UiA University of Agder

Bachelor

Ball and Beam

Ball and Beam project from Bosch Rexroth based on CtrlX, an Electromechanical System

ANDREAS AASEN AND FREDRIK VATNE RYLANDER

SUPERVISOR
Morten Ottestad and Babar Ali

University of Agder, 2023
Faculty of Engineering and Science
Department of Engineering Sciences

Acknowledgements

We would like to express our sincere gratitude to all collaborators in this project. Especially:

- Morten Ottestad, Assistant professor at the University of Agder, for providing guidance and expertise on the theoretical aspects of our project.
- Babar Ali, Business Developer (Automation and digitalization Norway) and IoT/Industry 4.0 Expert Europe North, Bosch Rexroth, for assisting us in initialization using and getting started with the CtrlX software. We are also grateful for providing us with the necessary hardware used in the project and for getting in contact with Bosch Rexroth's development team in Germany. Furthermore, we are grateful for the insightful visit we had to Bosch Rexroth in Langhus, where we gained valuable insights into the software's application.
- Jorg Giebler, Business Development Manager at Bosch Rexroth, has helped us from Germany to solve programming challenges and helpful conversations about using CtrlX.
- Michael Schafer, Application & Technical Support Motion Solutions DC-AE/STS1 Bosch Rexroth and Karsten Kreusch DC-AE/PRM1 Bosch Rexroth. Helped us create a 3D model for the 3D Viewer App for visual presentation of Ball and Beam.

Abstract

Modern electro-mechanical control systems prioritize efficient and precise regulation. In this bachelor's thesis, we address the Ball and Beam problem an inherently unstable system that requires continuous monitoring and correction for stabilization. The proposed solution successfully stabilizes the system using a Bosch Rexroth ctrlX drive with embedded ctrlX Core and servo motor, 3D-printed parts, carbon tubes, a cascade closed-loop controller, and a remote visual representation of the model, promoting learning and interaction through an HMI.

The project began with constructing a beam attached to the servo, prioritizing a low moment of inertia through the use of 3D-printed components and carbon tubes. The dynamics of the beam with the ball on it were simulated to design a regulator for the system, which was implemented in the CtrlX PLC and connected to a user-friendly HMI programmed with the WebIQ designer, communicating over ctrlX OPC UA server app.

The cascade closed loop controller, consisting of an inner velocity loop and outer position loop, was employed to regulate the ball on the beam. Stability was achieved by tuning the velocity control, the cascade loop, and a low-pass filter to address noise and vibration issues. A comparison with the digital twin confirmed that the beam's dynamics were approximately the same.

The HMI and, the visual representation of the model, allowed for remote operation and promoted learning through user interaction, enabling the operator to see the system's motion and control it from a distance. This feature proved beneficial for educational purposes and is an illustration of the practical applications where remote monitoring and control are necessary.

For future improvements, it is recommended to use a gearbox for the beam due to high moment of inertia mismatch, employ a servo better suited for high inertia loads with a lower maximum rotation speed, and utilize a different sensor and enhanced filter to suppress noise. These refinements will contribute to improvements in the electro-mechanical system, with a focus on stability, speed, and precision.

Keywords: Ball and Beam, CtrlX, simulation, Matlab Simscape multibody, Regulation technique, system identification, HMI, Ball and Beam mechanical design, low pass filter, Balluff Photoelectric Sensors

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the demand for precise and efficient control systems in various industries has grown a lot. Electromechanical control systems play an important role in achieving high levels of precision, reliability, and performance. One of the classic problems in control theory is the Ball and Beam system, a system that is unstable and non-linear that requires continuous monitoring and control to maintain stability. The main goal of this bachelor's thesis is to design, implement and optimize a Ball and Beam system using Bosch Rexroth's ctrlX Automation promotion set and control this externally, with a focus on promoting learning and interaction through a human-machine interface (HMI).

The Ball and Beam system consists of a ball that rolls on a beam that is controlled by a servo motor. The challenge lies in designing a controller that can regulate the position of the ball on the beam accurately and efficiently. This thesis proposes a solution using a Bosch Rexroth ctrlX Drive with integrated ctrlX Core and servo motor, 3D printed parts, carbon tubes, a cascaded closed loop controller, and a visual representation integrated into an interactive WebIQ HMI.

The assignment is organized as follows:

- Part 1 is an introduction.
- Part 2 is a theory part where differential equations describe the dynamics of the system. Control theory with design and tuning is also in this section where cascade regulation is the main theme.
- Part 3 describes the method of how the mechanical system is designed with the first mockup as a starting point for the actual ctrlX system which gives a detailed overview of the design and construction of the Ball and Beam system. Here with a focus on minimizing the moment of inertia using 3D printed parts and carbon tubes.
- Part 4 deals with PLC programming, design of digital twin in simscape and programming of HMI with emphasis on the advantages of external control and promotion of learning through user interaction.
- Part 5 results present from the method dealing with controls, system response, digital twin, and safety.
- Part 6 discusses the results against the method and project objectives. Future improvements and possible sources of error are discussed here.
- Part 7 contains the conclusions, where it is concluded whether the tasks in the project have been met. Further work and improvements are also addressed.

By designing and optimizing a Ball and Beam system with ctrlX, remote control capabilities, and an interactive HMI, this thesis aims to contribute to the development of efficient and precise PID control solutions suitable for both educational and practical applications.

Figure 1.1: Bosch Rexroth ctrlX Automation promoter set, Mnr:R91139860

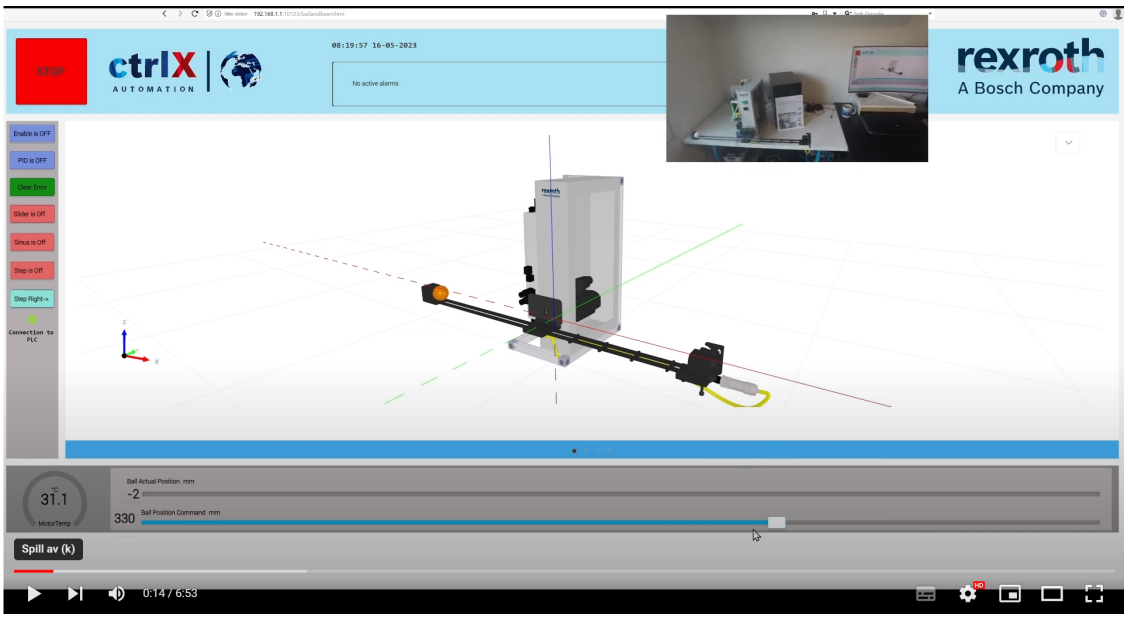

Figure 1.2: Demonstration video of the finished Ball and Beam system

Follow the link for a demonstration video of the finished Ball and Beam system including a comparison between the real system and the digital twin:

https://youtu.be/umBTghe88xY

All project files are accessible in Github by following this link, [26].

1.1 Project targets

It is important to establish clear objectives for the "ball and beam" system in order to ensure that it functions as intended. In order to accomplish this, system specifications for the "Ball and Beam" need to be set. By establishing goals, it is possible to maintain a consistent focus throughout the project, aligning the objectives with the customer's needs and what is feasible based on existing research and theoretical frameworks.

Figure 1.3: Bosch Rexroth ctrlX Automation promoter set with beam

The project tasks were provided by Bosch Rexroth, Fig. B.1, and additional tasks, 7 and 8, which were agreed upon in a Teams meeting on 09.02.23 with Joerg Giebler. The main targets for this project are:

1. Calculate differential equations of the mechanics
2. Design and build the mechanics of the ball and beam which is attached to the electric motor
3. Select a sensor for measuring the ball position
4. Simulation of the behavior using Matlab/ Simulink/ Simscape.
5. Design of a controller to make the system stable
6. Programming of the ctrlX PLC for interface with the WebIQ HMI
7. Design a digital twin and compare it to the real system
8. Design an intuitive HMI that displays the real-time state of the system and lets the operator interact for educational purposes

1.2 Literary review

The preparation of the Ball and Beam thesis was partly based on information found on the Bosch Rexroth website in the form of datasheets, application manuals, how-to-videos, and the CtrlX forum. Other sources for the theory were found in articles, online publications, lecture notes, and university websites. Youtube was used for informational purposes in the programming of the ctrlX PLC, WebIQ HMI, and Simscape. Datasheets for 3rd-party components were used as information. The built-in documentation in Matlab, Simscape, and Simulink were used as support when building models.

1.3 Bosch Rexroth

Bosch Rexroth AG is a global engineering company and a leading provider of drive and control technologies, systems, and solutions. With a rich history spanning over two centuries, the company has continuously evolved to meet the changing needs of various industries, including construction, automotive, aerospace, material handling, and renewable energy. As a subsidiary of the multinational Bosch Group, Bosch Rexroth operates in more than 80 countries and has more than 32.000 employees.

The company offers services within Mobile Hydraulics, Industrial Hydraulics, Linear Motion Technology, Assembly Technology, and Electric Drives and Controls. [4]

1.3.1 ctrlX Automation

The ctrlX promoter set belongs in the Electric Drives and Control segment of Bosch Rexroth where they promote:

"ctrlX AUTOMATION surmounts the classic boundaries between machine controls, the IT world and the Internet of Things. With a Linux real-time operating system, consistently open standards, app programming technology, web-based engineering and a comprehensive IoT connection, ctrlX AUTOMATION reduces components and engineering costs by 30 to 50%."[38]

Chapter 2

Theory

2.1 Ball motion calculations

The following calculations assume pure roll. Friction and viscous damping are neglected to describe the system with Newton's 2nd law of motion. An equation describing the motion of the ball can be derived based on this law.

$$\sum F = m_b \cdot \frac{d^2x(t)}{dt^2} \tag{2.1}$$

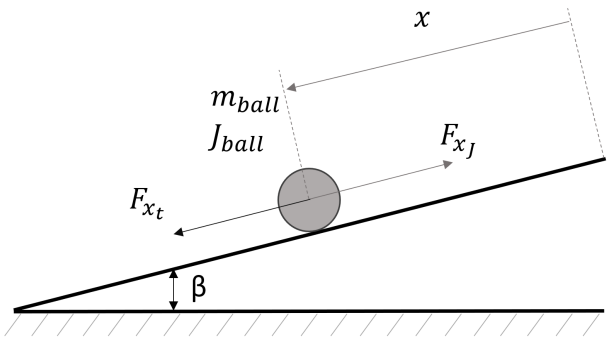

Figure 2.1: Free body diagram of the rolling force relations

The ball has inertia that will act as a force opposing the force generated by gravity. The force balance is shown in Fig. 2.1 and described:

$$m_b \cdot \frac{d^2x(t)}{dt^2} = F_{xt} - F_{xJ} \tag{2.2}$$

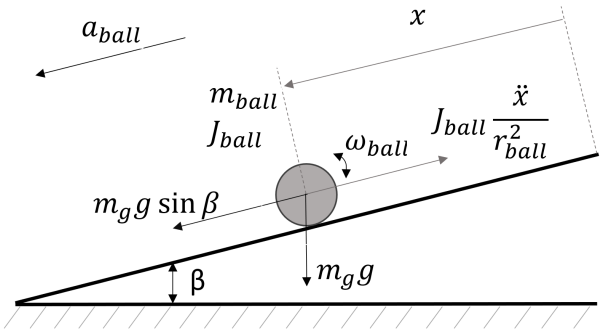

Figure 2.2: Free body diagram of the forces moving the ball

When the slope is not perpendicular to the gravitational force there will be a resulting force acting on the ball. The force balance in Fig. 2.2 is expressed with:

$$m_b \cdot \frac{d^2x(t)}{dt^2} = m_b \cdot sin\beta(t) - F_{xJ} \tag{2.3}$$

Classical mechanics for equations of motion, kinematics rectilinear motion. When acceleration is constant.

$$v_b = v_0 + a \cdot t \tag{2.4}$$

$$x = x_0 + v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \tag{2.5}$$

$$v_b^2 = v_0^2 + 2 \cdot a(x - x_0) \tag{2.6}$$

$$x = x_0 + \frac{1}{2}(v_0 + v)t \tag{2.7}$$

Classical mechanics for equations of motion, kinematics rotational motion. When acceleration is constant.

$$\omega = \omega_0 + a \cdot t \tag{2.8}$$

$$\theta = \theta_0 + \omega_0 \cdot t + \frac{1}{2} \cdot \alpha \cdot t^2 \tag{2.9}$$

$$\omega^2 = \omega_0^2 + 2 \cdot \alpha(\theta - \theta_0) \tag{2.10}$$

$$\theta = \theta_0 + \frac{1}{2}(\omega_0 + \omega)t \tag{2.11}$$

$$v = \omega \cdot r \tag{2.12}$$

$$\tau = I \cdot \alpha \tag{2.13}$$

$$a_N = \omega^2 \cdot r \tag{2.14}$$

$$a_T = r \cdot \alpha \tag{2.15}$$

Relating the ball inertia to the force it exerts when rolling must be done by the basis on the following formulas.
Ball moment of inertia thin wall.

$$J_b = \frac{2}{3} \cdot m_b \cdot r_{rotation}^2 \tag{2.16}$$

Ball rolling on beam, rotating about rail contact surface.

$$r_{rotation} = \sqrt{r_b^2 - \frac{Beam_w^2}{2^2}} \tag{2.17}$$

The ball angular velocity on beam:

$$\omega_b = \frac{v_b}{\sqrt{r_b^2 - \frac{Beam_w^2}{2^2}}} \tag{2.18}$$

The ball angular acceleration is:

$$\alpha_b = \frac{d\omega}{dt} = \frac{d^2\theta}{dt^2} \tag{2.19}$$

The ball velocity is:

$$v_b = \omega_b \cdot r_{rotation} \tag{2.20}$$

Cartesian coordinate system velocity:

$$|v_b| = \sqrt{\frac{d^2x}{dt^2} + \frac{d^2y}{dt^2} + \frac{d^2z}{dt^2}} \tag{2.21}$$

Cartesian coordinate system point position:

$$|p_b| = \sqrt{x^2 + y^2 + z^2} \tag{2.22}$$

The ball acceleration is:

$$\frac{d^2x(t)}{dt^2} = \alpha_b \cdot r_b \tag{2.23}$$

The torque is expressed:

$$\tau_b = r_r \cdot F_{xt} = J_b \cdot \alpha_b = J_b \frac{\frac{d^2x(t)}{dt^2}}{r_b} \tag{2.24}$$

By inserting eq 2.24 into eq 2.3

$$m_b \cdot \frac{d^2x(t)}{dt^2} = m_b \cdot sin\beta(t) - J_b \frac{\frac{d^2x(t)}{dt^2}}{r_b^2} \tag{2.25}$$

the following non-linear equation is derived:

$$\frac{d^2x(t)}{dt^2} = \frac{m_b \cdot r_b^2 \cdot g \cdot sin\beta(t)}{m_b \cdot r_b^2 + J_b} \tag{2.26}$$

By applying the small angle approximation to the equation it can be linearized. The angle β is assumed to be equal to the sine of angle *beta*

$$sin\beta(t) \approx \beta(t) \tag{2.27}$$

The resulting linearized equation is the mathematical expression of the ball acceleration if the angle is $\pm\ \pi/4$ rad.

$$\frac{d^2x(t)}{dt^2} = \frac{m_b \cdot r_b^2 \cdot g \cdot \beta(t)}{m_b \cdot r_b^2 + J_b} \tag{2.28}$$

Moving all the constant values into a vaiable C_b makes the equation easier to read.

$$\frac{d^2x(t)}{dt^2} = C_b \cdot \beta(t) \tag{2.29}$$

Taking the Laplace transform of the differential equation makes it easier to do calculations and simulations. This is due to the transform from time(t) to frequency(s) domain. Integration and derivation are simplified to algebraic expressions in the frequency domain.

$$s^2X(s) = C_b \cdot \beta(s) \tag{2.30}$$

Rearranging the equation to fit the transfer function that relates the beam angle to the ball position.

$$\frac{X(s)}{\beta(s)} = \frac{C_b}{s^2} \tag{2.31}$$

The ctrlX drive and servo system is considered to have a dynamic that allow for a assumption that $\beta_{cmd} = \beta_{act}$. The servo/drive plant is modeled as P_s and the ball and beam plant is named P_{bb}

$$P_s(s) = \frac{\beta(s)}{\beta(s)}, P_{bb}(s) = \frac{C_b}{s^2} \tag{2.32}$$

$P_s(s) = 1$ that leads to the following final plant

$$P(s) = P_s(s) \cdot P_{bb}(s) \Leftrightarrow P(s) = P_{bb}(s) \tag{2.33}$$

A transfer function that contains a double integrator $\frac{1}{s^2}$ will have two poles at the origin in a pole plot. Such a system is undamped and will need a feedback-loop and a controller to shift the poles to the left in order to stabilize. A PD-controller will dampen the system and the oscillations from a pure P- controller. PD-controller in Laplace:

$$G_{PD} = K_P + K_D s \Leftrightarrow G_{PD} = K_D[s + \frac{K_P}{K_D}] \Leftrightarrow G_{PD} = K_D[s + Z_{PD}] \tag{2.34}$$

The effect of a PD-controller is that it adds a zero, $Z_{PD} = \frac{K_P}{K_D}$, to the pole/ zero plot. The zero position has an affect on the transient response of the system, it will generally increase the bandwidth [51]. If the zero is put on the left side of the dominant poles it has the greatest effect the closer to the poles it is placed. The negative effect of adding a derivative control element is that it reacts to the rate of change, $\frac{de(t)}{dt}$, of the control error. Any noise on the feedback signal will trigger large control outputs from the derivative control. A low-pass filter on the position feedback can reduce some of the side effects.
A closed loop PD-position control uses the error as an angle input to correct for deviations from position command. The beam angle input is given by:

$$\beta_e(t) = K_P[x_{cmd}(t) - x_{act}(t)] - K_D\frac{dx_{act}(t)}{dt} \tag{2.35}$$

The Laplace transform:

$$\beta_e(s) = K_P[X_{cmd}(s) - X_{act}(s)] - K_D s(s) \tag{2.36}$$

Substituting $\beta(s)$ with eq 2.36 in eq 2.31:

$$\frac{X_{act}(s)}{\beta_e(s)} = \frac{C_b}{s^2} \tag{2.37}$$

That gives:

$$\frac{X_{act}(s)}{K_P[X_{cmd}(s) - X_{act}(s)] - K_D s(s)} = \frac{C_b}{s^2}$$

$$\frac{X_{act}(s)s^2}{C_b} = K_P[X_{cmd}(s) - X_{act}(s)] - K_D s(s)$$

$$\frac{X_{act}(s)}{X_{cmd}(s)} = \frac{C_b K_P}{s^2 + C_b K_D s + C_b K_P} \tag{2.38}$$

Comparing the transfer function to a standard second order transfer function

$$P(s) = \frac{X_{act}(s)}{X_{cmd}(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n^2 + \omega_n^2} = \frac{C_b K_P}{s^2 + C_b K_D s + C_b K_P} \tag{2.39}$$

makes it possible to calculate K_P and K_D gains. The natural frequency ω_n is key to calculate the damping ratio and settling time based on the system requirements.

$$K_P = \frac{\omega_n^2}{C_b} \tag{2.40}$$

$$K_D = \frac{2\zeta\omega_n}{C_b} \tag{2.41}$$

$$\zeta = \frac{-ln(\%OS/100}{\sqrt{\pi^2 + ln^2(\%OS/100)}} \tag{2.42}$$

$$T_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \tag{2.43}$$

2.2 Ball Dynamics

Calculating the ball's dynamics is important to evaluate how the ball will roll and move on the beam. By utilizing eq. 2.26, one can calculate the effect of acceleration on the ball based on the angle of impact. For the ball to move on the beam, the beam must be inflicted an angle and adhere to Newton's laws of acceleration, which are influenced by gravity, the beam must be rotated as depicted in Fig. 2.2. This rotation of the beam results in an addition of energy to the ball. Assuming that the beam rotates so that the ball has higher potential energy ($m \cdot g \cdot h$), the equation for centripetal acceleration will come into play by eq. 2.14. If the rotation velocity is high enough, the centripetal acceleration can be higher than the potential energy, therefore the ball can roll up

by equation if $a_b < a_N$. If the beam rotates so the centripetal force is less than the gravitational acceleration, then $a_b > a_N$ the ball will roll down.

$$a_b > a_N \tag{2.44}$$

$$a_b > \omega^2 \cdot r \tag{2.45}$$

$$\frac{(g \cdot sin\beta)}{1 + \frac{2 \cdot r_b^2}{3 \cdot (r_b^2 - \frac{Beam_w^2}{2^2})}} > (\frac{v_t}{r_{beam}})^2 \cdot ball_{position} \tag{2.46}$$

If the beam rotates away from the ball so the ball accelerates downward without losing contact with the beam. Then the centripetal force can add energy $a_b + a_N$ and Newton's laws see Fig. 2.2 and the ball will roll down.

2.3 Drive, Motor and Beam Dynamics

Bosch Rexroth provided the unit with a drive and motor, which has its own control layout and tuning mechanisms. To model the beam motion dynamics, a real-life step response was conducted to establish the actual beam motion response. This model is then used in the Matlab/ Simulink simulations for controller design and optimization.

Designing a system where a motor is driving a load requires a calculation of the mass moment of inertia of the motor in relation to that of the load. The relation between the inertia acting on the motor shaft and the rotor inertia should not be greater than 10:1. If the mass moment of inertia of the load is much greater than that of the motor, the load will start to drive the motor. Such conditions lead to poor control of the load and potentially high current draw.

The mass moment of inertia of the load must be calculated as reflected on the motor shaft. There is no need to do any calculations in a direct drive as the load is directly applied to the shaft so $n = 1$. If there is a gearbox between the load and the motor shaft, the following calculation must be done. J_M is the inertia of the rotor. J_L is the load inertia and J_{LM} is the load inertia as it is seen by the motor shaft.

$$J_{LM} = \frac{J_L}{n^2} \tag{2.47}$$

$$n_{opt} = \sqrt{\frac{J_{LM}}{J_M}} \tag{2.48}$$

Angular acceleration motor with gearbox.

$$\alpha_L = \frac{\alpha_M}{n} \tag{2.49}$$

Torque motor with gearbox.

$$T_M = \frac{T_L}{n} \tag{2.50}$$

The mass moment of inertia relation calculated:

$$Relation_J = \frac{J_{LM}}{J_M} \tag{2.51}$$

2.3.1 Finding a Transfer Function from a Step Response

A step input to a system and the following trace serve as the base for finding the transfer function that models the system. On inspection of the trace, the following values are found:

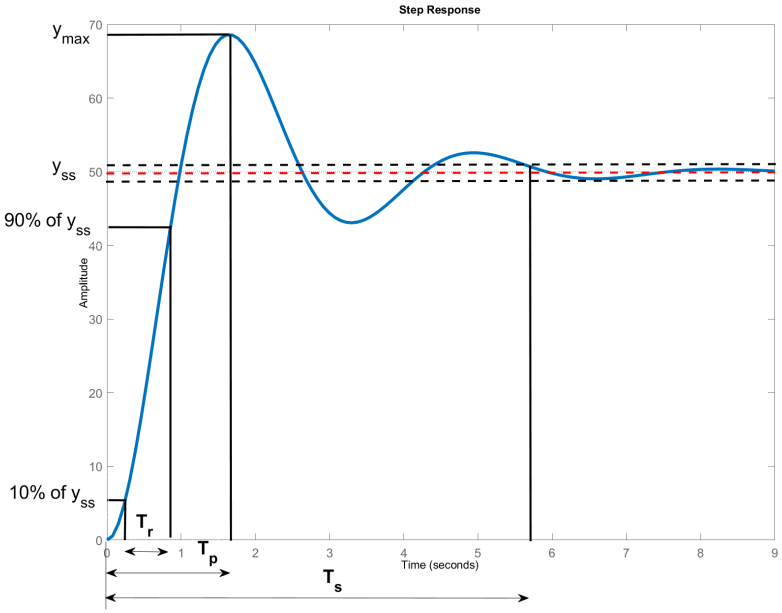

Figure 2.3: Step response of a 2nd order system

- Steady state value, y_{ss}
- Max value, y_{max}
- Peak time, T_p, time from step input to reach the highest peak
- Step input, U

Percent overshoot is calculated:

$$\%OS = \frac{y_{max} - y_{ss}}{y_{ss}} * 100 \tag{2.52}$$

Damping ratio ζ is found with:

$$\zeta = \frac{-ln(\%OS/100)}{\sqrt{\pi^2 + ln(\%OS/100)^2}} \tag{2.53}$$

The natural frequency ω_n is found from the peak time T_p:

$$\omega_n = \frac{\pi}{T_p\sqrt{1-\zeta^2}} \tag{2.54}$$

Lastly the gain, K is found:

$$K = \frac{y_{ss}}{U} \tag{2.55}$$

These values give the transfer function that equals the step response of the system:

$$P_s(s) = \frac{K}{\frac{s^2}{\omega_n^2} + \frac{2*\zeta}{\omega_n}s + 1} \tag{2.56}$$

2.3.2 Final Model

Combining the two transfer functions describes a model of the beam and ball system, G(s):

$$G(s) = P_s(s) * P_{bb} \tag{2.57}$$

The final closed-loop model adds a feedback H(s) that gives the model T(s):

$$T(s) = \frac{G(s)}{1 + G(s) * H(s)} \tag{2.58}$$

The open loop model with input R(s) and output C(s) Final

$$C(s) = R(s) * G(s) \tag{2.59}$$

2.4 Controller Design

The open loop frequency response is possible when both the drive/motor/beam and ball dynamics are modeled. Analyzing the resulting bode plot shows the stability of the closed-loop system. The system is stable if the magnitude is less than 0db at the frequency where the phase is ± 180 deg.

The controller design can be done with PID, lead- or lag- compensator. A PID- compensator is more of an iterative process where the values are updated based on the system response in the time domain. One method for tuning a PID system is the Ziegler-Nichols closed-loop tuning. Lead- and Lag-compensators are calculated from a desired overshoot and steady-state error requirement by analyzing the open loop frequency response bode plot in the frequency domain.

2.4.1 Ziegler-Nichols PID tuning

Ziegler-Nichols closed loop tuning involves exciting a control system with a step input to achieve a steady oscillating state, thereby obtaining the ultimate gain K_u and the period P_u. The P- controller is adjusted during this process. The obtained values are then used to set the corresponding gains in the table below:

	PID Type	K_p	K_i	K_d
	P	$0.5K_u$	0	0
	PI	$0.45K_u$	$0.54K_u/P_u$	0
	PD	$0.8K_u$	0	$0.10K_uP_u$
	PID	$0.6K_u$	$1.2K_u/P_u$	$0.075K_uP_u$

Table 2.1: Ziegler-Nichols Gains

The values calculated from the table are inserted into the following control law which is the output of the PID u(t), in the time domain, with error, e(t):

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{2.60}$$

The same equation in the frequency domain with output U(s) and error E(s):

$$U(s) = E(s)(K_p + K_i \frac{1}{s} + K_d s) = E(s) \frac{k_d s^2 + K_p s + K_i}{s} \tag{2.61}$$

2.4.2 Lead Compensator

The lead compensator is used to reduce the overshoot and realize a faster transient response. It can be compared to a PD- compensator in the time domain. A bode plot shows how the compensator increases the crossover frequency for a faster transient response and the increase in phase margin reduces the percentage overshoot.

Specification of the performance in the time domain	**Specification of the performance in the frequency domain**
Rise time,T_r from 10% to 90% of y_{ss}	Bandwidth, ω_{bw}
Percent overshoot	Phase margin, $\Phi_{desired}$
Settling time, T_s within +-5% of y_{ss}	Gain margin
Steady state error, $y_{ss} = e_{ss}$	Low frequency gain, K

Table 2.2: Relations between the time- and frequency- domain

By implementing a lead compensator, the open-loop frequency response is made to achieve the desired closed-loop performance in the time domain. To achieve the desired performance the 4 values in table 2.2 values have to be specified.
The following equations are used to construct the lead controller based values from Fig. 2.3.
The phase margin is found from the percentage overshoot requirement and calculated with

$$\Phi_{desired} = 70 - OS \tag{2.62}$$

The open loop bandwidth is approximately the same as the closed loop bandwidth, ω_{gc}. $\omega_{bw} \approx \omega_{gc}$.
The desired rise time is calculated

$$\omega_{bw} = \frac{1.8}{T_r} \tag{2.63}$$

The lead transfer function H_{Lead} is of type:

$$H_{Lead} = K\frac{aTs + 1}{Ts + 1} \tag{2.64}$$

K is the amplification found in the final value theorem that that meets the steady state error requirements. The $\sqrt{a}$ is the magnitude of where the lead phase is at its highest. T describes the middle point of the new cross-frequency, ω_{cnew}.

Finding the system type in a unity feedback system shows if the system has a steady state error or not. The system type is found by looking at the grade, s^n, of the s in the denominator of the system transfer function as eq(2.65) shows. The system type is the number of pure integrations in the forward path. z_n are the zeros and p_n are the poles:

$$system = \frac{K(s + z_1)(s + z_2)...}{s^n(s + p_1)(s + p_2)...} \tag{2.65}$$

Depending on what type of system, the following table shows the error in each case and what input to use, Step $u(t)$, ramp $tu(t)$ or parabola $\frac{1}{2}t^2u(t)$. $K_p = \lim_{s\to 0} G(s)$, $K_v = \lim_{s\to 0} sG(s)$ and $K_a = \lim_{s\to 0} s^2G(s)$
The final value theorem is used to calculate the gain needed to reach the steady-state error, e_{ss},

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s) \tag{2.66}$$

For the input R(s), step is $R(s) = \frac{1}{s}$, ramp is $R(s) = \frac{1}{s^2}$ and parabola is $R(s) = \frac{1}{s^3}$
The error $E(s)$ is expressed with the gain K and R(s) for the system F(s):

$$E(s) = \frac{1}{1 + KF(s)}R(s) \tag{2.67}$$

Input	Steady-state error formula	Type 0 Static error constant	Type 0 Error	Type 1 Static error constant	Type 1 Error	Type 2 Static error constant	Type 2 Error
Step, $u(t)$	$\frac{1}{1+K_p}$	K_p = Constant	$\frac{1}{1+K_p}$	$K_p = \infty$	0	$K_p = \infty$	0
Ramp, $tu(t)$	$\frac{1}{K_v}$	$K_v = 0$	∞	K_v = Constant	$\frac{1}{K_v}$	$K_v = \infty$	0
Parabola, $\frac{1}{2}t^2u(t)$	$\frac{1}{K_a}$	$K_a = 0$	∞	$K_a = 0$	∞	K_a = Constant	$\frac{1}{K_a}$

Figure 2.4: Unity feedback system type and error [58]

The steady state error, e_{ss}, is expressed with:

$$e_{ss} = e(\infty) = \lim_{s\to 0}(sE(s)) = \lim_{s\to 0}\left(\frac{sR(s)}{1+KF(s)}\right) \tag{2.68}$$

From the equation, eq(5.7) the gain K is found and based on a new bodeplot of $KG(s)$, the additional gain needed, Φ_m, is calculated with the added 5° to 15° phase shift. This added value is due to a shift in frequency as some gain is added from the lead compensator. Incorporating gain K ensures that the steady-state error meets the required specifications.

$$\Phi_m = \Phi_{desired} - \Phi_{new} + [5°to15°] \tag{2.69}$$

The $\sqrt{a}$ is the magnitude in db where the highest phase is added. a is calculated from:

$$a = \frac{1+\sin\Phi_m}{1-\sin\Phi_m} \tag{2.70}$$

The new cross frequency, ω_{cnew}, is where the magnitude $KH_{Lead} = \frac{1}{\sqrt{a}}$ in db crosses the 0db line.

$$KH_{Lead} = \frac{1}{\sqrt{a}} \Rightarrow 20\log\left(\frac{1}{\sqrt{a}}\right)[db] \tag{2.71}$$

T is found with the following equation, where $\omega_{cnew} = \omega_m$:

$$T = \frac{1}{\omega_m\sqrt{a}} \tag{2.72}$$

2.4.3 First Order Low-Pass Filter

A first-order low-pass filter has a cutoff frequency, ω_0 at -3db. At this point the magnitude has dropped to 70.7% of the low-frequency amplitude. A low-pass filter leaves the low frequency intact and reduces the magnitude of higher frequencies from the cut-off point. The continuous transfer function

$$H(s) = \frac{\omega_0}{s+\omega_0} \tag{2.73}$$

describes the lowpass filter.

A discrete-time conversion is done in matlab using code "c2d" which converts continuous transfer functions to discrete with a specified cycle time. The coefficients that are found are used in a recursive filter in the form:

$$y(n) = \sum_{k=0}^{K} b_k x(n-k) + \sum_{k=1}^{K} a_k y(n-k) \tag{2.74}$$

where the output values are influenced by the previous values. The digital version is an Infinite Impulse Response, IIR, which takes a set of coefficients derived from H(s). These coefficients, a_k and b_k, are used in the filter as seen below:

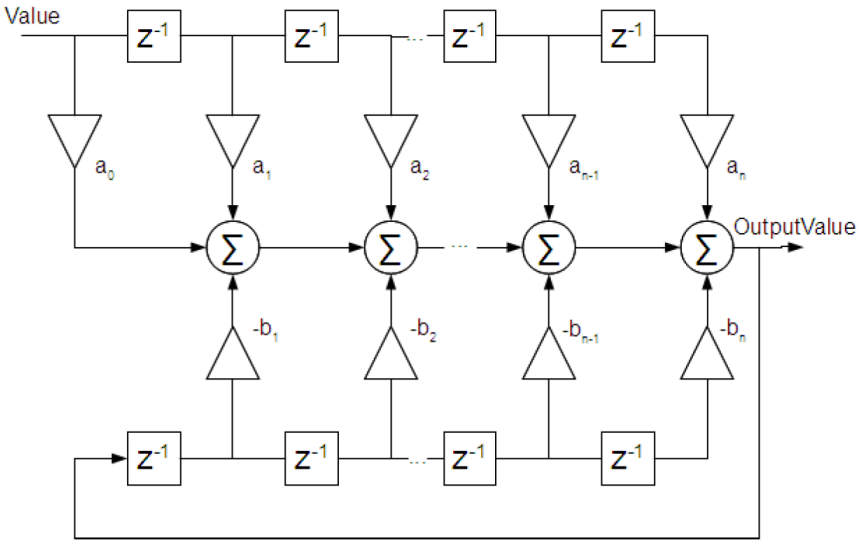

Figure 2.5: Digital IIR recursive filter, [41]

The implementation of a low-pass filter can cause a phase delay and a reduction in the overall system bandwidth. This can affect the system's stability, response time, and tracking performance.

2.4.4 Control of the Drive, Motor, and Beam

Bosch Rexroth has manufactured a Drive that incorporates a built-in control structure that optimizes the motor and drive to the load. This function, axis optimization, uses actual axis motion to establish the internal gains for optimal control. The theory and function are described in the Bosch Rexroth documentation [30].

2.4.5 Position Control Loop

A position control loop takes the command signal and compares it to the feedback signal. This signal is the error that the compensator feeds to the system to correct it.

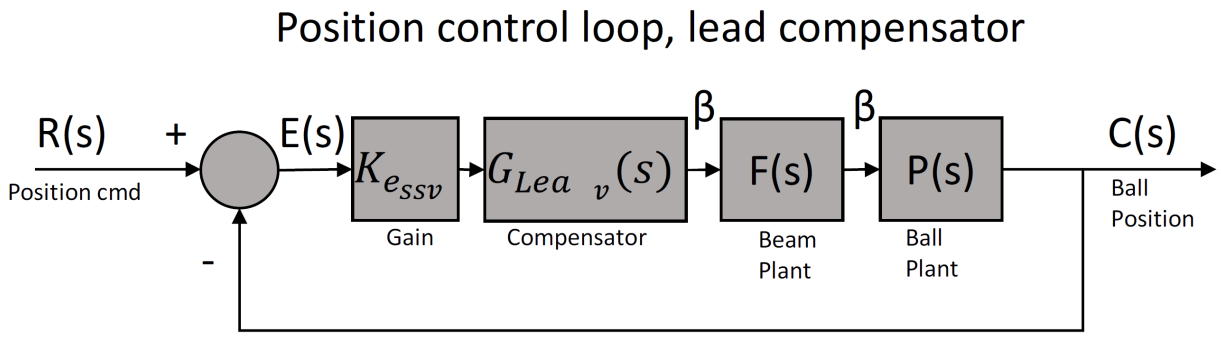

Figure 2.6: Position feedback loop

2.4.6 Velocity Control Loop

A velocity feedback loop is designed to maintain the commanded velocity by reducing the error, E(s), to zero. The output from the ball plant is a position. The velocity is found by derivation of the signal and fed back into the loop.

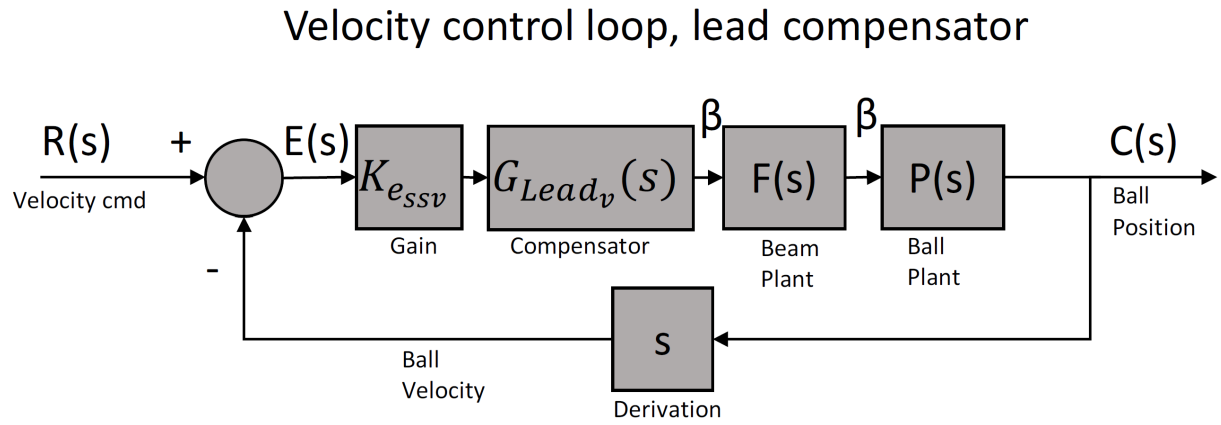

Figure 2.7: Velocity feedback loop

2.4.7 Cascade Loop Control

The cascade control loop combines the inner velocity loop with an outer position loop. The inner loop must be at least 3 times as fast as the outer for good control as explained by M. Ottestad [24].

Benefits of such a structure are as explained by Kaya, Tan & Atherton [16]:

- Improved disturbance rejection: The inner velocity loop can effectively attenuate disturbances before they propagate to the outer position loop for better performance
- Improve the system where load changes are present
- Faster response time: The inner velocity loop can typically be designed to respond faster than the outer position loop. This allows for a quicker response to changes in the reference input or disturbances, improving the overall system performance.
- Better stability. The inner loop stabilizes the plant dynamics, allowing the outer loop to focus on tracking the position reference and rejecting disturbances.

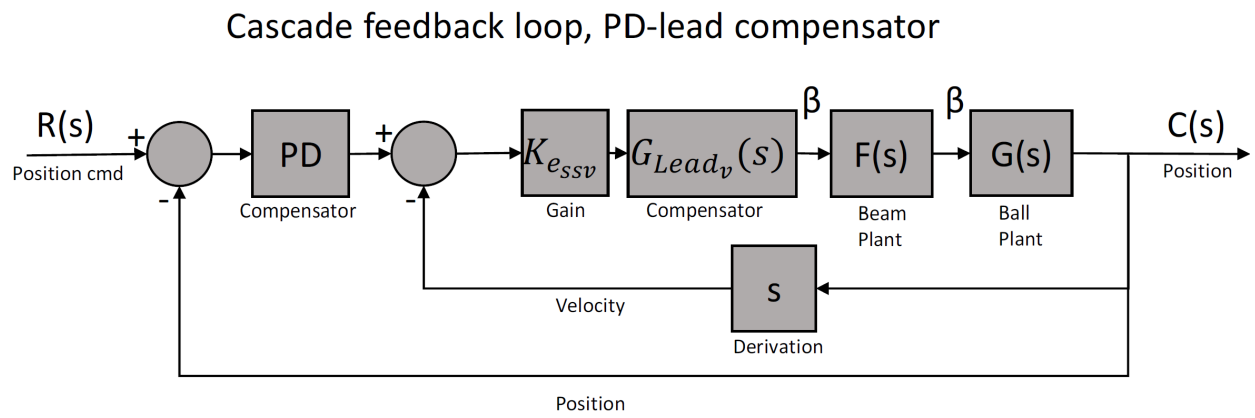

Figure 2.8: Cascade feedback loop, PD- lead compensator

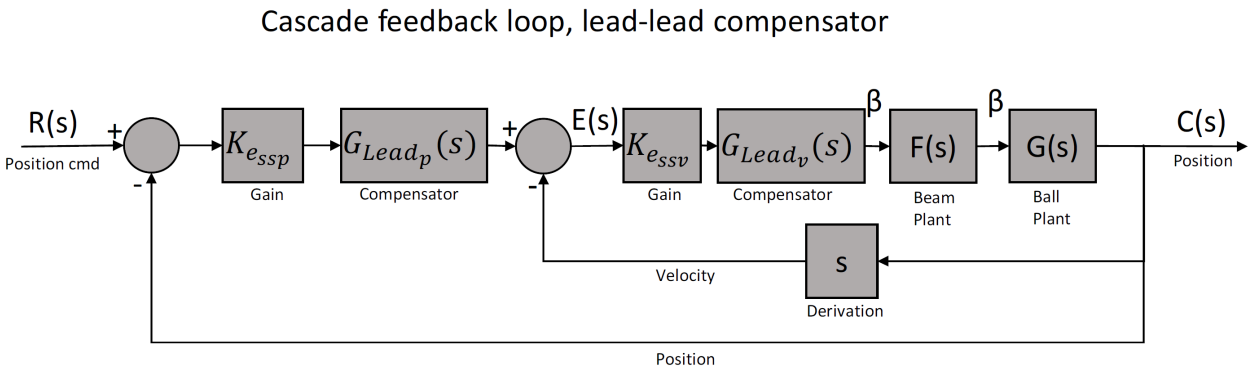

Figure 2.9: Cascade feedback loop, lead-lead compensator

Chapter 3

Methods in Design of Mechanical Systems and Electrical Systems

3.1 Target Specification

This project has not been previously operational, there is a lack of existing data to link to this project. To establish target specifications, relevant information was gathered from comparable projects [52], along with qualified guesses and calculations. These values are desired, but there will be room for the best possible result to be achieved.
Desired target specifications based on data:

- Steady state error <15%.
- Overshoot <20 %.
- Settling time is under 7 seconds regardless however the ball is on the beam.

Based on the machine not being able to create any dangerous situations "Any dangerous axis movement must be prevented" see Fig. B.1. Therefore, the machine must not be able to damage the environment or people:

- That there must be a max/min angle ±20 (degrees from a horizontal).
- Max angular acceleration of system is set to 27.5094 $[\frac{rad}{s^2}]$ F.3 With regard to safety. Max acceleration at the point of the ball shall not be beyond 9.81$[\frac{m}{s^2}]$ vertically.
- The beam shall not be harmful, it must be considered based on force max force 27.4040$[Nm]$ [45](ISO) and see Tbl. C.1.2.
- A table tennis ball or ping pong ball should be used for this experiment, as they are considered safe in case of accidental falls. These balls are chosen due to their lightweight nature, minimizing the risk of injury or damage if they happen to fall off during the experiment.
- Max speed of the tip is set to 1.2099$[\frac{m}{s}]$ [45](ISO) se table C.1.2.

3.2 System Specification

Specifications in the system				
Type	Weight	Dimensions	Material	Unit
Length beam roll area		613 (Balluff C.1)		mm
Length beam		710		mm
Table tennis ball ⋆ ⋆ ⋆	2.75 ± 0.3	40.1 - 40.15	ABS-plastic	g/mm
Center mount	42.49	68x66.8x56	Onyx	g/mm
Sensor mount	29.48	73.6x71.42x31	Onyx	g/mm
Cable holder 4 pieces	0.49	38x20x3	Onyx	g/mm
Dummy end	11.65	40x38x40	Onyx	g/mm
Dummy end cover	2.61	38x20x7	Onyx	g/mm
Tube 2 pcs	49.7	8x6x710	Carbon	g/mm
Base on axle	18	25.95x39.8x30	Onyx	g/mm
Alu side plate	41.73	96x55x30	EN-AW 3003	g/mm
Side plate sensor mount	1.55	16.25x16x3	EN-AW 3003	g/mm
Balluff sensor	53.327	See table:C.1		g/mm

Table 3.1: Table of specifications in the system.

3.3 Setup Ball on Beam

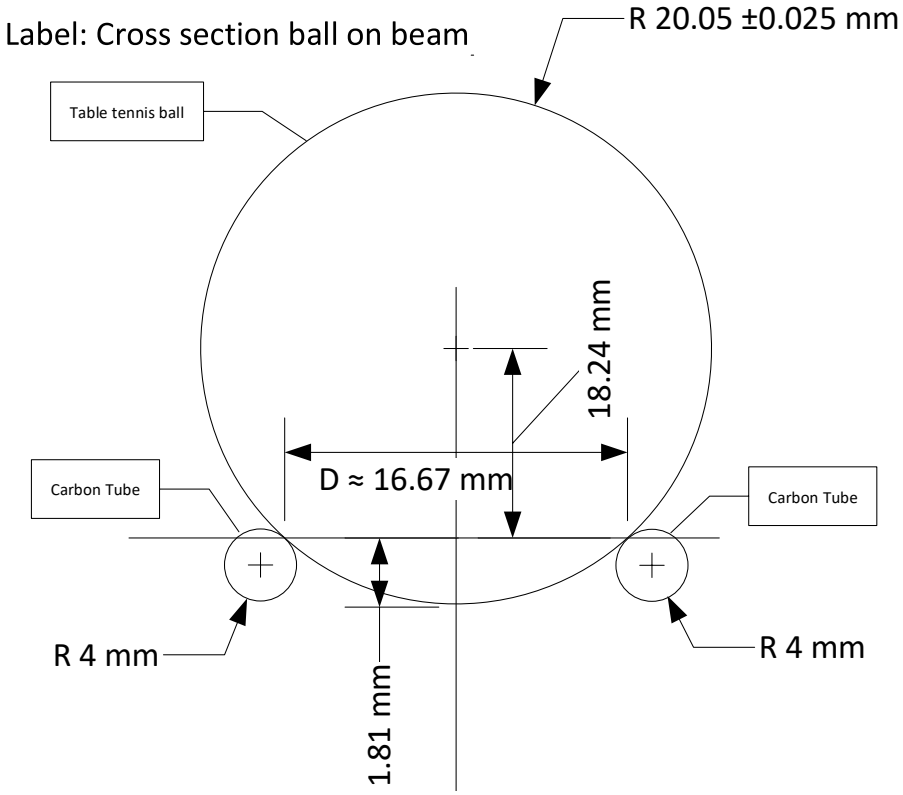

Figure 3.1: Cross section ball on beam

3.4 Mock-up, First Testing

Due to a delay in the receiving of CtrlX, the emphasis was on understanding the theoretical principles and specifications for controlling the system. It was decided to construct a mock-up in order to gain early experience with controlling the ball on beam, which will be useful in determining the appropriate methods for controlling the system.

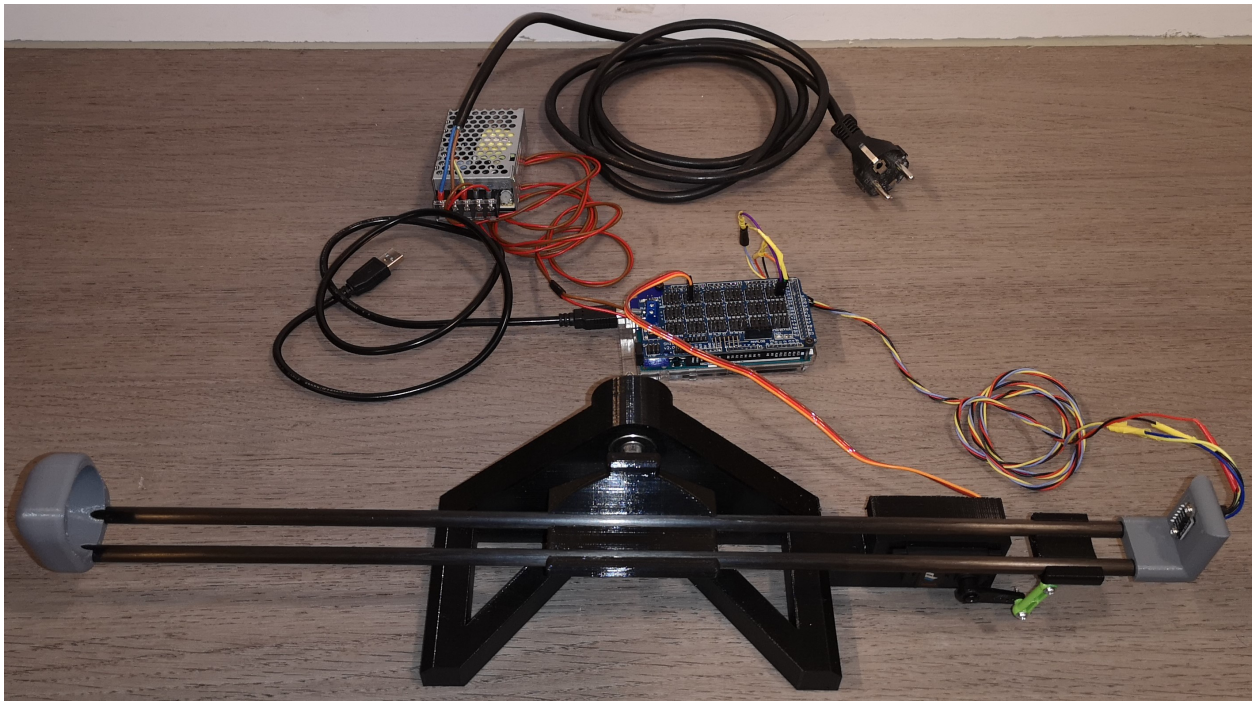

Figure 3.2: Overview mock-up

The first task for checking the ball on beam was used to 3D print the structure as shown in image Fig. 3.2. These were constructed through Solidworks. Measurements and dimensions are described under Tbl. 3.1 for the beam and ball. A servo with 180° with a resolution of 1° (Simulink set resolution) was used [8]. To control the servo, the sensor shield [27] was mounted on top of an Arduino mega 2560 R3 [2]. To measure the distance to the ball, the time-of-flight sensor, VL53L0X, was chosen [50] based on price, size, and precision/accuracy. The servo draws some current, so a separate power supply must be used [21].

3.5 Mock-up, beam parts made in Solidworks

The beam consists of four parts, the carbon tubes themselves, center attachment, end attachment and sensor end attachment. All the parts were drawn in Solidworks according to the desired length $468.8mm$ relation to the length and placement of the ball see Fig. 3.1. Parts were printed using PETG. After the parts were 3d printed, the parts were weighed to find the exact mass so that the model could be inserted into Solidworks.

3.6 Assemble the parts Mock-up

The beam is made from both 3d printed parts and carbon tubes that were bought at [5] and cut to the desired length. The parts that were 3d printed were saved as an STL file in Solidworks. Then they were loaded into Cura [55] and printed with standard settings with quality 0.2mm, support, build plate adhesion, 20 % infill and extruder temperature $240°C$ and plate temperature $70°C$. The parts were printed with printer $CrealityCR-10SPro$.

The base is placed on the shaft and the attachment was mounted on the beam so that it can be mounted quickly on and off with adjustment in the z-axis (up/down). Adjustment of the z-axis may be possible if one wishes to move the rotation from the center of the ball to the contact point of the ball.

3.7 Mock-up, Analysis of Ball Rolling down Rail

To evaluate that the differential equations are "correct" for this system, it was checked by measuring the ball rolling down the inclined plane. The plane of the plane was set to 5° and with the help of eq. 2.26 acceleration, speed and position were calculated analytically using classical mechanics eq. 2.4 and eq. 2.5. From there, this was plotted in Matlab and compared with measurement data obtained from the Tracker [53] that uses filmed video of the ball rolling down. From this it turns out that the equations within the measurement range agree on approximate with the physical system Fig. F.5, Fig. F.6, Fig. F.7, Fig. F.8 and Fig. F.9. Regarding the position signal it was noisy so a good filter is important for feedback control.

3.8 Assemble the parts final version for CtrlX

Experience from the mock-up model showed that adjustment options on the sensor and on the rotation axis can be profitable to have that option. Therefore, the final solution for the beam was designed with several adjustment options. Values for length and width were designed from experience gained in the mock-up model. There, the width of the beam was retained but the length was changed see Tbl. 3.1. The length of the beam was chosen based on the maximum load in the form of the ball rolling on the beam.

The parts for the beam consist of: carbon tube, Balluff sensor holder, base for beam, base on axle, cable holder, cover, end bracket, aluminum side plate, and side plate sensor mount. All the parts were 3D printed except for the side plates and carbon tubes. The 3d printing was done with the help of Mechatronics laboratory [54]
These were printed in Onyx on Markforged X7 [18]. This printing method was chosen because the quality of the print is higher compared to printing on commercial printers. With high strength [23] and visual appearance, it shows a professional result. With an infill of 60 %. Layer thickness is 100 μm.

All figures regarding mechanical parts were visualized using SolidWorks visualization [49].

3.8.1 Base on Servo Axle

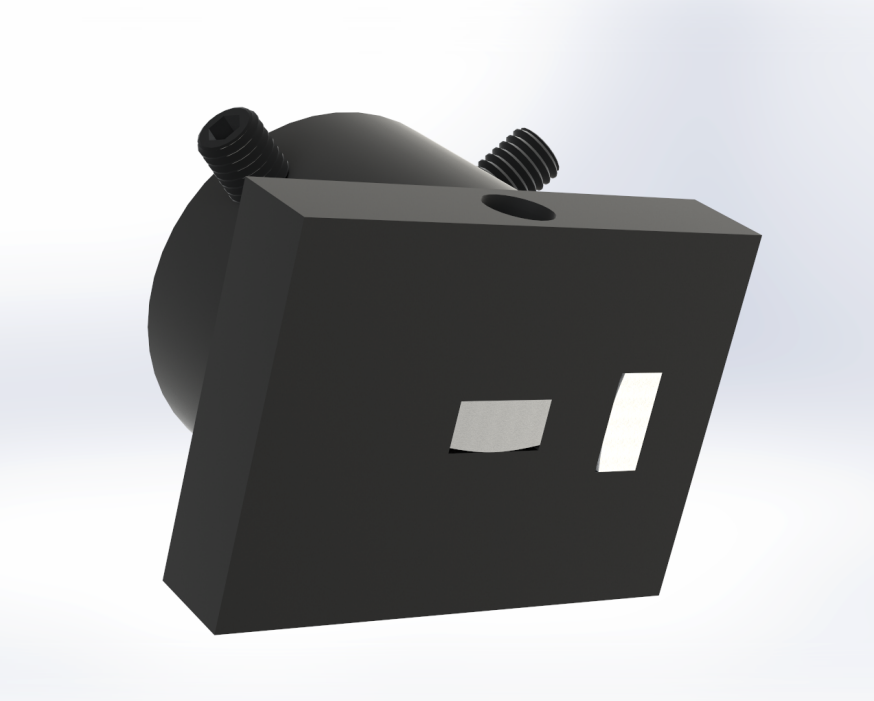

Figure 3.3: Base on axle

The base of the shaft Fig. 3.3 that is attached to the shaft of the servo is fixed with two sets of screws. In order to be able to adjust the beam to the axis of the serov, a nut is fitted in the center of the base of the shaft Fig. 3.3 and to attach the base to the shaft, a nut is also fitted in the side to ensure a solid attachment. Drawings and files for the part can be accessed at the provided link [26]. This also includes parts that are mentioned further below.

3.8.2 Base for Beam

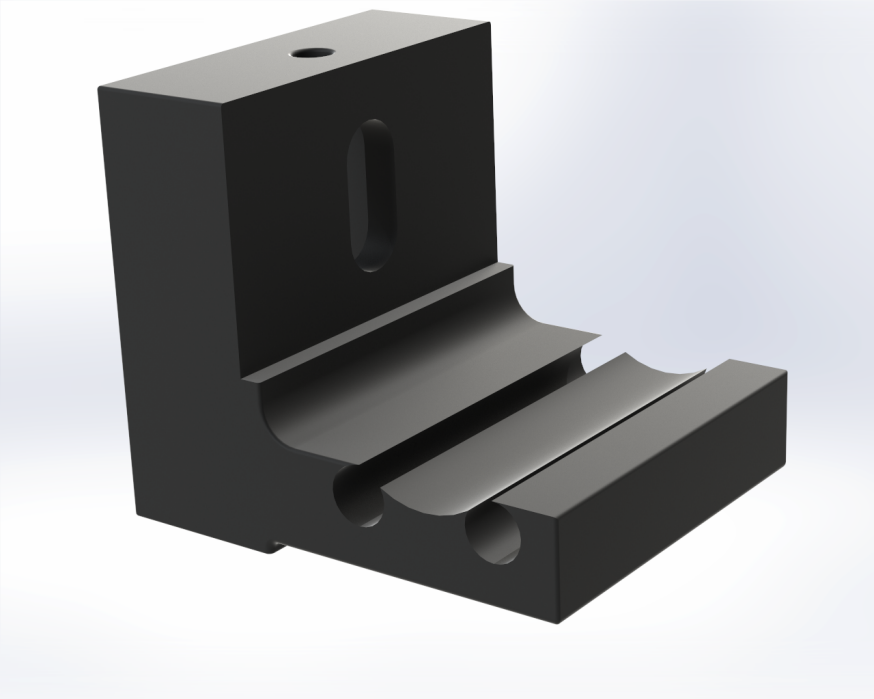

Figure 3.4: Base for beam

The base for the beam hold's the carbon tube in place and the base for the beam see Fig. 3.4 is mounted on the base for the axle 3.3. On the base for the beam part, the plate is also mounted which gives information about the adjustment height for the servo axis see Fig. 3.11.

3.8.3 Carbon Tube

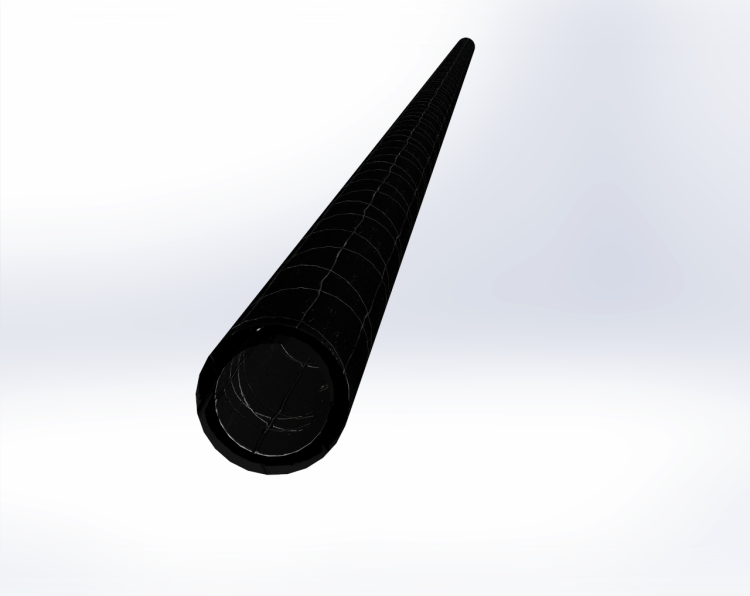

Figure 3.5: Carbon tube

The carbon tubes are shown in Fig. 3.5 and were adjusted in length according to table 3.1. The purpose of carbon tubes is that they have a surface that creates low friction and has a low specific weight in relation to the size. This is beneficial for keeping the moment of inertia low. They were bought in from Elefun [5].

3.8.4 Cable Holder

Figure 3.6: Cable holder

The cable holder (see Fig. 3.6) is designed to hold the cable that connects to the Balluff sensor (see Fig. 3.9). The holder is made up of four pieces, as shown in Fig. 3.11, and serves the dual purpose of supporting the beam and ensuring that the cable remains stable and stationary.

3.8.5 End Bracket

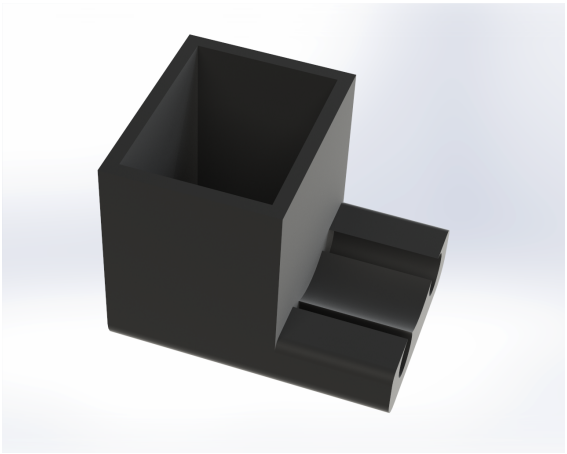

Figure 3.7: End bracket

Figure 3.8: Cover lid

The end bracket (see Fig. 3.7) can stop the ball from rolling outside the specified area. It must also have counter-balancing weights on it so that the beam is as balanced for equal movement dynamics. The lid is shown in Fig. 3.8.

3.8.6 Balluff Sensor

Figure 3.9: Balluff sensor [3]

The Balluff measurement sensor, Fig. 3.9), measures the distance the ball rolls or moves on the beam. The distance sensor is mounted in the Balluff Sensor holder, Fig. 3.10, where it is also calibrated based on the laser that radiates from the sensor see table for specification Tbl. C.1 and the data sheet [3].

3.8.7 Balluff Sensor Holder

Figure 3.10: Balluff sensor holder

The sensor holder (see Fig. 3.10) securely holds the Balluff sensor (see Fig. 3.9) and is mounted on the beam, as demonstrated in Fig. 3.11. The holder allows for horizontal adjustment and can be angled to ensure that it accurately detects the tabel tennis ball (see Tab. 3.1). For full assembely see Fig. 3.11

3.8.8 Full Assembly

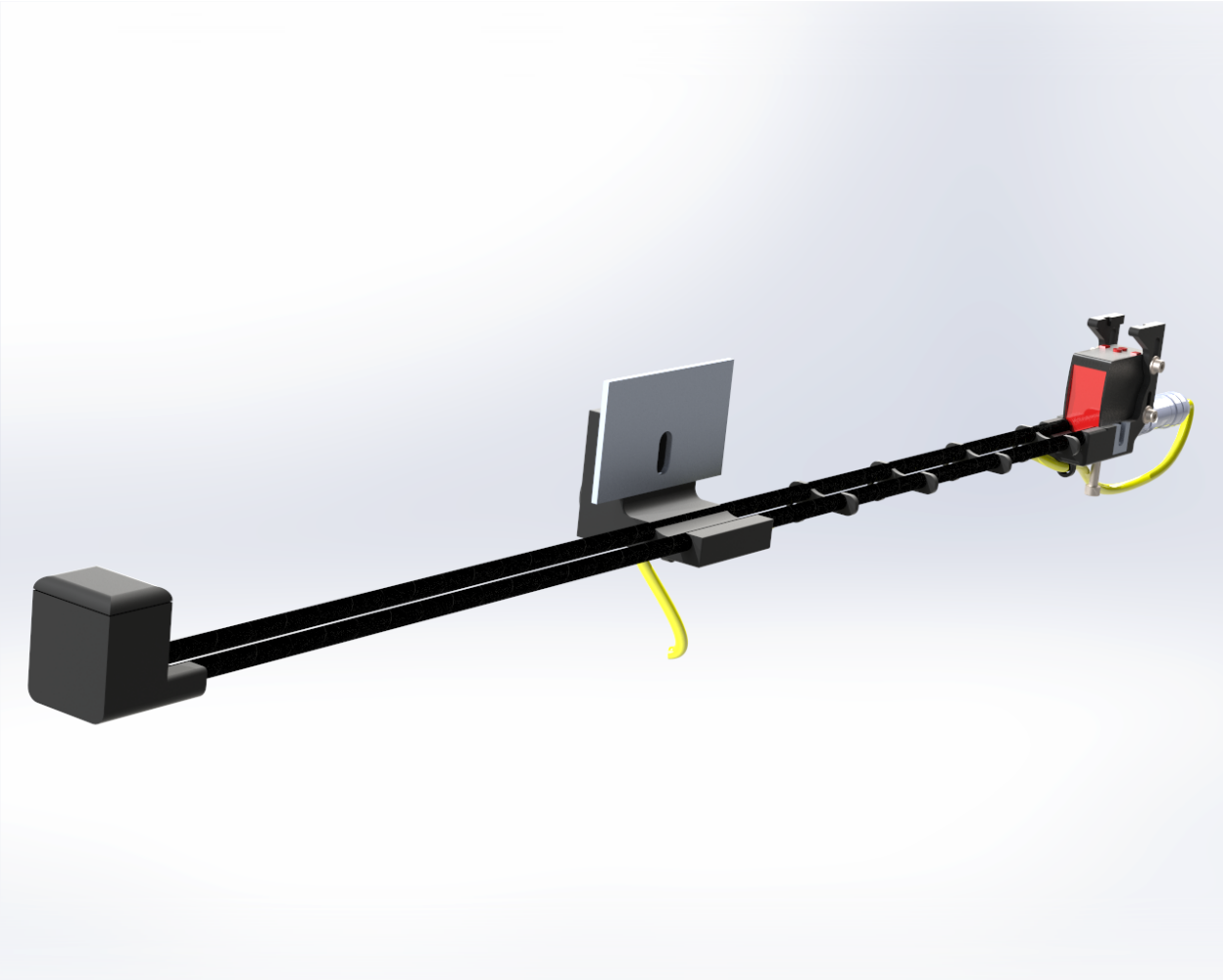

Figure 3.11: Full assembly

The entire assembly is shown in Fig. 3.11. It consists of all parts fully assembled without ball. It has mounted on M12 connector with cable for visual effect [17]. The part list for assembly parts see table C.1.3

3.9 Hardware

(a) Left

(b) Right

Figure 3.12: ctrlX Drive w/Core, servo motor and accessories

3.9.1 CtrlX CORE, Drive and Motor

The project is based on the ctrlX Automation promoter set from Bosch Rexroth with part no: R911398609, Fig 3.12. It consists of a drive with built-in ctrlX Core, a servo drive, two power supplies, and a contactor with on/off buttons. The cabling between the components is done according to the datasheet R911392532 [40]. Main components:

- Bosch Rexroth Servo motor, Mnr:914000211, Type:MSN03-B0BYN-HMSH1-NNNNN-NN [43]
- Bosch Rexroth ctrlX Drive w/Core, Controller: Mnr:911410005, Type: XMS1-W0023AN-S0314N2NNNN0NN, Power section, Mnr:911400865, Type:PMS1-W0023AN [40]
- RS PRO Power supply, Mnr:136-8321 Type:Ac-DC 240W-48V, 240VAC input [46]
- Phoenix Contact Power supply, Mnr:2866271 Type:MINI-PS-48-60DC/24DC/1 [25]
- Eaton Contactor, Mnr:276698 DILM9-10, Type:XTCEC007B10 [9]

3.9.2 Balluff Photoelectric Distance Sensor

The desired sensor for use on this project was the SICK OD2000 which has a resolution of 16 bit, repeatability of 20 [μm[, range of 100-600 [mm], output time $\geq$ 0.1333 [ms] and Response time $\geq$ 0.533 [ms] see [47].
After talks with Bosch Rexroth Langhus the Balluff sensor was provided by Bosch Rexroth Langhus based on what they had in store.

The sensor, Balluff BOD001N [3], was connected to the ctrlX Drive, XG31(pin 5) to the cable shield and XG31(pin 10) to sensor Q_A (pin 2) output. Pin 1 ($+U_B$) from the sensor was connected to the drive +24VDC and pin 3 ($-U_B$) was connected to the drive 0VDC. The sensor has to be calibrated to the right distance. This was done by pressing the Q_A button for >3s with the ball at the farthest position. The ball was moved to 10cm from the sensor and the Q_A button was pressed again. The range of the sensor was now the length of the ball's motion.

3.9.3 Elecrical circuit diagram Balluff sensor to ctrlX

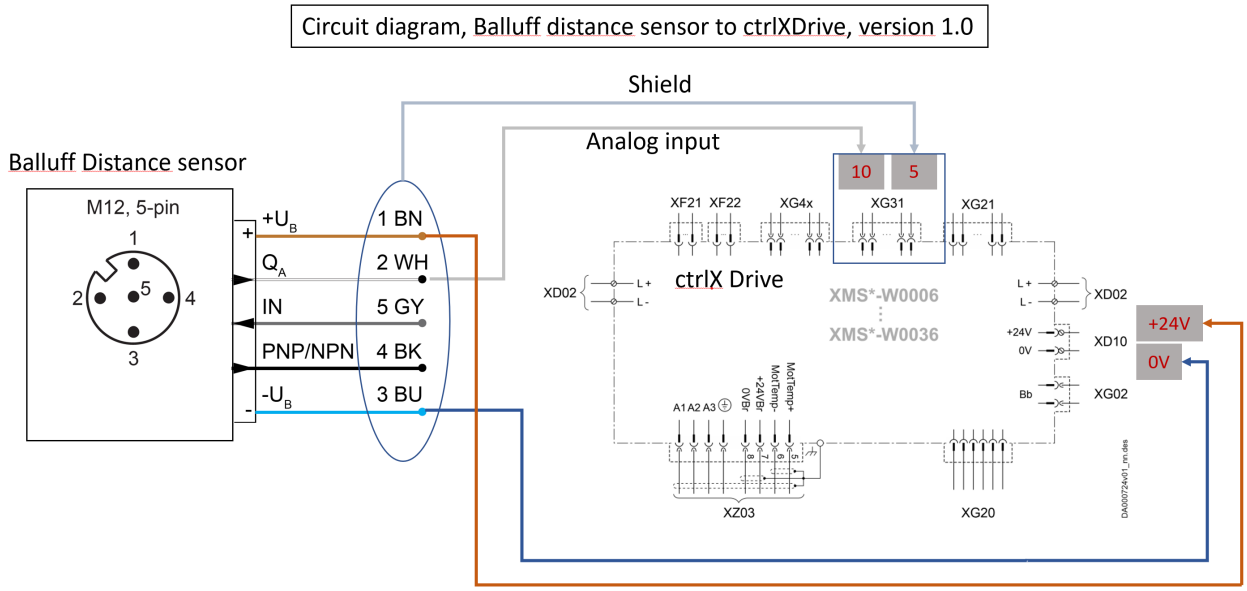

Figure 3.13: Circuit diagram for connection of Balluff sensor to ctrlX Drive

The circuit diagram for the drive is found in the Bosch Rexroth documentation, Drive system manual [29] and for the servo in the MS2S Synchronous Servomotors Operating manual [43].

3.9.4 Complete System

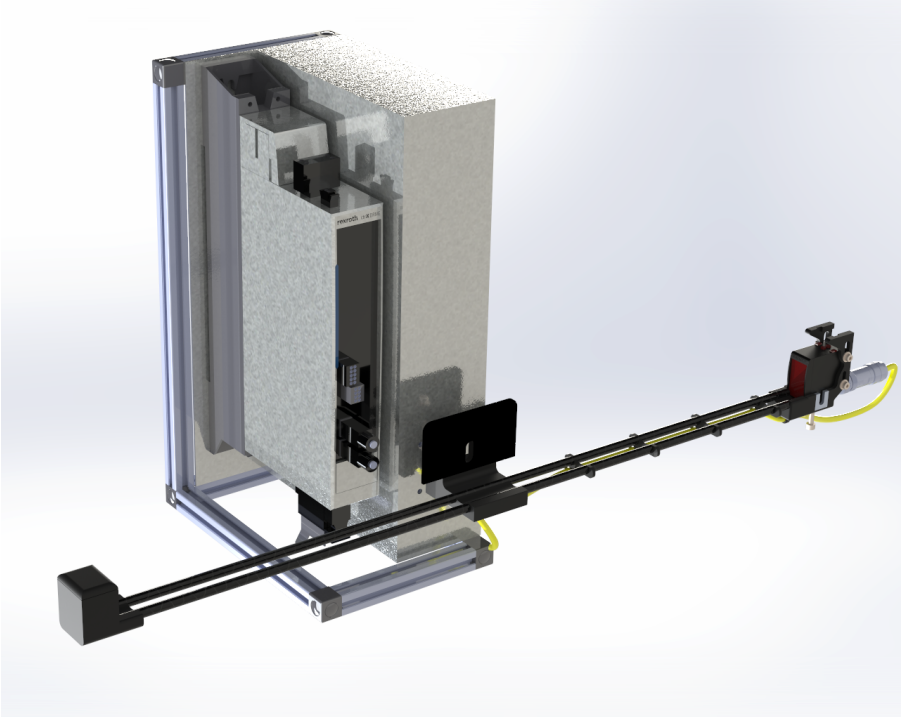

Figure 3.14: Complete system modeled in Solidworks

The system is constructed to be placed on the edge of a table or raised 20 [cm] for the beam to move the complete ±20[%]

Chapter 4

Methods in Design of Software Systems

4.1 Control System

The first priority is to create a control system with PID and get it operational.
The objective of managing and controlling the ball's movement on the beam is to create a control based on theoretical calculations and the realization of these calculations into real-life movements.

The drive and motor is a system package made by Bosch Rexroth with an already-defined control structure and optimization routines. The corresponding mathematical model of this system is estimated by using a step input and trace to establish the system identification. The following transfer function is combined with the ball and beam transfer function eq.(2.33)

Given that the ball-on-beam system is inherently unstable, any noise in the feedback signal can cause significant oscillations or steady-state errors that exceed the desired level of precision and accuracy. Implementing low-pass filters in the control loop is expected.

The cascade loop design consists of an inner loop with velocity feedback and an outer loop with position feedback. The inner loop must be tuned to be at least 3 times faster than the outer for good regulation. By introducing a velocity loop the control is less responsive to noise and better overall control is achieved together with the outer position loop. A drawback is the derivation of the actual feedback signal to get the velocity feedback. This can introduce noise so a low-pass filter will be necessary to improve the signal.

Control system development:

1. Calculation of ball transfer function
2. System identification of drive/ motor/ beam system to get a transfer function
3. Simulation in Matlab/ Simulink to get a bode plot of the system
4. Design of P controller and lead or lag compensators to get the desired overshoot and steady state margin
5. Apply these to the Matlab/ Simulink model
6. Find the break frequency of the low-pass filter and design it in Matlab for feedback filtering
7. Program the PLC to mimic the Matlab model
8. Test actual response and save traces.
9. Tune the controller to the best performance
10. Evaluate the simulated values with the physical values

4.1.1 System identification

System identification is done with a complete mechanical setup to trace the actual response of the system. The beam is in a horizontal position and a step input of one degree is given. The internal acceleration and velocity limits of the drive must be set to a value that is used during normal operation. The trace is saved as a .csv file directly from the PLC trace. Plotting this in matlab makes it possible to find the percentage overshoot, peak time, and gain. A transfer function was made with the eq. 2.52 to eq. 2.59

4.1.2 Matlab

With an identified system and a defined ball model further system response is possible to simulate in matlab/ simulink. These simulations make it possible to get a bode plot of the open loop response to construct a lead compensator for the velocity loop. The position loop gets a PD- controller so that it is easy to manipulate from the HMI and tune with the Ziegler- Nichols method in subsection 2.4.1.

Specification of velocity loop requirements.
The required performance of the velocity loop is

- Rise time, T_{rvel}=0.3s
- Percent overshoot, OS_{vel}<2%
- Settling time, T_{svel}<1.5s
- Steady state error, e_{ssvel}<0.1

Calculating $\Phi_{desired}$ and ω_{bw} from subsection 2.4.2 makes it possible to construct the lead compensator

Required gain to reach desired steady-state error in the velocity loop.
To find the required gain, K, so that the system has a steady state error within the specification, a final value calculation was done using eq. 2.67 and eq. 5.7. The reference signal was a ramp input.

Lead compensator calculation.
A new bodeplot, with the gain K, was added to the original plot. By analyzing these plots, variables a and T, were found as described in subsection 2.4.2 to construct the H(s) transfer function.

Lead- compensated bode plot

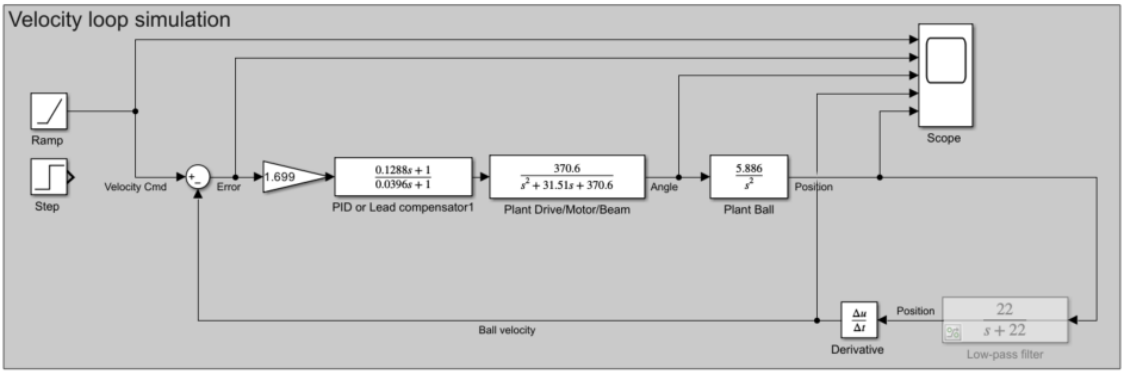

Figure 4.1: Velocity loop simulation in simulink

The velocity loop Fig. 4.1 displays the layout of the simulated system. The system response was simulated to see if the lead compensator was correct. A ramp input with slope = 1 was used to trace the input versus the feedback velocity.

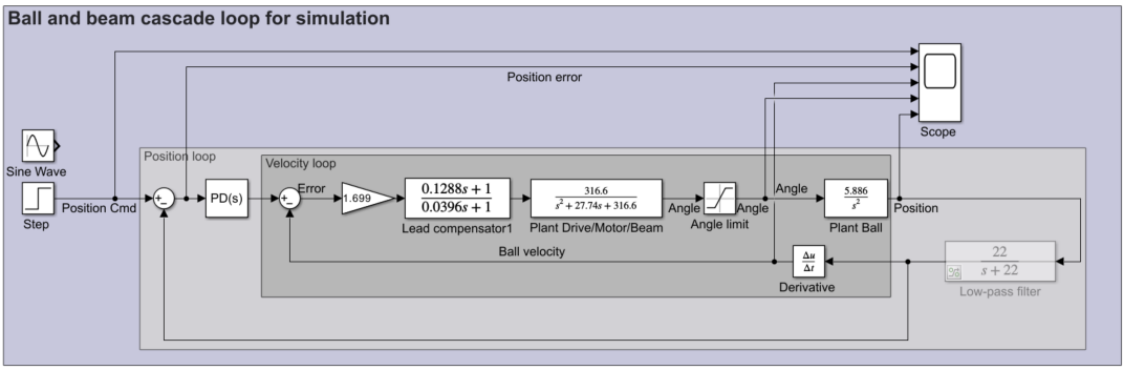

Figure 4.2: Cascade loop, position and Velocity loop simulation in simulink

Adding a second loop on the velocity loop makes the cascade loop Fig. 4.2. Tuning this loop required setting the system in a steady oscillating state by only using kp in the pid controller. The trace was saved and the values needed for the Ziegler- Nichols method, subsection 2.4.1, were extracted from the trace. The PD- gains were used together with the found values and the resulting values were set in the PID controller. Further tuning was done by setting different position commands, analyzing the plots, and adjusting the kp and kd values. The tuning was finished when the performance was within the specified ranges.

Filtering of the velocity loop feedback signal.
An Infinite Impulse Response filter was added to the position signal for the velocity loop feedback to filter out high-frequency noise according to subsection 2.4.3. The derivation of the signal is sensitive to fast changes since it is the rate of change that is calculated. The filter cutoff frequency was higher than the system bandwidth frequency. Adding this filter caused a change in dynamics due to a phase delay introduced by the filter.

4.2 Model in Simscape

A ball and beam simulation can provide valuable information about the system. In this case, it is done in 3D and the reason for this is that the ball that rolls does not lie on the tip of the ball, but rolls a little on the side with 2 contact surfaces. A 3D model also provides good visual feed back if the ball starts to bounce on the beam and can then jump off more easily.

4.2.1 Install Add-ins to SolidWorks and enable it for Matlab/Simscape

To be able to test the model's dynamics with a rolling ball, a model was made. The parts were drawn in Solidworks and are shown in Fig. 3.11. The first step that must be carried out after the parts have been drawn is to insert the "Simscape Multibody Link Plugin into SolidWorks". This is entered as an add-in in Solidworks.
First step of the installation "Install the Simscape Multibody Link Plugin", [20], and follow the installation guide that is described.
Then open Solidworks and "Enable Simscape Multibody Link Plugin in SolidWorks" which is performed with this guide [19]. The link also describes how the part drawn in Solidworks is transferred to Simscape. It is now possible to continue from Simscape and adapt the model further.

4.2.2 Solver

The setup of the model for the algebraic solution consists of Solver configuration and Solver selection (configurator Parameters). The solver configurator was automatically created and connected to the model after it was created in Simscape and comes with default settings. The configuration of the Configurator Parameters is set to type: Variable-step and solver: ode45.

4.2.3 Optimizing the Model

Now the model, after it has been exported to Simscape, can be opened in the Matlab Command Window using the command

```
smimport('full_ass_V7_Ball_and_Beam.xml')
```

Now a Simscape Multibody model is built from the CAD files based on the xml file that describes the connection between the parts.

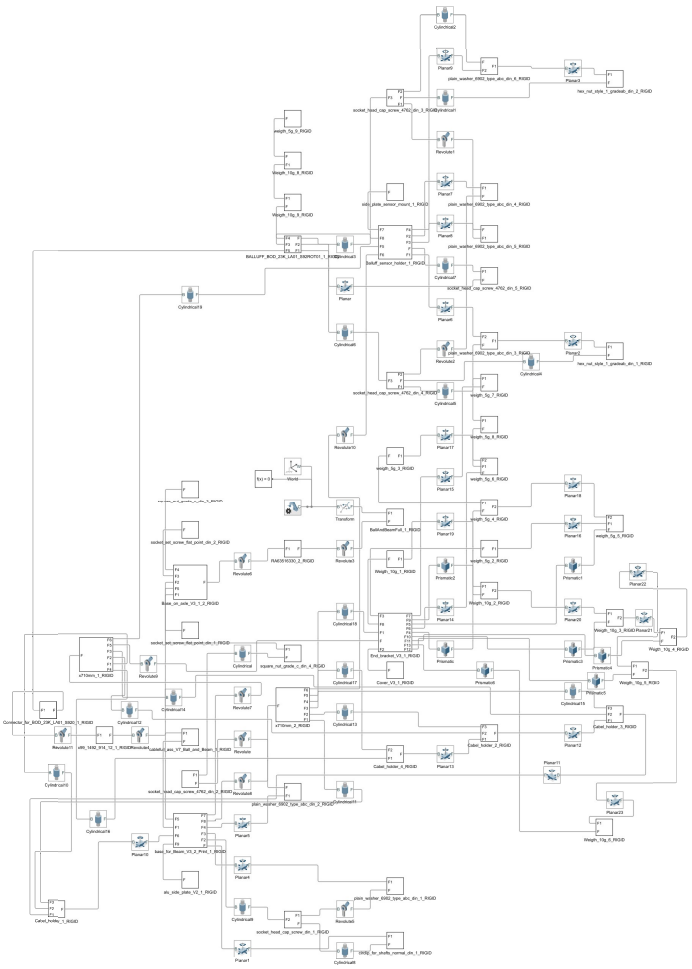

Figure 4.3: Raw model in Simscape

As shown, the model quickly becomes complex, see Fig. 4.3 and therefore it would be wise to systematize it.

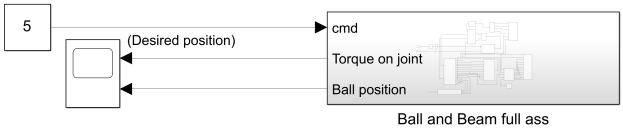

Figure 4.4: Top layer of the Simscape Ball and Beam

By systematizing the system, it will become more transparent. The most important signals are also extracted here see Fig. 4.4. In goes command and out goes the position of the ball on the beam and torque on the beam from the servo.

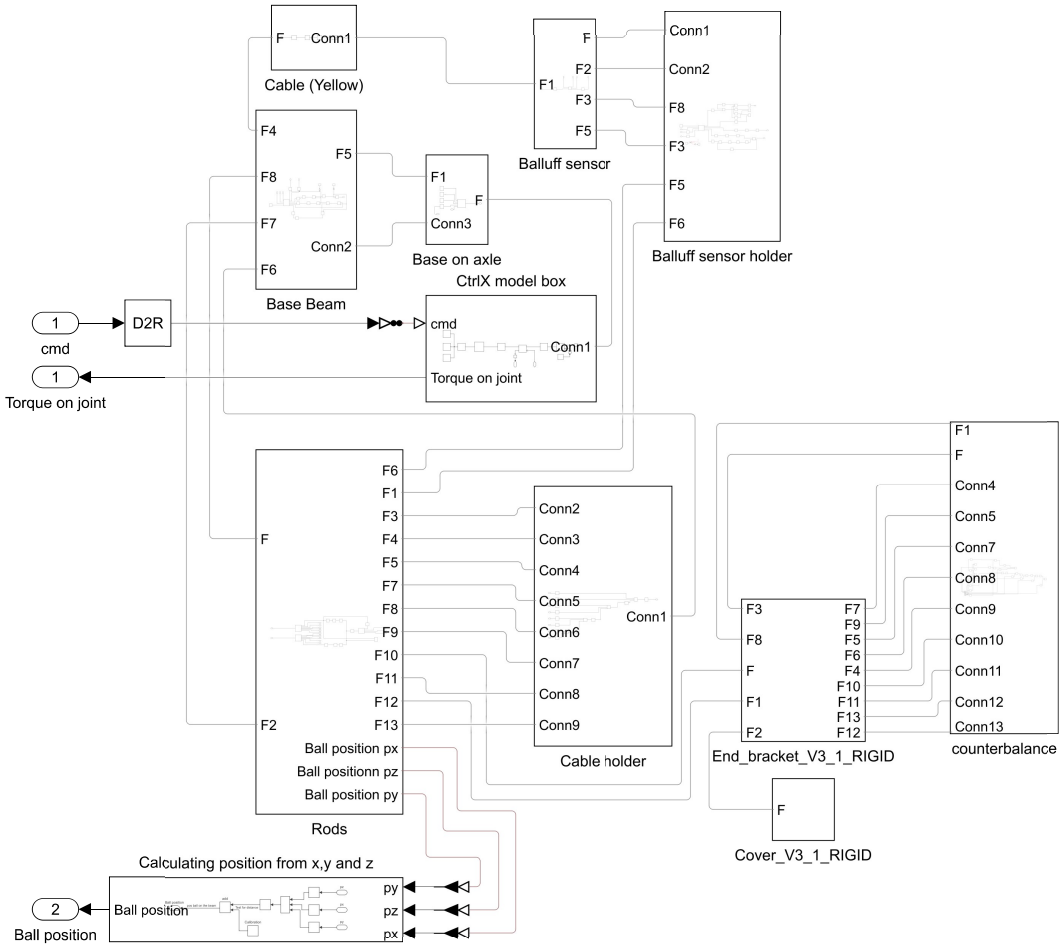

Figure 4.5: Second layer of the Simscape Ball and Beam

As shown in Fig. 4.5, it becomes more clear which systems are linked together with a description of what they do.

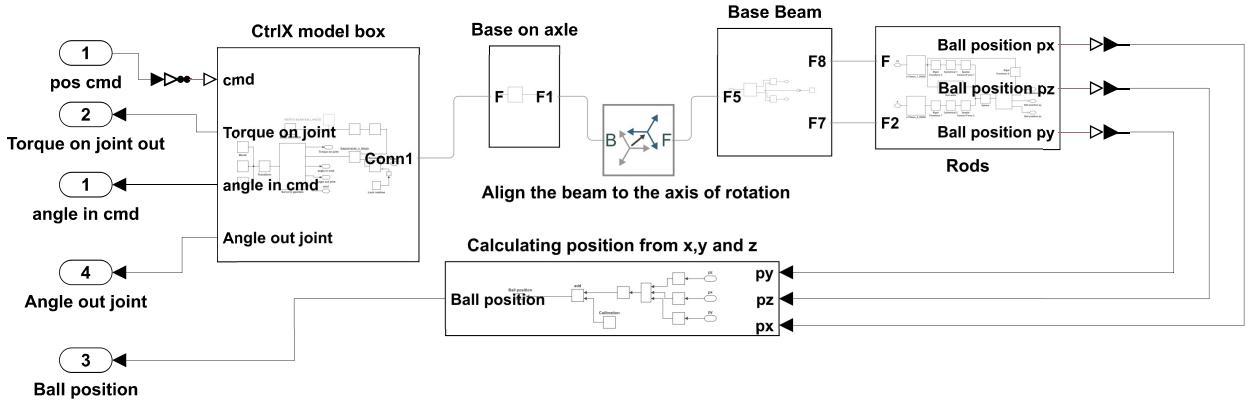

Figure 4.6: Simplification of second layer of the Simscape Ball and Beam

To optimize computer performance, it is crucial to simplify the model by removing unnecessary components that consume computing power. As demonstrated in Fig. 4.6, components that are not necessary were removed and instead modeled with a moment of inertia block. Further details on this process will be discussed at a later time.

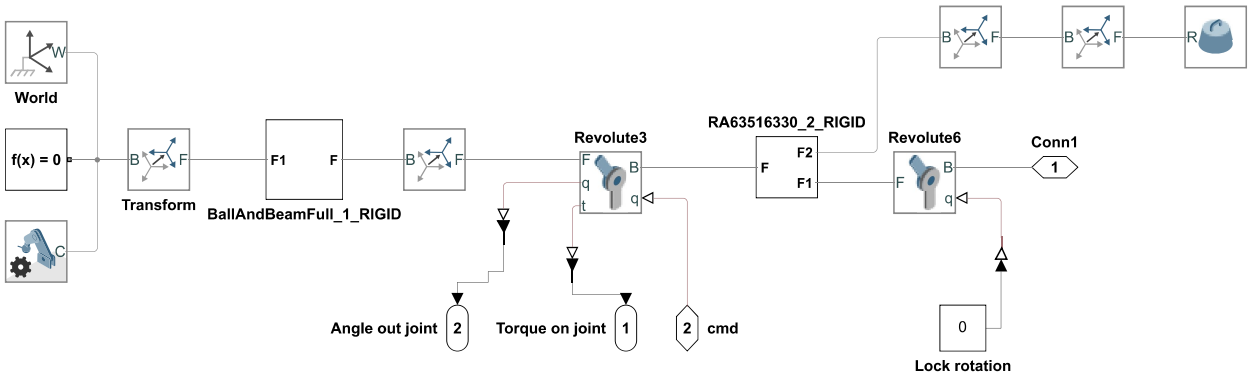

Figure 4.7: Rotation joint of the Simscape Ball and Beam

The "center" of the model in Fig. 4.7 is the servo itself which is oriented to the world frame (block Mechanism Configuration) where the direction of gravity is oriented to the Z axis which reflects the model's natural dynamics. The remaining parameters are set to default. Servo rotates through the shaft represented by joint Revolute 3, the shaft RA6351633_2_RIGID, and the servo housing BallAndBeamFull_1_RIGID. Into Revolute 3 goes the cmd that comes from the servo model. The servo model will further details on this process will be discussed at a later time. From this joint, the angle is sensed and the torque is sensed, whereupon the torque is sent back to the servo model.

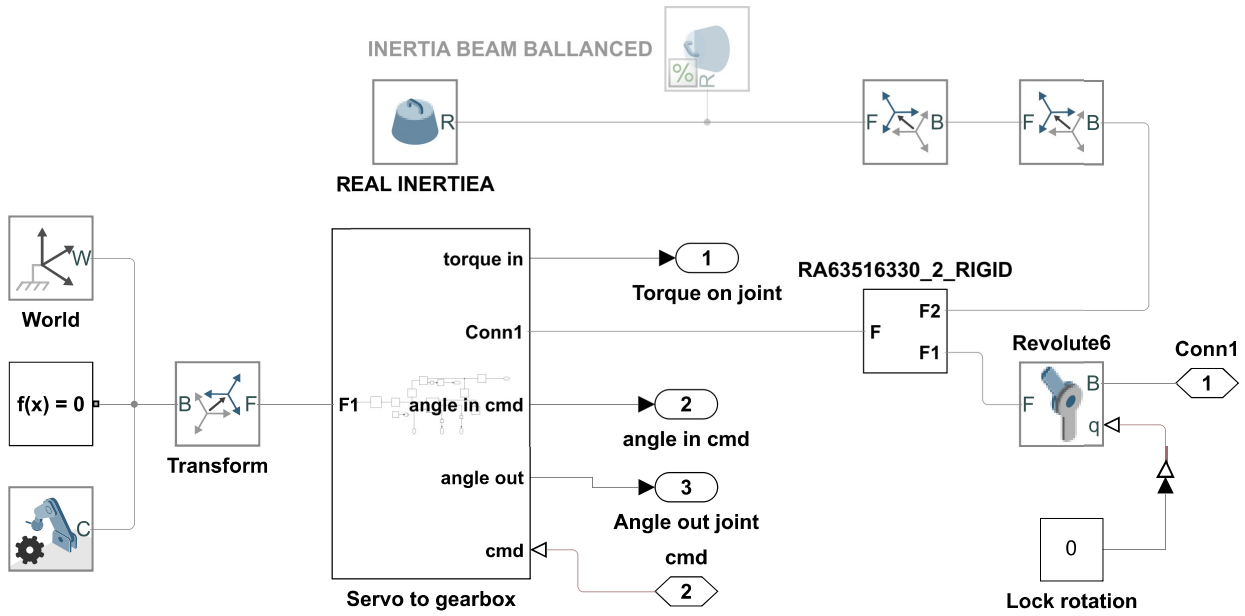

Figure 4.8: Rotation joint gearbox of the Simscape Ball and Beam

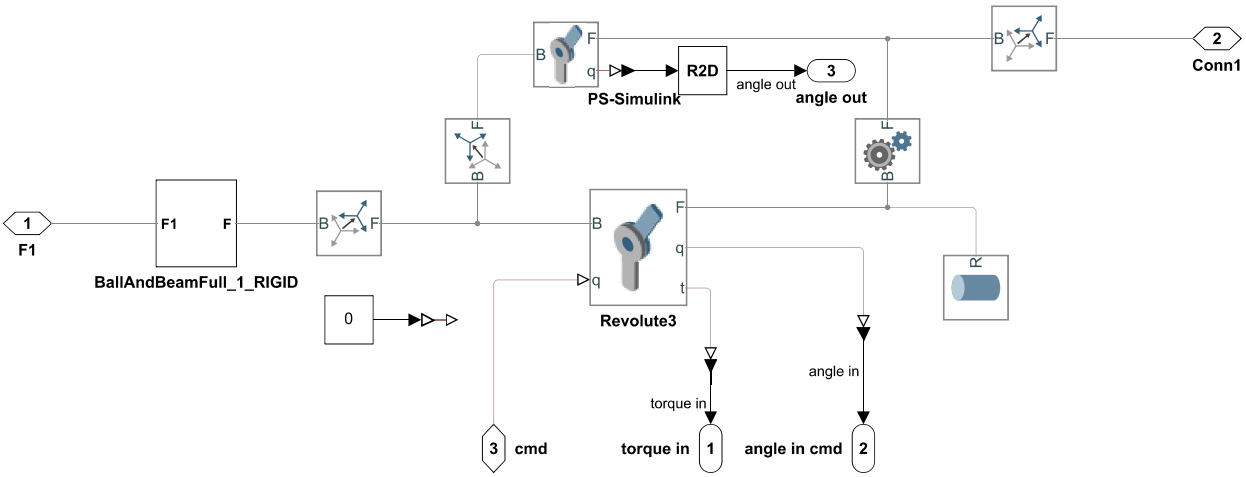

Figure 4.9: Servo to gearbox

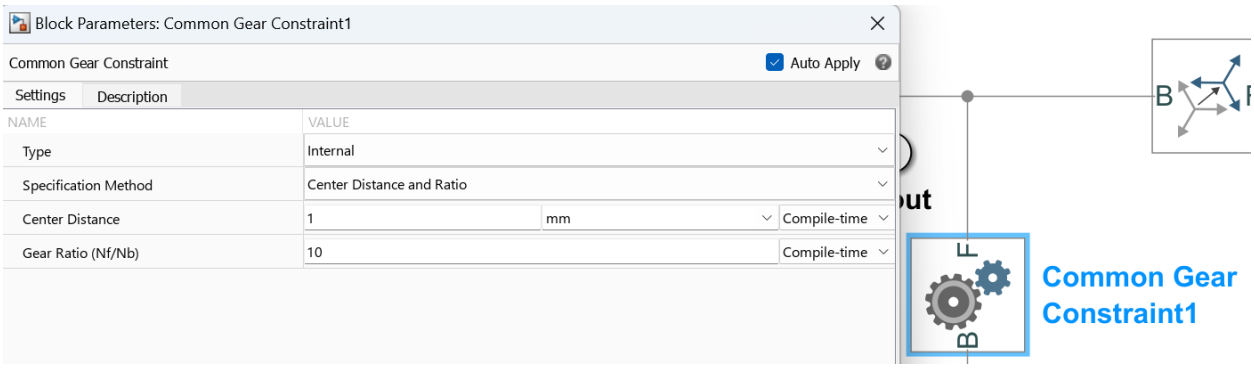

Figure 4.10: Common Gearbox

To incorporate a gearbox into the model, a new subsystem was added, as depicted in Fig. 4.8. The gearbox was constructed as an internal drive, as shown in Fig. 4.10, with parameters set for the gear ratio and diameter. It is crucial to note that the diameter of the gear must match that of Rigid Transform 6 ([1 0 0] [mm]), which is parallel to the common gear see Fig. 4.9. To facilitate proper operation, a revolute joint was inserted between the common gear and Rigid Transform 6 see Fig. 4.9, allowing the beam to rotate with the gearbox. From Revoulute the angle of the beam is to be measured.

Cmd into Revolute3 comes from the servo model. While details on this process will be addressed at a later time, the servo model plays a crucial role in enabling Revolute3 in Fig. 4.8 to function effectively. From this joint, the angle comes out and the torque comes out, whereupon the torque goes back to the servo model. Internally in Revolute3, the default parameters are taken from the xml file which in Matlab is called:

```
full_ass_V7_Ball_and_Beam_DataFile.m
```

This must be opened in a script and run from the same folder.

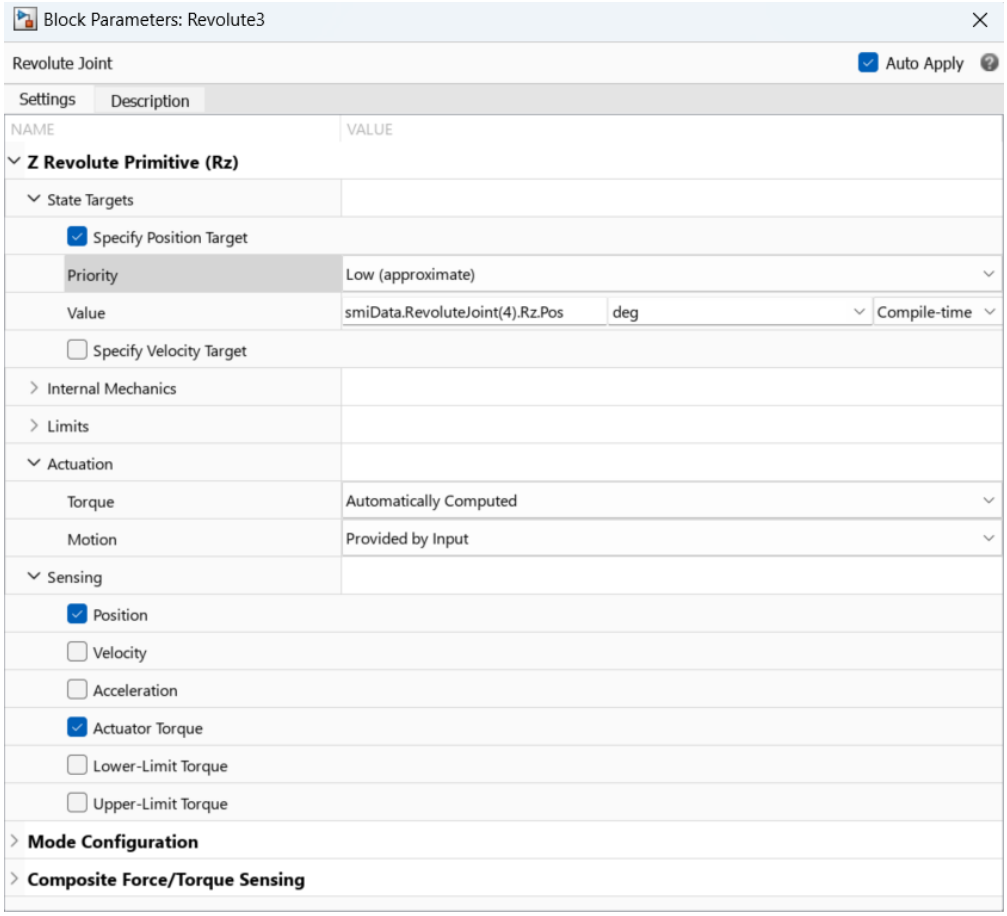

Figure 4.11: Revolute3

In Revolute 3 in Fig. 4.11 the following parameters were changed: sensing is changed to output position and actuator torque (Physical signal port that outputs the actuator torque acting on the joint primitive) and actuation where torque is set to automatically calculated and motion is provided by input.

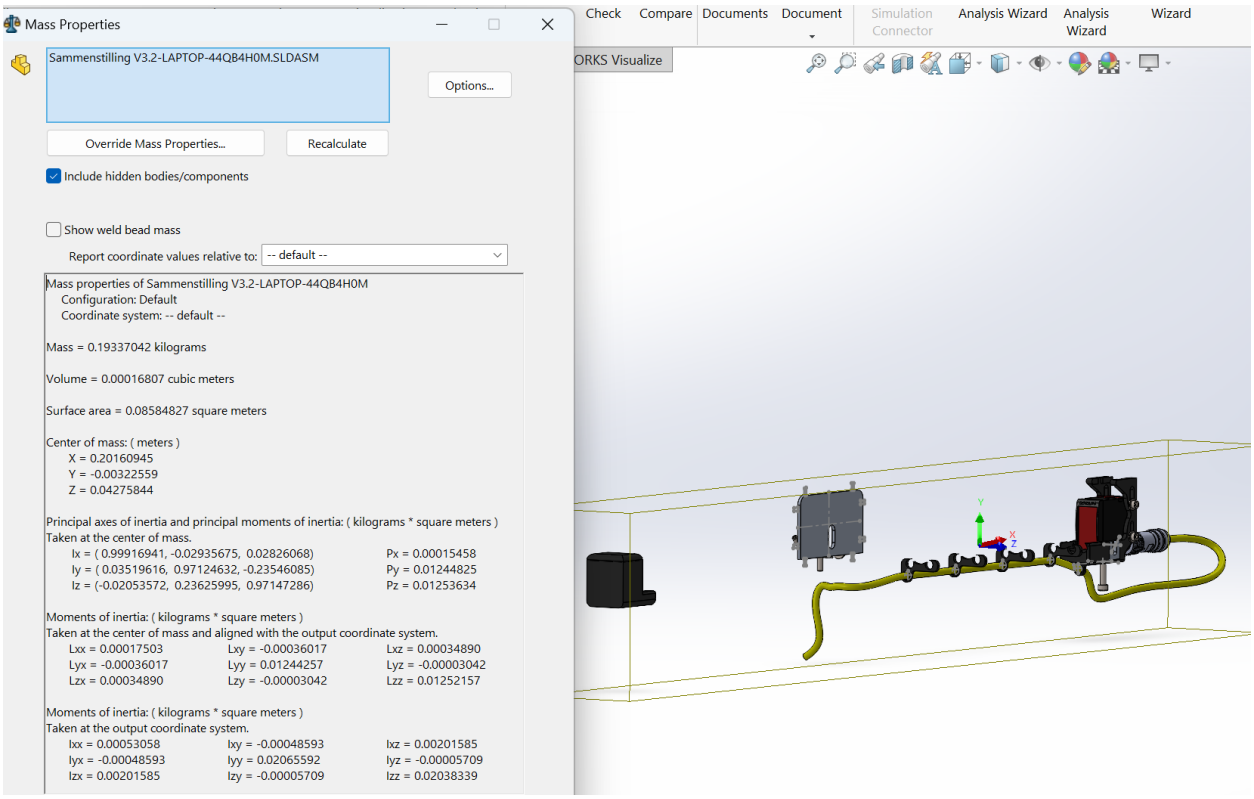

Figure 4.12: Correction moment of inertia

When the model was simplified many parameters from the model disappeared or were not used. Therefore, a moment of inertia was incorporated into the model, as depicted in Fig. 4.7 and Fig. 4.8, with the corresponding properties obtained from Fig. 4.12. These components that are represented are eg balluff sensor, holder, screws, washer, nuts, cable holder, cable and side plates. The moment of inertia was implemented from the Solidworks assembly.

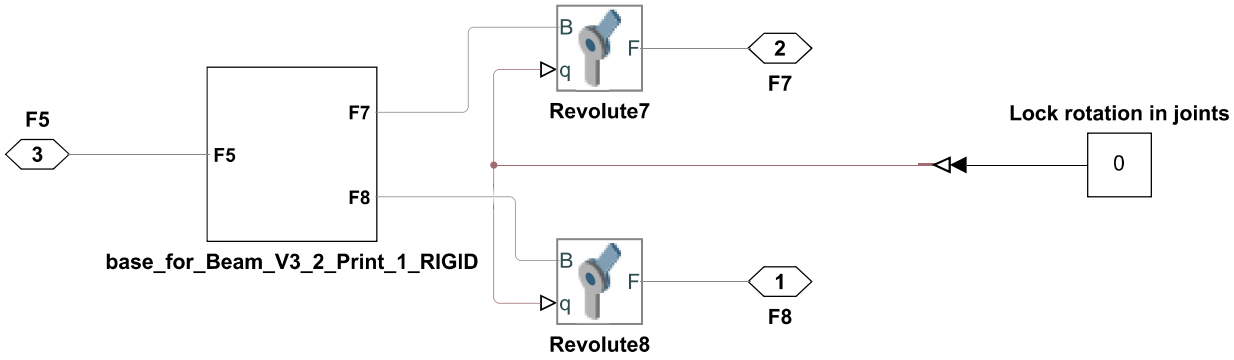

Figure 4.13: Locks for rotation in rods

Parts of the model are made as revolute, they must then be "locked" so that they do not rotate, see Fig. 4.13.

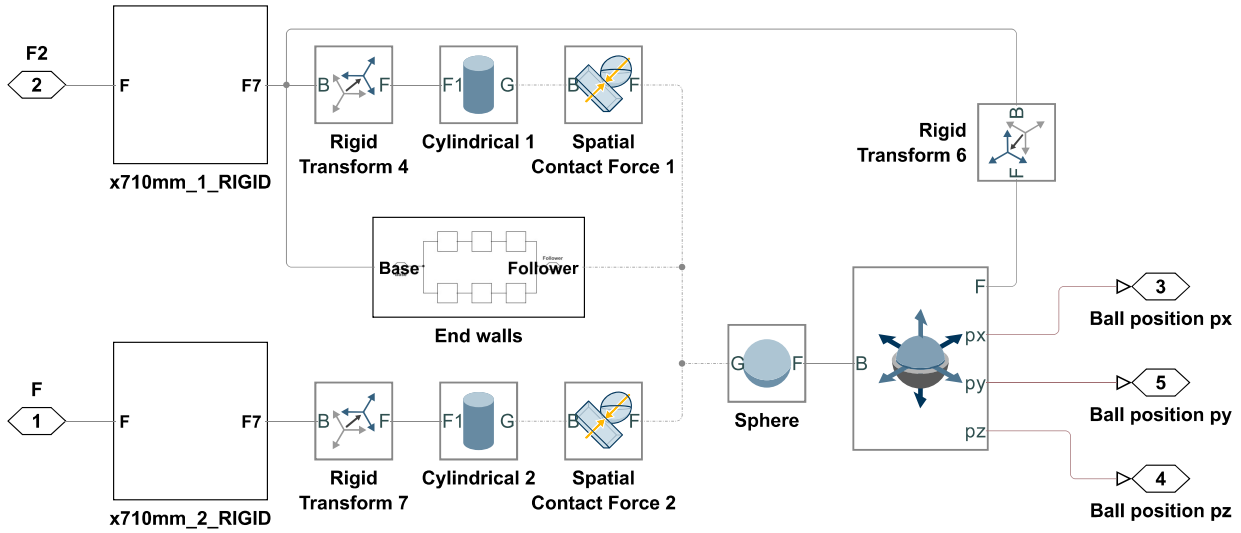

Figure 4.14: Rolling ball on rods

A rolling ball on rods needs further adaptation in order to simulate contact between the rods and the rolling ball, in addition it is necessary to set limits at the ends so that the ball does not roll of the beam.

The first step is to enter "Rigid Transform 4 and Rigid Transform 7" they are rotated about the X-axis by 90 deg for "Cylindrical 1 and Cylindrical 2". Cylindrical is modeled without mass and is modeled with the same size as the rods "x710mm_1_RIGID and x710mm_2_RIGID" to represent these. Frame F1 based on the geometric center of the side surface from export entire geometry see Fig. 4.14.

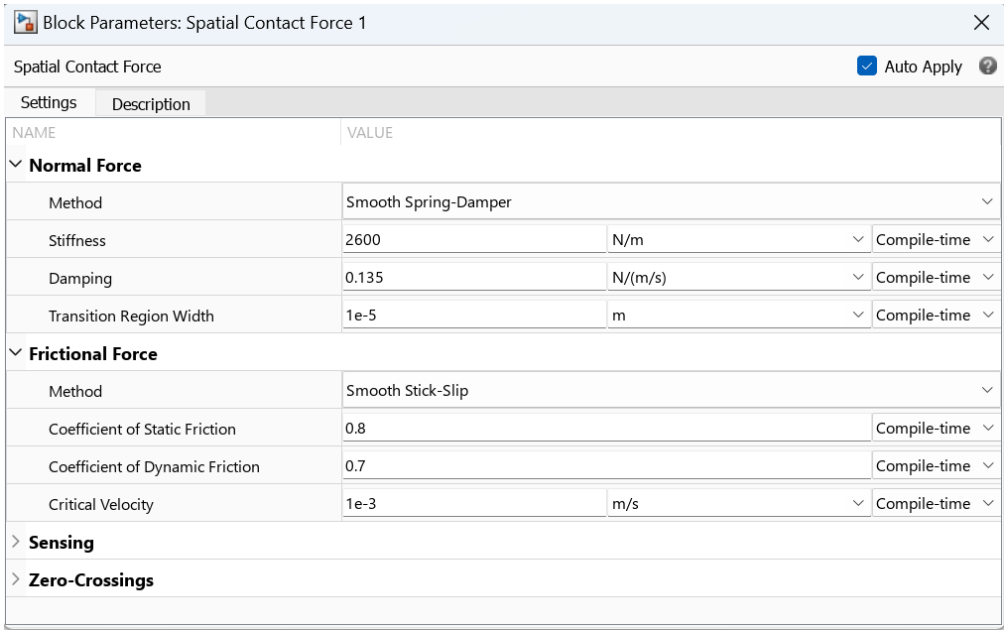

Figure 4.15: Spatial Contact Force

The contact between the ball and the cylindrical is determined through the use of a spatial contact force, as illustrated in Fig. 4.14. The values for normal force are adjusted in stiffness and damping and are provided by [22] and is illustrated 4.15. The values for frictional force are bases on which the ball should not slide.

The ball that roll are modeled with solid: sphere where inertia is provided by SolidWorks, shown in Fig. 4.16. Export the entire geometry to connect with the Spatial contact force. Frame 1, F is placed on the center of the ball in frame origin, the rest is set as default.

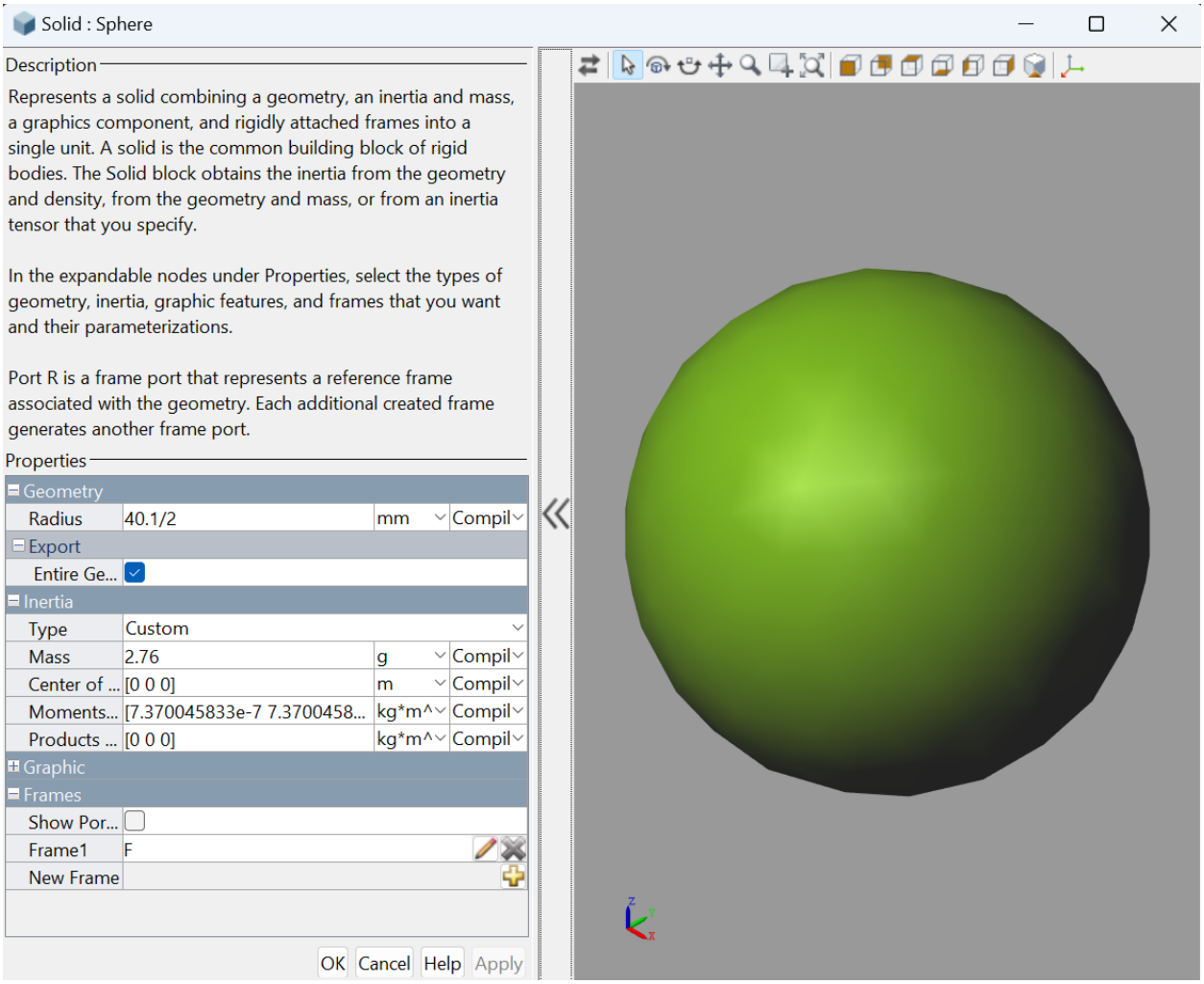

Figure 4.16: Ping-Pong Ball, Inertia calculated in SolidWorks

The ball is connected to the 6-DOF joint see Fig. 4.14 to simulate the ball moving on the beam. By retrieving the position and speed of the XYZ prismatic primitive, can be used as feedback on position and speed. See Fig. G.1 for specification.
In order for the ball to be placed correctly according to the "6-DOF joint", "Rigid Transform 6" is adjusted so that the ball starts in the center of the beam, see Fig. G.2.

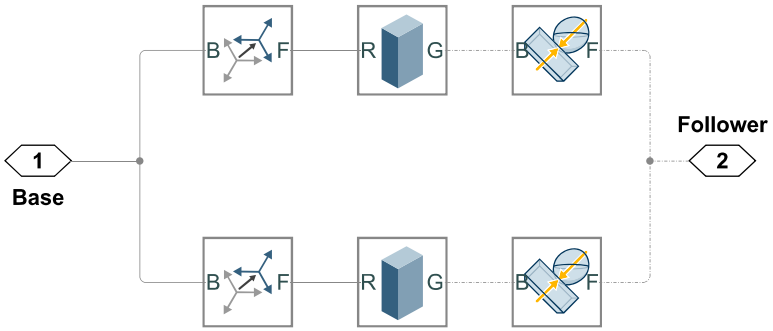

Figure 4.17: End walls

End walls are placed at the ends of the beam on both sides and are massless so as not to affect the model. The placement using rigid transform to place the walls in the end is viewed in Fig. G.4 and G.5. The contact between the ball and the wall consists of spatial contact force which has the same parameters as in Fig. 4.15. Through follower it is connected to the sphere as illustrated in Fig. 4.14.

Calculation of position cartesian coordinate system using eq. 2.22 as illustrated in Fig. G.3

4.2.4 Rexroth drive model

To simulate the model as accurately as possible in Simscape, the model of the servo and the driver was requested by contacting Bosch Rexroth [36]. This must be set up together by simulating the mechanical model (Ball and beam) with the servo and the drive. After the files have been obtained and unpacked in the folder, the model could be set up. The model of servo and drive comes with 3 options for version of matlab: R2018b, R2019b and R2020a. After trying the different ones it turned out that 2019b worked best in this case with matlab 2022b which is installed in this case on the data.

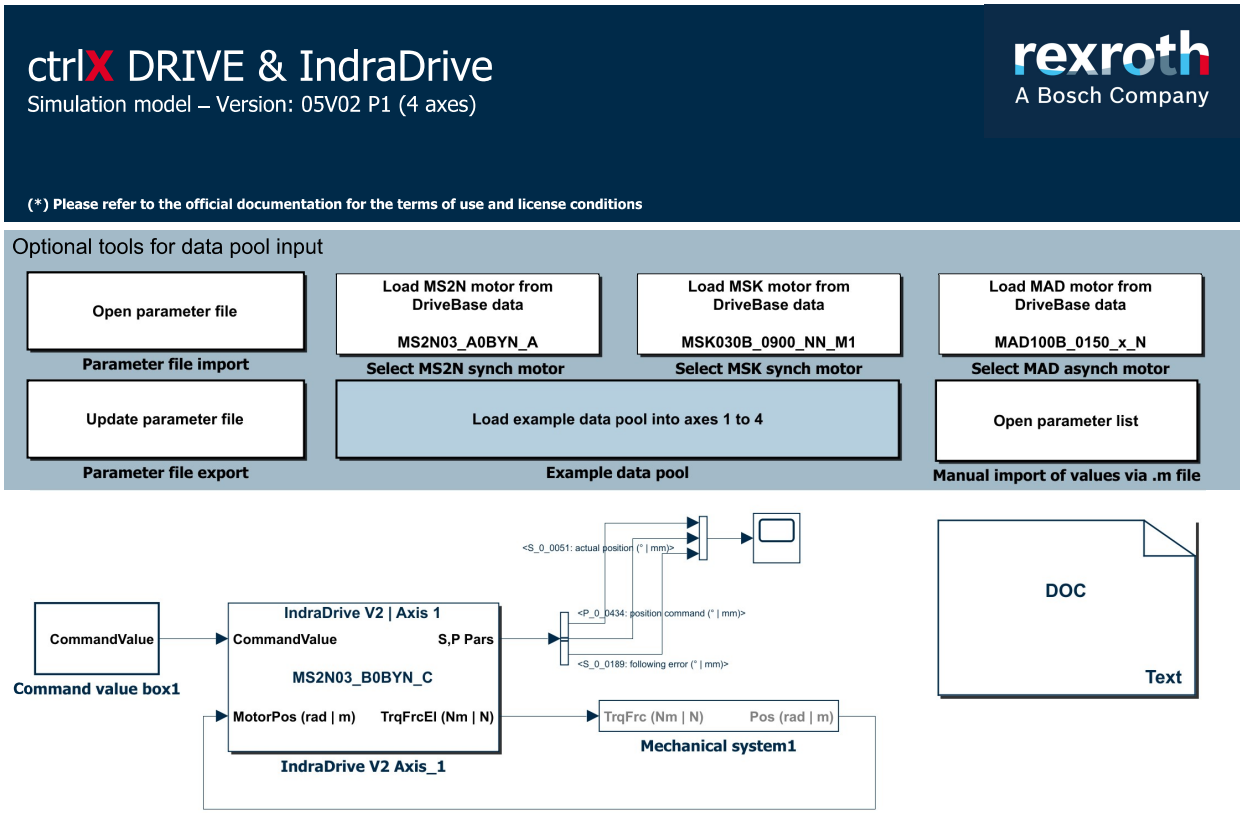

Figure 4.18: Rexroth Drive Model

The model "RexrothDriveModel_4Axes_R2019b" is opened, and "Optional tools for data pool input" appears at the top in Fig. 4.18. There, the parameters of the desired servo motor are loaded under "Load MS2N motor from DriveBasedata". The servo motor used in this case is named "MS2N03-B0BYN-HMSH1-NNNNN-NN" in the physical model. In the simulated model, only "MS2N03-B0BYN-C" is available. The difference between these is that "C" is SIL2 and H is SIL3, while the remaining parameters remain almost the same [1], some minor adjustments will require further attention.

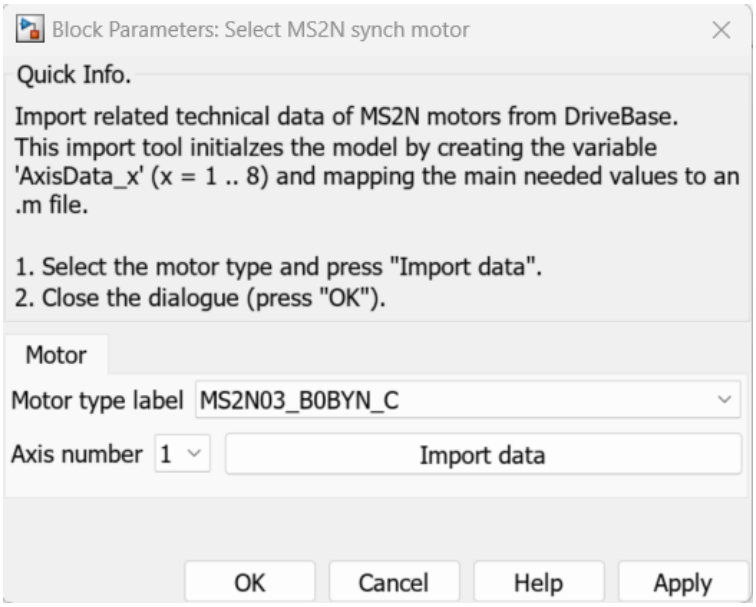

Figure 4.19: Select MS2N synch motor

When the motor is loaded and it appears in "IndraDrive V2 Axis_1" as "MS2N03-B0BYN-C" see Fig. 4.19. Then all the files inside the folder "internalR2019b" must be moved out to the same folder that the simulink model runs in. When the data is imported, "AxisData_1_user.m" is opened as a script, here values can be changed as desired if limitations in the model are desired. The following parameters are changed here acording to CtrlX Setup 8 and data [1] for optimised values in driver Matlab editor:

```
****************** Motor Data ************************
AxisData_1.DriveParameters.P_0_0510.Data = 3e-05;% (kgm^2)
************ Current Loop Configuration **************
AxisData_1.DriveParameters.S_0_0106.Data = 36.90;% (V/A)
************ Velocity Loop Configuration ************
AxisData_1.DriveParameters.S_0_0038.Data = 4774.6482919099;% (rpm)
AxisData_1.DriveParameters.S_0_0039.Data = -4774.6482919099;% (rpm)
AxisData_1.DriveParameters.S_0_0100.Data = 0.048;% (Nm/rad/s)
AxisData_1.DriveParameters.S_0_0101.Data = 0.50;% (ms)
AxisData_1.DriveParameters.S_0_0138.Data = 2;% (rad/s^2)
************ Position Loop Configuration ************
AxisData_1.DriveParameters.S_0_0104.Data = 0.20;% (1000/min)
************ Torque|Force Limitation Settings ********
AxisData_1.DriveParameters.P_0_4034.Data = 12.1;% (s)
AxisData_1.DriveParameters.P_0_4035.Data = 11.3;% (min)
```

Changes for default driver data std [1] Matlab editor:

```
****************** Motor Data ************************
AxisData_1.DriveParameters.P_0_0510.Data = 3e-05;% (kgm^2)
************ Current Loop Configuration **************
AxisData_1.DriveParameters.S_0_0106.Data = 36.90;% (V/A)
************ Torque|Force Limitation Settings ********
AxisData_1.DriveParameters.P_0_4034.Data = 12.1;% (s)
AxisData_1.DriveParameters.P_0_4035.Data = 11.3;% (min)
```

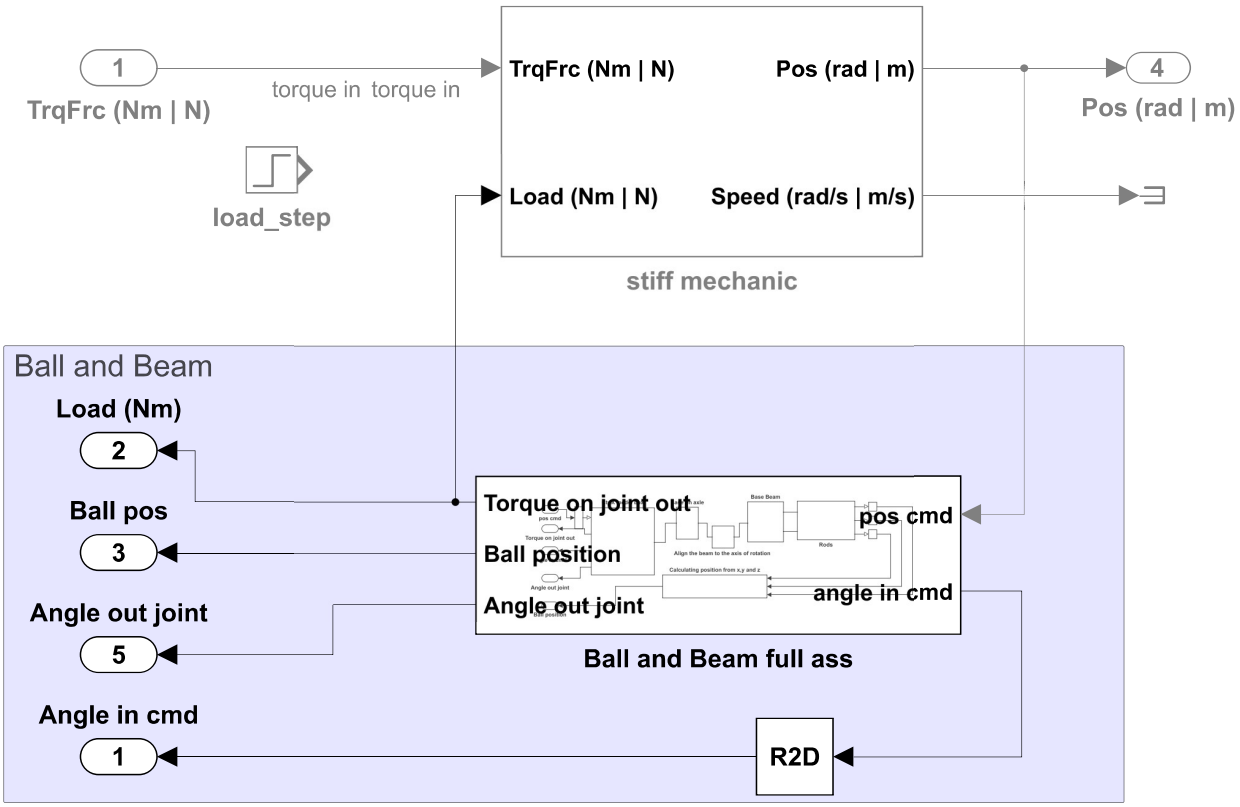

Figure 4.20: Ball and Beam add two mechanical system

Displayed in Fig. 4.20, the position is retrieved from "stiff mechanic" and entered into the cmd "Ball and Beam full ass" which is the desired position from the servo. The physical torque from the rotation in Nm which is sensed from "Revolute3" see Fig. 4.8 returns as load in "stiff mechanic".

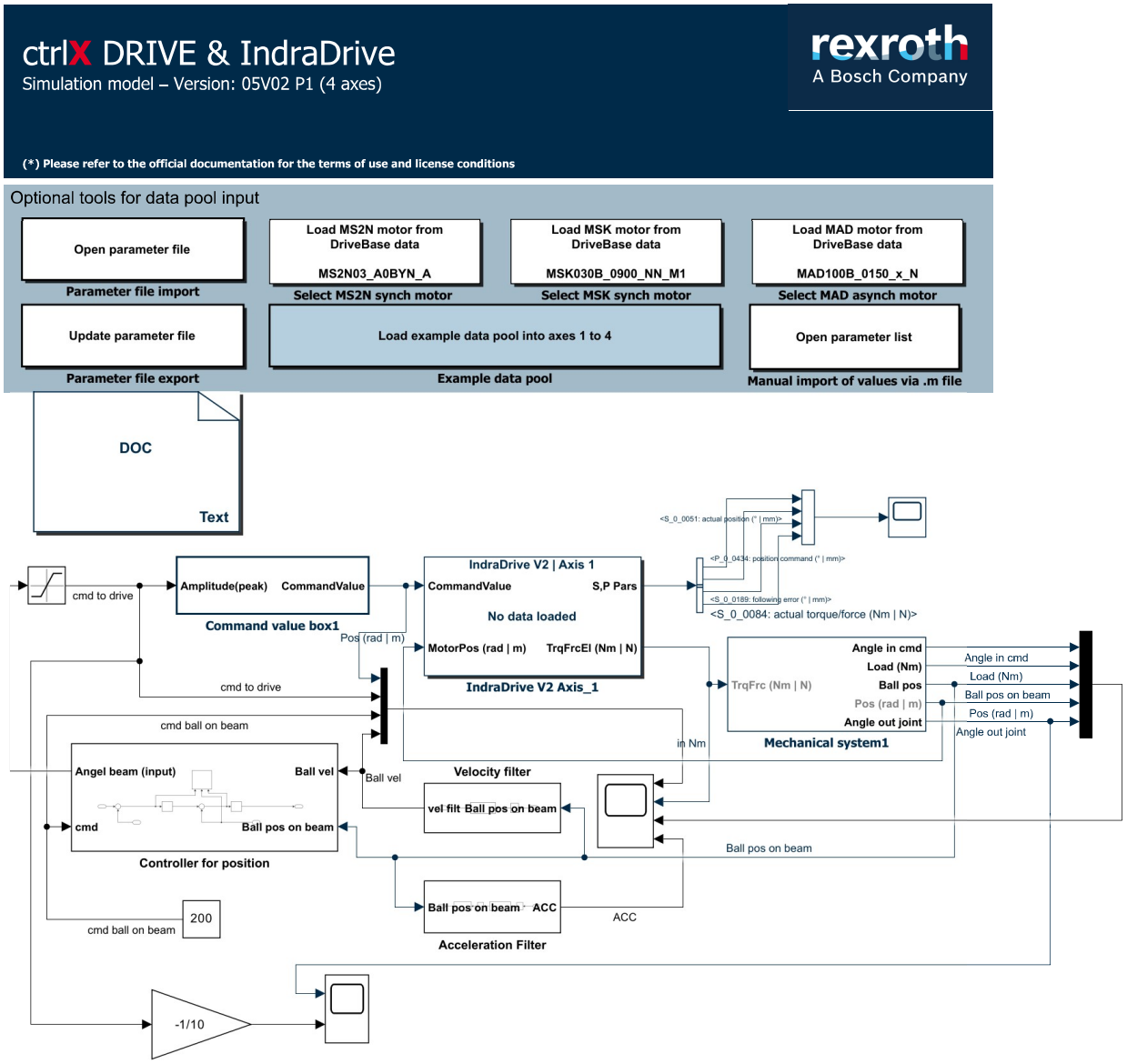

Figure 4.21: The entire model

Shown in Fig. 4.21, the driver and servo model is assembled with the mechanical part Ball and Beam. Adjustments have been made to "mechanical system1" as described earlier. Further endings have been made in "command value box1", "Controller for position", "velocity/acceleration filter" and "Saturation" has been created which will now be described further:

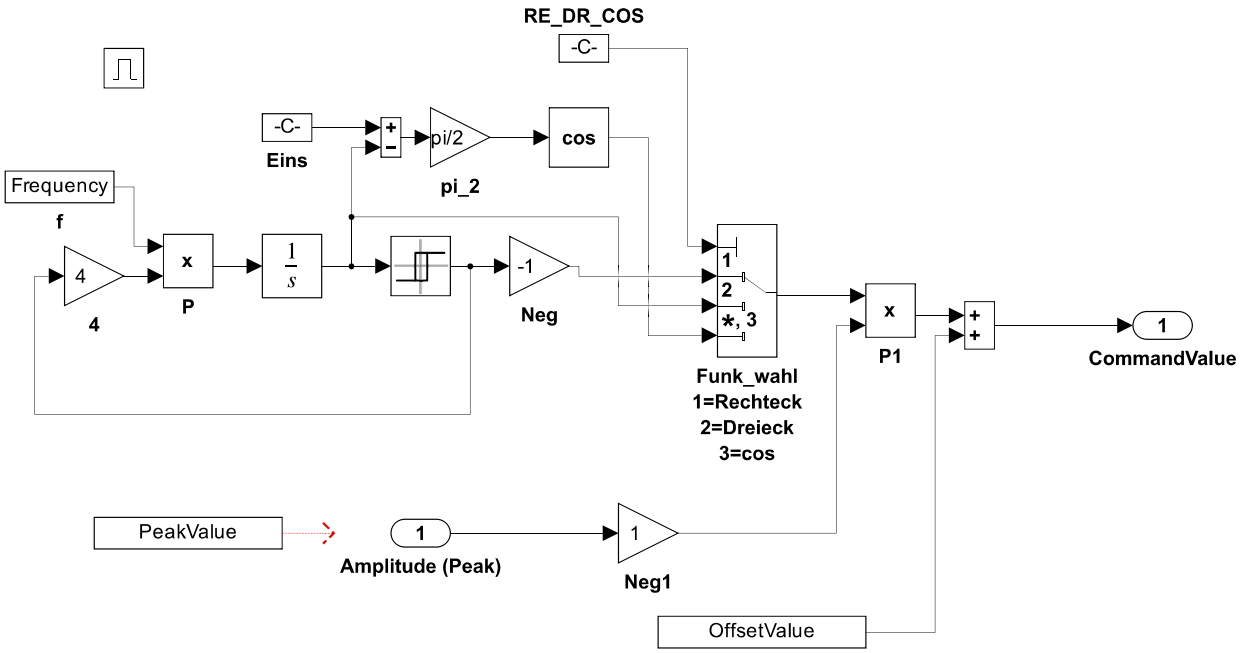

Figure 4.22: Changes in Command value box1

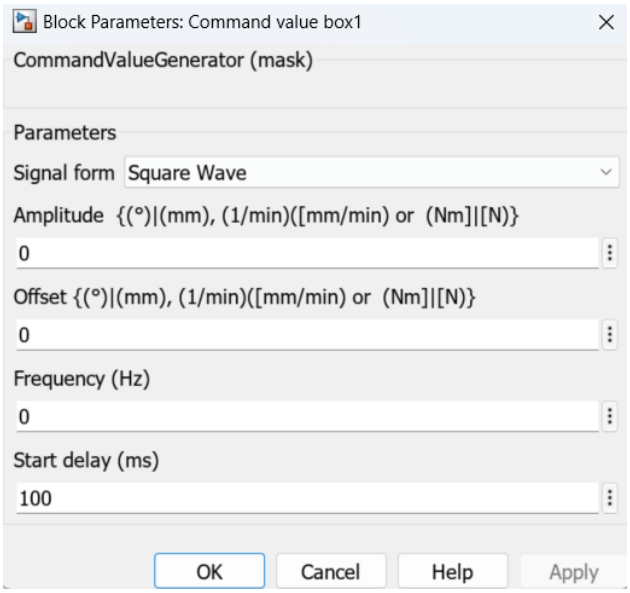

Figure 4.23: Command value box1

Changes that have been made in "Command value box1" are that "PeakValue" has been disconected and "Amplitude(Peak)" has been added see Fig. 4.22 which is connected to "Saturation2". Which is again connected to the "Controller for position".
The start parameters for "Command value box1" are that the signal is set to square wave and the start delay is set to 100 ms see Fig. 4.23.

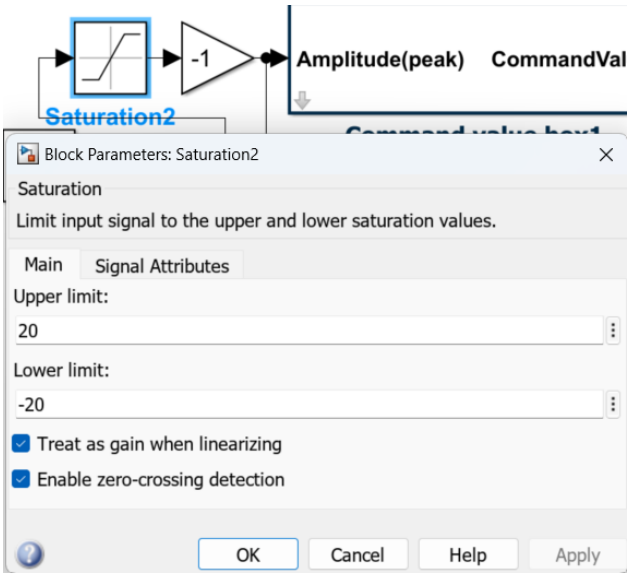

Figure 4.24: Saturation2

In order for cmd not to exceed the desired angle, "Saturation2" has been inserted, this has a ± 20 or 200 depending on gearbox ratio see Fig. 4.24.

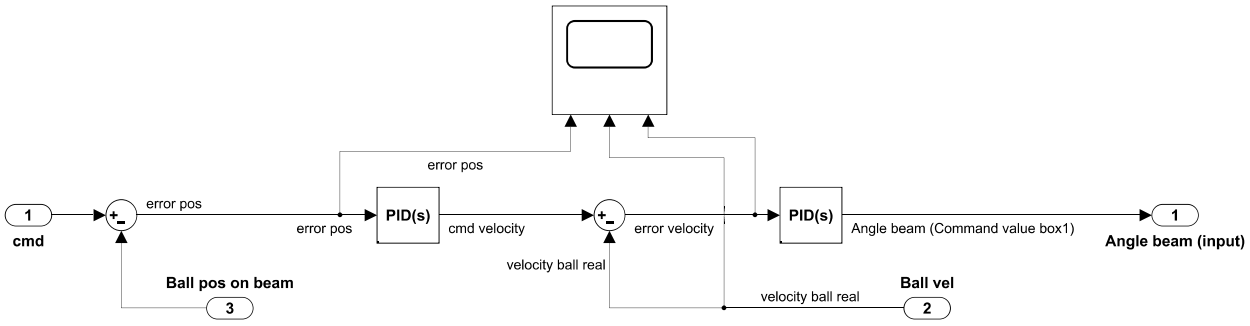

Figure 4.25: Controller for position

To control the position, the desired position is entered in "cmd" set through a controller that uses cascade regulation where the inner loop is speed and finally the angle is output as cmd to "Saturation2" see Fig. 4.25.

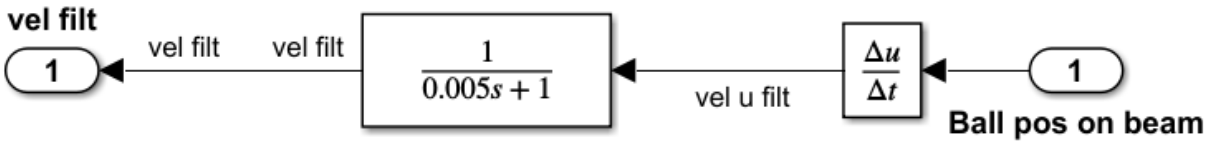

Figure 4.26: Low-pass filter velocity

The signal from the derivative is somewhat noisy, therefore it is filtered through a low-pass filter as shown in Fig. 4.26 and goes on to "Ball vel" see Fig. 4.25.

4.3 ctrlX software

All the software needed for the project was downloaded through the ctrlX store after registering an account. The store is accessed at [34].

Sources for development support

Bosch Rexroth data sheets:

- ctrlX Drive System [29]
- ctrlX PLC Engineering Application Manual [41]
- ctrlX Drive Controllers [40]
- ctrlX PLC Library Reference book [42]

The ctrlX Core is accessed through the ctrlX Works software. It can be downloaded by visiting the ctrlX store [35] Version 1.18.
The ctrlX Works program is the base for accessing devices and setting up real and virtual cores. It includes the programs I/O engineering, Drive engineering, and PLC engineering.
The ctrlX Core is running on a Linux real-time operating system that uses apps as programs. The following apps were installed using the ctrlX store:

- CTRLX CORE - PLC APP Version 1.18.2 [33]
- CTRLX CORE - MOTION APP Version 1.18.2 [31]
- CTRLX CORE - 3D VIEWER APP Version 1.20.1 [28]
- CTRLX CORE - OPC UA SERVER APP Version 1.18.1 [32]
- Smart HMI - WebIQ Server for ctrlX CORE [13]

The HMI was designed using the Smart HMI - WebIQ Designer [12]
Access [39] for a tutorial on installing the WebIQ designer and creating the HMI project. A step by step instruction on creating the HMI project is explained by Bosch Rexroth [44].

4.3.1 CtrlX Setup

Setting up the ctrlX-core was done by following the "How to" videos, under the "Basics" section, found on the Bosch Rexroth site [37]. The procedure is

1. Set the manual ethernet IP settings on the computer to: IPv4 address: 192.168.1.2, IPv4 mask: 255.255.255.0
2. Type https://192.168.1.1 in the browser and log in with the username: boschrexroth, password: boschrexroth. Accept the page even if the browser says it is not private.
3. Install the downloaded apps in "home/+ Install an app" by activating "Service mode". Switch to "Operation mode" when finished
4. Create a setup file to be able to store the configuration. Go to settings/setup/create and save a copy to the computer. This file stores the configuration.
5. Create a backup by going to settings/Backup & restore. Click "backup" and a file will be stored on the local computer.
6. Create an axis Y in the motion app. Go to motion/axes, switch to "service" mode, and click "+". Add an axis with the name "Axis_Y" and choose "rotational".

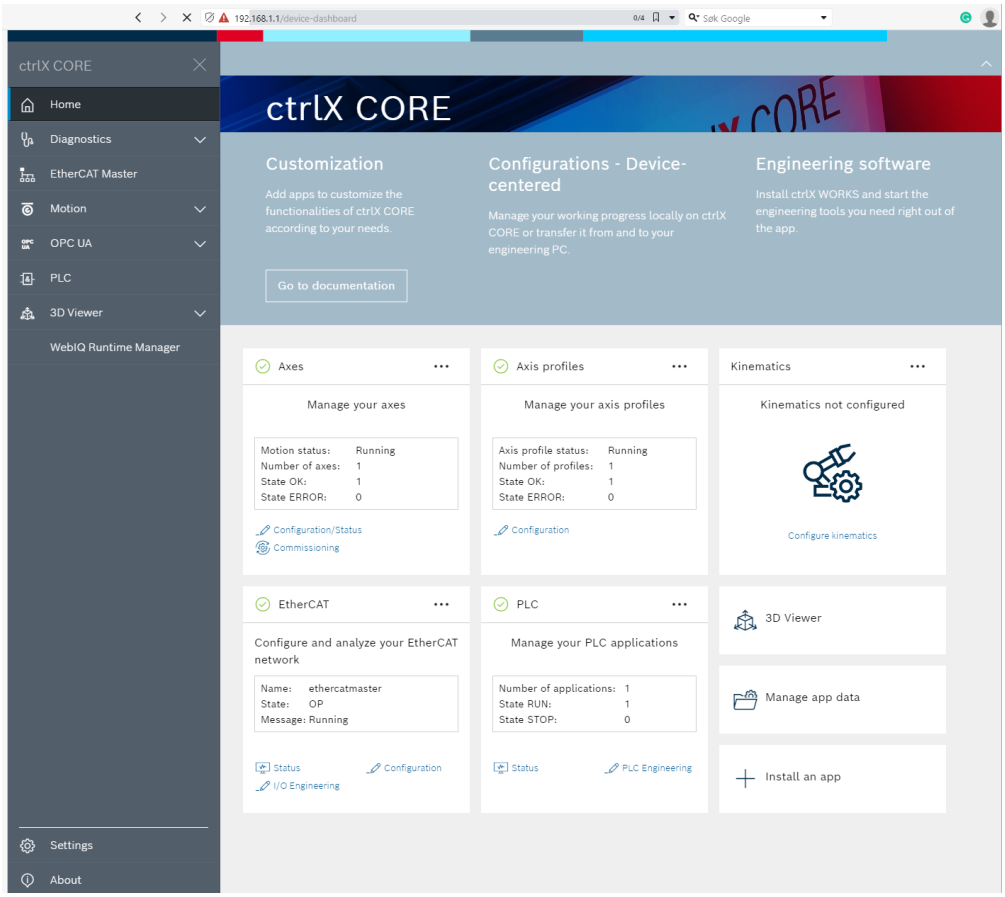

Figure 4.27: ctrlX Core home screen

7. Set up extra variables to be sent on the EtherCAT bus from the drive to the PLC. Click "EtherCAT Master", in the main menu, and choose the "IO/ ctrlX" tab in the right corner. This will open I/O engineering.

    - Go to "Online/Show online data.
    - Double click "ctrlX_DRIVE_XMS_SoE_" and choose "expert process data".
    - Click on "AT" in the "PDO List" screen. Click on "+Insert" in the lower screen "PDO content" and add the following signals: "P-0-0210 analog value", "S-0-051 Position feedback value 1", "S-0-0383 Motor temperature" and "P-0-0049 Effective torque value"
    - Go to and click on "Online/download" to send the new configuration to the ctrlX core.
    - Click "save" and exit.

8. The drive needed to be tuned due to the high mismatch between the mass moment of inertia of the load and motor see subsection "Inertia mismatch", chapter F.3, eq. 2.51. In this situation, the motor and drive needed a much slower controller to stabilize it. Normally a built-in tuning function would find the best settings, but with a high moment mismatch, this function did not work. The following values were changed in the drive, based on manual testing. Go to "EtherCat Master/ethercatmaster" and click on the "Drive Engineering" icon under "Actions" to open the Drive Engineering program.

    - Set absolute position for encoder 1 when the beam is in the horizontal position. Choose: Axis1/Motor, drive mechanics, measuring systems/Encoder 1, motor encoder/Position data reference encoder 1/Set absolute position for encoder 1
    - Click on "PM" in the toolbar to set the drive in parameterization mode.
    - Change the velocity control loop gain by clicking: Drive control/Axis control/Velocity control loop to access the parameter overview. Change Kp to 0.048 [Nm/(rad/s)] and Tn to 0.5[ms].

- Change the position control loop Kv-factor by clicking: Drive control/Axis control/Position control loop to access the parameter overview. Change Kv-factor to 0.20 [1000/min]
- Click on "OM" in the toolbar to set the drive in operation mode.
- Save and close the Drive engineering program.

The base setup is now done and it is ready for PLC programming.

4.3.2 PLC Setup

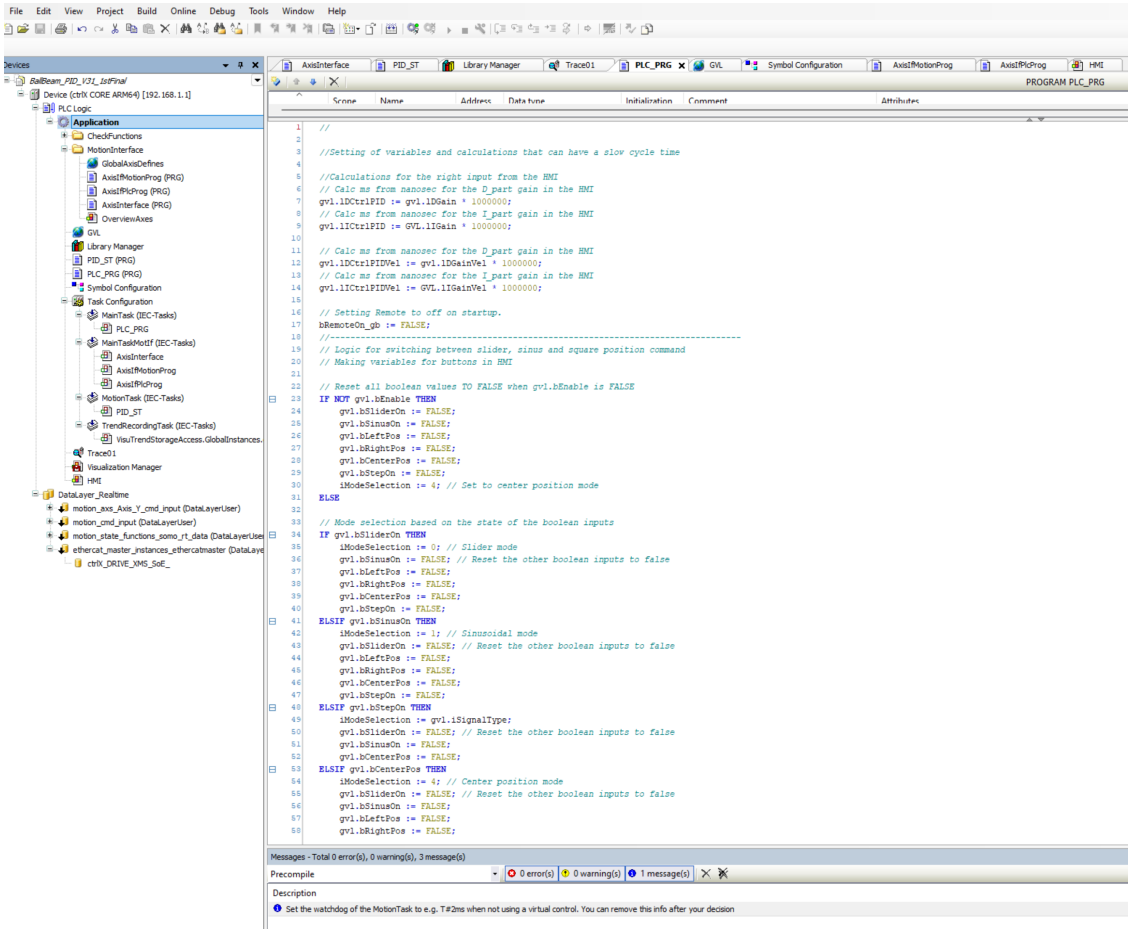

Figure 4.28: ctrlX PLC main layout

The PLC app is accessed through ctrlX Works or by clicking the PLC tab in the ctrlX menu and choosing the PLC icon in the right corner. This will open the PLC engineering program. The ctrlX PLC app runs a scalable runtime system based on CoDeSys V3, which is completed by comprehensive technology libraries by Bosch Rexroth. Documentation on CoDeSys can be found at their website [11].

1. Create a new project when prompted on startup of the program.

2. Insert a axis template by right clicking on "Application" and choose "insert template". Go to the "MotionInterface" tab and choose "AxisInterface" then click "Insert". This will insert a predefined template of a standard axis control. This template serves as the base for further programming. The items that are not needed, will be deleted.

3. Define the Axis_Y as the rotational axis controlled by the drive by modifying the GlobalAxisDefines, Fig.: D.2.

4. The AxisInterface(PRG) was modified. This program controls the motion of the axis.

5. Adding the package, CXA_LoopControl, to the library for PID functionality. right click on "Library" choose "Add Library" and go to Application/ctrlX Automation/Base/CXA_LoopControl. Add by clicking "OK".

6. Adding a GlobalVariable list so that the global variables can be sent to the data layer. Right-click "Application" then click on "Add object". Choose "Global Variable List"

7. Adding the symbol configuration object makes it possible to choose which variables to send to the datalayer. Right click "Application" then click on "Add object". Choose "Symbol Configuration".

8. The variables added to the EtherCAT in subsection 4.3.1 are included in the PLC by mapping the channel from the data layer to a PLC variable. Start by adding the ctrlX Core data variables on the data layer to the PLC. Right-click on "DataLayer_Realtime" and choose "/Edit/Insert all data from ctrlX CORE again". The "DataLayerNode I/O Mapping" tab pops up and the following variables are mapped to respective channels. Double click in the variable field that corresponds to the channel that will be mapped. Click the "..." tab and go to /Application/MotionInterface/AxisInterface/"variable to be mapped". Map iBallFeedback to AT.Analog_input_1, iTempMotor to AT.Motor_temperature and iForce to AT.Effective_torque_force_command_value.

4.3.3 PLC Programming

The programs are found in Github, [26].

Motion task

The main motion program, PID_ST, is added to the file structure and set as Motion Task with a 2ms cycle time. This task runs the control code and benefits from a low cycle time for better precision. A low cycle time updates the values and corrects the position more frequently. The program includes signal filters and PID controllers for the velocity and position loop. A challenge for the control is to filter out the high-frequency signal noise so that the derivation part of the position controller does not react to the noise. Using the filter function block, IL_IIRType01, has the effect of a low pass filter. The filter uses forward (numerator, b) and backward (denominator, a) coefficients found in Matlab by converting the low pass transfer function from continuous to discrete time with the c2d command.

Local HMI

The PLC app has a visualization toolbox where a local HMI can be made to simulate inputs and outputs. The HMI in Fig.: 4.29 shows the HMI made to interact with the PID_ST program.

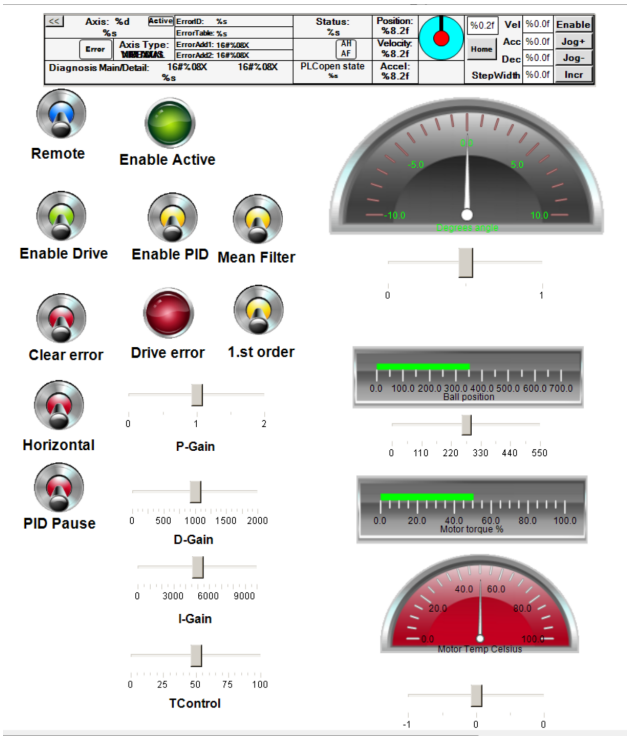

Figure 4.29: The local HMI used for setup and testing

Trace function

The PLC has a trace functionality to record signals in real-time. These values can be saved as a .csv file that is readable in Matlab. There is an issue with this file. It does not make individual columns for the signals in Matlab and will display the signals as is was the same. The .csv file has to be modified so that the signals are in individual columns before use in Matlab. To activate the trace functionality, click "Application" and choose /Add object/Trace. Double-click on "trace" and add the signal to be traced.

Test Axis motion

The loaded template has an HMI function that can be used to run the axis. Testing of axis movement was done through the template HMI, Fig.: 4.30, to verify the beam movement.

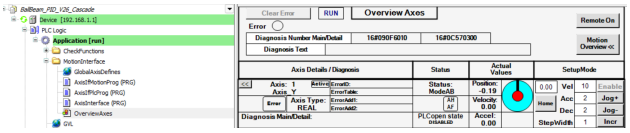

Figure 4.30: HMI from template for moving the axis

A small program was made D.1 to test the beam dynamics. The position signal was connected to a sinusoidal signal generator function block. The frequency and amplitude were controlled through the local HMI. The amplitude was set at 10 degrees and the frequency increased from 0 to where the beam started to deviate from the amplitude. The beam position, position command, and position error were traced. The sinus frequency at where the max amplitude reduces to $\frac{MaxAmplitude}{\sqrt{2}}$ is the -3 db, system bandwidth. -3 db is the same as $-3db = 20log(0.707)$ which is at 70.7% of the maximum amplitude. Two traces were done, one with no restriction in acceleration and rpm and the other with a 2 rad/s^2 acceleration/deceleration and 500 rpm maximum.

Filtering of the position signal
The position signal was traced to establish the amount of noise on the signal. The velocity loop bandwidth simulated in Matlab was the base for the filter. A low pass filter, IL_IIRType01, with a cut-off frequency set at 2 times the system bandwidth was implemented in the signal path and traced together with the original position signal.

Programming the velocity loop
The position signal is used to find the velocity by derivation. The discrete-time derivation is done by finding the difference in position (delta p) over the difference in time (delta t) $\frac{dp}{dt} = velocity$. It is done by using the cycle time as dt and subtracting the previous position from the current position. This resulting signal can be very noisy if the input signal also is noisy since the velocity is the rate of change of the position signal. A PID function block, IL_PIDType03, was used for the closed-loop functionality. The filtered velocity signal was used as feedback.

Tuning the velocity loop
The ball was put in the 0 mm position, then a 0.1 m/s step input was set. The ball velocity was traced to find the optimal kp gain for the control to reach a max overshoot of 5%. The trace was compared to the Matlab simulation to see how the simulation matches the real system.

Programming the position/ cascade loop
The closed loop position controller was added to the velocity loop to make a cascade controller. A PID function block, IL_PIDType03, was used for the closed-loop functionality. The input to the position loop was the ball position command and the feedback was the filtered position signal. The output was sent as an input to the velocity controller. The output from the velocity controller was the angle command for the beam position.

Tuning the cascade loop
The system was run with increasing position loop kp gain to introduce a steady oscillation for the Ziegler- Nichols tuning. Tuning the cascade loop was done based on the simulation done in Matlab. The control structure was a PD control where the D part was necessary to reduce the overshoot and increase the system bandwidth. A pure kp gain introduces increasing oscillations. A moderate kd value was chosen and the kp was increased until the specified response was met. The kd was increased to reduce the overshoot to within specifications. Using the Ziegler- Nichols method might yield better results. The values are tested and compared to the ones found manually.

4.3.4 PLC Program Flowcharts

Flowcharts clearly outline the individual programs' logic. These flowcharts will aid in understanding the program control flow, conditional statements, and loops. They show the program logic and simplify the comprehension of the program functions. The flowcharts will also provide a common language and understanding for future troubleshooting and system optimization.

Program structure

The ctrlX PLC app was programmed in ST language following the flow charts in this section. Building on the base template, Section 4.3.2/2, provided by Bosch Rexroth and preinstalled as an option in the PLC app makes the addition of custom code easier. The base functionality is described in the documentation, PLC Engineering Application [41] and PLC Library [42]. A full PLC project documentation is available in Github [26] The flowchart in Fig. 4.31 shows the motion logic control

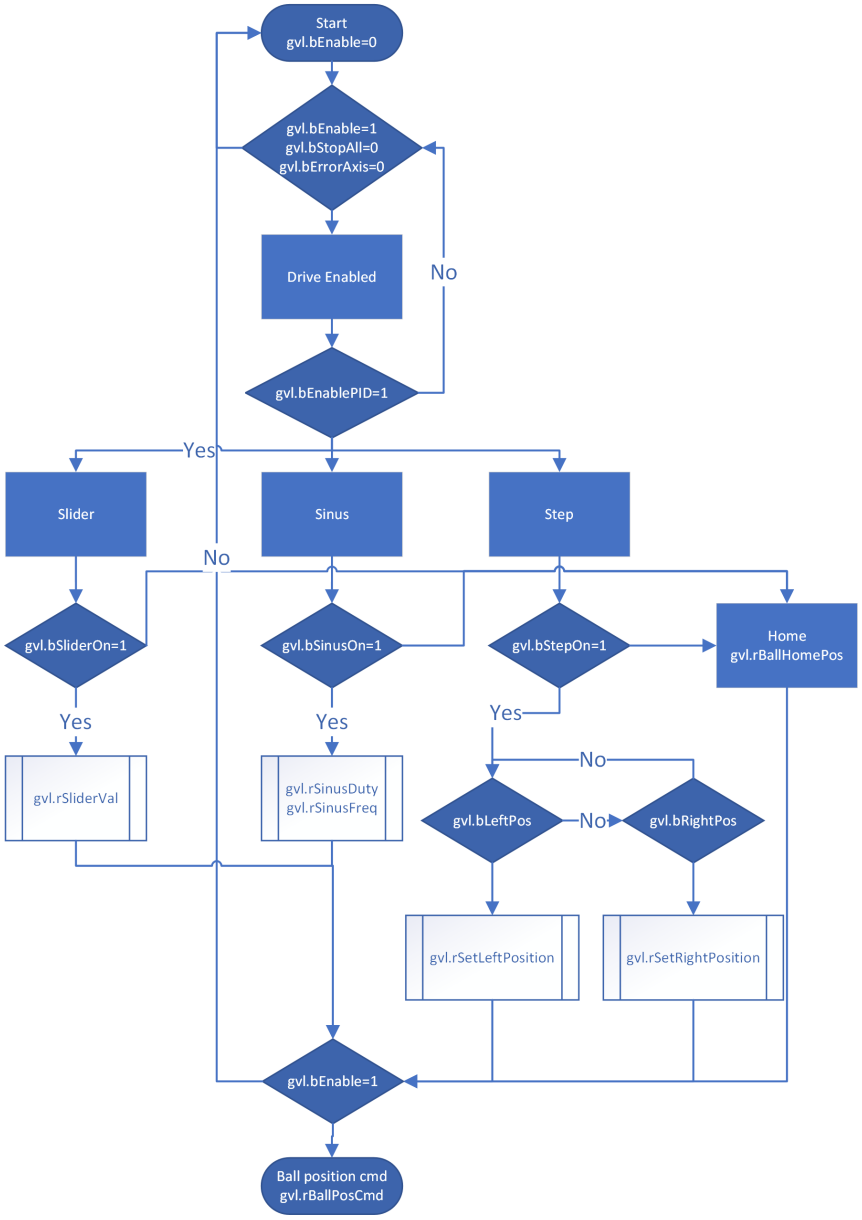

Figure 4.31: PLC/HMI Flow diagram, motion logic

structure that the HMI is programmed for.

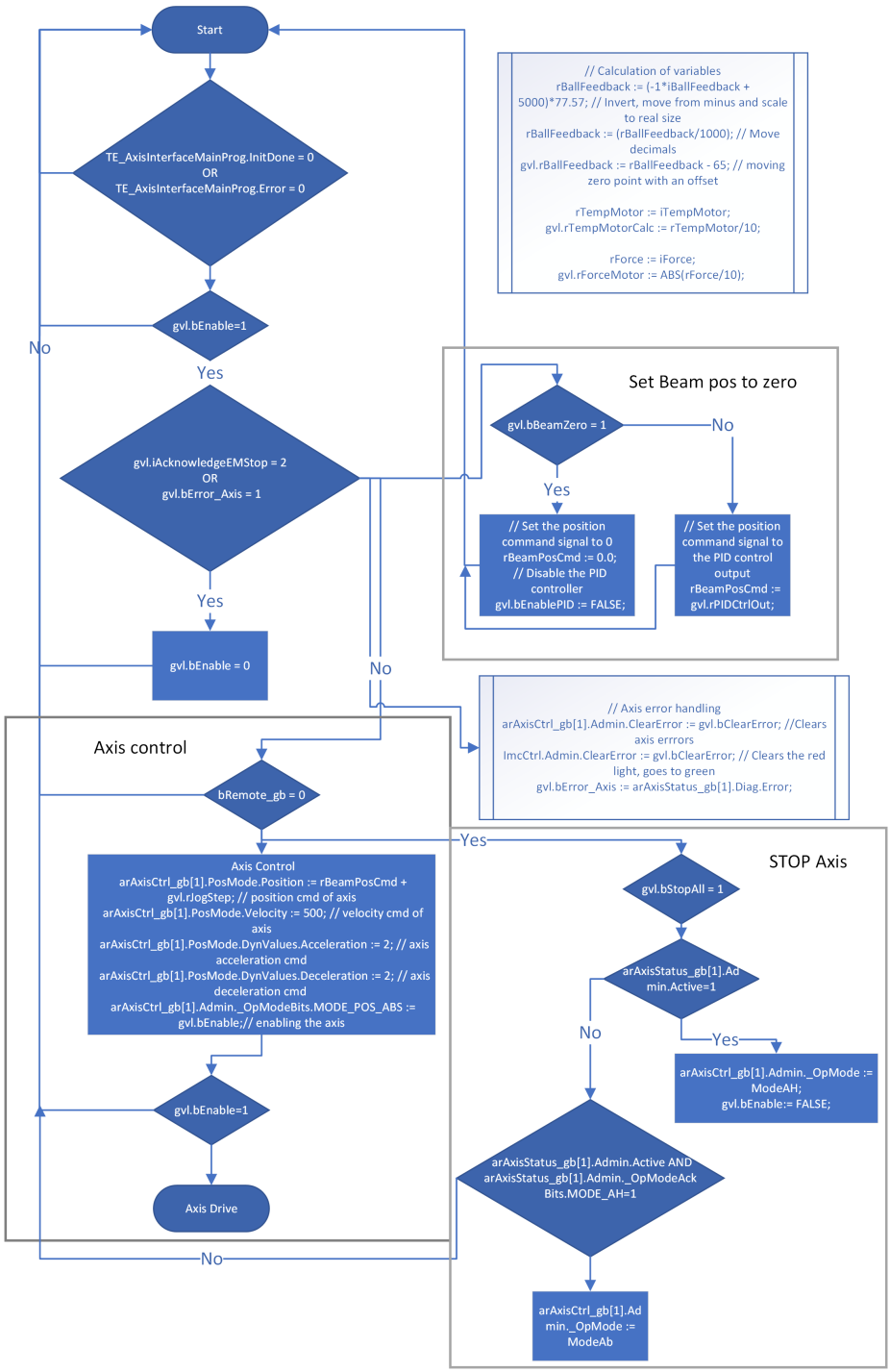

Figure 4.32: PLC Flow diagram, AxisInterface program

The flowchart in Fig. 4.32 shows the AxisInterface program in the PLC. It has the main interaction with the drive giving position command, enable, accel/decel, and velocity inputs. It also converts values and does calculations of the ball feedback, motor temperature, motor force, and ball feedback scaling/ offset. The axis error handling is defined and variables are monitored.

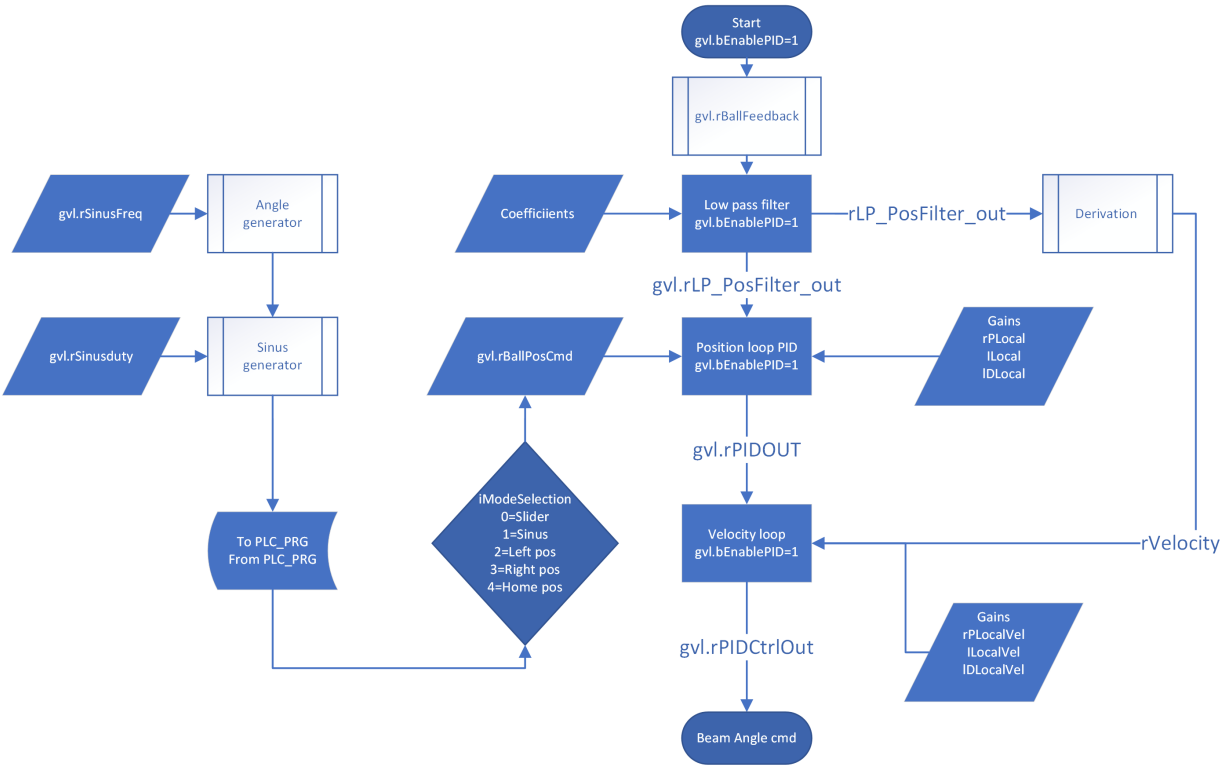

Figure 4.33: PLC Flow diagram, PID-ST controller program

The flowchart in Fig. 4.33 shows the PID controller logic with a sinus generator for sinusoidal inputs. The logic controlling the ball position command gets its input from the PLC_PRG. The IIR filter with its coefficients is directly after the position feedback.

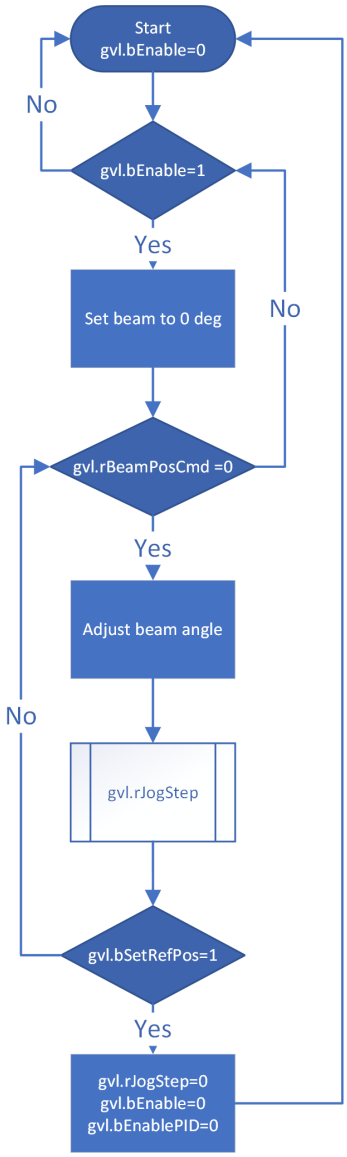

Figure 4.34: PLC Flow diagram, Beam Calibration logic

The flowchart in Fig. 4.34 shows the beam calibration logic in the PLC-PRG program. It is operated from the HMI and sets the axis at 0 degree, then adjusted and axis's absolute zero position is set.

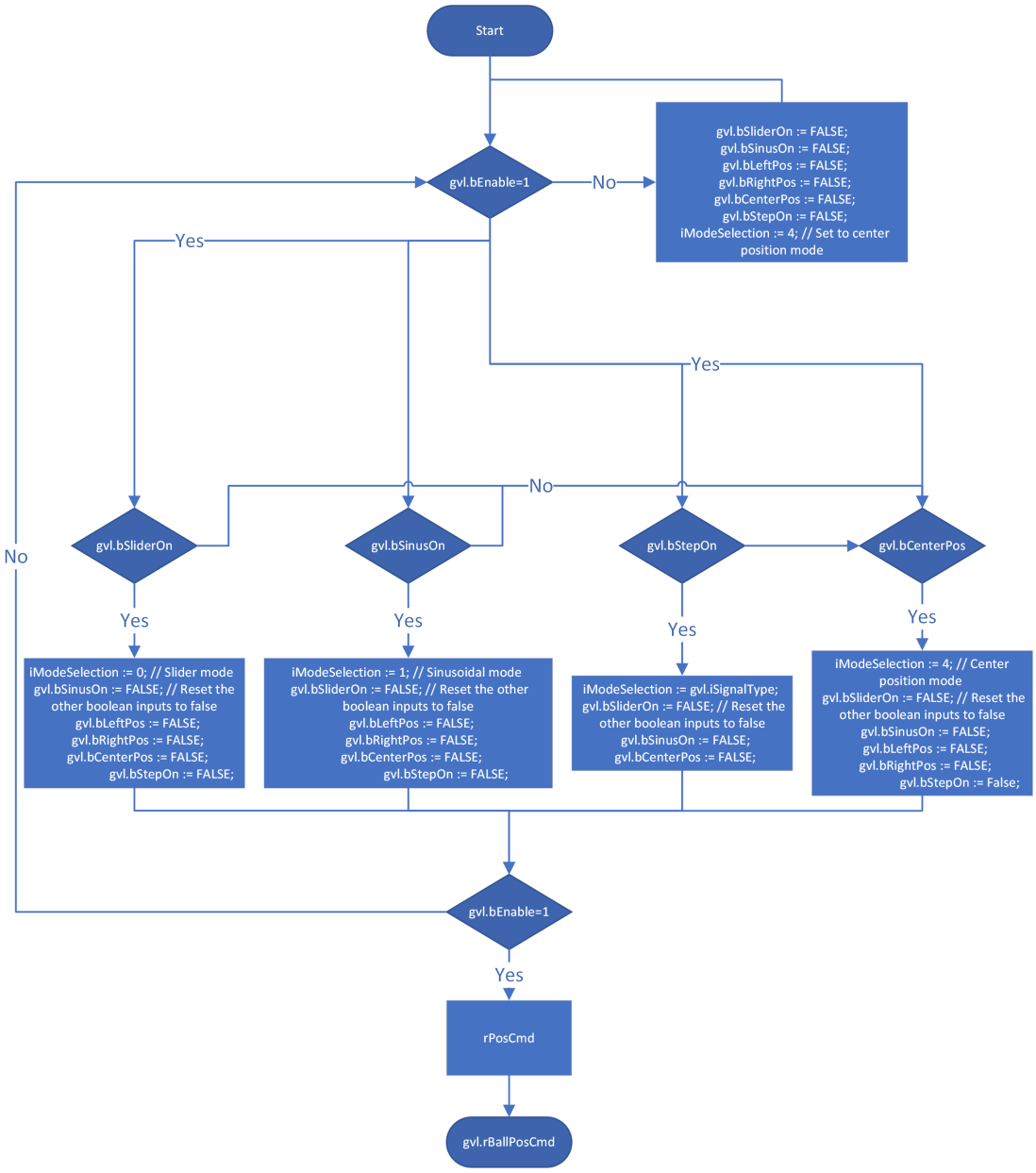

Figure 4.35: PLC Flow diagram, PLC-PRG controller program

The flowchart in Fig. 4.35 shows the mode selection logic. The inputs come from the HMI and select the different modes, SliderOn, SinusON, StepON, and CenterPos. The PLC_PRG also has additional D-part and I-part gain variable calculation to convert values from the HMI to the correct values in the PLC. The Beam calibration logic is also in the code and described in figure 4.34.

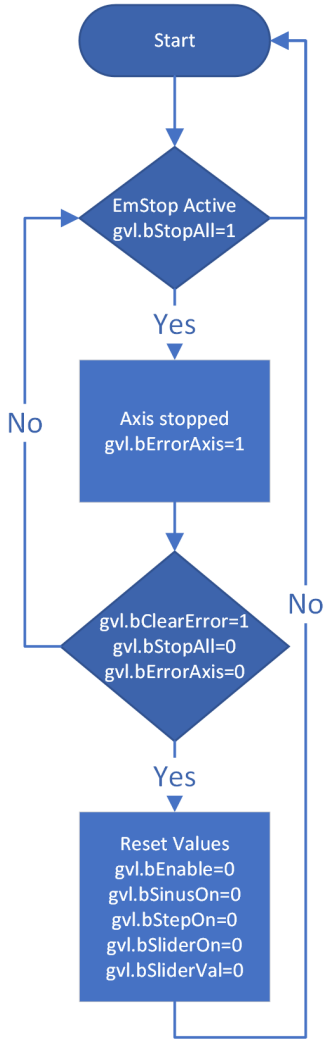

Figure 4.36: PLC Flow diagram, Emergency stop logic

The flowchart in Fig. 4.36 shows the logic when the emergency stop is pressed in the HMI. The gvl.bStopAll sets the drive to stop and the drive enable is set to false. Values are set to 0 when the emergency stop is reset and errors are cleared.

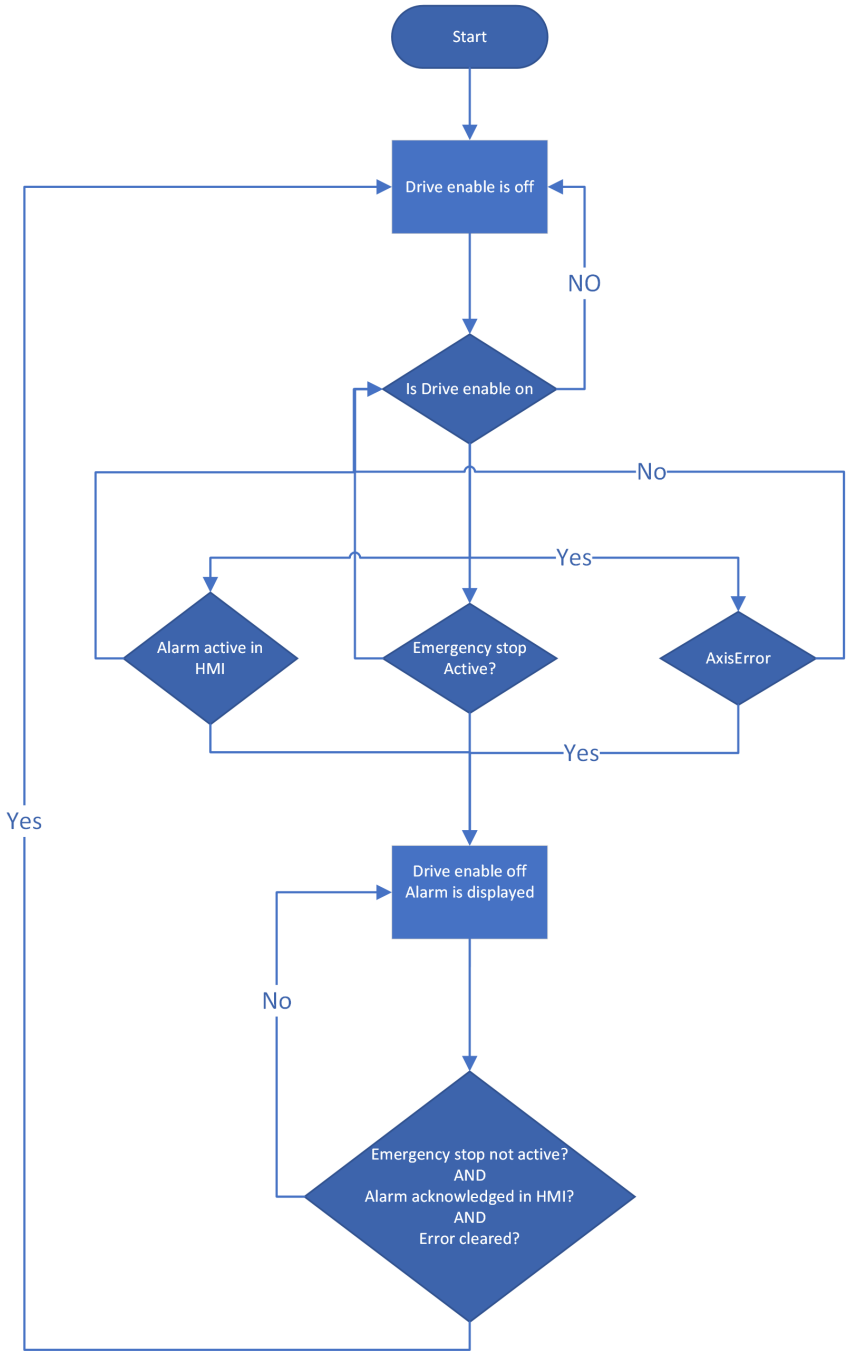

Figure 4.37: Flowchart Safety logic

To ensure safe operation without risk to life and health, the operation of the machine must be designed with protection against impact and pinching/squeezing hazards. The program includes safety measures to ensures safe operation of the machine under normal operation. The programming related to safety is shown in the Fig. 4.37. This is to make visualize the function of the machine. The standard related to collaborative robot [45] is used as a substrate to set the limitation Tbl. 3.1.

4.4 ctrlX Programming 3D Viewer

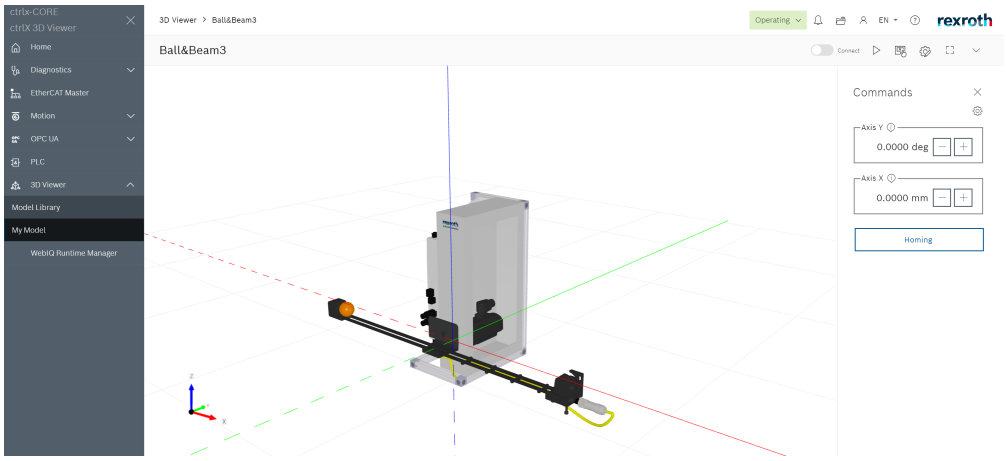

Figure 4.38: 3D viewer

The first step is to insert the model into the 3D viewer through ctrlX Core. The 3D viewer app must be installed according to Fig. 4.3. The next step will then be to import the model of the system into the 3d viewer app. This model was made with the help of Bosch Rexroth. The model was saved as STEP AP214 and was sent to Bosch Rexroth [7] who helped set up an XML file and STL file that together show the kinematics visual. On the XML file received, some changes of color and limitations were implemented on the model, see git for details [26] under programming. As shown in Fig. 4.38 it is a visual representation of how the model was after the adaptations in XML.

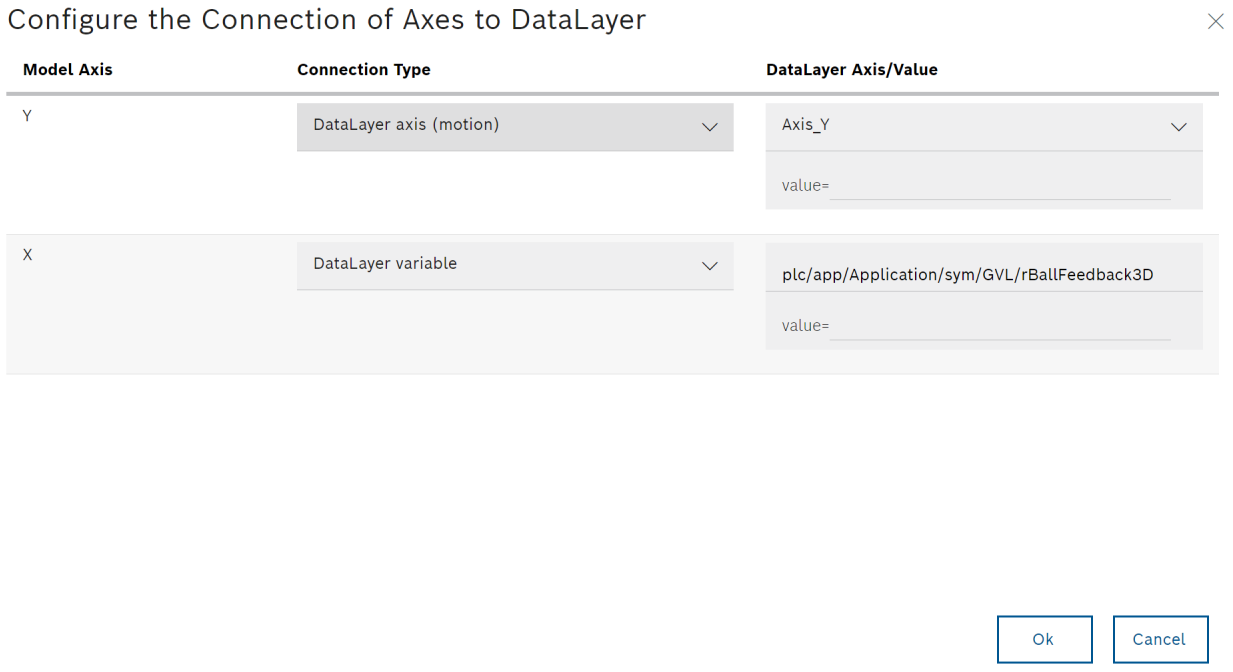

Figure 4.39: Axis Y and axis X

The axes are set up by linking the servo axis-Y to the Y-axis and the global variable, plc/app/Application/sym/GVL/rBallFeedback3D, from the PLC to the X-axis, Fig. 4.39.

The ball has to be moved to the left side of the beam which is the 0 position. The global variable, gvl.rBallFeedback3D, in the AxisInterface PRG is modified by subtracting -330 [mm] from the ball position value, gvl.rBallFeedback.

4.5 CtrlX HMI

A Human Machine Interface (HMI) should be made to interact with the user. As an app in a browser connected to the local LAN network. Inside the program, it should include a visual representation of the "Ball and Beam", a sliding bar for positioning the ball, pulsing the ball forward and back, random positioning, side-to-side oscillation, beam calibration and a warring/hazards/alarms visualization.

Sources used for development
For the development of the HMI the following sources were used to understand and implement the functionality of WebIQ:

- Bosch Rexroth How-To videos section /HMI/WebIQ [6]
- SMARTHMI WebIQ manual [48]
- SMARTHMI WebIQ Youtube channel [14]

A full version of the HMI project is found in Github [26].

4.5.1 Making of the HMI in WebIQ Designer

The HMI was designed to incorporate the following functionality:

- Fixed header with Bosch Rexroth logo, alarm list, time and emergency stop button
- Fixed left sidebar with buttons for drive enable, PID enable, error reset, slider on/off, sinus on/off, step on/off, connection status with ctrlX-Core and step direction.
- Fixed footer with motor temperature, ball position, and ball position command slider
- Center display with swipe functionality for multiple pages.
  - 1st page for 3d visualization
  - 2nd page for dials, Sinus settings, Step values, Position loop PID inputs, Velocity loop PID inputs, Position feedback filter coefficients, axis max rpm, axis accel/decel values, and beam calibration interface
  - 3rd page for trend graph with legend and signal selection
  - 4th page for alarms

The HMI was designed for a laptop display and a 16:9 tablet. The HMI is accessed through a browser, preferably Firefox, Chrome, or Vivaldi. The address is `http://192.168.1.1:10123/ballandbeamhmi/`. The login information is: Username: boschrexroth, Password: boschrexroth.

The WebIQ Runtime Manager app in ctrlX-Core and WebIQ Designer has login information: Username: admin, Password: boschrexroth1 The app is for managing the HMI projects and activating the project after it has been published to the server.

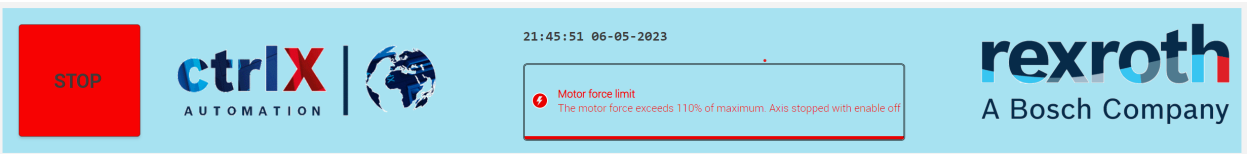

Figure 4.40: WebIQ HMI Header with alarm notification, Emergency stop, and real-time clock

The header shown in Fig. 4.40 has the emergency stop button, real-time clock, and alarm notification window.

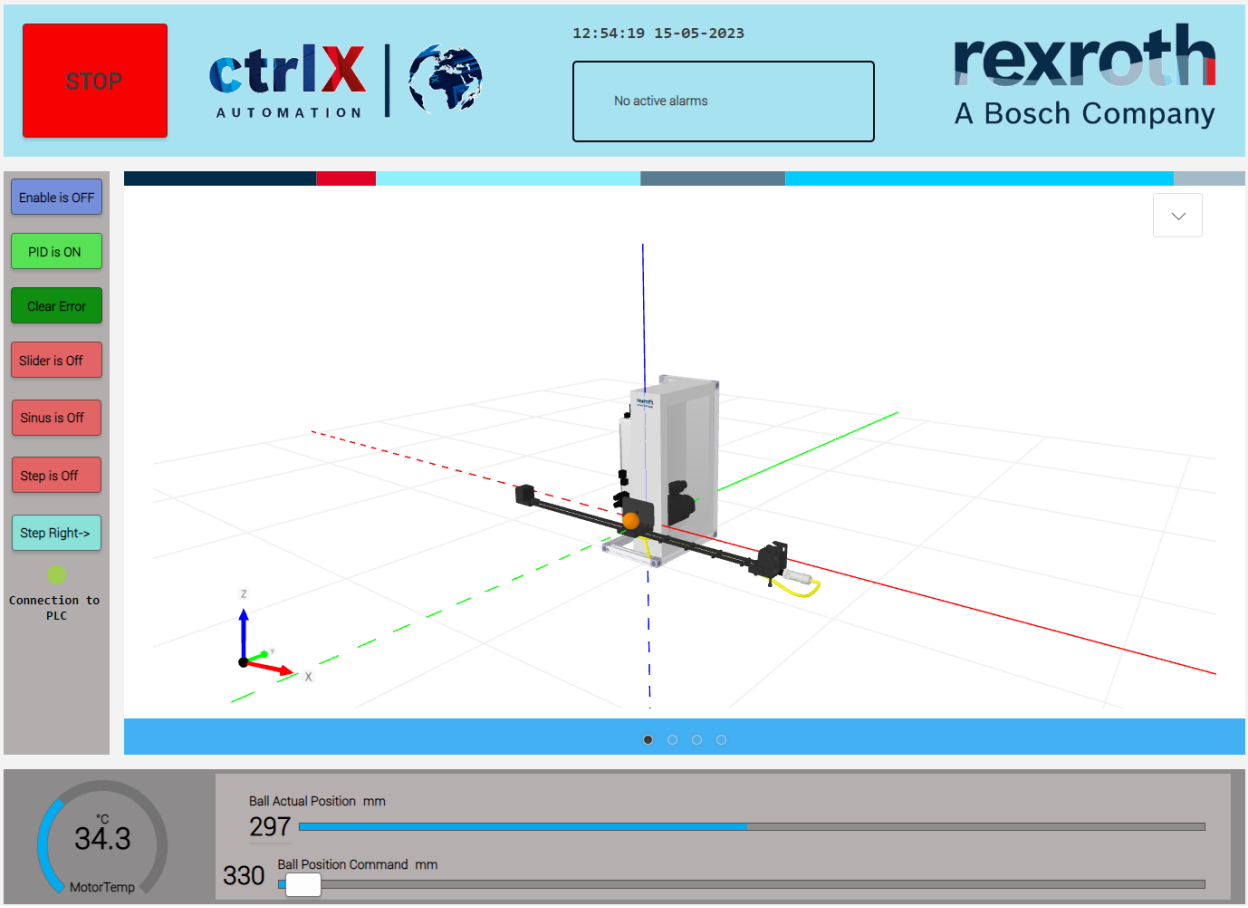

Figure 4.41: WebIQ HMI 1st page with 3d model

Fig. 4.41 shows the first page of the HMI. It displays the 3d- visual representation of the system. It moves in real-time following the actual position feedback signals from the system.

The buttons in the left vertical section do the following:

1. Enable is OFF/ON. Enables the drive. Turns off if the Emergency stop is activated, AxisError is activated, and after beam calibration has been performed. Must be on to move the beam
2. PID is OFF/ON. Enables the position and velocity controller PIDs. Turns off after beam calibration has been performed. Must be on to run Slider, Sinus, or Step functions.
3. Clear Error. Clears the drive error and resets the drive status light. Must be activated after a drive error. Activates automatically if AxisError is acknowledged in the alarm list on page 4.
4. Slider is OFF/ON. Activates the slider in the footer "Ball Position Command". Will turn Sinus and Step functions OFF if pushed while either of these is active. Will turn OFF if Emergency stop, beam calibration, or Alarms is activated
5. Sinus OFF/ON. Activates the sinusoidal movement of the beam. Amplitude, frequency, and duty values are set on page 2. Will turn OFF if Emergency stop, Alarms, beam calibration, or Slider is activated
6. Step OFF/ON. Activates the step function. Values for left and right position is set on page 2. Will turn OFF if Emergency stop, Alarms, beam calibration, or Slider is activated.
7. Step <-Left/Right->. Alternate between the values set on page 2.

The footer shows the motor temperature in degrees Celsius. The Slider sets a ball position command if the Slider function is active. The value shows the real-time position command value even if the slider function is OFF. The top bar shows the ball's actual position on the beam and the number displays the actual value.

The center display has a swipe functionality and will shift pages by swiping left or right. Pushing one of the circular dots at the bottom will also change to the corresponding page.

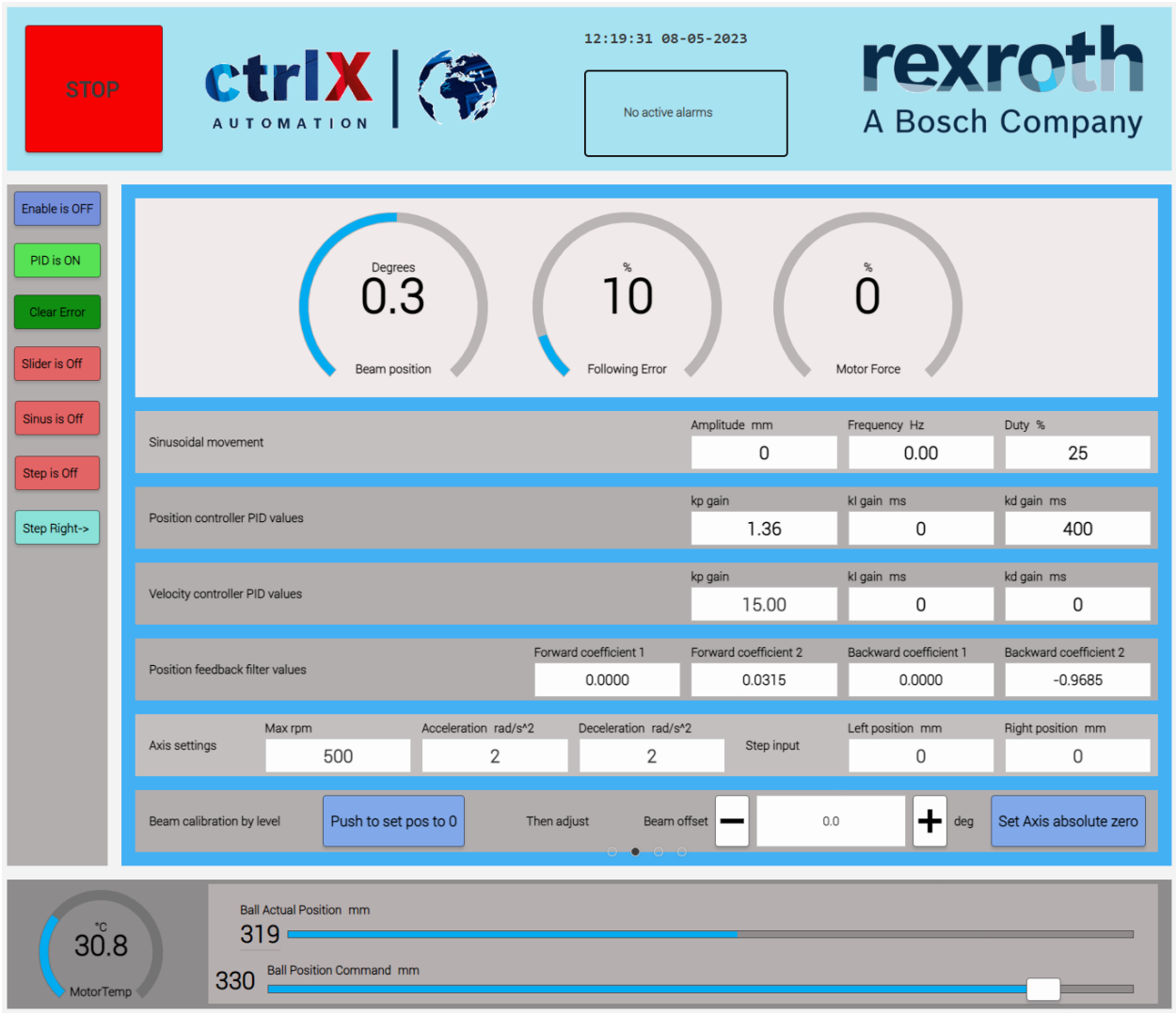

Figure 4.42: WebIQ HMI 2nd page with with dials and inputs

Page 2 in Fig. 4.42 shows the real-time values of the Beam position, following error and the motor nominal force in %.
The inputs set the values as described in the headings and are limited in the program as follows:

- Sinuoidal movement: Amplitude 0-400mm, Frequency 0-2Hz, Duty 25-75%
- Position controller PID values: kp 0-10, kI 0-10000, kd 0-2000
- Velocity controller PID values: kp 0-100, kI 0-10000, kd 0-2000
- Axis settings: Max rpm 0-1000, acceleration 0-27 rad/s^2, deceleration 0-27 rad/s^2
- Step input: Left position 0-400mm, Right position 100-500mm
- Beam offset: -20deg to 20deg

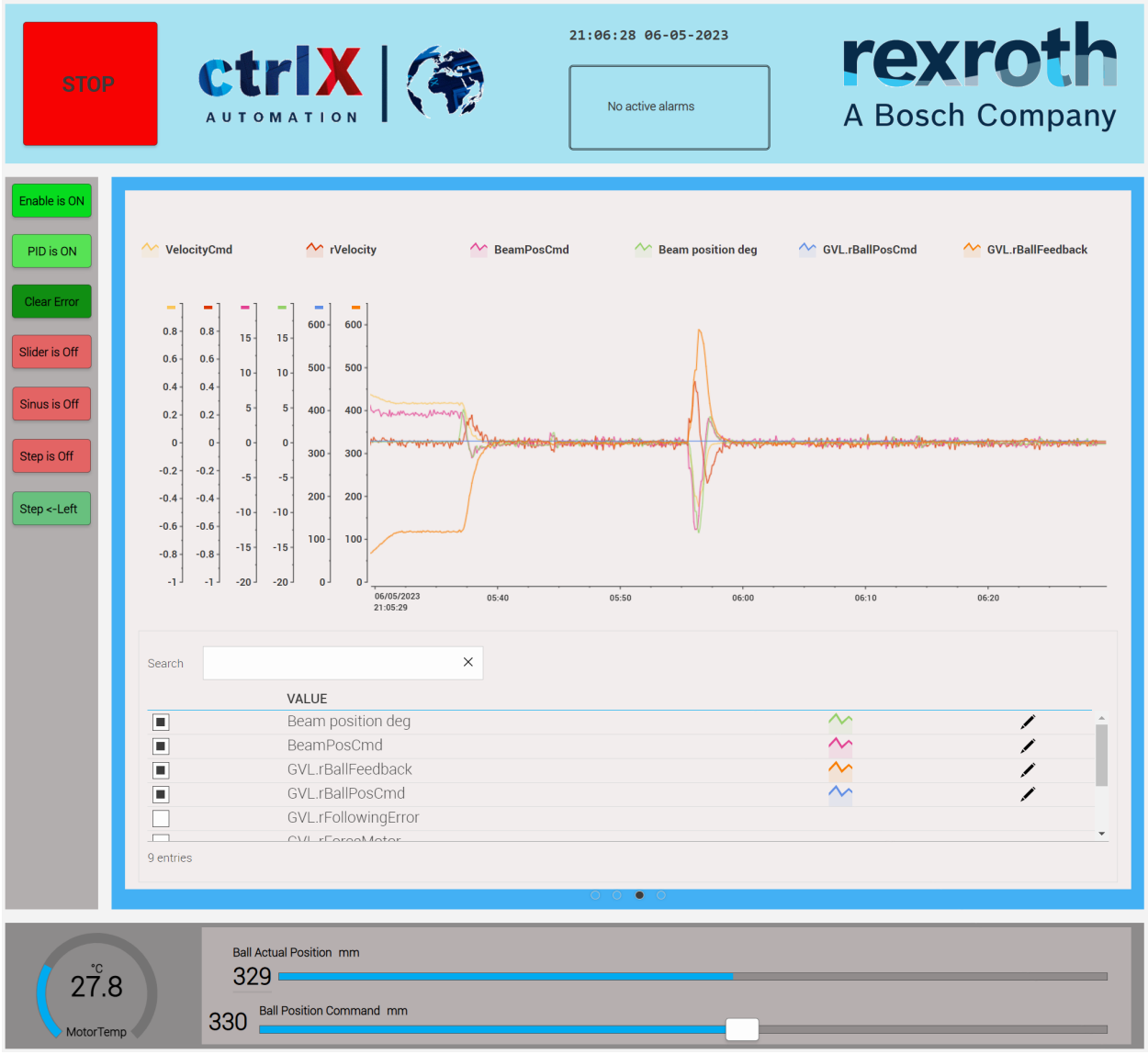

Figure 4.43: WebIQ HMI 3rd page with the trend, trend legend, and trend select

Page 3 in Fig. 4.43 show the trend of the following signals that can be switched on/off:

- Beam position deg, actual beam position in degrees
- BeamPosCmd, the beam position command
- GVL.rBallFeedback, actual ball position
- GVL.rBallPosCmd, the ball position command
- GVL.rFollowingError, the following error between ball position command and actual ball position
- GVL.rForceMotor, the actual force from the motor in % of nominal force
- GVL.rTempMotorCalc, the actual motor temperature in degrees Celsius
- VelocityCmd, Ball Velocity command
- rVelocity, actual ball velocity

Attention! The trend will not plot if the ctrlX Core is not time synchronized. Go to /settings/date&time/Assign-to-ctrlX-Core/ to assign the time to the controller.

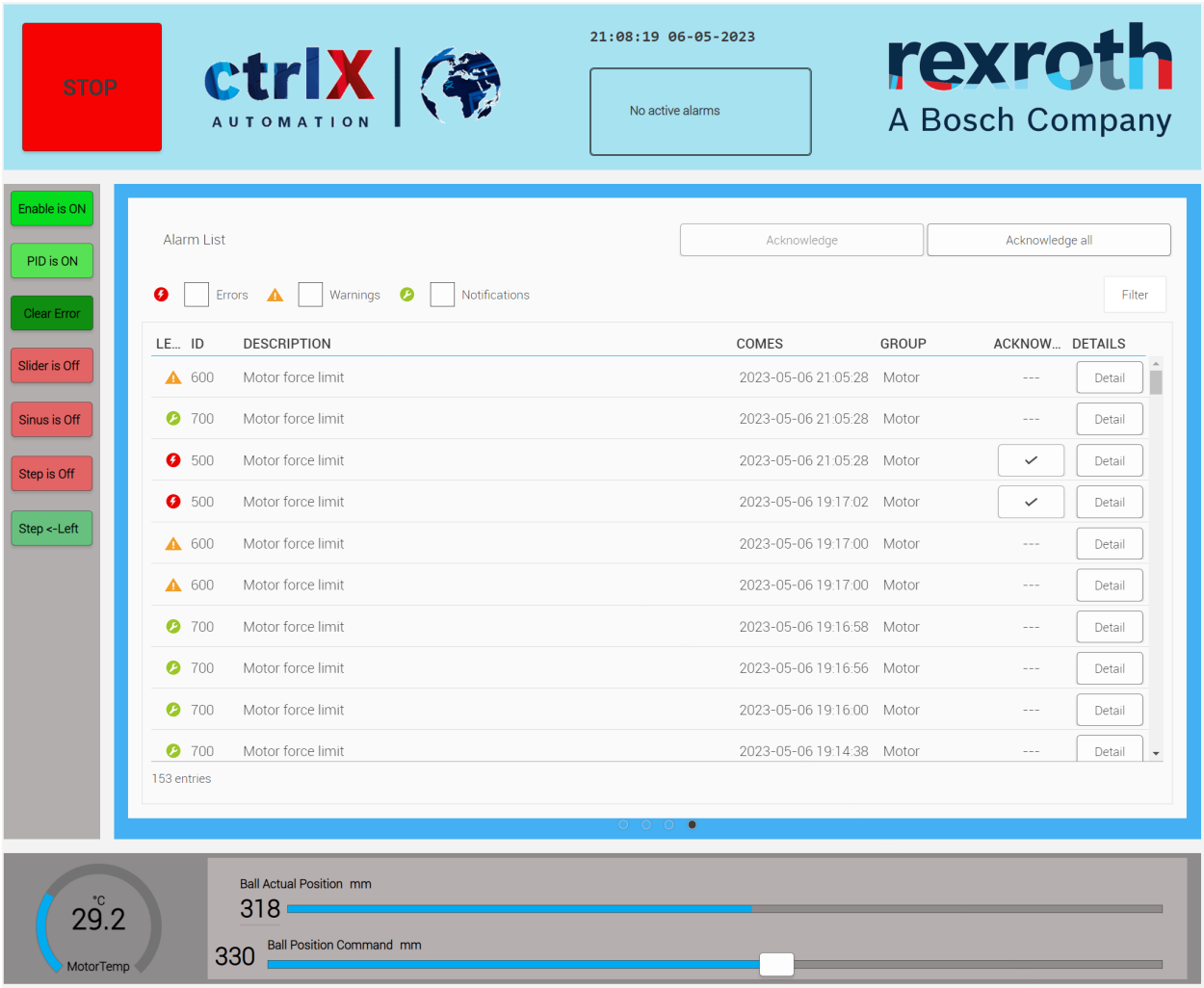

Figure 4.44: WebIQ HMI 4th page with alarms

Page 4 in Fig. 4.44 show the active and historical alarms. The alarms Emergency stop, AxisError, motor temperature and motor force limit has to be acknowledged. The alarms set the drive enable to 0 and will not set to 1 until the alarm has been acknowledged. The page also show warnings and notifications. These are informational. The following signals are monitored:

- Motor temperature: $>$ 120C Alarm(Sets enable=0), >80C Warning
- AxisError: Alarm (Sets enable=0)
- Emergency stop activated: Alarm (Sets enable=0)
- Motor force: $>$ 250% Alarm(Sets enable=0), >150% Warning, >100% Notification

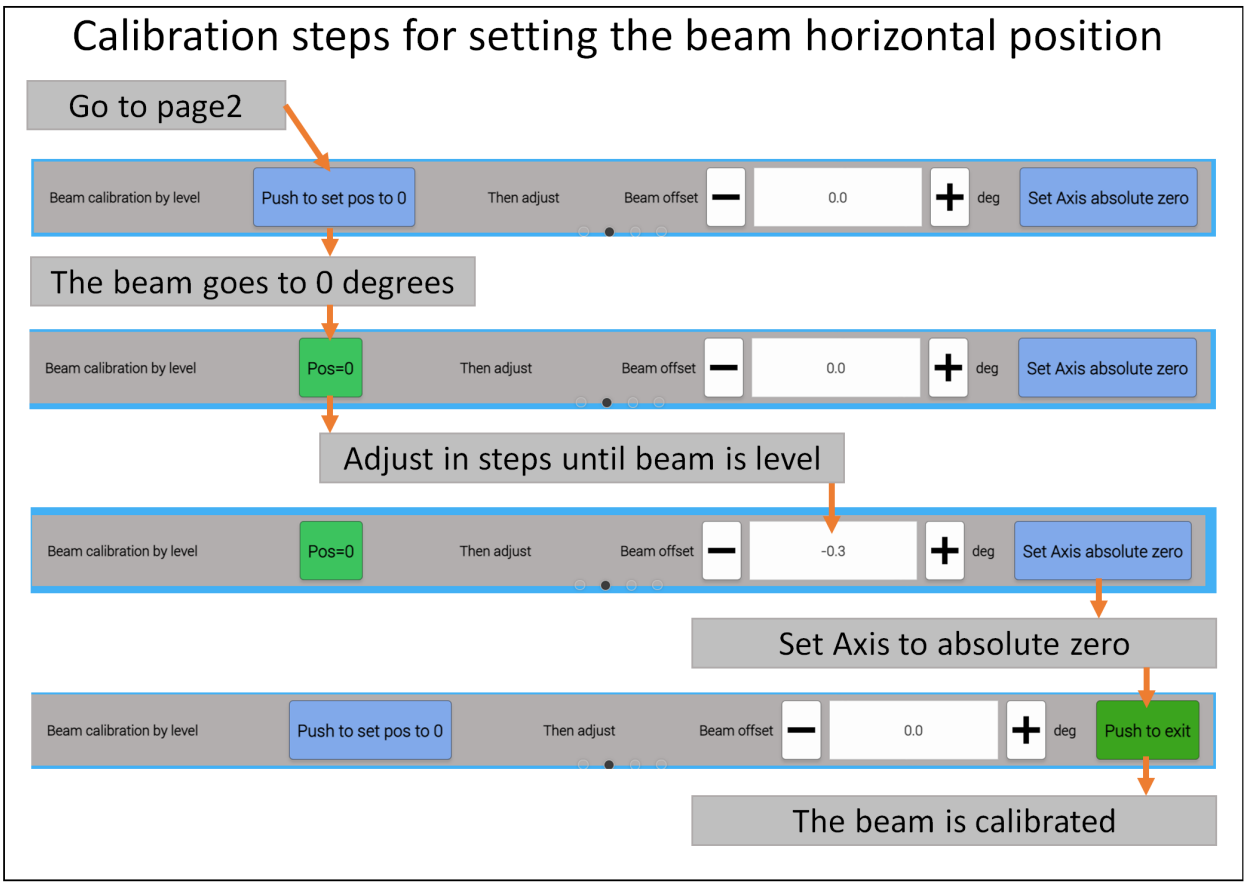

Figure 4.45: WebIQ HMI page 2, Beam Calibration procedure

The regualtion of the system is greatly influensed by the zeroing of the beam due to the lack of integration in the positioning loop. The axis is zeroed to horizontal beam position with a level by following the steps i Fig. 4.45.

4.5.2 Implementation of 3-D Model in the HMI

To include a model in the HMI, WebIQ was used to create an HMI. The intention is that the HMI should be able to be opened using a browser and can be operated with visual feedback without having to physically look at the system. In order to be able to communicate, the HMI was set up manually according to [6] under the HMI (steps 1, 2 and 12) so that the communication will work between the HMI and WebIQ.

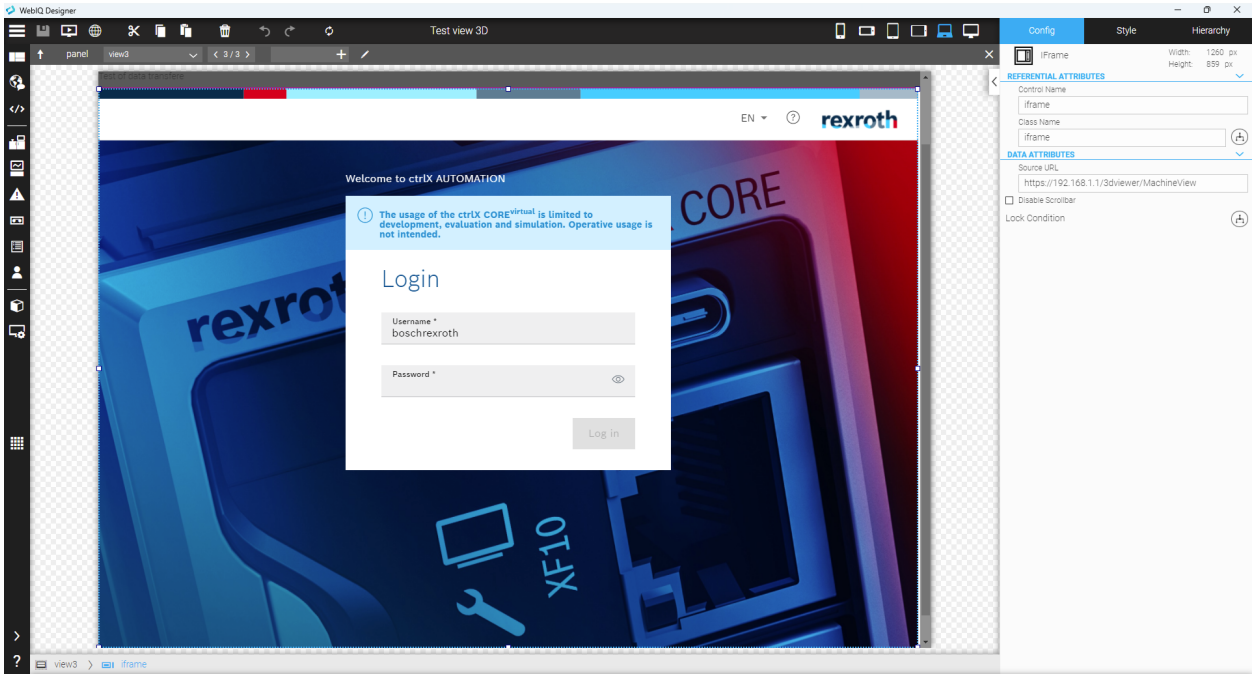

Figure 4.46: WebIQ iframe

The next step is to insert the link between the 3D viewer and the HMI. The link is obtained from CtrlX CORE under my model on the 3D viewer. This is inserted into WebIQ's iframe see Fig. 4.46. When this is published and one tries to run the HMI, an error message will appear stating that the certificate has not been approved or that the website has problems. The first step to being able to show 3D visualization of the machine is to use the right browser. There are 2 options Google Chrome and Mozilla Firefox. on these browsers, a notification will appear that the self-generated certificate (SSL) is not valid, this is because it has not been published online.

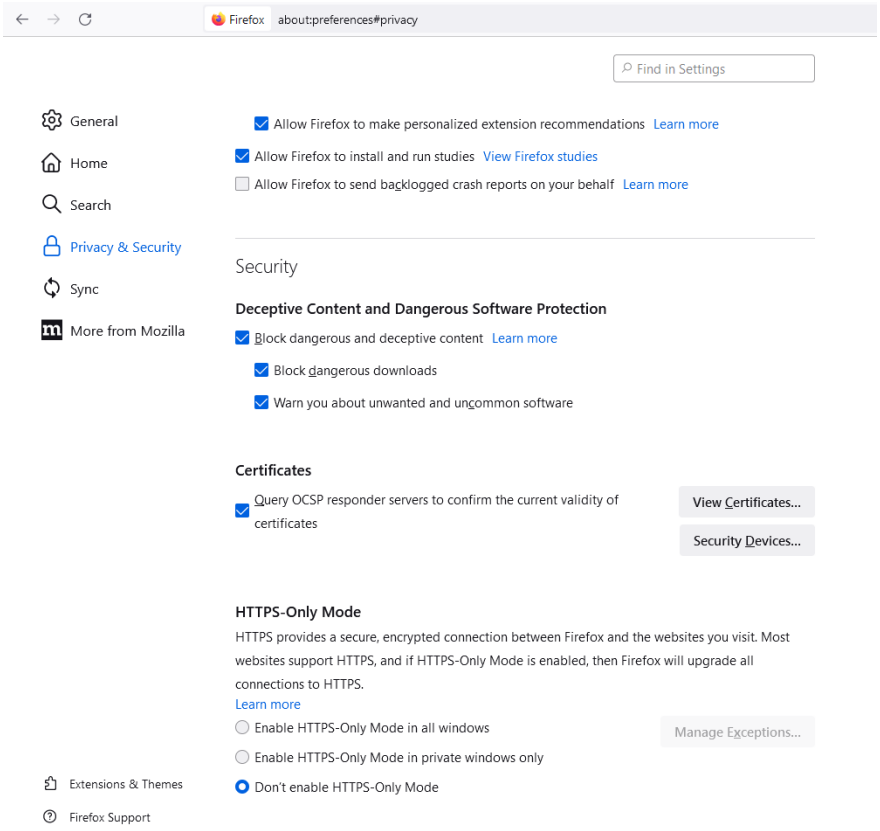

Figure 4.47: Mozilla firefox HTTPS

To be able to get past this on both Mozilla Firefox and Google Chrome, https must change to http so that WebIQ can display the 3D model. In Mozilla Firefox change this under settings/privacy & Security/HTTPS-Only Mode is set to: "Dont't enable HTTPS-Onlly Mode". see Fig. 4.47.

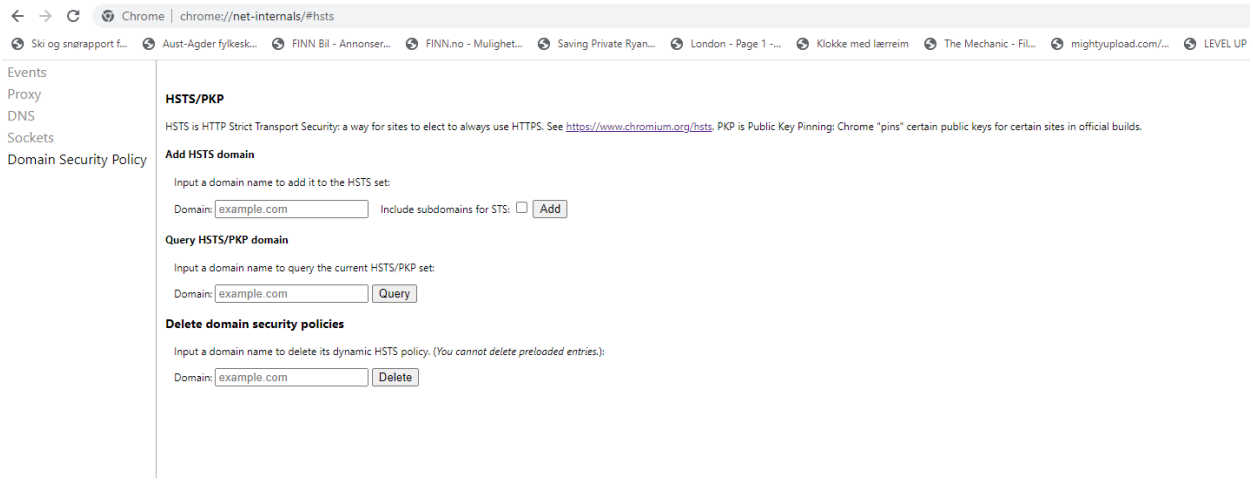

Figure 4.48: Google Chrome HTTPS to HTTP

Google Chrome automatically changes to https protocol, this must change to http. It can be done for the specific page as shown in Fig. 4.48. By inserting the link from the iframe see Fig. 4.46 `https://192.168.1.1/3dviewer/MachineView` link in **"Delete domain security policies"** and Delete. This is done by going to `chrome://net-internals/#hsts` in the search bar. Then the browser must be shut down and opened again, if this does not work, further options must be done as described below, see in the appendix. E.1.

Chapter 5

Results

5.1 Controller design

5.1.1 Ball Model

The ball properties were: Ball mass = 0.003kg, Ball radius = 0.02m. The ball transfer function from equation 2.31 with the ball constants yields:

$$\frac{X(s)}{\beta(s)} = \frac{5.886}{s^2} = P_{bb}(s) \tag{5.1}$$

5.1.2 System Identification

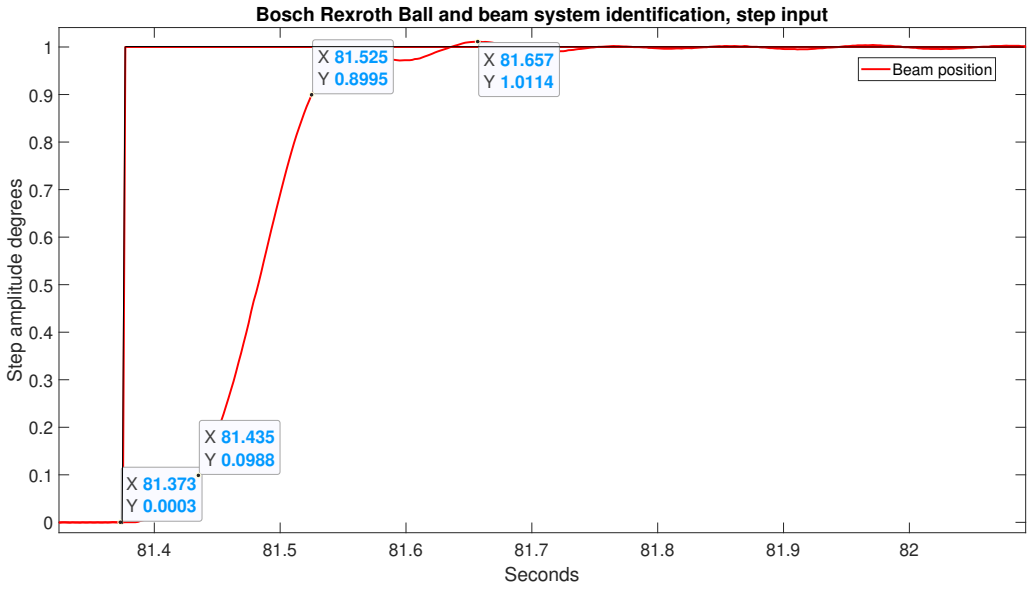

Figure 5.1: Drive/ motor/ beam step response

The system identification step input gave the following results:

- $y_{max} = 1.0114$
- $y_{ss} = 1$
- $T_r = 81.525 - 81.435 = 0.09s$
- $T_p = 81.657 - 81.373 = 0.284s$
- $K_a = \frac{1}{1} = 1$

Using eq. 2.52 to eq. 2.59 yields the drive/motor/beam transfer function:

$$P(s) = \frac{370.6}{s^2 + 31.51s + 370.6} \tag{5.2}$$

5.1.3 Lead Controller, Velocity Loop

The lead compensator was designed after the specification in 4.1.2 and ideal feedback without noise and low pass filter.
The desired phase margin, $\Phi_{desired}$, is

$$\Phi_{desired} = 70 - 5 = 65^\circ \tag{5.3}$$

The desired bandwidth, ω_{bw}, is

$$\omega_{bw} = \frac{1.8}{0.1} = 18[rad/s] \tag{5.4}$$

The actual phase margin is 62° and the gain margin is 14.6db at 19.2rad/s.

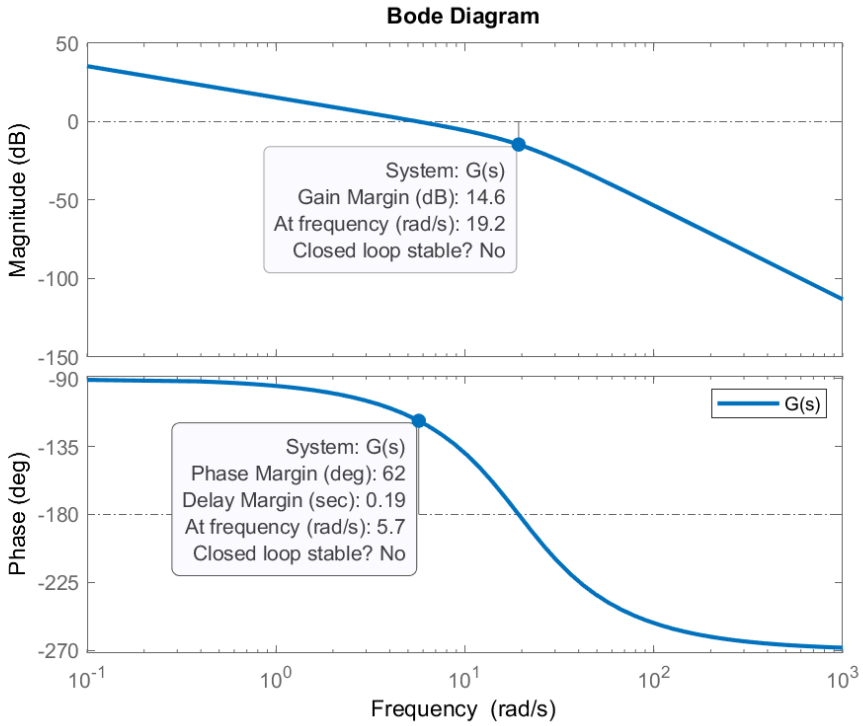

Figure 5.2: Velocity loop bodeplot

Finding the gain K with G(s), the drive/motor/beam and ball transfer function:

$$G(s) = P(s) * P_{bb}(s) * s = \frac{2181s}{s^2(s^2 + 31.51s + 370.6)} = \frac{2181}{s(s^2 + 31.51s + 370.6)} \tag{5.5}$$

It has a single integrator,s, so it is a type 1 system and will have an error for a ramp input. For a ramp input:

$$E(s) = \frac{1}{1 + \frac{2181}{s(s^2+31.51s+370.6)}} \frac{1}{s^2} \tag{5.6}$$

The final value theorem:

$$e_{ss} = e(\infty) = \lim_{s\to 0}(sE(s)) = \lim_{s\to 0}\left(\frac{s\frac{1}{s^2}}{1 + K\frac{2181}{s(s^2+31.51s+370.6)}}\right) = \lim_{s\to 0}\left(\frac{\frac{1}{s}}{1 + K\frac{2181}{s(s^2+31.51s+370.6)}}\right) = \tag{5.7}$$

$$\lim_{s\to 0}\left(\frac{s(s^2+31.51s+370.6)*\frac{1}{s}}{1*s(s^2+31.51s+370.6)+K*2181}\right)=\frac{370.6}{K*2181} \tag{5.8}$$

$\Rightarrow$

$$K=\frac{370.6}{e_{ss}*2181}=\frac{370.6}{0.1*2181}=1.699 \tag{5.9}$$

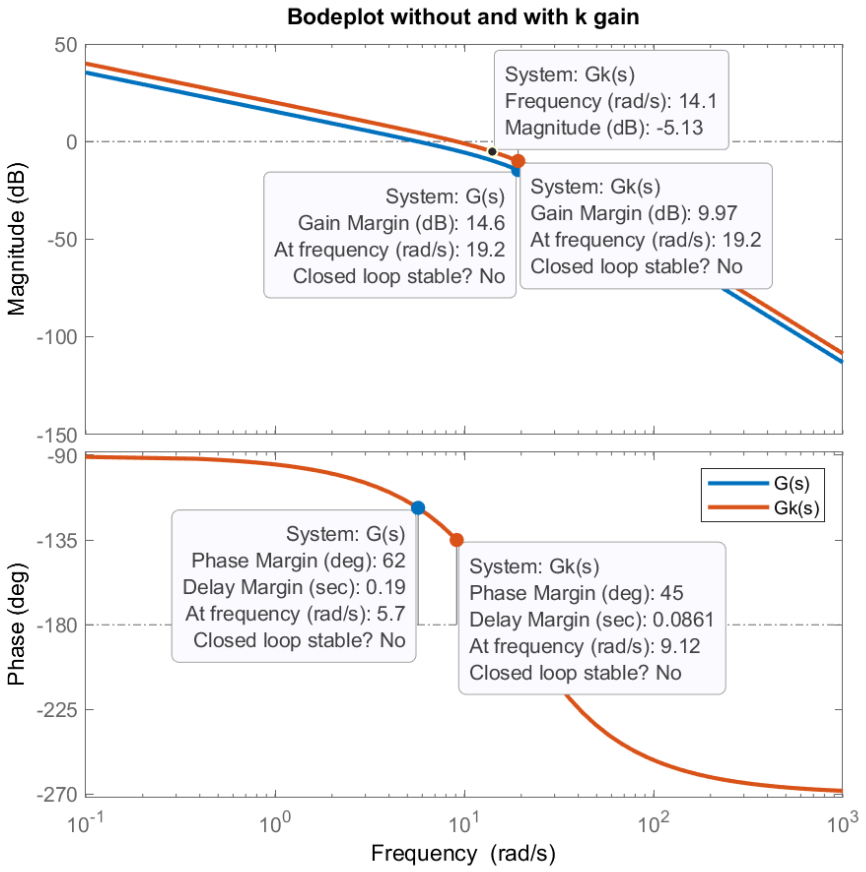

Figure 5.3: Velocity loop bodeplot with K Gain

A new plot with both G(s) and K*G(s)=Gk(s) shows the new phase and gain margins. 12° is added to the new phase margin $\Phi_{new}=45°$ and the required phase margin is calculated:

$$\Phi_m=(65-45)+12=32° \tag{5.10}$$

The value, a, can now be found:

$$a=\frac{1+\sin\Phi_m}{1-\sin\Phi_m}=3.2538 \tag{5.11}$$

Finding the frequency, ω_{cnew}, with the amplitude of the 0db crossing point:

$$KH_{Lead}=\frac{1}{\sqrt{a}}\Rightarrow 20\log(\frac{1}{\sqrt{a}})[db]=-5.12[db] \tag{5.12}$$

The frequency at -5.12db magnitude of Gk(s) is 14 rad/s, read from the plot. $\omega_m=14[rad/s]$ T is found:

$$T=\frac{1}{\omega_m\sqrt{a}}=0.0396 \tag{5.13}$$

The lead compensator is:

$$H_{lead}=K\frac{aTs+1}{Ts+1}=1.699\cdot\frac{0.1288s+1}{0.0396s+1} \tag{5.14}$$

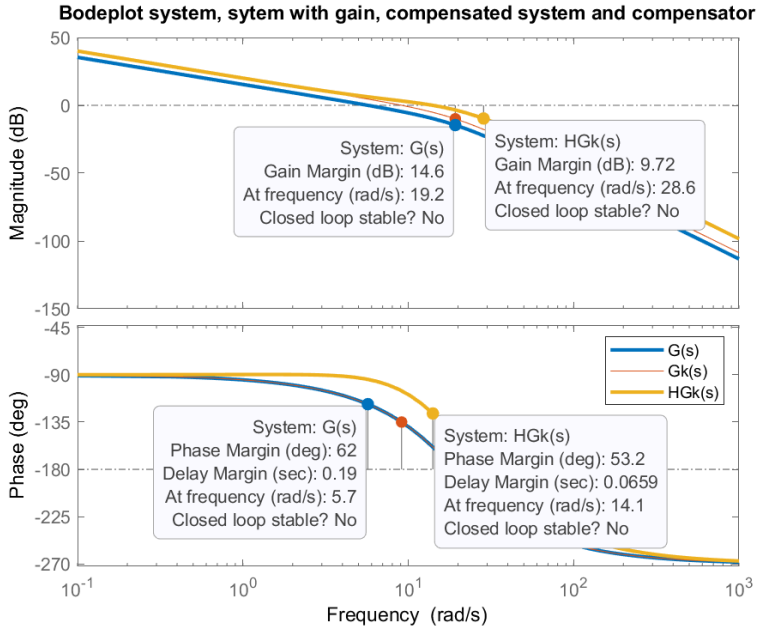

Figure 5.4: Velocity loop bodeplot with K Gain and lead compensator

The new bodeplot Fig. 5.4 show an increase in bandwidth of the compensated system, H_Gk_s from 19.2 [rad/s] to 28.6 [rad/s]. The same system has a decrease in phase margin from 62[deg] to 53.2[deg].

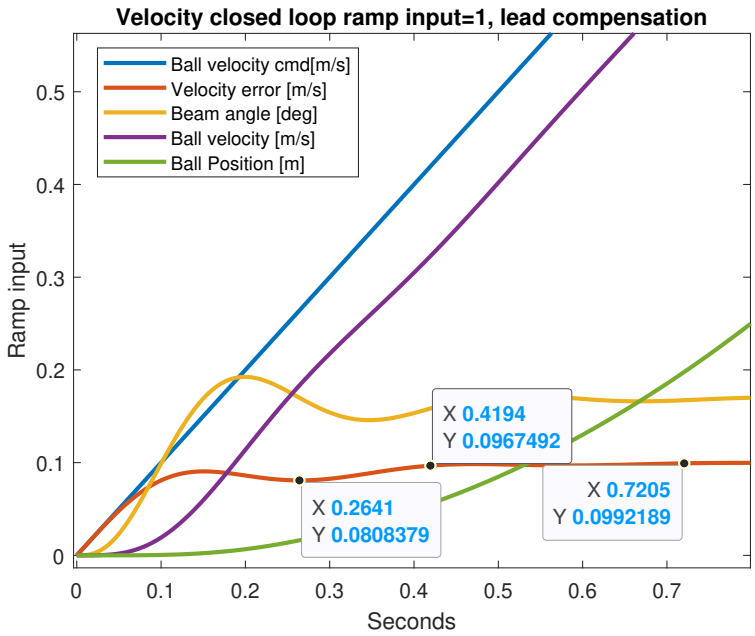

Figure 5.5: Velocity closed-loop plot with the lead compensator

Testing the velocity closed loop with the lead compensator in simulink with a ramp input with slope=1, shows that the ball velocity is at commanded speed in 0.15s. There is some instability showing overshoot at 0.26s and it is stable at 0.45s. The overshoot is $OS\%_{Lead} = \frac{(0.1-0.08084)*100}{(0.2641-0.1)} = 11.68[\%]$. Steady-state error, $e_{ssvelLeaderror} = 0.099 \approx 0.1$. Rise time, $T_{rvelLead} \approx 0.1s$. Settling time, $T_{svelLead} = 0.419s$

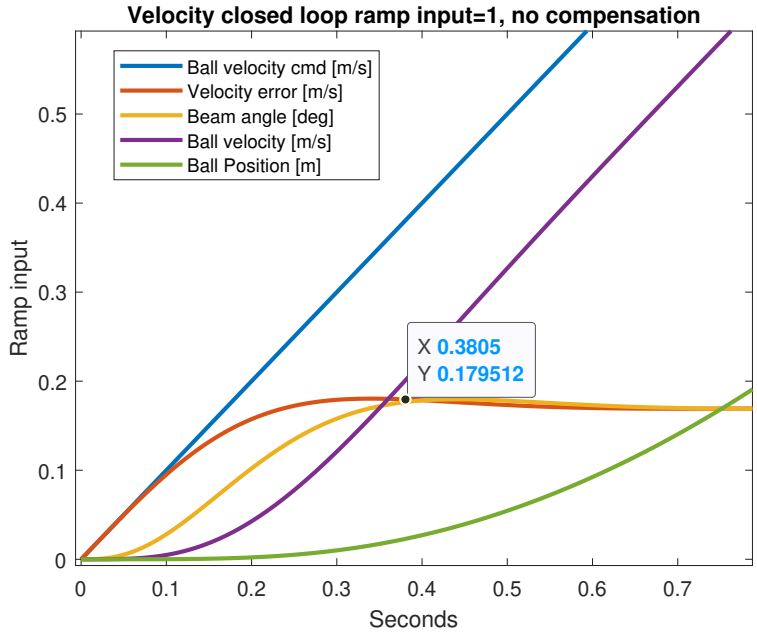

Figure 5.6: Velocity closed-loop plot without lead compensator

Comparing the compensated system to the uncompensated system, shows that the uncompensated system is more stable. Plot Fig. 5.6 shows that the ball velocity is at commanded speed in 0.38s and is stable at this point. Rise time, $T_{rvelLead} \approx 0.2s$. Steady-state error, $e_{ssvelerror} \approx 0.18$. Settling time, $T_{svel} \approx 0.4s$

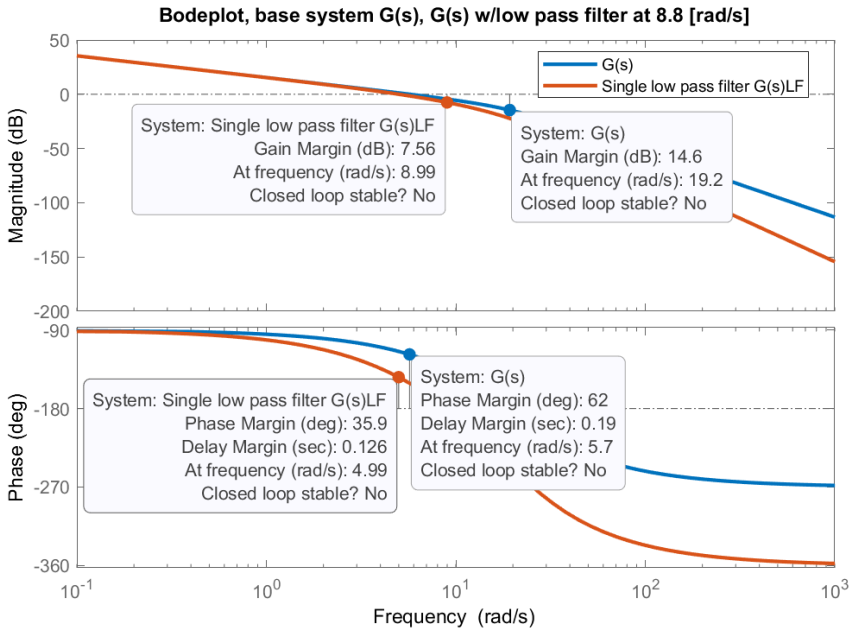

Figure 5.7: Velocity open loop bodeplot with original system G(s) and 1 low pass filter at 8.8$[rad/s]$

Adding a low-pass filter with a crossover frequency at $\omega_0 = 8.8$ rad/s (G(s)LF) to the original system G(s) shows the difference between phase delay and system delay bandwidth. The result is that adding a filter slows down the system by reducing the system bandwidth and phase margin. G(s) bandwidth is 19.2 [rad/s] while the G(s)LF is 7.56 [rad/s] as seen in Fig. 5.7

5.1.4 Ziegler-Nichols PID Tuning

After updating the simulink model Fig. 2.8 a new step input was given to the model for generating a steady oscillation for the Ziegler- Nichols tuning method. The models are without low pass filters in the feedback. The ultimate gain, $K_u = 8.2$, for steady oscillations, and the period, $P_u = 0.94s$,

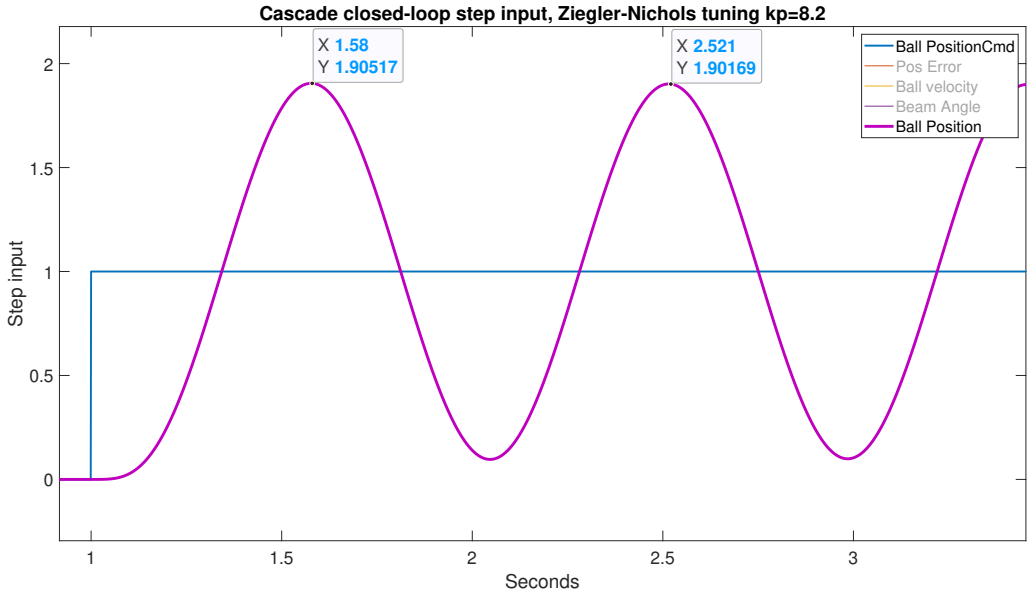

Figure 5.8: Cascade closed loop, steady oscillations, kp=8.2

were found from the plot in Fig. 5.8. Using the Tbl. 2.1 , yields $k_p = 0.8 * 8.2 = 6.56$ and $k_d = 0.10 * 8.2 * 0.94 = 0.77$.

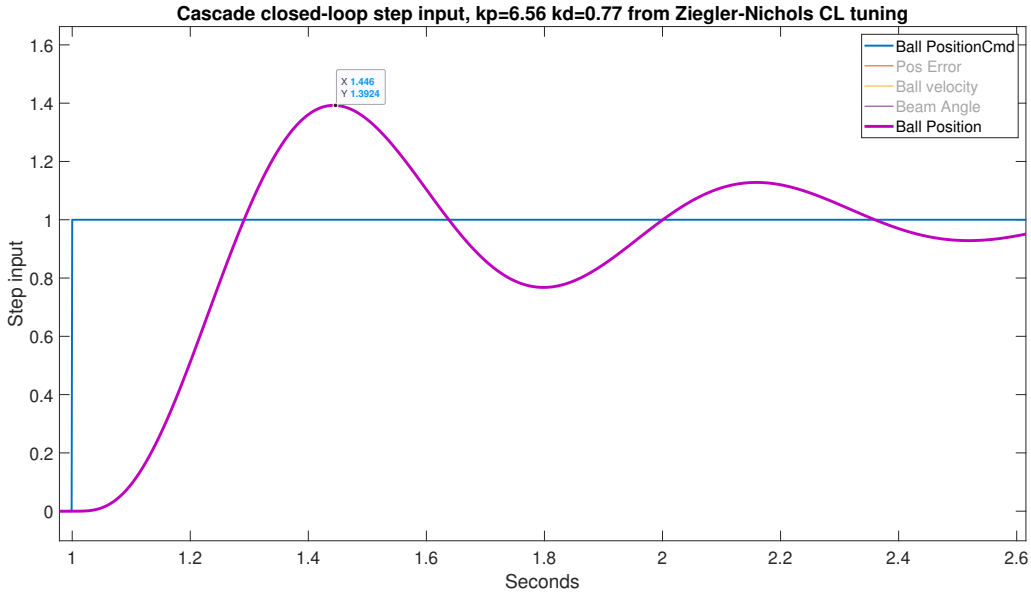

Figure 5.9: Cascade closed loop, Ziegler-Nichols results, kp=6.56, kd=0.77

Applying the calculated Ziegler- Nichols values results in a 39 [%] overshoot, Fig. 5.9.

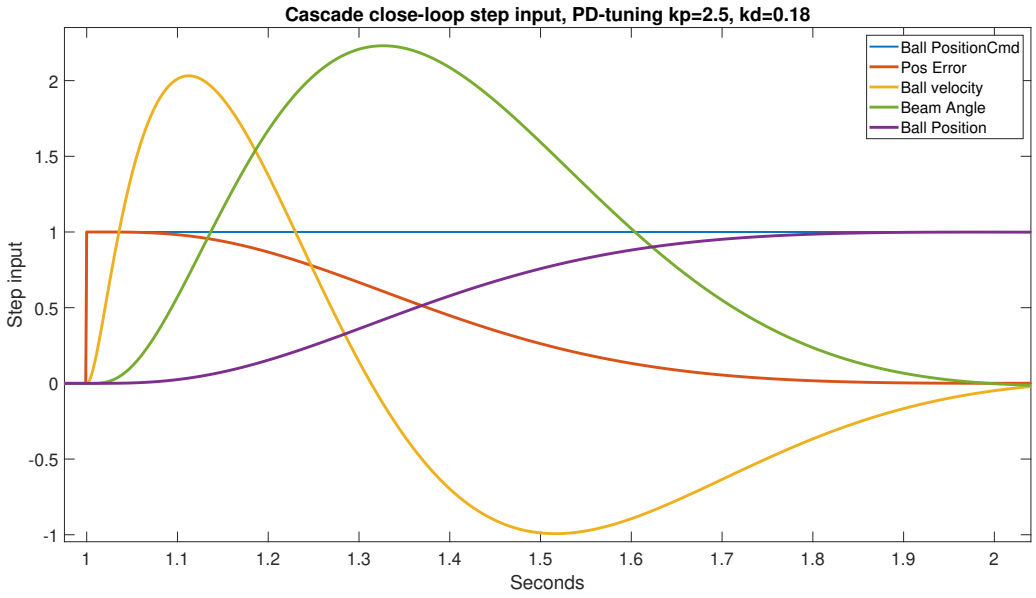

Figure 5.10: Cascade closed loop, manual tuning, kp=2.5, kd=0.18

Reducing the values to kp=2.5 and kd=0.18, by manual tuning, resulted in the response in Fig. 5.10.

5.2 Real System Response

The ball and beam system was used to get the results in this section.

5.2.1 Beam and Position Control Loop Response

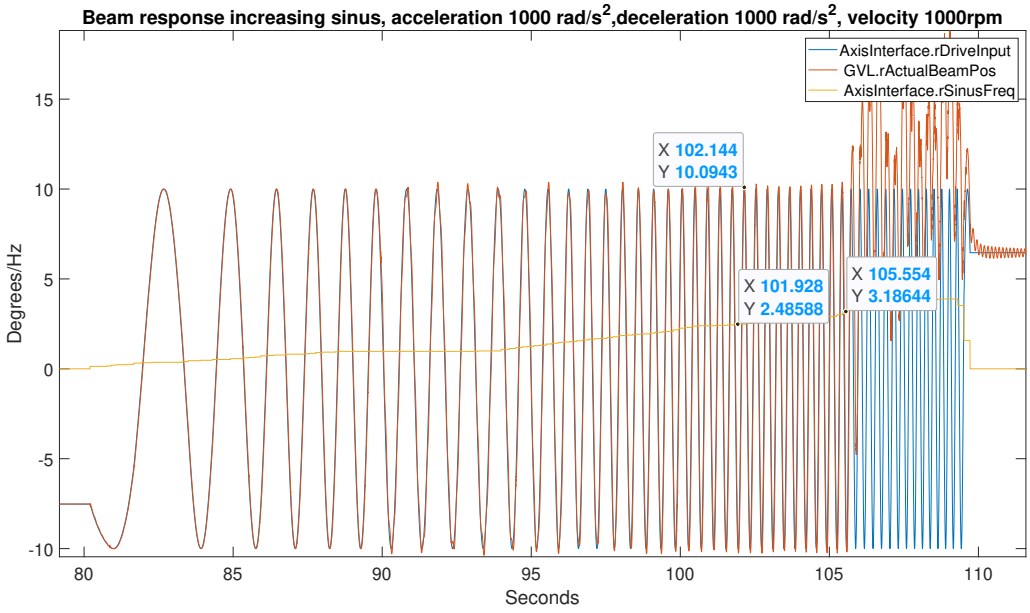

Figure 5.11: Testing the beam dynamics limit with 1000$[rad/s^2]$ Acceleration and deceleration, max velocity=1000 [rpm]

The beam was given a sinus signal with increasing frequency and constant amplitude at ±10 [deg]. Testing the beam without ball and axis acceleration and deceleration set at 1000 $[rad/s^2]$. Maximum axis speed set at 1000 [rpm]. The beam started to fall off the input signal at 2.49 [Hz] before it got unstable at 3.19 [Hz] as seen in Fig. 5.11. The actual beam mount to the shaft had moved 45 [deg] after the test.

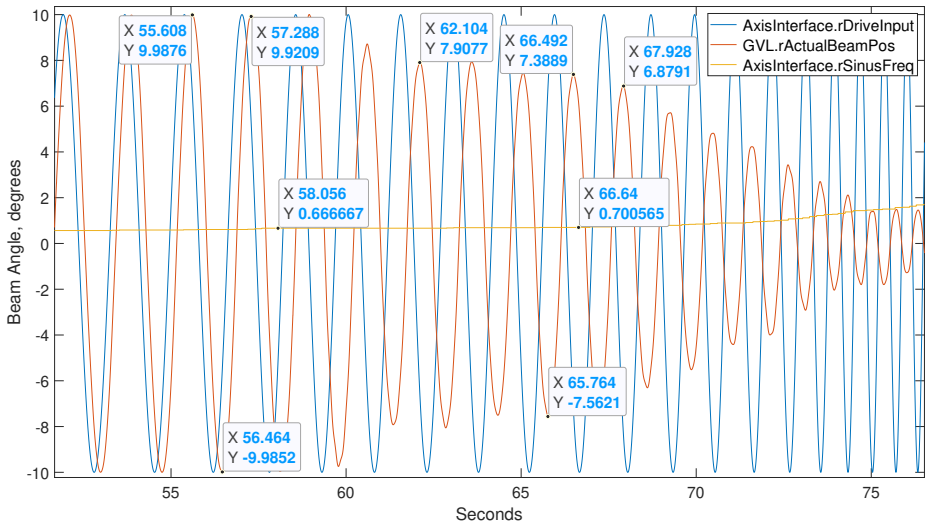

Figure 5.12: Testing the beam dynamics limit at 2 $[rad/s^2]$ and max 500 [rpm]

Testing the ball and beam positioning at $2[rad/s^2]$ Acceleration and deceleration, max velocity=500[rpm]. The beam started to fall off the input signal at 0.66 [Hz]. The signal dropped to 70[%] of the low-frequency amplitude at 0.7 [Hz] which gives a system bandwidth of $Beam_{bw} = 0.7\frac{1}{s} * 2\pi = 4.4[rad/s]$. The angular velocity at 56.464s to 57.288s is $\alpha_{beam} = \frac{0.349[rad]}{0.824s} = 0.42[rad/s]$.

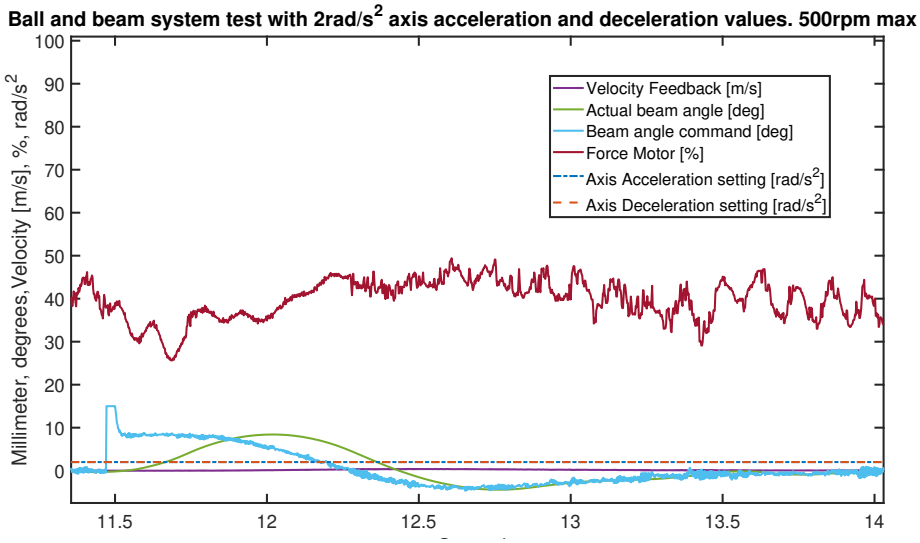

Figure 5.13: Testing the ball and beam positioning at $2[rad/s^2]$ Acceleration and deceleration, max velocity=500rpm

Fig. 5.13 shows a motor force ranging between approximately 25 to 50 [%] of max nominal torque. The beam position command has some noise that is reflected in the ForceMotor plot. Axis acceleration and deceleration are set at 2 $[rad/s^2]$. Maximum axis speed set at 500 [rpm].

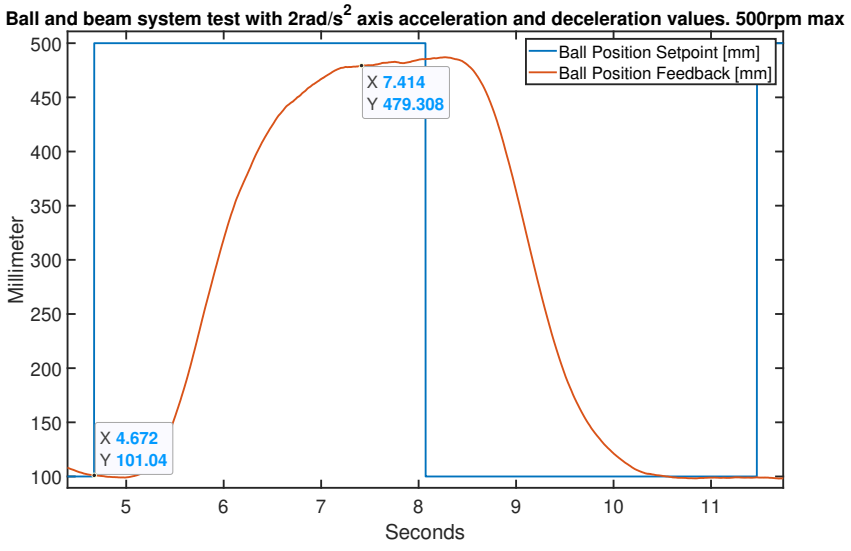

Figure 5.14: Testing the ball and beam positioning at $2[rad/s^2]$ Acceleration and deceleration, max velocity=500 [rpm]

The ball used $7.414 - 4.672s = 2.742s$ in a 400 [mm] step with a steady state error at $SS_{error} = 100 * \frac{(500-479[mm])}{400[mm]} = 5.25[\%]$ as shown in Fig. 5.14.

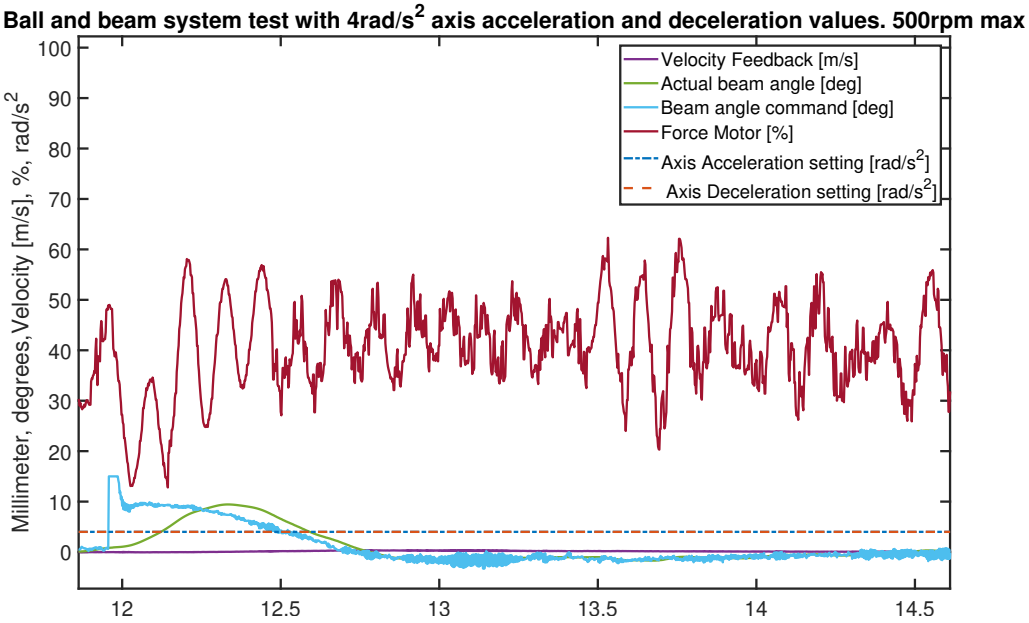

Figure 5.15: Testing the ball and beam positioning at $4[rad/s^2]$ Acceleration and deceleration, max velocity=500 [rpm]

Fig. 5.15 shows a motor force ranging between approximately 20 to 60 [%] of max nominal torque. The beam position command has some noise that is reflected in the ForceMotor plot. Axis acceleration and deceleration set at $4[rad/s^2]$. Maximum axis speed set at 500 [rpm].

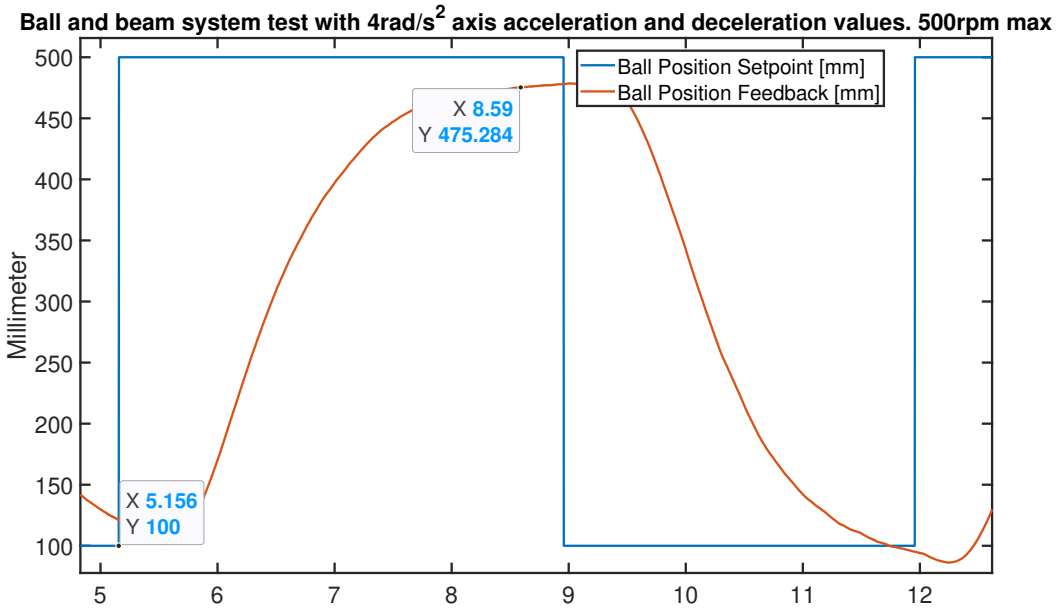

Figure 5.16: Testing the ball and beam positioning at $4[rad/s^2]$ Acceleration and deceleration, max velocity=500 [rpm]

The ball used $8.59 - 5.156s = 3.434s$ in a 400 [mm] step with a steady state error at $SS_{error} = 100 * \frac{(500-475[mm])}{400[mm]} = 6.25\%$ as shown in Fig. 5.16.

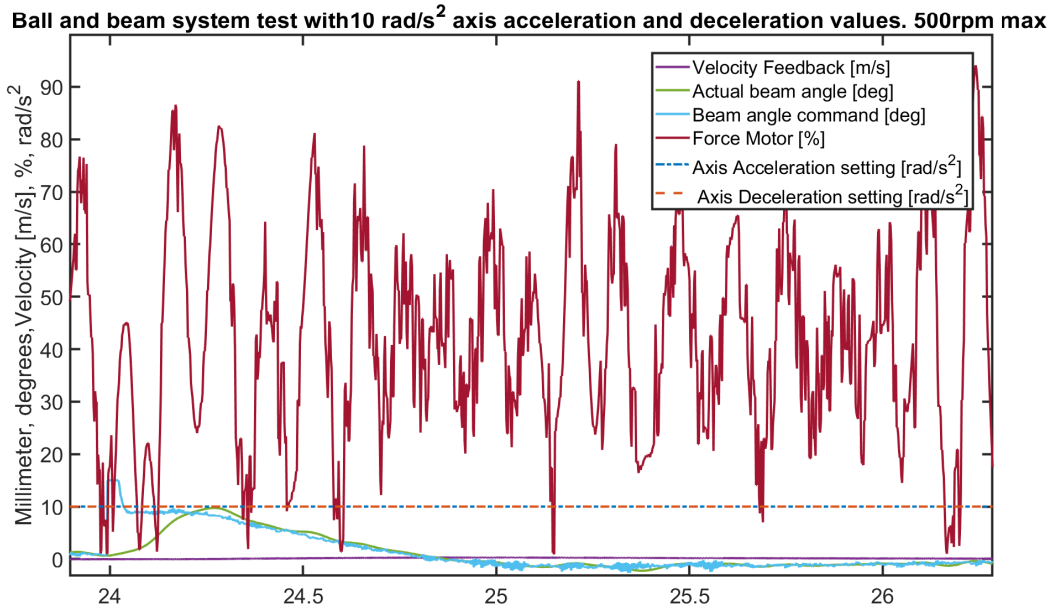

Figure 5.17: Testing the ball and beam positioning at $10[rad/s^2]$ Acceleration and deceleration, max velocity=500 [rpm]

Fig. 5.17 shows a motor force ranging between approximately 0 to 90 [%] of max nominal torque. The beam position command has some noise that is reflected in the ForceMotor plot. Axis acceleration and deceleration set at $10[rad/s^2]$. Maximum axis speed set at 500 [rpm].

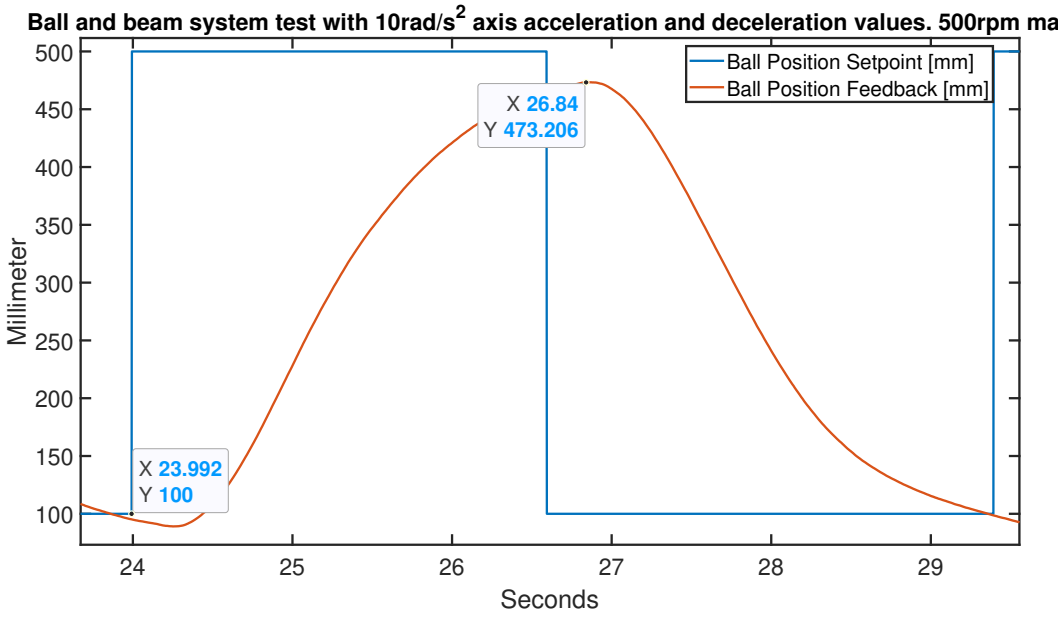

Figure 5.18: Testing the ball and beam positioning at $10[rad/s^2]$ Acceleration and deceleration, max velocity=500 [rpm]

The ball used $26.84 - 23.992s = 2.848s$ in a 400 [mm] step with a steady state error at $SS_{error} = 100 * \frac{(500-473[mm])}{400[mm]} = 6.75[\%]$ as shown in Fig. 5.18.

5.2.2 Filter Effects

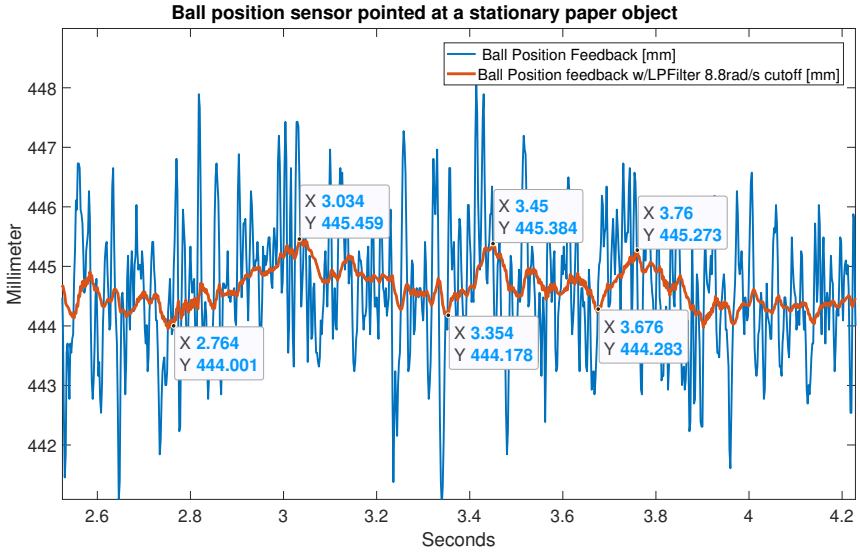

Figure 5.19: Position sensor signal at stationary paper

The Balluff distance sensor has a signal that oscillates. The red plot shows a low-pass filtered signal at 8.8 [rad/s] crossover. It shows in Fig. 5.19 that the amplitude is 1.6 mm and the period is 0.88s. The raw signal noise (Ball Position Feedback) has an amplitude of 3 to 5 [mm]. The values are an approximation and change but act as an indication. The enable was off and the drive was parked.

The position signal has noise and was filtered with a low-pass filter, at 2 times the $Beam_{bw}$, $LF = \frac{8.8}{s+8.8}$ and double low-pass filter $DLF = \frac{8.8}{s+8.8} * \frac{8.8}{s+8.8} = \frac{1239}{s^2+70.4s+1239}$ to visualize the filtering effect and introduced phase lag.

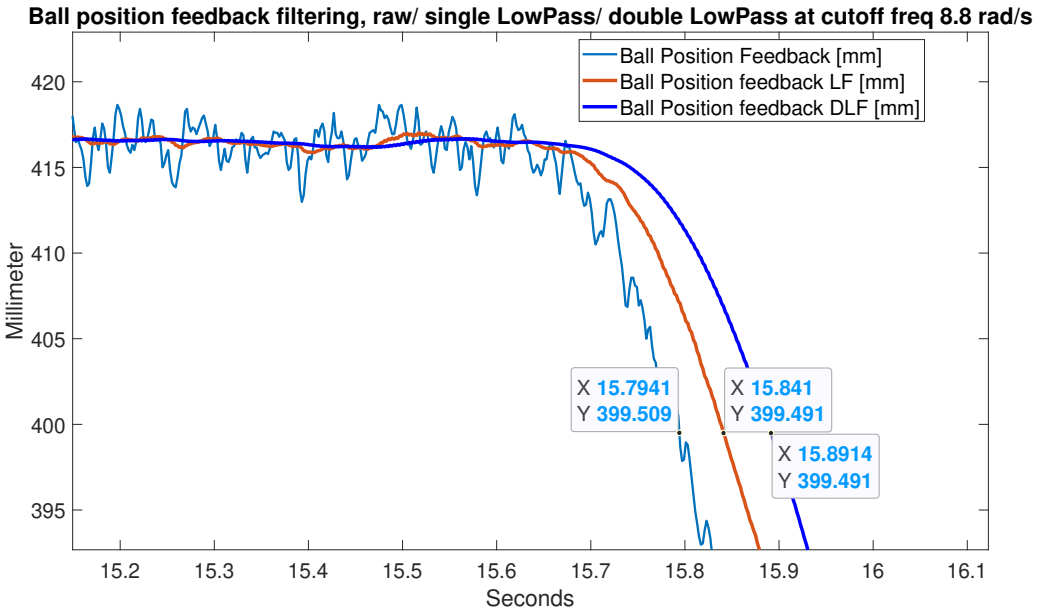

Figure 5.20: Testing the effects of low pass filters on the raw position signal

The results are seen in Fig. 5.20 where LF has a 47 [ms] phase lag and DLF has a 97 [ms] phase lag.

5.2.3 Velocity Loop Tuning

Velocity Signal Filtering

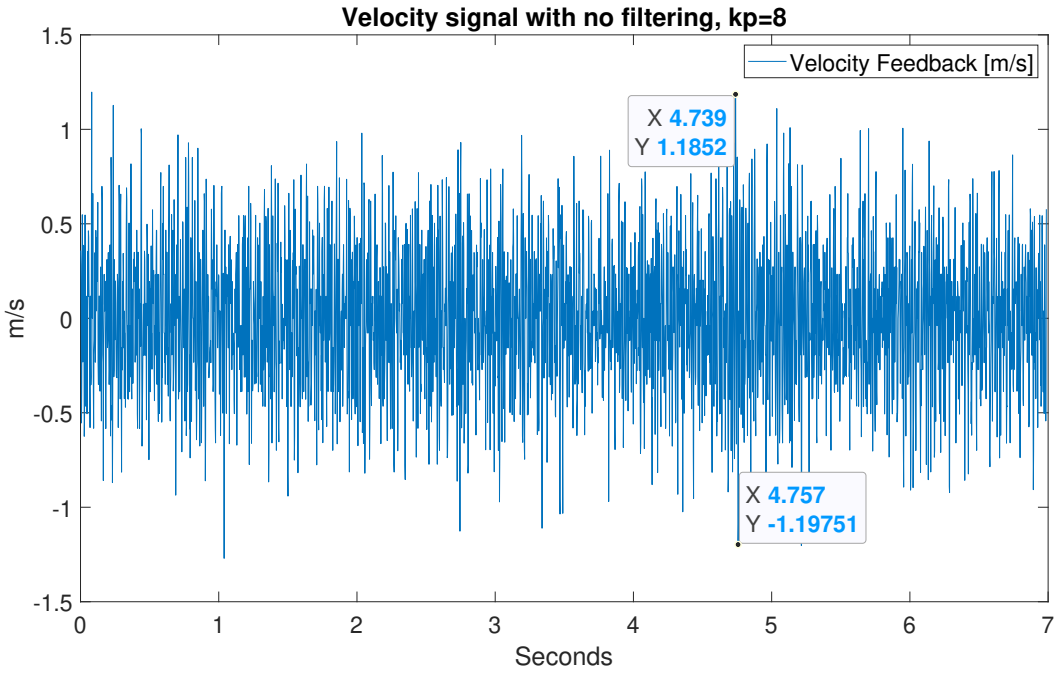

Figure 5.21: Velocity signal without filtering

The velocity feedback signal has noise in the range ±1 [m/s].

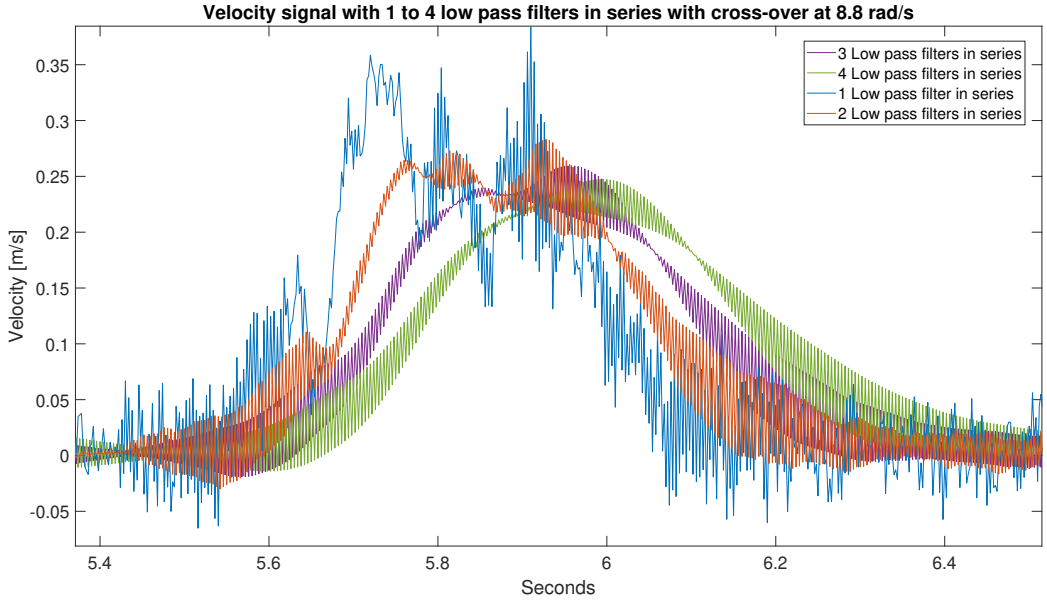

Figure 5.22: Velocity signal with 1-, 2-, 3- and 4 low-pass filters at crossover=8.8rad/s

Trying to see the different effects of where to filter the position and velocity signals show that it does not matter if the position signal is filtered or the velocity signal is filtered. Filtering the input signal to the velocity calculation or filtering the velocity signal yields the same results. The amount of filters in series has an effect on the filtering as seen in Fig. 5.22 where the increasing number of filters reduces the noise but increases the phase lag.

Velocity Loop Tuning

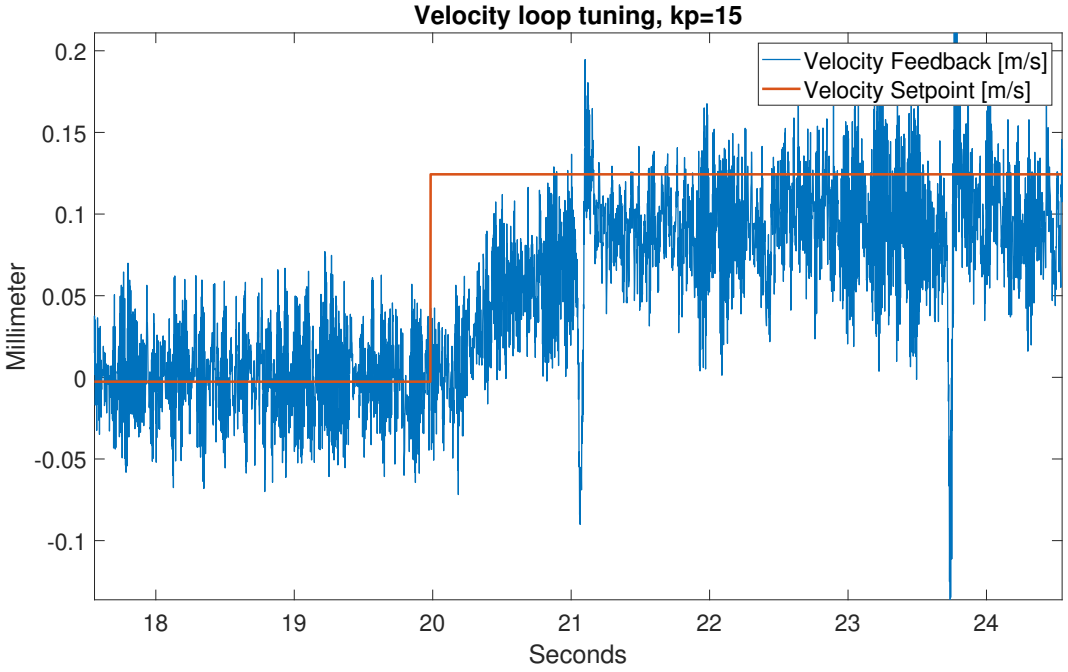

Figure 5.23: Velocity loop with position signal filter and kp=15

The position signal was filtered with one low pass filter, $LF = \frac{8.8}{s+8.8}$, at crossover 8.8 [rad/s]. Using a gain of kp=15 for the closed loop resulted in the step response as seen in Fig. 5.23. The gain was lowered from 25 which showed to be too aggressive.

5.2.4 Cascade Loop

The values found in the velocity loop tuning were used and the position loop was added to form the cascade loop.

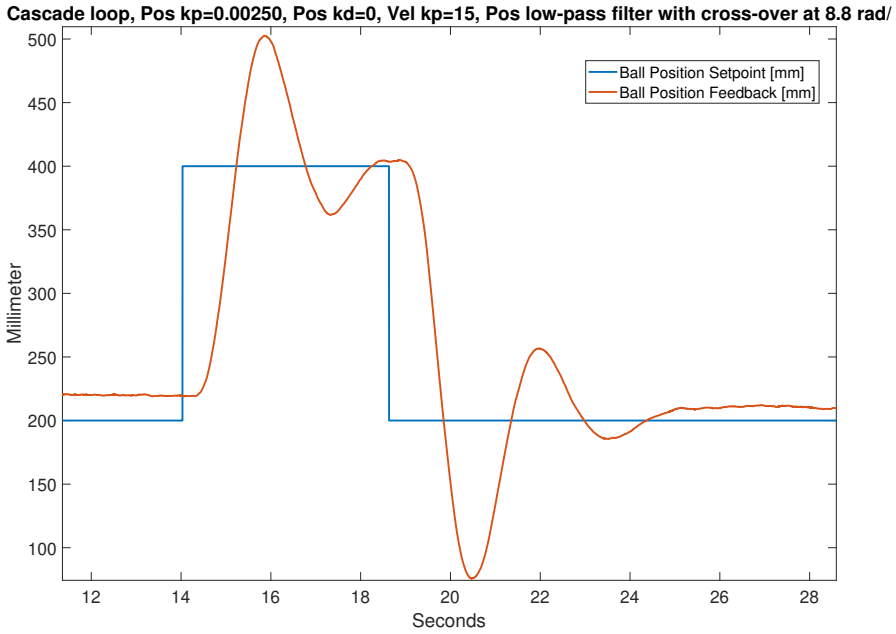

Figure 5.24: Cascade loop Pos PID kp=0.00250 kd=0, Vel kp=15, PosLowPass filter at 8.8 [rad/s]

The cascade loop is stable with only kp in the velocity loop and position loop. In Fig. 5.24 the system is given a step input and the ball is showing stable behavior.

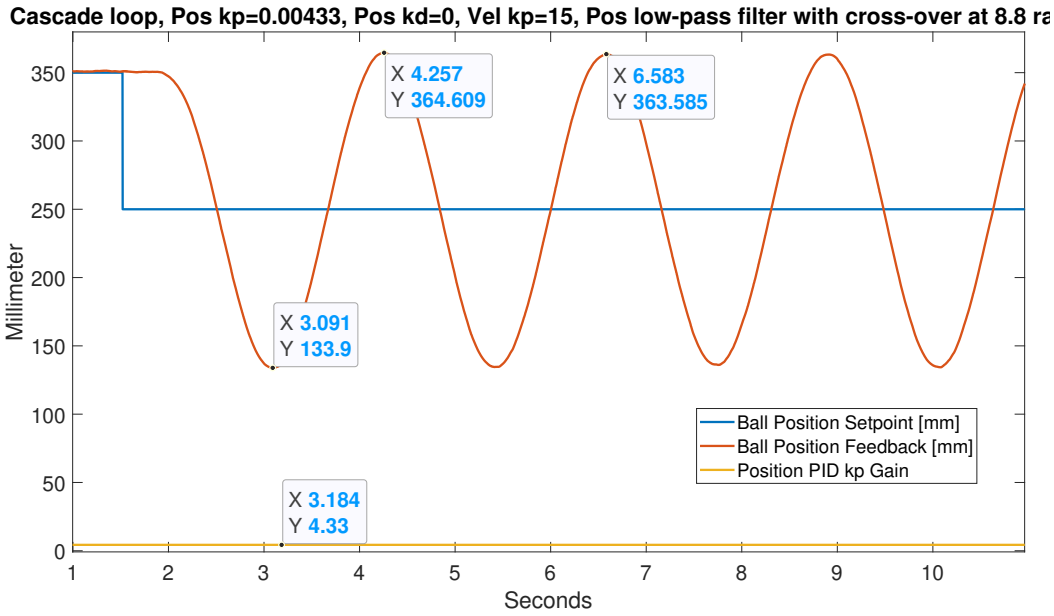

Figure 5.25: Cascade loop Ziegler-Nichols tuning, PosPID kp=0.00433 kd=0, Vel kp=15, PosLow-Pass filter at 8.8rad/s

The ultimate gain, $K_u = 0.00433$, for steady oscillations, and the period, $P_u = 6.583 - 4.257s = 2.326s$, were found from the plot in Fig. 5.8. Using the table 2.1 for a PD Type, yields $k_p = 0.8 * 0.00433 = 0.00346$ and $k_d = 0.10 * 0.00433 * 2.326 = 0.00101$.

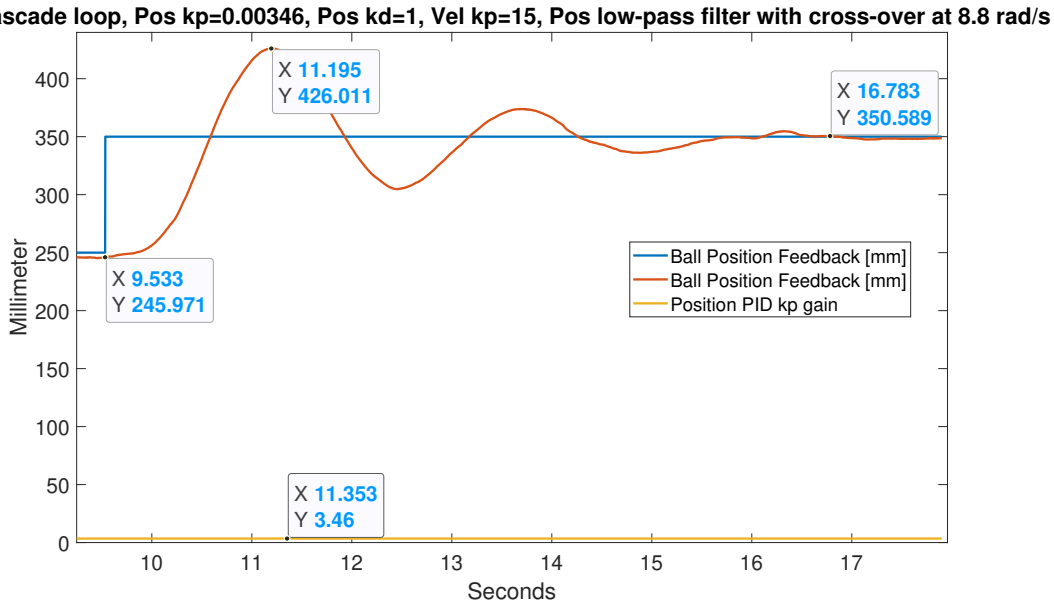

Figure 5.26: Cascade loop Ziegler-Nichols tuning, PosPID kp=0.00346 kd=0.001, Vel kp=15, PosLowPass filter at 8.8rad/s

Applying the calculated Ziegler-Nichols values found in Fig. 5.25 to the controller and testing gives the plot found in Fig. 5.26. The overshoot is $OS\% = 100 * \frac{((426-246)-100[mm]}{100[mm]} = 80[\%]$.

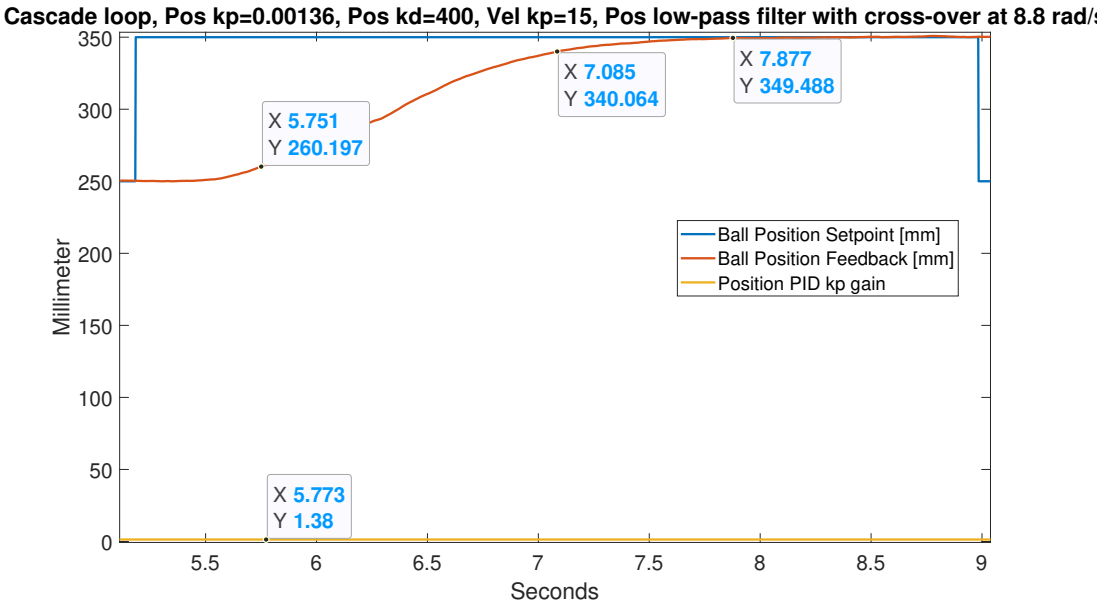

Figure 5.27: Cascade loop, 100 [mm] step, PosPID kp=0.00136 kd=400, Vel kp=15, PosLowPass filter at 8.8 [rad/s]

Using a step input with an amplitude 100mm shows the step response of the cascade system as shown in Fig. 5.27. The system has no over- or undershoot. The rise time is $T_r = (7.085-5.751)s = 1.334s$. The settling time is $(7.877-5.183)s = 2.694s$. The PID values were: position controller kp=0.00136, kd=400ms, velocity controller kp=15.

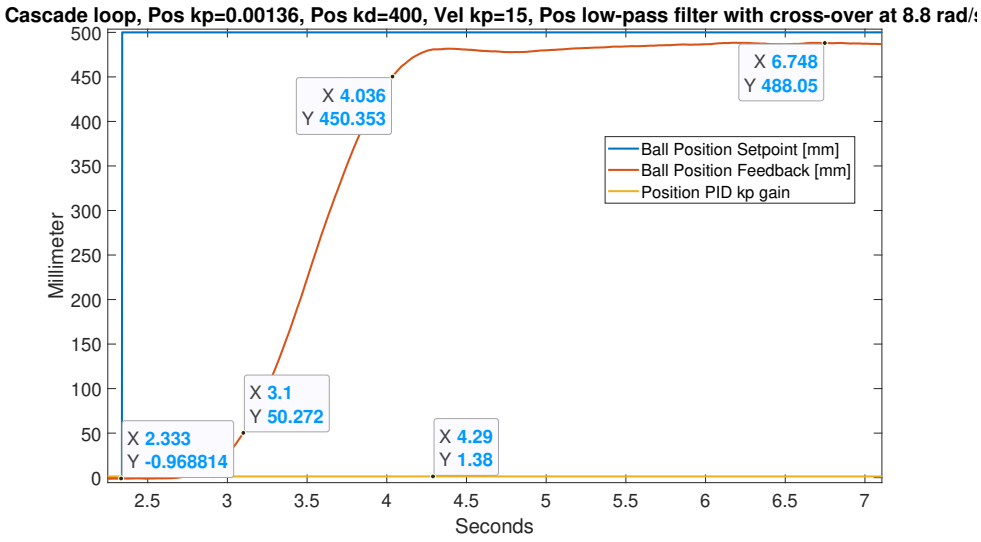

Figure 5.28: Cascade loop 500 [mm] step, PosPID kp=0.00136 kd=400, Vel kp=15, PosLowPass filter at 8.8 [rad/s]

Using a step input with an amplitude 500 [mm] shows the step response of the cascade system as shown in Fig. 5.28. The system has a undershoot of (500-482)mm=18mm which is $SS_{error} = 100 * \frac{18[mm]}{500[mm]} = 3.6\%$. The rise time is $T_r = (4.036 - 3.100)s = 0.936s$. The settling time is not reached and is at its lowest at 6.75s where it is 488 [mm] which is not within the $\pm 2\%$ of commanded value which in this case is $\pm 2\%$ of $500[mm] = \pm 10[mm]$. The value reached is $(500 - 488)[mm] = 12[mm]$. The PID values were: position controller kp=0.00136, kd=400ms, velocity controller kp=15.

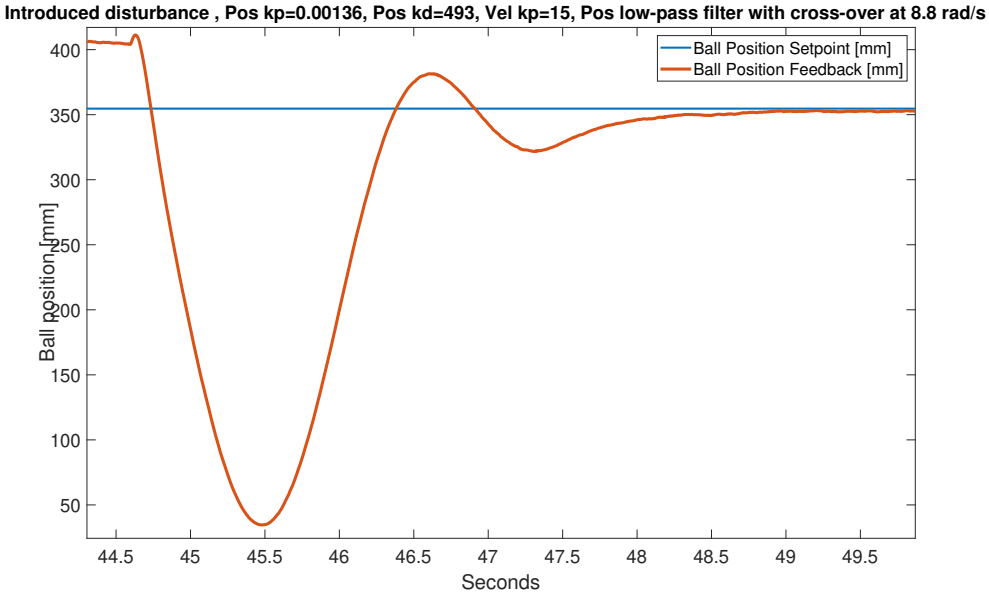

Figure 5.29: Introduced a disturbance to the Cascade loop

The system shows good disturbance rejection with a slight oscillation, approx $\pm 10\%$, before settling at the commanded position.

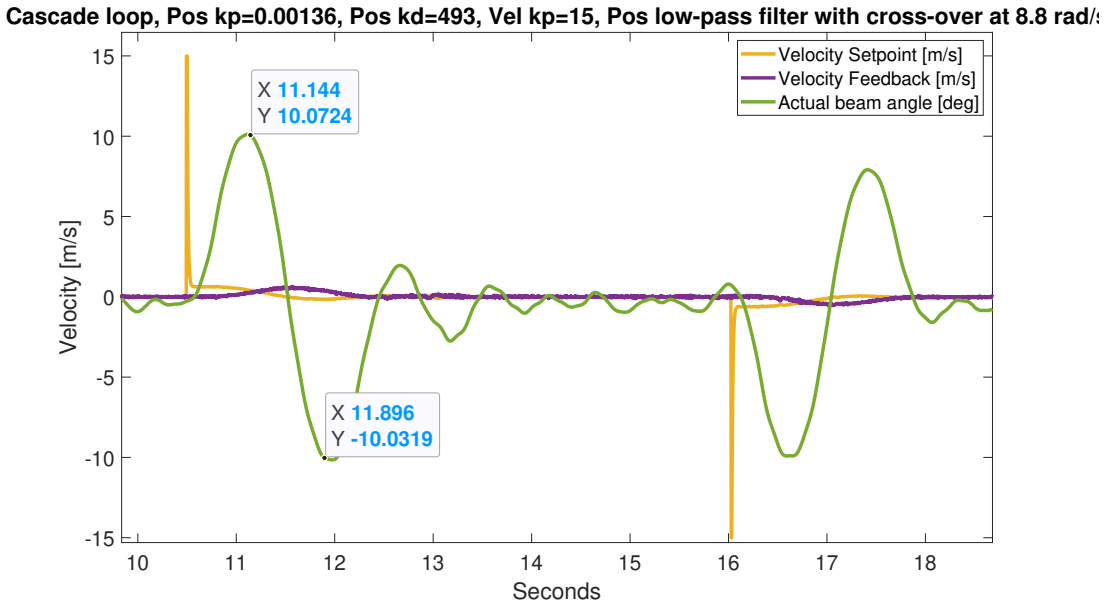

Figure 5.30: Cascade loop 500 [mm] step, PosPID kp=0.00136 kd=493, Vel kp=15, PosLowPass filter at 8.8 [rad/s], showing beam angle and velocities

With a step input of 500mm the angular velocity of the beam, Fig. 5.30, shows that the system is at its fastest (11.144s-11.896s), $\alpha_{beam} = \frac{0.35[rad]}{0.752s} = 0.467[rad/s]$, which is 0.047 [rad/s] faster than what was found in the beam response test, Fig. 5.12.

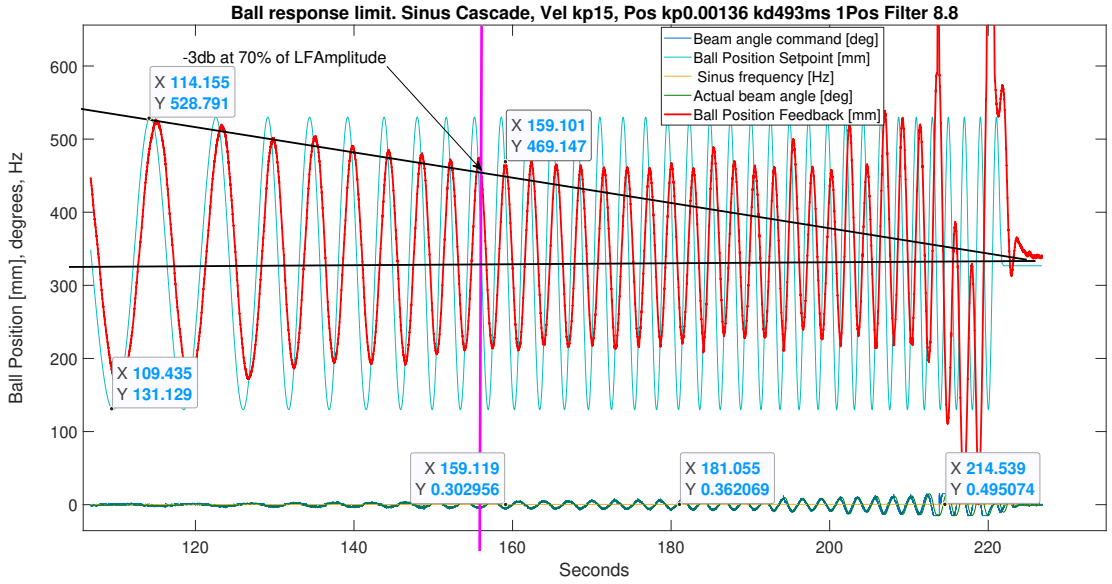

Figure 5.31: Sinus position input, amplitude=200 [mm], increasing frequency 0 to 0.5 [Hz]

Testing the ball positioning system response by using a sinus input with an amplitude of 200mm and increasing the frequency from 0 to 0.5 [Hz] show that the bandwidth is at 0.3 [Hz], $Sys_{bw} = 0.3Hz = 1.89[rad/s]$. This is the point where the position signal is at 70 [%] of the low frequency amplitude as shown in Fig. 5.31.

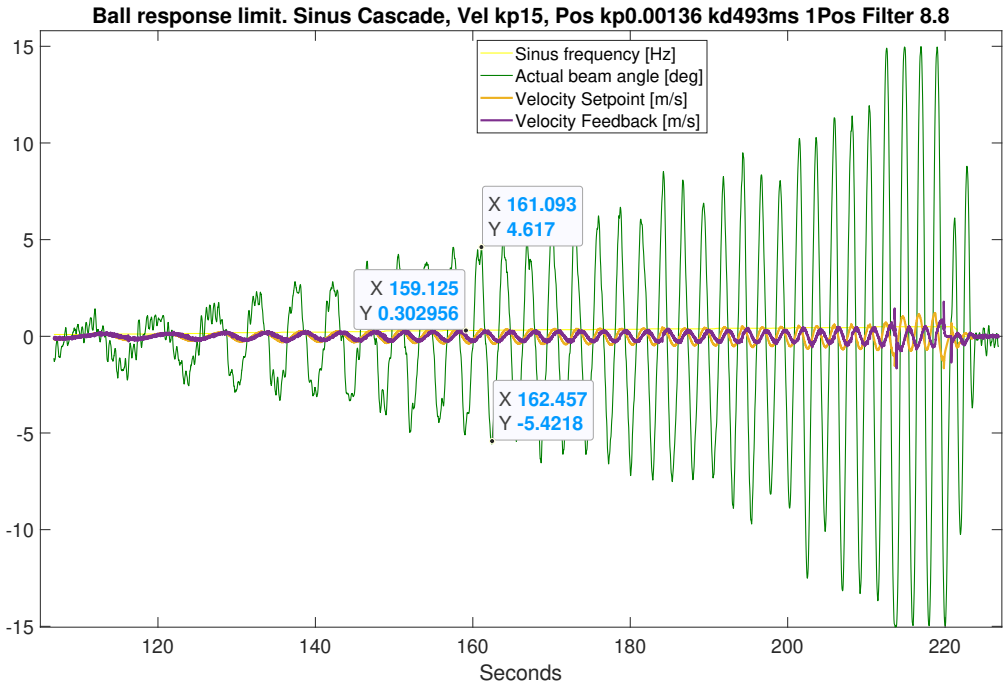

Figure 5.32: Sinus position input from data used in Fig. 5.31. Showing velocity command, feedback and beam angle

The plot Fig. 5.32 show that the beam angle velocity is within the beam system bandwidth where the ball position system is at $Sys_{bw} = 0.3[Hz] = 1.89[rad/s]$. The angular velocity was $\alpha_{beam} = \frac{0.37[rad]}{1.36s} = 0.27[rad/s]$.

5.3 Digital Twin and real measurements

The results from the experiments carried out with a digital twin provide a comparison of classical mechanics and compared against real measurements

5.3.1 Simscape, Real and Analyic

Data from Ctrlx Ball and Beam, Simscape model Ball and Beam, analytical and Tracker compiled to analyze:

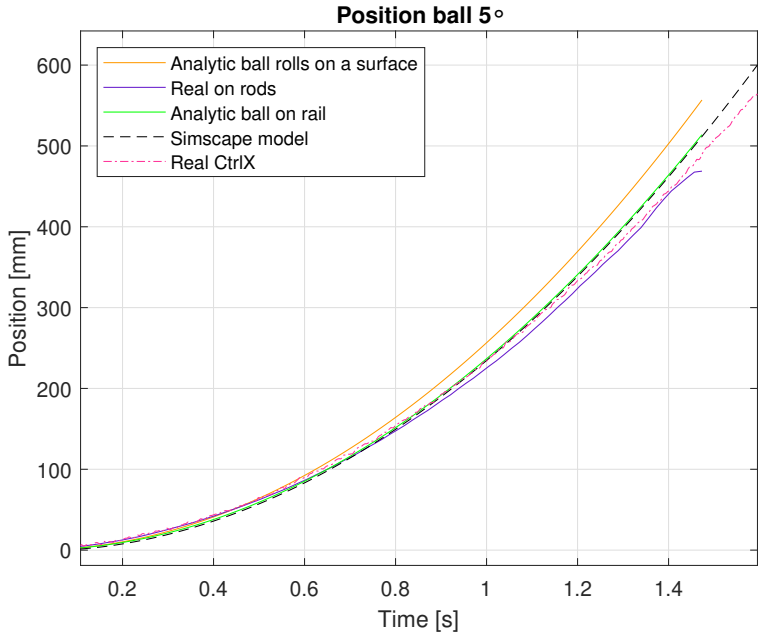

Figure 5.33: The position of the ball on the beam

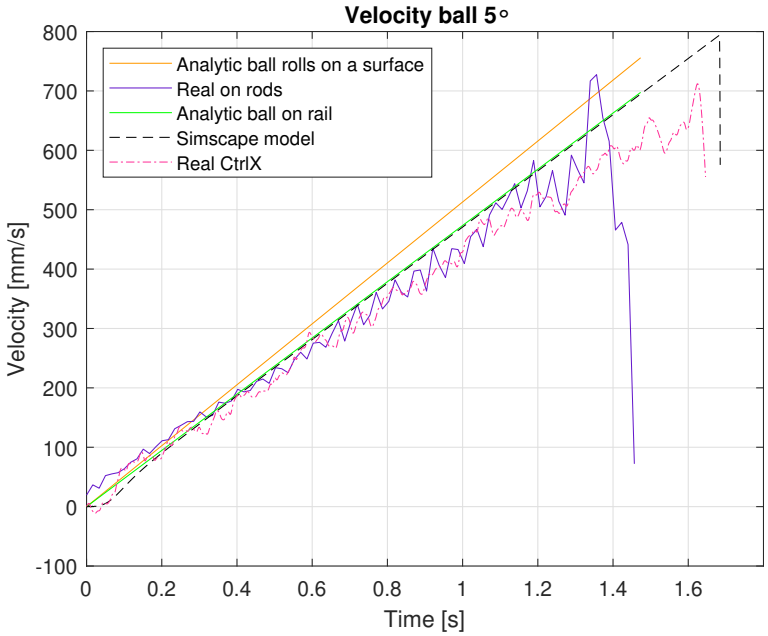

Figure 5.34: The ball's velocity on the beam

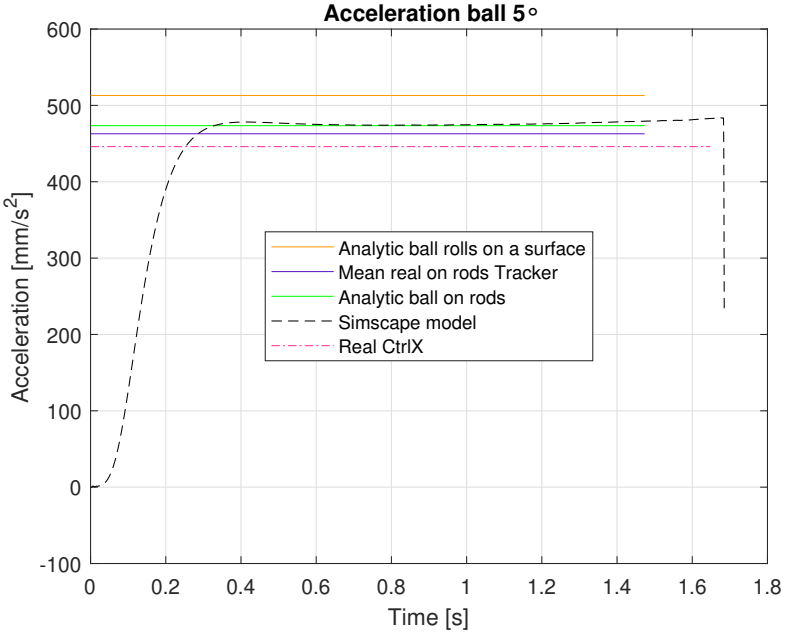

Figure 5.35: The ball's acceleration on the beam

To assess the connection between classical mechanics, physical measurements, and the Simscape model. Letting the ball roll on the beam with a slope of 5 degrees and from rest to the end of the beam, the ball's position, speed, and acceleration were linked. By looking at the Fig. 5.33. The yellow line demonstrates if the ball had rolled on a flat surface, for the rods see Fig. 5.33, Fig. 5.34 and Fig. 5.35 collord pink, green, purple and black.

Between simulation, analytical and real measurements the curves follow each other closely, but there is a slightly increased deviation. The discrepancy between the simulated model and the real measurement shows that the ball moves slightly further at the start than the simulated model, but after about 0.7 seconds they have reached the same distance and the real ball moved somewhat shorter at the same time. The blue curve Analytical ball rolls on a surface is a control measurement that shows that the balls roll a shorter distance in the same time if the balls roll on rods than if they roll on a surface see Fig. 5.33. As illustrated in Fig. 5.34, the same trend follows as for position in terms of speed versus time. The signal from real measurement CtrlX is somewhat noisy even if they have been filtered see Fig. 5.34.
see Matlab code F.2 for position, velocity, and acceleration and close-up of position and velocity see Fig. F.8 and Fig. F.9.

5.3.2 Beam Response Digital Twin

The beam's dynamic was tested by running the beam on a sine curve with increasing frequency and amplitude at ±10 degrees. It was tested with a direct drive which gives an inertia mismatch of 758.6320 eq. (2.51). It was also tested with a 1:10 ratio gearbox which gives an inertia mismatch of 7.5863 and an optimal gearbox according to eq 2.48 with a ratio of 1:27.5433 which gives an inertia mismatch of 1:1 , chapter F.3.

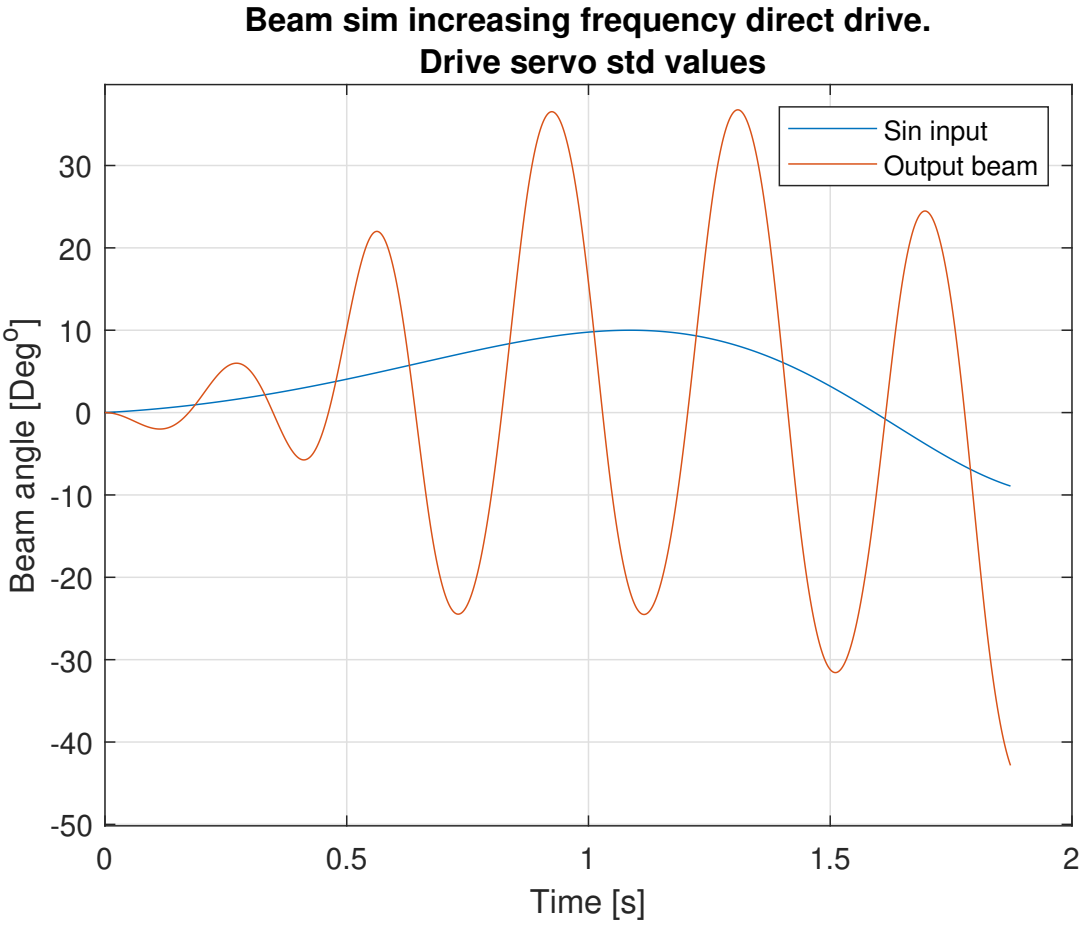

Figure 5.36: Beam sim increasing frequency. Gear-free direct mechanism. Drive std values

Analysis of Fig. 5.36 indicates that the beam does not track the input desired input position as it exhibits an overshoot of 25 degrees and an undershoot of 35 degrees. Furthermore, the beam oscillates around the desired position, indicating instability. These observations were made using standard drive values and direct drive mechanisms see Matlab editor 4.2.4.

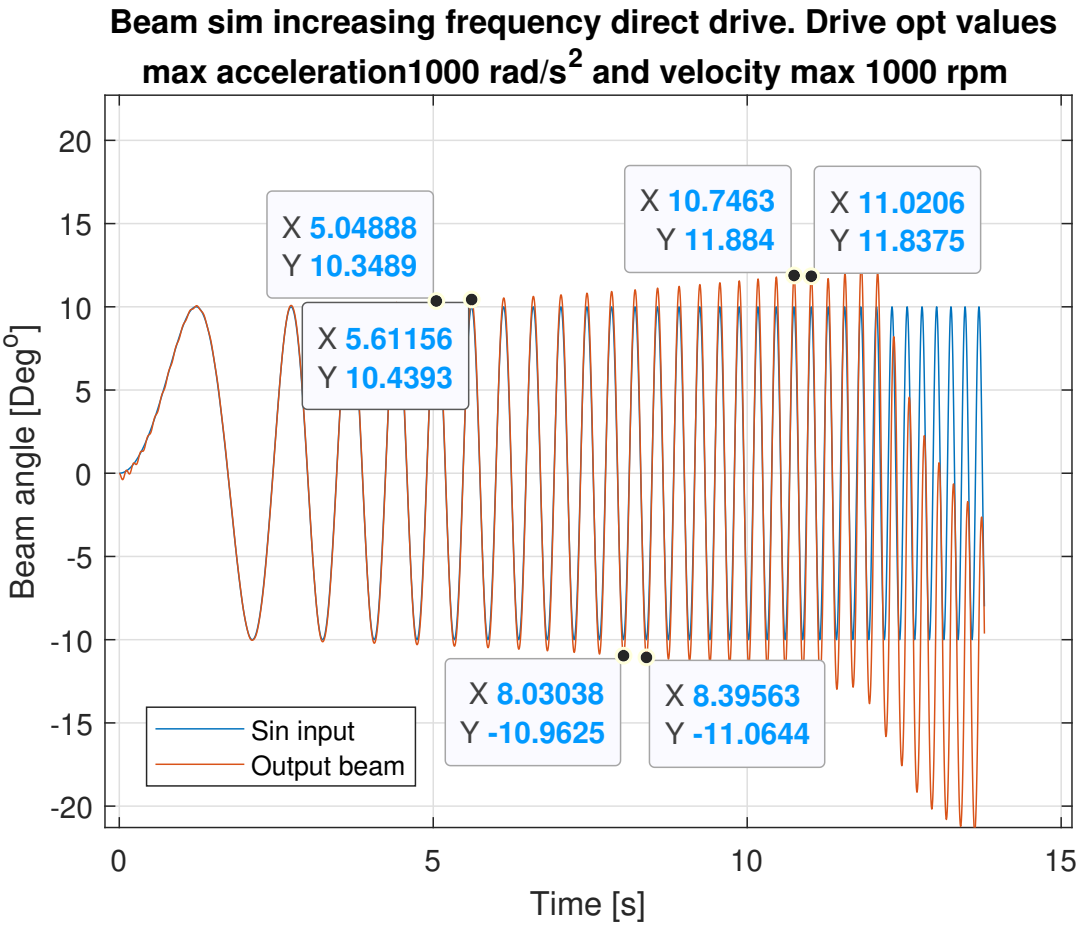

Figure 5.37: Beam response increasing sinus, max acceleration 1000 $[rad/s^2]$, max deceleration 1000 $[rad/s^2]$, max velocity 1000rpm

The beam was given a sinus signal with increasing frequency and constant amplitude at ± 10 degrees. Testing the beam without ball and axis acceleration and deceleration set at 1000 $[\frac{Rad}{s^2}]$. Maximum axis speed set at 1000 rpm. The beam started to overshoot the input signal at 1.78 $[Hz]$ and beagn more overshooting at 2.74 $[Hz]$ before it got unstable at 3.65 $[Hz]$ as seen in Fig. 5.37. The actual beam mount to the shaft had moved 45 [deg] after the test

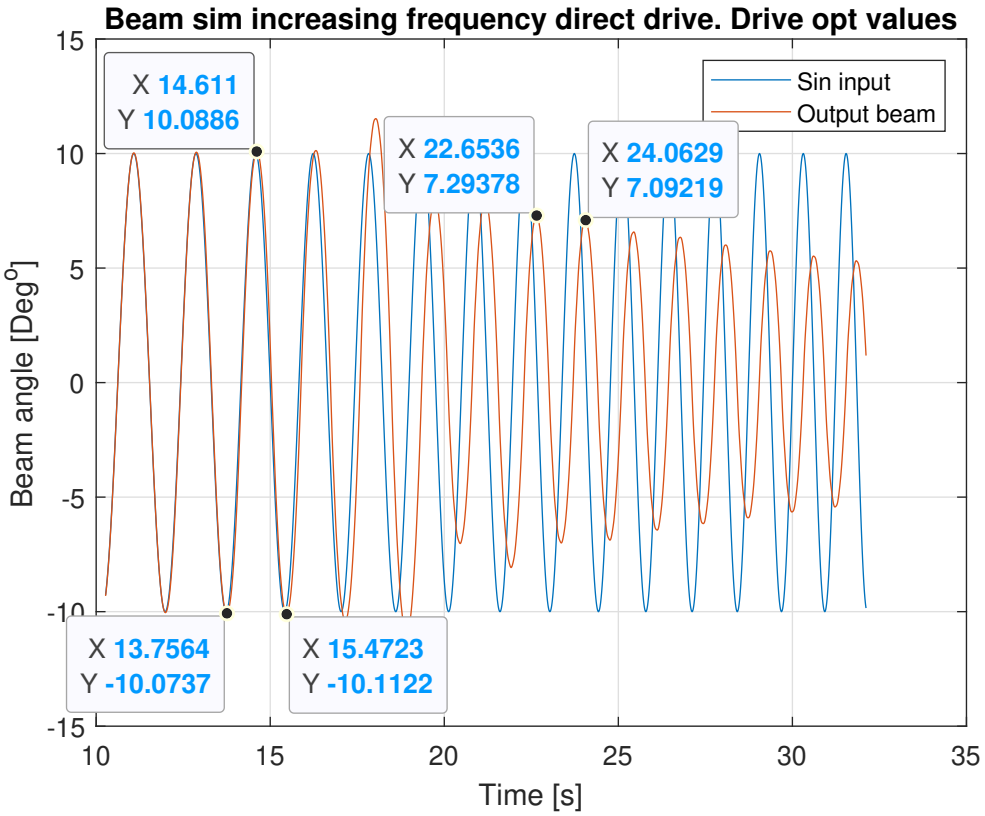

Figure 5.38: Beam sim increasing frequency direct drive. Drive opt values

After the values for the drive are changed to optimized see items 8 as shown see Matlab editor 4.2.4 the dynamic for the beam was changed and the beam started to go out of sync at 0.5838 $[Hz]$ see Fig. 5.38 and the signal dropped to 70 % at frequency 0.7096 $[Hz]$. This gives a system bandwidth of $Beam_{bw} = 0.7096\frac{1}{s} * 2\pi = 4.4584[rad/s]$. The angular velocity at 13.7564s-14.611s is $\alpha_{beam} = \frac{0.3519[rad]}{0.8546s} = 0.4118[rad/s]$.

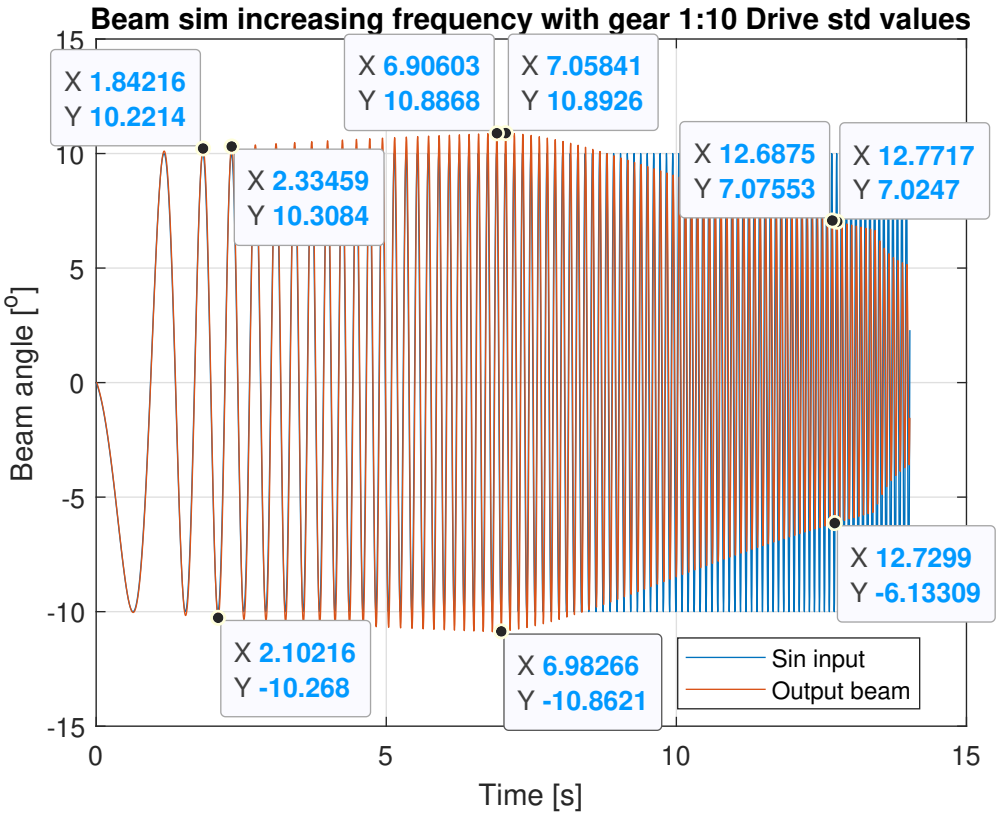

Figure 5.39: Beam sim increasing frequency with gear 1:10 Drive std values

After the values for the drive are set to standard as shown see Matlab editor 4.2.4 and it was tested with an optimal gearbox with 1:10, the dynamics of the beam was changed see Fig. 5.39 and the beam started to go out of sync at 2.0307 $[Hz]$ and has a max 0.89° overshoot at 6.5625 $[Hz]$ and the signal dropped to 70 [%] at frequency 11.88 $[Hz]$. This gives a system bandwidth of $Beam_{bw} = 11.88\frac{1}{s} * 2\pi = 74.62[rad/s]$. The angular velocity at 2.10216s-2.33459s is $\alpha_{beam} = \frac{0.35913[rad]}{0.23243.s} = 1.54509[rad/s]$.

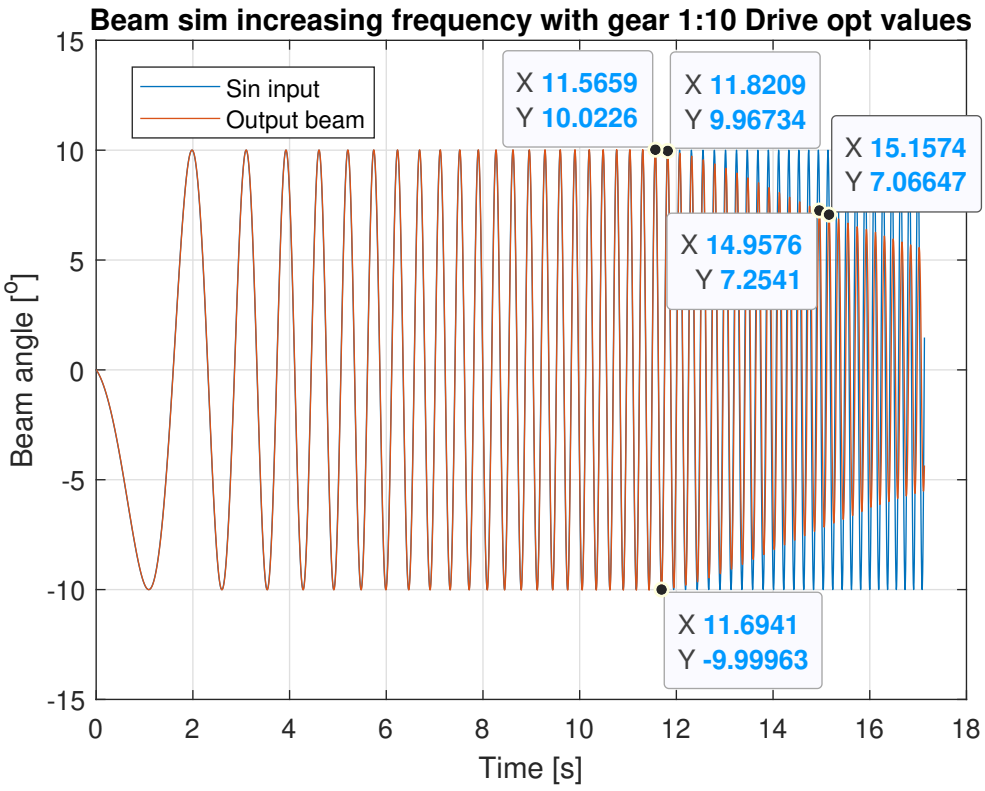

Figure 5.40: Beam sim increasing frequency with gear 1:10 Drive opt values

After the values for the drive are changed to optimized see items 8 as shownas shown see Matlab editor 4.2.4 and it was tested with an gearbox with 1:10, the dynamics of the beam was changed and the beam started to go out of sync at 3.9216 $[Hz]$ and the signal dropped to 70 [%] at frequency 5.0050 $[Hz]$. This gives a system bandwidth of $Beam_{bw} = 5.0050\frac{1}{s} * 2\pi = 31.4474[rad/s]$. The angular velocity at 11.6941s-11.8209s is $\alpha_{beam} = \frac{0.3485rad}{0.1268s} = 2.7484[rad/s]$.

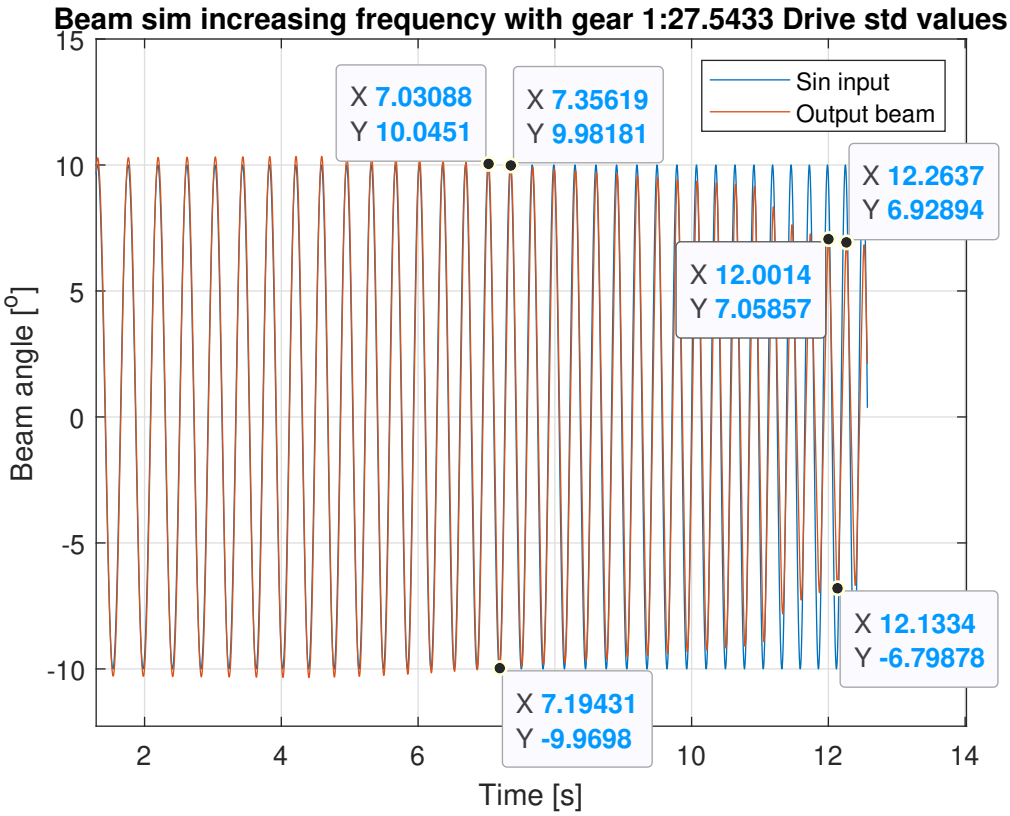

Figure 5.41: Beam sim increasing frequency with gear 1:27.5433 Drive std values

The values for the drive are set to standard values and it was tested with an optimal gearbox with 1:27.5433 see matlab chapter F.3 the dynamics of the beam were changed and the beam started to go out of sync at 3.0740 $[Hz]$ and the signal dropped to 70 [%] at frequency 3.8124$[Hz]$. This gives a system bandwidth of $Beam_{bw} = 5.3172\frac{1}{s} * 2\pi = 23.9541[rad/s]$. The angular velocity at 7.1943s-7.3561s is $\alpha_{beam} = \frac{0.3482rad}{0.1736s} = 2.1511[rad/s]$.

5.4 Ball Dynamic and Safety

The biomechanical limits based on real system, the maximum tangential velocity is set to $1.2099[\frac{m}{s}]$ which gives an angular angular velocity and $2.8697[\frac{Rad}{s}]Tbl.$C.1.2. The maximum clamping force is set to $27,404[Nm]$ C.1.2 which is higher than the servo can output $(3.75[Nm])$ [1].
In order for the ball to take off from the beam, the angular acceleration must also not exceed $27.5094[\frac{rad}{s^2}]$, chapter F.5, starting from the beam at 20 degrees with the ball on the tip in the lifted position.

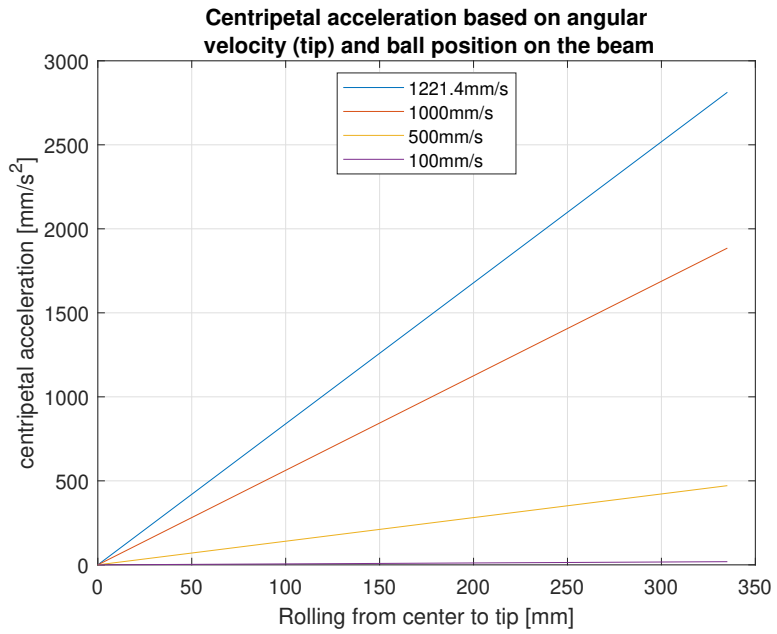

Figure 5.42: Centripetal acceleration based on angular velocity and position on the beam

Looking at Fig. 5.42 shows the theoretical centripetal acceleration see eq. 2.14 and Matlab script, chapter F.6, if the beam rotates at a tangential velocity shown in the Fig. 5.42 . If the ball is on the beam, the ball will experience an acceleration and it is different as the ball moves out onto the beam shown in the figure.

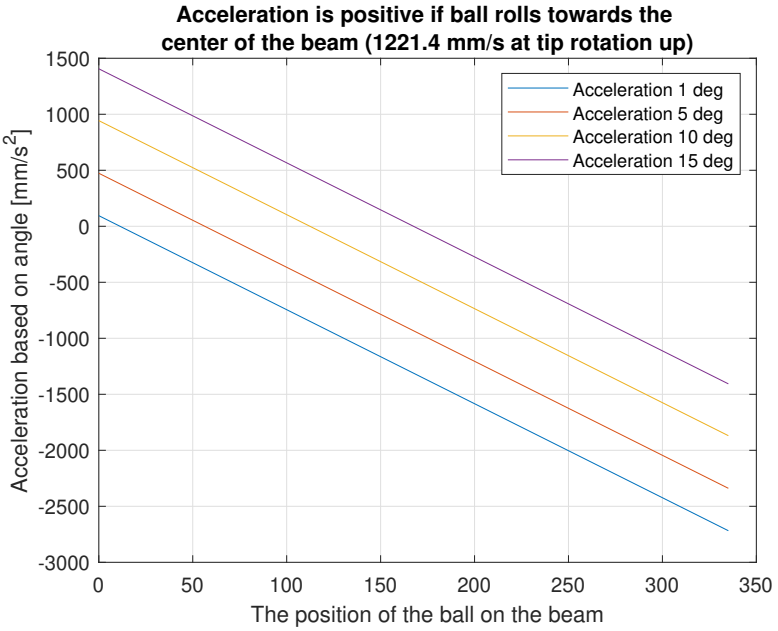

Figure 5.43: Acceleration is positive if ball rolls towards the center of the beam (1221.4 [mm/s] at tip rotation up).

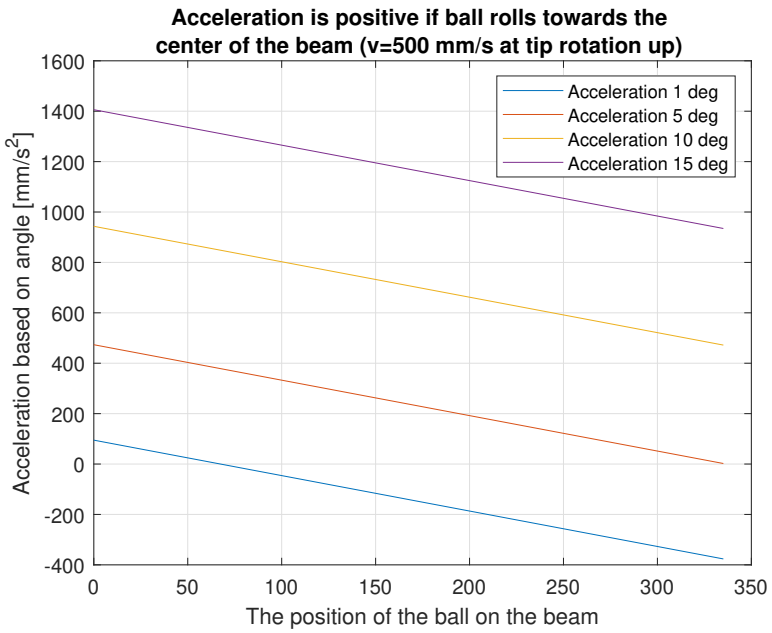

Figure 5.44: Acceleration is positive if ball rolls towards the center of the beam (v=500 mm/s at tip rotation up)

If the ball is positioned on the beam and rotates up with the tangential velocity shown in Fig. 5.43 and Fig. 5.44 the ball will be subjected to acceleration due to gravity and centripetal acceleration. By looking at the figure, it is shown which acceleration (centripetal and gravitational force) is changing according to where the ball lies on the beam and tangential speed see eq. 2.46.

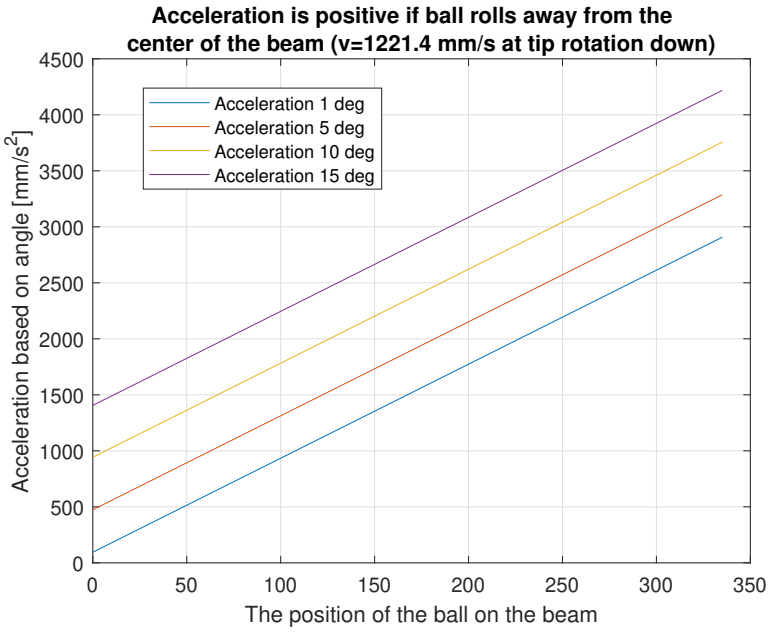

Figure 5.45: Acceleration is positive if ball rolls away from the center of the beam (v=1221.4 mm/s at tip rotation down)

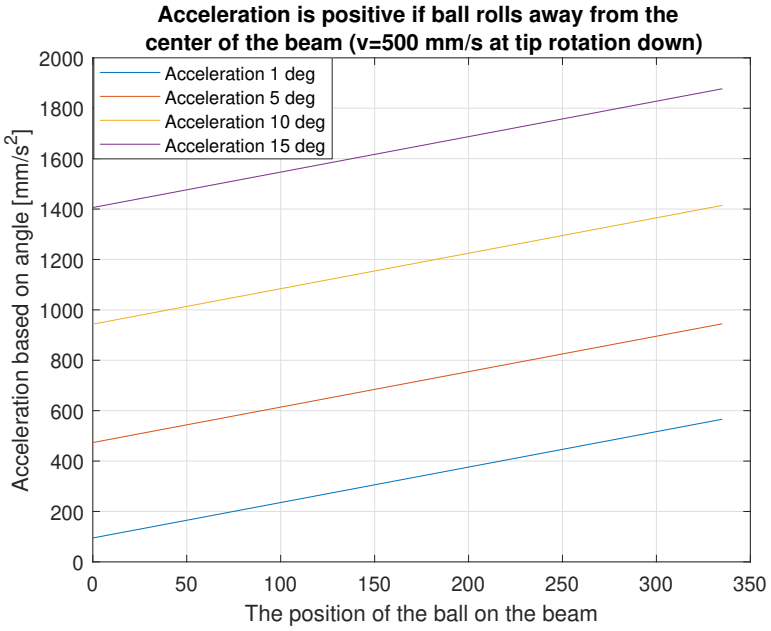

Figure 5.46: Acceleration is positive if ball rolls away from the center of the beam (v=500 mm/s at tip rotation down)

If the ball is positioned on the beam and rotates Down with the tangential velocity shown in Fig. 5.45 and Fig. 5.46 the ball will be subjected to acceleration due to gravity and centripetal acceleration. In the figure, it is shown which acceleration (centripetal and gravitational force) is changing according to where the ball lies on the beam and tangential speed see eq. 2.46.

Chapter 6

Discussions

6.1 Real vs expected performance

6.1.1 Simscape

Using Simscape and creating a digital twin can make testing new features, controllers and designs faster. This model has proved comparable to the physical system, which means that it could be further improved thru testing with a gearbox to see the response of the system. In the Simscape model, air resistance and the ball's surface are not taken into account, but they are considered ideal surfaces, which the physical system is not. If the digital twin had been introduced early, optimizations could have been implemented more quickly such as the gearbox. Testing other types of servos could also have been tested to find an optimal specification for the physical system.

6.1.2 Real performance

The ball's movement and positioning on the beam has been optimized through testing. The system is slowed down to move relatively slowly as it becomes rapidly unstable with faster movements of the beam. This comes from a high inertia mismatch which causes the beam to overshoot the desired position. Therefore, by using a gearbox, the beam could probably have faster rotations with acceleration and velocity. A measuring sensor that had been optimized for the length of the beam could possibly also have been good as the Balluff sensors can measure from 100 [mm] to 5000 [mm] Tbl. C.1 with higher accuracy and precision would also have been preferable. Since the system is unstable, noise will create more instability when the beam regulates noise on the signal. If the signal had been more stable, the beam regulation would probably have been more stable.

6.1.3 Matlab Comparison of Data

By comparing the ball that rolls with an angle on the beam of 5 degrees see Fig. 5.33, Fig. 5.34 and Fig. 5.35 it is clear that the ball rolls in the Simscape model almost exactly the same as the analytical one. The ball also rolls very similarly to physical measurements carried out with the Tracker and with the use of measurement data from the Balluff sensor. What emerges from this is that there is a connection, but it is not entirely accurate. A control plot (yellow color) has also been entered for if the ball had rolled on a flat ice for 2 rails it shows higher acceleration, which is logical as a ball will roll slower on 2 rails. Deviations that create a difference in the data are air resistance against the ball, and friction on the surface of the ball as it is somewhat rough. In addition, the ball has a seam inside that is somewhat thicker at the seam. This seam can create a moment of inertia that is not uniform and calculated acceleration is then difficult to calculate as this seam can move based on how the ball is placed on the beam. Another deviation is that the beam is not exactly 5 [deg] due to calibration errors. Other deviations can lie with the Balluff sensor as it has a resolution $\leq 5[mm]$ and an accuracy of $\pm 0.6[\%]$ see Tbl. C.1. Filtering the signal can also affect the signal and the noise that affects it. Tracker may also have the same calibration error as this program gives a "rough" estimation and is not accurate.

Matlab vs real system
The Matlab model and Simulink simulations are with an ideal system and without filtering the feedback signal. The real system has filtering and real dynamics like friction and air resistance. Comparing the two is not one-to-one, but gives an indication of behavior. The simscape model is a closer match to the actual system and the measurements indicate this. The velocity loop was tuned based on the findings in the Matlab/ Simulink simulation where the use of only a kp gain was sufficient. The gain was set to 15 to counteract the decrease in gain margin from the filter in the real system.

Tuning the outer position loop in the cascade controller, with the Ziegler-Nichols method, shows similarities in behavior, but the values in the simulation are different from the real system, simulation Fig. 5.8, Fig. 5.9 and real Fig. 5.25, Fig. 5.26. This confirms that the system behaves in a similar way, but the gains are not the same. The final tuned controllers show a similarity in behavior as seen in simulation Fig. 5.10 and real Fig. 5.27.

6.2 Noise and Filter

Filtering is an important part of regulation. A precise signal can provide better regulation, but it can create a phase lag or slow down response which in turn creates inertia. Signals often have to be filtered to remove noise or improve the resolution of the system. In this case, a Balluff sensor has been used which has a resolution of less than 5[mm] and an accuracy of $\pm 0.6[\%]FS$ and a repetitive accuracy of $0.024[\%]FS$ and a hysteresis of H max (% of Sr) 0.3[%]. By looking at Fig. 5.19, an amplitude of 1.6 [mm] is shown when a sheet is placed stationary in front of the sensor. In this case, the system has higher accuracy and resolution when using low-pass filters, which results in a phase lag that is negative for measurement in Fig. 5.20 the phase lag is 47 [ms]. There are other ways of filtering that may have a better ideal effect on this system such as Chebyshev Filter or Butterworth filter. The Butterworth filter can have a higher order for a faster cut of frequency and Chebyshev Filter can give a faster response but with a higher pass band ripple as shown in [56].

Designing a cascade loop has benefits in control stability but can also introduce challenges when there is noise in the feedback in the form of high and low frequencies. Derivation of the noisy feedback signal results in amplification of the noise as seen in Fig. 5.21. Adding filters reduces the noise, but also introduces phase lag and an increase in the gain margin that slows down the system. Fig. 5.22 and Fig. 5.20 show how adding filters slows down the signal. There is a balance between filtering and response. Choosing one filter in the ball position feedback, was chosen as the best solution in this system. Fig. 5.19 shows a low-frequency period of ≈0.45s. Filtering out the low-frequency oscillations was not possible and this resulted in a beam that has a small motion even when the ball is at 0% position error. The choice of cut-off frequency at 2 times the 4.4 $[rad/s]$ bandwidth found in the beam dynamic test in Fig. 5.12 was used as a base value to limit filtering out system dynamics. Other frequencies might yield better results.

6.3 Control Performance

The control performance is limited by the ball position feedback and the noise in the system. Slowing the system down by limiting the axis acceleration to 2 $[rad/s^2]$ results in less vibration, Fig. 5.13, but also a slower beam motion that limits the positioning time of the ball. The beam can move at 2.49 Hz with 1000 $[rad/s^2]$(no acceleration limit in the drive) as opposed to when limited to 2 $[rad/s^2]$ when the frequency dropped to 0.7 Hz. Increasing the acceleration as seen in Fig. 5.13 to Fig. 5.18 does not result in a faster settling time, but higher motor forces. The 2 $[rad/s^2]$ acceleration was the fastest being 0.1s faster than the second fastest, the 10 $[rad/s^2]$ acceleration. The current control parameters and system response as seen in Fig. 5.28 show good behavior with a fast rise time of 0.936s in its 500 [mm] transition. Tuning the control to have no overshoot proved the best for stable control. Positioning performance at a 100 [mm] step input was: rise time $T_r = 1.334s$, settling time $T_s = 2.694s$ and no overshoot or undershoot, Fig. 5.27.

Positioning performance at a 500 [mm] step input was: rise time $T_r = 0.936s$, settling time is not reached but the steady state error $e_{ss} = 3.6[\%]$ in 3.6s and no overshoot Fig. 5.28. The high mismatch of the mass moment of inertia between the load and the motor is also a limiting factor as proven in Simscape and discussed in the Gearbox section 6.5. Further improvements can be done if better position feedback or filtering is implemented.

6.4 System Identification

The system identification was done from a step input, Fig. 5.1, which is a simple but not an accurate representation of the system. The following calculations and plots are therefore not as precise as a more complex system identification can produce. Using Matlab with its system identification toolbox could improve the model and the simulations based on a more accurate model. Taking time constraints and the complexity of the project into account, a simple model was used to illustrate the steps in using a model for the simulation of the real system.

6.5 Gearbox

The beam has a moment of inertia that is approximately 759 times greater than the moment of inertia of the servo motor. Generally, the rule of thumb is this ratio is 1 and not higher than 10 and Bosch Rexroth generally recommends handling ≤ 6 and processing ≤ 1.5 [10]. According to Kollmorgen, [57] a direct drive system can be used to stiffen the system for higher inertia mismatch applications "Precisely controlling the system with good bandwidth can be achieved even with inertia mismatches exceeding 1000:1". According to an article from The University of Wollongong [15] demonstrates that a stiffer system will decrease oscillations in a high inertia mismatch system.

In this case, there is a torque mismatch which means that the servo does not drive the beam but that the beam can drive the servo which creates instability test of physics Fig. 5.11 and test of Simscape model Fig. 5.37. This creates limitations in the system and the driver is "dulled" down so that it cannot move because it can quickly overshoot. By simulating in Simscape, it is shown that the dynamics between the simulated model and the physical one are comparable Fig. 5.36 and 5.12. To prove that a gearbox will create better stability and faster regulation, an ideal gearbox was tested which simulated the ratio 1:10 and 1:27.5433 in gear ratio. It is clear from looking at the Fig. 5.39, Fig. 5.40 and Fig. 5.41 that a gearbox can create a regulation that is better than direct drive. Even if a gearbox would create a better regulation result, it would still probably have needed to be optimized through a velocity control loop in the drive Fig. 5.39. Although a gearbox is simulated, a physical gearbox will add a moment of inertia, viscosity, and backlash. This is not included in the simulated model and can create challenges in a physical system.

6.6 Regulation Technique

Designing the system as a cascade loop with an inner velocity loop and outer position loop has the advantage of adding a derivative to the system. This helps in stabilizing the marginally stable system and makes it possible to only have kp gains and still have stable control, Fig. 5.24. The derivative is added by converting the position signal to velocity by derivation. Simulation in Matlab and Simulink showed that the velocity loop could have a lead compensator to increase the response as seen in Fig. 5.5 and in Fig. 5.4 where the gain margin is reduced from 14.6 $[rad/s]$ to 9.72 $[rad/s]$. Compared to an uncompensated system, Fig. 5.6, the compensated had a 0.1s faster rise time and 0.8 less position error, but it had an 11.68[%] overshoot. The uncompensated system had a settling time that was 0.04s less and no overshoot. Based on these data a pure kp gain was chosen as the best controller for the velocity loop so as to not introduce instability in the inner loop.

Tuning the PID controllers with the Ziegler-Nichols method yielded high settings. The resulting values resulted in large overshoots, 80[%] in the real system and 39[%] in the model. A manual

approach was used to lower and optimize the control to a point of a slight undershoot for the best ball behavior. Reaching the steady state error of <20 [%] was achieved, but repeatability suffered from the ball not being uniform. The ball has a seam inside that makes the ball roll in a non-uniform way which makes small corrections hard. The controller does not have an integrator that corrects for deviation. Further improvement could be to include such a feature.

Regulation of the ball and beam is done in a classic way with cascade regulation with PID regulation. Other methods that could have been used are the Kalman filter and state space to predict and create a faster system that predicts and suppresses noise. This would probably make the response of the system faster and create greater stability.

6.7 Calibration of the Beam

The control system does not have an integrator to compensate for steady-state errors. Placing the beam in a horizontal position aids in improving the accuracy of the system when it is regulating around the axis zero range. The calibration is easily done through the HMI where the level on the beam acts as the reference. The accuracy improves after calibration.

6.8 HMI

The HMI was designed to let the operator have full control over the PID settings, ball positioning, filter values, calibration, axis settings, sinus movement, and step movement. Monitoring the system through visualization of the model, trends, and dials gives real-time feedback. The philosophy behind the layout is to give the operator the ability to try out different settings and get instant feedback on the behavior. This ability promotes learning and contributes to a better understanding of regulation technology. The HMI is accessible through ethernet cable and through WiFi which makes it flexible. The interface is optimized for laptops, monitors, and 16:9 tablets. Other formats have to be set up in the WebIQ designer to fit the individual screens. The operation of the HMI proved intuitive, responsive, and with low lag. The ctrlX Core has to have a valid time and date which can be synchronized with the internet or browsers. This setting has not been activated so manual setting in the ctrlX core has to be performed for the trend to work in the HMI. Future programming could automate this. The visualization of the model in the 3d viewer shows that the beam and ball moved in a synchronized motion with the real system.

6.9 ctrlX

6.9.1 ctrlX Core

ctrlX as a platform performed well. The interaction between Bosch Rexroth- and 3rd party- applications was manageable with the help of online tutorials and documentation. The ctrlX forum was a useful tool for solving issues along the way. There was one problem with trying to install the oscilloscope app where the ctrlX core locked up due to an incompatibility issue between the two. A full reset and update to version 18.1 from 12.1 had to be done with assistance from Bosch Rexroth Germany which responded immediately. The ctrlX Core topology, where the data layer is accessible across all apps, is a feature that served usefully and made programming more efficient. Some of the apps need a license that is available in the ctrlX store. This project got all licenses from Bosch Rexroth for free.

6.9.2 ctrlX PLC

The integration of the PLC app in the ctrlX core topology worked well. The PLC had preinstalled libraries for the axis interface, IIR filter, and PID loops which made the programming efficient. Starting with a predefined template for drive control takes care of the required base programming. Adding and deleting what is necessary to customize the program saves time. Having the possibility to create a custom virtual HMI proved to be a valuable tool for testing and verification during programming. An already prepared HMI is available. It was used for testing in the starting phase of programming to verify axis movement. This HMI has limits, set for the axis acceleration, that have to be modified if other values are desired. The template has a multitude of ways to control the axis and there might be better solutions to how the axis was programmed. The beam can have a rapid movement when the axis is enabled which is not a desired motion. Features like soft start, stop, pause and resume could be upgrades for new iterations. Transitions between modes can make the ball overshoot and hit the ends of the beam and bounce back. This makes it hard for the controller to compensate. When those situations came up, a manual interaction was made and the ball was put on the middle of the beam to calm it down.

6.10 Safety

The beam's rotation speed and acceleration are measured to max $0.42[\frac{rad}{s}]$ and the maximum angular acceleration is set to max $2[\frac{rad}{s^2}]$ see Fig. 5.12. Compared to calculations on centripetal acceleration and safety considerations in Tbl. C.1.2, this beam does not reach these maximum velocities. What is beneficial is that the ball cannot fall off, chapter 5.4, and the ball on the beam will not move unnecessarily in the opposite direction of the desired based on position due to centripetal acceleration see Fig. 5.44. The servo can give a maximum of 3.75 [Nm], and the maximum clamping force based on biomechanical limits is $27,404[Nm]$, chapter C.1.2, which is far above what the servo can deliver in terms of power and speed.

The system has limits that stop the axis if met. The HMI is monitoring the motor force and will stop if 250[%] of nominal torque is reached. The ctrlX core is set up with an acceleration limit at 10 $[rad/s^2]$ that will stop the axis if exceeded. Pushing the emergency stop in the HMI will stop the axis. It has to be acknowledged for the system to run again. The system can be stopped by pushing the black button on the front of the unit. This will disconnect the power to the drive.

Chapter 7

Conclusions

7.1 Were the goals met?

Problem statement 1:
Calculate differential equations of the mechanics
Calculations made with differential equations have been made for the ball movement. It is calculated as an ideal that the angle comes instantly. It has been calculated how the rotation affects the ball on the beam and as increasing tangential velocity. According to testing the dynamics is quite similar.

Problem statement 2:
Design and build of the mechanic's "Ball and Beam" which is attached to the electric drive and motor
A beam was built with the lowest possible inertia in mind, but the beam also had to have a suitable span. In this case, since the length of the beam affects the inertia so much, rods made of carbon were used. All parts are made of 3D printed parts. However, the inertia mismatch is still too high.

Problem statement 3:
Selection of the sensor for measuring the ball position
The sensor chosen for the "Ball and Beam" by Rexroth was based on what was available. It has a low resolution and has some noise which affects the precision.

Problem statement 4:
Simulation of the behavior using either Python control library or Matlab
The Matlab and Simulink simulations proved valuable in the design of the controller. By testing different versions and settings a cascade loop was ultimately applied to the real system where small tuning was necessary before the system was operable. The benefit of having verified the controller topology before use in an actual system saves time and improves safety during the commissioning phase. Challenges like noisy feedback are not considered in the model and differ greatly from the real system.

Problem statement 5:
Design of a controller to make the system stable The design of the controller was based on stability and robustness. Choosing a cascade controller with an inner velocity loop added a derivative to the system and stabilized it. Tuning of the controller was simplified by just having a kp gain in the velocity loop and a PD-controller in the outer position loop. The position D-part was added to increase the bandwidth and limit the overshoot, but was not needed for stability. All system target requirements were met except from the positioning repeatability due to a ball with a non-uniform mass.
The controller of the system does not give any over or undershoot and has acceptable steady state error in relation to the system specifications.
Regarding settling time the controller can reach the desired position at short distances but struggles to reach the target at the extreme points of the beam.

Problem statement 6:
Programming of the ctrlX PLC for interface with the WebIQ HMI Programming the PLC was efficient due to preinstalled libraries and templates. The ctrlX Core was automatically integrated with the PLC, sending data to the datalayer and OPC UA server without any issues. Accessing the data in the WebIQ designer was done by mapping the datalayer values to the HMI. The integration of the 3d viewer proved difficult. The beam in the model was working in the HMI, but the ball position has not been mapped to an axis and does not move. Later iterations should implement this function.

Problem statement 7:
Design a digital twin and compare it to the real system
The digital twin is used with the help of Simscape and Matlab. This has been compared against the physical system and the dynamics are very similar. This model has also been tested with a gearbox and shows that regulation can be faster when using a gearbox and the inertia mismatch affects the system less negatively. A gearbox with 1:10 would be possible to use, but ideally, it should be about 1:27.5. A gearbox would also be beneficial for the power draw as inertia mismatch can lead to higher wear and heat on the servo, and therefore its lifespan can be shortened.

Problem statement 8: Design an intuitive HMI that displays the real-time state of the system and lets the operator interact for educational purposes The HMI was made to be intuitive and display key information that the operator can interact with. Providing access to a multitude of settings of the control system and giving immediate feedback promotes learning and understanding. The HMI incorporates safety as emergency stop, torque limit, and temperature limit. The operator can choose to use a sinus, step, or slider input for ball positioning. Monitoring of the motion is in the trend or through the dials. The 3d visualization did move in the HMI except for the ball motion which can be improved for the next iterations.

Caution statement:
Any dangerous axis movement must be prevented
The beam is not to be able to do anything dangerous, limits have been set so that the beam cannot harm people. Due to the regulations regarding inertia mismatch, the beam has no danger of accelerating or reaching a dangerous velocity for people. The ball has a low mass and poses no danger to people. Crush damage is not realistic as the servo does not have enough torque.

7.2 Further work

Further work should be to mount a gearbox on the model and look at the possibilities in terms of filtering sensor noise and possibly implementing a Kalman filter with state space regulation. The 3d visualization should be finalized to show the ball motion in the HMI.

Appendix A

Definition of variables

F =force [N]
$m_b = m_{ball}$ =Mass of ball [kg]
$J_b = J_{ball}$ =Ball mass moment of inertia $[kg \cdot m^2]$
r_b =Ball radius [m]
$Beam_w$ = Beam width where ball contact rail [m]
$r_{rotatingonrail}$ = Ball rotating about contact surface of rail[m]
β = angle [°]
$a_{ball} = a_b$ = Acceleration ball $[\frac{m}{s^2}]$
$\omega_{ball} = \omega_b$ = Rotation velocity $[\frac{Rad}{s}]$
g = Gravitation $[\frac{m}{s^2}]$
$\ddot{x}$ = Acceleration $[\frac{m}{s^2}]$
m_g = Mass $[kg]$
v_b = Velocity ball $[\frac{m}{s}]$
a =Acceleration $[\frac{m}{s^2}]$
t = Time [s]
x_0 Initial start position [m]
v_0 Initial start velocity $[\frac{m}{s}]$
ω =Rotation velocity $[\frac{Rad}{s}]$
θ = Angle [°]
θ_0 = Initial start angle [°]
I =Mass moment of inertia $[kg \cdot m^2]$
α = Angular acceleration $[\frac{Rad}{s^2}]$
τ = Torque $[Nm]$
a_N = Normal acceleration$[\frac{m}{s^2}]$
a_T = Tangential acceleration$[\frac{m}{s^2}]$
α_b = Angular acceleration $[\frac{Rad}{s^2}]$
$|p_b|$ = Position ball [m]
τ_b = Torque ball $[Nm]$
r_r = Radius [m]
$|v_b|$ = Velocity ball $[\frac{m}{s}]$
J_{LM} = Load inertia motor shaft $[kg \cdot m^2]$
n = Gear ratio[-]
J_M = Inertia of the rotor $[kg \cdot m^2]$
J_L = Inertia of the load $[kg \cdot m^2]$
n_{opt} = Optimal gear ratio [-]
α_L = Angular acceleration Load $[\frac{Rad}{s^2}]$
α_M = Angular acceleration motor $[\frac{Rad}{s^2}]$
T_M = Torque motor [Nm]
T_L = Torque load [Nm]
$Relation_J$ = Inertia relation [-]
$s^2X(s)$ = Frequency(s) domain position

β_{cmd} = Angle command [°]
β_{act} = Angle act [°]
P_s = The servo/drive plant
P_{bb} = The ball and beam plant
K_P = Proportional gain
$K_D s$ = Gain derivative
$Z_{PD} = \frac{K_P}{K_D}$ = PD-controller adds a zero
$x_{cmd}(t)$ = Command position, time domain
$x_{act}(t)$ = Acting position, time domain
$X_{cmd}(s)$ = Command position, Frequency(s) domain
$X_{act}(s)$ = Acting position, Frequency(s) domain
s = Frequency(s) domain
ω_n = The natural frequency
ζ = Damping ratio
$\%OS$ = Percent overshoot
T_p = Peak time
T_r = Rise time
T_s = Settling time
K_u = Ultimate gain
P_u = Period
K_p = Proportional gain
K_i = Gain integral
K_d = Gain derivative
$e_{ss} = y_{ss}$ = Steady-state error
ω_{bw} = Bandwidth
$\Phi_{desired}$ = Phase margin
T = Middle point of the new cross-frequency ω_{cnew}
$\sqrt{a}$ = Magnitude of where the lead phase is at its highest
K_a = Acceleration constant
K_v = Velocity constant
K_p = Position constant
db = Amplitude in decibels
a_k = Denominator, feed-backward
b_k = Numerator feed-forward
y = Output digital IIR recursive filter
x = Input digital IIR recursive filter
n = Sample number, iteration number, or time period number
Std = Standard
Opt = Optimized

Appendix B

Bosch Rexroth project description

Ball and Beam

Description

The ball and beam model consists of a long beam which can be tilted by a Rexroth servo motor together with a ball rolling back and forth on top of the beam.

The model is unstable. Even if the beam is horizontal, it will swing to one side or the other, and the ball will roll off the end of the beam. To stabilize the ball, a control system (ctrlX CORE) measures the position of the ball and adjusts the beam accordingly.

Tasks

- Calculation of the motion of the mechanics (differential equations)
- Design and build of the mechanics “ball and beam” which is attached to the electric drive and motor
- Selection of the sensor for measuring the ball position
- Simulation of the behaviour using either Python control library or Matlab
- Design of e.g a state observer in order to make the system stabile
- Programming of a motion sequence for controlling the model

Caution

Any dangerous axis movement must be prevented.

Figure B.1: Ball and Beam - Rexroth ctrlX

Appendix C

Datasheet A

C.1 Tables

C.1.1 Balluff BOD 23K-LA01-S92

Balluff Sensor key features

Special characteristics	Washdown
Series	23K
Dimension	51 x 23 x 52.4 mm
Interface	Analog, voltage 0...10 V linear rising/falling PNP/NPN/Auto-Detect NO/NC
Principle of operation	Photoelectric distance sensor BOD
Principle of optical operation	Light time-of-flight
Beam characteristic	Collimated
Light type	Laser Red light
Light spot size	5.5 x 7 mm at 5 m
Range	100...5000 mm
Accuracy	± 0.6 % FS (Full Scale)
Repeat accuracy	0.024 % FS (Full Scale)
Resolution	≤ 5.0 mm
Connection	Connector, M12x1-Male, 5-pin
Housing material	ABS
Operating voltage Ub	18...30 VDC
Approval/Conformity	CE cULus Ecolab WEEE

C.1.2 Safety, Velocity and Torque

Safety with Ball and Beam, ISO 15066-2016 (Body part head)			
Type	Clamping force (safety)	Velocity (safety)	Source
Velocity tip(tangential) and ω		1.2099 $[m/s]$ and 2.8697 $[\frac{Rad}{s}]$ F.4	Formula A.2 page 28
Torque from servo	27.404 $[Nm]$ F.4		Tabel A.2 page 24

C.1.3 Assembly Parts, Ball and Beam

Screws, Nuts, Washers and Clip				
Description/Placement	Type	Dimension	Size	Quantity
Base axle	Set screws	16mm	M5	2
	Nuts	4 x 8 mm	M5	2
Base for beam	Screws	20mm	M5	1
	Washer	5x15mm	M5	1
	Bolt	60mm	M5	1
	Washer	5 x 10mm	M5	1
	E-ring	5mm		1
Sensorholder	Bolt	40mm	M4	2
	Washer	5 x 10mm	M5	4
	Nuts		M4	2
	Screw	20mm	M5	1

Appendix D

ctrlX PLC

```
//-------------------------------------------------------------------------------------------------
//----------------------------  Axis-access with AxisInterface ----------------------------
//-------------------------------------------------------------------------------------------------
IF NOT TE_AxisInterfaceMainProg.InitDone OR TE_AxisInterfaceMainProg.Error THEN
  RETURN; // do not continue, when Motion is not in RUN or initialization failed.
END_IF

// Calculation of variables
rBallFeedback := (-1*iBallFeedback + 5000)*77.57; // Invert, move from plus minus and scale to real size
rBallFeedback := (rBallFeedback/1000); // Move decimals
gvl.rBallFeedback := rBallFeedback - 65; // moving zero point

rTempMotor := iTempMotor;
gvl.rTempMotorCalc := rTempMotor/10;

rForce := iForce;
gvl.rForceMotor := ABS(rForce/10);

arAxisCtrl_gb[1].Admin.ClearError := gvl.bClearError;
bError_Axis :=  arAxisStatus_gb[1].Diag.Error;

//Angle Generator
AngleGenerator(
    Enable:= bSinus,
    InOperation=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    Pause:= ,
    Frequency:= rSinusFreq,
    InPause=> ,
    OutputAngle=> rSinusAngle,
    ActScanTime=> );

SinusSignal(
    Enable:= bSinus,
    InOperation=> rSiunsInOp,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    Angle:= rSinusAngle,
    CurveType:= 1,
    Duty:= 50,
    DutyRamp:= rSinusDutyRamp,
    Rounding:= ,
    RoundingRamp:=,
    OutputValue=> rSinusOut ,
    ActScanTime=> );

rDriveInput := rSinusOut * 10;

IF bRemoteOn_gb = FALSE THEN
  //------------------------------
  // Axis control
  //------------------------------
  //arAxisCtrl_gb[1].PosMode.Position := gvl.rPIDCtrlOut;
  arAxisCtrl_gb[1].PosMode.Position := rDriveInput ; // position cmd of axis rDriveInput
  arAxisCtrl_gb[1].PosMode.Velocity := 1000; // velocity cmd of axis
  arAxisCtrl_gb[1].PosMode.DynValues.Acceleration := 1000;
  arAxisCtrl_gb[1].PosMode.DynValues.Deceleration := 1000;
  arAxisCtrl_gb[1].Admin._OpModeBits.MODE_POS_ABS := gvl.bEnable;// enabling the axis

  //------------------------------
  // Axis-Command: Switch all axes to ModeAH with bStopAll = TRUE
  //------------------------------
  IF bStopAll = TRUE THEN
      IF arAxisStatus_gb[1].Admin.Active THEN
        arAxisCtrl_gb[1].Admin._OpMode := ModeAH;
      END_IF
  ELSE
    // set to Mode Ab when bStop All = FALSE
      IF arAxisStatus_gb[1].Admin.Active AND arAxisStatus_gb[1].Admin._OpModeAckBits.MODE_AH  THEN
        arAxisCtrl_gb[1].Admin._OpMode := ModeAb;
      END_IF
  END_IF

END_IF
```

Figure D.1: Program for testing the beam response with a sinusoidal input

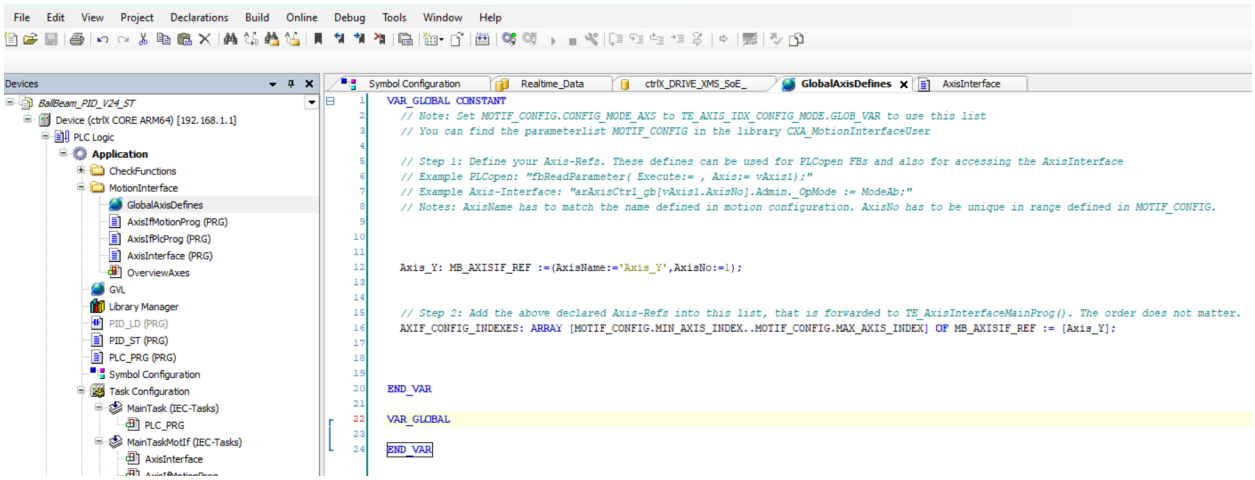

Figure D.2: Modified GlobalAxisDefines for axis Y

Appendix E

WebIQ

E.1 Google Chrome settings

If the 3D model still does not show, certain changes may need to be made in the settings for it to be displayed. Below shows how this can be done.

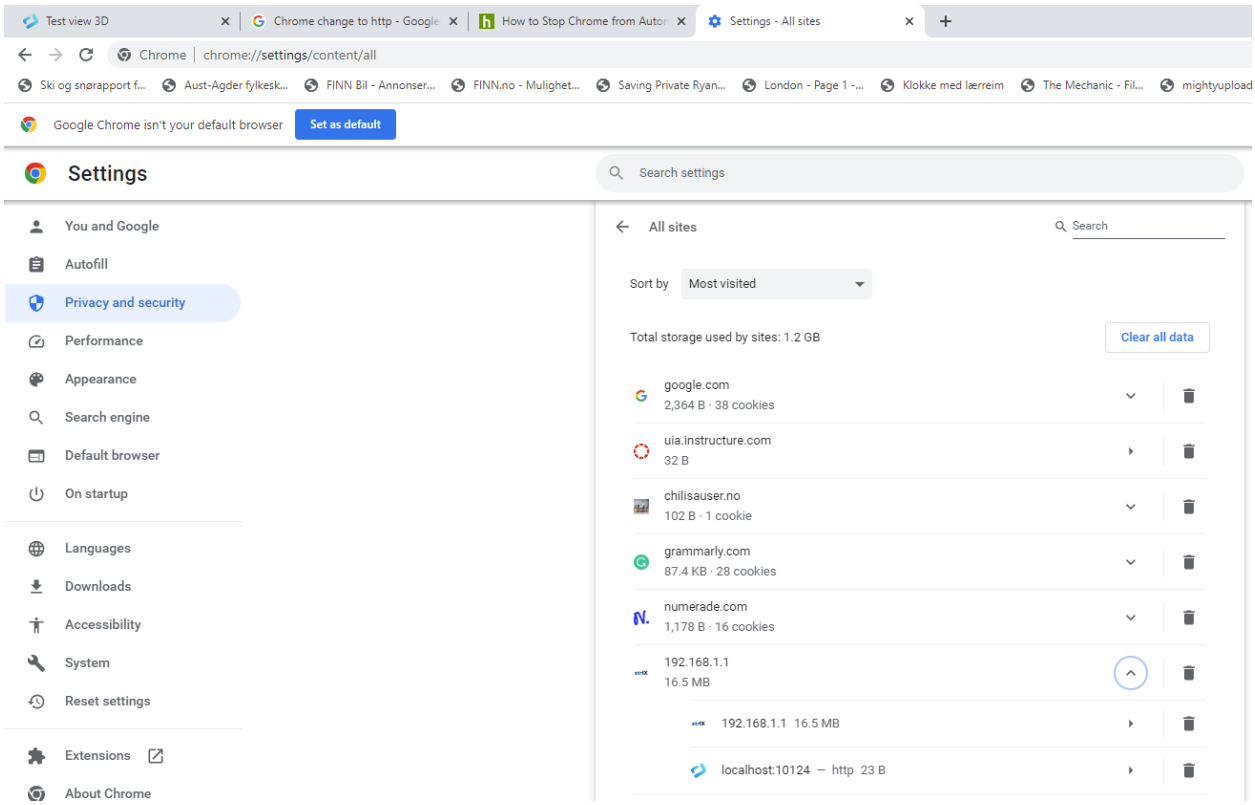

Figure E.1: Settings, Privacy and security

As shown in fig. E.1 must be changed under privacy and security settings. go to ipaddress `192.168.1.1` and press the drop-down menu.

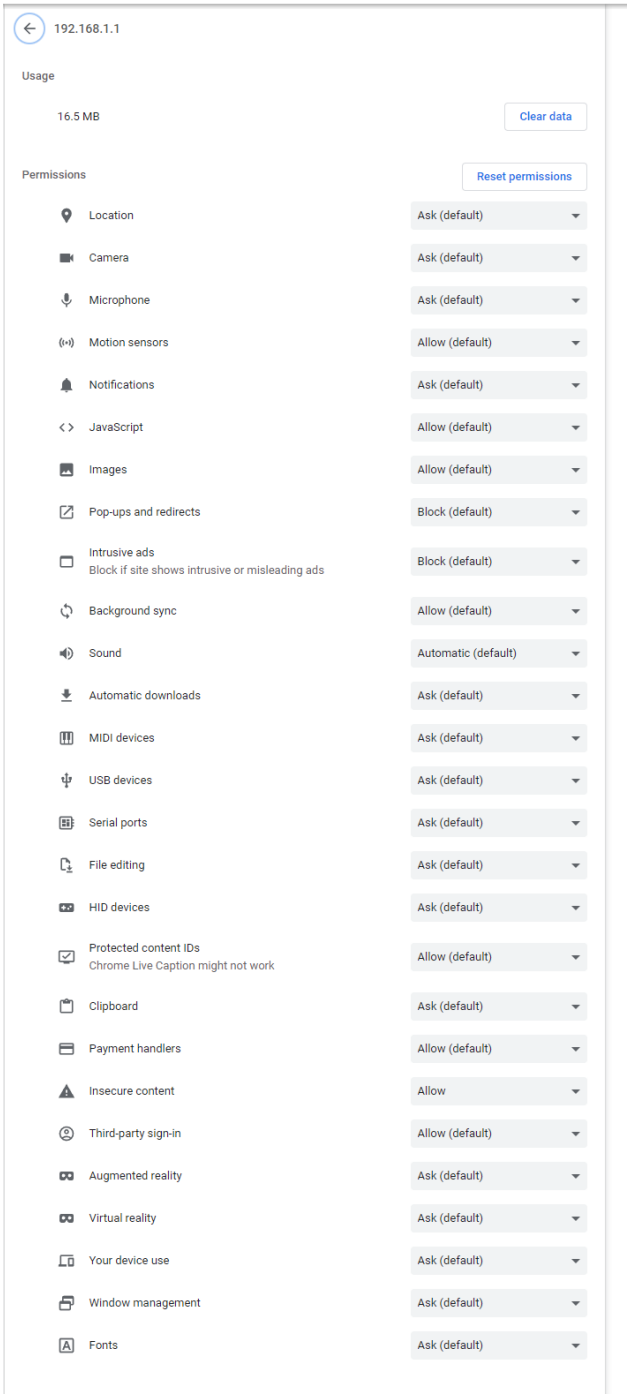

Figure E.2: Grant access 192.168.1.1

As shown in fig. E.2 access must be granted to **"Allow"** or **"Ask"**, several attempts may need to be made here.

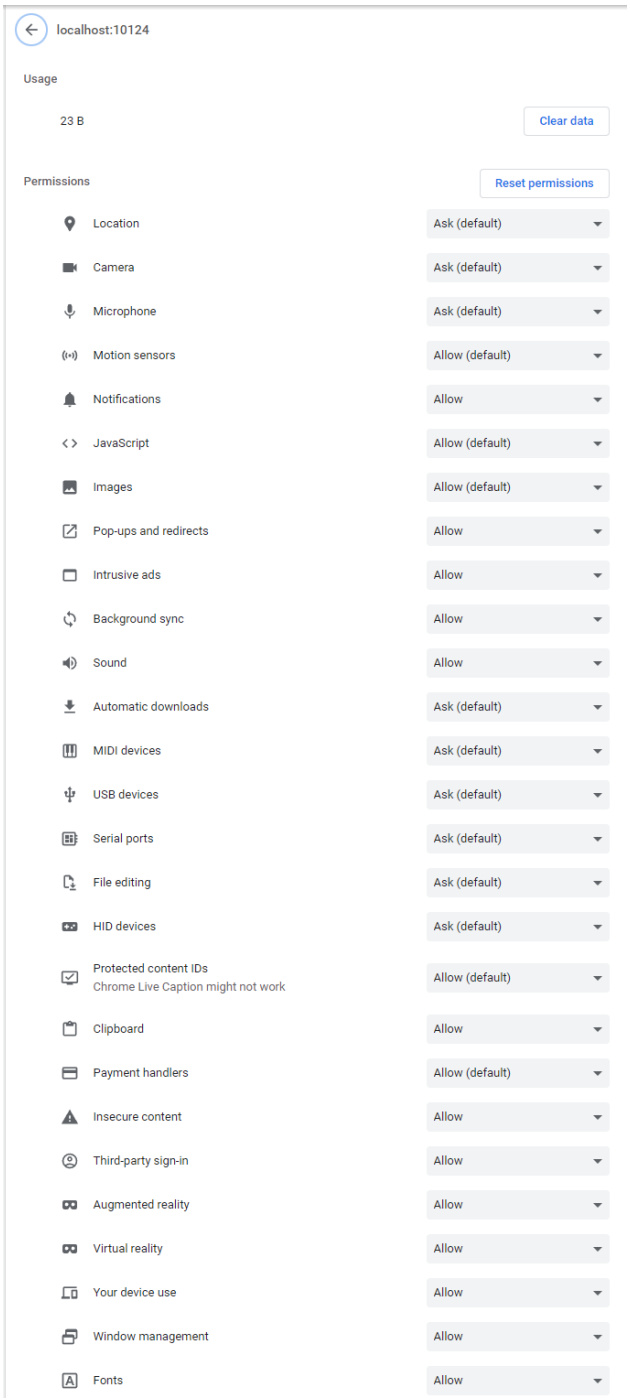

Figure E.3: Grant access localhost:10124

As shown in fig. E.3 access must be granted to **"Allow"** or **"Ask"**, several attempts may need to be made here.

Appendix F

Matlab

F.1 System, Lead compensator

Contents

Ball and beam filter and velocity controller calculations

```
clc; clear; close all;
s =tf('s');
```

Requirements

Velocity loop

```
Tr_vel = 0.3; %[s] Rise time
OS_vel = 5;%[%] Overshoot
Ts_vel = 1.5; %[s] Settling time
e_ss_vel = 0.1; % Steady state error
```

Low pass filter at 2 times the -3db frequency for the system response

```
Ts = 0.002; % [s] Cycle time/discrete sample time
Sys_resp_freq = 4.4;%[rad/s] System frequency at -3db magnitude
omega_zero = 2*Sys_resp_freq; %[rad/s] low pass filter cut off frequency


LF_s = (omega_zero)/(s+omega_zero) % normal frequency
 % Convert from continous time to discrete time
LF_z_d = c2d(LF_s,Ts);

% Get the coefficients
% den is the feed forward coefficients b0, b1, b2...
% num is the feed backward coefficients a1, a2, a3...
[num_fw_b, den_bw_a] = tfdata(LF_z_d,'v')
```

Model of ball motion

```
mb = 3e-3; % [kg] mass of ball
r_b = 0.02; % [m] ball radius
g = -9.81; % [m/s^2] gravity
J_b = (2/3)*mb*r_b^2; % [kgm^2] ball inertia


P_bb = -(mb*(r_b^2)*g)/((mb*(r_b^2))+J_b)/s^2%[m]%Ball transfer function
```

2nd order transfer function from step response of drive and motor

```
% Values found from step response plot of beam motion
s =tf('s');
start_value = 0;
U = 1; % Step input magnitude
y_ss = 1; % steady state
y_max = 1.0114; % overshoot
Tr_90 = 81.525; %[s] time to 90% of y_ss
Tr_10 = 81.435; %[s] time to 10% of y_ss
Tp_max = 81.657; %[s] time to peak
T_step = 81.373; %[s] start step
pstOS = ((y_max-y_ss)/y_ss)*100; %[%] percent overshoot
% Calculation
zeta = (-log(pstOS/100))/(sqrt(pi^2+log(pstOS/100)^2));% Damping ratio
T_p = Tp_max-T_step; %[s] Peak time
omega_n = pi/(T_p*sqrt(1-zeta^2)); %[rad/s] Natural frequency
K_a = y_ss/U; % Amplification gain
% Resulting transfer function
P_s = K_a/((s^2/(omega_n^2))+((2*zeta)/omega_n)*s+1)%[rad]
transfer function drive/motor
```

step response of ball on beam

```
G_s = P_s*P_bb*s; %[m/s]TF drive/motor/beam and ball, velocity output
G_s_LF = P_s*P_bb*LF_s*s;%[m/s]TF drive/motor/beam and
ball, velocity output and LowPass filter

figure
h = bodeplot(G_s); % open  loop bodeplot of drive/motor/beam and ball
legend('G(s)')
title('Bodeplot, base system G(s)')



figure
h = bodeplot(G_s); % open  loop bodeplot of drive/motor/beam and ball
hold on
h_LF = bodeplot(G_s_LF);%open loop bodeplot of
drive/motor/beam and ball
with LPfilter
legend('G(s)','Single low pass filter G(s)LF')
title('Bodeplot, base system G(s), G(s) w/low pass filter at 8.8 [rad/s]')
```

```
% Lead controller, velocity
phi_desired = 70-OS_vel; %[deg] the desired phase margin
omega_bw = 1.8/Tr_vel; %[rad/s] the desired bandwidth

% Gain from final value theorem
K = 370.6/(e_ss_vel*2181); %[-] Gain to reach e_ss_vel

% Gain K from final value theorem is added to G_s
Gk_s= K*G_s;
figure
h = bodeplot(G_s);
hold on
gk = bodeplot(Gk_s);
legend('G(s)','Gk(s)')
title('Bodeplot without and with k gain')



%margin(Gk_s); % bodeplot with gain and phase margins for Gk_s
[Gm_Gk_s,Pm_Gk_s,Wcg_Gk_s,Wcp_Gk_s] = margin(Gk_s);
```

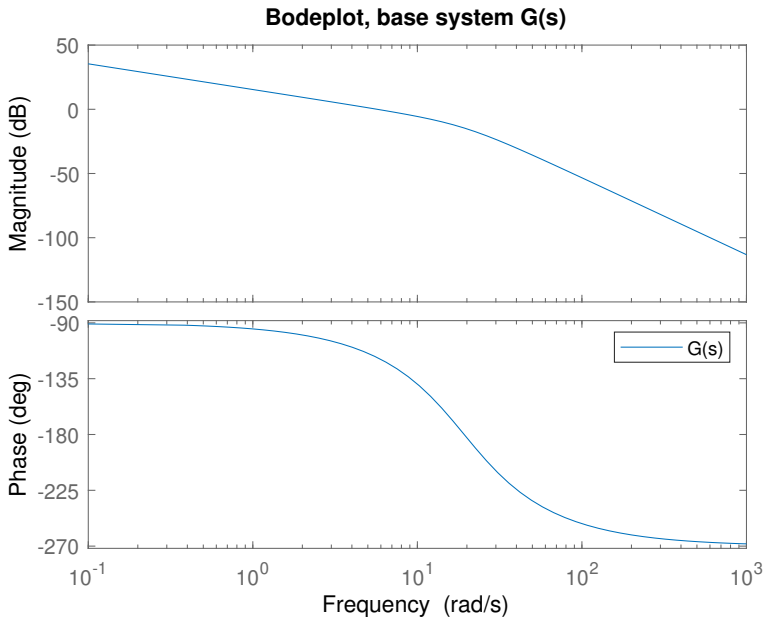

Figure F.1: Bode G(s)

Results from velocity closed loop step response bodeplot

```
% Values from bodeplot for Gk_s and G_s
phi_new = Pm_Gk_s;%[deg] phase margin for Gk_s
pm_add = 12;%[deg] + 5-15 deg added phase margin to counteract shift
phi_m = (phi_desired-phi_new)+pm_add;%[deg] the required addition of phase margin
```

lead compensator

```
% Finding s, T and omega_m for H(s)
a= (1+sind(phi_m))/(1-sind(phi_m));%[-]
```

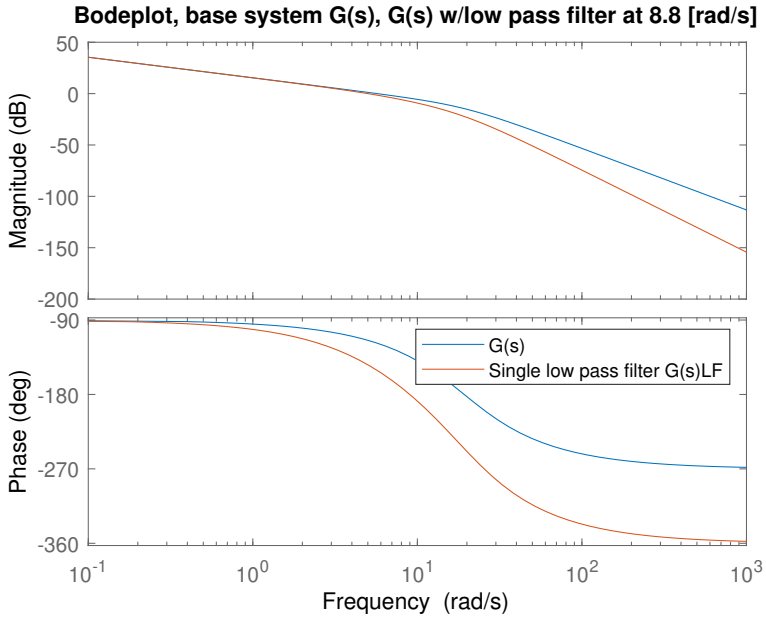

Figure F.2: Bode G(s), G(s)LF

```
KH_Lead = 20*log10(1/sqrt(a));

% From Gk_s bodeplot, omega_m at KH_Lead magnitude
omega_m = 14;%[rad/s]

T = 1/(omega_m*sqrt(a));

% Final lead compensator H(s)


H_s = ((a*T*s+1)/(T*s+1)) % Lead compensator without gain
HK_s =K*H_s; % Lead compensator with K gain
%H_G_s = H_s*G_s;% Lead compensated system, no gain
H_Gk_s = HK_s*G_s;% Lead compensated system, with K gain

% Bodeplot
figure
h = bodeplot(G_s);
hold on
gk = bodeplot(Gk_s);
%h_s = bodeplot(H_s); % bodeplot of lead compensator
%h_g_s = bodeplot(H_G_s); %Bodeplot of lead compensated system
h_gk_s = bodeplot(H_Gk_s); %Bodeplot of lead compensated system with gain
legend('G(s)','Gk(s)','HGk(s)')
title('Bodeplot system, system with gain and lead compensated system')
```

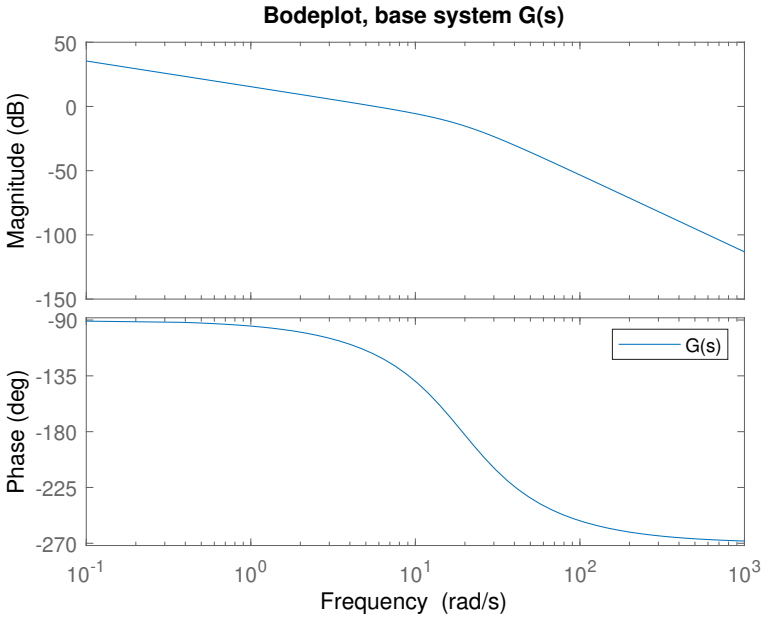

Figure F.3: Bode G(s), Gk(s)

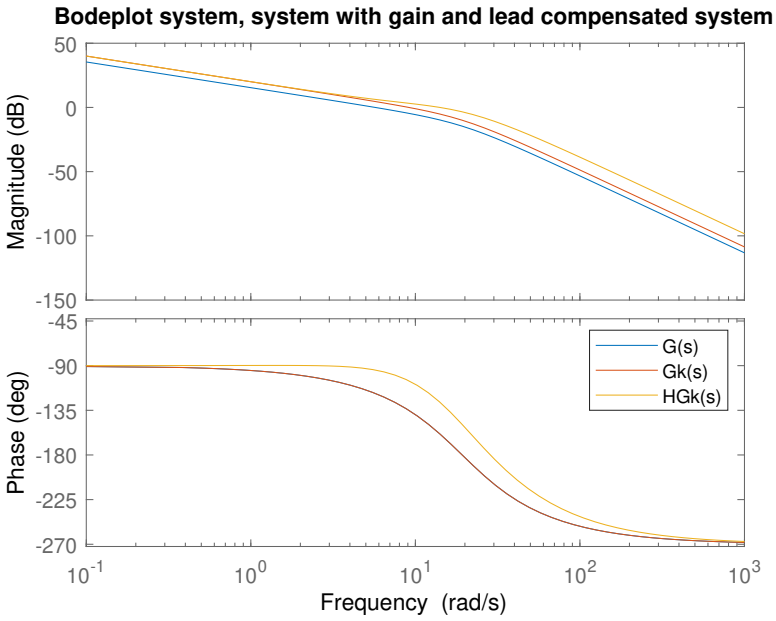

Figure F.4: Bode G(s),Gk(s),HGK(s)

F.2 Ball rolling

Contents

- Parameters from Beam and Ball
- Ball and Beam analytically calculates position, velocity and acceleration.
- Data from Simscape, 5 deg
- Save data form simscape to Excel file:
- Loads data from measurements in the CtrlX program (real measurements)
- Put in ctrlx time and pos values to plot for all simulated values

```
clear; clc; close all;
```

Parameters from Beam and Ball

```
s = 468.8; % [mm] lengt of surface
Theta = 5; % angle of the rolling surface
data_tracker = readtable('Tracker.xlsx'); % data from Tracker (film)
ping_pong_tracker = table2array(data_tracker);
g = 9810; %[mm/s^2]
time_ping_pong =ping_pong_tracker(:,1)-0.6529333333; % s,
% Correction of time to start at zero
liste_Ping_pong_tracker_x = (ping_pong_tracker(:,2)-0.009219700051)*10^3; % mm,
% Correction of position to start at zero
```

Ball and Beam analytically calculates position, velocity and acceleration.

```
a_hul = (3/5)*g*sind(Theta)*ones(size(time_ping_pong)); % rollers on
% surface higest acceleration (rolls on the tip).

% takes into account 2 rails
R_ball = 40.1*1/2; % mm
d_beam_17 = 16.667; % mm

a_hul_2_rail =(g*sind(5))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4)) ...
    )*ones(size(time_ping_pong)); % mm/s^2Acceleration analytically on rods

v_hul =  (3/5)*g*sind(Theta)*time_ping_pong;% mm/s, Velocity analytically on a surface
v_hul_2_rail =a_hul_2_rail.*time_ping_pong; % mm/s Velocity analytically on rods

s_hul =  (3/5)*g*sind(Theta)*(1/2)*time_ping_pong.^2;% mm,
% Position analytically on a surface
s_hul_2_rail = a_hul_2_rail(1)*(1/2)*time_ping_pong.^2;% mm analytically on rods

%% Max rotation speed if the ball is not to move on the beam
acceleration_5_deg = 473.451 % mm/s^2, classic mek based on newtons laws
omega_stabel = sqrt(acceleration_5_deg/335.1); % Rad/s, rotation of beam
if beam at 5 deg and ball at
% the farest from center. If the rotation is higher, the ball will roll up
% the beam instead of down at 5 deg.
% omega_stabel = 1.188639901935273%
v_at_tip = omega_stabel*0.3351; % m/s, tagential velocity. If the ball is
farthest on the beam
% v_at_tip = 0.39831323113851%
```

```
%Finds the numerical derivative from lists we have defined
% ourselves. Here we have

%Derivates the position to find the velocity
dtA = diff(time_ping_pong);%Calculates x2-x1 etc... That is,
% takes the next minus the previous and finds the difference between them
dy_A = diff(liste_Ping_pong_tracker_x);% Calculate y2-y1 aso...
yDer_A = dy_A./dtA;% find the numerical derivative by taking the
% difference in y divided by the difference in x

% Derivates the velocity to acceleration
dt2A = diff(time_ping_pong);%Calculates x2-x1 etc... That is,
% takes the next minus the previous and finds the difference between them
dy2_A = diff(yDer_A);% Calc y2-y1 aso...
dy2_A(end+1) = dy2_A(end);
yDer_2A = dy2_A./dt2A;% find the numerical derivative by taking the
% difference in y divided by the difference in x

yDer_2A_mean= yDer_2A(1:82,1);
mean_a_A = mean(yDer_2A_mean)*ones(size(time_ping_pong)); % Finds
% the average of the acceleration from the tracker
```

Data from Simscape, 5 deg

DATA FOM SIMSCAPE PLOT: data_pos_simscape = (Data_5Deg.signals(2).values(1:end,7)) ; data_vel_simscape = (Data_5Deg.signals(2).values(1:end,5)) ; data_acc_simscape = (Data_5Deg.signals(2).values(1:end,10)) ; data_time_simscape = (Data_5Deg.time) ;

Save data form simscape to Excel file:

Tabel_5deg_simacape = table(data_time_simscape, data_pos_simscape, ... data_vel_simscape,data_acc_simscape, 'VariableNames', {'Time', 'Position', 'Velocity', 'Acceleration'}); filename = 'Ball_roll_5deg_simscape.xlsx'; writetable(Tabel_5deg_simacape, filename);

```
data_5deg_simscape = readtable('Ball_roll_5deg_simscape.xlsx');
data_5deg_simscape = table2array(data_5deg_simscape);

time_simscape = data_5deg_simscape(:,1);
pos_simscape = data_5deg_simscape(:,2)-2;
vel_simscape = data_5deg_simscape(:,3);
acc_simscape = data_5deg_simscape(:,4);
```

Loads data from measurements in the CtrlX program (real measurements)

Specify the file name and location

```
filename = 'BallPos_5deg_beam_angle_roll_calib.trace.csv';
% Read the data from the CSV file
data_CtrlX = readmatrix(filename);
% Convert time values from milliseconds to seconds
data_CtrlX(:,3) = data_CtrlX(:,3) / 1000;
% Corectin axis to start at 0.
time_CtrlX = data_CtrlX(2007:2831,3)-3.985; %Time from start to end  of ball rolling
```

```
pos_CtrlX = data_CtrlX(2007:2831,10); %Time from start to end  of ball rolling

% Calculating the velocity and the acceleration for the CtrlX values
%Finds the numerical derivative from lists

%Derivates the position to find the velocity
dt_CtrlX = diff(time_CtrlX);%Calculates x2-x1 etc... That is,
% takes the next minus the previous and finds the difference between them
dy_CtrlX = diff(pos_CtrlX);% Calculate y2-y1 aso...
yDer_CtrlX = dy_CtrlX./dt_CtrlX;% find the numerical derivative by taking the
% difference in y divided by the difference in x



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use a low pass filter on signal position
% Define the filter parameters
fc = 6.1; % cutoff frequency in Hz
fs = 500; % sampling frequency in Hz
order = 2; % filter order
% Calculate the filter coefficients
[b, a] = butter(order, fc/(fs/2), 'low');
% Apply the filter to signal
x = yDer_CtrlX; % signal
y_CtrlX = filter(b, a, x);
% Plot the frequency response of the filter
% figure
% freqz(b, a, 1024, fs);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Derivates the velocity to acceleration (from velocity filtered)
dt2_CtrlX = diff(time_CtrlX);%Calculates x2-x1 etc... That is,
% takes the next minus the previous and finds the difference between them
dy2_CtrlX = diff(y_CtrlX);% Calc y2-y1 aso...
dy2_CtrlX(end+1) = dy2_CtrlX(end);
yDer2_CtrlX = dy2_CtrlX./dt2_CtrlX;% find the numerical derivative by taking the
% difference in y divided by the difference in x

mean_a_CtrlX222 = mean(yDer2_CtrlX(30:700))*ones(size(time_CtrlX)); %
```

Put in ctrlx time and pos values to plot for all simulated values

```
%Plot possission and time
figure
plot(time_ping_pong,s_hul, 'Color',[1 0.6 0])
hold on
plot(time_ping_pong,liste_Ping_pong_tracker_x, 'Color',[0.4 0.1 0.8])
hold on
plot(time_ping_pong,s_hul_2_rail, 'Color',[0 1 0])
grid on
plot(time_simscape',pos_simscape', '--', 'Color',[0 0 0])
hold on
plot(time_CtrlX, pos_CtrlX, '-.', 'Color',[1 0.2 0.6]);
xlabel('Time [s]')
```

```
ylabel('Position [mm]')
title('Position ball 5\circ')
legend('Analytic ball rolls on a surface','Real on rods Tracker', ...
    'Analytic ball on rail','Simscape model','Real CtrlX','location','northwest')

%Plot velocity and time
figure
grid on
plot(time_ping_pong,v_hul, 'Color',[1 0.6 0]);
hold on
plot(time_ping_pong(1:end-1),yDer_A, 'Color',[0.4 0.1 0.8])% Plot the
% numerical derivative
hold on
plot(time_ping_pong,v_hul_2_rail, 'Color',[0 1 0])
grid on
plot(time_simscape',vel_simscape', '--', 'Color',[0 0 0])
hold on
plot(time_CtrlX(1:end-1),y_CtrlX, '-.', 'Color',[1 0.2 0.6])% Plot
% the numerical derivative
xlabel('Time [s]')
ylabel('Velocity [mm/s]')
title('Velocity ball 5\circ')
legend('Analytic ball rolls on a surface','Real on rods Tracker', ...
    'Analytic ball on rail','Simscape model','Real CtrlX','location','northwest')

%Plot acceleration and time
figure
plot(time_ping_pong,a_hul, 'Color',[1 0.6 0])
hold on
plot(time_ping_pong,mean_a_A, 'Color',[0.4 0.1 0.8])
hold on
plot(time_ping_pong,a_hul_2_rail, 'Color',[0 1 0])
hold on
grid on
plot(time_simscape',acc_simscape', '--' , 'Color',[0 0 0])
hold on
plot(time_CtrlX,mean_a_CtrlX222, '-.', 'Color',[1 0.2 0.6])
xlabel('Time [s]')
ylabel('Acceleration [mm/s^2]')
title('Acceleration ball 5\circ')
legend('Analytic ball rolls on a surface','Mean real on rods Tracker', ...
    'Analytic ball on rods','Simscape model','Real CtrlX','location','best')
```

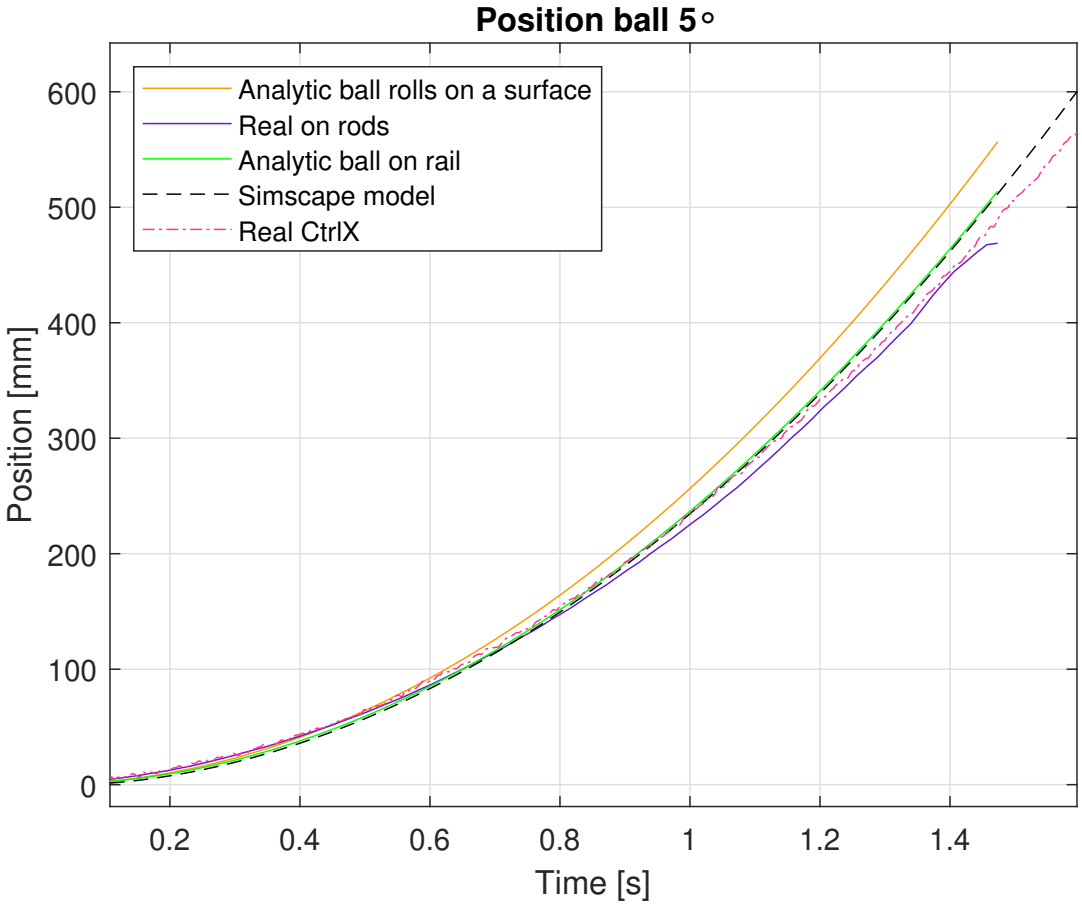

Figure F.5: Position 5deg rolling

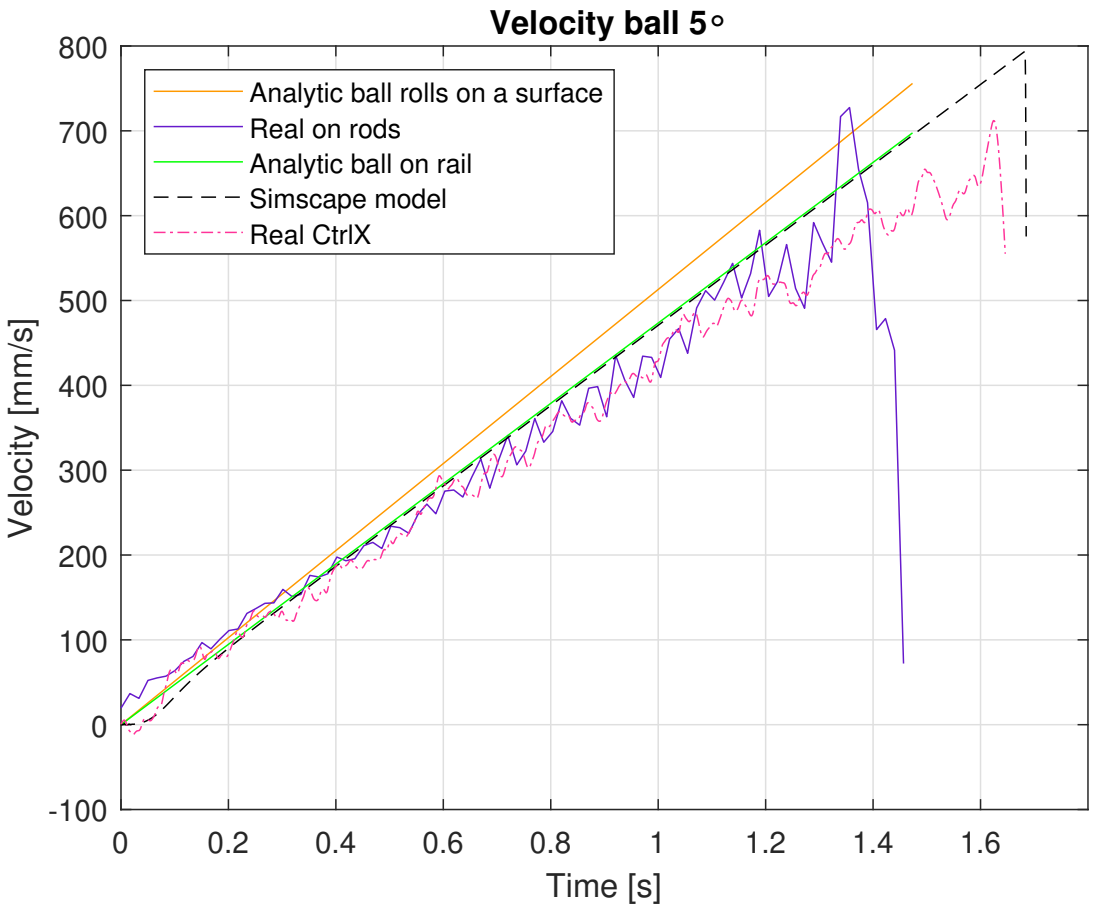

Figure F.6: Velocity 5deg rolling

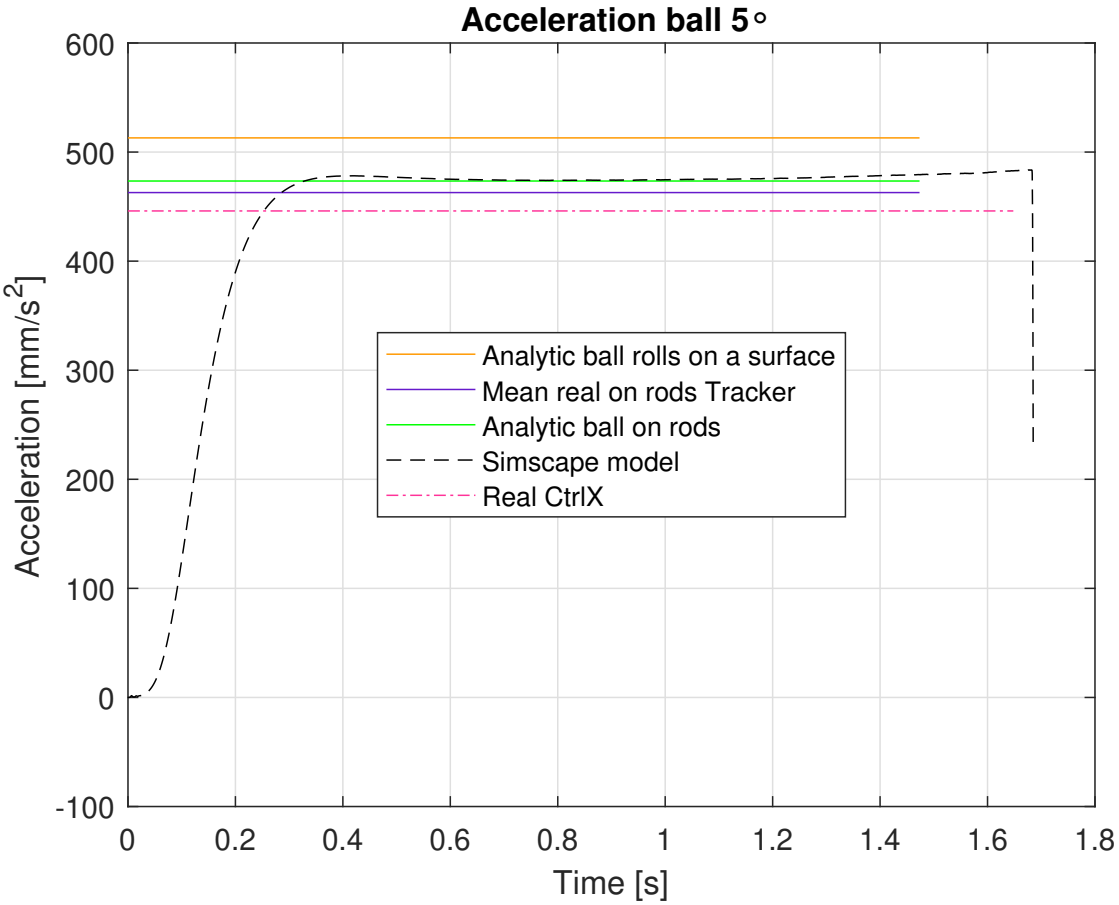

Figure F.7: Acceleration 5deg rolling

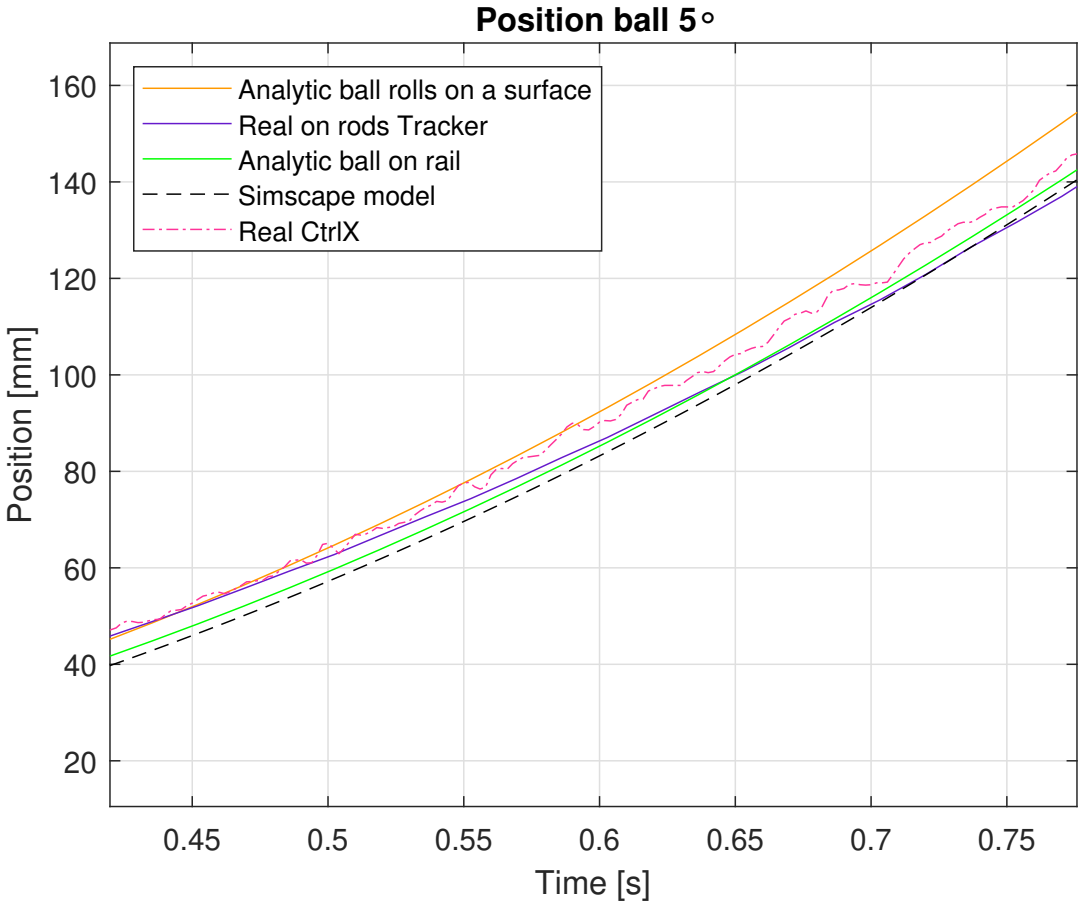

Figure F.8: Position 5deg rolling close up

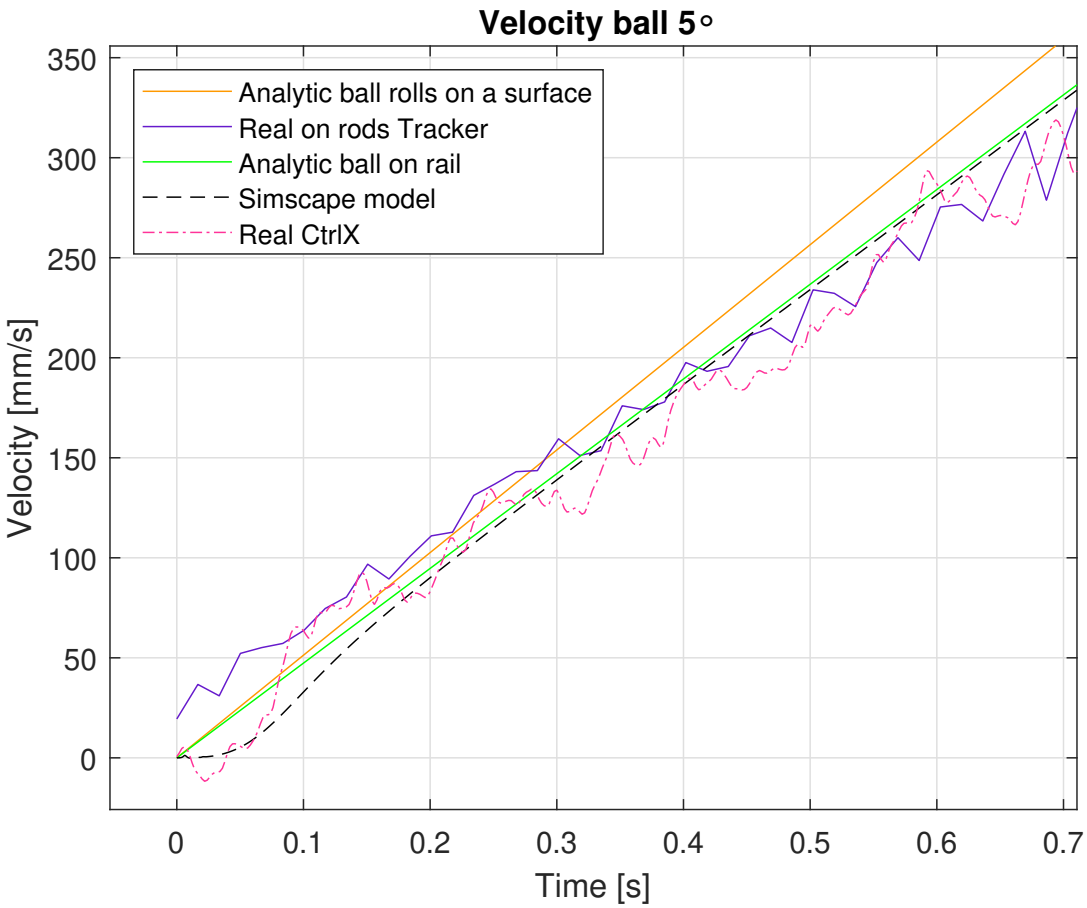

Figure F.9: Velocity 5deg rolling close up

F.3 Finds dimensions and weight

Here the weight of the ball and beam length are calculated to set the final specifications. Here we want to find the acceptable torque for the servo and velocity for the beam.

Data coms from the pdf: "MS2N Synchronous Servomotors"

First the servo specs:

Type: ***MS2N03-B0BYN-HMSH1-NNNNN-NN***

3 PM-MOTOR

n(N) 6470 1/min

n(max) 9000 1/min

U(max) AC 600 V

Figure F.10: servo MS2N03-B0BYN-HMSH1-NNNNN-NN

Beam with no ball center of mass Balanced:

Only the beam:

Mass properties of Sammenstilling V3.2-LAPTOP-44QB4H0M

Configuration: Default

Coordinate system: Coordinate System1

Mass = 0.40237903 kilograms

Volume = 0.00030401 cubic meters

Surface area = 0.19049638 square meters

Center of mass: (meters)

X = 0.00733656

Y = -0.00497760

Z = 0.04227204

Principal axes of inertia and principal moments of inertia: (kilograms * square meters)

Taken at the center of mass.

Ix = (0.99996290, -0.00808711, 0.00296667) Px = 0.00026788

Iy = (0.00830504, 0.81368510, -0.58124658) Py = 0.03606421

Iz = (0.00228667, 0.58124965, 0.81372207) Pz = 0.03615710

Moments of inertia: (kilograms * square meters)

Taken at the center of mass and aligned with the output coordinate system.

Lxx = 0.00027053 Lxy = -0.00028960 Lxz = 0.00010602

Lyx = -0.00028960 Lyy = 0.03609325 Lyz = -0.00004480

Lzx = 0.00010602 Lzy = -0.00004480 Lzz = 0.03612540

Moments of inertia: (kilograms * square meters)

Taken at the output coordinate system.

Ixx = 0.00099952 Ixy = -0.00030430 Ixz = 0.00023081

Iyx = -0.00030430 Iyy = 0.03683393 Iyz = -0.00012946

Izx = 0.00023081 Izy = -0.00012946 Izz = 0.03615703

With the ball on the tip:

Mass properties of Sammenstilling V3.2-LAPTOP-44QB4H0M

Configuration: Default

Coordinate system: Coordinate System1

Mass = 0.40520721 kilograms

Volume = 0.00030665 cubic meters

Surface area = 0.20033258 square meters

Center of mass: (meters)

X = 0.00962272

Y = -0.00488233

Z = 0.04232563

Principal axes of inertia and principal moments of inertia: (kilograms * square meters)

Taken at the center of mass.

Ix = (0.99996565, -0.00767245, 0.00313674) Px = 0.00026948

Iy = (0.00807443, 0.81612611, -0.57781742) Py = 0.03636670

Iz = (0.00187330, 0.57782290, 0.81616003) Pz = 0.03645922

Moments of inertia: (kilograms * square meters)

Taken at the center of mass and aligned with the output coordinate system.

Lxx = 0.00027196 Lxy = -0.00027704 Lxz = 0.00011308

Lyx = -0.00027704 Lyy = 0.03639547 Lyz = -0.00004450

Lzx = 0.00011308 Lzy = -0.00004450 Lzz = 0.03642798

Moments of inertia: (kilograms * square meters)

Taken at the output coordinate system.

Ixx = 0.00100753 Ixy = -0.00029608 Ixz = 0.00027812

Iyx = -0.00029608 Iyy = 0.03715890 Iyz = -0.00012824

Izx = 0.00027812 Izy = -0.00012824 Izz = 0.03647516

Contents

- This is the specs off beam:
- Data from servo
- Static torque if ball is at the tip of left side seen from front.
- Stop time when moving
- Inertia mismatch
- Motor torque during operation at standstill
- Expect continous operation

Specifications for real beam used in the physical system:

Mass properties of Sammenstilling V3.2-LAPTOP-44QB4H0M

Configuration: Default

Coordinate system: – default –

Mass = 0.30555769 kilograms

Volume = 0.00028934 cubic meters

Surface area = 0.17884038 square meters

Center of mass: (meters)

X = 0.12748815

Y = -0.00495633

Z = 0.03996905

Principal axes of inertia and principal moments of inertia: (kilograms * square meters)

Taken at the center of mass.

Ix = (0.99951750, -0.01678212, 0.02613662) Px = 0.00023352

Iy = (0.02707604, 0.88309313, -0.46841586) Py = 0.01772599

Iz = (-0.01522006, 0.46889752, 0.88312143) Pz = 0.01781718

Moments of inertia: (kilograms * square meters)

Taken at the center of mass and aligned with the output coordinate system.

Lxx = 0.00025042 Lxy = -0.00029277 Lxz = 0.00045820

Lyx = -0.00029277 Lyy = 0.01774112 Lyz = -0.00004543

Lzx = 0.00045820 Lzy = -0.00004543 Lzz = 0.01778516

Moments of inertia: (kilograms * square meters)

Taken at the output coordinate system.

Ixx = 0.00074606 Ixy = -0.00048584 Ixz = 0.00201519

Iyx = -0.00048584 Iyy = 0.02319555 Iyz = -0.00010596

Izx = 0.00201519 Izy = -0.00010596 Izz = 0.02275896

This is the specs off beam:

```
clc; close all; clear;
beam_length = 710 %mm
ball_max_wigth = 2.86 % g
alpha_max =  27.5094; % rad/s^2, Max angular acceleration.
velocity_max_tip = alpha_max*0.4216 % m/s^2 tangential acceleration max at tip.

% Assume the beam with the ball.
center_off_mass_from_rotation_dx = 2.33 % mm
center_off_mass_from_rotation_dy = 5.17 % mm

moment_of_initertia_beam = 0.03615703% kg*m^2, ball not considerd
Izz =  0.03647516 % kg*m^2 with the ball at the tip far est from sensor 335.01mm
Izz_real = 0.02275896 % kg*m^2 real moment of inertia


beam_length =

   710


ball_max_wigth =
```

```
    2.8600

velocity_max_tip =

   11.5980

center_off_mass_from_rotation_dx =

    2.3300

center_off_mass_from_rotation_dy =

    5.1700

moment_of_initertia_beam =

    0.0362

Izz =

    0.0365

Izz_real =

    0.0228
```

Data from servo

MS2N03-B____-____1

```
M_4_Nm = 1.80 %Holding torque
M_1_Nm = 1.3 % Dynamic braking torque
t_1_ms = 8 % Maximum connection time
t_2_ms = 35 % Maximum disconnection time

M_4_Nm =

    1.8000

M_1_Nm =

    1.3000
```

```
t_1_ms =

     8


t_2_ms =

    35
```

Static torque if ball is at the tip of left side seen from front.

```
% beam in equlibrium
beam_ball_at_tip = 0.33501% m, This is from center of beam to the
senter off ball at the tip longest way from sensor.
g = cosd(20)*9.81 % m/s^2
Static_torque_down_from_ball = ball_max_wigth*10^(-3)*g  % kg*m/s^2 = N,
force generated from gravety at tip horizontally
Static_torque_on_axle = Static_torque_down_from_ball*beam_ball_at_tip  %  Nm,
torque generated from weigth about rotation
```

```
beam_ball_at_tip =

    0.3350


g =

    9.2184


Static_torque_down_from_ball =

    0.0264


Static_torque_on_axle =

    0.0088
```

Stop time when moving

```
J_fremd = Izz_real % kg*m^2  External inertia [kgm2]
J_rot = 0.000030 % Moment of inertia of motor [kgm2]
J_ges = J_rot+J_fremd % Moment of inertia of complete system [kgm2]
n =  6470 % Nominal speed [1/min] (rated speed 100K)
M_1 = M_1_Nm
M_6 = Static_torque_on_axle % M 6 Load torque [Nm]
t_Br = (J_ges*n)/(9.55*(M_1+M_6))
```

```
J_fremd =

    0.0228


J_rot =

   3.0000e-05


J_ges =

    0.0228


n =

        6470


M_1 =

    1.3000


M_6 =

    0.0088


t_Br =

   11.7962
```

Inertia mismatch

J_ges_real = J_rot+J_fremd_real % Moment of inertia of complete system [kgm2]

```
IZZ_real = Izz_real % kg*m^2  External inertia [kgm2]
Inertia_Mismatch_Real = (IZZ_real)/J_rot % 706.7 inertia mismatch
is very high this becomes problematic.
%%%%%%%%%%%%%% Simulated system under  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
J_L = IZZ_real % kg*m^2  External inertia [kgm2]
n = 10 % Gearbox for 1/10
J_LM = J_L/(n^2) % The equivalent mass moment of inertia on the motor shaft will then be
Inertia_Mismatch_gear_box = J_LM/J_rot % <10, but is still over 10.
n_optimal = sqrt(J_L/J_rot) % Optimal gear ratio.


IZZ_real =

    0.0228
```

```
Inertia_Mismatch_Real =

  758.6320


J_L =

    0.0228


n =

    10


J_LM =

   2.2759e-04


Inertia_Mismatch_gear_box =

    7.5863


n_optimal =

   27.5433
```

Motor torque during operation at standstill

```
F_0 = 0.95 % Self cooling 60K,  tabel23 page 57 Pdf
M_0_60K = 0.73 % Nm, Tabel page 64
M_0_star = F_0*M_0_60K % Nm, the continuous torque that can be
output at standstill M0* page 57


F_0 =

    0.9500


M_0_60K =

    0.7300


M_0_star =

    0.6935
```

Expect continous operation

"Bosch Rexroth recommends to select the S1-60K characteristic curve. The characteristiccurves are specified for S1-100K and S1-60K. The motor utilization is predominantly influenced by the installation situation." page 56.

```
rated_torque_100K = 0.54 % Nm, Table page 64
max_toque_cold = 3.75 % Nm, Table page 64
max_toque_warm = 3.46 % Nm, Table page 64
Holding_torque = 1.8 % Nm, Table page 64
```

```
rated_torque_100K =

    0.5400


max_toque_cold =

    3.7500


max_toque_warm =

    3.4600


Holding_torque =

    1.8000
```

F.4 Safe force and velocity

Contents

- Energy limit values based on the body region model.
- Biomechanical limits

Here, maximum speed and power must be calculated on the body part with the lowest tolerance for injury/pain. Based on ISO 15066-2016

```
clc; close all; clear;
%  defenition of variable real system :
Beam_weigth = 0.30555769  % Kg, only beam. No ball.
m_ball = 0.00283 % kg Only the ball.
weigth_beam_with_ball = Beam_weigth+m_ball % Kg
beam_lengt_from_center =  0.4216 % m, Distance from center to the farest
```

```
Beam_weigth =

    0.3056


m_ball =

    0.0028


weigth_beam_with_ball =

    0.3084


beam_lengt_from_center =

    0.4216
```

Energy limit values based on the body region model.

```
max_juele_face = 0.11 % juele, Nm. page 28, tabel A.4.
```

```
max_juele_face =

    0.1100
```

Biomechanical limits

```
Maximum_permissible_force = 65 % N, Face masticatory muscle. Tabel A.2 page 24.

mass_Head = 4.4 % kg, page 27, tabel A.3.

% finding max velocity relativ to hiting the object.
```

```
m_R = (Beam_weigth/2+m_ball) % kg, formula A.4 page 29.
my = (1/mass_Head+1/m_R)^-1 % kg, formula A.3 page 29.

V_relativ_m = sqrt((max_juele_face*2)/my) % m/s max velocity
of the end of beam tip. formula A.2 page 28.
V_relativ_mm = V_relativ_m*10^3 % mm/s
omega = V_relativ_m/(beam_lengt_from_center) % Rad/s

% finding the max safe torque of from servo
safe_t_off = Maximum_permissible_force*beam_lengt_from_center % Nm


Maximum_permissible_force =

    65


mass_Head =

    4.4000


m_R =

    0.1556


my =

    0.1503


V_relativ_m =

    1.2099


V_relativ_mm =

   1.2099e+03


omega =

    2.8697


safe_t_off =

   27.4040
```

F.5 Velocity and aceleration

Here we want to find the maximum angular acceleration of the ball downwards in the worst-case scenario. This is to set a limit in the system to make sure we work within a safe limit.

Not considering folowing: air resistance and rolling resistance.

Assume: No sliding ogly pure roll and no friction do to rooling.

Angle $\pm 20°$, working in area of 40 deg.

Picture on the force acting:

```
clc;close all; clear;
% defenition of variable:
m = 2.75 % grams
g = 9.81 % m/s^2
Length_of_beam = 0.71 % m
max_length_from_center = 0.3351 % m,The furthest distance from the center the ball can travel



% The worst cenario is when the ball is at the tip and the beam is
% accselerating down from horizontal.
a_max_tip = 9.81 % m/s^2

% % finding max angular acceleration from top 20 deg going down:
g_20_down = cosd(20)*g
alpha_20_down =  g_20_down/max_length_from_center% rad/s^2


% finding max angular acceleration from horizontal going down:
alpha_0_down =  a_max_tip/max_length_from_center% rad/s^2

% alpha max based on worst scenario at the top:
alpha_max =  g_20_down/max_length_from_center% rad/s^2
```

The angular acceleration can not exid 29.2748$\frac{rad}{s^2}$ when the ball have a negativ acceleration horizontal going down. If so the ball will be in free fall and no contact.

The angular acceleration can not exid 27.5094$\frac{rad}{s^2}$ when the ball have a negativ acceleration at 20 deg at top going down. If so the ball will be in free fall and no contact.

Now we need to consider that we dont want the ball to leave the beam at the tip.

First plot the vertical acceleration $\pm 20°$:

```
angle = linspace(-20,20,400);
rad = deg2rad(angle);
angle_rad = (cos(rad));
vertica_acceleration = (alpha_max*max_length_from_center)./angle_rad;
plot(angle,vertica_acceleration)
grid on
xlabel('angel')
ylabel('vertical acceleration')
title('Tangential acceleration limitation down')
```

This means that if the ball stops at 20 then the accel is 9.21838 $\frac{m}{s^2}$ vertical

Finds the length of the arc it accelerates

```
arc_deg =linspace(0,2*pi/9,400); % Going from -20 deg to 20 deg.
arc_lengt_1 = max_length_from_center*arc_deg; % m lengt from -20 to 20 deg
```

Finding the speed at the end vertically.

```
theta = arc_deg ; % -20 to 20 deg
t = sqrt((2*theta)/alpha_max); % s, time from -20 deg to 20 deg.
omega_30 = alpha_max*t; % rot velocity from -20 to 20 deg over the arc lengt.
velocity_tang = omega_30*max_length_from_center; % m/s, tangential velosyty if alpha is const

plot(theta,velocity_tang)
grid on
xlabel('Lengt of travell rad')
ylabel('Velocity m/s')
title('Velocity ball at tip of beam')
hold off
```

Finding how high the ball will go if the speed and acceleration is known. Stops at 20 deg at top after travel -20 to 20 deg..

```
%h = 1/100 %m
% energi_potensiell = m*g*h
% from -20 deg  stops at 20deg how high and fare will the ball go.
% This is only to show if the system have limits unvanted senarios can
% happen.
H_max =((velocity_tang).^2.*(sind(70)).^2)/(2*g) *10^3 % mm in the air
R_max =((velocity_tang).^2.*(sind(2*70)))/(g) *10^3 % mm in the air
plot(velocity_tang,H_max)
hold on
plot(velocity_tang,R_max)
grid on
xlabel('velocity m/s')
ylabel('Ball will go up (H) and away(R) if stoped, mm')
legend('Ball can go vertically (H)','Ball can go horizontally (R)', 'Location','best')
title('How far can the ball fly if the Beam stops at 20^o')
```

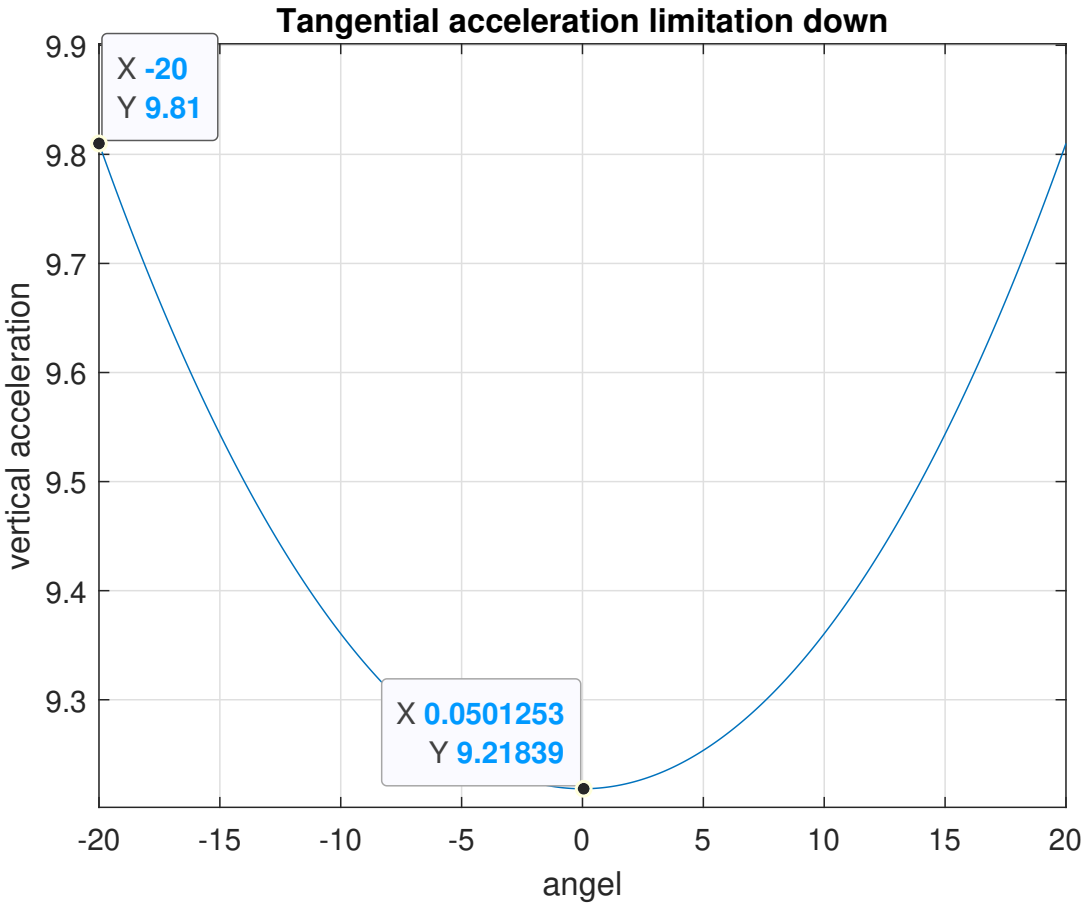

Figure F.11: Tangential acceleration limitation ball rotation down, $[m/s^2]$

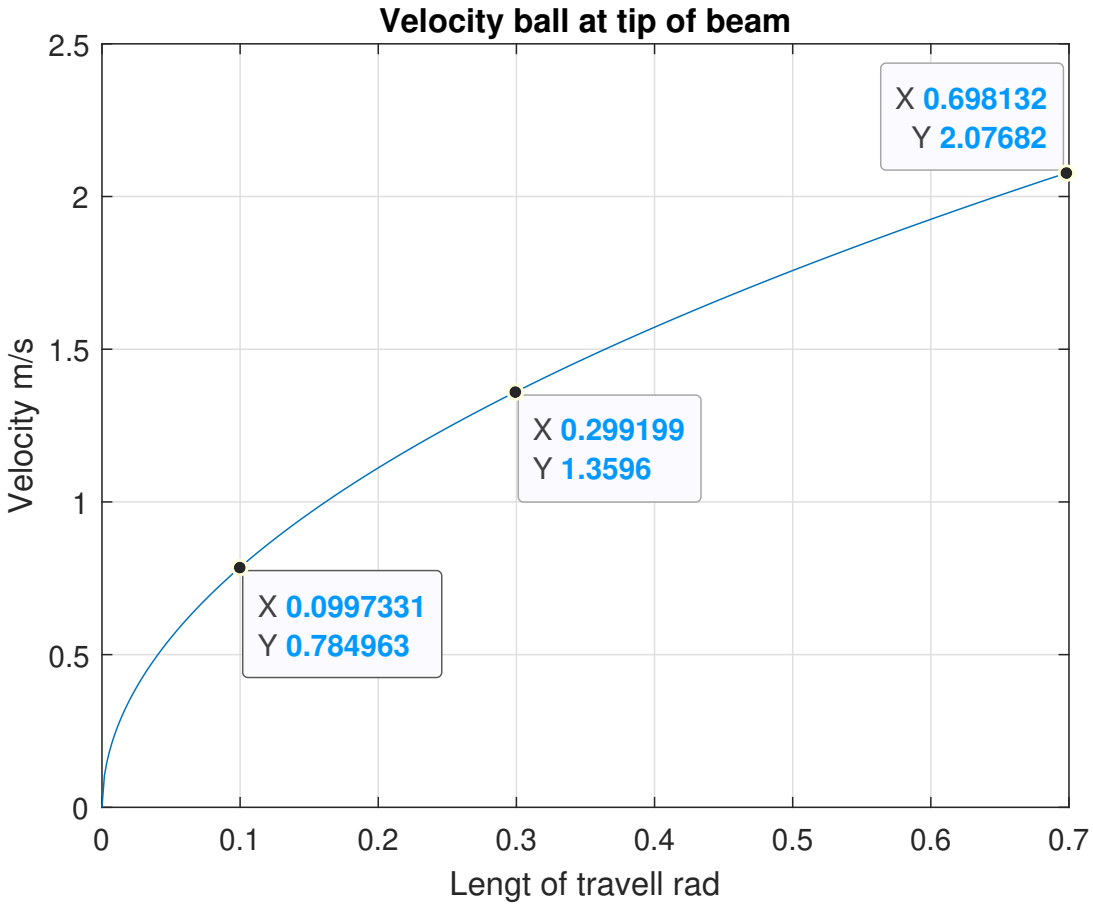

Figure F.12: Tangential velocity limitation ball rotation down, $[m/s]$

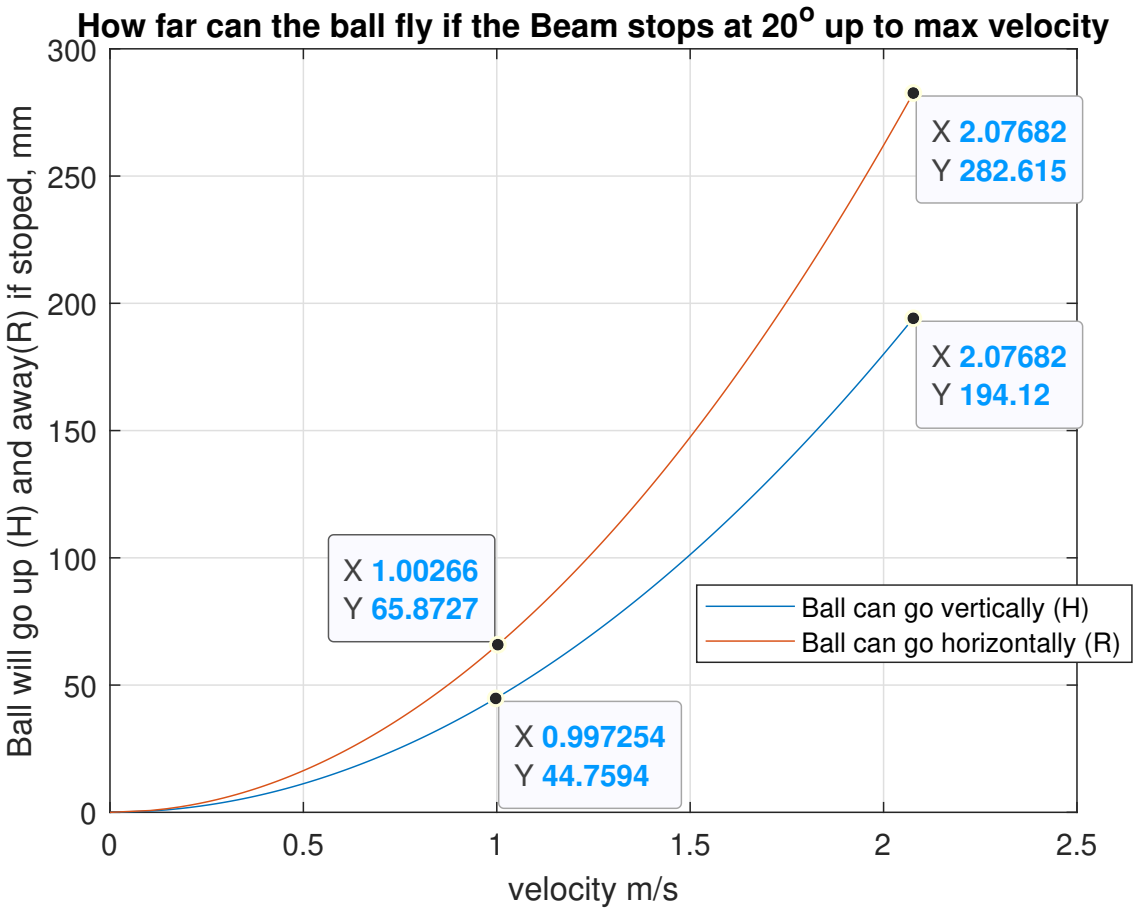

Figure F.13: Velocity ball can fly at a sudden stop

F.6 Centripetal acceleration

```
clc; close all; clear
% Roling ball and rotattation dynamics
%
% Testing omega, velocity, ball pos on beam is changing.
g = 9810; %[mm/s^2]
R_ball = 40.1*1/2; % mm
d_beam_17 = 16.667; % mm
V_relativ_mm = 1.209876597541056e+03 ; % mm/s

deg = linspace(-20,20,200); % deg, max min angel
ball_aks =(g*sind(deg))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4))); %  mm/s^2,
% acceleration based on angel
figure
plot(ball_aks,deg)
ylabel('Deg^o')
xlabel('Ball acceleration [mm/s^2]')
legend('Ball acceleration based on angle')
grid on



ball_aks_1deg =(g*sind(1))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4))) % mm/s^2,
% acceleration
ball_aks_5deg =(g*sind(5))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4))) % mm/s^2,
% acceleration
ball_aks_10deg =(g*sind(10))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4))) % mm/s^2,
% acceleration
ball_aks_15deg =(g*sind(15))/(1+(2*R_ball^2)/(3*(R_ball^2-d_beam_17^2/4))) % mm/s^2,
% acceleration




roling_length = linspace(0,335.1,200); % mm From center to tip.

r = 355.1; % radius mm
v = linspace(-1221.4,1221.4,200); % mm/s, tangential velocity at tip.
omega = v/r; % rad/s

figure
plot(omega,v)
grid on
xlabel('Omega [rad/s]')
ylabel('Angular velocity [m/s]')
legend('Rotation vel at tip and omega')


a_N_1221 = (V_relativ_mm/r).^2.*roling_length; % mm/^2,
% centripetal acceleration v=1221.4 [mm/s^2] (max)
a_N_1000 = (1000/r).^2.*roling_length; % mm/^2,
% centripetal acceleration v=1000 [mm/s^2]
a_N_500 = (500/r).^2.*roling_length; % mm/^2,
```

```
% centripetal acceleration v=500 [mm/s^2]
a_N_100 = (100/r).^2.*roling_length; % mm/^2,
% centripetal acceleration v=100 [mm/s^2]

figure
plot(roling_length,a_N_1221)
grid on
hold on
plot(roling_length,a_N_1000)
hold on
plot(roling_length,a_N_500)
hold on
plot(roling_length,a_N_100)
ylabel('centripetal acceleration [mm/s^2]')
xlabel('Rolling from center to tip [mm]')
legend('1221.4mm/s^2','1000mm/s^2','500mm/s^2','100mm/s^2', 'Location','best')
title('Centripetal acceleration based on angular velocity and position on the beam')



% The beam rotates towards the ball:

accel_1_V_1221 = ball_aks_1deg  - a_N_1221;% exampel 1
% senario 1 deg and tangential velocity (max):
accel_5_V_1221  = ball_aks_5deg  - a_N_1221;% exampel 2
% senario 5 deg and tangential velocity (max):
accel_10_V_1221 = ball_aks_10deg - a_N_1221;% exampel 3
% senario 10 deg and tangential velocity (max):
accel_15_V_1221 = ball_aks_15deg - a_N_1221;% exampel 4
% senario 15 deg and tangential velocity (max):


figure
plot(roling_length,accel_1_V_1221)
hold on
grid on
plot(roling_length,accel_5_V_1221)
hold on
plot(roling_length,accel_10_V_1221)
hold on
plot(roling_length,accel_15_V_1221)
hold on
ylabel('Acceleration based on angle (gravity)')
xlabel('The position of the ball on the beam')
legend('Acceleration 1 deg','Acceleration 5 deg','Acceleration 10 deg'...
    , 'Acceleration 15 deg')
title('Acceleration Direction Possitiv rolls towards the center ...
    of the beam (1221.4 mm/s at tip)')



%%%%%%%%%%%%%%%%%%%%

accel_1_V_500  = ball_aks_1deg  - a_N_500;% exampel 1
```

```
% senario 1  deg and tangential velocity (500):
accel_5_V_500  = ball_aks_5deg  - a_N_500;% exampel 2
% senario 5  deg and tangential velocity (500):
accel_10_V_500 = ball_aks_10deg - a_N_500;% exampel 3
% senario 10 deg and tangential velocity (500):
accel_15_V_500 = ball_aks_15deg - a_N_500;% exampel 4
% senario 15 deg and tangential velocity (500):


figure
plot(roling_length,accel_1_V_500)
hold on
grid on
plot(roling_length,accel_5_V_500)
hold on
plot(roling_length,accel_10_V_500)
hold on
plot(roling_length,accel_15_V_500)
hold on
ylabel('Acceleration based on angle (gravity)')
xlabel('The position of the ball on the beam')
legend('Acceleration 1 deg','Acceleration 5 deg' ...
    ,'Acceleration 10 deg', 'Acceleration 15 deg')
title('Acceleration Direction Possitiv rolls towards ...
    the center of the beam (500 mm/s at tip)')

%
% The beam rotates away from the ball:

accel_1_V_1221  = ball_aks_1deg  + a_N_1221;% exampel 1
% senario 1 deg and tangential velocity (max):
accel_5_V_1221  = ball_aks_5deg  + a_N_1221;% exampel 2
% senario 5 deg and tangential velocity (max):
accel_10_V_1221 = ball_aks_10deg + a_N_1221;% exampel 3
% senario 10 deg and tangential velocity (max):
accel_15_V_1221 = ball_aks_15deg + a_N_1221;% exampel 4
% senario 15 deg and tangential velocity (max):


figure
plot(roling_length,accel_1_V_1221)
hold on
grid on
plot(roling_length,accel_5_V_1221)
hold on
plot(roling_length,accel_10_V_1221)
hold on
plot(roling_length,accel_15_V_1221)
hold on
ylabel('Acceleration based on angle (gravity)')
xlabel('The position of the ball on the beam')
legend('Acceleration 1 deg','Acceleration 5 deg' ...
    ,'Acceleration 10 deg', 'Acceleration 15 deg')
title('Acceleration Direction Possitiv rolls away from ...
```

```
    the center of the beam (1221.4 mm/s at tip)')



%%%%%%%%%%%%%%%%%%%

accel_1_V_500  = ball_aks_1deg  + a_N_500;% exampel 1
% senario 1  deg and tangential velocity (500):
accel_5_V_500  = ball_aks_5deg  + a_N_500;% exampel 2
% senario 5  deg and tangential velocity (500):
accel_10_V_500 = ball_aks_10deg + a_N_500;% exampel 3
% senario 10 deg and tangential velocity (500):
accel_15_V_500 = ball_aks_15deg + a_N_500;% exampel 4
% senario 15 deg and tangential velocity (500):


figure
plot(roling_length,accel_1_V_500)
hold on
grid on
plot(roling_length,accel_5_V_500)
hold on
plot(roling_length,accel_10_V_500)
hold on
plot(roling_length,accel_15_V_500)
hold on
ylabel('Acceleration based on angle (gravity)')
xlabel('The position of the ball on the beam')
legend('Acceleration 1 deg','Acceleration 5 deg' ...
    ,'Acceleration 10 deg', 'Acceleration 15 deg')
title('Acceleration Direction Possitiv rolls away ...
    from the center of the beam (500 mm/s at tip)')


ball_aks_1deg =

   94.8056


ball_aks_5deg =

  473.4506


ball_aks_10deg =

  943.2979


ball_aks_15deg =

   1.4060e+03
```

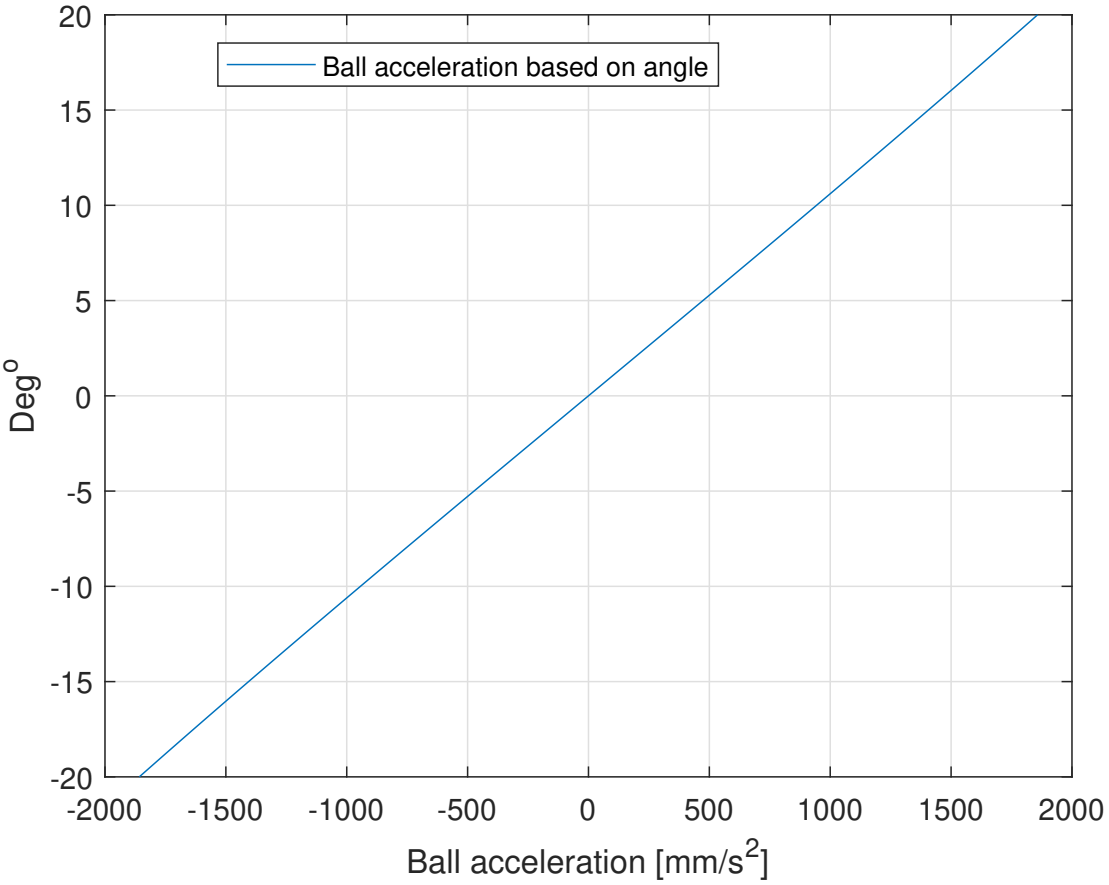

Figure F.14: Ball acceleration based on angle

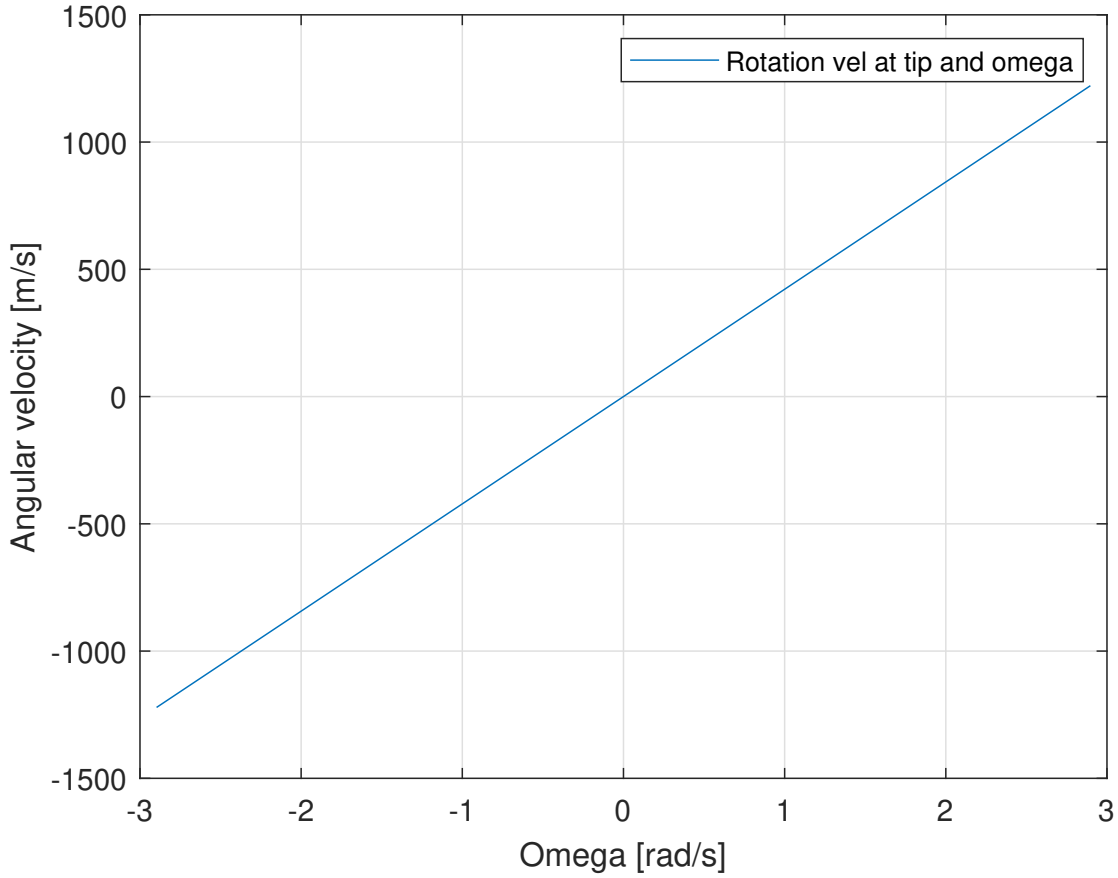

Figure F.15: Rotation vel at tip and omega

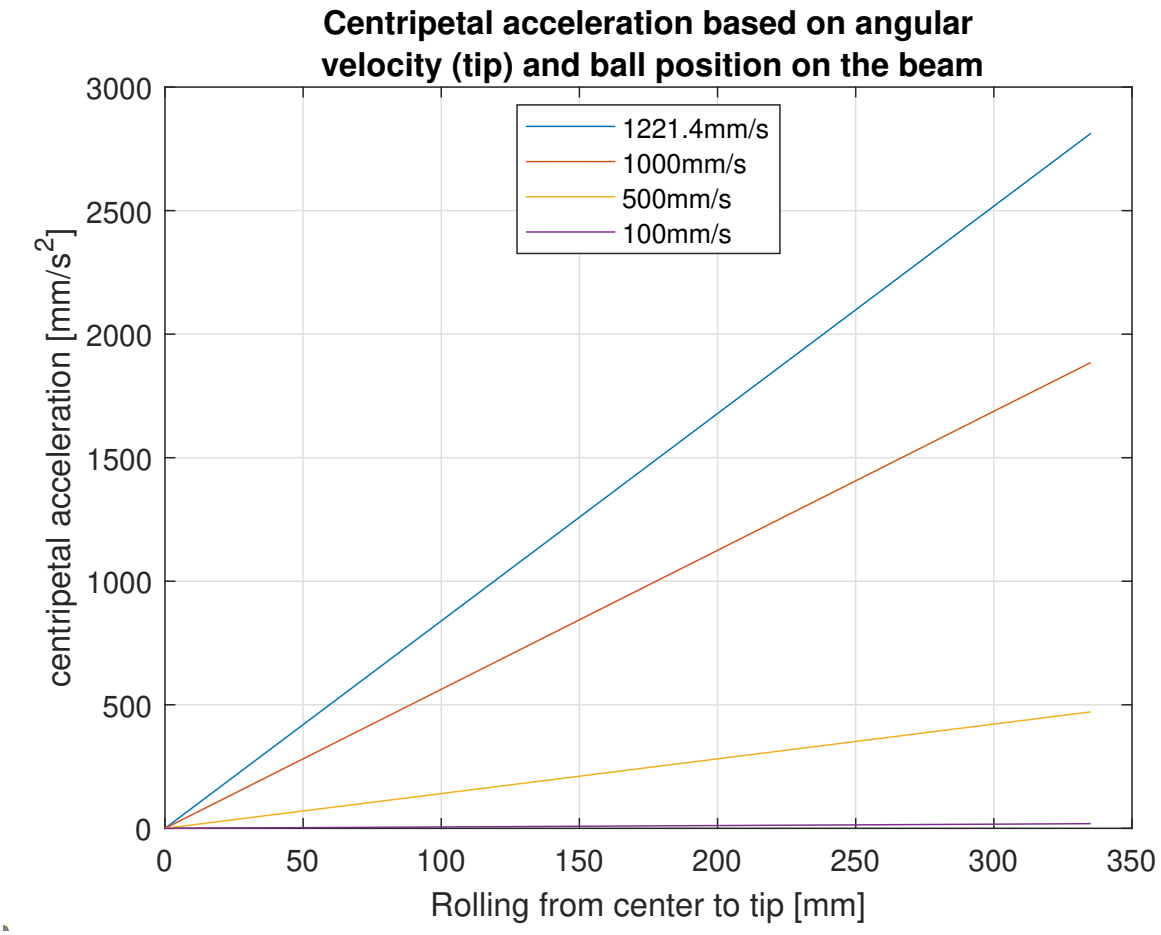

Figure F.16: Centripetal acceleration based on angular velocity and position on the beam

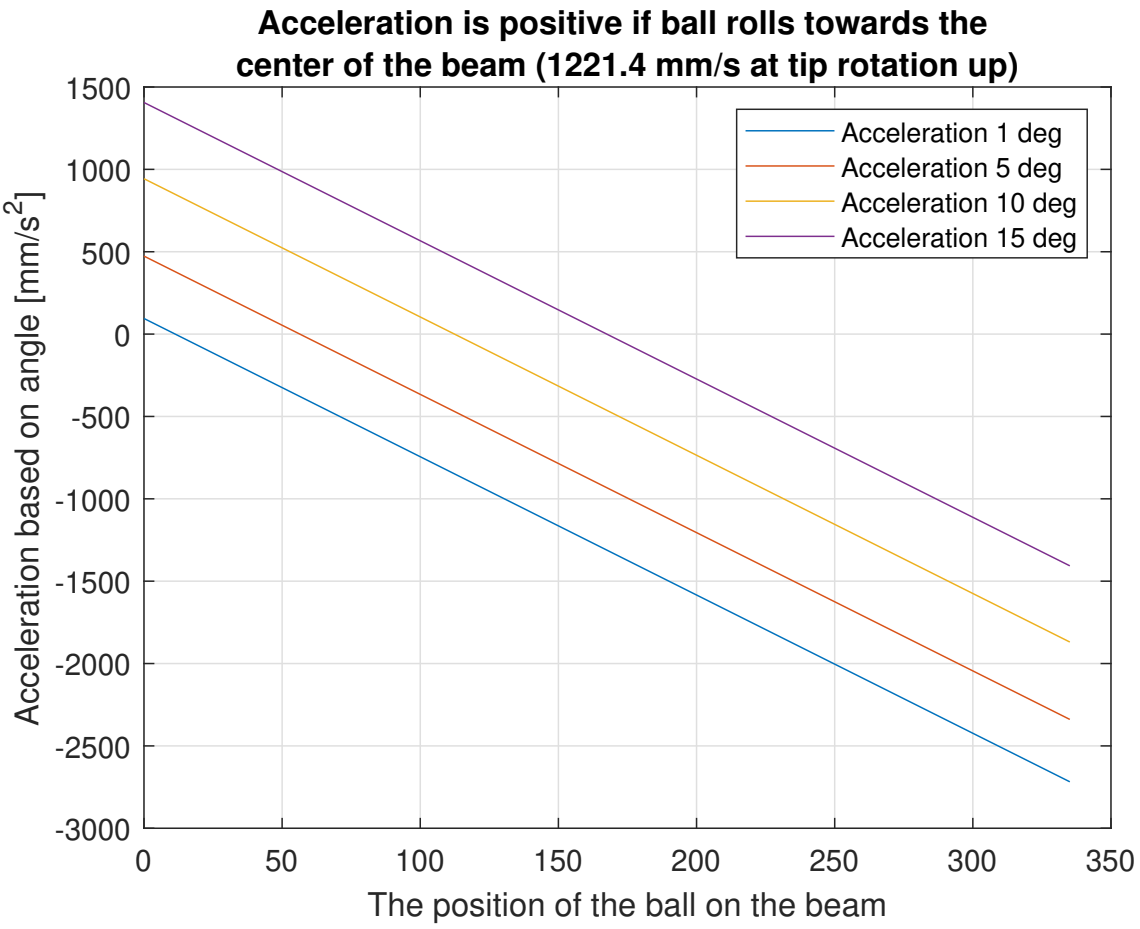

Figure F.17: Acceleration is positive if ball rolls towards the center of the beam (1221.4 mm/s at tip rotation up)

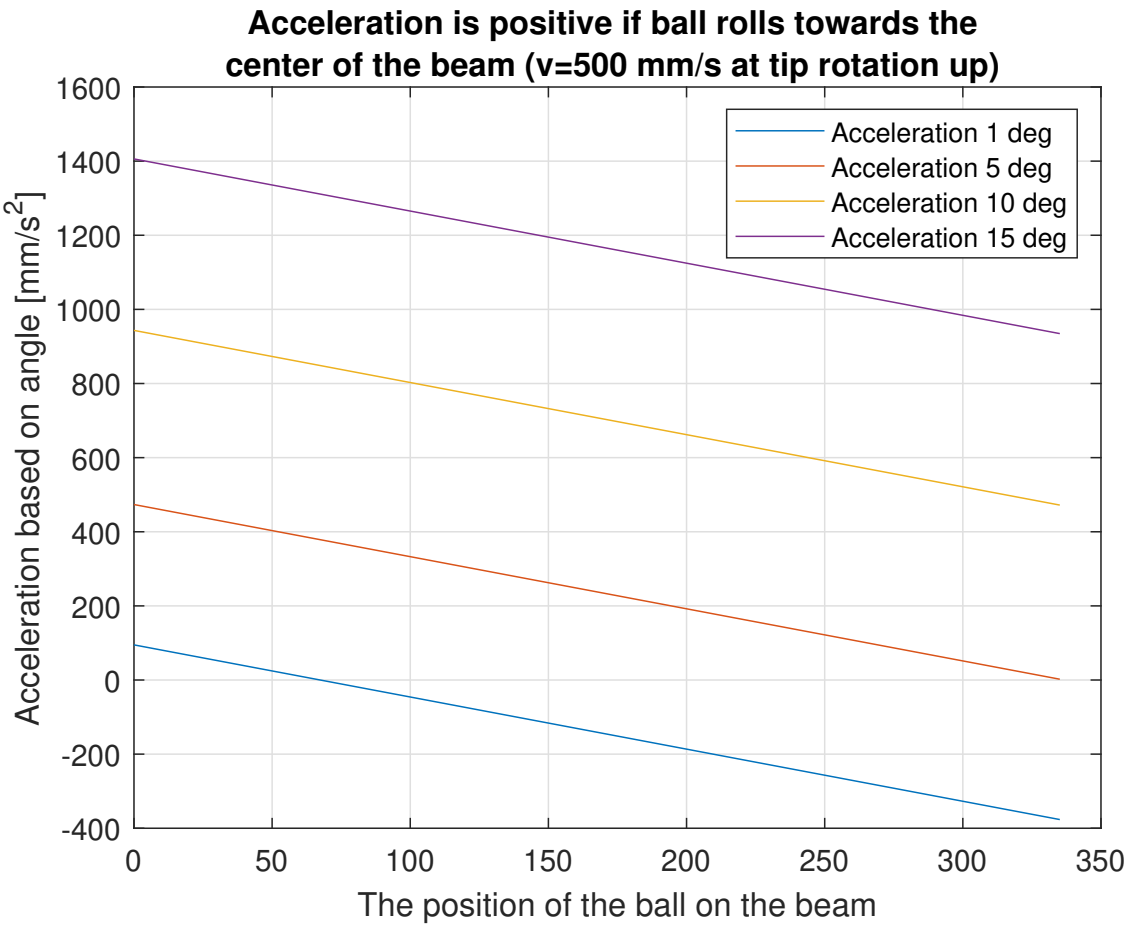

Figure F.18: Acceleration is positive if ball rolls towards the center of the beam (v=500 mm/s at tip rotation up)

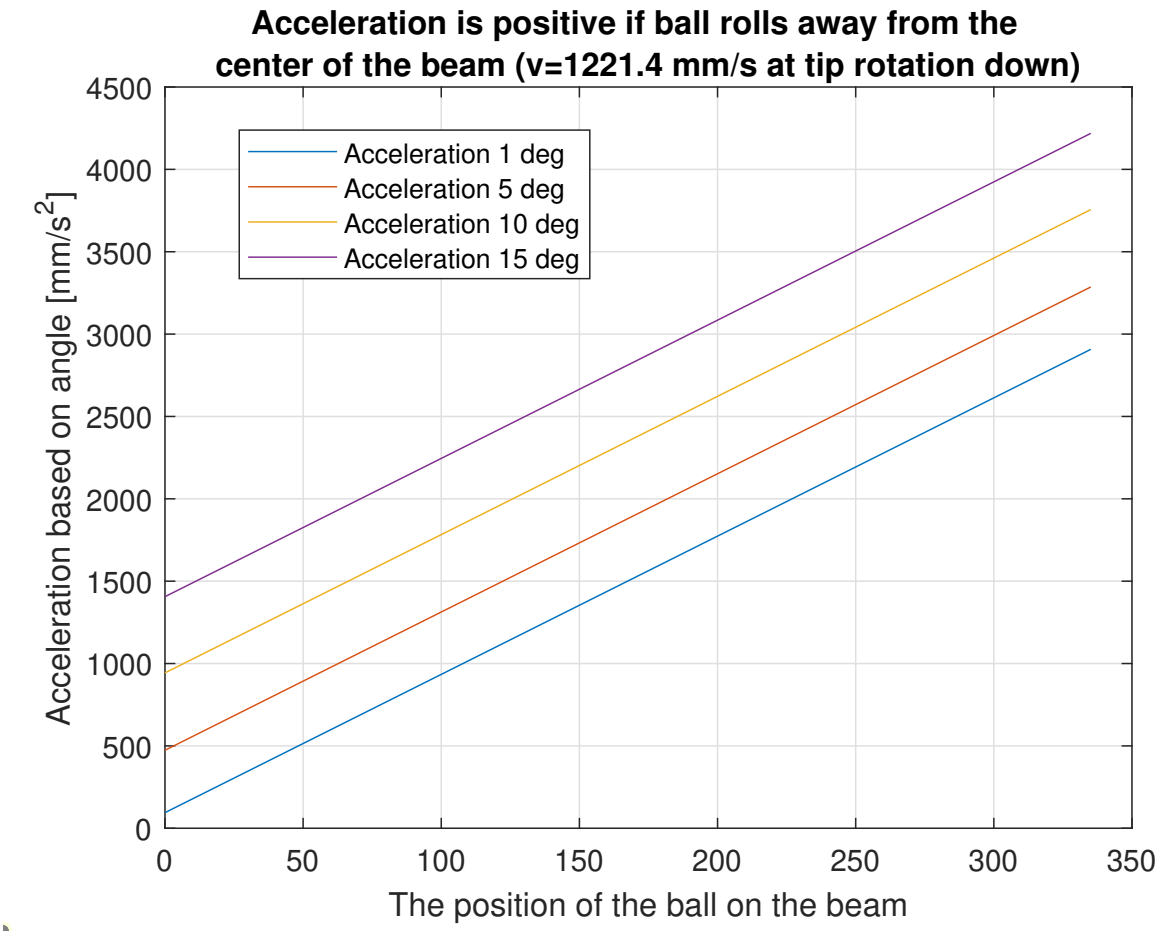

Figure F.19: Acceleration is positive if ball rolls away from the center of the beam (v=1221.4 mm/s at tip rotation down)

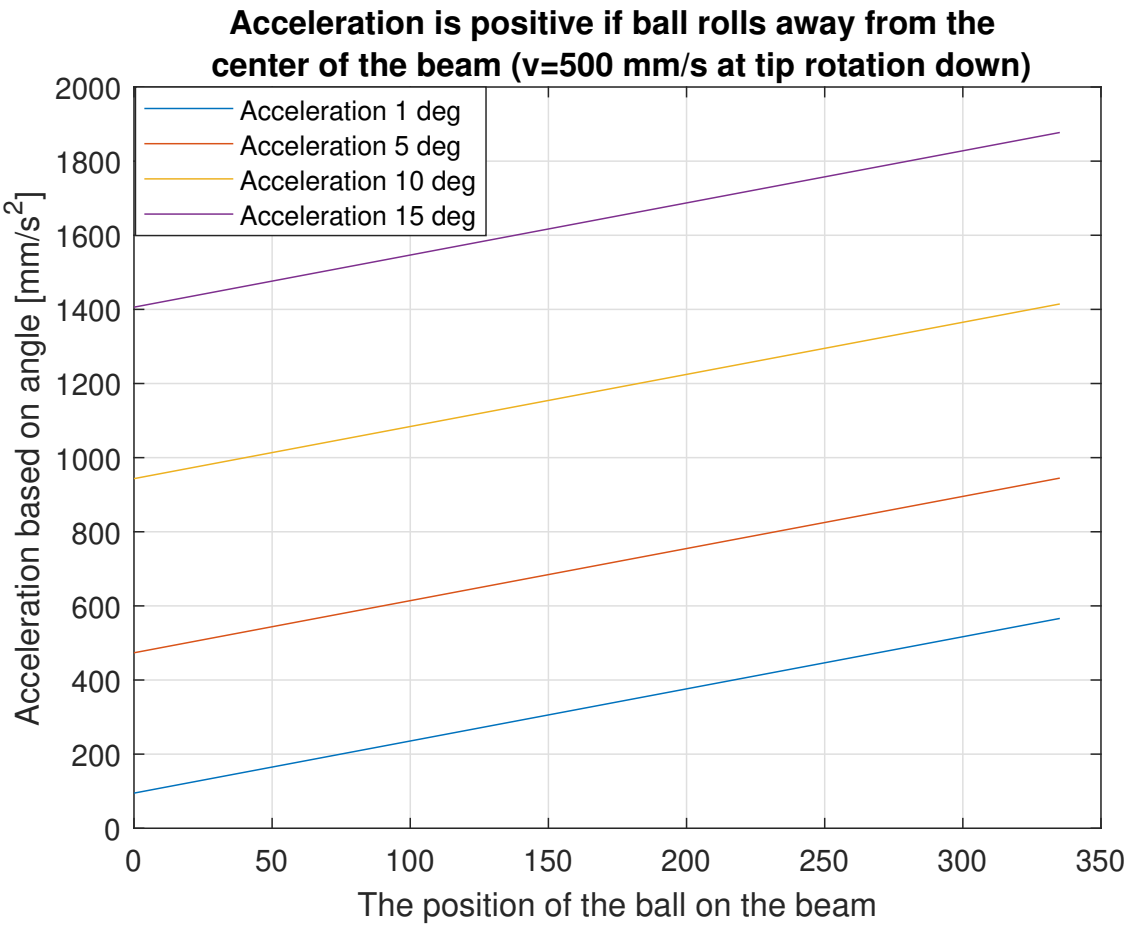

Figure F.20: Acceleration is positive if ball rolls away from the center of the beam (v=500 mm/s at tip rotation down)

Appendix G

Simscape

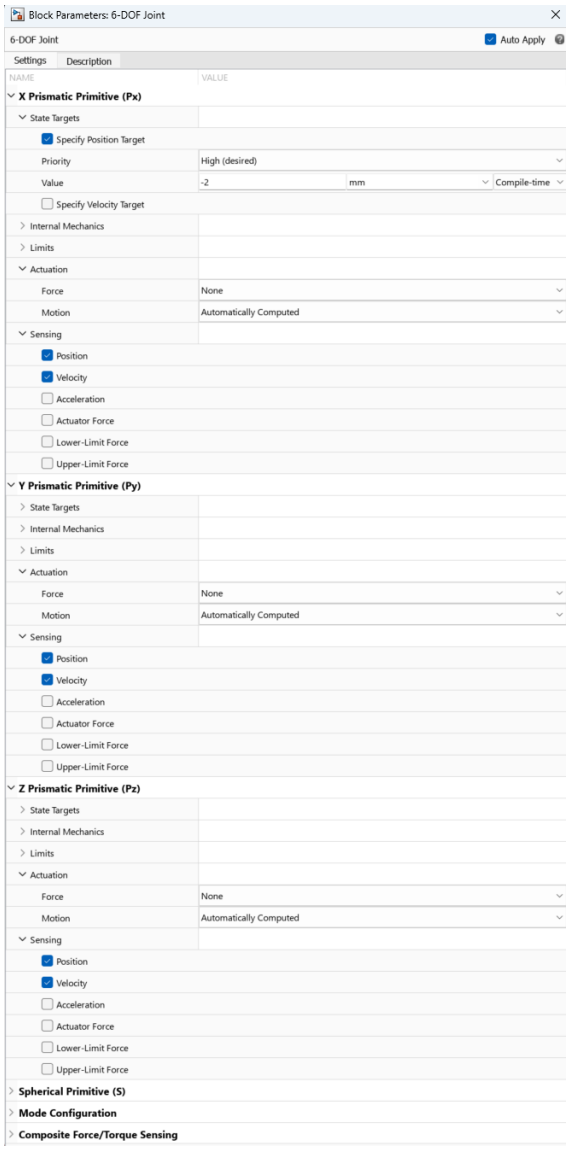

Figure G.1: 6-DOF joint

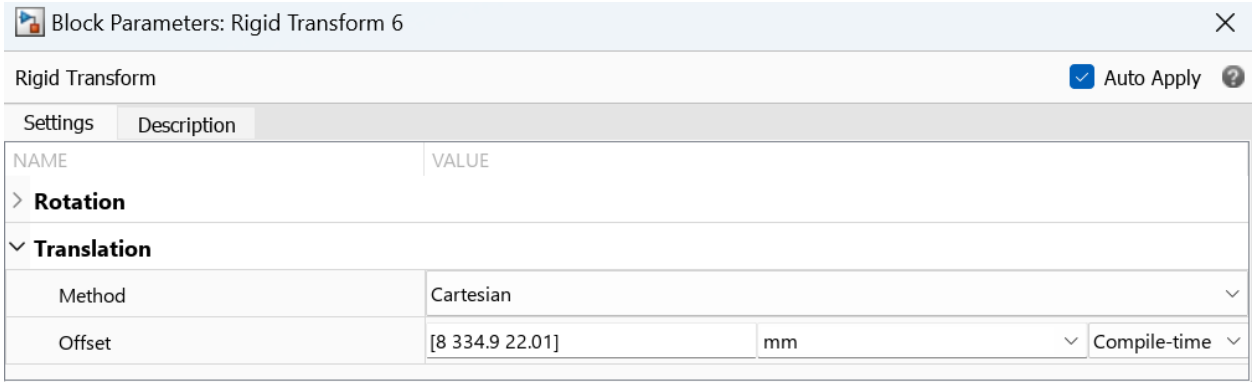

Figure G.2: Specifications for rigid transform 6

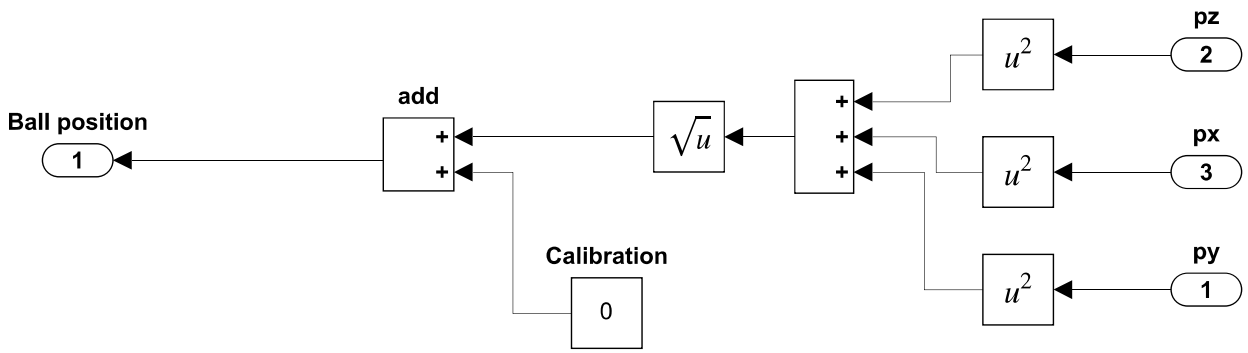

Figure G.3: Calculate position

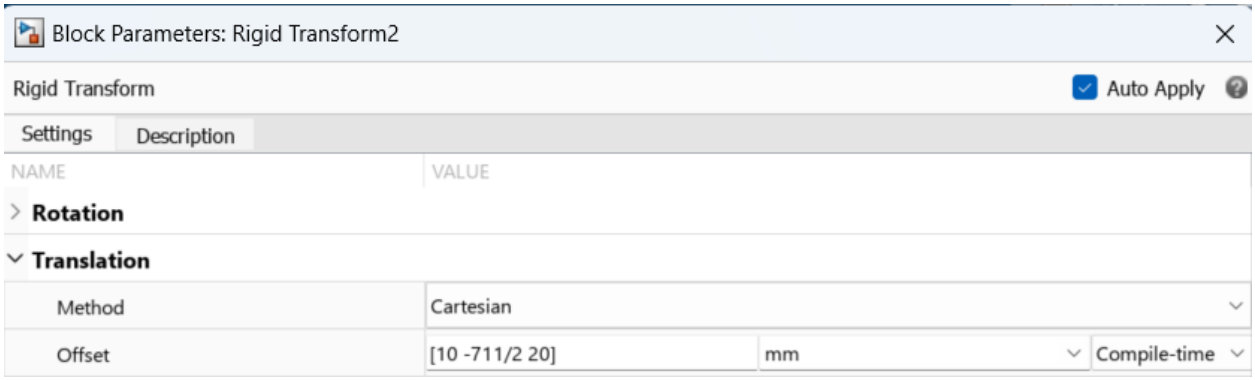

Figure G.4: End Wall 1

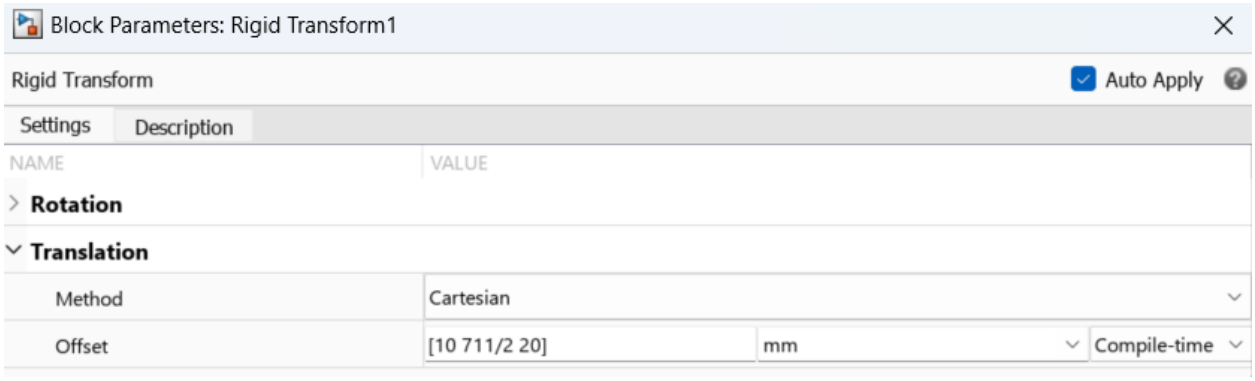

Figure G.5: End Wall 2

Bibliography

[1] Bosch Rexroth AG. *MS2N Synchronous Servomotors*. Tech. rep. 97803 Lohr a.Main, Germany, 2020. URL: https://docs.automation.boschrexroth.com/doc/3753406341/ctrlx-drive-motors-ms2s-synchronous-servomotors-project-planning-manual/latest/en/.

[2] Arduino. *Arduino MEGA 2560 Rev3, datasheet.pdf*. URL: https://docs.arduino.cc/hardware/mega-2560. 17.01.2023.

[3] *Balluff BOD 23K-LA01-S92*. URL: https://www.balluff.com/en-au/products/BOD001N. (accessed: 24.03.2023).

[4] *Bosch Rexroth AG*. URL: https://www.boschrexroth.com/en/dc/. (accessed: 09.05.2023).

[5] *carbonTub 8x 6x1000mm - Bronto*. URL: https://www.elefun.no/p/prod.aspx?v=25968. 11.02.2023.

[6] *COLLECTION OF HOW-TO VIDEOS, BLOGS AND EXAMPLES FOR CTRLX AUTOMATION*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Collection-of-How-to-videos-blogs-and-examples-for-ctrlX/ba-p/12343. (accessed: 11.04.2023).

[7] *CTRLX CORE - 3D VIEWER APP*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-CORE-3D-Viewer-App/ba-p/48336. (accessed: 11.04.2023).

[8] Dualsky. *DS8180, servo, digital MG, 56g, 10kg.cm@7.4V*. URL: http://shop.dualsky.com/ds8180-servo-digital-mg-56g-10kgcm74v_p0263.html. 17.01.2023.

[9] Eaton. *DILM910(230V50/60HZ) Contactor 3 pole*. URL: https://datasheet.eaton.com/datasheet.php?model=276698&locale=en_GB. (accessed: 14.05.2023).

[10] *General motor preselection Project planning/calculation, Bosch Rexrooth , R999000499/2019-03*. URL: https://www.boschrexroth.com/ics/cat/content/assets/Online/do/LT_LMS_CKX_Calculation_EN_20190726_095444.pdf. (accessed: 13.05.2023).

[11] Codesys Group. *Codesys.com*. URL: https://www.codesys.com. (accessed: 14.02.2023).

[12] WebIQ Smart HMI. *ctrlX WebIQ Design Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Smart-HMI-WebIQ-Designer/ba-p/18421. (accessed: 14.02.2023).

[13] WebIQ Smart HMI. *ctrlX WebIQ Server App Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Smart-HMI-WebIQ-Server/ba-p/18412. (accessed: 14.02.2023).

[14] WebIQ Smart HMI. *WebIQ The Web HMI SCADA Platform by SMART HMI*. URL: https://www.youtube.com/@webiqthewebhmiscadaplatfor5811/videos. (accessed: 14.02.2023).

[15] Chris Cook Jeff Moscrop and Faze Naghdy. *AN ANALYSIS OF MOTOR/LOAD INERTIA MISMATCH IN MACHINE TOOL SERVO SYSTEMS*. URL: https://www.sciencedirect.com/science/article/pii/S1474667017391498?ref=pdf_download&fr=RR-2&rr=7c6b35cd49a31c12. (accessed: 13.05.2023).

[16] Ibrahim Kaya, Nusret Tan, and Derek P. Atherton. "Improved cascade control structure for enhanced performance." In: *Journal of Process Control* 17.1 (2007), pp. 3–16. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2006.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0959152406001004.

[17] *M12-A Female cable connector*. URL: https://www.binder-usa.com/us-en/products/automation-technology/m12-a/99-1436-914-05-m12-a-female-cable-connector-5-80-100-mm-shieldable-screw-clamp-ip67-ul. (accessed: 24.03.2023).

[18] *Markforged X7*. URL: https://markforged.com/3d-printers/x7. (accessed: 21.03.2023).

[19] Matlab. *Enable Simscape Multibody Link Plugin in SolidWorks*. URL: https://se.mathworks.com/help/smlink/ref/linking-and-unlinking-simmechanics-link-software-with-solidworks.html. (accessed: 09.03.2023).

[20] Matlab. *Install the Simscape Multibody Link Plugin*. URL: https://se.mathworks.com/help/smlink/ug/installing-and-linking-simmechanics-link-software.html. (accessed: 09.03.2023).

[21] *Mean Well RS-25-5*. URL: https://www.meanwell-web.com/en-gb/ac-dc-single-output-enclosed-power-supply-output-rs--25--5. 17.01.2023.

[22] Mark Nagurka and Shuguang Huang. *A mass-spring-damper model of a bouncing ball*. URL: https://www.researchgate.net/publication/4118457_A_mass-spring-damper_model_of_a_bouncing_ball. (accessed: 24.04.2023).

[23] *Onyx(TM)*. URL: https://markforged.com/materials/plastics/onyx. (accessed: 21.03.2023).

[24] Morten Ottestad. *Lecture notes MAS246-G 22H Servoteknikk*. Tech. rep. 2022.

[25] PhoenixContact. *Power Supply Unit Primary Switched Narrow Design MINIPS4860DC/24DC/1*. URL: https://www.phoenixcontact.com/en-pc/products/dc-dc-converters-mini-ps-48-60dc-24dc-1-2866271. (accessed: 14.05.2023).

[26] *Project files Ball and Beam*. URL: https://github.com/fredryl/Files_for_bachelor_mechatronics_UiA. (accessed: 22.03.2023).

[27] RG Rajguru Electronics (I) Pvt.Ltd. *Arduino Mega 2560 R3 Sensor Shield V2.0.pdf*. URL: https://www.rajguruelectronics.com/Product/1352/Sensor%20shield%20for%20Arduino%20Mega.pdf. 17.01.2023.

[28] Rexroth. *ctrlX 3D viewer App Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-CORE-Motion-App/ba-p/13294. (accessed: 14.02.2023).

[29] Rexroth. *ctrlX DRIVE Drives systems*. URL: https://www.boschrexroth.com/en/us/search.html?q=R911386579&lang=EN&origin=header&s=download&num=10. (accessed: 04.03.2023).

[30] Rexroth. *ctrlX DRIVE Runtime AXSV03RS Functions Application Manual*. URL: https://docs.automation.boschrexroth.com/doc/1006866998/ctrlx-drive-runtime-axs-v-03rs-functions-application-manual/latest/en/. (accessed: 04.03.2023).

[31] Rexroth. *ctrlX Motion App Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-CORE-Motion-App/ba-p/13294. (accessed: 14.02.2023).

[32] Rexroth. *ctrlX OPC UA App Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-CORE-OPC-UA-Server-App/ba-p/13271. (accessed: 14.02.2023).

[33] Rexroth. *ctrlX PLC App Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-CORE-PLC-App/ba-p/13298. (accessed: 14.02.2023).

[34] Rexroth. *ctrlX Store*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/bg-p/dcdev_community-dev-blog/label-name/rex_c_Store. (accessed: 14.02.2023).

[35] Rexroth. *ctrlX Works Software*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/ctrlX-WORKS/ba-p/16448. (accessed: 14.02.2023).

[36] Rexroth. *DRIVE SIMULATION MODEL FOR MATLAB/SIMULINK*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Drive-simulation-model-for-MATLAB-Simulink/ba-p/55042. (accessed: 14.03.2023).

[37] Bosch Rexroth. *COLLECTION OF HOW-TO VIDEOS, BLOGS AND EXAMPLES FOR CTRLX AUTOMATION*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/Collection-of-How-to-videos-blogs-and-examples-for-ctrlX/ba-p/12343. (accessed: 14.02.2023).

[38] Bosch Rexroth. *ctrlX AUTOMATION Two Steps Ahead*. URL: https://apps.boschrexroth.com/microsites/ctrlx-automation/en/. (accessed: 14.05.2023).

[39] Bosch Rexroth. *ctrlX developR (Season 2) How to 5 Create a webbased HMI*. URL: https://www.youtube.com/watch?v=JPHyx5xX3-8. (accessed: 14.02.2023).

[40] Bosch Rexroth. *ctrlX DRIVE Drive Controllers, Supply Units*. URL: https://www.boschrexroth.com/en/us/search.html?q=R911392532&lang=EN&origin=header&num=10&s=download. (accessed: 03.03.2023).

[41] Bosch Rexroth. *ctrlX PLC Engineering PLC Programming System Application Manuals*. URL: https://docs.automation.boschrexroth.com/iirds/cdp-metadata.boschrexroth.de~iiDC~Product-ctrlX-PLC-Engineering/. (accessed: 03.03.2023).

[42] Bosch Rexroth. *ctrlX PLC Engineering, PLC Libraries, Reference Book*. URL: https://docs.automation.boschrexroth.com/iirds/cdp-metadata.boschrexroth.de~iiDC~Product-ctrlX-PLC-Engineering/. (accessed: 03.03.2023).

[43] Bosch Rexroth. *MS2S Synchronous Servomotors Operating Manual*. URL: https://docs.automation.boschrexroth.com/iirds/cdp-metadata.boschrexroth.de~iiDC~Product-MS2S/. (accessed: 14.05.2023).

[44] Bosch Rexroth. *WEBIQ - CREATE YOUR FIRST HMI PROJECT*. URL: https://developer.community.boschrexroth.com/t5/Store-and-How-to/WebIQ-Create-your-first-HMI-project/ba-p/51180. (accessed: 14.02.2023).

[45] *Robots and robotic devices - Collaborative robots*. ISO 15066-2016. 12.01.2023.

[46] RS. *RS PRO DIN Rail Power*. URL: https://docs.rs-online.com/ca24/A700000007350010.pdf. (accessed: 14.05.2023).

[47] *SICK OD2000*. URL: https://cdn.sick.com/media/pdf/5/35/935/dataSheet_OD2000-3502T15_6074384_en.pdf. (accessed: 13.05.2023).

[48] SMARTHMI. *WebIQ Manual-WebIQ Designer*. URL: https://www.smart-hmi.de/user/download/deliver/docs/documentation-manual-webiq-designer-2.13-e58c/index.html#recorder-manager. (accessed: 04.04.2023).

[49] *SolidWorks*. URL: https://www.solidworks.com. (accessed: 09.05.2023).

[50] STMicroelectronics. *Time-of-Flight ranging sensor*. URL: https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html#documentation. 17.01.2023.

[51] Shreyas Sundaram. *ECE 380 Control systems*. URL: https://engineering.purdue.edu/~sundara2/misc/ece380_notes.pdf. (accessed: 14.01.2023).

[52] Prof. Dawn Tilbury and Prof. Bill Messner. *Ball and Beam: Simscape Modeling*. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SimulinkSimscape. (accessed: 01.05.2023).

[53] *Tracker-5.1.5*. URL: https://physlets.org/tracker/. 06.04.2021.

[54] UIA. *Laboratorier*. URL: https://www.uia.no/om-uia/fakultet/fakultet-for-teknologi-og-realfag/institutter2/institutt-for-ingenioervitenskap/laboratorier. (accessed: 04.03.2023).

[55] *UltiMaker Cura 5.2.1*. URL: https://ultimaker.com/software/ultimaker-cura. 19.10.2022.

[56] The Open University. *2.7 Chebyshev and Butterworth filters*. URL: https://www.open.edu/openlearn/science-maths-technology/electronic-applications/content-section-2.7. (accessed: 14.05.2023).

[57] *White Paper Evolving the Rules of Inertia Matching, Kollmorgen*. URL: https://www.motiontech.com.au/wp-content/uploads/2021/11/Kollmorgen-White-Paper-Evolving-the-Rules-of-Inertia-Matching.pdf. (accessed: 13.05.2023).

[58] Jing Zhou. *Block Diagram Reduction and Steady-State Errors*. URL: https://uia.instructure.com/courses/10505/files/1759448?module_item_id=367344. (accessed: 04.04.2023).