CACSD Practical Session
Rotary Flexible Joint

Oscar Mauricio Agudelo, Bart De Moor
ESAT-STADIUS
Kasteelpark Arenberg 10, 3001 Heverlee
e-mail: `mauricio.agudelo@esat.kuleuven.be`
tel:(016)32 86 51

The original manual was written by Peter De Gersem and Jeroen Buijs.
This manual has been updated by Oscar Mauricio Agudelo.

2016

1 Aim of this Session

The subject of this session is the *Rotary Flexible Joint.* This setup consists of an arm, connected with springs and a joint to a hub, which can rotate in a horizontal plane, driven by a motor (fig. 1). You'll have to design a controller for this system, that lets the arm *track* a particular input-function (e.g. a step, or a sine-wave) as good as possible. This is a common problem in robotics, where one wants to position the tip of a flexible robot arm as quick as possible on the right place. The measured signals are the angle of the hub and the angle of the arm (both obtained with potentiometers), the output is the voltage to the motor. The controller is implemented in software on a PC, equipped with a data-acquisition card.

It's important that you read this manual completely before starting. And read section 3 again before doing experiments on the real setup.

2 Control Strategy

First of all, the CACSD course is concerned with *model-based* control system design. So the first thing to do is to find the *state-space* model of the system. Next, you can apply the model-based methods described in the course notes to design the controller. You will need a computer program to do so (*Matlab*), and in the meantime you can do some simulations and calculations to predict the response of the *closed-loop* system[1] in *Simulink.* When you're convinced that your controller has the right properties, it's time to try it out on the real system.

[1]The closed-loop system is the system consisting of the plant and the controller.

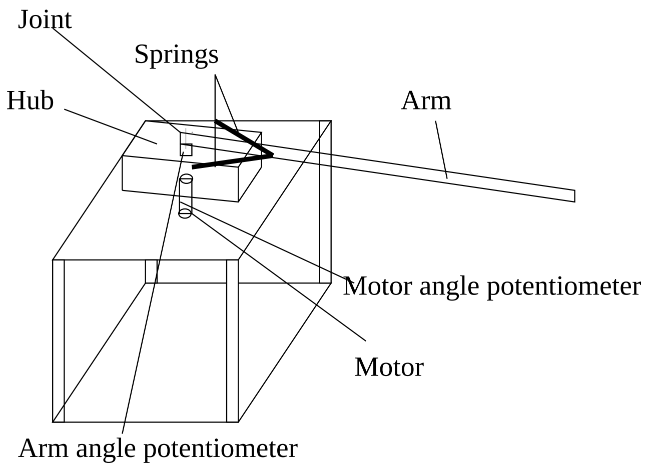

Figure 1: The rotary flexible joint setup.

2.1 The State-space Model

There are different possibilities to obtain the state-space model of a physical system. One of them is identification. In this case however, physical modeling, i.e. using some mechanical and electrical laws you've probably already forgotten, is simple enough, so that's the way we will do it. The derivation of the model is given, but make sure that you understand what is happening, not only because we might ask you some questions about it, but especially since some insight in how the model is derived may lead to a better understanding of the results you obtain later on. It's logical that good knowledge of the system you're going to control and of the model you use to represent this system, is the first step towards a well functioning controller.

First, we will model the mechanical part of the system. In a second step the electrical part will be modeled and the two models will be concatenated.

2.1.1 The Mechanical Part

In the modeling, you can ignore the friction. Fig. 2 shows a drawing of this simplified model. Because the equivalent stiffness of the two springs is quite difficult to calculate (fig. 3), we give the (linearized) formula right away:

$$K_{\text{stiff}} = [2\frac{R}{D^{3/2}}][(Dd - Rr^2)F_r + (D^{3/2}d - DLd + Rr^2L)K]$$

with $D = r^2 + (R-d)^2$, F_r the *spring restoring force*, and L the *spring rest length.* The meaning of this equivalent stiffness K_{stiff} is $T_s \simeq K_{\text{stiff}}\alpha$, where T_s is the torque executed by the springs, and α is the angle of the arm relative to the hub. This relation is linearized around $\alpha = 0$. Make sure you understand how this K_{stiff} could have been calculated!

The input of the mechanical system is the torque T (coming from the motor), the outputs are the angle of the hub θ and the angle of the arm (relative to the

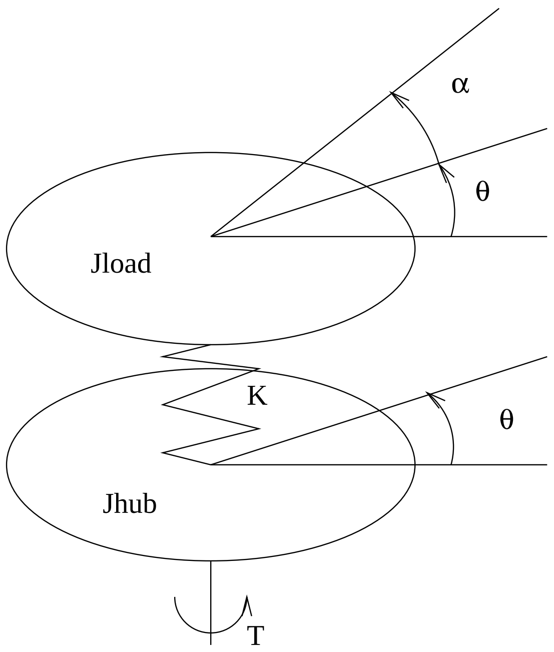

Figure 2: The simplified model of the rotary flexible joint setup.

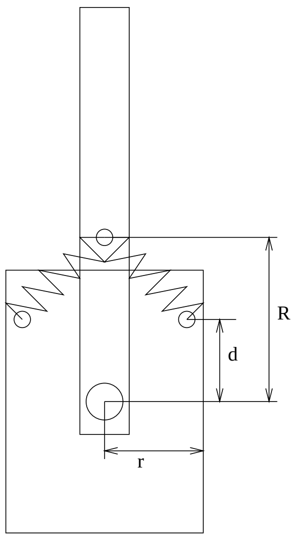

Figure 3: A detailed view on the spring fixation.

hub) α. As you might discover, the state-space model of this system contains four states. The most logical choice seems $\{\theta, \alpha, \dot{\theta}, \dot{\alpha}\}$, so we will stick to these.

We will use the Lagrange equations here to derive the model equations. They are given by

$$\begin{aligned} \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\alpha}}\right) - \frac{\partial L}{\partial \alpha} &= 0 \\ \frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} &= T \end{aligned}$$

where the Lagrangian $L = E_{kin,tot} - E_{pot,tot}$ and T is the torque coming from the motor. The total kinetic energy for the rotary flexible joint system can be written as

$$E_{kin,tot} = E_{kin,hub} + E_{kin,arm} = \frac{J_h \dot{\theta}^2}{2} + \frac{J_l(\dot{\alpha} + \dot{\theta})^2}{2}$$

where J_h and J_l are the inertias of the hub and the arm (or load) respectively. The total potential energy (from the springs) is given by $E_{pot,tot} = \frac{K_{\text{stiff}}\alpha^2}{2}$. Therefore the Lagrange equations lead to the following equations,

$$\begin{aligned} J_l\ddot{\alpha} + J_l\ddot{\theta} + K_{\text{stiff}}\alpha &= 0 \quad (1) \\ (J_l + J_h)\ddot{\theta} + J_l\ddot{\alpha} &= T \quad (2) \end{aligned}$$

2.1.2 The Electrical Part

Since our control input is not the torque T, but the electrical voltage V applied to the motor that generates this torque, we have to eliminate T by introducing the electrical equations of the motor. The equations governing a DC-motor are

$$V = IR_m + K_b\omega_m$$

$$T_m = K_m I\,,$$

in which R_m is the *Armature resistance*, K_b the *Back EMF constant*, K_m is the *Motor torque constant*, T_m is the torque produced by the motor at its shaft. The meaning of V, I and ω_m is clear, I hope. The motor drives a gearbox of ratio K_g (i.e. $\omega_m = K_g\omega_g = K_g\dot{\theta}$ and $T_g = K_g T_m$, in which ω_g and T_g are the rotation speed and the torque at the output of the gear). The torque T_g is of course the torque T in equation (2). Concluding, the relation between T and the applied voltage V is given by

$$T = \frac{K_m K_g}{R_m}V - \frac{K_m K_b K_g^2}{R_m}\dot{\theta} \quad (3)$$

Eq. (3) can be used to eliminate T from (2).

2.1.3 The Global Model

Solving equations (1) and (2) for $\ddot{\alpha}$ and $\ddot{\theta}$, a linear state space model is obtained. Since the methods we will use to control this system are based on linear models, this model will do fine. Nevertheless, think a second about what you should have done in case the obtained state equations were non-linear. The linear state equation can be written in the standard matrix form $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$,

$$\begin{bmatrix} \dot\theta \\ \dot\alpha \\ \ddot\theta \\ \ddot\alpha \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_{\text{stiff}}}{J_h} & \frac{-K_g^2 K_m K_b}{J_h R_m} & 0 \\ 0 & \frac{-(J_l+J_h)K_{\text{stiff}}}{J_l J_h} & \frac{K_g^2 K_m K_b}{J_h R_m} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \dot\theta \\ \dot\alpha \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_m K_g}{R_m J_h} \\ \frac{-K_m K_g}{R_m J_h} \end{bmatrix} V$$

Adding the output equation, you obtain a global state space model. The values of the different constants for the rotary flexible joint are given at the end of this manual. Make sure to convert them to the right units. Now the model is known, make a short open loop analysis of the system and the model, and try to formulate the control problem.

2.2 The Controller

The controller you're going to design is of LQR-type. The controller will be **designed in continuous time** and and it will be implemented in a digital computer. If the sampling time in the computer is small enough, we can consider that the controller would be operating in continuous time (of course, this is an approximation). This is the approach that would be followed in this practical session. Because the states θ and α are measured, and $\dot\theta$ and $\dot\alpha$ can be easily calculated from these measurements, no state-observer is needed.

The LQR-controller determines an optimal state-feedback gain K, such that the closed-loop system $A - BK$ (i.e. $\mathbf{u} = -K\mathbf{x}$) minimizes the quadratic performance index

$$J = \int (\mathbf{x}^t Q \mathbf{x} + \mathbf{u}^t R \mathbf{u}) dt\,.$$

The weight-matrices Q and R determine the relative importance of minimizing the states and the input-signal. You can take

$$Q = \begin{pmatrix} 350 & 0 & 0 & 0 \\ 0 & 1500 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}$$

and

$$R = 10$$

as initial values (this assumes your states are $\mathbf{x} = \begin{bmatrix} \theta & \alpha & \dot\theta & \dot\alpha \end{bmatrix}^t$, and that your state-space matrices are expressed in SI units). Use the Matlab-command[2] `lqr` to calculate K.

It might not be clear to you how to apply an input to the closed loop system, and how to simulate that in *Simulink* (e.g. for the step- or impulse-response): to position the arm at a certain angle, we have to tell the LQR-controller that

[2]To figure out how a Matlab-command works, type `help` *command* at the Matlab-prompt. Also if you don't know how to do something in *Matlab*, you should try `help`. If you're really stuck, you can always mail us for suggestions.

the θ-state has to go to θ_{desired} instead of zero. To obtain that behavior, we calculate the state-feedback not with $\mathbf{u} = -K\mathbf{x}$, but with $\mathbf{u} = -K(\mathbf{x} - \mathbf{x_d})$, where $\mathbf{x_d}$ equals $\begin{bmatrix} \theta_{\text{desired}} & 0 & 0 & 0 \end{bmatrix}$. So the input to your controller block K in *Simulink* has to be the difference between x and x_d[3].

While designing a controller, you need to evaluate the performance of your design, to be able to compare it with other designs. Some *evaluation criteria* are

- the **closed-loop eigenvalues (or pole-zero plot)**. They give information about the stability-margin, the damping and the eigenfrequency of the closed-loop system.
- the **step-response**. This gives information about settling-time, rise-time, overshoot, amount of input-signal…
- the **frequency-response**. This gives information about the closed-loop eigenfrequencies, the damping,…

With LQR, the choice of the weight-matrices is the most difficult part of the design. The strategy proposed here is simply *trial-and-error*: start with the initial values, evaluate the prestations, change one parameter and evaluate, change another parameter,… The ideal controller is stable, tracks the input as close as possible (small rise-time, low overshoot, small settling-time), and is not (too) nervous or noise-sensitive.

It's up to you to find good weight-matrices with trial-and-error. Keep a record of your different trials (make plots of the evaluation criteria), and argument why you are satisfied with your final design. Put the representative information in your report (see section 4). Don't spend *too* much time in this: **it's important that you have a satisfying controller, and that you understand what the effect is of changing parameters in Q and R.** It's not the aim of this practical session to try as much different values for Q and R as possible.

The result of your iterative design is a state-feedback gain K.

2.3 Implementation Details

The controller is implemented in *Simulink* on a digital computer with a data acquisition board. Normally *Simulink* is used to carry out simulations, but using the *Real-Time Windows Target* and the *Simulink Coder* of *Matlab*, *Simulink* can be used to control processes in real-time through a data acquisition system. This approach is used in the practical sessions.

The inputs of the system (angle of the hub and angle of the arm) are measured by means of a biased potentiometer[4] and a data-acquisition card in the computer: the voltage at the wiper-pin of the potentiometer is a measure for

[3]Another approach would be to calculate one closed-loop state space system from (A, B, C, D) and K with input x_d. But never mind.

[4]The two ends of the potentiometer are connected to the +12V and −12V terminals of the Power Supply, and the wiper-pin of the potentiometer is connected to the arm resp. the hub.

the angle of the arm and the hub. These voltages are converted to a digital values by an A/D-converter (voltage range: -10V to 10V, resolution: 16 bits). These digital values are read by the controller (real-time Simulink diagram containing the control algorithm), and are converted from voltage to engineering units (radians). We have a linear relationship. For the angle of the hub -2.63V to +2.63V corresponds with -90 to 90 degrees and for the angle of the arm -6.329V to 6.329V corresponds with -90 to 90 degrees.

The controller samples these voltages at a frequency of 200 Hz (that's an arbitrary choice: fast enough compared to the dynamics of the system, not too fast for the computer to handle). The states $\dot{\theta}$ and $\dot{\alpha}$ are calculated using the backward difference equation:

$$\dot{y} = \frac{y^f(t) - y^f(t - T_s)}{T_s} ,$$

where T_s is 1/200 s (5 ms). To use this technique, it's necessary to low-pass filter the raw measurements before calculating the difference, because these are very fluctuation- and noise-sensitive (that's why there's y^f instead of y). In the controller-program, you can try different values for the cut-off frequency ω_c of this low-pass filter to study the effects. A typical value is 2 Hz ($\omega_c = 2 \cdot 2\pi$ rad/s). The low-pass filter obeys the following law:

$$y_n^f = \frac{\omega_c T_s}{1 + \omega_c T_s} y_n + \frac{1}{1 + \omega_c T_s} y_{n-1}^f .$$

Make sure you understand what this filter is doing so that you understand where exactly the outputs should be filtered, since you will have to include this filter in your simulation schemes. Hint: Typically 50% of the students is doing this wrong.

Then the output voltage is calculated as $V = -K\mathbf{x}$ or $V = -K_1(\theta - \theta_{\text{desired}}) - K_2\alpha - K_3\dot{x} - K_4\dot{\alpha}$. Of course, the output voltage to the motor is limited: the controller limits it to ± 5V. This voltage is then converted to an analog value with the D/A-converter, and a power-amplifier applies this voltage to the motor of the hub.

The *Real-Time Windows Target* includes a real-time engine that runs in Windows kernel mode. This real-time engine loads I/O device drivers and establishes a connection with *Simulink*. For more details you can consult the documentation of this toolbox.

3 The Proof of the Pudding

Now it's time to implement your controller. A Simulink template has been created for you (see figure 4). This template has been configured appropriately for running in real-time and it contains two blocks: the "Analog inputs" block and the "Analog Output" block. The "Analog Inputs" block gives you the measurements of the sensors in engineering units each sampling time (5 ms). The "Analog Output" Block allows you to send the control action (volts) to the

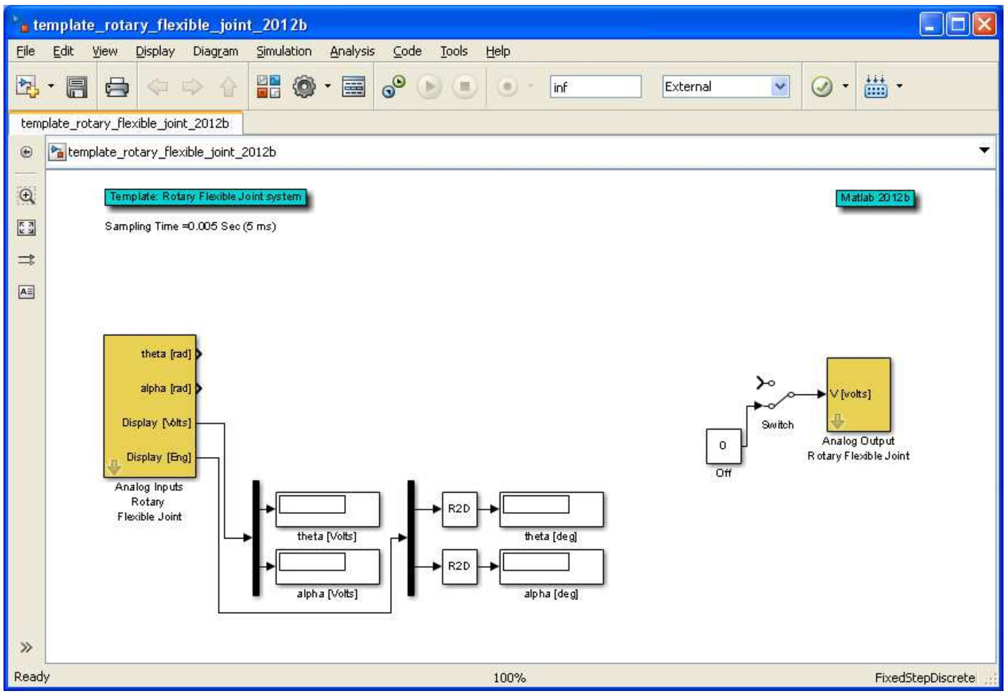
template_rotary_flexible_joint_2012b
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
External
Template: Rotary Flexible Joint system
Sampling Time =0.005 Sec (5 ms)
Matlab 2012b
theta [rad]
alpha [rad]
Display [Volts]
Display [Eng]
Analog Inputs Rotary Flexible Joint
theta [Volts]
alpha [Volts]
R2D
R2D
theta [deg]
alpha [deg]
Switch
0
Off
V [volts]
Analog Output Rotary Flexible Joint
Ready
100%
FixedStepDiscrete

Figure 4: Template of the real-time Simulink diagram.

plant. So, what you have to do is to complete the Simulink diagram by means of the necessary blocks in order to implement your control strategy.

You should plan your experiments before you come to the setup, because the time is limited. Try to spend your time at the setup as efficiently as possible.

The actual setup resides in room 02.58. On a table you find the setup, a power module, and a PC. Before you do anything, read these instructions thoroughly, because the setup is not that robust.

To use the set-up, you'll have to make a reservation first. To this end, use the online CACSD practical setup reservation system[5]. Normally the best thing to do is to make a reservation for two up to four hours, just in case something goes wrong. More practical information about the actual testing can be found in the webpage of the CACSD course in Toledo. Remember to regularly check this webpage for frequently asked questions and other advice. Initially, don't hesitate to exchange information between groups and help each other out. Eventually, don't hesitate to contact the CACSD assistants if unlisted problems arise.

3.1 Description of the software

The Simulink template is shown in Figure 4. The "Analog Inputs" block gives the measurements of the sensors (they are updated each 5 ms) in radians for the angles of the hub and the arm. Additionally this block has two special

[5]Check the webpage of the CACSD course in Toledo. Section: Practical sessions.

outputs, “Display [volts]” and “Display [Eng]”, which are used for visualization purposes. These two outputs are connected to a set of displays in order to provide a local visual measure of the process variables. Each display is updated each 0.25 seconds. **Don’t use “Display [volts]” or “Display [Eng]” in your control strategy because their update time is larger than 5 ms**. The “Analog Inputs” block has the following configuration parameters:

- **F_theta:** This is the Conversion factor from volts to radians for the angle of the hub. It has been set for you.
- **Off_theta:** This is a voltage value that is added to or subtracted from the voltage read by the Data acquisition system. It is used to set the zero position of the angle of the hub.
- **F_alpha:** This is the Conversion factor from volts to radians for the angle of the arm. It has been set for you.
- **Off_alpha:** This is a voltage value that is added to or subtracted from the voltage read by the Data acquisition system. It is used to set the zero position of the angle of the arm.

The “Analog Output” block allows the controller to send a voltage signal to the motor of the hub. At the input of this block there is an on-off switch. This switch allows you to switch on or switch off the motor. If you want to change its state, you only have to click it. Internally the “Analog Output” block includes a “Saturator” block which limits the output voltage from -5V to 5V.

In order to run the real-time Simulink diagram you have to follow these steps:

- In the Simulink window, and from the “Code” menu, point to “C/C++ Code”, and then click “Build Model”. During the Build Process the Simulink diagram is converted into C code and afterwards it is compiled in order to generate a special binary file, the “real-time application”.
- From the “Simulation” menu click “Connect To Target”. This action loads the “real-time application” into memory. Also, you can connect to the target from the toolbar by clicking .
- Finally, from the “Simulation” menu click “Run” for starting the execution of the real-time application. You can also start the execution from the toolbar by clicking .

For stopping the real-time application, you have to click “Stop” in the “Simulation” menu (or by clicking in the toolbar). This action automatically unloads the “real-time application” from memory.

The template was created using *Matlab* R2012b and *Simulink* 8.0. It is important to open such template using the mentioned *Matlab* version in order to avoid compatibility problems. The Simulink template is available in the webpage of the CACSD course in Toledo.

3.2 Try it out

If the setup is not ready, or if you think there's something wrong, contact the CACSD assistants. It's a good idea to check for loose cables.

Before testing your controller it is really important to calibrate the zero position for the angles of the hub and the arm. Once you have loaded the Simulink template, ensure that the position of the switch (it is connected to the "Analog Output" block) in the Simulink template is pointing to "Off". Then start the execution of the real-time application (the template) as it was explained in the previous section. Now, turn on the Power Module PA0103 by pressing the Red On/Off button. Once the real-time application is running you will observe the measurements of the sensors in the displays. Now, Turn the arm and the hub to zero degrees and check the "theta[volts]" display and the "alpha[volts]" display. If one of the values shown (or both) are not close to zero, you have to follow these steps:

- Double click the "Analog Inputs" block in order to see its configuration parameters.
- If there is an offset problem with the angle of the hub, then write the voltage read in the "theta[volts]" display with opposite sign in the "Off_theta" text box. If there is an offset problem with the angle of the arm, then write the voltage read in the "alpha[volts]" display with opposite sign in the "Off_alpha" text box.
- Click Ok.

After the previous actions, you will observe how the reading in the "theta[volts]" display and/or "alpha[volts]" display will be close to zero.
When you are ready for testing the controller, follow this guideline:

- Start your real-time application, keeping the position of the switch (it is connected to the "Analog Output" block) at "Off".
- Turn the arm and the hub to zero degrees and turn the motor on by clicking the switch in the Simulink diagram. Make a step change in the setpoint of angle of the hub. If the controller doesn't do what you expected, turn the motor off and reconsider your design. Maybe you used the wrong signs for the gains.

  If everything goes well, collect data from the system (angles, angular velocities, the voltage sent to the Motor, etc.) in order to generate some representative plots (see section "Report") in *Matlab*.
- Now, you have to work with a state-feedback controller (second controller) which uses only the states θ and $\dot{\theta}$ (i.e. the state feedback gains of α and $\dot{\alpha}$ are set to zero); try it out and you'll see the difference between the uncontrolled and controlled sweep of a robot-arm. Compare the two controllers (with and without feedback from the arm).

- Working again with the first controller (with feedback from the arm), generate the setpoint of the angle of the hub by means of a sine wave generator (an amplitude of 25 degrees is ok). Hold the frame while performing the sine response, because it can start slipping. You can vary the frequency of the sine wave from 0.5 Hz to 3 Hz in steps of 0.1 Hz in order to study the frequency response of the closed-loop system. Compare the frequency-response with the simulated response.
- When you have finished, stop the real-time application (your controller in *Simulink*), turn the power module off, and turn the computer off.

4 Report

It might be a good idea to read this section before you start your calculations, simulations and tests. It might give you an idea of the things you should do and the things you should't waste time on.

The report should be written preferably **in English**. One report per group suffices. This report will be used at the oral "examination" (see further).

Contents of the report:

1. Model and open loop analysis
   (a) Provide the linear state-space model of the system. Define clearly which are the state variables of the model and present the numerical values of the matrices A, B, C and D. The matrices C and D must be determined based on the sensors present in the setup.
   (b) Perform a short open loop analysis. Provide the poles and transmission zeros of the system. Is the system stable? controllable? observeable? stabilizable? detectable? minimal? Justify your answer.
   (c) State very clearly what are the control goals (e.g., stabilization, disturbance rejection, tracking of setpoints, fast response, etc.).
2. Design of the LQR controller and creation of the first closed-loop simulation diagram

   Here you have to create a Simulink diagram mainly containing the linear model (continuous time) of the plant and the LQR controller. The assumption here is that the entire state vector is available and therefore for this particular case the matrix C must be set equal to the identity matrix.

   (a) Provide the Simulink diagram and a very brief explanation of the blocks it contains.
   (b) Starting with the matrices Q and R given before, you should follow a systematic trial and error procedure to find a good pair of matrices Q and R for your controller that satisfy the control goals (to this end you

can use some of the evaluation criteria presented before, like for example the step response). Discuss the effect of changing the different diagonal entries of the matrices Q and R on the dynamics of the closed-loop system.

(c) With the chosen Q and R simulate the closed-loop system after a change in the setpoint of θ (this setpoint change must be set according to the physical limits of the setup. So, you must perform a realistic setpoint change!!!). Provide the simulation results and discuss them very briefly. Do not forget to check the magnitude of the control actions.

3. <u>Creation of the second closed-loop simulation diagram</u>

   In order to have a more realistic simulation, you have to create a Simulink diagram where the outputs of the linear model are the variables that are measured in the real setup (θ and α). Therefore in order to derive the entire state vector, you have to compute the derivatives of the measured outputs by using the backward difference equation and the low pass filter described in Section 2.3. Additionally in this diagram, you have to include the saturation of the actuator, the quantization effects of the A/D converters and the measurement noise.

   (a) Provide the Simulink diagram and explain the blocks it contains. Do not forget to mention how you set the saturation limits, the quantization interval and the magnitude of the measurement noise (can you estimate it from real measurements?).

   (b) By using the LQR found previously and a cut-off frequency of 2 Hz for the low-pass filters, apply the same step change as in 2(c) and simulate the closed-loop system. Compare the simulation results with those obtained with the first simulation diagram and discuss. If it is necessary you can adjust the values of Q and R to have a better performance. If you have to do this then add the new simulation results.

   (c) Investigate the role of cut-off frequency of the low pass filters. What is impact of the cut-off frequency on the dynamics of the closed-loop system? What happen when this parameter is set too small? or too large?

4. <u>Experimental results (“The proof of the pudding”)</u>

   By using the provided Simulink template, you have to implement and test the control system you designed before with the real setup.

   (a) Provide the real-time Simulink diagram and explain it very briefly.

   (b) Perform some setpoint tracking tests and make some representative plots (states, control actions). Try different setpoints. Compare the experimental results with those found with the second closed-loop simulation diagram (in this case you have to apply the same setpoint change and try to start the real system around the same initial conditions as in the simulation) and discuss. It is very likely that you have to retune your controller a little bit in order to have better results. So, do it if necessary and comment. Is there steady-state error? if so, what could be the reasons?

(c) Carry out some disturbance rejection tests and make some representative plots (states, control actions). To this end you can apply a small kick against the arm. Discuss the results.

(d) As indicated in Section 3.2, remove the feedback of the states α and $\dot{\alpha}$ and compare the response of closed-loop system with the one where the full state is fed back. What can you conclude? Discuss.

(e) Provide the experimental frequency response (input: θ_{desired}, outputs: θ and α) of the closed-loop system. To this end follow the instructions given in Section 3.2. Compare it with the simulated response. Comment.

(f) Corroborate your findings regarding the influence of the cut-off frequency of the low pass filters on the controlled system. Present some representative plots and discuss very briefly.

(g) Complementary material - "OPTIONAL". Although it is not mandatory, you are encouraged to make some short videos of the different experiments you have carried out.

5. <u>Conclusions</u>

   Every well written report ends with this section, so we will not make an exception here.

Apart from the technical correctness of your practicum, the quality and presentation of your report is also evaluated. So, make sure that everything is clear and well explained. For example, make sure that you label every axis of every plot, that your figures include legends if necessary, that in the text you discuss or address what is presented in every figure of the manuscript, that the text is free of typos and well redacted, etc. If you include a Simulink diagram that contains subsystems, you should also present the contents of these subsystems.

5 Examination

Once the report is finished, you can give us a copy. **Do not forget to send us the Simulink schemes together with the necessary m-files as well.** The oral exam will include some questions about the report and some general questions. Further information (dates of examination, how the points are distributed between exercises and practicum, etc.) can be found in the webpage of the CACSD course in Toledo.

6 Specifications

This section contains some data on the rotary flexible joint setup.

Motor armature resistance : $R_m = 2.6\,\Omega$

Motor torque constant : $K_m = 0.00767\ \frac{\mathrm{N\,m}}{\mathrm{A}}$

Motor back EMF constant : $K_b = 0.00767\ \frac{\mathrm{V}}{\mathrm{rad/s}}$

Motor gear ratio : $K_g = 70 : 1$ (The output is slower)

Arm inertia : $J_l = 0.0059$ kg m^2 (at the rotating axis!)

Hub inertia (includes motor and gears) : $J_h = 0.0021$ kg m^2 (at the rotating axis!)

Body anchor point (y-coordinate) : $d = 0.0318$ m

Arm anchor point : $R = 0.076$ m

Body anchor point (x-coordinate) : $r = 0.0318$ m

Spring stiffness : $K_{stiff} = 1.60856\ \frac{\mathrm{N}}{\mathrm{m}}$

Spring restoring force : $F_r = 1.33$ N (spring will not stretch before F_r is applied

Spring length at rest : $L = 0.0318$ m