CS726, Spring 2018

Homework 2 (due Wednesday 2/21/18 at 6:00pm)

No late submissions accepted, except by prior arrangement with the instructor.

1. Give an example of a sequence $\{x_k\}$ of positive real numbers, that decreases to zero Q-superlinearly but not Q-quadratically. (Refer to Appendix A.2 of **Numerical Optimization, 2nd Ed** on convergence rates.)

2. Find positive values of $\bar{\epsilon}$, γ_1, and γ_2 such that the Gauss-Southwell choice $d^k = -[\nabla f(x^k)]_{i_k}$, where $i_k = \arg\max_{i=1,2,\dots,n} |[\nabla f(x^k)]_i|$ satisfies the conditions on d_k required on d_k in Section 3.4 of the notes.

3. Write a Matlab code to apply various first-order methods to the convex quadratic function $f(x) = (1/2)x^T Ax$, where the positive definite matrix A is generated by the Matlab code fragment below (modified where necessary) to have eigenvalues in a prescribed range $[\mu, L]$, with $0 < \mu \leq L$. ($x^* = 0$ is obviously a global minimizer.) In all cases, start from a point `x0` generated by the Matlab command `randn(n,1)`, and run until $f(x^k) - f(x^*) \leq \epsilon$, where $\epsilon = 10^{-6}$. Implement the following methods:

   - Steepest descent with $\alpha_k \equiv 1/L$.
   - Steepest descent with exact line search.
   - Nesterov's optimal method from Chapter 4 of the notes.
   - Conjugate gradient method from p.112 of **Numerical Optimization, 2nd Ed** .

   (a) After you have debugged, make a version of your code called `firstorder.m`, that does the 10 trial runs and averages the number of iterates required for convergence, that *does not make a plot, and that prints nothing except four lines of output.* Use these lines of code as the last four lines of your Matlab file, to generate the required output:

```
fprintf(1,' steepest descent - fixed steps : %7.1f\n', av_sd);
fprintf(1,' steepest descent - exact steps : %7.1f\n', av_sde);
fprintf(1,' Nesterov                       : %7.1f\n', av_nest);
fprintf(1,' conjugate gradient             : %7.1f\n', av_cg);
```

where `av_sd`, `av_sde`, `av_nest`, `av_cg` are the average numbers of iterates (over the 10 random trials) for each of the four algorithms, computed by your code.

(b) Draw a plot of the convergence behavior on a typical run, plotting iteration number against $\log_{10}(f(x_k) - f(x^*))$. Use a single plot, with different colors for the different algorithms. You can use the "legend" command in Matlab to indicate which line belongs to which algorithm, e.g. `legend('SD:const','SD:exact','Nesterov','CG')`.

(c) Discuss your results. In particular, say whether the observed convergence behavior is consistent with the rates that we proved in class for some of these methods.

Here's the Matlab code fragment to generated the matrix A.

```
mu=0.01; L=1; kappa=L/mu;
n=100;
A = randn(n,n); [Q,R]=qr(A);
D=rand(n,1); D=10.^D; Dmin=min(D); Dmax=max(D);
D=(D-Dmin)/(Dmax-Dmin);
D = mu + D*(L-mu);
A = Q'*diag(D)*Q;
x0 = randn(n,1); % use a different x0 for each trial
```

Hand-in Instructions:

- Hand in your code `firstorder.m`, and its output (detailed in (a) and (b)).
- Hand in your discussion in part (c) and the answers to the other questions as a pdf file, preferably typeset, but at least neatly written out.
- Submit through Canvas.