University of Illinois **Fall 2021**

ECE 490: Programming Assignment 1
Unconstrained Optimization

Due: Monday, 20 February 2023 5:00pm

For this project we will explore different ways to solve the problem

$$\min_{x\in\mathbb{R}^n} x^T Q x + b^T x + c := f(x) \tag{1}$$

Recall that coding in Python is required. Your code should be flexible enough to run for different values of n, Q, b and c. Here Q is assumed to be a positive definite matrix. Use the following code snippet to generate different Q, b, c:

```python
import numpy as np
def get_Q_b_c(n):
    Q = np.random.rand(n,n)-0.5
    Q = 10*Q @ Q.T
    b = 5*(np.random.rand(n)-0.5)
    c = 2*(np.random.rand(1)-0.5)
    return Q,b,c
# Example:
Q,b,c = get_Q_b_c(10)
```

The assignment will be done in groups of 3. The code must be uploaded to Gradescope by the deadline. One group member will be required to run the code in front of a TA during the assigned office hour slots (**in-person**) and the grade will be handed out for all group members right then. Please look at the course website for the group assignment and the allotted time slots.

1. [**Constant step size**]
   Write your own code to solve (1) using gradient descent with constant step size. Choose the step size based on theoretical convergence of steepest descent. Write the code so that the error tolerance, ϵ, is a variable so that you can try out different values for ϵ. By error tolerance we mean your code stops the iterations and outputs your estimate for the minimum when the following condition is satisfied at iteration k
   $$\|\nabla f(x_k)\| < \epsilon.$$
   Please make sure to print the output x^*, $f(x^*)$, ϵ and the number of iterations required for convergence.

2. [**Armijo's rule**]
   Write your own code to solve (1) using an iterative method where the step size α_k is chosen using Armijo's rule (also known as backtracking line search). The book has guidelines on how to choose the parameters in Armijo's rule. The code should have the same termination criterion as given in the first problem. Please make sure to print the output x^*, $f(x^*)$, ϵ and the number of iterations required for convergence.

3. [**Matrix Inversion**]
   Use matrix inversion (found in packages such as **numpy.linalg** or **scipy.linalg** to check your solution to part (a). Print x^* and $f(x^*)$.

What will happen during the demonstration session:

You will be given asked to add a line to your code: **numpy.random.seed()** to fix the matrices Q, b, c produced, and you will be asked to use your code with a given value of n to generate the solutions. You are expected to be able to comfortably explain your code to the TA, and answer some follow-up questions.