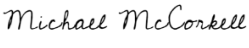
LAB 10: MODELING AND BALANCE CONTROL OF ROTARY INVERTED PENDULUM

Course Number	**MENG 3510**
Course Title	**Control Systems**
Semester/Year	**Winter / 2025**

Lab/Tutorial Report No.	**Lab 10**
Report Title	**Modeling and Balance Control of RIP**
Section No.	ONA
Group No.	2
Submission Date	03/30/2025
Due Date	03/30/2025

Student Name	Signature*	Total Mark
Joshua Mongal		/50
Mikaeel Khanzada		/50
Michael McCorkell		/50
		/50

* By signing above, you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a ZERO on the work or possibly more severe penalties.

https://academic-regulations.humber.ca/2021-2022/17.0-ACADEMIC-MISCONDUCT

LAB 10 Grading Sheet

Student Name: Joshua Mongal	**Student Name:** Mikaeel Khanzada
Student Name: Michael McCorkell	**Student Name:**

Part 1: Pendulum State-Space Model	**/10**
Part 2: State-feedback Balance Control Design	**/20**
Part 3: Implementing the Balance Controller	**/15**
General Formatting: Clarity, Writing style, Grammar, Spelling, Layout of the report	**/5**
Total Mark	**/50**

LAB 10: MODELING AND BALANCE CONTROL OF ROTARY INVERTED PENDULUM

OBJECTIVES

- To obtain the linear state-space representation of the rotary inverted pendulum
- To design a state-feedback control system that balances the pendulum in its upright vertical position
- To simulate the closed-loop system to ensure the specifications are met
- To implement the controller on the QUBE-Servo3 Rotary Pendulum system and eveluate its performance

DISCUSSIONS OF FUNDAMENTALS

ROTARY INVERTED PENDULUM MODEL

Consider the **Rotary Inverted Pendulum (RIP)** system model shown in Figure 1.

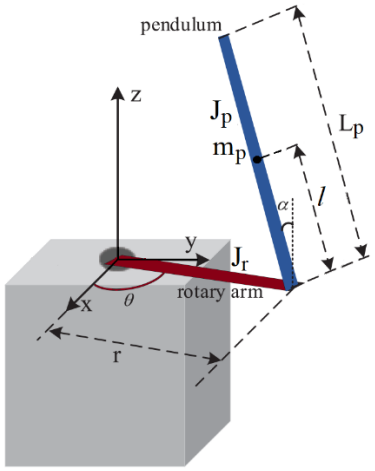


Figure 1: Rotary Inverted Pendulum System

The rotary arm pivot is attached to the **QUBE-Servo 2** system and is actuated. The arm has a length of r, a moment of inertia of J_r, and its angle θ increases positively when it rotates counter-clockwise (CCW). The servo (and thus the arm) should turn in the CCW direction when the control voltage is positive, $V_m > 0$.

The pendulum link is connected to the end of the rotary arm. It has a total length of L_p and mass of m_p and its center of mass is at $l = L_p/2$. The moment of inertia about its center of mass is J_p. The rotary pendulum angle α is zero when it is perfectly upright in the vertical position and increases positively when rotated CCW.

The **linearized equation of motion** of the system about the operating point ($\alpha \approx 0$), for zero initial conditions can be developed as follows,

$$J_r\ddot{\theta}(t) - m_p lr\ddot{\alpha}(t) = \tau(t) - b_r\dot{\theta}(t) \quad (1)$$

$$J_p\ddot{\alpha}(t) - m_p lr\ddot{\theta}(t) - m_p gl\alpha(t) = -b_p\dot{\alpha}(t) \quad (2)$$

where b_r and b_p are the viscous damping acting on the rotary arm and the pendulum link, respectively. The applied torque at the base of the rotary arm generated by the servo motor is,

$$\tau(t) = \frac{k_m}{R_m}\left(V_m(t) - k_m\dot{\theta}(t)\right) \quad (3)$$

PART 1: Pendulum State-Space Model

The linear state-space equations are,

$$\dot{\mathbf{x}}(\boldsymbol{t}) = \boldsymbol{A}\mathbf{x}(\boldsymbol{t}) + \boldsymbol{B}\mathbf{u}(\boldsymbol{t})$$

$$\mathbf{y}(\boldsymbol{t}) = \boldsymbol{C}\mathbf{x}(\boldsymbol{t}) + \boldsymbol{D}\mathbf{u}(\boldsymbol{t})$$

where x is the vector of state variables $(n \times 1)$, u is the control input vector $(r \times 1)$, y is the output vector $(m \times 1)$, A is the system matrix $(n \times n)$, B is the input matrix $(n \times r)$, C is the output matrix $(m \times n)$, and D is the feed-forward matrix $(m \times r)$.

The **system state** x defines the state variables needed to **model the system** and **output state** y defines the state variables that are **measured directly**.

For the rotary pendulum system, the state variables, output variables, and state-space matrices are defined as,

State Variables: $x_1(t) = \theta(t), \quad x_2(t) = \alpha(t), \quad x_3(t) = \dot{\theta}(t), \quad x_4(t) = \dot{\alpha}(t)$

Output Variables: $y_1(t) = \theta(t), \quad y_2(t) = \alpha(t)$

State Vector & Output Vector: $\mathrm{x}(t) = [\theta(t) \quad \alpha(t) \quad \dot{\theta}(t) \quad \dot{\alpha}(t)]^T, \quad \mathrm{y}(t) = [\theta(t) \quad \alpha(t)]^T$

State & Output Equations:

$$\dot{x}_1(t) = \dot{\theta}(t) \quad \rightarrow \quad \dot{x}_1(t) = x_3(t)$$

$$\dot{x}_2(t) = \dot{\alpha}(t) \quad \rightarrow \quad \dot{x}_2(t) = x_4(t)$$

$$\dot{x}_3(t) = \ddot{\theta}(t) \quad \rightarrow \quad \dot{x}_3(t) = \frac{1}{J_t}\left(m_p^2 l^2 r g \alpha(t) - b_r J_p \dot{\theta}(t) - m_p l r b_p \dot{\alpha}(t) + J_p \tau(t)\right)$$

$$\dot{x}_4(t) = \ddot{\alpha}(t) \quad \rightarrow \quad \dot{x}_4(t) = \frac{1}{J_t}\left(m_p g l J_r \alpha(t) - m_p l r b_r \dot{\theta}(t) - J_r b_p \dot{\alpha}(t) + m_p l r \tau(t)\right)$$

where, $J_t = J_p J_r - m_p^2 l^2 r^2$

State Space Matrices:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & m_p^2 l^2 r g / J_t & -b_r J_p / J_t & -m_p l r b_p / J_t \\ 0 & m_p g l J_r / J_t & -m_p l r b_r / J_t & -J_r b_p / J_t \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ J_p / J_t \\ m_p l r / J_t \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The state-space modeling of rotary inverted pendulum requires pre-determined parameters so **MATLAB** codes must be opened along with the **Simulink** model. The parameters are also used for balance control of the inverted pendulum in **Part 2**.

1. **Open** the provided **MATLAB** file named **`RIP_Parameters.m`** and **run** the script. This loads the required system parameters.

2. **Run** the following code in **MATLAB** to calculate the state-space matrices, create the state-space model and find the open-loop system poles using **`eig`** command.

```
Jt = Jr*Jp - mp^2*l^2*r^2;      % Calculate Jt

% State-Space Model Matrices
A = [0  0  1  0;
     0  0  0  1;
     0  mp^2*l^2*r*g/Jt   -br*Jp/Jt    -mp*l*r*bp/Jt;
     0  mp*g*l*Jr/Jt     -mp*l*r*br/Jt   -Jr*bp/Jt];

B = [0; 0; Jp/Jt; mp*l*r/Jt];
C = eye(2,4);
D = zeros(2,1);

% Convert torque input to voltage input
A(3,3) = A(3,3) - km*km/Rm*B(3);
A(4,3) = A(4,3) - km*km/Rm*B(4);
B = km * B / Rm;

sys_OL = ss(A,B,C,D)         % State-space model of open-loop system
Pole_OL = eig(A)             % Open-loop poles
```

Provide the code execution from **MATLAB Command** window.

```
sys_OL =

  A =
           x1       x2       x3       x4
   x1       0        0        1        0
   x2       0        0        0        1
   x3       0      152   -12.53  -0.5005
   x4       0    264.3   -12.39  -0.8702

  B =
         u1
   x1     0
   x2     0
   x3  56.98
   x4  56.32

  C =
       x1  x2  x3  x4
   y1   1   0   0   0
   y2   0   1   0   0

  D =
       u1
   y1   0
   y2   0

Continuous-time state-space model.

Pole_OL = 4×1
         0
   -22.1965
    13.5479
    -4.7538
```

3. What do you notice about the location of the open-loop poles? How does that affect the **stability** of the open-loop system? What is the **Type** of the open-loop system?

There is a pole at 0 while the others are in the negative region. This shows the open loop system is not stable. It is a Type 1 system.

PART 2: State-feedback Balance Control Design

The system can be **stabilized** and **regulated** to a desired performance by implementing a ***State-feedback control*** strategy. State-feedback control of a system is shown In Figure 2.

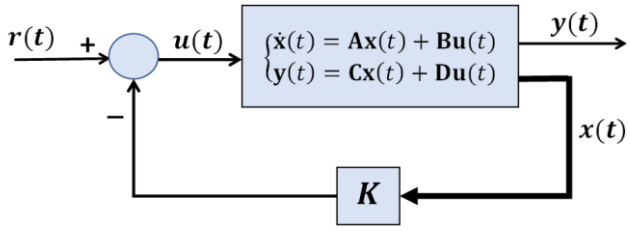


Figure 2: State-feedback Control System

The ***state-feedback control law*** is defined as,

$$\mathbf{u}(\boldsymbol{t}) = -\boldsymbol{K}\mathbf{x}(\boldsymbol{t}) + \mathbf{r}(\boldsymbol{t})$$

where K is the feedback gain and $r(t)$ is the reference state. The state-space model of the closed-loop system is,

$$\dot{\mathbf{x}}(\boldsymbol{t}) = (\boldsymbol{A} - \boldsymbol{BK})\mathbf{x}(\boldsymbol{t}) + \boldsymbol{B}\mathbf{r}(\boldsymbol{t})$$

$$\mathbf{y}(\boldsymbol{t}) = (\boldsymbol{C} - \boldsymbol{DK})\mathbf{x}(\boldsymbol{t}) + \boldsymbol{D}\mathbf{r}(\boldsymbol{t})$$

The refence state includes the desired rotary arm position, θ_r, and is defined as,

$$\mathbf{r}(t) = [\theta_r \quad 0 \quad 0 \quad 0]$$

CONTROLLABILITY

The state-feedback control system can be designed if the system is ***Controllable***. To check if the system is controllable, we can compute the rank of the ***Controllability*** matrix,

$$Q_c = [B \quad AB \quad A^2B \quad \cdots \quad A^{n-1}B]$$

where n is the dimension of the state vector $\mathrm{x(t)}$. The system is controllable if and only if the rank of its controllability matrix equals the number of states in the system,

$$rank(Q_c) = n$$

If the system is controllable, we can design a state-feedback control to place the poles at *desired locations*, e.g. in the left half of the s-plane to *stabilize* the system and to achieve a *desired performance*.

DESIRED PERFORMANCE CRITERIA

The control design and time-response specifications are:

1. Percent Overshoot: $O.S. \leq 5\%$
2. Settling time: $t_s \leq 1.5\ sec$
3. Maximum pendulum angle deflection: $|\alpha| < 15\ deg$
4. Maximum control effort/voltage: $|V_m| < 10\ V$

The necessary closed-loop poles are found from the desired percent overshoot and the settling time.

The pendulum deflection and control effort requirements are to be satisfied when the rotary arm is tracking the desired reference angle.

5. Follow the steps and run the provided **MATLAB** code for each step to design the state-feedback controller:

Step 1: Check controllability of the system

Calculate the controllability matrix Q_c using **`ctrb`** command and check **rank** of the matrix Q_c.

```
Qc = ctrb(A,B)                  % Controllability matrix
rank(Qc)                        % Find rank of the controllability matrix
```

Provide the code execution from **MATLAB Command** window.

```
Qc = 4×4
10^5 ×
         0    0.0006   -0.0074    0.1824
         0    0.0006   -0.0075    0.2474
    0.0006   -0.0074    0.1824   -3.5573
    0.0006   -0.0075    0.2474   -4.4699

ans = 4
```

Is the open-loop system controllable? **YES** X **NO**

Step 2: Determine the desired closed-loop pole locations

The rotary inverted pendulum system has **four** closed-loop poles. As depicted in Figure 3, poles p_1 and p_2 are the complex conjugate **dominant** poles and the system's behavior can be approximated by a ***second-order*** system.

$$p_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$

The dominant poles are chosen to satisfy the time-response specifications ($O.S. \leq 5\%,\ t_s \leq 1.5\ sec$).

$$O.S. = e^{-\pi\zeta/\sqrt{1-\zeta^2}} \quad , \quad t_s = \frac{4}{\zeta\omega_n}$$

The remaining closed-loop poles p_3 and p_4, are placed along the real-axis to the left, far from the dominant poles e.g. at $p_3 = -40$ and $p_4 = -50$, as shown in Figure 3.

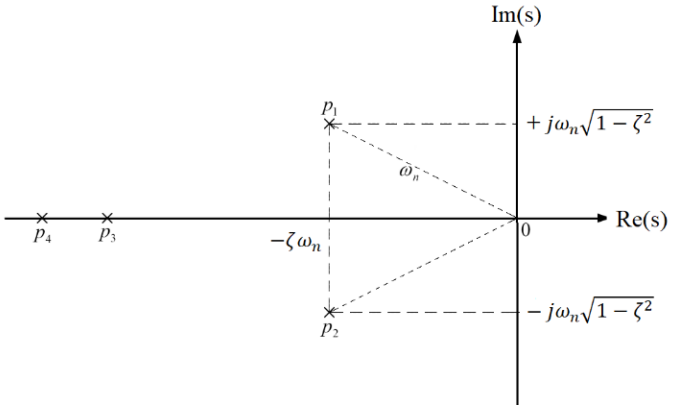


Figure 3: Desired closed-loop pole locations

Run the following code in **MATLAB** to determine the desired closed-loop pole locations:

```
OS = 0.05;                                          % Desired overshoot
ts = 1.5;                                           % Desired settling time
zeta = sqrt((log(OS))^2/(pi^2+(log(OS))^2))         % Desired damping ratio
wn = 4/(zeta*ts)                                    % Desired natural frequency

% Calculate the desired closed-loop poles
p1 = -zeta*wn + wn*sqrt(1-zeta^2)*i;
p2 = -zeta*wn - wn*sqrt(1-zeta^2)*i;
p3 = -40;
p4 = -50;

Pole_cl = [p1 p2 p3 p4]                             % Desired closed-loop poles
```

Provide the code execution from **MATLAB Command** window.

```
OS = 0.05;                                  % Desired overshoot
ts = 1.5;                                   % Desired settling time
zeta = sqrt((log(OS))^2/(pi^2+(log(OS))^2))         % Desired damping ratio

zeta = 0.6901

wn = 4/(zeta*ts)                            % Desired natural frequency

wn = 3.8641


% Calculate the desired closed-loop poles
p1 = -zeta*wn + wn*sqrt(1-zeta^2)*i;
p2 = -zeta*wn - wn*sqrt(1-zeta^2)*i;
p3 = -40;
p4 = -50;

Pole_cl = [p1 p2 p3 p4]                     % Desired closed-loop poles

  Pole_cl = 1×4 complex
      -2.6667 + 2.7965i  -2.6667 - 2.7965i -40.0000 + 0.0000i -50.0000 + 0.0000i
```

Step 3: Determine the state-feedback gain and verify the closed-loop poles

Determine the state-feedback gain K using **`place`** command and calculate the closed-loop system matrix $A_{cl} = A - BK$ and verify the closed-loop poles.

```
K = place(A,B,Pole_cl)          % State-feedback gain
A_cl = A-B*K;                   % Closed-loop system matrix

A_cl_poles = eig(A_cl)          % Closed-loop poles
```

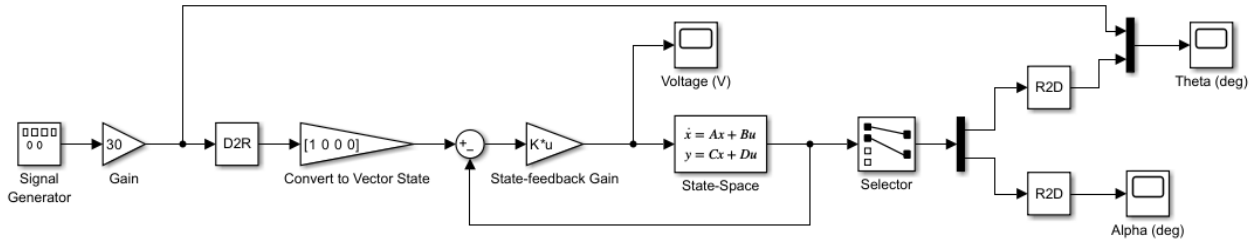
Provide the code execution from **MATLAB Command** window.

```
K = place(A,B,Pole_cl)          % State-feedback gain

  K = 1×4
       -4.5942   54.3474   -2.0828    3.5620

A_cl = A-B*K;                   % Closed-loop system matrix

A_cl_poles = eig(A_cl)          % Closed-loop poles

  A_cl_poles = 4×1 complex
     -50.0000 + 0.0000i
     -40.0000 + 0.0000i
      -2.6667 + 2.7965i
      -2.6667 - 2.7965i
```

Are the closed-loop poles located at the desired locations? **YES** X **NO**

Step 4: Simulate the designed state-feedback gain and verify the balanced control.

The following **Simulink** diagram is used to simulate the closed-loop response of the Rotary Pendulum using the designed state-feedback control.



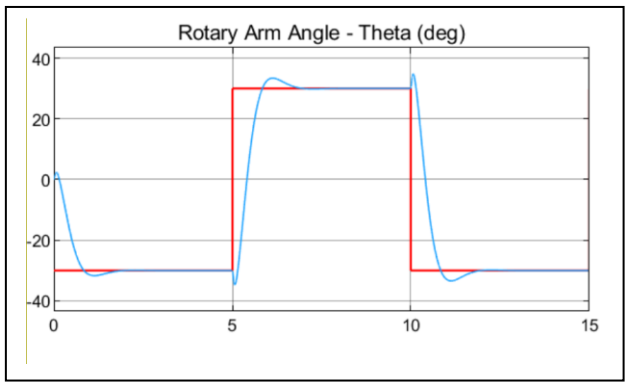
Open the provided **`Lab10_Pen_Linear_Model.slx`** Simulink file. ***NOTE: Select the file that is compatible with your MATLAB version.***

The **Signal Generator** block generates a **0.1Hz square wave** with an **amplitude of 1.**
The **Gain** block is used to set the desired rotary arm position in **degrees**.
The **State-feedback Gain** block is set the controller gain K to the designed value from the **MATLAB** workspace.
The **State-Space** block reads the $A, B, C, \text{and } D$ matrices from the **MATLAB** workspace to create the model.

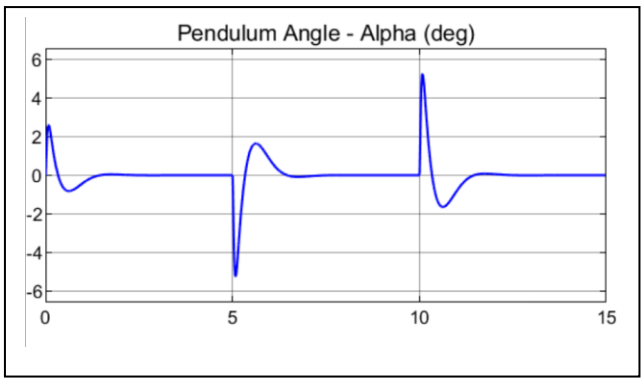
***NOTE**: Before conducting this experiment, you need to make sure that the lab files are configured and the system model $A, B, C, and\ D$ matrices and the controller gain K is loaded in **MATLAB** workspace.*

6. **Run** the simulation for **15 seconds**. Double-click and open the **Scope** blocks. Provide the plots below:

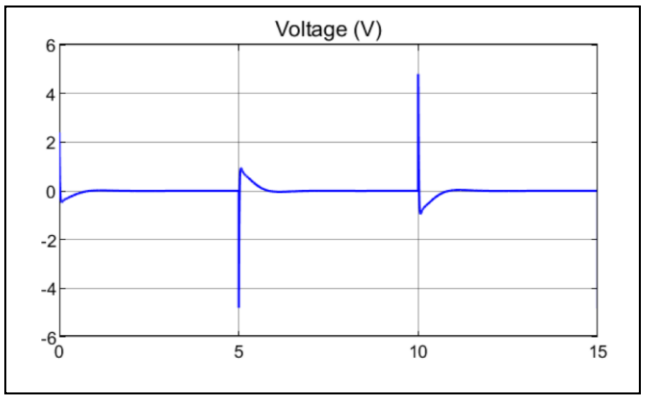
Reference Input & Rotary Arm Angle (deg)



Inverted Pendulum Angle (deg)



Voltage (V)



7. Measure the percentage of overshoot, settling time, the maximum pendulum deflection, and maximum voltage used, and complete **Table 1.**

Table 1: Specifications from linear model

Percent Overshoot (%O.S.)	Settling Time (sec)	Maximum Pendulum Deflection (deg)	Maximum Voltage (V)
5.6 %	1.728 s	5.262 degrees	4.811 V

Are the desired specifications satisfied? **YES** **NO** ✓

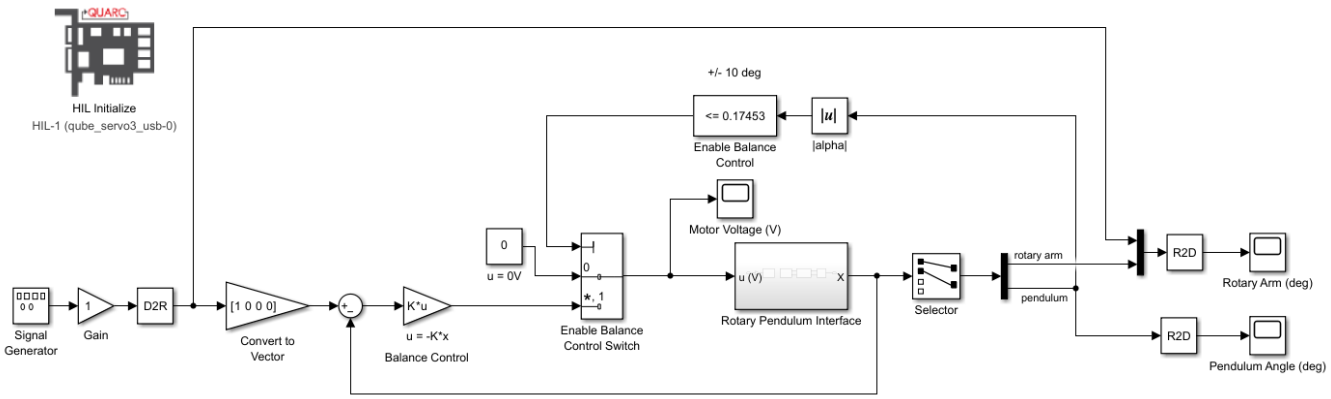
8. **Close** the Simulink diagram when you are done.

PART 3: Implementing the Balance Controller

In this section, the state-feedback control that was designed and simulated in the previous sections is run on the actual **QUBE-Servo3 Rotary Pendulum** device.

9. Set up the QUBE-Servo3 with the pendulum system by removing the inertia disk and adding the pendulum system. Plug in the pendulum encoder wire into **Encoder** plug in. **NOTE:** ***The encoder wire may need to be rotated and adjusted to ensure the pendulum is centered with the 0° mark on the QUBE base.***

10. Open the provided **`Lab10_Pen_Balance_Control.slx`** Simulink file. ***NOTE: Select the file that is compatible with your MATLAB version.***



The **Enable Balance Control Switch** block engages the balance control whenever the inverted pendulum angle is in the required range (less than 10°).

***NOTE**: Before conducting this experiment, you need to make sure that the lab files are configured and the controller gain K is loaded in **MATLAB** workspace.*

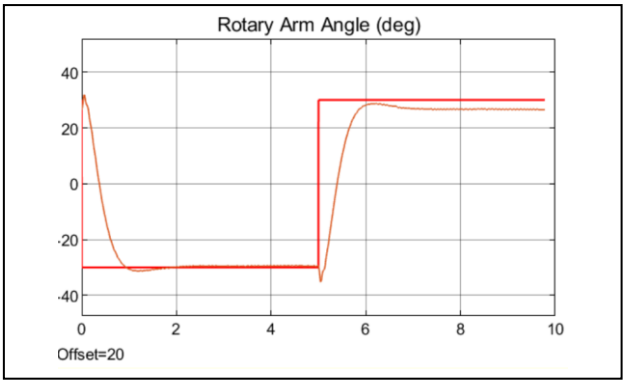
11. **Ensure the pendulum is in the hanging down, zero-degree position and motionless.**

12. **Run** the Simulink diagram. Once the system is running and the QUBE strip light turns **green**, manually bring up the pendulum to its upright vertical position. You should feel that the voltage kick-in when it is within the range where the Balance Control engages.

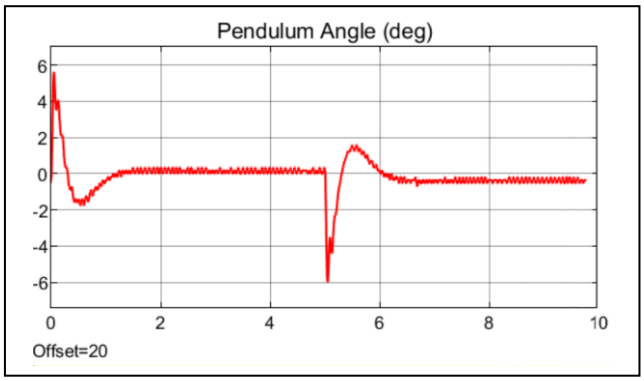
13. Once the pendulum is balanced, set the **Gain** block to **30** to make the rotary arm angle move between **±30°.**

14. Double-click and open the **Scope** blocks. Provide **10 seconds** of the plots below:

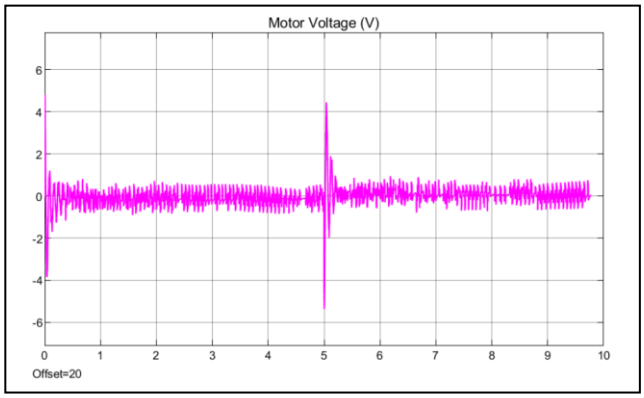
Reference Input & Rotary Arm Angle (deg)



Inverted Pendulum Angle (deg)



Motor Voltage (V)



15. Measure the percentage of overshoot, settling time, the maximum pendulum deflection, and maximum voltage used, and complete **Table 2.**

Table 2: Specifications from actual system

Percent Overshoot (%O.S.)	Settling Time (sec)	Maximum Pendulum Deflection (deg)	Maximum Voltage (V)
3.3 %	1.88 s	5.625 degrees	4.806 V

Are the desired specifications satisfied? If not, give one reason why there is a discrepancy.

The desired specifications are not completely satisfied as the settling time is higher than what is desired. One possible reason for this discrepancy is that there may be a delay in the computer receiving information like a computational lag or due to the sensor having delays leading to slower responses.

16. Click on the **STOP** button to stop the controller. ***Be careful, as the pendulum will fall down when the controller is stopped.***

17. **Power off** the QUBE-Servo3 system if no more experiments will be conducted.