1. (20 points) **Rootfinding and Fractals!**

Let $f$ be the complex function given by $f(z) = (z^2 - 1)(z^2 + 0.16)$, where $z = x + iy$. The equation has 4 roots given by $z_1 = 1$, $z_2 = -1$, $z_3 = 0.4i$, and $z_4 = -0.4i$. In this exercise, we consider finding the roots of $f(z)$ using Newton’s Method for finding roots.

Suppose that we are given $z_0 = x_0 + iy_0$ as a starting point for Newton’s method. Which root will Newton’s method find? As it turns out, there is no good way to predict which root the method will find. For this exercise, you will investigate which root Newton’s method finds for various starting points and will generate a beautiful fractal as a result!

First, complete the following function which performs Newton’s Method and returns the color corresponding to the root it found in at most 20 iterations or to failure (in the case it does not find a root):

```matlab
function color = Newton(z)
    
    % Input:
    % z = complex number
    
    % Output:
    % color = the color associated with the root found by Newton's Method
    % according to the following key:
    % color = 'y', if Newton's Method found z_1
    % 'r', if Newton's Method found z_2
    % 'b', if Newton's Method found z_3
    % 'g', if Newton's Method found z_4
    % 'k', if Newton's Method was unsuccessful
    % after 20 iterations.
    
    % More precisely, "unsuccessful" means that Matlab cannot reduce
    % (z^2-1)(z^2+0.16) to less than 0.0001 (in absolute value) in at most
    % 20 Newton iterations.

    Second, write a script that calls Newton.m with various starting values $z=x+iy$ where

    ```matlab
    x = linspace(0.15,0.55,150);
y = linspace(-0.15,0.15,150).
    ```

    Save the $x$ and $y$ coordinates and the colors returned by Newton.m in vectors as appropriate. Finally, plot the points according to the obvious scheme: if using $z = x+iy$ as a starting point for Newton’s method yielded $color = 'r'$, then plot $z = x+iy$ red.

    Turn in your Matlab function, your Matlab script, and the fractal that your code generated. It is fine to turn in a black-and-white plot of the fractal. In addition, write a brief summary of what this experiment shows us about using Newton’s method to solve nonlinear equations.

HELPFUL HINT:

- It is helpful to know how to convert between complex numbers and their real and imaginary parts.
  See the help for complex.m, real.m, and imaginary.m for examples of how this is done.
2. (10 points) Classify each of the following functions as either coercive or non-coercive showing why your classification is correct.

(a) \( x^3 + y^3 + z^3 - xyz \)
(b) \( x^2 + y^2 + z^2 - \sin(xyz) \).

3. (10 points) Show that each of the following functions is convex or strictly convex.

(a) \( 5x^2 + 2xy + y^2 - x + 2y + 3 \)
(b) \( 4e^{3x} - y + 5e^{x^2} + y^2 \).

4. (25 points) Down in the Valley...

Consider minimizing the function \( f(x_1, x_2) = (x_1 - 1)^2 + 10(x_2^2 - x_2)^2 \). This function has a unique minimum at \((x_1, x_2) = (1, 1)\). However, this function often proves difficult for optimization software to minimize due to the steep sides of the valley in which the minimum lies. On this exercise, we will minimize this function using the Steepest Descent Method.

First, implement the Steepest Descent Method; it can be found on p. 277 in your textbook. You will note that the second step in the for loop is to “Choose \( \alpha_k \) to minimize \( f(x_k + \alpha_k s_k) \).” This is a 1D minimization problem which does not correspond to a unique implementation. Here is how you will find an approximate solution to the above problem in your implementation.

Define \( f_k(\alpha) = f(x_k + \alpha s_k) \). Assume that \( \alpha_2 > \alpha_1 \) and either \( f_k(\alpha_2) < f_k(0) \) or \( f_k(\alpha_1) < f_k(0) \). (In practice this will mean cutting the interval in half (up to a maximum of 20 times) until this property occurs.) Define \( q(\alpha) \) to be the quadratic function that interpolates \((0, f_k(0)), (\alpha_1, f_k(\alpha_1)), \) and \((\alpha_2, f_k(\alpha_2))\). Let \( \alpha_k \) be the minimizer of \( q(\alpha) \) on \([0, \alpha_2]\).

Your implementation should be a Matlab function called SteepestDescentStep.m and should satisfy the following function prototype:

```matlab
function [xnew,fnew,gnew] = SteepestDescentStep(xk,fk,gk,Lmax)
% % Input:
% % xk = a column vector that corresponds to the current position
% % fk = f(xk)
% % gk = g(xk)
% % Lmax = maximum step length
% %
% % Output:
% % xnew = a column vector with the new position
% % fnew = f(xnew)
% % gnew = g(xnew)
% %
% The Steepest Descent step is computed according to the specifications above.
```

Second, write a script that will employ the Steepest Descent method in order to minimize \( f(x_1, x_2) \). Your script should be called Valley.m and should do the following:

- Make a detailed contour plot (with about 150 contour lines) of this function on \([-1, 1.5] \times [-1, 1.5]\). In order make this and all other contour plots for this assignment, use the Rosenbrock.m function which can be downloaded from the course website. This plot should be in a figure window by itself.
Using $(-0.5, 1.5)$ as a starting point, a maximum of 500 iterations, and a maximum step length of 1, execute the Steepest Descent Method until either the maximum number of iterations is reached or the function value becomes less than or equal to $1e-8$. (This is an acceptable stopping heuristic, since we know that the minimum value of the function is 0.) You should store the values of $x$ from each iteration. After SteepestDescentStep returns, you should make the above contour plot in a second figure window and then superimpose the successive Steepest Descent iterates on this plot using both 'k-' and 'k.'.

If any of the iterates go outside the range of the contour plot (for the Steepest Descent method), you should enlarge the contour plot so that the iterates are all superimposed on the contour plot.

Finally, write a 2 sentence summary discussing the performance of the Steepest Descent method as applied to this particular objective function.

Turn in all of your Matlab functions and script, your plots, and your summary. It is OK to turn in black and white plots.