FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION

OF HIGHER EDUCATION

ITMO UNIVERSITY

Report

on the practical task No. 3

“Task 3. Algorithms for unconstrained nonlinear optimization. First- and second-order methods”

Performed by

Alexandra Matveeva

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.

Problems

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:

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

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 approximation)

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 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

Gradient Descent

The **gradient** of a differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ at **a** is the n-dimensional **column**-vector $\nabla f(\boldsymbol{a})$ whose elements are

$$\frac{\partial f}{\partial x_i}|_a, i = 1, \dots, n.$$

Gradient descent method is based on the observation that if *f* is differentiable at **a**, then *f*(**x**) decreases **fastest** in a neighbourhood of **a** in the direction of $-\nabla f(\boldsymbol{a})$. One may write down the following formula:

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n - \beta_n \nabla f(\boldsymbol{a}_n), \beta_n > 0, n = 0, 1, \dots,$$

Starting with some initial approximation $\boldsymbol{a}_0$.

If step size β_n is chosen properly, then $f(\boldsymbol{a}_n) \geq f(\boldsymbol{a}_{n+1}) \geq f(\boldsymbol{a}_{n+2}) \geq \cdots$ and furthermore $\boldsymbol{a}_n \to x^*$ as $n \to \infty$, where x^* is a local minimum.

If *f* is convex, ∇f is Lipschitz and the choice of β_n is due to Barzilai-Borwein,

$$\beta_n^{BB} = \frac{|(\boldsymbol{a}_n - \boldsymbol{a}_{n-1})(\nabla f(\boldsymbol{a}_n) - \nabla f(\boldsymbol{a}_{n-1}))|}{\|\nabla f(\boldsymbol{a}_n) - \nabla f(\boldsymbol{a}_{n-1})\|^2},$$

then the uniform convergence is guaranteed.

(Nonlinear) Conjugate Gradient Descent

The conjugate gradient method is an iterative method for unconditional optimization in a multidimensional space. The main advantage of the method is that it solves a quadratic optimization problem in a finite number of steps.

$f: \mathbb{R}^n \to \mathbb{R}$ is a differentiable function, $\boldsymbol{a}_0$ is an initial approximation, one start in the steepest descent direction:

$$\Delta\boldsymbol{a}_0 = -\nabla f(\boldsymbol{a}_0).$$

Step size: $\alpha_0 \coloneqq \arg min_\alpha f(\boldsymbol{a}_0 + \alpha\Delta\boldsymbol{a}_0)$ and $\boldsymbol{a}_1 = \boldsymbol{a}_0 + \alpha_0\Delta\boldsymbol{a}_0$. After this iteration, the following steps with $n = 1, 2, \ldots$ constitute one iteration of moving along a subsequent conjugate direction s_n, where $s_0 = \Delta\boldsymbol{a}_0$:

- Calculate the steepest direction $\Delta\boldsymbol{a}_n = -\nabla f(\boldsymbol{a}_n)$

- Compute β_n according to certain formulas

- Update the conjugate direction $s_n = \Delta\boldsymbol{a}_n + \beta_n s_{n-1}$

- Find $\alpha_n = \arg min_\alpha f(\boldsymbol{a}_n + \alpha\boldsymbol{s_n})$

- Update the position: $\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \alpha_n\boldsymbol{s_n}$.

The choice of β_n (to guarantee the uniform convergence $\boldsymbol{a}_n \to x^*$ as $n \to \infty$ for convex *f* and Lipschitz ∇f) is due to Fletcher-Reeves or Polak-Ribiere:

$$\beta_n^{FR} = \frac{\Delta\boldsymbol{a}_n^T\Delta\boldsymbol{a}_n}{\Delta\boldsymbol{a}_{n-1}^T\Delta\boldsymbol{a}_{n-1}}, \beta_n^{PR} = \frac{\Delta\boldsymbol{a}_n^T(\Delta\boldsymbol{a}_n - \Delta\boldsymbol{a}_{n-1})}{\Delta\boldsymbol{a}_{n-1}^T\Delta\boldsymbol{a}_{n-1}}$$

Newton's method

$f: \mathbb{R} \to \mathbb{R}$ is a convex and twice differentiable. We should find a root of f' constructing a sequence a_n from an initial approximation a_0 so that $a_n \to x^*$ as $n \to \infty$, where $f'(x^*) = 0$.

From the Taylor expansion of *f* near a_n,

$$f(a_n + \Delta a) \approx T_f(\Delta a) \coloneqq f(a_n) + f'(a_n)\Delta a + \frac{1}{2}f''(a_n)(\Delta a)^2.$$

We use this quadratic function (with respect to Δa) as an approximant to f in a neighbourhood of a_n. The vertex of the corresponding parabola gives us the next point a_{n+1}. To find the vertex x-coordinate, we write:

$$0 = \frac{dT_f(\Delta a)}{d\Delta a} = f'(a_n) + f''(a_n)\Delta a \Rightarrow \Delta a = -\frac{f'(a_n)}{f''(a_n)}.$$

$f''(a_n) > 0$. Incrementing a_n by this Δa gives us a point closer x^*:

$$a_{n+1} = a_n + \Delta a = a_n - \frac{f'(a_n)}{f''(a_n)}.$$

It is proved that for the chosen class of *f* one has $a_n \to x^*$ as $n \to \infty$.

Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm combines two numerical minimization algorithms: the gradient descent method and the Gauss-Newton method. Levenberg-Marquardt is a popular alternative to the Gauss-Newton method of finding the minimum of a function $F(x)$ that is a sum of squares of nonlinear functions,

$$F(x) = \frac{1}{2}\sum_{i=1}^{m}[f_i(x)]^2.$$

Let the Jacobian of $f_i(x)$ be denoted $J_i(x)$, then the Levenberg-Marquardt method searches in the direction given by the solution p to the equations

$$(J_k^T J_k + \lambda_k I)p_k = -J_k^T f_k,$$

where λ_k are nonnegative scalars and I is the identity matrix. The method has the nice property that, for some scalar Δ related to λ_k, the vector p_k is the solution of the constrained subproblem of minimizing $\frac{\|J_k p + f_k\|_2^2}{2}$ subject to $\|p\|_2 \leq \Delta$.

Results

1. Linear approximantion

```
Linear approximation:
Linear Gradient Descent: a = 0.8555408090322413 b = 0.2900314995991977
Non-linear Conjugate Gradient Descent: a = 0.8583332783885913 b = 0.28882881640992464
Newton's methods:  a = 0.8583332533346696 b = 0.28882883878089954
Levenberg-Marquardt algorithm: a = 0.8583332560264156 b = 0.28882883742213394
```

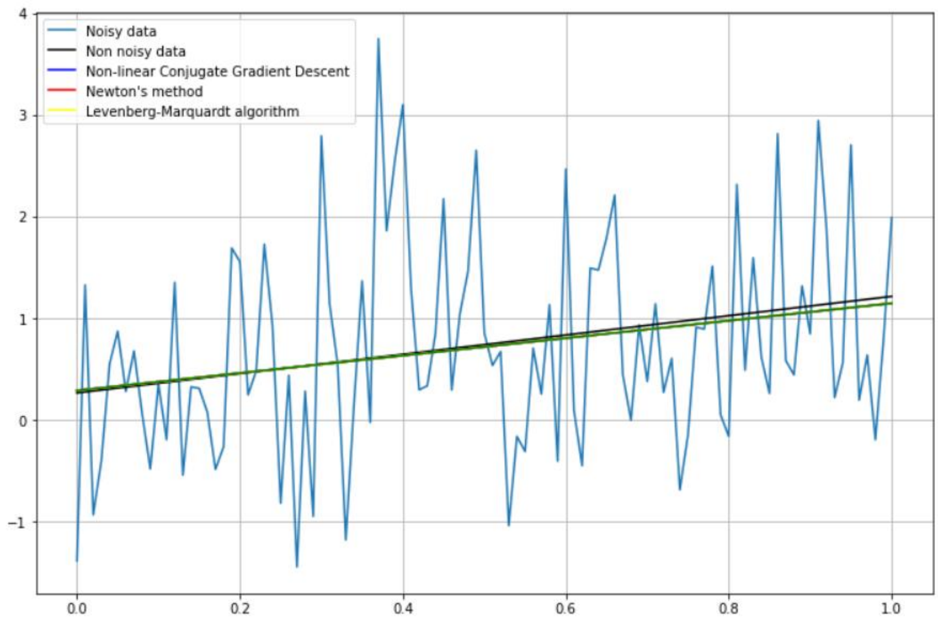

Fig. 1 - The results of the application of Gradient descent, Conjugate gradient method, Newton's method and the Levenberg-Marquardt algorithm for solving the linear approximation problem

2. Rational approximation

```
Rational approximation:
Gradient Descent: a = 0.44482525511030413 b = -0.6217851279144735
Non-linear Conjugate Gradient Descent: a = 0.472361332943737 b = -0.6030286933037283
Newton's methods:  a = 0.4723605026573971 b = -0.6030299191570843
Levenberg-Marquardt algorithm: a = 0.47237095369247195 b = -0.6030125412867636
```

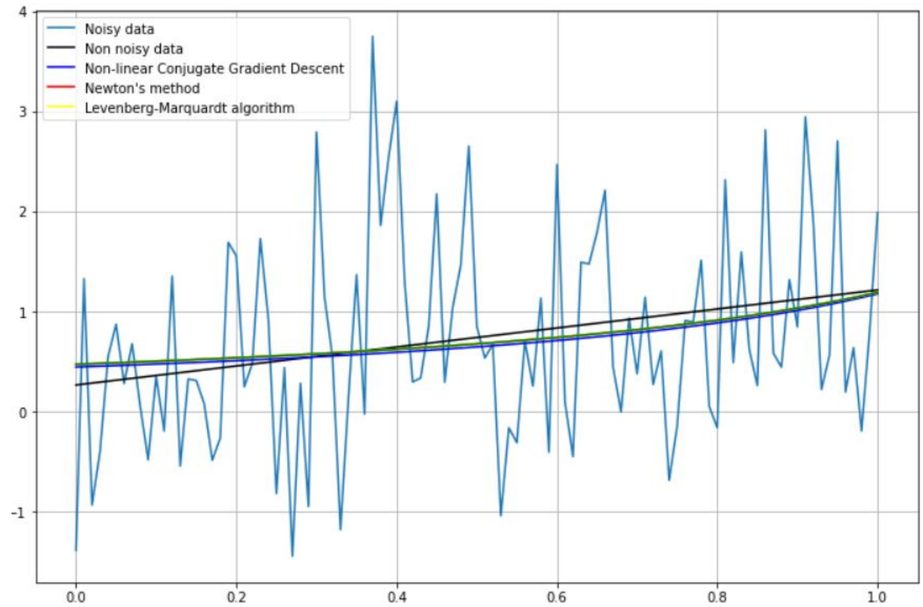

Fig. 2 - The results of the application of Gradient descent, Conjugate gradient method, Newton's method and the Levenberg-Marquardt algorithm for solving the rational approximation problem

It is well known that the optimization problem associated with linear approximation has a single solution, and therefore it is expected that these methods will give similar optimal values for a and b, regardless of the choice of initial approximations. In the case of rational approximation, significant nonlinearities arise, and therefore the choice of the initial approximation can significantly affect the result.

Conclusion

During the execution of practical task, the approximate solutions x: $f(x) \rightarrow min$ was found. To complete this task the first- (Gradient descent, Conjugate Gradient) and second-order (Newton's method, Levenberg-Marquardt algorithm) methods were used. We obtained the same results for each method. A detailed analysis can be found in the results section.

Appendix

GitHub Link: https://github.com/alex-mat-s/Algorithms/blob/main/Lab3.ipynb