FRTN65 Modeling and Learning from Data

Laboratory Exercise 2

Modeling and Simulation of Furuta Pendulum

Department of Automatic Control LTH
Lund University
Fall 2023

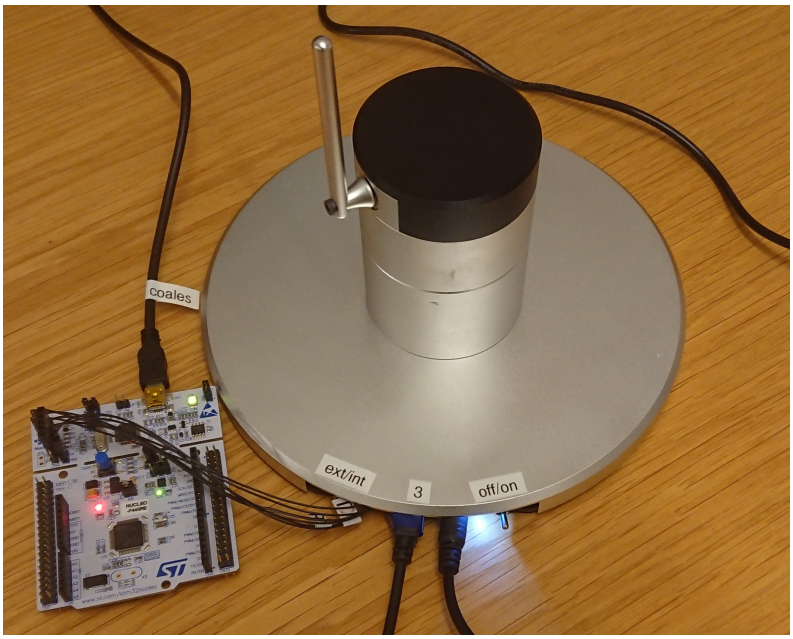

Figure 1 The Furuta pendulum

Hand in your final modelica code and a brief report in PDF format with results and short discussions for each task. You do not have to complete the parts marked as extra. You may ask for help and cooperate on the laboration, but the handin should be done individually. (Your are e.g. not allowed to copy material from another persons' report).

Introduction

In this lab you will be tasked with completing and extending an already existing Modelica model of the Furuta pendulum shown in Figure 1.

The purpose of this lab is for you to get some hands-on experience in working with object-oriented equation-based languages, so don't spend too much time trying to get the model and its parameters perfect. Before starting, it is highly recommended that you complete Exercise 8.

IMPORTANT: When working with Impact, please download your models and save them on your personal computer often! The workspaces saved on one remote computer will not be accessible on other remote computers, and the files will be purged from time to time. If you have not saved the files on your own personal computer, you risk to loose that work.

1. Hands-on parameter tuning

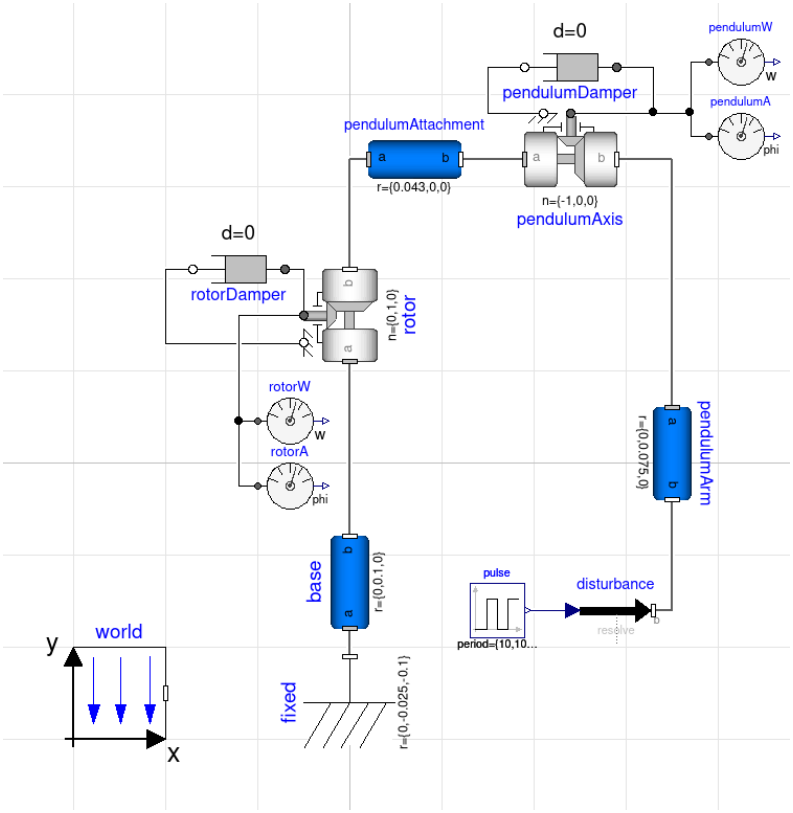

Figure 2 The given Modelica diagram

The supplied Modelica code implements a simple, yet accurate model of the Furuta pendulum, where frictions and drag are modeled as two dampening constants of the torques on the rotor and pendulum bearings. The Modelica block diagram can be seen in Figure 2, and a snapshot from the animation window in Figure 3.

Unfortunately, the dampening constants are not known and needs to be estimated in some manner. To do this, we have performed an experiment on the real Furuta pendulum were the pendulum was released from an almost upright position, and the angles of the rotor and pendulum bearings logged over time. The experiment data is plotted in the first part of the supplied notebook.

Task 1.1: Upload `FurutaPendulum.mo` to your workspace in Impact and identify and set the two dampening constants by simulating the model and comparing with the experimental data.

Hint: The initial angle can be set in the `pendulumAxis` *block under "Variables -> phi -> start".*

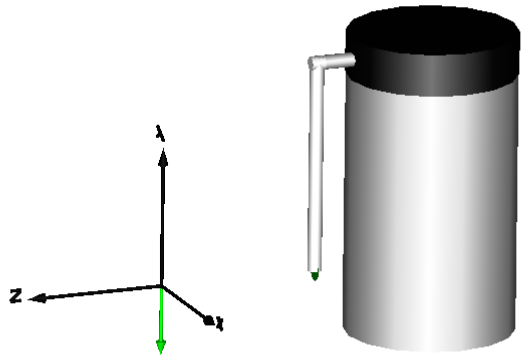

Figure 3 Image of the Furuta pendulum from the animation window

By changing the amplitude in the block `pulse`, external disturbances can be added to the pendulum.

Task 1.2: Set the initial pendulum angle to its original downright position, and try a disturbance with amplitude `{0.025, 0, 0}` and plot how the pendulum and rotor angles change over time. Try two more amplitudes of your choice.

2. Adding an additional pendulum arm

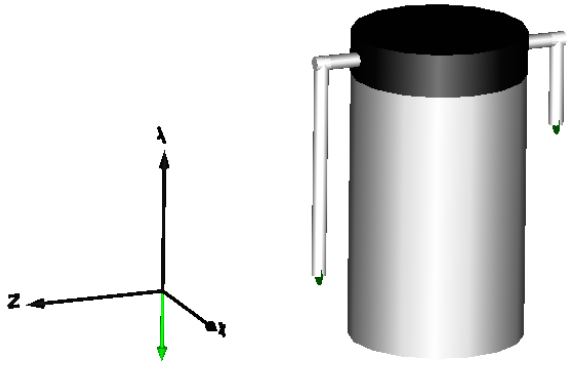

Figure 4 Image of the two-armed Furuta pendulum from the animation window

One powerful feature of having a good model, is that it becomes easy to try out extensions that are difficult and time consuming to perform on the real

process. In this part we want to extend our Furuta pendulum model with a second pendulum at the opposite side of the first, as shown in Figure 4.

To do this, we need

- Two `Modelica.Mechanics.MultiBody.Parts.BodyCylinder` objects representing the second `pendulumAttachment` of length 43 mm and `pendulumArm` of length 30 mm. Both should have the same density and diameter as the components of the first pendulum.
- A `Modelica.Mechanics.MultiBody.Joints.Revolute` object representing the second pendulum bearing. To activate the flange, check the `useAxisFlange` box in the settings window of the block.
- A `Modelica.Mechanics.Rotational.Components.Damper` object connected to the flange of the second pendulum bearing, with a dampening constant half as large as for the first pendulum bearing.
- Two `Modelica.Mechanics.Rotational.Sensors.{AngleSensor, SpeedSensor}` objects connected to the `axis` flange of the second pendulum bearing.
- A `Modelica.Mechanics.MultiBody.Forces.WorldForce` object connected to the end of the second pendulum, to enable the generation of external disturbances on this pendulum. Connect the `force[3]` connector to the same `pulse` block as the disturbance for the first pendulum, make sure you choose the option `[:]` in both drop-down menus to correctly connect all the dimensions in the connector.

IMPORTANT: For part 3 the second pendulum axis needs to have the same axis of rotation as the original, i.e. clock-wise in regards to a vector pointing to the central rotor. This is dictated by the parameter `n` in the `Revolute` block.

Task 2.1: Extend the existing model with this second pendulum. From an initial downright position of the two pendulums, simulate the model with the same disturbances as the previous part and plot how the two pendulum and the rotor angles change over time.

Hint: The second pendulum is easiest implemented with blocks in the Diagram window. Look at how the first pendulum is implemented, the second pendulum can be created almost identically.

3. Adding a damping controller

The two pendulums are quite sensitive to external disturbances, which can introduce a lot of oscillation on the arm positions. In this part we will demonstrate how a model can be used to perform initial control design for a real process, by implementing a state feedback controller to dampen these oscillations.

In state feedback the control signal $u(t) = -Lx(t)$ is directly proportional to the states via some feedback gain vector L, which we here will generate via a Linear-Quadratic Regulator (LQR). Given a linear state-space representation

$$\dot{x}(t) = Ax(t) + Bu(t)$$

the LQR provides a powerful way to generate the optimal control signal $u(t)$ such that the cost

$$J = ||x||_Q^2 + ||u||_R^2 = \int_0^\infty \Big(x(t)^T Q x(t) + u(t)^T R u(t) \Big) dt$$

is minimized. Here Q and R are tuneable cost matrices.

The dynamics of the Furuta pendulum is however not linear, but we can use the steps in part 7 of this technical report[1] to linearize it around the bottom position, yielding for the states $x = \Big(\phi, \dot{\phi}, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2 \Big)$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{d_1 c_1}{a_1 b_1 - c_1^2} & 0 & -\frac{d_2 c_2}{a_2 b_2 - c_2^2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{a_1 d_1}{a_1 b_1 - c_1^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\frac{a_2 d_2}{a_2 b_2 - c_2^2} & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \frac{b_1}{a_1 b_1 - c_1^2} + \frac{b_2}{a_2 b_2 - c_2^2} \\ 0 \\ \frac{c_1}{a_1 b_1 - c_1^2} \\ 0 \\ \frac{c_2}{a_2 b_2 - c_2^2} \end{pmatrix}$$

where $a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2$ are parameters depending on e.g. the weight or length of the pendulums, $\phi, \dot{\phi}$ the angle/angular velocity of the central rotor and $\theta, \dot{\theta}$ the angle/angular velocity of the first and second pendulum bearings respectively.

Your task is to implement the LQR controller for the double Furuta pendulum model. To your help, consider the following hints.

1. The supplied notebook contains the necessary code to generate a LQR gain vector for the double pendulum given some Q and R.
2. The controller needs to be able to actuate torque on the central rotor, add a `Modelica.Mechanics.Rotational.Sources.Torque` object and connect it to the `axis` flange on the `rotor` block. The `torque` block has a `RealInput tau` to connect $u(t)$ to.
3. In `FurutaPendulum.mo` there is an almost complete nested controller model. Complete it, create an instance and connect its output to `torque.tau` and its inputs to the correct sensor values.

Task 3.1: Implement the controller and simulate the model with the same disturbances as in part 1 and 2. Plot how the two pendulum and rotor angles change over time.

Feel free to try some different values of Q and R in the LQR design script.

Hint: It is a possibility that implementations of the controller can give an unstable system, e.g. via some coding error or if the disturbance is too large. Unfortunately, this can cause Impact to get stuck when simulating. A quick fix is to switch to the ***`Radau5ODE`*** *solver under "Experiment -> Advanced -> Solver", which instead throws an error in these cases. However, if Impact do get stuck, you can simply revisit* ***`JupyterHub`****, stop the server, and start it again.*

[1] `https://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTN10/2020/Gafvert1998.pdf`

4. (Extra) Adding a stabilizing controller

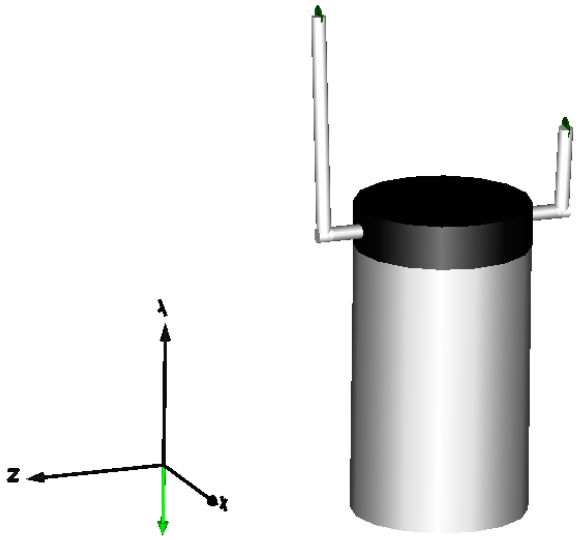

Figure 5 Image of the inverted two-armed Furutapendulum from the animation window

Controlling the pendulums in a downright position is not very hard, as the system is stable around this point. Instead you can consider the much harder task of balancing the pendulums in an upright position, as shown in Figure 5.

However, performing this feat for our Furuta pendulum model is actually not that difficult, but some changes have to be made in both the Modelica model, and the linearized model from part 3. More precise, you will have to

1. Re-linearize the model for the upright position to generate a new L vector. Here you can consider part 7 of the technical report, where it is clear that the linearizations around $(0, 0, \pi, 0, \pi, 0)$ and $(0, 0, 0, 0, 0, 0)$ differ only with a handful of signs (which?).
2. Trim Q and R, good initial values to try are $Q = I$ and $R = 1$.
3. Update your setpoint in the controller model.
4. Update the initial angles of the pendulum bearings to some value very close to 0.
5. Set the disturbance amplitude to some very small value, e.g. `{0.001,0,0}`.

Good Luck !