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
Mikhail Grigoryev (370852)
Semenova Valeria (370061)
Academic group J4133c
Accepted by
Dr Petr Chunaev

St. Petersburg
2022

Goal

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

Formulation of the problem

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

$$y_k = \alpha x_k + \beta + \delta_k, \quad x_k = 0.01k, \qquad \text{for the linear approximant,}$$

$$y_k = \frac{\alpha}{1+\beta x_k} + \delta_k, \quad x_k = 0.01k, \qquad \text{for the rational approximant,}$$

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

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

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

To solve the minimization problem, use the methods of Gradient Descent, Conjugate Gradient Descent, Newton's method and Levenberg-Marquardt algorithm. 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 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 iterations, etc.) and compare them with those from Task 2 for the same dataset.

Brief theoretical part

Optimization methods are essentially methods of finding optimal (depens on the task) values of target functions. Often optimization is just minimizing a certain function.

The solution to the optimization (minimization) problem is finding:

$$x^* \in Q : \quad f(x^*) = \min_{x \in Q} f(x) f(x)$$

Finding the argument which corresponds to the function minimum will let us find the value of the function at its minimum.

In this practical work local minima will be found, $f(x)$ can be nonlinear and no additional constrictions on the argument will be applied. Hence the title, ”unconstrained nonlinear optimization”.

In addition to that, in this work first- and second-order methods will be used, which utilize values of $f(x)$ and values of the first-order derivative $f'(x)$ (second-order methods additionally use values of the second derivative $f''(x)$). Those methods can be utilized on continuous functions, once or twice differentiable (depending on the order of the method).

Algorithms used in this practical work:

1. Gradient Descent was implemented from scratch with Barzilai-Borwein β-coefficients. Every iteration two function evaluations are made, thus for the total number of function evaluations the algoritm returns the number of iterations times two.
2. For Conjugated Gradient Descent the default Scipy.optimize.minimize implementation was used. The API provides options to return both numbers of function evaluations and iterations.
3. For Newton’s Method the default Scipy.optimize.minimize implementation was used. This algorithm technically is Quasi-Newton’s method, which means that unlike the full Newton’s method the inverse Hessian matrix H^{-1} is computed only approximately using numerical formulas (faster to compute than the Hessian itself). In this exact implementation, the method is Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.
4. For Levenberg-Marquardt Algorithm the Scipy.optimize.least_squares implementation was used. The API has no explicit way of returning the number of iterations. However, theoretically, every iteration of LM algorithm runs only one evaluation. Thus, in listing to this practical task the number of iterations is said to be equal to the number of function evaluations (which is returned by the API).

Results

Random parameters for generating noisy data were generated. They are shown below:

$$\alpha = 0.5488 \qquad \beta = 0.7152$$

Four methods of linear approximation were applied to the problem, the data obtained is presented in the table:

alg/param	(A, B)	Iterations	Function calls
Gradient Descent	(0.3256, 0.8573)	4	4
Conjugated Gradient Descent	(0.3249, 0.8577)	2	2
Quasi-Newton's Method	(0.3259, 0.8567)	2	2
Levenberg-Marquardt	(0.3248, 0.8577)	6	6
Exhaustive Search (Task 2)	(0.3243, 0.8579)	1000000	1000000
Gauss Search (Task 2)	(0.3285, 0.8555)	875	2520
Nelder-Mead Search (Task 2)	(0.3246, 0.8580)	29	56

All four methods gave close results, as expected (no non-linearity). For the same reason, those results were close to predictions of direct methods (same dataset, same α and β, same noise). Performance of first- and second-order methods was significantly better than of direct ones (methods needed no more than 6 iterations and function calls).

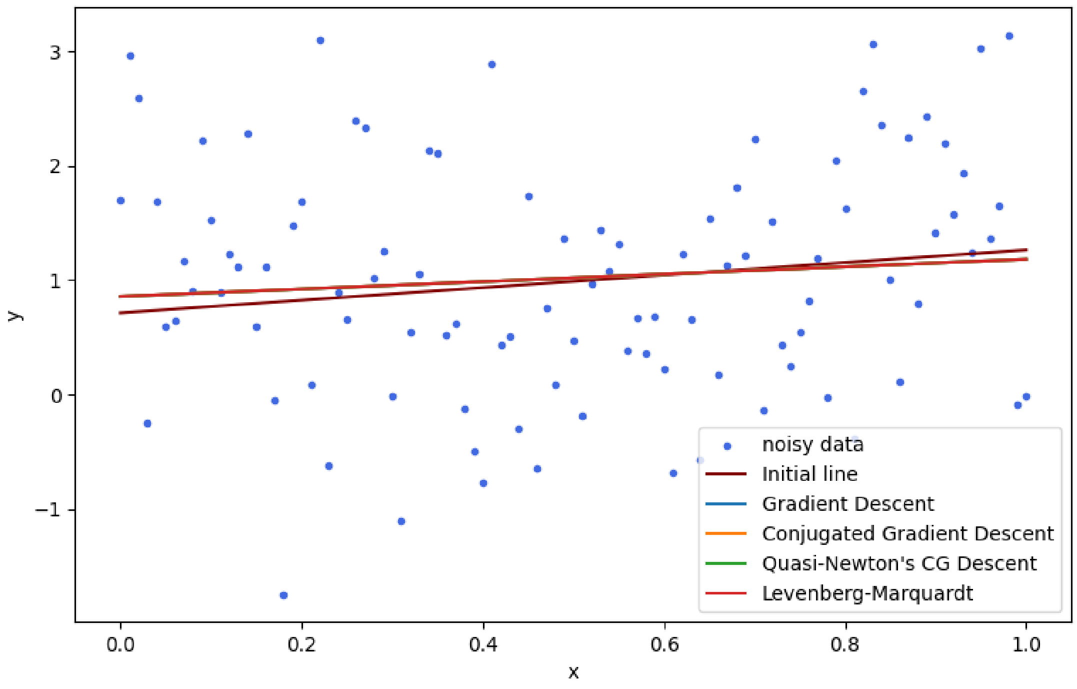

Figure 1: Results of applying Gradient Descent, Conjugated Gradient Descent, Quasi-Newton's Method and Levenberg-Marquardt Algorithm to solve the problem of linear approximation.

Those methods were again applied to solve the problem of rational approximation. Random parameters were:

$$\alpha = 0.9200 \qquad \beta = 0.9561$$

Obtained data is presented below:

alg/param	(A, B)	Iterations	Function calls
Gradient Descent	(0.9138, 0.8348)	9	18
Conjugated Gradient Descent	(0.9244, 0.8769)	10	57
Quasi-Newton's Method	(0.9219, 0.8686)	9	12
Levenberg-Marquardt	(0.9244, 0.8769)	13	13
Exhaustive Search (Task 2)	(1.0000, 0.0000)	1000000	1000000
Gauss Search (Task 2)	(1.0000, 0.0000)	50	144
Nelder-Mead Search (Task 2)	(1.1282, -0.4661)	43	83

Due to non-linearity, direct methods of Exhaustive Search and Gauss Search and Nelder-Mead Search converged to a minimum, different from all other methods, providing similar results. In terms of performance, again, methods utilizing derivatives converged significantly faster (taking only 10 to 13 iterations and up to 57 function calls) which contrasts slow direct methods.

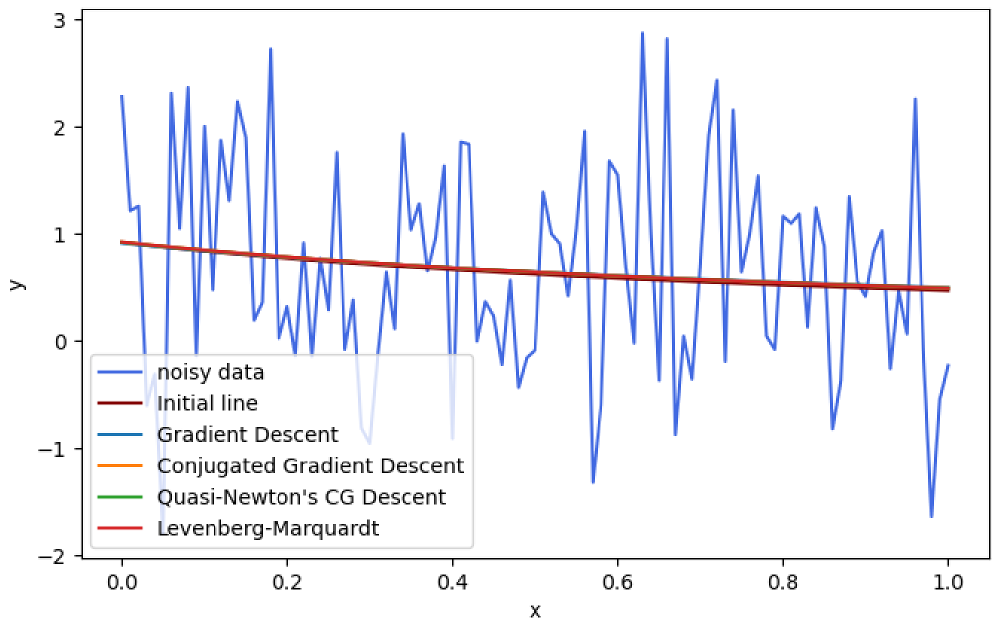

Figure 2: Results of applying Gradient Descent, Conjugated Gradient Descent, Quasi-Newton's Method and Levenberg-Marquardt Algorithm to solve the problem of rational approximation.

Conclusions

First- and second-order optimization method such as Gradient Descent, Conjugated Gradient Descent, Quasi-Newton's Method and Levenberg-Marquardt Algorithm were applied to tasks of unconstrained nonlinear optimization, tested against each other and direct methods and compared in terms of perfomance.

Appendix

GitHub link: https://github.com/Dormant512/itmo_lab_listings/blob/main/lab3.py.