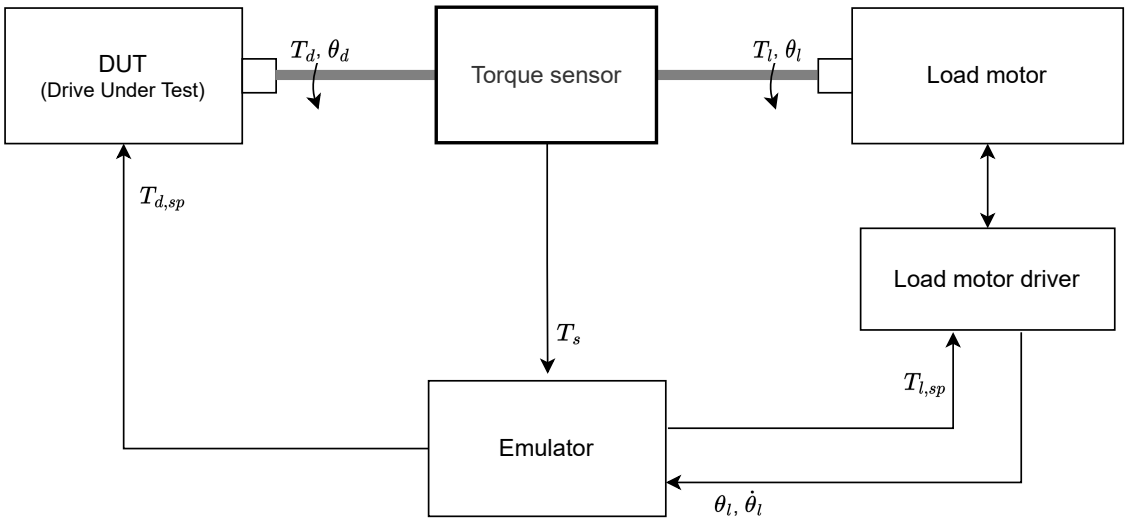
CHALMERS
UNIVERSITY OF TECHNOLOGY



Dynamic Emulation of Mechanical Loads in Electric Drives

Closed-loop emulation using a feedforward-tracking principle with compensator

Master's thesis in System, Controls & Mechatronics

Mikael Bengtsson

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Dynamic Emulation of Mechanical Loads in Electric Drives

Closed-loop emulation using a feedforward-tracking principle with compensator

Mikeal Bengtsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2024

Dynamic Emulation of Mechanical Loads in Electric Drives
Closed-loop emulation using a feedforward-tracking principle with compensator
Mikael Bengtsson

Supervisor: Jimmy Björkman, Aros Electronics
Examiner: Jonas Fredriksson, E2

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Emulator system block-diagram.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Dynamic Emulation of Mechanical Loads in Electric Drives
Closed-loop emulation using a feedforward-tracking principle with compensator
Mikael Bengtsson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis presents a method of emulating a mechanical load in the context of electric drive systems. It investigates the implementation of a closed-loop emulator, based on a feedforward-tracking principle, on an existing dynamometer system. Furthermore, it analyses the possibilities of using a torque sensor to enhance the emulation performance and accuracy. Physical modeling and system identification methods have been used to obtain a linear model of the dynamometer system. This model serves as the basis for a compensator in the emulator and is also used for system simulations. The results show that the proposed emulator has the ability to emulate linear loads with varying inertia and friction, both in simulations and on the physical system. The resulting emulated velocity exhibits some oscillations when emulating moderate to high inertia loads. The cause of this is believed to originate from delays in the system.

Keywords: dynamic emulation, system identification, LTI system, discretization, feedforward controller

Acknowledgements

I would like to start by thanking Aros Electronics, and notably Magnus Wide for taking interest in my application and providing me the opportunity to perform this project. Also, I'm equally grateful for the supervision from Jimmy Björkman at Aros Electronics who guided me through many of the technical aspects in the project.

Lastly, I would like to thank my supervisor and examiner at Chalmers, Prof. Jonas Fredriksson, for his insights and valuable advice throughout the project.

Mikael Bengtsson, Gothenburg, January 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

DUT	Drive Under Test
LM	Load Motor
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
ZOH	Zero-order hold
PID	Proportional-Integral-Derivative
PI	Proportional-Integral

Contents

List of Figures

List of Tables

1

Introduction

When designing electric machines and drives, a key element is to make sure that the drive system is optimised for the intended use. Each application has different requirements for the forces and torques that act on the motor. Therefore, it is crucial to verify and test the motor drive unit with the mechanical load acting on the system. This can, however, be hard to do in practice. The mechanical load can be hard to access physically due to size or inaccessibility, or it could require a time-consuming setup procedure. A more efficient way to test the motor drive unit would be to replicate the application load in a test bench environment.

Dynamic emulation, in the context of electric drive systems, refers to a concept based on a system that can replicate the mechanical load dynamics. This can partially be done in a software simulation environment or with a physical actuator acting on the motor shaft. In the latter case, a convenient way to achieve this is to use a secondary electric motor that can act as the mechanical load. Such a system can be referred to as a *dynamometer*.

1.1 Aim

Aros Electronics develops electric machines and motor control drivers for a wide range of applications. The end application is often industry-based, which could be hard or impossible to access. The ability to run in-house tests that resemble the real-world application during the early development phase would, therefore, be of great benefit. In the current test bench setup at Aros, the dynamic behaviour of the load motor dynamometer is currently restricted to static load tests in open-loop conditions. In order to fully capture the dynamics of the end application, a closed-loop emulation method is required. This could include emulation of non-linear dynamics such as stiction friction, varying inertia, etc. In some cases, smaller motors are to be tested, and the inherent inertia and friction of the dynamometer itself are greater than the intended load. These effects could be removed with the proposed method of emulating a load.

This thesis aims to develop a closed-loop dynamic emulation algorithm for the existing dynamometer system at Aros Electronics. More specifically, the primary goal is to analyse and implement a feedforward-tracking principle in the existing dynamometer system as presented in [5]. The motivation for choosing this method is that it serves as a foundation for a number of the other more recent methods,

[11][2], that aims to improve certain aspects of the original method. The suggested method also preserves the zero-pole structure of the desired mechanical load, which is a crucial factor when the emulated load is part of a closed-loop speed controller.

An important aspect from [5] is the reference torque input to the emulation algorithm. The method uses the internal torque reference obtained from the DUT (*drive under test*) as input to the emulation algorithm. This thesis will investigate the possibility of using measured torque from a torque sensor as input to the emulator, and present the outcome of doing so compared to using DUT torque reference. Using the existing torque-transducer in the test bench eliminates the need to communicate with the DUT during emulation, resulting in a more generic test setup. Another potential benefit is that the measured torque from the torque sensor is more accurate than the reference torque from the DUT since this is estimated internally and could be inaccurate.

1.2 Objective/Method

To achieve the desired aim, the proposed emulation method will first be implemented in a simulation environment using Matlab/Simulink. Measurements of the physical dynamometer system will be performed to identify possible constraints and limitations. These measurements will also be used to develop a realistic model of the dynamometer dynamics regarding friction and inertia, which will be used both in the emulator algorithm and as a simulation target. An analysis of the possible methods to include the torque sensor in the emulation will be performed. This analysis will be done on a system level and investigate possible effects on the emulator dynamics. Once the emulation method has been confirmed in simulation, the algorithm is implemented on the physical dynamometer system using Python. The implementation is then evaluated and compared against the simulated results in terms of emulation accuracy and the stability of the overall system.

1.3 Scope/Limitations

Since the thesis will cover several aspects, from theoretical analysis to physical implementation, some limitations on what the thesis will cover are necessary. Below is a list of relevant limitations:

- The report will not go into detail about the underlying control of the load motor itself. The work will rely on the existing control unit (Baumuller bmaxx 4400) and its built-in control algorithms. It is assumed that this drive is a highly dynamic vector-controlled drive that can be controlled accurately. Relevant constraints on the load motor and drive will, however, be analysed and presented.
- The report will not present any specifics regarding the emulator implementation (in Python). Although a significant amount of time has been devoted to this subject, it is regarded as being beyond the scope of the problem statement.
- The work will not go into detail regarding the specifications of the DUT itself. Since the dynamometer system is supposed to handle various drives to be tested, any analysis regarding the DUT will be limited to a generic case.

1.4 Outline

The report will initially cover the theoretical aspects of implementing the proposed emulation method. The analysis will also include a review of the literature and other possible emulation methods.

The methodology in chapter 3 presents the approaches used to develop the emulator system, both in simulations and in the physical system. Initially, a system model of the dynamometer plant is presented, where a system identification approach is performed to determine the model parameters. The section continues with an analysis of the torque sensor dynamics and load motor drive delays. The design of the emulator and its subsystems, based on numerical values of the different input parameters, are also covered.

The chapter ends with an investigation of how the torque sensor can be used together with the emulator. More specifically, how the DUT output torque can be estimated and used as an input to the emulator and how this affects the system's dynamics. This is done through a frequency analysis of the cases presented.

Finally, chapter 4 presents the simulation and experimental results of the models and methods derived in chapter 3. It contains open-loop simulation results for the dynamometer model together with results for the estimation of DUT output torque. The chapter then presents results for the complete emulator system, divided into the different cases of input torque that is analysed in the previous chapter. Each case consists of simulation and experimental results obtained after the emulator was implemented on the physical system.

2

Theory

This chapter provides a comprehensive overview of the foundational concepts and theoretical frameworks that underpin this study. It presents the theory behind dynamic emulation and a method for implementation on a digital system.

There are several research articles on the topic of dynamic emulation. A study of the literature shows that one of the early strategies implemented a simple inverse mechanical dynamic model as an emulator [3]. The method is straightforward but has disadvantages regarding the physical implementation in a digital system. Due to the use of an inverse model and its derivative terms, it is sensitive to measurement noise [9]. It is also restricted in that it implements the emulated load under open-loop conditions and accurate emulation of the load can, therefore, not be guaranteed.

The authors in [5] bring up the limitations of this method and present an alternative method based on a feedforward-tracking principle. This method has become state-of-the-art for dynamic load emulation [8] since it produces good results for both linear and non-linear loads and is well used in industrial settings. It also preserves the zero-pole structure of the emulated load. This is particularly important for the application in this thesis, considering that most test cases for the dynamometer will run a closed-loop speed controller on the DUT. As the main purpose of the emulator will be to test and evaluate the controller itself, it is vital that the emulated load keeps the exact dynamics as the real load in the end application.

2.1 Emulation fundamentals

This section presents the fundamental concepts behind dynamic emulation. Consider figure 2.1, where a *drive under test* (DUT) is driving a mechanical load described by the transfer function G_{load}.

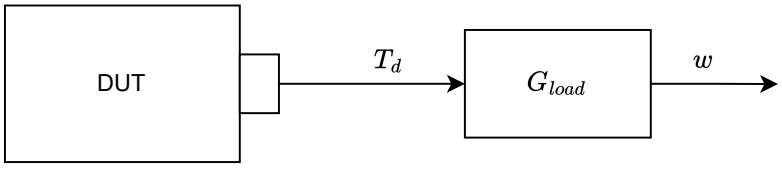


Figure 2.1: DUT connected to a mechanical load

If the mechanical load consists of a simple inertia and viscous friction element, the transfer function of the load can be written as:

$$\frac{\omega(s)}{T_d(s)} = G_{load}(s) = \frac{1}{Js + B} \tag{2.1}$$

where w is the output shaft speed and T_d is the torque produced by the DUT machine. To emulate the mechanical load a dynamometer system that consists of a secondary electric motor, *Load Motor* (LM) is used, see figure 2.2.

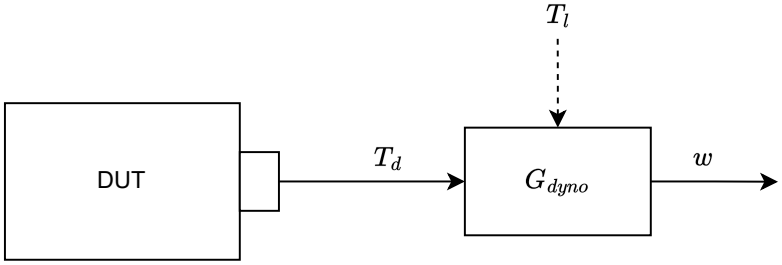


Figure 2.2: DUT connected to the dynamometer

It follows that:

$$\frac{\omega(s)}{T_d(s)} = G_{dyno}(s) \tag{2.2}$$

where $G_{dyno}(s)$ is the dynamometer system's transfer function, consisting of the load motor with connected shafts and bearings. The dynamics of $G_{dyno}(s)$ will generally be very different from the mechanical load dynamics of $G_{load}(s)$. The objective of an emulation algorithm is to control the LM torque T_l so that the dynamics seen from the DUT side equals the dynamics of the mechanical load.

2.2 Emulation with inverse load dynamics

An intuitive solution to achieve the goal of emulating the load is to use an inverse load dynamic model, as explained in [3] and [12], amongst others. In these methods, the shaft speed is measured, and an inverse load dynamic model is used to derive the desired load torque T_l. An example of such a model can take the form:

$$T_l = T_0 + a\omega + b\omega^2 + c\omega^3 + J_{em}\frac{d\omega}{dt} \tag{2.3}$$

where T_0, a, b and c are constants, J_{em} is the emulated inertia. A benefit of this model is that it can take any form the designer wishes and include non-linear friction elements. However, it does have the disadvantage of taking the time derivative of the shaft speed w, which can lead to increased noise levels if the measured shaft speed contains noise. In some cases [12], the shaft torque is also measured, and a shaft torque controller is used to derive the desired LM torque T_l.

For emulating a linear load with inertia J_{em} and viscous friction B_{em}, the principle of inverse load dynamics is well illustrated in [5] and can be seen in figure 2.3:

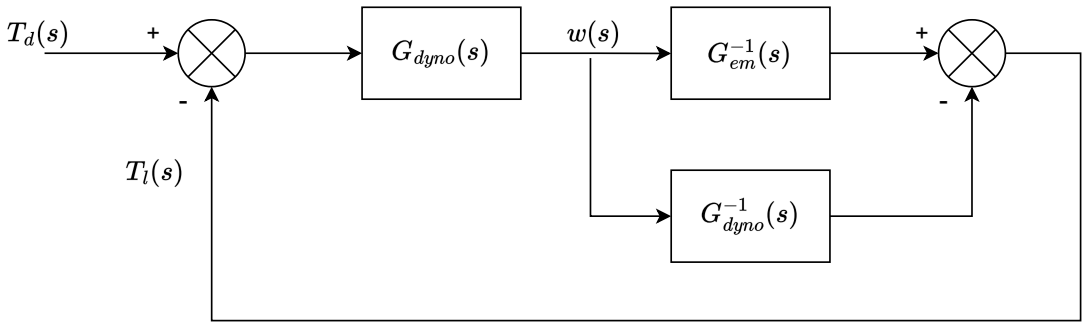


Figure 2.3: Emulation with inverse load dynamics

As the figure shows, the speed w is measured and given as an input to the inverse dynamics:

$$G_{em}^{-1}(s) = J_{em}s + B_{em} \tag{2.4}$$

The dynamics of the dynamometer itself is then compensated for and a resulting load torque T_l is acquired. By analyzing the system with T_d as input and w as output in figure 2.3, it can be shown that:

$$\frac{\omega(s)}{T_d(s)} = G_{em}(s) = \frac{1}{J_{em}s + B_{em}} \tag{2.5}$$

which shows that this system does achieve its purpose of emulating the load, $G_{load}(s) = G_{em}(s)$. As the authors in [5] also mention, this method has some problems when being implemented on a physical system due to sampling and discretization effects from utilizing the inverse dynamics. Furthermore, they also point

out that the overall zero-pole structure of the emulated load is not equal to that of the real mechanical load.

2.3 Emulation with feedforward-tracking controller

The following section describes the method used in [5] and [2], amongst others. The method can emulate both linear and non-linear loads without the need for an inverse model of the load dynamics. The concept aims to replicate the actual load's zero-pole structure by using a tracking controller for shaft speed and an analytical compensator. Figure 2.4 shows the concept of the speed tracking loop.

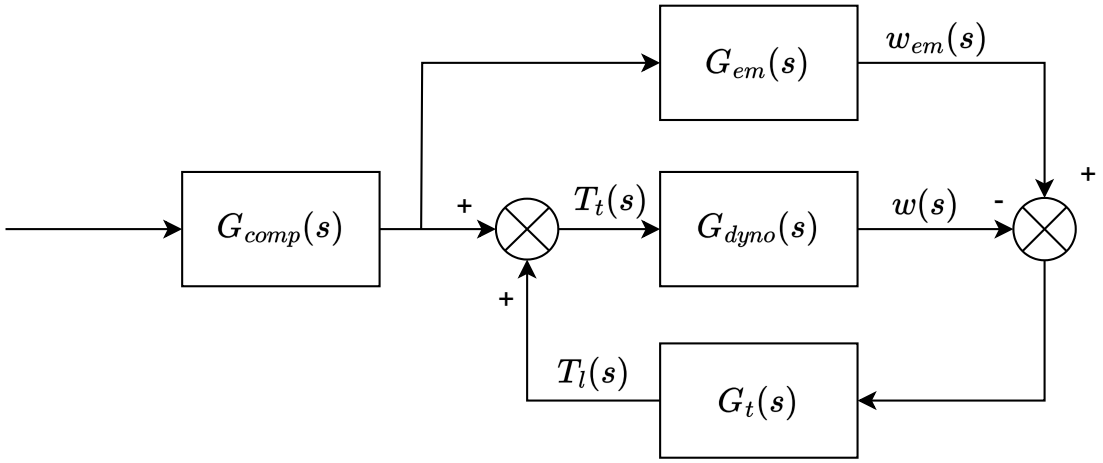


Figure 2.4: Emulation using a speed tracking controller

It can be shown that the transfer function from drive torque $T_d(s)$ to motor speed $\omega(s)$ is:

$$\frac{\omega(s)}{T_d(s)} = G_{comp}(s)G_{em}(s)\frac{(1/G_{em}(s)) + G_t(s)}{(1/G_{dyno}(s)) + G_t(s)} \tag{2.6}$$

If the aim is to isolate $G_{em}(s)$ in the right hand side of (2.6), then $G_{comp}(s)$ can be designed as the following:

$$G_{comp}(s) = \frac{(1/G_{dyno}(s)) + G_t(s)}{(1/G_{em}(s)) + G_t(s)} \tag{2.7}$$

and the resulting input-to-output relationship becomes:

$$\frac{\omega(s)}{T_d(s)} = G_{comp}(s)G_{em}(s)G_{comp}^{-1}(s) = G_{em}(s). \tag{2.8}$$

A nice property of this compensator is that the zero-pole structure of the load is preserved. As the authors in [5] highlight, $G_{comp}(s)$ contains the emulated load dynamics. This makes it impossible to emulate non-linear loads since $G_{em}(s)$ is

restricted to being a linear system due to the fact that $G_{comp}(s)$ is an LTI system in its nature. Their solution to this problem is to add the inverse dynamics $G_{em}^{-1}(s)$ as a feed-forward from the emulated speed w_{em}, as shown in figure 2.5. The compensator $G_{comp}(s)$ can now be derived as:

$$\frac{\omega(s)}{T_d(s)} = G_{em}(s)\frac{G_{dyno}(s)G_t(s)}{1 + G_{dyno}(s)G_t(s)} = G_{em}(s)G_{comp}^{-1}(s) \tag{2.9}$$

where

$$G_{comp}(s) = \frac{1 + G_{dyno}(s)G_t(s)}{G_{dyno}(s)G_t(s)} \tag{2.10}$$

As in (2.8), $G_{comp}(s)$ is added in series and the emulation is fulfilled. Notice that the speed controller $G_t(s)$ now outputs the shaft net torque T_t, rather than the load motor torque T_l as in figure 2.4.

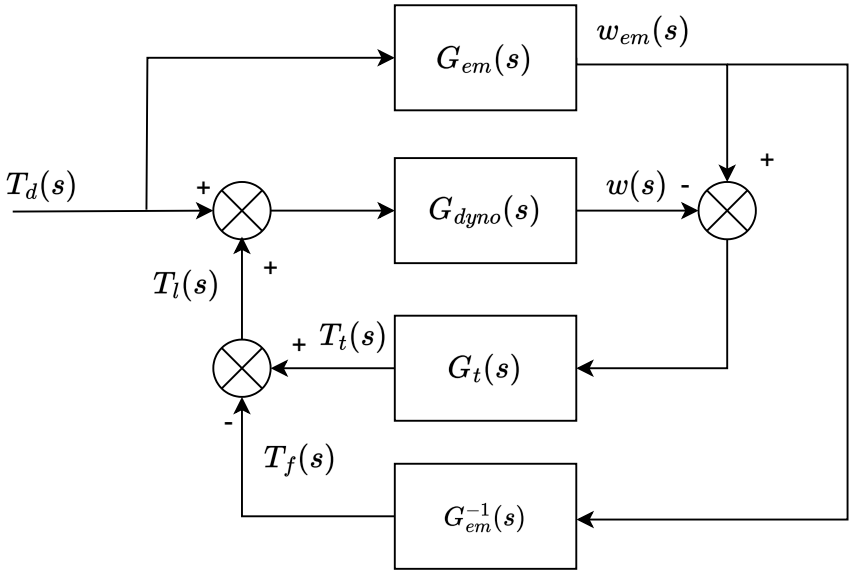


Figure 2.5: Emulation using a speed tracking controller with the addition of a feedforward path with the inverse load dynamics $G_{em}^{-1}(s)$

The system in figure 2.5 can be simplified by the fact that the feedforward torque $T_f = T_d$ due to the unity path of $G_{em}(s)G_{em}^{-1}(s) = 1$. By letting the output torque of $G_t(s)$ act directly on $G_{dyno}(s)$, the summation blocks can be removed together with the inverse load dynamics $G_{em}^{-1}(s)$, since it is not explicitly required. Finally, adding $G_{comp}(s)$ in series, the resulting emulator system can be seen as that of figure 2.6.

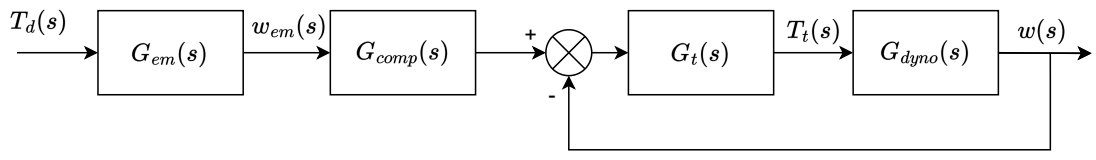


Figure 2.6: Simplified and compensated emulation system

2.4 Discretization

The compensator needs to be discretized to implement the emulator in a computer. The discrete representation of the system, with a sampling time of t_s, can be seen in figure 2.7. Recalling that the output of $G_t(z)$ is the shaft net torque $T_t = T_d + T_l$, the reference torque is then subtracted from the output of $G_t(z)$ to produce the desired load motor torque T_l. When implementing the emulation algorithm on the physical device, this torque is fed to the load motor. If the simple plant model in (2.2) is used in a simulation environment, the reference torque T_d is then added again to produce T_t which is the input to $G_{dyno}(s)$.

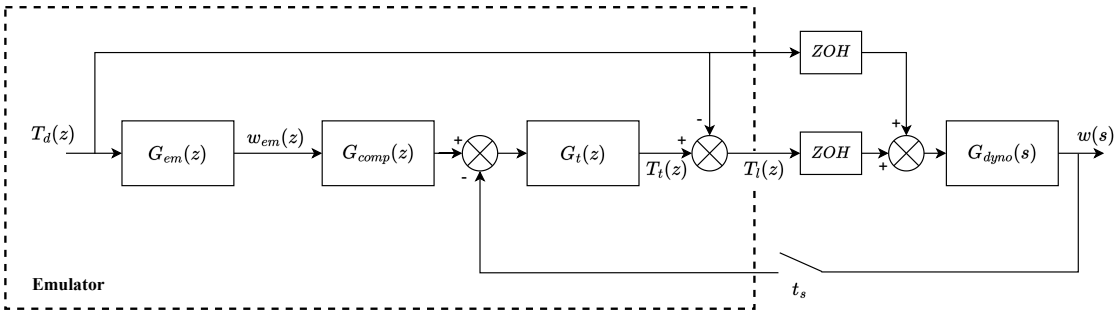


Figure 2.7: Discrete representation of the emulation system

As the authors in [5] highlight, $G_{comp}(z)$ system is improper. This means that the order of the numerator is higher than the denominator, and the system cannot be implemented as a regular LTI system in simulation or in a physical implementation. To solve this, a single delay $1/z$ can be added to $G_{comp}(z)$ to make the transfer function proper:

$$G_{comp}(z) = \frac{1 + G_{dyno}(z)G_t(z)}{G_{dyno}(z)G_t(z)} \times \frac{1}{z} \tag{2.11}$$

The addition of a single delay z^{-1} is enough to make $G_{comp}(z)$ proper as long as $G_{dyno}(s)$ is first order and $G_t(z)$ is a discrete PI controller. This delay is then compensated by multiplying the discretized load model $G_{em}(z)$ with z:

$$G_{em}(z) = z \cdot G_{em}(z) = z \cdot \mathcal{Z}\{G_h(s)G_{em}(s)\} \tag{2.12}$$

where $G_h(s)$ is the ZOH transfer function. As the authors in [5] point out, (2.12) corresponds to the discretization of $G_{em}(s)$ using the zero-pole matching method with a zero mapped to $z = 0$. This will cancel out the introduced delay of (2.11).

2.4.1 Zero-order hold method

To obtain an analytical expression for $G_{em}(z)$, with constants J_{em} and B_{em}, the zero-order hold representation of a first-order system $G(s)$ is presented here. Consider the system $G(s)$:

$$G(s) = \frac{1}{as+b} = \frac{1/a}{s+b/a} \tag{2.13}$$

where a and b are constants. The zero-order hold equivalent of this system is [7]:

$$G(z) = (1-z^{-1})\mathcal{Z}\left\{\frac{G(s)}{s}\right\} \tag{2.14}$$

By looking up the z-transform of $G(s)/s$ in a table of common transforms [1], the following expression for $G(z)$ can be obtained:

$$G(z) = (1-z^{-1})\frac{(\frac{1}{a})z(1-e^{-(b/a)t_s})}{(\frac{b}{a})(z-1)(z-e^{-(b/a)t_s})} \tag{2.15}$$

After some rearranging and simplification,(2.15) can be rewritten as:

$$G(z) = \frac{\frac{1}{b}(1-e^{-(b/a)t_s})}{z-e^{-(b/a)t_s}} \tag{2.16}$$

The expression (2.16) can be used for the discretization of both $G_{em}(s)$ and $G_{dyno}(s)$, where the constants a and b can be replaced with the corresponding parameters in each system.

3

Method

This chapter presents the methods used to reach the desired outcome of the project. The chapter contains a system modelling section and presents a method for designing the emulator system.

3.1 System Description

The dynamometer system consists of a three-phase PMSM load motor, including a drive system. The DUT is connected to the load motor via mechanical couplings on a rigid shaft with a torque sensor mounted between the two motors. Figure 3.1 shows the physical dynamometer system.

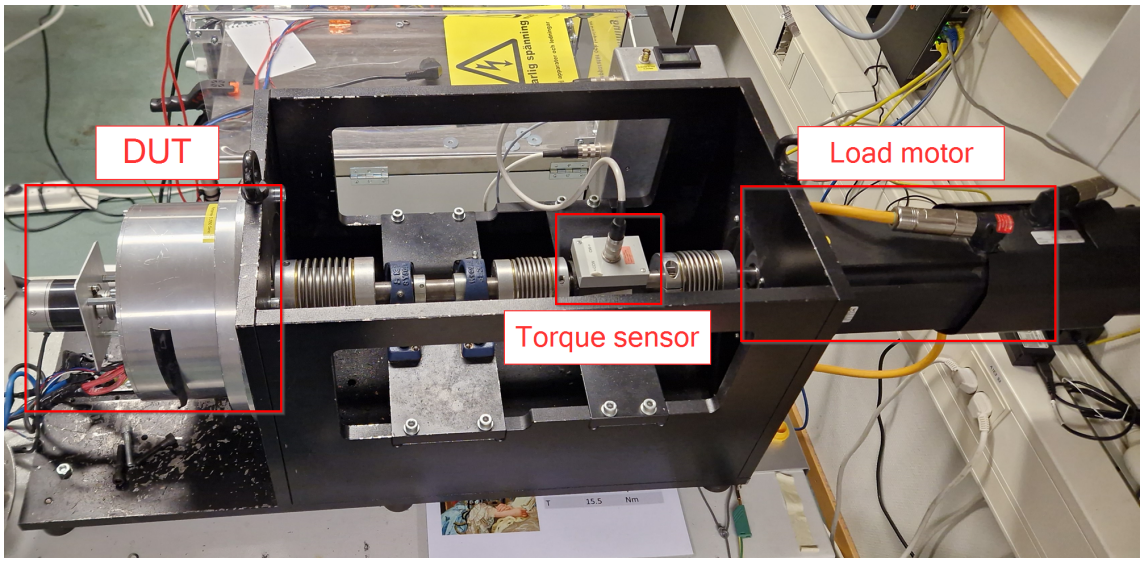


Figure 3.1: Physical Dynamometer system

Figure 3.2 presents a block diagram of the system, including communication links and signals. Here, T_d and T_l are the torque output from the DUT and LM, respectively. The emulator, implemented in Python, communicates with the LM, DUT and torque sensor over a CAN bus. $T_{d,sp}$ and $T_{l,sp}$ are the torque set-points for the DUT and load motor respectively. The output from the torque sensor is an analog voltage that is converted into CAN frames by an ADC to CAN module, resulting in the measured torque denoted as T_s. The position and velocity of the shaft, θ_l and $\dot{\theta}_l$, are sampled from the load motor driver.

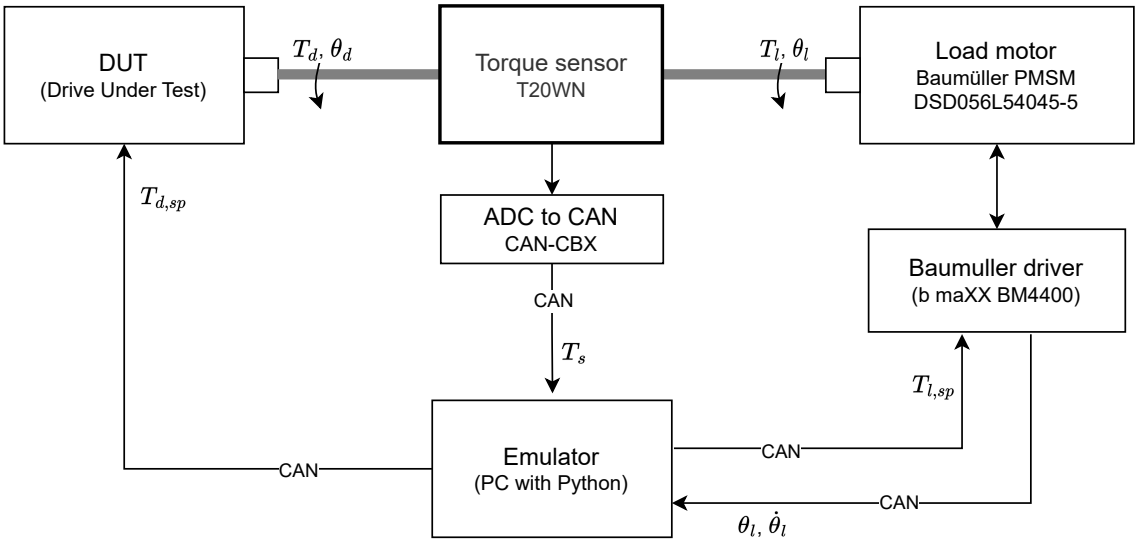


Figure 3.2: Block diagram of dynamometer system with communication links

Note that the set-point torque to the DUT $T_{d,sp}$ is not part of the emulation algorithm itself. It is included to run experimental tests on the emulator system. Naturally, set-points for DUT speed or position could also be used for experimental tests.

3.1.1 Specifications and limitations

This section presents relevant specifications for the physical system, which sets some of the limitations of the emulator itself.

Table 3.1 presents the most relevant specifications for the load motor. With these specifications in mind, the output torque from the emulator is naturally restricted to the nominal torque of the load motor. Also, the rotational speed that the emulator can operate at is restricted to that of the load motor. The load motor is configured to run in torque control mode during emulation. The torque controller is essentially a field-oriented current controller that accepts torque set-point values.

Load motor specifications

Nominal power	5400 W
Nominal current	10.5 A
Nominal Torque	11.4 Nm
Nominal Speed	4500 rpm

Table 3.1: Specifications for load motor

The emulator sample loop frequency f_s, which is also the rate of the internal control loop of the emulator, is set to run at 470 Hz. Initial tests on the system reveal that this is essentially the fastest loop frequency that can be obtained with the current setup. This moderate loop frequency is believed to be caused by a large overhead in the Python CANopen library. Table 3.2 shows some of the specifications related to the implementation of the emulator.

Emulator specifications

Sample frequency, f_s	470 Hz
Sample time, t_s	0.0021 s
CAN baud rate	500000 kbit/s

Table 3.2: Specifications for emulator implemented on digital system

3.2 Dynamometer model

A model of the dynamometer system is presented here for simulation and analysis. A linear MIMO state-space model of the system, including the torque sensor, is developed. The inclusion of the torque sensor in the model is to be able to simulate the case where the measured torque is used as an input to the emulation algorithm.

This section does not present the plant model to be used in the emulation algorithm, $G_{dyno}(s)$ as in figure 2.6. Since the emulation algorithm expects this model to be a first-order transfer function with a single input and output, the system model presented here is later simplified into such. This is presented in more detail in section 3.3.1.

Figure 3.3 represents the dynamometer system. It consists of the DUT, the load motor, and a torque sensor. The two discs correspond to each motor's inertia and frictional effects, including shafts and bearings on each side of the torque sensor.

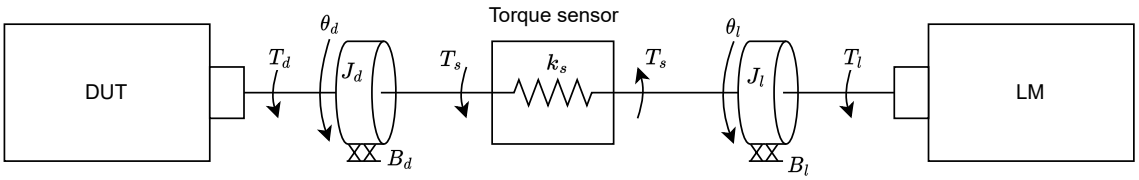


Figure 3.3: Dynamometer system model

Using a free body diagram on both of the discs in figure 3.3 results in the following equations of motion:

$$J_d\ddot{\theta}_d + B_d\dot{\theta}_d - T_s - T_d = 0 \tag{3.1}$$

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l + T_s - T_l = 0 \tag{3.2}$$

where T_d and T_l are the torque generated by the DUT and load motor, respectively. T_s represents the torque that the torque sensor exhibits and is modeled as a flexible shaft (spring). The measured torque T_s can be expressed as the displacement between the angles θ_l and θ_d times a torsion constant k_s:

$$T_s = k_s(\theta_l - \theta_d) \tag{3.3}$$

The flexible shaft is, in reality, just a rigid shaft with minimal displacement, and it will not affect the system's dynamics. Initial simulations using an analytically obtained value of the torsion constant results in unstable simulations in Simulink. Due to the nature of the high rigidity of the shaft, caused by the large value of k_s, the system exhibits high numerical stiffness, leading to numerical instability. A solution to this is to include a damping term in the torque sensor model, with a coefficient of k_c:

$$T_s = k_s(\theta_l - \theta_d) + k_c(\dot{\theta}_l - \dot{\theta}_d) \tag{3.4}$$

Since we are not interested in high-frequency dynamics, the damping term can be introduced without affecting the model dynamics in the lower frequency range. A more thorough analysis of this can be found in section 3.2.1.1.

The system equations in (3.1) and (3.2), together with the expression of T_s in (3.4), are converted to state-space representation:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{3.5}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{3.6}$$

with the resulting state-space variables:

$$\boldsymbol{x} = \begin{bmatrix} x_1 = \theta_d \\ x_2 = \theta_l \\ x_3 = \dot{\theta}_d \\ x_4 = \dot{\theta}_l \end{bmatrix} \tag{3.7}$$

and input vector:

$$\boldsymbol{u} = \begin{bmatrix} T_d \\ T_l \end{bmatrix} \tag{3.8}$$

From (3.1) and (3.2), the state derivatives can be found as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x_1} = x_3 \\ \dot{x_2} = x_4 \\ \dot{x_3} = \frac{1}{J_d}(T_d + x_2 k_s - x_1 k_s + x_3(-B_d - k_c) + x_4 k_c \\ \dot{x_4} = \frac{1}{J_l}(T_l - x_2 k_s + x_1 k_s + x_3 k_c + x_4(-B_l - k_c)) \end{bmatrix} \tag{3.9}$$

As the model is linear, the complete state-space representation of the system can be written in matrix form as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_s/J_d & k_s/J_d & (-k_c - B_d)/J_d & k_c/J_d \\ k_s/J_l & -k_s/J_l & k_c/J_l & (-k_c - B_l)/J_l \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/J_d & 0 \\ 0 & 1/J_l \end{bmatrix} \boldsymbol{u} \tag{3.10}$$

and as all states and the torque T_s are measured, the output matrix becomes:

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} = \begin{bmatrix} \theta_d \\ \theta_l \\ \dot{\theta}_d \\ \dot{\theta}_l \\ T_s \end{bmatrix} \Rightarrow \boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_s & k_s & -k_c & k_c \end{bmatrix}, \boldsymbol{D} = [0]_{5x2} \tag{3.11}$$

3.2.1 Model parameters

This section describes the estimation of the parameters in the model developed in the previous section. As can be seen from (3.10), the model requires values for the inertia and friction coefficients $J_{d,l}$, $B_{d,l}$ as well as a value for the torsion- and damping constant k_s and k_c.

3.2.1.1 Torsion and damping constants

The value for the torsion constant k_s is found by an analytical estimation. Since the torsion constant is that of a rigid shaft, it will be large, and in turn, the displacement between θ_l and θ_d will be very small. The effect of this is that the torque sensor, modeled as a flexible shaft, will not affect the overall dynamics of the system. Therefore, as long as k_s is large enough, the exact value of this constant is less important. With that in mind, k_s can be estimated with the torsion formula for a round shaft [4]:

$$k_s = \frac{GJ}{L} \tag{3.12}$$

where G is the shear modulus of the material, J is the inertia and L is the length of the shaft. With a given diameter of the shaft, the inertia J can be calculated as:

$$J = \frac{\pi d^4}{32} \tag{3.13}$$

The length and diameter of the torque sensor shaft are measured to $L = 10$ cm and $d = 25$ mm. The value of G is set to the shear modulus of aluminum, which is 28×10^9 Pa. This results in the following shaft torsion constant:

$$k_s = \frac{28 \times 10^9 \, (\pi 0.025^4)/32}{0.1} \approx 1.074 \times 10^4 \, \mathrm{N\,m\,rad^{-1}} \tag{3.14}$$

As mentioned earlier, a damping term in (3.4) is needed for simulation stability. The value is obtained by trial and error. A value close to that of k_s seems to yield stable simulations while also preserving the dynamics of the system model. In section 3.3.1 it is shown that the system model, with the inclusion of the torsion and damping constants, indeed has same dynamics as a simple first order model with only inertia and friction.

3.2.1.2 Inertia and friction estimation

The coefficients for inertia and friction are not explicitly available. Therefore, an identification method based on measurement data is used to estimate these. The method consists of applying input steps in velocity for one of the motors while measuring shaft torque and velocity. Acceleration is calculated from filtered velocity data and the coefficients for inertia and friction are estimated by linear regression (least-squares regression).

Although the DUT and LM motor torque estimates are available, they can potentially be considered inaccurate. A more reliable source of torque data is the torque sensor T_s. A key point in estimating the inertia and friction is to understand what is being measured by the torque sensor. For the case where the load motor is the driving source ($T_l > 0$), and the DUT is deactivated ($T_d = 0$), the measured torque T_s is the torque resulting from counteracting inertia and friction on the left-hand side of the torque sensor as in (3.1) with $T_d = 0$:

$$J_d\ddot{\theta}_d + B_d\dot{\theta}_d - T_s - T_d = 0 \rightarrow T_s = J_d\ddot{\theta}_d + B_d\dot{\theta}_d \tag{3.15}$$

Thus, using the measured torque T_s as observation data for the estimation method will only give estimations on J_d and B_d. Similarly, for estimating J_l and B_l, consider the case where the DUT is the driving source ($T_d > 0$), and LM is deactivated ($T_l = 0$):

$$J_l\ddot{\theta}_l + B_l\dot{\theta}_l + T_s - T_l = 0 \rightarrow -T_s = J_l\ddot{\theta}_l + B_l\dot{\theta}_l \tag{3.16}$$

With the this analysis in mind, we can create the data sets needed to perform a linear regression in the form:

$$\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{b} \tag{3.17}$$

where $\boldsymbol{Q}$ is a matrix of input variables, $\boldsymbol{b}$ is a vector of observed values and $\boldsymbol{x}$ is the parameter vector which is unknown. By rearranging (3.15) we can get it to the form of (3.17):

$$\underbrace{\begin{bmatrix}\ddot{\boldsymbol{\theta}}_{\boldsymbol{d}} & \dot{\boldsymbol{\theta}}_{\boldsymbol{d}}\end{bmatrix}}_{\boldsymbol{Q}}\underbrace{\begin{bmatrix}J_d \\ B_d\end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\boldsymbol{T}_{\boldsymbol{s}}}_{\boldsymbol{b}} \tag{3.18}$$

where $\ddot{\boldsymbol{\theta}}_{\boldsymbol{d}}$ is a vector of acceleration values, $\dot{\boldsymbol{\theta}}_{\boldsymbol{d}}$ a vector of velocity values and $\boldsymbol{T}_{\boldsymbol{s}}$ a vector of measured torque values. With the matrices $\boldsymbol{Q}$ and $\boldsymbol{b}$, we can estimate $\boldsymbol{x}$ in Matlab with:

$$\boldsymbol{x} = \boldsymbol{Q}\backslash\boldsymbol{b} \tag{3.19}$$

For the case of estimating J_d and B_d, input steps in velocity is given to the load motor while the DUT motor is deactivated, $T_d = 0$, as described earlier. Angular velocity $\dot{\theta}_d$ and torque T_s is measured. The acceleration, $\ddot{\theta}_d$, is obtained by differentiating velocity data which is low-pass filtered to reduce noise. The same procedure is performed for estimating J_l and B_l but with the DUT motor as the driving source and the load motor deactivated. Measurements from both cases is presented in figure 3.4 and figure 3.5. The measured velocity in both cases is shaft velocity measured by the encoder of the load motor, and since the shaft is stiff the following applies:

$$w_{measured} = \dot{\theta}_l \approx \dot{\theta}_d \tag{3.20}$$

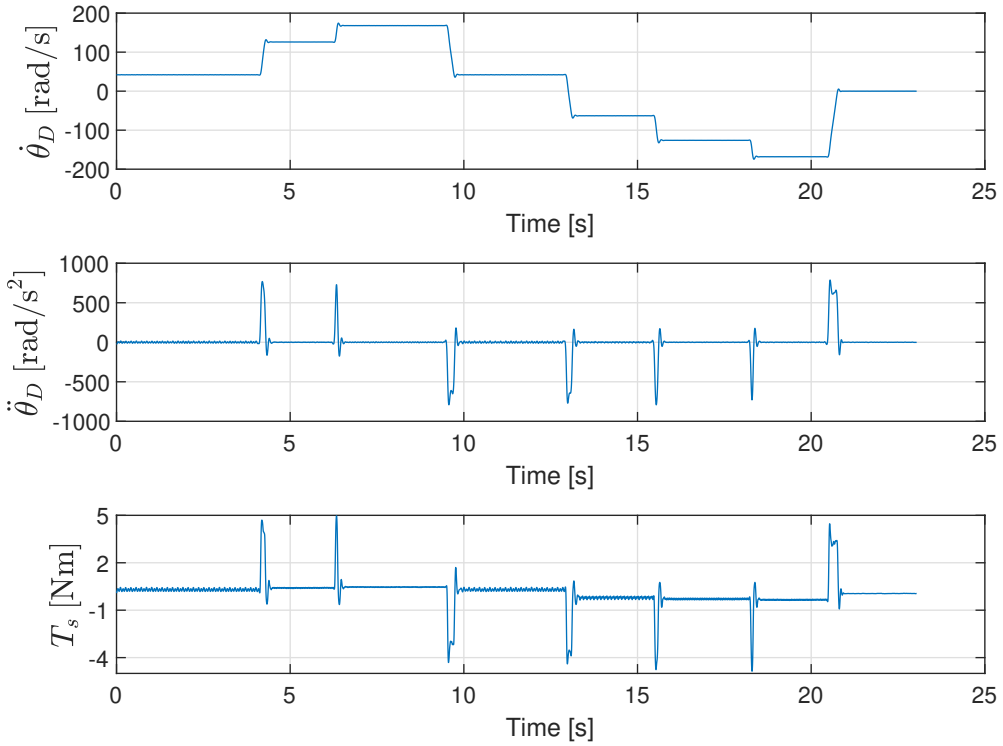


Figure 3.4: Measurement data for inertia and friction estimation. LM driving source

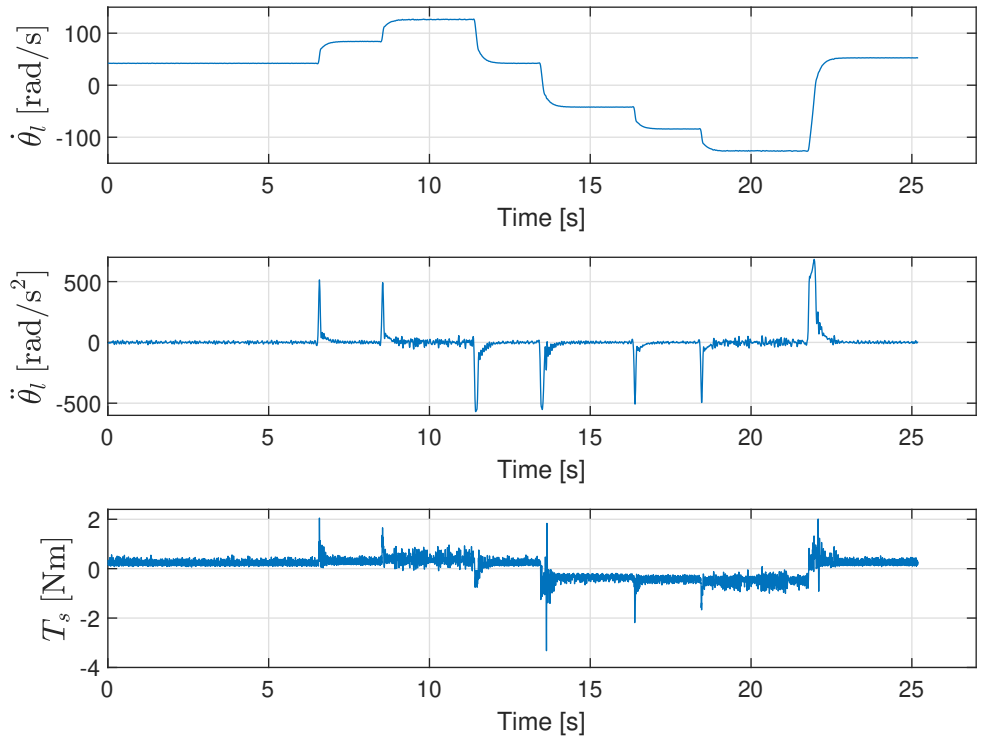


Figure 3.5: Measurement data for inertia and friction estimation. DUT driving source

The linear regression described in (3.17) is used on the measured data. The resulting estimated parameters are:

$$\begin{aligned} \tilde{J}_l &= 0.0014 \text{ kg m}^2 \\ \tilde{B}_l &= 0.0039 \text{ N m s rad}^{-1} \\ \tilde{J}_d &= 0.0057 \text{ kg m}^2 \\ \tilde{B}_d &= 0.0028 \text{ N m s rad}^{-1} \end{aligned} \tag{3.21}$$

The resulting values for inertia correspond well to the specified values for DUT and LM. The DUT rotor inertia specification is $J_d = 0.0058$ kg m^2, which is close to the estimated value. However, this value is specified for the motor rotor only and does not consider the shaft couplings and extensions present in the real system. Similarly, in the load motor's datasheet, the rotor inertia is specified to $J_l = 0.00057$ kg m^2. The estimated value $\tilde{J}_l = 0.0014$ kg m^2 is slightly higher. This is probably due to the additional inertia of the couplings and shaft extensions. The estimated values for friction cannot be compared to any real values since these are not specified for the motors.

3.2.1.3 Inertia and friction validation

To validate the estimated parameters for inertia and friction in (3.21), each side of the dynamometer system is simulated separately and compared to measured data. Simulation models for each sub-system can, intuitively, be derived by analyzing each side of the torque sensor, as can be seen in figure 3.6:

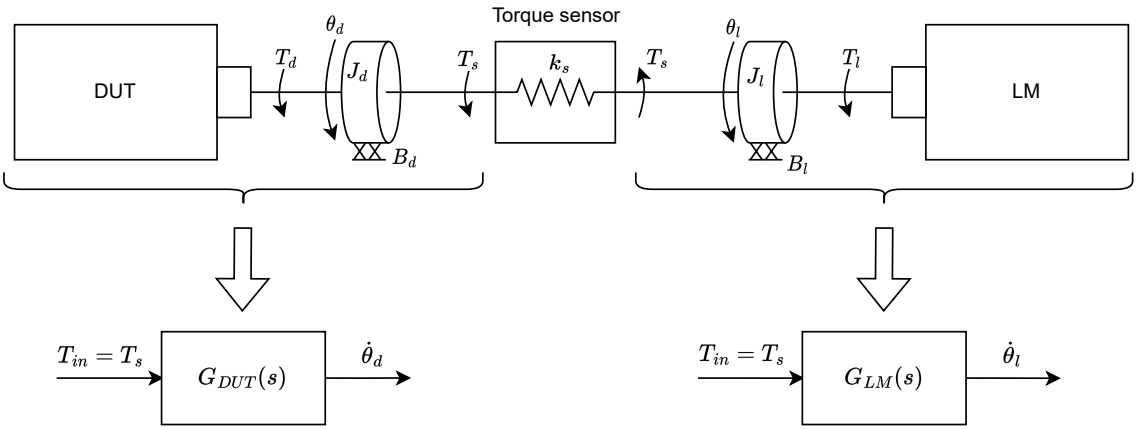


Figure 3.6: DUT and LM system models

For both systems in (3.15) and (3.16), given the input $T_s(t)$ and output $\dot{\theta}$, the corresponding transfer functions becomes:

$$G_{DUT}(s) = \frac{1}{\tilde{J}_d\, s + \tilde{B}_d} \tag{3.22}$$

$$G_{LM}(s) = \frac{1}{\tilde{J}_l\, s + \tilde{B}_l} \tag{3.23}$$

The models (3.22) and (3.23) is simulated in Matlab with the estimated parameters. For the system $G_{DUT}(s)$, LM is the driving source, and the input to the simulation is the measured torque T_s, see figure 3.4. The output speed $\dot{\theta}_d$ is compared to the actual measured speed of figure 3.4. The results of the comparison can be seen in figure 3.7.

In the same way, the system $G_{LM}(s)$ is simulated with the estimated values of $\tilde{J}_l$ and $\tilde{B}_l$. The input torque is the measured torque T_s for the case where the DUT is the driving source, as seen in figure 3.5. The resulting comparison between the model and the actual system can be seen in figure 3.8.

Evidently, both models exhibit some clear differences against the measured velocities, as can be seen in figure 3.7. The transient responses to a step change in velocity correspond well between simulated and measured outputs. This means that the inertia part of the model is correctly modeled. However, there are some mismatches when looking at the steady state output for different velocities. This would indicate that the friction, modeled as linear viscous friction, does not correspond with the real system. The velocity-dependent friction in the real system clearly has some

non-linear effects where the frictional force is higher at lower speeds.

The slightly worse model fit for the LM model in (3.8) is probably due to increased noise in the input data T_s compared to the case in figure (3.7). The inertia is also known to be less for the LM compared to the DUT, which would result in a lower T_s signal-to-noise ratio, leading to a worse estimation.

A non-linear model of the friction was initially developed and evaluated. However, this model can not be used in the emulator system since the proposed method in section 2.3 requires a first-order linear model of the dynamometer plant. Thus, a non-linear model would only improve our simulations and not the emulator's performance. Also, evaluating such a model could be hard since the inputs would be either output torque from the DUT or LM, which we do not know explicitly. Due to this, the proposed non-linear model is not presented here but in the appendix A.3, for future reference.

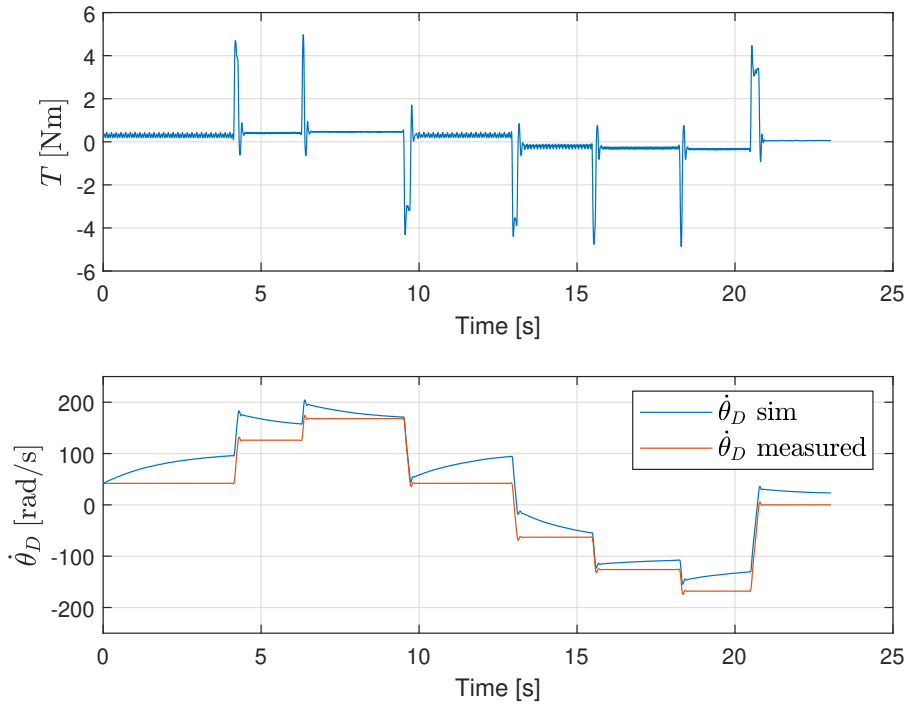


Figure 3.7: DUT system model verification. Simulated speed vs measured speed

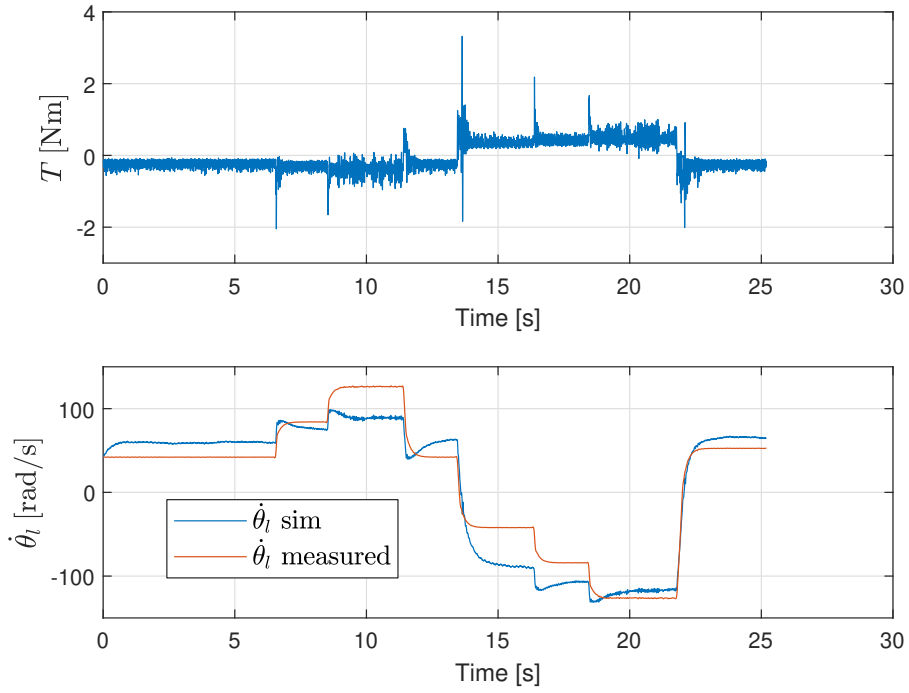


Figure 3.8: LM system model verification. Simulated speed vs measured speed

3.2.2 Torque sensor dynamics

This section presents the method for evaluating the dynamics of the analog torque sensor TW20N. Since the primary goal of this thesis is to investigate the usage of the torque sensor with the emulator, an analysis of the dynamics of the sensor is performed.

To evaluate any potential signal delays of the torque sensor, a measurement is set up where a step output torque is generated by the DUT. The current on one of the phases is measured using an oscilloscope and compared to the analog output of the torque sensor. More information regarding the specifics of the measurement can be found in the appendix section A.1.

The results from this measurement indicate that the sensor is characterized by a signal delay of approximately 1.7 ms. Furthermore, the datasheet of the sensor [14] reveals the presence of an internal low-pass filter with a cutoff frequency of 200 Hz. To accurately replicate this dynamic in the simulation environment, a model of the filter is developed, incorporating the specified cutoff frequency and input delay.

The general form of a first-order low-pass filter is:

$$H(s) = \frac{w_c}{s + w_c} \tag{3.24}$$

where w_c is the cut-off frequency in rad/s. The delay is represented by multiplying the transfer function by e^{-st_d} where t_d is the time delay in seconds. The resulting filter transfer function for the specified cut-off frequency of 200 Hz and a delay of 1.7 ms can therefore be derived as:

$$H(s) = \frac{w_c}{s + w_c} e^{-st_d} = \frac{2\pi 200}{s + 2\pi 200} e^{-s0.0017} \tag{3.25}$$

3.2.3 Load motor input delay

Similar to the previous subsection, an analysis of load motor torque output delay is performed. This is performed in order to investigate any delays from a given set-point torque command from the emulator to actual torque output on the shaft. This analysis is done since it is considered a crucial factor in the overall system behavior. The details of the measurement set-up for this analysis can be found in the appendix section A.2. The resulting load motor torque delay from this experiment is measured to approximately 2.7 ms. This delay is implemented in the simulation model of the emulator system as an output delay. This is presented in section 3.3.4.

3.2.4 Simulation model

Figure 3.9 shows the complete system model implemented in Simulink. This is used when simulating the system model in open-loop and in conjunction with the emulator. The dynamometer model used is the MIMO state-space model as presented in (3.10) and (3.11). The output matrix $\boldsymbol{C}$ is modified to output only the last two states, i.e., $\dot{\theta}_l$ and T_s. Since the angular velocity $\dot{\theta}_l$ is considered to be identical to $\dot{\theta}_d$, according to (3.20), the choice of which velocity state to use as output does not matter. The simulation also incorporates the torque sensor filter dynamics in the block `Ts_filt`, with the system presented in (3.2.2).

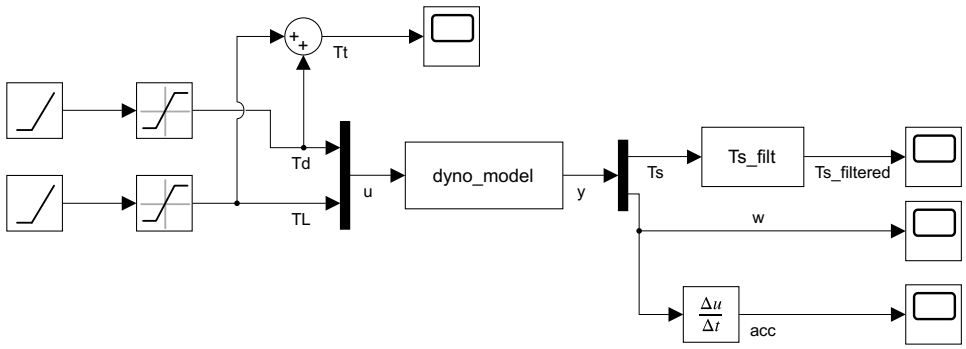


Figure 3.9: Open-loop simulation model for the dynamometer plant model

3.3 Emulator system design

The following section presents the design procedures for the emulation method presented in section 2.3. A discrete simplified model for the dynamometer plant G_{dyno} is presented based on numerical values for the inertia and friction parameters derived in 3.2.1.2. Analytical expressions for $G_t(z)$ and $G_{em}(z)$ are also presented. These are derived with control gains K_p and K_i as variables for the speed controller $G_t(z)$, and desired load inertia/friction parameters for the emulated load model $G_{em}(z)$.

3.3.1 Plant model for emulation

For the emulation method presented in section 2.3, a first-order plant $G_{dyno}(s)$ of the dynamometer system is required. Recalling that the purpose of $G_{dyno}(s)$ in (2.11) is to compensate for the inertia and friction of the dynamometer system. For this reason, a simplified model of the system plant can be created, where the coefficients for inertia and friction for the DUT and LM are lumped together:

$$G_{dyno}(s) = \frac{w(s)}{T(s)} = \frac{1}{(\tilde{J}_d + \tilde{J}_l)s + (\tilde{B}_d + \tilde{B}_l)} \tag{3.26}$$

In this model, the torque sensor element is omitted since it does not contribute anything to the model in this context. Figure 3.10 shows the step response of $G_{dyno}(s)$ together with the response of the system model (3.10) presented in (3.2). In the system model, the step is taken from input T_d to output $\dot{\theta}_d$, but the responses are the same for any combination of torque input T_l or T_d to $\dot{\theta}_l$ or $\dot{\theta}_d$.

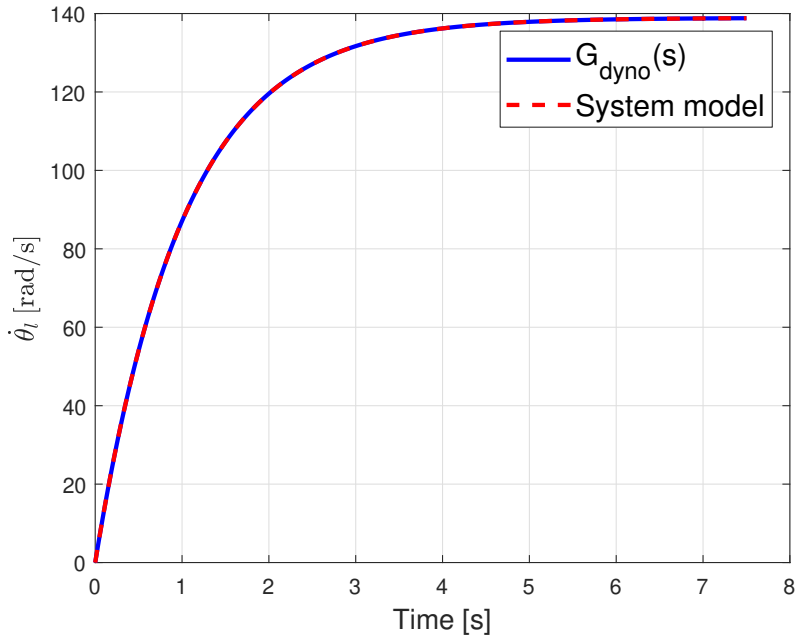


Figure 3.10: Step response for $G_{dyno}(s)$ and system model (T_d to $\dot{\theta}_d$)

The frequency response for both systems in figure 3.11 further proves that the two systems have equivalent dynamics.

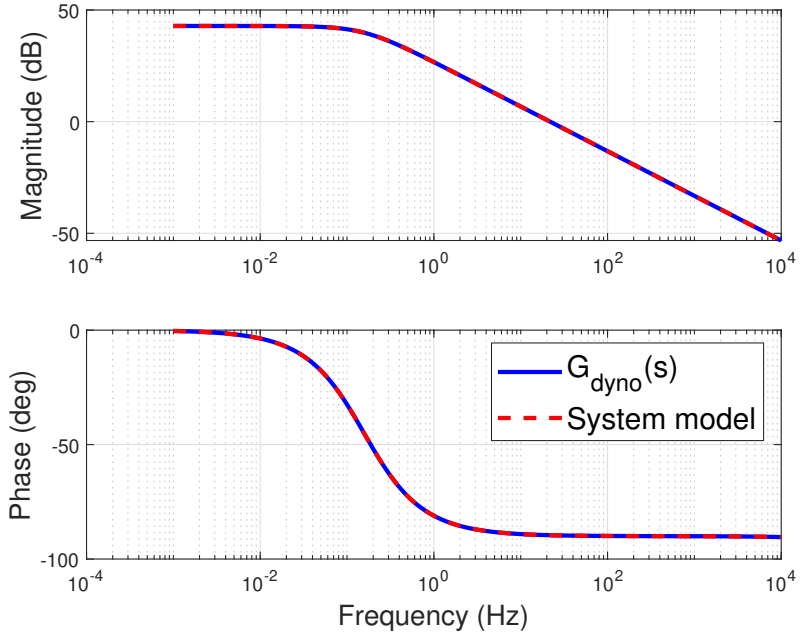


Figure 3.11: Frequency response for $G_{dyno}(s)$ and system model (T_d to $\dot{\theta}_d$)

With this in mind, the simplified model in (3.26) will be used in the emulation algorithm since it corresponds well with the system model in both time- and frequency domains. The plant model (3.26) is discretized with the zero-order hold method (2.16), using values for the inertia and friction coefficients from (3.21):

$$G_{dyno}(z) = \frac{0.2994}{z - 0.998} \tag{3.27}$$

3.3.2 Speed controller design

As the authors in [5] point out, the response of the controller $G_t(z)$ is of little concern since $G_{comp}(z)$ includes the closed-loop dynamics, and it is thus compensated for. However, this is only true if the model parameters in $G_{dyno}(z)$ are equal to the actual plant, which is often not the case. This is also shown in section 3.2.1.2 where the linear friction term did not capture the real plant dynamics. Also, the presence of sensor and actuator delays that is shown in section 3.2.2 and 3.2.3 will make the compensation by $G_{comp}(z)$ inaccurate. With this in mind, the speed controller $G_t(z)$ needs to be tuned to handle these uncertainties while retaining overall system stability.

The discrete controller is tuned in Matlab with 0 dB gain crossover frequency, w_c, as a tuning parameter against the plant system $G_{dyno}(z)$ in (3.27). Figure 3.12 shows a step response of the closed-loop system with the controller $G_t(z)$ and $G_{dyno}(z)$ as

plant. The controller is tuned for a crossover frequency w_c of 15 rad/s and 30 rad/s, respectively, which is similar to the response of the controller used in [5]. The tuned controller results in the following control gains K_p and K_i:

$$w_c = 15 \text{ rad/s} \Rightarrow \begin{cases} K_p = 0.089 \\ K_i = 0.86 \end{cases} \tag{3.28}$$

$$w_c = 30 \text{ rad/s} \Rightarrow \begin{cases} K_p = 0.18 \\ K_i = 3.16 \end{cases} \tag{3.29}$$

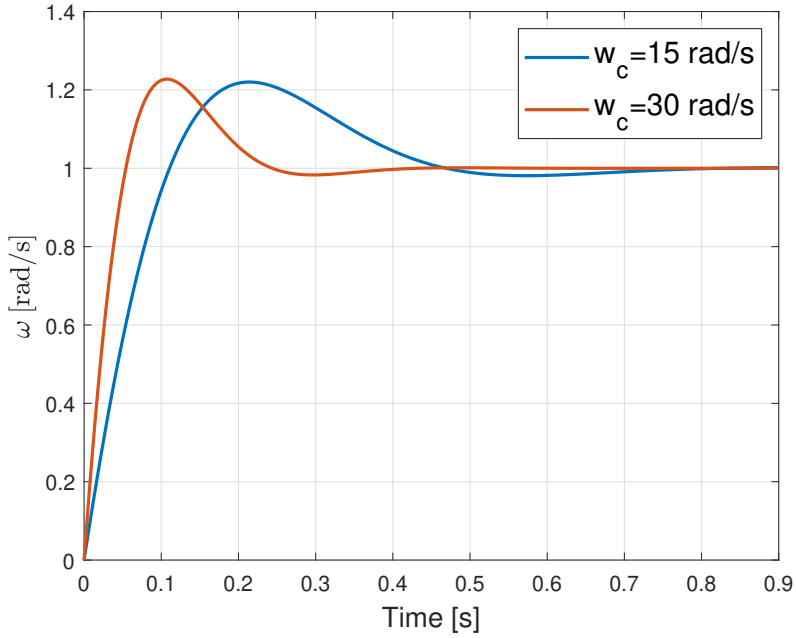


Figure 3.12: Step response for closed-loop system with $G_t(z)$ and system model $G_{dyno}(z)$

As expected, the response of w_c=15 rad/s is slower than that of w_c=30 rad/s. Both controllers manage to reach the target steady state below 1s. As the authors in [5] points out, the relatively slow response of G_t is considered not to be an issue due to the closed-loop compensation of G_{comp}.

Deriving an analytical expression for $G_t(z)$ starts with the expression for a continuous PI controller in parallel form:

$$G_t(s) = K_p + \frac{K_i}{s} \tag{3.30}$$

where K_p and K_i are the control gains for the proportional and integral parts, respectively. Using Backward Euler method as discretization method, $G_t(z)$ can be

represented as:

$$G_t(z) = K_p + K_i \frac{t_s z}{z-1} = \frac{z(K_p + K_i t_s) - K_p}{z-1} \tag{3.31}$$

As mentioned earlier, the control gains K_p and K_i are obtained by tuning the controller against the system plant in Matlab with different values for w_c. These gains can be inserted in (3.31).

3.3.3 Emulated load model

For linear loads with inertia and friction, the model (2.5) is used as the emulated load model. The discretization of $G_{em}(s)$ is done in the same way as for the system plant $G_{dyno}(s)$ by applying the zero-order hold method. For a load model $G_{em}(s)$ with inertia J_{em} and B_{em}, the discrete version can be expressed as the following according to 2.16

$$G_{em}(z) = \frac{(1/B_{em})(1 - e^{-(B_{em}/J_{em})t_s})}{z - e^{-(B_{em}/J_{em})t_s}} \tag{3.32}$$

The numerical value for all the constants in (3.32) can then conveniently be calculated for the specific load to be emulated.

3.3.4 Simulation model

The simulation model for the emulator is shown in figure 3.13. This is essentially the emulator part of the system presented in figure 2.7. To accommodate for the input delay on the load motor (as explained in 3.2.3), a discrete-time unit delay z^{-1} is placed on the output of the emulator, $T_{LM}(z)$ in figure 2.7. This will delay the input signal to the load motor by one sample, which is roughly equal to the measured delay of 2.7ms.

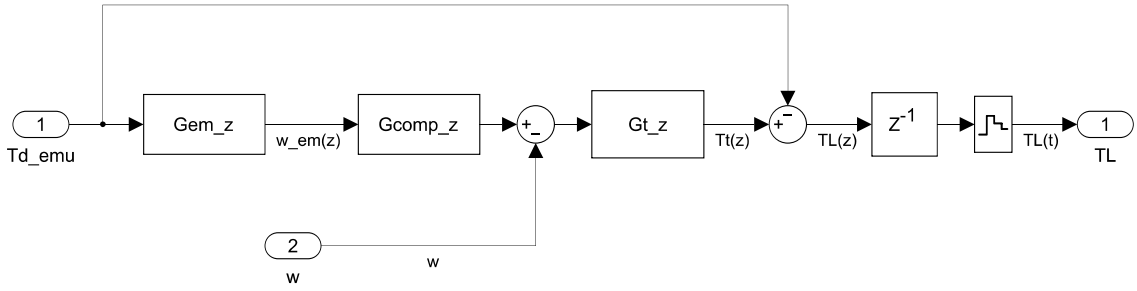


Figure 3.13: Emulator subsystem

The subsystems $G_{em}(z)$, $G_{comp}(z)$, and $G_t(z)$ used in the simulation model is created as explained in section 3.3. The sampling time used for discretization is set to the specified sampling time of $t_s = 2.1$ ms as specified in section 3.1.1.

3.3.4.1 Emulated load annotation

Throughout the thesis, a certain notation is used when differentiating between different emulated loads. This applies to both simulations and experiments. Each load case has a notation that represents the emulation of a load based on the total inertia and friction of the dynamometer itself, J_{dyn} and B_{dyn}, with multipliers n and m for each term:

$$Jem = n\, J_{dyn} = n\, (\tilde{J}_d + \tilde{J}_l)$$
$$Bem = m\, B_{dyn} = m\, (\tilde{B}_d + \tilde{B}_l)$$

For example, an emulated load with ten times the inertia and five times the friction of the dynamometer will have the notation: $10\mathrm{J}_{\mathrm{dyn}}5\mathrm{B}_{\mathrm{dyn}}$.

3.4 Input reference torque - System evaluation

A primary goal of this thesis is to investigate the use of torque measurements from the torque sensor as input to the emulation algorithm presented in section 2.3. The thought is that this will lead to more accurate and consistent emulation results since the torque sensor can potentially measure the *actual* output torque of the DUT and thus produce the correct emulation.

The proposed input to the emulator in section 2.3 is the DUT reference/set-point torque. One negative aspect of using this concept is that the set-point torque on the DUT is not guaranteed to be equal to the actual torque output. Even if the torque controller perfectly follows its target torque, the target torque is *estimated* based on phase currents and motor torque constant. On many of the motors planned to be tested in this dynamometer set-up, which is often sensorless PMSM machines, the accuracy of the estimated torque will, under some circumstances, be poor. Based on measurements, an estimation error of around $\pm 10\%$ is expected. Also, the accuracy can potentially deviate between DUT samples since it depends heavily on the type of drive controller implementation.

As will be seen from the frequency analysis performed in this section, using the measured torque from the sensor as input to the emulator has some distinct effects on the overall system dynamics. For this purpose, a few cases are presented in the following sections, where each case has different input sources of reference torque to the emulator. For each case, a Simulink model is presented together with a frequency response analysis for the corresponding system. When analyzing the frequency responses of each case, the emulator subsystem is designed as explained earlier in section 3.3. The crossover frequency of $G_t(z)$ is set to $w_c = 30$ rad/s for all the cases.

3.4.1 Case I - DUT set-point torque

The first case uses the internally generated set-point torque from the DUT as input to the emulator, as presented in section 2.3. This method is used in [5], and also illustrated in figure 2.7. Figure 3.14 shows the Simulink model for this case. The model has an adjustable error gain on the input to the emulator to simulate errors between the set-point torque and the actual DUT torque output T_d. The simulation also includes $G_{em}(s)$ from (2.5), which outputs the target/ideal velocity that the simulated velocity can be compared against.

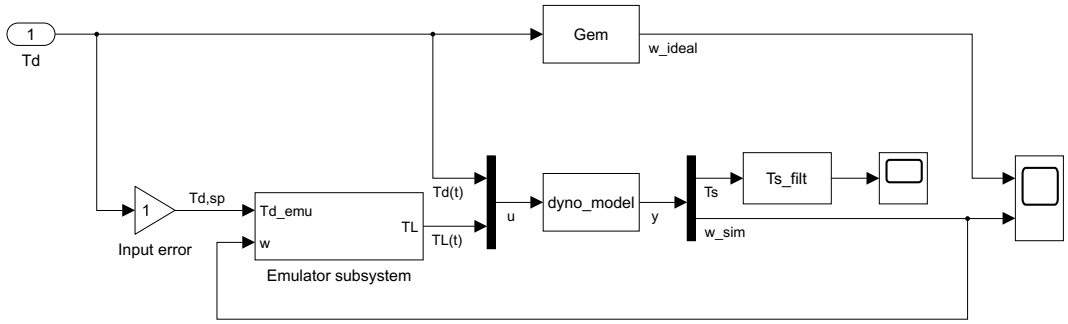


Figure 3.14: Simulink model for the emulator with feedforward-tracking controller. Input torque T_d is the DUT set-point torque.

To analyze the frequency response of the entire system, an equivalent SISO system can be created by analyzing the entire system from input T_d to output w in figure 3.14. This is done in Matlab using built-in functions for interconnecting dynamic systems. Prior to this step, the dynamometer plant model, including the output filter for T_s, is discretized to obtain all the systems in the discrete-time domain. The resulting frequency responses are shown in figure 3.15 and 3.16, for three different load emulations respectively. The annotation of each load case follows what is described in section 3.3.4.1.

As can be seen from these responses, the $1J_{dyn}1B_{dyn}$ load case is very similar to the response of the dynamometer plant model in 3.11, as it should be. Also, it is evident from figure 3.16 that adding friction to the emulated load does not seem to affect the dynamics of the entire system to any large extent. The simulation- and experimental results for this case are presented in section 4.2.1.

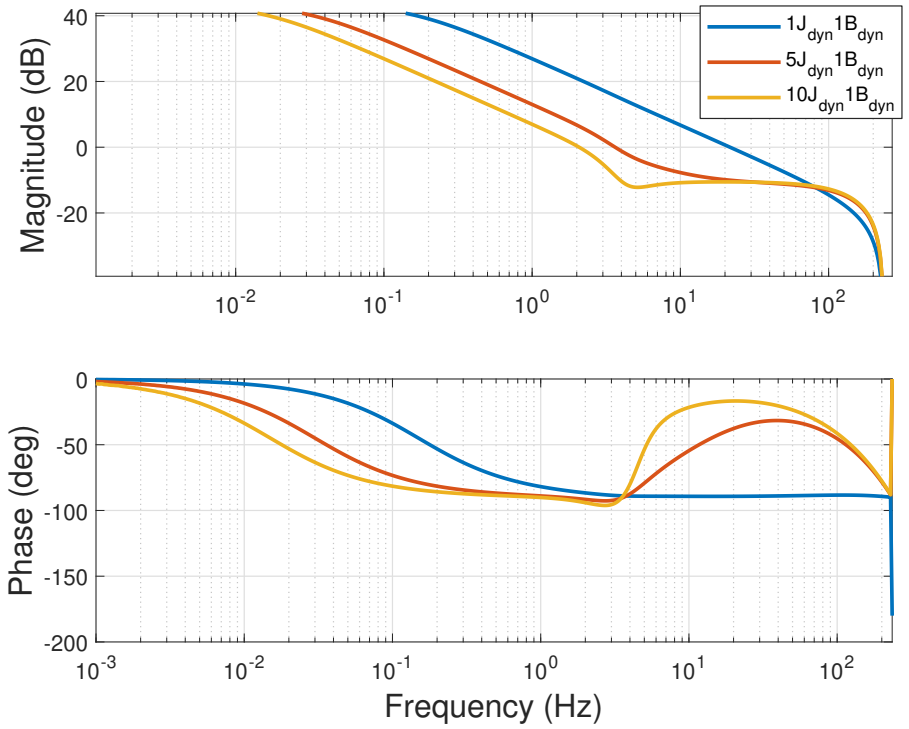


Figure 3.15: Frequency response for the complete system for case I. Different values for emulated load inertia.

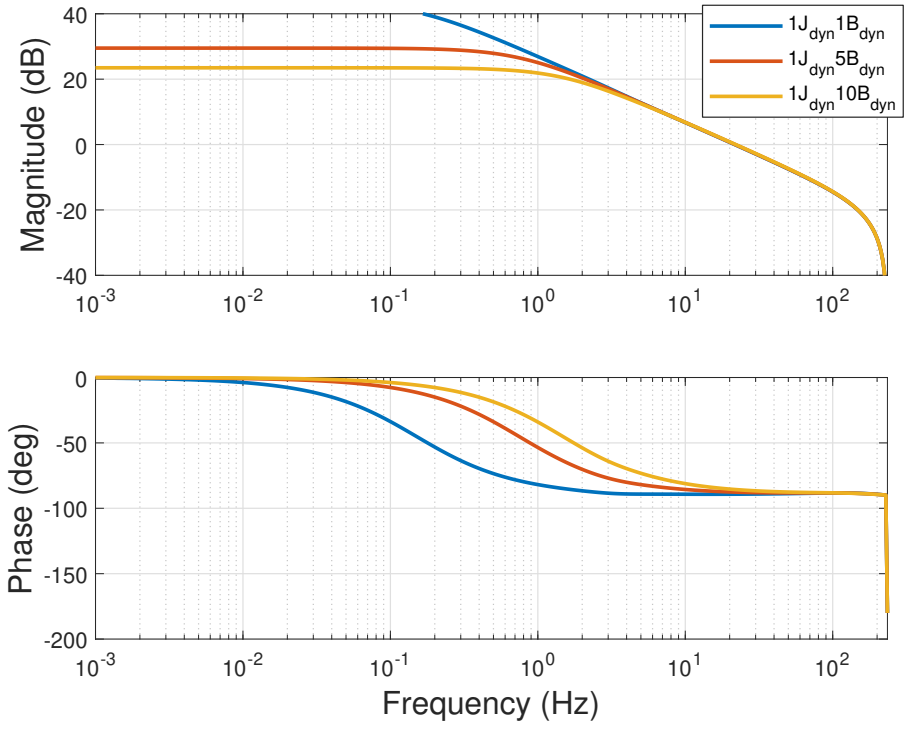


Figure 3.16: Frequency response for the complete system for case I. Different values for emulated load friction.

3.4.2 Case II - Measured torque without compensation

In the second case, the input reference torque is based entirely on the torque sensor readings. Recalling the system model equation 3.1:

$$J_d\ddot{\theta}_d + B_d\dot{\theta}_d - T_s - T_d = 0 \tag{3.33}$$

If the inertia and friction terms are neglected, the measured torque T_s can give a rough estimation of T_d by:

$$T_{d,est} = -T_s \tag{3.34}$$

Figure 3.17 shows the Simulink model for this case. A simulated DUT output torque T_d is generated and acts on the dynamometer model plant. This torque is then estimated using (3.34) and fed back as an input to the emulation system.

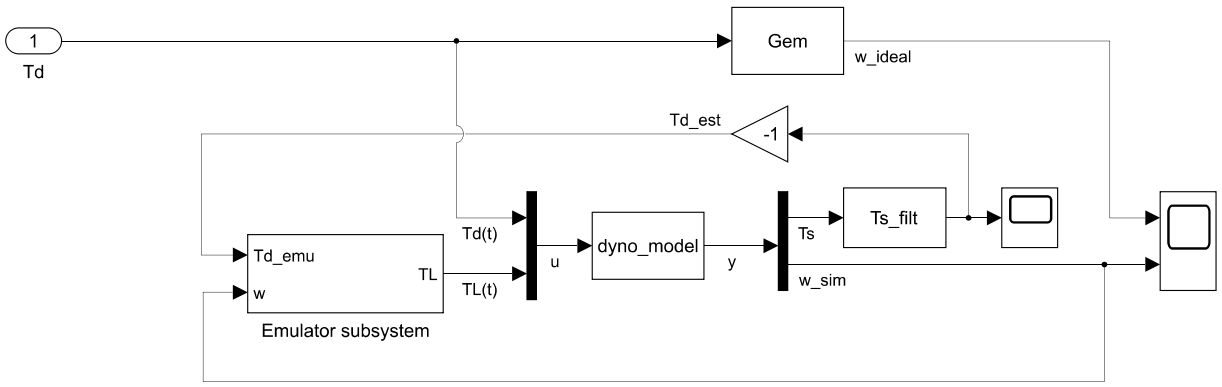


Figure 3.17: Simulink model for the emulator with feedforward-tracking controller. Input torque $T_{d,est}$ is the estimated torque as in (3.34).

As for the previous case, an equivalent SISO system is derived for the system in figure 3.17, and the frequency responses for different load emulations are presented in figure 3.18 and 3.19.

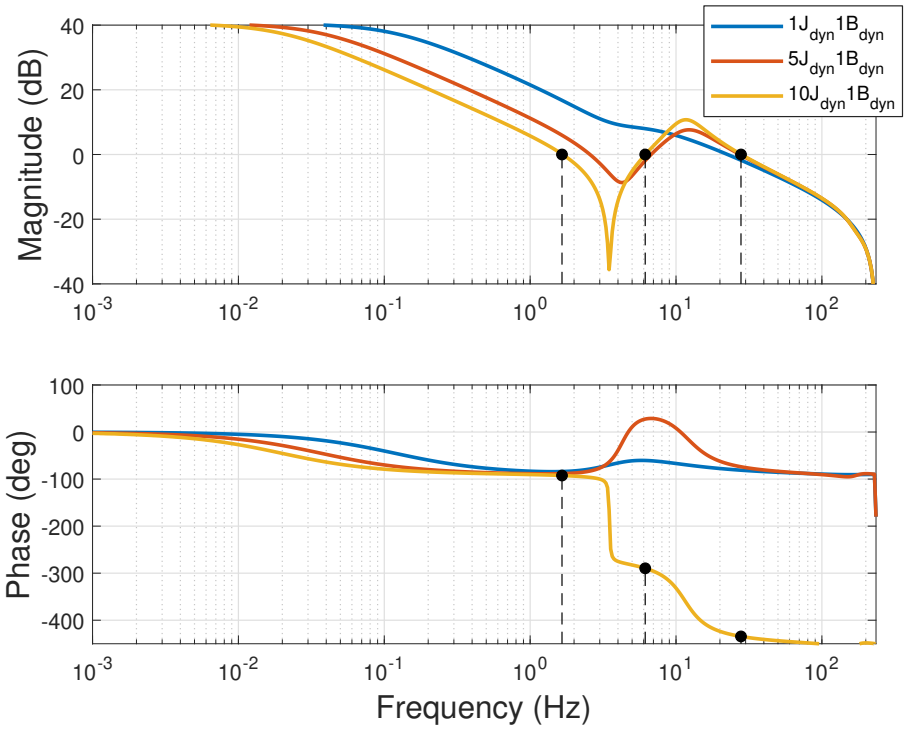


Figure 3.18: Frequency response for the complete system. 0 dB gain crossover frequencies are marked for $10J_{dyn}1B_{dyn}$ data

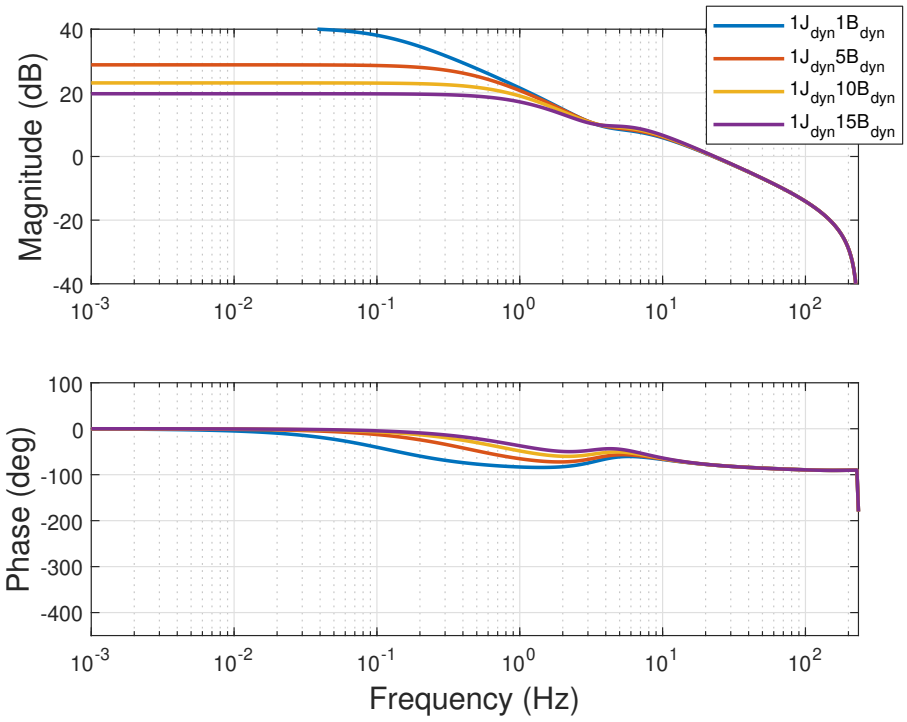


Figure 3.19: Frequency response for the complete system for case II. Load emulation with increasing friction.

An important implication from this method is that the measured torque T_s is the result of the *coupled* torque between T_d and T_l. Thus, the estimated torque $T_{d,est}$ will also be coupled with the load torque, and an additional feedback path is introduced in the system. This has distinct effects on the overall system stability, as seen from the frequency response in figure 3.18. Here, we can clearly observe resonance peaks for both $5\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$ and $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$ load cases at approximately $f \approx 10$ Hz. It is also very likely that the emulation for $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$ will be unstable since the phase margin is negative.

Figure 3.20 shows this instability effect on the complete system as a result of emulating higher inertia. It is evident that increasing the inertia in the emulated load also increases the magnitude of the resonance peak at $f = 10$ Hz, and makes the system more prone to instability. The effect on the system dynamics using different tuning parameters on $G_t(z)$ is shown in figure 3.21. The emulated load inertia is set to $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$. As seen from this figure, a slower speed controller $G_t(z)$ can make the system stable for this load case. However, the resonance peaks persist in all of the values for w_c. Figure 3.19 shows that emulating higher values for friction does not seem to generate any resonance or instability behavior.

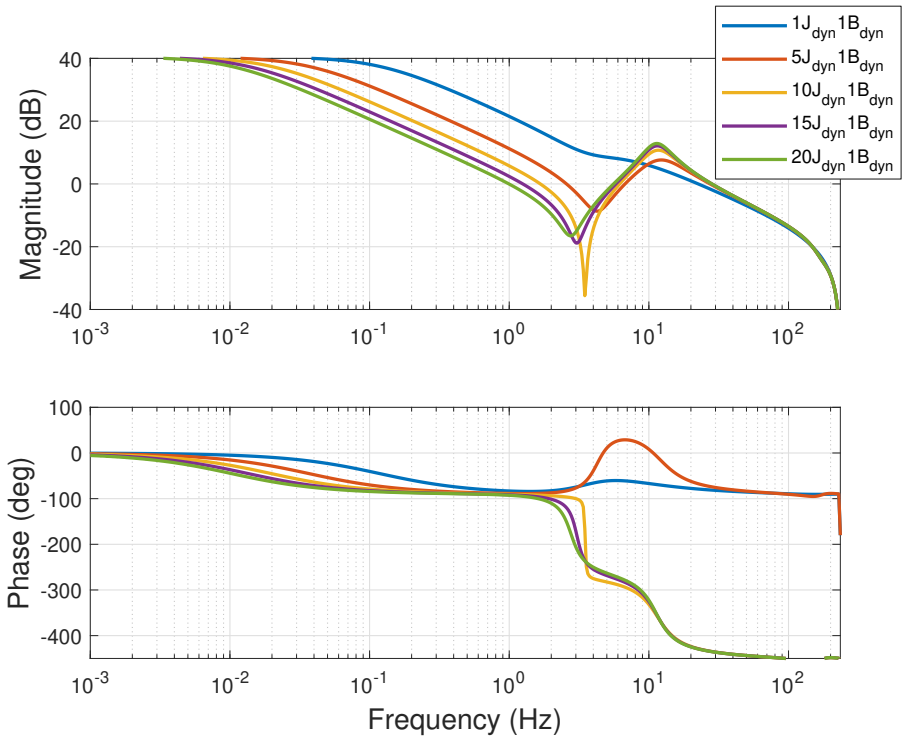


Figure 3.20: Frequency response for the complete system for case II. Load emulation with increasing inertia.

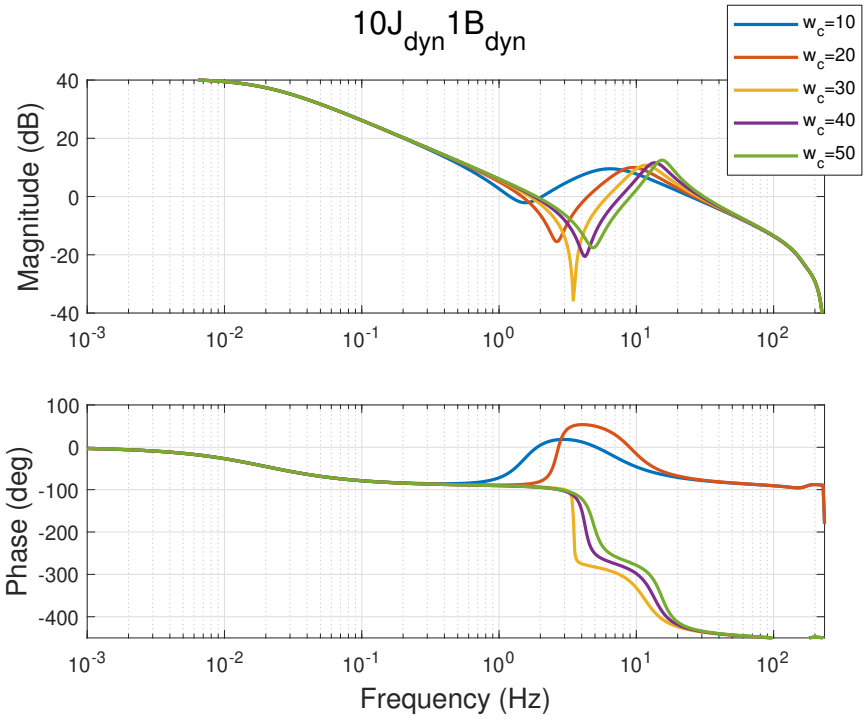


Figure 3.21: Frequency response for the complete system for case II. Load emulation with different tuning parameters w_c for the speed controller G_t.

3.4.3 Case III - Measured torque with compensation

The third and final case presents a possible solution to the instability effects observed in case II. Here, the acceleration and velocity terms in (3.1) are included. The idea is that by including the acceleration in the estimation, it will compensate for the contribution of T_l, effectively removing the coupling effect and thus the feedback path that is believed to introduce the instability issues and resonances.

By including the acceleration and velocity terms in (3.1), we get:

$$T_{d,est} = \tilde{J}_d\ddot{\theta}_d + \tilde{B}_d\dot{\theta}_d - T_s \tag{3.35}$$

where the inertia and friction parameters $\tilde{J}_d$ and $\tilde{B}_d$ correspond to the estimated values as presented in section 3.2.1.2. As can be seen from (3.35), the measured angular speed $\dot{\theta}_d$ must be differentiated to get the acceleration $\ddot{\theta}_d$. For this purpose, a filtered derivative function [6] is used, which combines differentiation and low-pass filtering:

$$G_{diff}(s) = \frac{Ks}{\tau s + 1} \tag{3.36}$$

where K is the gain factor and τ is the time constant for the low-pass filter. The input to 3.36 is thus angular speed $\dot{\theta}_d$ and output is filtered acceleration $\ddot{\theta}_d$. The gain factor is set to 1, and the time constant is calculated from a cut-off frequency f_c with:

$$\tau = \frac{1}{2\pi f_c} \tag{3.37}$$

Figure 3.22 shows the Simulink model for this case. The estimation block contains the model (3.35), together with the filtered derivative in 3.36. The estimation block subsystem can be seen in figure 3.23. When implementing the filter on the physical system, the transfer function in (3.36) is discretized in Matlab using Tustin's method and implemented as a regular difference equation.

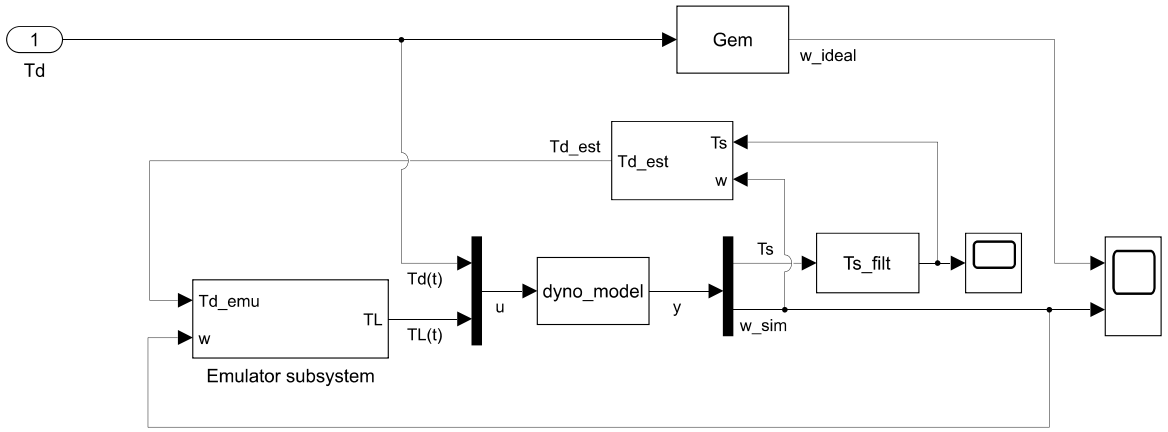


Figure 3.22: Simulink model for the emulator with a feedforward-tracking controller. Input torque $T_{d,est}$ is the estimated torque as in (3.35).

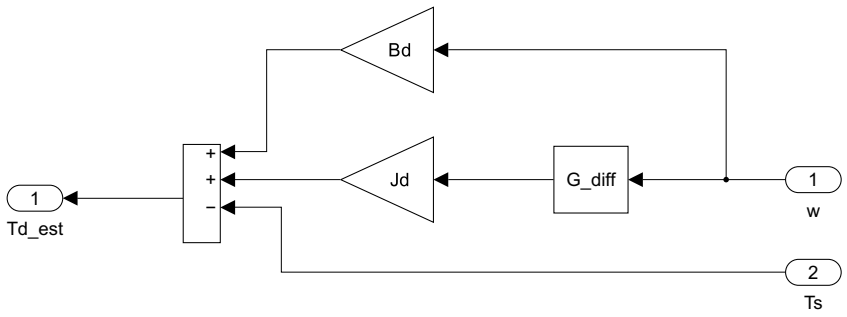


Figure 3.23: Subsystem block for $T_{d,est}$ as in (3.35)

As in the previous cases, the complete system in figure 3.22 is analyzed in the frequency domain. Figure 3.24 shows the resulting frequency responses where $G_{diff}(s)$ is set with a cut-off frequency of $f_c = 200$ Hz. As can be seen in the responses for the emulations $5\text{J}_{\text{dyn}}1\text{B}_{\text{dyn}}$ and $10\text{J}_{\text{dyn}}1\text{B}_{\text{dyn}}$, the resonance peaks are clearly suppressed and the phase margins are positive. The responses are very similar to those from case I in figure 3.15, indicating that this compensation method can preserve the system dynamics to the desired one.

On the physical system, however, the angular speed $\dot{\theta}_d$ will be subjected to noise. As a result of this, the low-pass filter in $G_{diff}(s)$ needs to be adjusted in order to produce a sufficient noise-free acceleration output. The implication of choosing a lower cut-off frequency results in smoother acceleration data and, in turn, a smoother $T_{d,est}$, but will also lead to a larger phase-delay of the signal and cause the estimation in (3.35) to lag behind.

Figure 3.25 shows the implication of lowering the cut-off frequency on the system dynamics. As can be seen, lowering the cut-off frequency re-introduces the resonance peak to some extent. Based on experimental measurements, setting the cut-off frequency to $f_c = 20$ Hz resulted in an acceptable compromise between noise suppression and phase delay.

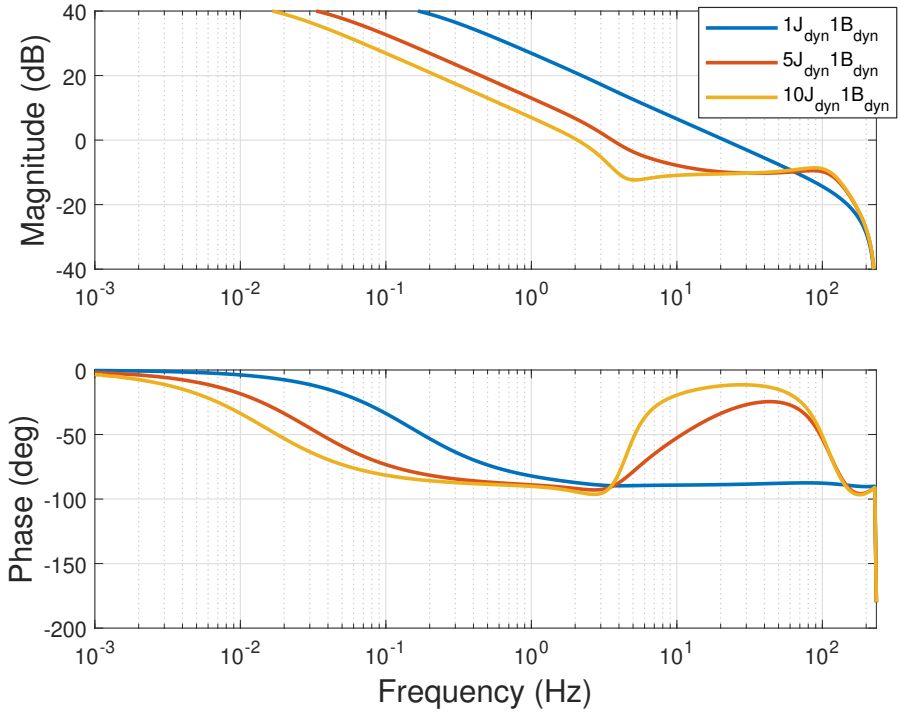


Figure 3.24: Frequency response for the complete system for case III with variations of emulated inertia. Cut-off frequency for $G_{diff}(s)$ is set to $f_c = 200$ Hz.

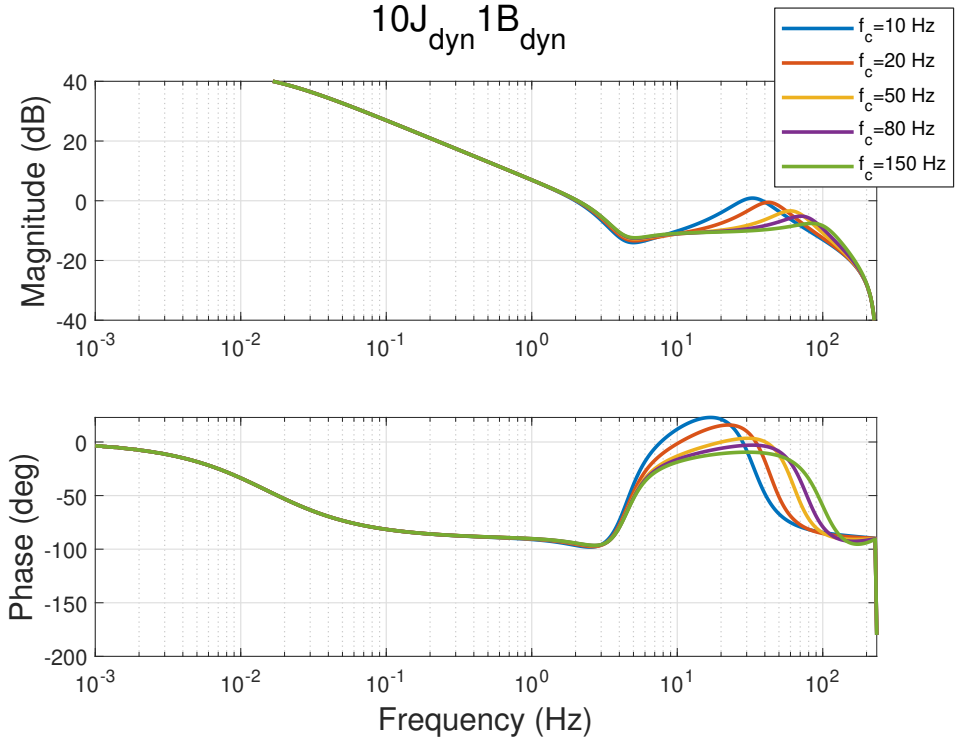


Figure 3.25: Frequency response for the complete system for case III with different cut-off frequency values for $G_{diff}(s)$. Emulated load is set to $10\mathrm{J}_{\mathrm{dyn}}1\mathrm{B}_{\mathrm{dyn}}$.

4

Results

The following chapter presents results from simulations and physical measurements on the dynamometer system. The chapter initially presents results from simulations of the system model. These simulations are performed in open-loop without the emulator system in order to evaluate the validity of the model itself. The chapter continues with results from the complete system, including emulator, for the different implementations presented in section 3.4.

4.1 Dynamometer model simulation

The linear state-space system model in (3.10) and (3.11) is simulated in an open-loop fashion using Matlab/Simulink, as presented in section 3.2.4. The results can be seen in figure 4.1. Recalling that the inputs to the system T_d and T_l is the torque generated by the DUT and load motor, respectively. In this simulation, they consisted of ramp inputs starting at $t = 0.5s$, $t = 3s$ with an amplitude of $T_d = 15\,\text{Nm}$ and $T_l = -11\,\text{Nm}$. The simulation will simulate a case where the velocity increases due to the output torque of the DUT. Then, at $t = 3s$, the load motor applies a torque in the opposite direction and the velocity should decrease until it reaches a steady state.

The output is the resulting angular speed $\dot{\theta}_l$ which can be said to be equal to $\dot{\theta}_d$, according to the motivation in 3.3.1. The figure also shows the resulting acceleration $\ddot{\theta}_l$. T_s is the simulated torque output from the torque sensor and T_t is the net torque on the axis.

It follows naturally that the resulting net torque T_t after t=3s is the torque required to rotate the axis with the steady state speed of $w \approx 600$ rad/s with the relationship:

$$T_t = T_d + T_l = \dot{\theta}_l B_{tot} = \dot{\theta}_l(\tilde{B}_d + \tilde{B}_l) = 600(0.0028 + 0.0039) \approx 4.0Nm \qquad (4.1)$$

Furthermore, it can be seen that the steady state value of T_s is that of (3.2), namely the contribution from T_l plus the additional frictional torque contribution from the $\tilde{B}_l\dot{\theta}_l$ term. From the plot, it is also obvious that the torque sensor output T_s is affected from both T_d and T_l, or rather the acceleration that comes from the applied torques.

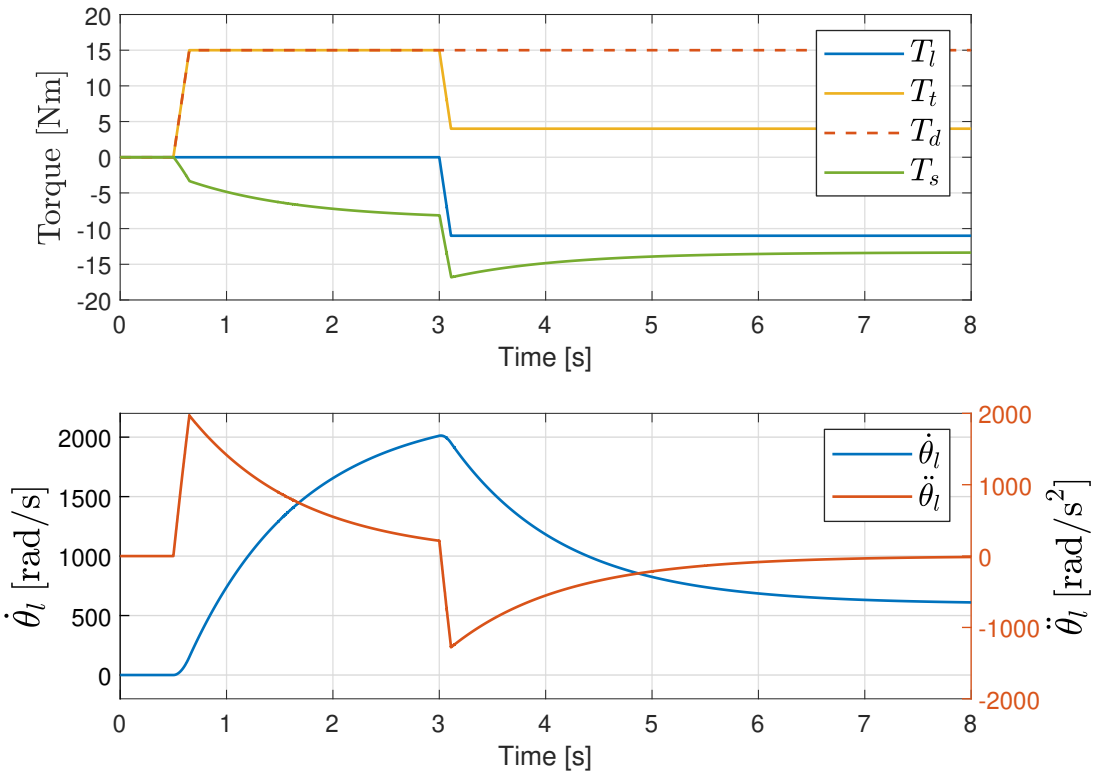


Figure 4.1: System model simulation

4.1.1 Estimated reference torque simulation

This section presents open-loop simulation data where the DUT output torque T_d is estimated based on torque sensor readings, as presented in section 3.4.2 and 3.4.3. The purpose of the simulations in this section is to evaluate the difference between the two estimations without the influence of the emulator. During these open-loop simulations, the emulator sub-systems in figures 3.17 and 3.22 is not included, and the inputs to the dynamometer plant T_l, T_d consists of arbitrary ramped pulses, similar to those presented in figure 4.1.

Figure 4.2 shows simulation data for the case where $T_{d,est} = -T_s$, as presented in section 3.4.2. The simulation results show that this estimation is quite poor when no torque from the load motor is applied, $T_l = 0$. This is because the inertial and frictional torques from $J_d\ddot{\theta}_d$ and $B_d\dot{\theta}_d$ is omitted from the estimation, and these torques are in the same order of magnitude as the counteracting torque that is being measured. The counteracting torque is that from the inertia and friction caused by the load motor since $T_l = 0$:

$$T_s = -(J_l\ddot{\theta}_l + B_l\dot{\theta}_l) + T_l = -(J_l\ddot{\theta}_l + B_l\dot{\theta}_l) \tag{4.2}$$

So, by not accounting for the inertia and friction of the DUT itself, the estimated reference torque is almost half of what it should be when $T_l = 0$. However, when a load torque is applied at $t = 3s$, the inertial and frictional torques from the DUT are much lower compared to the measured torque, and the estimation is more correct.

At $t = 5s$, a change in T_l is applied to show that the estimated torque is directly coupled with T_l and will, in turn, be affected by any changes in acceleration caused by T_l. This is referred to as the coupling effect in section 3.4.2.

Figure 4.3 shows simulation data for the third case where the DUT output torque T_d is estimated from (3.35) with the acceleration and velocity terms included, as presented in section 3.4.3. The cut-off frequency for the derivative filter is set to $f_c = 20$ Hz in this simulation. As can be seen, the estimation is improved compared to the previous result in figure 4.2. The estimation is also not as affected by changes in T_l. The small effects seen on $T_{d,est}$ when T_l changes are due to the derivative filter. The output acceleration from this filter has a notable phase delay due to the relatively low cut-off frequency. And since the torque sensor readings T_s are affected instantly by a change in T_l, the estimation in (3.35) inherits this change under some time before the acceleration term becomes effective. This effect can also be seen in the emulator simulation results for case III, in section 4.2.3.1.

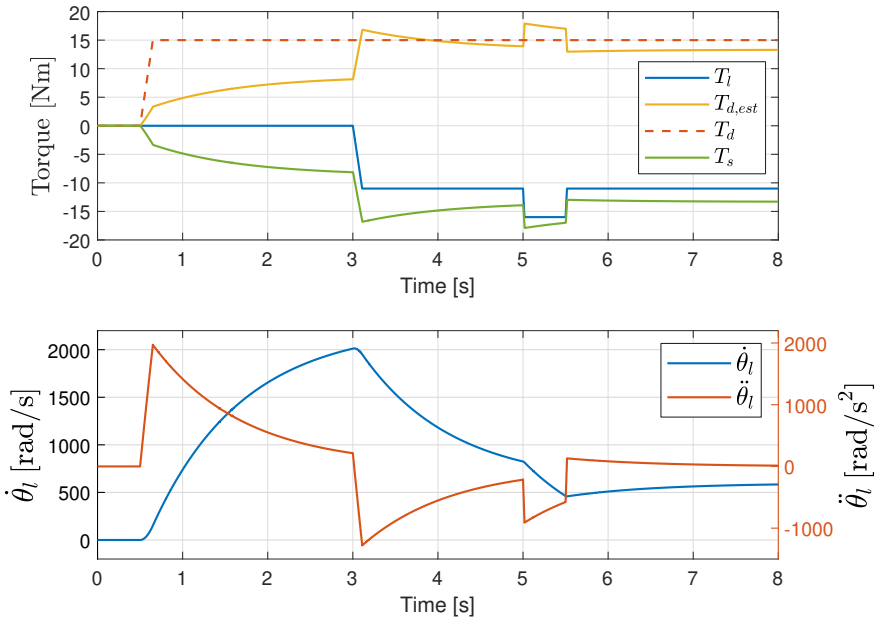


Figure 4.2: Simulation with estimated reference torque from (3.34). Acceleration and velocity terms are not included.

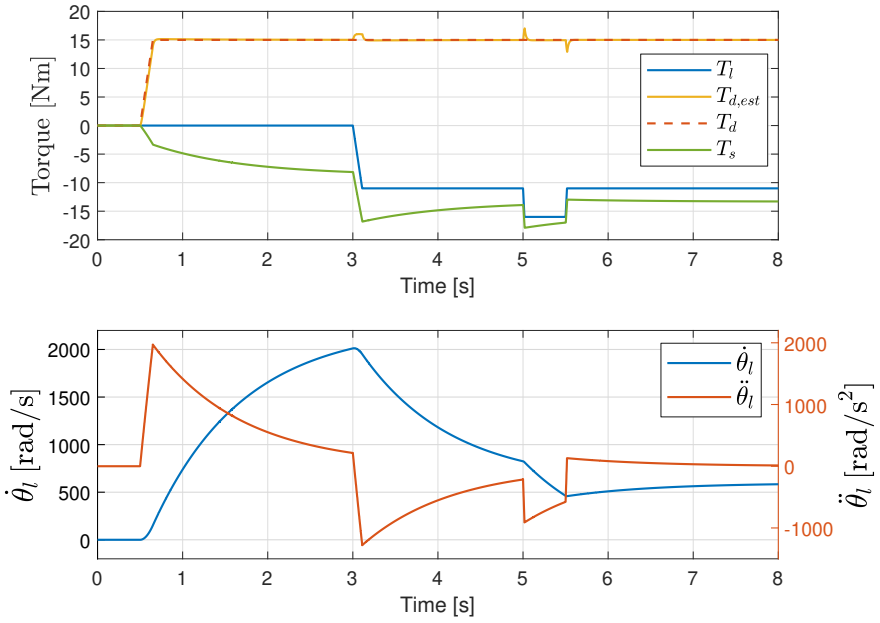


Figure 4.3: Simulation with estimated reference torque from (3.35). Acceleration and velocity terms are included.

4.2 Emulator system

The following section presents simulation results and measured data on the physical system to evaluate the performance of the emulator presented in section 2.3. The results are presented with different cases of input reference torque, as presented in section 3.4.

For each case (I-III), time-domain simulation and experimental results are presented and compared to the target velocity, which is the output of $G_{em}(s)$ in figures 3.14-3.22. This velocity acts as a benchmark for how well the emulator performs.

Throughout this section, results for different emulator load cases are presented. The annotation of each load case follows what is described in section 3.3.4.1.

4.2.1 Case I - DUT reference torque

The following section presents simulation and experimental results from the implemented emulator presented in section 3.4.1. For the simulation and in the physical system, the speed controller $G_t(z)$ is tuned with a crossover frequency of $w_c = 30$ rad/s, with the resulting control gains as specified in 3.29.

4.2.1.1 Simulation results

The simulation model in figure 3.14 is implemented and simulated in Simulink. The simulated DUT output torque T_d consists of pulses at different times. Initially, the input error gain is set to 1, which means that the input torque to the emulator is equal to T_d as it would be in an ideal condition.

Figure 4.4 and 4.5 show results for three different load cases each, with varying inertia and varying friction, respectively. The resulting output angular velocity is compared to the target velocity as explained earlier. The plots also show the difference between these two. The figures also present the resulting output torque T_l from the emulator for each load case.

As can be seen from these figures, the output velocity follows the target velocity very closely. The largest errors occur at the input transients. From the magnification in figure 4.4 at $t = 5.5\ s$, it can be seen that the emulator has some trouble in reacting to a large input change. This is likely due to the fact that the system has an actuator input delay on the load motor, as explained in 3.2.3.

Figure 4.6 presents results from emulating a load with lower inertia than that of the dynamometer itself. The emulated inertia is set to half while the friction remains unchanged. As can be expected, the load motor torque T_l now applies torque in the same direction as the DUT at the beginning of the input step to emulate a lower inertia than the dynamometer actually has.

Figure 4.7 presents results from a simulation where the input torque to the emulator $T_{d,sp}$ has an error of +15% with respect to the actual DUT output torque T_d. The emulated load, in this case, is set to $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$. As expected, the resulting emulated velocity deviates from the target velocity, which is based on the actual DUT torque.

It can also be seen that the emulated velocity exhibits larger overshoots at the input transients compared to the previous plots. This is due to the fact that the output torque from the emulator, T_l, will initially be too large at an input transition since the input torque is less than what the emulator sees.

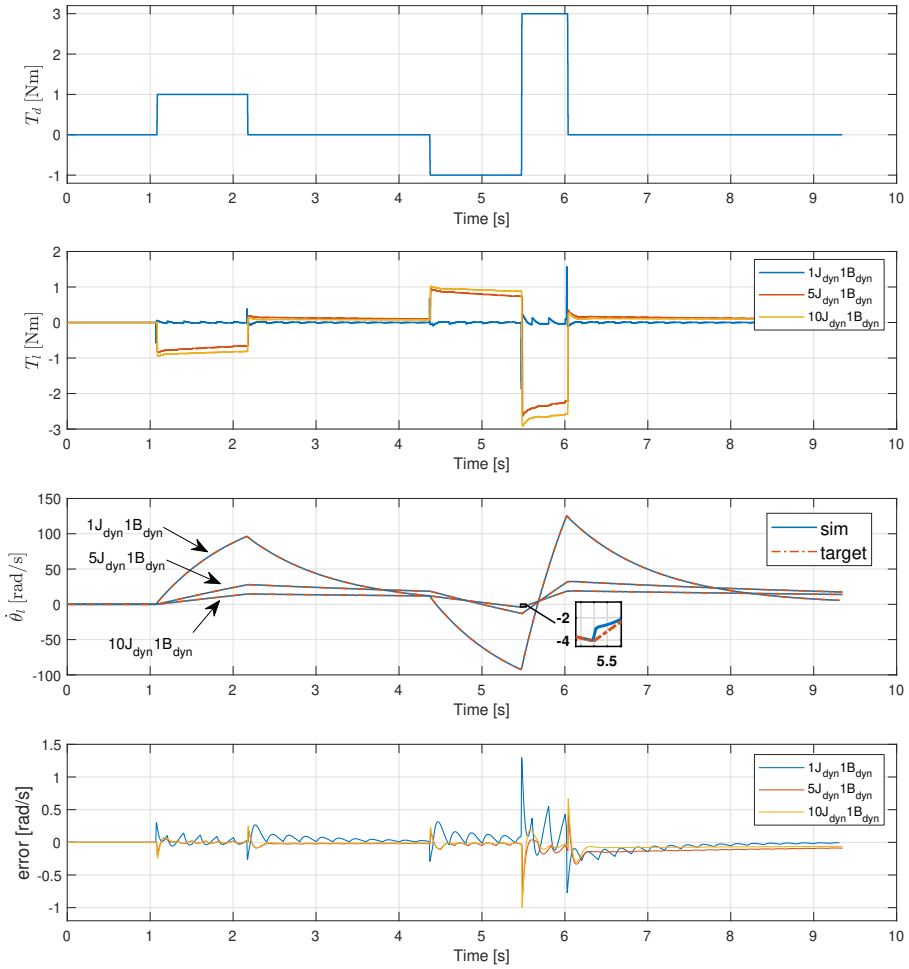


Figure 4.4: Simulation results for case I. Three different load emulations with varying inertia.

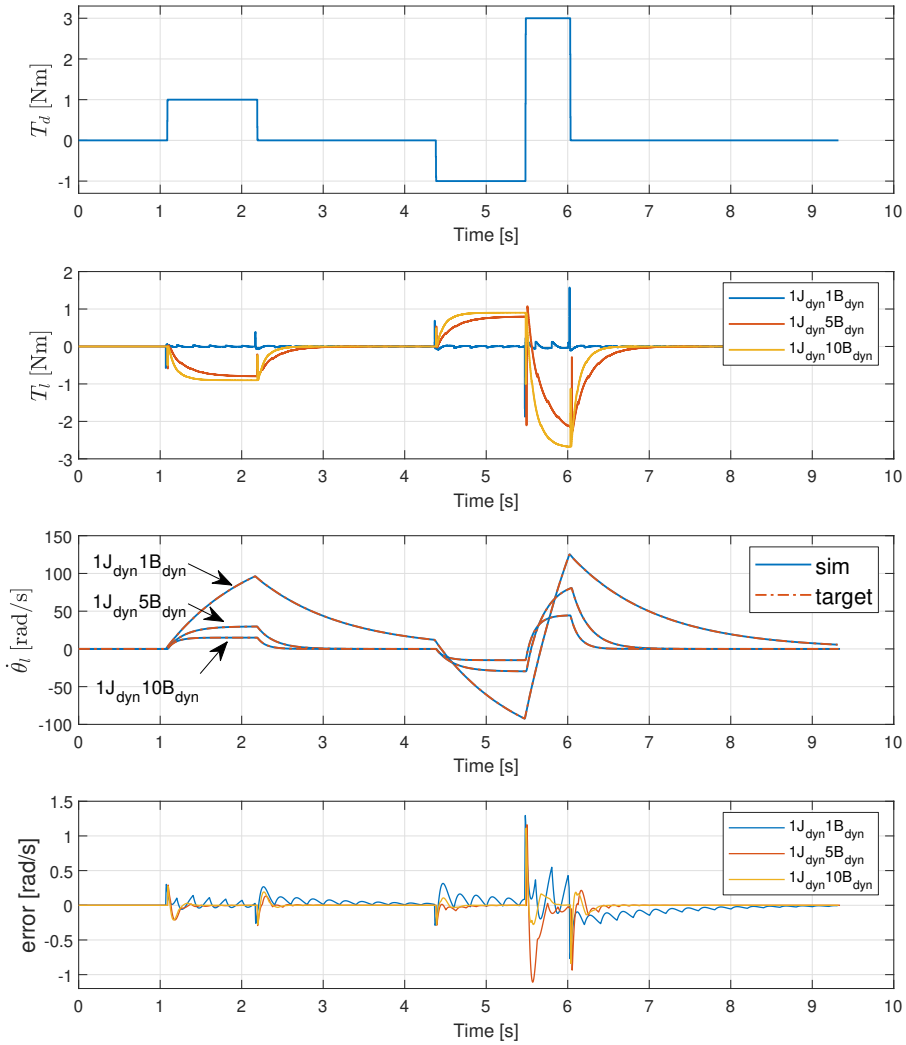


Figure 4.5: Simulation results for case I. Three different load emulations with varying friction.

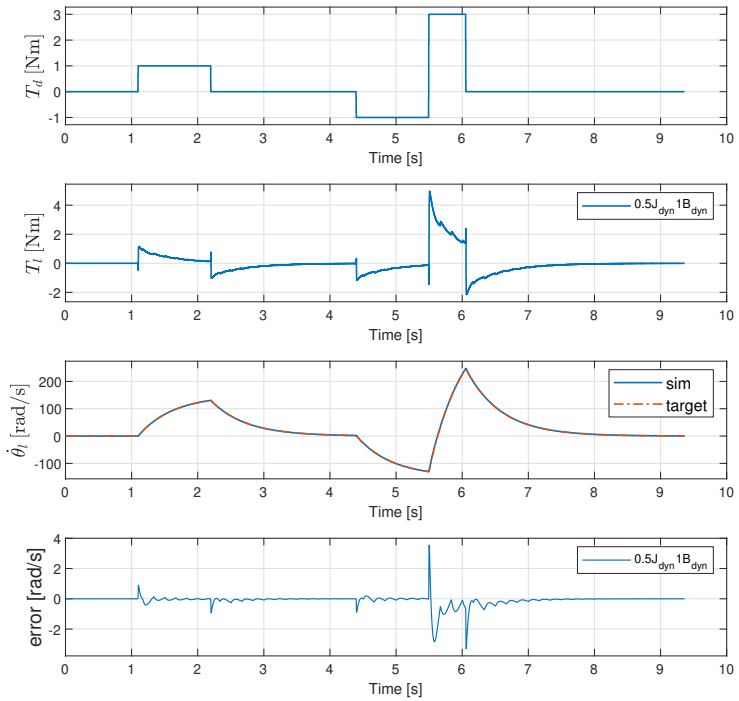


Figure 4.6: Simulation results for case I. Emulated load is set to $0.5\mathrm{J}_{\mathrm{dyn}}1\mathrm{B}_{\mathrm{dyn}}$.

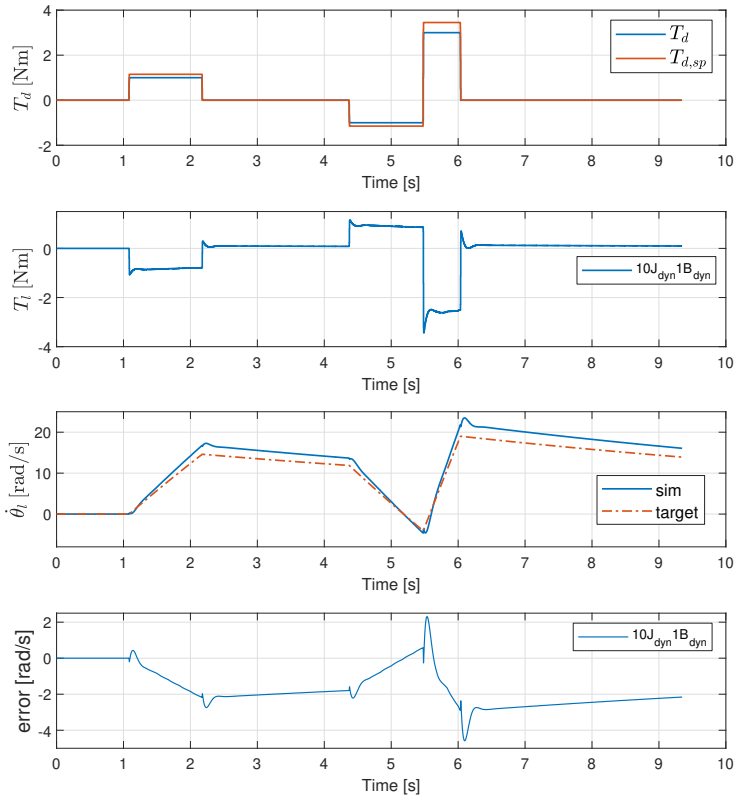


Figure 4.7: Simulation results for case I with an emulator input torque error of +15%. Emulated load is set to $10\mathrm{J}_{\mathrm{dyn}}1\mathrm{B}_{\mathrm{dyn}}$.

4.2.1.2 Experimental results

Figure 4.8-4.11 show experimental results for the emulation algorithm implemented on the dynamometer system, each with an emulated load of $5J_{dyn}1B_{dyn}$, $10J_{dyn}1B_{dyn}$, $1J_{dyn}10B_{dyn}$ and $0.5J_{dyn}1B_{dyn}$ respectively. Simulation results for these load cases are presented as well for comparison. On the physical system, the DUT is given set-point torques according to $T_{d,sp}$ in the figures, which is also the input torque to the emulator in this case.

These results show that the real system does manage to follow the given target speed trajectory. In figure 4.9, the largest errors occur at the input transients. At t=1.1 s, we see that the measured velocity lags. This is likely due to stiction friction in the dynamometer, which occurs when we go from $\dot{\theta}_l = 0$ to $|\dot{\theta}_l| > 0$. This effect is, of course, not included in the linear dynamometer model, and the emulator has no way to compensate for it.

The other magnification in figure 4.9, at t=5.5 s, shows that the measured velocity from the physical system exhibits a larger error to the target velocity than the simulated velocity. This effect is also highlighted for the simulated system in figure 4.4, explaining that the emulator is slow to react due to the input delay on T_l. From the measured velocity in figure 4.9, however, it would suggest that the input delay on T_l is larger on the physical system than on the simulated system since it experiences larger oscillations. In fact, increasing the input delay on T_l in the simulation model produces results that are more similar to the measured results. This can be seen in figure 4.12, where the input delay is increased from 1 sample to 4 samples.

The results also show the output torque T_l from simulations and the physical system. As can be seen from the results, there are some differences between the simulated output torque and the output torque on the physical system. This is likely caused by a model mismatch in the linear dynamometer model. Another equally likely explanation is that the exact shape of the output torque from the DUT, T_d, is not explicitly known. The input to the simulation dynamometer model is based on the set-point value of $T_{d,sp}$. If this is not equal to the actual output torque of the DUT then the output T_l of the physical emulator will be different since it has to compensate for either a larger or a lower actual torque output T_d.

Another important note is that the target velocity trajectory, which acts as our benchmark, is the target velocity for the set-point torque $T_{d,sp}$. As mentioned, the set-point torque can deviate from the actual DUT torque output on the physical system. Thus, in this case, we cannot be sure that the emulation produces the correct trajectory for the *actual* output torque but that it does so for the given set-point torque.

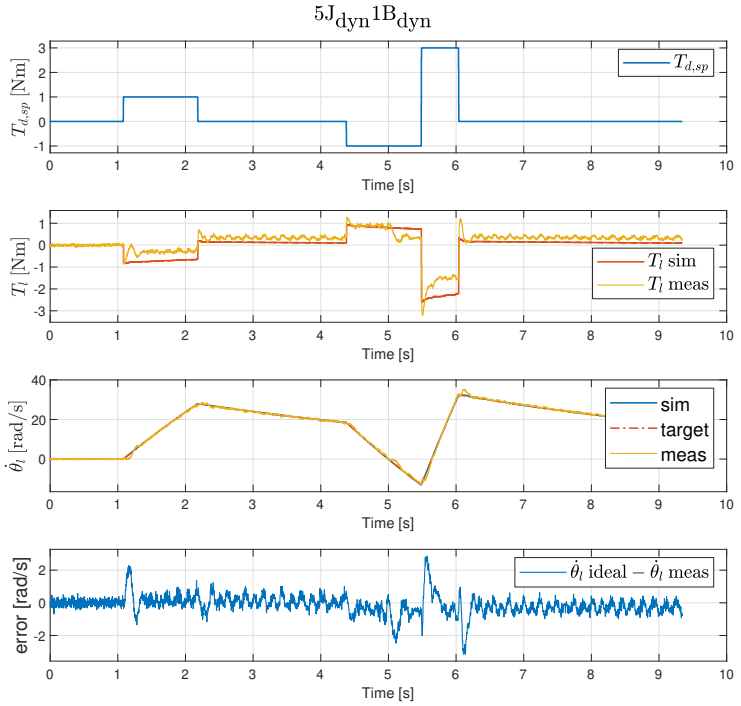


Figure 4.8: Experimental results for case I. Emulated load is set to $5J_{dyn}1B_{dyn}$.

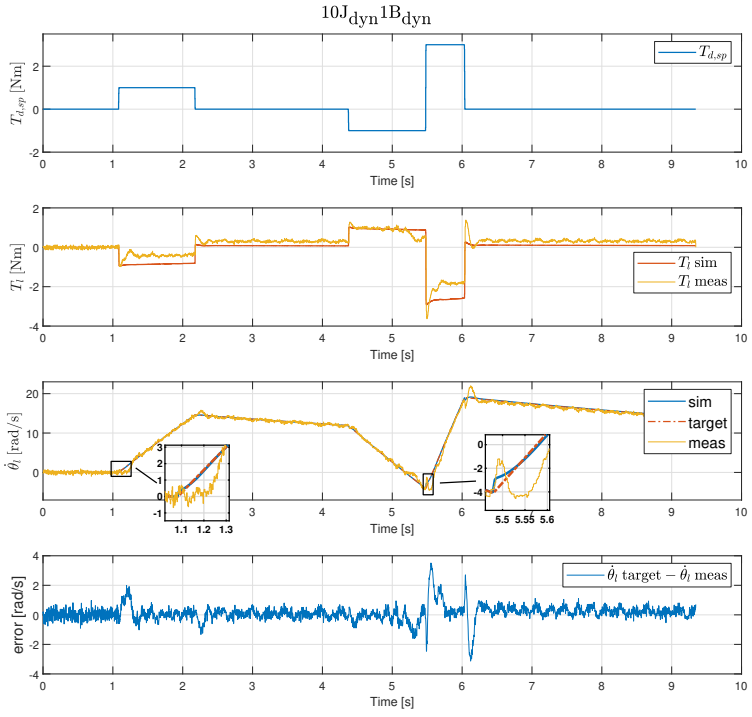


Figure 4.9: Experimental results for case I. Emulated load is set to $10J_{dyn}1B_{dyn}$.

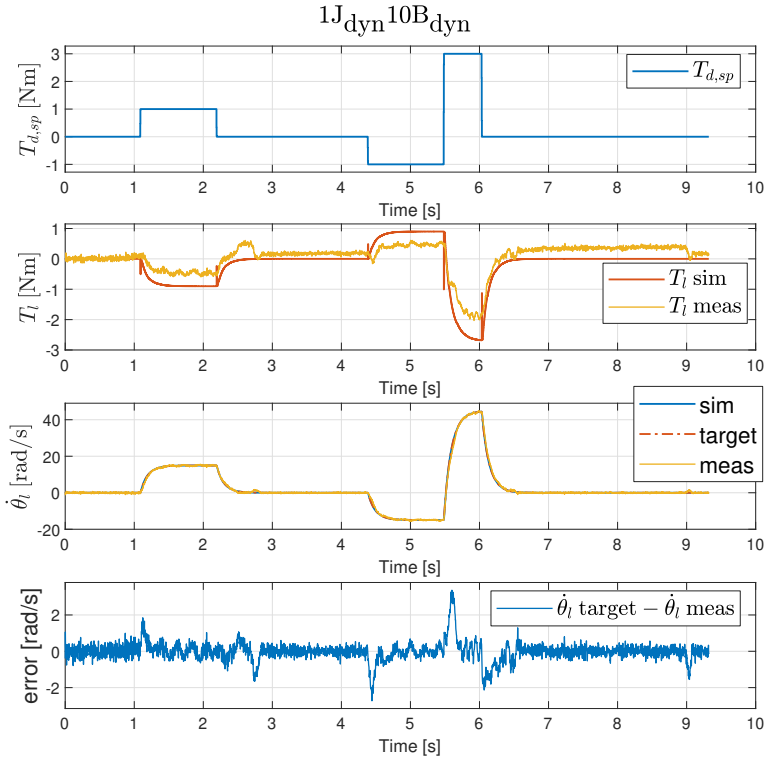


Figure 4.10: Experimental results for case I. Emulated load is set to $1J_{dyn}10B_{dyn}$.

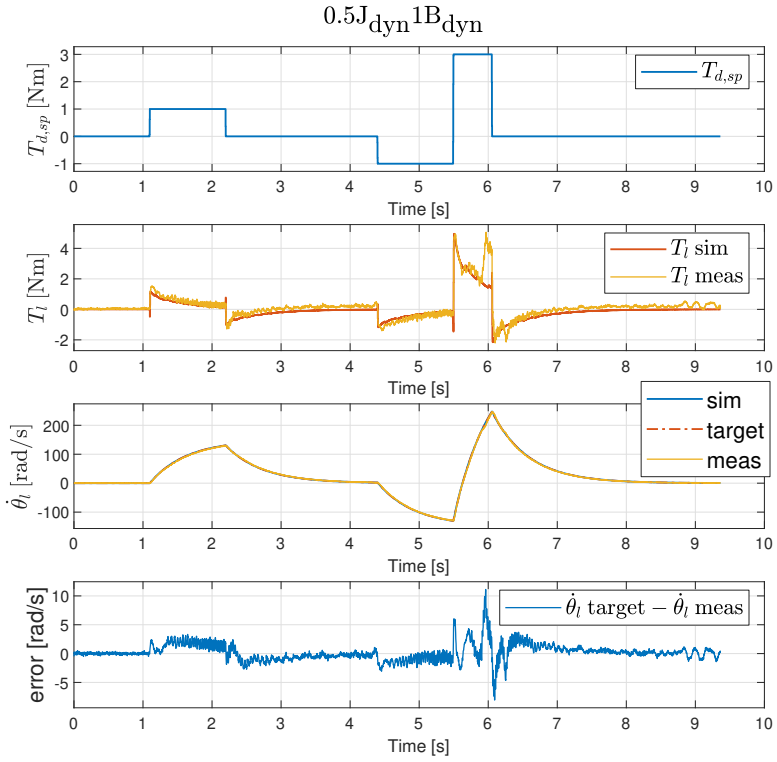


Figure 4.11: Experimental results for case I. Emulated load is set to $0.5J_{dyn}1B_{dyn}$.

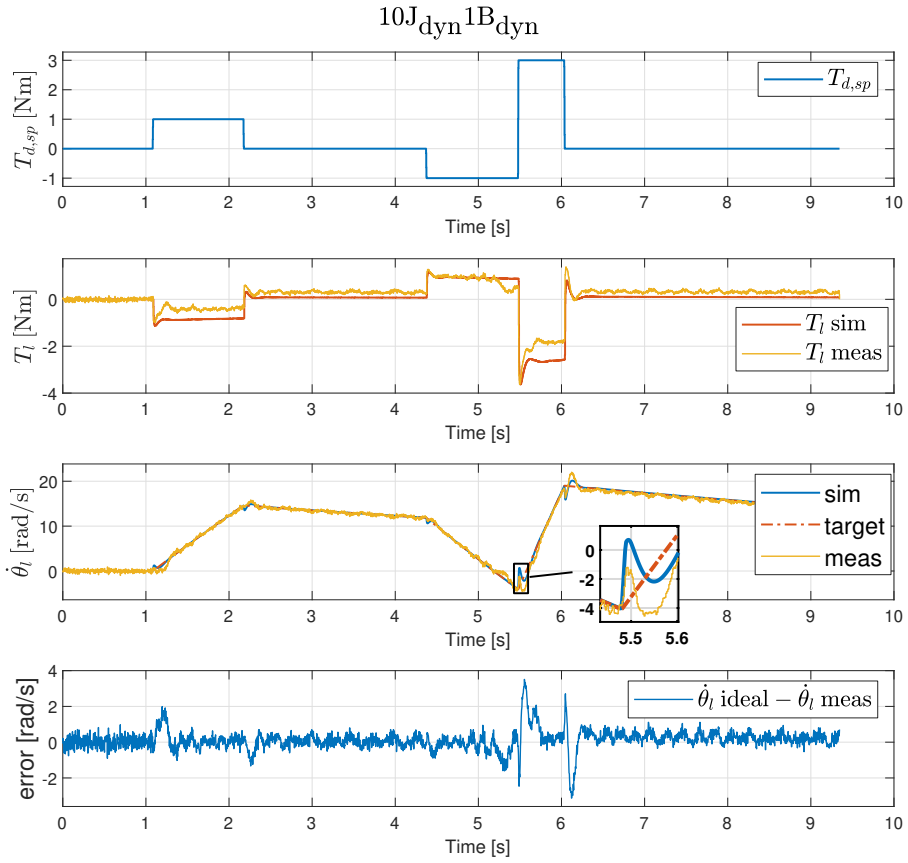


Figure 4.12: Experimental results for case I. Emulated load is set to $10J_{dyn}1B_{dyn}$. Input delay for T_l is increased from 1 sample to 4 samples.

4.2.2 Case II - Measured torque without compensation

The following section presents simulation and experimental results from the implementation presented in section 3.4.2. For simulations and the physical system, the speed controller $G_t(z)$ is tuned with a crossover frequency of $w_c = 30$ rad/s.

4.2.2.1 Simulation results

The simulation model in figure 3.17 is implemented and simulated in Simulink. As for the previous case, the simulated DUT output torque T_d consists of a sequence of pulses.

Figure 4.13, 4.14 and 4.15 shows results for three different load cases. As before, the resulting output angular velocity is compared to the target velocity, and the plots also show the difference between these two. The estimated input reference torque $T_{d,est}$ is also presented as the result of (3.34).

As can be seen from the simulation results, the output velocity exhibits significant oscillations at input transients. This is especially evident for the higher inertia load $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$. The frequency of these oscillations is measured to be approximately $f_{osc} \approx 10$ Hz, which corresponds well with the resonance peak observed in the frequency responses in section 3.4.2. Also worth noting is that emulating low inertia but high friction does not generate any oscillations, as expected, since adding friction to the emulation does not affect the overall system dynamics as seen in 3.4.2.

It can also be seen in figures 4.13-4.15 that the simulated output velocity has some error compared to the target velocity. This is due to the fact that the inertia and friction terms of (3.33) is omitted. The resulting $T_{d,est}$ will thus be slightly lower than the simulated output DUT torque T_d, which is what the target velocity is based on.

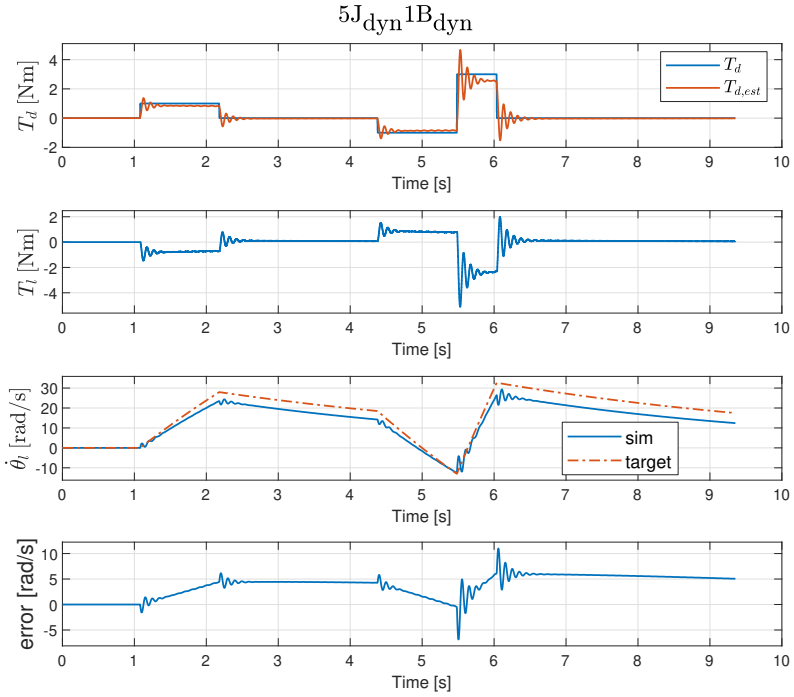


Figure 4.13: Simulation results for case II. Emulated load is set to $5J_{dyn}1B_{dyn}$.

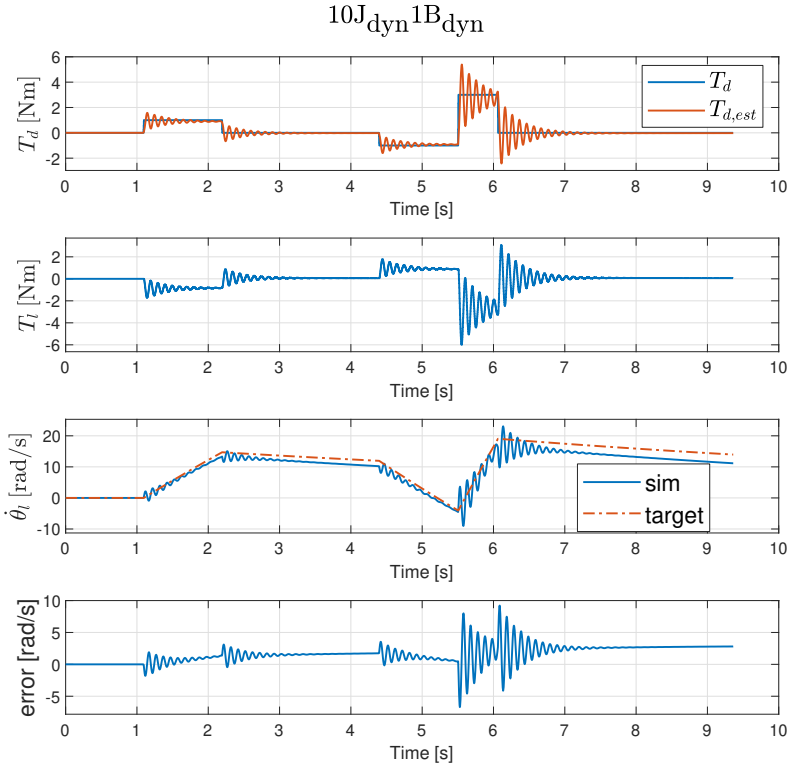


Figure 4.14: Simulation results for case II. Emulated load is set to $10J_{dyn}1B_{dyn}$.

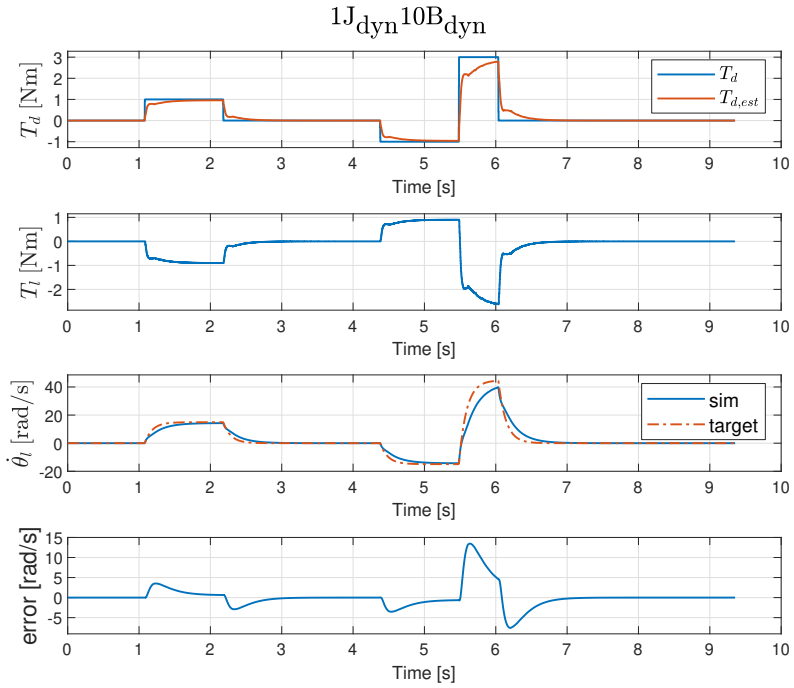


Figure 4.15: Simulation results for case II. Emulated load is set to $^{1}J_{dyn}{}^{10}B_{dyn}$.

4.2.2.2 Experimental results

Figure 4.16 and 4.17 show experimental results for the emulation algorithm implemented on the dynamometer system, each with an emulated load of $1J_{dyn}1B_{dyn}$ and $5J_{dyn}1B_{dyn}$ respectively. Simulation results for these load cases are presented as well for comparison. On the physical system, the DUT is given set-point torques according to $T_{d,sp}$ in the figures, and the input torque to the emulator is $T_{d,est}$ according to (3.34). The results also show the output from the torque sensor T_s, both from simulation and from the physical system.

In the results, the target velocity is no longer based on the set-point torque, $T_{d,sp}$, as for the previous cases. Instead, it is based on the input torque to the emulator $T_{d,est}$, to get a fair evaluation of how the physical emulator performs. Since we do not know the actual torque output from the DUT, comparing velocities based on this or the set-point torque will not yield any useful results. By having the same input $T_{d,est}$ to both G_{em} and the emulator itself, we measure how well the emulator tracks its target, which is the best performance indicator we can achieve from the circumstances. This is applied to both simulation data and experimental data in the results that follow.

As can be seen from figure 4.16, the physical emulator seems to handle the load case $1J_{dyn}1B_{dyn}$ without any issues. However, looking at the results for the other emulated load case $5J_{dyn}1B_{dyn}$, it is evident that the physical system exhibits major issues with stability. The output velocity starts oscillating at the very first input transient and persists for the rest of the sequence run time.

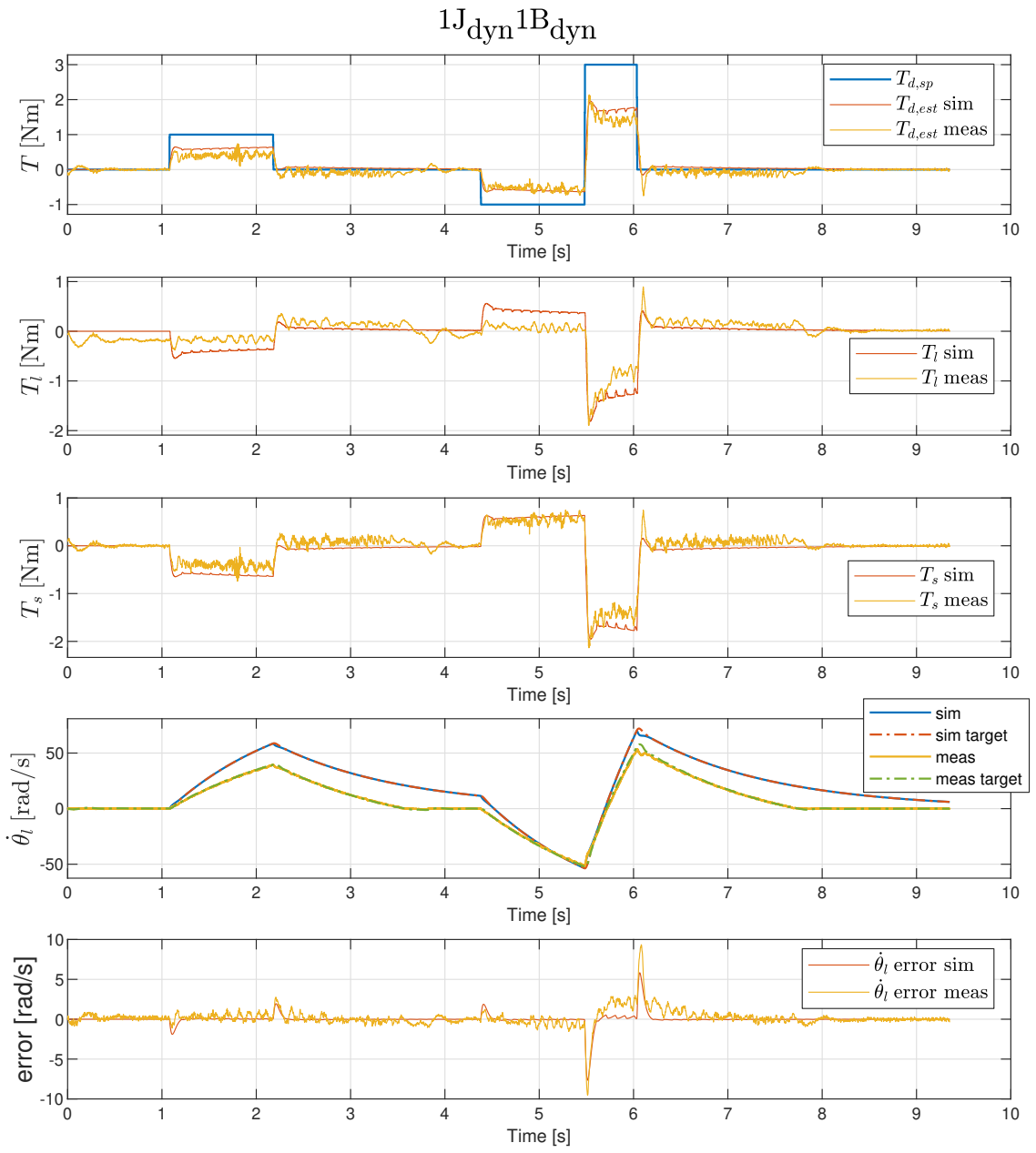


Figure 4.16: Experimental results for case II. Emulated load is set to $1J_{dyn}1B_{dyn}$.

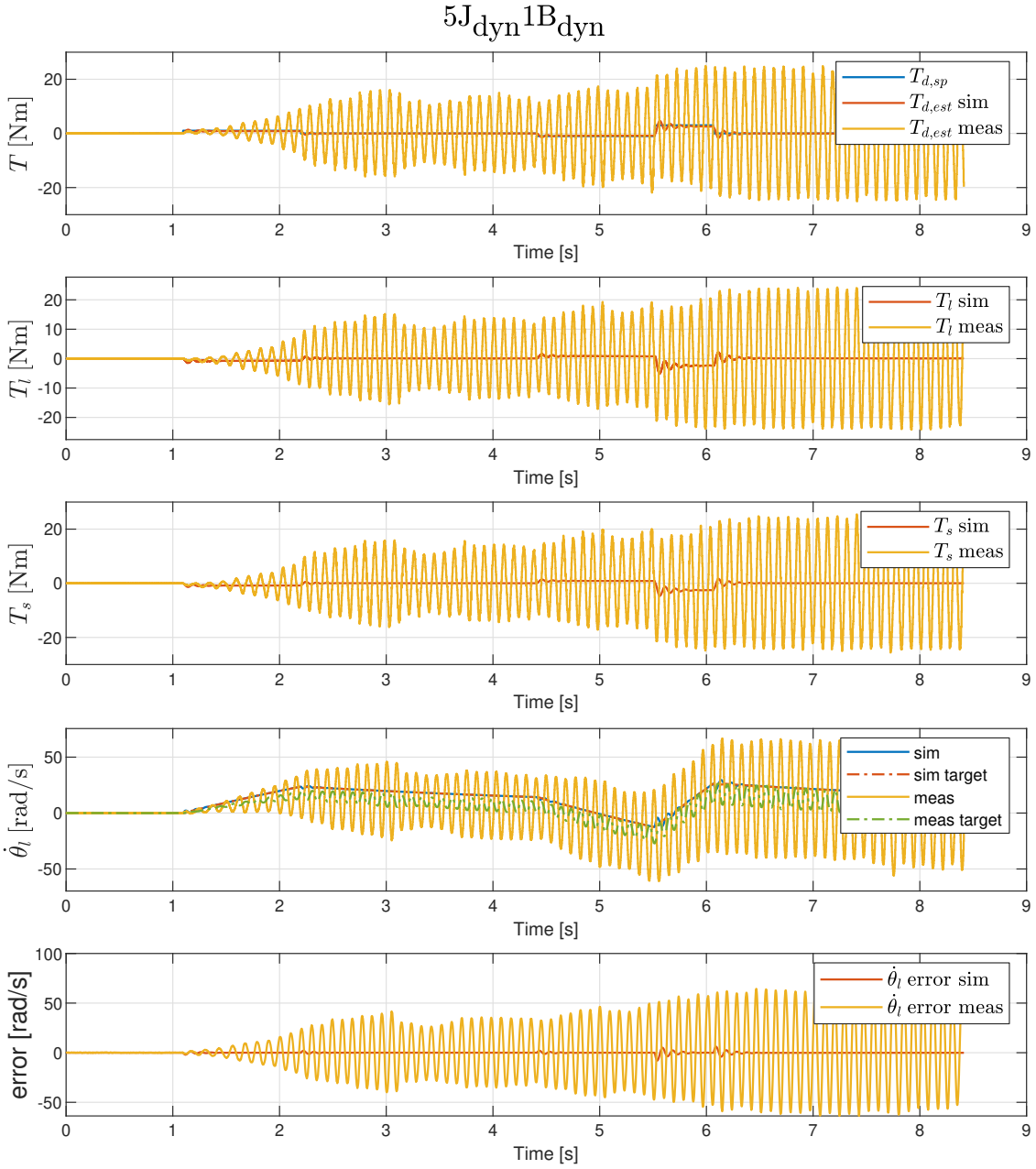


Figure 4.17: Experimental results for case II. Emulated load is set to $5J_{dyn}1B_{dyn}$.

4.2.3 Case III - Measured torque with compensation

The following section presents simulation and experimental results from the implementation presented in section 3.4.3. For simulations and the physical system, the speed controller $G_t(z)$ is tuned with a crossover frequency of $w_c = 30$ rad/s. The cut-off frequency for the derivative filter in 3.36 is set to $f_c = 20$ Hz unless otherwise noted.

4.2.3.1 Simulation results

The simulation model in figure 3.22 is implemented and simulated in Simulink. As for the previous cases, the simulated DUT output torque T_d consists of a sequence of pulses. The resulting output angular velocity is again compared to the target velocity, and the estimated input reference torque, $T_{d,est}$, based on (3.35) is also presented.

Figures 4.18-4.20 display the simulation results for three different emulated load cases. From all three load cases, the estimated torque $T_{d,est}$ can be seen to correspond well to the actual DUT output torque T_d, as is shown in the open-loop simulations in 4.1.1. In the load cases $5J_{dyn}1B_{dyn}$ and $10J_{dyn}1B_{dyn}$, $T_{d,est}$ can be seen to exhibit some over- and undershoot at input transients. This is due to the filtered acceleration in the estimation and its inherent phase delay. To further highlight this, figure 4.21 shows the results from two simulations where the cut-off frequency of the derivative filter is $f_c = 20$ Hz and $f_c = 200$ Hz. It is evident that the estimated torque $T_{d,est}$ with the higher cut-off frequency of $f_c = 200$ Hz has less of these over- and undershoots.

The simulated velocity from figure 4.19 has the largest errors at input transients. This is especially evident for emulated loads with larger inertia and when the input torque changes from -1 Nm to 3 Nm, as highlighted in the figure. Similar results can be seen for case I in section 4.2.1.1, where the input delay of the load motor is assumed to be the cause. However, the resulting errors in figure 4.19 can be seen to be larger compared to the results from case I and the oscillations are also slightly more evident. This is very likely due to the additional delay from the torque estimation. The torque output from the DUT, T_d, is applied almost instantly to the dynamometer plant and causes the velocity to deviate before the emulator can react since the input estimated torque $T_{d,est}$ lags behind. The origins of this delay are partly from the derivative filter and the integrated low-pass filter on the torque sensor itself.

Finally, figure 4.22 shows the implication of an input torque set-point error, similar to the simulation done for case I in figure 4.7. Here, one can clearly see the benefit of utilizing the torque sensor, with compensation, over simply relying on the set-point torque as input to the emulator, as in case I.

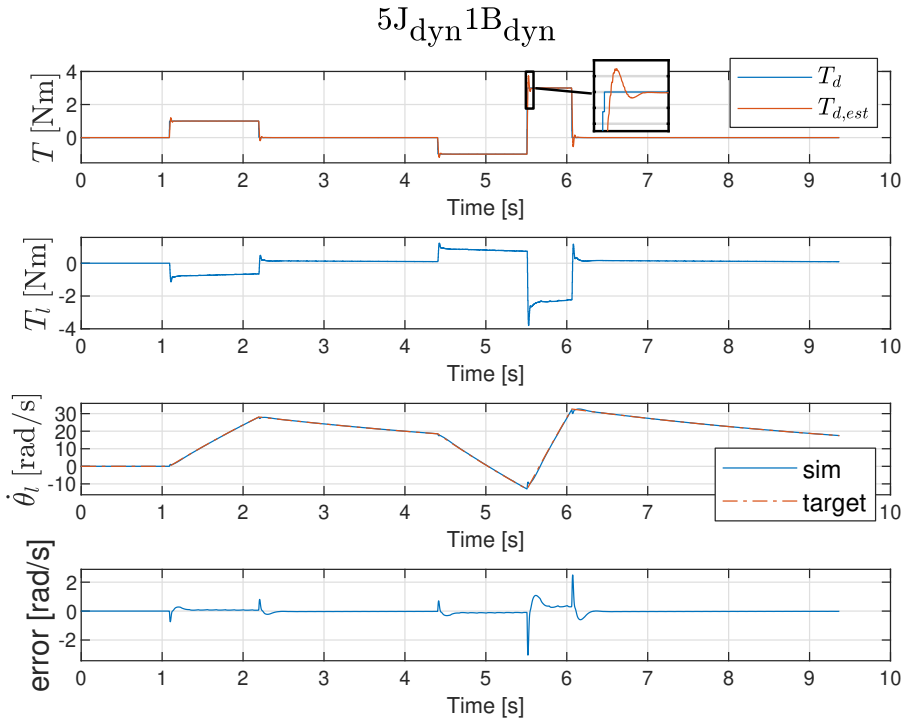


Figure 4.18: Simulation results for case III. Emulated load is set to $5J_{dyn}1B_{dyn}$.

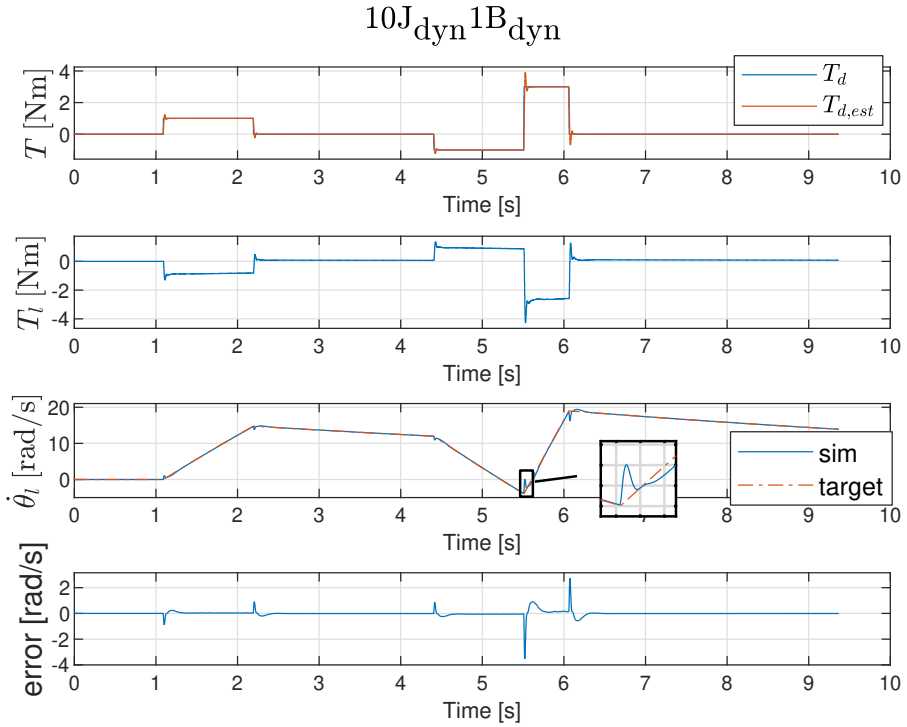


Figure 4.19: Simulation results for case III. Emulated load is set to $10J_{dyn}1B_{dyn}$.

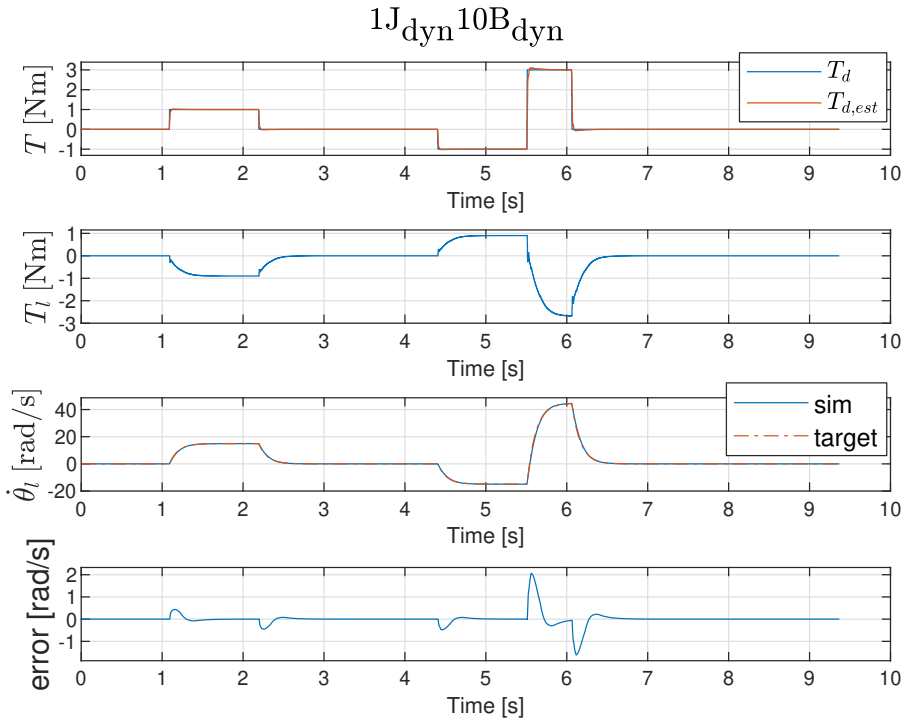


Figure 4.20: Simulation results for case III. Emulated load is set to $1J_{dyn}10B_{dyn}$.

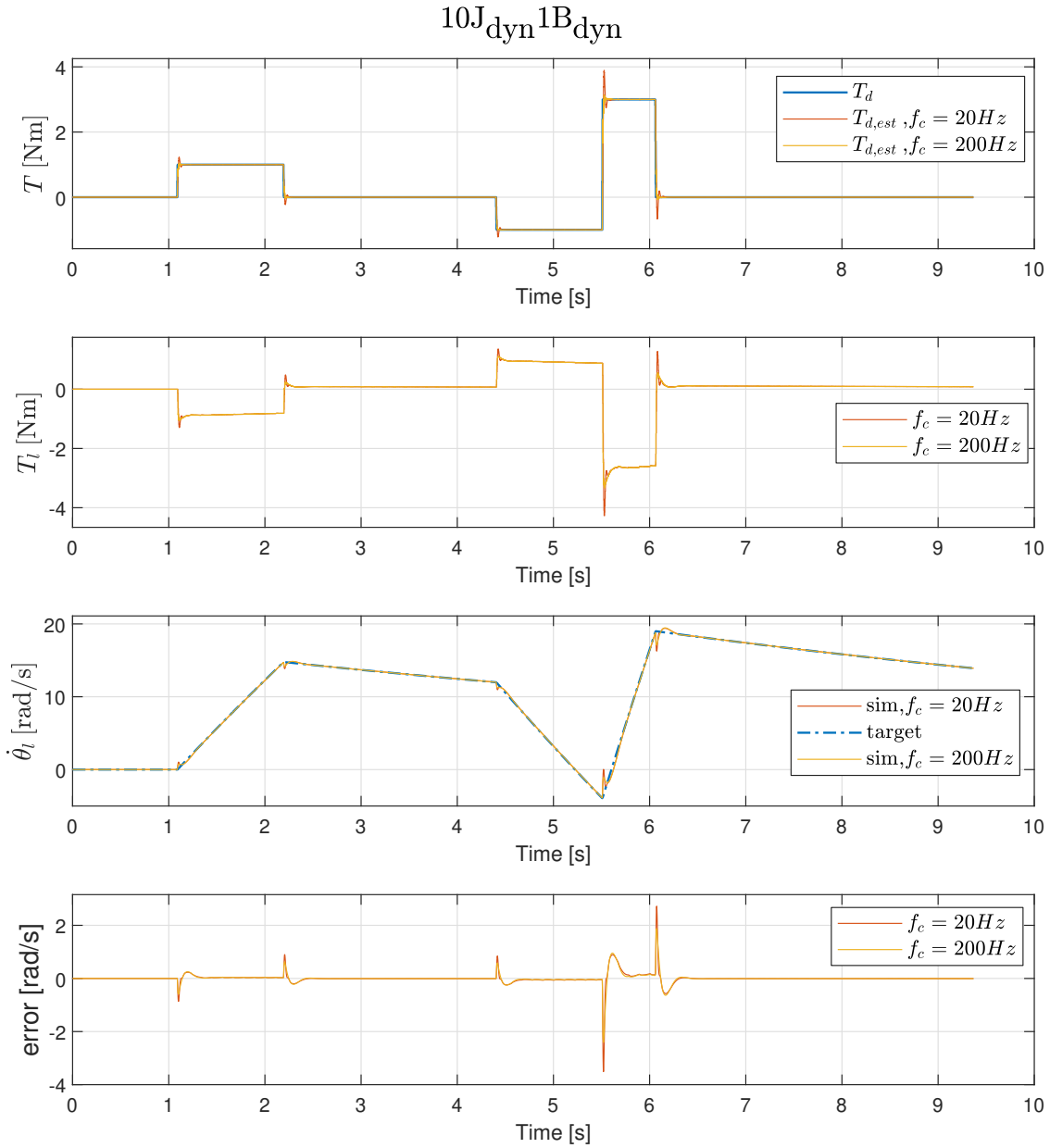


Figure 4.21: Simulation results for case III. Different values for the cut-off frequency of the derivative filter. Emulated load is set to $10J_{dyn}1B_{dyn}$.

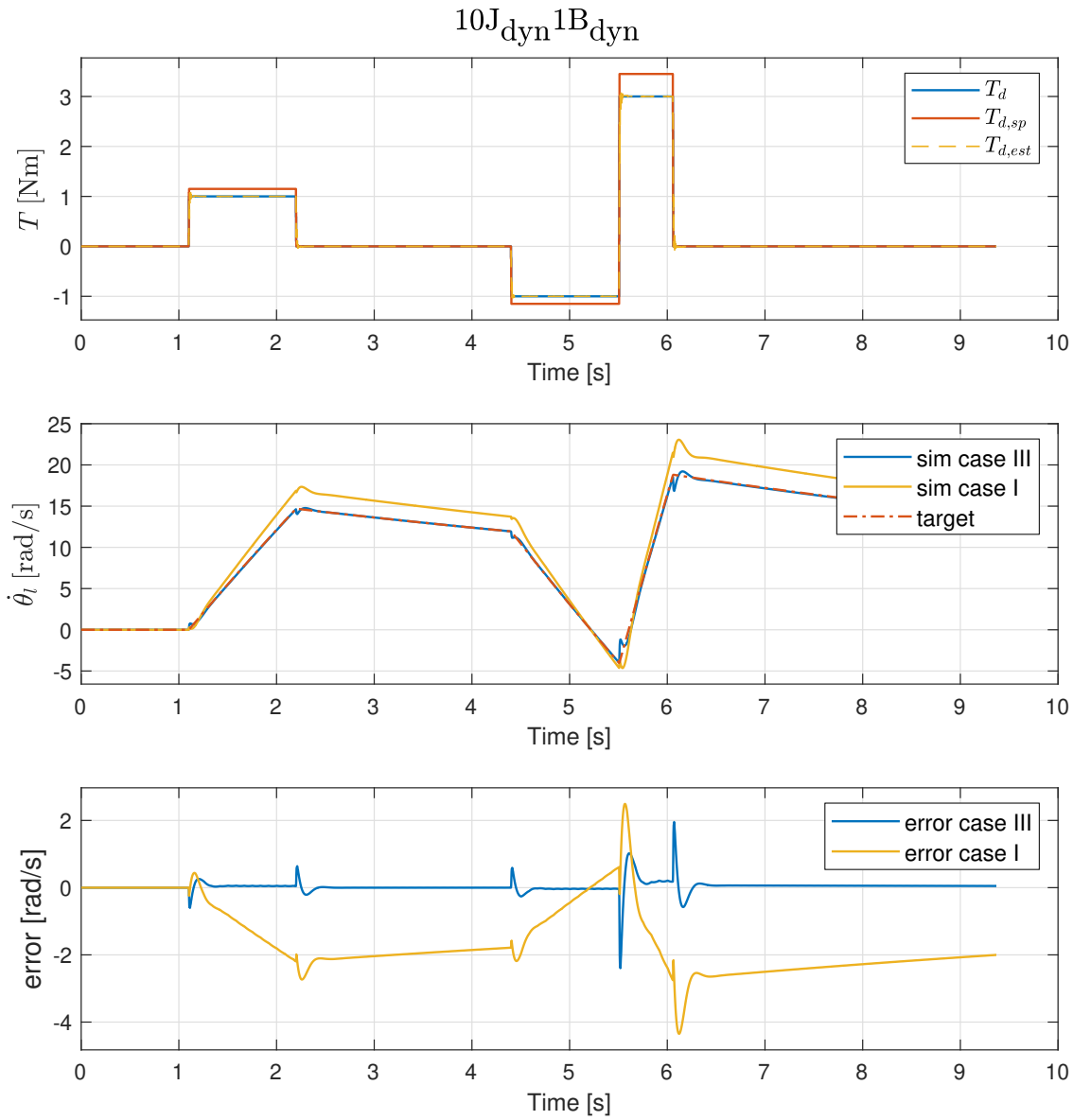


Figure 4.22: Simulation results for case III with a set-point input torque error of +15%, compared to case I simulated output. Emulated load is set to $10\mathrm{J}_{\mathrm{dyn}}1\mathrm{B}_{\mathrm{dyn}}$.

4.2.3.2 Experimental results

As for the experimental result in case II, section 4.2.2.2, the target velocity in the results that follows is based on the input torque to the emulator $T_{d,est}$ for both experimental and simulated data.

Figures 4.23 to 4.27 shows experimental results for the case where the input torque is the estimated torque according to (3.35). As before, simulation results for the different load cases are also presented for comparison.

From the results, we can observe that the measured output velocity does manage to follow its target velocity trajectory. The largest errors occur at the input transients, similar to the simulation results. This effect can be seen to increase as larger inertia is emulated, where the largest oscillations can be seen in figure 4.25 with the load case $15\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$. By comparing the simulation and measured errors, we see that the effect is larger in the physical system.

Similar to the analysis done for case I in section 4.2.1.2, an additional input delay on the physical system is thought to be the reason for this. Figure 4.28 shows the effect of increasing the simulated input delay from 1 sample to 4 samples for the load case $10\mathrm{J_{dyn}}1\mathrm{B_{dyn}}$. From the figure, we can clearly see that the simulation now corresponds better to the measured data. This further motivates the premise that the actual input delay to the load motor is larger than what is measured in section 3.2.3.

Figure 4.27 displays experimental results for emulating a load with lower inertia than the dynamometer. As can be seen, the emulator has no issues in following the target velocity.

An important note regarding the estimated torque for this case, $T_{d,est}$ from (3.35), is that it is based on the linear model in (3.1). As shown in section 3.2.1.3, this model does not capture the non-linear frictional dynamics. This is especially true for velocities close to zero where stiction friction is present [10]. This will have to be taken into consideration when analyzing the estimated DUT torque in the experimental measurements: the estimated torque is not necessarily equal to the actual DUT output torque. When the emulated load torque is high compared to the load contribution from the dynamometer dynamics, the estimation will likely be more accurate since the dynamometer dynamics have less contribution to the total load.

The mentioned uncertainty of $T_{d,est}$ in (3.35) is, however, not of large importance. The primary benefit of using this estimation is that load motor torque T_l is effectively compensated for, and the coupling effect that resulted in an unstable system in Case II is mostly gone.

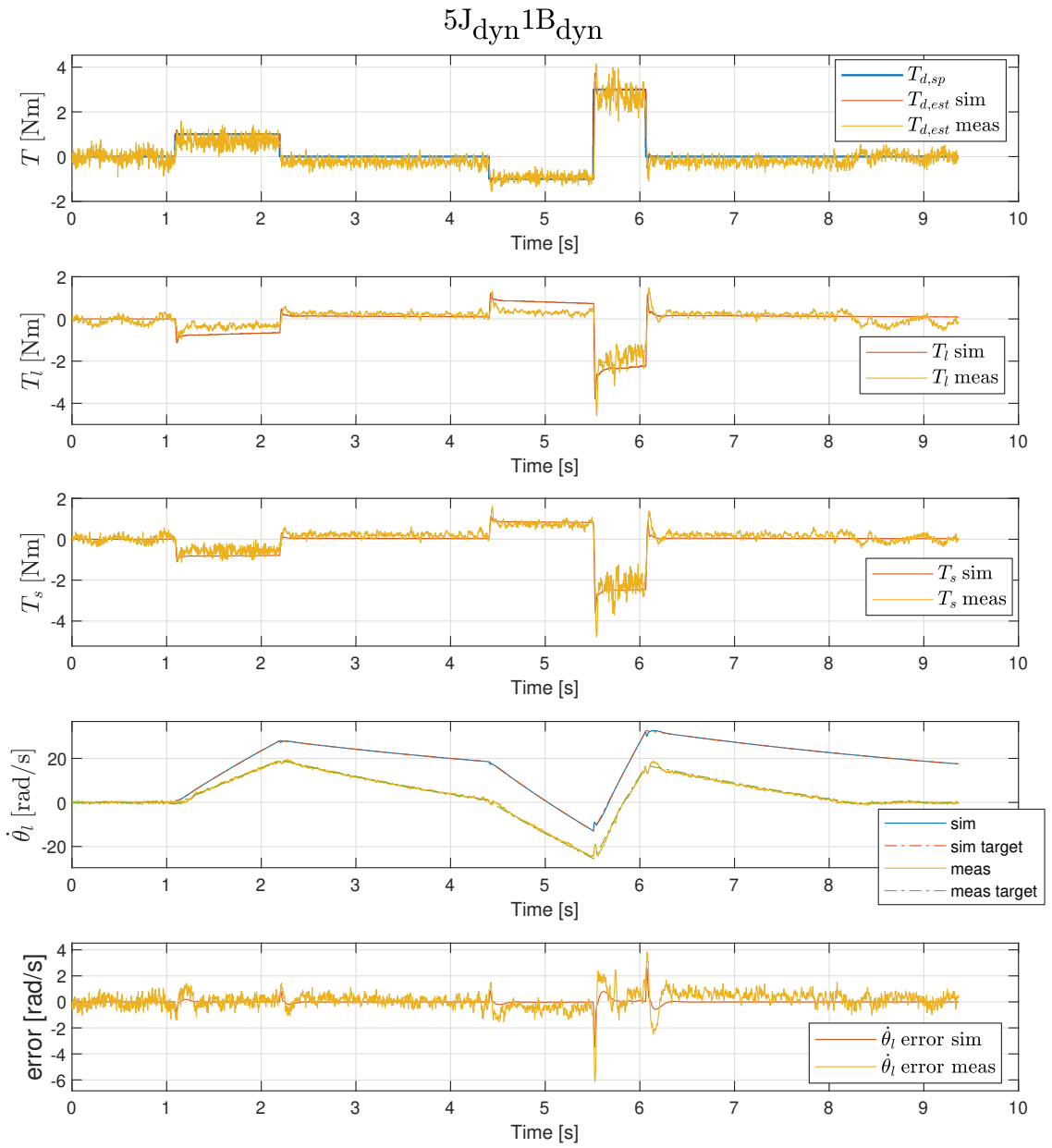


Figure 4.23: Experimental results for case III. Emulated load is set to $5J_{dyn}1B_{dyn}$.

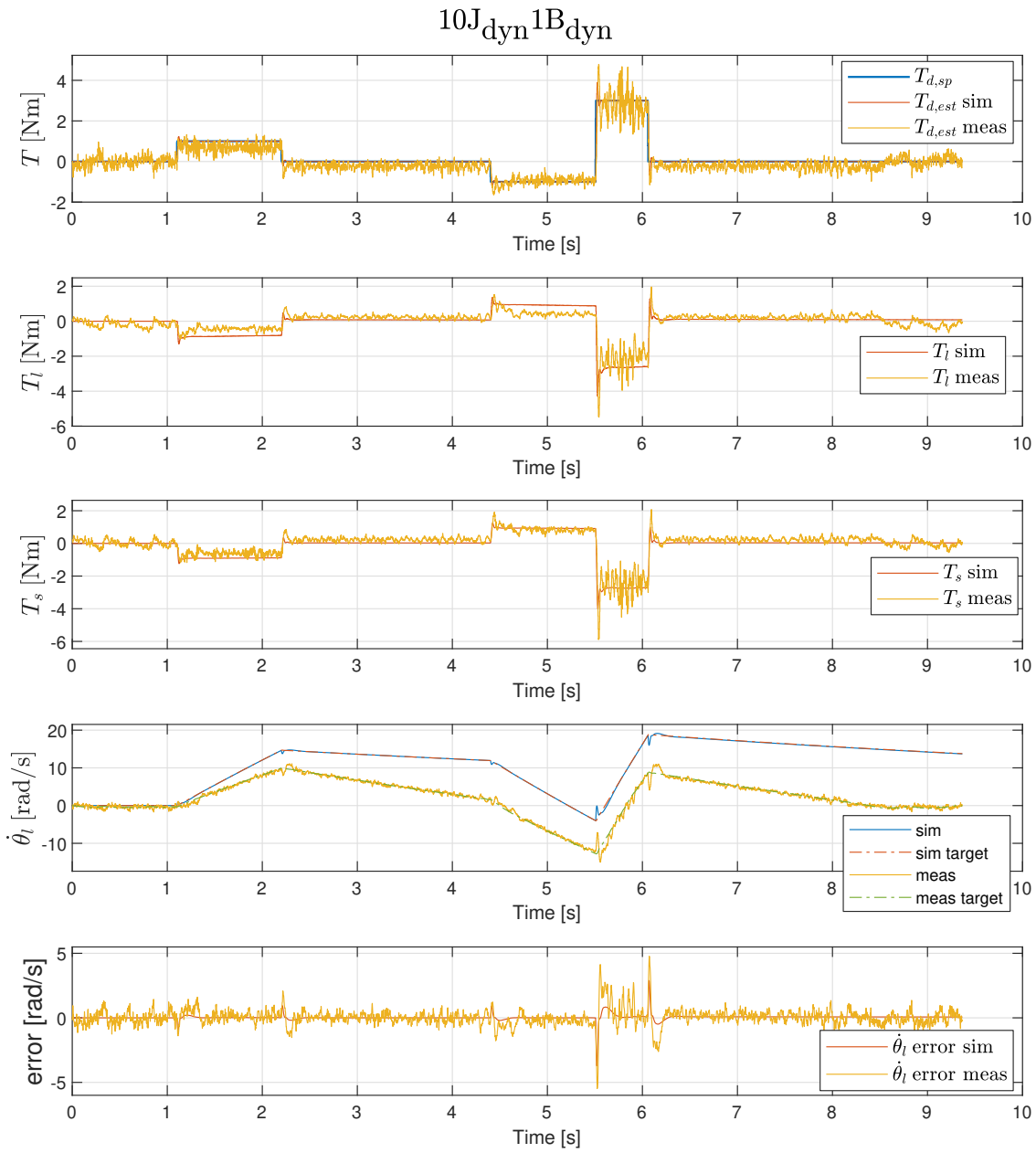


Figure 4.24: Experimental results for case III. Emulated load is set to $10J_{dyn}1B_{dyn}$.

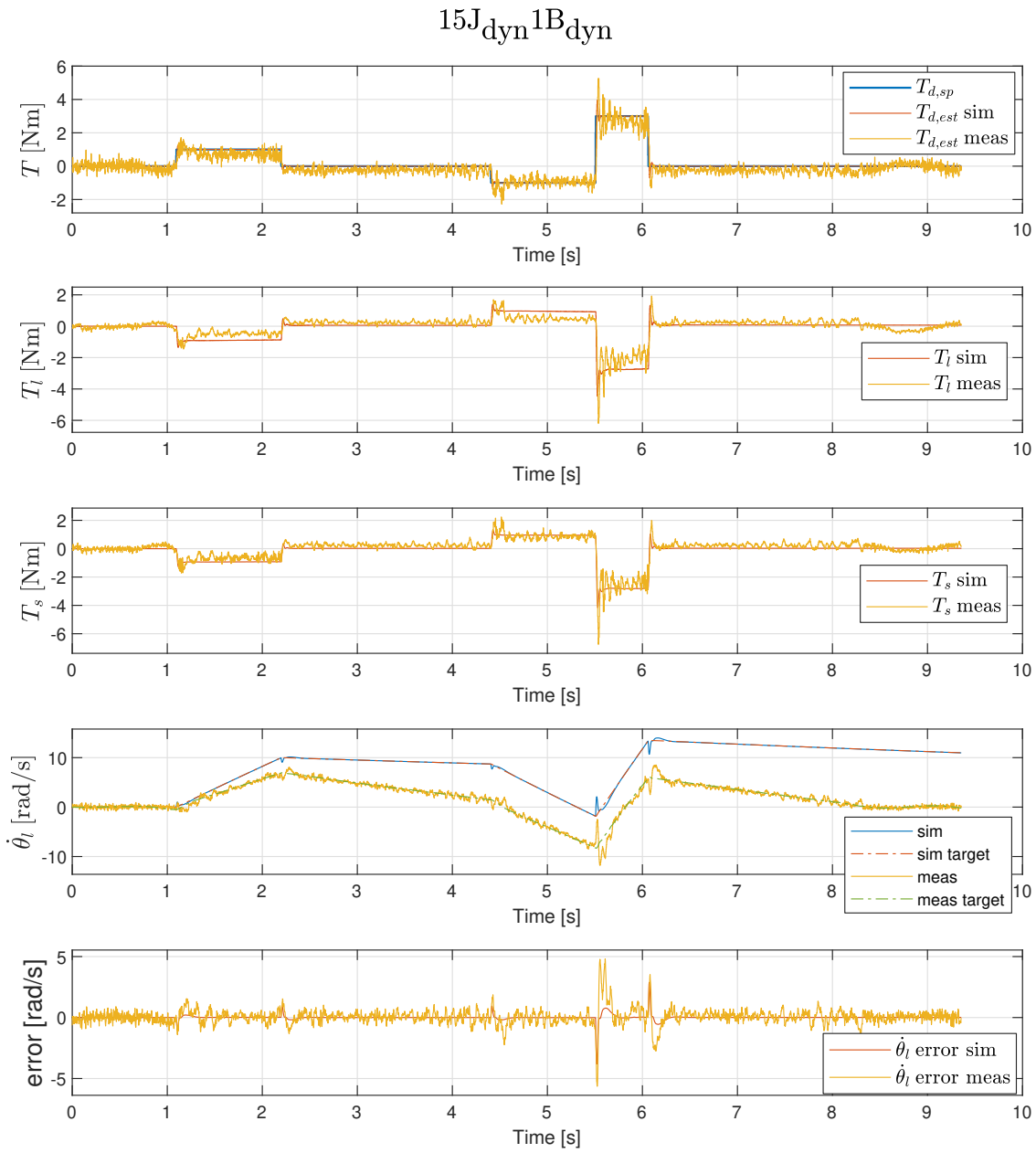


Figure 4.25: Experimental results for case III. Emulated load is set to 15J$_{dyn}$1B$_{dyn}$.

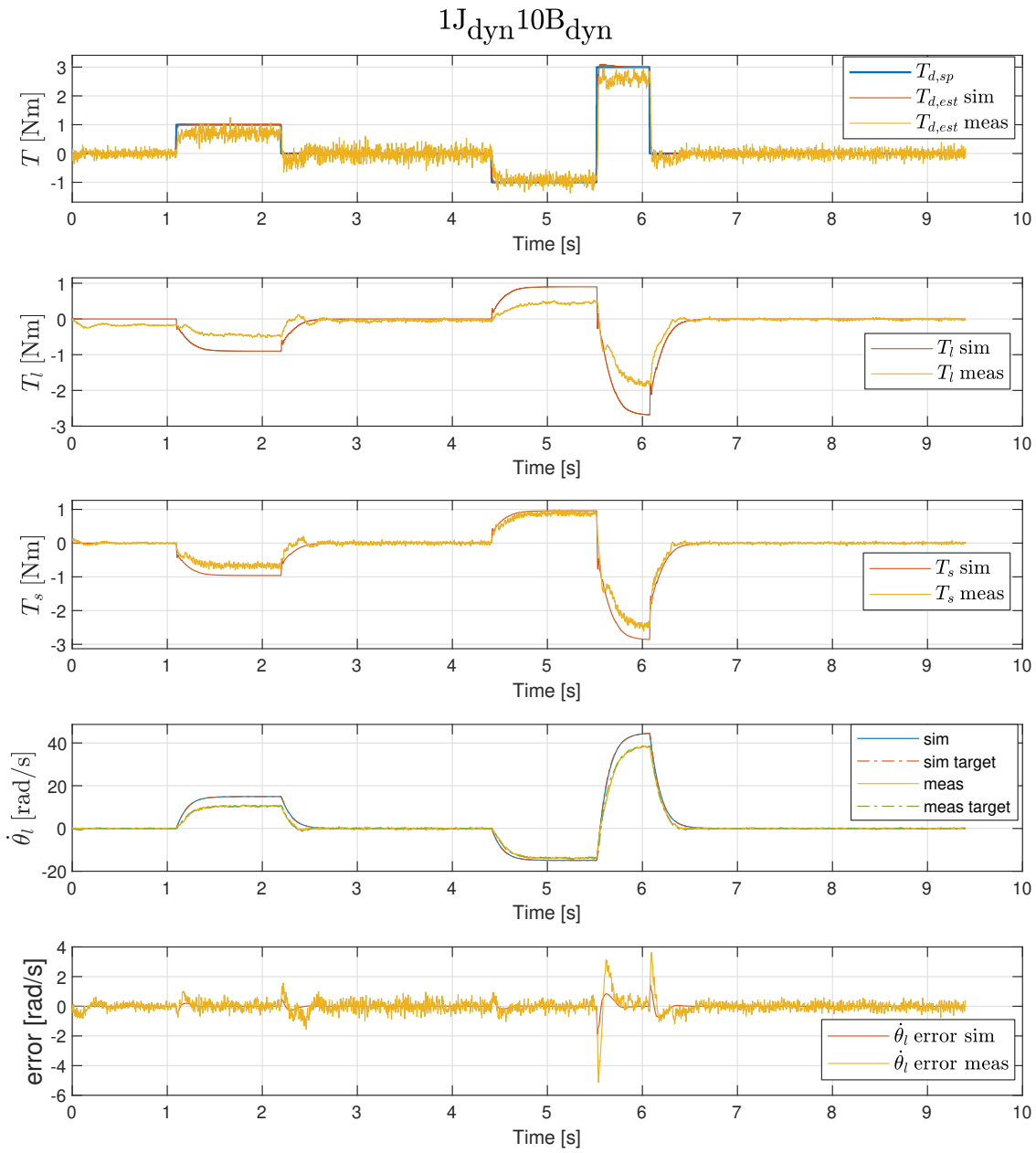


Figure 4.26: Experimental results for case III. Emulated load is set to $1J_{dyn}10B_{dyn}$.

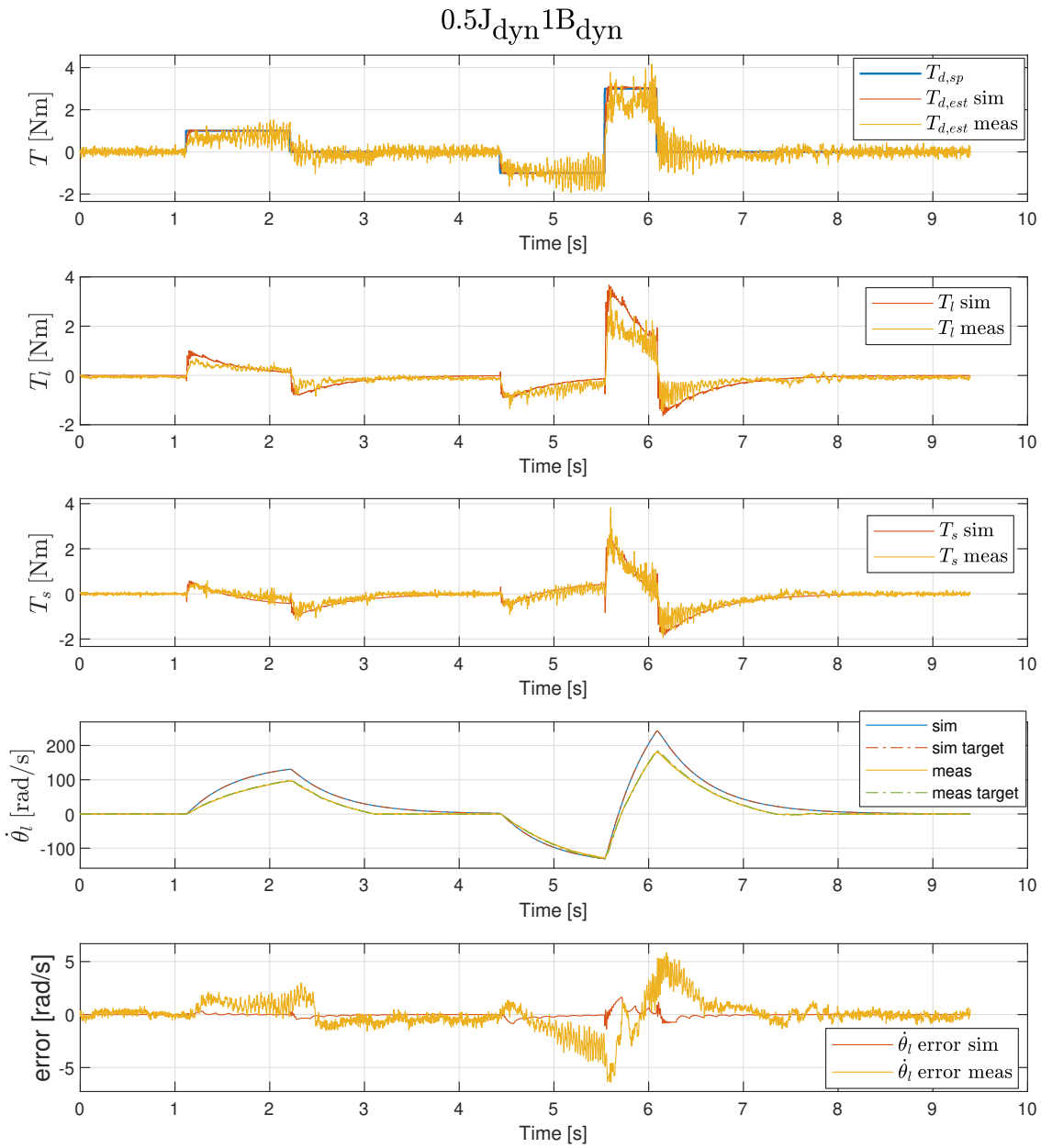


Figure 4.27: Experimental results for case III. Emulated load is set to $0.5J_{dyn}1B_{dyn}$.

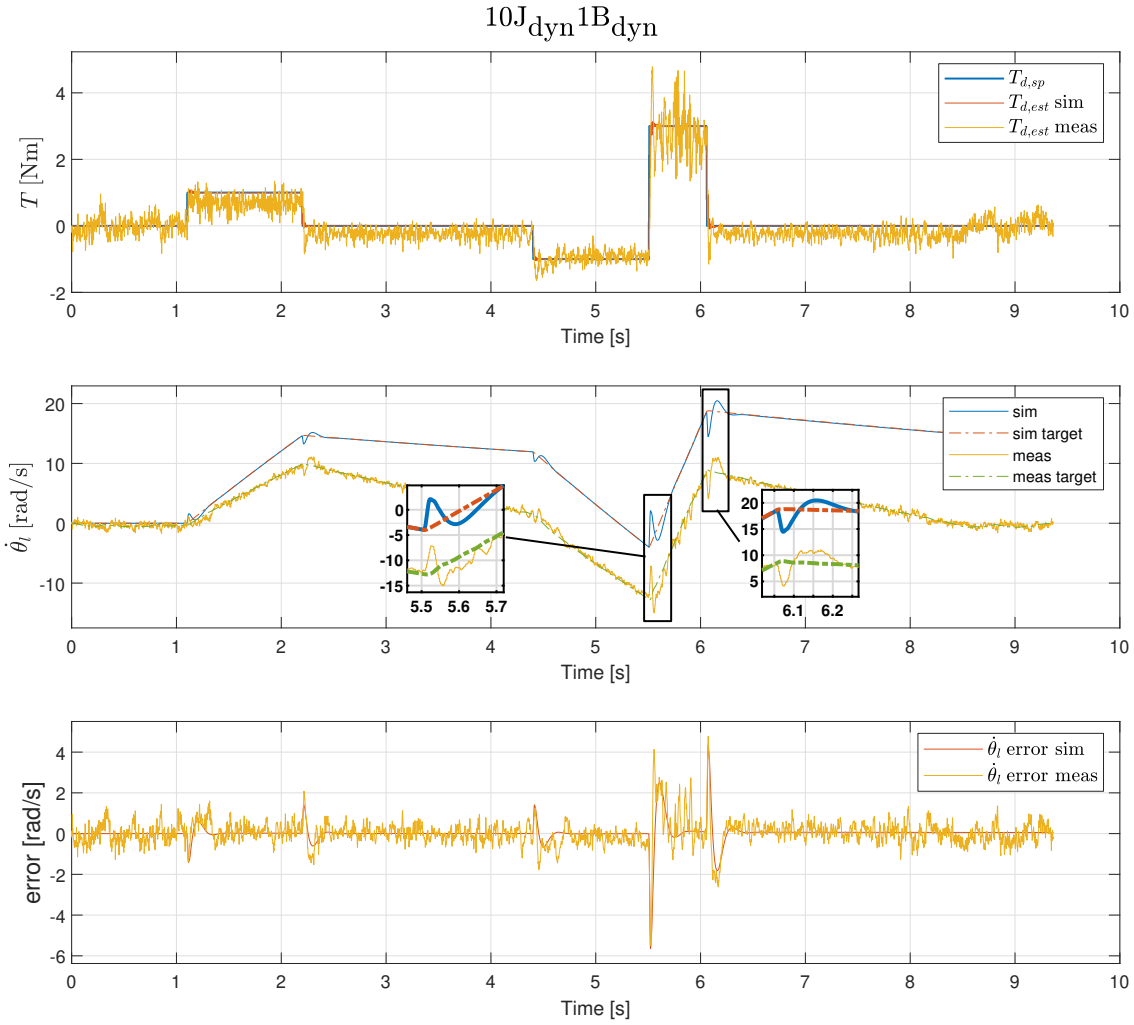


Figure 4.28: Experimental results for case III where the input delay to load motor is increased from 1 to 4 samples. Emulated load is set to $10J_{dyn}1B_{dyn}$.

5

Discussion

This chapter presents a discussion and analysis of the project outcome and results. It covers some of the most important qualities of the emulator performance and system model.

5.1 Emulator performance

5.1.1 System delay

The most noticeable source of errors can be observed at input transients where the output velocity oscillates. This is especially evident when emulating loads with moderate to high inertia. The effect can be seen for both case I and case III as seen in figures 4.9 and 4.24, but is larger for case III where the input torque to the emulator is estimated. The major cause of this behavior is likely to be due to the delay in the system. The input delay to the load motor is one part, but even without this delay, the system would still exhibit some oscillating behavior at input transients since there is an internal delay in the emulator itself. Most of this internal delay is believed to come from the estimation $T_{d,est}$ since the acceleration term in the estimation is filtered with a low-pass filter. In addition, the measured torque from the torque sensor T_s is also subject to a low-pass filter, as mentioned in section 3.2.2, causing additional delay.

Furthermore, the proposed emulation method has an intrinsic delay due to the fact that the tracking controller G_t is part of the compensator [13]. This could potentially be improved by investigating other emulation methods that do not have this component [11].

The load motor input delay, as explained in section 3.2.3, is also believed to be one of the major contributors to the total system delay. As is shown for both case I and case III in figures 4.12 and 4.28, an increased input delay from one to four samples resulted in a better match between the simulated and experimental data. This would motivate that the delay in the physical system is indeed higher than the measured delay in section 3.2.3. However, this measurement was only conducted once and it is likely that this delay could be varying. A further analysis could be made to determine the input delay more accurately.

5.1.2 DUT position and speed control mode

As one can see from the results, the only control mode used on the DUT in the experiments is torque control mode, where input set-points $T_{d,sp}$ were given. However, as can be seen in [5], which is the basis of the emulator used in this thesis, the developed emulator is extensively used in conjunction with a speed controller on the DUT. Using torque mode exclusively on the DUT for the experiments conducted for this thesis is done in order to simplify the testing and remove any instabilities that might occur from the cascaded speed- and torque controller in the DUT. The end goal of the emulator system is, however, to be able to operate the DUT in either position- or speed control mode since this is the majority of use cases for testing.

After verifying the performance of the emulator in DUT torque mode, speed controller mode was, in fact, tested for both case I and case III. The results were, however, unstable for any emulated loads with inertia higher than $1\mathrm{J}_{\mathrm{dyn}}$. There could be several reasons for this, one being that the speed controller on the DUT simply is not designed to handle the emulated load case that caused the instability. This analysis could easily be performed given the control structure and gains on the DUT, but it is beyond the scope of this thesis. Another likely explanation is the emulator system delay as explained in the previous section. Since both the DUT and the emulator are now working in a closed loop with the same tracking variable, shaft velocity, a delay in the emulator could easily cause the two closed-loop systems to become unstable.

5.1.3 DUT output torque estimation

Looking closer at experimental results for case III in figures 4.23-4.27, we can see that the estimated torque $T_{d,est}$ *meas* on the physical system tracks the set-point torque $T_{d,sp}$ quite well. However, it is clear that the estimation suffers from some noise effects. Partly from the torque sensor measurements T_s, as can be seen are not noise-free by any means, but also from the differentiation of the velocity. One interesting note from the measured torque estimations for all the load cases is that it tends to be slightly lower than the set-point torque $T_{d,sp}$. This is most evident in the figures at the first set-point torque step at t=1s and the third step at t=5.5s. This could indicate that the actual output torque of the DUT is slightly lower than the set-point torque during those pulses, which is not surprising since the output torque is known to deviate from the set-point torque, as explained earlier in this report.

It is, however, not possible to fully evaluate the accuracy of the estimated torque since we do not have a confident source of what the DUT output torque actually is. And since the estimation is based on the linear model for inertia and friction in (3.1), which is known to deviate from the physical system, some errors in the estimation is likely present. For further development of the estimator, some method of confirming against the actual DUT output torque would be valuable.

5.2 Dynamometer model

As is shown for the inertia and friction parameter validation in section 3.2.1.3, the linear dynamometer model does have poor accuracy in capturing the velocity-dependent friction in the physical system. This is not surprising since a rotational system of this kind, with large bearings and couplings to handle high forces, will likely exhibit non-linear friction. This would be most evident at low velocities where static and Stribeck friction is known to be a large factor [10].

The performance of the emulation method that is presented in this thesis does rely on the dynamometer dynamics, since it is part of the compensator system in (2.11). This model is limited to a first-order linear system model, thus a non-linear model is not feasible to use. One could argue that the performance would suffer if the model differs too much from the physical system. This effect is indeed observed in figure 4.9 at the first input transient, where static friction in the physical system results in a lag behavior of the emulator. However, since the emulation method used is a closed-loop system, some of the model errors will likely be handled by the speed controller G_t.

A possible improvement would be to evaluate an emulation method with non-linear control algorithms like the one presented in [11]. In fact, the authors in this article specifically mention non-linear friction compensation as a major motivator for the method. The non-linear friction model presented in appendix A.3 could be a good starting point for this.

6

Conclusion

This thesis investigates a feedforward-tracking emulation principle on an existing dynamometer system. More specifically, it examines the use of measured torque from a torque sensor as input to the emulation system. The thesis also presents a linear model of the dynamometer system, where a system identification approach is performed to determine the model parameters for inertia and friction. Furthermore, experimental measurements on sensor- and input delays are presented to enhance the system model.

It is shown that the resulting emulator can emulate linear loads with varying inertia and friction using a model-based estimation of DUT output torque based on measured torque from the torque sensor and shaft velocity. The largest errors could be observed for moderate to high inertia emulations at DUT set-point torque transients, where the resulting shaft velocity exhibits oscillations around the target velocity. The major cause is the system's delays.

6.1 Future work

There are multiple ways to enhance the performance and further develop the emulator presented in this thesis. One of the most evident issues is that the system proved unstable when running the DUT in speed control mode. This is unfortunate since most test cases for the DUT involve either position or speed control. Therefore, further development of the emulator proposed in this project should concentrate on resolving this issue, which is believed to occur due to system delays.

Analyzing the input delay to the load motor to resolve the delays in the system could be a first step. A solution to this might involve further analysis of the different configurations in the b maXX BM4400 driver and investigating if there are any inherent delays that could be improved or even removed entirely. The next step would be to investigate other ways to estimate the output DUT torque to reduce the delay caused by the low-pass filtered acceleration. Some state estimation methods might be used, such as a Kalman filter or other predictive methods, which could have a lower phase delay compared to a traditional low-pass filter.

Another way of improving the proposed method and potentially further reducing the system's delays is to use a real-time operating system as a base for the emulation algorithm. Currently, the emulator in the physical system is implemented in Python, which is not ideal for real-time applications since Python's runtime environment does

not offer real-time capabilities. In fact, running the emulator loop at a relatively constant time rate proved challenging, and failing to do so can cause some of the stability issues seen.

The emulated loads during the experiments in this thesis were limited to linear loads. However, as the authors in [5] prove, adding non-linear loads to the emulator should be relatively straightforward.

Finally, analyzing and implementing a different emulation method could also be interesting. A method where a non-linear dynamometer plant is used, like [11], could potentially improve some of the issues caused by model errors in terms of friction. If it is of less importance that the emulator should achieve exact dynamic matching, one could investigate one of the direct methods used in [3] which could potentially improve the stability.

Bibliography

[1] Erik Cheever. *Table of laplace and Z transforms.* URL: https://lpsa.swarthmore.edu/LaplaceZTable/LaplaceZFuncTable.html.

[2] Gan Chengwei, R. Todd, and J.M. Apsley. "Drive System Dynamics Compensator for a Mechanical System Emulator." In: *IEEE Transactions on Industrial Electronics, Industrial Electronics, IEEE Transactions on, IEEE Trans. Ind. Electron* 62.1 (2015), pp. 70–78. ISSN: 1557-9948. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.6823665&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[3] E.R. Collins and Y. Huang. "A programmable dynamometer for testing rotating machinery using a three-phase induction machine". In: *IEEE Transactions on Energy Conversion* 9.3 (1994), pp. 521–527. DOI: 10.1109/60.326471.

[4] Barry Dupen. *Applied Strength of Materials for Engineering Technology.* July 2021.

[5] Z. Hakan Akpolat, G.M. Asher, and J.C. Clare. "Dynamic emulation of mechanical loads using a vector-controlled induction motor-generator set". In: *IEEE Transactions on Industrial Electronics* 46.2 (1999), pp. 370–379. DOI: 10.1109/41.753776.

[6] "IEEE Recommended Practice for Excitation System Models for Power System Stability Studies". In: *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005)* (2016), pp. 1–207. DOI: 10.1109/IEEESTD.2016.7553421.

[7] Edward W. Kamen. "Applied Optimal Control and Estimation." In: *IEEE Transactions on Automatic Control* 39.8 (1994), p. 1773. ISSN: 0018-9286. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edsgao&AN=edsgcl.16061271&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[8] Kooksun Lee et al. "A Robust Emulation of Mechanical Loads Using a Disturbance-Observer". In: *Energies* 12.12 (2019). ISSN: 1996-1073. DOI: 10.3390/en12122236. URL: https://www.mdpi.com/1996-1073/12/12/2236.

[9] Carlos Matheus Rodrigues de Oliveira et al. "High-Accuracy Dynamic Load Emulation Method for Electrical Drives". In: *IEEE Transactions on Industrial Electronics* 67.9 (2020), pp. 7239–7249. DOI: 10.1109/TIE.2019.2942566.

[10] *Robot Applications.* https://www.mogi.bme.hu/TAMOP/robot_applications/index.html(visited 2024-01-16). Accessed: 2024-01-16.

[11] M. Rodic, K. Jezernik, and M. Trlep. "Dynamic emulation of mechanical loads - approach based on nonlinear control". In: *2005 European Conference on Power Electronics and Applications*. 2005, 10 pp.–P.10. DOI: 10.1109/EPE.2005.219287.

[12] P. Sandholdt et al. "A dynamometer performing dynamical emulation of loads with nonlinear friction." In: *Proceedings of IEEE International Symposium on Industrial Electronics* 2 (1996), p. 873. ISSN: 9780780333345. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edb&AN=92490616&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[13] Viktor Šlapák et al. "Emulation of Mechanical Loads in Electric Drives – An Overview of Methods." In: *Power Electronics and Drives* 8.1 (2023), pp. 53–64. ISSN: 2543-4292. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edsdoj&AN=edsdoj.87cc704a43324a57a847036f69d6581b&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[14] *T20WN*. B0673-9.0 en. HBM.

A

Appendix 1

A.1 Torque sensor measurement delay

This section briefly describes the measurement that is performed in order to evaluate any potential torque sensor output delay. An oscilloscope is configured with one channel measuring the current from one of the phases on the DUT. The second channel measures the analog output voltage from the torque sensor. A single output torque step demand is sent to the DUT, starting from zero torque output, and the resulting waveforms are triggered by the oscilloscope. This can be seen in figure A.1. The time between phase current rise and output torque rise is measured and results in a delay of approximately 1.7 ms.

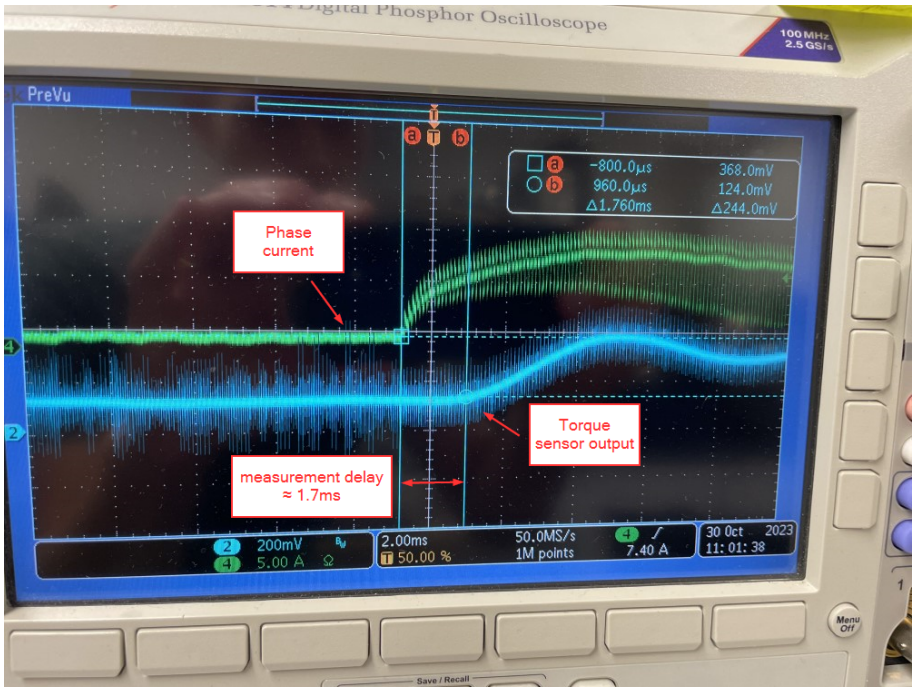


Figure A.1: Torque sensor delay measurement. Phase current (yellow) vs torque sensor output (blue).

A.2 Load motor set torque delay

An analysis of the load motor output torque delay is performed and the measurement set-up is presented here. The set-point torque demand to the load motor is sent from the emulator python program at every loop iteration of the emulator. This is done over CAN bus. The delay of interest is the time it takes from a command being sent on CAN bus to actual torque output on the shaft. An oscilloscope is configured to capture the CAN frames on one channel and the analog voltage from the torque sensor on a second channel. Starting from zero output torque, a set-point step in torque is sent and the resulting waveforms are captured on the oscilloscope. The result of this can be seen in figure A.2. As can be seen, the resulting time delay is measured to be approximately 2.7 ms.

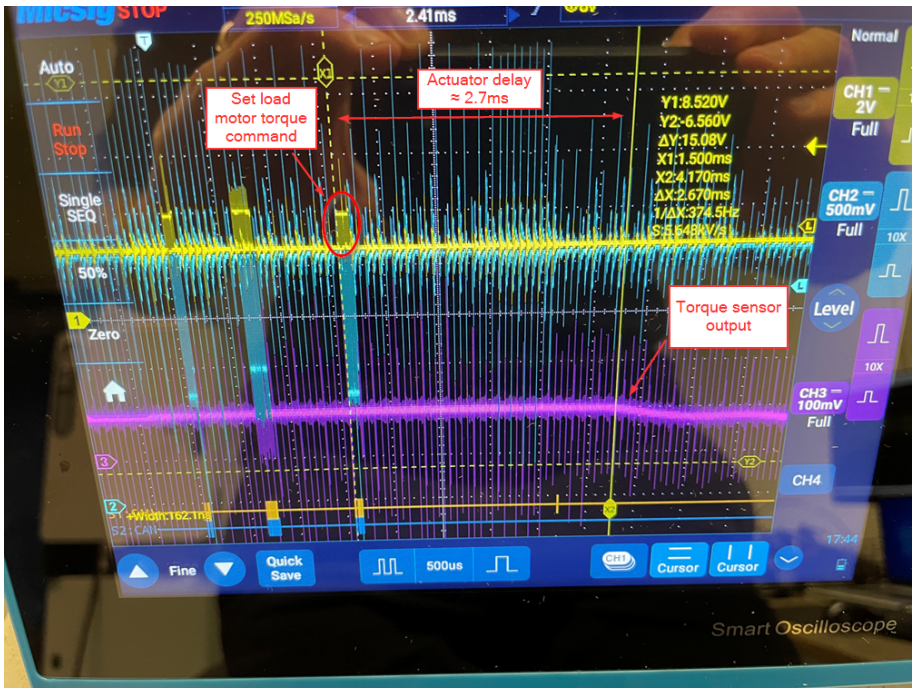


Figure A.2: Measurement of load motor set torque delay. Time measured between set torque command on CAN bus (yellow) and torque sensor output (purple).

A.3 Non-linear friction model

As is shown in section 3.2.1.3, the linear viscous friction model in (3.22) and (3.23) lead to mismatch between simulated and measured data. An alternative friction model is presented here that takes non-linear speed dependency into consideration.

By analysing the data in figure 3.7, the measured system exhibits higher friction at lower velocities compared to the linear model. This can be modelled as a power-law decay model:

$$T_{fv} = B\,|w|^n\, sgn(w) \tag{A.1}$$

where B is the viscous friction coefficient, n is the exponent that determine the decay ($0 < n < 1$), w is the angular velocity and $sgn(w)$ is the signum function that sets the correct direction of the friction.

The friction in the system also seem to have a static part, as can be seen in the very end of the graph in figure 3.7 where the system is at stand-still ($\dot{\theta}_D = 0$) even though the torque input is slightly larger than 0. This is, of course, not captured in the simulation since the model only contains the linear viscous friction. The system also exhibits some directional dependency. By looking closer at figure 3.7, there are higher frictional forces for positive rotation compared to negative rotation. Both of these effects, the static friction and the directional dependency, can be modelled as Coulomb friction with different coefficients for different directions:

$$T_{fc} = \mu \times sgn(w), \text{ where } \begin{cases} \mu = \mu_+ & \text{for } \omega > 0 \\ \mu = \mu_- & \text{for } \omega < 0 \\ \mu = 0 & \text{for } \omega = 0 \end{cases} \tag{A.2}$$

In order to be able to use the expression for T_{fc} in a differential equation that can be simulated, the expression can be rewritten as:

$$T_{fc} = \Delta\mu + \mu_{avg} \times sgn(w) \tag{A.3}$$

where $\Delta\mu$ is the mean of the friction coefficients for both directions, representing the base value regardless of direction:

$$\Delta\mu = \frac{\mu_+ + \mu_-}{2} \tag{A.4}$$

and μ_{avg} is half the difference between the coefficients. When multiplied by the sign of the angular velocity, this will give the correct friction force for each direction:

$$\mu_{avg} = \frac{\mu_+ - \mu_-}{2} \tag{A.5}$$

Basic arithmetic will prove that the resulting T_{fc} from (A.3) will be equal to that of (A.2), with the exception of $w = 0$.

The resulting friction model can be expressed as:

$$T_f(w) = T_{fv} + T_{fc} \tag{A.6}$$

With the linear models for the DUT and LM in 3.15 and 3.16 in mind, the models can then be modified with the non-linear friction term as the following:

$$J_d\ddot{\theta}_d + T_{f,d}(\dot{\theta}_d) - T_s = 0 \tag{A.7}$$
$$J_l\ddot{\theta}_l + T_{f,l}(\dot{\theta}_l) + T_s = 0 \tag{A.8}$$

or

$$\ddot{\theta}_d = \frac{1}{J_d}(T_s - T_{f,d}(\dot{\theta}_d)) \tag{A.9}$$
$$\ddot{\theta}_l = \frac{1}{J_l}(-T_s - T_{f,l}(\dot{\theta}_l)) \tag{A.10}$$

where $T_{f,d}(w)$ and $T_{f,l}(w)$ are the non-linear friction torques for the DUT and LM side respectively.

The parameters B, n in (A.1) and μ_+ and μ_- in (A.3) is fitted manually for each sub-system until the simulated angular velocity corresponded well to the measured velocity. The resulting parameters are presented here:

DUT parameters:

$$T_{f,d} = \begin{cases} B & = \tilde{B}_d = 0.0028\mathrm{N\,m\,s\,rad^{-1}} \\ n & = 0.9 \\ \mu_+ & = 0.2 \\ \mu_- & = -0.08 \end{cases} \tag{A.11}$$

LM parameters:

$$T_{f,l} = \begin{cases} B & = \tilde{B}_l = 0.0039\mathrm{N\,m\,s\,rad^{-1}} \\ n & = 0.8 \\ \mu_+ & = 0.16 \\ \mu_- & = -0.27 \end{cases} \tag{A.12}$$

The differential equations (A.9) and (A.10) for the DUT and LM model respectively, is simulated in Matlab with the corresponding input torque T_s from figures 3.4 and 3.5. The output angular velocities are compared to the measured data of each system. The results can be seen in figure A.3 and A.4. Compared to the results of the linear models in 3.7 and 3.8, the simulated velocity now matches the measured velocity to a higher degree.

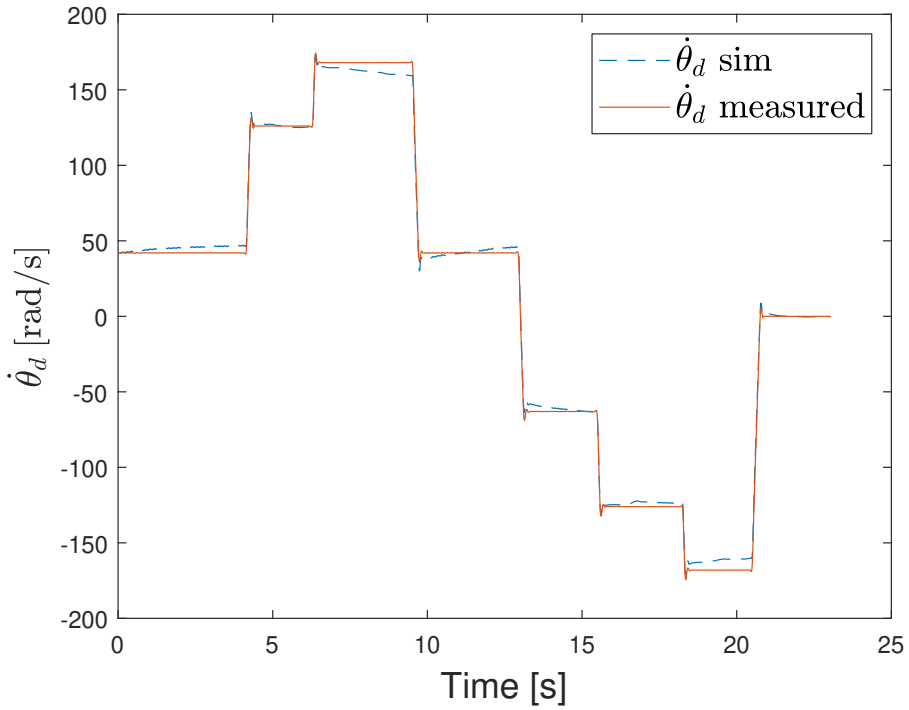


Figure A.3: DUT nonlinear friction model. Simulated speed vs measured speed

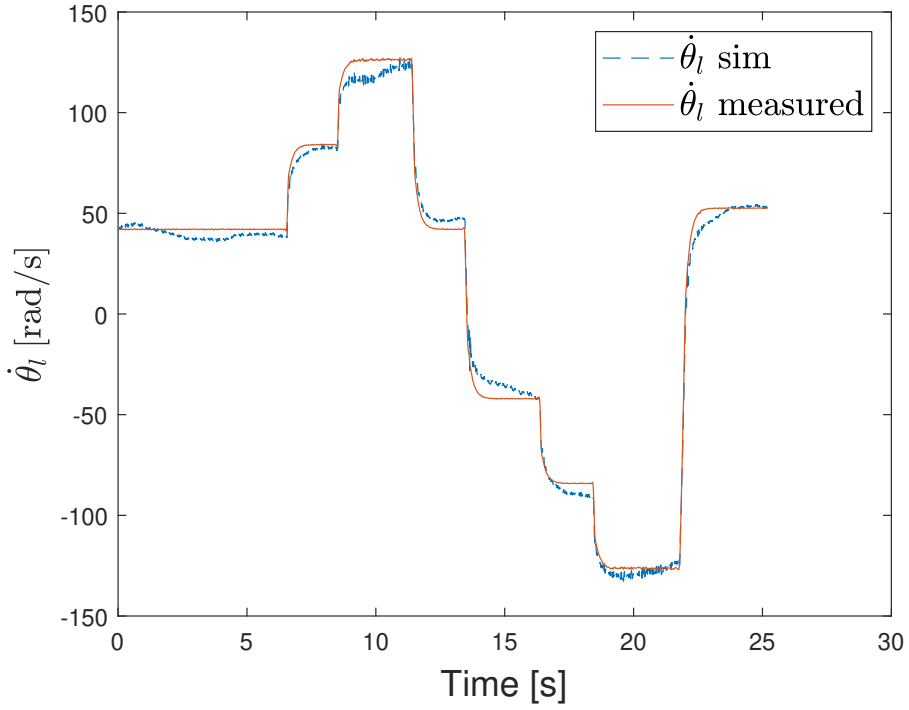


Figure A.4: LM nonlinear friction model. Simulated speed vs measured speed

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY