REAL-TIME ROBUST CONTROL OF A SERIES ELASTIC ACTUATOR SUBJECT TO UNCERTAINTIES

A Thesis

by

A. Talha KANSIZOĞLU

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Masters of Science

in the
Department of Mechanical Engineering

Özyeğin University
December 2018

REAL-TIME ROBUST CONTROL OF A SERIES ELASTIC ACTUATOR SUBJECT TO UNCERTAINTIES

Approved by:

Assistant Professor Barkan Uğurlu, Advisor
Department of Mechanical Engineering
Özyeğin University

Assistant Professor Özkan Bebek
Department of Mechanical Engineering
Özyeğin University

Assistant Professor Uğur Tümerdem
Department of Mechanical Engineering
Marmara University

Date Approved: December 24, 2018

To my family..

ABSTRACT

Humans are able to perform highly complicated tasks in their daily activities by interacting with the environment. However, for a robotic system, these tasks are quite challenging. To overcome these problems, robotic systems benefit from torque control approaches for enhanced environmental interaction capabilities. Furthermore, having a so-called ideal torque source at the joints may provide human-like movement functionalities for such systems. To that end, series elastic actuators (SEA) are highly preferred as a torque generator due to its numerous advantages such as low output impedance, high output torque bandwidth, and safety. A SEA consists of an electric motor, a reduction gear, and an elastic element. Having an elastic element between the motor and the output of the actuator makes the control problem of SEA highly complicated. Apart from modeling errors and non-linear disturbances, the environment is also unknown beforehand. An ideal actuator has to be robust against such uncertainties, and therefore, high fidelity control problem of SEAs is still an active research area. Hence, this thesis presents a comprehensive comparison study on various off-the-shelf advanced control methods. The study is supported via real-life experiments using the SEA unit that was designed in the Biomechatronics Laboratory of Ozyegin University.

ÖZETÇE

Günlük aktivitelerinde insanlar, çevreleriyle etkileşim halinde oldukları oldukça karmaşık görevleri yerine getirmektedir. Fakat bir robotik sistem için bu görevler birçok zorlukların aşılmasını gerektirmektedir. Robotik sistemler bu tür görevler için tork kontrol yaklaşımlarından oldukça fayda sağlarlar. Dahası, eklemlerinde ideal tork eyleyicilerinin bulunması robotik sistemlere insansı kabiliyetler kazandırması açısından önemlidir. Düşük çıkış empedansı, yüksek çıkış torku bant genişliği ve güvenli çalışma gibi avantajlara sahip olan Seri Elastik Eyleyici (SEE), tork üretici olarak yaygın olarak tercih edilmektedir. SEE yapısında bir elektrik motor, redüksiyon dişlisi ve elastik bir parça bulundurmaktadır. Bu elastik parça SEE'ların kontrol problemini oldukça karmaşıklaştırmaktadır. Modelleme hatalarının ve doğrusal olmayan bozucuların yanısıra, çevresel etkenler de önceden kestirilememektedir. Ideal bir eyleyicinin belirsizliklere karşı gürbüz olması beklenir, bu nedenle SEE kontrol problemi hala aktif bir araştırma konusudur. Bu tez, literatürde bulunan farklı gürbüz kontrol metodlarının performans karşılaştırmasını incelemektedir. Karşılaştırma çalışması Özyeğin Üniversitesi Biyomekatronik Laboratuvarında üretilen CoEx-SEA eyleyicisinin deneysel çalışmalarıyla desteklenmiştir.

ACKNOWLEDGEMENTS

I am profoundly grateful to Dr. Barkan Ugurlu, my thesis advisor who was always there with his support. This work would have been impossible without the contributions of my friend Mehmet Can Yildirim. I also would like to thank my thesis committee for their valuable time. My sincere thanks to the members of Biomechatronics and Robotic Laboratories of Ozyegin University. And finally, thanks to my friends and family for their massive support.

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK), with the project 215E138.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

SEA Series Elastic Actuator

DoB Disturbance Observer

HD Harmonic Drive

SSI Synchronous Serial Interface

PCB Printed Circuit Board

SPI Serial Peripheral Interface

PWM Pulse Width Modulation

SSH Secure Shell

GUI Graphical User Interface

CAD Computer Aided Design

RPi Raspberry Pi

BLDC Brushless DC Motor

RTOS Real-time Operating System

EM Electromagnetic Motor

CoEx-SEA Compliant Exoskeleton-SEA

RMS Root Mean Square

LPF Low-pass Filter

PID Proportional-Integral-Derivative

FF Feed-forward

RTOS Real-time Operating System

SMC Sliding Mode Controller

DF Differential Flatness

ZTC Zero Torque Control

CHAPTER I

INTRODUCTION

1.1 Torque-Controlled Actuators and Robots

In daily activities, humans perform various tasks that require physical interaction with the outer environment e.g. walking, running, manipulating objects and assembling. However, such tasks are usually challenging for robotic systems. While the robotic systems that adopts position control approaches are superior in the sense of performing pick and place, spray painting etc., they are unable to provide sufficient performance for applications that involve physical interactions[1]. The main reason is, controlling the contact forces between the robot and the environment is essential for accomplishing such tasks and this requires active force/torque control. Torque control can be achieved either by using a force/torque sensor at the end effector or by using torque sensing units at the joint level[2].

Force control problem of the robotic systems have been studied by researches since 1950s. The first applications of force control were remote manipulators [3]. Since then, force control research gained interest despite existing technical problems such as inaccurate force sensing, stability issues and bandwidth limitations which are mostly associated with actuation technology. To mention some of the pioneering works, mathematical modeling of the environment and controlling the end effector that is subject to infinite stiffness was studied to achieve the force control of a robotic arm[1]. In another example, to prevent the robot to interact with high stiffness and countering the ground forces, passive elements were used by Hirai et al. with P3 humanoid [4]. While useful in its own right, methods with no active torque control may be impractical to increase the capabilities of the robotic systems to the desired level.

Today, numerous torque-controlled robots are being developed such as human mimicking robotic hands: Dextrous Hand [5], humanoid robots: Robonaut and Valkyrie [6, 7], exoskeletons: MINDWALKER and LOPES [8, 9] and also as cobots and haptic devices. These systems usually rely on having an ideal torque actuator at their joints. Having an ideal torque generator at the joint level may provide human-like capabilities. On the contrary, ideal torque actuators do not exist in real-life applications. Nevertheless, synthesizing robust controllers for such actuators is possible. Since these systems involve in human interactions, robustness and safety are among the biggest concerns. Ideal torque actuators are expected to have low output impedance because back-drivability is vital for interactions and it is directly related to the impedance of the actuator. It also has to possess high-bandwidth control capability which indicates the highest frequency that the actuator can work at. Additionally, being able to counter the high impact due to instantaneous impact loads which may lead to instability or even damage the whole system.

Various actuation methods are currently available in torque-controlled robots with their own limitations and superiorities, leading to several trade-offs. Pneumatically actuated robots use pressurized air and servo valve systems to adjust the air flow; they operate at low costs and have high power to weight ratios with reduced friction compared to hydraulic systems[12]. However, especially for the mobile platforms, the constant need of pressurized air is a disadvantage and also may arise safety issues. Even so, due to its numerous advantages, pneumatic systems are used in many research studies especially in upper limb exoskeletons[10, 11]. Unlike in pneumatic systems, hydraulic actuators use pressurized oil instead of air. They exhibit high torque density, high bandwidth and short response times. These advantages make hydraulic systems suitable for the fast dynamic robotic systems such as legged robots [13, 14]. However, they have high impedance and nonlinear behavior, besides they are available at high costs and possess possible safety issues.

Unlike hydraulic and pneumatic actuators, electric motors with gear reduction were designed as stiff as possible to achieve the high precision position control tasks. However, force control applications suffer from high output impedance. To this end, Pratt et al. proposed a compliant actuation method with a series elasticity which is called Series Elastic Actuator (SEA). Next section will briefly explain the concept and main advantages of the SEAs.

1.2 Brief information on Series Elastic Actuators

A basic SEA design usually consists of an Electromagnetic Motor (EM), a reduction gear and an elastic element. The mechanical representation of the typical SEA can be seen in Figure 1. A great majority of EMs operate at high velocities and low torques. Therefore, reduction gears are used to increase their torque output. While it is a common practice to use a reduction gear to increase the output torque of EMs, it introduces some disadvantages such as friction and backlash[2]. Moreover, the reflected inertia is increased by the square of the gear ratio, thus, external torques are much more reflected and may damage the system. These facts compelled researchers to consider a new design approach. The use of series elasticity reduces the reflected inertia and it introduces certain advantages such as being able the counteract peak torques, possibility of storing the energy and high-fidelity force control[15].

The closed-loop force control of SEAs can be achieved by controlling the spring position due to a linear relation between spring deflection and the output torque. Therefore, the force control problem can be converted into a position control problem and without using any force sensors low cost and fine force measurement is achieved by measuring the spring deflection. The general advantages of the SEAs can be summarized as follows,

- low output impedance,

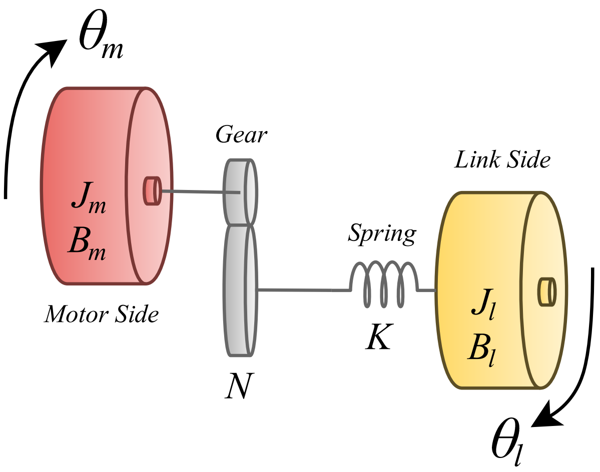


Figure 1: A simplified mechanical representation of a SEA

- high output torque control bandwidth,

- back drivability

- improved shock tolerance,

- energy storage,

- safety and dependability.

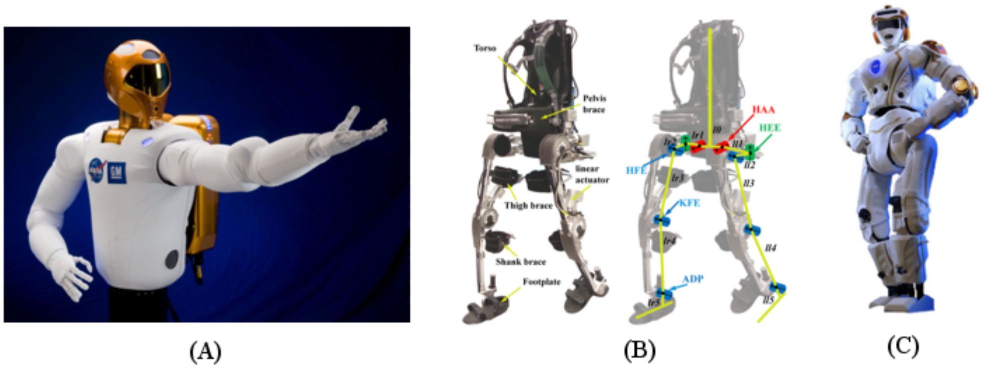


Figure 2: (A) NASA Robonaut [6], (B) MINDWALKER [8] (C) Valkyrie [7]

There are numerous robotic applications that use SEAs, e.g., Robonaut [6], Valkyrie [7], MINDWALKER [8] and LOPES [9]. While the concept of SEAs are essentially the same, different design approaches are proposed over the last two decades. A lower-extremity power exoskeleton LOPES uses SEA design that contains two bowden cables while Sariyildiz et al. was designed multiple springs to increase the control bandwidth. Moreover, a Compliant Exoskeleton-SEA (CoEx-SEA) was developed at the Biomechatronics Laboratory of Ozyegin University[16]. Thus mechanical design and the modeling of the SEAs will be discussed at Chapter II.

From the mechanical point of view, series elasticity augments the capabilities of EM based actuators. However, without robust controller methods, SEAs suffer from poor torque tracking performance. Therefore, supporting the design with robust controllers is a necessity and this issue will be discussed in the following section.

1.3 Control Problem of Series Elastic Actuators

Compliant actuators possess superiorities over stiff actuators, yet they evidently require complicated control methods. SEAs can be modeled as two inertiae systems, considering the motor side and the load side. The interaction between the motor and the load side occurs via a spring. The interaction causes spring to deflect and the deflection can be measured by means of encoder readings and converted as torque. However, high order dynamics of the SEAs requires advanced controllers instead of conventional methods.

In earlier studies, the torque control method of SEAs was kept relatively simple. Pratt et al. proposed a conventional Proportional-Integral-Derivative (PID) control to achieve torque tracking. However, the torque control performance was deemed insufficient, later they additionally implemented a Feed-forward (FF) term to cancel out the unmodeled dynamics [21]. In another example, a similar PD and feed-forward control scheme was presented in [24] for elastic joints. Yet, nonlinear disturbances of the system may degrade the overall performance, and therefore, these methods may not exhibit robust tracking.

Wyet proposed a cascaded control algorithm with an inner loop velocity control and outer loop torque control to minimize the nonlinearities such as backlash and stiction [22]. This method requires relatively less computational resources and allows higher sampling rates. It is claimed that, internal and external disturbances can be suppressed by using fast inner velocity control. By the same token, Heike el at, studied the stability of this cascaded control method [23].

The environment and the load dynamics of the SEAs highly vary. To ensure robustness, observer-based control methods are required to suppress the disturbances. SEAs are subjected to various disturbance sources, such as, unknown environment impedence, modeling uncertainties, nonlinearities such as friction and backlash and disturbances directly acting on the spring. To this end, numerous researchers implemented model based Disturbance Observer (DoB)s in their designs.

In [25], Oh and Kong proposed a robust control method based on a conventional PID controller with a model based feedback term and a DoB. The major difference was, DoB was applied to output torque directly instead of applying to motor side. Hence, this method provided precise torque control.

Sliding Mode Controller (SMC) was implemented on SEAs and torque control applications were investigated by numerous researchers. Due to its discontinuous nature, SMC successfully suppresses disturbances and known to be suitable for motion control systems. In [26], SMC based SEA torque control method was proposed and chattering problem was reported. To overcome this chattering phenomena, Sariyildiz et al. proposed a DoB based SMC algorithm to reduce the chattering[27].

Sariyildiz et al. also proposed a robust force controller design based on differential flatness and a higher order disturbance observer scheme[20]. They claimed that without knowledge of precise system parameters, the active disturbance rejection algorithm exhibits superior tracking performance.

Furthermore, in the literature more advanced controller methods are investigated such as Model Predictive Controller, Resonance Ratio Control etc. However, design and implementations of such controllers are complex and computationally expensive.

In this thesis study, several control methods were implemented and experiment results are presented for comparison purposes. Detailed controller schemes can be found at Chapter III.

1.4 Thesis Contribution

This thesis study essentially refers to the control problem of the CoEx-SEA that was designed in the Biomechatronics lab of Ozyegin University. Attaining the best performance of a SEA largely depends on its control structure, to this end, the main aim of this study is guiding the reader through choosing the superior control algorithm for a SEA. A comprehensive comparison study of various control schemes was the main focus of this thesis, comparison study is supported with an experimental study, controller performances were compared based on their Root Mean Square (RMS) error of reference tracking.

CHAPTER II

HARDWARE IMPLEMENTATION

2.1 Mechanical Hardware Implementation

This section briefly introduces the hardware implementation of CoEx-SEA that was designed as a torque source for the lower-limb exoskeleton joints developed at the Biomechatronics Laboratory of Ozyegin University. Human-wearable devices such as lower body exoskeletons are safety critic applications where an unexpected behavior may result in major problems. Therefore, the mechanical design of the CoEx-SEA was studied exhaustively for a long-lasting life duration. The mechanical components can be listed as follows,

I. Frameless Brushless DC Motor (BLDC) (Kollmorgen TBM-7631)

II. Reduction Gear (Harmonic Drive CSG-25, 1:100),

III. Custom Built Torsional Spring

IV. High Resolution encoders (Broadcom AS38-H39E-S13S)

The Computer Aided Design (CAD) figure of the CoEx-SEA can be seen in Figure 3. The BLDC motor was placed inside a custom built frame and directly connected to the input of the Harmonic Drive (HD). The output of the HD is connected to the spring at the inner circle. The outer radius of the spring is directly connected to the output cap of the actuator. In the experiment setup, a steel link was connected to the output cap as a mechanical load.

Design of the elastic element is crucial in the sense of overall performance. To this end, Yildirim et al. proposed an integrated design method for a SEA [16], the

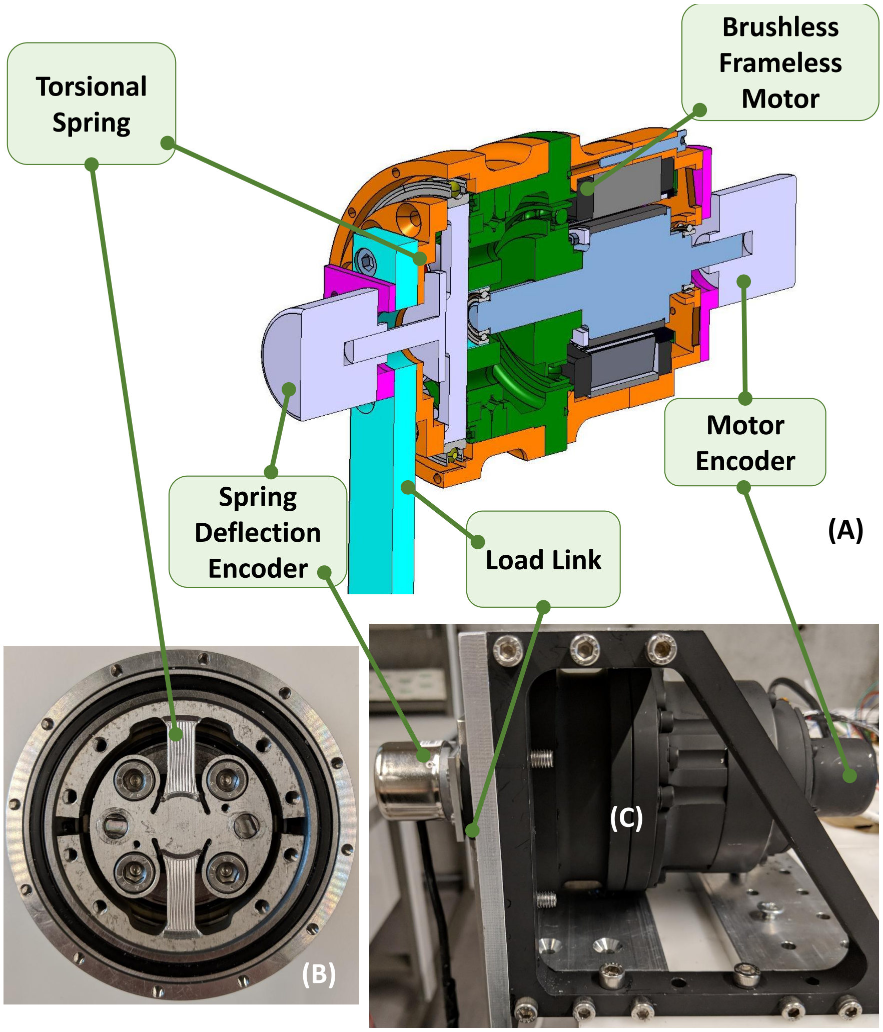


Figure 3: (A) CAD Data, (B) Torsional Spring (C) An image of CoEx-SEA [35]

torsional spring design can be seen in Figure 4 while CoEx-SEA parameters can be seen in Table 1.

Another important aspect of the design process is the sensor location. Non-collocated sensor positioning may lead to instability [15], therefore the deflection is directly measured.

Moreover, a load-cell based solution is an option for measuring the force but measuring the deflection argued to be more advantageous in the sense of simplicity of the

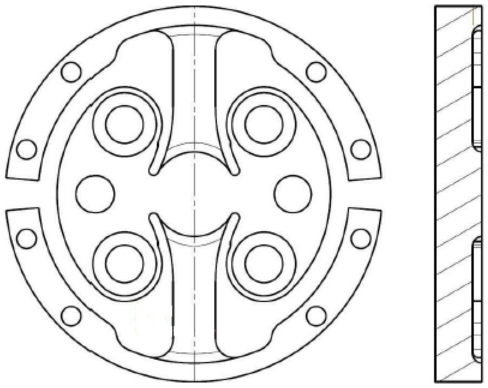


Figure 4: Torsional spring drawing [35]

implementation process [17]. In response to that matter, CoEx-SEA has two high-resolution encoders for measuring the motor angle and the deflection. The encoder shaft is connected to the center of the spring while the encoder itself is connected to the output cap, directly. Non-contact and lightweight encoders are commonly used in SEA designs, however mechanical implementation of ring type encoders requires precise mounting and highly affected by mechanical design constraints. As a consequence, Avago encoders were chosen to tackle these problems.

Table 1: CoEx-SEA specifications

Parameters	***Value***	***Unit***
Max. Angular Velocity	2,77	rad/s
Max. Continuous Torque	164	Nm
Peak Torque	460	Nm
Weight	3.4	kg
Dimensions (r x L)	53.5 x 132	mm x mm
Stiffness	91	Nm / deg
Torque Resolution	3.90	mNm

2.2 Electronic Hardware Implementation

The electronic hardware implementation of the one-degree-of-freedom (DoF) actuator controller set-up consists of three major parts; motor driver electronics, encoder electronics, and the computational unit. As a computation unit, a Raspberry Pi (RPi) was used. RPi is a product of Raspberry Pi Foundation and offers low-cost computing solutions. For this set-up, RPi meets the requirements for the following reasons,

- has relatively high computational capabilities,
- has required general purpose input-output pins,
- can run Real-time Operating System (RTOS),
- is available at low cost and portable size.

Precise torque control applications benefit from high-resolution measurements. Therefore, in the design, 23 bit Multi-turn and 16 bit Single-turn Broadcom Avago encoders was implemented. Avago encoders are widely used in industrial applications, therefore it inherits commonly used communication protocol called Synchronous Serial Interface (SSI). On the contrary, RPi does not have any SSI ports like most of the other embedded systems. SSI communication requires four wires apart from the supply voltage lines. These four lines are known as differential signals which means that, the logic data is transferred by the difference of voltage values in the lines for the noise immunity. Therefore, a Printed Circuit Board (PCB) is designed to overcome this problem; see Figure 5. To be able to read data from the encoders, Maxim MAX90 line drivers were used. While the line driver converts the differential signal into a single transmission line, a multiplexer was used to control the data flow of the encoders. The logic level converters were used to adjust the logic level to 3.3V which is compatible with Raspberry Pi. Additionally, the PCB has a general purpose

Serial Peripheral Interface (SPI) port, a regulated 5V output, a Pulse Width Modulation (PWM) output and encoder supply ports. In other words, the SSI encoders are converted into SPI protocol with data, clock, and chip select lines. SPI protocol is commonly used in micro-controllers and also available in RPi. The data transmission can be adressed by using standard SPI communication libraries.

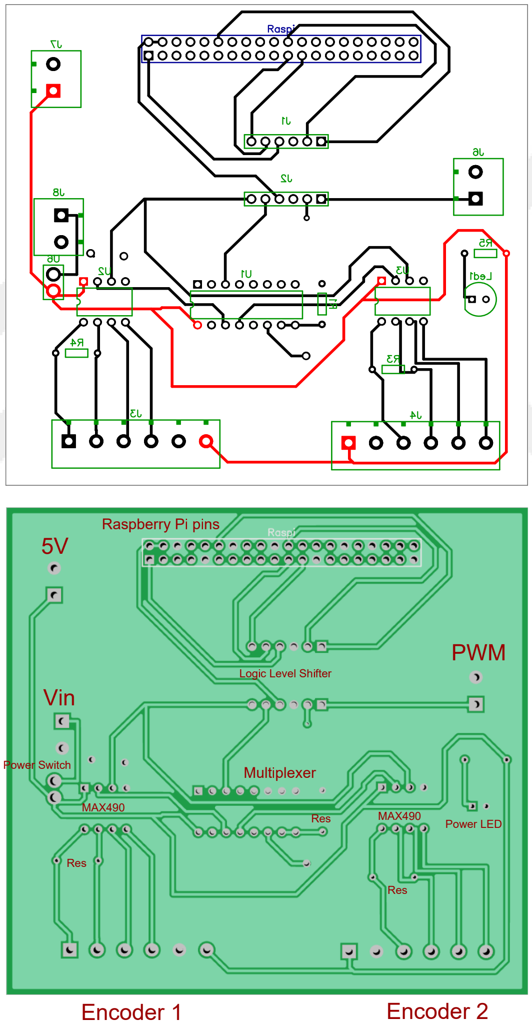


Figure 5: PCB design

To drive the brushless DC motors, a MAXON 70/10 motor driver was used in the set-up. Using a micro USB port, the motor driver can be programmed via the host PC. In PC, ESCON Studio software, a product of MAXON company, offers user-friendly controller configuration options. The motor parameters can be seen in

Table 2.

Table 2: Brushless DC motor parameters

Parameters	***Value***	***Unit***
Speed Constant	55.5	rpm/V
Thermal Time Constant	540	s
Number of Pole Pairs	6	-
Max Permissible Speed	2300	rpm
Nominal Current	10	A
Torque Sensitivity	0.172	Nm/A
Design Voltage	48	V

MAXON 70/10 motor driver has three motion control modes: open-loop speed, closed-loop speed, and current control mode. In the experiments, the current control mode was used. The controller gains can be adjusted by using potentiometers located on the driver. The controller can be tuned by visualizing the actual current versus the desired current by using ESCON Studio or by using analog outputs of the motor driver.

The motor driver accepts digital Pulse Width Modulation (PWM) and analog signal commands. For the compatibility both methods were covered in the design. To be able to send analog commands to the motor driver, 16-bit DAC8532 digital-to-analog converters were used. RPi has also a built-in PWM output. However, PWM signals may introduce noisy motor commands which could degrade the torque control performance, and therefore analog commands were used.

The PCB, digital-analog converter module, motor driver and the RPi can be presented as a single SEA controller electronic device. This device uses the ethernet port to communicate with the host computer. The host computer uses Secure Shell (SSH) protocol to program the SEA controller. Experiment data can be collected by using SSH protocol and visualized in the MATLAB environment. The block diagram of the SEA controller can be seen in Figure 6.

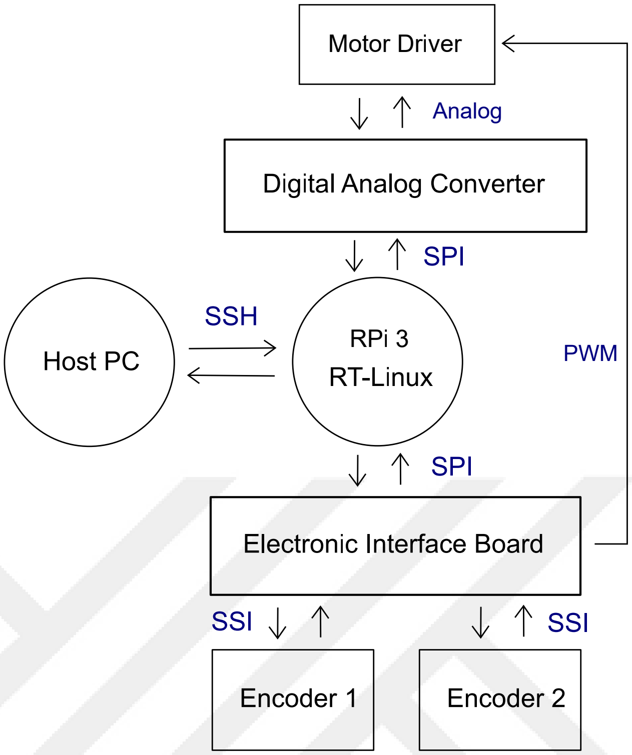


Figure 6: The block diagram of the SEA controller hardware implementation

Programming steps can be summarized as follows,

- robust torque control algorithm is implemented as a RTOS compatible C program,
- by using SSH, the host PC sends the ready-to-compile program to the SEA Controller and executes compiling process,
- the host PC runs the program and program outputs an experiment log file
- experiment log file is collected via SSH and visualized in MATLAB.

2.3 Real-Time Operating System and Programming

To ensure determinism and maintain the exact sampling time between two controller cycles, a RTOS was implemented to the joint-level SEA control units. Since the sampling rate of the controllers directly affects the bandwidth and even the stability, sampling rate should be as high as possible and this requirement needs to be handled accordingly. At this point, it is also important to understand that RTOS does not mean faster execution of programs.

To this end, a real-time patch maintained under the name of PREEMPT-RT was applied to the Linux kernel of the official Raspbian operating system that was provided by the Raspberry Pi foundation. The main idea of the PREEMPT-RT patch is making the kernel fully preemptive. Preemptive kernel prevents the OS from interrupting the controller algorithm because such interruptions may lead to large latencies.

This procedure has the following steps: i) downloading the Linux kernel for Raspbian OS, ii) applying the PREEMPT-RT real-time patch, the cross-compiling patched kernel on a different computer, iii) installing the kernel to the Raspbian OS.

Downloading Linux Kernel: Raspberry Pi foundation offers various Linux distributions on their official website. Compatible kernels for these distributions can be obtained from the kernel source tree of RPi.

Applying PREEMPT-RT Patch: PREEMPT-RT patch can be obtained from Linux Foundation for a compatible RPi kernel version. This patch can be downloaded and applied by using conventional kernel patch procedures. After a successful patching, the kernel has to be configured as a fully preemptive kernel, enabling high-resolution timer support by using ”menuconfig”.

Cross-compiling patched kernel and installing: Since kernel compiling process requires very long compiling time on a RPi due to its computational limits, the kernel can be compiled in a host PC. Cross-compiling of the kernel is well documented on the official Raspberry Pi Foundation website. Generated kernel images can be uploaded to a RPi by using SSH commands or directly through the internet.

Since the operating system being real-time compatible alone is not sufficient to achieve the real-time operating conditions also the control program has to be real-time compatible. RT programs are expected to have the following characteristics [18]

- The system time can be obtained by using different modes in Linux OS. Monotonic clock mode (CLOCK_MONOTONIC) is a relative clock returns the time passed since OS is booted. Thus, this prevents any other application to set the system time.

- Stack memory size is recommended to inform to the RT application explicitly to prevent page faults. Further explanation on page faults can be found at the official Linux Foundation Website [19].

- RT application has to be declared as an high priority task.

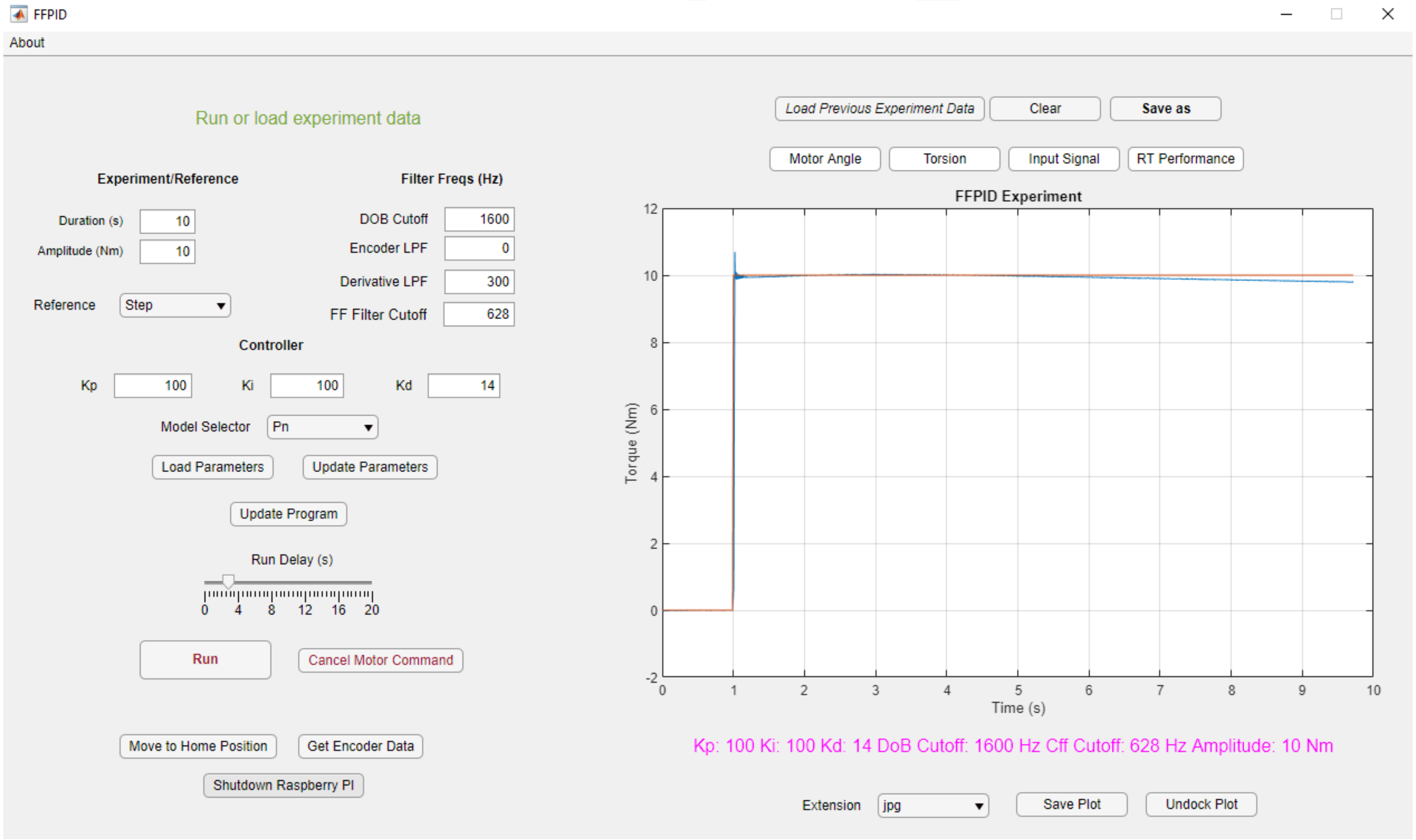
- Memory locking has to be performed at the beginning of the task. Thus, this prevents page faults from degrading the deterministic execution during the RT cycles.

- Delaying can be achieved by using clock_nanosleep() function of the API for precision.

SEA Electronics were integrated with RPi using SEA library. This library includes hardware-related functions, e.g., readEncoders() and motorWrite(). Avago encoders are able to store 39 bits of data in their memory, therefore in each cycle, 40 bits of data must be transmitted to RPi. By using standard SPI protocol functions, 40-bit data can be read via 5 subsequent transmissions of 8 bits. This 8-bit data package has to be combined and converted into meaningful measurement. As we have two encoders in the CoEx-SEA, these steps have to be repeated twice. Furthermore the digital-analog converter uses SPI protocol to communicate; in each cycle 2 transmissions of 8 bits are required. Therefore, the low-level hardware is handled by a library that is available for any controller implementation. Program codes can be found in the Appendix B6.

As mentioned in the previous section, controller algorithms were implemented by using C language. These programs need to be sent to RPi, compiled and executed. The experiment data must be retrieved. Since this process is not ideal for a quick prototyping, a Graphical User Interface (GUI) was designed by using MATLAB's AppDesigner to assist the user during the implementation process; see Figure 7.

This GUI essentially executes system commands that contain SSH protocol, e.g. updating the controller software, canceling the program, retrieving the data, visualizing the data and so on.



Duration (s) 10

Amplitude (Nm) 10

Reference Step

DOB Cutoff 1600

Encoder LPF 0

Derivative LPF 300

FF Filter Cutoff 628

Kp 100 Ki 100 Kd 14

Model Selector Pn

Kp: 100 Ki: 100 Kd: 14 DoB Cutoff: 1600 Hz Cff Cutoff: 628 Hz Amplitude: 10 Nm

Extension jpg

Figure 7: GUI for the SEA Controller

CHAPTER III

MODELING OF A SERIES ELASTIC ACTUATOR

As can be seen in Figure 8, a SEA is represented as a two-inertia system connected via a torsional spring. While the motor cylinder of the actuator is the first inertia, the link can be considered as the second inertia. A reduction gear that increases the output torque of the motor is placed between the motor side and torsional spring.

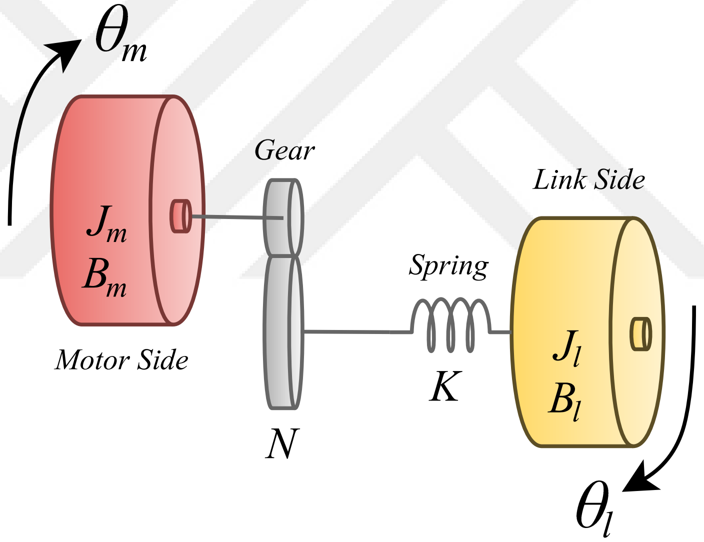


Figure 8: Mechanical representation of SEA

Since the output torque of the actuator is directly related to the spring deflection, the torque control problem of the series elastic actuator can be interpreted as a deflection control problem. Therefore the system states can be chosen as the motor and link angles and their respective velocities, while the model output is the deflection.

The link torque is

$$\tau_d = K\theta_d = K\left(\frac{\theta_m}{N} - \theta_l\right) \tag{1}$$

where τ_d is link torque, K is the spring constant, N is the gear ratio, θ_m, θ_l and θ_d are the motor, link, and deflection angles and $\dot{\bullet}$ and $\ddot{\bullet}$ represents the first and second derivatives, respectively.

The dynamic equations of the SEAs are as follows,

$$J_m\ddot{\theta}_m + B_m\dot{\theta}_m = \tau_m - \frac{\tau_d}{N} - d_1 \quad , \tag{2}$$

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l = \tau_d - d_2 \quad , \tag{3}$$

where J_m, J_l, B_m and B_l are inertiae and viscous friction coefficients of motor and link side, respectively. τ_m and τ_d are motor torque and link torque. The internal and external disturbances acting on the system are defined as d_1 and d_2. Apart from the disturbances directly acting on the spring and on the link side, the modeling uncertainties can also considered as disturbances. If we consider the load side dynamics,

$$(J_n + J_u)\ddot{\theta}_l + (B_n + B_u)\dot{\theta}_l = \tau_d - d_2 \quad , \tag{4}$$

where J_n and B_n are nominal inertia and viscous friction, J_u and B_u are the modeling uncertainties within inertia and viscous friction, these uncertainties can be included in d_2 as,

$$d_2 = J_u\ddot{\theta}_l + B_u\dot{\theta}_l + K_u\theta_d + d_{other} \quad , \tag{5}$$

in which θ_l and θ_d are the link angle and torsion, K_u is the uncertainty of the torsional spring while d_{other} represents any other disturbances such as unmodeled dynamics.

The mathematical model of the SEA can be represented by using a transfer function written in Laplace domain or in a state space form. While they both give the same system responses evidently, they differ in the sense of controller design procedures. In the next sections, modeling in Laplace domain and state space form is presented, respectively.

3.1 System Model in Laplace Domain

Using block diagrams, a SEA can be represented as displayed in Figure 9. Since we have a two-inertiae system, motor side and link dynamics can be represented by using two distinct transfer functions as shown below,

$$P_m = \frac{\theta_m(s)}{\tau_m(s) - KN^{-1}\theta_d(s)} = \frac{1}{J_m s^2 + B_m s} \tag{6}$$

P_m represents the motor side transfer function with the input-output relation of motor torque and the effect of output torque τ_d to the motor angle. By the same token,

$$P_l = \frac{\theta_l(s)}{K\theta_d(s)} = \frac{1}{J_l s^2 + B_l s} \tag{7}$$

P_l represents the link side transfer function with the input-output relation of output torque to the link angle.

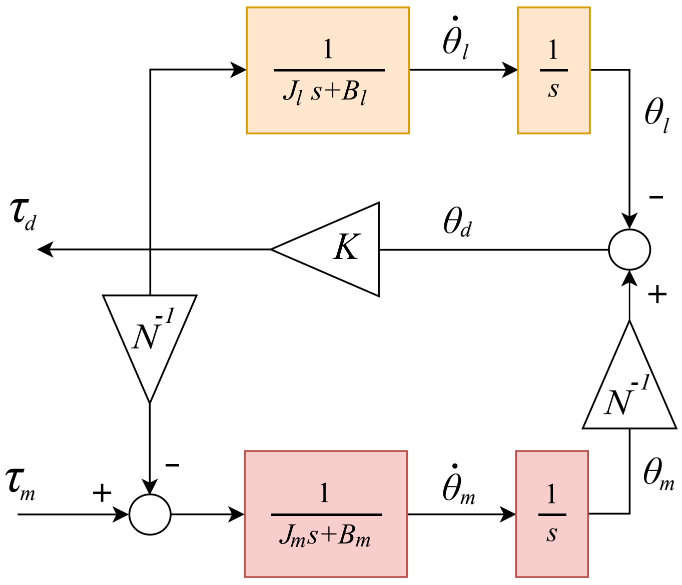


Figure 9: Block diagram of a SEA

To obtain the transfer function from the motor torque τ_m to the deflection θ_d, we can simply rewrite the relationship between the deflection and the output torque as follows,

$$\theta_d(s) = \theta_m(s)N^{-1} - \theta_l(s) \quad . \tag{8}$$

By substituting eq. (6) and (7) into (8),

$$\theta_d(s) = \frac{\big(\tau_m(s) - KN^{-1}\theta_d(s)\big)P_m}{N} - K\theta_d(s)P_l \tag{9}$$

In (9), the nominal model of the system can be represented by a single transfer function as below,

$$\theta_d(s) = \frac{P_m N^{-1}\tau_m}{1 + KN^{-2}P_m + KP_l} \tag{10}$$

Furthermore, by replacing P_m and P_l with model parameters, the 4th order transfer function can be obtained with system parameters as below,

$$\frac{\theta_d(s)}{\tau_m(s)} = \frac{N(J_l s^2 + B_l s)}{C_1 s^4 + C_2 s^3 + C_3 s^2 + C_4 s} \tag{11}$$

where, C_1, C_2, C_3 and C_4 can be expressed as in the following.

$$C_1 = (J_m J_l N^2) \tag{12}$$

$$C_2 = (J_m B_l N^2 + J_l B_m N^2) \tag{13}$$

$$C_3 = (B_m B_l N^2 + KJ_l + KJ_m N^2) \tag{14}$$

$$C_4 = (KB_l + KB_m N^2) \tag{15}$$

.

3.2 System Model in State Space

The dynamics of a general linearized system can be represented in a state space form as follows,

$$\begin{aligned} \dot{x} &= Ax + Bu + Fd \\ y &= Cx + Du \end{aligned} \tag{16}$$

In (16), A is the state matrix, B is the input matrix, C is the output matrix, D is feed-through matrix and F is the disturbance matrix. x is the state vector and the states of the system can be chosen as below,

$$x^T = [\theta_m \quad \dot{\theta}_m \quad \theta_l \quad \dot{\theta}_l] \tag{17}$$

The output vector is y; to obtain the deflection as an output, C matrix can be written as follows

$$y = Cx = [N^{-1} \quad 0 \quad -1 \quad 0]x \quad . \tag{18}$$

If we rewrite the dynamic equations in as in (19),

$$\ddot{\theta}_m = -\frac{K}{J_m N^2}\theta_m - \frac{B_m}{J_m}\dot{\theta}_m + \frac{K}{J_m N}\theta_l + \frac{\tau_m}{J_m} - \frac{d_1}{J_m} \quad , \tag{19}$$

and in (20),

$$\ddot{\theta}_l = \frac{K}{J_l N}\theta_m - \frac{B_l}{J_l}\dot{\theta}_l - \frac{K}{J_l}\theta_l - \frac{d_2}{J_l} \quad , \tag{20}$$

the state matrix A can be obtained as in the following,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K}{J_m N^2} & -\frac{B_m}{J_m} & \frac{K}{J_m N} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{K}{J_l N} & 0 & -\frac{K}{J_l} & -\frac{B_l}{J_l} \end{bmatrix} \tag{21}$$

The input vector of the SEA is,

$$b^T = \begin{bmatrix} 0 & \dfrac{1}{J_m} & 0 & 0 \end{bmatrix} \quad , \tag{22}$$

and as can be seen from the dynamic equations, disturbances d_1 and d_2 has an effect on the motor angle and link angle respectively by

$$f^T = \begin{bmatrix} 0 & \dfrac{d_1}{J_m} & 0 & \dfrac{d_2}{J_l} \end{bmatrix} \tag{23}$$

3.3 Environment Dynamics

The interaction of the SEA with an enviroment can be modeled with a spring-damper system as can be seen in Figure 10, where K_{env} represents the environment stiffness while B_{env} represents the damping of the environment, τ^L_{dist} and τ^S_{dist} represents disturbances acting on the link and the spring, respectively. Since the actuator interacts with the environment through the link, environment dynamics can also be included in the link dynamics such as,

$$P_l = \frac{\theta_l(s)}{K\theta_d(s)} = \frac{1}{J_l s^2 + (B_l + B_{env})s + K_{env}} \tag{24}$$

It can be concluded that, as the environment stiffness approaches to the infinity, which corresponds to the very stiff enviroment, the link movement is completely restricted. On the contrary, without any environment dynamics, actuator may not generate the desired torque due to lack of opposing torque with very stiff torsional springs.

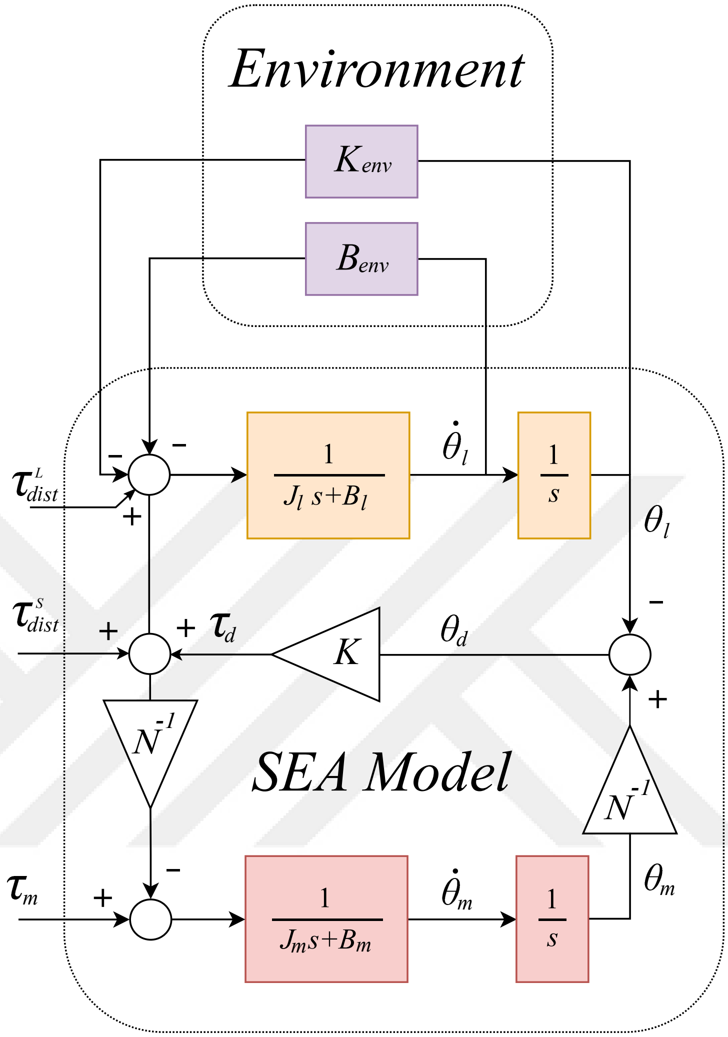


Figure 10: The SEA model with environment interaction

3.4 System Identification of the Motor Side

The system parameters of CoEx-SEA can be calculated theoretically via CAD data of the design. Yet CAD based estimations are not always accurate due to uncertainties in the mechanical manufacturing process. Determining the system parameters increases the overall performance of controllers. Therefore, motor parameters can be identified by applying system identification procedures to the motor side.

To identify the system parameters, stiction compensation is implemented to reduce the static friction. To this end, the link side of the actuator was detached, the ramp current input was fed to the system with different signal slopes, motor angle and velocity was measured by using the motor side encoder all respectively. In the controller program, a disturbance observer (DoB) to the motor side was implemented. By plotting the disturbance with respect to motor velocity, the friction and stiction curves of the actuator were obtained.

The DoB method was proposed by Ohnishi et al [36]. It is a method for estimating the disturbances acting on the system by using the nominal plant model. By using the CAD data, the motor nominal model is constructed. The motor side DoB block diagram can be seen in Figure 11 where $\dot{\theta}_m$ is motor velocity, J_m is the motor inertia, τ_m is motor torque and g is the DoB frequency.

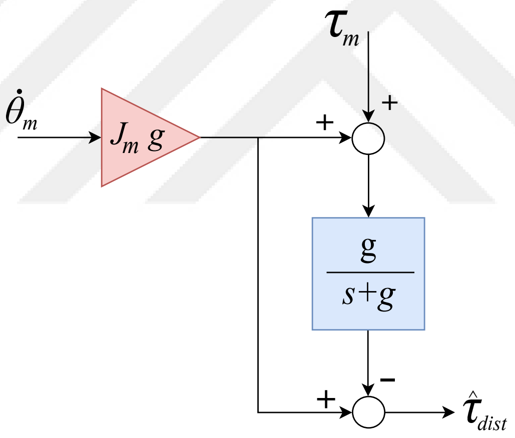


Figure 11: Block diagram of the DoB applied to the motor side

Uncompensated curve can be seen in Figure 12 by plotting the frictional disturbance with respect to angular velocity. Four experiments were conducted with different ramp input slopes, while 0.07395 Nm torque was enough for positive direction, 0.06982 Nm is required for negative direction to compensate the stiction. The stiction compensation was implemented by using the following algorithm and the compensated curve can be seen in Figure 13.

$$Stiction \quad compensation \begin{cases} +0.07395 & if \quad \dot{\theta}_m > 0 \\ -0.06982 & if \quad \dot{\theta}_m < 0 \\ 0 & if \quad \dot{\theta}_m = 0 \end{cases}$$

MATLAB System Identification Toolbox was used to identify the motor parameters. To identify the plant, the motor was fed with sinusoidal current signals of various frequencies without violating the velocity saturation limit. For the simplification of the fitting process, instead of the motor angle, identification was carried out by using the motor velocity of 11 different experiments. As a validation dataset, a chirp signal was fed to the motor to validate the identified model and the results are presented in a bode plot which shows the relation between the real experiment data and the simulation of identified parameters; see Figure 14. As the result, we were able to obtain a system that responses to the mathematical models we developed earlier in this chapter.

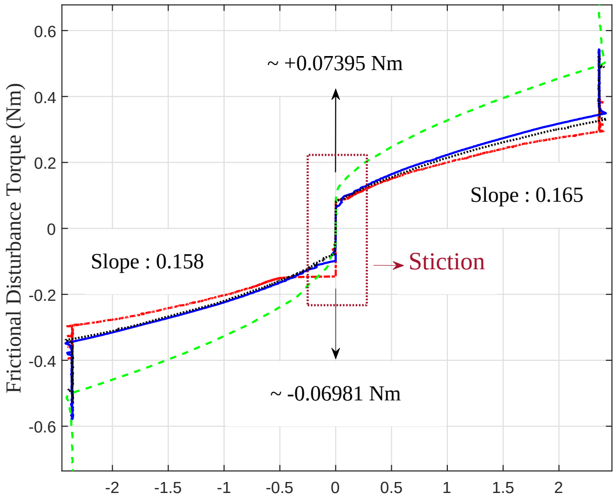


Figure 12: Uncompensated motor frictional torque curve (Given current slopes; red dotted-dashed line: 0.00002 A, black dotted line: 0.0001 A, blue line: 0.0002 A and green dashed line:0.002 A .)

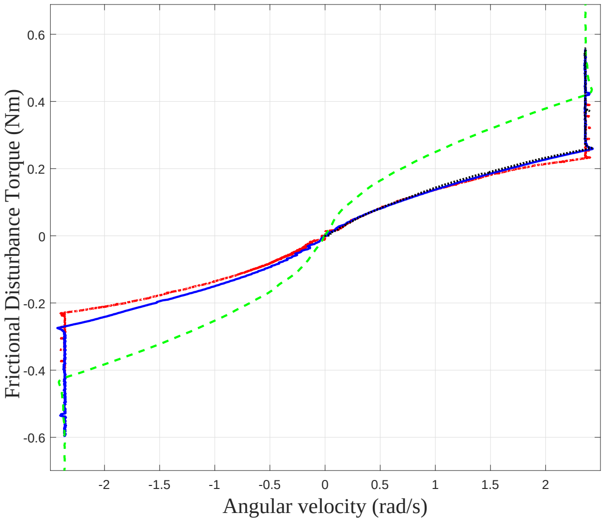


Figure 13: Compensated motor frictional torque curve (Given current slopes; red dotted-dashed line: 0.00002 A, black dotted line: 0.0001 A, blue line: 0.0002 A and green dashed line:0.002 A .)

Table 3: CoEx-SEA parameters

Parameters	***Explanation***	***Value***	***Unit***
J_m	Motor Inertia	2.781e-04	kg m^2
B_m	Motor Viscous Friction	1.5e-03	Nm s / rad
K	Spring constant	5000	Nm / rad
N	Gear ratio	100	-
J_l	Link Inertia	0.07	kg m^2
B_l	Link Viscous Friction	0.005	Nm s / rad

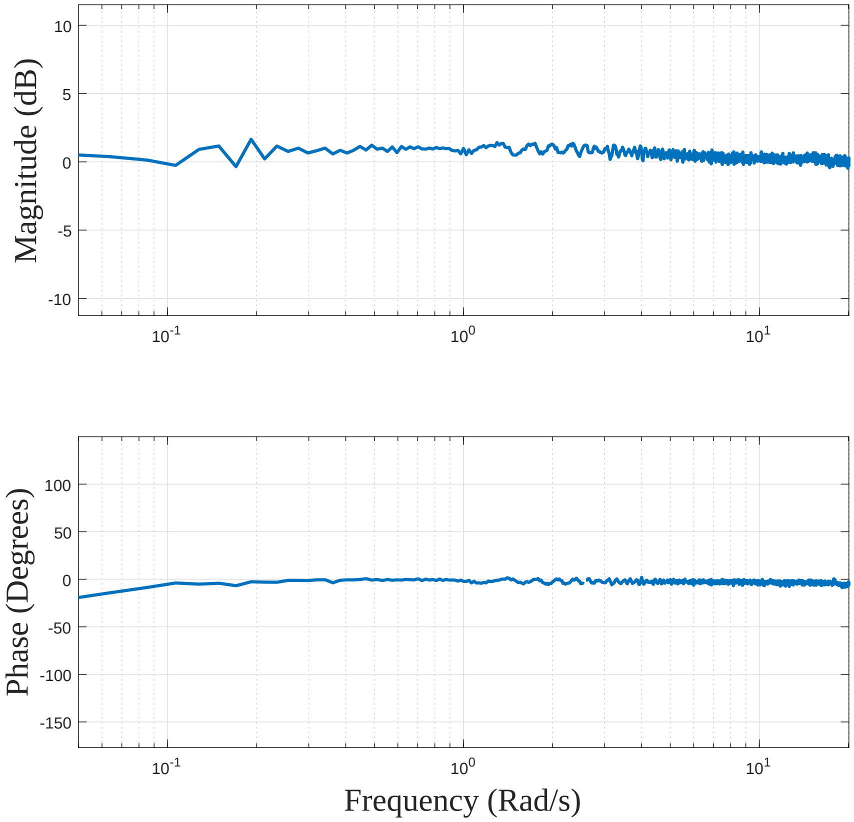


Figure 14: Frequency analysis of the real experiment and simulation data of the identified parameters

CHAPTER IV

ROBUST CONTROL OF SERIES ELASTIC ACTUATOR

Robust and high fidelity control problem of SEAs is still an active research area. Researchers proposed miscellaneous controller schemes to enhance the actuator performance. The following controllers were chosen due to their advanced characteristics. Majority of the following controllers contains a DoB to suppress the disturbances acting on the actuator. Block diagrams of the controllers are given, for the equation based controllers derivation of the equations with minor modifications are also presented.

While the standalone PID Controller is not performing sufficiently, Cascaded PID controllers are widely used in SEA implementations. To amplify its performance, Cascaded PID controller was also analyzed with a DoB. PID controller with a model based feed-forward term and a DoB, sliding mode controller with a DoB and the differential flatness controller with a DoB are state-of-the-art control methods that are studied in the scope of this thesis due to their advanced capabilities.

4.1 PID Controller

The Proportional-Integral-Derivative (PID) Controller is the most common feedback controller that is widely used in industry as well as in scientific research. Over the years, the popularity of the PID controllers increased because of its simplicity and effectiveness. The tuning process can be achieved either intuitively or by using practical tuning methods. The block diagram of the PID controller can be seen below in Figure 15.

The output of the PID controller includes the proportion, integral and the derivative combination of the error which is the difference between the reference torque and

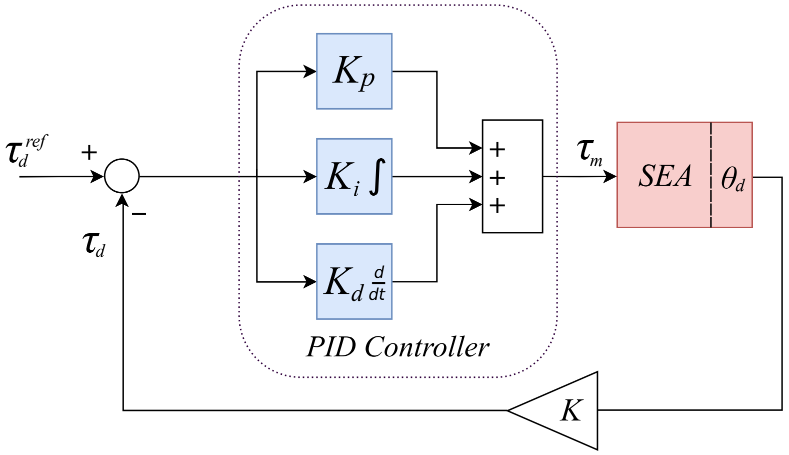


Figure 15: Block diagram of the Proportional-Integral-Derivative Controller

the output torque of the actuator multiplied with particular gain values. The proportional term increases the speed of system response while the integral term sums up the error variable and eliminates the steady-state error. Derivative term is directly related to the error changing rate and decreases the overshoot and oscillations[34].

The PID controller was applied to SEA by Pratt et. al. in earlier studies [15]. However, researchers argued the inconvenience of a stand-alone PID controllers for various reasons. In earlier works, they used a first order Low-pass Filter (LPF) instead of an integral term to ensure the stability[23]. However, the controller suffered from the nonlinear disturbances. Moreover, researchers combined a PID controller and a feed-forward term to cancel the unmodeled dynamics[21]. Another drawback of stand-alone PID controller is that controller gains cannot be increased freely due to measurement noise and system dynamics. Therefore the robustness of the controller is affected by these limitations.

Thus, the simulation and experiment of stand-alone PID controller is provided only as a step response to address the incommodiousness.

4.2 Cascaded PID Controller

The term cascaded controller usually refers to multiple control loops which the output of one generates a reference for the other. Cascaded controllers are believed to improve the stability and disturbance rejection property of the system and have been used in force control problem of SEA in [15, 22].

In [23], Heike et al. presented a detailed analysis on the stability of cascaded control schemes. It is claimed that with the high-frequency inner velocity loop, the outer loop force control exhibits superior tracking results. The cascaded controller configuration can be seen in Figure 16. In this control configuration, the outer loop feeds back the output torque of the actuator to the PID controller. The output of the PID controller generates the motor velocity reference and the PI controller achieves the velocity control at the inner loop with respect to calculated velocity reference.

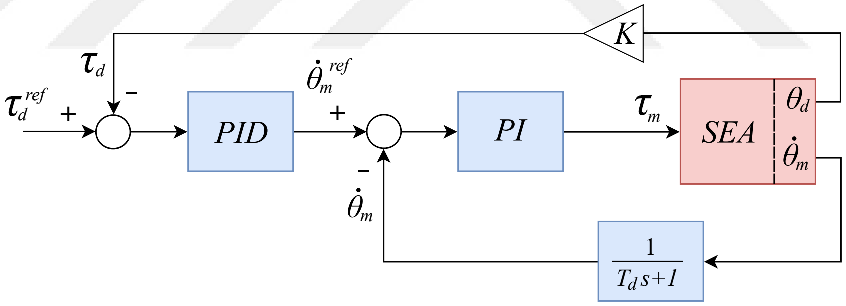


Figure 16: Block diagram of the Cascaded control with inner velocity loop

In [23], the controller stability analysis was presented based on the passivity of the system. Passivity theorem says the interaction of two passive systems is stable[28]. Interaction takes place between human and the actuator, as humans are passive systems if the controller passivity is guaranteed then the system stability is also ensured[23].

Therefore, the PID controller is constructed as,

$$\dot{\theta}_m^{des} = G_{PID}(\tau_d^{des} - \tau_d) \tag{25}$$

where τ_d and τ_d^{des} are the output torque and desired output torque of the actuator respectively, $\dot{\theta}_m^{des}$ is the desired motor velocity. G_{PID} is the PID controller in a Laplace domain and defined as,

$$G_{PID} = K_p^{PID} + \frac{K_i^{PID}}{s} + sK_d^{PID}\frac{f_1}{s+f_1} \quad , \tag{26}$$

where K_p^{PID}, K_i^{PID} and K_d^{PID} are the proportional, integral and derivative gains respectively, f_1 is the frequency of low-pass filter that filters the derivative of the torque measurement. The inner loop control can be defined as,

$$\tau_m = G_{PI}(\dot{\theta}_m^{des} - \theta_m s\frac{f_2}{s+f_2}) \quad , \tag{27}$$

where θ_m is motor torque, f_2 is the frequency of low-pass filter that filters the derivative of the motor angle. G_{PI} is the PI controller in a Laplace domain and defined as,

$$G_{PI} = K_p^{PI} + \frac{K_i^{PI}}{s} \quad , \tag{28}$$

where K_p^{PI} and K_i^{PI} are the proportional and integral gains respectively.

For the detailed derivations of the stability conditions based on the impedance of the system, refer to [23]. The proposed boundaries of the controller parameters are as follows,

$$
\begin{gathered}
f_2 > f_1 \\
K_p^{PI} > J_m \\
K_i^{PI} < 0.5K_p^{PI} \\
K_i^{PID} < 0.5K_p^{PID} \\
K_d^{PID} > \frac{4K_p^{PID}}{f_2^2}
\end{gathered}
\tag{29}
$$

4.3 Cascaded PID Controller with DoB

For the systems that are highly prone to disturbances, DoB based methods effectively increase the robustness of the system. Considering the disturbances during the design is highly preferred. In the literature, DoB's are applied to the Cascaded PID control methods[29].

As the inner velocity loop directly affects the overall performance, achieving the robust velocity control at the inner loop level may increase the tracking performance of the overall controller configuration. Therefore adding a motor side disturbance observer can compensate the disturbances sourced from the friction and external disturbances acting on the load side directly. The cascaded controller with a DoB configuration can be seen in Figure 17. The tuning of the controller follows the same rules as the previous method.

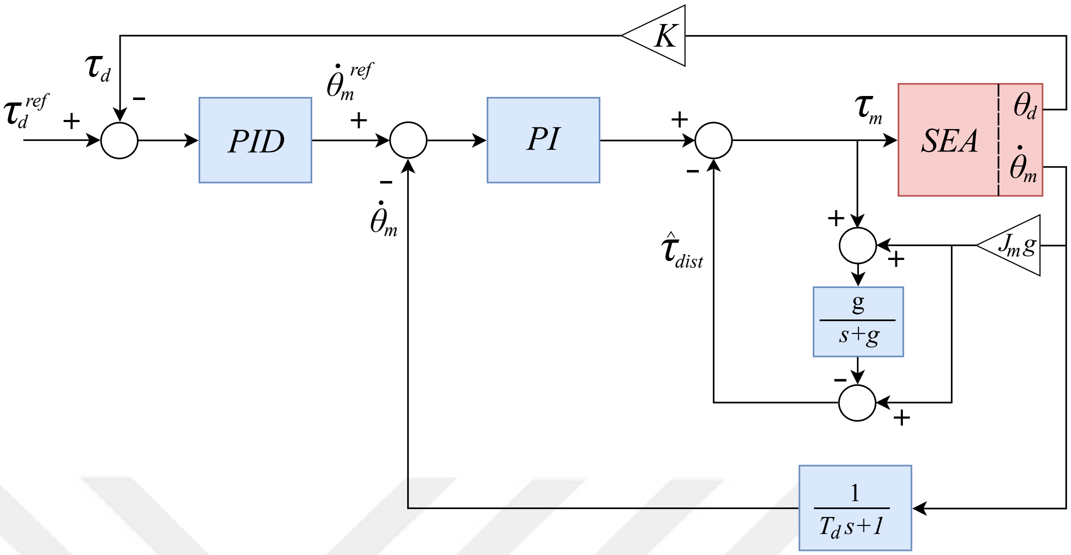


Figure 17: Block diagram of the Cascaded control with inner velocity loop and DoB

4.4 PID with model based feed-forward and DoB

With the feed-forward term, feedback term and a disturbance observer, Oh proposed a robust model-based control algorithm in [25]. High precision tracking is intended by directly measuring and feeding back the deflection. Thus, this approach is convenient for the SEAs in which the deflection can be directly measured. The controller scheme can be seen in Figure 18.

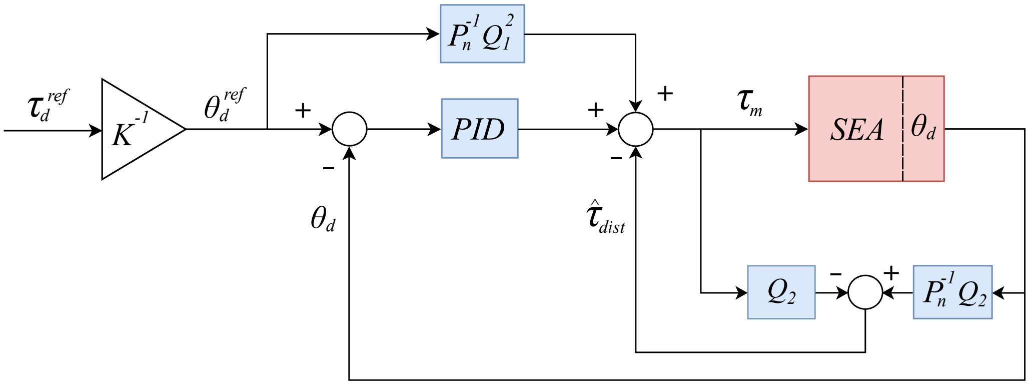


Figure 18: Block diagram of the PID with model based feed-forward and DoB

In this figure, P_n is the nominal model, Q_1 and Q_2 are the low-pass filter and

Butterworth filter respectively. The plant is modeled as a two-mass system that represents motor and the load dynamics.

The feed-forward term contains the inverse of the nominal model multiplied by a low-pass filter and implemented as $P_n^{-1}Q_1^2$ where Q_1 is a low pass filter which can be seen in equation (30). The feedback term is the PID controller which was tuned in accordance with the nominal plant. To overcome the disturbances, a disturbance observer was designed using the inverse of the nominal model (P_n^{-1}) [25]. To realize P_n^{-1}, a second order Butterworth filter with a cut-off frequency ω_c was used as follows[29],

$$Q_1(s) = \frac{\omega_v}{s + \omega_v} \tag{30}$$

$$Q_2(s) = \frac{\omega_c^2}{s^2 + \sqrt{2}\omega_c s + \omega_c^2} \tag{31}$$

Disturbance observer is implemented to overcome the disturbances caused by sources such as, effects of the external torques on the spring and link. Moreover, modeling errors can be considered as disturbances. Differently then other conventional DoB designs, DoB is applied to the deflection directly. The main reason for such synthesis is decoupling the torsional spring's elasticity from the load side dynamics in the sense of disturbances acting on the system. The closed-loop DoB has the following transfer functions[25],

$$G(s) = \frac{P}{1 - Q_2 + Q_2 P P_n^{-1}}(s) \tag{32}$$

$$H(s) = \frac{(1 - Q_2)P}{1 - Q_2 + Q_2 P P_n^{-1}}(s) \tag{33}$$

in which $G(s)$ represents the transfer function from input to the deflection, $H(s)$ represents the transfer function from disturbance to the deflection. If the filter Q_2 is unity, $H(s)$ goes zero, therefore the effect of the disturbance vanishes with DoB in closed loop.

4.5 Sliding mode control with DoB

A sliding mode controller (SMC) is known to be highly suitable for controlling the systems that suffer from modeling uncertainties and disturbances. Fundamentally, an SMC defines a sliding surface that is constructed by using system states, and the aim of the controller is defined as confining the sliding variable to a particular close neighborhood[27].

In accordance with to the sign of the sliding variable, the control action switches between different values within in a discrete signal profile. The switching action sensitivity and the decay rate of the sliding trajectory can be determined by controller parameters with respect to a well-known problem: chattering[28]. Sariyildiz et al. proposed an SMC controller with a disturbance observer to overcome the chattering phenomenon. In theory, disturbance observer reduces the control signal effort, therefore, the chattering effect is significantly eliminated by the presence of a disturbance observer.

SMC was implemented to the motor side of the SEA in [27], the desired motor angle is calculated by subtracting the link angle from the desired output torque of the actuator. Since the deflection of the CoEx-SEA is directly measured, driving the equations accordingly is beneficial in the sense of exposing less measurement noise. Although, the design of the SMC controller is theoretically the same. Therefore, simulations and experimental results are provided for modified controller equations only.

As a modification to Sariyildiz's proposed controller scheme, an SMC can be constructed as follows. The desired torque reference is directly related to the desired deflection by the spring constant,

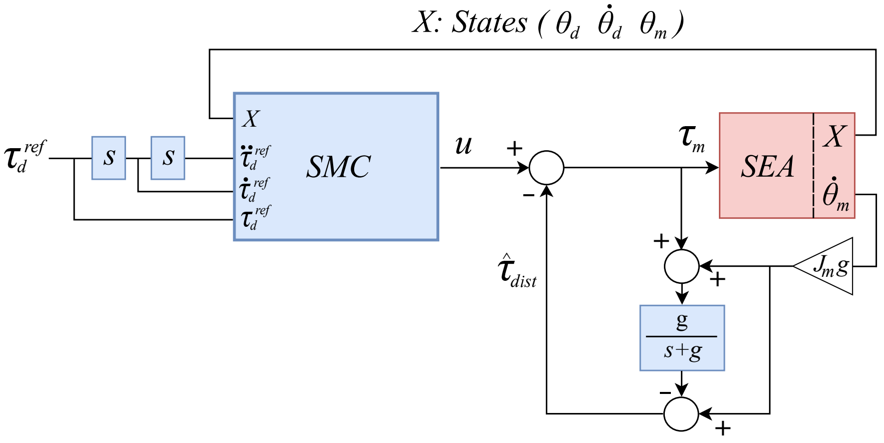


Figure 19: Block diagram of the Sliding Mode Controller with DoB

$$\tau_d^{des} = K\theta_d^{des} \tag{34}$$

where τ_d^{des} is desired torque and θ_d^{des} is desired deflection. The error can be defined as the difference between the desired deflection and the measured deflection,

$$e = \theta_d^{des} - \theta_d \tag{35}$$

In (35), θ_d is the measured deflection. The first and the second derivatives of the error is obtained to construct the sliding surface as follows,

$$\dot{e} = \dot{\theta_d}^{des} - \dot{\theta}_d \tag{36}$$

$$\ddot{e} = \ddot{\theta_d}^{des} - \ddot{\theta}_d \tag{37}$$

While $\ddot{\theta}_d^{des}$ is the second derivative of the reference signal, $\ddot{\theta}_d$ can be obtained from the dynamic equations by using the following:

$$\ddot{\theta}_d = \frac{\ddot{\theta}_m}{N} - \ddot{\theta}_l \tag{38}$$

If we rewrite the equations into (38),

$$\ddot{\theta}_d = \left(-\frac{K}{J_m N^3} - \frac{K}{J_l N}\right)\theta_m + \left(\frac{K}{J_m N^2} + \frac{K}{J_l}\right)\theta_l - \frac{B_m}{J_m N}\dot{\theta_m} + \frac{B_l}{J_l N}\dot{\theta}_l + \frac{\tau_m}{J_m N} + \tau_{dist} \quad (39)$$

where J_m, J_l, B_m and B_l are inertiae and viscous friction coefficients of motor and link sides respectively. N is the gear ratio and K is the torsional spring constant. θ_m and θ_l are motor and link angle, dots represents their successive derivatives. T_{dist} represents the disturbances. To extract the deflection from the equation (39) by using the relation between the motor angle, link angle and deflection that is $\theta_d = \theta_m N^{-1} - \theta_l$, the motor angle is divided by the gear ratio,

$$\ddot{\theta}_d = \left(-\frac{K}{J_m N^2} - \frac{K}{J_l}\right)\theta_m N^{-1} + \left(\frac{K}{J_m N^2} + \frac{K}{J_l}\right)\theta_l - \frac{B_m}{J_m N}\dot{\theta}_m + \frac{B_l}{J_l N}\dot{\theta}_l + \frac{\tau_m}{J_m N} + \tau_{dist} \quad (40)$$

and,

$$\ddot{\theta}_d = \left(-\frac{K}{J_m N^2} - \frac{K}{J_l}\right)\theta_d - \frac{B_m}{J_m N}\dot{\theta}_m + \frac{B_l}{J_l N}\dot{\theta}_l + u + \tau_{dist} \quad (41)$$

where

$$u := \frac{\tau_m}{J_m N} \quad (42)$$

Furthermore by using the relation between motor velocity, link velocity and derivative of torsion, that is $\dot{\theta}_l = \frac{\dot{\theta}_m}{N} - \dot{\theta}_d$,

$\dot{\theta}_l$ can be removed from the equation(41) as follows,

$$\ddot{\theta}_d = -\left(\frac{K}{J_m N^2} + \frac{K}{J_l}\right)\theta_d - \left(\frac{B_l B_m - JmBl}{J_m J_l N}\right)\theta_m - \frac{B_l}{J_l}\theta_d + u + \tau_{dist} \quad (43)$$

The sliding surface can be defined by using the error and derivative of the error,

$$\sigma = \dot{e} + ce \quad (44)$$

$$\dot{\sigma} = \ddot{e} + c\dot{e} \tag{45}$$

By substituting the error and its derivative into (45) $\dot{\sigma}$ is obtained as follows,

$$\dot{\sigma} = \theta_d^{\ddot{des}} + \left(\frac{K}{J_m N^2} + \frac{K}{J_l}\right)\theta_d + \left(\frac{B_l B_m - JmBl}{J_m J_l N}\right)\theta_m + \frac{B_l}{J_l}\theta_d - u - \tau_{dist} + c(\dot{\theta}_d^{des} - \dot{\theta}_d) \tag{46}$$

And the control action can be designed from the equation as (46),

$$u = \rho sign(\sigma) + \ddot{\theta}_d^{\;des} + \left(\frac{K}{J_m N^2} + \frac{K}{J_l}\right)\theta_d + \left(\frac{B_l B_m - JmBl}{J_m J_l N}\right)\theta_m + \frac{B_l}{J_l}\theta_d + c(\dot{\theta}_d^{des} - \dot{\theta}_d) \tag{47}$$

4.6 Differential Flatness Control with DoB

Differential Flatness (DF) control was proposed by M. Fliess et al. mainly for motion planning applications of the systems that are differentially flat. A system is called differentially flat if all system states, inputs and outputs are expressible with the different combinations of flat output variables and its derivatives that are differentially independent[30]. However, differential flatness control is not a robust control scheme since it highly suffers from uncertainties[20]. To this end, as an addition to the DF-based controller, Sariyildiz and Yu proposed a high-order disturbance observer in state space and tackled this problem in the controller design. Controller block diagram can be seen in Figure 20.

The general dynamic model of the SEA can be expressed as,

$$\dot{x} = f(x, u) \tag{48}$$

where x is the state vector and u is the system input, if the system is differentially flat, there exists a function ϕ such as,

$$y_{DF} = \phi(x, u, \dot{u}, \ddot{u}, ..., u_n) \tag{49}$$

where y_{DF} is differentially flat output variable, and ”n” is a finite integer. Flatness theory implies that state vector and the input of the system can be expressed as a combination of differentially flat output variable and its derivatives such as,

$$x = \kappa_x(y_{DF}, \dot{y}_{DF}, \ddot{y}_{DF}, ..., y_{DF}^{(r)}) \tag{50}$$

and

$$u = \kappa_x(y_{DF}, \dot{y}_{DF}, \ddot{y}_{DF}, ..., y_{DF}^{(r+1)}) \tag{51}$$

where r is a finite integer[20].

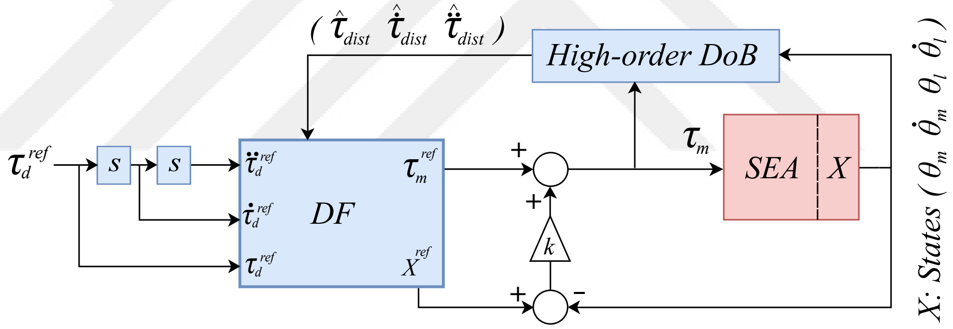

$(\hat{\tau}_{dist} \;\; \dot{\hat{\tau}}_{dist} \;\; \ddot{\hat{\tau}}_{dist})$
High-order DoB
τ_d^{ref}
s
s
$\ddot{\tau}_d^{ref}$
$\dot{\tau}_d^{ref}$
τ_d^{ref}
DF
τ_m^{ref}
X^{ref}
+
+
k
+
-
τ_m
SEA
X
X: States (θ_m $\dot{\theta}_m$ θ_l $\dot{\theta}_l$)

Figure 20: Block diagram of the DF+DoB controller

Controllability of the system is a must for having differentially flat property. Therefore controllability of the system has to be investigated. For the SEAs that have relatively high spring constants, there is a possibility that pole-zero cancellation may occur in the model. Therefore the system may not be fully controllable. Differently, from the original controller design, environment dynamics were also included during the derivation of the equations. Additionally, since there is a gear reduction, N gear ratio was also included. However environment dynamics were chosen as small as possible in implementation, it makes sense since the actuator in a force control mode will always interact with an environment.

SEA model for the controller design can be defined in the Laplace domain as,

$$A(s)x(s) + D(s) = B(s)u \tag{52}$$

where

$$A(s) = \begin{bmatrix} J_m s^2 + B_m s + KN^{-2} & -KN^{-1} \\ -KN^{-2} & J_l s^2 + (B_l + B_{env})s + (K + K_{env})N^{-1} \end{bmatrix} \tag{53}$$

where J_m, J_l, B_m and B_l are inertias and viscous frictions of motor and link side respectively, N is the gear ratio and K is the torsional spring constant. K_{env} and B_{env} are the environment stiffness and damping, respectively. θ_m and θ_l are motor and link angle, dots represents their successive derivatives, and $X(s) = [\theta_m\ \theta_l]^T$, $B(s) = [1\ 0]^T$, $D(s) = [d_1\ d_2]^T$, $u = \tau_m$

$D(s)$ is the disturbance vector where d_1 and d_2 are disturbances. In the controller design, disturbances and their derivatives are obtained from the high-order DoB in state space.

To generate the state and input references, equations are provided in [20], that are,

$$x^{ref}(s) = P(s)y_{DF} = P_1(s)y_{DF} + P_2(s)D(s) \tag{54}$$

$$u^{ref}(s) = Q(s)y_{DF} = Q_1(s)y_{DF} + Q_2(s)D(s) \tag{55}$$

where P(s) is obtained by solving,

$$C^T A(s)P(s)y_{DF} + C^T D(s) = 0 \tag{56}$$

where $C^T = [1\ 0]$ is orthogonal to the B(s). $Q_1(s)$ and $Q_2(s)$ of $Q(s)$ are obtained from,

$$Q_1(s) = (B^T(s)B(s))^{-1}B^T(s)A(s)P_1(s) \tag{57}$$

$$Q_2(s) = (B^T(s)B(s))^{-1}B^T(s)(A(s)P_2(s) + I) \tag{58}$$

Considering A(i,j) represents the i'th row and j'th column of A matrix, $x^{ref}(s)$ can be derived as,

$$x^{ref}(s) = \begin{bmatrix} A(2,2)y_{DF} - d_2 A(2,1)^{-1} \\ -A(2,1)y_{DF} \end{bmatrix} \tag{59}$$

and $u^{ref}(s)$ can be derived as

$$u^{ref} = A(1,1)A(2,2) - A(1,2)A(2,1) - A(1,1)A(2,1)^{-1}d_2 + d_1 \tag{60}$$

Desired output torque of the actuator is, $\tau_d^{des} = \theta_m N^{-1} - \theta_l$, therefore x^{ref} can be substituted as,

$$\tau_d^{des} = K(x^{ref}(1)N^{-1} - x^{ref}(2)) \tag{61}$$

Final equations of the DF controller that are obtained by substituting system matrices into (59, 60 and 61) are as follows,

$$x^{ref}(s) = \begin{bmatrix} J_l\ddot{y}_{DF} + (B_l + B_{env})\dot{y}_{DF} + N^{-1}(K + K_{env})y_{DF} + d_2 N^2 K^{-1} \\ K N^{-2} y_{DF} \end{bmatrix} \tag{62}$$

$$u^{ref} = \rho_1 \ddddot{y}_{DF} + \rho_2 \dddot{y}_{DF} + \rho_3 \ddot{y}_{DF} + \rho_4 \dot{y}_{DF} + d_{total} \tag{63}$$

where

$$\rho_1 = J_m J_l \tag{64}$$

$$\rho_2 = J_m(B_l + B_{env}) + J_l B_m \tag{65}$$

$$\rho_3 = KN^{-2}J_l + (K + K_{env})N^{-1}J_m + B_m(B_l + B_{env}) \tag{66}$$

$$\rho_4 = KN^{-2}(B_l + B_{env}) + N^{-1}(K + K_{env})B_m \tag{67}$$

$$d_{total} = d_1 + d_2 + B_m N^2 K^{-1}\dot{d}_2 + J_m N^2 K^{-1}\ddot{d}_2 \tag{68}$$

Desired torque reference is,

$$\tau_d^{des} = K\big[J_l N^{-1}\ddot{y}_{DF} + (B_l + B_{env})N^{-1}\dot{y}_{DF} + K_{env}N^{-2}y_{DF}\big] \tag{69}$$

Therefore desired differentially flat output reference can be defined as a function of desired output torque τ_d^{des}.

$$\ddot{y}_{DF}^{des} = NJl^{-1}\big[K^{-1}\tau_d^{des} - d_2 N K^{-1} - (B_l + B_{env})N^{-1}\dot{y}_{DF} - N^{-2}K_{env}y_{DF}\big] \tag{70}$$

Motor torque, τ_m was calculated via a state feedback with pole placement in original controller design, however as proposed in our work, with real implementations taking the derivative of the states and state references can be replaced with a PD controller. See Figure 21.

Therefore, the state feedback can be tuned with a proportional K_p and derivative gain K_d. While the PD controller gains of the motor side are defined as K_p^m and K_d^m, PD controller gains of link side are defined as K_p^l and K_d^l. The relationship between these parameters is proposed below,

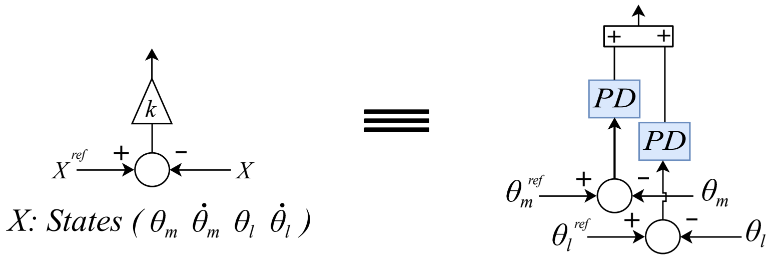


Figure 21: Pole placement versus double PD controller

$$
\begin{gathered}
K_p^m = -K_p \\
K_d^m = K_d \\
K_p^l = NK_p \\
K_d^l = -NK_d
\end{gathered}
\tag{71}
$$

To estimate the disturbances d_1 and d_2 and their derivatives, high-order DoB was implemented in [20]. By defining auxiliary variables z_1, z_2 and z_3 such as,

$$
\begin{gathered}
z_1 = \tau_{dist} + L_1 x \\
z_2 = \dot{\tau}_{dist} + L_2 x \\
z_3 = \ddot{\tau}_{dist} + L_3 x
\end{gathered}
\tag{72}
$$

where, x is the state vector that is $x = [\theta_m\ \dot{\theta}_m\ \theta_l\ \dot{\theta}_l]$, τ_{dist}, $\dot{\tau}_{dist}$ and $\ddot{\tau}_{dist}$ are the disturbance vector and its derivatives respectively. L_1,L_2 and L_3 are controller gains and can be tuned with respect to [20],

$$L_1 = 3f$$
$$L_2 = 3f^2$$
$$L_3 = f^3$$

$$(73)$$

where f is the frequency of DoB. The update equation of the auxiliary variables can be defined as,

$$\dot{z}_1 = -L_1 z_1 + z_2 + L_1(Ax + bu + L_1 x) - L_2 x$$
$$\dot{z}_2 = -L_2 z_1 + z_3 + L_2(Ax + bu + L_1 x) - L_3 x$$
$$\dot{z}_3 = -L_3 z_1 + L_3(Ax + bu + L_1 x)$$

$$(74)$$

and the estimated disturbances can be obtained from the equation (72),

$$\hat{\tau}_{dist} = z_1 - L_1 x$$
$$\hat{\dot{\tau}}_{dist} = z_2 - L_2 x$$
$$\hat{\ddot{\tau}}_{dist} = z_3 - L_3 x$$

$$(75)$$

where $\hat{\tau}_{dist}$, $\hat{\dot{\tau}}_{dist}$ and $\hat{\ddot{\tau}}_{dist}$ are the estimated disturbance vector and its derivatives.

CHAPTER V

SIMULATION RESULTS

In this section, simulation results are presented. In order to compare the theoretical performance of the controllers, simulation study was conducted via MATLAB Simulink environment running on the host PC that has i7-7700HQ CPU and 16 GB of RAM. Under this section, comparison was made between simulation results. Simulations were conducted with a fixed sampling time of 0.5 milliseconds via continuous-time models. Also tuning of the controllers were intuitive, therefore there was no systematical tuning criteria between different control methods.

In the simulations, environment was implemented with a constant stiffness and constant damping. These parameters can be seen in Table 4. Same model parameters were used for every controller scheme (see Table 3). Furthermore, to make the simulations more realistic, measurement noises were implemented to the SEA models and modeling errors were intentionally placed into the system model, see Table 4.

Simulation study investigated different scenarios such as:

Parameter Uncertainty: Due to modeling errors, model mismatches are expected in the real implementations. Therefore investigating the robustness to the parameter uncertainty in the simulations is a must. Robustness of the controllers are usually being analyzed via sensitivity functions, however for a type of nonlinear controllers, conventional sensitivity analysis is quite challenging. To this end, uncertainty is realized by investigating the robustness for one parameter; the inertia of the link. Since for the human wearable robotic systems, link inertia is expected to vary under multiple circumstances. With different link inertia values, RMS errors of step responses for each controller was obtained.

Table 4: Simulated SEA model parameters

Parameters	***Explanation***	***Modeling error***
J_m	Motor Inertia	2 J_{m_n}
B_m	Motor Viscous Friction	0.5 B_{m_n}
J_l	Link Inertia	1.5 J_{l_n}
B_l	Link Viscous Friction	5 B_{l_n}
K	Spring Stiffness	1.1 K_n
K_{env}	Stiffness of the enviroment	5000 Nm / rad
B_{env}	Damping of the enviroment	200 Nm s / rad

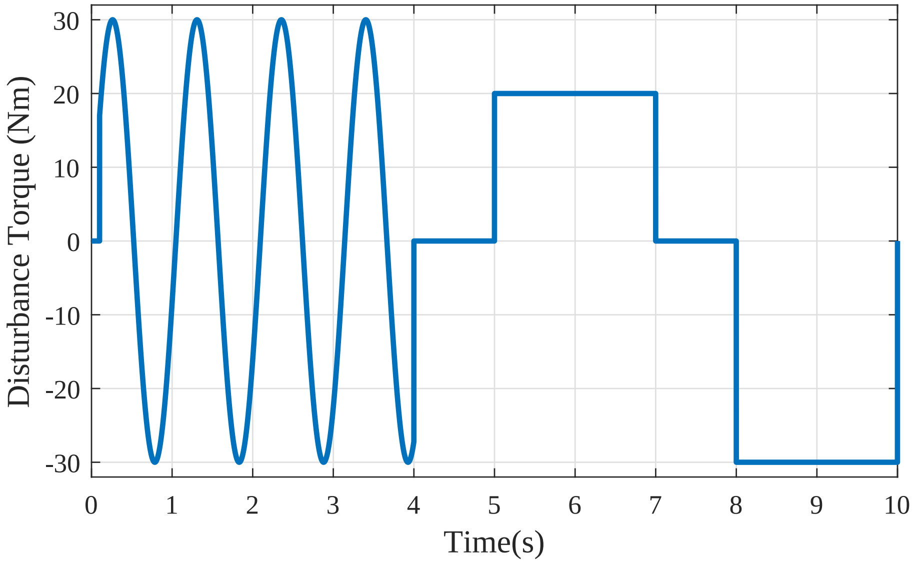


Figure 22: Disturbance applied to the simulations

Step Reference: Step responses of each controllers were investigated with the external disturbance acting on the link. This disturbance signal can be seen in Figure 22, contains respectively: sinusoidal signal (30 Nm amplitude with frequency of 6 rad/s), step signal (20 Nm amplitude) and negative step signal (-30 Nm amplitude). Also, the discontinuity between these signals were also considered as a disturbance. This disturbance signal was applied on the simulated SEAs for all controllers and their RMS errors are presented.

Square Reference: Tracking of the square signal for all controllers was also investigated as a different scenario under the same external disturbance signal (See Figure 22)

Sinusoidal Reference: Sinusoidal signal tracking of different frequencies (3 rad/s, 6 rad/s and 12 rad/s) was investigated for each controller, unlike previous scenarios, disturbance was a step signal (20 Nm amplitude at 7.5 second).

Bode Plots Tracking of the chirp signal, with the frequency range of 0.1 Hz to 25 Hz, was investigated and the frequency analysis was prepared. Bode diagrams are provided to compare the bandwidths of the controllers theoretically.

Comparison of the simulations were based on RMS values, step informations and standard deviation of the parameter uncertainty case.

5.1 PID Controller

Even though, its impractical the increase the proportional gain as much as we desire in real implementation, controller gains can be increased freely in simulation environment. Controller parameters are given in Table 5. Step response can be seen in Figure 23 and step information can be found in Table 6. Since the controller gains are not practical, scenarios that are mentioned earlier were not simulated with the PID controller.

Table 5: PID Controller simulation parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	15000
K_i	Integral gain	600
K_d	Derivative gain	30
w_d	Approx. Diff. Freq. PID	1600 Hz

Table 6: Step response information of the PID Controller simulation

RMS	0.264
Steady-state error	0.030
OS%	25.577
US%	3
Settling Time	0.028 s
Peak Time	0.018 s
Rise Time	0.012 s

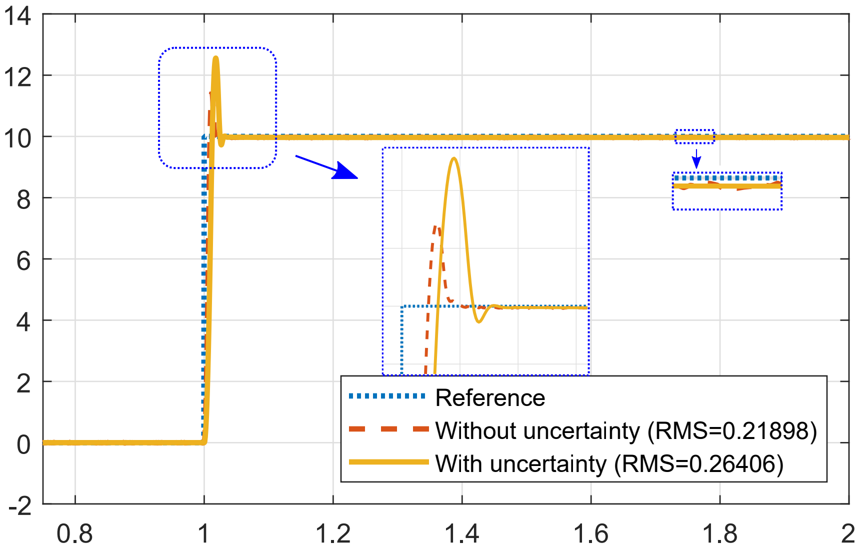


Figure 23: Step response of the PID controller simulation

5.2 Cascaded PID Controller

Since the measurement noise was implemented to the SEA model, derivative of the error signal in the PID controller was obtained via approximate differentiation method with the low-pass frequency of 1600 Hz (w_d). Motor velocity was also calculated via approximate differentiation with the low-pass frequency of 1000 Hz (w_{θ_m}). Selection of these frequencies was based on proposed boundaries $w_{\theta_m} < w_d$, in [23]. Also tuning of the Cascaded PID contoller was performed by following the same rules proposed in

[23]. Controller parameters are given in Table 7. Step response can be seen in Figure 24 and the step information can be found in Table 8.

Table 7: Cascaded PID Controller simulation parameters

Parameters	***Explanation***	***Value***
K_{p1}	Outer loop proportional gain	900
K_{i1}	Outer loop integral gain	200
K_{d1}	Outer loop derivative gain	3
w_d	Approx. Diff. Freq. PID	1600 Hz
K_{p2}	Inner loop proportional gain	0.005
K_{i2}	Inner loop integral gain	0.0005
w_{θ_m}	Approx. Diff. Motor velocity	1000 Hz

Table 8: Step response information of the Cascaded PID simulation

RMS	0.256
OS%	3.945
US%	0
Settling Time	0.018 s
Peak Time	0.016 s
Rise Time	0.012 s

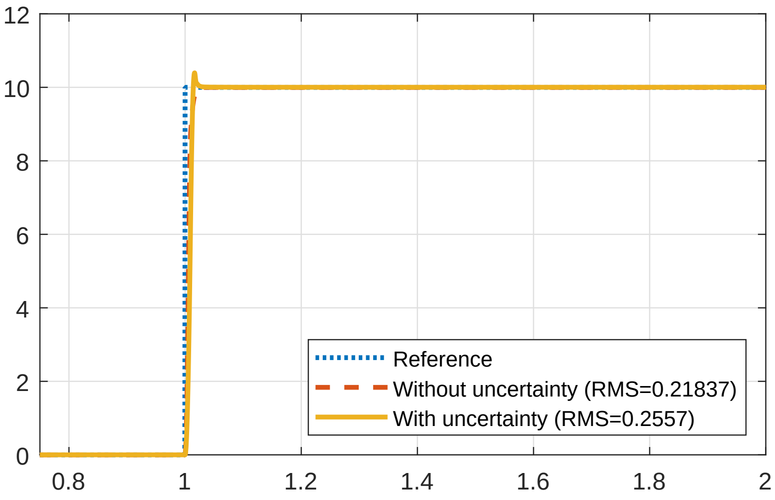


Figure 24: Step response of the Cascaded PID controller simulation

Disturbances were relatively suppressed by the controller, however as it can bee seen in the Figure 25, a steady state error was effecting the system response. For the square signal tracking see Figure 26. RMS values of all scenarios are provided in the Table 9.

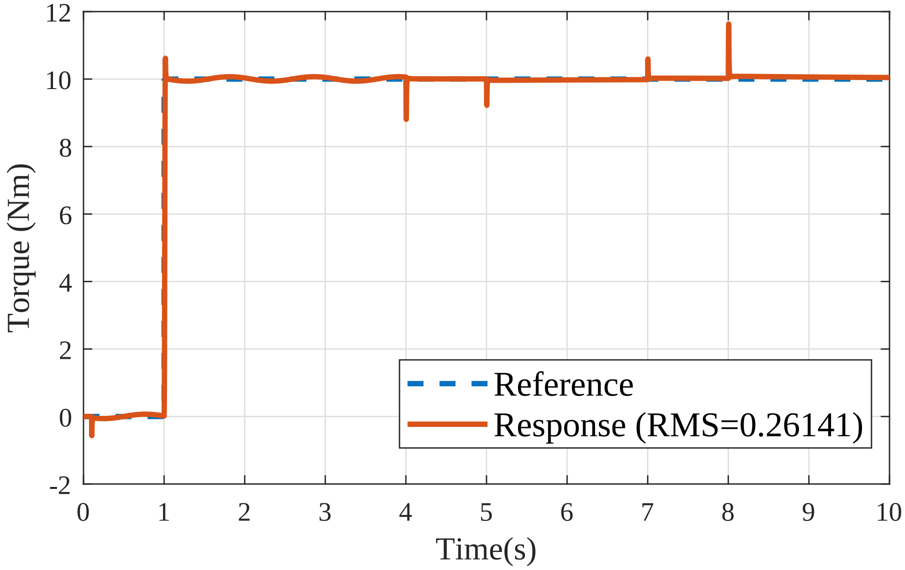


Figure 25: Tracking of the step reference with disturbance given in Figure 22 (Cascaded PID controller simulation)

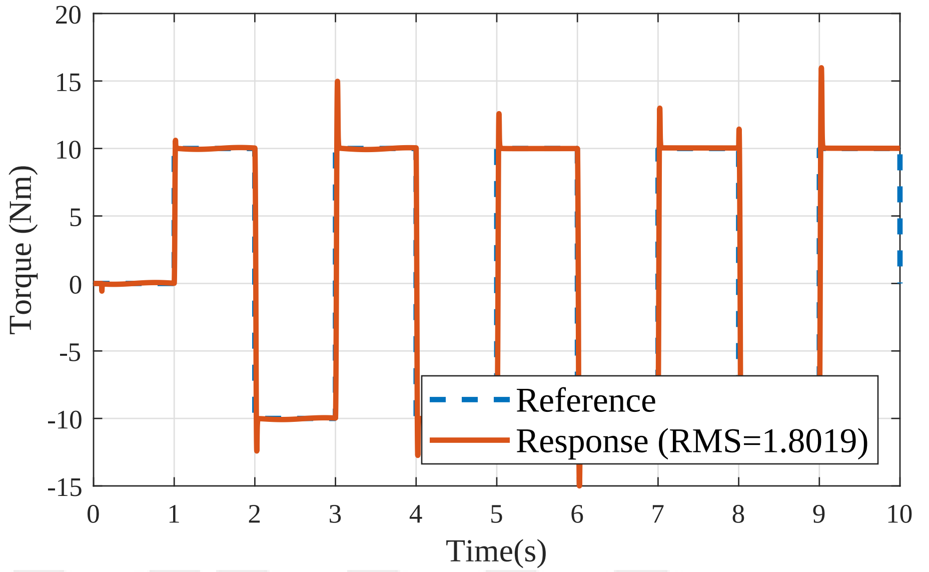


Figure 26: Tracking of the square reference with disturbance given in Figure 22 (Cascaded PID controller simulation)

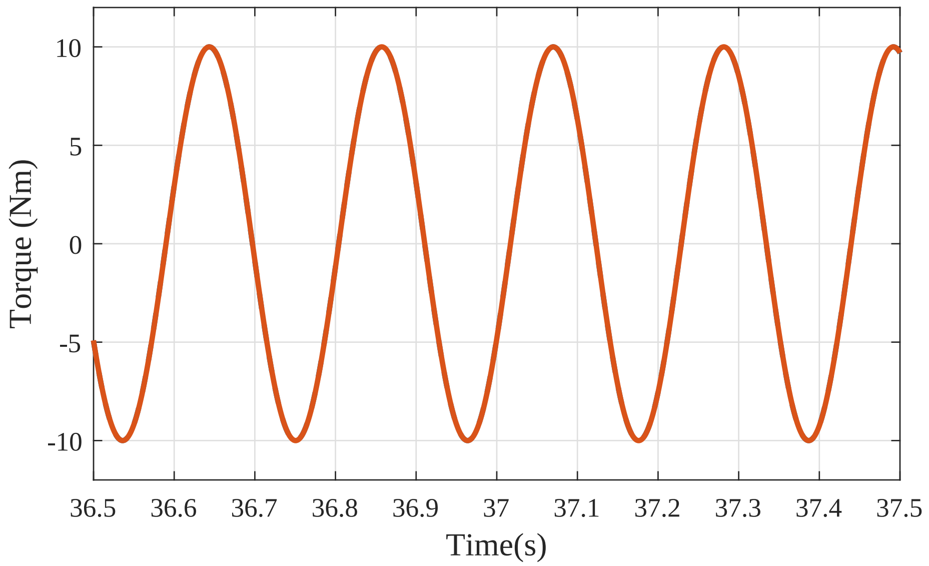


Figure 27: Tracking of the sinusoidal reference with frequency of 4.4 Hz (Cascaded PID controller simulation)

Table 9: RMS errors of the Cascaded PID simulation

Step with uncertainty	0.256
Step without uncertainty	0.218
Step with disturbance	0.261
Square with disturbance	1.802
Sinusoidal 3 rad/s	0.046
Sinusoidal 6 rad/s	0.052
Sinusoidal 12 rad/s	0.050
Chirp (0.5 Hz to 5 Hz)	0.008
Average	0.337

To demonstrate the tracking performance of a sinusoidal signal, a part of the chirp signal that corresponds to the frequency of 4.4 Hz is provided in Figure 27. The bode diagram of the controller can be seen in Figure 28. According to the plot, the bandwidth of the controller is calculated as approximately 20 Hz.

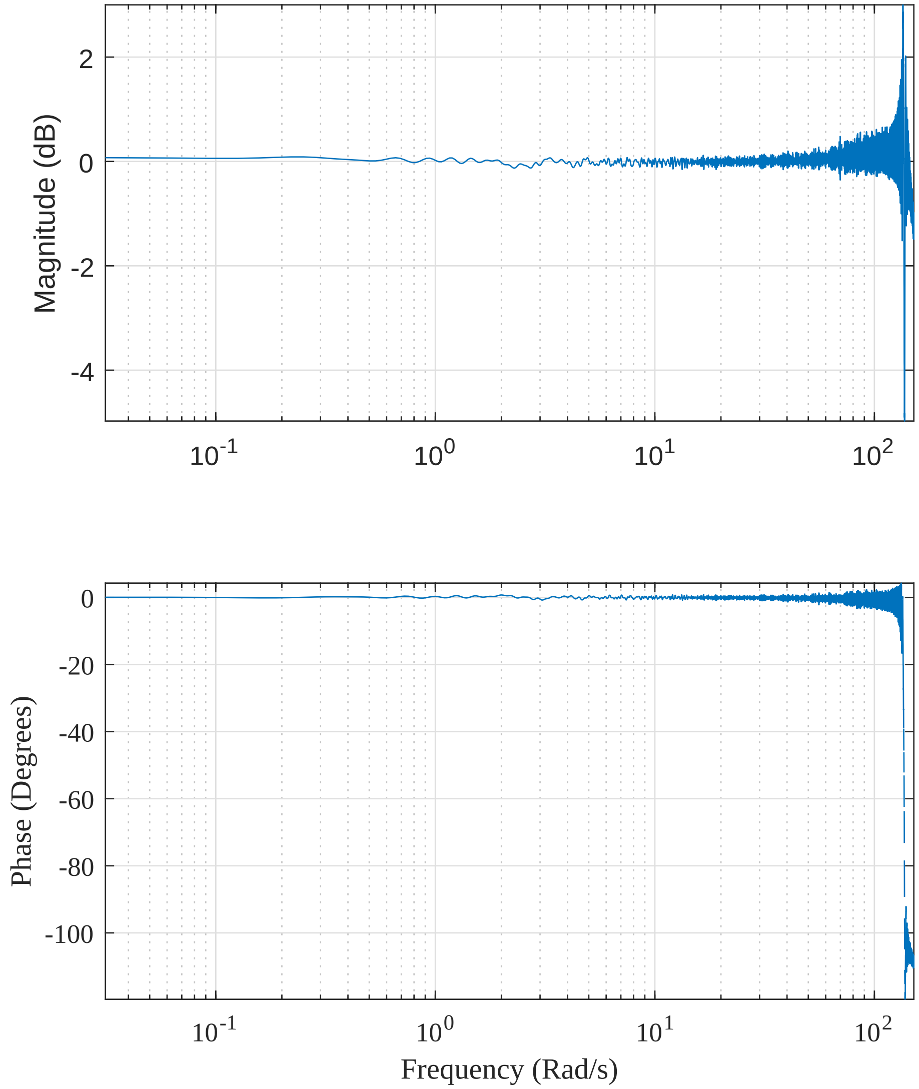


Figure 28: Bode plot of the Cascaded PID controller simulation

5.3 Cascaded PID Controller with DoB

Without changing the controller parameters, a DoB was implemented to the motor side of the SEA to increase the robustness of the Cascaded PID controller. Step response can be seen in Figure 29 and the step information can be found in Table 10.

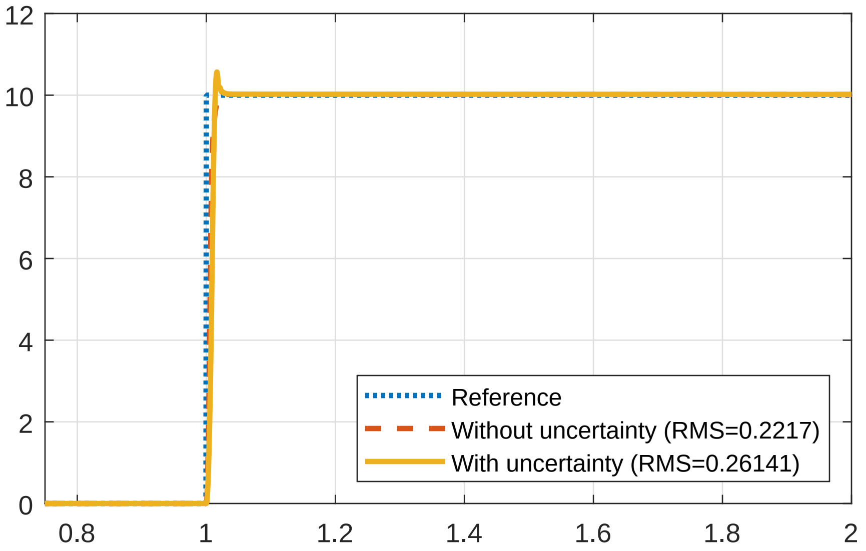


Figure 29: Step Response of the Cascaded PID Controller with DoB simulation

Table 10: Step response information of the Cascaded PID with DoB simulation

RMS	0.261
OS%	5.677
US%	0
Settling Time	0.019 s
Peak Time	0.016 s
Rise Time	0.012 s

The external disturbance acting on the SEA was already suppressed in the previous control scheme, Cascaded PID controller. However, adding the DoB eliminated the steady-state error, see Figure 30. For the square signal tracking see Figure 31. RMS values of all scenarios are provided in the Table 11.

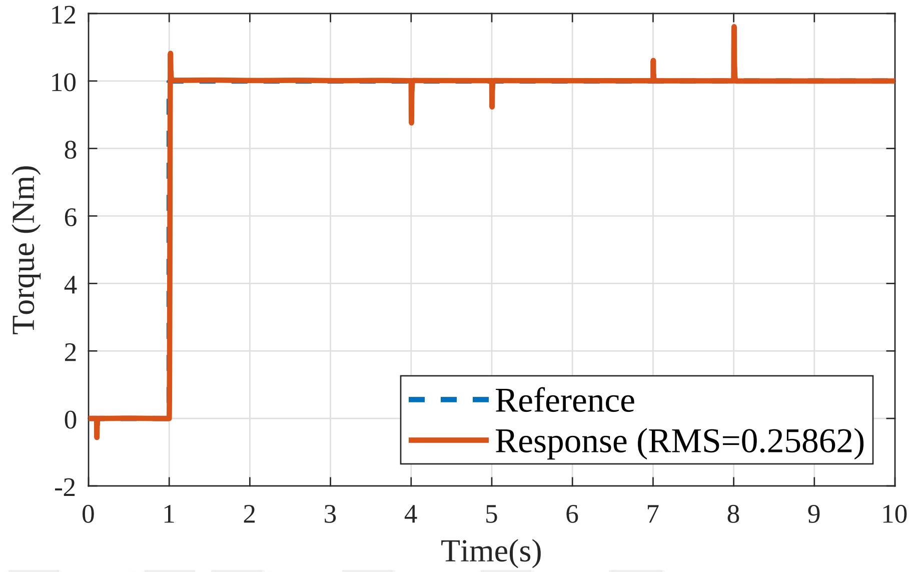


Figure 30: Tracking of the step reference with disturbance given in Figure 22 (Cascaded PID controller with DoB simulation)

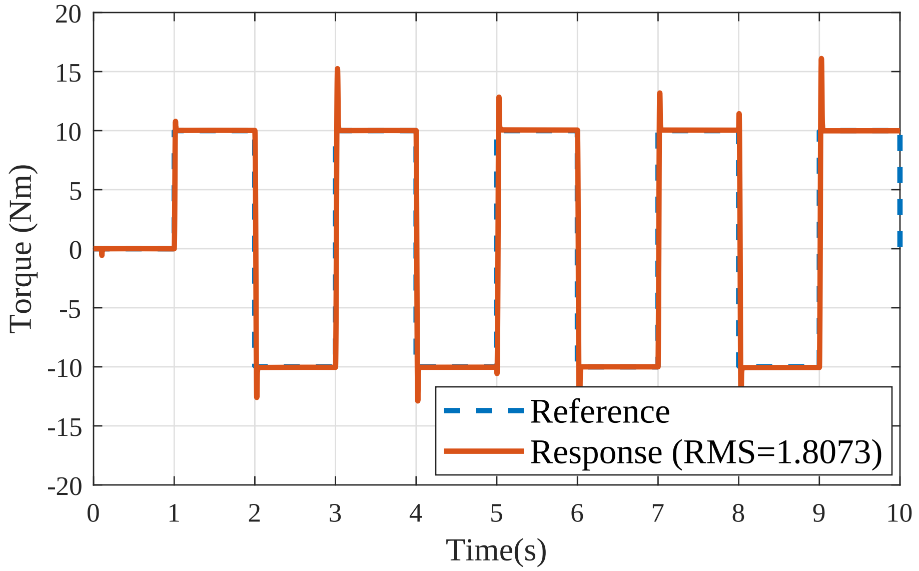


Figure 31: Tracking of the square reference with disturbance given in Figure 22 (Cascaded PID controller with DoB simulation)

Table 11: RMS errors of the Cascaded PID with DoB simulation

Step with uncertainty	0.261
Step without uncertainty	0.222
Step with disturbance	0.259
Square with disturbance	1.807
Sinusoidal 3 rad/s	0.038
Sinusoidal 6 rad/s	0.043
Sinusoidal 12 rad/s	0.042
Chirp (0.5 Hz to 5 Hz)	0.014
Average	0.336

To demonstrate the tracking performance of a sinusoidal signal, a part of the chirp signal that corresponds to the frequency of 4.4 Hz is provided in Figure 32. The bode diagram of the controller can be seen in Figure 28. According to the plot, the bandwidth of the controller has not been effected by the presence of the DoB. It was also calculated as approximately 20 Hz.

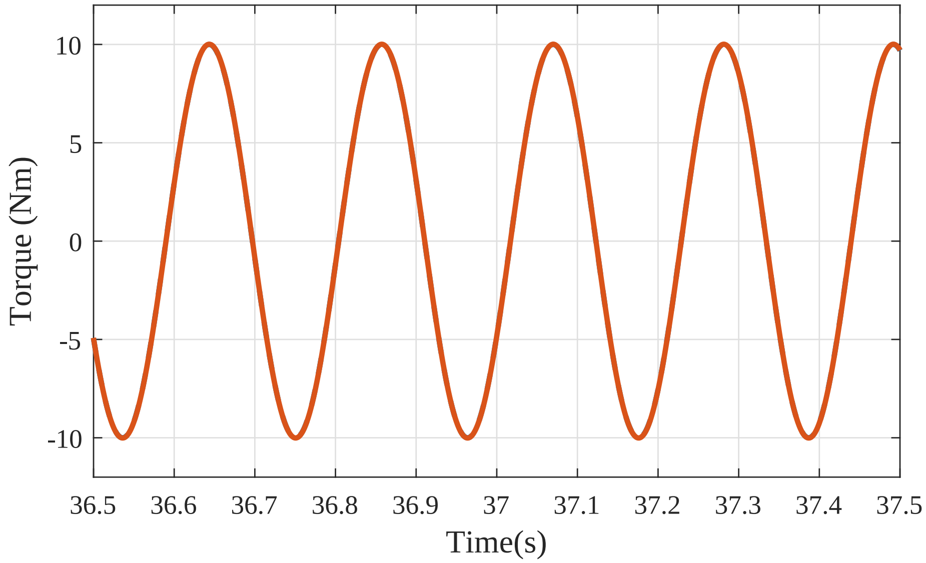


Figure 32: Tracking of the sinusoidal reference with frequency of 4.4 Hz (Cascaded PID controller with DoB simulation)

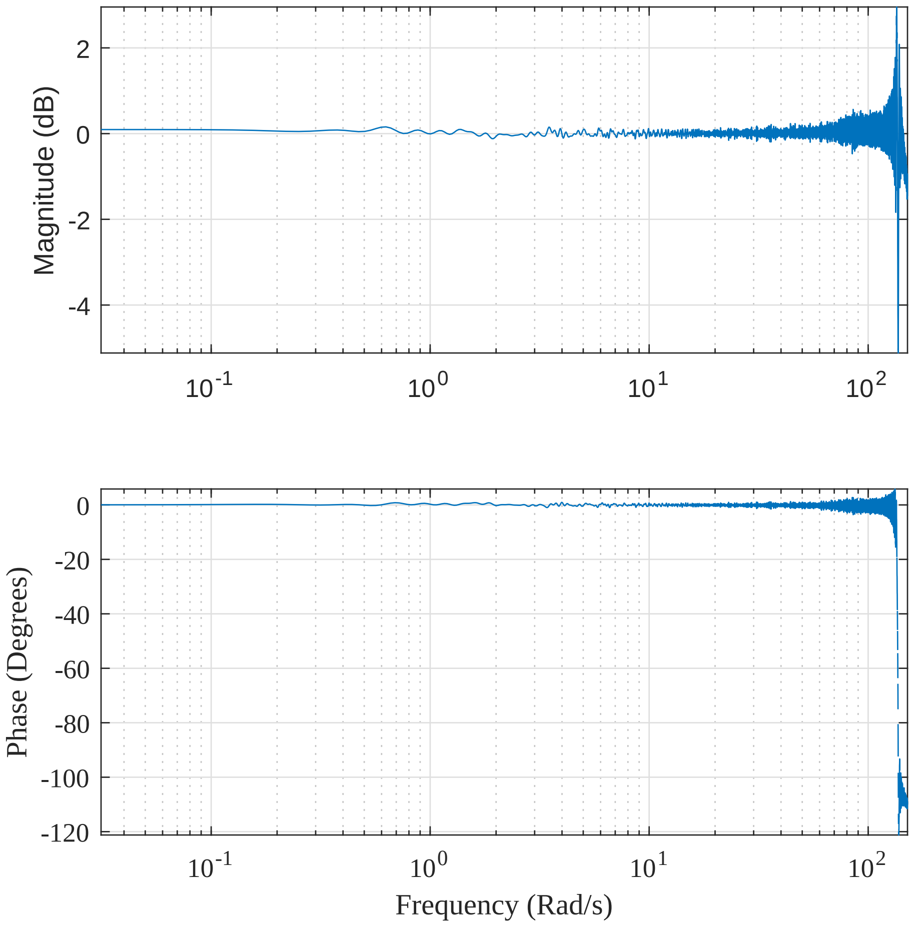


Figure 33: Bode plot of the Cascaded PID controller with DoB simulation

5.4 PID with model based feed-forward and DoB

Tuning of the FF+PID+DoB controller was based on the nominal model of the SEA and controller parameters can be seen in Table 12. Step response of the controller is provided in Figure 34 and step information can be found in Table 13.

Table 12: FF+PID+DoB simulation parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	100
K_i	Integral gain	100
K_d	Derivative gain	14
w_d	Approx. Diff. Freq. PID	1600 Hz
w_{Q_2}	DoB frequency	1600 Hz
w_{Q_1}	FF frequency	628 Hz

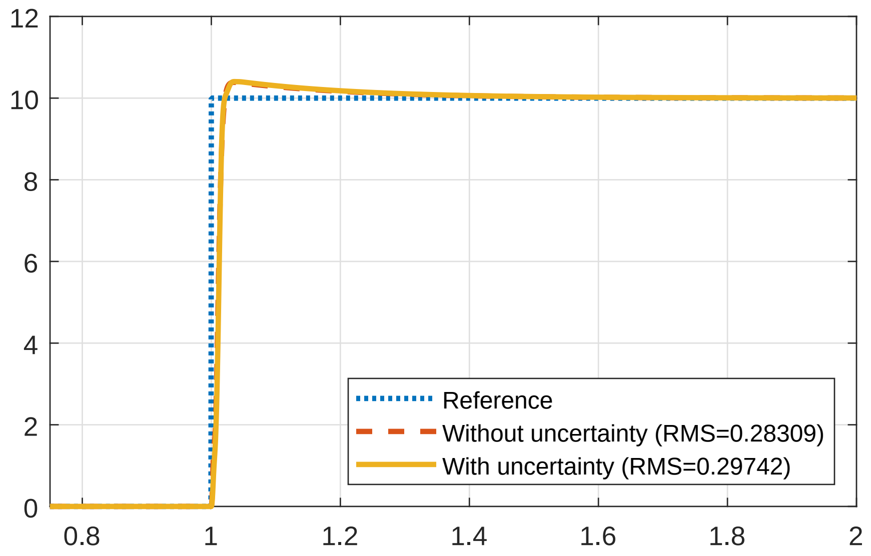


Figure 34: Step response of the FF+PID+DoB controller simulation

Table 13: Step response information of the FF+PID+DoB simulation

RMS	0.297
OS%	4.044
US%	0
Settling Time	0.179 s
Peak Time	0.036 s
Rise Time	0.016 s

Compared to the other control methods, disturbance rejection property of the

FF+PID+DoB controller was more favorable as it can be seen in Figure 35. Moreover, according to the RMS values and the standard deviation, FF+PID+DoB was more robust to the modeling uncertainty then any other controller scheme. For the square signal tracking see Figure 36. However, in overall, controller scheme exhibited least favorable RMS error values. RMS values of all scenarios are provided in the Table 14.

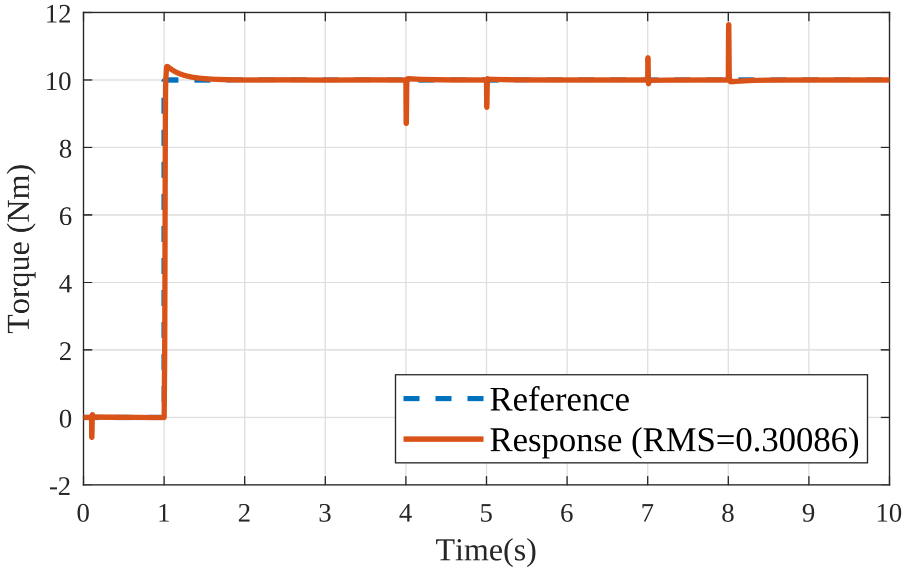


Figure 35: Tracking of the step reference with disturbance given in Figure 22 (FF+PID+DoB controller simulation)

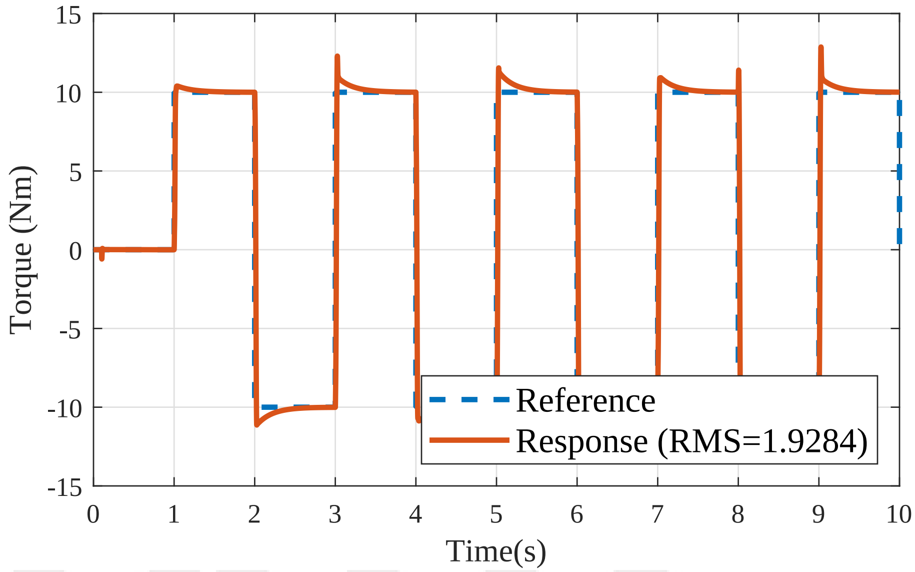


Figure 36: Tracking of the square reference with disturbance given in Figure 22 (FF+PID+DoB controller simulation)

Table 14: RMS errors of the FF+PID+DoB simulation

Step with uncertainty	0.297
Step without uncertainty	0.283
Step with disturbance	0.301
Square with disturbance	1.928
Sinusoidal 3 rad/s	0.062
Sinusoidal 6 rad/s	0.108
Sinusoidal 12 rad/s	0.198
Chirp (0.5 Hz to 5 Hz)	0.314
Average	0.436

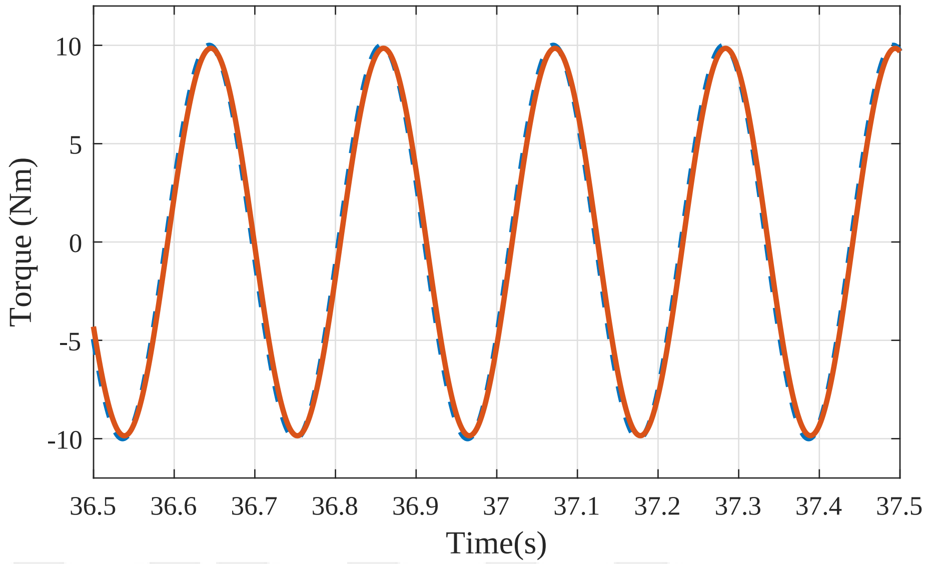


Figure 37: Tracking of the sinusoidal reference with frequency of 4.4 Hz (FF+PID+DoB controller simulation)

To demonstrate the tracking performance of a sinusoidal signal, a part of the chirp signal that corresponds to the frequency of 4.4 Hz is provided in Figure 37. Compared to the other methods, FF+PIF+DoB highly suffers from phase delays which increases the RMS errors of tracking. The bode diagram of the controller can be seen in Figure 38. According to the plot, the bandwidth of the controller was calculated as approximately 20 Hz.

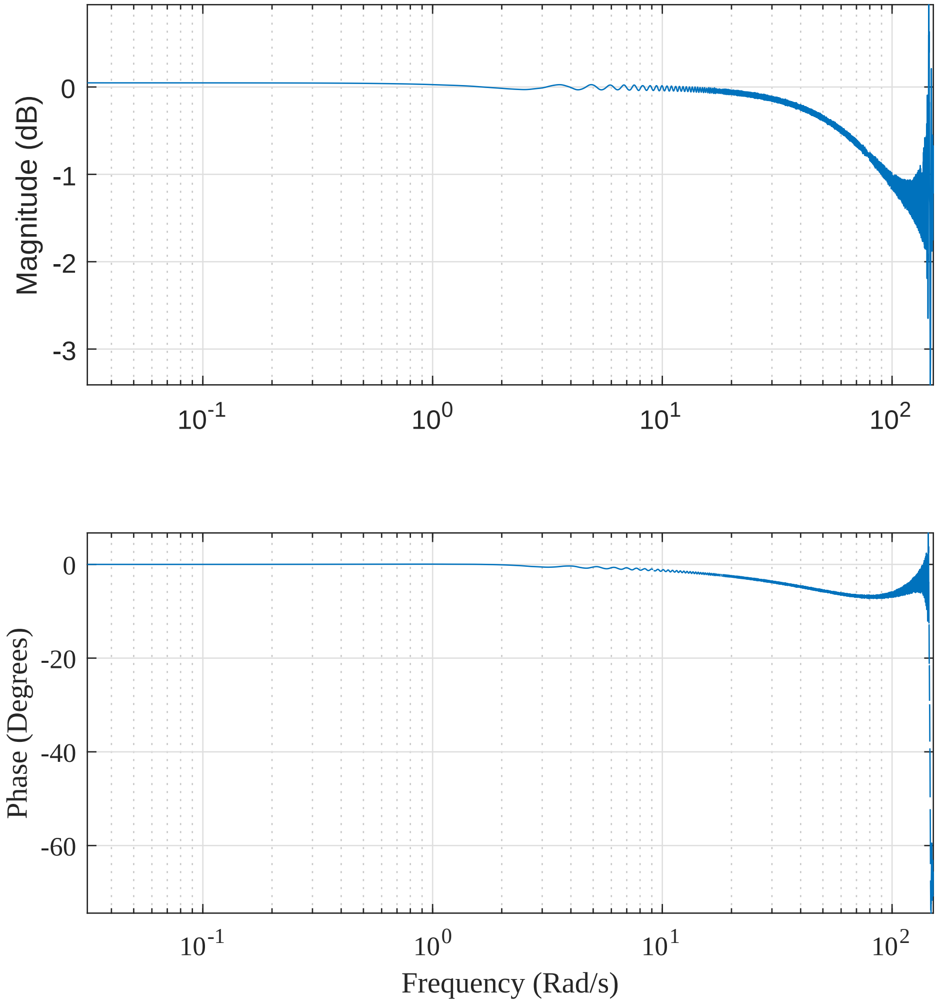


Figure 38: Bode plot of the FF+PID+DoB controller simulation

5.5 Sliding Mode Controller with DoB

Tuning process of the controller was achieved by adjusting the convergence rate of the sliding surface and the controller gain. Increasing the controller gain leads to more frequent switching action. Controller parameters can be seen in Table 15. Step response can be seen in Figure 39 and step information can be found in Table 16.

Step reference tracking under the external disturbance, tracking of square signal with the same external disturbance, a part of the chirp signal that corresponds to 4.4 Hz and the bode diagram is presented in the Figures 40, 41, 42 and 43, respectively. The bandwidth of the controller was calculated as approximately 21 Hz.

Table 15: SMC+DoB simulation parameters

Parameters	***Explanation***	***Value***
c	Convergence rate	300
ρ	SMC gain	12
w_{θ_m}	Approx. Diff. Freq. motor velocity	1600 Hz
w_{θ_d}	Approx. Diff. Freq. torsion	1600 Hz
w_{DoB}	DoB frequency	1600 Hz

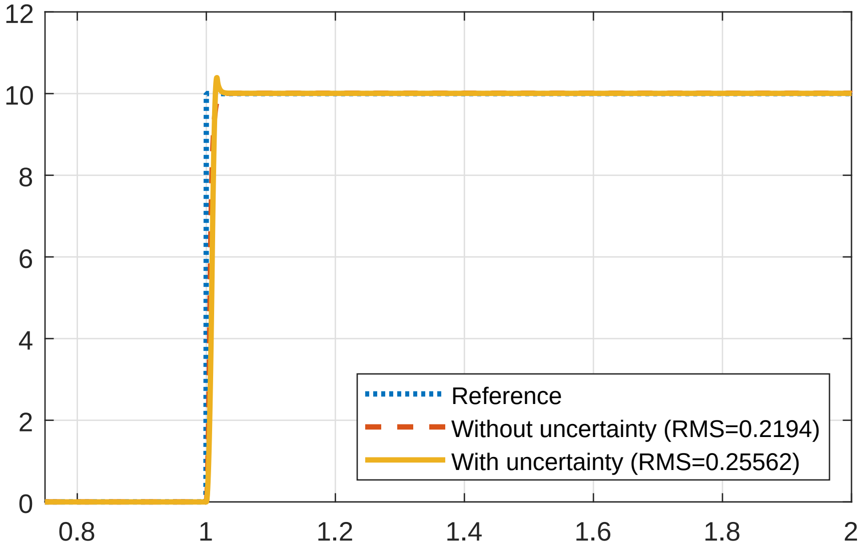


Figure 39: Step response of the SMC+DoB simulation

Table 16: Step response information of the SMC+DoB simulation

RMS	0.256
OS%	3.908
US%	0
Settling Time	0.019 s
Peak Time	0.016 s
Rise Time	0.012 s

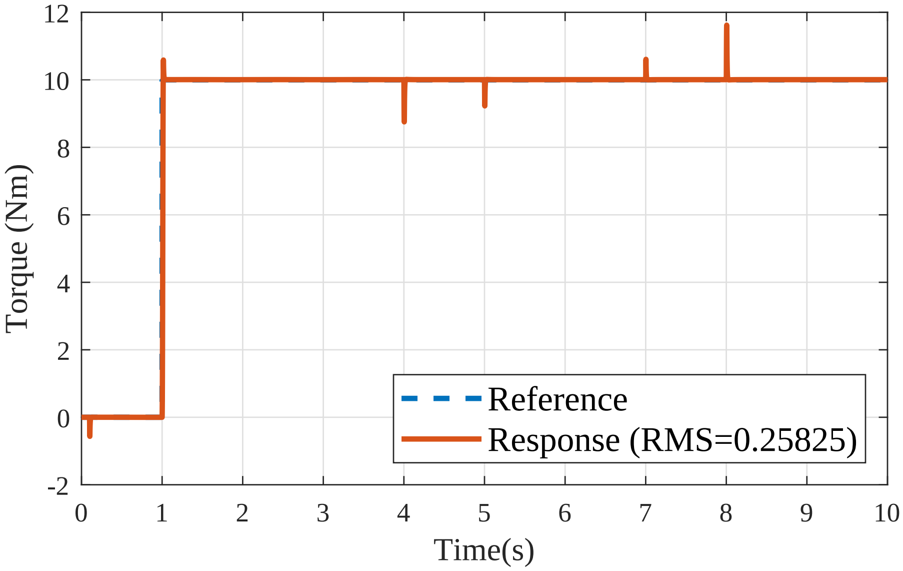


Figure 40: Tracking of the step reference with disturbance given in Figure 22 (SMC+DoB simulation)

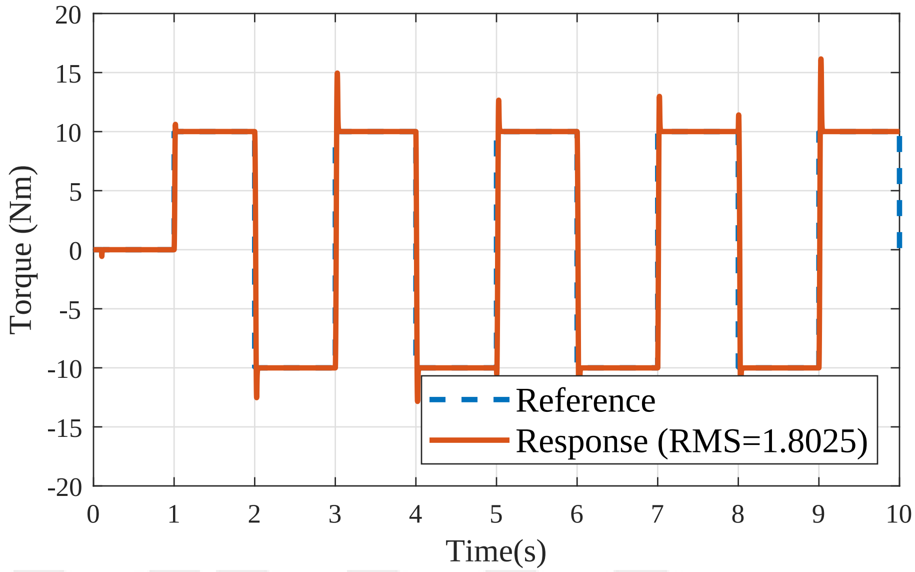


Figure 41: Tracking of the square reference with disturbance given in Figure 22 (SMC+DoB simulation)

Table 17: RMS errors of the SMC+DoB simulation

Step with uncertainty	0.256
Step without uncertainty	0.219
Step with disturbance	0.258
Square with disturbance	1.802
Sinusoidal 3 rad/s	0.039
Sinusoidal 6 rad/s	0.043
Sinusoidal 12 rad/s	0.043
Chirp (0.5 Hz to 5 Hz)	0.007
Average	0.333

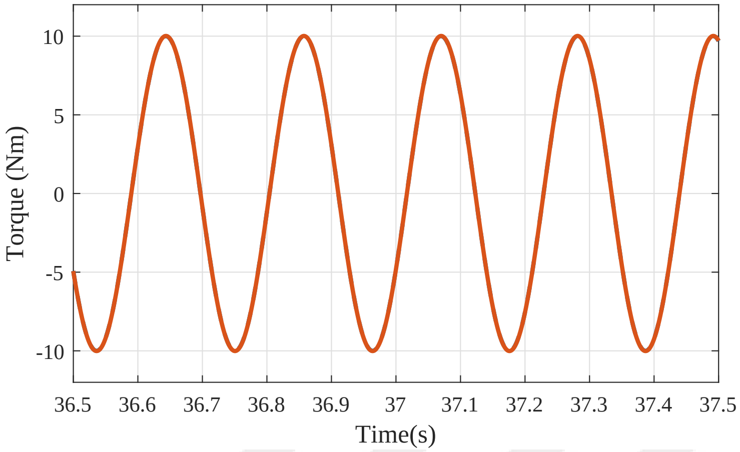


Figure 42: Tracking of the sinusoidal reference with frequency of 4.4 Hz (SMC+DoB simulation)

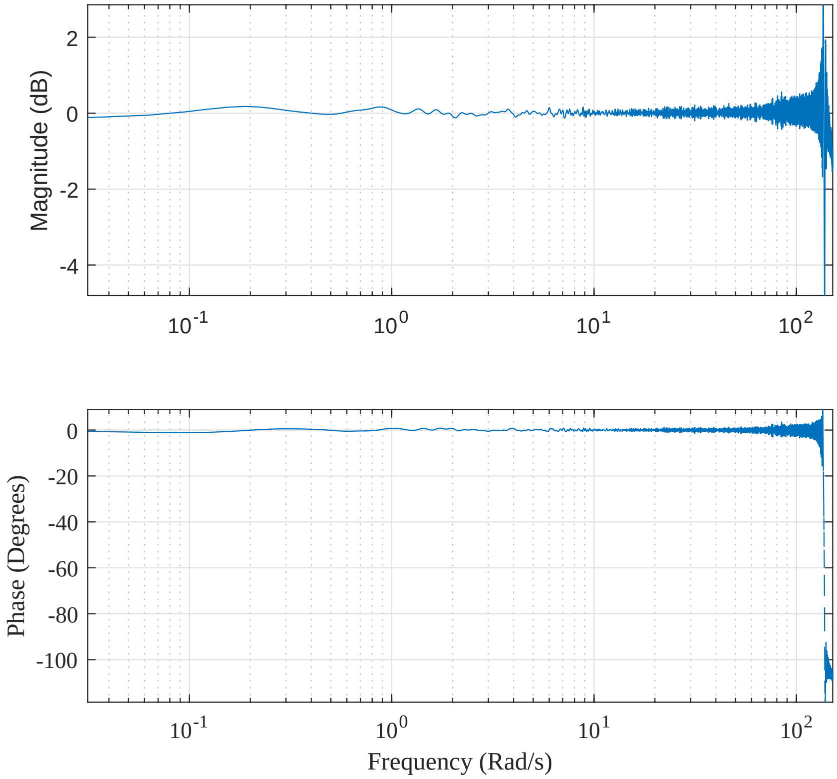


Figure 43: Bode plot of the SMC+DoB simulation

5.6 Differential Flatness Control with DoB

Since the pole placement method was replaced with a double PD controller as in Figure 21, tuning process of the controller was achieved by tuning the double PD controller in the state feedback. Controller parameters can be seen in Table 18. Step response is in Figure 44 and step information can be found in Table 19.

Step reference tracking under the external disturbance, tracking of square signal with the same external disturbance, a part of the chirp signal that corresponds to 4.4 Hz and the bode diagram is presented in the Figures 45, 46, 47 and 48, respectively. The bandwidth of the controller was calculated as approximately 18-19 hz.

Table 18: DF+DoB simulation parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	10
K_d	Derivative gain	0.1
w_d	Approx. Diff. Freq. PID	1600 Hz
w_{DoB}	DoB frequency	300 Hz

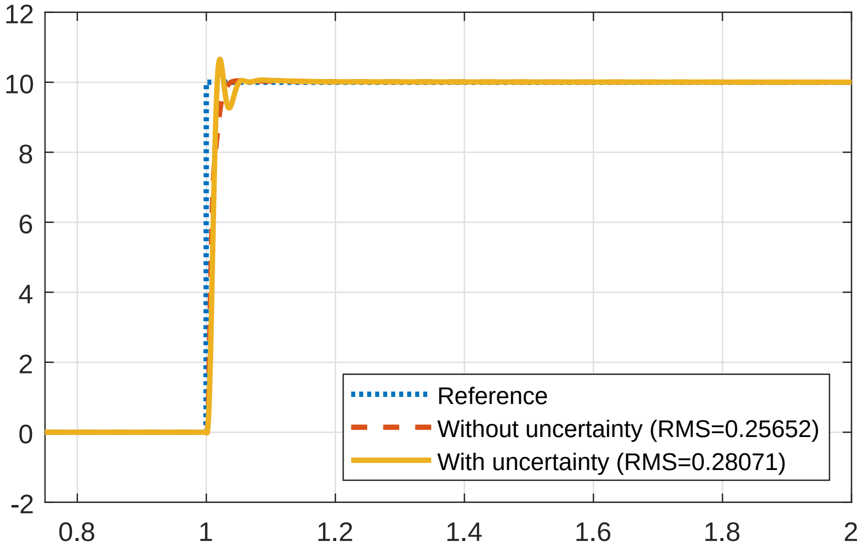


Figure 44: Step response of the DF+DoB controller simulation

Table 19: Step response information of the DF+DoB controller simulation

RMS	0.281
OS%	6.819
US%	7.37
Settling Time	0.046 s
Peak Time	0.021 s
Rise Time	0.014 s

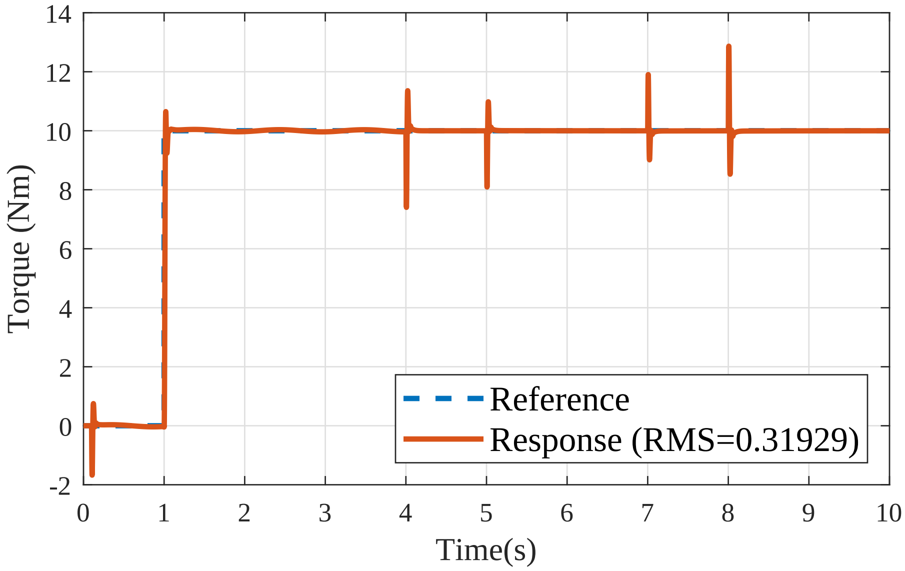


Figure 45: Tracking of the step reference with disturbance given in Figure 22 (DF+DoB controller simulation)

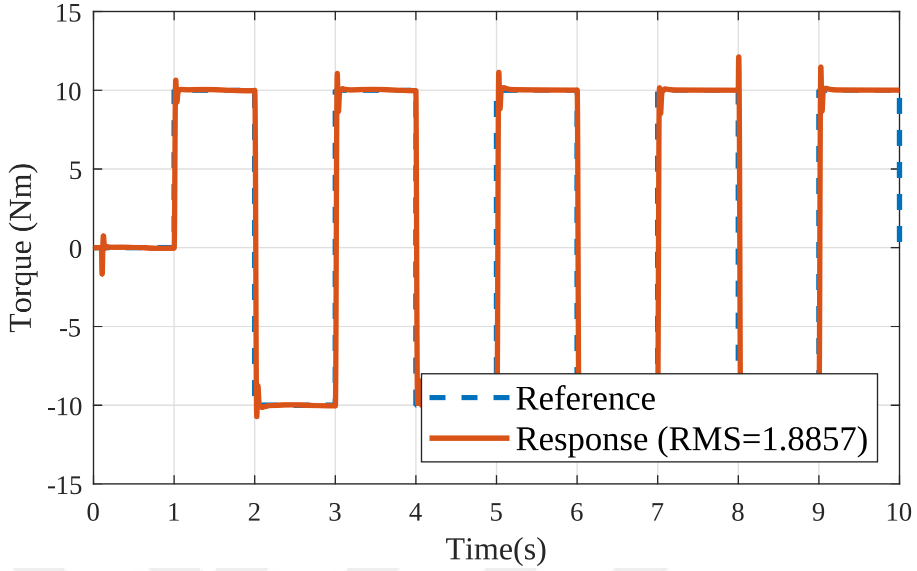


Figure 46: Tracking of the square reference with disturbance given in Figure 22 (DF+DoB controller simulation)

Table 20: RMS errors of the DF+DoB controller simulation

Step with uncertainty	0.281
Step without uncertainty	0.256
Step with disturbance	0.319
Square with disturbance	1.886
Sinusoidal 3 rad/s	0.095
Sinusoidal 6 rad/s	0.107
Sinusoidal 12 rad/s	0.137
Chirp (0.5 Hz to 5 Hz)	0.162
Average	0.405

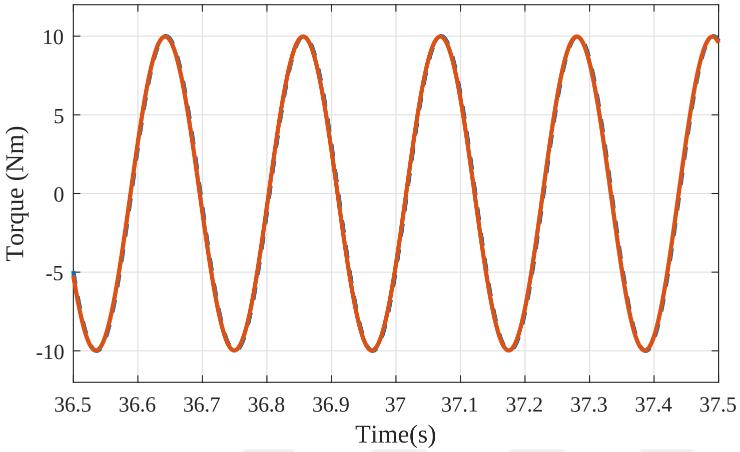


Figure 47: Tracking of the sinusoidal reference with frequency of 4.4 Hz (DF+DoB controller simulation)

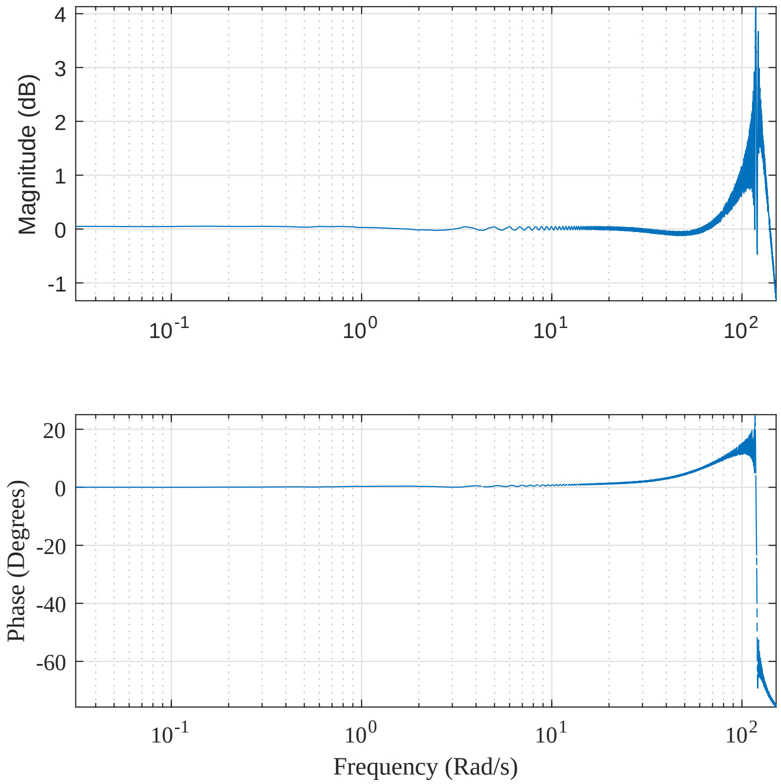


Figure 48: Bode plot of the DF+DoB controller simulation

5.7 Discussion

Robustness to the parameter uncertainty was investigated by calculating the standard deviation of the RMS errors for different inertia values of the link. All RMS errors are presented in the Table 21. In Figure 49 the standard deviation is presented in a bar graph. Since the mean RMS error is much bigger then the standard deviation, bar represents the quarter of the mean RMS error, and the error bar represents the standard deviation.

Table 21: RMS comparison of model uncertainty simulations

Controller	***Jl***	***2 Jl***	***5 Jl***	***10 Jl***	***20 Jl***	***50 Jl***	***100 Jl***
Cascaded PID	0.2183	0.2181	0.2178	0.2174	0.2170	0.2166	0.2164
Cascaded PID + DoB	0.2183	0.2182	0.2178	0.2175	0.2171	0.2167	0.2165
FF+PID+DoB	0.2194	0.2191	0.2186	0.2180	0.2175	0.2171	0.2169
SMC + DoB	0.2831	0.2830	0.2830	0.2829	0.2828	0.2827	0.2826
DF + DoB	0.2480	0.2462	0.2458	0.2454	0.2449	0.2442	0.2439

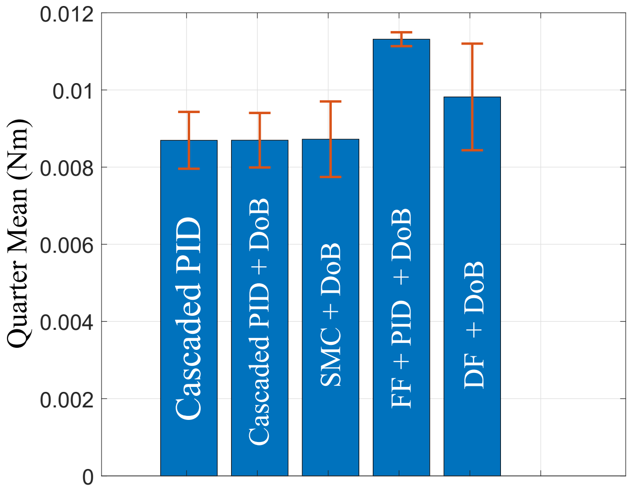


Figure 49: Standart deviation graph for model uncertainty

It can be concluded from the Figure 49 that, FF+PID+DoB controller has the lowest standard deviation. Therefore it can be said that FF+PID+DoB is relatively more robust to the parameter uncertainties compared to the other methods. Even though in simulations, it is possible to achieve better tracking results with higher controller gains, controllers were tuned according to physical limits.

In Table 22, RMS errors are given for; 3 rad/s sinusoidal, 6 rad/s sinusoidal, 12 rad/s sinusoidal and chirp signal with frequency range of 0.5 Hz to 5 Hz, respectively. RMS values are visualized in Figure 50 in order to be perceived easily by the reader.

Table 22: RMS comparison of sinusoidal tracking simulations

Controller	***3 rad/s***	***6 rad/s***	***12 rad/s***	***Chirp (0.5-5 Hz)***
Cascaded PID	0.046	0.052	0.050	0.008
Cascaded PID + DoB	0.038	0.043	0.0425	0.0139
FF+PID+DoB	0.062	0.108	0.198	0.314
SMC+DoB	0.0389	0.043	0.0427	0.007
DF+DoB	0.095	0.107	0.137	0.162

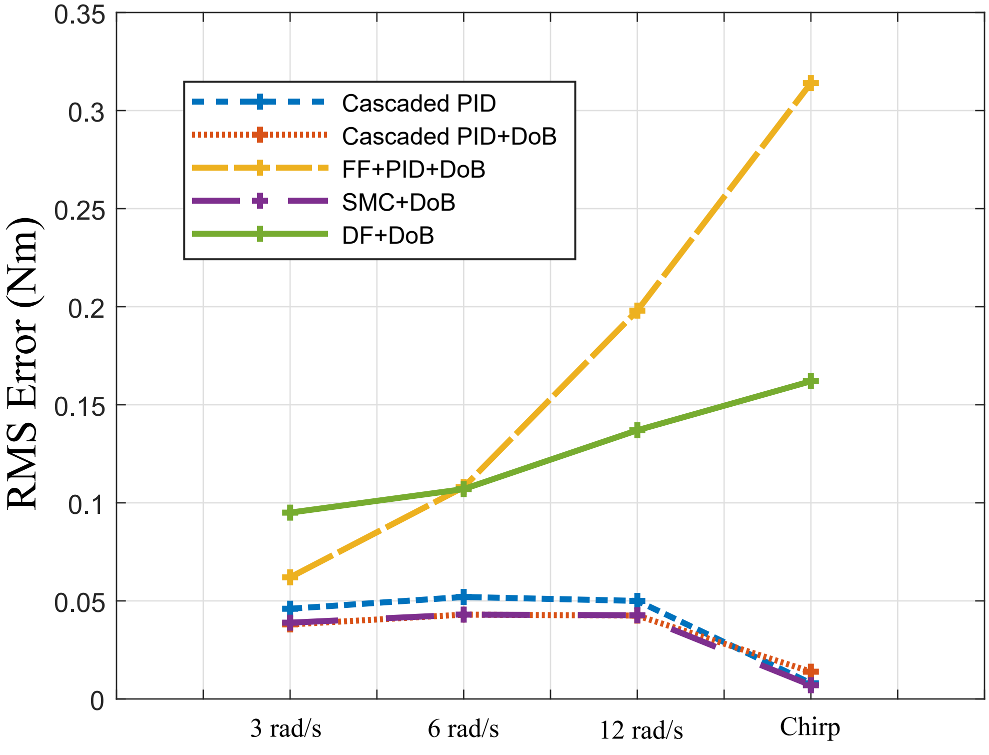


Figure 50: RMS comparison of sinusoidal tracking simulations with a graph

Average RMS value of the controllers were calculated by using the simulation data for tracking of step, square, sinusoidal and chirp references. Average RMS values can be seen in Table 23. Average RMS values are visualized in Figure 51 in order to be perceived easily by the reader. SMC+DoB has relatively lower RMS error compared to the other controller methods. Highest RMS value was the FF+PID+DoB, mostly due to phase delays. Bandwidths of the controllers in the simulations were almost equal to each other due to the motor saturation.

Table 23: Average RMS comparison of simulations

Controller	*RMS*
Cascaded PID	0.337
Cascaded PID + DoB	0.336
FF+PID+DoB	0.436
SMC+DoB	0.333
DF+DoB	0.405

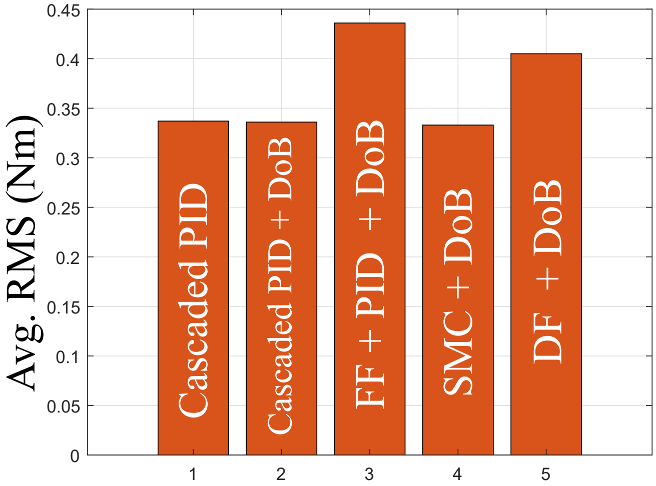


Figure 51: Average RMS comparison of simulations

In Table 24, step response informations of all controllers are given in a single table.

Table 24: Step response comparison of simulations

Controller	*OS%*	*Settling Time*	*Rise Time*	*Peak Time*
Cascaded PID	3.945	0.018 s	0.012 s	0.016 s
Cascaded PID + DoB	5.677	0.019 s	0.012 s	0.016 s
FF+PID+DoB	4.044	0.179 s	0.016 s	0.036 s
SMC+DoB	3.908	0.019 s	0.012 s	0.016 s
DF+DoB	6.819	0.046 s	0.014 s	0.021 s

CHAPTER VI

EXPERIMENT RESULTS

Experiments were conducted with a joint level controller board that runs a real-time Linux OS. Controllers were implemented in real-time C programs and discrete realizations of the continuous models were obtained via bilinear transformation. All experiments were conducted with a 2kHz sampling rate. Approximate differentiation method was used to obtain the derivatives of the encoder signals. Determination of the approximate differentiation frequencies varied between controllers depending on their sensitivities against the signal noise. Experiment data was transfered back to the host machine for visualization of the controller performances via MATLAB. Detailed hardware implementation can be found in earlier sections.

Experiment study investigated different scenarios such as:

Parameter Uncertainty: In the experiments, to be able to investigate the robustness against the environment uncertainty, link movement was restricted with a sponge in both directions as can be seen in Figure 52. Sponge represents the non-stiff environment, however as the sponge becomes compressed by the link movement, stiffness is being increased. Therefore, it can be said that in the experiments, SEA was subjected to a variable stiffness.

Step Reference: Step responses of each controllers were investigated with the restricted link movement and step informations are provided.

Square Reference: Tracking of the square signal for all controllers was also investigated with the restricted link movement

Sinusoidal Reference: Sinusoidal signal tracking of different frequencies (3 rad/s, 6 rad/s and 12 rad/s) was investigated for each controller with the restricted

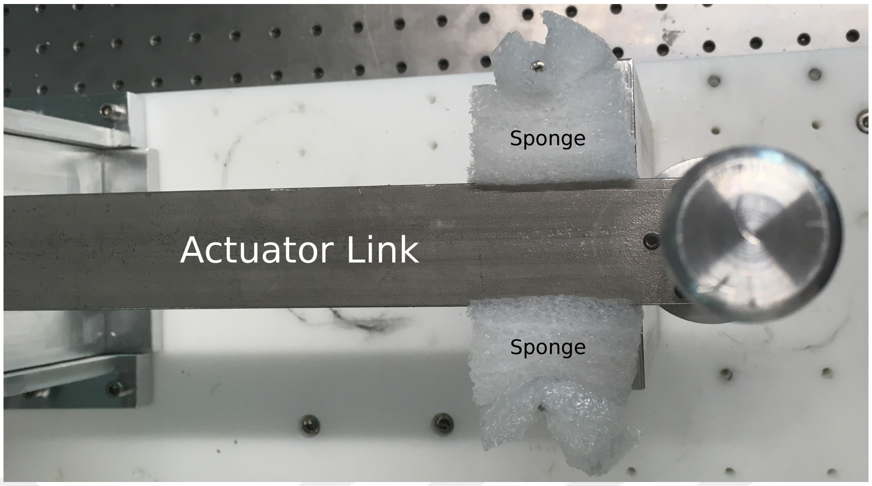


Figure 52: SEA link with subject to non-stiff environment

link movement

Bode Plots Tracking of the chirp signal, with the frequency range of 0.1 Hz to 25 Hz, were used to construct Bode diagrams.

Comparison of the experiments were based on RMS values and step informations of the controllers.

6.1 PID Controller

Implementation of the PID controller requires the deflection feedback and an approximate differentiation for the derivative term of the PID. While it is possible to increase the PID gains freely in the simulation environment, tuning of the PID controller in the experiment was problematic due to physical constrains such as encoder noise and quantization effects.

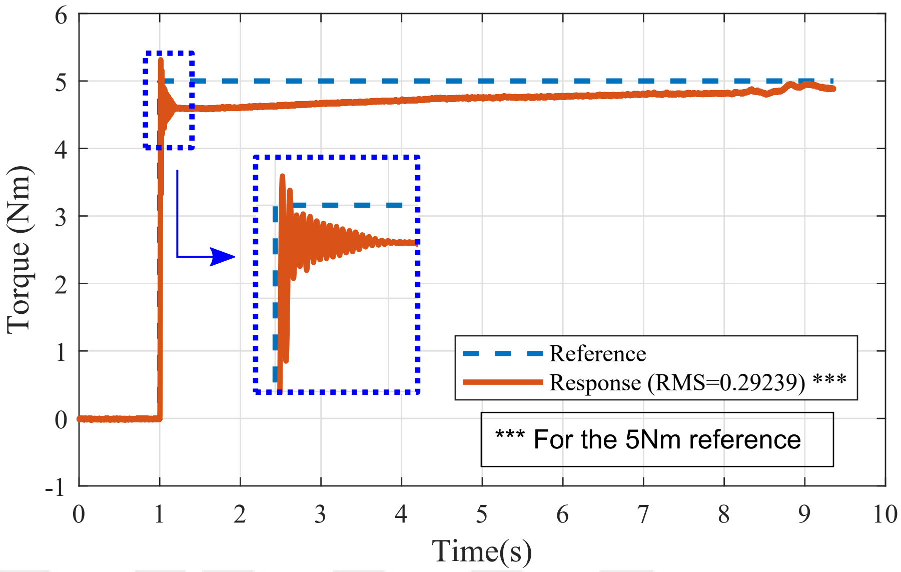


Figure 53: Step response of the PID Controller experiment (5Nm)

Moreover, PID controller is not robust against the changes in the enviroment. In the experiments, PID controller was unable to track the reference torque as it can be seen in Figure 53. Controller gains can be seen in Table 25. Increasing the proportional gain generates high oscillations and leads to instability. Since the link was subjected to a sponge, enviroment dynamics highly varies and PID controller fails to exhibit a robust performance.

Table 25: PID Controller experiment parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	1800
K_i	Integral gain	600
K_d	Derivative gain	12
w_d	Approx. Diff. Freq. PID	300 Hz

6.2 Cascaded PID Controller

Cascaded PID controller implementation contains PID, PI and approximate differentiation blocks. For the feedback, controller needs the deflection measurement and the motor velocity. Since we measure the motor angle, motor velocity was obtained via approximate differentiation. Controller gains were tuned with respect to parameter boundaries that proposed in [23], controller gains can be seen in Table 26.

Table 26: Cascaded PID Controller experiment parameters

Parameters	***Explanation***	***Value***
K_{p1}	Outer loop proportional gain	16
K_{i1}	Outer loop integral gain	7
K_{d1}	Outer loop derivative gain	0.8
w_d	Approx. Diff. Freq. PID	1600 Hz
K_{p2}	Inner loop proportional gain	0.045
K_{i2}	Inner loop integral gain	0.012
w_{θ_m}	Approx. Diff. Motor velocity	600 Hz

The step response of the controller can be seen in Figure 54, and step response information is presented in Table 27. Square and sinusoidal reference tracking performances can be seen in Figure 55 and 56, respectively. A part of the chirp signal that corresponds to frequency of 4.4 Hz is presented in Figure 57. RMS values can be found in Table 28. Bode plot of the controller is given in Figure 58, according to the plot, controller exhibited up-to 17Hz control bandwidth.

Table 27: Step response information of the Cascaded PID experiment

RMS	0.527
OS%	10.179
Settling Time	0.064 s
US%	11.6
Peak Time	0.034 s
Rise Time	0.022 s

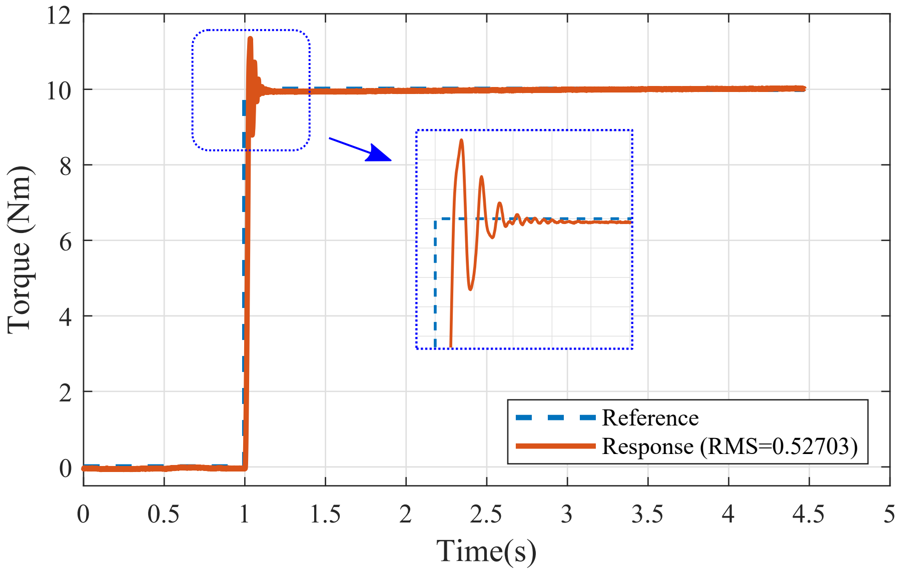


Figure 54: Step response of the Cascaded PID Controller experiment

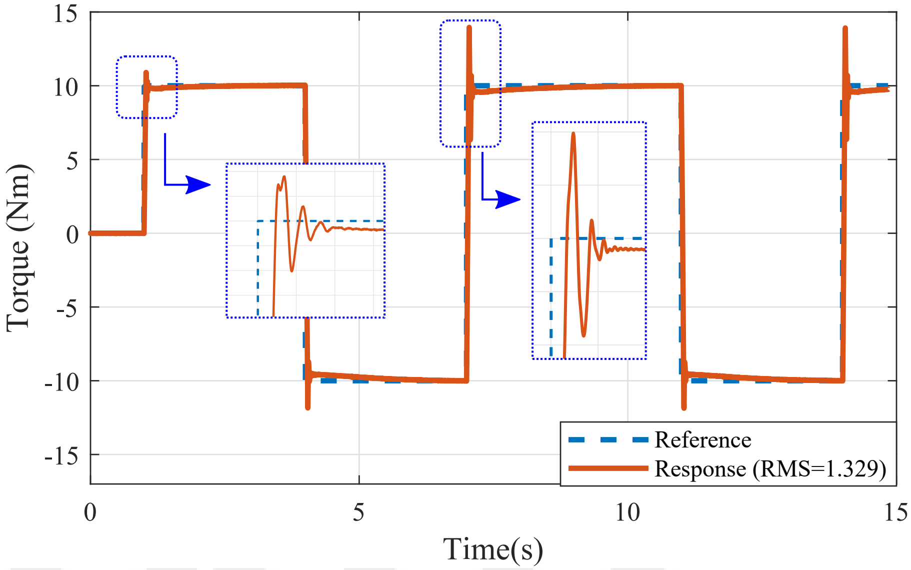


Figure 55: Square reference tracking of the Cascaded PID Controller experiment

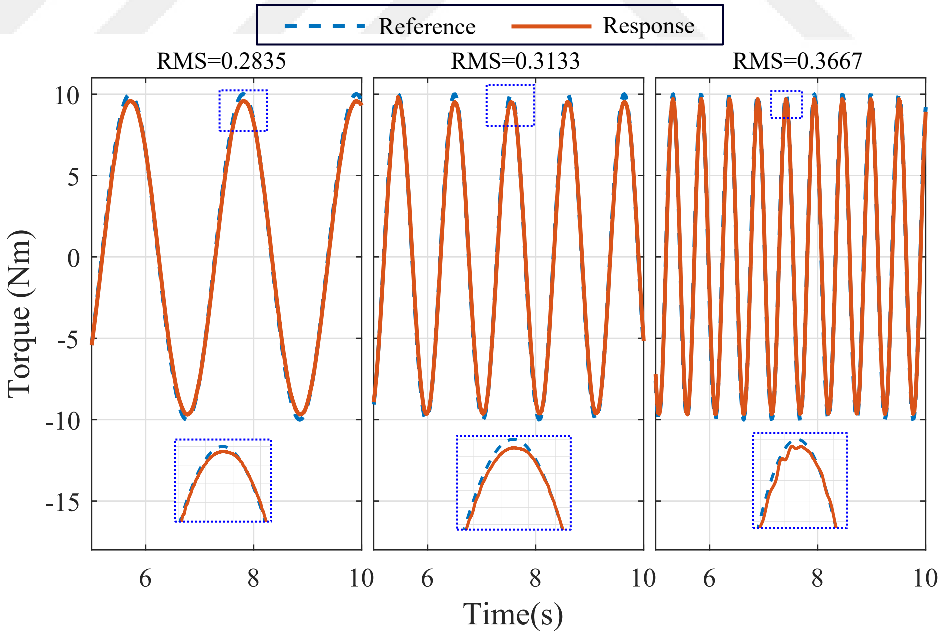


Figure 56: Sinusoidal (3,6 and 12 rad/s respectively) reference tracking of the Cascaded PID Controller experiment

Table 28: RMS errors of the Cascaded PID controller experiments

Step response	0.527
Square tracking	1.329
Stair tracking	-
Sinusoidal 3 rad/s	0.283
Sinusoidal 6 rad/s	0.313
Sinusoidal 12 rad/s	0.366
Chirp (0.5 Hz to 5 Hz)	0.452
Average	0.545

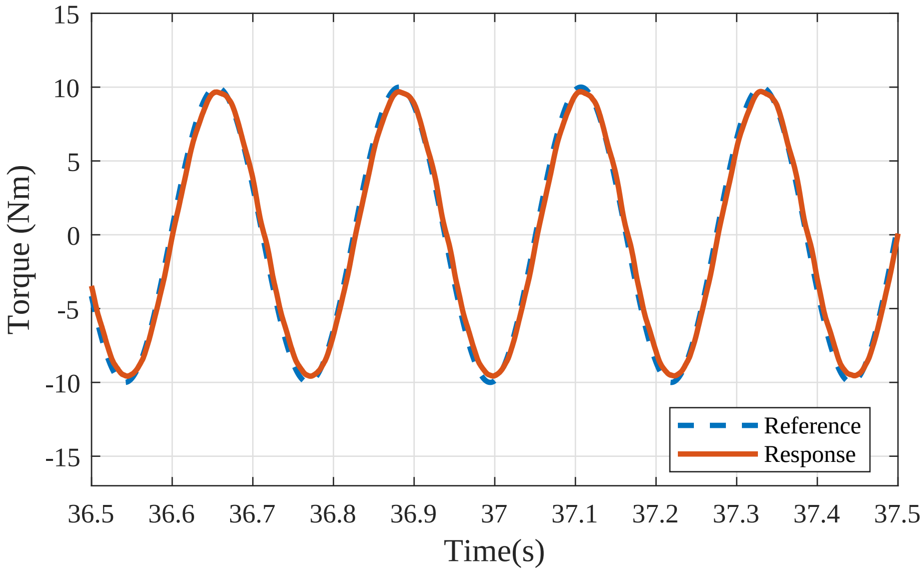


Figure 57: Sinusoidal reference (4.4 Hz) tracking of the Cascaded PID Controller experiment

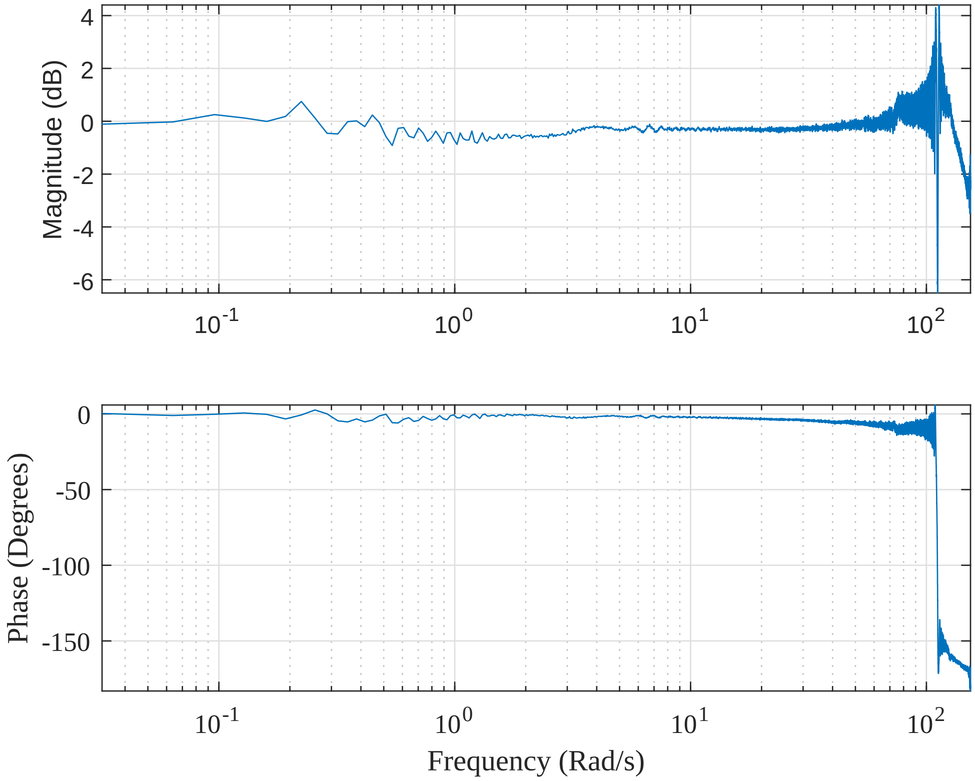


Figure 58: Bode plot of the Cascaded PID Controller experiment

6.3 Cascaded PID Controller with DoB

A DoB was implemented to the Cascaded PID controller to increase the robustness by having an accurate motor velocity control. Controller gains were reduced due to presence of the DoB, same tuning rules were inherited from [23]. Controller paremeters can be found in Table 29.

The step response of the controller can be seen in Figure 59, and step response information is presented in Table 30. Square and sinusoidal reference tracking performances can be seen in Figure 60 and 61 respectively. A part of the chirp signal that corresponds to frequency of 4.4 Hz is presented in Figure 62. RMS values can be found in Table 31.

Adding a DoB to the Cascaded PID controller reduced the RMS values in low frequency region. However, as the frequency of the reference signal is increased, RMS values are also increasing more then compared to without DoB case. Bode plot of the controller is given in Figure 63, according to the plot, controller exhibited up-to 17 Hz control bandwidth. It can be seen that the phase is shifting as the frequency increased.

Table 29: Cascaded PID Controller with DoB experiment parameters

Parameters	***Explanation***	***Value***
K_{p1}	Outer loop proportional gain	3.5
K_{i1}	Outer loop integral gain	0.6
K_{d1}	Outer loop derivative gain	0.023
w_d	Approx. Diff. Freq. PID	1600 Hz
K_{p2}	Inner loop proportional gain	0.15
K_{i2}	Inner loop integral gain	0.05
w_{θ_m}	Approx. Diff. Motor velocity	600 Hz
w_Q	DoB frequency	600 Hz

Table 30: Step response information of the Cascaded PID with DoB experiment

RMS	0.308
OS%	6.959
Settling Time	0.036 s
US%	0.487
Peak Time	0.021 s
Rise Time	0.015 s

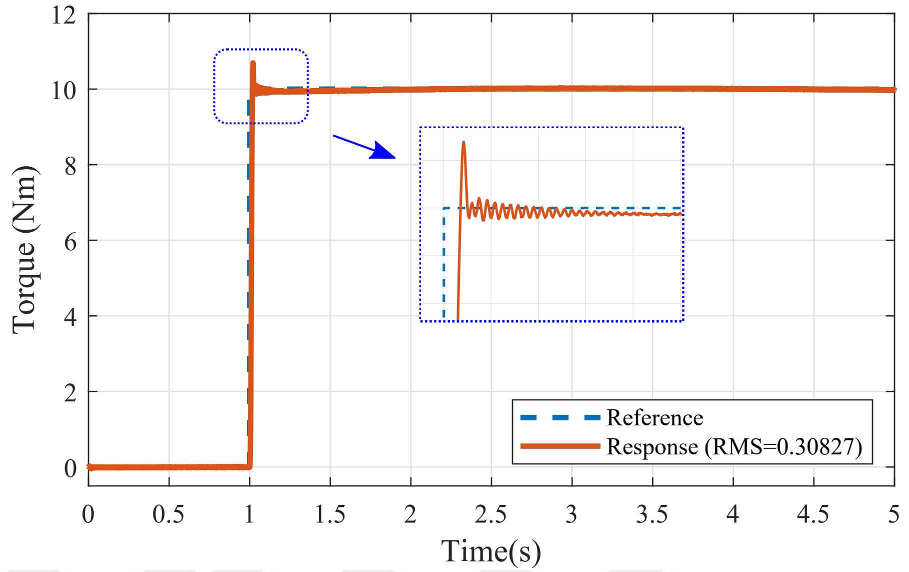


Figure 59: Step response of the Cascaded PID+DoB Controller experiment

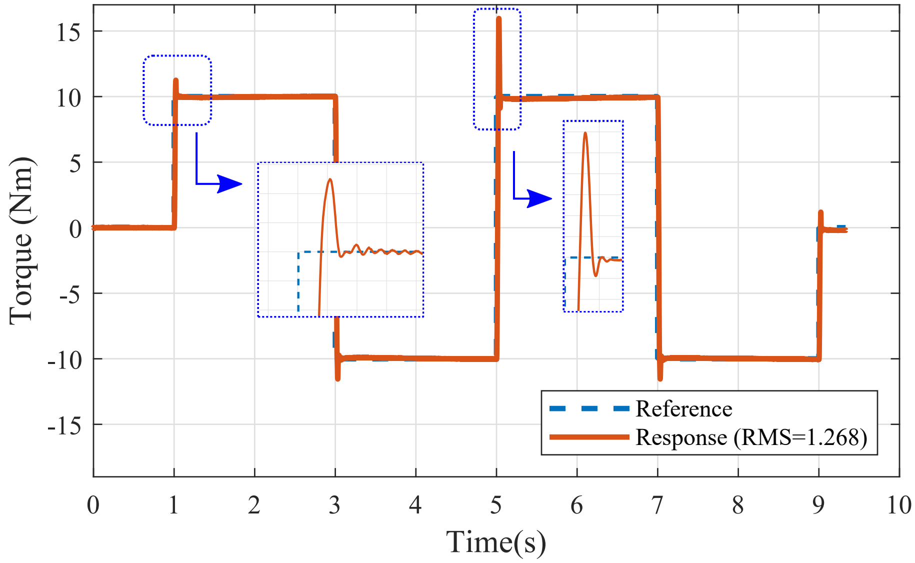


Figure 60: Square reference tracking of the Cascaded PID+DoB Controller experiment

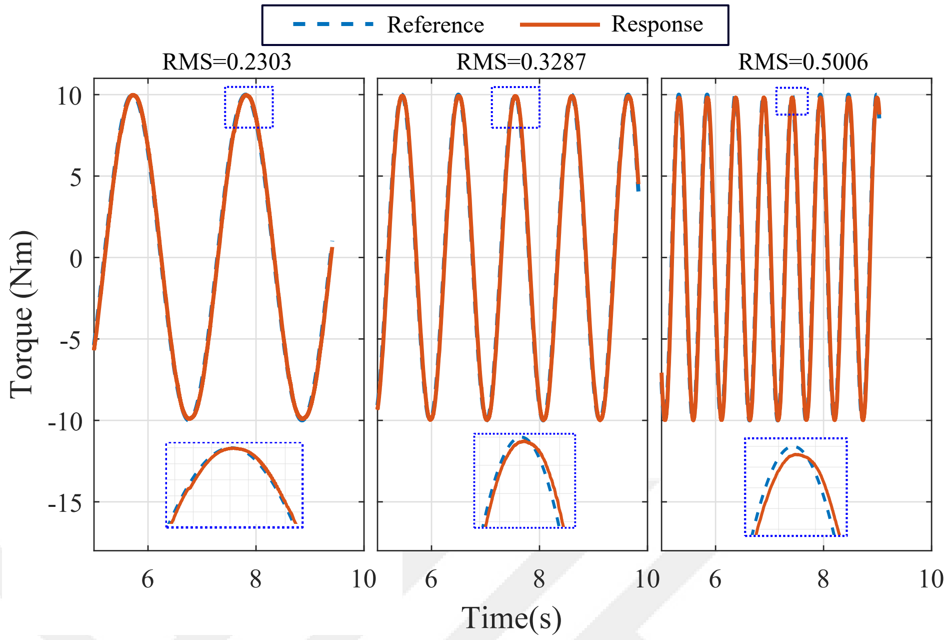


Figure 61: Sinusoidal (3,6 and 12 rad/s respectively) reference tracking of the Cascaded PID+DoB Controller experiment

Table 31: RMS errors of the Cascaded PID controller with DoB experiments

Step response	0.308
Square tracking	1.268
Stair tracking	0.209
Sinusoidal 3 rad/s	0.230
Sinusoidal 6 rad/s	0.329
Sinusoidal 12 rad/s	0.501
Chirp (0.5 Hz to 5 Hz)	0.804
Average	0.521363

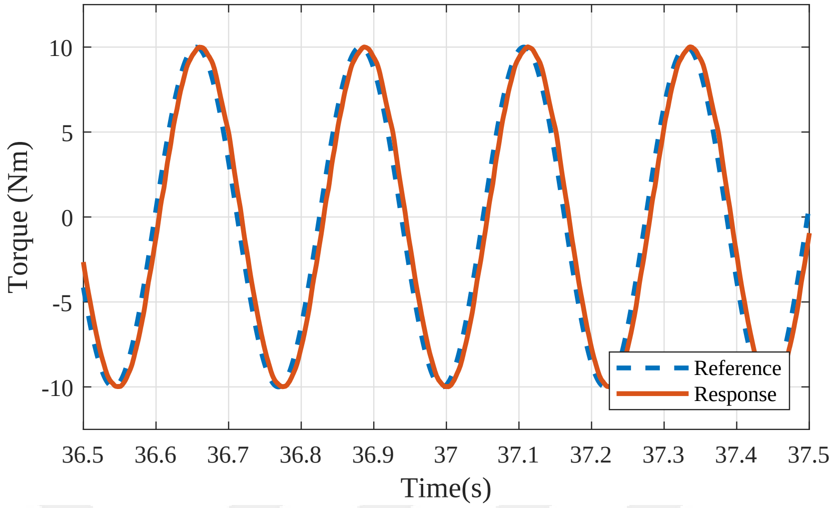


Figure 62: Sinusoidal reference (4.4 Hz) tracking of the Cascaded PID+DoB Controller experiment

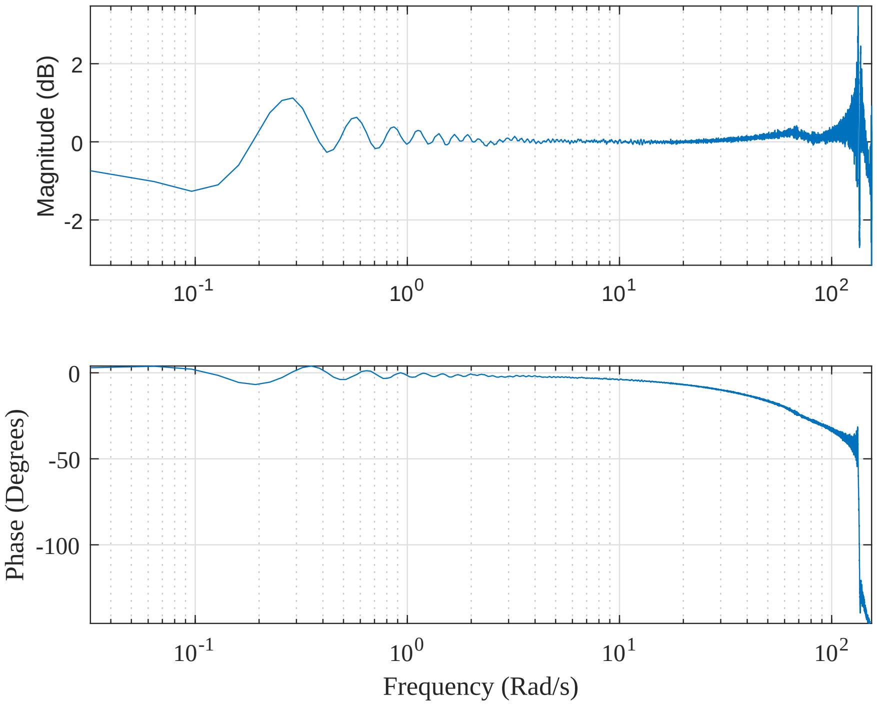


Figure 63: Bode plot of the Cascaded PID+DoB Controller experiment

6.4 PID with model based feed-forward and DoB

Identified model parameters were used to construct the nominal model of the SEA. The tuning process of the controller was performed empirically. Controller parameters can be seen in Table 32.

The step response of the controller can be seen in Figure 64, and step response information is presented in Table 33. Square and sinusoidal reference tracking performances can be seen in Figure 65 and 66 respectively. A part of the chirp signal that corresponds to frequency of 4.4 Hz is presented in Figure 67. RMS values can be found in Table 34. Bode plot of the controller is given in Figure 68, according to the plot, controller exhibited up-to 11 Hz control bandwidth.

Table 32: FF+PID+DoB experiment parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	100
K_i	Integral gain	100
K_d	Derivative gain	14
w_d	Approx. Diff. Freq. PID	300 Hz
w_{Q_2}	DoB frequency	1600 Hz
w_{Q_1}	FF frequency	628 Hz

Table 33: Step response information of the FF+PID+DoB

RMS	0.464
OS%	1.191
US%	0.019
Settling Time	0.076 s
Peak Time	0.081 s
Rise Time	0.036 s

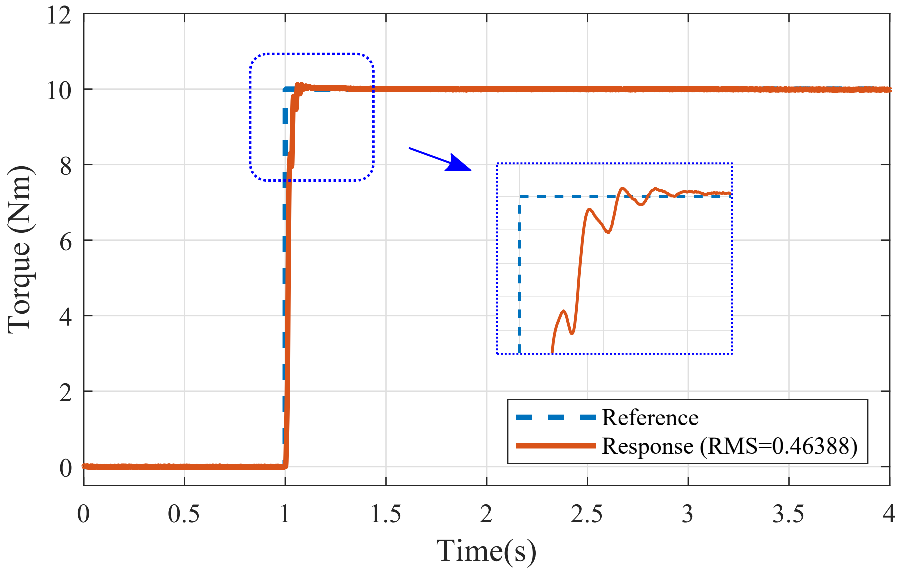


Figure 64: Step response of the FF+PID+DoB Controller experiment

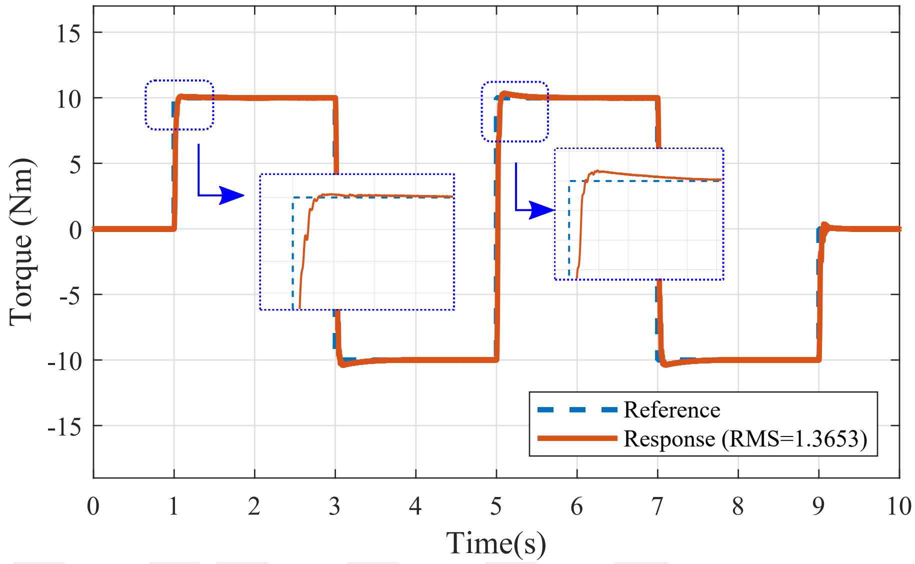


Figure 65: Square reference tracking of the FF+PID+DoB Controller experiment

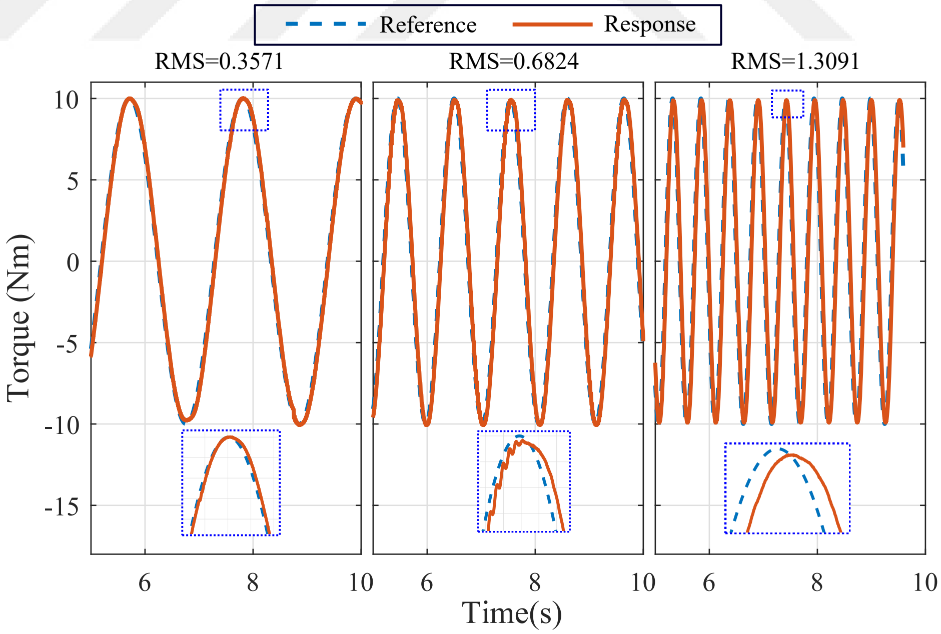


Figure 66: Sinusoidal (3,6 and 12 rad/s respectively) reference tracking of the FF+PID+DoB Controller experiment

Table 34: RMS errors of the FF+PID+DoB controller experiments

Step response	0.464
Square tracking	1.365
Stair tracking	0.228
Sinusoidal 3 rad/s	0.347
Sinusoidal 6 rad/s	0.682
Sinusoidal 12 rad/s	1.309
Chirp (0.5 Hz to 5 Hz)	1.060
Average	0.779

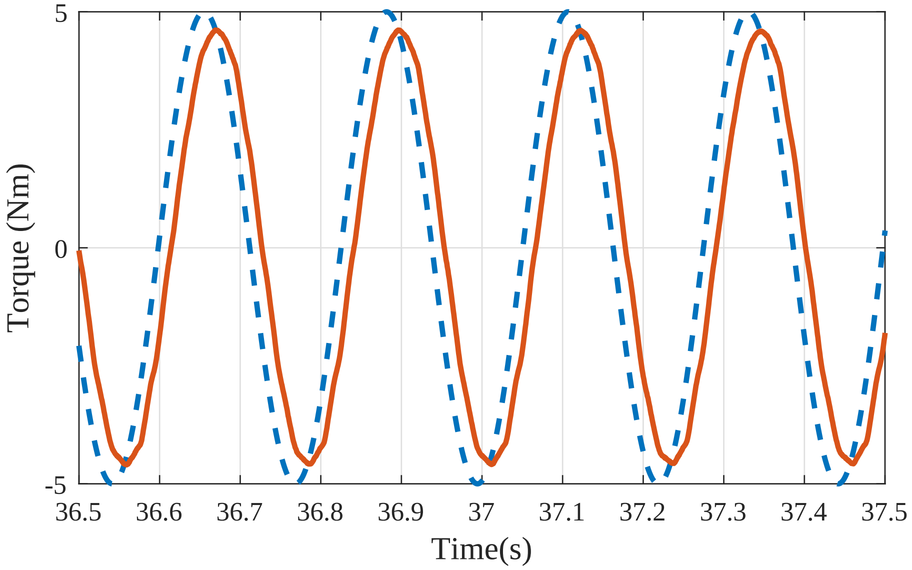


Figure 67: Sinusoidal reference (4.4 Hz) tracking of the FF+PID+DoB Controller experiment

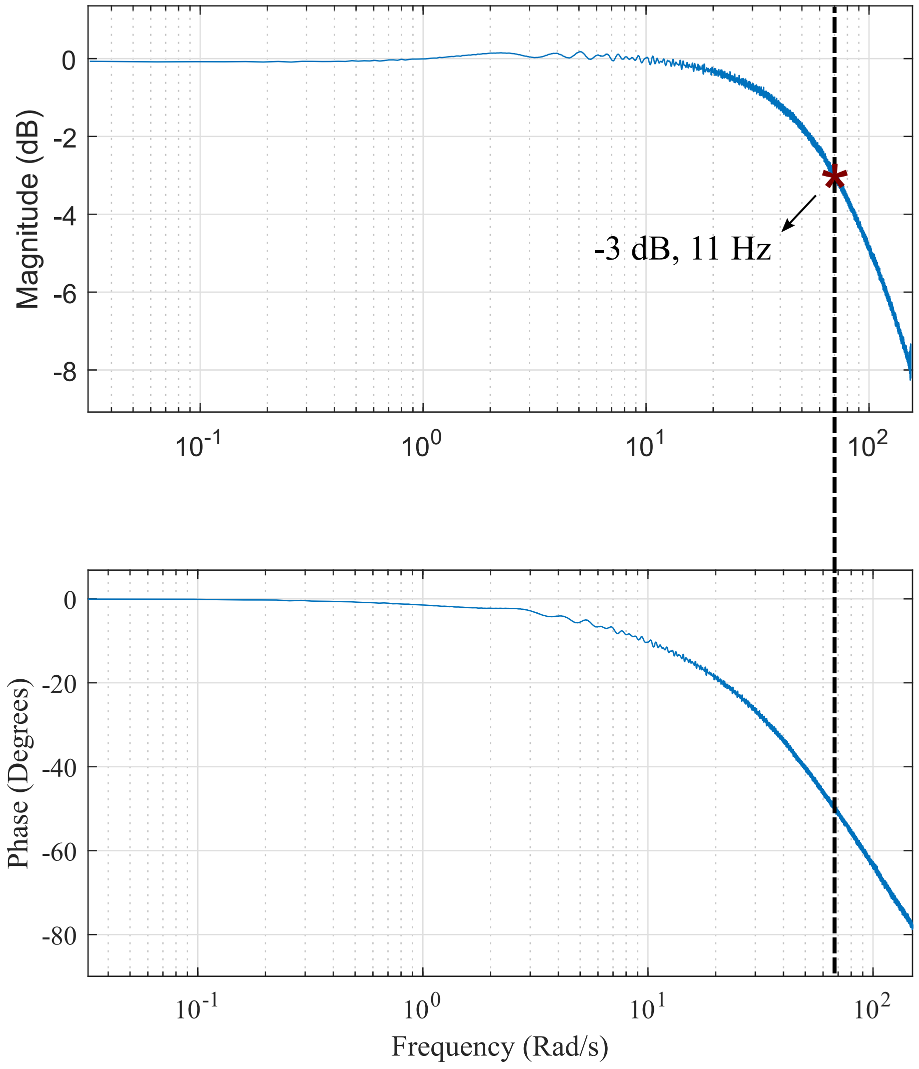


Figure 68: Bode plot of the FF+PID+DoB Controller experiment

6.5 Sliding Mode Controller with DoB

Unlike the simulation study, tuning of the SMC+DoB is relatively complicated due to physical constraints. SMC+DoB was tuned empirically by adjusting the convergence rate of the sliding surface and the controller gain. Controller parameters can be seen in Table 35.

Increasing the controller gain and the convergence rate leads more frequent switching action. However, in real implementation this switching behavior is limited by physical constraints. Moreover, chattering effect becomes more dominant.

The step response of the controller can be seen in Figure 69, and step response information is presented in Table 36. Square and sinusoidal reference tracking performances can be seen in Figure 70 and 71 respectively. A part of the chirp signal that corresponds to frequency of 4.4 Hz is presented in Figure 72. RMS values can be found in Table 37. Bode plot of the controller is given in Figure 73, according to the plot, controller exhibited up-to 19 Hz bandwidth.

Table 35: SMC+DoB experiment parameters

Parameters	***Explanation***	***Value***
c	Convergence rate	500
ρ	SMC gain	1.6
w_{θ_m}	Approx. Diff. Freq. motor velocity	1000 Hz
w_{θ_d}	Approx. Diff. Freq. torsion	1000 Hz
w_{DoB}	DoB frequency	1500 Hz

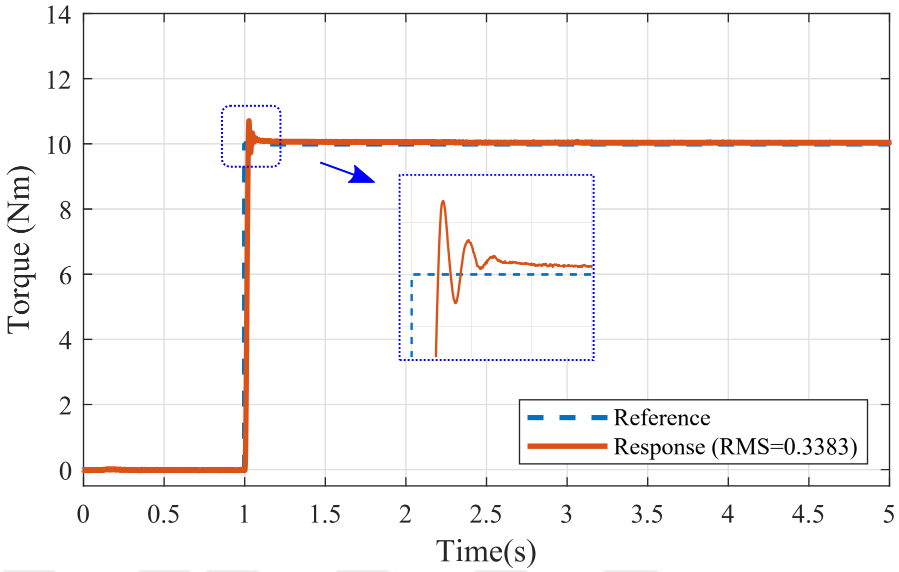


Figure 69: Step response of the SMC+DoB experiment

Table 36: Step response information of the SMC+DoB experiment

RMS	0.338
OS%	7.071
US%	2.780
Settling Time	0.053 s
Peak Time	0.026 s
Rise Time	0.020 s

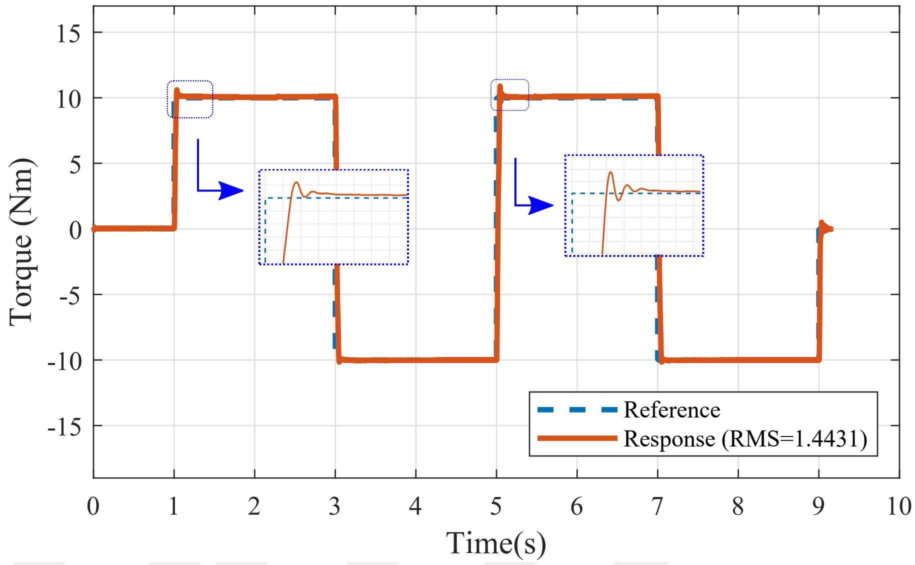


Figure 70: Square reference tracking of the SMC+DoB experiment

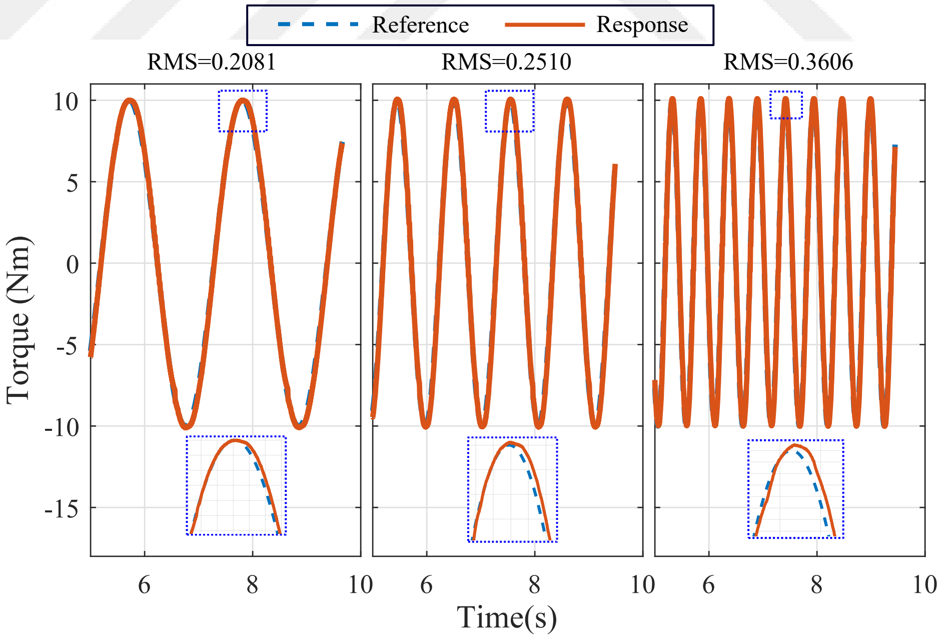


Figure 71: Sinusoidal (3,6 and 12 rad/s respectively) reference tracking of the SMC+DoB experiment

Table 37: RMS errors of the SMC+DoB experiments

Step response	0.338
Square tracking	1.443
Stair tracking	0.230
Sinusoidal 3 rad/s	0.208
Sinusoidal 6 rad/s	0.251
Sinusoidal 12 rad/s	0.361
Chirp (0.5 Hz to 5 Hz)	0.445
Average	0.468

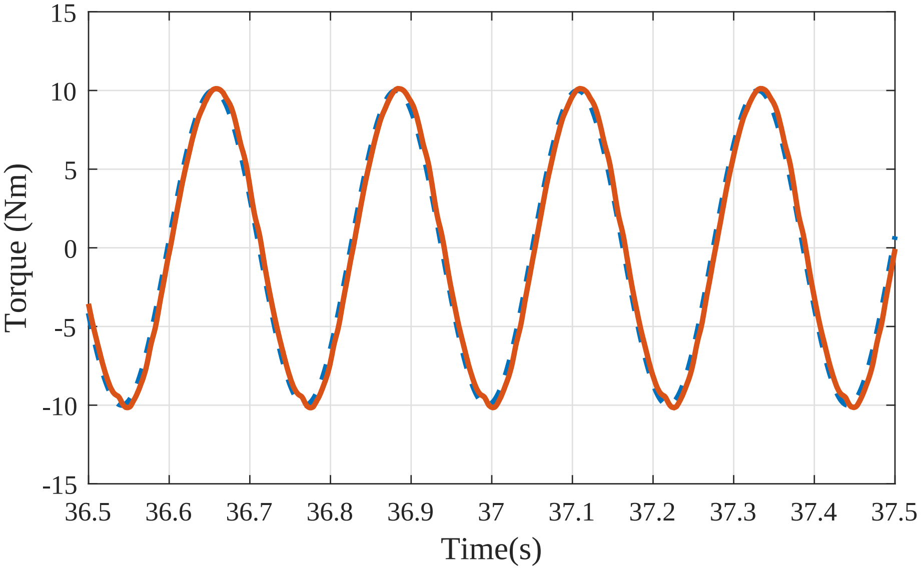


Figure 72: Sinusoidal reference (4.4 Hz) tracking of the SMC+DoB experiment

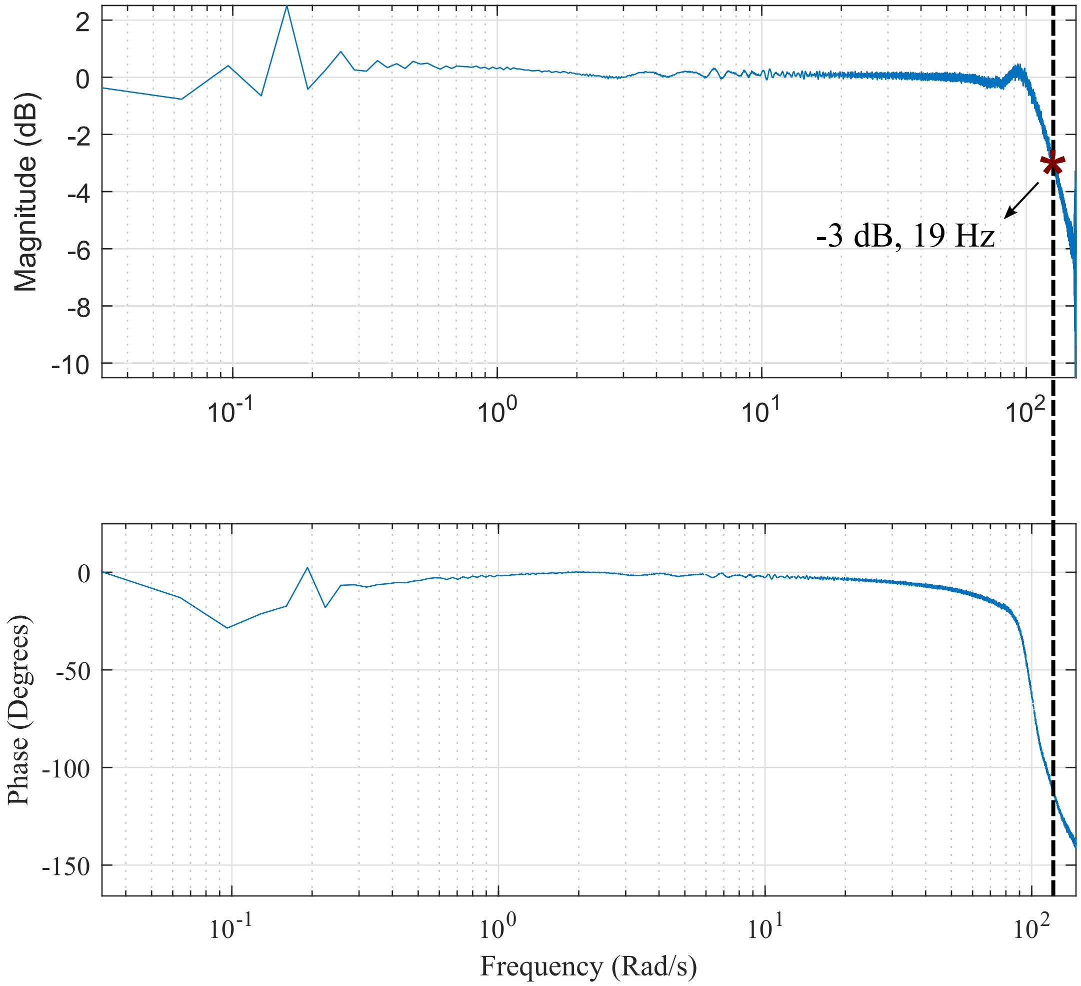


Figure 73: Bode plot of the SMC+DoB experiment

6.6 Differential Flatness Controller with DoB

Implementation of the DF+DoB method requires multiple integrator blocks, differentially flat reference generator, state feedback and an high-order DoB. Since the pole placement method was replaced with the double-PD controller, by adjusting the proportional and derivative gain of the state feedback controller, tuning was achieved empirically. Controller parameters can be seen in Table 38.

The step response of the controller can be seen in Figure 74, and step response information is presented in Table 39. Square and sinusoidal reference tracking performances can be seen in Figure 75 and 76 respectively. A part of the chirp signal that corresponds to a frequency of 4.4 Hz is presented in Figure 77. RMS values can be found in Table 40. Bode plot of the controller is given in Figure 78, according to the plot, controller exhibited up-to 13 Hz control bandwidth.

Table 38: DF+DoB experiment parameters

Parameters	***Explanation***	***Value***
K_p	Proportional gain	1.1
K_d	Derivative gain	0.21
w_d	Approx. Diff. Freq. PID	150 Hz
w_{DoB}	DoB frequency	300 Hz

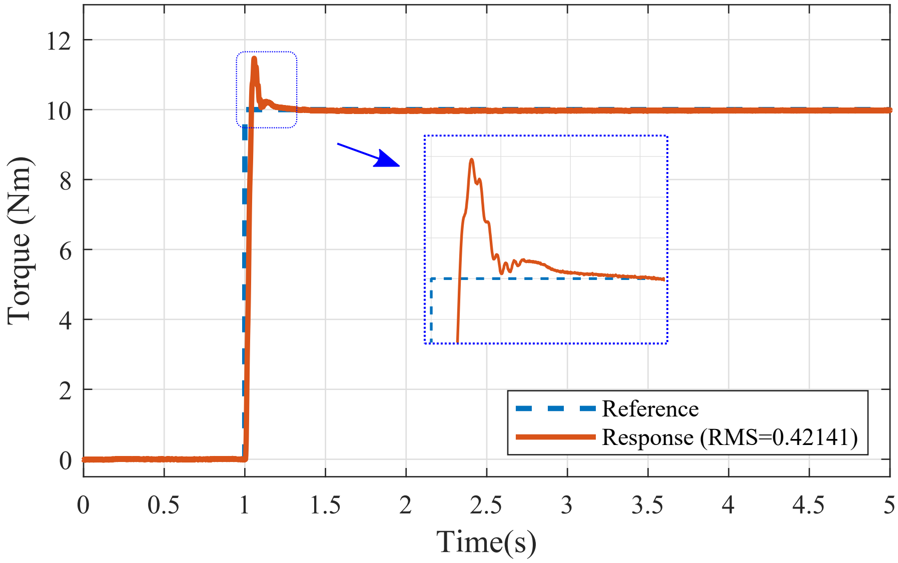


Figure 74: Step response of the DF+DoB Controller experiment

Table 39: Step response information of the DF+DoB controller experiment

RMS	0.421
OS%	14.673
US%	0.262
Settling Time	0.154 s
Peak Time	0.058 s
Rise Time	0.036 s

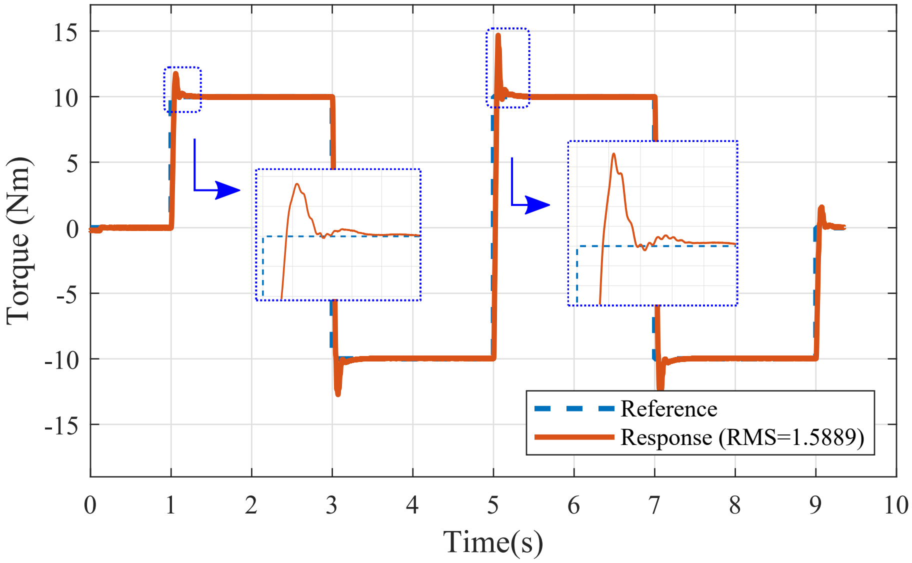


Figure 75: Square reference tracking of the DF+DoB Controller experiment

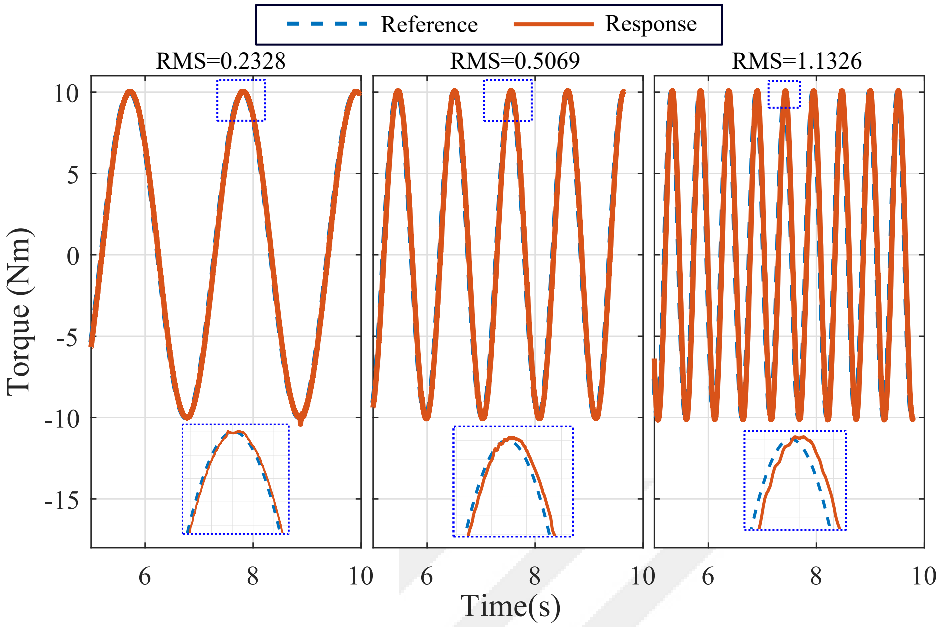


Figure 76: Sinusoidal (3,6 and 12 rad/s respectively) reference tracking of the DF+DoB Controller experiment

Table 40: RMS errors of the DF+DoB controller experiments

Step response	0.421
Square tracking	1.589
Stair tracking	0.307
Sinusoidal 3 rad/s	0.233
Sinusoidal 6 rad/s	0.507
Sinusoidal 12 rad/s	1.133
Chirp (0.5 Hz to 5 Hz)	0.849
Average	0.720

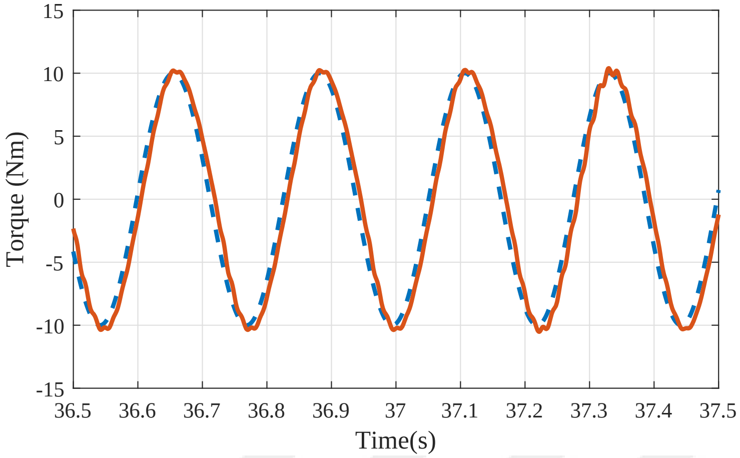


Figure 77: Sinusoidal reference (4.4 Hz) tracking of the DF+DoB Controller experiment

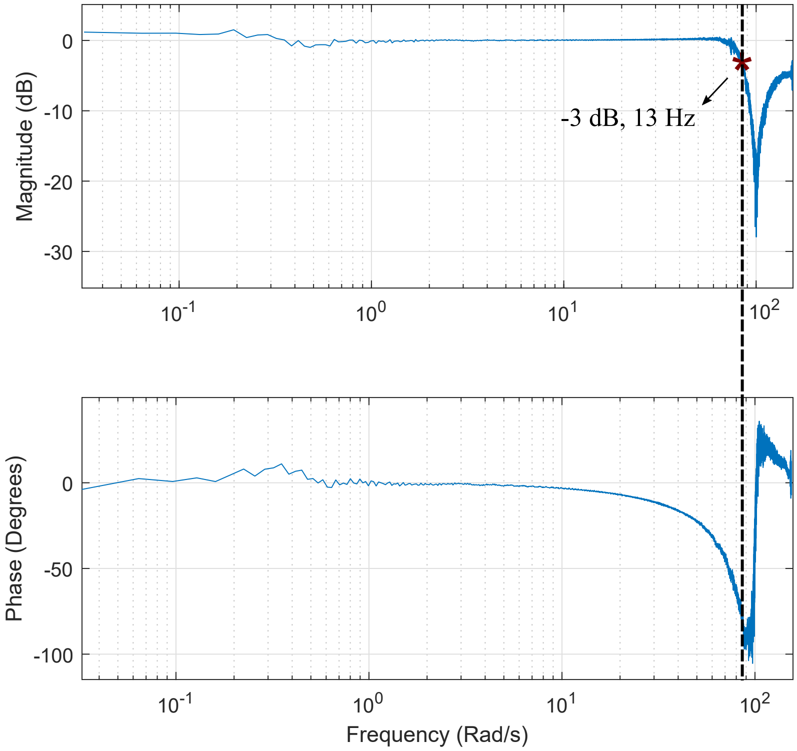


Figure 78: Bode plot of the DF+DoB Controller experiment

6.7 Discussion

In the experiments, torque control performance was only evaluated with the deflection measurement. Moreover, there were no systematic disturbance injection to the actuator. Thus, modeling uncertainty case could not be analyzed. Yet, by restricting the link movement with sponges, the effect of the environment stiffness still presented in the experiments. As the output link to squeeze the sponge, environment stiffness was increased.

Average RMS values of the controllers were calculated by using the experiment data of step, square, stair, sinusoidal and chirp reference. In Table 41, average RMS of error values are given, Figure 79 shows the same data in a bar graph alongside the simulation results. It can be seen from the figure that the experiment and simulation results are related. RMS of the error was the main criteria for the comparison study. Also the bode plots provided information regarding the control bandwidth of the controllers.

Table 41: Average RMS comparison of experiments

Controller	***RMS***
Cascaded PID	0.545
Cascaded PID + DoB	0.521
FF+PID+DoB	0.779
SMC+DoB	0.468
DF+DoB	0.720

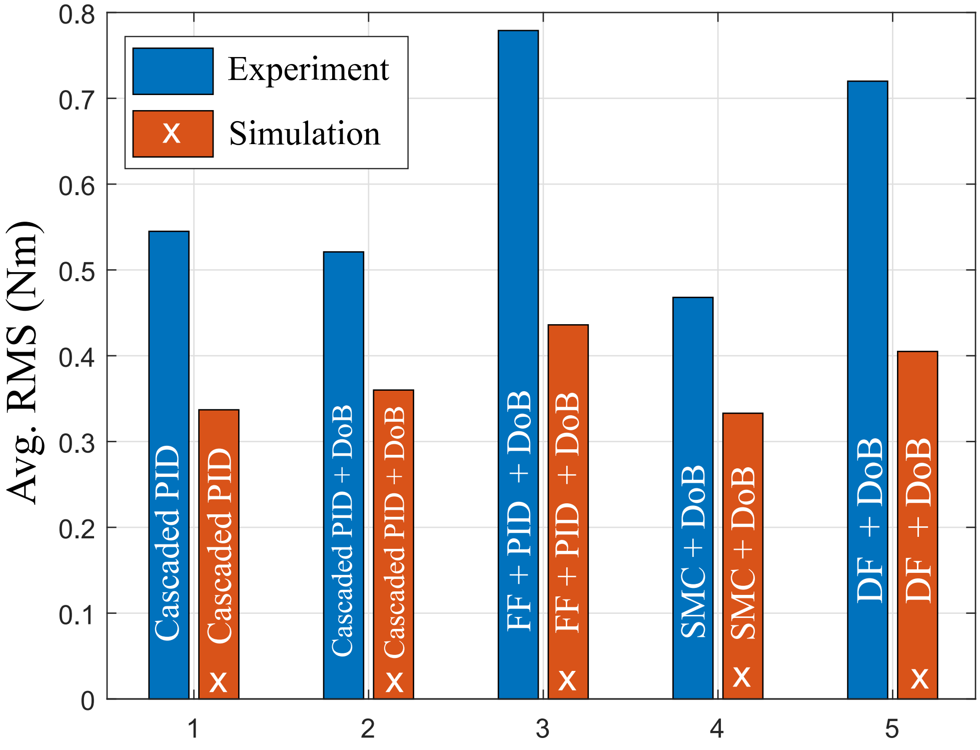


Figure 79: Average RMS comparison

Tuning of the controllers was empirical. There were numerous concerns during the tuning process. First of all, control region of the deflection was limited due to relatively stiff spring design. Considering the the equation $\tau_d = K\theta_d$, as the spring stiffness is increased, to be able to control the output torque of the actuator controlling the deflection has to be performed in a smaller region. Therefore, controlling the small deflection is highly suffering from the noise sensitivity. Also, studied control methods requires the derivative of the noisy data. Even though, the approximate differentiation technique filters the signal and allows higher derivative gains to be used, there is still a major trade-off. Cascaded control schemes and SMC+DoB was highly effected by the measurement noises. Also SMC+DoB method is known with chattering phenomena, adding a DoB reduced the chattering effect successfully, yet it was not eliminated completely. In both simulations and the experiments, SMC+DoB exhibited the lowest RMS values. However, chattering problem may degrade the comfort for the human wearable robotic systems.

As previously mentioned, PID controller exhibited poor performance therefore it was not a part of the comparison. Cascaded PID controller was tuned according to stability boundaries defined in [23]. As it can be seen in Figure 80, adding a DoB to the Cascaded PID controller improved the tracking performance in the low-frequency region, as the frequency increases, the RMS of the errors were increased noticeably. As the frequency of the DoB increased, the tracking performance may further be improved, however DoB frequency is limited by physical effects such as noise and sampling rate.

In Table 42, RMS errors are given for; 3 rad/s sinusoidal, 6 rad/s sinusoidal, 12 rad/s sinusoidal and chirp signal with frequency range of 0.5 Hz to 5 Hz, respectively. RMS values are visualized in Figure 80 in order to be perceived easily by the reader. Step responses of the controllers are given in Figure 81.

Table 42: RMS comparison of sinusoidal tracking simulations

Controller	***3 rad/s***	***6 rad/s***	***12 rad/s***	***Chirp (0.5-5 Hz)***
Cascaded PID	0.283	0.313	0.366	0.452
Cascaded PID + DoB	0.228	0.329	0.501	0.804
FF+PID+DoB	0.347	0.682	1.309	1.060
SMC+DoB	0.208	0.251	0.361	0.445
DF+DoB	0.233	0.507	1.133	0.849

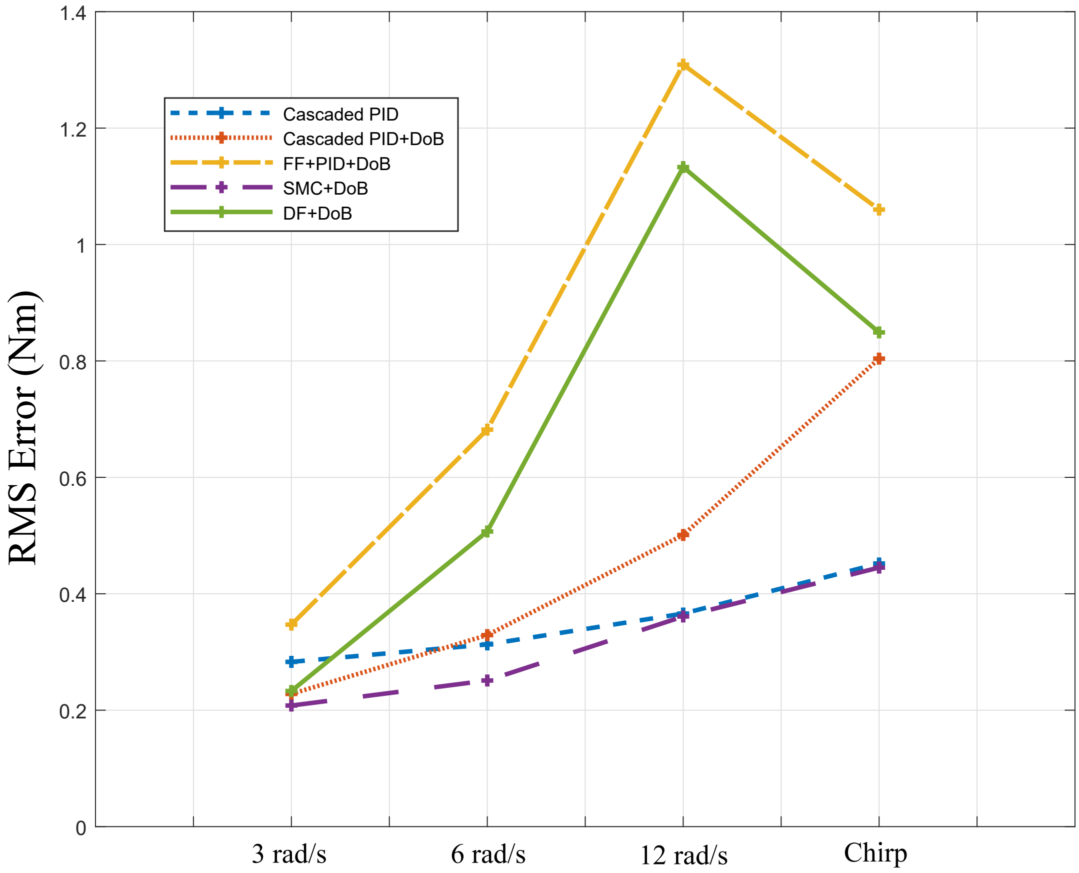


Figure 80: RMS comparison of sinusoidal tracking simulations with a graph

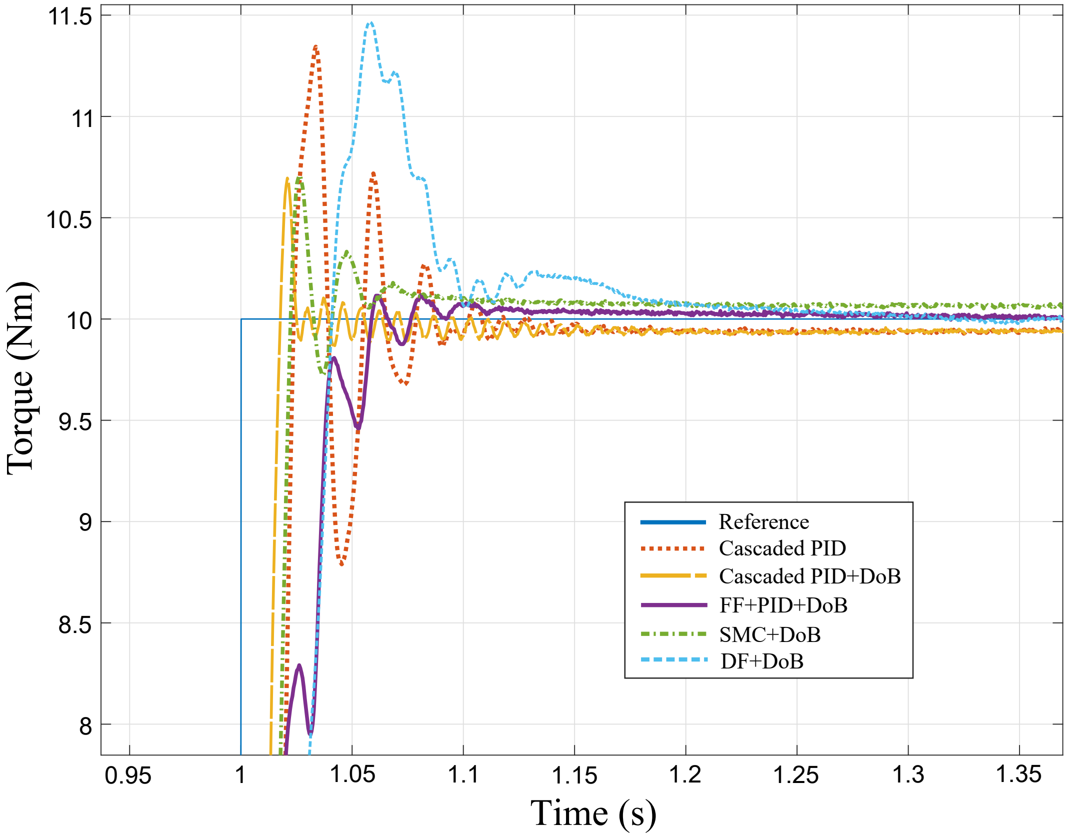


Figure 81: Experimental step response comparison of all controllers

Step response informations of all controllers are given in Table 43 and step responses are plotted in the same figure, see Figure 81. DF and FF+PIF+DoB controller eliminated the steady state error earlier then other control methods, however RMS results of both controllers has relatively poor performance compared to other controllers.

FF+PID+DoB controller was tuned according to the nominal plant, fine tuning was achieved empirically. Increasing the controller gain led to instability. FF+PID+DoB has an advanced disturbance rejection property thanks to its DoB structure applied to the deflection. However tuning of the FF+PID+DoB was problematic due to measurement issues. FF+PID+DoB exhibited relatively high RMS of error values. On the contrary, tuning of the DF controller was straightforward.

Table 43: Step response comparison of experiments

Controller	***RMS***	***OS%***	***Settling Time***
Cascaded PID	0.527	10.179	0.064 s
Cascaded PID + DoB	0.308	6.959	0.036 s
FF+PID+DoB	0.464	1.191	0.076 s
SMC+DoB	0.338	7.071	0.053 s
DF+DoB	0.421	14.673	0.154 s
	US%	***Peak Time***	***Rise Time***
Cascaded PID	11.6	0.034 s	0.022 s
Cascaded PID + DoB	0.487	0.021 s	0.015 s
FF+PID+DoB	0.0187	0.081 s	0.036 s
SMC+DoB	0.262	0.026 s	0.020 s
DF+DoB	0.262	0.058 s	0.036 s

Bandwidths of the controllers are compared in Table 44.

Table 44: Bandwidth comparison of controllers

Controller	***RMS***
Cascaded PID	17 Hz
Cascaded PID + DoB	17 Hz
FF+PID+DoB	11 Hz
SMC+DoB	19 Hz
DF+DoB	13 Hz

6.8 Zero Torque Control

Zero Torque Control (ZTC) experiment was conducted with only one method that is feed-forward, PID and DoB controller (FF+PID+DoB). Since the experiment results are only presented for a single controller, this section of the thesis is not a part of the comparison study. In this experiment, zero torque was given to the controller as a reference to realize the zero output impedance. FF+PID+DoB controller was used in the experiments due to its superior disturbance rejection property. Applying a DoB directly to the deflection improved the robustness against external disturbances. The results can be seen in Figure 82. Dotted line represents the link movement while the line is the output torque of the actuator.

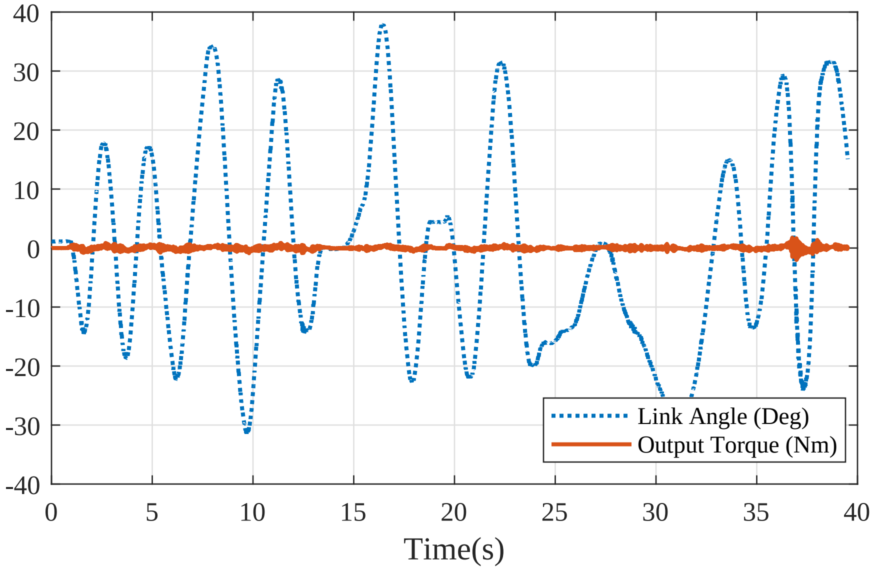


Figure 82: Zero torque control performance of FF+PIF+DoB

CHAPTER VII

CONCLUSION

This thesis presented a comparison study on various robust control methods. In the scope of this study, simulations and the practical implementations of the advanced control methods were investigated.

Modifications on the control schemes were made to increase the tracking performances of the controllers. In the sense of RMS error of tracking, SMC+DoB outperformed the other controller designs in both simulations and experiments. However, in the presence of modeling errors, FF+PID+DoB is successfully compensated the effects according to the simulations. Also, in the experiments it can be said that, disturbance rejection property of the FF+PID+DoB is superior compared to the other methods. However, FF+PID+DoB method exhibited poor bandwidth and larger RMS errors compared to other methods.

Presence of DoB in the system increases the robustness as expected. Adding a DoB to the Cascaded PID control scheme slightly reduced the RMS error in simulations. However, improvement of the performance was more noticeable in real implementations. Moreover, DoB decreased the control effort of the PID controller and the vibrations in the systems were partly reduced.

Implementation of the DF+DoB controller is relatively complicated compared to the other methods. However, tuning of the DF controller was straightforward with the modifications made in this thesis.

The frequency analysis of control methods were presented both in simulations and experiments. The bandwidths of the controllers were limited due to physical restrictions such as motor saturation. Cascaded control algorithms exhibited the highest control bandwidths. While FF+PID+DoB has the lowest bandwidth.

Considering all control methods that were investigated under this thesis, one of the main issue was the high stiffness of the torsional spring. Control problem of the output torque was limited in a small working region of deflection due to high stiffness. However, all robust control methods were successfully overcome the various limitations. It can be said that, model based control algorithms can benefit from design upgrades.

Appendices

APPENDIX A

GRAPHICAL USER INTERFACE

Controllers were written as C programs to run on RTOS. Without the GUI, following steps were needed to tune the controller; editing the program code, transferring codes to the RPi, compiling and executing the program, retrieving the results and visualizing the data on the host PC. Graphical user interface was designed to facilitate the process. GUI can be seen in Figure 7. GUI was designed in MATLAB for simplicity, portability and manipulability.

In the design, GUI expects user to fill the empty label boxes which are controller specific parameters e.g. FF+PID+DoB expects model parameters, controller gains and DoB frequency. Every button has a callback function, these callback functions contains MATLAB codes and they contain systems calls to the script files. In this section, Run callback function and Run system call codes are presented.

A.1 Run button callback function

```
function RunButtonPushed(app, event)
app.Label_2.Text = ’ ’; % Empty the textbox


% This is for the program run delay
app.Label.Text = strcat(’Waiting for ’, ’
  ’,string(app.RunDelaysSlider.Value),’ seconds’);
pause(app.RunDelaysSlider.Value);
app.Label.Text = ’Running..’;


% Clear the screen and run the program
```

```
cla(app.UIAxes);
system(’runGeneral.bat’)

%Importing and processing the experiment data
filename = ’General.log’;
delimiterIn = ’ ’;
headerlinesIn = 1;
dataTemp = importdata(filename,delimiterIn,headerlinesIn);
dataTemp = dataTemp.data;
field1 = ’Time’;  value1 = dataTemp(:,1);
field2 = ’Torsion’;  value2 = dataTemp(:,2);
field3 = ’MotorAngle’;  value3 = dataTemp(:,3);
field4 = ’Reference’;  value4 = dataTemp(:,4);
field5 = ’InputSignal’;  value5 = dataTemp(:,5);
expDataGeneraldata = struct(field1,value1,field2,value2,field3,
value3,field4,value4,field5,value5);
assignin(’base’, ’expDataGeneral’, expDataGeneraldata);
formatOut = ’dd_mm_yy_hh_MM_SS’;
datetimestr = datestr(now,formatOut);
titleS = strcat(’expDataGeneral’, datetimestr);
save(titleS,’expDataGeneraldata’);
app.status = 1;
app.Label.Text = ’Program finished!’;

% Plot the results to the screen
app.expData = expDataGeneraldata;
plot(app.UIAxes,app.expData.Time,app.expData.Torsion);
hold(app.UIAxes);
plot(app.UIAxes,app.expData.Time,app.expData.Reference);
```

```
app.UIAxes.YLabel.String = 'Torque (Nm)';

app.UIAxes.XLabel.String = 'Time (s)';

app.whichPlot = 3;

end
```

A.2 Run system call (runGeneral.bat)

```
plink -pw myPassword -t CoExSEA@192.168.137.115 "cd /home/CoExSEA/Main &&
 echo -e myPassword | sudo -S ./run"

pscp -r -pw myPassword CoExSEA@192.168.137.115:/home/CoExSEA/Main/General.log
 C:\Receiver
```

APPENDIX B

PROGRAM CODES

B.1 PID Controller Code

```
typedef struct PIDController_{

double KpGain;

double KiGain;

double KdGain;

double IntegralTerm;

double error[2];

double errorDot;

}PIDController;

PIDController PID;


double fPID(double Qdref, double Qd, double QdDotref, double QdDot){

PID.error[0] = Qdref-Qd;

PID.errorDot = QdDotref-QdDot;

PID.IntegralTerm = PID.IntegralTerm +

PID.IntegratorNum*(PID.error[0]+PID.error[1]);


return PID.KpGain*PID.error[0] + PID.KiGain*PID.IntegralTerm +

PID.KdGain*PID.errorDot;

}
```

B.2 Bilinear Transformation Code

This is a reference for implementing a transfer function via bilinear transformation. Low-pass filter, Butterworth filter, feed-forward terms, DoBs can be implemented in the same way.

```
typedef struct tustinBlock_{
double num[6];
double denum[6];
double output[6];
double input[6];
double freq;
}tustinBlock;
tustinBlock LPF;


double LPFBlock(double inPart){
double LPFOut;
LPF.freq = 1600;
LPF.num[0] = 1/(1 + 2/(LPF.freq*0.0005));
LPF.num[1] = LPF.num[0];
LPF.denum[0] = (1 - 2/(LPF.freq*0.0005)) * LPF.num[0];
/* Diff equation */
LPF.input[0] = inPart;
LPFOut = TorsionLPF.num[0] * LPF.input[0] + LPF.num[1] *
LPF.input[1] - LPF.denum[0]*LPF.output[0];
/* Update states */
LPF.input[1] =  LPF.input[0];
LPF.output[0] = LPFOut;
return LPFOut;
}
```

B.3 Approximate Differentiation Code

```
typedef struct tustinBlock_{
double num[6];
double denum[6];
double output[6];
double input[6];
double freq;
}tustinBlock;
tustinBlock TorsionDerivative; // LPF

double TorsionDerivativeBlock(double inPart){
// Input is torsion (rad) output is torsion velocity (rad/s)
double TorsionDerivativeOut;
/* Diff equation */
TorsionDerivative.input[0] = inPart*TorsionDerivative.freq;
TorsionDerivativeOut = TorsionDerivative.num[0] *
TorsionDerivative.input[0] + TorsionDerivative.num[1] *
TorsionDerivative.input[1] -
TorsionDerivative.denum[0]*TorsionDerivative.output[0];
/* Update states */
TorsionDerivative.input[1] =  TorsionDerivative.input[0];
TorsionDerivative.output[0] = TorsionDerivativeOut;
return inPart*TorsionDerivative.freq-TorsionDerivativeOut;
}
```

B.4 Sliding Mode Controller Block

```
double SMCCompute(double QdRef, double QdRefDot, double QdRefDotDot, ...
 double C, double P){
double eps = 0.1;
SMC.error = QdRef-SEA.torsion;
SMC.errorDot = QdRefDot-SEA.torsionVel;
SMC.sigma = SMC.errorDot + C * SMC.error;
double W1 = -(SEAnominal.Kn/(SEAnominal.Jmn*SEAnominal.Nm*SEAnominal.Nm) +
 SEAnominal.Kn/SEAnominal.Jln);
double W2 = (SEAnominal.Jln*SEAnominal.Bmn-SEAnominal.Jmn*SEAnominal.Bln)/
(SEAnominal.Jmn*SEAnominal.Jln*SEAnominal.Nm);

return  P*(SMC.sigma/(fabs(SMC.sigma) + eps)) + SEAnominal.Jmn*(QdRefDotDot
 - W1*SEA.torsion + W2* SEA.motorVel/SEAnominal.Nm +
  (SEAnominal.Bln/SEAnominal.Jln)*SEA.torsionVel + C*(SMC.errorDot));
}
```

B.5 Differential Flatness and high-order DoB Blocks

```
double DFUpdate(double TdRef, double TdRefDot, double TdRefDotDot){

DF.ydfo_dd =  (Nm/Jln)*(Kn_1*TdRef-(Kn_1*Nm)*DOB.Tdist[1]
 -((Bln+Benv)/Nm)*DF.ydfo_d -(Kenv/(Nm*Nm))*DF.ydfo);
DF.ydfo_ddd =  (Nm/Jln)*(Kn_1*TdRefDot-(Kn_1*Nm)*DOB.Tdist[3]
 -((Bln+Benv)/Nm)*DF.ydfo_dd -(Kenv/(Nm*Nm))*DF.ydfo_d);
DF.ydfo_dddd =  (Nm/Jln)*(Kn_1*TdRefDotDot-(Kn_1*Nm)*DOB.Tdist[5]
 -((Bln+Benv)/Nm)*DF.ydfo_ddd -(Kenv/(Nm*Nm))*DF.ydfo_dd);

DF.xRef[0] = (Jln*DF.ydfo_dd) + (Bln+Benv)*DF.ydfo_d +
 ((Kn+Kenv)/Nm)*DF.ydfo + (Nm*Nm*DOB.Tdist[1]/Kn);
```

```
DF.xRef[1] = (Jln*DF.ydfo_ddd) + (Bln+Benv)*DF.ydfo_dd +
 ((Kn+Kenv)/Nm)*DF.ydfo_d + (Nm*Nm*DOB.Tdist[3]/Kn);
DF.xRef[2] = Kn*Nm_2*DF.ydfo;
DF.xRef[3] = Kn*Nm_2*DF.ydfo_d;

DF.uRef = (Jmn*Jln)*DF.ydfo_dddd + (Jmn*(Bln+Benv) + Bmn*Jln)*DF.ydfo_ddd +

(Jln*Kn*Nm_2 + Jmn*(Kn+Kenv)*Nm_1+(Bln+Benv)*Bmn)*DF.ydfo_dd +
 ((Kn*Nm_2)*(Bln+Benv)+Nm_1*(Kn+Kenv)*Bmn)*DF.ydfo_d +

DOB.Tdist[0]/Nm + DOB.Tdist[1] + Bmn*Nm*Nm*Kn_1*DOB.Tdist[3] +
 Jmn*Nm*Nm*Kn_1*DOB.Tdist[5];

for (int i = 0; i < 4; i++)
{
DF.ydfo_d = DF.ydfo_d + DF.IntegratorNum*(DF.ydfo_dd + DF.ydfo_dd_Pre);
DF.ydfo = DF.ydfo + DF.IntegratorNum*(DF.ydfo_d + DF.ydfo_d_Pre);
DF.ydfo_dd_Pre = DF.ydfo_dd;
DF.ydfo_d_Pre = DF.ydfo_d;
}

return DF.uRef - ( fPID1(DF.xRef[0], DOB.states[0]) + fPID2(DF.xRef[2],
 DOB.states[2]) );
}

void DOBUpdate(double Tm){
for (int i = 0; i < 4; i++)
{
for (int j = 0; j < 4; j++)
```

```
{
DOB.Tempx[i] += DOB.A[i][j]*DOB.states[j];
}
DOB.Tempx[i] = DOB.Tempx[i] + DOB.B[i]*Tm + DOB.states[i]*DOB.LA[0];
}

for (int i = 0; i < 4; i++)
{
DOB.est_dz1[i] = -DOB.LA[0]*DOB.est_z1[i] + DOB.est_z2[i] +
 DOB.LA[0]*(DOB.Tempx[i]) - DOB.LA[1]*DOB.states[i];
DOB.est_dz2[i] = -DOB.LA[1]*DOB.est_z1[i] + DOB.est_z3[i] +
 DOB.LA[1]*(DOB.Tempx[i]) - DOB.LA[2]*DOB.states[i];
DOB.est_dz3[i] = -DOB.LA[2]*DOB.est_z1[i] + DOB.LA[2]*(DOB.Tempx[i]);
}

for (int i = 0; i < 4; i++)
{
DOB.est_z1[i] = DOB.est_z1[i] + DOB.IntegratorNum*(DOB.est_dz1[i]+DOB.est_dz1_pre[i]);
DOB.est_z2[i] = DOB.est_z2[i] + DOB.IntegratorNum*(DOB.est_dz2[i]+DOB.est_dz2_pre[i]);
DOB.est_z3[i] = DOB.est_z3[i] + DOB.IntegratorNum*(DOB.est_dz3[i]+DOB.est_dz3_pre[i]);
DOB.est_dz1_pre[i] = DOB.est_dz1[i];
DOB.est_dz2_pre[i] = DOB.est_dz2[i];
DOB.est_dz3_pre[i] = DOB.est_dz3[i];
}

for (int i = 0; i < 4; i++)
{
DOB.est_Tdis[i]   = DOB.est_z1[i] - DOB.LB[0]*DOB.states[i];
DOB.est_dTdis[i]  = DOB.est_z2[i] - DOB.LB[1]*DOB.states[i];
DOB.est_ddTdis[i] = DOB.est_z3[i] - DOB.LB[2]*DOB.states[i];
}

DOB.Tdist[0] = Jm*100*DOB.est_Tdis[1]; //d1
DOB.Tdist[1] = Jl*0.01*DOB.est_Tdis[3];  //d2
DOB.Tdist[2] = Jm*100*DOB.est_dTdis[1];  //d1 dot
DOB.Tdist[3] = Jl*0.01*DOB.est_dTdis[3];  // d2 dot
DOB.Tdist[4] = Jm*100*DOB.est_ddTdis[1]; // d1 dot dot
DOB.Tdist[5] = Jl*0.01*DOB.est_ddTdis[3]; // d2 dot dot

for (int j = 0; j < 4; j++){
DOB.Tempx[j] = 0;
}
}
```

B.6 SEALib: Custom Library

```
int setConfig(void){
if (!bcm2835_init())
return 1;
bcm2835_spi_begin();
DISABLEMultiplexer();
bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_LSBFIRST);
bcm2835_spi_setDataMode(BCM2835_SPI_MODE1);
bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_1024);
bcm2835_gpio_fsel(SPICS, BCM2835_GPIO_FSEL_OUTP);
bcm2835_gpio_write(SPICS, HIGH);
bcm2835_gpio_fsel(SPICSx, BCM2835_GPIO_FSEL_OUTP);
bcm2835_gpio_write(SPICSx, LOW);
bcm2835_gpio_fsel(D_RDY, BCM2835_GPIO_FSEL_INPT);
```

```
bcm2835_gpio_set_pud(D_RDY, BCM2835_GPIO_PUD_UP);

bcm2835_gpio_fsel(PIN_CS, BCM2835_GPIO_FSEL_OUTP);

bcm2835_gpio_fsel(PIN_EN, BCM2835_GPIO_FSEL_OUTP);

bcm2835_spi_chipSelect(BCM2835_SPI_CS0);

bcm2835_spi_setChipSelectPolarity(BCM2835_SPI_CS0, LOW);

}


void DACWrite(void) {

Write_DAC8532(0x30, Voltage_Convert(5.0,(10 + (SEA.inputSignal)/Kt)/4));

}


void EmergencyAutoSwitch(int maxPermissibleCommand, int limit){

/* This function is an soft emergency switch, if the input signal

 is higher than maxPermissibleCommand,it increases motor emergency

  condition by one and if motor emergency condition is higher than

  limit;function will reset motor input until the end of the program.

Default limit is 58

*/

if(SEA.inputSignal>maxPermissibleCommand){

experiment.MotorEmergency = experiment.MotorEmergency + 1;

 /* Increase motor emergency condition */

}

if(SEA.inputSignal<-maxPermissibleCommand){

experiment.MotorEmergency = experiment.MotorEmergency + 1;

 /* Increase motor emergency condition */

}

if(experiment.MotorEmergency>limit){

SEA.inputSignal = 0;

DACWrite();
printf("%s\n", "Program is terminated due to a possible instability!");
int status;
exit(status);
}
}

double unwrap(double encoderPosition, double encoderPositionOLD){
/* This function is an unwrap function for phase unwrapping.
 This piece of code does the same thing with the unwrap function of the MATLAB. */
double encoderPositionNEW;
double phase_PI = PI - 2.2204e-16;
double encoderPosition0 = 0;
encoderPositionNEW = encoderPosition*enc_constant + encoderPosition0;
double diff_phase = encoderPositionNEW - encoderPositionOLD;
if (diff_phase>phase_PI){
while(diff_phase>phase_PI){
encoderPosition0 = encoderPosition0 - PI2;
diff_phase = diff_phase - PI2;
}
}
if (diff_phase< -phase_PI){
while(diff_phase< -phase_PI){
encoderPosition0 = encoderPosition0 + PI2;
diff_phase = diff_phase + PI2;
}
}
encoderPositionNEW = encoderPosition*enc_constant + encoderPosition0;
return(encoderPositionNEW);
}
```

```
void readEncoders(void) {

/* This function reads encoders in every cycle */

bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_MSBFIRST);

bcm2835_spi_setDataMode(BCM2835_SPI_MODE3);

bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_512);

ENABLEMultiplexer();

ENABLEmotorangle();                              /* Select motor encoder */

union FiveByte data_read; data_read.bit64 = 0;   /* Define 5 Byte Data (40Bit) */

bcm2835_spi_transfern(data_read.bit8, 5U);       /* Read encoder */

ENABLEtorsion();                                 /* Select torsion encoder */

union FiveByte data_read2; data_read2.bit64 = 0; /* Define 5 Byte Data (40Bit) */

bcm2835_spi_transfern(data_read2.bit8, 5U);      /* Read encoder */

DISABLEMultiplexer();

bcm2835_spi_setBitOrder(BCM2835_SPI_BIT_ORDER_LSBFIRST);

bcm2835_spi_setDataMode(BCM2835_SPI_MODE1);

bcm2835_spi_setClockDivider(BCM2835_SPI_CLOCK_DIVIDER_1024);


/* Combine Data */

unsigned long value1 = data_read.bit8[4] | (data_read.bit8[3]<<8) |

 (data_read.bit8[2]<<16) | (data_read.bit8[1]<<24) | ((unsigned long long

  )data_read.bit8[0] <<32);

unsigned long value2 = data_read2.bit8[4] | (data_read2.bit8[3]<<8) |

 (data_read2.bit8[2]<<16) | (data_read2.bit8[1]<<24) | ((unsigned long long

  )data_read2.bit8[0] <<32);


SEA.encoderPositionUPD = unwrap((value1-motorPosZero), SEA.encoderPosition2_old);

// For Phase shifts

SEA.motorPos = (SEA.encoderPositionUPD/gearRate);

SEA.encoderPosition2_old = SEA.encoderPositionUPD;
SEA.torsion = (value2-torsionZero)*enc_constant-SEA.torsionInit;
SEA.motorVel = MotorPosDerivativeBlock(SEA.motorPos);
SEA.torsionVel = TorsionDerivativeBlock(SEA.torsion);
}
```

Bibliography

[1] J. K. Mills and A. A. Goldenberg, "Force and position control of manipulators during constrained motion tasks". in IEEE Transactions on Robotics and Automation, vol. 5, no. 1, pp. 30-46, Feb. 1989. doi: 10.1109/70.88015

[2] Robinson, David W., "Design and analysis of series elasticity in closed-loop actuator force control". Ph.D. Thesis Massachusetts Institute of Technology. Dept. of Mechanical Engineering.

[3] D. Whitney, "Historical perspective and state of the art in robot force control". Proceedings. 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 1985, pp. 262-268. doi: 10.1109/ROBOT.1985.1087266

[4] K. Hirai, M. Hirose, Y. Haikawa and T. Takenaka, "The development of Honda humanoid robot". Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, 1998, pp. 1321-1326 vol.2. doi: 10.1109/ROBOT.1998.677288

[5] Jacobsen, S. C., Wood, J. E., Knutti, D. F., & Biggers, K. B. , "The UTAH/M.I.T. Dextrous Hand: Work in Progress". (1984) The International Journal of Robotics Research, 3(4), 2150. https://doi.org/10.1177/027836498400300402

[6] R. O. Ambrose et al., "Robonaut: NASA's space humanoid,". in IEEE Intelligent Systems and their Applications, vol. 15, no. 4, pp. 57-63, July-Aug. 2000. doi: 10.1109/5254.867913

[7] N. Paine, J. S. Mehling J. Holley, N. A. Radford, G. Johnson C. Fok L. Sentis, "Actuator Control for the NASA-JSC Valkyrie Humanoid Robot: A Decoupled Dynamics Approach for Torque Control of Series Elastic Robots",. Journal of Field Robotics, J. Field Robotics 32 - 3, 1556-4959, https://doi.org/10.1002/rob.21556, doi:10.1002/rob.21556

[8] S. Wang et al., "Design and Control of the MINDWALKER Exoskeleton,". in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 23, no. 2, pp. 277-286, March 2015. doi: 10.1109/TNSRE.2014.2365697

[9] J. F. Veneman, R. Kruidhof, E. E. G. Hekman, R. Ekkelenkamp, E. H. F. Van Asseldonk and H. van der Kooij, "Design and Evaluation of the LOPES Exoskeleton Robot for Interactive Gait Rehabilitation,". in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 15, no. 3, pp. 379-386, Sept. 2007. doi: 10.1109/TNSRE.2007.903919

[10] T. G. Sugar et al., "Design and Control of RUPERT: A Device for Robotic Upper Extremity Repetitive Therapy,". in IEEE Transactions on Neural Systems

and Rehabilitation Engineering, vol. 15, no. 3, pp. 336-346, Sept. 2007. doi: 10.1109/TNSRE.2007.903903

[11] T. Noda, T. Teramae, B. Ugurlu and J. Morimoto, ”Development of an upper limb exoskeleton powered via pneumatic electric hybrid actuators with bowden cable,”. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014, pp. 3573-3578. doi: 10.1109/IROS.2014.6943062

[12] Caldwell, Darwin Gordon, ”Compliant polymeric actuators as robot drive units”. Ph.D. Thesis Department of Electronic Engineering, The University of Hull September 1989

[13] C. Semini, N. G. Tsagarakis, E. Guglielmino and D. G. Caldwell, ”Design and experimental evaluation of the hydraulically actuated prototype leg of the HyQ robot,”. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, 2010, pp. 3640-3645. doi: 10.1109/IROS.2010.5651548

[14] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi and M. Raibert, ”Autonomous navigation for BigDog,”. 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, 2010, pp. 4736-4741. doi: 10.1109/ROBOT.2010.5509226

[15] G. A. Pratt, M. M. Williamson, P. Dillworth, J. Pratt, K. Ulland, and A. Wright, Stiffness isnt everything,. in Proc. 4th Int. Symp. Experimental Robotics, Stanford, CA, 1995.

[16] M. C. Yildirim, P. Sendur, O. Bilgin, B. Gulek, G. G. Yapici and B. Ugurlu, ”An integrated design approach for a series elastic actuator: Stiffness formulation, fatigue analysis, thermal management,”. 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), Birmingham, 2017, pp. 384-389. doi: 10.1109/HUMANOIDS.2017.8246902

[17] N. Kashiri, J. Malzahn, N. Tsagarakis , ”On the Sensor Design of Torque Controlled Actuators: A Comparison Study of Strain Gauge and Encoder Based Principles,”. IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 1186-1194, 2017.

[18] Paul McKenney, ”A realtime preemption overview,”. https://lwn.net/Articles/146861/

[19] Online Source ”https://www.linuxfoundation.org/”.

[20] E. Sariyildiz, G. Chen and H. Yu, ”An Active Disturbance Rejection controller design for the robust position control of Series Elastic Actuators,”. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, 2016, pp. 266-272. doi: 10.1109/IROS.2016.7759065

[21] G. A. Pratt, P. Willisson, C. Bolton and A. Hofman, "Late motor processing in low-impedance robots: impedance control of series-elastic actuators,". Proceedings of the 2004 American Control Conference, Boston, MA, USA, 2004, pp. 3245-3251 vol.4. doi: 10.23919/ACC.2004.1384410

[22] G. Wyeth, Control issues for velocity sourced series elastic actuators,. presented at the Australian Conf. Robot. Autom., Auckland, New Zealand, Dec. 2006.

[23] H. Vallery, R. Ekkelenkamp, H. van der Kooij and M. Buss, "Passive and accurate torque control of series elastic actuators,". 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, 2007, pp. 3534-3538. doi: 10.1109/IROS.2007.4399172

[24] P. Tomei, "A simple PD controller for robots with elastic joints,". in IEEE Transactions on Automatic Control, vol. 36, no. 10, pp. 1208-1213, Oct. 1991. doi: 10.1109/9.90238

[25] S. Oh and K. Kong, "High-Precision Robust Force Control of a Series Elastic Actuator,". in IEEE/ASME Transactions on Mechatronics, vol. 22, no. 1, pp. 71-80, Feb. 2017. doi: 10.1109/TMECH.2016.2614503

[26] J. Bae, K. Kong and M. Tomizuka, "Gait phase-based smoothed sliding mode control for a rotary series elastic actuator installed on the knee joint,". Proceedings of the 2010 American Control Conference, Baltimore, MD, 2010, pp. 6030-6035. doi: 10.1109/ACC.2010.5531282

[27] E. Sariyildiz, Haoyong Yu, T. Nozaki and T. Murakami, "Robust force control of Series Elastic Actuators using Sliding Mode Control and Disturbance Observer,". IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, 2016, pp. 619-624. doi: 10.1109/IECON.2016.7792991

[28] Calanca, Andrea & Capisani, Luca & Fiorini, Paolo. "Robust Force Control of Series Elastic Actuators,". (2014) Actuators. 3. 182-204. 10.3390/act3030182.

[29] J. S. Mehling, J. Holley and M. K. O'Malley, "Leveraging disturbance observer based torque control for improved impedance rendering with series elastic actuators,". 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 1646-1651. doi: 10.1109/IROS.2015.7353588

[30] Fliess, Michel & Lvine, Jean & Martin, Philippe & Rouchon, Pierre. "Differential flatness and defect: An overview,". (1995). Banach Center Publications. 32. 10.4064/-32-1-209-225.

[31] A. T. Kansizoglu, E. Sariyildiz and B. Ugurlu, "A comparison study on observer-based force control of series elastic actuators,". 2018 IEEE 15th International Workshop on Advanced Motion Control (AMC), Tokyo, 2018, pp. 411-416. doi: 10.1109/AMC.2019.8371128

[32] J. Yang, S. Li, X. Chen, Qi Li, ”Disturbance rejection of dead-time processes using disturbance observer and model predictive control”, in Chemical Engineering Research and Design, vol. 89, no. 2, 2011, pp. 125-135

[33] M. Morari, J. H. Lee, Model predictive control: past, present and future, In Computers & Chemical Engineering, Volume 23, Issues 4-5, 1999, Pages 667-682

[34] Kiam Heong Ang, G. Chong and Yun Li, ”PID control system analysis, design, and technology,”. in IEEE Transactions on Control Systems Technology, vol. 13, no. 4, pp. 559-576, July 2005. doi: 10.1109/TCST.2005.847331

[35] Yildirim M.C., Kansizoglu A.T., Sendur P., Ugurlu B. , ”High Power Series Elastic Actuator Development for Torque-Controlled Exoskeletons,”. in: Wearable Robotics: Challenges and Trends. WeRob 2018. Biosystems & Biorobotics, vol 22. Springer, Cham

[36] K. Ohnishi, M. Shibata, and T. Murakami, ”Motion control for advanced mechatronics,. in IEEE Trans on. Mechatronics, vol. 1, no. 1, pp. 56-67, Mar. 1996.

VITA

Ahmet Talha Kansizoglu was born in Izmir, Turkey. He recieved his Bachelor's degree in Control and Automation Engineering in July 2016 at the Yildiz Technical University, Istanbul, Turkey. He is a M.Sc. Student in Ozyegin University under the supervision of Assist. Prof. Barkan Ugurlu. His research interests include control systems, robust control of series elastic actuators, and robotics.

Publications

Conference Paper: A. T. Kansizoglu, E. Sariyildiz, B. Ugurlu, ”A Comparison Study on Observer-based Force Control of Series Elastic Actuators” 2018 15th IEEE International Workshop on Advanced Motion Control, Tokyo, 2018

Conference Paper: M.C. Yildirim, A.T. Kansizoglu, P. Sendur, B. Ugurlu ”High Power Series Elastic Actuator Development for Torque-Controlled Exoskeletons” 2018 WeRob International Symposium on Wearable Robotics, Pisa 2018