FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report

on the practical task 3

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

Performed by

Ilya Lyalinov

Academic group J4134c

Accepted by

Dr Petr Chunaev

St. Petersburg

2021

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* $k = 0, \dots, 100$, *according to the following rule:*

$$y_k = \alpha x_k + \beta + \delta_k, \quad x_k = (1/100) * k$$

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. $F(x, a, b) = a / (1 + bx)$ *(rational approximant),*

by means of least squares through the numerical minimization (with precision eps = 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 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 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.) and compare them with those from Task 2 for the same dataset.*

Brief theoretical part

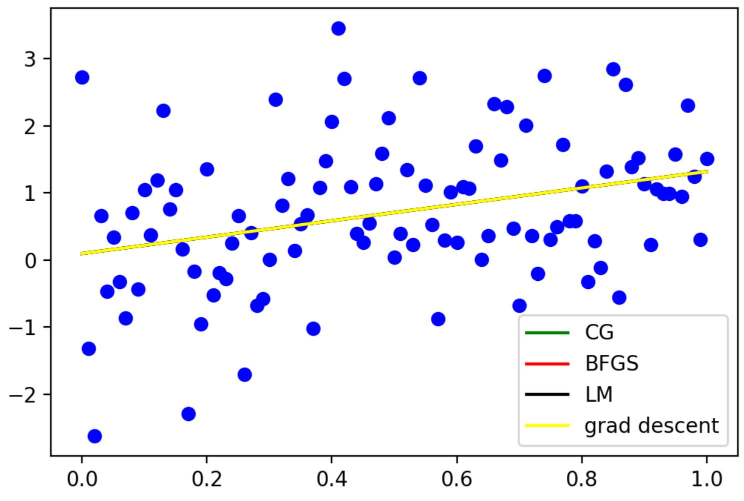
1) Gradient descent: method for finding a local minimum by moving along the gradient. At each iteration we obtain $x_j = x_{(j-1)} - \lambda * gradF(x_j)$ where x_j is our next approximate solution, *F* -- function that we try to minimize, *gradF* - is its gradient. Parameter λ may be constant or vary from step to step. In our implementation we use constant λ. Algorithm stops when

$|x_j - x_{j-1}| < eps$. Then x_j is obtained solution.

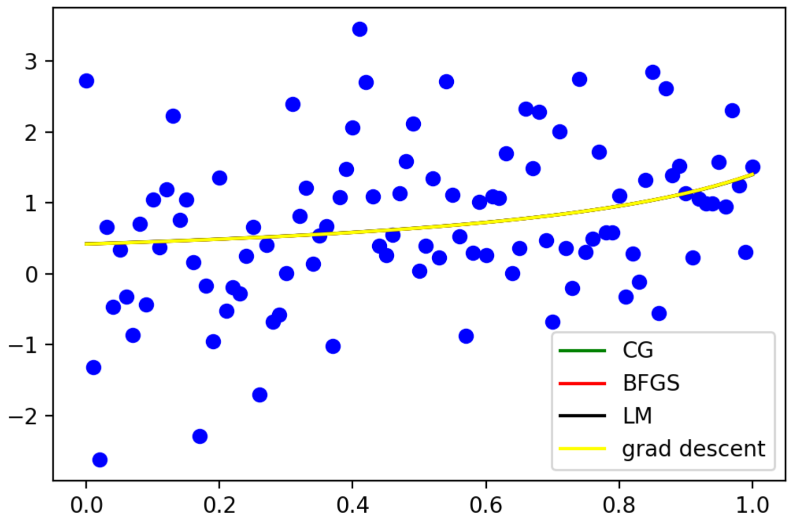
2) Conjugate gradient algorithm:
   It uses the same idea of movement in the opposite direction from the direction of the gradient at every step. But finds the next approximation using the information about the previous step direction. That allows to find the minimization point in fewer steps.
3) Newton's method:
   Newton's method approximates the function at a point by the quadratic part of its Taylor series. Quasi-Newtonian methods do not consider the Hesse matrix, but use its approximation. They also accumulate curvature information at each step and use it to find the next iteration's direction. Quasi-Newtonian methods are second-order pseudo-methods, so they are assumed to find the direction of motion more accurately and to solve the minimization problem faster than first-order methods (like gradient descent).
4) Levenberg-Marquardt algorithm:
   This algorithm uses the lambda parameter to change its behavior. With lambda going to infinity, this method behaves like a gradient descent. When the lambda is close to zero, it behaves like Newton's method. This behavior let one avoid the disadvantages of both Newton's and GD methods.

Results

We used quasi-Newtonian BFGS method instead of classical Newton's method.
Plot for linear approximation:



Plot for rational approximation:



Comparison of optimization algorithms (linear approximation):

Method	Number of iterations	$D(a_{min}, b_{min})$	Number of f-calculations	Number of gradf-calculations
CG	2	110.61	5	5
BFGS	4	110.61	6	6
LM	6	110.61	6	-
Gradient descent	1332	110.61	-	1332

Comparison of optimization algorithms (rational approximation):

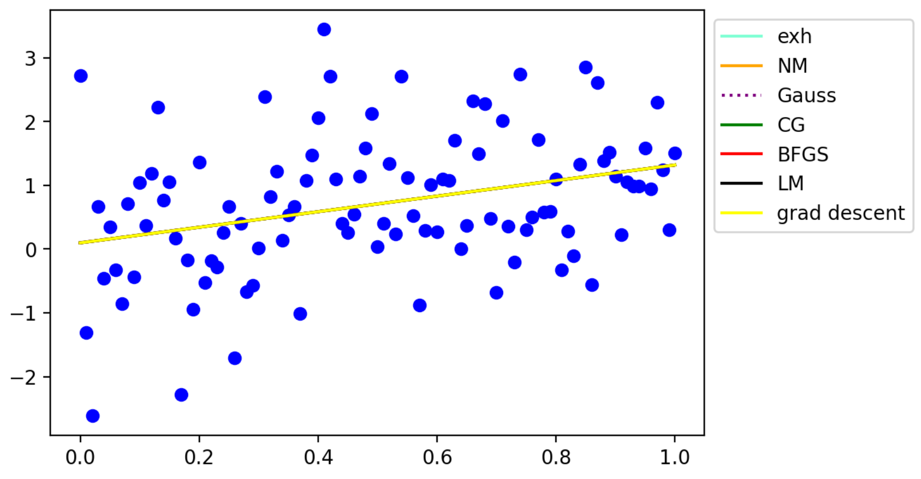
Method	Number of iterations	$D(a_{min}, b_{min})$	Number of f-calculations	Number of gradf-calculations
CG	8	113.98	22	22
BFGS	8	113.98	15	15
LM	27	113.98	27	-
Gradient descent	484	113.98	-	484

(a_{min}, b_{min}) is the obtained solution for minimization task. Linear optimization for all methods occurred faster than rational, excluding gradient descent. CG, as expected, required less iterations than gradient descent to finish. BFGS requires less *f*-evaluations than CG, given that the number of iterations are the same (see table for rational approximation). LM, as expected, is the fastest method in terms of *f*-calculations. The reason is that it eliminates the drawbacks of the Newton and gradient descent methods by selecting appropriate lambda parameter at each step.

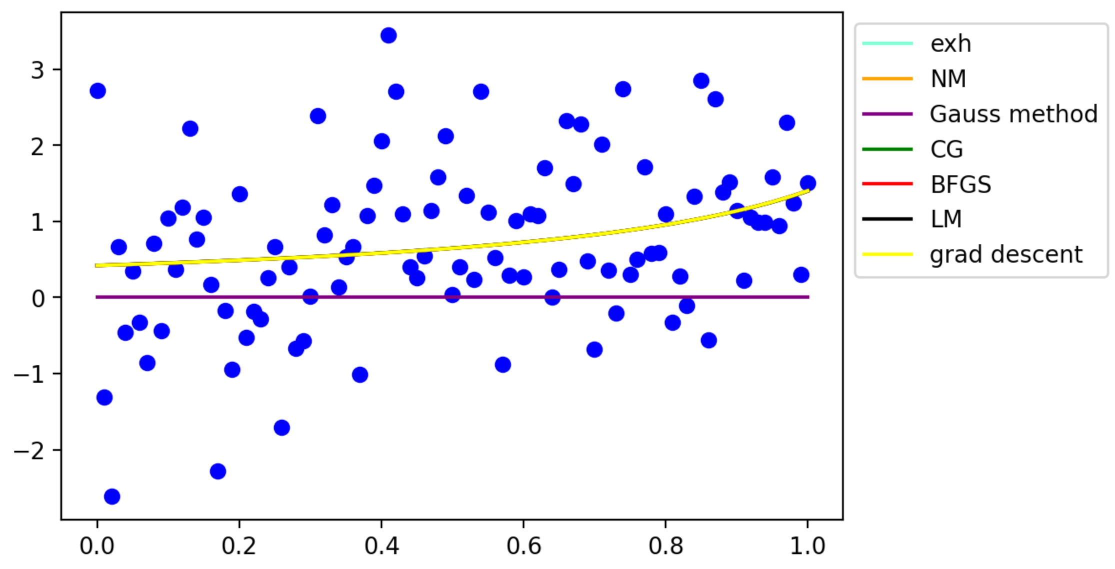
Comparison of optimization algorithms from 2nd and 3rd laboratory works:

Method	Number of iterations (linear approximant)	Number of iterations (rational approximant)	Number of f-calculations (rational approximant)	Number of f-calculations (rational approximant)
Exhaustive search	4001^2	$4001 * 2001$	4001^2	$4001 * 2001$
Nelder-Mead method	56	107	112	205
Gauss method	20	2	20 * 2 * 4001	2 * 2 * 4001
CG	2	8	5	22
BFGS	4	8	6	15
LM	6	27	6	27
Gradient descent	1332	484	1332 (gradf)	484 (gradf)

1st and 2nd order methods perform much faster than direct methods in terms of *f*-calculations on average. However, Nelder Mead's method turned out to be faster (in terms of f-calculations) for our initial parameters than the gradient descent, but slower than all other 1st and 2nd order methods.
Plot of approximations obtained with all methods (linear approximation):



Plot of approximations obtained with all methods (rational approximation):



Gauss method failed to find global minimum, but every other method performed almost the same, if we talk about the accuracy of the found parameters (a_{min}, b_{min}) (the maximum difference ofEuclidean norm of the minimization points obtained by different methods was less than epsilon).

Conclusion

We generated an array of noisy data and applied first and second-order nonlinear optimization methods to find the minimum of function D. We plotted obtained with each method approximations and analyzed the results in terms of number of iterations, *f*-evaluations, *gradf*-evaluations and value of D at the minimization point. We also compared performance characteristics of 1st and 2nd-order optimization algorithms with those characteristics of direct optimization methods of 2nd laboratory work.

Appendix

https://github.com/ilyalinov/Algorithms