Numerical Optimization 2024 - Project 1, Phase 1 (individual)

Deadline: Sunday, May 5, 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 talk about conjugate gradient and quasi-Newton methods in the next lectures - see lectures 5,6,7 from 2023).

In the first - individual - phase, you should program only the steepest descent (SD) and the Newton method (NM) and then test them on 15 rather simple problems specified below.

In the second - group - phase, you will work in teams/groups and you will try to improve your programs and solve more difficult problems and you will also program a conjugate gradient method and a quasi-Newton method.

In this phase, there are altogether 15 problems for both methods, i.e. 30 tasks. In your submission, you should visibly specify the percentage of these 30 tasks that you managed to solve successfully - this will be your "auxiliary percentage". You should also submit your auxiliary score into the special checkmarks form (for this reason, the tasks are numbered 1-30). The final score for Project 1 of each student will be computed as an avarage of the auxiliary percentage with the percentage your team/group will get after the evaluation of the team submission.

Problems to solve (more details are provided below):

(i) (SD and NM - 10 tasks; tasks nr. 1-10; 1-5 for SD, 6-10 for NM)
5 problems with 1 variable 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) (SD and NM - 10 tasks; tasks nr. 11-20; 11-15 for SD, 16-20 for NM)
5 problems corresponding to least-squares problems specified below.

(iii) (only SD - 5 tasks; tasks nr. 21-25)
5 quadratic problems of the form

$$\min_{x\in\mathbb{R}^n} f(x) := \frac{1}{2}x^T Qx - b^T x,$$

where $b = (1,1,\ldots,1) \in \mathbb{R}^n$ and Q is the so-called Hilbert matrix of dimension n given by $q_{ij} = 1/(i+j-1)$ for $i,j = 1,\ldots,n$ (q_{ij} denotes the element of Q in row i and column j). Choose $n = 5,8,12,20,30$.

(iv) (only NM - 5 tasks; tasks nr. 26-30)
5 problems with polynomials of degree 4 and 2 variables for which you know the solution.

For the line search, use the backtracking algorithm (Algorithm 3.1 in the Nocedal-Wright book).

Starting point:
For a starting point always take the zero vector of the appropriate dimension. Exceptions:

- In (ii), if the steepest descent method needs too many iterates, you can restart it at the current iterate.
- In (iii), if the steepest descent method needs too many iterates, you can find the solution first and then start near the solution.
- In (iv), if the Newton method fails from the zero starting point, you can choose a point near the solution as your starting point.

When is a problem "solved"?
An iterative method needs some stopping criteria. Such criteria typically correspond to convergence results. Since we usually expect the gradients to converge to zero, you can simply choose $\|\nabla f(x_k)\| \leq 10^{-6}$ as your stopping criterion. For some problems (particularly (iii) for $n = 20$ and $n = 30$), it may be hard to reach this accuracy and you may then choose $\|\nabla f(x_k)\| \leq 10^{-4}$ or even $\|\nabla f(x_k)\| \leq 10^{-3}$.

Thus, for the problems where you know all the local minimizers ((i),(iii), and (iv)) a problem is solved if your solution (final iterate) $\tilde{x}$ satisfies the above stopping criterion $\|\nabla f(\tilde{x})\| \leq 10^{-6}$ and it is also close to one of the local minimizers.

For the least-squares problems (ii), a problem is solved if your solution (final iterate) satisfies the above stopping criterion and the depicted polynomial matches the data points reasonably well (naturally, it does not really have to actually match the points, since this is only possible if the degree n is large enough) - see below.

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 the problems you worked with.

For the tasks in (i), (iii), and (iv), write all the local minimizers, your solution $\tilde{x}$, the values $\|\nabla f(\tilde{x})\|$ and $\|\tilde{x} - x^*\|$ (for the local minimizer x^* closest to $\tilde{x}$) and the number of iterates.

For the tasks in (iii), write also the eigenvalues $\lambda_1, \ldots, \lambda_n$ of Q, the number $(\lambda_n - \lambda_1)/(\lambda_n + \lambda_1)$, and the number λ_n/λ_1 (the condition number of matrix Q). For all iterates verify whether the inequality (3.29) from Theorem 3.3. from the Nocedal-Wright book is satisfied.

For the tasks in (ii) (see below), 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).

More details and hints for setting the problems:

- Hints for tasks (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.
- Hints for tasks (iii):
  The actual solution x^* can be found simply by solving the linear system $Qx = b$.
- Hints for tasks (iv):
  One can constuct a polynomial with 2 variables of degree 4 as follows:

$$f(x) = (q_1(x))^2 + (q_2(x))^2, \qquad x = (x_1, x_2)$$

where q_i for $i = 1,2$ are quadratic functions, i.e.,

$$q_i(x) = a_i x_1^2 + b_i x_2^2 + c_i x_1 x_2 + d_i x_1 + e_i x_2 + f_i \qquad (i = 1,2)$$

for some parameters a_i, b_i, c_i, d_i, e_i and f_i. Clearly, f is nonnegative and it equals zero if and only if $q_1(x) = q_2(x) = 0$. Thus, one just needs to choose these parameters so that it is easy to compute when $q_1(x) = q_2(x) = 0$. One can e.g. choose q_2 to be linear ($a_2 = b_2 = c_2 = 0$). For better understanding, look into Exercises 2.1 and 2.9 in the Nocedal-Wright book. For instance, the Rosenbrock function from Exercise 2.1 corresponds to

$$a_1 = -10, e_1 = 10, \quad d_2 = -1, f_2 = 1, \quad \text{(all the others are 0)},$$

while the function from Exercise 2.9 corresponds to

$$d_1 = 1, b_1 = 1 \quad \text{(all the others are 0)}.$$

- Details and hints for tasks (ii):
  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 and mainly into Section 10.2 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 $\nabla^2 f(x) = J^T J$, see formulas (10.3) and (10.5) - general case - and also the formula below (10.13) - the case with $r_j(x)$ linear.

  For instance, let $q = 10$, i.e. we have the interval $[-10,10]$. We choose $m = 100$ points a_j uniformly in $[-10,10]$ as $a_j = -10 + j/5$ for $j = 1,\ldots,100$, so $a_1 = -9.8$, $a_2 = -9.6$, $a_3 = -9.4$ and so on $a_{98} = 9.6$, $a_{99} = 9.8$, $a_{100} = 10$. Then, by plugging a_j into function sin we generate pairs $(a_j,b_j) = (a_j, \sin(a_j))$ given as $(a_1,b_1) = (-9.8, \sin(-9.8))$, $(a_2,b_2) = (-9.6, \sin(-9.6))$ and so on $(a_{99},b_{99}) = (9.8, \sin(9.8))$, $(a_{100},b_{100}) = (10, \sin(10))$. Then, given the degree n, say $n = 10$, the objective $f(x)$ is just (1/2 of) the sum of squares of $r_j(x)$, where $r_j(x)$ are just simple linear functions

$$r_j(x) = c_j^T x - b_j \quad \text{for} \quad c_j^T = (1, a_j, a_j^2, a_j^3, \ldots, a_j^{10}) \in \mathbb{R}^{11}.$$

In another words, the number b_j is just $\sin(a_j)$ and the vector c_j^T has components a_j^i for $i = 0,\dots,n$ and these two define the linear function r_j. Similarly, since $\nabla r_j(x) = c_j^T$, the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ are just the sums of $r_j(x)\nabla r_j(x) = (c_j^T x - b_j)c_j^T$ (vector) and $c_j c_j^T$ (matrix), respectively. Thus, once you generate the data pairs (a_j, b_j), it should be quite easy to implement f, ∇f and $\nabla^2 f$ and then run your programs.

The results will depend on how you choose the parameters q, m, and n. First of all, m should be always higher than n (say, $m = 100$ or even more). Next, if q is large (which allows sin to have several "bumps") and n is small (max $n-1$ "bumps" - local extrema), then the polynomial that best approximates the data (a_j, b_j) will be very close to the zero polynomial.

To get a better idea how to choose the parameters, you can help yourselves with the Taylor expansion of $g(x) = \sin(x)$. For instance, fix a degree n (say, between 5 and 50) and depict the graph of the Taylor expansion of $g(x) = \sin(x)$ of degree n. From that, you can see on how large interval $[-q, q]$ it provides a good approximation of $g(x) = \sin(x)$. 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.

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. You can also try a different function g, particularly, you can also choose q to a polynomial itself (of some degree) and you can see if you can actually recover it if you choose the same degree.

At the end, you should just pick which 5 problems you solved.