STAT 280: OPTIMIZATION
SPRING 2022
PROBLEM SET 4

For the parts that require coding, you may use any software or programming languages you like but please present your source codes and results in a way that is comprehensible to someone who is unfamiliar with that program (e.g., comment your codes appropriately, present your results using tables and graphs). I recommend using NumPy, Mathematica, Matlab, R so that you don't have to code things from scratch.

1. (a) Find all stationary points of the cubic polynomial

$$f(x, y) = x^3 + y^3 - 3x - 12y + 20.$$

Indicate which are the local maximizers and local minimizers.

(b) Prove that for any $x \geq 0$, $y \geq 0$, we always have

$$\frac{x^2 + y^2}{4} \leq e^{x+y-2}.$$

(c) Let $A \in \mathbb{S}^n$, $\mathbf{b} \in \mathbb{R}^n$, and $c \in \mathbb{R}$. Show that $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T} A\mathbf{x} + \mathbf{b}^\mathsf{T}\mathbf{x} + c$$

has a unique global minimizer iff $A \succ 0$. What is it?

(d) Let $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Show that $\mathbf{x}_*$ is a global minimizer of $\|A\mathbf{x} - \mathbf{b}\|^2$ iff $\mathbf{x}_*$ is a solution to $A^\mathsf{T} A\mathbf{x} = A^\mathsf{T}\mathbf{b}$.

(e) Let $A \in \mathbb{R}^{m\times n}$ be of rank n, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, $d \in \mathbb{R}$, and $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{c}^\mathsf{T}\mathbf{x} + d > 0\}$. Show that the global minimizer of $f : \Omega \to \mathbb{R}$,

$$f(\mathbf{x}) = \frac{\|A\mathbf{x} + \mathbf{b}\|^2}{\mathbf{c}^\mathsf{T}\mathbf{x} + d}$$

is given by

$$\mathbf{x}_* = (A^\mathsf{T} A)^{-1}(-A^\mathsf{T}\mathbf{b} + t\mathbf{c})$$

where t is a solution to a quadratic equation. Find t in terms of $A, \mathbf{b}, \mathbf{c}, d$.

2. Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(x, y) = \frac{1}{2}(ax^2 + by^2)$$

where $a, b > 0$. We will apply steepest descent with exact line search to f with the initial point $\mathbf{x}_0 = (x_0, y_0) = (b, a)$. (Note: In case it is not clear, you are supposed to do this problem 'by hand'. No coding required.)

(a) Show that f is strongly convex on $\mathbb{R}^2$. Find the global minimizer $\mathbf{x}_*$ and global minimum $f(\mathbf{x}_*)$.

(b) Show that steepest descent with exact line search will yield step size

$$\alpha_k = \frac{2}{a + b},$$

Date: May 15, 2022 (Version 1.0); due: May 27, 2022.

iterates

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \qquad x_k = \left(\frac{b-a}{a+b}\right)^k b, \qquad y_k = \left(\frac{a-b}{a+b}\right)^k a,$$

and function values

$$f(\mathbf{x}_k) = \frac{ab^2 + ba^2}{2}\left(\frac{b-a}{a+b}\right)^{2k}$$

for all $k \in \mathbb{N}$.

(c) Deduce that

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}_*\|}{\|\mathbf{x}_k - \mathbf{x}_*\|} = \left|\frac{a-b}{a+b}\right|, \qquad \lim_{k\to\infty} \frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_*)|}{|f(\mathbf{x}_k) - f(\mathbf{x}_*)|} = \left|\frac{a-b}{a+b}\right|^2.$$

In other words, in this case the sequence of iterates is linearly convergent both in the usual sense ($\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_*$ linearly) and the functional sense ($\lim_{k\to\infty} f(\mathbf{x}_k) = f(\mathbf{x}_*)$ linearly).

3. Implement steepest descent method and Newton method, both with backtracking line search, for minimizing a function of the form

$$f(x_1, \dots, x_{100}) = \sum_{j=1}^{100} c_j x_j - \sum_{i=1}^{500} \log\left(b_i - \sum_{j=1}^{100} a_{ij} x_j\right).$$

Your implementation just needs to work for this specific objective function (as opposed to an arbitrary f) but it should allow for (i) arbitrary input parameters $A \in \mathbb{R}^{500\times 100}$, $\mathbf{b} \in \mathbb{R}^{500}$, and $\mathbf{c} \in \mathbb{R}^{100}$, (ii) arbitrary backtracking line search parameters $c \in (0,1)$ and $\rho \in (0,1)$, (iii) abitrary starting point $\mathbf{x}_0$ and initial step size α_0, (iv) arbitrary tolerance $\varepsilon > 0$ for the stopping conditions (i.e., $\|\nabla f(\mathbf{x}_k)\| \le \varepsilon$ for steepest descent, $\lambda_k^2/2 \le \varepsilon$ for Newton).

(a) Note that this is an unconstrained optimization problem but the domain of this function is

$$\Omega := \left\{\mathbf{x} \in \mathbb{R}^{100} : b_i - \sum_{j=1}^{100} a_{ij} x_j > 0 \text{ for all } i = 1, \dots, 500\right\}.$$

Generate A and $\mathbf{b}$ randomly in a way that ensures $\Omega \neq \varnothing$, for example

```
A = randn(500, 100); b = A*randn(100, 1) + 2*rand(500,1);
```

in Matlab/Octave/Scilab syntax). Generate $\mathbf{c}$ randomly too. Set $\alpha_0 = 1$ and generate $\mathbf{x}_0$ randomly so that $\mathbf{x}_0 \in \Omega$.

(b) Let $\mathbf{x}_*$ be the output of your implementation. Let $e_k := f(\mathbf{x}_k) - f(\mathbf{x}_*)$ be the error at the kth iteration. Plot the log of the error $\log e_k$ against k in a graph, i.e., you want to see how $\log e_k$ decreases as k increases. Why did we use a log scale? What if we instead plot the error e_k against k?

(c) Do (b) for both steepest descent and Newton methods over a range of different backtracking parameters and tolerance:

$$c = 0.01,\ 0.05,\ 0.10,\ 0.25,\ 0.50,\ 0.75,\ 0.90,$$
$$\rho = 0.05,\ 0.25,\ 0.50,\ 0.75,\ 0.95,$$
$$\varepsilon = 10^{-3},\ 10^{-5},\ 10^{-8}.$$

Do (b) *without* line search, i.e., omit the line search step from steepest descent and Newton method.

(d) Comment on your results,[1] paying particular attention to (i) the convergence rates of steepest decent and Newton methods, (ii) how the two methods depend on on different choices of c, ρ, ε and whether you do line search or not.

[1]Since your numerical experiments rely on randomly generated $A, \mathbf{b}, \mathbf{c}$, and $\mathbf{x}_0$, you should repeat them at least 10 times just to be sure that what you observed is not a fluke. However, just present one set of graphs to support your conclusions.

4. We will apply Newton method to compute the inverse A^{-1} of an invertible matrix $A \in \mathbb{R}^{n\times n}$.

(a) Consider the function $g(X) = X^{-1}$ defined for invertible $n \times n$ matrices X. Show that the derivative of g at X is given by

$$[Dg(X)](H) = -X^{-1}HX^{-1}.$$

(b) Show that Newton method may be applied to an appropriate function to obtain the following iteration for computing the inverse of an invertible matrix $A \in \mathbb{R}^{n\times n}$

$$X_{k+1} = X_k(2I - AX_k). \tag{4.1}$$

(*Hint*: Emulate the univariate Newton method for computing reciprocal in Homework **1**, Problem **5**(c).) Note that like the univariate version this algorithm requires only addition and multiplication of matrices.

(c) Show that if we define error at step k by $E_k = I - AX_k$ (note that this vanishes exactly when $X_k = A^{-1}$), then

$$E_{k+1} = E_k^2 = E_{k-1}^4 = \cdots = E_0^{2^{k+1}}.$$

In other words, if (4.1) converges, then the convergence is quadratic.

(d) Implement the algorithm in (b) with initialization $X_0 = \alpha A^{\mathsf{T}}$, $0 < \alpha < 2/\|A\|^2$, and with a simple stopping criteria (e.g., stop when $\|X_{k+1} - X_k\|$ or $\|E_k\|$ is small).

(i) Compare the result X_* obtained by your implementation for 2×2 matrices A with random integer entries and for a 10×10 diagonal matrices A with random rational entries with the actual A^{-1}, which you know analytically. Check the accuracy of your implementation by observing the values of $\|X_* - A^{-1}\|$ (this is called the *forward error*, note that you can compute this only if you already know A^{-1}).

(ii) Compare the result X_* obtained by your implementation for randomly generated $n \times n$ matrices A with the result Y_* obtained by calling the matrix inversion function of the software you use. Do this for $n = 10, 10^2, 10^3$. Check the accuracy of your implementation by comparing the values of $\|I - AX_*\|$ and $\|I - AY_*\|$ (this is called the *backward error*, note that you can compute this even if you do not know A^{-1}).

Side note: In fact invertibility is not a requirement and you can even have a rectangular matrix $A \in \mathbb{R}^{m\times n}$. Initializing (4.1) by $X_0 = \alpha A^{\mathsf{T}}$ for any $0 < \alpha < 2/\|A\|^2$ produces a sequence that converges to $X_* = A^{\dagger} \in \mathbb{R}^{n\times m}$, the *Moore–Penrose inverse* of A. I recommend Stat 309 if you're interested to find out more.