**Faculty of Arts and Sciences**
**Department of Computer Science**
**CMPS 351 – Numerical Optimization**
**Assignment 2 – Due Feb 4, 2019**

Reading Material:

- Chapter 2 and 3 of Ref. 1 (Nocedal and Wright)

**1. Hessian Computation.** Compute the second derivatives of the Rosenbrock function analytically, and use them to write a function that returns the Hessian of the Rosenbrock function.

```
def rosen_hessian(x)
```

**2. Newton's Method with Backtracking Line Search.** Write a routine for a backtracking line search method using the Newton direction instead of the steepest direction of problem 3 of the last assignment. The header of your function should look something like this:

```
def newtont_bt(fun, grad, hess, x0)
```

where `fun`, `grad`, and `hess` are functions that return the objective function, its gradient, and its Hessian, respectively.

**3. Performance of Newton's Method.** Test the performance of the algorithm on Rosenbrock's function $f(x) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$ starting at the point $[-1.2\ 1.0]^t$ by finding the number of iterations till convergence to a gradient norm of $10^{-5}$.

- Generate a semilog plot of $|f^k - f^*|$ vs. iteration count.
- Write an expression for the expected convergence behavior of the algorithm near the minimum
- Comment on the convergence plot, and compare it to that of the steepest descent direction

**4. Problem in $\mathbf{R}^{100}$.** Consider the problem of minimizing the following function with $x \in \mathbf{R}^{100}$ and $a_i$ given vectors.

$$f(x) = -\sum_{i=1}^{500} \log(1 - a_i^t x) - \sum_{i=1}^{100} \log(1 - x_i^2)$$

- Derive expressions for the gradient and Hessian that would be needed for using Newton's method.
- Write the system of linear equations that needs to be solved at every iteration for finding the Newton direction.
- The backtracking line search routine we wrote earlier must be modified so that it does not return a step length that places $x$ outside the region where the objective function is defined, i.e., $\min_i(1 - a_i^t x)$ and $\min_i(1 - x_i^2)$ must be positive. We call this condition a feasibility condition.

  Modify `backtrack` so that it first decreases the step length until the feasibility condition is satisfied, and then continues decreasing it further until the admissibility condition is satisfied.
- Generate a random coefficient matrix $A$ ($100 \times 500$) whose columns serve as the vectors $a_i$. You may use, for example, the numpy function `randn`. Solve the problem starting from an initial (feasible) point. Plot the convergence behavior.