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
Denis Zakharov,
Maxim Shitilov
Academic group J4132c
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

The use of the methods of Gradient Descent, Conjugate Gradient Descent, Newton's method and Levenberg-Marquardt in the tasks of unconstrained nonlinear optimization.

Brief theoretical part

There are lots of ways of solving the minimization problem, in this lab we are going to use first- and second-order methods, lets describe them below.

- First-order methods:
  - Gradient Descent — is a gradient method is an algorithm to solve problems of the form $x^{k+1} = x^{k} - \eta^{k} \nabla f(x^{k})$. Gradient descent is the preferred way to optimize neural networks and many other machine learning algorithms but is often used as a black box;
  - Conjugate Gradient Descent — usually is used for solving unconstrained optimization problems such as energy minimization. The CG method is a mathematical technique that can be useful for the optimization of both linear and non-linear systems, it is generally used as an iterative algorithm, however, it can be used as a direct method, and it will produce a numerical solution..
- Second-order methods:
  - Newton's method — it's an iterative update of model parameters like gradient descent which core formula is
    $x^{k+1} = x^{k} - H(x^{k})^{-1} \nabla f(x^{k})$;
  - Levenberg-Marquardt algorithm — is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting. The LM interpolates between the GNA and the method of gradient descent. It (LM of course) is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum.
    The LM is used in many software applications for solving generic curve-fitting problems. By using the Gauss–Newton algorithm it often converges faster than first-order methods. However, like other iterative optimization algorithms, the LM finds only a local minimum, which is not necessarily the global minimum.

Results

1) Explain differences (if any) in the results obtained.

```
Linear optimization (gradient descent):
       fun: 114.32667113989531
  hess_inv: array([[ 0.05824111, -0.02912057],
       [-0.02912057,  0.01951078]])
       jac: array([0.00000000e+00, 1.90734863e-06])
   message: 'Optimization terminated successfully.'
      nfev: 18
       nit: 4
      njev: 6
    status: 0
   success: True
         x: array([1.53054235, 0.38240716])
```

Pic. 1 — Linear optimization (gradient descent)

```
Linear optimization (conjugate gradient method):
      fun: 114.32667113990726
      jac: array([-1.90734863e-05, -5.72204590e-05])
  message: 'Optimization terminated successfully.'
     nfev: 18
      nit: 3
     njev: 6
   status: 0
  success: True
        x: array([1.53054293, 0.38240657])
```

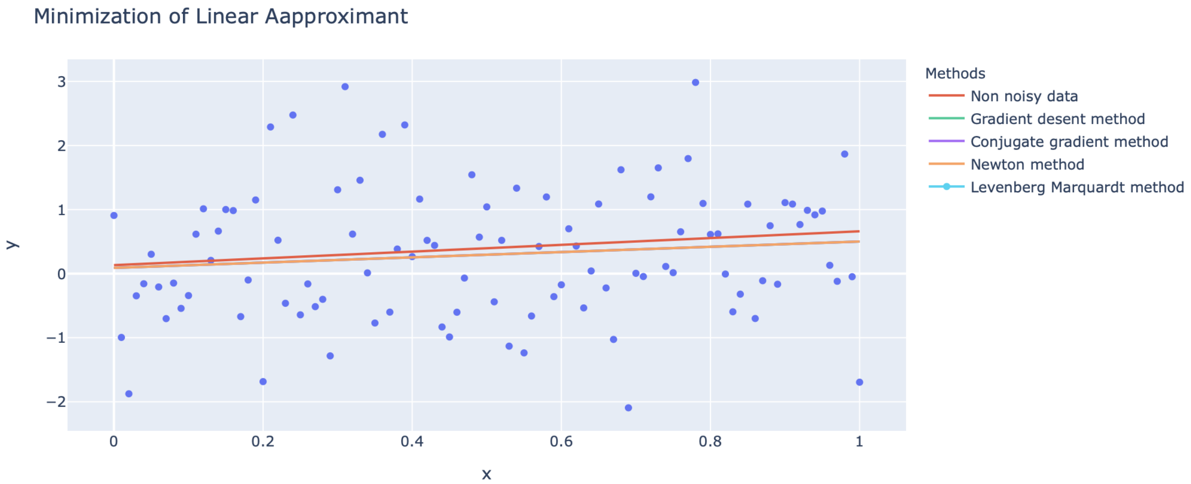
Pic. 2 — Linear optimization (conjugate gradient method)

```
Linear optimization (conjugate Newton's gradient):
      fun: 114.32667113990726
      jac: array([-1.90734863e-05, -5.72204590e-05])
  message: 'Optimization terminated successfully.'
     nfev: 18
      nit: 3
     njev: 6
   status: 0
  success: True
        x: array([1.53054293, 0.38240657])
```

Pic. 3 — Linear optimization (conjugate Newton's gradient)

```
Linear optimization (conjugate Newton's gradient):
 [1.53054237 0.38240715]
```

Pic. 4 — Linear optimization (Levenberg-Marquardt algorithm)



Pic. 5 — Linear approximation results

```
Rational optimization (gradient descent):
       fun: 119.3686959084351
  hess_inv: array([[0.00629002, 0.00440292],
       [0.00440292, 0.00442622]])
       jac: array([-0.00072384,  0.00071049])
   message: 'Optimization terminated successfully.'
      nfev: 42
       nit: 9
      njev: 14
    status: 0
   success: True
         x: array([ 0.74798084, -0.6165614 ])
```

Pic. 6 — Rational optimization (gradient descent)

```
Rational optimization (conjugate gradient method):
     fun: 119.36869590793354
     jac: array([-5.72204590e-06,  2.86102295e-06])
 message: 'Optimization terminated successfully.'
    nfev: 72
     nit: 11
    njev: 24
  status: 0
 success: True
       x: array([ 0.74798221, -0.61656139])
```

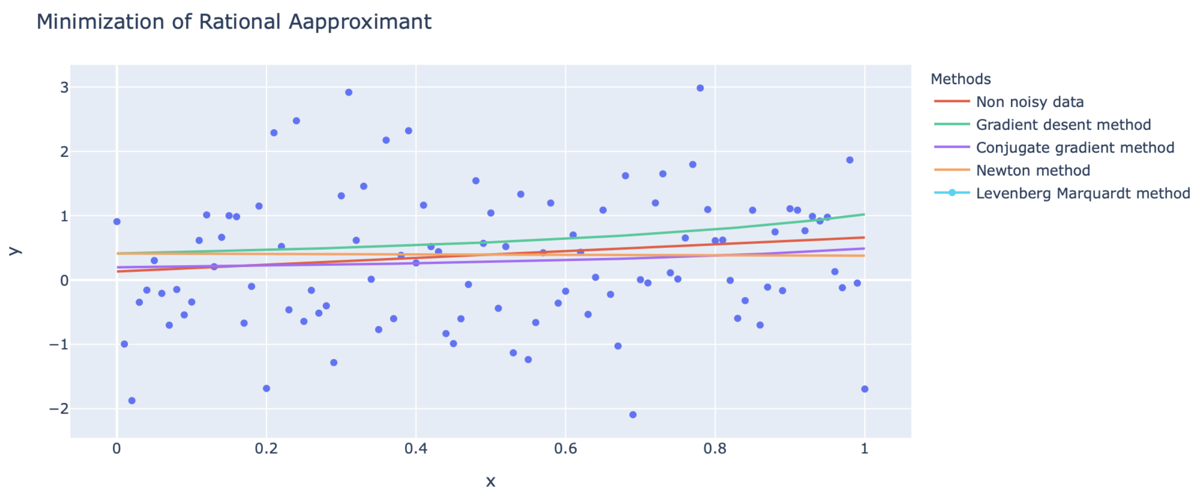
Pic. 7 — Rational optimization (conjugate gradient method)

```
Rational optimization (conjugate Newton's gradient):
      fun: 119.36869590793354
      jac: array([-5.72204590e-06,  2.86102295e-06])
  message: 'Optimization terminated successfully.'
     nfev: 72
      nit: 11
     njev: 24
   status: 0
  success: True
        x: array([ 0.74798221, -0.61656139])
```

Pic. 8 — Rational optimization (conjugate Newton's gradient)

```
Rational optimization (Levenberg-Marquardt algorithm):
 [ 0.74795446 -0.61658954]
```

Pic. 9 — Rational optimization (Levenberg-Marquardt algorithm)



Pic. 10 — Rational approximation results

Conclusions

As can be seen from the graphs and console output above, all methods have good results in minimizing the function. This happened because the value of the function and the parameters are close for both linear and rational functions. It is worth noting the difference in execution time and resource usage.

Questions for self-monitoring from Russian educational materials

1. Provide a definition of first- and second-order methods.

   First-order methods for minimization of target function $\boldsymbol{f}$ on the set $\boldsymbol{Q}$ are using values of its first derivative (gradient).

Second-order methods for minimization of target function ***f*** and values of ***f(x)*** и first derivative and second derivative (gradient, hessian) respectively.

2. Let a convex function ***f : [0, 1] → R*** with continuous derivatives ***f'*** and ***f''***. Which of the methods — coordinate descent or Newton's - the theoretical convergence rate is higher when f is optimized on the interval ***[0, 1]***?

At a local minimum (or maximum) ***x***, the derivative of the target function f vanishes: ***f'(x) = 0*** (assuming sufficient smoothness of ***f***).

Gradient descent tries to find such a minimum ***x*** by using information from the first derivative of ***f***: It simply follows the steepest descent from the current point. This is like rolling a ball down the graph of f until it comes to rest (while neglecting inertia).

Newton's method tries to find a point ***x*** satisfying ***f'(x) = 0*** by approximating ***f'*** with a linear function ***g*** and then solving for the root of that function explicitely (this is called Newton's root-finding method). The root of ***g*** is not necessarily the root of ***f'***, but it is under many circumstances a good guess *(the Wikipedia article on Newton's method for root finding has more information on convergence criteria)*. While approximating ***f'***, Newton's method makes use of ***f''*** (the curvature of ***f***). This means it has higher requirements on the smoothness of ***f***, but it also means that (by using more information) it often converges faster.

Appendix

DataLore : site. – URL: https://datalore.jetbrains.com/notebook/RemqSkuJwmr1PM4Gc3cBqB/74hSgqDSemOyOL9kJ5qC49 (circulation date: 19.09.2022)