ECE 147C
Solutions, Problem 4.2 (Selected Parameters)

The data provided was obtained from a system with transfer function

$$H(z) = \frac{z - 0.5}{(z - 0.3)(z - p)}$$

where p is unknown. Use the least-squares method to determine the value of p.

```
clc; clear all; close all;
```

Load data

```
my_data = load('id_p5.mat');
y = my_data.y; u = my_data.u;
clearvars -EXCEPT u y

ybar = y(2:end) - 0.3*y(1:end-1);
ubar = u(2:end) - 0.5*u(1:end-1);

whos
```

```
  Name         Size              Bytes  Class     Attributes

  u          101x1                 808  double
  ubar       100x1                 800  double
  y          101x1                 808  double
  ybar       100x1                 800  double
```

```
Y   = ybar(2:end)-ubar(1:end-1);
PHI = ybar(1:end-1);

whos Y PHI
```

```
  Name        Size              Bytes  Class     Attributes

  PHI        99x1                 792  double
  Y          99x1                 792  double
```

The parameter vector is found via

```
theta = inv(PHI'*PHI)*PHI'*Y
```

```
theta =

    0.7000
```

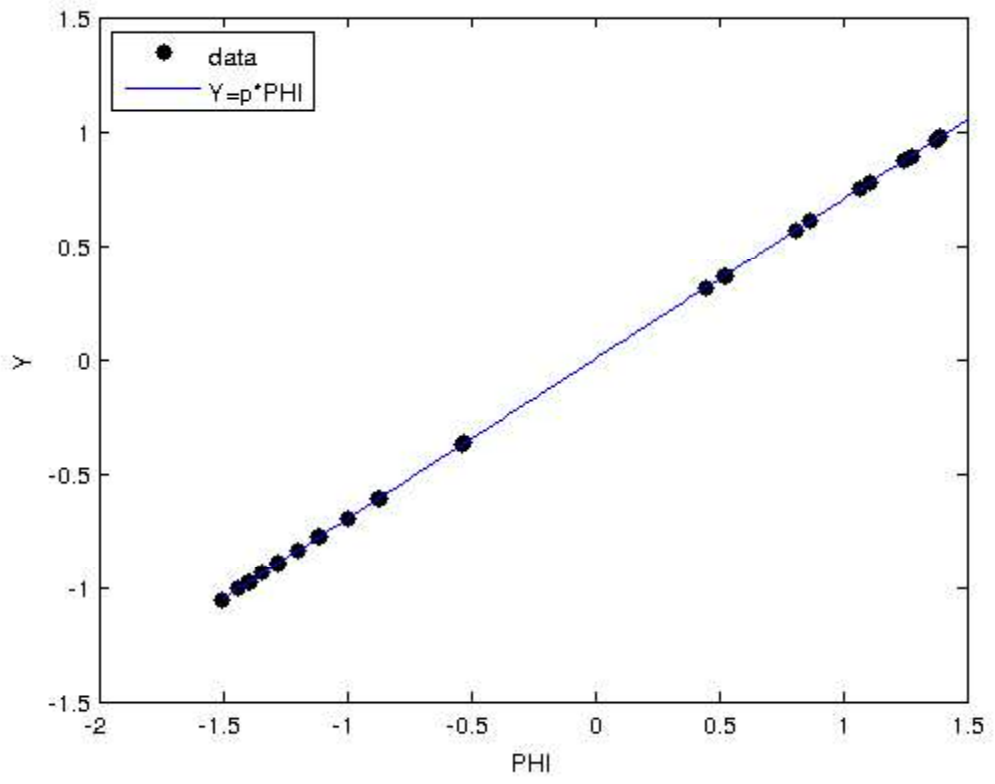
Hence the value of p is

```
p = theta
p =

    0.7000
```

Note that for this simple example, and are 1-dimensional, this problem is really just all about finding the slope of the Y-vs-PHI line:

```
figure; plot(PHI,Y,'k.','markersize',20); xlabel('PHI'); ylabel('Y');
x=linspace(-1.5,1.5);
y=x*p;
hold on; plot(x,y,'b-'); legend('data','Y=p*PHI','location','northwest')
```



ECE 147C
Solutions, Problem 5.1 (Input Magnitude)

A Simulink block that models a nonlinear spring-mass-damper system is provided.

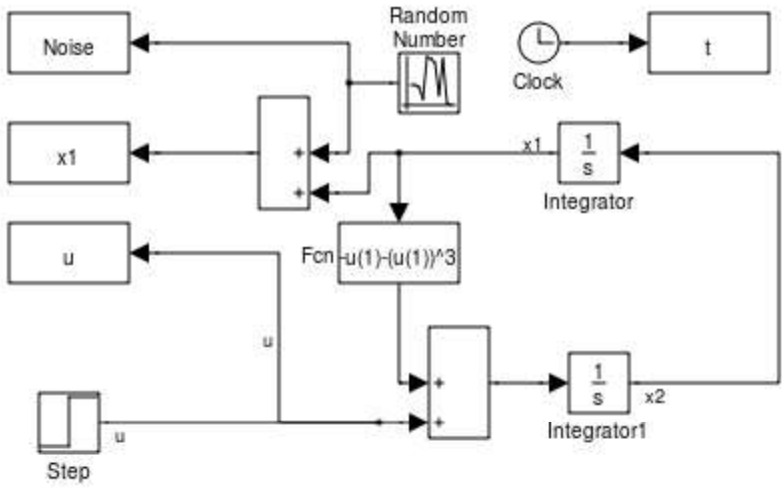
Contents

Part 1.

Use the Simulink block to generate the system's response to step inputs with amplitude 0.25 and 1.0 and no measurement noise.

```
clc; clear all; close all
warning off;

model_name = 'id_p3_mod';
open(model_name)
```



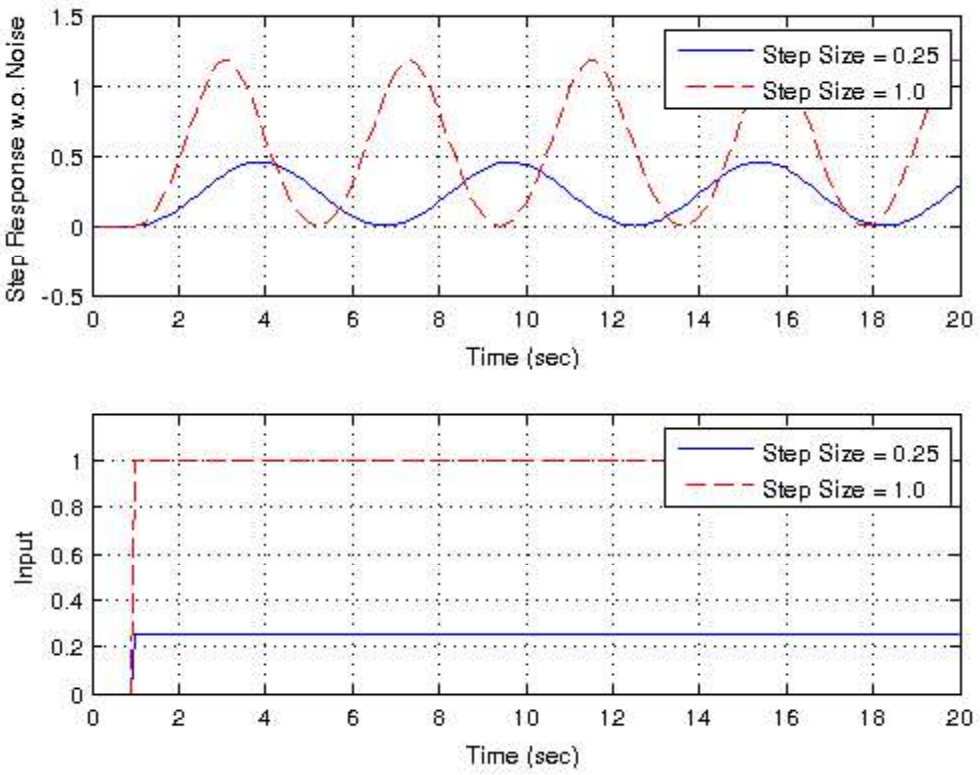
Run system.

```
sample = 0.1;
noisemag = 0.0;
noise = (noisemag)^2;
step_mag = 0.25;
sim(model_name,20);
U1 = u; Y1 = x1;
step_mag = 1;
```

```
sim(model_name,20);
U2 = u; Y2 = x1;
```

Plot.

```
figure(1);
subplot(211); plot(t,Y1,'b-',t,Y2,'r--'); grid on; xlabel('Time (sec)');
ylabel('Step Response w.o. Noise');
legend('Location','NorthEast','Step Size = 0.25','Step Size = 1.0');
subplot(212); plot(t,U1,'b-',t,U2,'r--'); grid on; xlabel('Time (sec)');
ylabel('Input'); axis([0,20,0,1.2])
legend('Location','NorthEast','Step Size = 0.25','Step Size = 1.0');
```



Part 2.

For each set of data, use the least-squares method to estimate the systems transfer function. Try a few values for the degrees of the numerator and denominator polynomials m and n. Check the quality of the model by following the validation procedure outlines above.

Important: write MATLAB scripts to automate these procedures. These scripts should take as inputs the simulation data upkq, ypkq, and the integers m, n.

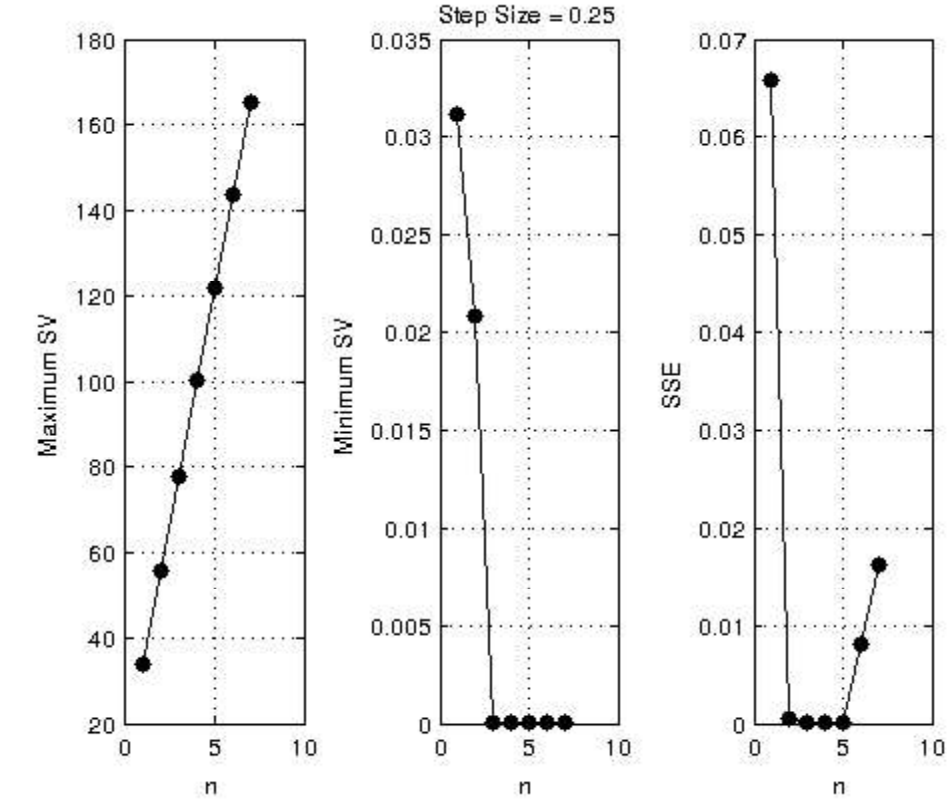
Please see the documentation for the my_id.m function in this same directory.

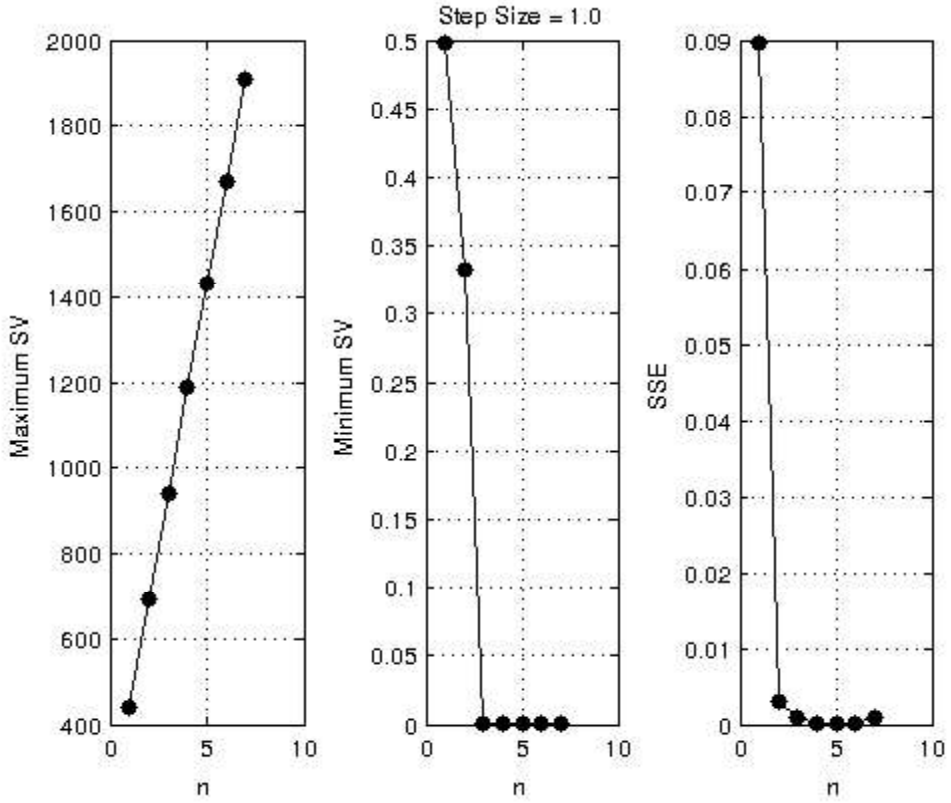
```
num1 = {};
num2 = {};
den1 = {};
den2 = {};

for n = 1:7
    m = n;
    [num1{n},den1{n},SSE1(n),MnSV1(n),MxSV1(n)] = my_id(U1,Y1,n,m);
    [num2{n},den2{n},SSE2(n),MnSV2(n),MxSV2(n)] = my_id(U2,Y2,n,m);
end
```

The lecture notes says that to evaluate the quality of the model, we look at the SSE and the max and min singular values.

```
i = 1:7;
figure(2);
subplot(133); plot(i,SSE1,'k.-','markersize',20); xlabel('n'); ylabel('SSE');
grid on;
subplot(132); plot(i,MnSV1,'k.-','markersize',20); xlabel('n');
ylabel('Minimum SV'); grid on;
title('Step Size = 0.25');
subplot(131); plot(i,MxSV1,'k.-','markersize',20); xlabel('n');
ylabel('Maximum SV'); grid on;
figure(3);
subplot(133); plot(i,SSE2,'k.-','markersize',20); xlabel('n'); ylabel('SSE');
grid on;
subplot(132); plot(i,MnSV2,'k.-','markersize',20); xlabel('n');
ylabel('Minimum SV'); grid on;
title('Step Size = 1.0');
subplot(131); plot(i,MxSV2,'k.-','markersize',20); xlabel('n');
ylabel('Maximum SV'); grid on;
```


Step Size = 0.25
Maximum SV
Minimum SV
SSE
n
n
n


Step Size = 1.0
Maximum SV
Minimum SV
SSE
n
n
n

In both the 0.25-step and 1.0-step cases, we observe that the SSE decreases sharply between n=1 to n=3. The minSV gets pretty small, but not too bad. Hence n = m = 3 (or 4) are the best values to be chosen.

Here are the estimates of the n=m=3 transfer function, for each of the two step sizes:

```
sys1 = tf(num1{3},den1{3})
sys2 = tf(num2{3},den2{3})
```

```
Transfer function:
-1.495e-14 s^3 + 0.006303 s^2 - 0.00246 s - 0.003591
-----------------------------------------------------
         s^3 - 2.965 s^2 + 2.943 s - 0.9774


Transfer function:
-4.78e-15 s^3 + 0.006303 s^2 - 0.002259 s - 0.003439
-----------------------------------------------------
         s^3 - 2.933 s^2 + 2.892 s - 0.9573
```

Part 3.

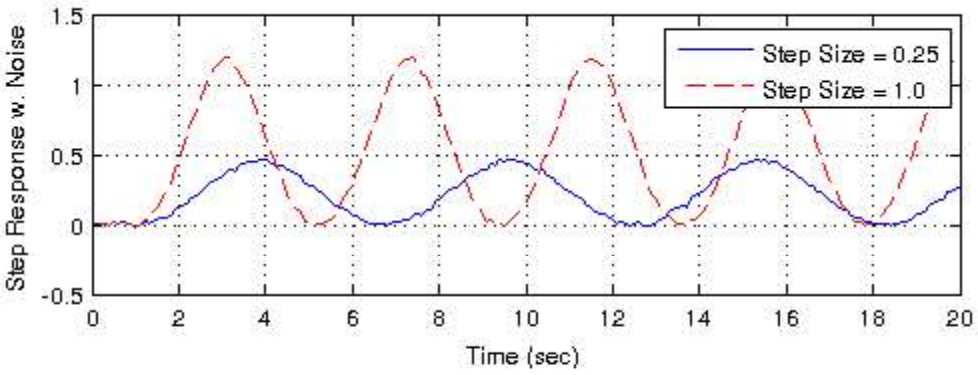
Use the Simulink block to generate the system's response to step inputs and measurement noise with intensity 0.01.

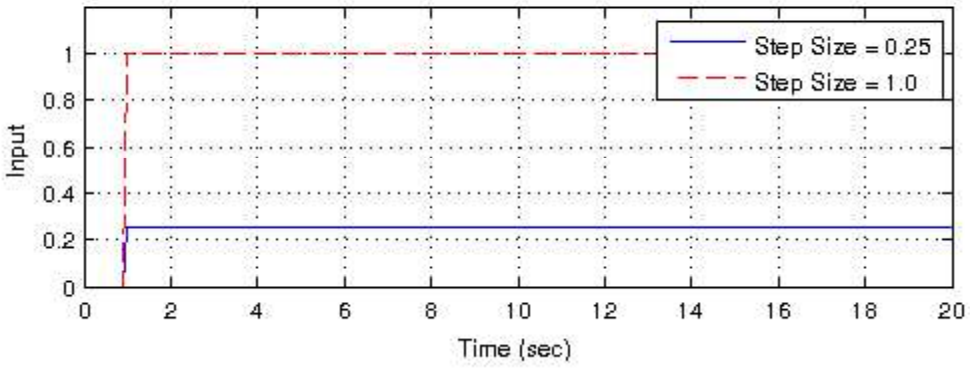
For the best values of m and n determined above, plot the SSE vs. the step size. Which step-size leads to the best model?

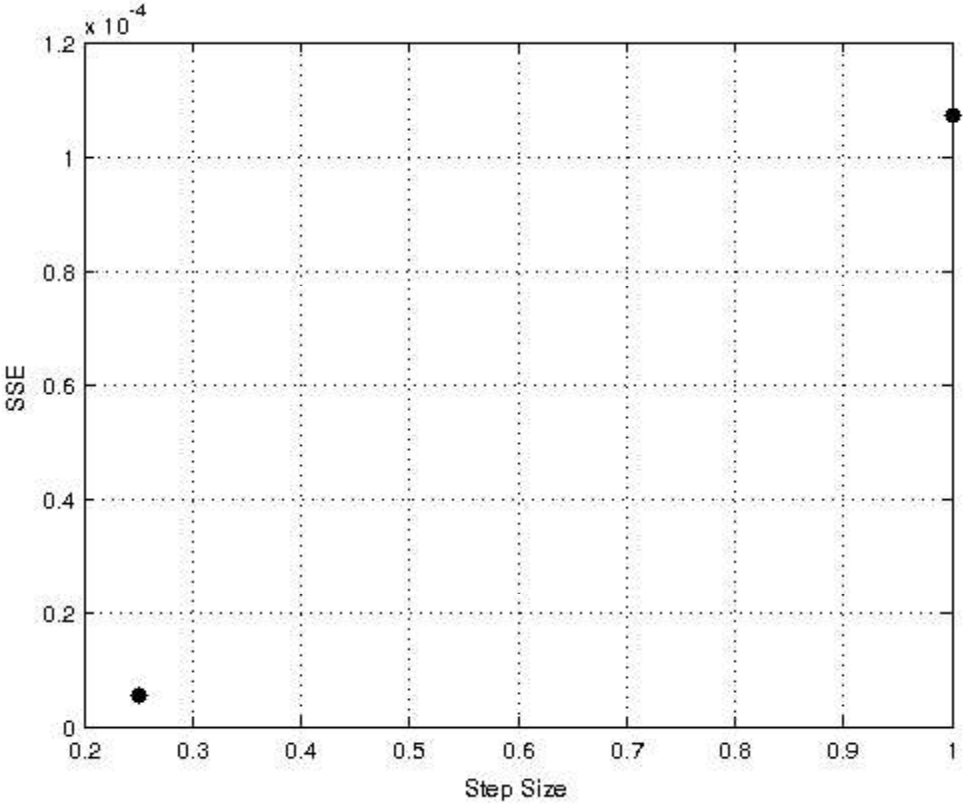
```
noisemag = 0.01;
noise = (noisemag)^2;
step_mag = 0.25;
sim(model_name,20);
U1 = u; Y1 = x1;
step_mag = 1;
sim(model_name,20);
U2 = u; Y2 = x1;

figure(4);
subplot(211); plot(t,Y1,'b-',t,Y2,'r--'); grid on; xlabel('Time (sec)');
ylabel('Step Response w. Noise');
legend('Location','NorthEast','Step Size = 0.25','Step Size = 1.0');
subplot(212); plot(t,U1,'b-',t,U2,'r--'); grid on; xlabel('Time (sec)');
ylabel('Input'); axis([0,20,0,1.2])
legend('Location','NorthEast','Step Size = 0.25','Step Size = 1.0');
ylabel('Input');

figure(5); plot([0.25,1.0],[SSE1(4),SSE2(4)],'k.','markersize',20);
ylabel('SSE'); xlabel('Step Size'); grid on;
```


Step Response w. Noise
Time (sec)
Step Size = 0.25
Step Size = 1.0


Input
Time (sec)
Step Size = 0.25
Step Size = 1.0


x 10^-4
SSE
Step Size

This figure shows the SSE vs. the step size for the 0.25-step and 1.0-step. Clearly, the SSE for a step of size 0.25 is smaller. However, choosing a small input will result in a small output which will be more difficult to measure in presence of noise.

ECE 147C
Solutions, Problem 5.2 (Model Order)

Use the data provided to identify the transfer function of the system. Use the procedure outlined above to determine the order of the numerator and denominator polynomials. Plot the largest and smallest singular value of R and the SSE as a function of n.

Important: write MATLAB R scripts to automate this procedure. These scripts should take as inputs the simulation data u(k), y(k), and the integers m, n.

Get data.

```
clc; clear all; close all;
my_data = load('id_p4.mat');
y = my_data.y; u = my_data.u;
clearvars -EXCEPT u y
l = length(y);
```

For each of a few guesses of model order, identify the numerator and denominator coefficients.

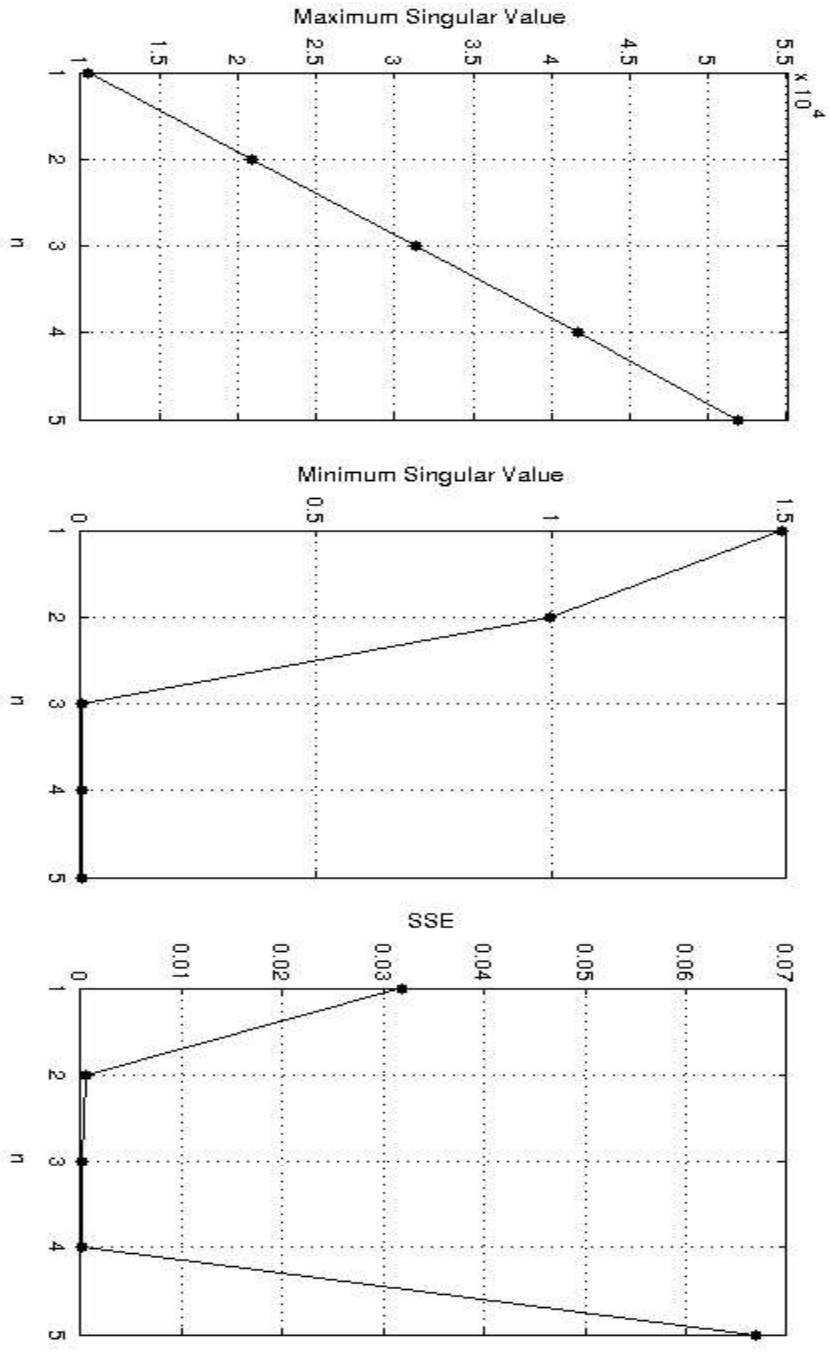
```
n_guesses = 1:5;

for n = n_guesses
    m = n;
    [num,den,SSE(n),MnSV(n),MxSV(n)] = my_id(u,y,n,m);
end
```

```
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  1.198276e-17.
```

Plot.

```
figure(1); set(gcf,'position',[660 391 1000  433])
subplot(131); plot(n_guesses,MxSV,'k.-','markersize',15); ylabel('Maximum
Singular Value');
grid on; xlabel('n');
subplot(132); plot(n_guesses,MnSV,'k.-','markersize',15); ylabel('Minimum
Singular Value');
grid on; xlabel('n');
subplot(133); plot(n_guesses,SSE,'k.-','markersize',15); ylabel('SSE');
xlabel('n'); grid on;
```



From the figure, n = m = 3 is the best choice.

ECE 147C
Solutions, Problem 5.3 (Sampling Frequency)

Contents

Consider a continuous-time system with transfer function

$$P(s) := \frac{4\pi^2}{s^2 + \pi s + 4\pi^2}$$

```
clc; clear all; close all;

my_sys =  tf(4*pi^2,[1, pi, 4*pi^2]);
```

Part 1.

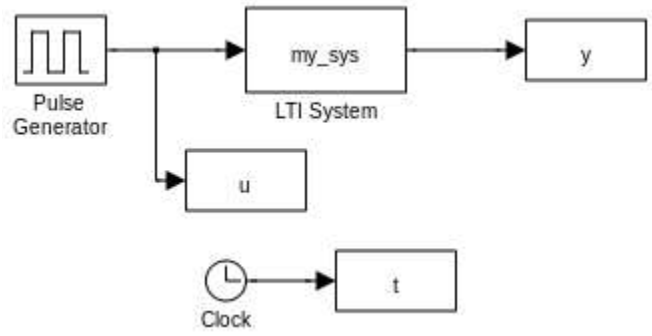
Build a Simulink model for this system and collect input/output data for an input square wave with frequency .25Hz and unit amplitude for two sampling frequencies Ts = .25sec and Ts = .0005sec.

```
model_name = 'P5_3_mod'

open(model_name)
```

```
model_name =

P5_3_mod
```



Run the system.

```
sample = 0.25;
sim(model_name,20);
U1 = u; Y1 = y;
sample = 0.0005;
sim(model_name,20);
U2 = u; Y2 = y;

clearvars -EXCEPT U1 U2 Y1 Y2
```

Part 2.

Identify the system's transfer function without down-sampling.

```
n = 2; m = 1;
[num,den,SSE,MnSV,MxSV] = my_id(U1,Y1,n,m);
fprintf('Identification using data sampled at steps of 0.25 sec:\n\n');
fprintf('SSE = %i\n\n',SSE); fprintf('Estimated');
% display(d2c(tf(num,den,0.25,'zoh'),'zoh'));
display(d2c(tf(num,den,0.25),'zoh'));

clearvars -EXCEPT U1 U2 Y1 Y2
```

```
Identification using data sampled at steps of 0.25 sec:

SSE = 1.742270e-04

Estimated
Transfer function:
 -0.001657 s + 39.48
---------------------
s^2 + 3.144 s + 39.48
```

```
n = 2; m = 1;
[num,den,SSE,MnSV,MxSV] = my_id(U2,Y2,n,m);
fprintf('Identification using data sampled at steps of 0.0005 sec:\n\n');
fprintf('SSE = %i\n\n',SSE); fprintf('Estimated');
%display(d2c(tf(num,den,0.0005,'zoh'),'zoh'));
display(d2c(tf(num,den,0.0005),'zoh'));

clearvars -EXCEPT U1 U2 Y1 Y2
```

```
Identification using data sampled at steps of 0.0005 sec:

SSE = 3.675928e-11

Estimated
Transfer function:
 -4.56e-09 s + 39.48
---------------------
s^2 + 3.142 s + 39.48
```

Part 3.

Identify the system's transfer function using the data collected with Ts = .0005sec but down-sampled.

```
n = 2; m = 1;
L = 0.25/0.0005;
i = 1:fix(length(U2)/L);
U3(i) = U2((i-1)*L+1);
Y3(i) = Y2((i-1)*L+1);
[num,den,SSE,MnSV,MxSV] = my_id(U3,Y3,n,m);
fprintf('Identification using data sampled at steps of 0.0005 sec but\n');
fprintf('down-sampled by a factor of %i:\n\n',L);
fprintf('SSE = %i\n\n',SSE); fprintf('Estimated');
% display(d2c(tf(num,den,0.0005*L,'zoh'),'zoh'));
display(d2c(tf(num,den,0.0005*L),'zoh'));
```

```
Identification using data sampled at steps of 0.0005 sec but
down-sampled by a factor of 500:

SSE = 1.223855e-15

Estimated
Transfer function:
        39.48
----------------------
s^2 + 3.142 s + 39.48
```

Part 4.

Briefly comment on the results.

With more data, we can better understand the behavior of our system. However, using a lot of data pairs may increase the estimation error due to possible singularity of resulting large matrices (and other similar issues). Therefore, it is better to collect more data and then down-sample the result before identification. As it can be seen, this approach has ended up with the least SSE and best approximation of the TF.