TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Università degli Studi di Roma "Tor Vergata"

School of Economics

Master of Science in

Finance and Banking

Thesis in Fixed Income

Multiple Yield-Curve Construction Using QuantLib

The Supervisor
Prof. Stefano Herzel

The Student
Lorenzo Gallina

Accademic year 2023/2024

Contents

1 Introduction

The 2008 global financial crisis marked a significant turning point in the construction of interest rate curves. Prior to the crisis, financial markets predominantly used a single-curve framework for both discounting cash flows and forecasting future interest rates, operating under the assumption that interbank offered rates, such as the London Interbank Offered Rate (LIBOR), were proxy for risk-free rates. LIBOR, which represented the unsecured lending rates among major global banks, was considered a reliable benchmark due to the high creditworthiness of these institutions and was therefore widely applied across financial contracts, including derivatives, loans, and mortgages.

The crisis, however, revealed inherent weaknesses in this single-curve approach. As financial instability intensified, interbank lending rates, including LIBOR, experienced sharp increases driven by heightened concerns over counterparty risk. Consequently, LIBOR ceased to be viewed as a risk-free rate, and the spread between LIBOR and the Overnight Indexed Swap (OIS) rate—an almost risk-free rate from overnight lending markets—expanded significantly. This widening, known as the LIBOR-OIS spread, underscored the inadequacy of a single curve in capturing the different risk profiles associated with various financial instruments.

In response, the financial industry adopted a multi-curve framework, introducing separate curves for discounting and projecting forward rates, each calibrated to the specific tenors and risks of the instruments involved. OIS-based curves became the standard for discounting, while instruments such as swaps, forward rate agreements, and futures formed the foundation for constructing forward rate curves.

1.1 Objectives and structure of the Thesis

Rather than delving into the theoretical foundations of the multi-curve framework, which have been addressed in prior research summarized in Chapter 2.3, this thesis focuses on the practical aspects of curve construction.
The main objectives of this work are:

1. To explore the evolution of interest rate curve construction practices post-2008, with a particular focus on the transition from single-curve to multi-curve methodologies. This will include a description of key academic and industry studies that have shaped current practices in interest rate modeling.

2. To demonstrate the construction of the Euro Short Term Rate (€STR) and the Euro Interbank Offered Rate (EURIBOR) curves using QuantLib, a widely used library in quantitative finance. The thesis will document the curve-building process step by step, using real market data sourced from Refinitiv, a leading provider of financial data.

3. To analyze the implications of the multi-curve framework for constructing Euribor forward curves. We will explore how different instruments and methodologies affect the accuracy and stability of the forward curve.

4. To evaluate the impact of different sets of market instruments on the construction of these curves, including deposits, synthetic deposits, OISs, swaps, and FRAs. The stability of the curves will be tested through stress tests on bid-ask spreads to assess their robustness under different market conditions.

The thesis is structured as follows:
we begin by examining the evolution of interest rate curve construction, tracing the shift from the single-curve approach before the financial crisis to the multi-curve framework adopted post-2008, and highlight the key studies that influenced this transition. Following this, we introduce the computational tools, such as QuantLib, and describe the data preparation process alongside the methodology applied for curve construction complemented by some key definitions.

Subsequently, we analyze the market instruments central to our study, detailing their role in building the €STR and Euribor curves. The focus then shifts to the construction of the €STR discount curve, where we analyze its behavior, especially during the ECB's reserve maintenance periods.
The next chapter presents the construction of the Euribor 3M curves, comparing outcomes with and without synthetic deposits to emphasize their practical relevance. This is followed by a comparison of the single-curve and dual-curve bootstrapping approaches, highlighting how the two different techniques influence the results.
Finally, we discuss the importance of using tenor-consistent instruments and evaluate various bootstrapping configurations, testing their stability under different market scenarios. The thesis concludes with reflections on the findings and proposes areas for future research.

2 Evolution of Interest Rate Curve Construction Practices

2.1 Post-Crisis Developments: Transition to a New Paradigm

The widening of the LIBOR-OIS spread, which began in early August 2007, marked the first clear sign of stress in the interbank lending market. Although there were temporary signs of stabilization through late September and early November 2007, conditions quickly deteriorated by the end of November as it became evident that financial institutions were incurring substantial losses. This persistent widening, shown in Figure 2.1, signaled a growing distrust in the interbank market, with LIBOR increasingly reflecting counterparty risk rather than serving as a risk-free benchmark.

As the crisis unfolded, it became clear that the single-curve framework, which relied on LIBOR as a proxy for the risk-free rate, was inadequate for capturing the evolving risk landscape. This realization prompted the transition to a multiple-curve framework, where separate curves were developed for discounting and projecting forward rates. The OIS rate, perceived as a more accurate reflection of near risk-free conditions, emerged as the preferred discounting standard, enabling a more precise pricing of derivatives and a better representation of interest rate risk (Cui, In, and Maharaj, 2016).

In the Eurozone, this shift led to the gradual adoption of the Euro OverNight Index Average (EONIA) and EURIBOR as the main references for interest rate modeling. However, concerns regarding the reliability and validity of benchmarks conceived before the crisis, such as EONIA, prompted further reforms. Therefore, in 2019, the ECB introduced the Euro Short Term Rate (€STR) as the new risk-free benchmark, which progressively replaced EONIA

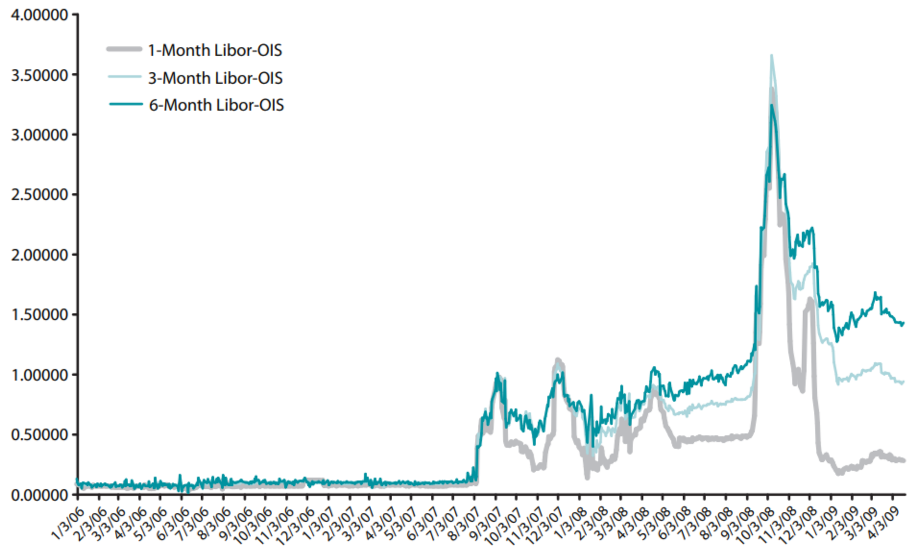

Figure 2.1: The LIBOR-OIS spreads from 2006 to 2009, capturing different tenors: 1-month (gray line), 3-month (light blue line), and 6-month (teal line). The spike during 2008 reflects the heightened counterparty risk and the breakdown of trust within interbank lending, signifying that LIBOR could no longer be regarded as risk-free. Source: St. Louis (2009) .

until its complete discontinuation in January 2022, while in January 2023, EURIBOR fully replaced LIBOR in the European context following its phasing out. We will delve into the reasons behind both transitions in Chapter 4 (ECB, 2021).

2.2 The Multiple-Curve Framework

In response to the 2008 financial crisis, central banks implemented unprecedented liquidity injections through their bank lending programs. This was particularly notable in the U.S., where such lending was tied to the Federal Funds rate, using short-term Treasury Bills as collateral. These risk-free assets quickly became the standard form of collateral in swap contracts (Charles River Development, 2016). However, as highlighted by Johannes and Sundaresan (2007), while collateralization helps to mitigate credit risk, it also influences swap rates due to the funding costs associated with maintaining collateral. This dynamic created the need for a distinct set of discount factors to price swaps accurately, leading to the development of the dual-curve stripping methodology. In this dual-curve approach, detailed in Chapter 3.4.4, market participants began using discount factors derived from the OIS curve for swap valuation. An OIS is an overnight interest rate swap where the floating rate is linked to the effective overnight rate published by central banks. Unlike LIBOR, which was based on interbank lending quotes, OIS rates are determined from actual transactions close to the central bank's target rate, making them more reliable. Different currency-specific OIS rates are applied for swap pricing; for instance, dollar-denominated OIS are based on the Fed Funds rate, while euro-denominated OIS historically used EONIA and, after the discontinuation in January 2022, the €STR. The adoption of this dual-curve approach gained widespread recognition in 2010 when LCH.Clearnet, a leading clearinghouse, officially announced its decision to employ OIS rate curves for discounting its \$218 trillion interest rate swap portfolio (LCH.Clearnet, 2010).

2.3 Relevant Literature

A considerable body of literature has examined the implications and methodologies associated with multiple-curve construction:

- Ametrano and Bianchetti (2013) offers a comprehensive overview of multiple-curve construction methods, discussing both theoretical concepts and practical implementation challenges. This paper remains a foundational reference in the field, offering insights valuable for both academic research and industry practice.

- Bianchetti and Carlicchi (2011) provides a detailed examination of the transition from single-curve to multiple-curve frameworks. Their work focuses on the implications for derivative pricing and how the SABR model is adapted for volatility calibration in a multi-curve environment.

- The Bank for International Settlements released a study analyzing yield curve dynamics post-crisis, offering insights into the influence of central bank policies on curve construction practices, highlighting the transition from single to multi-curve frameworks (BIS, 2013).

- The widely recognized industry white paper Numerix (2013) emphasizes the complexities involved in calibrating these curves and underscores the importance of sophisticated quantitative techniques to capture market conditions and effectively manage risk.

3 Tools, Methodology, and Key Definitions

3.1 Introduction to Computational Tools

This section presents the key computational tools and libraries used in this study. The choice of QuantLib, Pandas, NumPy, and Matplotlib is based on their widespread adoption and reliability within the quantitative finance community. Each tool plays a specific role in the workflow, from data processing and financial modeling to result visualization.

3.1.1 QuantLib

QuantLib is an open-source library for quantitative finance that offers an extensive collection of functions to price, model, and analyze multiple asset classes. In this thesis, QuantLib is employed to construct yield curves, allowing for the implementation of various market instruments such as OISs, swaps, and FRAs.
Much of the coding methodology in this study follows the practices detailed in Ballabio and Balaraman (2023), which provides comprehensive guidance on using QuantLib for financial modeling.

3.1.2 Pandas

Pandas is a versatile open-source library for data analysis and manipulation. In this work, Pandas is used to preprocess the financial data, such as importing datasets in formats like CSV and Excel, cleaning missing values, and calculating mid-rates from bid and ask prices. Additionally, Pandas is used to create data structures compatible with other libraries like

NumPy and Matplotlib.

3.1.3 NumPy

NumPy is a fundamental package for scientific computing in Python, providing support for large multi-dimensional arrays and matrices. In this study, NumPy's statistical functions assist in performing calculations for the final results.

3.1.4 Matplotlib

Matplotlib is a powerful plotting library in Python. This thesis uses Matplotlib to generate visualizations of the yield curves, illustrating their evolution over time and enabling comparisons between different curve construction techniques.

3.2 Data Preparation Methodology

The financial data used in this study is sourced from Refinitiv Eikon, a leading provider of market data. We utilized trading data snapshots containing bid and ask quotes for selected timeframes of the relevant assets.
The data preparation process begins with importing the raw data from Refinitiv into Excel for initial formatting. Next, the formatted data is imported into the Python environment using the Pandas library. Subsequently, mid-rates are calculated as the average of bid and ask prices, and maturities are converted into "periods" to comply with QuantLib's formatting standards.

3.3 Yield Curve Construction Methodology in QuantLib

The curve construction methodology in QuantLib follows these key steps:

1. **Instrument selection:** market instruments are chosen based on their "tenor consistency" with the yield curve's maturity profile.

2. **Rate Helpers:** QuantLib's rate helpers, such as `DepositRateHelper`, `FraRateHelper`, `FuturesRateHelper`, `SwapRateHelper`, and `DatedOISRateHelper`, encapsulate market quotes for different instruments and connect them to the curve construction

process. These functions transform raw market data into inputs that can be used for curve bootstrapping by linking each instrument's market rate to a specific point on the curve.

3. **Bootstrapping:** the `PiecewiseYieldCurve` class in QuantLib sequentially derives zero-coupon rates, with smooth interpolation ensured by the `PiecewiseLogCubicDiscount` function, the primary constructor used in this study.

4. **Validation and visualization:** the constructed curves are validated against market data and visualized to assess their accuracy and responsiveness to market conditions.

All the curves in this study are evaluated on July 12, 2024

3.4 Some Definitions

3.4.1 Zero Curve

The zero curve represents the series of zero-coupon interest rates for different maturities. A zero-coupon rate $R(0, t_i)$ is the yield on an instrument that pays no intermediate interest (coupons) and is redeemed at par at maturity t_i. The zero curve is fundamental for discounting future cash flows to their present value.

3.4.2 Forward Curve

The forward curve represents the market's expectations of future interest rates over specific periods. A forward rate $R(t_i, t_{i+1})$ is the interest rate agreed upon today for a loan that will occur in the future between times t_i and t_{i+1}. The forward curve is essential for pricing instruments that depend on future interest rates, such as forward rate agreements, futures, and interest rate swaps.

3.4.3 Conversion Between Rates and Discount Factors

3.4.3.1 From Zero/Forward Rates to Discount Factors

Given a zero or forward rate $R(t_i, t_{i+1})$ over the period $[t_i, t_{i+1}]$ with year fraction $\tau(t_i, t_{i+1})$, the discount factor $D(t_{i+1})$ is calculated according to the compounding convention. For zero rates, since $t_i = 0$, we have $D(t_i) = D(0) = 1$.

- Continuous compounding:

$$D(t_{i+1}) = D(t_i) \cdot e^{-R(t_i,t_{i+1})\cdot\tau(t_i,t_{i+1})}$$

- Simple compounding:

$$D(t_{i+1}) = D(t_i) \cdot \frac{1}{1 + R(t_i, t_{i+1}) \cdot \tau(t_i, t_{i+1})}$$

3.4.3.2 From Discount Factors to Zero/Forward Rates

Based on the previous equations, it can be shown that following the compounding convention:

- Continuous compounding:

$$R(t_i, t_{i+1}) = -\frac{1}{\tau(t_i, t_{i+1})} \ln\left(\frac{D(t_{i+1})}{D(t_i)}\right)$$

- Simple compounding:

$$R(t_i, t_{i+1}) = \frac{1}{\tau(t_i, t_{i+1})} \left(\frac{D(t_i)}{D(t_{i+1})} - 1\right)$$

The specific derivation of these formulas can be found in Veronesi (2010). *All the curves in this study are expressed using continuously compounded rates.*

3.4.4 Single vs Dual Curve Bootstrapping Approach

In yield curve construction, the single-curve approach employs a single curve for both discounting and projecting rates. While this simplifies the valuation process, it fails to account for the differences in risk profiles and liquidity premia inherent in collateralized swap transactions as outlined in Johannes and Sundaresan (2007).

Instead, in the dual-curve approach we constructs yield curves using two distinct curves:

- Zero Curve (from OIS/€STR) for discounting cash flows.
- Forward Curve (from Euribor) for projecting future floating rates.

Where:

- The OIS/€STR is an overnight risk-free rate. When constructing the yield curve based on €STR, we produce a zero curve because it reflects the term structure of risk-free rates used for discounting cash flows in a risk-free or collateralized framework.
- Euribor rates are term rates for various maturities (e.g., 1 month, 3 months, 6 months) and include credit and liquidity premiums. The market instruments linked to Euribor (FRAs, futures, swaps) are inherently forward-looking and provide information about future Euribor rates over specific periods.

Constructing a zero curve from Euribor instruments is less practical because Euribor rates are not considered risk-free, and the resulting zero curve would not be suitable for discounting in a risk-free framework. Since these zero curves are derived solely from instruments tied to specific Euribor maturities, they do not fully capture the term structure's dynamics. Consequently, they serve only as intermediary constructs for deriving the corresponding forward curves across different tenors. As such, some Euribor zero curves will be presented only in Chapter 7.1, to illustrate the comparison between the effects of single and dual curve discounting approaches.

4 Market Instruments

This chapter outlines the key financial instruments and benchmarks related to this research. We begin with a description of EURIBOR and €STR, the two most important benchmarks for the Eurozone. Next, we provide an overview of the market instruments used in this study for the construction of the yield curves.

4.1 EURIBOR

The Euro Interbank Offered Rate (EURIBOR) is an essential benchmark in the Eurozone that represents the average interest rate at which key european banks lend unsecured funds to each other in the interbank market. Introduced in December 1998, it is calculated for maturities ranging from one week to twelve months and is commonly used as a benchmark for financial products such as interest rate swaps, loans, and mortgages.
EURIBOR is considered a critical benchmark by the European Commission and supports financial instruments and contracts worth over €100 trillion in total outstanding value. The calculation of EURIBOR fixings is overseen by Global Rate Set Systems Ltd. (GRSS), ensuring the credibility and trustworthiness of the rate (EMMI, 2019a).

4.1.1 EURIBOR Methodology

The robustness and regulatory compliance of EURIBOR are secured through a hybrid methodology, which employs a bottom-up "waterfall" process for contributions from panel banks calibrated on transactions in the unsecured euro money market. The method consists of three levels:

- **Level 1:** contributions based on eligible unsecured euro money market transactions with a minimum notional amount of €10 million (reduced from €20 million in 2021). The contribution rate is the volume-weighted average rate of these transactions by tenor.

- **Level 2:** contributions derived from transactions across broader maturities or recent similar transactions when Level 1 inputs are not available. Methods include modified linear interpolation and the use of historical Level 1 data.
- **Level 3:** transaction-based contributions from markets highly related to the unsecured euro money market, including data below the threshold of €10 million. The European Money Market Institute (EMMI) regulates the documentation, validation, and application of all contributions by panel banks in a uniform manner.

This structure ensures that EURIBOR remains accurate and representative, with annual reviews for continuous alignment with market conditions (EMMI, 2019b).

4.1.2 Transition from LIBOR to EURIBOR

The transition from LIBOR to EURIBOR as the main benchmark for euro-denominated financial products was driven by global reform efforts that began in 2012, following manipulation scandals associated with LIBOR itself. The rate, published for the first time in 1986, was originally calculated based on daily surveys conducted by the British Bankers' Association (BBA), where a panel of major banks estimated the interest rates at which they could borrow funds in the London money markets. The highest and lowest submissions were discarded, and the average of the remaining middle values was reported as the LIBOR rate for various currencies and maturities. However, this method relied heavily on the banks' self-reported estimates rather than actual transaction data, creating opportunities for manipulation. In response to these shortcomings, the European Union's Benchmark Regulation (BMR) introduced stricter requirements, mandating that benchmarks be derived from actual transaction data to ensure greater transparency and reliability. As a result, euro-area financial products gradually requireded from LIBOR to EURIBOR, which is based on real market transactions and thus offers a more accurate reflection of prevailing interest rates (FSB, 2014).

4.2 €STR

Introduced by the ECB in October 2019, €STR is the new transaction-based benchmark rate for wholesale overnight unsecured borrowing in the eurozone. It reflects the rate at which euro-area banks are willing to borrow unsecured overnight funds from other financial institutions. In contrast to its predecessor, EONIA, €STR is based on a wider range of transactions, making it more stable and resilient as a reference interest rate (ECB, 2022).

4.2.1 €STR Methodology

€STR is calculated solely from borrowing transactions denominated in euros exchanged with financial counterparts reported by banks in compliance with Regulation (EU) No 1333/2014 (MMSR Regulation). Among the various instrument categories governed by the MMSR, €STR is based on overnight unsecured fixed-rate deposit transactions exceeding €1 million. These unsecured deposits are standardized and represent the most common method for executing arm's-length transactions through a competitive process, thus reducing the impact of individual factors that may influence rate volatility.
The calculation is conducted daily for each calendar TARGET2 day, utilizing a volume-weighted trimmed mean rounded to the third decimal place. The volume-weighted trimmed mean is computed as follows:

1. **Sorting:** transactions are sorted from the lowest rate to the highest rate.
2. **Summation:** summing is performed at each rate level.
3. **Exclusion:** the top and bottom 25% in volume terms are excluded.
4. **Averaging:** the remaining 50% of the volume-weighted distribution is averaged.

A pro rata calculation is implemented for volumes straddling the trimming thresholds to ensure that exactly 50% of the total eligible volume contributes to the calculation of the volume-weighted mean (ECB, 2019).

4.2.2 The Transition from EONIA to €STR

EONIA, published since January 4, 1999, and officially discontinued on January 3, 2022, was historically calculated as the weighted average of interest rates on overnight unsecured lending transactions conducted in the interbank market within the Eurozone. A selected panel of european banks reported the interest rates applied in their daily transactions, and these rates were weighted by transaction volumes.
However, several factors supported the transition from EONIA to €STR:

- **Implementation of benchmark regulations:** the BMR has established stricter guidelines for benchmark administrators, increasing the required accountability and integrity in setting rates such as EONIA.
- **Broadened market coverage:** €STR represents a broader set of institutions and transactions, providing a more accurate and resilient reference rate for the Eurozone money market.
- **Lower risk of manipulation:** with a significantly larger number of underlying transactions and without a high concentration of volumes by only a few contributors, €STR is less susceptible to manipulation, a primary issue with EONIA and panel-based rates.

4.3 Overview of Market Instruments Used

In this study, we will construct yield curves using specific sets of financial instruments, each crucial for capturing market expectations and risks within the multiple-curve environment. The theoretical framework and exact pricing formulas of these instruments are beyond the scope of this paper, as they have been extensively covered in prior research. Readers are referred to the seminal work by Ametrano and Bianchetti (2013), where the implications of pricing in a collateralized environment are thoroughly explored. Below is a description of each instrument, along with the corresponding Refinitiv Eikon RIC codes used in this research.

4.3.1 Deposits

Deposits are short-term loans provided by one bank to another, typically ranging from overnight up to ten years. These unsecured transactions are a fundamental component of the money market and play a crucial role in constructing the shortest segment of the €STR curve.
RIC Code: `EURDEPO=`

4.3.2 Synthetic Deposits

Synthetic deposits are artificial instruments derived from the EURIBOR-OIS basis spread, which will be used to construct the initial months of our Euribor curves. Their necessity arises from the challenge of finding the required instruments to build the first segment of the curve directly in the market. Due to their unique nature, they will be the only financial instruments explored in greater depth in Chapter 6.

4.3.3 Overnight Index Swaps (OIS)

OISs are interest rate swaps in which one party pays a fixed interest rate, while the other pays a floating rate indexed to an overnight rate, such as €STR. Due to their overnight nature, OISs are considered low-risk instruments, making them essential for constructing discount curves in the yield curve construction process.
RIC Code: `EURESTOIS=`

4.3.4 Interest Rate Swaps (IRS)

IRS are derivatives in which two parties exchange cash flows based on different interest rates. Typically, one party pays a fixed rate while the other pays a floating rate indexed to a benchmark such as EURIBOR. Swaps are fundamental for building the medium- to long-term segments of our curves, as they are quoted in the market for maturities of up to 50 years.
RIC Codes:

- `EURAB6EIRS=` (Euro Annual Bond 6 Month Euribor IRS)
- `EURAB3EIRS=` (Euro Annual Bond 3 Month Euribor IRS)

4.3.5 Forward Rate Agreements (FRA)

Forward Rate Agreements (FRAs) are contracts that determine the interest rate to be paid or received on an obligation starting at a future date. FRAs are commonly used to hedge interest rate risk and speculate on future interest rate movements. In yield curve construction, FRAs provide critical information for building forward curves for short- to medium-term maturities.
RIC Codes:

- `EUR0X6F=TPEU, EUR6MFRA=` (Euro 6 Month FRA)
- `EUR0X3F=TPEU, EUR3MFRA=` (Euro 3 Month FRA)

4.3.6 Futures

Futures contracts are standardized agreements to buy or sell an underlying asset at a predetermined future date and price. In the context of interest rate futures, such as the Euribor 3M futures contract, these instruments are critical for hedging and speculating on future movements in interest rates. For building the medium-term segment of our Euribor curves, we will use Euribor 3M Futures.
In this study, we omit the calculation of convexity adjustment, which accounts for the non-linearity of interest rate changes over time in the compensation of the posted collateral. Although the convexity adjustment can influence the accuracy of forward rates derived from futures, it is considered negligible in this analysis, as we are dealing with them only for short-term maturities. As shown in Burgess (2019), the convexity effect is minimal in this context and can therefore be safely disregarded.
RIC Code: `0#FEI:` (Euribor 3M Future)

4.3.7 ECB forward OISs

ECB forward OISs are financial instruments designed to address the interest rate risks associated with the ECB reserve maintenance periods. These periods are predefined intervals during which banks are required to hold a specified average level of reserves within the central bank. As depicted in Table 4.1, the maintenance periods typically start shortly after a monetary policy meeting, allowing any changes in interest rates decided by the ECB to be effective for the entire duration of the period. During these intervals, short-term money markets often fluctuate due to changes in liquidity demand, for reasons that will be further explored in Chapter 5.2. ECB forward OIS enable market participants to hedge against this uncertainty by locking in a fixed interest rate for future maintenance periods (ECB, 2016).

RIC Code: `EUESTECBF=ICAP`

As shown in Figure 4.1, the €STR remains stable over time, with substantial variations occurring only as a result of monetary policy decisions taken during the ECB meetings.

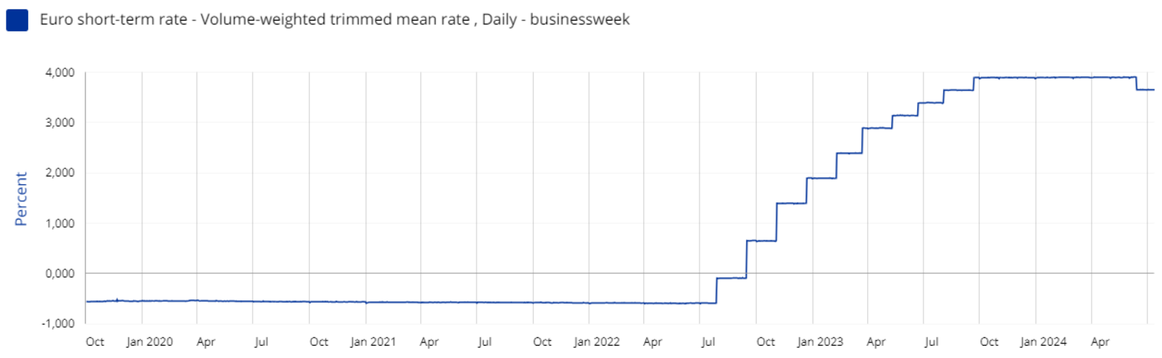

Figure 4.1: Historical series of the continuously compounded €STR from its first publication on October 1, 2019, to July 12, 2024. Source: ECB (2024).

MP	Relevant Governing Council meeting	Start of maintenance period	End of maintenance period	Length of the maintenance period (days)
8/2023	14 December 2023	20 December 2023	30 January 2024	42
1	25 January 2024	31 January 2024	12 March 2024	42
2	7 March 2024	13 March 2024	16 April 2024	35
3	11 April 2024	17 April 2024	11 June 2024	56
4	6 June 2024	12 June 2024	23 July 2024	42
5	18 July 2024	24 July 2024	17 September 2024	56
6	12 September 2024	18 September 2024	22 October 2024	35
7	17 October 2024	23 October 2024	17 December 2024	56
8	12 December 2024	18 December 2024	tbd	tbd

Table 4.1: Schedule of ECB Governing Council Meetings and Reserve Maintenance Periods for 2023-2024. Source: ECB (2023).

5 Construction of the €STR Discount Curve

5.1 Data Selection and Curve Building Methodology

This section presents the methodology followed for selecting market instruments and constructing the €STR discount curve from scratch. The curve is built using a hierarchical piecewise bootstrapping procedure that employs a combination of EUR Deposits, OISs, and ECB forward OIS rates summarized in Table 5.1.
The process is implemented through the use of various rate helpers, as defined in Chapter 3.3, which are instrumental in building different segments of the curve. Each helper serves as a building block for accurately fitting market data across different maturities. The procedure is detailed as follows:

1. **EUR deposits:** the initial segment of the curve is anchored using short-term EUR deposit rates. For this, `DepositRateHelper` is employed to incorporate the market data for the first three maturities: Overnight ('ON'), Tomorrow Next ('TN'), and Spot Next ('SN'). These helpers convert the deposit quotes into the necessary inputs for the curve construction, setting the foundation for the curve's short-term segment.

2. **OISs (first segment):** the curve is then further extended into the short-term horizon using the `OISRateHelper`. At this stage, only one OIS maturing on July 23rd, 2024, is included to avoid overlap with subsequent instruments.

3. **ECB forward OISs:** for capturing expectations related to the ECB's maintenance periods, the `DatedOISRateHelper` is utilized. This helper allows the curve to reflect

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid (%)**	**Ask (%)**	**Mid (%)**
Deposit	ON	July 12th, 2024	July 15th, 2024	3.660%	3.780%	3.720%
Deposit	TN	July 15th, 2024	July 16th, 2024	3.550%	3.900%	3.725%
Deposit	SN	July 16th, 2024	July 17th, 2024	3.550%	3.900%	3.725%
OIS	SW	July 16th, 2024	July 23rd, 2024	3.662%	3.666%	3.664%
Forward OIS ECB	Dated	July 24th, 2024	September 18th, 2024	3.637%	3.687%	3.662%
Forward OIS ECB	Dated	September 18th, 2024	October 23rd, 2024	3.426%	3.476%	3.451%
Forward OIS ECB	Dated	October 23rd, 2024	December 18th, 2024	3.355%	3.405%	3.380%
OIS	6M	July 16th, 2024	January 16th, 2025	3.467%	3.508%	3.488%
OIS	7M	July 16th, 2024	February 17th, 2025	3.429%	3.470%	3.450%
OIS	8M	July 16th, 2024	March 17th, 2025	3.396%	3.437%	3.416%
OIS	9M	July 16th, 2024	April 16th, 2025	3.355%	3.396%	3.375%
OIS	10M	July 16th, 2024	May 16th, 2025	3.317%	3.358%	3.338%
OIS	11M	July 16th, 2024	June 16th, 2025	3.282%	3.323%	3.302%
OIS	1Y	July 16th, 2024	July 16th, 2025	3.244%	3.285%	3.264%
OIS	15M	July 16th, 2024	October 16th, 2025	3.123%	3.164%	3.144%
OIS	18M	July 16th, 2024	January 16th, 2026	3.027%	3.068%	3.048%
OIS	21M	July 16th, 2024	April 16th, 2026	2.945%	2.995%	2.970%
OIS	2Y	July 16th, 2024	July 16th, 2026	2.887%	2.928%	2.907%
OIS	3Y	July 16th, 2024	July 16th, 2027	2.716%	2.757%	2.737%
OIS	4Y	July 16th, 2024	July 17th, 2028	2.625%	2.666%	2.646%
OIS	5Y	July 16th, 2024	July 16th, 2029	2.578%	2.619%	2.598%
OIS	6Y	July 16th, 2024	July 16th, 2030	2.558%	2.599%	2.578%
OIS	7Y	July 16th, 2024	July 16th, 2031	2.551%	2.592%	2.572%
OIS	8Y	July 16th, 2024	July 16th, 2032	2.554%	2.595%	2.574%
OIS	9Y	July 16th, 2024	July 18th, 2033	2.567%	2.608%	2.588%
OIS	10Y	July 16th, 2024	July 17th, 2034	2.583%	2.624%	2.604%
OIS	11Y	July 16th, 2024	July 16th, 2035	2.600%	2.641%	2.620%
OIS	12Y	July 16th, 2024	July 16th, 2036	2.619%	2.660%	2.640%
OIS	15Y	July 16th, 2024	July 18th, 2039	2.657%	2.698%	2.678%
OIS	20Y	July 16th, 2024	July 18th, 2044	2.627%	2.668%	2.648%
OIS	25Y	July 16th, 2024	July 16th, 2049	2.545%	2.586%	2.566%
OIS	30Y	July 16th, 2024	July 16th, 2054	2.464%	2.505%	2.484%
OIS	40Y	July 16th, 2024	July 16th, 2064	2.326%	2.376%	2.351%
OIS	50Y	July 16th, 2024	July 16th, 2074	2.217%	2.267%	2.242%
OIS	60Y	July 16th, 2024	July 17th, 2084	2.141%	2.191%	2.166%

Table 5.1: Market instruments for the €STR discount curve construction

the market conditions during these periods, providing a more accurate representation of forward expectations. For instance, the forward OIS starting on September 18th, 2024, and maturing on October 23rd, 2024, is added using the `DatedOISRateHelper`, which ensures that the forward rates are accurately incorporated into the curve during this period.

4. **OISs (second segment):** finally, to extend the curve into longer maturities up to a 50-year horizon, additional `OISRateHelper` instruments are introduced. These helpers enrich the curve with long-term data, ensuring that the constructed curve accurately represents the entire spectrum of maturities, from short to long term.

The curve is then built using QuantLib's `PiecewiseLogCubicDiscount` constructor. This ensures the curve remains smooth and continuous, with minimal jumps or inconsistencies across different maturities. The use of log-cubic interpolation guarantees that the curve smoothly transitions across different maturities, providing a precise representation of the underlying market data.
This study presents a point-in-time snapshot of the market, with curves fitted to data available on a specific date. However, these curves need constant recalibration as market data changes, which is why relying solely on fixings, the official benchmark rates published just once per day, is insufficient.

5.1.1 Plot of the Constructed Curve

Below, we present the €STR zero curve built using this procedure. On the x-axis, we have dates (maturities), while on the y-axis, we see the zero rates corresponding to each maturity.

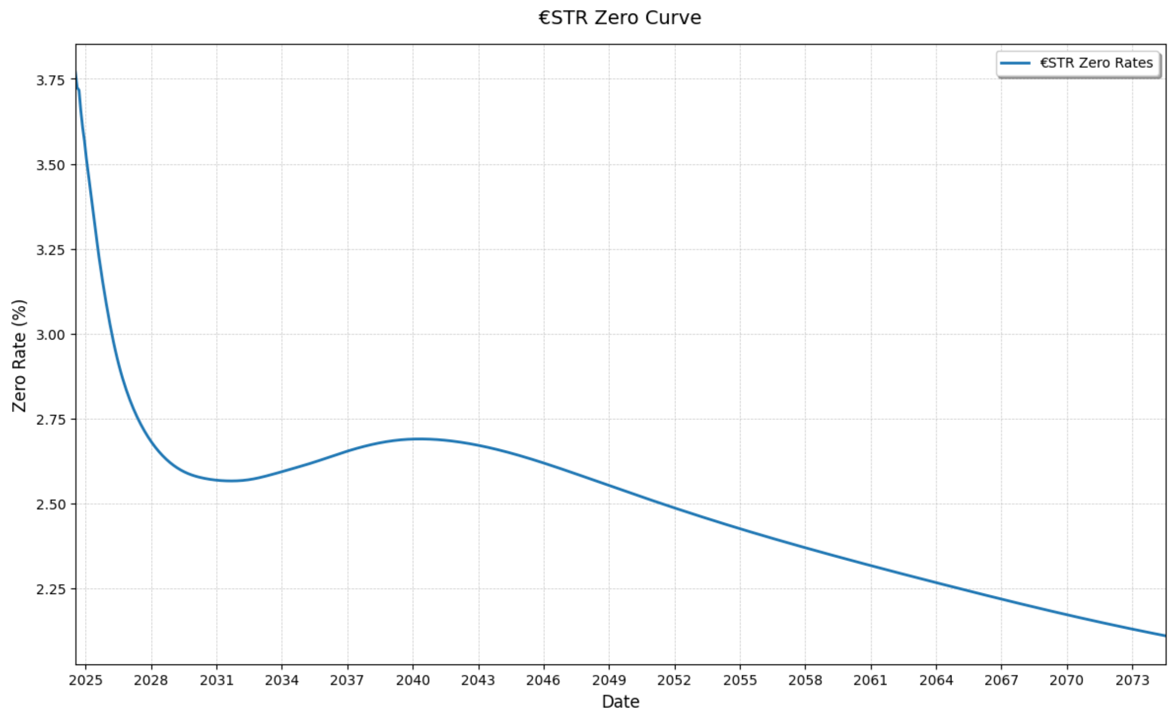

Figure 5.1: The €STR Zero Curve constructed using a piecewise bootstrapping methodology with the QuantLib `PiecewiseLogCubicDiscount` function. The curve represents the continuously compounded zero rates across different maturities, from 2024 to 2084, based on market data as of July 12th, 2024. The initial segment is anchored by short-term EUR deposit rates (ON, TN, SN), while the intermediate and long-term segments are constructed usin OIS and ECB forward OIS rates.

The smooth transition between different maturities indicates the effectiveness of the piecewise construction methodology.

5.2 Analysis of the Short-Term Segment of the Curve

In this section, we explore the initial segment of the curve to identify any abnormal fluctuations, such as potential "liquidity jumps." Our focus is on the dynamic behavior of forward rates over the first 12 months, with special attention to the ECB's reserve maintenance periods.

5.2.1 ECB Reserve Maintenance Periods and Liquidity Jumps

As introduced in section 4.3.7, the ECB reserve maintenance period is a predefined interval during which eurozone banks are required to maintain a specific average amount of reserves in their current accounts at their national central banks (NCBs) within the Eurosystem. These periods begin shortly after a monetary policy meeting, ensuring that any changes in policy rates are fully incorporated (ECB, 2016).

Since banks are required to meet these reserve requirements on an average basis, they have flexibility in daily liquidity management. This flexibility allows banks to adjust their liquidity positions in response to market conditions, which can lead to fluctuations in short-term interest rates. Particularly towards the end of a maintenance period, banks may engage in adjustments to meet their reserve requirements, potentially causing anomalous volatility in the money markets. This behavior can result in "liquidity jumps," which are abrupt changes in short-term rates due to shifts in supply and demand for liquidity.

Another example of such jumps is the "Turn-of-the-Year" effect, often occurring at the end of the calendar year. Empirical research by Ametrano and Bianchetti (2013) suggests that this phenomenon is linked to anticipatory adjustments for regulatory reporting of year-end balance sheets.

5.2.2 Methodology and Results

Using the €STR discount curve constructed in section 5.1, we generated overnight forward rates for the first 12 months using the *PiecewiseFlatForward* methodology. As shown in Ametrano and Bianchetti (2013), this method is appropriate for capturing gradual changes in market expectations without excessive smoothing.

Our results, illustrated in Figure 5.2, indicate that the overnight forward rates exhibit a consistent, stepwise decline over the 12-month period. This means that the rates decrease incrementally in small, predictable steps, rather than undergoing abrupt upward shifts driven by heightened liquidity demands, as discussed in the previous section. Such behavior contrasts with previous studies that observed jumps in short-term rates during the end of the year or in line with the end of ECB reserves maintenance periods as in Lang and Gehrig (2007). The absence of such jumps in our analysis may indicate a more stable liquidity environment or a different market adaptation to the ECB's liquidity management strategy since the introduction of €STR.

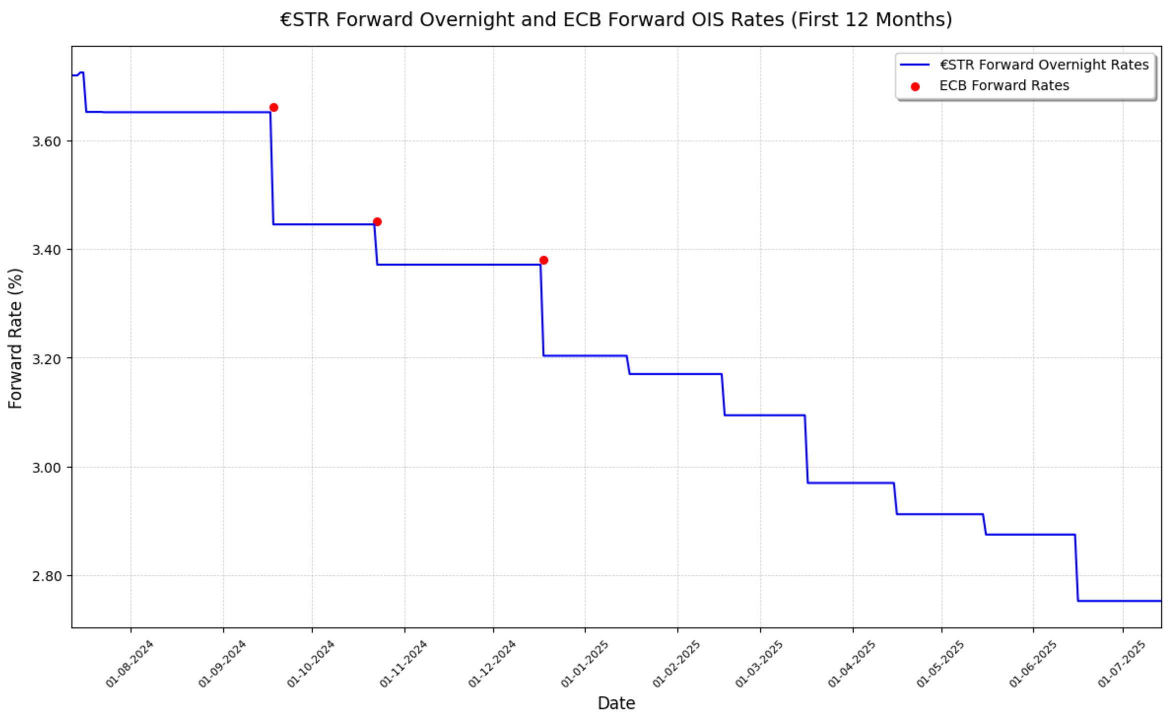

Figure 5.2: The chart illustrates the €STR overnight forward rates and the ECB forward OIS rates for the first 12 months from July 12, 2024. The blue line shows the overnight forward rates, derived from the discount curve using the bootstrapping methodology outlined in this section. The red dots represent the ECB forward OIS rates, plotted at the end of each reserve maintenance period to which they refer, one day before the start of the subsequent one.

5.2.3 Considerations Regarding the Liquidity Jump Hypothesis

While the hypothesis of liquidity jumps during ECB reserve maintenance periods and at the turn of the year is supported by various studies, our results suggest a more nuanced interpretation. The use of tailored market instruments, such as the ECB forward OISs, may be sufficient to mitigate these jumps.

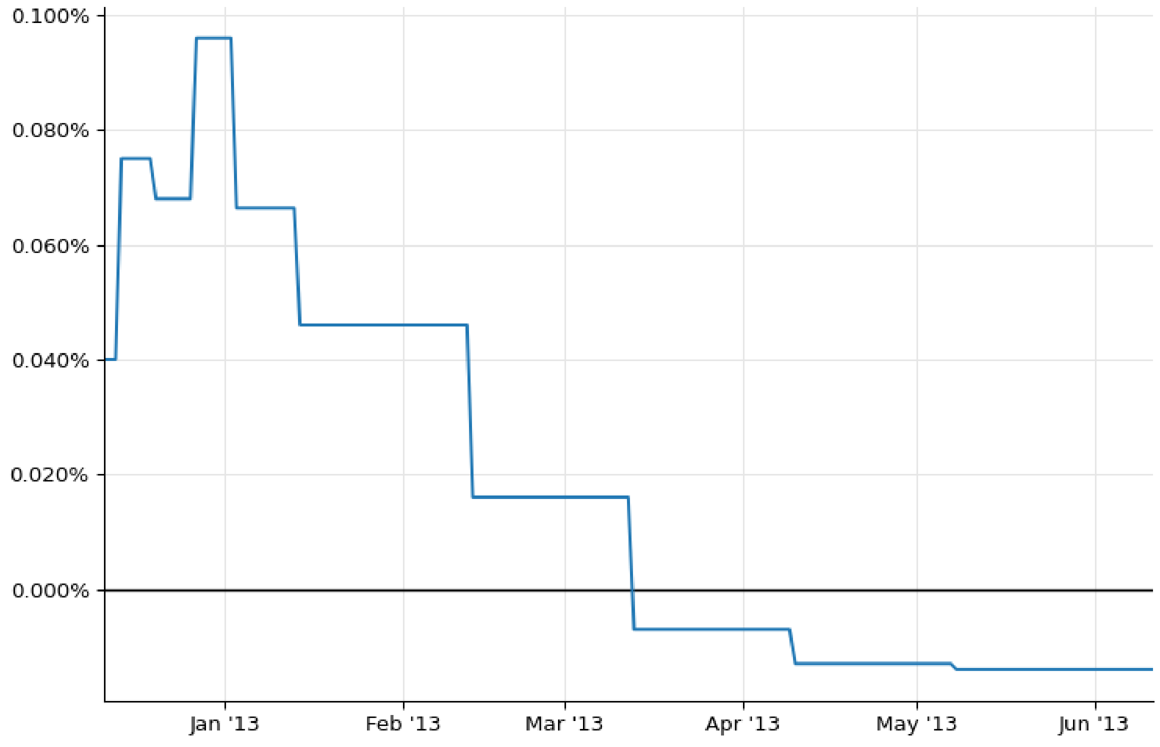

Figure 5.3: The Turn-of-the-Year Jump effect in the EONIA overnight forward rates, reproduced using QuantLib in the context of Ametrano and Bianchetti's research on multiple curve construction in Ballabio and Balaraman (2023). The graph represents the forward rates observed in late 2012 and early 2013, constructed with the same methodology applied to the €STR in Figure 5.2, showing a clear upward spike around the year-end.

The differences in our findings compared to earlier literature on the turn-of-the-year jump, as depicted in Figure 5.3, might be due to variations in market conditions, the structure of the banking sector, or changes in monetary policy implementation during the observed period. While a deeper investigation into this phenomenon is beyond the scope of our current study, it remains an interesting area for future research to understand these discrepancies more comprehensively.

6 Construction and Analysis of the Euribor 3M Curve

6.1 Introduction

In this chapter, we describe the construction of the Euribor 3M forward curves for two different cases: with and without the incorporation of synthetic deposits introduced in Section 4.3.2. As with the €STR curve in the previous chapter, the calibration process starts with the selection of an appropriate set of market instruments to accurately represent market conditions across the entire curve.
Table 6.1 provides a summary of the market instruments used to bootstrap both versions of the curve.

6.2 Curve Construction Without Synthetic Deposits

For the scenario without synthetic deposits, the following market instruments were adopted:

- **FRA:** the 0X3 FRA starting on July 16, 2024, and maturing on October 16, 2024 (0-3 months), used as the initial anchor point for the curve.
- **Futures:** Euribor 3M futures maturing between December 2024 and June 2026 (3 months - 2 years) to extend the curve into the medium-term horizon.
- **Swaps:** Euribor 3M swaps maturing between July 2027 and July 2074 (3-50 years) to extend the curve into the long-term horizon.

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid**	**Ask**	**Mid**
Synthetic Deposit	*ON*	*July 16th, 2024*	*July 17th, 2024*			*3.800%*
Synthetic Deposit	*1W*	*July 16th, 2024*	*July 23rd, 2024*			*3.719%*
Synthetic Deposit	*2W*	*July 16th, 2024*	*July 30th, 2024*			*3.712%*
Synthetic Deposit	*3W*	*July 16th, 2024*	*August 6th, 2024*			*3.719%*
Synthetic Deposit	*1M*	*July 16th, 2024*	*August 16th, 2024*			*3.730%*
Synthetic Deposit	*2M*	*July 16th, 2024*	*September 16th, 2024*			*3.723%*
FRA	0X3	July 16th, 2024	October 16th, 2024	3.654%	3.674%	3.664%
Futures	DEC4	December 18th, 2024	March 18th, 2025	€96.74	€96.745	€96.742
Futures	MAR5	March 19th, 2025	June 19th, 2025	€96.97	€96.975	€96.972
Futures	JUN5	June 18th, 2025	September 18th, 2025	€97.165	€97.17	€97.168
Futures	SEP5	September 17th, 2025	December 17th, 2025	€97.31	€97.315	€97.312
Futures	DEC5	December 17th, 2025	March 17th, 2026	€97.41	€97.415	€97.412
Futures	MAR6	March 18th, 2026	June 18th, 2026	€97.465	€97.47	€97.468
Swap	3Y	July 16th, 2024	July 16th, 2027	2.856%	2.897%	2.876%
Swap	4Y	July 16th, 2024	July 17th, 2028	2.778%	2.782%	2.780%
Swap	5Y	July 16th, 2024	July 16th, 2029	2.728%	2.732%	2.730%
Swap	6Y	July 16th, 2024	July 16th, 2030	2.704%	2.708%	2.706%
Swap	7Y	July 16th, 2024	July 16th, 2031	2.693%	2.697%	2.695%
Swap	8Y	July 16th, 2024	July 16th, 2032	2.695%	2.699%	2.697%
Swap	9Y	July 16th, 2024	July 18th, 2033	2.705%	2.709%	2.707%
Swap	10Y	July 16th, 2024	July 17th, 2034	2.718%	2.722%	2.720%
Swap	12Y	July 16th, 2024	July 16th, 2036	2.739%	2.780%	2.760%
Swap	15Y	July 16th, 2024	July 18th, 2039	2.768%	2.809%	2.788%
Swap	20Y	July 16th, 2024	July 18th, 2044	2.726%	2.767%	2.746%
Swap	25Y	July 16th, 2024	July 16th, 2049	2.630%	2.671%	2.650%
Swap	30Y	July 16th, 2024	July 16th, 2054	2.535%	2.576%	2.556%
Swap	40Y	July 16th, 2024	July 16th, 2064	2.382%	2.423%	2.402%
Swap	50Y	July 16th, 2024	July 16th, 2074	2.240%	2.281%	2.261%

Table 6.1: Market instruments for the Euribor 3M curve construction

The specific maturities for each market instrument were selected based on liquidity and data availability to ensure the constructed curve, plotted in Figure 6.1, accurately reflects current market conditions.

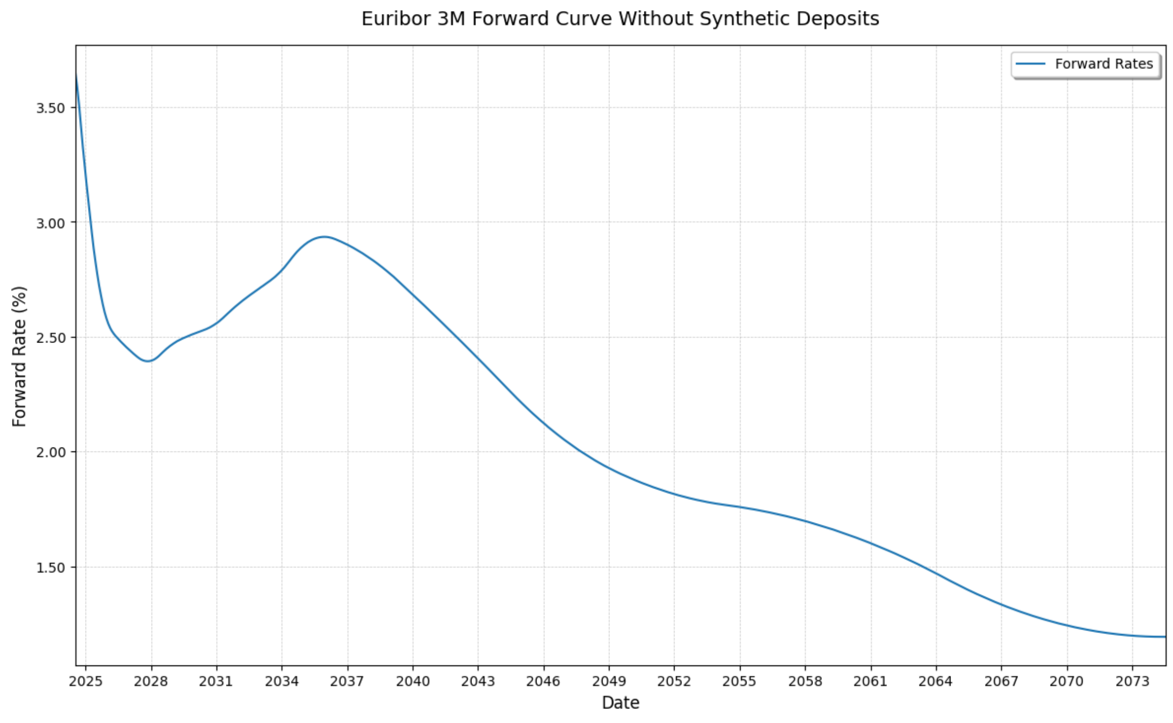

Figure 6.1: The Euribor 3M continuously compounded forward curve constructed without the inclusion of synthetic deposits. The curve shows the forward rates from July 12, 2024 to 2084 based on a bootstrapping procedure that solely utilizes market instruments such as FRAs, Euribor futures, and swaps. The absence of synthetic deposits results in a less accurate representation of forward rate expectations in the initial part of the curve, where backward interpolation is applied.

6.2.1 Limitations and the Need for Synthetic Deposits

Although the curve constructed without synthetic deposits captures the general shape of the yield curve, the lack of synthetic deposits leads to overestimation/underestimations of forward rates in the very short term, causing distortions in the initial segment. This occurs because the first anchor point is the 3-month pillar (the 0X3 FRA), leaving the preceding segment uncovered and reliant solely on backward interpolation.

This limitation in the multiple-curve construction framework arises from the unavailability of market instruments with maturities shorter than three months that are directly tied to the

Euribor 3M rate. The incorporation of synthetic deposits effectively addresses this gap.

6.3 Incorporation of Synthetic Deposits

Synthetic deposits play a crucial role in the construction of the initial segment of the yield curve. As shown in the following pages, these instruments are derived by modeling the basis (spread) between Euribor the OIS/€STR rates. As a result, this approach integrates €STR data into the curve construction process, allowing the presence of additional pillars to calibrate the short-term portion of the curve, enhancing its structure and accuracy.

6.3.1 Modeling the Basis between Euribor and the Discount Curve

As detailed in Ametrano and Bianchetti (2013), we start from the relationship between zero/forward rates and instantaneous forward rates, where $R_{\text{Eu}}(t_1, t_2)$ denotes the Euribor zero/forward rate from t_1 to t_2, and $\tau_{\text{Eu}}(t_1, t_2)$ is the year fraction for Euribor over that period:

$$R_{\text{Eu}}(t_1, t_2) \cdot \tau_{\text{Eu}}(t_1, t_2) = \int_{t_1}^{t_2} f_{\text{Eu}}(u)\, du \tag{6.1}$$

We express the instantaneous Euribor forward rate $f_{\text{Eu}}(t)$ as the sum of the instantaneous OIS/€STR-based forward rate $f_{\text{€STR}}(t)$ and the instantaneous basis $\delta_{\text{Eu}}(t)$, which represents the spread that captures additional credit and liquidity risks embedded in the Euribor rate compared to the risk-free €STR:

$$f_{\text{Eu}}(t) = f_{\text{€STR}}(t) + \delta_{\text{Eu}}(t) \tag{6.2}$$

Substituting Equation (6.2) into Equation (6.1), we obtain:

$$\begin{aligned} R_{\text{Eu}}(t_1, t_2) \cdot \tau_{\text{Eu}}(t_1, t_2) &= \int_{t_1}^{t_2} [f_{\text{€STR}}(u) + \delta_{\text{Eu}}(u)]\, du \\ &= R_{\text{€STR}}(t_1, t_2) \cdot \tau_{\text{€STR}}(t_1, t_2) + \Delta_{\text{Eu}}(t_1, t_2) \end{aligned} \tag{6.3}$$

where:

- $R_{\text{€STR}}(t_1, t_2)$ is the €STR zero/forward rate over the same period;
- $\tau_{\text{€STR}}(t_1, t_2)$ is the year fraction for €STR over the period $[t_1, t_2]$;
- $\Delta_{\text{Eu}}(t_1, t_2) = \int_{t_1}^{t_2} \delta_{\text{Eu}}(u)\, du$ is the cumulative basis between Euribor and €STR rates over the period $[t_1, t_2]$.

Assuming the same money market day-count convention *Actual/360* for $\tau(t_1, t_2)$, valid for both Euribor and OIS/€STR-based rates, we obtain the following relationship:

(6.4) $$R_{\text{Eu}}(t_1, t_2) \cdot \tau(t_1, t_2) = R_{\text{€STR}}(t_1, t_2) \cdot \tau(t_1, t_2) + \Delta(t_1, t_2)$$

The instantaneous basis $\delta(t)$ can be approximated as a polynomial of degree n:

(6.5) $$\delta(t) = \alpha + \beta \cdot t + \gamma \cdot t^2 + \dots$$

If modeled as a constant ($n = 0$) and integrating $\delta(t)$ over the period $[t_1, t_2]$, we obtain a flat cumulative basis:

(6.6) $$\Delta(t_1, t_2) = \int_{t_1}^{t_2} \delta(u)\, du = \alpha \cdot \tau(t_1, t_2)$$

Substituting this expression back into Equation (6.4), we have:

(6.7) $$R_{\text{Eu}}(t_1, t_2) \cdot \tau(t_1, t_2) = R_{\text{€STR}}(t_1, t_2) \cdot \tau(t_1, t_2) + \alpha \cdot \tau(t_1, t_2)$$

Dividing both sides by $\tau(t_1, t_2)$, we find:

(6.8) $$R_{\text{Eu}}(t_1, t_2) = R_{\text{€STR}}(t_1, t_2) + \alpha$$

Thus, α is simply the difference:

$$\alpha = R_{\text{Eu}}(t_1, t_2) - R_{\text{€STR}}(t_1, t_2) \tag{6.9}$$

For a simple constant polynomial, as employed in this work for simplicity, only one market quote is needed to determine α. In this case, we can use the Euribor 0x3 FRA rate, ensuring accurate repricing of the curve. To determine the value of α, or any additional coefficients for higher-degree polynomials ($\beta, \gamma, \ldots$), we solve a system of equations where the number of polynomial coefficients we wish to determine dictates the number of market quotes required. By substituting these market quotes and the corresponding OIS/€STR rates into Equation (6.4), we can solve for the coefficients $\alpha, \beta, \gamma, \ldots$.
Once the basis is determined, we can interpolate/extrapolate it to compute the synthetic deposit rates $R(0, t)$ for any given maturity t.

6.3.2 Comparison With and Without Synthetic Deposits

To assess the impact of including synthetic deposits, we compare both forward Euribor curves over the first six months. The inclusion of synthetic deposits primarily affects the curve's short end, mitigating the distorsion seen without them.
Figure 6.2 shows that incorporating synthetic deposits corrects the overestimation, which reaches up to 4.2 bps without them, leading to a curve that more accurately reflects short-term rate dynamics. As the six-month horizon advances, the two curves gradually converge, minimizing the initial discrepancies.

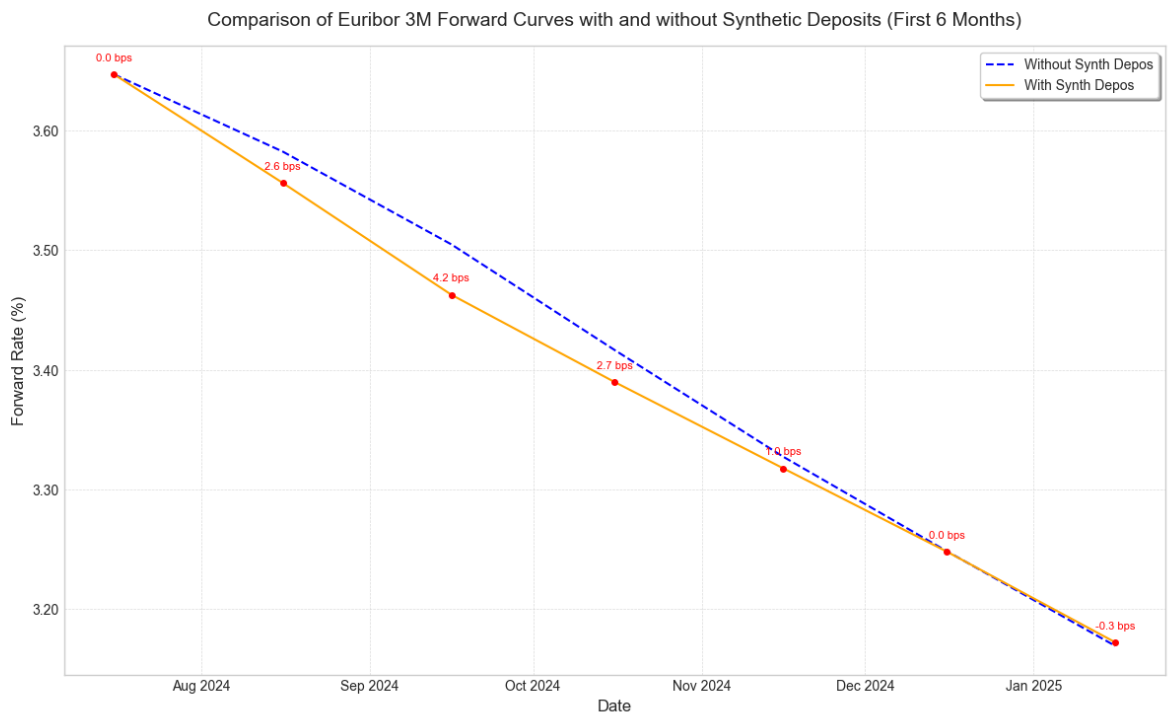

Figure 6.2: Comparison of the Euribor 3M forward curves with and without synthetic deposits over the first 6 Months. The blue dashed line depicts the forward curve constructed without synthetic deposits, while the orange solid line represents the forward curve with synthetic deposits included. Red markers and annotations indicate the basis point (bps) differences at key points along the curves, demonstrating how the non-inclusion of synthetic deposits causes distortion in the estimation of the shortest forward rates.

7 Single vs Dual Curve Bootstrapping: An Empirical Comparison

This chapter presents the empirical comparison between single-curve and double-curve discounting methods described in Chapter 3.4.4. As outlined in Chapter 2.2, the use of a second curve for discounting is central to the multiple curve framework.
The main objective is to demonstrate how using an exogenous discounting rate (from an external, risk-free curve like the OIS curve) in the dual-curve approach influences the construction of the Euribor 3M zero curve and the associated forward curve, compared to using an endogenous rate (internally derived from the same curve used for projection) in the single-curve approach.

7.1 Zero Curves Comparison

Figure 7.1 displays the Euribor 3M zero curves constructed using both the single-curve and dual-curve methodologies. An initial visual inspection indicates that the differences between the two curves are minimal.

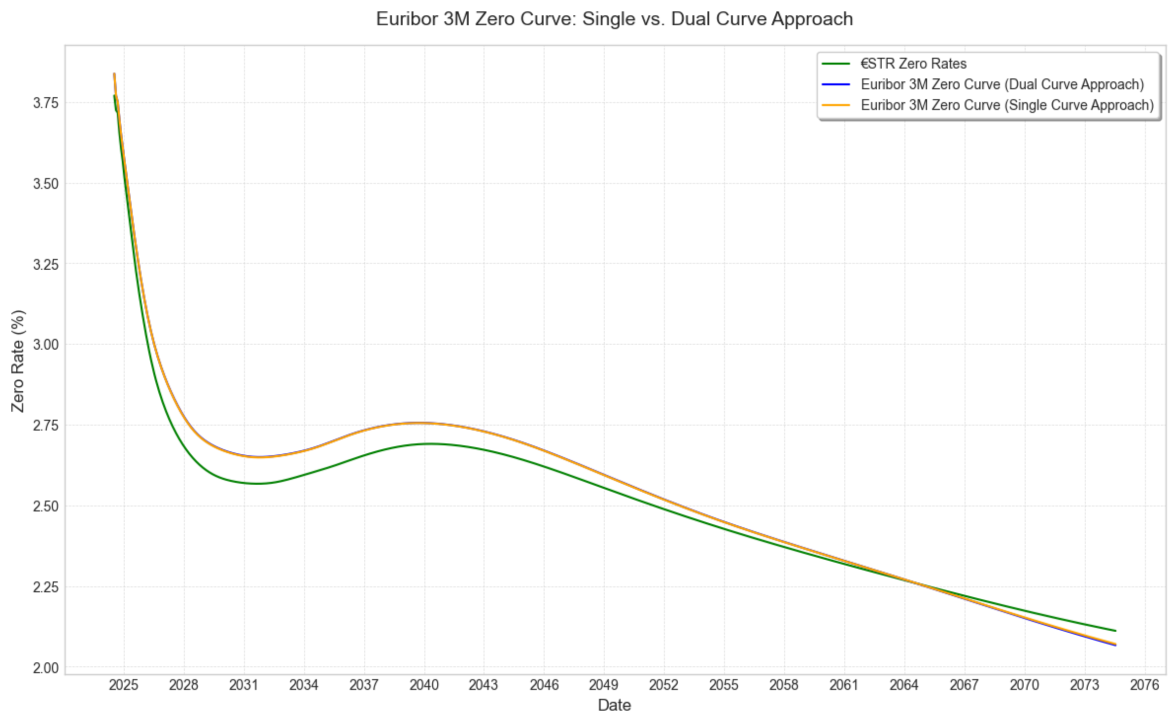

Figure 7.1: Comparison of Euribor 3M and €STR continuously compounded zero curves constructed using single vs dual curve approaches. The green line represents the €STR zero curve, serving as benchmark for comparison. The blue line shows the Euribor 3M zero curve built using the dual-curve approach, where an exogenous discounting rate is applied from the €STR curve. The orange line depicts the Euribor 3M zero curve built with the single-curve approach, where both projection and discounting are based on the Euribor curve itself.

Upon closer examination, two notable observations emerge:

- Both Euribor 3M curves tend to fall below the €STR curve toward the end of the timeline. This is counterintuitive, given that €STR is considered a risk-free benchmark and would typically be lower than an interbank rate like Euribor 3M.
- In the final segment, the Euribor 3M curve constructed using the single-curve approach is marginally higher than the one obtained using the dual-curve method.

7.2 Forward Curves Comparison

Figure 7.2 presents the forward curves derived using both the single-curve and dual-curve methodologies. As is possible to see in Figure 7.2, the differences in forward rates are relatively minor, particularly for shorter maturities where the two curves nearly overlap. The divergence between the two approaches becomes marginally more pronounced over extended horizons, with the single-curve method showing up to a 2.0 basis point difference at the longest maturities.

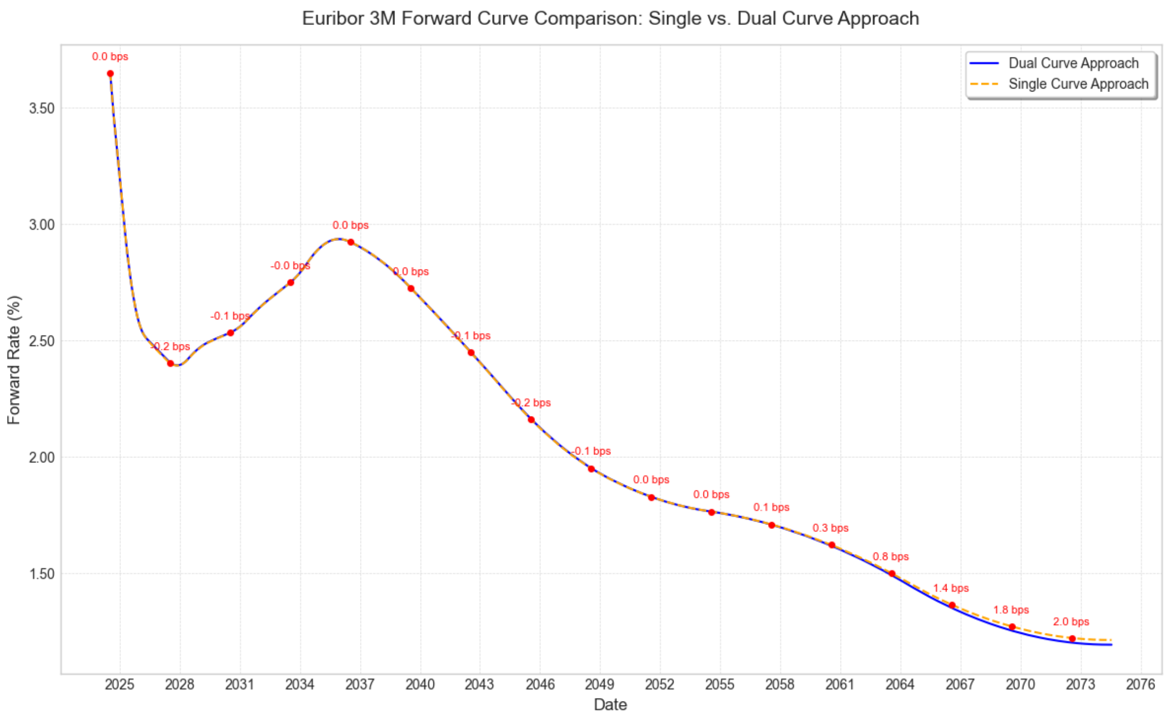

Figure 7.2: Comparison of the Euribor 3M forward curves derived using single vs dual curve approaches over the entire maturity horizon. The blue solid line represents the forward rates generated using the dual-curve methodology, while the orange dashed line shows the rates from the single-curve approach. The red markers and annotations indicate the basis point (bps) differences at key points along the forward curves.

While the differences are subtle in this example, they could become more significant under conditions of elevated market stress, increased volatility, or reduced liquidity. The dual-curve approach is theoretically a more accurate representation of risk premiums, as it allows for the separation of the projection curve (e.g., Euribor 3M) from the discounting curve (e.g., €STR or OIS). However, this example indicates that, in relatively stable market conditions, the impact of using the dual-curve method over the single-curve method may be limited. Further research is needed to determine whether these findings are consistent across different market regimes or under conditions of heightened uncertainty.

8 The Importance of a Tenor-Consistent Bootstrapping Set

In this chapter, we conduct a second empirical experiment to underline the second pillar of the multi-curve construction framework: the importance of selecting a set of market instruments that is consistent with the tenor of the Euribor curve we intend to build. To demonstrate this, we will construct an Euribor 6M forward curve, initially comparing it with the previously constructed 3M forward curve and then with an 'incorrect' 6M one built using a set of instruments with a different tenor (for convenience, we will use the same set used for the 3M curve).

The reason we consider the second Euribor 6M curve 'incorrect' is that instruments aligned with the tenor of the curve (e.g., Euribor 6M FRAs, Euribor 6M swaps) naturally contain the necessary information regarding market conditions, such as maturity-based liquidity and credit risk. This alignment provides the required granularity for accurate calibration of the curve, reducing the likelihood of mispricing financial instruments and ensuring that forward rates correspond to actual market expectations. In contrast, if we use instruments whose tenor does not match that of the curve, distortions can be introduced. For example, an Euribor 6M curve built using a bootstrapping set based on 3-month instruments might not accurately reflect real market conditions. This mismatch can lead to unreliable forward rate estimates and, consequently, incorrect pricing of derivatives.

8.1 Construction of the Euribor 6M forward Curve

The Euribor 6M forward curve is constructed similarly to the 3M one, with the only difference being the set of bootstrapping instruments used for calibration. The market instruments selected, detailed in Table 8.1 below, are as follows:

- **Synthetic deposits (0-6m):** created using the same method followed in Chapter 6 for the 3-month curve. They are fundamental for building the initial segment of the curve.
- **FRAs (6m-2y):** including the 0X6 FRA as the first anchor point, these are essential for extending the curve into the medium-term horizon.
- **Swaps (2y-50y):** these extend the curve further into the long-term horizon.

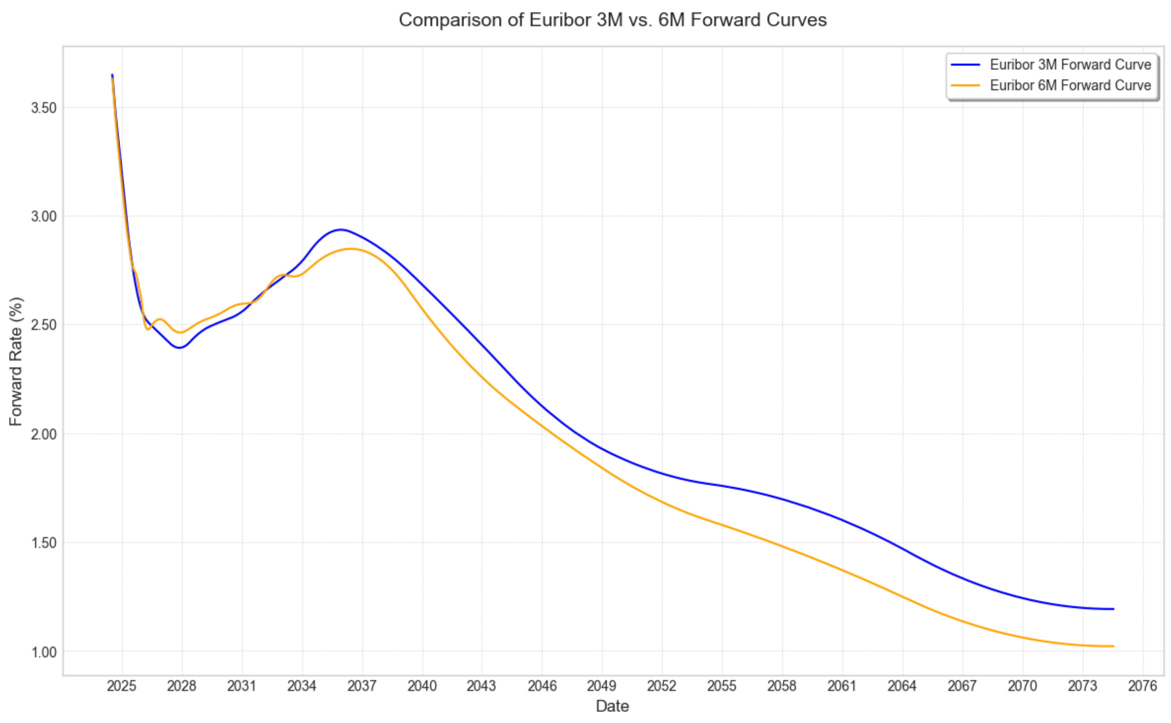

Figure 8.1: Comparison of Euribor 3M and 6M continuously compounded forward curves. The blue line represents the forward curve constructed for Euribor 3M, while the orange line represents the Euribor 6M forward curve.

Key observations on Figure 8.1:

- **Forward rates level:** the 3-month curve (blue) remains predominantly above the 6-month curve (orange) throughout the observed period.

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid**	**Ask**	**Mid**
Synthetic Deposit	*ON*	*July 16th, 2024*	*July 17th, 2024*			*0.039*
Synthetic Deposit	*1W*	*July 16th, 2024*	*July 23rd, 2024*			*0.038*
Synthetic Deposit	*2W*	*July 16th, 2024*	*July 30th, 2024*			*0.038*
Synthetic Deposit	*3W*	*July 16th, 2024*	*August 6th, 2024*			*0.038*
Synthetic Deposit	*1M*	*July 16th, 2024*	*August 16th, 2024*			*0.039*
Synthetic Deposit	*2M*	*July 16th, 2024*	*September 16th, 2024*			*0.038*
Synthetic Deposit	*3M*	*July 16th, 2024*	*October 16th, 2024*			*0.038*
Synthetic Deposit	*4M*	*July 16th, 2024*	*November 18th, 2024*			*0.037*
Synthetic Deposit	*5M*	*July 16th, 2024*	*December 16th, 2024*			*0.037*
FRA	0X6	July 16th, 2024	January 16th, 2025	3.652	3.672	3.662
FRA	1X7	August 16th, 2024	February 17th, 2025	3.553	3.573	3.563
FRA	2X8	September 16th, 2024	March 17th, 2025	3.453	3.473	3.463
FRA	3X9	October 16th, 2024	April 16th, 2025	3.368	3.388	3.378
FRA	4X10	November 18th, 2024	May 16th, 2025	3.281	3.301	3.291
FRA	5X11	December 16th, 2024	June 16th, 2025	3.206	3.226	3.216
FRA	6X12	January 16th, 2025	July 16th, 2025	3.134	3.154	3.144
FRA	7X13	February 17th, 2025	August 18th, 2025	3.059	3.079	3.069
FRA	8X14	March 17th, 2025	September 16th, 2025	2.990	3.010	3.000
FRA	9X15	April 16th, 2025	October 16th, 2025	2.928	2.948	2.938
FRA	10X16	May 16th, 2025	November 17th, 2025	2.874	2.894	2.884
FRA	11X17	June 16th, 2025	December 16th, 2025	2.816	2.836	2.826
FRA	12X18	July 16th, 2025	January 16th, 2026	2.774	2.794	2.784
FRA	18X24	January 16th, 2026	July 16th, 2026	2.577	2.597	2.587
Swap	3Y	July 16th, 2024	July 16th, 2027	2.935	2.945	2.940
Swap	4Y	July 16th, 2024	July 17th, 2028	2.841	2.851	2.846
Swap	5Y	July 16th, 2024	July 16th, 2029	2.787	2.797	2.792
Swap	6Y	July 16th, 2024	July 16th, 2030	2.759	2.769	2.764
Swap	7Y	July 16th, 2024	July 16th, 2031	2.745	2.755	2.750
Swap	8Y	July 16th, 2024	July 16th, 2032	2.739	2.749	2.744
Swap	9Y	July 16th, 2024	July 18th, 2033	2.745	2.755	2.750
Swap	10Y	July 16th, 2024	July 17th, 2034	2.749	2.759	2.754
Swap	12Y	July 16th, 2024	July 16th, 2036	2.769	2.779	2.774
Swap	15Y	July 16th, 2024	July 18th, 2039	2.786	2.796	2.791
Swap	20Y	July 16th, 2024	July 18th, 2044	2.718	2.728	2.723
Swap	25Y	July 16th, 2024	July 16th, 2049	2.6025	2.6325	2.617
Swap	30Y	July 16th, 2024	July 16th, 2054	2.506	2.516	2.511
Swap	40Y	July 16th, 2024	July 16th, 2064	2.326	2.330	2.328
Swap	50Y	July 16th, 2024	July 16th, 2074	2.169	2.173	2.171

Table 8.1: Market instruments for the Euribor 6M curve construction

- **Curve shape:** both curves show a sharp decline in the short term (2025), followed by a moderate increase and a gradual decline.
- **Convergence and divergence:** the curves converge in the very short segment (2025-2026) but diverge as the timeline extends. The 6-month curve remains consistently below the 3-month curve, reflecting the different risk profiles and expectations captured by the tenor-consistent instruments used in their construction.

8.2 Comparison of Correctly and Incorrectly Constructed Euribor 6M forward Curves

As anticipated, in this section, we examine the negative effects of using tenor-inconsistent instruments during bootstrapping.
In Figure 8.2, we compare two Euribor 6M forward curves: one constructed correctly using tenor-consistent instruments (orange line) and another constructed incorrectly using instruments based on the 3M tenor (red dashed line).

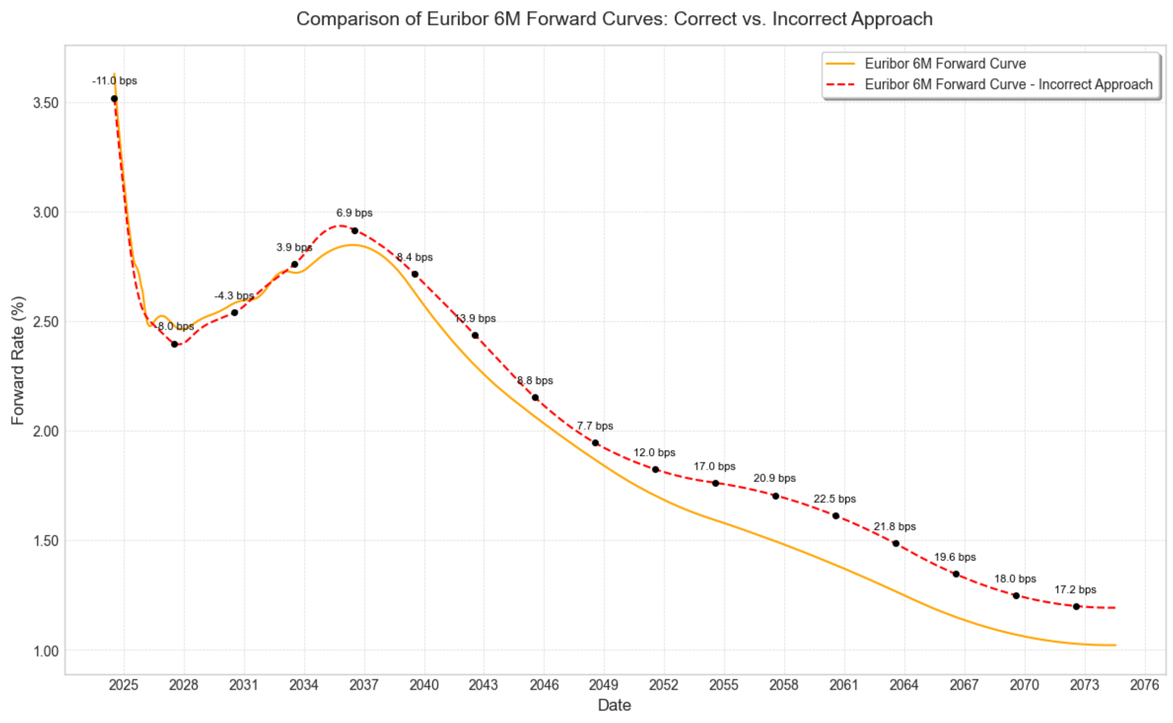

Figure 8.2: Comparison between correctly and incorrectly constructed Euribor 6M continuously compounded forward curves. The correct approach (orange) uses tenor-consistent instruments, while the incorrect approach (red dashed) employs instruments with a 3-month tenor.

The key observations from Figure 8.2 are as follows:

- **Initial discrepancies:** the curves diverge immediately, with the incorrectly constructed curve (red dashed) lying above the correctly constructed curve (orange) by approximately 11 basis points at the very short end (2025). This discrepancy arises because the instruments used in the incorrect approach do not accurately reflect the liquidity and risk premiums associated with the 6-month tenor.

- **Medium and long-term differences:** as the curves extend beyond the short term, the incorrectly constructed curve (red dashed) consistently overestimates forward rates, with discrepancies growing as large as 22.5 bps in the long term. This overestimation highlights the significant impact of using inappropriate tenor instruments, leading to potential mispricing in derivative markets.

9 Evaluating Different Combinations of Bootstrapping Sets

As outlined by Marco Bianchetti in his course slides for "Advanced Interest Rate Models and Markets" (University of Bologna, November 2024), the actual choice of the bootstrapping instruments is subject to many practical trading and risk management considerations: liquidity, bid-ask spreads, transaction costs, information to be included or not in the curve, etc. It's more an art than a science!
In this final chapter, we will therefore evaluate the effects of different sets of tenor-consistent bootstrapping instruments on the construction of the Euribor 3M forward curve. Additionally, we will examine the stability of the curves constructed using these sets by simulating various bid/ask spread shocks in our original dataset.

9.1 The Four Bootstrapping Sets

Four combinations of synthetic deposits, FRAs, futures, and swaps are analyzed. The first combination follows **the original set** described in Chapter 6, while the other combinations are configured as follows:
Combination 2: synthetic deposits + 0X3 FRA + futures + increased focus on swaps

- **Synthetic deposits (0-3m):** short-term synthetic deposits maturing between one week and two months, serving as the anchoring point for the curve's short end.
- **0X3 FRA (0-3m):** the initial anchor point for calibrating synthetic deposits, using a 0X3M FRA starting on July 16th, 2024, and maturing on October 16th, 2024.

- **Futures (3m-18m):** futures maturing between December 2024 and September 2025, extending the curve into the medium-term horizon.
- **Swaps (18m-50y):** long-term swaps with semi-annual floating legs, maturing between July 2026 and July 2074, which extend the curve into the long-term horizon.

Combination 3: synthetic deposits + 0X3M FRA + additional FRAs + swaps

- **Synthetic deposits (0-3m)**
- **0X3 FRA (0-3m)**
- **Additional FRAs (3m-2y):** FRAs maturing between October 2024 and July 2026, covering periods beyond the synthetic deposits and essential for extending the curve into the medium-term horizon.
- **Swaps (2y-50y)**

Combination 4: synthetic deposits + 0X3 FRA + additional FRAs + more focus on swaps

- **Synthetic deposits (0-3m)**
- **0X3 FRA (0-3m)**
- **Additional FRAs (3m-18m)**
- **Swaps (18m-50y)**

All curves are built using the `PiecewiseLogCubicDiscount` constructor and evaluated on July 12, 2024, as described in previous chapters.
Tables 9.1, 9.2, and 9.3 provide details on the market instruments used for the three additional combinations. Recall that the original combination utilizes the instruments listed in Table 6.1.

9.2 Comparative Analysis

Figure 9.1 compares the Euribor 3M forward curves constructed using the four different combinations over a five-year period from 2024 to 2029. This timeline clearly highlights the impact of using different sets, beyond 2027 swaps are employed across all four combinations.

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid**	**Ask**	**Mid**
Synthetic Deposit	*ON*	*July 16th, 2024*	*July 17th, 2024*			*3.800%*
Synthetic Deposit	*1W*	*July 16th, 2024*	*July 23rd, 2024*			*3.719%*
Synthetic Deposit	*2W*	*July 16th, 2024*	*July 30th, 2024*			*3.712%*
Synthetic Deposit	*3W*	*July 16th, 2024*	*August 6th, 2024*			*3.719%*
Synthetic Deposit	*1M*	*July 16th, 2024*	*August 16th, 2024*			*3.730%*
Synthetic Deposit	*2M*	*July 16th, 2024*	*September 16th, 2024*			*3.723%*
FRA	0X3	July 16th, 2024	October 16th, 2024	3.654%	3.674%	3.664%
Futures	DEC4	December 18th, 2024	March 18th, 2025	€96.74	€96.745	€96.742
Futures	MAR5	March 19th, 2025	June 19th, 2025	€96.97	€96.975	€96.972
Futures	JUN5	June 18th, 2025	September 18th, 2025	€97.165	€97.17	€97.168
Swap	18M	July 16th, 2024	January 16th, 2026	3.171%	3.211%	3.191%
Swap	2Y	July 16th, 2024	July 16th, 2026	3.043%	3.049%	3.046%
Swap	3Y	July 16th, 2024	July 16th, 2027	2.856%	2.897%	2.876%
Swap	4Y	July 16th, 2024	July 17th, 2028	2.778%	2.782%	2.780%
Swap	5Y	July 16th, 2024	July 16th, 2029	2.728%	2.732%	2.730%
Swap	6Y	July 16th, 2024	July 16th, 2030	2.704%	2.708%	2.706%
Swap	7Y	July 16th, 2024	July 16th, 2031	2.693%	2.697%	2.695%
Swap	8Y	July 16th, 2024	July 16th, 2032	2.695%	2.699%	2.697%
Swap	9Y	July 16th, 2024	July 18th, 2033	2.705%	2.709%	2.707%
Swap	10Y	July 16th, 2024	July 17th, 2034	2.718%	2.722%	2.720%
Swap	12Y	July 16th, 2024	July 16th, 2036	2.739%	2.780%	2.760%
Swap	15Y	July 16th, 2024	July 18th, 2039	2.768%	2.809%	2.788%
Swap	20Y	July 16th, 2024	July 18th, 2044	2.726%	2.767%	2.746%
Swap	25Y	July 16th, 2024	July 16th, 2049	2.630%	2.671%	2.650%
Swap	30Y	July 16th, 2024	July 16th, 2054	2.535%	2.576%	2.556%
Swap	40Y	July 16th, 2024	July 16th, 2064	2.382%	2.423%	2.402%
Swap	50Y	July 16th, 2024	July 16th, 2074	2.240%	2.281%	2.261%

Table 9.1: Market instruments for the Euribor 3M curve construction (Combo 2)

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid (%)**	**Ask (%)**	**Mid (%)**
Synthetic Deposit	*ON*	*July 16th, 2024*	*July 17th, 2024*			*3.800%*
Synthetic Deposit	*1W*	*July 16th, 2024*	*July 23rd, 2024*			*3.719%*
Synthetic Deposit	*2W*	*July 16th, 2024*	*July 30th, 2024*			*3.712%*
Synthetic Deposit	*3W*	*July 16th, 2024*	*August 6th, 2024*			*3.719%*
Synthetic Deposit	*1M*	*July 16th, 2024*	*August 16th, 2024*			*3.730%*
Synthetic Deposit	*2M*	*July 16th, 2024*	*September 16th, 2024*			*3.723%*
FRA	0X3	July 16th, 2024	October 16th, 2024	3.654%	3.674%	3.664%
FRA	1X4	August 16th, 2024	November 18th, 2024	3.579%	3.599%	3.589%
FRA	2X5	September 16th, 2024	December 16th, 2024	3.490%	3.520%	3.505%
FRA	3X6	October 16th, 2024	January 16th, 2025	3.412%	3.432%	3.422%
FRA	4X7	November 18th, 2024	February 17th, 2025	3.330%	3.360%	3.345%
FRA	5X8	December 16th, 2024	March 17th, 2025	3.240%	3.270%	3.255%
FRA	6X9	January 16th, 2025	April 16th, 2025	3.160%	3.190%	3.175%
FRA	7X10	February 17th, 2025	May 16th, 2025	3.089%	3.109%	3.099%
FRA	8X11	March 17th, 2025	June 16th, 2025	3.020%	3.040%	3.030%
FRA	9X12	April 16th, 2025	July 16th, 2025	2.947%	2.967%	2.957%
FRA	12X15	July 16th, 2025	October 16th, 2025	2.774%	2.794%	2.784%
FRA	15X18	October 16th, 2025	January 16th, 2026	2.625%	2.675%	2.650%
FRA	18X21	January 16th, 2026	April 16th, 2026	2.540%	2.590%	2.565%
FRA	21X24	April 16th, 2026	July 16th, 2026	2.495%	2.545%	2.520%
Swap	3Y	July 16th, 2024	July 16th, 2027	2.856%	2.897%	2.876%
Swap	4Y	July 16th, 2024	July 17th, 2028	2.778%	2.782%	2.780%
Swap	5Y	July 16th, 2024	July 16th, 2029	2.728%	2.732%	2.730%
Swap	6Y	July 16th, 2024	July 16th, 2030	2.704%	2.708%	2.706%
Swap	7Y	July 16th, 2024	July 16th, 2031	2.693%	2.697%	2.695%
Swap	8Y	July 16th, 2024	July 16th, 2032	2.695%	2.699%	2.697%
Swap	9Y	July 16th, 2024	July 18th, 2033	2.705%	2.709%	2.707%
Swap	10Y	July 16th, 2024	July 17th, 2034	2.718%	2.722%	2.720%
Swap	12Y	July 16th, 2024	July 16th, 2036	2.739%	2.780%	2.760%
Swap	15Y	July 16th, 2024	July 18th, 2039	2.768%	2.809%	2.788%
Swap	20Y	July 16th, 2024	July 18th, 2044	2.726%	2.767%	2.746%
Swap	25Y	July 16th, 2024	July 16th, 2049	2.630%	2.671%	2.650%
Swap	30Y	July 16th, 2024	July 16th, 2054	2.535%	2.576%	2.556%
Swap	40Y	July 16th, 2024	July 16th, 2064	2.382%	2.423%	2.402%
Swap	50Y	July 16th, 2024	July 16th, 2074	2.240%	2.281%	2.261%

Table 9.2: Market instruments for the Euribor 3M curve construction (Combo 3)

Instrument	**Tenor**	**Start Date**	**Maturity Date**	**Bid (%)**	**Ask (%)**	**Mid (%)**
Synthetic Deposit	*ON*	*July 16th, 2024*	*July 17th, 2024*			*3.800%*
Synthetic Deposit	*1W*	*July 16th, 2024*	*July 23rd, 2024*			*3.719%*
Synthetic Deposit	*2W*	*July 16th, 2024*	*July 30th, 2024*			*3.712%*
Synthetic Deposit	*3W*	*July 16th, 2024*	*August 6th, 2024*			*3.719%*
Synthetic Deposit	*1M*	*July 16th, 2024*	*August 16th, 2024*			*3.730%*
Synthetic Deposit	*2M*	*July 16th, 2024*	*September 16th, 2024*			*3.723%*
FRA	2X5	September 16th, 2024	December 16th, 2024	3.490%	3.520%	3.505%
FRA	3X6	October 16th, 2024	January 16th, 2025	3.412%	3.432%	3.422%
FRA	4X7	November 18th, 2024	February 17th, 2025	3.330%	3.360%	3.345%
FRA	5X8	December 16th, 2024	March 17th, 2025	3.240%	3.270%	3.255%
FRA	6X9	January 16th, 2025	April 16th, 2025	3.160%	3.190%	3.175%
FRA	7X10	February 17th, 2025	May 16th, 2025	3.089%	3.109%	3.099%
FRA	8X11	March 17th, 2025	June 16th, 2025	3.020%	3.040%	3.030%
FRA	9X12	April 16th, 2025	July 16th, 2025	2.947%	2.967%	2.957%
FRA	12X15	July 16th, 2025	October 16th, 2025	2.774%	2.794%	2.784%
Swap	18M	July 16th, 2024	January 16th, 2026	3.171%	3.211%	3.191%
Swap	2Y	July 16th, 2024	July 16th, 2026	3.043%	3.049%	3.046%
Swap	3Y	July 16th, 2024	July 16th, 2027	2.856%	2.897%	2.876%
Swap	4Y	July 16th, 2024	July 17th, 2028	2.778%	2.782%	2.780%
Swap	5Y	July 16th, 2024	July 16th, 2029	2.728%	2.732%	2.730%
Swap	6Y	July 16th, 2024	July 16th, 2030	2.704%	2.708%	2.706%
Swap	7Y	July 16th, 2024	July 16th, 2031	2.693%	2.697%	2.695%
Swap	8Y	July 16th, 2024	July 16th, 2032	2.695%	2.699%	2.697%
Swap	9Y	July 16th, 2024	July 18th, 2033	2.705%	2.709%	2.707%
Swap	10Y	July 16th, 2024	July 17th, 2034	2.718%	2.722%	2.720%
Swap	12Y	July 16th, 2024	July 16th, 2036	2.739%	2.780%	2.760%
Swap	15Y	July 16th, 2024	July 18th, 2039	2.768%	2.809%	2.788%
Swap	20Y	July 16th, 2024	July 18th, 2044	2.726%	2.767%	2.746%
Swap	25Y	July 16th, 2024	July 16th, 2049	2.630%	2.671%	2.650%
Swap	30Y	July 16th, 2024	July 16th, 2054	2.535%	2.576%	2.556%
Swap	40Y	July 16th, 2024	July 16th, 2064	2.382%	2.423%	2.402%
Swap	50Y	July 16th, 2024	July 16th, 2074	2.240%	2.281%	2.261%

Table 9.3: Market instruments for the Euribor 3M curve construction (Combo 4)

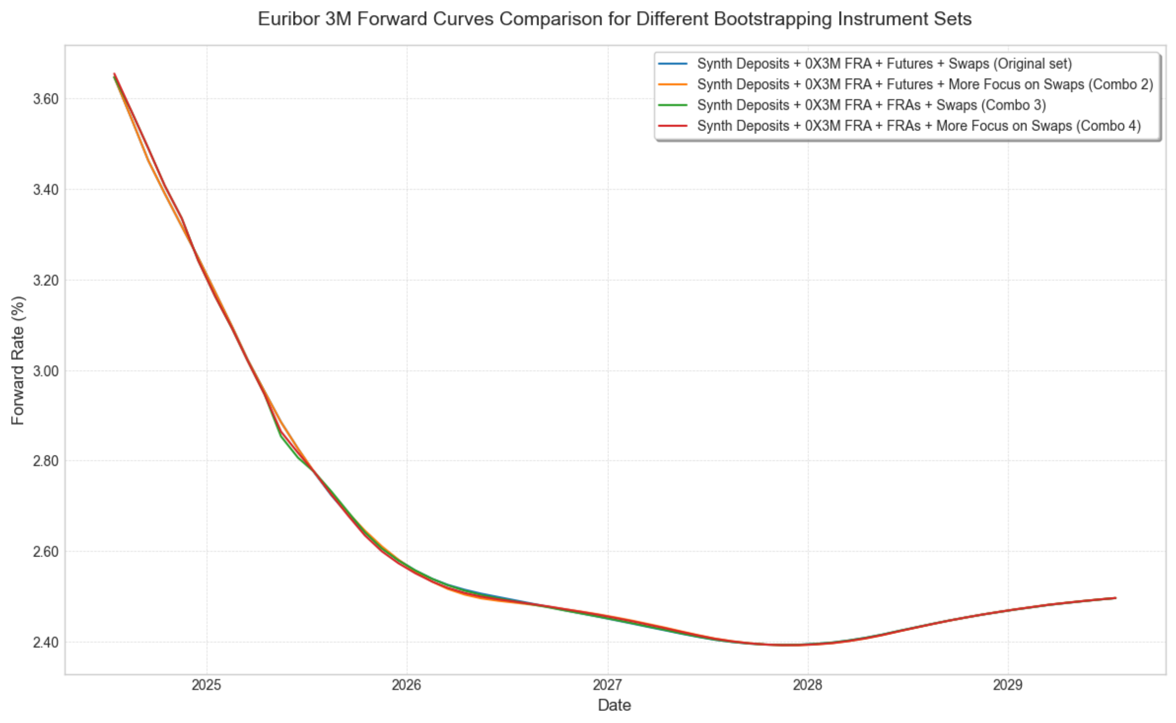

Figure 9.1: Comparison of Euribor 3M continuously compounded forward curves for different bootstrapping instrument sets. The figure displays the forward curves constructed using four distinct combinations of market instruments: the original set (blue line) includes synthetic deposits, the 0X3M FRA, futures, and swaps; Combo 2 (orange line) places more emphasis on swaps; Combo 3 (green line), replaces futures with additional FRAs to capture medium-term rates; and Combo 4 (red line) also uses a mix of FRAs but with an increased focus on swaps.

From Figure 9.1, it's evident that each configuration yields a relatively smooth forward curve, indicating that all these instrument configurations can be considered effective for bootstrapping. However, Combo 3 is slightly less smooth compared to the others, showing marginally more variability along the curve.

9.3 Stability Assessment

To evaluate the reached curve stability in each of our four bootstrapping sets —specifically, its sensitivity to distortions resulting from rate fluctuations in the bootstrapping instruments—we applied random perturbations to the bid-ask spreads in our dataset. This simulation replicates conditions such as synthetic bid/ask spreads from market contributors, liquidity constraints, or potential discrepancies in quote entries.

This "stress-test" of the bid-ask spread was conducted as follows:

- **Random variable generation:** for each market instrument class (e.g., OISs, Futures, etc.), we generated a random variable ε_i with a mean of zero and a standard deviation equal to the class-specific average bid-ask spread $\overline{S}$:

$$\varepsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \quad \text{where} \quad \sigma = \overline{S} = \frac{1}{N}\sum_{i=1}^{N} S_i$$

  Where:

  - N is the total number of instruments.
  - $S_i = A_i - B_i$ is the bid-ask spread for instrument i;
  - A_i and B_i are the ask and bid prices of instrument i, respectively.

- **Shock application:** a total of $k = 1,000$ new simulated bid and ask prices were obtained by applying the random shocks, ensuring that the bid-ask spread remains positive. For each instrument i in simulation k:

  1. **Apply random shock:**

$$B_i^{\text{shocked},(k)} = B_i\left(1 + \varepsilon_i^{(k)}\right)$$
$$A_i^{\text{shocked},(k)} = A_i\left(1 + \varepsilon_i^{(k)}\right)$$

  2. **Ensure positive bid-ask spread:** if $B_i^{\text{shocked},(k)} \geq A_i^{\text{shocked},(k)}$, adjust the shocked ask price to maintain a positive spread:

$$A_i^{\text{shocked},(k)} = B_i^{\text{shocked},(k)} + |A_i^{\text{shocked},(k)} - B_i^{\text{shocked},(k)}|$$

  3. **Calculate new mid-rates:**

$$M_i^{\text{shocked},(k)} = \frac{B_i^{\text{shocked},(k)} + A_i^{\text{shocked},(k)}}{2}$$

- **Curve reconstruction:** using the new mid-rates $M_i^{\text{shocked},(k)}$, we reconstructed the forward curves for each simulation k.

This process was then repeated for each of the four bootstrapping sets, the results are illustrated in Figure 9.2.

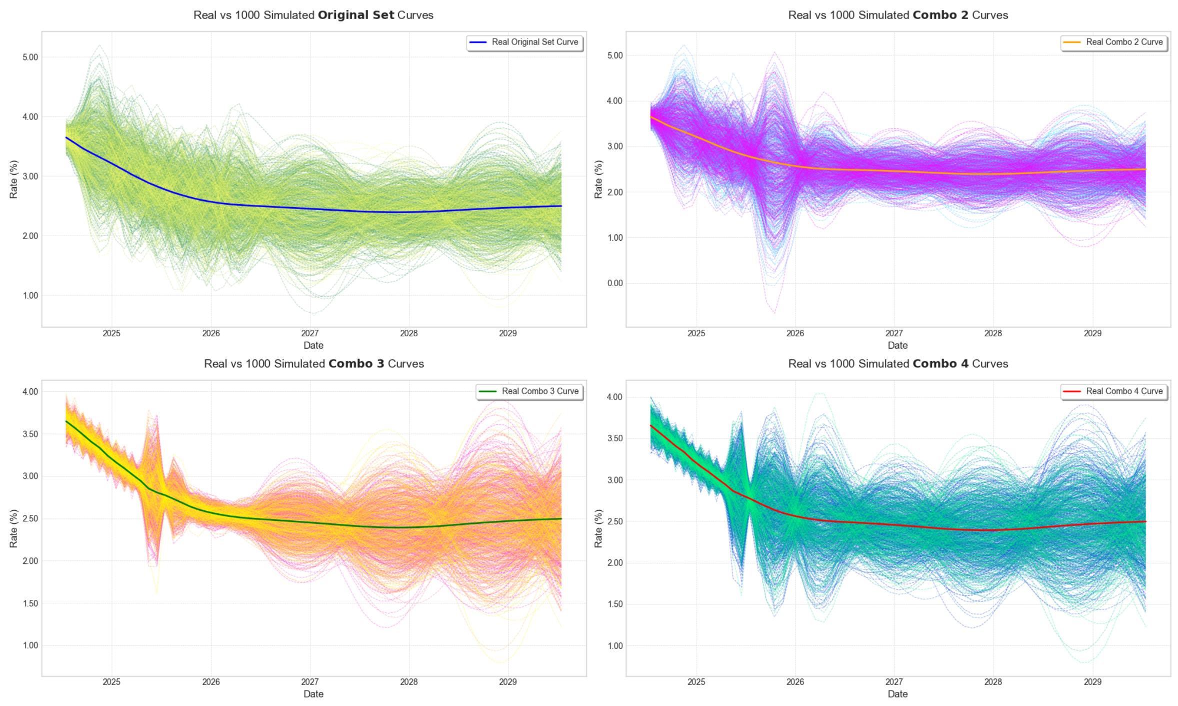

Figure 9.2: Comparison between the‘real’ and one thousand shock-simulated forward curves for each of the four bootstrapping sets. Each subplot represents a different bootstrapping set: the top left shows the "Original Set" (blue line) against 1,000 simulated scenarios in green, the top right displays "Combo 2" (orange line) against simulations in purple, the bottom left depicts "Combo 3" (green line) against simulations in red, and the bottom right shows "Combo 4" (red line) against simulations in teal. The visual spread in each subplot indicates the degree of sensitivity of each bootstrapping set to bid/ask spread shocks, highlighting the relative stability or instability of the forward curve across different sets.

9.3.1 Statistical Analysis

A statistical analysis was conducted to evaluate the stability achieved of each combination. We calculated the average root mean square deviation between the ‘real’ curve and each of the 1,000 simulations generated for the four bootstrapping sets. Additionally, we considered the average maximum and minimum fluctuations observed in each case.

The root mean square deviation for each simulation k was calculated using the following formula:

$$RMSD(k) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}^t(k) - y_t)^2}$$

where:

- $RMSD(k)$ is the root mean square deviation for simulation k.
- T is the number of time points on the forward curve.
- $\hat{y}^t(k)$ is the forward rate at time t from the simulated curve in simulation k.
- y_t is the corresponding forward rate at time t from the real curve.

We then computed the average root mean square deviation across all simulations:

$$\overline{RMSD} = \frac{1}{K}\sum_{k=1}^{K} RMSD(k)$$

where $K = 1000$ is the total number of simulations.

Additionally, we calculated the average maximum positive and negative deviations observed across all the simulations:

$$\text{Average Max-Positive Deviation} = \frac{1}{K}\sum_{k=1}^{K}\left(\max_t\left(\hat{y}^t(k) - y_t\right)\right) \times 10^4 \quad \text{(in basis points)}$$

$$\text{Average Max-Negative Deviation} = \frac{1}{K}\sum_{k=1}^{K}\left(\min_t\left(\hat{y}^t(k) - y_t\right)\right) \times 10^4 \quad \text{(in basis points)}$$

The factor 10^4 converts the rate differences into basis points for easier interpretation.

The key findings are as follows:

- **RMSDs:** the average RMSD ranged from 23.49 bps to 38.38 bps. Combo 3 demonstrated the lowest average RMSD at 23.49 bps, indicating superior stability under stress conditions.

- **Max positive and negative deviations:** the most significant deviations were observed in Combo 2 and the Original Set, with average max-positive deviations of 82.03 bps and 77.40 bps, respectively, and average max-negative of -80.56 bps and -77.55 bps. Combo 3 exhibited less extreme deviations, further highlighting its resilience under stressed market conditions.

- **Statistical significance:** we also conducted a one-sample t-test for the RMSD to determine if the average RMSD is statistically significantly different from zero. As expected, for each bootstrapping set, the t-statistic and p-value from the one-sample t-test indicate that the average RMSD is statistically significant at the 99,99%

The detailed statistical metrics for each bootstrapping set are presented in Table 9.4.

Bootstrapping Set	Average RMSD (bps)	Average Max-Positive Deviation (bps)	Average Max-Negative Deviation (bps)	t-statistic	p-value
Combo 3	23.49	47.93	-47.80	73.76	0.0000
Combo 4	28.20	57.96	-57.29	86.30	0.0000
Combo 2	37.30	82.03	-80.56	88.69	0.0000
Original Set	38.38	77.40	-77.55	105.91	0.0000

Table 9.4: Statistical metrics achieved by the four bootstrapping sets under market shock simulations

Conclusion

This analysis underscores the importance of instrument selection in the bootstrapping process. Configurations that emphasize FRAs, such as Combo 3 and Combo 4, demonstrate better stability with lower RMSD and lower average max deviations, making them more resilient under stress scenarios. In contrast, configurations with a more futures-focused mix, like the Original Set and Combo 2, exhibit higher RMSD and greater levels of drift.
In future work, it may be beneficial to explore hybrid configurations that balance the depth of FRAs with the liquidity of futures, aiming to optimize both stability and market representativeness in the forward curve construction.

10 Conclusions and Future Work

Conclusions

Throughout this research, we have examined the European multiple-curve framework for constructing different yield curves, particularly in light of the recent transition from EONIA to €STR. The main findings of this study are as follows:

- **Evolution of curve construction:** the shift from a single-curve to a multiple-curve framework represents one of the most significant changes in the fixed income landscape since the 2008 financial crisis. This evolution has fundamentally altered how yield curves are constructed and interpreted.
- **Instrument selection and bootstrapping:** the bootstrapping process used to construct forward curves highlights the importance of selecting an appropriate mix of financial instruments. An incorrectly constructed bootstrapping set, particularly one that fails to account for tenor consistency, can result in significant discrepancies in the final curve.
- **Stability of yield curves:** stress testing involving bid/ask spread simulations demonstrated that forward curves built with a bootstrapping set more focused on FRAs exhibited greater stability. This suggests that such configurations may be more resilient to market fluctuations.

Future Work

Several opportunities for further research have been identified throughout this thesis. The following directions are suggested for future investigation:

- **Refinement of dual-curve methodologies:** future research could explore the application of machine learning techniques to refine curve construction methodologies. Such approaches could enhance curve accuracy and stability under varying market conditions, optimizing the selection of bootstrapping sets in different market contexts.
- **Extended market shock simulations:** a broader analysis of market shock scenarios, including geopolitical events or severe market dislocations, could provide deeper insights into the resilience of yield curves. This would help in developing more robust risk management strategies, particularly in uncertain market conditions.
- **Incorporation of alternative reference rates:** future research may also consider incorporating alternative reference rates, such as SOFR, into a comprehensive curve construction framework. Investigating the influence of cross-currency swaps and international markets could reveal valuable insights into the dynamics and interconnections of global finance.

In conclusion, this thesis provides an empirical foundation that sheds light on the multiple-curve construction framework in the modern market environment. As financial markets continue to evolve, further research will be essential to address emerging challenges and harness potential opportunities in this dynamic landscape.

Bibliography

(ECB), European Central Bank (2022). *The euro short-term rate (€STR): Completing the transition to the new euro benchmark*. Accessed: 2024-08-18. URL: `https://www.ecb.europa.eu/pub/economic-bulletin/html/eb202204.en.html`.

(EMMI), European Money Markets Institute (2019a). *EURIBOR History*. Accessed: 2024-08-18. URL: `https://www.emmi-benchmarks.eu/euribor-org/about-euribor.html`.

Ametrano, Ferdinando M. and Marco Bianchetti (2013). "Everything You Always Wanted to Know About Multiple Interest Rate Curve Bootstrapping but Were Afraid to Ask". In: Available at SSRN: `https://ssrn.com/abstract=2219548` or `http://dx.doi.org/10.2139/ssrn.2219548`.

Ballabio, Luigi and Goutham Balaraman (2023). *QuantLib Python Cookbook*. Leanpub. URL: `https://leanpub.com/quantlibpythoncookbook`.

Bank, European Central (2019). "The Euro Short-Term Rate (€STR): Methodology and Policies". In: Accessed: 2024-09-22. URL: `https://www.ecb.europa.eu/stats/euro-short-term-rates/interest_rate_benchmarks/WG_euro_risk-free_rates/shared/pdf/ecb.ESTER_methodology_and_policies.en.pdf`.

Bianchetti, Marco and Mattia Carlicchi (2011). "Interest Rates After the Credit Crunch: Multiple-Curve Vanilla Derivatives and SABR". In: *arXiv preprint arXiv:1103.2567*. URL: `https://arxiv.org/pdf/1103.2567`.

BIS (2013). *The changing dynamics of the yield curve: its implications for financial stability*. BIS Quarterly Review.

Board, Financial Stability (2014). *Reforming Major Interest Rate Benchmarks*. Accessed: 2024-08-18. URL: `https://www.fsb.org/2014/07/r_140722/`.

Burgess, Nicholas (2019). “Convexity Adjustments Made Easy: An Overview of Convexity Adjustment Methodologies in Interest Rate Markets”. In: *Journal of Economics and Financial Analysis* 3.2, pp. 41–83. ISSN: 2521-6619. DOI: 10.1991/jefa.v3i2.a28. URL: https://ojs.tripaledu.com/jefa/article/view/49.

Charles River Development (2016). *Valuing Interest Rate Swaps: The Importance of Dual-Curve Stripping*. Accessed: 2024-09-03. URL: https://www.crd.com/insights/valuing-interest-rate-swaps-the-importance-of-dual-curve-stripping/.

Cui, Jia, Francis In, and Ephraim A. Maharaj (2016). “What Drives the LIBOR–OIS Spread? Evidence from Five Major Currency LIBOR–OIS Spreads”. In: *International Review of Economics & Finance* 45, pp. 150–164. URL: https://www.sciencedirect.com/science/article/pii/S1059056016300156.

ECB (2021). “€STR Transition Report”. In: URL: https://www.ecb.europa.eu/stats/euro-short-term-rates/interest_rate_benchmarks/WG_euro_risk-free_rates/shared/pdf/FactsheetTransitionEONIAtoEuroSTR.pdf.

ECB publishes indicative operational calendars for 2024 (2023). Accessed: 15 September 2023. European Central Bank. URL: https://www.ecb.europa.eu/press/pr/date/2023/html/ecb.pr230915~1f29267423.en.html.

European Central Bank (2016). *What are minimum reserve requirements?* https://www.ecb.europa.eu/ecb-and-you/explainers/tell-me/html/minimum_reserve_req.en.html. Accessed: 2024-08-18. URL: https://www.ecb.europa.eu/ecb-and-you/explainers/tell-me/html/minimum_reserve_req.en.html.

European Central Bank (ECB) (2024). “Euro short-term rate - Volume-weighted trimmed mean rate, Daily - businessweek”. In: *European Central Bank Website*. Last updated: 24 September 2024 08:04 CEST. URL: https://data.ecb.europa.eu/data/datasets/EST/EST.B.EU000A2X2A25.WT?chart.

European Money Markets Institute (EMMI) (2019b). *EURIBOR Hybrid Methodology*. Accessed: 2024-08-18. URL: https://www.emmi-benchmarks.eu/benchmarks/euribor/methodology/.

Johannes, Michael and Suresh Sundaresan (2007). “The Impact of Collateralization on Swap Rates”. In: *The Journal of Finance* 62.1, pp. 383–410.

Lang, Matthias and Thomas Gehrig (2007). “Modeling the Euro Overnight Rate”. In: *ScienceDirect*. URL: `https://www.sciencedirect.com/science/article/abs/pii/S0927539807000485`.

LCH.Clearnet (2010). *LCH.Clearnet Successfully Implements New Margining Process for OTC Interest Rate Swaps*. Accessed: 2024-09-03. URL: `https://secure-area.lchclearnet.com/media_centre/press_releases/2010-06-17.asp`.

Numerix (Oct. 2013). *Numerix Model Calibration: The Multiple Curve Approach*. Tech. rep. Accessed: 2024-09-22. Numerix. URL: `https://1library.net/document/yn4oevjz-numerix-model-calibration-the-multiple-curve-approach.html`.

St. Louis, Federal Reserve Bank of (2009). *What the Libor-OIS Spread Says*. Accessed: 2024-09-19. URL: `https://files.stlouisfed.org/files/htdocs/publications/es/09/ES0924.pdf`.

Appendix A: Replication of the Curves Constructed with QuantLib Using Eikon's Curve Builder

Here we present the curves constructed in this study, replicated using Eikon. Due to differences in instrument availability and functionalities within the Curve Builder, the following *adjustments* are made:

- All Euribor forward curves are constructed without synthetic deposits.
- The futures used are continuous contracts, differing from those in our study.
- It was not possible to include the ECB forward OIS.
- The curve interpolation is performed using cubic discounting instead of the log cubic discount method used in this study.
- The curves displayed are not on the same scale as those in our original study.

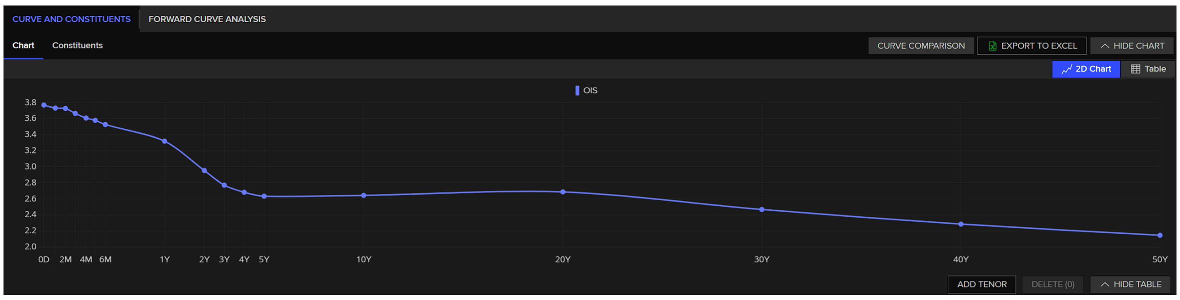

Figure 1: €STR Zero curve — Bootstrapped using the *adjusted* instrument set in Chapter 5.

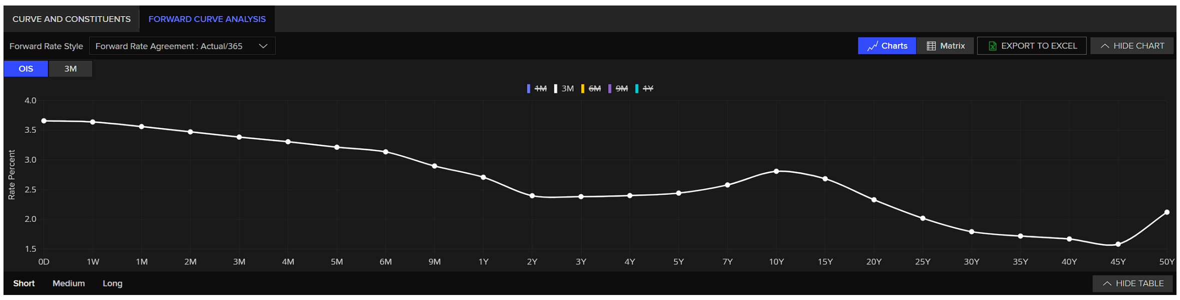
Forward Rate Style: Forward Rate Agreement : Actual/365

Figure 2: Euribor 3M forward curve — Bootstrapped using the *adjusted* original instrument set in Chapter 6.

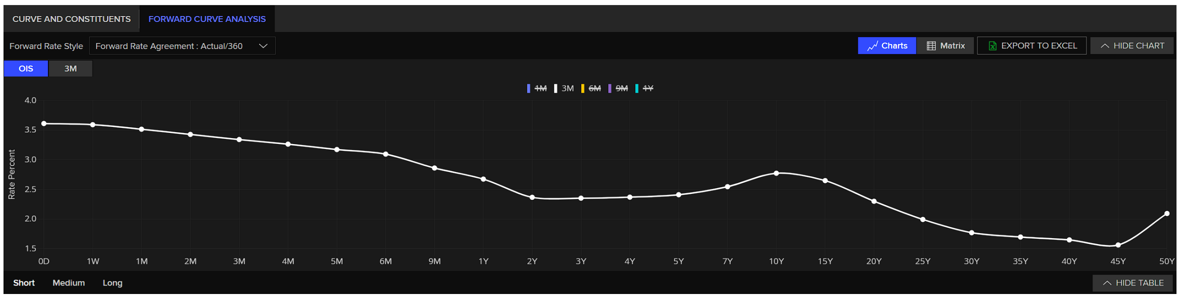
Forward Rate Style: Forward Rate Agreement : Actual/360

Figure 3: Euribor 3M forward curve — Bootstrapped using the *adjusted* combination 2 instrument set in Chapter 9.

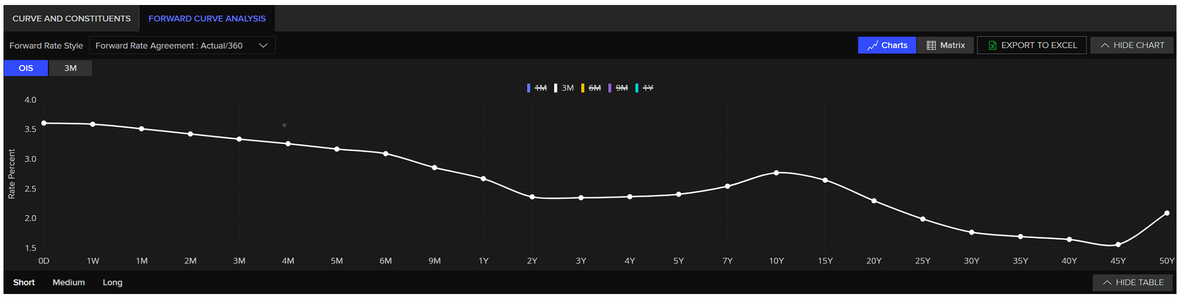
Forward Rate Style: Forward Rate Agreement : Actual/360

Figure 4: Euribor 3M forward curve — Bootstrapped using the *adjusted* combination 3 instrument set in Chapter 9.

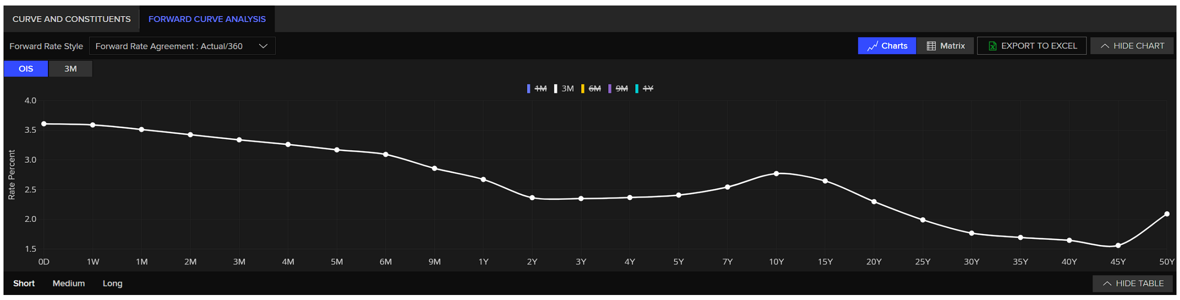

Figure 5: Euribor 3M forward curve — Bootstrapped using the *adjusted* combination 4 instrument set in Chapter 9.

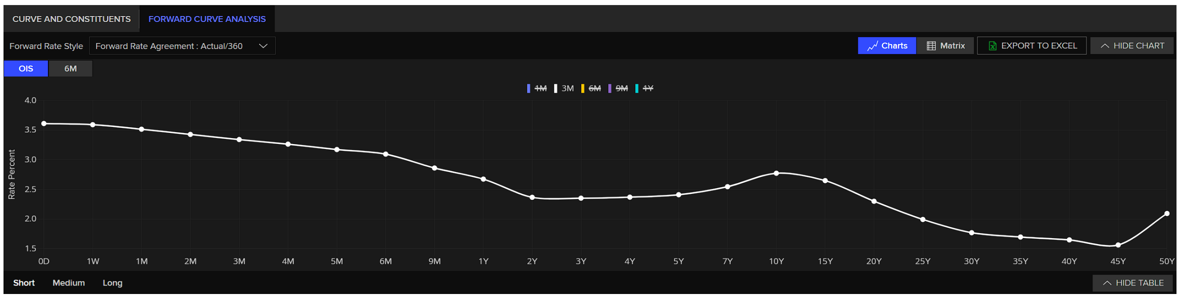

Figure 6: Euribor 6M forward curve — Bootstrapped using the *adjusted* instrument set in Chapter 8.

Appendix B: Python Code

Below is the Python code used for this study. You can download the data and the Jupyter notebook with the commented code at the following github link: https://github.com/MITRIDAT3/Multiple-Curve-Construction-Study

.1 Data Import and Preparation

```
# Import necessary libraries
import pandas as pd
import numpy as np
import QuantLib as ql
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.ticker import PercentFormatter
from matplotlib.ticker import FuncFormatter
from datetime import datetime
from scipy.stats import ttest_1samp

# Input file paths
ois_file_path = 'Dati/EURESTOIS=.xlsm'
forwards_ois_ecb_file_path = 'Dati/EUESTECBF=ICAP.xlsm'
deposit_file_path = 'Dati/EURDEPO=.xlsm'
fra3M_file_path = 'Dati/EUR3MFRA=.xlsm'
fra6M_file_path = 'Dati/6M/EUR6MFRA=.xlsm'
swap3M_file_path = 'Dati/EURAB3EIRS=.xlsm'
swap6M_file_path = 'Dati/6M/EURAB6EIRS=.xlsm'
```

```
futures3M_file_path = 'Dati/FUTURE3M.xlsm'

# ECB maintenance periods
ecb_maintenance_periods = {
    "JAN24": {"start": ql.Date(31, ql.January, 2024), "end": ql.
       Date(12, ql.March, 2024)},
    "MAR24": {"start": ql.Date(13, ql.March, 2024), "end": ql.Date
       (16, ql.April, 2024)},
    "APR24": {"start": ql.Date(17, ql.April, 2024), "end": ql.Date
       (11, ql.June, 2024)},
    "JUN24": {"start": ql.Date(12, ql.June, 2024), "end": ql.Date
       (23, ql.July, 2024)},
    "JUL24": {"start": ql.Date(24, ql.July, 2024), "end": ql.Date
       (17, ql.September, 2024)},
    "SEP24": {"start": ql.Date(18, ql.September, 2024), "end": ql.
       Date(22, ql.October, 2024)},
    "OCT24": {"start": ql.Date(23, ql.October, 2024), "end": ql.
       Date(17, ql.December, 2024)},
    "DEC24": {"start": ql.Date(18, ql.December, 2024), "end": "tbd"
       }
}

# Function to read data and calculate average rates
def load_and_process_data(file_path, sheet_name):
    data = pd.read_excel(file_path, sheet_name=sheet_name, engine='
       openpyxl')
    data['Mid'] = (data['Bid'] + data['Ask']) / 2
    return data

# Load and process data
ois_data = load_and_process_data(ois_file_path, 'Sheet1')
forwards_ois_ecb_data = load_and_process_data(
   forwards_ois_ecb_file_path, 'Sheet1')
```

```
deposit_data = load_and_process_data(deposit_file_path, 'Sheet1')
fra_data3M = load_and_process_data(fra3M_file_path, 'Sheet1')
fra_data3M_0x3_rate = pd.DataFrame({'Maturity':['0X3','1X4'], 'Bid'
    :[3.654,3.579],'Ask':[3.674,3.599]}) # 12/07/2024 0x3 rate,
    different contributor than the other fra rates
fra_data3M_0x3_rate['Mid'] = (fra_data3M_0x3_rate['Bid'] +
    fra_data3M_0x3_rate['Ask']) / 2
fra_data6M = load_and_process_data(fra6M_file_path, 'Sheet1')
fra_data6M_0x6_rate = pd.DataFrame({'Maturity':['0X6'], 'Bid'
    :[3.652],'Ask':[3.672]}) # 12/07/2024 0x6 rate, different
    contributor than the other fra rates
fra_data6M_0x6_rate['Mid'] = (fra_data6M_0x6_rate['Bid'] +
    fra_data6M_0x6_rate['Ask']) / 2
swap_data3M = load_and_process_data(swap3M_file_path, 'Sheet1')
swap_data6M = load_and_process_data(swap6M_file_path, 'Sheet1')
futures_data3M = load_and_process_data(futures3M_file_path, 'Sheet1
    ')

# Merge fra 3M and 6M data
fra_data3M = pd.concat([fra_data3M_0x3_rate, fra_data3M],
    ignore_index=True)
fra_data6M = pd.concat([fra_data6M_0x6_rate, fra_data6M],
    ignore_index=True)

# Set Golden ratio figure size
width = 12
height = width / 1.618
```

.2 Convert Maturities and Set Evaluation Date

```
# Map to convert calendar units to QuantLib periods
period_map = {
```

```python
    "D": ql.Days,
    "W": ql.Weeks,
    "M": ql.Months,
    "Y": ql.Years
}

# Map for fixing days
fixing_days_map = {
    "ON": 0,
    "TN": 1,
    "SN": 2,
    "SW": 2,
    "2W": 2,
    "3W": 2
}

# Function to convert maturities to QuantLib periods, handling
    special cases
def convert_to_period(maturity):
    special_maturity_format = {
        "ON": ql.Period(1, ql.Days),
        "TN": ql.Period(2, ql.Days),
        "SN": ql.Period(3, ql.Days),
        "SW": ql.Period(1, ql.Weeks)
    }

    if maturity in special_maturity_format:
        return special_maturity_format[maturity]

    try:
        if 'X' in maturity:
            start, end = maturity.split('X')
            return ql.Period(int(start), ql.Months), ql.Period(int(end), ql.Months)
        num = int(maturity[:-1])
    except ValueError:
        raise ValueError(f"Invalid maturity format: {maturity}")

    unit = maturity[-1]
    return ql.Period(num, period_map[unit])

# Function to convert QuantLib dates to datetime for matplotlib
def _to_datetime(_date):
    return datetime(_date.year(), _date.month(), _date.dayOfMonth())

# Convert maturities to QuantLib periods
ois_data['Period'] = ois_data['Maturity'].apply(convert_to_period)
deposit_data['Period'] = deposit_data['Maturity'].apply(convert_to_period)
fra_data3M['Start_Period'], fra_data3M['End_Period'] = zip(*fra_data3M['Maturity'].apply(convert_to_period))
fra_data6M['Start_Period'], fra_data6M['End_Period'] = zip(*fra_data6M['Maturity'].apply(convert_to_period))
swap_data3M['Period'] = swap_data3M['Maturity'].apply(convert_to_period)
swap_data6M['Period'] = swap_data6M['Maturity'].apply(convert_to_period)

# Process forward OIS ECB data within maintenance periods
# Remove unused maintenance periods
del ecb_maintenance_periods["JAN24"]
del ecb_maintenance_periods["MAR24"]
del ecb_maintenance_periods["APR24"]
del ecb_maintenance_periods["JUN24"]
del ecb_maintenance_periods["DEC24"]

# Remove unused forward OIS ECB data
forwards_ois_ecb_data = forwards_ois_ecb_data[(forwards_ois_ecb_data.index > 0) & (forwards_ois_ecb_data.index < 4)]

# Set start and end dates for maintenance periods and forward OIS ECB maturities
period_keys = list(ecb_maintenance_periods.keys())
forwards_ois_ecb_data['Start_Period'] = [ecb_maintenance_periods[key]['start'] for key in period_keys[:len(forwards_ois_ecb_data)]]
forwards_ois_ecb_data['End_Period'] = [ecb_maintenance_periods[key]['end'] + 1 for key in period_keys[:len(forwards_ois_ecb_data)]]
forwards_ois_ecb_data.drop(columns=['Maturity'], inplace=True)

# Set IMM dates for Euribor futures maturities
def imm_date_corrected(data):
    month_str = data[:3]  # First three characters represent the month (e.g., 'JUL')
    year_str = data[3:]   # Remaining characters represent the year (e.g., '4')

    month = datetime.strptime(month_str, '%b').month
    year = 2020 + int(year_str)  # Corrected to handle years in 2020s

    # Calculate the next IMM date
    date = datetime(year, month, 1)
    return ql.IMM.nextDate(ql.Date(date.day, date.month, date.year))
```

```python

futures_data3M['Start_Date'] = futures_data3M['Maturity'].apply(imm_date_corrected)
futures_data3M = futures_data3M[(futures_data3M.index > 4) & (futures_data3M.index < 11)]

# Define maturities based on Ametrano and Bianchetti study
ESTR_curve_deposit_maturities_chosen = ['ON', 'TN', 'SN']
fra3M_maturities_chosen = ['2X5','3X6', '4X7', '5X8', '6X9', '7X10', '8X11', '9X12','12X15','15X18', '16X19', '18X21','21X24']
fra_maturities_chosen = ['1X7', '2X8', '3X9', '4X10', '5X11', '6X12', '7X13', '8X14', '9X15', '10X16', '11X17', '12X18', '18X24']
swap_maturities_chosen = ['3Y', '4Y', '5Y', '6Y', '7Y', '8Y', '9Y', '10Y', '12Y', '15Y', '20Y', '25Y', '30Y', '40Y', '50Y']
futures_maturities_chosen = ['DEC4', 'MAR5', 'JUN5', 'SEP5', 'DEC5', 'MAR6']

# DEBUG: Print data
DEBUG_DATI = False
if DEBUG_DATI:
    print(f"Deposit Data: {deposit_data}")
    print(f"OIS Data: {ois_data}")
    print(f"Forwards OIS ECB Data: {forwards_ois_ecb_data}")
    print(f"FRA Data: {fra_data3M}")
    print(f"Swap Data: {swap_data3M}")
    print(f"Futures Data: {futures_data3M}")

# Set the evaluation date for the curves
settle_date = ql.Date(12, 7, 2024)
ql.Settings.instance().evaluationDate = settle_date
```

.3 Building the €STR Discount Curve

```
def estr_curve_builder(
    ois_data=ois_data,
    deposit_data=deposit_data,
    forwards_ois_ecb_data=forwards_ois_ecb_data,
    graph=True):

    # Initialize an empty list to contain €STR curve helpers
    estr_helpers = []

    # Initialize the €STR curve
    estr = ql.Estr()

    # 1) Deposits
    # Add helpers for the first part of the curve from selected
       deposits
    selected_depo_data = deposit_data[deposit_data['Maturity'].isin
       (ESTR_curve_deposit_maturities_chosen)]

    estr_helpers += [
        ql.DepositRateHelper(
            ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)),
            ql.Period(1, ql.Days),
            fixing_days_map[row['Maturity']],
            ql.TARGET(),
            ql.Following,
            False,
            ql.Actual360()
        )
        for _, row in selected_depo_data.iterrows()
    ]

    # 2) OIS pre-ECB
    ois_pre_ecb = ois_data[ois_data.index < 1] # Watch for overlapping with forward OIS ECB
    # Add helpers for the second part of the curve from selected OIS pre-ECB
    estr_helpers += [
        ql.OISRateHelper(
            2, row['Period'], ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), estr
        )
        for index, row in ois_pre_ecb.iterrows()
    ]

    # 3) Forward OIS ECB dates
    # Add helpers for the third part of the curve from selected Forward OIS ECB
    estr_helpers += [
        ql.DatedOISRateHelper(
            row['Start_Period'], row['End_Period'], ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), estr
        )
        for index, row in forwards_ois_ecb_data.iterrows()
    ]

    # 4) OIS post-ECB
    ois_post_ecb = ois_data[ois_data.index > 7] # Watch for overlapping with forward OIS ECB
    estr_helpers += [
        ql.OISRateHelper(
            2, row['Period'], ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), estr
        )
        for index, row in ois_post_ecb.iterrows()
```

```python
    ]

    # DEBUG: Print details of all helpers
    DEBUG_ESTR = True
    if DEBUG_ESTR:
        print("€STR Helpers:")
        for helper in estr_helpers:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            pillar_date = helper.pillarDate()
            print(f"Type: {helper_type}, Start Date: {start_date},
                Maturity Date: {maturity_date}, Quote: {quote},
                Pillar Date: {pillar_date}")

    # Build the €STR discount curve
    estr_curve = ql.PiecewiseLogCubicDiscount(0, ql.TARGET(),
        estr_helpers, ql.Actual365Fixed())
    estr_curve.enableExtrapolation() # Enable extrapolation for
        dates beyond the available data

    # Calculate discount factors for specified dates
    dates = [settle_date + ql.Period(i, ql.Months) for i in range
        (0, 50 * 12 + 1)]

    # Convert dates to datetime format
    dates_dt = [_to_datetime(date) for date in dates]
    estr_zero_rates = [
        estr_curve.zeroRate(date, ql.Actual365Fixed(), ql.
            Continuous).rate()
        for date in dates
    ]

    # Create the €STR yield curve graph and display the €STR curve instruments summary
    if graph:
        plt.figure(figsize=(width, height))
        estr_zero_rates_percent = [rate * 100  for rate in estr_zero_rates]
        plt.plot(dates_dt, estr_zero_rates_percent, label='€STR Zero Rates', linewidth=2)
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y:.2f}'))
        plt.xlim(dates_dt[0], dates_dt[-1])
        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Zero Rate (%)', fontsize=12)
        plt.title('€STR Zero Curve', fontsize=14, pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.5)
        plt.tight_layout()
        plt.show()

        # Create a DataFrame to display the €STR curve instruments summary
        combined_data_estr = pd.concat([
            selected_depo_data.assign(Instrument='Deposit'),
            ois_pre_ecb.reset_index(drop=True).assign(Instrument='OIS'),
            forwards_ois_ecb_data.assign(Instrument='Forward OIS
               ECB'),
            ois_post_ecb.reset_index(drop=True).assign(Instrument='
               OIS')
        ]).reset_index(drop=True)  # Reset the index here


        # Set Tenor based on available information
        combined_data_estr['Tenor'] = combined_data_estr['Maturity'
           ].combine_first(pd.Series(['Dated'] * len(
           combined_data_estr)))


        # Extract start_date and maturity_date from helpers
        start_dates = []
        maturity_dates = []

        for helper in estr_helpers:
            start_dates.append(helper.earliestDate())
            maturity_dates.append(helper.maturityDate())

        combined_data_estr['Start Date'] = start_dates
        combined_data_estr['Maturity Date'] = maturity_dates

        # Reorder columns
        estr_data_table = combined_data_estr[['Instrument', 'Tenor'
           , 'Start Date', 'Maturity Date', 'Bid', 'Ask', 'Mid']]

        # Round values to three decimal places
        estr_data_table = estr_data_table.round(3)

        # Display the DataFrame
        print("€STR Curve Instruments Summary:")
```

```
        display(estr_data_table)


    return estr_curve, estr_helpers, estr_zero_rates

estr_curve, estr_helpers, estr_zero_rates = estr_curve_builder()
```

.4 €STR Short-Term Rate Analysis

```
# To simplify analysis, we turn to flat forward rates instead of
   log-cubic discounts
estr_curve_ff = ql.PiecewiseFlatForward(
    0, ql.TARGET(), estr_helpers, ql.Actual365Fixed()
)
estr_curve_ff.enableExtrapolation()

# Restrict the plot to the first 12 months
end = ql.TARGET().advance(settle_date, ql.Period(12, ql.Months))

# Calculate the forward rates for the first 12 months
dates = [
    ql.Date(serial)
    for serial in range(settle_date.serialNumber(), end.
       serialNumber() + 1)
]

# Convert QuantLib dates to datetime
dates_dt = [d.to_date() for d in dates]

rates_ff = [
    estr_curve_ff.forwardRate(
        d, ql.TARGET().advance(d, 1, ql.Days), ql.Actual360(), ql.
            Simple
    ).rate()
    for d in dates
]




# Convert QuantLib dates in 'Start_Period' to datetime objects
forwards_ois_ecb_data['End_Period_dt'] = forwards_ois_ecb_data['
    End_Period'].apply(_to_datetime)

# Convert ECB forward dates to datetime
ecb_forward_dates = forwards_ois_ecb_data['End_Period_dt'].tolist()

# Extract ECB forward rates from the 'Mid' column
ecb_forward_rates = forwards_ois_ecb_data['Mid'].values/100

plt.figure(figsize=(width, height))
rates_ff_percent = [rate * 100 for rate in rates_ff]
ecb_forward_rates_percent = [rate * 100 for rate in
    ecb_forward_rates]

plt.plot(dates_dt, rates_ff_percent, label='€STR Forward Overnight
    Rates', color='blue', linestyle='-', linewidth=1.5)
plt.scatter(ecb_forward_dates, ecb_forward_rates_percent, label='
    ECB Forward Rates', color='red', marker='o', s=30)

plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%d-%m-%Y'
    ))
```

```
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y:.2f}'))
plt.gca().tick_params(axis='x', labelsize=8, rotation=45)

plt.xlim(dates_dt[0], dates_dt[-1])
plt.xlabel('Date', fontsize=12)
plt.ylabel('Forward Rate (%)', fontsize=12)
plt.title('€STR Forward Overnight and ECB Forward OIS Rates (First 12 Months)', fontsize=14, pad=15)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=True, fancybox=True)
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.7)
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.5)

plt.tight_layout()
plt.show()
```

.5 Construction of the Euribor 3M without Synthetic Deposits

```
def euribor_curve_builder_no_synth_depo(
    estr_curve=estr_curve,
    fra_data3M=fra_data3M,
    futures_data3M=futures_data3M,
    swap_data3M=swap_data3M,
    futures_maturities_chosen=futures_maturities_chosen,
    swap_maturities_chosen=swap_maturities_chosen,
    dual_curve_approach=True,
    graph=True):

```

```
    # Define the 3-month Euribor instance
    euribor3m = ql.Euribor3M()

    # 1) 0x3 FRA
    # First pillar of the EURIBOR curve: the 0x3M FRA
    first_pillar_3m = fra_data3M[fra_data3M['Maturity'] == '0X3']

    fra_0x3_3m_helper = []
    fra_0x3_3m_helper.append(ql.FraRateHelper(ql.QuoteHandle(ql.
       SimpleQuote(first_pillar_3m['Mid'].values[0] / 100)),
                                              first_pillar_3m['Start_Period'
                                                 ].values[0], euribor3m))


    # 2) Futures
    # Add futures for the second part of the curve from selected
       FRA
    future_helpers = []
    selected_data = futures_data3M[futures_data3M['Maturity'].isin(
       futures_maturities_chosen)]
    for index, row in selected_data.iterrows():
        future_helpers.append(
            ql.FuturesRateHelper(
                ql.QuoteHandle(ql.SimpleQuote((row['Mid']))),
                row['Start_Date'],
                euribor3m,
                ql.QuoteHandle(),
            )
        )

    # Use €STR as the discount curve for the dual curve approach
    discount_curve = ql.YieldTermStructureHandle(estr_curve)


    # 3) Swaps
    # Add helpers for the third part of the curve from selected swaps
    swap_helpers = []
    selected_data = swap_data3M[swap_data3M['Maturity'].isin(swap_maturities_chosen)]

    if dual_curve_approach:
        for index, row in selected_data.iterrows():
            swap_helpers.append(ql.SwapRateHelper(
                ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), row['Period'], ql.TARGET(), ql.Annual, ql.Unadjusted,
                ql.Thirty360(ql.Thirty360.BondBasis), euribor3m, ql.QuoteHandle(), ql.Period(0, ql.Days), discount_curve))
    else:
        for index, row in selected_data.iterrows():
            swap_helpers.append(ql.SwapRateHelper(
                ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), row['Period'], ql.TARGET(), ql.Annual, ql.Unadjusted,
                ql.Thirty360(ql.Thirty360.BondBasis), euribor3m, ql.QuoteHandle(), ql.Period(0, ql.Days)))

    # Debug: Print details of all EURIBOR helpers
    DEBUG_EUR = False
    if DEBUG_EUR:
        for helper in fra_0x3_3m_helper + future_helpers + swap_helpers:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Type: {helper_type}, Start Date: {start_date}, Maturity Date: {maturity_date}, Quote: {quote}")

    # Build the zero curve
    euribor3m_curve_no_synth_depo = ql.PiecewiseLogCubicDiscount(
        2, ql.TARGET(), fra_0x3_3m_helper + future_helpers + swap_helpers, ql.Actual365Fixed()
    )
    euribor3m_curve_no_synth_depo.enableExtrapolation()

    # Build the forward curve
    spot_date = euribor3m_curve_no_synth_depo.referenceDate()  # Set the evaluation date as the spot date
    dates = [spot_date + ql.Period(i, ql.Months) for i in range(0, 50 * 12 + 1)]
    dates_dt = [_to_datetime(date) for date in dates]

    euribor3m_f_curve_no_synth_depo = [
        euribor3m_curve_no_synth_depo.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    # Plot the forward EURIBOR curve
    if graph:
        plt.figure(figsize=(width, height))
        euribor3m_f_curve_no_synth_depo_percent = [rate * 100 for rate in euribor3m_f_curve_no_synth_depo]
        plt.plot(dates_dt, euribor3m_f_curve_no_synth_depo_percent, label='Forward Rates', linewidth=1.5)
```

```python
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y:.2f}'))
        plt.xlim([dates_dt[0], dates_dt[-1]])
        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Forward Rate (%)', fontsize=12)
        plt.title('Euribor 3M Forward Curve Without Synthetic Deposits', fontsize=14, pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.5)
        plt.tight_layout()
        plt.show()


    return euribor3m_curve_no_synth_depo, euribor3m_f_curve_no_synth_depo, fra_0x3_3m_helper, future_helpers, swap_helpers, spot_date, euribor3m

# Build the Euribor curve with no synthetic deposits
euribor3m_curve_no_synth_depo, euribor3m_f_curve_no_synth_depo, fra_0x3_3m_helper, future_helpers, swap_helpers_dual, spot_date, euribor3m = euribor_curve_builder_no_synth_depo()
```

.6 Calculation of Basis and Creation of Synthetic Deposits

```python
def synthetic_deposit_builder(
    euribor3m_curve_no_synth_depo=euribor3m_curve_no_synth_depo,
    estr_curve=estr_curve,
    spot_date=spot_date,
    print_alpha=True):


# ATTENTION: since we are using a 0x3 FRA -> t1 = 0, so forward rate is equal to the zero rate.
    # Bootstrap the 0X3-month rate from the existing 3-month EURIBOR curve
    euribor3m_0X3M_rate = euribor3m_curve_no_synth_depo.forwardRate(
        spot_date, ql.TARGET().advance(spot_date, 3, ql.Months), ql.Actual360(), ql.Simple).rate()

    # Bootstrap the 0X3-month rate from the €STR curve
    estr_0X3M_rate = estr_curve.forwardRate(
        spot_date, ql.TARGET().advance(spot_date, 3, ql.Months), ql.Actual360(), ql.Simple).rate()

    # Calculate the spread (alpha) between the 3-month forward EURIBOR rate and the 3-month forward €STR rate
    alpha = euribor3m_0X3M_rate - estr_0X3M_rate

    if print_alpha:
        print(f"Alpha: {alpha}")

    # Construct helpers for synthetic deposits
    synth_deposit_helper_ON_to_2M = []

    # Define a list of tuples representing the maturities for which synthetic deposits will be created
    # Each tuple contains a number (n) and a unit of time (Days, Weeks, Months)
    for n, units in [
        (1, ql.Days),
        (1, ql.Weeks),
        (2, ql.Weeks),
        (3, ql.Weeks),
        (1, ql.Months),
        (2, ql.Months),
    ]:
        # Calculate the synth deposits rates from the €STR curve for the given duration and adjust the Euribor rate by adding the previously calculated spread (alpha)
        estr_0Xx_3M_rate = estr_curve.forwardRate(
            spot_date, ql.TARGET().advance(spot_date, n, units), ql.Actual360(), ql.Simple
        ).rate()

        synth_euribor3m_deposit_rate = estr_0Xx_3M_rate + alpha

        # Create a synthetic deposit helper with the calculated forward rate and add it to the list
        synth_deposit_helper_ON_to_2M.append(
            ql.DepositRateHelper(
                ql.QuoteHandle(ql.SimpleQuote(synth_euribor3m_deposit_rate)),  # Adjusted forward rate
                ql.Period(n, units),  # Duration for the synthetic deposit
                2,  # Settlement days
                ql.TARGET(),  # Calendar
                ql.Following,  # Business day convention
                False,  # End of month flag
```

```python
                    ql.Actual360(),  # Day count convention
                )
            )

    # DEBUG: Print details of each synthetic deposit helper
    DEBUG_SYNTH = False
    if DEBUG_SYNTH:
        print("Synthetic Deposit Helpers:")
        for helper in synth_deposit_helper_ON_to_2M:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Type: {helper_type}, Start Date: {start_date},
                Maturity Date: {maturity_date}, Quote: {quote}")

    return synth_deposit_helper_ON_to_2M

# Build synthetic deposit helpers
synth_deposit_helper_ON_to_2M = synthetic_deposit_builder()
```

.7 Construction of the Euribor 3M Forward Curve with Synthetic Deposits

```python
def euribor_curve_builder_with_synth_depo(
    synth_deposit_helper_ON_to_2M=synth_deposit_helper_ON_to_2M,
    fra_0x3_3m_helper=fra_0x3_3m_helper,
    future_helpers=future_helpers,
    swap_helpers_dual=swap_helpers_dual,
    graph=True):

```

```python
    # Construct the corrected EURIBOR curve using synthetic deposit
        helpers
    euribor3m_curve_with_synth_depo = ql.PiecewiseLogCubicDiscount(
        2, ql.TARGET(), synth_deposit_helper_ON_to_2M +
           fra_0x3_3m_helper + future_helpers + swap_helpers_dual,
           ql.Actual365Fixed()
    )
    euribor3m_curve_with_synth_depo.enableExtrapolation()

    # DEBUG: Print details of the corrected EURIBOR curve helpers
    DEBUG_EU_CORR = False
    if DEBUG_EU_CORR:
        print("Corrected Euribor Helpers:")
        for helper in synth_deposit_helper_ON_to_2M +
           fra_0x3_3m_helper + future_helpers + swap_helpers_dual:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Helper Type: {helper_type}, Start Date: {
               start_date}, Maturity Date: {maturity_date}, Quote:
               {quote}")

    dates = [spot_date + ql.Period(i, ql.Months) for i in range(0,
       50 * 12 + 1)]
    dates_dt = [_to_datetime(date) for date in dates]

    # Construct the corrected forward EURIBOR curve
    euribor3m_f_curve = [
        euribor3m_curve_with_synth_depo.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
```

```
]


# Display the 3M-Euribor curve instruments summary
    -------------------------------------------------------------

synthetic_data_3m = None
if graph:
    # For synthetic deposits, we only have helper data, so we
       extract it from the helpers
    synthetic_data_3m = pd.DataFrame({
        'Instrument': ['Synthetic Deposit'] * len(
           synth_deposit_helper_ON_to_2M),
        'Maturity': ['ON', '1W', '2W', '3W', '1M', '2M'],
        'Start Date': [helper.earliestDate() for helper in
           synth_deposit_helper_ON_to_2M],
        'Maturity Date': [helper.maturityDate() for helper in
           synth_deposit_helper_ON_to_2M],
        'Bid': [''] * len(synth_deposit_helper_ON_to_2M),
        'Ask': [''] * len(synth_deposit_helper_ON_to_2M),
        'Mid': [helper.quote().value()*100 for helper in
           synth_deposit_helper_ON_to_2M]
    })

    # Combine with other instrument data
    combined_data_euribor3m = pd.concat([
        synthetic_data_3m,  # Include synthetic deposits
        fra_data3M[fra_data3M['Maturity'] == '0X3'].assign(
           Instrument='FRA'),
        futures_data3M[futures_data3M['Maturity'].isin(
           futures_maturities_chosen)].assign(Instrument='
           Futures'),
        swap_data3M[swap_data3M['Maturity'].isin(
           swap_maturities_chosen)].assign(Instrument='Swap')
    ]).reset_index(drop=True)  # Reset the index here


    # Set Tenor based on available information
    combined_data_euribor3m['Tenor'] = combined_data_euribor3m[
       'Maturity']

    # Extract start_date and maturity_date from all helpers
    start_dates = []
    maturity_dates = []

    for helper in synth_deposit_helper_ON_to_2M +
       fra_0x3_3m_helper + future_helpers + swap_helpers_dual:
        start_dates.append(helper.earliestDate())
        maturity_dates.append(helper.maturityDate())

    combined_data_euribor3m['Start Date'] = start_dates
    combined_data_euribor3m['Maturity Date'] = maturity_dates

    # Reorder columns
    euribor3m_data_table = combined_data_euribor3m[['Instrument
       ', 'Tenor', 'Start Date', 'Maturity Date', 'Bid', 'Ask',
        'Mid']]

    # Round values to three decimal places
    euribor3m_data_table = euribor3m_data_table.round(3)

    # Display the DataFrame
    print("3M-Euribor Curve Instruments Summary:")
    display(euribor3m_data_table)
#---------------------------------------------------------------

return euribor3m_curve_with_synth_depo, euribor3m_f_curve,
   synthetic_data_3m



euribor3m_curve_with_synth_depo, euribor3m_f_curve,
synthetic_data_3m = euribor_curve_builder_with_synth_depo()
```

.8 Comparing Forward Euribor Curves With and Without Synthetic Deposits

```



# Calculate dates for 50 years and first 6 months
dates = [spot_date + ql.Period(i, ql.Months) for i in range(0, 50 *
    12 + 1)]
dates_6m = [spot_date + ql.Period(i, ql.Months) for i in range(0, 6
    + 1)]
dates_dt = [_to_datetime(date) for date in dates]
dates_dt_6m = [_to_datetime(date) for date in dates_6m]

# Calculate forward rates without synthetic deposits for 50 years
euribor3m_f_curve_no_synth_depo = [
    euribor3m_curve_no_synth_depo.forwardRate(
        d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates
]

# Calculate forward rates with synthetic deposits for 50 years
```

```python
euribor3m_f_curve = [
    euribor3m_curve_with_synth_depo.forwardRate(
        d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates
]

# Calculate forward rates without synthetic deposits for the first
   6 months
euribor3m_f_curve_no_synth_depo_first6m = [
    euribor3m_curve_no_synth_depo.forwardRate(
        d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates_6m
]

# Calculate forward rates with synthetic deposits for the first 6
   months
euribor3m_f_curve_first6m = [
    euribor3m_curve_with_synth_depo.forwardRate(
        d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates_6m
]

# Plot: Overlapping Curves for the First 6 Months
# Set the plot style for a clean aesthetic
plt.style.use('seaborn-whitegrid')

# Create the figure
plt.figure(figsize=(width, height))
euribor3m_f_curve_no_synth_depo_first6m_percent = [rate * 100 for
   rate in euribor3m_f_curve_no_synth_depo_first6m]
```

```
euribor3m_f_curve_first6m_percent = [rate * 100 for rate in
    euribor3m_f_curve_first6m]
plt.plot(dates_dt_6m,
    euribor3m_f_curve_no_synth_depo_first6m_percent, label='Without
    Synth Depos', color='blue', linestyle='--', linewidth=1.5)
plt.plot(dates_dt_6m, euribor3m_f_curve_first6m_percent, label='
    With Synth Depos', color='orange', linestyle='-', linewidth=1.5)
plt.title('Comparison of Euribor 3M Forward Curves with and without
     Synthetic Deposits (First 6 Months)', fontsize=14, pad=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Forward Rate (%)', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=1))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha
    =0.7)
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha
    =0.5)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=
    True, fancybox=True)

# Annotate differences in basis points every 1 month
interval = 1  # Annotate every 1 month
for i in range(0, len(dates_dt_6m), interval):
    date = dates_dt_6m[i]
    rate_no_synth = euribor3m_f_curve_no_synth_depo_first6m_percent
        [i]
    rate_with_synth = euribor3m_f_curve_first6m_percent[i]
    diff = rate_no_synth - rate_with_synth
    diff_bps = diff * 100  # Convert difference to basis points (
        already in percentage)
    plt.annotate(f'{diff_bps:.1f} bps',
                     (date, rate_with_synth),
                     textcoords="offset points",
                     xytext=(0, 10),
                     ha='center',
                     fontsize=8,
                     color='red')

# Add markers to indicate annotation points
plt.plot(dates_dt_6m[::interval], euribor3m_f_curve_first6m_percent
    [::interval], 'o', color='red', markersize=4)

# Adjust layout to prevent overlap
plt.tight_layout()

# Display the third plot
plt.show()
```

.9 Comparison of Dual Curve and Single Euribor 3M Forward Curves

```
#Set the evaluation date for the curves
dates = [spot_date + ql.Period(i, ql.Months) for i in range(0, 50 *
     12 + 1)]
dates_dt = [_to_datetime(date) for date in dates]

# Construct the Euribor zero curve
euribor3m_zero_rates = [
     euribor3m_curve_with_synth_depo.zeroRate(date, ql.
        Actual365Fixed(), ql.Continuous).rate()
     for date in dates
]

```

```
# Construct the Euribor curve with single curve approach
_,_,_,_, swap_helpers_single,*_ =
   euribor_curve_builder_no_synth_depo(
                                    dual_curve_approach=False,
                                    graph=False)

euribor3m_curve_single = ql.PiecewiseLogCubicDiscount(
    2, ql.TARGET(), synth_deposit_helper_ON_to_2M +
       fra_0x3_3m_helper + future_helpers + swap_helpers_single, ql
       .Actual365Fixed()
)
euribor3m_curve_single.enableExtrapolation()

# Calculate zero rates for the single curve approach
euribor3m_zero_rates_single = [
    euribor3m_curve_single.zeroRate(date, ql.Actual365Fixed(), ql.
       Continuous).rate()
    for date in dates
]

# Construct the forward curve for the single curve approach
euribor3m_f_curve_single = [
    euribor3m_curve_single.forwardRate(
        d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates
]

# First plot: €STR Zero Rates + euribor3m Zero Curves (Single vs
   Dual Approach)

# Create a new figure
plt.figure(figsize=(width, height))
```

```

# Calculate the values in percentage
estr_zero_rates_percent = [rate * 100 for rate in estr_zero_rates]
euribor3m_zero_rates_percent = [rate * 100 for rate in
    euribor3m_zero_rates]
euribor3m_zero_rates_single_percent = [rate * 100 for rate in
    euribor3m_zero_rates_single]

# Plot
plt.plot(dates_dt, estr_zero_rates_percent, label='€STR Zero Rates'
    , color='green', linewidth=1.5)
plt.plot(dates_dt, euribor3m_zero_rates_percent, label='Euribor 3M
    Zero Curve (Dual Curve Approach)', color='blue', linewidth=1.5)
plt.plot(dates_dt, euribor3m_zero_rates_single_percent, label='
    Euribor 3M Zero Curve (Single Curve Approach)', color='orange',
    linewidth=1.5)
plt.title('Euribor 3M Zero Curve: Single vs. Dual Curve Approaches'
    , fontsize=14, pad=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Zero Rate (%)', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha
    =0.7)
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha
    =0.5)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=
    True, fancybox=True)

# Adjust layout to prevent overlap
plt.tight_layout()
```

```python

# Display the plot
plt.show()

# Second plot: Forward Curves for Single vs Dual Approach with
    Annotations

# Create a new figure
plt.figure(figsize=(width, height))

# Calculate the values in percentage
euribor3m_f_curve_percent = [rate * 100 for rate in
    euribor3m_f_curve]
euribor3m_f_curve_single_percent = [rate * 100 for rate in
    euribor3m_f_curve_single]

# Plot
plt.plot(dates_dt, euribor3m_f_curve_percent, label='Dual Curve
    Approach', color='blue', linewidth=1.5)
plt.plot(dates_dt, euribor3m_f_curve_single_percent, label='Single
    Curve Approach', color='orange', linestyle='--', linewidth=1.5)
plt.title('Euribor 3M Forward Curve Comparison: Single vs. Dual
    Curve Approaches', fontsize=14, pad=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Forward Rate (%)', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))

# Add annotations
interval = 36  # Annotate every 36 months (3 years)
for i in range(0, len(dates_dt), interval):
    date = dates_dt[i]
    rate_dual = euribor3m_f_curve_percent[i]
    rate_single = euribor3m_f_curve_single_percent[i]
    diff = rate_single - rate_dual
    diff_bps = diff * 100  # Convert difference to basis points (
        already in percentage)
    plt.annotate(f'{diff_bps:.1f} bps',
                 (date, rate_single),
                 textcoords="offset points",
                 xytext=(0, 10),
                 ha='center',
                 fontsize=8,
                 color='red')

# Add markers to indicate annotation points
plt.plot(dates_dt[::interval], euribor3m_f_curve_single_percent[::
    interval], 'o', color='red', markersize=4)

# Add grid, legend, and display the plot
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha
    =0.7)
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha
    =0.5)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=
    True, fancybox=True)

# Adjust layout to prevent overlap
plt.tight_layout()

# Display the plot
plt.show()
```

.10 Construction of the Euribor 6M Forward Curve and Comparison with the Euribor 3M Forward Curve - The Importance of Being Tenor Consistent

```python
euribor6m = ql.Euribor6M()
# 1 & 2) FRA 0x6 + other FRAs

fra_helpers_6m = []
for index, row in fra_data6M.iterrows():
    fra_helpers_6m.append(ql.FraRateHelper(ql.QuoteHandle(ql.
       SimpleQuote(row['Mid'] / 100)),
                                             row['Start_Period'], euribor6m
                                                ))

discount_curve = ql.YieldTermStructureHandle(estr_curve)

# 3) SWAP
# Aggiungo gli helpers per la terza parte della curva dai SWAP
swap_helpers_6m = []
selected_data = swap_data6M[swap_data6M['Maturity'].isin(
    swap_maturities_chosen)]
for index, row in selected_data.iterrows():
    swap_helpers_6m.append(ql.SwapRateHelper(
        ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)), row['
           Period'], ql.TARGET(), ql.Annual, ql.Unadjusted,
        ql.Thirty360(ql.Thirty360.BondBasis), euribor6m, ql.
           QuoteHandle(), ql.Period(0, ql.Days), discount_curve))

# Build the zero curve
euribor6m_curve_no_synth_depo = ql.PiecewiseLogCubicDiscount(
    2, ql.TARGET(), fra_helpers_6m + swap_helpers_6m, ql.
        Actual365Fixed()
)
euribor6m_curve_no_synth_depo.enableExtrapolation()

#======================build and add synthetic deposits
    =====================

# Set the spot date to one day after the settlement date (fixing
    convention)
d = ql.TARGET().advance(spot_date, 1, ql.Days)

# Bootsrap del tasso forward a 6 mesi dalla curva EURIBOR a 6 mesi
    preesistente
euribor6m_forward_curve_no_synth_depo =
    euribor6m_curve_no_synth_depo.forwardRate(
    d, ql.TARGET().advance(d, 6, ql.Months), ql.Actual360(), ql.
        Simple).rate()

# Bootstrap the forward 6-month rate from the existing 6-month
    EURIBOR curve
euribor6m_0X6M_rate = euribor6m_curve_no_synth_depo.forwardRate(
    spot_date, ql.TARGET().advance(spot_date, 6, ql.Months), ql.
        Actual360(), ql.Simple).rate()

# Bootstrap the forward 6-month rate from the €STR curve
estr_0X6M_rate = estr_curve.forwardRate(
    d, ql.TARGET().advance(d, 6, ql.Months), ql.Actual360(), ql.
        Simple).rate()

# Calculate the spread (alpha) between the 6-month forward EURIBOR
    rate and the 3-month forward €STR rate
alpha = euribor6m_0X6M_rate - estr_0X6M_rate
```

```python

# Construct helpers for synthetic deposits
synth_deposit_helper_ON_to_5M = []

for n, units in [
    (1, ql.Days),
    (1, ql.Weeks),
    (2, ql.Weeks),
    (3, ql.Weeks),
    (1, ql.Months),
    (2, ql.Months),
    (3, ql.Months),
    (4, ql.Months),
    (5, ql.Months),
]:
        # Calculate the forward rate from the €STR curve for the
           given duration and adjust the Euribor forward rate by
           adding the previously calculated spread (alpha)
    estr_0Xx_6M_rate = estr_curve.forwardRate(
        spot_date, ql.TARGET().advance(spot_date, n, units), ql.
           Actual360(), ql.Simple
    ).rate()

    synth_euribor6M_0Xx__deposit_rate = estr_0Xx_6M_rate + alpha

    # Create a synthetic deposit helper with the calculated forward
        rate and add it to the list
    synth_deposit_helper_ON_to_5M.append(
        ql.DepositRateHelper(
            ql.QuoteHandle(ql.SimpleQuote(
               synth_euribor6M_0Xx__deposit_rate)),  # Adjusted
               forward rate
            ql.Period(n, units),  # Duration for the synthetic
               deposit
            2,  # Settlement days
            ql.TARGET(),  # Calendar
            ql.Following,  # Business day convention
            False,  # End of month flag
            ql.Actual360(),  # Day count convention
        )
    )

euribor6m_curve = ql.PiecewiseLogCubicDiscount(
    2, ql.TARGET(), synth_deposit_helper_ON_to_5M + fra_helpers_6m
       + swap_helpers_6m, ql.Actual365Fixed()
)
euribor6m_curve.enableExtrapolation()


# DEBUG: Print details of each synthetic deposit helper
DEBUG_6M = False
if DEBUG_6M:
    print("Euribor 6M Helpers:")
    for helper in synth_deposit_helper_ON_to_5M + fra_helpers_6m +
       swap_helpers_6m:
        helper_type = type(helper).__name__
        start_date = helper.earliestDate()
        maturity_date = helper.maturityDate()
        quote = helper.quote().value()
        print(f"Type: {helper_type}, Start Date: {start_date},
           Maturity Date: {maturity_date}, Quote: {quote}")



# Build the forward curve
```

```
euribor6m_f_curve = [
    euribor6m_curve.forwardRate(
        d, euribor6m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates
]

# Calculate the forward rates using the wrong approach
euribor6m_f_curve_wrong = [
    euribor3m_curve_with_synth_depo.forwardRate(
        d, euribor6m.maturityDate(d), ql.Actual360(), ql.Simple
    ).rate()
    for d in dates
]



#---------------------------------------------------------------
# Display the instruments in the 6M-Euribor curve



synthetic_data_6m = pd.DataFrame({
    'Instrument': ['Synthetic Deposit'] * len(
       synth_deposit_helper_ON_to_5M),
    'Maturity': ['ON', '1W', '2W', '3W', '1M', '2M', '3M', '4M', '5
       M'],
    'Start Date': [helper.earliestDate() for helper in
       synth_deposit_helper_ON_to_5M],
    'Maturity Date': [helper.maturityDate() for helper in
       synth_deposit_helper_ON_to_5M],
    'Bid': [''] * len(synth_deposit_helper_ON_to_5M),
    'Ask': [''] * len(synth_deposit_helper_ON_to_5M),
    'Mid': [helper.quote().value()*100 for helper in
       synth_deposit_helper_ON_to_5M]
})


# Combine with other instrument data
combined_data_euribor6m = pd.concat([
    synthetic_data_6m,  # Include synthetic deposits
    fra_data6M.assign(Instrument='FRA'),
    swap_data6M[swap_data6M['Maturity'].isin(swap_maturities_chosen
        )].assign(Instrument='Swap')
]).reset_index(drop=True)  # Reset the index here


# Set Tenor based on available information
combined_data_euribor6m['Tenor'] = combined_data_euribor6m['
    Maturity']

# Extract start_date and maturity_date from all helpers
start_dates = []
maturity_dates = []

for helper in synth_deposit_helper_ON_to_5M + fra_helpers_6m +
    swap_helpers_6m:
    start_dates.append(helper.earliestDate())
    maturity_dates.append(helper.maturityDate())

combined_data_euribor6m['Start Date'] = start_dates
combined_data_euribor6m['Maturity Date'] = maturity_dates

# Reorder columns
data_table_euribor6m = combined_data_euribor6m[['Instrument', '
    Tenor', 'Start Date', 'Maturity Date', 'Bid', 'Ask', 'Mid']]

# Round values to three decimal places
```

```python
data_table_euribor6m = data_table_euribor6m.round(3)

# Display the DataFrame
print("Euribor 6M Curve Instruments Summary:")
display(data_table_euribor6m)
#-------------------------------------------------------------

#Plots

# Set up the plot style
plt.style.use('seaborn-whitegrid')

# Calculate the values in percentage
euribor3m_f_curve_percent = [rate * 100 for rate in
    euribor3m_f_curve]
euribor6m_f_curve_percent = [rate * 100 for rate in
    euribor6m_f_curve]
euribor6m_f_curve_wrong_percent = [rate * 100 for rate in
    euribor6m_f_curve_wrong]

# Create figure for the first plot
plt.figure(figsize=(width, height))

# Plot 1: Euribor 3M vs 6M Forward Curves
plt.plot(dates_dt, euribor3m_f_curve_percent, label='Euribor 3M
    Forward Curve', color='blue', linewidth=1.5)
plt.plot(dates_dt, euribor6m_f_curve_percent, label='Euribor 6M
    Forward Curve', color='orange', linewidth=1.5)
plt.title('Comparison of Euribor 3M vs. 6M Forward Curves',
    fontsize=14, pad=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Forward Rate (%)', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha
    =0.7)
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha
    =0.5)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=
    True, fancybox=True)

# Display the first plot
plt.tight_layout()
plt.show()

# Create figure for the second plot
plt.figure(figsize=(width, height))

# Plot 2: Euribor 6M Forward Curve vs. Incorrect Approach
plt.plot(dates_dt, euribor6m_f_curve_percent, label='Euribor 6M
    Forward Curve', color='orange', linewidth=1.5)
plt.plot(dates_dt, euribor6m_f_curve_wrong_percent, label='Euribor
    6M Forward Curve - Incorrect Approach', color='red', linestyle='
    --', linewidth=1.5)
plt.title('Comparison of Euribor 6M Forward Curves: Correct vs.
    Incorrect Approach', fontsize=14, pad=15)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Forward Rate (%)', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha
    =0.7)
```

```
plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha
    =0.5)
plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=
    True, fancybox=True)

# Annotate differences in basis points between the correct and
    wrong 6M forward curves
interval = 36  # Annotate every 36 months (3 years)
for i in range(0, len(dates_dt), interval):
    date = dates_dt[i]
    rate_correct = euribor6m_f_curve_percent[i]
    rate_wrong = euribor6m_f_curve_wrong_percent[i]
    diff = rate_wrong - rate_correct
    diff_bps = diff * 100  # Convert difference to basis points (
        already in percentage)
    plt.annotate(f'{diff_bps:.1f} bps',
                 (date, rate_wrong),
                 textcoords="offset points",
                 xytext=(0, 10),
                 ha='center',
                 fontsize=8,
                 color='black')

# Add markers to indicate annotation points
plt.plot(dates_dt[::interval], euribor6m_f_curve_wrong_percent[::
    interval], 'o', color='black', markersize=4)

# Display the second plot
plt.tight_layout()
plt.show()
```

.11 Constructing and Evaluating Euribor 3M Forward Curves via Diverse Bootstrapping Instrument Configurations

```python
def different_i_b_sets_curves_builder(
    synth_deposit_helper_ON_to_2M=synth_deposit_helper_ON_to_2M,
    fra_0x3_3m_helper=fra_0x3_3m_helper,
    swap_helpers_dual=swap_helpers_dual,
    estr_curve=estr_curve,
    euribor3m_curve_with_synth_depo=euribor3m_curve_with_synth_depo,
    futures_data3M=futures_data3M,
    swap_data3M=swap_data3M,
    fra_data3M=fra_data3M,
    synthetic_data_3m=synthetic_data_3m,
    graph=True,
    graph_comparison=True):

    ## Combination 2: Greater focus on swaps ==============================================================

    futures_maturities_chosen_combo2 = ['DEC4', 'MAR5', 'JUN5'] # We cannot reduce the maturities of the futures further or the algorithm will not converge
    focus_swap_maturities_chosen = ['18M','2Y', '3Y', '4Y', '5Y', '6Y', '7Y', '8Y', '9Y', '10Y', '12Y', '15Y', '20Y', '25Y', '30Y', '40Y', '50Y']

    # The 0x3M FRA and SYNTHETIC DEPOSITS are the same as in combination 1

    # FUTURES
    selected_data = futures_data3M[futures_data3M['Maturity'].isin(
        futures_maturities_chosen_combo2)]
    futures_helpers_combo2 = []
    for index, row in selected_data.iterrows():
        futures_helpers_combo2.append(
            ql.FuturesRateHelper(
                ql.QuoteHandle(ql.SimpleQuote((row['Mid']))),
                row['Start_Date'],
                euribor3m,
                ql.QuoteHandle(),
            )
        )

    # SWAPS
    discount_curve = ql.YieldTermStructureHandle(estr_curve)
    focus_swap_helpers = []
    selected_data = swap_data3M[swap_data3M['Maturity'].isin(
        focus_swap_maturities_chosen)]
    for index, row in selected_data.iterrows():
        focus_swap_helpers.append(ql.SwapRateHelper(
            ql.QuoteHandle(ql.SimpleQuote(row['Mid'] / 100)),
            row['Period'],
            ql.TARGET(),
            ql.Annual,
            ql.Unadjusted,
            ql.Thirty360(ql.Thirty360.BondBasis),
            euribor3m,
            ql.QuoteHandle(),
            ql.Period(0, ql.Days),
            discount_curve))

    # DEBUG: Print details of EURIBOR combo 2 helpers
    DEBUG_EU_COMBO2 = False
```

```python
    if DEBUG_EU_COMBO2:
        print("Euribor combo 2 helpers:")
        for helper in synth_deposit_helper_ON_to_2M + fra_0x3_3m_helper + futures_helpers_combo2 + focus_swap_helpers:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Type: {helper_type}, Start Date: {start_date}, Maturity Date: {maturity_date}, Quote: {quote}")

    # Build the curve
    all_helpers_combo2 = synth_deposit_helper_ON_to_2M + fra_0x3_3m_helper + futures_helpers_combo2 + focus_swap_helpers

    euribor3m_curve_combo2 = ql.PiecewiseLogCubicDiscount(
        2, ql.TARGET(), all_helpers_combo2, ql.Actual365Fixed()
    )
    euribor3m_curve_combo2.enableExtrapolation()

    # Build the forward curve
    dates = [spot_date + ql.Period(i, ql.Months) for i in range(0, 50 * 12 + 1)]
    dates_dt = [_to_datetime(date) for date in dates]

    euribor3m_f_curve_combo2 = [
        euribor3m_curve_combo2.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    # Plot the forward curve and display the instruments in the
        combo
        2-------------------------------------------------------------

    if graph:
        plt.figure(figsize=(width, height))
        euribor3m_f_curve_combo2_percent = [rate * 100 for rate in
            euribor3m_f_curve_combo2]

        # Plot the data
        plt.plot(dates_dt, euribor3m_f_curve_combo2_percent, label=
            'Forward Rates', linewidth=1.5)
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%
            Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y,
             _: f'{y:.2f}'))
        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Forward Rate (%)', fontsize=12)
        plt.title('3M Euribor Forward Curve: Synth Deposits + 3M
            Deposit + Futures + More Focus on Swaps (Combo 2)',
            fontsize=14, pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True,
            shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth
            =0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5,
             alpha=0.5)
        plt.tight_layout()
        plt.show()

```

```python
        # Create a DataFrame to display the instruments in the combo 2
        combined_data_combo2 = pd.concat([
            synthetic_data_3m.assign(Instrument='Synthetic Deposit'),
            fra_data3M[fra_data3M['Maturity'] == '0X3'].assign(Instrument='FRA'),
            futures_data3M[futures_data3M['Maturity'].isin(futures_maturities_chosen_combo2)].assign(Instrument='Futures'),
            swap_data3M[swap_data3M['Maturity'].isin(focus_swap_maturities_chosen)].assign(Instrument='Swap'),
        ]).reset_index(drop=True)  # Reset the index here


        # Set Tenor based on available information
        combined_data_combo2['Tenor'] = combined_data_combo2['Maturity']

        # Extract start_date and maturity_date from all helpers
        start_dates = []
        maturity_dates = []

        for helper in synth_deposit_helper_ON_to_2M + fra_0x3_3m_helper + futures_helpers_combo2 + focus_swap_helpers:
            start_dates.append(helper.earliestDate())
            maturity_dates.append(helper.maturityDate())

        combined_data_combo2['Start Date'] = start_dates
        combined_data_combo2['Maturity Date'] = maturity_dates

        # Reorder columns
        data_table_combo2 = combined_data_combo2[['Instrument', '
           Tenor', 'Start Date', 'Maturity Date', 'Bid', 'Ask', '
           Mid']]

        # Round values to three decimal places
        data_table_combo2 = data_table_combo2.round(3)

        # Display the DataFrame
        print("3M-Euribor Curve Instruments Summary (Combo 2):")
        display(data_table_combo2)

    ## Combination 3: SYNTH DEPOSITIS, FRA and SWAPS
       ====================================================================


    # Synth depos, the 0x3M FRA and SWAPS are the same as in
       combination 1

    # FRA
    fra_helpers_combo3 = []
    for index, row in fra_data3M.iterrows():
        fra_helpers_combo3.append(ql.FraRateHelper(ql.QuoteHandle(
           ql.SimpleQuote(row['Mid'] / 100)),
                                             row['Start_Period'],
                                                euribor3m))

    DEBUG_EU_COMBO3 = False
    if DEBUG_EU_COMBO3:
        print("Euribor combo 3 helpers:")
        for helper in synth_deposit_helper_ON_to_2M +
           fra_helpers_combo3 + swap_helpers_dual:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Type: {helper_type}, Start Date: {start_date},
               Maturity Date: {maturity_date}, Quote: {quote}")

    # Build the curve
    all_helpers_combo3 = synth_deposit_helper_ON_to_2M +
       fra_helpers_combo3 + swap_helpers_dual

    euribor3m_curve_combo3 = ql.PiecewiseLogCubicDiscount(
        2, ql.TARGET(), all_helpers_combo3, ql.Actual365Fixed()
    )

    euribor3m_curve_combo3.enableExtrapolation()

    # Build the forward curve
    euribor3m_f_curve_combo3 = [
        euribor3m_curve_combo3.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    # Plot the forward curve and display the instruments
       -------------------------------------------------------------

    if graph:
        plt.figure(figsize=(width, height))
        euribor3m_f_curve_combo3_percent = [rate * 100 for rate in
           euribor3m_f_curve_combo3]
        plt.plot(dates_dt, euribor3m_f_curve_combo3_percent, label=
           'Forward Rates', linewidth=1.5)
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y:.2f}'))
        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Forward Rate (%)', fontsize=12)
        plt.title('Euribor 3M Forward Curve: Synth Deposits + 0X3M FRA + FRAs + Swaps (Combo 3)', fontsize=14, pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5, alpha=0.5)
        plt.tight_layout()
        plt.show()

        # Create a DataFrame to display the instruments in the combo 3
        combined_data_combo3 = pd.concat([
            synthetic_data_3m.assign(Instrument='Synthetic Deposit'),
            fra_data3M.assign(Instrument='FRA'),
            swap_data3M[swap_data3M['Maturity'].isin(swap_maturities_chosen)].assign(Instrument='Swap'),
        ]).reset_index(drop=True)  # Reset the index here

        # Set Tenor based on available information
        combined_data_combo3['Tenor'] = combined_data_combo3['Maturity']

        # Extract start_date and maturity_date from all helpers
        start_dates = []
        maturity_dates = []

        for helper in synth_deposit_helper_ON_to_2M + fra_helpers_combo3 + swap_helpers_dual:
            start_dates.append(helper.earliestDate())
            maturity_dates.append(helper.maturityDate())

        combined_data_combo3['Start Date'] = start_dates
        combined_data_combo3['Maturity Date'] = maturity_dates

        # Reorder columns
        data_table_combo3 = combined_data_combo3[['Instrument', 'Tenor', 'Start Date', 'Maturity Date', 'Bid', 'Ask', 'Mid']]

        # Round values to three decimal places
        data_table_combo3 = data_table_combo3.round(3)

        # Display the DataFrame
        print("Euribor 3M Curve Instruments Summary (Combo 3):")
        display(data_table_combo3)




    # Combination 4: FRA with focus on swaps =======================================================================

    fra3M_maturities_chosen_combo4 = ['0x3','1X4','2X5','3X6','4X7','5X8','6X9','7X10','8X11','9X12','10X13','12X15']

    # Synth depos and the 0x3 FRA are the same as in combination 1.
        Swasp are the same as in combination 2

    # FRA
    fra_helpers_combo4 = []
    selected_data = fra_data3M[fra_data3M['Maturity'].isin(
        fra3M_maturities_chosen_combo4)]
    for index, row in selected_data.iterrows():
        fra_helpers_combo4.append(ql.FraRateHelper(ql.QuoteHandle(
            ql.SimpleQuote(row['Mid'] / 100)),
                                                   row['Start_Period'],
                                                       euribor3m))

    DEBUG_EU_COMBO4 = False
    if DEBUG_EU_COMBO4:
        print("Euribor combo 4 helpers:")
        for helper in synth_deposit_helper_ON_to_2M +
            fra_helpers_combo4 + focus_swap_helpers:
            helper_type = type(helper).__name__
            start_date = helper.earliestDate()
            maturity_date = helper.maturityDate()
            quote = helper.quote().value()
            print(f"Type: {helper_type}, Start Date: {start_date},
                Maturity Date: {maturity_date}, Quote: {quote}")

    # Build the EURIBOR curve
    all_helpers_combo4 = synth_deposit_helper_ON_to_2M +
        fra_helpers_combo4 + focus_swap_helpers

    euribor3m_curve_combo4 = ql.PiecewiseLogCubicDiscount(
        2, ql.TARGET(), all_helpers_combo4, ql.Actual365Fixed()
    )

```

```python
    euribor3m_curve_combo4.enableExtrapolation()

    # Build the EURIBOR forward curve
    euribor3m_f_curve_combo4 = [
        euribor3m_curve_combo4.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    # Plot the forward curve and display the instruments --------------------------------------------------------------

    if graph:
        plt.figure(figsize=(width, height))
        euribor3m_f_curve_combo4_percent = [rate * 100 for rate in euribor3m_f_curve_combo4]
        plt.plot(dates_dt, euribor3m_f_curve_combo4_percent, label='Forward Rates', linewidth=1.5)
        plt.gca().xaxis.set_major_locator(mdates.YearLocator(3))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y:.2f}'))
        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Forward Rate (%)', fontsize=12)
        plt.title('Euribor 3M Forward Curve: Synth Deposits + 0x3M FRA + FRAs + More Focus on Swaps (Combo 4)', fontsize=14, pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True, shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth=0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5,
            alpha=0.5)
        plt.tight_layout()
        plt.show()

        # Create a DataFrame to display the instruments in the
           combo 4
        combined_data_combo4 = pd.concat([
            synthetic_data_3m.assign(Instrument='Synthetic Deposit'
               ),
            fra_data3M[fra_data3M['Maturity'].isin(
               fra3M_maturities_chosen_combo4)].assign(Instrument='
               FRA'),
            swap_data3M[swap_data3M['Maturity'].isin(
               focus_swap_maturities_chosen)].assign(Instrument='
               Swap'),
        ]).reset_index(drop=True)

        # Set Tenor based on available information
        combined_data_combo4['Tenor'] = combined_data_combo4['
           Maturity']

        # Extract start_date and maturity_date from all helpers
        start_dates = []
        maturity_dates = []

        for helper in synth_deposit_helper_ON_to_2M +
           fra_helpers_combo4 + focus_swap_helpers:
            start_dates.append(helper.earliestDate())
            maturity_dates.append(helper.maturityDate())

        combined_data_combo4['Start Date'] = start_dates
        combined_data_combo4['Maturity Date'] = maturity_dates

        # Reorder columns
        data_table_combo4 = combined_data_combo4[['Instrument', '
           Tenor', 'Start Date', 'Maturity Date', 'Bid', 'Ask', '
           Mid']]

        # Round values to three decimal places
        data_table_combo4 = data_table_combo4.round(3)

        # Display the DataFrame
        print("3M-Euribor Curve Instruments Summary (Combo 4):")
        display(data_table_combo4)


    #==================COMBINED COMPARISON PLOT OF ALL 4
       COMBINATIONS==============================

    # Combined plot

    dates = [spot_date + ql.Period(i, ql.Months) for i in range(0,
       5 * 12 + 1)]
    dates_dt = [_to_datetime(date) for date in dates]

    euribor3m_f_curve = [
        euribor3m_curve_with_synth_depo.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    euribor3m_f_curve_combo2 = [
        euribor3m_curve_combo2.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    euribor3m_f_curve_combo3 = [
        euribor3m_curve_combo3.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    euribor3m_f_curve_combo4 = [
        euribor3m_curve_combo4.forwardRate(
            d, euribor3m.maturityDate(d), ql.Actual360(), ql.Simple
        ).rate()
        for d in dates
    ]

    # Plot the combined comparison of all 4 combinations
    if graph_comparison:
        plt.figure(figsize=(width, height))
        euribor3m_f_curve_percent = [rate * 100 for rate in
           euribor3m_f_curve]
        euribor3m_f_curve_combo2_percent = [rate * 100 for rate in
           euribor3m_f_curve_combo2]
        euribor3m_f_curve_combo3_percent = [rate * 100 for rate in
           euribor3m_f_curve_combo3]
        euribor3m_f_curve_combo4_percent = [rate * 100 for rate in
           euribor3m_f_curve_combo4]

        plt.plot(dates_dt, euribor3m_f_curve_percent, label='Synth
           Deposits + 0X3M FRA + Futures + Swaps (Original set)',
           linewidth=1.5)
        plt.plot(dates_dt, euribor3m_f_curve_combo2_percent, label=
            'Synth Deposits + 0X3M FRA + Futures + More Focus on
            Swaps (Combo 2)', linewidth=1.5)
        plt.plot(dates_dt, euribor3m_f_curve_combo3_percent, label=
            'Synth Deposits + 0X3M FRA + FRAs + Swaps (Combo 3)',
            linewidth=1.5)
        plt.plot(dates_dt, euribor3m_f_curve_combo4_percent, label=
            'Synth Deposits + 0X3M FRA + FRAs + More Focus on Swaps
            (Combo 4)', linewidth=1.5)

        plt.gca().xaxis.set_major_locator(mdates.YearLocator(1))
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%
            Y'))
        plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda y,
             _: f'{y:.2f}'))

        plt.xlabel('Date', fontsize=12)
        plt.ylabel('Forward Rate (%)', fontsize=12)
        plt.title('Euribor 3M Forward Curves Comparison for
            Different Bootstrapping Instrument Sets', fontsize=14,
            pad=15)
        plt.legend(loc='upper right', fontsize=10, frameon=True,
            shadow=True, fancybox=True)
        plt.grid(True, which='major', linestyle='--', linewidth
            =0.5, alpha=0.7)
        plt.grid(True, which='minor', linestyle=':', linewidth=0.5,
             alpha=0.5)
        plt.tight_layout()
        plt.show()


    return euribor3m_f_curve, euribor3m_f_curve_combo2,
        euribor3m_f_curve_combo3, euribor3m_f_curve_combo4, dates
```

```python

euribor3m_f_curve, euribor3m_f_curve_combo2,
    euribor3m_f_curve_combo3, euribor3m_f_curve_combo4, dates =
    different_i_b_sets_curves_builder()
```

.12 Stability Assessment of Bootstrapping Sets

```python
num_simulations = 1000

# Funzione per generare variabili casuali con media zero e
    deviazione standard pari alla media dello spread
def generate_random_variable(spread, size):
    std_dev = spread.mean()
    return np.random.normal(0, std_dev, size)

# Funzione per applicare lo shock ai dati mantenendo il bid-ask
    spread positivo
def apply_shock(data, num_simulations):
    data['Spread'] = data['Ask'] - data['Bid']
    simulations = []

    for _ in range(num_simulations):
        random_variable = generate_random_variable(data['Spread'],
            len(data))
        bid_shocked = data['Bid'] * (1 + random_variable)
        ask_shocked = data['Ask'] * (1 + random_variable)

        # Assicurarsi che il bid-ask spread sia sempre positivo
        ask_shocked = np.where(bid_shocked >= ask_shocked,
            bid_shocked + abs(data['Spread']), ask_shocked)

        shocked_data = data.copy()
        shocked_data['Bid'] = bid_shocked
        shocked_data['Ask'] = ask_shocked
        shocked_data['Shocked_Spread'] = shocked_data['Ask'] -
            shocked_data['Bid']
        shocked_data['Mid'] = (shocked_data['Bid'] + shocked_data['
            Ask']) / 2

        simulations.append(shocked_data.copy())

    return simulations

# Applicazione degli shock e ricostruzione delle curve per ogni
    simulazione
s_ois_data_sim = apply_shock(ois_data, num_simulations)
s_deposit_data_sim = apply_shock(deposit_data, num_simulations)
s_fra_data3M_sim = apply_shock(fra_data3M, num_simulations)
s_forwards_ois_ecb_data_sim = apply_shock(forwards_ois_ecb_data,
    num_simulations)
s_swap_data3M_sim = apply_shock(swap_data3M, num_simulations)
s_futures_data3M_sim = apply_shock(futures_data3M, num_simulations)


DEBUG_SHIFTDATA = False
if DEBUG_SHIFTDATA:

    for i in range(num_simulations):
        print(s_ois_data_sim[i].head(10))

# Placeholder per curve
euribor3m_f_curve_sim = []
euribor3m_f_curve_combo2_sim = []
euribor3m_f_curve_combo3_sim = []
euribor3m_f_curve_combo4_sim = []
```

```

# Placeholder per tracking errors e metriche di errore
tracking_errors_original_set = []
tracking_errors_combo2 = []
tracking_errors_combo3 = []
tracking_errors_combo4 = []

max_variation_original_set = []
min_variation_original_set = []

max_variation_combo2 = []
min_variation_combo2 = []

max_variation_combo3 = []
min_variation_combo3 = []

max_variation_combo4 = []
min_variation_combo4 = []

for i in range(num_simulations):
    try:
        s_estr_curve, s_estr_helpers, s_zero_rates =
           estr_curve_builder(

            ois_data=s_ois_data_sim[i],
                deposit_data=s_deposit_data_sim[i],
                    forwards_ois_ecb_data=
                       s_forwards_ois_ecb_data_sim[i],
                        graph=False
        )

        s_euribor3m_curve_no_synth_depo,
           s_euribor3m_f_curve_no_synth_depo, s_fra_0x3_3m_helper,
           s_future_helpers, s_swap_helpers_dual, spot_date,
           euribor3m = euribor_curve_builder_no_synth_depo(

            estr_curve=s_estr_curve,
                fra_data3M=s_fra_data3M_sim[i],
                    futures_data3M=s_futures_data3M_sim[i],
                            swap_data3M=s_swap_data3M_sim[i],
                                graph=False
        )

        s_synth_deposit_helper_ON_to_2M = synthetic_deposit_builder
           (

            euribor3m_curve_no_synth_depo=
               s_euribor3m_curve_no_synth_depo,
                estr_curve=s_estr_curve,
                    print_alpha=False
        )

        s_euribor3m_curve_with_synth_depo, s_euribor3m_f_curve, _ =
            euribor_curve_builder_with_synth_depo(

            synth_deposit_helper_ON_to_2M=
               s_synth_deposit_helper_ON_to_2M,
                fra_0x3_3m_helper=s_fra_0x3_3m_helper,
                    future_helpers=s_future_helpers,
                        swap_helpers_dual=s_swap_helpers_dual,
                            graph=False
        )

        s_euribor3m_f_curve, s_euribor3m_f_curve_combo2,
           s_euribor3m_f_curve_combo3, s_euribor3m_f_curve_combo4,
           dates = different_i_b_sets_curves_builder(

            synth_deposit_helper_ON_to_2M=
               s_synth_deposit_helper_ON_to_2M,
                fra_0x3_3m_helper=s_fra_0x3_3m_helper,
                    swap_helpers_dual=s_swap_helpers_dual,
                        estr_curve=s_estr_curve,
                            euribor3m_curve_with_synth_depo=
                               s_euribor3m_curve_with_synth_depo,
                                futures_data3M=s_futures_data3M_sim
                                   [i],
                                    swap_data3M=s_swap_data3M_sim[i
                                       ],
                                        fra_data3M=s_fra_data3M_sim
                                           [i],
                                            graph=False,
                                                graph_comparison=
                                                   False,
        )

        euribor3m_f_curve_sim.append(s_euribor3m_f_curve)
        euribor3m_f_curve_combo2_sim.append(
           s_euribor3m_f_curve_combo2)
        euribor3m_f_curve_combo3_sim.append(
           s_euribor3m_f_curve_combo3)
        euribor3m_f_curve_combo4_sim.append(
           s_euribor3m_f_curve_combo4)

        #=========================Statistical Analysis
           ============================================

        # Calculate Differences Between Curves
        differences_original_set = np.array(euribor3m_f_curve) - np
           .array(s_euribor3m_f_curve)
```

```python
        differences_combo2 = np.array(euribor3m_f_curve_combo2) - np.array(s_euribor3m_f_curve_combo2)
        differences_combo3 = np.array(euribor3m_f_curve_combo3) - np.array(s_euribor3m_f_curve_combo3)
        differences_combo4 = np.array(euribor3m_f_curve_combo4) - np.array(s_euribor3m_f_curve_combo4)

        # Calculate Tracking Error and other metrics (in basis points) for the first 10 years
        tracking_error_original_set = np.sqrt(np.mean(differences_original_set**2)) * 10000
        tracking_error_combo2 = np.sqrt(np.mean(differences_combo2**2)) * 10000
        tracking_error_combo3 = np.sqrt(np.mean(differences_combo3**2)) * 10000
        tracking_error_combo4 = np.sqrt(np.mean(differences_combo4**2)) * 10000

        # Append tracking errors to the lists
        tracking_errors_original_set.append(tracking_error_original_set)
        tracking_errors_combo2.append(tracking_error_combo2)
        tracking_errors_combo3.append(tracking_error_combo3)
        tracking_errors_combo4.append(tracking_error_combo4)

        # Append max and min variations to the lists
        max_variation_original_set.append(np.max(differences_original_set) * 10000)
        min_variation_original_set.append(np.min(differences_original_set) * 10000)

        max_variation_combo2.append(np.max(differences_combo2) * 10000)
        min_variation_combo2.append(np.min(differences_combo2) * 10000)

        max_variation_combo3.append(np.max(differences_combo3) * 10000)
        min_variation_combo3.append(np.min(differences_combo3) * 10000)

        max_variation_combo4.append(np.max(differences_combo4) * 10000)
        min_variation_combo4.append(np.min(differences_combo4) * 10000)

    except RuntimeError as e:
        print(f"Simulation {i} failed due to convergence error: {e}")
        continue

# ---------------------------------------------------------------------------

# Plot the forward curves with the simulated curves
dates_dt = [_to_datetime(date) for date in dates]

fig, axs = plt.subplots(2, 2, figsize=(20, 12))

# Define color maps for each set of simulated curves
colors_simulated_original = plt.cm.summer(np.linspace(0, 1, len(euribor3m_f_curve_sim)))
colors_simulated_combo2 = plt.cm.cool(np.linspace(0, 1, len(euribor3m_f_curve_combo2_sim)))
```

```python
colors_simulated_combo3 = plt.cm.spring(np.linspace(0, 1, len(
    euribor3m_f_curve_combo3_sim)))
colors_simulated_combo4 = plt.cm.winter(np.linspace(0, 1, len(
    euribor3m_f_curve_combo4_sim)))


# First subplot: Original Set Curve with all simulated curves
for i, curve in enumerate(euribor3m_f_curve_sim):
     axs[0, 0].plot(dates_dt, [rate * 100 for rate in curve],
        linestyle='--', color=colors_simulated_original[i],
        linewidth=0.7, alpha=0.4)  # Plot simulated curves first
        with color
axs[0, 0].plot(dates_dt, [rate * 100 for rate in euribor3m_f_curve
    ], label='Real Original Set Curve', color='blue', linewidth=2)
    # Plot the true curve on top
axs[0, 0].xaxis.set_major_locator(mdates.YearLocator(1))
axs[0, 0].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axs[0, 0].yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
axs[0, 0].set_xlabel('Date', fontsize=12)
axs[0, 0].set_ylabel('Rate (%)', fontsize=12)
axs[0, 0].set_title(r'Real vs 1000 Simulated $\mathbf{Original\ Set
    }$ Curves', fontsize=14, pad=15)
axs[0, 0].legend(fontsize=10, frameon=True, shadow=True, fancybox=
    True)
axs[0, 0].grid(True, which='major', linestyle='--', linewidth=0.5,
    alpha=0.7)

# Second subplot: Combo 2 Curve with all simulated curves
for i, curve in enumerate(euribor3m_f_curve_combo2_sim):
     axs[0, 1].plot(dates_dt, [rate * 100 for rate in curve],
        linestyle='--', color=colors_simulated_combo2[i], linewidth
        =0.7, alpha=0.4)
```

```
axs[0, 1].plot(dates_dt, [rate * 100 for rate in
    euribor3m_f_curve_combo2], label='Real Combo 2 Curve', color='
    orange', linewidth=2)  # Plot the true curve on top
axs[0, 1].xaxis.set_major_locator(mdates.YearLocator(1))
axs[0, 1].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axs[0, 1].yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
axs[0, 1].set_xlabel('Date', fontsize=12)
axs[0, 1].set_ylabel('Rate (%)', fontsize=12)
axs[0, 1].set_title(r'Real vs 1000 Simulated $\mathbf{Combo\ 2}$
    Curves', fontsize=14, pad=15)
axs[0, 1].legend(fontsize=10, frameon=True, shadow=True, fancybox=
    True)
axs[0, 1].grid(True, which='major', linestyle='--', linewidth=0.5,
    alpha=0.7)

# Third subplot: Combo 3 Curve with all simulated curves
for i, curve in enumerate(euribor3m_f_curve_combo3_sim):
    axs[1, 0].plot(dates_dt, [rate * 100 for rate in curve],
        linestyle='--', color=colors_simulated_combo3[i], linewidth
        =0.7, alpha=0.4)
axs[1, 0].plot(dates_dt, [rate * 100 for rate in
    euribor3m_f_curve_combo3], label='Real Combo 3 Curve', color='
    green', linewidth=2)  # Plot the true curve on top
axs[1, 0].xaxis.set_major_locator(mdates.YearLocator(1))
axs[1, 0].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axs[1, 0].yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
axs[1, 0].set_xlabel('Date', fontsize=12)
axs[1, 0].set_ylabel('Rate (%)', fontsize=12)
axs[1, 0].set_title(r'Real vs 1000 Simulated $\mathbf{Combo\ 3}$
    Curves', fontsize=14, pad=15)
```

```python
axs[1, 0].legend(fontsize=10, frameon=True, shadow=True, fancybox=
    True)
axs[1, 0].grid(True, which='major', linestyle='--', linewidth=0.5,
    alpha=0.7)

# Fourth subplot: Combo 4 Curve with all simulated curves
for i, curve in enumerate(euribor3m_f_curve_combo4_sim):
    axs[1, 1].plot(dates_dt, [rate * 100 for rate in curve],
        linestyle='--', color=colors_simulated_combo4[i], linewidth
        =0.7, alpha=0.4)
axs[1, 1].plot(dates_dt, [rate * 100 for rate in
    euribor3m_f_curve_combo4], label='Real Combo 4 Curve', color='
    red', linewidth=2)  # Plot the true curve on top
axs[1, 1].xaxis.set_major_locator(mdates.YearLocator(1))
axs[1, 1].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
axs[1, 1].yaxis.set_major_formatter(FuncFormatter(lambda y, _: f'{y
    :.2f}'))
axs[1, 1].set_xlabel('Date', fontsize=12)
axs[1, 1].set_ylabel('Rate (%)', fontsize=12)
axs[1, 1].set_title(r'Real vs 1000 Simulated $\mathbf{Combo\ 4}$
    Curves', fontsize=14, pad=15)
axs[1, 1].legend(fontsize=10, frameon=True, shadow=True, fancybox=
    True)
axs[1, 1].grid(True, which='major', linestyle='--', linewidth=0.5,
    alpha=0.7)

plt.tight_layout()
plt.show()



#=========================Statistical Analysis
    ============================================
# Calculate the mean tracking error and the mean max/min variation
    for each set of curves
mean_tracking_error_original_set = np.mean(
    tracking_errors_original_set)
mean_max_variation_original_set = np.mean(
    max_variation_original_set)
mean_min_variation_original_set = np.mean(
    min_variation_original_set)

mean_tracking_error_combo2 = np.mean(tracking_errors_combo2)
mean_max_variation_combo2 = np.mean(max_variation_combo2)
mean_min_variation_combo2 = np.mean(min_variation_combo2)


mean_tracking_error_combo3 = np.mean(tracking_errors_combo3)
mean_max_variation_combo3 = np.mean(max_variation_combo3)
mean_min_variation_combo3 = np.mean(min_variation_combo3)

mean_tracking_error_combo4 = np.mean(tracking_errors_combo4)
mean_max_variation_combo4 = np.mean(max_variation_combo4)
mean_min_variation_combo4 = np.mean(min_variation_combo4)

# Create a list of tuples for easy sorting
results = [
    ("Original Set", mean_tracking_error_original_set,
        mean_max_variation_original_set,
        mean_min_variation_original_set,
        tracking_errors_original_set),
    ("Combo 2", mean_tracking_error_combo2,
        mean_max_variation_combo2, mean_min_variation_combo2,
        tracking_errors_combo2),
    ("Combo 3", mean_tracking_error_combo3,
        mean_max_variation_combo3, mean_min_variation_combo3,
        tracking_errors_combo3),
    ("Combo 4", mean_tracking_error_combo4,
        mean_max_variation_combo4, mean_min_variation_combo4,
        tracking_errors_combo4),
]

# Sort the list by the mean tracking error
results.sort(key=lambda x: x[1])

# Print the combined analysis
for name, mean_te, mean_max_var, mean_min_var, errors in results:
    # Perform one-sample t-test
    t_stat, p_value = ttest_1samp(errors, 0)
    significance = "statistically significant" if p_value < 0.05
        else "not statistically significant"

    # Print the combined analysis for each combo
    print(f"Analysis for {name}:")
    print(f"  Mean Tracking Error: {mean_te:.2f} bps")
    print(f"  Mean Max Variation: {mean_max_var:.2f} bps")
    print(f"  Mean Min Variation: {mean_min_var:.2f} bps")
    print(f"  t-statistic: {t_stat:.2f}")
    print(f"  p-value: {p_value:.4f}")
    print(f"  Result: The mean is {significance} (at 95% confidence
         level).")
    print("
        ------------------------------------------------------------------------
        ")
```

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutti coloro che, direttamente o indirettamente, hanno contribuito alla realizzazione di questo lavoro. In primo luogo, ringrazio il Professor Herzel per avermi dato l'opportunità di approfondire i temi trattati in questa tesi, per la sua fiducia e per l'infinita disponibilità dimostrata. Un sentito ringraziamento va al Professor Bianchetti, che mi ha gentilmente condiviso le slide del suo corso e mi ha messo in contatto con figure da cui ho tratto informazioni fondamentali per lo sviluppo della mia ricerca. Un grazie speciale a Mattia Mantovani, per avermi fornito l'ispirazione iniziale dalla quale è nata l'idea del progetto. Infine, un ringraziamento di cuore a tutti i miei amici e familiari, il cui supporto costante è stato fondamentale durante questi mesi.