IOE 511/MATH 562: Homework № 3

Due: 2/22 (by 11:59 EST), PART 1

PART 1

1. Consider the following strongly convex quadratic function

$$f(x) = \frac{1}{2}x^T Ax + b^T x + c,$$

where $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$.

   (a) Show that Newton's method with a unit step size, i.e., $x_{k+1} \leftarrow x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$, converges in a single iteration from any starting point $x_0 \in \mathbb{R}^n$. Please be precise with your answer. What is x^*? Is Newton's method well-defined?

   (b) In class, we proved a local quadratic convergence result for Newton's method. Does your answer to part (a) violate the conditions of the theorem we proved in class? Please be precise with your answer. Comment on the neighborhood and the rate.

2. Show that if $0 < c_2 < c_1 < 1$, there may be no step lengths that satisfy the Wolfe conditions, i.e.,

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1\alpha_k \nabla f(x_k)^T d_k \tag{1}$$
$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k. \tag{2}$$

3. Show that, if $\nabla f(x_k)^T d_k \neq 0$, the curvature condition (second Wolfe condition, (2)) implies

$$s_k^T y_k > 0,$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Note: $c_2 \in (0, 1)$.

PART 2 **Due:** 3/8 (by 11:59 EST), PART 2

MATLAB SECTION OF HOMEWORK

In this and all future exercises that involve coding, you will need to upload your Matlab files to Canvas, while continuing to upload the pdf file with your written answers. In addition to reading your pdf file and grading your answers to the questions posed in the exercise, we may check correctness of your code by running it, possibly on a different set of inputs than you were asked to explore in the homework. **Comments below apply to this and all future homeworks that ask for Matlab implementations, so keep this cover page handy**.

There will be **precise instructions given for how to name your Matlab files and the functions implemented in them, and what functionality they should have**. You must follow these instructions exactly to receive credit. The implementation part of your work will be graded by a computer script; if you deviate from the instructions even in a small way, the script code will fail and you will not receive points for this part of the problem. A `framework` Matlab file can be found on Canvas under the `Code` folder. You are encouraged to use this template code for the homeworks and project, however, this is not required. **What is required is that the inputs and outputs of your main functions are consistent with those required, and that the main function is named appropriately;** see below for details.

Many people use *print statements, pauses and other mechanisms* in execution as they debug and test their code. Once you have the version of your code you are ready to submit, remove all of them from your code. When executed, your Matlab function should only print to the screen or otherwise display what you have been asked to print/display/output, and should not pause mid-execution.

Please create a zip file and upload all your code in one file for each Homework. The file name should be: `LastName_FirstName_Homework#.zip`. Note also that, in addition to writing and executing some code, each implementation exercise will ask questions and request that you report specific results based on your computation. These results and answers to these questions **must be included in the pdf file of your solutions**; it is not sufficient for you to submit code that, when executed, produces those results (i.e., the grader of the homework should not have to run your code to produce the results you were asked to report yourself).

One final comment... **make sure you comment your code!**

Some tips/suggestions for coding/debugging

- Comment your code!
- Printing output is one of the best ways to debug code
- Use unit test to check all components (no matter how small) of your code
- Start small and simple
- *Why are bugs/issues so hard to find?* Usually, it bugs/issues are small/tiny (e.g., a plus sign should have been a minus sign), and hard to find since we do not expect such a simple part of the code to be wrong. From personal experience, bugs are either where you least expect to find them or right in front of you in the simplest operations of the code.
- And, of course, comment your code!

4. In this section of the homework, you will *investigate the performance of several unconstrained optimization methods that we have studied in class on three unconstrained optimization problems*. **The goal is to understand the merits and limitations of each method**.

   Algorithms:

   1. Gradient Descent (with a backtracking line search);
   2. Newton's Methods (with a backtracking line search, and modification, see end of document);
   3. BFGS ($H_0 = I$; skip update if $s_k^T y_k$ is not sufficiently positive, i.e., skip if $s_k^T y_k < \epsilon \|s_k\|_2 \|y_k\|_2$);
   4. LBFGS (with memory $m \in \{2, 5, 10\}$; $H_k^{(0)} = I$; skip update if $s_k^T y_k$ is not sufficiently positive).

   See end of document and your notes for more details about the methods.

   Problems:

   1. Rosenbrock:

   $$f(x) = (1 - w)^2 + 100(z - w^2)^2, \text{ where } x = [w\ z]^T \in \mathbb{R}^2.$$

   Starting points: $x_0 = [1.2\ 1.2]^T$, $x_0 = [-1.2\ 1]^T$. Note: $x^* = [w^*\ z^*]^T = [1\ 1]^T$.

   2. Function 2:

   $$f(x) = \sum_{i=1}^{3} (y_i - w(1 - z^i))^2, \text{ where } x = [w\ z]^T \in \mathbb{R}^2,$$

   where $y = [1.5\ 2.25\ 2.625]^T$. Starting point: $x_0 = [1\ 1]^T$. Note: the term involving z takes z to the power of i.

   3.. Function 3:

   $$f(x) = \frac{\exp(z_1) - 1}{\exp(z_1) + 1} + 0.1 \exp(-z_1) + \sum_{i=2}^{n} (z_i - 1)^4, \text{ where } x = [z_1\ z_2\ \cdots\ z_n]^T \in \mathbb{R}^n,$$

   where $n \geq 2$. Starting point: $x_0 = [1\ 0\ \cdots\ 0]^T$. For this problem let $n \in \{2, 10, 100, 1000\}$.

   Constants: Use the following set of constants: $\bar{\alpha} = 1$; $c_1 = 10^{-4}$; $\tau = 0.5$; max_iters $= 1000$; $\epsilon = 10^{-6}$, $\beta = 10^{-6}$.

   Termination Conditions: You will implement two types of termination conditions. The first is guided by the theory (i.e., gradient is small at minimizers)

   $$\|\nabla f(x_k)\|_{\text{inf}} \leq \epsilon \max\{\|\nabla f(x_0)\|_{\text{inf}}, 1\}.$$

   The second condition is to make sure that your code does not run forever: $k <$ max_iters, where k is the iteration counter.

   Deliverables:

[1] Matlab implementation of the all methods. Similar to the code framework provided (but not necessarily using the framework) your main code (`optSolver` in the provided) should take as input: *problem*, *method*, *options*, and ouput: *final iterate* and *function value* of a given method on a given problem, i.e.,

```
[x,f]=optSolver_LastName_FirstName(problem,method,options).
```

It is of paramount importance that you name your Matlab function as described above. Note, the inputs *problem*, *method*, *options* are `structs` in Matlab. See framework code and https://www.mathworks.com/help/matlab/ref/struct.html for more information.

[2] A short report of your findings. Guidance provided below.

Tasks & Guiding questions:

[1] Derive the formulae for the gradient and Hessian of the two problems.

[2] Implement functions that given x (for this assignment, you can hard-code the parameters of the problems, e.g., y in Function 2) computes the function value, gradient and Hessian of the objective functions.

[3] Implement the algorithms mentioned above. The code should be implemented such that each method can be run from a main function (e.g., a function similar to `optSolver`) with **exactly the same inputs and outputs as the framework code, and named as specified above**.

[4] Prepare a short report describing the behavior of the different algorithms on each of the problems. Use the following as guiding questions:

- Does the method converge to a solution (if it fails, explain why)?
- For the modified Newton method, were there iterations in which the Hessian was modified?
- For the BFGS method, were there iterations in which the update was skipped?
- What seems to be the convergence rate (linear, superlinear, ...)? Base your answer on the output, not on what you might think it should be.
- What can you say about the computation time?

Hint: It is very easy to make mistakes when coding derivatives; much easier than getting them right the first time (in my experience). As a debugging tool, you can use techniques from *Section 8.1, Numerical Optimization* to compute approximations of the derivatives (e.g., finite difference approximations to the derivatives) at different points and compare those to the results of your code.

Method details:

Algorithm 1 Backtracking Line Search (*Numerical Optimization*, Chapter 3, Algorithm 3.1)

Inputs: $\bar{\alpha} > 0$, $\tau \in (0,1)$, $c_1 \in (0,1)$

Set $\alpha \leftarrow \bar{\alpha}$
repeat until $f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f(x_k)^T d_k$
    $\alpha \leftarrow \tau \alpha$
end (repeat)
Terminate with $\alpha_k = \alpha$

Modified Newton's Method (subroutine to compute η_k):

Algorithm 2 Modified Newton's Method (subroutine to compute η_k)

(*Numerical Optimization*, Chapter 3, Algorithm 3.3)

Inputs: A (matrix), $\beta > 0$

if $\min_i A_{ii} > 0$, set $\eta_0 \leftarrow 0$
else set $\eta_0 \leftarrow -\min_i A_{ii} + \beta$
end if
for $k = 0, 1, 2, \ldots$
    Attempt to apply the Cholesky algorithm to obtain $LL^T = A + \eta_k I$
    if the factorization is completed successfully
        stop and **return** L
    else set $\eta_k \leftarrow \max\{2\eta_k, \beta\}$
    end (if)
end (for)
Terminate with L (or A and η_k)

In Matlab, you can use `[R,p] = chol(B);` to compute the Cholesky factorization of the matrix B. See https://www.mathworks.com/help/matlab/ref/chol.html for more details.

L-BFGS two-loop recursion:

Algorithm 3 L-BFGS two-loop recursion (*Numerical Optimization*, Chapter 7, Algorithm 7.4)

Inputs: m (memory length), H_k^0 (initial Hessian approximation), $\{s_{k-m}, \ldots, s_{k-1}\}$ and $\{y_{k-m}, \ldots, y_{k-1}\}$ (curvature pairs)

Set $q \leftarrow \nabla f(x_k)$
for $i = k-1, k-2, \ldots, k-m$
    Set $\alpha_i \leftarrow \rho_i s_i^T q$ (where $\rho_i = 1/(s_i^T y_i)$)
    Set $q \leftarrow q - \alpha_i y_i$
end (for)
Set $r \leftarrow H_k^0 q$
for $i = k-m, k-m+1, \ldots, k-1$
    Set $\beta \leftarrow \rho_i y_i^T r$
    Set $r \leftarrow r + s_i(\alpha_i - \beta)$
end (for)
Terminate with result $H_k \nabla f(x_k) = r$