Numerical Optimization - Project 1, Phase 1 (individual phase)

Deadline: Sunday, May 22, 23:59.

General description:

The overall task of Project 1 is to program the four methods for unconstrained optimization: steepest descent, Newton method, a conjugate gradient method and a quasi-Newton method (we will cover conjugate gradient and quasi-Newton methods in the next lectures).

It is up to you whether you choose the line search or the trust region approach.

It is also up to you which of the (nonlinear) conjugate gradient methods and which of the quasi-Newton methods you do.

In the first phase, you should just program these methods a test them on 10 rather simple problems specified below. You do not have to program the methods perfectly to make them "solve" such simple problems; in the second phase, you will work in teams and you will try to improve your programs and solve more difficult problems.

Altogether there are 10 problems and 4 methods, i.e. 40 tasks. In your submission, you should visibly specify the percentage of these 40 tasks you managed to solve successfully - this will be your "auxiliary percentage". Your final (individual) score for Project 1 will be computed by multiplying the auxiliary percentage with the percentage your team will get after the evaluation of the team submission.

Problems to solve:

(i) 5 problems with 1 or 2 variables for which you know the solution - do not use a quadratic objective, you can use e.g. polynomials of the degree higher than 2, but also other types of functions.

(ii) 5 problems corresponding to least-squares problems specified below.

(iii) Additionally, try also to run your methods of the following two functions:

$$\text{(Rosenbrock function)} \quad f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad \text{take } x_0 := (-1.2, 1) \tag{1}$$

and

$$f(x) = 150(x_1 x_2)^2 + (0.5x_1 + 2x_2 - 2)^2, \quad \text{take } x_0 = (-1, 0). \tag{2}$$

Your methods are **not** required to solve these 2 problems. The purpose is just to see what happens (maybe, what can go wrong).

Hint for (i):

Constructing a polynomial with a known minimizer is not very difficult. For instance, $(x-a)(x-b)(x-c)$ is a polynomial of degree 3 with the roots at a, b and c. Thus, one can integrate it to obtain a polynomial of degree 4 with the stationary points at a, b and c.

Regarding the least-squares problems, consider the function $g : \mathbb{R} \to \mathbb{R}$ given simply by $g(x) = \sin(x)$. Pick an interval $[-q, q]$ for some $q > 0$ and generate m points $a_j \in [-q, q]$ for $j = 1, \ldots, m$ randomly (or uniformly). Take (a_j, b_j) for $b_j = g(a_j) = \sin(a_j)$ for $j = 1, \ldots, m$ as your data points.

The idea is to approximate points (a_j, b_j) which come from the graph of function sin by a polynomial of some degree n.

More precisely, consider the function $\phi(x;t) = x_0 + x_1 t + x_2 t^2 + \ldots + x_n t^n$ which is a polynomial of degree n in variable t with *parameters* $x = (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1}$ which specify the polynomial. We are trying to choose the parameters x so that $\phi(x;t)$ approximates the data points (a_j, b_j), i.e. when we plug in a_j into $\phi(x;t)$ for t it should be close to the value b_j.

This leads to the optimization problem of minimizing function $f : \mathbb{R}^{n+1} \to \mathbb{R}$ given by

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} \big(\phi(x; a_j) - b_j\big)^2 =: \frac{1}{2} \sum_{j=1}^{m} r_j^2(x)$$

for the residuals $r_j(x) = \phi(x; a_j) - b_j$.

Have a look into the beginning of Chapter 10 in the Nocedal Wright book for some basic info about the least-squares problems. Therein, you will also find very simple formulas for first- and second-order derivatives of f. Note that the residuals $r_j(x)$ are linear in x, so $\nabla^2 f(x) = J(x)^T J(x)$, see formulas (10.3) and (10.5).

To run your test, you can try q between 10 and 100, m around 100 and start with $n = 5$ and then keep increasing it (e.g. $n = 10, 20, 30, 50, 100$). As you are increase n, make sure that m is larger than n. You have plenty of freedom to choose which problems you try to solve, so if you fail to solve some problems, you can just try different ones. For instance, if you have difficulties solving problems for large $n = 50, 100$, you can choose only more moderate values $n = 5, 10, 20$ and instead you can try different values of q, m, etc. You can also try a different function g. At the end, you should just pick which 5 problems you solved.

You can also compare the obtained polynomial with the Taylor expansion of $g(x) = \sin(x)$ of the same degree. For instance, you can fix n, then depict the graph of the Taylor expansion of degree n in order to see on how large interval $[-q, q]$ it provides a good approximation. You can then choose q accordingly, generate the data points (a_j, b_j), solve the least-squares problem and see if the obtained polynomial resembles the Taylor expansion.

When is a problem "solved"?

An iterative method needs some stopping criteria. Such criteria typically correspond to convergence results. Typically, we expect that the gradients converge to zero. For the first phase, you can simply choose $\|\nabla f(x_k)\| \leq 10^{-5}$ as your stopping criterion.

Thus, for the first 5 problems where you know the actual solution x^*, a problem is solved if your solution (final iterate) $\tilde{x}$ satisfies the above stopping criterion $\|\nabla f(\tilde{x})\| \leq 10^{-5}$ and also $\|\tilde{x} - x^*\| \leq 10^{-5}$.

For the second 5 problems, a problem is solved if your solution (final iterate) satisfies the above stopping criterion and the depicted polynomial matches the data points reasonably well.

What should your submission contain?

All the files containing the codes and a brief report.

In the beginning of the report, state clearly your "auxiliary percentage" (the percentage of tasks solved).

Next, specify 10 problems you solved. For the first 5 problems, write the actual solution x^*, your solution $\tilde{x}$, the values $\|\nabla f(\tilde{x})\| \leq 10^{-5}$ and $\|\tilde{x} - x^*\| \leq 10^{-5}$ and the number of iterates (for each of 4 methods).

For the second 5 problems, specify the function g, degree n of the polynomial and the final optimal polynomial and depict the data points (a_j, b_j) and the graphs (function g, its Taylor expansion of degree n, and the optimal polynomial).

Finally, describe what happened when you tried to minimize functions (1) and (2). It is perfectly ok if your methods failed to find the solution.