Numerical Optimization 2024 - Project 1, Phase 2 (group)

Deadline: Sunday, May 19, 23:59.
General description:
The overall task of Project 1 is to program the four methods for unconstrained optimization: steepest descent (SD), Newton method (NM), a conjugate gradient method (CG) and a quasi-Newton method (QN).

In the second phase, you can get altogether 40 points plus 10 bonus points by solving/implementing the following tasks and running the tests specified later:

1. **Newton method with Hessian modification (5 points):** see Section 3.4 of the Nocedal-Wright book - you can choose one of the suggested strategies.
2. **linear CG (5 points).**
3. **nonlinear CG (5+5=10 points):** the Fletcher-Reevers method (F-R) and the Polak-Ribiere method (P-R).
4. **QN methods (5+5=10 points + 5 bonus points):** the BFGS method (5 points) and the SR1 method with line-search (5 points). Additionally, 5 bonus points can be obtained for the SR1 method within the trust-region framework (Chapter 4 of the N-W book), see Algorithm 6.2 of the N-W book.
5. **derivatives approximation (5+5=10 points):** what if the user only provides the objective function in a way that we can evaluate it at any point, but there is no formula defining the function and we do not know the derivatives? In Section 8.1 (which we will *not* cover during the lectures), you can find the methods for approximating the gradient (5 points) and the Hessian (5 points) of a function, using only its function values.
6. **outperforming the NM with a QN (3+2=5 bonus points):** 3 bonus points will be awarded if you can find a problem, for which the Newton method finds a solution, but it is slower than one of the quasi-Newton methods (simply compare the time each method needs to reach a solution). Careful, while computing the next iterate of the Newton method, do not compute the inverse matrix! Solve the linear system instead. If it helps, you might compare the versions of NM and QN that do not work with explicitly given derivatives and compute the approximations (see above). Additional 2 points will be given, however, if you can do it using explicit derivatives.

Testing/problems to solve:
Consider the following two functions:

$$\text{(Rosenbrock function)} \quad f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$

with 3 different starting points x_0:

$$(1.2, 1.2), \quad (-1.2, 1), \quad (0.2, 0.8)$$

and

$$f(x) = 150(x_1 x_2)^2 + (0.5x_1 + 2x_2 - 2)^2 \tag{2}$$

with 3 different starting points x_0:

$$(-0.2, 1.2), \quad (3.8, 0, 1), \quad (1.9, 0.6).$$

Thus, there are altogether 6 "problems" to solve (2 functions with 3 starting points). Moreover, there are altogether 12 different "methods" to try: the standard NM (mainly for comparison purposes) and the NM with Hessian modification, 2 (nonlinear) CG methods, 2 QN methods, and, moreover, each of these 6 methods should be used both with exact derivatives and with approximated derivatives. This results into $6 \cdot 12 = 72$ runs.

For the line search, use the backtracking algorithm (Algorithm 3.1 in the Nocedal-Wright book). For the stopping criterion, use simply $\|\nabla f(x_k)\| \leq 10^{-6}$. If you have problems reaching this accuracy with some method, you can use a lower accuracy (e.g. 10^{-5} or 10^{-4}) or another reasonable stopping criterion (it is up to you to decide what might be a reasonable stopping criterion).

Additional 12 runs should be made in case you choose to do the bonus SR1 method within the trust-region framework. Similarly, relevant additional runs should be made if you choose to do the bonus of outperforming the NM with a QN.

Finally, to test the linear CG, additional 10 runs should be made for the Hilbert matrix from the first phase of the project (part (iii)) - solve these 5 problems with linear CG as well as with the standard SD again for comparison. Here, however, use the exact line-search!

Note that, unlike in the first phase, it is not necessarily a problem if a method does not reach the solution. If you encounter problems, you should try to understand whether it means there is an error in your code (in which case you should correct it) or whether that particular method should not be expected to converge in that particular situation.

What should your submission contain?

All the files containing the codes and a report.

In the beginning of the report, state clearly which of the tasks 1. - 6. you have implemented. Then, print all the runs you performed. After each run, provide a short summary with number of iterations, final iterate x_k, the size $\|\nabla f(x_k)\|$, and the distance to the solution. For function (2) there are 2 global minimizers with the minimal value 0 - it should be easy for you to identify them.