1 Theoretical Background

1.1 Nelson-Siegel model

The Nelson-Siegel model, introduced in 1987 by Charles R. Nelson and Andrew F. Siegel, is one of the most widely used methods to represent the term structure of interest rates. The main goal of the model was to provide a practical, flexible, and parsimonious approach to estimating yield curves for financial markets. The modeling of the yield curve provides a representation of the relationship between yields and the maturities of bonds or other financial instruments.

The Nelson-Siegel model (1987) is expressed as:

$$R(t) = \beta_0 + \beta_1 \left(\frac{1 - e^{-t/\tau}}{t/\tau} \right) + \beta_2 \left(\frac{1 - e^{-t/\tau}}{t/\tau} - e^{t/\tau} \right) \tag{1}$$

where $R(t)$ is the yield at time t, and $\beta_0, \beta_1, \beta_2$ and τ are parameters, with $\tau > 0$.

The parameters have specific interpretation:

- β_0 : *The Long-Term Level*
  The constant β_0 represents the long-term level of the yield curve, which is the value the curve converges to as the time to maturity t becomes very large (goes to infinity);

- β_1 : *The Slope*
  The parameter β_1 controls the initial slope of the yield curve. A positive or negative β_1 reflects the steepness of the curve (upward or downward). Since the term $\frac{1-e^{-\tau/t}}{\tau/t} \to 1$ when $t \to 0$ and gradually decreases to 0 as $t \to \infty$, β_1 primarily influences short-term maturities. The sum $\beta_0 + \beta_1$ determines the starting value of the yield curve for $t = 0$;

- β_2 : *The Curvature*
  The parameter β_2 captures the curvature of the yield curve. A higher value indicates greater concavity or convexity in the curve's shape. Since the term $\frac{1-e^{-\tau/t}}{\tau/t} - e^{-\tau/t} \to 0$ for both short-term ($t \to 0$) and long-term ($t \to \infty$), β_2 affects the middle segment of the curve;

- τ : *The Decay Factor*
  The parameter τ governs the rate of decay of the effects of β_1 and β_2. It determines how quickly the short-term and medium-term components of the curve converge to the long-term level (β_0).

1.2 Nelson-Siegel-Svesson model

The Nelson-Siegel-Svensson (1994) model is an extension of the Nelson-Siegel model designed to provide additional flexibility in fitting the term structure of interest rates. The Nelson-Siegel-Svensson model introduce two new parameters β_3 and τ_2 making the model capable of capturing more complex yield curve shapes. The form of the Svensson model is as follows:

$$R(t) = \beta_0 + \beta_1 \left(\frac{1 - e^{-t/\tau_1}}{t/\tau_1} \right) + \beta_2 \left(\frac{1 - e^{-t/\tau_1}}{t/\tau_1} - e^{-t/\tau_1} \right) + \beta_3 \left(\frac{1 - e^{-t/\tau_2}}{t/\tau_2} - e^{-t/\tau_2} \right) \tag{2}$$

The first three terms ($\beta_0, \beta_1, \beta_2$) and τ_1 retain the same roles as in the Nelson-Siegel model.
The new parameters [**comparative**]:

- β_3 : *Additional Curvature*
  β_3 introduces a second curvature term into the model, allowing the yield curve to exhibit a second hump or dip at maturities not captured by β_2. The term $\frac{1-e^{-\tau_2/t}}{\tau_2/t} - e^{-\tau_2/t}$ is controlled by the second decay factor (τ_2): it is zero at both short ($t \to 0$) and very long maturities ($t \to \infty$);

- τ_2 : *Decay Factor for the Additional Curvature*
  While τ_1 determines the speed of convergence for the original curvature term (β_2), τ_2 determines where the second hump or dip introduced by β_3 will appear on the yield curve.

The Svensson model offers greater flexibility by incorporating an additional curvature term and decay factor, allowing it to fit complex and irregular yield curves more effectively. However, the model's increased complexity requires more data and careful calibration to avoid overfitting, while the additional parameters make interpretation less intuitive.
The Svensson model is widely applied in bond pricing, accurately fitting yield curves for discounting cash flows. It aids in risk management by offering insights into yield curve shapes to manage interest rate risk. Its flexibility also makes it effective for forecasting complex yield curve movements.

1.3 Gradient Descent Method for Optimization

Gradient descent is an optimization algorithm commonly used in training machine learning models. It operates on a convex function and iteratively adjusts its parameters to minimize a given function until it reaches a local minimum. The process begins with the initialization of parameter values, including the starting point $x_{(0)}$, the number of iterations N, and the descent direction $d^{(i)}$. After initialization, the algorithm employs calculus to update these values iteratively, thereby reducing the cost function.

$$x^{(i)} = x^{(i-1)} + \alpha d^{(i)} \tag{3}$$

where,

$$d^{(i)} = -\nabla f(x^{(i)}) \tag{4}$$

A gradient measures the rate of change in all weights concerning the variation in error. It can be interpreted as the slope of a function: a higher gradient indicates a steeper slope, allowing the model to learn more rapidly, whereas a zero gradient signifies that learning has halted. Mathematically, a gradient is defined as a partial derivative with respect to its inputs. The size of the steps taken by gradient descent toward the local minimum is determined by the learning rate, which controls the speed of convergence towards the optimal weights.

For the gradient descent algorithm to effectively reach the local minimum, it is crucial to set an appropriate learning rate—neither too high nor too low. If the learning rate is too large, the algorithm may oscillate around the minimum without converging, as it overshoots due to excessive step sizes. Conversely, if the learning rate is too small, convergence will occur, but the process may be excessively slow.

A sufficiently good step size can be approximated by a *line search*, this method starts from a suitable α_0 and reduces it until a sufficient decrease in the objective function is given.When gradient descent is unable to further reduce the cost function and stabilizes at a particular level, it is said to have converged.[**builtin˙gradient˙descent**]

1.4 Newton's Method for Optimization

Newton's method is similar to gradient descent method, but update step is computed using second-order Taylor expansion. It can be extended to solve optimization problems by finding the minima or maxima of a real-valued function $f(x)$. It refines the solution iteratively using the first and second derivatives. Given an approximation X_n, the next iteration is computed as:

$$X_{n+1} = X_n - \frac{f'(X_n)}{f''(X_n)} \tag{5}$$

Intuitively, this process adjusts X_n based on the ratio of the gradient $f'(X_n)$ and the curvature $f''(X_n)$. If the function is convex (positive curvature), the update moves X_n toward the minimum. If it is concave (negative curvature), the update moves toward the maximum. This iterative process continues until a stopping criterion is met or the solution converges.
Newton's method exhibits *quadratic convergence*, meaning the number of correct digits approximately doubles with each iteration. However, its efficiency depends on several factors. The initial guess significantly influences convergence; a good starting point ensures rapid progress, while a poor one may lead to divergence. The function itself must also behave well near the minimum, with continuous first and second derivatives. Discontinuities or singularities can hinder convergence. Finally, the method typically stops when the change in x between iterations becomes negligible or when the function value is sufficiently close to zero.
While Newton's method is highly efficient, it is not always reliable. If the initial guess is far from the true critical point, or if the function exhibits complex behavior such as oscillations or multiple roots, the method may fail to converge.[**geeksforgeeks˙newton**]

2 Data Collection

The real data selected for comparison and for calculating the loss function were collected from three continents, with one representative country per continent chosen to reflect the overall trend of its respective geographical area. Furthermore, all data has been taken from Investing.com.

For Europe, we selected weekly data spanning 262 observations, corresponding to approximately five years, on short-term bonds (1-year) and medium-term bonds (5-year) for Italy. For Asia, we selected weekly data spanning 262 observations, corresponding to approximately five years, on short-term bonds (1-year) and medium-term bonds (5-year) for Japan. Lastly, representing the American continent, we selected weekly data spanning 262 observations, corresponding to approximately five years, on short-term bonds (1-year) and medium-term bonds (5-year) for the United States.

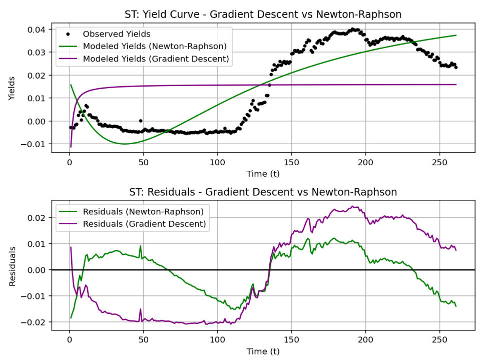
The decision to present only the charts and data related to the study of Italian bonds in this report is motivated by two main factors. Firstly, the models analyzed showed a lower fit to Italian bond data compared to those for U.S. and Japanese bonds. This discrepancy can likely be attributed to the greater variability observed in the Italian bond data. Secondly, the mathematical conclusions derived from the study of Italian bonds are essentially the same as those that would be obtained from analyzing the yields of bonds from the other two countries. Therefore, for reasons of space and to avoid redundancy, we chose to focus solely on the Italian bond data.

Method	**Loss**	**Running Time(s)**	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	0.0782	0.2023	0.0160	-0.0469	0.0088	-	1.0001	-
Newton NS	0.0169	0.4638	0.0613	-0.0437	-0.1632	-	30.3037	-
Gradient NSS	0.0739	0.3907	0.0172	-0.0420	0.0129	-0.0388	0.9999	2.0002
Newton NSS	0.0162	1.7732	0.0904	-0.0891	0.1842	-0.3801	28.5757	39.9692

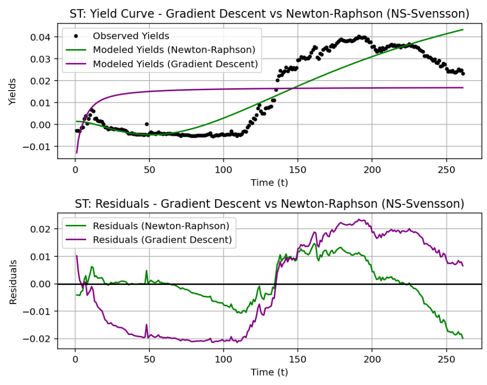
Table 1: Final comparison: Short Term, Italy, reasoned starting point, gradient step size determined via line search, constant(=1) Newton step size.

Method	**Loss**	**Running Time(s)**	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	0.0560	0.2809	0.0182	0.0080	0.0643	-	1.5008	-
Newton NS	0.0185	0.5068	-0.0127	0.0204	0.1490	-	24.4118	-
Gradient NSS	0.0534	0.3252	0.0169	-0.0036	0.0530	0.0440	1.5004	3.0004
Newton NSS	0.0480	1.5748	0.0132	0.0859	-0.3766	0.2993	1.7110	3.3741

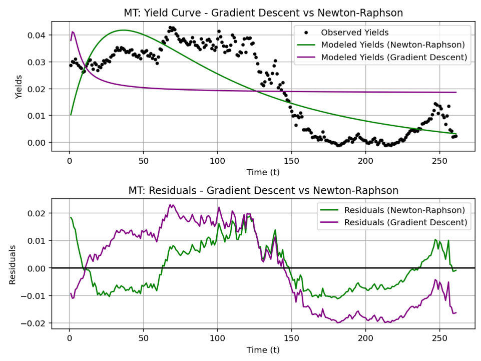
Table 2: Final comparison: Medium Term, Italy, reasoned starting point, gradient step size determined via line search, constant(=1) Newton step size.



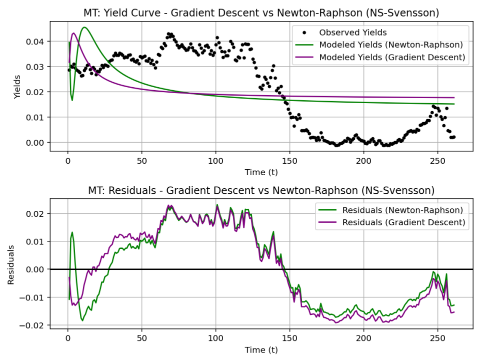
(a) Calibration of NS model for one-year maturity bonds



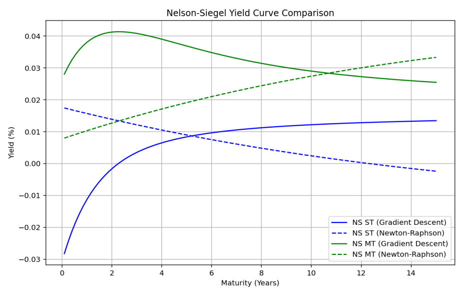
(b) Calibration of NSS model for one-year maturity bonds



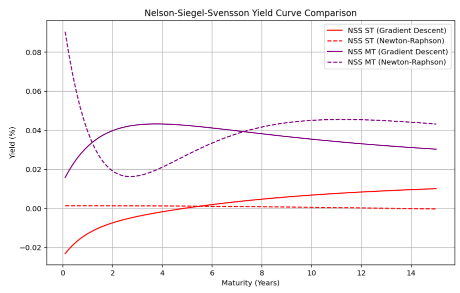
(c) Calibration of NS model for five-year maturity bonds



(d) Calibration of NSS model for five-year maturity bonds



(e) NS yield curve comparison



(f) NSS yield curve comparison

Figure 2: Calibration of NS and NSS models for short- and medium-term Italian Government bonds. The gradient step size is determined via line search and the Newton step size is constant(=1).

Method	Loss	Running Time(s)	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	0.0805	0.2761	0.0151	-0.2048	0.0952	-	0.1050	-
Newton NS	0.0781	0.5633	0.0159	-0.2256	-0.0098	-	0.2246	-
Gradient NSS	NaN	0.0858	NaN	NaN	NaN	NaN	NaN	NaN
Newton NSS	0.0296	1.6970	0.1017	-0.0160	-0.3636	-0.3099	2.1215	49.6788

Table 3: Final Comparison: Short Term, Italy, starting point [0.10, -0.20, 0.10, 0.1] and gradient step size determined via line search and constant (=1) Newton step size.

Method	Loss	Running Time(s)	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	NaN	0.0332	NaN	NaN	NaN	-	NaN	-
Newton NS	0.0169	0.4816	0.0613	-0.0437	-0.1632	-	30.3037	-
Gradient NSS	NaN	0.0486	NaN	NaN	NaN	NaN	NaN	NaN
Newton NSS	0.0162	1.8072	0.0904	-0.0891	0.1842	-0.3801	28.5757	39.9692

Table 4: Final Comparison: Short Term, Italy, reasoned starting point, constant (=1) gradient step size and constant (=1) Newton step size.

Method	Loss	Running Time(s)	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	271.5968	0.0221	1.0336	0.0003	0.0530	-	0.9962	-
Newton NS	0.0169	0.4511	0.0613	-0.0437	-0.1632	-	30.3037	-
Gradient NSS	319.4348	0.0490	1.1187	-0.0005	0.0524	0.0333	0.9954	1.9973
Newton NSS	0.0162	1.7909	0.0904	-0.0891	0.1842	-0.3801	28.5757	39.9692

Table 5: Final Comparison: Short Term, Italy, reasoned starting point, the gradient step size decreases by a constant factor (=0.1) and constant (=1) Newton step size.

Method	Loss	Time (s)	β_0	β_1	β_2	β_3	τ_1	τ_2
Gradient NS	5.08×10^{50}	0.0233	5.75×10^{9}	-3.22×10^{26}	-4.41×10^{41}	-	1.46×10^{21}	-
Newton NS	0.0169	0.5491	0.0613	-0.0437	-0.1632	-	30.3037	-
Gradient NSS	3.78×10^{50}	0.0625	2247.1625	3.97×10^{18}	-1.90×10^{36}	-1.90×10^{36}	-1.05×10^{21}	-6.12×10^{36}
Newton NSS	0.0162	1.5465	0.0904	-0.0891	0.1842	-0.3801	28.5757	39.9692

Table 6: Final Comparison: Short Term, Italy, reasoned starting point, the gradient step size decreases by a constant factor (=0.9) and constant (=1) Newton step size.

Method	Loss	Running time(s)	beta0	beta1	beta2	beta3	tau1	tau2
Gradient NS	0.0782	0.1983	0.0160	-0.0469	0.0088	-	1.0001	-
Newton NS	0.1090	0.4520	0.0233	-0.0070	0.0413	-	1.0046	-
Gradient NSS	0.0739	0.3840	0.0172	-0.0420	0.0129	-0.0388	0.9999	2.0002
Newton NSS	0.1171	1.6610	0.0235	-0.0042	0.0457	0.0133	1.0026	2.0028

Table 7: Final Comparison: Short Term, Italy, reasoned starting point, the gradient step size determined via line search and the Newton step size decreases by a constant factor (=0.1).

Method	Loss	Running time (s)	beta0	beta1	beta2	beta3	tau1	tau2
Gradient NS	0.0782	0.1902	0.0160	-0.0469	0.0088	-	1.0001	-
Newton NS	0.0174	0.4773	0.0594	-0.0408	-0.1637	-	28.8631	-
Descent NSS	0.0739	0.3254	0.0172	-0.0420	0.0129	-0.0388	0.9999	2.0002
Newton NSS	0.0306	1.7602	0.0389	-0.0842	0.5985	-0.6009	6.2868	9.5068

Table 8: Final Comparison: Short Term, Italy, reasoned starting point, the gradient step size determined via line search and the Newton step size determined via line search.

3 Comment on the result

3.1 Comments on the behavior of optimization models

From a mathematical point of view, in the case of a **reasonable starting point**, Newton's method proves to be more accurate and reliable than Gradient Descent, as evidenced by the lower final loss achieved for the NS model: 0.0169 for Newton versus 0.0782 for Gradient Descent. This confirms the superior effectiveness of Newton in quickly finding the optimal solution. However, the running time for Newton is higher (around 0.4638 seconds) compared to Gradient Descent (0.2023 seconds). The main reason for this difference lies in the computation of the Hessian matrix and its inversion, which are computationally expensive operations for Newton, while Gradient Descent only requires the computation of the gradient, making it more computationally efficient.

Otherwise, when the **initial point is far from the optimal solution**, Gradient Descent significantly worsens its performance, often failing to converge at all, as indicated by NaN values in the tables. This behavior highlights the method's sensitivity to the choice of a well-calibrated starting point. On the other hand, Newton shows greater robustness, managing to converge to an acceptable solution despite the higher computational cost.

With a **constant step size set to 1**, Gradient Descent tends to diverge or produce highly inaccurate results, as shown by the NaN values obtained in some scenarios. This behavior reflects the method's sensitivity to the step size value: if too large, the method may oscillate around the solution or even diverge, never reaching convergence.

Instead, with a constant decreasing step size, two scenarios can be observed:

- **Small Step (rate = 0.1)**: In this case, Gradient Descent shows extremely slow but stable convergence. This type of step minimizes the risk of divergence, but at the cost of slow convergence and poor final solution quality. The final losses remain very high, at 271.5968 for NS and 319.4348 for NSS.
- **Large Step (rate = 0.9)**: In this case, Gradient Descent produces highly unstable or divergent results, as evidenced by extremely large values (5.0810^{50}).

This distinction highlights how the behavior of Gradient Descent is strongly influenced by the initial step size: a step size that is too small drastically slows down convergence, while a step size that is too large leads to disastrous results.

The Newton method with a constant step size (=1) shows a good ability to converge quickly to an optimal solution. The final loss of the Newton method with line search (0.0174) is almost identical to that obtained with the constant step size, but it is still a more adaptive approach that could offer advantages in more complex scenarios. Using constant decreasing step size (rate = 0.1) slows down the convergence speed, as indicated by the higher loss value (0.1171), and leads to a significant increase in execution time (1.6610 seconds). While this approach may contribute to greater numerical stability, the trade-off in terms of solution quality and computational performance is negative.

Finally, both the **Nelson-Siegel** and **Nelson-Siegel-Svensson** models show discrepancies between the observed and modeled data. Although the NS and NSS models are parsimonious and useful for describing yield curves, they may not be flexible enough to capture the full complexity of the bond market. The fit between the estimated and observed yields is not perfect, and this issue is reflected in the residuals' graphs, suggesting that the models do not fully capture the structure of the historical data.

In conclusion, **Gradient Descent** is more time-efficient than **Newton**, as evidenced by the lower running times in several scenarios. However, it suffers from instability and divergence in numerous configurations, requiring careful step size selection or the use of techniques such as line search to improve performance. Newton, despite its higher computational cost, remains more robust and accurate, making it particularly effective in complex scenarios or with poorly chosen starting points.

3.2 Economic interpretation

Both the ECB and the Fed responded to the inflation surge in 2022 with aggressive rate hikes, followed by expectations of gradual rate cuts as inflation peaked in 2024. This tightening phase led to a sharp upward shift in short-term yields, reflecting higher borrowing costs and a contractionary stance aimed at curbing inflation. As inflationary pressures began to ease, markets started pricing in future rate cuts, leading to a stabilization or slight decline in longer-term yields. In contrast, the BoJ maintained its zero-interest rate policy, in line with its historical approach, prioritizing economic stability and avoiding abrupt policy shifts. This long-standing accommodative stance resulted in a flatter yield curve, as market expectations for future rate hikes remain low and Japan's economy continues to grapple with subdued inflation and weak growth. The divergence in monetary policy paths highlights fundamental differences in economic conditions and central bank mandates, with the Fed and ECB focusing on inflation control while the BoJ remains committed to sustaining economic activity in a low-inflation environment.

4 Literature for other methods to do model calibration

4.1 The Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm was developed in the early 1960's to solve nonlinear least squares problems.The *Levenberg-Marquardt (LM)* algorithm combines gradient descent and Gauss-Newton iteration, with the aim of improving the limitations of these two methods, which are:

- **Gradient descent** updates parameters using the gradient but struggles with slow convergence in narrow valleys and is highly sensitive to the learning rate.
- **Newton's method** incorporates second-order derivatives for faster convergence but requires computing and inverting the Hessian matrix, which can be computationally expensive and sensitive to initialization.

The LM algorithm introduces a damping factor λ to transition between gradient descent and Gauss-Newton adaptively. The update rule is:

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i) \tag{6}$$

Then, if the error decreases, λ is reduced, favoring Gauss-Newton for faster convergence. Otherwise, if the error increases, λ is increased, shifting towards gradient descent for stability. Marquardt further refined this approach by replacing the identity matrix in the update rule with the diagonal of the Hessian, leading to:

$$x_{i+1} = x_i - (H + \lambda diag(H))^{-1} \nabla f(x_i) \tag{7}$$

This modification ensures larger steps in directions of low curvature and smaller steps in steep regions, effectively addressing the issue of narrow error valleys. Finally, while the LM method is not theoretically optimal, it performs exceptionally well in practice. Its primary drawback is the need for matrix inversion, which can become computationally expensive for large-scale models. Despite this, for problems involving a few hundred parameters, the LM algorithm is significantly faster than standard gradient descent methods and remains one of the most widely used approaches for non-linear optimization. [**ranganathan2004levenberg**]