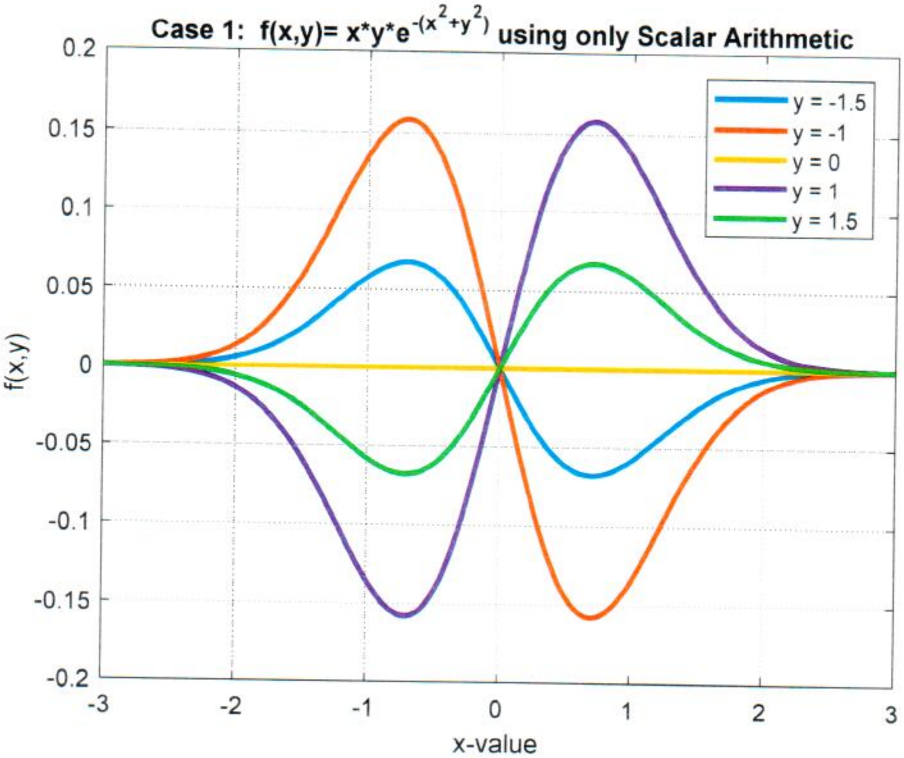
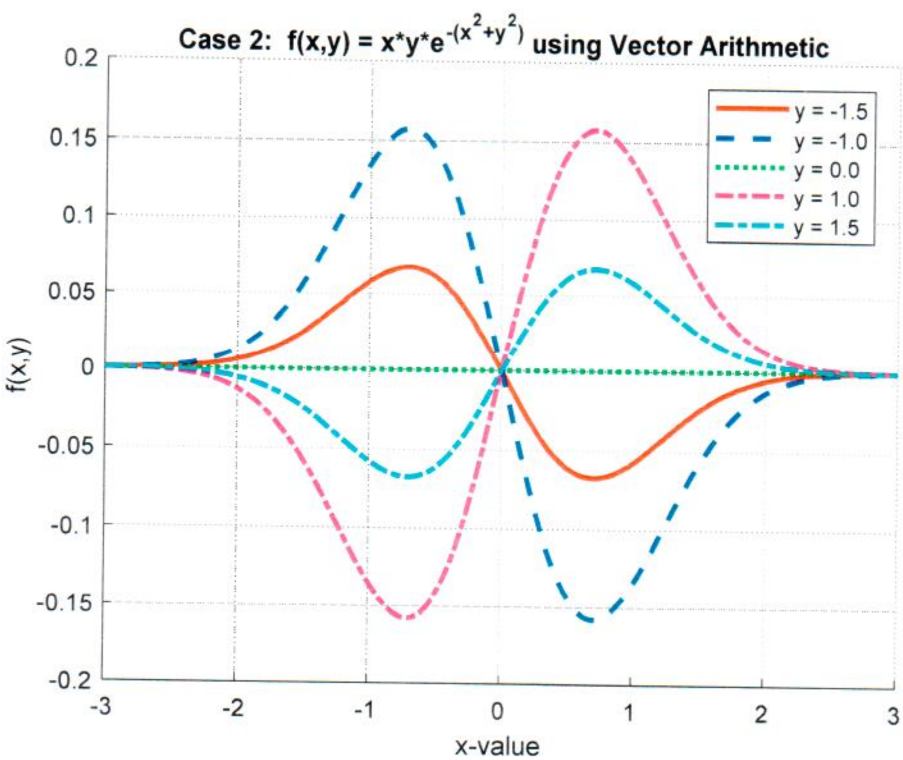
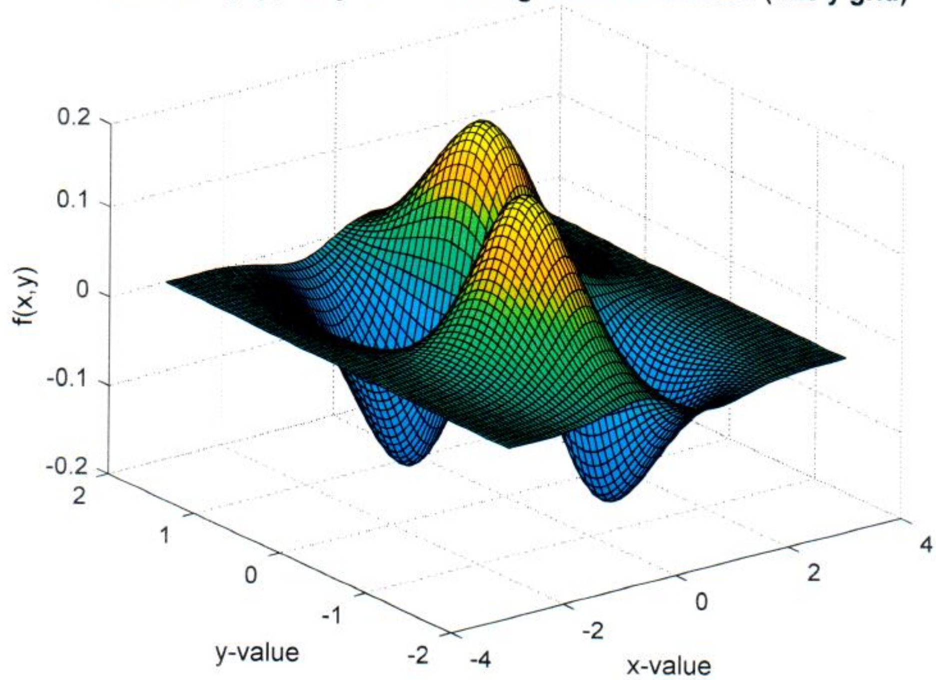
Results from Instructor-Led Example for Lab #2b

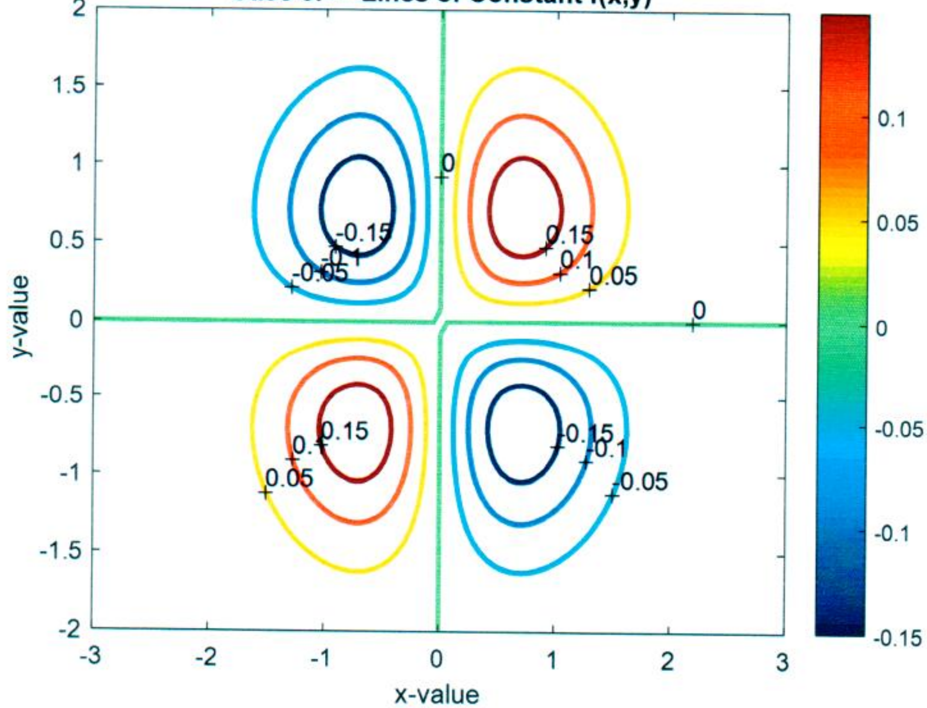




Case 3: $f(x,y) = x*y*e^{-(x^2+y^2)}$ using Matrix Arithmetic (fine y grid)



Case 3. -- Lines of Constant f(x,y)

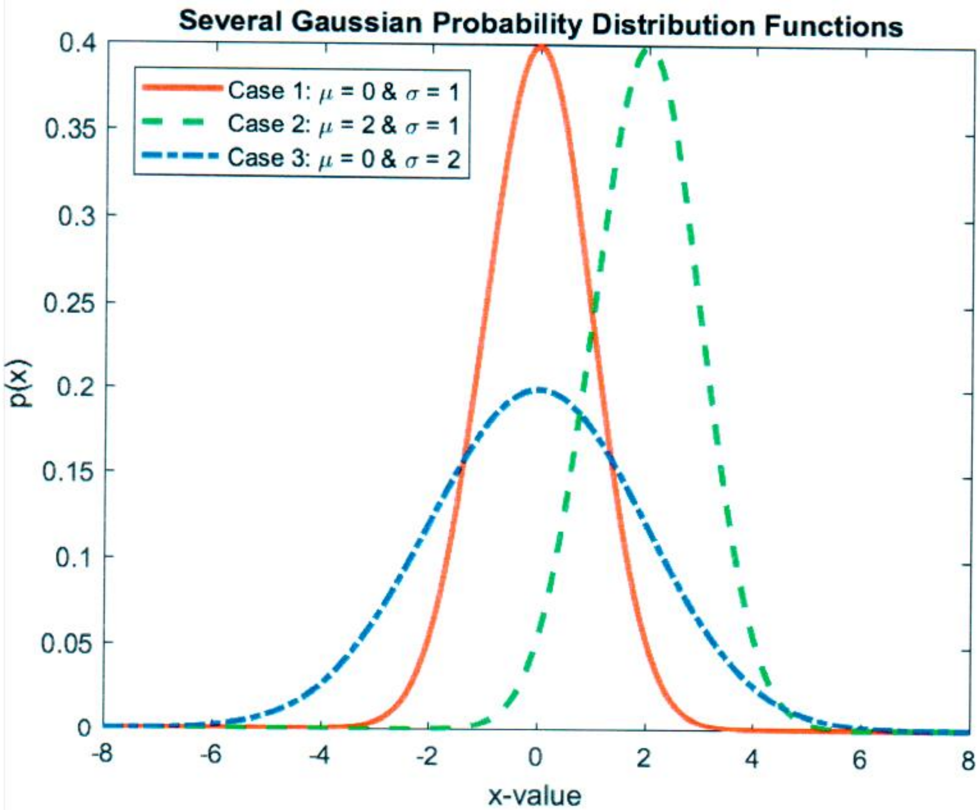


```
%
%   FXY_1.M    Evaluating and plotting functions with Matlab
%
%   Basic Problem Description
%     Compute and plot a simple 2-D function, f(x,y), over a continuous range
%     of x values for a few discrete values of y.
%   The function is;  f(x,y) = x*y*exp(-(x^2+y^2))
%   Do three cases:
%      Case 1:  use nested loops and only scalar arithmetic
%      Case 2:  use a single loop and vector arithmetic, as needed
%      Case 3:  use no loops and array manipulations (with the meshgrid function)
%
%   File prepared by J. R. White, UMass-Lowell  (last update: Sept. 2017)
%

%
     clear all,   close all,   nfig = 0;
%
     y = [-1.5 -1 0 1 1.5];  Ny = length(y);  % few specific values of y (snapshots)
     Nx = 201;  x = linspace(-3,3,Nx)';        % define x values (col vector)
%
%   Case 1  --  scalar arithmetic
     f1 = zeros(Nx,Ny);                       % initialize storage space for f(x,y)
     for j = 1:Ny
       for i = 1:Nx
         f1(i,j) = x(i)*y(j)*exp(-(x(i)^2 + y(j)^2)); % do needed calcs (no dots)
       end
     end
%
     nfig = nfig+1;  figure(nfig)
     plot(x,f1,'LineWidth',2), grid
     title('Case 1:  f(x,y)= x*y*e^{-(x^2+y^2)} using only Scalar Arithmetic')
     xlabel('x-value'), ylabel('f(x,y)')
     legend('y = -1.5','y = -1','y = 0','y = 1','y = 1.5')  % simple but static
%
%   Case 2  --  vector arithmetic
     f2 = zeros(Nx,Ny);                       % initialize storage space for f(x,y)
     for j = 1:Ny
       f2(:,j) = x*y(j).*exp(-(x.*x + y(j)^2));  % do needed vector calcs  (need dots
here)
     end
%
     st = cell(Ny,1);
     ps = ['r- ';'b--';'g: ';'m-.';'c-.'];
     nfig = nfig+1;  figure(nfig),   hold on
     for j = 1:Ny
       plot(x,f2(:,j),ps(j,:),'LineWidth',2)
       tt = sprintf('y = %3.1f ',y(j));   st(j) = cellstr(tt);
     end
     title('Case 2:  f(x,y) = x*y*e^{-(x^2+y^2)} using Vector Arithmetic')
     xlabel('x-value'), ylabel('f(x,y)'), grid on
     legend(st)                               % more complex but dynamic
     hold off
%
%   Case 3  --  plot full surface map (just for fun -- not required as part of quiz)
%   coarse y grid (not recommended)
```

```
      [X,Y] = meshgrid(x,y);
      f3 = X.*Y.*exp(-(X.*X + Y.^2));
      nfig = nfig+1;  figure(nfig)
      surf(x,y,f3)
      title('Case 3:  f(x,y) = x*y*e^{-(x^2+y^2)} using Matrix Arithmetic (coarse y
grid)')
      xlabel('x-value'), ylabel('y-value'),zlabel('f(x,y)')

%   finer y grid (recommended)
      Ny = 51;  y = linspace(-2,2,Ny);            % define y values
      Nx = 101;  x = linspace(-3,3,Nx)';          % define x values
      [X,Y] = meshgrid(x,y);
      f3 = X.*Y.*exp(-(X.*X + Y.^2));
      nfig = nfig+1;  figure(nfig)
      surf(x,y,f3)
      title('Case 3:  f(x,y) = x*y*e^{-(x^2+y^2)} using Matrix Arithmetic (fine y grid)')
      xlabel('x-value'), ylabel('y-value'),zlabel('f(x,y)')
%
%   one final final plot, how about a contour plot where we set the levels?
      nfig = nfig+1;   figure(nfig)
      [cs,h] = contour(X,Y,f3,[-0.15 -0.1 -0.05 0.0 .05 0.1 0.15 ]);
      clabel(cs),  colorbar,  colormap(jet),  grid
      set(h,'LineWidth',2)
      title('Case 3. --  Lines of Constant f(x,y)')
      xlabel('x-value'), ylabel('y-value')
%
%   end of problem
```

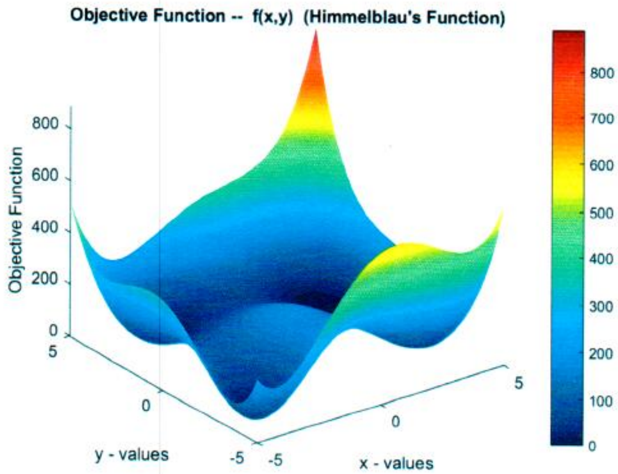


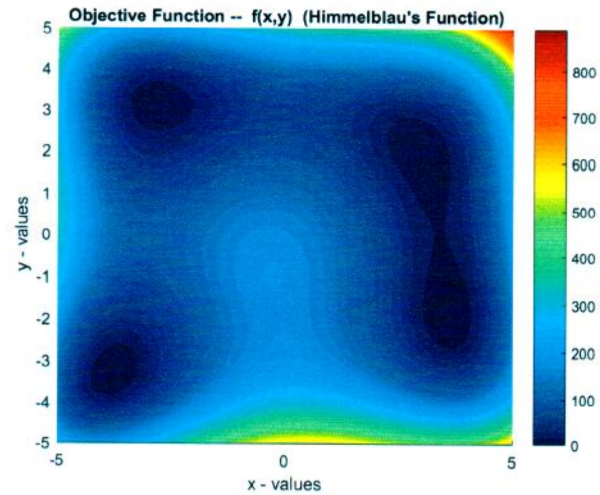
Several Gaussian Probability Distribution Functions
Case 1: μ = 0 & σ = 1
Case 2: μ = 2 & σ = 1
Case 3: μ = 0 & σ = 2
p(x)
x-value
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
-8
-6
-4
-2
0
2
4
6
8

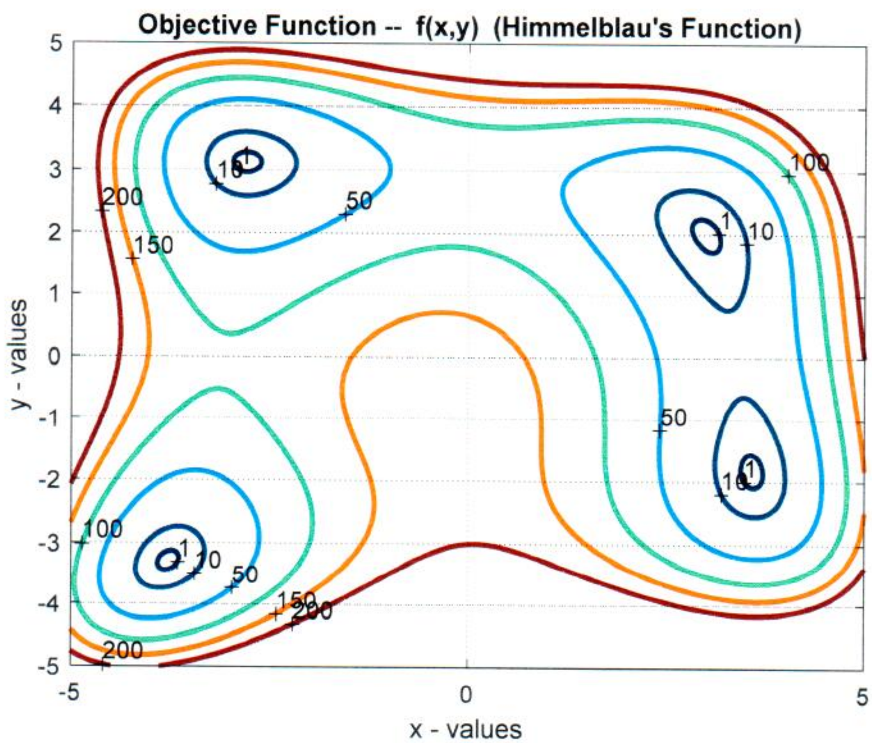
```
%
%  GAUSSIAN_Lab2b.M   Plots Gaussian Distribution Function for different
%                      mean values and standard deviations
%
%  This lab exercise illustrates several aspects of programming within the Matlab
%  environment.  The goal here is simply to evaluate and plot the Gaussian
%  probability distribution function for different means and standard
%  deviations.  The function of interest is:
%
%                   1                (x-mu)^2
%       p(x) = ------------  exp(- ----------)
%              sig*sqrt(2pi)          2*sig^2
%
%  where sig = standard deviation and mu = mean value
%
%  We want to look at three cases for different mu and sig combinations.  Since the
%  probability distribution is a function of two variables, independent variable x
%  and case number n, we will store the resultant probabilities in a 2-D array
%  P(x,n).  Thus, this file also illustrate how to evalaute and plot a function of
%  two variables within Matlab.  For this implementation, the vector approach is
%  used where a single for ... end loop over the number of cases is used.
%
%  File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

    clear all,  close all,   nfig = 0;
%
%  define the mean and standard deviation values for the three cases of interest
    mu = [0 2 0];          % mean values
    sig = [1 1 2];         % standard deviations
    Nc = length(mu);       % number of cases
%
%  define x-vector and initialize 2-D array for function values
    xo = -8;  xf = 8;  Np = 161;  x = linspace(xo,xf,Np)';   p = zeros(Np,Nc);
%
%  loop over number of cases (do proper element-by-element arithmetic within loop)
    for n = 1:Nc
      cc1 = 1/(sig(n)*sqrt(2*pi));    cc2 = (x - mu(n)).^2/(2*sig(n)^2);
      p(:,n) = cc1*exp(-cc2);
    end
%
%  plot p(x,n) for all cases on single plot
    nfig = nfig+1;   figure(nfig)
    plot(x,p(:,1),'r-',x,p(:,2),'g--',x,p(:,3),'b-.','LineWidth',2), grid
    title('Several Gaussian Probability Distribution Functions')
    xlabel('x-value'),ylabel('p(x)')
    legend(['Case 1: \mu = ',num2str(mu(1)),' & \sigma = ',num2str(sig(1))],...
           ['Case 2: \mu = ',num2str(mu(2)),' & \sigma = ',num2str(sig(2))],...
           ['Case 3: \mu = ',num2str(mu(3)),' & \sigma = ',num2str(sig(3))], ...
           'Location','NorthWest')
%
%  end of program
```

Objective Function #2 Results (Himmelblau's Function)







Comments:

Note that the approximate locations of the minima of f(x,y) are obvious from the labeled contour plot. The minima locations can also be seen to some degree in the other plots (i.e., the surface plots), but the locations are displayed much less clearly. Thus, if the primary goal is to locate the minima, then the labeled contour plot is probably the best choice here. However, the first (upper left) surface plot is probably the best for showing the overall functional behavior of this particular 2-D function. Thus, the choice of the "best" visualization option is often selected based on the goals for the study of interest…

```
%
%   OBJ_FUNC2_Lab2b.M    Function evaluation and 3-D plotting in Matlab
%
%   Plot f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2  to find local min
%
%   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 Himmelblau function that is used in studying
%   optimization methods and in illustrating the fact that objective functions can
%   have several local minima (here there are four local mimina).
%
%   File prepared by J. R. White, UMass-Lowell  (last update: Sept. 2017)
%

      clear all,  close all,  nfig = 0;
%
%   identify basic problem data
      x = -5:0.01:5;  Nx = length(x);        % range for x
      y = -5:0.02:5;  Ny = length(y);        % range for y
      [xx,yy] = meshgrid(x,y);               % matrix of independent variables
%
%   compute objective function (be careful with "dot" arithmetic)
      f = (xx.^2 + yy - 11).^2 + (xx + yy.^2 - 7).^2;
%   if time permits, uncomment the following line of code to show what it does...
%      f(f>250) = nan;  % removes elements for f>250 (sets to nan which is not plotted)
%
%   now plot the function in a number of ways
      nfig = nfig+1;   figure(nfig),
      surf(xx,yy,f), shading interp; colormap jet;  colorbar
      title('Objective Function --  f(x,y)  (Himmelblau''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; colormap jet;  colorbar
      title('Objective Function --  f(x,y)  (Himmelblau''s Function)')
      xlabel('x - values'),ylabel('y - values')
      zlabel('Objective Function')
      axis tight
%
      nfig = nfig+1;   figure(nfig)
      vc = [ 1 10 50 100 150 200];
      [con,han] = contour(xx,yy,f,vc); grid; colormap jet;
      clabel(con), set(han,'LineWidth',2)
      title('Objective Function --  f(x,y)  (Himmelblau''s Function)')
      xlabel('x - values'),ylabel('y - values')
%
%   end of problem
```