CS726, Spring 2018

Homework 6 (due Friday 4/13/18 6:00pm)

Please submit your answers in the order listed below.

Hand in your code and results, and answers to the questions, via Canvas. Give your files the names `EBLS.m`, `CG_FR.m`, `CG_PRplus.m`, `SteepDescent.m`, and `comments.txt`. (The last file should contain the outputs from your codes, and your written responses to the questions about the code.)

1. Question 5.11 from the text.
2. Question 6.3 from the text.
3. Question 6.4 from the text.
4. Write efficient codes for the Fletcher-Reeves nonlinear conjugate gradient algorithm, the Polak-Ribière method with the modification (5.45), and the Steepest Descent method. For all methods, use the `EBLS.m` routine for extrapolation-bracketing line search, that you coded in an earlier assignment, to find an approximate α_k that satisfies the strong Wolfe conditions. Modify EBLS slightly so that its calling sequence is as follows:

   ```
   function [x,alpha,nf,ng] = EBLS(fun, x, d, alpha_start)
   ```

   where the new output parameters `nf` and `nf` return the number of function and gradient evaluations, respectively, that were performed inside EBLS.

   Terminate your nonlinear CG and steepest descent codes when the following condition is satifsied:

   $$\|\nabla f(x_k)\|_\infty \leq \texttt{nonCGparams.toler} * (1 + |f(x_k)|),$$

   or else when the number of iterations exceeds `nonCGparams.maxit`, whichever comes first.

   In your submitted codes, use the following parameter settings for StepSize:

   $$c_1 = 10^{-2}, \quad c_2 = 0.3, \quad \text{maxit} = 100,$$

   and set `alpha_start` to 1 at each call to EBLS.

   Test your codes using `nonlinearcg.m`. Note that the function to be minimized is Powell's singular function, coded as `xpowsing.m`. (These files are supplied on Canvas.)

   Try modifying the line search parameters (c_1, c_2, and `alpha_start`) to see if you can improve the performance of the algorithms, and comment on your experiences.

   Further Details: You Matlab program `SteepDescent.m` should have the following first line:

   ```
   function [inform,x] = SteepDescent(fun,x,sdparams)
   ```

   The inputs `fun` and `x` are as described in Homework 3, while `sdparams` is the following structure:

```
sdparams = struct('maxit',100000,'toler',1.0e-5);
```

The output `inform` is a structure containing two fields: `inform.status` is 1 if the gradient tolerance is achieved and 0 if not, while `inform.iter` is the number of steps taken. The output `x` is the solution structure, with point, function, and gradient values at the final value of x_k.

Note that the global variables `numf` and `numg` are reported by the program `nonlinearcg.m` and incremented by the function evaluation routines. Be sure to set these to zero at the start of each of your routines for nonlinear CG and steepest descent, and to update them after each call to `EBLS`.

Do not print out the value of x at each iteration!

Your codes for `CG_PRplus.m` and `CG_FR.m` should have similar input arguments and similar outputs to `SteepDescent.m`.