```
                             Graphics and ODE Solvers
Let's use the ode solvers to set up some data to plot.
  We'll solve the Van der Pol equation
         y''-mu*(1-y^2)*y'+y=0    y(0)=2,  y'(0)=0
       (This oscillator has a nonlinear damping term.)
   1) By HAND: Rewrite it as a first order system of equations:
       y1'=y2
       y2'=mu*(1-y1^2)*y2-y1
   2) Create a Matlab function with inputs t and y and mu.
      It should return a COLUMN with the right hand side values in it.
      To identify y1 and y2 use vector indexing: y(1) and y(2)
      >> f=@(t,y,mu) [ y(2); mu*(1-y(1).^2).*y(2)-y(1) ];
   3) Solve the ODE using ode45 for 0<t<20 and y(0)=2, y'(0)=0.
      The ode45 suite of functions takes parameter values starting in
      the 5th input position. The 4th position is saved for options so
      we use an empty list in the 4th position and mu=2 in the 5th.
      >> ode45(f,[0 20],[2 0],[],2)
      Watch the plot appear of y1 and y2 as functions of time.
   4) Usually, we want to manipulate the data beyond just getting a plot.
      We can do this with the following command:
                               >> [t,y] = ode45(f,[0 20],[2 0],[],2);
    Now plot y1 versus t:      >> plot(t,y(:,1))
    And then plot y2 versus t: >> plot(t,y(:,2))
    To create both use:        >> plot(t,y(:,1),'og',t,y(:,2),'or')
  Notice that t is a vector of times. The variable y is a matrix with
  one variable for each column and time extending by row.
  So y(:,2) gives y2 over time.

Plotting:
   1)  Let's plot y1 versus y2 (The Phase Plane):  >> plot(y(:,1),y(:,2))
        This is a parametric plot (y1 and y2 are really functions of t).
        It is useful for seeing if the behavior is really periodic.
  Q: Is the solution of the Van der Pol equation with mu=2 periodic?
   2) Now let's put the plots in two figure windows next to each other:
     >> plot(y(:,1),y(:,2))
     >> figure(2)
     >> plot(t,y(:,1),'og',t,y(:,2),'or')
    You can "dock" these figures to the command window.
   3) Titles, labels and legends.  You can set them with these commands:
      >>  title('Van der Pol Solutions')
      >>  xlabel('y(1)')
      >>  ylabel('y(2)')
      >> legend('y(1)','y(2)')
        Now you can move the legend around if you wish with the mouse.
        To get rid of the legend, use:  >> legend off
    Misc:    Try >> grid on, >> grid off     >> box on,>> box off
   4) Multiple plots on one page:
      >> subplot(2,2,1)         % create 2x2 array of figures and use 1st
      >> plot(t,y(:,1),'og'
      >> subplot(2,2,3)         % subplot 3 is below subplot 1.
      >> plot(t,y(:,2),'or')
      >> subplot(2,2,[2 4])     % single plot in space of subplots 2 and 4
      >> plot(y(:,1),y(:,2))
     Now try the title command:  >> title('phase plane') %subplot title


More ODEs on back...
```

```
Stiff Problems:        Large mu makes the problem "Stiff"
  Try timing the solver as you increase the value of mu.
  time the solver:   >> tic; ode45(f,[0 20],[2 0],[],2); toc;
  increase mu:       >> tic; ode45(f,[0 20],[2 0],[],10); toc;
  increase mu again: >> tic; ode45(f,[0 20],[2 0],[],100); toc;
This is getting slow.  Notice the small timestep: Let's switch to a stiff solver.
  try with ode15s:   >> tic; ode15s(f,[0 20],[2 0],[],100); toc;
Notice the timestep now and compare times for calculation.
  increase the timespan:   >> tic; ode15s(f,[0 200],[2 0],[],100); toc;
  Time with mu=2:    >> tic;ode15s(f,[0 200],[2 0],[],2); toc;
  Time ode45-mu=2:   >> tic; ode45(f,[0 200],[2 0],[],2); toc;
  Time ode23-mu=2:   >> tic; ode23(f,[0 200],[2 0],[],2); toc;

Q: Which solver (of these three) is the best for mu=2?  For mu=100?

Options:
 There are many options for the ode solvers.
 Learn more from >> doc odeset
 1) Read it to find the default values for the relative error tolerance.
 2) Create an option object holding the relative tolerance to 1e-5.
                                      >> options = odeset('RelTol',1e-5);
    now use the options with ode45:   >> ode45(f,[0 200],[2 0],options,2);
    Increase the relative error tolerance until the timeplot changes visibly.
    Did it change height or did it change its timing?
 3) Set the RelTol and an initial guess for a timestep.
 >> options=odeset('RelTol',1e-6,'InitialStep',10.0)
 >> tic; ode45(f,[0 200],[2 0],options,2); toc
Even with a huge first timestep, the adaptive method works--and doesn't take too long.

Locating Events:
 You can have the ode solver identify features of the solution (an "Event").
 We can find the times of each maximum of the solution y(t).
 Create a function (USING A FUNCTION FILE) which returns three values:
   1) a value that is zero at the event (for a maximum that could be y'(t))
   2) a flag that is 1 if we should stop solving and 0 to continue solving
   3) a direction flag: -1/1/0 for decreasing/increasing/either event values.
 We want a maximum so y'(t) should be decreasing through 0 so we use -1.
 The file can look like:
 function [value, stop, direction] = event(t, y, mu)
   value=y(2);
   stop=0;
   direction=-1;
 Save it as event.m and then create an options setting to turn on event finding.
 >> options = odeset(options, 'Events', @event)  % updates previous options
 >> [t,y,te,ye]=ode45(f,[0 200],[2 0],options,2);
 WARNING: if you try to use an anonymous function for events, ode45 may give an
          error about input or output variables.
 Now te are the times for the events, ye are the y values at the events.
 >> plot(t,y(:,1),'-b', te, ye(:,1),'or') % curve in blue, events are red
 >> Period=te(end)-te(end-1);
 >> fprintf('The period of oscillation is %8.4f\n',Period)

 Figure out how to find the times for both minimums and maximums. Plot
 the curve with a red circle on both min and max values.

If you use "doc ode45" you will see about 1/3 of the way down a table of all the
ODE solvers and tips on when to use each one.
```