Inverted Pendulum - State-state model and LQR

The equations of motion of the inverted pendulum are given as follows (for the derivation of the equations, please see the equations of motion notes posted on canvas).

$$(I + ml^2)\ddot{\emptyset} - mgl\emptyset = ml\ddot{x} \tag{1}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\emptyset} = F \tag{2}$$

Step 1: Define the state variables:

The state variables can be given as follows.

$$\begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix}$$

The state-space model is given by

$$[\dot{x}] = [A][x] + [B][u]$$

Therefore, the state-state model for the inverted pendulum can be defines as

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = [A] \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + [B]F$$

Step 2: Find $[A]$ and $[B]$ for the state-space model above, using the equations of motions

Rearrange Equation (2) as:

$$\ddot{\emptyset} = \frac{F - (M + m)\ddot{x} - b\dot{x}}{-ml} \tag{3}$$

Rearrange Equation (1) as:

$$\ddot{x} = \frac{(I + ml^2)}{ml}\ddot{\emptyset} - g\emptyset \tag{4}$$

Substitute Equation (3) into Equation (4):

$$\ddot{x} = \frac{(I + ml^2)}{ml} . \frac{F - (M + m)\ddot{x} - b\dot{x}}{-ml} - g\emptyset \tag{5}$$

Rearrange Equation (5) to obtain $\ddot{x}$ for matrices $[A]$ and $[B]$:

$$\ddot{x}(1 + \frac{(I + ml^2)}{ml}\frac{(M + m)}{-ml}) = \frac{(I + ml^2)}{ml} . \frac{(F - b\dot{x})}{-ml} - g\emptyset$$

$$\ddot{x}\left(1 + \frac{(I + ml^2)}{ml}\frac{(M + m)}{-ml}\right) = \frac{(I + ml^2)}{ml} . \frac{b\dot{x}}{ml} - g\emptyset - \frac{(I + ml^2)F}{ml\, ml}$$

$$\ddot{x}\left(\frac{-(ml)^2 + (I + ml^2)(M + m)}{-(ml)^2}\right) = \frac{(I + ml^2)}{ml} . \frac{b\dot{x}}{ml} - g\emptyset - \frac{(I + ml^2)F}{ml\, ml}$$

$$\ddot{x}\big(-(ml)^2 + (I + ml^2)(M + m)\big) = \frac{(I + ml^2)}{ml} . \frac{b\dot{x}(-(ml)^2)}{ml} - g\emptyset(-(ml)^2) - \frac{(I + ml^2)F(-(ml)^2)}{ml\, ml}$$

$$\ddot{x}\big(-(ml)^2 + (I + ml^2)(M + m)\big) = -(I + ml^2)b\dot{x} + g\emptyset(ml)^2 + (I + ml^2)F$$

$$\ddot{x}\big(Mml^2 + I(M+m)\big) = -(I+ml^2)b\dot{x} + g\emptyset(ml)^2 + (I+ml^2)F$$

$$\ddot{x} = \frac{-(I+ml^2)b\dot{x} + g\emptyset(ml)^2 + (I+ml^2)F}{\big(Mml^2 + I(M+m)\big)}$$

Show the first and second rows of the $[A]$ and $[B]$ matrices in the state-space model ($[\dot{x}] = [A][x] + [B][u]$):

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{Mml^2 + I(M+m)} & \frac{g(ml)^2}{Mml^2 + I(M+m)} & 0 \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{(I+ml^2)}{Mml^2 + I(M+m)} \\ 0 \\ \square \end{bmatrix} F$$

Now, find $\ddot{\emptyset}$, and complete the state-space model:

$(I+ml^2)\ddot{\emptyset} - mgl\emptyset = ml\ddot{x}$ (1)

$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\emptyset} = F$ (2)

Rearrange Equation (1) as:

$$\frac{(I+ml^2)\ddot{\emptyset} - mgl\emptyset}{ml} = \ddot{x} \qquad (6)$$

Substitute Equation (6) into Equation (2):

$$(M+m)\frac{(I+ml^2)\ddot{\emptyset} - mgl\emptyset}{ml} + b\dot{x} - ml\ddot{\emptyset} = F$$

Rearrange to find $\ddot{\emptyset}$:

$$(M+m)\frac{(I+ml^2)\ddot{\emptyset}}{ml} - \frac{mgl(M+m)\emptyset}{ml} + b\dot{x} - ml\ddot{\emptyset} = F$$

$$(M+m)\frac{(I+ml^2)\ddot{\emptyset}}{ml} - ml\ddot{\emptyset} - \frac{mgl(M+m)\emptyset}{ml} + b\dot{x} = F$$

$$\left[(M+m)\frac{(I+ml^2)}{ml} - ml\right]\ddot{\emptyset} - \frac{mgl(M+m)\emptyset}{ml} + b\dot{x} = F$$

$$\left[\frac{(M+m)(I+ml^2) - (ml)^2}{ml}\right]\ddot{\emptyset} - \frac{mgl(M+m)\emptyset}{ml} + b\dot{x} = F$$

$$\left[\frac{(M+m)(I+ml^2) - (ml)^2}{ml}\right]\ddot{\emptyset} = \frac{mgl(M+m)\emptyset}{ml} - b\dot{x} + F$$

$$[(M+m)(I+ml^2) - (ml)^2]\ddot{\emptyset} = mgl(M+m)\emptyset - mlb\dot{x} + mlF$$

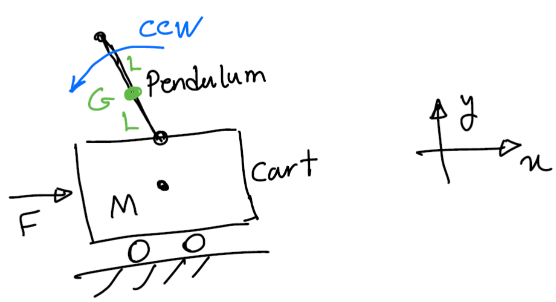
$$\ddot{\emptyset} = \frac{mgl(M+m)\emptyset - mlb\dot{x}}{[(M+m)(I+ml^2) - (ml)^2]} + \frac{mlF}{[(M+m)(I+ml^2) - (ml)^2]}$$

$$\ddot{\emptyset} = \frac{mgl(M+m)\emptyset - mlb\dot{x}}{[Mml^2 + (M+m)(I+ml^2)]} + \frac{mlF}{[Mml^2 + (M+m)(I+ml^2)]}$$

The state-space model will be given by:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{Mml^2 + I(M+m)} & \frac{g(ml)^2}{Mml^2 + I(M+m)} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{Mml^2 + I(M+m)} & \frac{mgl(M+m)}{Mml^2 + I(M+m)} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{(I+ml^2)}{Mml^2 + I(M+m)} \\ 0 \\ \frac{ml}{Mml^2 + I(M+m)} \end{bmatrix} F$$

Inverted pendulum assignment:



1. Find the equations of motion of the inverted pendulum (show all the steps. Don't skip steps). Use the FBD given in the notes on canvas.
2. Find the transfer functions in the following forms:

$$\frac{\emptyset(s)}{F(s)} \text{ and } \frac{X(s)}{F(s)}$$

3. Find the state-state model in the following form:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = [A] \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + [B]F$$

$$y = [C] \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + [D]F$$

Use the following parameters:

M = .5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;

4. Use the following commands in MATLAB to obtain the transfer function:

```
states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x'; 'phi'};
sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs)
```

5. Use the following command to convert state-state model to transfer functions.

```
sys_tf = tf(sys_ss)
```

6. Examine how the system responds to an impulsive force applied to the cart employing the MATLAB command impulse. (Use the same parameters as in part 3 for all the calculations).

```
q = (M+m)*(I+m*l^2)-(m*l)^2;
s = tf('s');
P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I + m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);
P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*l)*s/q - b*m*g*l/q);
sys_tf = [P_cart ; P_pend];
inputs = {'u'};
outputs = {'x'; 'phi'};
set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)
t=0:0.01:1;
impulse(sys_tf,t);
title('Open-Loop Impulse Response')
```

7. Find the zeros and poles of the system using the following commends:

```
[zeros poles] = zpkdata(P_pend,'v')
```

8. Find the Open-loop step response:

```
t = 0:0.05:10;
u = ones(size(t));
[y,t] = lsim(sys_tf,u,t);
plot(t,y)
title('Open-Loop Step Response')
axis([0 3 0 50])
legend('x','phi')
```

9. Design a PID controller for the pendulum. Start with Kp = 1, Ki = 1, Kd = 1. Is the response acceptable:

```
Kp = 1;
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 1, Ki = 1, Kd = 1'});
```

10. Change the parameters of the PID as below and explain if the response is acceptable.

```
Kp = 100;
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 100, Ki = 1, Kd = 1'});
```

11. Change the parameters of the PID as below and explain if the response is acceptable.

```
Kp = 100;
Ki = 1;
Kd = 20;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title({'Response of Pendulum Position to an Impulse Disturbance';'under PID Control: Kp = 100, Ki = 1, Kd = 20'});
```

The same equations from the reference:

https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

The Linear Quadratic Regulator (LQR) approach to solve the state-space model:

Finding the optimal values of the control parameters

Minimize a quadratic cost function, J:

$$J = \int_0^\infty (x^T Q x + u^T R u)\, dt$$

Where, Q minimizes the state of the system (to drive the state of the system to zero), and R controller effort (how much effort). What control value, k, produces the lowest cost J.

For this purpose, let's introduce a matrix P, where $P=P^T$.

Add and subtract:

$$J = {x_0}^T P x_0 - {x_0}^T P x_0 + \int_0^\infty (x^T Q x + u^T R u)\, dt$$

Move ${x_0}^T P x_0$ into the integral:

$$J = {x_0}^T P x_0 + \int_0^\infty \left[\frac{d}{dt}(x^T P x) + x^T Q x + u^T R u\right] dt$$

It is valid because:

$$\int_0^\infty \left[\frac{d}{dt}(x^T P x)\right] dt = \left[\frac{d}{dt}(x^T P x)\right]_0^\infty = 0 - {x_0}^T P x_0$$

Then, find the derivative below:

$$\frac{d}{dt}(x^T P x) = \dot{x}^T P x + x^T P \dot{x}$$

Substitute for $\dot{x}$, using $[\dot{x}] = [A][x] + [B][u]$ (or use this form: $\dot{x} = Ax + Bu$ for simplicity:

$$\frac{d}{dt}(x^T P x) = (Ax + Bu)^T P x + x^T P (Ax + Bu)$$

Substitute in J:

$$J = {x_0}^T P x_0 + \int_0^\infty [(Ax + Bu)^T Px + x^T P(Ax + Bu) + x^T Qx + u^T Ru]\, dt$$

Apply the transpose:

$$J = {x_0}^T P x_0 + \int_0^\infty [(x^T A^T + u^T B^T)Px + x^T P(Ax + Bu) + x^T Qx + u^T Ru]\, dt$$

Multiply the terms:

$$J = {x_0}^T P x_0 + \int_0^\infty [x^T A^T Px + u^T B^T Px + x^T PAx + x^T PBu + x^T Qx + u^T Ru]\, dt$$

Rearrange:

$$J = {x_0}^T P x_0 + \int_0^\infty [x^T A^T Px + x^T PAx + x^T PBu + x^T Qx + u^T B^T Px + u^T Ru]\, dt$$

Factor x^T:

$$J = {x_0}^T P x_0 + \int_0^\infty [x^T (A^T Px + PAx + Qx) + x^T PBu + u^T B^T Px + u^T Ru]\, dt$$

Factor x:

$$J = {x_0}^T P x_0 + \int_0^\infty [x^T (A^T P + PA + Q)x + u^T B^T Px + x^T PBu + u^T Ru]\, dt$$

Rearrange the terms:

$$J = {x_0}^T P x_0 + \int_0^\infty [x^T (A^T P + PA + Q)x + u^T Ru + x^T PBu + u^T B^T Px]\, dt$$

The goal is to minimize u. The term ${x_0}^T P x_0$ does not depend on u, so we can remove it from the cost function.

$$J = \int_0^\infty [x^T (A^T P + PA + Q)x + u^T Ru + x^T PBu + u^T B^T Px]\, dt \tag{7}$$

Rewrite $u^T Ru + x^T PBu + u^T B^T Px$ in the following form by adding the term $x^T (PBR^{-1} B^T P)x$ and subtracting the same term $x^T (PBR^{-1} B^T P)x$.

$$u^T Ru + x^T PBu + u^T B^T Px + [x^T (PBR^{-1} B^T P)x - x^T (PBR^{-1} B^T P)x] \tag{8}$$

We will rewrite the above equation in a different form, which can help in solving the problem.

Start by defining the following term:

$$(u + R^{-1} B^T Px)^T R(u + R^{-1} B^T Px) - x^T (PBR^{-1} B^T P)x \tag{9}$$

Apply the transpose to the first term (knowing that $P^T = P$, and $(R^{-1})^T = R^{-1}$):

$$(u^T + x^T PBR^{-1})R(u + R^{-1} B^T Px) - x^T (PBR^{-1} B^T P)x$$

Multiply the first terms:

$$u^T R u + u^T R R^{-1} B^T P x + x^T P B R^{-1} R u + x^T P B R^{-1} R R^{-1} B^T P x - x^T (P B R^{-1} B^T P) x$$

$RR^{-1} = I$, therefore:

$$u^T R u + u^T B^T P x + x^T P B u + x^T P B R^{-1} B^T P x - x^T (P B R^{-1} B^T P) x$$

Which is the same as the Equation (8):

$$(u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x) - x^T (P B R^{-1} B^T P) x$$

Therefore, Equations (8) and (9) are the same, as below:

$$\begin{aligned}(u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x) - x^T (P B R^{-1} B^T P) x \\ = u^T R u + x^T P B u + u^T B^T P x + [x^T (P B R^{-1} B^T P) x - x^T (P B R^{-1} B^T P) x]\end{aligned}$$

We can now use Equation (9) instead of Equation (8) in Equation (7)

$$\begin{aligned}J = \int_0^\infty [x^T (A^T P + P A + Q) x + (u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x) \\ - x^T (P B R^{-1} B^T P) x] \, dt\end{aligned} \quad (10)$$

Move the term $PBR^{-1}B^TP$ into the first term:

$$J = \int_0^\infty [x^T (A^T P + P A + Q - P B R^{-1} B^T P) x + (u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x)] \, dt$$

In order to minimize the cost function J, we need to minimize both terms in the above equation.

The first term is:

$$A^T P + P A + Q - P B R^{-1} B^T P = 0$$

This equation is called the Algebraic Riccati Equation (ARE). The ARE can be solved numerically. The goal is to find P that solves the ARE.

The second term in J is:

$$(u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x) = 0$$

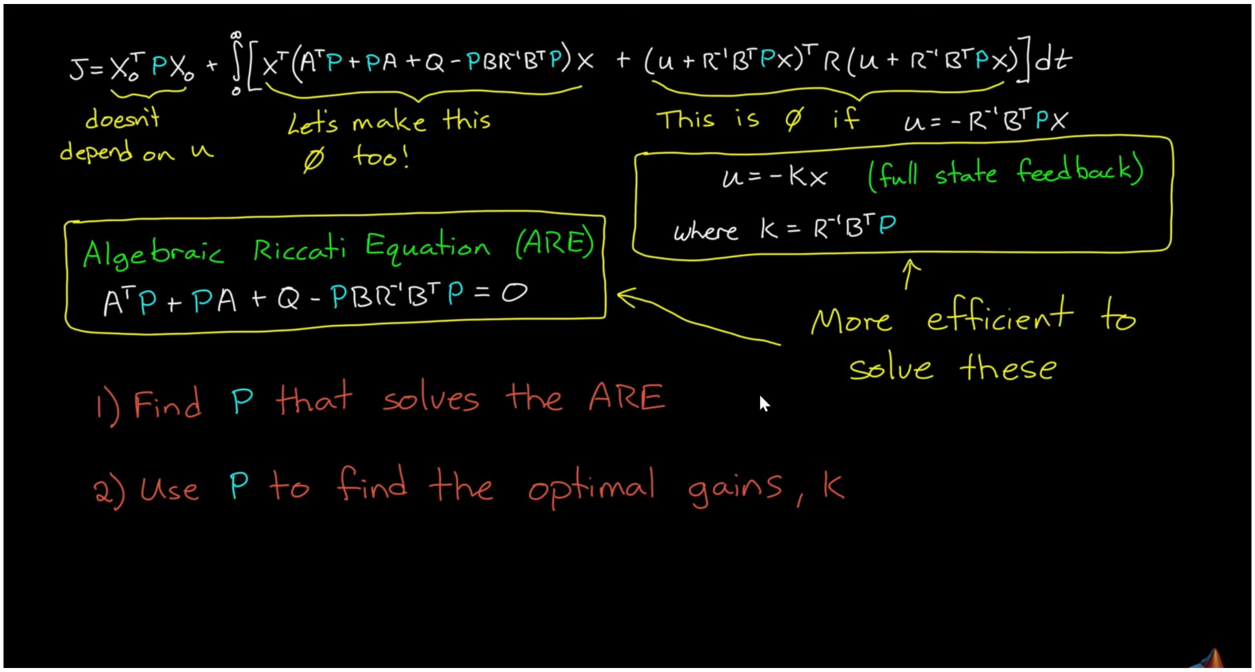
The input effort u is a function of the controller gain k as below:

$$u = -kx$$

If $k$is chosen as $k = R^{-1} B^T P$ then $u + R^{-1} B^T P x = 0$.

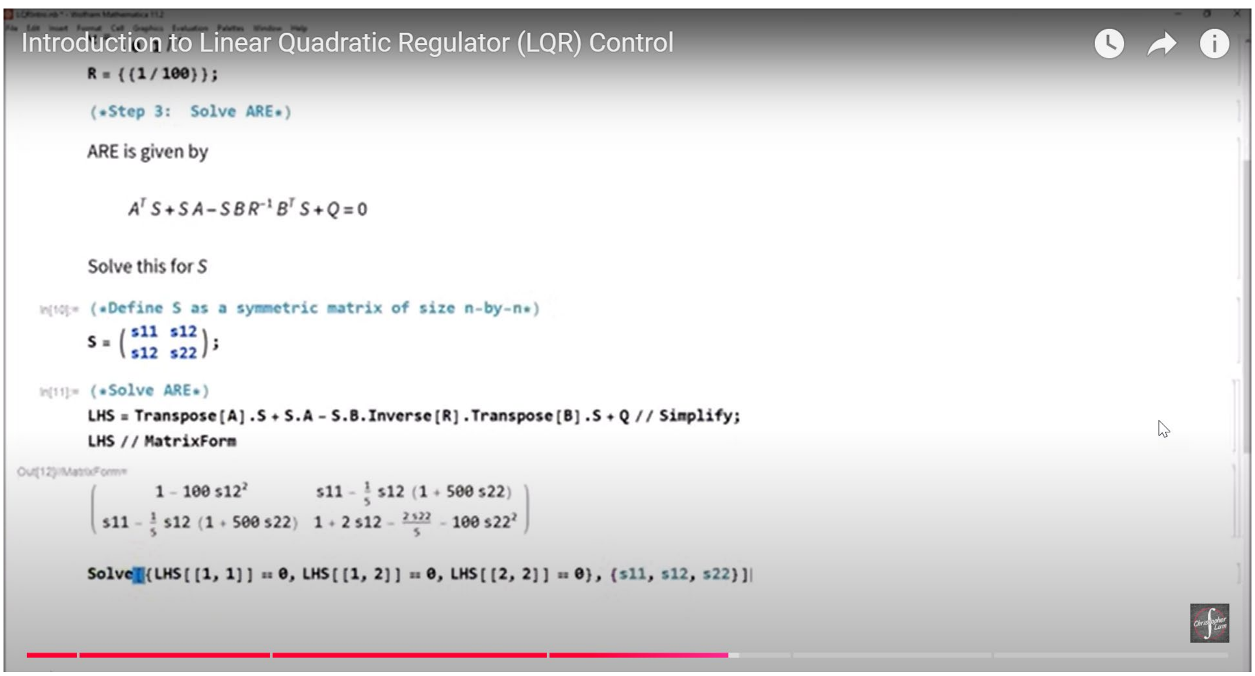
Therefore, P is obtained from ARE, and then substituted in $u = -kx$, where $k = R^{-1} B^T P$.

More information about the Algebraic Riccati Equation (ARE):



Why the Riccati Equation Is important for LQR Control

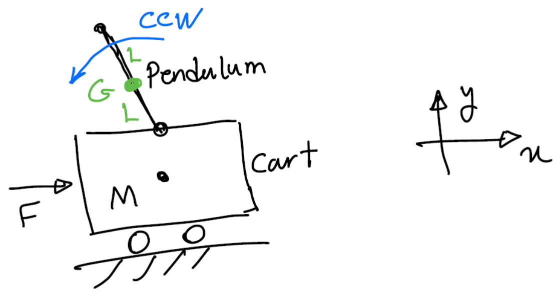
https://www.youtube.com/watch?v=ZktL3YjTbB4



Introduction to Linear Quadratic Regulator (LQR) Control

https://www.youtube.com/watch?v=wEevt2a4SKI

Example: Solve the inverted pendulum control problem with LQR.



The equations of motion of the inverted pendulum are given as follows (for the derivation of the equations, please see the equations of motion notes posted on canvas).

$$(I + ml^2)\ddot{\emptyset} - mgl\emptyset = ml\ddot{x} \tag{1}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\emptyset} = F \tag{2}$$

Parameters are given as:

M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;

The state-space model will be given by:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-(I+ml^2)b}{Mml^2 + I(M+m)} & \dfrac{g(ml)^2}{Mml^2 + I(M+m)} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-mlb}{Mml^2 + I(M+m)} & \dfrac{mgl(M+m)}{Mml^2 + I(M+m)} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{(I+ml^2)}{Mml^2 + I(M+m)} \\ 0 \\ \dfrac{ml}{Mml^2 + I(M+m)} \end{bmatrix} F$$

```
p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices


A = [0     1          0        0;
   0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
   0     0          0        1;
   0 -(m*l*b)/p      m*g*l*(M+m)/p  0];
B = [    0;
   (I+m*l^2)/p;
       0;
     m*l/p];
C = [1 0 0 0;
```

```
    0 0 1 0];
D = [0;
    0];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x'; 'phi'};

sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs)

sys_ss =

  A =
                x    x_dot      phi  phi_dot
   x            0        1        0        0
   x_dot        0  -0.1818    2.673        0
   phi          0        0        0        1
   phi_dot      0  -0.4545    31.18        0

  B =
              u
   x          0
   x_dot  1.818
   phi        0
   phi_dot  4.545

  C =
            x    x_dot      phi  phi_dot
   x        1        0        0        0
   phi      0        0        1        0

  D =
       u
```

x 0

phi 0

Continuous-time state-space model.

Reference:

https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

Solving the Algebraic Riccati Equation (ARE):

R is mxm, where ‘m’ is the number of the inputs

Q is nxn, where ‘n’ is the number of the state variables

```
R=1;
%Q=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
Q=[1 0 0 0; 0 0 0 0; 0 0 1 0; 0 0 0 0];
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices
>> A = [0     1              0           0;
0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
0     0              0           1;
0 -(m*l*b)/p       m*g*l*(M+m)/p  0];
B = [     0;
(I+m*l^2)/p;
0;
m*l/p];
C = [1 0 0 0;
0 0 1 0];
D = [0;
```

0];

Using MATLAB lqr command in solving the ARE:

```
>> [K,S,P] = lqr(A,B,Q,R)

K =

  -1.0000  -1.6567  18.6854   3.4594


S =

   1.5567   1.2067  -3.4594  -0.7027
   1.2067   1.4554  -4.6827  -0.9467
  -3.4594  -4.6827  31.6320   5.9839
  -0.7027  -0.9467   5.9839   1.1397


P =

 -0.8494 + 0.8323i
 -0.8494 - 0.8323i
 -5.5978 + 0.4070i
 -5.5978 - 0.4070i
```

Solving the Riccati equation (*without* using the MATLAB lqr command):

Derivation of the Algebraic Riccati Equation (ARE) is given in the above sections in this tutorial:

```
>> R=1;
%Q=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
```

Q=[1 0 0 0; 0 0 0 0; 0 0 1 0; 0 0 0 0]; %this Q is the same as the one used in the Michigan Control Tutorial

M = 0.5;

m = 0.2;

b = 0.1;

I = 0.006;

g = 9.8;

l = 0.3;

p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices

A = [0   1   0   0;

0 -(I+m*l^2)*b/p (m^2*g*l^2)/p   0;

0   0   0   1;

0 -(m*l*b)/p   m*g*l*(M+m)/p 0];

B = [0;

(I+m*l^2)/p;

0;

m*l/p];

C = [1 0 0 0;

0 0 1 0];

D = [0;

0];

Develop a symmetric P matrix as the solution of the ARE:

syms P11 P12 P13 P14 P21 P22 P23 P24 P31 P32 P33 P34 P41 P42 P43 P44

P=[P11 P12 P13 P14; P12 P22 P23 P24; P13 P23 P33 P34; P14 P24 P34 P44]

The Algebraic Riccati Equation (ARE):

Riccati = A'*P+P*A+Q-P*B*inv(R)* B'* P

or

Riccati =transpose(A)*P+P*A+Q-P*B*inv(R)* transpose(B)* P

MATLAB result for the ARE as a function of P:

Riccati =

[1 - P14*((1000*P12)/121 + (2500*P14)/121) - P12*((400*P12)/121 + (1000*P14)/121), P11 - (2*P12)/11 - (5*P14)/11 - P22*((400*P12)/121 + (1000*P14)/121) - P24*((1000*P12)/121 + (2500*P14)/121), (147*P12)/55 + (343*P14)/11 - P23*((400*P12)/121 + (1000*P14)/121) - P34*((1000*P12)/121 + (2500*P14)/121), P13 - P24*((400*P12)/121 + (1000*P14)/121) - P44*((1000*P12)/121 + (2500*P14)/121)]

[P11 - (2*P12)/11 - (5*P14)/11 - P12*((400*P22)/121 + (1000*P24)/121) - P14*((1000*P22)/121 + (2500*P24)/121), 2*P12 - (4*P22)/11 - (10*P24)/11 - P22*((400*P22)/121 + (1000*P24)/121) - P24*((1000*P22)/121 + (2500*P24)/121), P13 + (147*P22)/55 - (2*P23)/11 + (343*P24)/11 - (5*P34)/11 - P23*((400*P22)/121 + (1000*P24)/121) - P34*((1000*P22)/121 + (2500*P24)/121), P14 + P23 - (2*P24)/11 - (5*P44)/11 - P24*((400*P22)/121 + (1000*P24)/121) - P44*((1000*P22)/121 + (2500*P24)/121)]

[(147*P12)/55 + (343*P14)/11 - P12*((400*P23)/121 + (1000*P34)/121) - P14*((1000*P23)/121 + (2500*P34)/121), P13 + (147*P22)/55 - (2*P23)/11 + (343*P24)/11 - (5*P34)/11 - P22*((400*P23)/121 + (1000*P34)/121) - P24*((1000*P23)/121 + (2500*P34)/121), (294*P23)/55 + (686*P34)/11 - P23*((400*P23)/121 + (1000*P34)/121) - P34*((1000*P23)/121 + (2500*P34)/121) + 1, (147*P24)/55 + P33 + (343*P44)/11 - P24*((400*P23)/121 + (1000*P34)/121) - P44*((1000*P23)/121 + (2500*P34)/121)]

[P13 - P12*((400*P24)/121 + (1000*P44)/121) - P14*((1000*P24)/121 + (2500*P44)/121), P14 + P23 - (2*P24)/11 - (5*P44)/11 - P22*((400*P24)/121 + (1000*P44)/121) - P24*((1000*P24)/121 + (2500*P44)/121), (147*P24)/55 + P33 + (343*P44)/11 - P23*((400*P24)/121 + (1000*P44)/121) - P34*((1000*P24)/121 + (2500*P44)/121), 2*P34 - P24*((400*P24)/121 + (1000*P44)/121) - P44*((1000*P24)/121 + (2500*P44)/121)]

subs(Riccati, {P11 P12 P13 P14 P22 P23 P24 P33 P34 P44},{1, 2, 3, 4, 5, 6, 7, 8, 9, 10})

ans =

[-57479/121, -108143/121, -605306/605, -153237/121]

[-108143/121, -203006/121, -228736/121, -287494/121]

[-605306/605, -228736/121, -1264921/605, -1619191/605]

[-153237/121, -287494/121, -1619191/605, -407422/121]

double(ans)

ans =

1.0e+03 *

-0.4750 -0.8937 -1.0005 -1.2664

-0.8937	-1.6777	-1.8904	-2.3760
-1.0005	-1.8904	-2.0908	-2.6763
-1.2664	-2.3760	-2.6763	-3.3671

tf = issymmetric(ans)

tf =

logical

1

Convert P to X in other to solve the equations as a function of X(i) using the MATLAB fsolve command using the following mapping (Also replace , with ;)

x(1) =P11

x(2) =P12

x(3) =P13

x(4) =P14

x(5) = P22

x(6) =P23

x(7) =P24

x(8) =P33

x(9) = P34

x(10) = P44

Riccati =

[1 - X(4)*((1000*X(2))/121 + (2500*X(4))/121) - X(2)*((400*X(2))/121 + (1000*X(4))/121); X(1) - (2*X(2))/11 - (5*X(4))/11 - X(5)*((400*X(2))/121 + (1000*X(4))/121) - X(7)*((1000*X(2))/121 + (2500*X(4))/121); (147*X(2))/55 + (343*X(4))/11 - X(6)*((400*X(2))/121 + (1000*X(4))/121) - X(9)*((1000*X(2))/121 + (2500*X(4))/121); X(3) - X(7)*((400*X(2))/121 + (1000*X(4))/121) - X(10)*((1000*X(2))/121 + (2500*X(4))/121)]

[X(1) - (2*X(2))/11 - (5*X(4))/11 - X(2)*((400*X(5))/121 + (1000*X(7))/121) - X(4)*((1000*X(5))/121 + (2500*X(7))/121); 2*X(2) - (4*X(5))/11 - (10*X(7))/11 - X(5)*((400*X(5))/121 + (1000*X(7))/121) - X(7)*((1000*X(5))/121 + (2500*X(7))/121); X(3) + (147*X(5))/55 - (2*X(6))/11 + (343*X(7))/11 - (5*X(9))/11 - X(6)*((400*X(5))/121 + (1000*X(7))/121) - X(9)*((1000*X(5))/121 + (2500*X(7))/121); X(4) + X(6) - (2*X(7))/11 - (5*X(10))/11 - X(7)*((400*X(5))/121 + (1000*X(7))/121) - X(10)*((1000*X(5))/121 + (2500*X(7))/121)]

[(147*X(2))/55 + (343*X(4))/11 - X(2)*((400*X(6))/121 + (1000*X(9))/121) - X(4)*((1000*X(6))/121 + (2500*X(9))/121); X(3) + (147*X(5))/55 - (2*X(6))/11 + (343*X(7))/11 - (5*X(9))/11 - X(5)*((400*X(6))/121 + (1000*X(9))/121) - X(7)*((1000*X(6))/121 + (2500*X(9))/121); (294*X(6))/55 + (686*X(9))/11 - X(6)*((400*X(6))/121 + (1000*X(9))/121) - X(9)*((1000*X(6))/121 + (2500*X(9))/121) + 1; (147*X(7))/55 + X(8) + (343*X(10))/11 - X(7)*((400*X(6))/121 + (1000*X(9))/121) - X(10)*((1000*X(6))/121 + (2500*X(9))/121)]

[X(3) - X(2)*((400*X(7))/121 + (1000*X(10))/121) - X(4)*((1000*X(7))/121 + (2500*X(10))/121); X(4) + X(6) - (2*X(7))/11 - (5*X(10))/11 - X(5)*((400*X(7))/121 + (1000*X(10))/121) - X(7)*((1000*X(7))/121 + (2500*X(10))/121); (147*X(7))/55 + X(8) + (343*X(10))/11 - X(6)*((400*X(7))/121 + (1000*X(10))/121) - X(9)*((1000*X(7))/121 + (2500*X(10))/121); 2*X(9) - X(7)*((400*X(7))/121 + (1000*X(10))/121) - X(10)*((1000*X(7))/121 + (2500*X(10))/121)]

This matrix is symmetric. Therefore, leave only the upper triangular matrix (the lower triangular matrix gives the same repeating equations because the matrix is symmetric):

UpperTriangularRiccati =

[1 - X(4)*((1000*X(2))/121 + (2500*X(4))/121) - X(2)*((400*X(2))/121 + (1000*X(4))/121);
X(1) - (2*X(2))/11 - (5*X(4))/11 - X(5)*((400*X(2))/121 + (1000*X(4))/121) - X(7)*((1000*X(2))/121 + (2500*X(4))/121);

(147*X(2))/55 + (343*X(4))/11 - X(6)*((400*X(2))/121 + (1000*X(4))/121) - X(9)*((1000*X(2))/121 + (2500*X(4))/121);

X(3) - X(7)*((400*X(2))/121 + (1000*X(4))/121) - X(10)*((1000*X(2))/121 + (2500*X(4))/121);

2*X(2) - (4*X(5))/11 - (10*X(7))/11 - X(5)*((400*X(5))/121 + (1000*X(7))/121) - X(7)*((1000*X(5))/121 + (2500*X(7))/121);

X(3) + (147*X(5))/55 - (2*X(6))/11 + (343*X(7))/11 - (5*X(9))/11 - X(6)*((400*X(5))/121 + (1000*X(7))/121) - X(9)*((1000*X(5))/121 + (2500*X(7))/121);

X(4) + X(6) - (2*X(7))/11 - (5*X(10))/11 - X(7)*((400*X(5))/121 + (1000*X(7))/121) - X(10)*((1000*X(5))/121 + (2500*X(7))/121);

(294*X(6))/55 + (686*X(9))/11 - X(6)*((400*X(6))/121 + (1000*X(9))/121) - X(9)*((1000*X(6))/121 + (2500*X(9))/121) + 1;

(147*X(7))/55 + X(8) + (343*X(10))/11 - X(7)*((400*X(6))/121 + (1000*X(9))/121) - X(10)*((1000*X(6))/121 + (2500*X(9))/121);

2*X(9) - X(7)*((400*X(7))/121 + (1000*X(10))/121) - X(10)*((1000*X(7))/121 + (2500*X(10))/121)]

Solving the ARE (the same UpperTriangularRiccati as above):

```
F = @(X)  [ 1 - X(4)*((1000*X(2))/121 + (2500*X(4))/121) - X(2)*((400*X(2))/121 + (1000*X(4))/121);

 X(1) - (2*X(2))/11 - (5*X(4))/11 - X(5)*((400*X(2))/121 + (1000*X(4))/121) - X(7)*((1000*X(2))/121 +
(2500*X(4))/121);

(147*X(2))/55 + (343*X(4))/11 - X(6)*((400*X(2))/121 + (1000*X(4))/121) - X(9)*((1000*X(2))/121 +
(2500*X(4))/121);

X(3) - X(7)*((400*X(2))/121 + (1000*X(4))/121) - X(10)*((1000*X(2))/121 + (2500*X(4))/121);

2*X(2) - (4*X(5))/11 - (10*X(7))/11 - X(5)*((400*X(5))/121 + (1000*X(7))/121) - X(7)*((1000*X(5))/121 +
(2500*X(7))/121);

X(3) + (147*X(5))/55 - (2*X(6))/11 + (343*X(7))/11 - (5*X(9))/11 - X(6)*((400*X(5))/121 +
(1000*X(7))/121) - X(9)*((1000*X(5))/121 + (2500*X(7))/121);

X(4) + X(6) - (2*X(7))/11 - (5*X(10))/11 - X(7)*((400*X(5))/121 + (1000*X(7))/121) -
X(10)*((1000*X(5))/121 + (2500*X(7))/121);

(294*X(6))/55 + (686*X(9))/11 - X(6)*((400*X(6))/121 + (1000*X(9))/121) - X(9)*((1000*X(6))/121 +
(2500*X(9))/121) + 1;

(147*X(7))/55 + X(8) + (343*X(10))/11 - X(7)*((400*X(6))/121 + (1000*X(9))/121) -
X(10)*((1000*X(6))/121 + (2500*X(9))/121);

2*X(9) - X(7)*((400*X(7))/121 + (1000*X(10))/121) - X(10)*((1000*X(7))/121 + (2500*X(10))/121)]



>> x0 = [0;0;0;0;0;0;0;0;0;0];

>> [X,fval] = fsolve(F,x0)
```

```
Solver stopped prematurely.

fsolve stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 1.000000e+05.

X =

   1.2050
   0.7185
```

```
  -0.1117
  -0.0672
   0.7830
  -0.0734
  -0.0703
   0.0661
  -0.0092
   0.0045


fval =

  -0.0016
   0.0001
  -0.0001
  -0.0040
  -0.0023
  -0.0011
  -0.0111
   0.0009
  -0.0015
  -0.0300
```

```
options=optimset('MaxFunEvals',100000);
options=optimset(options,'MaxIter',100000);
>> x0 = [0;0;0;0;0;0;0;0;0;0];
>> [X,fval] = fsolve(F,x0,options)
```

No solution found.

fsolve stopped because the last step was ineffective. However, the vector of function values is not near zero, as measured by the value of the function tolerance.

<stopping criteria details>

Equation solved, inaccuracy possible.

The vector of function values is near zero, as measured by the value

of the function tolerance. However, the last step was ineffective.

<stopping criteria details>

If we choose the guessed initial values very close to the solution for x0, the solution will be found by MATALB. Otherwise, matlab can not solve the problem.

Here, x0 is chosen as the exact solution that we already found from the MATLAB lqr command above. This is just a practice to show that if the initial guess of x0 is chosen as the solution, the solution of the nonlinear system of 10 equations above give the same result as the lqr just as a confirmation:

```
>> x0 = [1.5567 1.2067 -3.4594 -0.7027 1.4554 -4.6827 -0.9467 31.6320 5.9839 1.1397];
>> [X,fval] = fsolve(F,x0,options)
```

Equation solved.

fsolve completed because the vector of function values is near zero

as measured by the value of the function tolerance, and

the problem appears regular as measured by the gradient.

<stopping criteria details>

X =

1.5567 1.2067 -3.4594 -0.7027 1.4554 -4.6827 -0.9467 31.6320 5.9839 1.1397

Therefore, P is obtained from ARE, and then substituted in $u = -kx$, where $k = R^{-1}B^TP$.

```
R=1;
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices
A = [0     1              0           0;
0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
0     0              0           1;
0 -(m*l*b)/p       m*g*l*(M+m)/p  0];
B = [     0;
(I+m*l^2)/p;
0;
m*l/p];
C = [1 0 0 0;
0 0 1 0];
D = [0;
0];
```

We have this from lqr:

```
S = [1.5567    1.2067   -3.4594   -0.7027;
    1.2067    1.4554   -4.6827   -0.9467;
   -3.4594   -4.6827   31.6320    5.9839;
   -0.7027   -0.9467    5.9839    1.1397]
```

```
P=S;
k=R^(-1)*transpose(B)*P
k =

   -1.0001   -1.6570   18.6855    3.4592
```

This is the same result as the result from the MATLAB lqr command shown above.

If we need to show the coefficients in decimal (up to 3 decimal points) rather than fraction:

vpa(sRiccati,3)

If we need to simplify the equation:

sRiccati= simplify(Riccati)

Creating vectors and matrices:

```
S = sym('S%d%d', [4 4])

S =

[S11, S12, S13, S14]
[S21, S22, S23, S24]
[S31, S32, S33, S34]
[S41, S42, S43, S44]

V = sym('V%d', [10 1])

V =

 V1
 V2
 V3
 V4
 V5
 V6
 V7
 V8
 V9
V10
```