STAT 280: OPTIMIZATION
SPRING 2017
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, Scilab, or Octave (the last two use Matlab syntax but are open source and freely downloadable) so that you don't have to code things from scratch.

1. 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 $\mathbb{R}^n$. 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

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

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

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

where $\|\cdot\|$ denotes the 2-norm. 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).

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

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

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)

Date: May 12, 2017 (Version 1.0); due: May 25, 2017.

abitrary starting point $\mathbf{x}_0$ and initial step size α_0, (iv) arbitrary tolerance $\varepsilon > 0$ for the stopping conditions (i.e., $\|\nabla f(x_k)\| \le \varepsilon$ 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

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

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:

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

(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.

3. In the lectures, we saw that Newton method, when applied for computing square root $A^{1/2}$ of a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ (in fact any $n \times n$ complex matrix), yields the following iteration:

$$X_kP_k + P_kX_k = A - X_k^2,$$
$$X_{k+1} = X_k + P_k.$$

(a) Using induction or otherwise, show that if $X_0 \in \mathbb{R}^{n\times n}$ commutes with A, i.e., $AX_0 = X_0A$, then X_k commutes with A for all $k = 0, 1, 2, \dots$. Hence deduce that the Newton iteration above may be written as

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A), \tag{3.1}$$

assuming that X_k is invertible.

(b) Show that if $X \in \mathbb{R}^{n\times n}$ commutes with A, then it commutes with $A^{1/2}$.

(c) Show that if we define error at step k by $E_k = X_k - A^{1/2}$, then

$$E_{k+1} = \frac{1}{2}X_k^{-1}E_k^2,$$

assuming that X_k is invertible and $\|\cdot\|$ is a submultiplicative matrix norm. Hence deduce that convergence is quadratic if there exists $M > 0$ such that $\|X_k^{-1}\| \le M$ for all $k = 0, 1, 2, \dots$.

[1]Since your numerical experiments depend 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 obtain an analogue of (3.1) for computing the inverse A^{-1} of an invertible matrix $A \in \mathbb{R}^{n\times n}$ (in fact the Moore–Penrose inverse of any $m \times n$ complex matrix).

(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

$$[Dg(X)](Y) = -X^{-1}YX^{-1}.$$

(Hint: Use the expansion in lecture $(I - A)^{-1} = I + A + A^2 + \cdots$ alongside our usual first order approximation trick $g(X + Y) \approx g(X) + [Dg(X)](Y)$ to get the expression.)

(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 $A \in \mathbb{R}^{n\times n}$

$$X_{k+1} = X_k(2I - AX_k). \tag{4.2}$$

(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 (4.2) converges, then the convergence is quadratic. In fact one can show that for any $A \in \mathbb{R}^{m\times n}$, initializing (4.2) by $X_0 = \alpha A^{\mathsf{T}}$ for any $0 < \alpha < 2/\|A\|_2^2$ produces a sequence that converges to the Moore–Penrose inverse $A^{\dagger}$. In particular if $A \in \mathbb{R}^{n\times n}$ is invertible, then $\lim_{k\to\infty} X_k = A^{-1}$.

(d) Implement the algorithm in (b) with the initialization described in (c) with a simple stopping criteria (e.g. stop when $\|X_{k+1} - X_k\|_F$ or $\|E_k\|_F$ 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 $\|X_* - A^{-1}\|_F$ (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 $n = 10, 10^2, 10^3, 10^4$. Check the accuracy of your implementation by comparing the values of $\|I - AX_*\|_F$ and $\|I - AY_*\|_F$ (this is called the *backward error*, note that you can compute this even if you do not know A^{-1}).