9 Additional Topics

9.1 Combined Analysis

Combined analysis

Purpose : use all the tools to evaluate the system stability

Time domain: Root locus of G
- Step response of G_{CL}

Frequency domain:
- Nyquist plot of G
- Bode plots of G: margins < gain K_g, phase γ

Given: G(s)

Find: combined analysis : Quad plot

Nyquist plot	Margins on Bode plot
Root locus for Gain=1	Step response

Examples : open MATLAB codes

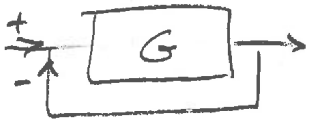
aircraft roll mode
- Nyquist plot : STABLE
- root locus —"—
- step response —"—

Bode plot : INSUFFICIENT phase margin

B11.9 - STABLE : N-plot
r locus
step response
- INSUFFICIENT phase margin

B11.8 UNSTABLE

Combined analysis of aircraft roll model.



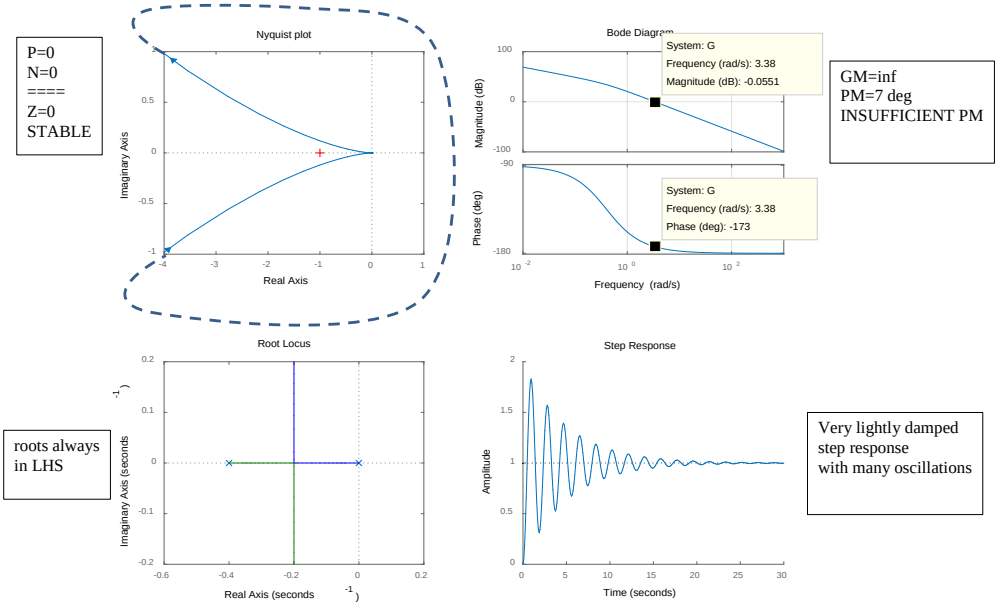
$$G(s) = \frac{K}{Js^2 + cs} = \frac{114}{10s^2 + 4s}$$

$p_1 = 0$

$p_2 = -0.4$.

See plot

455/523


P=0
N=0
====
Z=0
STABLE
Nyquist plot
Imaginary Axis
Real Axis
Bode Diagram
System: G
Frequency (rad/s): 3.38
Magnitude (dB): -0.0551
Magnitude (dB)
System: G
Frequency (rad/s): 3.38
Phase (deg): -173
Phase (deg)
Frequency (rad/s)
GM=inf
PM=7 deg
INSUFFICIENT PM
Root Locus
Imaginary Axis (seconds -1)
Real Axis (seconds -1)
roots always
in LHS
Step Response
Amplitude
Time (seconds)
Very lightly damped
step response
with many oscillations

2017 04 17

Aircraft roll model combined analysis (cont.)

Nyquist.

✓ stable

$Z = 0$.

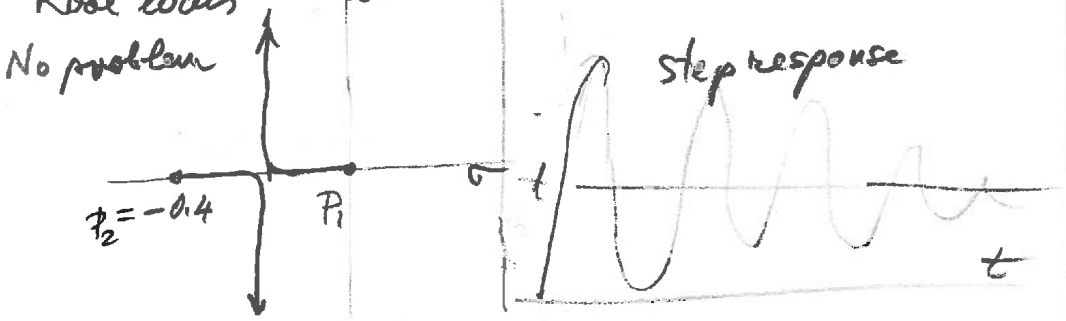
Bode Margins

$(K_g)_{dB} = \infty$ ✓

$\gamma^\circ = 7^\circ$ Not sufficient

Root locus

No problem



Step response

Conclusion

- Nyquist and root locus indicate stable system.
- Margin analysis warns about low phase margin: → need compensator
- step response — oscillatory

457/523

Problem B 11.9

$$G(s) = \frac{20(s^2 + s + 0.5)}{s(s-1)(s+10)}$$

$p_1 = 0$

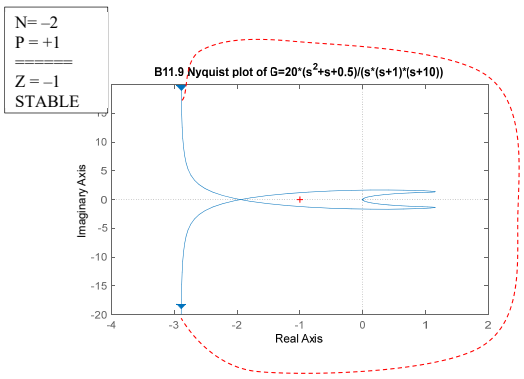
$p_2 = +1$ } $P = 1$

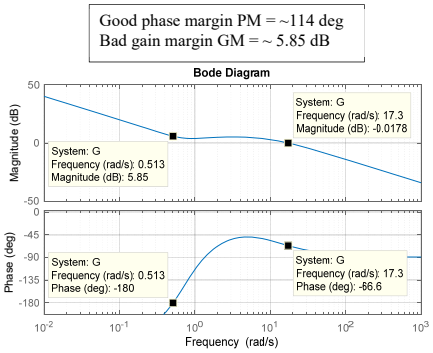
$p_3 = -10$

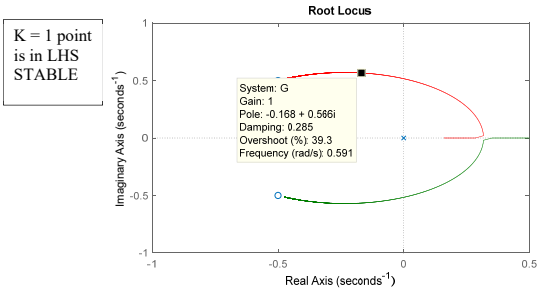
Nyquist	Bode
root locus	step response

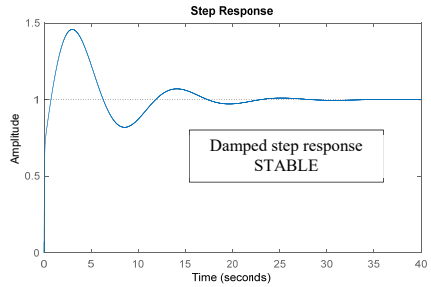
see plot

458/523


N= –2
P = +1
======
Z = –1
STABLE
B11.9 Nyquist plot of G=20*(s^2+s+0.5)/(s*(s+1)*(s+10))
Imaginary Axis
Real Axis

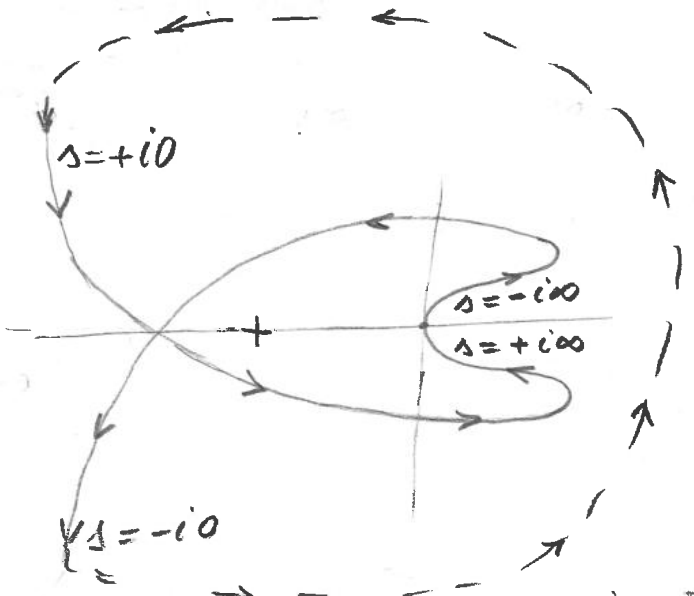

Good phase margin PM = ~114 deg
Bad gain margin GM = ~ 5.85 dB
Bode Diagram
Magnitude (dB)
Phase (deg)
System: G
Frequency (rad/s): 0.513
Magnitude (dB): 5.85
System: G
Frequency (rad/s): 17.3
Magnitude (dB): -0.0178
System: G
Frequency (rad/s): 0.513
Phase (deg): -180
System: G
Frequency (rad/s): 17.3
Phase (deg): -66.6
Frequency (rad/s)

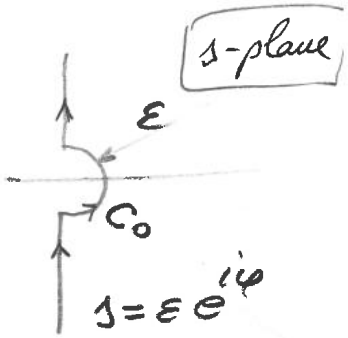

K = 1 point
is in LHS
STABLE
Root Locus
Imaginary Axis (seconds^-1)
Real Axis (seconds^-1)
System: G
Gain: 1
Pole: -0.168 + 0.566i
Damping: 0.285
Overshoot (%): 39.3
Frequency (rad/s): 0.591


Step Response
Amplitude
Time (seconds)
Damped step response
STABLE

Problem B11.9 (cont.)

Nyquist plot





$$G^\circ = \frac{\bar{z}_0}{-1} = -2\,\frac{1}{\varepsilon}\,e^{-i\varphi}$$

$$G^\circ = \frac{2}{\varepsilon}\,e^{i(180^\circ-\varphi)}$$

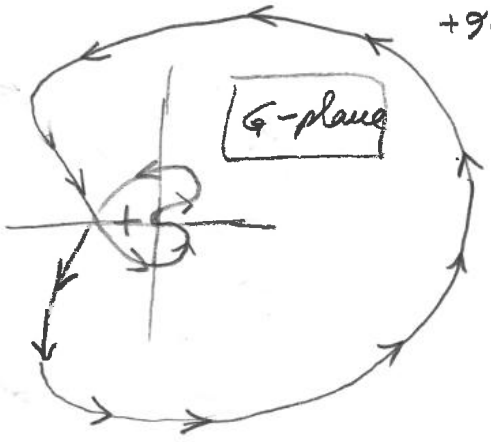
$-90^\circ < \varphi < 90^\circ$

$s = \varepsilon e^{i\varphi}$

$-90^\circ < \varphi < 90^\circ$

$-90^\circ < \angle G^\circ < -270^\circ$

$+90^\circ$



$N = -2$

$P = 1$

$Z = -1$

Stable

460/523

(B11.9 cont.)

Bode plot

Good phase margin: $\gamma \approx 114°$ @ $16.6 \frac{rad}{sec}$

Bad gain margin (at low freq.): $K_g = -5.3dB$ @ $0.55 \frac{rad}{sec}$

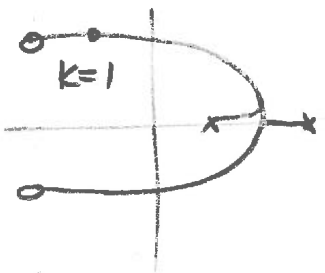
Good gain margin at higher $\omega > 16 \frac{rad}{sec}$, $\angle G$ ($\angle G$ does not cross $-180°$)

There could be problems at low ω ($\omega \sim 0.5$ rad/sec) due to imperfections

Root locus

K=1 in LHS

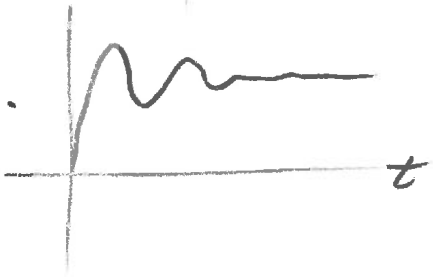
Stable



Step response

damped oscillation.

STABLE



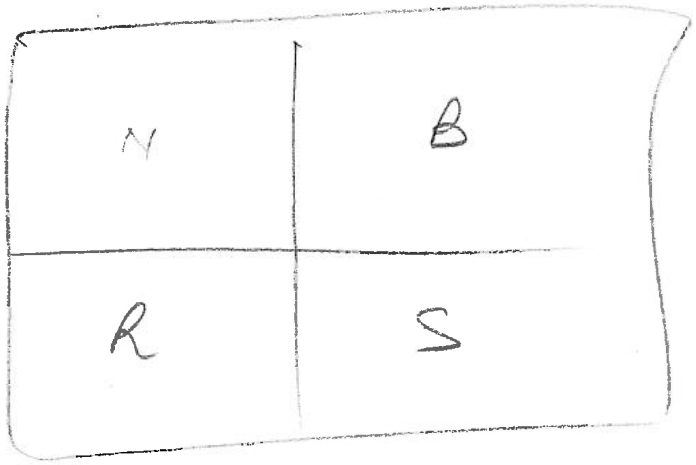
461/523

Problem B11.8

$$G(s) = \frac{320(s+2)}{s(s+1)(s^2+8s+44)}$$

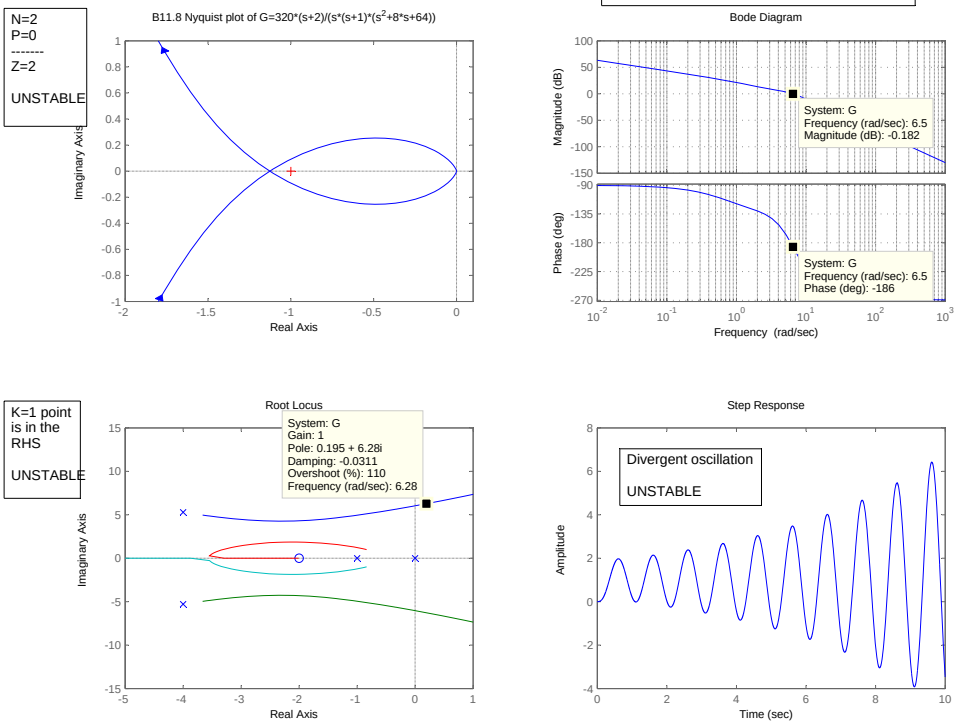
4 Poles: 0
-1
$-4 \pm i5.3$

all in LHS



see plot

462/523

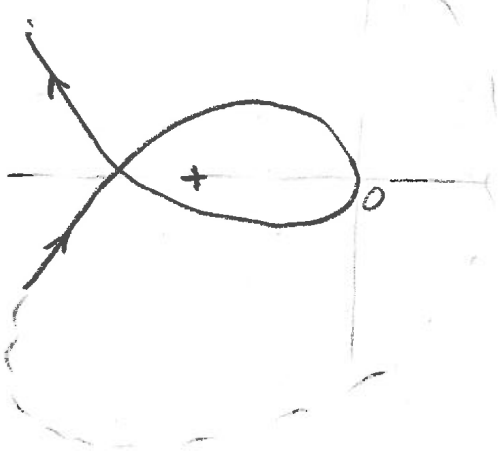

N=2
P=0

Z=2
UNSTABLE
B11.8 Nyquist plot of G=320*(s+2)/(s*(s+1)*(s^2+8*s+64))
Imaginary Axis
Real Axis
Negative phase and gain margins: UNSTABLE
Bode Diagram
Magnitude (dB)
System: G
Frequency (rad/sec): 6.5
Magnitude (dB): -0.182
Phase (deg)
System: G
Frequency (rad/sec): 6.5
Phase (deg): -186
Frequency (rad/sec)
K=1 point is in the RHS
UNSTABLE
Root Locus
System: G
Gain: 1
Pole: 0.195 + 6.28i
Damping: -0.0311
Overshoot (%): 110
Frequency (rad/sec): 6.28
Imaginary Axis
Real Axis
Step Response
Divergent oscillation
UNSTABLE
Amplitude
Time (sec)

463/523

Problem B 11.8 (cont.)

Nyquist plot



$P = 0$ no poles in RHS

$N = 2$

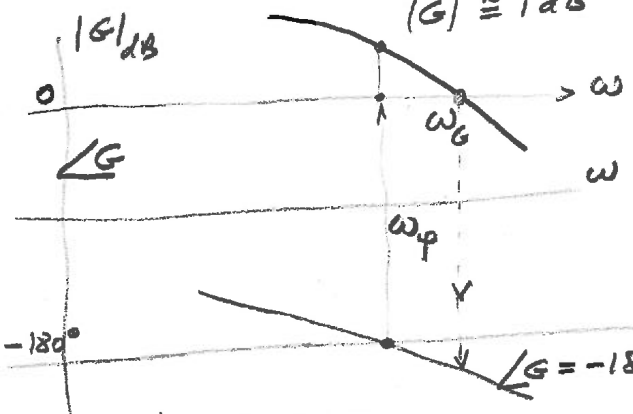
$Z = P + N = 2$ zeros in RHS of $1 + G(s)$

UNSTABLE

because

$$G_{CL} = \frac{G(s)}{1 + G(s)}$$

Bode plot : margins



$K_g = -1\,dB < 0\,dB$

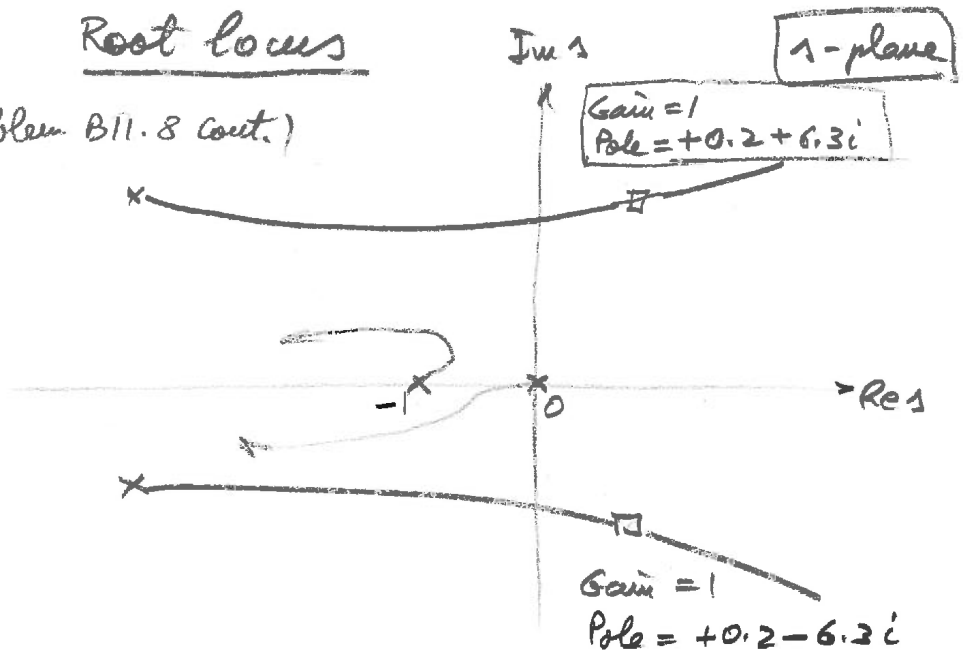
$\gamma = -5° < 0$

UNSTABLE

464/523

Root locus

(Problem B11.8 cont.)

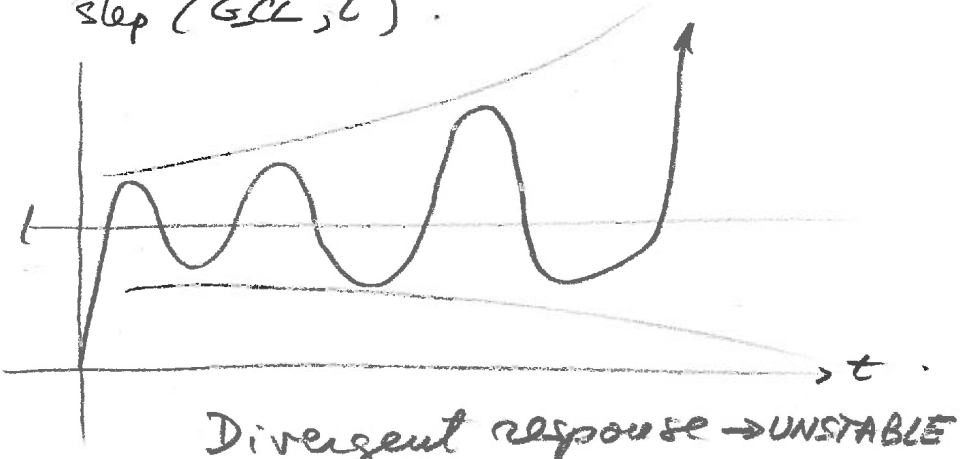


Roots in RHS → UNSTABLE.

Step response

G_{CL} = feedback(G,1).

step(GCL, t).



Divergent response → UNSTABLE

All four criteria predicted an UNSTABLE closed loop response.

ACTION: DO NOT close the loop!

9.2 Control Systems Designer (MATLAB SISO Tool)

CSD

Control System Designer (CSD) tool is an interactive GUI environment for analyzing and modifying feedback control systems.

- CSD was introduced in R2015a
- previous, we used "SISOTool"

To start CSD, type in Command Window:

```
>> controlSystemDesigner
```

The legacy sisotool command is also recognized:

```
>> sisotool
```

Before starting CSD, run a MATLAB program to create the basic system G/S).

- Single input, single output (SISO)
- Graphical User Interface (GUI)
- Requires separate definition of plant transfer function G/S) through an m file

467/523

v

CSD

Control System Designer

Design single-input, single-output (SISO) controllers

Description

The **Control System Designer** app lets you design single-input, single-output (SISO) controllers for feedback systems modeled in MATLAB or Simulink (requires Simulink Control Design™ software).

Using this app, you can:

- Design controllers using:
  - Interactive Bode, root locus, and Nichols graphical editors for adding, modifying, and removing controller poles, zeros, and gains.
  - Automated PID, LQG, or IMC tuning.
  - Optimization-based tuning (requires Simulink Design Optimization™ software).
  - Automated loop shaping (requires Robust Control Toolbox™ software).
- Tune compensators for single-loop or multiloop control architectures.
- Analyze control system designs using time-domain and frequency-domain responses, such as step responses and pole-zero maps.
- Compare response plots for multiple control system designs.
- Design controllers for multimodel control applications.

Open the Control System Designer App

- MATLAB Toolstrip: On the **Apps** tab, under **Control System Design and Analysis**, click the app icon.
- MATLAB command prompt: Enter `controlSystemDesigner`.
- Simulink model editor: Select **Analysis > Control Design > Control System Designer**.

Examples

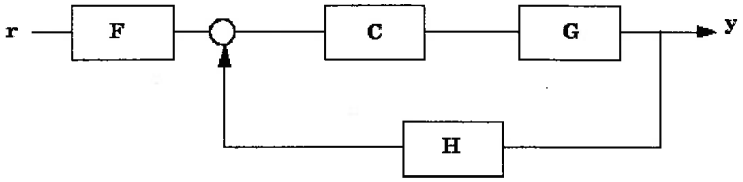
- "Control System Designer Tuning Methods"

3 CSD

- "Bode Diagram Design"
- "Root Locus Design"
- "Design Compensator Using Automated Tuning Methods"
- "Design Multiloop Control System"
- "Analyze Designs Using Response Plots"
- "Compare Performance of Multiple Designs"
- "Multimodel Control Design"

Programmatic Use

`controlSystemDesigner` opens the **Control System Designer** app using the following default control architecture:



The architecture consists of the LTI objects:

- G — Plant model
- C — Compensator
- H — Sensor model
- F — Prefilter

By default, the app configures each of these models as a unit gain.

`controlSystemDesigner(plant)` initializes the plant, G, to `plant`. `plant` can be any SISO LTI model created with `ss`, `tf`, `zpk` or `frd`, or an array of such models.

`controlSystemDesigner(plant,comp)` initializes the compensator, C, to the SISO LTI model `comp`.

4 CSD

Control System Designer

`controlSystemDesigner(plant,comp,sensor)` initializes the sensor model, *H*, to `sensor`. `sensor` can be any SISO LTI model or an array of such models. If you specify both `plant` and `sensor` as LTI model arrays, the lengths of the arrays must match.

`controlSystemDesigner(plant,comp,sensor,prefilt)` initializes the prefilter model, *F*, to the SISO LTI model `prefilt`.

`controlSystemDesigner(views)` opens the app and specifies the initial graphical editor configuration. `views` can be any of the following character vectors, or a cell array of multiple character vectors.

- `'rlocus'` — Root locus editor
- `'bode'` — Open-loop Bode Editor
- `'nichols'` — Open-loop Nichols Editor
- `'filter'` — Bode Editor for the closed-loop response from prefilter input to the plant output

In addition to opening the specified graphical editors, the app plots the closed-loop, input-output step response.

`controlSystemDesigner(views,plant,comp,sensor,prefilt)` specifies the initial plot configuration and initializes the plant, compensator, sensor, and prefilter using the specified models. If a model is omitted, the app uses the default value.

`controlSystemDesigner(initData)` opens the app and initializes the system configuration using the initialization data structure `initdata`. To create `initdata`, use `sisoinit`.

`controlSystemDesigner(sessionFile)` opens the app and loads a previously saved session. `sessionFile` is the name of a session data file on the MATLAB path. This data includes the current system architecture and plot configuration, and any designs and responses saved in the **Data Browser**.

To save a session, in the **Control System Designer** app, on the **Control System** tab, click **Save Session**.

See Also

Apps
Control System Tuner

2-137

470/523

5
CSD

2 Functions — Alphabetical List

Functions
`pidTuner | sisoinit`

Introduced in R2015a

‘Control System Designer’ MATLAB Tool

Aircraft Roll Motion Autopilot Development
PD Example

Table of Contents:

1 CONTROL SYSTEM DESIGN OBJECTIVES

Consider the aircraft roll transfer function $G(s) = \frac{K}{Js^2 + cs} = \frac{114}{10s^2 + 4s}$

Design a feedback control system to control the aircraft roll motion with the following control design objectives:

Design Objective 1: Control the unconstraint aircraft motion resulting from an aileron input. Have a autopilot system that can maintain the aircraft at a constant bank angle

We will achieve this objective through feedback (FB)

Design Objective 2: Achieve a reasonable aircraft roll response.

Design Objective 3: Ensure safe and stable operation of the feedback control system

We define 'reasonable response' using two control design specifications:

- DS1: Fast response time as measured by rise time
  $t_r \le 1.5$ sec
- DS2: maximum percentage overshoot for step input less than 20%
  $M_p \le 20\%$

We ensue safe and stable operation of the feedback control system by meeting a third design specification based on stability margins:

- DS3: $GM = 10\,\text{dB}$, $PM = 60°$

2 CREATE PLANT TRANSFER FUNCTION G

Run m-file `"aircraft_roll_model.m"` to create the plant transfer function G in the Workspace (Figure 1)

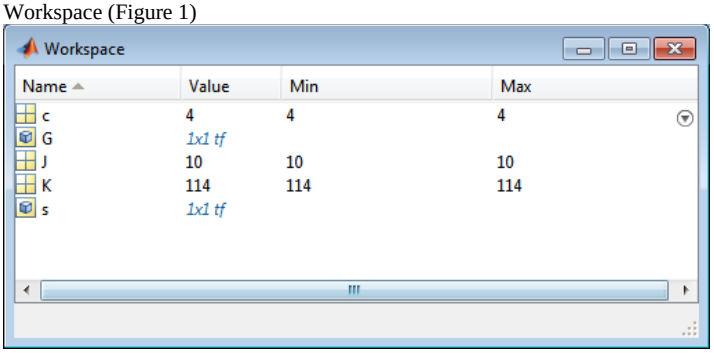


Figure 1

3 CREATE 'CONTROL SYSTEM DESIGNER' MODEL

3.1 OPEN 'CONTROL SYSTEM DESIGNER' SOFTWARE TOOL

In the Command Window, type "`controlSystemDesigner`"
`>>controlSystemDesigner`
(legacy name "`sisotool`", also works, i.e., `>>sisotool`)

The Control System Designer GUI opens as shown in Figure 2.

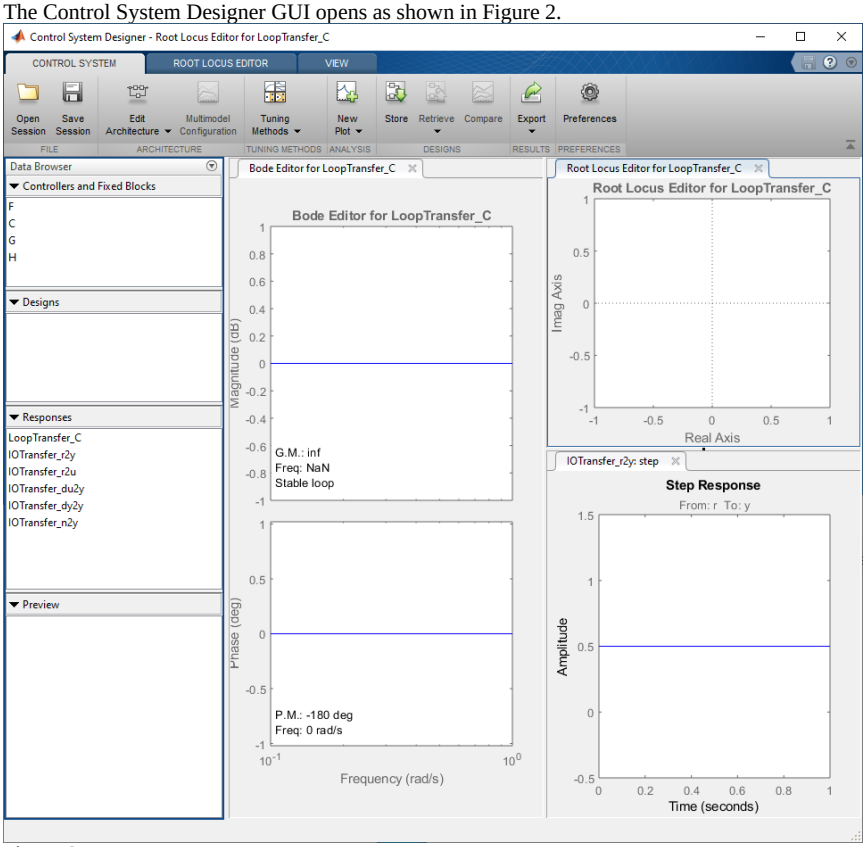


Figure 2

The default configuration has four blocks (F, C, G, H) as indicated in the LHS pane. All the blocks in this configuration are set to be unitary transfer functions by default (the plant block G, which is selected in the upper LHS pane, is shown in the lower LHS pane with Value: 1).

3.2 IMPORT PLANT TRANSFER FUNCTION G

Press “Edit Architecture” button to import the system G data from the Workspace (Figure 3).

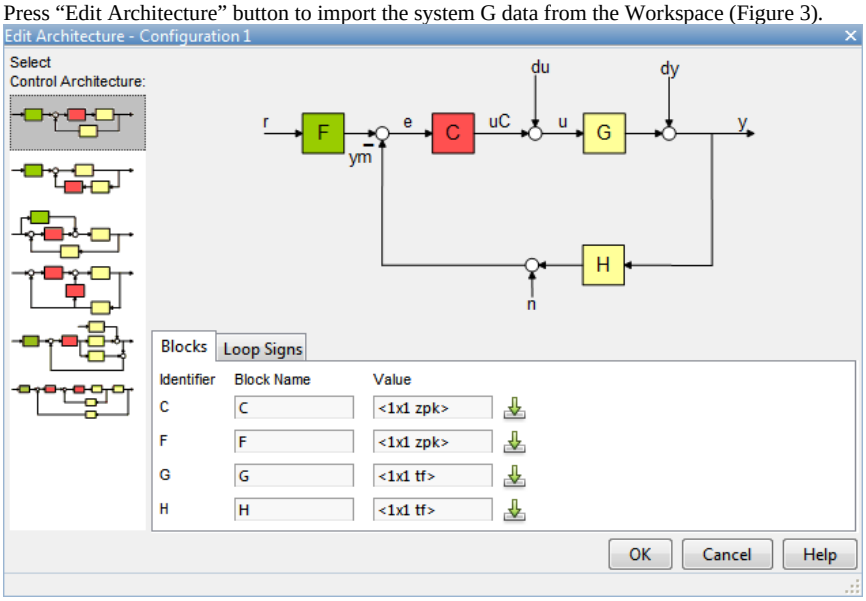


Figure 3

Note that the first control system configuration is selected by default (this is the configuration we work with, no need to change it.)
Press the down arrow icon at the RHS end of the G row to open the “Import Data for G” dialog box and select ‘G’ from the Base Workspace (Figure 4).

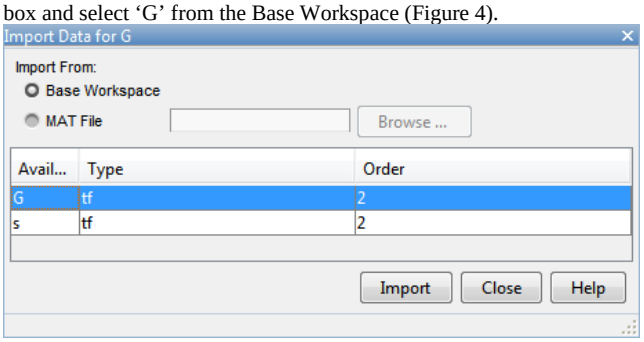


Figure 4

Press ‘Import’ to return to ‘Edit Architecture’ dialog box and press OK to return to the main view.

Now, our system $G(s) = \dfrac{114}{10s^2 + 2s}$ is shown in the lower LHS pane of the model (Figure 5)

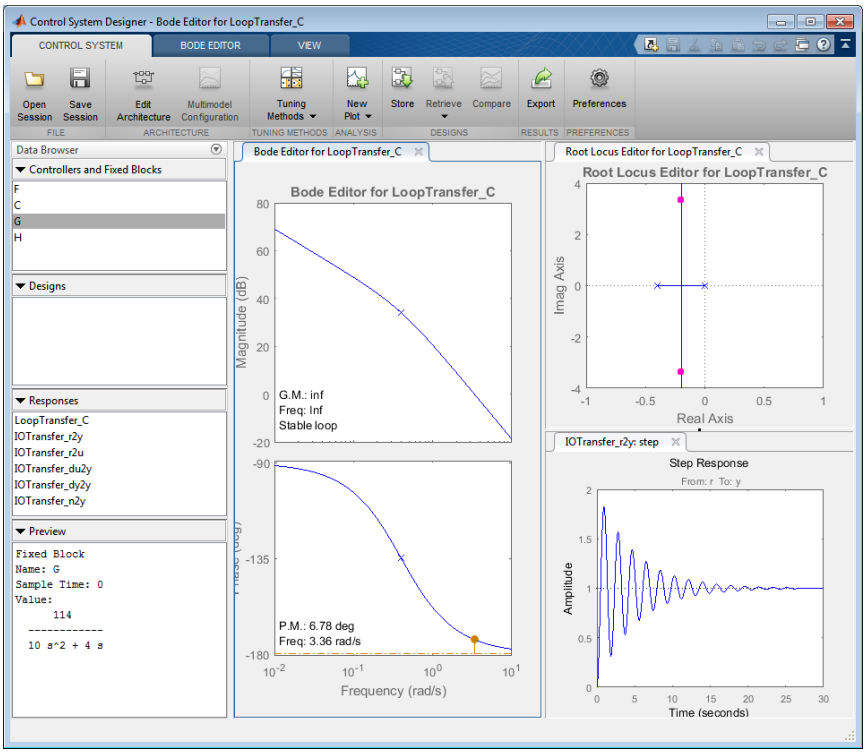


Figure 5

3.3 ADD DESIGN SPECIFICATIONS

Right-click on the Bode Editor window and select Design Requirements → New. The following window shown in Figure 6 appears:

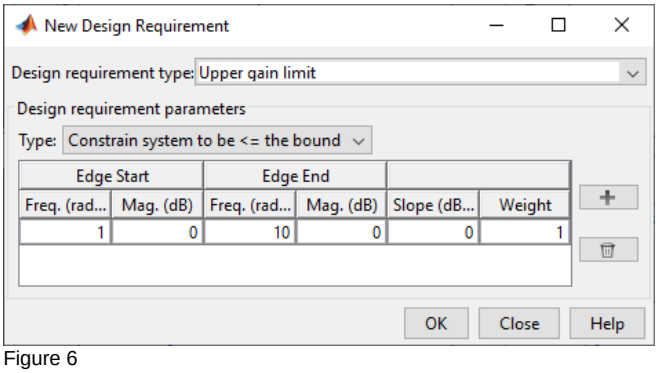


Figure 6

From the pull-down menu "Design requirement type" select "Gain & Phase margins" and enter the required DS3 values 10dB and 60deg to make the window look like shown in Figure 7.

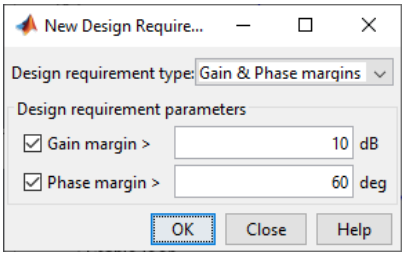


Figure 7

Press OK. Your screen should look like Figure 8.

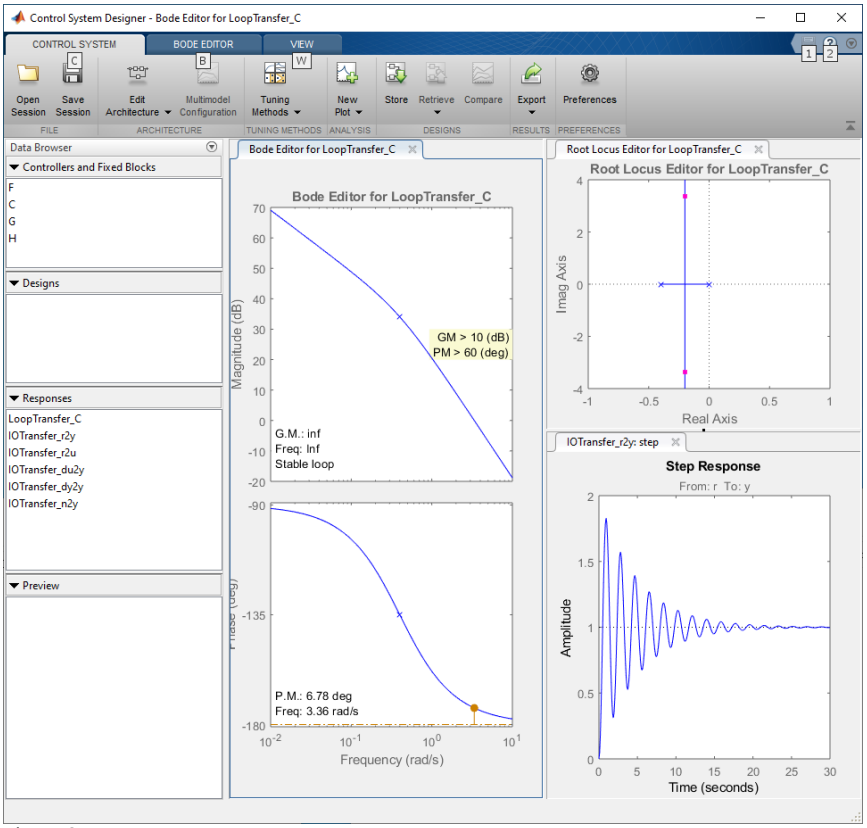


Figure 8

3.4 STORE THE BASELINE DESIGN AS G

Press ‘Store’ button to store this design; it will shown as ‘Design1’ in the second LHS pane. Save the CSD Sesion: press ‘Save Session’ button and save your session with the name ‘CSD_aircraft_Session_20200116’. The extension of the file, if shown, should be ‘.mat’. Your screen should look like Figure 9.

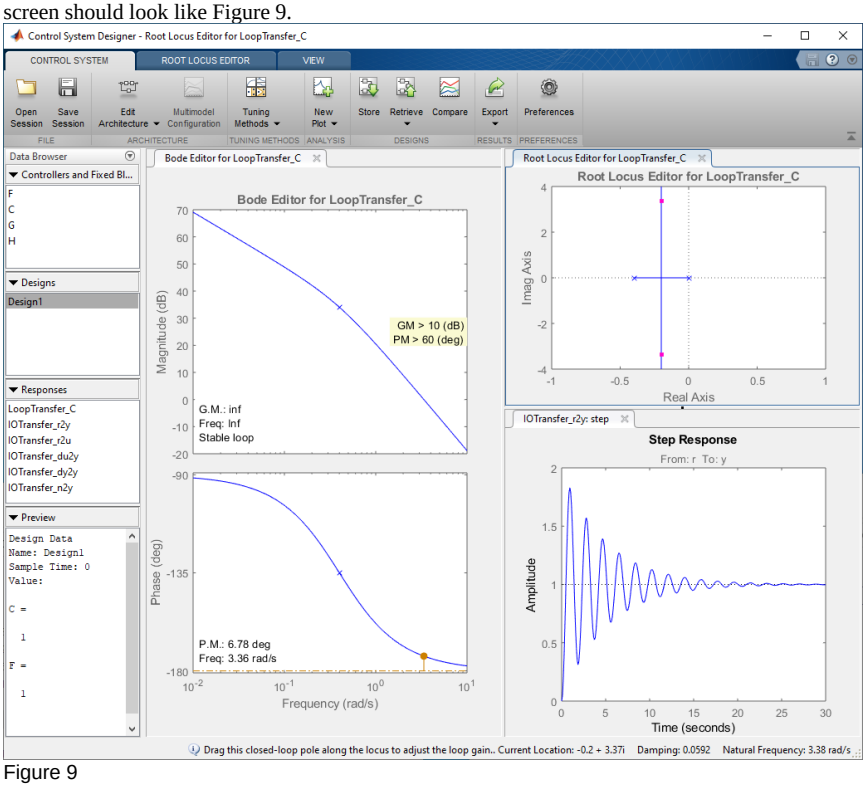


Figure 9

3.5 Record the Baseline Performance and Stability Reading of G Design

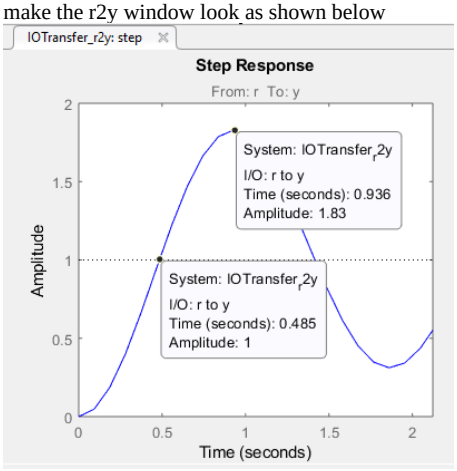
Record the following readings from this CSD windows:

- GM: inf
- PM: 6.78 deg at 3.36 rad/s
- CL poles $-0.2+3.37i$ and $-0.2-3.37i$ shown as the two red dots in the Root Locus Editor. Click on them and read the values at the bottom of the window.
- A lightly damped step response 'r2y: step' (i.e., from input r to output y) with a large overshoot

It is apparent that the response is unsatisfactory because:

- many oscillations until it settles down to $x_{ss}=1$
- large overshoot
- insufficient phase margin

Zoom into the 'r2y: step' window and use data tips to mark the rise time t_r and overshoot M_p and make the r2y window look as shown below



3.6 Verify Design Specifications

It is apparent from the reading taken so far that:

DS1: t_r=0.485 sec < 1.5 sec indicating that DS1 is satisfied
DS2: M_p=83% > 20% indicating that DS2 is NOT satisfied
DS3: GM= inf; PM=6.78 deg indicating that DS3 is NOT satisfied

A controller must be design to improve system performance and satisfy all the design specifications.

4 PID CONTROLLER USING AUTOMATIC PID TUNING TO GENERATE DESIGN1

In this section we explain how a PD controller can be added and tuned to give a better system performance that satisfies all the design specifications.

Press ‘Tuning Methods’ and select ‘PID’ from the pull-down menu. The ‘PID Tuning’ window opens up (Figure 10).

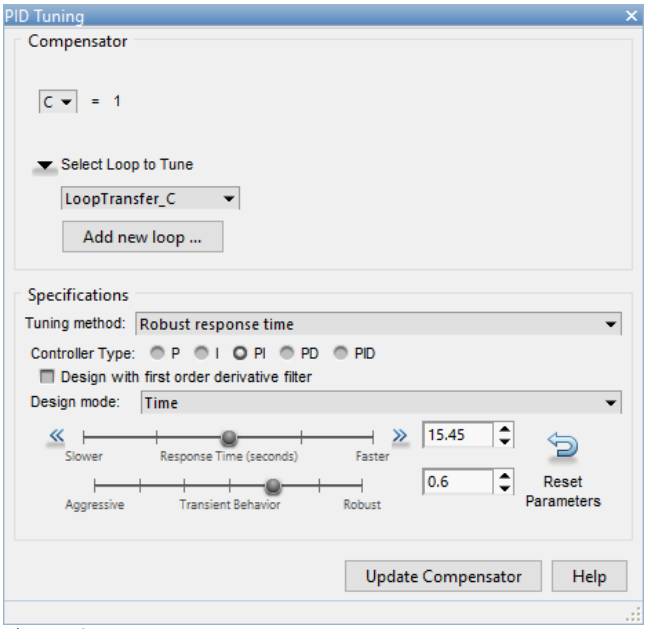


Figure 10

4.1 CHOOSE PD CONTROLLER AND ADJUST ‘RESPONSE TIME’ RANGE

Select ‘PD’ on ‘Controller Type’ line. Press the >> arrow on the RHS of the ‘Response Time’ bar to see ‘1.545’ in the RHS window. Use the up/down scroll (or just type in the value) until it shows the value 1.5 (see Figure 11).

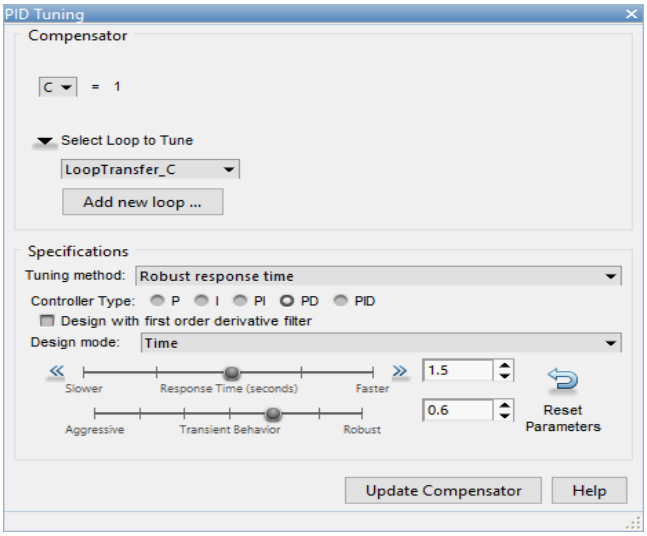


Figure 11

Press ‘Update Compensator’button. The ‘PID Tuning’ window looks as shown in Figure 12.

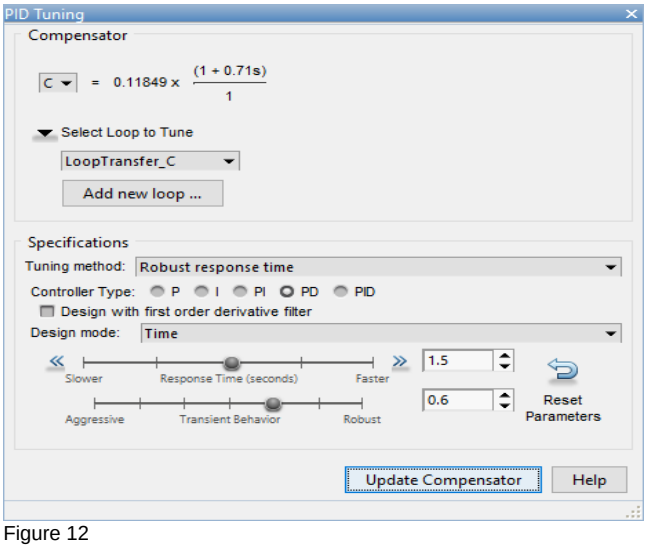


Figure 12

5 EVALUATE DESIGN1 MODEL

Close ‘PID Tuning’ widow and return to the ‘Control System Designer’ main window. Press ‘Store’. A new name appears in the ‘Designs’ LHS window; the new name is ‘Design2’. Restore the original view in the step response window using the zoom -out button. Now, the main window looks as shown in Figure 13.

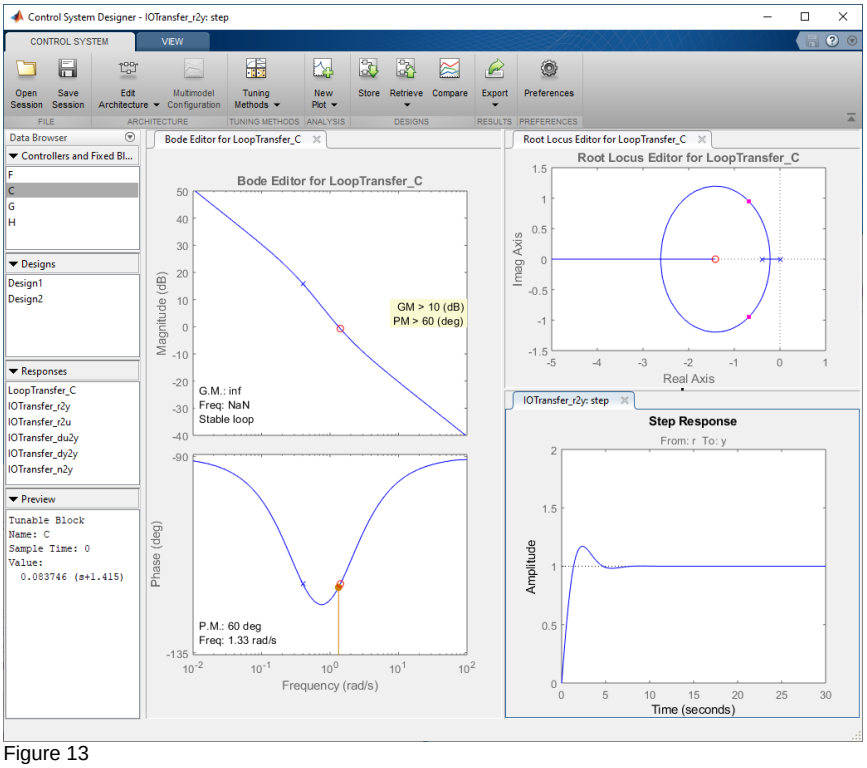


Figure 13

Note that the controller formula appear as

```
Tunable Block
Name: C
Sample Time: 0
Value:
  0.083746 (s+1.415)
```

5.1 Performance Indicators of Design2 Model

Next, verify the status of DS1 and DS2 conditions. Recall

- DS1: Fast response time as measured by rise time
  $t_r \le 1.5$ sec
- DS2: maximum percentage overshoot for step input less than 20%
  $M_p \le 20\%$

Use datatips to read t_r and M_p on the 'Step Response' plot for the closed loop system 'IOTransfer_r2y: step' (Figure 14).

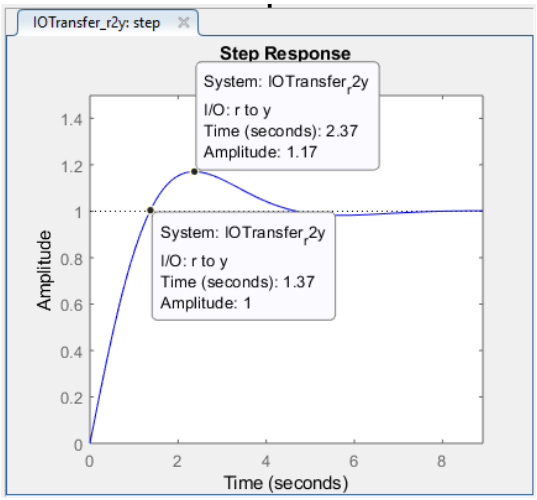


Figure 14

It is apparent that:

- t_r=1.37 sec < 1.5 sec which means that DS1 is satisfied
- M_p=17% < 20% which means that DS2 is satisfied.

5.2 Stability Margins of Design2 Model

The stability margins can be read in the Bode plot of Figure 13; they are much better than in the original design:

- GM: inf which is better than $GM = 10\,\text{dB}$ required by DS3
- PM: 60 deg, which meets the value $PM = 60°$ required by DS3

The stability margin criteria are satisfied which means that DS3 is satisfied.

We state that all three design specification DS1, DS2, DS3, have been met and the control system design process has completed successfully.

6 COMPARE THE STEP RESPONSE OF THE INITIAL AND FINAL DESIGNS

To compare the initial and final designs, do the following:

Press ‘Compare’ button and a ‘Compare Designs’ window pops up (Figure 15).

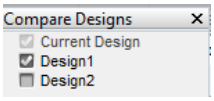


Figure 15

Check box in front of ‘Design1’. The step response plot contains the two responses overlapped (Figure 16).

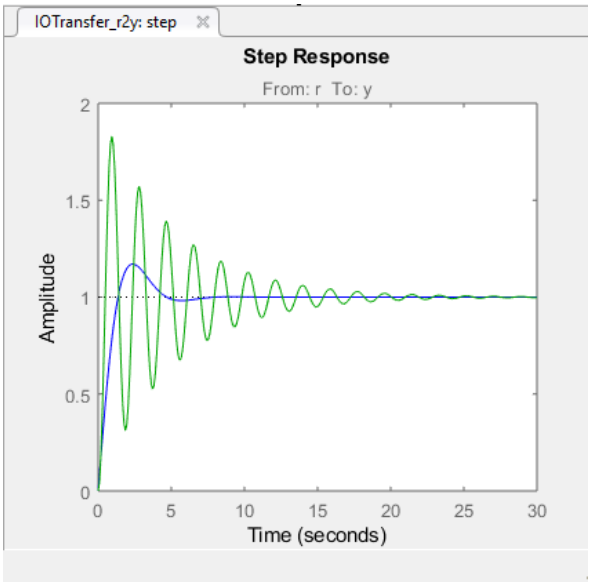


Figure 16

The initial design ‘Design1’ is shown in green, whereas the current design is shown in blue.

The response improvements achieved through this controller design process are quite apparent.

aircraft_roll_model.m

```
%% description
%{
aircraft roll model for use with CSD Tool
%}
%% initialization
clc                     % clear command window
clear                   % clear workspace
% close all               % close all plots
format compact
s=tf('s');
%% aircraft model
K=114;
J=10;   % inertia
c=4;    % damping
G=K/(J*s^2+c*s)     % plant
controlSystemDesigner
```

‘Control System Designer’ MATLAB Tool

Aircraft Roll Motion Autopilot Development
PID Example

Table of Contents:

1 CONTROL SYSTEM DESIGN OBJECTIVES

Consider the aircraft roll transfer function $G(s) = \frac{K}{Js^2 + cs} = \frac{114}{10s^2 + 4s}$

Design a feedback control system to control the aircraft roll motion with the following control design objectives:

Design Objective 1: Control the unconstraint aircraft motion resulting from an aileron input. Have a autopilot system that can maintain the aircraft at a constant bank angle

We will achieve this objective through feedback (FB)

Design Objective 2: Achieve a reasonable aircraft roll response.

Design Objective 3: Ensure safe and stable operation of the feedback control system

We define 'reasonable response' using two control design specifications:

- DS1: Fast response time as measured by rise time
  $t_r \le 1.5$ sec
- DS2: maximum percentage overshoot for step input less than 20%
  $M_p \le 20\%$

We ensue safe and stable operation of the feedback control system by meeting a third design specification based on stability margins:

- DS3: $GM = 10\,\text{dB}$, $PM = 60°$

2 CREATE PLANT TRANSFER FUNCTION G

Run m-file `"aircraft_roll_model.m"` to create the plant transfer function G in the Workspace (Figure 1)

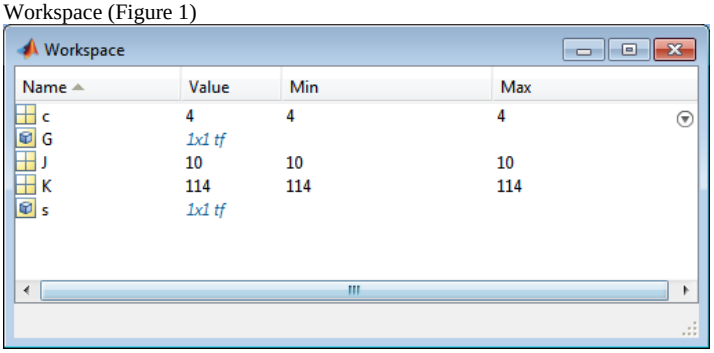


Figure 1

3 CREATE 'CONTROL SYSTEM DESIGNER' MODEL

3.1 OPEN 'CONTROL SYSTEM DESIGNER' SOFTWARE TOOL

In the Command Window, type "`controlSystemDesigner`"
`>>controlSystemDesigner`
(legacy name "`sisotool`", also works, i.e., `>>sisotool`)

The Control System Designer GUI opens as shown in Figure 2.

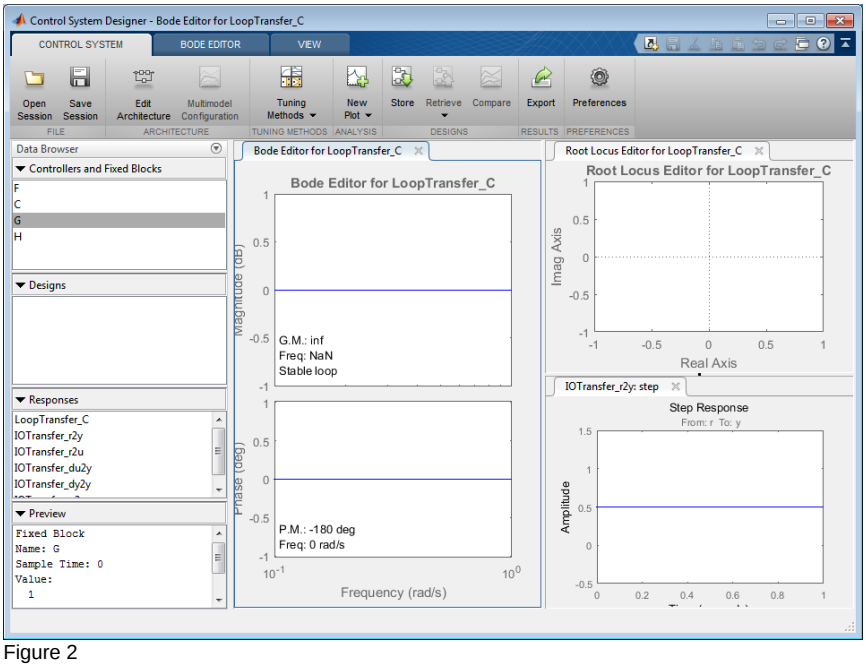


Figure 2

The default configuration has four blocks (F, C, G, H) as indicated in the LHS pane. All the blocks in this configuration are set to be unitary transfer functions by default (the plant block G, which is selected in the upper LHS pane, is shown in the lower LHS pane with Value: 1).

3.2 IMPORT PLANT TRANSFER FUNCTION G

Press “Edit Architecture” button to import the system G data from the Workspace (Figure 3).

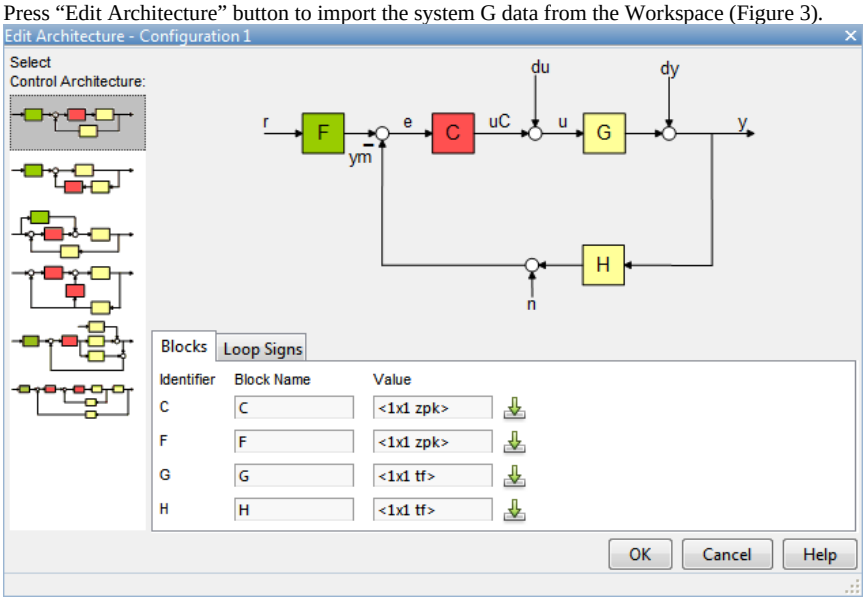


Figure 3

Note that the first control system configuration is selected by default (this is the configuration we work with, no need to change it.)
Press the down arrow icon at the RHS end of the row G to open the “Import Data for G” dialog box and select ‘G’ from the Base Workspace (Figure 4).

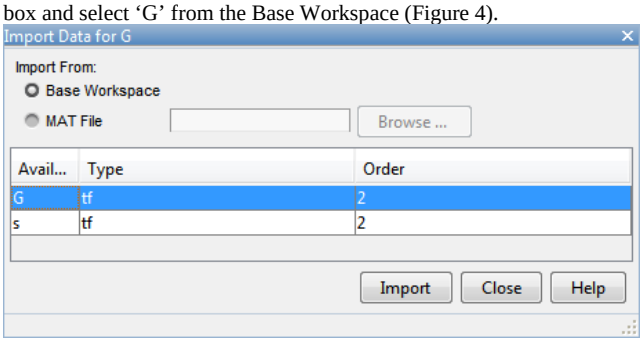


Figure 4

Press ‘Import’ to return to ‘Edit Architecture’ dialog box and press OK to return to the main view. Now, our system $G(s) = \dfrac{114}{10s^2 + 2s}$ is shown in the lower LHS pane of the model (Figure 5)

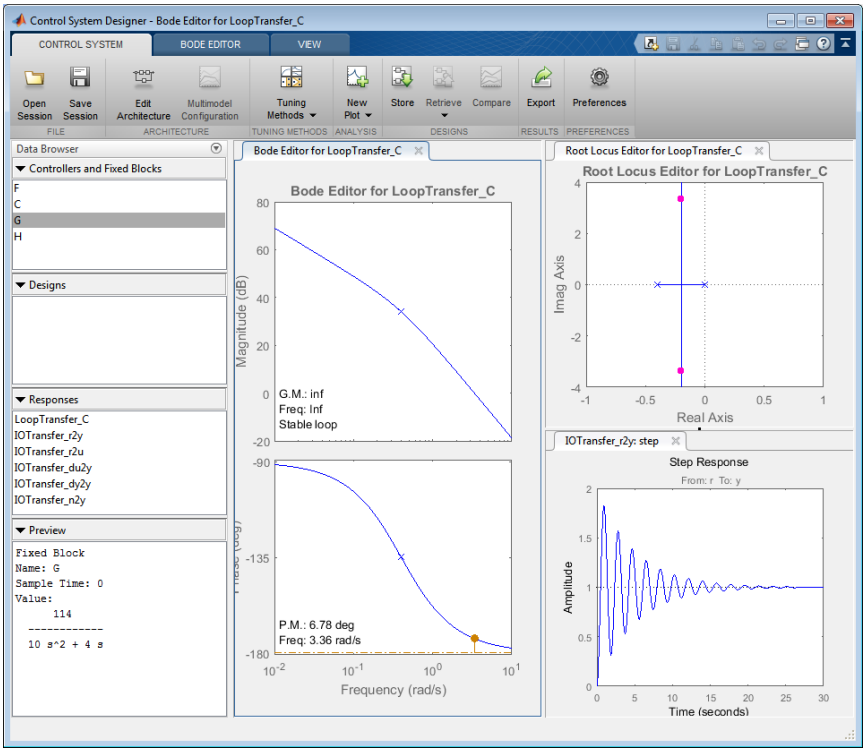


Figure 5

3.3 Store the Baseline Design as G

Press 'Store' button to store this design; it will shown as 'Design1' in the second LHS pane; double click on it to allow name edit and rename it as 'G'. Your screen should look like Figure 6.

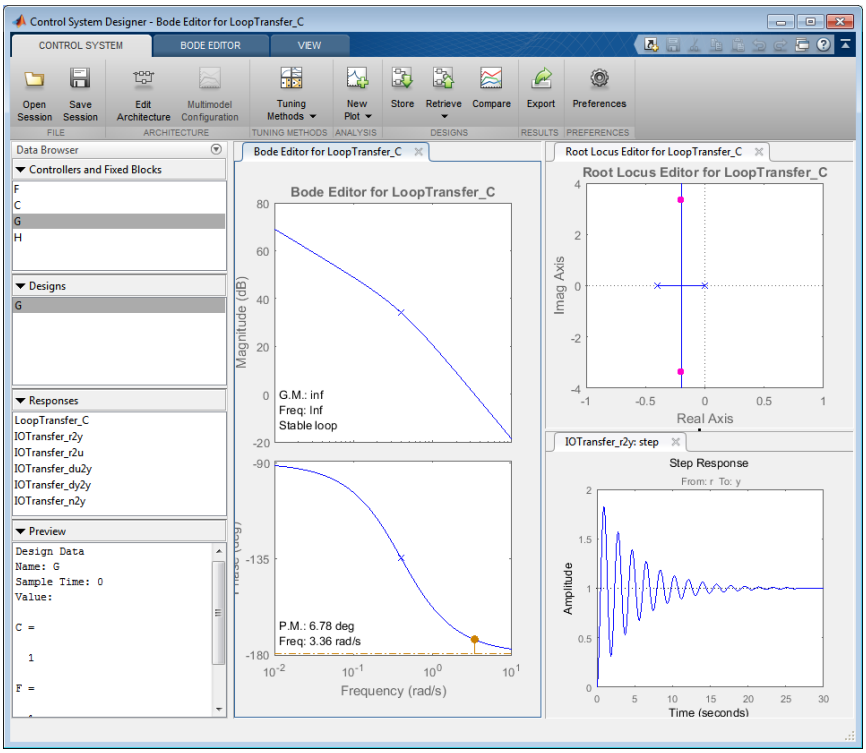


Figure 6

3.4 Save the CSD Sesion

Press 'Save Session' button and save your session with the name 'CSD_aircraft_Session_20161219'. The extension of the file, if shown, should be '.mat'.

3.5 RECORD THE BASELINE PERFORMANCE AND STABILITY READING OF G DESIGN

Record the following readings from this windown:

- GM: inf
- PM: 6.78 deg at 3.36 rad/s
- CL poles $-0.2+3.37i$ and $-0.2-3.37i$ shown as the two red dots in the Root Locus Editor
- A lightly damped step response 'r2y' (i.e., from input r to output y) with a large overshoot

It is apparent that the response is unsatisfactory because:

- many oscillations until it settles down to $x_{ss}=1$
- large overshoot ($x_p \approx 1.8$, $M_p \approx 80\%$)
- insufficient phase margin

4 PID CONTROLLER USING AUTOMATIC PID TUNING TO GENERATE DESIGN1

Press 'Tuning Methods' and select 'PID' from the pull-down menu. The 'PID Tuning' window opens up (Figure 7).

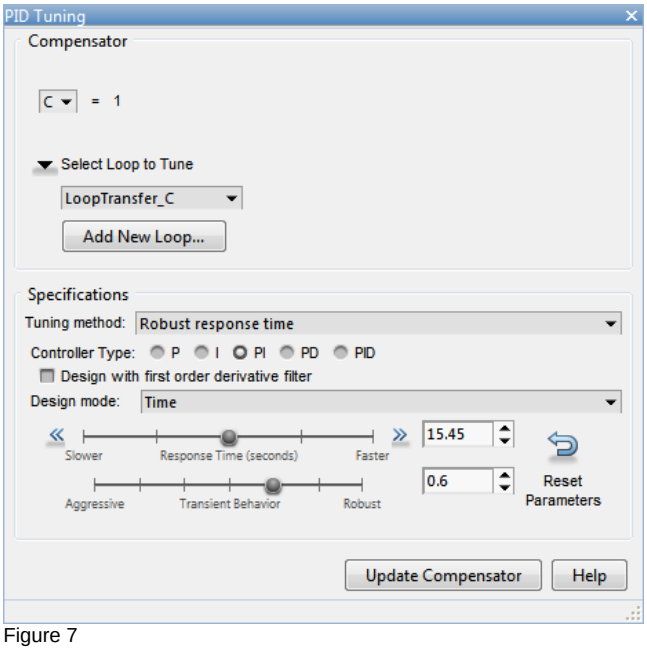


Figure 7

4.1 Adjust 'Response Time' Range

Select 'PID' on 'Controller Type' line. Press the >> arrow on the RHS of the 'Response Time' bar to see '1.545' in the RHS window (Figure 8).

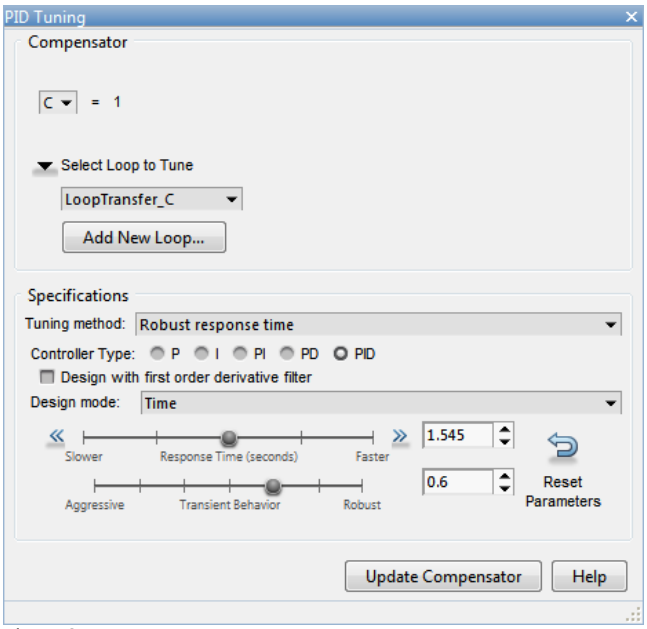


Figure 8

4.2 Adjust the 'Time Response' and 'Transient Behavior'

Use the up/down arrows to make the 'Time Response' reading in the RHS window '1.422' and the 'Transient Behavior' reading in the RHS window be '0.6'.

Press 'Update Compensator'button. The 'PID Tuning' window looks as shown in Figure 9.

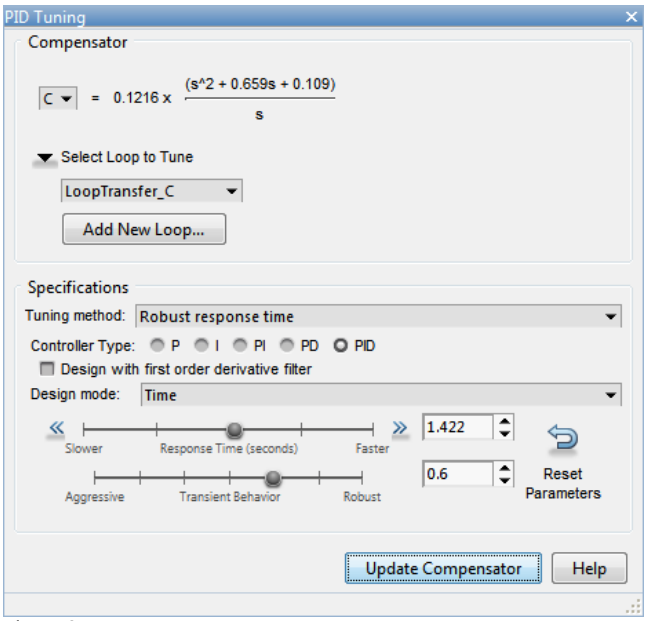


Figure 9

5 EVALUATE DESIGN1 MODEL

Close ‘PID Tuning’ widow and return to the ‘Control System Designer’ main window. Press ‘Store’. A new name appears in the ‘Designs’ LHS window; the new name is ‘Design1’. Now, the main window looks as shown in Figure 10.

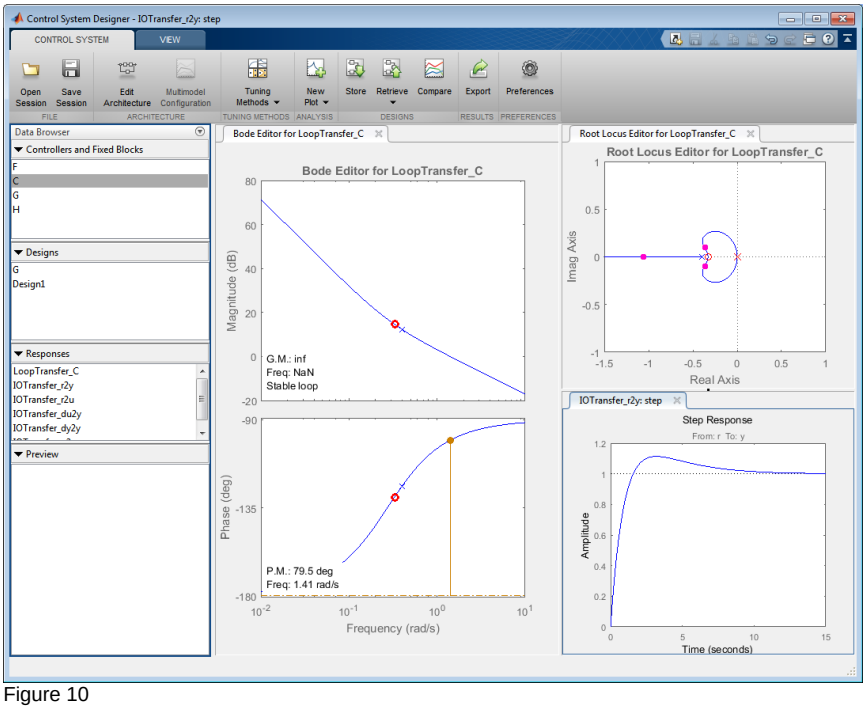


Figure 10

Note that the compensator formula appear as

```
C =

  0.11236 (s+0.3118)^2
  --------------------
            s
```

5.1 STABILITY MARGINS OF DESIGN1 MODEL

The stability margins can be read in the Bode plot; they are much better:

- GM: inf which is better than $GM = 10\,\text{dB}$ required by DS3
- PM: 79.5 deg, which is better tan , $PM = 60°$ required by DS3

The stability margin criteria are satisfied and hence we can say that DS3 condition has been met.

5.2 PERFORMANCE INDICATORS OF DESIGN 1 MODEL

Next, verify the status of DS1 and DS2 conditions. Recall

- DS1: Fast response time as measured by rise time
  $t_r \le 1.5$ sec
- DS2: maximum percentage overshoot for step input less than 20%
  $M_p \le 20\%$

We use datatips to read t_r and M_p on the 'Step Response' plot for the closed loop system 'IOTransfer_r2y' (Figure 11).

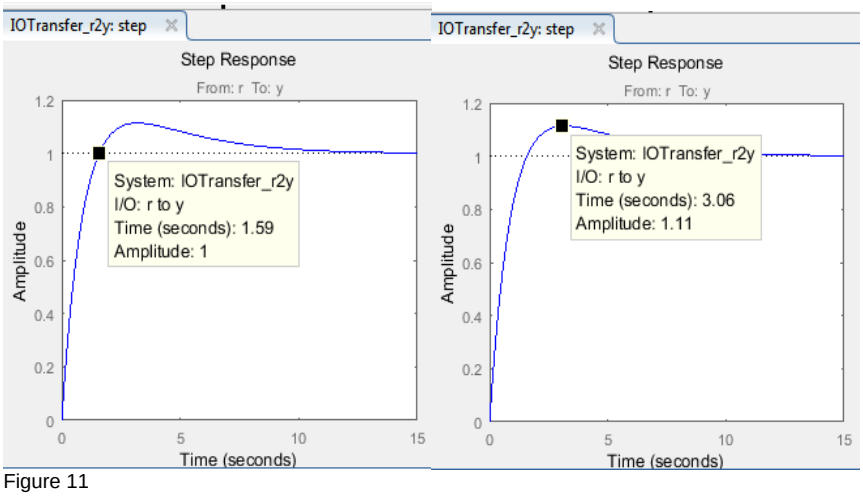


Figure 11

It is apparent that $t_r = 1.59\,\text{sec}$, $M_p = 11\%$

It seems that DS2 has been also satisfied byt DS1 is not yet fully satisfied (though it is very close to it).

6 MANUAL ADJUSTMENT OF THE CONTROLLER – DESIGN2 MODEL

To further improve the system performance and meet the design specifications, we perform manual tuning in the Bode diagram.

6.1 GRAB AND MOVE COMPENSATOR ZERO TO IMPROVE DESIGN

Grab the compensator Zero shown by a red circle on th Bode Phase plot (Figure 12).

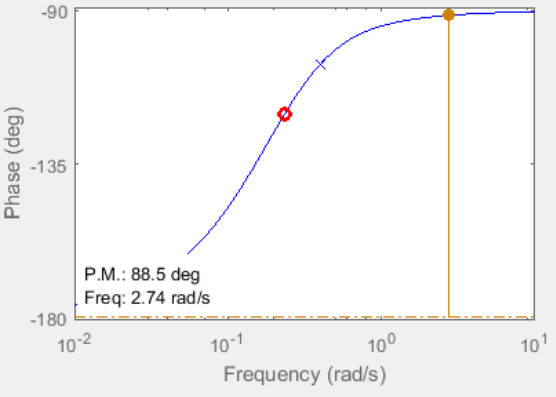


Figure 12

Move this red circle slightly to improve the response time t_r while maintaing GM: inf and a large PM.

This manual adjustment operation must be performed with great delicacy and in very small steps. Continous monitoring of the changes in stability margin should be done continously. If stability is lost, than it should be restored immediately through a backward step and fine adjustment should be continued with delicacy. Note that the gain margin GM is prone of jumping from ‘inf’, which is OK, to a large negative value which is not OK!

When a satisfactory situation has been met, press ‘Store’; a new design named ‘Design2’ appears in the ‘Designs’ LHS pane. The overall picture is shown in Figure 13.

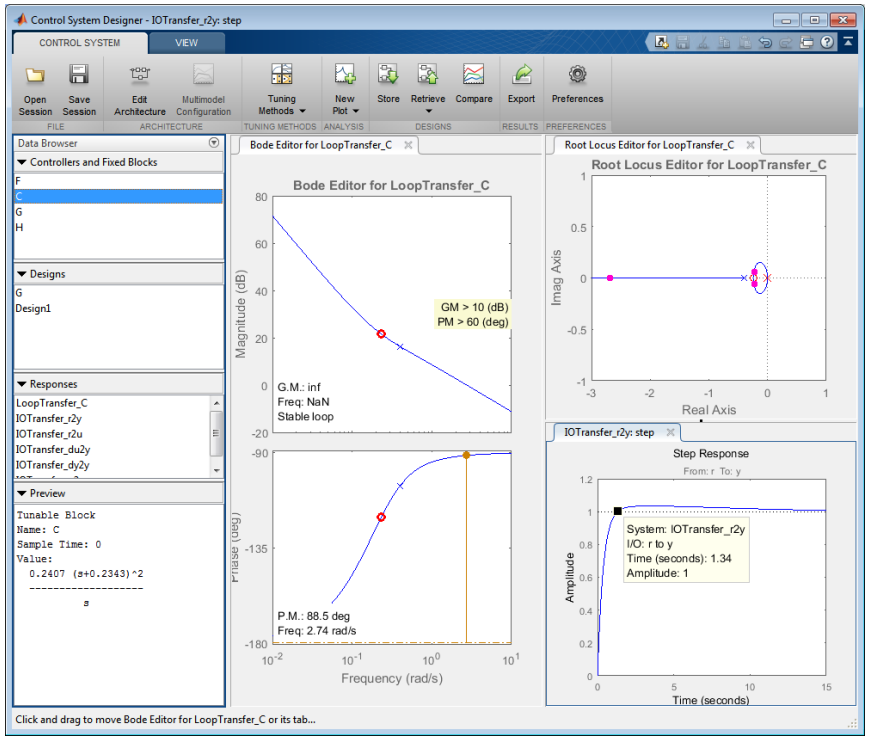


Figure 13

6.2 Adjust Compensator in Compensator Editor

One can also adjust the compensator directly by entering values for its properties. Right click on 'C' in the upper LHS pane and choose 'Open Selection' from the pulldown menu. The 'Compensator Editor' window opens as shown in Figure 14. The compensator properties, e.g., the 'Real Part' of the Complex Zero can be entered here.

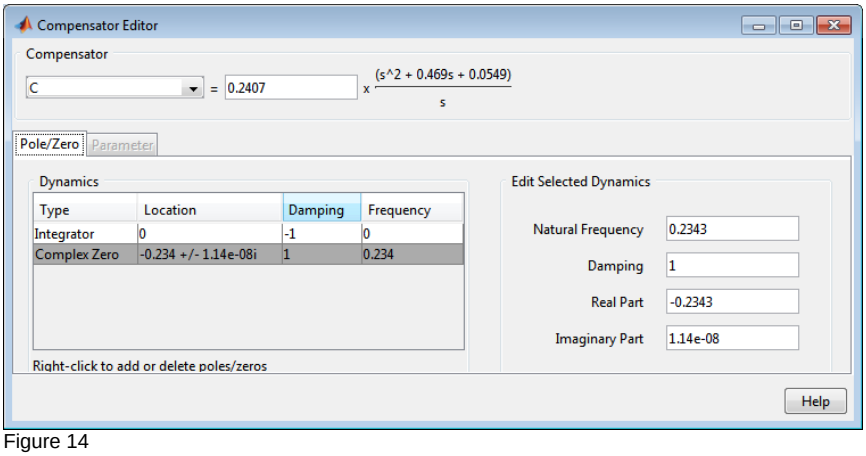


Figure 14

6.3 SAVE THE CSD SESION

To save the latest designs, press the 'Save Session' button and save your session with the name 'CSD_aircraft_Session_20161219'. The file extension, if shown, should be '.mat'.

7 EVALUATE DESIGN2 MODEL

7.1 STABILITY MARGINS OF DESIGN2 MODEL

The stability margins can be read on the Bode plots of Figure 13; they are very good:

- GM: inf which is better than $GM = 10\,\text{dB}$ required by DS3
- PM: 88.5 deg, which is better than $PM = 60°$ required by DS3

The stability margin criteria are satisfied and DS3 condition has been met.

7.2 PERFORMANCE INDICATORS OF DESIGN2 MODEL

Figure 13 indicates that the step response has a faster response time and a smaller overshoot than before (Figure 15).

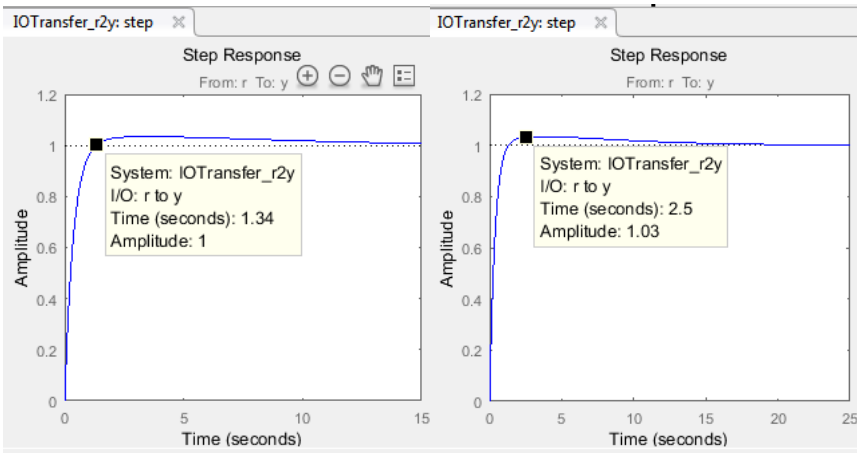


Figure 15

From Figure 15, we read: $t_r = 1.34\,\text{sec}$, $M_p = 3\%$.

It seems that DS1 and DS2 have been met.

The only drawback this this Design2 situation could be that the system seems to take longer than before to settle down to the steady state value.

Further tweaking of the controller could be attempted to overcome this aspect if considered important.

However, for now, we state that all three design specification DS1, DS2, DS3 have been met and the control system design process can be considered complete.

8 COMPARE THE STEP RESPONSE OF THE INITIAL AND FINAL DESIGNS

To compare the initial and final designs, do the following:

Press ‘Compare’ button and a ‘Compare Designs’ window pops up (Figure 16).

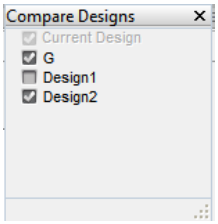


Figure 16

Check boxes in front of ‘G’ and ‘Design2’. The step response plot contains the two responses overlapped (Figure 17).

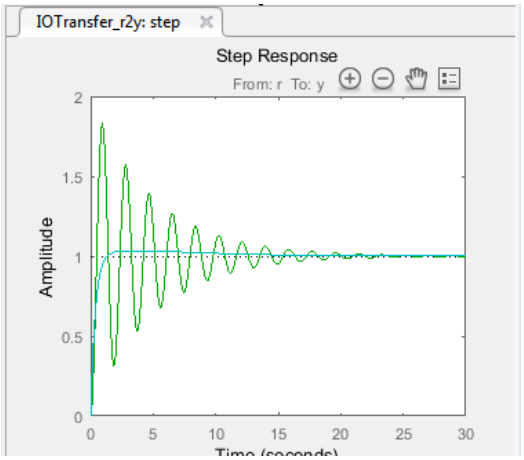


Figure 17

The initial design ‘G’ is shown in green, whereas the final design ‘Design2’ is shown in blue.

The response improvements achieved through this controller design process are quite apparent.

9.3 State-Space Representations

'ss1def

STATE-SPACE REPRESENTATION

State-space representation is a time domain model of the form:

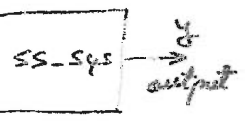
$$\frac{d}{dt}z = Az + Bu \quad \text{(dynamic system)} \quad (1a)$$
$$y = Cz + Du \quad \text{(output)} \quad (1b) \qquad (1)$$

The variables involved in Eq.(1) are:

z = column of state variables, length N_z
u = column of input variables, $\rightarrow N_u$
y = column of output variables $\rightarrow N_y$
A, B, C, D = state space matrices

$$\underset{N_z}{\left[A \right]}^{N_z} \quad \underset{N_z}{\left[B \right]}^{N_u}$$

$$\underset{N_y}{\left[C \right]}^{N_z} \quad \underset{N_y}{\left[D \right]}^{N_u}$$



The SS representation is a time-domain representation, whereas the TF representation was a Laplace-domain representation.

MATLAB accepts both TF and SS representations.

503/523

la SS

1st order system
State-space representation

Recall

$$T\dot{x}(t) + x(t) = f(t) \qquad (1)$$

Rewrite (1) as :

$$\dot{x} = -\frac{1}{T}x + f$$

or

$$\frac{d}{dt}x = -\frac{1}{T}x + \frac{1}{T}f \qquad (2)$$

$\dot{z}$ A z B f

$$\begin{cases} \frac{d}{dt}z = Az + Bu \\ y = Cz + Du \end{cases} \qquad \begin{cases} z = x \\ u = f \\ y = x \\ A = -1/T \\ B = 1/T \\ C = 1 \\ D = 0 \end{cases}$$

504/523

2nd Order System

state-space representation

2 SS | def To derive the S.S representation of a 2nd def system, recall its 2nd order ordinary differential equation in standard form, i.e.,

$$\ddot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = \omega_n^2 f(t) \qquad (2).$$

Write (2) in terms of x and $\dot{x}$ only, i.e.,

$$\frac{d}{dt}\dot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = \omega_n^2 f(t) \qquad (3)$$

Add the identity $\frac{d}{dt}x = \dot{x}$ and write (3) as an extended 1st order system, i.e.,

$$\begin{cases} \frac{d}{dt}x = \dot{x} \\ \frac{d}{dt}\dot{x} = -2\zeta\omega_n \dot{x} - \omega_n^2 x + \omega_n^2 f(t) \end{cases} \qquad (4)$$

Express (4) in matrix form, i.e.,

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} f(t) \qquad (5)$$

(dynamic system)

$$y = [x]$$ (output)

505/523

SS1dof Define :

$$z = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad u = [f(t)] \quad y = [x]$$

$$N_z = 2 \qquad N_u = 1 \qquad N_y = 1$$

$$A = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} \qquad (6)$$

$$C = [1 \quad 0] \qquad D = [0]$$

(6) → (5) :

$$\begin{cases} \frac{d}{dt} z = Az + Bu \\ y = Cz + Du. \end{cases} \qquad (7)$$

Eqs. (6), (7) form the SS representation of the dynamic system (2). The TF representation of the same system is

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

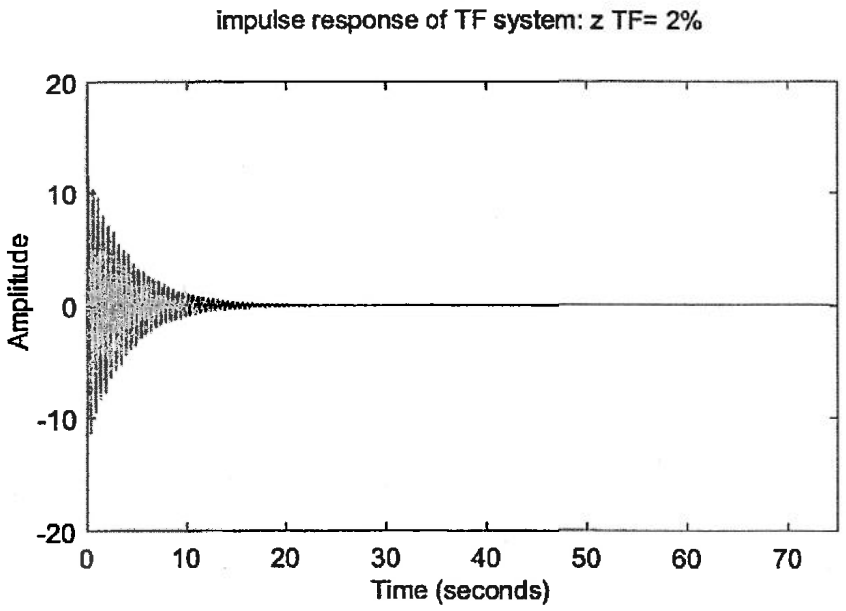
The TF and SS representations are interchangeable in MATLAB

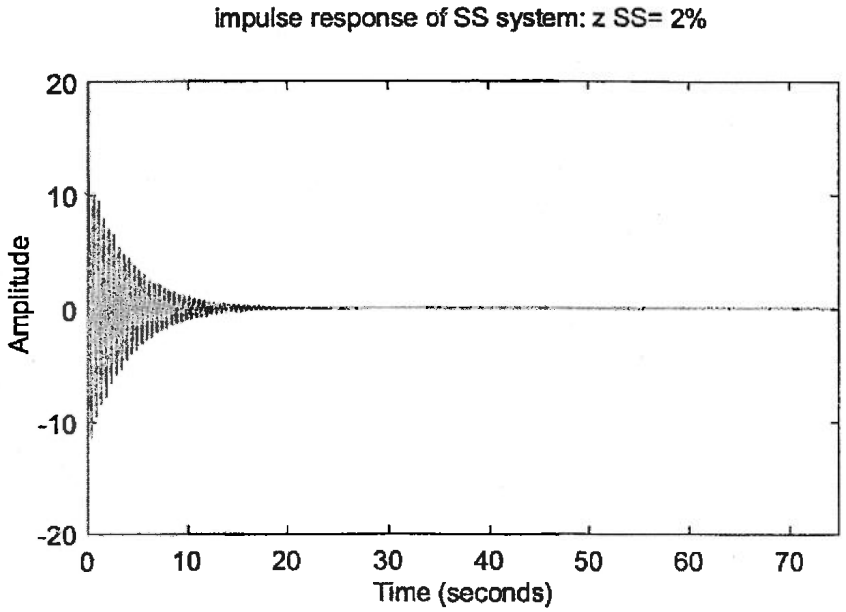
506/523

4
SS idol

```
Comparison of TF and SS representations
initial data: fn, Hz;   z% =
     2     2
TF poles =
  -0.2513 +12.5639i
  -0.2513 -12.5639i
f_TF, Hz;   z_TF% =
    1.9996    2.0000
    1.9996    2.0000
SS poles =
  -0.2513 +12.5639i
  -0.2513 -12.5639i
f_SS, Hz;   z_SS% =
    1.9996    2.0000
    1.9996    2.0000
```

507/523


impulse response of TF system: z TF= 2%
Amplitude
Time (seconds)
20
10
0
-10
-20
0
10
20
30
40
50
60
70


impulse response of SS system: z SS= 2%
Amplitude
Time (seconds)
20
10
0
-10
-20
0
10
20
30
40
50
60
70

508/523

G:\EMCH516\EMCH516_2...\single_dof_time_response_TF_SS.m Page 1

```
%% DESCRIPTION
%{
SISO system time response
Comparison of TF and SS representations
%}
%% initialization
opengl hardwarebasic    % switch to basic hardware graphics functions
clc                     % clear command window
clear                   % clear workspace
% close all               % close all plots
format compact
tol=1e-10;  % tolerance for discarding machine zero
%% DEFINE PARAMETERS
% ----- data for class -----
fn=2; wn=2*pi*fn; % plunge frequency, Hz
z=2e-2;      % plunge damping ratio
%% DEFINE TIME RANGE
T=1/fn; % time scale
% dt=1e-3;
% tmax=50*T;
tmax=150*T;
Nt=1000; dt=tmax/Nt;
t=0:dt:tmax; % time range
%% CALCULATE SISO TRANSFER MATRIX
G=tf([wn^2],[1 2*z*wn wn^2]);
%% CALCULATE POLES OF G
[~,poles_TF,~]=zpkdata(G); s_TF=poles_TF{1,1};     % poles
% display(s_TF,'poles of G')
f_TF=abs(imag(s_TF)/(2*pi));                            % frequencies
zz=-real(s_TF)./abs(s_TF); z_TF=zz.*(abs(zz)>tol); % damping
%% CALCULATE IMPULSE RESPONSE OF TF SYSTEM
figure(1);
subplot(2,1,1); impulse(G,t);
title(['impulse response of TF system: z TF= 'num2str(z_TF(1)*100) '%'])
%% DISPLAY TF RESULTS
display('Comparison of TF and SS representations')
display([fn z*100], 'initial data: fn, Hz;   z%');
display(s_TF, 'TF poles');
display([f_TF z_TF*100], 'f_TF, Hz;   z_TF%');
%% CALCULATE SISO STATE SPACE MODEL
%---------------- state space matrices -----------
A=[ 0          1  ;
    -wn^2   -2*z*wn];
B=[  0 ;
    wn^2];
C=[1 0];
D=[0];
ss_sys=ss(A,B,C,D);
%% EXTRACT POLES, FREQUENCY, DAMPING
[~,zz,poles_SS]=damp(ss_sys);
```

509/523

G:\EMCH516\EMCH516_2...\single_dof_time_response_TF_SS.m Page 2

```
% display(ss,'poles of ss_sys')
s_SS=poles_SS;                          % poles
f_SS=abs(imag(poles_SS))/(2*pi);        % frequencies
z_SS=zz.*(abs(zz)>tol);                 % damping
%% DISPLAY SS RESULTS
% display(' ');
display(s_SS, 'SS poles');
display([f_SS z_SS*100], 'f_SS, Hz;   z_SS%');
%% CALCULATE IMPULSE RESPONSE OF SS SYSTEM
subplot(2,1,2); impulse(ss_sys,t);
title(['impulse response of SS system: z SS= 'num2str(z_SS(1)*100) '%'])
```

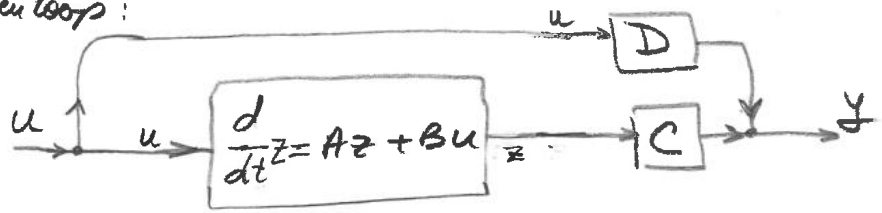
510/523

20181024

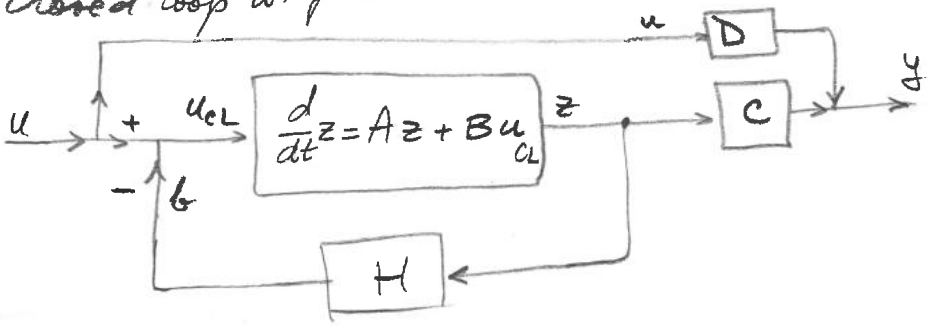
FB of SS system

$$\frac{d}{dt}z = Az + Bu \qquad (1a)$$
$$y = Cz + Du \qquad (1b)$$

open loop:



closed loop w. feedback H:



$$u_{CL} = u - b = u - Hz$$

$$\frac{d}{dt}z = Az + Bu_{CL} = Az + Bu - BHz$$

$$\frac{d}{dt}z = (A - BH)z + Bu$$

$$\boxed{A_{CL} = A - BH}$$

same A_{CL} as for simplified model

$$\begin{cases} \frac{d}{dt}z = A_{CL}z + Bu \\ y = Cz + Du \end{cases}$$

SS sys w. feedback H

511/523

2
20160801

H for velocity feedback

Recall $z = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ (6)

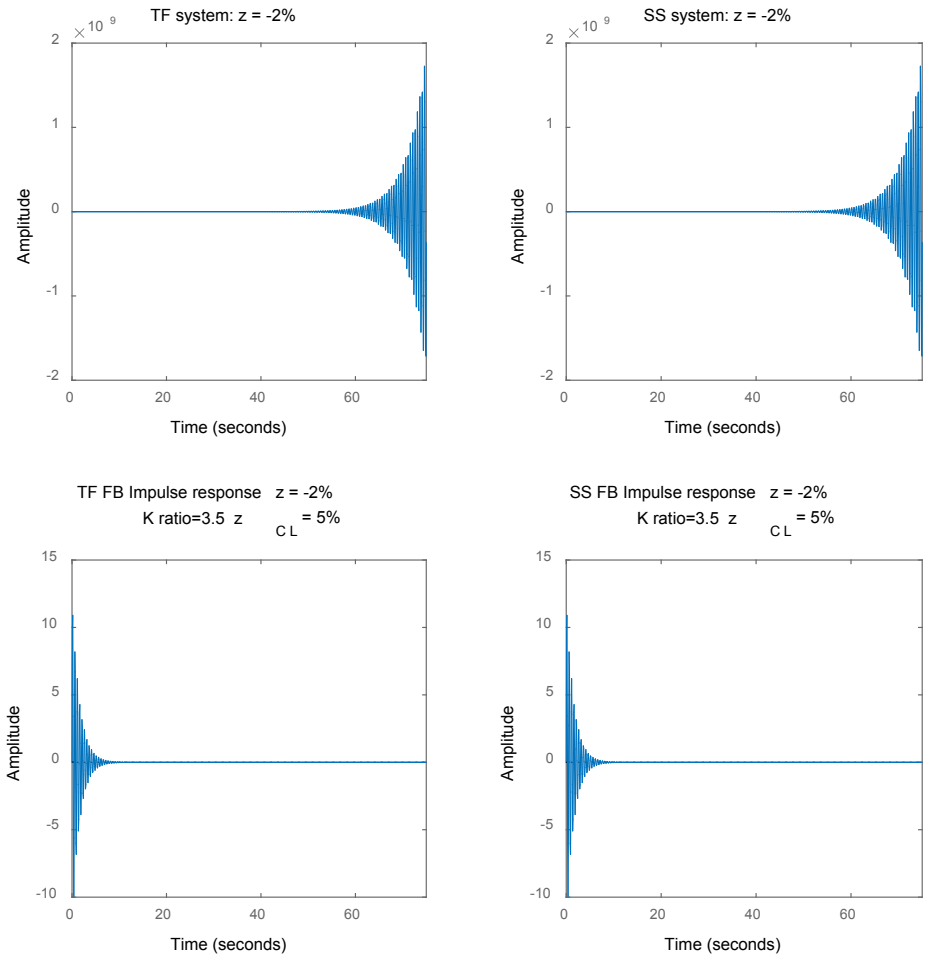
For velocity feedback use

$$H = \begin{bmatrix} 0 & K \end{bmatrix} \quad (7)$$

Then $b(t) = Hz(t) = \begin{bmatrix} 0 & K \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = K\dot{x}(t)$ (8)

feedback signal $b(t)$
is proportional to velocity $\dot{x}(t)$.

512/523


TF system: z = -2%
×10 9
Amplitude
Time (seconds)
SS system: z = -2%
×10 9
Amplitude
Time (seconds)
TF FB Impulse response z = -2%
K ratio=3.5 z = 5%
CL
Amplitude
Time (seconds)
SS FB Impulse response z = -2%
K ratio=3.5 z = 5%
CL
Amplitude
Time (seconds)

513/523

G:\EMCH516\EMCH51...\FB_single_dof_time_response_TF_SS.m Page 1

```
%% DESCRIPTION
%{
SISO system time response with feedback
Comparison of TF and SS representations
use feedback function in TF
use direct definition of A_CL in SS
%}
%% initialization
opengl hardwarebasic    % switch to basic hardware graphics functions
clc                     % clear command window
clear                   % clear workspace
% close all               % close all plots
format compact;
tol=1e-10;  % tolerance for discarding machine zero
%% DEFINE PARAMETERS
% ----- data for example -----
% ----- data for HW -----
m=2;            % mass, kg
fn=2; wn=2*pi*fn; % plunge frequency, Hz
z=-2e-2;      % plunge damping ratio
z_ratio=2.5;        % desired z_ratio defined as zCL/|z|
display('HW06a')
display([fn z*100], 'initial data: fn, Hz;   z%');
%% CALCULATE critical FB gain
K_cr=2*abs(z)/wn; % critical FB gain
%% DEFINE TIME RANGE
T=1/fn; % time scale
% dt=1e-3;
% tmax=50*T;
tmax=150*T;
Nt=1000; dt=tmax/Nt;
t=0:dt:tmax; % time range
%% CALCULATE SISO TRANSFER MATRIX
G=tf(wn^2,[1 2*z*wn wn^2]);% transfer function G
%% EXTRACT TF POLES, FREQUENCY, DAMPING of G
[~,poles_TF,~]=zpkdata(G); s_TF=poles_TF{1,1};     % poles
% display(s_TF,'poles of G')
f_TF=abs(imag(s_TF)/(2*pi));                            % frequencies
zz=-real(s_TF)./abs(s_TF); z_TF=zz.*(abs(zz)>tol); % damping
%% CALCULATE IMPULSE RESPONSE OF TF SYSTEM
figure(1);
subplot(2,2,1); impulse(G,t);
title(['TF system: z = ' num2str(z_TF(1)*100) '%'])
%% CALCULATE SISO STATE SPACE MODEL
%---------------- state space matrices -----------
AA=[ 0          1  ;
    -wn^2   -2*z*wn];
BB=[  0 ;
    wn^2];
CC=[1 0];
```

514/523

G:\EMCH516\EMCH51...\FB_single_dof_time_response_TF_SS.m Page 2

```
DD=[0];
ss_sys=ss(AA,BB,CC,DD);
%% EXTRACT SS POLES, FREQUENCY, DAMPING
[~,zz,poles_SS]=damp(ss_sys);
% display(ss,'poles of ss_sys')
s_SS=poles_SS;                          % poles
f_SS=abs(imag(poles_SS))/(2*pi);        % frequencies
z_SS=zz.*(abs(zz)>tol);                 % damping
%% CALCULATE IMPULSE RESPONSE OF SS SYSTEM
subplot(2,2,2); impulse(ss_sys,t);
title(['SS system: z = ' num2str(z_SS(1)*100) '%'])
%% ADD VELOCITY FEEDBACK TO IMPROVE DAMPING
K_ratio=1+z_ratio;
K=K_ratio*K_cr; % FB gain
display([z_ratio K_ratio K],'z_ratio    K_ratio   K')
H=K*tf([1 0],1);
%% TF SYSTEM WITH FB
G_CL=feedback(G,H);
[~,p_CL_TF,~]=zpkdata(G_CL); s_CL_TF=p_CL_TF{1,1};
f_CL_TF=imag(s_CL_TF)/(2*pi);    % frequencies
zz=-real(s_CL_TF)./abs(s_CL_TF); z_CL_TF=zz.*(abs(zz)>tol);% damping
%% CALCULATE TF IMPULSE RESPONSE WITH FB
subplot(2,2,3); impulse(G_CL,t)
T1=['TF FB Impulse response'];
T2=['   z = ' num2str(z*100) '%'];
T3=['K ratio=' num2str(K_ratio)];
T4=['  z_C_L= ' num2str(z_CL_TF(1)*100) '%'];
line1=[T1 T2];
line2=[T3 T4];
title({line1; line2})
%% SS SYSTEM WITH FB
% ---- SS velocity feedback MATRIX-----
H=[0 K]; % FB matrix
% ================= state space matrices with FB =================
AA_CL=AA-BB*H;  % state matrix AA with FB
% AA_CL=feedback(ss_sys,H); % this does not work; H should be also ss
CC_CL=CC-DD*H;  % state matrix CC with FB
ss_sys_CL=ss(AA_CL, BB, CC_CL, DD);
%% EXTRACT POLES, FREQUENCY, DAMPING
[~,zz_CL_SS,poles_CL_SS]=damp(ss_sys_CL);
%ss_CL=eig(AA0);
% display(ss_CL,'poles of G_CL')
s_CL_SS=poles_CL_SS;                      % poles
f_CL_SS=abs(imag(poles_CL_SS))/(2*pi);    % frequencies
z_CL_SS=zz_CL_SS.*(abs(zz_CL_SS)>tol);       % damping
%% CALCULATE IMPULSE RESPONSE OF SS SYSTEM WITH FB
subplot(2,2,4); impulse(ss_sys_CL,t);
title(['impulse response of SS FB system: zCL SS= 'num2str(z_CL_SS(1)*100) '%'])
T1_SS=['SS FB Impulse response'];
T2_SS=['   z = ' num2str(z*100) '%'];
```

G:\EMCH516\EMCH51...\FB_single_dof_time_response_TF_SS.m Page 3

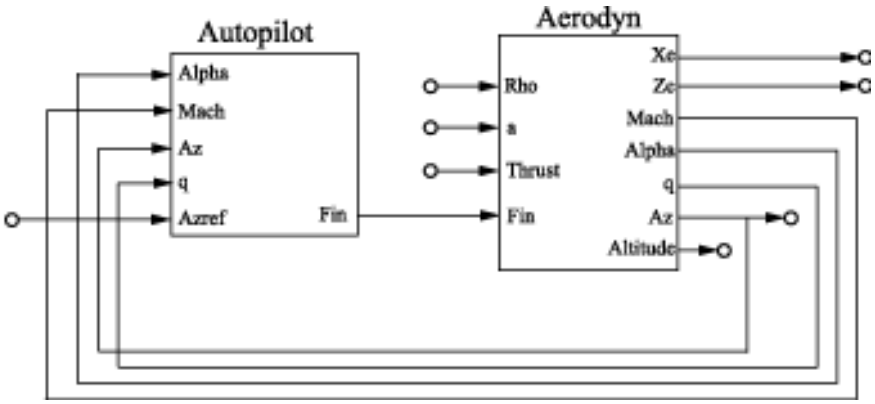
```
101 T3_SS=['K ratio=' num2str(K_ratio)];
102 T4_SS=['  z_C_L= ' num2str(z_CL_TF(1)*100) '%'];
103 line1_SS=[T1_SS T2_SS];
104 line2_SS=[T3_SS T4_SS];
105 title({line1_SS; line2_SS})
106 %% DISPLAY TF RESULTS
107 display(s_TF, 'TF poles');
108 display([f_TF z_TF*100], 'f_TF, Hz;   z_TF%');
109 %% DISPLAY TF FB RESULTS
110 display(s_CL_TF, 'TF FB poles');
111 display([f_CL_TF z_CL_TF*100],'f_CL, Hz;   zh_CL %');
112 %% DISPLAY SS RESULTS
113 % display(' ');
114 display(s_SS, 'SS poles');
115 display([f_SS z_SS*100], 'f_SS, Hz;   z_SS%');
116 %% DISPLAY SS FB RESULTS
117 % display(' ');
118 display(s_CL_SS, 'SS FB poles');
119 display([f_CL_SS z_CL_SS*100],'f_CL_SS, Hz;   z_CL_SS%');
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
```

516/523

MIMO Feedback Loop

This example shows how to obtain the closed-loop response of a MIMO feedback loop in three different ways.

In this example, you obtain the response from `Azref` to `Az` of the MIMO feedback loop of the following block diagram.



You can compute the closed-loop response using one of the following three approaches:

- Name-based interconnection with `connect`
- Name-based interconnection with `feedback`
- Index-based interconnection with `feedback`

You can use whichever of these approaches is most convenient for your application.

Load the plant `Aerodyn` and the controller `Autopilot` into the MATLAB® workspace. These models are stored in the datafile `MIMOfeedback.mat`.

```
load(fullfile(matlabroot,'examples','control','MIMOfeedback.mat'))
```

`Aerodyn` is a 4-input, 7-output state-space (`ss`) model. `Autopilot` is a 5-input, 1-output `ss` model. The inputs and outputs of both models names appear as shown in the block diagram.

Compute the closed-loop response from `Azref` to `Az` using `connect`.

```
T1 = connect(Autopilot,Aerodyn,'Azref','Az');
```

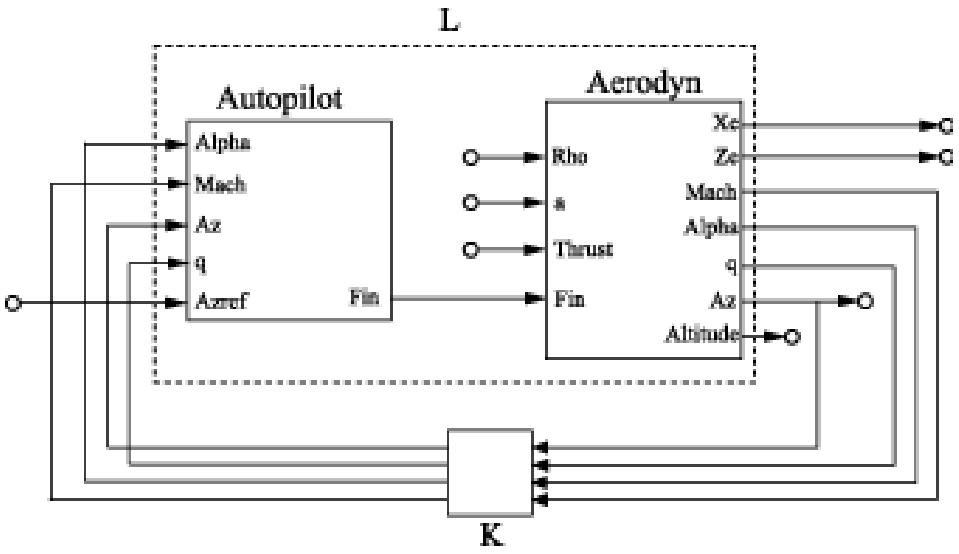
```
Warning: The following block inputs are not used: Rho,a,Thrust.
Warning: The following block outputs are not used: Xe,Ze,Altitude.
```

The `connect` function combines the models by joining the inputs and outputs that have matching names. The last two arguments to `connect` specify the input and output signals of the resulting model. Therefore, `T1` is

a state-space model with input `Azref` and output `Az`. The `connect` function ignores the other inputs and outputs in `Autopilot` and `Aerodyn`.

Compute the closed-loop response from `Azref` to `Az` using name-based interconnection with the `feedback` command. Use the model input and output names to specify the interconnections between `Aerodyn` and `Autopilot`.

When you use the `feedback` function, think of the closed-loop system as a feedback interconnection between an open-loop plant-controller combination `L` and a diagonal unity-gain feedback element `K`. The following block diagram shows this interconnection.




```
L = series(Autopilot,Aerodyn,'Fin');

FeedbackChannels = {'Alpha','Mach','Az','q'};
K = ss(eye(4),'InputName',FeedbackChannels,...
              'OutputName',FeedbackChannels);

T2 = feedback(L,K,'name',+1);
```

The closed-loop model `T2` represents the positive feedback interconnection of `L` and `K`. The `'name'` option causes `feedback` to connect `L` and `K` by matching their input and output names.

`T2` is a 5-input, 7-output state-space model. The closed-loop response from `Azref` to `Az` is `T2('Az','Azref')`.

Compute the closed-loop response from `Azref` to `Az` using `feedback`, using indices to specify the interconnections between `Aerodyn` and `Autopilot`.

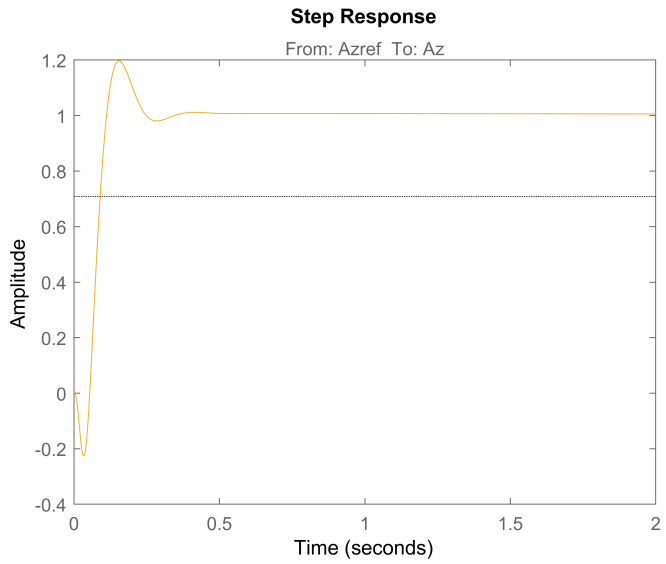
```
L = series(Autopilot,Aerodyn,1,4);
K = ss(eye(4));
T3 = feedback(L,K,[1 2 3 4],[4 3 6 5],+1);
```

The vectors `[1 2 3 4]` and `[4 3 6 5]` specify which inputs and outputs, respectively, complete the feedback interconnection. For example, `feedback` uses output 4 and input 1 of `L` to create the first feedback interconnection. The function uses output 3 and input 2 to create the second interconnection, and so on.

`T3` is a 5-input, 7-output state-space model. The closed-loop response from `Azref` to `Az` is `T3(6,5)`.

Compare the step response from `Azref` to `Az` to confirm that the three approaches yield the same results.

```
step(T1,T2('Az','Azref'),T3(6,5),2)
```



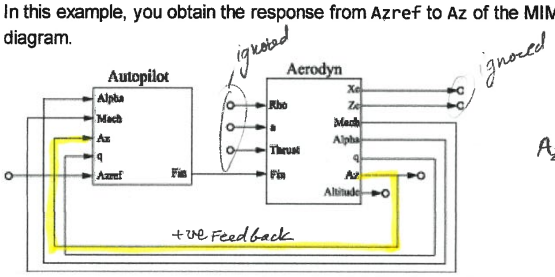
MIMO Feedback Loop - MATLAB & Simulink http://www.mathworks.com/help/control/ug/build-a-model-of-a-multi-i...

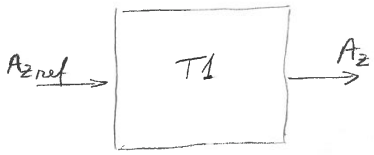
MIMO Feedback Loop

This example shows how to obtain the closed-loop response of a MIMO feedback loop in three different ways.

Open This Example

In this example, you obtain the response from `Azref` to `Az` of the MIMO feedback loop of the following block diagram.





Compute the closed-loop response from `Azref` to `Az` using `connect`.

T1

Input output

```
T1 = connect(Autopilot,Aerodyn,'Azref','Az');
```

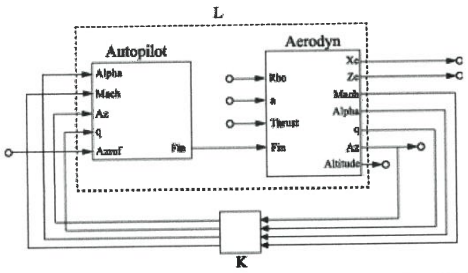
```
Warning: The following block inputs are not used: Rho,a,Thrust.
Warning: The following block outputs are not used: Xe,Ze,Altitude.
```

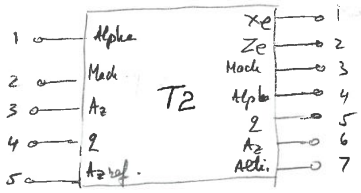
The `connect` function combines the models by joining the inputs and outputs that have matching names. The last two arguments to `connect` specify the input and output signals of the resulting model. Therefore, `T1` is a

520/523

T2 Compute the closed-loop response from `Azref` to `Az` using name-based interconnection with the `feedback` command. Use the model input and output names to specify the interconnections between `Aerodyn` and `Autopilot`.

When you use the `feedback` function, think of the closed-loop system as a feedback interconnection between an open-loop plant-controller combination `L` and a diagonal unity-gain feedback element `K`. The following block diagram shows this interconnection.






```
L = series(Autopilot,Aerodyn,'Fin');

FeedbackChannels = {'Alpha','Mach','Az','q'};
K = ss(eye(4),'InputName',FeedbackChannels,...
               'OutputName',FeedbackChannels);

T2 = feedback(L,K,'name',+1);
```

series construct using variable Fin

unit ss system from Alpha, Mach, Az, q to Alpha, Mach, Az, q

match names

+ve feedback

521/523

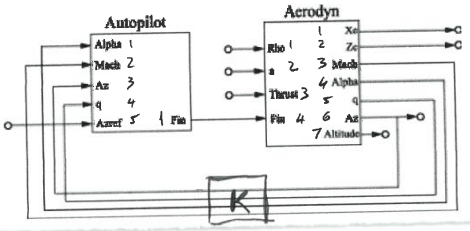
T3

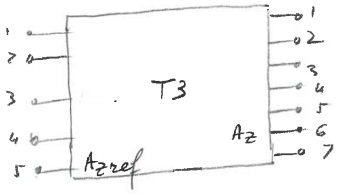
Compute the closed-loop response from Azref to Az using feedback, using indices to specify the interconnections between Aerodyn and Autopilot.

```
L = series(Autopilot,Aerodyn,1,4);
K = ss(eye(4));
T3 = feedback(L,K,[1 2 3 4],[4 3 6 5],+1);
```

The vectors [1 2 3 4] and [4 3 6 5] specify which inputs and outputs, respectively, complete the feedback interconnection. For example, feedback uses output 4 and input 1 of L to create the first feedback interconnection. The function uses output 3 and input 2 to create the second interconnection, and so on.

T3 is a 5-input, 7-output state-space model. The closed-loop response from Azref to Az is T3(6,5).

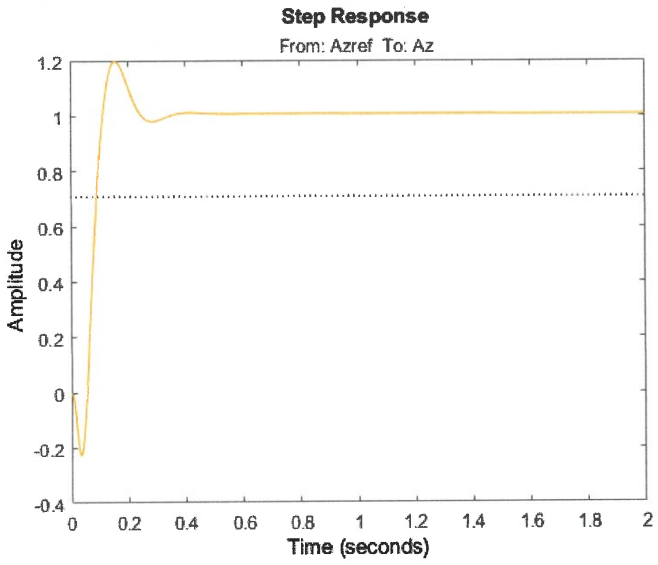




522/523

Compare the step response from `Azref` to `Az` to confirm that the three approaches yield the same results.

```
step(T1,T2('Az','Azref'),T3(6,5),2)
```



523/523