Laboratory Report 3: DC Motor with Resonant Load

Lum Borovci Michele Zarantonello Federico Bergami Riccardo Grosso
Group: 11, Shift: 1

October 19, 2025

1 Introduction

The purpose of LAB3 is to design and implement several control techniques in the DC Motor Setup with a Hub & Beam (also known as a resonant load) modification. For this setup, a collocated version of the plant is considered, which assumes the plant is a rigid body with no elastic transmission. Desired Control Performance is then achieved using PID and State-Space Methods, namely Eigenvalue Allocation using Static State Feedback and Linear Quadratic Regulation, for both Nominal and Robust Tracking Conditions.

For a more complete representation, a non-collocated version of the plant includes this transmission property but needs to be treated with other control techniques such as Gain and Phase Stabilization. Moreover, an estimation of two constituent plant parameters is done via Linear Regression for a more accurate description of the plant beyond the given nominal values.

1.1 Activity Goal

The goal of this laboratory session is to design a series of controllers that provide a fast response and a small steady state error for tracking step position references of the resonant load angular position. The measure for the required response speed is given by the settling time requirement $t_{s,5\%} \leqslant 0.85\,s$ and overshoot $M_p \leqslant 30\%$, whereas the settling time is assessed qualitatively based on the model and tracking (Nominal or Robust) conditions, depending on the control design. The Step Reference we test all the designs with is the angular hub position $\vartheta_{h,ref} = 50^\circ$, unless otherwise stated.

The DC Motor with a Beam & Hub load setup is a representative model for second-order systems of mechanical transmission, and this Lab activity showed us the approach, steps and assessment of results for the control design for such systems, with the latter phase done only the collocated version of the plant dynamics, treating the more general non-collocated case only at an introductory level. The performance requirements to be met in all designs are stated below:

- Perfect steady-state tracking of constant set points
- Step response (hub side) with settling time $t_{s,5\%} \leqslant 0.85\,s$ and overshoot $M_p \leqslant 30\%$

There are two main implementations of the Plant and its corresponding controllers. First, the frequency-domain implementation, where $P(s)$ and $C(s)$ are transfer functions, and on which only the PID design is implemented (see Schematic in Figure 28 of the Appendix). Second, the State-Space implementation which is used for all the other designs, is shown in Figure 32 of the Appendix.

As previously mentioned, all design implementations are made on either Nominal or Robust Tracking conditions. Their respective Simulink implementations are displayed in Figures 33 and 34 of the Appendix, and their control laws for the State Space implementation do not change in structure as the designs vary. The only change is the dimensions of the constituent gains and matrices specific to the design method being implemented. The control law for the PID implementation is displayed in Figure 31 of the Appendix.

1.2 System and Model

The system consists of a DC Motor controlled by means of a voltage driver. The motor shaft drives a planetary gearbox that in turn transfers the rotary motion to a hub mounted on top of the whole assembly with an output shaft. A beam is mounted to the hub through a circular joint (with axis aligned to the output shaft) and attached with a pair of springs that maintain the angular alignment of the beam with the hub (in static conditions). The two components are shown separately in Figure 1:

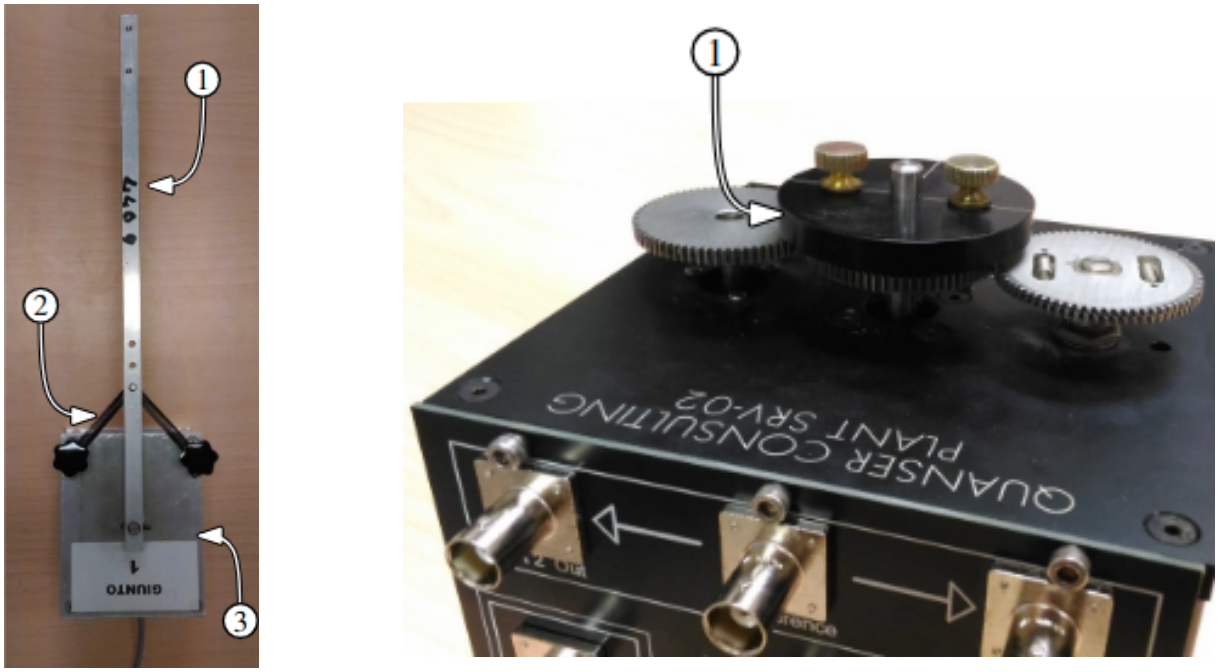


Figure 1. Main components of the setup - Oscillating beam component (left) and rotary hub (right). The beam hub is to be mounted in place of the black rotary hub (n°1, right).

Two sensors collect data for the system output. A rotary encoder connected to the motor shaft measures the motor's position and consequently the position of the hub θ_h (rigidly connected to the motor). A potentiometer mounted between the hub and the beam provides us with a voltage that can be converted to an angle measuring the displacement of the beam with respect to the hub θ_d. The sensors are shown in Figure 2. Together, the two measurements let us calculate the absolute position of the beam θ_b.

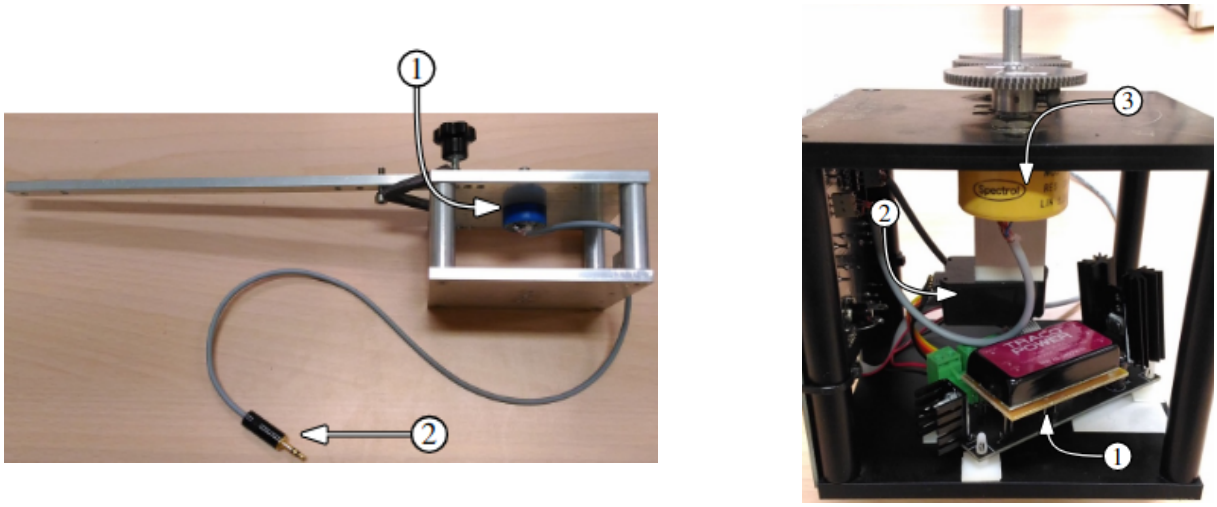


Figure 2. View of the sensors in the setup - Potentiometer (n°1, left) and rotary encoder (n°2, right).

The lumped-element configuration of the system is then cast as an LTI system of equations, numbered at (19) in the Methods Paper for LAB3. With some approximations, the following transfer function is obtained:

$$P_{u\to\vartheta_b}(s) \;=\; \frac{\vartheta_b(s)}{u(s)} \;=\; \frac{1}{Ns}\,\frac{k_{drv}\,k_t\,k}{R_{eq}\,D'_\tau(s) \;+\; k_t k_e(J_b s^2 + B_b s + k)} \tag{1}$$

where

$$D'_\tau(s) = J_{eq}J_b s^3 + (J_{eq}B_b + J_b B_{eq})s^2 + \left[B_{eq}B_b + k\left(J_{eq} + \frac{J_b}{N^2}\right)\right]s + k\left(B_{eq} + \frac{B_b}{N^2}\right) \tag{2}$$

This equation, that will be referred to as $P(s)$ from now on, represents the relation between a voltage u applied to the motor and the angular position of the beam ϑ_b. The parameters are explained in Appendix 11.

The Accurate Model of the system is implemented using a MATLAB + Simulink environment, and for the **Lab Environment Model**, the Simulink Real-Time Desktop Toolbox, which enables real-time interfacing with physical systems, is additionally used.

In an effort to streamline the testing and validation process, upon acquiring a working model for the controller designs in this - and all of the other - laboratories, an extra step was taken in the creation of a **Master Script** to automate the entire process of testing. From setting up the plant parameters, to tuning the controller, and finally plotting and displaying the results, this Master Script runs everything in one click, and each significant laboratory task has its own Master Script. A GitHub Repository which includes a readme file for the usage of these scripts is accessiblein this link:
`https://github.com/lumborovci/CELAB_2025`.

2 Tasks, Methodologies and Results

2.1 Assignment #1: Parameter Estimation

Following the procedure detailed in Section 5 of the LAB3 Methods Paper, the Beam Viscous Friction $\hat{B}_b$ and the elastic Hub-Beam joint stiffness $\hat{k}$ are estimated to create a plant representation specific to our motor.

2.1.1 Accurate Model Estimation

At first, we ran the estimation procedure on the data acquired by the free evolution of a second-order ODE for the beam displacement angle $\vartheta_d = \vartheta_b - \vartheta_h$ given by the following equation:

$$J_b \ddot{\vartheta}_d + B_b \dot{\vartheta}_d + k\vartheta_d = 0 \tag{3}$$

With the assumptions that the hub displacement angle ϑ_h remains fixed by having that $\omega_h = 0$, and no generic torque disturbances are present by imposing $\tau_{d,b} = 0$. This ODE admits an attenuating oscillatory solution when the system is underdamped, i.e. $\delta < 1$:

$$\vartheta_d(t) = Ae^{\sigma t}\cos(\omega t + \phi) \tag{4}$$

The procedure essentially records the absolute-valued peaks of this oscillation at points $\{t_k, \vartheta_d(t_k)\}$ and runs Linear Regression on the log-scaled version of this data (also known as the Logarithmic Decrement Regression) at points $\{t_k, \log[\vartheta_d(t_k)]\}$. The obtained slope $\hat{a}$ and intercept $\hat{b}$ of the regression line allow us to obtain several parameters of the response dynamics in a plug-in fashion. Starting from the intercept, we obtain the damping factors $\hat{\xi}$ and $\hat{\delta}$, and continue to get the damped and natural frequencies $\hat{\omega}$ and $\hat{\omega}_n$. Note that (81) and (83) are the equation numbers reported in the LAB3 Methods Paper:

$$\hat{\xi} = \hat{a} \xrightarrow{(71)} \hat{\delta} = \frac{\hat{\xi}}{\sqrt{\pi^2 + \hat{\xi}^2}} \xrightarrow{(81)} \hat{\omega} = \frac{1}{M}\sum_{k=0}^{M-1}\frac{\pi}{T_k}; \quad (T_k = t_{k+1} - t_k) \xrightarrow{(83)} \hat{\omega}_n = \frac{\hat{\omega}}{\sqrt{1-\hat{\delta}^2}} \tag{5}$$

And finally, we get the Beam Viscous Friction $\hat{B}_b$ and the elastic Hub-Beam joint stiffness $\hat{k}$ via:

$$\boxed{\hat{B}_b = 2\,J_b\,\hat{\delta}\,\hat{\omega}_n} \qquad \boxed{\hat{k} = J_b\,\hat{\omega}_n^2} \tag{6}$$

Running the Procedure on the Accurate model of the ODE produces the estimates outlined in the left-hand column of Table 1 of Section 2.1.1. To qualitatively show the accuracy of these estimates, we fit an estimate of the damping envelope $Ae^{\hat{\sigma}t}$ where $\hat{\sigma} = -\hat{\delta}\hat{\omega}_n$ on the response signal. The alignment of the envelope with the signal is shown in Figure 3:

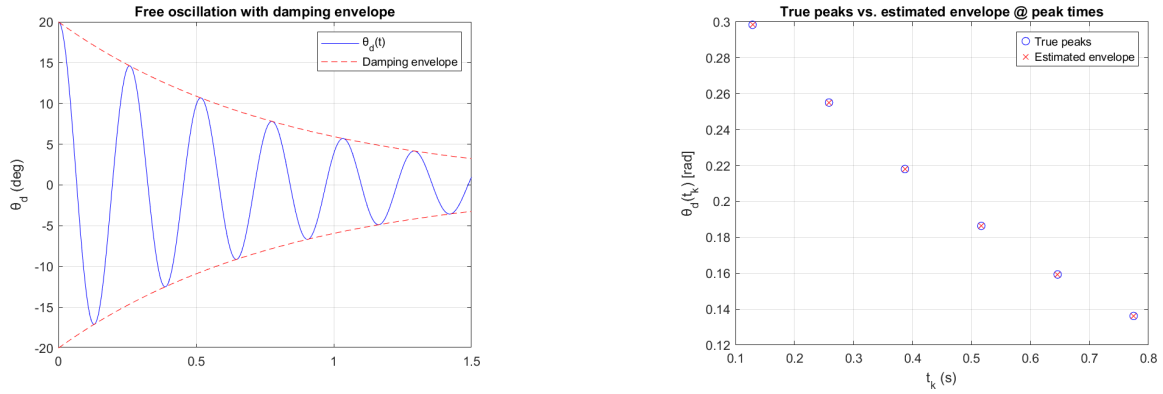


Figure 3. Accurate Model - Estimated Envelope fit (left) and zoomed-in peak-to-envelope tangent points (right)

The estimation procedure built around the Accurate Model will provide different values compared to those estimated in the Lab Environment with the real motor and resonant load, but still a trial on the Accurate Model serves as a 'sanity check' on the overall Parameter Estimation Assignment, providing us with an idea of the results we should expect in the laboratory environment.

2.1.2 Lab Environment Estimation

After recording the beam oscillations in the lab environment while keeping the hub fixed, we adapt the same estimation procedure after doing some necessary pre-processing of the acquired data. The pre-processing generates a smoothed-out version of the raw signal data by removing part of the noise through a rolling average. A larger window size of the averaging function made for a smoother signal, but with the tradeoff of a signal amplitude smaller than the recorded one. Results of the pre-processing are displayed in Figure 4.

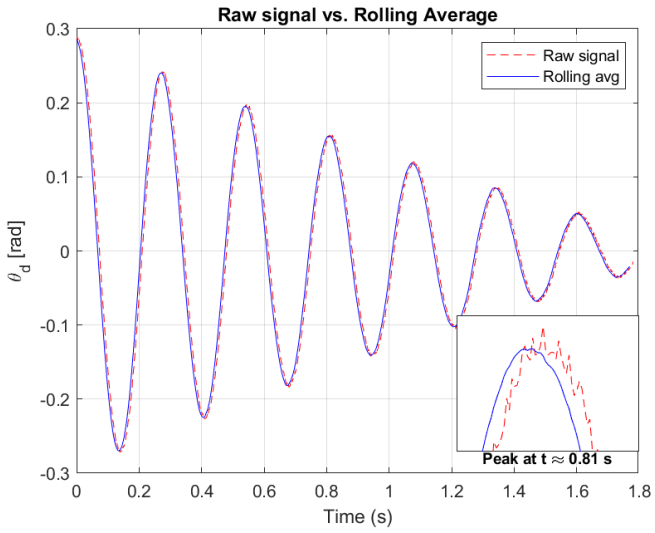


Figure 4. Raw hub–beam deflection signal and its rolling-average reconstruction. The inset image zooms on the peak at $t \approx 0.81$ s, highlighting the noise reduction achieved by the rolling average.

The aforementioned smoothness-to-value tradeoff of the reconstructed signal will have apparent effects when we carry out the envelope fitting as we did in Section 2.1.1 for the Accurate Model.

Having now obtained a reconstructed signal of the model evolution, we obtain the parameters of interest in similar fashion using log-scale Linear Regression. The results are outlined in the right-hand

column of Table 1, and the estimated envelope fit is displayed in Figure 5.

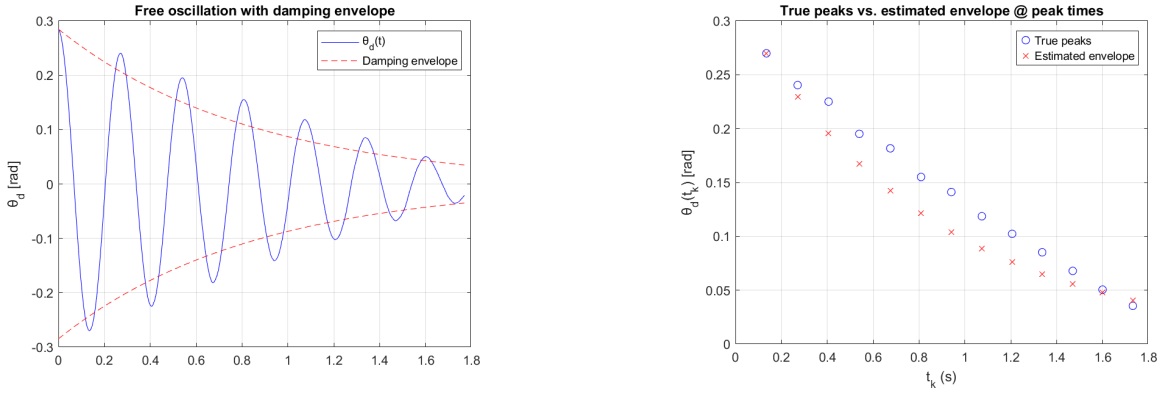


Figure 5. Lab Environment - Estimated Envelope fit (left) and zoomed-in peak-to-envelope tangent points (right). Notice the underestimation of the signal as a result of signal reconstruction and measurement imperfections.

In the following table we display the results for both models on which the estimation procedure was carried on.

Parameter	Accurate Model	Lab Environment Model
$\hat{\delta}$	4.99×10^{-2}	5.01×10^{-2}
$\hat{\omega}_d$	2.432×10^{1}	2.361×10^{1}
$\hat{\omega}_n$	2.435×10^{1}	2.364×10^{1}
$\hat{B}_b$	$\mathbf{3.40 \times 10^{-3}}$	$\mathbf{3.32 \times 10^{-3}}$
$\hat{k}$	$\mathbf{8.30 \times 10^{-1}}$	$\mathbf{7.82 \times 10^{-1}}$

Table 1: Estimated parameters comparison between the Accurate Model and the Lab Environment Model.

2.2 Assignment #2: Position PID Control

The first task following the estimation of the parameters of the model was to design a PID controller able to track a step reference for the angular position of the beam ϑ_b. The PID Controller equation is:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \tag{7}$$

Where $u(t)$ is the controller's output, $e(t)$ is the reference to output error, and K_P, K_I, K_D are the gains that scale the proportional, integral and derivative actions.

2.2.1 PID gains computation

To compute suitable gains for our control problem followed the standard **Bode Method**, which requires the plant's transfer function $P(s)$ and the dynamic specifications ($t_{s,5\%}$, M_p) to compute gain and phase margins between the current closed-loop transfer function and the desired one, to finally select the appropriate gains.
From the transient specifications specified in the Introduction 1.1 we calculated the damping factor δ and the crossover frequency ω_{gc}, then determined K_P, T_D,T_I, and subsequently K_I and K_D. The parameters used to compute the gains are:

$$\alpha = 4a, \qquad \frac{1}{T_L} = 10\,\omega_{gc}, \qquad T_W = \frac{t_{s5\%}}{5} \tag{8}$$

To achieve satisfactory responses (shown in the next subsection) we had to tighten our dynamic requirements to enforce a better response, corresponding to $M_p = 24\%$ and $t_{s5\%} = 0.55s$, thus obtaining $\omega_{gc} = 13.2\,rad/s$ and $\delta = 0.4136$ and the following gains for the PID:

$$K_P = 21.33, \qquad K_I = 85.08, \qquad K_D = 1.337$$

The script calculating these gains is reported in Section A.1.1 of the Appendix.

2.2.2 Model Comparison: Position PID Control Results

In this section the results obtained by applying a step reference to the Accurate Model, Black Box and real system are reported. Each graph compares the plots of the step reference and the output with Anti Windup enabled and disabled. The step time was delayed by $0.7\,s$, necessary to compute and correct the bias of the potentiometer that allows for the reading of θ_d in the lab environment.

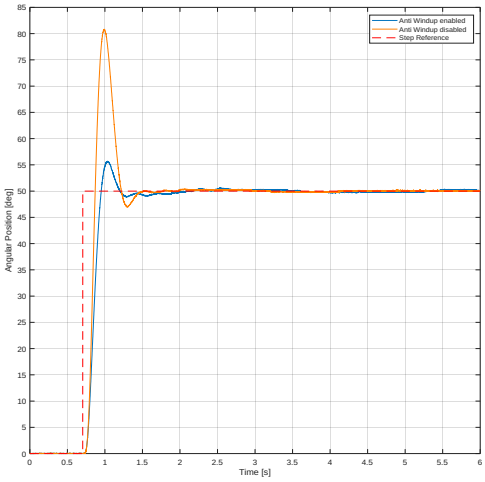


Figure 6. Accurate model response to a 50° step reference

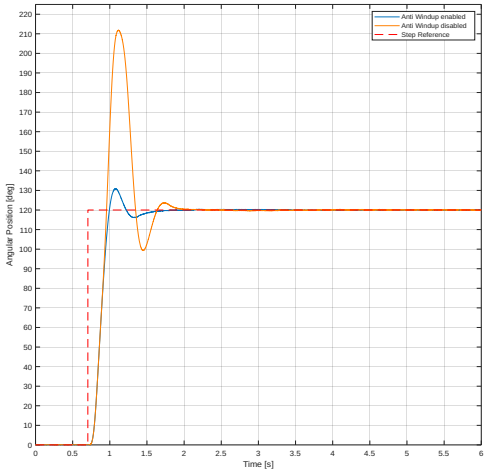


Figure 7. Accurate model response to a 120° step reference

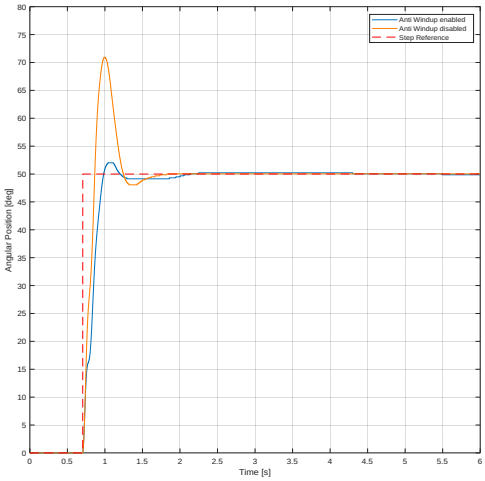


Figure 8. Black Box response to a 50° step reference

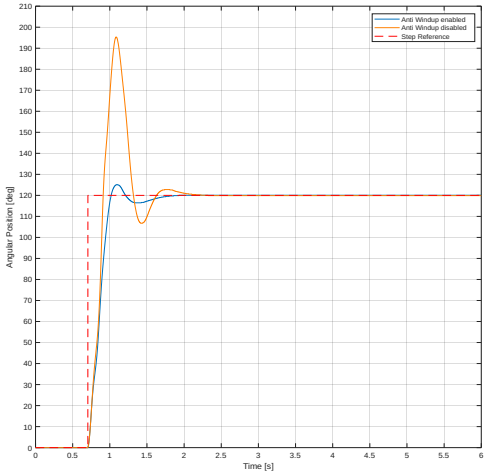


Figure 9. Black Box response to a 120° step reference

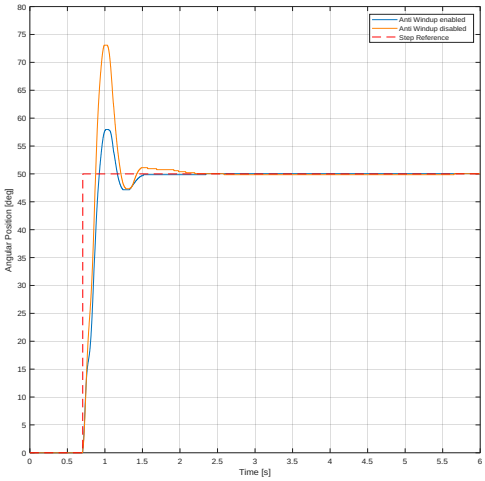


Figure 10. Real system response to a 50° step reference

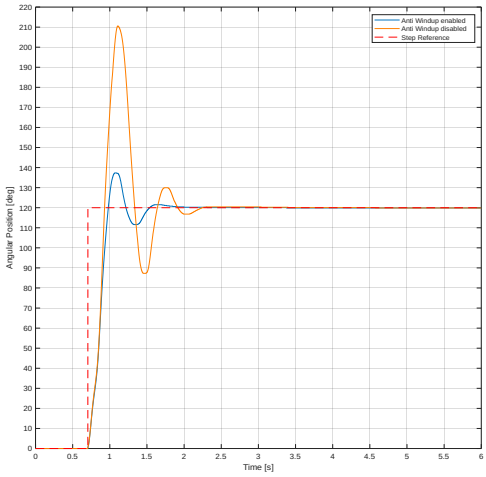


Figure 11. Real system response to a 120° step reference

Enabling the Anti Windup proved to be an effective method to decrease the overshoot, that would otherwise exceed the specifications. The simulated responses showed a less pronounced oscillatory behavior (compared to the real system) after the first peak: it is especially evident in the response to a 120° step, probably due to the increased weight of the linear approximations in the creation of the models (shown in the Appendix, from Figure 28).

2.3 Assignment #3: Position State Space Control using Eigenvalue Placement Design

After the PID design, we then employed state-space control via Eigenvalue Placement as detailed in Section 2.3 of the LAB3 Methods Paper. The objective of this activity was to validate the controller's ability to meet time-domain performance specifications in both the Accurate and Lab Environment models.

2.3.1 Accurate and Lab Environment Models: Nominal Eigenvalue Placement Design

It is immediate to notice that the state $\boldsymbol{x} = [\vartheta_h, \vartheta_d, \dot{\vartheta}_h, \dot{\vartheta}_d]^T$ is not fully accessible. In fact, while angular positions of the hub ϑ_h and beam ϑ_d are directly measured by the encoder and the potentiometer, the corresponding velocities $\dot{\vartheta}_h$ and $\dot{\vartheta}_d$ are not. To acquire estimates of these measurements, we implemented a real derivative via a High-Pass filter. The continuous-time filter used was:

$$H_1(s) = \frac{\omega_c^2 s}{s^2 + 2\delta\omega_c s + \omega_c^2} \tag{9}$$

with $\omega_c = 2\pi \cdot 50\,\text{rad/s}$ and $\delta = 1/\sqrt{2}$. These parameters strike a good balance between accurate differentiation and good rejection of high-frequency noise. The state-feedback controller was designed to meet the same performance specifications as mentioned in 1.1. To this end, the desired closed-loop set of eigenvalues were then assigned according to the following placement:

$$\lambda_{c,\{1,2\}} = \omega_n e^{j(-\pi \pm \phi)}, \qquad \lambda_{c,\{3,4\}} = \omega_n e^{j(-\pi \pm \phi/2)} \tag{10}$$

where the angle ϕ is defined as a function of the damping factor:

$$\phi = \arctan\left(\frac{\sqrt{1-\delta^2}}{\delta}\right) \tag{11}$$

This configuration ensures the placement of complex-conjugate poles in the left-half complex plane with appropriate spacing to match the desired dynamics.

Using MATLAB's `place` function, we computed the state-feedback gain matrix $K \in \mathbb{R}^{1\times 4}$, which was implemented for both the Accurate Model (see A.1.2 in the appendix for complete Matlab code and Simulink model) and the physical system of the Lab Environment.

2.3.2 Model Comparison: Nominal Eigenvalue Placement Design Results

A step reference of 50° was applied to the hub angular position. From the corresponding step response (Fig. 12), several observations can be made regarding the closed-loop system performance.

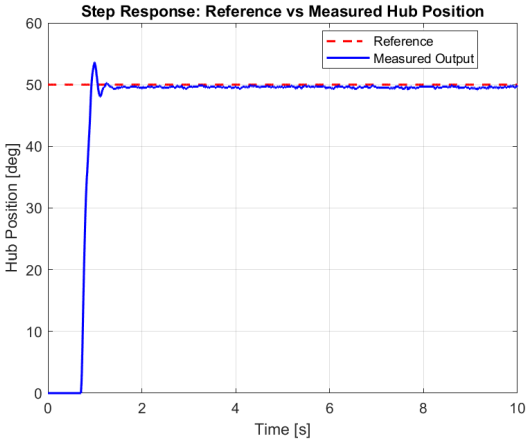


Figure 12. Lab Environment: Step Response

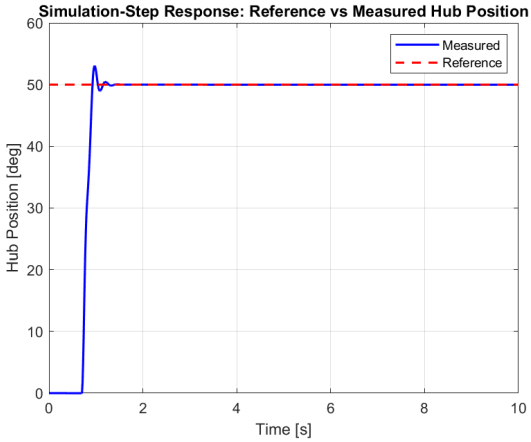


Figure 13. Accurate Model: Step Response

First, the settling time satisfies the design requirement (see tab 2). This fast response is primarily due to the deliberate placement of the dominant closed-loop poles, which were selected to achieve high natural frequency and moderate damping.Second, the overshoot remains well below the maximum allowed value of 30% (see tab 2), demonstrating that the damping ratio chosen in the eigenvalue placement was appropriate to limit transient excursions.

However, not all specifications are fully met. Notably, the system exhibits a nonzero steady-state error, which does not satisfy the ideal tracking objective. This behavior can be attributed to the limitations of the nominal design assumptions. In particular, the control law guarantees zero steady-state error only under nominal conditions, i.e., in the absence of model uncertainties, unmodeled dynamics, and external disturbances.

In the Lab Environment however, the actual system deviates from the nominal model due to non-ideal simplifications, parameter mismatches, and the presence of disturbances. These factors make for the mismatch between the Accurate and Lab Environment models.

To confirm the theoretical expectations, the simulation results shown in Figure 13 clearly demonstrate that, when the designed controller is applied to the system model with nominal parameters—i.e., in the absence of modeling errors, unmodeled dynamics, and external disturbances—the steady-state error is essentially zero.

	Prescribed Specs	Lab Environment	Accurate Model
t_s [s]	0.85	0.23	0.29
M_p [%]	30	8	6
e_∞ [deg]	0	≈ 0.4	≈ 0.01

Table 2: Comparison of step responses to a 50° input: prescribed specifications vs. laboratory and simulation

It is worth noting, however, that the error is not exactly zero, but rather on the order of 10^{-2}. This minor discrepancy arises from numerical effects inherent to the simulation environment, such as floating-point arithmetic precision and the discretization of continuous-time dynamics. These effects are typical in digital implementations and do not invalidate the conclusion that the controller achieves practically perfect tracking under ideal conditions.

In summary, the designed controller achieves fast and well-damped transient behavior with good tracking performance, but Lab Environment imperfections introduce a small but persistent steady-state error and minuscule oscillations in the final output.

2.3.3 Accurate and Lab Environment Models: Robust Eigenvalue Placement Design

The issues encountered with the nominal tracking controller — most notably the presence of a steady-state error in the step response — can be addressed by adapting a robust control strategy. In our implementation, robustness was achieved by augmenting the nominal state-feedback controller with integral action, a standard approach for rejecting constant disturbances and compensating for model inaccuracies.

To this end, the state vector was extended by including a new integrator state x_I, which integrates the error between the reference signal and the measured hub angle. The resulting augmented state vector becomes $\boldsymbol{x_e} = [x_I, x]^T \in \mathbb{R}^5$ and the augmented system is described by: $\Sigma_e = (A_e, B_e, C_e, 0)$

The control law applied takes the form:

$$u = -K_e x_e = -\begin{bmatrix} K_I & K \end{bmatrix} \begin{bmatrix} x_I \\ x \end{bmatrix} \tag{12}$$

where K_I is the gain associated with the integral state, and $K \in \mathbb{R}^{1\times 4}$ is the gain matrix associated with the original plant states, designed as in the nominal case.

For the design of K_e, five eigenvalues were placed as follows:

- Four complex-conjugate poles to shape the dominant second-order dynamics and provide adequate damping and speed.
- One additional real pole to govern the dynamics of the integrator.

Specifically, the chosen eigenvalues are:

$$\lambda_{c,\{1,2\}} = \omega_n e^{j(-\pi \pm \phi)}, \quad \lambda_{c,\{3,4\}} = \omega_n e^{j(-\pi \pm \phi/2)}, \quad \lambda_{c,5} = -\omega_n \tag{13}$$

where ω_n, δ, and ϕ are defined based on the same time-domain specifications used in the nominal controller design.

2.3.4 Lab Environmnent: Nominal to Robust Eigenvalue Placement Design Results

As shown in Figure 14, the response of the closed-loop system with integral action clearly demonstrates the benefits of this approach.

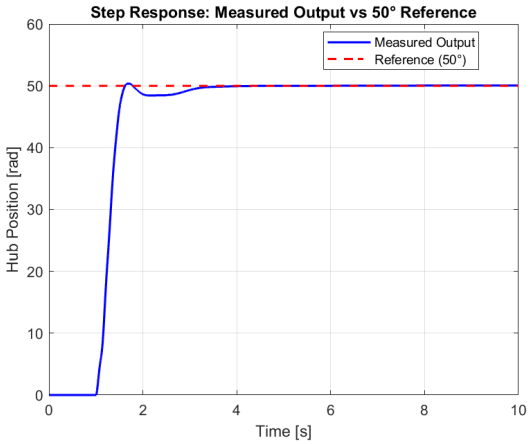


Figure 14. Step response corresponding to a 50° reference input with Robust Perfect Tracking Controller

	Prescribed	Nominal	Robust
$t_s\,[s]$	0.85	0.23	0.52
$M_p\,[\%]$	30	8	0.7
$e_\infty\,[deg]$	0	≈ 0.4	≈ 0.005

Table 3. Performance comparison of nominal and robust tracking controllers in the Lab Environment

The most significant improvement is the elimination of the steady-state error. In contrast to the nominal controller—which showed a steady-state deviation of approximately 0.4° under experimental conditions—the robust controller achieves near-perfect tracking, with the final output settling extremely close to the desired 50° setpoint. The resulting controller ensures robust perfect tracking of constant reference inputs, eliminating steady-state error even in the presence of modeling imperfections and constant disturbances, while maintaining desirable transient characteristics.

2.4 Assignment #4: Nominal Position State-Space Control using LQR methods

In this section, the request was to implement a controller using **Linear Quadratic Regulator** (LQR) design methods. As described in Section 6.1 of the LAB3 Methods Paper, LQR is an optimal control strategy used to determine a state-feedback control law $\boldsymbol{K}$ that minimizes a quadratic cost function:

$$\boldsymbol{J} = \int_0^{+\infty} \boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t)\; dt \tag{14}$$

$$\text{subject to: } \dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$

where $\boldsymbol{Q}$ is the state weighting matrix and $\boldsymbol{R}$ is the control weighting matrix of the cost function, quadratic in both $\boldsymbol{x}$ and $\boldsymbol{u}$. Therefore, this is a method that defines a control law by optimizing for state deviations from equilibria and control effort of the input.

The solution to this minimization problem comes in the well-recognized form of static-state feedback $\boldsymbol{K}$ such that the control input

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x} \tag{15}$$

stabilizes the system with optimal performance.

The result of the optimization depends on the choice of the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, whose entries define how state and input component costs relate to each other. This means that the quality of our

control design directly depends on how well we can identify our system dynamics and extract these costs from our model.

2.5 Accurate and Lab Environment Models: Nominal LQR via the Symmetric Root Locus (SRL)

One of the techniques used to define the cost matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ is the **Symmetric Root Locus** (SRL). This method is used when dealing with single-input, single-output systems with unitary state cost (i.e. $Q = 1$). The cost function in this case is

$$J = \int_0^{+\infty} y^2(t) + ru^2(t)\, dt \tag{16}$$

subject to

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \tag{17}$$

In this case, the only weight used is r, which is the scalar control input weight. It can be proven that, in this particular case, the closed-loop eigenvalues associated with the LQR design are the roots of:

$$p(s) = D(-s)D(s) + \frac{1}{r}N(-s)N(s) \tag{18}$$

where $G(s) = \frac{N(s)}{D(s)}$ is the input-output transfer function of the system.

Using this particular characteristic, the control weight r can be chosen using the root locus of the function $G(-s)G(s)$ parametrized in the gain $\frac{1}{r}$ to define a desired closed-loop behaviour.

To complete this task, the assignment suggested sketching the Root Locus and deciding the right gain $\frac{1}{r}$ by choosing one such that we remain in the admissible region of the root locus defined by the design specifications stated in Section 1.1. This Root Locus is shown in Fig. 15(a), where the blue lines represent the open trapezoidal admissible region in the left-half plane. This led to a choice of:

$$\frac{1}{r} = 3500 \;\rightarrow\; r = \frac{1}{3500}$$

Which led to the closed-loop eigenvalues being feasibly inside the admissible region. We can see this placement in Fig. 15(b).

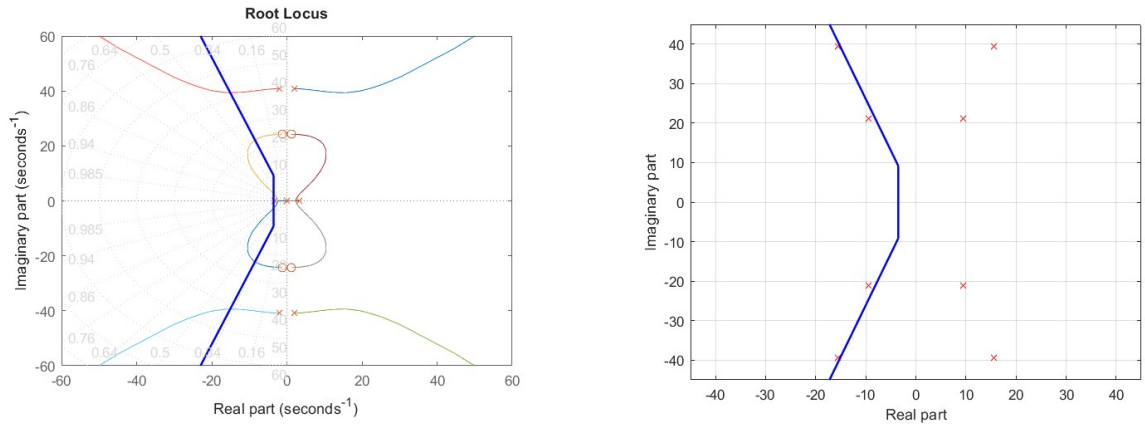


Figure 15. Entire root locus (left image) and resulting position of closed-loop eigenvalues (right image). The admissible region is defined by the left half-plane delimited by the blue lines.

After fixing r, the static state feedback $\boldsymbol{K}$ can be computed using the Matlab function `lqry` as `lqry(G,1,r)`. Finally, to have nominal tracking, a feedforward action is added to $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$ (as in the case of eigenvalue placement) to have unitary static gain under nominal conditions.

The Simulink scheme for the accurate model is the same as the nominal tracking problem with eigenvalues placement and can be found in Section A.3.3 of the Appendix.

2.5.1 Model Comparison: Nominal LQR via the Symmetric Root Locus (SRL) Results

The designed controller was tested on both the Accurate Model and the Lab Environment. The response of the systems, under a 50° Step Reference as usual, is shown in Fig. 16.

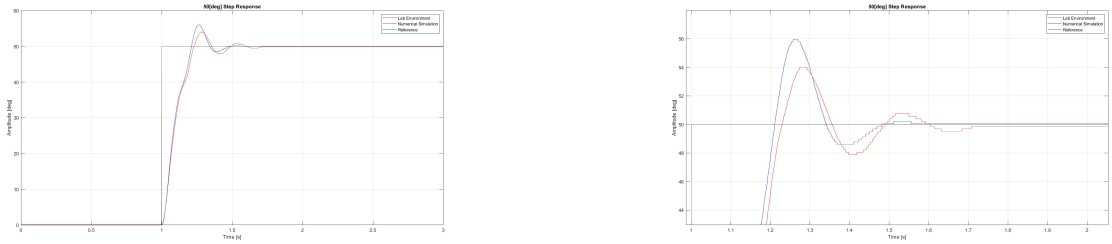


Figure 16. Step response of the controller in different setups (left) and close-up view of final part of the transient (right). Red Line is for the Accurate Model Response and the Blue line is for the Lab Environment Response

An observation that can be made about the results is again, the distinction in the performance of the Accurate (red line) Model and the Lab Environment (blue line) model. In particular, the prior has a larger overshoot than the latter. These results are listed in the Table 4.

	Accurate Model	Lab Environment
$t_r\,[s]$	0.1598	0.1718
$t_s\,[s]$	0.3141	0.32415
$M_p\,[\%]$	11.8705	8.3032

Table 4. Metrics comparison (LQR via SRL)

2.6 Accurate and Lab Environment Models: Nominal LQR via Bryson's Rule

Another technique used to define the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ of the LQR problem is Bryson's Rule. This method defines the cost matrices by imposing maximum allowable deviations of the state and control input. To use this form, the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are designed as diagonal matrices:

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{nn} \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} r_{11} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{mm} \end{bmatrix} \tag{19}$$

where

$$q_{ii} = \frac{1}{\overline{x_i}^2} \quad and \quad r_{ii} = \frac{1}{\overline{u_i}^2}$$

with $\overline{x_i}$ and $\overline{u_i}$ being, respectively, the maximum allowed i-th state component and maximum allowable i-th control component, which define the cost entries as reciprocals of squared norms.

In this case, including the state cost matrices, the cost function was

$$\boldsymbol{J} = \int_0^{+\infty} \boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + r\, u^2(t)\, dt \tag{20}$$

Using the proposed values given in the assignment paper:

$$\overline{\theta_h} = \frac{0.3\times 50\times \pi}{180}\,[rad] \qquad \overline{\theta_d} = \frac{\pi}{36}\,[rad] \qquad \overline{\dot{\theta}_h} = \overline{\dot{\theta}_d} = 0\,[rad] \qquad \overline{u} = 10\,[V]$$

After the definition of the matrices, the LQR was performed using the Matlab function `lqr(G,Q,R)`. The first tentative design led to a controller that controls the system with a good response, with performances that met the requirements stated in Section 1.1, so no other values were tried further.

2.7 Model Comparison: Nominal LQR via Bryson's Rule Results

The results of the controller tests are shown in Fig. 17 and in Table 5.

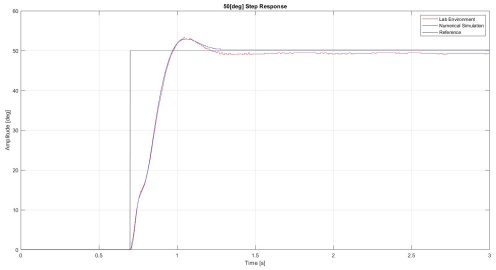


Figure 17. Step response of the controller designed using Bryson's Rule. Red Line is for the Accurate Model Response and the Blue line is for the Lab Environment Response

	Accurate Model	Lab Environment
$t_r\,[s]$	0.1998	0.2029
$t_s\,[s]$	0.4091	0.4343
$M_p\,[\%]$	5.7554	8.0292

Table 5. Metrics Comparison (Bryson's rule)

As expected, this nominal tracking controller cannot cancel the steady-state error, but the system is driven very efficiently, leading to a good response. As shown in Table 5, the performance is again slightly worse in the laboratory environment.

2.8 Assignment #5: Robust Tracking using LQR via Bryson's Rule

In this section, the controller has been designed using LQR design via Bryson's rule, following the same procedure outlined in Section 2.6, this time adapted for Robust Tracking conditions. This was done using integral action of constant hub position set-points. In this case it is required to use Bryson's Rule to define the matrices cost matrices to achieve the same performance requirements as stated in Section 1.1, same as in the Nominal case.

In this case, the only difference from the corresponding nominal controller is the addition of another weight for the integral action. The assignment gave five tentative values for the integral action weight q_{11}, which were $\{0.01,\ 0.1,\ 1{,}10,\ 100\}$. In Figure 18 we can see the performance of the controllers tuned with each of these candidate weights for the Accurate Model. The best choice, which also meets the design specifications, is $q_1 = 0.01$.

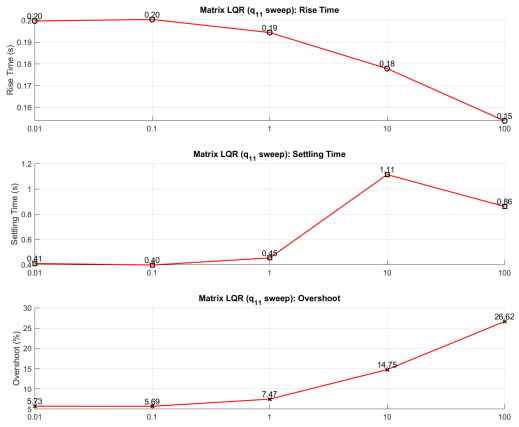


Figure 18. Performance of controller with different integral action weights

The controller was also tested using the Blackbox model, as an attempt to compensate for the Lab Environment.

2.8.1 Model Comparison: Robust LQR via Bryson's Rule Results

The results of the controller tests for both the Accurate and Black Box Models are shown in Fig. 19.

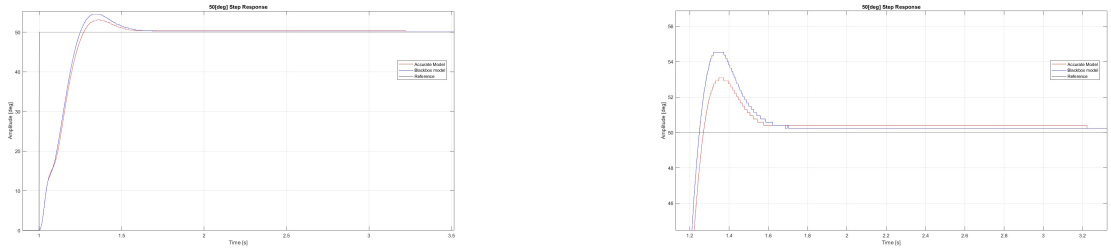


Figure 19. Step response of the robust controller designed using Bryson's Rule. Entire response (left) and close-up view (right). Red Line is for the Accurate Model Response and the Blue line is for the Lab Environment Response

Some observations can be made. We expect the integral action to completely eliminate the steady-state error. In the plots, the error is not eliminated, but it is decreasing over time. This can be the result of a 'weak' integral action, causing the controller to slowly converge to the reference. The results of these tests are reported in Table 6.

	Accurate Model	Lab Environment
$t_r\,[s]$	0.1598	0.1718
$t_s\,[s]$	0.3141	0.32415
$m_p\,[\%]$	11.8705	8.3032

Table 6: Metrics comparison.

2.9 Assigment #6: Nominal Tracking using FS-LQR

2.9.1 Preliminary Setup for FS-LQR

Frequency-Shaped LQR extends the design of static state feedback control to allow or reject specific parts of the frequency response of the system. This is done by casting some or all of the costs of the control effort or accuracy Matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ into the frequency domain, and creating a State-Space Realization of a filter which rejects specific portions in this domain, based on the desired response.

More particularly, in electromechanical systems like the motor with resonant load, we want to take care to avoid resonance and antiresonance. The prior results from complex conjugate poles in the plant's transfer function and shows up as a peak in Bode Magnitude plots, whereas the latter results from complex conjugate zeros in the plant's transfer function and shows up as a dip in Bode Magnitude plots. In resonant frequency, the system is at a regime which allows for maximal power transfer from control input to plant output, whereas in anti-resonant frequency, the system is at a regime of very lossy power transfer, where plant output is influenced very little, even by very large control input effort. For responsive and stable control design, we typically want to avoid both of these regimes.

Identifying unwanted frequencies: For Second-Order Systems, identification of the Resonant Frequency comes down to finding the natural frequency of the system ω_n, whereas for higher-order systems, after a State-Space Realization has been made, the spectrum of the State Matrix $\boldsymbol{A}$ gives all problematic resonant/antiresonant modes. For our system with transformed state $\boldsymbol{x'} = [\vartheta_h, \vartheta_d, \dot{\vartheta}_h, \dot{\vartheta}_d]^T$, the state matrix $\boldsymbol{A}$ has one complex conjugate pair $p_{1,2} = -2.09 \pm 41.15$, directly revealing that the resonant frequency to filter out of the response is $w_r = 41.15$. This can also be confirmed using formula (39) given in the Methods Paper for LAB3, giving the same result:

$$w_r = \sqrt{\frac{k}{J_b} + \frac{k}{J_{eq} * N^2}} \approx 41.15[rad/s] \tag{21}$$

Choosing the Weights to Frequency-Shape: For a system whose state vector is comprised of state components and their derivatives, resonant and antiresonant dynamics are more noticeable in the original state components. For example, in our setup, we can easily conclude that the beam is undergoing resonance by looking at the time evolution of ϑ_d, but this is not easily distinguishable if we look at the time evolution for $\dot{\vartheta}_d$. In FS-LQR design, we therefore target costs corresponding to zeroth-order states for frequency-shaping so that the penalty for unwanted dynamics is applied most effectively. This is why, in the Assignment Paper of LAB 3, we are instructed to frequency-shape the second diagonal entry of the $\boldsymbol{Q}$ matrix as $q_{22}(\omega)$, penalizing resonance from the spectrum of ϑ_d.

Following the procedure outlined in Task (9) of the Assignment Paper of LAB3, we construct the state space realization for the frequency-shaping transfer matrix $\boldsymbol{H_Q}(\boldsymbol{s})$, augment the state space model of our setup from Σ to Σ_A along with augmenting the costs into $\boldsymbol{Q_A}$ and $\boldsymbol{R_A}$, and use this newly extended system to solve the standard LQR problem for the feedback matrix $\boldsymbol{K_A}$ and to obtain the Feedforward Gains $\boldsymbol{N_A}$ and N_u for Nominal Tracking Conditions. The state space extension makes $\boldsymbol{x} = [\boldsymbol{x}, \boldsymbol{x_Q}]^T \in \mathbb{R}^4$. Finally, we validate this design by sweeping over three candidate values for the static portion of the cost q_{22} comprising the frequency-shaped cost $q_{22}(\omega)$:

$$q_{22} \in \{0.01, 1, 100\} \tag{22}$$

2.9.2 Accurate Model: Nominal FS-LQR Results

Following the preliminary steps and the controller design procedure outlined in Section 2.9.1, we record the responses for three candidate values of q_{22}. The MATLAB Code Snippet which designs all three controllers is reported in Section A.1.4 of the Appendix. All three responses satisfy the performance requirements of $Mp \leqslant 30\%$, $t_{s,5\%} \leqslant 0.85s$ and have the desired effect of attenuating the resonant mode of the system, which can be visually verified in the transient region, having no oscillatory behavior after frequency-shaping. The responses are displayed in Figure 20:

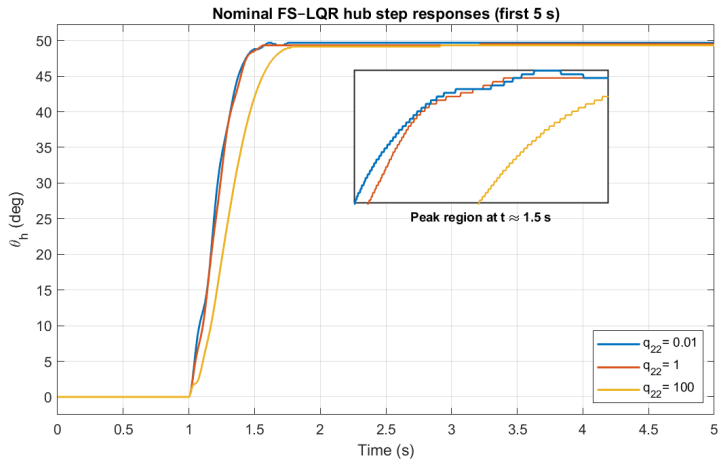


Figure 20. Accurate Model: 50° Hub Position Step Response

We also show how each q_{22} value affects the resonant mode of the system. The results are displayed in Figure 21, where the first and second subplots (top to bottom) display the real and imaginary parts of the closed-loop resonant poles, and the bottom third subplot shows the shift in the real part, from before and after applying the FS-LQR control design. A positive shift means the resonant mode has been amplified, and a negative shift means the resonant mode has been attenuated.

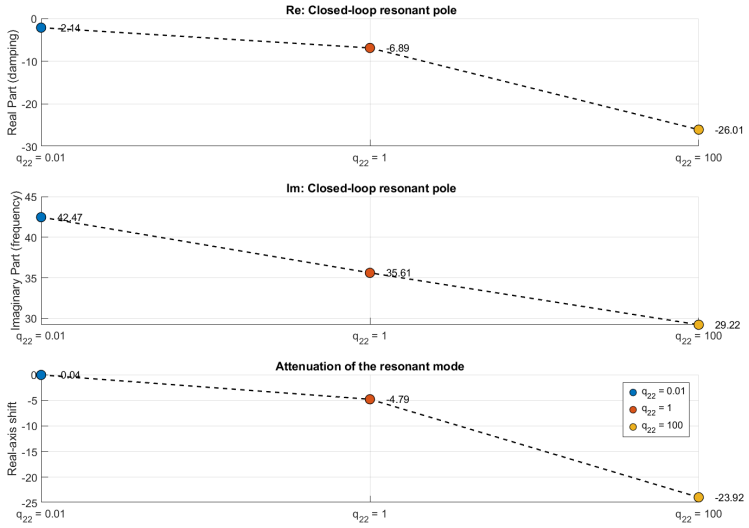


Figure 21. Accurate Model: The real and imaginary part of the resonant poles after applying FS-LQR Control (Top to bottom, first and second subplots) and the shift in the real part of the resonant mode before and after applying the control law (Bottom third subplot)

2.9.3 Black Box Model: Nominal FS-LQR Results

Due to logistical impediments, we resorted to testing with the Black Box model of the plant, and implemented the Bias Estimation mechanism to remove the offset between the real position ϑ_d and its voltage reading. As instructed in the Assignment Paper of LAB3, the Bias Estimation mechanism (reported in Figure 35 of the Appendix) is active in the interval$(t_0, t_1) = [0.2\,\text{s}, 0.7\,\text{s})$ after which Controller Input is enabled for $t \in [0.7s, \infty)$. The results we got were very similar to the accurate model, with the few differences outlined in this section.

The Step Responses remained very similar and differed only in the response with a weight of $q_{22} = 100$, which has a larger steady state error than its accurate model counterpart. The responses can be seen in Figure 22, and the resonant mode shifts due to frequency-shaping can be seen in Figure 23.

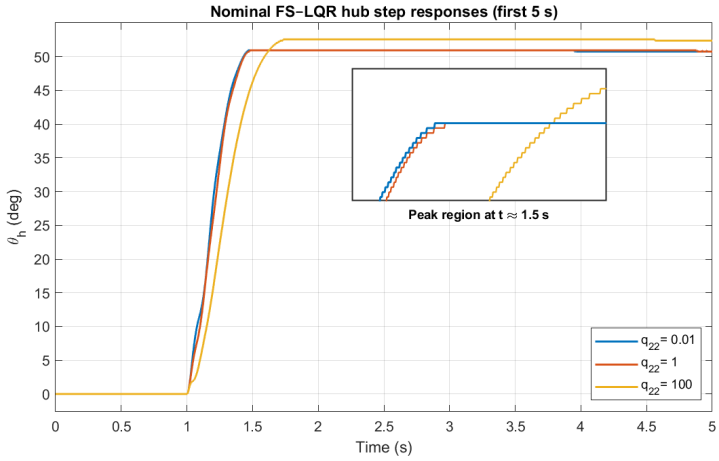


Figure 22. Black Box Model: 50° Hub Position Step Response (Left)

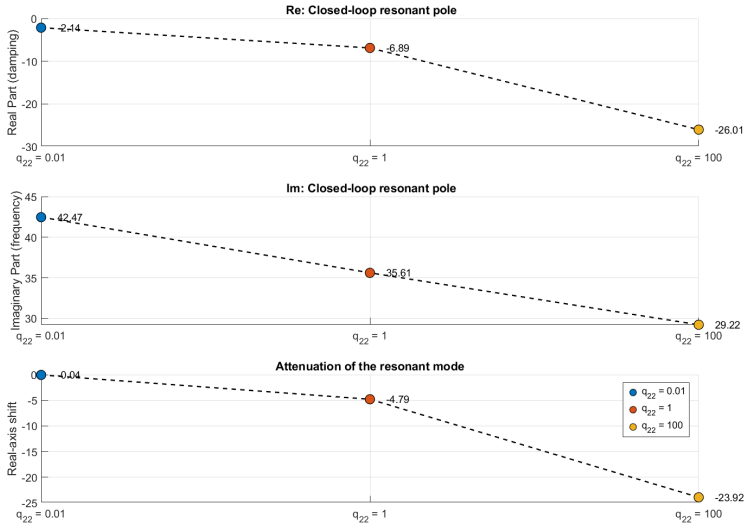


Figure 23. Black Box Model: The real and imaginary part of the closed-loop resonant poles (Top to bottom, first and second subplots) and shift in the resonant mode (Bottom third subplot)

2.9.4 Model Comparison: Nominal FS-LQR Results

To make a model comparison between performance metrics for Nominal Tracking using FS-LQR, we outline them in Table 7:

	Accurate Model			**Black-Box Model**		
	$q_{22} = 0.01$	$q_{22} = 1$	$q_{22} = 100$	$q_{22} = 0.01$	$q_{22} = 1$	$q_{22} = 100$
t_r [s]	0.332	0.339	0.444	0.303	0.311	0.389
t_s [s]	0.431	0.428	0.625	1.089	1.098	1.241
M_p [%]	0.00	0.00	0.00	0.35	0.35	3.18

Table 7: Accurate vs. Black Box Model comparison for three q_{22} designs

2.10 Model Comparison: Nominal tracking Methods Summary

In this section, the comparison of all state space controller designs under Nominal conditions is displayed, namely *Eigenvalues placement, Symmetric Root Locus (SRL), Bryson's Rule and Frequency-Shaped LQR*, . All designs are compared with their Accurate Model implementations and their Lab Environment counterparts in Table 8 (with the Black-Box model in FS-LQR instead):

	Eigs. Placement		**SRL**		**Bryson's(LQR)**		**FS-LQR**	
	Acc.	Lab.	Acc.	Lab.	Acc.	Lab.	Acc.	Blackb.
$t_r[s]$	0.17	-	0.1598	0.1718	0.1998	0.2029	0.332	0.303
$t_s[s]$	0.29	0.23	0.3141	0.32415	0.4091	0.4343	0.431	1.089
$M_p[\%]$	6	8	11.8705	8.3032	5.7554	8.0292	0.0	0.35

Table 8: Results for Nominal Tracking Designs

2.11 Assignment #7: Robust Position State-Space Control using LQR methods

2.11.1 Accurate Model: Robust FS-LQR

The procedure for FS-LQR Control Design for Robust Tracking Conditions with Integral action is very similar to that outlined in Section 2.9.1 for Nominal Tracking, with the additional step of extending the state space representation and cost matrices once more to accomodate for the Integral action term x_I. The code snippet making this possible is reported in Section A.1.5 of the Appendix. This augments Σ_A into Σ_e by one dimension, and the state cost matrix turns from $\boldsymbol{Q_A}$ into $\boldsymbol{Q_e}$ with one additional first diagonal entry q_I. This makes the state $\boldsymbol{x} = [x_I, \boldsymbol{x}, \boldsymbol{x_Q}]^T \in \mathbb{R}^5$. After the state space realization has been set up, what is left is to tune for the new cost entry q_I. We created a set of candidate q_I values to test:

$$q_I \in \{0.01, 0.1, 1, 10, 100\} \tag{23}$$

The design task comes down to finding the best pair of costs q_I in (23) and q_{22} in (22) through cross-testing, which we do by automating the MATLAB + Simulink simulation trials for every possible design combination. The Step Responses are summarized in Table 9. From the three best combinations highlighted in green, we see that we get a considerably good control performance for any q_{22} value when $q_I = 0.01$. We now assess performance for these three designs only in Figure 24, and display the eigenvalue shifts in Figure 25.

$\boldsymbol{q_{22} = 0.01}$				$\boldsymbol{q_{22} = 1}$				$\boldsymbol{q_{22} = 100}$			
q_I	t_r [s]	t_s [s]	M_p [%]	q_I	t_r [s]	t_s [s]	M_p [%]	q_I	t_r [s]	t_s [s]	M_p [%]
0.01	0.321	0.410	1.07	0.01	0.338	0.425	0.00	0.01	0.432	0.602	0.70
0.1	0.290	2.344	7.55	0.1	0.291	3.224	7.55	0.1	0.358	4.061	11.23
1	0.236	1.591	18.05	1	0.237	1.737	19.13	1	0.284	2.112	25.36
10	0.173	0.832	30.94	10	0.188	0.904	33.45	10	0.214	1.097	38.85
100	0.144	0.876	40.65	100	0.134	0.572	46.40	100	0.183	0.979	47.84

Table 9: 50° step–response metrics for FS-LQR design combinations under robust-tracking conditions. Combinations satisfying the performance requirements are highlighted in green.

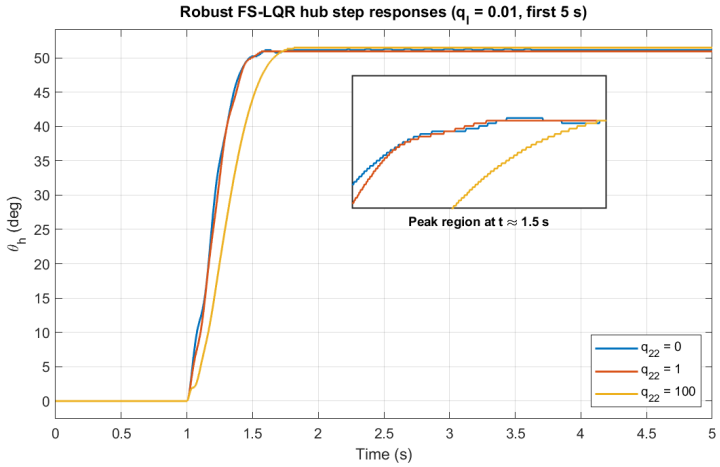


Figure 24. 50° Hub Position Step Response for q_{22} value sweep with $q_I = 0.01$

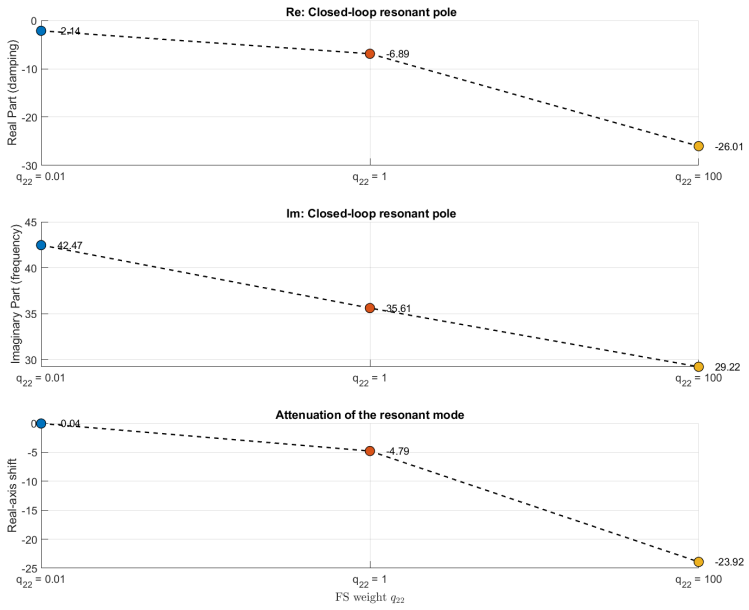


Figure 25. The real and imaginary part of the closed-loop resonant poles (Top to bottom, first and second subplots) and shift in the resonant mode (Bottom third subplot) for each q_{22} with $q_I = 0.01$

2.11.2 Black Box Model: Robust FS-LQR

In similar fashion to the Black Box model setup in Section 2.9.3, we follow the same procedure and display our results, again for a fixed $q_I = 0.01$ as the one giving the three best responses. When moving from the Accurate to the Black Box model, we again notice an increase in the steady state error of the response, especially for the one with weight of $q_{22} = 100$, as can be seen in Figure 26. The resonant mode shifts are shown in Figure 27 :

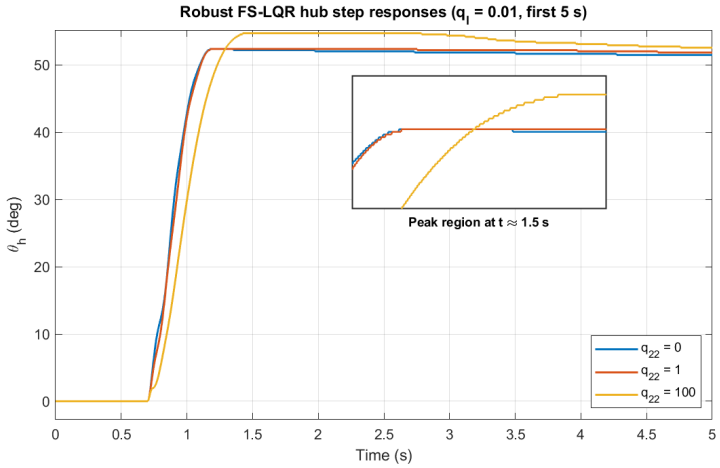


Figure 26. 50° Hub Position Step Response (Left) for q_{22} value sweep with $q_I = 0.01$

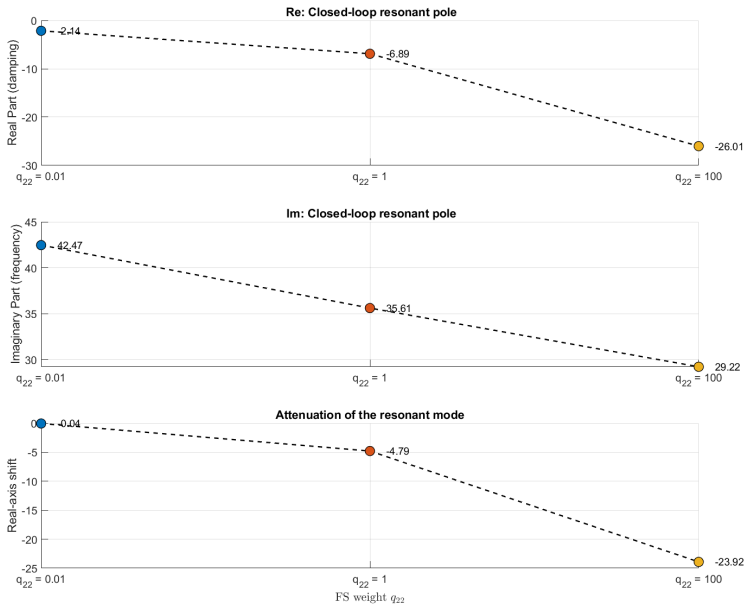


Figure 27. Black Box Model: The real and imaginary part of the closed-loop resonant poles (Top to bottom, first and second subplots) and shift in the resonant mode (Bottom third subplot)

2.11.3 Robust FS-LQR: Model Comparison

To make a model comparison between performance metrics for Robust Tracking using FS-LQR, we outline them in Table 10:

	Accurate Model			**Black-Box Model**		
	$q_{22} = 0.01$	$q_{22} = 1$	$q_{22} = 100$	$q_{22} = 0.01$	$q_{22} = 1$	$q_{22} = 100$
t_r [s]	0.321	0.338	0.432	0.288	0.294	0.357
t_s [s]	0.410	0.425	0.602	1.063	1.079	3.970
M_p [%]	1.07	0.00	0.70	3.19	2.83	8.19

Table 10: Robust-tracking metrics comparison for the three q_{22} designs (fixed $q_I = 0.01$).

A Appendix

A.1 Matlab code

A.1.1 PID with Anti Windup Controller Design

```
%% Plant transfer function
s = tf('s');

Dtau1 = (J_eq*mld.Jb) * s^3 + ...
        (J_eq*mld.Bb + mld.Jb*B_eq) * s^2 + ...
        (B_eq*mld.Bb + mld.k*(J_eq + mld.Jb/(gbox.N)^2)) * s + ...
         mld.k*(B_eq + mld.Bb/(gbox.N)^2);

P =     1/(gbox.N * s) * ...
        ( (drv.dcgain*mot.Kt*(mld.Jb * s^2 + mld.Bb * s + mld.k)) /...
        (R_eq*Dtau1 + mot.Kt*mot.Ke*(mld.Jb * s^2 + mld.Bb * s + mld.k)) );

%% Initial values for Bode Method - ts5% = 0.85 s, Mp = 30%
ts5 = 0.55; % Settling time at 5%
Mp = 0.24;  % Maximum Overshoot

d = log(1 / Mp) / sqrt(pi^2 + (log(1 / Mp))^2);         % Damping
wgc = 3 / (d * ts5);                                    % Crossing frequency
phim = atan(2 * d / (sqrt(sqrt(1 + 4*d^4) - 2 * d^2))); % Phase Margin

%% Computation of the Frequency Response of P(j*wgc)
[mag, phase_deg] = bode(P, wgc);
phase = phase_deg*pi/180;
Dk = 1 / mag;
Dphi = -pi +phim -phase;

%% Computation of the PID Gains
a = 4;  % Alpha

Td = (tan(Dphi) + sqrt(tan(Dphi)^2 + 4/a)) / (2*wgc); % Derivative Time
Ti = a * Td;                                          % Integral Time

Kp = Dk * cos(Dphi); % Proportional Gain
Kd = Kp * Td;        % Derivative Gain
Ki = Kp / Ti;        % Integral Gain

%% Anti Windup Constant
Tw = ts5 / 5;
Kw = 1 / Tw;

%% Real Filter Implementation
TL = 1 / (10 * wgc);
```

A.1.2 Nominal Perfect Tracking Controller Implementation

```
%System Matrices
A = [0, 0, 1, 0;
     0, 0, 0, 1;
     0, mld.k/(gbox.N^2*Jeq), -1/Jeq*(Beq + mot.Ke*mot.Kt/Req), 0;
     0, -mld.k/mld.Jb - mld.k/(gbox.N^2*Jeq), -mld.Bb/mld.Jb+1/Jeq*(Beq + mot.Ke*mot.Kt/Req) , -mld.Bb/mld.Jb
          ;];

```

```
B = [0, 0, mot.Kt*drv.dcgain/(gbox.N*Jeq*Req), -mot.Kt*drv.dcgain/(gbox.N*Jeq*Req)].';
C = [1, 0, 0, 0];
D = 0;


% Calculation for Nominal Tracking
N_mat = [A, B;
          C, 0];

N_res = linsolve(N_mat, [0, 0, 0, 0, 1].');

Nx = N_res(1:4);
Nu = N_res(5);

%  Desired eigenvalues
p1 = wn*exp(1j*(-pi+phi));
p2 = wn*exp(1j*(-pi-phi));
p3 = wn*exp(1j*(-pi+phi/2));
p4 = wn*exp(1j*(-pi-phi/2));

p_nom = [p1, p2, p3, p4];

% Calculation Feedback Matrix
K_nom = acker(A,B,p_nom);
```

A.1.3 Robust Perfect Tracking Controller Implementation

```
%System Matrices
A = [0, 0, 1, 0;
     0, 0, 0, 1;
     0, mld.k/(gbox.N^2*Jeq), -1/Jeq*(Beq + mot.Ke*mot.Kt/Req), 0;
     0, -mld.k/mld.Jb - mld.k/(gbox.N^2*Jeq), -mld.Bb/mld.Jb+1/Jeq*(Beq + mot.Ke*mot.Kt/Req) , -mld.Bb/mld.Jb
          ;];

B = [0, 0, mot.Kt*drv.dcgain/(gbox.N*Jeq*Req), -mot.Kt*drv.dcgain/(gbox.N*Jeq*Req)].';
C = [1, 0, 0, 0];
D = 0;

%Matrices for Robust Tracking with Integral Action
A_e = [0, C;
       [0;0;0;0], A;];
B_e = [0; B];
C_e = [0, C];


% Calculation for Nominal Tracking
N_mat = [A, B;
         C, 0];

N_res = linsolve(N_mat, [0, 0, 0, 0, 1].');

Nx = N_res(1:4);
Nu = N_res(5);

%  Desired eigenvalues
p1 = wn*exp(1j*(-pi+phi));
p2 = wn*exp(1j*(-pi-phi));
p3 = wn*exp(1j*(-pi+phi/2));
p4 = wn*exp(1j*(-pi-phi/2));
```

```
p5 = -wn;

p_rob = [p1, p2, p3, p4, p5];
Ke = place(A_e,B_e,p_rob);
K_rob = Ke(2:5);
Ki_rob_calc = Ke(1);
%Hand tuned Ki to archieve best results by trial and error
Ki_rob = 100;
```

A.1.4 Nominal Tracking FS-LQR (Accurate and Black Box Models): Controller Design Function

```
\begin{lstlisting}[language=Matlab]
%% State Feedback Design using Matrix-Valued, FS-LQR

q2_vals = [0.01, 1, 100];

% Arrays to store e.m. comparison results
real_vals = zeros(size(q2_vals));
imag_vals = zeros(size(q2_vals));
attenuation_vals = zeros(size(q2_vals));

function [A_aug, B_aug, omega_0, K_fb, eig_A] = fs_lqr(q2_value, A_prime, B_prime, C_prime, D_prime)

    clear K_fb; % make way for feedback to be designed

    % Approximated system  (prepare for the SRL)
    sysG = ss(A_prime, B_prime, C_prime, D_prime);

    %% Obtaining the resonant frequency of the Natural System Evolution:

    eig_A = eigs(A_prime);

    omega_0 = abs(imag(eig_A(find(imag(eig_A) ~= 0, 1))));

    %% Building the Frequency-Shaped Components (See Assignment Paper)

    % Define the frequency-shaping filter

    H = sqrt(q2_value)*tf([0 0 (omega_0)^2], [1 0 (omega_0)^2]);

    % SSM Realization of H:

    Hss.A = [0 1; -omega_0^(2), 0];
    Hss.B = [0;1];
    Hss.C = [sqrt(q2_value)*omega_0^(2), 0];
    Hss.D = 0;
    A_q = Hss.A;
    B_q = [zeros(2,1), Hss.B, zeros(2)];
    C_q = [zeros(1,2); Hss.C; zeros(2)];
    D_q_vec = [1/((0.3)*(5)), Hss.D, 0, 0];
    D_q = diag(D_q_vec);
    % Augment A_prime and B_prime
    A_aug = [A_prime, zeros(4,2) ; B_q, A_q];
    B_aug = [B_prime; 0;0];
    C_aug = [C_prime, zeros(1,2)];
```

```
    D_aug = 0;
    Q_aug = [D_q'*D_q, D_q'*C_q; C_q'*D_q, C_q'*C_q];
    %% Solving for the Nominal Feedforward Gains:
    M = [A_aug,B_aug;C_aug,D_aug];
    Sol = M \ [0;0;0;0;0;0;1];
    Nx = Sol(1:6, :);
    Nu = Sol(7,:);
    R_aug = 1 / (10^2);  % Still bound input by 10V
    K_fb = lqr(A_aug, B_aug, Q_aug, R_aug);
    assignin('base', 'A_q', A_q);
    assignin('base', 'B_q', B_q);
    assignin('base', 'C_q', C_q);
    assignin('base', 'D_q', D_q);
    assignin('base', 'Nx', Nx);
    assignin('base', 'Nu', Nu);
    assignin('base', 'K_fb', K_fb);
end
```

A.1.5 Robust Tracking with Integral Action FS-LQR (Accurate and Black Box Models): Controller Design Function

```
%% State Feedback Design using Matrix-Valued, FS-LQR
q2_vals = [0.01, 1, 100];
qi_vals = [0.01, 0.1, 1, 10, 100];
% Arrays to store e.m. comparison results
real_vals = zeros(size(q2_vals));
imag_vals = zeros(size(q2_vals));
attenuation_vals = zeros(size(q2_vals));
function [Ae, Be, omega_0, K_vals, eig_A] = fs_lqr(q2_value, qi_vals, A_prime, B_prime, C_prime)
    clear K_fb; % make way for feedback to be designe
    % Obtaining the resonant frequency of the Extended Natural System Evolution:
    eig_A = eigs(A_prime);
    % find the first eigenvalue of nonzero imaginary part:
    omega_0 = abs(imag(eig_A(find(imag(eig_A) ~= 0, 1))));
    % Building the Frequency-Shaped Components (See Assignment Paper)
    % Define the frequency-shaping filter / SSM Realization of transfer function H(s):
    Hss.A = [0 1; -omega_0^(2), 0];
    Hss.B = [0;1];
    Hss.C = [sqrt(q2_value)*omega_0^(2), 0];
    Hss.D = 0;
    % SSM Realization of the Filtering Matrix H_Q(s):
    A_q = Hss.A;
    B_q = [zeros(2,1), Hss.B, zeros(2)];
    C_q = [zeros(1,2); Hss.C; zeros(2)];
    D_q_vec = [1/((0.3)*(5)), Hss.D, 0, 0];
    D_q = diag(D_q_vec);
    % Augment A_prime and B_prime
    A_aug = [A_prime, zeros(4,2) ; B_q, A_q];
    B_aug = [B_prime; 0;0];
    C_aug = [C_prime, zeros(1,2)];
    D_aug = 0;
    Q_aug = [D_q'*D_q, D_q'*C_q; C_q'*D_q, C_q'*C_q];
    % Extended Matrices to include Integral Action
    Ae = [0, C_aug; zeros(6,1), A_aug];
    Be = [0; B_aug];
    Ce = [0, C_aug];
    fprintf("Checking system matrices before LQR...\n");
    fprintf("Controllability rank: %d (expected: %d)\n", rank(ctrb(Ae, Be)), size(Ae,1));
    % Defining Cost Matrices:
```

```
    R_aug = 1 / (10^2);  % Still bound input by 10V
    Re = R_aug;
    K_vals = [];
    for i = 1:length(qi_vals)
        % Define Qe fresh every time
        Qe = blkdiag(qi_vals(i), Q_aug);  % 1 + 6 = 7 state cost
        % Check definiteness
        fprintf("Min eigenvalue of Qe (%d): %.4e\n", i, min(eig(Qe)));
        fprintf("Min eigenvalue of Re: %.4e\n", min(eig(Re)));
        % Compute LQR gain
        K_fb = lqr(Ae, Be, Qe, Re);  % Expect 1 7
        % Store gain
        K_vals = [K_vals; K_fb];
    end
    % push into workspace (for the H_Q(s) SSM Block):
    assignin('base', 'A_q', A_q);
    assignin('base', 'B_q', B_q);
    % Solving for the Nominal Feedforward Gains:
    M = [A_aug, B_aug; C_aug, D_aug];
    Sol = M \ [0;0;0;0;0;0;1];
    Nx = Sol(1:6, :);
    Nu = Sol(7,:);
    assignin('base', 'Nx', Nx);
    assignin('base', 'Nu', Nu);
end
```

A.1.6 Symmetric Root Locus Sketching

```
%% SRL (Symmetric Root Locus) sketching
%Variables to define the Admissible Region
phi = atan(sqrt(1-delta^2)/delta);
x_max = -3/t_settle;
x_min = -200;
m1 = tan(phi);
m2 = tan(-phi);
x_range = [x_min, x_max];
y_range1 = m1* x_range;
y_range2 = m2* x_range;
%Range of gains to draw a partial root locus
k_range = linspace(2e3,3.5e3, 1e3);
%Root Locus sketching (with admissible region)
rlocus(sysG*sysGp,k_range);
hold on;
line(x_range, y_range1, 'Color','b', 'LineWidth',1.5);
line([x_max x_max], [m1*x_max,m2*x_max], 'Color','b', 'LineWidth',1.5);
line(x_range, y_range2, 'Color','b', 'LineWidth',1.5);
xlim([-100 100]);
ylim([-100 100]);
grid on;
K_final = 3.5e3; %Chosen K (smaller K s.t. all the poles are inside the region)
r = 1/K_final;
K_fb = lqry(sysG, 1, r); %Resulting Optimal State Feedback Gain
```

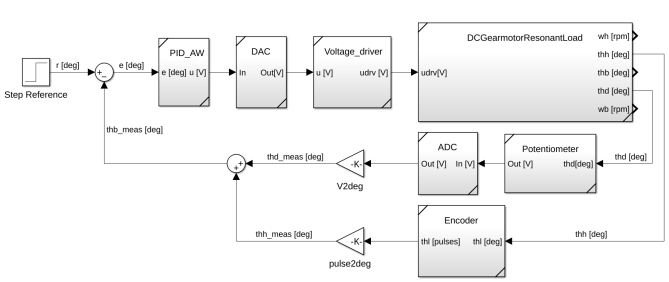
A.2 Data sheet

Variable	Value	Description
R_a	2.6 Ω	Armature resistance
L_a	180 μH	Armature inductance
k_e	7.68 $\times$ 10^{-3} Vs/rad	Electric (BEMF) constant
k_t	7.682 $\times$ 10^{-3} Nm/A	Torque constant
J_m	3.90 $\times$ 10^{-7} $kg\,m^2$	Rotor inertia
N	14	Gearbox ratio
J_{72}	1.4 $\times$ 10^{-6} $kg\,m^2$	Moment of inertia of external 72-tooth gear
J_d	3.0 $\times$ 10^{-5} $kg\,m^2$	Moment of inertia of extra disc
$\hat{B}_{eq}$	1.0459 x 10^{-6} $N\,m\,s/rad$	Estimated viscous friction coefficient, motor side
k_{drv}	0.6	Voltage driver static gain
$f_{c,drv}$	1.2 kHz	Voltage driver cut-off frequency
R_s	0.5 Ω	Shunt resistance
J_h	6.84 $\times$ 10^{-4} $kg\,m^2$ (Alu), 5.1 $\times$ 10^{-4} $kg\,m^2$ (Quanser)	Hub moment of inertia
τ_{sf}	0.0061 $N\,m$(estimated)	Static Friction Coefficient
J_b	1.4 $\times$ 10^{-3} $kg\,m^2$ (Alu), 2.0 $\times$ 10^{-3} $kg\,m^2$ (Quanser)	Beam moment of inertia
$\hat{B}_b$	3.32 x 10^{-3} $N\,m\,s/rad$	Estimated Beam viscous friction coefficient
$\hat{k}$	0.782 $N\,m/rad$	Estimated Beam joint stiffness
u_a	-	Supply voltage to the armature circuit
i_a	-	Current to the armature circuit
u_e	-	Back electromotive force (BEMF)
τ_m	-	Motor side torque
ω_m	-	Motor side speed
θ_m	-	Motor side position
τ_l	-	Load side torque
ω_l	-	Load side speed
θ_l	-	Load side position
τ_d	-	Disturbance torque applied to the load inertia
τ_h	-	Hub torque
ω_h	-	Hub speed
ϑ_h	-	Hub position
ω_b	-	Beam speed
ϑ_b	-	Beam position
ϑ_d	-	Beam displacement angle ($\vartheta_b - \vartheta_h$)
$\tau_{d,h}$	-	Disturbance torque on hub inertia
$\tau_{d,b}$	-	Disturbance torque on beam inertia
u	-	Voltage driver input voltage
u_{drv}	-	Voltage driver output voltage

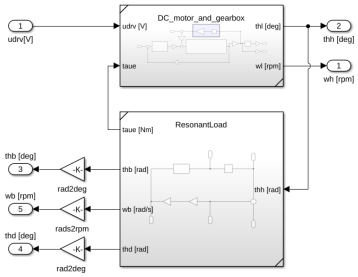
Table 11: Complete list of system variables and parameters

A.3 Simulink Schemes

A.3.1 Complete Model with PID Control scheme



Complete Control Scheme



GearMotor and Resonant Load

Figure 28. Complete Model with PID Control scheme

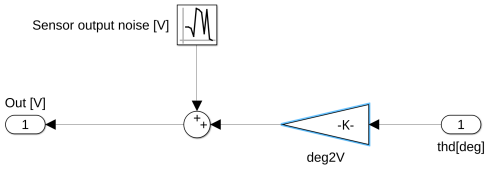
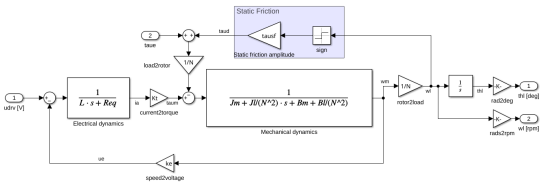
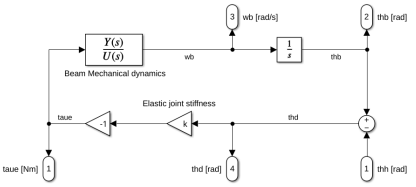


Figure 29. Potentiometer



DC Motor and Gearbox



Resonant Load

Figure 30. Subsystems for GearMotor and Resonant Load

A.3.2 PID with Anti Windup

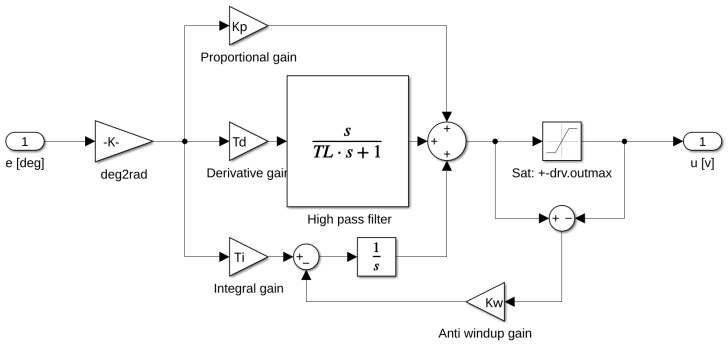


Figure 31. PID with Anti Windup control scheme

A.3.3 State Space Control scheme

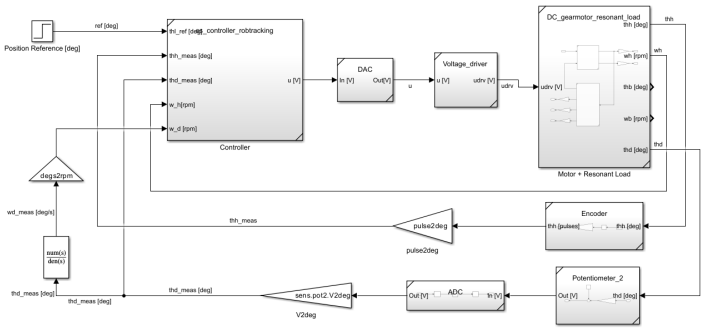


Figure 32. Simulink state-space control scheme with accurate model used for tests.

A.3.4 Nominal tracking Control Law

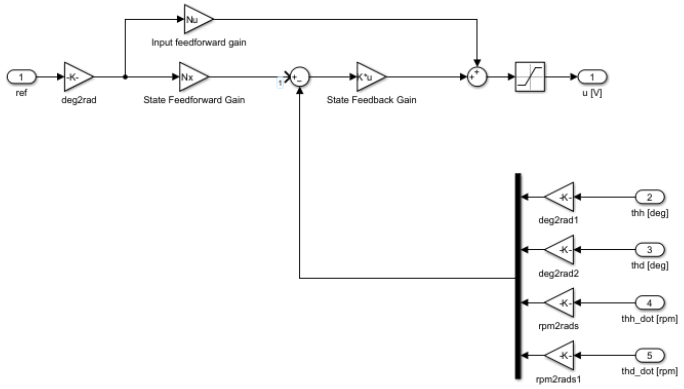


Figure 33. Nominal tracking Simulink control scheme

A.3.5 Robust tracking with integral action Control Law

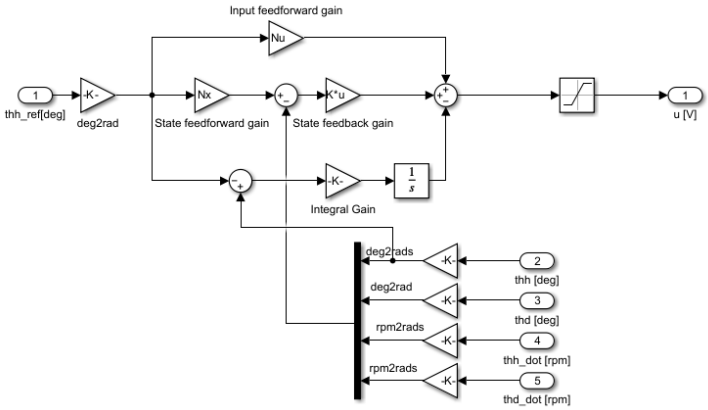


Figure 34. Robust tracking with integral action Simulink control scheme

A.3.6 Beam Displacement Sensing Scheme

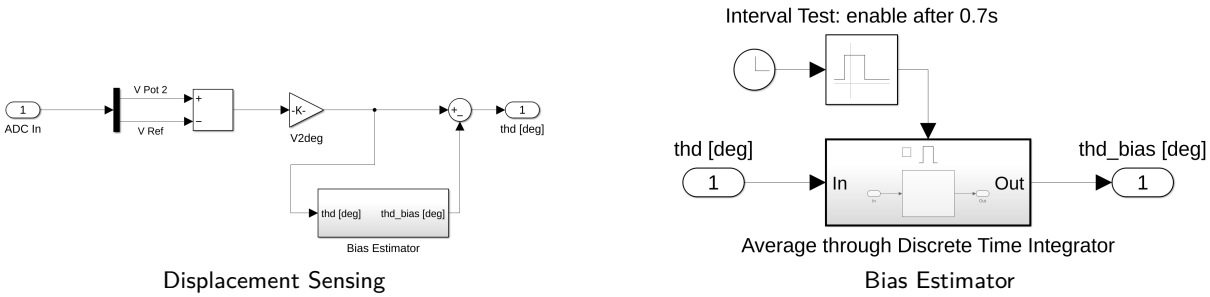


Figure 35. Beam Displacement Sensing Scheme