ELEC4631 Lab 3

Linear Quadratic Regulator (LQR) Controller Design for a Flexible Joint Robot

Term 2 2022

1 General information

1. This lab is conducted as both a face-to-face lab (unless there is a stay-at-home order in place) and as a **remote lab**. For remote lab sessions please follow the Remote Lab Instructions for ELEC4631 to attend it (can be found on Moodle).

2. The lab tasks must be carried out and completed during the lab time to be marked by demonstrators.

3. Lab tasks are divided into well defined time checkpoints: tasks that are scheduled to be completed in first **45 minutes**, in the first **1 hour 30 minutes**, and in **2 hours 0 mins.** Students must pace themselves to complete the tasks and have them assessed at the defined time checkpoints.

4. Students must not arrive late at the lab by more than 10 minutes. Demonstrators can refuse to allow late students into the lab.

Special notice: To run this remote lab smoothly please read this lab note thoroughly and ensure that you complete the pre-lab exercises before attending the lab. Operating a robot arm remotely requires real-time supervision by a demonstrator who is physically in the lab. As one demonstrator can only supervise one robot arm experiment at a time, well-prepared Matlab and Simulink exercises on your own computer is essential to make the process more efficient.

2 Preparatory tasks

To prepare for this lab, you will need to:

1. Revisit the lecture notes on state feedback control and LQR design. You will find it is very useful bringing the lecture notes with you to the lab.

2. Complete the pre-lab exercises as given in Sections 5.2 and 6 of this note.

3. Learn the Matlab commands `ss` and `lqr` for setting up state-space models and designing LQR controllers in Matlab. Type on the Matlab command line the command, e.g., `help lqr`. Enhance your command of the Matlab Simulink Desktop Real-time environment for system simulation and control that was introduced in Lab 1.

3 Objective

The objective of this lab is to design a state-feedback controller using the linear quadratic regulator (LQR) design technique to reset the orientation of one flexible joint on a robot arm.

1. Students are required to design and test their controllers on **Matlab simulation first**, and be able to show the demonstrators that they have used the method of LQR design to generate the simulation results. This expected to take about 2 hours of the lab time.

2. Once the demonstrator is satisfied with your control design and simulation results, students will be allowed to carry out **control experiments on robot arm system**.

3. Comparing with the results of simulation and experiments, one should realise and understand the issues that cause **the difference between the theoretical simulation and the real-world application**, such as mechanical frictions in the system, which are often non-linear, as well as other non-ideal electrical factors. You are encouraged to discuss your thoughts with the demonstrator during the lab.

4 Introduction to 2DFSJ system

In this lab, you are **remotely operating** a Two-Degree-of-Freedom (2DOF) Serial Flexible Joint (2DSFJ) robot arm system from QUANSER. It is a complex and sophisticated mechanical/electrical equipment with two harmonic drive gearboxes and four precision digital position sensors, suitable for studying advanced control topics.

4.1 Safe use of the 2DFSJ system

Although the safety risk of operating this robot system is low, It is important to follow the procedure below at all times:

1. Carefully **limit your control signal output to avoid the arm moving too fast or damaging the drives**. The control signal for this system is the current, and it is should be ≤ 0.94 A.

2. You **must ask your demonstrator to check** your Simulink settings and control simulation results **before** applying the control signal to the actual plant. There is a dedicated power switch for the power unit of the robot, in remote operation you can only ask the demonstrator to switch it on for you.

3. The robot arms **may act violently and damage** with improper controls. To avoid this you are required to: A. Carefully check your controller design and confirm the expected control results with simulations, and B. **Limit control output to a small range to test drive the system** when applying your controller at the first instance. A cut-off switch in the control program should be implemented in order to reset the control output to zero quickly.

4.2 Brief description of the 2DFSJ robot arm system

The 2DSFJ system consists of two DC motors with harmonic gearboxes (1 and 2), each driving a link (7 and 8) in serial linkage, as shown in Figure 1. The links are rigid, but the first link (7) is coupled to the drive M0 (1) by a flexible joint (9) by springs (15). It carries at its end the drive M1 (2), which is again coupled to the second link (8) via another flexible joint (10) by springs (16). As you can imagine, with two flexible joints precision position control of this serial linked robot arm system is not a trivial matter.

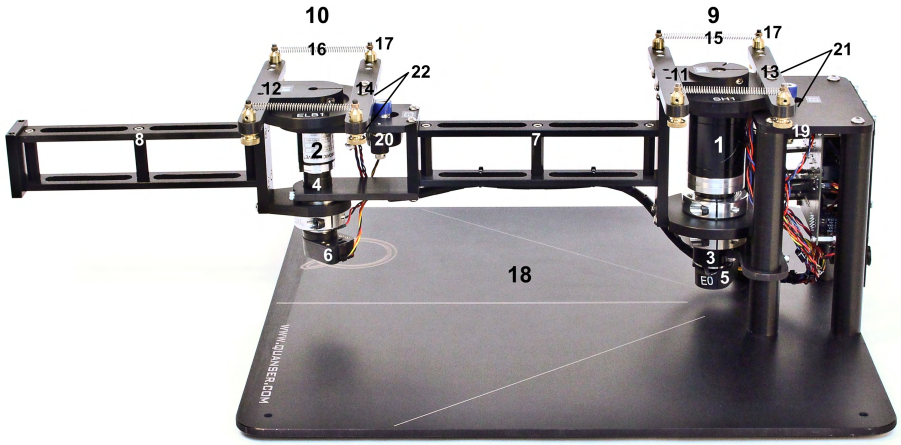

Figure 1: 2DOF Serial Flexible Joint Robot Arm System

There are also encoders and position sensors. Please make sure to familiarise yourself with the system and identify its components, as listed in the table in Fig. 2.

ID	Description	ID	Description
1	Harmonic Drive #1 (Shoulder)	**12**	Flexible Joint #1 Load Transition
2	Harmonic Drive #2 (Elbow)	**13**	Flexible Joint #2 Actuated Transition
3	DC Motor #1 (Shoulder)	**14**	Flexible Joint #2 Load Transition
4	DC Motor #2 (Elbow)	**15**	Flexible Joint #1 Spring
5	Motor #1 Encoder	**16**	Flexible Joint #2 Spring
6	Motor #2 Encoder	**17**	Spring Fixture Mechanism
7	Rigid Link #1	**18**	Base Plate
8	Rigid Link #2	**19**	Flexible Joint #1 Encoder
9	Flexible Joint Assembly #1	**20**	Flexible Joint #2 Encoder
10	Flexible Joint Assembly #2	**21**	Flexible Joint #1 Limit Switches
11	Flexible Joint #1 Actuated Transition	**22**	Flexible Joint #2 Limit Switches

Figure 2: 2DFSJ Nomenclature

4.3 A robot arm demonstration

A Simulink Desktop Real-Time program **RobotArmTest2019b.slx** has been created to test and warm up the robot arm system. It uses two closed-loop position controllers to move stage 1 (shoulder) and stage 2 (elbow) joints independently, as shown in Fig 3. Under your demonstrator's supervision you can try it out. It could also be used to set up the initial positions of the robot arm for the control experiment later in this lab.

1. Ask your demonstrator manually position both arms to be straight along the centre line.

2. Open **Matlab** and run **RobotArmTest2019b.slx** real-time control program as you have learned in Lab 1.

3. Both arms will start to repeatedly move in the opposite direction following 2 sine-wave setpoints. If you stop the program at 5, 25, 45, 65... seconds the arms' angle would be at (15^o (for SH)/-80^o (for ELB) .

5 State-space representation of 2DFSJ robot

The 2DOF flexible joint robot consists of two interconnected stages, the first stage 1 and second stage 2, the schematic of which is shown in the Fig. 4. The first stage consists of two masses, from left to right, with (rotational) inertias J_1 and J_2 (that also double as labels for these masses) that are connected by a rotational spring with spring constant K_{s1}. The mass J_1 is referred to as the *first actuated flexible joint* while the mass J_2 is referred to as the *first rigid link*. B_1 and B_2 denote the viscous damping coefficients of

the friction forces acting on J_1 and J_2, respectively. J_1 is attached to a DC motor labelled Drive 1 that generates a torque T_1 which directly rotates the mass. Similarly, the second stage consists of two masses, from left to right, with inertias J_3 and J_4 connected by a rotational spring with spring constant K_{s2}. The mass J_3 is referred to as the *second actuated flexible joint* while the mass J_4 is referred to as the *second rigid link*. B_3 and B_4 denote the viscous damping coefficients of the friction forces acting on the masses J_3 and J_4, respectively. J_3 is connected to a DC motor labeled Drive 2 that generates a torque T_2 which directly rotates the mass.

A simplifying assumption that is sometimes made is that link coupling is neglected (in reality this can only be approximately true and may not be valid in certain scenarios). That is, it is assumed that the two stages are *decoupled* in the sense that the motors attached to each stage can rotate it independently of the dynamics of the other stage. However, note from Fig. 4 that Drive 2 is attached to the first rigid link on the first stage. Also note that in Fig. 4, θ_1, θ_2, θ_3, and θ_4 denote the rotation or angular displacement of each of the masses from left to right, respectively, with respect to the rotation axis that is common to all four masses and the two motors. A convention is taken that displacement

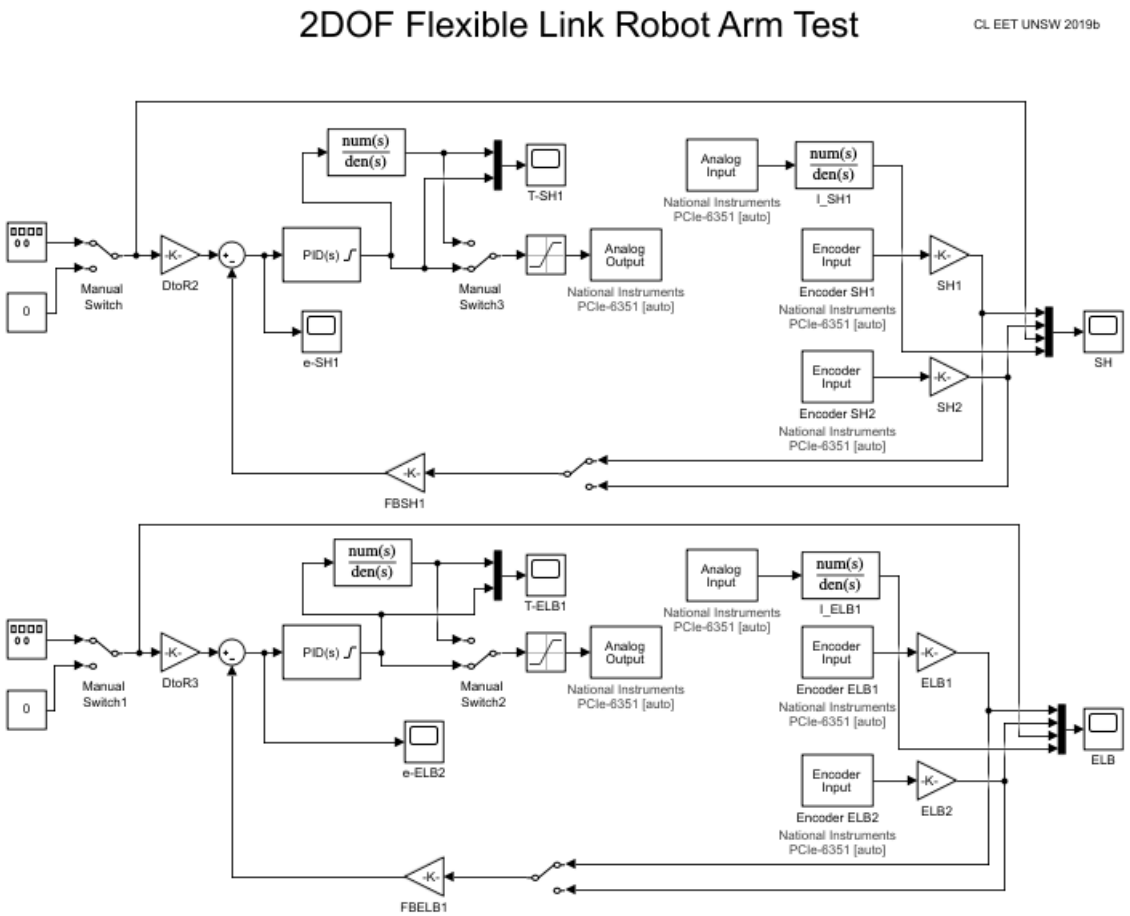

Figure 3: Flexible Link Robot Arm Test Program

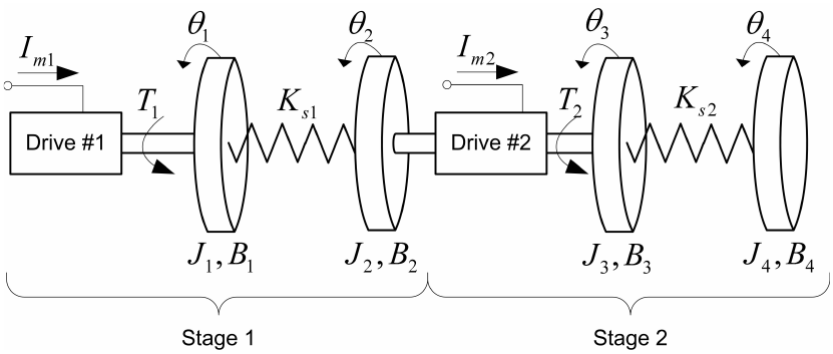

Figure 4: Schematic of two degree of freedom flexible joint

is positive in the counterclockwise direction when the robot is viewed from the top. In the following, the equations of motion for the first stage will be derived.

5.1 Dynamic equations for the first stage

For this lab **students only study the control of the first stage**. The schematic of the first stage is shown in Fig. 5.

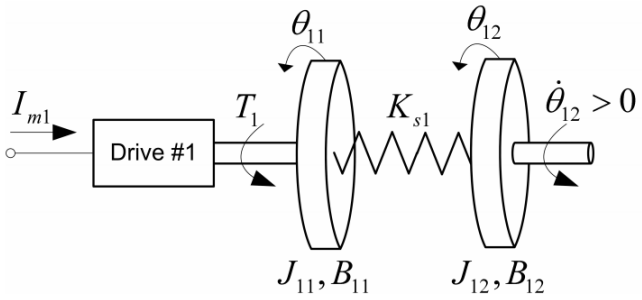

Figure 5: Schematic of the first stage

Here θ_{11} and θ_{12}, denotes the angular displacement of the first actuated flexible joint and the first rigid link, respectively, with respect to a common rotational axis that passes through the center of Drive 1. J_{11} and J_{12} denote the harmonic Drive 1 equivalent output load inertia (corresponding to the first actuated flexible joint) and the first link load inertia compounded with the second link assembly (corresponding to the first rigid link), respectively. B_{11} and B_{12} denote the first actuated flexible joint viscous damping coefficient and compounded first rigid link viscous damping coefficient, respectively. The angular displacements θ_{11} and θ_{12} are also referred to as the *generalised coordinates* for the first stage.

The dynamic equations of this system is given as the following:

$$
\begin{aligned}
J_{11}\ddot{\theta}_{11} - K_{s1}\theta_{12} + K_{s1}\theta_{11} &= K_{t1}I_{m1} - B_{11}\dot{\theta}_{11}, \\
J_{12}\ddot{\theta}_{12} + K_{s1}\theta_{12} - K_{s1}\theta_{11} &= -B_{12}\dot{\theta}_{12}
\end{aligned}
$$

which is obtained according to its kinetic and potential energy using Euler-Lagrange method. The derivation of the equations can be found in the Appendix to these lab notes. You will need to study and understand it.

The values of the relevant physical parameters for Stage 1 are as follows:

$$
\begin{aligned}
J_{11} &= 63.73091 \times 10^{-3} \text{ kg-m}^2, \\
J_{12} &= 230.41858 \times 10^{-3} \text{ kg-m}^2, \\
K_{s1} &= 9.0 \text{ N-m/rad}, \\
K_{t1} &= 8.925 \text{ N-m/A}, \\
B_{11} &= 4.5 \text{ N-s/m} \\
B_{12} &= 70.364 \times 10^{-3} \text{ N-s/m}.
\end{aligned}
$$

5.2 State-space equations for the first stage

Using the pair of Euler-Lagrange equations above, we can construct a state-space representation for the first stage. Define the state vector $x = (x_1, x_2, x_3, x_4)^T$, with $x_1 = \theta_{11}$, $x_2 = \theta_{12}$, $x_3 = \dot{\theta}_{11}$, and $x_4 = \dot{\theta}_{12}$, and take as input $u_1 = I_{m1}$. Here we are concerned with controlling the state with state feedback control, so only the equation for the evolution of the state is of actual interest. However, to set up a state space model in Matlab, the matrices C and D for the state-space model still need to be specified. In this case, we assume that all state variables are available at the output of the system (this is actually the case for the robot), and set the output to be $y_1 = x$, i.e., $C = I_{4\times4}$ and $D = 0_{4\times1}$.

Pre-lab Exercise 1: Starting from the Euler-Lagrange equations and using the values of the parameters given in Section 5, derive the following state-space representation for the system with state x, input u_1, and zero output, as specified above:

$$
\begin{aligned}
\dot{x} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -141.21876526 & 141.21876526 & -70.60937934 & 0 \\ 39.05935016 & -39.05935016 & 0 & -305.37467942 \times 10^{-3} \end{bmatrix} x \\
&\quad + \begin{bmatrix} 0 \\ 0 \\ 140.04193569 \\ 0 \end{bmatrix} u_1, \\
y_1 &= x.
\end{aligned}
$$

Remarks:

1. The calculated results in Matlab may have slightly different numbers at long decimal points for double-precision floating-point data. There are two main causes: the first is that the data has been truncated manually during the calculation process, such as the numbers printed for the equation above only has eight digits after the decimal point, although they have much longer digits in Matlab storage space. Similarly, you may record and use the values in between the calculations from the Matlab display, which has only four digits after decimal point, as the default format is short, therefore truncating the data further. The second is because the hardware has a limited number of bits to represent a real number in different computers, therefore the resulting value is necessarily truncated.

2. In this lab the second stage will be fixed to the first stage, so the control will be enacted only on the first stage actuated by Drive #1. A similar state-space model can be written down for the second stage but since we will not be controlling the second stage, we will not need this model and discuss it here.

6 Control task and LQR design

As discussed in Lecture 7, an LQR controller is a stabilising controller. That is, it brings the state of the system asymptotically back to 0. The purpose of this lab is to demonstrate the application of the LQR controller to "reset" the position of the robot arm that has been moved from its initial position (which serves as the zero initial state) back to this position. Note that LQR controllers can also be used to move the state from a zero initial state to another state, but this requires something more than what was covered in Lecture 7 and will be treated in Lecture 10 (trajectory tracking). So, for this lab you will focus only on using LQR to bring the state back to a zero state condition.

6.1 LQR design

The task will be to reset the angles θ_{11} and θ_{12} to 0 from initial non-zero positions. The system is set to be initially positioned with $\theta_{11} = \theta_{12} = \pi/12$ rad (15°), and $\dot{\theta}_{11} = \dot{\theta}_{12} = 0$ rad/s. The task will be achieved by implementing a linear quadratic regulator to drive Stage 1 to a desired configuration with θ_{11}, θ_{12}, $\dot{\theta}_{11}$, and $\dot{\theta}_{12}$ all equal to 0. The control signal is the output current to Drive 1, and it is desired to limit its continuous output current to not more than 0.944 A.

Before designing an LQR controller for each stage, you should check if the dynamic model for each stage is controllable.

Pre-lab Exercise 2: Compute the controllability matrix for the first stage model obtained in Pre-lab Exercise 1. What is the determinant and rank of this matrix? (use the Matlab command `det` and `rank`).

From the above exercise, you should find that the system is controllable. However,

from a first glance it is easy to think that it is not. The reason it may not seem controllable is that θ_{12} is coupled to θ_{11} via a spring and cannot be independently actuated. If the system is stable and θ_{11} settles to the value $\theta_{11,ss}$ then θ_{12} will be forced to settle to the same value by the springs. To force θ_{12} to settle to a different value would require an additional actuator, such another motor, to act on θ_{12}, which is not available. So, while it is possible to move the system to a state of the form $(c_1, c_2, 0, 0)^T$ with $c_1 \neq c_2$, the system *cannot* be kept in such a state *indefinitely*. Controllability only guarantees that there is a control law that brings the system to that state at any desired time $t_f > 0$, there is *no* requirement to be able to keep it there. Whenever $c_1 \neq c_2$, it can be shown that there will not exist a control law to keep the system at the state $(c_1, c_2, 0, 0)^T$ (try to prove this for yourself). The only possibility is that the system will immediately leave this state once it reaches it and an appropriate control law can bring it back there in a finite but *non-zero* time. So, the best that can be done will be a back and forth movement to and from the state $(c_1, c_2, 0, 0)^T$. However, in this lab $c_1 = c_2$ so this difficulty is not encountered.

To design an LQR controller, we need to set the matrices Q and R, they are design parameters. Since we are interested in forcing θ_{11} and θ_{12} to zero from a non-zero initial condition $\pi/12$ rad $= 15^o$, while $\dot{\theta}_{11}$ and $\dot{\theta}_{12}$ are initially at 0, using a heuristic known as *Bryson's Rule*, one possible choice for Q is,

$$Q = \begin{bmatrix} \frac{\bar{q}_1}{(\pi/12)^2} & 0 & 0 & 0 \\ 0 & \frac{\bar{q}_2}{(\pi/12)^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where $\bar{q}_1$ and $\bar{q}_2$ are some nonnegative constants. Note that for this choice of Q we have that

$$x^T Q x = \frac{\bar{q}_1}{(\pi/12)^2}\theta_{11}^2 + \frac{\bar{q}_2}{(\pi/12)^2}\theta_{12}^2.$$

That is, θ_{1j} is weighted by $\bar{q}_j$ multiplied with the reciprocal of the square of its initial conditions, for $j = 1, 2$. For R, with Bryson's Rule we set $R = \frac{\bar{R}}{0.944^2}$, with $\bar{R}$ a real positive constant. Thus, u^2 will be weighted by $\bar{R}$ multiplied by the reciprocal of the square of the maximum continuous Drive 1 current allowed, which is 0.944 A.

The overall infinite horizon LQR cost to be minimised in the design is therefore

$$\begin{aligned} J_\infty(x_0) &= \int_0^\infty (x(s)^T Q x(s) + R u(s)^2) ds \\ &= \frac{\bar{q}_1}{(\pi/12)^2}\int_0^\infty \theta_{11}(s)^2 ds + \frac{\bar{q}_2}{(\pi/12)^2}\int_0^\infty \theta_{12}(s)^2 ds + \frac{\bar{R}}{0.944^2}\int_0^\infty u(s)^2 ds. \end{aligned}$$

Assessment Exercise 1 (4 points, assessed at 45 minutes):

1. Create a new Matlab script named **eqn_robot_stage1.m** to set up your state-space model and to design your LQR controller.

2. Use the Matlab command `ss` in your Matlab script to set up a state-space model for Stage 1 using the formulas you used in Pre-lab Exercise 1. Name the system matrices in your model, A, B, C, and D. Note that individual robots can have slightly different parameter values J_{11}, J_{12}, K_{s1}, K_{t1}, B_{11} and B_{12}, so the A, B, C, D matrices will be different for each robot. In particular, the A, B, C, D matrices you will compute based on these parameters will have numerical entries different from those in Pre-Lab Exercise 1. **Please ask your demonstrator for the parameter values of the robot that you are working on.**

6.2 LQR Control simulation

Assessment Exercise 2 (6 points, assessed at 1 hour 30 minutes):

1. In the same script **eqn_robot_stage1.m** as in **Assessment Exercise 1**, verify that $(Q^{1/2}, A)$ is an observable pair, and use the command `lqr` to design an LQR controller for $\bar{q}_1 = 2.5$, $\bar{q}_2 = 20$ and $\bar{R} = 2$, and store the multivariable feedback gains in a row vector named K1.

2. Type **simulink** in Matlab command window to open a new Simulink model. Create a closed-loop control structure including a state-space model of Stage 1 and its state feedback LQR controller as shown in Fig. 6, and add a setpoint and a scope as shown. Run script **eqn_robot_stage1.m** and start the simulation to observe the control action on the model of Stage 1 (state $x(t)$ and the control signal $u(t)$).

3. Fix $\bar{q}_1 = 2.5$ and individually vary the values of $\bar{q}_2$ and $\bar{R}$ (one at a time) from their nominal values, e.g. $\bar{q}_2$ =15 to 25 and $\bar{R} = 3.5$ down to 0.5. Recompute the LQR controller for each new values of $\bar{q}_2$ and $\bar{R}$, checking each time that $(Q^{1/2}, A)$ is observable (for each choice of $\bar{q}_2$ and $\bar{R}$, one different feedback gain vector K1 is obtained). Attach the LQR controller in a feedback configuration with Stage 1 in Simulink, and simulate the closed-loop system with the initial state specified earlier. Study the influence of the weights on how the angular displacements θ_{11} and θ_{12} vary with time. Make sure to observe the control output signal as well. effort as well. Provide plots for these and state the observed influences with meaningful explanations. There should be be at least five plots in total (nominal, $15 \leq \bar{q}_2 < 20$ and nominal R, $20 < \bar{q}_2 \leq 25$ and nominal R, nominal $\bar{q}_2$ and $2 < \bar{R} \leq 3.5$, and nominal $\bar{q}_2$ and $0.5 \leq \bar{R} < 2$).

7 Control implementation and experiment

The final Simulink Desktop Real-Time program required to implement the state-space feedback control is shown in Fig. 7. As you can see there are two parallel control loops in it - both has the same structure and the same state-space feedback controller, except the top section is for the actual robot, while the bottom section is for the theoretical model.

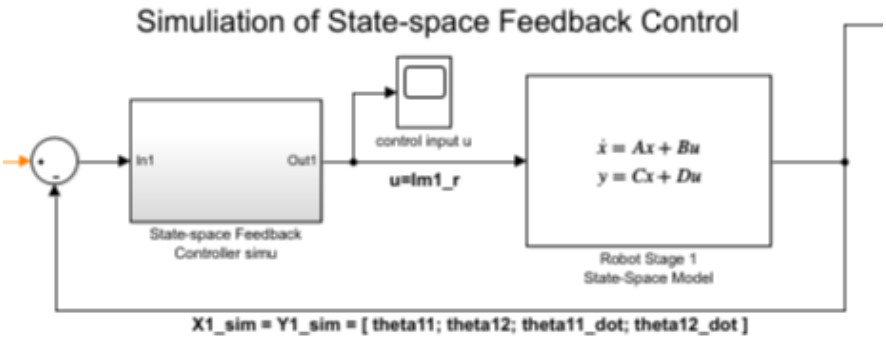

Figure 6: Simulink model for stage 1 control simulation

It is a very useful structure which allows you to carry out the control simulations as well as the actual control on the robot arm at the same time (good for result comparison and system analysis).

A pre-configured template **elec4631_robot_stage1.slx** which contains the control structure for the robot in the top section, the setpoint and scopes, as well as a function block at the bottom left for locking up the robot stage 2 (ELB), is located in the folder Documents:\MATLAB. You will need to plug-in the control simulation blocks you have created in **Assessment Exercise 2** to complete this control system. Note that some signal lines here are multi-variable. You should save this Simulink program in your own file name for further use.

7.1 Controller implementation

Open the template **elec4631_robot_stage1.slx**. Integrate the simulation model you have just created to the bottom section of the template (see Fig. 7). Once completed save this model in your own file name.

7.1.1 Assign parameters to all variables in Simulink model

You will need to assign parameters to all variables in the system and control gains first.

1. Open and Run Matlab script **setup_robot_parameters.m**, located in Documents \MATLAB, to assign the physical parameters of the robot to the variables used in the system. You can read this m file for details.

2. Run your script **eqn_robot_stage1.m** to define the state-space model, as well as your LQR controller, in Workspace. Double check that A, B, C, D and K1 are there.

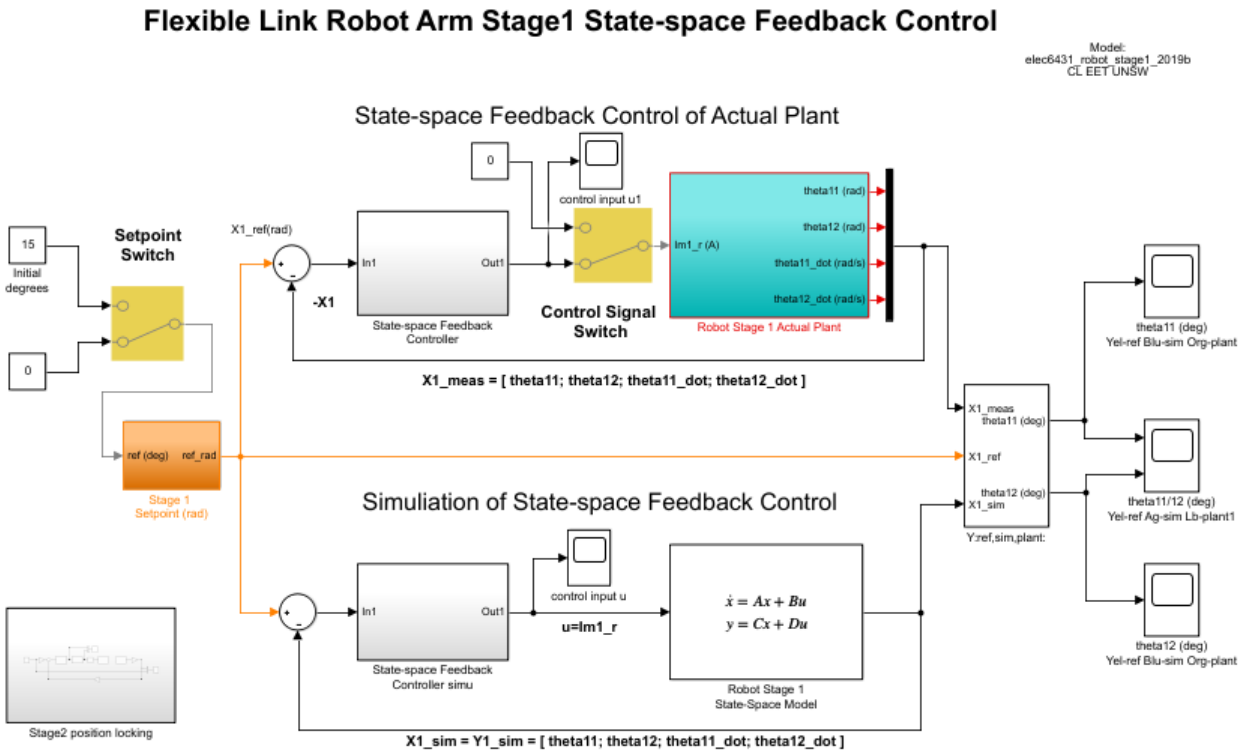

Figure 7: Simulink model for stage 1 control experiment

7.1.2 Test run the control program (the robot power is off)

You are required to to verify your controller using the simulation loop only, before applying it to the actual robot arm system.

1. Disable the control output to the robot in the top section: Make sure that the **Control Signal Switch** is switched to **Constant 0** (double-click it to flip over).

2. Set the setpoint to Zero: flip the **Setpoint Switch** on the left of the model to **Constant 0**.

3. Run simulation only (Nothing should happen to the real robot arm): click the **Run** button. Open the scopes. Flip the **Setpoint Switch** from **constant 0** to **15**, the simulated model of Stage 1 will move away from zero and settles at the initial position to at 15 degrees. Flipping the **Setpoint Switch** to **Constant 0** will bring the angle back to 0 degree.

   You should have a number of sets of K1 designed in Assessment Exercise 2, verify them here on the robot. To do this, each time you want to test a new value of K1 you will need to stop the simulation, assign the new value to K1, reconnect to the target, and run the simulation again.

4. Show your simulation results to your demonstrator. And check with him that if you can start the control experiments on the real robot.

7.2 Control experiments

Assessment Exercise 3 (10 points, assessed at 2 hours 0 minutes): Now you are ready to run the full control experiments. Before the power of robot can be switched on make sure that the control program is not running and the **Setpoint Switch** is on **Constant 0**, and **Control Signal Switch** is connected back to the robot arm.

1. Ask a demonstrator to come to your robot desk and switch on the power of the robot for you.

2. Run at Desktop Real-Time mode. The robot should not move as the Setpoint is zero. Flip the **Setpoint Switch** to **Constant -15** so the Stage 1 arm will been driven away from its central position and let the dynamics settle. This step sets the first stage to an initial position of -15^o for the LQR controller to reset back to the central position of 0^o. Observing it on the scopes, it may take a while to return to the central position depending on how fast the system will settle. Note that the plots now show both the results of simulation (theoretical) and actual plant responses.

3. Flip the **Setpoint Switch** back to **Constant 0**, the LQR controller will turn the Stage 1 arm back to its initial central position with transients. Record the plots observed on the scopes in your lab notebook.

4. Repeat the experiment using the different K1 that you have designed, and record the results (for each new controller gains you will need to stop the program, assign the new value to K1, and re-run the controller again). In your lab notebook, make observations and analysis regarding the effects of varying LQR parameters $\bar{q}_2$ and $\bar{R}$, as well as comparing the simulation results and the actual experimental results.

7.3 Extended control experiments

1. You may have noticed that the position of stage 1 is not converging to zero compared to the simulation results. It may have either static errors or oscillations around zero. Please discuss with your demonstrator for what may have caused this.

2. **Under a demonstrator's guidance**, try more different weights with higher values of $\bar{q}_2$ and lower values of $\bar{R}$). It may lead to better control results. However this may as well cause the control output (which is capped to 0.944 amp) to saturate. What is your conclusion? Discuss your thoughts with your demonstrator.

8 Appendix

8.1 Kinetic and potential energy

The first stage has both kinetic and potential energy. The potential energy of the first stage, $\mathcal{P}_1$, is given by

$$\mathcal{P}_1 = \frac{1}{2}K_{s1}(\theta_{12}(t) - \theta_{11}(t))^2.$$

The kinetic energy $\mathcal{K}_1$ has two components, $\mathcal{K}_{11}$ and $\mathcal{K}_{12}$. The first component, $\mathcal{K}_{11}$, is the rotational energy of the first motorized flexible joint and is given by

$$\mathcal{K}_{11} = \frac{1}{2}J_{11}\dot{\theta}_{11}^2,$$

where J_{11} is the harmonic Drive #1 equivalent output load inertia. The second component, $\mathcal{K}_{12}$, is the rotational kinetic energy of the first link compounded with the second link assembly, and is given by

$$\mathcal{K}_{12} = \frac{1}{2}J_{12}\dot{\theta}_{12}^2,$$

where J_{12} is the first link load inertia compounded with the second link assembly. Thus the total kinetic energy of the first stage is:

$$\mathcal{K}_1 = \mathcal{K}_{11} + \mathcal{K}_{12} = \frac{1}{2}J_{11}\dot{\theta}_{11}^2 + \frac{1}{2}J_{12}\dot{\theta}_{12}^2.$$

8.2 Generalised forces

There will be generalised forces Q_{21} and Q_{22} acting on each of the generalized coordinates θ_{11} and θ_{12}, respectively. They are given by:

$$\begin{aligned} Q_{21} &= T_1 - B_{11}\dot{\theta}_{11} \\ Q_{22} &= -B_{12}\dot{\theta}_{12}, \end{aligned}$$

where $T_1 = K_{t1}I_{m1}$ is the torque produced by Drive #1 at the load (i.e., the load driving torque). Here, K_{t1} is the Drive #1 torque constant and I_{m1} is the Drive #1 actuating current.

8.3 Lagrangian and equations of motion

The equations of motion for the generalised coordinates θ_{11} and θ_{12} can be determined via the Lagrangian $\mathcal{L}_1$ defined by

$$\mathcal{L}_1 = \mathcal{K}_1 - \mathcal{P}_1 = \frac{1}{2}J_{11}\dot{\theta}_{11}^2 + \frac{1}{2}J_{12}\dot{\theta}_{12}^2 - \frac{1}{2}K_{s1}(\theta_{12}(t) - \theta_{11}(t))^2.$$

The equations of motion for θ_{11} and θ_{12} are then given by the Euler-Lagrange equations:

$$\frac{\partial}{\partial t}\left(\frac{\partial L_1}{\partial \dot{\theta}_{11}}\right) - \left(\frac{\partial L_1}{\partial \theta_{11}}\right) = Q_{11},$$
$$\frac{\partial}{\partial t}\left(\frac{\partial L_1}{\partial \dot{\theta}_{12}}\right) - \left(\frac{\partial L_1}{\partial \theta_{12}}\right) = Q_{12}.$$

Thus the Euler-Lagrange equations for the first stage are thus:

$$\begin{aligned} J_{11}\ddot{\theta}_{11} - K_{s1}\theta_{12} + K_{s1}\theta_{11} &= K_{t1}I_{m1} - B_{11}\dot{\theta}_{11}, \\ J_{12}\ddot{\theta}_{12} + K_{s1}\theta_{12} - K_{s1}\theta_{11} &= -B_{12}\dot{\theta}_{12} \end{aligned}$$

(**Note: You could also have directly derived these equations of motion directly using the methods explained in Chapter 2 of Nise's text for ELEC3114 on rotational mechanical systems. Try this!**)

(end of elec4631 lab 3 note)