Laboratory Exercise 2 - Modeling and Simulation of Furuta Pendulum

Theodora Gaiceanu

1 Hands-on parameter tuning

1.1 Task 1.1

First, I changed the arm in the upright position. I did this by setting the start value of the *phi* parameter from the **pendulumAxis** block to 0. The result of the simulation can be seen in Figure 1.

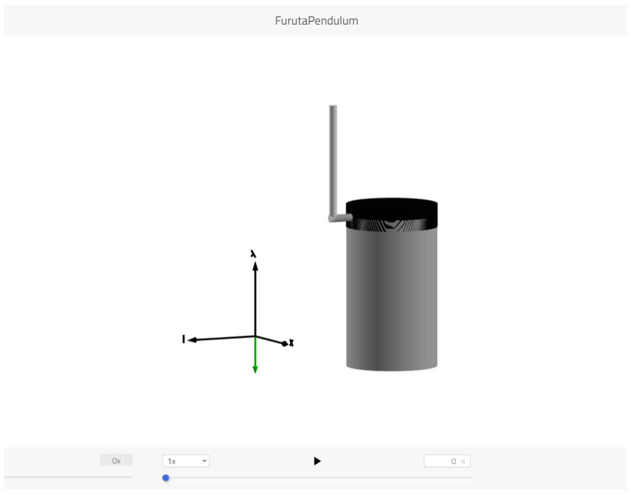
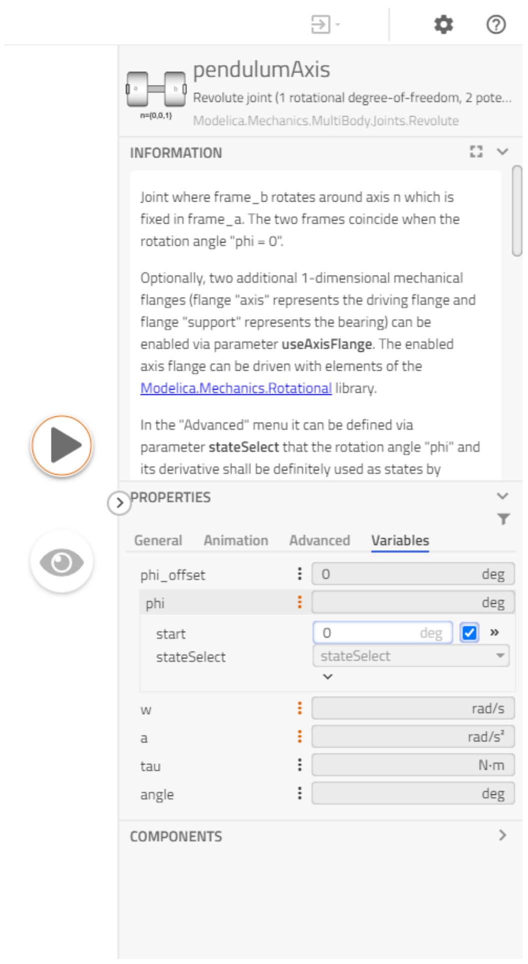


Figure 1: Simulation for upright position - simulation

But then I changed the value of the *phi* parameter from the **pendulumAxis** block to 180. So, right now the pendulum is again in the downright position. The result of the simulation can be seen in Figure 3.


pendulumAxis

Revolute joint (1 rotational degree-of-freedom, 2 pote...

Modelica.Mechanics.MultiBody.Joints.Revolute

INFORMATION

Joint where frame_b rotates around axis n which is fixed in frame_a. The two frames coincide when the rotation angle "phi = 0".

Optionally, two additional 1-dimensional mechanical flanges (flange "axis" represents the driving flange and flange "support" represents the bearing) can be enabled via parameter **useAxisFlange**. The enabled axis flange can be driven with elements of the Modelica.Mechanics.Rotational library.

In the "Advanced" menu it can be defined via parameter **stateSelect** that the rotation angle "phi" and its derivative shall be definitely used as states by

PROPERTIES

General | Animation | Advanced | Variables

phi_offset: 0 deg
phi: deg
start: 0 deg ✓
stateSelect: stateSelect
w: rad/s
a: rad/s²
tau: N·m
angle: deg

COMPONENTS

Figure 2: Parameter for upright position

Completion after feedback From the given data, it can be noticed that both the rotor and the pendulum angle have dampening constants that enable the pendulum to drop its arm, but the system does not have a stable behaviour. The signal is oscillating, being unable to keep a constant value. Therefore, when tuning the dampening constants, I searched for values that should be small enough to enable the pendulum to keep oscillating. In other words, the pendulum needed to be first in the upright position, then, after applying a disturbance, the arm should go down, but it should continue to move from left to right during the experiment.

Consequently, after running some experiments, I chose the value of 0.05

for the dampening constant of the rotor. And I chose 0.000001 for the dampening constant of the pendulum. With these dampening constants, the arm went down and it was moving from one way to the other all the time. These parameters can be set by changing the d parameter from the **rotorDamper** block and **pendulumDamper** block. The values can be observed in the block diagram, in Figure 5.

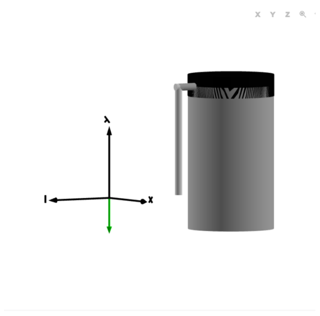
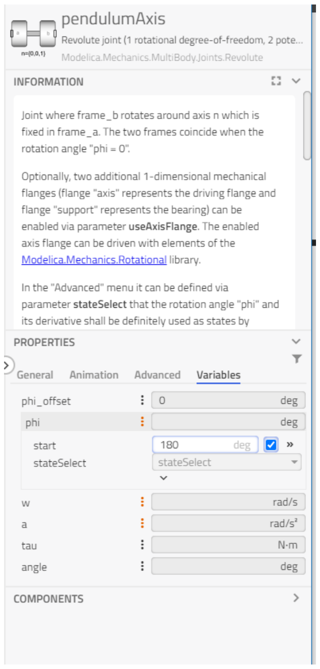


Figure 3: Simulation for downright position - simulation



pendulumAxis
Revolute joint (1 rotational degree-of-freedom, 2 pote...
Modelica.Mechanics.MultiBody.Joints.Revolute
n={0,0,1}

INFORMATION

Joint where frame_b rotates around axis n which is fixed in frame_a. The two frames coincide when the rotation angle "phi = 0".

Optionally, two additional 1-dimensional mechanical flanges (flange "axis" represents the driving flange and flange "support" represents the bearing) can be enabled via parameter **useAxisFlange**. The enabled axis flange can be driven with elements of the Modelica.Mechanics.Rotational library.

In the "Advanced" menu it can be defined via parameter **stateSelect** that the rotation angle "phi" and its derivative shall be definitely used as states by

PROPERTIES

General | Animation | Advanced | Variables

phi_offset	0	deg
phi		deg
start	180	deg ✓
stateSelect	stateSelect	
w		rad/s
a		rad/s²
tau		N·m
angle		deg

COMPONENTS

Figure 4: Parameter for downright position

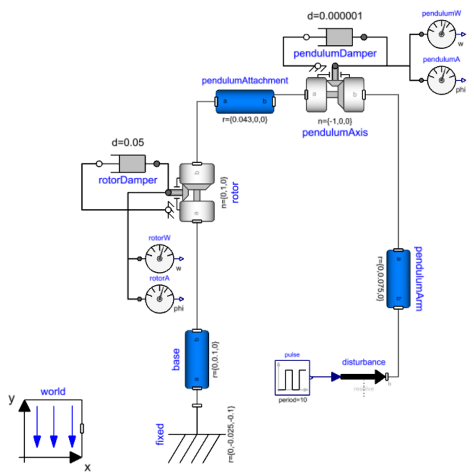
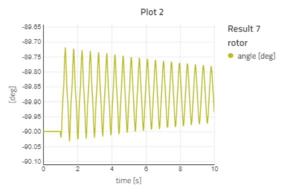


Figure 5: Block diagram with dampening

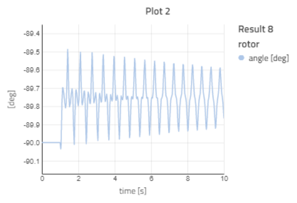
1.2 Task 1.2

Here, the amplitude of the disturbances can be changed by setting the first term of the *amplitude* parameter in the **pulse** block. Also, I tried two more amplitudes: 0.05 and 0.075. In the simulation, the arm moved higher and higher when I increased the amplitude.

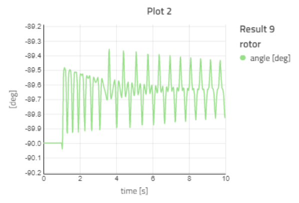
The behaviour of the rotor angles in time can be observed in Figure 6.



(a) amplitude 0.025



(b) amplitude 0.05

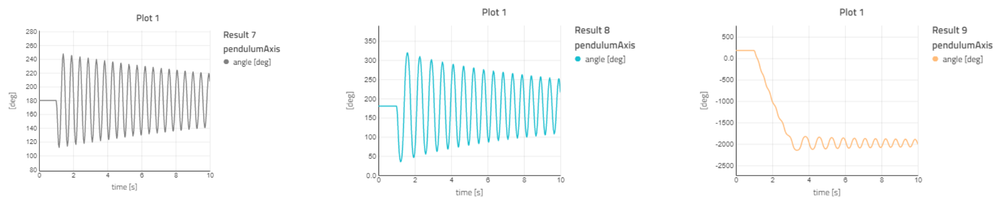


(c) amplitude 0.075

Figure 6: Plots for the rotor

The behaviour of the pendulum angles in time can be observed in Figure 7.

Completion after feedback From both Figures 6 and 7, it can be seen



(a) amplitude 0.025 (b) amplitude 0.05 (c) amplitude 0.075

Figure 7: Plots for the pendulum

that the system is unstable. The dampening constants by themselves are not enough to prevent the rotor or the arm from oscillating. Even with a small amplitude for the disturbance (Figure 6a and Figure 7a), the signal is oscillating. And, as it is expected, by increasing the amplitude of the disturbance, the signal is oscillating more. The last amplitude, of 0.075, is actually too high, as it can be noticed in Figure 7c. Here, the disturbance is so high, that the arm spins around.

2 Adding an additional pendulum arm

Here, one needs to add some new blocks that define the second pendulum. So, for the **pendulumAttachement2** block, it is necessary a **Modelica.Mechanics.MultiBody.Parts.BodyCylinder** object. For this object, the first component of the r parameter has to be -0.043 (in order to be in the opposite direction), the *diameter* needs to be 0.005, the *density* should be set to 3700 and the color should be changed also. Then, for the pendulum bearing, a **Modelica.Mechanics.MultiBody. Joints.Revolute** is needed. For this object, one has the activate the option *useAxisFlange* and to set n to the negative x axis. A **Modelica.Mechanics. Rotational.Components.Damper** object is also necessary, an the d parameter should be set to 0.0000005. Moreover, for the second pendulum bearing, there are needed two sensors, represented by **Modelica.Mechanics. Rotational.Sensors.{AngleSensor, SpeedSensor}**. These sensors should be connected to the axis flange of the pendulum bearing. Then, the arm of the second pendulum is represented by another **Modelica.Mechanics. MultiBody.Parts.BodyCylinder** object. For the arm, the first component of the r parameter needs to be 0.03, the *diameter* should be 0.005, and the *density* should be 3700.

The expanded pendulum can be observed in Figure 8.

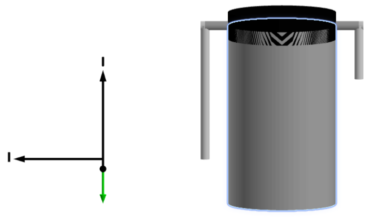


Figure 8: Pendulum with two arms

Also, the schema for the expanded pendulum can be observed in Figure 9.

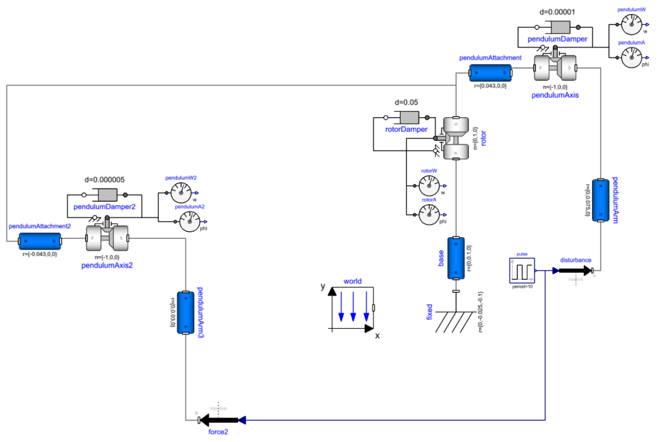
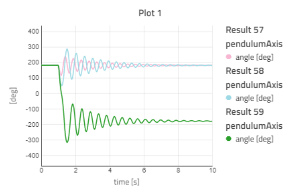
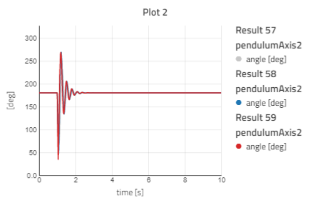


Figure 9: Schema for the pendulum with two arms

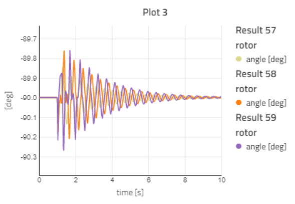
The behaviour of the two pendulum and rotor angles over time can be analysed in Figure 10. The same amplitudes as in Task 1 have been used for the disturbance. So, as it can be observed, the two pendulums have bigger oscillations when the amplitude increases. This is also true for the rotor angle.



(a) First pendulum



(b) Second pendulum



(c) Rotor

Figure 10: Plots for the extended pendulum - without a controller

Completion after feedback From Figure 10a, it can be actually seen the same behaviour as in the previous case, where the pendulum had only one arm. For the amplitudes 0.025 and 0.05 (pink and blue lines), the signal just oscillates. But for amplitude 0.075, the arm spins around, the disturbance being too high (green line). For the second arm, changing the amplitude of the disturbance does not really make a big difference, as it can be observed in Figure 10b. The rotor has also a similar behaviour compared to the one arm case. By analysing Figure 10c, the signal is oscillating more when the amplitude increases (yellow line - 0.025, orange line - 0.05, purple line - 0.075).

3 Adding a damping controller

First of all, as the second pendulum axis needs to have the same axis of rotation as the original, the **n** parameter in the **Revolute** block for the second pendulum should be set to the positive x axis. Also, the **torque** block needs to be connected to the **axis** flange on the **rotor** block. Therefore, the final schema can be noticed in Figure 11.

Also, the source code needs to be completed. The equation $u(t) = -Lx(t)$ should be added inside the code of the LQR controller. Also, L should be initialised with the values from the notebook. Moreover, the controller needs to be connected to the right inputs and output. The new parts inserted in the code can be seen in the Listing below.

```
model FurutaPendulum "Furuta pendulum"

   parameter Real pendulumA_start = -Modelica.Constants.pi;
```

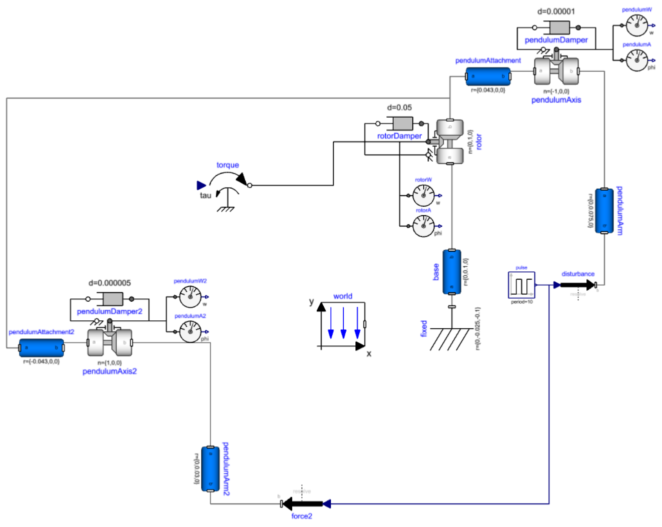
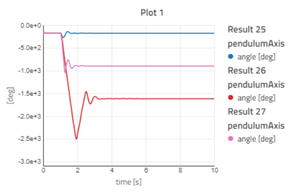


Figure 11: Final schema for the pendulum with two arms

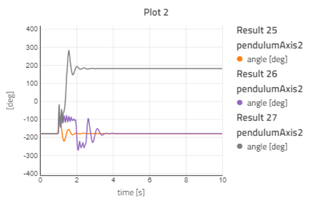
```

 model ControllerLQR
   Modelica.Blocks.Interfaces.RealInput phi, dphi, theta1,
 dtheta1, theta2, dtheta2;
   Modelica.Blocks.Interfaces.RealOutput u(start=0);
   Real x[6];
   Real L[6];
 equation
   x = {phi+3.14/2, dphi, theta1+3.14, dtheta1, theta2
+3.14, dtheta2};
   L = {100, 9.65721819, 103.07438354,  -1.40000991,
87.40106372,  -3.88918398};
   u = -L*x;
 end ControllerLQR;

 ControllerLQR controller_1;

 ...

 //phi, dphi, theta1, dtheta1, theta2, dtheta2
 connect(rotorA.phi, controller_1.phi);
 connect(rotorW.w, controller_1.dphi);
 connect(pendulumA.phi, controller_1.theta1);
 connect(pendulumW.w, controller_1.dtheta1);
 connect(pendulumA2.phi, controller_1.theta2);
```

```
    connect(pendulumW2.w, controller_1.dtheta2);

    connect(controller_1.u, torque.tau);
    //connect(controller_1.u, torque.tau);
    annotation (
      versionDate="2014-02-04",
      Commands(file="Furuta.mos" "Simulate Furuta pendulum",
   file="Animate.mos"
          "Animate Furuta pendulum"),
      experiment(NumberOfIntervals=5000, StopTime=10),
      Diagram(coordinateSystem(preserveAspectRatio=false,
   extent={{-100,-100},{100,
                100}}),     graphics),uses(Modelica(version =
   "3.2.3")));

end FurutaPendulum;
```
Listing 1: Source code

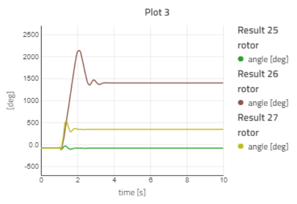
After completing the code and the schema, the same amplitudes for the disturbance have been used. As it can be noticed in Figure 12, for every amplitude the signal becomes stable now after a short transitory period.



(a) First pendulum



(b) Second pendulum



(c) Rotor

Figure 12: Plots for the stable pendulum

Completion after feedback Now, even for the amplitude of 0.075, the first arm succeeds to maintain a stable signal after 4 seconds (Figure 12a, pink line). From Figure 12b, it can be seen that the second arm also reaches a stable value for any disturbance, even though for amplitudes 0.05 (purple line) and 0.075 (grey line), the transitory period is a little longer than the one for the amplitude of 0.025. The rotor has the same stable behaviour. But, one thing can be observed from both the rotor and the arms: the signal is stable for any amplitude, but only for the amplitude of 0.025 the signal remains at the same value after the transitory period. For the other amplitudes, the signal changes its stationary value, as the pendulum moves more. Also, from the simulation, it can be observed that for the disturbance with amplitude 0.025, the pendulum moves its arms from left to right a little, then it stops. For the other disturbances, it stops also, but after moving its arms more. That means that the disturbances with amplitudes 0.05 and 0.075 are a little bit too high.

4 (Extra) Adding a stabilizing controller

First of all, I changed the values of the matrices Q and R, so that $Q = I$ and $R = 1$. The modified Python code can be observed in Listing 2. I also updated the new resulted values for L in the source code.

```
# Define the cost matrices
Q = np.zeros((6,6))
Q[0,0] = 1
#Q[1,1] = 1e-3
Q[1,1] = 1
Q[2,2] = 1
#Q[3,3] = 1e-3
Q[3,3] = 1
Q[4,4] = 1
#Q[5,5] = 1e-3
Q[5,5] = 1

R = np.zeros((1,1))
#R[0,0] = 1e-4
R[0,0] = 1
```

Listing 2: Trimming Q and R

Moreover, the initial angle of the two arms needs set to 0 (the upright position). This can be done by setting the *phi* variable from the **pendulumAxis** and **pendulumAxis2** blocks to 0. Also, the disturbance amplitude from the **pulse** block needs to be changed to a very small value (0.00001).

Lastly but not least, I increased the dampening constants. The first pendulum has a dampening constant of 0.1 and the second pendulum has a dampening constant of 0.05.

The new schema and the simulation can be seen in Figure 13 and Figure 14.

The behaviour of the two pendulum and of the rotor angles over time can be observed in Figure 15.

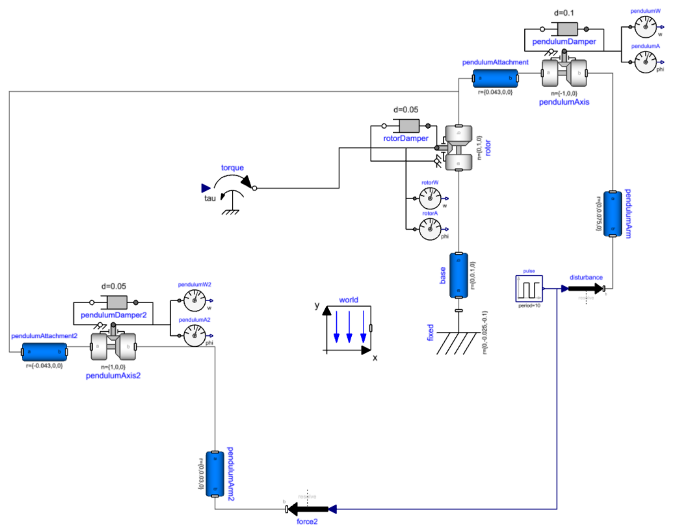


Figure 13: Schema for the pendulum with two arms - new dampening constants

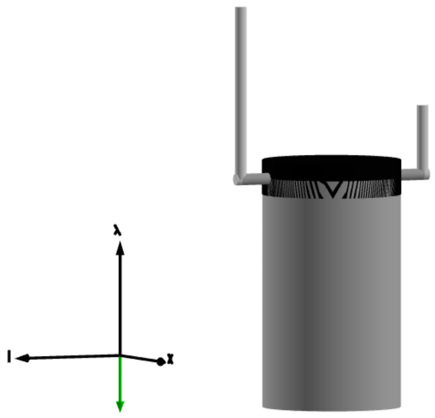
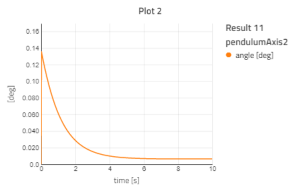
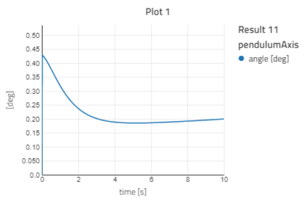


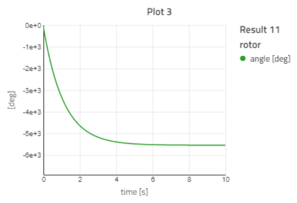
Figure 14: Simulation for the pendulum with two arms - upright position



(a) First pendulum



(b) Second pendulum



(c) Rotor

Figure 15: Plots for the new pendulum