HW 3, Student Assignment (Due Friday 9/7/07`)

PART 1 Running Simulink for the Pendulum
Find a computer that can run MATLAB 7 and Simulink. Copy to that computer the Simulink model file pendulum.mdl, the m-file pendeom.m and the MATLAB script called PendAnal.m. These can be found on the class web site at

http://cobweb.ecn.purdue.edu/~andrisan/Courses/AAE421_Fall_2007/AAE421_Buffer_F07/Pendulum/

Start MATLAB. Type simulink from the MATLAB command window. From the Simulink widow open the file pendulum.mdl to view the graphical representation of the pendulum model. Click on the scope to bring up a window that will contain the plots of the dynamic response of the pendulum. With the model window active, select the Start option from the Simulation menu. The time history of the pendulum should appear on the Scope window. One time history s for θ and the other is for the rate of change of θ.

The time histories of θ, the rate of change of θ, and time are saved in the MATLAB workspace in matrices ynlsim and tnlsim. We will use these later.

From the MATLAB command window run the script PendAnal.m by typing PendAnal. A linear simulation will be created and executed from the nonlinear Simulink model. Comparisons of the nonlinear and linear simulations are plotted.

Modify the script PendAnal.m to include your name in the titles of the two figures and hand them in to verify that you have done all this.

PART 2
The script PendAnal.m generates Jacobian matrices (A matrix) for two reference conditions. The first is for x_R=[0 rad, 0 rad/sec]^T, and the second is for x_R=[0 rad, 1 rad/sec]^T. In HW#2 you analytically generated these Jacobian matrices by taking the indicated partial derivatives and evaluating the partials at the first reference condition, x_R=[0,0]^T. Specifically find

$$\text{Jacobian} = \text{A matrix} = \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}} = \begin{bmatrix} \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_2} \\ \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_2} \end{bmatrix}$$

where g_1, g_2, x_1 and x_2 were defined earlier for the pendulum. Compare the A matrix you analytically computed in HW2 with the first A matrix computed by the script PendAnal.m.

Simulink Model for the Simple Pendulum

Simulink is a MATLAB toolbox for simulating dynamic systems, i.e., for determining time responses of linear or nonlinear systems. It is a graphical environment that is quite easy to use with a little practice. It is very easy to add control systems to dynamical models.

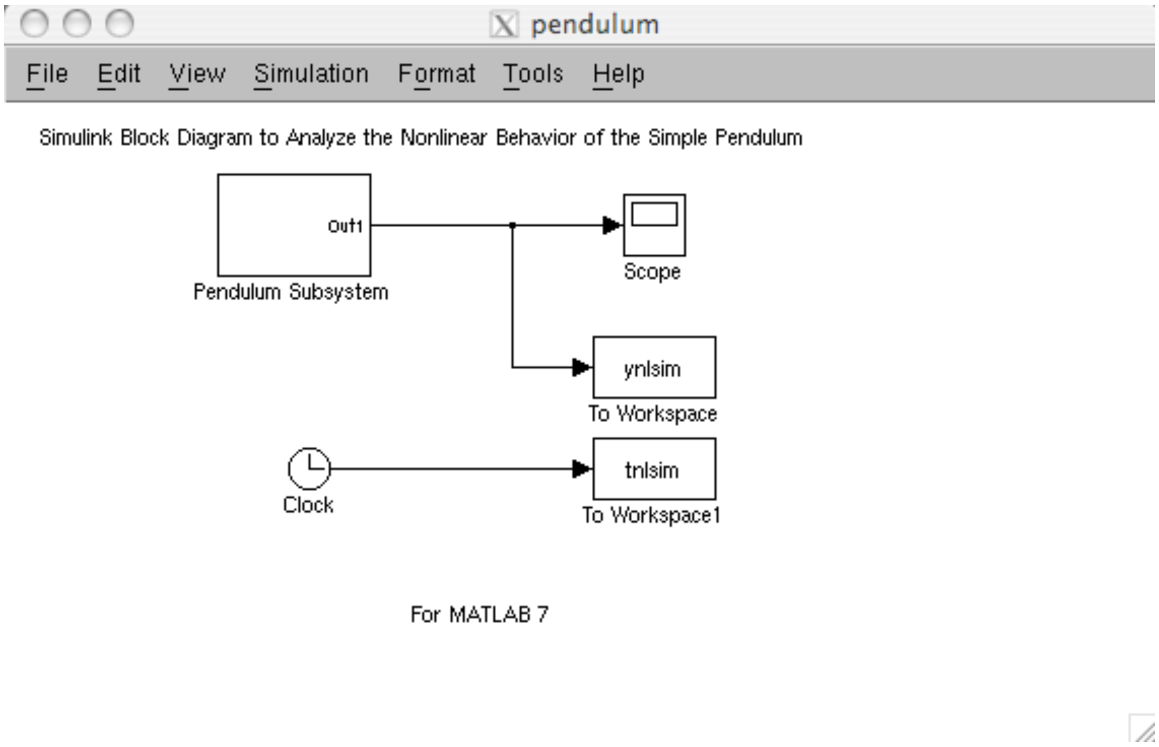
All software described in this document can be found in the class web site
http://cobweb.ecn.purdue.edu/~andrisan/Courses/AAE421_Fall_2007/AAE421_Buffer_F07/Pendulum/

The graphical Simulink model is contained in two files. The first is the Simulink model file. For the pendulum this file is called pendulum.mdl. Files that end in .mdl are interpreted by MATLAB as Simulink model files.

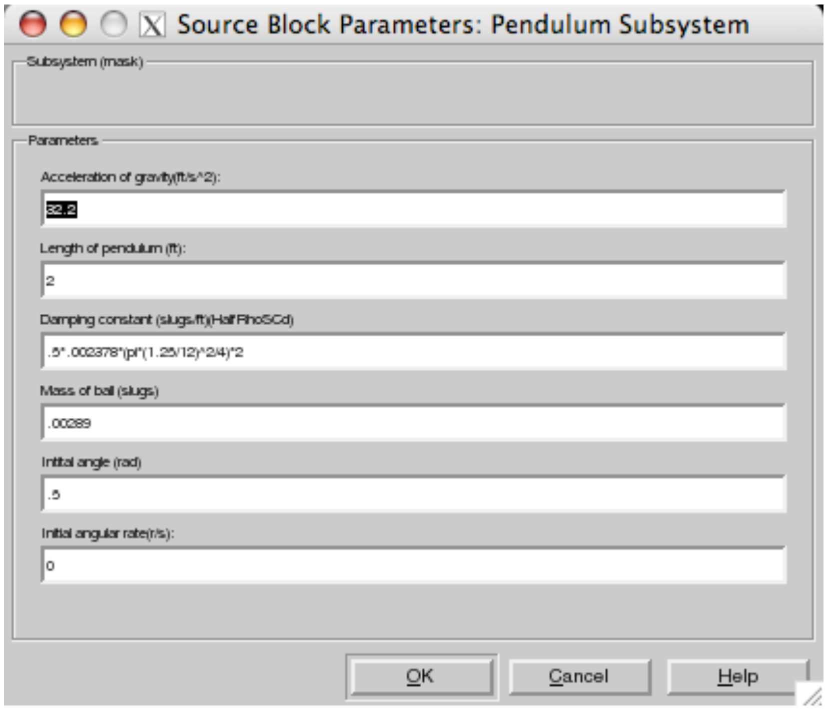
The Simulink toolbox is executed from the MATLAB command window by typing simulink or by clicking on the Simulink icon. The contents of the .mdl files need not be directly viewed or edited. Instead .mdl files are opened in MATLAB when Simulink is active from the file menu of any Simulink window. The graphical depiction of the simulation model is then displayed and the can be modified in a graphical way. Dynamical simulations may also be run from the Simulink window using the Start option from the Simulation menu.

The actual differential equations of the pendulum are stored in the MATLAB m-file called pendeom.m. This file is writen in S-function protocol. A brief description of this protocol is attached. The m-file below contains the nonlinear differential equation model for the pendulum written in S-function protocol.

If you click (or double click or right click or something click) on the Simulink block called “Pendulum Subsystem”, a window of parameters for the pendulum will pop up. This is where g, l, the drag parameter (HalfRhoSCd) and the initial conditions are set. The subsystem “Pendulum Subsystem” is said to be Masked and you are looking at the parameters of the mask. These parameters are passed to the S-function programmed in pendeom.m


pendulum
File Edit View Simulation Format Tools Help
Simulink Block Diagram to Analyze the Nonlinear Behavior of the Simple Pendulum
Out1
Pendulum Subsystem
Scope
ynlsim
To Workspace
Clock
tnlsim
To Workspace1
For MATLAB 7

The above figure is the result of opening pendulum.mdl from the Simulink window.



Acceleration of gravity(ft/s^2): 32.2

Length of pendulum (ft): 2

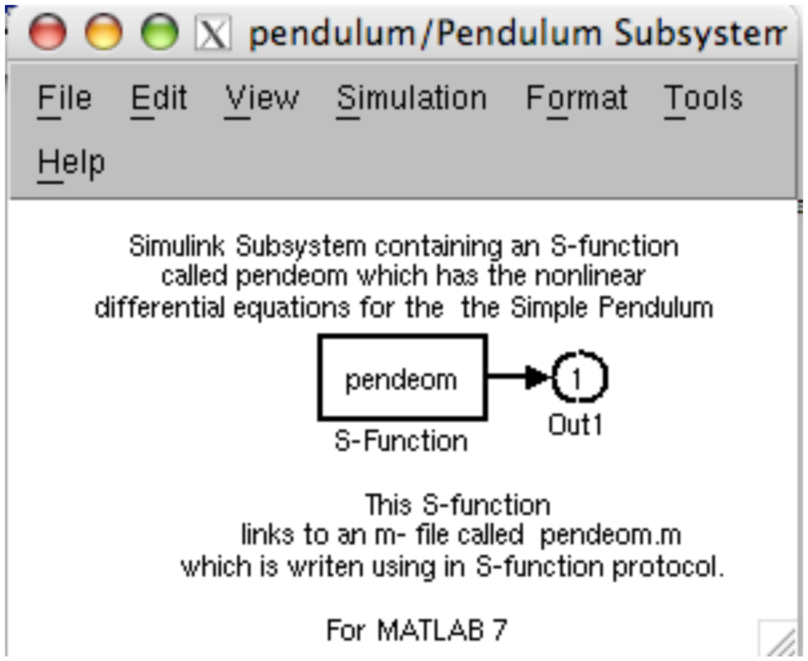
Damping constant (slugs/ft)(HalfRhoSCd): .5*.002378*(pi*(1.25/12)^2/4)*2

Mass of ball (slugs): .00289

Initial angle (rad): .5

Initial angular rate(r/s): 0

The above figure is the result of double clicking (or right clicking) the simulink block called “Pendulum Subsystem”.


pendulum/Pendulum Subsystem
File Edit View Simulation Format Tools
Help
pendeom
S-Function
1
Out1

Simulink Subsystem containing an S-function called pendeom which has the nonlinear differential equations for the the Simple Pendulum

This S-function links to an m- file called pendeom.m which is writen using in S-function protocol.

For MATLAB 7

The above figure is the result of first selecting the "Pendulum Subsystem" block and then selecting "Look Under Mask" from the edit menu. It shows that the differential equations for the pendulum are in the file called "pendeom". This file must be written in S-Function format.

The m-file describing the pendulum (pendeom.m)

This is the contents of the file pendeom.m which is written using the S-function protocol. It contains the differential equations in for the simple pendulum and the output equations.

```
function [sys,x0,str,ts] =
pendeom(t,x,u,flag,g,l,HalfRhoSCd,mass,theta0,thetadot0)
%   S-file for pendulum (with damping term, 8/31/07)
%   This is an S-file subsystem that models a simple pendulum.
%   g is the acceleration of gravity.
%   l is the length of the pendulum.
%   HalfRhoSCd is the collected drag term (.5*rho*S*Cd)
%   mass is the mass of the ball that forms the massive part of the
pendulum.
%   theta0, thetadot0 are the initial condition on theta and thetadot.
switch flag,
  case 0,         % Initialization
    [sys,x0,str,ts]=mdlInitializeSizes(theta0,thetadot0);
  case 1,         % Compute derivatives of continuous states
    sys=mdlDerivatives(t,x,u,g,l,HalfRhoSCd,mass) ;
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u);  % Compute output vector
  case 4,                  % Compute time of next sample
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,                  % Finished. Do any needed
    sys=mdlTerminate(t,x,u);
  otherwise                % Invalid input
    error(['Unhandled flag = ',num2str(flag)]);
end

%************************************************************
%*                    mdlInitializeSizes                    *
%************************************************************
function [sys,x0,str,ts]=mdlInitializeSizes(theta0,thetadot0)
% Return the sizes of the system vectors, initial
% conditions, and the sample times and offets.
sizes = simsizes;   % Create the sizes structure
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);      % load sys with the sizes structure
x0  = [theta0,thetadot0];    % Specify initial conditions for all
states
str = [];    % str is always an empty matrix
ts  = [0 0]; %initialize the array of sample times


%************************************************************
%*                    mdlDerivatives                        *
%************************************************************
function sys=mdlDerivatives(t,x,u,g,l,HalfRhoSCd,mass)
% Compute derivatives of continuous states
x_dot = [x(2); -(g/l)*sin(x(1))-HalfRhoSCd*l*x(2)*x(2)*sign(x(2))/mass]
;
sys = [x_dot(1),x_dot(2)] ;

%************************************************************
%*                    mdlUpdate                             *
%************************************************************
function sys=mdlUpdate(t,x,u)
% Compute update for discrete states. If necessary, check for
% sample time hits.
sys = [];    % Empty since this model has no discrete states.

%************************************************************
%*                    mdlOutputs                            *
%************************************************************
function sys=mdlOutputs(t,x,u)
% Compute output vector given current state, time, and input
sys = x ;

%************************************************************
%*                    mdlGetTimeOfNextVarHit                *
%************************************************************
function sys=mdlGetTimeOfNextVarHit(t,x,u)
% Return the time of the next hit for this block.  Note that
% the result is absolute time.  Note that this function is
% only used when you specify a variable discrete-time sample
% time [-2 0] in the sample time array in sampleTime = 1;
sys = [] ;

%*************************************************************
%*                     mdlTerminate                          *
%*************************************************************
function sys=mdlTerminate(t,x,u)
% Perform any necessary tasks at the end of the simulation
sys = [];
```

MATLAB Script to analyze the pendulum(PendAnal.m)

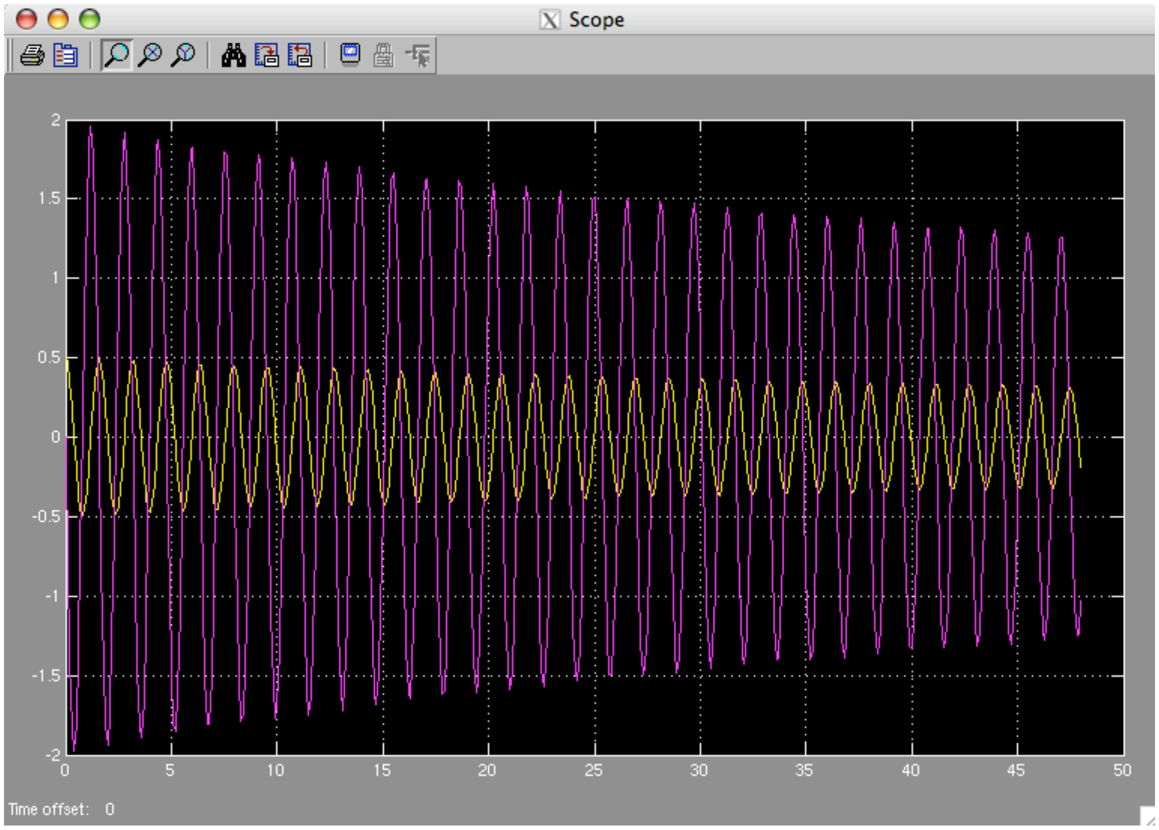
The MATLAB script below should be run only after the Simulink simulation has been run. This allows the nonlinear simulation ot be stored in the matrices ynlsim and tnlsim.

```
% Script to analyse the pendulum system.
% Run the nonlinear simulation (pendulum.mdl) using Simulink before
executing this script.
% The first plot is for a reference solution with
% Theta=0, and Thetadot=0.
% The second plot is for a reference solution with
% Theta=0, and Thetadot not=0. This better matched the damping.


disp(' ');disp('Start here');disp(' ');
disp('Find trim condition')
[X,U,Y,DX]=trim('pendulum')
disp('Reference point about which to linearize')
xR=[0;0]
[a,b,c,d]=linmod('pendulum',xR,[]) %Find linear model of nonlinear
pendulum
b=[0;0]
c=[1,0;0,1]
d=[0;0]
[Wn,Z]=damp(a) % Find properties of the poles of linearized system
period=2*pi/Wn(1)
% Periods in 10 second
disp('Periods in 48 seconds')
P48=48/period
PSYS=ss(a,b,c,d) %Create a linear state space system
disp('Initial condition for linear simulation')
x0=[.5;0] %Use this initial condition for linear simulation
Tfinal=max(tnlsim);
t=0:.1:Tfinal';
u=zeros(size(t));
[y,t]=lsim(PSYS,u,t,x0); %Do linear simulation
%Plot the linear results and the nonlinear results
%  which have been stored in tnlsim and ynlsim.
figure(1)
str1=['YOUR NAME: For Pendulum linearized about theta=
',num2str(xR(1)),', thetadot= ',num2str(xR(2))];
str2=[', Wn= ',num2str(Wn(1)),', Zeta= ',num2str(Z(1))];
```

```
subplot(211)
plot(t,y(:,1),'-',tnlsim,ynlsim(:,1),'x')
title([str1,str2])
xlabel('time (sec)')
ylabel('theta (rad)')
legend('linear sim','nonlinear sim')
subplot(212)
plot(t,y(:,2),'-',tnlsim,ynlsim(:,2),'x')
xlabel('time (sec)')
ylabel('thetaDot (r/s)')
legend('linear sim','nonlinear sim')

% Do everything over again at another linearization point
disp(' ');disp('New reference point');disp(' ');
%xR=[0;1]
xR=[0;.59]  % This seems to give the best match between the time
response of the linear system to the nonlinear system.
[a,b,c,d]=linmod('pendulum',xR,[])
b=[0;0];
c=[1,0;0,1];
d=[0;0];
[Wn,Z]=damp(a)
PSYS=ss(a,b,c,d);
disp('Initial condition for linear simulation')
x0=[.5;0] %Use this initial condition for linear simulation
t=0:.1:Tfinal';
u=zeros(size(t));
[y,t]=lsim(PSYS,u,t,x0);
figure(2)
str1=['YOUR NAME: For Pendulum linearized about theta=
',num2str(xR(1)),', thetadot= ',num2str(xR(2))];
str2=[', Wn= ',num2str(Wn(1)),', Zeta= ',num2str(Z(1))];
subplot(211)
plot(t,y(:,1),'-',tnlsim,ynlsim(:,1),'x')
title([str1,str2])
xlabel('time (sec)')
ylabel('theta (rad)')
legend('linear sim','nonlinear sim')
subplot(212)
plot(t,y(:,2),'-',tnlsim,ynlsim(:,2),'x')
xlabel('time (sec)')
ylabel('thetaDot (r/s)')
legend('linear sim','nonlinear sim')
```



The above plot was generated by Simulink when the simulation was run using the "Simulation Start" command. There are two variables plotted versus time. The smaller one is theta and the larger one is thetadot. The simulation is run for 48 seconds aand there are about 30 cycles of the pendulum in that time.

Output from the script called PendAnal.m

```
Start here

Find trim condition
X =
   1.0e-13 *
   -0.1209
   -0.0000

U =
   Empty matrix: 0-by-1

Y =
   Empty matrix: 0-by-1

DX =
   1.0e-12 *
   -0.0000
```

```
    0.1947
Reference point about which to linearize
xR =
     0
     0

a =
         0    1.0000
  -16.1000   -0.0000

b =
   Empty matrix: 2-by-0

c =
   Empty matrix: 0-by-2

d =
     []

b =
     0
     0

c =
     1     0
     0     1

d =
     0
     0

Wn =
    4.0125
    4.0125

Z =
   1.0e-07 *
    0.1748     <<<< Note poles are undamped, zeta=0.1748e-7
    0.1748     <<<< Note poles are undamped, zeta=0.1748e-7

period =
    1.5659

Periods in 48 seconds
P48 =
   30.6531
```

```
a =
                 x1           x2
   x1             0            1
   x2         -16.1  -1.402e-07


b =
       u1
   x1   0
   x2   0


c =
       x1  x2
   y1   1   0
   y2   0   1


d =
       u1
   y1   0
   y2   0

Continuous-time model.
Initial condition for linear simulation
x0 =
    0.5000
         0
```

New reference point

```
xR =
     0
     1

a =
         0    1.0000
  -16.1000   -0.0280

b =
   Empty matrix: 2-by-0

c =
   Empty matrix: 0-by-2

d =
     []
```

```
Wn =
    4.0125
    4.0125

Z =
    0.0035
    0.0035
Initial condition for linear simulation
x0 =
    0.5000
         0
>>
```

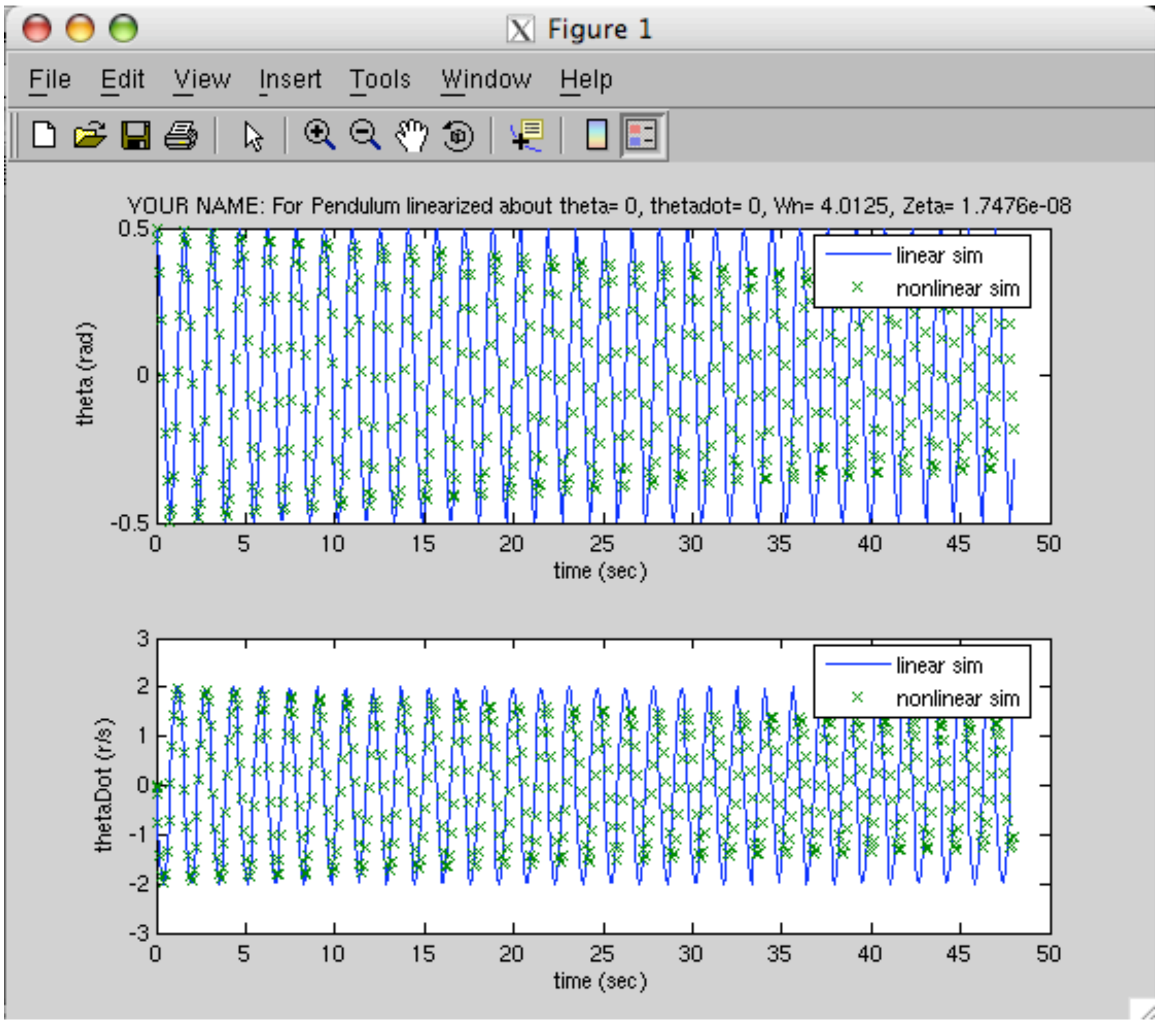
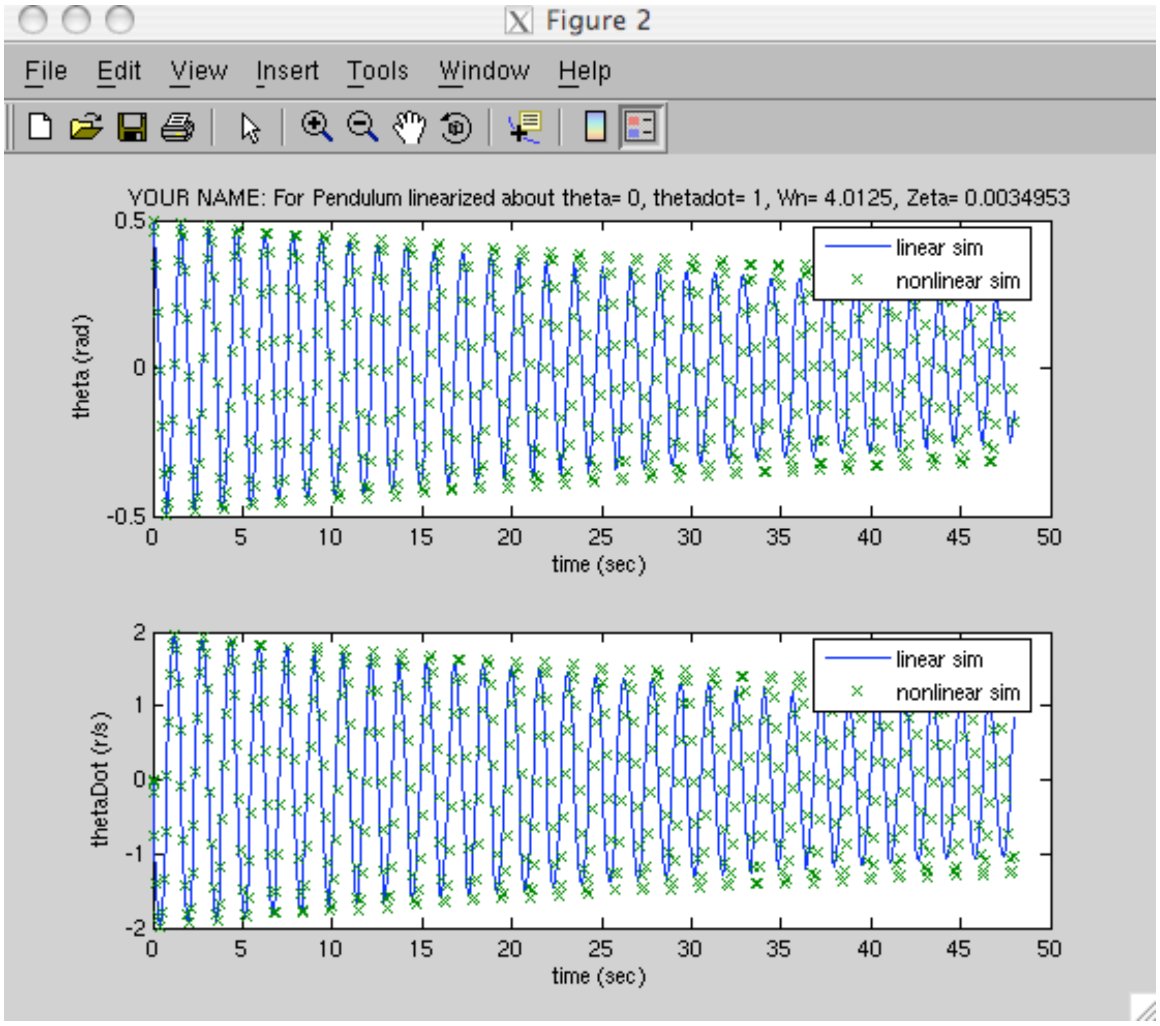


Figure 1
File Edit View Insert Tools Window Help
YOUR NAME: For Pendulum linearized about theta= 0, thetadot= 0, Wn= 4.0125, Zeta= 1.7476e-08
theta (rad)
time (sec)
linear sim
nonlinear sim
thetaDot (r/s)
time (sec)
linear sim
nonlinear sim

This figure was generated by PendAnal.m and is for the first reference condition. The linear system has too little damping (no damping) compared for to the nonlinear simulation. Both simulations use the same Cd.



This figure was generated by PendAnal.m and is for the second reference condition. . The linear system now has too much damping compared for to the nonlinear simulation. If we linearized about $X_R=[0,.59]^T$ the two simulations would agree better.