and torque VSM

  float a =       15.13  ;
  float b =   2.319e-16  ;
  float c =       1.596 ;
  float d =  -5.151e-17 ;

  float K_tot = 3*a*theta_deflection*theta_deflection+2*b*theta_deflection+c;
  float Torque_tot =
    a*theta_deflection*theta_deflection*theta_deflection+b*theta_deflection*theta_deflection+c*theta_deflection

// adaptive admittance filter
// Ga = Ga - (t_int*advantage-Torque_tot)*sign(t_int)*0.01;
// if(Ga<0.07){
//  Ga = 0.07;
//  }
```

```
//  if(Ga>20){
//  Ga=20;
//  }
  Ga = 1;

 // High level control
 float intertorque = interaction_force*0.16;
 float Kpay =2.2*9.82*0.3+1.2*9.82*0.15;
 float k_0 = 0.2;
  float C = Ga/Kpay + k_0;
  t_int = intertorque + C*Kpay*sinus2(penpos);


// payload estimation
//  float payload = Torque_tot/sinus2(penpos);
//  float filtered1 = (adcFilter1.filter(payload));
  float pay_er = abs(Torque_tot) - abs(T_pay*sinus2(penpos));           // slightly more
     robust payload caluculation. no division by zero.
  T_pay = T_pay + pay_er*0.05;

// reference calculation from admittance filter
  float ddtheta_ref = (t_int-Ba*dtheta_ref-Ga*theta_ref)/Ma;
  dtheta_ref = dtheta_ref +ddtheta_ref*tsamples;
  theta_ref = theta_ref +dtheta_ref*tsamples;
  penref = theta_ref;

// Errors
errorpenpos[0] = penref - penpos;     // error pendulum position
float errorpenvel = (errorpenpos[0] - errorpenpos[1]) * invers_tsamples;     //
     differentiation of the error
Errorint = Errorint + tsamples * (errorpenpos[0] + errorpenpos[1]) * 0.5;   // euler
     integration
errorpenpos[1] = errorpenpos[0];
float k_t = 0.0281;

float filtered1 = T_pay;


float x1 = filtered1;
float yy1 = K_tot;

 float g1[18] = {3.00955479863725, 2.26948522723677 , -0.604640550059455 , -0.446081029762172
     ,-0.037848137404820 7 ,0.0503935572058490  ,0.0427203837619960, 0.00837910515501254
     ,-0.000551700986604355, -0.00150555385063816 , -0.00195592248023878 ,
     -0.000420044419782067, -0.000192743543259711 ,5.68734684033775e-05 , 3.40415070878142e-05
     , 7.76897817872564e-06 , 3.92597776562606e-06 ,2.15397849014455e-06};

 float K_1 = g1[0] + g1[1] * x1 + g1[2] * yy1 + g1[3] * x1 * x1 + g1[4] * x1 * yy1 + g1[5] *
     yy1 * yy1 + g1[6]*x1*x1*x1 +g1[7]*x1*x1*yy1 + g1[8]*x1*yy1*yy1 + g1[9]*yy1*yy1*yy1 +
     g1[10]*x1*x1*x1*x1 + g1[11]*x1*x1*x1*yy1 + g1[12]*x1*x1*yy1*yy1 + g1[13]*x1*yy1*yy1*yy1 +
     g1[14]*x1*x1*x1*x1*x1 + g1[15]*x1*x1*x1*x1*yy1 + g1[16]*x1*x1*x1*yy1*yy1+
     g1[17]*x1*x1*yy1*yy1*yy1;

float g2[21] = { 0.866660933975961,  16.6549030037834, -2.28615642443827,  -6.26309227484011
     ,3.90072949202062 ,-1.14512070267785  ,0.758196752918755, -0.287306392844755
     ,-0.296035813759342 ,0.209449922775070, -0.0393598394194426,  0.0110744916899031,
     0.0114616435068732  ,0.0119396616266815,  -0.0137439691664128,  0.000741491312478620
     ,-0.000162036726504868, -0.000199182000466815,  -0.000165152659197599,
     -0.000207185193518740,  0.000320148346465096};

  float K_2 = g2[0] + g2[1] * x1 + g2[2] * yy1 + g2[3] * x1 * x1 + g2[4] * x1 * yy1 + g2[5] *
     yy1 * yy1 + g2[6]*x1*x1*x1 +g2[7]*x1*x1*yy1 + g2[8]*x1*yy1*yy1 + g2[9]*yy1*yy1*yy1 +
     g2[10]*x1*x1*x1*x1 + g2[11]*x1*x1*x1*yy1 + g2[12]*x1*x1*yy1*yy1 + g2[13]*x1*yy1*yy1*yy1 +
     g2[14]*yy1*yy1*yy1*yy1 + g2[15]*x1*x1*x1*x1*x1 + g2[16]*x1*x1*x1*x1*yy1 +
     g2[17]*x1*x1*x1*yy1*yy1+ g2[18]*x1*x1*yy1*yy1*yy1 + g2[19]*x1*yy1*yy1*yy1*yy1 +
     g2[20]*yy1*yy1*yy1*yy1*yy1;


float g3[21]={0.00326900575179457,  -0.00149016998451351 , 0.000621987586641654 ,
     0.000340809667572545 , -5.35407771882420e-05 ,-8.58783853519878e-05,-3.59385481040775e-05,
     3.01762836605809e-06 , 4.26900952034046e-06 , 7.10167883015404e-06 , 1.73770352427394e-06
     ,-5.48962421002208e-08, -1.92130778515887e-07, -1.51357275267619e-07
     ,-3.19140692164059e-07, -3.12953954467934e-08, -3.86001146580276e-10 ,3.39244295186396e-09
     , 2.70469618135053e-09 , 2.29679415056819e-09  ,5.89433273531944e-09};


  float K_3 = g3[0] + g3[1] * x1 + g3[2] * yy1 + g3[3] * x1 * x1 + g3[4] * x1 * yy1 + g3[5] *
     yy1 * yy1 + g3[6]*x1*x1*x1 +g3[7]*x1*x1*yy1 + g3[8]*x1*yy1*yy1 + g3[9]*yy1*yy1*yy1 +
```

```
    g3[10]*x1*x1*x1*x1 + g3[11]*x1*x1*x1*yy1 + g3[12]*x1*x1*yy1*yy1 + g3[13]*x1*yy1*yy1*yy1 +
    g3[14]*yy1*yy1*yy1*yy1 + g3[15]*x1*x1*x1*x1*x1 + g3[16]*x1*x1*x1*x1*yy1 +
    g3[17]*x1*x1*x1*yy1*yy1+ g3[18]*x1*x1*yy1*yy1*yy1 + g3[19]*x1*yy1*yy1*yy1*yy1 +
    g3[20]*yy1*yy1*yy1*yy1*yy1;

float g4[21] = {0.147066455458241,  -0.108697066299643, 0.0422367250206870
   ,0.0286031384922363 ,-0.00781255170216556,  -0.00454173783942967, -0.00323156584039833
   ,0.000703961605555293,  0.000476975486853568  ,0.000305641487724262,  0.000163042792306716
    ,-2.81152977454508e-05  ,-2.65030147152356e-05  ,-1.41152513393258e-05
   ,-1.18512062955196e-05, -3.02371726607541e-06 ,4.00331077015555e-07,
   5.31787718266138e-07, 3.05969379498834e-07, 1.86431896776983e-07, 1.96492621202475e-07};


float K_4 = g4[0] + g4[1] * x1 + g4[2] * yy1 + g4[3] * x1 * x1 + g4[4] * x1 * yy1 + g4[5] *
   yy1 * yy1 + g4[6]*x1*x1*x1 +g4[7]*x1*x1*yy1 + g4[8]*x1*yy1*yy1 + g4[9]*yy1*yy1*yy1 +
   g4[10]*x1*x1*x1*x1 + g4[11]*x1*x1*x1*yy1 + g4[12]*x1*x1*yy1*yy1 + g4[13]*x1*yy1*yy1*yy1 +
   g4[14]*yy1*yy1*yy1*yy1 + g4[15]*x1*x1*x1*x1*x1 + g4[16]*x1*x1*x1*x1*yy1 +
   g4[17]*x1*x1*x1*yy1*yy1+ g4[18]*x1*x1*yy1*yy1*yy1 + g4[19]*x1*yy1*yy1*yy1*yy1 +
   g4[20]*yy1*yy1*yy1*yy1*yy1;
float F_pre = K_2+K_4*n*((filtered1)/K_tot+1)+n*beta*(filtered1)/k_t;


// gain scheduling
u = - K_1 * penvel - K_2 * penpos - K_3 * motorvel - K_4 * motorpos + F_pre * penref;

// transformation to current
float I = u;
//float I = u/k_t;            // current input to the motor. The motor controller needs to
   run in current control mode before this method is relevant

// saturation funtion
float I_max = 5;               // max current input to the motor. It is the saturation value
   set in the ESCON motor contro program. The maximum should be kept at 5.5 A. From zhongyi
   paper max current is 8 A, and max speed is approx 15000 RPM
if (I > I_max) {
  I = I_max;
} else if (I < -I_max) {
  I = -I_max;
}

// determine the direction the motor should run
int dir;
if (I >= 0) {                           // defining the direction conditions.
  dir = 1;
  digitalWrite(DIR_pin, dir);
} else {
  dir = 0;
  digitalWrite(DIR_pin, dir);
  I = abs(I);                          // setting the current to be positive if the input
   is negative as the duty cycle can only be positive
}

duty_update = (I / I_max + 0.1) * 4095;          // new duty cycle value. Has to be positive
   and a value between 0 and 4095 as 12 bit is set

if (duty_update >= 4030) {
  duty_update = 4030;
}

analogWrite(PWM_pin, duty_update);       // sending the new duty to the motor control

// send data each iteration. To speed up the arduino, data can be stored in an array and then
    send at the end
  Serial.print(penref,3);
  Serial.print(',');
  Serial.print(penpos,3);
  Serial.print(',');
////  Serial.print(penvel,3);
////  Serial.print(',');
////  Serial.print(theta_deflection,3);
////  Serial.print(',');
////  Serial.print((0.074*9.81*0.1)*sinus2(penpos),3);
////  Serial.print(',');
//  Serial.print(t_int, 3);
//  Serial.print(',');
  Serial.print(motorpos/n, 3);
  Serial.print(',');
  Serial.print(T_pay, 3);
```

```
469   Serial.print(',');
      Serial.print(Torque_tot, 3);
471   Serial.print(',');
//  Serial.print(K_tot, 3);
473 //  Serial.print(',');
      Serial.print(t_int, 3);
475   Serial.print('\n');
}
477

479 // sign function
static inline int8_t sign(float val) {
481   if (val < 0) return -1;
      else if (val >= 0) return 1;
483 }

485

// sinus function used to calculate the sine of a value, by using look-up tables. Way faster
      than actually calculating sin(x). Includes interpolation for increased resolution.
487 float sinus2(float rad)
{
489   int index = 0;
      float sine = 0;
491   int negative = 0;
      int quadrant = 0;
493

      /*Obtaining a positive angle: for cosinus positive and negative angles gives the same
         result*/
495   if (rad < 0)
      {
497     rad *= -1;
        negative = 1;
499   }

501   /*Check in which quadrant the angle is*/
      //First in 360 degrees
503   rad -= (int)(rad / (2 * PI)) * 2 * PI;

505   if (rad != PI / 2)
      {
507     quadrant = (int)(rad / (PI / 2));
        rad -= PI / 2 * quadrant;
509   }

511   if ((quadrant == 0 || quadrant == 2))
      {
513     for (size_t f = 0; f < 92; f++)
        {
515       if (ANGLE[f] > rad)
          {
517         index = f;
            break;
519       }
        }
521   }
      else if (quadrant == 1 || quadrant == 3)
523   {
        rad = PI / 2 - rad;
525     for (size_t f = 0; f < 92; f++)
        {
527       if (ANGLE[f] > rad)
          {
529         index = f;
            break;
531       }
        }
533

      }
535   //Interpolation
      if (index != 0)
537   {
        sine = SINTABLE[index] - (ANGLE[index] - rad) *
539            (SINTABLE[index] - SINTABLE[index - 1]) / (ANGLE[index] - ANGLE[index - 1]);
      }
541   else
      {
543     sine = SINTABLE[index];
      }
```

```
545
  if ((quadrant == 2 || quadrant == 3) && (negative == 0))
547 {
    sine *= -1;
549 }
  else if ((quadrant == 0 || quadrant == 1) && (negative == 1))
551 {
    sine *= -1;
553 }

555 return sine;
}
557

559 // cosine function
float cosinus2(float rad)
561 {
  int index = 0;
563 float cosi = 0;
  int quadrant = 0;
565

  /*Obtaining a positive angle: for cosinus positive and negative angles gives the same
     result*/
567 if (rad < 0)
  {
569   rad *= -1;
  }
571

  /*Check in which quadrant the angle is*/
573 //First in 360 degrees
  rad -= (int)(rad / (2 * PI)) * 2 * PI;
575 //Quadrant

577

  if (rad != PI / 2)
579 {
    quadrant = (int)(rad / (PI / 2));
581   rad -= PI / 2 * quadrant;
  }
583 if (quadrant == 0 || quadrant == 3)
  {
585   for (size_t f = 0; f < 92; f++)
    {
587     if (ANGLE[f] > rad)
      {
589       index = f;
        break;
591     }
    }
593 }
  else if (quadrant == 1 || quadrant == 2)
595 {
    rad = PI / 2 - rad;
597   for (size_t f = 0; f < 92; f++)
    {
599     if (ANGLE[f] > rad)
      {
601       index = f;
        break;
603     }
    }
605

  }
607 //Interpolation
  if (index != 0)
609 {
    cosi = COSTABLE[index] - (ANGLE[index] - rad) *
611         (COSTABLE[index] - COSTABLE[index - 1]) / (ANGLE[index] - ANGLE[index - 1]);
  }
613 else
  {
615   cosi = COSTABLE[index];
  }
617

  if (quadrant == 1 || quadrant == 2)
619 {
    cosi *= -1;
621 }
```

```
623     return cosi;
}
```