AE 551 Flight Dynamics II

MATLAB Session

Bella Kim

Instructor: Dr. Keshmiri

Contents

Session 1: April 17th, 2015

In this session, we will focus on developing the state space matrices (A, B, C, and D) and some useful commands in MATLAB when you interpreting the dynamics system behavior. This session is very similar to what you have already done in Homework 8.

1 Developing State Space Matrices

Given the stability and control derivatives, you can develop A, B, C, and D matrices. These matrices came from the linearized state space model. [Ref.2]

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} \tag{1}$$
$$y = C\vec{x} + D\vec{u} \tag{2}$$

y is the output you desire.

- A is called the state matrix.
- B is called the input matrix.
- C is called the output matrix.
- D is called the direct transmission matrix.

BE CAREFUL for the dimension of the matrices. If you want to get just one output meaning 1 by 1 matrix, then you have to form C matrix as 1 by 4 matrix. (Assuming we have 4 states) D matrix is usually 0 matrix unless you need the control input to calculate the output you desired.

- A: n $\times$ n matrix ($\vec{x}$ was assumed as n by 1 vector)
- B: n $\times$ r ($\vec{u}$ was assumed as r by 1 vector)
- C: m $\times$ n matrix (y was assumed m by 1 matrix)
- D: m $\times$ r matrix

1.1 Calculating Stability and Control Derivatives

See page 319 and 348 in Roskam Book. You have equations to calculate dimensional stability and control derivatives. When you deal with the several flight conditions like you did in homework 8, function file will be very useful to calculate several conditions.
To make the function file in MATLAB, you need to define input and output of the function. In our case, input will be given stability and control derivatives. Output will be dimensional stability and control derivatives.

Input		**Output**	
I_B	α	X_u	X_{T_u}
$\bar{q}$	S	X_α	X_{δ_e}
$\bar{c}$	b	Z_u	Z_α
m	U_1	$Z_{\dot{\alpha}}$	Z_q
C_{D_u}	C_{D_1}	Z_{δ_e}	M_u
$C_{T_{x_u}}$	$C_{T_{x_1}}$	M_{T_u}	M_α
C_{D_α}	$C_{D_{\delta_e}}$	M_{T_α}	$M_{\dot{\alpha}}$
C_{L_u}	C_{L_1}	M_q	M_{δ_e}
C_{L_α}	$C_{L_{\dot{\alpha}}}$	Y_β	Y_p
C_{L_q}	$C_{L_{\delta_e}}$	Y_r	Y_{δ_a}
C_{m_u}	C_{m_1}	Y_{δ_r}	L_β
$C_{m_{T_u}}$	$C_{m_{T_1}}$	L_p	L_r
C_{m_α}	$C_{m_{T_\alpha}}$	L_{δ_a}	L_{δ_r}
$C_{m_{\dot{\alpha}}}$	C_{m_q}	N_β	N_{T_β}
$C_{m_{\delta_e}}$	C_{y_p}	N_p	N_r
C_{y_r}	$C_{y_{\delta_a}}$	N_{δ_a}	N_{δ_r}
$C_{y_{\delta_r}}$	C_{y_β}	I_{xx_s}	I_{yy_s}
C_{l_β}	C_{l_p}	I_{zz_s}	I_{xz_s}
C_{l_r}	$C_{l_{\delta_a}}$		
$C_{l_{\delta_r}}$	C_{n_β}		
$C_{n_{T_\beta}}$	C_{n_p}		
C_{n_r}	$C_{n_{\delta_a}}$		
$C_{n_{\delta_r}}$			

Table 1: *Longitudinal and Lateral-Directional stability and control derivatives Input and output for MATLAB function file*

For function file syntax, it looks like the following

```
function [output1, output2, ..., output n]
          = function_name(input1, input2, ... , input n)
       output1 = equation; %this is just example.
end
```

Keep in mind the following.

- Don't forget to convert the moment of inertia from body coordinate to the stability coordinate system.
- You can refer the transformation matrix (Eqn. 5.94) on page 346 in Roskam book.
- If you will use the other transformation, please see homework 1 solution. In this case, keep in mind that the moment of inertia is second order tensor. ($I_S = H_B^S \cdot I_B \cdot H_S^B$)
- You have to convert this before you calculate the dimensional stability and control derivative.
- I_{yy_B} will be same in the stability coordinate system. ($I_{yy_B} = I_{yy_s}$)
- You can assume given θ_1 is same as α_1.

```
function [X_u, X_Tu, X_alpha, X_de,...
          Z_u, Z_alpha, Z_alpha_dot, Z_q, Z_de,...
          M_u, M_Tu, M_alpha, M_Talpha, M_alpha_dot, M_q, M_de,...
          Y_beta, Y_p, Y_r, Y_da, Y_dr,...
          L_beta, L_p, L_r, L_da, L_dr,...
          N_beta, N_Tbeta, N_p, N_r, N_da, N_dr,...
          I_xx, I_yy, I_zz, I_xz]...
          = control_derivative(MOI_body, alpha,...
          qbar, S, cbar, b, m, U1,...
          C_Du, C_D1,C_Txu, C_Tx1, C_Dalpha,...
          C_Dde, C_Lu, C_L1, C_Lalpha, C_Lalpha_dot,...
          C_Lq, C_Lde, C_mu, C_m1, C_mTu, C_mT1, C_malpha,...
          C_mTalpha, C_malpha_dot, C_mq, C_mde,...
          C_ybeta, C_yp, C_yr, C_yda, C_ydr,...
          C_lbeta, C_lp, C_lr, C_lda, C_ldr,...
          C_nbeta, C_nTbeta, C_np, C_nr, C_nda, C_ndr)

MOI_s = convert_frame(MOI_body, alpha);
I_xx = MOI_s(1,1);
```

```
I_yy = MOI_s(2,2);
I_zz = MOI_s(3,3);
I_xz = MOI_s(1,3);

% Longitudinal
X_u = -qbar*S*(C_Du + 2*C_D1)/(m*U1);
X_Tu = qbar*S*(C_Txu + 2*C_Tx1)/(m*U1);
X_alpha = -qbar*S*(C_Dalpha - C_L1)/m;
X_de = -qbar*S*C_Dde/m;

Z_u = -qbar*S*(C_Lu + 2*C_L1)/(m*U1);
Z_alpha = -qbar*S*(C_Lalpha + C_D1)/m;
Z_alpha_dot = -qbar*S*cbar*C_Lalpha_dot/(2*m*U1);
Z_q = -qbar*S*cbar*C_Lq/(2*m*U1);
Z_de = -qbar*S*C_Lde/m;

M_u = qbar*S*cbar*(C_mu+2*C_m1)/(I_yy*U1);
M_Tu = qbar*S*cbar*(C_mTu + 2*C_mT1)/(I_yy*U1);
M_alpha = qbar*S*cbar*C_malpha/I_yy;
M_Talpha = qbar*S*cbar*C_mTalpha/I_yy;
M_alpha_dot = qbar*S*cbar^2*C_malpha_dot/(2*I_yy*U1);
M_q = qbar*S*cbar^2*C_mq/(2*I_yy*U1);
M_de = qbar*S*cbar*C_mde/I_yy;

%Lateral-directional
Y_beta = qbar*S*C_ybeta/m;
Y_p = qbar*S*b*C_yp/(2*m*U1);
Y_r = qbar*S*b*C_yr/(2*m*U1);
Y_da = qbar*S*C_yda/m;
Y_dr = qbar*S*C_ydr/m;

L_beta = qbar*S*b*C_lbeta/I_xx;
L_p = qbar*S*b^2*C_lp/(2*I_xx*U1);
L_r = qbar*S*b^2*C_lr/(2*I_xx*U1);
L_da = qbar*S*b*C_lda/I_xx;
L_dr = qbar*S*b*C_ldr/I_xx;

N_beta = qbar*S*b*C_nbeta/I_zz;
N_Tbeta = qbar*S*b*C_nTbeta/I_zz;
N_p = qbar*S*b^2*C_np/(2*I_zz*U1);
N_r = qbar*S*b^2*C_nr/(2*I_zz*U1);
```

```
N_da = qbar*S*b*C_nda/I_zz;
N_dr = qbar*S*b*C_ndr/I_zz;
```

1.2 Calculating A and B matrices

Please read Roskam book for detail derivation on page 318-320 [longitudinal] and page 348-349 [lateral-directional]. Also, homework 7 solution is presenting more detail.

```
 function [A_long, B_long, A_lat, B_lat]...
 = dynamic(U1, Z_alpha_dot, M_alpha_dot,...
           alpha, X_u, X_Tu, X_alpha, Z_u,...
           Z_alpha, Z_q, M_u, M_alpha,...
           M_Tu, M_q, X_de, Z_de, M_de,...
           I_xz, I_xx, I_zz,...
           Y_beta, Y_p, Y_r,...
           N_beta, N_Tbeta, L_beta, N_p, L_p, N_r, L_r,...
           Y_da, Y_dr, N_da, N_dr, L_da, L_dr)

theta_1 = alpha;
g = 32.2;
% U1 = U1*1.6878; %conversion from kts to ft/sec
%Longitudinal
k_long = [1,      0,        0,            0;
          0,      1,        0,            0;
          0,      0,   U1-Z_alpha_dot,    0;
          0,      0,    -M_alpha_dot ,    1];

A_long = (k_long)\[0,0,0,1;...
        -g*cosd(theta_1), X_u+X_Tu, X_alpha, 0;...
        -g*sind(theta_1), Z_u,     Z_alpha, U1+Z_q;...
         0,      M_u+M_Tu, M_alpha+M_Tu, M_q];
B_long = (k_long)\[0; X_de; Z_de; M_de];

%Lateral-directional
A_1 = I_xz/I_xx;
B_1 = I_xz/I_zz;
k_lat = [1,0,0,0,0;...
         0,1,0,0,0;...
         0,0,U1,0,0;...
         0,0,0,1,-A_1;...
         0,0,0,-B_1,1];
```

```
A_lat = k_lat\[0,0,0,1,0;...
                    0,0,0,0,1;...
    g*cosd(theta_1), 0, Y_beta, Y_p, Y_r-U1; ...
      0,0,L_beta,L_p,L_r;...
      0,0,N_beta+N_Tbeta,N_p,N_r];

B_lat = (k_lat)\[0,0; 0,0; Y_da, Y_dr; L_da, L_dr; N_da, N_dr];
end
```

1.3 Defining C and D matrices

When you define C and D matrices, you have to have the desired output. For example, if you desire the flight path angle, γ as the output, the equation looks like the following.

$$\gamma = \theta - \alpha \tag{3}$$

There is no control term so D matrix will be zero. For C matrix, be aware the dimension of the output first. You have just one desired output (1 by 1 matrix) so your C matrix will be 1 by 4 matrix to end up with 1 by 1 matrix.

$$y = C\overrightarrow{x} = \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ u \\ q \end{bmatrix} = \theta - \alpha = \gamma \tag{4}$$

What if you have two outputs as pitch rate and pitch angle (q and θ)?

- First, you have to consider the dimension of your desired output. In this case, you have 2 by 1 matrix. (It is column vector like the state vector) [You don't need the control term in this case either]
- Second, consider the dimension of the state vector. (4 by 1) So if you consider the multiplication of the matrix, your C matrix dimension will be 2 by 4 matrix.
- Lastly, form your C matrix.

$$y = C\overrightarrow{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ u \\ q \end{bmatrix} = \begin{bmatrix} \theta \\ q \end{bmatrix} \tag{5}$$

2 Some useful commands in MATLAB

- damp: this will show pole, natural frequency and damping ratio

  $$[Wn, Z, P] = damp(SYS)$$

  In SYS, you can input A matrix or transfer function. To define the transfer function, you have to use 'tf' command. If you type 'tf(num, den)' then you can get the transfer function. numerator and denominator will be defined row array with the coefficient of the polynomial.

- step: this will show the step response of your system

  $$step(sys)$$

- ss2tf: To use this function you need A,B,C and D matrices What this command does is that it will give the transfer function with given A,B,C and D matrix.

  $$[num, den] = ss2tf(A, B, C, D)$$

  IF you have multiple inputs, you have to specify the input when you use this command as follows.

  $$[num, den] = ss2tf(A, B, C, D, input)$$

  Input can be defined by just a number. (1,2,..., n)

- tf2ss: If you input the transfer function or system, this will give you the state space matrices.

  $$[A, B, C, D] = tf2ss(num, den)$$

2.1 Examples from Ref. 2

1. *tf2ss* simple example

```
num = [10 10];
den = [1 6 5 10];
[A,B,C,D] = tf2ss(num,den)
A =
    -6    -5   -10
     1     0     0
     0     1     0
B =
     1
     0
     0
C =
     0    10    10

D =
     0
```

2. *ss2tf* simple example: single input system

```
A = [0 1 0; 0 0 1; -5.008 -25.1026 -5.03247];
B = [0; 25.04; -121.005];
C = [1 0 0];
D = [0];
[num,den] = ss2tf(A,B,C,D)
num =     0         0   25.0400    5.0080

den =    1.0000    5.0325   25.1026    5.0080
```

3. *ss2tf* simple example: multiple input system

```
A = [0 1; -25 -4];
B = [1 1; 0 1];
C = [1 0; 0 1];
D = [0 0; 0 0];
[NUM, den] = ss2tf(A,B,C,D,1)

NUM =
     0     1     4
     0     0   -25
den =

    1.0000    4.0000   25.0000

[NUM, den] = ss2tf(A,B,C,D,2)

NUM =
         0    1.0000    5.0000
         0    1.0000  -25.0000

den =
    1.0000    4.0000   25.0000
```

4. Example with dynamic model from homework 8
   Please download 'Dynamic_model.mat' file from the blackboard.

```
load('Dynamic_model.mat')

damp(A_long)

%Single input and output case [Longitudinal]
C = [1 0 0 0];
D = zeros(1,1);

[num1, den1] = ss2tf(A_long, B_long, C, D)

sys1 = tf(num1,den1)

figure(1)
step(sys1)

%Multiple input and output case [Lateral-Directional]
C1 = [1 0 0 0 0; 0 0 0 0 1];
D1 = zeros(2,2);

[num2, den2] = ss2tf(A_lat, B_lat, C1, D1, 1)

sys2 = tf(num2(1,:),den2)

figure(2)
```

```
step(sys2)

sys3 = tf(num2(2,:),den2)

figure(3)
step(sys3)

[num4, den4] = ss2tf(A_lat, B_lat, C1, D1, 2)

sys4 = tf(num4(1,:),den4)
figure(4)
step(sys4)

sys5 = tf(num4(2,:),den4)
figure(5)
step(sys5)
```

3 SISO tool

SISO tool is for single input and single output system. It is very useful when you consider root locus and bode plot. Also, you can adjust gain to see how the system will behave. To access the sisotool, you can simply type 'sisotool' in MALTAB.
If you want to start with your system, you can type 'sisotool(sys)'. sys will be your transfer fucntion.

Session 2: April 24, 2015

1 SISO tool

In this session, we are going to explore SISO (Single Input / Single output) tool in MATLAB.

1.1 What is 'SISO' tool?

SISOtool allows you to design a single-input/single-output (SISO) compensator using root locus, Bode diagram, Nichols and Nyquist techniques. You can also automatically design a compensator using this GUI. [Ref. 1]

1.2 Why do we have to use this?

This tool is very effective to work for Root Locus and Bode plot. You can draw by hand but it will take a lot of time. Also, if you have a lot of poles, zeros, and complex conjugate, this tool will be generate plots for you and can design controllers easily.

1.3 How can you access this tool in MATLAB?

Type 'sisotool(sys)'. 'sys' should be the transfer function by using tf(num,den).

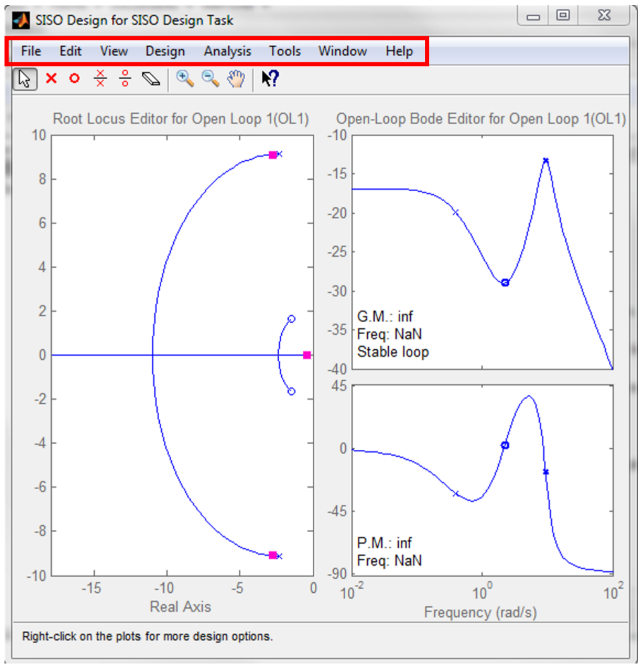

SISO Design for SISO Design Task
File
Edit
View
Design
Analysis
Tools
Window
Help
Root Locus Editor for Open Loop 1(OL1)
Real Axis
Open-Loop Bode Editor for Open Loop 1(OL1)
G.M.: inf
Freq: NaN
Stable loop
P.M.: inf
Freq: NaN
Frequency (rad/s)
Right-click on the plots for more design options.

Figure 1: *SISO tool GUI*

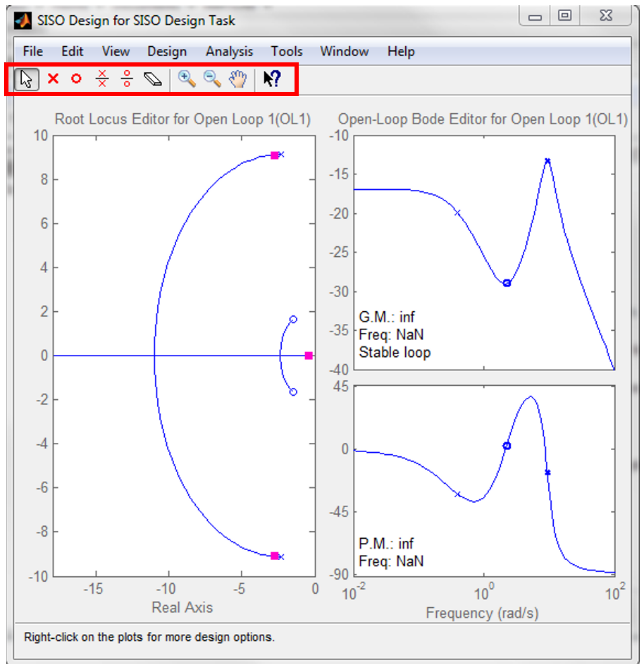
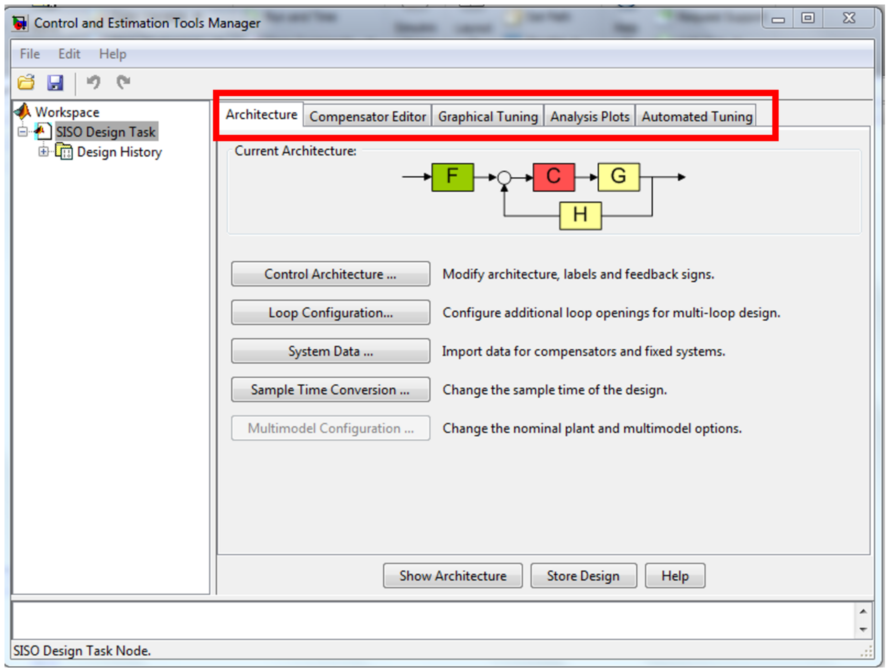

SISO Design for SISO Design Task
File
Edit
View
Design
Analysis
Tools
Window
Help
Root Locus Editor for Open Loop 1(OL1)
Real Axis
Open-Loop Bode Editor for Open Loop 1(OL1)
G.M.: inf
Freq: NaN
Stable loop
P.M.: inf
Freq: NaN
Frequency (rad/s)
Right-click on the plots for more design options.

Figure 2: *SISO tool GUI*



Modify architecture, labels and feedback signs.

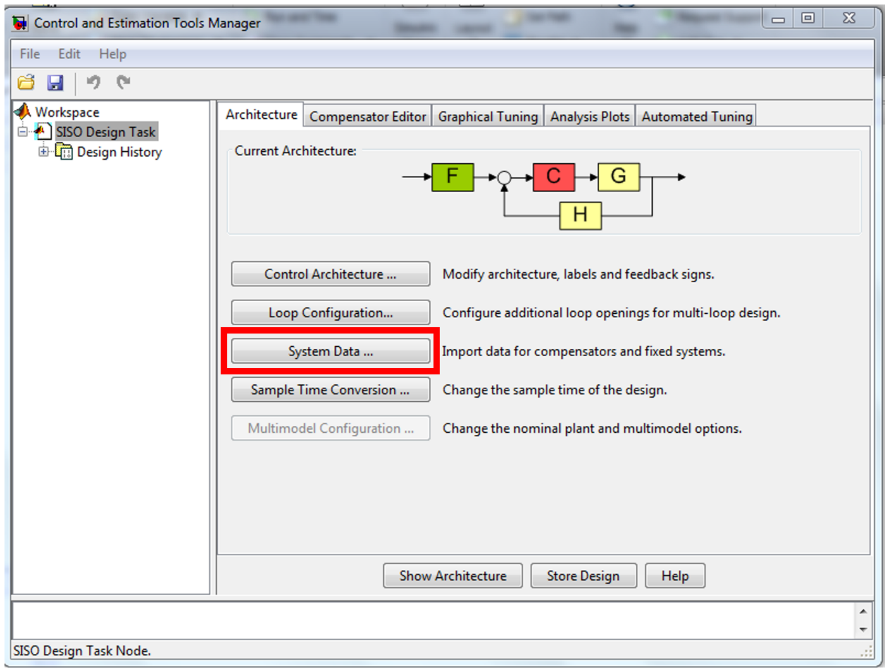
Configure additional loop openings for multi-loop design.

Import data for compensators and fixed systems.

Change the sample time of the design.

Change the nominal plant and multimodel options.

Figure 3: *SISO tool control panel*



Modify architecture, labels and feedback signs.

Configure additional loop openings for multi-loop design.

Import data for compensators and fixed systems.

Change the sample time of the design.

Change the nominal plant and multimodel options.

Figure 4: *SISO tool control panel*

1.4 Example

Let's do an example for sisotool. Please load the data file from Blackboard. Then, make a transfer function from lateral-directional equations.

```
load('Dynamic_model.mat')
%Multiple input and output case [Lateral-Directional]
C1 = [1 0 0 0 0];
D1 = zeros(1,2);
[num2, den2] = ss2tf(A_lat, B_lat, C1, D1, 1)
sys = tf(num2(1,:),den2)
```

Then, if you enter 'sisotool(sys)' in the command window, you should see the following figure.

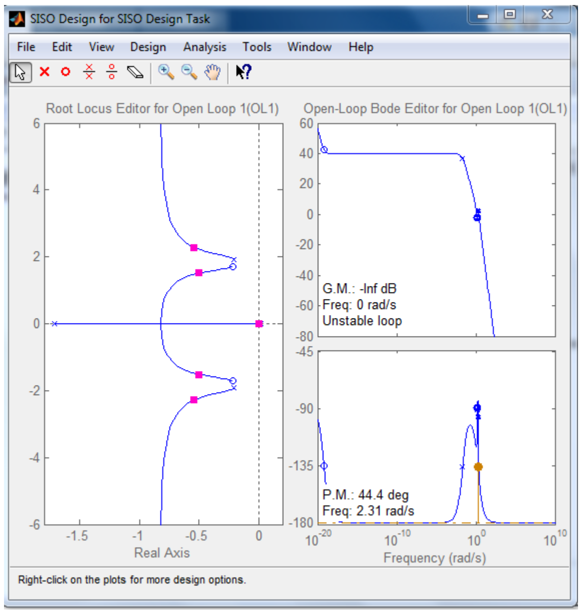

SISO Design for SISO Design Task
File Edit View Design Analysis Tools Window Help
Root Locus Editor for Open Loop 1(OL1)
Open-Loop Bode Editor for Open Loop 1(OL1)
Real Axis
Frequency (rad/s)
G.M.: -Inf dB
Freq: 0 rad/s
Unstable loop
P.M.: 44.4 deg
Freq: 2.31 rad/s
Right-click on the plots for more design options.

Figure 5: *Example 1 sisotool window*

Try to add one pole (@ -4) and zero (@ -1) using following buttons.

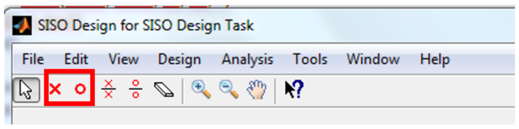

SISO Design for SISO Design Task
File Edit View Design Analysis Tools Window Help

Figure 6: *Buttons for adding poles and zeros*

Note: Following buttons are for adding complex conjugate of zeros and poles.

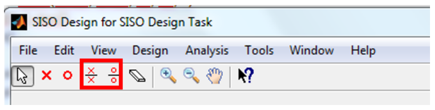

SISO Design for SISO Design Task
File Edit View Design Analysis Tools Window Help

Figure 7: *Buttons for adding complex conjugate poles and zero*

After you add poles and zeros, you should see the result as follows. Trick: You can adjust the location of poles and zeros in

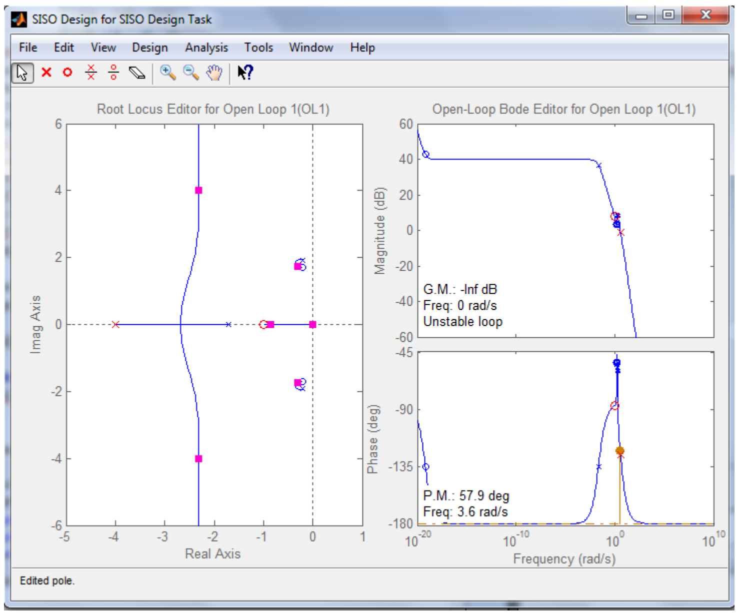
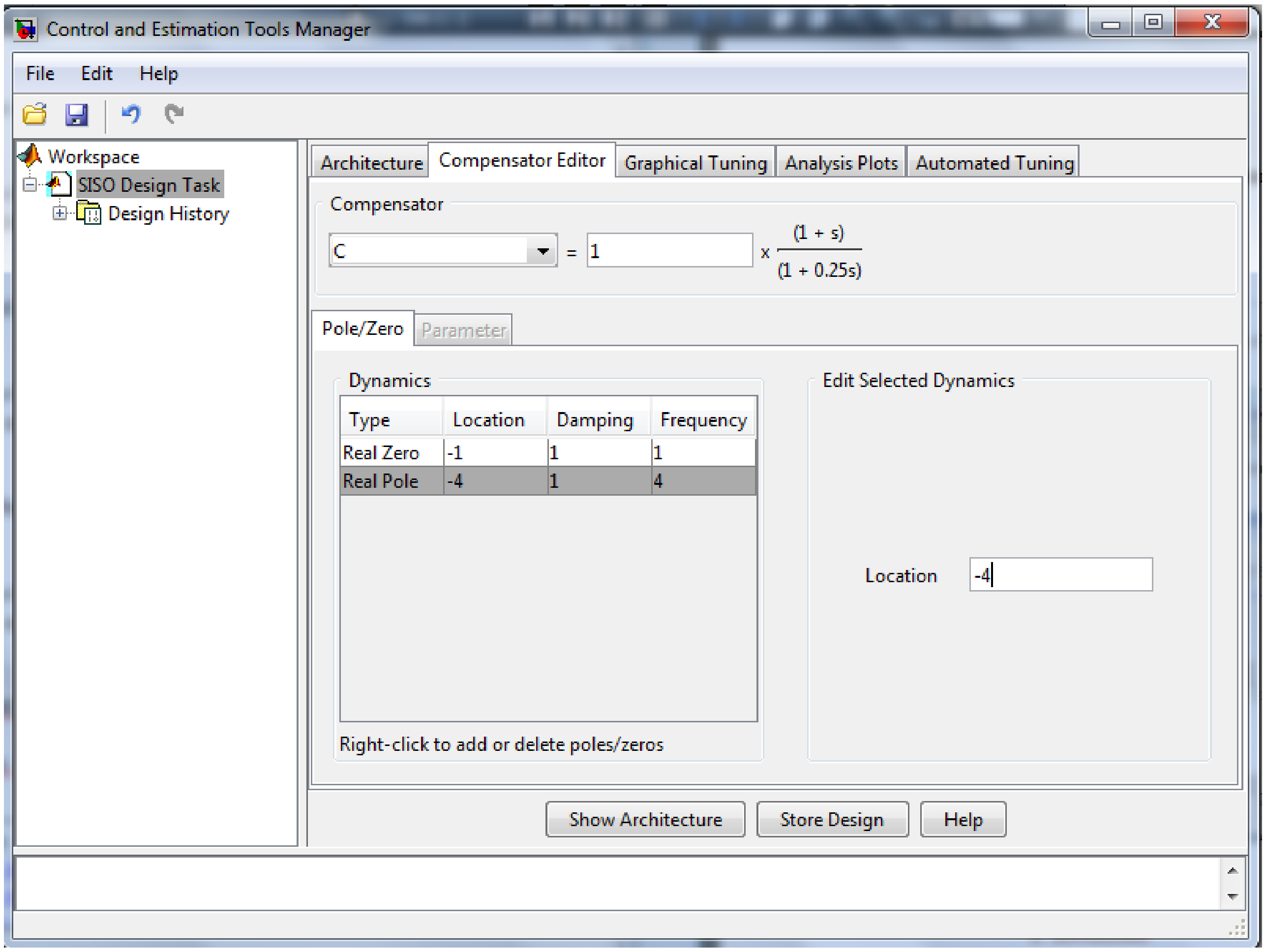

SISO Design for SISO Design Task
File Edit View Design Analysis Tools Window Help
Root Locus Editor for Open Loop 1(OL1)
Imag Axis
Real Axis
Open-Loop Bode Editor for Open Loop 1(OL1)
Magnitude (dB)
G.M.: -Inf dB
Freq: 0 rad/s
Unstable loop
Phase (deg)
P.M.: 57.9 deg
Freq: 3.6 rad/s
Frequency (rad/s)
Edited pole.

Figure 8: *Adding a pole and zero*

the following panel. Also, in this panel, you can add or remove the poles and zeros as well. In addition, you can adjust the gain as much as you want.

- Important points
  - The transfer function was presented in Bode form. (The way you learned, meaning the coefficient of s was 1, is call Root Locus form.)



$C = 1 \times \frac{(1+s)}{(1+0.25s)}$

Type	Location	Damping	Frequency
Real Zero	-1	1	1
Real Pole	-4	1	4

Location: -4

Figure 9: *Compensator panel*

Let's see the step response. You should see the following result.

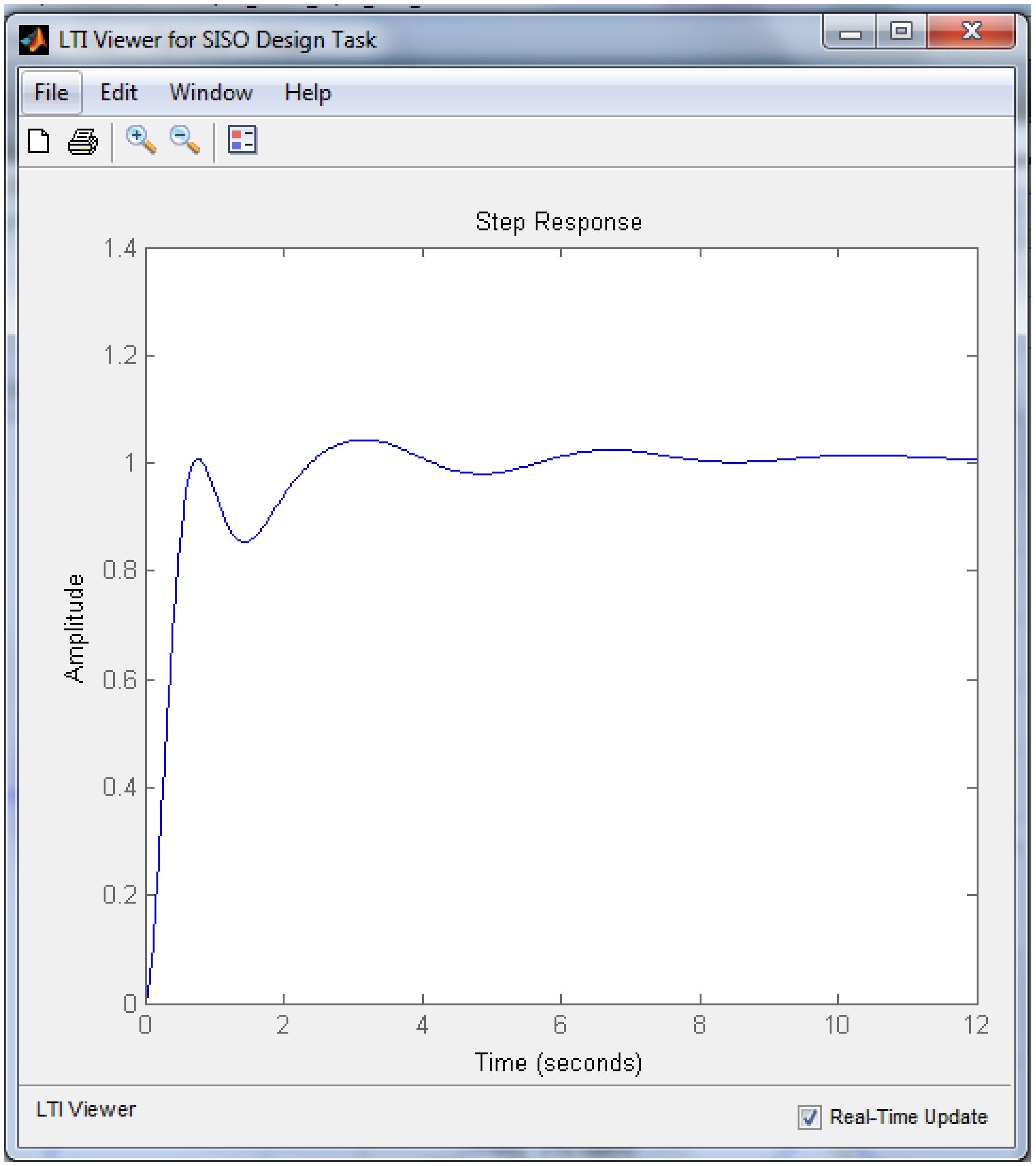


Figure 10: *Step Response*

2 SIMULINK

In this section, we are going to explore SIMULINK application in MATLAB.

2.1 What is SIMULINK?

Simulink® is a block diagram environment for multidomain simulation and Model-Based Design. It supports simulation, automatic code generation, and continuous test and verification of embedded systems.

Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB®, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis. [Ref. 1]

2.2 What is the significance of SIMULINK?

SISOtool deals only with the single input and single output. But SIMULINK is capable of multi-input and multi-output.

2.3 Access to SIMULINK

You can enter 'simulink' in the command window or you can use the following button in MATLAB window.

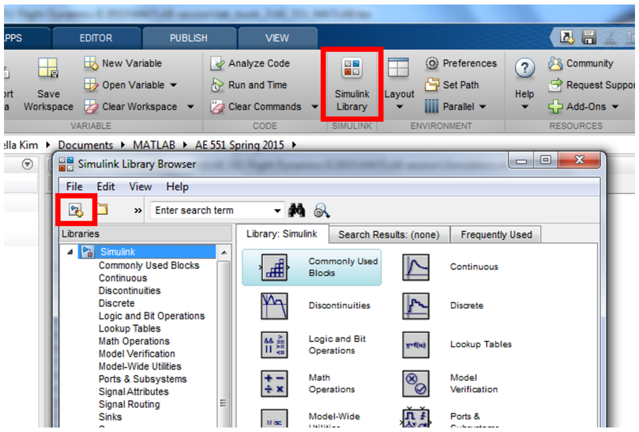


Figure 11: *Access to SIMULINK*

Let's make simple diagram as follows. You need step, sum, transfer function, and scope block from the library.

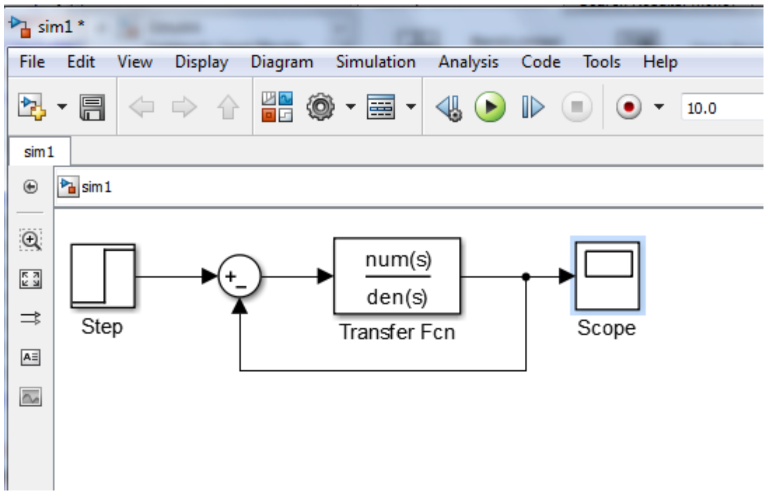


Figure 12: *Example of closed loop diagram*

May 1, 2015

Today's lab will focus MIMO (multiple input and multiple output) system when you use SIMULINK. For more detail of information, please look Powerpoint slide from Dr. Downing.

1 MIMO system

1.1 What is MIMO system?

It is multi-input, multi-output system. Recall SISOtool in the last session. For MIMO system, SIMULINK will be used to simulate.

1.2 Example in SIMULINK

We are going to deal with the lateral-directional equations only to see MIMO system.

1. Cessna 310

$$A_{lat} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0.1030 & 0 & -0.2495 & -0.0030 & -0.9925 \\ 0 & 0 & -7.2654 & -2.1565 & 0.2853 \\ 0 & 0 & 7.7303 & -0.0812 & -0.4725 \end{bmatrix} \tag{6}$$

$$B_{lat} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0.0822 \\ 11.4020 & 1.2728 \\ -0.8994 & -6.1671 \end{bmatrix} \tag{7}$$

We need new state space matrices. For including the two control surfaces, the state vector will look like the following.

$$\overrightarrow{\mathbf{x}} = \begin{bmatrix} \phi \\ \psi \\ \beta \\ p \\ r \\ \delta_a \\ \delta_r \end{bmatrix} \tag{8}$$

To do this, we need the servo dynamics. This was assumed same as the powerpoint notes.

$$\dot{\delta_a} = -10\delta_a + 10\delta_{acmd} \tag{9}$$
$$\dot{\delta_r} = -10\delta_r + 10\delta_{rcmd} \tag{10}$$

Then, the new A and B matrices will be the following.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.1030 & 0 & -0.2495 & -0.0030 & -0.9925 & 0 & 0.0822 \\ 0 & 0 & -7.2654 & -2.1565 & 0.2853 & 11.4020 & 1.2728 \\ 0 & 0 & 7.7303 & -0.0812 & -0.4725 & -0.8994 & -6.1671 \\ 0 & 0 & 0 & 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -10 \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 10 & 0 \\ 0 & 10 \end{bmatrix} \tag{12}$$

Let's assume C and D matrix as follows:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$D = zeros(7,2); \tag{14}$$

Find the gain for control from Root locus. You have to make state space matrices separately in terms of δ_a and δ_r. You can use SISOtool or 'rlocus' command in MATLAB. I picked the gain as follows:

$$K_\phi = 0.1 \tag{15}$$

$$K_\psi = 2 \tag{16}$$

$$K_r = 0.02 \tag{17}$$

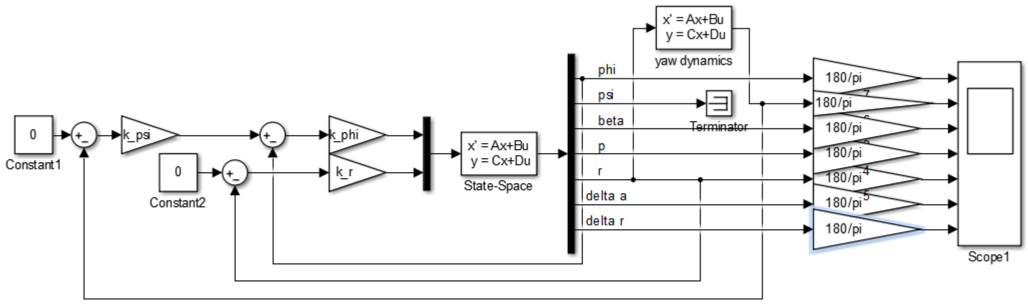
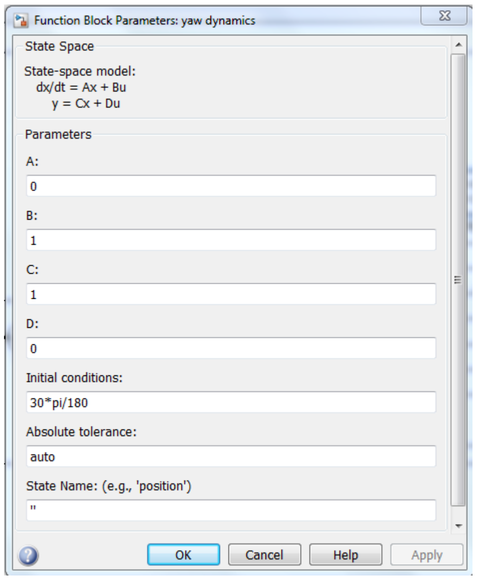


Figure 13: *SIMULINK diagram*


Function Block Parameters: yaw dynamics

State Space

State-space model:
dx/dt = Ax + Bu
y = Cx + Du

Parameters

A:
0

B:
1

C:
1

D:
0

Initial conditions:
30*pi/180

Absolute tolerance:
auto

State Name: (e.g., 'position')
''

Figure 14: *Yaw dynamics state space*

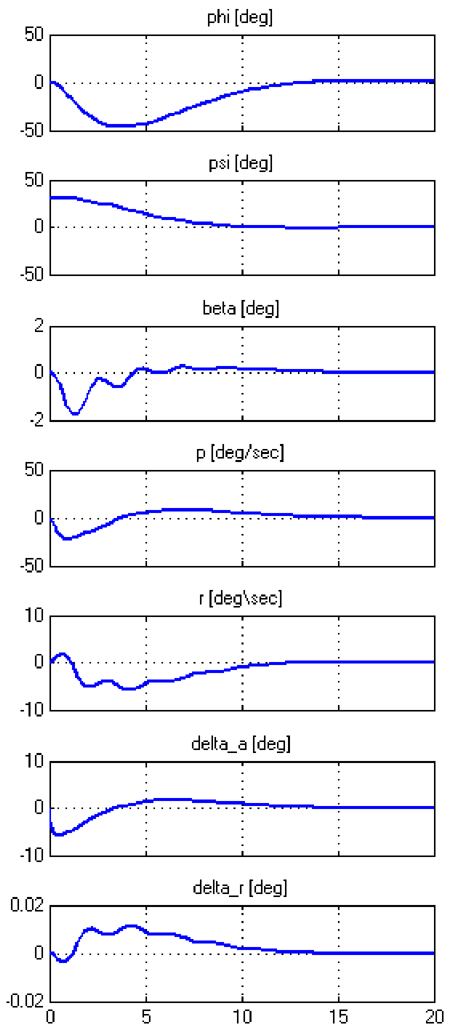


Figure 15: *Simulation result*

- Summary
  1. To find the gain for the controller, form state space matrices.
  2. If you found the gain, form SIMULINK diagram.
  3. Don't forget to put yaw dynamics block.
  4. Simulate and compare the result.

Bibliography

[1] MATLAB Documentation

[2] Katsuhiko Ogata, Modern Control Engineering, Prentice Hall, Sep.4, 2009