MATH 519/619 Optimization | Project #1
Winter quarter 2005 | Miguel Á. Carreira-Perpiñán

This project studies the **line-search Newton-CG algorithm for unconstrained optimisation** (chapters 5–6 of the book *Numerical Optimization* by Nocedal and Wright; read carefully pages 102–112 and 135–141). It consists of programming some Matlab functions and running them on some examples, commenting on the results.

Note: you may discuss issues with each other, but you have to produce your own solutions for every part.

I Matlab functions

Write the following Matlab functions, **using the templates provided** and following strictly the convention given for the input and output arguments:

1. `Olincg`: linear conjugate gradient algorithm 5.2 (p. 111) to solve a positive-definite linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$.
2. `Olincg1`: modification of `Olincg` that stops iterating when encountering negative or zero curvature directions. This is intended to be called from `Onewtoncg`.
3. `linesearch`: backtracking line search (procedure 3.1, p. 41ff). Use the following defaults: initial step length $\overline{\alpha} = 1$, rate of decrease of the step length $\rho = 0.8$, sufficient-decrease Wolfe condition $c = 10^{-4}$. (Encapsulating the line search in a separate function allows us to use this line search in other optimisation algorithms, and to modify the line search without affecting functions that call it.)
4. `Onewtoncg`: line-search Newton-CG algorithm 6.1 (p. 140), using `linesearch` and `Olincg1`. Use the same convergence criterion as in `Osteepdesc`, i.e., stop when $\|\nabla f(\mathbf{x}_k)\| \le$ `tol` OR $k \ge$ `maxit` (tolerance achieved or maximum number of iterations achieved). The argument `convcrit` determines what forcing sequence to pass to `Olincg1`: $\eta_k = 0.5$ for `'linear'`, $\eta_k = \min(0.5, \sqrt{\|\nabla f(\mathbf{x}_k)\|})$ for `'superlinear'`, and $\eta_k = \min(0.5, \|\nabla f(\mathbf{x}_k)\|)$ for `'quadratic'`.
5. For each problem tested in the evaluation, write one driver file that sets up the problem, solves it and displays the results. As an example, see `driver.m`. Since the combinations of different `tol`, different forcing sequence, etc. result in several problems, just send me two representative drivers, one for `Olincg` and another for `Onewtoncg`.

Programming advice:

- Write the functions in the order above. Use my functions `fcontours`, `plotseq`, etc.
- Make sure you understand how `Osteepdesc` works because your `Onewtoncg` program should look very similar to it. Follow the convention in `Osteepdesc` for:
  - default values for arguments
  - passing as an argument an arbitrary function handle and its parameters: `f, paramf`
  - obtaining the value of the function, gradient and Hessian: `[ff g H] = f(x',paramf:);`
- To get more decimals in Matlab do `format long`. Use `set(gca,'DataAspectRatio',[1 1 1]);` to avoid distorted plots.
- Program thinking of n dimensions, not 2. It's more general and usually easier.
- Avoid fancy features I don't ask for: error-checking of arguments, informative messages, etc. This is not a programming course.

II Evaluation

Note: do not simply report results; discuss them (clearly and concisely).

1. `Olincg`. Test it with a system where $\mathbf{A}$ is the Hilbert matrix of order n ($a_{ij} = \frac{1}{i+j-1}$ for $i, j = 1 \dots, n$), $\mathbf{b} = (1, \dots, 1)^T$ and with initial point $\mathbf{x}_0 = \mathbf{0}$. Try dimensions $n = 3,\ 5,\ 8,\ 12,\ 20$. Use the following Matlab functions: `hilb`, `cond`.
   (a) Report the condition number of $\mathbf{A}$ and the number of iterations required to reduce the norm of the residual $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ below 10^{-6}. Discuss the results in view of theorem 5.1 (p. 103).
   (b) Plot f_k and $\|\mathbf{r}_k\|$ (the latter using `semilogy`) as a function of the iteration index k. At each iteration, does the value of f_k decrease? How about the value of $\|\mathbf{r}_k\|$?

2. `Onewtoncg`. Apply it to the Rosenbrock function in 2 dimensions (eq. (2.23) in the book) from two initial points $\mathbf{x}_0 = (1.2, 1.2)$ and $\mathbf{x}_0 = (-1.2, 1)$ (cf. exercise 3.1).
    (a) Plot the iterates over the function contours (use `fcontours` and `plotseq`). Plot $\|\mathbf{x}_k - \mathbf{x}^*\|$ (where $\mathbf{x}^*$ is the exact minimiser) and $f(\mathbf{x}_k)$ as a function of k. Report the number of Newton iterations, and the number of CG iterations ran inside each Newton iteration. How large is the condition number of $\nabla^2 f(\mathbf{x}^*)$?
    (b) Explore the effects of the following and comment on the results:
        i. The value of `tol`, e.g. try 10^{-6} and 10^{-10}.
        ii. The line search: use different values of the initial step length, e.g. $\overline{\alpha} = 1,\ 0.5,\ 0.2$. Also, try not using a line search at all and fixing the step length to $\alpha = 1$.
        iii. The forcing sequence (use `convseq` to estimate empirically the convergence rate).
    (c) Repeat everything for `Osteepdesc` and compare the results with those of `Onewtoncg`, in terms on convergence rate, number of iterations and (roughly) in terms of *actual* computational cost.
    (d) Finally, run `driver1.m`, which runs `Onewtoncg` with function `F6_1b` (n dimensions). Comment on the results.

III What you have to submit

1. By email, your code for the Matlab functions (`Olincg`, `Olincg1`, `linesearch`, `Onewtoncg`), and a driver file for each of the two problems tested. Pack all files into a **single** file (tar.gz or zip) and use as the email subject `[MATH519] Project 1`.

2. In paper or preferably in electronic form, your evaluation of the methods. Do not print the methods' results (figures, tables), instead have the driver file reproduce them (I will run the driver file and check it against your evaluation).