OPTIMAL CONTROL THEORY

Final Project

Hakan AŞIK
518182008

Time Domain criteria selection

Time Domain Criterias are selected as(ξ=0.707 and wn=8.88 rad/s);
% OS= 5
Tp=0.5 s
Ts=0.68

Thus desired characteristic for 2nd order system:

$$s^2 + (2\xi w_n)s + w_n{}^2 = 0$$

Infinite LQR design

Consider an LTI system

$$\dot{x}(t) = Ax(t) + Bu(t), \quad t \geq 0, \quad x(0) = 0,$$

$$y(t) = Cx(t) + Du(t), \quad t \geq 0,$$

where *A* 0 *Rn* H *n*, *B* 0 *Rn* H*m*, *C* 0 *Rp* H *n*, *D* 0 *Rp* H*m*, are system matrix, input matrix, output matrix and feed forward matrix, respectively. *X* is the state vector, *u* is the control input vector, and *y* is the output vector. The conventional LQR problem is to obtain the control input *u** which minimizes the following cost function.

$$J(u) = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt,$$

where *Q* = *QT* is a positive semidefinite matrix that penalizes the departure of system states from the equilibrium, and *R* = *RT* is a positive definite matrix that penalizes the control input [19]. The solution of the LQR problem, the optimal control gain *K*, can be determined via the following Lagrange multiplier based optimization technique.

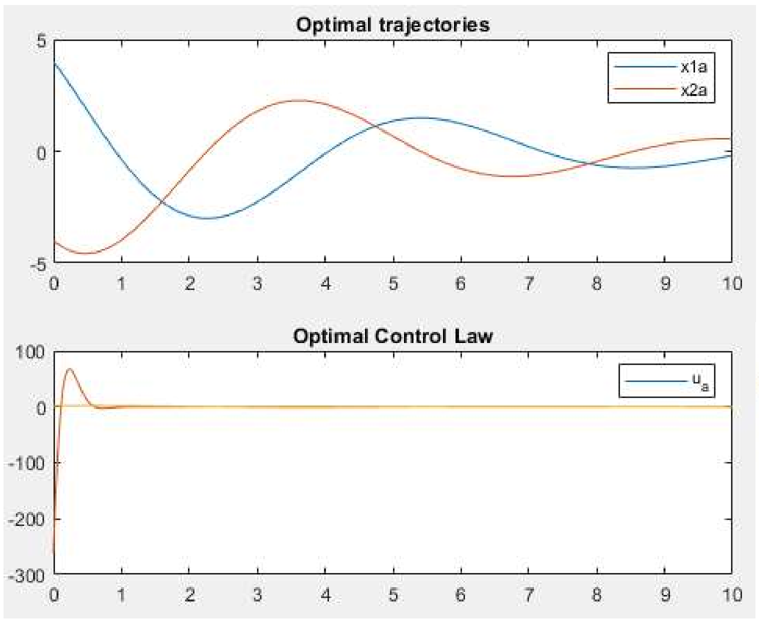
$$K = R^{-1}B^T P.$$

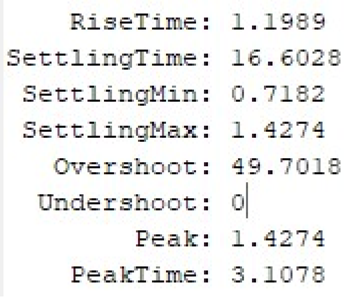
The optimal state feedback control gain matrix *K* of LQR can be found by solving the following ARE.

$$A^T P + PA + Q - PBR^{-1}B^T P = 0.$$

where *P* is the solution of *ARE*. The elements of weighting matrices *Q* and *R* are important components of an LQR optimization process.

a) Q=eye(2) and R=1





```
      RiseTime: 1.1989
  SettlingTime: 16.6028
   SettlingMin: 0.7182
   SettlingMax: 1.4274
     Overshoot: 49.7018
    Undershoot: 0
          Peak: 1.4274
      PeakTime: 3.1078
```

b) By solving the ARE, the transformation matrix (P) between states and co-states can be obtained.

One of the essential features of LQR is that Q should be a symmetric positive semidefinite matrix and R should be a positive definite matrix. So, the weighting matrices Q and R, and the solution of ARE are chosen as

$$Q = \begin{bmatrix} q1 & 0 \\ 0 & q2 \end{bmatrix}, \text{ R=r and P=}\begin{bmatrix} p11 & p12 \\ p21 & p22 \end{bmatrix},$$

Using the Lagrange optimization techniques, the state feedback gain matrix K can be calculated as

$$K = R^{-1} * B^T * P = \frac{1}{r}\begin{bmatrix} 0 & 1 \end{bmatrix}\begin{bmatrix} p11 & p12 \\ p21 & p22 \end{bmatrix}$$

$$= \frac{1}{r}\begin{bmatrix} p21 & p22 \end{bmatrix}$$

Putting Q,R and P into ARE gives,

$$\begin{bmatrix} q1 & 0 \\ 0 & q2 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} p12^2 + 2 * p21 & p12 * p22 + p22 - p11 \\ p22 * p21 - p11 + p22 & p22^2 - 2 * p21 \end{bmatrix}$$

Here it can be found that $q1 = \frac{p1\ ^2+2*p21}{r}$ and $q2 = \frac{p22^2-2*p21}{r}$ and $p11 = \frac{p12*p22+p22-p1}{r}$.

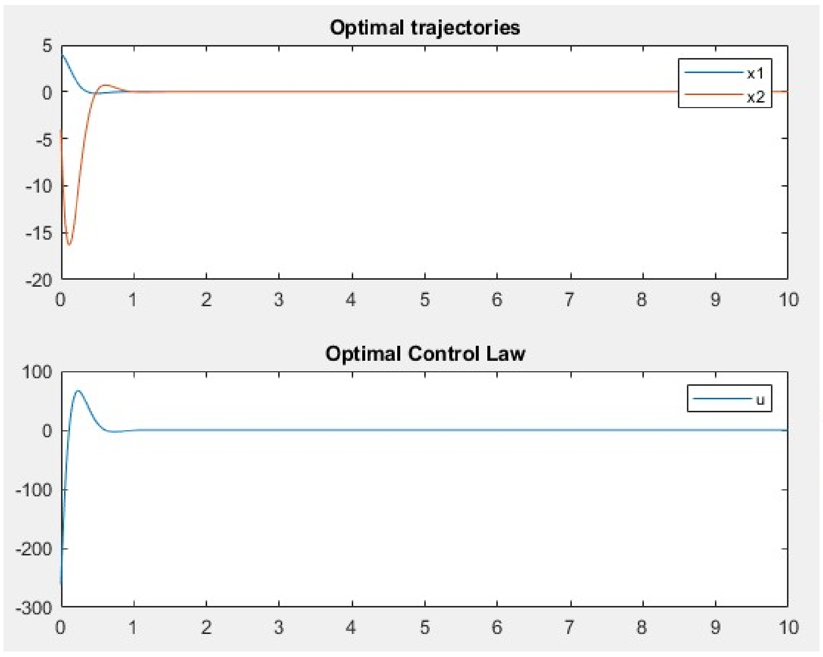
The closed loop state equation of the system can be written as

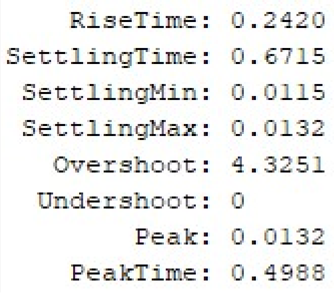
$$\dot{x}(t) = [A - BK]\,x(t) = [A - BR^{-1}B^T P]\,x(t).$$

According to direct substitution method for the pole placement controller design, the actual characteristic equation of the system is compared with the desired characteristic equation to obtain the expressions for p13, p23 and p33 in terms of state model and design specifications in time domain. Therefore, the actual characteristic equation of the system can be represented as

$det|\lambda I - (A - BK)| = 0$ yields → r*s^2 + (p22*s) + p12 + 100

By comparing actual and desired characteristic equation, p21 and p22 can be found and q1,q2 and p11 can be computed. Thus, necessary Q and P matrices along with R designated to be Positive definite matrix are developed to design LQR satisfying time domain criteria.





```
     RiseTime: 0.2420
 SettlingTime: 0.6715
  SettlingMin: 0.0115
  SettlingMax: 0.0132
    Overshoot: 4.3251
   Undershoot: 0
         Peak: 0.0132
     PeakTime: 0.4988
```

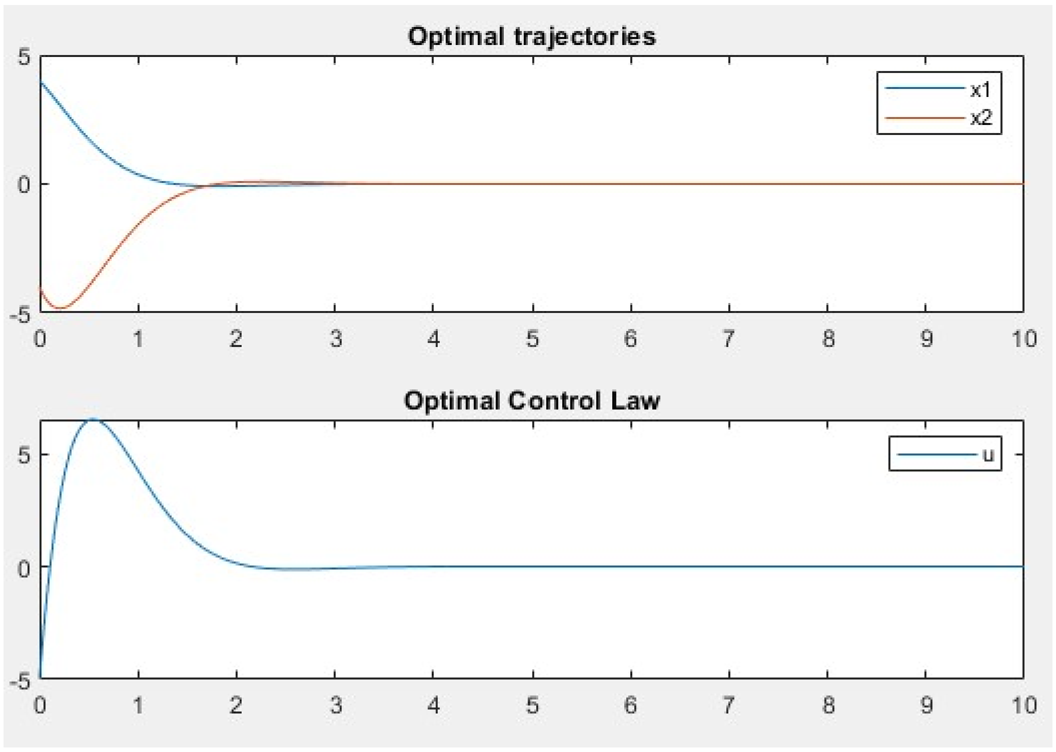
Genetic Algorithm

In this project, Genetic algorithm is used as an evolutionary search algorithm in order to find global minimum of cost function.

Performance index (i.e cost function here) is defined by using certain time domain criteria, namely:

$$\text{Cost function}=t_{rise}^{max} * t_{rise}^{min} * u^2 + t_{settling}^{max} * t_{settling}^{min} * u^2 + t_{peak}^{max} * t_{peak}^{min} * u^2$$

100 populations are raced against each other selecting top 10 elite population for the next generations. Selected populations are mutated and rest of the populations are crossed over in order to improve the results.



As such, small overshoot and smooth curve has been obtained despite deterioration of rise and settling time. Cost function may be changed along with increasing population and generation number.

Time domain criteria code:

```
clc
clear all

%System Matrices
A=[0 1;-1 0];
B=[0;1];

%symbolic variables
syms p11 p12 p22 s q1 q2
P=[p11 p12;p12 p22]

%Time domain parameters:
zeta=0.707; % damping ratio
omegaN=8.88 % rad/s
desiredEq= s^2+2*zeta*omegaN*s+omegaN^2
Coeffdesired= coeffs(desiredEq)

%Weight matrices
Q=[q1 0;0 q2];
r=100;
R=r;

chareq=det(eye(2)*s-A+B*inv(R)*B'*P)
CoeffActual= coeffs(chareq,s) %to separate symbolic equation coefficients

Pmatrix=solve(CoeffActual==Coeffdesired)
p12=double(Pmatrix.p12)
p22=double(Pmatrix.p22)
dP=ARE(P,A,B,R,Q)

p12=p12; p22=p22; Pmat=subs(dP)
q1=solve(Pmat(1,1)==0); q2=solve(Pmat(2,2)==0) ;p11=solve(Pmat(1,2)==0)
P_calculated=subs(P) %Symbolic substitution
Q_calculated=subs(Q)
%Check matrix Positive definiteness
% if p=0 --> Positive definite otherwise Not
[~,p] = chol(P_calculated)
[~,q] = chol(Q_calculated)

%% a)
Q_eye=eye(2); R_eye=10
[K_eye,P_lqreye,E_eye]=lqr(A,B,Q_eye,R_eye)
sys=ss(A-B*K_eye,eye(2),eye(2),zeros(2,2));

t=0:0.01:10;
x_a=initial(sys,[4;-4],t);

x1a=[1 0]*x_a';
x2a=[0 1]*x_a';
u_a=-(K_eye*x_a');

plot1=subplot(2,1,1);
plot(t,x1a,t,x2a)
legend(plot1,{'x1a','x2a'})
title('Optimal trajectories')
```

```matlab
plot2=subplot(2,1,2);
plot(t,u_a)
legend(plot2,{'u_a'})
title('Optimal Control Law')

hold on

timecriteria_a=stepinfo(sys);
timecriteria_a(1,2)

%% b)
[K,P_lqr,E]=lqr(A,B,double(Q_calculated),R)

B1=[0;1];
sys1=ss(A-B*K,B1,eye(2),zeros(2,1));

x=initial(sys1,[4;-4],t);

x1=[1 0]*x';
x2=[0 1]*x';
u=-(K*x');

figure
plot1=subplot(2,1,1);
plot(t,x1,t,x2)
legend(plot1,{'x1','x2'})
title('Optimal trajectories')

plot2=subplot(2,1,2);
plot(t,u)
legend(plot2,{'u'})
title('Optimal Control Law')

timecriteria_b=stepinfo(sys1)
```

<u>Genetic Algorithm code:</u>

Main.m

```
clear all
clc
A=[0 1;-1 0];
B=[0;1];
C=[1 0;0 1];
D=zeros(2,1);

%h=ss(A,B,C,D)
%step(h)

%number of states
n=2;
%number of inputs
p=1;
%number of outputs
q=2;

%Q=[q11 0 0 0; 0 q22 0 0]
%R=[r11]
%indiv=[q11;q22;r11]

%number of individuals
nind=100;
%number of chromosomes
nc=3;

ngt=100;
Jemin=2.38*10;
[K,P,Jt]=genetic_algor(A,B,C,D,ngt,nind,nc,Jemin);

B1=[0;1];
sys1=ss(A-B*K,B1,C,D);

t=0:0.01:10;
x=initial(sys1,[4;-4],t);
x1=[1 0]*x';
x2=[0 1]*x';
u=-(K*x');

figure
plot1=subplot(2,1,1);
plot(t,x1,t,x2)
legend(plot1,{'x1','x2'})
title('Optimal trajectories')

plot2=subplot(2,1,2);
plot(t,u)
legend(plot2,{'u'})
title('Optimal Control Law')
timecriteria_b=stepinfo(sys1)
Jt(end-1)
K
P
```

```
function [K,P,Jt]= genetic_algor(A,B,C,D,ngt,nind,nc,Jmin)
%K =genetic_alg(A,B,C,D,ngt,nind,nc,Jmin)

%number of indiviuals : nind
%number of chromosomes: nc
%number of generations: ngt

pini=rand(nc,nind);

Jt=zeros(1,ngt);
ng=1;
while ng<ngt && Jt(ng)<Jmin
    %evolution of individuals
    J=zeros(nind,1);
    for k=1:nind
        indiv=pini(:,k);
        J(k)=cost_fnc(A,B,C,D,indiv);
    end

    [~,ind]=sort(J);
    Jt(ng)=J(ind(1));
    plot(Jt);pause(0.1)
    pini=pini(:,ind);
    %obtain K gain for chromosomes yielding minimum J
    Q=diag(pini(1:2,1))
    R=(pini(3,1))

    [K,P]=lqr(A,B,Q,R);
    %new populatiion generation
    npini=zeros(nc,nind);
    %Elite selection
    %number of individual of elites
    nie=10;
    npini(:,1:nie)=pini(:,1:nie);
    %mutation
    nim=10;
    %mutate elite population
    cm=randi(3,1,nim); % randomly generated number in between 1:3 as matrix
[1x10]
    for km=1:nim
        ie=pini(:,km);
        ie(cm(km))=rand;
        npini(:,nie+km)=ie;
    end
    %Cross-over
    %random cross-over
    for ncrov=1:80
        ic=randi(100,1,2);
        npini(:,nie+km+ncrov)=[pini(1:2,ic(1));pini(3,ic(2))];
    end

    pini=npini;
    ng=ng+1;
end
```

```
function J= cost_fnc(A,B,C,D,indiv)

Q=diag(indiv(1:2));
R=indiv(3);
K=lqr(A,B,Q,R);

B1=[0;1];
sys1=ss(A-B*K,B1,C,D);

t=0:0.01:10;
x=initial(sys1,[4;-4],t);
u=-(K*x');

timecriteria_b=stepinfo(sys1);

RTmatr=timecriteria_b.RiseTime;
RTmat=RTmatr(RTmatr~=0 & isfinite(RTmatr));
RTmax=max(RTmat);
RTmin=min(RTmat);

STmatr=timecriteria_b.SettlingTime;
STmat=STmatr(STmatr~=0 & isfinite(STmatr));
STmax=max(STmat);
STmin=min(STmat);

PTmatr=timecriteria_b.PeakTime;
PTmat=PTmatr(PTmatr~=0 & isfinite(PTmatr));
PTmax=max(PTmat);
PTmin=min(PTmat);

J=RTmax*RTmin*sum(u.^2)+STmax*STmin*sum(u.^2)+PTmax*PTmin*sum(u.^2);
```