FUNDAMENTALS OF COMPUTATIONAL MATHEMATICS  
Prof. Manuel Aprile

Prof. Francesco Marchetti  
University of Padova

**Project: Yield Curve Model Calibration**

**Ibrahim Uali**

**Riccardo Caruso**

**Enrico Berto**

**Elisa De Colle**

February 16, 2024

Contents

[1 The Nelson-Siegel model 3](#_Toc158728867)

[1.1 Estimation difficulties: The problems Of optimization and multicollinearity 3](#_Toc158728868)

[2 Extended Nelson-Siegel function or Svensson model 5](#_Toc158728869)

[3 The project 5](#_Toc158728870)

[3.1 Data collection 5](#_Toc158728871)

[3.2 Creation of the dataframe 6](#_Toc158728872)

[3.3 Parameter estimation 9](#_Toc158728873)

[4 Gradient-Descent method 9](#_Toc158728874)

[4.1 GRADIENT DESCENT METHOD - US 11](#_Toc158728875)

[5 Newton method 14](#_Toc158728876)

[5.1 Newton method - US 15](#_Toc158728877)

[6 Comparison of Gradient Descent and Newton methods 17](#_Toc158728878)

[Other approaches implemented in the optimization of the model 17](#_Toc158728879)

[ORDINARY LEAST SQUARES – RIDGE REGRESSION APPROACH: 18](#_Toc158728880)

[7 Quasi-Newton Methods and BFGS algorithm 19](#_Toc158728881)

[7.1 BFGS optimization 20](#_Toc158728882)

[7.2 Code implementation of BFGS algorithm 20](#_Toc158728883)

[8 The Levenberg-Marquardt (LM) algorithm 23](#_Toc158728884)

[8.1 Code implementation of LM algorithm and comparison with BFGS 25](#_Toc158728885)

[Bibliography: 27](#_Toc158728886)

# 1 The Nelson-Siegel model

The Nielson Siegel model is a model used for the prediction of yield curves, developed by Charles Nelson and Andrew Siegel in 1987 at the University of Washington.

This model aims to predict the movements of the yield curve basing itself on various terms to maturity, such as: flat, hump and S shapes. The formulated model is the following:

With equal to yield at time t, while , and are parameters associated to the long-run yield level, slope of the yield and curvature of the yield respectively.

The last parameter, , is the one influencing the rate at which yields converge to the long-term level.

This model is widely used by economists because it has several characteristics that tend to make it a versatile and efficient model. First of all, it respects the restrictions imposed by the economic and financial theory, moreover its approximation avoids in-sample overfitting, such as that it leads to an increase in forecasting capacity.

In addition, versatility refers to the fact that it can take any yield curve form that has been empirically observed in the market.

For these reasons the model is extensively used by central banks and monetary policy makers, while fixed-income portfolio managers use it in order to immunize their portfolios.

Academic literature abounds with instances where researchers have derived significant conclusions by employing the Nielson-Siegel model: Diebold and Li (2006) benchmarked the Nielson-Siegel against other models in the term structure forecasts, the finding was that, especially for longer forecast horizons, it performs very well. Coroneo, Nyhlom and Vidava-Koleva (2008) test to which degree the Nielson-Siegel model approximate an arbitrage-free model.

Although there exist also other models used to estimate the term structure of interest rates, most of them have been proven to have undesirable economic properties, moreover they have been seen to be black-box models, so they are not easy to be interpreted and understood by humans.

## 1.1 Estimation difficulties: The problems Of optimization and multicollinearity

The estimation problems addressed to this model refer to his lack of linearity and to the presence of multicollinearity between variables.

Since the parameter that causes non – linearity in the model is λ, or the so-called ‘shape parameter’, many estimation approaches are based on the condition that λ is fixed, in order to linearize the model and perform the estimation by using Ordinary Least Squares.

We are going to investigate this approach more in the detail later, for now we just state that the main problem of this model is that it can become heavily collinear depending on the estimate or fixed shape parameter, λ.

The issue of multicollinearity is that, when dealing with it, the estimation of parameters performed by using least squares can lead to unstable and highly variable coefficient estimates.

Since the Nielson-Siegel model is highly non-linear, we need to estimate its parameters by using non - linear optimization techniques, like for example the Gradient Descent, always considering that this model tends to be very sensitive to the starting values used in the optimization. This issue refers to what is called the “Optimization problem”.

In order to investigate further these problems, we consider a more general representation of the model:

A number of mathematical equations

Description automatically generated

Here we divided the model into three main components, , and that represent respectively the level, the slope, and the curvature of the spot curve.

Deeping further in the behavior of the model, we can see that when the time to maturity grows to infinity, the slope and the curvature components goes to zero and the long-term spot rate converges to a constant level of interest, that is .

is the spread, it measures the slope of the term structure, a negative value represents an upward.

The degree of curvature is determined by , which is the rate at which the slope and curvature components decay to zero.

Λ here determines the location of the maximum/minimum value of the curvature component, it determines both the shape of the curvature component and the hump/through of the term structure.

When dealing with Ordinary Least Squares estimation, the multicollinearity problem refers to the high correlation between the slope (r1) and the curvature (r2) components, this correlation heavily depends on the choice of λ, that is usually set to be 1.37 or 3, other than that, it depends also on the choice of the time to maturity vector.

To summarize, Gilli et al. (2010) identifies three major problems with estimating this model by non-linear methods or OLS:

* The optimization problem is non-convex, showing multiple local optima.
* The model is badly conditioned for a certain range of parameters, the estimates are unstable before small price movements.
* The value of the decay factor (λ) directly affects the correlation between the loadings of the slope and curvature factors, the problem here could be in the forecasting use of the model.

# 2 Extended Nelson-Siegel function or Svensson model

In order to extend the Nielson – Siegel model and make it possible to capture economics change in case of market crisis, Svensson added an extra term to the original model, allowing for a second hump in the yield curve.

This fifth parameter is able to capture a greater variety of yield curve shapes, that can fit in a good way also yields in time of monetary crisis.

The model is computed as follows:

The parameters present in the original model have the same interpretation that they had in the first one, while and determine position and magnitude of a possible second hump in the yield curve.

# 3 The project

## 3.1 Data collection

We started by collecting the historical data of various European bonds, we selected four times to maturity: one, two, three, five and ten years, in order to cover the whole maturity range and distribute the data along the whole period.

We chose not to select bonds with very short maturities, because it is possible that they don’t show sufficient liquidity.

The data selected are both from European and non-European countries, to be more precise we chose bonds from the following four countries: United States, Portugal, Germany and South Korea.

The selection of the countries was based on the need to incorporate bonds from diverse market economies. This approach allows for an examination of the Nelson-Siegel model's efficacy across different economic conditions and market dynamics.

United states bonds are believed to be one of the safest investments, moreover they are highly liquid and considered close to risk-free, since they are backed by the U.S. government’s; given their stability and reliability they tend to be very popular in the financial market.

German bonds have kind of the same characteristics of the US ones in terms of stability and reliability in the Eurozone, they have a high credit rating, AAA, that states for a very low risk of default.

Given their reliability they are often used as a benchmark in the bond market, in the sense that their yields are often compared to those of other European bonds, moreover they serve as a reference point for assessing credit risk in the Eurozone, this characteristic make these bonds highly liquid.

Portugal bonds, on the other hands are less liquid than the German one, even if they are relatively liquid, and they are considered to be less reliable due to the economic conditions of the country.

For what concern the choice of South Korean bonds, we selected them because they tend to show very interesting yield curves which present an unusual behavior respect to the ones typical of developed economies, this may be due to the fact that the Korean currency is weak with respect to the US dollar.

The time frame we are referring to goes from the first of January 2022 to the first of December, 2023.

## 3.2 Creation of the dataframe

We started with the creation of a global Data frame in order to be able to work directly on it, at first, we imported on python the data downloaded from the website ‘investing.com’. Since the data we found where divided by each maturity, at the beginning, we had five data frames for each country.

The data we obtained included additional information beyond what is required for implementing the Nelson-Siegel model. This supplementary information encompasses explicit market values of the bonds, such as opening, high, and low prices, as well as percentage changes. In order to get rid of them we applied the function *‘clear\_df’,* which purpose is to drop useless columns and set the date of the month as index of the cleared dataframes. Please find this function in the script *‘fun.py’*.

After filtering out unnecessary columns, we are left with a separate data frame for each maturity level and country. Each data frame contains a column representing bond prices indexed by month.

In order to perform the Nelson-Siegel model we need a data frame for each date, which reports the yield of the bonds with respect to the maturity. In order to obtain this, we developed the following function, *‘join\_df\_date’:*

A group of text on a white background

Description automatically generated

Using this command we successfully merged our data frames with a column for price and the other one for maturity for each month, resulting in a unified list of data frames for each month. We created this type of data frame for each country with the same function.

For example, the data frame obtained for the German bonds referring to the date 1st of August 2023 is the following:

A screenshot of a graph

Description automatically generated

This is just a line of the data frame created for German bonds, which contains all the month together.

Now, to achieve our objective of creating a comprehensive global dataframe encompassing all countries, the remaining step involves merging the four individual dataframes we generated for each country into a single unified dataframe.

In order to do that we used the following command:

A close-up of a text

Description automatically generated

Next, we aim to enhance these data frames by incorporating additional columns containing predicted yields derived from the Nelson-Siegel and the Nelson-Siegel Svensson models. To achieve this, utilized the function previously described theoretically, implemented in Python., whit that function we managed to compute the predicted R(t).

A computer screen shot of a code

Description automatically generated

The function *compute\_R* is then looped across the data frame of each country contained in the global one, *‘all\_df\_joint,* to obtain the predictions of the yield curve for each date for every country according to the maturities at disposal.

In the function mentioned above we also added the possibility to predict the curve by means of the Nelson – Siegel – Svensson model as you can see in the ‘else’ statement, the advantages of this extended model are going to be exploited further in this paper.

The predictions that we are going to get with this curve won’t be any good but will change once we have optimized the function.

That is because this model relies on some parameters, that are:

* The time, for which the expected return has to be computed.
* The parameters insight the model, which are β0, β1, β2, τ. (β3 and τ2 for the Nelson-Siegel-Svensson).

The setting of the starting parameters is quietly important, especially for the optimization methods that we are going to use later to optimize our model. We underline how in particular the Gradient Descent method tends to be very sensitive to the initial parameters; inaccurate initial parameter values can lead the optimization function to diverge towards infinity.

We based our selection of appropriate values on insights from relevant academic papers, moreover our choice has been made after some trials of the behavior of the optimization methods with different starting values. We noticed that for example, by setting all the parameters to 0.01 the Gradient Descent method was diverging.

To kick off our exploration of optimal parameters, we adopted a randomized approach, selecting values from a range recommended by various academics, notably Gilli et al. (2010). This initial phase involved systematically testing different parameter combinations to identify the most promising configurations.

The ranges suggested by them are:

* 0 β0 15
* -15 β1 30
* -30 β2 30
* -30 β3 30
* 0 τ 2.5
* τ2 5.5

We performed several trials with different combinations of values. At the end we found different starting values for each countries, given the different economic conditions they lie in.

Focusing example of United States, the evidence that emerged from our empirical work, that we were able to find mostly by looking at the plots of the Nelson Siegel compared with the ones of the Historical data, was that values higher than 1 for the betas did not perform well. Consequently, we proceeded with subsequent trials using lower beta values based on this observation.

Moreover, the plots suggested to us that the value of β3 should be negative.

After more or less (tot) number of trials, we decided to select the following values, according to the higher performance they were showing for both the Gradient Descent and the Newton methods:

* β0 = 0.03
* β1 = 0.015
* β2 = 0.010
* β3 = -0.0020
* τ = 2.32
* τ2 = 12.35

Added the Nelson Siegel model values to our data frames we had all the data needed in order to compute the sum of squared difference between observed and model predicted yields and optimize it according to the parameters.

A similar approach was adopted to incorporate Nelson-Siegel Svensson values.

## 3.3 Parameter estimation

The sum of squared difference between observed and model predicted yields is computed as follows:

This is the function to compute the sum of squared residuals for the Nelson-Siegel model, once again in the python code we also presented the version for the Nelson-Siegel-Svensson model.

A computer code with black text

Description automatically generated

Now we can proceed with the minimization of ƒ, adjusting the parameters via de Gradient Descent and the Newton methods.

In order to use these two methods, we imported them at the beginning, adjusting a bit at our convenience the ones presented in class, during Laboratory 02; we imported *‘Lab02\_solutions’* as lb in order for our code to be more compact.

# 4 Gradient-Descent method

We initiate the process with the Gradient Descent method, which operates on a continuous differentiable function, denoted as ƒ, aiming to minimize it.

The procedure relies on a descent direction, d, and a step size, α.

The descent direction is the gradient of ƒ calculated in the point before the one we are trying to update, while α is found by performing approximate line search, which ensures that we give to our descent direction a stable minimization following the ‘Armijo-Goldstein Condition’.

At the beginning, we set the approximate line search function to represent a condition that is not approximate, in python code: *‘apx\_LS = False’.*

The parameters we choose were the following ones:

* β0 = 0.01
* β1 = 0.05
* β2 = 0.2
* β3 = 0.03
* τ, τ2 = 1

By using these conditions, we noticed that the Gradient Descent method was diverging, the following are some of the plots that we obtained, we present the example of the 1st of January 2022, for each country.

A graph with a line graph and a line graph

Description automatically generated with medium confidenceA graph with a line graph and a line graph

Description automatically generated with medium confidence

The same situation occurs every month.

After several tests we ended up with good starting values, different for each country given the different economic conditions.

For example, for US we went for the parameters presented at the beginning of this paper and . We found that this value was a good meeting point between a good computational time and a good approximation. For example, was taking much less iterations but was overstepping a lot, while for an alpha smaller than 0.65 the time taken was too much.

Moreover, we set *‘apx\_LS = True’.*

Given that, the updated x is going to be the previous one plus alpha multiplied by the descent direction.

The procedure is going to stop either when the number of iterations reaches the maximum decided at the beginning or when the value of the last iteration is below a certain threshold decided at the beginning.

We established the maximum number of iterations to be 100.

The functions we are referring to are named in python as *‘gradient\_descent’* and *‘apx\_line\_search’* and they are stored in the script *‘Lab.py’.*

Before performing the minimization, we initiated four lists, with the aim of storing the values found by the minimization. The lists are one for the parameters and one for the value of *f*, both of them were created for the Nielson Siegel and the Nielson Siegel Svensson model.

We ran a for loop to perform the gradient descent methods in each data frame for every country we are referring to. Once the optimization has been done, we collected the values of the parameters stored in the lists in a data frame that we saved in Excell. For the complete code please refer to the script *‘Data.py’.*

We did the same also for the Nielson Siegel Svensson model, to compare which of the two models is more precise in fitting the historical data.

By running the code, we obtained that for most of the data provided the Gradient Descent method performed one-hundred iterations, which is the maximum number of iterations we decided at the beginning, clearly this states that the method is converging slowly.

Anyway, even when the methods performed the maximum number of iterations, the graph results to be sufficiently precise.

Also moving to the plots one can clearly notice that, when the gradient descent performed less than one hundred iterations, the Nielson Siegel curve, and the Nielson Siegel Svensson curve as well, are fitting the historical yield data in a very precise way.

In order to plot the curves, we extracted the last parameters’ values from the data frames of parameters stored by the gradient descent method, this is because the last value should be the optimal one. Once we have obtained them, they can be used to compute the Nelson Siegel and the Nelson Siegel Svensson curve.

## 4.1 GRADIENT DESCENT METHOD - US

We are going to present some examples of different situations by looking at some of the plots that we obtained. All the plots can be found in the zip file. We start by showing the plots of the Nelson Siegel and Nelson Siegel Svensson curve for US bond on the 1st of January 2022, for which the Gradient Descent performed 100 iterations for both the Nielson Siegel and Nielson Siegel Svensson.

A graph with a line and a dotted line

Description automatically generatedA graph with a line and a dotted line

Description automatically generated

It is evident that, if we take into consideration data that performed the maximum number of iterations, the plots are still reflecting the historical data. However, if we take into considerations plots of dates for which the gradient descent method performed less than 100 iterations, the curves are slightly more precise.

This is the case of US bonds on the 1st of June 2022:

A graph with a line and a dotted line

Description automatically generated A graph with a line and a dotted line

Description automatically generated

Nelson Siegel Nelson Siegel Svennson

We underline that in this case the Gradient descent method performed 94 iterations for the Nelson Siegel and 100 for the Nielson Siegel Svensson.

Furthermore, we noticed that in the case of bonds with high volatility the Nelson Siegel and Nelson Siegel Svensson model optimized by the gradient descent method are not able to perfectly predict the yield curve, even though the extended model tends to slightly outperform the original one.

Here is the example of US bonds on the 1st of March 2022:

A graph with a line and a line

Description automatically generated with medium confidenceA graph with a line and a line

Description automatically generated with medium confidence

In the location of the hump neither the original Nielson Siegel nor the extended version are able to fit the historical data. Here the model is converging very slow, if we set the possibility to run more iterations the Gradient Descent method would probably find the optimal parameters and the model would fit the historical data in a better way, but this procedure would require a huge computational time.

One issue that we noticed about the Newton method was in predicting Reversed Yield curves.

First of all, let’s investigate what Reversed Yield curves are. When the yield curve inverts, it means that short-term interest rates are higher than long-term rates, the yield decreases the further away the maturity date is.

Inverted yields curve used to be an unusual occurrence, but in times of inflation this phenomenon tends to happen more frequently. Focusing on the US, in this time of raising interest rates, low maturity now tends to have higher yields.

That is why the inverted curve is a reliable indicator of a recession. It suggests that investors have little confidence in the near-term economic conditions and believe that interest rates will fall in the future.

Let’s see the example of October 2023:

A graph with blue and orange lines

Description automatically generatedA graph with a line and a line

Description automatically generated with medium confidence

We will see that in the cases mentioned above the Newton method is more precise and we will explore the reasons behind this.

# 5 Newton method

The second method we implemented is the Newton method.

This procedure for functions optimization is similar to the Gradient Descent one, except that in this case the updating step is computed using second-order Taylor expansion. The ‘d’ we were referring to in the Gradient Descent method in this case is the opposite of the gradient at point xᵢ, while alpha is the inverse of the Hessian at point xᵢ.

In order for this method to be effective, it's crucial to verify that the Hessian matrix is Positive Definite. If it's not, the method cannot be applied as the Hessian would be non-invertible. To mitigate this issue, a common practice is to incorporate a regularization term, typically denoted as , into the matrix. This regularization term ensures the matrix becomes Positive Definite, allowing for the successful application of the method. The regularization term mentioned above has been called in the code ‘Damping factor’ and has been set to be 0.5.

Once again, to use this method, we implemented the code learned during laboratory hours, that we imported at the beginning of our script.

The function *‘newton\_method’* is looped across all the data frames we created at the beginning for each country, in explicit we have twenty-three different data frames for every country we selected, one for every month of our time frame.

At each iteration the Nelson Siegel (and Nelson Siegel Svensson) parameters are optimized according to the Newton’s method, the number of iterations performed to achieve the optimal parameters are then returned by the *‘newton\_method’* function itself.

We stored the values of every iteration in an Excel file, if it is of interest to you, please find it in the zip file.

Once we have done that, in order to plot the curves predicted by the Nelson Siegel and Nelson Siegel Svensson models with respect to the observed yield curve, we need to select the parameters found by the last iteration performed (that should be the optimal ones) and compute the expected returns by means of the original and extended version of the Nelson Siegel model with those parameters.

The following piece of code shows this procedure:

A screenshot of a computer code

Description automatically generated

From this code we are going to obtain the plots. Let us present some difference and evidence between various yield curves

In Newton's methods, even when the algorithm reaches the maximum number of iterations, the resulting predicted plot consistently exhibits a remarkably precise fit to the historical yield data.

Moreover, the Nelson Siegel Svensson model tends to converge faster with respect to the original one.

## 5.1 Newton method - US

Here is the example for US bonds on the 1st of January 2022:

A graph with a line and a dotted line

Description automatically generatedA graph with a line and a line

Description automatically generated

The Nelson Siegel and the Nielson Siegel Svensson curves closely mirror the historical yields data, exhibiting a remarkably strong correspondence. The plots in this case are very similar to the ones of the Gradient Descent Methods.

Both curves tend to be very precise, even if the extended model is slightly better.

While studying the Gradient Descent method we noticed that the model wasn’t that good in fitting the curve of bonds with high volatility, if we apply the Newton method this issue emerges but is way less significant.

Let’s take for example the plots for US bonds on the 1st of March 2022:

A graph with blue and orange lines

Description automatically generatedA graph with a line and a line

Description automatically generated

In the case of the Gradient Descent, the Nelson Siegel model wasn’t able to fit the historical yield in the location of the hump. However, with our current method, we observe an improved fit, more able to capture the characteristics of the hump.

The same occurs for inverted yield curves, let’s see the case of October 2023:

A graph with a line and a blue line

Description automatically generatedA graph with a line and a line

Description automatically generated

This is given by the fact that the Newton method tends to converge faster to the optimal solution given the initial parameters, so the errors between the observed and predicted data are minimized in less iterations.

Furthermore, since the Newton method involves computing the Hessian, it effectively incorporates information about the curvature of the function. As a result, it can more accurately capture the presence of humps in the data, leading to improved fitting performance.

# 6 Comparison of Gradient Descent and Newton methods

Based on the results we have presented so far, it's evident that disparities exist between the outcomes obtained using the Gradient Descent method and those using Newton's method. These discrepancies extend beyond mere implementation nuances and likely stem from inherent divergences in the optimization strategies employed by each method.

First, we noticed that in the case of the Gradient Descent method convergence to the optimal values is way slower, while Newton always converges in less iterations. This difference is given by the fact that the former method doesn’t consider information about the curvature of the function, while the latter is able to capture them by employing the Hessian.

Newton's method has stronger constraints in terms of the differentiability of the function than Gradient descent. If the second derivative of the function is undefined in the function's root, then we can apply gradient descent on it but not Newton's method.

Moreover, the fact that the second derivative is required to apply the Newton method makes it more computationally expensive with respect to the Gradient Descent one, especially in the case of high-dimensional problems.

Newton and gradient descent method are gradient based algorithms, given that, a starting point is chosen and then the algorithm is forced to explore a small area around it. For this reason, other methods tend to outperform with respect to them, since we are trying to optimize a function with multiple local optima.

The gradient based method depends heavily on the starting points, so if the ones chosen are not the correct ones, this can lead to heavy errors in the optimization. However, from a financial perspective, employing the model in a financial frame allows a good initial guess of the parameters, leading to the possibility to make use of gradient-based methods in an efficient way.

Newton and gradient descent method are gradient based algorithms, given that, a starting point is chosen and then the algorithm is forced to explore a small area around it. For this reason, other methods tend to outperform with respect to them, since we are trying to optimize a function with multiple local optima.

Moreover, for what concern the Svensson model, proof have been found that gradient based methods are unable to exploit the extra term of the extended model, so they seem not to be that efficient while studying this algorithm.

# Other approaches implemented in the optimization of the model

We will see the most common approaches used to get an estimation of the model parameters.

We underline that, the approaches that are most used by analyst are, ordered by number of times used:

1. ORDINARY LEAST SQUARES
2. QUASI-NEWTON METHOD: BFGS algorithm
3. GRADIENT DESCENT METHOD

### ORDINARY LEAST SQUARES – RIDGE REGRESSION APPROACH:

Due to the convenience and the simplicity of linearizing the model, grid search or OLS approach are the most used estimation procedure to optimize the Nielson-Siegel model.

This approach is performed by fixing the parameter of the shape λ.

The main issues recurring in these approaches are that they can occur in the complication that the Nielson-Siegel model becomes heavily collinear, and this depends on the fixed shape parameter.

This problem is solved by performing the so-called ridge regression: in explicit a regularization term is added to the traditional least squares objective function of linear regression.

The regularization term is composed by a regularization parameter, which controls the strength of the regularization, multiplied by the sum of the squared values of the coefficients.

The aim of this regularization term is to help to prevent overfitting and to stabilize the model by providing a balance between a well fit to the data and the necessity of keeping coefficients far from being too large.

In this approach the non-linear problem is transformed into a linear one by fixing λ, that is the parameter responsible of the lack of linearity in the model; once the model has been linearized, in order to obtain parameters that ensures the best fit, they are estimated by OLS, conditional upon a grid of the fixed shape parameter.

The parameters are estimated by minimizing the sum of squared error, by using a grid search in order to determine the optimal λ, which is chosen according to the value of the R², and linear regression.

Once the optimal λ has been detected, there is the need to test the degree of multicollinearity of the two factors, if this degree is too high the results need to be re-estimated using ridge regression.

The main advantage of this method is that it does not depend on any starting values, while non-linear optimization techniques give estimates that are very sensitive to the starting value of the optimization.

On the other hand, as we stated before, in this case it must be kept in mind that the Nielson-Siegel model is very sensitive to the choice of the λ (which is usually set to be 1.37, Diebold and Li (2003) or 3, Fabozzi et al. (2006)). To understand the magnitude of this issue de Pooter (2007) studied that with different λ fixed, the remaining parameters can take extreme values.

Moreover, the correlation between the two parameters varies depending on the remaining maturity of the financial instruments chosen in the bootstrap.

# 7 Quasi-Newton Methods and BFGS algorithm

We solved our optimization problem first using gradient descent and then using Newton method. We highlighted the major differences between these 2 models. Summarizing, the Newton method:

* Does not need a learning rate parameter
* Converges much faster than gradient descent
* Is sensitive to initial conditions, more than gradient descent, especially if our objective function is non-convex.
* Is very computationally expensive, with a computational time of O(, due to the fact that we have to compute the Hessian and its inverse

We can obtain a method which is a sort of hybrid between gradient descent and Newton’s method, where we can have faster convergence than gradient descent, but lower operational cost per iteration than Newton’s method. This class of optimization methods is called *quasi-Newton methods.*Recall that in Newton’s method, we have to make the following update at each iteration:

where is a positive definite Hessian. If instead of the Hessian we use an approximation , we can have a much faster algorithm comparing to Newton’s method. This happens because the positive definite matrix B is updated iteration to iteration using information computed from previous steps, so we compute fewer new quantities at each iteration.  
  
A common feature to all the quasi-Newton methods is that the Hessian approximation B must satisfy the *quasi-Newton condition* (or *secant equation*):

which is obtained from the first order Taylor expansion of about . This condition essentially states that the product of the approximation of the inverse Hessian () and the change in the variable space () should approximate the difference in gradients (). In other words, it expresses a relationship between changes in the variable space and changes in the gradient.

We can understand quite easily this condition in one dimension case, where we replace the second derivative with its finite difference approximation, but things could complicate when dealing with n-dimensional secant condition.

The problem now is that our equation only has n components, while B is in general a symmetric n × n matrix with n(n+1)/2 components. We are dealing with an undetermined system (while in the one-dimension case we had a square system).

The quasi-Newton method solves this problem by imposing further constraints on B to solve for it. These additional constraints depend on the specific quasi-Newton method used. In this case, we focus on the BFGS method, which is one of the most popular quasi-Newton methods.

## 7.1 BFGS optimization

The name of the method comes from the names of its creators: Broyden, Fletcher, Goldfarb, and Shanno, who each came up with the algorithm independently in 1970.

To determine a scheme for B in n>1 dimensions, we will need additional constraints. Two of them are positive-definiteness and symmetry of B(these properties should be valid in each update).

A third property we want is for to be sufficiently close to at each update . We make use of the matrix norm to characterize this property. But we recall from the Newton method that we need the Hessian’s inverse (and not the Hessian itself), so we can compute it directly:

,

where and

Instead of requiring the full Hessian matrix at the point to be computed as ,,,, the approximate Hessian at stage *k* is updated by the addition of two symmetric rank-one matrices:

In order to maintain the symmetry and positive definiteness of , the update form can be chosen as:

We impose the secant condition and we choose and . We then have:

Substituting and into the previous equation we have the BFGS update:

## 7.2 Code implementation of BFGS algorithm

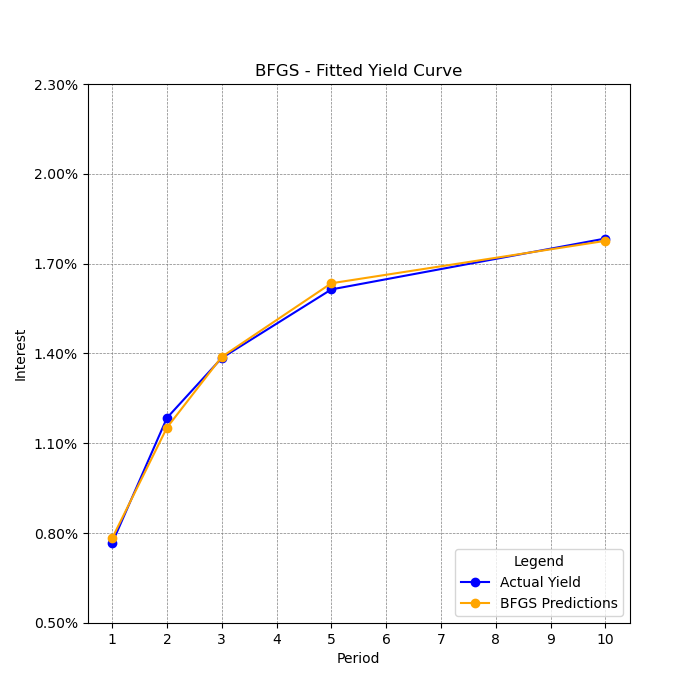
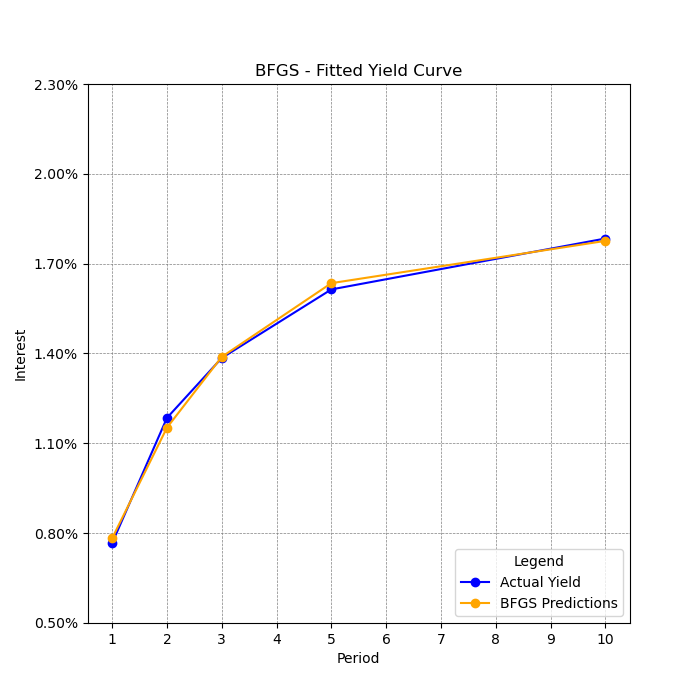
We started the code by defining 4 empty vectors to store values for parameters and f values for NS and NSS model.

We then implemented a for cicle in which, for each country at each point in time:

1. The function params() is minimized
2. The results are saved in two lists
3. The bond’s yield with NS and NSS are computed on the last results and saved into two dataframes
4. The graph of NS and NSS model with BFGS method are plotted against our sample data
5. The parameters and f values for NS and NSS are appendend into the corresponding dataframes
6. The dataframes are saved into excel files



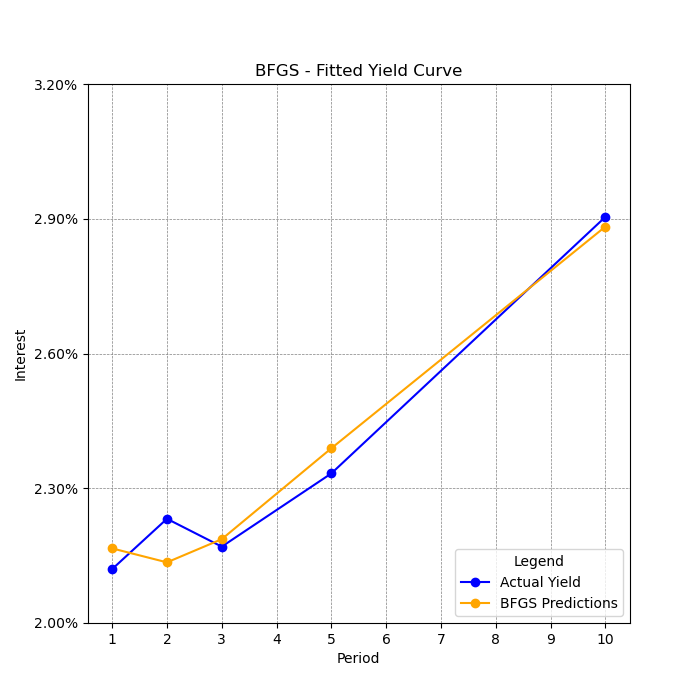
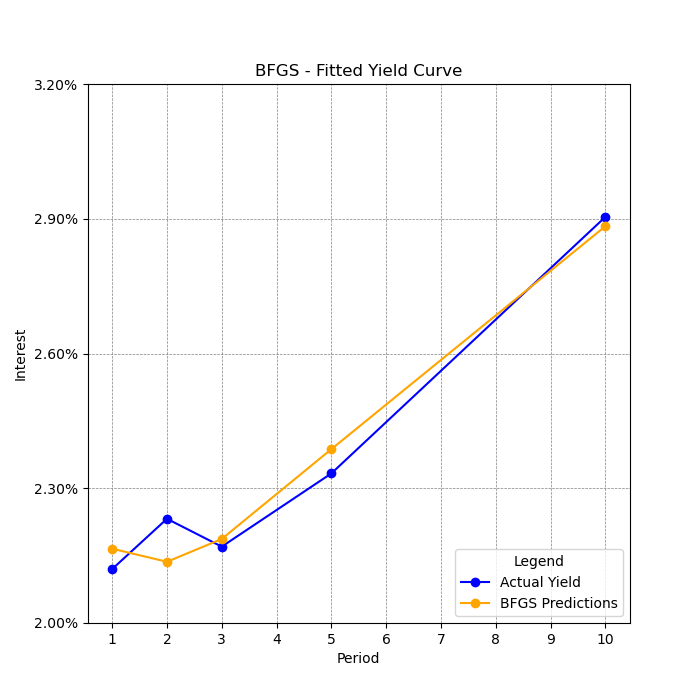
Let’s now observe the resulting graphs. Here is the example for US bonds on the 1st of January 2022:



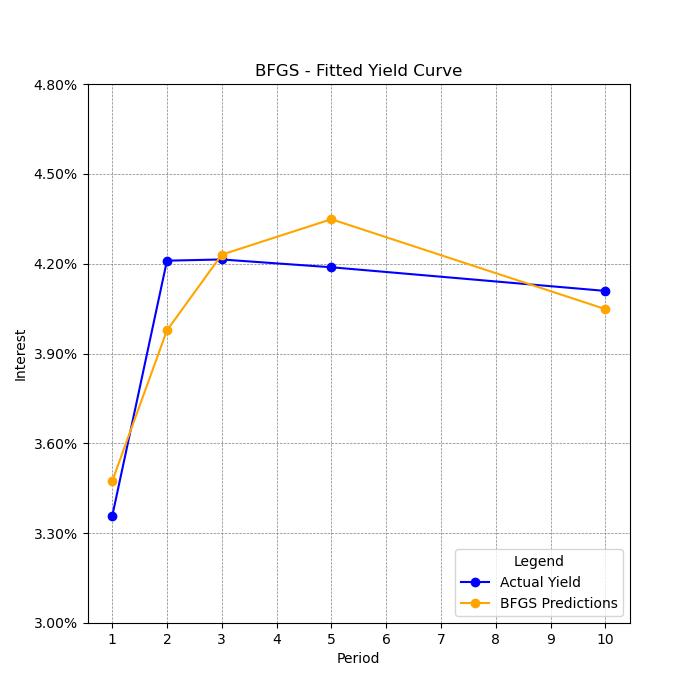
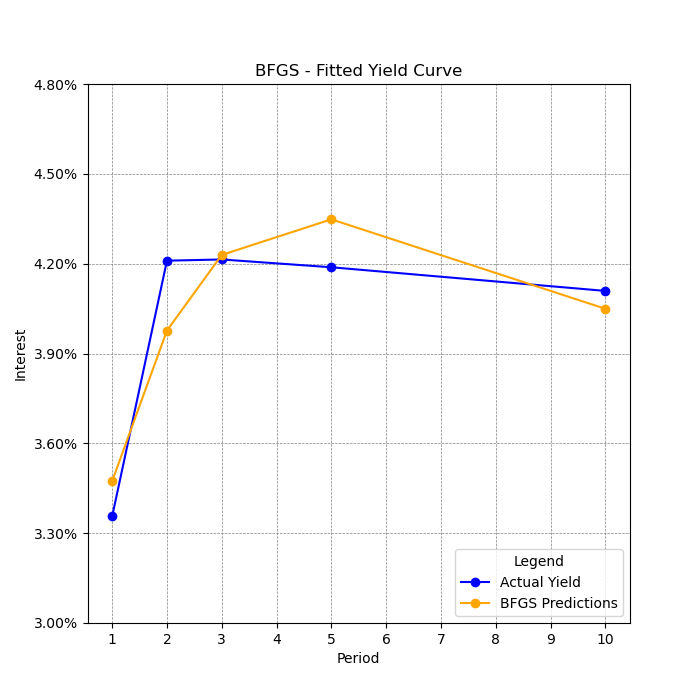
As in the Newton case, here both NS and NSS curves fit almost perfectly the historical yields data., Unfortunately, we cannot appreciate the difference between the 2 models just by looking at these graphs.   
  
  
The main difference between BFGS and the previously mentioned methods is the faster convergence: NS model converges in just 3 iterations with BFGS method, while both Gradient descent and Newton took 100 iterations (which we recall is the maximum value we set for iterations). NSS using BFGS is slower than NS, taking 5 iterations to optimize the function.

Unlike the other methods, here we cannot observe any notable difference between the graphs of NS and NSS. This happens for all countries at all points in time.

Here an example for Portugal bonds on the 1st of November 2022:



And another example for South Korean bonds on the 1st of September 2022:



What instead seems to be clear (especially from these 4 graphs) is that the BFGS finds itself in difficulty in cases where bond yields do not constitute a real curve. In this case the BFGS method approximates the returns with a curvilinear shape and, consequently, turns out to be less precise.

# 8 The Levenberg-Marquardt (LM) algorithm

The last method we used to fit our data is the Levenberg-Marquardt algorithm (from now on we’ll refer to it as “LM”), which is used to solve non-linear least squares problems. This algorithm is a blend of two minimization methods: the gradient descent method and the Gauss-Newton method. It was first published in 1944 by Kenneth Levenberg[[1]](#footnote-1) and then rediscovered in 1963 by Donald Marquardt[[2]](#footnote-2).   
  
The main advantages of LM are:

* Fast convergence: The LM algorithm can converge faster than standard gradient descent method.
* Robustness: The LM algorithm is more robust than Newton algorithm, which means that in many cases it finds a solution even if it starts very far off the final minimum.
* Efficiency in small datasets: Particularly effective in scenarios with limited data.

While the main limitations are:

* High computational time: The LM algorithm tend to be slower than Newton algorithm and, for large datasets, it is therefore better to use another algorithm with a lower computational time.
* Calibration of lambda: Choosing the right damping factor can be challenging and may require experimentation.

Let’s now observe the algorithm’s formulation. Given a set of empirical pairs of independent and dependent variables, we want to find the parameters of the model curve so that we minimize the sum of the squares of the residuals :

Like with the other methods, we have to provide an initial guess for the vector , which will be:

* An uniformed standard guess like in case with only one minimum
* A guess as close as possible to the final solution in case with multiple minima

In each iteration step, the parameter vector is replaced by a new estimate . The function is approximated as:

is the gradient of with respect to .

The sum of square residuals is minimized at a zero gradient with respect to . The first-order approximation of is:

If we take derivative of this approximation with respect to and set the result equal to zero, we get:

where is the Jacobian matrix of size .

The multiplication ) yields a square matrix which, in combination with vector of size n resulting from the product on the right-hand side of the equation, form a set of linear equations, which can be solved for .

The most important point of this model is the addition of the damping factor to the last equation, which becomes:

where is the identity matrix.

Small values of results in a Newton update, while large values of results in a gradient descent update. is initialized to be large so that first updates are small steps in the steepest-descent direction. If any iteration happens to result in a worse approximation, then λ is increased.

Otherwise, as the solution improves, is decreased, the LM method approaches the Gauss-Newton method, and the solution typically accelerates to the local minimum.

Unfortunately, for large values of , the step will be taken approximately in the direction opposite to the gradient. If either the length of the calculated step or the reduction of sum of squares from the latest parameter vector fall below predefined limits, iteration stops, and the last parameter vector is considered to be the solution.

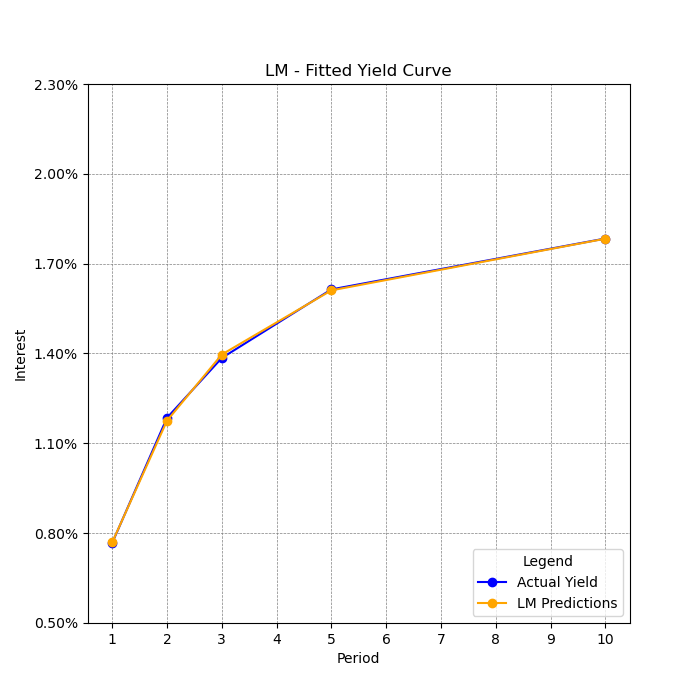
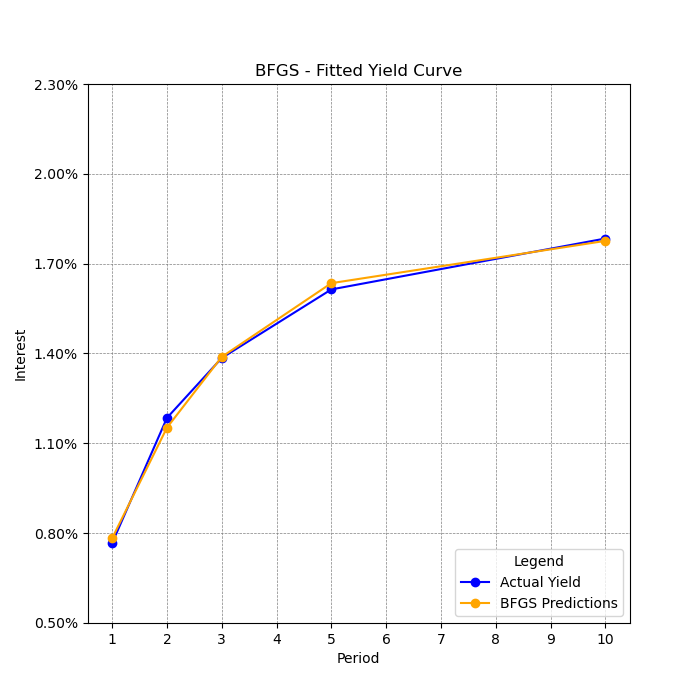
To make the solution scale invariant Marquardt's algorithm solved a modified problem with each component of the gradient scaled according to the curvature. This provides larger movement along the directions where the gradient is smaller, which avoids slow convergence in the direction of small gradient. Fletcher (1971) simplified the form, replacing the identity matrix I with the diagonal matrix consisting of the diagonal elements of :

## 8.1 Code implementation of LM algorithm and comparison with BFGS

Here we show the code implementation of LM, which is very similar to that of BFGS method.

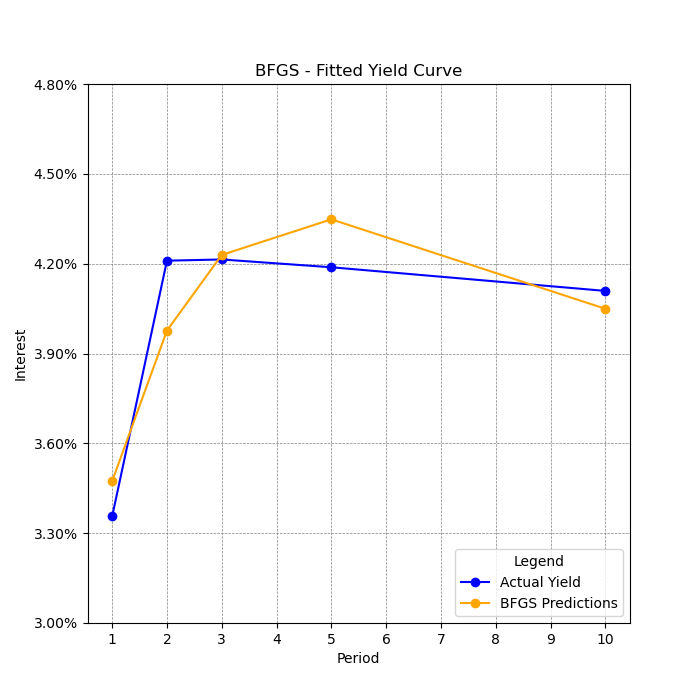
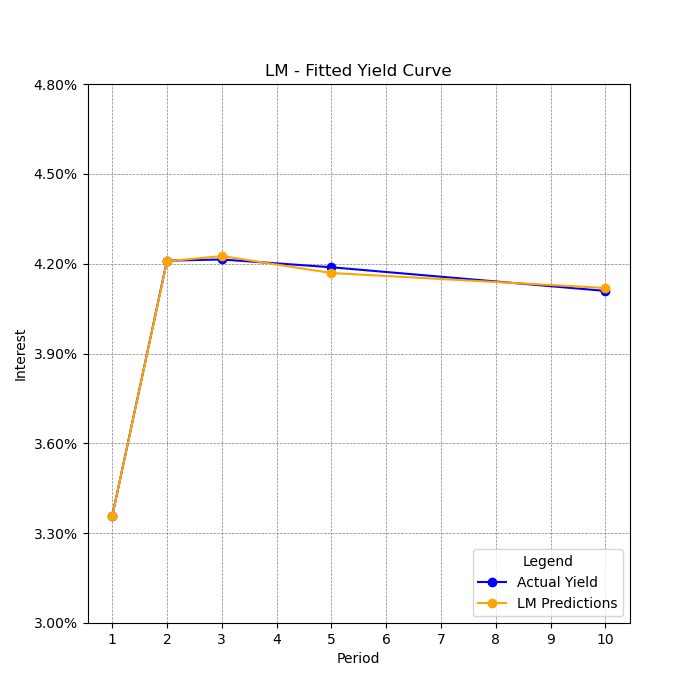


We can now observe how this last model performs on our sample data. Here we show the graph for Us bonds on the 1st of January 2022 (comparison between LS method and BFGS method for NS model):

The fit in the LM case is almost perfect, improving the improving the already excellent result obtained with BFGS. Furthermore, this is achieved with the same number of iterations as BFGS, i.e. 3.

In the LM case what conviced us the most is that this method has a very good fit to sample also when there is some volatility. We report again a comparison between LM and BFGS but now on South Korean bonds on the 1st of September 2022:

We clearly see the difference in fit between the 2 methods. LM does not tend to create a curve on bond yields like BFGS does. The result is an almost perfect fit also when volatility shows up.

# Bibliography:

Diebold, F. X., Li, C., and Yue, V. Z., “Global Yield Curve Dynamics and Interactions: A Generalized Nelson-Siegel Approach”, Journal of Econometrics, 146 (2008), 351-363

Fabozzi, F. J., Martellini, L. And Priaulet, P., “Predictability in the Shape of the Term Structure of Interest Rates”, Journal of Fixed Income (June, 2005) 40-53

Annaert J., Claes A.G.P., De Ceuster M.J.K., Zhang H., “Estimating Nielson-Siegel: A ridge regression approach”, Universiteit Antwerpen (2010)

P. Manousopoulos, M. Michalopoulos, “Comparison of non-linear optimization algorithms for yield curve estimation”, European Journal of Operational Research 192 (2009) 594–60

Wirz, Matt, and Saeedy, Alexander. “Bond Markets Forecast Long Financial Freeze for Russia.” Wall Street Journal, March 15, 2022.

Gilli et al., “Calibrating the Nelson-Siegel-S

Nocedal, J., Wright, S., Numerical Optimization (Chapter 6). Springer, 2nd edition, 2006

Goldfarb, D.F., “A family of variable-metric methods derived by variational means”, Math. Comp. 24, 23-26 (1970)

Shanno D., “Conditioning of quasi-Newton methods for function minimization”, Math. Comp. 24, 23-26 (1970)

Gavin, H. P. (2022). “The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems”, Duke University.

Levenberg, K. (1944). “A method for the Solution of Certain Non-Linear Problems in Least Squares”. *Quarterly of Applied Mathematics.* **2** (2): 164–168*.*

Marquardt, D. (1963*). "*An Algorithm for Least-Squares Estimation of Nonlinear Parameters*". SIAM Journal on Applied Mathematics*. **11** (2): 431–441

Fletcher, R. (1971) “A modified Marquardt subroutine for non-linear least squares (technical report)”, Harwell, Atomic Energy Research Establishment,

1. Levenberg, Kenneth (1944). “A method for the Solution of Certain Non-Linear Problems in Least Squares”. Quarterly of Applied Mathematics. **2** (2): 164–168. [↑](#footnote-ref-1)
2. Marquardt, Donald (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". SIAM Journal on Applied Mathematics. **11** (2): 431–441 [↑](#footnote-ref-2)