ss1dof

STATE-SPACE REPRESENTATION

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

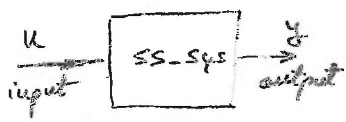
$$\frac{d}{dt} z = A z + B u \quad \text{(dynamic system)} \quad (1a)$$
$$y = C z + D u \quad \text{(output)} \quad (1b) \qquad (1)$$

The variables involved in Eq.(1) are:

z = column of state variables, length N_z
u = column of input variables, — " — N_u
y = column of output variables — " — N_y

A, B, C, D = state space matrices

$$\begin{array}{cc} & N_z \\ N_z & \left[\; A \; \right] \end{array} \qquad \begin{array}{cc} & N_u \\ N_z & \left[\; B \; \right] \end{array}$$



$$\begin{array}{cc} & N_z \\ N_y & \left[\; C \; \right] \end{array} \qquad \begin{array}{cc} & N_u \\ N_y & \left[\; D \; \right] \end{array}$$

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

MATLAB accepts both TF and SS representations.

1a
SS

1^{st} 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

$$\underset{z}{\frac{d}{dt}x} = \underset{A}{-\frac{1}{T}}\,\underset{z}{x} + \underset{B}{\frac{1}{T}}\,\underset{f}{f} \qquad (2)$$

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

$$\left[\begin{array}{l} z = x \\ u = f \\ y = x \\ A = -1/T \\ B = 1/T \\ C = 1 \\ D = 0 \end{array}\right.$$

state-space representation

SS 1dof To derive the SS representation of a 1-dof system, recall its 2^{nd} 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 1^{st} order system, i.e.,

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

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)

ss1dof Define :

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

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

$$\left.\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} \\ C &= [1 \quad 0] \quad D = [0] \end{aligned}\right\} \quad (6)$$

(6) → (5) :

$$\left\{\begin{aligned} \frac{d}{dt} z &= Az + Bu \\ y &= Cz + Du. \end{aligned}\right. \quad (7)$$

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

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

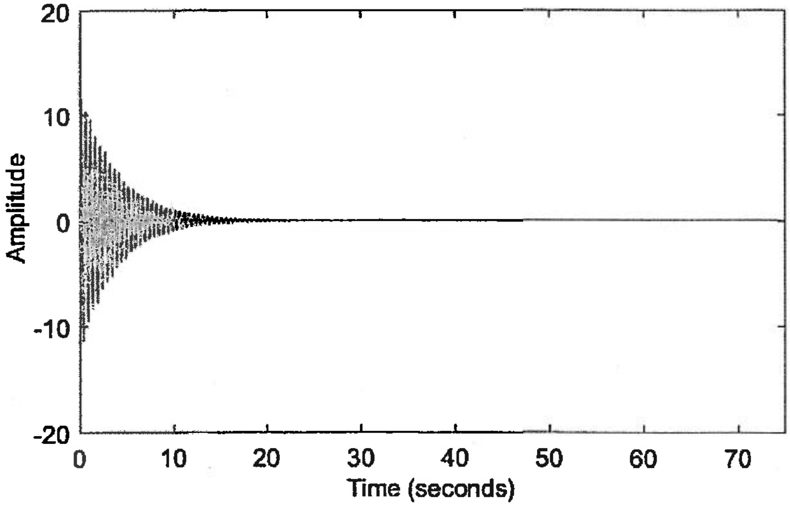
The TF and SS representations are interchangeable in MATLAB

4
ss1dof

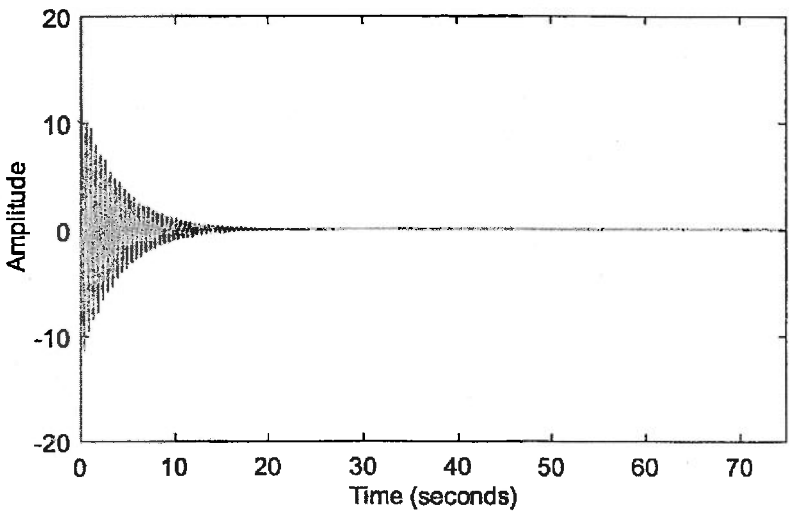
```
Comparison of TF and SS representations
initial data: fn, Hz;    z% =
     2     2
TF poles =
  -0.2513 +12.5639i
  -0.2513 -12.5639i
f_TF, Hz;    z_TF% =
    1.9996    2.0000
    1.9996    2.0000
SS poles =
  -0.2513 +12.5639i
  -0.2513 -12.5639i
f_SS, Hz;    z_SS% =
    1.9996    2.0000
    1.9996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

5
SS 1dof

impulse response of TF system: z TF= 2%



impulse response of SS system: z SS= 2%



```
%% 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);
```

```
% 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) '%'])
```

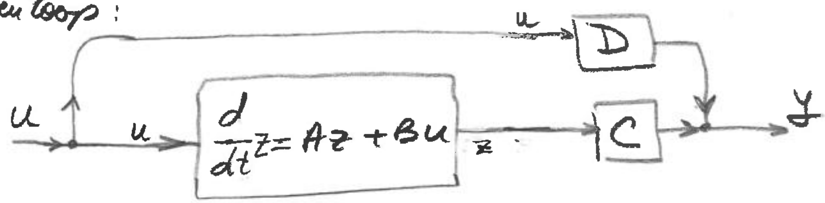
20181024

FB of SS system

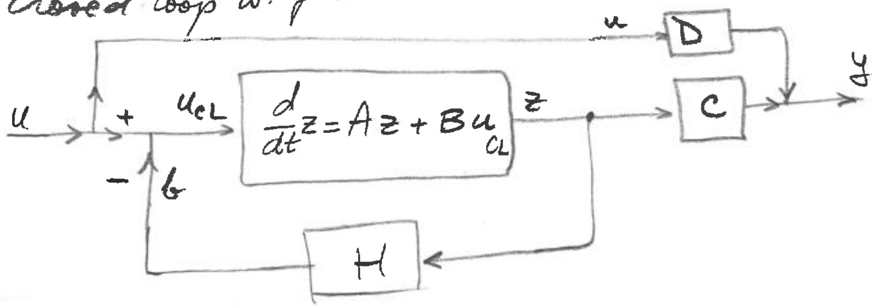
$$\frac{d}{dz} z = Az + Bu \qquad (1a)$$

$$y = Cz + Du \qquad (1b)$$

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

2

20160801

$\underline{H}$ for velocity feedback

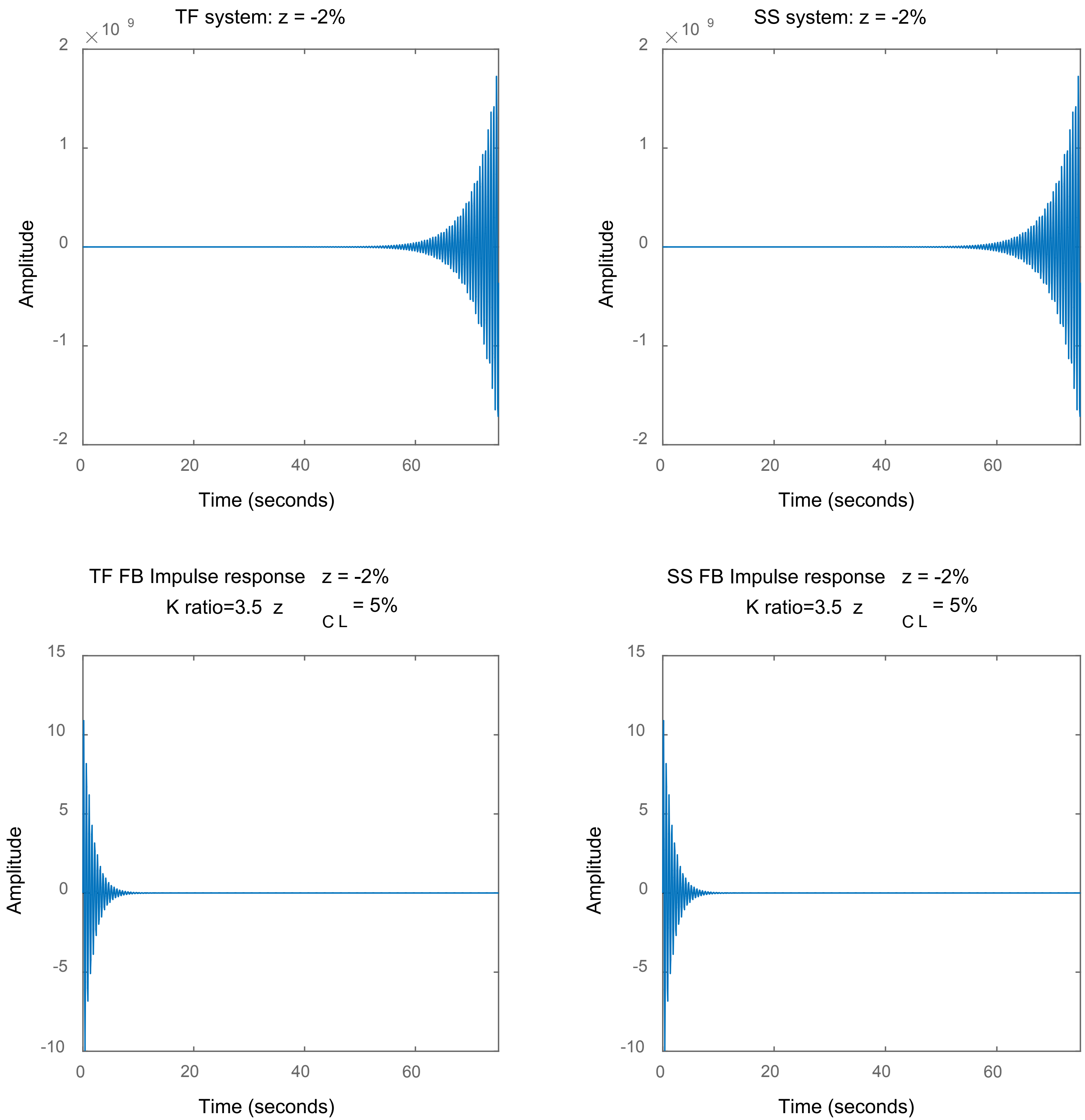
Recall $z = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ (6)

For velocity feedback use

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

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

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


TF system: z = -2%
×10 9
2
1
0
-1
-2
Amplitude
0
20
40
60
Time (seconds)
SS system: z = -2%
×10 9
2
1
0
-1
-2
Amplitude
0
20
40
60
Time (seconds)
TF FB Impulse response z = -2%
K ratio=3.5 z CL = 5%
15
10
5
0
-5
-10
Amplitude
0
20
40
60
Time (seconds)
SS FB Impulse response z = -2%
K ratio=3.5 z CL = 5%
15
10
5
0
-5
-10
Amplitude
0
20
40
60
Time (seconds)

```
%% 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];
```

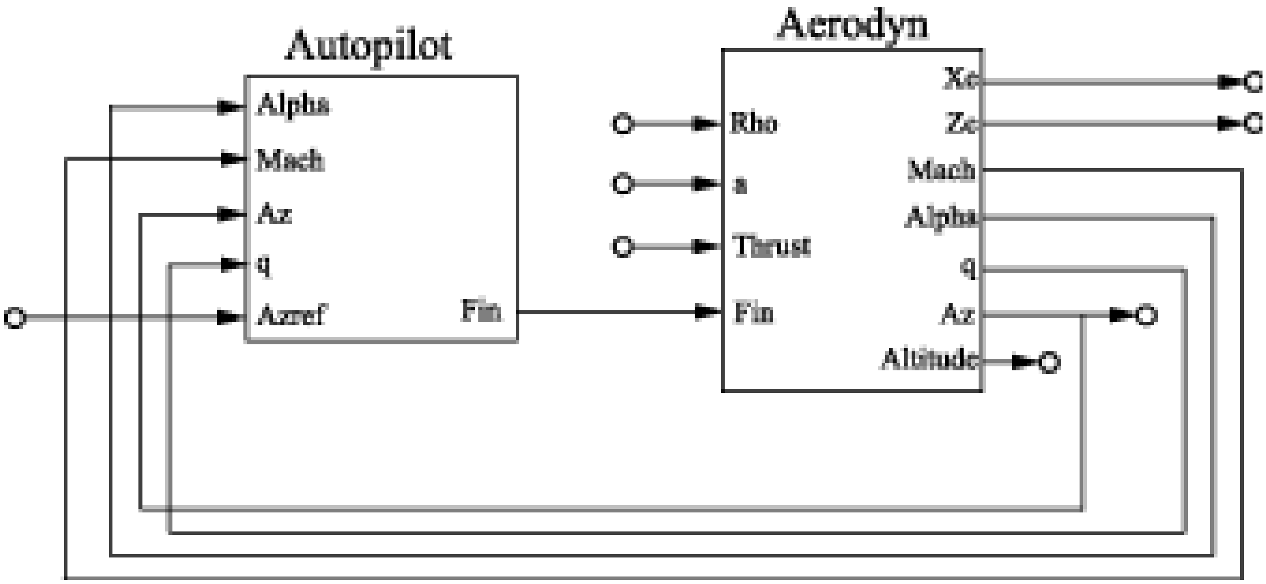
```
DD=[0];
ss_sys=ss(AA,BB,CC,DD);
%% EXTRACT SS POLES, FREQUENCY, DAMPING
[~,zz,poles_SS]=damp(ss_sys);
% display(ss,'poles of ss_sys')
s_SS=poles_SS;                         % poles
f_SS=abs(imag(poles_SS))/(2*pi);       % frequencies
z_SS=zz.*(abs(zz)>tol);                % damping
%% CALCULATE IMPULSE 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) '%'];
```

```
T3_SS=['K ratio=' num2str(K_ratio)];
T4_SS=['  z_C_L= ' num2str(z_CL_TF(1)*100) '%'];
line1_SS=[T1_SS T2_SS];
line2_SS=[T3_SS T4_SS];
title({line1_SS; line2_SS})
%% DISPLAY TF RESULTS
display(s_TF, 'TF poles');
display([f_TF z_TF*100],'f_TF, Hz;   z_TF%');
%% DISPLAY TF FB RESULTS
display(s_CL_TF, 'TF FB poles');
display([f_CL_TF z_CL_TF*100],'f_CL, Hz;   zh_CL %');
%% DISPLAY SS RESULTS
% display(' ');
display(s_SS, 'SS poles');
display([f_SS z_SS*100],'f_SS, Hz;   z_SS%');
%% DISPLAY SS FB RESULTS
% display(' ');
display(s_CL_SS, 'SS FB poles');
display([f_CL_SS z_CL_SS*100],'f_CL_SS, Hz;   z_CL_SS%');
```

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`.

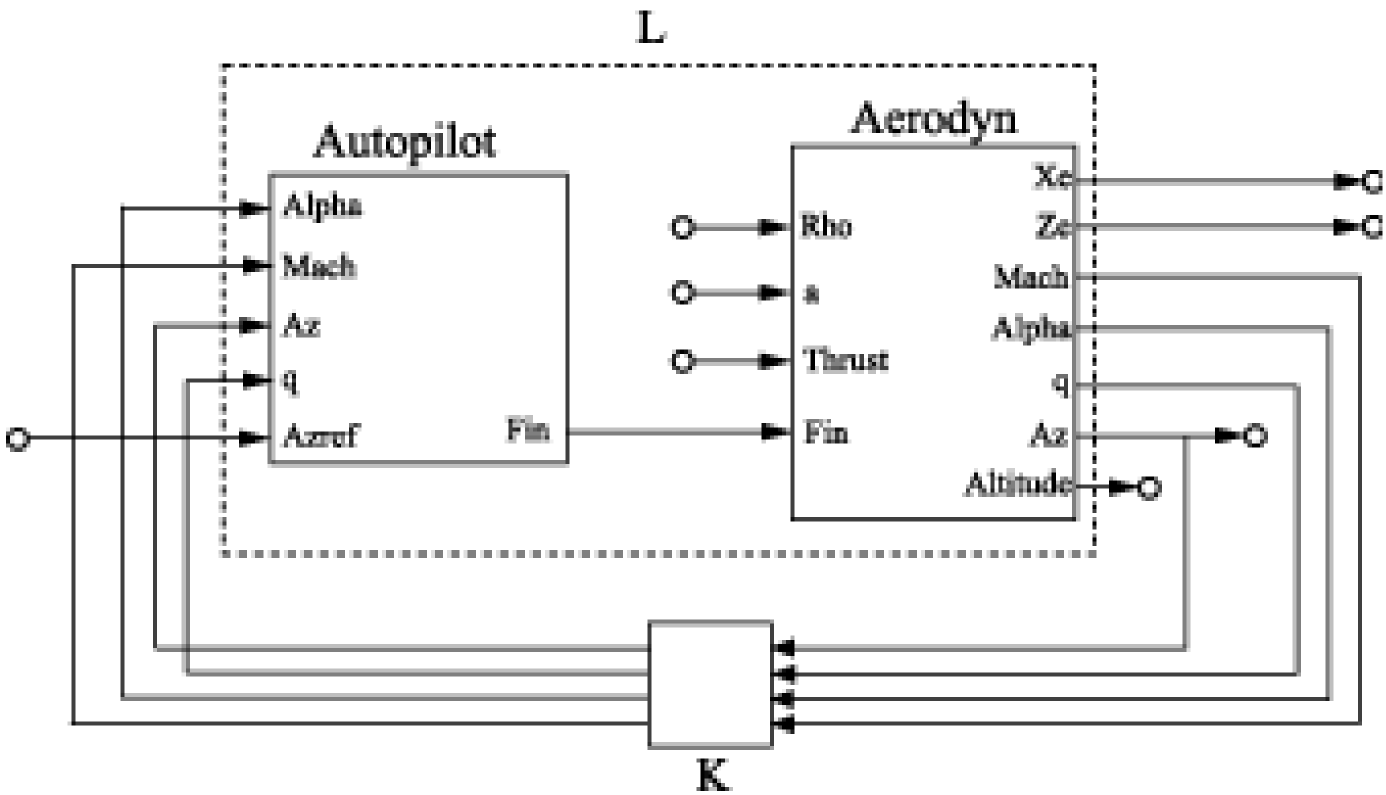
```
T1 = connect(Autopilot,Aerodyn,'Azref','Az');
```

```
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`.

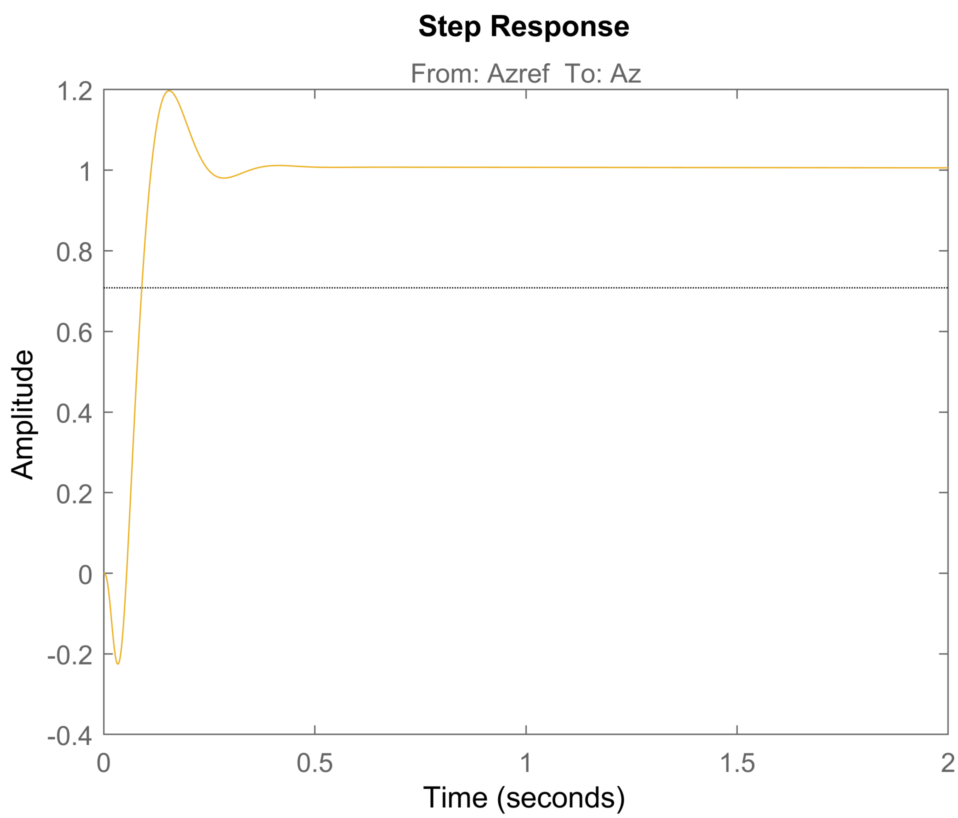
```
L = series(Autopilot,Aerodyn,1,4);
K = ss(eye(4));
T3 = feedback(L,K,[1 2 3 4],[4 3 6 5],+1);
```

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)
```



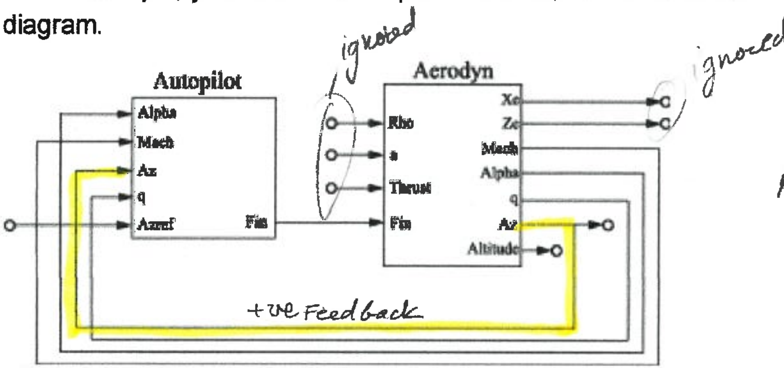
Copyright 2015 The MathWorks, Inc.

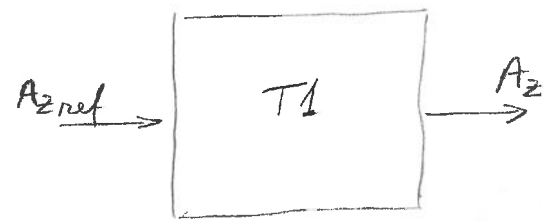
MIMO Feedback Loop

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');
```

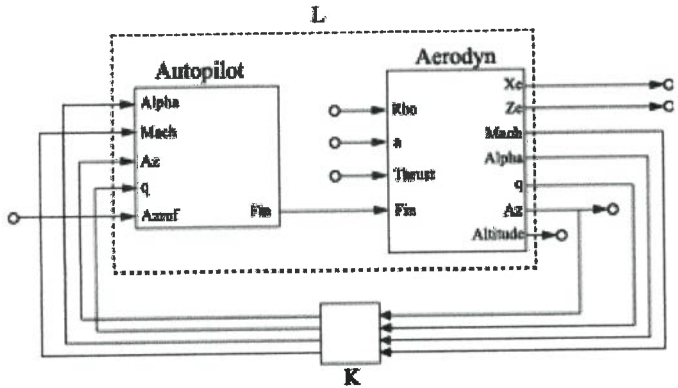
```
Warning: The following block inputs are not used: Rho,a,Thrust.
Warning: The following block outputs are not used: Xe,Ze,Altitude.
```

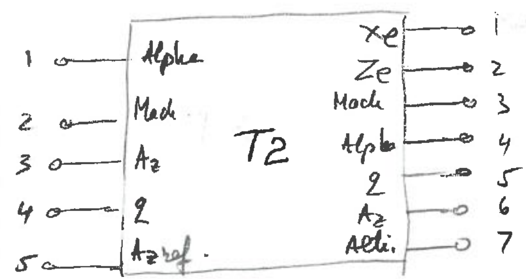
The connect function combines the models by joining the inputs and outputs that have matching names. The last two arguments to `connect` specify the input and output signals of the resulting model. Therefore, `T1` is a

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');
```
series construct using variable Fin

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

match names

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

T3

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

Fin out Fin in

Alpha Mach Az q Alpha Mach Az q

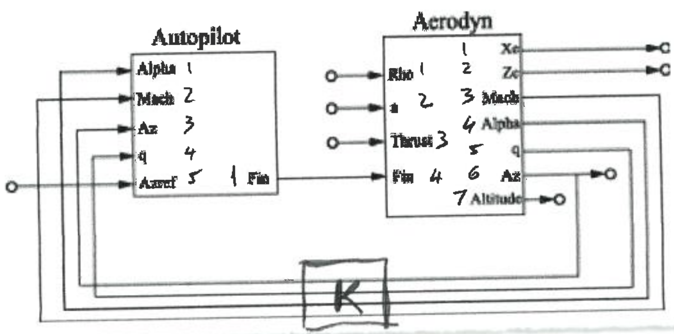
+ve feedback

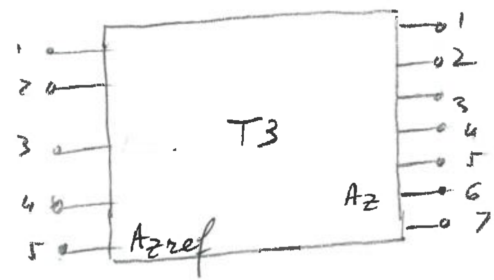
```
L = series(Autopilot,Aerodyn,1,4);
K = ss(eye(4));
T3 = feedback(L,K,[1 2 3 4],[4 3 6 5],+1);
```

feedback (G, H, input, output, ±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)
```