Pole-Placement Method

Pole placement method is one of the classic control theories and has an advantage in system control for desired performance. Theoretically pole placement is to set the desired pole location and to move the pole location of the system to that desired pole location to get the desired system response. Mathematically once the system transfer function is defined, the desired transfer function should be also defined, then each coefficient in the same order in polynomial is compared to be the same. This pole placement control method results the desired system response and is easy to fine the gain mathematically but the accuracy of system transfer function is significantly important and is expensive to implement in the high order system.

LQR

Linear Quadratic Regulator (LQR) is the optimal theory of pole placement method. LQR algorithm defines the optimal pole location based on two cost function.

To find the optimal gains, one should define the optimal performance index firstly and then solve algebraic Riccati equation. LQR does not have any specific solution to define the cost function to obtain the optimal gains and the cost function should be defined in iterative manner.

There are lots of control theory related to LQR so the reader should study for more information. Basically the reader should study state-space representation, state feedback control, performance index, and Riccati equation. These topics will not be dealt with since it exceeds the scope of purpose of our course work.

Next we will see what is pole placement, how to implement it in MATLAB, and how to fine the optimal gains. Finally we will compare the system response in pole placement and LQR.
Pole-Placement Method

For explanation of pole placement method we assume a system which is pendulum. The pendulum system is the second order system and is defined as,

```matlab
clc
clear all
close all

num = 1;
den = [1 0.05 10];
sys = tf(num,den)
figure(1)
step(sys)
grid on
legend('pendulum')
hold off

% In step response, the response tells that when the external force acts on
% the pendulum, the pendulum swings freely a lot because of a small damping
% ratio. Now we want to control this pendulum system not to have
% oscillation with no steady state error.
% This is the second order system so we can choose two desired pole
% location to move the original poles to there. The second order system is
% very well analyzed in many text books. In this case we will focus on
% removing oscillation caused by low damping.
% In s domain, the damping ratio, zeta, is defined as the distance from the
% imaginary axis. The original poles are located at
roots(den)

% The complex ploes are placed at 0.025 from the imaginary axis. Now we
% want to move these poles to -2 from the axis so the desired pole location
% will be -2 + 3.1622i and -2 - 3.1622i. To move to the desired pole
% location we have to fine feedback gains. Mathematically this computation
% is very simple but MATLAB supports simple function for this called
% 'place'. (refer to 'place' in MATLAB help

% Our desired pole location is.
desiredpole = [-2+3.1622i -2-3.1622i];

% Before applying 'place' we need to convert our transfer function to state
```
% space representation. To do that,
[A,B,C,D] = tf2ss(num,den)

% Now we apply 'place' function to get gain K
K = place(A,B,desiredpole)

% K is the state feedback gain, in other word, to move original poles to
% the desired pole location we have to have feedback loop with K.
% The closed loop state space representation with gain K will be,
Anew = A-B*K
Bnew = B;
Cnew = C;
Dnew = D;

% Thus our closed loop system is,
[numnew, dennew] = ss2tf(Anew,Bnew,Cnew,Dnew);
sysnew = tf(numnew,dennew)
t = 0:0.01:10;
figure(2)
step(sys,t)
hold on
step(sysnew,t)
grid on
legend( 'ori sys' , 'new sys' )
hold off

% There is still steady state error, and to compensate we should find
% reference gain. But we will not cover this in our course.
% (Can be added)

Transfer function:

\[ \frac{1}{s^2 + 0.05 s + 10} \]

ans =

\[-0.0250 + 3.1622i\]
\[-0.0250 - 3.1622i\]

A =

\[\begin{bmatrix}
-0.0500 & -10.0000 \\
1.0000 & 0
\end{bmatrix}\]
B =
  1
  0

C =
  0  1

D =
  0

K =
  3.9500  3.9995

A_{\text{new}} =
  -4.0000  -13.9995
  1.0000     0

Transfer function:
4.441e-016 s + 1
-----------------
s^2 + 4 s + 14
Linear Quadratic Regulator

Now we will find the gain based on optimal pole location which is determined by two cost function. LQR is also very well known controller so refer to any control text book for more information about weight function.

We will use 'lqr' supported by MATLAB to find the optimal gains for the system. Our goal is to minimize the performance index and the solution of algebraic Riccati equation is the method to do. We can start with,

```matlab
w = 10;
Q = w*C'*C;
R = 0.1;
K = lqr(A,B,Q,R)

% Our new system will be,
Alqr = A-B*K
Blqr = B;
Clqr = C;
Dlqr = D;

[numlqr, denlqr] = ss2tf(Alqr,Blqr,Clqr,Dlqr);
```
syslqr = tf(numlqr,denlqr)
t = 0:0.01:10;
figure(3)
step(sys,t)
hold on
step(sysnew,t)
hold on
step(syslqr,t)
grid on
legend( 'ori sys', 'new sys', 'lqr sys')
hold off

% You need to change the w and R value for obtain different system
% response. The definition of Q and R is defined in many text book.
% Based on Q and R, you can get the system response as same as original
% system or better response than pole placement method.

K =
   2.8287    4.1421

Alqr =
   -2.8787  -14.1421
      1.0000         0

Transfer function:
    4.441e-016 s + 1
-------------------------
s^2 + 2.879 s + 14.14