Nonlinear Model

$$m y'' = mg + F_s + F_d$$

where

$$F_s = -k(y + \varepsilon y^3)$$ restoring spring force

$$F_d = -c\left[y' + 0.05(y')^2 \frac{y'}{|y'|}\right]$$ damp force

↖ used to give proper sign

y = vertical deviation of lander mass relative to the point when the lander feet just touch the surface (no compression)

ICs $y(0) = 0$ and $y'(0) = v_0$ lander speed at touchdown

State Form

let $x_1 = y$

$x_2 = y'$

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -\frac{k}{m}(x_1 + \varepsilon x_1^3) - \frac{c}{m}\left[x_2 + 0.05 x_2^2 \frac{x_2}{|x_2|}\right] + g$$

∴ in matrix form

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{\varepsilon k}{m} x_1^3 - \frac{0.05c}{m} x_2^2 \frac{x_2}{|x_2|} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

where $u = g$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\,u$$

← output is $x_1(t)$ = lander deviation

Equilibrium Point

set derivative of state vector to zero...

$$\frac{dx_1}{dt} = 0 = x_{2e} \rightarrow \boxed{y'_{equil} = 0}$$ (ok)

$$\frac{dx_2}{dt} = 0 = -\frac{k}{m}(x_{1e} + \varepsilon x_{1e}^3) + g$$

$$\boxed{\frac{\varepsilon k}{m} x_{1e}^3 + \frac{k}{m} x_{1e} - g = 0}$$

solve this for x_{1e} see matlab

↳ solution gives $\boxed{y_e = x_{1e}}$ ↗

Spring and Damping Forces (Linear vs Nonlinear)

Spring force

$$F_s^{NL}(y) = -k(y + \varepsilon y^3)$$ nonlinear

$$F_s^{L}(y) \simeq F_s(y_e) + \frac{d}{dy} F_s(y)\Big|_{y=y_e} (y - y_e)$$ linear

$$\boxed{F_s^{L}(y) \simeq F_s^{NL}(y_e) - k(1 + 24 y_e^2)(y - y_e)}$$

damping treated in similar fashion see matlab

see matlab

Comments

From Matlab code and plot, the linear spring model is simply a straight line that is tangent to the nonlinear curve at the linearization pt.

In the vicinity of $y = y_e$, the linear and nonlinear models are very similar. However, as one deviates from the reference linearization point the difference grow larger.

note that the damping force is treated in a similar fashion

Linearization

let $x = x_r + \delta x$

$u = u_r + \delta u$

where $x_r = x_e$

↖ equilibrium pt. once lander has come to rest

from the Class Notes

$$\frac{d}{dt}\delta x = \left(A + J_x\big|_{x_r,u_r}\right)\delta x + \left(B + J_u\big|_{x_r,u_r}\right)\delta u + G$$

$J_u \to 0$ since t not a function of u; $J_u \to 0$ since gravity does not change; $G \to 0$ since x_r, u_r satisfies state eqn

∴ linear model is unforced and given by

$$\frac{d}{dt}\delta x = \left(A + J_x\big|_{x_r,u_r}\right)\delta x$$

where

$$J_x\big|_{x_r,u_r} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}\Bigg|_{x_r,u_r} = \begin{bmatrix} 0 & \\ -\frac{24k}{m}x_{1e}^2 & -\frac{0.10}{m}x_{2e} \end{bmatrix} \quad \frac{y_{2e}}{|x_{2e}|}$$

but $x_{1e} = y_e$ and $x_{2e} = 0$ ← at rest

$$\therefore \frac{d}{dt}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} - \frac{24k}{m}y_e^2 & -\frac{c}{m} \end{bmatrix}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\delta u$$

$$\delta y = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} + [0]\,\delta u \quad \leftarrow \delta u = 0$$

↑ initial conditions

$$x_0 = \begin{bmatrix} 0 \\ v_0 \end{bmatrix} \quad \therefore \quad \delta x_0 = x_0 - x_r = \begin{bmatrix} 0 \\ v_0 \end{bmatrix} - \begin{bmatrix} y_e \\ 0 \end{bmatrix}$$

$$\therefore \delta x_0 = \begin{bmatrix} -y_e \\ v_0 \end{bmatrix}$$

* for nonlinear case

for linear case

- OK, we now have both linear and nonlinear models and we can do the desired simulations.
- We also have a visualization of the spring and dampening forces to help explain the observations

→ see lander.m and lander_eqns.m + results file

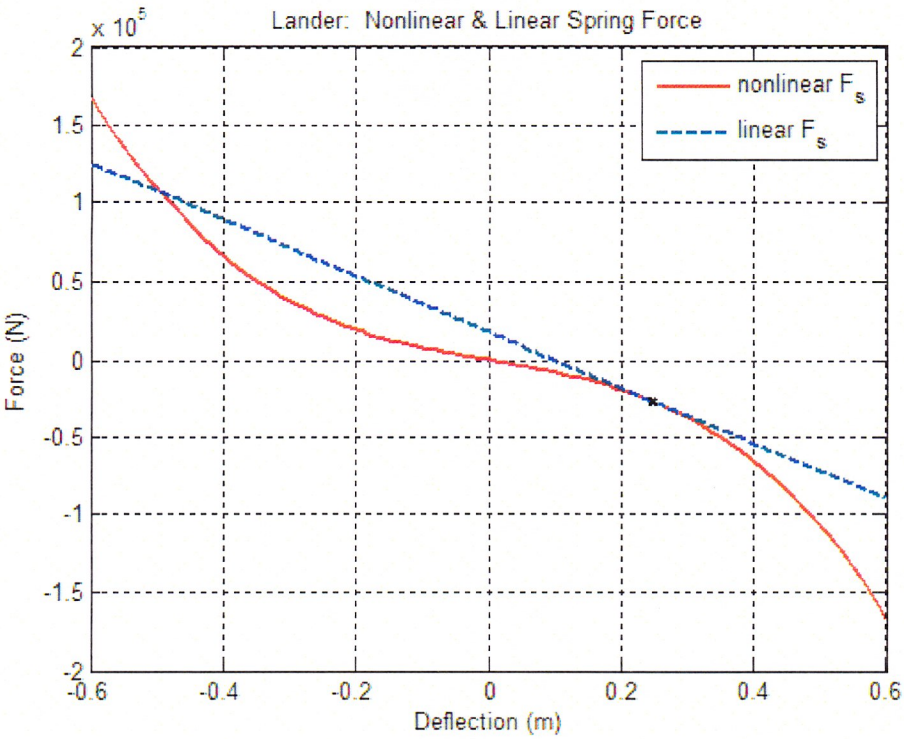

Fig. 1 Linear vs nonlinear spring force.

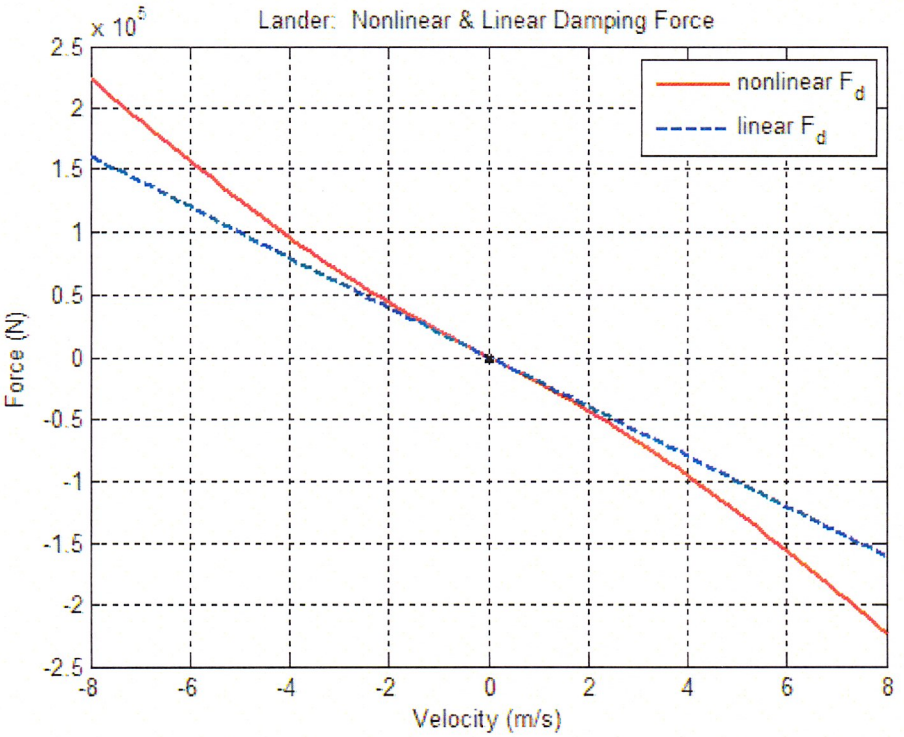

Fig. 2 Linear vs nonlinear damping (friction) force.

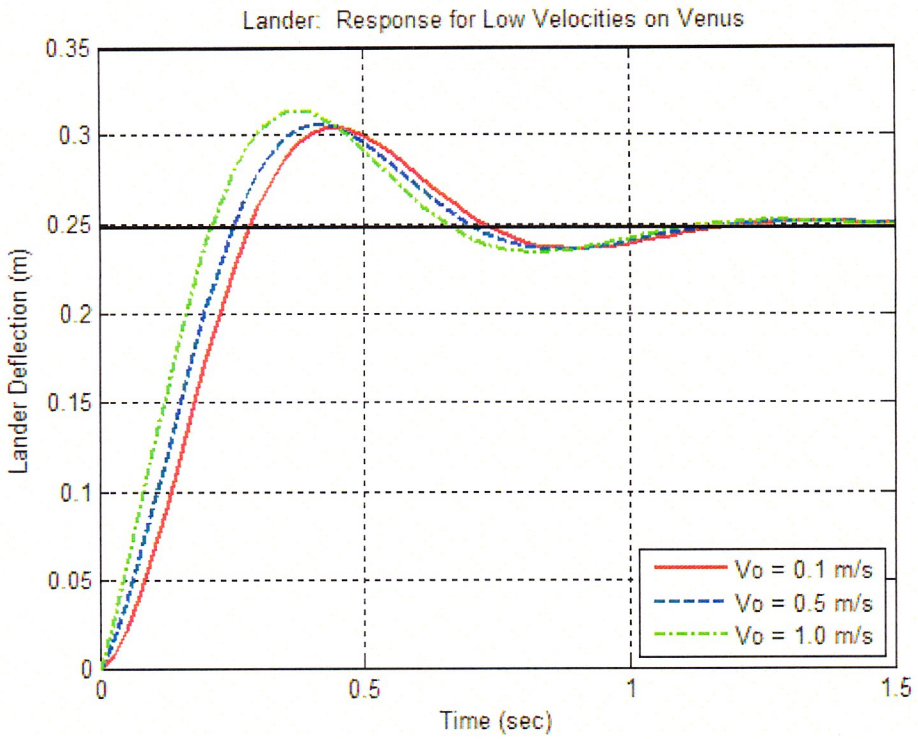

Fig. 3 Response of system for low initial speeds.

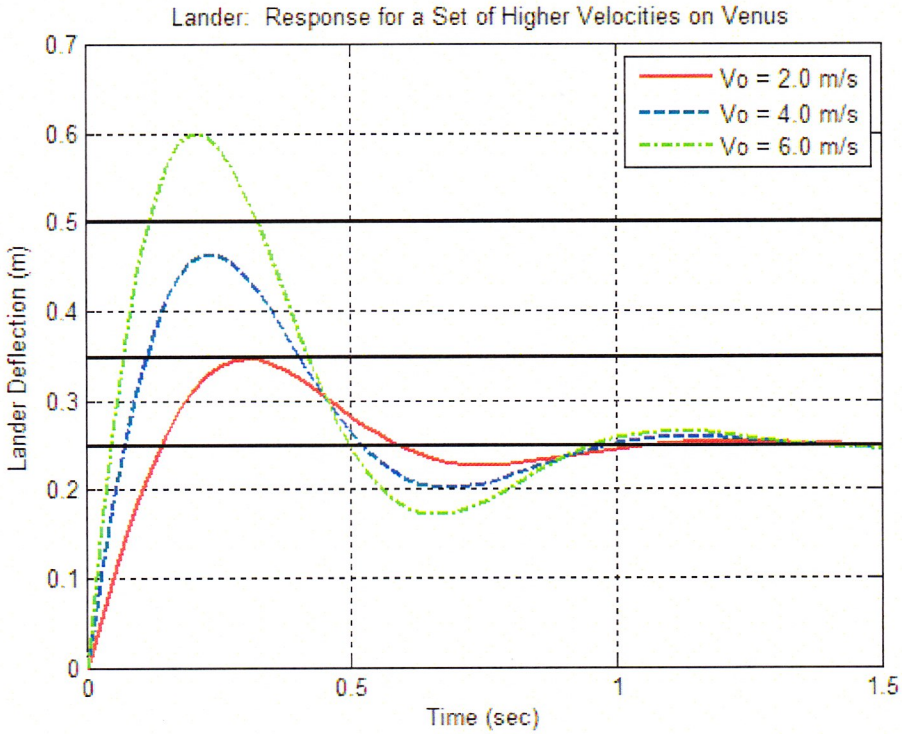

Fig. 4 Response of system for higher initial speeds.

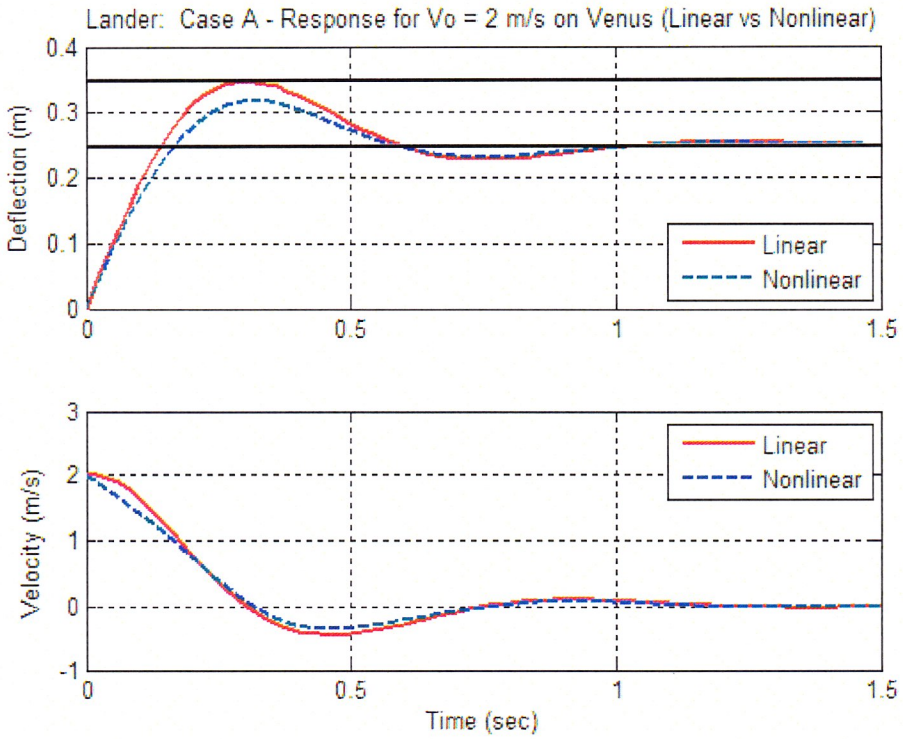

Fig. 5 Linear vs. nonlinear responses for near optimal landing speed.

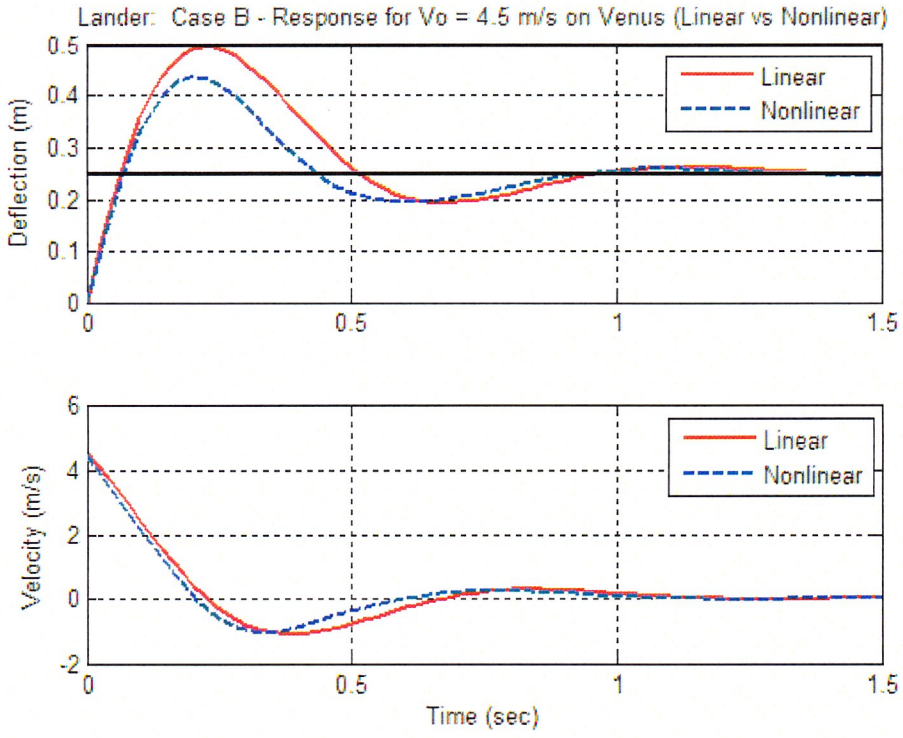

Fig. 6 Linear vs. nonlinear responses for near maximum landing speed.

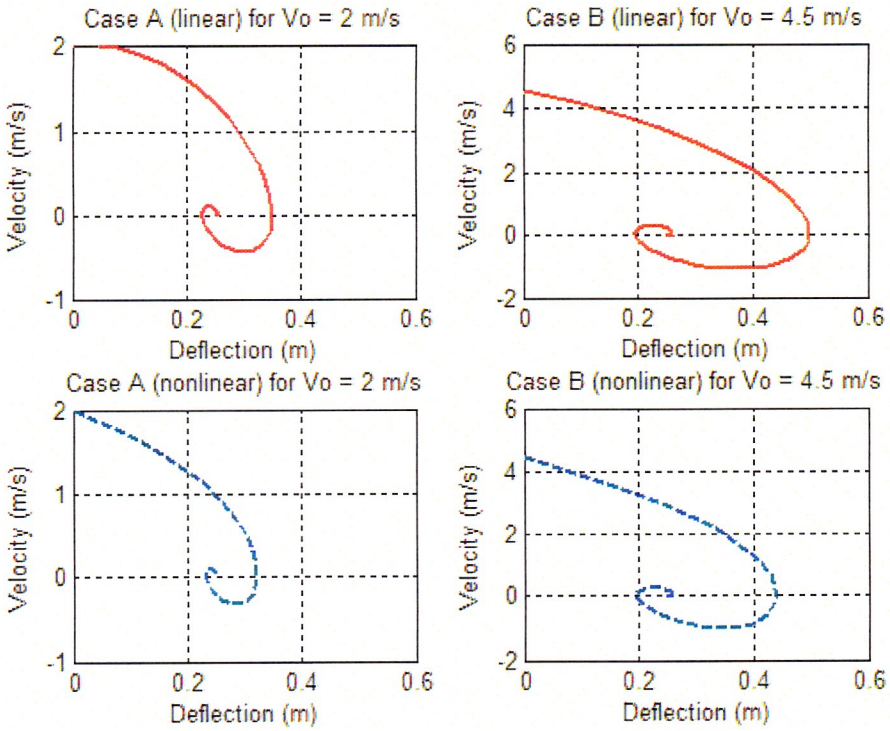

Fig. 7 Linear vs. nonlinear response profiles in the phase plane view.

Some Comments:

1. The plots of the friction force versus deflection (Fig. 1) and damping force versus velocity (Fig. 2) are critical to understanding what is happening in this application. Clearly the real forces are quite nonlinear, especially the spring force for deflections beyond ±0.2 m. However, for relatively small deflections and reasonable velocities, the linear approximation to these forces is not too bad (note that the linear forces represent tangents to the nonlinear force curves at the linearization point).

   However, since the maximum deflection allowed is 0.5 m, then the linear spring force model should give a reasonable estimate over the full range of operation. Also, since the maximum initial velocity allowed is under 5 m/s (see subsequent simulations), then this force is also modeled adequately with a simple linear approximation around equilibrium (with V = 0). Thus, based on the observations from Figs. 1 and 2, we should expect a reasonable comparison between the linear and nonlinear simulations. Also, since the linear approximations tend to under predict the magnitude of both restoring forces for large positive deviations, the linear model should over predict the actual observed lander deflection -- giving a conservative prediction of the maximum overshoot and the maximum V_o allowed before the legs buckle.

2. The above discussion (Comment #1) sets the stage for the analysis of the actual simulations. For example, it shows that the linear model can be used to study a range of initial velocities with the confidence that the results will be fairly reasonable, and on the conservative side --

meaning that the simulations should be fairly close to those expected for the real (nonlinear) system, yet the results for the maximum allowed velocity will be a conservative upper limit on the real system.

In particular, Figs. 3 and 4 show the expected behavior (from the linear model) for a range of initial velocities. Here we see the expected overshoot of the lander relative to its equilibrium value ($y_e \approx 0.25$ m) but, in all cases, the system comes to rest with minimal bouncing in well under 2 seconds. Also, we see that the maximum overshoot criterion of 0.1 m relative to y_e will be satisfied with a V_o of just about 2 m/s and that the absolute maximum initial velocity before failure is around 4.5 m/s with the linear model (i.e. before the maximum deflection exceeds its design limit of 0.5 m).

3. Now, with the results of the linear model for guidance, a formal comparison of the linear and nonlinear simulations are done at near optimum velocity ($V_o = 2$ m/s) and at maximum initial velocity ($V_o = 4.5$ m/s). In both cases (see Figs. 5 and 6), the linear and nonlinear models show similar behavior and, as expected, the lander deflection for the nonlinear (real) model was less than predicted by the linear model -- because of the under prediction of the linearized restoring spring and damping forces as discussed above. Overall, however, the linear model (in this case) is quite reasonable, with a slight (conservative) over prediction of the magnitude of motion of the lander as it settles on the surface of Venus. However, with increased V_o, the differences are clearly larger, since the system is further removed from the linearization point. This type of behavior is typical, since the farther one goes from the linearization point, the greater the difference in the approximate (linear) and actual (nonlinear) forces in this system. However, since this system, by design, should always stay reasonably close to the equilibrium point, both models give similar dynamic behavior.

4. Finally, in Fig. 7, we present the phase plane plots -- velocity vs. deflection -- for both the linear and nonlinear models at the two initial velocities analyzed above. These again show similar behavior for the linear and nonlinear models, only with a different view this time. Here the focus is on V vs. y, with time removed from the visualization. The simulation starts at some nonzero initial velocity, but zero deflection, and they end at the equilibrium position, y_e, and zero velocity (i.e. at rest on Venus). The phase plane plots show a spiral inward to a focal point (equilibrium point) as expected for a stable system. This is just a different view of the same dynamics as shown/discussed in previous plots.

Code Printout:

```
>> lander
       Summary Information for Venus Lander Problem

  Equilibrium deflection (m)      :   0.2487
  Equilibrium velocity  (m/s)     :   0.0000
  Maximum deflection allowed (m)):   0.5000
  Linearized system matrices (around equilibrium deflection)

a =
            x1       x2
   x1        0        1
   x2   -59.62   -6.667

b =
        u1
   x1   0
   x2   1

c =
        x1  x2
   y1   1   0

d =
        u1
   y1   0

Continuous-time model.
Case A - Input Vo (m/s) for max overshoot of 0.1 m for landing on Venus: 2
Case B - Input Vo (m/s) for max deflection without buckling on Venus: 4.5
>>
```

```
%
%   LANDER.M    Earth/Lunar/Martian/Venetian Lander Problem
%
%   File prepared by J. R. White, UMass-Lowell (Feb. 2014)

    clear all,   close all,   nfig = 0;
%
%   base data for problem
    m = 3000;         % vehicle mass (kg)
    c = 20000;        % damping factor (N-s/m)
    k = 72000;        % spring constant (N/m)
    mdefl = 0.5;      % max compression of lander legs (m)
    gg = [9.81 1.60 3.72 8.92];  % gravity on earth, moon, mars, venus (m/s^2)
    titl = ['Earth'; 'Moon '; 'Mars '; 'Venus'];
    kopt = menu('Choose Landing Site', ...
                '         Earth          ', ...
                '          Moon          ', ...
                '          Mars          ', ...
                '         Venus          ');
    g = gg(kopt);     ct = titl(kopt,:);
%
%   find equilibrium deflection
    p = [8*k/m 0 k/m -g];   r = roots(p);   ydote = 0;
%
%   choose real root as desired equilibrium point
    for i = 1:3,  if imag(r(i)) == 0,  ye = r(i);   end,  end
    fprintf('       Summary Information for %s Lander Problem \n\n',ct);
    fprintf('  Equilibrium deflection (m)     : %8.4f \n',ye);
    fprintf('  Equilibrium velocity  (m/s)    : %8.4f \n',ydote);
    fprintf('  Maximum deflection allowed (m)): %8.4f \n',mdefl);
%
%   plot spring force versus deflection (nonlinear and linear)
    fsye = -k*(ye + 8*ye^3);                              % spring force at equil pt
    y = -0.6:.01:0.6;   fsnl = -k*(y + 8*y.^3);           % nonlinear spring force
    dy =  y-ye;  fsl = fsye - k*(1 + 24*ye^2)*dy;         % linear spring force
    nfig = nfig+1;   figure(nfig)
    plot(y,fsnl,'r-',y,fsl,'b--',ye,fsye,'kx','LineWidth',2),grid
    v = axis;  v(1) = -0.600001;   v(2) = 0.600001;   axis(v)
    title('Lander:  Nonlinear & Linear Spring Force')
    xlabel('Deflection (m)'),ylabel('Force (N)')
    legend('nonlinear F_s','linear F_s')
%
%   plot damping force versus velocity (nonlinear and linear)
    fdydote = -c*(ydote + 0.05*ydote.^2*sign(ydote));     % damping force at equil pt
    ydot = -8:0.2:8;                                      % nonlinear damping force
    fdnl = -c*(ydot + 0.05*ydot.^2.*sign(ydot));
    dydot = ydot-ydote;                                   % linear damping force
    fdl = fdydote - c*(1 + 0.1*ydote.*sign(ydote))*dydot;
    nfig = nfig+1;   figure(nfig)
    plot(ydot,fdnl,'r-',dydot,fdl,'b--',ydote,fdydote,'kx','LineWidth',2),grid
    v = axis;  v(1) = -8;   v(2) = 8;   axis(v)
    title('Lander:  Nonlinear & Linear Damping Force')
```

```
  xlabel('Velocity (m/s)'),ylabel('Force (N)')
  legend('nonlinear F_d','linear F_d')

%
%   create LTI system
  fprintf('  Linearized system matrices (around equilibrium deflection) \n');
  A = [0  1;-(k/m)*(1+24*ye^2) -c/m];   B = [0 1]';   C = [1 0];   D = 0;
  sys1 = ss(A,B,C,D)
%
%  simulate time domain response for several low velocities (m/s)
  t = linspace(0,1.5,61);
  zze = ye*ones(size(t));  zzb = zze+0.1;  zzm = mdefl*ones(size(t));
  zz = zeros(size(t));    u = zz';
  v1 = 0.1;   v2 = 0.5;   v3 = 1.0;
  dy1 = lsim(sys1,u,t,[-ye v1]');
  dy2 = lsim(sys1,u,t,[-ye v2]');
  dy3 = lsim(sys1,u,t,[-ye v3]');
  y1 = ye + dy1;  y2 = ye + dy2;   y3 = ye + dy3;
  nfig = nfig+1;   figure(nfig)
  plot(t,y1,'r-',t,y2,'b--',t,y3,'g-.',t,zze,'k-',t,zzb,'k-','LineWidth',2),grid
  title(['Lander:  Response for Low Velocities on ',ct])
  xlabel('Time (sec)'),ylabel('Lander Deflection (m)')
  legend('Vo = 0.1 m/s','Vo = 0.5 m/s','Vo = 1.0 m/s','Location','SouthEast')
%
%  simulate time domain response for a set of higher velocities (m/s)
  v4 = 2.0;    v5 = 4.0;    v6 = 6.0;
  dy4 = lsim(sys1,u,t,[-ye v4]');
  dy5 = lsim(sys1,u,t,[-ye v5]');
  dy6 = lsim(sys1,u,t,[-ye v6]');
  y4 = ye + dy4;  y5 = ye + dy5;   y6 = ye + dy6;
  nfig = nfig+1;   figure(nfig)
  plot(t,y4,'r-',t,y5,'b--',t,y6,'g-.',t,zze,'k-',t,zzb,'k-',t,zzm,'k-', ...
       'LineWidth',2),grid
  title(['Lander:  Response for a Set of Higher Velocities on ',ct])
  xlabel('Time (sec)'),ylabel('Lander Deflection (m)')
  legend('Vo = 2.0 m/s','Vo = 4.0 m/s','Vo = 6.0 m/s')
%
%  modify C and D matrices so that the velocity is also a response of interest
%  --> first response is deviation from equil position
%  --> second response is velocity (positive is downward)
  C = eye(2);   D = [0 0]';   sys2 = ss(A,B,C,D);
%
%  Further analysis of nominal design (max overshoot of 0.1 m)
%  Also let's compare the nonlinear vs linear model
%
%  Case A  request input initial velocity from user
  va0 = input(...
  ['Case A - Input Vo (m/s) for max overshoot of 0.1 m for landing on ',ct,': ']);
%
%  Linear Case
  ta0 = 0;    taf = 1.5;   tal = ta0:0.02:taf;
  zza = zeros(size(tal));      ua = zza';
  dyal = lsim(sys2,ua,tal,[-ye va0]');   yal = ye + dyal(:,1);
```

```
%
  Nonlinear Case
    ftx = @(t,x) lander_eqns(t,x,m,k,c,g);
    ya0 = [0 va0]';
    [tanl,yanl] = ode45(ftx,[ta0,taf],ya0);
    zzae = ye*ones(size(tanl));  zzao = zzae + 0.1;
%
%   plot Linear and Nonlinear results for Case A
    nfig = nfig+1;   figure(nfig)
    subplot(2,1,1)
    plot(tal,yal,'r-',tanl,yanl(:,1),'b--',tanl,zzae,'k-', ...
         tanl,zzao,'k-','LineWidth',2),grid
    title(['Lander:  Case A - Response for Vo = ',num2str(va0),' m/s on ', ...
          ct,' (Linear vs Nonlinear)'])
    ylabel('Deflection (m)')
    legend('Linear','Nonlinear','Location','SouthEast')
    subplot(2,1,2),plot(tal,dyal(:,2),'r-',tanl,yanl(:,2),'b--','LineWidth',2),grid
    xlabel('Time (sec)'),ylabel('Velocity (m/s)')
    legend('Linear','Nonlinear','Location','NorthEast')
%
%   Further analysis of max velocity case (don't let legs buckle)
%   Also let's compare the nonlinear vs linear model
%
%   Case B - request input initial velocity from user
    vb0 = input(...
    ['Case B - Input Vo (m/s) for max deflection without buckling on ',ct,': ']);
%
%   Linear Case
    tb0 = 0;    tbf = 1.5;   tbl = tb0:.02:tbf;
    zzb = zeros(size(tbl));      ub = zzb';
    dybl = lsim(sys2,ub,tbl,[-ye vb0]');   ybl = ye + dybl(:,1);
%
%   Nonlinear Case
    yb0 = [0 vb0]';
    [tbnl,ybnl] = ode23(ftx,[tb0,tbf],yb0);
    zzbe = ye*ones(size(tbnl));   zzbm = mdefl*ones(size(tbnl));
%
%   plot Linear and Nonlinear cases
    nfig = nfig+1;   figure(nfig)
    subplot(2,1,1),plot(tbl,ybl,'r-',tbnl,ybnl(:,1),'b--',tbnl,zzbe,'k-', ...
         tbnl,zzbm,'k-','LineWidth',2),grid
    rr = axis;  rr(4) = mdefl+0.005;  axis(rr);
    title(['Lander:  Case B - Response for Vo = ',num2str(vb0),' m/s on ', ...
           ct,' (Linear vs Nonlinear)'])
    ylabel('Deflection (m)')
    legend('Linear','Nonlinear')
    subplot(2,1,2),plot(tbl,dybl(:,2),'r-',tbnl,ybnl(:,2),'b--','LineWidth',2),grid
    xlabel('Time (sec)'),ylabel('Velocity (m/s)')
    legend('Linear','Nonlinear')
%
%   let's try some Phase Plane plots (VELOCITY vs POSITION)
    nfig = nfig+1;   figure(nfig)
```

```
    subplot(2,2,1), plot(yal,dyal(:,2),'r-','LineWidth',2),grid
    title(['Case A (linear) for Vo = ',num2str(va0),' m/s'])
    xlabel('Deflection (m)'),ylabel('Velocity (m/s)')
    axis([0 .6 -1 2])
    subplot(2,2,2), plot(ybl,dybl(:,2),'r-','LineWidth',2),grid
    title(['Case B (linear) for Vo = ',num2str(vb0),' m/s'])
    xlabel('Deflection (m)'),ylabel('Velocity (m/s)')
    axis([0 .6 -2 6])
    subplot(2,2,3), plot(yanl(:,1),yanl(:,2),'b--','LineWidth',2),grid
    title(['Case A (nonlinear) for Vo = ',num2str(va0),' m/s'])
    xlabel('Deflection (m)'),ylabel('Velocity (m/s)')
    axis([0 .6 -1 2])
    subplot(2,2,4), plot(ybnl(:,1),ybnl(:,2),'b--','LineWidth',2),grid
    title(['Case B (nonlinear) for Vo = ',num2str(vb0),' m/s'])
    xlabel('Deflection (m)'),ylabel('Velocity (m/s)')
    axis([0 .6 -2 6])
%
% end of simulation
```

```
%
% LANDER_EQNS.M  File for nonlinear simulation of the suspension
%                system of a landing craft
%
function xdot = lander_eqns(t,x,m,k,c,g)
xdot = zeros(length(x),1);
xdot(1) = x(2);
Fs = -k*x(1) - 8*k*x(1)^3;
Fd = -c*x(2) - 0.05*c*x(2)^2*sign(x(2));
xdot(2) = g + (Fs + Fd)/m;
```