EE324: Experiment 3
Inverted Pendulum

Akhilesh Chauhan 21d070010
Uvesh Mohammad 21d070084
Parth Arora 21d070047
Batch-16

November 8, 2023

Contents

1 Overview of the experiment

1.1 Aim

To design and implement control action for maintaining a pendulum in the upright position (even when subjected to external disturbances) through LQR technique in an Arduino Mega.

1.2 Objective

The objective of the experiment was to:

- To restrict the pendulum arm vibration (α) within $\pm$ 3 degrees
- To restrict the base angle oscillation (θ) within $\pm$ 30 degrees

1.3 Materials required

- Inverted pendulum setup
- Arduino mega
- A-B cable
- Decoder shield
- Power supply
- Screw driver
- Jumpers
- Wires and Wire stripper

2 Design

2.1 Control Algorithm

2.1.1 State Space Modelling

The state space vector contains four variables defined as

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$$
$$\text{where, } \; x_1 = \theta, \; x_2 = \alpha, \; x_3 = \frac{\partial \theta}{\partial t}, \; x_4 = \frac{\partial \alpha}{\partial t}$$

θ is the angle made by the horizontal arm from its initial position while α is the angle made by the main pendulum arm from the top. Thus the pendulum is to be kept about $\alpha = 0$. The input $u(t)$ is a scalar and the voltage given to the motor.

The linear state space equation is given by

$$\frac{d}{dt}\mathbf{x}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$$
$$\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$$

The matrices are given as

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{rM_p^2 l_p^2 g}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & \frac{K_t K_m (J_p + M_p l_p^2)}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \\ 0 & \frac{M_p l_p g (J_{eq} + M_p r^2)}{J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2} & \frac{-M_p l_p K_t r K_m}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{K_t (J_p + M_p l_p^2)}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \\ \frac{M_p l_p K_t r}{(J_p J_{eq} + M_p l_p^2 J_{eq} + J_p M_p r^2) R_m} \end{bmatrix}$$
$$C = \mathbb{I}_4$$
$$D = \mathbf{O}_{4\times 1}$$

The variables used in the matrices are defined next.

- M_p = Mass of pendulum assembly = $0.027\ kg$
- l_p = Length of pendulum COM from pivot = $0.153\ m$
- L_p = Total length of pendulum = $0.191\ m$
- r = Length of arm pivot to pendulum pivot = $0.0826\ m$
- J_m = Motor shaft moment of inertia = $3 \times 10^{-5}\ kg\, m^2$
- M_{arm} = Mass of arm = $0.028\ kg$
- g = gravitational acceleration constant = $9.81\ m/s^2$
- J_{eq} = Equivalent moment of inertial about pivot axis = $1.23 \times 10^{-4}\ kg\, m^2$
- J_p = Pendulum moment of inertia about pivot axis = $1.1 \times 10^{-4}\ kg\, m^2$
- R_m = Motor armature resistance = $3.3\ \Omega$
- K_t = Motor torque constant = $0.2797\ N\, m$
- K_m = Motor back-emf constant = $0.2797\ V/(rad/s)$

2.1.2 LQR Algorithm

The LQR algorithm states that given the state space model stated previous find the input u such that the cost function J is minimized, defined as

$$J = \int_0^\infty \mathbf{x}(t)^T Q \mathbf{x}(t) + \mathbf{u}(\mathbf{t})^T R \mathbf{u}(t)\ dt$$

In our case, Q is 4×4 positive semi-definite diagonal weighing matrix while R is a positive scalar. In other words, we want to find a control gain 1×4 matrix K in the feedback law $u = \mathbf{K}x$ such that J is minimized.

The MATLAB function *lqr* is used which takes in input A, B, Q and R and returns K. The values of K that were used finally were

$$\mathrm{K} = \begin{bmatrix} -3.6 & 94.9836 & -0.30389 & 0.44321 \end{bmatrix}$$

2.2 Arduino Programming

Main code is given below:

```
void loop()
{
  //create a 16 bit variable to hold the encoders position
  uint16_t encoderPosition;
  //let's also create a variable where we can count how many times we've tried to obtain th
  uint8_t attempts;


  //if you want to set the zero position before beggining uncomment the following function
  //setZeroSPI(ENC_0);
  //setZeroSPI(ENC_1);

  //once we enter this loop we will run forever
  while(1)
  {
    //set attemps counter at 0 so we can try again if we get bad position
    attempts = 0;

    //this function gets the encoder position and returns it as a uint16_t
    //send the function either res12 or res14 for your encoders resolution
    encoderPosition = getPositionSPI(ENC_0, RES14);

    //if the position returned was 0xFFFF we know that there was an error calculating the c
    //make 3 attempts for position. we will pre-increment attempts because we'll use the nu
    while (encoderPosition == 0xFFFF && ++attempts < 3)
    {
      encoderPosition = getPositionSPI(ENC_0, RES14); //try again
    }

    if (encoderPosition == 0xFFFF) //position is bad, let the user know how many times we t
```

```
{
  Serial.print("Encoder 0 error. Attempts: ");
  Serial.print(attempts, DEC); //print out the number in decimal format. attempts - 1 i
  Serial.write(NEWLINE);
}
else //position was good, print to serial stream
{

  Serial.print("Encoder 0: ");
  Serial.print(encoderPosition, DEC); //print the position in decimal format
  Serial.write(NEWLINE);
}

encoder_position_0 = encoderPosition;
//////////again for second encoder//////////////////////////////

//set attemps counter at 0 so we can try again if we get bad position
attempts = 0;

//this function gets the encoder position and returns it as a uint16_t
//send the function either res12 or res14 for your encoders resolution
encoderPosition = getPositionSPI(ENC_1, RES14);

//if the position returned was 0xFFFF we know that there was an error calculating the c
//make 3 attempts for position. we will pre-increment attempts because we'll use the nu
while (encoderPosition == 0xFFFF && ++attempts < 3)
{
  encoderPosition = getPositionSPI(ENC_1, RES14); //try again
}

if (encoderPosition == 0xFFFF) //position is bad, let the user know how many times we t
{
  Serial.print("Encoder 1 error. Attempts: ");
  Serial.print(attempts, DEC); //print out the number in decimal format. attempts - 1 i
  Serial.write(NEWLINE);
}
else //position was good, print to serial stream
{

  Serial.print("Encoder 1: ");
  Serial.print(encoderPosition, DEC); //print the position in decimal format
  Serial.write(NEWLINE);
}

encoder_position_1 = encoderPosition;
//For the purpose of this demo we don't need the position returned that quickly so let
//delay() is in milliseconds
```

```
    theta = encoder_position_1*2*3.14/16384;
    alpha = encoder_position_0*2*3.14/16384 - 3.14;

    d_theta = (theta - prev_theta)/0.01;
    d_alpha = (alpha - prev_alpha)/0.01;

    prev_theta = theta;
    prev_alpha = alpha;

    torque = (k1*theta + k2*alpha + k3*d_theta + k4*d_alpha)/10;

    Serial.print("alpha: ");
  Serial.println(alpha);
    Serial.print("theta: ");
    Serial.println(theta);
    if(torque > 0)
    {
      analogWrite(6, min(torque,255));
      analogWrite(7, 0);
    }
    else
    {
      analogWrite(7, min(torque,255));
      analogWrite(6, 0);
    }

    Serial.println(torque);

    // delay(100);
  }
}
```

3 Experiment

3.1 Observations and Results

Our observation shows that:

- deviation in θ is maintained under 27 degrees.
- deviation in α is maintained under 2.5 degrees.
- The system is exceptionally stable against sharp and strong attacks. The system is within stated parameters.

- The control matrix manages to keep the system upright.The values of K are given below:

$$K = \begin{bmatrix} -3.6 & 94.9836 & -0.30389 & 0.44321 \end{bmatrix}$$

- The system is resilient and meets the constraints set up.

3.2 Challenges Faced and their solutions

We had to deal with various challenges during the experiment which are given below:

- We had to deal with vibrations and instability within the system.
- We had to achieve pendulum stability through Q Matrix Optimization.
- We had to balance control and stability in Pendulum Movement.
- We had to deal with circuit issues like troubleshooting circuit anomalies and overcoming circuit connection challenges.
- We once faced with IC replacement ans Short-Circuit Problems.
- We had to maintain theta within specified bounds,modify the Q matrix for theta constraint and enhance pendulum stability within specified boundaries.