Math 4650/MSSC 5650 Final Exam, Spring 2023

Instructor: Greg Ongie

Due: Fri., May 12, 11:59 PM

Instructions:

1. You must explain all reasoning. It is not sufficient to just write the correct answer. Correct answers with no justification will receive no credit.

2. You must submit your completed exam to the D2L dropbox by 11:59pm on Friday, May 12. No late work will be accepted.

3. To speed up grading, **submit your exam as one ".pdf" file or one MS Word ".docx" file**. All code and plots need to be included in the single document.

4. **RULES FOR OUTSIDE SOURCES**
   It is permissible to use:

   (a) The textbook.

   (b) All resources on D2L, including posted lecture notes and supplemental notes, and HW solution guides.

   (c) *Your* course notes.

   (d) The internet for base-level questions: MATLAB syntax, basic definitions of terms, etc.

   (e) Me, via email, for clarification on the problem statement, but I will not give out hints for any of the problems.

   It is NOT permissible to consult:

   (a) Any classmates, other students, other professors, etc.

   (b) Anybody else's course notes (it is okay if you had previously borrowed someone else's notes because you missed a day, etc).

   (c) The internet for anything remotely approaching the actual question asked.

   - For example, it is okay to search the internet for "How to compute eigenvalues in MATLAB", but it is not okay to search the internet for the actual question.
   - If you are in doubt about what is okay, email me!

Any instances of plagiarism, working with classmates, or other cheating, will result in a score of 0 for the entire final exam.

This exam has five problems and is worth 100 points.

- **Math 4650 Students**: Complete the first four problems. Each problem is worth 25 points. Problem 5 may be attempted for up to 10 points extra credit.
- **MSSC 5650 Students**: Complete all five problems. Each problem is worth 20 points.

Problem 1. Find all critical points of the following functions and classify them according to whether they are local/global minimizers, local/global maximizers, or saddle points.

(a) $f(x, y) = 8x + 12y + x^2 - 2y^2$

(b) $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$

(c) $f(x, y) = -\ln(1-x-y) - \ln(x) - \ln(y)$ over its domain $\{(x, y) : x > 0, y > 0, x+y < 1\}$.

Problem 2. In `problem2.m` you are given a "mystery function" $f : \mathbb{R}^{500} \to \mathbb{R}$ to minimize. This script loads MATLAB functions `f` and `grad` that return values and gradients of f, respectively. The mystery function f has the following properties:

- f is continuously differentiable.
- f is L-smooth, with $L = 4$.
- The minimum of f is 0, i.e., $0 = \min_{\mathbf{x} \in \mathbb{R}^{500}} f(\mathbf{x})$.

Your task is to find the best approximate minimizer `x` you can using *any algorithm of your choosing* given a fixed budget of 1000 gradient evaluations. Here "best" is measured by how close the final value of `f(x)` is to 0 (the minimum of f).

Ground rules:

- The initial iterate must be $\mathbf{x}_0 = \mathbf{0}$ (in MATLAB `x = zeros(500,1)`).
- You need to verify that `grad` was called at most 1000 times. This is checked at the end of the script.
- There is no limit on how many times `f` can be called.

You will be graded according to the following criteria:

$0.4990 >$ `f(x)` ≥ 0.0100	up to 60% points possible
$0.0100 >$ `f(x)` ≥ 0.0075	up to 80% points possible
$0.0075 >$ `f(x)` ≥ 0.0050	up to 100% points possible
$0.0050 >$ `f(x)`	up to 100% points possible, plus 5 points extra credit

To get full credit your write-up must include:

- A printout of your MATLAB code.
- Final values of `f(x)` and `grad('count')` (these are automatically output by the script)
- A log plot of the cost (this is automatically output by the script)
- A short discussion (2-4 sentences) on why you chose the algorithm you did and the parameter choices you settled on.

Problem 3. Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ given by $f(x, y) = x^4 + y^4$. Note f has a unique global minimum at $(x, y) = (0, 0)$. This problem will have you examine the iterates of gradient descent with exact line search for minimizing this function.

(a) Derive a formula for the *exact line-search* step-size, i.e., given any point $(x, y) \in \mathbb{R}^2$ with $(x, y) \neq (0, 0)$ find a formula for the step-size t^* given by

$$t^* = \arg\min_{t \geq 0} f((x, y) - t\nabla f(x, y))$$

(b) Implement gradient descent with exact line search in MATLAB for this function by modifying the provided script `problem3.m`. Use the following settings: start from the initial iterate $(x_0, y_0) = (0.5, 1)$, and use the exit condition $\|\nabla f(x_k, y_k)\| \leq 10^{-16}$ (these are the default settings in the script).

In your write-up, include the following:

- A derivation of the exact line-search rule in part (a).
- A printout of your MATLAB code from part (b).
- A plot of the trajectory of the iterates output (code to make this plot is included in `problem3.m`).
- A few sentences (2-3) describing the trajectory of the iterates and a possible explanation of why they look the way they do.

Problem 4. The *Gauss-Newton* method is an approximation to Newton's method for solving non-linear least squares problems of the form

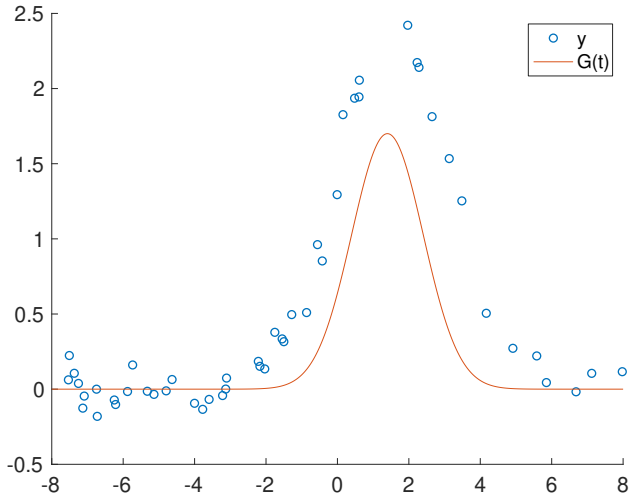
$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{g}(\mathbf{x}) - \mathbf{y}\|^2$$

where $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function and $\mathbf{y} \in \mathbb{R}^m$ is a constant vector. The Gauss-Newton iterations are given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}_\mathbf{g}(\mathbf{x}_k)^\top \mathbf{J}_\mathbf{g}(\mathbf{x}_k) + \varepsilon \mathbf{I})^{-1} \mathbf{J}_\mathbf{g}(\mathbf{x}_k)^\top (\mathbf{g}(\mathbf{x}_k) - \mathbf{y})$$

where $\mathbf{J}_\mathbf{g}(\mathbf{x}_k)$ is the Jacobian of $\mathbf{g}$ evaluated at the kth iterate $\mathbf{x}_k$, and $\varepsilon > 0$ is a parameter close to zero that ensures the matrix being inverted is nonsingular.

In this problem you will use the Gauss-Newton method to find the best fit of a Gaussian curve of the form $G(t) = a\exp(-b(t-c)^2)$ to scattered data points $(t_1, y_1), ..., (t_m, y_m)$. Note $G(t)$ depends on three parameters: a is the height of the Gaussian, b determines its width, and c determines its center. An illustration is shown below, where the data points (t_i, y_i) are shown in blue, and the curve $G(t)$ is shown in red.



We can set this up as a non-linear least-squares problem where the optimization variable is $\mathbf{x} = (a, b, c) \in \mathbb{R}^3$ and $\mathbf{g}(a, b, c) \in \mathbb{R}^m$ is the vector of evaluations of the Gaussian curve with parameters a, b, c at the points $t_1, t_2, ..., t_m$, i.e.,

$$\mathbf{g}(a, b, c) = \begin{bmatrix} a\exp(-b(t_1 - c)^2) \\ a\exp(-b(t_2 - c)^2) \\ \vdots \\ a\exp(-b(t_m - c)^2) \end{bmatrix},$$

and $\mathbf{y} = (y_1, ..., y_m)$ is the vector of data values to be fit.

(a) Determine the Jacobian matrix $\mathbf{J}_\mathbf{g}(a, b, c)$ for the function $\mathbf{g}(a, b, c)$ above (Note: this matrix has dimensions $m \times 3$).

(b) Using the script `problem4.m` as a staring point, define the Jacobian of $\mathbf{g}$ in MATLAB and implement the Gauss-Newton algorithm. You may set $\varepsilon = 10^{-8}$.

(c) Starting from the initialization $(a, b, c) = (0, 0, 0)$, run 20 iterations of the Gauss-Newton method to find the best fit parameters a, b, c.

(d) Plot the resulting Gaussian curve $G(t)$ with the optimal values of a, b, c on top of the scattered data.

In your write-up, include the following:

- A derivation of the Jacobian found in part (a).
- A printout of your MATLAB code from part (b).
- The best-fit parameters a, b, c that you found.
- The plot requested in part (d).

Problem 5 (MSSC)**.**

(a) A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is called *convex* if $\nabla^2 f(\mathbf{x}) \succeq 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Prove that if f is convex then

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \quad \text{for all} \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

(Hint: Start from Taylor's approximation theorem – see Theorem 1.24 in Beck)

(b) Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, *convex*, and L-smooth. Assume that f has at least one global minimizer $\mathbf{x}^* \in \mathbb{R}^n$ and let $f^* = f(\mathbf{x}^*) = \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$. Define $\{\mathbf{x}_k\}_{k \geq 0}$ to be the iterates obtained by running gradient descent with constant stepsize $t = 1/L$ starting from any initial point $\mathbf{x}_0 \in \mathbb{R}^n$. Show that if the iterates $\{\mathbf{x}_k\}_{k \geq 0}$ are bounded[1] then $f(\mathbf{x}_k) \to f^*$ as $k \to \infty$.

(Hint: You may use the fact that under these assumptions we have $\|\nabla f(\mathbf{x}_k)\| \to 0$ as $k \to 0$; see Theorem 4.25 in Beck.)

[1] i.e., there exists a $B > 0$ such that $\|\mathbf{x}_k\| \leq B$ for all $k \geq 0$