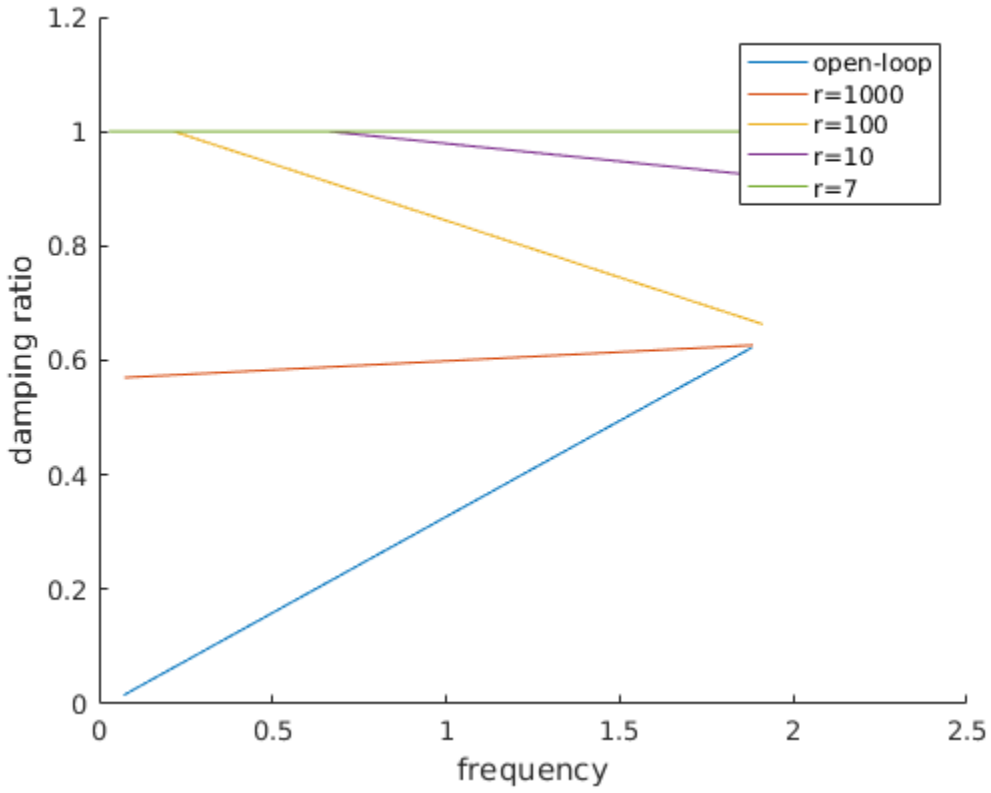
Table of Contents

part a)

```
clear; close all; clc;
load('f16_long.mat');

% normalizing Q, R using Bryson's rule
R1 = [1/(5^2) 0; 0 1/(deg2rad(25)^2)];
Q1 = [1/(500^2) 0 0 0;
      0 1/(deg2rad(2.3)^2) 0 0;
      0 0 1/(deg2rad(17.2)^2) 0;
      0 0 0 1/(deg2rad(0.5)^2)];

C = eye(4);
D = zeros(4,2);
% open-loop system
sys = ss(Along,Blong,C,D);
[Wn, zeta] = damp(sys);
figure(1); hold on;
plot(Wn, zeta);
ylabel('damping ratio');
xlabel('frequency');

% closed-loop system
r = 1000;
[K,S,E] = lqr(Along, Blong, Q1, r*R1, 0);
sys = ss(Along-Blong*K, zeros(4,2), C, D);
[Wn, zeta] = damp(sys);
plot(Wn, zeta);

r = 100;
[K,S,E] = lqr(Along, Blong, Q1, r*R1, 0);
sys = ss(Along-Blong*K, zeros(4,2), C, D);
[Wn, zeta] = damp(sys);
plot(Wn, zeta);

r = 10;
[K,S,E] = lqr(Along, Blong, Q1, r*R1, 0);
sys = ss(Along-Blong*K, zeros(4,2), C, D);
[Wn, zeta] = damp(sys);
plot(Wn, zeta);

r = 7;
[K,S,E] = lqr(Along, Blong, Q1, r*R1, 0);
sys = ss(Along-Blong*K, zeros(4,2), C, D);
```

```
[Wn, zeta] = damp(sys);
plot(Wn, zeta);
ylim([0 1.2]);
legend('open-loop', 'r=1000', 'r=100', 'r=10', 'r=7');
```

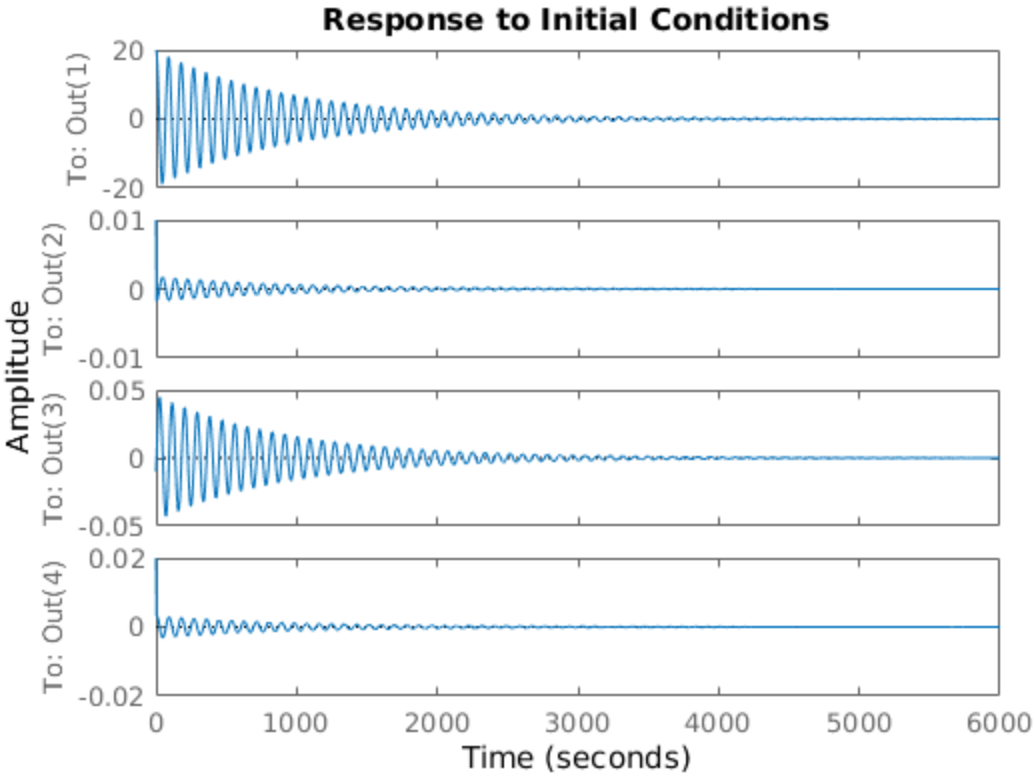


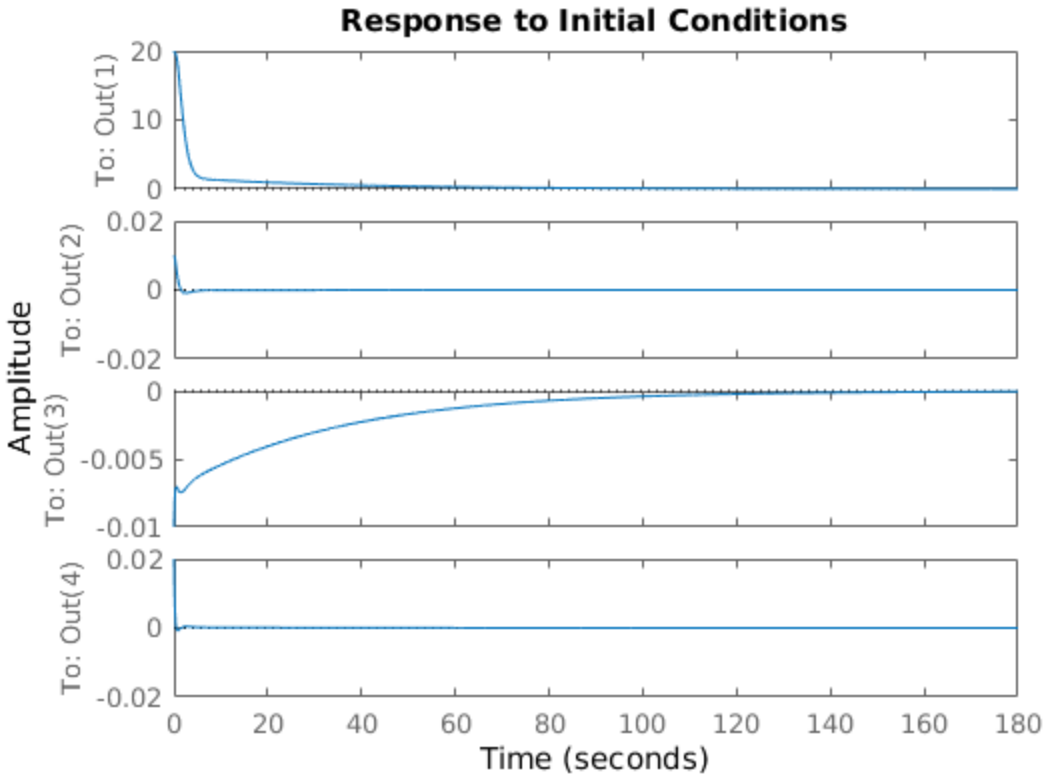
Part b)

open-loop system

```
figure(2);
sys = ss(Along,Blong,C,D);
x0 = [20; 0.01; -0.01; 0.02];
initial(sys, x0);

% closed-loop system
r = 3;
[K,S,E] = lqr(Along, Blong, Q1, r*R1, 0);
sys = ss(Along-Blong*K, zeros(4,2), C, D);
figure(3);
initial(sys, x0);
[y,t,x] = initial(sys, x0);
input_u = -K*x';

% with several tuning iteration, it turns out that r=3 is the value
 that
% maximizes the use of the inputs.
```

```
% making sure that throttle and elevator are not over their limits
if sum(abs(input_u(1,:))>5)>=1
    display('throttle exceeds its bound');
end
if sum(abs(input_u(2,:))>deg2rad(25))>=1
    display('elevator input exceeds its bound');
end
```





Part c)

```
disturbance = eye(2)*10^(-4);
W_d = Blong*disturbance*Blong'; % process noise
W_n = [1 0; 0 10^(-5)];     % sensor noise

C = [1 0 0 0; 0 -1 1 0];
D = zeros(2,2);
L = lqr(Along', C', W_d, W_n);

syms s
compensator = K*inv(s*eye(4)-Along+L'*C+Blong*K)*L';
% this compensator looks very wrong. Thus, I didn't print them out.

% % plant
% sys = ss(Along, Blong, C, D);
% [y,t,x] = initial(sys,x0);
% figure(4);
% % plot for airspeed change
% plot(t,y(:,1));
% % plot for flight path angle
% figure(5);
% plot(t,y(:,2));
```

Published with MATLAB® R2016a