4M17 Coursework Assignment 1

Gradient based minimisation

1. Write a program that finds a local minimum of a function with a single scalar argument, starting from a single user input that specifies an initial guess for the location of the minimum. Make sure that your program brackets the minimum (i.e. identifies an interval which must contain the minimum) before attempting the sequence of iterations that lead you to the optimal location. Assume that both the function itself and its analytic derivative are available as separate subroutine calls.

   [A note on bracketing: your line search program should take as inputs the function f, its derivative f' (optional) and a starting point x_0. It should then discover automatically an interval which *must* contain a minimum, e.g. by finding three points $x_1 < x_2 < x_3$ for which $f(x_1) > f(x_2), f(x_2) < f(x_3)$.]

2. Test your program on the following function: $f(x) = x^4 \cos(1/x) + 2x^4$, and try out various starting points. Analyse carefully the convergence rate of your program.

3. Modify your program so that it accepts a function with a vector argument and search direction, and performs the one-dimensional "line search" for the minimum only along the chosen search direction. Implement the "Wolfe conditions" to be used as criteria for terminating the one-dimensional line search.

4. Using first the "Steepest Descent" and then the "Conjugate Gradients" algorithms, find the minimum of the Rosenbrock function:
$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$
   Plot the trajectory of the algorithm in 2D and analyse the convergence rate.

   [Optional: write code to implement the derivative-free Nelder-Mead algorithm (take care that all cases are covered in the decision tree which selects the next point in the simplex) and use it to minimise the Rosenbrock function]

5. Write a program that performs "Steepest Descent" and "Conjugate Gradients" iterations using *exact* line search for quadratic functions of the form
$$f(x) = \frac{1}{2} x^T A x - b^T x$$
   where $x \in \mathcal{R}^N$ is a real vector with N components, A is an $N \times N$ real symmetric matrix. Remember that for quadratic forms only, exact (i.e. not iterative) line search is possible. Try it out with matrices defined in the file `4M17_assignment1.mat` and choose b is a random real vector with N components drawn from a $U[-1, 1]$. Do use the analytic derivative of the quadratic form, but do not use any $O(N^3)$ matrix operations in your optimiser, that of course would defeat the purpose of the exercise. The `.mat` file has six matrices in it, called `A10,A100,A1000,B10,B100,B1000`. Run your algorithm using each of these matrices. Analyse the relative efficiencies of the two algorithms for $N = 10$, 100, 1000 for the A and B matrices. Measure computational cost simply by the number of function and/or derivative evaluations.

6. Repeat 5, but use your own iterative line search algorithm that you wrote earlier (and the Wolfe conditions) rather than exact line search. Compare the efficiencies and convergence rates to what you obtained in 5.

Your report

- The deadline for handing in your report on this assignment is 9am on 15 January, 2016. You need to submit an electronic copy via Moodle. Please also submit a coursework cover sheet to to David Gautrey (dpg23, IN1-07). Undergraduates: make sure that **your name does not appear in the report, or in the file name, only your Coursework Candidate Number (CCN)**. MPhil and postgraduates: make sure your name and CRSID (Raven ID, or Engineering computer username) does appear on your report.

- You are free to choose any programming language or environment for this assignment, but you will need to load the matrices in task 5 into MATLAB first, and save them in a format that your programs are able to read.

- You should include figures and *associated explanations* (no more than 200 words each) arising from tasks 2, 4, 5 and 6. **Plots without explanations will receive very few marks**. Attach the source code of all programs you have used in an appendix.
- This assignment counts as 50% of your final grade, so you should be spending roughly 10-15 hours on it. Questions on this assignment should be directed to Dr Gábor Csányi (gc121@cam.ac.uk)

October 2015
Gábor Csányi