Experiment 3

Inverted Pendulum

Shounak Das, 21D070068
Prajapati Kishan K, 21D070048
Vinay Sutar, 21D070078

October 2023

Contents

1 Aim

- To design and implement a control system using the LQR technique on an Arduino Mega that effectively maintains a pendulum in the upright position.
- Mitigating pendulum arm vibration (α) to within ± 3 degrees and controlling base angle oscillation (θ) within ± 30 degrees

2 Getting Started

To embark on our experimental journey involving the inverted pendulum control system, we initiated our efforts by delving into the foundational concepts of the Linear Quadratic Regulator (LQR) technique.

The essence of the LQR algorithm is rooted in its capacity to derive the control input, denoted as $u(t)$, that minimizes the quadratic cost function J (as indicated in Equation 2), designed for a given Linear Time-Invariant (LTI) system, as represented by Equation 1.

In this context, matrix Q, with dimensions 4 by 4, and matrix R, with dimensions 1 by 1, play pivotal roles. Our responsibility was configuring these matrices to acquire the optimal control gain K, where the control input u is determined by the equation $u = -Kx$.

LQR represents dynamic systems using state-space models and finds the best control inputs to maintain system states. It computes the best control gains by solving the corresponding matrix Riccati equation, allowing accurate control of linear time-invariant systems.

$$\frac{d}{dt}x = Ax + Bu \tag{1}$$

The cost function that needs to be minimized is:

$$J = \int_0^\infty x^T Q x + u^T R u \, dt \tag{2}$$

Where: x is the state vector.
u is the control input.
Q is the state weighting matrix.
R is the control effort weighting matrix.
A and B are matrices that describe the dynamics of the system.

3 Method and Overview

3.1 Choice of Q and R Matrices

The choice of the matrices Q and R is critical to achieving the desired control performance. The matrices Q and R are typically selected based on the specific control objectives and the trade-off between tracking performance and control effort. A common approach is to tune these matrices iteratively.

For instance, increasing the values in Q will place more importance on minimizing the state error, while increasing the values in R will prioritize minimizing control effort. The optimal values for Q and R can be obtained through trial and error, simulation, or optimization techniques.

3.2 System Dynamics

The system dynamics are governed by the differential equation: $\frac{d}{dt}x = Ax+Bu$ Here, x is the state vector, representing the system's variables such as position and velocity. Matrix A describes how the state evolves over time, and matrix B relates the control input u to the state dynamics.

3.3 Implementation

The integration of the LQR algorithm in MATLAB with the Arduino platform involves transferring the control gain K obtained from MATLAB to the Arduino code. The control input u in the Arduino code is determined by: $u = Kx$ Here, K is the optimal control gain computed using the LQR technique. The Arduino code uses this control input to drive the inverted pendulum system.

The Matlab and Arduino codes are provided at the end of the report.

4 Using the matrices Q, R and K for control

4.1 K matrix

The foundation of the LQR (Linear Quadratic Regulator) control strategy is the K matrix. It establishes the control gains, which are then utilized to compute the system's control input or action. The K matrix is important because it can direct the system toward stability and the desired outcome. You may successfully influence the system's behavior by changing the values in the K matrix. Stronger control is correlated with higher levels in K, while gentler, less forceful control is associated with lower values. A key component of adjusting the control strategy to meet certain performance and stability objectives is the K matrix.

4.2 Q matrix

The Q matrix, which indicates the relative significance of various state variables in the system's performance, is essential to LQR control. It outlines the expenses related to each state variable's departure from the intended or benchmark value. You can successfully shape the control strategy by prioritizing or de-prioritizing particular state variables by modifying the values in the Q matrix. This enables you to modify the control system's behavior in order to achieve particular performance goals. For instance, you would give the equivalent state in the Q matrix a higher value if keeping the pendulum's angle was more important than controlling its angular velocity.

4.3 R matrix

The cost of control effort or input is represented by the R matrix. It calculates the cost of giving the system control inputs. This practically means that it takes into account the energy or resources needed to maintain system control. By modifying the values in the R matrix, you can achieve a compromise between control effort and performance. A preference for a less aggressive or more energy-efficient control method is indicated by a higher value in the R matrix, whereas a lower value places more emphasis on improving system performance at the price of control effort. The significance of the R matrix is in its ability to customize the control technique to make it more effective and appropriate for practical applications.

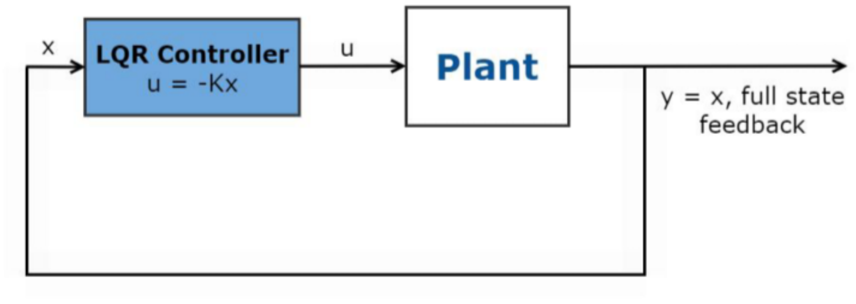

Figure 1: LQR Control

After determining the K matrix, the code can be uploaded to the Arduino Board, enabling the observation of the system's response. This real-time observation allows for adjustments in the Q and R matrices. For instance, if the pendulum rotates more slowly in the horizontal direction compared to the vertical direction, the weight of can be increased. Similarly, modifying the R matrix can decrease the penalty for control input, encouraging a more energy-efficient control strategy.

Fine-tuning the Q and R matrices is crucial as it strikes a balance between system stability, performance, and control effort. This balance is essential for achieving the desired stability and behavior of the inverted pendulum system. The process becomes iterative, allowing for continuous refinement until the system exhibits the desired response.

5 Code : Matlab and Arduino

We employed a publicly accessible MATLAB code, specifically the LQR function, to compute the control gain matrix K. This was done using the provided system parameters A, B, C, and D, which were outlined in the lab manual. In addition to these, we assigned arbitrary values for the weighting matrices Q and R.

We further assessed the system's performance by examining the step response graph, which enabled us to gauge whether the angular position θ and α adhered to the predefined specifications.

5.1 Matlab code for this procedure :

```
Mp=0.027
lp=0.153
Lp=0.191
r=0.0826
Jm=0.00003
Marm=0.028
g=9.81
Jeq=0.000123
Jp=0.00011
Beq=0
Bp=0
Rm=3.3
Kt=0.02797
Km=0.02797

A=[0 0 1 0; 0 0 0 1;
0 (r*Mp*Mp*lp*lp*g)/(Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r) -(Kt*Km*(Jp+Mp*lp*lp))
/((Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r)*Rm) 0;

0  Mp*lp*g*(Jeq+Mp*r*r)
/(Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r) -Mp*lp*Kt*r*Km
/((Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r)*Rm) 0]
B=[0; 0; (Kt*(Jp+Mp*lp*lp))/((Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r)*Rm);
(Mp*lp*Kt*r)/((Jp*Jeq+Mp*lp*lp*Jeq+Jp*Mp*r*r)*Rm)]
C=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]
D=[0;0;0;0]

Q1 = [15 0 0 0; 0 0.5 0 0; 0 0 0.5 0; 0 0 0 0.1];
R=1
[K1,S1,P1] = lqr(A,B,Q1,R);

sys1 = ss(A-B*K1,B,C,D);
step(sys1)
disp(K1)
```

5.2 The Main part of Arduino code we wrote is as follows :

```
/* SPI pins */
#define ENC_0            2
#define ENC_1            3
#define SPI_MOSI        51
#define SPI_MISO        50
#define SPI_SCLK        52
/*-------------------------------------------------------*/

/*---------Declaring and initializing variables--------------*/
long int theta;
long int thetaOld;
long int alpha;
long int alphaOld;
  //let’s also create a variable where we can count how many times we’
  tried to obtain the position in case there are errors
long int attempts;
long start_alpha=(long)getPositionSPI(ENC_0, RES14);
double dTheta=0;
double dAlpha=0;

double dT;

long time=0;
```

```
double Output=0;

double k[4]={-3.6056,   48.1143,   -1.8058,    6.2060};

/*----modifying theta , dthetha , alpha , dalpha----------------------------*/

 AlphaRead = getPositionSPI(ENC_0, RES14);
    alpha= (long)AlphaRead-start_alpha;

    dT=msDelay*(1e-3);
    dAlpha=(double)((alpha-alphaOld)/(dT));

ThetaRead = getPositionSPI(ENC_1, RES14);
    theta = ThetaRead-StartingTheta;

    dTheta=((double)(theta-thetaOld)/dT);

/*------------------------------------------------------------------------*/


     K = {3.6742 , 48.63 , 1.83 , 6.28};
   Output = K[0]*theta + K[1]*alpha + K[2]*dTheta + K[3]*dAlpha;
    if(Output>0)
    {
      analogWrite(8,0);
      if(Output>255) Output=255;
      analogWrite(9,abs(Output));
    }
    else
    {
      analogWrite(9,0);
      if(Output<-255) Output=-255;
      analogWrite(8,abs(Output));
    }
    thetaOld=theta;
    alphaOld=alpha;
    delay(msDelay);
  }
}
```

6 Challenges Faced

6.1 Fine-Tuning Q and R matrices

Adjusting the four parameters within the Q matrix, which are pivotal in error control, proved to be a complex task. While we had a broad understanding of how these adjustments would influence the pendulum's performance, achieving precise tuning posed difficulties.

Refining these values demanded thoughtful decision-making. We needed to determine which parameter to modify based on the specific enhancements we aimed to achieve in the pendulum's behavior. The process was intricate because the alterations were often subtle. Maintaining a vigilant watch over the pendulum for extended periods became essential to detect any positive changes.

On occasion, our adjustments led to less desirable outcomes in the pendulum's response. When this occurred, we were required to revert to our previous values and explore alternative modifications, or even consider adjustments in other system parameters. This iterative fine-tuning process necessitated not only patience but also a keen observational acumen to ensure that the pendulum's behavior remained in alignment with our intended objectives.

6.2 Iterative Experimentation and Troubleshooting

Throughout our work on the upright pendulum project, we regularly found ourselves in situations where our system didn't behave as we initially expected. We had to engage in troubleshooting to resolve issues and implement various adjustments. These adjustments included alterations to the control method, modifications to the circuitry, and fine-tuning of mechanical components. This iterative process served as a valuable learning experience, offering insights into problem-solving and the ability to adapt effectively to unforeseen challenges.

6.3 Circuit Design and Motor Driver Challenges

Designing the electrical circuit for our pendulum control system presented us with a series of intricate challenges. We grappled with issues involving motor drivers, circuit components, and the overall electrical connections. Ensuring precise compatibility between the motor driver and the microcontroller or computer was essential to achieve accurate pendulum control. However, this compatibility wasn't always straightforward to achieve. Notably, we frequently had to replace the L293D motor driver due to recurring faults, which added complexity to our project.

Furthermore, a substantial portion of our time, roughly one-third of our lab sessions, was dedicated to debugging the circuit. Eventually, through this troubleshooting process, we uncovered that a faulty power source was at the root of the issue. Addressing and overcoming these challenges required a combination of theoretical knowledge and practical troubleshooting skills.

7 Results, Observations and Inferences

- Q Matrix Entries:
  - **State Variables Weight** (Qii): When specific state variables in the Q matrix are given higher weights, it leads to a greater emphasis on the corresponding state's behavior. For instance, if we assign a higher value to Q(1,1) (alpha), the control system will prioritize maintaining the upright position of the pendulum, i.e., alpha.
  - **Off-Diagonal Entries** (Qij, i j): These entries indicate the importance of the relationships between different state variables. Non-zero off-diagonal terms in the Q matrix indicate that changes in one state variable significantly affect another. For instance, a non-zero Q(1,3) implies that changes in alpha (position) may substantially influence alphadot (angular velocity).Here we have **set the off diagonal entries to be zero** assuming independence between differnt state variables to make the problem simpler.
- **R Matrix Entries**:

  Control Effort Weight (Rii): Larger values in the diagonal of the R matrix imply a higher cost associated with control effort. A higher R(1,1) means that the system will aim for more energy-efficient control. Lower control efforts will be favored, which may result in smoother, but potentially slower, responses.

After extensive fine tuning to get the best possible results we got the values of Q and R matrices to be :

$$\mathbf{Q} = \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad \text{wrt. } \mathbf{R} = [1]$$

This gave us our Control Gain Matrix **K**:

$$\mathbf{K} = \begin{bmatrix} -3.6742 & 48.63 & -1.83 & 6.28 \end{bmatrix}$$

- **Final Results**
  - **Pendulum angle error** = 2°
  - **Shaft angle error** = 24°

8 Challenges Faced

Fine-Tuning Q and R Matrices

Adjusting the four values in the Q matrix, which play a critical role in controlling errors, was indeed challenging. We had a general idea of how these adjustments would impact the pendulum's behaviour, but achieving precise tuning was a bit tricky. To refine these values, we had to make thoughtful decisions about which one to modify, depending on the specific improvements we sought in the pendulum's performance. This process proved to be demanding because the changes we made were often subtle. We had to keep a very close eye on the pendulum for extended periods to detect any positive changes.

At times, our adjustments led to less favourable behaviour in the pendulum's response. When that happened, we had to backtrack to our previous values and try different modifications or even explore adjustments in other settings. This iterative fine-tuning process required both patience and a keen observational eye to ensure that the pendulum's behaviour aligned with our intended goals.

Iterative Experimentation & Troubleshooting

While working on our upright pendulum project, we frequently encountered situations where our system didn't perform as anticipated. We had to troubleshoot issues and make adjustments. We often needed to modify the control method, make changes to the circuits, and fine-tune mechanical components. This iterative process provided valuable insights into problem-solving and adapting to unexpected circumstances.

Circuit Design and Motor Driver Challenges

Developing the electrical circuit for our pendulum control system introduced several intricate challenges. We grappled with issues related to motor drivers, circuit components, and overall electrical connections. Ensuring the motor driver's precise compatibility with the microcontroller or computer was critical for accurate pendulum control.