EML 5311

Control System Theory.

HW - 8

Problem - 1

① The feature used to reduce the impact of initial conditions drops some of initial data and only takes into consideration the output at it's steady state value estimating that the system has reached steady state at that time. This helps in improving the accuracy of the estimation. Whereas, ~~sub~~ transient state of the data may ~~skey~~ skew the estimation.

② input_type = 'prbs' has the best identification results and input_type = 'step' has the worst identification results.

(MATLAB code)

Scanned with CamScanner

Problem - 2

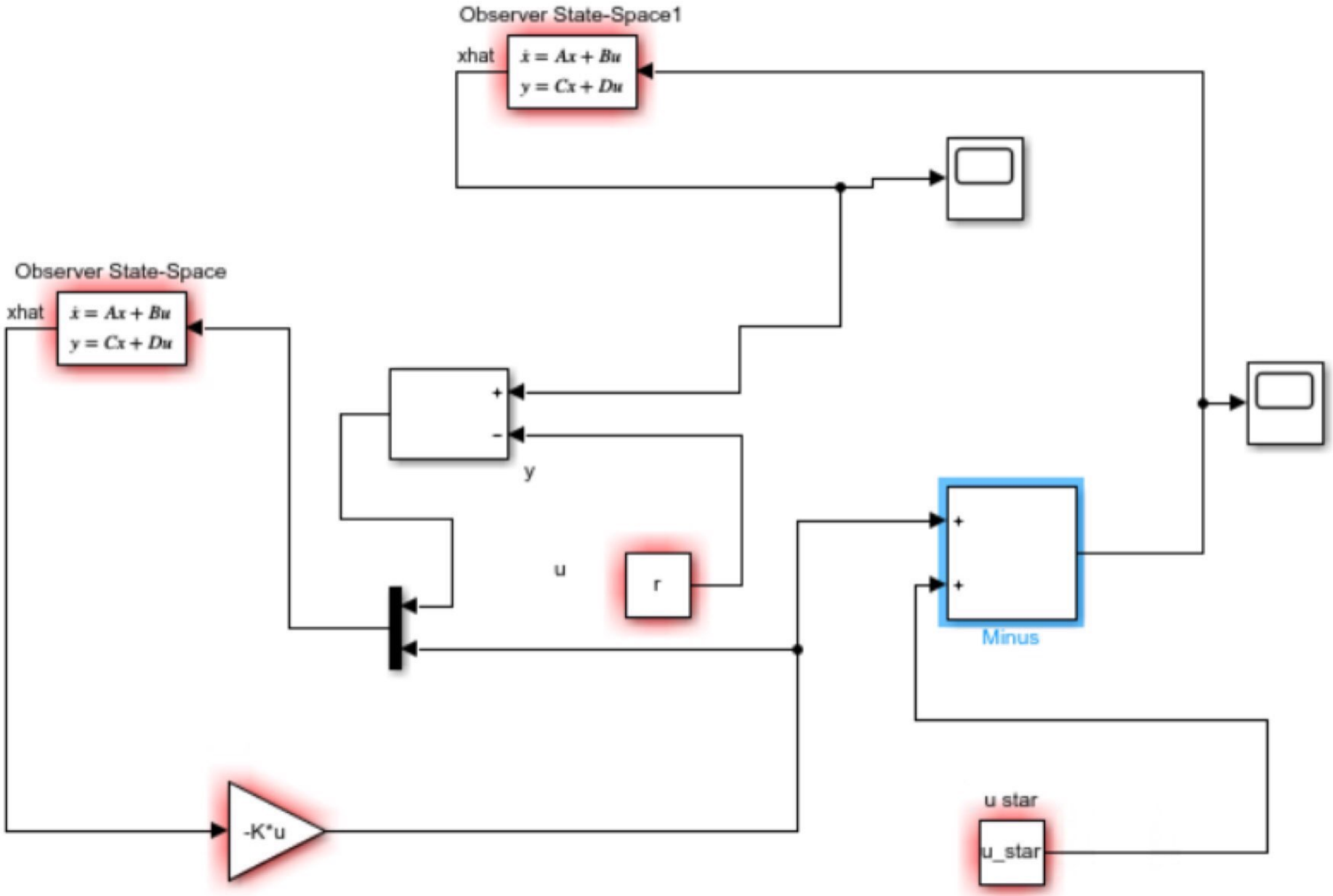
② In order to get a good estimate I chose my ~~fre~~ sampling frequencies such that I could get a smooth curve keeping the samples close to each other while keeping it completely random. I also picked a small sampling period T_s. I set my decay and cycle times so that the system has a faster response and that the frequencies are tested faster. The initial X_0 value was set to around ~~a decade~~ 20 so that it drops the transient state.

③ From the result it is evident that the sinesweep method is cost effective and less time consuming. Since, we were able to model the plant's frequency response by just sampling over few frequencies rather than doing so over infinite or a huge number of frequencies which is time consuming and extrem costly. Therfore, sinesweep method is much more effective in practice. ~~oz~~

Problem - 3

(MATLAB, Simulink).

Scanned with CamScanner



Plant Problem 4 –

① Given

$$m\ddot{d} = mg - \frac{\alpha I^2}{d^2}$$

Let $x_1 = d$

$x_2 = \dot{d}$

$$\Rightarrow \dot{x_1} = x_2$$

$$\dot{x_2} = g - \frac{\alpha I^2}{md^2} = g - \frac{\alpha I^2}{m x_1^2}$$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ g - \frac{\alpha I^2}{md^2} \end{bmatrix} = \begin{bmatrix} x_2 \\ g - \frac{\alpha I^2}{m x_1^2} \end{bmatrix} = f(\bar{x}, \bar{u}).$$

$$y = x_1 = h(\bar{x}, \bar{u})$$

② Finding equilibrium points (x^*, u^*).

$$\Rightarrow 0 = \dot{x_2}^*$$

$$\Rightarrow 0 = g - \frac{\alpha I^2}{m (x_1^*)^2}$$

$$\Rightarrow (x_1^*)^2 = \frac{\alpha I^2}{mg}$$

$$\Rightarrow x_1^* = I\sqrt{\frac{\alpha}{mg}}$$

$$\Rightarrow I = x_1^* \sqrt{\frac{mg}{\alpha}}$$

The equilibrium points are

$$(x^*, u^*) = \left(\begin{pmatrix} I\sqrt{\frac{\alpha}{mg}} \\ 0 \end{pmatrix}, x_1^* \sqrt{\frac{mg}{\alpha}} \right)$$

Scanned with CamScanner

③ Equilibrium points $(\bar{x}_{eq}, \bar{u}_{eq})$ with desired position d_0.

$$\Rightarrow x_1^* = d_0.$$

$$\Rightarrow d_0 = I\sqrt{\frac{\alpha}{mg}} \Rightarrow I = \left(\sqrt{\frac{mg}{\alpha}}\right) d_0.$$

New equilibrium points are.

$$(\bar{x}_{eq}, \bar{u}_{eq}) = \left(\begin{pmatrix} d_0 \\ 0 \end{pmatrix}, d_0\sqrt{\frac{mg}{\alpha}}\right).$$

④. Linearizing around d_0.

$$\tilde{x} = x - \bar{x}_{eq}$$

around the equilibrium point $\left(\begin{pmatrix} d_0 \\ 0 \end{pmatrix}, d_0\sqrt{\frac{mg}{\alpha}}\right)$.

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ \frac{-\alpha I^2}{m}\left(\frac{-2}{x^3}\right) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{2\alpha I^2}{m x_1^3} & 0 \end{bmatrix}_{(\bar{x}_{eq}, \bar{u}_{eq})}.$$

$$A = \begin{bmatrix} 0 & 1 \\ \frac{2\alpha}{m d_0}\left(\frac{mg}{\alpha}\right) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{2g\alpha}{d_0} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial I} \\ \frac{\partial f_2}{\partial I} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{-2\alpha I}{m x_1^2} \end{bmatrix}_{(\bar{x}_{eq}, \bar{u}_{eq})} = \begin{bmatrix} 0 \\ \frac{-2\alpha}{d_0}\sqrt{\frac{g}{m\alpha}} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{-2}{d_0}\sqrt{\frac{\alpha g}{m}} \end{bmatrix}$$

Scanned with CamScanner

$$C = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$D = 0$$

⑤ $$A = \begin{bmatrix} \frac{2g}{d_0} & 0 \\ 0 & 1 \end{bmatrix}$$

Since the Eig(A) = $\frac{2g}{d_0}$ and 1 and they are in SRHP the system is unstable at this equilibrium point.

Problem – 5

Given – $m\ddot{d} = -mg + \frac{\alpha I^2}{d^2}$

① let $x_1 = d$

$x_2 = \dot{d}$

$\dot{x_1} = x_2$

$\dot{x_2} = -g + \frac{\alpha I^2}{m x_1^2}$

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ -g + \frac{\alpha I^2}{m x_1^2} \end{bmatrix}$$

$y = x_1$

② Let $\overline{x_{eq}}$ and $\overline{u_{eq}}$ be the equilibrium points of this system

$\Rightarrow 0 = x_2^*$

$\Rightarrow 0 = -g + \frac{\alpha I^{*2}}{m x_1^* m} \Rightarrow g = \frac{\alpha (I^*)^2}{m (x_1^*)^2}$

Scanned with CamScanner

The equilibrium points are.

$$(\bar{x}_{eq}, \bar{u}_{eq}) = \left(\begin{pmatrix} I^* \sqrt{\frac{\alpha}{mg}} \\ 0 \end{pmatrix}, x_1^* \sqrt{\frac{gm}{\alpha}} \right)$$

③ Equilibrium points corresponding to position d_0.

$$(\bar{x}_{eq}, \bar{u}_{eq}) = \left(\begin{pmatrix} d_0 \\ 0 \end{pmatrix}, d_0 \sqrt{\frac{gm}{\alpha}} \right).$$

④ Linearizing around $(\bar{x}_{eq}, \bar{u}_{eq})$.

$$A = \begin{bmatrix} 0 & 1 \\ \frac{\alpha I^2}{m}\left(\frac{-2}{x_1^3}\right) & 0 \end{bmatrix}_{(\bar{x}_{eq}, \bar{u}_{eq})}.$$

$$A = \begin{bmatrix} 0 & 1 \\ \frac{\alpha (d_0 \sqrt{\frac{mg}{\alpha}})^2 (2) g \alpha}{\alpha \alpha (d_0)^3} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-2g}{d_0} & 0 \end{bmatrix}.$$

$$B = \begin{bmatrix} 0 \\ \frac{2\alpha I}{m x_1^2} \end{bmatrix}_{(\bar{x}_{eq}, \bar{u}_{eq})} = \begin{bmatrix} 0 \\ \frac{2\alpha d_0 \sqrt{\frac{mg}{\alpha}}}{m d_0^2} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{2}{d_0}\sqrt{\frac{\alpha g}{m}} \end{bmatrix}$$

$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$ $\quad D = 0$. (It is unstable).

⑤ $A = \begin{bmatrix} \frac{-2g}{d_0} & 0 \\ 0 & 1 \end{bmatrix}$ $\quad (\because R_1 \Leftrightarrow R_2)$

Eig $(A) = \frac{-2g}{d_0}$ and 1 (Therefore, unstable since the Eig(A) are in SRHP).

Scanned with CamScanner

Problem – 6

Given – $V_s(t) = LI^{\cdot} + IR$.

$$m\ddot{d} = -mg + \frac{\alpha I^2}{d^2} - \beta \dot{d}$$

① Let $x_1 = d$

$x_2 = \dot{d}$

$x_3 = I$

Now,

$V_s = L\dot{x_3} + x_3 R$.

$m\dot{x_2} = -mg + \frac{\alpha x_3^2}{x_1^2} - \beta x_2$.

$\Rightarrow \dot{x_1} = x_2$

$\dot{x_2} = -g + \frac{\alpha x_3^2}{x_1^2 m} - \frac{\beta}{m} x_2$.

$\dot{x_3} = \frac{V_s}{L} - \frac{x_3 R}{L}$.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} x_2 \\ -g + \frac{\alpha x_3^2}{m x_1^2} - \frac{\beta x_2}{m} \\ \frac{V_s}{L} - \frac{x_3 R}{L} \end{bmatrix} = f(\bar{x}, \bar{u}).$$

$y = x_1 = h(\bar{x}, \bar{u})$.

② Equilibrium points corresponding to do ue $(\bar{x}_{eq}, \bar{u}_{eq})$.

$\Rightarrow 0 = x_2^*$

$\Rightarrow 0 = -g + \frac{\alpha x_3^{*2}}{m x_1^{*2}} - \frac{\beta x_2^*}{m}$.

$\frac{\beta x_2^*}{m} = -g + \frac{\alpha (x_3^*)^2}{m (x_1^*)^2} \Rightarrow \frac{\alpha (x_3^*)^2}{m (x_1^*)^2} = g$

Scanned with CamScanner

$$0 = \frac{V_s^*}{L} - \frac{X_3^* R}{L} \qquad (\because x_1^* = d_0).$$

$$V_s^* = X_3^* R.$$

$$\left(\begin{pmatrix} d_0 \\ 0 \\ d_0\sqrt{\frac{mg}{2}} \end{pmatrix}, d_0\sqrt{\frac{mg}{\alpha}}(R) \right) = (\bar{x}_{eq}, \bar{u}_{eq}).$$

(3) Linearizing to find equilibrium point. to get an LTI system.

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-2q}{m} & \frac{-\beta}{m} & \frac{2\alpha}{d_0}\sqrt{\frac{q}{m\alpha}} \\ 0 & 0 & -R/L \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1/L \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$D = 0$$

$$A = \begin{bmatrix} \frac{-2q}{m} & -\beta/m & \frac{2\alpha}{d_0}\sqrt{\frac{q}{m\alpha}} \\ 0 & 1 & 0 \\ 0 & 0 & -R/L \end{bmatrix}$$

Eig (A) = $\frac{-2q}{m}$, 1 and $-R/L$.

Since $X_2 = 1$ lies in the SRHP this equilibrium point is unstable.

Scanned with CamScanner

Problem - 7

① Yes, it is possible to implement the controller C(s) to control this ~~nonlinear~~ plant. Because irrespective of the values of these ~~variables and~~ constants. The closed loop should be able to come to the equilibrium point by computing ~~it~~ error and refining it.

② & ③ (MATLAB & SIMULINK)

Scanned with CamScanner

Table of Contents

an example showing how to construct input, run the

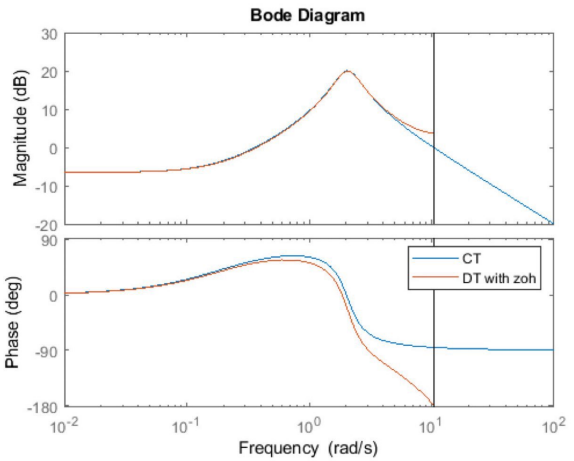
simulink block 'plantForExperiment.slx' and extract the output.

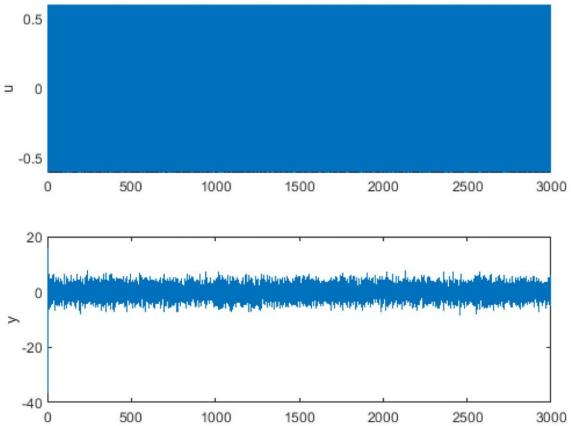
```
clear all


Ts  = 0.3;
Fs = 1/Ts;
% The sampling period (Ts) must be defined for the
% simulink model to run
% warning: do not make Ts larger than 0.01.

%constructing the plant
numPlant_CT=[10 2];
denPlant_CT=[1 1 4.25];
plant_CT_TF=tf(numPlant_CT,denPlant_CT);
[plantFreqResp_CT,freqs_forPlant_CT_rad]=freqs(numPlant_CT,numPlant_CT);
%Beware: the plant is a discrete time transfer function
plant_DT_TF = c2d(plant_CT_TF,Ts,'zoh');
[numPlant_DT,denPlant_DT] = tfdata(plant_DT_TF,'v');

%1 : compute the true freq response, show Bode plot,
numFreqValues = 100;
[plantFreqResp,freqValuesHz_plant] =
 freqz(numPlant_DT,denPlant_DT,numFreqValues,Fs);
freqs_forPlant_rad = freqValuesHz_plant*2*pi;

figure
bode(plant_CT_TF,plant_DT_TF)
legend('CT','DT with zoh')

% [A,B,C,D] = tf2ss(numPlant_DT,denPlant_DT);
% plant_DT_SS = ss(A,B,C,D,Ts);
% state_dim  =length(denPlant_DT)-1;
% x0 = randn(state_dim,1);
%------


%construct input signal

N = 10000;
```

```
T = [0:1:N-1]'*Ts;
%input_type = 'GaussianWhite';
input_type = 'prbs';
%input_type = 'step';
% input_type = 'sweptsine';
noise_stdv = 0.1;

uLevel = 0.6; %max value of input allowed
switch input_type
    case 'prbs'
        u = idinput(length(T),'prbs',[0 1],[-uLevel, uLevel]);
    case 'GaussianWhite'
        u = randn(length(T),1)*sqrt(uLevel);
 case 'sweptsine'
        f0 = 0.002;
        u = uLevel*sin(2*pi*f0*T.*T);
    case 'step'
        u = -uLevel*ones(size(T));
        ind = floor(length(T)/5);
        u(ind:end) = uLevel;

    otherwise
        error('bad')
end
%clip:
u(find(u>uLevel))=uLevel;
u(find(u<-uLevel))=-uLevel;

%------

% u_struct =  [];
% t = [0:1:10000]'*Ts;
% u = sin(2*pi*0.01*t.^2);
u_struct.time = T;
u_struct.signals.values = u;
u_struct.signals.dimensions = 1;
t_final=T(end);

simOut=sim('plantForExperiment')

%extract y
y_struct = simOut.y_struct;
tout = y_struct.time;
y = y_struct.signals.values;

%plot
figure
subplot(211)
plot(T,u);ylabel('u');
subplot(212);
plot(T,y);ylabel('y');
```

Warning: The PRBS signal delivered is the 10000 first values of a full sequence

```
of length 16383.

simOut =

  Simulink.SimulationOutput:
                    tout: [10002x1 double]
                y_struct: [1x1 struct]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]
```

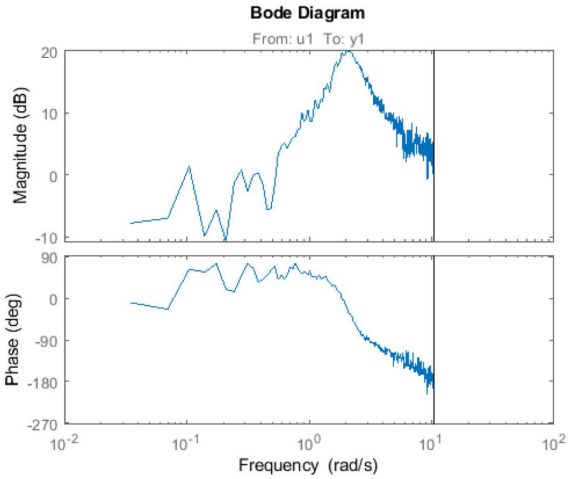




Step 2: estimate G(jw) = Y(jw)/U(jw) from u(t),y(t),

by using the etfe command Step 2.1: put the data in the right format that etfe wants: idDataForThePlant = iddata(y,u,Ts); Phat_DT = etfe(idDataForThePlant);

```
figure
Q=6000;
numFreqRes=300;
uetfe=u(end-Q+1:end);
yetfe=y(end-Q+1:end);
idDataForThePlant = iddata(yetfe,uetfe,Ts);
smoothingWindowSize = Q/10;
Phat_DT = etfe(idDataForThePlant,smoothingWindowSize,numFreqRes);
bode(Phat_DT)
```



compare with the true plant's frequency response.

Beware: the plant and the estimated plant are both discrete time transfer functions, so the frequency axis only goes up to the Nyquist frequency, which is half of the sampling frequency.

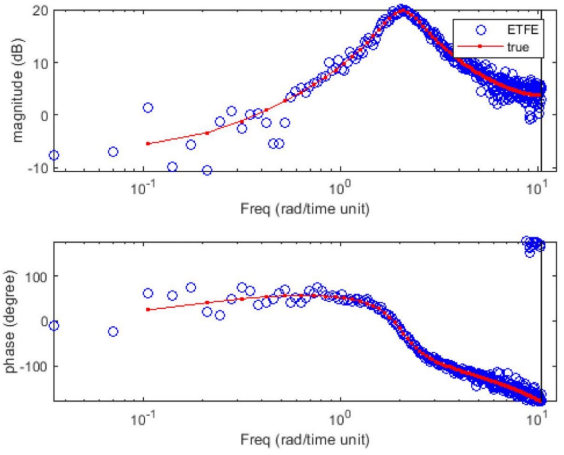
```
plantEstFreqResp=Phat_DT.ResponseData;
plantEstFreqResp = plantEstFreqResp(:);
if ~strcmp(Phat_DT.FrequencyUnit,'rad/TimeUnit')
    freqs_forETFE_rad = 2*pi*Phat_DT.Frequency;
else
    freqs_forETFE_rad = Phat_DT.Frequency;
end
```

compare the true with estimated

```
figure
gainph=[];
phaph=[];

subplot(211);%mag plot
%true plant:
```

```
gainph(1)  =
 semilogx(freqs_forETFE_rad,20*log10(abs(plantEstFreqResp)),'bo');
 hold on
gainph(2)  =
 semilogx(freqs_forPlant_rad,20*log10(abs(plantFreqResp)),'r.-');
% gainph(3)  =
 semilogx(freqs_forPlant_CT_rad,20*log10(abs(plantFreqResp_CT)),'r-');
% show the freq = pi location
plot(Fs*[pi, pi],20*log10([min(abs(plantFreqResp))
 max(abs(plantFreqResp))]),'k-');
axis tight
yy = ylim;
plot(Fs*[pi, pi],yy,'k-');
xlim([0 1.2*Fs*pi]);
xlabel('Freq (rad/time unit)');
ylabel('magnitude (dB)');
legend(gainph,'ETFE','true','true(CT)');

subplot(212); %phase plot
phaph(1)  = semilogx(freqs_forETFE_rad,angle(plantEstFreqResp)*180/
pi,'bo'); hold on
phaph(2)  = semilogx(freqs_forPlant_rad,angle(plantFreqResp)*180/
pi,'r.-');
% phaph(3)  =
 semilogx(freqs_forPlant_CT_rad,angle(plantFreqResp_CT)*180/pi,'r-');
% show the freq = pi location
axis tight
yy = ylim;
plot(Fs*[pi, pi],yy,'k-');
xlim([0 1.2*Fs*pi]);
xlabel('Freq (rad/time unit)');
ylabel('phase (degree)');
```

Warning: Ignoring extra legend entries.



Published with MATLAB® R2020a

Table of Contents

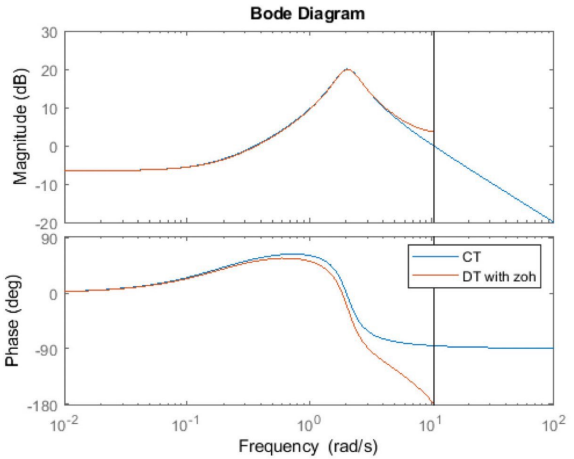
an example showing how to construct input, run the

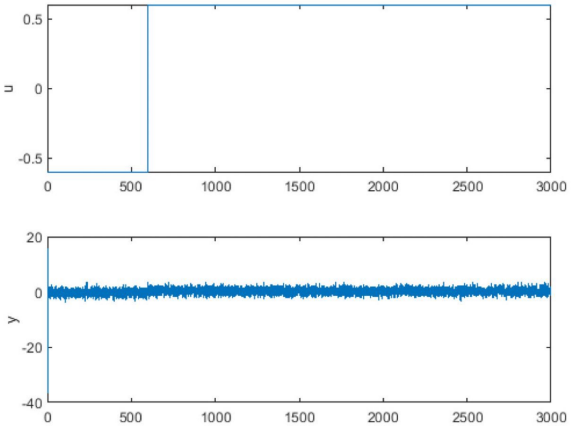
simulink block 'plantForExperiment.slx' and extract the output.

```
clear all


Ts  = 0.3;
Fs = 1/Ts;
% The sampling period (Ts) must be defined for the
% simulink model to run
% warning: do not make Ts larger than 0.01.

%constructing the plant
numPlant_CT=[10 2];
denPlant_CT=[1 1 4.25];
plant_CT_TF=tf(numPlant_CT,denPlant_CT);
[plantFreqResp_CT,freqs_forPlant_CT_rad]=freqs(numPlant_CT,numPlant_CT);
%Beware: the plant is a discrete time transfer function
plant_DT_TF = c2d(plant_CT_TF,Ts,'zoh');
[numPlant_DT,denPlant_DT] = tfdata(plant_DT_TF,'v');

%1 : compute the true freq response, show Bode plot,
numFreqValues = 100;
[plantFreqResp,freqValuesHz_plant] =
 freqz(numPlant_DT,denPlant_DT,numFreqValues,Fs);
freqs_forPlant_rad = freqValuesHz_plant*2*pi;

figure
bode(plant_CT_TF,plant_DT_TF)
legend('CT','DT with zoh')

% [A,B,C,D] = tf2ss(numPlant_DT,denPlant_DT);
% plant_DT_SS = ss(A,B,C,D,Ts);
% state_dim  =length(denPlant_DT)-1;
% x0 = randn(state_dim,1);
%------


%construct input signal

N = 10000;
```

```
T = [0:1:N-1]'*Ts;
%input_type = 'GaussianWhite';
%input_type = 'prbs';
input_type = 'step';
% input_type = 'sweptsine';
noise_stdv = 0.1;

uLevel = 0.6; %max value of input allowed
switch input_type
    case 'prbs'
        u = idinput(length(T),'prbs',[0 1],[-uLevel, uLevel]);
    case 'GaussianWhite'
        u = randn(length(T),1)*sqrt(uLevel);
 case 'sweptsine'
        f0 = 0.002;
        u = uLevel*sin(2*pi*f0*T.*T);
    case 'step'
        u = -uLevel*ones(size(T));
        ind = floor(length(T)/5);
        u(ind:end) = uLevel;

    otherwise
        error('bad')
end
%clip:
u(find(u>uLevel))=uLevel;
u(find(u<-uLevel))=-uLevel;

%------

% u_struct =  [];
% t = [0:1:10000]'*Ts;
% u = sin(2*pi*0.01*t.^2);
u_struct.time = T;
u_struct.signals.values = u;
u_struct.signals.dimensions = 1;
t_final=T(end);

simOut=sim('plantForExperiment')

%extract y
y_struct = simOut.y_struct;
tout = y_struct.time;
y = y_struct.signals.values;

%plot
figure
subplot(211)
plot(T,u);ylabel('u');
subplot(212);
plot(T,y);ylabel('y');


simOut =
```

```
Simulink.SimulationOutput:
                    tout: [10002x1 double]
                y_struct: [1x1 struct]

    SimulationMetadata: [1x1 Simulink.SimulationMetadata]
          ErrorMessage: [0x0 char]
```

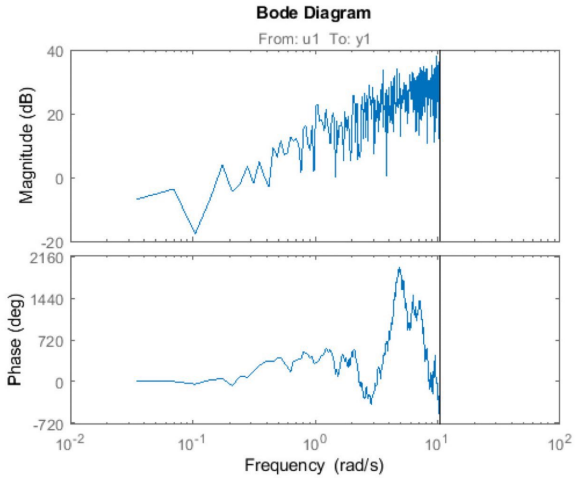




Step 2: estimate G(jw) = Y(jw)/U(jw) from u(t),y(t),

by using the etfe command Step 2.1: put the data in the right format that etfe wants: idDataForThePlant = iddata(y,u,Ts); Phat_DT = etfe(idDataForThePlant);

```
figure
Q=6000;
numFreqRes=300;
uetfe=u(end-Q+1:end);
yetfe=y(end-Q+1:end);
idDataForThePlant = iddata(yetfe,uetfe,Ts);
smoothingWindowSize = Q/10;
Phat_DT = etfe(idDataForThePlant,smoothingWindowSize,numFreqRes);
bode(Phat_DT)
```



compare with the true plant's frequency response.

Beware: the plant and the estimated plant are both discrete time transfer functions, so the frequency axis only goes up to the Nyquist frequency, which is half of the sampling frequency.

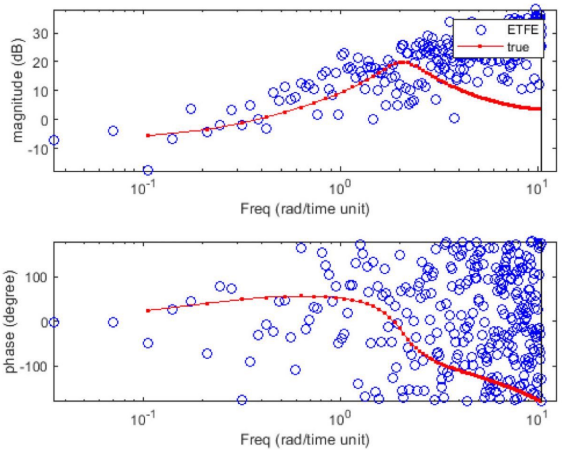
```
plantEstFreqResp=Phat_DT.ResponseData;
plantEstFreqResp = plantEstFreqResp(:);
if ~strcmp(Phat_DT.FrequencyUnit,'rad/TimeUnit')
    freqs_forETFE_rad = 2*pi*Phat_DT.Frequency;
else
    freqs_forETFE_rad = Phat_DT.Frequency;
end
```

compare the true with estimated

```
figure
gainph=[];
phaph=[];

subplot(211);%mag plot
%true plant:
```

```
gainph(1)  =
 semilogx(freqs_forETFE_rad,20*log10(abs(plantEstFreqResp)),'bo');
 hold on
gainph(2)  =
 semilogx(freqs_forPlant_rad,20*log10(abs(plantFreqResp)),'r.-');
% gainph(3)  =
 semilogx(freqs_forPlant_CT_rad,20*log10(abs(plantFreqResp_CT)),'r-');
% show the freq = pi location
plot(Fs*[pi, pi],20*log10([min(abs(plantFreqResp))
 max(abs(plantFreqResp))]),'k-');
axis tight
yy = ylim;
plot(Fs*[pi, pi],yy,'k-');
xlim([0 1.2*Fs*pi]);
xlabel('Freq (rad/time unit)');
ylabel('magnitude (dB)');
legend(gainph,'ETFE','true','true(CT)');

subplot(212); %phase plot
phaph(1)  = semilogx(freqs_forETFE_rad,angle(plantEstFreqResp)*180/
pi,'bo'); hold on
phaph(2)  = semilogx(freqs_forPlant_rad,angle(plantFreqResp)*180/
pi,'r.-');
% phaph(3)  =
 semilogx(freqs_forPlant_CT_rad,angle(plantFreqResp_CT)*180/pi,'r-');
% show the freq = pi location
axis tight
yy = ylim;
plot(Fs*[pi, pi],yy,'k-');
xlim([0 1.2*Fs*pi]);
xlabel('Freq (rad/time unit)');
ylabel('phase (degree)');
```

Warning: Ignoring extra legend entries.



Published with MATLAB® R2020a

Table of Contents

an example of sine sweep identification of frequency response of a KNOWN plant, so that the answer can be verified

```
Prabir Barooah
I have coded the algorithm and verified that it works, meaning, it estimat
the frequency response of the plant quite accurately even with large amoun
sensor noise, as long as the parameters that are up to the designer (trans
number of periods to average over etc.) are chosen appropriately. You only
appropriate values of those parametrs and hit the run button.
1. The default sine sweep parameters  I have included below
ARE DELIBERATELY POORLY CHOSEN.  YOU WILL HAVE TO DO A FEW SINGLE-SINE EXP
(USE THE FIRST SEGMENT OF THE SCRIPT MARKED "PRELIMINARY INVESTIGATION")
-- AND THINK -- TO FIGURE OUT APPROPRIATE VALUES
2. The script estimates the frequency response of the plant P(jw) at one
specific frequency. You will need to modify the script a tiny bit to
repeat the estimation at a number of frequency values.
```

```
clear all

conductSingleExp = 1;
conductSineSweepExp = 1; %specify the parameters(see below) before you
 set this to 1, otherwise
% the script will not run

%%------ specify the plant ----
% You can of course try with a different plant, but I'd suggest that
 at
% first you do not touch this part.
P_TF = tf([10, 2],[1 1 4.25]);
[num,den]  = tfdata(P_TF,'v');
[A,B,C,D] = tf2ss(num,den);
Plant = ss(A,B,C,D);
n = size(A,1);
```

```
A_u = 1; %Amplitude of input sinusoid
%%------
```

user inputs (common):

you can vary these parameters: start by setting them to 0, which will make the system identification task easier. Then you should increase them and check

```
%how large you can make them and still identify the plant accurately
 (you
%will have to decide for yourself how accurate is accurate enough)
std_dev_noise  = 0.1; %standard deviation of the sensor noise
X0factor = 10;
```

more user inputs

```
if conductSingleExp==1
    Ts = 1/550;
    freqSingleExp = 100; %rad/time unit
end
```

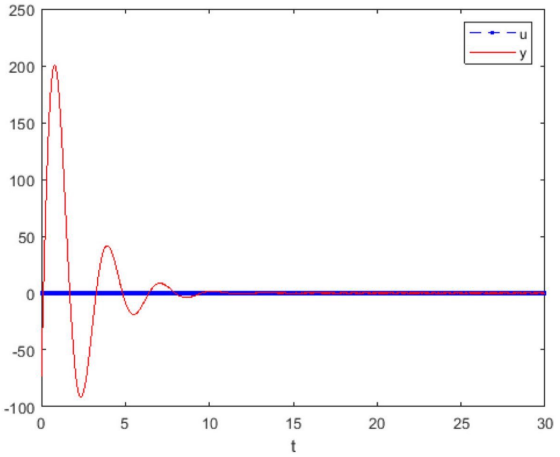
user inputs (for sine sweep only)

```
if conductSineSweepExp==1
    %--------------------------
    omega_array =
 [100,10,0.08,0.7,0.4,0.9,0.6,0.02,15,40,65,89,1,450,3,260,3.20]; %it
 needs to be a row or column vecor of frequencies in
    % "radians/time unit"
    %--------------------------

    %--------------------------
    decay_time = [4]; %in "time units"
    %--------------------------

    %--------------------------
    num_cycles2average_nom = [100]; %needs to be a positive integer
    %--------------------------

    Fs = [300]; %sampling period in "samples/time unit"
    % choose it large enough to give you enough samples in one period
 of the sinusoid
    % (depends on the highest frequency you use, of course), but not
 so
    % high that the simulation takes foreever.
    Ts = 1/Fs;

end
```

----- PRELIMINARY INVESTIGATION - ---

RESPONSE TO SINE INPUT AT SOME ARBITRARY FREQUENCY

```
if conductSingleExp ==1
    x0 = X0factor*randn(n,1); %somewhat random initial state
    time = [0:Ts:30]';
    noise = randn(length(time),1)*std_dev_noise;
    u = A_u*sin(freqSingleExp*time);
    y_at_omega = lsim(Plant,u,time,x0)+noise;

    figure
    plot(time,u,'b.--',time,y_at_omega,'r');
    xlabel('t'); legend('u','y');
end
```



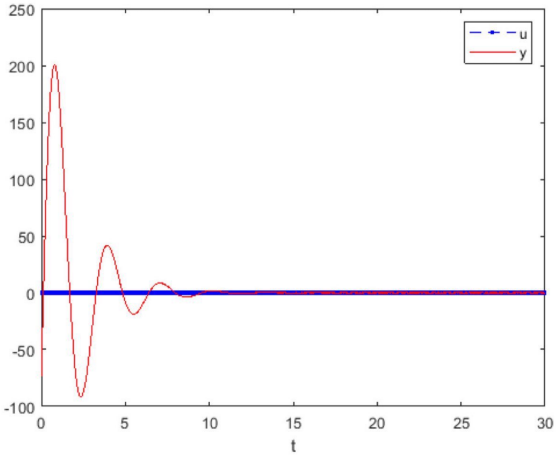
---- SINE SWEEP ---

```
if conductSineSweepExp==1

    g_hat_array = nan*ones(length(omega_array),1);
    theta_hat_array = nan*ones(length(omega_array),1);
```

```
    for omega_index = 1:length(omega_array)

        omega = omega_array(omega_index);

        num_cycles2average = num_cycles2average_nom
+5*ceil(omega);%this increases
        % the value of N for higher frequencies.

        t_final = decay_time+(2*pi/omega)*num_cycles2average;
        time = [0:Ts:t_final]';
        u = A_u*sin(omega*time);

        x0 = X0factor*randn(n,1); %random initial state
        noise = randn(length(time),1)*std_dev_noise;
```

do experiment to colect data

```
        y_at_omega = lsim(Plant,u,time,x0)+noise;
```




```
        %cut the transient period out, so that the data is consistent
with the
        %theory of sine in sine out
        inds2average = [ceil(decay_time/Ts):1:length(time)]';
        N = length(inds2average); %number of samples to average over
```

```
        cosine_vector = cos(omega*time);
        sine_vector = sin(omega*time);
        ZcN = y_at_omega(inds2average)'*cosine_vector(inds2average);
        ZsN = y_at_omega(inds2average)'*sine_vector(inds2average);
        g_hat_omega = 2/A_u/N*sqrt(ZcN^2+ZsN^2); %gain est
        theta_hat_omega = atan2(ZcN,ZsN); %phase est, in rad

        %save estimates
        g_hat_array(omega_index) = g_hat_omega;
        theta_hat_array(omega_index) = theta_hat_omega;

        disp(['done with freq = ',num2str(omega),' rad/sec']);
```

```
done with freq = 100 rad/sec

done with freq = 10 rad/sec

done with freq = 0.08 rad/sec

done with freq = 0.7 rad/sec

done with freq = 0.4 rad/sec

done with freq = 0.9 rad/sec

done with freq = 0.6 rad/sec

done with freq = 0.02 rad/sec

done with freq = 15 rad/sec

done with freq = 40 rad/sec

done with freq = 65 rad/sec

done with freq = 89 rad/sec

done with freq = 1 rad/sec

done with freq = 450 rad/sec

done with freq = 3 rad/sec

done with freq = 260 rad/sec

done with freq = 3.2 rad/sec
```

```
    end
```

compute the true magnitude and phase at a large number of frequencies:

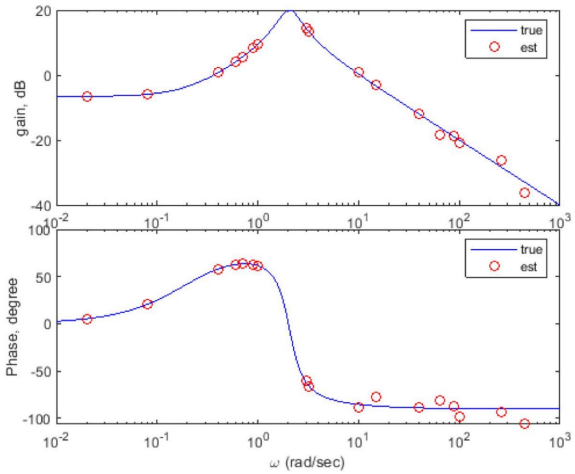
```
w = logspace(-2,3,1000);
[Gjw] = freqresp(Plant,w);
Gjw = Gjw(:);
```

superimpose the estimate and the true frequency response on the same Bode plot

```
bodefig = figure
ax1 = axes('position',[0.1300 0.55 0.7750 0.4])
semilogx(w,20*log10(abs(Gjw)),'b-');
hold on;
semilogx(omega_array,20*log10(g_hat_array),'ro');
ylabel('gain, dB');
legend('true','est');

ax2 = axes('position',[0.1300 0.1 0.7750 0.4])
semilogx(w,angle(Gjw)*180/pi,'b-');
hold on;
semilogx(omega_array,theta_hat_array*180/pi,'ro');
xlabel('\omega (rad/sec)');
ylabel('Phase, degree');
legend('true','est');
```

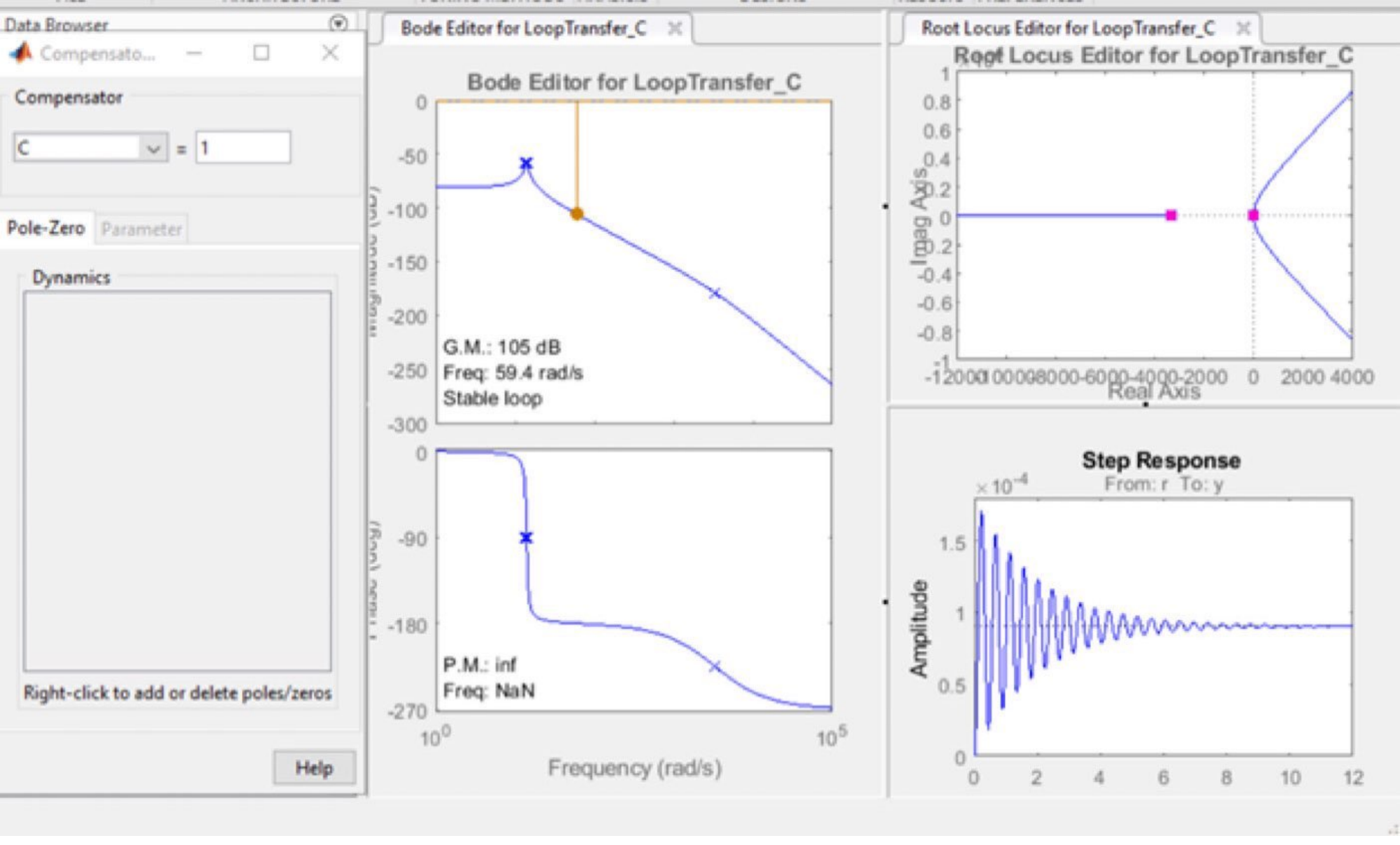
```
bodefig =

  Figure (2) with properties:

      Number: 2
        Name: ''
       Color: [0.9400 0.9400 0.9400]
    Position: [440 298 560 420]
       Units: 'pixels'

  Use GET to show all properties


ax1 =

  Axes with properties:

              XLim: [0 1]
              YLim: [0 1]
            XScale: 'linear'
            YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1300 0.5500 0.7750 0.4000]
            Units: 'normalized'

  Use GET to show all properties


ax2 =

  Axes with properties:
```

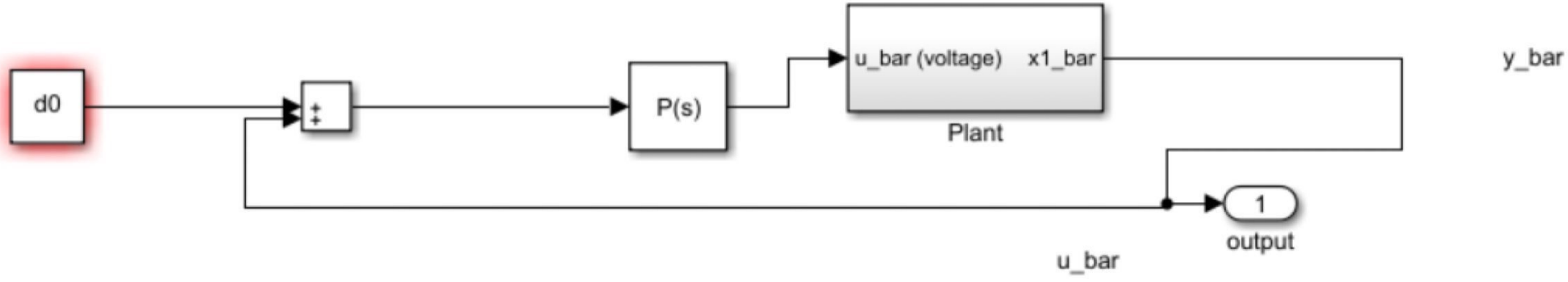
```
            XLim: [0 1]
            YLim: [0 1]
          XScale: 'linear'
          YScale: 'linear'
   GridLineStyle: '-'
        Position: [0.1300 0.1000 0.7750 0.4000]
           Units: 'normalized'

  Use GET to show all properties
```




```
end
```

Published with MATLAB® R2020a





```
clc
clear
alpha = 10^-4;
m = 0.5;
g = 9.81;
R = 50;
L = 0.015;
d_0 = 0.1;
beta = 0.5;
x2bar_init = 1;
x1bar_init = 0;
x3bar_init = 1;

A = [0 1 0; -2*g/d_0 -beta/m (2*alpha/d_0)*sqrt(g/(alpha*m));0 0 -R/L];
B = [0 0 1/L]';
C = [1 0 0];
D = 0;
%
% SS = ss(A , B , C, D);
[NUM, DEN] = ss2tf(A,B,C,D);
H = tf(NUM,DEN);

v = sisotool(H);

sim('plant_ElectromagnetUpward')
```