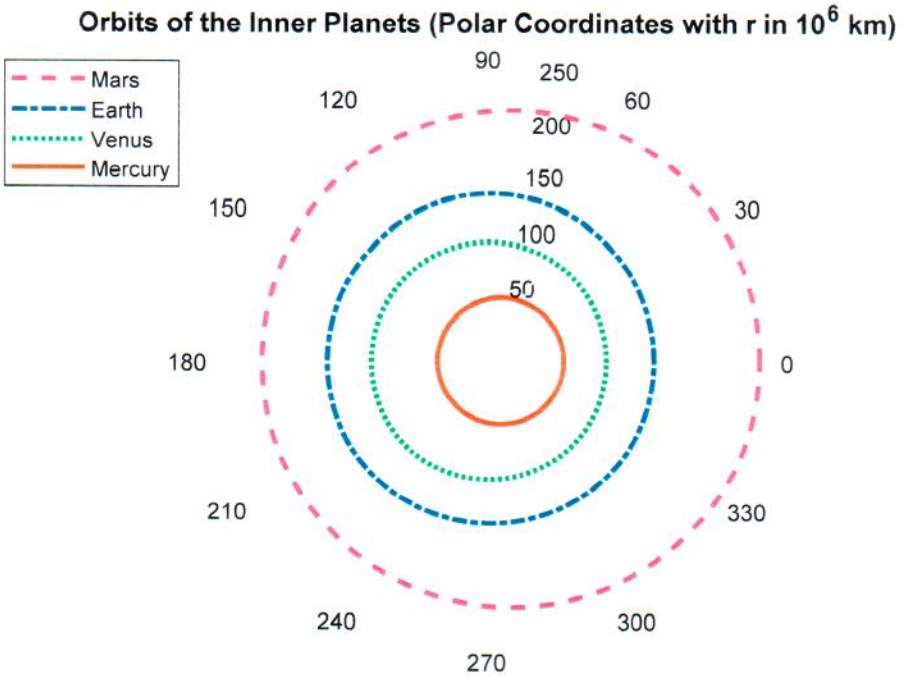
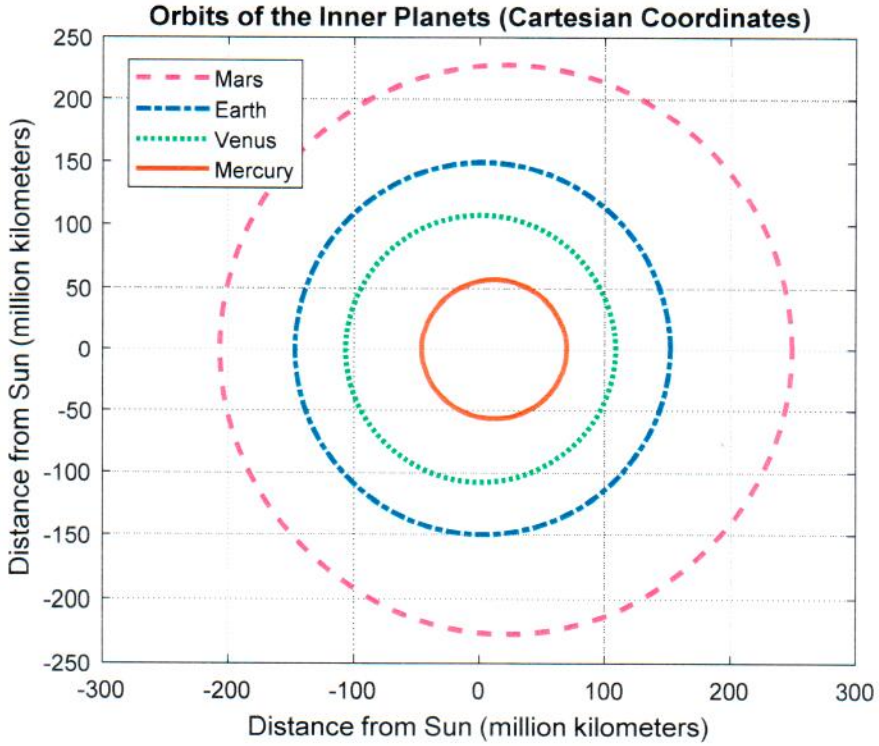
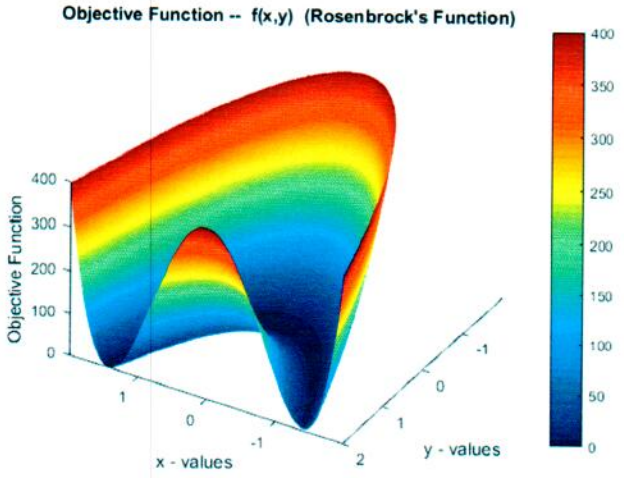

Orbits of the Inner Planets (Polar Coordinates with r in 10^6 km)
Mars
Earth
Venus
Mercury
90
250
120
60
200
150
150
30
100
50
180
0
210
330
240
300
270

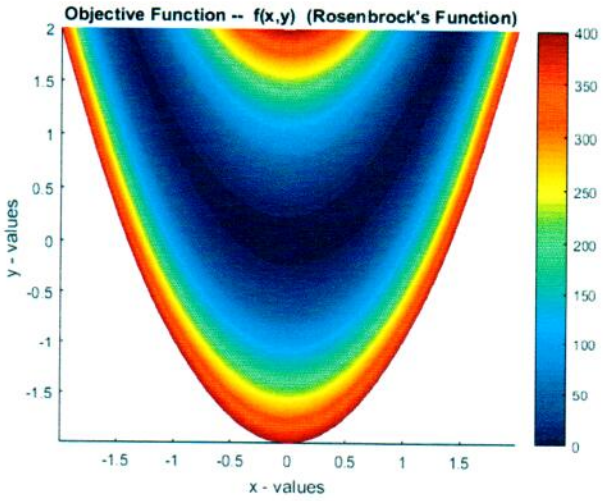

Orbits of the Inner Planets (Cartesian Coordinates)
Mars
Earth
Venus
Mercury
Distance from Sun (million kilometers)
250
200
150
100
50
0
-50
-100
-150
-200
-250
-300
-200
-100
0
100
200
300
Distance from Sun (million kilometers)

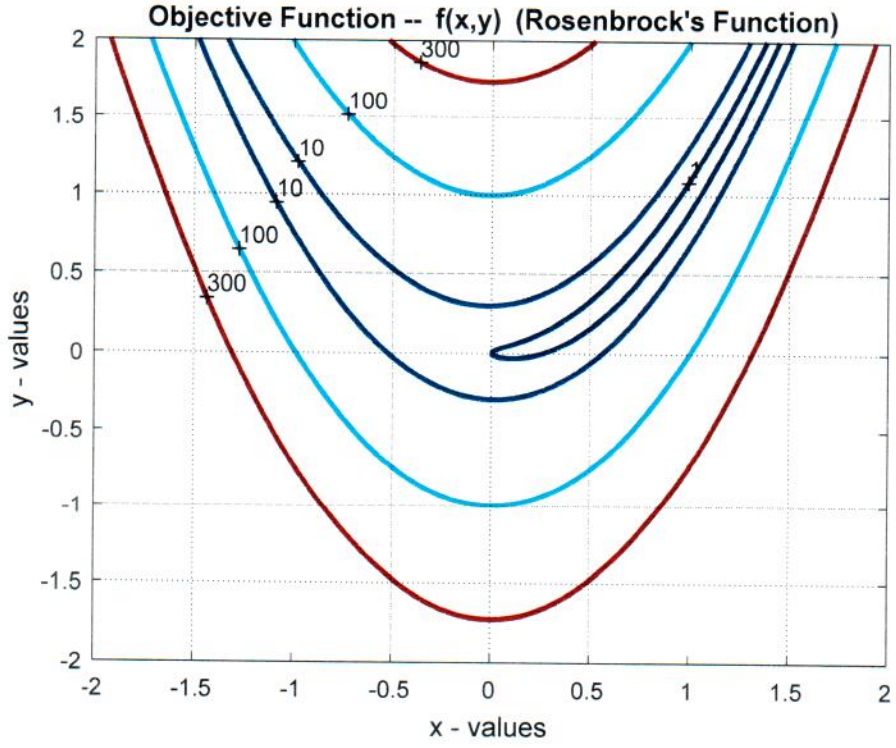
```
%
%  ORBITS_1.M   Plots Orbits of the Inner Planets
%
%  This HW illustrates several aspects of programming within the Matlab
%  environment.  The goal here is simply to evaluate and plot the orbits of
%  the four inner planets of our solar system.  This will use a 2-D array
%  for storage of information, and address plotting the orbits on Cartesian and
%  polar coordinate axes.  The proper use of 'dot arithmetic' is critical here...
%
%  Reference:  This problem came from Prob. 5.15 in the book, Matlab - An
%  Introduction With Applications, by A. Gilat (2004).  Note that the units of P
%  are in millions of kilometers.
%
%  File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

  clear all,  close all,   nfig = 0;
%
%  define the orbital constants for four planets (mercury, venus, earth, & mars)
  P = [269.2 15913 8964 2421];         % P values (10^6 km)
  e = [0.206 0.00677 0.0167 0.0934];   % eccentricity (unitless)
  Np = length(P);                      % number of planets
%
%  define theta vector and initialize 2-D array for function values
  Nth = 181;  thd = linspace(0,360,Nth)';  thr = thd*pi/180;   r = zeros(Nth,Np);
%
%  loop over number of planets and compute r vs thr for each planet
  for j = 1:Np
    r(:,j) = e(j)*P(j)./(1 - e(j)*cos(thr));
  end
%
%  as a quick sanity check, calc the average radius of the earth from the sun and
%  compare to the 'known value' of about 93 million miles  (result = 93.04  --  OK)
  cf = 1.60934;                    % conversion factor (km/mile)
  EarthR = mean(r(:,3))/cf;        % ave earth-sun distance in 10^6 miles
  disp('Average radius from sun to earth in millions of miles: ');  disp(EarthR)
%
%  create polar plot of the orbits
  ss = ['r- ';'g: ';'b-.';'m--'];
  nfig = nfig+1;   figure(nfig)
  for j = Np:-1:1          % reverse order because Matlab had scaling problems
    h = polar(thr,r(:,j),ss(j,:));hold on;
    set(h,'LineWidth',2)  % increase the line width for better visualization
  end
  legend('Mars','Earth','Venus','Mercury','Location','NorthWest')
  title('Orbits of the Inner Planets (Polar Coordinates with r in 10^6 km)')
%
  chk = 0;    % chk = 1, do forward order just to check current Matlab version
  if chk == 1
    nfig = nfig+1;   figure(nfig)
    for j = 1:Np
      h = polar(thr,r(:,j),ss(j,:));hold on;
      set(h,'LineWidth',2)  % increase the line width for better visualization
    end
  end
%
```

```
%   create x,y plot of the orbits (keep same order as above for consistency)
      nfig = nfig+1;    figure(nfig)
      for j = Np:-1:1
        x = r(:,j).*cos(thr);  y = r(:,j).*sin(thr);
        plot(x,y,ss(j,:),'LineWidth',2),grid on,hold on,axis equal
      end
      xlim([-300 300]);  ylim([-250 250]);
      xlabel('Distance from Sun (million kilometers)')
      ylabel('Distance from Sun (million kilometers)')
      legend('Mars','Earth','Venus','Mercury','Location','NorthWest')
      title('Orbits of the Inner Planets (Cartesian Coordinates)')
%
%   end of program
```

Objective Function #3 Results








```
>> obj_func_3
Minimum value of the objective function:    0.0000
x - location of minimum of f:               1.0000
y - location of minimum of f:               1.0000
```

```
%
%  OBJ_FUNC_3.M    Function evaluation and 3-D plotting in Matlab
%
%  Plot f(x,y) = 100*(y - x^2)^2 + (1 - x)^2  to find local minimum
%
%  This file computes and plots the objective function f(x,y).  The goal here is
%  to visualize the functional behavior and to find any minima that may occur.
%  Matlab has a variety of capabilities here, and several alternatives are used to
%  view this two-dimensional function using the surf and contour commands.
%
%  This function is the well-known Rosenbrock function that is used in studying
%  optimization methods. Although this function has an obvious minimum at x = y = 1,
%  we actually use Matlab's fminsearch to find this value (just to illustrate its
%  use)...
%
%  File prepared by J. R. White, UMass-Lowell  (last update: Sept. 2017)
%

  clear all,  close all,  nfig = 0;

%  identify basic problem data
  x = -2:0.004:2;   Nx = length(x);      % range for x
  y = -2:0.005:2;   Ny = length(y);      % range for y
  [xx,yy] = meshgrid(x,y);               % matrix of independent variables

%  compute objective function (be careful with "dot" arithmetic)
  f = 100*(yy - xx.^2).^2 + (1 - xx).^2;
  f(f > 400) = nan;                      % eliminate region where f > 400

%  now plot the function in a number of ways
  nfig = nfig+1;   figure(nfig)
  surf(xx,yy,f), shading interp; colorbar, colormap jet
  title('Objective Function --  f(x,y)  (Rosenbrock''s Function)')
  xlabel('x - values'),ylabel('y - values'),zlabel('Objective Function')
  axis tight
%
  nfig = nfig+1;   figure(nfig)
  surf(xx,yy,f), view(-150,45), shading interp; colorbar, colormap jet
  title('Objective Function --  f(x,y)  (Rosenbrock''s Function)')
  xlabel('x - values'),ylabel('y - values'),zlabel('Objective Function')
  axis tight
%
  nfig = nfig+1;   figure(nfig),
  surf(xx,yy,f), view(2), shading interp; colorbar, colormap jet
  title('Objective Function --  f(x,y)  (Rosenbrock''s Function)')
  xlabel('x - values'),ylabel('y - values'),zlabel('Objective Function')
  axis tight
%
  nfig = nfig+1;   figure(nfig)
  vc = [1 10 100 300];
  [con,han] = contour(xx,yy,f,vc); grid, colormap jet
  clabel(con), set(han,'LineWidth',2)
  title('Objective Function --  f(x,y)  (Rosenbrock''s Function)')
  xlabel('x - values'),ylabel('y - values')
%
  nfig = nfig+1;   figure(nfig)
```

```
    vc = [0.01 0.1 1 ];
    [con,han] = contour(xx,yy,f,vc); grid, colormap jet
    clabel(con), set(han,'LineWidth',2)
    r = [0.5 1.5 0.5 1.5];  axis(r)
    title('Objective Function --  f(x,y)  (Rosenbrock''s Function)')
    xlabel('x - values'),ylabel('y - values')
%
%   the contour plot clearly shows a distinct minimum in the region around x = y = 1.
%   just for fun, let's use fminsearch to formally find this minimum
    fz = @(z)  100*(z(2) - z(1)^2)^2 + (1 - z(1))^2;
    [locm,fm] = fminsearch(fz,[0.5 0.5]);  % use x = y = 0.5 as a starting guess
    fprintf('Minimum value of the objective function: %8.4f \n',fm);
    fprintf('x - location of minimum of f:            %8.4f \n',locm(1));
    fprintf('y - location of minimum of f:            %8.4f \n\n',locm(2));
%
%   end of problem
```