EE128 Lab 6d Writeup

Chris Mitchell, Shangge Liu

December 2nd, 2018

Contents

1 Purpose

The purpose of this lab is to combine the prior 3 weeks of work on pendulum balancing with a swing-up controller in order to implement a self-erecting inverted pendulum. This serves as the culmination of a semester of control systems.

2 Prelab

For the swing-up controller, we used an energy-pumping method, which requires us to calculate the energy of the pendulum.

2.1 Energy of Pendulum

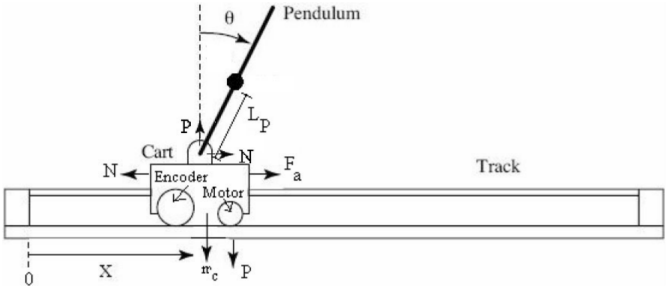

Figure 1: Physical Setup

2.1.1 Energy Calculation

For the energy of the pendulum, one can write it as a function of θ and $\dot{\theta}$.

$$E(\theta, \dot{\theta}) = E_{Kinetic} + E_{Potential}$$

$$E_{kinetic} = \frac{1}{2} J\dot{\theta}^2$$

$$E_{potential} = -mgL_p(1 - cos(\theta))$$

$$E(\theta, \dot{\theta}) = \frac{1}{2} J\dot{\theta}^2 - mgL_p(1 - cos(\theta))$$

Note that potential energy is set such that $E(0, 0) = 0$.

2.1.2 Energy at rest with $\theta = \pi$

$$E(\pi, 0) = -2mgL_p$$

2.1.3 Derivative of Energy

$$J\ddot{\theta} = mgL_p sin(\theta) - mL_p cos(\theta)\ddot{x}$$

$$\frac{dE}{dt} = J\dot{\theta}\ddot{\theta} - mgL_p sin(\theta)\dot{\theta}$$

$$\frac{dE}{dt} = (mgL_p sin(\theta) - mL_p cos(\theta)\ddot{x})\dot{\theta} - mgL_p sin(\theta)\dot{\theta}$$

$$\frac{dE}{dt} = -mL_p\ddot{x}\dot{\theta}cos(\theta)$$

From this equation, we see that to apply the necessary mechanical energy as fast as possible, we should apply the maximal possible positive cart acceleration when $\dot{\theta}cos(\theta) < 0$ and the maximal possible negative cart acceleration when $\dot{\theta}cos(\theta) > 0$. Since, given our motion equations, $\ddot{x}$ is maximal when V is maximal, we are left with a "bang-bang" controller model:

$$V = \begin{cases} 0 & \text{when } E(\theta,\dot{\theta}) \geq 0 \\ V_{\max} & \text{when } E(\theta,\dot{\theta}) < 0 \text{ and } \dot{\theta}\cos(\theta) < 0 \\ -V_{\max} & \text{when } E(\theta,\dot{\theta}) < 0 \text{ and } \dot{\theta}\cos(\theta) > 0 \end{cases}$$

Figure 2: Bang-Bang Controller Equation for our System

In this case, $V_{max} = 6V$, and in practice we will approximate this such that we don't have discontinuities, as discussed later.

2.2 Obtaining Derivatives while Minimizing Phase Shift

Since our linear observer no longer holds for our system (as it is linear and therefore only valid for $\theta \approx 0$, we need a new method of approximating the derivative. Instead, we will use the following approximation of a pure Laplace derivative, essentially a derivative with a low-pass filter:

$$D(s) = s * H_{L_p}(s) = s\frac{1}{c_{L_p}s + 1}$$

We want c_{L_p} to be the largest value such that at the natural frequency $\omega_0 = \sqrt{\frac{g}{L_p}}$, the max absolute phase shift is less than or equal to $5°$.

$$|\phi(H_{L_p}(j\omega_0)| = |-\arctan c_{L_p}\omega_0| \leq 5°$$

$$tan(\pm 5°) = \pm 0.0875 = c_{L_p}\omega_0$$

$$c_{L_p} = \frac{0.875}{\sqrt{9.8/0.3302}} = 0.01606$$

3 Lab Results and Analysis

3.1 Simulink Models

We set up our simulink models by following the instructions on the lab sheet, here are the models we built.

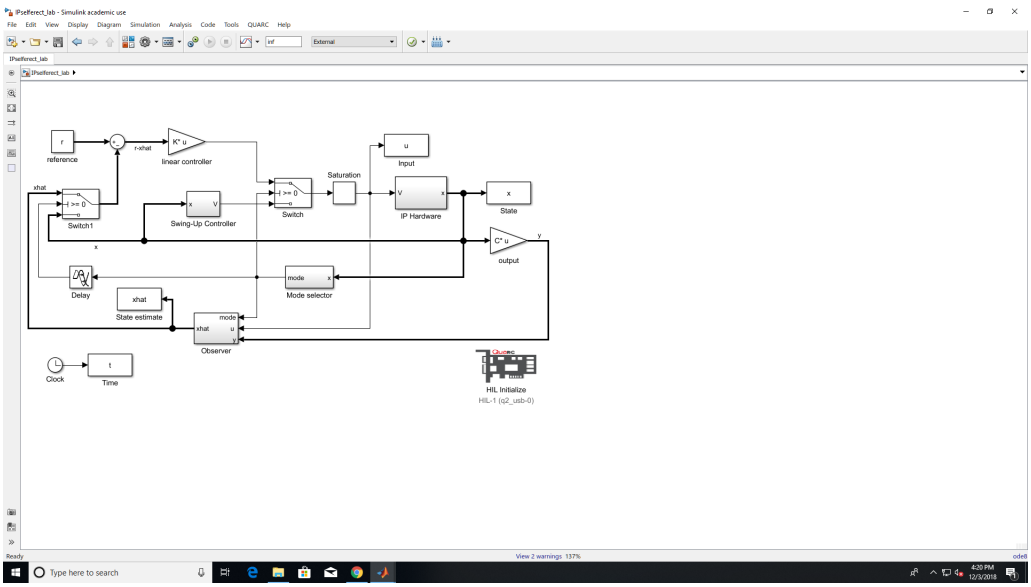

Figure 3: Block Diagram with subsystems

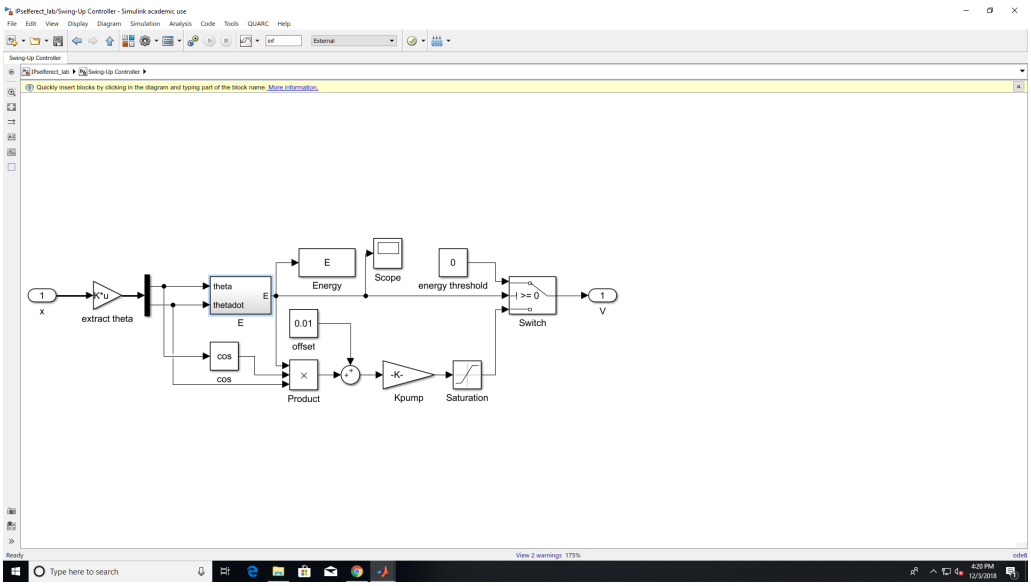

Figure 4: Swingup Controller Subsystem

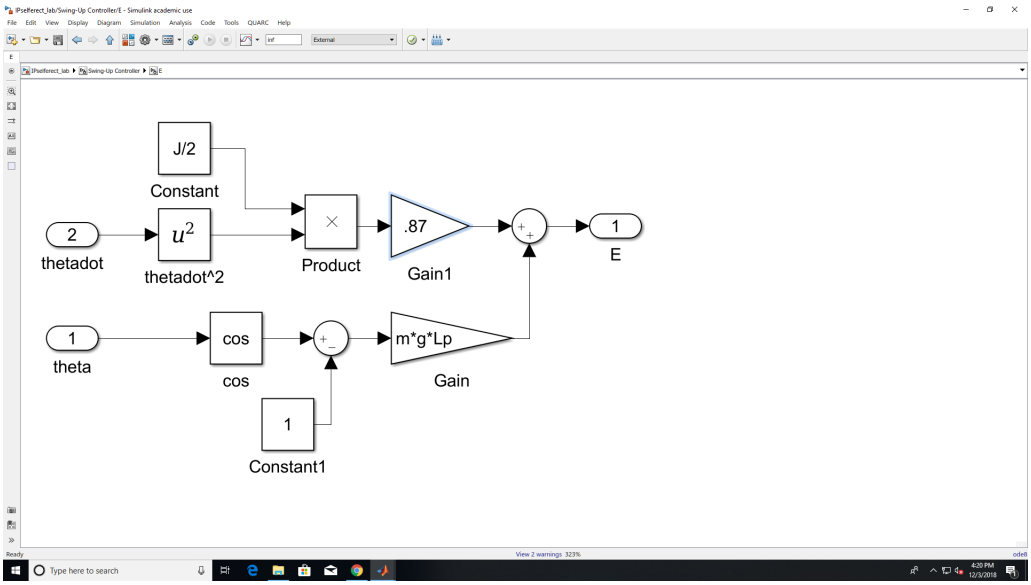

Figure 5: Block Diagram of Energy Calculation

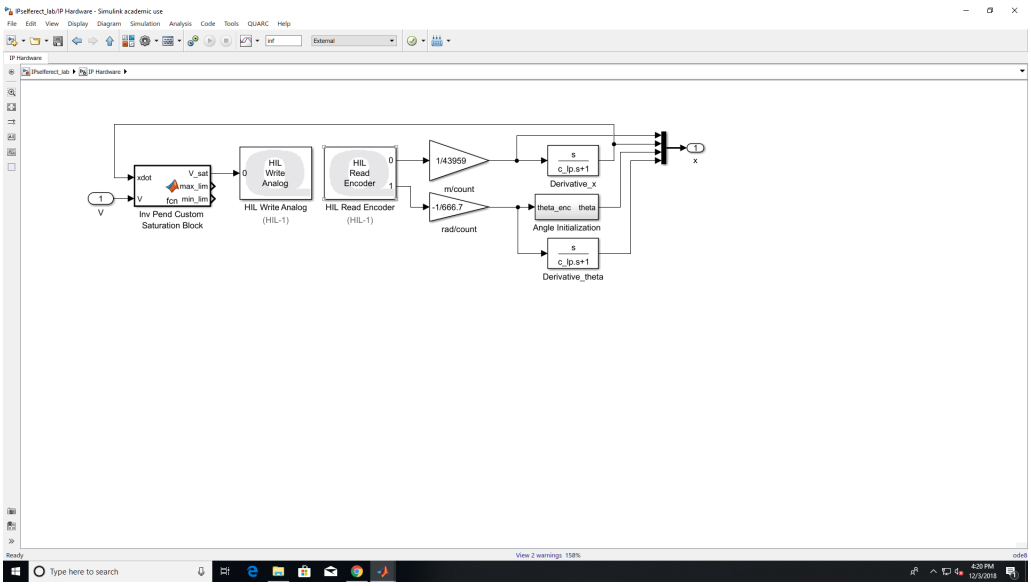

Figure 6: IP Hardware Subsystem

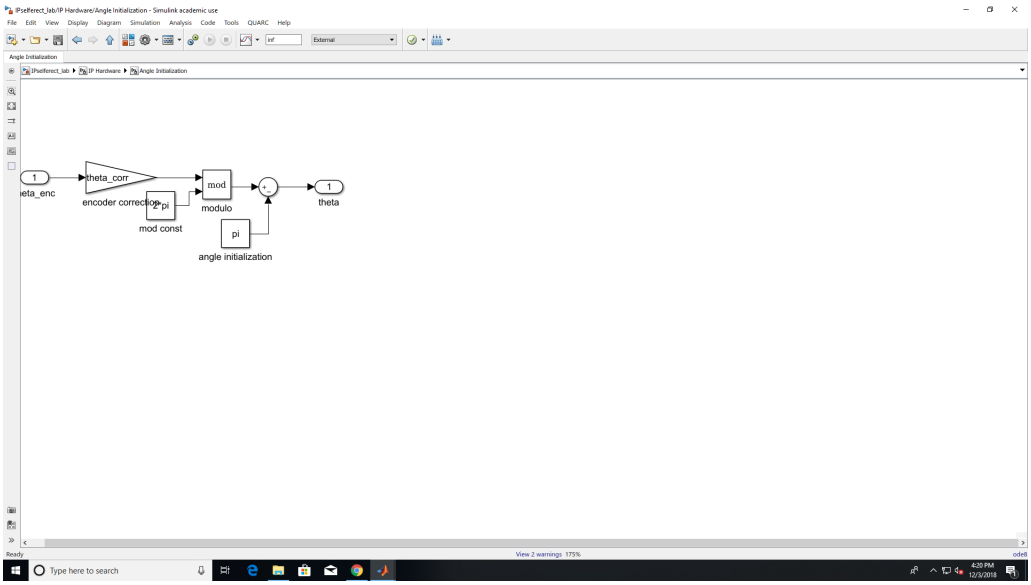

Figure 7: Angle Initialization

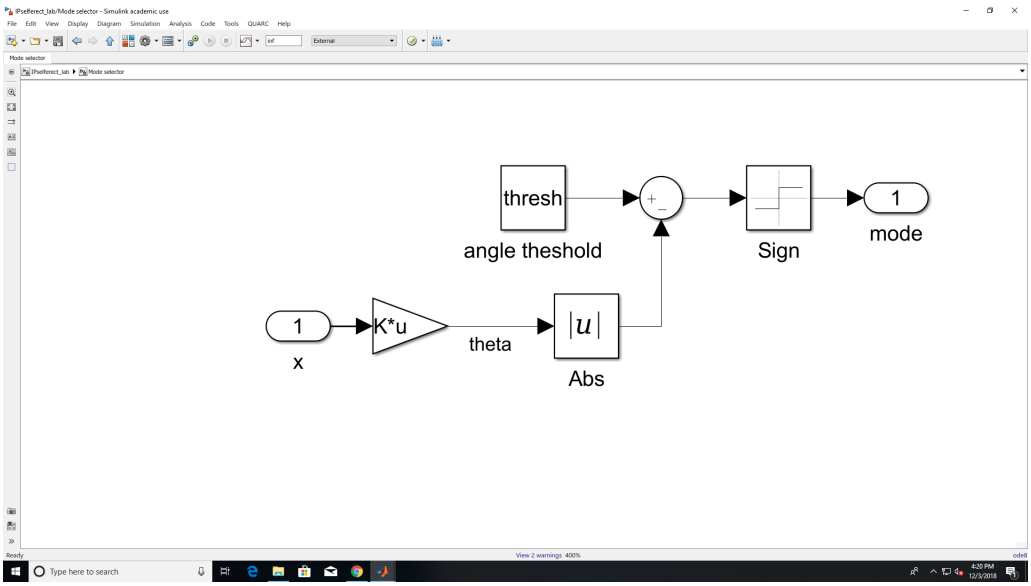

Figure 8: Mode Selector Subsystem

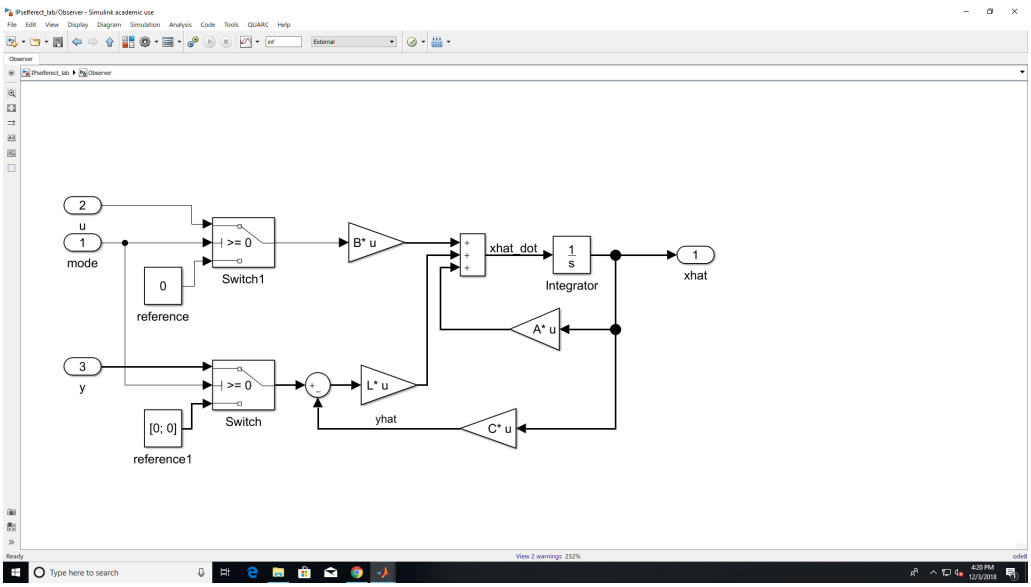

Figure 9: Observer Subsystem

When designing the swing up controller, we built the energy function block to calculate the mechanical energy of the system right now. The block was built using the parameters and functions derived in the prelab. And we used

the cosine function to drive the system and provide energy at first. To pump in the right amount of energy, we set our Kpump at 50. After that, we used a switch to compare the actual energy and the energy of the final stable state, which is 0. When E is larger than zero, the input voltage decreases and stop providing energy anymore.

3.2 System Response and Plots

After buliding, connecting and running our model, we successfully swing the pendulum up and keep it up vertically. Here is the video link:

https://youtu.be/RJb3emAefok

Here are the plots of the system response:

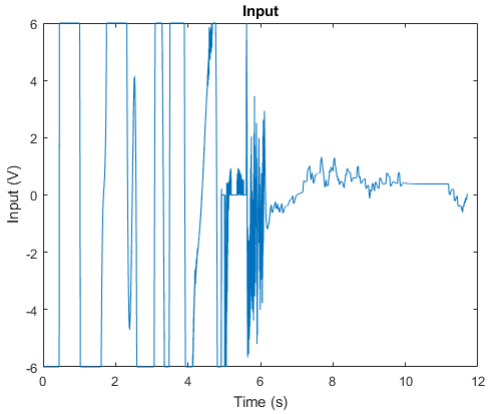

Figure 10: Plot of the Input Voltage

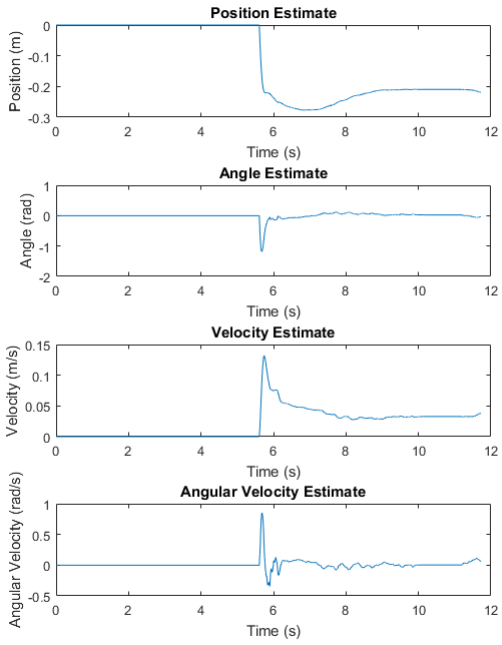

Figure 11: Plot of the Observer State Estimate

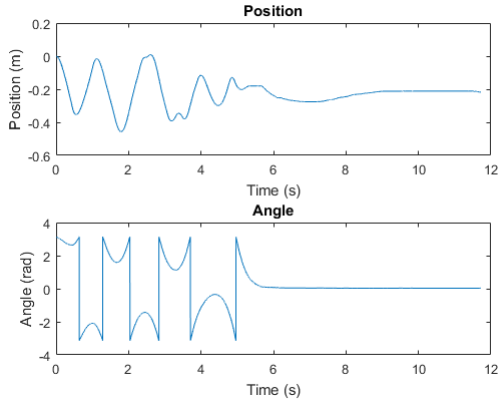

Figure 12: Plot of the Position and Angle

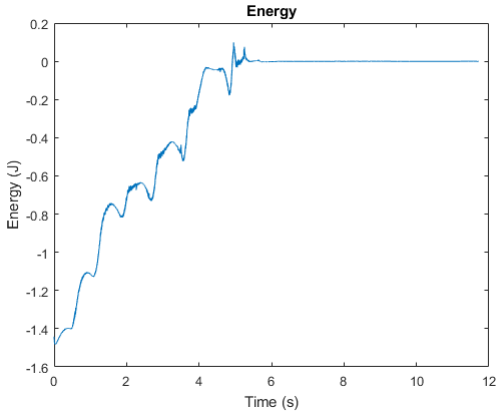

Figure 13: Plot of the Energy

Appendices

A Code for the System Setup

```
    a22 = -6.81;
a23 = -1.5;
a42 = 15.47;
a43 = 25.67;

b2 = 1.52;
b4 = -3.46;

A = [0 1 0 0; 0 a22 a23 0; 0 0 0 1; 0 a42 a43 0];

B = [0; b2; 0 ; b4];

C = [1 0 0 0; 0 0 1 0];
D = [0; 0];

sys = ss(A, B, C, D);

q1 = 5;
q3 = 1;
r = 2;

q1_bar = q1/(0.3^2);
q3_bar = q3/(0.05^2);
```

```
r_bar = r/36;

Q = diag([q1_bar 0 q3_bar 0]);
R = r_bar;
N = 0;

K = lqr(sys, Q, R, N);

obs_poles = [-10 + 15j, -10 - 15j, -12 + 17j, -12 - 17j];
L = place((A-B*K)', C', obs_poles)';

thresh = 10*pi/180;
time_delay = .5;
Kpump = 50;
SAT = 6;
start_time = 1;
c_lp = .01606;
```

B Code for Figures Plotting

```
    clf;

subplot(3, 1, 1);
plot(tout, y(:, 1), '-');
title('Position for Sine Case')
xlabel('Time (s)')
ylabel('Position (m)')
xlim([0 10])

subplot(3, 1, 2);
plot(tout, y(:, 2), '-');
title('Angle for Sine Case')
xlabel('Time (s)')
ylabel('Angle (rad)')
xlim([0 10])

subplot(3, 1, 3);
plot(tout, u, '-');
title('Input for Sine Case')
xlabel('Time (s)')
ylabel('Input (V)')
xlim([0 10])
```

C Code for the Energy Function

```
    function E = energy(theta, thetadot)
Lp = .3302;
m = .230;
g = 9.8;

J = (4/3)*m*Lp*Lp;

E = .5*J*thetadot^2 - m*g*Lp*(1-cos(theta));
```

D Code for the Custom Saturation Block

```
    function [usat, maxlim, minlim]= custom_saturation_block(xdot, u)

maxlim = min(8, max(4, 10*xdot+5));
    minlim = min(-4, max(-8, 10*xdot-5));

    usat = max(minlim, u);
    usat = min(maxlim, usat);
    end
```

E Code for the Angle Initialization

```
    function theta_enc = angle_initialization(theta)
    theta = mod(theta_enc, 2*pi) - pi;
end
```

F Code for the Hardware Lab

```
    g = 9.81;       % m/s^2
m = 0.23;       % kg (long)
Lp = 0.3302;     % m (long)
M = 0.94;       % kg
r = 0.00635;    % m
Rm = 2.6;       % ohms
Kt = 0.00767;   % N*m/A
Km = 0.00767;   % V*s/rad
Kg = 3.71;      % no units
Jm = 3.9e-7;    % kg*m^2
J = 4/3*m*Lp^2;    % kg*m^2
J = 0.9*J; % this is to account for the J being off!

% bookkeeping constants
```

```
a = Rm*(r^2*(M + m/4) + Jm*Kg^2);
b = 4*Lp*a/3;
c = r^2*(M + m) + Jm*Kg^2;
gamma = Rm*((M+m)*r^2 + Jm*Kg^2);

% for x1 = x, x2 = x_dot, x3 = theta, x4 = theta_dot
% see lab 6a solutions for derivation
A = [0             1                 0 0; ...
     0 -Kt*Km*Kg^2/a -3/4*m*g*Rm*r^2/a 0; ...
     0             0                 0 1; ...
     0  Kt*Km*Kg^2/b          g*Rm*c/b 0]
B = [0; r*Kt*Kg/a; 0; -r*Kt*Kg/b]
C = [1 0 0 0; 0 0 1 0];
D = zeros(2,1);

%%% Calculate state feedback gain matrix (overridden in LQR)
P1 = [-1.9-10*1i -1.9+10*1i -1.6-1.3*1i -1.6+1.3*1i];
K = acker(A,B,P1);

%%% Calculate observer gain matrix
P2 = [-10-15*1i -10+15*1i -12-17*1i -12+17*1i];
Lt = place(A.',C.',P2);
L = Lt.';

% threshold for switching to stabilizing controller
thresh = 10/360*2*pi;

% compute LQR gain for low position penalty
Q = diag([7/(0.3^2) 0 4/(0.05^2) 0]);
R = 2/64;
[K S e] = lqr(A,B,Q,R);

% coefficient in the TF for the derivative
c_lp = 1/20/pi;

% Kpump
Kpump = 50;

% Reference input
r = [0 0 0 0]';

theta_corr = 1;
```

G Code for Plotting Energy

```
    clf;

plot(t, E, '-')
title('Energy')
xlabel('Time (s)')
ylabel('Energy (J)')
```

H Code for Estimates Plotting

```
    clf;

subplot(4, 1, 1)

plot(t, xhat(:,1), '-')
title('Position Estimate')
xlabel('Time (s)')
ylabel('Position (m)')

subplot(4, 1, 2)

plot(t, xhat(:,2), '-')
title('Angle Estimate')
xlabel('Time (s)')
ylabel('Angle (rad)')

subplot(4, 1, 3)

plot(t, xhat(:,3), '-')
title('Velocity Estimate')
xlabel('Time (s)')
ylabel('Velocity (m/s)')

subplot(4, 1, 4)

plot(t, xhat(:,4), '-')
title('Angular Velocity Estimate')
xlabel('Time (s)')
ylabel('Angular Velocity (rad/s)')
```

I Code for Plotting the Input

```
    clf;
```

```
plot(t, u, '-')
title('Input')
xlabel('Time (s)')
ylabel('Input (V)')
```

J Code for Plotting x

```
    clf;

subplot(2, 1, 1)

plot(t, x(:,1), '-')
title('Position')
xlabel('Time (s)')
ylabel('Position (m)')

subplot(2, 1, 2)

plot(t, x(:,3), '-')
title('Angle')
xlabel('Time (s)')
ylabel('Angle (rad)')
```