CPSC 406, Term I, 2016
Assignment 2, due Thursday, October 6

Please show all your work: provide a hardcopy of the entire assignment (including plots and programs); in addition, e-mail your MATLAB programs to `cs406@ugrad.cs.ubc.ca`. When e-mailing your programs, include your name and student ID in the message's title.

Do not e-mail a complete assignment: only the programs.

Answer at most 4 out of the following 5 questions. Each is worth 25 marks for a maximum total of 100.

1. Consider the function
$$f(\mathbf{x}) = \frac{1}{2}x_1^2 - \frac{1}{4}x_2^4 + \frac{1}{327}.$$
   (a) Find all critical points and determine for each whether it is a minimum, maximum or saddle point.
   (b) What happens if Newton's method is applied to find each of these critical points (i.e., is it expected to converge from a sufficiently close initial guess and how fast)?

2. The Rosenbrock function $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ has a unique minimizer that can be found by inspection.

   Apply Newton's method and the BFGS method, both with and without weak line search (so, in total, four runs), starting from $\mathbf{x}_0 = (0, 0)^T$. Use the stopping criterion $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < 10^{-7}(1 + \|\mathbf{x}_k\|)$ for a sueccessful termination and exceeding 2000 iterations (or just blowing up) for an unsuccessful termination.

   Compare performance.

3. Prove the following statements:
   (a) There is only one symmetric rank-one update formula, that is the SR1 method.

   [Guidance: You can start off by writing down the update as $B_{k+1} = B_k + \gamma \mathbf{v}\mathbf{v}^T$, where γ is either 1 or -1, and impose $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$. Show that $\mathbf{v}$ must be a multiple of $\mathbf{y}_k - B_k\mathbf{s}_k$, and proceed to complete the argument.]
   (b) For SR1, even if B_k is positive definite, B_{k+1} may not have the same property.
   (c) The BFGS update formula satisfies the secant equation $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$.

4. MATLAB (version R2015a) has a library function called `fminsearch`. It implements an algorithm that its documentation refers to as 'Nelder-Mead simplex direct search'. Another library function, called `fminunc`, also has the user option of not specifying the gradient (whereupon an algorithm referred to as 'quasi-Newton' is employed). Another option of `fminunc`, which does require the user to specify the gradient, uses an algorithm referred to as 'trust-region'. The first two of these three possibilities have the same user requirement, then.

   (a) Run all three programs for minimizing the function

   $$f(\mathbf{x}) = x_1^4 + x_1 x_2 + (1 + x_2)^2.$$

   Try both initial guesses $\mathbf{x}_0 = (.75, -1.25)^T$ and $\mathbf{x}_0 = (0, 0.3)^T$ for a total of 6 library solver calls. Your script may employ something like this:

   ```
   [x,fx,flag,OUTPUT] = fminsearch(@fun,x0);
   [x,fx,flag,OUTPUT] = fminunc(@fun,x0);
   options = optimoptions('fminunc','GradObj','on');
   [x,fx,flag,OUTPUT] = fminunc(@fung,x0,options);
   ```

   Record the number of iterations **and** the number of function evaluations (the latter is a better indicator of the true computational cost if f were complicated to evaluate) required for each of the three variants, available in OUTPUT.

   (b) Repeat the runs of the previous items, this time for the function

   $$f(\mathbf{x}) = \frac{1}{2}\left\{[1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2\right\}.$$

   Try both initial guesses $\mathbf{x}_0 = (8, 0.2)^T$ and $\mathbf{x}_0 = (8, 0.8)^T$ for a total of 6 solver function calls.

   Make observations and discuss.

5. (a) Continuing with the setup of Question 4, repeat similar runs for the larger problem defined by the script

   ```
   nx = 15; n = nx^2;
   H = delsq(numgrid('S',nx+2));
   xe = ones(n,1);
   b = H*xe;
   phi = .5*(x'*(H*x)) - b'*x;
   gradphi = H*x - b;
   ```

   starting with the initial guess $\mathbf{x}_0 = \mathbf{0}$ (so, a total of 3 library solver calls).

   (b) Run the two variants of `fminunc` also for $nx = 31$, and perhaps also $nx = 63$, noting that the number of unknowns is $n = nx^2$. Don't bother with the direct search method.

Observe and discuss.

How many iterations (as a function of n) do you expect Newton's method to require for this problem?