FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report
on the practical task No. 3

Algorithms for unconstrained nonlinear optimization. First- and second order methods

Performed by
Dmitry Grigorev, Maximilian Golovach
J4133c

Accepted by
Dr Petr Chunaev

St. Petersburg
2022

1. Goal of the work

The goal of this work is compound and consists of the following points:

- to become familiar with first- and second order methods in the problem of unconstrained nonlinear optimization;
- to apply the methods on the practical problem and compare them with each other;
- to compare the obtained results with the results of Task 2.

2. Formulation of the problem

Let $\alpha,\ \beta \in (0,1)$ are two arbitrary values, $x_k = \frac{k}{100},\ k = 0, \ldots, 100$ and y_k are defined by i.i.d. $\delta_k \sim N(0,1)$ according to the following rule:

$$y_k = \alpha x_k + \beta + \delta_k.$$

Given such two parametrized families of functions as:

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

we need to find the optimal function for given data $(x_k, y_k)_{k=0}^{100}$ according to the least squares:

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

using

- gradient descent,
- conjugate gradient descent,
- Newton's method,
- Levenberg-Marquardt method.

All functions which will be obtained have to be visualized with the given data and the line which generates these data. Furthermore, we have to compare the algorithms in terms of precision, number of iterations and number of function evaluations using the results.

3. Brief theoretical part

Suppose that we are given with a function $f: G \subset \mathbb{R}^n \to \mathbb{R}$ where G is connected open set in $\mathbb{R}^n$. We need to minimize this function on G.

3.1. Gradient descent method

Let the function f to be once continuously differentiable on G ($f \in C^1(G)$). The gradient descent method utilizes the fundamental fact that the gradient of function ∇f in a point is directed towards the fastest growth of the function among all other directions (**steepest ascent**) and, vice versa, $-\nabla f$ — the direction of the **steepest descent**. At each step of the algorithm next point is resulted by moving from the previous point in the direction of the antigradient at this point.

The gradient descent has such parameters as initial point x_0 and α which is called. **learning rate**. Researcher can both set fixed value to α, decrease it as the progress of optimization lasts or find optimal value at each step by means of line search methods (for instance, by means of Exhaustive Search or Golden Section Search).

Despite this method is simple, it has such problems as 1) being prone to criss-cross pattern of moves between two consequential points in the valleys of function what slows the rate of convergence and 2) slow convergence near the points where $\nabla f(x) \approx 0$.

Algorithm 1 Gradient descent method algorithm

Require: $f \in C^1(G)$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.

$k \leftarrow 0$

while $\|x_k - x_{k-1}\| > \varepsilon$ **do**

$k \leftarrow k + 1$

$x_k \leftarrow x_{k-1} - \alpha \nabla f(x_{k-1})$

end while

Ensure: $\widehat{x} = x_k$ — point of local minimum or critical point

3.2. Conjugate gradient descent

Originally, the conjugate optimization method was introduced for solving the linear system of equations problem with symmetric positive-definite matrix. The main idea of this method is that the twice differentiable function with non-degenerate second derivative at given point behaves like a quadratic paraboloid.

Let $f \in C^2(G)$, $H = \frac{d^2 f}{dx^2}$ and we are given with an initial point $x_0 \in G$ and precision $\varepsilon > 0$. At each step in point x_k the algorithm uses not only the direction of the steepest descent $-\nabla f(x_k)$ but the additional direction s_k which is conjugated to s_{k-1} in the sense of orthogonality with respect to $\mathbf{H}_{k-1} = H(x_{k-1})$:

$$s_k^T \mathbf{H}_{k-1} s_{k-1} = 0.$$

The conjugacy is obtained by means of the following equation which defines s_k:

$$s_k = -\nabla f(x_k) + \beta_k s_{k-1}.$$

Algorithm 2 Conjugate gradient descent algorithm

Require: $f \in C^2(G)$, $H = \frac{d^2 f}{dx^2}$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.

$k \leftarrow 0$

$g_0 \leftarrow -\nabla f(x_0)$

$s_0 \leftarrow g_0$

$\alpha_0 \leftarrow \arg\min_{\alpha > 0} f(x_0 + \alpha s_0)$

$x_1 \leftarrow x_0 + \alpha_0 s_0$

while $\|x_k - x_{k-1}\| > \varepsilon$ **do**

$k \leftarrow k + 1$

$g_k \leftarrow -\nabla f(x_k)$

Obtain β_k from the known formula (see below)

$s_k \leftarrow g_k + \beta_k s_{k-1}$

$\alpha_k \leftarrow \arg\min_{\alpha > 0} f(x_k + \alpha s_k)$

$x_k \leftarrow x_k + \alpha_k s_k$

end while

Ensure: $\widehat{x}$ — point of local minimum or critical point

There are several ways to calculate β_n. The classical one is **Fletcher-Reeves**:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$

The another one is **Polak–Ribière**:

$$\beta_k = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}$$

The advantage in using Polak–Ribiere formula is that it automatically reset its moves (i.e. $s_k \approx g_k$) when little progress is made over the last iteration. This property can speed up the convergence near the solution point [1].

The conjugate gradient methods have such an advantage as no matrix operations are required in their algorithms but they are not robust. These methods fix the problem of gradient descent with behavior in function valleys.

3.3. Newton's method

Newton's method is the representative of **Trust region approach algorithms**. The trust region approach assumes that the objective function f is well approximated by a quadratic function $q_k(\delta)$ obtained by truncating the Taylor series for $f(x_k + \delta)$. In the case of Newton's approach q_k is as follows:

$$f(x_k + \delta) \approx q_k(\delta) = f(x_k) + g_k^T \delta + \frac{1}{2}\delta^T \mathbf{H}_k \delta,$$

where $g_k = \nabla f(x_k)$, $\mathbf{H}_k = H(x_k) = \frac{d^2 f}{dx^2}(x_k)$. The next point x_{k+1} is resulted from the minimization of $q_k(\delta)$ which is equivalent to solving the system $\mathbf{H}_k \delta = -g_k$. The main assumption here is that $\mathbf{H}_k$ have to be positive definite for all k, otherwise the method can converge to point which is not a point of local minimum.

Algorithm 3 Newton's method algorithm

Require: $f \in C^2(G)$, $H = \frac{d^2 f}{dx^2}$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.

$k \leftarrow 0$

while $\|x_k - x_{k-1}\| > \varepsilon$ **do**

$g_k \leftarrow \nabla f(x_k)$

$\mathbf{H}_k \leftarrow H(x_k)$

$\delta_k \leftarrow -\mathbf{H}_k^{-1} g_k$

$x_{k+1} \leftarrow x_k + \delta_k$

$k \leftarrow k + 1$

end while

Ensure: $\widehat{x}$ — point of local minimum or critical point

Newton's method may fail to converge when $\mathbf{H}_k$ is not positive definite.

3.4. Levenberg-Marquardt method

Levenberg-Marquadt method[1] is called to solve the problem of Newton's method in non-guaranteed positive definiteness of $\mathbf{H}_k$. Here δ_k is computed from the system:

$$(\mathbf{H}_k + \nu \mathbf{I})\delta_k = -g_k,$$

[1] for sure, there is another Levenberg-Marquardt algorithm which uses Jacobian in its calculations.

where $\nu \geq 0$ is chosen so that the matrix $\mathbf{H}_k + \nu\mathbf{I}$ is positive definite. This method is a combination of Newton's and Gradient Descent methods so it is possible to adjust the behavior of the algorithm depending on the properties of function in current point. It is possible to choose the optimal parameter ν by any method among line search methods:

$$\nu^* = \arg\min_{\nu \geq 0} f(\delta_k(\nu)),\ \delta_k(\nu) = -(\mathbf{H}_k + \nu\mathbf{I})^{-1} g_k.$$

4. Results

First of all, the data $(x_k, y_k)_{k=0}^{100}$ as required in the formulation of the problem section 2 are obtained with $\alpha = 0.785$ and $\beta = 0.31$. We need to approximate them by a linear function F and by a rational function with precision $\varepsilon = 10^{-3}$.

4.1. The case of $F(x, a, b) = ax + b$

The appropriate line $F(x, a, b)$ was found by means of least squares with applications of all aforementioned methods. In the graph 1 the contour plot for the function $D(a, b)$ is demonstrated. It is expected that all these methods' solutions will be close to the vertix of this paraboloid and the corresponding lines will be close to the line $\alpha x + \beta$ which generated the data. The point $x_0 = (a_0, b_0) = (0.1, 0.1)$ was chosen as the initial point for each method. The stop criteria for all considered methods are **number of iterations** and **closeness of two consequential points**.

Since the explicit function expression is provided, one can obtain all necessary derivatives in this task.

$$\frac{\partial D}{\partial a} = 2\sum_{k=0}^{100} x_k(ax_k + b - y_k),\ \frac{\partial D}{\partial b} = 2\sum_{k=0}^{100}(ax_k + b - y_k),$$

$$\frac{\partial^2 D}{\partial a^2} = 2\sum_{k=0}^{100} x_k^2,\ \frac{\partial^2 D}{\partial a \partial b} = 2\sum_{k=0}^{100} x_k y_k,\ \frac{\partial^2 D}{\partial b^2} = 202.$$

In the Gradient Descent, Conjugate Gradient Descent and Levenberg-Marquardt methods we used Golden Section Search method as line search for their parameters on each step. This is correct since the function is convex. As a result we obtained solutions which are illustrated in the graph 2. All of these solutions are too close for sure to each other as expected and these lines well approximate the line $\alpha x + \beta$.

Now we should analyze the results with respect to precision, function evaluations (here f-eval., number of iterations, number of gradient (∇f-eval.) and hessian evaluations

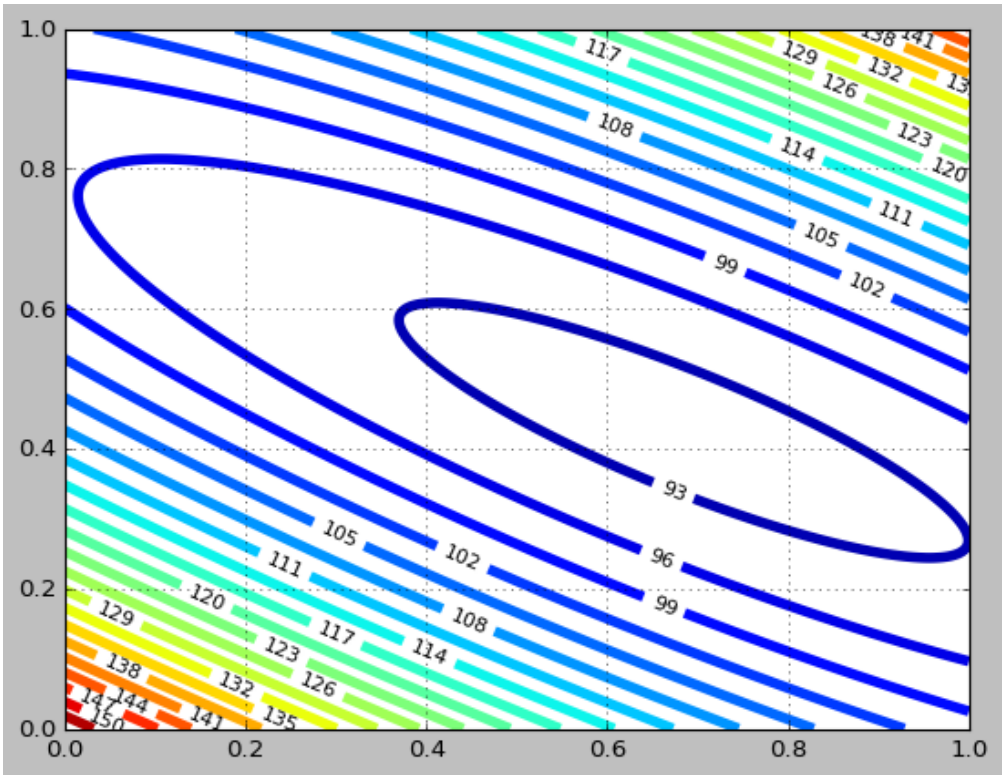

Figure 1: The graph of f(x)

(∇^2f-eval.) and number of matrix inversions. These indicators are presented in the table 1.

	Iterations	f-eval.	∇f-eval.	$\nabla^2 f$-eval	matrix inv.	precision
Gradient Descent	7	126	7	–	–	≈ 0.154
Conjugate Gradient Descent	4	73	5	–	–	≈ 0.154
Newton	2	0	2	2	2	≈ 0.154
Levenberg-Marquardt	2	46	2	2	42	≈ 0.154

Table 1: Algorithms' indicators in the case of linear approximation.

As one can see, Gradient Descent method required more function evaluations than other methods and more iterations to converge. It is connected with disappearance of gradient near the minimum point. In the case of Conjugate Gradient Descent there are less iterations were required as a result of correction of ordinary Gradient Descent problems. Two other methods undergo with inversion of matrices. In Levenberg-Marquardt algorithm inversion of matrices are related to line search of optimal parameter ν. Nevertheless, both these methods

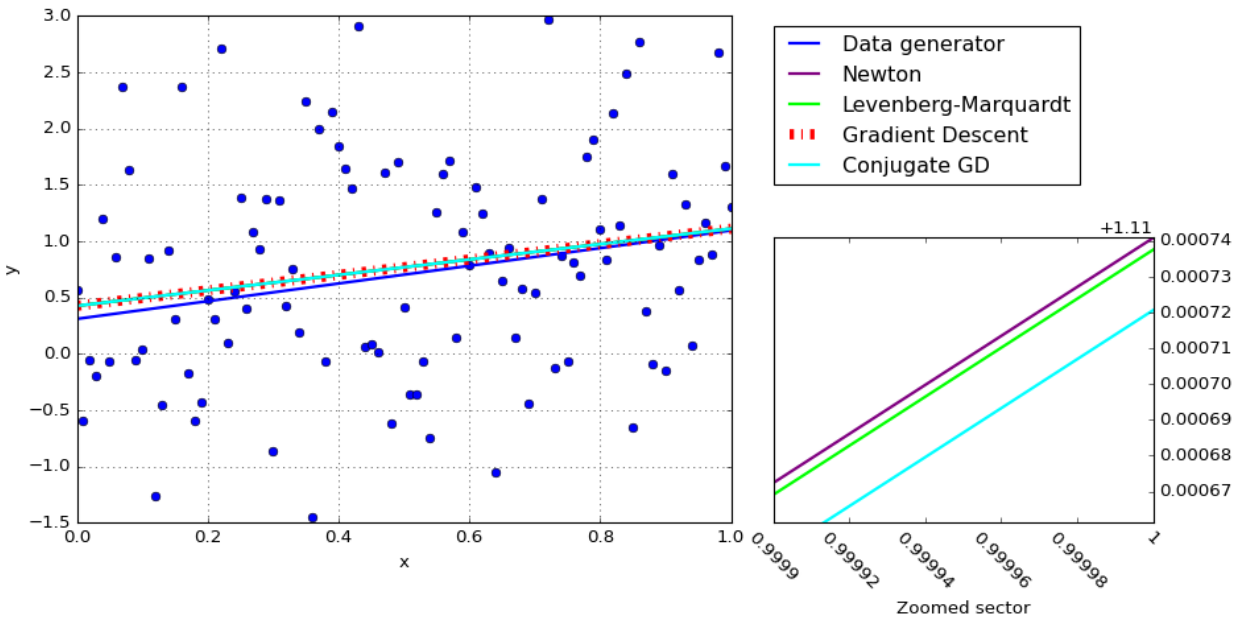

Figure 2: The data, the line which generated these data and fitted lines of linear functions. Since the lines are too close, the additional plot with zoomed sector is provided on the right size of the figure.

converged quite fast since they use more information about the optimizing function but the inversion of matrices might be expensive in the problems of higher dimensions. It should be mentioned that these algorithms converged more quickly to the same solution as opposed to the zero-order methods which were considered in the **Task 2**.

4.2. The case of $F(x, a, b) = \frac{a}{1+bx}$

Here are also 4 aforementioned methods were used to find the optimal curve. In the graph 2 the contour plot for the function $D(a, b)$ in this case is provided. The optimum is located on the edge of the area $\{0 < a, b < 1\}$ and this is the point $(mean(y), 0)$ where $mean(y) \approx 0.769$ is average of $y_{k=0}^{101}$ as was derived in the **task** 2. It is expected that all these methods' solutions will be close to this point and the corresponding lines will be close to the horizontal line $b = mean(y)$. The point $x_0 = (a_0, b_0) = (0.6, 0.2)$ was chosen as the initial point for each method. The main problem here is that every method is prone to leave the borders of given area and that is why we have to return optimization process to the area. The stop criteria for all considered methods are **number of iterations** and **closeness of two consequential points**.

As the explicit function expression is provided, one can obtain all necessary derivatives in this task.

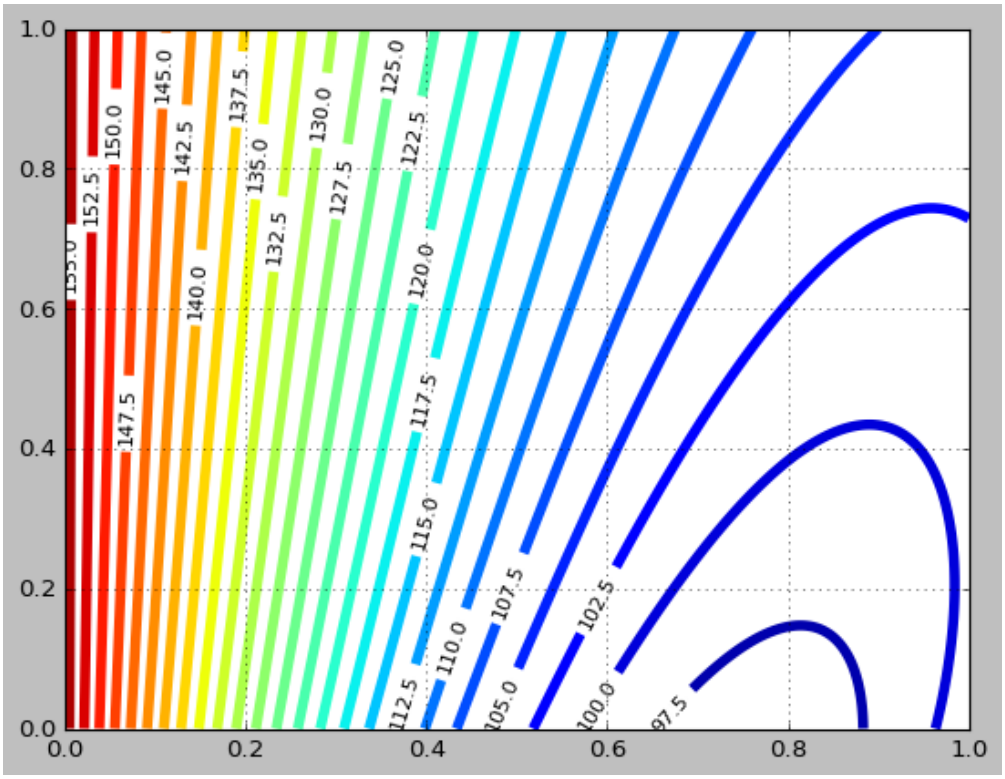

Figure 3: The graph of f(x)

$$\frac{\partial D}{\partial a} = 2\sum_{k=0}^{100} \frac{1}{1+bx_k}\left(\frac{a}{1+bx_k} - y_k\right), \quad \frac{\partial D}{\partial b} = -2\sum_{k=0}^{100} \frac{ax_k}{(1+bx_k)^2}\left(\frac{a}{1+bx_k} - y_k\right),$$

$$\frac{\partial^2 D}{\partial a^2} = 2\sum_{k=0}^{100} \frac{1}{(1+bx_k)^2},$$

$$\frac{\partial^2 D}{\partial a \partial b} = 2\sum_{k=0}^{100} \frac{1}{(1+bx_k)^2}\left(\frac{-2ax_k}{1+bx_k} + x_k y_k\right),$$

$$\frac{\partial^2 D}{\partial b^2} = 2\sum_{k=0}^{100} \frac{a^2 x_k^2}{(1+bx_k)^3}\left(\frac{3}{1+bx_k} - 2y_k\right).$$

In the Gradient Descent, Conjugate Gradient Descent and Levenberg-Marquardt methods we used Golden Section Search method as line search for their parameters on each step. The use of Golden Section is correct because the target function is convex in the considered region as has been seen in the graph. As a result we obtained solutions which are illustrated in the graph 4. All of these solutions are close to each other as expected and these lines span near the aforementioned horizontal line.

Le us analyze the results of applied methods with respect to function evaluations (here f-eval., number of iterations, number of gradient (∇f-eval.) and hessian evaluations

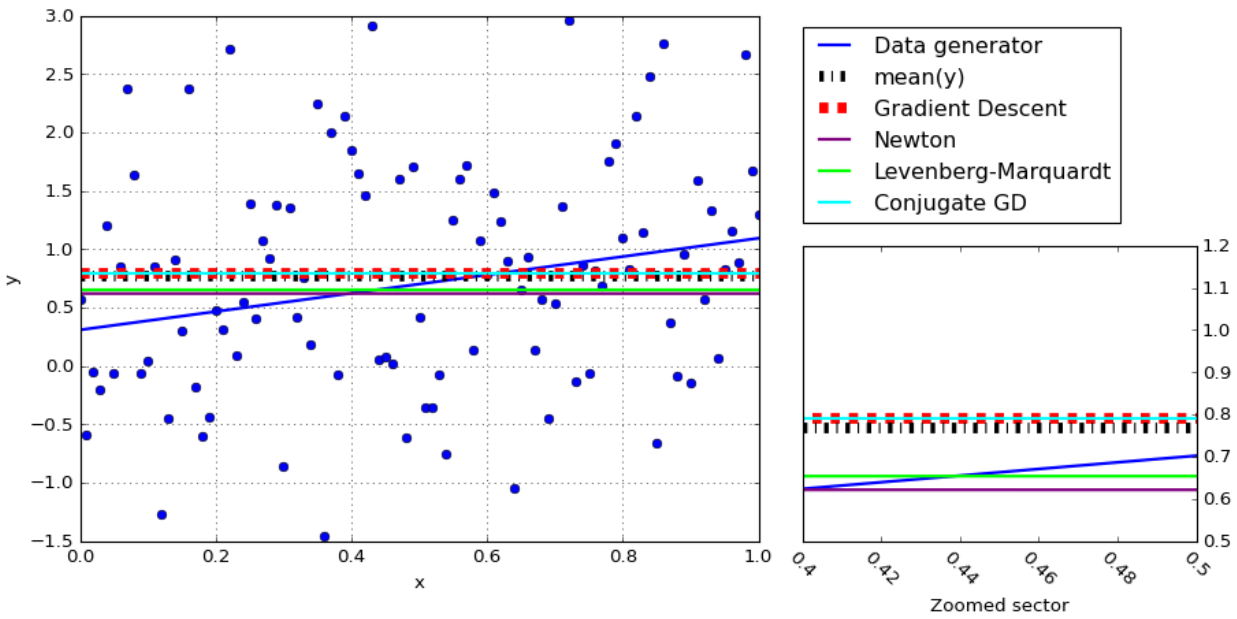

Figure 4: The data, the line which generated these data and fitted lines of rational functions. Since the lines are too close, the additional plot with zoomed sector is provided on the right size of the figure.

(∇^2f-eval.) and number of matrix inversions. Measurements of these indicators are provided in the table 2.

	Iterations	f-eval.	∇f-eval.	$\nabla^2 f$-eval	matrix inv.
Gradient Descent	36	650	36	–	–
Conjugate Gradient Descent	37	669	38	–	–
Newton	3	0	3	3	3
Levenberg-Marquardt	2	46	2	2	42

Table 2: Algorithms' indicators in the case of rational approximation.

As one can see, two gradient methods required too much iterations and function evaluations to converge. Nonetheless, their corresponding lines practically coincide with the horizontal line. Two other methods stuck in other point and did not reach this line but converged with less number of iterations. It is worth mentioning that Levenberg-Marquardt method did lots of matrices inversions in order to find optimal parameter ν. Besides, these algorithms here also converged more quickly to their solutions as opposed to the zero-order

methods which were considered in the **Task 2** but here second-order methods did not converge to optimum due to being stuck at the border.

5. Data structures and design techniques used in algorithms

In the implementation of the algorithms and for random variables generation the Python's package Numpy is used since it allows to vectorize calculations what accelerates the evaluations.

6. Conclusion

As the result of this work, we considered first- and second-order methods of optimization in application two the task of data approximation. The methods were compared with each other and with methods from the **Task 2** by the number of iterations required to obtain the solution with given precision, the number of function and its derivatives evaluations and precision. The results are satisfactory. All of these methods were implemented in Python programming language. Discussion for data structures and design techniques which were used in the implementations is provided. The work goals were achieved.

7. Appendix

Algorithms implementation code is provided in [2].

Bibliography

1. Cavazzuti M. Optimization Methods. — Springer Berlin, Heidelberg, 2013. — ISBN: 9783642311864.
2. Grigorev D., Golovach M. Code repository. — `https://github.com/dmitry-grigorev/AlgoAnalysisDevelopment`. — 2022.
3. Deisenroth M. P., Faisal A. A., Ong C. S. Mathematics for Machine Learning. — Cambridge University Press, 2020.