Indian Institute of Technology, Bombay — Chemical Engineering
CL603: Optimization — Assignment 4, 2023
Date Due: 11 Feb 2023 (Sat) by 11.59 PM (midnight).
Mode of Submission: Moodle submission and MATLAB Grader
Important Instructions:(i) There is NO concept of LATE SUBMISSION. So, please submit by the deadline. If you do not submit by the deadline, then you would not get any credits. (ii) Students working in Matlab should submit assignment (Assignment4_Matlab) on moodle (and also on matlab grader: see instructions below), and (ii) Students working in Python should submit assignment (Assignment4_Python) on moodle.

Topic: Line Search Steepest Descent, Newton and Quasi-Newton Method

Aim

To implement Steepest Descent, Newton Method and Quasi-Newton Method for unconstrained minimization of a function.
Use $x_{initguess} = [1.5\ 1.5]^T, N = 15000$ as the maximum number of iterations and $\epsilon = 10^{-6}$ as the tolerance on square of gradient-norm. Thus, the iterations should terminate if either the maximum number of iterations have been reached or if $||\nabla f(x^k)||^2 < \epsilon$.

Python Code To Submit

A Python script is a standalone file which can contain everything from functions, computations, plotting schemes, etc. This time we will use **Python Grader** for evaluation of your Python Submission. On Moodle specific template is uploaded, you have to write your logic as per the template.

Note: (1) Do not edit any statement provided on template.
(2) Submit a python file named `tut_04_ROLLNO.py where ROLLNO is your roll number`

Your python script should do the following:

1. Implement Steepest Descent with in-exact line search methods i.e. at iteration $(k+1)$ we have
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \nabla f(\mathbf{x}^k) \tag{1}$$

2. Implement Newton method with in-exact line search methods i.e. at iteration $(k+1)$ we have
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k [\nabla^2 (f(\mathbf{x}^k))]^{-1} \nabla f(\mathbf{x}^k) \tag{2}$$

3. Implement BFGS for update of Hessian inverse method with inexact line search i.e. at iteration $(k+1)$ we have
$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \mathbf{C}^k\ \nabla f(\mathbf{x}^k) \tag{3}$$
Take initial $\mathbf{C}$ matrix to be identity matrix.

Note: (1) You can call your Gradient and Hessian Function that you have already created in Assignment 2 whenever you need to evaluate the gradient and hessian.
(2) Consider initial guess of $\mathbf{x}$ as value of first iteration $(k = 1)$.
(3) For the step-length computation in-exact line search methods, use a backtracking strategy to choose an acceptable α. The parameters for this backtracking strategy are: $\overline{\alpha} = 5$,$\rho = 0.8$, c $= 0.1$. **Refer to Algorithm Backtracking Line Search in the notes to see the pseudo code.**

- Your code should generate plots as listed below:

1. Plot X versus iteration number i.e. x_1 versus iteration number and x_2 versus iteration number in same figure but as separate subplots. In each plot, you should have three curves corresponding to Steepest Descent with inexact line search, Newton with inexact line search, and BFGS inverse with inexact line search, respectively
2. Generate a figure which shows the value of $f(x)$ versus iteration number. Once again have three curves.
   Label the axes and give title in each figure you generate.

- Your code should also print the final (converged) value of $\mathbf{x}$, and the corresponding $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$. In case the iterations do not converge within the specified upper limit of N iterations, your code should print that "Maximum iterations reached but convergence did not happen" and also print the latest values of $\mathbf{x}$, the function and the gradient.

MATLAB Code To Submit

- Submit MATLAB code on **MATLAB Grader**. Please find link for MATLAB grader. `https://grader.mathworks.com/courses/96342-cl-603-optimization` Your code will be auto-graded on MATLAB grader. A template file is available on matlab grader. Write your code in that template file. You can run the code to check its correctness/outputs. After you are satisfied, then submit the code. Important: You will be able to submit the code ONLY ONCE. After submission you will be able to see your marks and errors (if any), but will not be able to modify your code. Thus, submit only after you are satisfied with your code. Submit before the deadline.

- Also submit all your MATLAB files (you can zip them) on Moodle for our records. Grading will however be on matlab grader.

Test functions for optimization

- Four test functions with an optimum point (X^*), optimum function value $F(X^*)$, and converged value of gradient $\nabla F(x^*)$ are given below to check the correctness of your code.

Table 1: Test function for Steepest Descent

S.no	Test function	Optimum point(X^*)	$F(X^*)$	$\nabla F(x^*)$
1	$f(x_1,x_2) = (x_1-1)^2 + (x_2-1)^2 - x_1x_2$	$[2 \;\; 2]^T$	-2	$[0.5962 \;\; 0.5962]^T \times 10^{-3}$
2	$f(x_1,x_2) = x_1^2 + x_2^2 + (0.5x_1+x_2)^2 + (0.5x_1+x_2)^4$	$[0 \;\; 0]^T$	0	$[0.2444 \;\; 0.4887]^T \times 10^{-3}$
3	$f(x_1,x_2) = -0.0001(\lvert sin(x_1)sin(x_2)exp(\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert)\rvert)^{0.1}$	$[1.3501 \;\; 1.3501]^T$	-2.06	$[0.1487 \;\; 0.1487]^T \times 10^{-3}$
4	$f(x_1,x_2) = (x_1+2x_2-7)^2 + (2x_1+x_2-5)^2$	$[1 \;\; 3]^T$	0	$[0.619 \;\; 0.2217]^T \times 10^{-3}$

Table 2: Test function for Newton Method

S.no	Test function	Optimum point(X^*)	$F(X^*)$	$\nabla F(x^*)$
1	$f(x_1,x_2) = (x_1-1)^2 + (x_2-1)^2 - x_1x_2$	$[2 \;\; 2]^T$	-2	$[0.5962 \;\; 0.5962]^T \times 10^{-3}$
2	$f(x_1,x_2) = x_1^2 + x_2^2 + (0.5x_1+x_2)^2 + (0.5x_1+x_2)^4$	$[0 \;\; 0]^T$	0	$[-0.7319 \;\; 0.4043]^T \times 10^{-3}$
3	$f(x_1,x_2) = -0.0001(\lvert sin(x_1)sin(x_2)exp(\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert)\rvert)^{0.1}$	$[1.3468 \;\; 1.3468]^T$	-2.06	$[-0.574 \;\; -0.574]^T \times 10^{-3}$
4	$f(x_1,x_2) = (x_1+2x_2-7)^2 + (2x_1+x_2-5)^2$	$[1 \;\; 3]^T$	0	$[-0.361 \;\; -0.567]^T \times 10^{-3}$

Table 3: Test function for Quasi-Newton Method

S.no	Test function	Optimum point(X^*)	$F(X^*)$	$\nabla F(x^*)$
1	$f(x_1,x_2) = (x_1-1)^2 + (x_2-1)^2 - x_1x_2$	$[2 \;\; 2]^T$	-2	$[0.5962 \;\; 0.5962]^T \times 10^{-3}$
2	$f(x_1,x_2) = x_1^2 + x_2^2 + (0.5x_1+x_2)^2 + (0.5x_1+x_2)^4$	$[0 \;\; 0]^T$	0	$[-0.7341 \;\; 0.2150]^T \times 10^{-3}$
3	$f(x_1,x_2) = -0.0001(\lvert sin(x_1)sin(x_2)exp(\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\rvert)\rvert)^{0.1}$	$[1.3516 \;\; 1.3516]^T$	-2.06	$[0.4853 \;\; 0.4853]^T \times 10^{-3}$
4	$f(x_1,x_2) = (x_1+2x_2-7)^2 + (2x_1+x_2-5)^2$	$[1 \;\; 3]^T$	0	$[-0.053 \;\; -0.661]^T \times 10^{-3}$

Learning

You will learn the following by completing this assignment,

- Implementation of Steepest descent, Newton and Quasi Newton methods for minimizing a function.
- Implementation of back-tracking approach for finding step length in the inexact line search method.

Learning is fun. Best of Luck!