

Latest Rho-Synthesis Example

7-44

Internal Letter



Rockwell International

Date: . April 12, 1992

No. . FILE #: SUJT9206.CHI

TO: (Name, Organization, Internal Address)

FROM: (Name, Organization, Internal Address, Phone)

. Sujeet Chand
. Control & Signal Processing
. 083,D/325,A18
. x4295

. Robert W. Bass
. Control & Signal Processing
. 083,D/325,A24
. x4255

Subject . Rho Synthesis of STOL Autopilot for Collins Commercial Avionics

cc: Allen Firstenberg, Charles J. Sitter (CCAD)

Using the typical STOL (Short Take-Off & Landing) aircraft dynamics model suggested by Ming-Shong Lan, I have designed an "optimally rhobust" autopilot consisting of 12 gains, six applicable as multiplicative gains to:

forward velocity
vertical velocity
pitch
pitch rate
elevator actuator state
throttle actuator state

to produce the *elevator command input*, and the other six applicable to the same six variables as multiplicative gains to produce the *throttle command input*. These twelve gains K_{ij} ($i = 1, 2, 3, 4, 5, 6; j = 1, 2$) define an optimal feedback gain matrix $K = (K_{ij})$ which is presented numerically in the *MATLAB* diaries which are appended.

The resulting design seems to me vastly superior to that published. There the response time was 2 seconds; here the response time is 0.6 second, i.e. more than *three times* faster. The published transient responses are over 4 seconds; mine are over 2 seconds, and could have been over just 1 second.

Note that *all six* state variables are "maximally de-coupled" and behave like first-order over-damped lags. The closed-loop poles are all real and negative.

Note that when the open-loop transient responses are plotted on the same graph with the closed-loop responses, the latter cannot even be seen!

Note also that the scales for the forward speeds and the vertical speeds are in the same (arbitrary) units; however, the scale should be *divided by ten* when scanning the pitch and pitch rate (because I had to multiply them by ten to make them visible).

Chuck Sitter can use the K -matrix derived below as the basis for further simulations involving maneuvers or varying input commands.

MATLAB DIARY 04_19B92.TXT

```
» diary d:\matlab\txtfiles\04_10b92.txt
» load d:\matlab\matfiles\04_09b92.mat
```

The problem is to design a state-vector feedback control law

$$u = -Kx$$

for a given dynamical coefficient matrix A and input distribution matrix B such that the system evolves in time t according to

$$\dot{x} = Ax + Bu, \quad (\dot{} = d/dt, \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m)$$

where in the present case $n = 6$ and $m = 2$. The state variables are

- x_1 = forward speed
- x_2 = vertical speed
- x_3 = pitch angle
- x_4 = pitch rate
- x_5 = elevator state (lags elevator command)
- x_6 = throttle state (lags throttle command)

while the control variables are

- u_1 = elevator command
- u_2 = throttle command

For a simplified model of a STOL aircraft autopilot we take that given in equations (28)-(31) of the paper "Design of Desirable Airplane Handling Qualities via Optimal Control" by G.K.L. Kriechbaum & R.W. Stineman, *J. Aircraft*, vol. 9, No. 5, May 1972, pp. 365-369. However we simplify this model by reducing n from $n = 7$ to $n = 6$ by omitting state corresponding to a spoiler actuator and by reducing m from $m = 3$ to $m = 2$ by omitting spoiler command. Accordingly, the model assumed is defined by:

```
» A
```

```
A =
```

```
Columns 1 through 4
```

```
-0.065000000000000000 -0.107800000000000000 -15.4700000000000000 0
-0.434000000000000000 -0.570000000000000000 -88.4000000000000001 0.391000000000000000
0 0 0 1.000000000000000000
0.000720000000000000 -0.004180000000000000 -0.648000000000000000 -1.657000000000000000
0 0 0 0
0 0 0 0
```

```
Columns 5 through 6
```

```
0 61.400000000000000000
-99.8199999999999999 0
0 0
-29.0800000000000000 0
-14.0000000000000000 0
0 -5.0000000000000000
```

» B
B =

```
0 0
0 0
0 0
0 0
1 0
0 1
```

Now choose to search over the η -range [1,9] by defining $etamin \equiv mn = 1$
and $etamax \equiv mx = 9$:

» mn=1.0

mn =

1

» mx=9.0

mx =

9

Divide this interval into 80 subdivisions, each of width 0.1 by taking
the number of η -points to be $N = 81$:

» N=81

N =

81

Leave the number of t -search points at $N_c = 100$.

» Nc

Nc =

100

Reorder RAM & hard-disk to garbage-collect:

» pack

Call the main rho-synthesis routine:

» [Krh,etrh,wbsschv,etfi,etrsp,etchi,P]=wrlqrpl1(A,B,mn,mx,N,Nc)

jkount =

10

jkount =

20

jkount =

30

jkount =

40

jkount =

50

jkount =

60

jkount =

70

jkount =

80

Here is the information to scale the printed-out graph:

maxRho =

0.08742906527972

maxSig =

0.07005784911881

minTresp =

0.54286976005121

minMu =

5.67853676898450

minMp =
1.000000000000000
maxMp =
1
minOmegaBW =
3.46237633135620
maxOmegaBW =
29.66222127732514
minOmeg =
9.252610754035262e+03
maxOmeg =
9.069165435430252e+05
maxkpinf =
0.17610170378078
maxChi =
5.797389849182520e+04
minChi =
6.757835583867614e+03
rho =
0.08717443997328
sig =
0.05774354792960
trsp =
0.58856081941424
etarh =
8.100000000000000

Here are the Bass characteristics of the various optimal designs:

rho =
0.08717443997328
sig =
0.05774354792960
trsp =
0.58856081941424
mu =
5.89052956239885
Mpeakres =
1
omegaBW =
25.60543901841574
Omega =
5.988903706152698e+05
kappinf =
0.16976402365983
omegstr =
0
lambd =
10.61912345137113
gamm =
1.218146449191510e+02
tstr =
0.09416973110628
tstrsg =
0.07062729832971

Here is the gain matrix corresponding to maximum rho:

```
Krh =
  1.0e+03 *
Columns 1 through 4
-0.00462068698884  0.15627460414694  -3.73730539775403  -0.74332871045858
  0.02319409781308  -0.03373621660596  0.56616316694374  0.13205856948485
Columns 5 through 6
  0.14563795505610  -0.00323771613605
-0.00323771613629  0.08519100075932
etrh =
  8.10000000000000
wbsschv =
  1.0e+05 *
  0.00000087174440
  0.00000057743548
  0.00000588560819
  0.00005890529562
  0.00001000000000
  0.00025605439018
  5.98890370615270
  0.00000169764024
  0
  0.00010619123451
  0.00121814644919
  0.00000094169731
  0.00000070627298
etfi =
  3.80000000000000
etrsp =
  9
etchi =
  1
```

Here is the P-matrix corresponding to the optimal rho:

```
P =
  1.0e+05 *
Columns 1 through 4
  0.00011843599314  -0.00049050612881  0.00853473557076  0.00192893840693
-0.00049050612880  0.01672274454611  -0.29325954566381  -0.06571420575799
  0.00853473557056  -0.29325954566350  5.72557056887719  1.19009722144999
  0.00192893840689  -0.06571420575797  1.19009722145085  0.26138303775324
-0.00004620686989  0.00156274604147  -0.03737305397754  -0.00743328710459
  0.00023194097813  -0.00033736216606  0.00566163166944  0.00132058569485
Columns 5 through 6
-0.00004620686989  0.00023194097813
  0.00156274604147  -0.00033736216605
-0.03737305397761  0.00566163166913
-0.00743328710461  0.00132058569479
  0.00143437955056  -0.00003237716136
-0.00003237716136  0.00073991000759
» Krh
Krh =
  1.0e+03 *
Columns 1 through 4
-0.00462068698884  0.15627460414694  -3.73730539775403  -0.74332871045858
  0.02319409781308  -0.03373621660596  0.56616316694374  0.13205856948485
Columns 5 through 6
  0.14563795505610  -0.00323771613605
-0.00323771613629  0.08519100075932
```

```

> Krh'
ans =
  1.0e+03 *
  -0.00462068698884    0.02319409781308
  0.15627460414694   -0.03373621660596
  -3.73730539775403   0.56616316694374
  -0.74332871045858   0.13205856948485
  0.14563795505610   -0.00323771613629
  -0.00323771613605   0.08519100075932

```

```

> norm(Krh)
ans =
  3.860604783732912e+03

```

The open-loop (uncontrolled) plant dynamics is not stable:

```

> eigAv=eig(A)
eigAv =
  0.03051868938891
 -0.05553521615910
 -1.13349173661490 + 0.59352630294891i
 -1.13349173661490 - 0.59352630294891i
 -14.00000000000000
 -5.00000000000000

```

But the closed-loop poles are all real and negative:

```

> Acl=A-B*Krh
Acl =
  1.0e+03 *
  Columns 1 through 4
  -0.0000650000000000   -0.0001078000000000   -0.0154700000000000           0
  -0.0004340000000000   -0.0005700000000000   -0.0884000000000000   0.0003910000000000
  0 0 0 0.0010000000000000
  0.0000007200000000   -0.0000041800000000   -0.0006480000000000   -0.0016570000000000
  0.00462068698884   -0.15627460414694   3.73730539775403   0.74332871045858
  -0.02319409781308   0.03373621660596   -0.56616316694374   -0.13205856948485
  Columns 5 through 6
  0 0.0614000000000000
  -0.0998200000000000 0
  0 0
  -0.0290800000000000 0
  -0.15963795505610   0.00323771613605
  0.00323771613629   -0.09019100075932

```

```

> eigAclv=eig(Acl)
eigAclv =
  1.0e+02 *
  -1.12111014225458
  -0.10626776718316
  -0.70023639598471
  -0.20249265563054
  -0.19555129854906
  -0.19555129855222

```

The response-time is just:

```

> trsp
trsp =
  0.5050000000000000

```

To verify this (we should get 2.17) to present numerical accuracy:

```

> 1/norm(expm(Acl*trsp))
ans =
  1.27500206361171

```

So the actual response time is somewhat less than $trsp \approx 0.505$ (the numerical inaccuracy is due to the fact that we took $Nc = 100$; to get greater accuracy, we would have to take $Nc = 1000$ (and then the program would take hours instead of minutes to run).

To similar coarse accuracy, the eta-value giving the maximum rho is:

```
» etrh
etrh =
  8.100000000000000
```

Now check the frequency-response characteristics of the total closed loop system by the following n -dimensional generalization of the Bode plot:

```
» [Mp1,omegst1]=peakrspl(Acl,wmx,200)
Mp1 =
  1.07327762657443
omegst1 =
  7
```

Now list the Wiberg-improved Bass characteristics:

```
» lstbssch(wbsschv)
rho =
  0.08717443997328
sig =
  0.05774354792960
trsp =
  0.58856081941424
mu =
  5.89052956239885
Mpeakres =
  1
omegaBW =
  25.60543901841574
Omega =
  5.988903706152698e+05
kappinf =
  0.16976402365983
omegstr =
  0
lambd =
  10.61912345137113
gamm =
  1.218146449191510e+02
tstr =
  0.09416973110628
tstrsg =
  0.07062729832971
```

```
» Mpeakres=wbsschv(5)
Mpeakres =
  1
```

The discrepancy between the Mp from the plot and from the synthesis program (i.e. 1.0 vs 1.073) has to do with the omega-range and the coarseness or refinement of the grid used in the two different computations.

```

> P
P =
  1.0e+05 *
Columns 1 through 4
  0.00011843599314 -0.00049050612881  0.00853473557076  0.00192893840693
-0.00049050612880  0.01672274454611 -0.29325954566381 -0.06571420575799
  0.00853473557056 -0.29325954566350  5.72557056887719  1.19009722144999
  0.00192893840689 -0.06571420575797  1.19009722145085  0.26138303775324
-0.00004620686989  0.00156274604147 -0.03737305397754 -0.00743328710459
  0.00023194097813 -0.00033736216606  0.00566163166944  0.00132058569485
Columns 5 through 6
-0.00004620686989  0.00023194097813
  0.00156274604147 -0.00033736216605
-0.03737305397761  0.00566163166913
-0.00743328710461  0.00132058569479
  0.00143437955056 -0.00003237716136
-0.00003237716136  0.00073991000759
> eigPv=eig(P)
eigPv =
  1.0e+05 *
  5.98890370608720
  0.01497384771697
  0.00080313950914
  0.00122128081743
  0.00003355129855
  0.00003355129855

```

Now plot transient responses of the closed-loop system to initial state impulsive displacements (choose final time $T_f = 1$ sec and number of points to plot as $N = 200$):

```

> N=200
N =
  200
> Tf=1.0
Tf =
  1

```

Choose as outputs the first four of the six state variables, but multiply the latter two by 10 to make them more visible on the same graph:

```

C =
  1   0   0   0   0   0
  0   1   0   0   0   0
  0   0  10   0   0   0
  0   0   0  10   0   0

```

First take an initial impulsive displacement in forward speed:

```

> x0=[1 0 0 0 0 0]'
x0 =
  1
  0
  0
  0
  0
  0

```

Now call the closed-loop design simulation:

```

> cldsgnd4(Acl,B,C,x0,Tf,N)
jkount =
  20

```

```
jkount =  
    40  
.  
.  
.  
    180  
jkount =  
    200
```

The result is plotted. Next, null out the initial state and choose an impulsive displacement in pitch rate:

```
> pack  
  
> x0=0*x0  
x0 =  
    0  
    0  
    0  
    0  
    0  
    0  
    0  
> x0(4)=1:0  
x0 =  
    0  
    0  
    0  
    1  
    0  
    0
```

```
> pack  
  
> cldsgnd4(Ac1,B,C,x0,Tf,N)  
jkount =  
    20  
.  
.  
.  
jkount =  
    200
```

The result is plotted.

Both sets of closed-loop transient responses to initial state displacements display excellently damped characteristics.

Using the above matrices A , B , C , K , and $A_{cl} \equiv A - B \cdot K$, further simulations of the preceding rho-synthesis design can be obtained.

```
> save d:\matlab\matfiles\04_10a92.mat  
> quit  
672478336 flop(s).
```

```

function [Krh,etarh,wibsschrs,etafi,etatrsp,etachi,P]=wrlqrp11(A,B,mn,mx,N,Nc)
etamin=mn;etamax=mx;[n1,n]=size(A);Nm1=N-1;In=eye(n);etdl=(etamax-etamin)/Nm1;
x=0*ones(1,N);y1=x;y2=x;y3=x;y4=x;y5=x;y6=x;y7=x;y8=x;y9=x;y0=x;
[K0,v0]=wrlqrdsf(A,B,1000,etamin);vt=[v0(3),v0(12),v0(13)];tlim=max(vt);
tlim=2*tlim;
for j=1:N,
    if 10*fix(j/10)==j, jkount=j, end;
    eta=etamin+(j-1)*etdl;
    [Keta,wibsschrs,P]=wrlqrds1(A,B,Nc,eta,tlim);v=wibsschrs;x(j)=eta;
    y1(j)=v(1);y2(j)=v(2);y3(j)=v(3);y4(j)=v(4);vtmp=2*[v(3),v(12),v(13)];
    y5(j)=v(5);y6(j)=v(6);y7(j)=v(7);y8(j)=v(8);tlim=max(vtmp);y9(j)=tlim;
    y0(j)=norm(B*Keta)/v(1);
end
[maxy1,jindx]=max(y1);etarho=etamin+(jindx-1)*etdl;tlim=y9(jindx);
[maxy2,jindxfi]=max(y2);etafi=etamin+(jindxfi-1)*etdl;
[miny3,jindxtrsp]=min(y3);etatrsp=etamin+(jindxtrsp-1)*etdl;
[miny0,jindxchi]=min(y0);etachi=etamin+(jindxchi-1)*etdl;
y1m=max(abs(y1)); % max of rho;
y2m=max(abs(y2)); % max of sig;
y3m=max(abs(y3)); % max of tr;
y3min=miny3; % min of tr;
y4m=max(abs(y4)); % max of mu;
y4min=min(abs(y4)); % min of mu;
y5m=max(abs(y5)); % max of Mp;
y5min=min(abs(y5)); % min of Mp;
y6m=max(abs(y6)); % max of omegaBW;
y6min=min(abs(y6)); % min of omegaBW;
y7m=max(abs(y7)); % max of Omeg;
y7min=min(abs(y7)); % min of Omeg;
z=max(abs(y8)); % max of kapinf;
y0m=max(abs(y0)); % max of chi=norm(B*K)/rho
y0min=min(abs(y0)); % min of chi=norm(B*K)/rho
maxRho=y1m
maxSig=y2m
minTresp=y3min
minMu=y4min
minMp=y5min
maxMp=y5m
minOmegaBW=y6min
maxOmegaBW=y6m
minOmeg=y7min
maxOmeg=y7m
maxkapinf=z
maxChi=y0m
minChi=y0min
pause; % hit any key to continue
z1=y1/y1m;z2=y2/y2m;z3=y3/y3m;z4=y4/y4m;z5=y5/y5m;z6=y6/y6m;z7=y7/y7m;z8=y8/z;z0=y0/y0m
plot(x,z1,x,z2,x,z3,'w',x,z4,'+w',x,z5,'*w',x,z6,'ow',x,z7,'xw',x,z8,':w',x,z0,'%w');
title('COST v RHOBUSTNESS of STABILITY & FIDELITY; tr; k; Mp; mu; wBW; chi');
xlabel('DESIGN parameter ETA');ylabel('criteria MAGNITUDES');pause;meta wrlqrp1f;
plot(y6,z1,y6,z2,y6,z3,'w',y6,z4,'+w',y6,z5,'*w',y6,z7,'xw',y6,z8,'ow',y6,z0,'%w');
title('COST v RHOBUSTNESS of STABILITY & FIDELITY; tr; k; Mp; mu; chi');
xlabel('[n-dimensional] BANDWIDTH omegaBW');ylabel('criteria MAGNITUDES');pause;
meta wrlqrp1f;pause;etrh=etarho;
[Krh,wibsschrs,P]=wrlqrds1(A,B,Nc,etrh,tlim);v=wibsschrs;rho=v(1)
sig=v(2)
trsp=v(3)
etarh=etrh
pause;lstbssch(v);pause;end

```

```

function [K,wibsschrs,P]=wrlqrds1(A,B,N,eta,tlim)
[n1,n]=size(A);[n,m]=size(B);In=eye(n);
Aeta=A+eta*In;
Qeta=A'*A+2*eta*(A+A')+5*eta*eta*In;
Seta=(A'+2*eta*In)*B;
R=B'*B;Rinv=inv(R);
[K,Peta]=lqr2(Aeta,B,Qeta,R,Seta);
Acl=A-B*K;P=Peta;
Qwib=2*eta*Peta+Peta*(B*Rinv*B')*Peta+(Qeta-Seta*Rinv*Seta');
[wibsschrs]=wibsschl(Acl,Qwib,N,tlim);
end

```

```

function [Mp,omegstr]=peakrspl(A,wmx,N)
i=sqrt(-1);Np1=N+1;x=0*ones(1,Np1);y=x;xtmp=0;Etmp=0*A;Ftmp=Etmp;
[n1,n]=size(A);In=eye(n);
    % wmax=(1+sqrt(2))*norm(A)/(sqrt(2)-1);
wmax=wmx;wdel=wmax/N;x(1)=0;den=norm(inv(A));y(1)=1.0;
for j=2:Np1,
    x(j)=(j-1)*wdel;
    xtmp=x(j);
    Etmp=xtmp*i*In-A;
    Ftmp=inv(Etmp);
    y(j)=norm(Ftmp)/den;
end
[Mp,jindx]=max(y);
omegstr=(jindx-1)*wdel;
plot(x,y);
title('norm(inv(j*w*In - A))/norm(inv(A)) for 0 < w < wmax');
xlabel('Frequency w');ylabel('magnitude');
meta peakrspl;pause;
end

```

```

function cldsgnd4(A,B,C,x0,Tf,N)
tstp=Tf/N;Np1=N+1;[n1,n]=size(A);[n,m]=size(B);[1,n]=size(C);
t=0: tstp :Tf;
for i=1:Np1
    tau=t(:,i);
    X=expm(A*tau);
    xnew=X*x0;
    y(:,i)=C*xnew;
        if i==20*fix(i/20)
            jkount=i
        end
end

end

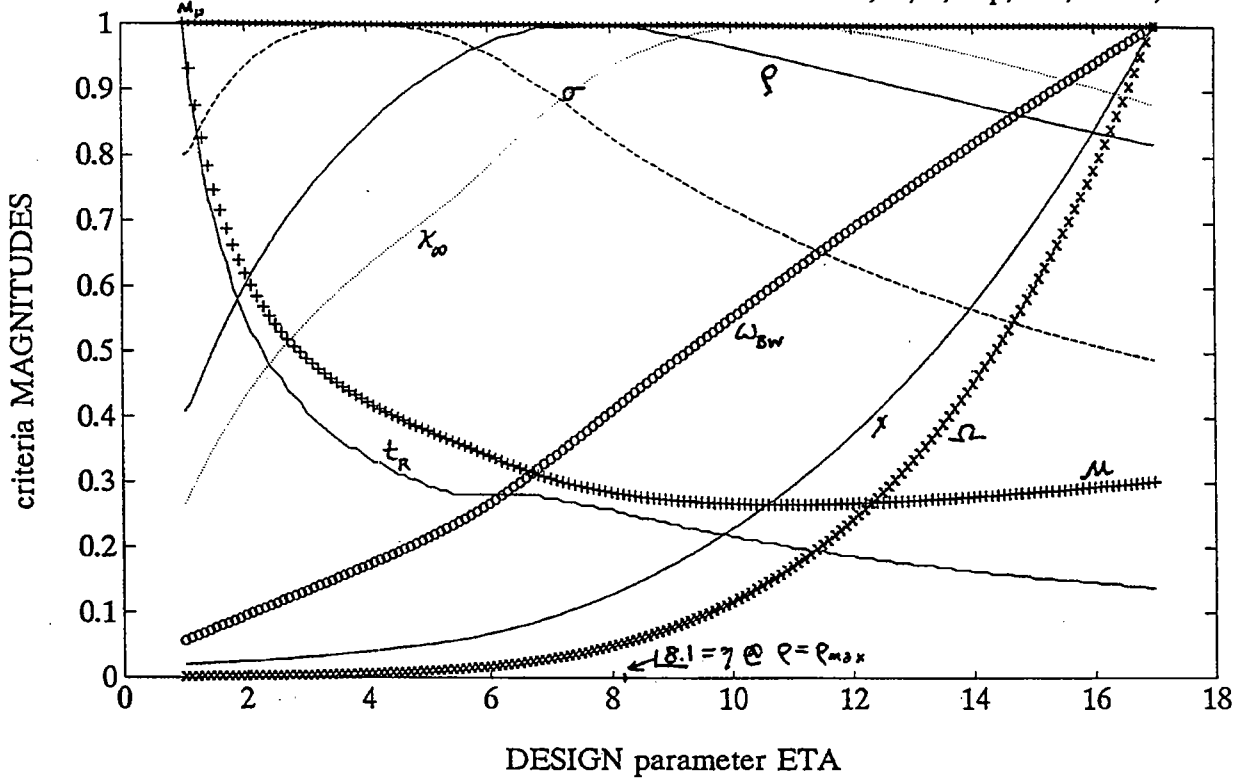
plot(t,y),
title('Closed-Loop Initial State-Displacement Response');

% text(0.001,-0.2,'outputs');text(0.0035,0.2,'disturbance estimate error');
% text(0.003,0.6,'[scaled] total error norm');

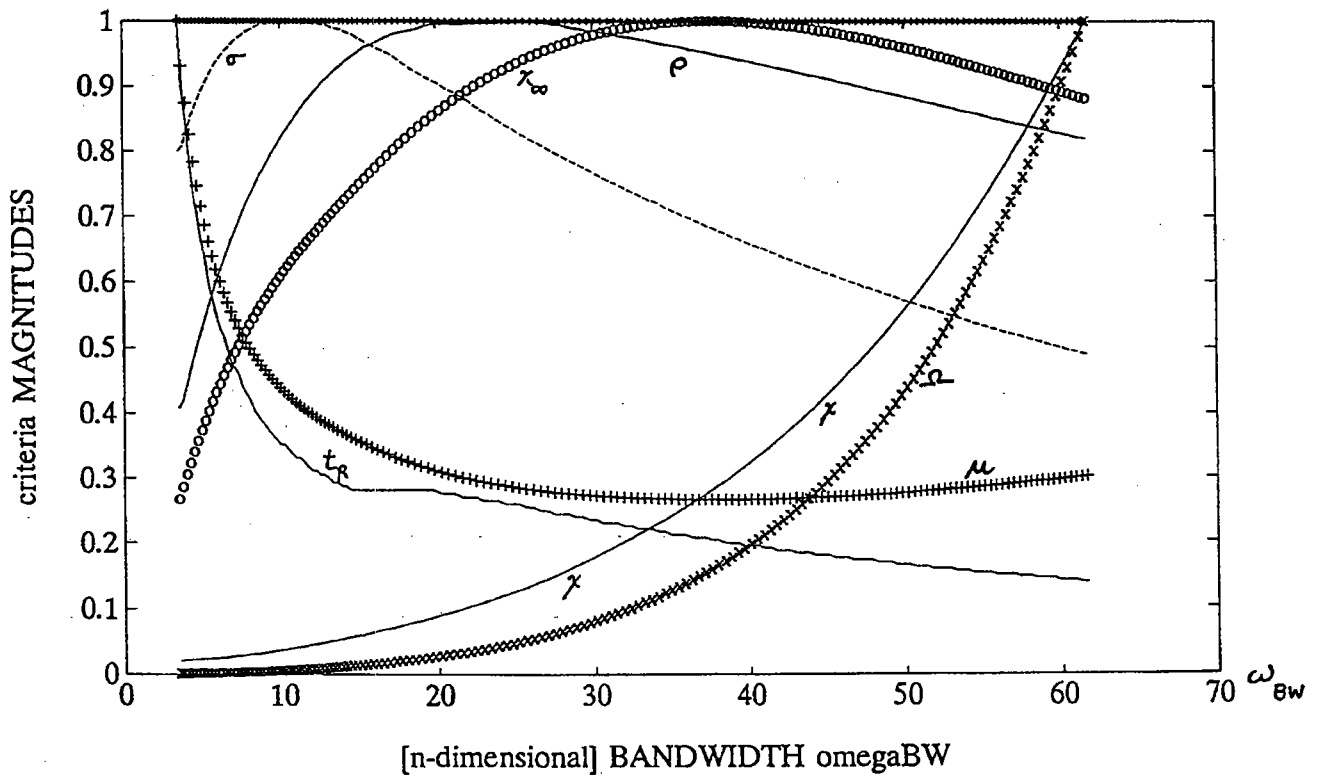
xlabel('Time t'),ylabel('Outputs');pause;
meta cldsgnd4;
pause;
end

```

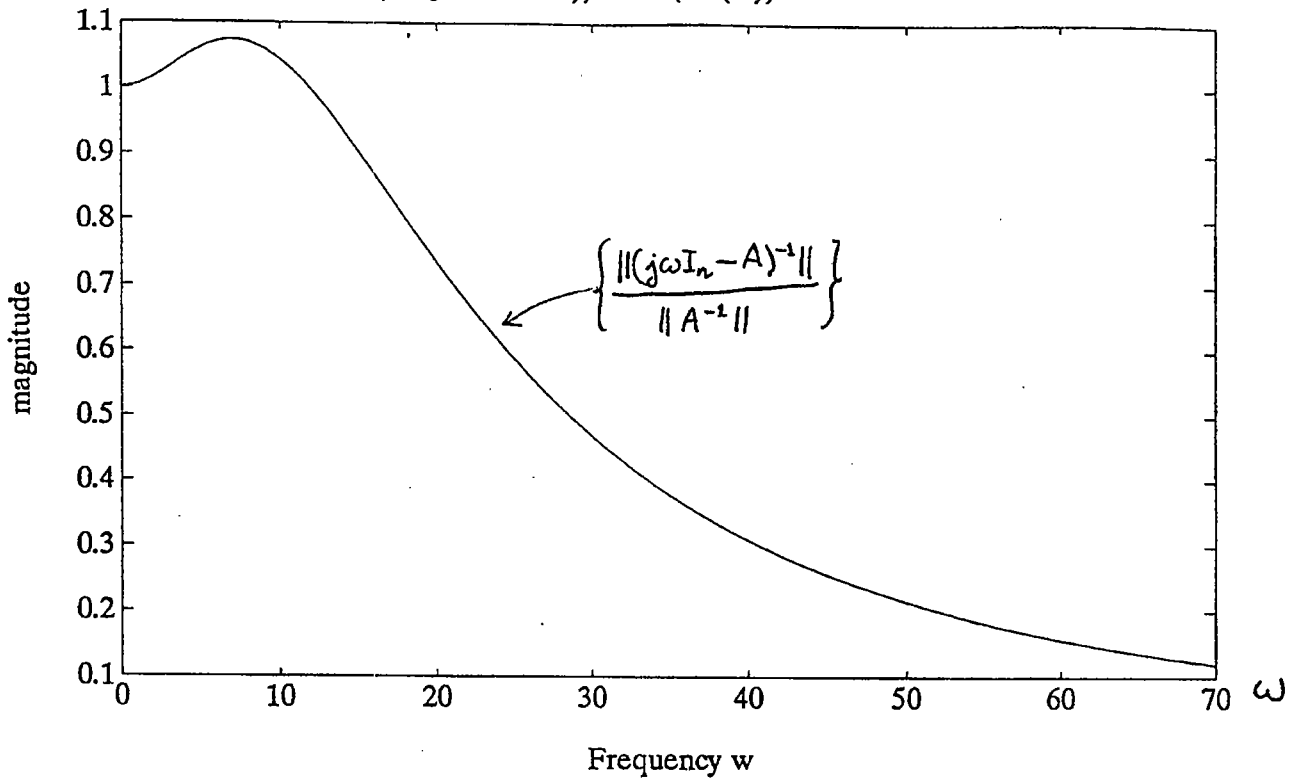
COST v ROBUSTNESS of STABILITY & FIDELITY; t_r ; k ; M_p ; μ ; ω_{BW} ; χ



COST v ROBUSTNESS of STABILITY & FIDELITY; t_r ; k ; M_p ; μ ; χ

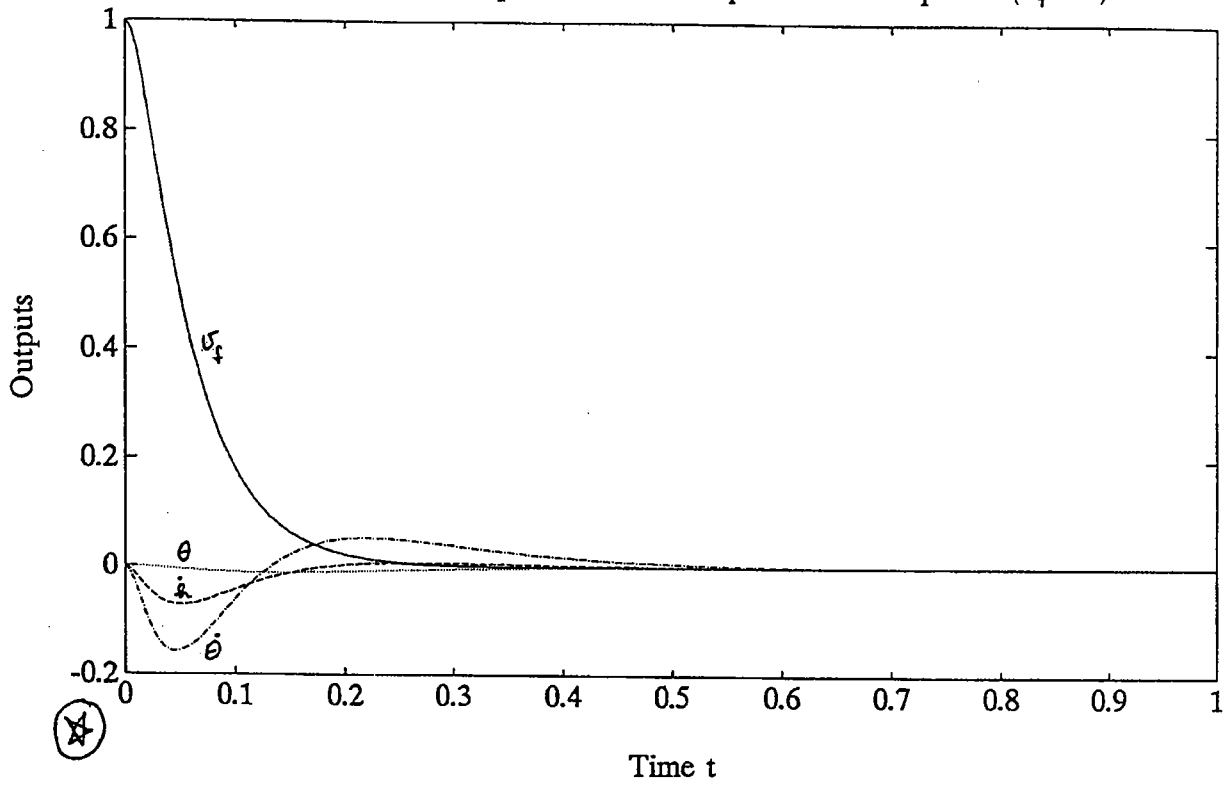


norm(inv(j*w*I_n - A))/norm(inv(A)) for 0 < w < wmax

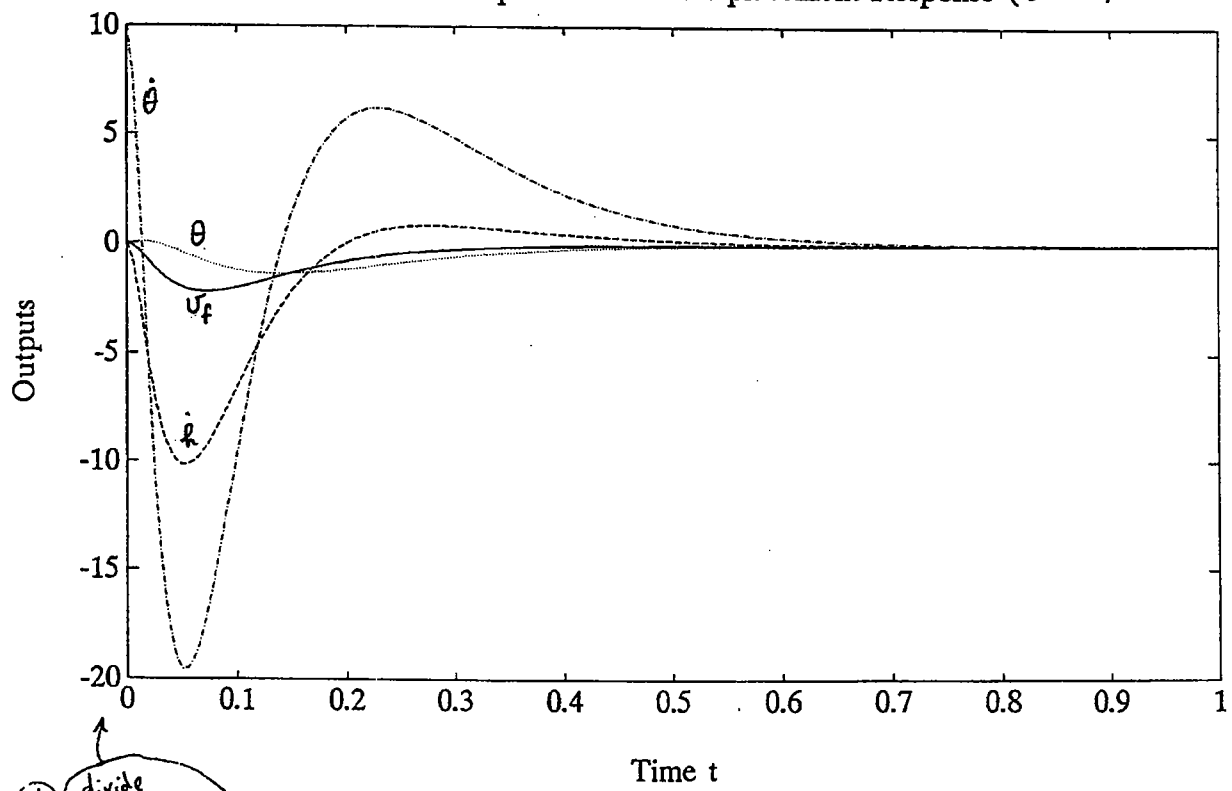


n-dimensional Bode plot

Closed-Loop Initial State-Displacement Response ($U_f = 1$)



Closed-Loop Initial State-Displacement Response ($\dot{\theta} = 1$)



MATLAB DIARY 04_12E92.TXT

```

> diary d:\matlab\txtfiles\04_12e92.txt
> pack
> load d:\matlab\matfiles\04_12b92.mat

```

The variables needed for a comparative simulation of open-loop vs closed-loop response are as follows:

```
> A
```

```
A =
```

```
Columns 1 through 4
```

```

-0.0650000000000000 -0.1078000000000000 -15.470000000000000 0
-0.4340000000000000 -0.5700000000000000 -88.400000000000001 0.3910000000000000
0 0 0 1.0000000000000000
0.0007200000000000 -0.0041800000000000 -0.6480000000000000 -1.6570000000000000
0 0 0 0
0 0 0 0

```

```
Columns 5 through 6
```

```

0 61.4000000000000000
-99.819999999999999 0
0 0
-29.0800000000000000 0
-14.0000000000000000 0
0 -5.0000000000000000

```

```
> B
```

```
B =
```

```

0 0
0 0
0 0
0 0
1 0
0 1

```

```
> C
```

```
C =
```

```

1 0 0 0 0 0
0 1 0 0 0 0
0 0 10 0 0 0
0 0 0 10 0 0

```

```
> D
```

```
D =
```

```

0 0
0 0
0 0
0 0

```

The feedback gain matrix is:

```
> K
```

```
K =
```

```
1.0e+03 *
```

```
Columns 1 through 4
```

```

-0.00462068698884 0.15627460414694 -3.73730539775403 -0.74332871045858
0.02319409781308 -0.03373621660596 0.56616316694374 0.13205856948485

```

```
Columns 5 through 6
```

```

0.14563795505610 -0.00323771613605
-0.00323771613629 0.08519100075932

```

The control channel selected for first simulation is the elevator command:

```
> iu
iu =
  1
```

The selected simulation time is $T_f = 2$ seconds:

```
> Tf
Tf =
  2
```

Now call the simulation routine:

```
> cmprdsn(A,B,C,D,K,iu,Tf)
> pack
```

Now select the throttle command channel for the response simulation:

```
> iu=2
iu =
  2
```

```
> cmprdsn(A,B,C,D,K,iu,Tf)
```

The simulations are complete.

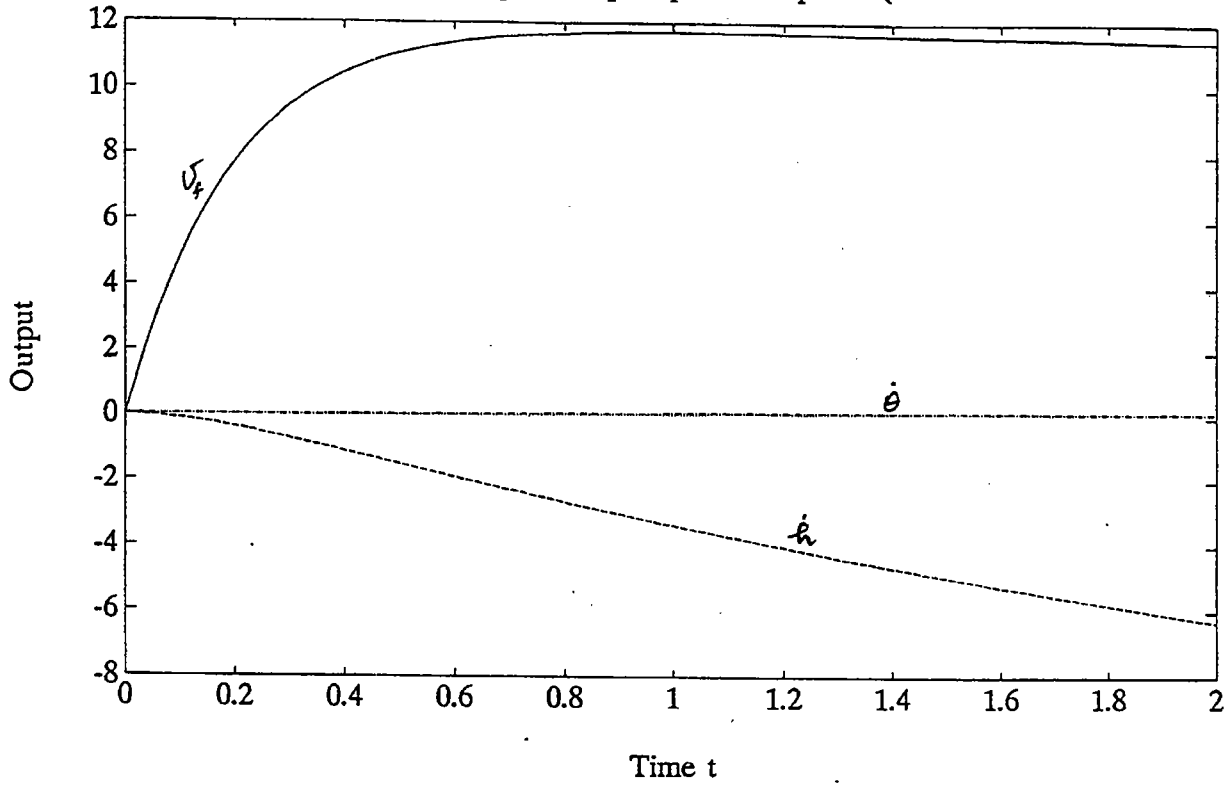
```
> quit
989405 flop(s).
```

```

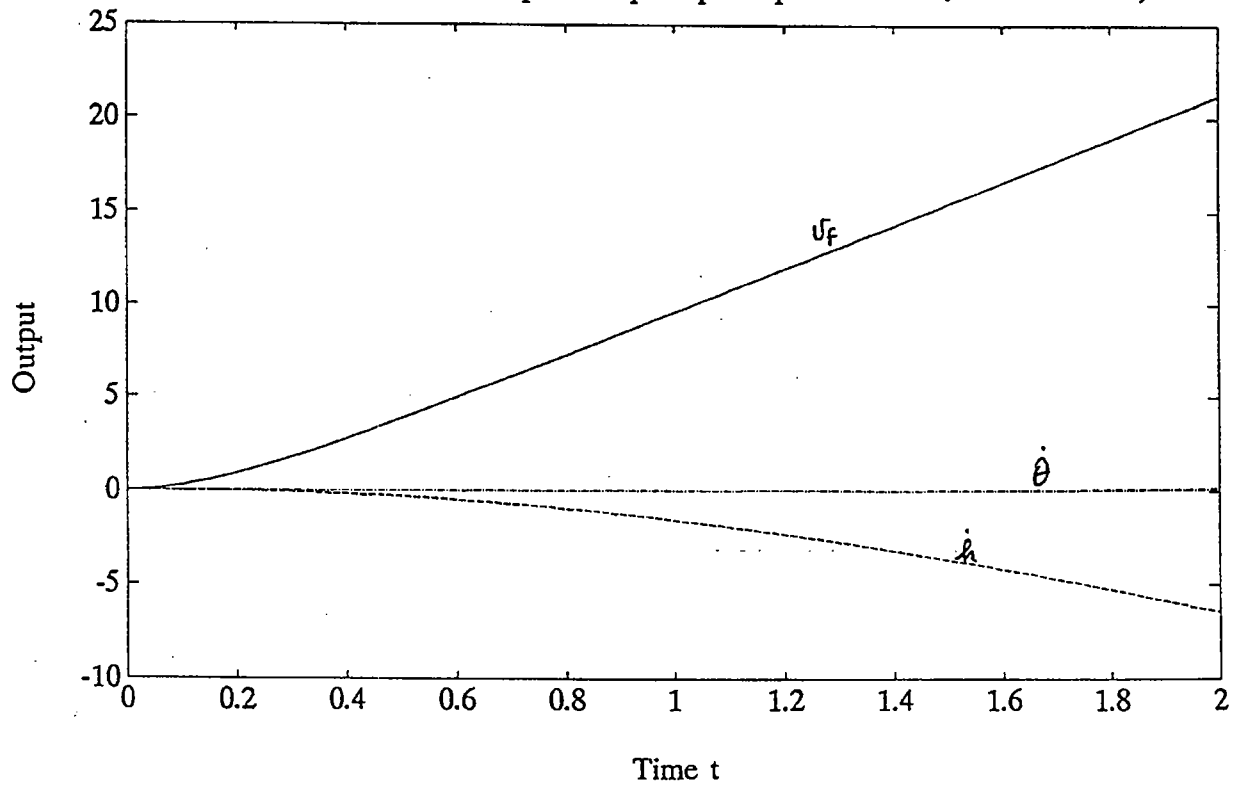
function cmprdsgn(A,b,c,d,k,iu,Tf)
Acl=A-b*k;dt=Tf/100.0;
t=0: dt :Tf;
y1=step(A,b,c,d,iu,t);
y2=step(Acl,b,c,d,iu,t);
y3=impulse(A,b,c,d,iu,t);
y4=impulse(Acl,b,c,d,iu,t);
plot(t,y1,t,y2),title('Open- vs Closed Loop Step Response');
xlabel('Time t'),ylabel('Output');meta oclrsp;
pause;
plot(t,y3,t,y4),title('Open- vs Closed Loop Impulse Responses');
xlabel('Time t'),ylabel('Output');meta oclrsp;
pause;
plot(t,y3),title('Open-Loop Impulse Response');
xlabel('Time t'),ylabel('Output');meta olrsp;
pause;
plot(t,y4),title('Closed-Loop Impulse Response');
xlabel('Time t'),ylabel('Output');meta clrsp;
pause;
plot(t,y1),title('Open-Loop Step Response');
xlabel('Time t'),ylabel('Output');meta olrsp;
pause;
plot(t,y2),title('Closed-Loop Step Response');
xlabel('Time t'),ylabel('Output');meta clrsp;
pause;
w=logspace(-1,1);
[mag1,phase1]=bode(A,b,c,d,iu,w);
[mag2,phase2]=bode(Acl,b,c,d,iu,w);
[re1,im1]=nyquist(A,b,c,d,iu,w);
[re2,im2]=nyquist(Acl,b,c,d,iu,w);
disp('Open-Loop Gain & Phase Margins & Crossover Frequencies');
[Gm1,Pm1,Wcg1,Wcp1]=margin(mag1,phase1,w);
pause;
disp('Closed-Loop Gain & Phase Margins & Crossover Frequencies');
[Gm2,Pm2,Wcg2,Wcp2]=margin(mag2,phase2,w);
pause;
loglog(w,mag1),title('Open-Loop Magnitude Response');
pause;
print;
semilogx(w,phase1),title('Open-Loop Phase Response');
pause;
print;
semilogy(mag1,phase1),title('Open-Loop Nichols Plot');
pause;
print;
loglog(w,mag2),title('Closed-Loop Magnitude Response');
pause;
print;
semilogx(w,phase2),title('Closed-Loop Phase Response');
pause;
print;
semilogy(mag2,phase2),title('Closed-Loop Nichols Plot');
pause;
print;
plot(re1,im1),title('Open-Loop Nyquist Plot');
pause;print;
plot(re2,im2),title('Closed-Loop Nyquist Plot');
pause;print;end

```

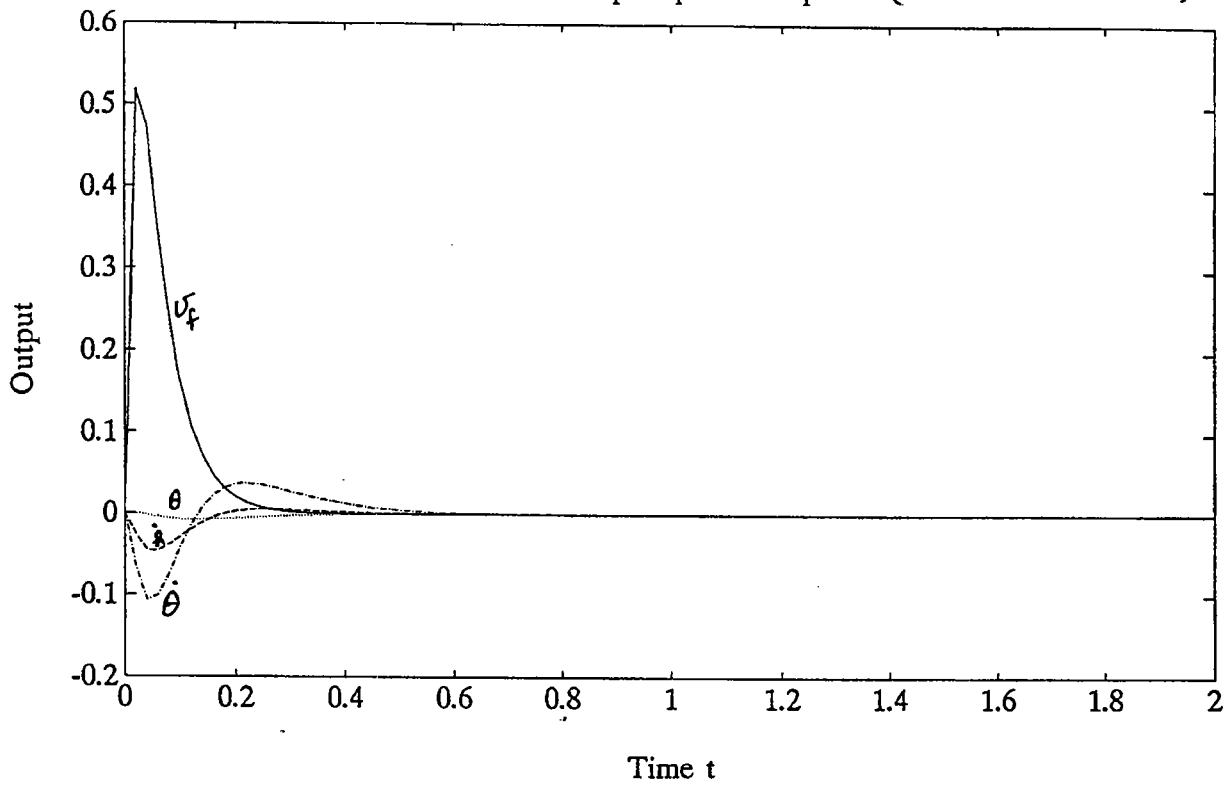
Open-Loop Impulse Response (Throttle Command)



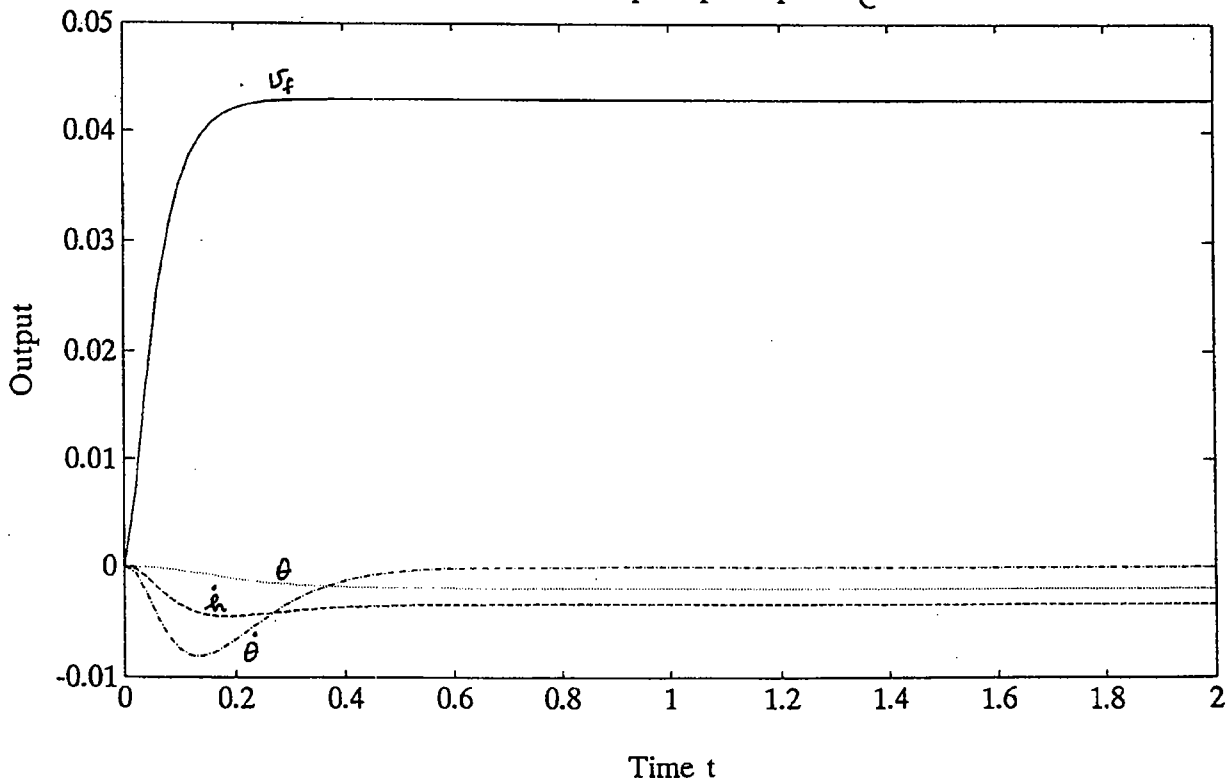
Open-Loop Step Response (Throttle Command)



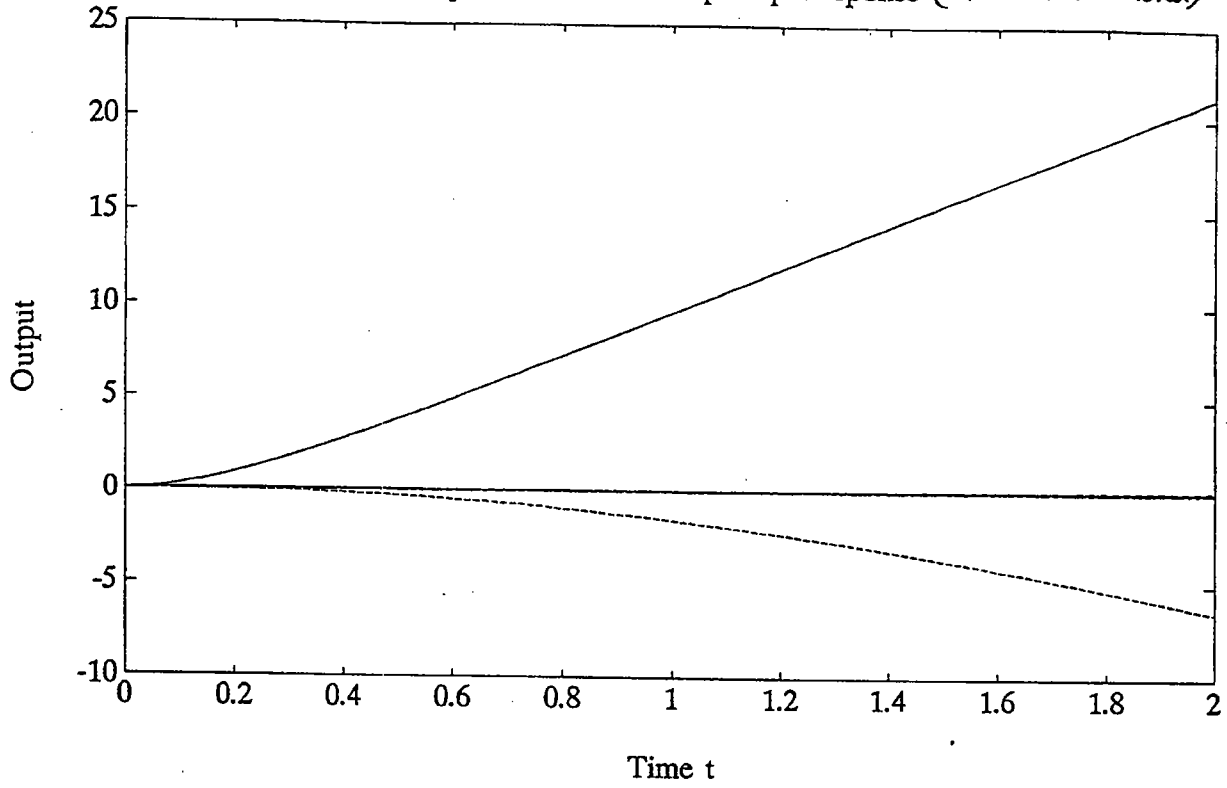
Closed-Loop Impulse Response (Throttle Command)



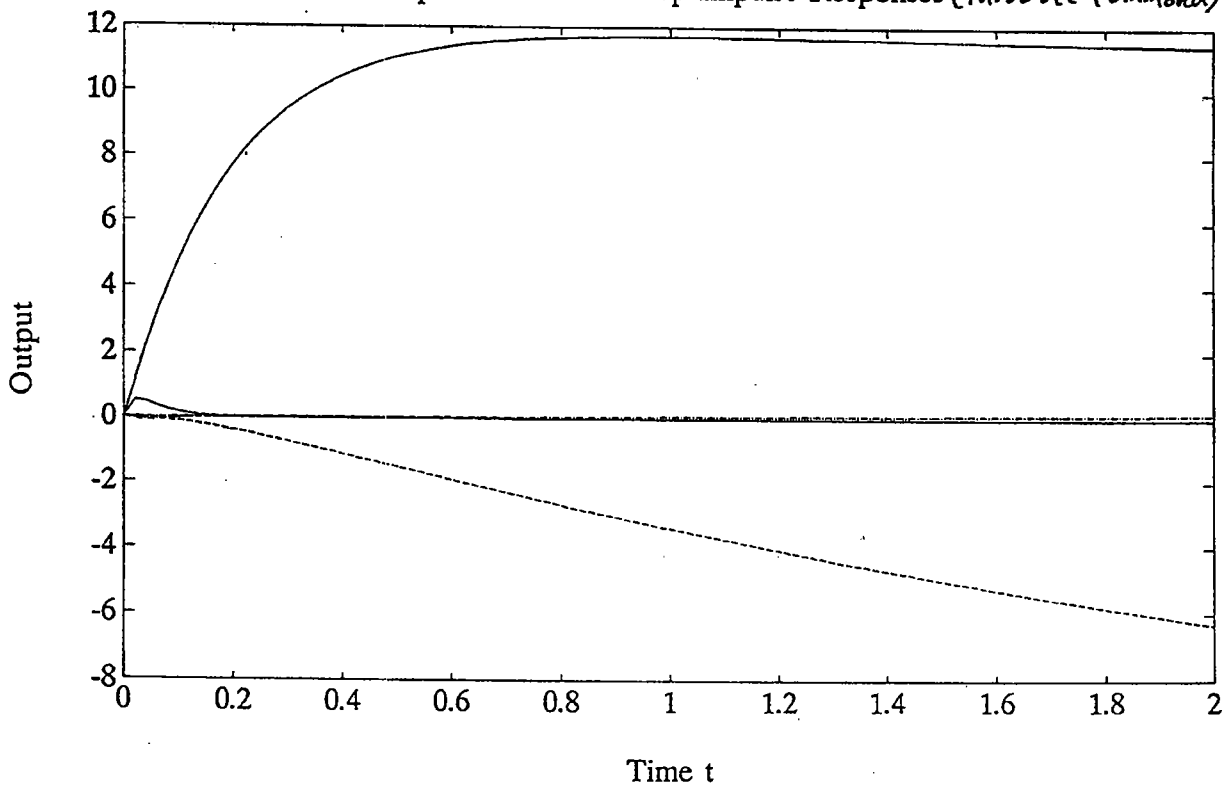
Closed-Loop Step Response (Throttle Command)



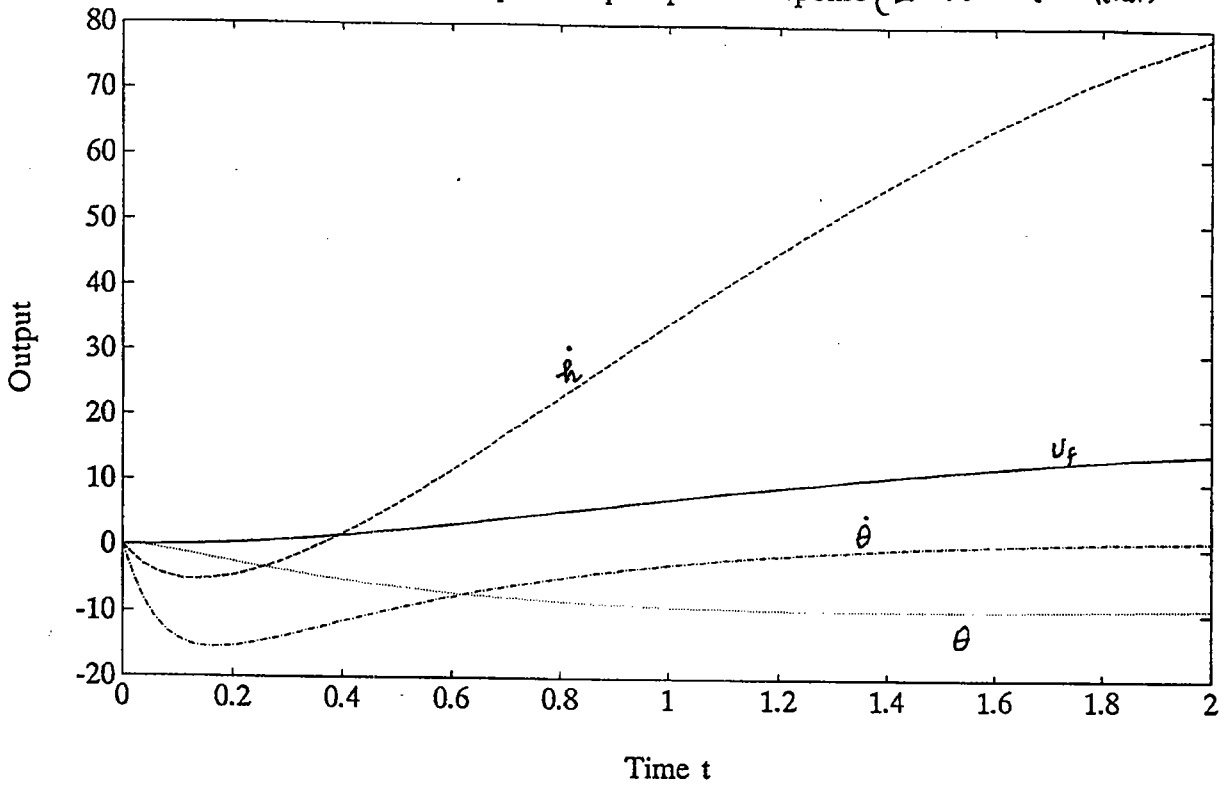
Open- vs Closed Loop Step Response (Throttle Command)



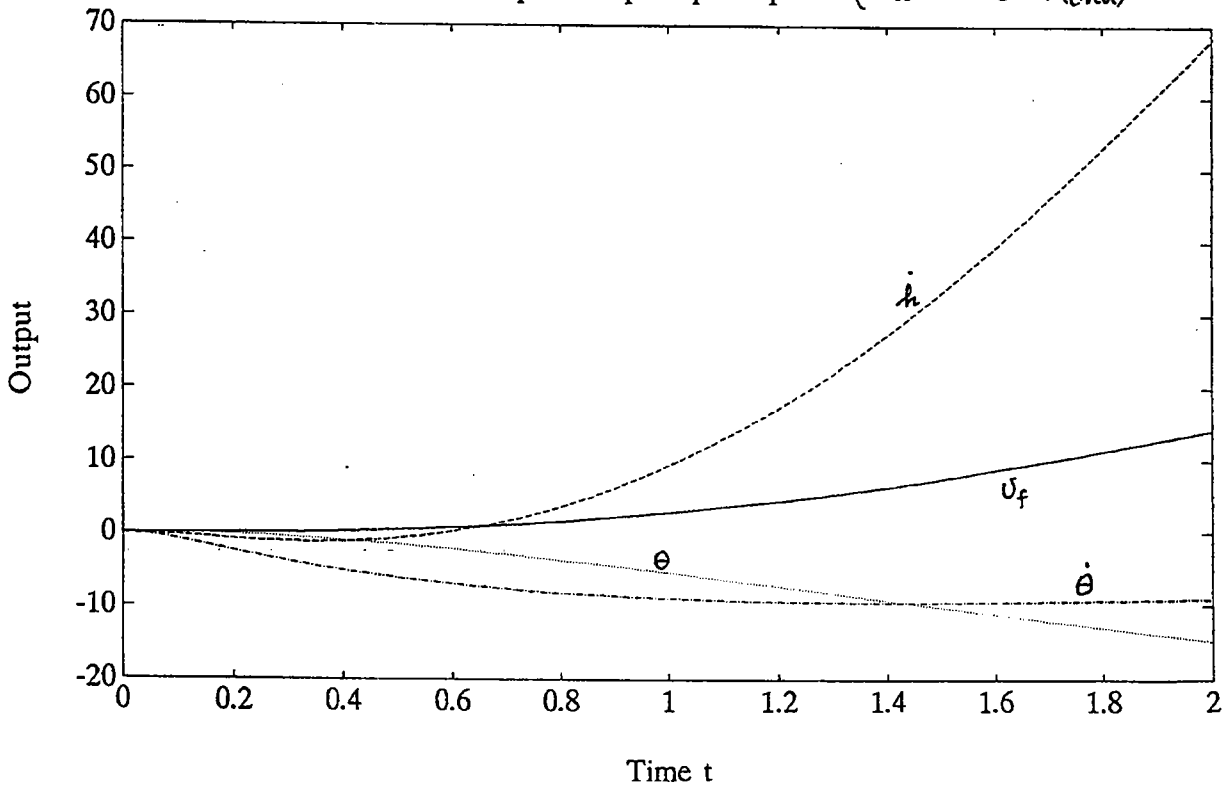
Open- vs Closed Loop Impulse Responses (Throttle Command)



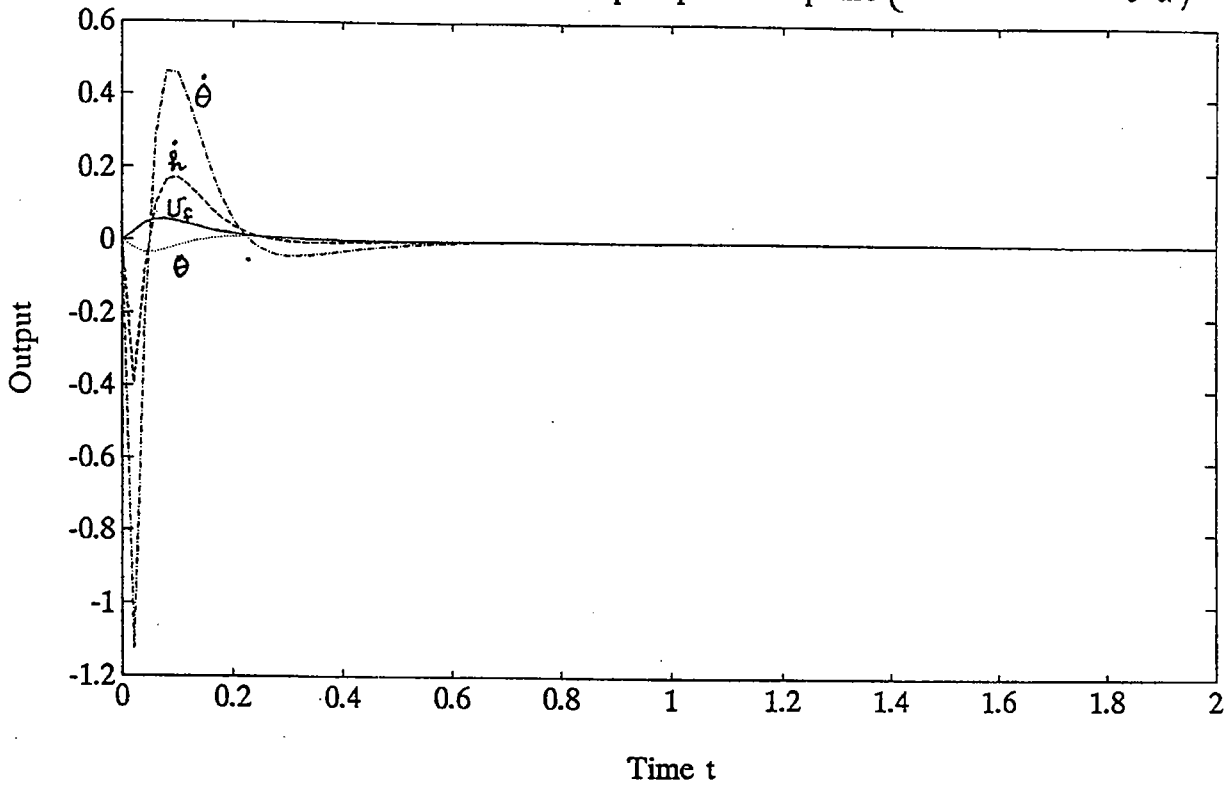
Open-Loop Impulse Response (Elevator Command)



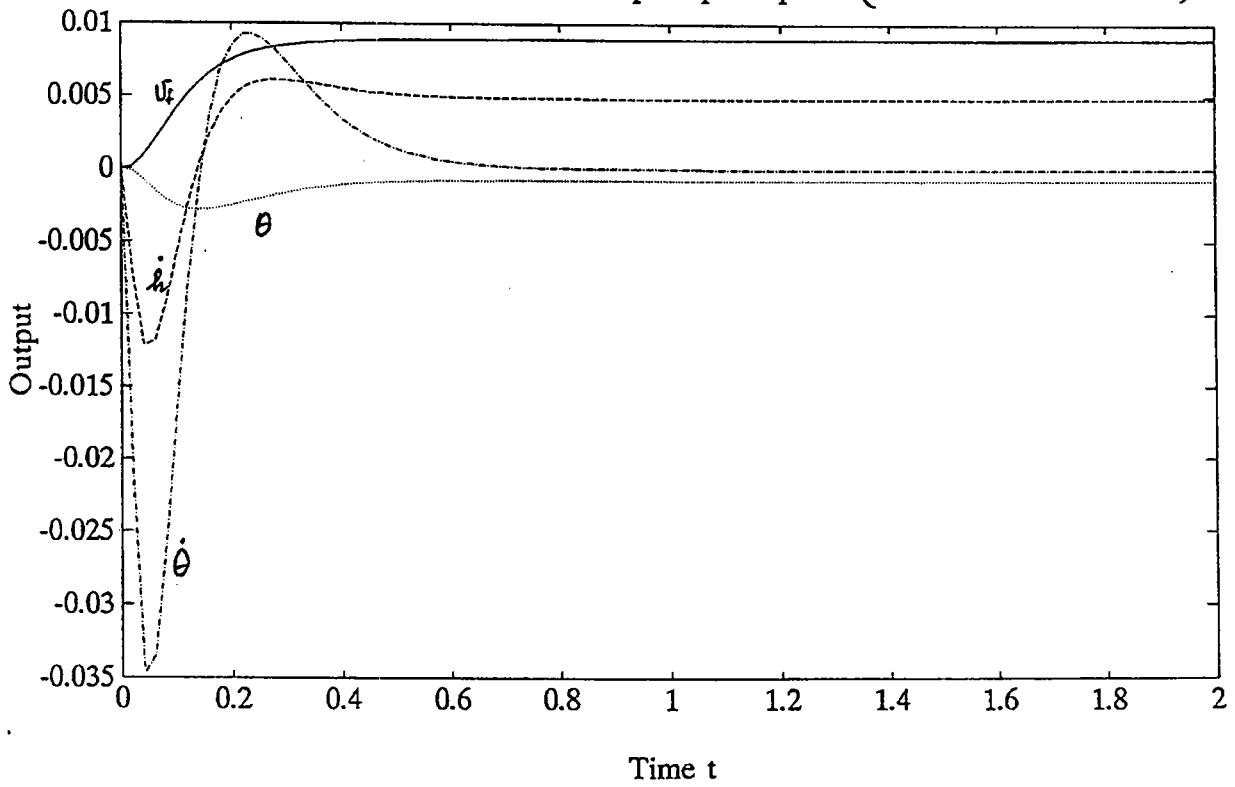
Open-Loop Step Response (Elevator Command)



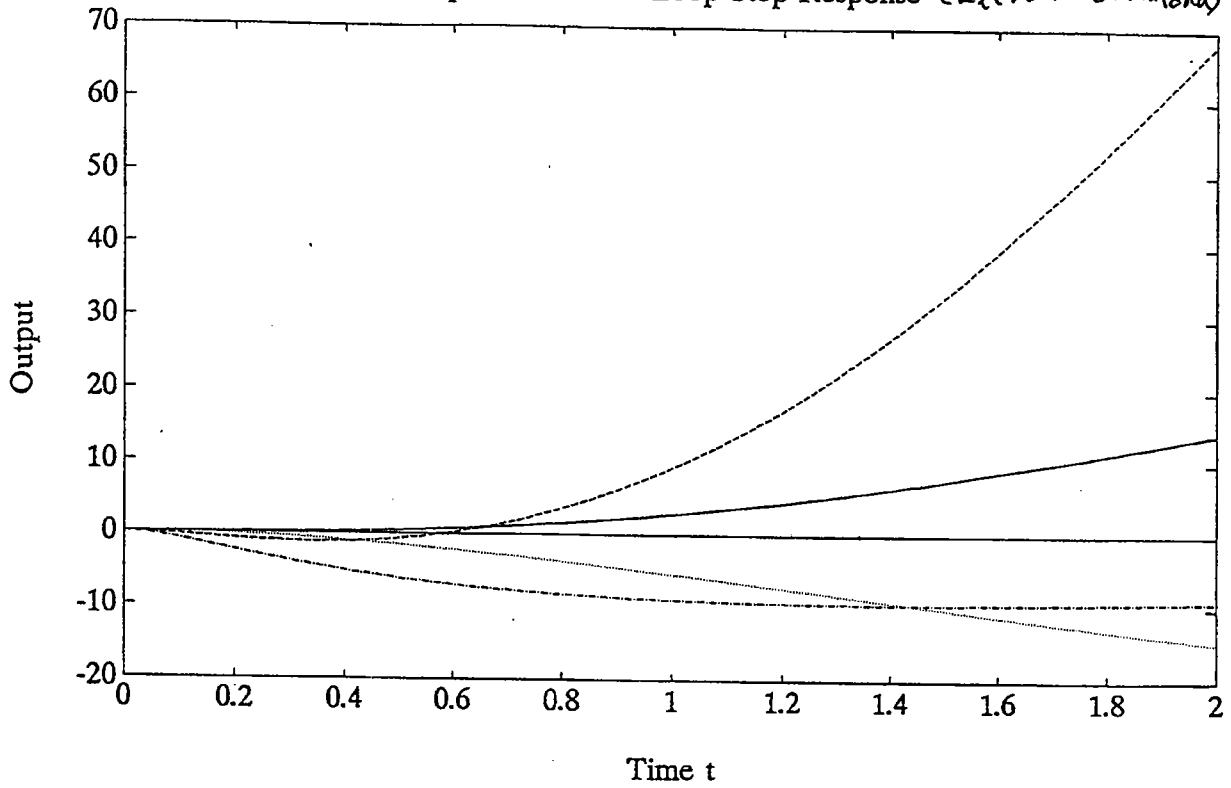
Closed-Loop Impulse Response (Elevator Command)



Closed-Loop Step Response (Elevator Command)



Open- vs Closed Loop Step Response (Elevator Command)



Open- vs Closed Loop Impulse Responses (Elevator Command)

