TUHH
Technische Universität Hamburg

ICS
Institute of Control Systems

Control Lab
CSTD1

Identification and Control of a Torsional Plant

23rd April, 2021

Room: Online | N-1.077

Summer Semester 2021

Contents

About this Document

This document is intended to help you with the design task *CSTD1—Identification and Control of a Torsional Plant* of the *Control Lab* practical course. It is written mainly in the style of a tutorial and should provide you with all the necessary tools and MATLAB commands to solve the task.

This document is accompanied by several MATLAB files that you need to modify and execute in order to develop your own design.

`SysIdent_ParameterEstimation.m' is the main script for parameter estimation.

`SysIdent_BuildMeasurementMatrix.m' contains data preprocessing and arrangement algorithms for parameter estimation.

`LeadLag_Design.m' is the main script for lead-lag compensator design and analysis.

`LQG_Design.m' is the main script for LQG compensator design and analysis.

`OpenLoopModel_Sim.slx'** and **`OpenLoopModel_Exp.slx' will be used for simulation and experiment based parameter identification, respectively.

`LeadLag_Sim.slx'** and **`LeadLag_Exp.slx' contain setups for simulation and experimental use of the lead-lag compensator.

`LQG_Sim.slx'** and **`LQG_Exp.slx' contain setups for simulation and experimental use of your LQG and LQGi designs.

The code is guaranteed to work with MATLAB 2019a 64bit, other versions might not be supported. You can get the latest MATLAB version from `https://www.tuhh.de/rzt/usc/Matlab/index.html` or use the pool computers.

In this document, you will encounter blocks that indicate MATLAB code:

```
[MATLAB COMMANDS]=USEFUL(TOOLS)
```

These are meant to get you started. You can (and should) use the `help` command within MATLAB to find out more about a particular command.

Another thing that you will encounter are preparation tasks, for instance:

Preparation: What can you tell from such stability margins about the plant?

These are meant to prepare you for the question session that will take place prior to conducting the experiment.

Task

Design and compare a lead-lag compensator and 2 LQG controllers for the torsional plant based on a model to identify. Attach all necessary MATLAB and SIMULINK files that were provided to you no later than one week before the experiment via email to the responsible Tutor and Supervisor. You will get an email when your preparation is not sufficient to pass the Lab and will get time to revise your design.

Checklist

- ☐ I read this whole document carefully.
- ☐ I did all preparation tasks and can explain them.
- ☐ I completed **`SysIdent_ParameterEstimation.m'**.
- ☐ I completed **`SysIdent_BuildMeasurementMatrix.m'**.
- ☐ I completed **`LeadLag_Design.m'**.
- ☐ I completed **`LQG_Design.m'**.
- ☐ I submitted all MATLAB files.
- ☐ I submitted my answers to all preparation questions as a pdf file.

1 PLANT

The ECP Torsional Plant in Fig. 1 has a configuration of a series of three disks and two torsional springs, k_1 and k_2, respectively. By changing the number of masses and/or their location on a given disk, its moment of inertia J_i, $i = 1, 2, 3$ is accordingly changed. Each disk model includes a damping constant d_i, $i = 1, 2, 3$.

A brushless servo DC motor at the bottom of the apparatus is used to excite the series of spring-disks with a torque signal T. The angular position θ_i, $i = 1, 2, 3$ of each disk can be measured by means of an incremental encoder.

Given the voltage input to the DC motor as the input and the measured angles of the three disks as outputs, the plant, in its general configuration, is a SIMO system (Single-Input-Multiple-Output). This configuration is used for identification purposes.

However, for the control experiments, the plant is regarded as a SISO system. The controlled variable is the position of the top disk, θ_3, i.e. only this signal is used for feedback (see Figure 2).

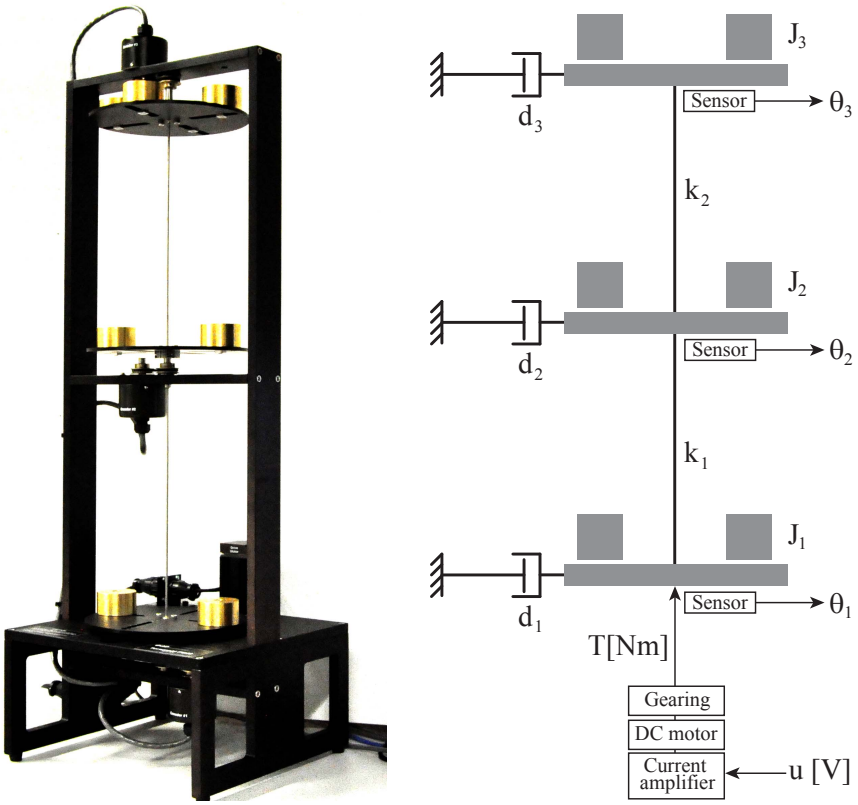

Figure 1: Torsional apparatus photography and schematic.

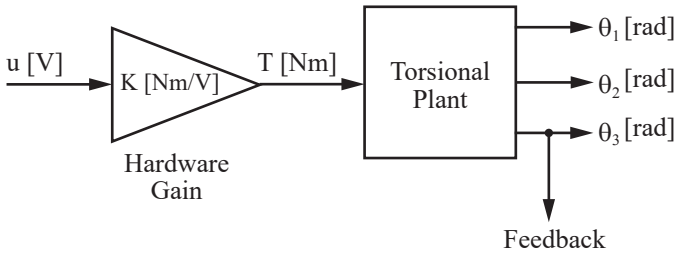

Figure 2: Input and output signals of the torsional plant.

1.1 Control Objectives

Given a reference step of 360°, as the input signal, to the closed-loop system (reference command for θ_3), the following specifications are defined to test the performance of the designed controllers (see Figure 3):

1. Settling time $t_s < 3\,\mathrm{s}$ (of $\pm 5\,\%$)
2. Rise time $t_r < 1.5\,\mathrm{s}$ (from $10\,\%$ to $90\,\%$ to the set point)
3. Peak overshoot $M_p < 10\,\%$ (relative to set point)
4. Steady state error $e_{ss} < 2°$ (for $t > 20\,\mathrm{s}$)

Meanwhile, it has to be guaranteed that the controller does not exceed the system limits (see Section 1.5). Special attention needs to be paid to the maximum twist angle of the springs to avoid a permanent damage (see Section 1.5). A suitable controller should realize a trade-off among these objectives.

1.2 Physical Plant Model

Assuming that non-linear friction is negligible, we can derive the differential equation of motion for each disk by summing torques acting on it, according to Euler's equations of motion (rotational form of Newton's second law):

$$\begin{aligned} T &= J_1\ddot{\theta}_1 + d_1\dot{\theta}_1 + k_1(\theta_1 - \theta_2) \\ 0 &= J_2\ddot{\theta}_2 + d_2\dot{\theta}_2 - k_1(\theta_1 - \theta_2) + k_2(\theta_2 - \theta_3) \\ 0 &= J_3\ddot{\theta}_3 + d_3\dot{\theta}_3 + k_2(\theta_3 - \theta_2) \end{aligned} \tag{1}$$

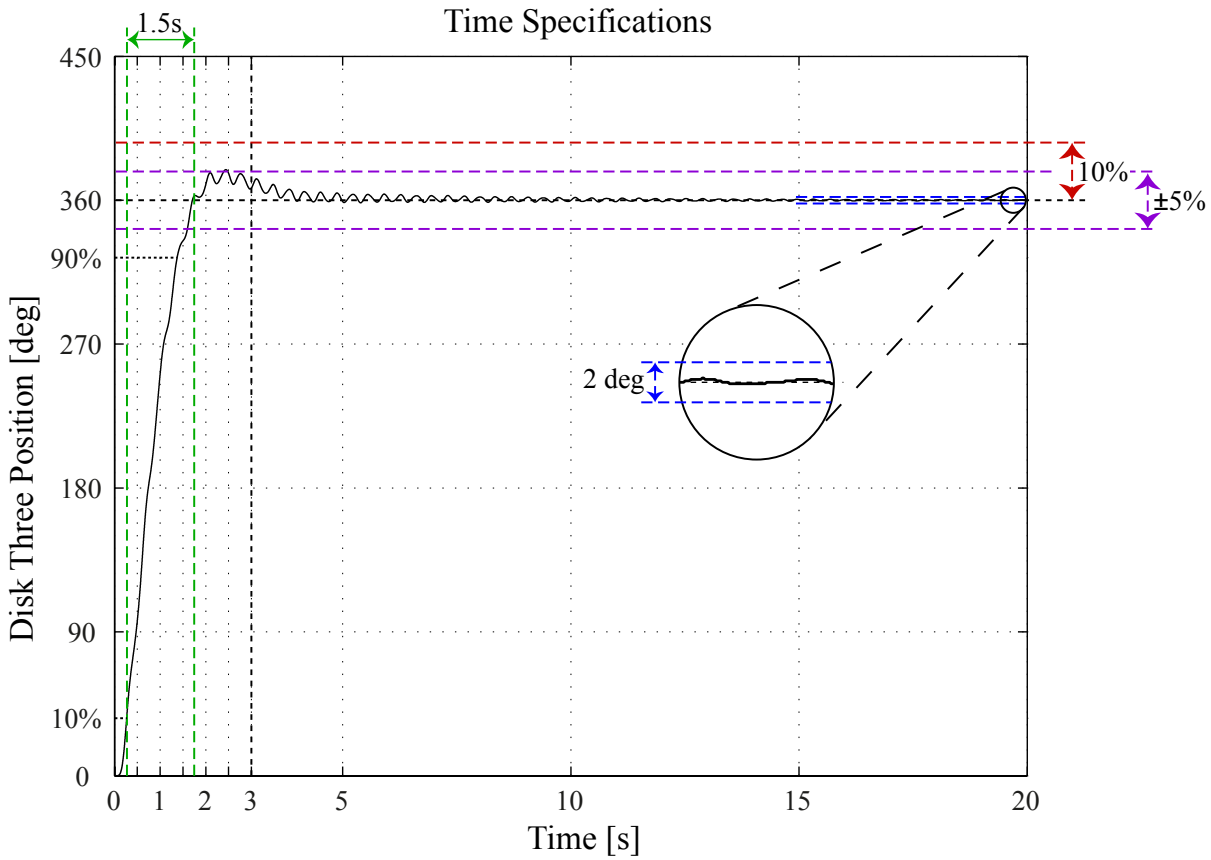

Figure 3: Time constraints to be fulfilled by the controllers.

For the three disks labeled $i = 1, 2, 3$, J_i is the inertia of the ith disk; d_i is the damping coefficient acting on the ith disk; θ_i is the rotated angle of the ith disk; k_1 and k_2 are the torsional stiffnesses of two perpendicular torsional springs.

To bring the differential equations of motion (1) into state space form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx, \end{aligned} \tag{2}$$

the state vector x is chosen as: $x = [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, \theta_3, \dot{\theta}_3]^{\mathrm{T}}$, where $\dot{\theta}_i$ denotes the angular velocity of the ith disk. The control input u represents the torque applied from the DC motor. The angular positions of the three disks θ_1, θ_2 and θ_3 are defined as the output y.

A state space realization of the torsional plant is

$$\dot{x} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{k_1}{J_1} & -\frac{d_1}{J_1} & \frac{k_1}{J_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{J_2} & 0 & -\frac{(k_1+k_2)}{J_2} & -\frac{d_2}{J_2} & \frac{k_2}{J_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_2}{J_3} & 0 & -\frac{k_2}{J_3} & -\frac{d_3}{J_3} \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{J_1} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} u \tag{3}$$

Prep. 1.1: Is the system in Eq. (3) controllable and observable?

1.3 Frequency Response

A bode diagram of the open-loop plant is shown in Figure 4. Two resonant peaks can be seen at frequencies, $\omega \approx 18\,\frac{\text{rad}}{\text{s}}$ and $\omega \approx 31\,\frac{\text{rad}}{\text{s}}$, respectively. Thus, a pair of complex poles is to be expected at the corresponding locations. The resonant peaks may induce oscillations in the closed-loop response of the plant; it is desired to suppress such oscillations as much as possible.

Prep. 1.2: What would be the closed-loop bandwidth under feedback with a constant gain of 1, i.e. $K(s) = 1$ in Fig. 6.?

Prep. 1.3: Given the control objectives, what closed-loop bandwidth is needed?

Additionally, one can see a decay of $20\,\frac{\text{dB}}{\text{dec}}$ at the low frequency range; this indicates integral action in the open-loop plant. This is expected as the controlled output is a position signal, which comes from integrating a velocity signal. Therefore, the open-loop plant is marginally stable.

Figure 4 also shows the gain and phase margins of the system.

Prep. 1.4: What are the gain and phase margins of the system?

Prep. 1.5: What can you say about the closed-loop stability by looking at those margins? (under feedback with a constant gain of 1, i.e. $K(s) = 1$ in Fig. 6).

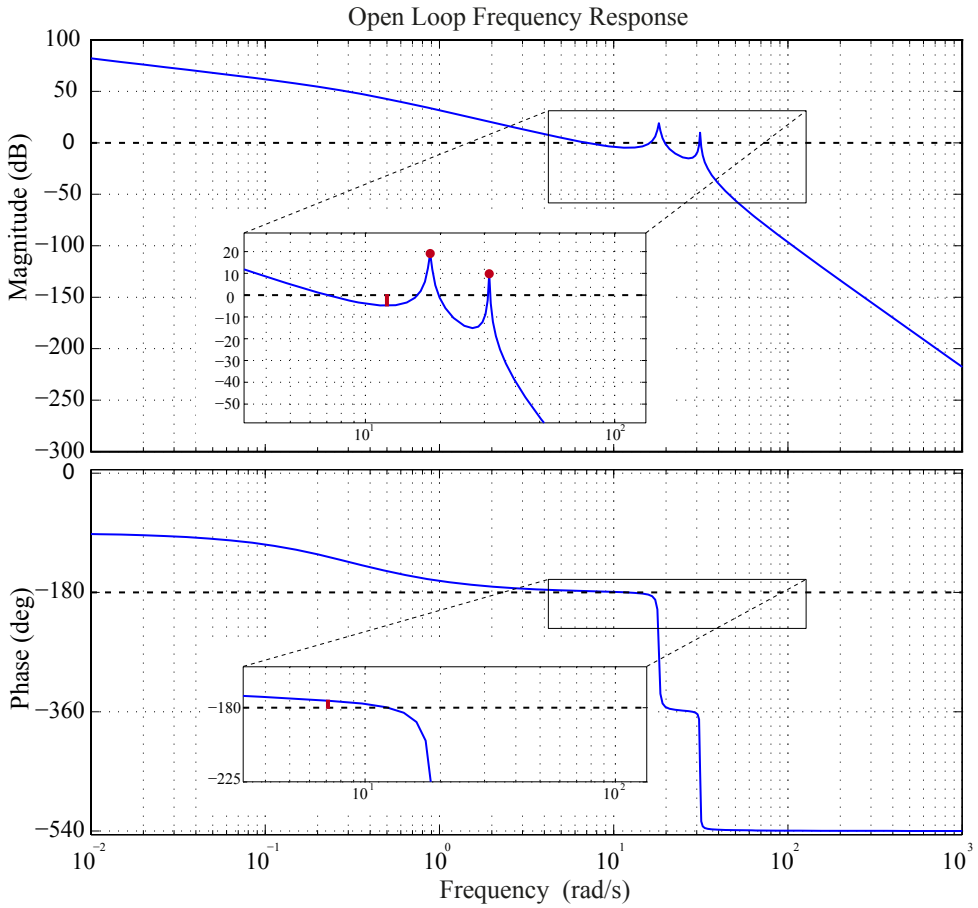

Figure 4: Bode plot of the torsional plant.

1.4 System Hardware Gains

1.4.1 Input

The excitation motor with a toothed belt has a gearing ratio of $k_{\text{pull}} = 3$ and a torque current gain of $k_{\text{t}} = 0.086\,\frac{\text{N m}}{\text{A}}$. It is connected to the output of a DC current amplifier. The amplifier contains a current control loop and receives its input signal as a voltage from the I/O real-time card. A constant amplifier gain of $k_{\text{a}} = 1.5\,\frac{\text{A}}{\text{V}}$ is assumed. The resulting overall hardware gain k_{hw} can be calculated as

$$k_{\text{hw}} = k_{\text{a}} k_{\text{t}} k_{\text{pull}} = 0.387\,\frac{\text{N m}}{\text{V}}$$

which stands for the ratio of the torque over the voltage applied to the system (at the bottom disk).

1.4.2 Output

The angular positions of each disk are measured via incremental encoders. An encoder is a device that produces a train of pulses as it rotates. The hardware gain for the three encoders is the same; it is

$$k_{\text{enc}} = \frac{360}{16000} \ \frac{\text{deg}}{\text{pulses}}$$

which stands for the ratio of a complete revolution of encoder over the generated pulses.

1.5 System Limits

Given the hardware gain and the maximum allowed voltage of $u_{\max} = \pm 10\,\text{V}$, the maximum torque applied to the bottom disk is about $T_{\max} = u_{\max} \cdot k_{\text{hw}} = \pm 3.87\,\text{N m}$. It is recommended to limit the maximum output voltage to $\pm 5\,\text{V}$. To avoid permanent squirm of the springs, the maximum torsional limit for each spring has to be restricted $\pm 25°$. It is also recommended to twist the springs not more than $\pm 20°$. Once the limit is reached, the experiment should stop immediately. To sum up, the following two system limits have to be observed when working with the real plant:

- $u_{\max} \leq \pm 5\,\text{V} \rightarrow T_{\max} \leq \pm 1.94\,\text{N m}$;
- $|\theta_2 - \theta_1| < 20°$ and $|\theta_3 - \theta_2| < 20°$.

2 OUTLINE OF THE LAB PROJECT

Although a linearized dynamic model of the plant has been derived, it still remains to accurately identify the involved physical parameters, i.e. inertia, torsional stiffness and damping coefficients. To estimate the unknown parameters, a grey-box identification technique is applied. Based on the identified model, three different controllers are to be designed and compared: a lead-lag compensator, a standard LQG controller and an LQG controller with integral action. The designed controllers need to be validated by simulation first, then implemented on the real plant. Several criteria are predefined for their performance assessment.

This laboratory experiment consists of the following tasks:

- Estimate the unknown parameters in the linearized model based on grey-box identification
- Design and tune a lead-lag compensator and the two LQG controllers based on the identified model
- Implement the designed controllers on the real plant and compare their performance

3 PARAMETER IDENTIFICATION

All methods for designing and analysing control systems that have been introduced in the course are model based, i.e. it is assumed that a dynamic plant model in the form of a transfer function or state space realization is available. In practice, obtaining such a model can take up a significant part of the time.
An alternative is to obtain a plant model experimentally by measuring the response of the plant to suitable test signals; this approach is known as system identification.
The goal of this task is to accurately identify the system parameters contained in the state space representation of the plant in Eq. (3). A suitable excitation signal is to be designed by the student. This signal will be fed to the plant and by minimizing the least squares error between the actual and the estimated torque, a set of parameters will be obtained. Finally, the identified system will be validated by comparing the actual and the estimated torque.

3.1 Least Square Estimation

To achieve this task, the least square estimation approach is used. The goal of the method is to minimize the sum of squares of the errors:

$$V(p) = \sum_{l=0}^{\infty} e^2(l) = \sum_{l=0}^{\infty} (y^2_{measured} - y^2_{estimated}) \tag{4}$$

between the measured data and the estimated ones, by posing a parameter vector 'p'.
Assume that we are observing a process - characterized by a quantity y(t) - at time instants t = 0, T, 2T, . . . where T is a given sampling time. Then, we can write the equation:

$$y(k) = m_1(k)p_1 + m_2(k)p_2 + ... + m_n(k)p_n + e(k) \tag{5}$$

where the vector: $M = [m_1(k), m_2(k), ...m_n(k)]$ is the vector containing the other variables that affect y(k), such as the inputs u(k), u(k-1), y(k-1), etc. And the vector 'p' is the vector containing the parameters to be estimated.

To drive the error to zero, it is required that:

$$Y = Mp = \underbrace{\begin{bmatrix} m_1(k) & \dots & m_n(k) \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}}_{p} \tag{6}$$

The requirement for this is to have knowledge of the system order, so that the matrix M is filled properly and all parameters are estimated. Hence, assuming M has full rank, we can write:

$$p = (M^T M)^{-1} M^T Y \tag{7}$$

where p is the vector containing the parameters to be estimated, and Y is the vector containing the outputs at every time step.

3.2 Generate Excitation Signals

To collect all the required data, the input signal should be rich enough. This means that the input signal should have the minimum amount of energy to excite all of the system parameters, which is the number of frequencies contained in the signal. This is the notion of Persistent excitation (PE). Mathematically, the PE condition insures that the matrix M is invertable. Band-Limited white noise contains the spectrum of all frequencies, which makes it PE of any order, step input is PE 1, etc.
MATLAB provides various possibilities to generate excitation signals, for example the block Band-Limited White Noise in SIMULINK.
You can also add a low-pass filter to filter out higher frequencies.

Prep. 3.1: What would be a good maximum frequency for the excitation signals?

3.3 Collect a Set of Input-Output Data

The system can be written in the form:

$$\hat{y}(k) = -a_1 y(k-1) ... - a_n y(k-n) + b_1 u(k-1) ... + b_n (k-n) = Mp \tag{8}$$

This is known as the difference equation. Thus measurement data should be retrieved from the plant along with the input signals, which represent the matrix M. The matrix P is then calculated using the least square estimation.
Include your excitation signal, choose a sampling time and run the SIMULINK model **`SysIdent_OpenLoopModel.slx'**. Save the data in workspace as a mat-file. One point needed to be addressed here is that, in off-line simulation, we can simply regard torques generated by the DC motor as the input of the plant u. However, in real-time experiments, torques have to be converted into voltages written to the I/O real-time card through the hardware gain k_{hw}. Do not forget the system limits.

Arrange the measured data in a form that allows to compute the unknown parameters by solving a linear least squares problem (as mentioned above): given the differential equation (1), the excitation signals u and the measured output $\theta_{1,k}$, $\theta_{2,k}$ and $\theta_{3,k}$ at each sampling instant k, the following augmented difference equation contains the input-output information at all sampling instants and has the form:

$$\underbrace{\begin{pmatrix} T_1 \\ 0 \\ 0 \\ \vdots \\ T_k \\ 0 \\ 0 \end{pmatrix}}_{T} = \underbrace{\begin{pmatrix} \ddot{\theta}_{1,1} & 0 & 0 & \dot{\theta}_{1,1} & 0 & 0 & \theta_{1,1}-\theta_{2,1} & 0 \\ 0 & \ddot{\theta}_{2,1} & 0 & 0 & \dot{\theta}_{2,1} & 0 & -\theta_{2,1}-\theta_{1,1} & \theta_{2,1}-\theta_{3,1} \\ 0 & 0 & \ddot{\theta}_{3,1} & 0 & 0 & \dot{\theta}_{3,1} & 0 & \theta_{3,1}-\theta_{2,1} \\ & \vdots & & & \vdots & & \vdots & \vdots \\ \ddot{\theta}_{1,k} & 0 & 0 & \dot{\theta}_{1,k} & 0 & 0 & \theta_{1,k}-\theta_{2,k} & 0 \\ 0 & \ddot{\theta}_{2,k} & 0 & 0 & \dot{\theta}_{2,k} & 0 & -\theta_{2,k}-\theta_{1,k} & \theta_{2,k}-\theta_{3,k} \\ 0 & 0 & \ddot{\theta}_{3,k} & 0 & 0 & \dot{\theta}_{3,k} & 0 & \theta_{3,k}-\theta_{2,k} \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ d_1 \\ d_2 \\ d_3 \\ k_1 \\ k_2 \end{pmatrix}}_{p} \tag{9}$$

where the torque vector T contains the torques applied by the DC motor at the bottom disk. The observation Matrix M contains measured entries $\theta_{i,k}$ and entries calculated from

measurements $\dot{\theta}_{i,k}$ and $\ddot{\theta}_{i,k}$. The vector p contains all the unknown physical parameters to be estimated.

3.4 Fix the m-file

To organize the differential equation in this form after measurements are collected, the m-file **`SysIdent_BuildMeasurementMatrix.m'** is provided, in which a part of the code needs to be completed. The execution of this file occurs inside of **`SysIdent_ParameterEstimation.m'**; it is important to open this function first as it is the main file of this part of the experiment.

To construct the M matrix, the angular velocity $\dot{\theta}_{i,k}$ and the angular acceleration $\ddot{\theta}_{i,k}$ need to be calculated from measured angular position $\theta_{i,k}$ of the i^{th} disk at each sampling time k by numerical differentiation. One can use a central difference approach to calculate the angular velocity as

$$\dot{\theta}_{i,k} = \frac{\theta_{i,k+1} - \theta_{i,k-1}}{2T_s} \tag{10}$$

and the angular acceleration as

$$\ddot{\theta}_{i,k} = \frac{\theta_{i,k+1} + \theta_{i,k-1} - 2\theta_{i,k}}{T_s^2}. \tag{11}$$

When doing this in the experiment, it is necessary to low-pass filter the measured angular positions before using it to calculate velocity (10) and acceleration (11), in order to reduce the influence of measurement noise. The calculated velocity and acceleration need to be filtered as well for identification. In the given m-file, parts of the code are missing. They are required to be filled in before submission. The missing parts are:

1 The calculation of velocity and acceleration according to (10) and (11) by means of central difference.

2 Signal filtering: a second order Butterworth low pass filter is recommended. In this experiment, the same filter can be employed to filter the angular position, velocity and acceleration. You can use the following commands

```
[B,A] = butter(N,Wn,'low')
    Y = filtfilt(B, A, X)
```

3.5 Identification of Unknown Parameters

The objective of the identification is to estimate the unknown parameter vector p, so that the difference between the given excitation, which is the torque vector T applied to the bottom disk, and estimated torque $\hat{T}(p)$ is minimized (see Exercise 7.1, of the lecture notes *Control Systems Theory and Design*):

$$\min_p \sum_{i=1}^{n} (\hat{T}(p) - T)^2 \tag{12}$$

Starting with (9), where

$$T = Mp\,, \tag{13}$$

use the approach discussed in the lecture notes *Control Systems Theory and Design*, Chapter 7, to derive a formula for the solution to (12) and write the code for the estimation of p as part of the function **`SysIdent_ParameterEstimation.m'**.

3.6 State Space Realization

By replacing the unknown physical parameters in Equation (3) with the identified values, the state space model of the linearized system can be generated with the given m-file **`SysIdent_StateSpaceConstruction.m'**. Save the state space model, as it will be used for the controller designs.

3.7 Validate the Identified Model

To examine the accuracy of the identified model, you can excite the SIMULINK model with excitation signals that have not been used in the identification experiment, and compare the given torque T and estimated torque $\hat{T}(p) = Mp$, where T and M are generated by the new signals, whereas p is the original identified parameter vector.

To assess the accuracy, compute the sum of the squared errors of those signals (see Figure 5). This part must be completed by the students as part of the function **`SysIdent_ParameterEstimation.m'**.

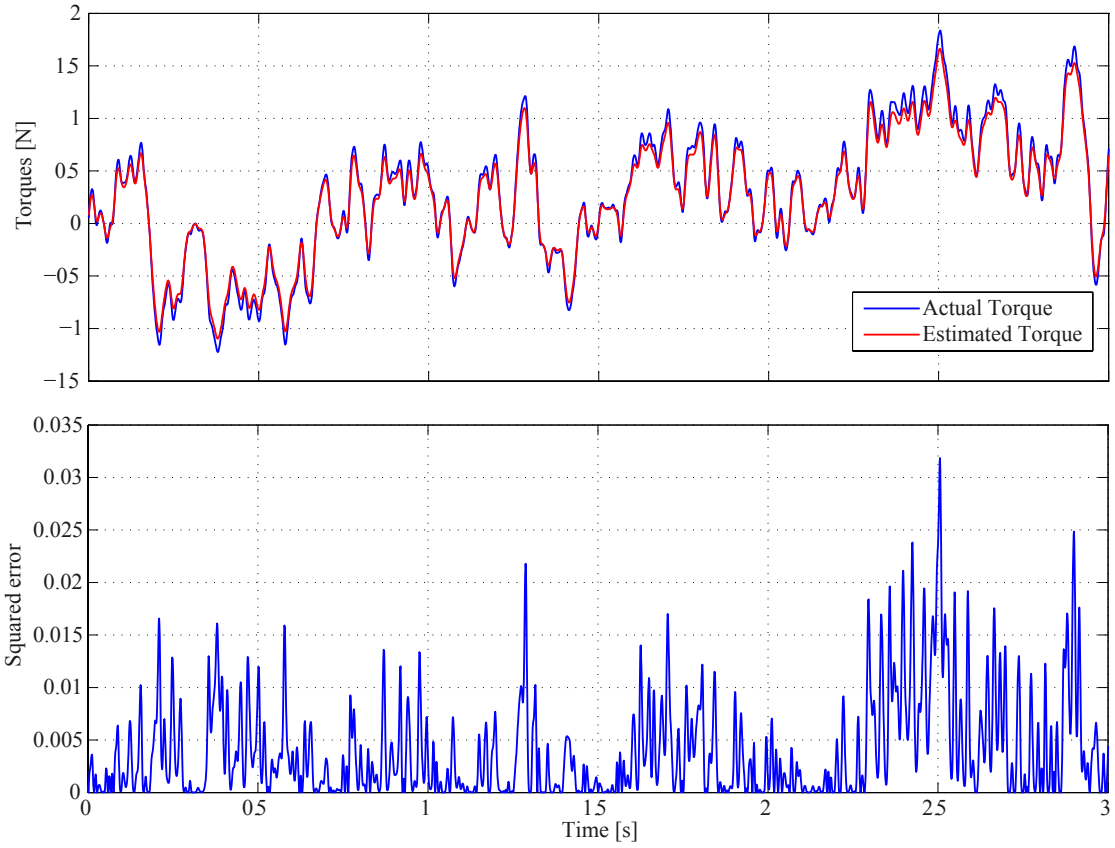

Figure 5: System identification validation.

4 CONTROLLER DESIGN

In this section, the identified model will be used to design three different controllers: a lead-lag compensator and two LQG regulators (without and with integral action).

The controllers must satisfy as close as possible the design specifications described in Section 1.1. At the same time, it has to be guaranteed that the controller does not exceed the system limits. Special attention needs to be paid to the maximum twist angle of the springs to avoid a permanent damage.

4.1 Lead-Lag Compensator

First a classical lead-lag compensator is to be designed. By adding poles and zeros to the open-loop system, the lead and lag compensators influence the phase and gain at

different frequencies, leading to better transient behaviour and steady state accuracy. The control loop has the structure shown in Figure 6.

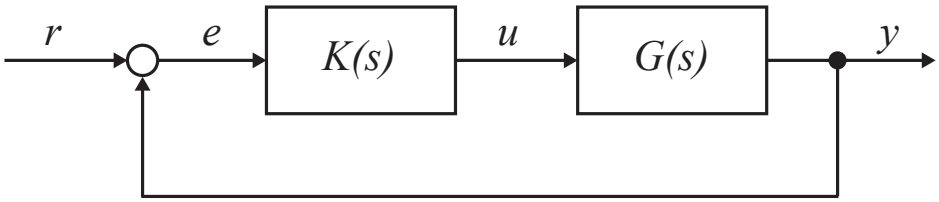

Figure 6: Closed-loop system.

Here $G(s)$ is the plant model and $K(s)$ the controller to be designed. The signal r is the angular reference in [deg], u is the control input in [Nm], y is the angular position of the third disk, i.e. θ_3.

Prep. 4.1: What are the different control objectives of a lead and a lag compensators?

Prep. 4.2: How are the relative pole/zero locations of lead and lag compensators?

Follow the next steps to design your lead-lag compensator.

1 Calculate the transfer function from the control input u to the third disk output θ_3 in the open-loop system. Use the information of its Bode diagram to design a phase lead compensator to achieve a phase margin larger than 30 deg with a closed-loop bandwidth of $\omega_b > 1$ rad/s.

2 Design a lag compensator to improve the performance (e.g. high-frequency suppression) of the closed-loop system already designed, without significantly changing the behavior achieved by the designed lead compensator.

3 For fine-tuning of the lead-lag compensator use the MATLAB Toolbox *SISO tool* to achieve the performance criteria.

The code for designing your lead-lag compensator must be included in the file **`LeadLag_Design.m'**.

5 Test the performance of the designed lead-lag compensator with a step reference input using the SIMULINK model **`LeadLag_Sim.slx'**. Before you run the simulation, make sure the designed compensator is written to the corresponding block, and the previously identified state space model of the plant exists in the Workspace.

In Figure 9, a typical step response is shown, for a reference command of $r = 360$ deg for θ_3, using a lead-lag compensator.

4.1.1 Loop Shaping Approach

As mentioned before, some guidelines to design a lead-lag compensator can be found in the *Introduction to Contol* lecture notes in Chapter 4. Those can be useful as a starting point to iterate your design. For a broader perspective on how to tune your lead-lag compensator, e.g. with *SISO tool*, or to design your lead-lag compensator differently, the idea of loop shaping is now briefly reviewed.

Loop shaping is a classical approach to design controllers for SISO systems. The idea is to *shape* the open loop transfer function to satisfy some restrictions. As a graphical method, it is very intuitive. It allows to infer the overall performance of the closed-loop system from the frequency response of the open loop.

Consider the loop shown in Figure 6 and define $L(s) = G(s)K(s)$. In Figure 7, three frequency regions are defined: performance region, transition region and robustness region. Figure 7 shows a desired shape for the open loop gain $L(s)$. Basically, it is desired to have high gains at low frequencies (e.g. to obtain integral action) and a rapid decay at high frequencies (e.g. to reduce the effect of high-frequency noise or possible model uncertainties). In the center region (bandwidth), the active limits of operation of the system are determined. In this region, stability must be ensured, therefore the need of a sufficient phase margin at the crossover frequency. A sustained 20 dB/dec roll-off achieves a phase margin of 90 deg and is hence desired (see Section 4.4, *Bode's Gain-Phase Theorem*, in the lecture notes 'Introduction to Control Systems'). Recall that the phase margin of a system is related to its damping properties, i.e. a large phase margin implies a good damping, i.e. good transient behaviour.

A first step is always to establish the bandwidth of the system. One can employ a lead compensator to adjust the slope of the system at the transition region; thus, the phase margin. A lag compensator can be used to increase the gain at low frequencies or to decrease the gain at high frequencies. It is important to mention that, for minimum phase systems, a loop shaping approach requires only to look at the Bode magnitude plot.

You can tune your lead-lag compensator by keeping in mind the shape of a desired

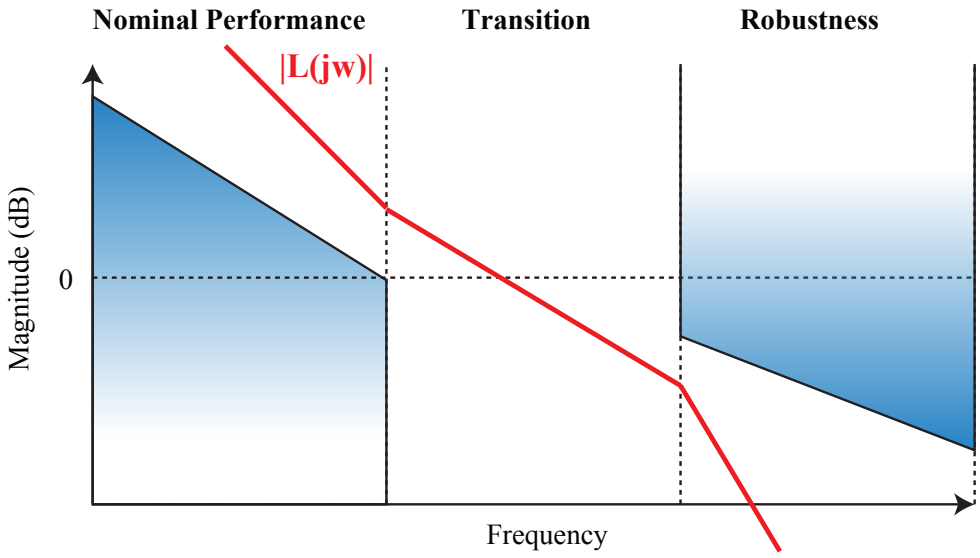

Figure 7: Loop shaping restrictions.

open-loop gain $L(s)$, or you can design a compensator by successively adding different stages (e.g. leads, lags, filters, notches etc.) to your plant $G(s)$ in order to shape $L(s)$.

In the following it is described how to design a lead-lag compensator by employing the loop shaping approach. MATLAB code for the design of your lead-lag compensator must be included in the file **`LeadLag_Design.m'**.

1 From the open-loop plant Bode diagram, find the gain that achieves the desired bandwidth.

2 Design a lead compensator to adjust the slope one decade above and below the crossover frequency to 20 dB/dec.

3 Design two notch filters to reduce the resonance of both peaks. The frequency region of the peaks is close to the crossover frequency, be careful to not change the bandwidth of the system.

A notch filter $F(s)$ has a pure complex zero pair at ω_N that defines the notch and two poles to have a proper filter and 0 dB gain everywhere else except at ω_N, i.e.

$$F(s) = \frac{(s + j\omega_N)(s - j\omega_N)}{s^2 + q\omega_N s + \omega_N^2}, \tag{14}$$

where q defines the amplitude of the notch. Give a try to the following code to feel familiar with the behaviour of a notch filter.

```
%A Bode plot of a Notch filter at 3 rad/s is shown
  wn = 3;
  q = 0.5;
  F = tf([1 0 wn^2],[1 q*wn wn^2]);
  bode(F)
```

4 Test your design; if required increase the phase gain of the lead compensator to increase a possible phase loss due to the lag compensator.

In Figure 8 one can see the adjustments that are performed on the open-loop (in blue) frequency response with the inclusion of the compensators listed above.

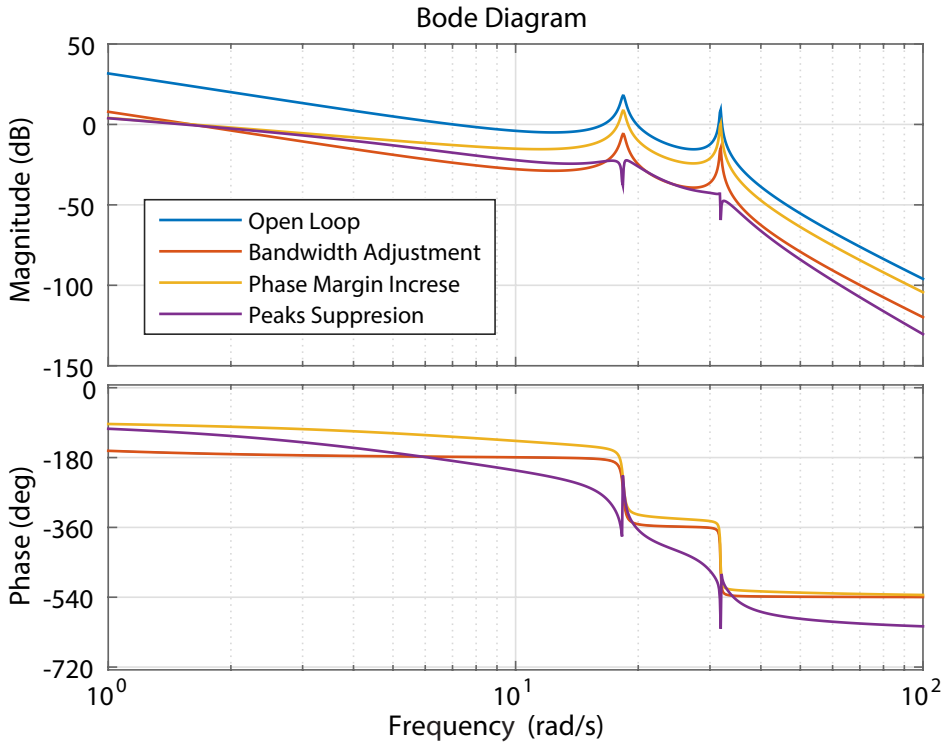

Figure 8: Loop shaping procedure.

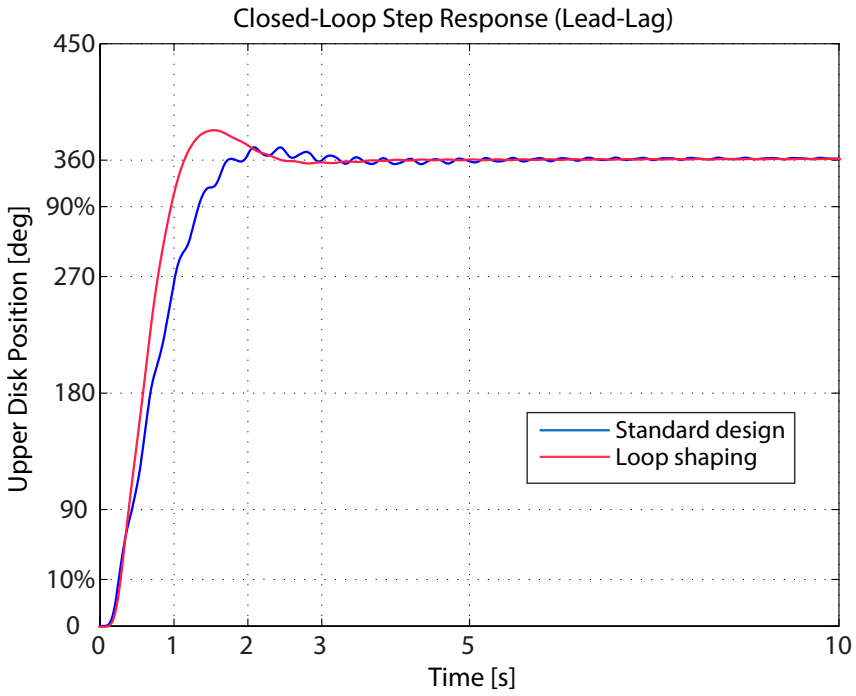

Figure 9: A typical step response of the closed-loop system with two lead-lag compensators.

4.2 LQG Controller

For comparison with the classical lead-lag compensator, an LQG controller is to be designed in this experiment. In this scheme, only the third output ($y = \theta_3$) is used for feedback. Figure 10 shows the simplified control diagram. It combines a linear quadratic estimator (LQE) and a linear quadratic regulator (LQR). Moreover, a static prefilter (constant gain v) is added to loop.

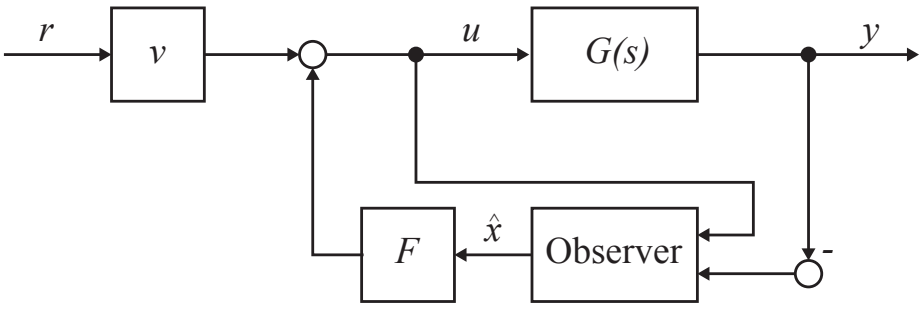

Figure 10: LQG control structure with a prefilter.

Prep. 4.3: What are the trade-offs to consider when an LQG controller is to be designed, and how are those trade-offs tuned?

Follow the next steps to design an LQG controller.

1 In the m-file **`LQG_Design.m'**, write the MATLAB code for the design of an LQG controller with a static prefilter v.

- Determine the prefilter v.
- Determine the estimator gain vector L and the state feedback gain F with the MATLAB command `lqr()`.
- Tune the weighting matrices so that the tracking specifications are satisfied.

2 Test your designed LQG controller by implementing it in the SIMULINK model **`LQG_Sim.slx'**. Use the same input reference as for the lead-lag compensator; compare the closed-loop response of the two controllers.

Figure 11 shows a typical the step response for a reference command of $r = 360$ [deg], using an LQG controller. Here the, performance of the LQG controller seems to be slightly better.

Prep. 4.4: What are the possible reasons for a better performance of an LQG controller compared to the lead-lag controller?

Obtain the closed-loop Bode plots for both designs (lead-lag and LQG) and compare.

4.3 LQG Controller with Integral Action

Introduce the step disturbances by changing the corresponding variable in the simulations of both the lead-lag and the LQG design. You should observe a behaviour similar to what is shown in Figure 12. The first disturbance ($t = 4$s) is an output disturbance. Due to the integral action of the plant, the output tracks the reference signal. The second disturbance ($t = 7$s) is an input disturbance, both controllers are severely affected. Thus the need for a new design.

To completely bring to zero any steady state error and to cope with disturbances, the designed LQG controller has to be extended. To achieve this, integral action must be

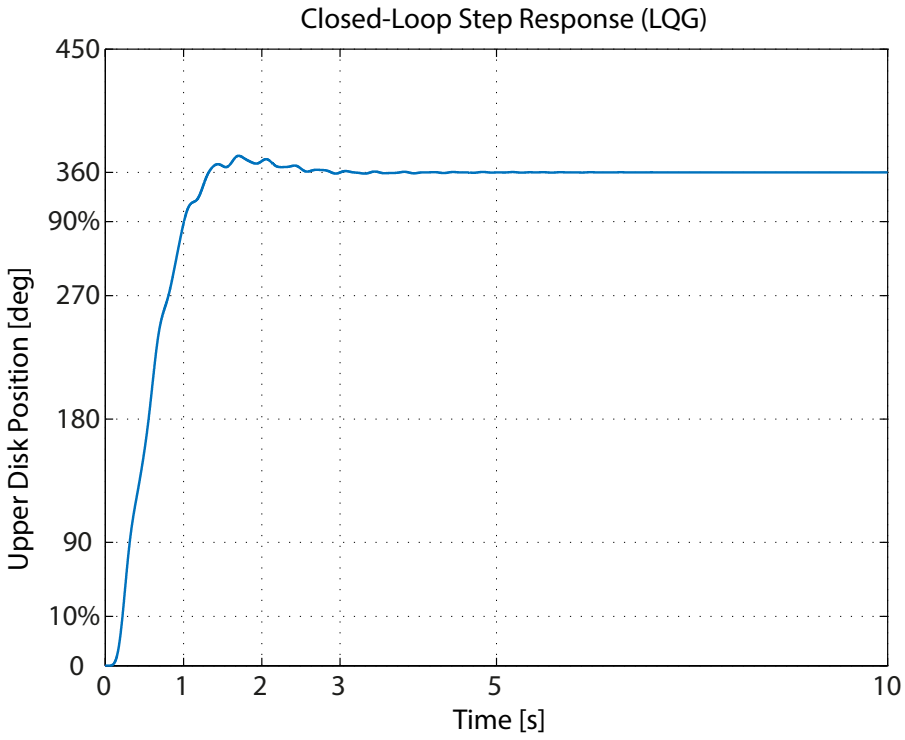

Figure 11: Step response of the closed-loop system with an LQG controller.

included in the LQG design. For this, the error signal must be integrated, i.e. a direct measurement of the error will take place (see Figure 13).

Refer to the Problem 5.15 in the lecture notes *'Control Systems Theory and Design'* for details.

Prep. 4.5: Why does the steady state error vanish when integral action is included?

Prep. 4.6: What modifications to the plant are required to design an LQG controller when integral action is considered in the loop?

Follow the next steps to design an LQG controller with integral action.

1 Construct the corresponding augmented system.

2 In the m-file **`LQG_Design.m'** write the MATLAB code to design an LQG controller with integral action.

- Determine the new state feedback gain F with the MATLAB command `lqr()`.

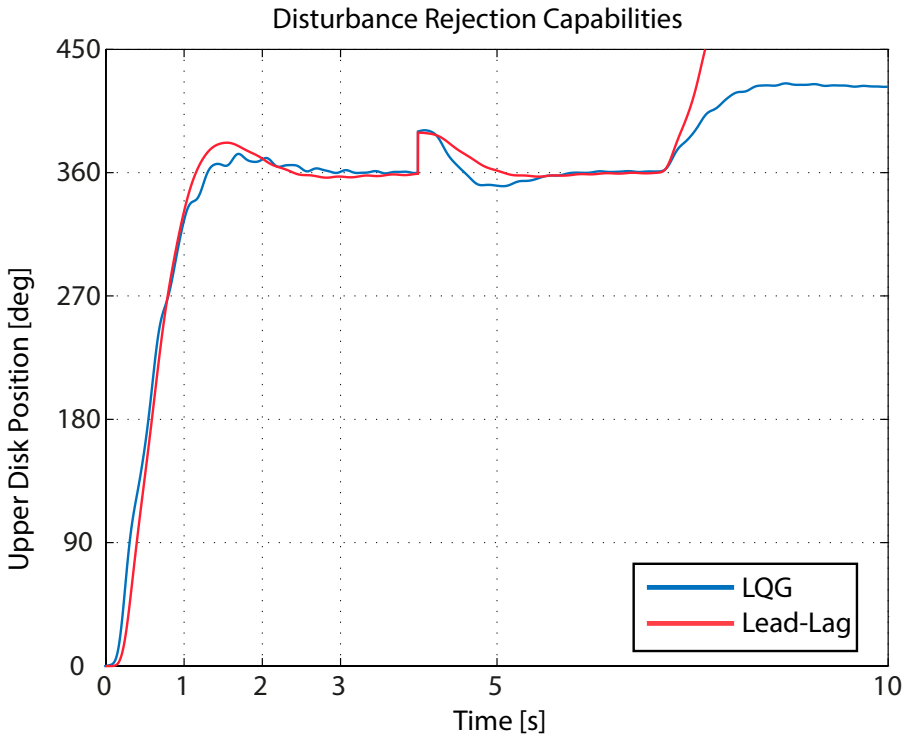

Figure 12: Behaviour of output and input step disturbances with a lead-lag compensator and an LQG controller.

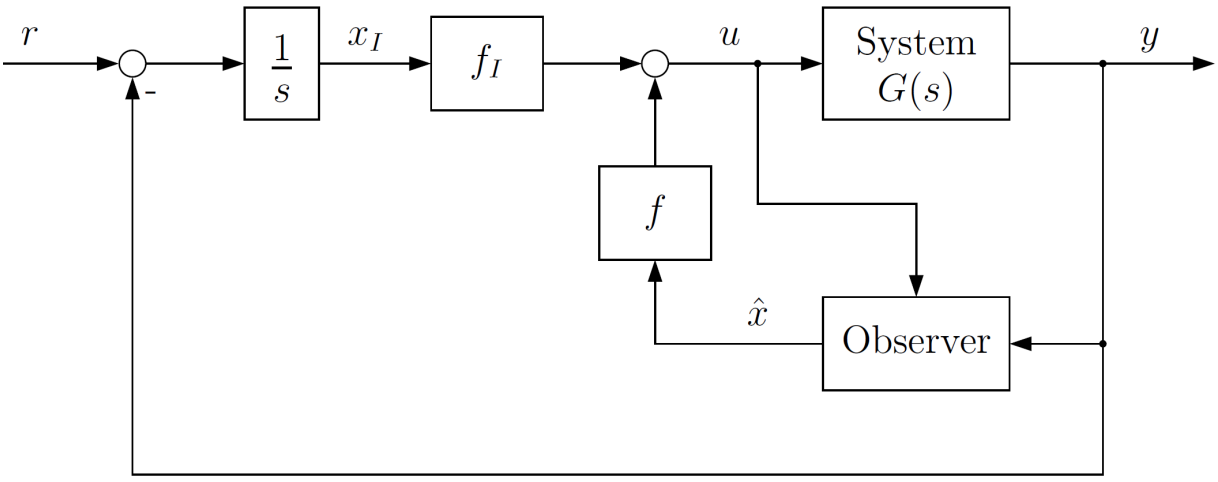

Figure 13: LQG control structure with integral action.

- Tune the weighting matrices so that the tracking specifications are satisfied.

3 Test your newly designed LQG controller by implementing it in the SIMULINK

model `LQG_Sim.slx'. Use the same input reference and disturbances as before, compare the closed-loop response of all the controllers (see Figure 14). Notice how the input disturbance is suppresed almost instantaneously.

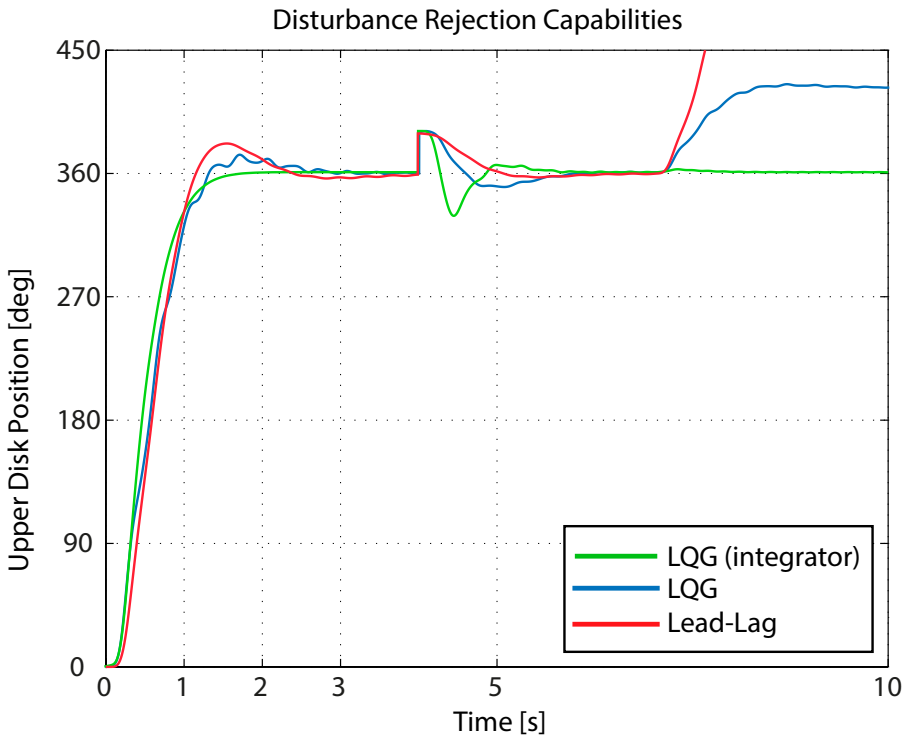

Figure 14: All designed controllers.

Notice also the rejection to high-frequency resonance. Plot the closed-loop Bode diagram of this design and compare to the previous controllers.

5 EXPERIMENT

Before working with the real plant, students should prepare themselves by going through the preliminary tasks and carefully reading the safety instructions contained in the next section. The main tasks in this section are similar to the preliminary ones, but the object is now the real plant. All MATLAB m-files and SIMULINK models needed for the real-time experiment are saved in the project computer. The real-time system employed in this torsional plant is the TwinCAT automation suite provided by Beckhoff. In order to do this one must click on 'Activate Configuration' in the visual studio file for the torsional plant - *TorsionalPlant.sln*.

The following tasks are to be fulfilled in the laboratory:

1. For the parameter estimation part of the experiment, use the provided SIMULINK model **`OpenLoopModel_Exp.slx'**. Add appropriate excitation signals to the input and choose the sampling time $T_s = 1$ ms. Check that the conversion between torque and voltage is correctly implemented. Make sure that the excitation signals do not exceed the system limits. The block *Analog Output* works in real time and connects the user's input to the I/O real-time card. Another real-time block *Encoder Input* reads back the measurements from the encoders. To run the SIMULINK model, click the *Play* button; it will automatically run on the real system after compilation of the code.

2. Collect the excitation signals and measured outputs. Estimate the parameters by solving a linear least squares problem.

3. Construct the A, B, C and D matrices of the state space model and validate the estimated parameters by generating different input-output data and comparing the response of the identified model with that of the real plant.

4. Tune your lead-lag compensator based on the on-line identified model, so that the tracking specifications are fulfilled as much as possible for the new identified system.

   NOTE: A more accurate representation of the torsional plant would include a friction model. Therefore, controllers with low input signals might show a poorer performance in the real system than in simulations. Make sure your controllers

has high gain at low frequencies high control action to avoid this issue as good as possible.

5. Open the SIMULINK model **`LeadLag_Exp.slx'** and write the designed compensator in the controller block. Save the measurements as mat-file for later comparison.

6. Tune your designed LQG controllers based on the on-line identified model, so that the tracking specifications are fullfilled.

7. Use the given SIMULINK model **`LQG_Exp.slx'**, for the standard LQG design and for the LQG controller with integral action. Compare the results with the lead-lag compensator.

8. Change the plant dynamics by adding two masses to the upper disk to test the robustness of the two controllers. Discuss the results of the comparison.

5.1 Safety Instructions

Before working with the real plant, please read and follow the instructions below carefully to prevent damages and accidents:

- **Before powering up** the ECP amplifier, check that all moveable parts (disks and additional weights) are fixed in a proper way and all screws are tightened.
- Check whether the red **emergency stop button** is connected to the amplifier.
- **Do not touch the plant directly** if the amplifier is powered up.
- Take care about your **hair and cloths** (e. g. ties, scarves, etc.). Avoid to be caught in the rotating apparatus.
- **Do not uncover** or touch any interior parts of the amplifier or PC until the power plugs have been disabled.
- Switch off the power plugs when **switching fuses**.

A motionless system is **not** sufficient to **assume** that it will **keep motionless**. To prevent unwanted and uncontrolled movements of the apparatus please follow the following steps during the process of experiments:

1. Always boot up the PC first.
2. Start *TorsionalPlant.sln*-file in Visual Studio and Activate Configuration.
3. Start MATLAB SIMULINK and load a given SIMULINK model.
4. Check if the **emergency stop simulink-block** is connected and set up in a proper way.
5. To prevent a spring twist angle more than ±20 deg, check if the boundary in the **saturation simulink-block** is less than 5 V or 1.94 Nm.
6. Turn on the amplifier for the experiments.
7. Run the SIMULINK model to run the experiment.
8. Switch off the amplifier after the experiment.
9. After finishing all experiments, switch off the PC and the amplifier to set the I/O Real-Time Card default output value to zero.

If the plant moves out of control, which leads to a **permanent spring squirm**, press the red emergency button immediately.