Lund University
Faculty of Engineering
Name: ZHANG Yifei
Email: yi4840zh-s@lu.se

Date: November 17, 2023

Laboratory Exercise 2
Modeling and Simulation of Furuta Pendulum

Answer to Task 1: Hands-on parameter tuning

1. Task 1.1

   Adjust the initial angle of the pendulum so that the initial angle of the arm is facing upwards, as the figure 1 shows.

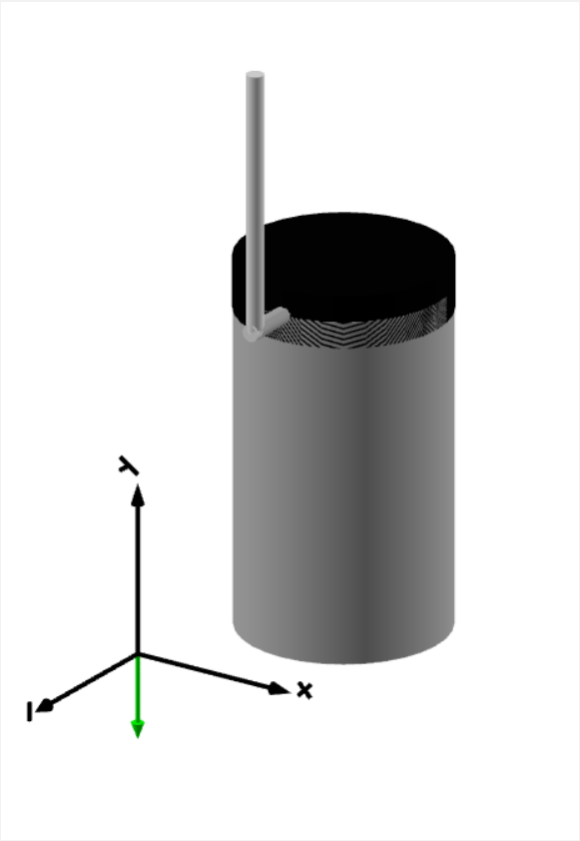

Figure 1: Task1.1 Pendulum

Adjust the parameters 'd' for 'rotorDamper' and 'pendulumDamper' and simulate. Observe the graphs of the 'ϕ' values for 'pendulumA' and 'rotorA' until the output graphs in the Jupyter notebook are similar. Obtain values for pendulumA's $d = 0.000005$ and rotorA's $d = 0.03$. The results are shown as follow.

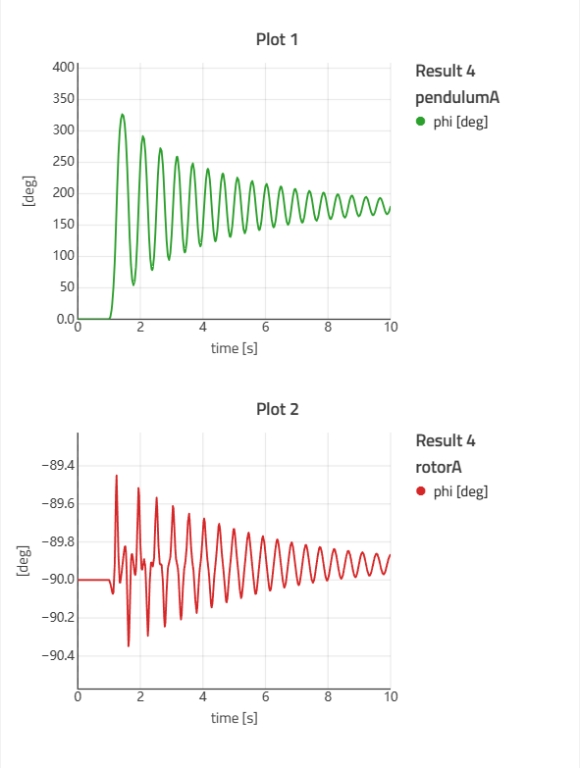

Figure 2: Task1.1 Results

The Modelica diagram of task1 is as figure 3.

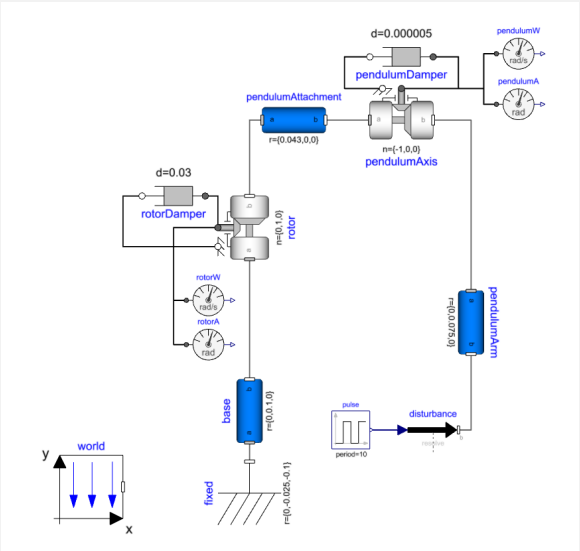

Figure 3: Task1 diagram

2. 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. Then try the amplitude 0.05, 0, 0 and 0.01, 0, 0.The results are shown as follow.

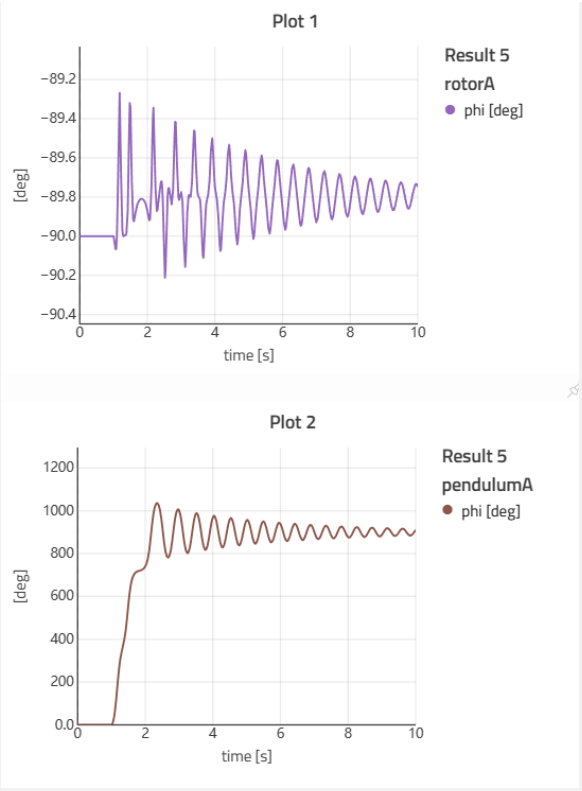

Figure 4: Task1.2 0.025

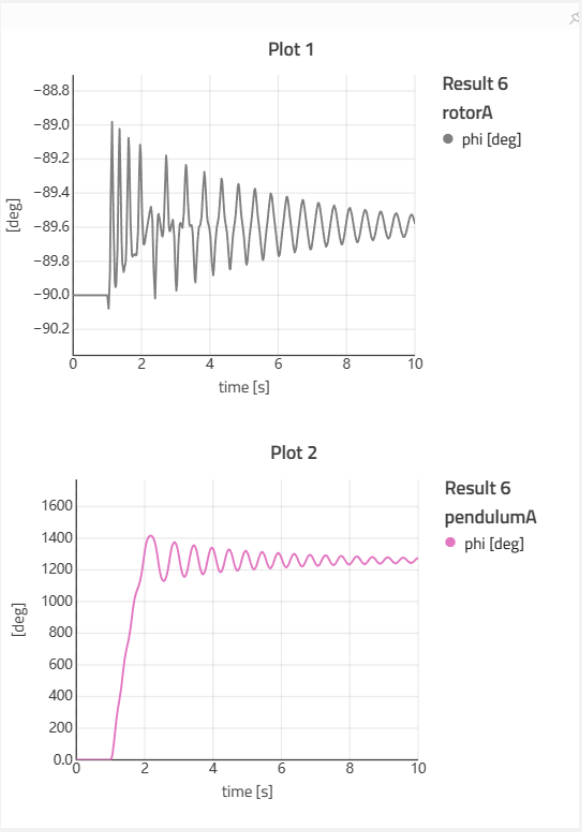

Figure 5: Task1.2 0.05

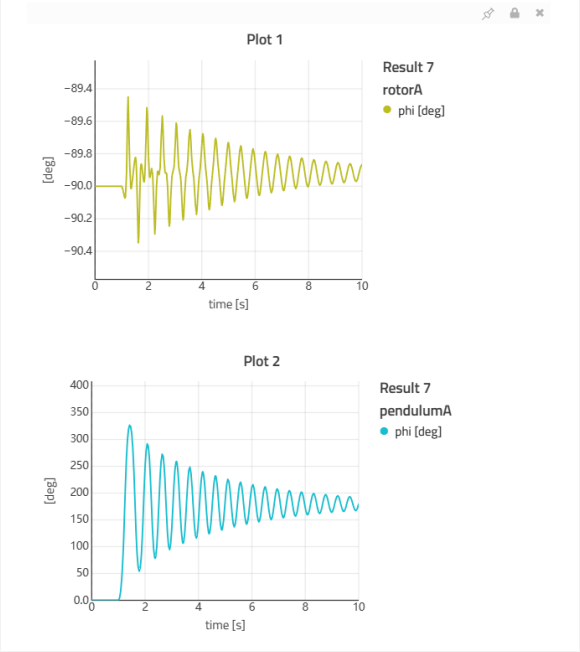

Figure 6: Task1.2 0.01

Answer to Task 2: Adding an additional pendulum arm

Add the required model for the experiment and complete the modeling of the double inverted pendulum. The Modelica diagram of task2 is as figure 7.

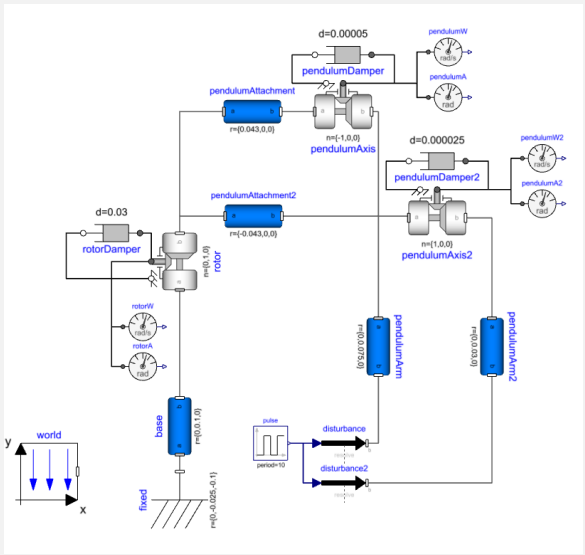

Figure 7: Task2

The pendulum of task2 is as figure 8

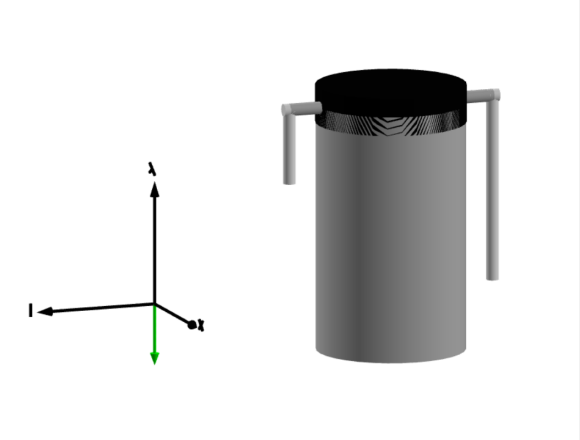

Figure 8: Task2 Pendulum

Then plot the change of two pendulum and the rotor angles.The results are shown as follow.

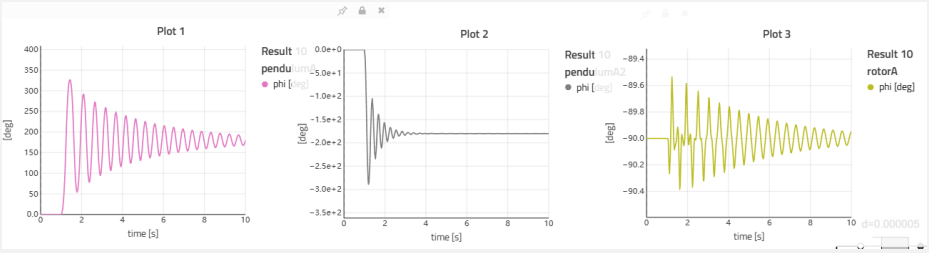

Figure 9: Task2 0.01

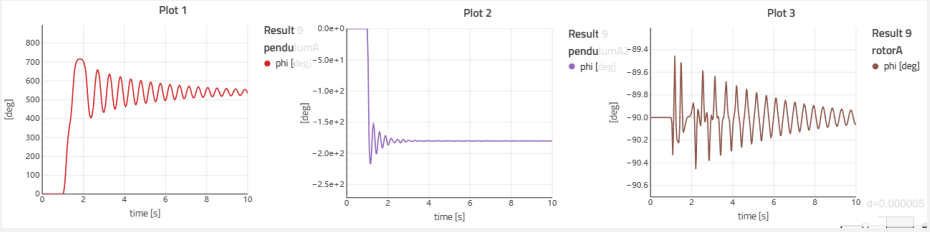

Figure 10: Task2 0.025

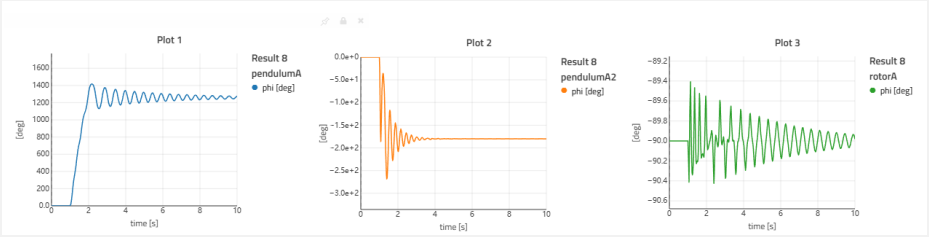

Figure 11: Task2 0.05

Answer to Task 3: Adding a damping controller

Add the torque, instantiate the model with the appropriate input and output parameters. Utilize Jupyter Notebook to compute the L matrix and use it as a parameter to calculate 'u'. L is shown as follow.

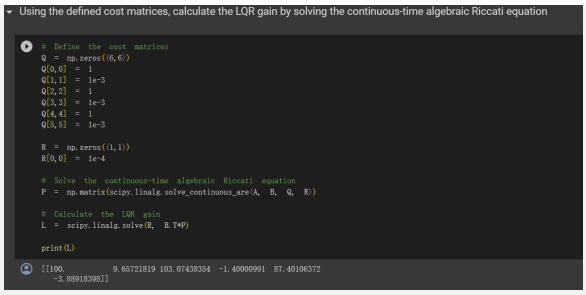

Figure 12: Task3 L

The Modelica diagram of task3 is as figure 13.

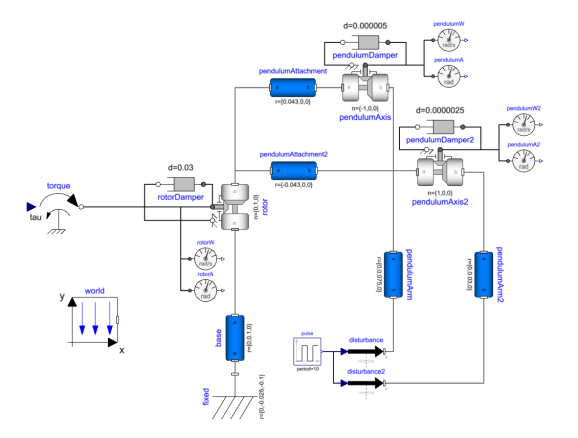

Figure 13: Task3

simulate the model with the same disturbances as in part 1 and 2. The results are shown as follow.

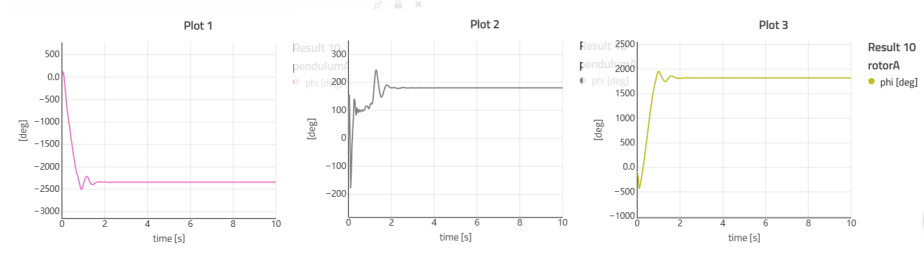

Figure 14: Task3 0.01

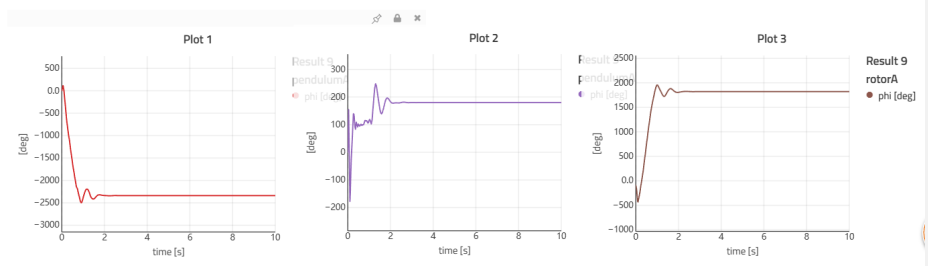

Figure 15: Task3 0.025

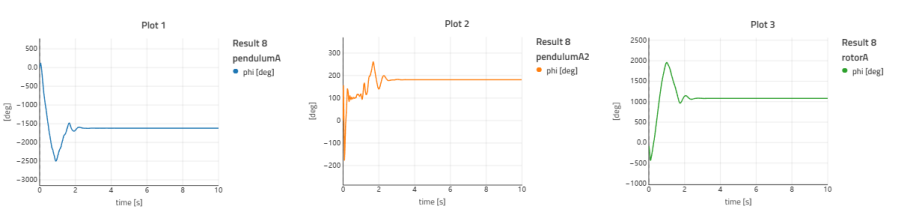

Figure 16: Task3 0.05

Next, by modifying the values in the Q and R matrices, I obtained a new L matrix. I used this matrix as a parameter for simulation, and the results are depicted in the figure.

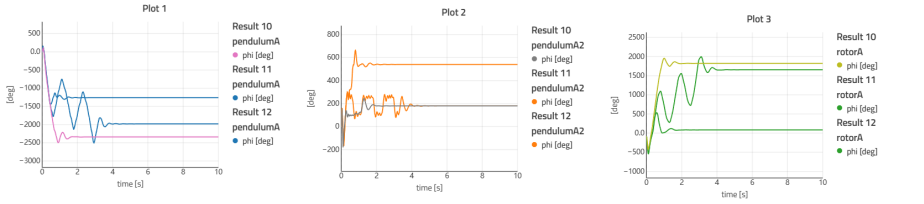

Figure 17: Task3 L123

As seen in the figure, changes in the L matrix lead to variations in the system's convergence speed and oscillation amplitude.

Answer to Task 4: Adding a stabilizing controller

By altering the A and B matrices, and setting Q = I and R = 1, a new L matrix was obtained. A, B and L is shown as follow.

```python
import numpy as np
import scipy.linalg

# Initialize constants
J = 125e-06
l1 = 0.075
l2 = 0.030
m1 = l1*np.pi*0.0025*0.0025*3700
m2 = l2*np.pi*0.0025*0.0025*3700
r = 0.043
g = 9.82

# Parameter declaration
a1 = J + m1*r*r
b1 = 1/3 * m1*l1*l1
c1 = 1/2 * m1*r*l1
d1 = 1/2 * m1*g*l1

a2 = J + m2*r*r
b2 = 1/3 * m2*l2*l2
c2 = 1/2 * m2*r*l2
d2 = 1/2 * m2*g*l2

# State Space declaration
A = np.zeros((6,6))
# A[0,1] = 1
# A[1,2] = -(d1*c1)/(a1*b1 - c1**2)
# A[1,4] = -(d2*c2)/(a2*b2 - c2**2)
# A[2,3] = 1
# # A[3,2] = -(a1*d1)/(a1*b1 - c1**2)
# A[3,2] = (a1*d1)/(a1*b1 - c1**2)
# A[4,5] = 1
# A[5,4] = -(a2*d2)/(a2*b2 - c2**2)
# # A[5,4] = (a2*d2)/(a2*b2 - c2**2)

A[0,1] = 1
A[1,2] = -(d1*c1)/(a1*b1 - c1**2)
A[1,4] = -(d2*c2)/(a2*b2 - c2**2)
A[2,3] = 1
A[3,2] = (a1*d1)/(a1*b1 - c1**2)
A[4,5] = 1
A[5,4] = (a2*d2)/(a2*b2 - c2**2)

B = np.zeros((6,1))
# B[1,0] = b1/(a1*b1 - c1**2) + b2/(a2*b2 - c2**2)
# B[3,0] = c1/(a1*b1 - c1**2)
# # B[5,0] = c2/(a2*b2 - c2**2)
# B[5,0] = -c2/(a2*b2 - c2**2)

B[1,0] = b1/(a1*b1 - c1**2) + b2/(a2*b2 - c2**2)
B[3,0] = -c1/(a1*b1 - c1**2)
B[5,0] = -c2/(a2*b2 - c2**2)
```

Figure 18: Task4 AB

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

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

# Solve the continuous-time algebraic Riccati equation
P = np.matrix(scipy.linalg.solve_continuous_are(A, B, Q, R))

# Calculate the LQR gain
L = scipy.linalg.solve(R, B.T*P)

print(L)
```

```
[[   1.          1.24272358  374.73428371   26.07636106 -236.45846667
  -10.57966294]]
```

Figure 19: Task4 L

The initial angle was set to 0.05 radians, and a small disturbance was introduced after one second. The results are shown in the figure.

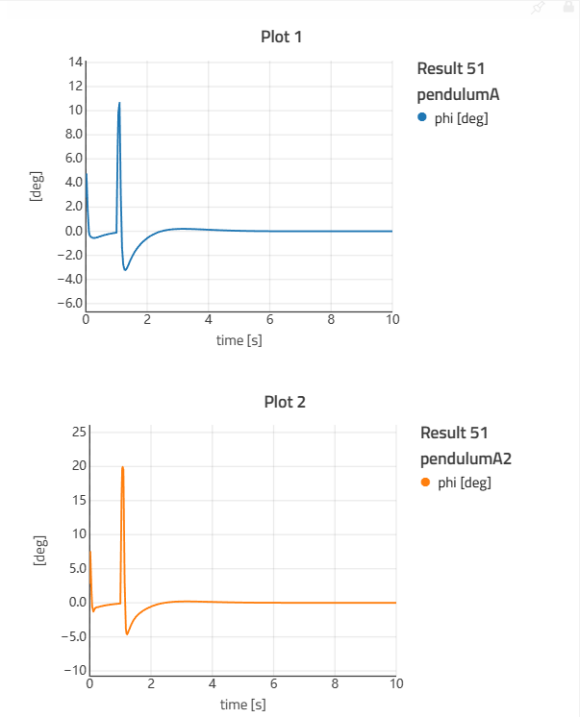

Figure 20: Task4 Plot

The Modelica diagram of task4 is as figure 21.

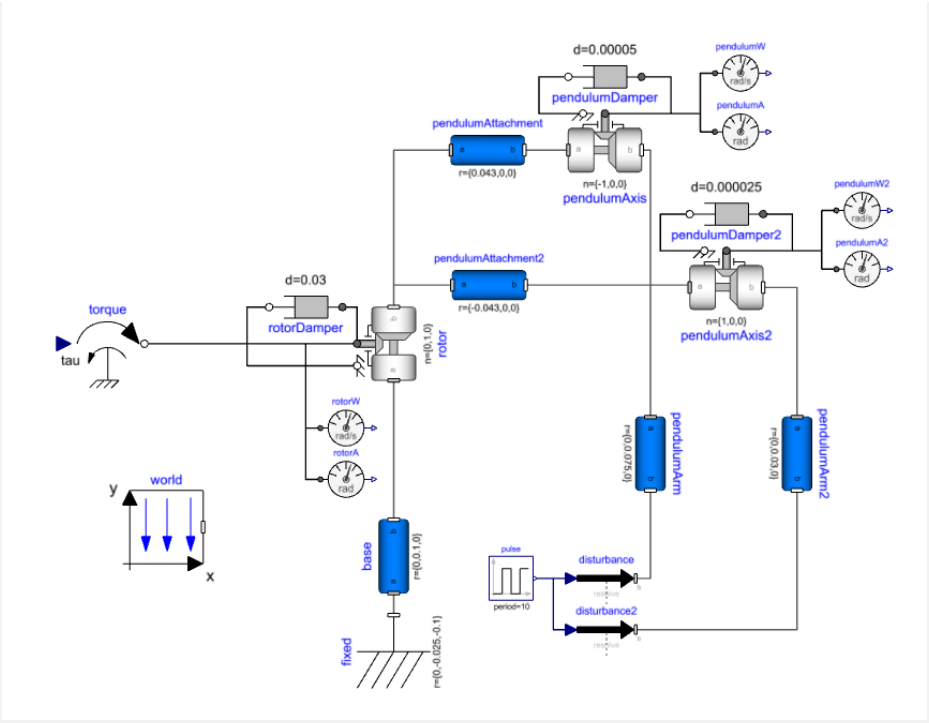

Figure 21: Task4 Diagram

The pendulum of task4 is as figure 22

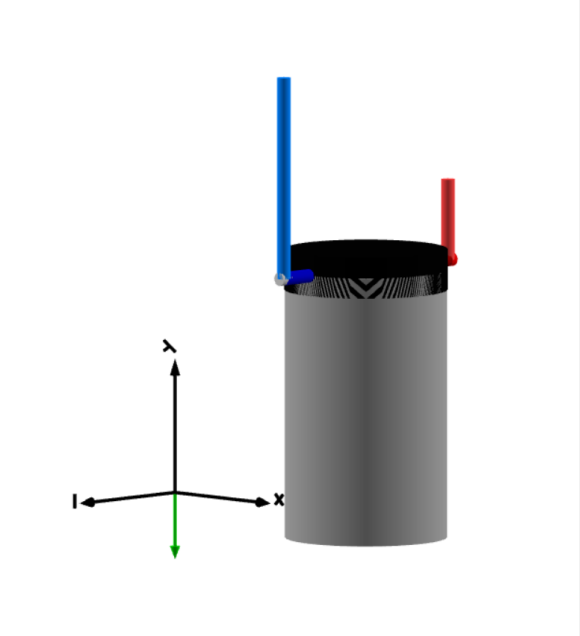

Figure 22: Task4 Pendulum