ME 748: Optimum Design of Mechanical Elements and Systems
Spring 2007; Project-1;

Due: 7th March 2007 (2 day extension max)

This is a team project with two students per team.

Project Goals and Description of Tasks

The most popular all **unconstrained** minimization methods are **conjugate gradient** and **Quasi-Newton** schemes. We will also learn later that unconstrained algorithms can also be used for **constrained** problems, with suitable modifications.

The main purposes of this project are: (1) to understand and appreciate optimization algorithms, (2) to learn to use commercial numerical libraries for solving our optimization problems. **Most times, writing your own optimization code is not a good option, but sometimes worth considering.**

Conjugate Gradient Algorithms (Fletcher-Reeves and Polak-Ribiere)

The specific algorithms you will be implementing are two: Fletcher-Reeves CG, and Polak-Ribiere CG. Your codes should any function as an input argument. Your two matlab codes should be called fminuncFR_TeamName, and fminuncPR_TeamName, and the usage is identical to fminunc from MATLAB:

Start with
```
>> options = optimset('TolX',1e-6,'TolFun',1e-6,'LargeScale','off','MaxIter',1000, 'GradObj','on');
>> x0= [0; 0]; % set to initial guess
```
Then call MATLAB code
```
>>[X,FVAL,EXITFLAG] = fminunc(myfun,x0,options);
```
or your FR CG code
```
>>[X,FVAL,EXITFLAG]= fminuncFR_TeamName(myfun,x0,options);
```
or your PR CG code
```
>> [X,FVAL,EXITFLAG]= fminuncPR_TeamName(myfun,x0,options);
```

As far as possible, your code should mimic MATLAB. **Most importantly, your code should not run endlessly, make MATLAB crash, but terminate gracefully (even if you are unable to find a minimum)**! You can use the linesearch method made available to you or develop your own. You are not allowed to use any of the MATLAB numerical functions like polyfit (except multiplication, addition etc). Eventually, you are responsible for modifying the line search and fine tuning it, i.e., the proper working of this code is your responsibility.

Test problems

Compare your algorithms against MATLAB on appropriate test problems. Examples include.

1. **Quadratic functions:** min $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
2. **Rosenbrock's Function**: This function is geometrically described as a narrow "banana" shaped valley. This is a hand-constructed test problem, and is the doom of any code that cannot traverse narrow valleys. The mathematical form of the function is:

   min $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

   The traditional starting point for this problem is (-1.2, 1.0). **For consistency, make all your algorithms start from this point. Also, for consistency, all the codes must use the same convergence criterion.** The optimal solution is (1.0,1.0).
3. **Spring Equilibrium: Use the 2-spring system, discussed in class, as a base model.**

Selection of test problems is a major task in comparative performance evaluation of optimization codes. The following books give an exhaustive list of test problems.

1. W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes*. *Lecture Notes in Economics and Mathematical Systems*, Springer Verlag, Berlin, Germany, 1981.
2. W. Hock and K. Schittkowski. *More Test Examples for Nonlinear Programming Codes*. *Lecture Notes in Economics and Mathematical Systems*, Springer Verlag, Berlin, Germany, 1981.

Tabulate your results

Tabulate the number of function, gradient, and hessian calls each of the algorithm/code required to find the minimum (if it did reach the minimum). **Include the variable and function values at the termination point, number of iterations, convergence criteria (important!), step length (if applicable) and the equivalent function evaluations (the number of function evaluations required if finite difference derivatives were used) in the table.**

Plot trajectories

Plot trajectories of the algorithm/code as it approaches the minimum for representative runs. These are extremely informative in understanding the algorithms. The trajectory is a line plot of all the points at which the function value was evaluated. Plot the contours of the function and overlay the trajectory on the contour plot.

Write a report on your findings

Using the results of Tasks 4 and 5, make a judgment on the best algorithm for optimizing unconstrained engineering systems, considering both accuracy and efficiency. Report your conclusions on the algorithms in a brief report. **Please emphasize any algorithmic insight that you obtained from either the table of results or the trajectories. Your report should contain at most 2 pages (of text), one table summarizing all the results (Task 4 above) followed by as many pages of attached figures as you like**. Also, include the listing of your code and the results of a typical optimization session using your code.

Grading criterion

The percentage of the total grade for tasks 2-6 above is as follows:

2. 60% for developing a working code.
3. 25% for the report clarity and presentation.
4. 15% with a significant weight on the insights you provide. Please don't spend time re-describing the problem. Instead focus on the discussion of the results. Handwritten reports are ok, but handwritten tables or trajectories are not ok.

Please email your report and codes to me.