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

Rami Al-Naim

J4132c

Accepted by

Dr Petr Chunaev

St. Petersburg
2020

1 Goal

The use of first- and second-order methods (Gradient Descent, Conjugate Gradient Descent, Newton's method and Levenberg-Marquardt algorithm) in the tasks of unconstrained nonlinear optimization.

2 Formulation of the problem

Numbers $\alpha \in (0, 1)$ and $\beta \in (0, 1)$ were generated. Furthermore, the noisy data x_k, y_k generated, where $k = 0, ..., 100$, according to the following rule:

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

where $\delta\tilde{N}(0, 1)$ are values of a random variable with standard normal distribution. These noised data were approximated by the following linear and rational functions: 1. $F(x, a, b) = ax + b$ (linear approximant), 2. $F(x, a, b) = a1 + bx$ (rational approximant),

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

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

The algorithms were compared based on execution time, number of iterations and mean squared error.

3 Brief theoretical part

Gradient Descent is a first-order minimization algorithm. It updates the prediction in direction opposite to the gradient (direction of the fastest direction increase) and magnitude of it. During this work this algorithm was implemented in Python.

Conjugate Gradient Descent is a method similar to Gradient Descent. On each step the direction is calculated with respect to target function's gradient and direction on previous step. The implementation of Conjugate Gradient Descent from `scipy` module is used.

Newton's method is a second-order method of optimization. It uses information about target function's curvature (second-order derivative) and Taylor's expansion. The implementation of the method from `scipy` module is used.

Levenberg-Marquardt algorithm is a second-order optimization method which minimizes the sum of squared errors and widely used in problems of multidimensional curve-fitting problem. The Levenberg-Marquardt algorithm's implementation from `scipy` module were used.

4 Results

All algorithms were used to optimize the same noised data. The statistics of mean squared error, number of iterations and elapsed time were collected and presented in Table 1 for linear approximation and in Table 2. The results of linear approximation is present in the Figure 1 and Figure 2 for rational approximation.

4.1 Linear approximation

Methods	MSE	# of iterations	elapsed_time
Gradient Descent	0.007529	1000	23.90
Conjugate Gradient	0.007529	27	7.51
Levenberg-Marquardt	0.007529	12	7.62
Newton	0.007529	3	10.00

Table 1: Comparison of different linear approximation methods

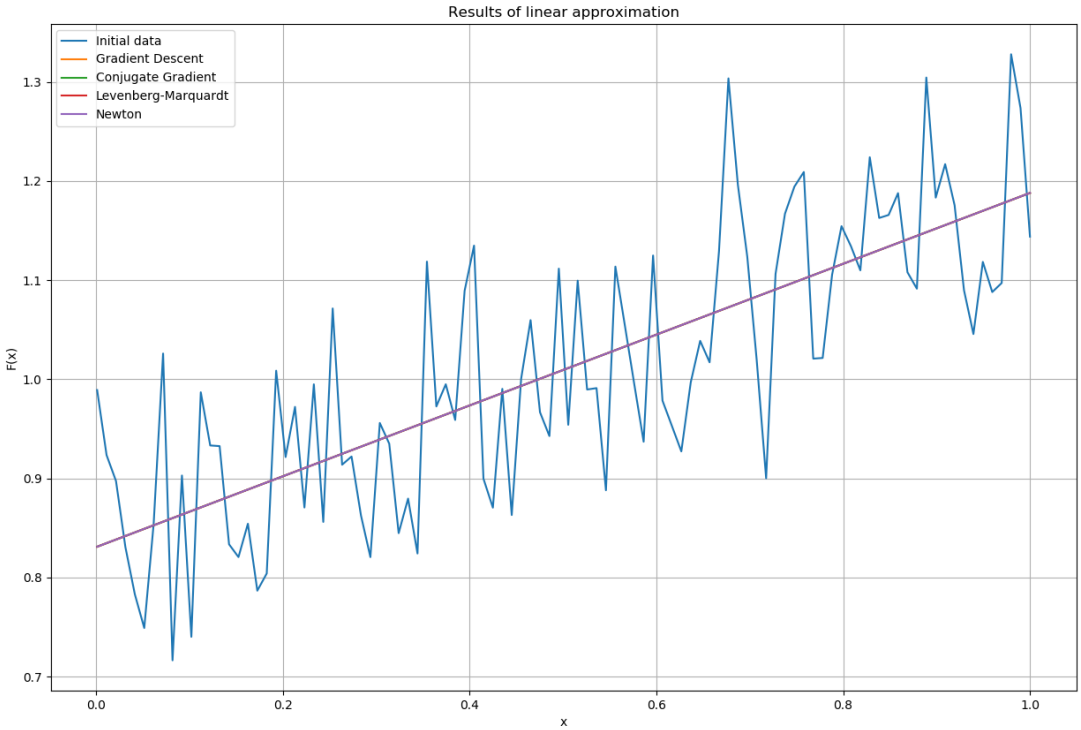

Figure 1: Results of linear approximation

4.2 Rational approximation

Methods	MSE	# of iterations	elapsed_time
Gradient Descent	1.189139	1000	23.90
Conjugate Gradient	0.007487	35	7.51
Levenberg-Marquardt	0.007487	15	7.62
Newton	0.007487	8	10.00

Table 2: Comparison of different rational approximation methods

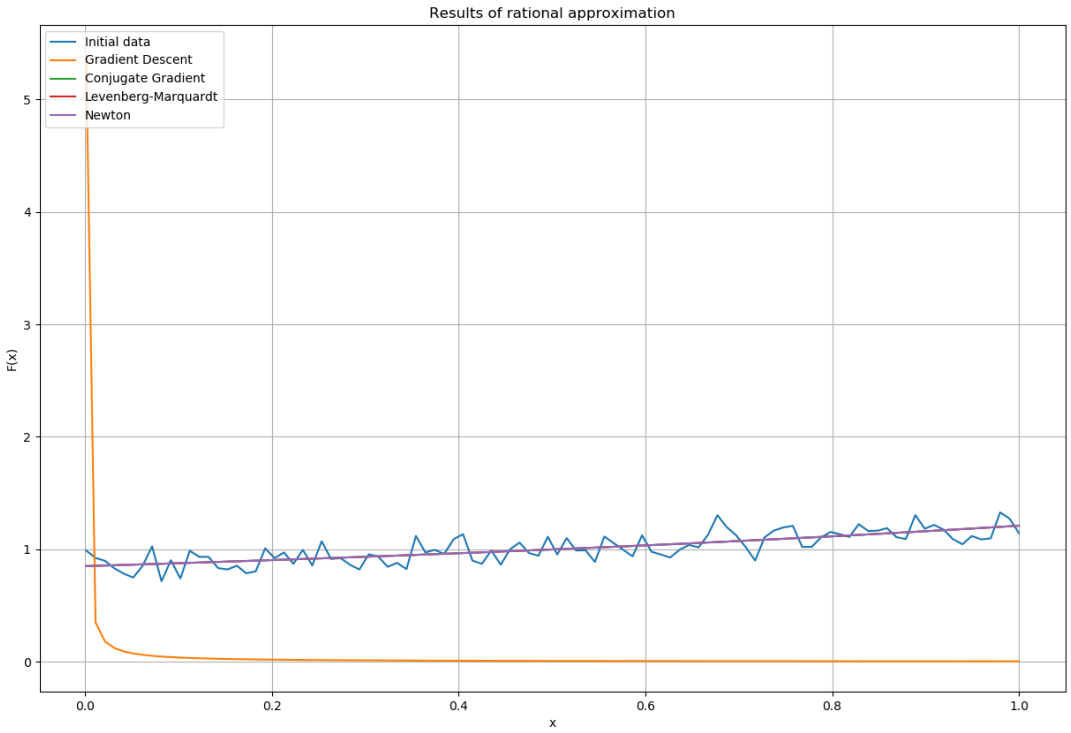

Figure 2: Results of rational approximation

5 Conclusions

During this work optimization algorithms of first-order (Gradient Descent and Conjugate Gradient) and second-order (Levenberg-Marquardt and Newton) were compared with each other on the same noisy data.

From the Figure 1 and Table 1 it can be seen that all methods apart from Gradient Descent provided the same MSE for linear approximation. Conjugate Gradient Descent showed the best execution time and Newton method find the solution using the least amount of iterations.

Similar to the linear approximation case, from the Figure 1 and Table 1 it can be seen that all methods apart from Gradient Descent provided the same MSE for rational approximation. Again, Conjugate Gradient Descent showed the best execution time and Newton method find the solution using the least amount of iterations. But in case of rational approximation Gradient Descent provided visibly worse approximation and did not managed to find a minimum due to non linearity of a rational function.

Appendix

The Python code for this task can be found here: GitHub.