[AE450] Scoring rubric for the Homework Assignment #2

DISCLAIMER:

> 1. This guideline may contain some typo and errors.
> 2. Although we use MATLAB and/or Simulink to do some dirty works, it is highly recommended to see the behind of the curtain.
> 3. There may several other manageable approaches, solutions, and answers to a single problem. Therefore, all reasonable solutions may take full credits.
> 4. Productive cooperations and discussions are highly welcomed. However, if your answers and submissions are found to be directly shared between other classmates, there will be a major deduction on you scores. To take full credits, you need to contact and justify yourself to T.A. in charge of.
> 5. If you run into any troubles regarding the assignment #2 (error reporting, claims, etc.), take the contacts below.
>
> Assistance: Seongheon Lee (skynspace@kaist.ac.kr)
>
> - Transferred from Kiduck Kim (kdkim@ascl.kaist.ac.kr), due to the military service [2018/11/8~2018/12/6]

```
>> NOTICE
% Shaded area describes a programming code snippet or outputs from the programming code.
% All codes and scripts are written in MATLAB.
% You may need to install some libraries regarding control theories (e.g. control system toolbox).
% Scripts are also available @[https://github.com/SKYnSPACE/AE450HW2]
% Take the hyperlink below.
```

https://github.com/SKYnSPACE/AE450HW2

- **MAJOR SCORING CRITERIA (ACCUMULATIVE!)**

Criteria	Points	Notes
Submissions exceeded the final due(14.NOV.2018).	-40%	Discounted from the total score you get.
Submissions without relevant reports and/or documents.	-30%	Discounted from the total score you get.
Submissions without source codes, or scripts.	-20%	Discounted from the total score you get.
Missing references.	-10%	Discounted from the total score you get.

- **ALL-OR-NOTHING**

  This is a step-by-step assignment. You cannot complete the next part correctly while leaving the previous parts incorrect. Therefore, you'll get full credits for the completed parts, but there will be no points at all for the missing or incomplete parts.

- **[VERSION CONTROL]** [Ver. 1.0 @30/NOV/2018] Published the first edition.

  [Ver. 1.1 @9/DEC/2018] Correction made on the Figure 6.2. (The xlabel, and ylabel changed.)

NOMENCLATURE

> Please refer to the textbook *"Flight Dynamics Principles: A linear Systems Approach to Aircraft Stability and Control"*[1] by *Michael V. Cook* for the nomenclature. Some important notations are listed below.

- Approximate Expressions for the Dimensionless Longitudinal Aerodynamics Stability Derivatives (*Cook, Appendix 8*)

Derivative	Description	Expression	Comments
X_u	Axial force due to veolcity	-	Drag and thrust effects due to velocity perturbation
X_w	Axial force due to "incidence"	-	Lift and drag effects due to incidence perturbation
X_q	Axial force due to pitch rate	-	Tailplane drag effect, usually negligible
$X_{\dot{w}}$	Axial force due to downwash lag	-	Tailplane drag due to downwash lag effect
Z_u	Normal force due to velocity	-	Lift effects due to velocity perturbation
Z_w	Normal force due to "incidence"	-	Lift and drag effects due to incidence perturbation
Z_q	Normal force due to pitch rate	-	Tailplane lift effect
$Z_{\dot{w}}$	Normal force due to downwash lag	-	Tailplane lift due to downwash lag effect (added mass effect)
M_u	Pitching moment due to velocity	-	Mach dependent, small at low speed
M_w	Pitching moment due to "incidence"	-	Pitch stiffness, dependent on static margin
M_q	Pitching moment due to pitch rate	-	Pitch damping, due mainly to tailplane
$M_{\dot{w}}$	Pitching moment due to downwash lag	-	Pitch damping due to downwash lag effect at tailplane

TASK: AIRCRAFT LONGITUDINAL CONTROL

> Your task is to design longitudinal controller for an aircraft. The dynamics model can be found from any textbook or web-site. There are plenty of it.

Part 1. DYNAMIC MODEL OF A SYSTEM (15pts.)

> Please find out any linearized model of longitudinal motion of an aircraft from any textbook or web-site. The system dynamics shall be given in the form. Please refer to lecture note for details.
>
> $$\dot{\vec{\mathbf{x}}} = A\vec{\mathbf{x}} + B\vec{\mathbf{u}},$$
> $$\text{where, } \vec{\mathbf{x}} = [u, w, q, \theta]^T.$$

1.1. Longitudinal Motion of an Aircraft

- Dimensional form:

$$\begin{bmatrix} m & -\mathring{X}_{\dot{w}} & 0 & 0 \\ 0 & m - \mathring{Z}_{\dot{w}} & 0 & 0 \\ 0 & -\mathring{M}_{\dot{w}} & I_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \mathring{X}_u & \mathring{X}_w & \mathring{X}_q - mW_e & -mg\cos\theta_e \\ \mathring{Z}_u & \mathring{Z}_w & \mathring{Z}_q + mU_e & -mg\sin\theta_e \\ \mathring{M}_u & \mathring{M}_w & \mathring{M}_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} \mathring{X}_\eta & \mathring{X}_\tau \\ \mathring{Z}_\eta & \mathring{Z}_\tau \\ \mathring{M}_\eta & \mathring{M}_\tau \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \tau \end{bmatrix} \quad \text{(Cook, 4.65)}$$

- Non-dimensional form:

$$M\dot{\vec{\mathbf{x}}} = A'\vec{\mathbf{x}} + B'\vec{\mathbf{u}}$$

$$\begin{bmatrix} m' & -\frac{X_{\dot{w}}\bar{\bar{c}}}{V_0} & 0 & 0 \\ 0 & m' - \frac{Z_{\dot{w}}\bar{\bar{c}}}{V_0} & 0 & 0 \\ 0 & -\frac{M_{\dot{w}}\bar{\bar{c}}}{V_0} & I'_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & X_q\bar{\bar{c}} - m'W_e & -m'g\cos\theta_e \\ Z_u & Z_w & Z_q\bar{\bar{c}} + m'U_e & -m'g\sin\theta_e \\ M_u & M_w & M_q\bar{\bar{c}} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} V_0X_\eta & V_0X_\tau \\ V_0Z_\eta & V_0Z_\tau \\ V_0M_\eta & V_0M_\tau \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \tau \end{bmatrix} \quad \text{(Cook, EXAMPLE 4.3)}$$

$$\text{where, } m' = \frac{m}{\frac{1}{2}\rho V_0 S}, \text{ and } I'_y = \frac{I_y}{\frac{1}{2}\rho V_0 S\bar{\bar{c}}}$$

- Summarized form:

$$\dot{\vec{\mathbf{x}}} = A\vec{\mathbf{x}} + B\vec{\mathbf{u}}$$

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_u & x_w & x_q & x_\theta \\ z_u & z_w & z_q & z_\theta \\ m_u & m_w & m_q & m_\theta \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} x_\eta & x_\tau \\ z_\eta & z_\tau \\ m_\eta & m_\tau \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \tau \end{bmatrix} \quad \text{(Cook, 4.67)}$$

1.2. Parameters, Aerodynamic coefficients, and Derivatives

> Data of McDonnell F-4C Phantom taken from (Cook, EXAMPLE 4.2)[1]

- Parameters and coefficients

  Flight path angle: $\gamma_e = 0^\circ$

  Body incidence: $\alpha_e = 9.4^\circ$

  Velocity: $V_0 = 178m/s$

  Mass: $m = 17642kg$

  Pitch MoI: $I_{yy} = 165669kg \cdot m^2$

  MAC: $\bar{\bar{c}} = 4.889m$

  Wing area: $S = 49.239m^2$

  Air density: $\rho = 0.3809kg/m^3$

  Gravitational Acc: $g = 9.81m/s^2$

- Aerodynamic derivatives

Motion variables	X	Z	M
u	0.0076	-0.7273	0.0340
w	0.0483	-3.1245	-0.2169
$\dot{w}$	0	-0.3997	-0.5910
q	0	-1.2109	-1.2732
η	0.0618	-0.3741	-0.5581

- Concise form taken from (Cook, EXAMPLE 4.3)[1]:

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 7.181 \times 10^{-4} & 4.570 \times 10^{-3} & -29.072 & -9.678 \\ -0.0687 & -0.2953 & 174.868 & -1.601 \\ 1.73 \times 10^{-3} & -0.0105 & -0.4462 & 1.277 \times 10^{-3} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 1.041 \\ -6.294 \\ -4.888 \\ 0 \end{bmatrix} \cdot \eta$$

1.3. MATLAB Script

```
%% Part 1. DYNAMIC MODEL OF A SYSTEM
disp(repmat('=',1,80));
disp('Part 1. DYNAMIC MODEL OF A SYSTEM');

% Parameters and coefficients
gamma_e = 0;
alpha_e = 9.4;
V_0 = 178;
m = 17642;
I_yy = 165669;
c_barbar = 4.889;
S = 49.239;
rho = 0.3809;
g = 9.81;
D2R = pi/180;

% Derivatives
X_u = 0.0076;       Z_u = -0.7273;      M_u = 0.0340;
X_w = 0.0483;       Z_w = -3.1245;      M_w = -0.2169;
X_wdot = 0.0;       Z_wdot = -0.3997;   M_wdot = -0.5910;
X_q = 0.0;          Z_q = -1.2109;      M_q = -1.2732;
X_eta = 0.0618;     Z_eta = -0.3741;    M_eta = -0.5581;

% Non-dimensional form
mPrime = m/(1/2*rho*V_0*S);
I_yyPrime = I_yy/(1/2*rho*V_0*S*c_barbar);
W_e = V_0 * sin(alpha_e*D2R);
U_e = V_0 * cos(alpha_e*D2R);

M = [mPrime, -X_wdot*c_barbar/V_0, 0, 0 ;
    0, mPrime-Z_wdot*c_barbar/V_0, 0, 0 ;
    0, -M_wdot*c_barbar/V_0, I_yyPrime, 0;
    0, 0, 0, 1];
APrime = [X_u, X_w, X_q*c_barbar-mPrime*W_e, -mPrime*g*cos(alpha_e*D2R);
        Z_u, Z_w, Z_q*c_barbar+mPrime*U_e, -mPrime*g*sin(alpha_e*D2R);
        M_u, M_w, M_q*c_barbar 0;
        0, 0, 1, 0];
BPrime = [V_0*X_eta;
        V_0*Z_eta;
        V_0*M_eta;
        0];

% Summarized form
A = M\APrime
x_u = A(1,1);       z_u = A(2,1);       m_u = A(3,1);
x_w = A(1,2);       z_w = A(2,2);       m_w = A(3,2);
x_q = A(1,3);       z_q = A(2,3);       m_q = A(3,3);
x_theta = A(1,4);   z_theta = A(2,4);   m_theta = A(3,4);
```

```
B = M\BPrime
x_eta = B(1,1);     z_eta = B(2,1);     m_eta = B(3,1);

disp(repmat('=',1,80));
```

1.4. Terminal Outputs

```
================================================================================
Part 1. DYNAMIC MODEL OF A SYSTEM

A =

    0.0007    0.0046  -29.0720   -9.6783
   -0.0687   -0.2953  174.8681   -1.6006
    0.0017   -0.0104   -0.4464    0.0013
         0         0    1.0000         0


B =

    1.0408
   -6.2939
   -4.8885
         0

================================================================================
```

Part 2. TRANSIENT RESPONSE OF THE SYSTEM (10pts.)

> Find out natural frequencies and damping ratio from the given system matrix (A) in **Part 1.**

2.1. The Characteristic Equation

The longitudinal characteristic equation most commonly factorises into two pairs of complex roots, which describe the phugoid and short-period stability modes respectively.

$$(s^2 + 2\zeta_p\omega_p s + \omega_p^2)(s^2 + 2\zeta_s\omega_s s + \omega_s^2) = 0 \qquad \text{(Cook, 4.67)}$$

Eigenvalues of the system matrix(A) are the roots of the characteristic equation (i.e., $|\lambda I - A| = 0$). After sorting the elements from the eigenvalues of the system matrix(A) by their real parts, one can get the natural frequencie and damping ratio of the short-period, and phugoid(lightly damped low-frequency oscillation) as follows:

$$\omega_s = \sqrt{\lambda_1 \lambda_2} \qquad [rad/s]$$
$$\zeta_s = -\frac{\lambda_1 + \lambda_2}{2\omega_s}$$

$$\omega_p = \sqrt{\lambda_3 \lambda_4} \qquad [rad/s]$$
$$\zeta_p = -\frac{\lambda_3 + \lambda_4}{2\omega_p}$$

2.2. MATLAB Script (followed by Section 1.3.)

```
%% Part 2. TRANSIENT RESPONSE OF THE SYSTEM
disp(repmat('=',1,80));
disp('Part 2. TRANSIENT RESPONSE OF THE SYSTEM');

disp('** Eigenvalues **');
lambda = sort(eig(A),'ComparisonMethod','real')

disp('** Short Period **');
omega_s = sqrt(lambda(1)*lambda(2))
zeta_s = -(lambda(1)+lambda(2))/2/omega_s

disp('** Phugoid **');
omega_p = sqrt(lambda(3)*lambda(4))
zeta_p = -(lambda(3)+lambda(4))/2/omega_p

disp(repmat('=',1,80));
```

2.3. Terminal Outputs

```
================================================================================
Part 2. TRANSIENT RESPONSE OF THE SYSTEM
** Eigenvalues **

lambda =

  -0.3634 - 1.3636i
  -0.3634 + 1.3636i
  -0.0071 - 0.0770i
  -0.0071 + 0.0770i

** Short Period **

omega_s =

    1.4112


zeta_s =

    0.2575
```

```
** Phugoid **

omega_p =

    0.0774


zeta_p =

    0.0921

================================================================================
```

Part 3. MODEL APPROXIMATION (10pts.)

> Construct short period approximatie model and phugoid approximate model from **Part 1**. Compare natural frequencies and damping ratio from **Part 1**.

3.1. Reduced-order Longitudinal Models

Longitudinal motions of an aircraft;

$$\dot{\vec{\mathbf{x}}} = A\vec{\mathbf{x}} + B\vec{\mathbf{u}}$$

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_u & x_w & x_q & x_\theta \\ z_u & z_w & z_q & z_\theta \\ m_u & m_w & m_q & m_\theta \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} x_\eta \\ z_\eta \\ m_\eta \\ 0 \end{bmatrix} \eta \qquad \text{(Cook, 4.67)}$$

reduces to the following simplest forms:

- Short period

$$\dot{\overrightarrow{\mathbf{x}_\mathbf{s}}} = A_s\overrightarrow{\mathbf{x}_\mathbf{s}} + B_s\vec{\mathbf{u}}$$

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} z_w & z_q \\ m_w & m_q \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} z_\eta \\ m_\eta \end{bmatrix} \eta \qquad \text{(Cook, 6.15)}$$

One can get the natural frequency and the damping ratio from the eigenvalues of A_s matrix, or the following approximation formula:

$$\begin{aligned} \omega_s &= \sqrt{m_q z_w - m_w U_e} \qquad [rad/s] \\ \zeta_s &= -\frac{m_q + z_w}{2\omega_s} \end{aligned} \qquad \text{(Cook, 6.20)}$$

- Phugoid

$$\dot{\overrightarrow{\mathbf{x}_\mathbf{p}}} = A_p\overrightarrow{\mathbf{x}_\mathbf{p}} + B_p\vec{\mathbf{u}}$$

$$\begin{bmatrix} \dot{u} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_u - x_w\left(\frac{m_u U_e - m_q z_u}{m_w U_e - m_q z_w}\right) & -g \\ \left(\frac{m_u z_w - m_w z_u}{m_w U_e - m_q z_w}\right) & 0 \end{bmatrix} \begin{bmatrix} u \\ \theta \end{bmatrix} + \begin{bmatrix} x_\eta - x_w\left(\frac{m_\eta U_e - m_q z_\eta}{m_w U_e - m_q z_w}\right) \\ \left(\frac{m_\eta z_w - m_w z_\eta}{m_w U_e - m_q z_w}\right) \end{bmatrix} \eta \quad \text{(Cook, 6.33)}$$

One can get the natural frequency and the damping ratio from the eigenvalues of A_p matrix, or the following approximation formula:

$$\begin{aligned} \omega_p &= \sqrt{\frac{-g z_u}{U_e}} \qquad [rad/s] \\ \zeta_p &= -\frac{x_u}{2\omega_p} \end{aligned} \qquad \text{(Cook, 6.36)}$$

3.2. MATLAB Script (followed by Section 2.2.)

```
%% Part 3. MODEL APPROXIMATION
disp(repmat('=',1,80));
disp('Part 3. MODEL APPROXIMATION');

disp('** Short Period **');
A_s = [z_w z_q;
       m_w m_q];
B_s = [z_eta;
       m_eta];
% Calculation by Eigenvalues
reducedLambda_s = eig(A_s)
reducedOmega_s = sqrt(reducedLambda_s(1)*reducedLambda_s(2))
reducedZeta_s = -(reducedLambda_s(1)+reducedLambda_s(2))/2/reducedOmega_s
% % Calculation by (Cook, 6.20)
% reducedOmega_s = sqrt(m_q*z_w - m_w*U_e)
% reducedZeta_s = -(m_q + z_w)/2/reducedOmega_s

disp('** Phugoid **');
A_p = [x_u - x_w*(m_u*U_e - m_q*z_u)/(m_w*U_e - m_q*z_w), -g;
       (m_u*z_w - m_w*z_u)/(m_w*U_e - m_q*z_w), 0];
```

```
B_p = [x_eta - x_w*(m_eta*U_e - m_q*z_eta)/(m_w*U_e - m_q*z_w);
        (m_eta*z_w - m_w*z_eta)/(m_w*U_e - m_q*z_w)];
% Calculation by Eigenvalues
reducedLambda_p = eig(A_p)
reducedOmega_p = sqrt(reducedLambda_p(1)*reducedLambda_p(2))
reducedZeta_p = -(reducedLambda_p(1)+reducedLambda_p(2))/2/reducedOmega_p
% % Calculation by (Cook, 6.20)
% reducedOmega_s = sqrt(-g*z_u/U_e)
% reducedZeta_s = -x_u/2/reducedOmega_p

disp(repmat('=',1,80));
```

3.3. Terminal Outputs

```
================================================================================
Part 3. MODEL APPROXIMATION
** Short Period **

reducedLambda_s =

  -0.3709 + 1.3496i


reducedOmega_s =

    1.3996


reducedZeta_s =

    0.2650

** Phugoid **

reducedLambda_p =

   0.0007 + 0.0783i
   0.0007 - 0.0783i


reducedOmega_p =

    0.0783


reducedZeta_p =

   -0.0086

================================================================================
```

Part 4. SETTING UP A DESIGN CRITERIA (10pts.)

> Set up a target short period colsed-loop system dynamics in terms of natural frequency and damping ratio.

4.1. The Thumb Print Criterion

The thumb print criterion provides guidance for aeroplane designers and evaluators concerning the best combinations of longitudinal short-period mode damping and frequency to give good handling qualities. However, it must be remembered that the information provided is empirical and based entirely on pilot opinion. The common form of presentation of the criterion is shown in the following figure retrieved from (Cook, Figure 10.2):

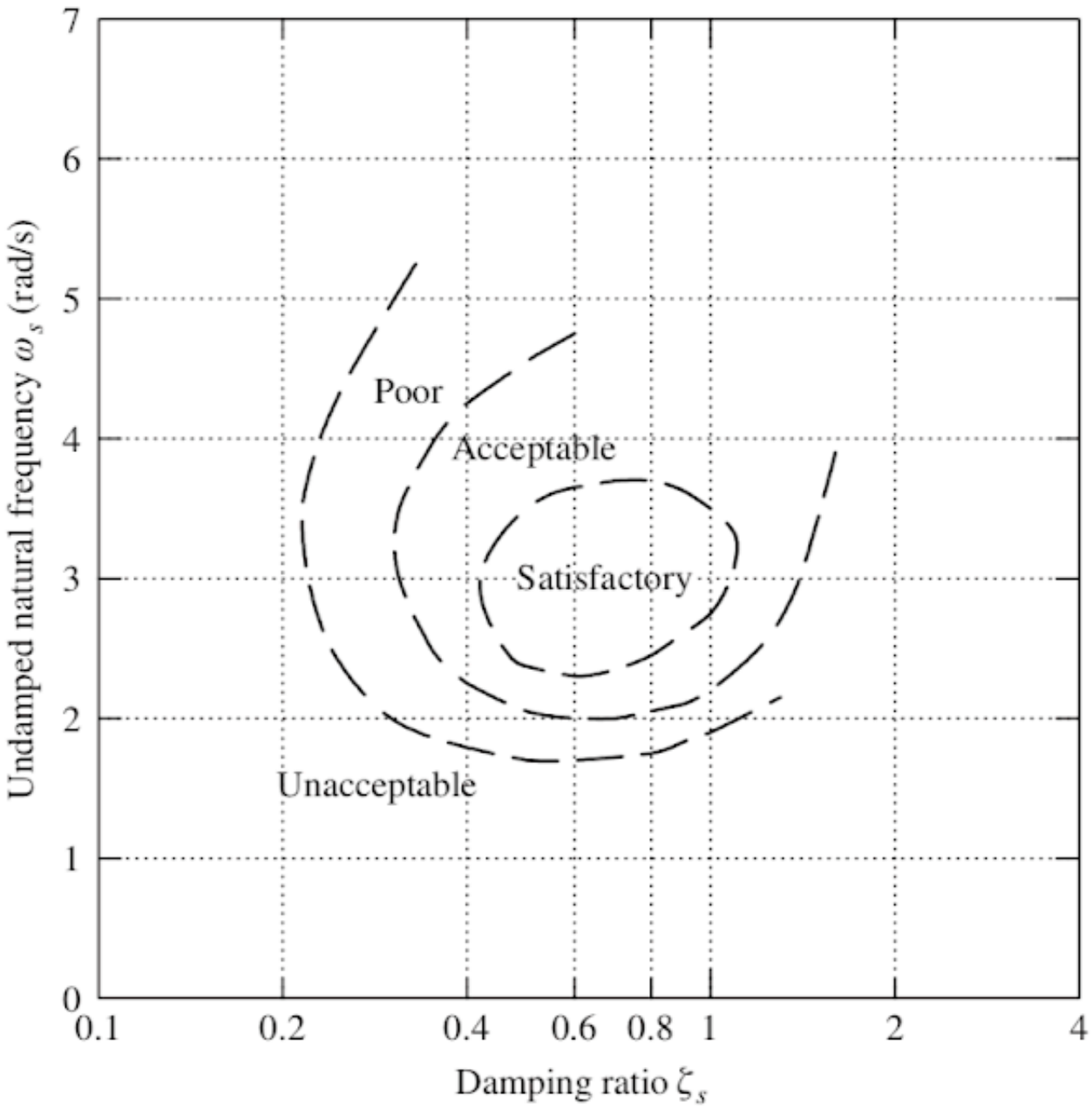


Figure 4.1. Longitudinal short-period pilot opinion contours: The thumb print criterion

From now on, to get a satisfactory short-period response, we will start designing a closed-loop feedback controller, which targets $\omega_s = 3\ [rad/s]$ of natural frequency, and $\zeta_s = 0.7$ of damping ratio. This leads us to figure out the dominant pole locations from the roots of the polynomial equation $s^2 + 2\zeta\omega_s s + \omega_s^2 = 0$, which yields $s_{1,2} = -2.1 \pm 2.1424i$.

4.2. MATLAB Script (followed by Section 3.2.)

```
%% Part 4. SETTING UP A DESIGN CRITERIA
disp(repmat('=',1,80));
disp('Part 4. SETTING UP A DESIGN CRITERIA');

desiredOmega_s = 3
desiredZeta_s = 0.7

syms s;
eqn = s^2+2*desiredZeta_s*desiredOmega_s*s + desiredOmega_s.^2 == 0;
desiredLambda_s = eval(solve(eqn,s))

disp(repmat('=',1,80));
```

4.3. Terminal Outputs

```
================================================================================
Part 4. SETTING UP A DESIGN CRITERIA

desiredOmega_s =

     3


desiredZeta_s =

    0.7000


desiredLambda_s =

  -2.1000 - 2.1424i
  -2.1000 + 2.1424i

================================================================================
```

Part 5. CONTROLLER DESIGN (20pts.)

> Design a feedback control law either in pitch angle feedback or pitch angle plus pitch rate feedback to meet the requirements of the closed-loop systems. You can use either Root-locus or pole placement technique in the similar way to the lecture note.

5.1. Transfer Functions

Transfer functions describing short-period response to elevator can be written as follows:

$$\frac{w(s)}{\eta(s)} = \frac{z_\eta \left(s + U_e \frac{m_\eta}{z_\eta}\right)}{s^2 - (m_q + z_w)s + (m_q z_w - m_w U_e)} \qquad \text{(Cook, 6.17)}$$

$$\frac{q(s)}{\eta(s)} = \frac{m_\eta \left(s - z_w\right)}{s^2 - (m_q + z_w)s + (m_q z_w - m_w U_e)} \qquad \text{(Cook, 6.18)}$$

Since, $q(s)$ can be approximated as $s\theta(s)$, transfer function from the elevator control to pitch angle can be described as follows:

$$\frac{\theta(s)}{\eta(s)} = \frac{m_\eta \left(s - z_w\right)}{s\left[s^2 - (m_q + z_w)s + (m_q z_w - m_w U_e)\right]}$$

Filling in the parameters and coefficients yields the following transfer functions for the controller design.

$$G_\eta^q(s) = \frac{q(s)}{\eta(s)} = \frac{-4.888s - 1.444}{s^2 + 0.7418s + 1.967}$$
$$G_\eta^\theta(s) = \frac{\theta(s)}{\eta(s)} = \frac{q(s)}{s \cdot \eta(s)} = \frac{-4.888s - 1.444}{s^3 + 0.7418s^2 + 1.967s}$$

5.2. Pitch Angle Feedback (P Controller)

> Actuator(lagging effect) is neglected.

Pitch angle feedback controller design can be described as follows:

Figure 5.1. Pitch angle feedback controller

$$\begin{aligned}
\theta &= G_\eta^\theta K_\theta \cdot e_\theta \\
&= G_\eta^\theta K_\theta \cdot (\theta_d - \theta) \\
(1 + G_\eta^\theta K_\theta) \cdot \theta &= G_\eta^\theta K_\theta \cdot \theta_d \\
\frac{\theta}{\theta_d} &= \frac{G_\eta^\theta K_\theta}{1 + G_\eta^\theta K_\theta}
\end{aligned}$$

Slight abuse of the notations let us find the closed-loop transfer function from the reference input θ_d to the output θ:

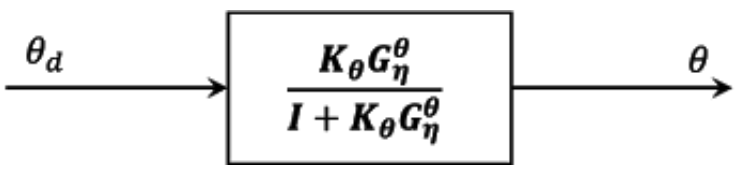
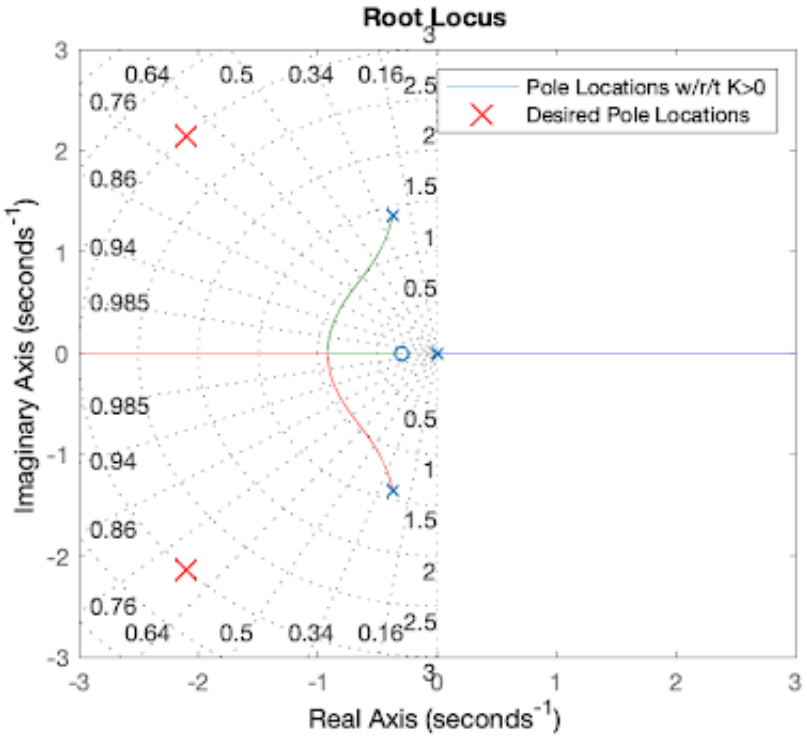


Figure 5.2. Closed-loop transfer function of the P controller

Here, the characteristic equation can be derived from the denominator of the transfer function $1 + K_\theta G_\eta^\theta(s) = 0$ (multiplication order does not matter for the single input, single output system); which satisfies $K_\theta G_\eta^\theta(s) = -1 = 1\angle 180^\circ$ (i.e., $|G_\eta^\theta(s)| = \frac{1}{K_\theta}$, $\angle G_\eta^\theta(s) = \pm 180^\circ$). MATLAB function `rlocus(G)` ease the process to find the roots of s with respect to the varing K_θ. Positive gain gives unstable pole location, while negative gain gives stable response. In eiter cases, non of them can meet the design criteria by adjusting pitch angle feedback gain only.



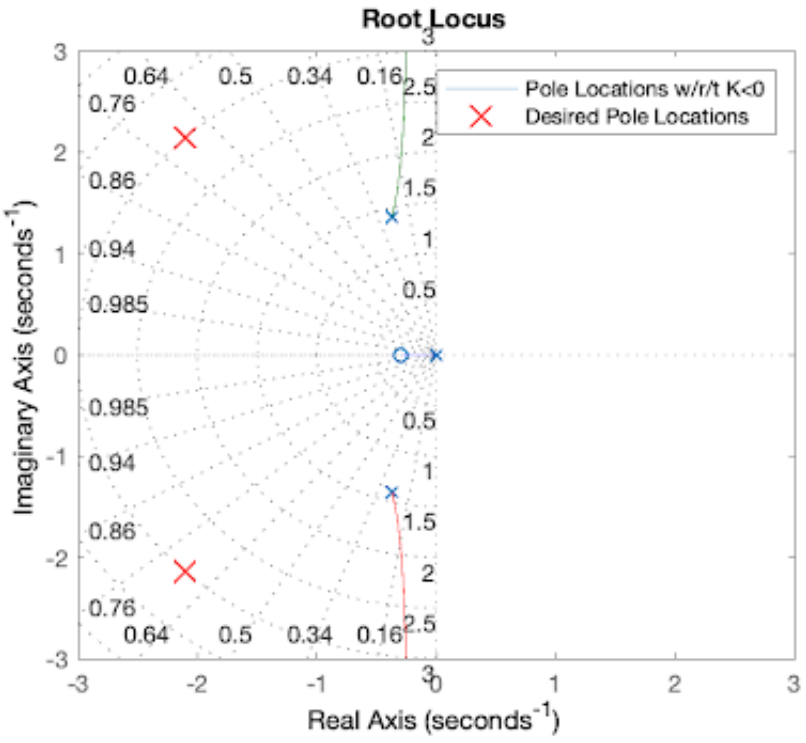


Figure 5.3. Root locus for the pitch angle feedback controller

5.3. Pitch Angle and Rate Feedback (PD Controller)

> Actuator(lagging effect) is neglected.

Pitch angle and rate feedback controller design can reshape the root locus by imposing a zero to the system. By assuming $\dot{\theta} = q$, we can design a PD-like controller as follows:

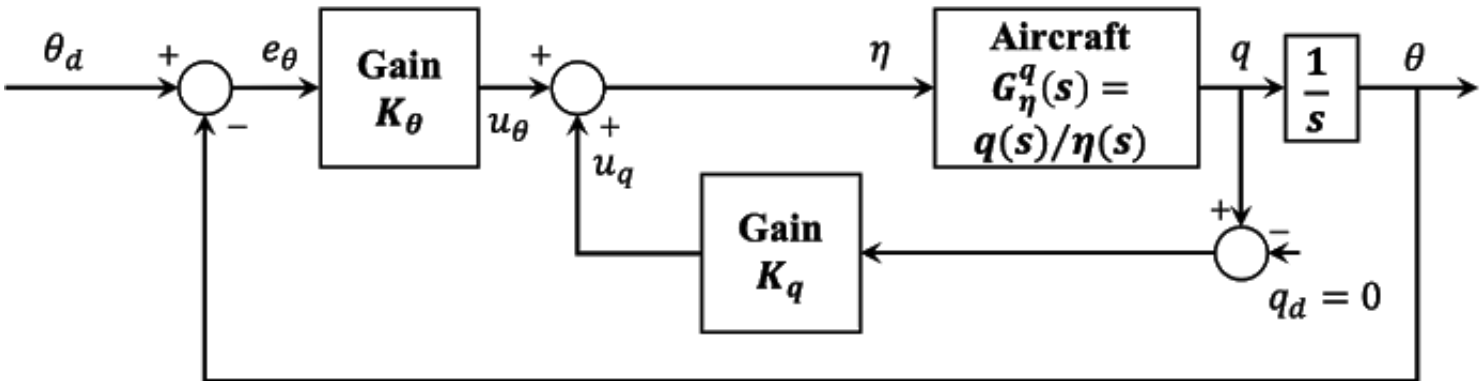


Figure 5.4. Pitch angle and rate feedback controller (1)

Slight abuse of the notations let us find the transfer function from the u_θ to the q;

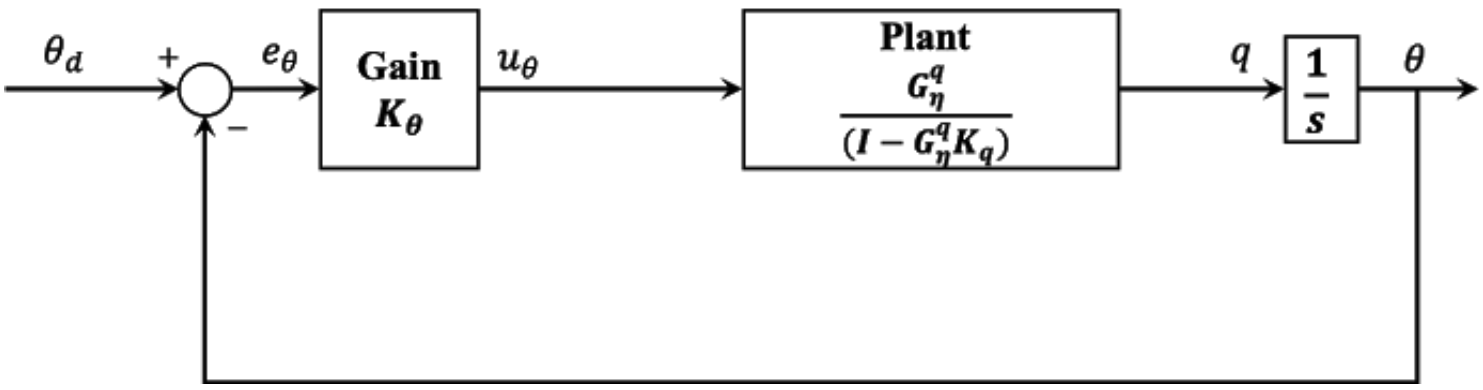


Figure 5.5. Pitch angle and rate feedback controller (2)

or u_θ to the θ, which will be helpful in using MATLAB function `rlocus(G)` to ease the design process:

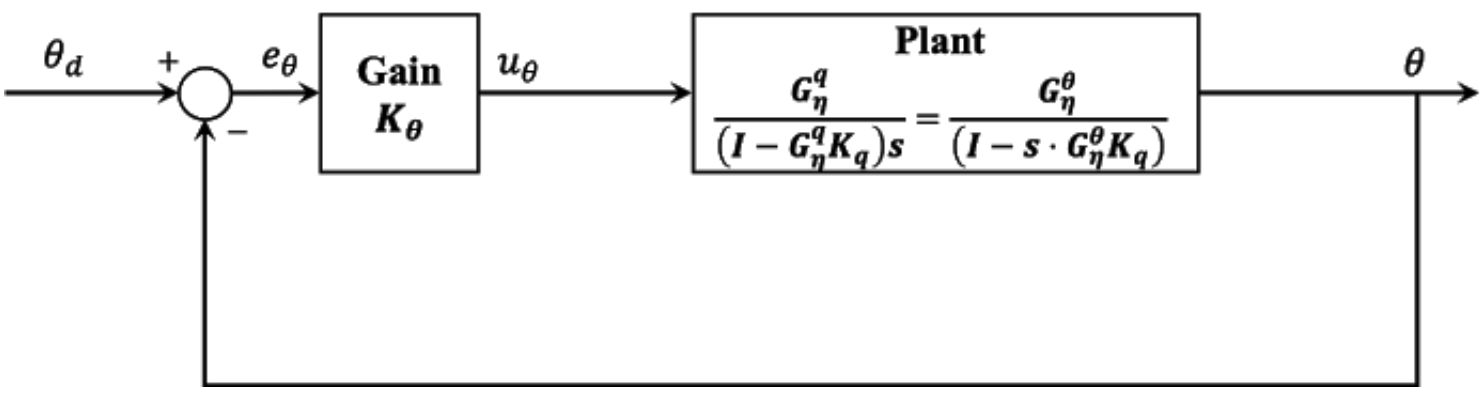
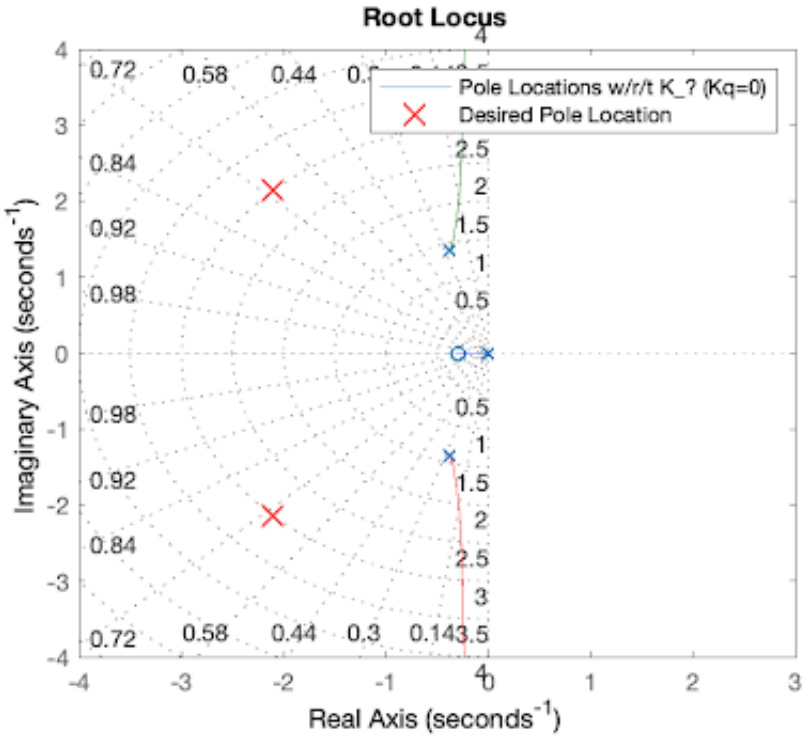


Figure 5.6. Pitch angle and rate feedback controller (3)

Figure 5.7. shows the root locus with respect to the $K_\theta < 0$ with the fixed K_q gain. When the K_q is zero, the result is the same as the Figure 5.3., whereas increasing the K_q moves the poles towards the desired location. It means further increasing the K_q with the proper K_θ can meet the design requirements as we settled in the Section 4.1.



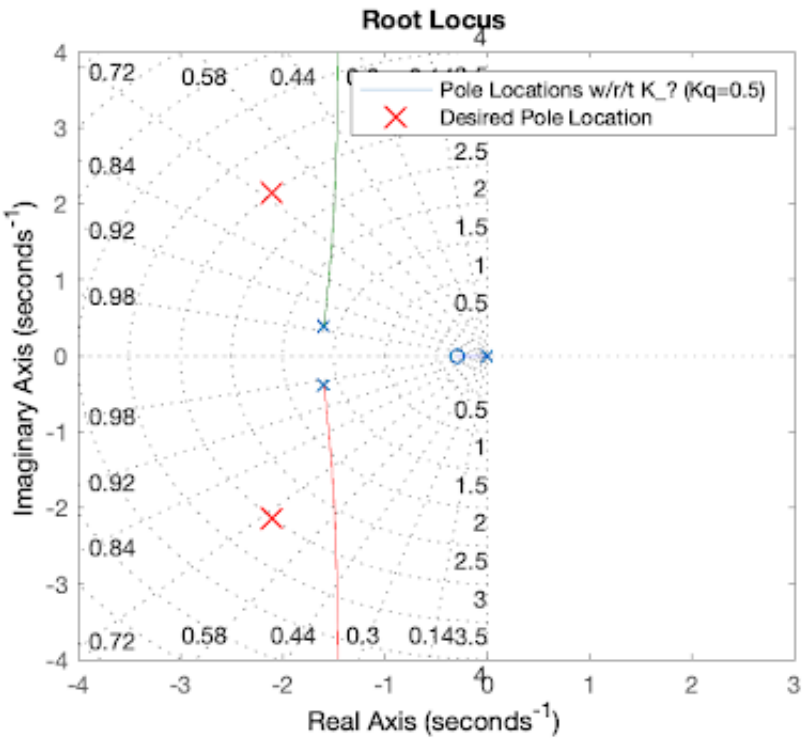


Figure 5.7. Root locus for the pitch angle plus rate feedback controller

As we have done in the Section 5.2., the closed-loop transfer function of the PD controller can be described as follows (Figure 5.8. uses the $s \cdot G_\eta^\theta = G_\eta^q$ relationship):

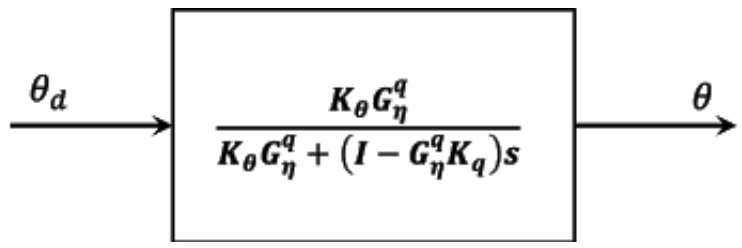


Figure 5.8. Closed-loop transfer function of the PD controller (1)

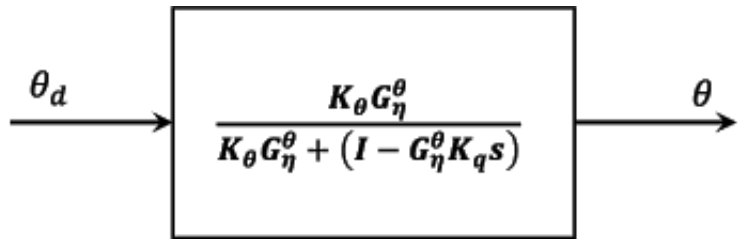


Figure 5.9. Closed-loop transfer function of the PD controller (2)

From the closed-loop transfer function, we can get the characteristic equation $1 + K_\theta G_\eta^\theta - sK_q G_\eta^\theta = 0$. Letting ϵ as a ratio between the θ and q gain, $K_q = \epsilon K_\theta$, we can rearrange the characteristic equation as follows:

$$K_\theta G_\eta^\theta - \epsilon s K_\theta G_\eta^\theta =$$
$$K_\theta (1 - \epsilon s) G_\eta^\theta = -1 = 1 \angle 180^\circ$$

, which means a zero is added to the system designed is the Section 5.2., adding phase lead to the system.

Since $s_{1,2} = -2.1 \pm 2.1424i$, should be the roots of the characteristic equation above, proper gains can be derived by solving following two equations (first find ϵ from the phase condition, and tune the K_θ):

$$|K_\theta(1-\epsilon s_1)G_\eta^\theta(s_1)| = 1$$
$$\angle((1-\epsilon s_1)G_\eta^\theta(s_1)) = \pm 180^\circ$$

You can also take the trial and error approach to find the proper K_q(or ϵ), and K_θ conditions. Below is the root locus when $K_q = 0.754$, and the desired pole location can be achieved when the gain $K_\theta = -1.41$.

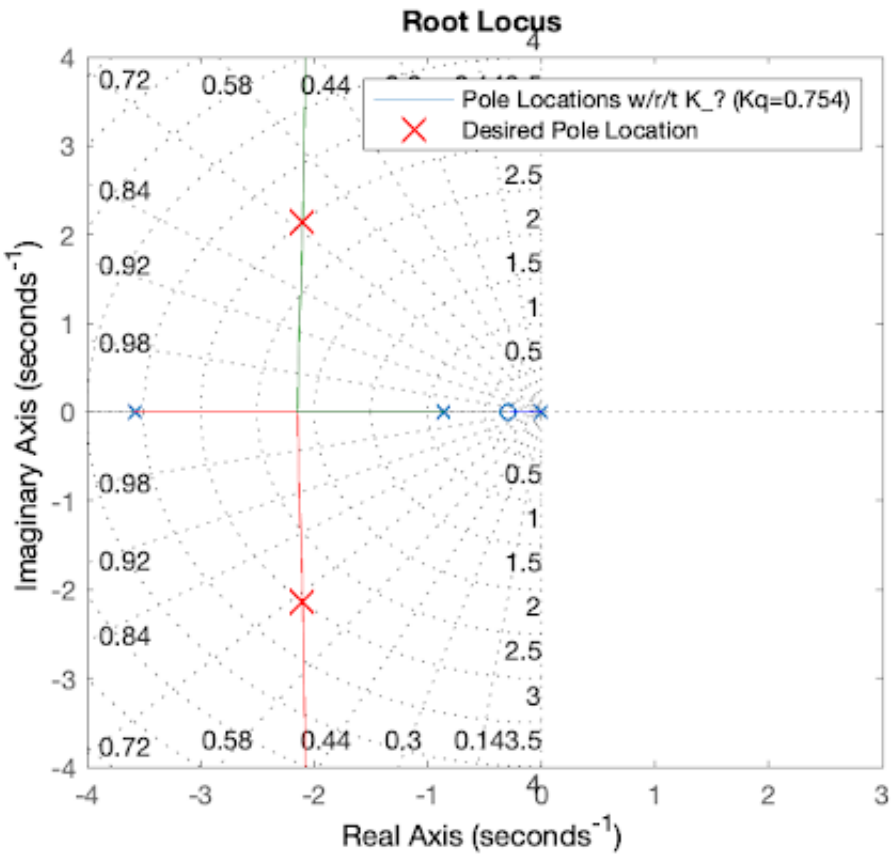


Figure 5.10. Root locus for the final design

5.4. MATLAB Script (followed by Section 4.2.)

```
%% Part 5. CONTROLLER DESIGN
disp(repmat('=',1,80));
disp('Part 5. CONTROLLER DESIGN');

% Transfer Functions
disp('** Transfer Functions **');
num1 = [m_eta, -m_eta*z_w];
den1 = [1, -(m_q+z_w), (m_q*z_w - m_w*U_e)];
G_eta2q = tf(num1, den1)

num2 = [m_eta, -m_eta*z_w];
den2 = [1, -(m_q+z_w), (m_q*z_w - m_w*U_e), 0];
G_eta2theta = tf(num2, den2)

% Pitch Angle Feedback
figure(1)
set(gcf, 'Position',[100 100 400 800])

subplot(2,1,1)
title('Pitch Angle Feedback w/ Positive gain')
rlocus(G_eta2theta)
hold on;
plot(real(desiredLambda_s),imag(desiredLambda_s),'rx','Markersize',10)
xlim([-3 3]); ylim([-3 3]); grid on; legend('Pole Locations w/r/t K>0','Desired Pole Locations');
hold off;

subplot(2,1,2)
title('Pitch Angle Feedback w/ Negative gain')
rlocus(-G_eta2theta)
hold on;
plot(real(desiredLambda_s),imag(desiredLambda_s),'rx','Markersize',10)
xlim([-3 3]); ylim([-3 3]); grid on; legend('Pole Locations w/r/t K<0','Desired Pole Locations');
```

```
hold off;

% Pitch Angle and Rate Feedback
figure(2)
set(gcf, 'Position',[500 100 400 800])

subplot(2,1,1)
Kq = 0;
G_utheta2theta = minreal(G_eta2q / ((1 - Kq*G_eta2q)*tf([1 0],1)));
rlocus(-G_utheta2theta)
hold on;
plot(real(desiredLambda_s),imag(desiredLambda_s),'rx','Markersize',10)
xlim([-4 3]); ylim([-4 4]); grid on; legend('Pole Locations w/r/t K_θ (Kq=0)','Desired Pole Location');
hold off;

subplot(2,1,2)
Kq = 0.5;
G_utheta2theta = minreal(G_eta2q / ((1 - Kq*G_eta2q)*tf([1 0],1)));
rlocus(-G_utheta2theta)
hold on;
plot(real(desiredLambda_s),imag(desiredLambda_s),'rx','Markersize',10)
xlim([-4 3]); ylim([-4 4]); grid on; legend('Pole Locations w/r/t K_θ (Kq=0.5)','Desired Pole
Location');
hold off;


disp('** Gain Tuning (Tune Kq, Ktheta near the magnitude==1, and the phase angle==+-180[deg]) **');
Kq = 0.754;
Ktheta=-1.41;
length = abs(Ktheta * evalfr(G_eta2theta,desiredLambda_s(1)) - desiredLambda_s(1) * Kq *
evalfr(G_eta2theta,desiredLambda_s(1)))
phase = angle(Ktheta * evalfr(G_eta2theta,desiredLambda_s(1)) - desiredLambda_s(1) * Kq *
evalfr(G_eta2theta,desiredLambda_s(1)))

figure(3)
set(gcf, 'Position',[900 100 400 350])
G_utheta2theta = minreal(G_eta2q / ((1 - Kq*G_eta2q)*tf([1 0],1)))
rlocus(-G_utheta2theta)
hold on;
plot(real(desiredLambda_s),imag(desiredLambda_s),'rx','Markersize',10)
xlim([-4 3]); ylim([-4 4]); grid on; legend('Pole Locations w/r/t K_θ (Kq=0.754)','Desired Pole
Location');
hold off;

disp(repmat('=',1,80));
```

5.5 Terminal Outputs

```
================================================================================
Part 5. CONTROLLER DESIGN
** Transfer Functions **

G_eta2q =

     -4.888 s - 1.444
  ----------------------
  s^2 + 0.7418 s + 1.967

Continuous-time transfer function.


G_eta2theta =

       -4.888 s - 1.444
  --------------------------
  s^3 + 0.7418 s^2 + 1.967 s

Continuous-time transfer function.

** Gain Tuning (Tune Kq, Ktheta near the length=1, and the phase angle==+-180[deg]) **
```

```
magnitude =

    1.0005


phase =

    3.1409


G_utheta2theta =

      -4.888 s - 1.444
  -------------------------
  s^3 + 4.428 s^2 + 3.055 s

Continuous-time transfer function.

================================================================================
```

Part 6. CLOSED-LOOP TIME RESPONSES (20pts.)

> Compare closed-loop time responses to initial conditions for the open-loop(**Part 1**) and closed-loop system by feedback control. You can plot the time responses of state variables for comparison.

6.1. Simulink Block Diagram

Simulink block diagram to show the open-loop(**Part 1**, without any control) and closed-loop(designed in the **Part 5**) time responses to initial conditions can be constructed as follows:

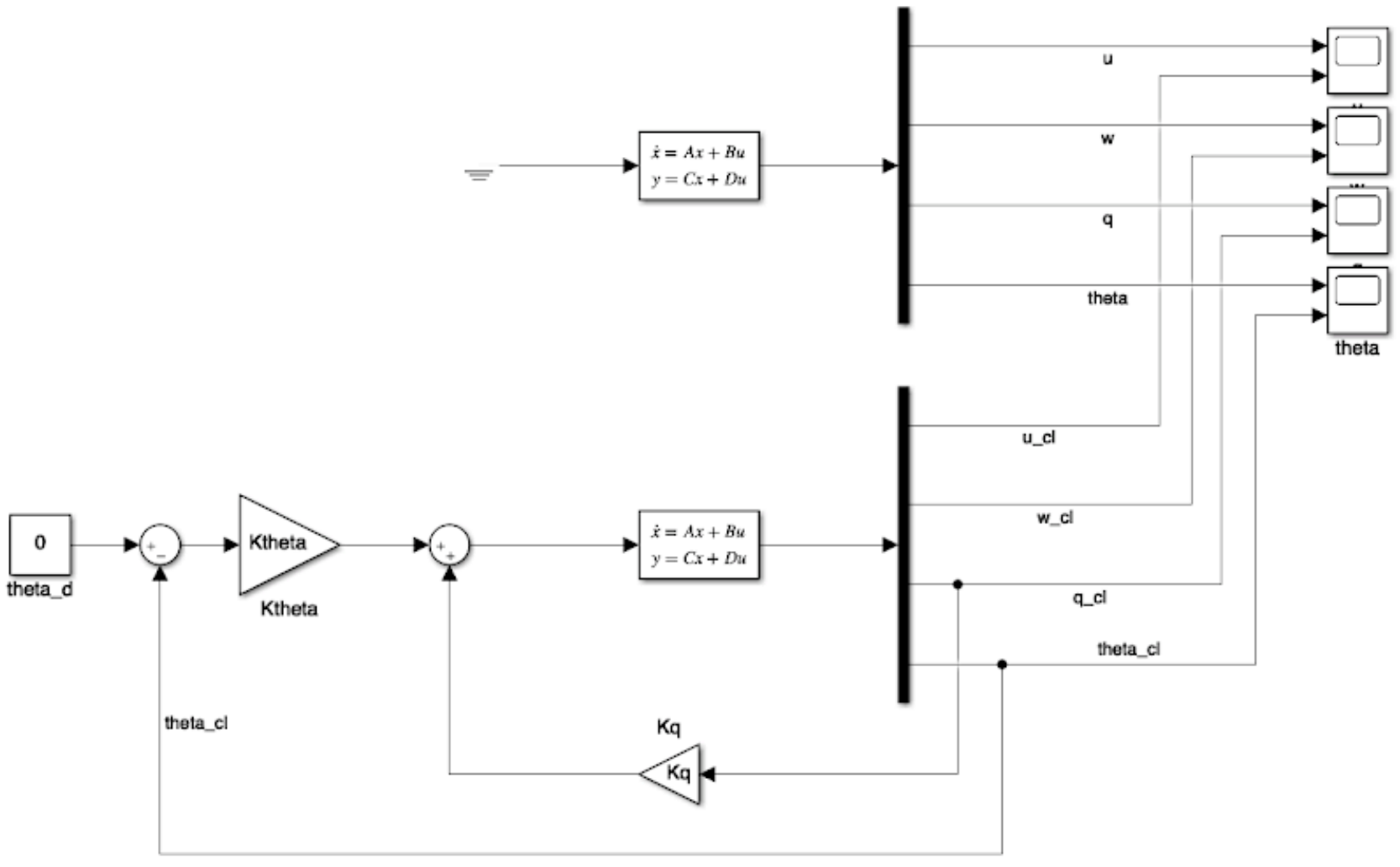


Figure 6.1. Block diagrams

From the initial conditions (perturbations from the equilibrium point) of $[u_0, w_0, q_0, \theta_0] = [1, 1, 1, 1]$, open-loop response shows stable but slowly converging motion, while the designed contoller settles down the irritating responses quickly as desired.

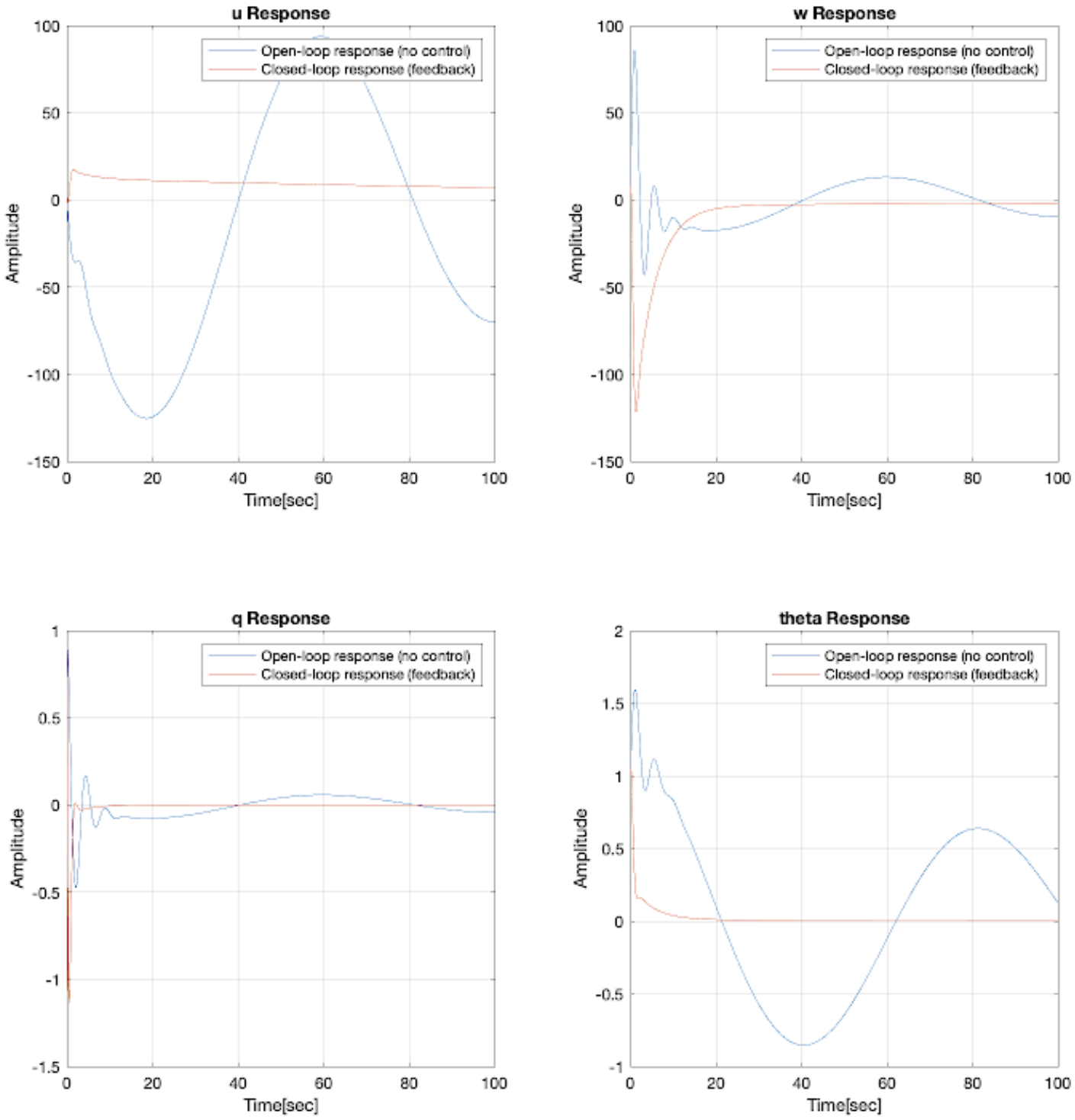


Figure 6.2. Time responses to initial conditions

6.2. MATLAB Script (followed by Section 5.4)

> Download the HW2Part6.slx file first from the website: https://github.com/SKYnSPACE/AE450HW2

```
%% Part 6. CLOSED-LOOP TIME RESPONSES
disp(repmat('=',1,80));
disp('Part 6. CLOSED-LOOP TIME RESPONSES');

x0 = [1;1;1;1];     % initial(perturbed) state
C = eye(4);         % output variables
D = zeros(4,1);     % no disturbances

disp('** Background simulation running... **');
sim('HW2Part6');
disp('** DONE! **');

figure(4)
set(gcf, 'Position',[300 50 800 800])

subplot(2,2,1)
plot(uOut.time, uOut.signals(1).values, uOut.time, uOut.signals(2).values)
title('u Response'); xlabel('Amplitude'); ylabel('Time[sec]');
legend('Open-loop response (no control)','Closed-loop response (feedback)');
grid on;

subplot(2,2,2)
plot(wOut.time, wOut.signals(1).values, wOut.time, wOut.signals(2).values)
```

```
title('w Response'); xlabel('Amplitude'); ylabel('Time[sec]');
legend('Open-loop response (no control)','Closed-loop response (feedback)');
grid on;

subplot(2,2,3)
plot(qOut.time, qOut.signals(1).values, qOut.time, qOut.signals(2).values)
title('q Response'); xlabel('Amplitude'); ylabel('Time[sec]');
legend('Open-loop response (no control)','Closed-loop response (feedback)');
grid on;

subplot(2,2,4)
plot(thetaOut.time, thetaOut.signals(1).values, thetaOut.time, thetaOut.signals(2).values)
title('theta Response'); xlabel('Amplitude'); ylabel('Time[sec]');
legend('Open-loop response (no control)','Closed-loop response (feedback)');
grid on;


disp(repmat('=',1,80));
```

6.3. Terminal Outputs

```
================================================================================
Part 6. CLOSED-LOOP TIME RESPONSES
** Background simulation running... **
** DONE! **
================================================================================
```

Part 7. PHASE-LEAD COMPENSATOR (15pts.)

> Propose a phase-lead compensator design. Do your best to fix the parameters of the compensator to meet the same close-loop system dynamics. Demonstrate the benefit of using compensator compared to other approches in **Part 5** with your best effort.

7.1. Phase Lead Compensator

The Phase-lead compensator design can be described as follows with ($p > z$):

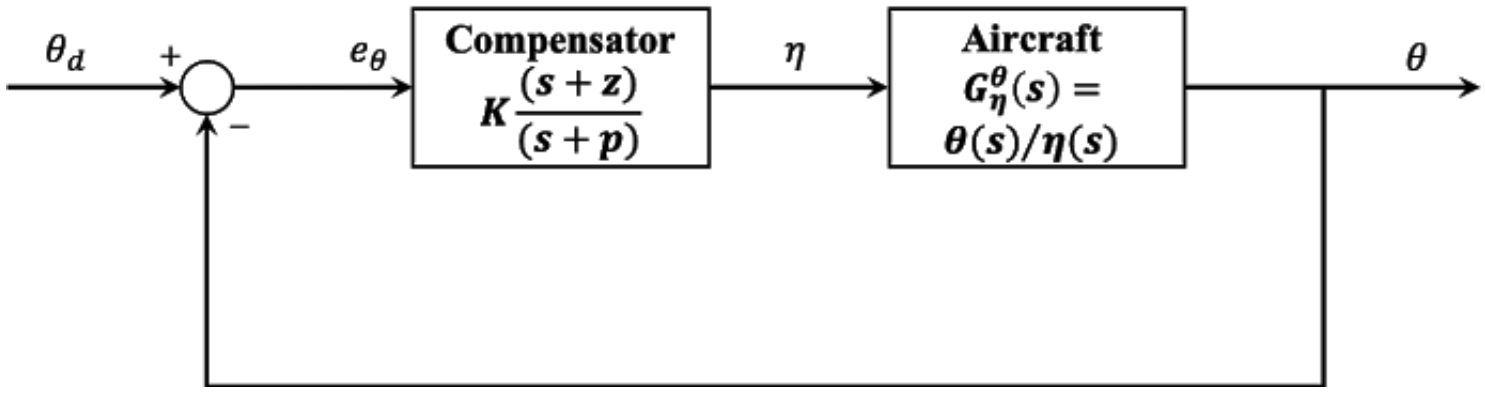


Figure 7.1. Feedback design by lead compensator

With MATLAB `sisotool`, we can design a proper lead compensator easily on top of the Bode plot, and root locus by just dragging the zero, pole, and root(K gain) locations. Figure 7.2. shows the `sisotool` containing plant dynamics(G_η^θ) and design critera($\omega_s = 3, \zeta_s = 0.7$) only. Now, we should put our root locations where the semi-circle and straight lines crosses.

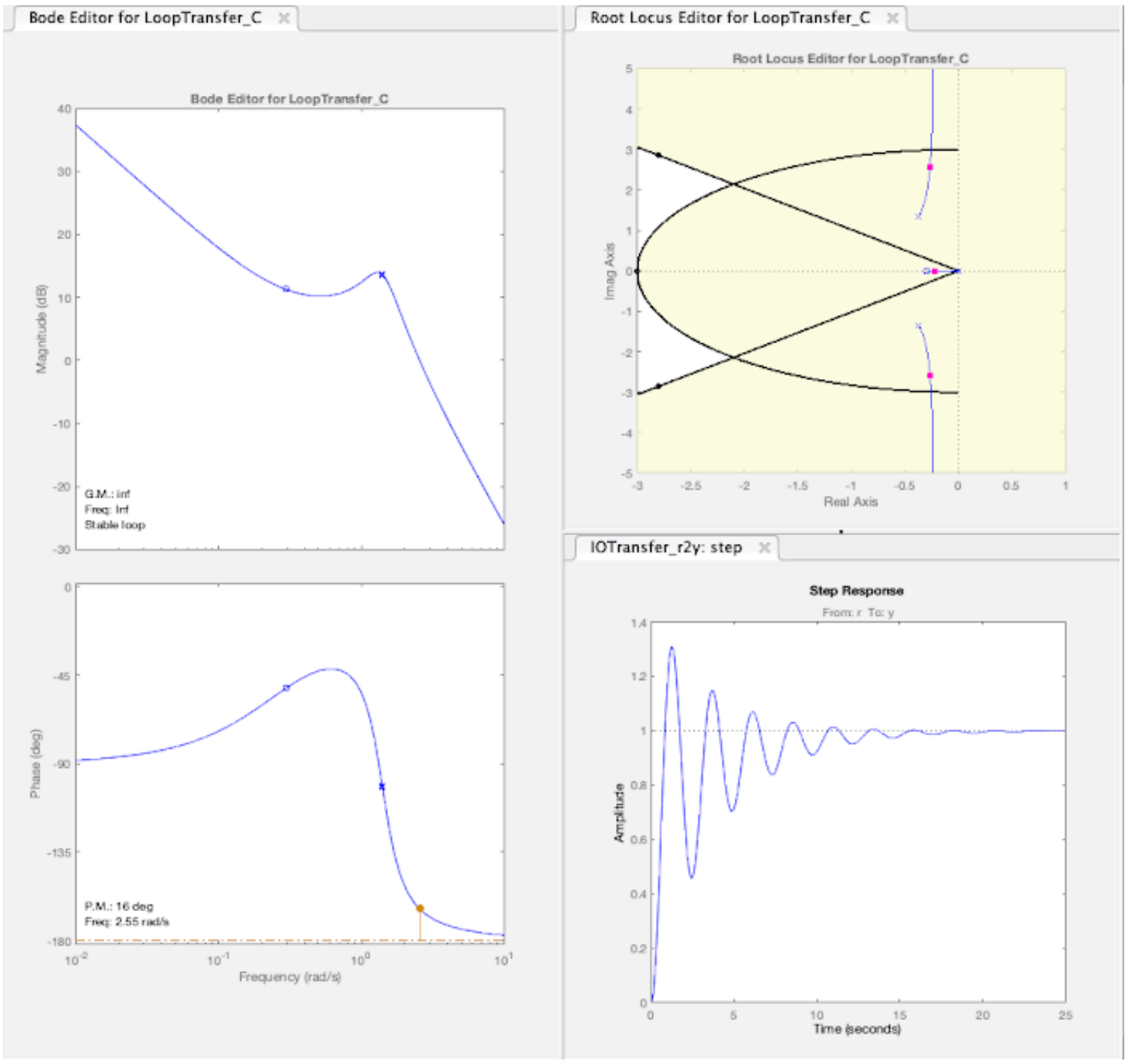


Figure 7.2. Setting design requirements on the MATLAB sisotool

After putting a lead compensator (phase lead around the frequencies from 1 to $10[rad/s]$, $z = -1.4134, p = -12.362$) with the gain $K = 7.583$, we can put the roots at the desired locations. (cf. a root near the origin does not dominate this system, since the zero around $s = -0.295$ cancels out the root.)

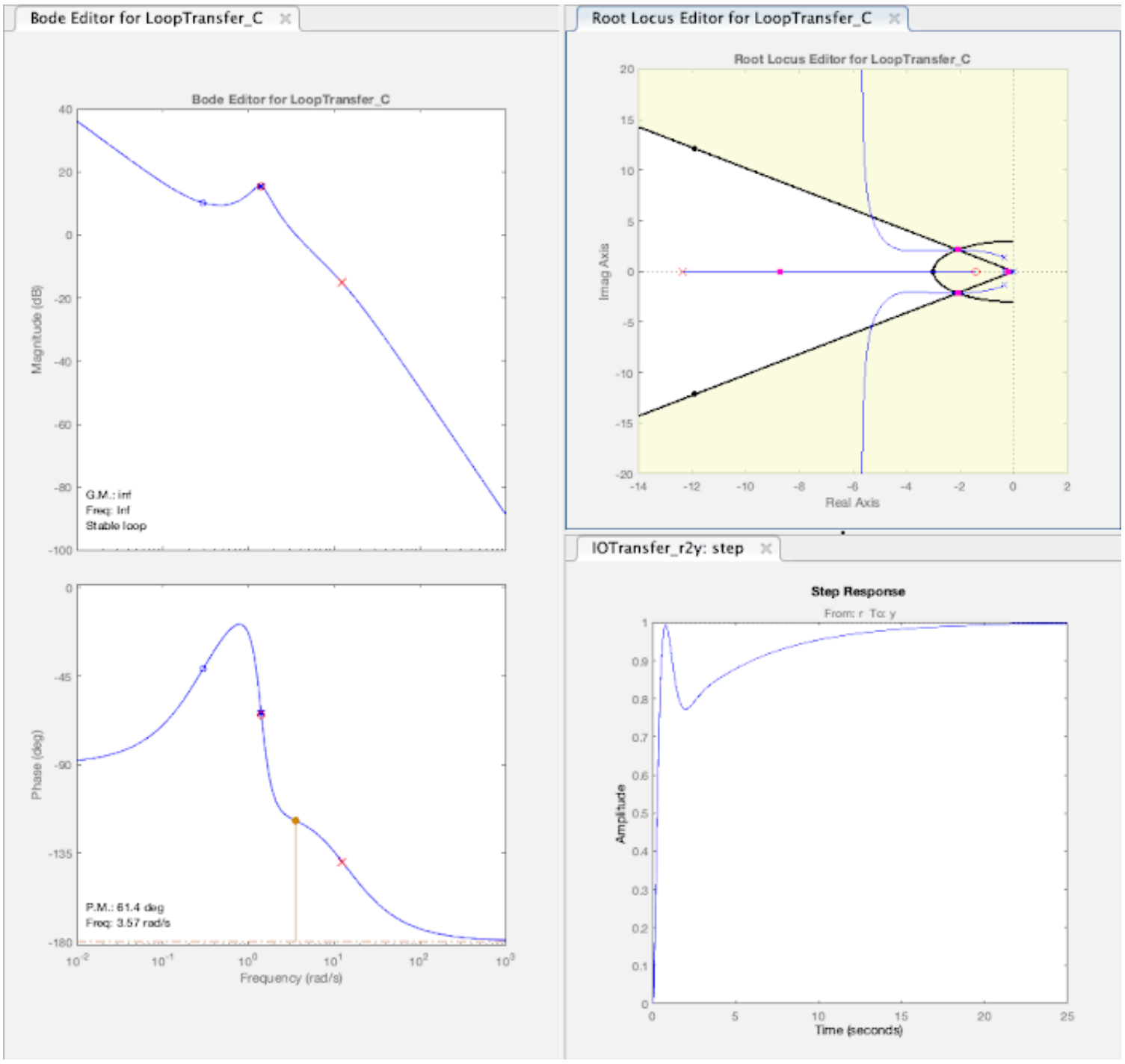


Figure 7.3. Phase-lead compensator design using MATLAB sisotool

Benefits of using lead compensator over PD controller propsed in the Part 5 can be found from the Brian's YouTube video: https://youtu.be/xLhvil5sDcU?t=337 , which explains lead-lag compensators. Please watch! He delivers very informative and educational contents in a short time.

7.2 MATLAB Scripts (followed by Section 6.2)

> Download the HW2Part7.mat file first from the website: https://github.com/SKYnSPACE/AE450HW2

```
%% Part 7. PHASE-LEAD COMPENSATOR
disp(repmat('=',1,80));
disp('Part 7. PHASE-LEAD COMPENSATOR');

disp('** Loading sisotool... **');

% Design form the scratch
% G = -G_eta2theta;
% sisotool(G);

% Load from the saved file
sisotool('HW2Part7.mat');

disp('** DONE! **');

disp(repmat('=',1,80));
```

7.3 Terminal Outputs

```
================================================================================
Part 7. PHASE-LEAD COMPENSATOR
** Loading sisotool... **
** DONE! **
================================================================================
```

REFERENCES

1. Cook, Michael V. *Flight dynamics principles: a linear systems approach to aircraft stability and control.* Butterworth-Heinemann, 2012.↩↩↩