CS726, Fall 2016

Homework 5 (due Monday 11/7/16 in class)
Please submit your answers in the order listed below.

Hand in hard copies of your code and results, and answers to the questions, in class on the due date. In addition, put exactly four files in the dropbox HW5 on the `learn@UW` site. Give these files the names `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.)

The piazza site contains a Matlab routine called `StepSize.m` which implements the line-search routine Algorithm 3.5 in the text, which includes the "zoom" procedure of Algorithm 3.6. The following specifics are used in this implementation

- $\alpha_{\max}$ is the Matlab-defined quantity "realmax".
- In the extrapolation step of Algorithm 3.5, we set $\alpha_{i+1} = 3\alpha_i$.
- The interpolation step of the zoom procedures performs quadratic interpolation using the values of ϕ and ϕ' at α_{lo} and the value of ϕ at α_{hi}. (An appropriate modification of formula (3.58) is used.) Safeguarding ensures that α_j is not within a distance $0.2|\alpha_{\text{hi}} - \alpha_{\text{lo}}|$ of the endpoints.

The header line of StepSize is

```
function [alfa,x] = StepSize(fun, x, d, alfa, params)
```

where the input parameters are:

fun - a pointer to a function (such as obja, objb, objc)

x - a strucure with three fields x.p, x.f, and x.g in which x.p contains the point x, while x.f and x.g contain the function and gradient values corresponding to x. On input, x.p is set to the starting point values `x = struct('p', [-1.2, 1.0]);` while x.f and x.g are set to the corresponding function and gradient values.

d - a vector containing the search direction

alfa - the initial value of the step length (set it to 1 for this project)

params - a structure containing parameter values for the test, for example
`params struct('c1',0.01,'c2',0.5,'maxit',100);`

The routine should call `fun` to evaluate the objective function and gradient, as follows:

```
x.f = feval(fun,x.p,1);
```

and

```
x.g = feval(fun,x.p,2);
```

respectively.

The output parameters of StepSize are

alfa - the value α_k satisfying the strong Wolfe conditions

x - on output, x.p contains the final vector $x_{k+1} = x_k + \alpha_k d_k$ while x.f and x.g contain the function and gradient values at x_{k+1}.

The purpose of returning these values is to avoid computing them again before the next call to the StepSize routine.

1. Question 5.11 from the text.
2. Question 6.4 from the text.
3. 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 `StepSize.m` routine that you coded in earlier assignments to find an approximate α_k that satisfies the strong Wolfe conditions. Terminate 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:

   ```
   params = struct('c1',0.01, 'c2',0.3, 'maxit',100)
   ```

   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 also supplied on piazza.)

   Try modifying the line search parameters to see if you can improve the performance of the algorithms, and comment on your experiences in the file `comments.txt`.

   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 above, while `sdparams` is the following structure:

   ```
   sdparams = struct('maxit',10000,'toler',1.0e-4);
   ```

   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 `SteepDescent.m`.

   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`.