FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION

OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

on the practical task No. 2

“Algorithms for unconstrained nonlinear optimization. Direct methods”

Performed by Pavel Belenko, J4132c

Accepted by Dr Petr Chunaev

St. Petersburg

2021

***Goal**:*

The use of direct methods (one-dimensional methods of exhaustive search, dichotomy, golden section search; multidimensional methods of exhaustive search, Gauss (coordinate descent), Nelder-Mead) in the tasks of unconstrained nonlinear optimization

Formulation of the problem:

I. *Use the one-dimensional methods of exhaustive search, dichotomy and golden section search to find an approximate (with precision* $\varepsilon = 0.001$*) solution* $x: f(x) \to min$ *for the following functions and domains:*

1. $f(x) = x^3$, $x \in [0, 1]$;
2. $f(x) = |x - 0.2|$, $x \in [0, 1]$;
3. $f(x) = x \sin\frac{1}{x}$, $x \in [0.01, 1]$.

Calculate the number of f*-calculations and the number of iterations performed in each method and analyze the results. Explain differences (if any) in the results obtained.*

II. *Generate random numbers* $\alpha \in (0,1)$ *and* $\beta \in (0,1)$*. Furthermore, generate the noisy data* $\{x_k, y_k\}$*, where* $k = 0, \ldots, 100$*, according to the following rule:*

$$y_k = \alpha x_k + \beta + \delta_k, \quad x_k = \frac{k}{100},$$

where $\delta_k \sim N(0,1)$ *are values of a random variable with standard normal distribution. Approximate the data by the following linear and rational functions:*

1. $F(x, a, b) = ax + b$ *(linear approximant),*
2. $F(x, a, b) = \frac{a}{1+bx}$ *(rational approximant),*

by means of least squares through the numerical minimization (with precision $\varepsilon = 0.001$*) of the following function:*

$$D(a, b) = \sum_{k=0}^{100} (F(x_k, a, b) - y_k)^2.$$

*To solve the minimization problem, use the methods of exhaustive search, Gauss and Nelder-Mead. If necessary, set the initial approximations and other parameters of the methods. Visualize the data and the approximants obtained in a plot separately for each type of **approximant** so that one can compare the results for the numerical methods used. Analyze the results obtained (in terms of number of iterations, precision, number of function evaluations, etc.).*

Brief theoretical part:

Optimization methods are numerical methods for finding optimal (in a sense) values of objective functions, for example, within the framework of mathematical models of certain processes. Optimization methods are widely used in data analysis and machine learning.

Let the target function f = f(x), where x – multi-dimensional vector from some subset Q Euclidean spaces R^m. Subset Q may be restricted or match with Rm space. Let consider the task of minimization the f - function on the Q subset. If we want to solve the maximization problem, we need to consider F(x) = −f(x) instead f(x).

We must find such $x^* \in Q$: $f(x^*) = min(f(x))$, $\forall x \in Q$. Otherwise, $x^* = arg(min(f(x))), \forall x \in Q$. The specified formulation of the optimization problem implies the search for a global minimum f on Q. However, below we will consider the search for a local minimum f on Q where the $f(x^*)$ value is minimally only in some neighborhood of the point x^*. Note that it is often much easier to find a local minimum than a global one.

If the optimization problem includes additional conditions on x^* (in the form of a system of S equations and inequalities), then it is called conditional. Otherwise, the optimization problem is called unconditional. Note also that depending on the type f and S there are linear and nonlinear optimization problems. Here and below, we will consider only the problems of unconditional nonlinear optimization. This means that the function f in general, it is nonlinear, and the conditions of S does not apply.

In this work we are interested in direct optimization methods (zero-order optimization methods). Such methods are used in the searching of x^* only with f - function values. These methods are applicable for continuous functions f = f (x) of one-dimensional input vector x^* on Q = [0, 1]. It turns out that direct methods can be used for a wide class of functions f. This, however, is compensated by a rather low convergence rate of the corresponding iterative processes.

Results:

All calculations were performed in the Jupiter lab development environment. NumPy, pandas, matplotlib and SciPy modules were used in the process.

I. One-dimensional direct methods:

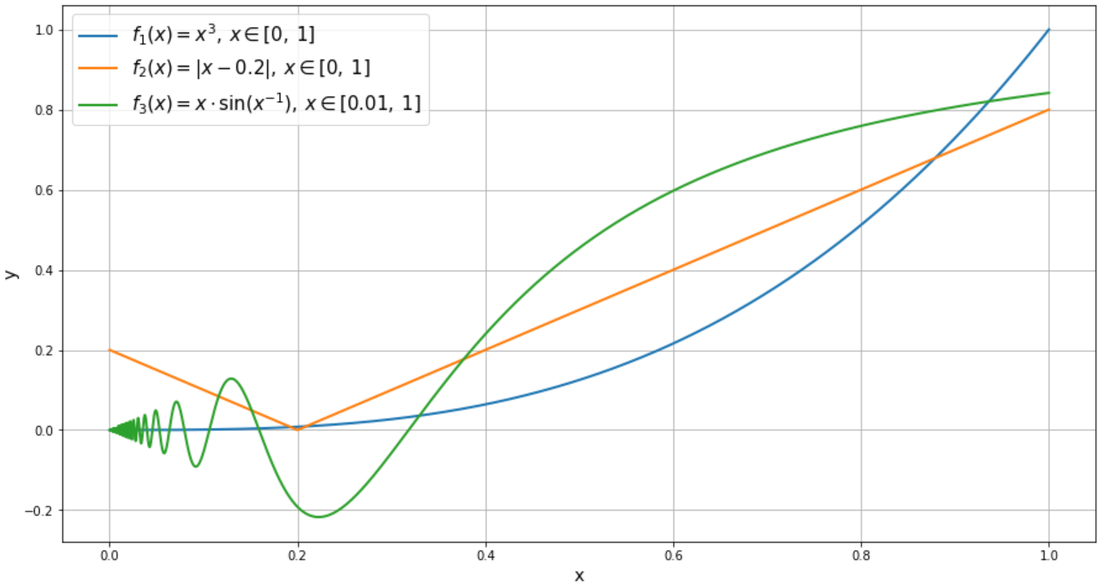

Figure 1 – plots of one-dimensional functions, for each of which it is required to find a minimum

$$f_1(x) = x^3,\ \ x \in [0,\ 1]$$
$$minimum\ by\ exhaustive\ search: x_1^{min} = 0.0$$
$$minimum\ by\ dichotomy\ method: x_1^{min} \in [0,\ 0.001]$$
$$minimum\ by\ golden\ dection\ method: x_1^{min} \in [0,\ 0.001]$$

$$f_2(x) = |x - 0.2|,\ \ x \in [0,\ 1]$$
$$minimum\ by\ exhaustive\ search: x_2^{min} = 0.2$$
$$minimum\ by\ dichotomy\ method: x_2^{min} \in [0.2,\ 0.201]$$
$$minimum\ by\ golden\ dection\ method: x_2^{min} \in [0.2,\ 0.2]$$

$$f_3(x) = x \cdot \sin(x^{-1}),\ \ x \in [0.01,\ 1]$$
$$minimum\ by\ exhaustive\ search: x_3^{min} = 0.223$$
$$minimum\ by\ dichotomy\ method: x_3^{min} \in [0.222,\ 0.223]$$
$$minimum\ by\ golden\ dection\ method: x_3^{min} \in [0.222,\ 0.223]$$

Figure 2 – Result of the calculating of minimum of functions from figure 1, using different optimization methods (calculations were performed with accuracy of $\varepsilon = 10^{-3}$)

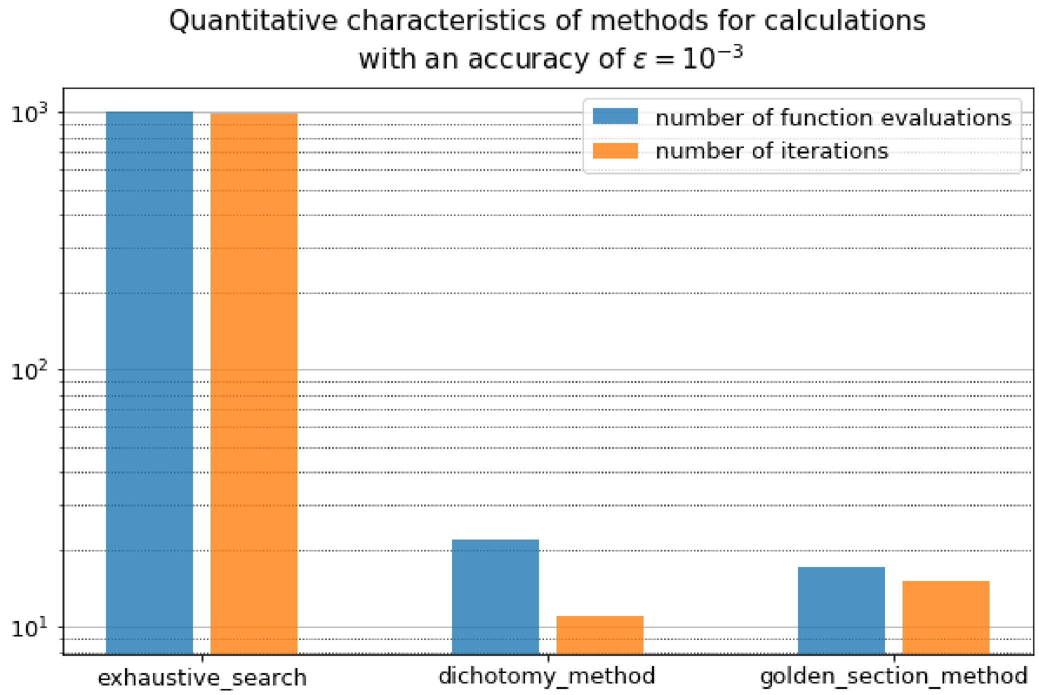

Figure 3 – average number of function evaluations and iterations for different optimization methods (calculations were performed with accuracy of $\varepsilon = 10^{-3}$)

From Figures 1 and 2 we can see that the implemented methods work correctly. Figure 3 shows that:

1) *The exhaustive search method works in the worst way. The number of its iterations and function calls exceeds these values for other methods by more than an order of magnitude.*

2) *The dichotomy method, in comparison with the golden ratio method, requires more function calls, but fewer iterations to achieve the set accuracy. This result is to be expected, because the golden section method is an optimization of the dichotomy method with fewer function calls, but this does not guarantee a reduction in the number of iterations.*

II. Multi-dimensional direct methods:

From Figure 4 we can see that the implemented methods work correctly. All approximation methods give the same result with a given accuracy:

1) *Linear approximant: y = (1.105 ± 0.001) * x + (0.528 ± 0.001)*
2) *Rational approximant: y = (0.720 ± 0.001) / (1 + x * (0.586 ± 0.001))*

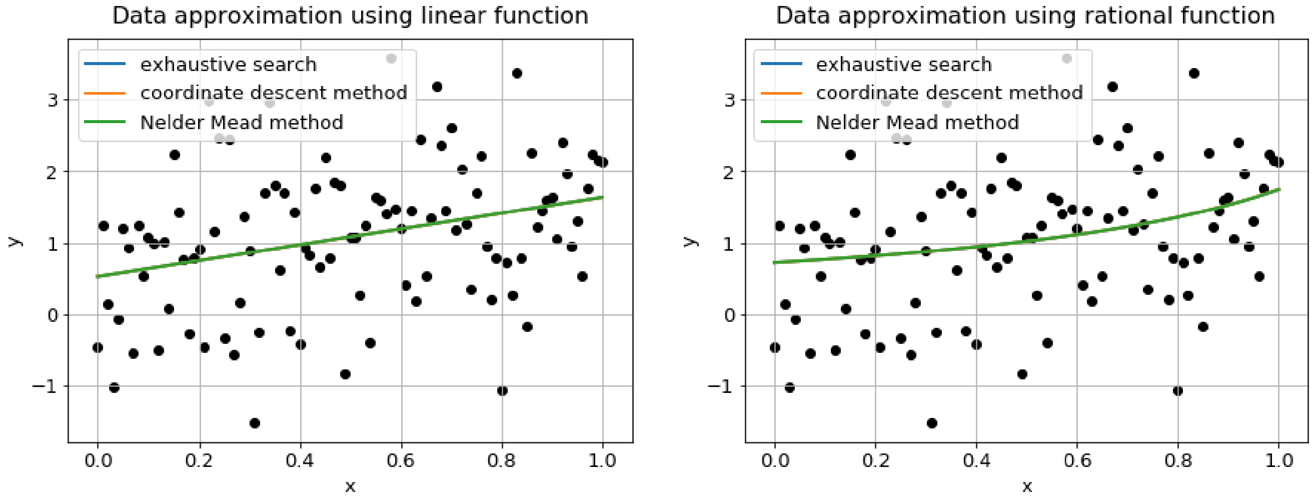

Figure 4 – visualization of the data and the obtained approximants for each type of approximant and each optimization method (calculations were performed with accuracy of $\varepsilon = 10^{-3}$*)*

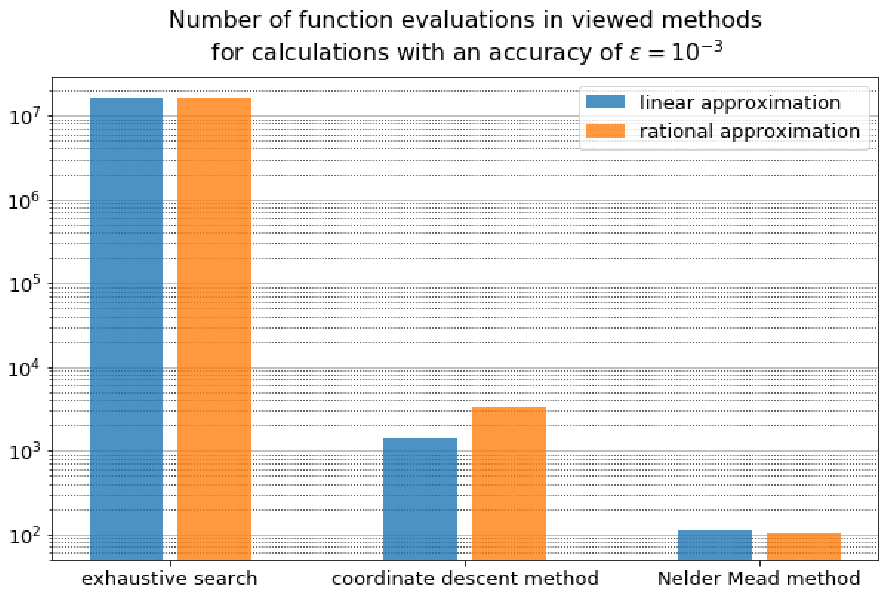

Figure 5 – number of function evaluations for different optimization methods (calculations were performed with accuracy of $\varepsilon = 10^{-3}$*)*

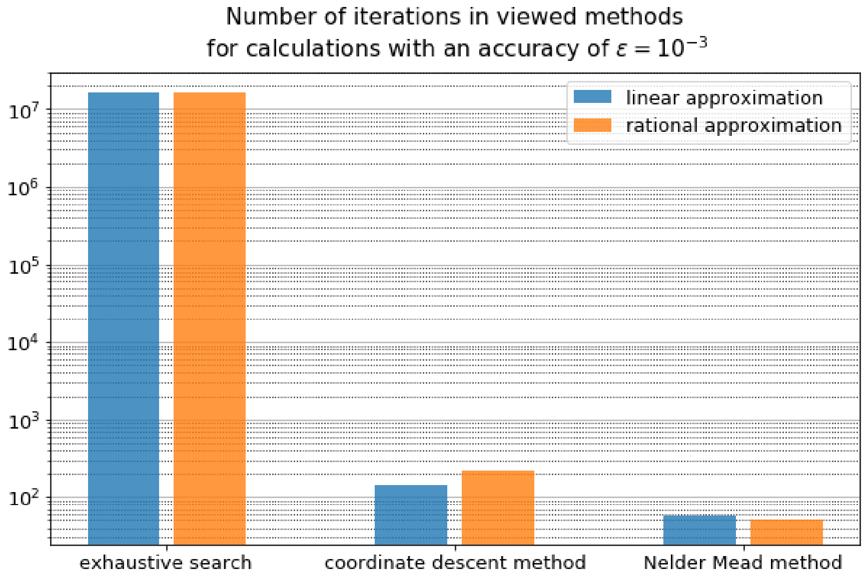

Figure 6 – number of iterations for different optimization methods (calculations were performed with accuracy of $\varepsilon = 10^{-3}$)

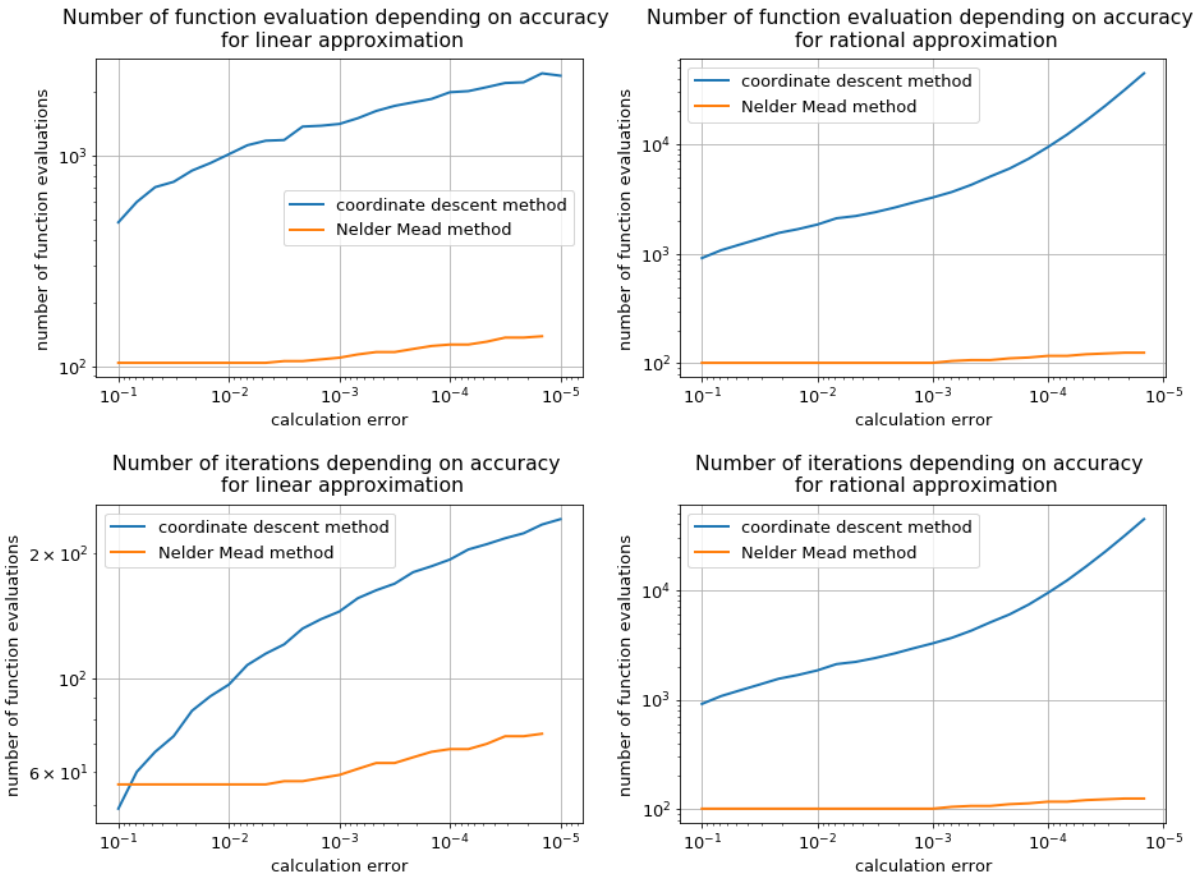

Figure 7 – graphs of the dependence of the number of iterations and the number of function evaluations for Gauss coordinate descent and Nelder-Mead methods. (A comparative analysis for the exhaustive search method is not provided here due to the inexpediency of using it to achieve high accuracy)

The following conclusions can be drawn from figures 5, 6 and 7:

1) *in the multidimensional case, the exhaustive search method is also inferior to other methods.*

2) *the Gauss (coordinate-descent) method is inferior to the Nelder-Mead method both in terms of the number of function evaluations and iterations. However, the Gauss method still works several orders of magnitude faster than the exhaustive search method.*

It is also worth mentioning that the characteristics of the Gauss method can vary greatly depending on the one-dimensional minimization method that is applied at each iteration. In the version I implemented, the scipy.optimize.minimize_scalar() function was used for this.

Conclusions:

As a result of the work, methods of one-dimensional and multidimensional optimization were implemented. These methods have been successfully applied in the tasks of determining the minimum and approximation of noisy data. As a result of comparative analysis, it can be concluded that:

1) *the exhaustive search method is inferior to other methods in both one-dimensional and multidimensional cases. However, the method can be used to find the initial approximation.*

2) *In the one-dimensional case, the dichotomy method loses to the golden section method by the number of function calls, but wins by the number of iterations*

3) *In the multidimensional case, the Gauss method is inferior to the Nelder-Mead method both in terms of the number of function calls and iterations. However, the Gauss method still works several orders of magnitude faster than the exhaustive search method*

Appendix:

To get acquainted with the code, follow the link:

https://github.com/belpablo/Algorithms-21