References:


75. [http://www.anft.net/f-14/f14-detail-dimensions-01.htm](http://www.anft.net/f-14/f14-detail-dimensions-01.htm)

76. [http://www.anft.net/f-14/f14-detail-dimensions.htm](http://www.anft.net/f-14/f14-detail-dimensions.htm)

77. [http://www.combataircraft.com/aircraft/ff14_p_04_1.jpg](http://www.combataircraft.com/aircraft/ff14_p_04_1.jpg)

Appendix A

1. Least Effort controller methodology M-file:

%%%Dissertation: Pitch and Heave Control, 2014

%%% finding open loop heave rate due to control surface on wing and tail
wing_heave=[63.9979 4769.161907];
den=[1 1.2956 4.4867];
sys1=tf(wing_heave,den); %Given Transfer function in Whalley &Ebrahimi,2000)

tail_heave=[2 149.02412]; %heave response to suggested input on tail
sys2=tf(tail_heave,den);

%%% finding open loop pitch rate due to control surface on wing and tail
wing_pitch=[-6.8847 -4.77316];
sys3=tf(wing_pitch,den); %Given Transfer function in (Whalley &Ebrahimi,2000)

tail_pitch=[-100 -69.33];
sys4=tf(tail_pitch,den);

%%%Optimization
clc
syms n;
format shorteng
Q1=[wing_heave(1,2)+(tail_heave(1,2)*(n))
wing_pitch(1,2)+(tail_pitch(1,2)*(n));wing_heave(1,1)+(tail_heave(1,1)*(n))
wing_pitch(1,1)+(tail_pitch(1,1)*n)];
k1=50;
b=k1*[6;1]

%%%Find minimum J
J=(1+(n^2))*transpose(b)*transpose(inv(Q1))*(inv(Q1))*b;
[N,D]=numden(J);
NUM=sym2poly(N);
DEN=sym2poly(D);
[q,d]=polyder(NUM,DEN);
J=tf(NUM,DEN);

%%%Plot J
n=-100:0.01:100;
um=[NUM(1,1).*n.^4+NUM(1,2).*n.^3+NUM(1,3).*n.^2+NUM(1,4).*n.+1+NUM(1,5)];
den=[DEN(1,1).*n.^4+DEN(1,2).*n.^3+DEN(1,3).*n.^2+DEN(1,4).*n.+1+DEN(1,5)];
X=num./den;
a=tf(q,d);
P1=roots(q)

%finding J at n1
cl=(P1(1,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(cl)
wing_pitch(1,2)+tail_pitch(1,2)*(cl);wing_heave(1,1)+tail_heave(1,1)*(cl)
wing_pitch(1,1)+tail_pitch(1,1)*cl];
J10=(1+(cl)^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

%finding J at n2
c2=(P1(2,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(c2)
wing_pitch(1,2)+tail_pitch(1,2)*(c2);wing_heave(1,1)+tail_heave(1,1)*(c2)
wing_pitch(1,1)+tail_pitch(1,1)*c2];
J20=(1+(c2)^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

%finding J at n3
c3=(P1(3,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(c3)
wing_pitch(1,2)+tail_pitch(1,2)*(c3);wing_heave(1,1)+tail_heave(1,1)*(c3)
wing_pitch(1,1)+tail_pitch(1,1)*c3];
J30=(1+c3^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

%finding J at n4
c4=(P1(4,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(c4)
wing_pitch(1,2)+tail_pitch(1,2)*(c4);wing_heave(1,1)+tail_heave(1,1)*(c4)
wing_pitch(1,1)+tail_pitch(1,1)*c4];
J40=(1+c4^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

%finding J at n5
c5=(P1(5,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(c5)
wing_pitch(1,2)+tail_pitch(1,2)*(c5);wing_heave(1,1)+tail_heave(1,1)*(c5)
wing_pitch(1,1)+tail_pitch(1,1)*c5];
J50=(1+c5^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

%finding J at n6
c6=(P1(6,1))
Q=[wing_heave(1,2)+tail_heave(1,2)*(c6) wing_pitch(1,2)+tail_pitch(1,2)*(c6)
wing_heave(1,1)+tail_heave(1,1)*(c6) wing_pitch(1,1)+tail_pitch(1,1)*c6];
J60=(1+c6^2)*transpose(b)*transpose(inv(Q))*(inv(Q))*b;

J=[J10;J20;J30;J40;J50;J60]

%J minimum after simulation is at gain ratio n2
n1=c2;
Q=[wing_heave(1,2)+tail_heave(1,2)*(n1)
wing_pitch(1,2)+tail_pitch(1,2)*(n1);wing_heave(1,1)+tail_heave(1,1)*(n1)
wing_pitch(1,1)+tail_pitch(1,1)*n1];

h=(inv(Q)*b)' %Feedback marix
k=[1;n1];
I=[1 0;0 1];
f=0.8  %outer loop feeback gain
G=[wing_heave(1,2) tail_heave(1,2); wing_pitch(1,2) tail_pitch(1,2)]/x;
F=[f 0; 0 f];
S=[1 0.005;-.005 1];
F=[f 0; 0 f];
P=(inv(G)+k*(h))*S*inv(I-F*S)
H=inv(P)*k*h+F;
y=inv(I+G*P*H)*G*P
dis=inv(I+G*P*H)
a=P*H;
g11=sys1*a(1,1)+sys2*a(2,1);
g12=sys1*a(1,2)+sys2*a(2,2);
g21=sys3*a(1,1)+sys4*a(2,1);
g22=sys3*a(1,2)+sys4*a(2,2);
sys=[sys1 sys2; sys3 sys4];
Gss=[g11 g12; g21 g22];
z=inv(I+Gss)*sys*P;
z1=inv(I+Gss);

2. Inverse Nyquist Array methodology m.file:

G=tf([14.96 95150 95150*1.898];85.2 [124000 124000*2.037]),[[1 12 20] [1,0000 103.2250 252.5000]; [1 12 20] [1,0000 103.2250 252.5000]]); % Transfer Function Matrix
k1=[1000 0; 0 1];% first precompensator
k2=tf([-1.57 -1.57*20.6] [1.205 1.205*11.6]; [1.08 1.08*146.3] [-0.189 -0.189*101.4]);[[1 158.5] 1 158.5];[1 158.5];[1 158.5];%2nd Compensator
Q=inv(G*k1*k2); % Inversed Overall Transfer Function Matrix
gershband(Q); % creates Figures 4.11 and 4.12
function gershband(a,b,c,d,e)
%GERSHBAND - Finds the Gershorin Bands of a nxn LTI MIMO SYS model
% The use of the Gershgorin Bands along the Nyquist plot is helpful for finding the coupling grade of a MIMO system.
%
% Syntax: gershband(SYS) - computes the Gershgorin bands of SYS
% gershband(SYS,'v') - computes the Gershgorin bands and the Nyquist array of SYS
% Inputs:
% SYS - LTI MIMO system, either in State Space or Transfer Function representation.
%
% Example:
% g11=tf(2,[1 3 2]);
% g12=tf(0.1,[1 1]);
% g21=tf(0.1,[1 2 1]);
% g22=tf(6,[1 5 6]);
% G=[g11 g12; g21 g22];
% gershband(G);
%
% Other m-files required: sym2tf, ss2sym
% Subfunctions: center, radio
% See also: rga
%
% Author: Oskar Vivero Osornio
% email: oskar.vivero@gmail.com
% Created: February 2006;
% Last revision: 11-May-2006;
% May be distributed freely for non-commercial use,
% but please leave the above info unchanged, for
% credit and feedback purposes
%--------------------- BEGIN CODE ---------------------%---------------------

---

Determines Syntax

\[ n = \text{nargin}; \]

\text{switch} \ n
case 1
% Transfer Function Syntax
\text{switch} \ \text{class}(a)
case 'tf'
% Numeric Transfer Function Syntax
\text{g} = a;
case 'sym'
% Symbolic Transfer Function Syntax
\text{g} = \text{sym2tf}(a);
end
e = 0;
case 2
% Transfer Function Syntax with Nyquist Array
\text{switch} \ \text{class}(a)
case 'tf'
% Numeric Transfer Function Syntax
\text{g} = a;
case 'sym'
% Symbolic Transfer Function Syntax
\text{g} = \text{sym2tf}(a);
end
e = 1;
case 4
% State Space Syntax
\text{g} = \text{ss2sym}(a,b,c,d);
\text{g} = \text{sym2tf}(g);
e = 0;
case 5
% State Space Syntax
\text{g} = \text{ss2sym}(a,b,c,d);
\text{g} = \text{sym2tf}(g);
e = 1;
end
%-----------------------------------------------------------------
\[ [n,m] = \text{size}(g); \]
\text{w} = \text{logspace}(-1,6,200);
\text{q} = 0:(\pi/50):(2\pi);
\text{for} \ i = 1:n
\text{for} \ j = 1:m
\text{if} \ i == j
\text{figure}(i)
\text{nyquist}(g(i,i));
\text{grid} \ \text{on}
title(['Nyquist Diagram of G(',num2str(i),',',num2str(j),')'])
\text{for} \ iest = 1:n
\text{for} \ jest = 1:m
\text{if} \ iest == jest
\text{hold} \ \text{on}

---
C=\text{center}(g(i,j), w);
R=\text{radio}(g(\text{iest}, \text{jest}), w);
\text{for } k=1: \text{length}(C)\text{ plot}(\{R(k) \times \cos(q)\} + \text{real}(C(k)), \{R(k) \times \sin(q)\} + \text{imag}(C(k)), 'g-')\text{ end}\text{ hold off end end end end end end}
if e==1
\text{figure}(n+1)\text{ nyquist}(g)\text{ grid on end}
\%\%----------- Subfunction -----------
\text{function } C = \text{center}(g, w)\text{ g=tfsym}(g)\text{ C=subs}\{g, \text{complex}(0, w)\}\text{ function } R = \text{radio}(g, w)\text{ g=tfsym}(g)\text{ R=abs(subs}\{g, \text{complex}(0, w)\})\text{ %---------- END OF CODE ----------}
3. Simulation Models

![Simulation Model Diagram]

**Figure A.1** Closed loop by least effort controller with $f = 0.8$ Simulation model

![Simulation Model Diagram]

**Figure A.2** Closed loop by Inverse Nyquist Array Controller Simulation Model