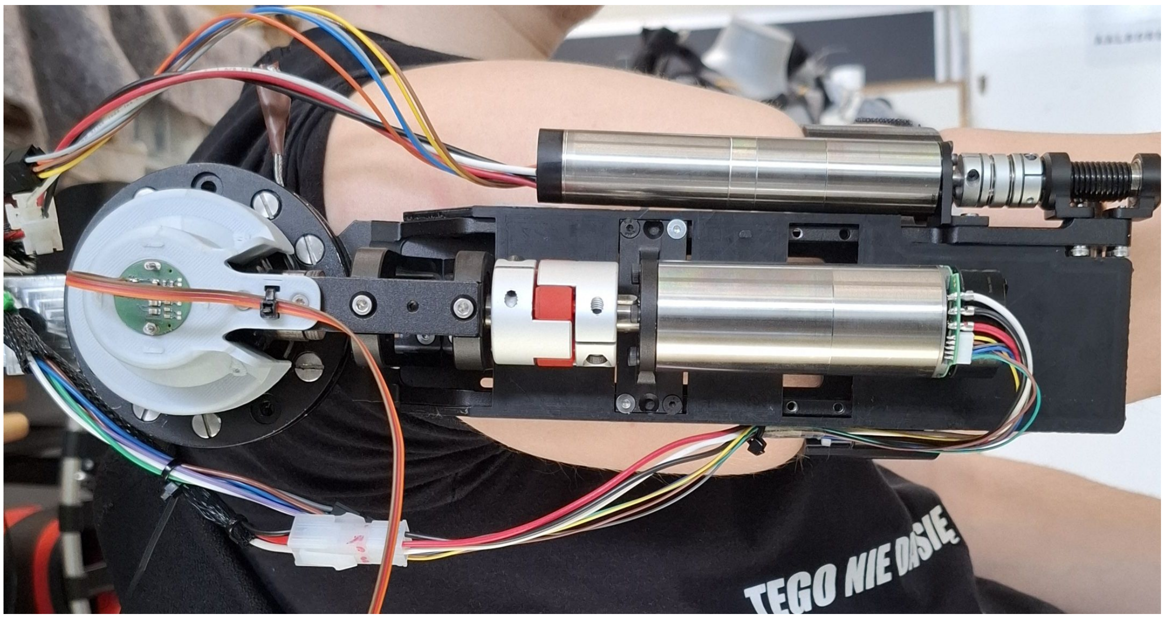
Neural Network PID Control of a Variable-Stiffness Shoulder Exoskeleton with Load Prediction

Project report

ROB10 - Group 1068



Aalborg Universitet
Det Tekniske Fakultet for IT og Design

AALBORG UNIVERSITY
STUDENT REPORT

The Technological Faculty for IT og Design
Niels Jernes Vej 10, 9220 Aalborg Øst
http://www.aau.dk

Title:
Neural Network PID Control of a Variable-Stiffness Shoulder Exoskeleton with Load Prediction

Theme:
Technological Project Work

Project Period:
February 2025 - May 2025

Project Group:
1068

Participant(s):
Julian Witold Wagner 20231011

Supervisor(s):
Shaoping Bai

Copies: 1

Page Numbers: 80

Date of Completion:
3. June 2025

Abstract:

This project presents the design and implementation of a control system for a variable-stiffness shoulder exoskeleton aimed at reducing work-related musculoskeletal disorders (WMSDs) during overhead or static load-bearing tasks. A dual-layer control architecture was developed: a low-level controller using a neural network-scheduled PID (NNPID) for torque compensation and adaptability, and a high-level controller based on a Force Myography (FMG) armband to predict load. The NNPID dynamically adjusts control gains in real-time, improving performance across varying loads and joint stiffness. A Long Short-Term Memory (LSTM) network was employed to interpret biosignals for load estimation, enhancing intention-aware assistance. The system was modeled and simulated in Simulink. Implementation of the demonstration on embedded hardware using an Arduino Due and Maxon motor system. Experimental results demonstrate better adaptability under nonlinear dynamics compared to conventional PID controllers. The findings underscore the feasibility and advantages of intelligent control strategies in industrial wearable robotics.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Aalborg University, 24. May 2025

The author give thanks to the supervisor Shaoping Bai for the guidance and assistance. A special thanks to Arunabha Majumder, student Ph.D., for designing, assembling the exoskeleton and support during the project. Additionally, a special thanks to Yu Zhu, student Ph.D.,for for his introduction and assistance to the BIOX AAL band.

Julian Witold Wagner
jwagne23@student.aau.dk

Contents

1 Introduction **1**
1.1 Initial problems statement 1

2 State of the art **2**
2.1 Industrial Exoskeletons 2
2.2 VSM - Based exoskeleton 7
2.3 Control of Exoskeletons 9

3 Requirements **13**
3.1 Final problem statement 13
3.2 Design Requirements 13
3.3 Delimitations 13

4 Exoskeleton design adn modeling **14**
4.1 Exoskeleton 14
4.2 Modeling 18

5 Controller Design **24**
5.1 Low-level control 24
5.2 High-level control 25
5.3 Final Design 26

6 Implementation **27**
6.1 Low-Level Control 27
6.2 High-level controll 36

7 Testing **41**
7.1 Low-Level Controller 41
7.2 High-level control 64

8 Discussion **67**
8.1 Modeling 67
8.2 Low-level control 67
8.3 High-level control 68
8.4 Other Consideration 68

9 Conclusion **69**

Bibliography **71**

Appendices **73**

Appendix A Github repository **74**

Appendix B Gravity center and Moment of Inertia calculations **75**

Appendix C Data gathered in Simulink **76**

Appendix D Elbow joint tuned PID **80**

1 Introduction

Between 2018 and 2020[1], there were 75.4 thousand nonfatal injuries reported. Out of those, over 20% were due to Work-related Musculoskeletal Disorders (WMSDs). These injuries are not only common but also a leading cause of lost work time and reduced productivity across various industries. WMSDs [2] are caused by fixed or constrained body positions, continual repetition of movements, force concentrated on specific parts of the body, such as the hand or wrist, or a pace of work that does not allow sufficient recovery between movements. Tasks that demand overhead work or static load-bearing postures are particularly demanding and frequently result in shoulder, neck, and upper back strain.

In response to these challenges,[3] exoskeleton technology has emerged as a promising solution for assisting workers during physically demanding tasks. These systems are designed to reduce muscular strain, delay fatigue, and lower the risk of injury by providing mechanical support to the body. Among recent developments, variable stiffness exoskeletons offer enhanced adaptability, adjusting their support in real time based on the task at hand. They also enable workers to maintain natural movement while still receiving the necessary assistance, particularly during physically taxing activities such as holding tools overhead or performing repetitive upper-limb actions

However, for active exoskeletons to provide effective and intuitive assistance, they must be capable of predicting or detecting the user's intention. It is vital to detect when a user is initiating movement, adjusting posture, or applying force so that the system can ensure seamless reaction to support the user without interfering with voluntary motion. Achieving this requires a control strategy that not only interprets user intent in real time, but also adjusts support accordingly based on task dynamics. This report focuses on the development of such a control system, aimed at enabling responsive and adaptive assistance in upper-limb exoskeletons used for overhead or static work tasks.

1.1 Initial problems statement

Work-related musculoskeletal disorders remain a major issue, especially in tasks involving overhead work or static load-bearing. Existing exoskeletons may struggle to adapt to different tasks or user movements, which may lead to reduced support or restricted mobility. There is a need for a control strategy that enables real-time, intention-aware assistance to reduce fatigue and injury risk, particularly in overhead work scenarios.

"How can user-intent-aware control be implemented to manage stiffness and support in variable stiffness exoskeletons?"

2 State of the art

The development of exoskeletons, has seen significant progress over last year. These devices are designed to augment human performance, support physical rehabilitation, and reduce occupational injuries in industrial settings. A growing body of research and commercial innovation is driving advancements in their mechanical design, control strategies, and integration with human operators.

This chapter provides a comprehensive overview of the current state of the art in exoskeleton technologies. The study commences with a survey of industrial exoskeletons, with a focus on their applications and design. Subsequently, an exploration of devices and systems developed utilising Variable Stiffness Mechanism (VSM) principles is presented, with the objective of enhancing adaptability and safety during human-robot interaction. The final section of this study examines the control of exoskeletons in which a review of various approaches to exoskeleton control is presented and discussed.

2.1 Industrial Exoskeletons

Industrial exoskeletons can be broadly categorized into two main groups: passive and active systems.

Passive Exoskeletons

Passive exoskeletons operate without powered actuators, instead utilizing mechanical components such as springs and levers to support human motion and redistribute physical loads. These systems are typically lighter and afford a greater range of motion, but provide limited assistance when handling heavy or variable loads.
One example is the **Ottobock Shoulder** exoskeleton, which uses a spring/leverage system to relieve shoulder strain and weighs approximately 1.9 kg [4]. Its design is compact and ergonomic, as seen in Figure 2.1.

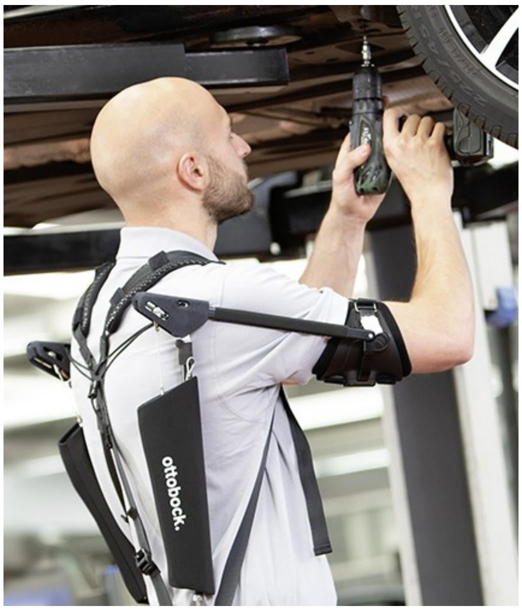


Figure 2.1: Ottobock Shoulder exoskeleton. Taken from[4]

Another similar system is the **Auxivo DeltaSuit**, which provides support through an adjustable spring mechanism offering torque between 5.2 Nm and 6.6 Nm. The DeltaSuit features a sleek design with minimal protruding components [5], shown in Figure 2.2.

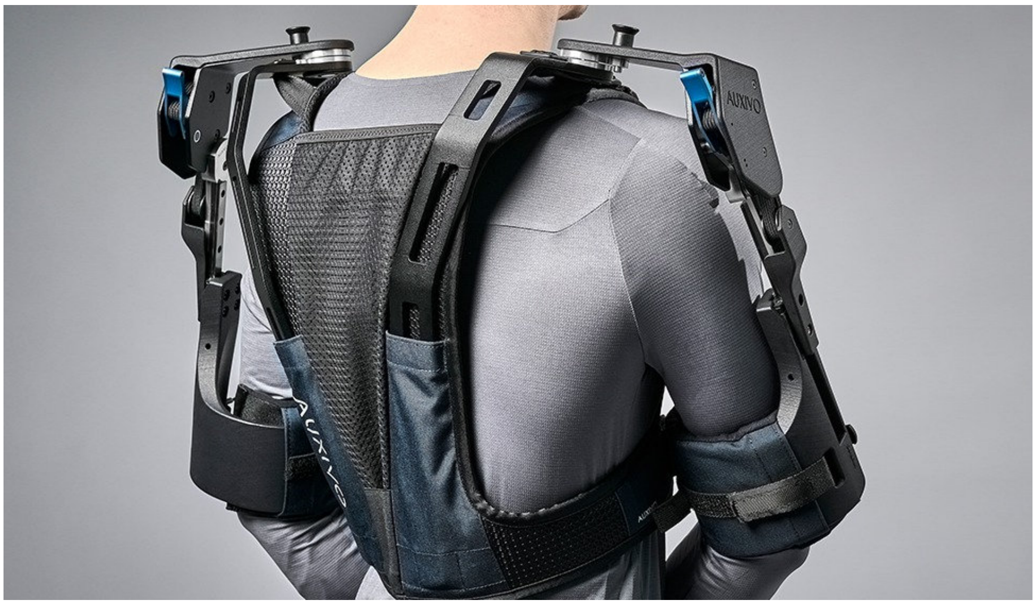


Figure 2.2: Auxivo DeltaSuit exoskeleton. Taken from[5]

The **SuitX IX Shoulder Air** is a 2.22 kg passive exoskeleton that stores energy during downward arm motion and releases it during lifting [6]. Its design, depicted in Figure 2.3, builds upon previous passive systems.

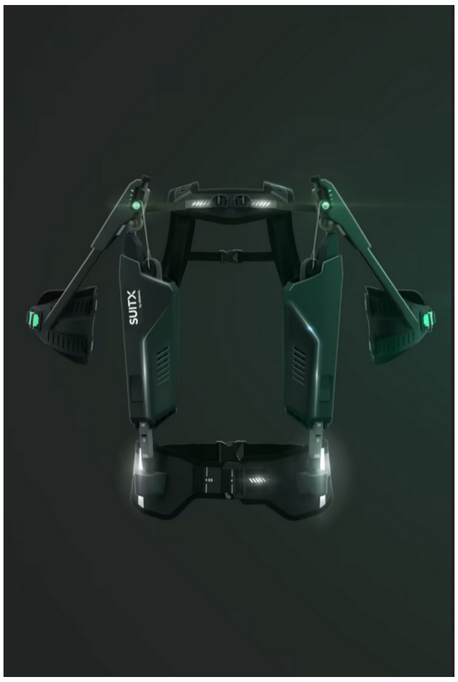


Figure 2.3: SuitX IX Shoulder Air exoskeleton. Taken from[6]

Ekso Bionics' EVO is another spring-based exoskeleton offering 2.2–6.8 kg of lift assistance per arm [7], illustrated in Figure 2.4.

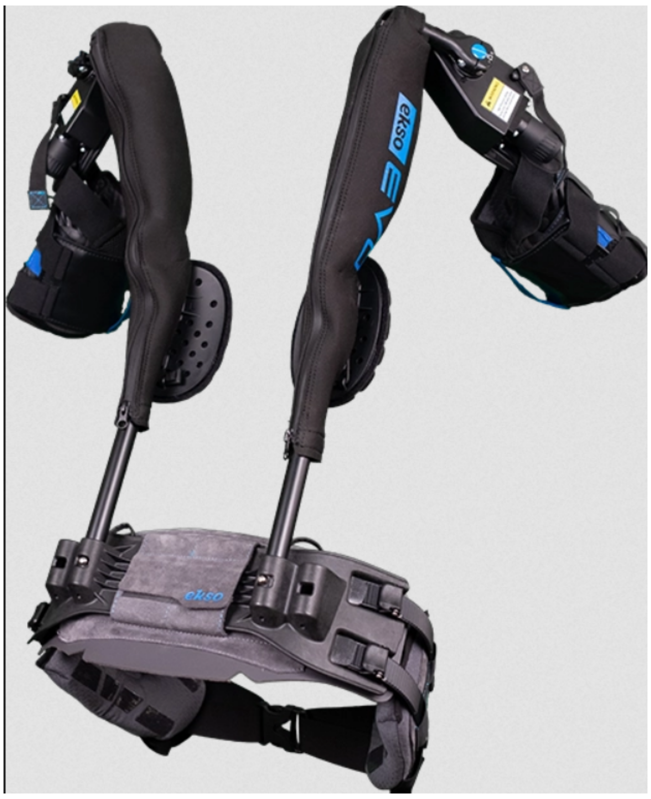


Figure 2.4: Ekso EVO exoskeleton. Taken from[7]

Lastly, the **HAPO UP** model by Ergosanté offers up to 3.8 kg of support per arm through a lightweight design similar to its counterparts [8], as shown in Figure 2.5.

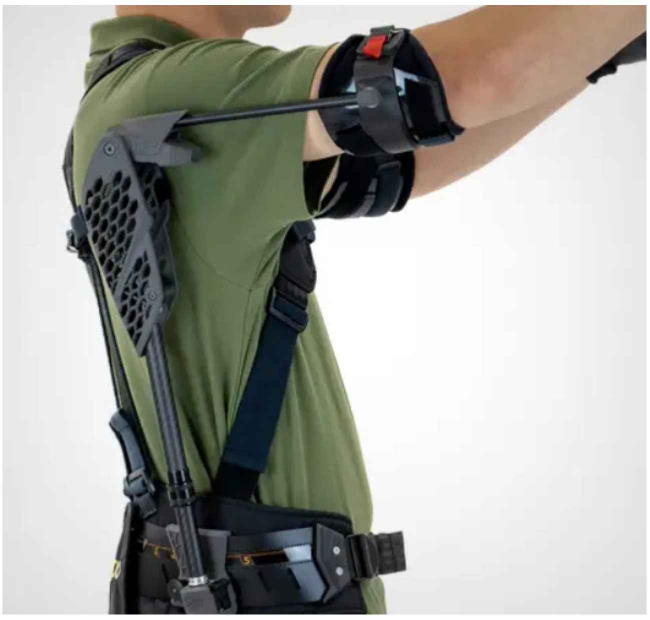


Figure 2.5: HAPO UP exoskeleton. Taken from[8]

Despite their shared “backpack-style” form factor and ergonomic benefits, passive exoskeletons are limited in terms of lifting power and adaptability. Some designs integrate mechanical adjustment systems to offer varying levels of support for different tasks.

Active Exoskeletons

In contrast, active exoskeletons utilize powered actuators—such as electric motors or pneumatics—to provide stronger and adjustable support. These systems are more complex and typically heavier, but excel in tasks requiring higher force output and real-time adaptability.
The **ExoIQ S700**, shown in Figure 2.6, is a pneumatically powered shoulder exoskeleton that weighs 6.5 kg and provides up to 5 kg of assistance per arm, powered by a battery-operated compressor [9].

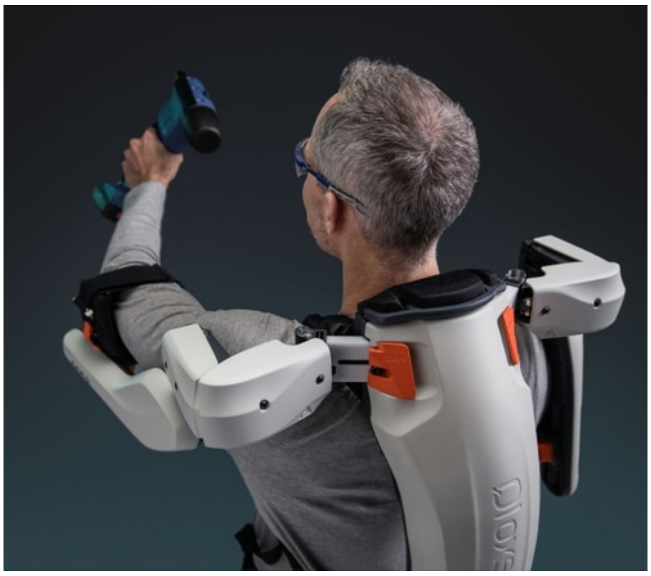


Figure 2.6: ExoIQ S700 exoskeleton. Taken from[9]

Another advanced model is the **Agadexo Shoulder** exoskeleton, a semi-active system enhanced by artificial intelligence that adapts support based on detected intent and payload. It supports up to 8 kg per arm and payloads up to 25 kg, with an operating time of approximately 8 hours [10], visualized in Figure 2.7.

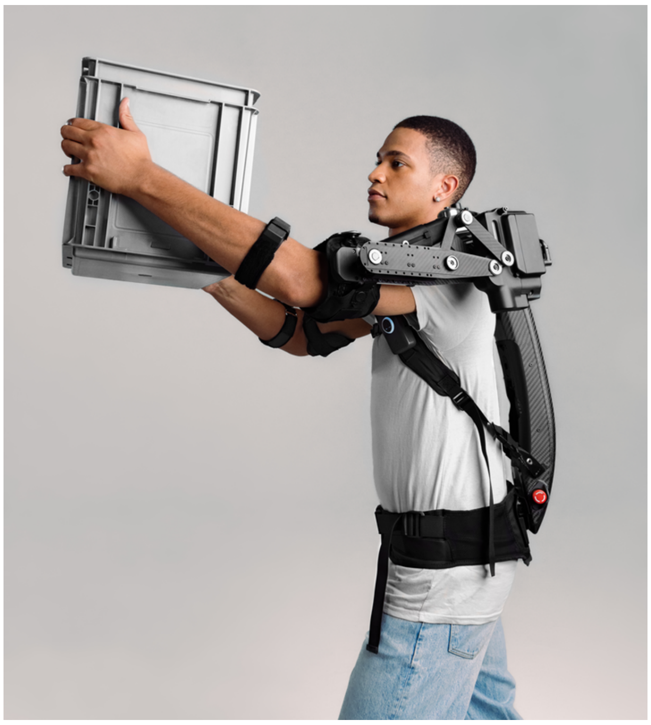


Figure 2.7: Agadexo Shoulder exoskeleton. Taken from [10]

For heavy-duty industrial applications, the **Ant-WA1** exoskeleton provides over 40 kg of lifting assistance and weighs 8.4 kg. It includes four electric motors located at the shoulders and hips, making it highly capable for lifting-intensive environments [11], as depicted in Figure 2.8.

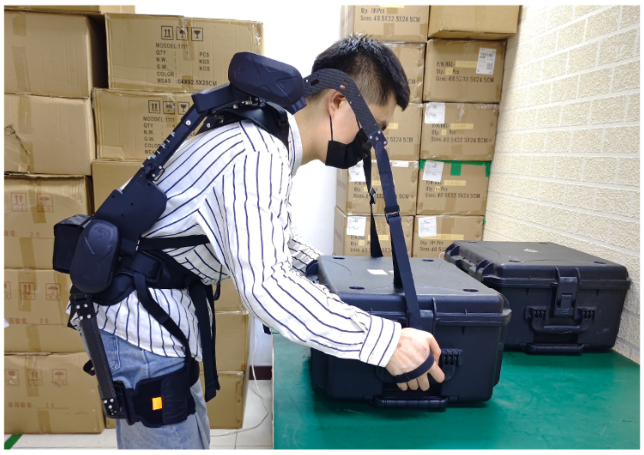


Figure 2.8: Ant-WA1 exoskeleton. Taken from [11]

In conclusion, passive exoskeletons present a lightweight, cost-effective solution for reducing fatigue and musculoskeletal strain during repetitive or overhead tasks. Their simple mechanical design ensures reliability, minimal maintenance, and a high degree of user mobility. However, their fixed assistance levels and limited adaptability might make them insufficient for tasks involving variable or heavy loads.

On the other hand, active exoskeletons leverage powered actuators, sensors, and sometimes artificial intelligence to dynamically adjust the level of assistance based on user intent and task demands. These systems are better suited for complex, high-load industrial applications where assistance and task adaptability are critical. Despite their advantages in performance, active exoskeletons come with trade-offs such as increased weight, energy dependency, and greater system complexity, which can impact usability and cost.

2.2 VSM - Based exoskeleton

At Aalborg University (AAU), there has been sustained research into the development of upper-limb exoskeletons, with a particular emphasis on enhancing human-robot interaction through adaptive mechanical properties [12]. The VIEXO project, in particular, focuses on incorporating variable impedance to better emulate the compliant and responsive nature of human arm movement. This approach is intended to improve both the safety and efficiency of exoskeleton systems in dynamic work environments.

A core element of this approach is the implementation of Variable Stiffness Mechanisms (VSMs), which enable the mechanical stiffness of joints to be tuned either structurally or in real time. These mechanisms form the basis for various actuation strategies that enable exoskeletons to offer the right level of assistance depending on user intention, movement phase, or external load, as illustrated in Figure 2.9.

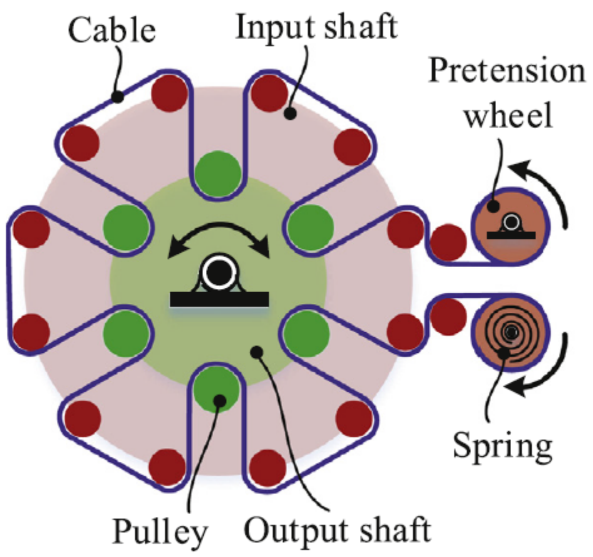


Figure 2.9: Concept of VSM. Taken from [13]

Building on the goals of the VIEXO project, recent research at AAU has explored multiple designs of Variable Stiffness Mechanisms (VSMs) tailored for passive and semi-passive exoskeletons. These VSMs are designed to passively joint stiffness passively or semi-actively to align with biomechanical demands during industrial tasks. Several innovative concepts have been developed using zero-length four-bar linkages combined with compliant elements, such as linear springs and cable-pulley systems [13], [14].

One of the earliest implementations featured a reconfigurable revolute joint where stiffness profiles (softening, linear, or hardening) could be adjusted via cable routing and spring pretension, without active actuation. This compact and modular architecture demonstrated strong potential for wearable applications, enabling energy-efficient and safe torque support in multi-joint systems.

Subsequent advancements have led to the integration of VSM into wearable shoulder exoskeletons [15], [16], seen in Figure 2.10. These designs applied parametric optimization methods to match human arm torque requirements across a wide range of motion. Notably, the latest prototype demonstrated three degrees of freedom (DOFs), compact dimensions (under 0.8 kg), and adjustable pretension modules, providing customizable assistance while preserving shoulder mobility.

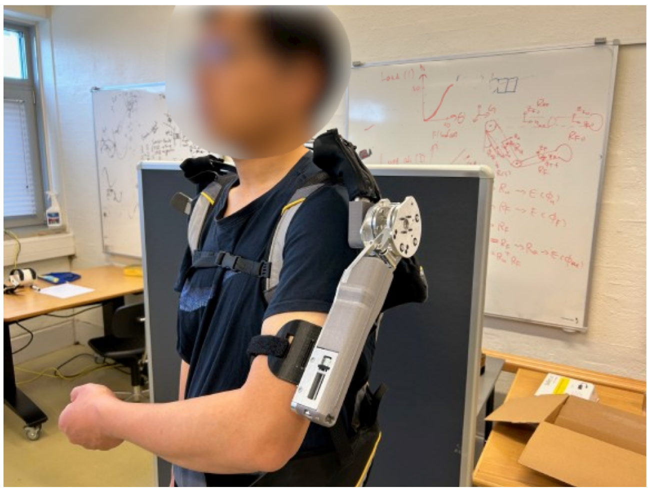


Figure 2.10: Passive shoulder exoskeleton equipped with VSM. Taken from [16]

Experimental validation, including surface electromyography (sEMG) analysis, confirmed that these VSM-based exoskeletons effectively reduce muscle activity in key muscle groups such as the anterior deltoid and biceps brachii during overhead load lifting. Compared to commercial systems like the Skelex 360, AAU's VSM exoskeletons achieved greater muscle activity reduction while maintaining full mobility.

The most recent development at AAU involves a hybrid exoskeleton prototype shown in Figure 2.11 capable of operating in multiple modes. This multi-mode configuration allows the exoskeleton to not only provide passive support via the VSM, but also to engage a motor for additional torque assistance when required. Furthermore, when no assistance is needed, both the VSM and motor can be disengaged, enabling free and unobstructed movement for the user.

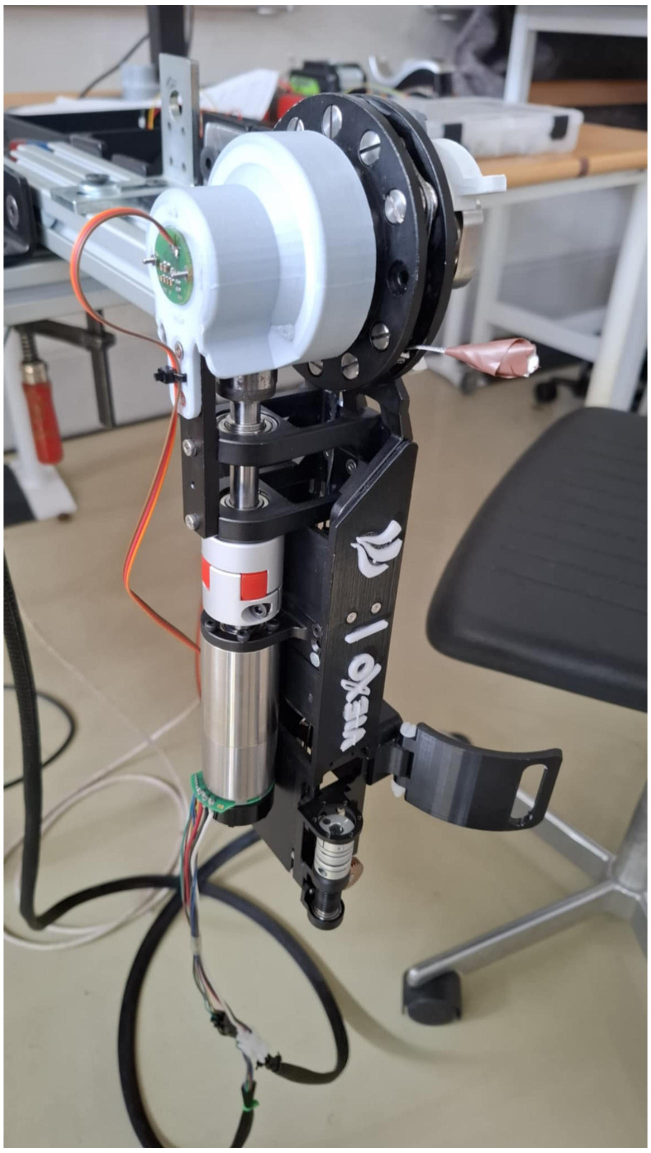


Figure 2.11: Shoulder exoskeleton prototype developed at AAU.

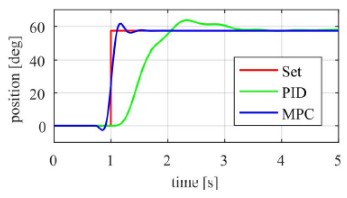
Despite their benefits, VSMs present challenges due to inherent nonlinearities—particularly when deployed in systems requiring large motion ranges. These complexities necessitate robust and adaptive control strategies to ensure intuitive and reliable human assistance.

2.3 Control of Exoskeletons

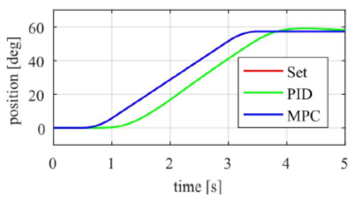
Effective control of assistive exoskeletons presents a complex challenge due to the nonlinear and human-in-the-loop nature of such systems. The integration of Variable Stiffness Mechanisms (VSMs), nonlinear joint dynamics, and user intention introduces the need for advanced control strategies that can ensure stability, responsiveness, and adaptability across various tasks. In this section, several promising control approaches are reviewed, based on recent literature covering predictive control, impedance-based methods, hybrid high- and low-level structures, and learning-based intent prediction.

Model Predictive Control (MPC) and Nonlinear MPC

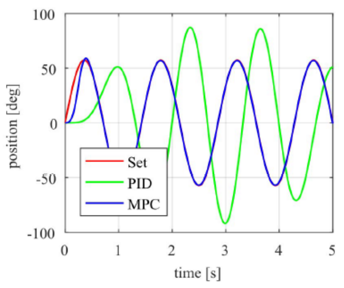
Model Predictive Control (MPC) is gaining traction in exoskeleton applications due to its ability to handle multi-variable systems with constraints, especially when nonlinearity is involved. In solution proposed by Szumowski et al. [17], MPC is applied to a system composed of a DC motor with gear reduction, a spring-based variable stiffness mechanism, and an inertial load. Although the spring element introduces nonlinear behavior, the authors formulate an optimization problem suitable for matrix-based solution methods. The controller calculates the optimal spring stiffness and then actuates the motor accordingly. Results demonstrate significant performance improvements over a traditional PID controller, particularly in tracking accuracy and control effort, as shown in Figure 2.12a, Figure 2.12b and Figure 2.12c.



(a) MPC step signal.
Taken from [17]



(b) MPC ramp signal.
Taken from [17]



(c) MPC sin wave signal.
Taken from [17]

Figure 2.12: Response of MPC to different signals

A similar effort is presented by Tahamipour et al. [18], where a Nonlinear MPC (NMPC) is used in a simulated exoskeleton system. The NMPC controller predicts the desired torque using a human-exoskeleton model and includes a filtering layer that defines the level of assistance provided to the user. This filtering is manually tuned, allowing user-specific adaptation. While position tracking showed only modest improvement compared to a PID baseline, the NMPC exhibited superior energy efficiency and smoother control actions. However, this work remains at the simulation stage, and real-world validation is needed.

Impedance Control with Intention Prediction

Impedance control remains one of the most widely used strategies for compliant human-robot interaction, owing to its ability to regulate the mechanical interaction dynamics between human and exoskeleton. The Khan et al. [19] implement an impedance controller in a 7-DOF exoskeleton, with two actively controlled joints at the elbow and shoulder. To enhance intention detection, they employ a Radial Basis Function Neural Network (RBFNN) trained on data from a Muscle Circumference Sensor (MCS)—a novel alternative to conventional electromyography (EMG) sensors. However, the study highlights several limitations: MCS data is only effective on muscle groups with significant volume change, such as the biceps, and it is not suitable for estimating desired movement speed. These constraints pose challenges for real-time implementation and suggest that additional sensing or hybrid models may be required.

Hierarchical Control with Mode Switching

An interesting hybrid control architecture is explored by Ding et al. [20], where the focus is on assisting workers during overhead tasks. The proposed system employs a two-layer control strategy. The low-level controller consists of a PD controller with force feedback and a feedforward disturbance observer to reduce steady-state errors and compensate for user-induced perturbations. A high-level controller determines the assistive mode based on IMU-derived shoulder flexion angles and joint accelerations. Three assistive sub-modes are defined—holding, raising, and lowering—each with customized control parameters. Additionally, a safety mode is triggered when shoulder acceleration exceeds a defined threshold, temporarily setting the desired torque to zero. Although the system shows promising reductions in muscle activation, it requires extensive calibration and lacks robustness across varied tasks. The authors suggest that future work could incorporate EMG signals to improve intention detection within the high-level controller.

Predictive Control Using Deep Learning

Souza et al. [21] investigate two Long Short-Term Memory (LSTM)-based neural network strategies for predicting torque commands in an exoskeleton controller. The first strategy, PJ+ID, predicts future joint positions and computes torque via inverse dynamics. The second, ID+PT, uses inverse dynamics on past joint positions to directly predict the desired torque. The latter method outperformed the former due to a lower-dimensional prediction target (four values instead of twenty), which significantly reduced training complexity and improved accuracy. Although trained on simulated data, the study points out a key limitation: the human-exoskeleton interaction loop is not fully modeled during training. This indicates the need for further research in combining physical sensing with neural prediction to close the control loop in a real-world setting.

Neural Network PID Control for Variable Stiffness

Hu et al. [22] design a control scheme tailored for a variable stiffness joint intended for upper-limb rehabilitation. Their solution leverages a Neural Network PID (NN-PID) controller that dynamically adjusts the PID gains K_P,K_I,K_D based on desired torque, external disturbances, and current stiffness settings. Using MATLAB Simulink, they first model the joint dynamics and optimize the gain parameters under different stiffness conditions. This training data is then fed into a Backpropagation Neural Network with one hidden layer using a tansig activation function and a purelin output layer. The NN is trained offline.

Simulation results show notable improvements in system performance, including reduced rise and settling times for step responses. The controller is further validated in hardware, where a subject performs elbow flexion from 0° to 90° and back, with an 8 Nm resistance torque applied. Experiments with both low and high stiffness settings confirm the controller's effectiveness in torque trajectory tracking. This work demonstrates the feasibility of intelligent gain tuning for exoskeleton control, offering a scalable path for adaptive rehabilitation systems.

Online Learning-Based Impedance Adaptation
Xiong at al [23] propose a novel learning-based control model for portable elbow exoskeletons that provides both Assist-as-Needed (AAN) and Resist-as-Needed (RAN) functionalities. Uniquely, the controller operates without external sensors such as EMG or force sensors, and without passive compliance mechanisms like springs. Instead, it relies solely on internal joint position and velocity feedback. The model combines an Iterative Learning Mechanism (ILM), which gradually learns feedforward torque based on previous task errors, with an Online Impedance Adaptation Mechanism (OIAM) that adjusts joint stiffness and damping in real time. In AAN mode, both mechanisms work together to minimize tracking error and compensate for involuntary user movement. In RAN mode, only the impedance adaptation is used to provide resistance, simulating a virtual adjustable spring-damper system that increases effort required from the user. Experiments on three human subjects using the lightweight POW-EXO exoskeleton (0.425 kg) show that the controller effectively adapts to individual user dynamics, improving accuracy and involvement over repeated trials. This model stands out as a low-sensor, multifunctional solution for personalized exoskeleton assistance and training, particularly in home-based or minimally-instrumented settings.

Force Myography-Based Control

While electromyography (EMG) remains a standard for intention detection, it suffers from practical issues such as signal variability due to electrode placement, skin conditions, and the need for preparation. Islam et al. [24], the authors explore the use of Force Myography (FMG) as a more robust alternative. FMG sensors measure changes in muscle volume and pressure using pressure bands, offering a higher signal-to-noise ratio and more consistent readings under dynamic conditions. In this study, FMG data is used to estimate the external load being carried by the user and to determine the appropriate assistive torque. Results show promising accuracy in payload estimation and torque adaptation. However, the authors also note that FMG systems are sensitive to external interactions—particularly when the exoskeleton structure applies pressure on the sensor band—which can result in incorrect torque predictions. This highlights the need for careful integration between wearable sensing and the mechanical interface.

Overall, the literature reveals a trend toward hybrid and adaptive control systems that integrate robust low-level controllers with high-level intent recognition and context awareness. Methods such as MPC and NMPC address system nonlinearity and constraint handling, while impedance-based and hierarchical controllers facilitate safe human-robot interaction. Learning-based approaches, including FMG and neural networks, offer new avenues for user intention prediction but require further development for real-time reliability and integration with wearable hardware. Moving forward, control strategies that combine mechanical compliance with intelligent, sensor-driven adaptation are likely to play a central role in achieving intuitive, responsive, and task-aware exoskeleton assistance.

3 Requirements

3.1 Final problem statement

Thus, based on the findings and research conducted in chapter 2, it can be concluded that multiple control strategies, both at the low and high levels, are available and can be combined to control exoskeletons incorporating Variable Stiffness Mechanisms. Based on these findings, the problem statement is refined to:

"What type of control strategy can best support an upper-limb exoskeleton equipped with a variable stiffness mechanism?"

3.2 Design Requirements

Based on the research conducted, the objectives of this project can be formulated as a set of design requirements that must be addressed to achieve a complete solution. These are defined as follows:

1. The controller has to be implemented into embedded system board.
2. The overshoot of the system response must not exceed 5% of the reference signal amplitude.
3. The rise time of the controller must be within 1 ± 0.1 seconds.
4. The trajectory tracking error should not exceed $5°$.
5. The intention prediction system must detect the desired joint angle within $5°$ of error.
6. The intention prediction system must reliably distinguish between movement initiation and position holding.
7. The intention prediction system must detect the load that the user is holding within 0.5kg.

3.3 Delimitations

One major limitation of the current exoskeleton design is the use of a non-backdrivable motor. This means the motor cannot be driven passively by external forces acting on its output shaft. As a result, when the motor is idle but still connected with the coupler, it effectively locks the exoskeleton in place. This restricts natural movement and limits usability in certain scenarios.
Due to this constraint, implementation and testing of control algorithms for the VSM had to be conducted theoretically using Simulink simulations. Practical validation would be carried out on an alternative exoskeleton system.

4 Exoskeleton design adn modeling

This chapter describes setup of the exoskeleton design and dynamic modeling.

4.1 Exoskeleton

The entire exoskeleton system consists of four main components: the ESCON 50/5 motor controller, an Arduino Due microcontroller, a Maxon EC motor, and the physical exoskeleton frame. Each component plays a crucial role in the system's operation and will be briefly described below. The Figure 4.1 shows whole exoskeleton system fixed on the table with (1) exoskeleton fixed to frame, (2) Control Hardware and (3) safety switch whuchg would be used to stop any unwanted behaviour.

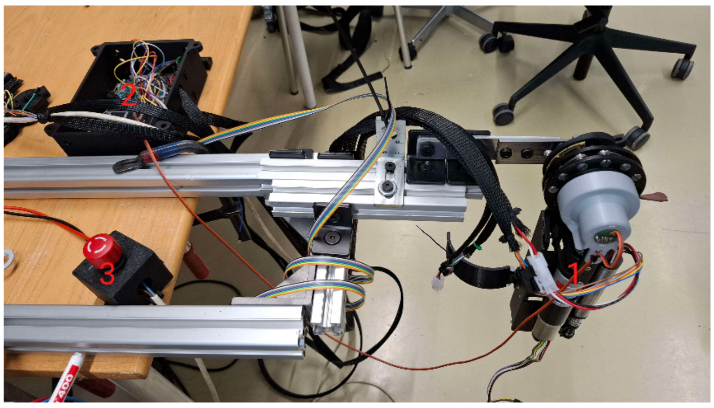


Figure 4.1: Whole exoskeleton setup

Motor

The motor used in the exoskeleton is a brushless EC motor [25] manufactured by Maxon, as seen in Figure 4.2. It includes an integrated Hall sensor for position and speed feedback. Additionally, the motor is paired with a high-ratio gearbox featuring a 1:1221 reduction. This setup enables high torque output, which is necessary for the exoskeleton's operation. The presence of the Hall sensor supports precise closed-loop control of the motor's behavior.

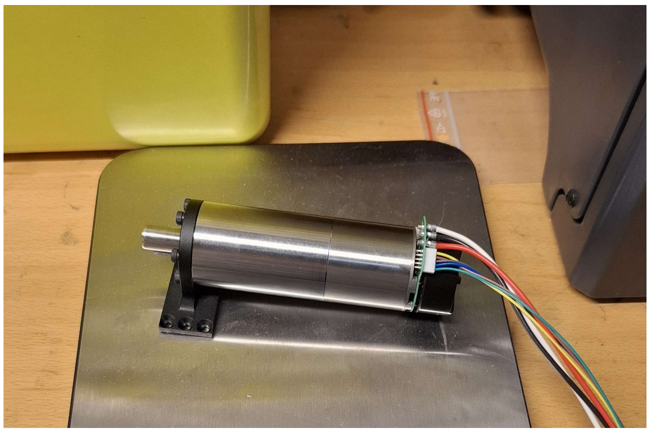


Figure 4.2: Maxon motor removed from frame

Control Hardware

The motor is driven by the ESCON 50/5 controller [26], selected for its configurability. The controller parameters were configured using Maxon's Motion Studio software. An Arduino Due microcontroller serves as the main interface between the sensors and the ESCON controller. It is responsible for generating the control signals and handling data acquisition from the exoskeleton's sensors.

Initially, the Arduino Due [27] used PWM (Pulse Width Modulation) for speed control. However, due to issues with the PWM frequency, the control method was switched to analog voltage control. The maximum motor speed of 10,600 rpm corresponds to an analog input of 3.3 V—the maximum output voltage by the Arduino Due. To prevent unintended movement when the motor is idle, a minimum offset voltage of 0.05 V is used. This ensures the ESCON receives a clean zero-speed signal, avoiding the interpretation of small PWM fluctuations as movement commands. Both the ESCON and Arduino Due are housed in a custom 3D-printed enclosure to improve system integration and safety, shown in Figure 4.3.

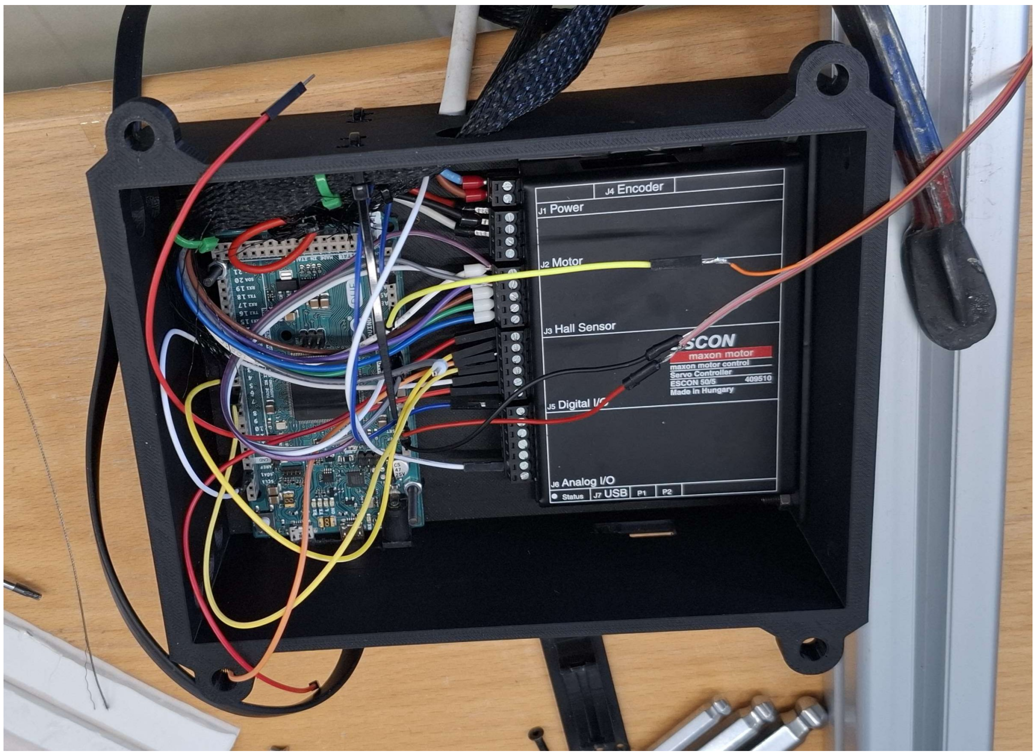


Figure 4.3: ESCON 50/5 and Arduino Due in custom box

Sensors

In addition to the Hall sensor built into the motor, the exoskeleton is equipped with two absolute encoders mounted on the components of the Variable Stiffness Mechanism (VSM). One encoder RMB20VB [28] tracks the position of the bevel gear connected to the pendulum, shown in Figure 4.4, while the other AksIM [29] monitors deflection in the VSM, shown in Figure 4.5. These sensors are crucial for feedback control and system evaluation.

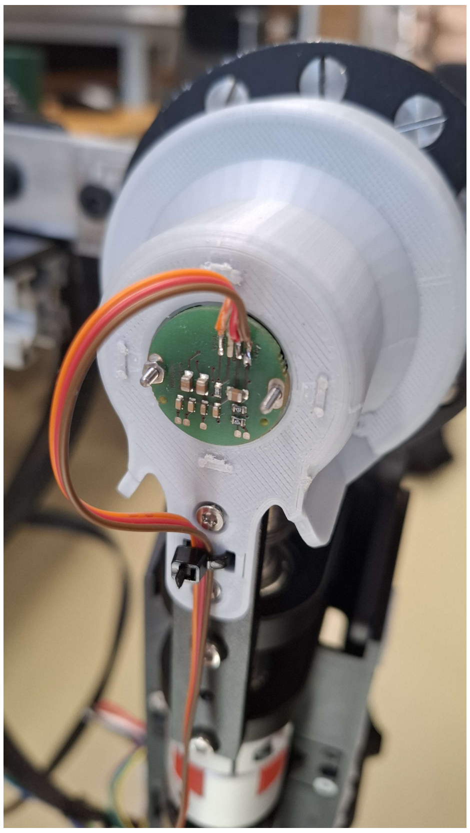


Figure 4.4: Exoskeleton's position encoder

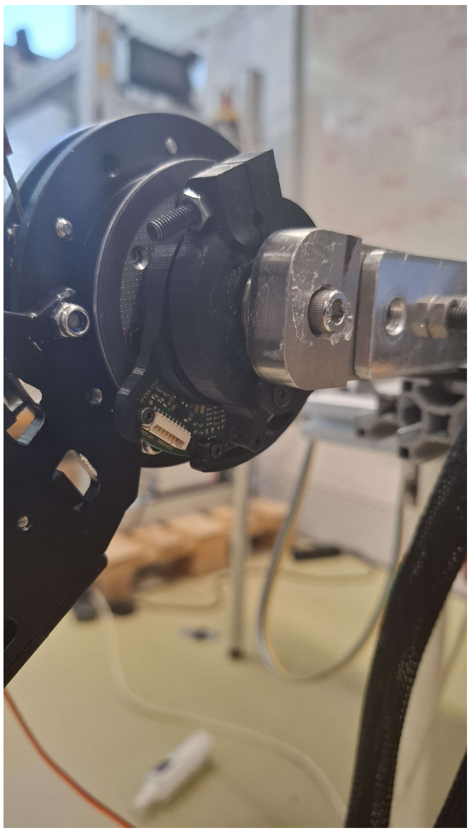


Figure 4.5: Deflection encoder

Sensor calibration

To calibrate the encoders, a protractor was attached to the exoskeleton. Angular positions were manually set using the protractor, and corresponding raw encoder values were read using the Arduino. The full calibration setup is shown in Figure 4.6
The angle 0° was exoskeleton's natural position with VSM and motor disengaged and 180° was upritght position of the exoskeleton. The collected data was then plotted, and a linear fit was applied to establish a mapping between raw sensor readings and physical angles. This calibration process enables accurate angle estimation during experiments.
The equation used later to calculate the angle of the shaft was determined as:

$$\theta_p = -0.324x + 181.75 \tag{4.1}$$

The final linear fit and raw data can be seen in Figure 4.7, the offset comes from manual adjustments after initial equation was implemented on Arduino.

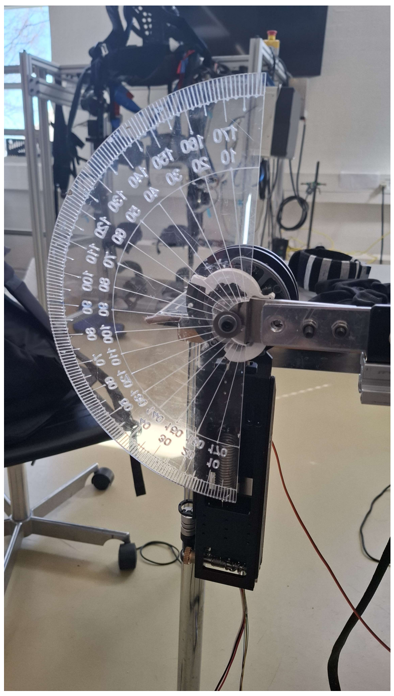


Figure 4.6: Encoder calibration setup

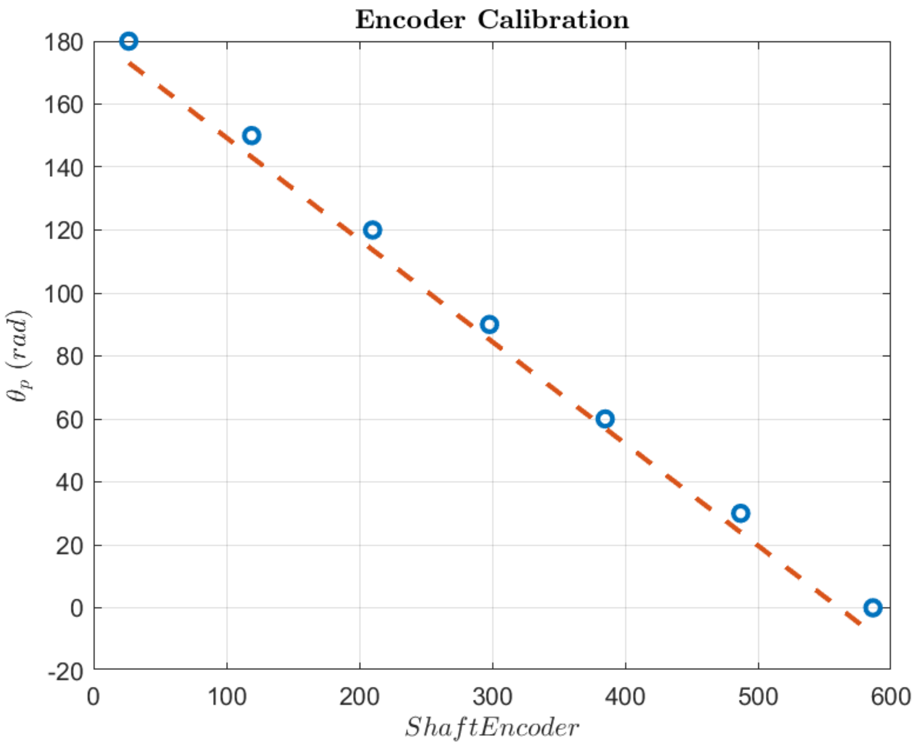


Figure 4.7: Calibration data from shaft encoder and linear fit

4.2 Modeling

This section will discuss modeling of the system and its implementation of low and high level control.

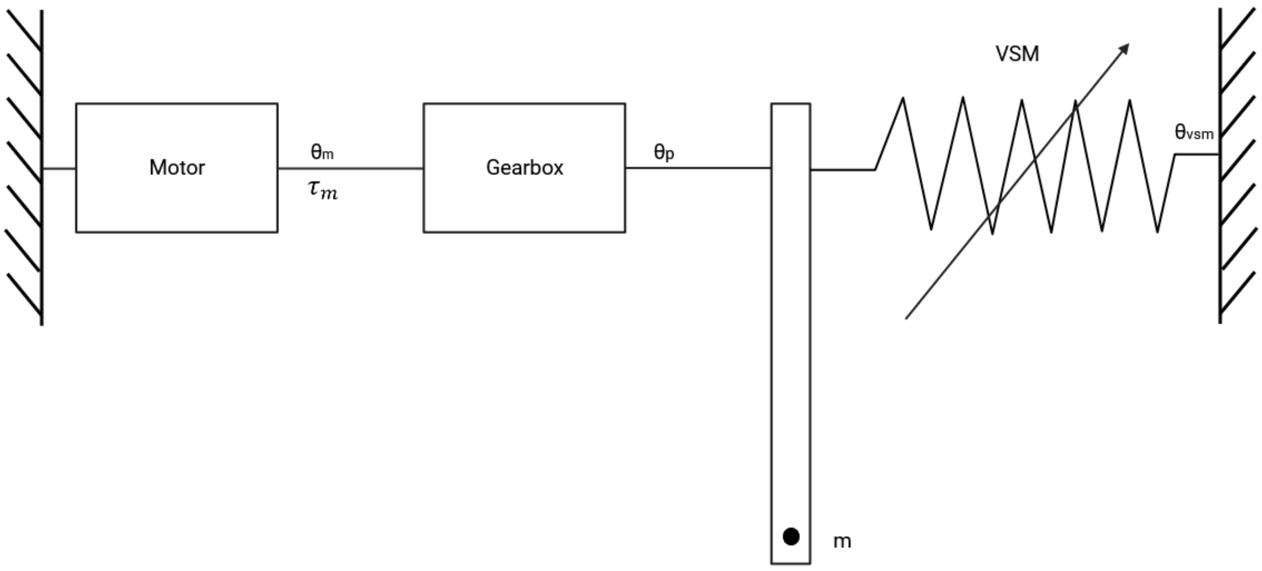


Figure 4.8: Free body diagram of the exoskeleton system

Based on free body diagram Figure 4.8 the system has been modeled as followed. All forces acting on the motor can written as:

$$\tau_m + \frac{\tau_{vsm}}{N} - \frac{\tau_p}{N} = \frac{J_p}{N}\ddot{\theta}_p + \frac{B}{N}\dot{\theta}_p \tag{4.2}$$

This can be rearranged to have pendulum acceleration as output of equation.

$$\frac{N\tau_m}{J_p} + \frac{\tau_{vsm}}{J_p} - \frac{\tau_p}{J_p} - \frac{B_p}{J_p}\dot{\theta}_p = \ddot{\theta}_p \tag{4.3}$$

The torque acting from pendulum and additional payload can be written as

$$\tau_p = l_{exo}m_{exo}g\sin(\theta_p) + l_{pay}m_{pay}gsin(\theta_p) \tag{4.4}$$

Which can be rearranged to:

$$\tau_p = (l_{exo}m_{exo} + l_{pay}m_{pay})g\sin(\theta_p) \tag{4.5}$$

The torque from VSM has been modeled based on previous experiment in which torque was measured against deflection of the VSM, Figure 4.9 shows data gathered in experiment and polynomial fit.

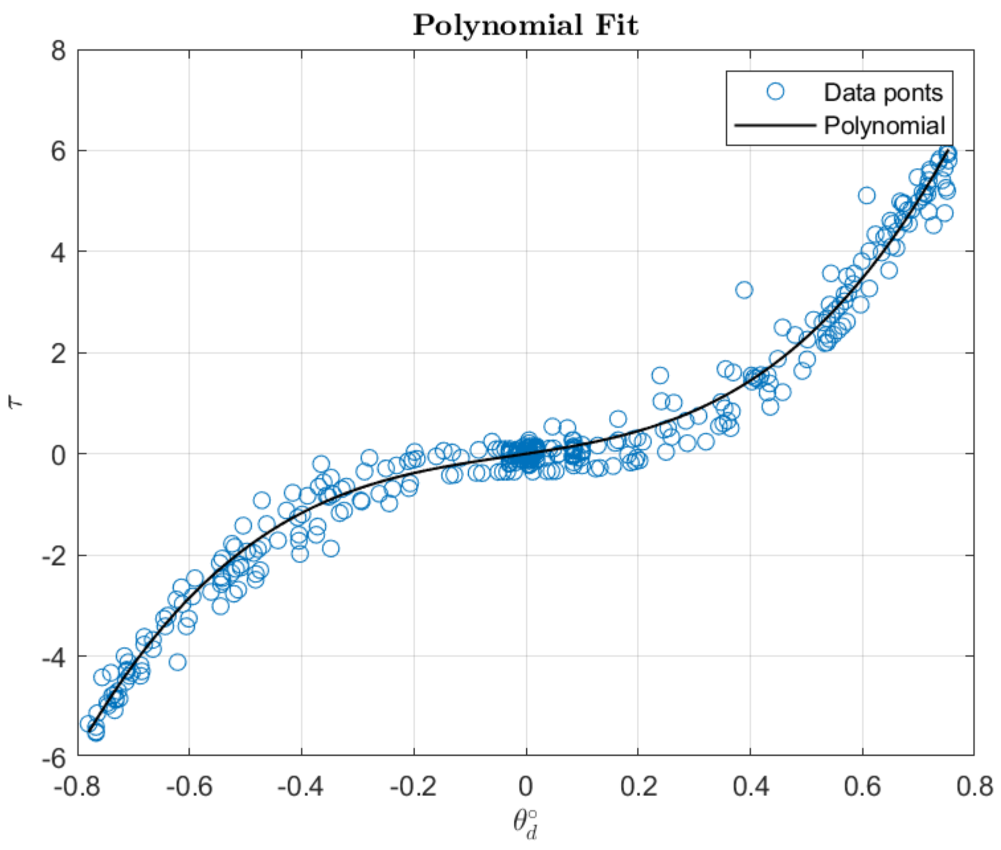


Figure 4.9: Polynimal fit of the data gathered on VSM

Final equation that was calulated by Matlab was. The constant value at the end was removed to make polynomial equal to 0 while deflection is equal 0.

$$\tau_{vsm}(\theta_d) = 9.9697\theta_d^3 + 0.8738\theta_d^2 + 1.6786\theta_d \tag{4.6}$$

Where:

$$\theta_d = \theta_{vsm} - \theta_p \tag{4.7}$$

Therefore final equation used to model the plant can be written as:

$$\ddot{\theta}_p = \frac{N\tau_m}{J_p} + \frac{\tau_{vsm}(\theta_d)}{J_p} - \frac{(l_{exo}m_{exo} + l_{pay}m_{pay})g\sin(\theta_p)}{J_p} - \frac{B_p}{J_p}\dot{\theta}_p \tag{4.8}$$

All variables used in Equation 4.8 are presented in Table 4.1.

Variable	**Description**
θ,$\dot{\theta}$,$\ddot{\theta}$	Pendulum's position, velocity and acceleration respectively
τ_m	Torque generated by motor
$\tau_{VSM}(\theta_d)$	Torque generated by VSM
τ_p	Tourqe generated by pendulum and additional payload
N	Final gear ratio
J_p	Pendulum's moment of inertia
B_p	Pendulum's damping ratio
l_{exo},l_{pay}	Exoskeleton's and payload's center of gravity respectively in regards to pivot point
m_{exo},m_{pay}	Exoskeleton's and payload's mass respectively in regards to pivot point

Table 4.1: Description of variables

Simulink model

Final equation Equation 4.8 was used in Plant of the model made in Simulink shown in Figure 4.10

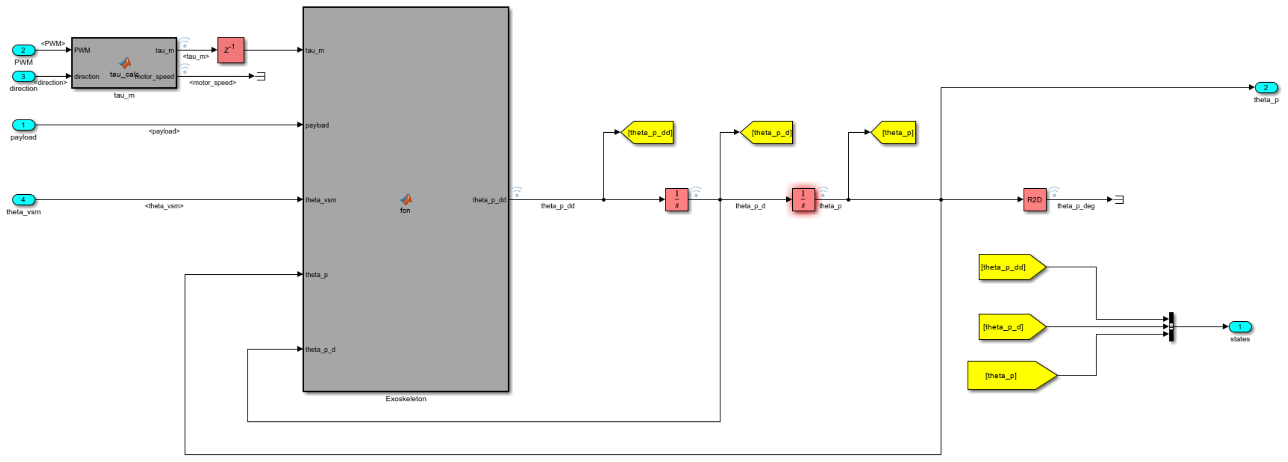


Figure 4.10: Simulink plant

The final gear ratio was calculated by multiplying the motor's gearbox and bevel's gear ratio, which were found to be 1:1221 and 1:2, respectively. Therefore, the final gear ratio is 1:2442. The mass of the exoskeleton can be easily obtained, the remaining values presented in Table 4.1. Variables Explained proved more challenging to obtain. Therefore, in order to approximate the moment of inertia of the exoskeleton, the exoskeleton itself was assumed to be a point mass, occupying one-third of the length of the exoskeleton. This point mass was attached to two other masses, which represented the motors. The new center of mass and moment of inertia were thus calculated to be 0.2401 m and 0.0318 kgm^2, respectively. The complete calculations can be found in Appendix B.
Moreover, it is not possible to predict the damping of the system. Therefore, this value has been empirically established on the basis of recorded data gathered from the shaft encoder while the VSM was locked at a position of 90°. Following a comprehensive evaluation of the available options, it

was determined that the optimal value for the system as a whole was $0.45 Nms/rad$. The empirical data and the results obtained from the Simulink model are presented in Figure 4.11 and Figure 4.12, respectively.

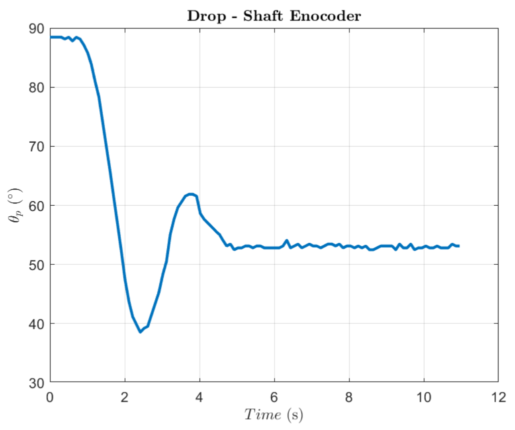


Figure 4.11: Recorded pendulum's position based on shaft encoder

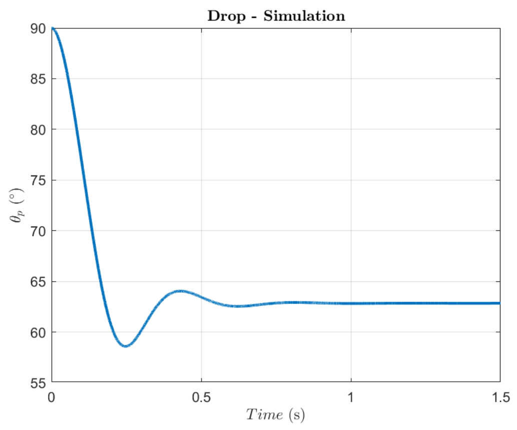


Figure 4.12: Recreated experiment in Simulink

The discrepancy between real and simulated data might be due to unaccounted friction in the system or wrongly predicted center of mass.

To simulate how control signals propagate through the system—from the Arduino, through the ESCON controller, and finally to the motor—a simplified motor model was implemented in Simulink. The objective of this model was to approximate the behavior of the motor under the influence of control inputs generated by a PID controller.

The first step was to convert the PID controller's output into signals that are compatible with the ESCON's control interface. In the real system, the Arduino outputs either a PWM signal or an analog voltage to command motor speed. In the simulation, the PID output is mapped to a PWM duty cycle using a scaling formula based on speed tourque constant $16.1 \frac{rad}{min\, mNm}$ and the maximum output voltage of 3.3V and a 12-bit PWM resolution. A snippet of the function block used for this conversion is shown below in Listing 4.1

```
function [PWM,direction] = PWM_calc(PID_out)
% PID signal to PWM convertion
speed = PID_out*16.1e3;

I = (speed/3.3)*4095; %12-bit resolution

% Calculate direction
if I<0
    direction = -1;
else
    direction = 1;
end

II = abs(I);

if II >= 4095 % Overflow prevention
    PWM = 4095;
else
    PWM = round(II); % Change to integer
end
```

Listing 4.1: PWM calculation from PID output

This simple conversion maps the PID output into a corresponding PWM signal, where:

$$speed = PID_{out} * 16.1e3 \tag{4.9}$$

$$PWM = (speed/3.3) * 4095 \tag{4.10}$$

Here, 4095 represents the maximum value for a 12-bit signal. The direction of rotation is determined based on the sign of the PID output: a positive value indicates forward motion, while a negative value indicates reverse. To prevent overflow, the PWM value is clamped at a maximum of 4095.

Next, to estimate the mechanical response of the motor, the PWM value is used to compute the motor speed. This simulated speed is then converted into torque using the again motor's speed/torque constant K_T. A simplified function block handling this computation is shown below in Listing 4.2

```
function [tau_m,motor_speed] = tau_calc(PWM, direction)
% PWM to speed convertion
motor_speed = PWM/4095*10600; % RPM

% Speed to torque convertion
tau_m = motor_speed/(16.1e3)*direction; % Nm
```

Listing 4.2: Torque calculation based on PWM input

The constants used in simulation are summarised in Table 4.2

Variable	Value
N	2442
J_p	0.0318 kgm^2
B_p	$0.45\frac{Nms}{rad}$
l_{exo},l_{pay}	0.2401m, 0.2m
m_{exo}	1.938kg
K_T	$16.1\frac{rad}{min\ mNm}$

Table 4.2: Variables used in model

5 Controller Design

The subsequent chapter will provide a discussion on the conceptual ideal design of the system. It is acknowledged that the preceding research and requirements have been given due consideration in the formulation of this study.

5.1 Low-level control

As discussed in chapter 2, several control strategies were considered for low-level control of the exoskeleton, including:

- Model Predictive Control (MPC)
- Nonlinear Model Predictive Control (NMPC)
- Impedance control (with neural network assistance)
- Classical PD or PID controllers
- Neural Network Scheduled PID

While MPC and NMPC offer strong performance in managing multivariable and nonlinear dynamics, they require accurate system modeling and substantial computational resources. These requirements exceed the capabilities of embedded systems such as the Arduino platform, especially under real-time constraints.

PID controllers, on the other hand, are computationally lightweight and well-suited for microcontroller-based platforms. However, a standard PID controller tuned at one operating point often performs poorly when system dynamics change—such as with varying joint angles or external loads. This limitation is especially problematic in systems incorporating Variable Stiffness Mechanisms (VSMs), where the nonlinearity is significant.

Impedance control, particularly when combined with neural network-based intention prediction as presented in [19], offers a biologically inspired approach to compliant control. However, even in this approach, the computational overhead of real-time neural estimation can be prohibitive for resource-limited hardware unless heavily optimized.

Given these constraints and goals, this project will implement a PID controller with a neural network-based gain scheduler for low-level control. This hybrid approach combines the real-time feasibility of PID with adaptability to nonlinear system behavior via intelligent gain tuning.

This approach enables flexible and responsive control across different operating regimes without requiring complex physical modeling or computationally expensive real-time optimization.

5.2 High-level control

For the purposes of this design, only sensing modalities and prediction strategies discussed in section 2.3 are considered. These include:

- Inertial Measurement Units (IMUs)
- Force Myography (FMG) armbands
- Electromyography (EMG) armbands
- Prediction of assistive tourque based on joint position history

Each of these methods presents distinct advantages and trade-offs:

IMUs are relatively easy to implement and provide reliable measurements of joint orientation and acceleration. However, they offer limited information about the external load being carried or the user's muscular effort, which reduces their effectiveness in predicting user intention under varying load conditions.

FMG armbands measure pressure changes in the forearm caused by muscle contractions. They are non-invasive, can be worn over clothing, and generally offer a higher signal-to-noise ratio than EMG. FMG can estimate both the user's intention and the approximate load on the limb. However, they may be sensitive to external pressure, such as contact with the exoskeleton frame, which can interfere with signal quality.

EMG armbands detect electrical activity from muscle activation and can offer high-fidelity intention prediction. Yet, they require direct skin contact, careful sensor placement, and consistent skin conditions, making them less suitable for industrial or long-term use.

Joint position history-based prediction, often used in conjunction with machine learning models, can be effective for tracking movement trends. However, this approach struggles with intention classification, especially in static or load-bearing tasks, and does not provide insights into external force or user effort.

Given the application context—an upper-limb exoskeleton designed for industrial use—FMG armbands offer the best balance of practicality and data richness. Their ease of deployment, ability to be worn over clothing, and relevance for estimating muscular activity and load make them an ideal candidate. To complement the FMG, an IMU placed on the upper arm will be used to measure joint angles in real time, enabling closed-loop control and tracking.

To interpret these continuous, time-dependent signals, a Long Short-Term Memory (LSTM) Neural Network is selected. LSTMs are particularly effective for time-series data like FMG and IMU signals, as they can capture long-range dependencies and temporal dynamics essential for user intention prediction.

5.3 Final Design

Based on the research conducted and trade-offs discussed in Chapter 2, the control system for the variable stiffness exoskeleton is structured into two hierarchical layers, each addressing distinct control challenges:

Low-Level Control

The low-level controller is tasked with accurate joint tracking and stabilization despite the nonlinear behavior introduced by the Variable Stiffness Mechanism (VSM). A conventional PID controller is employed due to its computational efficiency, which makes it suitable for implementation on an embedded platform such as Arduino. However, to compensate for the inherent limitations of fixed-gain PID in nonlinear systems, a neural network-based gain scheduler is integrated. This scheduler adjusts the PID parameters dynamically based on the estimated load and target joint angle, enhancing robustness and adaptability across varying conditions.

High-Level Control

The high-level controller is responsible for user intention recognition and adaptive task mode switching. For this purpose, a Force Myography (FMG) band is used in conjunction with an IMU sensor placed on the arm. FMG offers a non-invasive and user-friendly sensing modality that can detect muscle pressure changes even over clothing, making it well-suited for industrial environments. The IMU provides complementary kinematic data, such as joint angles and movement patterns. An LSTM neural network is utilized to analyze temporal patterns in these continuous signals, classifying the user's intent—such as movement, load application, or holding position—with high temporal resolution and adaptability.

Final overview of the system can be seen in Figure 5.1

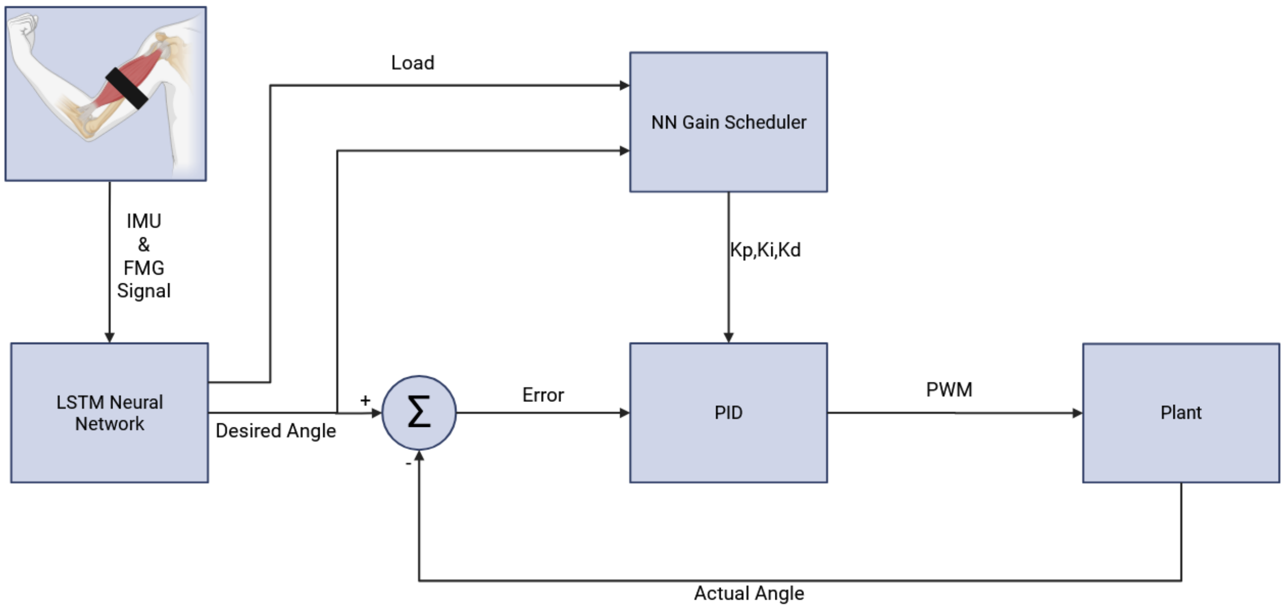


Figure 5.1: Overview of the system

6 Implementation

Based on design in chapter 5 and delimitation in chapter 3 this chapter will go in depth into implementation.

6.1 Low-Level Control

This section describes the implementation of the PID controller, the data acquisition process, and the training procedure for the neural network-based scheduling system, which is later used during testing.

Controller

The proposed control strategy for the exoskeleton combines a Neural Network PID (NN PID) controller with a torque observer. The observer estimates the torque currently acting on the system, which is then subtracted from the PID controller's output. This counteracting torque helps to stabilize the exoskeleton by compensating for natural gravitational movement.
The torque observer is based on the rearranged system dynamics shown in Equation 4.8.

$$\tau_m = \frac{J_p}{N}\ddot{\theta}_p + \frac{B_p}{N}\dot{\theta}_p - \frac{\tau_{vsm}(\theta_d)}{N} + \frac{(l_{exo}m_{exo} + l_{pay}m_{pay})g\sin(\theta_p)}{N} \tag{6.1}$$

This estimated torque is subtracted from the PID output, which is calculated based on the position error. The motivation behind this approach is to address a limitation in simple PID position control: when the motor does not actively generate torque, the pendulum can swing freely by up to 30° even with the VSM locked at 90°. This unopposed motion causes instability, overshooting, and rapid rise times when the desired angle is below the locked VSM angle. The observer compensates for this by providing an opposing torque input.
The complete control system implemented in Simulink, including both the observer and the initial PID controller, is shown in Figure 6.1. This PID controller is later replaced with the NNS PID for evaluation.

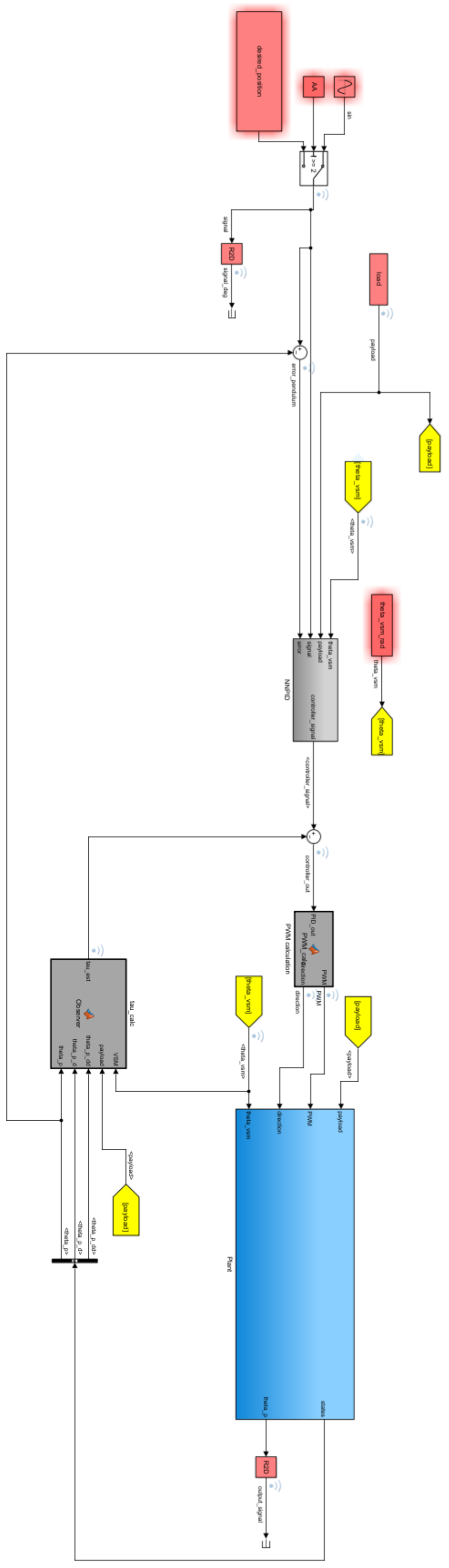


Figure 6.1: Control system in Simulink

Although the current setup requires manual input of the load, this step will eventually be replaced by a high-level control module capable of predicting the load automatically.

Optimiser

Since the neural network requires large quantities of training data, the PID controller must be tuned for various scenarios, including different VSM lock positions, target angles, and payloads. This creates a large set of combinations, making manual tuning impractical. To automate the tuning process, an autotuning system was developed. The optimiser utilitised Particle Swarm Optimization (PSO) to tune the PID gains (K_P, K_I, K_D) for the shoulder exoskeleton controller. PSO is a population-based global optimizer that searches for optimal parameters by simulating social behavior among particles. In this setup, 5 particles were initialized randomly within defined bounds $[0,100]$ for each parameter. The simulation was run with guessed values and based on cost function Equation 6.2 the performance was evaluated. The optimizer was allowed to run for up to 3 iterations per test case. To refine the results, a hybrid approach was used by integrating MATLAB's `fmincon` which is a local minima search method.

$$cost = \int Error^2 \tag{6.2}$$

Error is firstly squared in order to prevent negative area. Additionally to speed up optimisatiom in case of small cost, lower than 1 it is set up as 0
Once all predefined scenarios are processed, the optimised PID gains are saved to a CSV file for later use in neural network training. Each scenario was defined by a combination of three theoretical input variables:

1. **VSM lock position** — the position at which the VSM was engaged. This value represents reading that would be obtained from a shaft encoder.
2. **VSM deflection** - deflection of the VSM. Based on that value and lock position the target value was defined.

$$\theta_{target} = \theta_{VSM} + \theta_{def} \tag{6.3}$$

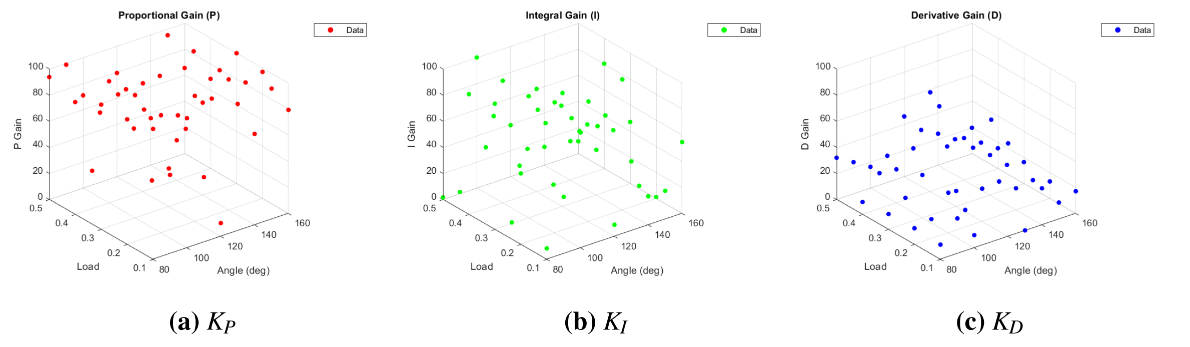
3. **Load** - load applied to the pendulum

The combination of these variables forms the the training dataset, with all values used during simulation shown in Table 6.1.

Variable	**Values**
VSM lock position	60°,90°,120°
VSM deflection	-40°,-30°,-20°,-10°,0°,10°,20°,30°,40°
Load	0.1 kg, 0.2 kg, 0.3 kg, 0.4 kg, 0.5 kg

Table 6.1: Variables used in training

Althighether optimiser gathered 135 data points, whchich can be seen in Appendix C. Furthermore, visualised data can be seen Figure 6.2, where X and Y axis are inputs to the optimiser and on Z axis is the output K_P, K_I, K_D respectively.



(a) K_P **(b)** K_I **(c)** K_D

Figure 6.2: Output from optimiser for VSM locked at 120°

Neural Network

To train the neural network effectively for PID parameter prediction, a prevoiusly gathered dataset consisting of three input features—VSM, Desired Position, and Load—and their corresponding K_p, K_i, and K_d outputs was collected and imported into the system. Figure 6.3 shows how does inputs are being processed in order to obtained PID outputs.

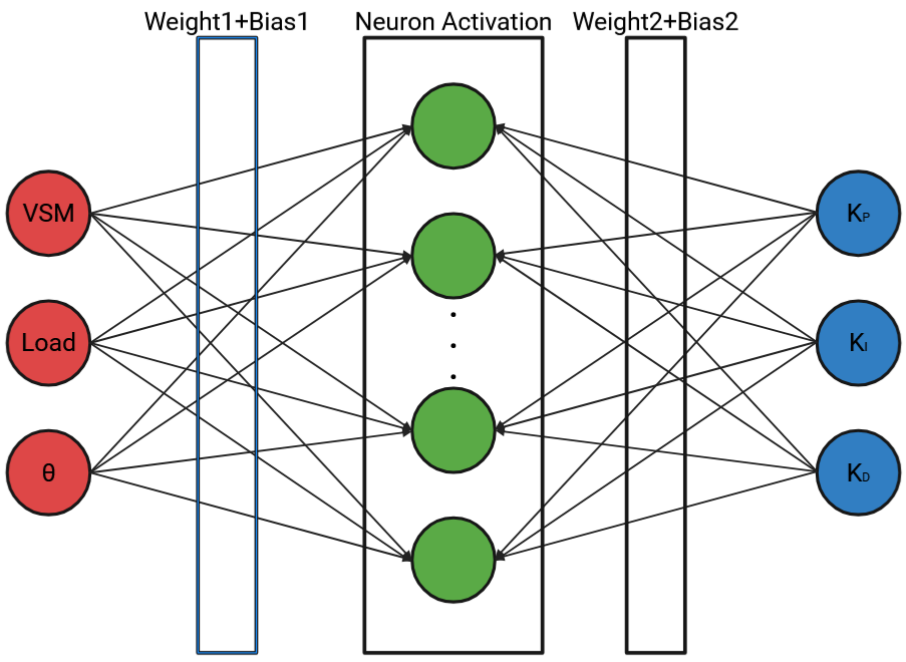


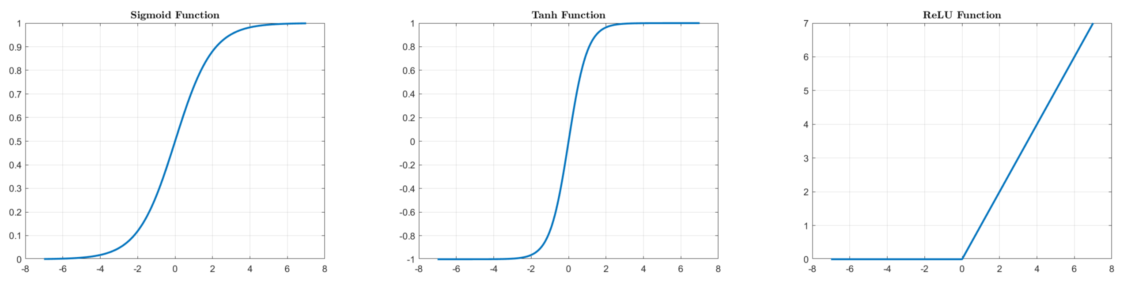
Figure 6.3: Diagram of Neural Network

Given the nature of this prediction task and the lack of prior benchmarks for comparison, it was decided to explore multiple training strategies to evaluate the network's performance.

To accomplish this, three different optimization algorithms were implemented and compared:

- **Gradient Descent:** A basic optimization method where the weights are iteratively adjusted in the opposite direction of the gradient of the loss function. It is simple and widely used but may converge slowly or get trapped in local minima.
- **Adam:** An improvement over standard gradient descent that computes adaptive learning rates for each parameter. It uses running averages of both the gradients and their squares to provide efficient and stable updates.
- **Particle Swarm Optimization (PSO):** A population-based metaheuristic that simulates social behavior observed in flocks of birds or fish schools. Each ”particle” (a potential solution) adjusts its position based on its own experience and the experience of its neighbors, optimizing the network weights based on the global best solution.

Each of these optimizers was tested under the same network architecture: a single hidden layer, taking in 3 input features and producing 3 outputs. The number of neurons has been empirically determined to be 20. This should be sufficient to produce varied output, and the Arduino will be capable of rapid calculation. The output activation function was deliberately set to none (linear), to ensure direct regression output of the predicted PID values. Moreover hidden layer activation function was also varied across three common choices to assess their impact on performance. The functions are Sigmoid, Tanh, and ReLU (Rectified Linear Unit) all of them can be seen in Figure 6.4a, Figure 6.4b, and Figure 6.4c



(a) Sigmoid **(b)** Tanh **(c)** ReLu

Figure 6.4: Neuron activation functions

The weights and biases of the neural network were initialized randomly. Before training the data is being split into Training and Test data. The training set accounts for 80% of the total dataset, with the remaining 20% designated for testing purposes. During training, the network computed mean squared error (MSE) at each epoch for each output K_p, K_i, K_d, which was tracked and plotted to visualize learning progress over time. Similarly the test data is used to evaluate performance of the Network against unknown data. The training was conducted for 1000 epochs with a learning rate of 0.02. Figure 6.5 shows that mean squared error of training and testing data converges.

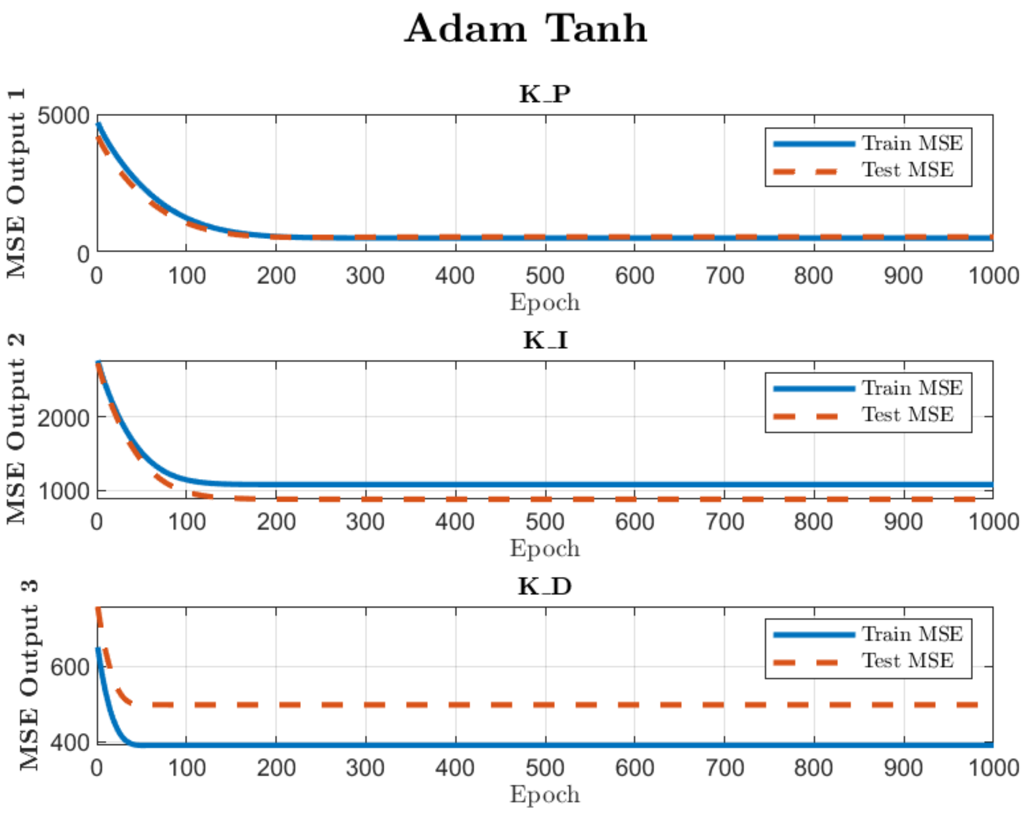


Figure 6.5: Example of the out put from Neural network training

The trained matrices for weights and biases were later saved and used in Simulation, final overview of the low-level control implemented in Simulink can be seen in Figure 6.6

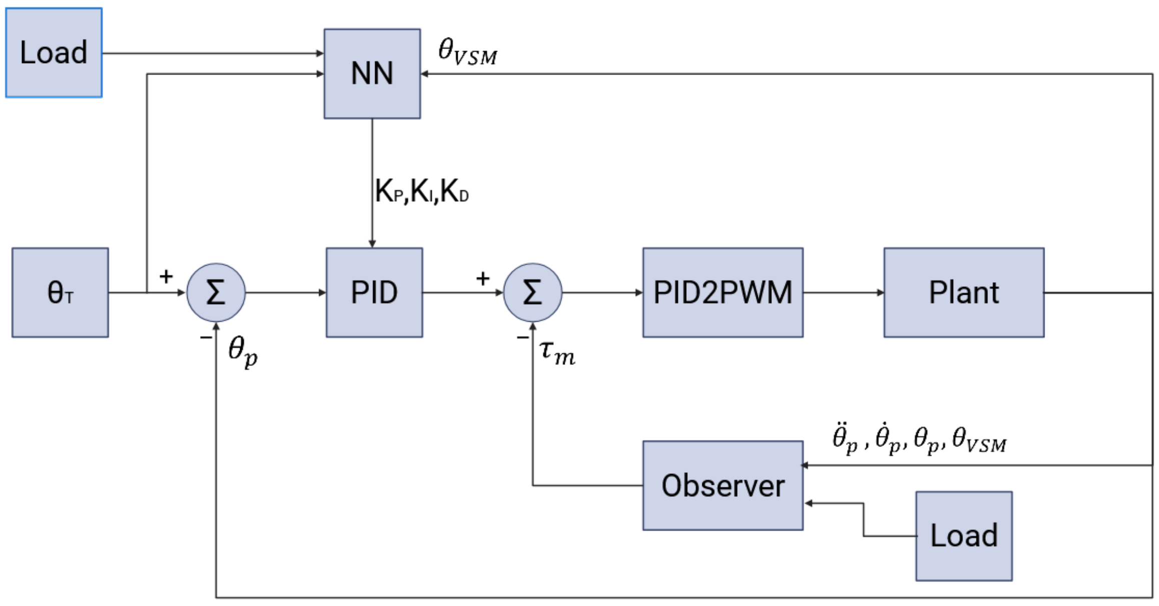


Figure 6.6: Final Low-level control diagram implemented in Simulink

Hardware Implementation

As previously mentioned as in chapter 4 due to back drivability issues of prototype exoskeleton, the actual hardware implementation would be done on different elbow exoskeleton. Although the exoskeleton is equipped with VSM the configuration is different, as it can be seen in Figure 6.7. The motor acts on pendulum through VSM not directly like in shoulder exoskeleton.

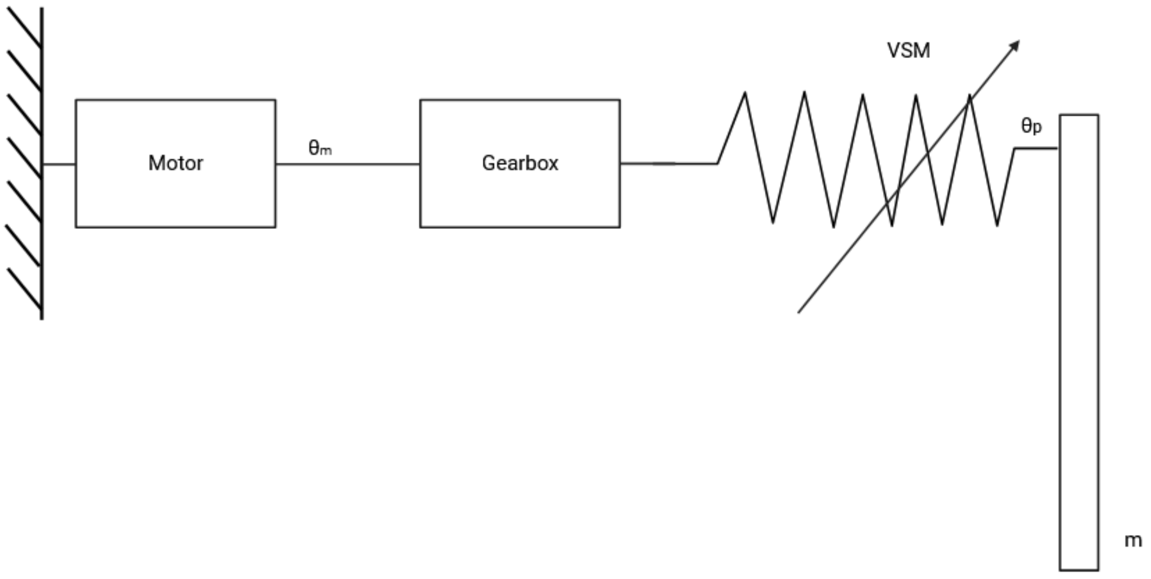


Figure 6.7: Free body diagram of the elbow joint exoskeleton

As in this setup VSM can not be locked neural network implemented to the will have only two inputs load on pendulum and desired position. Furthermore all data has to be gathered by manually tuning PID values. Figure 6.8 shows whole setup of the exoskeleton, (1) Maxon motor, (2) Arduino Due and (3) Escon controller. To ensure safety and prevent exoskeleton from damaging the whole setup (4) safety switch was added to setup.

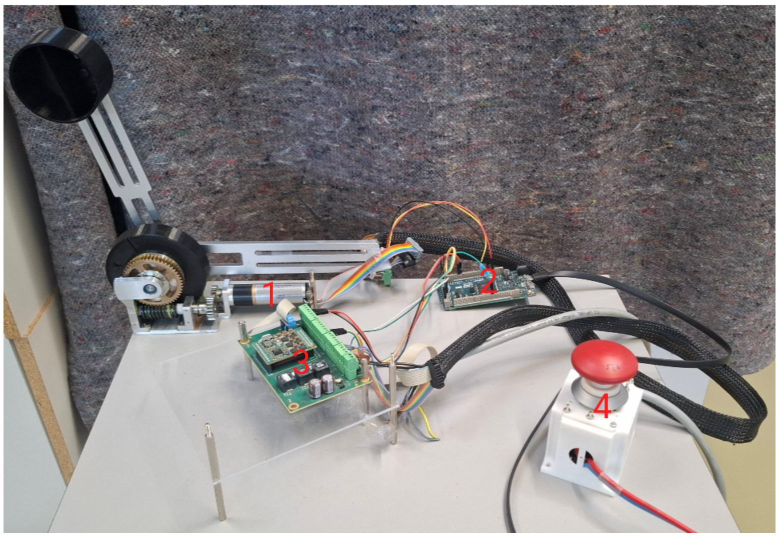


Figure 6.8: Elbow exoskeleton setup

Furthermore, the angle 0° was defined in full elbow's extension and 180° in full flexion, as shown in Figure 6.9.

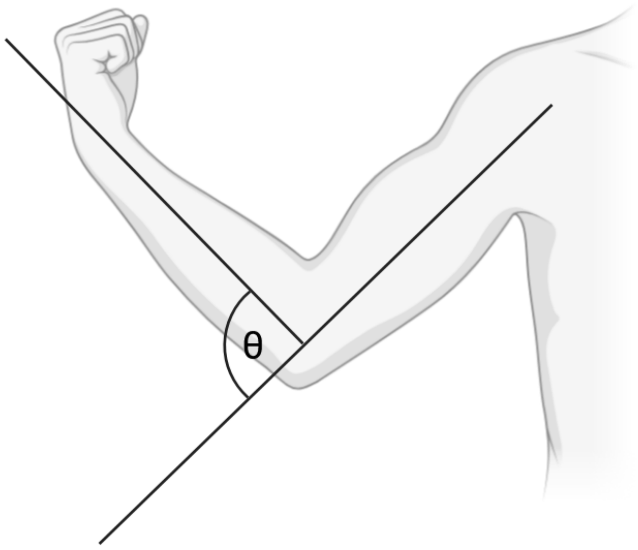


Figure 6.9: Angle of the elbow joint of the exoskeleton

The variables used during tuning are summarised in Table 6.2.

Variable	**Values**
Target angle	30°,45°,60°,75°,90°
Load	0.06 kg, 0.075 kg, 0.1 kg, 0.16 kg, 0.175 kg, 0.2kg

Table 6.2: Variables used in training of elbow exoskeleton

The Ziegler–Nichols tuning method was implemented in order to obtain approximation of the most optimal K_p, K_i, K_d. This method requires increasing K_p value while keeping the rest as 0 until the system start to oscillate, then using this K_p values and oscillation period by equations shown in Table 6.3 approximations of optimal values can estimated. The equations used were no overshoot.

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.8\bar{3}T_u$	–	$\frac{0.54K_u}{T_u}$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.10K_uT_u$
Classic PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$\frac{1.2K_u}{T_u}$	$0.075K_uT_u$
Pessen Integral Rule	$0.7K_u$	$0.4T_u$	$0.15T_u$	$\frac{1.75K_u}{T_u}$	$0.105K_uT_u$
Some Overshoot	$0.33K_u$	$0.5T_u$	$0.33T_u$	$\frac{0.66K_u}{T_u}$	$0.11K_uT_u$
No Overshoot	$0.2K_u$	$0.5T_u$	$0.33T_u$	$\frac{0.4K_u}{T_u}$	$0.066K_uT_u$

Table 6.3: PID Tuning Parameters Based on Ziegler–Nichols Method Variants

As those values were still making system osculate to further improve accuracy of the system the values the final values were obtain by keeping K_p same as obtained and dividing K_i, K_d by some value. A trial and error method used to obtain proper K_i, K_d values. Data obtained through this processes can be seen in Appendix D. This data was used as input for Neural Network and trained as described before.

In order to accelerate the calculation of the gains, the BasicLinearAlgebra[30] library for Arduino was implemented. This method was found to be significantly more efficient in terms of calculation speed than the previously utilised for loop approach.

6.2 High-level controll

For high-level control, the primary goal is to accurately predict the load a user is carrying, as this information is essential for the observer used in low-level control. Therefore, the current implementation of high-level control focuses on estimating the carried load based on biosignal data.

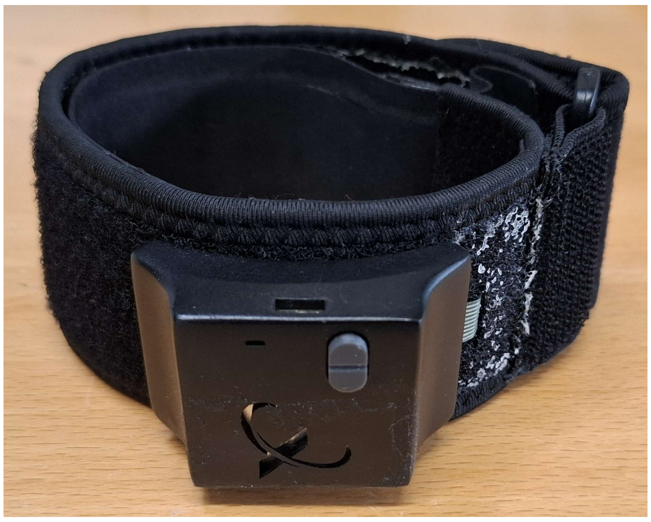


Figure 6.10: BioX FMG arm band

The wearable sensor used for this purpose is the BIOX FMG (Force Myography) armband, shown in Figure 6.10. This armband is equipped with eight force-sensitive resistor (FSR) sensors positioned evenly around the band, as well as an inertial measurement unit (IMU) and a gravimeter. The FSR sensors detect pressure through changes in resistance, which is converted into a voltage signal. These signals are transmitted via Bluetooth to a computer, where a MATLAB script receives and logs the data, whole setup can Figure 6.11.

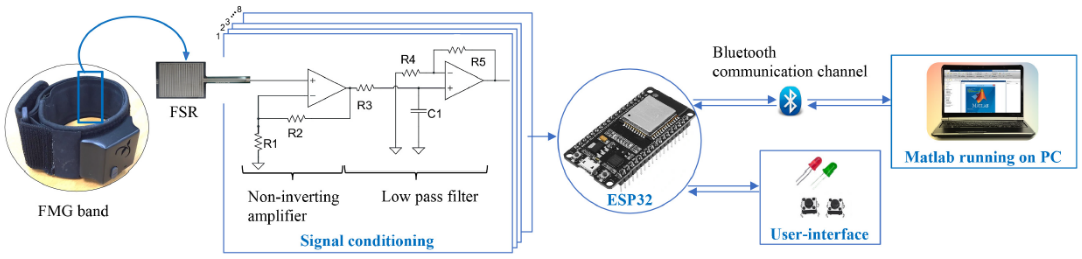


Figure 6.11: Overview of the setup for BioX arm band, taken from [31]

During initial connection and testing, it was observed that the FSR output varied slightly, even for the same subject without repositioning the armband. This variation is likely due to the internal auto-calibration mechanism of the FSRs, which adjusts the resistance reference based on the initial applied force, amplifying the signal to improve resolution. To ensure consistency, this calibration was disabled, reducing the signal range to 0–1. Additionally, care was taken to wear the armband with similar tightness across all tests. Example of the signal received form the arm band can be seen in Figure 6.12.

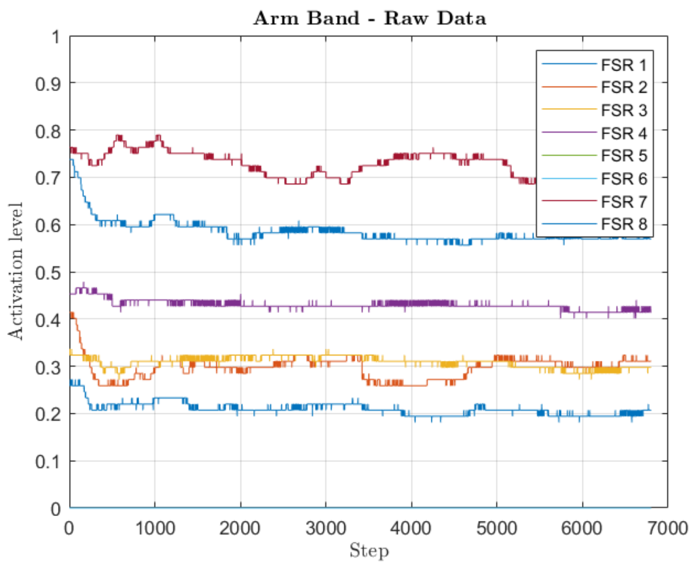


Figure 6.12: Example of data gathered from the band

Data gathering

To collect training data, the armband was worn continuously without removal during the recording session. The placement of the band was consistently positioned on the right arm in the middle of the biceps, with sensor number eight situated on the upper portion of the biceps, facing outwards from the subject.

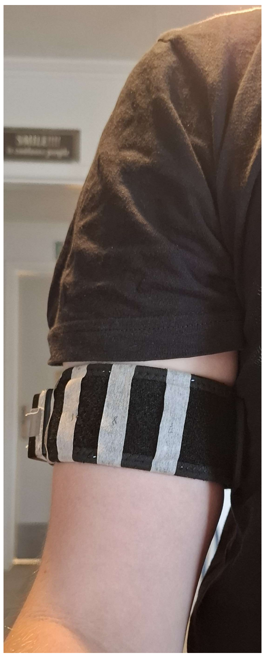


Figure 6.13: Arm band placed on the subject

Subjects were asked to hold various weights with their elbow flexed at a 90° angle for 10 seconds per sample, as shown in Figure 6.14. The following weights were used in the experiments:

- 0 kg
- 1 kg
- 2 kg
- 3 kg
- 4 kg
- 5 kg

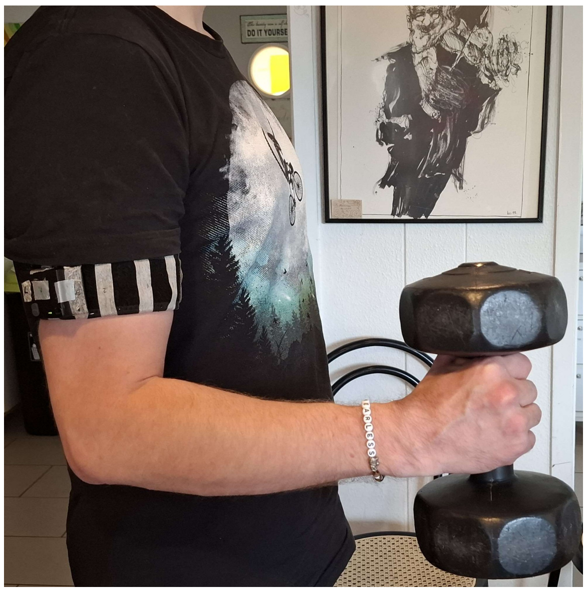


Figure 6.14: Subject during data gathering while holding the 5kg weight

Neural Network Setup

The neural network for load prediction was implemented in MATLAB using a Long Short-Term Memory (LSTM) architecture, as previously introduced in 5. This model is designed to estimate the weight carried by the user based solely on force myography (FMG) signals. It is set up as a sequence-to-one regression model, where an input sequence of sensor readings is used to predict a single scalar output representing the carried load in kilograms.

The input consists of eight FSR signals, each of which is normalized using z-score normalization to account for inter-subject variability. Output labels correspond to the known weights held by the subject. Raw data is stored in CSV format, with each file representing a specific recording session. The data is segmented into fixed-length windows of 5 seconds (100 time steps) and formatted into input-output pairs. These pairs are then shuffled and split into training and test datasets, with 80% of the data used for training.

The structure of the neural network is visualized in Figure 6.15

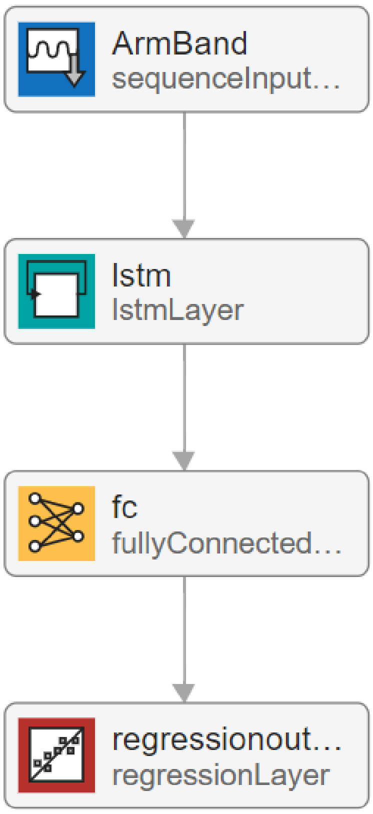
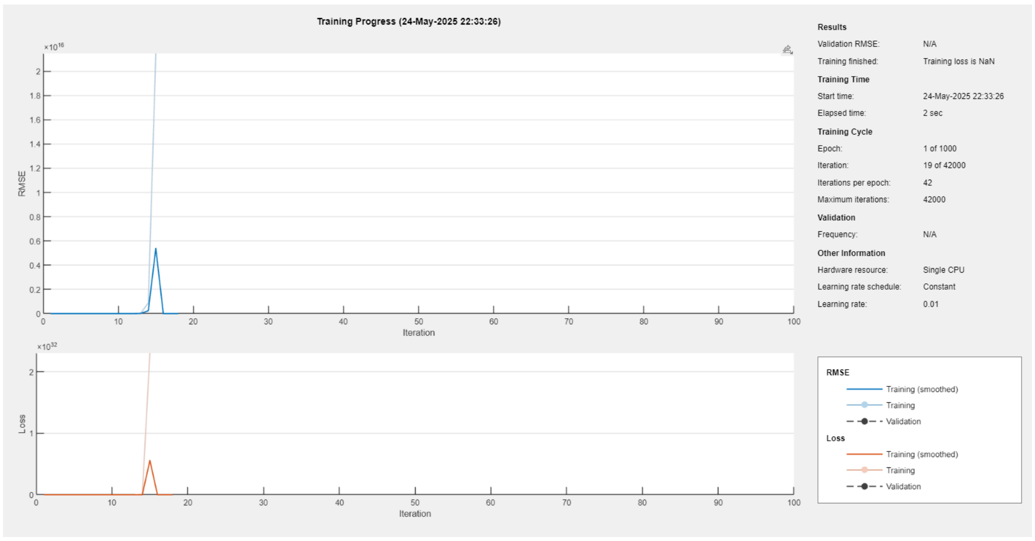


Figure 6.15: LSTM Network visualised in Deep Network Designer

Network Architecture:

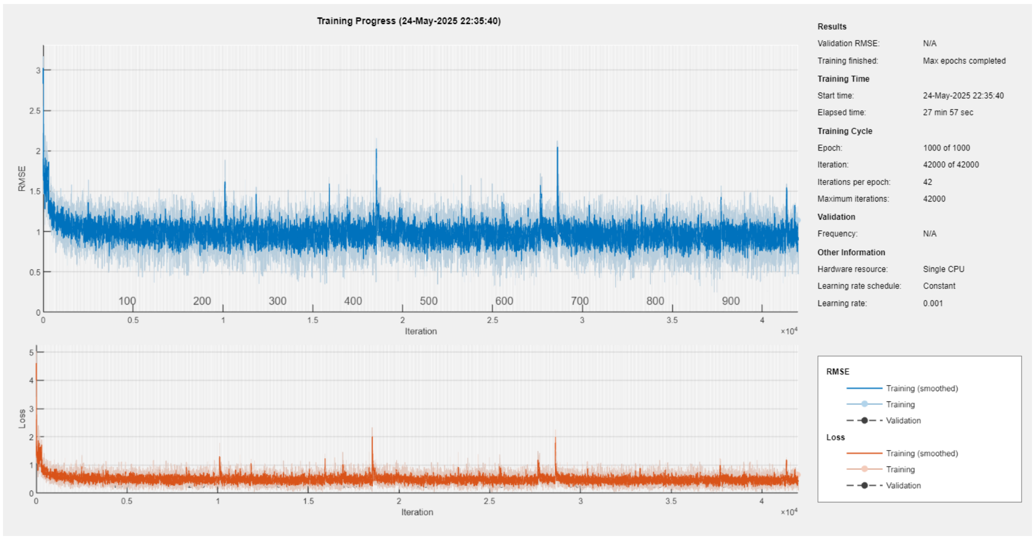
- sequenceInputLayer: Accepts 8-dimensional FSR input vectors
- lstmLayer: Processes temporal patterns across the time series
- fullyConnectedLayer: Reduces the output to a single scalar value
- regressionLayer: Computes loss between predicted and actual load values using Mean Squared Error (MSE)

The network was trained using two commonly used optimizers: Adam and Stochastic Gradient Descent (SGD), both available in MATLAB's training functions as discussed in 6.1. Each model was trained for 1000 epochs with a mini-batch size of 32. The results of the training process are shown Figure 6.16 and Figure 6.17.



```
Results
Validation RMSE:          N/A
Training finished:        Training loss is NaN
Training Time
Start time:               24-May-2025 22:33:26
Elapsed time:             2 sec
Training Cycle
Epoch:                    1 of 1000
Iteration:                19 of 42000
Iterations per epoch:     42
Maximum iterations:       42000
Validation
Frequency:                N/A
Other Information
Hardware resource:        Single CPU
Learning rate schedule:   Constant
Learning rate:            0.01
```

Figure 6.16: Training progress for SGD



```
Results
Validation RMSE:          N/A
Training finished:        Max epochs completed
Training Time
Start time:               24-May-2025 22:35:40
Elapsed time:             27 min 57 sec
Training Cycle
Epoch:                    1000 of 1000
Iteration:                42000 of 42000
Iterations per epoch:     42
Maximum iterations:       42000
Validation
Frequency:                N/A
Other Information
Hardware resource:        Single CPU
Learning rate schedule:   Constant
Learning rate:            0.001
```

Figure 6.17: Training progress for Adam

The blue line (RMSE) represents the Root Mean Squared Error, which measures how close the network's predictions are to the actual known data points. A lower RMSE indicates more accurate predictions. This metric reflects the overall performance of the network. The orange line (Loss) shows the value being minimized during training. It drives the learning process by guiding how the network updates its internal parameters.
As seen SGD network fails at before 20th epoch and attempts were made to fix that but probably due to some signals from FSR being 0 it kept failing. Therefore only Adam was used in further testing.
The trained network is saved for later use in a modified version of the MATLAB script that was originally used to gather the data. This modified version includes continuous data streaming and segmentation into windows of size 8×100 for real-time prediction.

7 Testing

In order to check and evaluate systems performance a number of tests have been designed. All tests has been designed to check if requirements in chapter 3 have been fullfiled. The test were divided into two categories Low- and High-level control. Subsequent sections describe the testing scenario and results.

7.1 Low-Level Controller

This section outlines the testing scenarios and presents the results of the low-level controller performance, both in Simulink simulation and on the physical hardware.

Simulation

To evaluate the Neural Network PID (NNPID) controller in simulation, the VSM was tested under conditions different from those used during training. Specifically, the VSM's locked position was set to $100°$, and two target deflections of $+35°$ and $-35°$ were tested, with an attached load of 350 g. These test parameters were selected to assess the controller's adaptability to previously unseen conditions.

Additionally, a trajectory tracking task was implemented, following a sinusoidal trajectory centered at $100°$ with a $\pm35°$ amplitude at a frequency of 0.5 Hz.

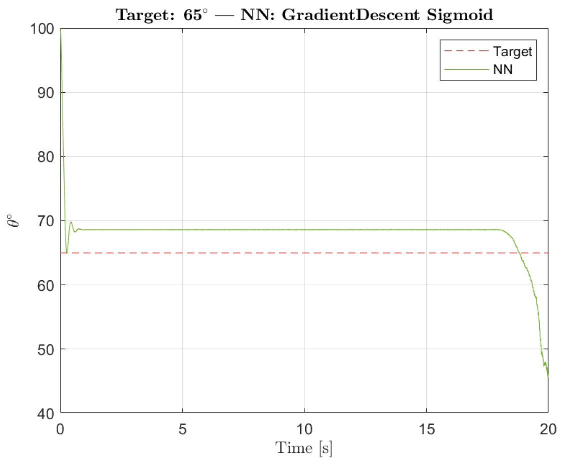
Results

The results have been divided into step response and sin wave response. Data was obtained from Simulink by running simulation and each time changing Neural Network.

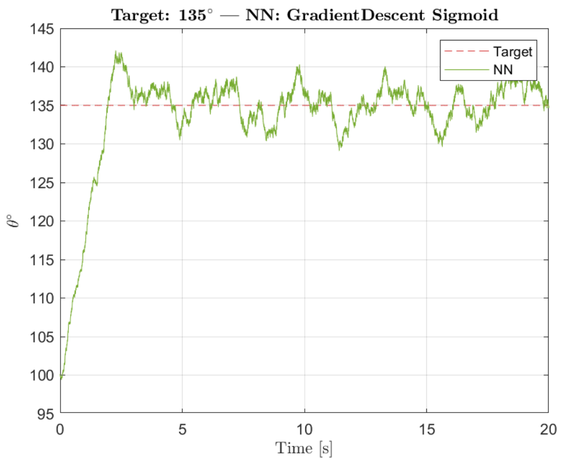
Step response

The following section will present the results for the step response of the system.

Gradient Descent

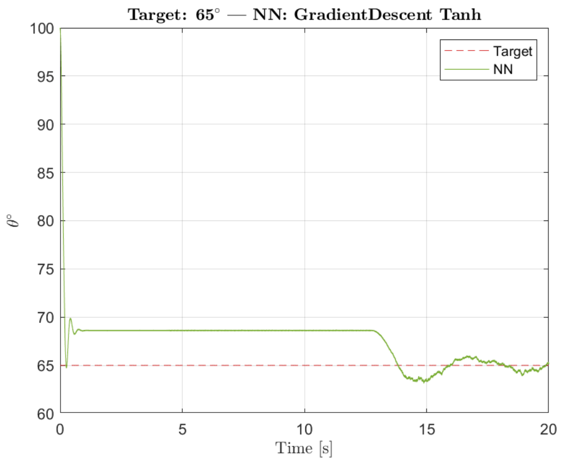


(a) Target 65° with load 0.35kg

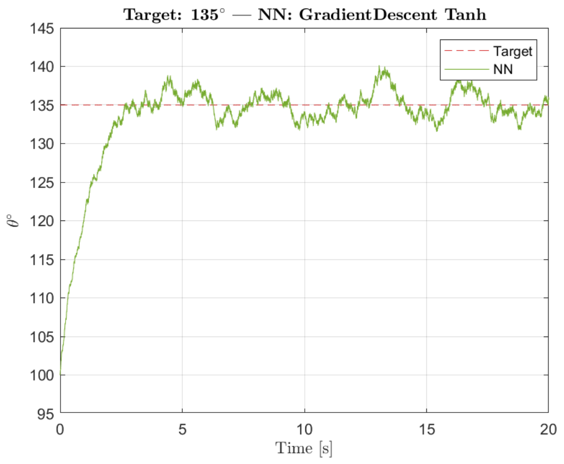


(b) Target 135° with load 0.35kg

Figure 7.1: Results for Gradient Descent with Sigmoid activation function

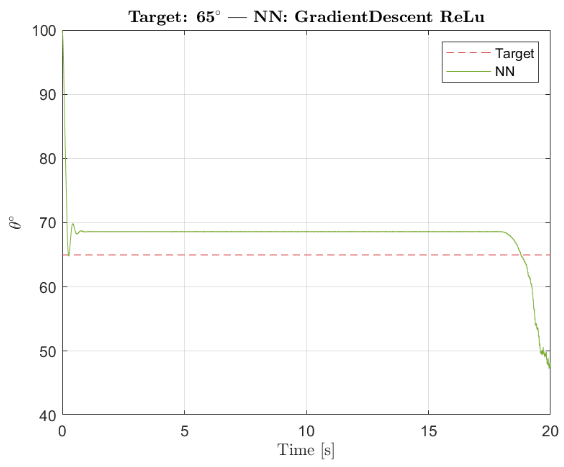


(a) Target 65° with load 0.35kg

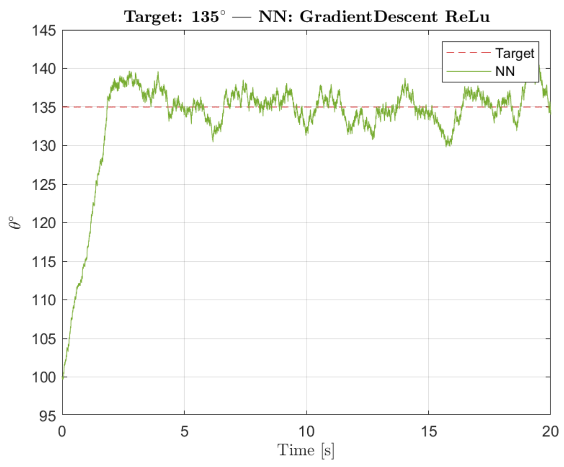


(b) Target 135° with load 0.35kg

Figure 7.2: Results for Gradient Descent with Tanh activation function



(a) Target 65° with load 0.35kg

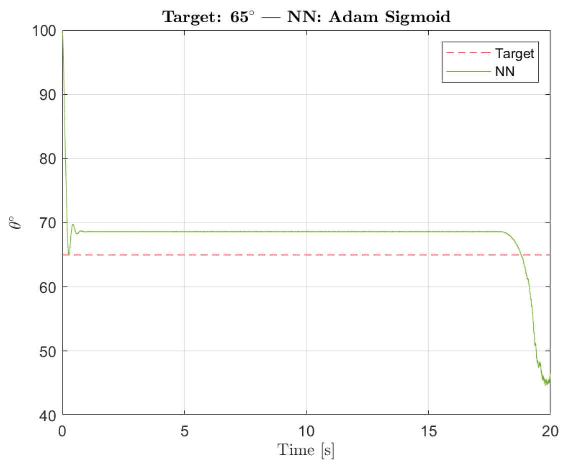


(b) Target 135° with load 0.35kg

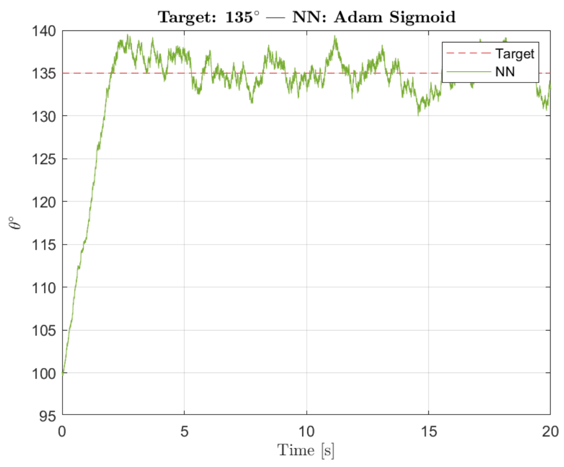
Figure 7.3: Results for Gradient Descent with ReLu activation function

As can be seen in Figure 7.1a, Figure 7.2a and Figure 7.3a, in all cases of target angle of 65° the exoskeleton reaches steady state error of 69° in smooth manner. Additionally cases in with Sigmoid and ReLu activation function suddenly drop below target and only Tanh function after drop reaches target. Moreover none of the cases shown in Figure 7.1b, Figure 7.2b and Figure 7.3b outperform other. All of them show high oscillation of the exoskeleton of around 5° with no sign of settling. Additionally smaller oscillations between timestep can be seen making the signal uneven.

Adam

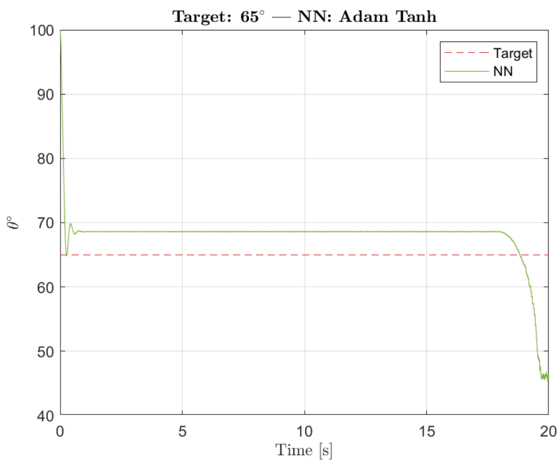


(a) Target 65° with load 0.35kg

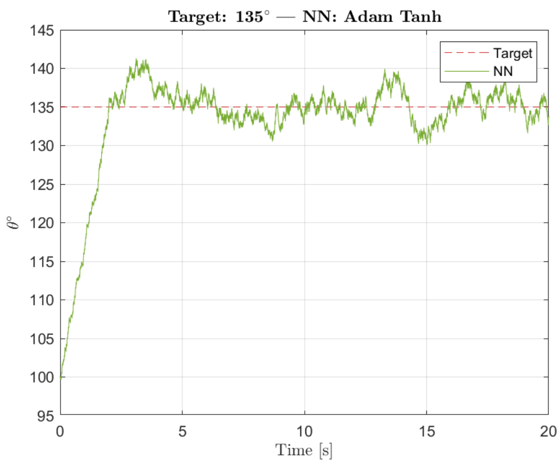


(b) Target 135° with load 0.35kg

Figure 7.4: Results for Adam with Sigmoid activation function

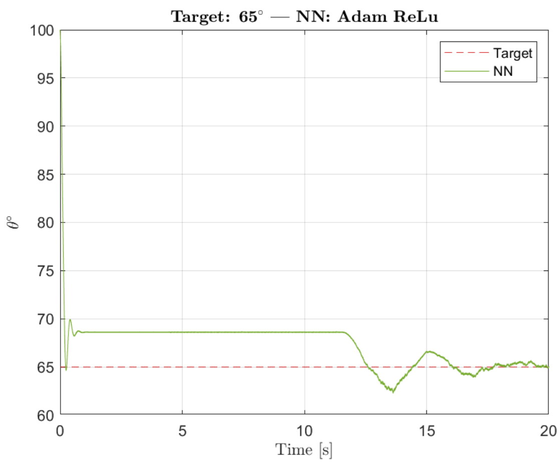


(a) Target 65° with load 0.35kg

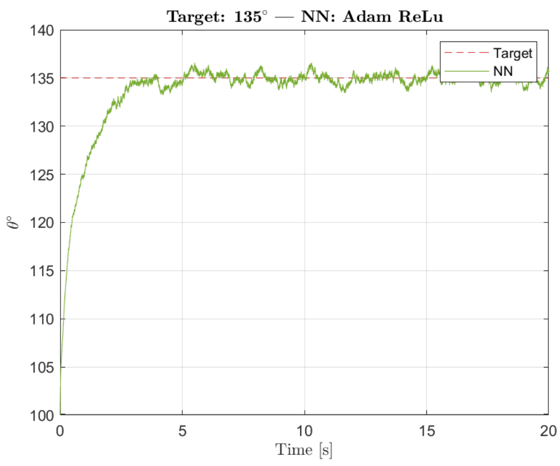


(b) Target 135° with load 0.35kg

Figure 7.5: Results for Adam with Tanh activation function



(a) Target 65° with load 0.35kg

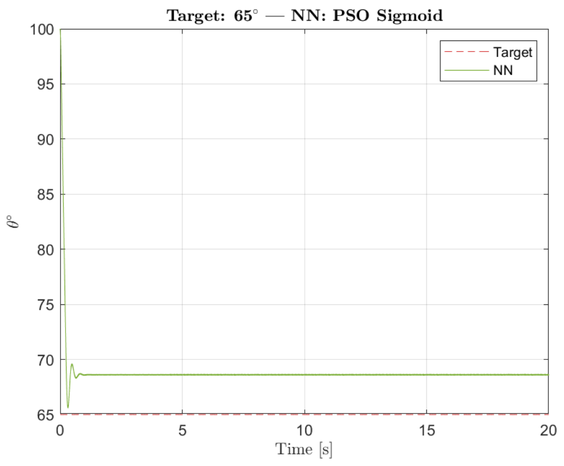


(b) Target 135° with load 0.35kg

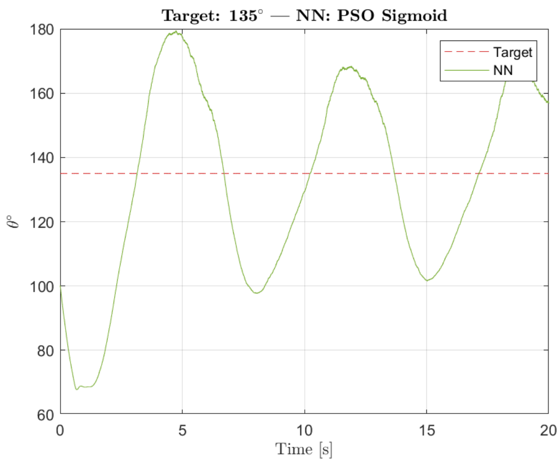
Figure 7.6: Results for Adam with ReLu activation function

As can be seen in Figure 7.4a, Figure 7.5a and Figure 7.6a, similarly to Gradient Descent cases of target angle of 65° the exoskeleton reaches steady state error of 69° in smooth manner. The difference is that this time it is ReLu activation function that shows sign of settling on target and rest cases show sudden drop below target. Furthermore out of all cases shown Figure 7.4b, Figure 7.5b and Figure 7.6b, the ReLu case shows promising results. Although small oscillations occur between time step making signal not very clear, it settles on target angle with very small oscillations up to 1.5°. The other cases have more unstable response of oscillations of 5°.

PSO

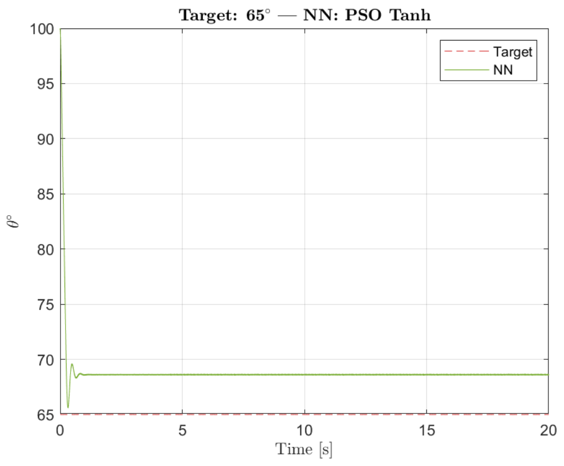


(a) Target 65° with load 0.35kg

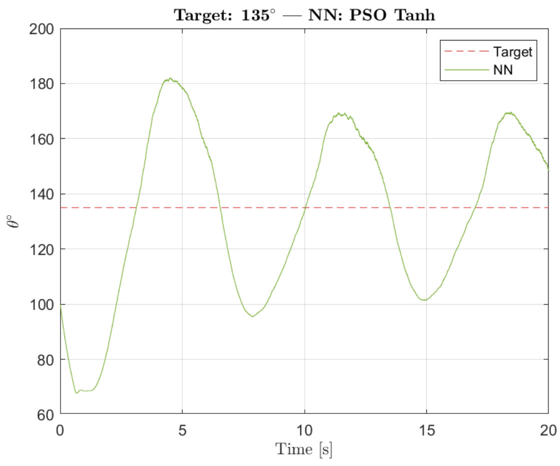


(b) Target 135° with load 0.35kg

Figure 7.7: Results for PSO with Sigmoid activation function

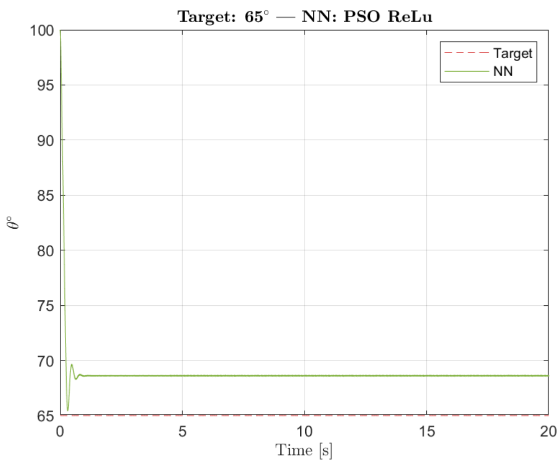


(a) Target 65° with load 0.35kg

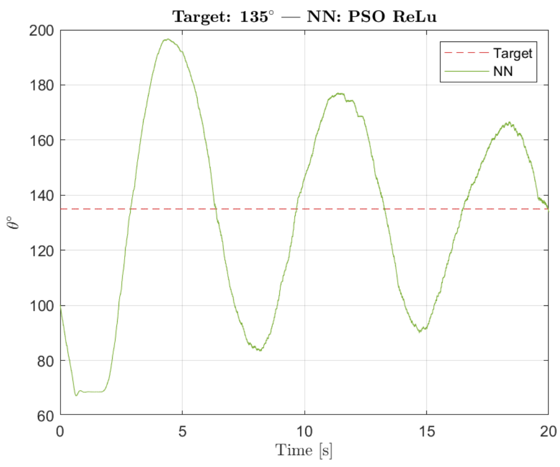


(b) Target 135° with load 0.35kg

Figure 7.8: Results for PSO with Tanh activation function



(a) Target 65° with load 0.35kg



(b) Target 135° with load 0.35kg

Figure 7.9: Results for PSO with ReLu activation function

All cases of 65° seen in Figure 7.7a, Figure 7.8a and Figure 7.9a, have same response quick raise and steady state error at 69°. Also all cases for 135° shown Figure 7.7b, Figure 7.8b and Figure 7.9b, have same response of big oscillations around target.

Sin wave response

The following section will present the results for the sin wave response of the system.

Gradient Descent

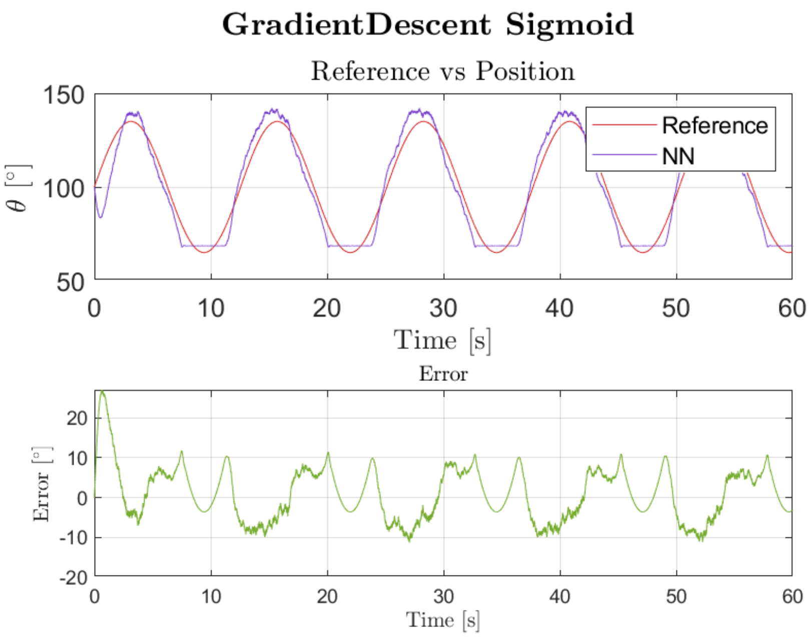


Figure 7.10: Trajectory tracking for Gradient Descent with Sigmoid activation function

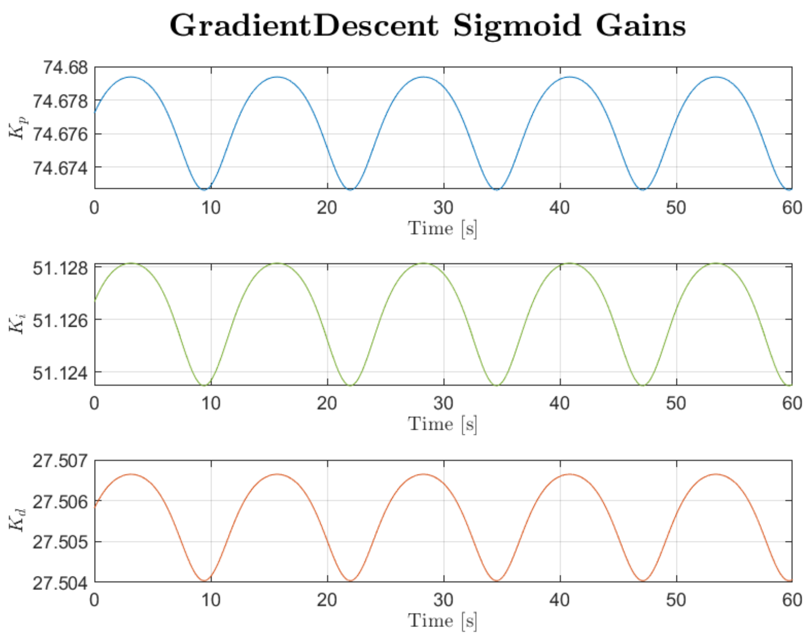


Figure 7.11: Gain scheduling for Gradient Descent with Sigmoid activation function

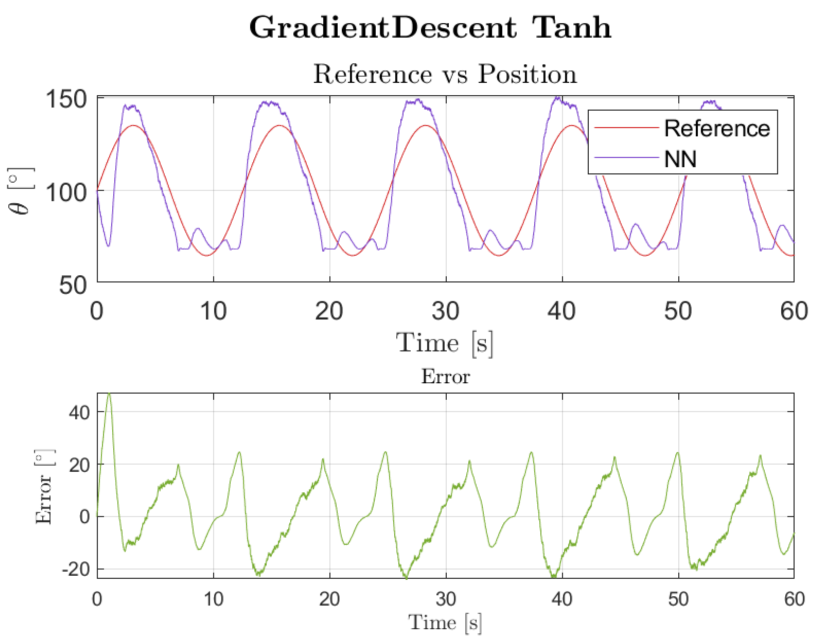


Figure 7.12: Trajectory tracking for Gradient Descent with Tanh activation function

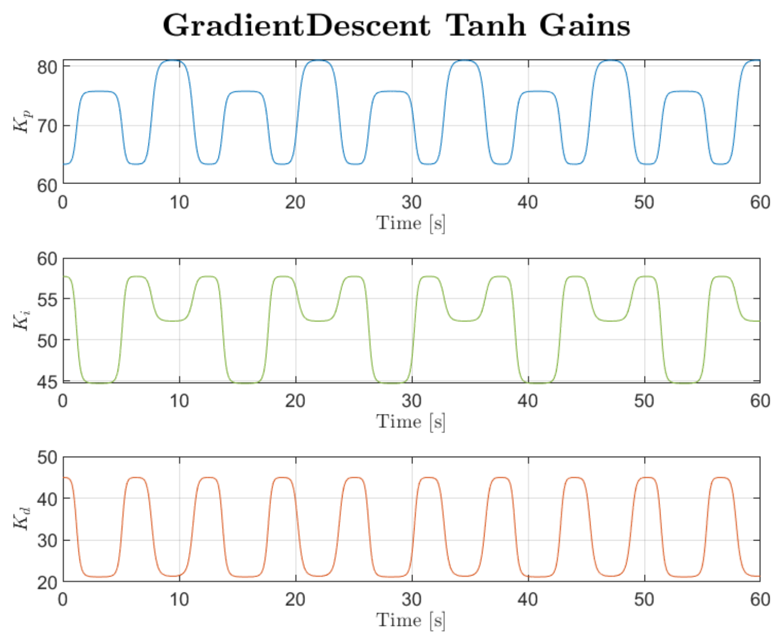


Figure 7.13: Gain scheduling for Gradient Descent with Tanh activation function

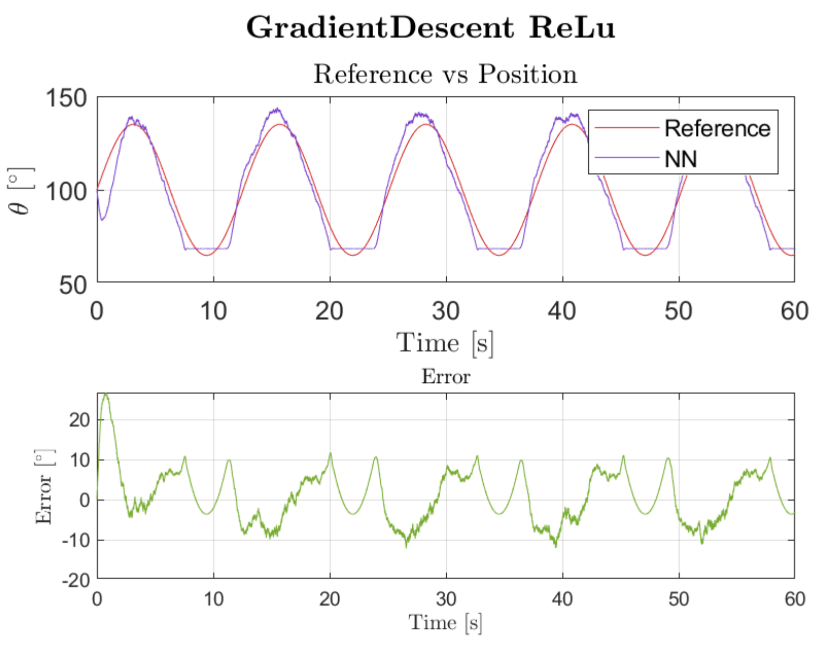


Figure 7.14: Trajectory tracking for Gradient Descent with ReLu activation function

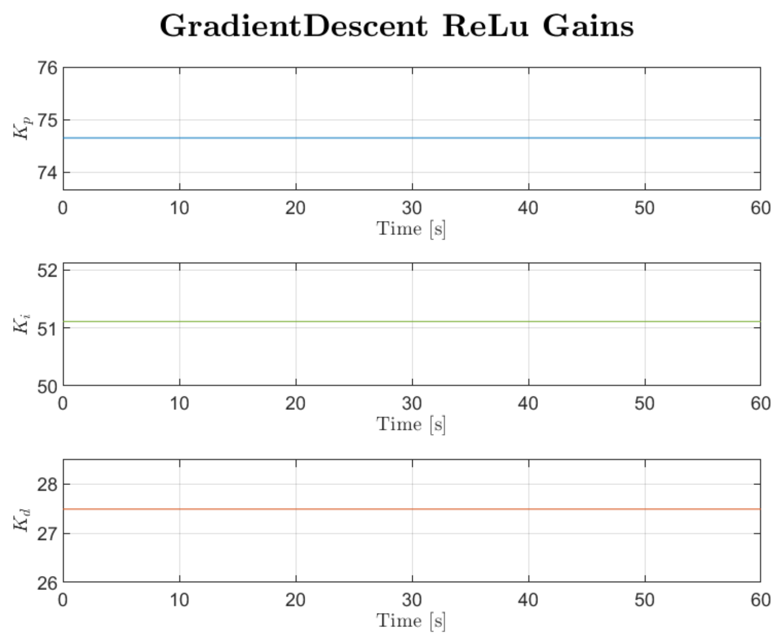


Figure 7.15: Gain scheduling for Gradient Descent with Relu activation function

The results for neural networks trained with Gradient Descent are shown in Figure 7.10, Figure 7.12, and Figure 7.14. Sigmoid and ReLU activation functions gave better results with errors around ±10°, while Tanh resulted in errors up to ±20°. Additionally, the Tanh-based network showed significant changes to PID gains (Figure 7.13). Although scheduling with Sigmoid (Figure 7.11) was functional, the changes were marginal. ReLU-based scheduling (Figure 7.15) maintained constant gains.

Adam

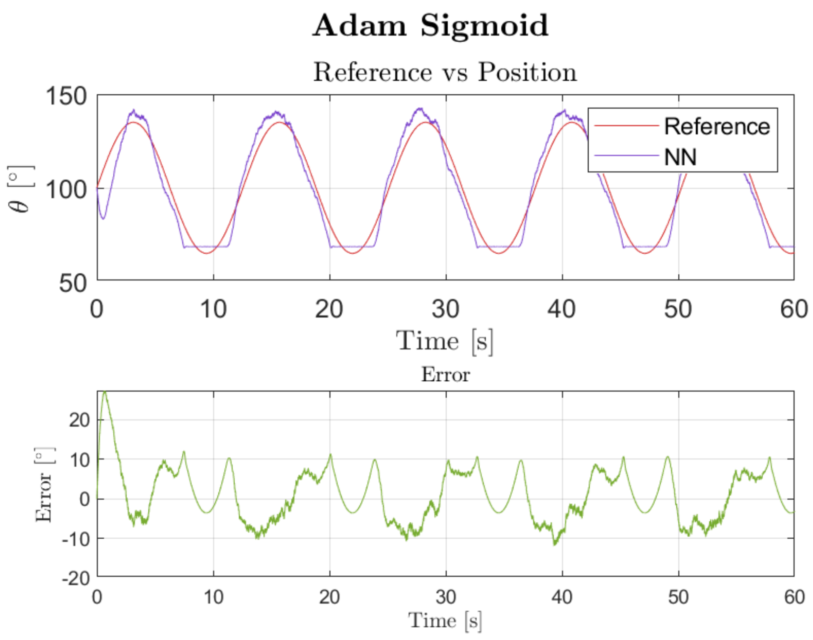


Figure 7.16: Trajectory tracking for Adam with Sigmoid activation function

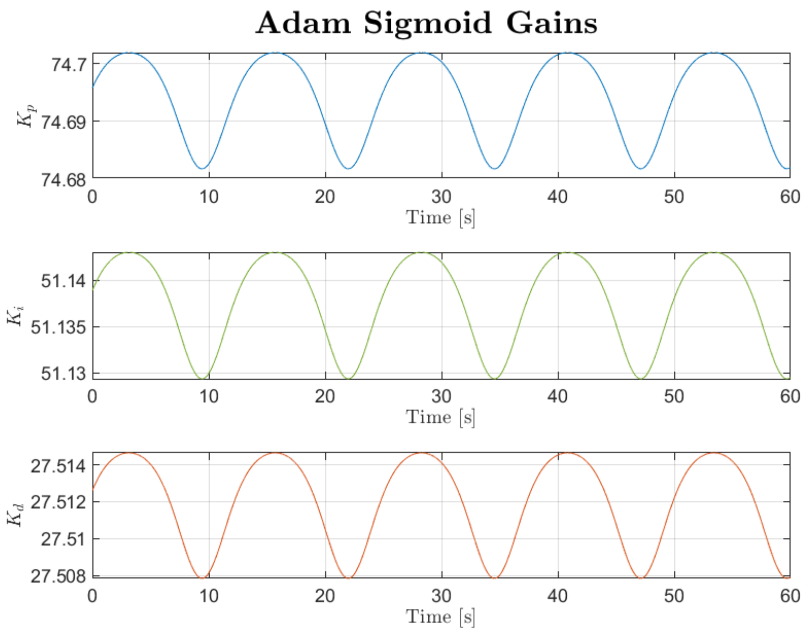


Figure 7.17: Gain scheduling for Adam with Sigmoid activation function

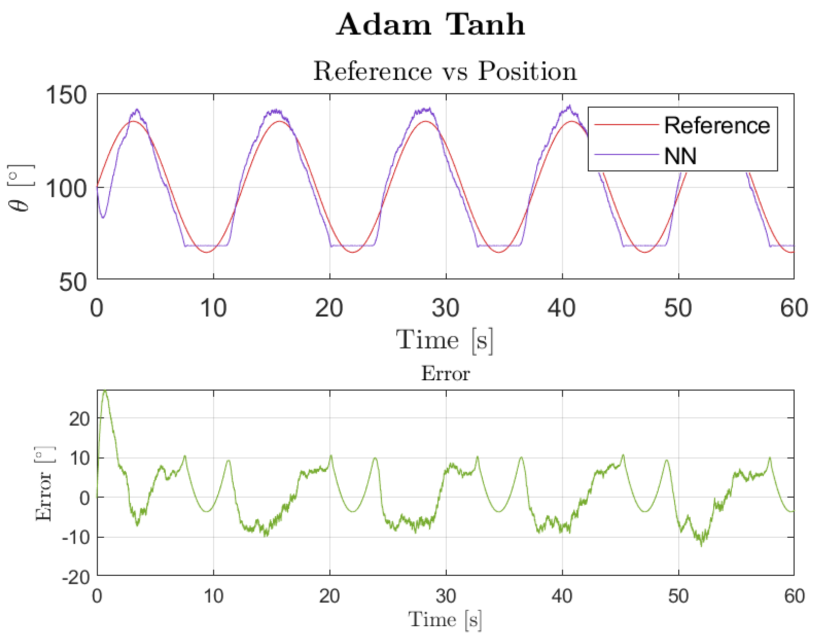


Figure 7.18: Trajectory tracking for Adam with Tanh activation function

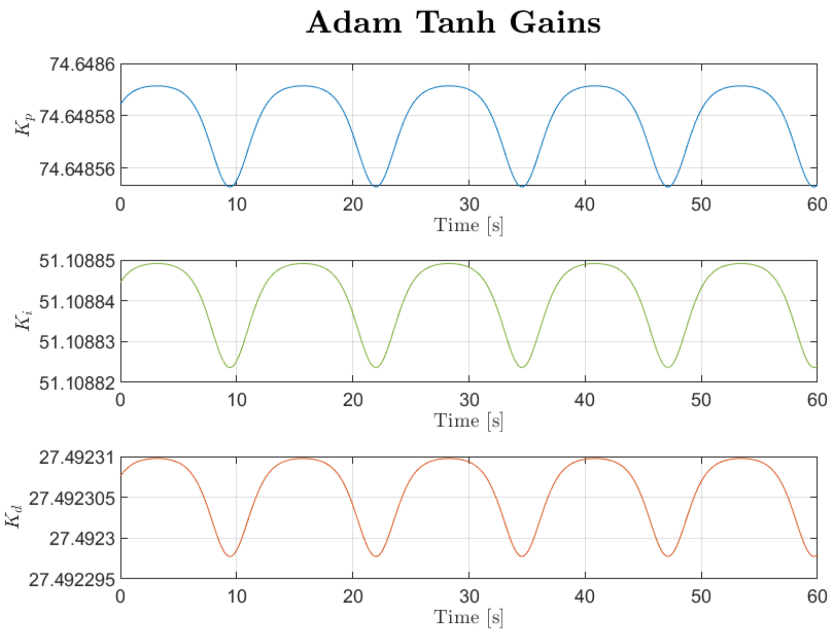


Figure 7.19: Gain scheduling for Adam with Tanh activation function

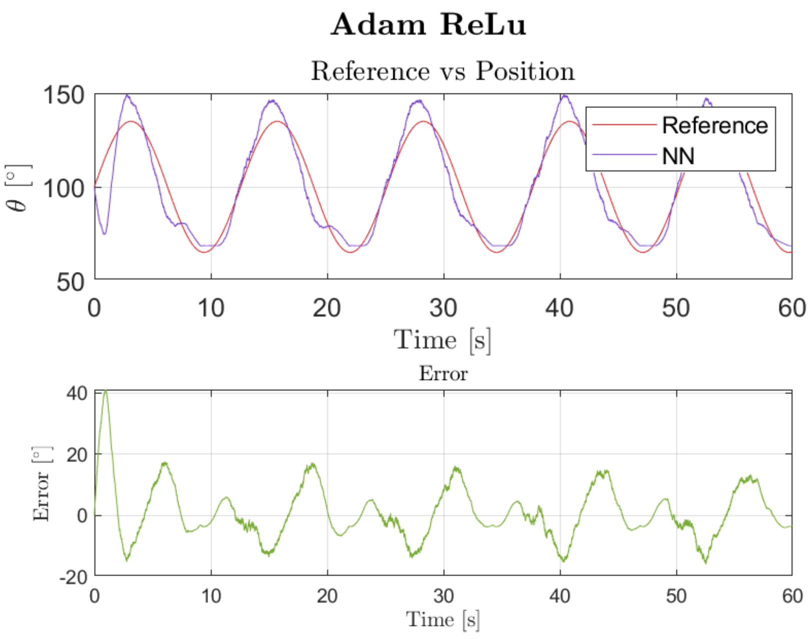


Figure 7.20: Trajectory tracking for Adam with ReLu activation function

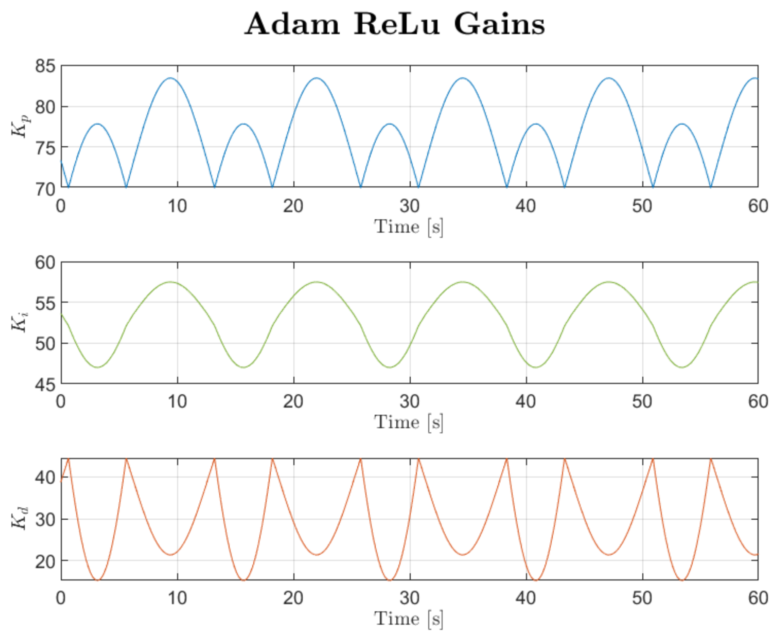


Figure 7.21: Gain scheduling for Adam with Relu activation function

Adam based trained neural networks that used Sigmoid and Tanh, seen in Figure 7.16 and Figure 7.18 respectively, have position error of $\pm 10°$. Although the ReLu based neural network achieved lower accuracy it only has peaks of 20° when target is 135°. While it has small error while reaching 65°. Furthermore, the more accurate networks, the Simoid and Tanh have marginal change in the gains, which is presented in Figure 7.17 and Figure 7.19. The network that uses ReLu activation schedules the gains more aggressively and overall changes are more distinct, as shown in Figure 7.21.

PSO

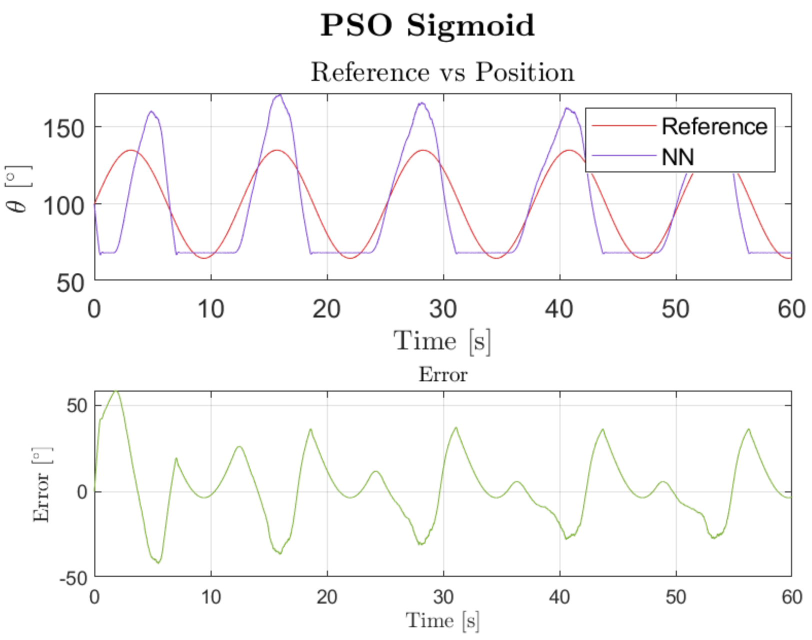


Figure 7.22: Trajectory tracking for PSO with Sigmoid activation function

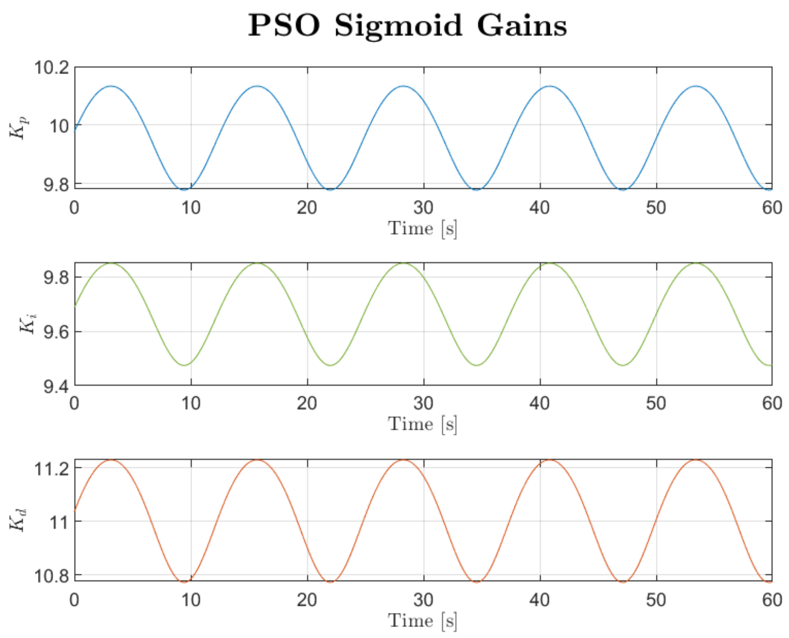


Figure 7.23: Gain scheduling for PSO with Sigmoid activation function

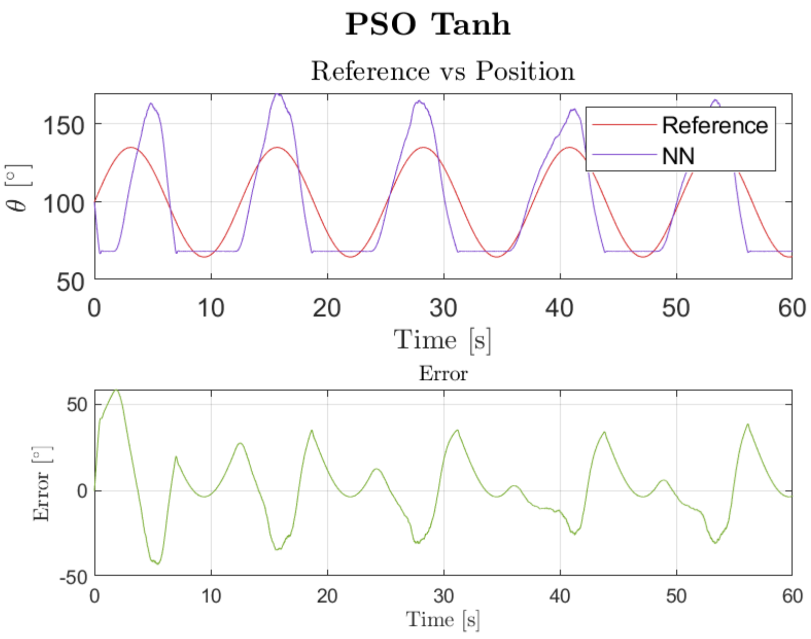


Figure 7.24: Trajectory tracking for PSO with Tanh activation function

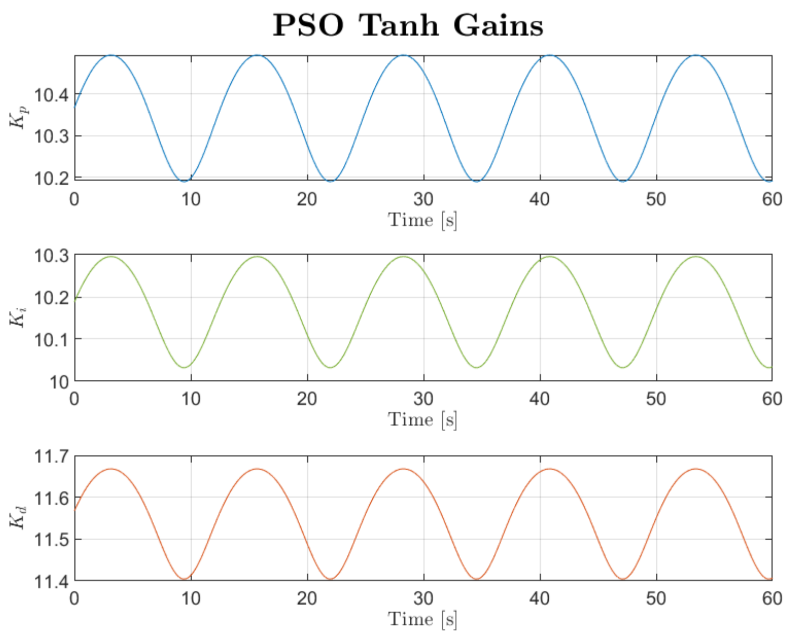


Figure 7.25: Gain scheduling for PSO with Tanh activation function

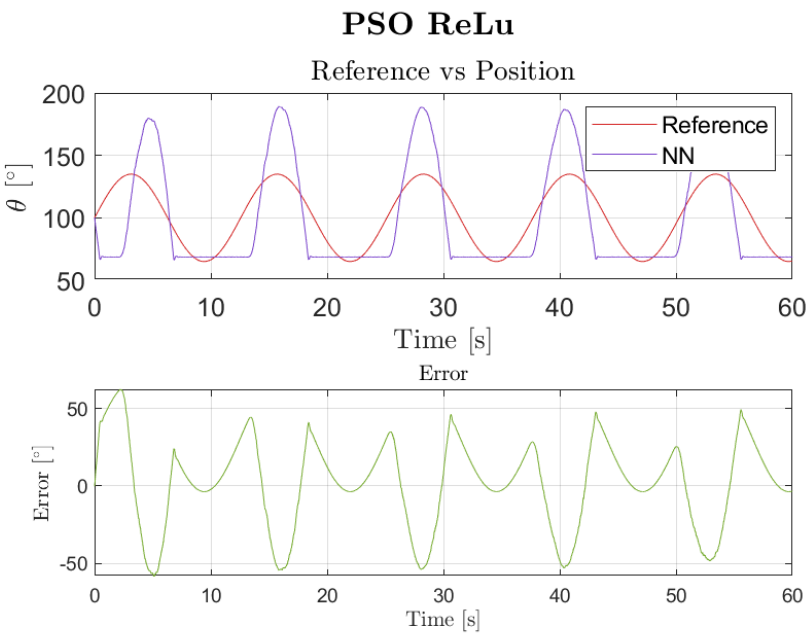


Figure 7.26: Trajectory tracking for PSO with ReLu activation function

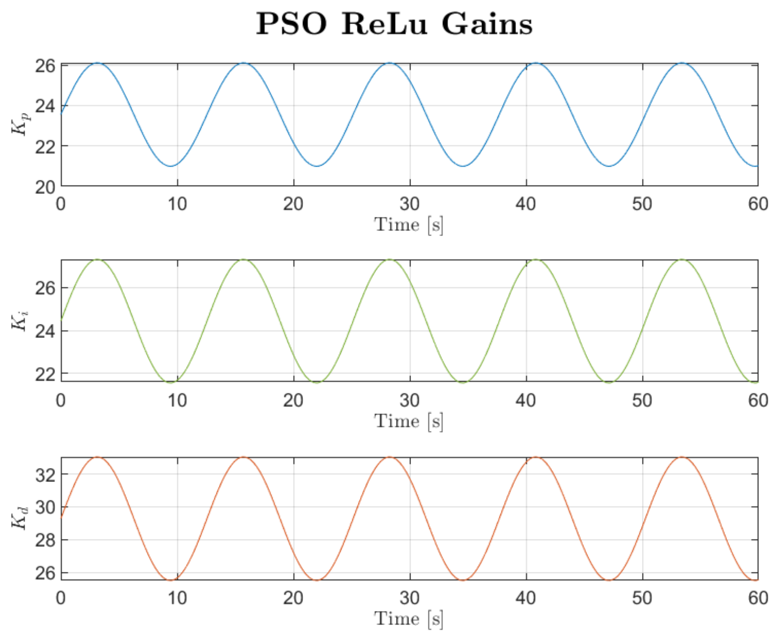


Figure 7.27: Gain scheduling for PSO with Relu activation function

The PSO results seen in Figure 7.22, Figure 7.24 and Figure 7.26 show low accuracy. Regardless the activation function the error oscillates ±50°. Moreover it can be seen that while target is 135° the controller overshoots and target of 65° exoskeleton stops goes quickly to target and stops until signal raises again to 135°. Furthermore the gains creatad by shown in Figure 7.22, Figure 7.24 and Figure 7.26.

Results Discussion

Overall performance of the NNPID in Simulation varies across used neural network and used activation function the summarised results can be seen in Table 7.1. The main issue seen is big discrepancy of results between 135° and 65°. As neural network is as good as training data it learns from. Therefore the source of this behavior is probably due to either poor data obtained from the optimiser for in that specific range or overall data output from optimiser did not have any corelation between data points which resulted in semi-working Neural Network.

Futhermore, most unreliable results were seen from PSO as target of 135° render system unstable, similar response was seen in sin wave response. This was probably due to overfitting data by PSO as it looks globally for local minima. Furthermore the Gradient Descent results show better response compared to PSO, but the response both for step and sin wave are not desirable. The poor results from this network might be due to training algorithms getting stuck in local minium.

Out of all results the neural network trained with Adam and using ReLu activation function gave best results for step reaction. The reason to it's success is that the fact that adam algorithm doesn't get stuck in firs local minima and is capable of finding of better fit for data unlike gradient descent. Although sin wave results are still not satisfactory it is one of better performing networks in that category. Furthermore issues of networks with sin wave might be due to the fact that they were all trained on step input.

Training Function	**Activation Function**	**Step - 65°**			**Step - 135°**			**Sin wave**	
		Rise Time	Error	Other	Rise Time	Error	Other	Tracking error	Other
Gradient Descent	Sigmoid	<1s	4°	Unstable	2s	5°	Oscialtion around target	±10°	Small changes in gain values
	Tanh	<1s	4°	Stable	2.5s	5°	Oscialtion around target	±20°	Gain shcheduling works
	ReLu	<1s	4°	Unstable	2s	5°	Oscialtion around target	±10°	Constant values of gains
Adam	Sigmoid	<1s	4°	Unstable	2s	5°	Oscialtion around target	±10°	Small changes in gain values
	Tanh	<1s	4°	Unstable	2s	5°	Oscialtion around target	±10°	Small changes in gain values
	ReLu	<1s	4s	Stable	2.5s	1.5°	Small oscialtion around target	±15°	Gain shcheduling works
PSO	Sigmoid	<1s	4°	Stable	-	-	Unstable	±40°	Small changes in gain values
	Tanh	<1s	4°	Stable	-	-	Unstable	±40°	Small changes in gain values
	ReLu	<1s	4°	Stable	-	-	Unstable	±50°	Gain shcheduling works

Table 7.1: Summarised results from testing

Physical Testing

For testing on the physical elbow exoskeleton, combinations of two different loads (83 g and 238 g) and two target angles (37° and 82°) were used. These conditions were also outside the training dataset and intended to evaluate the NNPID's generalization capability.

Trajectory tracking was additionally tested on hardware, using a sinusoidal path centered at 90° with a ±30° amplitude and a 0.5 Hz frequency. For comparison, a conventional PID controller gains were chosen from the gathered data one set per load and used as a performance benchmark.

Results

The results are divided into two categories: step response and sin wave tracking. Data was gathered using Matlab code which had live plotting of the data send from Arduino and GUI that enabled to change target value. A Before each test the for step response the initial target was set to 0°. For the sin wave as some PID values were unstable the code would move the arm to 90° in two step signals before changing the singal to sin wave.

Step response

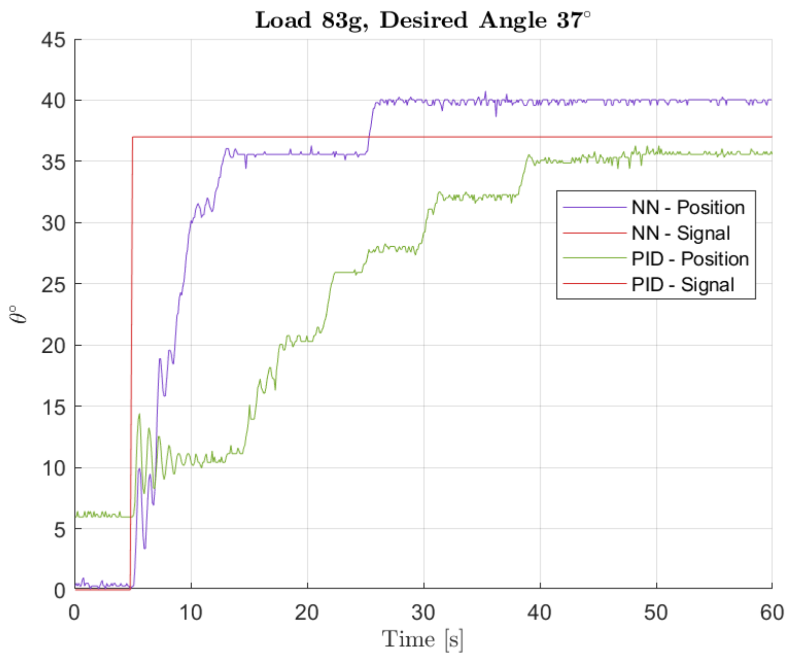


Figure 7.28: Results for Load 83g and Target angle 37°

As shown in Figure 7.28, the NNPID exhibits a fast response to the step signal. While some jerk movement is observed, the controller with a rise time of approximately 6 seconds, reaches the target without overshooting and maintains 2° steady-state error untill it overshoots the refrence signal by 3°. In contrast, the PID controller has 40 seconds rise time and steady state-error is 2°.

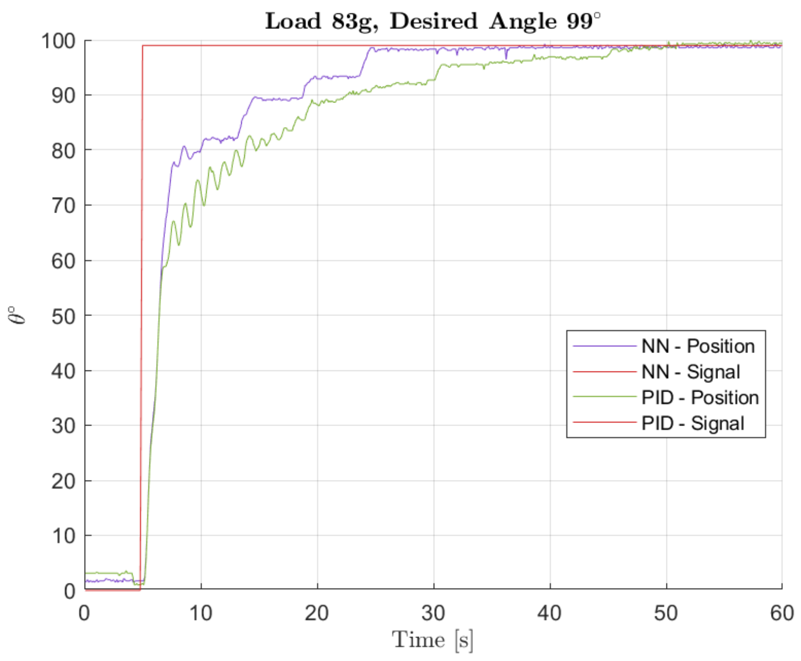


Figure 7.29: Results for Load 83g and Target angle 99°

Figure 7.29 in this setup NNPID slightly outperformed normal PID. Rise time NNPID is 10second but it takes another 10 second to reach target with 0° steady error. Moreover the NNPID had moments in which position us doesn't change. The PID controller, follows similar trajectory as NNOID but is slower to reach 90% of signal in 20 seconds and achives target in 40seconds.

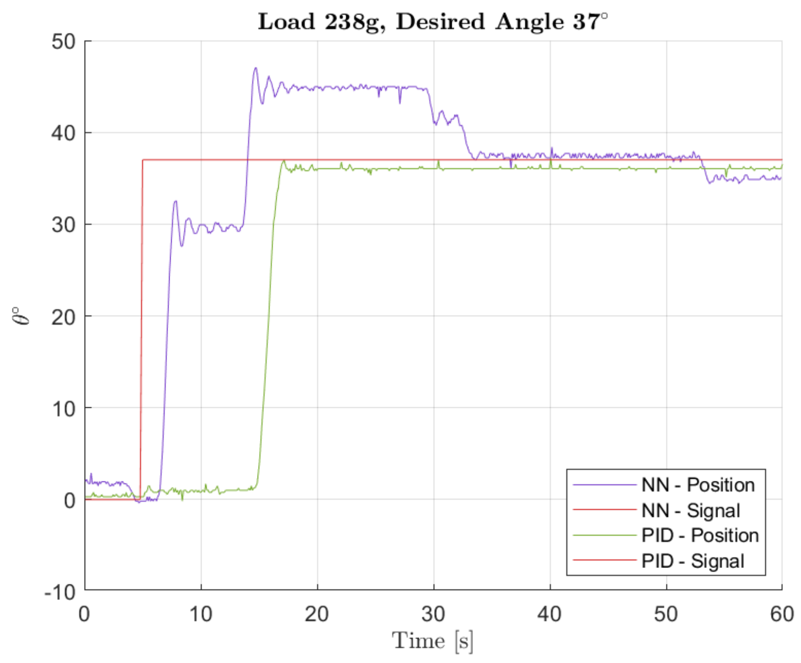


Figure 7.30: Results for Load 238g and Target angle 37°

In Figure 7.30, NNPID is has fast response but first settles for 5 seconds in 30° position to overshoot by around 10° after 20 second it reaches the target but at the end of recording it goes to 35°. PID response is better in terms of rise time and steady state error which is about 2°. The response to the signal is slow as it takes 10 seconds for motor to start moving.

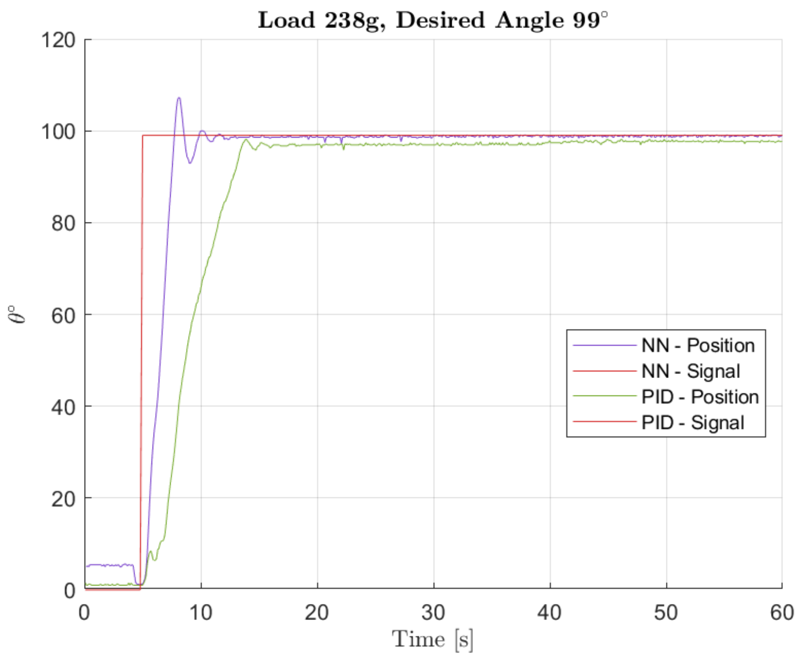


Figure 7.31: Results for Load 238g and Target angle 99°

As illustrated in Figure 7.31, both controllers have quick response. NNPID has rise time of 2.5 seconds and settle in 6 seconds. Addionnaly NNPID overshoots 7°. The PID controller has 9 second rise time and 3° steady state error

Sin wave response
show results from sin wave trajectory tracking.

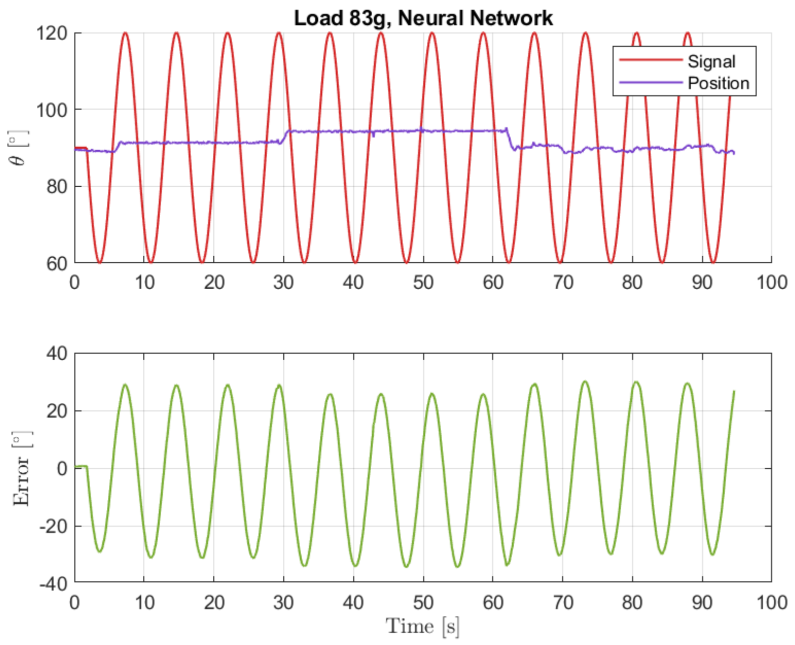


Figure 7.32: Results for trajectory track with Neural Network load 83g

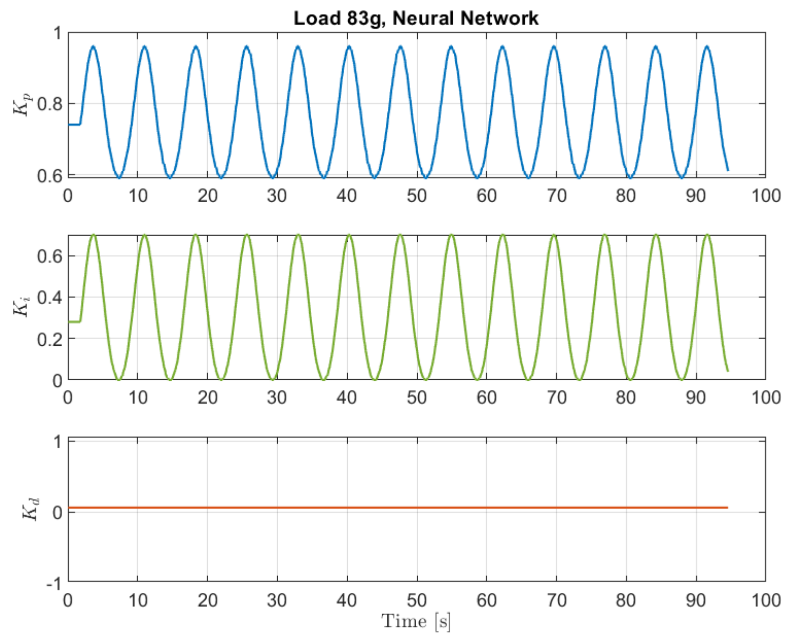


Figure 7.33: PID gains change for trajectory track with Neural Network and load 83g

Figure 7.32 shows that the trajectory tracing is not working with NNPID and position is changes only by couple degrees. Nevertheless, Figure 7.33 shows that Neural Network actively changes PID gains based on given angle.

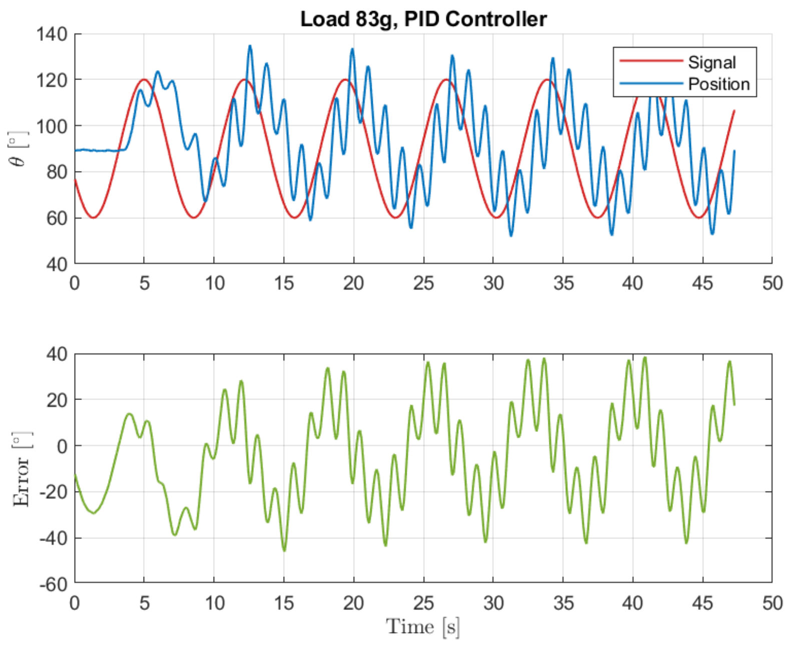


Figure 7.34: Results for trajectory track with PID load 83g

As seen in Figure 7.34 the arm tracks the wave signal, but not very efficiently as the position constantly osciclates around signal.

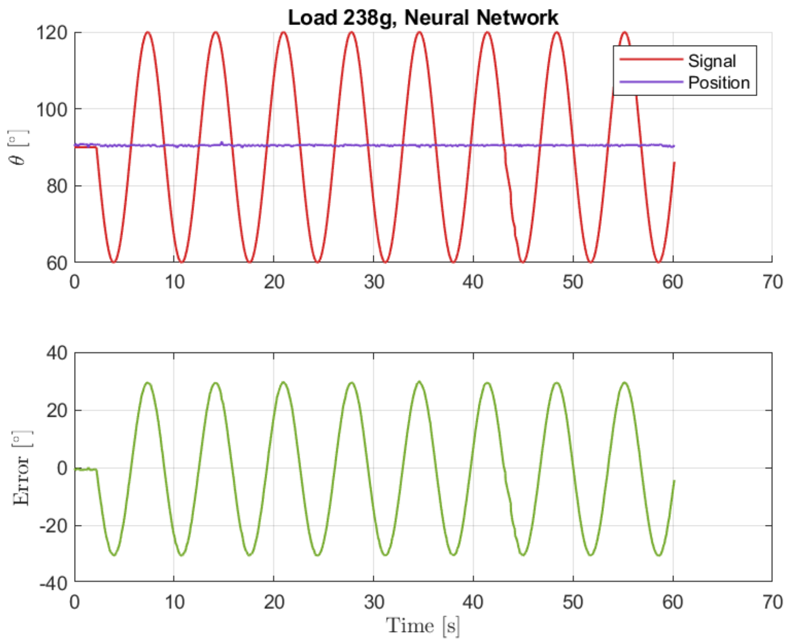


Figure 7.35: Results for trajectory track with Neural Network load 238g

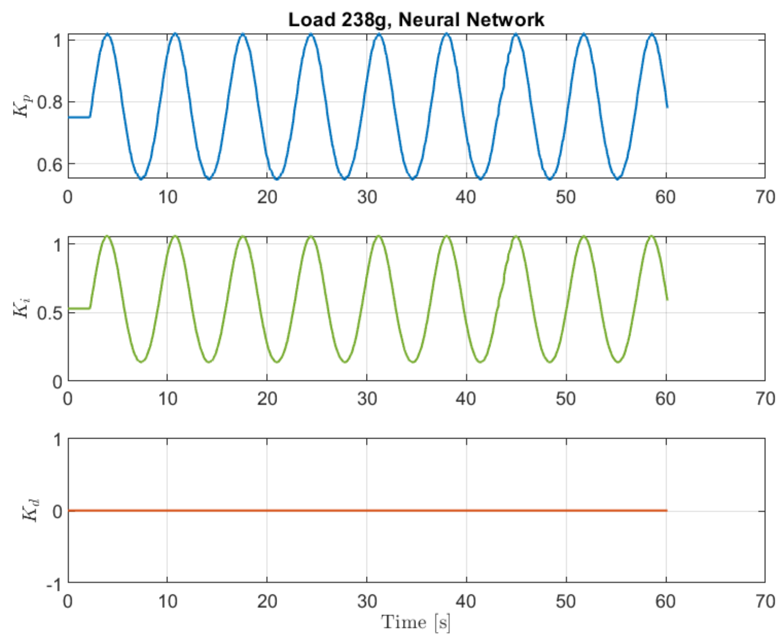


Figure 7.36: PID gains change for trajectory track with Neural Network and load 238g

Figure 7.35 shows that gains produced by Neural Network are not suitable for trajectory tracking as the arm is in steady position. Furthermore Figure 7.36 shows that gains scheduling works in real time, Moreover the values produced for this scenario are different from 83g case.

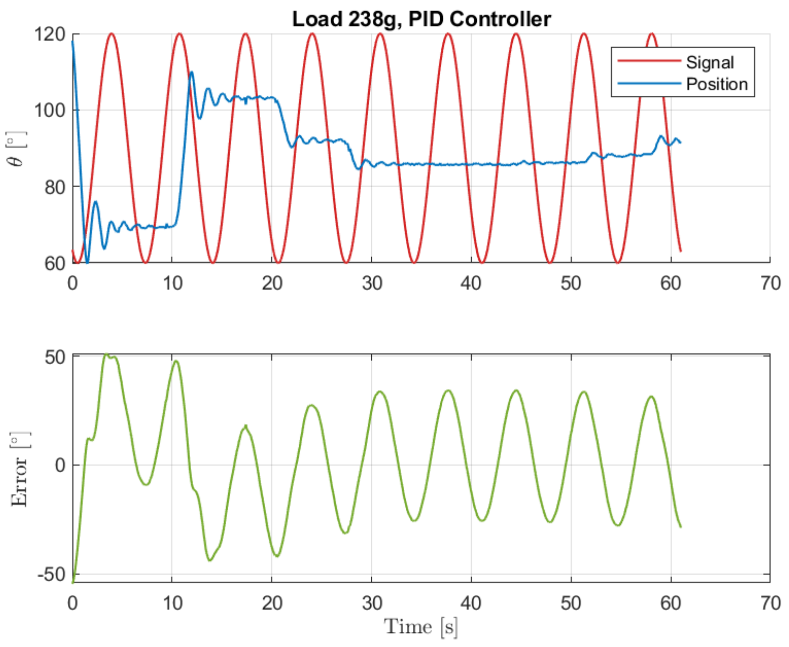


Figure 7.37: Results for trajectory track with PID load 238g

In Figure 7.37 the peroformance of PID controller can be seen. Overall the position does not follow sin wave signal. after two moves the position settles at 90° positio

In the case of a load of 37g, both the NNPID and the regular PID demonstrated an inadequate response. Despite the presence of small oscillations in NNPID, they are insufficient to be considered as successful tracking. It is observable that for 238g, the response for NNPID is more favourable, yet the controller is out of phase and exhibits an additional jerk on top of the sinusoidal waveform. This configuration exhibits superior performance in comparison to the conventional PID controller, as the response approaches a square signal prior to settling at 90°.

Results Discussion
The results show promising performance of the proposed system. Although a properly tuned PID would yield better results than the NNPID, it would be difficult to determine PID values for every case. Therefore, NNPID demonstrates its superiority, being based on scarce data. Based on performance of the both tuned values could definitely be improved. This can be seen as the quality of the network correlates directly with the quality of the data points. As for trajectory tracking, the main reason for poor results is that the PID values were tuned for a step response with high initial error. During trajectory tracking, small errors arise at each time step between the reference and the current position. Therefore, the training function should also be trained in multiple different scenarios that the initial state of the exoskeleton is not equal to zero for example 30° to 90°.

7.2 High-level control

The testing procedure followed the same setup as the data collection phase, with the key difference being that a continuous MATLAB script was used to record data in real time while running the uploaded neural network for live load prediction. Tests were conducted on three subjects, who were tasked with lifting and holding various loads, as well as swapping them while data recording continued. The loads used during testing were 1kg, 4kg,and 5kg.

7.2.1 Results

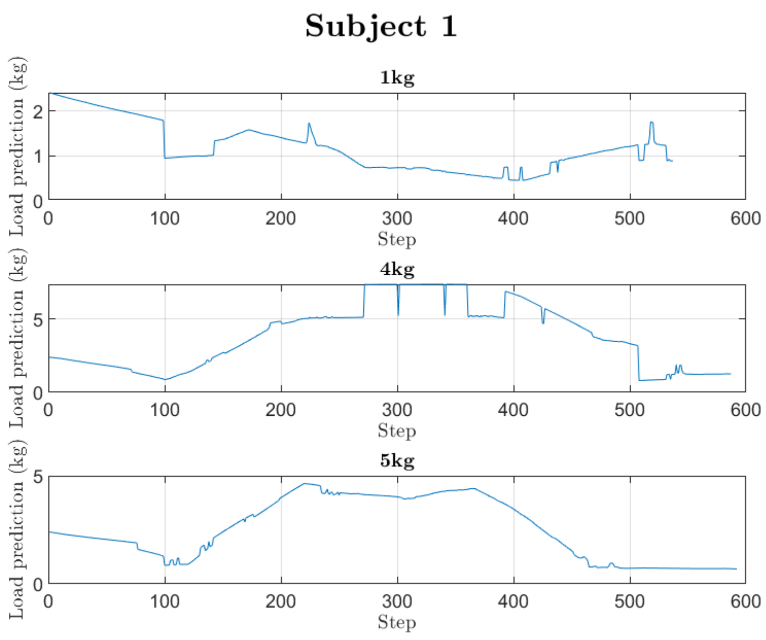


Figure 7.38: Results obtained from Subject 1

Figure 7.38 shows that before grabbing the load the network's initial prediction is about 2kg. For Subject 1 in 1kg scenario the prediction oscilates around 1kg. In 4kg scenario the prediction to 5kg and 7kg. In last scenario prediction is just below 5kg.

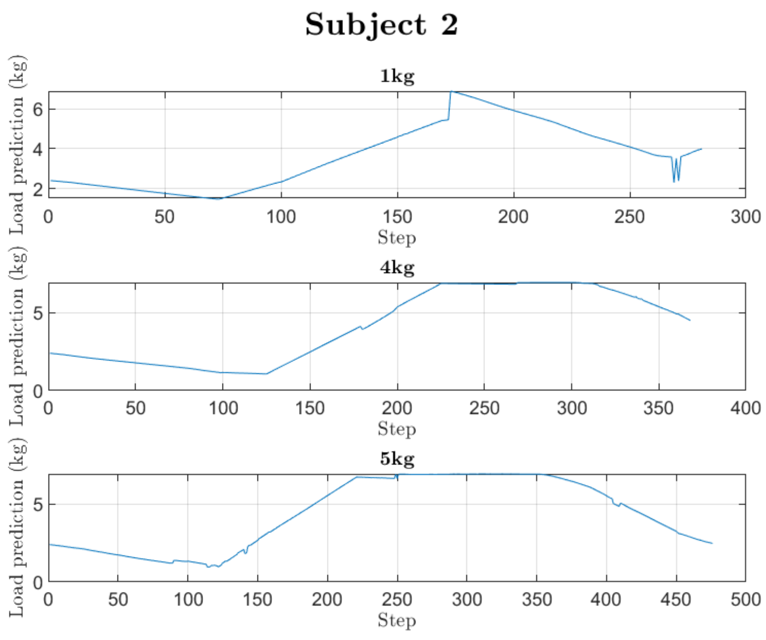


Figure 7.39: Results obtained from Subject 2

Figure 7.39 shows that before grabbing the load the network's initial prediction is about 2kg. For Subject 2 in 1kg scenario the prediction jumps to 7kg. In 4kg and 5kg scenarios goes to 7kg.

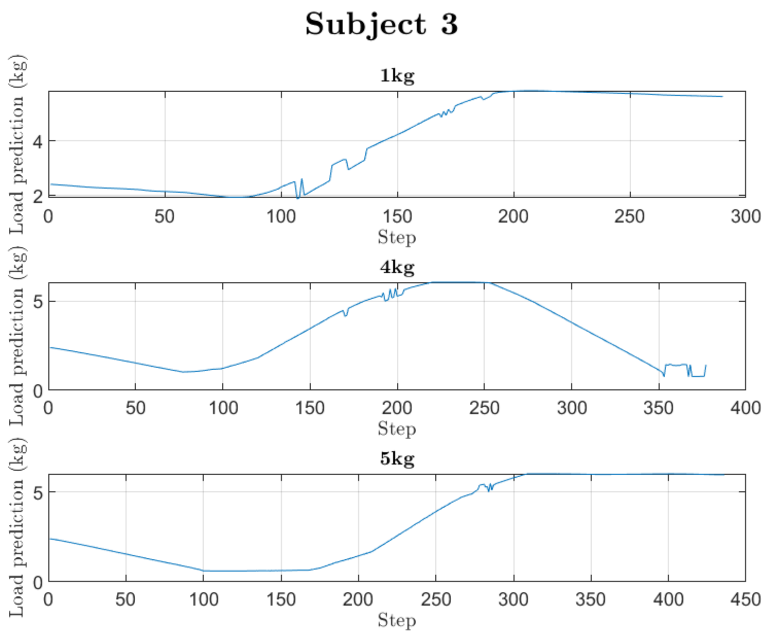


Figure 7.40: Results obtained from Subject 3

Figure 7.40 shows that before grabbing the load the network's initial prediction is about 2kg. For Subject 3 in 1kg scenario the prediction jumps to 7kg. In 4kg and 5kg scenarios goes to 6kg.

Results Discussion

The results show that live prediction does not work accurately. This is due poor data. Although a lot of care was put into gathering data it was proved difficult to get consistent data. There are multiple factors that have an effect on signal like circumference of the arm, biceps circumference. This changes overall tightness of the band during data gathering and testing. Moreover some subjects had to be rejected from data gathering due to arm band being too short to go around their arms. Although testing was a failure if more reliable way to ensure clarity of data the system would be able to predict holding the load in the arm.

Furthermore, based on raw data it seams that main sensor activated by holding the weight in elbow bend 90° is FSR number 7 which is on more on the inside of the arm. Although it was expected to have sensors 7 and 8 to have higher activation than other, that fact limits prediction as only 2 out of 8 sensors can hold valuable information. Additionally the muscles that are responsible on the back, therefore shoulder activation can not be detected through armband.

Moreover it was also noticed that grip the participant were using also change the activation of the muscles in arm which also affected the prediction. Therefore during testing they were asked to hold the weight in same way as during data gathering.

8 Discussion

The implementation of a control system for a shoulder exoskeleton equipped with a Variable Stiffness Mechanism (VSM) has provided insight into industrial-grade exoskeletons. Furthermore, this project has presented a unique opportunity to explore the challenges associated with the control of non-linear systems and the prediction of user intention. Although the system was not incorporated into the final exoskeleton, the simulation and demonstration were successfully carried out.

8.1 Modeling

The main issue with the model was its low accuracy. Although the simulated behavior resembled that of the physical system, the numerical values showed noticeable discrepancies. This was likely due to unaccounted friction within the system. Additionally discrepancy between real life and model of the exoskeleton, was that moment of inertia and center of gravity was calculated wrongly. For the future simulations the CAD model of the exoskeleton should be used to obtain that data. Moreover, as the spring's elongation for the VSM can be changed. It is possible that worm gear was moved between VSM torque test and idle system performance check.

Although the optimiser created in MATLAB performed well, the initial fmincon function often got stuck around the initial guess. This resulted in all the data points being very close to each other. To introduce more randomness to the search, the PSO was introduced to the optimiser. While the optimizer produced usable results, the existence of multiple gain combinations that satisfied the cost function suggests potential variability in the training data. This could have negatively impacted the neural network's ability to generalize, as it relies on consistent patterns within the dataset. Consequently, the quality and consistency of training data became a limiting factor.

8.2 Low-level control

The performance of the neural network controller was found to be highly dependent on three factors: the quality of the training data, the choice of training algorithm, and the activation function used. Results from physical testing indicated that the neural network can perform reliably—provided it operates within the range of data it was trained on. While the network performed well during step responses—which were part of the training set—it failed during trajectory tracking scenarios for which it had no prior data. It failed during trajectory tracking scenarios, which it did not have data on. This suggests that the training dataset needs to be expanded to include wider variety of scenarios. Moreover, the mechanical bandwidth of both exoskeletons should be assessed to determine whether the selected oscillation frequencies were appropriate for the system. Without aligning the controller's dynamic requirements with the actuator capabilities.

8.3 High-level control

The load prediction system utilized only 8 out of the 21 available signal channels from the armband sensors. Furthermore, the testing scenarios were static, which limited the model's ability to generalize to dynamic situations. Future implementations should focus on dynamic data collection, such as moving the shoulder up and down while maintaining a fixed elbow angle, to better simulate real-world conditions. These more complex movement patterns could also be leveraged to train the intention prediction model.

8.4 Other Consideration

Furthermore in order to test system in real time Matlab code was implemented to send desired angle and record data. It was noticed noticed that plotting has considerable delay about 5 seconds. It did not affected the system's performance as Matlab script was just listening to what Arduino was sending through Serial port. This delay affected the performance of the prediction as each prediction was actually prediction from 5 seconds ago. This delay was probably due to how Matlab is processing Serial data and plots that data in real-time.

Finally the system was not tested on humans. This was due to delimitations that were described in Chapter chapter 3. The system was not tested on human subjects due to project limitations, as outlined in Chapter chapter 3.
Human testing was not conducted in this project, primarily due to the mechanical design limitations of the exoskeleton described in Chapter chapter 3. Furthermore, based on the performance of the control systems during physical testing, the reliability of the current setup was not sufficient to justify human trials.

9 Conclusion

In this thesis, a complete control framework for a robotic exoskeleton system was developed, simulated, and tested. The simulation environment was implemented in Simulink, where automatic PID tuning was employed to obtain baseline control performance. Building on this, nine different neural network architectures were trained and evaluated for integration into a nonlinear control system. One of these, based on the Adam optimizer with ReLU activation, demonstrated the most promising results in simulation.

The chosen neural network-based PID controller (NNPID) was then deployed on an embedded system (Arduino Due) to validate real-time performance. While some performance metrics were met in hardware, others—such as rise time and overshoot under specific conditions—did not consistently meet the specified thresholds. These outcomes highlight the challenges of translating simulation performance into embedded implementation.

In parallel, an LSTM-based load prediction module was developed and tested. Although the implementation was completed successfully, its accuracy was insufficient to meet the defined criteria. The intention prediction system, originally planned as part of the work, could not be realized within the available time frame and remains an open area for future development.

Overall, while not all functional requirements outlined in Chapter 3 were fully achieved, the project successfully demonstrated the feasibility of using neural network-based control for nonlinear systems and embedded applications. The insights gained from the simulation, hardware integration, and neural network evaluations provide a strong foundation for future work aimed at refining controller robustness and extending system capabilities to include prediction-based assistance.

Future Works

One of the primary limitations observed in this study was the quality of the training data used for the neural network-based controller. Future work should therefore prioritize improving and validating a high-fidelity model of the new exoskeleton. With a reliable model in place, data for neural network training could be generated automatically using optimization algorithms, reducing the need for manual data collection and thereby saving significant development time and effort.

Moreover, to properly train the neural network, the initial conditions should vary by starting from different joint angles rather than always beginning from the position in which the VSM was locked. This would expand the input space of the neural network to four key features: the VSM lock position, the current joint position, the target position, and the load. Incorporating these variables would allow the network to generalize more effectively across a wider range of motion scenarios.

For the load prediction component, future efforts should focus on improving the consistency of data acquisition. In the current setup, variations in the tightness of the armband significantly affected signal quality. A more robust solution would involve using two armbands — one on the upper arm and one on the forearm — to better estimate the load position relative to the exoskeleton and provide richer sensory input. Additionally, the load prediction model should be retrained on a more diverse dataset, including movements such as forearm lifting while varying shoulder positions.

Finally, the intention prediction system, which was not implemented in this work, represents a promising direction. An LSTM-based model could be trained using data captured in a motion capture laboratory in combination with armband sensors (FSR and IMU). As with the load prediction, using two armbands during data collection could significantly improve the robustness and accuracy of the intention prediction system.

Bibliography

[1] CPWR, Musculoskeletal disorders (msds) in construction, insert-release-date-here.
URL: https://www.cpwr.com/research/data-center/data-dashboards/musculoskeletal-disorders-in-construction/, (Retrieved: 19-March-2025).

[2] CCOHS, Work-related musculoskeletal disorders (wmsds), insert-release-date-here.
URL: https://www.ccohs.ca/oshanswers/diseases/rmirsi.html, (Retrieved: 19-March-2025).

[3] M. A. Gull, S. Bai, and T. Bak, A review on design of upper limb exoskeletons, eng, 2020, ISSN: 22186581.

[4] Exoskelette, Ottobock shoulder exoskeleton, Accessed: 2025-05-23.
URL: https://www.exoskelette.com/en/exoskeletons/ottobock-shoulder/.

[5] Exoskelette, Auxivo deltasuit, Accessed: 2025-05-23.
URL: https://www.exoskelette.com/en/exoskeletons/auxivo-deltasuit/.

[6] SuitX, Suitx ix shoulder air exoskeleton, Accessed: 2025-05-23.
URL: https://www.suitx.com/en/products/ix-shoulder-air-exoskeleton.

[7] E. Bionics, Ekso evo exoskeleton, Accessed: 2025-05-23.
URL: https://eksobionics.com/ekso-evo/.

[8] HAPO, Hapo up exoskeleton, Accessed: 2025-05-23.
URL: https://www.hapo.eu/en/hapo-exoskeleton/25-hapo-up.html.

[9] ExoIQ, S700 shoulder exoskeleton, Accessed: 2025-05-23.
URL: https://www.exoiq.com/schulter-exoskelett-s700.

[10] Agade, Agadexo shoulder exoskeleton, Accessed: 2025-05-23.
URL: https://agade-exoskeletons.com/en/product/.

[11] L. Exo, Ant-wa1 handling assist robot, Accessed: 2025-05-23.
URL: https://www.luban-exo.de/en/shop/ant-wa1-handling-assist-robot-11.

[12] VIEXO, Viexo.
URL: https://www.viexo.aau.dk/, (Retrieved: 19-March-2025).

[13] Z. Li and S. Bai, A novel revolute joint of variable stiffness with reconfigurability, eng, *Mechanism and machine theory*, vol. 133, pp. 720–736, 2019, ISSN: 0094-114X.

[14] Z. Li, S. Bai, O. Madsen, W. Chen, and J. Zhang, Design, modeling and testing of a compact variable stiffness mechanism for exoskeletons, eng, *Mechanism and machine theory*, vol. 151, pp. 103 905–, 2020, ISSN: 0094-114X.

[15] A novel passive shoulder exoskeleton designed with variable stiffness mechanism, eng, *IEEE robotics and automation letters*, vol. 7, no. 2, pp. 2748–2754, 2022, ISSN: 2377-3766.

[16] Y. Zhu, F. Balser, M. Shen, and S. Bai, Design and evaluation of a novel passive shoulder exoskeleton based on a variable stiffness mechanism torque generator for industrial applications, eng, 2024, ISSN: 22186581.

[17] M. Szumowski and M. Ogonowski, Model predictive control for variable stiffness elasticity actuator, eng, in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, IEEE, 2018, pp. 131–136, ISBN: 9781538661437.

[18] S. M. Tahamipour-Z, S. K. Hosseini Sani, A. Akbarzadeh, and I. Kardan, An assistive strategy for compliantly actuated exoskeletons using non-linear model predictive control method, eng, pp. 982–987, 2018.

[19] A. M. Khan, D.-w. Yun, M. A. Ali, J. Han, K. Shin, and C. Han, Adaptive impedance control for upper limb assist exoskeleton, eng, vol. 2015-, no. June, pp. 4359–4366, 2015, ISSN: 1050-4729.

[20] S. Ding, A. Narayan, F. A. Reyes, S. Han, and H. Yu, Design and control of a novel active shoulder exoskeleton for overhead work assistance, eng, *IEEE/ASME transactions on mechatronics*, vol. 29, no. 6, pp. 4768–4777, 2024, ISSN: 1083-4435.

[21] A. O. Souza, J. Grenier, F. Charpillet, P. Maurice, and S. Ivaldi, Towards data-driven predictive control of active upper-body exoskeletons for load carrying, eng, pp. 59–64, 2023, ISSN: 2162-7576.

[22] B. Hu, B. Mao, S. Lu, and H. Yu, Design and torque control base on neural network pid of a variable stiffness joint for rehabilitation robot, eng, *Frontiers in neurorobotics*, vol. 16, pp. 1 007 324–1 007 324, 2022, ISSN: 1662-5218.

[23] X. Xiong, C. D. Do, and P. Manoonpong, Learning-based multifunctional elbow exoskeleton control, eng, *IEEE transactions on industrial electronics (1982)*, vol. 69, no. 9, pp. 9216–9224, 2022, ISSN: 0278-0046.

[24] M. R. U. Islam and S. Bai, Payload estimation using forcemyography sensors for control of upper-body exoskeleton in load carrying assistance, eng, *Modeling, identification and control*, vol. 40, no. 4, pp. 189–198, 2019, ISSN: 0332-7353.

[25] Maxon Motor AG, Ecxfl32s kl a htq 24v, Motor datasheet, accessed May 31, 2025, 2025.
URL: https://www.maxongroup.com/camroot/pdf/b82d71300923/b82d71300923_1.pdf.

[26] Maxon Motor AG, Escon 50/5 - hardware reference, Hardware reference manual, accessed May 31, 2025, 2021.
URL: https://www.maxongroup.com/medias/sys_master/root/9350561792030/409510-ESCON-50-5-Hardware-Reference-En.pdf.

[27] Arduino, Arduino due, Product page, accessed May 31, 2025.
URL: https://store.arduino.cc/products/arduino-due.

[28] RLS d.o.o., Data sheet: Rmb20 rotary magnetic encoder module (rmb20d01), Datasheet, accessed May 31, 2025, 2023.
URL: https://www.rls.si/eng/fileuploader/download/download/?d=1&id=5&title=Data+sheet%3A+RMB20+rotary+magnetic+encoder+module+%28RMB20D01%29.

[29] RLS d.o.o., Data sheet: Aksim off-axis rotary absolute magnetic encoder (mbad01), Datasheet, accessed May 31, 2025, 2021.
URL: https://www.rls.si/eng/fileuploader/download/download/?d=1&id=96&title=Data+sheet%3A+AksIM%E2%84%A2+Off-Axis+Rotary+Absolute+Magnetic+Encoder+%28MBAD01%29.

[30] T. Stewart, Basiclinearalgebra, Accessed: 2025-05-24.
URL: https://github.com/tomstewart89/BasicLinearAlgebra.

[31] Robust payload recognition based on sensor-over-muscle-independence deep learning for the control of exoskeletons, eng, *IEEE transactions on circuits and systems. II, Express briefs*, vol. 70, no. 9, pp. 1–1, 2023, ISSN: 1549-7747.

Appendices

A Github repository

The github repository for this project can be found at:

https://github.com/Psycho-Night/P10-Exo

B Gravity center and Moment of Inertia calculations

The total mass of the system is:

$$m_{\text{tot}} = m_{m1} + m_{m2} + m_{\text{ex}} = 0.444\,\text{kg} + 0.261\,\text{kg} + 1.233\,\text{kg} = 1.938\,\text{kg} \tag{B.1}$$

The positional parameters are defined as:

$$y_{m1} = 0.17\,\text{m} \tag{B.2}$$
$$x_{m2} = 0.05\,\text{m} \tag{B.3}$$
$$y_{m2} = 0.17\,\text{m} \tag{B.4}$$
$$y_{\text{ex}} = 0.28\,\text{m} \tag{B.5}$$

The moment of inertia around the base is given by:

$$J_p = m_{m1}y_{m1}^2 + m_{m2}(x_{m2}^2 + y_{m2}^2) + m_{\text{ex}}\left(\frac{y_{\text{ex}}}{3}\right)^2 \tag{B.6}$$

The coordinates of the center of gravity are:

$$x_{\text{grav}} = \frac{m_{m2}x_2}{m_{\text{tot}}} \tag{B.7}$$
$$y_{\text{grav}} = \frac{m_{m1}y_{m1} + m_{m2}y_{m2} + m_{\text{ex}}y_{\text{ex}}}{m_{\text{tot}}} \tag{B.8}$$

The distance from the origin to the center of gravity is:

$$r = \sqrt{x_{\text{grav}}^2 + y_{\text{grav}}^2} \tag{B.9}$$

C Data gathered in Simulink

VSM [°]	Desired Position [°]	Load [kg]	Kp	Ki	Kd
60	20	0.1	81.472	9.754	15.761
60	20	0.2	99.023	29.755	11.851
60	20	0.3	95.909	96.318	12.712
60	20	0.4	99.025	73.783	15.241
60	20	0.5	91.057	67.219	12.008
60	30	0.1	77.358	83.044	18.815
60	30	0.2	98.169	96.165	12.654
60	30	0.3	66.712	81.115	23.141
60	30	0.4	99.014	40.513	12.375
60	30	0.5	100	63.046	17.541
60	40	0.1	67.459	97.955	48.384
60	40	0.2	85.69	13.378	19.343
60	40	0.3	98.758	98.747	31.247
60	40	0.4	69.913	27.904	26.051
60	40	0.5	76.044	33.199	12.952
60	50	0.1	59.342	13.976	8.437
60	50	0.2	82.81	28.948	12.338
60	50	0.3	91.318	60.034	11.885
60	50	0.4	75.379	2.124	72.904
60	50	0.5	79.388	76.993	49.672
60	60	0.1	9.595	89.512	42.425
60	60	0.2	47.223	87.572	3.975
60	60	0.3	72.026	86.752	65.392
60	60	0.4	37.716	93.323	20.791
60	60	0.5	89.777	65.107	65.292
60	70	0.1	76.266	32.095	43.276
60	70	0.2	84.016	63.1	1.276
60	70	0.3	57.569	63.162	44.387
60	70	0.4	68.004	35.897	50.911
60	70	0.5	55.693	18.272	95.055
60	80	0.1	92.698	24.682	37.251
60	80	0.2	96.025	14.163	19.526
60	80	0.3	79.889	24.331	29.259
60	80	0.4	100	72.203	13.846
60	80	0.5	48.015	39.745	35.1
60	90	0.1	55.449	45.454	25.171
60	90	0.2	86.974	79.829	20.001
60	90	0.3	60.919	52.831	28.518
60	90	0.4	67.189	4.83	28.609
60	90	0.5	68.894	10.48	19.927
60	100	0.1	91.266	16.537	20.268
60	100	0.2	57.315	99.816	13.375
60	100	0.3	100	59.083	13.85
60	100	0.4	99.01	0.99	18.778
60	100	0.5	23.294	65.144	19.586

Table C.1: Data obtained from optimiser - VSM locked at 60°

VSM [°]	Desired Position [°]	Load [kg]	Kp	Ki	Kd
90	50	0.1	56.389	68.636	11.647
90	50	0.2	92.893	19.459	12.926
90	50	0.3	90.885	5.204	12.207
90	50	0.4	44.043	97.122	31.804
90	50	0.5	75.825	23.725	15.984
90	60	0.1	34.883	100	33.682
90	60	0.2	98.702	1.123	12.079
90	60	0.3	67.106	11.675	16.095
90	60	0.4	57.749	99.016	23.279
90	60	0.5	88.665	0.99	22.405
90	70	0.1	69.913	27.904	26.051
90	70	0.2	76.044	33.199	12.952
90	70	0.3	100	28.489	34.769
90	70	0.4	82.81	28.948	12.338
90	70	0.5	91.318	60.034	11.885
90	80	0.1	75.379	2.124	72.904
90	80	0.2	79.388	76.993	49.672
90	80	0.3	9.595	89.512	42.425
90	80	0.4	47.223	87.572	3.975
90	80	0.5	72.026	86.752	65.392
90	90	0.1	37.716	93.323	20.791
90	90	0.2	89.777	65.107	65.292
90	90	0.3	76.266	32.095	43.276
90	90	0.4	84.016	63.1	1.276
90	90	0.5	57.569	63.162	44.387
90	100	0.1	68.004	35.897	50.911
90	100	0.2	55.693	18.272	95.055
90	100	0.3	22.685	25.962	68.469
90	100	0.4	17.999	78.459	7.314
90	100	0.5	34.498	87.124	87.502
90	110	0.1	63.613	68.258	17.577
90	110	0.2	100	98.013	28.372
90	110	0.3	95.544	61.467	22.876
90	110	0.4	65.963	34.397	21.903
90	110	0.5	56.032	65.646	28.656
90	120	0.1	65.074	20.913	20.125
90	120	0.2	100	83.807	11.966
90	120	0.3	99.276	41.907	15.786
90	120	0.4	76.117	0.99	20.842
90	120	0.5	56.11	35.638	17.954
90	130	0.1	67.503	84.029	12.338
90	130	0.2	77.855	13.582	11.622
90	130	0.3	93.107	48.316	11.7
90	130	0.4	90.112	70.929	11.954
90	130	0.5	39.231	85.244	14.774

Table C.2: Data obtained from optimiser - VSM locked at 90°

VSM [°]	Desired Position [°]	Load [kg]	Kp	Ki	Kd
120	80	0.2	95.774	54.351	12.382
120	80	0.3	95.606	96.267	11.808
120	80	0.4	86.069	92.122	9.59
120	80	0.5	93.785	1.631	31.993
120	90	0.1	60.278	43.476	26.992
120	80	0.1	99.907	8.453	11.3
120	90	0.2	98.867	98.69	14.972
120	90	0.3	99.01	0.99	19.809
120	90	0.4	29.097	47.066	27.256
120	90	0.5	98.871	1.398	24.243
120	100	0.1	99.318	88.398	9.775
120	100	0.2	90.129	100	30.812
120	100	0.3	94.034	93.247	32.904
120	100	0.4	93.252	93.626	25.811
120	100	0.5	70.998	100	16.261
120	110	0.1	49.81	88.839	93.8
120	110	0.2	85.635	83.503	13.229
120	110	0.3	24.327	68.246	81.135
120	110	0.4	82.7	24.187	37.471
120	110	0.5	53.396	50.675	20.35
120	120	0.1	10.241	8.957	71.271
120	120	0.2	96.175	91.892	25.15
120	120	0.3	29.15	86.873	11.898
120	120	0.4	82.891	78.578	75.978
120	120	0.5	79.329	39.409	46.055
120	130	0.1	97.025	53.046	0.235
120	130	0.2	89.683	76.764	26.209
120	130	0.3	55.046	53.278	40.643
120	130	0.4	84.184	0.908	30.115
120	130	0.5	32.417	16.888	31.303
120	140	0.1	69.715	21.923	28.071
120	140	0.2	100	100	16.605
120	140	0.3	70.733	34.588	30.572
120	140	0.4	30.491	29.775	32.161
120	140	0.5	35.976	13.944	24.087
120	150	0.1	100	21.733	12.774
120	150	0.2	93.256	14.154	18.466
120	150	0.3	91.197	45.279	18.711
120	150	0.4	94.227	38.182	24.321
120	150	0.5	95.06	41.149	15.456
120	160	0.1	79.304	54.711	16.999
120	160	0.2	96.936	1.147	13.067
120	160	0.3	99.811	47.151	16.432
120	160	0.4	68.737	80.325	15.59
120	160	0.5	65.537	9.474	19.803

Table C.3: Data obtained from optimiser - VSM locked at 120°

D Elbow joint tuned PID

Table D.1: Training data for Neural Network on Arduino

$\theta°$	**Load (g)**	K_P	K_I	K_D
30	60	1.16	1.589708405	0.04463448
45	60	1.26	1.550698152	0.1012473
60	60	1.04	0.391592066	0.045573957
75	60	0.54	0.489490083	0.036459166
90	60	0.24	0.302417582	0.03228225
30	75	0.5	0.86380402	0.03228225
45	75	1.5	0.952380952	0.1155
60	75	1.1	0.705806866	0.09428925
75	75	1.1	0.405588103	0.1098405
90	75	1.1	0.28038643	0.09547395
30	100	1.56	0.989533777	0.010821096
45	100	0.9	0.796460177	0.0839025
60	100	1.2	0.995437578	0.0954756
75	100	0.78	0.415252545	0.003223292
90	100	0.82	0.321990372	0.03445631
30	160	1.72	1.889071939	0.06890664
45	160	0.88	0.907450374	0.02503248
60	160	0.86	0.673519334	0.005798034
75	160	0.9	0.43956044	0.02027025
90	160	0.66	0.311982983	0.023037795
30	175	1.64	1.751785833	0.011518897
45	175	0.86	1.378945427	0.035399225
60	175	0.78	0.722556739	0.00326898
75	175	0.8	0.495547813	0.028413
90	175	0.6	0.33941451	0.0233343
30	200	1.5	2.070607723	0.01103355
45	200	1	1.180202109	0.003728175
60	200	0.78	0.682014917	0.002943807
75	200	0.7	0.479452055	0.038544
90	200	0.58	0.306270627	0.02416425