Inverted Pendulum

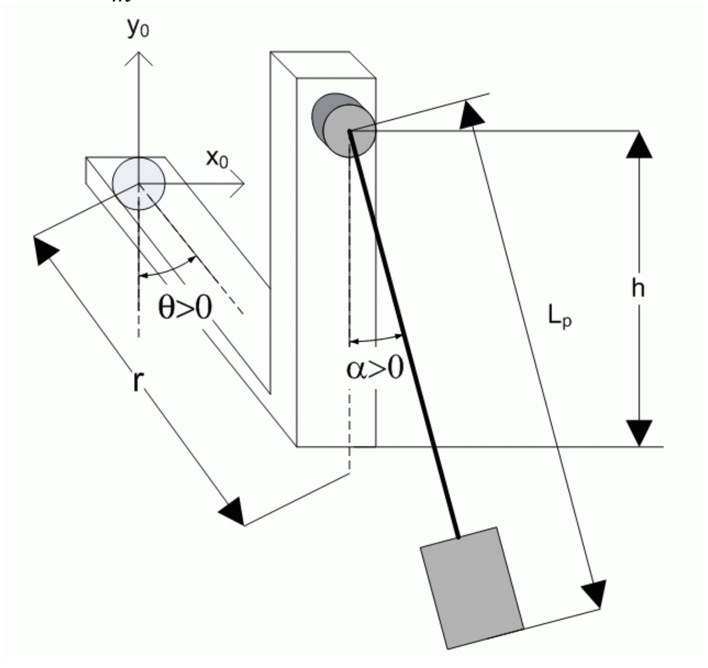
Group Number 3 – Monday Batch

OV Shashank 14D070021 : Ayush Agrawal 14D070042 : Pratik Brahma 14D070003

AIM

To balance the rotatory inverted pendulum in the inverted position given the following transient specifications: Design an LQR control, that is tune the Q weighing matrix, such that the closed-loop response meets the following specifications:

1. Arm Regulation: $|\theta(t)| < 30^\circ$
2. Pendulum Regulation: $|\alpha(t)| < 3^\circ$
3. Control input limit: $V_m < 12\,V$



LQR CONTROL

Here we use LQR control instead of PID control because of the difficulty in tuning the PID parameters for the rotatory inverted pendulum setup and also because a certain control on the input is required and the balance is on a non-equilibrium point which would require more precise control.

The LQR control for the inverted pendulum is applied on its linearized model. It is the theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the LQ problem. One of the main results in the theory is that the solution is provided by the linear–quadratic regulator (LQR), a feedback controller whose equations are given below. For a linear system described by $\dot{x} = Ax + Bu$ with a cost function defined as

$$J = \int_0^\infty [x^T Q x + u^T R u] dt$$

the feedback control law that minimizes the value of the cost $u = -Kx$

Where the matrix K is calculated using the MATLAB lqr command by supplying it the matrices A, B, Q and R. The diagonal matrices Q and R are tweaked to obtain the optimal empirical values for K.

Inverted Pendulum

Group Number 3 – Monday Batch

OV Shashank 14D070021 : Ayush Agrawal 14D070042 : Pratik Brahma 14D070003

THE K MATRIX

After many iterations, the following is the K matrix which was used

$$K = [-3.354 \quad 57.206 \quad -2.041 \quad 7.4801]$$

Here is the MATLAB code to do the same

```
% Code to solve for the optimum state feedback law u = -Kx
% given the state-space model for the inverted pendulum

%% Mechanical Parameters in SI Units
Mp = 0.027;                 % Mass of the pendulum assembly
lp = 0.153;                 % Length of pendulum centre of mass from pivot
Lp = 0.191;                 % Total length of pendulum
r = 0.08260;                % Length of arm pivot to pendulum pivot
Jm = 3E-5;                  % Motor shaft moment of inertia
Marm = 0.028;               % Mass of arm
g = 9.810;                  % Gravitational acceleration constant
Jeq = 1.23E-4;              % Equivalent moment of inertia about the motor
shaft pivot axis
Jp = 1.1E-4;                % Pendulum moment of inertia about its pivot axis
Beq = 0;                    % Arm viscous damping
Bp = 0;                     % Pendulum viscous damping

%% Electro-Mechanical Parameters in SI Units
Rm = 3.3;                   % Motor armature resistance
Kt = 0.02797;               % Motor torque constant
Km =  0.02797;              % Motor back-electromotive force constant

%% State-Space Definition
denom = (Jp*Jeq + Mp*lp^2*Jeq + Jp*Mp*r^2);     %denominator term

% The A Matrix
A = [0, 0, 1, 0;
      0, 0, 0, 1;
      0, r*Mp^2*lp^2*g / denom, -Kt*Km*(Jp + Mp*lp^2) / (Rm*denom), 0;
      0, Mp*lp*g*(Jeq + Mp*r^2) / denom, -Mp*lp*Kt*r*Km / (Rm*denom), 0];
A
% The B Matrix
B = [0; 0; Kt*(Jp + Mp*lp^2) / (Rm*denom); Mp*lp*Kt*r / (Rm*denom)];

%% The Quadratic Minimisation Matrices (Tuning Component)
% The Q Matrix: State variables Cost
Q = [45, 0, 0, 0;
      0, 25, 0, 0;
      0, 0, 4.25, 0;
      0, 0, 0, 3];          % x'Qx where x = [theta, alpha, dtheta/dt,
dalpha/dt]
% The R Matrix: Input Cost
R = 4;                            % Because we have only one input

%% The LQR Solution using the in-built MATLAB Function
[K, ~, ~] = lqr(A, B, Q, R);
disp(K);
```

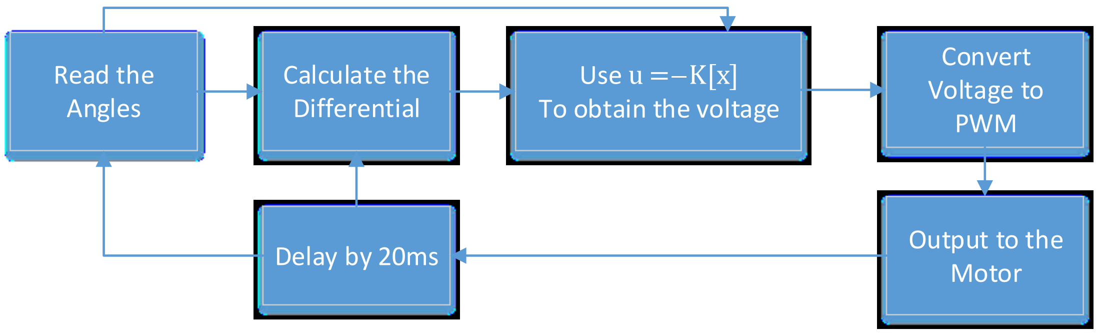
Inverted Pendulum

Group Number 3 – Monday Batch

OV Shashank 14D070021 : Ayush Agrawal 14D070042 : Pratik Brahma 14D070003

ARDUINO CODE AND FLOW

Code Flow



Cut-Down Arduino Code

```
//Code for LQR Control of the Inverted Pendulum
#include <Metro.h>      //Metro libary for timed differential calculations

float theta,alpha, alpha_dash, theta_dash;
float alpha_prev = 0;
float theta_prev = 0;
Metro diffMetro = Metro(20);    //Metro instance for 20 ms

float k1 = -3.3541;
float k2 = 57.2059;
float k3 = -2.0408;
float k4 = 7.4801;

//Reads theta for pick = true and alpha otherwise
int encoderDecoderRead(bool pick);

void loop() {
  if(diffMetro.check() == 1){
    theta = encoderDecoderRead(true)*PI/1024;
    alpha = ((encoderDecoderRead(false))*PI)/1024;
    alpha_dash = (alpha - alpha_prev)/0.02;
    theta_dash = (theta - theta_prev)/0.02;
    alpha_prev = alpha;
    theta_prev = theta;

    float voltage = k1*theta + k2*alpha + k3*theta_dash + k4*alpha_dash;
    if(voltage>0)
    {
      analogWrite(pin2,min(voltage*255/12.0,255));
      analogWrite(pin1,0);
    }
    else
    {
      analogWrite(pin2,0);
      analogWrite(pin1,min(-voltage*255/12.0,255));
    }
  }
}
```

Inverted Pendulum

Group Number 3 – Monday Batch

OV Shashank 14D070021 : Ayush Agrawal 14D070042 : Pratik Brahma 14D070003

PROBLEMS FACED

- The calibration of the sensors required some experimentation to get the angles right for the calculation
- The initial guess for the factors did not work out well because the swing in theta was too high and hence we couldn't control the inverted pendulum
- Also after limiting the theta swing we faced the issue for jerky control of the inverted pendulum which further required careful tuning of the pendulum.