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
Daan Rodríguez

Academic group
J4132c
Accepted by
Dr Petr Chunaev

St. Petersburg
2023

Goal

Use first- and second-order methods (gradient descent, nonlinear conjugate, gradient descent, Newton's method, and Levenberg-Marquardt algorithm) to find the optimal a and b of the numerical minimization function.

Formulation of the problem

Use the gradient descent, nonlinear conjugate, gradient descent, Newton's method, and Levenberg-Marquardt algorithm to solve the minimization problem. by means of least squares through the numerical minimization (with precision $\mathcal{E}$ = 0.001) of the following function:$D(a,b) = \sum_{k=0}^{100}(F(x_k,a,b) - y_k)^2$. Visualize the data and the approximants obtained in a plot separately for each type of approximant.

Brief theoretical part

1. Algorithms introduction:

   1.1 Gradient descent: Finding the parameters and their gradients, and using the gradients to update the parameters, forms a loop. Repeat this cycle until convergence

   1.2 Conjugate, gradient descent:

   Calculate the steepest direction at the current position a_n, which is $-\nabla f(a_n)$.

   Calculate the step size $_n$ based on certain formulas. Here, the choice of the formula (which could be Fletcher-Reeves or Polak-Ribiere, among others) is made to ensure uniform convergence for convex functions with Lipschitz gradients.

   Update the Conjugate Direction: Update the conjugate direction sn, where sn represents the current direction, using the following formula

   1.3 Newton's method:

   A concave function is formed by Taylor expansion (to the quadratic term) of a certain point, and the position of the next point is determined through the minimum value b/2a of the concave function, and the iterative loop.

   1.4 LMA :

   The Levenberg-Marquardt algorithm (LMA) is an iterative optimization method used for solving nonlinear least squares problems. It aims to minimize the sum of squared differences between observed data points and the corresponding predictions from a nonlinear function. LMA combines elements of both the Gauss-Newton algorithm and Gradient Descent method through the introduction of a damping factor, denoted as x. This damping factor allows LMA to smoothly transition between the two methods. When x is small, the algorithm behaves more like Gauss-Newton, emphasizing the quadratic approximation and second-order information. As increases, it becomes more like Gradient Descent, focusing on first-order information and permitting larger steps. The choice of the damping factor x is crucial and is dynamically adjusted at each iteration to ensure convergence. This flexibility makes LMA a powerful tool for solving nonlinear least squares problems.

2. problem analysis:

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

α	Random variable between 0 to 1

β	Random variable between 0 to 1
k	Range from 1 to 100
δ k x	Radom variable with standard normal distrubution 1/k
y	y=x* α + β +noise(variable to be fitted)
Minimumize function(error function)	$(a,b) = \sum_{k=0}^{100} (F(x_k, a, b) - y_k)^2$
The variable of Minimumize function	a and b from linear function and rational function

Linear function:$D(a,b) = \sum_{k=0}^{100}((a/k + b) - (1/\mathrm{k} * \alpha + \beta + \mathrm{noise}))^2$

Rational function:$D(a,b) = \sum_{k=0}^{100}((\frac{a}{1+b/k}) - 1/\mathrm{k} * \alpha + \beta + \mathrm{noise})^2$

We treat it as a three-dimensional problem and plot

Hypothesis: α =0.5 $\beta = 0.5$ $\mathrm{noise} = 0.5$

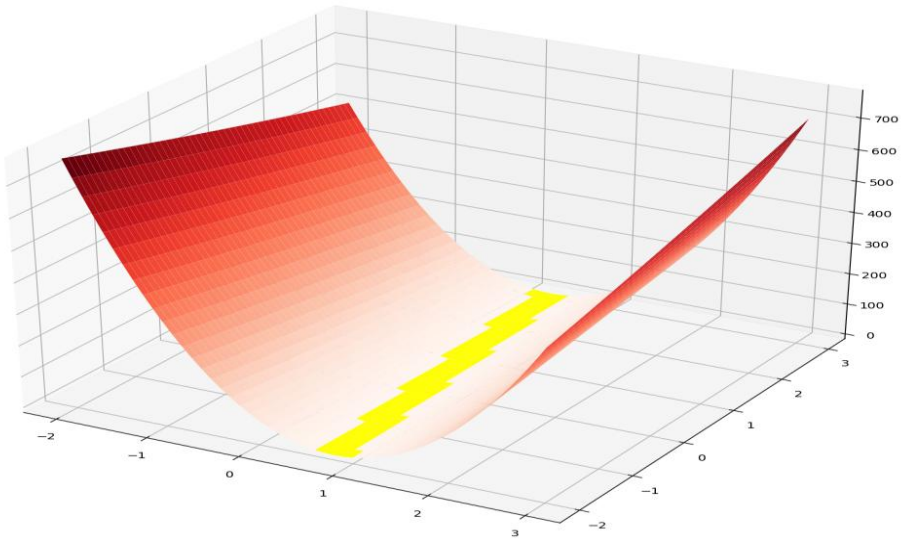


Figure 1 - Draw linear_image

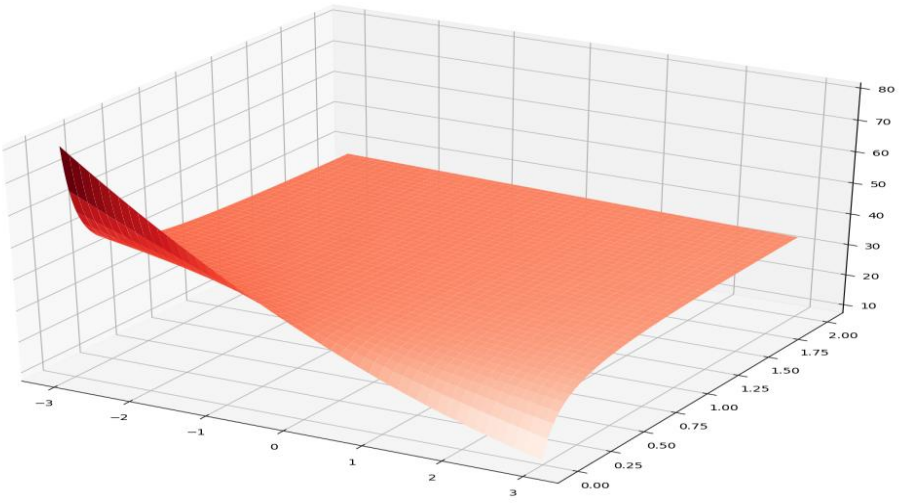


Figure 2 - Draw rational_image of b ∈（0.2）

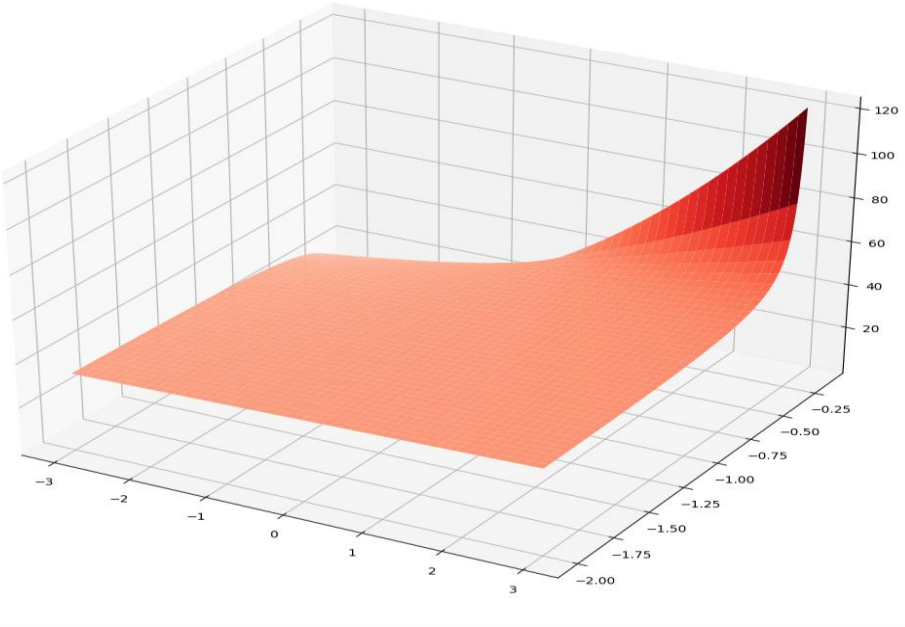


Figure 3 - Draw rational_image of b ∈（-2.-0.15）

As shown in the figure, the above problem can be regarded as the problem of finding the minimum value of D(a,b) in the three-dimensional space as α, β, and noise change.

Result :

1. Gradient descent

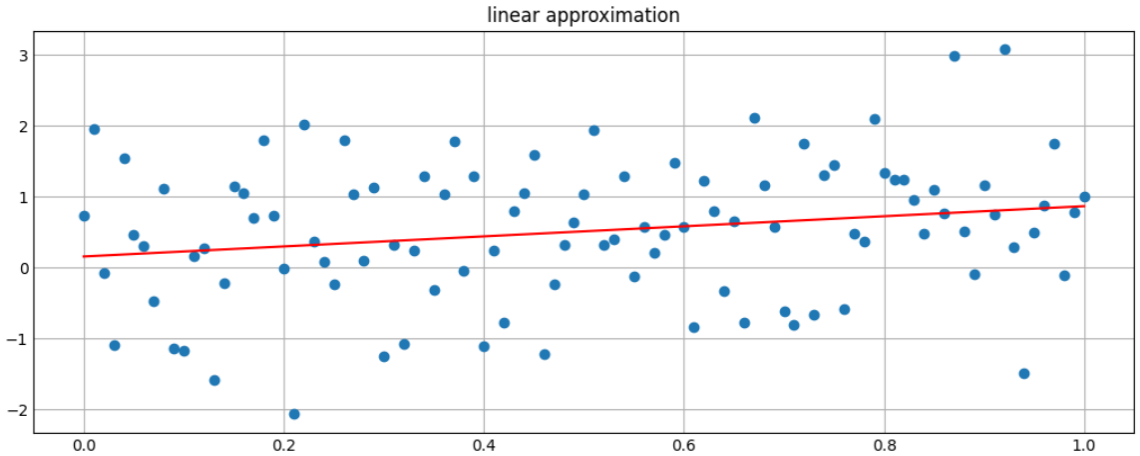


Figure 4 – linear approximation of gradient descent

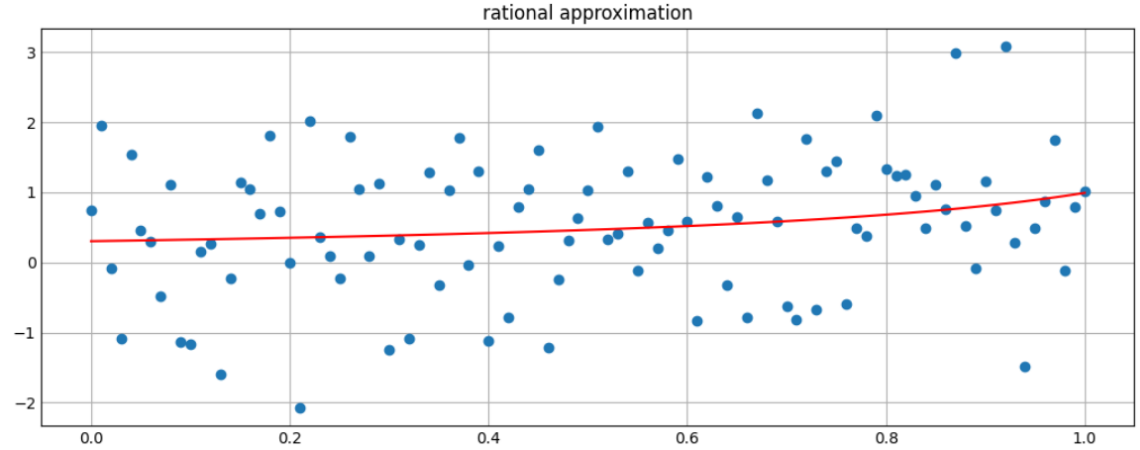


Figure 5 - rational approximation of gradient descent

2. Conjugate Gradient descent method of optimization

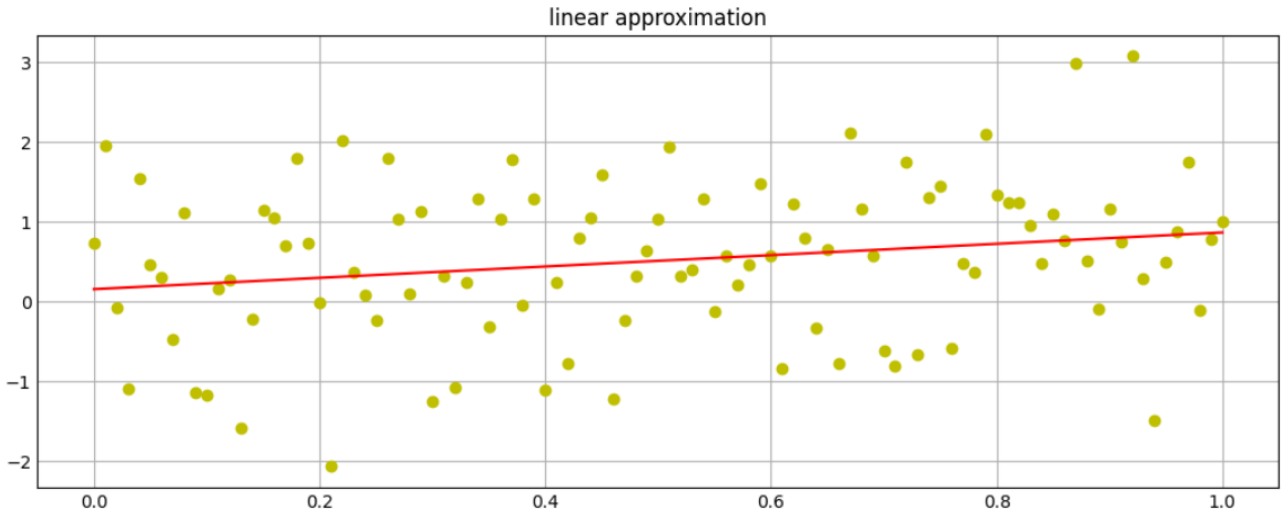


Figure 6 – linear approximation of Conjugate Gradient descent

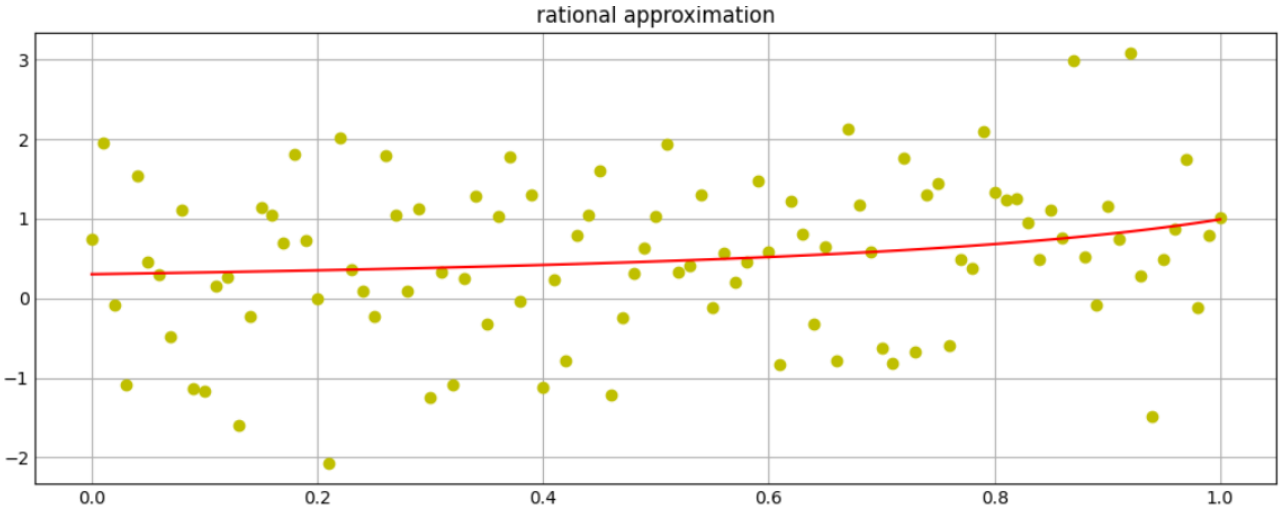


Figure 7 - linear approximation of Conjugate Gradient descent

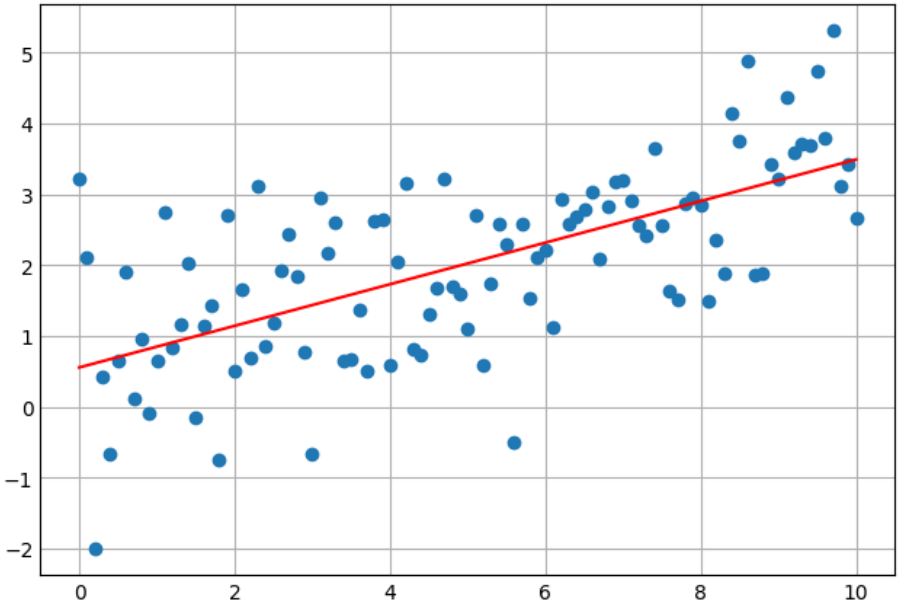
```
Start training for <function linear at 0x000001F6EDEC0540>
Optimization terminated successfully.
         Current function value: 93.795082
         Iterations: 2
         Function evaluations: 15
         Gradient evaluations: 5
Founded values: (0.7114828691773584, 0.15345925194663798), real values: (0.613310219569985, 0.20951634500409944)

Start training for <function rational at 0x000001F6EDE73BA0>
Optimization terminated successfully.
         Current function value: 94.001457
         Iterations: 11
         Function evaluations: 91
         Gradient evaluations: 30
Founded values: (0.30118294072019436, -0.6954890887334927), real values: (0.613310219569985, 0.20951634500409944)
```

Figure 8 – Conjugate Gradient descent approximation result

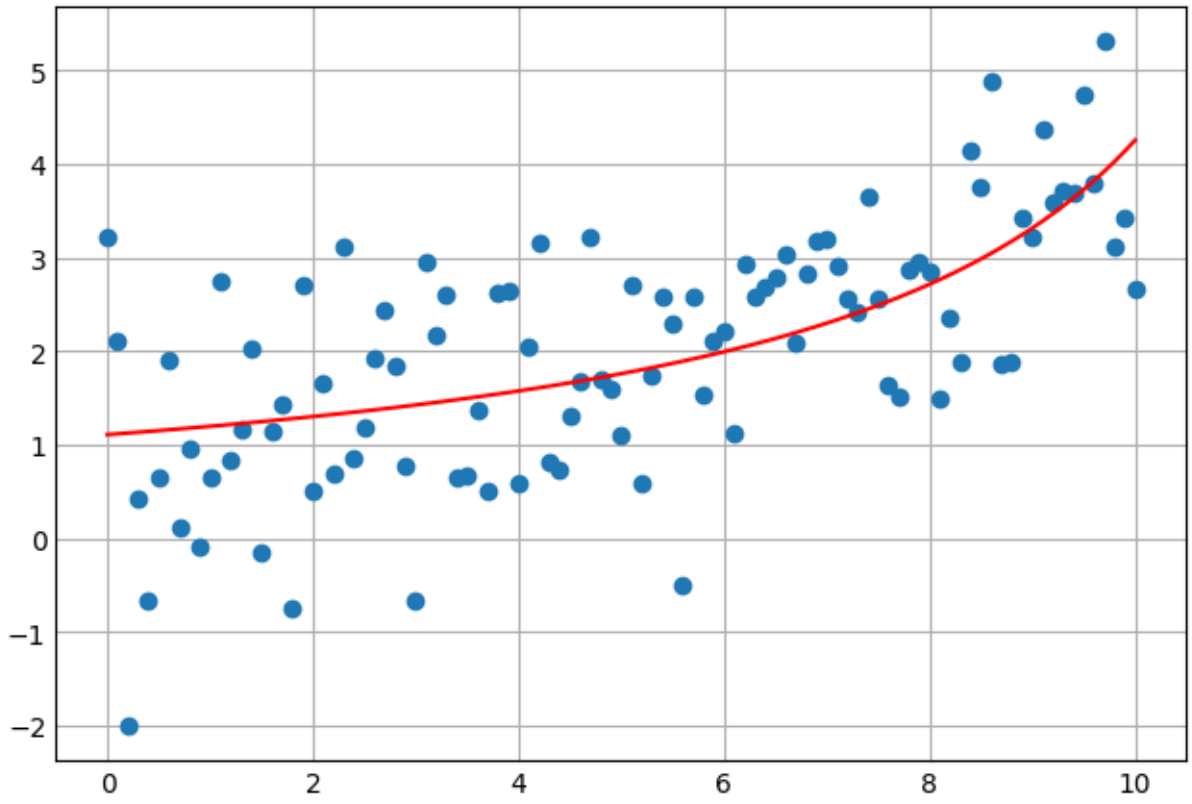
The code uses the Conjugate Gradient Descent optimization method to find the best-fitting parameters and for both linear and rational approximations

Overall, the code is designed to fit linear and rational approximations to noisy data and visualize the results. The analysis can focus on evaluating the quality of the fit and the effectiveness of the optimization method chosen.

3. Newton's method



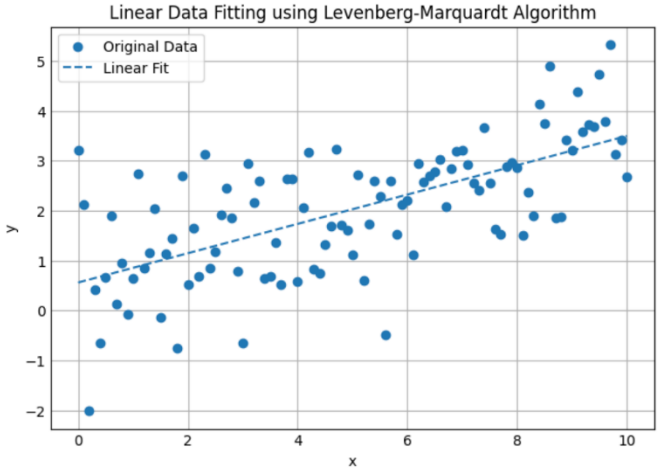
```
         Current function value: 99.557299
         Iterations: 2
         Function evaluations: 32
         Gradient evaluations: 20
         Hessian evaluations: 3
[0.29353059 0.55852663]
```

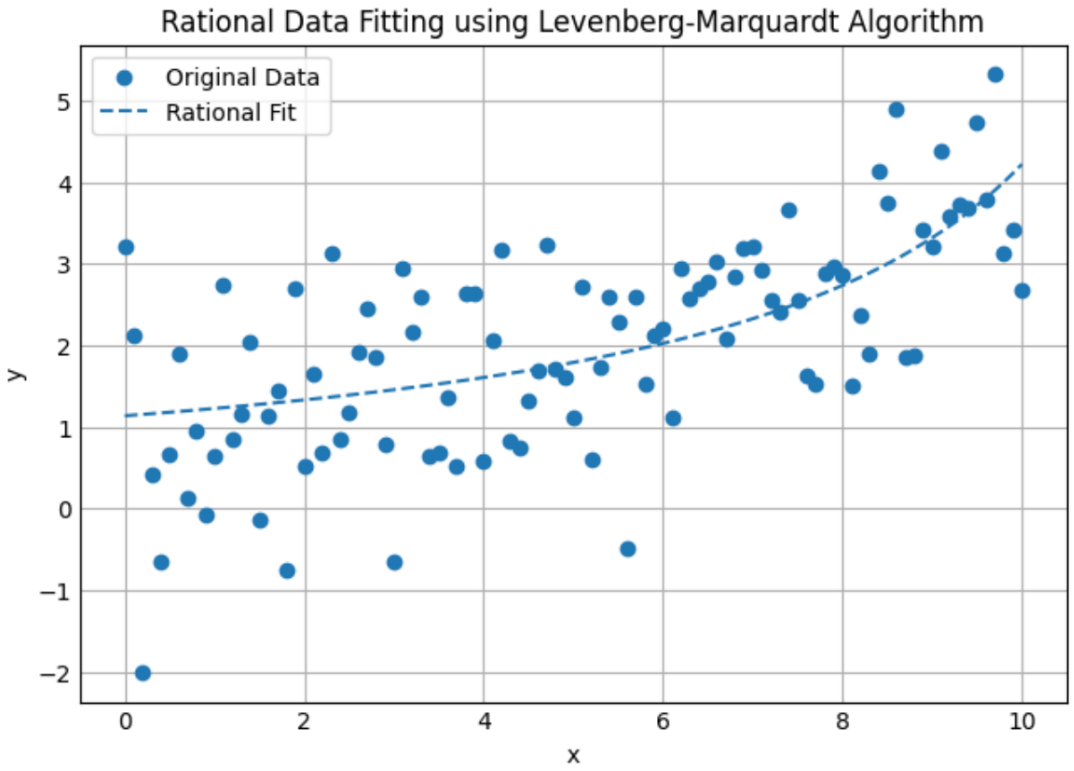



```
Optimization terminated successfully.
         Current function value: 99.835026
         Iterations: 3
         Function evaluations: 14
         Gradient evaluations: 14
         Hessian evaluations: 3
[ 1.11108926 -0.07389253]
```

Because Newton's method needs to update the hessian matrix and gradient for each iteration, the speed of the manual algorithm will be greatly reduced, so the optimize.minimize function is used

4. Levenberg-Marquardt algorithm (LMA)






```
Optimization terminated successfully:(linear) True
Current function value:(linear) 49.778649189287
Function evaluations:(linear) 6
[0.29354682 0.55845141]
Optimization terminated successfully:(rational) True
Current function value:(rational) 49.884659062109606
Function evaluations:(rational) 39
[ 1.13520455 -0.07308604]
```

LMA also suffers from the problem of slow running due to repeated updates of derivatives. LMA uses the scipy.optimize.least_square function.

Why LMA is a second-order method Conclusions: Because LMA requires a set of residual vectors to update the JACOBIAN matrix when updating parameters.

Conclusions

In this task, we embarked on a comprehensive exploration of approximating noisy data using two distinct types of functions: linear and rational. We employed four numerical optimization methods—Gradient Descent, Conjugate Gradient Descent, Newton's method, and the Levenberg-Marquardt algorithm—to minimize a loss function and derive optimal parameter values for each type of function approximation. Our analysis encompassed both the efficiency and accuracy of these methods, and we compared our findings with those of a previous task involving a similar dataset.

Data Generation: We initiated the task by generating random values α and β from a uniform distribution to introduce variability into our noisy dataset. These random coefficients were combined with a standard normal distribution variable $\delta+$ to create the noisy data points $\{x+, y+\}$.

Model Approximation: Two function approximations were chosen for our study. A linear approximant, represented by $F(x, a, b) = ax + b$, and a rational approximant, $F(x, a, b) = a / (bx + 4)$, were employed to approximate the noisy data. The rational approximant was used to capture more complex relationships.

Numerical Optimization: We implemented four numerical optimization methods to minimize the loss function $D(a, b)$. Each method was executed with precision $\varepsilon = 0.001$ and appropriately initialized parameters. The methods included Gradient Descent, Conjugate Gradient Descent, Newton's method, and the Levenberg-Marquardt algorithm, each offering a unique approach to optimization.

Visualization: The results of our analysis were visually presented through plots. Separate plots were created for each type of function approximant, displaying both the noisy data points $\{x+, y+\}$ and the approximations derived using the four optimization methods. This visual representation allowed for a clear comparison of the results.

Analysis and Comparison: We conducted a comprehensive analysis of the results obtained from each optimization method. Key aspects such as the number of iterations, precision achieved, and the number of function evaluations were scrutinized. This analysis provided insights into the efficiency and accuracy of each method and revealed variations in performance across different function approximants.