PID Control of a Robot Via Position Feedback

Nick Pham

System Modeling

The equations of motion for this cart pendulum system are

$$(m_c + m_p)\ddot{x} + c\dot{x} = F + m_p r \left(\sin(\theta)\dot{\theta}^2 - \cos(\theta)\ddot{\theta}\right)$$
$$\left(J + m_p r^2\right)\ddot{\theta} + \gamma\dot{\theta} = -m_p r g \sin(\theta) - m_p r \ddot{x}\cos(\theta)$$

which can be linearized to

$$(m_c + m_p)\ddot{x} + m_p r\ddot{\theta} + c\dot{x} = f(t)$$
$$m_p r\ddot{x} + \left(J + m_p r^2\right)\ddot{\theta} + \gamma\dot{\theta} - m_p g r\theta = 0$$

In order to write the state space model, solve for $\ddot{\theta}$ and $\ddot{x}$. Let $M = m_c + m_p$ and $I = J + m_p r^2$.

$$\ddot{\theta} = \frac{f(t) - c\dot{x} - M\ddot{x}}{m_p r}$$
$$0 = m_p r\ddot{x} + I\frac{f(t) - c\dot{x} - M\ddot{x}}{m_p r} + \gamma\dot{\theta} - m_p g r\theta$$
$$\ddot{x}\left(m_p r - I\frac{M}{m_p r}\right) = m_p g r\theta - \gamma\dot{\theta} + \frac{Ic\dot{x}}{m_p r} - \frac{If(t)}{m_p r}$$
$$\ddot{x}\left(\frac{m_p^2 r^2 - IM}{m_p r}\right) = m_p g r\theta - \gamma\dot{\theta} + \frac{Ic\dot{x}}{m_p r} - \frac{If(t)}{m_p r}$$

Let $\mu = m_p^2 r^2 - IM$

$$\ddot{x}\left(\frac{\mu}{m_p r}\right) = m_p g r\theta - \gamma\dot{\theta} + \frac{Ic\dot{x}}{m_p r} - \frac{If(t)}{m_p r}$$
$$\ddot{x} = \frac{g}{\mu}\theta - \frac{\gamma m_p r}{\mu}\dot{\theta} + \frac{I}{\mu}(c\dot{x} - f(t))$$

Similarly,

$$
\begin{aligned}
\ddot{x} &= \frac{f(t) - c\dot{x} - m_p r\ddot{\theta}}{M} \\
0 &= m_p r \frac{f(t) - c\dot{x} - m_p r\ddot{\theta}}{M} + I\ddot{\theta} + \gamma\dot{\theta} - m_p g r\theta \\
\ddot{\theta}\left(I - \frac{m_p^2 r^2}{M}\right) &= m_p g r\theta - \gamma\dot{\theta} + \frac{m_p r}{M}(c\dot{x} - f(t)) \\
\ddot{\theta}\left(-\frac{\mu}{M}\right) &= m_p g r\theta - \gamma\dot{\theta} + \frac{m_p r}{M}(c\dot{x} - f(t)) \\
\ddot{\theta} &= -\frac{M}{\mu} m_p g r\theta + \frac{M}{\mu}\gamma\dot{\theta} - \frac{m_p r}{\mu}(c\dot{x} - f(t))
\end{aligned}
$$

This means that the state space equations can be written in matrix form

$$
\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{g}{\mu} & \frac{Ic}{\mu} & -\frac{\gamma m_p r}{\mu} \\ 0 & -\frac{M m_p g r}{\mu} & -\frac{m_p r c}{\mu} & \frac{M\gamma}{\mu} \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\frac{I}{\mu} \\ -\frac{m_p r}{\mu} \end{bmatrix} f(t)
$$

Because the system can only observe x and θ, the output is

$$
\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}
$$

Stability Analysis

Using the A, B, C, D matrices computed in the previous section, simulate the state space model in MATLAB. Because some of the eigenvalues of A have positive real parts, the system is currently unstable.

The a reasonable step input might be an angle of 0.1. The resulting step response is shown in Figure 1. As can be seen, the unstable response was indicated by the eigenvalues of A.

Full State Feedback

To implement a feedback control system of the form $u = -Kx$, the new state space model would be

$$
\begin{aligned}
\dot{x} &= Ax + Bu = Ax - BKx = (A - BK)x \\
y &= Cx + Du
\end{aligned}
$$

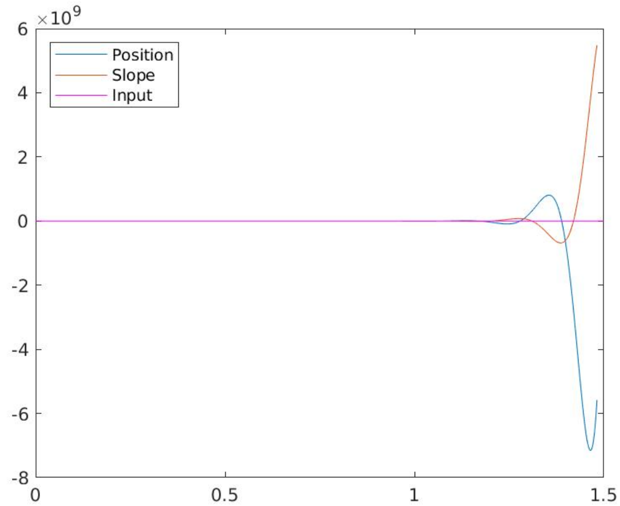


Fig. 1: Unstable step response of open loop system.

Thus, for a stable system, the eigenvalues of $(A - BK)$ must have negative real parts. MATLAB's `place()` function is used to set K for a given vector of eigenvalues, which can be seen in the attached code. Figure 2 shows the step responses for increasing eigenvalues. As can be seen, with larger magnitude eigenvalues the system allows less deviation and has quicker stabilization than with smaller eigenvalues.

Hardware Implementation

The system was tested with the following K values, and the system response was observed.

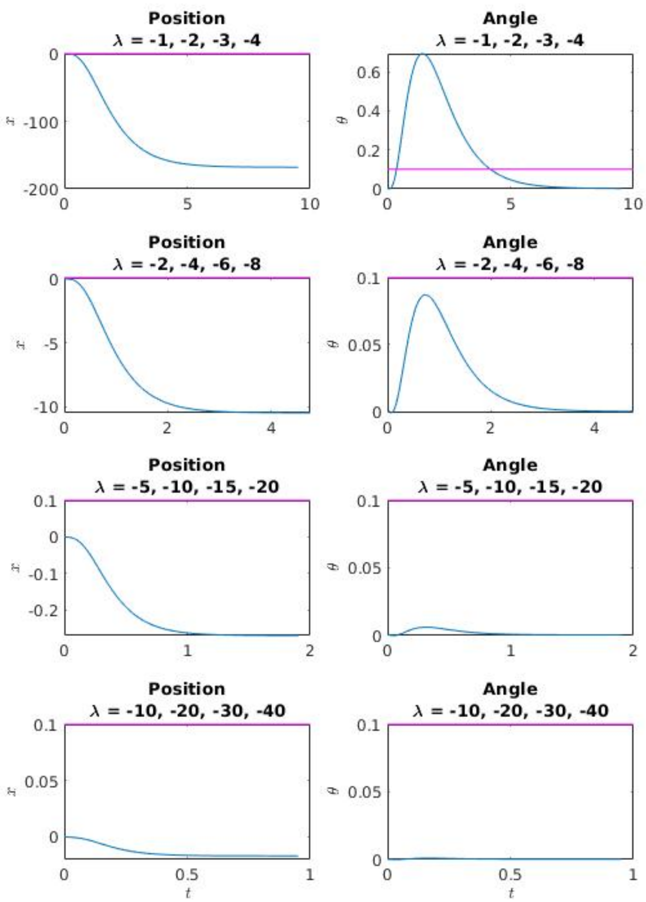


Fig. 2: Step response of closed loop system with different controllers.

k_p^x	k_d^x	k_p^θ	k_d^θ	Observation	Explanation
20	40	-100	-20	Balanced and Smooth	Factory Setting
30	40	-100	-20	More Reactive to presses. Wants to keep x centered but not as resistant to pushes for balancing.	Increasing the importance of the x control in relation to the θ control means that the angle control will be less effective.
40	40	-100	-20	Very jittery, overshoots even without a tap.	The large x proportional control means that the system attempts to correct any errors with a large force, which results in overshoot. This oscillation prevents the system from converging.
10	40	-100	-20	Drifts somewhat in x, but quite robust in θ.	Now the tradeoff works the other way, where the effectiveness of the angle control is prioritized.
5	40	-100	-20	The pendulum can be pushed in the x axis, while it remains upright despite large forces.	Because there is so little emphasis on the x control, the system only attempts to correct for it when there is no external input. Otherwise, it works hard to keep the pendulum upright.
20	40	-50	-20	Easy to knock over.	This strong deemphasis on the angular control means that the system is easy to tip over. The gears were also slipping quite a bit, which meant that the angular controller did not have a good mechanism to control the system, contributing to the tipping.
20	40	-150	-20	Decent at balancing despite the slipping gears. Drifts in x.	The large proportional angular control allows the system to balance despite physical imperfections.
20	40	-150	-30	Jittery behavior reduces convergence.	At this point, the physical system barely worked because of the gears, but increasing the derivative control resulted in a decrease in convergence.

The major takeaways from this lab were the importance in balancing controller parameters in multiple input single output systems, where prioritizing one input results in reduced performance for the other. Another takeaway was the difficulty of implementing ideal controllers in hardware, where many imperfections are difficult to model.

Part 1

Contents

1.a

```
% Given Constants
g   = 9.81;         % gaccel[m/s^2]
mp  = 0.230;        % massofpendulum[kg]
l   = 0.6413;       % lengthofpendulum[m]
r   = l/2;          % radiustoCOMofpendulum[m]
J   = (1/3)*mp*l^2; % inertiaofpendulumrotatingabout1end[kg-m^2]
y   = 0.0024;       % pendulumdamping[N-m*s]
mc  = 0.38;         % massofcart[kg]
c   = 0.90;         % cartdamping[N-s/m]

% Derived Constants
Mhat = mp + mc;
Jhat = J + mp*r^2;
mu = mp^2 * r^2 - Jhat^2 * Mhat^2;

A = [0 0 1 0;
     0 0 0 1;
     0, g/mu, (Jhat*c)/mu, -(y*mp*r)/mu;
     0, -(Mhat*mp*r*g)/mu, -(mp*r*c)/mu, (Mhat*y)/mu]

B = [0; 0; -Jhat/mu; -(mp*r)/mu]

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

D = [0; 0]
```

```
A =

   1.0e+03 *

         0         0    0.0010         0
         0         0         0    0.0010
         0    2.2782    0.0115   -0.0000
         0   -0.1025   -0.0154    0.0003


B =

         0
         0
  -12.8140
  -17.1268


C =

     1     0     0     0
     0     1     0     0


D =

     0
     0
```

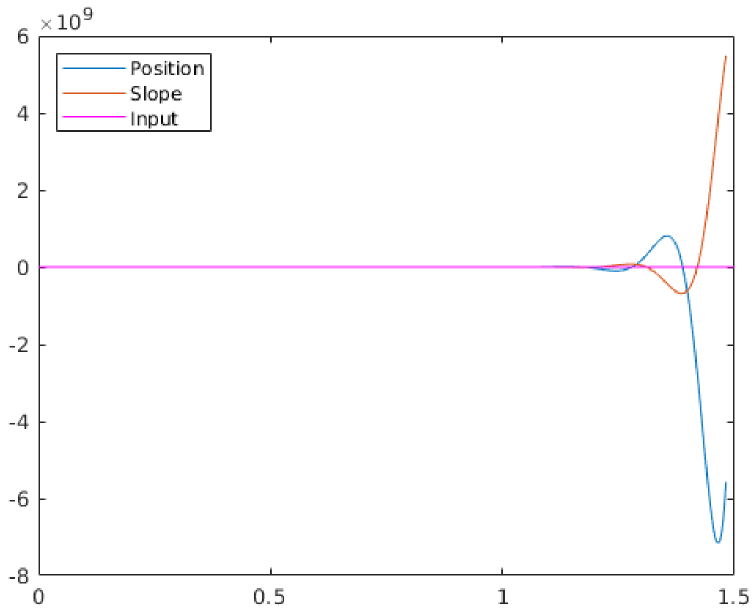
1.b

```
sys = ss(A, B, C, D)
eig(A)
```

```
opt = stepDataOptions('StepAmplitude', 0.1)
[y, t, x] = step(sys, opt);

figure(1); clf;
plot(t, y)
hline = refline(0, 0.1)
hline.Color = 'm'

legend("Position", "Slope", "Input", 'Location', 'Northwest')
saveas(gca, 'ES155Lab2_1b_step.jpg')
```

```
sys =

  A =
            x1       x2       x3       x4
   x1        0        0        1        0
   x2        0        0        0        1
   x3        0     2278    11.53  -0.0411
   x4        0   -102.5   -15.41     0.34

  B =
           u1
   x1       0
   x2       0
   x3  -12.81
   x4  -17.13

  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.


ans =

   0.0000 + 0.0000i
  19.8901 +28.6414i
  19.8901 -28.6414i
 -27.9076 + 0.0000i


opt =

  step with properties:

      InputOffset: 0
    StepAmplitude: 0.1000


hline =

  Line with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [0 1.5000]
              YData: [0.1000 0.1000]
              ZData: [1×0 double]

  Use GET to show all properties


hline =

  Line with properties:
```

```
            Color: [1 0 1]
        LineStyle: '-'
        LineWidth: 0.5000
           Marker: 'none'
       MarkerSize: 6
  MarkerFaceColor: 'none'
            XData: [0 1.5000]
            YData: [0.1000 0.1000]
            ZData: [1×0 double]

Use GET to show all properties
```



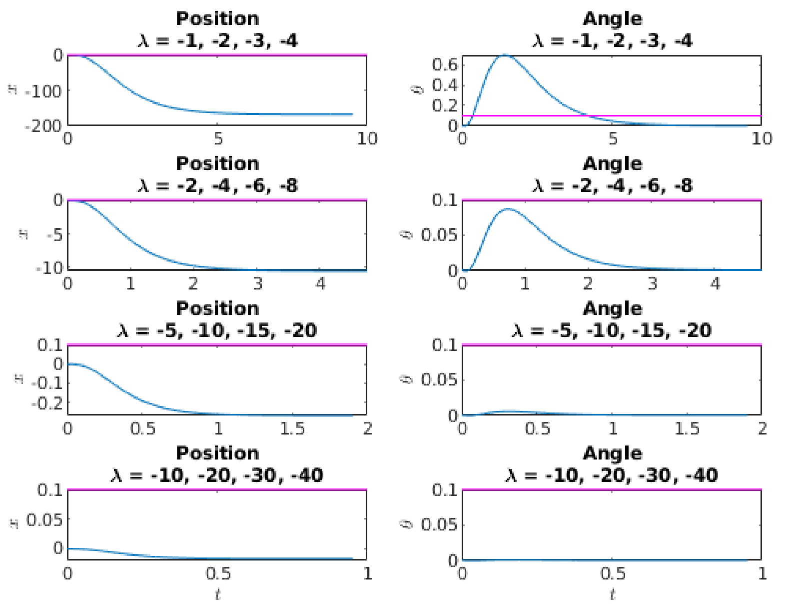
1.c

```
% the smaller poles allow the pendulum to be pushed around before reaching
% an equilibrium, while the large value poles keep the pendulum at the
% equilibrium point theta = 0

figure(2); clf;

plotCount = 1;
pMultipliers = [1, 2, 5, 10]
for i = 1:length(pMultipliers)
    p = [-1, -2, -3, -4];
    p = p.*pMultipliers(i)
    K = place(A, B, p)

    sys = ss(A- B*K, B, C, 0);
    opt = stepDataOptions('StepAmplitude', 0.1);
    [y, t, x] = step(sys, opt);

    titles = ["Position"; "Angle"];
    ylabels = ["$x$", "$\theta$"]
    for j = 1:2
        subplotIdx = plotCount + j -1
        subplot(length(pMultipliers),2, subplotIdx)
        plot(t, y(:,j))
        hline = refline(0, 0.1);
        hline.Color = 'm';
        title({char(titles(j)), ['\lambda = ', num2str(p(1)), ', ', num2str(p(2)), ', ', num2str(p(3)), ', ', num2str(p(4))]})
        ylabel(char(ylabels(j)), 'Interpreter', 'latex')
    end

    plotCount = plotCount + 2;
end
```

```
subplot(length(pMultipliers), 2, plotCount - 2)
xlabel('$t$', 'Interpreter', 'latex')

subplot(length(pMultipliers), 2, plotCount - 1)
xlabel('$t$', 'Interpreter', 'latex')
```

```
pMultipliers =

     1     2     5    10


p =

    -1    -2    -3    -4


K =

   -0.0006  -33.9358    0.5078   -1.6570


ylabels =

  1×2 string array

    "$x$"    "$\theta$"


subplotIdx =

     1


subplotIdx =

     2


p =

    -2    -4    -6    -8


K =

   -0.0095  -50.6099    0.3358   -2.1122


ylabels =

  1×2 string array

    "$x$"    "$\theta$"


subplotIdx =

     3


subplotIdx =

     4


p =

    -5   -10   -15   -20


K =

   -0.3719 -119.7142   -0.4862   -3.2488


ylabels =
```

```
  1×2 string array

    "$x$"    "$\theta$"


subplotIdx =

     5


subplotIdx =

     6


p =

   -10   -20   -30   -40


K =

   -5.9507 -289.5229   -3.2349   -4.1117


ylabels =

  1×2 string array

    "$x$"    "$\theta$"


subplotIdx =

     7


subplotIdx =

     8
```



Published with MATLAB® R2018a