Automatic Control Laboratory, ETH Zürich
Profs. M. Morari, J. Lygeros

Manual prepared by: P. Brunner, F. Ullmann, S. Richter, C. Fischer
Revision from: February 16, 2013

IfA Fachpraktikum - Experiment 3.7A :

Flexible Shaft A

Comment: This experiment is the first in a series of two experiments. Here we will

- analyze the system and
- design a PI controller.

The experiment can be continued with experiment 3.7B, where two more controllers will be designed.

Based on experiment 3.7, the characteristics of so-called flexible structures are introduced. Generally speaking, flexible structures are mechanical systems which can be considered and modeled as many mass points connected by elastic springs, as for instance robots and large satellite dishes. As a simple physical model of such a flexible structure, we are using a shaft consisting of two flywheels connected with a torsional spring. A drive motor accelerates this flexible shaft whereas a load motor is used to simulate a load torque. The goal is to control the speed of the shaft.

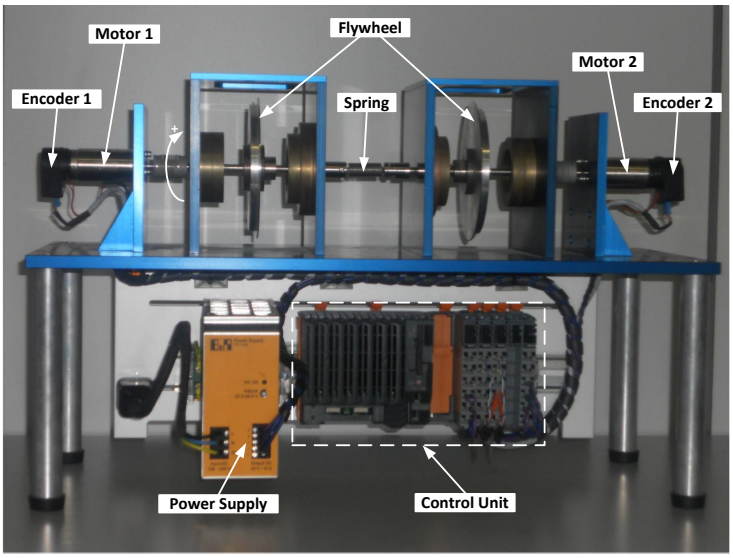

Figure 1: Hardware Setup

Experiment 3.7A is dedicated to the analysis of a continuous-time state space model, and a first feedback controller design. Based on the model, typical phenomena like the antiresonance and resonance behavior of such flexible structures will be investigated. Afterwards, the state space model will be verified using a frequency response method. Finally, a *PI controller* will be designed. It will be shown that PI controllers yield unsatisfactory results on both reference tracking and disturbance rejection due to the resonance and antiresonance behavior of the flexible shaft.

For the preparation of this experiment at home you need to download the following files from the IfA Fachpraktikum website `http://control.ee.ethz.ch/~ifa-fp/` (provided as a zip-file on the individual experiment's page):

`updateVarParamFlexShaft.m`	MATLAB script for the controller design. Here you have to insert your solutions to all the tasks; in the other files no modifications are required.
`simFlexShaft.mdl`	SIMULINK model of the flexible shaft used to measure step responses.
`continuousTimeSsModelFlexShaft.m`	Continuous-time state space model.
`ConstantParameterFlexibleShaft`	Folder with MATLAB functions containing constant parameters for the SIMULINK model.
`initFcnFlexShaft.m`	MATLAB script to initialize the SIMULINK model.
`stopFcnFlexShaft.m`	MATLAB script to display the data recorded by the SIMULINK model.

During the lab session you will need additional files that are provided locally on your machine.

Remark:

- The tasks in Chapter 2 have to be completed before the lab session.
- The tasks in Chapter 3 are covered during the lab session.

Contents

Chapter 1

Problem Setup and Notation

1.1 Setup

Figure 1 shows the hardware setup for this lab experiment. The flexible shaft consists of a shaft with two flywheels that are connected by a torsional spring. At both ends of the shaft a brushed DC motor is mounted. Motor 1 is used to manipulate the shaft, whereas motor 2 is used to disturb the shaft by simulating a load torque. Both motor angles can be measured independently by an incremental encoder, although in the present experiment only encoder 1 is employed. The angular velocity of motor 1 is measured indirectly by differentiating the corresponding angle.

The flexible shaft is controlled by a real time application running on the control unit. The control unit consists of (from left to right in Figure 1) a CPU module, two counter modules and a motor module, and works with 24V DC voltage provided by the power supply. The motor module generates a PWM signal for each motor; for motor 2 we use the available current controller mode. The real-time application was built from a SIMULINK model. The same model also serves as a interface to interact with the control unit from the lab PC over Ethernet.

1.2 Modeling

In this section, the modeling of the flexible shaft is presented. It is not necessary to understand the modeling in detail, although this enhances the overall comprehension. Figure 1.1 shows the schematic for the flexible shaft including the two brushed DC motors. Table 1.1 introduces the used notation.

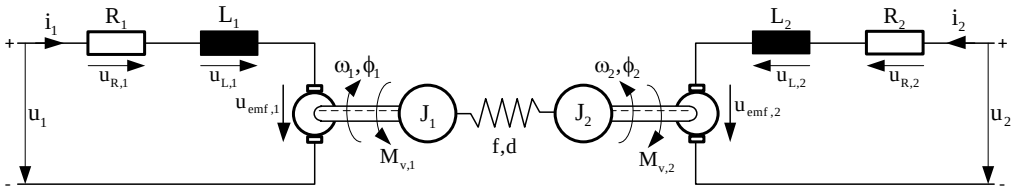

Figure 1.1: Schematic of Flexible Shaft

Notation	Meaning	Unit
u_j	motor input voltage	V
i_j	motor current	A
R_j	motor resistance	Ω
L_j	motor inductance	H = Vs/A
$u_{\text{emf},j}$	back-emf voltage (electro-motive force)	V
$K_{u,j}$	back-emf constant	Vs/rad
$M_{t,j}$	motor torque	Nm
$K_{t,j}$	torque constant	Nm/A
J_j	mass inertia of flywheel and shaft	kgm^2
ω_j	angular velocity	rad/s
$\omega_{m,j}$	measured angular velocity	rad/s
ϕ_j	angle	rad
$\Delta\phi$	angular difference	rad
$M_{v,j}$	viscous friction in bearings	Nm
$K_{v,j}$	coefficient of viscous friction	Nms/rad
f	spring constant	Nm/rad
d	damping coefficient of spring	Nms/rad

Table 1.1: Notation. Note that $j \in \{1, 2\}$.

Mechanical Part

To get the equations of the mechanical part of the flexible shaft, the principle of angular momentum is used. In terms of friction we only consider the viscous friction given as

$$M_{v,1}(t) = K_{v,1} \cdot w_1(t) \ , \quad M_{v,2}(t) = K_{v,2} \cdot w_2(t).$$

This yields the following differential equations for the flywheels and the angles:

$$J_1 \cdot \frac{d}{dt}\omega_1(t) = -f \cdot (\phi_1(t) - \phi_2(t)) - d \cdot (\omega_1(t) - \omega_2(t)) - K_{v,1} \cdot \omega_1(t) + M_{t,1}(t) \tag{1.1a}$$

$$J_2 \cdot \frac{d}{dt}\omega_2(t) = f \cdot (\phi_1(t) - \phi_2(t)) + d \cdot (\omega_1(t) - \omega_2(t)) - K_{v,2} \cdot \omega_2(t) + M_{t,2}(t) \tag{1.1b}$$

$$\frac{d}{dt}\phi_1(t) = \omega_1(t) \tag{1.1c}$$

$$\frac{d}{dt}\phi_2(t) = \omega_2(t) \tag{1.1d}$$

This is obviously a fourth-order system (with states ω_1, ω_2, ϕ_1, ϕ_2). But if we look more closely at the mechanical part of our plant, then we actually see only three energy mass storages present in the system:

1. The right flywheel storing kinetic energy.
2. The torsional spring storing potential energy.
3. The left flywheel storing kinetic energy.

Therefore, there must be an algebraic relation which reduces our system to a third-order system. Examining the differential equations (1.1a) and (1.1b), we see that we can substitute the angular difference

$$\Delta\phi(t) = \phi_1(t) - \phi_2(t), \tag{1.2}$$

which replaces the absolute angles ϕ_1, ϕ_2 as states. Combining (1.1c) and (1.1d) then leads to

$$\frac{d}{dt}\Delta\phi(t) = \frac{d}{dt}(\phi_1(t) - \phi_2(t)) = \omega_1(t) - \omega_2(t). \tag{1.3}$$

From here on we assume that the spring damping is neglectable and that both flywheels (including the shaft they are mounted on) are symmetric, i.e.

$$\begin{aligned} d &= 0 \\ J &= J_1 = J_2 \\ K_v &= K_{v,1} = K_{v,2}. \end{aligned}$$

Finally, we obtain the following differential equations for the mechanical part:

$$J \cdot \frac{d}{dt}\omega_1(t) = -f \cdot \Delta\phi(t) - K_v \cdot \omega_1(t) + M_{t,1}(t) \tag{1.4a}$$

$$J \cdot \frac{d}{dt}\omega_2(t) = f \cdot \Delta\phi(t) - K_v \cdot \omega_2(t) + M_{t,2}(t) \tag{1.4b}$$

$$\frac{d}{dt}\Delta\phi(t) = \omega_1(t) - \omega_2(t) \tag{1.4c}$$

Electrical Part

We assume that the parameters of both motors are equal, i.e.

$$\begin{aligned} R &= R_1 = R_2 \\ L &= L_1 = L_2 \\ K_u &= K_{u,1} = K_{u,2} \\ K_t &= K_{t,1} = K_{t,2}. \end{aligned}$$

Motor 1. As we are using permanent magnet excited brushed DC motors, the following general relations hold:

$$M_{t,1}(t) = K_t \cdot i_1(t) \tag{1.5}$$

$$u_{\text{emf},1}(t) = K_u \cdot \omega_1(t) \tag{1.6}$$

Starting with Kirchhoff's second law we have

$$u_{L,1}(t) = u_1(t) - u_{R,1}(t) - u_{\text{emf},1}(t)$$

and derive the following differential equation for the current:

$$L \cdot \frac{d}{dt} i_1(t) = u_1(t) - R \cdot i_1(t) - K_u \cdot \omega_1(t). \tag{1.7}$$

Figure 1.2 shows the block diagram for motor 1 resulting from (1.5) and (1.7).

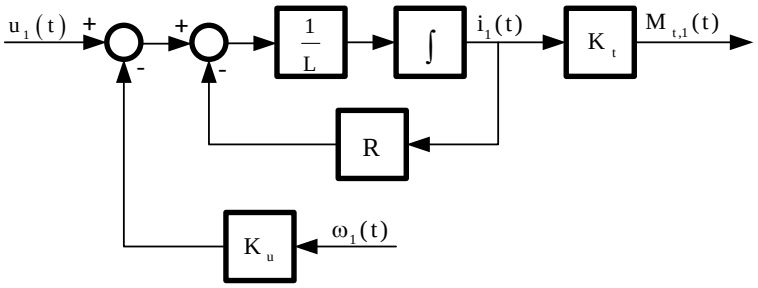

Figure 1.2: Block Diagram of Motor 1

Assuming that the motor current dynamics can be neglected compared to the dynamics of the mechanical system, we set $\frac{d}{dt} i_1(t) = 0$ in (1.7) and obtain for the current

$$i_1(t) = \frac{1}{R} \cdot \left(u_1(t) - K_u \cdot \omega_1(t) \right),$$

which leads by inserting into (1.5) to the algebraic relation

$$M_{t,1}(t) = \frac{K_t}{R} \cdot \left(u_1(t) - K_u \cdot \omega_1(t) \right). \tag{1.8}$$

Motor 2. For motor 2, the same equations as for motor 1 would apply. However, as motor 2 is current-controlled by the motor module of the control unit and we assume that this current controller is ideal, we only use the algebraic relation

$$M_{t,2}(t) = K_t \cdot i_2(t). \tag{1.9}$$

Measurement

The angular velocity $\omega_1(t)$ is measured indirectly by differentiating the angle $\phi_1(t)$. As the angle is measured with an incremental encoder with limited resolution, the velocity measurement carries noise and must therefore be lowpass filtered. We assume that this can be approximated (in the frequency range of interest) by a first-order lowpass filter

$$\frac{d}{dt} \omega_{m,1}(t) = \frac{1}{T_1} \cdot \left(\omega_1(t) - \omega_{m,1}(t) \right), \tag{1.10}$$

where $\omega_{m,1}(t)$ is the measured angular velocity.

Complete Model

With the definitions

$$\begin{aligned} x(t) &\triangleq \begin{bmatrix} \omega_{m,1}(t) & \omega_1(t) & \omega_2(t) & \Delta\phi(t) \end{bmatrix}^T \\ w(t) &\triangleq \begin{bmatrix} u(t) & z(t) \end{bmatrix}^T \triangleq \begin{bmatrix} u_1(t) & i_2(t) \end{bmatrix}^T \\ y(t) &\triangleq \omega_{m,1}(t) \end{aligned}$$

and by using the equations (1.4), (1.8), (1.9) and (1.10), the **state space model** of the flexible shaft becomes:

$$\begin{aligned} \frac{d}{dt} x(t) &= \underbrace{\begin{bmatrix} -\frac{1}{T_1} & \frac{1}{T_1} & 0 & 0 \\ 0 & -\frac{1}{J}\left(K_v + \frac{K_t \cdot K_u}{R}\right) & 0 & -\frac{f}{J} \\ 0 & 0 & -\frac{K_v}{J} & \frac{f}{J} \\ 0 & 1 & -1 & 0 \end{bmatrix}}_{=A_c} \cdot x(t) + \underbrace{\begin{bmatrix} 0 & 0 \\ \frac{K_t}{J \cdot R} & 0 \\ 0 & \frac{K_t}{J} \\ 0 & 0 \end{bmatrix}}_{\substack{=[B_{c,u} \;\; B_{c,z}] \\ =B_c}} \cdot \underbrace{\begin{bmatrix} u(t) \\ z(t) \end{bmatrix}}_{=w(t)} \\ y(t) &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}}_{=C_c} \cdot x(t) + \underbrace{\begin{bmatrix} 0 & 0 \end{bmatrix}}_{\substack{=[D_{c,u} \;\; D_{c,z}] \\ =D_c}} \cdot w(t) \; . \end{aligned} \tag{1.11}$$

This model is illustrated in Figure 1.3.

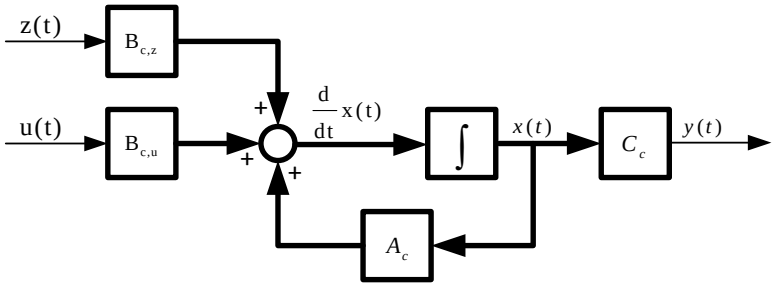

Figure 1.3: Block Diagram of Continuous-Time State Space Model

For a better understanding, the inputs and outputs of the system are discussed again:

- $u(t)$ denotes the manipulating input, i.e. the voltage of motor 1.
- $z(t)$ denotes the disturbance input, i.e. the current of motor 2.
- $y(t)$ denotes the output, i.e. the measured angular velocity of motor 1.

The state space representation (1.11) can be used to derive the transfer function models. The **input-output transfer function** $G(s)$ can be computed as

$$\frac{Y(s)}{U(s)} = G(s) = C_c \cdot (s \cdot I - A_c)^{-1} \cdot B_{c,u} + D_{c,u} \ , \tag{1.12}$$

and the **disturbance transfer function** $G_z(s)$ as

$$\frac{Y(s)}{Z(s)} = G_z(s) = C_c \cdot (s \cdot I - A_c)^{-1} \cdot B_{c,z} + D_{c,z} \ . \tag{1.13}$$

This finally yields the block diagram shown in Figure 1.4.

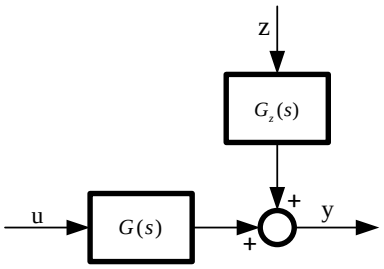

Figure 1.4: Block Diagram of Continuous-Time Transfer Function Model

Chapter 2

Preparation@Home A

In this exercise, we first compute the continuous-time transfer function models and analyze them step by step using, among other things, the Bode diagram. Finally, we are familiar with the flexible shaft and therefore in position to design a first feedback controller, namely a PI controller.

2.1 Instructions

1. The exercises of this chapter must be completed before the lab session of Chapter 2.
2. Download the provided zip-file from the IfA homepage and unpack it to your preferred folder. Then, start MATLAB and go to this folder.
3. Open the script `updateVarParamFlexShaft.m`. Make sure that the following values are set: `modeSel=0`, `cmd=1`, `J=402e-6` (in `CONFIGURATION`).
4. Now you are ready to solve the tasks given in the next section of this chapter step by step. Write your code directly into `updateVarParamFlexShaft.m` at the marked positions. In this file, the continuous-time state space model is available in the structure `syscr`.
5. In some tasks you should run the simulator to measure some responses. In this case, proceed as follows:
   (a) Open the SIMULINK model `simFlexShaft.mdl`.
   (b) Configure `updateVarParamFlexShaft.m` according to the given task, i.e. set the function selector `fcnSel` and the controller selector `controllerSel` to the appropriate value (in `CONFIGURATION`). Then, save the file.
   (c) Start the simulation in `simFlexShaft.mdl`. This automatically loads the parameters of `updateVarParamFlexShaft.m` into the model and starts with measuring the desired response.
   (d) As soon as the measurements are completed, the simulation stops automatically and the recorded data is displayed. View the plot and save it for your records.
6. After having completed all the tasks: Store the file `updateVarParamFlexShaft.m` safely; you will need it in the lab session to test your controller on the real hardware.

2.2 Analysis of the Flexible Shaft Model

Task 1: Transfer Function Models

Convert the continuous-time state space model given in the MATLAB structure `syscr` to the input-output transfer function model $G(s)$ and to the disturbance transfer function model $G_z(s)$ using the MATLAB functions `ss` and `tf`. Write your code into `updateVarParamFlexShaft.m` at marking `Task A.1` and run the file.

Task 2: Bode Diagram

Plot the Bode diagram of the input-output transfer function $G(s)$ and the disturbance transfer function $G_z(s)$ using the MATLAB function `bode`. Write your code into `updateVarParamFlexShaft.m` at marking `Task A.2` and run the file.

Task 3: Analysis based on Bode Diagrams

In this task you will do a *qualitative* analysis of the Bode diagrams of the input-output transfer function $G(s)$ and the disturbance transfer function $G_z(s)$.

1. In the magnitude plot of the input-output transfer function you see a downward and an upward peak in the frequency range from 1 to 10rad/s. What are the names of these peaks?
2. What poles and zeros (real and conjugate complex) does the input-output transfer function $G(s)$ have according to the Bode diagram derived in task 2? What is the relative degree of $G(s)$?
3. What poles and zeros (real and conjugate complex) does the disturbance transfer function $G_z(s)$ have according to the Bode diagram derived in task 2? What is the relative degree of $G_z(s)$?

Task 4: Poles and Zeros of the Transfer Function Models

For each subtask, write your code into `updateVarParamFlexShaft.m` at marking `Task A.4` and run the file.

1. Use the MATLAB function `eig` to calculate the eigenvalues of the system matrix A_c respectively the poles of the input-output transfer function $G(s)$.
2. Calculate the zeros of the transfer function $G(s)$ using the MATLAB function `tzero`.
3. Does the disturbance transfer function $G_z(s)$ also have zeros?
4. Do the poles of the two transfer function models $G(s)$ and $G_z(s)$ coincide with each other? If yes, why is this so and is this generally the case? Hint: Pole-zero cancellations.
5. Where does the difference between those two transfer function models come from? Hint: Reflect on the input-output relations.

Task 5: Step Responses

Use the simulator (as described in step 5 of Section 2.1) to plot the input-output step response and the disturbance step response. Do not forget to set the function selector in the MATLAB script `updateVarParamFlexShaft.m` to the corresponding value (`2` or `3`). Note that the simulator will apply to $G(s)$ or $G_z(s)$ a profile consisting of several positive and negative steps.

Task 6: Increased Moment of Inertia

Set the moment of inertia J in the MATLAB script `updateVarParamFlexShaft.m` to the value $1002 \cdot 10^{-6}\text{kgm}^2$ (in `CONFIGURATION`).

For the following subtasks you can reuse the code of `Task A.2` and `Task A.4`.

1. Plot the Bode diagram of the input-output transfer function $G(s)$. What impact does the increased moment of inertia have?
2. Calculate the poles and zeros of the transfer function $G(s)$. Has the increased moment of inertia an influence on the poles and zeros? If yes, do you have any physical explanation for this?

2.3 PI Controller

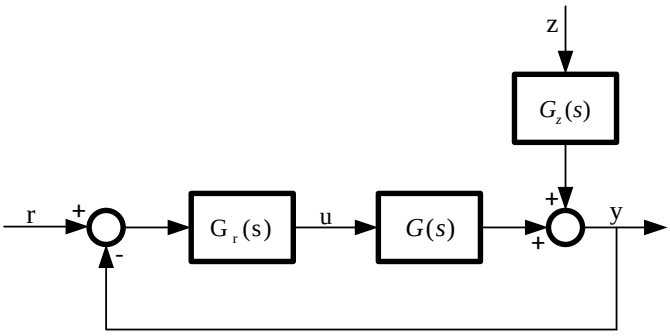

Figure 2.1: Block Diagram of Feedback Structure

After having analyzed the plant models, we are now able to design a PI controller. Figure 2.1 shows the feedback structure used in the experiment, where the transfer function $G_r(s)$ of the controller is given as

$$G_r(s) = K_p \cdot \left(1 + \frac{K_i}{s}\right), \tag{2.1}$$

and the loop transfer function is defined as $L(s) = G(s) \cdot G_r(s)$. Before we can start with the actual controller tuning, we set some **design criteria**:

- The gain crossover frequency of the loop transfer function L(s) shall be below the antiresonance frequency, i.e. the frequency where the magnitude $|L(j\omega)|$ crosses the 0dB-axis shall be smaller than the antiresonance frequency.
- The gain crossover frequency should be as large as possible.
- The phase margin shall be greater than 50°.

- The bounds $[-18\text{V} + 18\text{V}]$ of the control action u shall be respected by the controller.

Task 7: Controller Design

Set the moment of inertia J in the MATLAB script `updateVarParamFlexShaft.m` back to the value $402 \cdot 10^{-6}\text{kgm}^2$.

1. Why must the gain crossover frequency of the loop transfer function be smaller than the antiresonance frequency?
2. Design a PI controller that meets the specified requirements. Write your code into `updateVarParamFlexShaft.m` at marking `Task A.7` and run the file.
   Hint: one possible procedure is as follows:
   - Decide on desired crossover frequency.
   - Tune K_i such that the phase margin at desired crossover frequency is enough.
   - Tune K_p to obtain the desired crossover frequency.

   For this procedure you need a Bode plot of $L(s)$.
3. Use the simulator (as described in step 5 of Section 2.1) to plot the reference step response and the closed-loop disturbance step response. Do not forget to set the corresponding function selector (`4` or `5`) and make sure that the controller selector is set to `pid`.

   Note: For the reference step response, a reference profile consisting of a sequence of several positive and negative steps is applied. For the the closed-loop disturbance step response, the shaft is successively operated at three reference velocities (positive, zero, negative) and in each of them, a sequence of positive and negative disturbance steps are applied.

Remark: We do the design in continuous time, despite the fact that we are applying digital control. The provided SIMULINK model implements the discrete-time equivalent of (2.1) that results by trapezoidal integration (Tustin's method, [1, §8.3]), namely $G_r(z) = K_p \cdot \left(1 + K_i \cdot \frac{T(z+1)}{2(z-1)}\right)$.

Chapter 3

Lab Session A

In this lab session, we verify the transfer function models of the flexible shaft by applying some automatically generated test signals, and we test the PI controller design of Chapter 2 by applying it to the real hardware.

3.1 Instructions

1. Double-click the icon `ifa_3_7.ps1` on the desktop of the lab computer. This will download all necessary files into `C:\Scratch\Flex_Shaft` and start MATLAB automatically.

2. Copy the MATLAB script `updateVarParamFlexShaft.m` generated during the home exercises to the current MATLAB folder and open it. Set the mode selector to `modeSel=1` and make sure that the inertia is set to `J=402e-6`.

3. Open the SIMULINK model `targetSysFlexShaft.mdl`, which serves as graphical user interface (GUI) to interact with the control unit. Figure 3.1 shows the GUI with some highlighted elements.

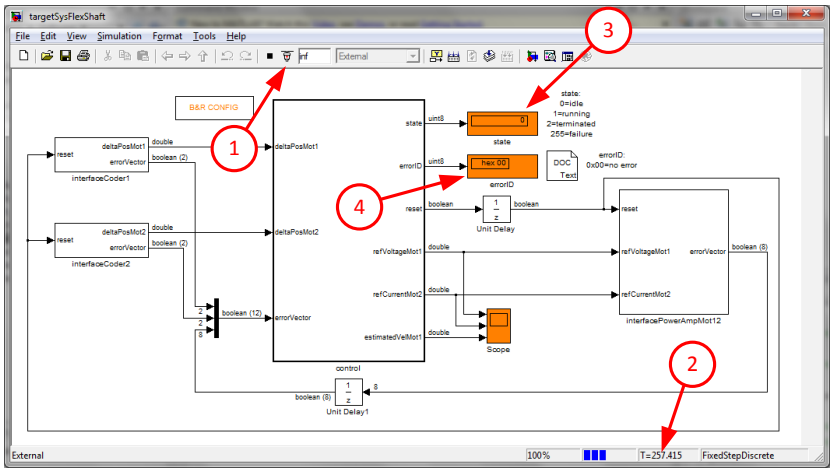

Figure 3.1: GUI Flexible Shaft

4. Power on the control unit of the flexible shaft with the switch on the back side, and wait until booting is completed and the application is running, which is indicated by a green shining `R/E` light (this is the top most) on the front side of the CPU module.

5. Now you are ready to complete the tasks according to the description in the next section of this chapter. For each task proceed as follows:

   (a) Connect (Fig. 3.1(1)) the GUI to the control unit, and wait until the absolute timer (Fig. 3.1(2)) is running.

   (b) Configure `updateVarParamFlexShaft.m` according to the given task, i.e. set the function selector `fcnSel` and the controller selector `controllerSel` to the appropriate value. Set `cmd=1` and run the file. The control unit should go to state *running* (Fig. 3.1(3)). The flexible shaft starts rotating, and the measurements are in progress.

   (c) Wait until the control unit is in state *terminated*, meaning that the measurements are completed. In the meantime you can observe the real system and look at the scope of the GUI. Note:

   - You can stop the flexible shaft at any time by proceeding with step 5d!
   - If the state goes to *failure*, then a hardware error occurred. For more information, check the `errorID` (Fig. 3.1(4)). To reset the error, proceed with step 5d.
   - If SIMULINK shows an error message box, then restart from step 5a.

   (d) Set `cmd=0` in `updateVarParamFlexShaft.m` and run the file. The control unit goes back to state *idle*, and the flexible shaft stops rotating.

   (e) Disconnect (Fig. 3.1(1)) the GUI from the control unit. The recorded data is automatically loaded and displayed. View the plot and save it for your records.

   (f) If the obtained results are not satisfactory enough, adjust the parameters of your controller in `updateVarParamFlexShaft.m` and repeat from step 5a.

6. After having completed all the tasks:

   (a) Store the file `updateVarParamFlexShaft.m` safely; you will need it again for Part B.

   (b) Power off the control unit of the flexible shaft.

3.2 Model Verification

Task 8: Frequency Responses

1. First, measure the input-output frequency response, $G(j\omega)$. Do not forget to set the corresponding function selector. The response will be automatically plotted to the Bode diagram together with the response of the transfer function model. Finally view the diagram and summarize the observations you make.
2. Optional: Do the same for the disturbance frequency response, $G_z(j\omega)$.

Remark: In this experiment, a number of sine-waves of distinct frequencies are sequentially applied to the system, and for each frequency, after the system is stationary again, the DFT of the input and output is measured synchronously using the Goertzel algorithm [2, §5.5]. Note that to prevent the dynamics from getting influenced by stiction, the experiment is not done around the 0-speed point; instead a constant offset is superimposed to the sine-waves. Further note that measuring the whole frequency response takes about a quarter of an hour.

Task 9: Step Responses

1. Measure the input-output step response by setting the corresponding function selector. The response as well as the input will be automatically plotted to a diagram together with the responses of the model. Finally, view the diagram and summarize your observations.
2. Do the same for the disturbance step response.

3.3 PI Controller

Task 10: Verification of PI Controller

1. Measure the reference step response. Do not forget to set the corresponding function selector, and make sure that the controller selector is set to `pid`. As before, the recorded data is plotted into a diagram. View the diagram and comment on the observations you make. You might want to compare the step response with the simulated response.
2. Do the same for the closed-loop disturbance step response.

3.4 Conclusion

Task 11: Conclusion

1. Fill in the online feedback form found in `MyExperiments` on the laboratory courses registration page. Each participant has to give his/her own feedback. Thank you for your inputs! This helps us to improve our experiments.
2. Discuss the experiment with your supervisor to get the confirmation (Testat).

Bibliography

[1] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Pearson Prentice Hall, 5th edition, 2006.

[2] Daniel Ch. von Grünigen. *Digitale Signalverarbietung*. Fachbuchverlag Leibzig im Carl Hanser Verlag, January 2008.