Caleb Logemann
MATH 565 Continuous Optimization
Homework 1

1. Problem 1
   Compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ of the Rosenbrock function
   $$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

2. Problem 8
   Suppose that f is a convex function. Show that the set of global minimizers of f is a convex set.

   Proof. Let f be a convex function. This implies that □

3. Problem 9
   Consider the function $f(x_1, x_2) = (x_1 + x_2^2)^2$. At the point $x^T = (1, 0)$ we consider the search direction $p^T = (-1, 1)$. Show that p is a descent direction and find all minimizers of the problem (2.10).

4. Problem 13
   Show that the sequence $x_k = 1/k$ is not Q-linearly convergent, though it does converge to zero.

5. Problem 14
   Show that the sequence $x_k = 1 + (0.5)^{2^k}$ is Q-quadractically convergent to 1.

6. Consider the following fixed point iteration scheme:
   $$x_{k+1} = x_k - \frac{[g(x_k)]^2}{g(x_k + g(x_k)) - g(x_k)}$$
   Prove that if this method is converges to a root x^* of $g(x)$ such that $g'(x^*) \neq 0$ and $g''(x^*) \neq 0$, then the rate of convergence is quadractic: $p = 2$.

   Proof. □

7. Implement the method from problem 6 in Matlab (or Python if you prefer). Use it to solve problems 8 and 9.

   The following function implements the method from problem 6.

```
import numpy as np
def problem7(g, x0, TOL, MaxIter):
    x = np.zeros(MaxIter+2)
    x[0] = x0
    nIter = 0
    mstop = 1
    gx = g(x[nIter])
    delta = -np.square(gx)/(g(x[nIter] + gx) + gx)

    while nIter <= MaxIter and mstop:
        nIter+=1
        x[nIter] = x[nIter - 1] + delta
        gx = g(x[nIter])
        if abs(gx) < TOL and abs(delta) < TOL:
            mstop = 0
        else:
            delta = -np.square(gx)/(g(x[nIter] + gx) + gx)
    x = x[:nIter+1]
    return x
```

8. The van der Waal equation

$$\left(P + \frac{a}{V^2}\right)(V - b) = nRT$$

generalizes the ideal gas law $PV = nRT$. In each equation, P represents the pressure (atm), V represents the volume (liters), n is the number of moles of gas, and T represents the temperature (K). R is the universal gas constant and has the value

$$R = 0.08205 \frac{\text{liters} \cdot \text{atm}}{\text{mole} \cdot \text{K}}$$

Determine the volume of 1 mole of isobutane at a temperature of $T = 313K$ and a pressure of $P = 2$ atm, given that, for isobutane, $a = 12.87 \text{atm} \cdot \text{liters}^2$ and $b = 0.1142$ liters. Compare this to the value predicted from the ideal gas law. You may use any one of your methods (make clear in your writeup which one you are using, what initial guesses or intervals you are using, etc...).

The following script uses the method implemented in problem 7 to find the volume that satisfies the van der Waal equation.

```
import numpy as np
import matplotlib.pyplot as plt
execfile('01_7.py')

P = 2
a = 12.87
b = 0.1142
n = 1
R = 0.08205
T = 313
g = lambda V: (P + a/np.square(V))*(V - b) - n*R*T
V0 = 12

TOL = 1e-10
MaxIter = 1000

sol = problem7(g, V0, TOL, MaxIter)
print 'The␣volume␣of␣the␣isobutane␣is␣{0:.10f}'.format(sol[-1])

x = np.linspace(1,20,10000)
plt.plot(x, g(x), '-', sol[-1], g(sol[-1]), 'o')
```

9. According to ArchimedesâĂŹ law, when a solid of density ÏČ is placed in a liquid of density ρ, it will sink to a depth h that displaces an amount of liquid whose weight equals the weight of the solid. For a sphere of radius r, Archimedes law is

$$\frac{1}{3}\pi\left(3rh^2 - h^3\right)\rho = \frac{4}{3}\pi r^3 \sigma$$

Given $r = 5$, $\rho = 1$, and $\sigma = 0.6$, determine h. You may use any one of your methods (make clear in your writeup which one you are using, what initial guesses or intervals you are using, etc...).

```
import numpy as np
import matplotlib.pyplot as plt
execfile('01_7.py')

r = 5.0
rho = 1.0
sigma = .6
g = lambda h: 1.0/3.0 * math.pi * (3.0*r*np.square(h) - np.power(h, 3.0)
    ↪ )*rho - 4.0/3.0*math.pi*np.power(r, 3.0)*sigma
h0 = 5.67

TOL = 1e-10
MaxIter = 1000

sol = problem7(g, h0, TOL, MaxIter)
print 'The␣depth␣of␣the␣sphere␣is␣{0:.10f}'.format(sol[-1])

x = np.linspace(1,10,10000)
plt.plot(x, g(x), '-', sol[-1], g(sol[-1]), 'o')
```