UÇAK İNİŞ SİSTEMİNİN BELİRLENMESİ, TASARIMI VE SİMULİNK ORTAMINDA İNCELENMESİ

DETERMINATION, DESIGN AND INVESTIGATION OF AIRCRAFT LANDING SYSTEM IN SIMULINK ENVIRONMENT

Arda Can İREN

Zonguldak Bülent Ecevit Üniversitesi Elektrik Elektronik Mühendisliği Zonguldak, Türkiye
Acan.iren@mf.karaelmas.edu.tr

ÖZET

Bu çalışmada uçak iniş sistemlerinin belirlenmesi, matematiksel hesaplamalarının yapılması, modellenmesi ve simülasyonlarının incelenmesi hakkında araştırmalarda bulunulmuştur. Simülasyonların incelenmesinde Matlab Simulink ortamı kullanılmıştır.

Anahtar kelimler: Uçak, VTOL, Simulink, İniş Sistemleri, Hesaplamalar

ABSTRACT

In this study, researches were carried out on the determination of aircraft landing systems, making mathematical calculations, modeling and examining their simulations. Matlab Simulink environment was used to analyze the simulations.

Keywords: Aircraft, VTOL, Simulink, Landing Systems, Calculations

1.GİRİŞ

ILS(Instrument Landing System) yani aletli iniş sistemi, pistin başına yerleştirilimiş vericiler aracılığıyla uçakların inişine yardımcı olan bir hassas yaklaşma sistemidir. ILS, uçağın pist başına kadar hassas bir şekilde yaklaşmasını sağlayan yardımcı bir sistemdir. ILS, bulut tavanının alçak olduğu görüş faktörlerinin ise kötü olduğu hava koşullarında, uçağın alçak bir biçimde piste yaklaşmasını ve piste elektronik cihazlarla emniyetli iniş yapmasını sağlar. Pilota istikamet ve süzülüş hattı hakkında bilgi verir.

Pilotların özellikle sisli ve karlı havalar gibi görüş mesafesinin çok düşük olduğu zamanlarda inişlerini büyük ölçüde kolaylaştıran sistemin ILS sisteminin içerisinde bulundurduğu teçhizatlar hakkında bilgi verilecek olunursa;

Yön Bulucu(Localizer) Ünitesi: Pist yüzeyinde yatay doğrultuda ve iniş yönünün tersine doğru çalışan bir sistemdir. Uçağı pist eksenine doğru yönlendirir. Pistin sonundan itibaren 1000 feet (300 metre) uzaklıktadır. Localizer vericileri 25 nm'ye kadar sinyal yayınlar. Bu da uçağın pist uzantısından 25 nm mesafeden itibaren sinyali almaya başladığı anlamına gelir. Uçağa yalnızca pistin yön bilgisini sağlar.Şekil 1'de örnek olarak bir localizer ünitesi gösterilmiştir.

Şekil1

Süzülüş Hattı Belirleyicisi(Glide-Path Unit): Pistin dikeyinde, yaklaşık 3°'lik bir eğimle yayın yapan bir sistemdir. Bu sayede uçağı pist başına doğru yaklaşık 3°'lik bir açıyla yönlendirir. Bu açı, uçağın en uygun süzülme açısı olarak kabul edilir. Şekil 2'de örnek olarak bir glide path ünitesi gösterilmiştir.

Şekil 2

Orta ve Dış Marker Üniteleri(Middle and Outer Marker Beacons Üniteleri): Localizer ve Glide-Path ekipmanı, sinyallerini belirli mesafelerde dikey olarak kesen sistemlerdir. Alçalan uçak pilotlarına, pist başına olan mesafeyi bildirir. Dış, orta ve iç olmak üzere üç işaretçi bulunur. Dış işaretçi, pist başına yaklaşık 4 deniz mili (7240 m) mesafededir. Orta

işaretçi, pist başına 3500 fit (1050 m) mesafededir. İç işaretçi ise pist başından 50 fit (15 m) uzaklığa yerleştirilir. Ayrıca ILS'ler, uçağın 60 metre kadar alçalabildiği CAT1, 30 metre kadar alçalabildiği CAT2 ve sıfır karar yüksekliğindeki CAT3 olmak üzere üç kategoriye ayrılır. Şekil 3'de bir outer marker istasyonu gösterilmiştir.

Şekil 3

Bu sistemlerin daha anlaşılabilmesi için çalışma prensibinin de iyice kavranılması gerekmektedir.

Çalışma prensibi: Bu sistem, iki radyo vericisinden oluşur ve birbirleriyle küçük bir açıda kesişen yayın konisi oluşturur. Uzak verici pistin uzak tarafında yatay düzlemde yayın yapar, yakındaki cihaz ise düşey düzlemde yayına geçerek sistemi etkinleştirir. Kokpitte bulunan iniş sistemi, gelen sinyalleri algılayarak güvenli bir iniş sağlar. ILS sisteminin piste yerleştirilebilmesi için havaalanı çevresindeki radyo frekanslarını engellemeyecek şekilde olması gerekmektedir. Karar Yüksekliği (Decision Height) ve Pist Görüş Mesafesi (Runway Visual Range) açısından üç farklı ILS yaklaşma şekli vardır:

1: Karar yüksekliği > 60 m (200 ft), Görüş Mesafesi > 800m veya Pist Görüş Mesafesi > 550m

2: 60 m (200 ft) > Karar yüksekliği > 30 m (100 ft), Pist Görüş Mesafesi > 350m

3:a) Karar yüksekliği < 30m (100 ft), Pist Görüş Mesafesi > 200m

b) Karar yüksekliği < 15m (50 ft), 200m > Pist Görüş Mesafesi > 50m

c) Karar yüksekliği Minimum yok, Pist Görüş Mesafesi Minimum yok.

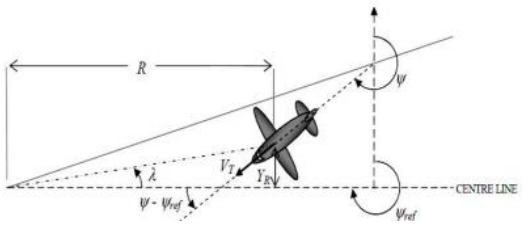
Kullanımı: Kontrollü bir havaalanında, hava trafik kontrolü, belirlenmiş başlıklar yoluyla uçağı yerelleştirici rotaya yönlendirerek, uçakların birbirine çok yaklaşmamasını (ayrılığı muhafaza etmesini) sağlarken aynı zamanda mümkün olduğunca gecikmeden kaçınılmasını sağlar. ILS'de aynı anda birkaç mil uzakta birkaç uçak bulunabilir. Geliş istikametine dönmüş ve yerelleştirici rotanın iki buçuk derece yakınında bulunan bir uçağın (rota sapma göstergesi tarafından gösterilen yarım ölçekli sapma veya daha azı) yaklaşmaya hazır olduğu söylenir. Tipik olarak, bir uçak son yaklaşma noktasından (belirtilen irtifada süzülüş eğimi önlemesi) en az 2 deniz mili (3,7 km) önce kurulur.

En uygun yoldan uçak sapması, uçuş ekibine bir ekran kadranı aracılığıyla gösterilir (bir analog sayaç hareketinin, ILS alıcısından gönderilen voltajlar yoluyla rota çizgisinden sapmayı gösterdiği zamandan itibaren bir aktarım).

ILS alıcısından gelen çıktı, görüntüleme sistemine (baş aşağı gösterge ve varsa baş üstü gösterge) gider ve bir Uçuş Kontrol Bilgisayarına gidebilir. Bir uçak iniş prosedürü, otopilotun veya Uçuş Kontrol Bilgisayarının doğrudan hava aracını uçurduğu ve uçuş ekibinin operasyonu izlediği yerde birleştirilebilir veya uçuş ekibinin yerelleştiriciyi ve süzülme eğimi göstergelerini merkezde tutmak için uçağı manuel olarak uçurduğu yerde ayrık olabilir. **[1]**

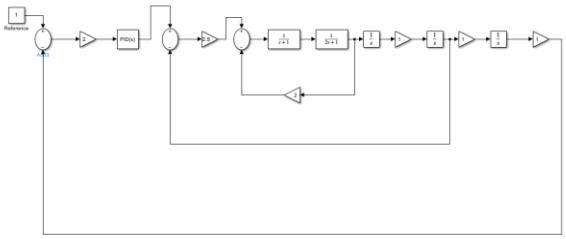
2.Sistem Modellemesi

Sistemin yanal ışık yönlendirme sistemi şekil 4'de gösterilmektedir.



Şekil 4

Sistemin modellenmesi için Matlab Simulink ortamı kullanılmıştır. Bu platformda öncelikle hedeflenen uçuş iniş sisteminin modellenmesi bloklar aracılığıyla gerçeklenmiştir. Burada reference, gain, adder, transfer fcn, integratör, pid controller blokları kullanılmıştır. Şekil 5'de tasarlanan modelleme gösterilmektedir.



Şekil 5

Sistem üç geri bildirim döngüsü içerir. En dıştaki döngü, yön açısının geri bildirimini içerir. Diğer iki döngü, tüm sistemin makul bir dinamik yanıta sahip olmasını sağlamak için gereken geri bildirim sinyallerini sağlar. Referans yön açısıdır, pilot tarafından veya uçaktaki başka bir sistemden elektronik bir sinyal olarak sağlanabilir.

Yuvarlanma oranını (p) kanatçık sapma açısı deltası (δ) ile ilişkilendiren transfer fonksiyonu denklemde belirtilmiştir.

$$\frac{p(s)}{\delta(s)} = \frac{Ka}{sTa + 1} \quad (1)$$

Burada Ka bir kazanç faktörüdür ve Ta bu basitleştirilmiş modelle ilişkili bir zaman sabitidir. Benzer şekilde, δ ve PHI Φ yuvarlanma açısı arasındaki ilişki denklem (2)'dir.

$$\frac{\emptyset(s)}{\delta(s)} = \frac{Ka}{s(sTa + 1)} \quad (2)$$

Kanatçık servosu aşağıdaki şekilde temsil edilir

ilişki:

$$\frac{1}{sT + 1} (3)$$

Yukarıda açıklanan üç transfer fonksiyonu, (4), (5) ve (6) denklemlerindeki gibi bir durum uzayı formuna dönüştürülebilir.

$$\frac{dp}{dt} = -\frac{1}{Ta} p + \frac{Ka}{Ta} \delta (4)$$

$$\frac{d\emptyset}{dt} = p (5)$$

$$\frac{d\delta}{dt} = -\frac{1}{T} \delta + \frac{1}{T} e_s \quad (6)$$

Yuvarlanma açısı ve rota açısı aşağıdaki denklemle ilişkilidir:

$$\frac{d\psi}{dt} = \frac{g}{Vt} \emptyset (7)$$

burada g yerçekimi sabitidir. Geri besleme sisteminin yapısı aşağıdaki denklemlerle tanımlanır:

$$e_s = K_v e_v - K_r p (8)$$

$$e_v = K_d e_d - \emptyset (9)$$

$$e_d = \psi c - \psi (10)$$

Şekil 5, Yanal Kiriş Kılavuzunun çıktıyla karşılaştırılan bir referansa sahip olduğunu göstermektedir. Bu karşılaştırma bir PI denetleyicisi tarafından yapılır. Bu PI denetleyicisinin biçimi aşağıdaki gibidir:

$$C = G_c \left(1 + \frac{k_i}{s}\right) (\lambda_{ref} - \boldsymbol{\lambda}) (11)$$

Uçak ile pistin merkez hattı arasındaki açısal hatayı tahmin etmek için denklem (12) dikkate alınacaktır.

$$\sin(\boldsymbol{\lambda}) = \frac{\boldsymbol{Y_R}}{\boldsymbol{R}} (12)$$

Küçük açı yaklaşımı için, denklem (12) denklem (13) gibi olur.

$$\lambda = \frac{Y_R}{R} \quad (13)$$

Yukarıda açıklanan denklemlerden bu sistem için durum uzay modelini elde etmek mümkündür. İlk adım, aşağıdaki gibi olan durum vektörünü tanımlamaktır:

$$\begin{bmatrix} \lambda \\ Y_R \\ \psi \\ \emptyset \\ p \\ \delta \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

Vektör durumu tanımlandıktan sonra, ikinci adım vektörün her durumunun türevinin alınmasıdır.

$$\begin{bmatrix} \dot{\lambda} \\ \dot{Y_R} \\ \dot{\psi} \\ \dot{\emptyset} \\ \dot{p} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix}$$

Son olarak, (1) ila (13) denklemleri birleştirilir ve durum değişkenleri ikame edilir ve türev denklemlerine indirgenir. İlk durum **λ**, denklemin (13) türevinden elde edilir.

$$\dot{\lambda} = \frac{\dot{Y_R}}{R}$$

where $\dot{Y_R} = Vt\psi$, thus:

$$\dot{\lambda} = \frac{V_t}{R} \psi$$

Ya da

$$\dot{x_1} = \frac{V_t}{R} x_3 \quad (14)$$

Burada $\dot{Y_R} = Vt\psi$ olduğu bilinmektedir.

Bununla ilişkili olarak

$$\dot{x_2} = Vtx_3 (15)$$

elde edilir.

7. denklemden;

$$\dot{x_3} = \frac{g}{Vt} x_4 (16)$$

$$\dot{\phi} = p$$

$$\dot{x_4} = x_5 (17)$$

Denklemleri elde edilir.

4. denklemden

$$\dot{x_5} = -\frac{x_5}{Ta} - \frac{K_a}{T_a} \quad (18)$$

ve 6,8,9 ve 10. Denklemlerden

$$\dot{x_6} = -\frac{x_6}{\tau} + \frac{KvKd}{\tau} x_3 - \frac{k_v}{\tau} x_4 - \frac{k_r}{\tau} x_5 \quad (19)$$

Denklemi elde edilir. 14,15,16,17,18 ve 19. denklemler Matlab'da kullanılacak sistemin durum uzayı modelidir. **[2]**

3. MATLAB SİMULİNK'de SİMİLASYONU

Similasyonda sonuçlar elde etmek için alınan değerler aşağıdaki gibidir:

m = 11

J = 1

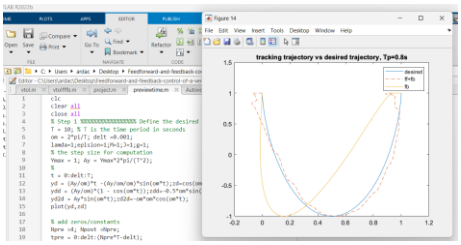
λ = 1

g = 9.81

ε = Değişen

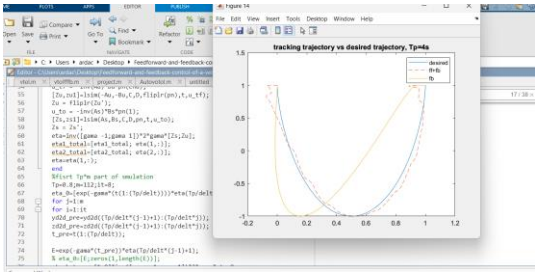
Fiziksel uçakta bağlantı terimi küçüktür ve bu yüzden "ε " değeri 0.05 ile 0.1 arasında değişecek şekilde seçilmiştir.

Similasyonda VTOL uçağın hazırlanan kodlar ışığında çıkan sonuçları görülmektedir. **[3]**



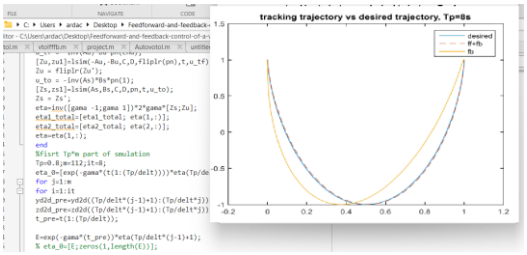
Şekil 6

Burada Tp=0.8s için VTOL uçağın belirlenen yarım daire şeklindeki (0,1)'den (1,1)'e gitmeyi hedeflediği bir yörüngede yapmış olduğu hareket gösterilmektedir. Şekil 7'de ise Tp=4s için yörüntedeki yapmış olduğu hareket gösterilmektedir.



Şekil 7

Buradan hedeflenen yörüngeyi takibin süre Tp değeri arttıkça doğruya daha da yaklaşıldığı söylenebilir. Şekil 8'de ise Tp=8 değerinde izlediği doğruya en yakın yörünge hareketi gözükmektedir.



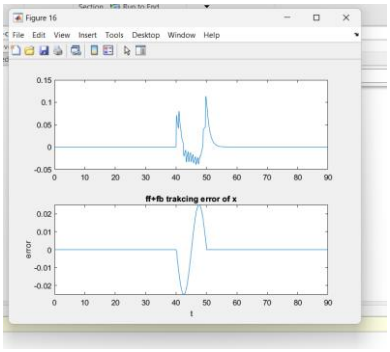
Şekil 8

Bu hesaplamalarda ve analizde kullanılan kod ekler kısmında verilmiştir.

Ayrıca yapılan simülasyonda Tp=8, m=11 için ve geri besleme=K;

$$K = \begin{bmatrix} 0 & 0 & 316.2278 & 25.1487 & 0 & 0 \\ -316.2278 & -226.7874 & 0 & 0 & 797.7686 & 257.8210 \end{bmatrix}$$

İçin ileri besleme ve geri besleme için hata grafiği şekil 9'daki gibidir.



Şekil 9

KAYNAKLAR

[1]https://en.wikipedia.org/wiki/Instrument_landing_system

[2] Instrument Landing System Lateral Beam Guidance System Based on Sliding Mode Control Technique

[3] Output Tracking: Control of a VTOL

EKLER

```
clc
clear all
close all
% Step 1 %%%%%%%%%%%%%%%%%% Define the desired output %%%%%%
T = 10; % T is the time period in seconds
om = 2*pi/T; delt =0.001;
lamda=1;eplsion=1;M=1;J=1;g=1;
% the step size for computation
Ymax = 1; Ay = Ymax*2*pi/(T^2);
%
t = 0:delt:T;
yd = (Ay/om)*t -(Ay/om/om)*sin(om*t);zd=cos(om*t);
ydd = (Ay/om)*(1 - cos(om*t));zdd=-0.5*om*sin(om*t);
yd2d = Ay*sin(om*t);zd2d=-om*om*cos(om*t);
plot(yd,zd)

% add zeros/constants
Npre =4; Npost =Npre;
tpre = 0:delt:(Npre*T-delt);
tpost = 0:delt:(Npost*T-delt);
%
t = 0:delt:(1+Npre+Npost)*T;
yd = [zeros(size(tpre)) yd ones(size(tpost))];
ydd = [zeros(size(tpre)) ydd zeros(size(tpost))];
yd2d = [zeros(size(tpre)) yd2d zeros(size(tpost))];
%
zd = [ones(size(tpre)) zd Ymax*ones(size(tpost))];
zdd = [zeros(size(tpre)) zdd zeros(size(tpost))];
zd2d = [zeros(size(tpre)) zd2d zeros(size(tpost))];
plot(t,yd,t,ydd,t,yd2d)

iteration=16; % number of iterations
% Initializing
eta2_total=[]; eta2_hat=[];% store unstable solution at each iteration
eta1_total=[];eta1_hat=[]; % store stable solution at each iteration
etau = zeros(size(yd));


iteration=16; % number of iterations
% Initializing
eta2_total=[]; eta2_hat=[];% store unstable solution at each iteration
eta1_total=[];eta1_hat=[]; % store stable solution at each iteration
etau = zeros(size(yd));


%%%calculate approxiated eta
gama=sqrt(lamda*M*g/J/eplsion);
eta=etau;
Au=gama; As=-gama; Bu=1/(2*gama); Bs=-1/(2*gama); C=1; D=0;

%calcualte eta_stable for later use
for i=1:iteration
pn=(lamda*M*yd2d.*cos(eta)+lamda*(M*zd2d+M*g).*sin(eta)-lamda*M*g*eta)/(J*eplsion);
u_tf = -inv(Au)*Bu*pn(end);
[Zu,zu1]=lsim(-Au,-Bu,C,D,fliplr(pn),t,u_tf);
Zu = fliplr(Zu');
u_to = -inv(As)*Bs*pn(1);
[Zs,zs1]=lsim(As,Bs,C,D,pn,t,u_to);
Zs = Zs';
eta=inv([gama -1;gama 1])*2*gama*[Zs;Zu];
eta1_total=[eta1_total; eta(1,:)];
eta2_total=[eta2_total; eta(2,:)];
eta=eta(1,:);
end
%fisrt Tp*m part of smulation
Tp=0.8;m=112;it=8;
eta_0=[exp(-gama*(t(1:(Tp/delt))))*eta(Tp/delt+1);zeros(1,Tp/delt)];
for j=1:m
for i=1:it
yd2d_pre=yd2d((Tp/delt*(j-1)+1):(Tp/delt*j));
zd2d_pre=zd2d((Tp/delt*(j-1)+1):(Tp/delt*j));
t_pre=t(1:(Tp/delt));

E=exp(-gama*(t_pre))*eta(Tp/delt*(j-1)+1);
% eta_0=[E;zeros(1,length(E))];
eta_hat_pre=[1 0]*inv([gama -1;gama 1])*2*gama*eta_0;
pn=(lamda*M*yd2d_pre.*cos(eta_hat_pre)+lamda*(M*zd2d_pre+M*g).*sin(eta_hat_pre)-lamda*M*g*eta_hat_pre)/(J*eplsion);

u_tf = -inv(Au)*Bu*pn(end);
[Zu,zu2]=lsim(-Au,-Bu,C,D,fliplr(pn),t_pre,u_tf);
Zu = fliplr(Zu');

u_to = -inv(As)*Bs*pn(1);
[Zs,zs2]=lsim(As,Bs,C,D,pn,t_pre,u_to);
Zs = Zs'+E;
eta_0=[Zs;Zu];
end
eta_f=inv([gama -1;gama 1])*2*gama*[Zs;Zu];
eta1_hat=[eta1_hat eta_f(1,:)];
eta2_hat=[eta2_hat eta_f(2,:)];
end
```

```
%rest of t-Tp*m part of simulation
eta_0=eta_0(:,(Tp/delt+1-(length(t)-Tp*m/delt)):end);
for i=1:it
yd2d_pre=yd2d(Tp*m/delt+1:length(t));
zd2d_pre=zd2d(Tp*m/delt+1:length(t));
t_pre=t(1:(length(t)-Tp*m/delt));

E=exp(-gama*(t_pre))*eta(Tp*m/delt+1);
eta_hat_pre=[1 0]*inv([gama -1;gama
1])*2*gama*eta_0;
pn=(lamda*M*yd2d_pre.*cos(eta_hat_pre)+lamda*(
M*zd2d_pre+M*g).*sin(eta_hat_pre)-
lamda*M*g*eta_hat_pre)/(J*eplsion);

u_tf = -inv(Au)*Bu*pn(end);
[Zu,zu]=lsim(-Au,-Bu,C,D,fliplr(pn),t_pre,u_tf);
Zu = fliplr(Zu');

u_to = -inv(As)*Bs*pn(1);
u_to = Zs(end);
[Zs,zs]=lsim(As,Bs,C,D,pn,t_pre,u_to);
Zs = Zs'+E;
eta_0=[Zs;Zu];
end
eta_f=inv([gama -1;gama 1])*2*gama*[Zs;Zu];
eta1_hat=[eta1_hat eta_f(1,:)];
eta2_hat=[eta2_hat eta_f(2,:)];

%simulation
eta1=eta1_hat;eta2=eta2_hat;
uff=1/eplsion*[-
eplsion*sin(eta1)*M.*yd2d+eplsion*cos(eta1).*(M*zd
2d+M*g);cos(eta1)*M.*yd2d+sin(eta1).*(M*zd2d+M*
g)];
figure(3)
plot(t,uff)
xlabel('time')
ylabel('inverse input')
legend('u_1ff,','u_2ff')

U=[t;[uff(1,:);uff(2,:)]];
x0=[0 0 1 0 0 0]';
[t,x]=ode45('vtol',t,x0,[],U);

figure(4)
subplot(211)
plot(t,yd,t,x(:,1))
xlabel('time')
ylabel('output')
legend('desired trajectory x','tracking trajectory
x_{ff}')
subplot(212)
plot(t,zd,t,x(:,3))
xlabel('time')
ylabel('output')
legend('desired trajectory z','tracking trajectory
z_{ff}')

%feedback controller
A=[0 1 0 0 0 0;
   0 0 0 0 -g 0;
   0 0 0 1 0 0;
   0 0 0 0 0 0;
   0 0 0 0 0 1;
   0 0 0 0 0 0];
B=[0 0;0 eplsion/M;0 0;1/M 0;0 0;0 lamda/J];
C=[1 0 0 0 0 0;0 0 1 0 0 0];
D=0;
R=1/100000*eye(2);Q=C'*C;
[K1,S,e]=lqr(A,B,Q,R);
A1=A-B*K1;
sys_close=ss(A1,B,C,D);
go= dcgain(sys_close);%calculate go
r=inv(go)*[yd;zd];
[y1,t,x1]=lsim(sys_close,r,t,x0);%simulate

figure(5)
subplot(211)
plot(t,y1(:,1),t,yd)
title('desired trajectory vs fb simulation,T=5')
xlabel('t')
ylabel('x')
legend('tracking trajectory x_{fb}','desired trajectory')
subplot(212)
plot(t,y1(:,2),t,zd)
title('desired trajectory vs fb simulation,T=5')
xlabel('t')
ylabel('z')
legend('tracking trajectory z_{fb}','desired trajectory')


%add feedback to feedforward vs feedback alone in
ODE45
U=[t';[uff(1,:);uff(2,:)]];
k=[yd;ydd;zd;zdd;eta1;eta2];

[t3,x3]=ode45('vtolffffb',t,x0,[],U,k,K1);
figure(6)
subplot(211)
plot(t,yd,t,x3(:,1))
xlabel('time')
ylabel('output x')
legend('desired trajectory x','x_{ff+fb}')
subplot(212)
plot(t,zd,t,x3(:,3))
xlabel('time')
ylabel('output z')
legend('desired trajectory z','z_{ff+fb}')

figure(7)
subplot(211)
plot(t,x3(:,1),t,yd,t,y1(:,1),'--')
title('desired trajectory vs fb vs fb+ff T=5')
xlabel('t')
ylabel('x')
legend('x_{ffffb}','desired trajectory','x_{fb}')
subplot(212)
plot(t,x3(:,3),t,zd,t,y1(:,2),'--')
title('desired trajectory vs fb vs fb+ff T=5')
xlabel('t')
ylabel('z')
legend('z_{ffffb}','desired trajectory','z_{fb}')


ufb=-K1*(x3'-k);
figure(8)
plot(t,ufb(1,:),t,ufb(2,:),t,uff(1,:),'--',t,uff(2,:),'--')
title('ff+fb input vs fb input T=5')
xlabel('t')
ylabel('u')
legend('u1_{fb}','u2_{fb}','u1_{ff}','u2_{ff}')

figure(14)
```

```
plot(yd,zd)
hold on
plot(x3(:,1),x3(:,3),'--')
hold on
plot(x1(:,1),x1(:,3))
legend('desired','ff+fb','fb')
title('tracking trajectory vs desired trajectory, Tp=4s')

figure(15)
subplot(211)
plot(t,yd,t,x1(:,1),t,x3(:,1))
xlabel('t')
ylabel('x')
legend('desired','fb','ff+fb')
subplot(212)
plot(t,zd,t,x1(:,3),t,x3(:,3))
xlabel('t')
ylabel('z')
legend('desired','fb','ff+fb')

error1=yd-x3(:,1)';
error2=zd-x3(:,3)';
figure(16)
subplot(211)
plot(t,error1)
subplot(212)
plot(t,error2)
xlabel('t')
ylabel('error')
title('ff+fb trakcing error of x')
```