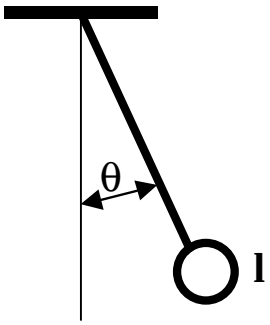
A&AE 421 Dynamic Analysis of a Simple Pendulum

(with corrected damping term, 1/12/01)

Assume that a simple pendulum consists of a ball on a string. Gravity tends to make the pendulum (ball) return to the vertical position. If the ball is given an initial angle, θ, and let go it will oscillate. The aerodynamic drag on the ball tends to make the ball slow down and eventually stop.



Pendulum differential equation

$$\ddot{\theta} = -\frac{\mathbf{g}}{\mathbf{l}}\sin(\theta) - .5\rho \mathbf{S} C_D \mathbf{l}\dot{\theta}^2[\mathrm{sign}(\dot{\theta})]/\mathbf{m}$$

where **g** is the acceleration of gravity**, l** is the length of the pendulum, ρ is the air density, **S** is the cross-sectional area of the ball of the pendulum, **m** is the mass of the ball, and $\mathbf{C_D}$ is the drag coefficient of the ball. Motion variable θ is the angle that the pendulum makes with a vertical line. The sign function is required to insure that the drag force always resists the motion due to the rate of change of angle θ.

Pendulum state space model

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

$$\dot{\bar{\mathbf{x}}} = \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_2 \\ -\frac{\mathbf{g}}{\mathbf{l}}\sin(\mathbf{x}_1) - .5\rho \mathbf{S} C_D \mathbf{l}\mathbf{x}_2^{\,2}\mathrm{sign}(\mathbf{x}_2)/\mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_1(\mathbf{x}_1, \mathbf{x}_2) \\ \mathbf{g}_2(\mathbf{x}_1, \mathbf{x}_2) \end{bmatrix}$$

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://roger.ecn.purdue.edu/~andrisan/Courses/AAE421_S2001/Index.html

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. The contents of the .mdl files need not be directly viewed or edited. Instead .mdl files are opened in MATLAB when Simulink is active. 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

Student Assignment (Due Thursday, 1/18/01)

PART 1 Running Simulink for the Pendulum
Find a computer that can run MATLAB 5 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://roger.ecn.purdue.edu/~andrisan/Courses/AAE421_S2001/Docs_Out/Docs_Out.html

Start MATLAB. Type simulink form the MATLAB command window. 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.

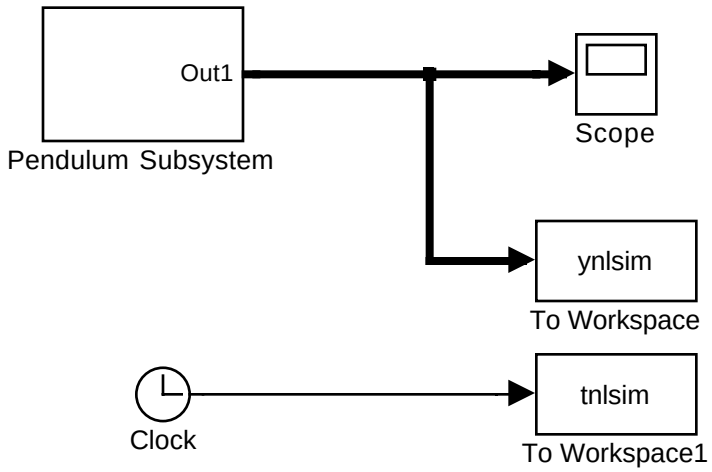
Modify the script PendAnal.m to include your name in the titles of the two figures and hand them in to verify that you have done all this.

PART 2
The script PendAnal.m generates Jacobian matrices (a matrix) for two reference conditions. The first is for $x_R=[0,0]^T$, and the second is for $x_R=[0,1]^T$. Analytically generate these Jacobian matrices by taking the indicated partial derivatives and evaluating the partials at the reference conditions. Specifically find

$$\text{Jacobian = a matrix} = \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}} = \begin{bmatrix} \dfrac{\partial \mathbf{g_1}}{\partial \mathbf{x_1}} & \dfrac{\partial \mathbf{g_1}}{\partial \mathbf{x_2}} \\ \dfrac{\partial \mathbf{g_2}}{\partial \mathbf{x_1}} & \dfrac{\partial \mathbf{g_2}}{\partial \mathbf{x_2}} \end{bmatrix}$$

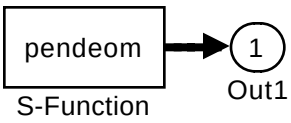
where g_1, g_2, x_1 and x_2 were defined earlier for the pendulum. Then evaluate the Jacobian at for $x_R=[0,0]^T$, and at $x_R=[0,1]^T$. Compare these two results to the a matrices computed by the script PendAnal.m.

Simulink Block Diagram to Analyze the Nonlinear Behavior of the Simple Pendulum



With corrected damping term (1/12/01)

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.

With corrected damping term (1/12/01)

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.

```
function [sys,x0,str,ts] =
pendeom(t,x,u,flag,g,l,HalfRhoSCd,mass,theta0,thetadot0)
%   S-file for pendulum (with corrected damping term, 1/12/01)
%   This is an S-file subsystem that models a simple pendulum.
%   g is the acceleration of gravity.
%   l is the length of the pendulum.
%   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.
% Point about which to linearize
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 ten seconds')
P10=10/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
t=0:.1:10';
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=['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]
[a,b,c,d]=linmod('pendulum',xR,[])
b=[0;0];
c=[1,0;0,1];
d=[0;0];
```

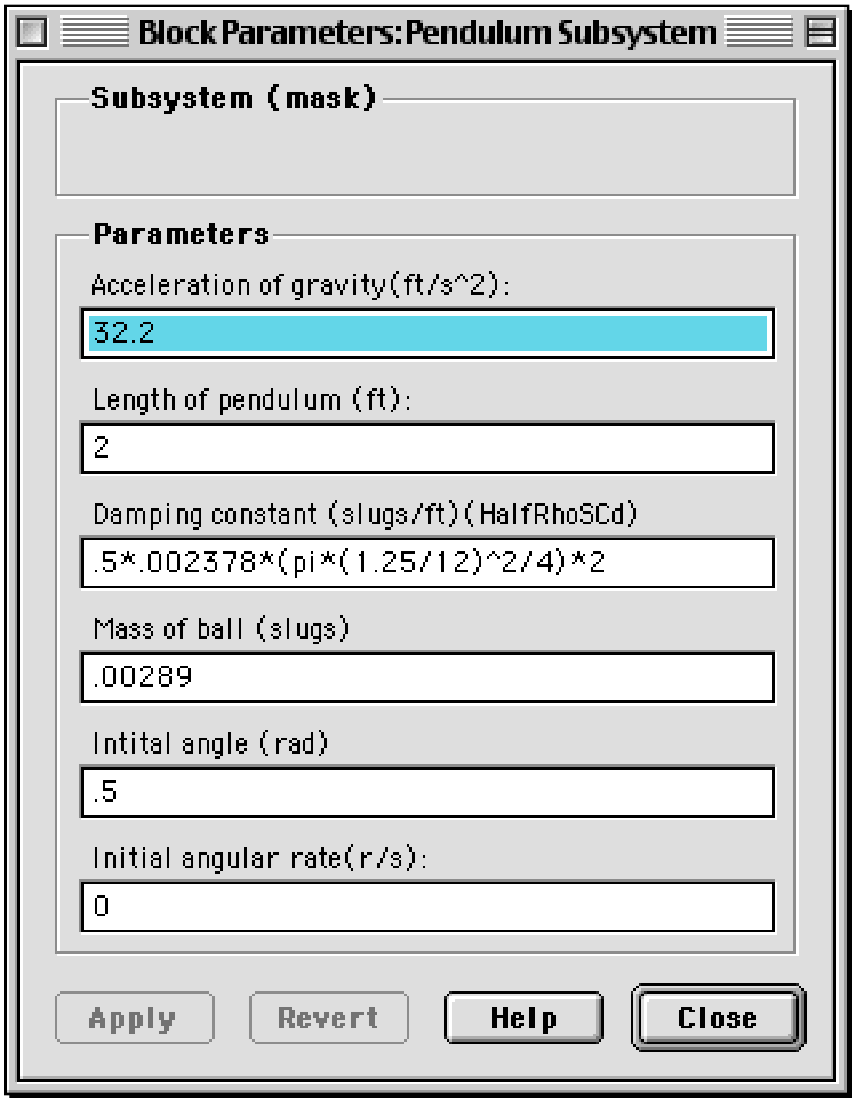
```
[Wn,Z]=damp(a)
PSYS=ss(a,b,c,d);
disp('Initial condition for linear simulation')
x0=[.5;0] %Use this initial condition for linear simulation
t=0:.1:10';
u=zeros(size(t));
[y,t]=lsim(PSYS,u,t,x0);
figure(2)
str1=['For Pendulum linearized about theta=
',num2str(xR(1)),', thetadot= ',num2str(xR(2))];
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')
```

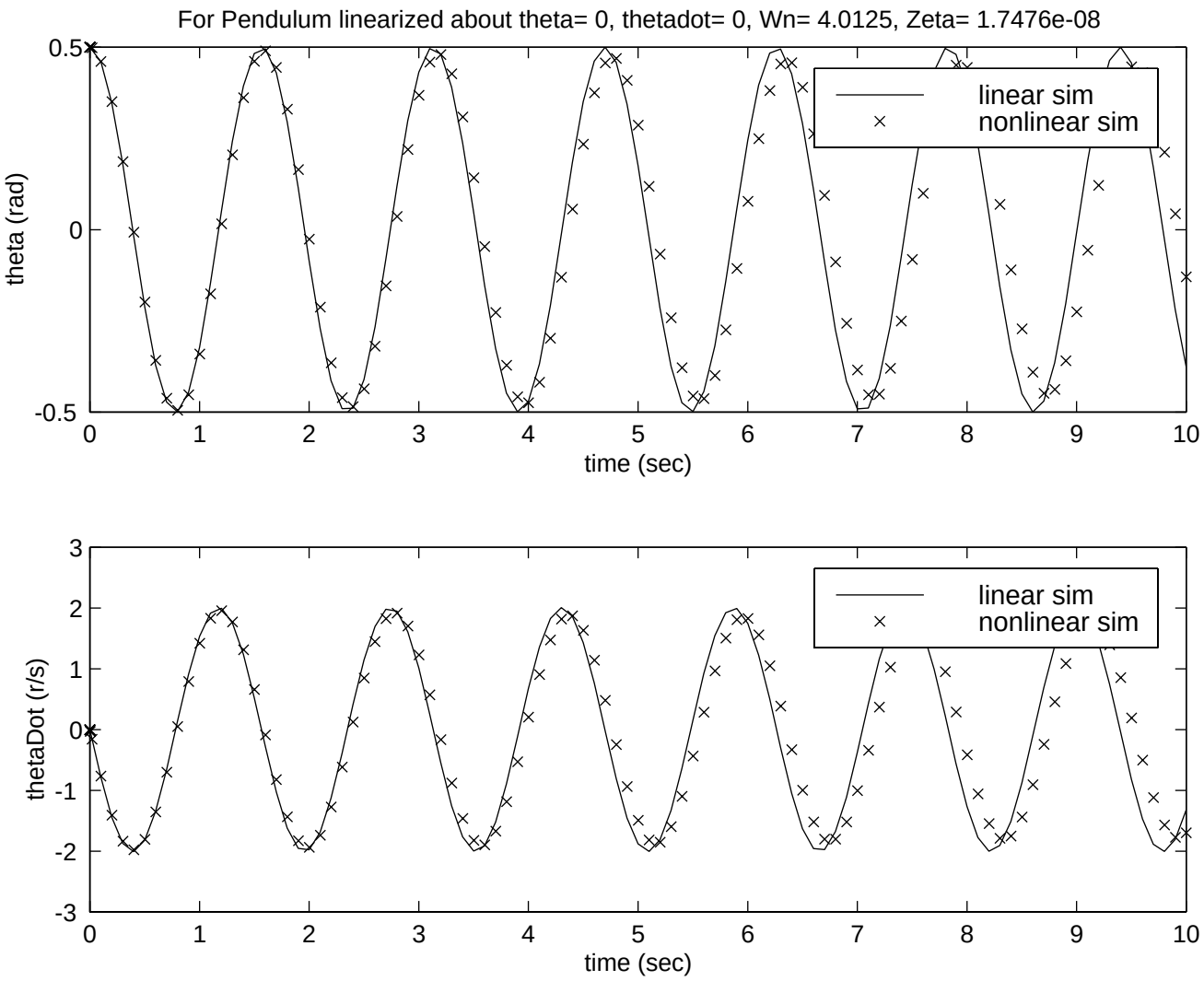
Output from the above script

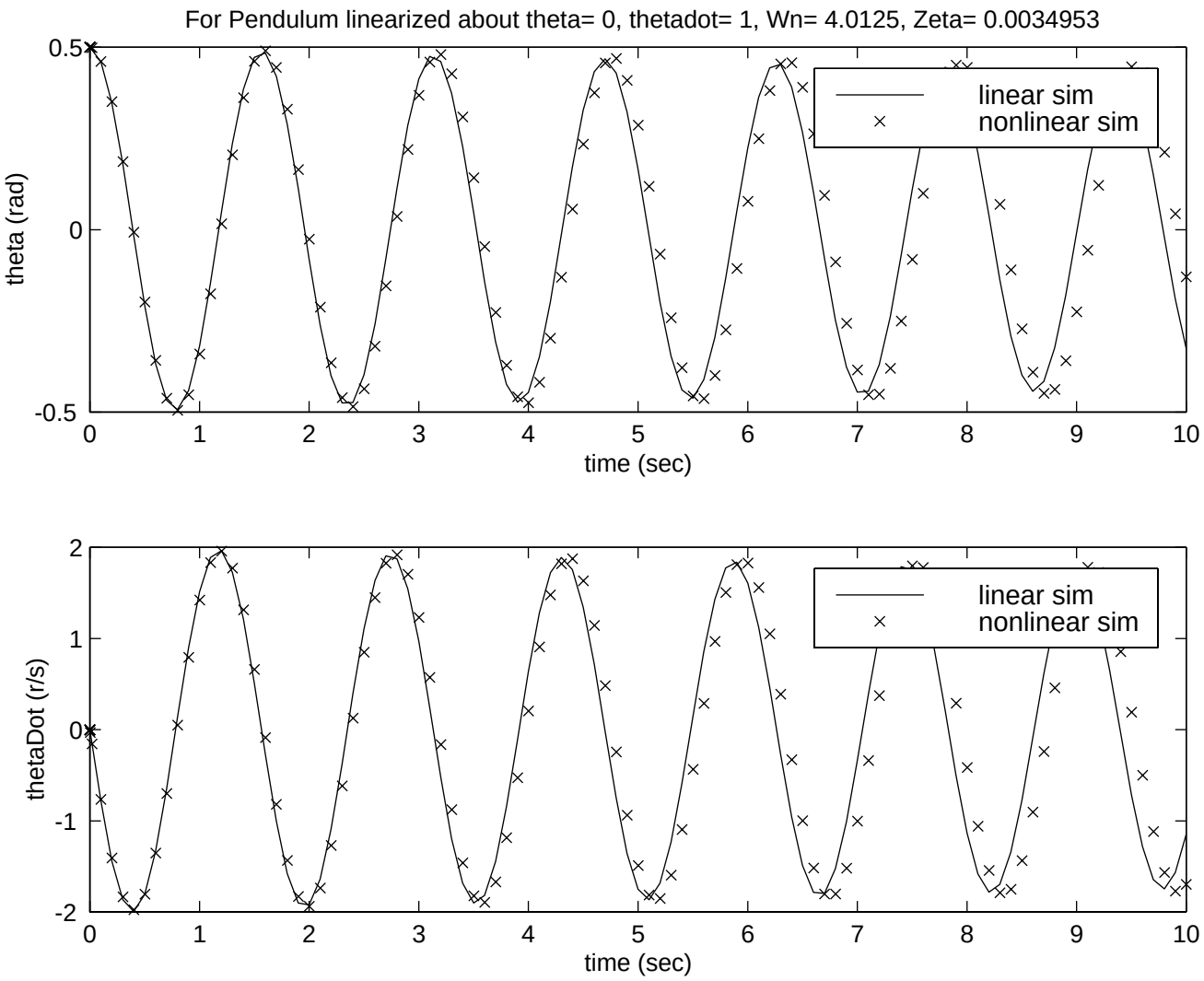
```
Start here

Find trim condition
X =
     -1.20915443998539e-14
     -3.50494170623093e-26
U =
   Empty matrix: 0-by-1
Y =
   Empty matrix: 0-by-1
DX =
     -3.50494170623093e-26
      1.94673864837647e-13
Reference point about which to linearize
xR =
```

```
     0
     0
a =
                         0                         1
         -16.0999999997317       -1.402463019451e-07
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 =
          4.01248052951434
          4.01248052951434
Z =
      1.74762595506195e-08
      1.74762595506195e-08
period =
            1.565910478808
Periods in ten seconds
P10 =
          6.38606110332193

a =
                        x1           x2
           x1            0            1
           x2        -16.1  -1.4025e-07


b =
                        u1
           x1            0
           x2            0
```

```
c =
                        x1            x2
           y1            1             0
           y2            0             1

d =
                        u1
           y1            0
           y2            0

Continuous-time system.
Initial condition for linear simulation
x0 =
                       0.5
                         0

New reference point

xR =
     0
     1
a =
                         0          1.00000000000047
         -16.0999999997316       -0.0280494007755387
b =
   Empty matrix: 2-by-0
c =
   Empty matrix: 0-by-2
d =
     []
Wn =
          4.01248052951528
          4.01248052951528
Z =
       0.00349526939373441
       0.00349526939373441
Initial condition for linear simulation
x0 =
                       0.5
                         0
»
```


Block Parameters: Pendulum Subsystem
Subsystem (mask)
Parameters
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
Intital angle (rad)
.5
Initial angular rate(r/s):
0
Apply
Revert
Help
Close


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


For Pendulum linearized about theta= 0, thetadot= 1, Wn= 4.0125, Zeta= 0.0034953
theta (rad)
time (sec)
linear sim
nonlinear sim
thetaDot (r/s)
time (sec)
linear sim
nonlinear sim