KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

FACULTY OF ECONOMICS
AND BUSINESS

Multi-Curve Bootstrapping of Interest Rate Curves

Steff Helsen
m9710225

Thesis submitted to obtain the degree of

MASTER OF ACTUARIAL AND FINANCIAL ENGINEERING

Promotor: Prof. Dr. Wim Schoutens

Academic year: 2022-2023

Contents

Preface

This thesis reflects the work of nearly a year. The first half was spent researching and coding, the second half was spent writing, re-researching and re-coding. Personally I am rather pleased with the outcome, I hope you will like it just as much.

Even though I am the single author of this work, I would like to acknowledge some invaluable contributors without whom this would not have been possible.

First of all, I would like to thank my promoter prof. dr. Wim Schoutens. Wim, thank you for accompanying me on this journey and for your guidance.

I would also like to extend a big thank you to the people at Triodos. A lot of effort was required to set up this project but they went through with it anyway. I did this work within Group Modelling and Valuations (GMV), who provided me with the market data, but also a clear motivation: their interest in the outcome made this thesis more than an academic exercise.

Carsten Grevink, Deputy Head of GMV, was my day-to-day advisor for this thesis. We had a lot of discussions about the direction to take and he was always prepared to free up some time for me. He also reviewed drafts of this thesis. Carsten, hartelijk dank!

Last but not least I want to thank my life partner Jordane. She supported me morally for this thesis, and also for the entire project of my studies. She picked up my share of the household work and the full time care of our children Tania and Samuel when I needed to reserve my time for this work. She also reviewed drafts of this thesis and drew the logo of RateHike. Jordane, un petit mot pour exprimer un grand sentiment: merci!

Heverlee, May 14, 2023.

Chapter 1

Introduction

1.1 Motivation

The principles behind bootstrapping are well known, but with respect to the details the algorithm is mostly treated as a black box. The goal of this thesis is to open up the box and present bootstrapping in all its glory (without shying away from the ugly details). One way I will achieve this transparency is by building a software package for bootstrapping.

As with a lot of concepts in finance, the history of bootstrapping clearly shows an epoch *before* the 2008 financial crisis and one *after*. Before that event the IBOR rates were assumed to be risk-free. This means they all belonged to one curve (the risk-free curve) which could be used both for forecasting interest rates and for discounting payments. This is known as the single-curve framework. After the crisis it was understood the risk-free assumption does not apply to these curves and the single IBOR curve (for each currency) was split up in multiple curves (for each currency-tenor pair) and a separate discounting curve was used; this is called the multi-curve framework. Obviously, this has a big impact on the bootstrapping process; this shift was an important motivation for writing this thesis.

1.2 Bootstrapping

Bootstrapping exists as a concept in statistics, but this is unrelated to the concept *bootstrapping interest rate curves* in finance. Probably both names come from a reference to the saying "pulling yourself up by the bootstraps", but that is the only relationship. Bootstrapping interest rate curves is a process whereby an interest rate curve is constructed using observable market data in a specific way.

Interest rate curves are an important object within finance, mostly used in the domains of *fixed income*[1] and *ALM*. As a mathematical object, an interest rate curve is just a mapping of time to an interest rate $T \mapsto r(T)$. This does not say very much, since I have not specified anything about the interest rates in question.

[1] This domain treats interest bearing instruments, whereas the domain of *equity* treats dividend paying instruments.

The cleanest interest rates are zero coupon rates, linked to zero coupon bonds. A zero coupon bond is essentially a loan without intermediate interest payments, the investor pays a price P (today) and receives an amount C_1 back at some point in the future T_1. If I am to receive a cash flow C_2 at time T_1 and want to know its value today, by no-arbitrage the value has to be $C_2 \frac{P}{C_1}$. We have translated the information of the zero coupon bond into a *discount factor* $\mathrm{df}_1 = \frac{P}{C_1}$ to calculate the present value of a future cash flow on T_1. To translate this discount factor into an interest rate, one also needs to choose how the interest rate will be *compounded*. For *simple compounding*, the equation to solve reads

$$\frac{1}{1 + r(T_1)T_1} = \mathrm{df}_1,$$

and for *continuous compounding*

$$\mathrm{e}^{-r(T_1)T_1} = \mathrm{df}_1.$$

This is already an easy example of *bootstrapping*: we take a financial instrument (in this case the zero coupon bond) and from the market quote of this instrument today we deduce information about the interest rate curve. We can make choices (like the compounding; a lot more choices will be discussed in chapter 4) because we are choosing how to *encode* the information. For different choices the information does not change, just the way it is encoded. It is important to know the choices made, otherwise it is impossible to recover the information from the encoding.

At this point we have bootstrapped one instrument onto the curve and the curve contains information about one point in the future T_1. Suppose in the market there is a $T_1 \mathrm{x} T_2$ forward rate agreement quoted r_{FRA}. This means today we can lock in the interest rate r_{FRA} for a deposit starting on T_1 and maturing on T_2. If we would know the discount factors df_1 and df_2 for T_1 and T_2, we could calculate r_{FRA} as

$$r_{\mathrm{FRA}} = \frac{1}{T_2 - T_1}\left(\frac{\mathrm{df}_1}{\mathrm{df}_2} - 1\right).$$

We do know df_1 and we also know r_{FRA}, so we can solve this equation for df_2 and add a second point to the curve.

This shows the principle of bootstrapping: we take a set of instruments and their quotes on the market, and we consider the instruments in order of increasing maturity. From each instrument we extract information to add a point to the curve, using the intermediate curve built in the process. The points added in this way to the curve are called *pillars*. We also need to equip the curve with an *interpolation method*, so that we can find values between the pillars.

The example also gives some insight in the mechanics of bootstrapping: to determine the next pillar, we take a market quote, consider how we would determine this quote based on an existing curve (this is called *fixing* the quote), and then invert the process to determine the next point on the curve. This inversion can be done mathematically, but also numerically: take a guess for the next point on the curve, do the fixing, calculate the error with the market fixing, and take a new guess, reducing the error to an acceptable threshold. As will be explained, the numerical approach is more powerful than the mathematical approach.

In appendix A you find an example of bootstrapping that is a bit more elaborated than the example of this section.

1.3 Which curves?

One can consider a lot of interest rate curves, like government curves for the major economies, or corporate curves for companies with a certain credit quality. I will consider only curves for the major international indices.

1.3.1 IBOR curves

For each major currency there is a set of indices called LIBOR or EURIBOR[2]. The indices represent the rates at which banks of the highest credit quality can lend to and from one another for different *tenors* (maturities): 1 week, 1 month, 3 months, 6 months and 12 months. The rates are not determined by actual transactions, but on submissions of a selection of banks (the *panel*). The EURIBOR indices only consider EURO transactions, LIBOR indices are available for most major currencies: there is a GBP LIBOR, a USD LIBOR and a JPY LIBOR, also called TIBOR. There is even a EURO LIBOR. I will refer to all these indices (and the curves based on them) as IBOR[3].

Before the 2008 financial crisis, the IBOR rates were assumed to be risk-free. A single IBOR curve was constructed for each currency. The EURIBOR 3 month index was the 3 month point on *the* EURIBOR curve, and the EURIBOR 6 month index was the 6 month point on the same curve. This is called the *single-curve* framework. It is a logical step to go from the risk-free assumption to the use of one curve: if one takes out a 3 month deposit today and at the end of the period one rolls forward that investment for another 3 months, no-arbitrage dictates that the expected return of this strategy is equal to taking out a 6 month deposit today, if all the interest rates in question are risk-free.

The financial crisis showed that the risk-free assumption does not hold for the IBOR rates: there is credit risk even when lending to the IBOR panel banks, and liquidity concerns also need to be taken into account. The result is that today each IBOR index has its own curve. For example one considers instruments based on the EURIBOR 3 month rate to build the EUR3M curve, and instruments based on the EURIBOR 6 month rate to build the EUR6M curve: the indices represent *risky* rates, and the no-arbitrage argument no longer applies. This is called the *multi-curve* framework.

I have specified these curves in the multi-curve framework as being determined by the currency and the tenor of the underlying index, but it is also possible to think of them as relating to the payment frequency of the instruments that are based on them. A floating rate will be applied during a period equal to the tenor of the index, so the payment dates are separated by the tenor.

In recent times there have been some scandals related to the collusion of IBOR panel banks to manipulate the indices. This has led to reforms on how the indices are constructed. In fact, the LIBOR system is being discontinued (FCA, 2021). It is not clear yet how the market will evolve with respect to this change. The discontinuation of the EURIBOR

[2]LIBOR stands for the London InterBank Offered Rates; EURIBOR stands for EURo InterBank Offered Rates.

[3]Sometimes they are collectively called xIBOR, sometimes the term LIBOR is used for all curves and indices of this type.

indices is apparently not on the cards (EMMI, 2022b), although somewhat mysteriously a fallback rate has been introduced: the EFTERM[4] (EMMI, 2022a).

1.3.2 Overnight curves

Another type of interest rate curve exists for all major currencies: the overnight curve, based on the overnight rate.

The overnight rate is again an official index. The overnight rate for USD used to be the Federal Funds Rate, which was an unsecured rate. This has been changed to the SOFR, the Secured Overnight Financing Rate. For GBP the overnight index is the SONIA: the Sterling OverNight Index Average. For EURO we used to have the EONIA: EURO OverNight Index Average, but this has been replaced by the ESTR: the EURO Short-Term Rate.

There are forward looking market instruments based on these overnight rates (for example an Overnight Index Swap (OIS)), allowing an overnight curve to be bootstrapped. Actually, apart from the overnight deposit the OISs are the only instruments used to bootstrap the overnight curve (see table 4.1 on p27), so these curves are also called OIS curves.

Remark. In the single-curve framework the overnight rates were also considered to be part of the single curve for each currency, this all fits within the risk-free assumption. In the multi-curve framework you could also consider the OIS curve as just another IBOR curve (with as tenor 1 day). But as this curve is now considered to be the risk-free curve, it is treated separately.

1.3.3 Collateralized transactions

One of the major changes in financial markets since the financial crisis is that today most transactions are collateralized (see for example Grbac and Runggaldier, 2015). When two financial institutions enter into a transaction, they open a collateral account at the other institution. Each day, the value of the transaction (or netted value across transactions) is provided on the collateral account.

This construction can be set up bilaterally between two counterparties, but most of the time a *Central Clearing Platform* is used to provide these services.

This collateral account serves as a normal deposit account; the market convention is that this collateral account pays the overnight rate. As long as the transaction has not matured, or the counterparty has not defaulted, the money in the account is property of the counterparty. Once the transaction matures the money in the account is handed over to settle the transaction. Also if the counterparty defaults, legally the ownership of these funds transfers. Since the money in the account represents the value of the transaction, this means the transaction is now free of counterparty credit risk.

As such, the interest rate paid on these collateral accounts can be considered the risk-free rate. In case of default there might be legal fees that apply, and since the transaction

[4] EURO forward-looking ESTR-based term rate

unwinds immediately there might be portfolio balancing costs, but these are assumed negligible. So in the multi-curve framework, the IBOR curves are not assumed to be risk-free any more, this role has been taken over by the overnight curves.

An early article describing the shift from single-curve to multi-curve and the impact on the equations is Mercurio (2009). I did not follow this "old versus new"-approach, I just describe the multi-curve framework. I do include references to the single-curve framework where I think they add insight.

1.3.4 Curves considered

For setting up the bootstrapping I used 5 curves: for EURO the ESTR, EUR3M and EUR6M curves, and for GBP the SONIA and the GBP6M curves.

As mentioned, LIBOR is being discontinued and I do not know how the market will evolve. I took the opportunity to consider the GBP6M curve while it still existed as a nice addition to the test set, even if it is not useful for the future. This is also the reason for just considering 1 GBP LIBOR curve.

For the EURIBOR curves it would be possible to consider other tenors (like the EUR1M and the EUR12M curves), but the instruments using the EUR3M and EUR6M indices as underlying are the most liquid. It is common practice to use the EUR3M curve as a proxy for shorter tenors and the EUR6M curve for longer tenors.

1.4 Challenges

Bootstrapping does not get a huge amount of attention, in contrast to interest rate modelling. However, to learn something from the evolution of an interest rate curve, having an accurate idea of where it starts is also very important. Also, studying bootstrapping can give great insight in the mechanics of financial markets and financial instruments.

I think the problem is that *conceptually*, bootstrapping is not really a challenge, although in practice it is far from easy[5]. The difficulties in bootstrapping are getting all the details right with respect to date arithmetic. Also, the difference of missing a day, or counting the number of days slightly differently, might seem acceptable: when bootstrapping a curve without the necessary rigour, the error you make is of the order of 10^{-5}, which is not very big. If you are benchmarking your bootstrapping implementation and use inputs with 6 decimals, a perfect implementation will get you only to 10^{-7}.

Of course, when setting up a bootstrapping procedure, the only way to go is to try to get everything right, even if in a benchmarking exercise this is only two orders of magnitude better than a quick and dirty approach. Also, the limitation of the benchmarking precision to 10^{-7} is only caused by the use of 6 decimal digits for passing information around. Using more digits would lead to much better results.

[5]To cite my thesis advisor at Triodos: "Anyone who says bootstrapping is easy has never tried it."

1.5 Implementation

As part of the work for this thesis, I created a software package to perform the bootstrapping (Helsen, 2023). For this implementation I chose the name RateHike, and as slogan *Ramp up the quality of your interest rate curves.* The name is a gentle wink to Triodos, whose head office is located on the domain "De ReeHorst".

I consider the code to be an integral part of this thesis, but it is too long to include. I refer to the repository in the references. The implementation details are discussed in chapter 6.

1.6 Alternative approaches

This subsection discusses two alternatives for constructing a curve based on market information: *curve fitting* and *the linear algebra approach* (for the lack of a better term). I briefly mention them here for completeness, I have not investigated them.

Sometimes these alternatives are also referred to as bootstrapping (and sometimes bootstrapping is called curve fitting). I want to clearly distinguish between the approaches.

1.6.1 Curve fitting

In the curve fitting approach a number of mathematical base curves is selected, which have a *suitable* form for interest rate curves, and it is assumed that all interest rate curves are a linear combination of this set of base curves. Fitting a curve to market data then consists of finding the coefficients of the linear combination, which can be done by standard minimization techniques. This method was pioneered by Nelson and Siegel (1987) and extended by Svensson (1994).

Comparing this method to bootstrapping shows some differences: bootstrapping does not have any constraints with respect to finding the values on the pillars, but imposes a single interpolation function between pillars. Curve fitting imposes a restriction on the form of the entire curve, but as a combination of base functions this is more flexible than a single interpolation function. The downside is that the value of the interest rate curve on the pillars is less flexible.

This method is used by the ECB in determining its interest rate curve (Nymand-Andersen, 2018).

In Ametrano and Bianchetti (2013) and Ron (2000) it is claimed that this method is not precise enough for pricing purposes. Also in Nymand-Andersen (2018) it is made clear the goal of the exercise is not to retrieve accurate pricing.

For forecasting the evolution of the curve this approach can be very useful, especially when considering longer time horizons: the number of factors to predict is greatly reduced.

1.6.2 The linear algebra approach

As a theoretical exercise, one can write down all cash flows for a selection of instruments. If one arranges these cash flows in a matrix C with for each instrument a separate row, and for each cash flow date a separate column, one can write the equation

$$C \cdot D = P,$$

with D the vector of discount factors (one for each cash flow date) and P the vector of market prices of the instruments. The linear algebra approach is then to solve this system for the discount factors D as the cash flows C and the prices P are known.

This approach only applies to instruments with deterministic cash flows. Comparing this solving strategy to bootstrapping one also has to ask whether it is more robust to solve for each market instrument individually, or to solve for all of them combined. Another disadvantage of this method is that it does not use the interpolation algorithm when determining the curve. See section 4.2 where it is explained how this is a strong point for bootstrapping.

Remark. In the single-curve framework this method was used more often, because it was possible to work around the restriction of deterministic cash flows. When using the same curve to fix the rates and to discount the payments, both effects cancel out (see also the calculations in subsection 4.3.4). In the multi-curve framework such simplifications do not occur.

I feel the linear algebra approach is not a valid method of constructing a curve in the multi-curve framework.

1.7 Roadmap

In the next chapter I will review the literature.

Chapter 3 will introduce terminology, notations, details with respect to date arithmetic and the valuation theorems on which both single-curve bootstrapping and multi-curve bootstrapping are based. Chapter 4 will give all the details of how the bootstrapping algorithm works and all the choices that are available, except for interpolation, which is treated separately in chapter 5.

In Chapter 6 you find a description of the RateHike code: the choices made, the structure of the objects and a description of the functions. Finally, the results can be found in chapter 7. This chapter also contains a section highlighting the contributions of this thesis.

To build up towards the bootstrapping algorithm, the text uses multiple small building blocks. Figure 1.1 contains a dependency flowchart, showing the building blocks, where they are explained, and how they interrelate.

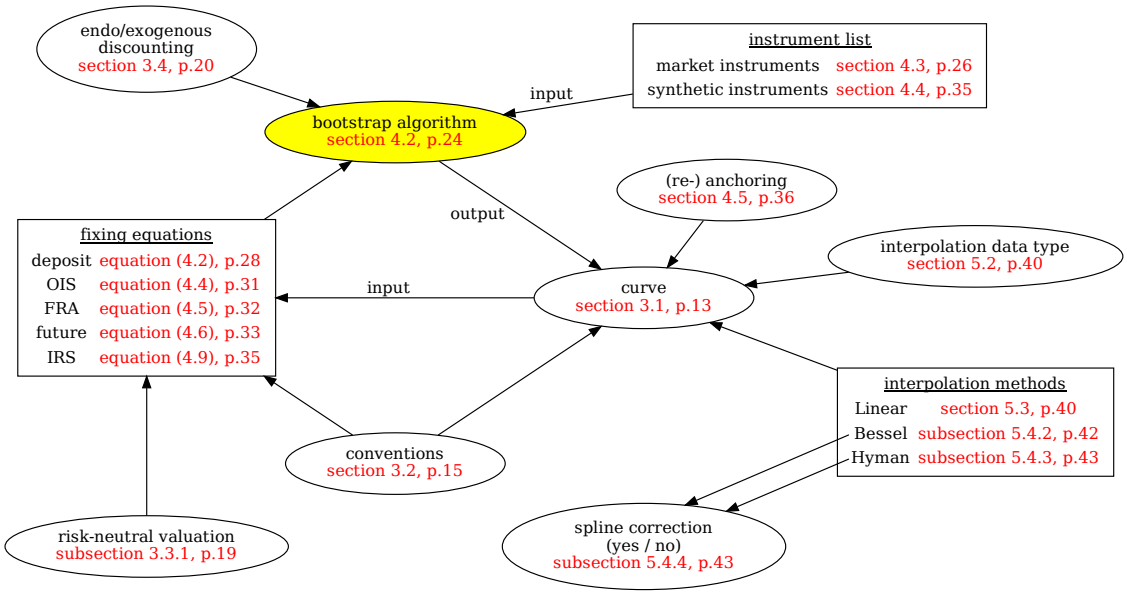

Figure 1.1: Dependency flowchart of the bootstrap algorithm and its building blocks as presented in this thesis.

Chapter 2

Literature review

Detailed literature about bootstrapping is rather rare. This chapter starts with some historical accounts of the shift from the single-curve to the multi-curve framework. The second section discusses the main references for this thesis. Then I briefly comment on some other related articles. I also found some master's theses whose subject is related, these are mentioned in a separate section. In the last section I discuss some open source implementations of bootstrapping algorithms.

In terms of handbooks, Hull (2012) seems to be the main reference with respect to bootstrapping. However, in that book only the principles are explained, so it is not a useful reference for this work.

2.1 From single-curve to multi-curve

From a bootstrapping perspective, the only reference I have found with some historical perspective is Ametrano and Bianchetti (2013), which is also our main reference (see the next section).

Broadening the scope to articles about interest rate modelling, a nice account of the shift from the single-curve to the multi-curve framework can be found in the introduction of two articles: Martin (2018) and Henrard (2010). Actually, Henrard had foreseen the problem: Henrard (2007) was published 1 month before the financial crisis broke out.

A more in-depth treatment of the evolution from the single-curve to the multi-curve framework can be found in Grbac and Runggaldier (2015, Chapter 1).

2.2 Main references

My main source for the bootstrapping theory and the practical details is Ametrano and Bianchetti (2013). This is actually an updated version of Ametrano and Bianchetti (2009); this earlier version was written before the financial crisis and only discusses the single-curve framework.

Ametrano and Bianchetti (2013) contains a lot of details on bootstrapping. I am very

grateful to have had that article to start my thesis from. I see my thesis as a continuation of that article. What my thesis adds is mainly to fill the gap between the theory as presented in that article and the implementation of these ideas. I was also not entirely convinced of the mathematical foundations as presented in that article, I feel my presentation of this in sections 3.3 and 4.3 is more to the point.

One point which Ametrano and Bianchetti (2013) only briefly mention is the interpolation. At the outset of the work I had also thought choosing a good interpolation scheme would not be too hard; I had listed it as a secondary goal to investigate the impact of different interpolation schemes thoroughly. In section 4.2 you can read why the interpolation scheme is an integral part of a bootstrapping algorithm and chapter 5 has all the details on interpolation. While developing RateHike I quickly understood the importance of this element so it got upgraded to be part of the main goal.

With respect to interpolations, more literature can be found. Most articles propose a new interpolation scheme. The goal of this thesis was to implement a robust and standard bootstrapping algorithm, not to investigate all possible ways of doing this, so most of these articles I did not use. I did manage to sneak in a suggestion of my own, see subsection 5.4.4.

Hagan and West (2006) starts by giving a thorough review of standard interpolation schemes in the context of interest rate curves together with implementation details, so this became my main reference with respect to the interpolation. It also proposes its own special interpolation scheme, but I did not include this in the RateHike implementation. I explain why in section 5.5. The authors also published a follow-up article (Hagan & West, 2008), but apart from a nice summarizing table the later article does not add anything over the earlier article.

With respect to market conventions, the main references are the publications of the International Swaps and Derivatives Association (ISDA). However, I did not have access to them. Instead I used OpenGamma (2013).

2.3 Other articles

An often cited article with respect to bootstrapping is Ron (2000). It pre-dates the multi-curve framework and does not contain details about conventions, so to me this is mostly of historical importance.

Most other references are written by modellers, so (again) convention details are mostly left out. Examples of this are Mercurio (2010) and Fujii et al. (2009) (and their modelling follow-up Fujii et al. (2011)). I used those articles mostly as a cross-reference for the theory in Ametrano and Bianchetti (2013).

With respect to interpolation I want to mention one other reference: Adams (2001). In this article the author values *smoothness* over all other properties of yield curve interpolation schemes, and postulates that *global interpolation schemes* are the best to achieve such smoothness. This is quite in contrast with Hagan and West (2006), who value *locality* more. As explained in chapter 5 I agree with Hagan and West.

2.4 Master's theses

I found 3 master's theses treating a related subject that deserve a mention.

Spoor (2013) has a similar goal to my thesis. He gives a lot of theoretical details (closely following Fujii et al. (2009, 2011)) but does not give any implementation details; he refers to OpenGamma (see next section) without being precise on how it was used. In contrast to all the theoretical detail, the worked examples are overly simplistic.

In Iebesh (2020) the focus is on interpolation. The author closely follows Hagan and West (2006, 2008), and reaches the same conclusions. To me this work did not add any value to the two mentioned articles.

Finally, El Menouni (2015) is more of a theoretical nature and does not discuss bootstrapping as such. A stochastic calculus framework for pricing derivatives in a multi-curve framework is developed. There is a lot of overlap with bootstrapping so I did feel it deserved a mention.

2.5 Implementations

A lot of commercial products have a bootstrapping method. However, without access to the code or the documentation they were of no help for this work. I found two open source implementations, which I could of course use.

Quantlib (The QuantLib contributors, 2022, v1.27.1) is a free/open-source library for quantitative finance. The authors of our main reference Ametrano and Bianchetti (2013) are also contributors to QuantLib, one is even a founder. However, the bootstrapping code in the QuantLib library does not seem to be related to the equations in the article. QuantLib is a vast library, offering a lot more functionality; basically it aims to cover most (or even all) calculations in the field of Quantitative Finance. As such, it is hard to understand the bootstrapping procedure by just looking at the code. Together with incomplete documentation, this makes the bootstrapping in QuantLib lack transparency (from my point of view). I did use QuantLib as a benchmarking tool in the early development stages of RateHike.

Remark. Just to make it absolutely clear: I do not claim RateHike is in any way better than QuantLib, except in clarity. This is achieved by being singly focussed on bootstrapping, as opposed to the vast scope of functionality QuantLib aims to achieve.

In the thesis of Spoor (2013) I encountered the software OpenGamma Strata (OpenGamma, 2022). The only bootstrapping implementation I could find in the code used the numerical linear algebra approach (see subsection 1.6.2), so I did not use this software.

Chapter 3

Notation, terminology, definitions, theorems

In the domain of fixed income a lot of confusing terminology exists. Most concepts and objects have more than one name, and some names are used for more than one object or concept. In the first section I lay out how the terminology is used in this thesis, at the same time fixing the notation.

The second section lists and explains all conventions that will be used and how they apply, together with some examples. In the third section valuation theory is discussed. The fourth section discusses the shift from the single-curve framework to the multi-curve framework. The chapter ends with a specialist section on using the IBOR curve for discounting.

3.1 Terminology and notations

Mathematically an interest rate curve is a mapping $T \mapsto r(T)$, assigning an interest rate to future times, using the convention that *now* is represented by $T = 0$. Typically an interest rate curve is represented by a number of *pillars*, which is a finite set $\{(T_i, r(T_i))\}_{i=1}^n$, and an *interpolation* method, which allows to find $r(T)$ for $T \in [T_1, T_n] \setminus \{T_i\}_i$. Sometimes also one or two *extrapolation* algorithms are given for $T < T_1$ and $T > T_n$. I will add a pillar $(0, 0)$ at the start of the curves and only consider the value of the curve for the closed interval $[T_0, T_n] = [0, T_n]$.

In this thesis I will only consider *zero coupon curves*. Sometimes I will call them zero curves, sometimes just interest rate curves. They will be denoted by $\mathcal{C}$, adding a subscript when a differentiation between multiple curves is needed, like $\mathcal{C}_{\text{EUR6M}}$ or $\mathcal{C}_{\text{SONIA}}$. An interest rate curve can be looked at from different ways: zero rates, discount factors, forward rates, ... So a distinction will be made between the curve as an object and the quantities:

Definition 3.1.1 (quantities of an interest rate curve $\mathcal{C}$).

zero coupon rate

$$\mathcal{C}^z(T),$$

discount factor

$$\mathcal{C}^d(T) = \mathrm{e}^{-\mathcal{C}^z(T)T},$$

instantaneous forward rate

$$\mathcal{C}^f(T) = -\frac{d \ln \mathcal{C}^d(T)}{dT},$$

FRA rate

$$\mathcal{C}^F(T_1, T_2) = \frac{1}{T_2 - T_1}\left(\frac{\mathcal{C}^d(T_1)}{\mathcal{C}^d(T_2)} - 1\right).$$

Remark. One needs to be careful with respect to conventions; in this definition there are implicit assumptions about both the day count conventions and the compounding conventions. In the next section conventions will be treated extensively. In this case the assumptions are that the curve is indexed by the year fraction, that the chosen day count convention is additive and that the curve uses continuous compounding.

This presentation of a curve is non-standard; typically interest rate curves get two time parameters: $\mathcal{C}(t, T)$, indicating that the yearly interest rate applicable at time t for a cash flow at time T is $\mathcal{C}^z(t, T)$. This allows to discuss the dynamics of an interest rate curve evolving through time. In this thesis only interest rate curves at time $t = 0$ are considered, so to ease the notation the first parameter will be dropped: $\mathcal{C}(T)$.

Remark. When considering FRA rates $\mathcal{C}^F(T_1, T_2)$ I do use two time parameters: the start date and the end date, but this still represents *today's* value of the FRA rate. Typically these rates are indexed by 3 time parameters: $\mathcal{C}^F(t, T_1, T_2)$ to indicate the time t value of this rate, but again: here only $t = 0$ is treated.

A time T represents a *year fraction* and is measured using a *day count convention*. Sometimes we need to measure the year fractions of consecutive periods $S_i \to S_{i+1}$ from a sequence of payment dates $S_1, \ldots, S_n$. Those are called *accrual factors* and will be denoted τ_i.

The curves and indices considered will be indicated fully as the EURIBOR (3/6) month (rate/curve), the GBP 6 month (rate/curve) and the (EURO/GBP) overnight (rate/curve), but also in shorthand as EUR3M, EUR6M, GBP6M, ESTR and SONIA. They will also be referenced in general by the terms IBOR and OIS. These references can both indicate the underlying index, or the curve. If the context is not 100% clear this will be made explicit.

For bootstrapping in the multi-curve framework one can distinguish the *discounting curve* and the *forecasting curve*. The latter is also called *forwarding curve*. When using a separate discounting curve for bootstrapping, this is an *exogenous* discounting curve, when using the same curve for discounting and forecasting this is called an *endogenous* discounting curve.

As part of this framework, the OIS curve is used for discounting when bootstrapping each

IBOR curve. This fact is sometimes stressed by the term *dual-curve bootstrapping*.

Modellers sometimes use the term *stochastic basis* to refer to the multi-curve framework. The *basis* refers to the OIS-IBOR spread. In terms of bootstrapping this term is slightly inappropriate since a deterministic curve is built.

3.2 Conventions

Most of the information in this section can be classified as general knowledge. I did use OpenGamma (2013) to confirm the information, but some of it was not present, other things were out of date. The application of the conventions as described in this section did allow me to benchmark the bootstrapping process with respect to Bloomberg.

When calculating an interest amount, you need to know the interest rate, and the period to calculate the interest on. This period needs to be expressed as a (fractional) number of years. This gets messy quickly: first you need to know the exact dates and then you need to know how to convert these dates to year fractions. I will not treat this subject exhaustively, I will only define the conventions that are used in the calculations for the selection of curves considered in this thesis.

Exact dates will always be presented in *yyyy-mm-dd* format, sometimes prepended by the weekday if relevant. When the year is not important, dates will be denoted *mm-dd*.

3.2.1 Spot lag

There exist forward starting contracts, but also spot starting contracts. They are designed to start immediately, but a *spot lag* applies.

spot lag
: the delay between the signing of a contract and the date it enters into force. For forward starting contracts the delay between the fixing of the rate and the start date.

For the EURO instruments used the spot lag is always 2 business days, for the GBP instruments it is 0 days. For forward starting contracts the mechanics are somewhat complicated. Figure 3.1 illustrates this. Suppose *today*, Wednesday 2022-11-30, one enters into an FRA 3x9 (being a Forward Rate Agreement that starts in three months and matures in nine months) fixed at r_{FRA}. One first calculates the spot date Friday 2022-12-02 and then moves the spot date forward (three months in this case) to the start date Thursday 2023-03-02. Next one counts backwards the spot lag to find the *fixing date* Tuesday 2023-02-28. The object of the FRA contract is then the exchange of r_{FRA} against the EUR6M rate determined on the fixing date, for the period from the start date to the maturity date and for a predetermined notional amount. This EUR6M rate fixed on the fixing date represents a 6 month deposit between banks on the EURIBOR panel, which would also start on the start date and run to the maturity date.

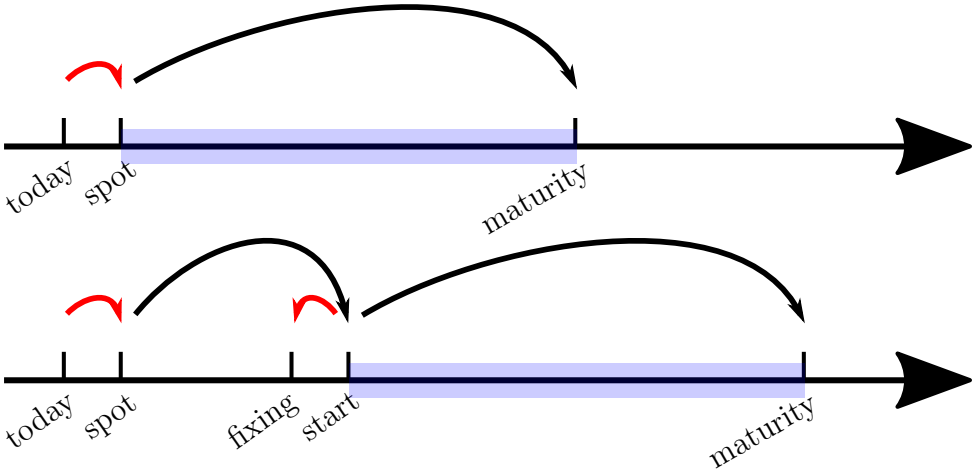

Figure 3.1: Timelines for a spot starting (*top*) and a forward starting instrument (*bottom*). The spot lag is indicated by the short jumps in red. The interest period is highlighted in blue.

3.2.2 Payment dates

For instruments with a regular payment schedule, it is necessary to calculate the payment dates. For quarterly payments, if a payment is on 06-15, this means the next payment date will be on 09-15. There exist two special conventions with respect to payment dates:

end-of-month convention
: payments dates fall on the last day of a month.

third-Wednesday convention
: payment dates fall on the third Wednesday of a month.

Only one instrument type uses the third-Wednesday convention: futures. All other instruments apply the end-of-month convention, but only if the spot date is the last business day of the month. If this is not the case, no special payment dates convention applies.

An example of how to apply the end-of-month convention: if for a quarterly paying instrument one payment date is 04-30, the next payment date will be 07-31.

3.2.3 Rolling conventions

A requirement for a payment date is that it is a business day. If a theoretical payment date falls in a weekend or on a holiday, you need to apply the rolling conventions.

Following
: move the payment date to the next business day.

Modified Following
: move the payment date to the next business day, except if this would be in the next month; in that case move it to the previous business day.

Modified Following is often shortened to *ModFollowing*. Instruments with a maturity less than a month get rolling convention Following, those with a maturity of one month or more get Modified Following.

3.2.4 Holidays

In order to know the holidays, a calendar needs to be agreed on.

TARGET2
: holiday calendar of the Trans-European Automated Real-time Gross settlement Express Transfer system. Holidays are defined as 01-01, Good Friday, Easter Monday, 05-01, 12-25 and 12-26. (European Central Bank, 2000)

UK
: holiday calendar of the UK financial system. All holidays as TARGET2, except 05-01, but if a holiday falls in a weekend it is moved to the next weekday. Additional holidays on the first Monday of May (early spring bank holiday), the last Monday of May (spring bank holiday) and the last Monday of August (summer bank holiday). Also some extra days for royal jubilees. (Pyper, 2015)

The TARGET2 calendar applies to the EURO instruments, the UK calendar applies to the GBP instruments.

3.2.5 Day count conventions

To assign a *year fraction* to a period between two calendar dates, day count conventions are used.

ACT/365
: the number of days between start date and end date, divided by 365.

ACT/360
: the number of days between start date and end date, divided by 360.

30E/360
: $$\big(\text{year(enddate) - year(startdate)}\big)+ \frac{1}{12}\big(\text{month(enddate) - month(startdate)}\big)+ \frac{1}{360}\big(\text{day(enddate) - day(startdate)}\big),$$
where a correction is applied to the day of month: for the 31st, 30 is used.

30E/360 is the "European version" of the 30/360 day count convention, also called Eurobond basis. Other versions exist, with different corrections applied.

So the year fraction for the period 2022-11-30 – 2023-05-31 is $\frac{182}{365} = 0.49863$ using ACT/365, it is $\frac{182}{360} = 0.50556$ using ACT/360, and using 30E/360 we find $1+\frac{-6}{12}+\frac{0}{360} = 0.5$.

GBP instruments use ACT/365 as a day count convention. Most EURO instruments use ACT/360; an exception is the interest rate swap (IRS) which will be treated in subsection 4.3.7.

3.2.6 Compounding

An interest rate r applicable until a date S with year fraction T can be *compounded* in different ways.

simple compounding
an investment of 1 grows to $1 + rT$ at S.

periodic compounding
an investment of 1 grows to $(1 + \frac{r}{m})^{mT}$ at S, where m is the number of annual compounding periods, with $m = 1$, 2, 3, 4, 6 or 12.

continuous compounding
an investment of 1 grows to e^{rT} at S.

For example for a half year investment ($T = 0.5$) at an interest rate of 6%, the investment pays back 1.03 with simple compounding, 1.03038 with monthly compounding ($m = 12$) and 1.03046 with continuous compounding.

When looking over 1 period not bigger than a year, periodic compounding is identical to simple compounding: consider an annual interest rate of 4% over a period of 3 months ($T = 0.25$). With simple compounding we get $1 + rT = 1.01$ at maturity. With periodic compounding we set m to 4 and we get $(1 + \frac{r}{m})^{mT} = 1.01$. This is the same because the period needs to be set such that $mT = 1$, which means that the exponent vanishes and that $rT = \frac{r}{m}$.

Since for all instruments considered each interest payment is limited to 1 period which is not bigger than a year, we can consider all instruments to use simple compounding. Assigning periodic compounding to some of them might be more pleasing from a theoretical viewpoint, but there really is no difference.

3.3 Instrument valuation

In this section I discuss the theory behind instrument valuation, this is necessary to derive the fixing equations in the next chapter. This is a very rich theory; I will not develop it in this thesis, I will just present the relevant results, and only in the simplified setting needed in what follows.

Valuation theory is a part of stochastic calculus; this has been worked out in a number of handbooks. A complete treatment from the ground up can be found in Bingham and Kiesel (2010). An application of the theory to interest rate modelling can be found in Brigo and Mercurio (2006).

3.3.1 Risk-neutral valuation

The theory will be formulated for instruments with just one cash flow to keep the notation as simple as possible. The generalization to instruments with multiple cash flows consists of adding indices and summation signs. The general theorem calculates a price for every $t \in [0, T]$; here I present the special case $t = 0$ as a corollary. For the general theorem, see Brigo and Mercurio (2006, Proposition 2.1.2 p26).

Corollary 3.3.1 (to risk-neutral valuation). *Under the right conditions, the price of a financial instrument paying a cash flow N at time T is equal to*

$$\mathbb{E}\left[\frac{N}{B(T)}\right],$$

where $B(T)$ is the time T value of a bank account starting at $B(0) = 1$ in the currency of the instrument and accruing interest continuously at the risk-free rate $r(T)$, and $\mathbb{E}$ is the expectation under the measure associated with this bank account.

$\mathbb{E}$ is called the *risk-neutral measure*, also lending its name to the theorem.

Remark. In Brigo and Mercurio (2006) this theorem is proven for a general bank account, and later specified to be the risk-free bank account in equation (2.22) on p.39.

One clearly recognizes that the price is a discounted cash flow. But this equation is hard to apply, since the risk-free interest rate $r(T)$ is stochastic, so $B(T)$ is a stochastic discount factor in the equation.

Using *the change of numéraire technique*[1], this can be simplified. The following corollary is based on Brigo and Mercurio, 2006, equation (2.20) on p.38.

Corollary 3.3.2 (to risk-neutral valuation). *Under the right conditions, the price of a financial instrument paying a cash flow N at time T is equal to*

$$P(0, T)\,\mathbb{E}^T[N],$$

with $P(0, T)$ today's value of a zero-coupon bond paying 1 in the currency of the instrument at time T and $\mathbb{E}^T$ the measure associated with this zero-coupon bond.

The measure $\mathbb{E}^T$ is called the *forward measure*. An interesting fact is $P(0, T) = \mathbb{E}[1/B(T)]$, so we have moved from a stochastic discount factor to a deterministic one by taking expectations. Note that this implies the deterministic discount factor still depends on the risk-free rate.

The price to pay for this "trick" is that another pricing measure is used, but if this causes problems later on in a calculation, the change of numéraire technique should be applied again at that point.

One might be wondering what the *right conditions* might be and why this equation does not always hold. In practice, you can always just apply this equation. But to prove it

[1]The change of numéraire technique (also called *the change of measure technique*) is an element of the theory of stochastic calculus. I will not explain this technique, the name allows to use it intuitively without a full theoretical understanding.

mathematically, one needs to make assumptions on the market: that it is arbitrage free (no free lunch) and that it is complete. The details are beyond the scope of this text.

A final link that needs to be made is to go from corollary 3.3.2 to a zero coupon curve. Using information from the market, we can get information on the expectation $\mathbb{E}^T[N]$ of the cash flows for the selected instruments that are linked to the forecasting curve $\mathcal{C}$. The fixing equations that will be developed in the next chapter then serve as the constraints that are imposed on the discount factors of this curve; solving for these discount factors gives us the information to build the curve.

3.3.2 Pricing under collateral

In Ametrano and Bianchetti (2013) the bootstrapping theory is based on a new valuation theory (called *pricing under collateral*). Essentially this theory is the same as the theory of risk-neutral valuation: investments are hedged, and the gains and losses of the hedging accrue interests. For risk-neutral valuation this interest rate is assumed to be the risk-free rate, for pricing under collateral this is assumed to be the collateral rate.

As stated before, the rate paid in the market on the collateral account is the overnight rate. So it is clear this is the rate used in the pricing under collateral theory. But it is not necessary to use this new theory: as the overnight rate is the rate paid on collateralized transactions, one can very well assume this rate is risk-free and apply risk neutral valuation.

Luckily Proposition 3.3 on p.12 of Ametrano and Bianchetti (2013) is identical to corollary 3.3.2 where the collateral rate replaces the risk-free rate. From a theoretical perspective it might be more appealing to develop a different theory for the new framework, but I think the new framework can still be built on the old theory.

3.4 From single-curve to multi-curve

The transition from the single- to the multi-curve framework is best illustrated by considering *FRA rates*. FRAs will be discussed in detail in subsection 4.3.5, here I will only introduce the elements needed to discuss the transition.

An FRA is a Forward Rate Agreement. It is a contract that locks in an interest rate for a future period by specifying that the contract rate will be exchanged against the market deposit rate for the period considered. So when the start of the period arrives, you can take out a deposit at the market rate, and your portfolio of this deposit together with the FRA nets out to be a deposit at the FRA contract rate.

FRAs are constructed so that they have an initial price of 0: the contract rate is fixed to achieve exactly that. Let r_{FRA} be the (deterministic) contract rate for an FRA from T_1 to T_2, and let r^*_{depo} be the future (and hence stochastic) deposit rate for the same period. At T_2 we know the realization r_{depo} of the stochastic rate r^*_{depo} and there is an exchange of interest payments: Nr_{FRA} versus Nr_{depo}, where N is the notional amount of the contract.

Remark. This is not true for market FRA contracts, but the treatment here uses *textbook*

FRA rates[2]. When discussing the FRA instrument in detail in subsection 4.3.5 this difference will be made clear.

Using corollary 3.3.2 one can write the value of the contract as

$$N\omega P(0,T_2)\tau\,\mathbb{E}^{T_2}[r_{\text{FRA}} - r^*_{\text{depo}}] = N\omega P(0,T_2)\tau\,(r_{\text{FRA}} - \mathbb{E}^{T_2}[r^*_{\text{depo}}]),$$

where τ is the year fraction between T_1 and T_2 using the day count convention of the instrument and $\omega = \pm 1$ is a factor to indicate the side of the contract under consideration.

So to achieve the initial price of zero we need to set

$$r_{\text{FRA}} = \mathbb{E}^{T_2}[r^*_{\text{depo}}]. \tag{3.1}$$

Remember $\mathbb{E}^{T_2}$ is an expectation under the measure associated with the risk-free zero-coupon bond paying 1 at T_2.

In the single-curve framework, r^*_{depo} is a value derived from the same interest rate curve as the one specifying how to take expectations, and one can work this out to be

$$\mathbb{E}^{T_2}[r^*_{\text{depo}}] = \frac{1}{\tau}\left(\frac{\mathcal{C}^d(T_1)}{\mathcal{C}^d(T_2)} - 1\right), \tag{3.2}$$

see Brigo and Mercurio (2006, Proposition 2.5.1 p38). This is exactly the FRA rate from definition 3.1.1, so the definition is consistent with the property just found. Bootstrapping FRA rates in the single-curve framework consists of combining equations (3.1) and (3.2), plugging in the market quote r_{FRA} and the discount factor $\mathcal{C}^d(T_1)$ (from an intermediate version of the curve) and solving for $\mathcal{C}^d(T_2)$.

In the multi-curve framework, it is not possible to work out the expectation in equation (3.1). The way to work around this is to use equation (3.2) as a *definition* of the forecasting curve $\mathcal{C}$. This is actually very natural: as the FRA rates are expectations under a measure linked to the discounting rate, defining the forecasting curve in function of these expectations is the logical next step. In Ametrano and Bianchetti (2013) this idea is not made clear and it is hidden in the mathematical derivations. Mercurio (2010) has a clear explanation of this shift.

So we will use equations (3.1) and (3.2) both for bootstrapping in the single- and in the multi-curve framework. The measure $\mathbb{E}^{T_2}$ used to take the expectation is always connected to the discounting curve: the *forward measure*. So in the multi-curve framework, $\mathcal{C}$ represents the forecasting curve and relation (3.2) *defines* the curve whereas in the single-curve framework, $\mathcal{C}$ represents the single curve and relation (3.2) is a property.

Remark. In this section I considered an FRA that exchanges a single fixed payment against a single floating payment. Other instruments, like an interest rate swap (IRS) exchange multiple fixed payments against multiple floating payments. The same framework can be used in that case, the interest rate for each floating payment is r^*_{depo} for the deposit over the referenced period.

Remark. Along with the shift from a single-curve to a multi-curve framework, there has also been a shift from forward rates to FRA rates. In definition 3.1.1 I called $\mathcal{C}^F$ "the FRA

[2]Also called *prototypical* or *standard* FRA rates.

rate", but historically this used to be the *forward rate*. By combining equations (3.1) and (3.2) we can conclude that in the single-curve framework the FRA rate and the forward rate are equal. In this sense, the FRA rate is the natural generalization of the forward rate to the multi-curve framework.

I will rephrase the very important nuance introduced in this section one more time: for both the single-curve framework and the multi-curve framework corollary 3.3.2 can be used; the only difference that remains is how the curves are used. In the single-curve framework, one assumes there exists only one IBOR curve for each currency and one assumes this curve is risk-free. So when pricing cash flows based on an IBOR index, one uses this curve both for forecasting and for discounting. In the multi-curve framework, one assumes the OIS curve is risk-free and uses this curve for discounting. A different IBOR curve is constructed for each currency-tenor combination; these curves are used for forecasting.

3.5 IBOR discounting in the multi-curve framework

Before the financial crisis, IBOR curves were assumed to be risk-free and cash flows were discounted using them. Does the multi-curve framework imply IBOR curves should only be used for forecasting, and not for discounting? The answer is *yes and no*.

In Hull and White (2013) it is argued that to value a derivatives portfolios, one should use the OIS curve to determine the *no-default value* of this portfolio, after which the same curve should be used to determine the value adjustments for a possible default event.

But this article only considers derivatives portfolios, so we are in the *trading book* of financial institutions. Traditionally (and for regulatory reporting) valuations in the *banking book* are done differently. Reading between the lines of the aforementioned article, one could conclude that when valuing a *banking book* portfolio, if the contractual cash flows are discounted with a zero coupon curve that represents the riskiness of the portfolio, one ends up with the present value of the portfolio already corrected for the risk.

This is by no means a claim of the authors, and already quite far from the core of this thesis. So perhaps I should restate the answer in this case to *maybe*.

In practice IBOR curves are used for this purpose. But this implies some other questions, specifically with respect to *anchoring*, see section 4.5.

Chapter 4

Bootstrapping

This chapter will explain the details of the bootstrapping algorithm and the choices that need to be made. I will also reference the options available in RateHike.

The first section discusses the choices one needs to make for the curve. The second section gives the details of the bootstrap algorithm. Section 3 gives the information on instrument level: fixing equations and conventions. At the end there are two specialist sections on synthetic instruments and anchoring.

4.1 Properties of a zero coupon curve

The bootstrap algorithm will extract market information from a set of instruments and encode that information in the zero coupon curve. It needs to take into account the conventions used by the instruments in order to correctly interpret the market quotes. Without this knowledge a quote is just a number; the algorithm needs to know the market conventions to transform this number into information. The algorithm also needs to know how to encode this information: it will again output a set of numbers, which together with a set of conventions represent the information. The conventions attached to the output (the zero coupon curve) are largely up to the user. Here I list these choices.

compounding
: A natural choice is *continuous compounding*: this makes the interpretation of zero coupon rates quite easy and also allows for the simplest mathematical manipulations. This is one of the few points where RateHike does not allow a flexible choice to the user.

day count convention
: In Ametrano and Bianchetti (2013) it is stated that day count convention used for an interest rate curve is increasing and additive. I chose ACT/365 for all the curves (but RateHike allows the user to make a different choice).

instrument set
: I did not investigate this as part of this thesis; I just adopted the standard Bloomberg inputs, see subsection 4.3.1. Again, RateHike allows other choices.

synthetic instruments
This is discussed in section 4.4.

interpolation scheme
This is a very important property that can be broken up in the interpolation data type and the interpolation method. Both will be treated in chapter 5.

anchoring
This amounts to choosing the date for which $t = 0$: *today's date* or *spot date*. For GBP curves, with a spot lag of 0 days, these two dates are the same so there is only one logical choice, for EURO curves the spot lag is 2 business days and the question becomes more complicated. this is treated in section 4.5.

endogenous or exogenous discounting
The multi-curve framework consists of bootstrapping the OIS curve *endogenously* and to use this OIS curve as an *exogenous* discounting curve for the IBOR curve bootstrapping, so these were the choices made for this thesis. Once more, RateHike does not impose this on its users.

Apart from these *functional* choices, some technical choices also need to be made: what is the convergence threshold and the maximum number of function evaluations when adding a new point to the curve, what is the convergence threshold and the maximum number of iterations for the multiple sweeps necessary for bootstrapping a curve with cubic spline interpolation? Do we store the curve as discount factors or as zero rates, and if we use zero rates do we use numbers or percentages? Do we index the pillars by the dates, or by the year fractions? These choices will be documented in chapter 6.

I would like to stress that all these properties of a curve need to be chosen before constructing it, these properties will be used in the bootstrapping procedure and will impact the result. See for example appendix A, where a worked example shows how the choice of interpolation scheme results in different zero coupon rates.

Inversely, when using a curve to discount cash flows or to forecast interest rates, it is important to use it exactly the way it was constructed. Using non-transparent software or data providers to obtain a bootstrapped curve might make that impossible.

4.2 The algorithm

Bootstrapping is an inverse problem. The direct problem is to determine the quote for a financial instrument using an interest rate curve. The theory and equations for this direct problem are given in section 4.3. Bootstrapping turns this around: it takes in a set of instruments and their market quotes to construct a curve. It consists of three levels: the outer level loops multiple times over the entire input set until the output has converged, the middle level loops over each instrument in the input set and the inner level adds a point on the curve based on one instrument. Here the algorithm will be explained bottom-up, starting with the inner level. The algorithm is presented in pseudo-code as algorithm 1.

input: set of instruments **output:** interest rate curve	
Initialize a 1-point curve: $(t = 0,\ \mathrm{df} = 1)$ **While** not yet converged **and** not yet reached max iterations:	*outer*
For each instrument in input (in order of increasing maturity):	*middle*
Guess a discount factor for the maturity of the instrument Update the curve with that discount factor Update the interpolation coefficients Calculate the fixing of the instrument on this updated curve Calculate the error of this fixing wrt the market quote Make a better guess until this error is "close enough" to 0	*inner*

Algorithm 1: Bootstrapping algorithm. The outer, middle and inner levels are indicated by the indentation and in the margin.

To start the algorithm, the curve is initialized with only one point, representing a discount factor of 1 on the spot date.

In the inner level, the algorithm updates one pillar on the curve, either adding it or recalculating it. The algorithm uses the intermediate version of the curve as constructed up to that point in time, together with the market quote of the instrument under consideration. It will update the curve on the pillar representing the maturity date of the instrument. It does so by taking an initial guess for that value, determining the quote for the instrument based on the updated curve (using the equations that will be developed in the next section), and optimizing the guess until this fixing matches the market quote. An extra step that needs to be taken after updating a pillar on the curve is updating the interpolation coefficients, as the interpolation can be used in calculating the fixing of this instrument, as well as all instruments to follow.

The middle level of the algorithm loops one time through the set of instruments in order of increasing maturity. For each instrument, it will (re)calculate the point on the curve corresponding to the maturity date by applying the inner level.

Finally, the outer level of the algorithm applies the middle level multiple times. It stops when a maximum number of iterations is reached, or when the updates of the discount factors of the curve are smaller than a predetermined threshold. The outer level is only relevant for curves equipped with a spline interpolation. For linear interpolation, every new iteration in the outer level would calculate the same values. RateHike deals with this by setting the maximum number of outer level iterations to 1 when linear interpolation is chosen.

The goal of the inner level is to invert the fixing equations of the input instruments. It does so by numeric optimization. For the simplest instruments, it would also be possible to invert the fixing equations mathematically, which would theoretically be faster and more accurate. In practice the numerical optimization is not slower and just as accurate. Rather than having a different treatment for different instruments, the same numerical optimization is applied across the board.

This way of bootstrapping is also a natural method of dealing with missing information.

If you are determining the pillar for T_i, you can use all information already encoded in the curve up to that point in time. Moreover, the numerical solving strategy assigns a guessed value to the pillar T_i, after which all information for each $T \in [0, T_i]$ is available. If you look through the sets of instruments and the fixing equations of the next section, you will easily find examples of instruments that need to be added without all intermediate information being available, look for example to the IRSs with maturities of 15 years and more.

4.3 Market instruments

This section describes the market instruments used as an input to the bootstrapping: the selection of instruments, the conventions that apply and the fixing equations. It builds on the previous chapter. The derivation of the fixing equations is done by first principles: the requirements for the instrument interest rates are written down and converted into fixing equations using the theory of the previous chapter.

This approach was also taken in Ametrano and Bianchetti (2013), but as stated in subsection 3.3.2 there a new theory *pricing under collateral* is used as a foundation, here *risk neutral valuation* is used.

4.3.1 Instrument selection

The list of instruments used to bootstrap a curve can be debated. Typically you want to choose *liquid* instruments that are good in representing the curve. I did not investigate this subject, I just used the instruments chosen by Bloomberg for my examples. Table 4.1 lists the selection of instruments for each curve. Between parenthesis you find the maturity of the instrument for all instruments, and the start of the instrument for futures and FRAs.

The future strip for the EUR3M curve contains one future for each of the first 6 months and then 6 more quarterly futures. These quarterly futures are indicated in *italics* in table 4.1 and start in March, June, September and December, so the specific start and end will change depending on the current month.

For the EUR6M curve the Bloomberg input set changed during 2022: at the start of the year it contained FRAs 7x13, 8x14, 10x16 and 11x17 as well; these instruments were not selected any more at the end of the year. This is related to the market liquidity of these instruments.

4.3.2 Conventions

How to apply the different conventions across instruments was already discussed in section 3.2. Here I give a small recap. The instrument specific exceptions will be discussed later in this section.

spot lag
EURO instruments: 2 business days,
GBP instruments: 0 days.

ESTR	EUR3M	EUR6M	SONIA	GBP6M
depo (1 DY)	depo (3 MO)	depo (6 MO)	depo (1 DY)	depo (6 MO)
OIS (1 WK)	future (1x4)	FRA (1x7)	OIS (1 WK)	IRS (1 YR)
OIS (2 WK)	future (2x5)	FRA (2x8)	OIS (2 WK)	IRS (18 MO)
OIS (1 MO)	future (3x6)	FRA (3x9)	OIS (1 MO)	IRS (2 YR)
OIS (2 MO)	future (4x7)	FRA (4x10)	OIS (2 MO)	IRS (3 YR)
OIS (3 MO)	future (5x8)	FRA (5x11)	OIS (3 MO)	IRS (4 YR)
OIS (4 MO)	future (6x9)	FRA (6x12)	OIS (4 MO)	IRS (5 YR)
OIS (5 MO)	*future (8x11)*	FRA (9x15)	OIS (5 MO)	IRS (6 YR)
OIS (6 MO)	*future (11x14)*	FRA (12x18)	OIS (6 MO)	IRS (7 YR)
OIS (7 MO)	*future (14x17)*	IRS (2 YR)	OIS (7 MO)	IRS (8 YR)
OIS (8 MO)	*future (17x20)*	IRS (3 YR)	OIS (8 MO)	IRS (9 YR)
OIS (9 MO)	*future (20x23)*	IRS (4 YR)	OIS (9 MO)	IRS (10 YR)
OIS (10 MO)	*future (23x26)*	IRS (5 YR)	OIS (10 MO)	IRS (12 YR)
OIS (11 MO)	IRS (3 YR)	IRS (6 YR)	OIS (11 MO)	IRS (15 YR)
OIS (1 YR)	IRS (4 YR)	IRS (7 YR)	OIS (1 YR)	IRS (20 YR)
OIS (18 MO)	IRS (5 YR)	IRS (8 YR)	OIS (18 MO)	IRS (25 YR)
OIS (2 YR)	IRS (6 YR)	IRS (9 YR)	OIS (2 YR)	IRS (30 YR)
OIS (3 YR)	IRS (7 YR)	IRS (10 YR)	OIS (3 YR)	IRS (40 YR)
OIS (4 YR)	IRS (8 YR)	IRS (11 YR)	OIS (4 YR)	IRS (50 YR)
OIS (5 YR)	IRS (9 YR)	IRS (12 YR)	OIS (5 YR)	
OIS (6 YR)	IRS (10 YR)	IRS (15 YR)	OIS (6 YR)	
OIS (7 YR)	IRS (11 YR)	IRS (20 YR)	OIS (7 YR)	
OIS (8 YR)	IRS (12 YR)	IRS (25 YR)	OIS (8 YR)	
OIS (9 YR)	IRS (15 YR)	IRS (30 YR)	OIS (9 YR)	
OIS (10 YR)	IRS (20 YR)	IRS (40 YR)	OIS (10 YR)	
OIS (11 YR)	IRS (25 YR)	IRS (50 YR)	OIS (12 YR)	
OIS (12 YR)	IRS (30 YR)		OIS (15 YR)	
OIS (15 YR)	IRS (40 YR)		OIS (20 YR)	
OIS (20 YR)	IRS (50 YR)		OIS (25 YR)	
OIS (25 YR)			OIS (30 YR)	
OIS (30 YR)			OIS (40 YR)	
OIS (40 YR)			OIS (50 YR)	
OIS (50 YR)				

Table 4.1: Bootstrapping instruments per curve. Between brackets the maturity for deposits, OIS and IRS, and "start month x maturity month" for FRA and futures. See the text for remarks with respect to the EUR3M and EUR6M instrument selection.

payment dates (exception: futures)
if the spot date is the last business date: end-of-month
else: no payment date convention.

rolling convention
if the maturity is less than 1 month: Following,
else: Modified Following.

holiday calendar
EURO instruments: TARGET2 calendar,
GBP instruments: UK calendar.

day count convention (exception: EURO IRSs)
GBP instruments: ACT/365,
EURO instruments: ACT/360.

compounding
All payments can be considered as simple compounding.

4.3.3 Deposits

A deposit (in full: certificate of deposit) is a financial instrument for which the depositor pays a notional amount N at spot date and receives her money back at the maturity date, together with an interest amount. The maturity of the deposit will be one business day for a deposit paying the overnight rate, and 3 months or 6 months for deposits linked to an IBOR index (other tenors exist, but are not considered here).

A deposit paying the EUR6M rate quoted at r_{depo} on Wednesday 2022-11-30 starts on Friday 2022-12-02 and runs to Friday 2023-06-02. The interest amount is $Nr_{\text{depo}}\frac{182}{360}$. (Note I used the conventions of the previous subsection: a spot lag of 2 business days and a day count convention of ACT/360; no rolling was necessary.)

Deposit rates are already zero coupon rates in a sense, so *fixing* is a fancy term here. The deposit pays as interest the underlying index, so fixing the deposit rate consists of reading of the value of the index from the curve. The zero coupon curve does not necessarily use the same conventions, so this value still needs to be transformed; this conversion is done implicitly by the choice of presenting all fixing equations in terms of the discount factors. An example is worked out in Appendix A, where this conversion is shown explicitly.

Let $T_{\mathcal{C}}$ and T_{depo} be the year fractions between the spot and maturity dates as calculated by the day count convention of respectively the curve and the instrument. Note that an investment of 1 grows to $1 + r_{\text{depo}}T_{\text{depo}}$ at the maturity date, so this represents a discount factor

$$\mathcal{C}^d(T_{\mathcal{C}}) = \frac{1}{1 + r_{\text{depo}}T_{\text{depo}}}. \tag{4.1}$$

Fixing r_{depo} on a curve $\mathcal{C}$ is done by inverting equation (4.1):

$$r_{\text{depo}} = \frac{1}{T_{\text{depo}}}\left(\frac{1}{\mathcal{C}^d(T_{\mathcal{C}})} - 1\right). \tag{4.2}$$

Remark. Note the use of the different day count conventions. To extract information from the curve one needs to use the day count convention of the curve. To convert a yearly interest rate to a periodical payment, one needs to apply the conventions of the instrument in question. This remark holds for all instruments.

4.3.4 Overnight index swaps

An Overnight Index Swap (OIS) exchanges a fixed interest rate r_{OIS}, paid over a certain period (e.g. 1 month), to the overnight interest rate accumulated over the same period. r_{OIS} is fixed in such a way that the initial value of the transaction is 0. These are already quite complex mechanics, but the equations for the fixing are still relatively simple.

Remark. The floating payment is defined as the compounded overnight rate over a certain period, but this rate is varying; this is not to be confused with the periodic compounding convention which applies to fixed interest rates.

If the maturity of an OIS is less than a year, there is only 1 payment date: the maturity date. For maturities bigger than a year, the payments are yearly. If the maturity is not an exact multiple of 1 year, the broken period is at the start.

For example a EURO OIS quoted r_{OIS} on Wednesday 2022-11-30 with a maturity of 18 months runs from Friday 2022-12-02 to Monday 2024-06-03 and the fixed leg pays $Nr_{\text{OIS}}\frac{182}{360}$ on Friday 2023-06-02 and $Nr_{\text{OIS}}\frac{367}{360}$ on Monday 2024-06-03, with N the notional amount. The floating leg pays on the same dates, using the same year fractions, but the amounts are the compounded overnight interest rates over the payment period.

To derive the OIS fixing equation the discount factors for all payment dates *and* all business days are needed, so let S_0 be the spot date of the instrument, let S_i^{pay}, $i = 1, \ldots, n$ be all the payment dates and let S_j^{bus}, $j = 1, \ldots, m$ be all the business days in the life of the contract. Let $T_{i,\mathcal{C}}^{\text{pay}}$ and $T_{j,\mathcal{C}}^{\text{bus}}$ be the year fractions between S_0 and S_i^{pay} respectively S_j^{bus}, using the day count conventions of the curve, with the convention $T_0^{\text{pay}} = T_0^{\text{bus}} = 0$. Let $\tau_{i,\text{OIS}}^{\text{pay}}$ be the day count between S_{i-1}^{pay} and S_i^{pay} using the day count convention of the instrument and $\tau_{j,\text{OIS}}^{\text{bus}}$ similar for the business days. Let $\mathbb{E}^{T_j^{\text{bus}}}[r_j^*]$ be the expectation of the overnight rate applicable at S_j^{bus}. Note that the last day of both series is the maturity date of the instrument, so $S_m^{\text{bus}} = S_n^{\text{pay}}$ and $T_{m,\mathcal{C}}^{\text{bus}} = T_{n,\mathcal{C}}^{\text{pay}}$, which will be used to simplify the equations.

To fix the transaction at a value of 0 it is necessary that today's value of the payments of the fixed and floating legs are equal. To figure out the value of the floating leg, let us start with an OIS that has only 1 period. The floating payment at the end of the period represents the compounded interest on a notional of N:

$$N \prod_{j=1}^{m} (1 + r_j \, \tau_{j,\text{OIS}}^{\text{bus}}) - N,$$

with r_j the realizations of the r_j^*.

Today's value of the second term is $-N\mathcal{C}^d(T_{m,\mathcal{C}}^{\text{bus}})$. To determine today's value of the first term, we use that the value on S_m^{bus} is $N \prod_{j=1}^{m}(1 + r_j \, \tau_{j,\text{OIS}}^{\text{bus}})$. the expectation of the value

on S^{bus}_{m-1} is

$$N\prod_{j=1}^{m-1}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})\cdot(1+\mathbb{E}^{T^{\text{bus}}_{m,\mathcal{C}}}[r^*_m]\tau^{\text{bus}}_{m,\text{OIS}})=N\prod_{j=1}^{m-1}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})\frac{\mathcal{C}^d(T^{\text{bus}}_{m-1,\mathcal{C}})}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})},$$

where equation (3.2) was used. If we move back one more business day, the expected value on S^{bus}_{m-2} is

$$\begin{aligned}
&N\prod_{j=1}^{m-2}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})\frac{\mathcal{C}^d(T^{\text{bus}}_{m-1,\mathcal{C}})}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}(1+\mathbb{E}^{T^{\text{bus}}_{m-1,\mathcal{C}}}[r^*_{m-1}]\tau^{\text{bus}}_{m-1,\text{OIS}})\\
&\qquad=N\prod_{j=1}^{m-2}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})\frac{\mathcal{C}^d(T^{\text{bus}}_{m-1,\mathcal{C}})}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}\frac{\mathcal{C}^d(T^{\text{bus}}_{m-2,\mathcal{C}})}{\mathcal{C}^d(T^{\text{bus}}_{m-1,\mathcal{C}})}\\
&\qquad=N\prod_{j=1}^{m-2}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})\frac{\mathcal{C}^d(T^{\text{bus}}_{m-2,\mathcal{C}})}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}.
\end{aligned}$$

If we continue to do this, the expectation on the start date for the final value is

$$N\frac{\mathcal{C}^d(T^{\text{bus}}_0)}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}=N\frac{1}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}.$$

Putting things together, the present value of the final payment of the floating leg is

$$\begin{aligned}
\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})\,\mathbb{E}\big[N\prod_{j=1}^{m}(1+\,r_j\,\tau^{\text{bus}}_{j,\text{OIS}})-N\big]&=\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})\cdot(N\frac{1}{\mathcal{C}^d(T^{\text{bus}}_{m,\mathcal{C}})}-N)\\
&=N\big(1-\mathcal{C}^d(T^{\text{pay}}_{1,\mathcal{C}})\big),
\end{aligned}$$

where we used $T^{\text{bus}}_{m,\mathcal{C}}=T^{\text{pay}}_{n,\mathcal{C}}$ and $n=1$. Actually, a bit more work is needed to make the derivation above rigorous. In order to apply the expectation on each factor individually it is necessary to *condition* the expectations and use the *power law*. This only impacts the mathematical formulas, the logic and the result stay the same.

If we repeat the same analysis for the second period of a multi-period OIS, we find the expected present value of the floating payment at the end of the second period is

$$N\big(\mathcal{C}^d(T^{\text{pay}}_{1,\mathcal{C}})-\mathcal{C}^d(T^{\text{pay}}_{2,\mathcal{C}})\big).$$

Using the cancellation when this is added to the result for the first period, we can conclude that the general expression for the expected present value of the floating leg payments is

$$N\big(1-\mathcal{C}^d(T^{\text{pay}}_{n,\mathcal{C}})\big), \tag{4.3}$$

independent of the number of the number of periods.

To achieve an initial value of 0 for the OIS, the result in (4.3) needs to be equal to the expected present value of the fixed leg:

$$\sum_{i=1}^{n}\mathcal{C}^d(T^{\text{pay}}_{i,\mathcal{C}})\,N\,r_{\text{OIS}}\,\tau^{\text{pay}}_{i,\text{OIS}}=N\big(1-\mathcal{C}^d(T^{\text{pay}}_{n})\big).$$

Rearranging the terms, the fixing equation for an OIS becomes

$$r_{\text{OIS}} = \frac{1 - \mathcal{C}^d(T^{\text{pay}}_{n,\mathcal{C}})}{\sum_{i=1}^{n} \mathcal{C}^d(T^{\text{pay}}_{i,\mathcal{C}})\, \tau^{\text{pay}}_{i,\text{OIS}}}. \tag{4.4}$$

In the course of the analysis we found a nice property for the value of the floating leg. The value of this leg can be summarized by the difference of the discounting factors on the start date (which for $t = 0$ is equal to 1) and on the maturity date. This is called the *telescoping property*.

Notice this simplification occurred for the compounding of interest rates within one payment period, and also for the addition of payments across consecutive periods. The reason behind this is that every payment is *at par*: the interest rate paid and the discount factor applied are read of from the same curve and both effects cancel out.

This simplification was at the heart of a lot of calculations in the single-curve framework. The fact that this relationship breaks when considering different forecasting and discounting curves is an important element in why a multi-curve framework needed to be developed.

4.3.5 Forward rate agreements

An x x y forward rate agreement (FRA) can be considered to be a forward starting deposit; it starts x months from now and matures y months from now. At the start date, the quoted FRA rate will be exchanged against the value of the underlying index at the fixing date, which is a spot lag before the start date (see figure 3.1). The tenor of the underlying index is always equal to $y - x$ months.

If an FRA would really be a forward deposit, the interest amount would be paid on the maturity date, but actually the interest amount is paid on the starting date. The discount factor from the maturity date to the start date is contractually agreed to be linked to the underlying index at the start date. Another difference is that for a deposit the depositor pays the notional amount at the start date, and receives the notional together with the interest on the maturity date, for an FRA the notional amounts are not exchanged.

A EURO 3x9 FRA quoted at r_{FRA} on Wednesday 2022-11-30 starts on Thursday 2023-03-02 and matures on Monday 2023-09-04. Let r^*_{depo} represent the EUR6M deposit rate on Tuesday 2023-02-28. The cash flow generated by the instrument on Thursday 2023-03-02 is

$$N\omega \frac{(r^*_{\text{depo}} - r_{\text{FRA}})\frac{186}{360}}{1 + r^*_{\text{depo}}\frac{186}{360}}$$

where N is the notional amount of the contract and $\omega = \pm 1$ is a factor indicating the position taken in the contract.

Remark. Note the fixing of the EUR6M rate is done on start date minus spot lag. However, if on Tuesday 2023-02-28 you would enter into a EUR6M deposit, the spot lag would apply again and the spot date of the deposit would be Thursday 2023-03-02, perfectly aligned with the start date of the FRA.

Remark. Theoretically, the cash flow to be exchanged will be known on the fixing date, and is due on the start date. However, if the contract is collateralized, the settlement will occur on a daily basis, taking into account the market fluctuations, until at fixing date the value does not change any more. Legally, the money in the collateral account will remain the property of the debtor until the start date.

If you compare this cash flow equation with the one that was used in section 3.4, you will notice a difference because the cash flow falls on the start date and contractually is discounted from the maturity date with the underlying IBOR index. This is the difference between the *textbook* FRA as was treated in section 3.4, and the *market* FRA that is being treated here.

This also means we cannot use *definition* (3.2) directly to translate an observed FRA rate to discount factors: we need to apply corollary 3.3.2. Using this corollary to determine the fixing of the FRA results in a *convexity adjustment* to be applied because the discounting curve and the forecasting curve will both evolve between today and the payment date (and they are not perfectly correlated). This convexity adjustment can be calculated (modelled, actually) based on the volatilities and the correlation of the point on the two curves representing the start date.

I do not include an equation for the convexity adjustment, one can either present a theoretical construction as a definition, or a model-dependent quantity where all components need a precise definition for the equation to make sense. For an example of both flavours of these equations, see Ametrano and Bianchetti (2013, Appendix C, equations (124) and (130)).

In Ametrano and Bianchetti (2013) this convexity adjustment is assumed to be negligible, based on Mercurio (2010). I will take the same approach, for mostly practical reasons: introducing volatility information in the bootstrapping process would add a lot of complexity for a very small impact. I do want to stress that the conclusion of Mercurio (2010) is that *typically* this convexity adjustment is negligible, but not always.

The nice thing is that if we assume the convexity adjustment to be 0, the valuation of the market FRA coincides with the valuation for the textbook FRA and we can apply equations (3.1) and (3.2) again to use the quoted FRA rate to bootstrap the next pillar.

Let $T_{s,\mathcal{C}}$ and $T_{m,\mathcal{C}}$ be the year fractions from the spot date to the start date and the maturity date respectively in the day count convention of the forecasting curve, and let τ_{FRA} be the year fraction from the start date to the maturity date in the day count convention of the instrument, then

$$r_{\text{FRA}} = \frac{1}{\tau_{\text{FRA}}}\left(\frac{\mathcal{C}^d(T_{s,\mathcal{C}})}{\mathcal{C}^d(T_{m,\mathcal{C}})} - 1\right). \tag{4.5}$$

4.3.6 Futures

Futures are special instruments. They use Third-Wednesday as a payment date convention. A futures contract is quoted on the market as $100(1 - r_{\text{fut}})$, with r_{fut} the implied futures rate. At inception, its value is 0. At the start date, in theory there is an exchange of $N(1 - r^*_{\text{depo}})$ against $N(1 - r_{\text{fut}})$, where N is the *notional amount* and r^*_{depo} is the value of the underlying index on the fixing date. Of course both payments are netted and just

the difference is paid by the party making a loss. Moreover, these contracts are daily margined, so by the payment date the settlement has already occurred.

These mechanics are contractually arranged. One notices immediately that no day counts are being applied, so the implied futures rate is not a yearly interest rate. Because of the difference in the mechanics of the payout calculation, the risk profile of a futures contract is different from an FRA. This means hedging will also be different, and this implies a convexity adjustment is needed when translating an implied futures rate to an FRA rate. In fact, this convexity adjustment does not need to be calculated; one can download the convexity adjusted futures rates from Bloomberg, and probably the same holds for other data providers. The mathematical equations behind the adjustments are given in Appendix B.

After the translation from a price to a rate and the application of the convexity adjustment, the fixing equation is just the same as equation (4.5) for the FRA. This formulation of the fixing equation is also based on the fact that the translated and adjusted rate is what the bootstrapping procedure gets as an input; this is the fixing it needs to reproduce.

$$\tilde{r}_{\text{fut}} = \frac{1}{\tau_{\text{fut}}}\left(\frac{\mathcal{C}^d(T_{s,\mathcal{C}})}{\mathcal{C}^d(T_{m,\mathcal{C}})} - 1\right), \tag{4.6}$$

where $\tilde{r}_{\text{fut}}$ is the translated and adjusted futures rate, τ_{fut} is the year fraction from the start date to the maturity date using the instrument day count convention, and $T_{s,\mathcal{C}}$ and $T_{m,\mathcal{C}}$ are the year fractions from the spot date to respectively the start date and the maturity date, using the day count convention of the curve.

4.3.7 Interest rate swaps

An Interest Rate Swap (IRS, also just *swap*) is an instrument that exchanges a periodic fixed rate payment against a periodic floating rate payment, where the floating rate payment is linked to an IBOR index[1]. The payment schedules of the fixed and floating *legs* can be different and the day count conventions of the two legs might also differ.

A GBP IRS uses ACT/365 as day count convention for both legs. For a EURO IRS the fixed leg uses 30E/360 and the floating leg uses ACT/360. However, in the fixing equations the year fractions of the floating leg drop out of the calculations and only the day count convention of the fixed leg is used (as is shown at the end of this subsection). So in RateHike the fixed leg day count convention is assigned to the instrument as a whole, as this is the only day count convention used in the calculations.

The floating leg pays a frequency based on the tenor of the index it is linked to; e.g. the floating leg of an IRS on the EUR6M index pays every 6 months. For a GBP IRS, the fixed leg pays the same period as the floating leg, for a EURO IRS the fixed leg pays a yearly coupon, irrespective of the underlying index.

[1]Here only vanilla fixed-floating swaps are considered, although different kind of swaps exist: basis swaps (exchanging one index to another) and cross-currency swaps (exchanging cash flows in different currencies) are two examples.

leg	payment date	fixing date	accrual factor
fixed	Mon 2023-12-04		362/360
(30E/360)	Mon 2024-12-02		358/360
	Tue 2025-12-02		360/360
float	Fri 2023-06-02	Wed 2023-05-31	182/360
(ACT/360)	Mon 2023-12-04	Thu 2023-11-30	185/360
	Mon 2024-06-03	Thu 2024-05-30	182/360
	Mon 2024-12-02	Thu 2024-11-28	182/360
	Mon 2025-06-02	Thu 2025-05-29	182/360
	Tue 2025-12-02	Fri 2025-11-28	183/360

Table 4.2: Example of the dates involved in the mechanics of an IRS.

Take for example a 3 year IRS on the EUR6M index quoted r_{IRS} on Wednesday 2022-11-30. The spot date is Friday 2022-12-02. Table 4.2 contains the payment dates for both legs, and for the floating leg also the fixing dates. It also contains the *accrual factors*: the length of the period for which the interest rate applies. The interest rates paid are r_{IRS} for the fixed leg and the *previous* fixing for the floating leg: the interest rate fixed at S_i (minus the spot lag) applies to the period up to S_{i+1}, so the interest paid at S_{i+1} is the interest rate fixed at S_i (minus the spot lag) multiplied by the accrual factor for the period from S_i to S_{i+1}.

To formulate the fixing equation for an IRS corollary 3.3.2 will be used with a forecasting curve $\mathcal{C}_{\text{for}}$ and a discounting curve $\mathcal{C}_{\text{dis}}$. For the fixing equation, both curves are known, when you use this in the bootstrapping context, $\mathcal{C}_{\text{dis}}$ is known and $\mathcal{C}_{\text{for}}$ is being constructed.

Let S_i^{fix}, $i = 1 \dots n$ be the payment dates of the fixed leg, let S_j^{float}, $j = 1 \dots m$ be the payment dates of the floating leg, and let $S_0^{\text{fix}} = S_0^{\text{float}}$ be the spot date. Let $T_{i,\text{dis}}^{\text{fix}}$ be the year fractions from the spot date to S_i^{fix} using the day count conventions of the discounting curve. Let $\tau_{i,\text{IRS-fix}}$ be the accrual factors for the fixed payment periods, calculated using the day count conventions of the fixed leg of the instrument. Let $T_{j,\text{for}}^{\text{float}}$ and $T_{j,\text{dis}}^{\text{float}}$ be the year fraction from the spot date to S_j^{float}, using the day count conventions of respectively the forecasting curve and the discounting curve. Let $\tau_{j,\text{IRS-float}}$ be the accrual factors for the floating payment periods, calculated using the day count conventions of the floating leg of the instrument, and let $\mathbb{E}^{T_{j,\text{for}}^{\text{float}}}[r_j^*]$ be the expectation of the floating rate to be paid at S_j^{float} for each j, so fixed at S_{j-1}^{float} minus the spot lag.

The swap rate is set so that the initial value of the instrument is 0. Equating the fixed and the floating payments, we find

$$\sum_{i=1}^{n} \mathcal{C}_{\text{dis}}^d(T_{i,\text{dis}}^{\text{fix}})\, N\, r_{\text{IRS}}\, \tau_{i,\text{IRS-fix}} = \sum_{j=1}^{m} \mathcal{C}_{\text{dis}}^d(T_{j,\text{dis}}^{\text{float}})\, N\, \mathbb{E}^{T_{j,\text{for}}^{\text{float}}}[r_j^*]\, \tau_{j,\text{IRS-float}}, \tag{4.7}$$

where N is the notional amount.

Using equation (3.2), which in this case is a definition, we find for each j:

$$\mathbb{E}^{T_{j,\text{for}}^{\text{float}}}[r_j^*] = \frac{1}{\tau_{j,\text{IRS-float}}} \left(\frac{\mathcal{C}_{\text{for}}^d(T_{j-1,\text{for}}^{\text{float}})}{\mathcal{C}_{\text{for}}^d(T_{j,\text{for}}^{\text{float}})} - 1 \right). \tag{4.8}$$

This time we do not get the pleasure of the mass-elimination of a telescoping sum, but at least the accrual periods drop out of the right hand side when we combine equations (4.7) and (4.8). Rearranging to get a clean fixing equation, we find:

$$r_{\text{IRS}} = \frac{\sum_{j=1}^{m} \mathcal{C}^d_{\text{dis}}(T^{\text{float}}_{j,\text{dis}}) \left(\frac{\mathcal{C}^d_{\text{for}}(T^{\text{float}}_{j-1,\text{for}})}{\mathcal{C}^d_{\text{for}}(T^{\text{float}}_{j,\text{for}})} - 1 \right)}{\sum_{i=1}^{n} \mathcal{C}^d_{\text{dis}}(T^{\text{fix}}_{i,\text{dis}})\, \tau_{i,\text{IRS-fix}}}. \tag{4.9}$$

4.4 Synthetic instruments

In the case of an IBOR curve you want all the market instruments to be related to the IBOR index of that specific tenor, so the first instrument you can use is the deposit with as maturity that tenor. To facilitate the discussion, let us consider the EUR6M curve. The first instrument with as underlying interest rate the EUR6M index is exactly the EURIBOR 6 month deposit. The only information available before the 6 month point on the curve is the discount factor of 1 for $t = 0$. When you bootstrap for example a 1x7 FRA onto the curve, you use the discount factor for the 1 month point on the curve and the market quote of the FRA to determine the 7 month discount factor.

Algorithmically there is no problem with only using interpolated information between the $t = 0$ and $t = 0.5$ point, this is a perfectly valid way of encoding the provided market information. *Functionally* there are some remarks to be made about this approach. The information you can extract from the short end of the bootstrapped curve will only be driven by the choice of interpolation algorithm. If the goal is that this short end of the curve is a representation of the market, one needs to introduce synthetic instruments.

At Bloomberg they use a method based on an exotic FRA interpolation scheme. Ametrano and Bianchetti (2013) also talk about a method to add synthetics based on FRAs, and also mention a method of adding synthetic instruments based on deposits. I was not convinced by these methods.

I was however inspired by this to propose another way of defining the short end of an IBOR curve by using the short end of the OIS curve and a linear spread.

I start by determining the first pillar of the forecasting curve and also the corresponding point of the discounting curve:

$$\big(T_{1,\text{for}},\ \mathcal{C}_{\text{for}}(T_{1,\text{for}})\big) \qquad \text{and} \qquad \big(T_{1,\text{for}},\ \mathcal{C}_{\text{dis}}(T_{1,\text{for}})\big).$$

I interpret the spread as a linear function of time with 0 intercept:

$$\Delta = \big(\mathcal{C}_{\text{for}}(T_{1,\text{for}}) - \mathcal{C}_{\text{dis}}(T_{1,\text{for}})\big)/T_{1,\text{for}}.$$

Then I get the pillars of the discounting curve *before* the first point of the forecasting curve, I correct them for the spread and add these corrected pillars to the forecasting curve:

$$\{\big(T_{i,\text{dis}},\ \mathcal{C}_{\text{dis}}(T_{i,\text{dis}}) + \Delta \cdot T_{i,\text{dis}}\big)\,|\,T_{i,\text{dis}} < T_{1,\text{for}}\}.$$

This idea is very natural and might have been explored before, but I was not able to find a reference of this technique in the literature.

This synthetics method is called "ois" in RateHike. If you want to bootstrap a curve without this procedure, you should use synthetics method "no", which is the default.

Remark. The "ois" method only applies to the bootstrapping of IBOR curves, using information from the exogenous discounting curve. It does not apply to the bootstrapping of OIS curves.

4.5 Anchoring

Anchoring the zero coupon curve means choosing the spot date of the curve, the date for which $t = 0$.

Two logical choices are today's date and the spot date of the financial instruments. Of course, for the GBP curves there is no difference as they have a spot lag of 0 days, so let us consider the EURO curves with a spot lag of 2 business days. Suppose we are Wednesday 2022-11-30. The first instrument to bootstrap onto the curve is a deposit, either the 1 day deposit (for the ESTR curve), or the 3 or 6 month deposit (for the EURIBOR curves). These instruments have a spot date of Friday 2022-12-02. To bootstrap the deposit onto a curve with spot date 2022-12-02, we would use the deposit fixing equation (4.2). But to bootstrap it onto a curve with spot date 2022-11-30, we would use the FRA fixing equation (4.5). We would effectively treat it as a *forward deposit*.

So it is possible to choose other dates to anchor the curve. RateHike uses the spot lag as defined by the currency as a default value, but it is possible to override this behaviour by specifying another date.

One can also consider *re-anchoring* a curve. After having bootstrapped a curve with a spot lag of 2 business days, it can only be used to discount cash flows back to the spot date. If it is necessary to discount cash flows back to today's date, one can transform the curve for this purpose. In RateHike, three methods exist for this purpose: "date", "ontn" and "11". If you do not want to re-anchor the curve, you should use the re-anchoring method "no" (which is the default).

Method "date" just redefines the spot date. In RateHike, the time dimension of curves is indexed by the year fractions. So if one bootstraps a curve with spot date 2022-12-02, the 1 year discount factor represents the discount factor to translate a cash flow on 2023-12-02 to 2022-12-02. If we would just *relabel* the spot date of the curve to 2022-11-30, we would shift the entire curve by two days, and the 1 year discount factor would apply to a cash flow on 2023-11-30 to translate it to 2022-11-30. By doing this, one hopes the error made by shifting everything with 2 days is not too big.

The theoretically correct method of discounting a future cash flow back to 2022-11-30 when using a curve with a spot date of 2022-12-02 would be to consider an overnight deposit (starting today, maturing the next business day) and a *tomorrow-next* deposit (starting the next business day, maturing one business day later). The interest rates paid on these instruments would be the same as the 1 day deposit used in bootstrapping the OIS curve. Using these two instruments, one can determine the discount factor that applies between 2022-11-30 and 2022-12-02. We can use this discount factor to transform the curve: rather than just reinterpreting the year fractions, we convert them to the dates to which they apply, and recalculate them based on today's date, which is the new spot

date of the curve. The discount factors of the original curve are then multiplied with the extra discounting for the first two days. As cherry on the cake we add the 1 day and 2 day points to the new curve, based on the overnight and the tomorrow-next deposit. This method is called "ontn" in RateHike (for OverNight/Tomorrow-Next).

Another solution to the puzzle could be to imagine the period between today and the spot date is a neutralized period, where interest rates and discounting have no role to play. This can be achieved by prepending two pillars to the curve (for today and the next business day) with discount factors of 1. This way time will still play its role in the neutralized period, but interest rates do not. This method is called "1-1" in RateHike.

Remark. These methods are relevant when using the curve for discounting. A curve used for forecasting should be anchored using the spot lag, and not re-anchored.

Chapter 5

Interpolation methods

In this chapter I will discuss the interpolation methods available for bootstrapping an interest rate curve. I will restrict myself to the most commonly used interpolation schemes. The discussion here and the implementation heavily lean on Hagan and West (2006, 2008).

In general, when thinking about interpolation, one has a number of data points (x_i, y_i) and one looks for a function assigning values to the points inbetween. Theoretically, this is no different for interpolating an interest rate curve, but as explained in section 4.2 the bootstrapping algorithm already uses the interpolation scheme when building the curve. So the interpolation algorithm contributes to determining the pillars of the curve. In appendix A you can find an example showing this effect. This makes choosing an interpolation scheme for zero coupon curves quite different from just choosing a scheme for a fixed set of points.

Nevertheless, in this chapter I will present the interpolation schemes for a general set of points $(x_i, y_i)_i$. As with the fixing equations it is the interaction with the solving approach to bootstrapping that makes the interpolation contribute to determining the pillars, this does not need to be taken into account when implementing the interpolation method.

The first section discusses how to choose between all the interpolation schemes. The second section treats the interpolation data types. Then there is a section on linear interpolation and one on cubic spline interpolation. I end with a section that briefly discusses interpolation schemes that are not implemented in RateHike.

5.1 Choosing between interpolation methods

In this thesis I will only choose an interpolation method as part of a global search between all combinations of choices to make for a zero coupon curve. This will be discussed in section 7.1.

Hagan and West (2008, Table 1 p79) do give a table with a qualitative comparison between interpolation methods. I do not reproduce that here, since I do not know how it was created, but it is clear from that table that choosing an interpolation algorithm is a trade-off between a number of desirable properties.

The interpolation methods considered for RateHike all have a decent overall score the

table of Hagan and West (2008).

Comparing Adams (2001) and Hagan and West (2006, 2008), one notices the former article puts a prime on *global interpolation schemes*, which means using information of all pillars to determine the interpolation function for each subinterval, while the latter articles emphasize the importance of *locality* of interpolation schemes, which means using as little pillars as possible in each subinterval.

I agree with Hagan and West (2006, 2008): when considering a interest rate curve *locality* is clearly a desirable property. In a global interpolation scheme, changing any pillar on your curve will impact the entire curve. In a local scheme only the neighbouring intervals will be impacted. Also when considering bootstrapping locality is desirable: when adding an extra pillar onto a curve it not logical to impact the entire curve. A global interpolation scheme implies the *round-trip test* as introduced in subsection 7.1.2 would not work.

RateHike only implements local interpolation schemes.

5.2 Interpolation data type

For an interest rate curve, one can choose between multiple quantities to interpolate on: the main options are the zero rate $\mathcal{C}^z(T)$, the discount factors $\mathcal{C}^d(T)$ and the logarithm of the discount factors $\log \mathcal{C}^d(T)$. These are also the possibilities implemented in RateHike.

Historically, a fourth option was taken into account: interpolation on the logarithm of the zero rates. Since we have had prolonged periods of negative interest rates (and also of the bootstrapped zero rates) for the main currencies, I think this practice has to be abandoned.

Sometimes interpolation schemes are defined on the product $\mathcal{C}^z(T)T$, but this is equivalent to interpolating on $\log \mathcal{C}^d(T)$ since $\log \mathcal{C}^d(T) = -\mathcal{C}^z(T)T$. In the literature one also finds interpolation schemes on the instantaneous forward rates. I consider these to be exotic, see section 5.5.

5.3 Linear interpolation

Linear interpolation is a very easy interpolation scheme. Given a set of points $\{(x_i, y_i)\}_{i=1}^n$, one connects these points with a straight line to determine the intermediate values. Mathematically, for $x \in [x_{i-i}, x_i]$, one assigns the value

$$y(x) = \frac{x_i - x}{x_i - x_{i-1}} y_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} y_i = y_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} (y_i - y_{i-1}). \tag{5.1}$$

In RateHike this method is called"linear".

It is well known that a linear interpolation on the log of the discount factors implies a step function behaviour for the instantaneous forward rates, and a linear interpolation on the zero rates implies a "saw-tooth" behaviour for the instantaneous forwards.

This makes the *visual test* as defined in section 7.1 a useful tool for choosing between interpolation schemes: one can immediately spot the effects on the derived quantities.

5.4 Cubic spline interpolation

The main goal of cubic spline interpolation is to obtain a smoother interpolation than linear interpolation. In the context of interest rate curve interpolation one specifically wants to avoid the artefacts discussed at the end of the previous section.

Remark. From a mathematical perspective, the interpolation methods concerned cannot be called "cubic splines". This name is given to a piecewise third degree polynomial interpolation where the first *and* second derivatives are continuous. As can be seen from the conditions in subsection 5.4.1 the methods concerned only require the first derivative to be continuous. However, in the context of interest rate curve interpolation the abuse of this term is common. Lacking a better term, I will also use "cubic spline" for these methods[1].

This section is based on Hagan and West (2006). In the first subsection I give a general framework for cubic splines, the following subsections will discuss concrete implementations.

5.4.1 General framework

Take n points $\{(x_i, y_i)\}_{i=1}^n$. In each interval $[x_i, x_{i+1}]$ for $i = 1, \ldots, n-1$ a third degree polynomial will be used to determine the intermediate values:

$$a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{5.2}$$

The first set of conditions will be imposed on all implementations:

conditions

1. The value on the left of each interval $[x_i, x_{i+1}]$ is equal to y_i.
2. The value on the right of each interval $[x_i, x_{i+1}]$ is equal to y_{i+1}.
3. On the interior points $i = 2, \ldots, n-1$ the left derivative is equal to the right derivative.

Together, these are $3n - 4$ conditions on $4n - 4$ coefficients, leaving n conditions to be imposed.

Expressing conditions 1 and 3 in terms of the coefficients, we find

$$a_i = y_i, \tag{5.3}$$

$$b_{i+1} = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2, \tag{5.4}$$

where equation (5.3) holds for $i < n$ and equation (5.4) holds for $i < n - 1$. We also use these equations to define numbers a_n and b_n, extending (5.3) to $i = n$ and (5.4) to $i = n - 1$.

[1] In de Boor (2001, p105) you can find a nice reflection of an authority in the domain of splines struggling with the same subject. He decides against calling them cubic splines.

Working out condition 2 gives

$$a_{i+1} = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3, \tag{5.5}$$

for $i < n$.

Using simple algebra on equations (5.4) and (5.5) one can derive the following relations for $i = 1, \ldots, n-1$:

$$\begin{aligned} m_i &= \frac{a_{i+i} - a_i}{x_{i+1} - x_i}, \\ c_i &= \frac{3m_i - b_{i+1} - 2b_i}{x_{i+1} - x_i}, \\ d_i &= \frac{b_{i+1} + b_i - 2m_i}{(x_{i+1} - x_i)^2}, \end{aligned} \tag{5.6}$$

The cleverness of this approach is that it can be turned around. Given the points $\{(x_i, y_i)\}_{i=1}^n$, define a_i using (5.3) and choose n values b_i. If you calculate the coefficients c_i and d_i using (5.6), then a_i, b_i, c_i and d_i together form the coefficients for a set of spline functions with the conditions 1–3 as described at the start of this subsection.

So the b_i can be interpreted as the n conditions still to be specified. This makes specifying the splines rather mechanical, which is quite convenient in this context. The following subsections will contain different ways of specifying b_i.

5.4.2 Bessel splines

According to Hagan and West (2006), there is some confusion between *Bessel interpolation* and *Hermite interpolation* in the industry. The Hermite method uses both function values and function derivatives to define an interpolation. The Bessel method first fits a quadratic polynomial through every 3 consecutive points and then uses the derivatives of this quadratic interpolation as input for Hermite interpolation. That is the theoretical background of the method; the equations for the b coefficients in the Bessel scheme are

$$\begin{aligned} b_1 &= \frac{1}{x_3 - x_1}\left(\frac{(x_3 + x_2 - 2x_1)(y_2 - y_1)}{x_2 - x_1} - \frac{(x_2 - x_1)(y_3 - y_2)}{x_3 - x_2}\right), \\ b_i &= \frac{1}{x_{i+1} - x_{i-1}}\left(\frac{(x_{i+1} - x_i)(y_i - y_{i-1})}{x_i - x_{i-1}} + \frac{(x_i - x_{i-1})(y_{i+1} - y_i)}{x_{i+1} - x_i}\right), \\ b_n &= \frac{1}{x_n - x_{n-2}}\left(\frac{(2x_n - x_{n-1} - x_{n-2})(y_n - y_{n-1})}{x_n - x_{n-1}} - \frac{(x_n - x_{n-1})(y_{n-1} - y_{n-2})}{x_{n-1} - x_{n-2}}\right), \end{aligned} \tag{5.7}$$

where the middle equation applies to $1 < i < n$. Together with (5.3) and (5.6) these equations calculate all the spline coefficients.

RateHike refers to this method as "bessel".

5.4.3 Hyman splines

This method is called *Monotone Preserving Cubic Spline* in Hagan and West (2006), but I do not think this is standard terminology. I prefer to refer to the method as Hyman. It is based on Hyman (1983), but multiple interpretations of that article are possible. For instance in the R package "stats" (R Core Team, 2022), the function "splinefun" has a method "hyman", which only applies to strictly monotonic inputs.

The version as described by Hagan and West (2006) mixes two interpolation schemes, one for when the inputs are monotonic around the point in question, and one for when they are not.

For the endpoints, Hyman (1983) defines

$$
\begin{aligned}
b_1 &= \frac{(-2x_1 + x_2 + x_3)m_1 - (x_2 - x_1)m_2}{x_3 - x_1}, \\
b_n &= \frac{(-x_{n-2} - x_{n-1} + 2x_n)m_{n-1} - (x_n - x_{n-1})m_{n-2}}{x_3 - x_1},
\end{aligned}
\tag{5.8}
$$

where the m_i are borrowed from (5.6). Hagan and West (2006) propose another version where the b coefficients are just 0 for the endpoints:

$$
b_1 = b_n = 0. \tag{5.9}
$$

If either $y_{i-1} \leqslant y_i \leqslant y_{i+1}$ or $y_{i-1} \geqslant y_i \geqslant y_{i+1}$ for $1 < i < n$, the curve is said to be locally monotonic at x_i. In this case you can calculate b_i ensuring the interpolation is also monotonic: first define

$$
b_i^* = \frac{3m_{i-1}m_i}{m_{i-1} + m_i + \min(m_{i-1}, m_i)}, \tag{5.10}
$$

and then apply the correction

$$
b_i = \begin{cases} \min(\max(0, b_i^*), 3m_{i-1}, 3m_i) & \text{if the curve is locally increasing at } x_i, \\ \max(\min(0, b_i^*), 3m_{i-1}, 3m_i) & \text{if the curve is locally decreasing at } x_i. \end{cases} \tag{5.11}
$$

If the curve is not locally monotonic at x_i, define

$$
b_i = 0. \tag{5.12}
$$

The spline coefficients for Hyman are defined by equations (5.3), (5.6), (5.10), (5.11) and (5.12), together with endpoint conditions (5.8) or (5.9).

With endpoint conditions (5.8) RateHike refers to the method as "hyman", and with (5.9) as "hyman0".

5.4.4 Corrected cubic splines

When considering the different cubic spline versions for a zero coupon curve implementation, the discrete and instantaneous forward rates in the last interval had big swerves.

This is documented in chapter 7, see subsection 7.2.2; specifically the right-hand sides of figures 7.1 and 7.2.

This incited me to try to apply a correction to the spline implementations. The method I tried was very simple and can be explained in 1 sentence: in the last interval the cubic spline interpolation is replaced by a linear interpolation. Despite this simplicity, the results were very good.

This idea is very natural and might have been explored before, but I was not able to find a reference of this technique in the literature.

5.5 Other interpolation schemes

Many other interpolation schemes exist. I will consider natural splines, financial splines, and forward monotone convex splines.

For both the natural and the financial cubic splines, one requires that the global function is twice differentiable, which are $n - 2$ extra conditions (on top of those mentioned at the start of subsection 5.4.1). For the natural spline one also requires that the second derivative at the endpoints is 0, for the financial spline one requires the first derivatives at the endpoints to be 0.

These two methods are *global* methods: a change in one point will impact all polynomials. As explained in section 5.1 I only considered *local* methods.

People have also tried to devise interpolation schemes on the instantaneous forward rates directly. Simple schemes on these rates fail miserably, see for example the method Piecewise Linear Continuous Forwards in Hagan and West (2006, p97).

In Hagan and West (2006, p108) another method is proposed, which is called *Forward Monotone Convex Spline*. This method is very involved, and imposes quite some structure on the curve, both implicitly and explicitly. Moreover, even if the method is local with respect to instantaneous forwards, recovering the zero rates or discount factors requires integration from the start of the curve to the point for which the zero rate or the discount factor is needed. So for me this is a global method.

Chapter 6

Implementation: RateHike

This chapter contains details about the implementation of RateHike: argumentations why certain choices were made, the structure of the objects defined, and an explantion how the code performs the bootstrapping.

There is some overlap between this chapter and the documentation of the code. The documentation is meant for users to learn how to use it, this chapter is more about explaining the low-level details and the link with the theory in the previous chapters.

6.1 Getting started

This section describes how to get the code and how to load it in an R session.

The code is available on GitHub (Helsen, 2023). The version used for producing the results in this thesis is labelled "thesis-final". To use the code, there are 3 options, you can either manually download the code and *source* it or *attach* it, or you can *install* it directly (without manually downloading).

To install the package, you need the "devtools" package, and in an R session you give the command `devtools::install_github("spuddie/RateHike")`. This needs to be done only once. To load the package you do `library(RateHike)` at the start of each session.

When you go the other way and download the package manually, you can source it by first setting the correct working directory: `setwd("/path/to/RateHike")`, and then use `source("inst/nopkg/_GLOBAL_.R")`. To attach the downloaded package you need the command `source("inst/nopkg/_ATTACH_.R")`.

Each approach has its advantages and disadvantages. Installing the package is the cleanest, sourcing it allows to run the code in debug mode so that you can investigate what is happening under the hood and the attach option is a bit of a hybrid; this sources the code, while hiding all the created functions. This can be used if you have modified the code for instance. Another advantage of sourcing or attaching over installing is that you can directly use every function defined in the package, not only the functions that have been explicitly exported. This is a small advantage, as the functions that are not exported are technical helpers which are not directly useful to the end user.

6.2 Implementation choices

For the implementation I chose R Statistical Software (R Core Team, 2022, v4.2.2). R is a nice prototyping environment, it has a nice high-level language interface while at the same time allowing low-level hacks. The fact that it was available at Triodos was also important.

I chose not to use any third-party packages. This goes somewhat against the grain of the R philosophy, but I wanted to make a fully transparent implementation. Another argument is that a package without dependencies is more future proof.

For bootstrapping, using object-oriented programming (OOP) is very nice. For example, you can loop over the list of instruments, and call `fixing(inst_list[[i]], curve)`. The OOP dispatching will then make sure each instrument uses the correct `fixing` method.

In R, there are multiple object-oriented systems to choose from: S3, S4 and R6 (Wickham, 2019). I chose S3, which is described as informal, ad hoc and minimal in the previous reference. It is somewhat different from OOP for most other languages; traditionally methods belong to objects, in R S3 methods belong to generic functions. To continue the previous example: you define a generic function `fixing` and you assign methods like `fixing.rh_ois` and `fixing.rh_fra` to this generic. If you then call the generic function `fixing` on an object with class **rh_ois** or **rh_fra**, the dispatcher will select the correct method. In this setup it is quite logical for an object to have more than one class. For example, rather than dispatching on the instrument type to apply the correct day count conventions, I assign another class to each instrument to indicate which day count convention should be applied: **rh_act365**, **rh_act360** or **rh_30e360**. The full list of classes for the instruments and the curves is discussed in section 6.3.

For the code documentation I used Roxygen2 (Wickham et al., 2022). The code is also protected by unit tests, using the testthat package (Wickham, 2011). The tests are executed automatically by GitHub Actions each time a change is made. These test runs are done across multiple platforms and multiple versions of R.

One of the goals of the implementation was simplicity, in order to transparently show all the details of the bootstrapping. This means for example that the code does not contain functions to calculate the valuation of financial instruments. The only thing needed for bootstrapping is the ability to *fix* the financial instruments, which means calculating the interest rate that applies.

To add synthetic instruments, or to do the re-anchoring, some complex code has been added to RateHike, but all of this is optional; avoiding these advanced features is the default, a user needs to specifically choose to use these parts of the code.

6.3 Objects

As a naming convention, all classes in RateHike start with "rh_".

RateHike assigns a lot of classes to each object. The primary classes are **rh_curve**,

rh_instrument and **rh_instrument_list**, representing a zero coupon curve, a financial instrument, and a list of financial instruments respectively. Details on the structure of each object can be found from within R with the command `?`, for example `?rh_curve` (after loading the package).

6.3.1 The curve

A typical curve will have up to 7 classes assigned, based on the choices made in the settings supplied. For example:

rh_curve The main class indicating this is a zero coupon curve.

rh_act365 The day count convention assigned.

rh_interp_logdf The interpolation data type.

rh_interp_linear The interpolation method.

rh_anchor_no The re-anchoring scheme to apply.

If the chosen interpolation scheme was "spline", then the class **rh_interp_linear** would be replaced by 2 or 3 other classes: the literal **rh_interp_spline**, a class for the choice of spline function (like **rh_interp_bessel**), and possibly a class indicating to apply the spline correction as discussed in subsection 5.4.4: **rh_interp_splinecorr**. Each of these classes is used in some dispatching mechanism.

The curve object has a lot of fields specifying things like the today date and the spot date. It also contains information about the pillars and the interpolation coefficients. For that, a choice needs to be made how to store this data. In RateHike the data stored is determined by the interpolation data type as chosen by the user. This makes interpolating easy: the interpolation function does not know the data type it operates on. When storing and retrieving data the conversions are applied (if necessary).

When storing zero rates, one could choose to store them as numbers, or as percentages. When storing this type of data as floating point numbers, both choices are equivalent. In Ratehike, interest rates are always treated as numbers, so 0.02 is used to store an interest rate of 2%.

To index the data, RateHike uses the year fractions as calculated by the curve day count convention. Another choice would be to use the specific dates. The only effect of this choice is when the conversion between dates and year fractions needs to be applied: when storing the data, or when retrieving it. I am convinced that indexing the data by year fractions is more efficient, but I have not investigated this; the difference will in any case be academic.

6.3.2 Instruments and instrument lists

Instruments are assigned up to 6 classes. For example:

rh_instrument The main class indicating this is a financial instrument.

rh_deposit The type of instrument.

rh_act360 The day count convention assigned.

rh_modfollowing The rolling convention to apply.

rh_end_of_month Only set if the end-of-month convention applies.

For an IRS, an extra class is assigned to indicate whether the fixed leg pays yearly (**rh_sched_yearly**) or whether it follows the floating leg payments (**rh_sched_tenor**).

An instrument list only has two classes assigned, the literal **rh_instrument_list**, and a class indicating whether synthetic instruments should be added (e.g. **rh_snt_ois**).

6.4 Functions and methods

As a naming convention, all global functions in RateHike start with "rh_". S3 generic functions do not follow these conventions as the assigned methods already use the class names starting with "rh_".

bootstrap

The `bootstrap` function is the main function of the package. It uses the function `try_pillar` in combination with the standard R function `uniroot` to determine the next pillar; this constitutes the inner level of the bootstrap algorithm (algorithm 1 p.25).

The `bootstrap` function also performs the outer and middle levels of algorithm 1: it iterates through the instrument list, adding one pillar for each instrument. For linear interpolation, one such iteration is sufficient. For spline interpolation, one needs to iterate multiple times through the instrument list, as new points influence the interpolation coefficients of points added before. For each iteration RateHike compares the discount factors to the values of the previous iteration. When this set of discount factors converges, the algorithm stops.

try_pillar

The auxiliary function `try_pillar` has as inputs a curve, an instrument, a discount factor and a pillar. It will use the function `update_discount_factors` to try the new discount factor for the given pillar on the curve. It then returns the difference of the market quote and the fixing of the instrument. This auxiliary function is used within to the standard R function `uniroot` to find the value for the pillar that results in the market quote when fixing the instrument.

update_discount_factors

The function `update_discount_factors` uses the class of the curve to determine how the data should be written. Then it calls the function `calc_interp_coeff`. This latter function calculates the other interpolation coefficients. It is necessary

to do this calculation each time a value changes, since this interpolation is used immediately after. To do the calculations as efficient as possible, the changed pillar is passed to `calc_interp_coeff` and only the impacted coefficients are recalculated.

calc_interp_coeff, calc_spline_coeff_b

`calc_interp_coeff` is used to calculate the interpolation coefficients when the curve data gets updated. It does so for linear interpolation (if the curve has class **rh_interp_linear**), and it also directly calculates the spline coefficients a, c and d (if the curve has class **rh_interp_spline**). This is possible since this is part of the general spline framework as set out in subsection 5.4.1. To calculate the b coefficient it calls `calc_spline_coeff_b`, which uses the class based on the specific spline interpolation to decide how this needs to be calculated (**rh_interp_bessel**, **rh_interp_hyman** or **rh_interp_hyman0**). It also takes into account whether the spline correction needs to be applied or not.

fixing

The `fixing` function has two inputs: a financial instrument and a curve. It dispatches to a separate method for each instrument type. These methods each implement one of the fixing equations of chapter 4: (4.2) for deposits, (4.4) for OISs, (4.5) for FRAs, (4.6) for futures and (4.9) for IRSs.

add_synthetics

The `add_synthetics` function takes in an instrument list, and adds synthetic instruments to this list as explained in section 4.4.

re_anchor

Re-anchoring is discussed in section 4.5. The function `re_anchor` uses the class of a curve to decide which option to apply. It has a second argument, to pass the 1 day deposit for the option "ontn". For the other options this second parameter is not needed.

daycount

The `daycount` function has 3 arguments. The first argument can be a curve or an instrument, and is only used to dispatch the function to the correct method. The other two arguments are from-date and to-date to define the period on which to calculate the year fraction.

get_curve_data, get_discount_factors, get_insta_fwd, get_zero_rates

These functions have a curve as input, and either a set of dates, a set of times, or a set of pillars. If no second input is given, all pillars will be returned. `get_curve_data` is a technical function: it just returns data from the curve, applying the interpolation if necessary. It does not take into account what data it is working on. `get_discount_factors`, `get_insta_fwd` and `get_zero_rates` are well-named functions: they extract a certain type of information from a curve. They call `get_curve_data` and apply the correct transformation, based on how the data is stored within the curve and which information is requested.

graph

The `graph` function makes a nice plot of a zero coupon curve: it plots the zero

coupon rate, the discrete forward rate and the instantaneous forward rate (all on the left Y-axis), and also the discount factors (on the right Y-axis). By default it plots the entire curve, but the parameter `max_t` allows to plot only part of the curve. You can also specify exactly the times vector you want on the X-axis, or the number of points to be used. You can also set the forwarding period for the discrete forward rates.
The curve graphs in the next chapter were all produced by this function.

benchmark, leave_one_out, roundtrip
These functions implement the tests as described in subsection 7.1.2.

busday_adjust, calc_next_date, calc_schedule
These are technical functions applying the correct conventions to calculate sets of dates.

set_bootstrap_options, set_interpolate, set_times
These are technical functions that complete the structure of a zero coupon curve object, based on the settings given as input.

set_conventions, set_mat_date, set_schedule, set_spot_date
These are technical functions that complete the structure of market instruments, based on the settings given as input.

get_rh_param, set_rh_param
These functions query and change the values of the configuration parameters.

6.5 Configuration parameters

The RateHike package has 4 configuration parameters.

The parameter `root_find_tol` is initialized at 10^{-12}. This parameter determines when the algorithm is satisfied when determining the next discount factor while processing an instrument in the inner level of algorithm 1 (p.25).

The parameter `root_find_interval_radius` is also used for this root finding step (inner level of algorithm 1): rather than passing an initial guess for the next discount factor, an interval is passed in which we expect the next discount factor will be found. In practice the algorithm does make an initial guess, and the radius determined by this parameter is added on each side. The root finding algorithm is smart enough to enlarge the interval if no root is found; this parameter only impacts the performance. The value is initialized at 10^{-3} resulting in each root to be found in a small number of function evaluations (typically 5–10).

The parameter `spline_update_diff_tol` impacts how many times the algorithm loops through the instrument list when bootstrapping with a spline interpolation (outer level of algorithm 1). Once the discount factors are not updated by more than this parameter, the algorithm considers the bootstrapping as finished. RateHike has an initial value of 10^{-12} for this parameter.

Finally, for the same algorithm a maximum number of iterations is fixed by the parameter `spline_max_sweeps` (outer level of algorithm 1). It is initialized at a value of 10. Using a range of interpolation specifications and a variety of market data, the bootstrap routine needs 5, 6 or 7 sweeps to achieve the threshold.

Setting the parameters means making a choice on the accuracy / performance trade-off. Taking into account that machine precision is of the order of 10^{-16} when using floating point numbers, I think the tolerances are small enough. In terms of performance, on an ordinary laptop each bootstrap takes 2-3 seconds, so there was no reason to reduce the accuracy. Given this small runtime, I have not tried to optimize the parameter `root_find_interval_radius`.

6.6 Theoretical considerations

This section gathers two minor remarks on how the implementation and the theory are connected.

6.6.1 From single-curve to multi-curve

Looking at the shift from the single-curve to the multi-curve framework from an implementation perspective, the impact is minor: only one line of code is needed to handle this shift. Indeed, the only fixing equation where both the discounting and the forecasting curves are needed is equation (4.9) for the IRSs. Implementing this equation one needs to ensure the correct curve is used to discount the cash flows: the exogenous discounting curve or the endogenous single curve, but that is the only difference.

This does not reflect the impact of the shift from the single-curve to the multi-curve framework on the domain of fixed income, which was quite considerable. Also, the fact that only one line of code is impacted is true for this specific bootstrapping implementation, but other approaches (such as the linear algebra approach, see subsection 1.6.2) have been impacted to the point that they are not valid any more!

6.6.2 The pillar finding strategy

As was mentioned in section 6.4, the R function `uniroot` is used to solve for each pillar. This function implements a one dimensional root finding algorithm. In this section I discuss why this is a valid strategy.

Alternative strategies are to minimize the absolute or squared difference between the fixing outcome and the market quote. As most reference works on numerical algorithms will tell you, finding the root of a monotonous function is much simpler than minimizing its square or its absolute value. The reason for this is that you can *bracket* a root between a positive value and a negative value and reduce the bracket. See for example Press et al. (2007).

In the previous paragraph I claimed the fixing equations are monotonous as a function of the discount factor to be added. I will build an argument on the interpolated discount

factors in the last interval to reach that conclusion. It is nice to take a step back and consider what we are investigating. The equations in this thesis were all constructed for the direct problem of fixing financial instruments on a curve, but here we need to consider the inverted problem.

So put yourself in the situation where you have a partially bootstrapped curve and you want to add a new pillar to it. The financial instrument that is being presented for this task might need the discount factors of multiple points on the curve. For information up to the pillar that was added previously, there is no problem whatsoever, you can just read this value of the intermediate version of the curve. But the instrument might also need information from the last interval. So we consider these interpolated values both as a function of time and of the value of the endpoint of the interval.

Investigating equation (5.1) for linear interpolation, and equations (5.3), (5.6) and (5.7) for Bessel interpolation, you can see that the interpolation coefficients in each interval are linear functions of the endpoint. This means that when you take a fixed point $x^* \in [x_i, x_{i+1}]$ the interpolated value $y(x^*)$ is a linear function of y_{i+1}.

Assume the interpolation data type is the discount factor. Inputting this linear relationship in the fixing equations for the deposit, the OIS, the FRA and the future, and taking the derivative, we easily find the fixing equation is monotonic. For the IRS the case is more complicated. In a linear interpolation setting, one can use the fact that the discount factors are monotonous in the last interval to prove the fixing formula is monotonous, but in Bessel interpolation this monotonicity does not necessarily hold. However, using the fact that the derivative at the left hand side of the last interval is the slope of the quadratic function through the last three points, one can show the discount factor is "nearly" monotonous in this last interval. Then one can look carefully at the fixing equation: this uses equally spaced discount factors in the last interval. In order for this equation *not* to be monotonous, the discount factors in the last interval should be balanced on the increasing part and the decreasing part. Setting up this analysis carefully makes this incompatible with the "near monotonicity".

So now I have argued all fixing formulas are monotonic in the discount factor we are trying to add, when using the discount factor as interpolation data type for linear and Bessel interpolation. To prove the same thing when the zero rates and the logarithm of the discount factors are used as interpolation data type, it suffices to note that the transformations required to translate these data types into discount factors are monotonic.

To prove the same thing when Hyman interpolation is used is more tricky. It has monotonicity built in, but the dependence of the interpolated values in the last interval on the right hand side is not linear; in essence it is quadratic, and a number of corrections are applied based on certain conditions. To prove monotonicity of the pricing equations in this case, one would need to carefully analyse the different cases and the corrections applied. I did not complete this analysis. I was unconvinced by the results using Hyman interpolation, making this analysis less necessary. On the other hand: all calculations went fine, so I am quite convinced the strategy is applicable in this case too.

Chapter 7

Results

In this chapter I present the results of the bootstrapping implementation across different choices for the curve.

I start by describing the test approach: the choices considered and which tests were done. The analysis of the test results follows in the second section. The third section contains my preferred set of choices. The chapter ends with a section discussing the contributions of this thesis.

Obviously, all results and graphs are produced with RateHike.

7.1 Test approach

As was made clear in section 4.1, a lot of choices can be made about the properties of a curve. In order to determine the best set of choices, I did an exhaustive search across all reasonable combinations of choices. For all combinations I bootstrapped the OIS and the IBOR curves and calculated some metrics to help choose between all possibilities. In this section I describe the curves considered, the choices investigated, and the metrics applied.

7.1.1 Curves and choices considered

As explained in subsection 1.3.4, I consider the EURO curves ESTR, EUR3M and EUR6M, and the GBP curves SONIA and GBP6M. For all curves I used data of 2022-04-30, 2022-09-30 and 2022-11-30.

I took the curves to be continuously compounded, the day count convention to be ACT/365, and as input the same set as Bloomberg (see table 4.1 on p27).

The IBOR curves will be built using the OIS curve as exogenous discounting curve.

For all curves I tried 4 interpolation methods (linear, bessel, hyman, and hyman0) on 3 choices for the interpolation data type (the zero rates, the discount factors and the logarithm of the discount factors). The spline methods were applied with and without spline correction.

For the synthetic instruments I tried the options “no” and “ois” for the IBOR curves.

For the anchoring of the curves, I always anchored the GBP curves on the today date. The EURO curves were anchored at the today date plus 2 business days. I did not include re-anchoring in these tests; the choice for re-anchoring is a secondary choice about how to use the curve, not so much a primary choice in how to build the curve. Also, the metrics considered in the next subsection do not make sense on a re-anchored curve.

I did not try multiple values for the configuration parameters; I used the default values as described in section 6.5.

7.1.2 Test metrics

For some combinations of choices for the curve settings I had available the bootstrapped curve as calculated by Bloomberg. In this case I could *benchmark* the RateHike results to the Bloomberg results. In the early stages of the project I used QuantLib to calculate bootstrapped curves in parallel to the RateHike calculations, these results were also used to benchmark RateHike.

A second metric I consider is the *round-trip test*: after using a set of market instruments to bootstrap a curve, I fix this set of instruments on the curve and calculate the difference with respect to the market quotes. In a sense, the curve was constructed to produce exactly those quotes, so one could consider this test only to be a superficial check of the implementation. If the implementation is correct and convergence was reached in the bootstrapping algorithm, one expects this round-trip test result to be smaller than the parameter `root_find_tol`. Apart from showing problems with the implementation, this round-trip test also shows the importance of knowing all choices that were made during bootstrapping; if this is not the case, one cannot reproduce the market quotes of the input instruments!

Remark. As mentioned in section 5.1, one can only expect the round-trip test to give results within `root_find_tol` when choosing *local* interpolation methods, but all available methods in RateHike have this property.

The final metric I consider is borrowed from data science: the *leave-one-out test*. For a set of input instruments, one instrument is removed from the set before bootstrapping. After bootstrapping the curve, the removed instrument is fixed on the curve and the fixing is compared to the market quote. One cannot expect to achieve a precise fixing, but it is a useful comparison between different choices for the curve. To perform this test I chose 3 instruments for each set.

Another test might be to take a set of market instruments (not included in the input data set) and benchmark their fixing or pricing, but the data for performing this test was not available. In a sense, the leave-one-out test is a proxy for this test. The advantage of the leave-one-out test is that the instruments tested are representative for the curve by construction.

7.1.3 Visual test

Apart from the metrics described in the previous subsection, I also applied a *visual test*: the zero rates, the discount factors, the instantaneous forward rates and the discrete

forward rates are displayed graphically and inspected visually. This test is not very scientific, but it does allow a common sense check of the curves. It also allows to spot some artefacts that might appear in the forward rates for certain combinations of choices.

7.1.4 Qualitative evaluation: pricing an IRS

Another test that was applied was to take an example of an IRS with the EUR6M index as underlying and investigate in detail the valuation of this instrument: which floating rates were predicted by the bootstrapped IBOR curve and which discount factors came out of the bootstrapped OIS curve. These numbers were compared to alternative curves: the curves as provided by Bloomberg and a Quantlib setup.

The reason for calling this test "qualitative" is that it is only a one-off comparison that was used to gain insight in how the RateHike curves compare to other curves when used in a production setting.

This test was done together with Triodos. The test shows the importance of knowing the details of how the curve was built, without that knowledge it is impossible to get the proper information from it.

RateHike performed very well in a comparison with the curves currently used at Triodos. The work from this thesis also helped to better understand how the current curves should be configured and used.

7.2 Analysis

Now I present the results of the tests. The benchmarking and the round-trip test can be described in a short subsection. The biggest subsection with a lot of graphs is dedicated to the visual test, after which the leave-one-out test will be discussed, followed by a note on the performance.

7.2.1 Benchmarking and round-trip test

For most combinations of choices RateHike achieved a maximal benchmarking precision. For data from Bloomberg this was of the order of 10^{-7}, as the data was available up to six decimal digits. For the comparison with Quantlib the precision achieved was of the order of 10^{-13}, which is the accuracy as required by the configuration parameters.

The only exception to achieving the maximal benchmarking precision was for the EUR6M curves. For these Bloomberg uses a specialized set of synthetics that I was not able to reproduce.

As expected and required, all round-trip tests produce results within 10^{-12}, which was the setting used for parameter `root_find_tol`. It is not useful to analyse further which set of choices produced the best results, all results qualify as a success.

7.2.2 Visual test

The presentation of this subsection focusses on the EUR6M curve for 2022-11-30. This curve is the most important curve from the set of curves considered. When analysing the other curves and the other dates, the conclusions are the same, but these results are not shown.

In figures 7.1 and 7.2 you see the EUR6M curve of 2022-11-30 across all combinations of choices for the interpolation method and the interpolation data type, without application of the spline correction and without the addition of synthetic instruments.

The first thing you notice when looking at the graphs are the artefacts on the instantaneous forward rates and the discrete forward rates, which I will treat together as "the forwards". For linear interpolation the well-known saw-tooth and staircase artefacts clearly show. For bessel and hyman0 interpolation the behaviour of the forwards after 40 years is clearly less desirable. Looking at the chunky behaviour of these forwards for hyman and hyman0 between 10 years and 40 years I think this interpolation methods should be discarded. As they were specifically designed to preserve monotonicity I was expecting them to perform best, but looking at the results my conclusion is that their construction is too artificial.

One also notices the wild behaviour of the forwards on the short end.

The most important conclusion from looking at the graphs is that the zero rates and the discount factors are well behaved and pretty similar across all combinations. If you zoom in you start to see minor artefacts on the short end (for all combinations) and around the 18 year point (for all combinations except the bessel interpolation).

Another observation one can make is that the results for using the discount factors as interpolation data type on the one hand, and the logarithm of the discount factors on the other hand, are very similar for the spline methods.

Figure 7.3 shows linear interpolation and bessel interpolation with spline correction, both with the OIS synthetics added. The forwards are still pretty wild on the short end, but they are better behaved than without synthetic instruments. For the forward behaviour after the 40 year point one sees the bessel (corr) curve is better behaved than without the correction. The behaviour after correction pretty much resembles the behaviour of the curves with linear interpolation, as it was designed to do.

Remark. This behaviour in the last interval impacts my preferences as given in section 7.3, so it is the behaviour of the linear interpolation that decides which interpolation data type is used for the splines.

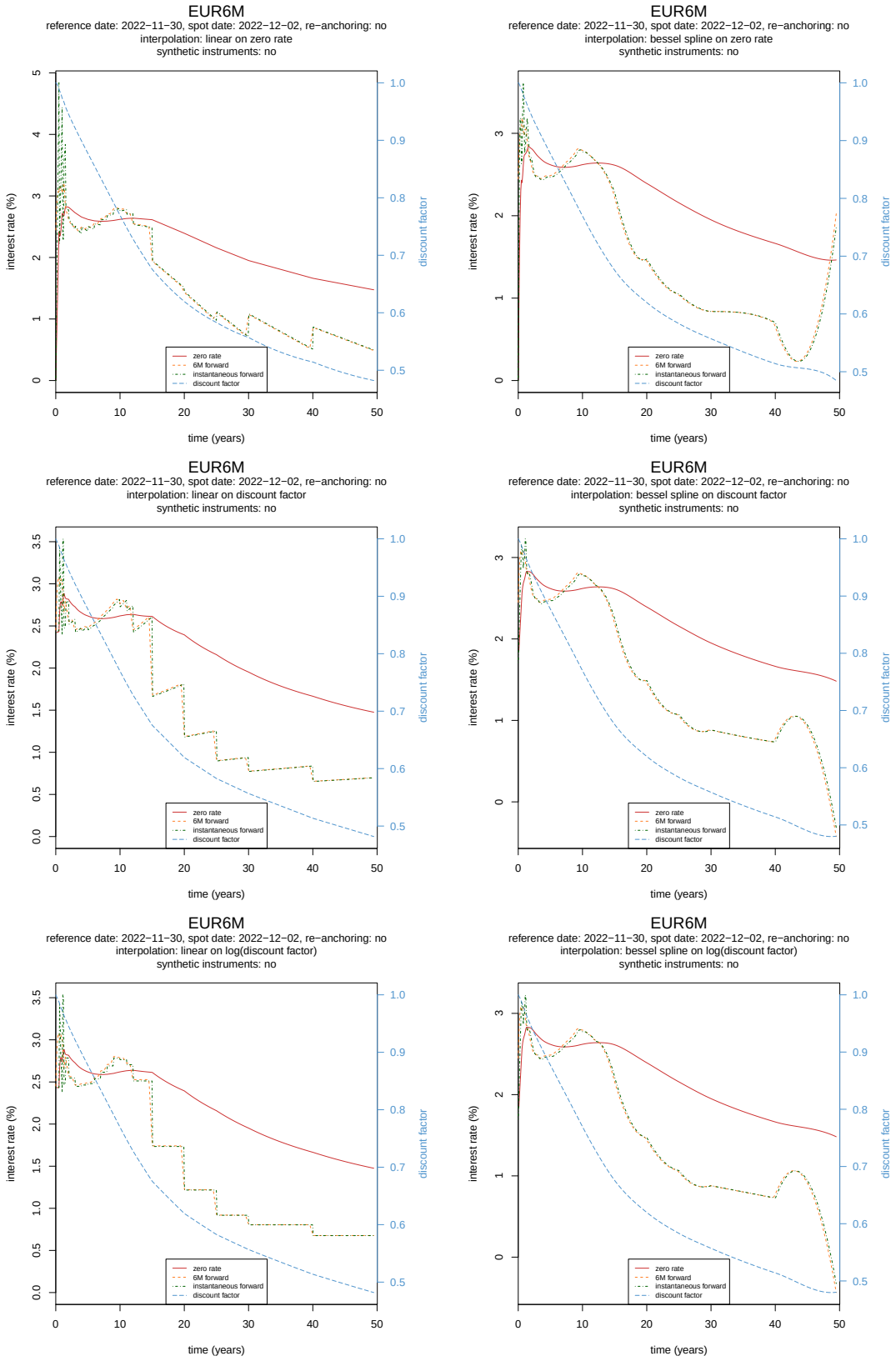

Figure 7.1: EUR6M curve for 2022-30-11 without synthetics. *On the left*: linear interpolation, *On the right*: bessel interpolation. *Top*: interpolation on the zero rates, *centre*: on the discount factors, *bottom*: on the log of the discount factors. On the left axis the zero rates, the discrete forward and the instantaneous forward rates are plotted, on the right axis the discount factors are plotted. Discussion in the text.

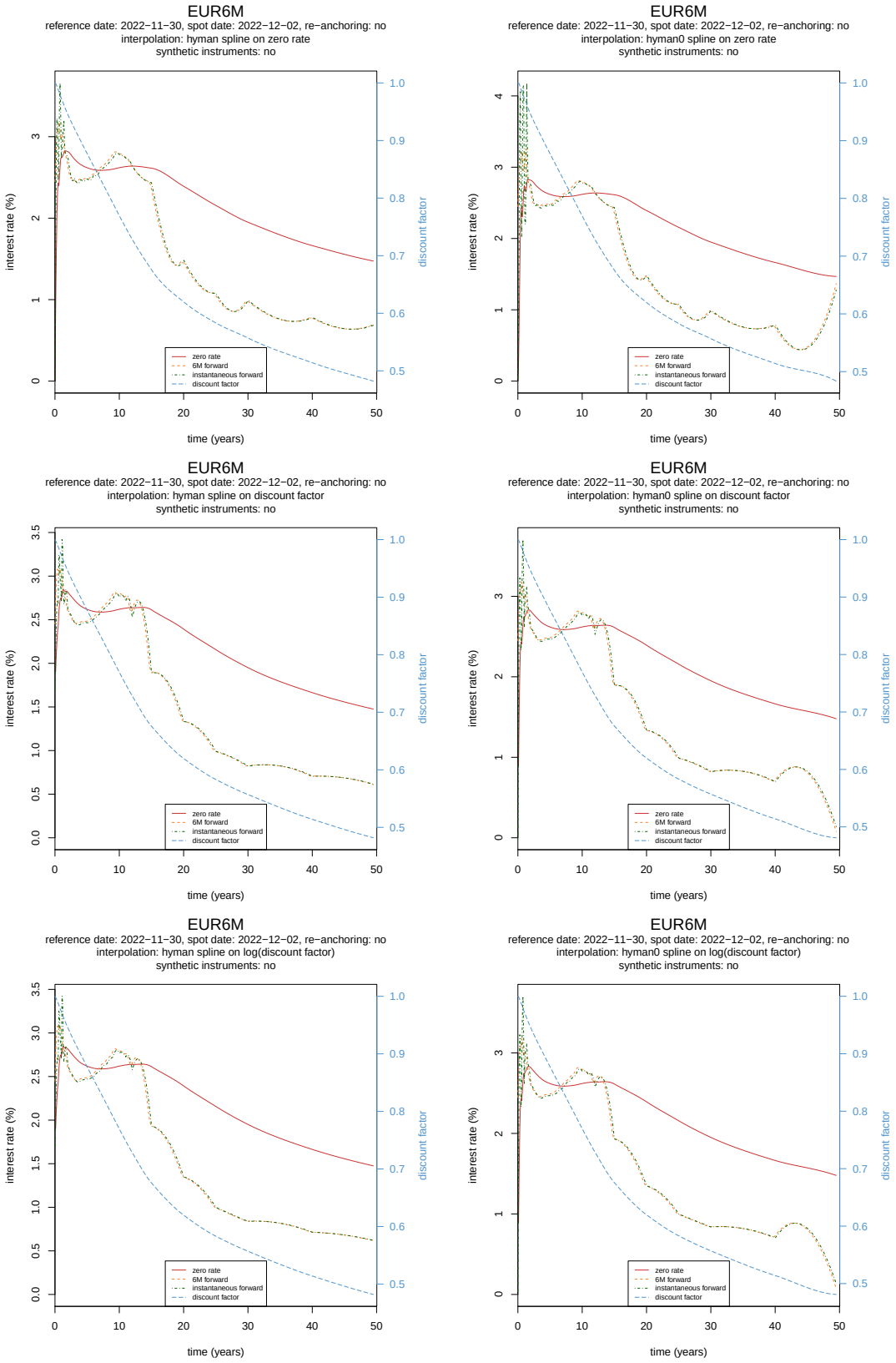

Figure 7.2: EUR6M curve for 2022-30-11 without synthetics. *On the left*: hyman interpolation, *On the right*: hyman0 interpolation. *Top*: interpolation on the zero rates, *centre*: on the discount factors, *bottom*: on the log of the discount factors. On the left axis the zero rates, the discrete forward and the instantaneous forward rates are plotted, on the right axis the discount factors are plotted. Discussion in the text.

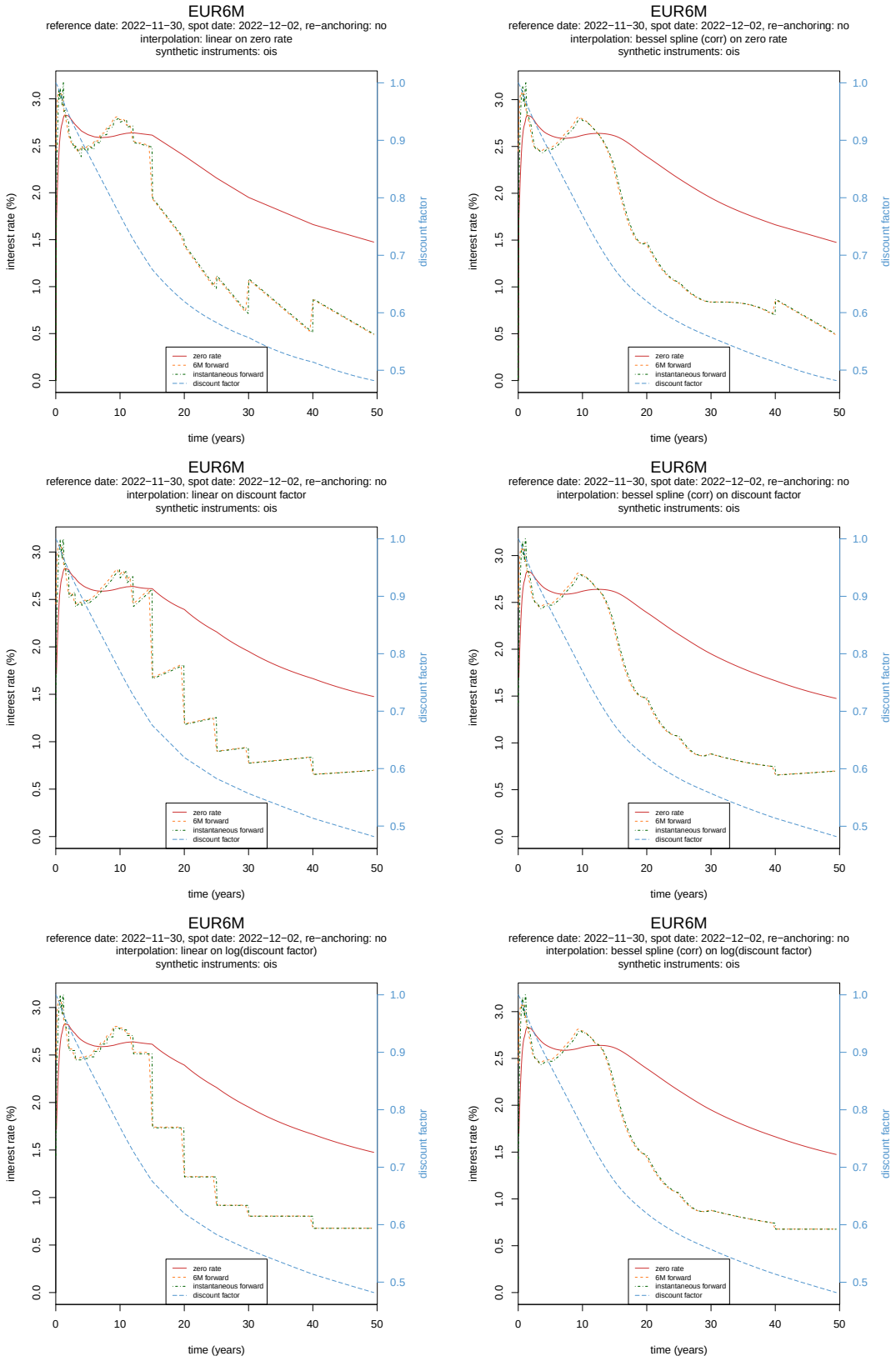

Figure 7.3: EUR6M curve for 2022-30-11 with OIS synthetics. *On the left*: linear interpolation, *On the right*: bessel interpolation with spline correction. *Top*: interpolation on the zero rates, *centre*: on the discount factors, *bottom*: on the log of the discount factors. On the left axis the zero rates, the discrete forward and the instantaneous forward rates are plotted, on the right axis the discount factors are plotted. Discussion in the text.

7.2.3 Leave-one-out test

For the leave-one-out test I collected a lot of results across all combinations of curves, dates and bootstrapping choices. To gain insight I compared this distributions of these errors along multiple axes. In figure 7.4 I present this distribution for the SONIA and EUR3M curves split out across the interpolation data type and the interpolation method. The results for the ESTR, EUR3M and GBP6M curves also show that Bessel interpolation on the log of the discount factors performs well, but they are less outspoken.

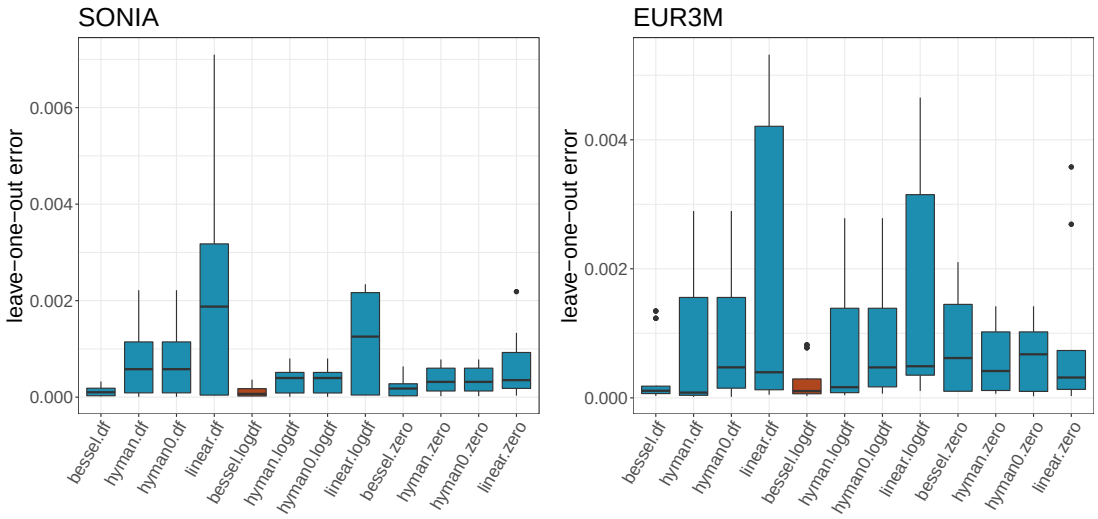

Figure 7.4: Box plot for a combination of choices for the SONIA and EUR3M curves: the interpolation method (Bessel, Hyman, Hyman0, or linear) and the interpolation data type (discount factors, log(discount factors), or zero rates). Highlighted is my preferred combination as discussed in section 7.3: Bessel interpolation on the logarithm of the discount factors.

7.2.4 A note on performance

As mentioned before, performance is very good: on an ordinary laptop one can bootstrap an OIS curve and an IBOR curve together in a couple of seconds for every set of choices.

When comparing the performance across different choices, one clearly sees that linear interpolation is fastest; this is expected as only 1 outer loop of algorithm 1 needs to be performed, whereas the spline interpolations require multiple sweeps.

Between the spline interpolations there is no performance differential: every method requires 5–7 sweeps to converge. Also when comparing the number of times the function `fixing` was called gave very similar results. If we compare the average number of fixings done for every call to the inner loop of algorithm 1, the result is around 6.5 for all spline methods.

This number is a bit higher for linear interpolation: around 9. It is quite conceivable that for the spline methods more fixings are performed in the first sweep and less in the later sweeps, which would explain this. I have not looked into this.

7.3 Final choices

Based on the analyses of the previous section, my preference is to use the following set of choices:

- ◇ Bessel interpolation
- ◇ on the logarithm of the discount factors,
- ◇ applying the spline correction,
- ◇ adding OIS synthetics to the IBOR curves,
- ◇ using ACT/365 as daycount convention,
- ◇ continuously compounded zero rates,
- ◇ anchoring the curve with the spot lag as defined by the currency,
- ◇ not re-anchoring the curve,
- ◇ using the input set from Bloomberg.

In figures 7.5 and 7.6 you find the graphs for the two GBP and the three EURO curves for 2022-11-30 constructed with these settings.

At Triodos they tend toward using linear interpolation on the log of the discount factors. The forwards behave as a step function, which is not ideal, but better than a saw-tooth behaviour. The main advantage over Bessel interpolation is the simplicity: on the one hand this makes it easier to use the curve across multiple systems; on the other hand in a production setting the outcomes are relatively easy to understand and check.

7.4 Contributions

I would like to end by highlighting the contributions made by this thesis. It builds on Ametrano and Bianchetti (2013), but the mathematical foundations are cleaned up and the transition from the single- to the multi-curve framework is more clearly explained.

I added to that work the steps to go from the theory towards an implementation. To that end the work of Hagan and West (2006, 2008) on interpolation schemes for interest rate curves also needed to be included.

I created the software package RateHike (Helsen, 2023) in R. This is not just the cherry on the cake: it was *necessary* to make this implementation in order to understand all the elements that go into bootstrapping and their interaction.

An important conclusion of this work is that using a bootstrapped curve from a data provider or from a closed source software package is dangerous. If you do not understand all the details of how the curve is constructed, it is impossible to extract correct information from it.

I also propose the spline correction and the OIS synthetics; as remarked before it is quite possible these methods have already been introduced elsewhere.

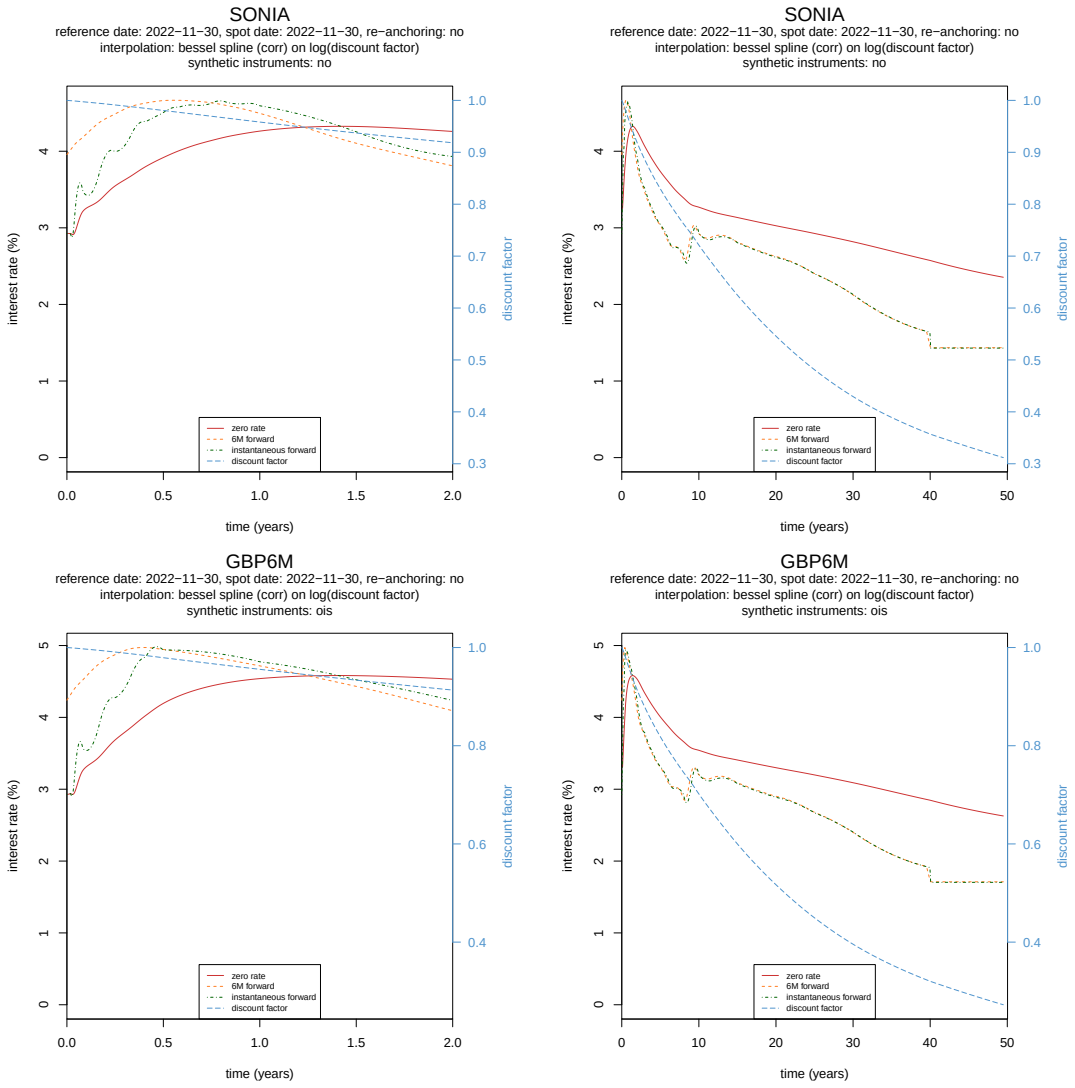

Figure 7.5: GBP curves for my preferred set of choices for 2022-11-30. *Top*: the SONIA curve, *bottom*: the GBP6M curve. *On the left*: the short end of the curve (up to 2 years), *on the right*: the full curve (50 years). On the left axis the zero rates, the discrete forward and the instantaneous forward rates are plotted, on the right axis the discount factors are plotted.

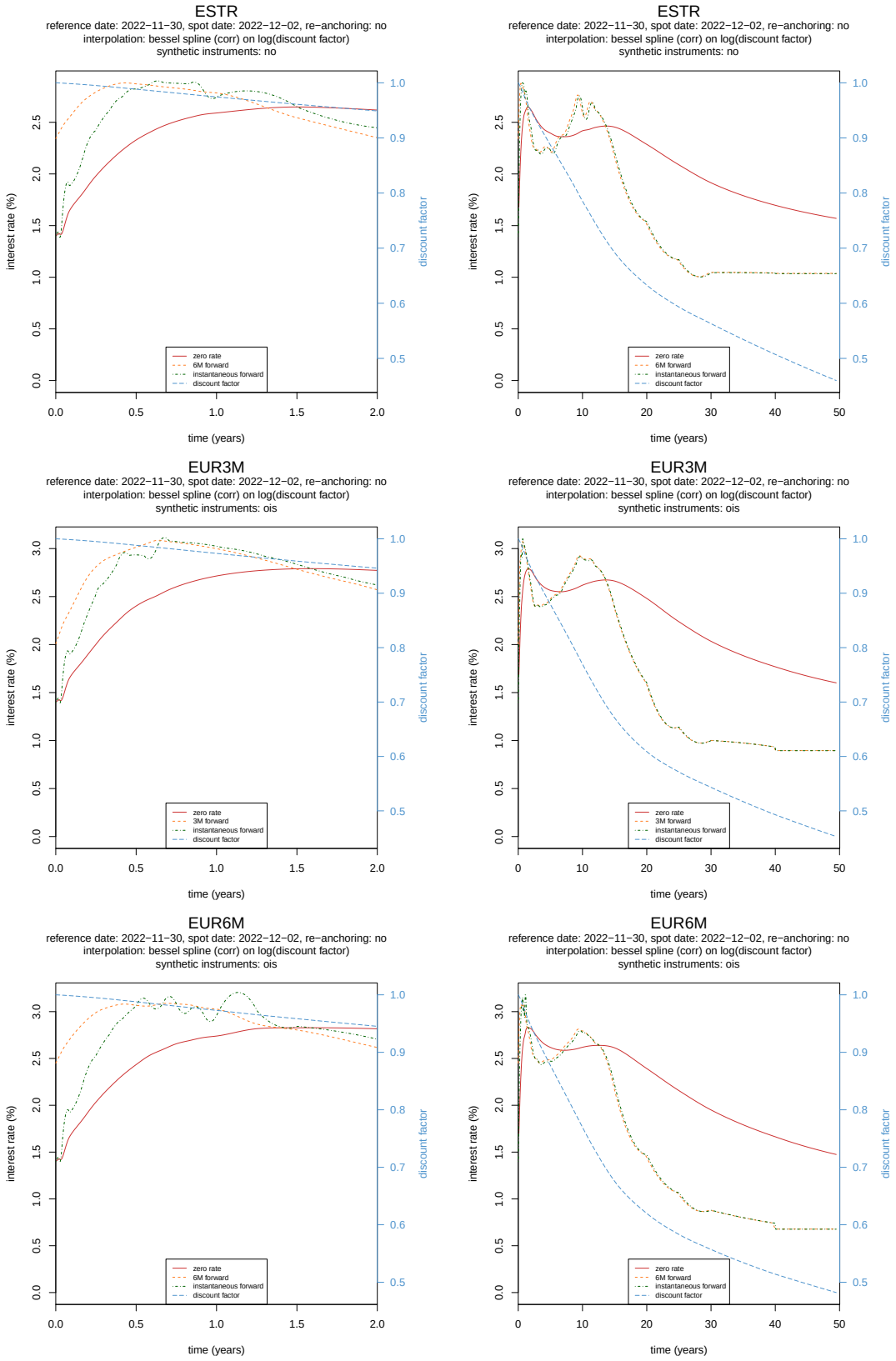

Figure 7.6: EURO curves for my preferred set of choices for 2022-30-11. *Top*: the ESTR curve, *centre*: the EUR3M curve, *bottom*: the EUR6M curve. *On the left*: the short end of the curve (up to 2 years), *on the right*: the full curve (50 years). On the left axis the zero rates, the discrete forward and the instantaneous forward rates are plotted, on the right axis the discount factors are plotted.

Appendix A

Worked out bootstrapping example

This appendix contains a worked example of bootstrapping. I start with a simple set of 3 instruments and bootstrap them on a curve. The first instrument is a deposit, it shows how the different conventions of the curve and the instrument impact the zero rates. The second instrument is an FRA that starts on the maturity date of the deposit. This is used to show how to interpret the curve for forward starting instruments. The third instrument is again an FRA, but the dates do not connect to the other instruments. This is used to show how the interpolation method impacts the bootstrapped curve.

The details on the date calculations are explained in chapter 3, the details of the interpolations can be found in chapter 5. The bootstrapped rates in this appendix are calculated by RateHike, a script "examples-appA.R" is included in the package producing all the results.

As input the instruments from table A.1 are used. The quotes are exaggerated on purpose, using realistic rates would result in differences that are a lot smaller.

instrument	today date	spot date	quote	start date	maturity date
deposit	2022-11-30	2022-12-02	10%	(spot)	2023-06-02
FRA 1	2022-11-30	2022-12-02	12%	2023-06-02	2023-12-04
FRA 2	2022-11-30	2022-12-02	15%	2023-09-04	2024-03-04

Table A.1: Selection of instruments for the EUR6M curve quoted on 2022-11-30.

I bootstrap a curve using these instruments with the following choices:

- ◇ anchored on 2022-12-02
- ◇ day count convention ACT/365
- ◇ linear interpolation
- ◇ on the zero rates
- ◇ endogenous discounting curve
- ◇ continuous compounding

The deposit rate

The bootstrapped zero coupon rate on 2023-06-02 is 9.890 923%. This is the maturity date of the deposit. The difference between the deposit rate of 10% and the bootstrapped rate is explained by the different conventions that need to be applied on the two rates: the deposit rate is simply compounded with a day count convention of ACT/360, the bootstrapped rate is continuously compounded with a day count convention of ACT/365.

Indeed, using the deposit rate we can calculate that an initial deposit of 1 million EURO will grow to

$$€1\,000\,000 \times (1 + 0.1 \times \frac{182}{360}) = €1\,050\,555.56$$

on the maturity date. Doing the same calculations with the bootstrapped rate, but using the different conventions, we find

$$€1\,000\,000 \times \mathrm{e}^{(0.09890923 \times \frac{182}{365})} = €1\,050\,555.55.$$

The small error made is a rounding error. Using more significant digits for the bootstrapped rate gives the exact result.

So essentially the two different interest rates represent the same information, they just use different conventions to encode this information.

The first FRA rate

The bootstrapped zero coupon rate on the maturity date of the first FRA is 10.856 425%. I will again compare this to the FRA rate of 12% by looking at the return of investments. We already know that a 1 million EURO deposit will grow to € 1 050 555.56 on the start date of the FRA. If we apply the FRA rate to this amount, this will grow to

$$€1\,050\,555.56 \times (1 + 0.12 \times \frac{185}{360}) = €1\,115\,339.82$$

on the maturity date. So we calculated the cumulative return of first investing in the deposit and then investing in the FRA. Looking at the return as determined by the curve by investing the same initial amount over the entire period, we find

$$€1\,000\,000 \times \mathrm{e}^{(0.10856425 \times \frac{182+185}{365})} = €1\,115\,339.81.$$

Again, the small error is explained by rounding.

The second FRA rate

The bootstrapped zero coupon rate on the maturity date of the second FRA is 12.081 403%. To verify this we proceed as for the first FRA, but now we still need to calculate how much an initial investment would be worth on the starting date. For this we need to interpolate; as we have set up our curve to be linearly interpolated on the zero rates, we can still do this calculation by hand.

The zero coupon rate on the start date of the second FRA, as calculated by RateHike, is 10.381 503%. To verify this calculation, we use the zero coupon rates of 2023-06-02 and 2023-12-04, which are the pillars of the curve, to confirm

$$9.890923\% \times \frac{91}{185} + 10.856425\% \times \frac{94}{185} = 10.381502\%.$$

Once more, using more decimals would get rid of the rounding error.

So an investment of 1 million EURO will grow to

$$€1\,000\,000 \times \mathrm{e}^{(0.10381503 \times \frac{182+94}{365})} = €1\,081\,664.68$$

on the start date of the second FRA. Applying the FRA quote to this amount, we find the investment will grow to

$$€1\,081\,664.68 \times (1 + 0.15 \times \frac{182}{360}) = €1\,163\,690.92$$

on the maturity date.

To calculate what this initial investment would be worth on that date using the curve, we get

$$€1\,000\,000 \times \mathrm{e}^{(0.12081403 \times \frac{458}{365})} = €1\,163\,690.92.$$

Interpolation effects

In this section I abandon the interpolation setting used up till now. I bootstrap the same instruments across different settings for the interpolation data type and method. Table A.2 gives the zeros rates for all choices on a set of dates of interest.

method	data type	2023-06-02	2023-09-04	2023-12-04	2024-03-04
linear	zero	9.890 923	10.381 503	10.856 425	12.081 403
linear	df	9.890 923	10.484 063	10.856 425	12.143 208
linear	logdf	9.890 923	10.543 252	10.856 425	12.178 876
bessel	zero	9.890 923	10.774 010	10.856 425	12.317 935
bessel	df	9.890 923	10.206 812	10.856 425	11.976 131
bessel	logdf	9.890 923	10.197 888	10.856 425	11.970 753
hyman	zero	9.890 923	10.472 982	10.856 425	12.136 530
hyman	df	9.890 923	10.310 694	10.856 425	12.038 732
hyman	logdf	9.890 923	10.320 836	10.856 425	12.044 844
hyman0	zero	9.890 923	10.472 982	10.856 425	12.136 530
hyman0	df	9.890 923	10.310 694	10.856 425	12.038 732
hyman0	logdf	9.890 923	10.320 836	10.856 425	12.044 844

Table A.2: Zero coupon rates (in %) for the instrument list from in this appendix for some dates of interest when bootstrapped with different choices for the interpolation data type and interpolation method. See the discussion in the text.

Note that for the first and the third date there is no impact. These are pillars of the curves. Even if the data is stored differently for the different interpolation data types, the conversion is accurate up to machine precision.

The second date is the start date of FRA 2. The zero rates on that date are different across the different choices: the only inputs to calculate this result are the pillars for the first and the third dates, so this shows the effect of the choice of interpolation scheme.

The last date is the maturity date of FRA 2. This is again a pillar of the curve. To determine this pillar the market quote of FRA 2 is used, together with the curve information for the start date. So the differences observed for this pillar are a knock-on effect of the interpolation differences on the start date of the instrument. For completeness I'd like to mention that the fixing of the second FRA is 15% for all curves, so the round-trip test is successful.

Note that interpolation methods "hyman" and "hyman0" give the same results. This is because the difference between these two only appears in the end intervals. When bootstrapping the last instrument onto the curve in the first iteration of the outer loop of algorithm 1, there will be a difference between the two methods, as at that time the interpolated value for the start date is located within the end interval, but when iterating the outer loop until convergence, this difference disappears.

Appendix B

Futures: convexity adjustment

As mentioned in subsection 4.3.6, a futures rate cannot be immediately translated to an FRA rate. The differences in the contractual payments for the two instruments make that a convexity adjustment needs to be applied. This convexity adjustment depends on the evolution of the curve from now to the start date of both instruments, and more specifically on the divergence of the points on the curve representing the start date and the maturity date of both instruments.

This quantity cannot be calculated as such, it needs to be modelled, and different choices of model exist. In a sense, in the multi-curve framework, the most natural would be to use a multi-curve Libor Market Model to estimate this difference, as done in Mercurio (2010, appendix B).

However, I use the same conventions as Bloomberg: on the Bloomberg screen that shows the futures values, you can see the convexity adjustment that is applied. They follow Kirikos and Novak (1997), which is based on a Hull-White model.

Let T be the year fraction from the spot date to the start date of the instrument and let τ be the year fraction from the start date to the maturity date of the instrument, both in the day count convention of the instrument. Let a be the mean reversion speed and σ be the volatility of the Hull-White model. Define

$$\Lambda = \frac{\sigma^2}{2a^3}(1 - \mathrm{e}^{-2aT})(1 - \mathrm{e}^{-a\tau})^2,$$

$$\Phi = \frac{\sigma^2}{2a^3}(1 - \mathrm{e}^{-aT})^2(1 - \mathrm{e}^{-a\tau}),$$

$$Z = \Lambda + \Phi.$$

To calculate the equivalent FRA rate $\tilde{r}_{\text{fut}}$ from a quoted futures price F_q, the equations to apply are

$$F_a = \mathrm{e}^{-Z} F_q + 100(1 - \mathrm{e}^{-Z})(1 + \frac{1}{\tau}),$$

$$\tilde{r}_{\text{fut}} = 1 - \frac{F_a}{100},$$

where F_a stands for the adjusted futures price.

To calculate the quoted futures price F_q from the equivalent FRA rate $\tilde{r}_{\text{fut}}$, the equations are

$$F_a = 100(1 - \tilde{r}_{\text{fut}}),$$

$$F_q = \mathrm{e}^Z F_a - 100(\mathrm{e}^Z - 1)(1 + \frac{1}{\tau}).$$

I repeat the remark from subsection 4.3.6: while it is useful to understand the need for this convexity adjustment and to know how it is applied, it is not strictly necessary for a bootstrapping implementation since the equivalent FRA rates are available from the data providers.

Bibliography

Adams, K. (2001). Smooth interpolation of zero curves. *Algo Research Quarterly*, *4*, 11–20.

Ametrano, F. M., & Bianchetti, M. (2009). Smooth yield curves bootstrapping for forward libor rate estimation and pricing interest rate derivatives. In D. Brigo (Ed.), *Modelling interest rates: Latest advances for derivatives pricing* (pp. 3–42). Risk Books.

Ametrano, F. M., & Bianchetti, M. (2013). Everything you always wanted to know about multiple interest rate curve bootstrapping but were afraid to ask. *SSRN Electronic Journal*. https://ssrn.com/abstract=2219548

Bingham, N. H., & Kiesel, R. (2010). *Risk-neutral valuation: Pricing and hedging of financial derivatives* (2nd edition). Springer.

Brigo, D., & Mercurio, F. (2006). *Interest rate models - theory and practice: With smile, inflation and credit* (2nd edition). Springer Berlin Heidelberg.

de Boor, C. (2001). *A practical guide to splines* (Revised). Springer.

El Menouni, Z. (2015). *Pricing interest rate derivatives in the multi-curve framework with a stochastic basis* (Master's thesis). KTH, Mathematical Statistics. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-163274

EMMI. (2022a). Efterm. https://www.emmi-benchmarks.eu/benchmarks/efterm

EMMI. (2022b). The future is euribor. https://www.emmi-benchmarks.eu/communication/news/the-future-is-EURIBOR

European Central Bank. (2000). Long-term calendar for TARGET closing days. https://www.ecb.europa.eu/press/pr/date/2000/html/pr001214_4.en.html

FCA. (2021). FCA announcement on future cessation and loss of representativeness of the LIBOR benchmarks. https://www.fca.org.uk/publication/documents/future-cessation-loss-representativeness-libor-benchmarks.pdf

Fujii, M., Shimada, Y., & Takahashi, A. (2009). A note on construction of multiple swap curves with and without collateral. *CARF Working Paper Series*, (CARF-F-154). https://ssrn.com/abstract=1440633

Fujii, M., Shimada, Y., & Takahashi, A. (2011). A market model of interest rates with dynamic basis spreads in the presence of collateral and multiple currencies. *Wilmott Magazine*, (54), 61–73.

Grbac, Z., & Runggaldier, W. J. (2015). *Interest rate modeling: Post-crisis challenges and approaches*. Springer International Publishing.

Hagan, P. S., & West, G. (2006). Interpolation methods for curve construction. *Applied mathematical finance*, *13*(2), 89–129.

Hagan, P. S., & West, G. (2008). Methods for constructing a yield curve. *Wilmott Magazine*, 70–81.

Helsen, S. (2023). RateHike. https://github.com/spuddie/RateHike

Henrard, M. P. A. (2007). The irony in the derivatives discounting. *Wilmott Magazine*, (30), 92–98.

Henrard, M. P. A. (2010). The irony in derivatives discounting part ii: The crisis. *Wilmott Magazine*, (2), 301–316.

Hull, J. C. (2012). *Options, futures, and other derivatives* (8th edition). Pearson Education.

Hull, J. C., & White, A. (2013). LIBOR versus OIS: The Derivatives Discounting Dilemma. *Journal of Investment Management*, *11*.

Hyman, J. M. (1983). Accurate monotonicity preserving cubic interpolation. *SIAM journal on scientific and statistical computing*, *4*(4), 645–654.

Iebesh, A. (2020). *Interpolation of yield curves* (Master's thesis). Mälardalen University. https://www.researchgate.net/publication/342437744_Interpolation_of_Yield_curves

Kirikos, G., & Novak, D. (1997). Convexity conundrums. *Risk Magazine*, *3*(10), 60–61. https://powerfinance.com/convexity

Martin, M. R. W. (2018). An overview of post-crisis term structure models. In M. Mili, R. Samaniego Medina, & F. di Pietro (Eds.), *New methods in fixed income modeling: Fixed income modeling* (pp. 85–97). Springer International Publishing. https://doi.org/10.1007/978-3-319-95285-7_5

Mercurio, F. (2009). Interest rates and the credit crunch: New formulas and market models. *Bloomberg Portfolio Research Paper*, (2010-01-FRONTIERS). https://ssrn.com/abstract=1332205

Mercurio, F. (2010). Libor market models with stochastic basis. *SSRN Electronic Journal*. https://ssrn.com/abstract=1563685

Nelson, C. R., & Siegel, A. F. (1987). Parsimonious modeling of yield curves. *The Journal of business (Chicago, Ill.)*, *60*(4), 473–489.

Nymand-Andersen, P. (2018). Yield curve modelling and a conceptual framework for estimating yield curves. *European Central Bank Statistics Paper Series*, *27*.

OpenGamma. (2013). OpenGamma interest rate instruments and market conventions guide. https://quant.opengamma.io/Interest-Rate-Instruments-and-Market-Conventions.pdf

OpenGamma. (2022). Strata. https://github.com/OpenGamma

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd edition). Cambridge University Press.

Pyper, D. (2015). Bank and public holidays. *House of Commons Briefing Papers*, (SN06170). https://researchbriefings.files.parliament.uk/documents/SN06170/SN06170.pdf

R Core Team. (2022). R: A language and environment for statistical computing. https://www.R-project.org

Ron, U. (2000). A practical guide to swap curve construction. *Bank of Canada, Working Papers*. https://www.bankofcanada.ca/2000/08/working-paper-2000-17

Spoor, J. (2013). *Double effect : Multiple interest rate curve bootstrapping* (Master's thesis). Universiteit Twente. http://essay.utwente.nl/63747

Svensson, L. E. (1994). Estimating and interpreting forward interest rates: Sweden 1992 - 1994. *NBER Working Paper Series*, (4871).

The QuantLib contributors. (2022). Quantlib: A free/open-source library for quantitative finance. https://github.com/lballabio/quantlib

Wickham, H. (2011). Testthat: Get started with testing. *The R Journal*, *3*, 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf

Wickham, H. (2019). *Advanced r* (2nd edition). Chapman & Hall. https://adv-r.hadley.nz

Wickham, H., Danenberg, P., Csárdi, G., & Eugster, M. (2022). Roxygen2: In-line documentation for r [R package version 7.2.3]. https://CRAN.R-project.org/package=roxygen2

Faculty of Economics and Business
Naamsestraat 69 bus 3500
3000 Leuven, BELGIË
tel. +32 16 32 66 12
fax +32 16 32 67 91
www.feb.kuleuven.be