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

Dmitriy Prusskiy

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:

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

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

Exhaustive search is a brute-force method of finding an optimal solution to the problem. When finding the optimum of one-dimensional functions, it calculates the value of the function at each point in the given range with defined precision. For example, if we want to find the minimum of the function in the range from 0 to 1 with precision 0.001, we have to make 1000 calls of the function.

For the multiple-dimensional problems, the number of function calls needed to find optimum using exhaustive search increases as an exponential function of the number of dimensions. If we wanted to find the minimum of the 2D function in the range from 0 to 1 with precision 0.001, we should make 10^6 calls. In practice, this method is used to find initial approximations in a broad search space.

Dichotomic search in a broad sense is a search algorithm that operates by selecting between two distinct alternatives (dichotomies) at each step. It is a specific type of divide and conquer algorithm. Therefore, it has to take substantially fewer steps than exhaustive search to find an optimum of the function.

The **golden-section search** is a technique for finding an extremum (minimum or maximum) of a function inside a specified interval. For a strictly unimodal function with an extremum inside the interval, it will find that extremum, while for an interval containing multiple extremas (possibly including the interval boundaries), it will converge to one of them. If the only extremum of the interval is on a boundary of the interval, it will converge to that boundary point. The method operates by successively narrowing the range of values on the specified interval, which makes it relatively slow, but very robust. The technique derives its name from the fact that the algorithm maintains the function values for four points whose three interval widths are in the ratio 2-φ:2φ-3:2-φ where φ is the golden ratio. These ratios are maintained for each iteration and are maximally efficient. Except for boundary points, when searching for a minimum, the central point is always less than or equal to the outer points, assuring that a minimum is contained between the outer points.

The idea of the **Gauss method** of multidimensional function optimization is that in each iteration, the minimisation is carried out only with respect to one vector component of the multidimensional variable x. The method is simple but hardly efficient. Problems appear when the level lines of a function to be optimized are strongly elongated along the “diagonal” line $x_1 = x_2$. If the initial approximation is on $x_1 = x_2$, then the process gets stuck.

The **Nelder–Mead method** is a commonly applied numerical method used to find the minimum or maximum of an objective function in a multidimensional space. The Nelder–Mead technique is a heuristic search method that can converge to nonstationary points on problems that can be solved by alternative methods. The idea of the method is to find function values at the vertices of simplex in the search space, define a vertex with maximum function value, and then reflect this point with respect to the gravity center of the other points.

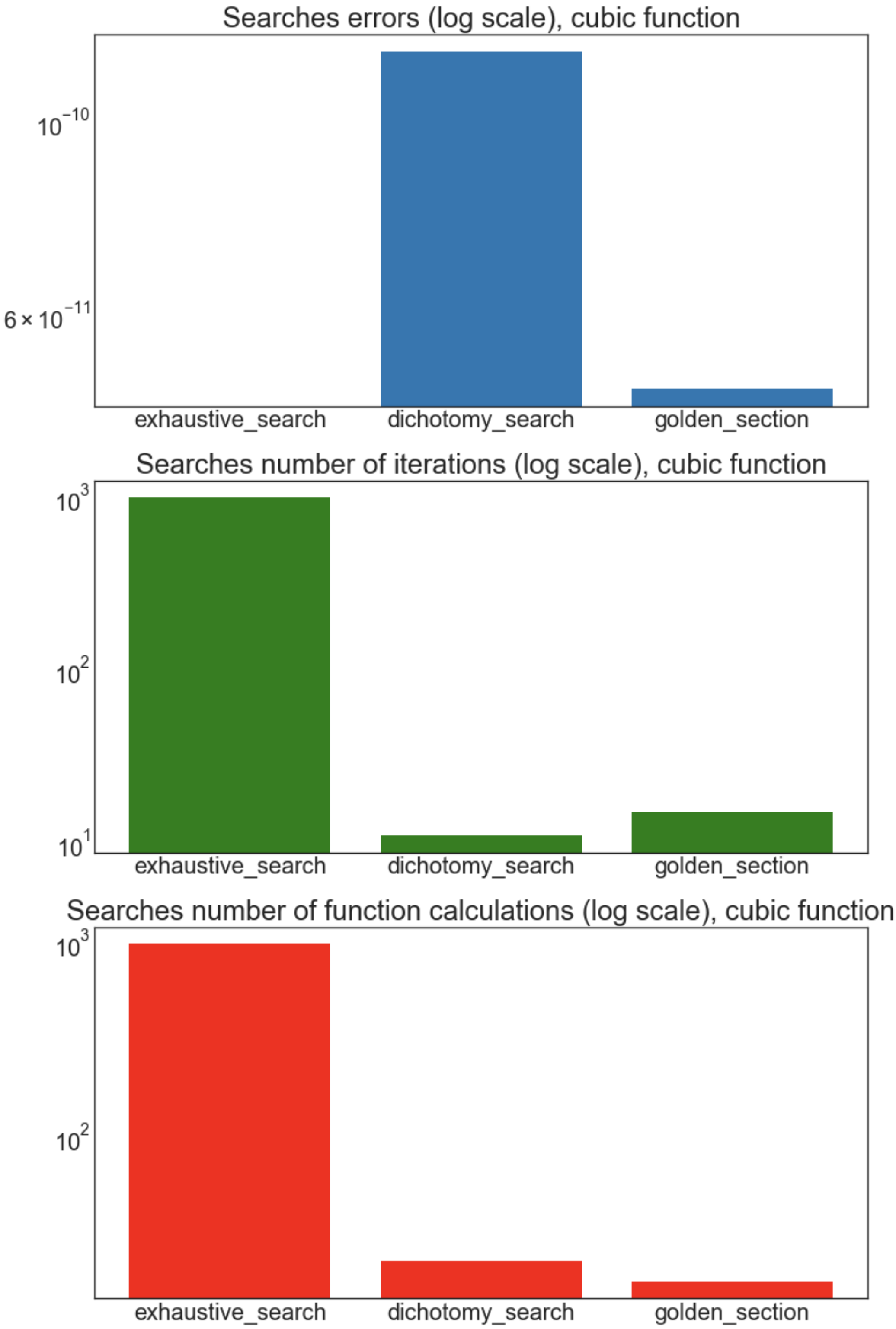
Results

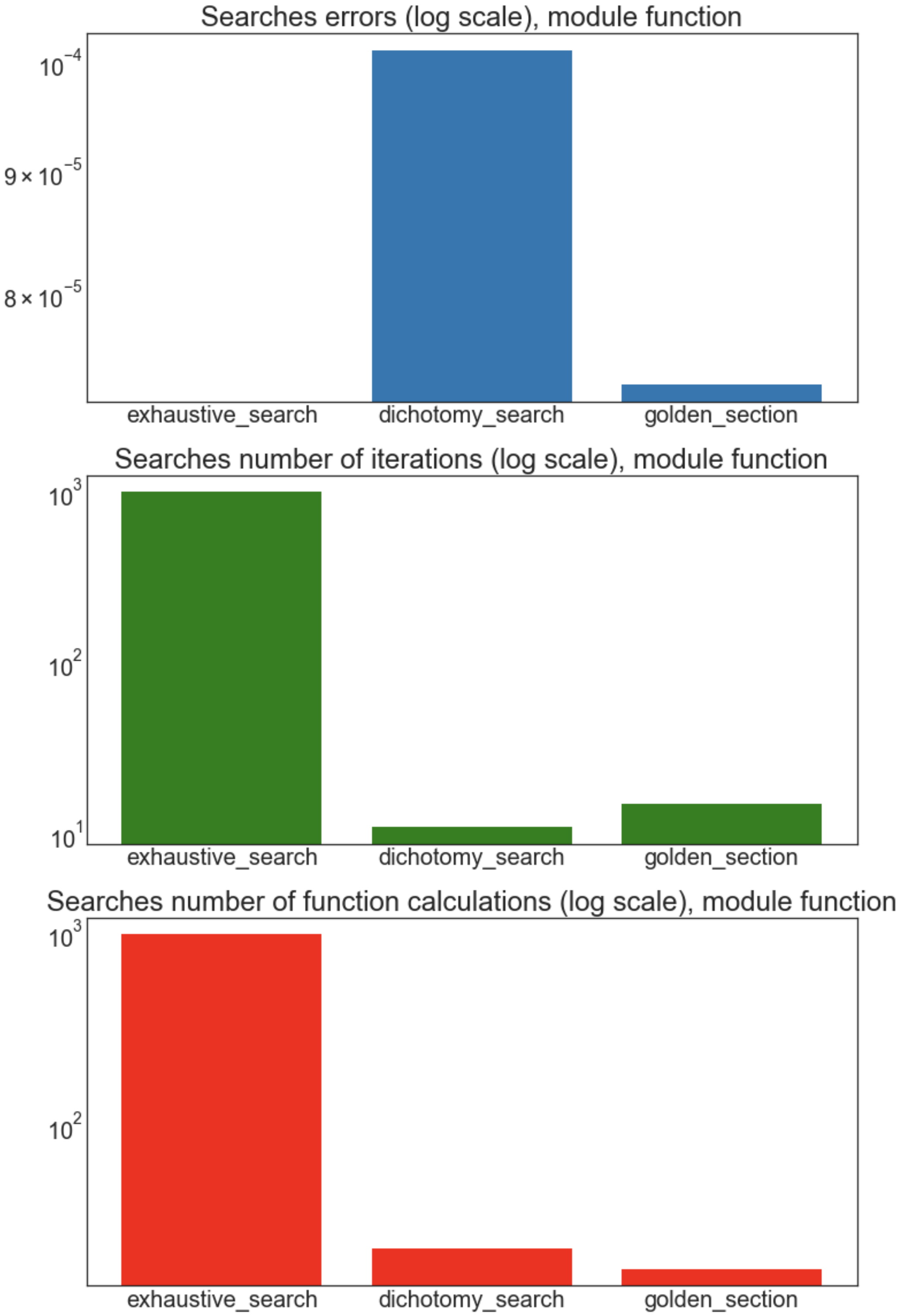
The results were obtained using Python 3.8.5.

One-dimensional optimization problem

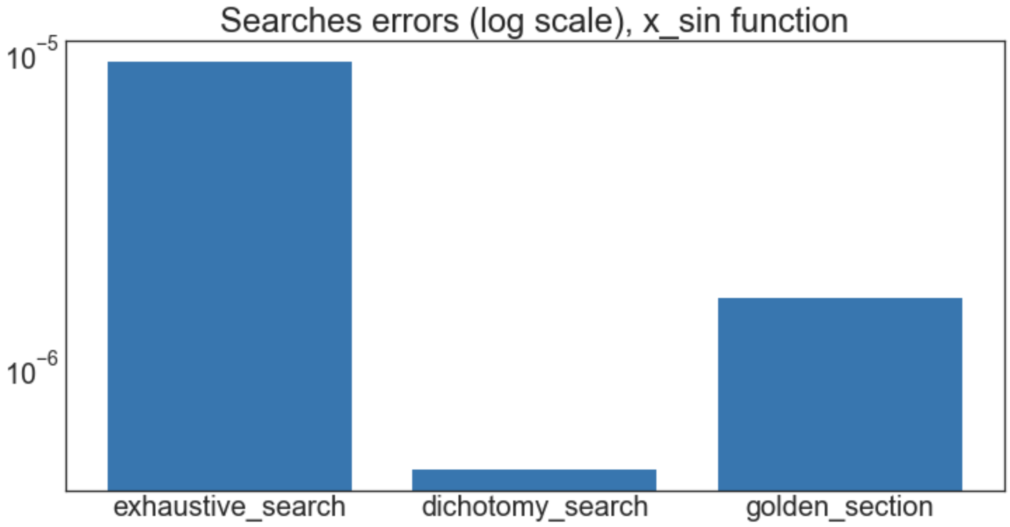
All algorithms converged to approximately the same minimal values of the functions. The calculated values are in agreement with theoretically expected values of minimums of the functions. Exhaustive search sometimes gives the best results, but it takes much more iterations and function calls than other algorithms. The best by number of iterations is dichotomy search, but it makes two function calculations on each iteration so the golden section method (makes 1 function calculation on each iteration) is the best in this way.

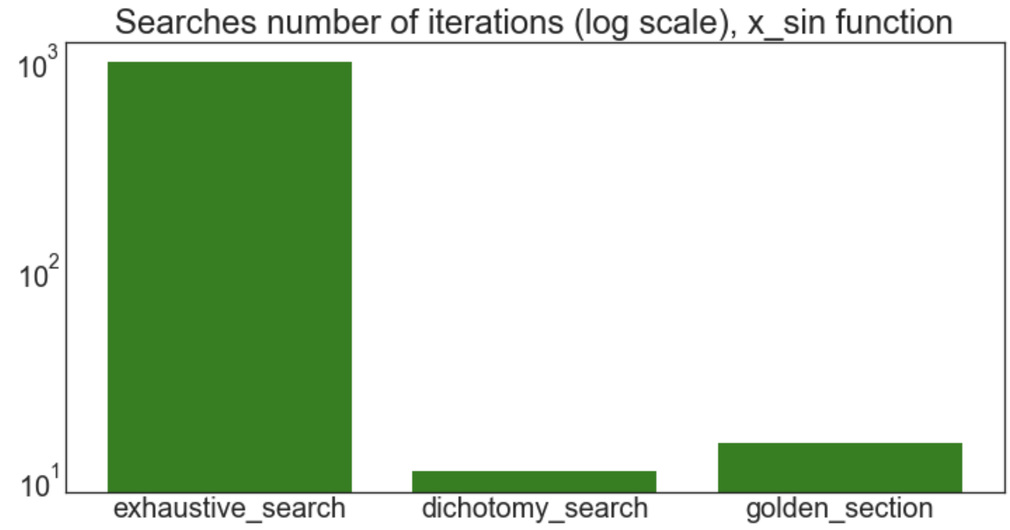
Below are graphs with the results of the first part of the study.

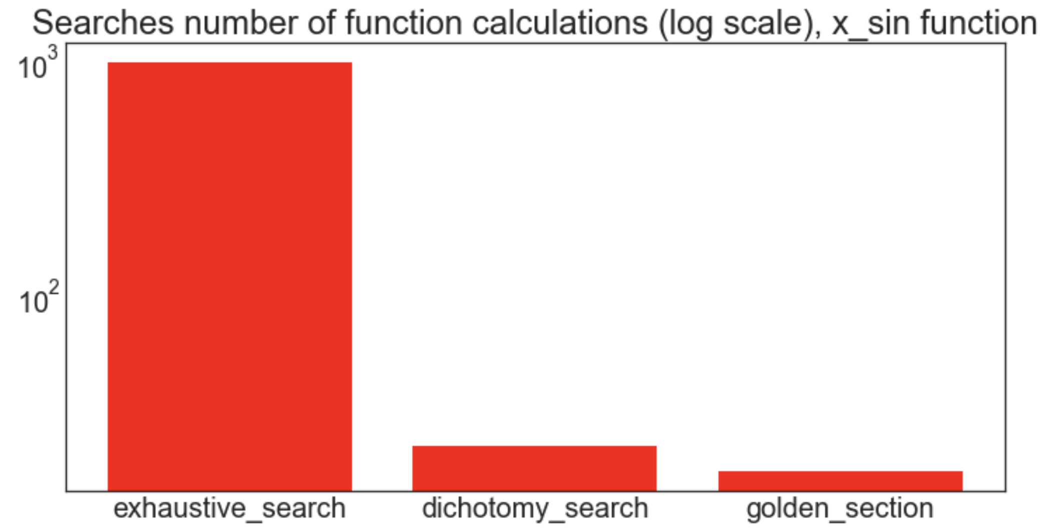

Searches errors (log scale), cubic function
10^{-10}
6×10^{-11}
exhaustive_search
dichotomy_search
golden_section
Searches number of iterations (log scale), cubic function
10^3
10^2
10^1
exhaustive_search
dichotomy_search
golden_section
Searches number of function calculations (log scale), cubic function
10^3
10^2
exhaustive_search
dichotomy_search
golden_section



Searches errors (log scale), module function
10^{-4}
9×10^{-5}
8×10^{-5}
exhaustive_search
dichotomy_search
golden_section
Searches number of iterations (log scale), module function
10^{3}
10^{2}
10^{1}
exhaustive_search
dichotomy_search
golden_section
Searches number of function calculations (log scale), module function
10^{3}
10^{2}
exhaustive_search
dichotomy_search
golden_section







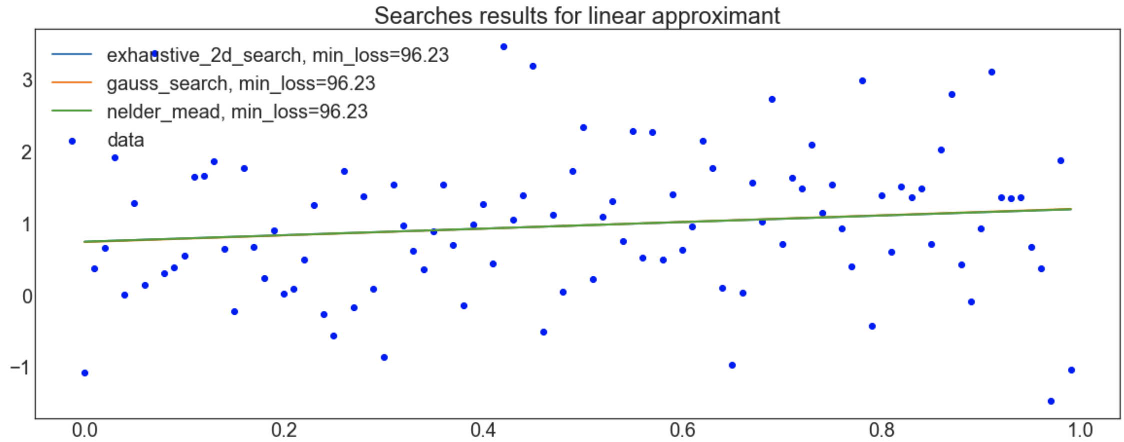
Two-dimensional optimization problem

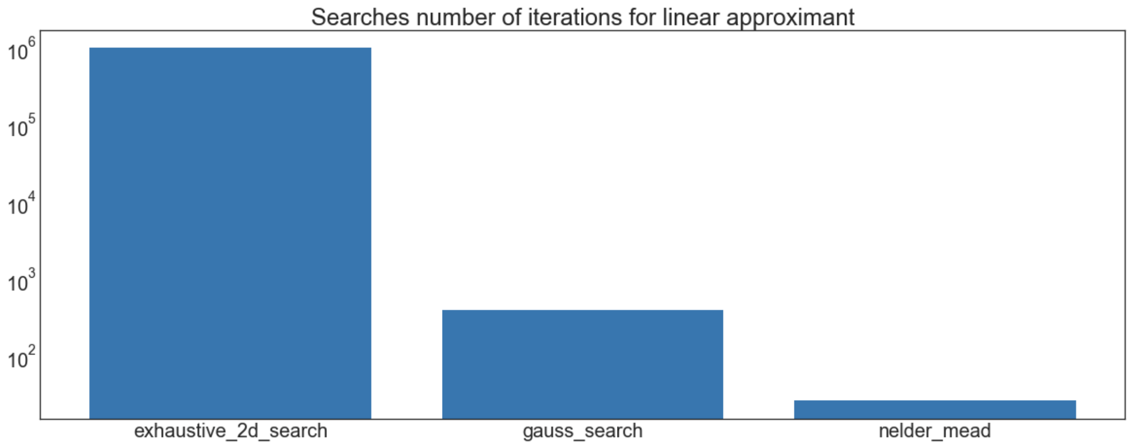
The Gauss method used golden-section search as a one-dimensional algorithm under the hood. For the Nelder-Mead algorithm, the scipy.optimize.minimize method was used.

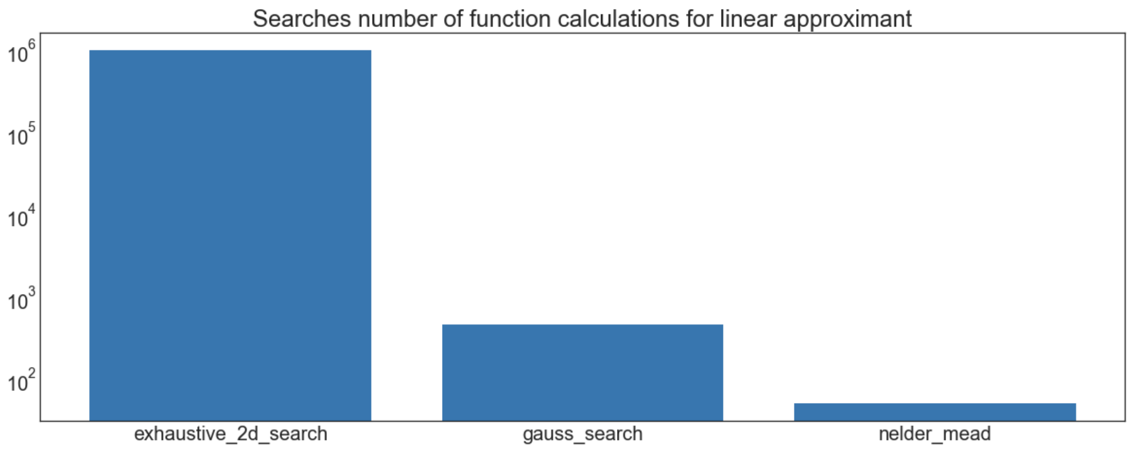
For **linear approximation**, results obtained by all three methods are almost the same. But brute-force method makes 10^6 iterations, while the Gauss method - about 500, and the Nelder-Mead algorithm - about 30 iterations. The number of function calls for these methods is about the same as iterations.

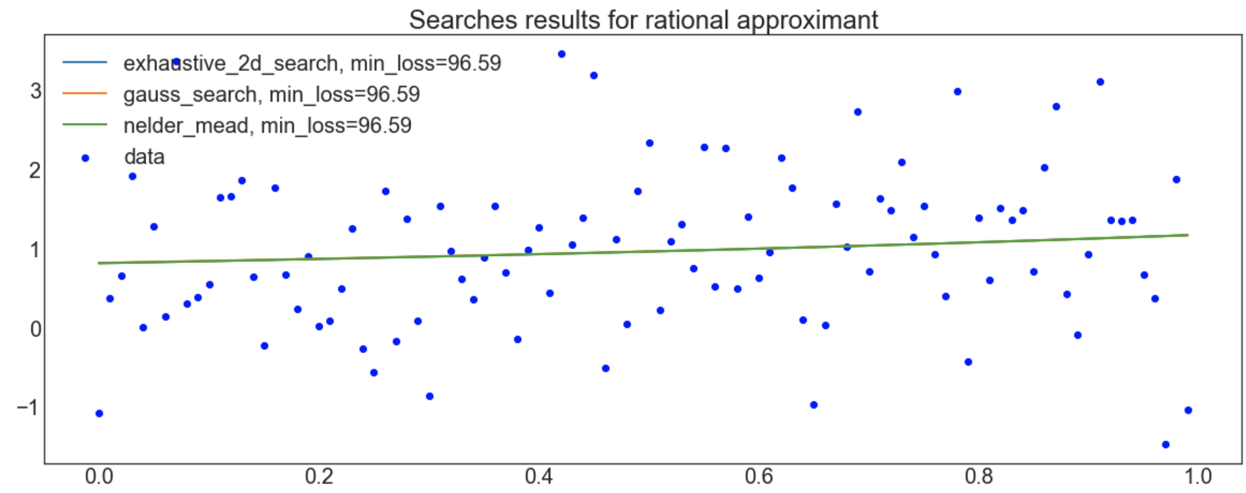
For **rational approximation**, results are also almost the same between methods and slightly worse than for linear approximation. Relations between numbers of iterations and numbers of function calculations are the same as in the linear approximation.

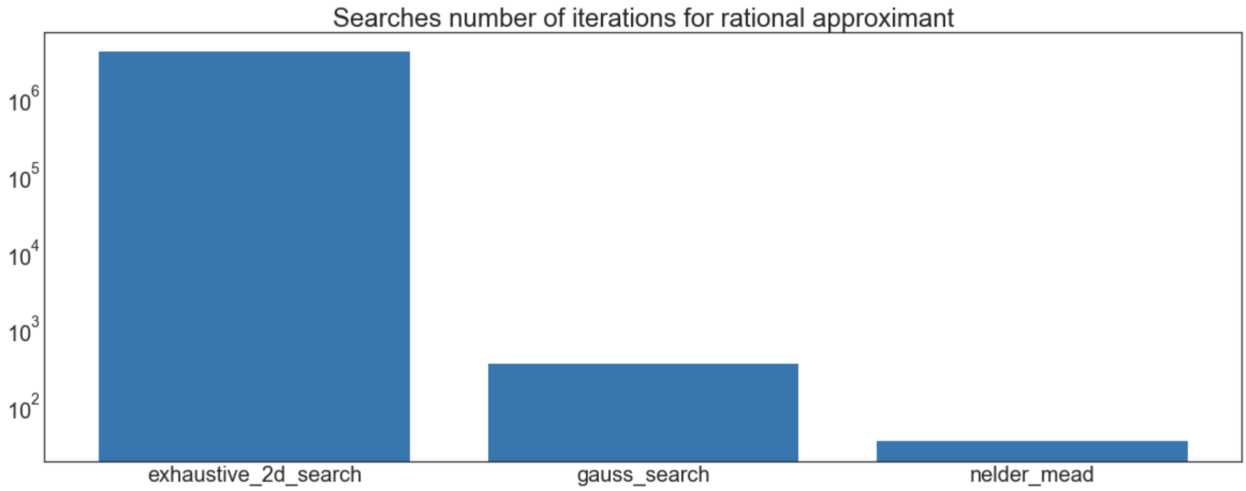
Below are graphs with the results of the second part of the study.

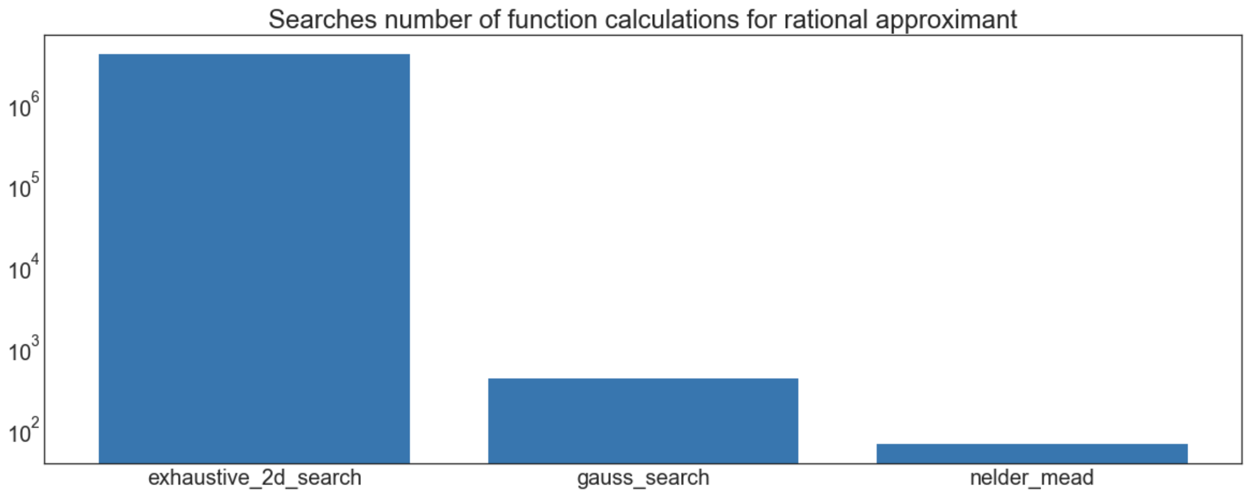












Conclusions

The goal of this study was to use direct methods of optimization in the one- and two-dimensional optimization tasks and compare obtained results for these algorithms. Analysis of the graphs presented in the Results section shows that all examined methods converge to optimal values, but brute-force methods during convergence make much more computations than more complex methods.

Appendix

Source code can be found at https://github.com/T1MAX/itmo_algorithms/tree/main/task_2