The Clean Thesis Style

Ricardo Langner

August 26, 2015
Version: My First Draft

Clean Thesis Style University

CleanThesis

Department of Clean Thesis Style
Institut for Clean Thesis Dev
Clean Thesis Group (CTG)

Documentation

The Clean Thesis Style

Ricardo Langner

1. Reviewer Jane Doe
Department of Clean Thesis Style
Clean Thesis Style University

2. Reviewer John Doe
Department of Clean Thesis Style
Clean Thesis Style University

Supervisors Jane Doe and John Smith

August 26, 2015

Ricardo Langner
The Clean Thesis Style
Documentation, August 26, 2015
Reviewers: Jane Doe and John Doe
Supervisors: Jane Doe and John Smith

Clean Thesis Style University
Clean Thesis Group (CTG)
Institut for Clean Thesis Dev
Department of Clean Thesis Style
Street address
Postal Code and City

Contents

1 Introduction

1.1 Motivation and Problem Statement

1.2 Results

1.2.1 Some References

[**WEB:GNU:GPL:2010**; **WEB:Miede:2011**]

1.3 Thesis Structure

Chapter 2

We derived a class of discount curve construction and extrapolation methods, based on a class of interest rate models called *exogenous short-rate models*. That means: constructing a static Yield Curve at a given date, by using some specific financial instruments with different maturities quoted at this date. Then, defining what are the discount rates beyond the longest maturity observed for these financial instruments. The extrapolated part of the curve is typically necessary for the pricing of long-term insurance liabilities. In the framework that we propose, Yield Curve forecasts can be obtained by using a **functional principal components analysis** on the model parameters.

Chapter 3

We were interested in obtaining forecasts for multiple time series, by taking into account the potential nonlinear relationships between their observations. For this purpose, we used a specific type of regression model on an augmented dataset of lagged time series. Our model is inspired by dynamic regression models, with the response variable's lags included as predictors, and is known as **random vector functional link (RVFL) neural networks**. The RVFL neural networks have been successfully applied in the past, to solving regression and classification problems. The novelty of our approach is to apply an RVFL model to multivariate time series, under two separate regularization constraints on the regression parameters.

Chapter A

The goal of ensemble learning is to combine two or more statistical/machine learning models - the base learners - into one, in order to obtain an ensemble model. The ensemble model is expected to have an improved out-of-sample error over the base models. We apply two popular ensemble learning methods to multiple time series forecasting: **bootstrap aggregating**, known as bagging, **boosting**, and **stacked generalization**, known as stacking. The base learners that we use, are the RVFL introduced in the previous paragraph.

Chapter B

Kernel regularized least squares (KRLS) learning methods are applied to Yield Curve forecasting. Two types of formulations of the forecasting problem are tested. One relying on a popular framework called Dynamic Nelson-Siegel, and another one, in which we apply the KRLS directly to the explain the spot rates (response variable) as a function of the observation dates and time to maturities (covariates).

Chapter C

We present a **bayesian quasi-randomized vector functional link neural network model** (BQRVFL), with one hidden layer. It's a penalized regression model on an augmented data set, in which we assume that a prior multivariate gaussian distribution governs the regression parameters. The BQRVFL model is presented, along with the associated formulas for confidence interval around its predictions. It is then applied as a workhorse for **bayesian optimization** of machine learning cross-validation functions. The machine learning cross-validation functions that we consider are those associated to the selection of hyperparameters of RVFL and KRLS models.

2 A swap curve for insurance risk management, based on no arbitrage short-rate models

2.1 Context

The new Solvency II directive defines the calculation of European insurers' technical provisions as the sum of two components, the Best Estimate Liabilities (BEL) and the Risk Margin (RM). The Best Estimate Liabilities (BEL) are defined as the average discounted value of the insurer's future cash-flows, weighted by their probability of occurrence. The Risk Margin is a supplemental amount required for covering the non-hedgeable risks, by involving a capital lockup.

In order to discount the cash-flows relevant in the calculation of the BEL and Risk Margin, an *appropriate* term structure of discount factors is needed. From the no arbitrage pricing theory developed by [HP81], and widely used in insurance *market consistent* pricing of liabilities, the zero rates related to the stochastic discount factors have to be *risk-free*. That is, free from any counterparty credit risk.

There is no easy answer to the question of defining such a *risk-free* rate for insurance liabilities. It could be related the insurer's own assets return, where the liabilities are *perfectly* backed by the assets. But in a *market consistent* approach as required in Solvency II, since not every liability is perfectly backed by the assets, a more fundamental *risk-free* rate also needs to be derived. Deriving such a *common risk-free* rate from market-quoted instruments is also aimed at increasing transparency and comparability of balance sheets across European countries.

For years, in banking, the construction of a term structure of *risk-free* discount factors was based on the assumption that banks are not subject to counterparty credit risk when lending to each other, and liquidity was not an issue. In this context, interbank rates (loosely called LIBOR hereafter) were seen as the best proxies for *risk-free* rates.

From the 2007-2008 financial crisis onwards, the spreads between swaps rates with different tenors started to widen, partly due to the increased reticence of banks to

lend to each other. Today, LIBOR is no longer considered as a proxy for *risk-free* rates, and market operators have increasingly started to use Overnight Interest Swaps (OIS) discounting (see [HW12] for example).

Comparatively in the European Insurance market, throughout the quantitative impact studies (the QIS) leading to Solvency II, the questions of *risk-free* term structure construction for valuation have been tackled for years by the CEIOPS and later by the EIOPA (see [CC10] for example). The difficulty in defining a fundamental *risk-free* rate for the insurance market, mainly arises from the fact that a pure market *risk-free* rate could introduce a lot of unwanted market volatility into the insurer's balance sheet. Hence, this discount curve has been adjusted with different spreads through the QIS, and until its most recent specification, making it somewhat, less *consistent* with the market.

As of June 2015 (see [EIO15]), the term structure of discount factors for insurers' liability cash-flows is indeed derived from LIBOR EUR swap (IRS hereafter) rates, as the market for vanilla swaps is considered as 'Deep, Liquid, and Transparent' (the DLT assumption). A credit risk adjustment (CRA) is prescribed by the directive, consisting in a parallel shift applied to LIBOR swap rates. The parallel shift shall not be lower than -35bps or greater than -10 bps. Furthermore, a matching adjustment and a volatility adjustment are other optional parallel shifts which could be applied to the constructed curve.

The volatility adjustment is designed to be used in case of a crisis, causing the widening of sovereign or corporate bonds spreads. On the other hand, the matching adjustment is used in cases where the liabilities are predictable, that is, almost *perfectly* backed. In this paper, we focus on discount curve construction. The matching premium and the volatility adjustment are not further discussed.

Beyond the data and curve adjustments concerns, and considering curve construction methods, [AB13] distinguish between two types of methods: *best fit* methods, and *exact fit* methods. Best fit methods, such as [NS87] and [Sve94] are widely used by central banks. Exact fit methods such as cubic splines methods on the other hand, generally have at least as much parameters as input market products, and provide an exact fit to market data.

While the latter type of methods would be adapted for no arbitrage pricing and trading, the former type are useful for forecasting the yield curve in real world probability (see [DL06] for example). They fit the curve parsimoniously with a few parameters; in an attempt to mimic the factors explaining the variance of the yield curve changes (see [Lit+91] for details). There is another class of models,

which combine the idea of using a factors structure, which is the absence of dynamic arbitrages in the curve diffusion, see [Chr+11] for example.

The extrapolation of the constructed curve is also an important subject matter for insurers and pension funds. Indeed, some of their liability cash-flows may have very long maturities, spanning beyond the longest liquid maturities available for market-quoted instruments. The question is, how would spot rates for such long maturities be determined?

As of 2016 in Solvency II, the construction and extrapolation of the swap curve is made by using the Smith-Wilson method described in
[SW01] and in the technical specifications [EIO15]. The Smith-Wilson method constructs the swap curve by exactly fitting the market IRS rates adjusted from a CRA. After a chosen maturity - the last liquid point (LLP), equal to 20 years -, the forward rate is forced by regulatory rules, to converge at an exogenously specified speed to a fixed long term level called the Ultimate Forward Rate (UFR). The UFR is derived as the sum of expected Euro inflation and expected real rates. As of 2016, it is equal to 4.2%.

For discount curve construction and extrapolation, we propose a method which relies on closed-form formulas for discount factors available in exogenous (or no arbitrage) short-rate model. It could be both an *exact fit* and *best fit* method, depending on the data at hand, and on how the curve is calibrated to these data. In this framework, the time-varying function ensuring an exact fit to market implied discount factors in exogenous short-rate models is considered to be a piecewise constant function, whose steps become model's parameters. The interpolation of the curve at dates comprised between quoted maturities directly comes from the properties of the model. Pseudo-discount curves can also be constructed in a dual curve environment, by using our method, along with the techniques described for example in [Whi12] and [AB13].

The static discount curve calibrated to market data can then be extrapolated to longer, unobserved maturities, with the forward rates converging to an *ultimate forward rate*. Extrapolation is done by using the same model that the one used for interpolation. On this particular point, our model is hence closer to the [SW01] model than to models which use different methods for interpolation and extrapolation (such as cubic splines for interpolation, and a modified version of [NS87] for extrapolation). We describe ways to either derive an UFR from the data, or to constraint the model to converge to a given UFR.

When it comes to forecasting and/or simulation, if one is interested in no arbitrage pricing, then she can use simulations under a risk neutral probability of the corresponding, consistent (in the sense of [BC99]) exogenous short-rate model. Otherwise, forecasts of the yield curve under the historical probability can be obtained by making use of a functional principal components analysis on the model parameters. Functional principal components analysis is described in [RD91] and [RS05]. It has been applied to forecasting mortality rates by [HU07], and in finance, it has been applied for example in [**Benko2007Functional**].

The advantage of the model presented in this paper, is that, it reconciles in some sense models like [DL06] or [SW01]. Its direct link with an exogenous short-rate model (consistency in the sense of [BC99]) and the possibility of achieving an *exact fit* to swap data means that it could be used as an input for pricing in a risk neutral probability. In addition, although it could be less interpretable than models like [DL06] (in terms of level, slope, and curvature), it could also be used for forecasting the yield curve parsimoniously in historical probability, as demonstrated in sections 2.4.3 and 2.4.4.

In the next sections, we describe the model proposed for discount curve construction and extrapolation, and explain how it could be calibrated to market data. Then, we explain how to obtain forecast of the discount curve, by using the model's parameters. To finish, some numerical examples based on [HW06], [And07], [AP10], [AB13] are presented.

2.2 Curve construction and extrapolation

The class of models proposed for discount curve construction and extrapolation relies on short-rate models with a time-varying mean-reversion parameter: exogenous short-rate models. In this section, we provide details on how they are derived.

In the sequel, let Y denote a Lévy process and W a standard brownian motion. We assume that all introduced processes are defined with respect to a stochastic basis $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{Q})$. For every considered Lévy process Y, its cumulant function is denoted by κ, i.e., $\kappa(\theta) = \log \mathbb{E}\left[e^{\theta Y_1}\right]$. As a matter of example, some cumulant functions are given in Table 2.1 for the Brownian motion and for two class of Lévy subordinators parametrized by a single variable λ which inversely controls the jump size of the Lévy process. We refer the reader to [**Cont2003**] for more details on Lévy processes. We assume that, under a risk-neutral probability measure $\mathbb{Q}$, the short-term interest

	Lévy measure	Cumulant
Brownian motion	$\rho(dx) = 0$	$\kappa(\theta) = \frac{\theta^2}{2}$
Gamma process	$\rho(dx) = \frac{e^{-\lambda x}}{x} 1_{x>0} dx$	$\kappa(\theta) = -\log\left(1 - \frac{\theta}{\lambda}\right)$
Inverse Gaussian process	$\rho(dx) = \frac{1}{\sqrt{2\pi x^3}} \exp\left(-\frac{1}{2}\lambda^2 x\right) 1_{x>0} dx$	$\kappa(\theta) = \lambda - \sqrt{\lambda^2 - 2\theta}$

Tab. 2.1.: Examples of Lévy measures and cumulants

rate is either governed by an extended Lévy-driven Ornstein-Uhlenbeck process (Lévy-driven OU)

$$dX_t = a(b(t) - X_t)dt + \sigma dY_{ct}, \tag{2.1}$$

or an extended CIR process

$$dX_t = a(b(t) - X_t)dt + \sigma\sqrt{X_t}dW_t, \tag{2.2}$$

where the long-term mean parameter b is assumed to be a deterministic function of time, a is a positive parameter which controls the speed of mean-reversion and σ is a positive volatility parameter. Concerning the Lévy-driven OU specification 2.1, we use an additional positive parameter c which appears as an increasing change of time $t \to ct$. This parameter can also be interpreted as a volatility parameter but, contrarily to σ, it controls jump frequency (an increase of c makes the underlying Lévy process jumps more frequently). Let X_0 be the value at time t_0 of the process X. The use of Lévy processes as a driver of short rate or default intensity dynamics stems from the fact that processes driven by some Lévy processes could provide better fit on time series of bond returns than when driven by a Brownian motion. For more details on term structure modeling with Lévy processes, the reader is referred for instance to [**Cariboni2004**; **Crepey2012**; **Eberlein1999**; **Kluge2005**].

Remark: Specification 2.1 corresponds to a Lévy Hull-White extended Vasicek model. However, in the seminal Hull-White approach (see [HW90]), the initial term-structure is given as a model input and the function b is defined in such a way that the input term-structure is reproduced by the model. In our approach, contrarily to the Hull and White framework, the deterministic function b is *directly* calibrated on market quotes of interest-rate products.

Under the previous short-rate models, arbitrage-free prices of zero-coupon bonds can be expressed analytically.

2.2.1 Curve explicit analytical expressions

We rely on a standard pricing framework where, in absence of arbitrage opportunity, the value at time t_0 of a default-free zero-coupon bond with maturity time t is given by

$$P(t_0,t) = \mathbb{E}_{\mathbb{Q}}\left[\exp\left(-\int_{t_0}^{t} X_u du\right) \mid \mathcal{F}_{t_0}\right], \tag{2.3}$$

where $\mathcal{F}$ is the natural filtration of the short-rate process X.

When the mean-reverting level b is a deterministic function of time, the following proposition, which is a classical result in the theory of affine term-structure models, gives an analytical expression for $P(t_0,t)$ in the class of Lévy-driven OU models.

Proposition: In the Lévy-driven OU model (2.1), the value at time t_0 of a default-free zero-coupon bond with maturity t is given by

$$P(t_0,t) = \exp\left(-\phi(t-t_0)X_0 - a\int_{t_0}^{t} b(u)\phi(t-u)du - c\psi(t-t_0)\right) \tag{2.4}$$

where the functions ϕ and ψ are defined by

$$\phi(s) := \frac{1}{a}\left(1-e^{-as}\right), \tag{2.5}$$

$$\psi(s) := -\int_0^s \kappa\left(-\sigma\phi(s-\theta)\right)d\theta. \tag{2.6}$$

Proof. Using Itô's lemma, the Lévy-driven OU process is such that, for any $t > t_0$

$$X_t = e^{-a(t-t_0)}X_0 + a\int_{t_0}^{t} b(\theta)e^{-a(t-\theta)}d\theta + \sigma\int_{t_0}^{t} e^{-a(t-\theta)}dY_{c\theta}. \tag{2.7}$$

and, using (2.1) and (2.7), the integral $\int_{t_0}^{t} X_u du$ can be reformulated as

$$\int_{t_0}^{t} X_u du = \phi(t-t_0)X_0 + a\int_{t_0}^{t} b(u)\phi(t-u)du + \sigma\int_{t_0}^{t}\phi(t-u)dY_{cu}. \tag{2.8}$$

Expression (2.4) is obtained from (2.3) and (2.8) and by using Lemma 3.1 in [**Eberlein1999**]. □

Remark: Note that the function ϕ does not depend on the Lévy process specification. Moreover, for most Lévy processes, the integral of the cumulant transform in (2.6) has no simple closed-form solution but can be easily computed numerically. The reader is referred to [**Hainaut2007**] for examples of Lévy processes for which the function ψ defined by (2.6) admits a closed-form expression.

Similar analytical expressions are available under model specification 2.2 where the underlying short-rate process follows an extended CIR process with deterministic long-term mean parameter b.**proposition** In the extended CIR model (2.2), the value at time t_0 of a zero-coupon bond with maturity t is given by

$$P(t_0,t) = \exp\left(-X_0\varphi(t-t_0) - a\int_{t_0}^{t}\varphi(t-u)b(u)du\right) \tag{2.9}$$

where φ is given by

$$\varphi(s) := \frac{2(1-e^{-hs})}{h+a+(h-a)e^{-hs}} \tag{2.10}$$

and $h := \sqrt{a^2+2\sigma^2}$.

Proof. For any maturity date t, thanks to the Feynman-Kac formula, the function $\tilde{P}$ defined for any u such that $t_0 \leq u \leq t$ by

$$\tilde{P}(u,x) := \mathbb{E}_{\mathbb{Q}}\left[\exp\left(-\int_u^t X_u du\right) \mid X_u = x\right]$$

is solution of the following PDE

$$\frac{\partial\tilde{P}(u,x)}{\partial u} + a\left(b(u)-x\right)\frac{\partial\tilde{P}(u,t)}{\partial x} + \frac{1}{2}\sigma^2 x\frac{\partial^2\tilde{P}(u,t)}{\partial x^2} - \tilde{P}(u,t)x = 0, \tag{2.11}$$

with the final condition $\tilde{P}(t,x) = 1$, for all x. It is straightforward to check that the function $\tilde{P}$ defined by

$$\tilde{P}(u,x) = \exp\left(-x\varphi(t-u) - a\int_u^t\varphi(t-s)b(s)ds\right)$$

with φ given by (2.10) is solution of PDE (2.11).

□

Depending on the chosen term-structure model, Proposition 2.2.1 or Proposition 2.2.1 can be used to compute present values of market instruments involved in the curve construction.

2.2.2 Piecewise-constant long-term mean parameter $b(t)$

Typically, in order to obtain an *exact* fit in the Hull-White extended Vasicek model (that is, a model from class 2.1 with $c = 1$ and a Brownian motion as Lévy driver), we have to choose:

$$b(t) = \frac{1}{a}\frac{df^M}{dt}(0,t) - f^M(0,t) + \frac{1}{2}\frac{\sigma^2}{a^3}\left(1-e^{-2at}\right) \tag{2.12}$$

where $t \mapsto f^M(0,t)$ are the market implied instantaneous forward rates. In our framework, the market-implied discount curve $P^M(0,t)$ at date $t_0 = 0$ is constructed by considering a piecewise constant function $t \mapsto b(t)$, whose steps are derived from vanilla (IRS) or overnight swaps (OIS) cash-flows. We let $T_1, \ldots, T_n$, be the maturities of market quoted IRS, with Credit Risk Adjustment (CRA), or OIS. We assume that the function $t \mapsto b(t)$ is piecewise-constant, with:

$$b(t) = b_i, \quad \text{for } T_{i-1} \leq t < T_i, \quad i = 1, \ldots, n \tag{2.13}$$

$$b(t) = b_{n+1}, \quad \text{for } t \geq T_n \tag{2.14}$$

and $T_0 = t_0 = 0$.

Under this specification of the long-term mean parameter, closed-form formulas can be obtained for the discount factors. Under model class 2.1 and given that $t_0 = 0$, the integral term in equation (2.4) becomes

$$I_{n+1}(t) = \sum_{k=1}^{n} b_k \left(\xi(t - T_{k-1} \wedge t) - \xi(t - T_k \wedge t)\right) \tag{2.15}$$

for any $t \leq T_n$ and

$$I_{n+1}(t) = \sum_{k=1}^{n} b_k \left(\xi(t - T_{k-1}) - \xi(t - T_k)\right) + b_{n+1}\xi(t - T_n) \tag{2.16}$$

for $t > T_n$ where ξ is defined as

$$\xi(s) := s - \phi(s), \; s \geq 0. \tag{2.17}$$

Under model class 2.2 and given that $t_0 = 0$, the integral term in equation (2.9) becomes

$$I_{n+1}(t) = \sum_{k=1}^{n} b_k \left(\eta(t - T_{k-1} \wedge t) - \eta(t - T_k \wedge t)\right) \tag{2.18}$$

for any $t \leq T_n$ and

$$I_{n+1}(t) = \sum_{k=1}^{n} b_k \left(\eta(t - T_{k-1}) - \eta(t - T_k)\right) + b_{n+1}\eta(t - T_n) \tag{2.19}$$

for $t > T_n$ where η is defined as

$$\eta(s) := 2\left[\frac{s}{h+a} + \frac{1}{\sigma^2}\log\frac{h + a + (h-a)e^{-hs}}{2h}\right]. \tag{2.20}$$

The previous result can also be found in [**Bielecki2014**] under a more general form. [**Schlogl2000**] also consider an extended CIR model with piecewise-constant

parameter in order to construct initial yield-curves but prices of zero-coupon bonds are given under a recursive form.

Recall that our aim is to construct a discount curve by fitting the mean-reversion function b on quoted swaps observed for different standard maturities. The interpolation of the curve at intermediary dates between quoted swaps maturities directly comes from the properties of the model. Contrarily to the Hull-White approach where an exogenous term-structure is given as a model input, in our approach, the fitted risk-neutral short-rate model is, by construction, consistent in the sense of [BC99].

In the next section, we explain how model parameters can be calibrated in different situations, for the construction of OIS and IRS (with CRA) discount curves. We focus our presentation on the particular extended Vasicek short-rate model

$$dX_t = a(b(t) - X_t)dt + \sigma dW_t. \tag{2.21}$$

This model belongs to class 2.1 with $c = 1$ and with a Brownian motion as Lévy driver. However, the presented calibration method can easily be adapted to other mean-reverting models of class 2.1 and 2.2. Pseudo-discount curves could also be constructed in a dual curve environment, by using this method along with the techniques described for example in [AB13] and [Whi12].

2.2.3 Calibration of the model

This section is about the calibration of our model. Section 2.2.3 discusses the calibration of the liquid part of the curve to swaps, 2.2.3 is about calibration by using swaps, caps and swaptions, and 2.2.3 describes curve extrapolation. Section 2.3 describes another way of choosing the parameters, by applying cross-validation to forecasts under the historical measure.

Calibration of the liquid part

Considering that there are N quoted swaps used for constructing the discount curve as of today, and at most M coupon payment dates for all the swaps, we let V be the vector of current values for the market swaps with length equal to N. C is the $N \times M$ matrix containing in each row, the swaps' coupon payments. P is the vector of discount factors that we are trying to derive, having a length equal to M.

Three methods might be envisaged for calibrating the model, depending on the data at hand:

- A method to be used **if an exact fit is required**, and can be found. That is, if we require $V = CP$

- A method to be used **if an exact fit cannot necessarily be found**, but approximated

- A method to be used **when the dataset is noisy, and a smooth curve is required**

These three methods are described hereafter, and numerical examples can be found in section 2.4.

- **If an exact fit is required**, then it is possible to guess *reasonable* values for a and σ (say, a between 0.05 and 1, and σ between 1% and 5%), and use an iterative curve calibration (also known as *bootstrapping*, but different from statistical bootstrap resampling) to solve $V = CP$. On figure 2.1, we can observe that the discount rates obtained for 1000 values of $a \in [0.1, 10]$ and $\sigma \in [0, 0.1]$ do not exhibit particular differences at quoted swaps maturities. The corresponding forward rates in figure 2.2 exhibit more differences.

  This type of method was used for any vanilla swap before the 2007 crisis, no matter its tenor. It is relevant only for extracting discount factors from OIS which are considered to be perfectly collateralized, or in Solvency II context. As of 2016, single curve construction in Solvency II, is applied to IRS, along with a parallel CRA, comprised between 10bps and 35bps.

  In order to describe the curve's calibration procedure, we will use a formulation similar to the one in [AP10]. We let $T_1 < \ldots < T_n$ be the maturity dates of OIS or IRS minus CRA, with the same currency on both legs. The swap payment dates occur at dates t_j, with a frequency belonging to $\{1\,month,\ 3\,months,\ 6\,months,\ 1\,year\}$.

  The single curve construction, in the specific Hull & White-consistent case treated in this paper, is made as follows:

  1. Guess a and σ: any *reasonable* values for a and σ will produce an *exact* fit for discount factors and discount rates (cf. figure 2.1)

2. *Loop on i:* At each step T_i corresponding to the i^{th} input swap maturity, suppose that the discount factors and b_js are known for any $t_j < T_i$

3. Make a guess for b_i

4. Use results from section 2.2.2, to derive the discount factors at intermediate swap payment dates: $T_{i-1} \leq t_j \leq T_i$. No interpolation is required.

5. Calculate V_i, the value of the i^{th} swap. While $V_i \neq 0$ return to point 2. Typically, the points 3 to 5 are solved iteratively with a root search algorithm.

- **If no solution is available for equation** $V = CP$ by iterative curve calibration, then similarly to [And07], it is possible to search for P minimizing:

$$\frac{1}{2N}\left(V - CP\right)^T W^2 \left(V - CP\right) \tag{2.22}$$

W, a diagonal matrix of weights is used. These weights are based on inverse duration, such as proposed by [Bli97]; with elements:

$$w_j = \frac{1/d_j}{\sum_{j=1}^{N} 1/d_j} \tag{2.23}$$

Weights such as w_j's are commonly used to give more importance to the short end of the curve, which is hence fitted more accurately. But other weighting schemes might be envisaged.

- **If the swaps data are noisy, or if one is interested in fitting smoothly noisy bonds data** a third method could be envisaged. It consists in penalizing the possibly large changes in forward rates' (approximate) second derivatives and/or in b_is. The objective function to be minimized is:

$$\frac{1}{2N}\left(V - CP\right)^T W^2 \left(V - CP\right) + \lambda_1 \sum_{i=1}^{N} (f_i'' - f_{i-1}'')^2 + \lambda_2 \sum_{i=1}^{N} (b_i - b_{i-1})^2 \tag{2.24}$$

where f_i'' is the approximate second derivative (using finite differences) of the discrete forward rates at time T_i. Typically, λ_1 and λ_2 can be found by cross-validation. An example can be found in section 2.4.

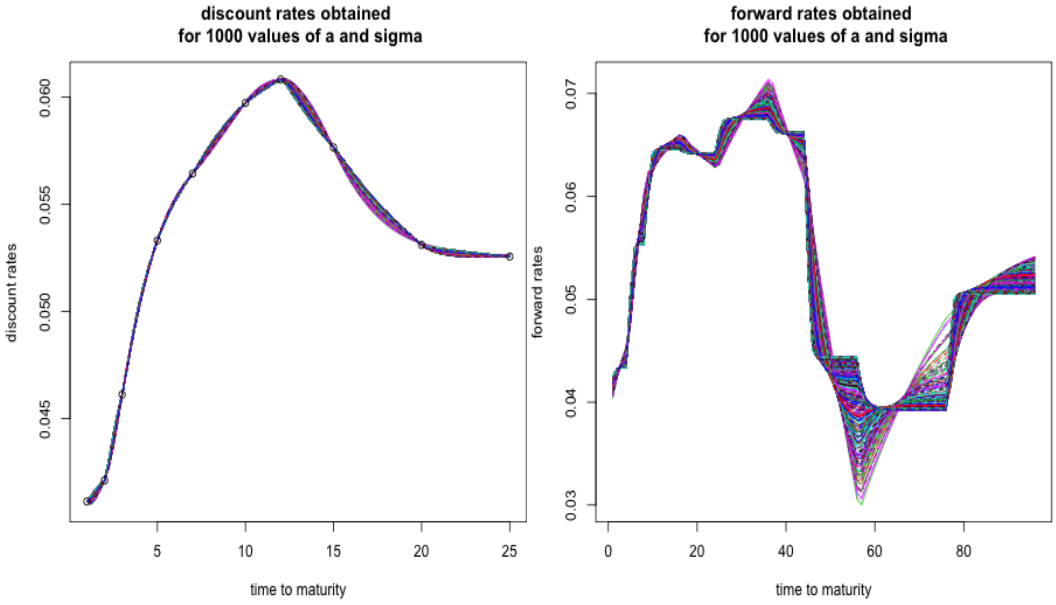

Fig. 2.1.: Discount rates obtained for 1000 values of $a \in [0.1, 10]$ and $\sigma \in [0, 0.1]$

Fig. 2.2.: Forward rates obtained for 1000 values of $a \in [0.1, 10]$ and $\sigma \in [0, 0.1]$

Calibration using derivatives

Another way for picking a and σ might be to calibrate the underlying short-rate model to a set of caps and swaptions. The optimization procedure would involve the following steps: choosing a and σ, construct the initial curve with an *exact fit* using the results described in section 2.2.3; use it as an input for theoretical caps and swaptions prices formulas implied by the underlying short-rate model, until a and σ which minimize the difference between theoretical and market prices for caps and swaptions are found.

Curve extrapolation

Using the Hull and White extended Vasicek model, it is possible to derive the instantaneous forward rates from the discount factors formula. We can write:

$$f(0,t) = -\frac{\partial log(P(0,t)}{\partial t} = X_0 e^{-at} + a\int_0^t e^{-a(t-u)} b(u)du - \frac{\sigma^2}{2}\phi^2(t) \qquad (2.25)$$

Hence, in our framework, using the fact that $t \mapsto b(t)$ is piecewise constant, we can also write:

$$f^M(0,t) = X_0 e^{-at} + a\sum_{i=1}^{n} b_i \left[\phi(t - T_{i-1} \wedge t) - \phi(t - T_i \wedge t)\right] + a b_{n+1}\phi(t - T_n \wedge t) - \frac{\sigma^2}{2}\phi^2(t) \tag{2.26}$$

This formula directly provides an input for the simulation of Hull & White short-rate, with parameters a, σ and $b_1, \ldots, b_n$ previously calibrated to market data.

Hence, let t grow to ∞, we have:

$$f^M(0,\infty) = b_{n+1} - \frac{\sigma^2}{2a^2} \tag{2.27}$$

If we assume that the UFR is exogenously chosen, and denote it by f_∞, we are able to derive the parameter b_{n+1} as:

$$b_{n+1} = f_\infty + \frac{\sigma^2}{2a^2} \tag{2.28}$$

This enables to re-write equation (2.19), when extrapolation is required, as:

$$I_{n+1}(t) = \sum_{k=1}^{n} b_k \left(\xi(t - T_{k-1} \wedge t) - \xi(t - T_k \wedge t)\right) + \left(f_\infty + \frac{\sigma^2}{2a^2}\right)\xi(t - T_n \wedge t) \tag{2.29}$$

If **a fixed *ultimate forward rate* (UFR) is defined exogenously**, one can increase or decrease the parameter a, to achieve a convergence of $f^M(0,t)$ to f_∞ at a pre-specified maturity. A period of convergence τ_{cv} after the *Last Liquid Point* (LLP) is defined. Starting from a low value such as $a = 0.1$, a is increased until:

$$f^M(0, LLP + \tau_{cv}) = f_\infty$$

or

$$|f^M(0, LLP + \tau_{cv}) - f_\infty| < tol$$

for a given σ, and a given numerical tolerance tol.

Otherwise, an ***ultimate forward rate* (UFR) can be derived from market data**. A static discount curve is fitted to a fraction of the quoted swaps available, called the *training* set. After the construction of the curve on this fraction of the data, we

evaluate how well, when extrapolated to a given exogenous UFR, it would price the remaining swaps in a *test* set.

The LLP provided by the prudential authority (as of 2016, a maturity 20 years), could be used to define the frontier between the *training* and *test* set. Otherwise, one can define a percentage of the swaps data to be used as a *training* dataset, for example 80% or 90% of the available swaps.

Both of these methods for curve extrapolation are applied in the numerical examples, in section 2.4.

2.3 Forecasting with Functional PCA

The idea that a few principal components explain a major part of the changes in bonds returns originates from [Lit+91]. This idea is now well accepted and applied to yield curve forecasting; the interested reader could refer to [DL06] or [Chr+11] for example.

We use a similar rationale, but apply it somewhat differently. The changes in the swap curve over time, are explained by the changes observed in the calibrated parameters b_is over time. Considering the fact that our model for fitting each cross section of yields is already *overparametrized* (as it uses at least as much parameters as swap rates available in the input dataset), the use of models such as an unrestricted Vector Autoregressive (VAR) to predict the b_is could lead to poor forecasts, with high variance.

Functional Principal Components Analysis in the spirit of [RD91] and [RS05], and more precisely Functional Principal Components Regression, was hence seen as one of the most immediate candidate to achieve a reduction of the problem's dimension. This method is used for example by [HU07] for forecasting log mortality rates. It has also been applied in finance, for example in [**Benko2007Functional**].

We consider functional data of the form:

$$b_x^{a,\sigma}(t) \tag{2.30}$$

These are the parameters b_is obtained by fitting each cross section of swap rates; observed at increasing times $t \in \{t_1, \dots, t_N\}$, for increasing maturities $x \in \{x_1, \dots, x_p\}$.

The calibration method is the one described in section 2.2.3, with a and σ kept fixed over time.

Finding the Functional Principal Components

Using the approach described in details in [RS05], we let **B** be the matrix containing at line i and column j:

$$\mathbf{B}_{i,j} = b_{x_j}^{a,\sigma}(t_i) \tag{2.31}$$

With $i = 1, \ldots, N$ and $j = 1, \ldots, n$, $n > p$. For each cross section of b_is calibrated at time t_i, a cubic spline interpolation is applied to $x \mapsto b_x^{a,\sigma}(t_i)$, so that the b_is values are now equally spaced on a larger grid of maturities spanning $[x_1, x_p]$. Let w be the fixed interpolation step applied to $x \mapsto b_x^{a,\sigma}(t_i)$ on $[x_1, x_p]$, and:

$$\mathbf{V} = \frac{1}{N}\mathbf{B}^T\mathbf{B} \tag{2.32}$$

V is the covariance matrix of the b_i's, when we consider that the columns of **B** have been centered. We are then looking for the vectors $\xi^{a,\sigma}$, the (approximate) functional principal components, verifying:

$$w\mathbf{V}\xi^{a,\sigma} = \rho\xi^{a,\sigma} \tag{2.33}$$

This is equivalent to searching the eigenvalues and eigenvectors of **V**, so that:

$$\mathbf{V}u = \lambda u \tag{2.34}$$

and $\rho = w\lambda$. This problem of finding eigenvalues and eigenvectors of **V** is typically solved by using the Singular Value Decomposition (SVD) of **B**, and taking the normalized right singular vectors as functional principal components. The interested reader can refer to [**jolliffe2002principal**] and [RS05] for details. Another interesting resource on Functional Principal Component Analysis is [**shang2014survey**].

Forecasting using Principal Components regression

Having obtained the functional principal components, a least squares regression of the cross sections of b_is is carried out. The b_is are expressed as a linear combination of the previously constructed functional principal components, plus an error term:

$$\forall t \in \{t_1, \ldots, t_N\}\,,\; b_x^{a,\sigma}(t) = \beta_{t,0} + \sum_{k=1}^{K} \beta_{t,k}\xi_k^{a,\sigma}(x) + \epsilon_t(x) \tag{2.35}$$

K is the number of functional principal components. These functional principal components are not highly correlated by construction, so that we can use univariate time series forecasts for each of the $K+1$ time series, and h-step ahead forecasts of the b_is as:

$$\hat{b}_x^{a,\sigma}(t+h) = \hat{\beta}_{t+h|t,0} + \sum_{k=1}^{K} \hat{\beta}_{t+h|t,k} \xi_k^{a,\sigma}(x) \tag{2.36}$$

Once the forecasts $\hat{b}_x^{a,\sigma}(t+h)$ are obtained, they can be plugged into formulae from section 2.2.2 to deduce h-step ahead forecasts for the discount factors and discount rates.

For choosing *good* values for a, σ and K, we typically used a cross-validation on grids of values for these three parameters, and rolling origin estimation/forecasting, as described in section 2.4.

2.4 Numerical examples

In order to illustrate how the methods described in the previous sections work, we use IRS and OIS data from [And07], [AP10], [AB13], an example of bonds data from [HW06]; *a curve where all cubic splines produce negative forward rates*. For forecasting the curves, we use market EUR 6M IRS data, (from which we give detailed summaries) with a CRA adjustment equal to 10bps.

For the data from [AP10], we assume that the swaps cash-flows payments occur on an annual basis as for OIS. From [AB13], we consider mid quotes from Eonia OIS and 6-month Euribor IRS as of December 11, 2012. These data sets are all reproduced in the appendices.

In section 2.4.1, four calibration methods are tested to illustrate section 2.2.3. The method proposed in this paper [1] is denoted by CMN. It is compared to two iterative curve calibration methods, with linear (LIN) and natural cubic splines (SPL) interpolation on missing dates, and the [SW01] method (SW). Section 2.4.1 also illustrates 2.2.3. We use a dataset from [And07]; a direct *bootstrapping* without regularization produces wiggly spot and forward rates. The effects of the regularization of approximate second derivative for forward rates and calibrated b_i's is illustrated. Such a regularization could also be applied to noisy bonds data.

[1] Actually applied to Hull & White model, but which can be applied to other short-rate models.

In section 2.4.1, the interpolation method is tested on a *curve where all cubic methods produce negative forward rates*, from [HW06]. Section 2.4.2 illustrates the possible extrapolation methods described in section 2.2.3. 2.4.3 illustrates the curves' forecasting method introduced in section 2.3.

The discount factors usually display no particular subtleties, so they are deliberately omitted. We present discount rates and discrete forwards instead, and the discrete forwards are taken to be 3-month forward rates.

2.4.1 Curve calibration

On swaps data from [AP10]

Below on figures 2.3, 2.4, 2.5, 2.6, are the discount rates and discrete forwards obtained for the four methods described in the previous section; two *bootstrapping* methods, with linear (LIN) and natural cubic splines (SPL) interpolation on missing dates, the [SW01] method (SW), and the method described in 2.2.3 with an exact fit, denoted as CMN. The discount rates are presented as a dashed line, and the forward rates as a plain colored line.

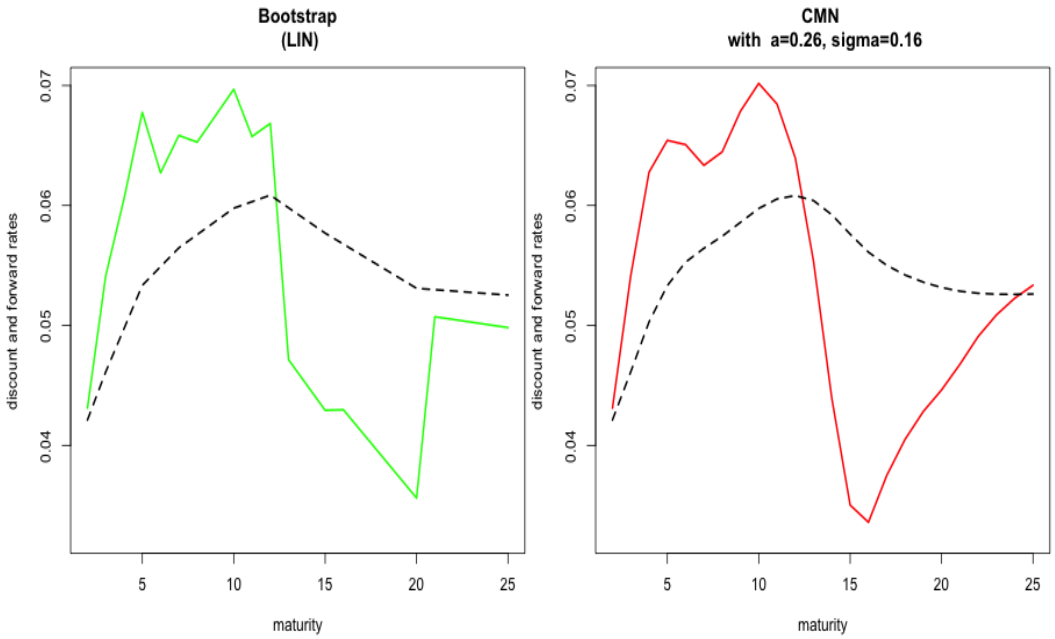

Fig. 2.3.: *Bootstrapping* with linear interpolation

Fig. 2.4.: CMN applied to Hull and White extended Vasicek

As demonstrated on figures 2.3, 2.4, 2.5 and 2.6, the discount rates produced by the four methods are quite similar. The discrete forward rates better exhibit the differences between them.

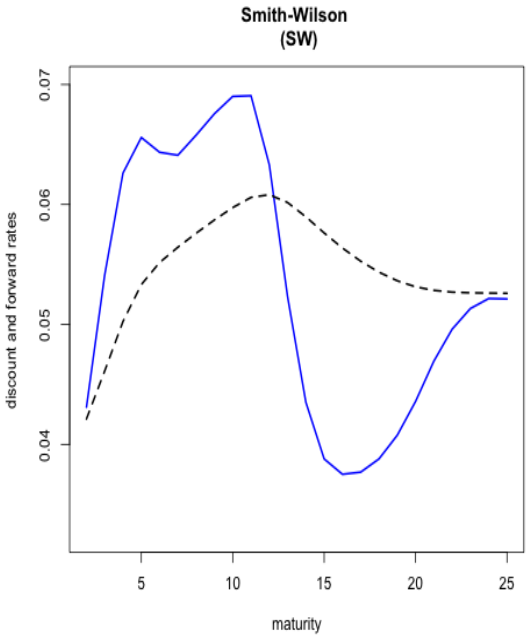

Fig. 2.5.: Smith-Wilson method

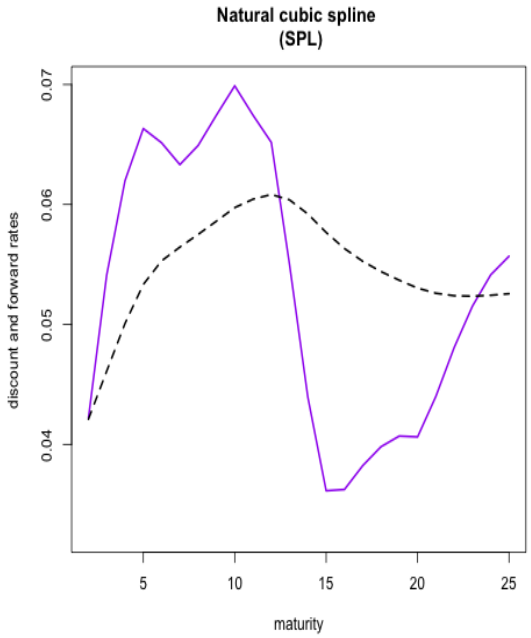

Fig. 2.6.: Natural cubic spline

Curve construction with linear interpolation between quoted swaps maturities (on figure 2.3), produces a saw-tooth like forward curve, which might not be desirable, and the other methods produce more regular forward curves.

For the method described in this paper - denoted as CMN on figure 2.4 - and applied to the extended Vasicek model, the discrete forwards (with an exact fit required, as described in section 2.2.3) reflect the fact that the discount factors' construction relies on a piece-wise constant function, with slight changes in first derivatives at quoted swap maturities. This effect remains very reasonable however, as the discrete forward curve is highly similar to those produced by the other models, and doesn't exhibit large changes at quoted swap maturities.

With a=0.2557, σ=0.1636, the parameters b_is from table 2.2 are obtained. They are presented along with the parameters ξ_is obtained by the method in [SW01], with $a = 0.1$. $a = 0.1$ is actually given as default parameter by Solvency II's technical specifications, and using the notations from QIS5 technical specifications.

On noisy swaps data

This section illustrates what may happen if the method from section 2.2.3 is applied directly to noisy data, without regularization of the parameters. We use data from [And07].

Tab. 2.2.: Parameters obtained for CMN and Smith-Wilson

Maturity	b_i	ξ_i
1	0.0661	-16.680
2	0.1894	23.556
3	0.2523	-0.8413
5	0.2523	-8.9116
7	0.2523	3.3552
10	0.2806	7.9600
12	0.2523	-14.098
15	0.2089	3.9119
20	0.2553	3.4828
25	0.2616	-1.9497

Figure 2.7 on the left describes the discount and forward rates obtained without regularization, with $a = 0.3655$ and $\sigma = 0.0037$. On the right, figure 2.8 describes the discount and forward rates obtained by minimizing the objective function in equation (2.24), and using the parameters $\lambda_1 = 1e-08$ and $\lambda_2 = 1e-05$, $a = 9.8891$ and $\sigma = 0.3957$.

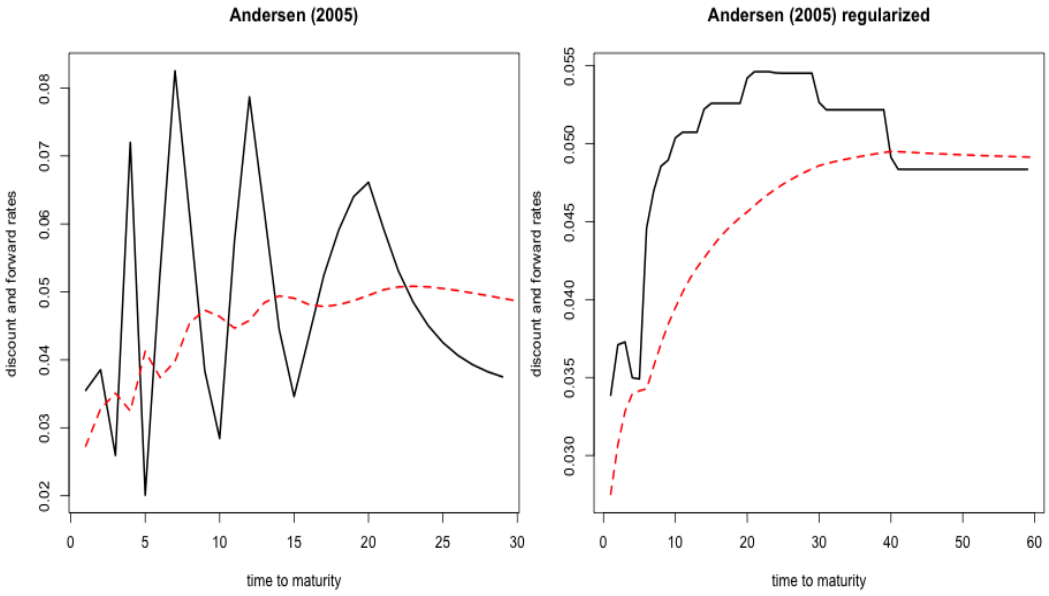

Fig. 2.7.: Curve calibration without regularization

Fig. 2.8.: Curve calibration with regularization

In order to pick λ_1 and λ_2, we make a grid search on couples (λ_1, λ_2). For each (λ_1, λ_2), a minimization based on derivatives is applied, with multiple restarts of the minimization algorithm. Multiple restarts avoid getting trapped into local minima.

Tab. 2.3.: Parameters obtained for unregularized and regularized CMN

Maturity	unregularized b_i	regularized b_i
0.5	0.0253	0.0281
1	0.1100	0.0363
1.5	-0.0078	0.0383
2	0.0929	0.0383
2.5	-0.0005	0.0380
3	-0.1360	0.0352
4	0.2901	0.0358
5	0.1975	0.0478
7	0.1654	0.0497
10	-0.0056	0.0515
12	0.1315	0.0533
15	0.1392	0.0554
20	0.0688	0.0553
30	0.039	0.0491

Table 2.3 contains both the unregularized and regularized b_is. The unregularized ones naturally exhibit a higher variance, because an exact fit to each swap rate in the noisy dataset is required. The regularized b_is exhibit a lower variance, at the expense of a higher bias in the fitting of the data from [And07].

On *a curve where all cubic methods produce negative forward rates*, with data from [HW06]

The dataset from this section is used in [HW06], and is described as *a curve where all cubic methods produce negative forward rates*. It is reproduced in the appendices.

Figure 2.9 illustrates the discount rates (dashed line), and discrete forward rates (plain coloured line) obtained with a linear interpolation of the bond yields. The discrete forward remain positive on all maturities, but again exhibit a sawtooth profile. As expected, the natural cubic spline on figure 2.10 produces negative discrete forward rates on this dataset.

Figures 2.11 and 2.12 present the results obtained on data from [HW06]. Figure 2.12 presents the sign of discrete forward rates as a function of a and σ. We consider that discrete forward rates' sign is negative if a least one discrete forward rate is negative. We observe on both figures 2.11 and 2.12 that a low value of a might produce negative forward rates on maturities comprised between 15 and 20. But a high value always produces positive forward rates.

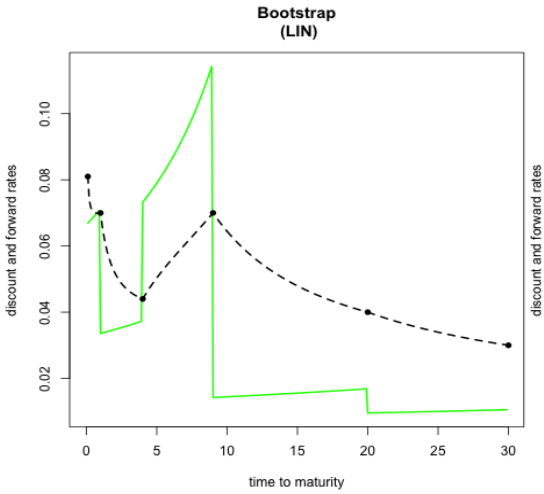

Fig. 2.9.: Linear interpolation on *a curve where all cubic methods produce negative forward rates*

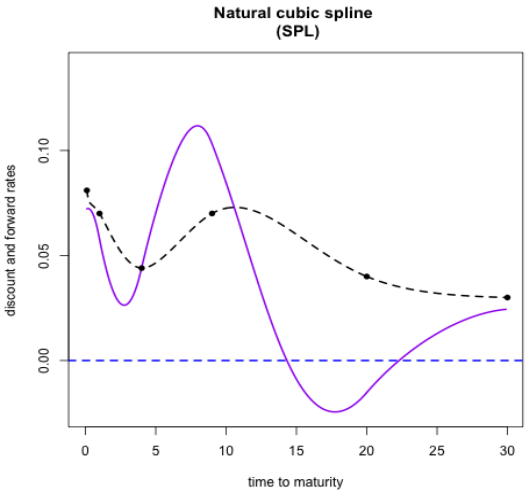

Fig. 2.10.: Natural cubic spline interpolation on *a curve where all cubic methods produce negative forward rates*

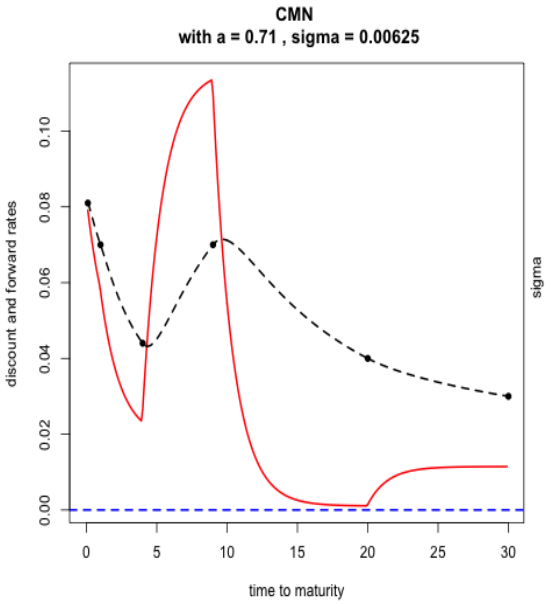

Fig. 2.11.: CMN interpolation on *a curve where all cubic methods produce negative forward rates*

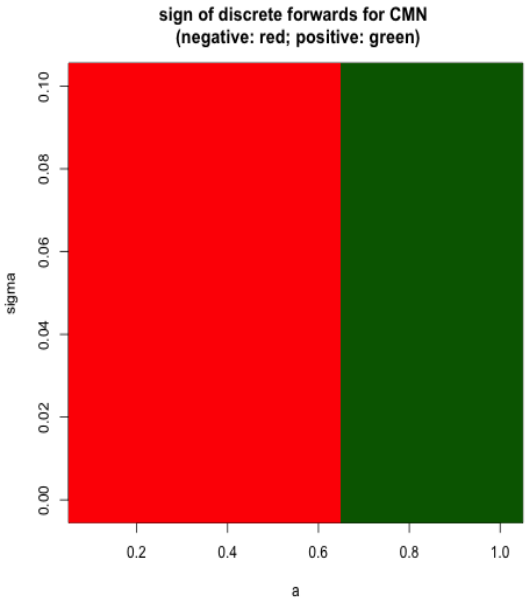

Fig. 2.12.: Sign of discrete forwards for CMN as function of a and σ, on *a curve where all cubic methods produce negative forward rates*

This is explained by what we saw in section 2.2.3: in the Hull and White extend Vasicek case, a controls the speed of convergence of forward rates to the UFR: the higher the a, the faster the convergence of forward rates to the UFR on long-term maturities. The parameters obtained by CMN interpolation (for producing figure 2.11), with $a = 0.71$ and $\sigma = 0.0062$ are presented in table 2.4.

Tab. 2.4.: Parameters obtained CMN with $a = 0.71$ and $\sigma = 0.0062$ on [HW06] data

Maturity	b_i
0.1	0.0718
1	0.0351
4	0.0018
9	0.1162
20	0.0011
30	0.0114

2.4.2 Curve extrapolation on data from [AB13]

In this section, we use the extrapolation methods described in 2.2.3, on OIS and IRS (with CRA adjustment equal to 10bps) data from [AB13].

With Solvency II technical specifications, on IRS + CRA

Extrapolation to a fixed UFR equal to 4.2% is tested, using CMN and the Smith-Wilson method. For both methods, the Last Liquid Point (LLP) is equal to 20 years, and convergence to the UFR is forced to 40 years after the LLP.

For the CMN method, the parameters are $a = 0.174$ and $\sigma = 0.0026$, and for the Smith-Wilson method, $a = 0.125$. The resulting discount and forward curves are presented in figures 2.13 and 2.14, and the parameters b_is and ξ_is in table 2.5.

The discount and forward curves produced by both methods are similar, as seen on figures 2.13 and 2.14. The convergence of the Smith-Wilson method to the UFR seems to be slighty faster. This is caused by the fact that for CMN, we use instantaneous forward rates to assess the convergence to the UFR, whereas for the Smith-Wilson method, we use discrete forwards.

With OIS data, and a data driven UFR

For this example, we use OIS data from [AB13] presented in the appendices. A training set containing 14 swap rates (90% of the dataset) with increasing maturities starting at 1 and ending at 20 is made up.

Tab. 2.5.: Parameters for CMN (b_i) and Smith-Wilson (ξ_i) extrapolation

Maturity	b_i	ξ_i
1	0.0019	-2.5888
2	0.0112	0.7585
3	0.0266	0.1415
4	0.0352	1.3153
5	0.0438	0.4726
6	0.0378	-0.8809
7	0.0399	1.2010
8	0.0387	-0.8965
9	0.0338	-0.3536
10	0.0376	0.7268
11	0.0363	-0.1582
12	0.0353	1.2852
13	0.0312	-1.9866
14	0.0239	0.4161
15	0.0285	0.7056
16	0.0211	-0.7112
17	0.0208	-1.7105
18	0.0182	1.9922
19	0.0248	-1.5542
20	0.0172	0.5125
21	0.0272	1.0148
22	0.0189	-2.1158
23	0.0025	3.4051
24	0.0021	-3.7822
25	0.0020	2.7013
26	0.0239	-2.8668
27	0.0195	2.2513
28	0.0274	-0.8877
29	0.0202	-7.1463
30	0.0326	8.5322

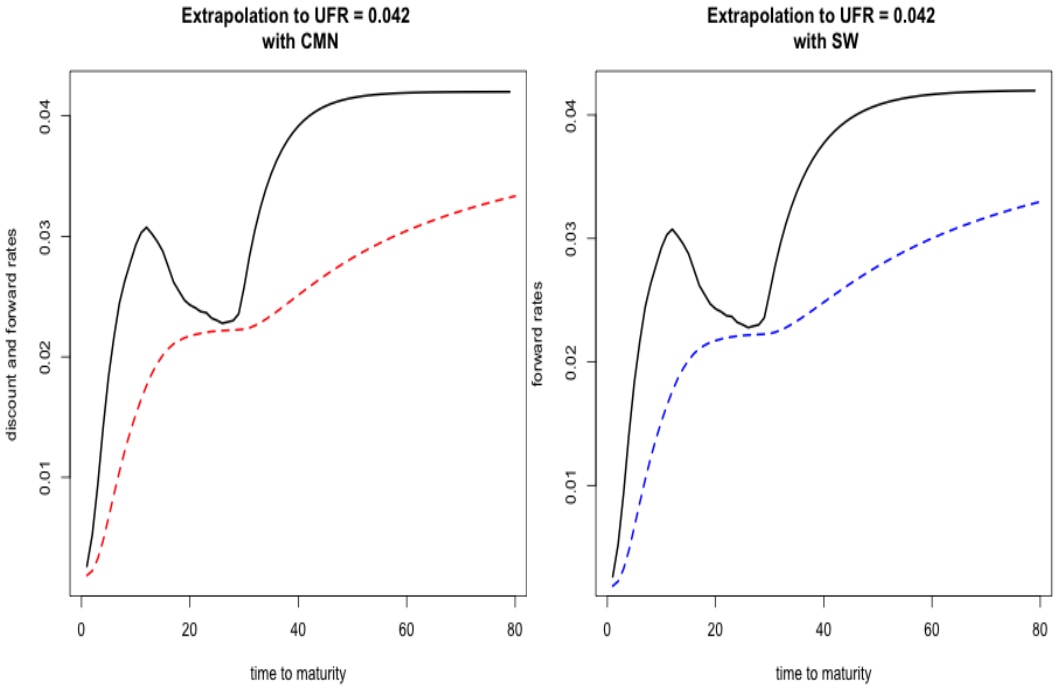

Fig. 2.13.: Extrapolation to $UFR = 4.2\%$ with CMN

Fig. 2.14.: Extrapolation to $UFR = 4.2\%$ with Smith-Wilson

This training set is used to construct the discount curve, which is then extrapolated to 30-year maturity and beyond, using different values for the UFR. The two remaining swaps, with maturities equal to 25 and 30, are placed into the test set.

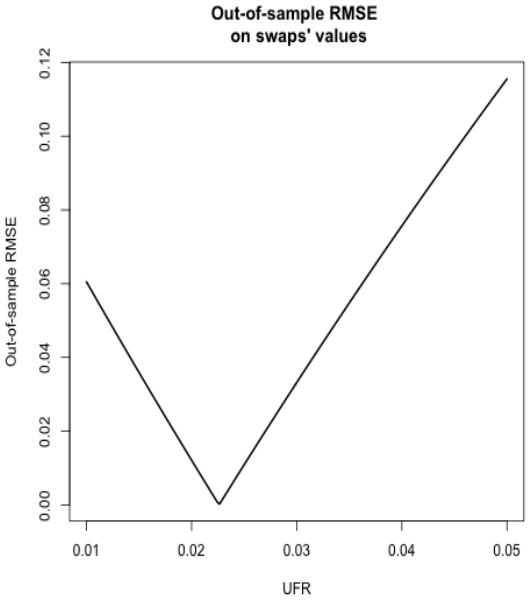

Fig. 2.15.: Out-of-sample RMSE on swap values, as a function of UFR

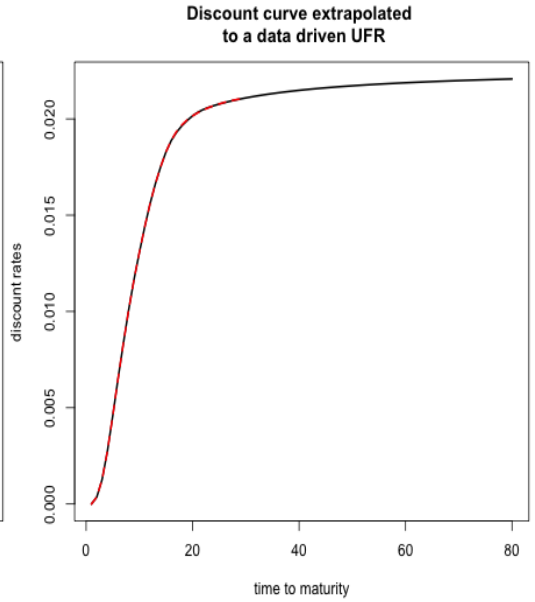

Fig. 2.16.: Extrapolation of OIS curve to a data driven $UFR = 0.0226$

Figure 2.15 presents the out-of-sample RMSE obtained on swaps values from the test set, as a function of UFR. This error decreases until $UFR = 0.0226$ (notice that this value would depend on the step chosen on the grid of UFRs), and then, starts to increase again. Figure 2.16 displays the discount curve constructed on the training set, extrapolated to a 80-year maturity with an UFR equal to 0.0226 (the one minimizing the out-of-sample RMSE on the chosen grid of UFRs) is presented.

2.4.3 12-months ahead forecast on historical IRS + CRA

In this section, we apply ideas from section 2.3 to real world IRS data observed monthly from december 2013 to april 2016, adjusted from a CRA equal to 10bps.

Figure 2.17 and table 2.6 are to be read together. They contain the informations on the spot rates derived from the IRS data adjusted from a CRA, using CMN with $a = 0.3655$ and $\sigma = 0.0037$ (other values than $a = 0.3655$ and $\sigma = 0.0037$ would produce the same results as the fitting is exact for many different values of these parameters).

The static curves are generally upward sloping, and as time passes, lower and lower spot rates are encountered. In addition, negative rates are observed in table 2.6; which is coherent with the current context.

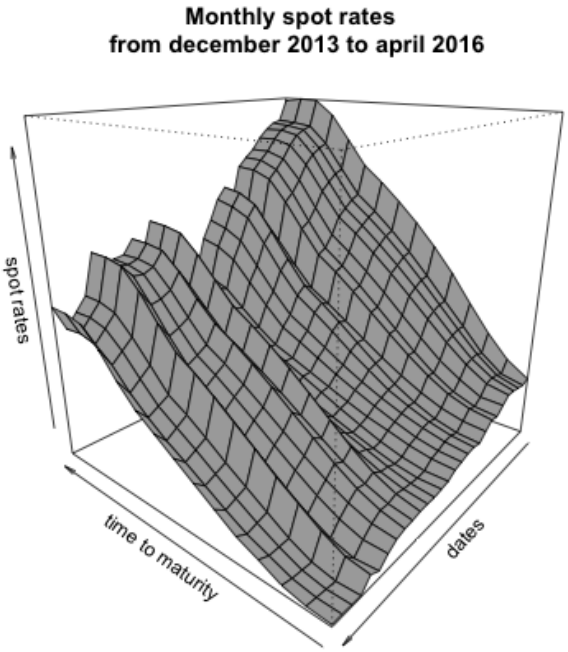

Fig. 2.17.: Spot rates observed from december 2013 to april 2016

Tab. 2.6.: Descriptive statistics for the spot rates observed from december 2013 to april 2016

Maturity	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.
1	-0.0026	-0.0008	0.0000	0.0003	0.0019	0.0031
3	-0.0023	0.0002	0.0009	0.0013	0.0028	0.0065
5	-0.0008	0.0017	0.0030	0.0037	0.0056	0.0117
10	0.0046	0.0059	0.0090	0.0101	0.0132	0.0211
15	0.0063	0.0097	0.0127	0.0141	0.0179	0.0258
20	0.0069	0.0113	0.0144	0.0157	0.0199	0.0272
30	0.0071	0.0118	0.0155	0.0164	0.0208	0.0270

Tab. 2.7.: Average out-of-sample error on real world IRS data + CRA

Method	Parameters	Avg. OOS error
CMN - `auto.arima`	$K = 5$, $a = 1$, $\sigma = 0.1555$	0.0031
CMN - `ets`	$K = 5$, $a = 1$, $\sigma = 0.2$	0.0037
NS - `auto.arima`	$\lambda = 1.8889$	0.0031
NS - `ets`	$\lambda = 1.8889$	0.0035
NSS - `auto.arima`	$\lambda_1 = 21$, $\lambda_2 = 21$	0.0027
NSS - `ets`	$\lambda_1 = 7$, $\lambda_2 = 3$	0.0035

Benchmarking the model

Benchmarks are subjective. The one presented in this section does not aim at showing that one method is always superior to the other. It aims at showing that the method presented in this paper produces forecasts which are (more than) reasonable, and actually close to other well-known methods forecasts (on this given dataset).

Forecasts from the model presented in section 2.3 are hence compared to those of two other models constructed in the spirit of by the [DL06]. The cross sections of yields described by figure 2.17 and table 2.6 are fitted by the [NS87] model (NS), and its extension by [Sve94] (NSS). The formulas for the spot rates from these models are respectively:

$$R^M(t,T) = \beta_{t,1} + \beta_{t,2}\left[\frac{1-e^{-T/\lambda}}{T/\lambda}\right] + \beta_{t,3}\left[\frac{1-e^{-T/\lambda}}{T/\lambda} - e^{-T/\lambda}\right] \tag{2.37}$$

and

$$R^M(t,T) = \beta_{t,1} + \beta_{t,2}\left[\frac{1-e^{-T/\lambda_1}}{T/\lambda_1}\right] + \beta_{t,3}\left[\frac{1-e^{-T/\lambda_1}}{T/\lambda_1} - e^{-T/\lambda_1}\right] \quad (2.38)$$

$$+ \beta_{t,4}\left[\frac{1-e^{-T/\lambda_2}}{T/\lambda_2} - e^{-T/\lambda_2}\right] \quad (2.39)$$

Forecasts $\hat{R}^M(t+h,T)$ are obtained by fitting univariate time series to the parameters $\beta_{t,i}, i = 1,\dots,4$ with automatic ARIMA (`auto.arima`) and exponential smoothing (`ets`) models from [HK08]. This automatic selection is done only for the sake of the benchmarking exercise, and in order to conduct the experience in fairly similar conditions for all the methods. In practice, a visual inspection and an actual study of the univariate time series would of course be required.

For all the methods the six methods, CMN, NS, NSS with `auto.arima` and `ets`, we obtain 12-months ahead forecasts, from rolling estimation windows of a fixed 6 months length, starting in december 2013. That is, the models are trained on 6 months data, and predictions are made on 12 months data; successively. The average out-of-sample RMSE are then calculated for each method, on the whole surface of observed and forecasted yields.

The *best* parameters for CMN are obtained by cross-validation, with $K \in \{2,3,4,5,6\}$, 5 values of a comprised between 0.9 and 1, and 10 values of σ comprised between 0 and 0.2. For NS and NSS, λ_1 and λ_2 are chosen by cross-validation, using the rolling estimation/forecasting we have just described.

Bootstrap simulation of 12-months ahead spot rates

In this section, we use the last 12 months of the dataset to construct the functional principal components. Using 12 months as the length of the fixed window for estimation, we get an average out-of-sample RMSE of 0.0026 (on a smaller number of testing samples than the 6 months estimation window, of course).

An AR(1) is fitted to the observed univariate time series $(\beta_{t,i})_t$, $i = 0,\dots,K$, with $a = 1$, $\sigma = 0.0089$, and $K = 3$ chosen by cross-validation. The three functional principal components' characteristics are summarized in table 2.8. We notice that the first functional principal component explains already 99.2415% of the changes in b_is, and the first three functional principal components selected by cross-validation explain 99.9220%.

Tab. 2.8.: Importance of Principal components

Indicator	PC1	PC2	PC3
Standard deviation	0.1286	0.2461	0.2246
Proportion of variance (in %)	99.2415	0.5489	0.1315
Cumulative Proportion (in %)	99.2415	99.7904	99.9220

Figure 2.18 presents the autocorrelation functions of the residuals of AR(1) fitted to $(\beta_{t,i}),\ i = 0, \dots, 3$ from april 2015 to april 2016. The residuals from AR(1) fitted to $(\beta_{t,i})_t,\ i = 1, \dots, 3$ could be considered as stationary, but those from the AR(1) fitted to $(\beta_{t,0})_t$ seems to be closer to an AR(4).

We denote these residuals by $(\epsilon_{t,i})_t,\ i = 0, \dots, 3$. In order to obtain simulations for the $(\beta_{t,i})_t,\ i = 0, \dots, 3$, it is possible to use a Gaussian hypothesis on the residuals. We choose to create one thousand bootstrap resamples with replacement of the $(\epsilon_{t,i})_t,\ i = 0, \dots, 3$ [2], denoted as $(\epsilon^*_{t,i})_t,\ i = 0, \dots, 3$, and create new pseudo values for $(\beta_{t,i}),\ i = 0, \dots, 3$:

$$\beta^*_{t,i} = \beta_{t,i} + \epsilon^*_{t,i},\ i = 0, \dots, 3$$

Having done this, AR(1) forecasts $\beta^*_{t+h|t,i}$ can be obtained, in order to construct:

$$\hat{b}^{a,\sigma,*}_x(t+h) = \hat{\beta}^*_{t+h|t,0} + \sum_{k=1}^{K} \hat{\beta}^*_{t+h|t,k} \xi^{a,\sigma}_k(x) \tag{2.40}$$

The $\hat{b}^{a,\sigma,*}_x(t+h)$ can then be plugged into formulae **??** and 2.19 to deduce simulations of h-step ahead forecasts for the discount factors and discount rates.

The simulations (1000) of 12-months ahead discount rates are presented in figures 2.19 and 2.20.

2.4.4 6-months and 36-months ahead forecast on longer historical data

In this second example, we use interest rate swaps data from the Federal Reserve Bank of St Louis website [3] observed monthly, from july 2000 to september 2016, with maturities equal to $1, 2, 3, 4, 5, 7, 10, 30$, and a tenor equal to three months.

[2] Even if for $\epsilon_{t,0}$, considering figure 2.18, this makes a strong stationarity assumption on the residuals.
[3] Available at https://fred.stlouisfed.org/categories/32299

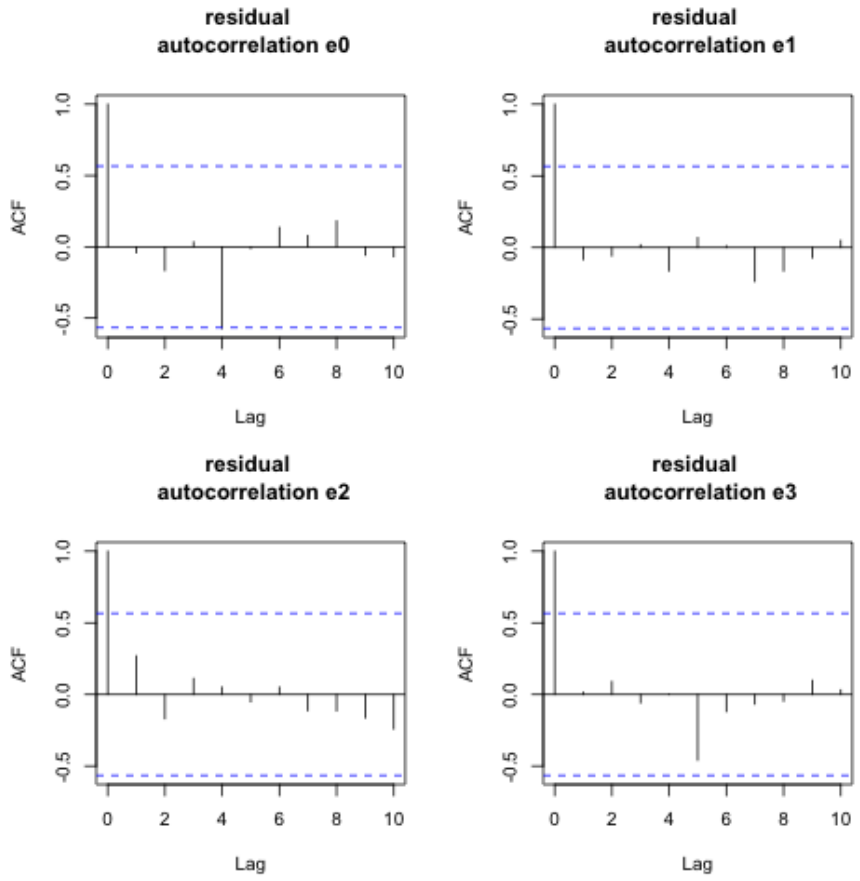

Fig. 2.18.: Autocorrelation functions for the residuals of univariate time series(AR(1)) on β_0, β_1, β_2, β_3

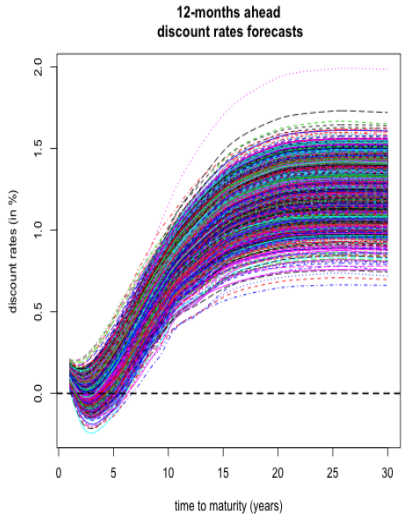

Fig. 2.19.: Curves simulated with principal components from april 2015 to april 2016, and bootstrap ressampling of the residuals

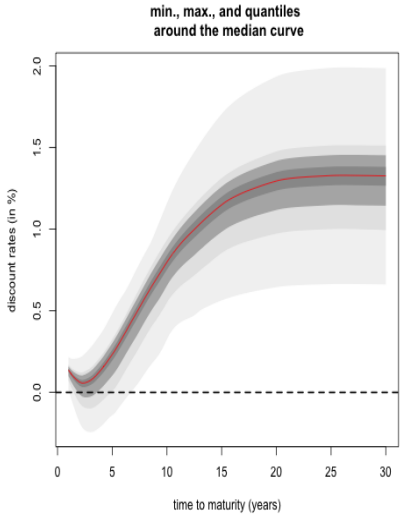

Fig. 2.20.: Min., Max., and quartiles around the median curve for the simulations

Tab. 2.9.: Descriptive statistics for fitted parameters b_is from april 2015 to april 2016

Maturity	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.
1	-0.0026	-0.0021	-0.0010	-0.0013	-0.0004	-0.0003
3	0.0000	0.0026	0.0040	0.0035	0.0048	0.0058
5	0.0025	0.0076	0.0030	0.0092	0.0108	0.0143
10	0.0115	0.0174	0.0090	0.0188	0.0208	0.0230
15	0.0122	0.0168	0.0127	0.0192	0.0220	0.0228
20	0.0117	0.0148	0.0144	0.0171	0.0195	0.0211
30	0.0080	0.0116	0.0155	0.0134	0.0150	0.0178

In figure 2.21, we represent the eight time series of swap rates, observed for each maturity $1, 2, 3, 4, 5, 7, 10, 30$, between july 2000 and september 2016. The swap rates for different maturities generally exhibit a decreasing trend, and are nearly equal to 0 by the end of 2016 for the shortest maturities.

Starting in 2006, the spreads between swap rates with different maturities start to narrow, until the end of 2007, and swap rates for short maturities are relatively high. This is the period corresponding to the Liquidity and Credit Crunch 2007-2008. Table 3.8 below presents the descriptive statistics for the data.

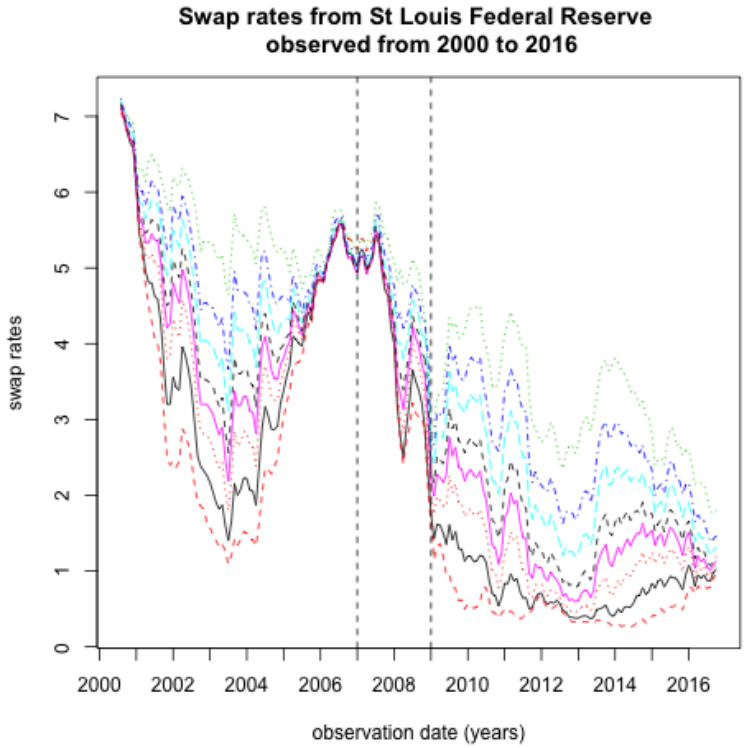

Fig. 2.21.: Swap rates data (in %) from St Louis Federal Reserve Bank, at maturities $1, 2, 3, 4, 5, 7, 10, 30$

Tab. 2.10.: Descriptive statistics for St Louis Federal Reserve data

Maturity	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.
1	0.0026	0.0050	0.0134	0.0211	0.0336	0.0705
2	0.0037	0.0078	0.0182	0.0239	0.0390	0.0712
3	0.0046	0.0108	0.0236	0.0269	0.0422	0.0714
4	0.0060	0.0134	0.0280	0.0296	0.0439	0.0715
5	0.0078	0.0167	0.0316	0.0319	0.0456	0.0717
7	0.0119	0.0215	0.0368	0.0354	0.0483	0.0720
10	0.0139	0.0261	0.0419	0.0388	0.0502	0.0724
30	0.0175	0.0327	0.0465	0.0440	0.0537	0.0720

We transformed these swap rates into zero rates by using a single curve calibration (that is, ignoring the counterparty credit risk) with linear interpolation between the maturities; one of the methods used in section 2.4.1. Then, as in the previous section, NS, NSS, CMN are used for fitting and forecasting the curves, with `auto.arima` applied to the factors.

We obtain 6-months and 36-months ahead forecasts, from rolling training/testing windows (as in the last section) with respectively, a fixed 6 and 36 months length. The average out-of-sample RMSE are then calculated for each method, on the whole set of observed and forecasted yields.

The *best* hyperparameters - associated with the lowest out-of-sample average RMSE - for each model are obtained through a search on a grid of values. For a 6-months horizon, they are (using the notations from section 2.4.3):

- NS: $\lambda = 1.6042$
- NSS: $\lambda_1 = 1.6250$ $\lambda_2 = 1.6250$
- CMN: $a = 177.8279$, $\sigma = 3.9473e-04$, $K = 6$

and for a 36-months horizon:

- NS: $\lambda = 1.4271$
- NSS: $\lambda_1 = 1.575$ $\lambda_2 = 1.575$
- CMN: $a = 14.6780$, $\sigma = 0.0011$, $K = 4$

The following results are obtained for the out-of-sample average RMSE:

Tab. 2.11.: Descriptive statistics for out-of-sample RMSE, for training window = 6 months, and testing window = 6 months

Method	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.	Std. Dev
NS	0.00101	0.00269	0.00409	0.00481	0.00595	0.01530	0.00296
NSS	0.00102	0.00269	0.00411	0.00481	0.00595	0.01537	0.00296
CMN	0.00115	0.00256	0.00396	0.00468	0.00580	0.01600	0.00302

Tab. 2.12.: Descriptive statistics for out-of-sample RMSE, for training window = 36 months, and testing window = 36 months

Method	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.	Std. Dev
NS	0.00356	0.00703	0.01044	0.01489	0.01609	0.21500	0.0213
NSS	0.00300	0.00690	0.01114	0.01484	0.01689	0.21570	0.0201
CMN	0.00402	0.00945	0.01279	0.01452	0.01917	0.03710	0.0070

Using tables 2.11, 3.9 and figures 2.22 and 2.23, we observe that CMN give results which are close to those from NS and NSS, with a lower average out-of-sample RMSE in both cases. For a training window equal to six months, and testing window of six months, the results obtained by the three methods are pretty similar, and the same performance is observed for the three during the financial crisis. For a training and testing window of thirty six months length, CMN has a lower mean and standard deviation for out-of-sample RMSE overall, but doesn't perform the best in the period of financial crisis 2007-2009.

2.5 Conclusion

In this paper, we introduced a method for swap discount curve construction and extrapolation. This method relies on the closed form formulas for discount factors available in exogenous short-rate models. We presented different ways to calibrate and extrapolate the model on different data sets from the existing literature. Moreover, we showed that the model's parameters contain a certain predictive power, enabling to obtain swap curves' forecasts, with predictive distribution.

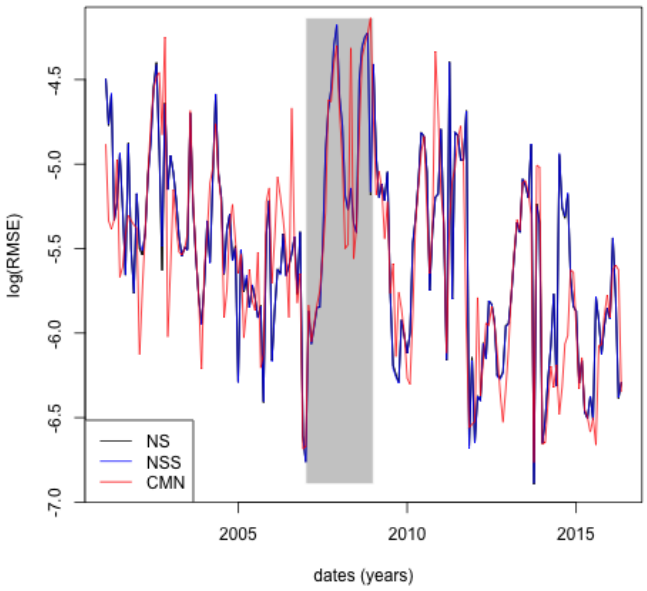

Fig. 2.22.: log(out-of-sample RMSE) for training window = 6 months, and testing window = 6 months

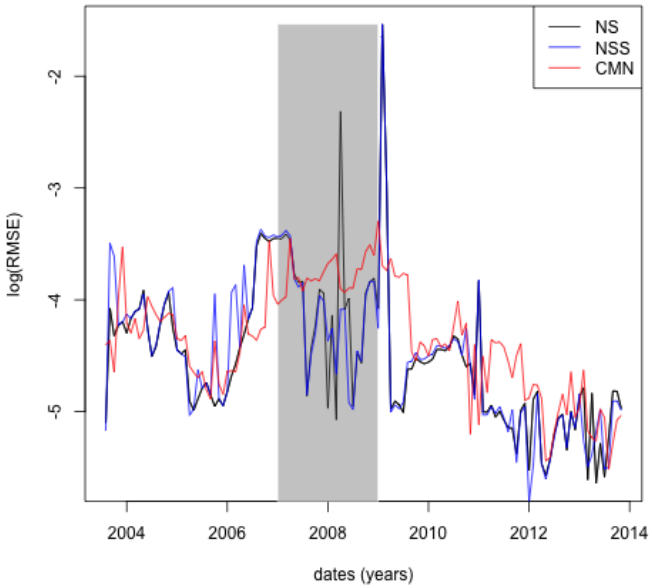

Fig. 2.23.: log(out-of-sample RMSE) for training window = 36 months, and testing window = 36 months

2.6 Appendix

2.6.1 Data from [AP10]

Maturity	Swap Par Rate
1	4.20%
2	4.30%
3	4.70%
5	5.40%
7	5.70%
10	6.00%
12	6.10%
15	5.90%
20	5.60%
25	5.55%

2.6.2 Data from [And07]

Maturity	Swap Par Rate
0.5	2.75%
1	3.10%
1.5	3.30%
2	3.43%
2.5	3.53%
3	3.30%
4	3.78%
5	3.95%
7	4.25%
10	4.50%
12	4.65%
15	4.78%
20	4.88%
30	4.85%

2.6.3 Data from [HW06]

Maturity	Continuous yield
0.1	8.10%
1	7.00%
4	4.40%
9	7.00%
20	4.00%
30	3.00%

2.6.4 Data from [AB13]

Maturity	EUR6M IRS	Eonia OIS
1	0.286%	0.000%
2	0.324%	0.036%
3	0.424%	0.127%
4	0.576%	0.274%
5	0.762%	0.456%
6	0.954%	0.647%
7	1.135%	0.827%
8	1.303%	0.996%
9	1.452%	1.147%
10	1.584%	1.280%
11	1.703%	1.404%
12	1.809%	1.516%
13	1.901%	-
14	1.976%	-
15	2.037%	1.764%
16	2.086%	-
17	2.123%	-
18	2.150%	-
19	2.171%	-
20	2.187%	1.939%
21	2.200%	-
22	2.211%	-
23	2.220%	-
24	2.228%	-
25	2.234%	2.003%
26	2.239%	-
27	2.243%	-
28	2.247%	-
29	2.251%	-
30	2.256%	2.038%
35	2.295%	-
40	2.348%	-
50	2.421%	-
60	2.463%	-

3 Multiple time series forecasting using quasi-randomized functional link neural networks

3.1 Introduction

In this chapter, we are interested in obtaining forecasts for multiple time series, by taking into account the potential nonlinear relationships between their observations. This type of problem has been tackled recently by [Ext+16], who applied kernel regularized least squares to a set of macroeconomic time series. The Long Short-Term Memory neural networks (LSTM) architectures (introduced by [HS97]) are another family of models, which are currently widely used for this purpose. As a basis for our model, we will use (quasi-)randomized neural networks known as Random Vector Functional Link neural networks (RVFL networks hereafter)

The forecasting method described in this chapter, provides useful inputs for Insurance quantitative Risk Management models; the interested reader can refer to [Bon+15] for example.

To the best of our knowledge, randomized neural networks were introduced by [Sch+92], and the RVFL networks were introduced by [Pao+94]. An early approach for multivariate time series forecasting using neural networks is described in [Cha+92]. They applied a *back propagation* algorithm from [Rum+88] to trivariate time series, and found that the combined training of the series gave better forecasts than a separate training of each individual series. The novelty of the approach described in this chapter is to derive an RVFL model for multiple time series, under two separate regularization constraints on the parameters, as it will be described in details in section 3.2.3.

RVFL networks are *multilayer feedforward* neural networks, in which there is a *direct link* between the predictors and the output variable, aiming at capturing the linear relationships. In addition to the *direct link*, there are new features: the hidden nodes (the dataset is augmented), that help to capture the nonlinear relationships between the time series. These new features are obtained by random simulation over a given

interval. More details on the *direct link* and the hidden nodes will be provided in the next section.

The RVFL networks have been successfully applied to solving different types of classification and regression problems; see for example [DC10]. More specifically, they have been applied to univariate time series forecasting by [Ren+16]. A comprehensive survey can be found in [ZS16]; where a large number of model specifications are tested on classification problems, including changing the range of hidden layer's randomization.

Here, we will use RVFL networks with one hidden layer. And instead of relying on fully randomized nodes, we will use sequences of deterministic quasi-random numbers. Indeed, with fully randomized nodes, the model fits obtained are dependent on the choice of a simulation *seed*. Typically, a different fitting solution would be obtained for each *seed*.

In our various numerical examples from section B.4, we will apply the RVFL networks to forecasting trivariate time series, notably (but not only) in a Dynamic Nelson Siegel (DNS) framework (see [NS87], [DL06]). We will obtain point forecasts and predictive distributions for the series, and see that in this RVFL framework, one (or more) variable(s) can be stressed, and influence the others. More precisely, about this last point, it means that it is possible, as in dynamic regression models ([Pan12]) to assign a specific future value to one regressor, and obtain forecasts of the remaining variables. Another advantage of the model described here, is its ability to integrate multiple other exogenous variables, without overfitting in-sample data.

3.2 Description of the model

The general procedure for obtaining the model's optimal parameters and predictions is summarized in figure 3.1.

This procedure is described in details in the next sections, especially sections 3.2.3 and 3.2.4.

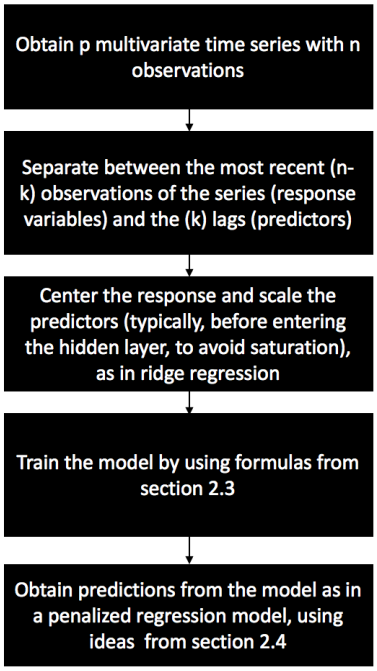
Obtain p multivariate time series with n observations
Separate between the most recent (n-k) observations of the series (response variables) and the (k) lags (predictors)
Center the response and scale the predictors (typically, before entering the hidden layer, to avoid saturation), as in ridge regression
Train the model by using formulas from section 2.3
Obtain predictions from the model as in a penalized regression model, using ideas from section 2.4

Fig. 3.1.

3.2.1 On a single layer RVFL networks

We rely on single layer feed forward neural networks (SLFN). Considering that an output variable $y \in \mathbb{R}^n$ is to be explained by a set of observed predictors $Z^{(j)} \in \mathbb{R}^n$, $j \in \{1, \ldots, p\}$, the RVFL networks we will use to explain y can be described for $i \in \{1, \ldots, n\}$ as:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j Z_i^{(j)} + \sum_{l=1}^{L} \gamma_l \, g\left(\sum_{j=1}^{p} W^{(j,l)} Z_i^{(j)} \right) + \epsilon_i$$

g is called *activation function*, L is the number of nodes in the hidden layer, $W^{(j,l)}$ are elements of the hidden layer, and the parameters β_j and γ_l are to be learned from the observed data $Z^{(j)}$, $j \in \{1, \ldots, p\}$. The ϵ_i's are the residual differences between the output variable values and the RVFL model.

This type of model can be seen as a one explaining y_i, by finding a compromise between linear and potentially non-linear effects of the original predictors $Z^{(j)}$ and transformed predictors

$$\Phi(\mathbf{Z})^{(l)} = g\left(\sum_{j=1}^{p} W^{(j,l)} Z_i^{(j)} \right)$$

$\{1, \ldots, L\}$ on the response. Common choices for function g in neural networks regression are the sigmoïd activation function

$$g : \; x \mapsto \frac{1}{1 + e^{-x}}$$

the hyperbolic tangent function,

$$g : \; x \mapsto tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

or the Rectified Linear Units, known as ReLU

$$g : \; x \mapsto max(x, 0)$$

The main differences between the RVFL framework and a *classical* SLFN framework are:

- The inclusion of a linear dependence between the output variable and the predictors: the *direct link*, $\beta_0 + \sum_{j=1}^{p} \beta_j Z_i^{(j)}$

- The elements $W^{(j,l)}$ of the hidden layer are typically not trained, but randomly and uniformly chosen on a given interval. Different ranges for these elements of the hidden layer are tested in [ZS16]).

Solving for the optimal parameters β_j's and γ_l's can be done by applying directy a least squares regression of y on the set of observed and transformed predictors. But since these input predictors are likely to be highly correlated - especially in our setting, with time series data - we do not search each of these parameters on the entire line, but in restricted regions where we have:

$$\sum_{j=1}^{p} \beta_j^2 \leq u$$

and

$$\sum_{l=1}^{L} \gamma_l^2 \leq v$$

for $u, v \in \mathbb{R}^*$. That is, applying some kind of Tikhonov regularization or ridge regression ([HK70]) of y on the set of observed and transformed predictors. Having two constraints instead of one, allows for more flexibility in the covariance structure between the predictors and the output, with β_j's and γ_l's moving in separate balls. For these constraints to be applicable, the input variables will need to be standardized, so as to be expressed on the same scales, and the response variable will be centered.

Imposing these restrictions to the model's parameters increases their interpretability - by reducing their variance -, at the expense of a slight increase in in-sample bias. It also prevents the model from overfitting the data as in ridge regression ([HK70]). One of the advantages of RVFL networks is that they are relatively fast to train, due to the availability of closed-form formulas for the model's parameters, as it will be presented in the next section.

On the other hand, RVFL networks incorporate some randomness in the hidden layer, which makes each model relatively dependent on the choice of a simulation *seed*. Each *seed* would indeed produce a different set of parameters β_j's and γ_l's for the model. For that reason, we will also use sequences of deterministic quasi-random numbers in the hidden layer. The elements $W^{(j,l)}$ of the hidden layer are taken from a quasi-random (deterministic) *sobol* sequence on $[0, 1]$, which is shifted in such a way that they belong to $[-1, 1]$.

Sobol sequences are part of quasi-random numbers, which are also called *low discrepancy* sequences. As described intuitively in [BT97], the discrepancy of a sequence of N points in a subcube $V \in [0,1)^d$ is defined as:

$$sup_{V \in [0,1)^d} | \frac{number\ of\ points\ in\ V}{N} - v(V) |$$

where $v(V)$ is the volume of V. It describes how well the points are dispersed within V. The idea is to have points which are more or less equidispersed in V. [JK08] describe the generation of the i^{th} term, j^{th} component ($x_{i,j}$) of a Sobol sequence. The generation starts with obtaining the binary representation of i. That is, obtaining i as:

$$i = \sum_k i_k 2^k = (\dots\ i_3\ i_2\ i_1)_2$$

For example, $5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ can be expressed as $(101)_2$ in binary representation. Then, by using the sequence of bits describing i in base 2, we can obtain $x_{i,j}$ as:

$$x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \dots \tag{3.1}$$

Where $\oplus$ is a bitwise exclusive-or operation, and the numbers $v_{i,j}$ are called the direction numbers, defined for $k \geq 1$ as:

$$v_{k,j} = \frac{m_{k,j}}{2^k} = \frac{2a_{1,j}m_{k-1,j} \oplus 2^2 a_{2,j} m_{k-2,j} \oplus \dots \oplus 2^{s_j - 1} a_{s_j - 1,j} m_{k-s_j+1,j} \oplus 2^{s_j} m_{k-s_j,j} \oplus m_{k-s_j,j}}{2^k} \tag{3.2}$$

A few details on equation 3.2:

- The bitwise exclusive-or operation $\oplus$ applied to two integers p and $q \in \{0,1\}$ returns 1 if and only if one of the two (but not both) inputs is equal to 1. Otherwise, $p \oplus q$ is equal to 0.

- The second term of the equation relies on primitive polynomials of degree s_j, with coefficients $a_{i,j}$ taken in $\{0,1\}$:

$$x^{s_j} + a_{1,j} x^{s_j - 1} + a_{2,j} x^{s_j - 2} + \dots + a_{s_j - 1,j} x + 1 \tag{3.3}$$

- The terms $m_{k,j}$ are obtained recursively, with the initial values $m_{1,j}, m_{2,j}, \ldots, m_{k-s_j,j}$ chosen freely, under the condition that $m_{k,j}, 1 \leq k \leq s_j$ is odd and less than 2^k.

A more complete treatment of *low discrepancy* sequences can be found in [Nie92]. And an example with $s_j = 3$, $a_{1,j} = 0$, $a_{2,j} = 1$ is given in [JK08].

3.2.2 Applying RVFL networks to multivariate time series forecasting

We consider $p \in \mathbb{N}^*$ time series $(X_t^{(j)})_{t\geq 0}, j = 1, \ldots, p$, observed at $n \in \mathbb{N}^*$ discrete dates. We are interested in obtaining simultaneous forecasts of the p time series at time $n+h$, $h \in \mathbb{N}^*$, by allowing each of the p variables to be influenced by the others (in the spirit of VAR models, see [Lüt05]).

For this purpose, we use $k < n$ lags of each of the observed p time series. The output variables to be explained are:

$$Y^{(j)} = \left(X_n^{(j)}, \ldots, X_{k+1}^{(j)}\right)^T \tag{3.4}$$

for $j \in \{1, \ldots, p\}$. Where $X_n^{(j)}$ is the most contemporaneous observed value of the j^{th} time series, and $X_{k+1}^{(j)}$ was observed k dates earlier in time for $(X_t^{(j)})_{t\geq 0}$. These output variables are stored in a matrix:

$$\mathbf{Y} \in \mathbb{R}^{(n-k)\times p}$$

and the predictors are stored in a matrix:

$$\mathbf{X} \in \mathbb{R}^{(n-k)\times(k\times p)}$$

where $\mathbf{X}$ consists in p blocks of k lags, for each one of the observed p time series. For example, the j_0^{th} block of $\mathbf{X}$, for $j_0 \in \{1, \ldots, p\}$ contains in columns:

$$\left(X_{n-i}^{(j_0)}, \ldots, X_{k+1-i}^{(j_0)}\right)^T \tag{3.5}$$

with $i \in \{1, \ldots, k\}$. It is also possible to add other regressors, such as dummy variables, indicators of special events, but for clarity, we consider only the inclusion of lags.

As described in the previous section, an additional layer of transformed predictors is added to $\mathbf{X}$, in order to capture the potentially non-linear interactions between the predictors and the output variable. Adding the transformed predictors to the original ones, leads to a new matrix of predictors with dimensions $(n-k)\times(k\times p+L)$, where L is the number of nodes in the hidden layer. We are then looking for simultaneous predictions

$$\hat{X}^{(j)}_{n+h|n,\ldots,1} =: \hat{X}^{(j)}_{n+h}$$

for $h \in \mathbb{N}^*$, and $j \in \{1,\ldots,p\}$. This, is a *multi-task learning* problem (see [Car98]), in which the output variables will all share the same set of predictors.

For example, we have $p = 2$ time series $(X^{(1)}_{t_1},\ldots,X^{(1)}_{t_5})$ and $(X^{(2)}_{t_1},\ldots,X^{(2)}_{t_5})$ observed at $n = 5$ dates $t_1 < \ldots < t_5$, with $k = 2$ lags, and $L = 3$ nodes in the hidden layer. In this case, the response variables are stored in:

$$\mathbf{Y} = \begin{pmatrix} X^{(1)}_{t_5} & X^{(2)}_{t_5} \\ X^{(1)}_{t_4} & X^{(2)}_{t_4} \\ X^{(1)}_{t_3} & X^{(2)}_{t_3} \end{pmatrix}$$

The predictors are stored in:

$$\mathbf{X} = \begin{pmatrix} X^{(1)}_{t_4} & X^{(1)}_{t_3} & X^{(2)}_{t_4} & X^{(2)}_{t_3} \\ X^{(1)}_{t_3} & X^{(1)}_{t_2} & X^{(2)}_{t_3} & X^{(2)}_{t_2} \\ X^{(1)}_{t_2} & X^{(1)}_{t_1} & X^{(2)}_{t_2} & X^{(2)}_{t_1} \end{pmatrix}$$

And the coefficients in the hidden layer are:

$$\mathbf{W} = \begin{pmatrix} W^{(1,1)} & W^{(1,2)} & W^{(1,3)} \\ W^{(2,1)} & W^{(2,2)} & W^{(2,3)} \\ W^{(3,1)} & W^{(3,2)} & W^{(3,3)} \\ W^{(4,1)} & W^{(4,2)} & W^{(4,3)} \end{pmatrix}$$

3.2.3 Solving for $\hat{\beta}$'s and $\hat{\gamma}$'s

We let y be the j_0^{th} column (out of p) of the response matrix $\mathbf{Y}$, and $\Phi(\mathbf{X})$ be the matrix of transformed predictors obtained from $\mathbf{X}$ by the hidden layer described at the beginning of section 3.2.1. We also denote the set of regression parameters associated with this j_0^{th} time series, as:

$$\beta^{(j_0)}_m =: \beta_m$$

and

$$\gamma^{(j_0)}_l =: \gamma_l$$

for $m \in \{1, \ldots, k\}$; $l \in \{1, \ldots, L\}$. Solving for the regression parameters for the j_0^{th} time series, under the constraints

$$\sum_{m=1}^{k \times p} \beta_m^2 \leq u$$

and

$$\sum_{l=1}^{L} \gamma_l^2 \leq v$$

for $u, v \in \mathbb{R}^*$, leads to minimizing a penalized residual sum of squares. Hence, for vectors $\beta \in \mathbb{R}^{(k \times p)}$ and $\gamma \in \mathbb{R}^L$ containing the regression parameters, we obtain the Lagrangian:

$$\mathcal{L}(\mathbf{X}; \beta, \gamma) = (y - \mathbf{X}\beta - \Phi(\mathbf{X})\gamma)^T (y - \mathbf{X}\beta - \Phi(\mathbf{X})\gamma) + \lambda_1 \beta^T \beta + \lambda_2 \gamma^T \gamma$$

where λ_1 and λ_2 are Lagrange multipliers. Taking the first derivatives of $\mathcal{L}$ relative to β and γ leads to:

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{X}; \beta, \gamma)}{\partial \beta} &= -y^T \mathbf{X} - \mathbf{X}^T y + 2(\mathbf{X}^T \mathbf{X})\beta + \mathbf{X}^T \Phi(\mathbf{X})\gamma + (\Phi(\mathbf{X})\gamma)^T \mathbf{X} + 2\lambda_1 \beta \\
&= 2(\mathbf{X}^T \mathbf{X} + \lambda_1 I_{k \times p})\beta - y^T \mathbf{X} - \mathbf{X}^T y + \mathbf{X}^T \Phi(\mathbf{X})\gamma + (\Phi(\mathbf{X})\gamma)^T \mathbf{X} \\
&= 2(\mathbf{X}^T \mathbf{X} + \lambda_1 I_{k \times p})\beta - 2\mathbf{X}^T y + 2\mathbf{X}^T \Phi(\mathbf{X})\gamma
\end{aligned}$$

where $I_{k \times p}$ is the identity matrix with dimensions $(k \times p) \times (k \times p)$ and equivalently

$$\frac{\partial \mathcal{L}(\mathbf{X}; \beta, \gamma)}{\partial \gamma} = 2(\Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \lambda_2 I_L)\gamma - 2\Phi(\mathbf{X})^T y + 2\Phi(\mathbf{X})^T \mathbf{X}\beta$$

where I_L is the identity matrix with dimensions $L \times L$. And setting these first derivatives to 0 leads to:

$$\begin{cases} (\mathbf{X}^T \mathbf{X} + \lambda_1 I_{k \times p})\beta + \mathbf{X}^T \Phi(\mathbf{X})\gamma = \mathbf{X}^T y \\ (\Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \lambda_2 I_L)\gamma + \Phi(\mathbf{X})^T \mathbf{X}\beta = \Phi(\mathbf{X})^T y \end{cases}$$

That is:

$$\begin{pmatrix} \mathbf{X}^T \mathbf{X} + \lambda_1 I_{k \times p} & \mathbf{X}^T \Phi(\mathbf{X}) \\ \Phi(\mathbf{X})^T \mathbf{X} & \Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \lambda_2 I_L \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{X}^T y \\ \Phi(\mathbf{X})^T y \end{pmatrix}$$

Now, if we denote:

$$A = \begin{pmatrix} \mathbf{X}^T\mathbf{X} + \lambda_1 I_{k\times p} & \mathbf{X}^T\Phi(\mathbf{X}) \\ \Phi(\mathbf{X})^T\mathbf{X} & \Phi(\mathbf{X})^T\Phi(\mathbf{X}) + \lambda_2 I_L \end{pmatrix} =: \begin{pmatrix} B & C^T \\ C & D \end{pmatrix}$$

and $S = D - CB^+C^T$. Then, using the algorithm described in [Cor09] for blockwise matrix inversion, we obtain:

$$A^+ = \begin{pmatrix} B^+ + B^+C^TS^+CB^+ & -B^+C^TS^+ \\ -S^+CB^+ & S^+ \end{pmatrix} =: \begin{pmatrix} A_1^+ & A_2^+ \\ A_3^+ & A_4^+ \end{pmatrix}$$

where S^+ and B^+ are the Moore-Penrose pseudo-inverse ([Pen55]) of matrixes S and B. Hence for each column y of $\mathbf{Y}$, we have the solutions:

$$\begin{pmatrix} \hat{\beta} \\ \hat{\gamma} \end{pmatrix} = \begin{pmatrix} A_1^+ & A_2^+ \\ A_3^+ & A_4^+ \end{pmatrix} \begin{pmatrix} \mathbf{X}^T y \\ \Phi(\mathbf{X})^T y \end{pmatrix}$$

And the whole set of parameters, for all the p observed time series is given by:

$$\begin{pmatrix} \underline{\hat{\beta}} \\ \underline{\hat{\gamma}} \end{pmatrix} := \begin{pmatrix} A_1^+ & A_2^+ \\ A_3^+ & A_4^+ \end{pmatrix} \begin{pmatrix} \mathbf{X}^T\mathbf{Y} \\ \Phi(\mathbf{X})^T\mathbf{Y} \end{pmatrix}$$

The objective function to be minimized (the least squares) is convex, and so is the set of feasible solutions. The solutions $\underline{\hat{\beta}}$ and $\underline{\hat{\gamma}}$ found here, are hence global minima.

3.2.4 h-steps ahead forecasts and use of dynamic regression

Having obtained the optimal set of parameters $\underline{\hat{\beta}}$ and $\underline{\hat{\gamma}}$ as described in the previous section, a new set of predictors is constructed by using the former output variables contained in response matrix $\mathbf{Y}$'s columns. The first k elements of each one of the p columns of $\mathbf{Y}$, which are the most contemporaneous values of the p series, constitute the new predictors.

Hence, if we denote the new predictors as:

$$\mathbf{X}_n^* = \left(X_n^{(1)}, \ldots, X_{n-k+1}^{(1)}, \ldots, \ldots, X_n^{(p)}, \ldots, X_{n-k+1}^{(p)}\right) \tag{3.6}$$

The 1-step ahead forecasts are obtained as:

$$\left(\hat{X}_{n+1}^{(1)}, \ldots, \hat{X}_{n+1}^{(p)}\right) = (\mathbf{X}_n^* \quad \Phi(\mathbf{X}_n^*)) \begin{pmatrix} \hat{\beta} \\ \hat{\gamma} \end{pmatrix}$$

The h-step ahead forecasts are obtained in a similar fashion; with the new forecasts $\left(\hat{X}_{n+1}^{(1)}, \ldots, \hat{X}_{n+1}^{(p)}\right)$ being added to the set of most contemporaneous values of the p series, and used as part of the new predictors.

In order to obtain confidence intervals around the point forecasts, we fit an ARIMA model to the in-sample residuals ϵ_i of each one of the p time series, as in dynamic regression models (see [Pan12]). An illustration can be found in the next section. Other models for the autocorrelated residuals could be envisaged, though.

3.3 Numerical examples

3.3.1 A Dynamic Nelson-Siegel example

The following examples are not exhaustive benchmarks, but aim at illustrating the forecasting capabilities of the model described in this chapter. All the results on RVFL use the ReLU activation function. We use calibrated discount rates data from Deutsche Bundesbank website, observed on a monthly basis, from the beginning of 2002 to the end 2015. There are 167 curves, observed at 50 maturities in the dataset. We obtain curves' forecasts in a Dynamic Nelson Siegel [NS87] framework (DNS), in the spirit of [DL06] and other similar models [1].

In figure B.2, we present the data that we use, and table B.1 contains a summary of these data; the minimum, maximum, median, first and third quartiles of the discount rates observed at given maturities. There are alternate cycles of increases and decreases of the discount rates, with generally a decreasing trend. Some of the discount rates, at the most recent dates, and lower maturities, are negative.

In the DNS framework, the spot interest rates observed at time t, for time to maturity τ are modeled as:

$$R_t(\tau) = \alpha_{1,t} + \alpha_{2,t}\left(\frac{1 - e^{-\tau/\lambda}}{e^{-\tau/\lambda}}\right) + \alpha_{3,t}\left(\frac{1 - e^{-\tau/\lambda}}{e^{-\tau/\lambda}} - e^{-\tau/\lambda}\right) \tag{3.7}$$

The factor loadings 1, $\left(\frac{1-e^{-T/\lambda}}{e^{-T/\lambda}}\right)$ and $\left(\frac{1-e^{-T/\lambda}}{e^{-T/\lambda}} - e^{-T/\lambda}\right)$ are used to represent the level, slope, and curvature of the Yield Curve. We obtain estimations of

[1] [DR13]: "*there are by now literally hundreds of DNS applications involving model fitting and forecasting*"

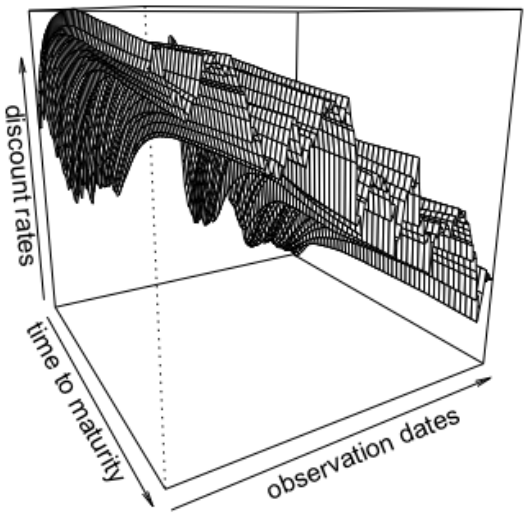

Fig. 3.2.: Observed discount rates from Deutsche Bundesbank website, from 2002 to the end 2015

$\alpha_{i,t}, i = 1, \ldots, 3$ for each cross-section of yields by fixing λ, and doing a least squares regression on the factor loadings. The three time series $\alpha_{i,t}, i = 1, \ldots, 3$ associated to the loadings for each cross-section of yields, are those that we wish to forecast simultaneously, by using an RVFL model.

This type of model (DNS) cannot be used for no-arbitrage pricing as is, but it could be useful for example, for *stressing* the yield curve factors under the historical probability. It can however be made arbitrage-free, if necessary (see [DR13]). We will benchmark the RVFL model applied to the three time series $\alpha_{i,t}, i = 1, \ldots, 3$,

Tab. 3.1.: Summary of observed discount rates from Deutsche Bundesbank website, from 2002 to the end 2015

Maturity	Min	1st Qrt	Median	3rd Qrt	Max
1	-0.116	0.858	2.045	3.072	5.356
5	0.170	1.327	2.863	3.807	5.146
15	0.711	2.616	3.954	4.702	5.758
30	0.805	2.594	3.962	4.814	5.784
50	0.749	2.647	3.630	4.590	5.467

against ARIMA and VAR models. [DL06] applied an autoregressive AR(1) model separately to each one of the parameters, $\alpha_{i,t}, i = 1, \ldots, 3$.

We will apply to these parameters' series: an ARIMA model ([HK08]), and a Vector Autoregressive model (VAR, see [Pfa+08] and [Lüt05]); with the parameter λ of the DNS factor loadings, used as an *hyperparameter* for the time series cross-validation. In the RVFL and the VAR model, the number of lags is also used as an *hyperparameter* for the cross-validation. For the RVFL, the most recent values of $\alpha_{i,t}, i = 1, \ldots, 3$ are stored in matrix $\mathbf{Y}$, as described in section 3.2.3, ordered by *date of arrival*, whereas matrix $\mathbf{X}$ contains the lags of the three series.

A rolling forecasting methodology (see [Ber+15]) is implemented in order to obtain these benchmarks. A fixed 12 months-length window for training the model, and the following 12 months for testing, the origin of the training set is then advanced of 1 month, and the training/testing procedure is repeated. The measure of forecasting performance is the Root Mean Squared Error ($RMSE$).

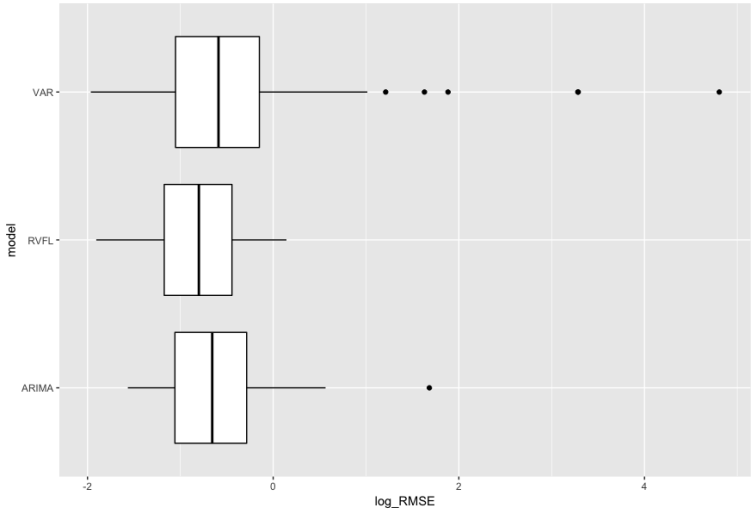

Fig. 3.3.: Distribution of out-of-sample $log(RMSE)$, for ARIMA, VAR, and RVFL

Figure 3.3 presents boxplots for the distribution of out-of-sample errors obtained in the cross-validation procedure, and figure 3.4 presents the 12 months-ahead out-of-sample errors over time. ARIMA (separate ([HK08]) ARIMA models applied to each series $\alpha_{i,t}, i = 1, \ldots, 3$) gives good results, as already suggested by [DL06]. They are nearly comparable to results from RVFL, but a bit more volatile, with an outlier point observed on the $log(RMSE)$ box plot.

The unrestricted VAR model results include more volatility than the two other methods on this specific example, especially in the period of financial crisis going from 2007 to 2009, as seen on figure 3.4. Table 3.2 is to be read in conjuction with the $log(RMSE)$ box plot presented in figure 3.3. It summarises the results

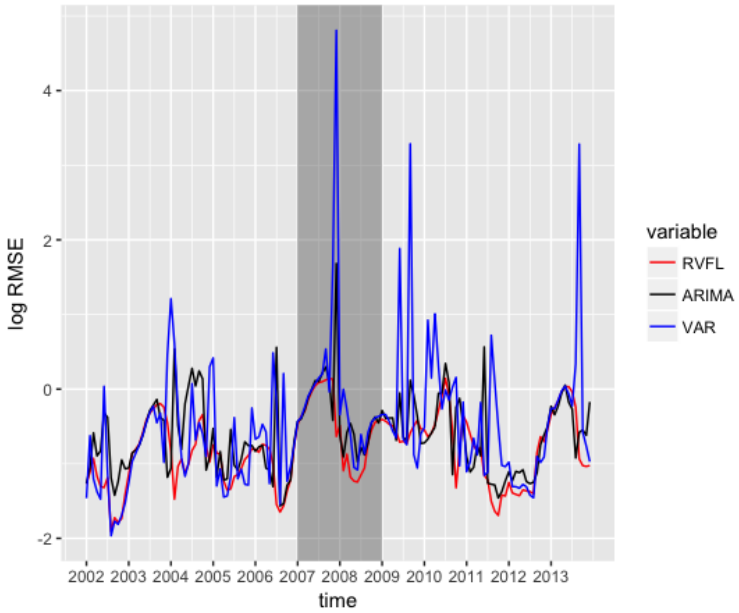

Fig. 3.4.: Out-of-sample $log(RMSE)$, for ARIMA, VAR, and RVFL over time

Tab. 3.2.: Comparison of 12 months ahead out-of-sample $RMSE$, for the ARIMA, RVFL, and VAR

Method	Min	1st Qrt	Median	Mean	3rd Qrt	Max
RVFL	0.1487	**0.3092**	**0.4491**	**0.5041**	**0.6414**	**1.1535**
ARIMA	0.2089	0.3470	0.5187	0.6358	0.7516	5.3798
VAR	**0.1402**	0.3493	0.5549	1.9522	0.8619	122.2214

obtained by the different methods on the out-of-sample $RMSE$. Table 3.3 contains 95% confidence intervals around the mean of the differences between the three methods.

Another advantage of RVFL over ARIMA or AR(1) in this context is that, it would be possible to add other variables to the RVFL regression, such as inflation, or dummy variables for external events, and combine their effects. It is also possible to stress one variable, and see the effects on the other variables, as presented in the appendix section 3.5.1: the parameter $\alpha_{1,t}$ is increased (from 0.75 to 1.25) and decreased

Tab. 3.3.: 95% confidence interval around the difference of out-of-sample $RMSE$

Method	Lower bound	Upper bound	Mean
RVFL - ARIMA	-0.2116	-0.0518	**-0.1317**
RVFL - VAR	-3.1888	0.2927	**-1.4480**
ARIMA - VAR	-2.9937	0.3610	**-1.3163**

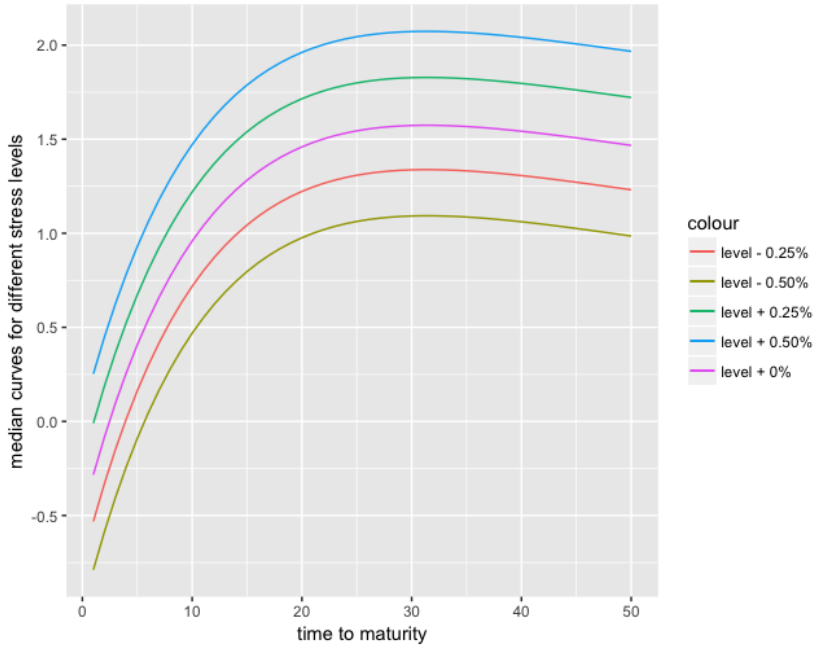

Fig. 3.5.: 12 months-ahead median curves, for stressed yield curve level $\alpha_{1,t}$

(from 0.75 to 0.25), and the other parameters $\alpha_{2,t}$ and $\alpha_{3,t}$ forecasts move slightly, consecutively to these *stresses*. The corresponding median forecast curves for these stresses, and some additional ones, are presented in figure 3.5.

3.3.2 Forecasting 1 year, 10 years and 20 years spot rates

For this second example, we forecast the 1-year, 10-years and 20-years spot rates time series from the previous dataset, on a 12-months horizon. As described in the previous section, we use a rolling forecasting methodology, with a training window of 12 months length.

Figure 3.6 presents the three time series of data, and a summary of the data can be found in tables 3.4 and 3.5.

Tab. 3.4.: Summary of the data for 1 year, 10 years and 20 years spot rates time series (in %)

Method	Min	1st Qrt	Median	Mean	3rd Qrt	Max
1y rate	-0.116	0.858	2.045	2.062	3.072	5.356
10y rate	0.560	2.221	3.581	3.322	4.354	5.570
20y rate	0.790	2.685	4.050	3.782	4.830	5.850

The three time series globally exhibit a decreasing trend, and are highly positively correlated. The spot rates for short-term maturities can also be negative, as it has

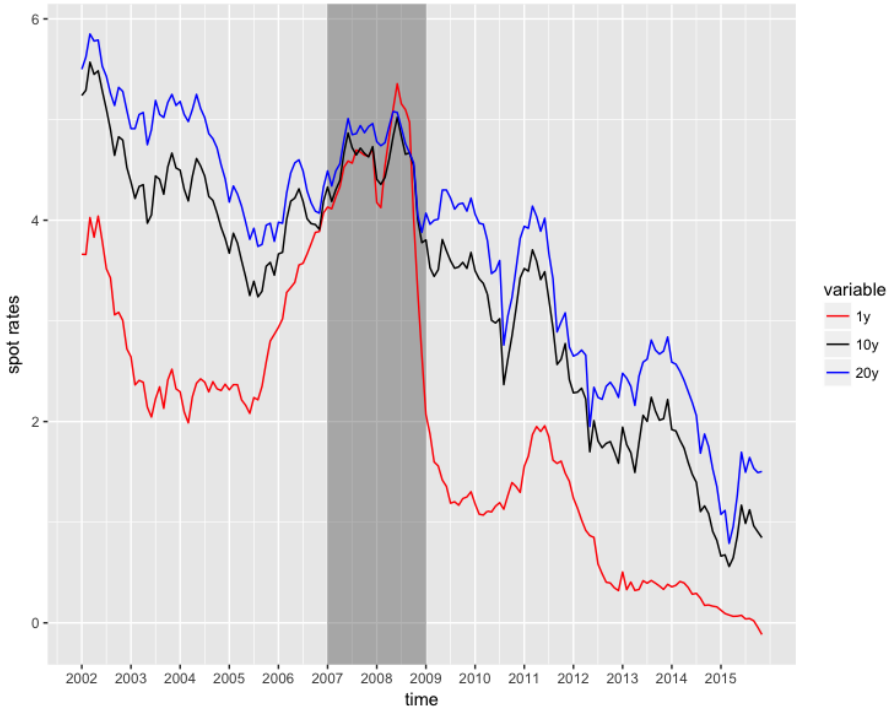

Fig. 3.6.: 1-year, 10-years and 20-years spot rates time series data

Tab. 3.5.: Summary of data for 1 year, 10 years and 20 years spot rates time series

Correlations	1y rate	10y rate	20y rate
1y rate	1.0000	0.8729	0.8118
10y rate	0.8729	1.0000	0.9900
20y rate	0.8118	0.9900	1.0000

been observed recently in 2016. The spreads between the spot rates time series are extremely narrow during the 2007-2009 crisis. The tables below contain the results of a comparison between the RVFL model and an unrestricted VAR model (with one lag, *best* parameter found) on the forecasting problem. The *best* RVFL model, with the lowest out-of-sample $RMSE$, uses one lag, four hidden nodes, and $\lambda_1 = 5.80$, $\lambda_2 = 19.66$.

Tab. 3.6.: Comparison of 12 months ahead out-of-sample $RMSE$, for the RVFL, and VAR

Method	Min	1st Qrt	Median	Mean	3rd Qrt	Max
RVFL	0.1675	**0.2906**	**0.4704**	**0.5452**	**0.6469**	**1.8410**
VAR	**0.1382**	0.4025	0.6469	1.0310	1.0750	13.020

Tab. 3.7.: 95% confidence interval around the difference of out-of-sample $RMSE$

Method	Lower bound	Upper bound	Mean
RVFL-VAR	-0.2622	-0.7087	**-0.4854**

3.3.3 Forecasting on a longer horizon, with a longer training window

In this third example, as in section 3.3.1, we apply the DNS framework to the forecasting of spot rates. But with a longer training set (36 months), and a longer horizon for the test set (36 months as well). We use interest rate swaps data from the Federal Reserve Bank of St Louis website [2], observed on a monthly basis from july 2000 to september 2016, with maturities equal to $1, 2, 3, 4, 5, 7, 10, 30$, and a tenor equal to three months.

On figure 3.7, we present three of the eight time series of swap rates, observed for time to maturities equal to $3, 10$ and 30. The swap rates for different maturities generally exhibit a decreasing trend, and are nearly equal to 0 by the end of 2016, for the shortest maturities.

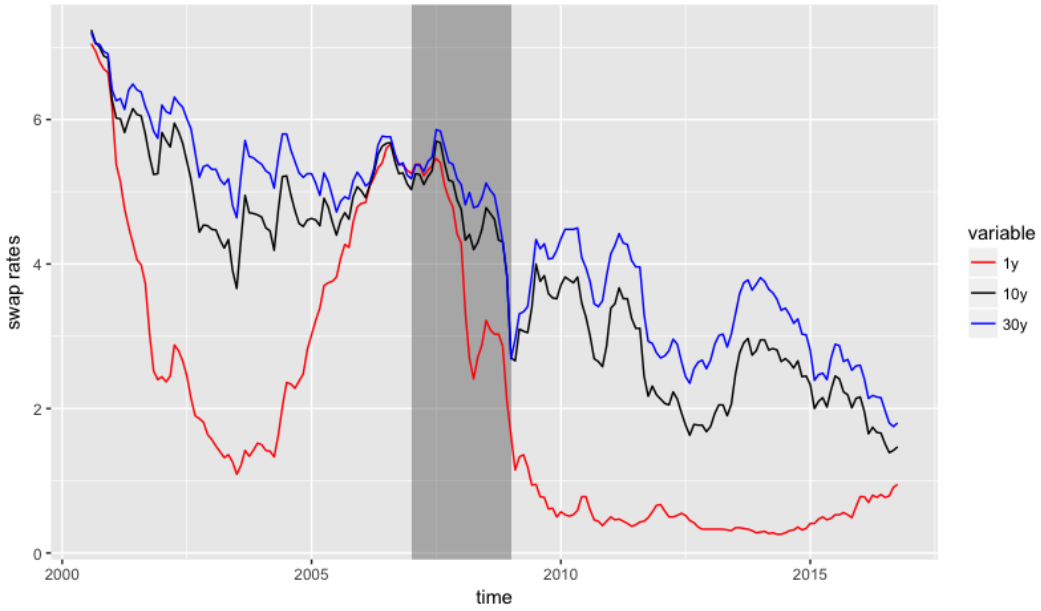

Fig. 3.7.: Swap rates data (in %) from St Louis Federal Reserve Bank, at maturities $1, 10, 30$

We also observe that the spreads between swap rates with different maturities start to narrow in 2006 until the end of 2007, and the swap rates for short term maturities are relatively high during the same period. This is the period corresponding to the

[2] Available at https://fred.stlouisfed.org/categories/32299

Tab. 3.8.: Descriptive statistics of St Louis Federal Reserve data for 1y, 10y and 30y swap rates (in %)

Maturity	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.
1	0.260	0.500	1.340	2.108	3.360	7.050
10	1.390	2.610	4.190	3.881	5.020	7.240
30	1.750	3.270	4.650	4.404	5.375	7.200

Liquidity and Credit Crunch 2007-2008. Table 3.8 presents the descriptive statistics for these three time series.

All the swap rates (for all the maturities available) were then transformed into zero rates, by using a single curve calibration methodology (that is, ignoring the counterparty credit risk) with linear interpolation between the swaps' maturities. Then, the Nelson & Siegel model was used for fitting and forecasting the curves in a DNS framework, with both `auto.arima` and the RVFL model presented in this chapter, applied to the three factors. In the fashion of section 3.3.1. But now, we obtain 36-months ahead forecasts, from a rolling training windows with a fixed 36 months length. The average out-of-sample $RMSE$ are then calculated for each method.

The *best* hyperparameters - associated with the lowest out-of-sample average $RMSE$ - for each model are obtained through a search on a grid of values. We have:

- DNS with ARIMA (`auto.arima`): $\lambda = 1.4271$ (Nelson Siegel parameter)
- DNS with RVFL: **number of lags** for each series: 1, **activation function**: ReLU, **number of nodes** in the hidden layer: 45, $\lambda_1 = 4.6416$, $\lambda_2 = 774.2637$ (RVFL parameters) and $\lambda = 24$ (Nelson Siegel parameter)

With these parameters, the results detailed in table 3.9 are obtained, for the out-of-sample $RMSE$. A 95% confidence interval around the difference of out-of-sample $RMSE$ between ARIMA (applied to each one of the three factors) and RVFL is presented in table 3.10.

Tab. 3.9.: Descriptive statistics for out-of-sample $RMSE$, with rolling training window = 36 months, and testing window = 36 months

Method	Min.	1st Qrt	Median	Mean	3rd Qrt	Max.	Std. Dev
ARIMA	0.0036	**0.0070**	**0.0104**	0.0149	0.0161	0.2150	0.0213
RVFL	**0.0032**	0.0078	0.0115	**0.0120**	**0.0148**	**0.0256**	**0.0055**

Tab. 3.10.: 95% confidence interval around the difference of out-of-sample $RMSE$

Method	Lower bound	Upper bound	Mean
RVFL-ARIMA	-0.0064	0.0007	**-0.0028**

Figure 3.8 presents the evolution of the out-of-sample $log(RMSE)$ over the training/testing windows. The grey rectangle indicating the Liquidity and Credit crunch is larger here, because in this example, a training set starting in 2004 has its test set starting 36 months later, in 2007. Again, we observe that the results from the RVFL model exhibit a low out-of-sample error, along with a low volatility.

Fig. 3.8.: Out-of-sample $log(RMSE)$, for ARIMA and RVFL over time

Figure 3.9, presents the convergence of the out-of-sample $log(RMSE)$ for the DNS + RVFL model from this section, as a function of $log(\lambda_1)$ and $log(\lambda_2)$. λ_1 and λ_2 both range from 10^{-2} to 10^4, with ten equally-spaced points each (hence, a grid of one hundred points $(log(\lambda_1), log(\lambda_2), log(RMSE))$.

The number of nodes in the hidden layer is equal to 45, and the value of λ, parameter from the [NS87] model presented in section 3.3.1, is fixed and equal to 24. The one hundred points $(log(\lambda_1), log(\lambda_2), log(RMSE))$ that we use for figure 3.9 can be found in appendix .0.5.

There is a rectangular region at the top, in the middle of the figure, where the $log(RMSE)$ is the lowest. In this region, the lowest value of the out-of-sample

$log(RMSE)$ is observed for $\lambda_1 = 4.6416$ and $\lambda_2 = 464.1589$ and the out-of-sample $RMSE$ is equal to 0.01206 (75^{th} point in appendix .0.5).

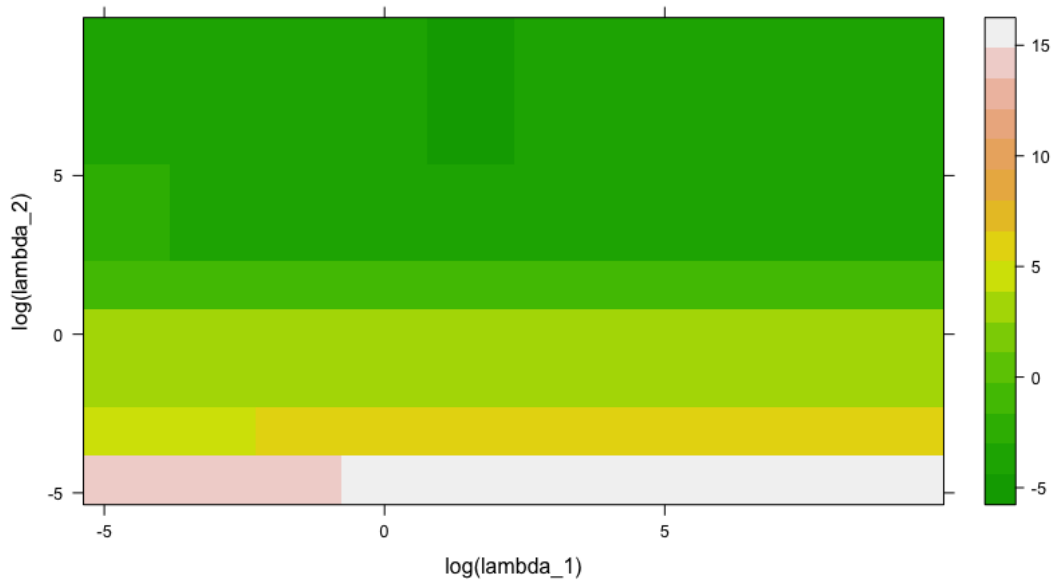

Fig. 3.9.: Out-of-sample $log(RMSE)$, as a function of λ_1 and λ_2

3.4 Conclusion

We present a model which could be used for multiple time series forecasting, based on a single layer quasi-randomized neural network. In this model, the lags of the different time series are used as in a dynamic regression model, and include the response variable lags. An additional layer of variables is added to the regression, whose nodes are not trained but obtained from a low discrepancy sequence. It is possible to add new variables to the regression, as indicators of special events, or to stress one variable, and observe the implied effect on the others' forecast. The model is tested on raw historical spot rates, and in a Dynamic Nelson Siegel framework. It produces *robust* forecast results when compared to other usual (unpenalized) models in the same framework.

3.5 Appendix

3.5.1 Mean forecast and confidence intervals for $\alpha_{i,t}, i = 1, \ldots, 3$ forecasts

$\alpha_{1,t}$:

	alpha1	y_lo80	y_hi80	y_lo95	y_hi95
13	0.7432724	0.6852024	0.8013425	0.6544620	0.8320829
14	0.7357374	0.6776673	0.7938074	0.6469269	0.8245478
15	0.7378042	0.6797342	0.7958742	0.6489938	0.8266147
16	0.7408417	0.6827717	0.7989118	0.6520313	0.8296522
17	0.7407904	0.6827204	0.7988604	0.6519800	0.8296009
18	0.7404501	0.6823801	0.7985201	0.6516396	0.8292605
19	0.7403603	0.6822903	0.7984303	0.6515498	0.8291707
20	0.7403981	0.6823281	0.7984681	0.6515876	0.8292085
21	0.7404788	0.6824087	0.7985488	0.6516683	0.8292892
22	0.7404786	0.6824086	0.7985487	0.6516682	0.8292891
23	0.7404791	0.6824091	0.7985491	0.6516686	0.8292895
24	0.7404758	0.6824058	0.7985458	0.6516654	0.8292863

$\alpha_{2,t}$:

	alpha2	y_lo80	y_hi80	y_lo95	y_hi95
13	-1.250640	-1.351785	-1.149495	-1.405328	-1.095952
14	-1.243294	-1.344439	-1.142149	-1.397982	-1.088606
15	-1.241429	-1.342574	-1.140284	-1.396117	-1.086741
16	-1.243868	-1.345014	-1.142723	-1.398557	-1.089180
17	-1.244483	-1.345628	-1.143338	-1.399171	-1.089795
18	-1.241865	-1.343010	-1.140719	-1.396553	-1.087177
19	-1.240814	-1.341959	-1.139669	-1.395502	-1.086126
20	-1.240371	-1.341516	-1.139226	-1.395059	-1.085683
21	-1.240237	-1.341382	-1.139092	-1.394925	-1.085549
22	-1.240276	-1.341421	-1.139131	-1.394964	-1.085588
23	-1.240329	-1.341474	-1.139184	-1.395017	-1.085641
24	-1.240308	-1.341453	-1.139163	-1.394996	-1.085620

$\alpha_{3,t}$:

	alpha3	y_lo80	y_hi80	y_lo95	y_hi95
13	4.584836	4.328843	4.757406	4.215410	4.870840
14	4.546167	4.307253	4.849862	4.163634	4.993482
15	4.527651	4.201991	4.803004	4.042913	4.962082
16	4.513810	4.216525	4.850160	4.048812	5.017874
17	4.517643	4.176214	4.828735	4.003502	5.001446
18	4.523109	4.203064	4.866713	4.027406	5.042371
19	4.522772	4.178489	4.848760	4.001079	5.026170
20	4.521846	4.191833	4.866066	4.013374	5.044524
21	4.521382	4.177560	4.854171	3.998472	5.033259
22	4.521451	4.186714	4.864755	4.007248	5.044222
23	4.521772	4.178995	4.857897	3.999300	5.037591
24	4.521862	4.184734	4.864155	4.004903	5.043987

3.5.2 Stressed forecast ($\alpha_{1,t} + 0.5\%$) and confidence intervals for $\alpha_{i,t}, i = 1, \ldots, 3$ forecasts

$\alpha_{1,t}$:

	alpha1	y_lo80	y_hi80	y_lo95	y_hi95
13	1.25	1.19193	1.30807	1.16119	1.33881
14	1.25	1.19193	1.30807	1.16119	1.33881
15	1.25	1.19193	1.30807	1.16119	1.33881
16	1.25	1.19193	1.30807	1.16119	1.33881
17	1.25	1.19193	1.30807	1.16119	1.33881
18	1.25	1.19193	1.30807	1.16119	1.33881
19	1.25	1.19193	1.30807	1.16119	1.33881
20	1.25	1.19193	1.30807	1.16119	1.33881
21	1.25	1.19193	1.30807	1.16119	1.33881
22	1.25	1.19193	1.30807	1.16119	1.33881
23	1.25	1.19193	1.30807	1.16119	1.33881
24	1.25	1.19193	1.30807	1.16119	1.33881

$\alpha_{2,t}$:

	alpha2	y_lo80	y_hi80	y_lo95	y_hi95
13	-1.222568	-1.323713	-1.121423	-1.377256	-1.067880
14	-1.219401	-1.320546	-1.118256	-1.374089	-1.064713
15	-1.211361	-1.312506	-1.110216	-1.366049	-1.056673

```
16 -1.216213 -1.317358 -1.115068 -1.370901 -1.061525
17 -1.215266 -1.316411 -1.114121 -1.369954 -1.060578
18 -1.211474 -1.312619 -1.110329 -1.366162 -1.056786
19 -1.210329 -1.311474 -1.109183 -1.365017 -1.055640
20 -1.209610 -1.310755 -1.108465 -1.364298 -1.054922
21 -1.209648 -1.310793 -1.108503 -1.364336 -1.054960
22 -1.209601 -1.310746 -1.108456 -1.364289 -1.054913
23 -1.209688 -1.310833 -1.108542 -1.364376 -1.055000
24 -1.209653 -1.310798 -1.108508 -1.364341 -1.054965
```

$\alpha_{3,t}$:

```
alpha3   y_lo80   y_hi80   y_lo95   y_hi95
13 4.500390 4.244398 4.672961 4.130964 4.786394
14 4.482948 4.244035 4.786643 4.100415 4.930263
15 4.441841 4.116182 4.717194 3.957103 4.876272
16 4.441744 4.144459 4.778094 3.976746 4.945808
17 4.439609 4.098181 4.750701 3.925469 4.923413
18 4.447151 4.127105 4.790755 3.951448 4.966412
19 4.446164 4.101881 4.772152 3.924471 4.949562
20 4.445421 4.115407 4.789641 3.936949 4.968099
21 4.445411 4.101589 4.778200 3.922501 4.957289
22 4.445491 4.110754 4.788795 3.931287 4.968262
23 4.445937 4.103160 4.782062 3.923465 4.961756
24 4.446012 4.108885 4.788306 3.929053 4.968137
```

.0.3 Mean forecast and confidence intervals for $\alpha_{i,t}, i = 1, \ldots, 3$ forecasts

$\alpha_{1,t}$:

	alpha1	y_lo80	y_hi80	y_lo95	y_hi95
13	0.7432724	0.6852024	0.8013425	0.6544620	0.8320829
14	0.7357374	0.6776673	0.7938074	0.6469269	0.8245478
15	0.7378042	0.6797342	0.7958742	0.6489938	0.8266147
16	0.7408417	0.6827717	0.7989118	0.6520313	0.8296522
17	0.7407904	0.6827204	0.7988604	0.6519800	0.8296009
18	0.7404501	0.6823801	0.7985201	0.6516396	0.8292605
19	0.7403603	0.6822903	0.7984303	0.6515498	0.8291707
20	0.7403981	0.6823281	0.7984681	0.6515876	0.8292085
21	0.7404788	0.6824087	0.7985488	0.6516683	0.8292892
22	0.7404786	0.6824086	0.7985487	0.6516682	0.8292891
23	0.7404791	0.6824091	0.7985491	0.6516686	0.8292895
24	0.7404758	0.6824058	0.7985458	0.6516654	0.8292863

$\alpha_{2,t}$:

	alpha2	y_lo80	y_hi80	y_lo95	y_hi95
13	-1.250640	-1.351785	-1.149495	-1.405328	-1.095952
14	-1.243294	-1.344439	-1.142149	-1.397982	-1.088606
15	-1.241429	-1.342574	-1.140284	-1.396117	-1.086741
16	-1.243868	-1.345014	-1.142723	-1.398557	-1.089180
17	-1.244483	-1.345628	-1.143338	-1.399171	-1.089795
18	-1.241865	-1.343010	-1.140719	-1.396553	-1.087177
19	-1.240814	-1.341959	-1.139669	-1.395502	-1.086126
20	-1.240371	-1.341516	-1.139226	-1.395059	-1.085683
21	-1.240237	-1.341382	-1.139092	-1.394925	-1.085549
22	-1.240276	-1.341421	-1.139131	-1.394964	-1.085588
23	-1.240329	-1.341474	-1.139184	-1.395017	-1.085641
24	-1.240308	-1.341453	-1.139163	-1.394996	-1.085620

$\alpha_{3,t}$:

	alpha3	y_lo80	y_hi80	y_lo95	y_hi95
13	4.584836	4.328843	4.757406	4.215410	4.870840

```
14 4.546167 4.307253 4.849862 4.163634 4.993482
15 4.527651 4.201991 4.803004 4.042913 4.962082
16 4.513810 4.216525 4.850160 4.048812 5.017874
17 4.517643 4.176214 4.828735 4.003502 5.001446
18 4.523109 4.203064 4.866713 4.027406 5.042371
19 4.522772 4.178489 4.848760 4.001079 5.026170
20 4.521846 4.191833 4.866066 4.013374 5.044524
21 4.521382 4.177560 4.854171 3.998472 5.033259
22 4.521451 4.186714 4.864755 4.007248 5.044222
23 4.521772 4.178995 4.857897 3.999300 5.037591
24 4.521862 4.184734 4.864155 4.004903 5.043987
```

.0.4 Stressed forecast ($\alpha_{1,t} + 0.5\%$) and confidence intervals for $\alpha_{i,t}, i = 1, \ldots, 3$ forecasts

$\alpha_{1,t}$:

```
 alpha1  y_lo80  y_hi80  y_lo95  y_hi95
13   1.25 1.19193 1.30807 1.16119 1.33881
14   1.25 1.19193 1.30807 1.16119 1.33881
15   1.25 1.19193 1.30807 1.16119 1.33881
16   1.25 1.19193 1.30807 1.16119 1.33881
17   1.25 1.19193 1.30807 1.16119 1.33881
18   1.25 1.19193 1.30807 1.16119 1.33881
19   1.25 1.19193 1.30807 1.16119 1.33881
20   1.25 1.19193 1.30807 1.16119 1.33881
21   1.25 1.19193 1.30807 1.16119 1.33881
22   1.25 1.19193 1.30807 1.16119 1.33881
23   1.25 1.19193 1.30807 1.16119 1.33881
24   1.25 1.19193 1.30807 1.16119 1.33881
```

$\alpha_{2,t}$:

```
 alpha2    y_lo80    y_hi80    y_lo95    y_hi95
13 -1.222568 -1.323713 -1.121423 -1.377256 -1.067880
14 -1.219401 -1.320546 -1.118256 -1.374089 -1.064713
15 -1.211361 -1.312506 -1.110216 -1.366049 -1.056673
16 -1.216213 -1.317358 -1.115068 -1.370901 -1.061525
17 -1.215266 -1.316411 -1.114121 -1.369954 -1.060578
```

```
18 -1.211474 -1.312619 -1.110329 -1.366162 -1.056786
19 -1.210329 -1.311474 -1.109183 -1.365017 -1.055640
20 -1.209610 -1.310755 -1.108465 -1.364298 -1.054922
21 -1.209648 -1.310793 -1.108503 -1.364336 -1.054960
22 -1.209601 -1.310746 -1.108456 -1.364289 -1.054913
23 -1.209688 -1.310833 -1.108542 -1.364376 -1.055000
24 -1.209653 -1.310798 -1.108508 -1.364341 -1.054965
```

$\alpha_{3,t}$:

```
alpha3   y_lo80   y_hi80   y_lo95   y_hi95
13 4.500390 4.244398 4.672961 4.130964 4.786394
14 4.482948 4.244035 4.786643 4.100415 4.930263
15 4.441841 4.116182 4.717194 3.957103 4.876272
16 4.441744 4.144459 4.778094 3.976746 4.945808
17 4.439609 4.098181 4.750701 3.925469 4.923413
18 4.447151 4.127105 4.790755 3.951448 4.966412
19 4.446164 4.101881 4.772152 3.924471 4.949562
20 4.445421 4.115407 4.789641 3.936949 4.968099
21 4.445411 4.101589 4.778200 3.922501 4.957289
22 4.445491 4.110754 4.788795 3.931287 4.968262
23 4.445937 4.103160 4.782062 3.923465 4.961756
24 4.446012 4.108885 4.788306 3.929053 4.968137
```

.0.5 Out-of-sample $log(RMSE)$ as a function of $log(\lambda_1)$ and $log(\lambda_2)$

```
    log_lambda1 log_lambda2  log_error
1     -4.605170   -4.605170 13.5039336
2     -3.070113   -4.605170 14.3538621
3     -1.535057   -4.605170 14.7912525
4      0.000000   -4.605170 14.8749375
5      1.535057   -4.605170 14.8928025
6      3.070113   -4.605170 14.8962203
7      4.605170   -4.605170 14.8975776
8      6.140227   -4.605170 14.8976494
9      7.675284   -4.605170 14.8976630
10     9.210340   -4.605170 14.8976715
11    -4.605170   -3.070113  3.9701229
12    -3.070113   -3.070113  4.9644144
```

13	-1.535057	-3.070113	5.2607788
14	0.000000	-3.070113	5.3208833
15	1.535057	-3.070113	5.3326856
16	3.070113	-3.070113	5.3351784
17	4.605170	-3.070113	5.3357136
18	6.140227	-3.070113	5.3358306
19	7.675284	-3.070113	5.3358557
20	9.210340	-3.070113	5.3358610
21	-4.605170	-1.535057	3.6692928
22	-3.070113	-1.535057	3.5643607
23	-1.535057	-1.535057	3.5063072
24	0.000000	-1.535057	3.4942438
25	1.535057	-1.535057	3.4904354
26	3.070113	-1.535057	3.4881352
27	4.605170	-1.535057	3.4888202
28	6.140227	-1.535057	3.4880668
29	7.675284	-1.535057	3.4886911
30	9.210340	-1.535057	3.4886383
31	-4.605170	0.000000	3.6224691
32	-3.070113	0.000000	3.8313407
33	-1.535057	0.000000	3.8518759
34	0.000000	0.000000	3.8163287
35	1.535057	0.000000	3.7993471
36	3.070113	0.000000	3.7948073
37	4.605170	0.000000	3.7937787
38	6.140227	0.000000	3.7935546
39	7.675284	0.000000	3.7935063
40	9.210340	0.000000	3.7934958
41	-4.605170	1.535057	-1.2115597
42	-3.070113	1.535057	-1.0130537
43	-1.535057	1.535057	-0.5841784
44	0.000000	1.535057	-0.3802817
45	1.535057	1.535057	-0.3109827
46	3.070113	1.535057	-0.2931830
47	4.605170	1.535057	-0.2891586
48	6.140227	1.535057	-0.2882824
49	7.675284	1.535057	-0.2880932
50	9.210340	1.535057	-0.2880524
51	-4.605170	3.070113	-2.0397856
52	-3.070113	3.070113	-3.1729003
53	-1.535057	3.070113	-3.8655596
54	0.000000	3.070113	-3.9279840

```
55     1.535057    3.070113 -3.9508440
56     3.070113    3.070113 -4.0270316
57     4.605170    3.070113 -4.0831569
58     6.140227    3.070113 -4.0953167
59     7.675284    3.070113 -4.0979447
60     9.210340    3.070113 -4.0985113
61    -4.605170    4.605170 -2.6779260
62    -3.070113    4.605170 -3.4704808
63    -1.535057    4.605170 -4.0404935
64     0.000000    4.605170 -4.2441836
65     1.535057    4.605170 -4.3657802
66     3.070113    4.605170 -4.3923094
67     4.605170    4.605170 -4.3862826
68     6.140227    4.605170 -4.3830293
69     7.675284    4.605170 -4.3820703
70     9.210340    4.605170 -4.3818486
71    -4.605170    6.140227 -3.5007058
72    -3.070113    6.140227 -3.8389274
73    -1.535057    6.140227 -4.1969545
74     0.000000    6.140227 -4.3400025
75     1.535057    6.140227 -4.4179718
76     3.070113    6.140227 -4.3797034
77     4.605170    6.140227 -4.3073100
78     6.140227    6.140227 -4.2866647
79     7.675284    6.140227 -4.2820357
80     9.210340    6.140227 -4.2810151
81    -4.605170    7.675284 -3.7236774
82    -3.070113    7.675284 -4.0057353
83    -1.535057    7.675284 -4.2699684
84     0.000000    7.675284 -4.3569254
85     1.535057    7.675284 -4.4156364
86     3.070113    7.675284 -4.3842360
87     4.605170    7.675284 -4.2759516
88     6.140227    7.675284 -4.2425876
89     7.675284    7.675284 -4.2346514
90     9.210340    7.675284 -4.2328863
91    -4.605170    9.210340 -3.7706268
92    -3.070113    9.210340 -4.0406387
93    -1.535057    9.210340 -4.2776907
94     0.000000    9.210340 -4.3498230
95     1.535057    9.210340 -4.4095778
96     3.070113    9.210340 -4.3860089
```

```
97      4.605170    9.210340 -4.2647179
98      6.140227    9.210340 -4.2273624
99      7.675284    9.210340 -4.2183911
100     9.210340    9.210340 -4.2163638
```

A

Multiple time series forecasting using ensembles of quasi-randomized functional link neural networks

A.1 Introduction

The goal of ensemble learning is to combine two or more individual statistical/machine learning models - the base models or base learners - into one, in order to obtain an ensemble model, with an improved out-of-sample error over the base models.

Ensemble learning is widely used in the winning solutions of machine learning competitions. It allows to achieve very good performances indeed, at the expense of being relatively less interpretable than the base learners. As a consequence, choosing to use ensemble models or a base model for solving a given problem, could sometimes be seen as finding a trade-off between the desire for interpretability and the desire for an highly increased performance. That said, some techniques such as variable importance for ensemble of trees, can give a sense of which predictor contributes the most to the perfomance of the model.

In this chapter, we apply three popular ensemble learning methods to multiple time series forecasting: Bootstrap aggregating ([Bre96a]), known as bagging, Boosting ([Fri01], [BY03], [Hot+10]), and stacked generalization([Wol92]), known as stacking. The base learners that we use for bagging and stacking, are the quasi-random vector functional link neural networks introduced in [Mou+18]. For the boosting algorithm, we use a slightly modified version of the model from [Mou+18], in which the regression model parameters are not constrained in a ridge regression fashion (more details in section A.2).

In the next section, we give an overview of the base learners. Then, we describe how we use these models as the basic components of our bagged/stacked ensembles. To finish, we present some numerical examples of use of these new models for forecasting multiple time series.

A.2 Description of the base models

The base learner is described in lengths in [Mou+18]. It is a single layer feed forward neural networks (SLFN). We have an output variable $y \in \mathbb{R}^n$, which has to be explained by a set of observed predictors $X^{(j)} \in \mathbb{R}^n$, $j \in \{1, \ldots, p\}$. The RVFL networks that we use to explain y is described for $i \in \{1, \ldots, n\}$ as:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_i^{(j)} + \sum_{l=1}^{L} \gamma_l \, g\left(\sum_{j=1}^{p} W^{(j,l)} X_i^{(j)}\right) + \epsilon_i \tag{A.1}$$

Where g is the *activation function*, L the number of nodes in the hidden layer, $W^{(j,l)}$ are elements of the hidden layer, and the parameters β_j and γ_l are to be learned from the observed data $X^{(j)}$, $j \in \{1, \ldots, p\}$. The ϵ_i's are the residual differences between the output variable values and the RVFL model.

This type of model can be seen as a one explaining y_i, by finding a compromise between linear and potentially non-linear effects of the original predictors $X^{(j)}$ and transformed predictors

$$\Phi(\mathbf{X})^{(l)} = g\left(\sum_{j=1}^{p} W^{(j,l)} X_i^{(j)}\right)$$

$\{1, \ldots, L\}$ on the response. In this paper, we use the Rectified Linear Units activation function, known as ReLU

$$g: \; x \mapsto max(x, 0)$$

But other choices of activation functions, such as the sigmoid function $g: \; x \mapsto \frac{1}{1+exp(-x)}$ or the hyperbolic tangent $g: \; x \mapsto tanh(x)$ (or others) can be envisaged.

The elements $W^{(j,l)}$ of the hidden layer are taken from a quasi-random (deterministic) *sobol* sequence on $[0, 1]$, which is shifted in such a way that they belong to $[-1, 1]$. In the case of bagged and stacked ensembles, solving for the optimal β_j's and γ_l's in the base-learners is done by searching these parameters, in restricted regions where we have:

$$\sum_{j=1}^{p} \beta_j^2 \leq u$$

and

$$\sum_{l=1}^{L} \gamma_l^2 \leq v$$

for $u, v \in \mathbb{R}^*$. That is, by applying a regularization to these unknown parameters. In the case of boosting, we do not restrict the regression parameters' norms to be inferior to u and v. Instead, we rely on the boosting algorithm described in [Hot+10] (details in section A.3.2), with the base-learners being the unrestricted regression models described by equation (A.1).

If we consider $p \in \mathbb{N}^*$ time series $(X_t^{(j)})_{t\geq 0}, j = 1, \ldots, p$, observed at $n \in \mathbb{N}^*$ discrete dates, we are interested in obtaining simultaneous forecasts of the p time series at time $n+h$, $h \in \mathbb{N}^*$, by allowing each of the p variables to be influenced by the others. We use $k < n$ lags of each of the observed p time series. So that, the output variables to be explained are:

$$Y^{(j)} = \left(X_n^{(j)}, \ldots, X_{k+1}^{(j)}\right)^T \tag{A.2}$$

for $j \in \{1, \ldots, p\}$. $X_n^{(j)}$ is the most contemporaneous observed value of the j^{th} time series, and $X_{k+1}^{(j)}$ was observed k dates earlier in time for $(X_t^{(j)})_{t\geq 0}$. These output variables are stored in a matrix:

$$\mathbf{Y} \in \mathbb{R}^{(n-k)\times p}$$

and the predictors are stored in a matrix:

$$\mathbf{X} \in \mathbb{R}^{(n-k)\times(k\times p)}$$

where $\mathbf{X}$ consists in p blocks of k lags, for each one of the observed p time series.

An additional layer of transformed predictors is added to $\mathbf{X}$, in order to capture the potentially non-linear interactions between the predictors and the output variable. This also serve as a way to do achieve an automated feature engineering. Adding the transformed predictors to the original ones, leads to a new matrix of predictors with dimensions $(n-k) \times (k \times p + L)$, where L is the number of nodes in the hidden layer.

For example, we have $p = 2$ time series $(X_{t_1}^{(1)}, \ldots, X_{t_5}^{(1)})$ and $(X_{t_1}^{(2)}, \ldots, X_{t_5}^{(2)})$ observed at $n = 5$ dates, $t_1 < \ldots < t_5$, with $k = 2$ lags, and $L = 3$ nodes in the hidden layer. In this case, the response variables are stored in:

$$\mathbf{Y} = \begin{pmatrix} X_{t_5}^{(1)} & X_{t_5}^{(2)} \\ X_{t_4}^{(1)} & X_{t_4}^{(2)} \\ X_{t_3}^{(1)} & X_{t_3}^{(2)} \end{pmatrix}$$

The predictors are stored in:

$$\mathbf{X} = \begin{pmatrix} X_{t_4}^{(1)} & X_{t_3}^{(1)} & X_{t_4}^{(2)} & X_{t_3}^{(2)} \\ X_{t_3}^{(1)} & X_{t_2}^{(1)} & X_{t_3}^{(2)} & X_{t_2}^{(2)} \\ X_{t_2}^{(1)} & X_{t_1}^{(1)} & X_{t_2}^{(2)} & X_{t_1}^{(2)} \end{pmatrix}$$

And the coefficients in the hidden layer are:

$$\mathbf{W} = \begin{pmatrix} W^{(1,1)} & W^{(1,2)} & W^{(1,3)} \\ W^{(2,1)} & W^{(2,2)} & W^{(2,3)} \\ W^{(3,1)} & W^{(3,2)} & W^{(3,3)} \\ W^{(4,1)} & W^{(4,2)} & W^{(4,3)} \end{pmatrix}$$

We let y be the j_0^{th} column (out of p) of the response matrix $\mathbf{Y}$, and $\Phi(\mathbf{X})$ be the matrix of transformed predictors obtained from $\mathbf{X}$ by the hidden layer. We also denote the set of regression parameters associated with this j_0^{th} time series, as:

$$\beta_m^{(j_0)} =: \beta_m$$

and

$$\gamma_l^{(j_0)} =: \gamma_l$$

for $m \in \{1, \ldots, k\}$; $l \in \{1, \ldots, L\}$. Solving for the regression parameters for the j_0^{th} time series, under the constraints

$$\sum_{m=1}^{k \times p} \beta_m^2 \leq u$$

and

$$\sum_{l=1}^{L} \gamma_l^2 \leq v$$

for $u, v \in \mathbb{R}^*$, leads to minimizing a penalized residual sum of squares:

$$\mathcal{L}(\mathbf{X}; \beta, \gamma) = (y - \mathbf{X}\beta - \Phi(\mathbf{X})\gamma)^T (y - \mathbf{X}\beta - \Phi(\mathbf{X})\gamma) + \lambda_1 \beta^T \beta + \lambda_2 \gamma^T \gamma$$

where λ_1 and λ_2 are Lagrange multipliers. Otherwise, without the constraints (in the case of boosting, see section A.3.2), the problem to be solved is simply a least squares regression problem on the augmented dataset, as described in equation A.2. By denoting:

$$A = \begin{pmatrix} \mathbf{X}^T\mathbf{X} + \lambda_1 I_{k\times p} & \mathbf{X}^T\Phi(\mathbf{X}) \\ \Phi(\mathbf{X})^T\mathbf{X} & \Phi(\mathbf{X})^T\Phi(\mathbf{X}) + \lambda_2 I_L \end{pmatrix} =: \begin{pmatrix} B & C^T \\ C & D \end{pmatrix}$$

and $S = D - CB^+C^T$. And:

$$A^+ = \begin{pmatrix} B^+ + B^+C^TS^+CB^+ & -B^+C^TS^+ \\ -S^+CB^+ & S^+ \end{pmatrix} =: \begin{pmatrix} A_1^+ & A_2^+ \\ A_3^+ & A_4^+ \end{pmatrix}$$

where S^+ and B^+ are the Moore-Penrose pseudo-inverse of matrices S and B. The whole set of parameters, for all the p observed time series is given by:

$$\begin{pmatrix} \hat{\beta} \\ \hat{\gamma} \end{pmatrix} := \begin{pmatrix} A_1^+ & A_2^+ \\ A_3^+ & A_4^+ \end{pmatrix} \begin{pmatrix} \mathbf{X}^T\mathbf{Y} \\ \Phi(\mathbf{X})^T\mathbf{Y} \end{pmatrix}$$

A.3 Ensembles of RVFL

As mentioned in the introduction, we will use bagging, boosting, and stacking to construct ensemble models, consisting of the RVFL models from the previous section A.2. A short description of these techniques will be made in next sections A.3.1, A.3.2 and A.3.3.

A.3.1 Bagging

The Bootstrap ([ET86], [Efr92]) uses multiple random replications of a given data set, to obtain standard errors of model parameters, for example. In the context of ensemble learning, these replications of the original data set are used to produce various predictions of the base models, which are then aggregated - in the case of regression problems, they are averaged, and in the case of classification problems, majority vote could be used - to obtain a single prediction with less out-of-sample variance. This procedure is called bootstrap aggregating or *bagging*.

In order to illustrate the benefits of the bagging procedure, we consider n different base learners, with out-of-sample prediction errors equal to $\epsilon_1, \ldots, \epsilon_n$, and assume that the distribution of these errors are centered around 0:

$$\mathbb{E}\left[\epsilon_i\right] = 0$$

for $i = 1, \ldots, n$. In addition, we have:

$$\mathbb{E}\left[\epsilon_i \epsilon_j\right] = \gamma$$

for $i \neq j$; that is, the covariance between the errors is equal to γ. And:

$$Var(\epsilon_i) = \sigma^2$$

for $i = 1, \ldots, n$. The out-of-sample mean squared error (MSE) of the aggregated (averaged) model including these n base models is equal to:

$$\begin{aligned}
MSE = \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n}\epsilon_i\right)^2\right] &= \frac{1}{n^2}\mathbb{E}\left[\left(\sum_{i=1}^{n}\epsilon_i\right)^2\right] \\
&= \frac{1}{n^2}\mathbb{E}\left[\sum_{i=1}^{n}\epsilon_i^2 + 2\sum_{i<j}\epsilon_i\epsilon_j\right] \\
&= \frac{1}{n^2}\left(\sum_{i=1}^{n}\mathbb{E}\left[\epsilon_i^2\right] + 2\sum_{i<j}\mathbb{E}\left[\epsilon_i\epsilon_j\right]\right) \\
&= \frac{1}{n^2}\left(n\sigma^2 + 2\frac{n(n-1)}{2}\gamma\right) \\
&= \frac{\sigma^2}{n} + \frac{(n-1)}{n}\gamma
\end{aligned}$$

From this expression of the MSE, we can observe that if n is high ($n \to \infty$), that is, if there are several base learners in the ensemble, and the out-of-sample expected error is low, the out-of-sample MSE of the ensemble prediction will decrease, and can eventually be reduced to γ. If, in addition, the model predictions are perfectly uncorrelated, i.e $\gamma = 0$, then the out-of-sample MSE is further decreased.

Having a low value for γ, along with a low out-of-sample expected error, helps in achieving a lower out-of-sample prediction error for the ensemble model. Decorrelation in ensemble learning is hence important, and consists in increasing the diversity in the base learners, in order to obtain a low value for γ.

This decorrelation among the base learners is achieved for example in Random Forest models [Bre01], by growing each tree in the forest, with only a subset of the predictors available. Also, an example of use of decorrelation, specifically for ensembles of neural networks is demonstrated in [Ros96]. The author presents three approaches for achieving *disagreement* between the networks: one in which they are trained independently and aggregated with the hope that their predictions

are somewhat different; a second one, in which different activation functions or achitecture (typically, more or less hidden layers or nodes in the hidden layers etc.) for each base learner. A third approach consists in training the individual networks on different subsamples of the original training set. The case of decorrelation learning for RVFL is treated in [AW14].

In both papers, [Ros96] and [AW14], the procedure implemented in order to obtain a decorrelation of the base learners is denoted as Negative Correlation Learning (NCL). The general idea is to minimize the penalized Root Mean Squared Error:

$$\sum_{i=1}^{n} \Big[(y_i - f_k(\mathbf{x}_i))^2 + \lambda p_k(\mathbf{x}_i) \Big]$$

where $\mathbf{x}_i \in \mathbb{R}^p$ is the i^{th} observation of the training data set, with p features. f_k is a base learner with $k \in \{1 \dots B\}$, and B is the number of bootstrap resamples. p_k is a penalty term decorrelating the current network's error with the errors of the networks previously trained. λ is a Lagrange multiplier, a regularization parameter, preventing the correlation between the successive base learners in-sample errors from being high. For example, p_k could be defined as:

$$p_k(\mathbf{x}_i) = (y_i - f_k(\mathbf{x}_i)) \sum_{j<k} (y_i - f_j(\mathbf{x}_i))$$

In this chapter, we use the third approach described in [Ros96] (although many other approaches can be envisaged), which is actually more *subsampling* than bootstrapping. Multiple samples (training) data sets are constructed by randomly picking a fraction `col_sample_coeff` of the covariates, and a subset (using the fraction `row_sample_coeff`) of the lines, as described in figure A.1 (using notations from section A.2).

Taking a subset of the lines is done by respecting the serial dependence of the series and taking consecutive observations (that is, without skipping any observation) for each bootstrap resample.

The predictions associated to each training set resample are then averaged, in order to obtain a single prediction, with an uncertainty around it. Some numerical examples on the application of bagging to RVFL base learners can be found in section B.4

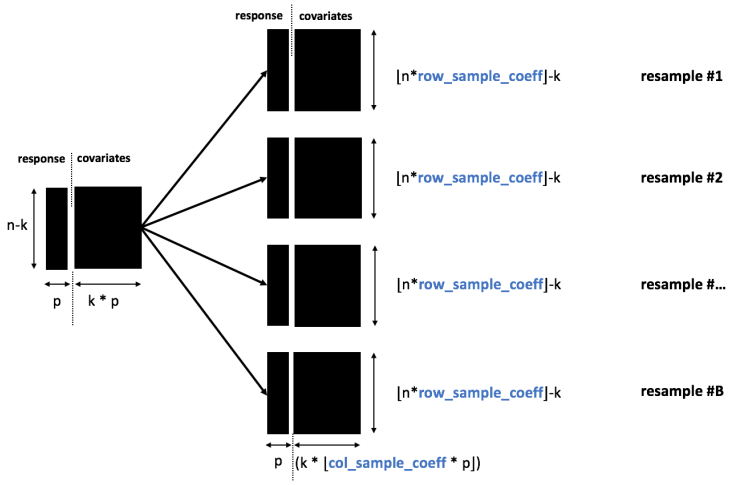

Fig. A.1.: Construction of B bootstrap resamples by using the initial data

A.3.2 Boosting

The boosting approach adopted here, is described in [Hot+10] and implemented in R package `mboost`. The general idea of the algorithm is to fit the in-sample residuals with base-learners, iteratively and slowly, but to stop learning before the out-of-sample error starts to worsen. With the response being y and the covariates stored in $\mathbf{x}$, we are interested in constructing a regression function $\hat{f}$, so that:

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{B} \nu \hat{f}_j(\mathbf{x})$$

B is the number of boosting iterations, ν is a learning rate parameter preventing the base learners from fitting the residuals too quickly, and $\hat{f}_j$, $j \in \{1, \ldots, B\}$ are the base learners, which are obtained as:

$$\left(\hat{f}_1, \ldots, \hat{f}_p\right) = argmin_{f_1,\ldots,f_p} \rho \left(y, \sum_j \nu f_j\left(\mathbf{x}\right)\right)$$

In this expression, ρ is a loss function to be minimized (for example $x \mapsto x^2$, or $x \mapsto |x|$). And in the case of multivariate time series, we are doing multitask learning. Each series (their most contemporaneous observations) share the same set of predictors (their lagged observations) as in the RVFL model described in section A.2, but without regularization of the regression models' parameters.

The same number of boosting iterations and the same learning rate are used for all the observed time series. It would be possible to consider different numbers of iterations and different learning rates for each time series, but in this situation there

would be as much as parameters as time series, and the regularization parameters would be trickier to optimize.

A.3.3 Stacking

Stacked generalization or stacking, was introduced in [Wol92]. It is also presented in [Bre96b] for regression models. The idea behind this procedure is to construct new predictors for the training data set, by using diverse predictions of multiple models, which are combined by a meta-learner model. A simple example of 2-fold stacking applied to regression is described hereafter. It is possible to imagine examples with more stacked layers:

- Divide the original training data set $\mathbf{X} \in \mathbb{R}^{n \times p}$, into two parts: `part1` and `part2`, each with $n/2$ rows
- Train a base learner model on `part1` and obtain $n/2$ predictions on `part2`.
- Train the same base learner model on `part2` and obtain $n/2$ predictions on `part1`.
- Train a meta-learner model on the new data set `part3`, consisting in the the original predictors, plus the predictors constructed by applying the base learner model on `part1` and `part2`.

Typically, in this procedure, various types of base learners are used, to increate the diversity of new predictors. Since we are using time series data here, which inherently exhibit a serial dependence between the observations, we have to adapt the procedure described previously. In the case of a 2-fold stacking again, we divide the original training data set $\mathbf{X} \in \mathbb{R}^{n \times p}$ (with p time series observed at n dates) into two parts: `part1` and `part2`; each consisting of $n/2$ rows. `part1` contains the most ancient observations and `part2`, the most recent observations of the time series contained in $\mathbf{X}$. Also, we use multiple resamples of the base learner, as follows (this procedure is also described in figure A.2):

- Train B resampled RVFL models on `part1`, using a random subset of `part1` in each resampling iteration. Obtain B sets of predictions ($B \times p$ new predictors) with these models, over the whole horizon of `part2`.
- Create a new enriched data set `part3`, containing the observed time series from `part2`, and the $B \times p$ additional series predicted from `part1` on `part2`.

- Use a meta-learning model to obtain predictions on the new, enriched dataset, `part3`.

For the latter point, we consider a few meta-learner models that have been used in the past in the literature ([Bre96b]). In general, the idea is that a relatively simple meta-learner will achieve a good performance. We use the following models for this purpose:

- Linear **least squares regression** model.
- Linear least squares regression model **with positive coefficients** ([LH74], [LH95]), as used in [Bre96b].
- **Ridge regression** ([HK70]).

These selected meta-learners are trained on `part3`, with the most contemporaneous observations of the series as responses, and their respective lags as predictors. Some numerical examples on the application of stacking to RVFL base learners can be found in section B.4

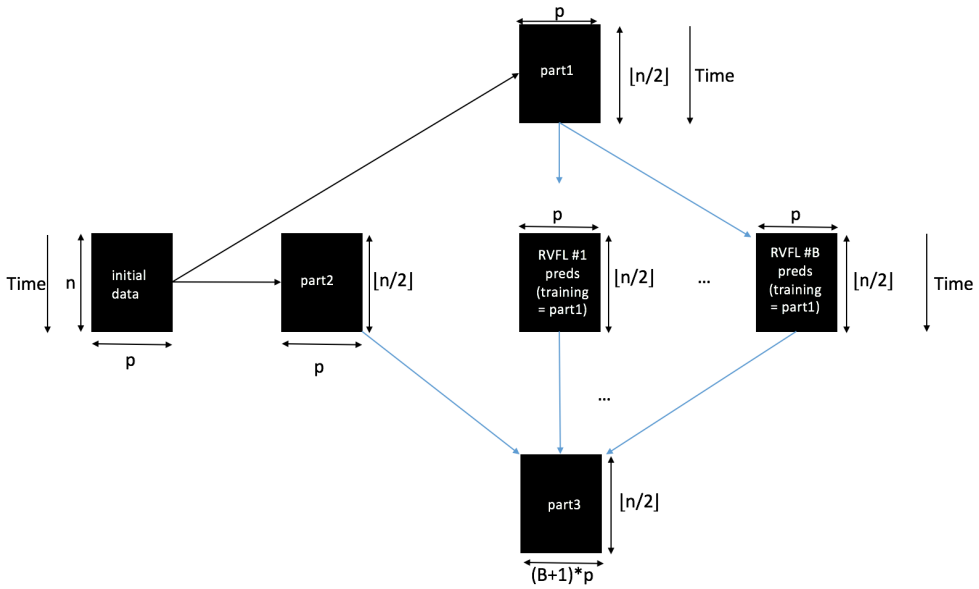

Fig. A.2.: Construction of the enriched/stacked dataset

A.4 Numerical examples

For the numerical examples, we obtain forecast of the Treasury Bill rates among other macroeconomic variables, in a *data-rich* environment. We use data from [Gre00]. Four of the time series are considered: the Treasury Bill rates, the real consumption,

the consumer price index, and the real expenditure. They are all observed on a quaterly basis.

We annualize the last three time series, but keep the Treasury Bill rates unchanged. So that, the four time series are nearly on the same scale.

The out-of-sample Root Mean Squared Error (RMSE) is used as a measure of performance of the different implemented models.

A.4.1 Descriptive statistics

Figure A.3 presents the resulting time series' data obtained after the few transformations described in the previous paragraph. Table A.1 contains a summary of these data, where we can see that the four time series are now nearly expressed in the same scale.

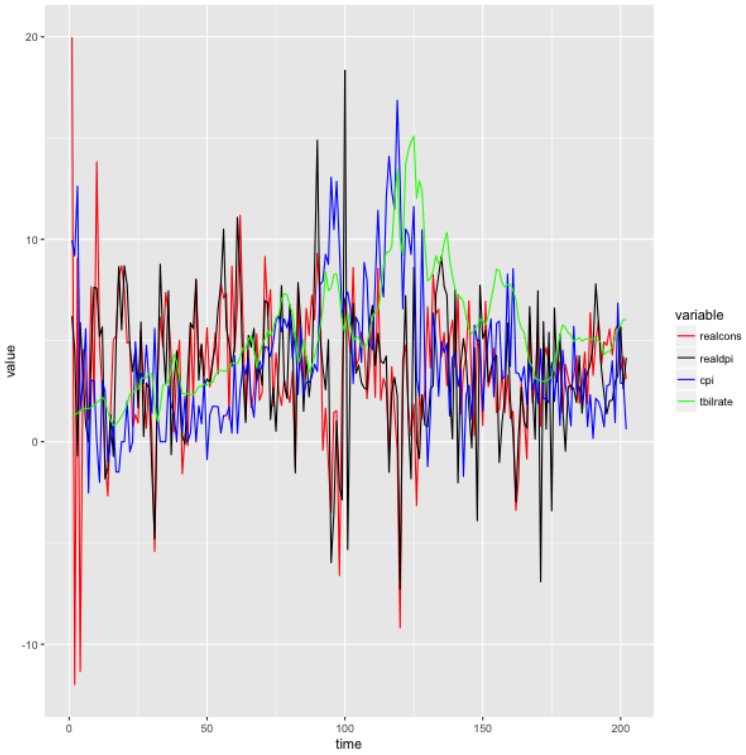

Fig. A.3.: Treasury Bills rates and transformed Real consumption, Real disposable income, and Inflation data from [Gre03] in a *data-rich* environment

Tab. A.1.: Summary statistics for the four transformed time series: Treasury Bills and transformed Real consumption, Real disposable income, and Inflation data from [Gre03]

Series	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev.
Real consumption	-11.990	1.814	3.637	**3.513**	5.414	19.978	**3.546**
Real disposable income	-7.275	1.613	3.390	**3.423**	5.518	18.358	**3.486**
Inflation	-2.530	1.755	3.135	**3.936**	5.593	16.864	**3.407**
Treasury Bills rates	0.810	3.138	5.050	**5.270**	6.715	15.090	**2.829**

Tab. A.2.: **Box-Pierce test for the independence** in the time series observations

Series	`X-squared`	`p-value`
Real consumption	0.0133	**0.9082**
Real disposable income	1.3912	**0.2382**
Inflation	86.428	2.2e-16
Treasury Bills rates	186.35	2.2e-16

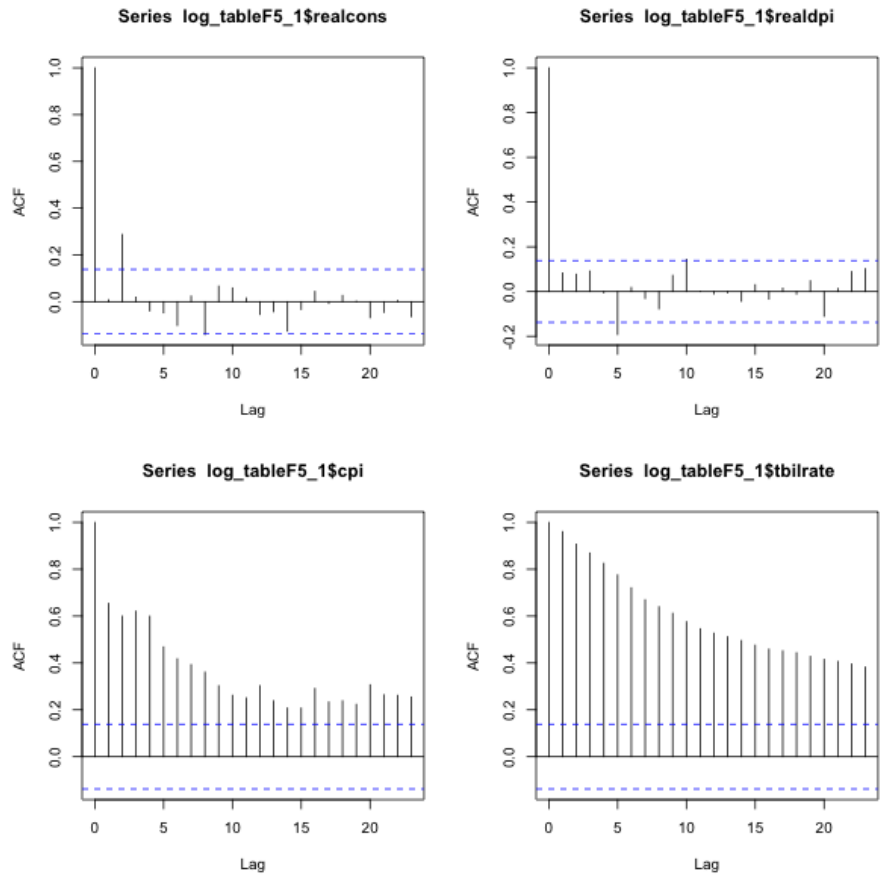

Fig. A.4.: Autocorrelation function for the four times series: Treasury Bills rates, Real consumption, Real disposable income, Inflation

The Box-Pierce tests ([BP70], [LB78], [HH93]) results presented in table A.2, and the autocorrelation functions presented in A.4, show that the two time series for Real consumption and Real disposable income could be considered as stationary after the transformations. Whereas for the Treasury Bills rates and Inflation, there is still a non-negligible autocorrelation within the series.

Another interesting information is given by figure A.5. We observe that the distribution of the data for inflation and Treasury Bills rates is skewed, when compared to the data for Real consumption and Real disposable income.

Also, based on the correlations displayed in figure A.5, the Real consumption is globally decreasing when the Treasury Bills rates and the inflation are increasing. But when the Real disposable income increases, the Real consumption increases as well.

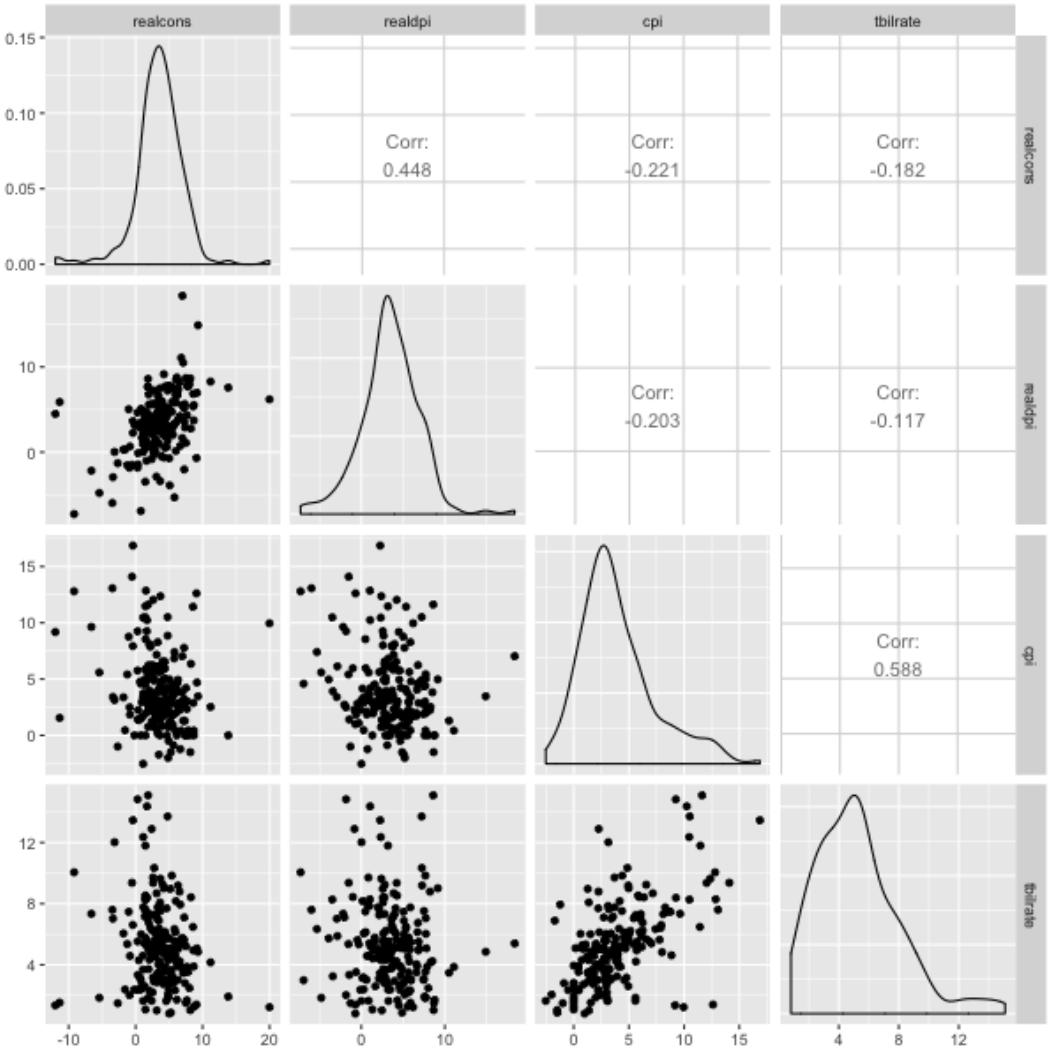

Fig. A.5.: Correlation plot for the four times series: Treasury Bills rates, Real consumption, Real disposable income, Inflation

In section A.4.2, we apply the bagging and stacking procedures described in sections A.3.1 and A.3.3 to the dataset we have just described.

A rolling forecasting cross-validation method (see [Ber+15] and figure B.3). At the beginning of the procedure, the training set contains 24 points (4 years), and the test set contains 4 points (1 year) and 8 points (2 years). The training set is then advanced of one point forward (rolled), and the procedure is repeated until no more data for the training set are found.

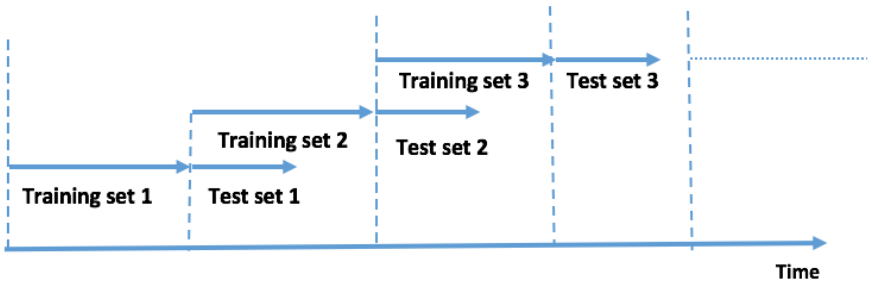

Fig. A.6.: Training and testing sets in rolling forecasting cross-validation

As we said at the beginning of section B.4, we use the Root Mean Squared Error (RMSE) as a measure of the out-of-sample error. All the benchmarks are made on what is defined as `part3` in section A.3.3, in order to have a similar perimeter for bagging and stacking. The results obtained by the ensemble learners are compared to those obtained by the base RVFL model, which are presented here:

For the bagging example, we use $B = 100$ resamples of the data, and calculate the average out-of-sample RMSE obtained by the ensemble model on all the testing sets (see figure B.3). The dataset used, is the one defined as `part3` in section A.3.3.

For the boosting example, we use $B = 10$ boosting iterations of the least squares regression model from formula A.1. The tuning parameters are the learning rate of the boosting algorithm denoted as ν, the number of nodes in the hidden layer for the base-learner `nb_hidden`, the fraction of the initial time series considered as predictors `col_sample`, the number of lags.

For the stacking example, The dataset used, is also the one defined as `part3` in section A.3.3. The tuning parameters are the number of boostrap resamples, B, and the hyperparameter of the ridge regression model is the regularization parameter.

A.4.2 Summary of results

A summary of the out-of-sample RMSE is presented in figure A.7 in a logarithmic scale, for all the methods that we described and tested in the previous sections. The numerical values corresponding to these boxplots can be found in appendix A.5.1, in their original scale.

The base RVFL model is denoted as `ridge2`. `bagging`, `stack_ridge`, `stack_lm`, `stack_nnls`, and `glmboost` respectively denote the bootstrap aggregating method from section A.3.1, stacking with ridge regression, with ordinary least squares, least squares regression with positive coefficients from section A.3.3, and boosting from section A.3.2. We also added an unrestricted Vector AutoRegression (VAR) model to the benchmarks, denoted as `VAR`, which mostly helps us in assessing if the other models' implementation/predictions are not off track.

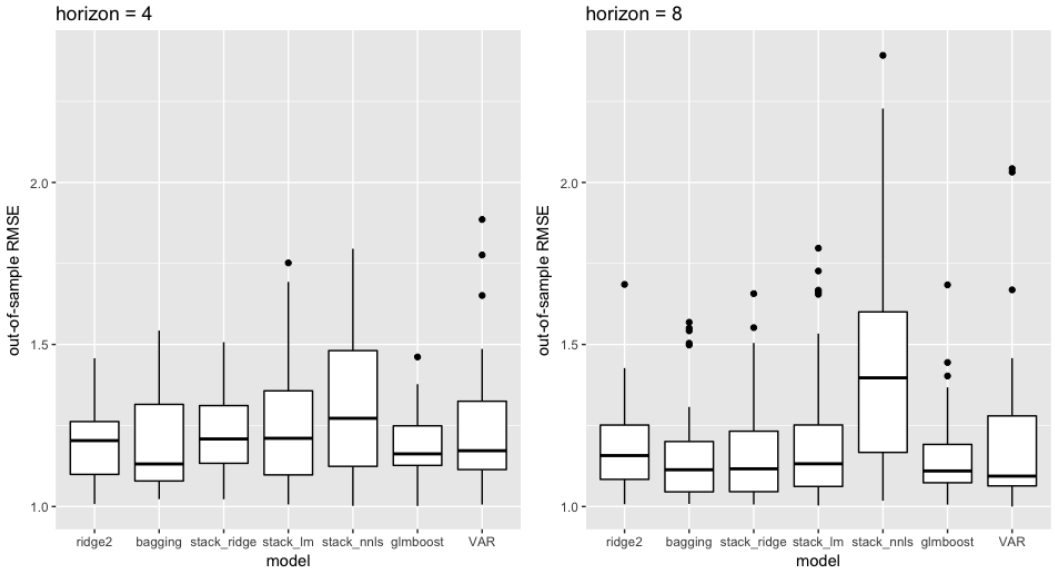

Fig. A.7.: Out-of-sample RMSE distribution for the different tested models

Bagging is performing the best on average (cf. appendix A.7 for details) on both horizons, 4-steps ahead and 8-steps ahead, and is followed by boosting. The stacking algorithm works best with the ridge regression model taken as a meta-learner, but is failing with the other meta learners (ordinary least squares and least squares regression with positive coefficients). Overall, the boosting algorithm is also very robust, and displays the lowest standard deviation of all the models tested.

The *best* hyperparameters obtained for each algorithm are presented below, for the different horizons of projection. The best hyperparameter found for stacking with ordinary least squares, with a constraint on the regression parameters or not, is the number of lags equal to 1 and $B = 1$.

For horizon = 4

Tab. A.3.: *Best* hyperparameters for the base RVFL

Method	lags	nodes	λ_1	λ_2
RVFL	3	100	10000	100

Tab. A.4.: *Best* hyperparameters for the bagging of RVFLs

Method	lags	nodes	λ_1	λ_2	B	col. sample	row. sample
Bagging	1	1	21.54435	21.54435	100	0.9	0.9

Tab. A.5.: *Best* hyperparameters for the stacking with ridge regression

Method	lags	λ	B	col. sample	row. sample
Stacking (ridge)	3	21.54435	1	0.9	0.8

Tab. A.6.: *Best* hyperparameters for the boosting algorithm

Method	lags	nodes	ν	B	col. sample
Boosting	3	1	0.2020798	10	1

Tab. A.7.: *Best* hyperparameters for the VAR

Method	lags	type of regressor
VAR	1	const

For horizon = 8

Tab. A.8.: *Best* hyperparameters for the base RVFL

Method	lags	nodes	λ_1	λ_2
RVFL	3	2	21.54435	10000

Tab. A.9.: *Best* hyperparameters for the bagging of RVFLs

Method	lags	nodes	λ_1	λ_2	B	col. sample	row. sample
Bagging	1	1	21.54435	21.54435	100	0.9	0.9

Tab. A.10.: *Best* hyperparameters for the stacking with ridge regression

Method	lags	λ	B	col. sample	row. sample
Stacking (ridge)	3	21.54435	1	0.9	0.9

Tab. A.11.: *Best* hyperparameters for the boosting algorithm

Method	lags	nodes	ν	B	col. sample
Boosting	3	1	0.175	10	1

Tab. A.12.: *Best* hyperparameters for the VAR

Method	lags	type of regressor
VAR	1	none

Below, in figure A.8, we can observe how the different methods perform through time. We observe that the models' errors are highly correlated, which is reassuring concerning their implementation.

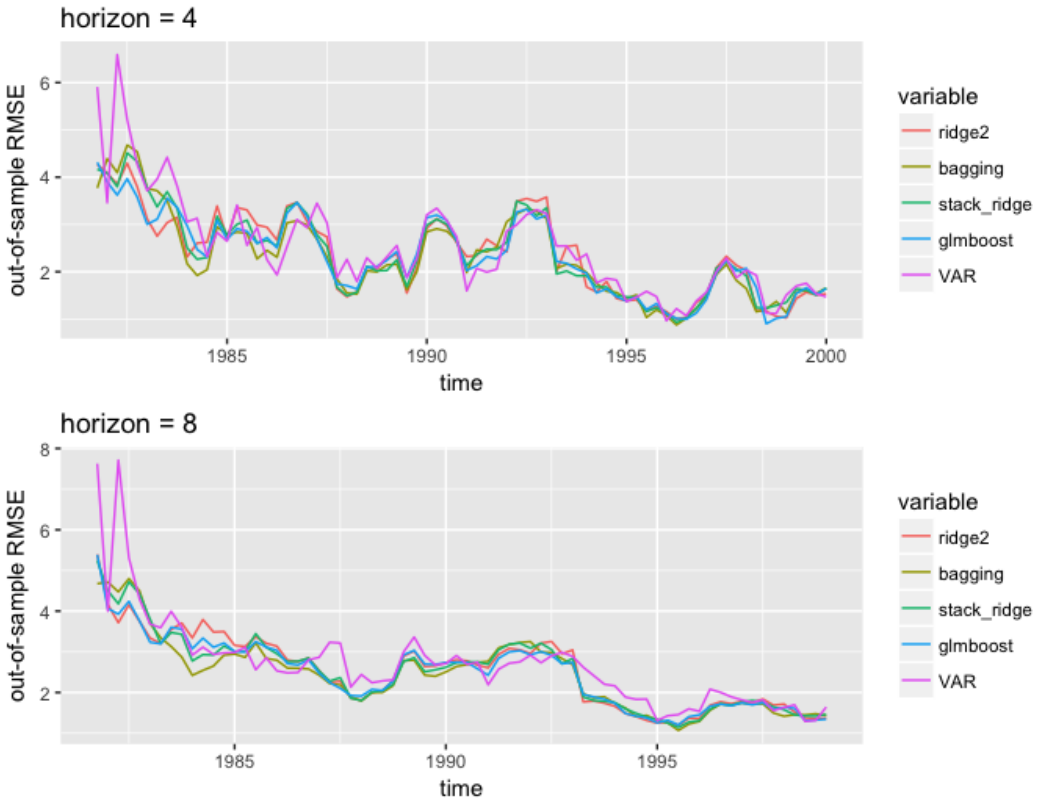

Fig. A.8.: Out-of-sample RMSE over time for the different tested models + VAR

The details of the correlations between these errors can be found in appendix A.5.2. The VAR model generally exhibits a higher variance than the other models (excluding stacking with `stack_lm` and `stack_nnls`), but is still able to perform better than them at times (due to the high variance).

A.5 Appendix

A.5.1 Out-of-sample errors summary

horizon = 4

- Min., 1st Qu., Median, Mean, 3rd Qu., Max

```
ridge2           bagging          stack_ridge      stack_lm
Min.   :0.9995   Min.   :0.8746   Min.   :0.9291   Min.   :0.9784
1st Qu.:1.5529   1st Qu.:1.5908   1st Qu.:1.6426   1st Qu.:1.7379
Median :2.3605   Median :2.1484   Median :2.2407   Median :2.1715
Mean   :2.3763   Mean   :2.3034   Mean   :2.3676   Mean   :2.5058
3rd Qu.:3.0261   3rd Qu.:2.8957   3rd Qu.:3.0690   3rd Qu.:3.0031
Max.   :4.2952   Max.   :4.6781   Max.   :4.5133   Max.   :5.7641

stack_nnls          glmboost          VAR
Min.   :0.9874      Min.   :0.8996    Min.   :1.041
1st Qu.:1.7273      1st Qu.:1.6477    1st Qu.:1.680
Median :2.2839      Median :2.2443    Median :2.146
Mean   :2.5115      Mean   :2.3261    Mean   :2.498
3rd Qu.:3.0494      3rd Qu.:3.0648    3rd Qu.:3.070
Max.   :6.0210      Max.   :4.3121    Max.   :6.068
```

- Standard deviation

```
   ridge2   bagging stack_ridge    stack_lm  stack_nnls    glmboost   VAR
0.8804324 0.8881329   0.8870853   1.0549136   1.0889430   0.8238531   1.051120
```

horizon = 8

- Min., 1st Qu., Median, Mean, 3rd Qu., Max

```
 ridge2          bagging        stack_ridge       stack_lm
Min.   :1.192   Min.   :1.066   Min.   :1.143   Min.   :1.253
1st Qu.:1.740   1st Qu.:1.717   1st Qu.:1.754   1st Qu.:1.813
Median :2.663   Median :2.467   Median :2.656   Median :2.694
Mean   :2.529   Mean   :2.440   Mean   :2.509   Mean   :2.676
3rd Qu.:3.154   3rd Qu.:2.868   3rd Qu.:3.041   3rd Qu.:3.065
Max.   :5.394   Max.   :4.800   Max.   :5.243   Max.   :6.032
```

```
stack_nnls          glmboost          VAR
Min.   : 1.164      Min.   :1.203     Min.   :1.289
1st Qu.: 2.056      1st Qu.:1.718     1st Qu.:1.920
Median : 2.775      Median :2.683     Median :2.648
Mean   : 3.331      Mean   :2.478     Mean   :2.696
3rd Qu.: 4.069      3rd Qu.:3.024     3rd Qu.:2.964
Max.   :10.935      Max.   :5.387     Max.   :7.711
```

- Standard deviation

```
    ridge2     bagging stack_ridge    stack_lm  stack_nnls    glmboost         VAR
0.8955453   0.8883188   0.9093582   1.0406195   1.8865643   0.8547240   1.1482777
```

A.5.2 Correlation of out-of-sample errors

horizon = 4

```
            ridge2   bagging      stack_ridge  glmboost       VAR
ridge2      1.0000000 0.9350414   0.9658978    0.9627834      0.8422500
bagging     0.9350414 1.0000000   0.9796091    0.9388819      0.8746823
stack_ridge 0.9658978 0.9796091   1.0000000    0.9731609      0.8758017
glmboost    0.9627834 0.9388819   0.9731609    1.0000000      0.8840573
VAR         0.8422500 0.8746823   0.8758017    0.8840573      1.0000000
```

horizon = 8

```
               ridge2  bagging stack_ridge  glmboost       VAR
ridge2      1.0000000 0.9280983   0.9693424 0.9909838 0.8185879
bagging     0.9280983 1.0000000   0.9855720 0.9481774 0.8522099
stack_ridge 0.9693424 0.9855720   1.0000000 0.9804522 0.8561227
glmboost    0.9909838 0.9481774   0.9804522 1.0000000 0.8622763
VAR         0.8185879 0.8522099   0.8561227 0.8622763 1.0000000
```

B Forecasting discount curves with Kernel Regularized Least Squares

B.1 Introduction

In this chapter, we apply Kernel Regularized Least Squares (KRLS) learning methods ([JC14], [Ext+16], [Fer+17]) to Yield Curve forecasting. By *Yield Curve*, we actually mean *discount curves*. That is, we consider that the curves used in the examples do not include any counterparty credit risk, and focus on the forecasting problem. Two types of formulations of the spot rates' forecasting problem are tested here. One relying on the popular Dynamic Nelson-Siegel (DNS) framework from [DL06], and another one, in which we apply the KRLS directly to the discount curves' observation dates and time to maturities, to model the spot rates.

In the DNS framework [DL06], each cross-section of yields observed over time is fitted by using the Nelson-Siegel [NS87] model. The fitting of each cross-section observed over time, produces three time series of parameters (more details in the next section) representing the evolution of the level, slope, and curvature of the Yield Curve. A KRLS model is applied to forecasting the time series of parameters, using a technique which is similar to the one described in [Ext+16]. And to finish, the forecast obtained for the trivariate time series are plugged into the Nelson-Siegel model formula, to deduce forecast for the cross-sections of yields.

The second approach based on KRLS is a machine learning/data-driven one, in which we put no specific constraint on the model to reproduce the specific Yield Curve stylized facts. The regularization parameters inherent to the KRLS models will act as implicit constraints, that cause the model to converge as close as possible to reproducing these stylized facts. In this latter approach, we are mostly interested in the model with the *best* out-of-sample error. As a consequence, the technique as is, is probably less adapted than the former framework based on DNS (in its arbitrage-free version) to no-arbitrage pricing (if no-arbitrage pricing is required).

To introduce KRLS, we start by describing the ridge regression [HK70] and the *kernel trick* applied to ridge regression. Then, we make a link between the ridge regression and KRLS.

In a ridge regression setting, we want to explain an observed variable $y \in \mathbb{R}^n$, as a linear function of p predictors stored in a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. For $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, p\}$ we have:

$$X_{ij} =: \mathbf{x}_i^{(j)}$$

We will denote the i^{th} row of $\mathbf{X}$ as $\mathbf{x}_i$, and its j^{th} column as $\mathbf{x}^{(j)}$. Hence, we are searching for the parameters $\beta = (\beta_1, \dots, \beta_p)^T$ verifying:

$$ArgMin_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^T \beta\right)^2$$

under the constraint

$$||\beta||_2^2 \leq s$$

The solution to this problem is given directly by the formula:

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X} + \lambda I_{p \times p}\right)^{-1} \mathbf{X}^T y$$

where λ is a Lagrange multiplier having a unique correspondance with s, and a regularization parameter preventing the model from overfitting the observed data contained in y. In the case where we want to explain y as a function Φ of the predictors, we have a similar expression:

$$\hat{\beta} = \left(\Phi(\mathbf{X})^T\Phi(\mathbf{X}) + \lambda I_{p \times p}\right)^{-1} \Phi(\mathbf{X})^T y$$

where:

$$\Phi(\mathbf{X})_{ij} = \Phi(\mathbf{x}_i^{(j)})$$

Now, by using the Woodbury identity ([GVL12] and [Wel10]) for $\mathbf{P}$ and $\mathbf{R}$ positive definite

$$\left(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\right)^{-1} \mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T \left(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}\right)^{-1}$$

The solution to the ridge regression problem can be re-written as:

$$\hat{\beta} = \Phi(\mathbf{X})^T \left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T + \lambda I_{n \times n}\right)^{-1} y$$

This relationship can be useful in the case where $n << p$. That is, when there is a high number of predictors compared to the number of observations ([Ext+16]). Indeed, with this new relationship, we are no longer calculating/inverting a $p \times p$ matrix, but a $n \times n$ matrix. That's the *kernel trick*. And if some new observations arrive, and are stored in $\mathbf{X}^*$, the new values predicted by the model will be given by:

$$y^* = \Phi(\mathbf{X}^*)\hat{\beta} = \Phi(\mathbf{X}^*)\Phi(\mathbf{X})^T \left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T + \lambda I_{n\times n}\right)^{-1} y$$

Which we re-write as:

$$y^* = \mathbf{K}^* \left(\mathbf{K} + \lambda I_{n\times n}\right)^{-1} y$$

$\mathbf{K}$ is a *kernel*; the empirical covariance matrix of $\Phi(\mathbf{X})^T$ (modulo a $1/p$ factor), in the case where the rows of $\Phi(\mathbf{X})$ are centered. Now, in the case of KRLS, the problem we are trying to solve is:

$$ArgMin_{c\in\mathbb{R}^n} \sum_{i=1}^{n} \left(y_i - K_i^T c\right)^2$$

where K_i is the i^{th} row of $\mathbf{K}$, with:

$$\mathbf{K}_{ij} =: K(\mathbf{x}_i, \mathbf{x}_j) = f(||\mathbf{x}_i - \mathbf{x}_j||_1) \ \ or \ \ f(||\mathbf{x}_i - \mathbf{x}_j||_2)$$

The cost of computing the whole kernel $\mathbf{K}$, for any i and j is a quadratic function of the number of observations, n. Meaning that, the most interesting cases for using the KRLS method would be those in which n is not too high.

As described in [JC14], two approaches can be used to interpret/motivate the KRLS learning method: a similarity-based view and the superposition of Gaussians view. Here, we refer only to the first one, which is the most intuitive to us, and is also the one described in [Fer+17]. Indeed here, the i^{th} observation of the response, y_i is explained as a linear combination of functions measuring the similarity/dissimilarity between its characteristics gathered in $\mathbf{x}_i$, and the other observations from the training set, $\mathbf{x}_j$, $j \neq i$:

$$y_i = \sum_{j=1}^{n} c_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{B.1}$$

But again, to prevent the model from overfitting the observed data and not being able to generalize well, we need to constrain the parameters c under a certain norm defined by the matrix $\mathbf{K}$ [Hof+08]:

$$||c||_K^2 = c^T \mathbf{K} c \leq s$$

So that the solution to this new (constrained) problem is:

$$\hat{c} = (\mathbf{K} + \lambda I_{n \times n})^{-1} \mathbf{y}$$

λ is a Lagrange multiplier having a unique correspondance with s, and a regularization parameter. And for new observations arriving for the model, we have a solution which is identical to the one that we had for kernel ridge regression:

$$y^* = \mathbf{K}^* (\mathbf{K} + \lambda I_{n \times n})^{-1} y$$

Many other types of kernels could be envisaged for $\mathbf{K}$, allowing to take into account nonlinearities and the various complexities of the covariance structure. One of the most popular kernels is the Gaussian kernel, also called *squared exponential* kernel, defined for $i < j$ by:

$$K_{Gauss}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{2l^2}\right)$$

where l is a characteristic length-scale controlling the distance between peaks of the covariance function and σ^2 is the marginal variance, obtained when $\mathbf{x}_i = \mathbf{x}_j$. Both l, σ^2 are used as the learning model's hyperparameters, along with the regularization parameter λ.

This kernel is however often judged as being too smooth for most typical optimization problems [RW06]. Some other kernels that could be interesting for machine learning (see [RW06]) belong to the Matérn class of covariance functions. If we define $r := ||\mathbf{x}_i - \mathbf{x}_j||_2$, the most used for machine learning problems (see [RW06]) are:

$$\mathbf{K}_{ij} = K_{3/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) exp\left(-\frac{\sqrt{3}r}{l}\right)$$

and

$$\mathbf{K}_{ij} = K_{5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) exp\left(-\frac{\sqrt{5}r}{l}\right)$$

Figure B.1 below, provides more insights on the kernels we have just defined; K_{Gauss}, $K_{Matérn3/2}$ and $K_{Matérn5/2}$, for $\sigma^2 = 1$ and $l = 1$.

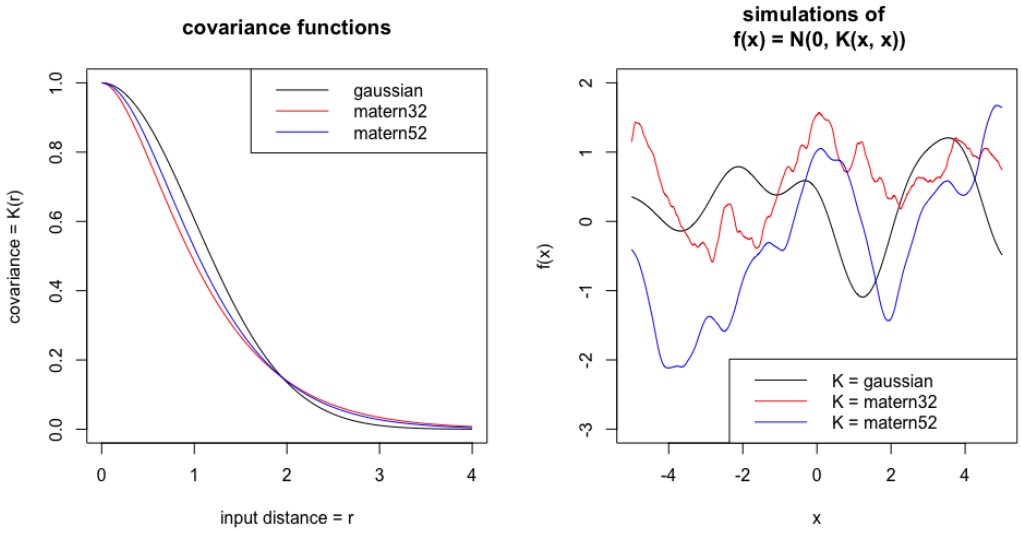

Fig. B.1.: **Left:** covariance functions; **Right:** random simulations from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, K)$, with $\sigma^2 = 1$ and $l = 1$. The sample functions on the right were obtained using a discretization of the x-axis of 1000 equally-spaced points

We observe in figure B.1 (**left**) that: the smoother the kernel, the higher the covariance associated to observations that are close to each other (that are similar). This relationship is inverted as the distance between the observations grows, but with a lower magnitude. The Gaussian kernel is the more flexible of the three kernels. Then, comes the kernel Matérn 5/2, and to finish, the kernel Matérn 3/2 (cf. figure B.1, **right**).

An interesting feature of KRLS learning, is the possibility to derive estimators for the marginal effects of the covariates $\mathbf{x}^{(j)}$ on the response. For example, (and like in [Fer+17]), since we have the relationship (B.1), we can write for a fixed $j_0 \in \{1, \ldots, p\}$ and for any $k \in \{1, \ldots, n\}$, :

$$\frac{\partial y_i}{\partial \mathbf{x}_k^{(j_0)}} = \sum_{j=1}^{n} c_j \frac{dK(\mathbf{x}_i, \mathbf{x}_j)}{d\mathbf{x}_k^{(j_0)}} = c_k \frac{\partial K(\mathbf{x}_i, \mathbf{x}_k)}{\partial \mathbf{x}_k^{(j_0)}} \tag{B.2}$$

That's an approximation of how much of an increase (at the first order) we obtain in y_i, for a slight change in $\mathbf{x}_k^{(j_0)}$. An average marginal effect of the j_0^{th} covariate on the i^{th} observation of the response y can thus be obtained as:

$$\frac{1}{n}\sum_{k=1}^{n}\frac{\partial y_i}{\partial x_k^{(j_0)}} = \frac{1}{n}\sum_{k=1}^{n} c_k \frac{\partial K(\mathbf{x}_i, \mathbf{x}_k)}{\partial \mathbf{x}_k^{(j_0)}} \tag{B.3}$$

Two types of KRLS' model formulations will be described in sections B.2 and B.3. One relying on the famous Dynamic Nelson-Siegel (DNS) framework ([DL06]), and another one explaining the spot rates as a function of the time to maturity and date of observation.

More specifically in sections B.2.2 and B.3.2, we derive formulas for the sensitivities of the response variable y_i to a change in the covariates $\mathbf{x}^{(j)}$, in each one of the two frameworks. We do this for Gaussian, Matérn 3/2 and Matérn 5/2 kernels. These sensitivities can then be plugged into formula B.3, in order to obtain average marginal effects of the covariates on the response.

B.2 Description of the DNS-KRLS model and model's sensitivity

B.2.1 The DNS-KRLS model

In the DNS framework ([DL06]), the spot interest rates observed at time t, for time to maturity τ are modeled as:

$$R_t(\tau) = \alpha_{1,t} + \alpha_{2,t}\left(\frac{1-e^{-\tau/\eta}}{e^{-\tau/\eta}}\right) + \alpha_{3,t}\left(\frac{1-e^{-\tau/\eta}}{e^{-\tau/\eta}} - e^{-\tau/\eta}\right) \tag{B.4}$$

If these spot interest rates $R_t(\tau)$ are observed at increasing dates $t = t_1 < \ldots < t_n$, for increasing time to maturities $\tau = \tau_1 < \ldots < \tau_p$, the factor loadings in the DNS framework are the vectors (of length p):

$$(1, \ldots, 1)^T$$

and

$$\left(\frac{1-e^{-\tau_1/\eta}}{e^{-\tau_1/\eta}}, \ldots, \frac{1-e^{-\tau_p/\eta}}{e^{-\tau_p/\eta}}\right)^T$$

and

$$\left(\frac{1-e^{-\tau_1/\eta}}{e^{-\tau_1/\eta}}-e^{-\tau_1/\eta},\ldots,\frac{1-e^{-\tau_p/\eta}}{e^{-\tau_p/\eta}}-e^{-\tau_p/\eta}\right)^T$$

These vectors are used to represent respectively the level, slope, and curvature of the Yield Curve. Estimations of $\alpha_{i,t}, i=1,\ldots,3$ are obtained for each cross-section of yields (that is, for each fixed date t) by taking a fixed η, and doing a least squares regression of the spot rates observed at time t on these factor loadings.

The three time series $(\alpha_{i,t})_t, i=1,\ldots,3$ associated to the loadings for each cross-section of yields, are those that we wish to forecast simultaneously, by using KRLS learning. For doing this, we store the most contemporaneous values of the three time series $(\alpha_{i,t})_t, i=1,\ldots,3$ in a response matrix $\mathbf{Y}$, and their lags in a matrix of predictors $\mathbf{X}$.

Considering the $p\in\mathbb{N}^*$ time series $(\alpha_t^{(j)})_{t\geq 0}, j=1,\ldots,p$ (with $p=3$), observed at $n\in\mathbb{N}^*$ discrete dates. We are interested in obtaining simultaneous forecasts of the p time series at time $n+h$, $h\in\mathbb{N}^*$, by allowing each of the p variables to be influenced by the others (in the spirit of VAR models, see [Lüt05]). We use $k<n$ lags of each of the observed p time series. Hence, the output variables (columns of $\mathbf{Y}$) to be explained are:

$$Y^{(j)}=\left(\alpha_n^{(j)},\ldots,\alpha_{k+1}^{(j)}\right)^T \tag{B.5}$$

for $j\in\{1,\ldots,p\}$. Where $\alpha_n^{(j)}$ is the most contemporaneously observed value of the j^{th} time series, and $\alpha_{k+1}^{(j)}$ was observed k dates earlier in time for $(\alpha_t^{(j)})_{t\geq 0}$. These output variables are stored in:

$$\mathbf{Y}\in\mathbb{R}^{(n-k)\times p}$$

and the predictors are stored in:

$$\mathbf{X}\in\mathbb{R}^{(n-k)\times(k\times p)}$$

where $\mathbf{X}$ consists in p blocks of k lags, for each one of the observed p time series. For example, the j_0^{th} block of $\mathbf{X}$, for $j_0\in\{1,\ldots,p\}$ contains in columns:

$$\left(\alpha_{n-i}^{(j_0)},\ldots,\alpha_{k+1-i}^{(j_0)}\right)^T \tag{B.6}$$

with $i \in \{1, \ldots, k\}$. If we consider the $p = 3$ time series $(\alpha_{t_1}^{(1)}, \ldots, \alpha_{t_5}^{(1)})$, $(\alpha_{t_1}^{(2)}, \ldots, \alpha_{t_5}^{(2)})$ and $(\alpha_{t_1}^{(3)}, \ldots, \alpha_{t_5}^{(3)})$ observed at $n = 5$ dates $t_1 < \ldots < t_5$, with $k = 2$ lags, the response variables are stored in:

$$\mathbf{Y} = \begin{pmatrix} \alpha_{t_5}^{(1)} & \alpha_{t_5}^{(2)} & \alpha_{t_5}^{(3)} \\ \alpha_{t_4}^{(1)} & \alpha_{t_4}^{(2)} & \alpha_{t_4}^{(3)} \\ \alpha_{t_3}^{(1)} & \alpha_{t_3}^{(2)} & \alpha_{t_3}^{(3)} \end{pmatrix}$$

The predictors are stored in:

$$\mathbf{X} = \begin{pmatrix} \alpha_{t_4}^{(1)} & \alpha_{t_3}^{(1)} & \alpha_{t_4}^{(2)} & \alpha_{t_3}^{(2)} & \alpha_{t_4}^{(3)} & \alpha_{t_3}^{(3)} \\ \alpha_{t_3}^{(1)} & \alpha_{t_2}^{(1)} & \alpha_{t_3}^{(2)} & \alpha_{t_2}^{(2)} & \alpha_{t_3}^{(3)} & \alpha_{t_2}^{(3)} \\ \alpha_{t_2}^{(1)} & \alpha_{t_1}^{(1)} & \alpha_{t_2}^{(2)} & \alpha_{t_1}^{(2)} & \alpha_{t_2}^{(1)} & \alpha_{t_1}^{(1)} \end{pmatrix}$$

It is also possible to add other regressors to $\mathbf{X}$, such as dummy variables, or indicators of special events. In this situation, and as discussed in [Ext+16], we can avoid the constraining of these dummy variables, in a Kernel ridge regression with unpenalized terms. Here, we consider only the inclusion of the observed time series' lags in the model.

B.2.2 Sensitivity of the response to a change in the covariates

Here, the response is the matrix $\mathbf{Y}$ described in the previous section. Thus, we are deriving the sensitivity of level, slope, and curvature, to the changes in their associated lags.

We let r^2 be:

$$r^2 := ||\mathbf{x}_i - \mathbf{x}_k||_2^2 = (\mathbf{x}_i - \mathbf{x}_k)^T(\mathbf{x}_i - \mathbf{x}_k)$$

Where $\mathbf{x}_i$ is the i^{-th} line of matrix $\mathbf{X}$. Hence:

$$\frac{\partial r^2}{\partial \mathbf{x}_k^{(j_0)}} = -2\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)$$

And:

$$\frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}} = \frac{1}{2r}\frac{\partial r^2}{\partial \mathbf{x}_k^{(j_0)}} = -\frac{1}{r}\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)$$

As a consequence:

- For the **Gaussian kernel**

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto \sigma^2 exp\left(-\frac{r^2}{2l^2}\right)$$

We have:

$$\begin{aligned}
\frac{\partial K_{Gauss}(\mathbf{x}_i, \mathbf{x}_k)}{\partial \mathbf{x}_k^{(j_0)}} &= -\frac{\sigma^2}{2l^2} exp\left(-\frac{r^2}{2l^2}\right) \frac{\partial r^2}{\partial \mathbf{x}_k^{(j_0)}} \\
&= -\frac{\sigma^2}{2l^2} exp\left(-\frac{r^2}{2l^2}\right) \left[-2\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)\right] \\
&= \frac{\sigma^2}{l^2} exp\left(-\frac{r^2}{2l^2}\right) \left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right) \\
&= \frac{\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)}{l^2} K_{Gauss}(\mathbf{x}_i, \mathbf{x}_k)
\end{aligned}$$

- For the **Matérn 3/2 kernel**

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto \sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) exp\left(-\frac{\sqrt{3}r}{l}\right)$$

We have:

$$\begin{aligned}
\frac{\partial K_{3/2}(\mathbf{x}_i, \mathbf{x}_k)}{\partial \mathbf{x}_k^{(j_0)}} &= \sigma^2 \frac{\sqrt{3}}{l} exp\left(-\frac{\sqrt{3}}{l}r\right) \left[\frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}} - \frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}} \left(1 + \frac{\sqrt{3}}{l}r\right)\right] \\
&= \sigma^2 \frac{\sqrt{3}}{l} \frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}} exp\left(-\frac{\sqrt{3}}{l}r\right) \left[1 - \left(1 + \frac{\sqrt{3}}{l}r\right)\right] \\
&= \sigma^2 \frac{\sqrt{3}}{l} exp\left(-\frac{\sqrt{3}}{l}r\right) \frac{\sqrt{3}\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)}{l}
\end{aligned}$$

- For the **Matérn 5/2 kernel**

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) exp\left(-\frac{\sqrt{5}r}{l}\right)$$

We have:

$$
\begin{aligned}
\frac{\partial K_{5/2}(\mathbf{x}_i, \mathbf{x}_k)}{\partial \mathbf{x}_k^{(j_0)}} &= \sigma^2 exp\left(-\frac{\sqrt{5}}{l}r\right)\left[\left(\frac{\sqrt{5}}{l}\frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}} + \frac{5}{3l^2}\frac{\partial r^2}{\partial \mathbf{x}_k^{(j_0)}}\right) - \frac{\sqrt{5}}{l}\frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}}\left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right)\right] \\
&= \sigma^2 exp\left(-\frac{\sqrt{5}}{l}r\right)\left[\frac{5}{3l^2}\frac{\partial r^2}{\partial \mathbf{x}_k^{(j_0)}} - \frac{\sqrt{5}}{l}\frac{\partial r}{\partial \mathbf{x}_k^{(j_0)}}\left(\frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right)\right] \\
&= \sigma^2 exp\left(-\frac{\sqrt{5}}{l}r\right)\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)\left[-2\frac{5}{3l^2} + \frac{1}{r}\frac{\sqrt{5}}{l}\left(\frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right)\right] \\
&= \sigma^2 exp\left(-\frac{\sqrt{5}}{l}r\right)\frac{5}{3l^2}\left(1 + \frac{\sqrt{5}r}{l}\right)\left(\mathbf{x}_i^{(j_0)} - \mathbf{x}_k^{(j_0)}\right)
\end{aligned}
$$

From these expressions of the sensitivities, we can derive the average marginal effect of a covariate on the response, as demonstrated in equation B.3. These formulas will be valid in the DNS framework described in section B.2, where only one length-scale l parameter is required. Similar types of formulas could be derived in the KRLS framework from section , but they would include two length-scale parameters.

B.3 Description of the KRLS model applied to observed dates and time to maturities

B.3.1 Description of the model

In this other framework, we consider that the response variable is the spot interest rate observed at time t, for time to maturity τ, $R(t, \tau)$. The predictors are the observation date, and the time to maturity.

In this setting, we use the following weighted distance between the vectors (t, τ) in the Gaussian, Matérn 3/2 and Matérn 5/2 kernels:

$$
r = \sqrt{\frac{(t_i - t_j)^2}{l_1^2} + \frac{(\tau_i - \tau_j)^2}{l_2^2}}
$$

So that here, the spot rates values are explained as linear combination of distances between vectors of time to maturities and observation dates (t_i, τ_i) and (t_j, τ_j). In this setting, if we consider 10 spot rates observed at 2 dates $t_1 < t_2$ and 5 time to maturities $\tau_1, \ldots, \tau_5$, the response variable is:

$$\mathbf{Y} = (R(t_1, \tau_1), \ldots, R(t_1, \tau_5), R(t_2, \tau_1), \ldots, R(t_2, \tau_5))^T$$

and the predictors are

$$\mathbf{X} = \begin{pmatrix} \tau_1 & t_1 \\ \vdots & \vdots \\ \tau_5 & t_1 \\ \tau_1 & t_2 \\ \vdots & \vdots \\ \tau_5 & t_2 \end{pmatrix}$$

If some new observations arrive at time t_3 in the model, these new observations will be stored in:

$$\mathbf{X}^* = \begin{pmatrix} \tau_1 & t_3 \\ \vdots & \vdots \\ \tau_5 & t_3 \end{pmatrix}$$

In this other setting, it is also possible to add other regressors such as dummy variables, or indicators of special events. Again, and as suggested in the previous section and in [Ext+16], we can avoid the constraining of these dummy variables, in a Kernel ridge regression with unpenalized terms.

For example, if we wanted to add another indicator $(I_t)_t$ observed at times $t_1 < t_2$, we would have to consider the following matrix of predictors:

$$\mathbf{X} = \begin{pmatrix} \tau_1 & t_1 & I_{t_1} \\ \vdots & \vdots & \vdots \\ \tau_5 & t_1 & I_{t_1} \\ \tau_1 & t_2 & I_{t_2} \\ \vdots & \vdots & \vdots \\ \tau_5 & t_2 & I_{t_2} \end{pmatrix}$$

And another weighted distance in the Gaussian, Matérn 3/2 and Matérn 5/2 kernels, taking into account the new indicator $(I_t)_t$:

$$r = \sqrt{\frac{(t_i - t_j)^2}{l_1^2} + \frac{(\tau_i - \tau_j)^2}{l_2^2} + \frac{(I_{t_i} - I_{t_j})^2}{l_3^2}}$$

B.3.2 Sensitivity of the spot rates to a change in observation date and time to maturity

In this framework, and as mentioned in the previous section, we consider the following measure of similarity/dissimilarity between $\mathbf{x}_i = (t_i, \tau_i)$ and $\mathbf{x}_k = (t_k, \tau_k)$:

$$r^2 = \frac{(t_i - t_k)^2}{l_1^2} + \frac{(\tau_i - \tau_k)^2}{l_2^2}$$

The associated kernels are the following ones:

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto K_{Gauss}(r) = \sigma^2 exp\left(-\frac{r^2}{2}\right)$$

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto K_{Matérn3/2}(r) = \sigma^2 \left(1 + \sqrt{3}r\right) exp\left(-\sqrt{3}r\right)$$

and

$$(\mathbf{x}_i, \mathbf{x}_k) \mapsto K_{Matérn5/2}(r) = \sigma^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) exp\left(-\sqrt{5}r\right)$$

It is possible to obtain the sensitivity of the spot rates to a change in the observation date, $\frac{\partial R(t_k,\tau)}{\partial t_k}$ for any τ. Even if it is actually very difficult to predict in which direction the spot rates will move, this type of indicators could still serve as decision-assistance tools for risk management. We have:

$$\frac{\partial r^2}{\partial t_k} = -\frac{2(t_i - t_k)}{l_1^2}$$

and

$$\frac{\partial r}{\partial t_k} = -\frac{1}{r}\frac{(t_i - t_k)}{l_1^2}$$

Thus, we have:

$$\begin{aligned}\frac{\partial K_{Gauss}(r)}{\partial t_k} &= -\frac{\sigma^2}{2} exp\left(-\frac{r^2}{2}\right)\frac{\partial r^2}{\partial t_k} \\ &= \sigma^2 exp\left(-\frac{r^2}{2}\right)\frac{(t_i - t_k)}{l_1^2}\end{aligned}$$

$$\begin{aligned}\frac{\partial Matérn3/2}{\partial t_k} &= \sigma^2\sqrt{3}\frac{\partial r}{\partial t_k} exp\left(-\sqrt{3}r\right)\left[1-(1+\sqrt{3}r)\right] \\ &= -3\sigma^2 r exp\left(-\sqrt{3}r\right)\frac{\partial r}{\partial t_k} \\ &= 3\sigma^2 exp\left(-\sqrt{3}r\right)\frac{(t_i - t_k)}{l_1^2}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial Matérn5/2}{\partial t_k} &= \sigma^2 exp\left(-\sqrt{5}r\right)\left[\left(\sqrt{5}\frac{\partial r}{\partial t_k} + \frac{5}{3}\frac{\partial r^2}{\partial t_k}\right) - \sqrt{5}\frac{\partial r}{\partial t_k}\left(1+\sqrt{5}r+\frac{5}{3}r^2\right)\right] \\ &= \sigma^2 exp\left(-\sqrt{5}r\right)\left[\frac{5}{3}\frac{\partial r^2}{\partial t_k} - \sqrt{5}\frac{\partial r}{\partial t_k}\left(\sqrt{5}r+\frac{5}{3}r^2\right)\right] \\ &= \sigma^2 exp\left(-\sqrt{5}r\right)\left[-\frac{10}{3}\frac{(t_i - t_k)}{l_1^2} + \frac{\sqrt{5}}{r}\frac{(t_i - t_k)}{l_1^2}\left(\sqrt{5}r+\frac{5}{3}r^2\right)\right] \\ &= \sigma^2 exp\left(-\sqrt{5}r\right)\frac{5}{3}\left[1+\sqrt{5}r\right]\frac{(t_i - t_k)}{l_1^2}\end{aligned}$$

In the next section, B.4, we present the results obtained by the DNS-KRLS model from section B.2 and the KRLS model from section B.3 on a training/testing dataset[1].

B.4 Numerical examples

In this section, we present the results obtained by the DNS-KRLS model from section B.2 and the KRLS model from section B.3. The examples are not exhaustive benchmarks, but aim at illustrating the forecasting capabilities of the models.

[1] for a more complete treatment, a validation set would be added, in order to verify that the models are not 'overtrained' on this training/testing set

We use calibrated discount rates data from Deutsche Bundesbank website, observed on a monthly basis, from the beginning of 2002 to the end 2015. There are 167 curves, observed at 50 maturities in the dataset. Only 15 time to maturities are used in these examples (in years): $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20, 25, 30$.

In figure B.2, we present the data that we use, and table B.1 contains a summary of these data; the minimum, maximum, median, first and third quartiles of the discount rates observed at given maturities. There are alternate cycles of increases and decreases of the discount rates, with generally a decreasing trend. Some of the discount rates, at the most recent dates, and lower maturities, are negative.

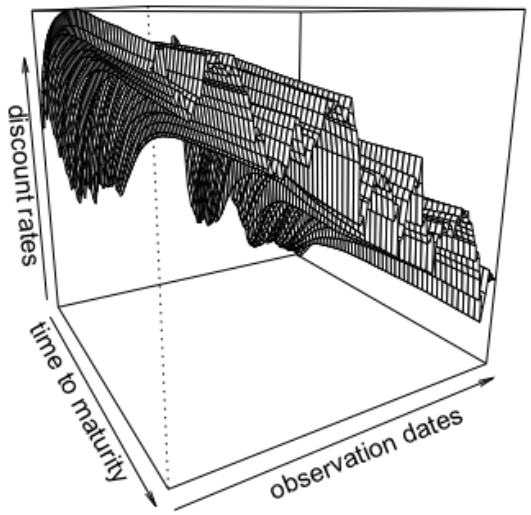

Fig. B.2.: Observed discount rates from Deutsche Bundesbank website, from 2002 to the end 2015

Tab. B.1.: Summary of observed discount rates from Deutsche Bundesbank website, from 2002 to the end 2015

Maturity	Min	1st Qrt	Median	3rd Qrt	Max
1	-0.116	0.858	2.045	3.072	5.356
5	0.170	1.327	2.863	3.807	5.146
15	0.711	2.616	3.954	4.702	5.758
30	0.805	2.594	3.962	4.814	5.784

A rolling forecasting methodology (see [Ber+15]) is implemented in order to obtain the benchmarks between the models. It is described in figure B.3. A fixed 12 months/36 months-length window for training the model, and the following 12 months/36 months for testing, the origin of the training set is then advanced of 1 month, and the training/testing procedure is repeated. The measure of forecasting performance is the Root Mean Squared Error ($RMSE$).

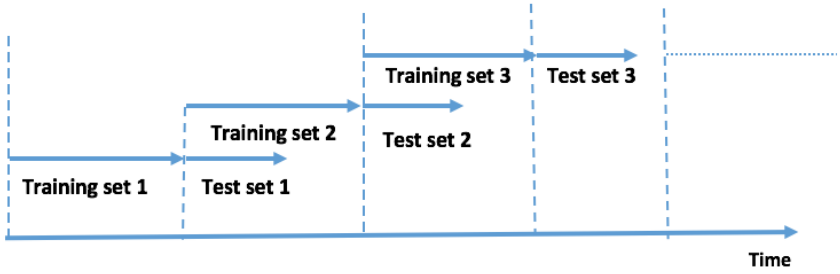

Fig. B.3.: rolling forecasting cross-validation sets

We use similar grids for all the models, in order to ease the comparability of the results, and avoid too much manual tweaking of the hyperparameters and overtraining the available data.

Hence for both models, DNS-KRLS (from section B.2) and KRLS (from section B.3), we consider 5 values of σ (variance parameter), l, l_1, l_2 (length-scale parameters for Gaussian, Matérn 3/2 and Matérn 5/2 kernels) and λ (the regularization parameter for all the kernels) regularly spaced between $[10^{-2}, 10^{2}]$: $0.01, 0.1, 1, 10, 100$.

For the additional parameter η in the DNS-KRLS model, we use 5 values comprised (regularly spaced) between the minimum of the observed time to maturities and the maximum of the observed time to maturities (on $[1, 30]$): $1, 8.25, 15.5, 22.75, 30$.

B.4.1 Cross-validation results

The results obtained after the cross-validation procedure are reported in table B.2 and B.3. The results obtained by considering an automatic ARIMA modeling ([HK08]) of the three time series are also indicated, to serve as a benchmark.

For the DNS model and all the types of kernels (Gaussian, Matérn 3/2 and Matérn 5/2), the *optimal* number of lags is respectively equal to 5 and 6 for the 12-months

Tab. B.2.: Average out-of-sample RMSE for training set length = 12 months and test set length = 12 months

Model	σ	l	l_1	l_2	λ	η	**RMSE**
Gaussian	10	-	0.01	10	10	-	0.5839150
Matérn 3/2	100	-	1	100	0.01	-	**0.5136373**
Matérn 5/2	1	-	0.01	10	0.1	-	0.5781184
DNS-Gaussian	0.1	10	-	-	0.01	15.5	0.6041652
DNS-Matérn 3/2	100	1	-	-	0.01	22.75	**0.6038667**
DNS-Matérn 5/2	100	1	-	-	0.01	15.5	0.6041580
DNS-ARIMA	-	-	-	-	-	1	0.6751660

Tab. B.3.: Average out-of-sample RMSE for training set length = 36 months and test set length = 36 months

Model	σ	l	l_1	l_2	λ	η	**RMSE**
Gaussian	10	-	0.1	10	100	-	1.170690
Matérn 3/2	100	-	1	10	0.01	-	**1.003246**
Matérn 5/2	10	-	10	100	10	-	1.134833
DNS-Gaussian	0.01	0.01	-	-	0.01	30	1.264533
DNS-Matérn 3/2	0.01	0.01	-	-	0.01	30	**1.264533**
DNS-Matérn 5/2	0.01	0.01	-	-	0.01	30	1.264533
DNS-ARIMA	-	-	-	-	-	1	1.281937

and 36-months horizons. The last 12 months and 36 months of the data are respectively considered as training sets for obtaining these graphs.

We observe that no matter the length of the training/testing window (either 12 months or 36 months), or the method employed (either DNS-KRLS or KRLS), the Matérn 3/2 kernel performs better than the other models. It is even performing better on this specific problem with a KRLS model, considering the similarities between vectors of time to maturities and observation dates. The other kernels are probably too flexible for the purpose, so that they are both overfitting the data a bit.

In the next sections B.4.2 and B.4.3, we examine these results further, by looking at the out-of-sample root mean squared error (RMSE) obtained over time, and the projected discount/discount factors'/discrete forward curves obtained with the optimal parameters.

B.4.2 Out-of-sample RMSE over time

The following figures, B.4 and B.5, present the out-of-sample RMSE obtained over time by all the KRLS models, for horizons equal to 12 and 36. On figure B.4 (on the right), we observe that despite being the best model on average, the Matérn 3/2 is not necessarily the best during the liquidity/credit crunch (especially in 2008) for an horizon of 36 months. On figure B.5, we observe that all the kernels lead to very similar results, with only slight differences. During the crisis, and for an horizon of 36 months, despite displaying more volatile and less accurate results elsewhere, the DNS-ARIMA is performing better than the other models.

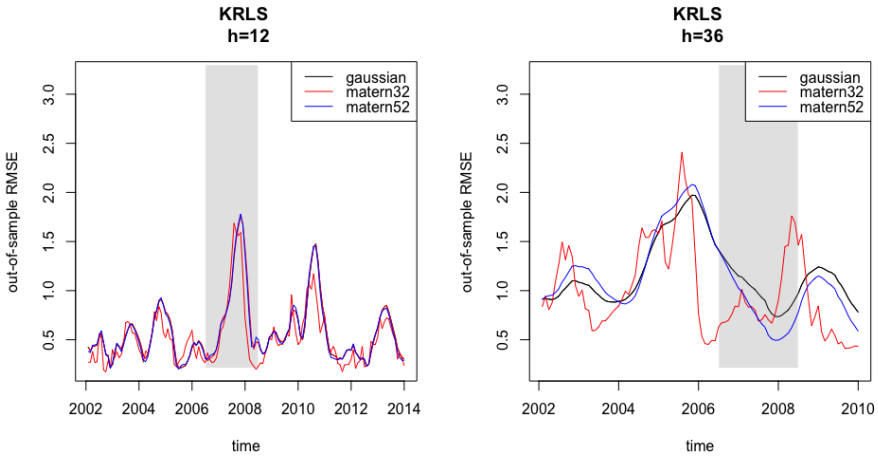

Fig. B.4.: Out-of-sample RMSE over time for KRLS models, with horizon = 12 and horizon = 36

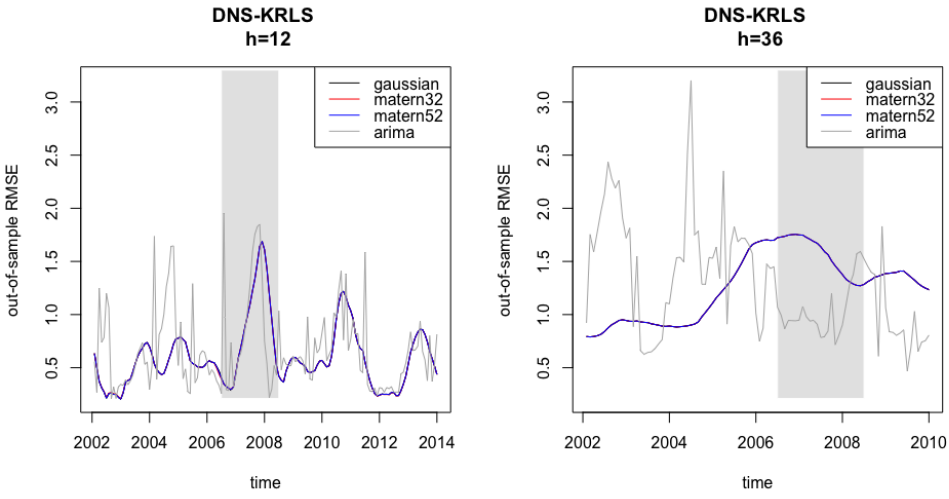

Fig. B.5.: Out-of-sample RMSE over time for DNS-KRLS, with horizon = 12 and horizon = 36

The following boxplots provide a more detailed description of the distribution of the out-of-sample error, and serves as a complement for tables B.2 and B.3 (one can also

report to appendix B.6.1 for the details on these boxplots). Notably, it confirms the high variance and low median results observed for Matérn 3/2 and DNS-ARIMA for an horizon equal to 36 months. All the other results are otherwise pretty close to each other.

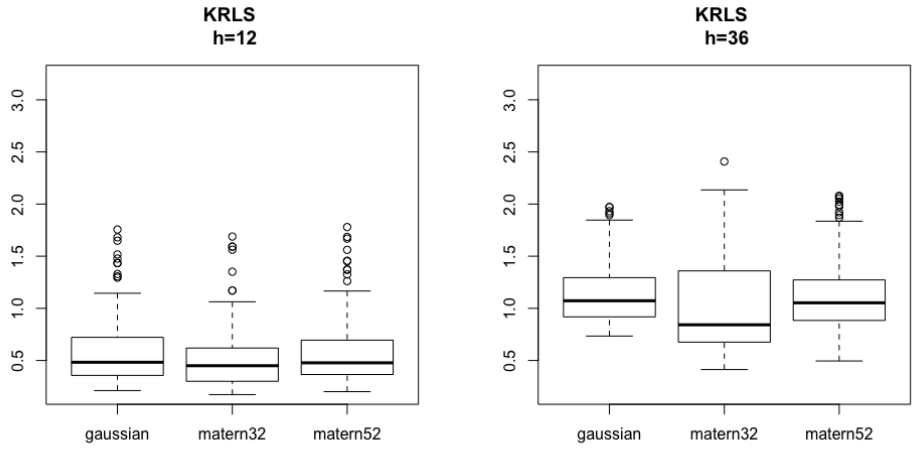

Fig. B.6.: Boxplots of Out-of-sample RMSE over time, for KRLS models

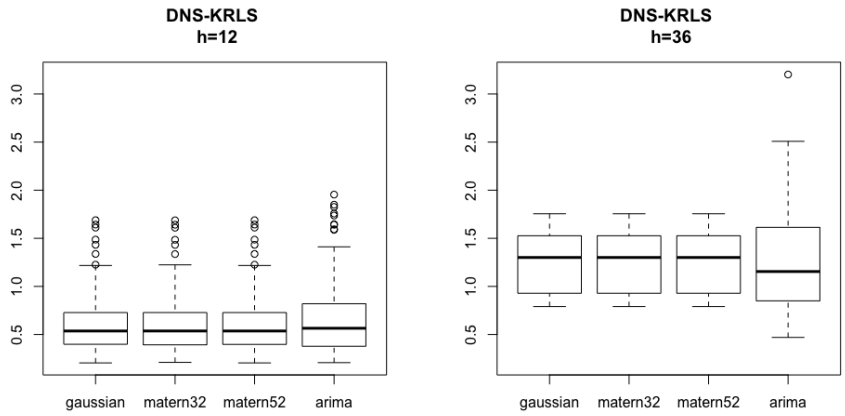

Fig. B.7.: Boxplots of Out-of-sample RMSE over time, for DNS-KRLS models

B.4.3 Implied forecast term-structure of discrete forward rates

The following figure, B.8, presents the 12-months ahead and 36-months ahead ($h = 12$ and $h = 36$) forecasts obtained in the KRLS framework, by considering a Matérn 3/2 kernel. Similarly, figure B.9, presents the 12-months ahead and 36-months ahead ($h = 12$ and $h = 36$) forecasts obtained in the DNS-KRLS framework, by considering a Matérn 3/2 kernel for forecasting the factors.

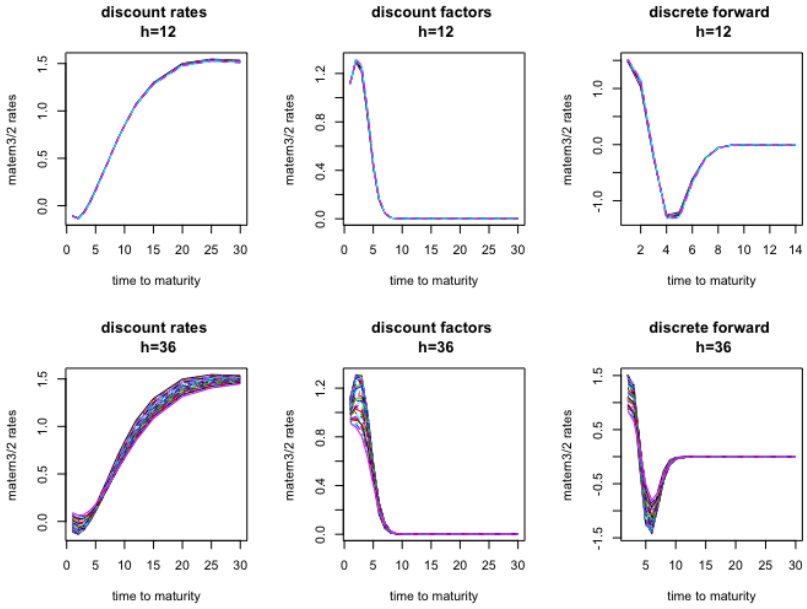

Fig. B.8.: Forecasts of discount rates, discount factors and discrete forward rates for Matérn 3/2 for horizon = 12 and horizon = 36

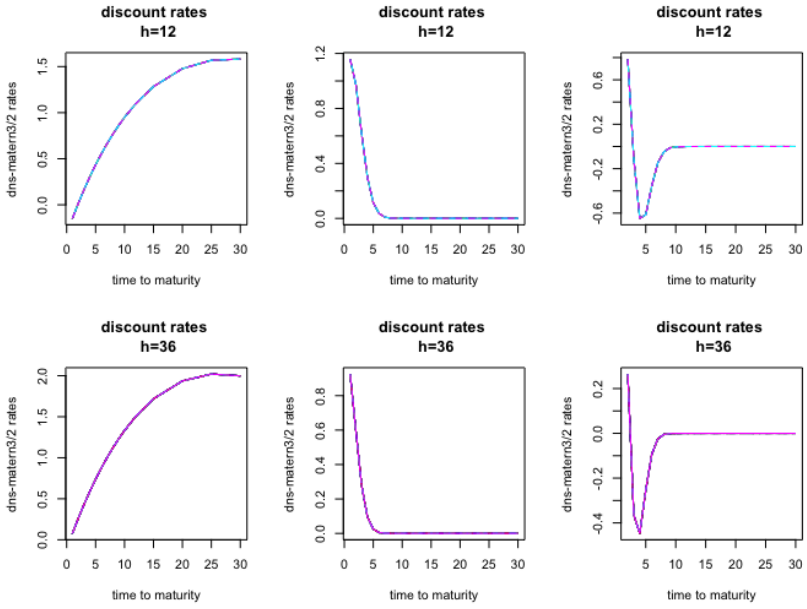

Fig. B.9.: Forecasts of discount rates, discount factors and discrete forward rates for DNS-Matérn 3/2 for horizon = 12 and horizon = 36

These (figures B.8 and B.9) can be compared to a more familiar model, the one in which the level, slope, and curvature are modelled separately with an ARIMA model in a DNS framework (see figure B.10).

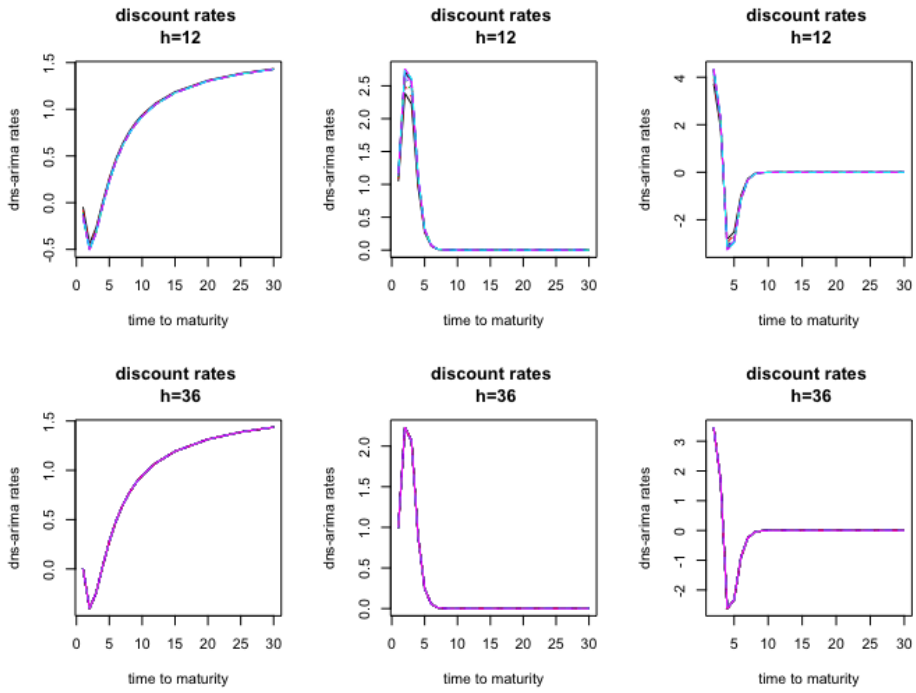

Fig. B.10.: discount rates, discount factors and discrete forward rates for DNS-ARIMA

We observe that the discount (rates and factors) and forward curves obtained by each model do, indeed, exhibit the same patterns as actual market discount and discrete forward curves. In particular, for the projected negative rates, we observe projected discount factors that are greater than 1.

B.5 Conclusion

In this chapter, we discussed the forecasting of discount curves, by using Kernel Regularized Least Squares (KRLS). The KRLS techniques are capable of learning nonlinear response variables, by taking into account various and complex types of covariance structures between the predictors. The response must have a relatively low number of examples, though, because of the quadratic cost of computing the kernels.

The model is highly interpretable, as it explains the responses as linear combinations of similarities/dissimilarities between the examples. Some sentivity indicators of the response as a function of the predictors are derived, and they could consitute useful decision-assistance tools.

Two types of KRLS models are considered here, specifically for the discount curves. One relying on the famous Dynamic Nelson-Siegel (DNS) framework, and another one explaining the spot rates as a function of the time to maturity and date of observation. Both types of KRLS models deliver some robust forecasts of the discount curves, as the kernels hyperparameters implicitly constrain the model to reproduce the Yield Curve's stylized facts.

B.6 Appendix

B.6.1 Summary of out-of-sample errors for all the models (in %)

KRLS, h = 12 (cf. figure B.6)

```
gaussian          matern32          matern52
Min.   :0.2114   Min.   :0.1726   Min.   :0.2012
1st Qu.:0.3582   1st Qu.:0.3009   1st Qu.:0.3669
Median :0.4828   Median :0.4498   Median :0.4777
Mean   :0.5839   Mean   :0.5136   Mean   :0.5781
3rd Qu.:0.7216   3rd Qu.:0.6158   3rd Qu.:0.6914
Max.   :1.7562   Max.   :1.6886   Max.   :1.7795
```

KRLS, h = 36 (cf. figure B.6)

```
  gaussian          matern32          matern52
Min.   :0.7343   Min.   :0.4130   Min.   :0.4953
1st Qu.:0.9193   1st Qu.:0.6775   1st Qu.:0.8860
Median :1.0727   Median :0.8422   Median :1.0529
Mean   :1.1707   Mean   :1.0032   Mean   :1.1348
3rd Qu.:1.2824   3rd Qu.:1.3548   3rd Qu.:1.2639
Max.   :1.9723   Max.   :2.4080   Max.   :2.0797
```

DNS-KRLS, h = 12 (cf. figure B.7)

```
gaussian          matern32          matern52           arima
Min.   :0.2048   Min.   :0.2103   Min.   :0.2048   Min.   :0.2077
1st Qu.:0.4016   1st Qu.:0.3934   1st Qu.:0.3997   1st Qu.:0.3779
Median :0.5366   Median :0.5368   Median :0.5366   Median :0.5649
Mean   :0.6042   Mean   :0.6039   Mean   :0.6042   Mean   :0.6752
3rd Qu.:0.7281   3rd Qu.:0.7277   3rd Qu.:0.7281   3rd Qu.:0.8190
Max.   :1.6884   Max.   :1.6870   Max.   :1.6884   Max.   :1.9544
```

DNS-KRLS, h = 36 (cf. figure B.7)

```
gaussian          matern32          matern52           arima
```

```
 Min.   :0.7900   Min.   :0.7900   Min.   :0.7900   Min.   :0.4698
 1st Qu.:0.9294   1st Qu.:0.9294   1st Qu.:0.9294   1st Qu.:0.8527
 Median :1.3005   Median :1.3005   Median :1.3005   Median :1.1545
 Mean   :1.2645   Mean   :1.2645   Mean   :1.2645   Mean   :1.2819
 3rd Qu.:1.5158   3rd Qu.:1.5158   3rd Qu.:1.5158   3rd Qu.:1.6033
 Max.   :1.7549   Max.   :1.7549   Max.   :1.7549   Max.   :3.2024
```

B.6.2 Summary of KRLS Matérn 3/2 and DNS-ARIMA forecasts (in %) for horizon = 12 and horizon = 36

KRLS Matérn 3/2

horizon = 12 (cf. figure B.8)

```
1y                 5y                10y               30y
 Min.   :-0.1135   Min.   :0.1538   Min.   :0.8424   Min.   :1.509
 1st Qu.:-0.1105   1st Qu.:0.1588   1st Qu.:0.8441   1st Qu.:1.514
 Median :-0.1091   Median :0.1636   Median :0.8456   Median :1.520
 Mean   :-0.1097   Mean   :0.1632   Mean   :0.8453   Mean   :1.520
 3rd Qu.:-0.1084   3rd Qu.:0.1680   3rd Qu.:0.8468   3rd Qu.:1.526
 Max.   :-0.1078   Max.   :0.1708   Max.   :0.8471   Max.   :1.533
```

horizon = 36 (cf. figure B.8)

```
1y                  5y                10y               30y
 Min.   :-0.11114   Min.   :0.1066   Min.   :0.6661   Min.   :1.449
 1st Qu.:-0.08006   1st Qu.:0.1120   1st Qu.:0.6823   1st Qu.:1.472
 Median :-0.02946   Median :0.1273   Median :0.7121   Median :1.495
 Mean   :-0.02263   Mean   :0.1324   Mean   :0.7262   Mean   :1.494
 3rd Qu.: 0.03020   3rd Qu.:0.1490   3rd Qu.:0.7619   3rd Qu.:1.517
 Max.   : 0.09165   Max.   :0.1804   Max.   :0.8374   Max.   :1.535
```

DNS-ARIMA

horizon = 12 (cf. figure B.10)

```
1y                  5y                10y               30y
 Min.   :-0.15142   Min.   :0.2295   Min.   :0.9237   Min.   :1.429
 1st Qu.:-0.14819   1st Qu.:0.2305   1st Qu.:0.9242   1st Qu.:1.429
```

```
Median :-0.13914   Median :0.2333   Median :0.9256   Median :1.429
Mean   :-0.12468   Mean   :0.2379   Mean   :0.9279   Mean   :1.430
3rd Qu.:-0.11405   3rd Qu.:0.2412   3rd Qu.:0.9296   3rd Qu.:1.431
Max.   :-0.04527   Max.   :0.2628   Max.   :0.9405   Max.   :1.434
```

horizon = 36 (cf. figure B.10)

```
1y                 5y                10y               30y
Min.   :0.003647   Min.   :0.2782   Min.   :0.9482   Min.   :1.437
1st Qu.:0.003647   1st Qu.:0.2782   1st Qu.:0.9482   1st Qu.:1.437
Median :0.003647   Median :0.2782   Median :0.9482   Median :1.437
Mean   :0.003647   Mean   :0.2782   Mean   :0.9482   Mean   :1.437
3rd Qu.:0.003647   3rd Qu.:0.2782   3rd Qu.:0.9482   3rd Qu.:1.437
Max.   :0.003647   Max.   :0.2782   Max.   :0.9482   Max.   :1.437
```

C

A Bayesian Quasi-Randomized neural network, and its application to the optimization of black box functions

C.1 Introduction

In this chapter, we present a Bayesian Quasi-Random Vector Functional Link (BQRVFL) neural network model, with one hidden layer. It is a hybrid penalized regression/neural networks model on an augmented data set. In this hybrid regression model, we assume that the regression parameters are governed by a prior distribution, and that the hidden layer's nodes are (quasi-)randomized. As a prior distribution for the regression parameters, we use a multivariate Gaussian distribution, and for the simulation of the nodes in the hidden layer, we use a Sobol sequence.

To the best of our knowledge, randomized neural networks were introduced by [Sch+92], and the Random Vector Functional Link neural networks (RVFL) were introduced by [Pao+94]. RVFL networks are *multilayer feedforward* neural networks, in which there is a *direct link* between the predictors and the output variable, aiming at capturing the linear relationships. And in addition to the *direct link*, there are new features, the hidden nodes (the dataset is augmented).

The RVFL networks have been successfully applied to solving different types of classification and regression problems; see for example [DC10]. Here, we will use RVFL networks with only one hidden layer.

With the BQRVFL model presented here, it is possible to obtain predictions from a nonlinear model (actually, a combination of a linear and a nonlinear model), along with confidence intervals around the model's predictions. The choice of this relatively simple Gaussian prior should not, however, prevent the user from checking the confidence intervals around the predictions.

There are other types of models, which are also capable of achieving this purpose - obtain predictions from a nonlinear model, along with confidence intervals around the model's predictions - like Gaussian process regression models [RW06] and random forests models with confidence intervals ([Wag+14]). Depending on the problem at hand, and the desire from the user for a higher accuracy and/or interpretability, one of these alternative models could also be considered.

In section C.2, we present the BQRFVL model, with one hidden layer. Formulas for the estimation of its parameters are given, along with formulas for the calculation of confidence intervals around the predictions. The model's predictions on a validation set, and on simulated data from [Sap+14] are also presented.

In section C.3, the model is used as a workhorse for Bayesian optimization [Moc+78]. This type of optimization methods are useful for finding minima or maxima of black box functions, whose evaluations are expensive, and gradients are not necessarily available in a closed form. It has been shown to be very effective on challenging optimization functions ([Jon01] and [Sno+12]).

The idea of Bayesian optimization is to optimize an alternative, cheaper function called the acquisition function, rather than the main, expensive objective function. For doing this, the uncertainty around the predictions of an alternative machine learning model - called the surrogate model - is used in a way that will be described in more details in section C.3.

The surrogate model's posterior distribution tries to approximate the objective function in the best way. And this posterior distribution is enhanced, as more points of the objective function are evaluated. Gaussian process regression models are often used are surrogate models ([Sno+12] for example). Here, the surrogate model will be the BQRVFL model presented in C.2.

We apply Bayesian optimization using the BQRVFL as a surrogate model, to the minimization of the cross-validation error (produced on discount curve forecasting) of four machine learning models. One of these models is based on an RVFL applied to multivariate time series forecast ([Mou+18]). The second one applies Kernel Regularized Least Squares (see chapter B) (KRLS hereafter) directly to the curves' observation dates and time to maturities. The two remaining models are based on the popular DNS framework ([DL06]), and are a DNS-KRLS and a DNS-RVFL (more details here or refer to another section).

C.2 Description of the model

C.2.1 Estimation of the parameters and confidence intervals

We consider a response variable $y \in \mathbb{R}^n$ that has to be explained as a function of p predictors stored in a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. The data set $\mathbf{X}$ is augmented by a set of new predictors:

$$g\left(\mathbf{X}W\right)$$

where g is an activation function, as in of neural networks models, and $W \in \mathbb{R}^{p \times L}$ are the nodes in the hidden layer, obtained from a deterministic sobol sequence of quasi-random numbers ([Nie92] and [BT97]). L is the number of nodes in the hidden layer.

The parameters W are used to combine the p columns of $\mathbf{X}$ into L new variables, hence taking into account the potential nonlinear relationships between the response y and the predictors $\mathbf{X}$ when passed through the activation function g. Having obtained a new set of predictors stored in columns in a matrix $\mathbf{Z} = [\mathbf{X}\ g\left(\mathbf{X}W\right)]$, we apply a Bayesian linear regression model, to explain y as a function $\mathbf{Z}\beta$ of the new $p + L$ predictors. We assume that y could be explained as:

$$y = \mathbf{Z}\beta + \epsilon$$

where $\beta \sim \mathcal{N}(0_{\mathbb{R}^{p+L}}, I_{p+L})$ are the parameters of the model that have to be estimated, and $\epsilon \sim \mathcal{N}(0_{\mathbb{R}^n}, \sigma^2 I_n)$ is the error term. It is possible to show, in a Bayesian linear regression setting (see [RW06] for example, for details) that for new observations arriving in the model and stored in matrix $\mathbf{Z}_*$, we have the following properties for the associated predictions:

$$y* \sim \mathcal{N}(\mathbf{Z}_* \mu_{\beta|y,\mathbf{Z}},\ \mathbf{Z}_* \Sigma_{\beta|y,\mathbf{Z}} \mathbf{Z}_*^T + \sigma^2 I_n)$$

with

$$\mu_{\beta|y,\mathbf{Z}} = \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T + \sigma^2 I_n\right)^{-1} y$$

and

$$\Sigma_{\beta|y,\mathbf{Z}} = I_n - \mathbf{Z}^T \left(\mathbf{Z}\mathbf{Z}^T + \sigma^2 I_n\right)^{-1} \mathbf{Z}$$

By using the Woodbury identity ([GVL12] and [Wel10]) for $\mathbf{P}$ and $\mathbf{R}$ positive definite

$$\left(\mathbf{P}^{-1} + \mathbf{B}^T \mathbf{R}^{-1} \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T \left(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R}\right)^{-1}$$

and by noting $\sigma^2 = \lambda$, this is equivalent to having the same result for y^*, but with:

$$\mu_{\beta|y,\mathbf{Z}} = \left(\mathbf{Z}^T\mathbf{Z} + \lambda I_p\right)^{-1}\mathbf{Z}^T y$$

and

$$\Sigma_{\beta|y,\mathbf{Z}} = I_p - \left(\mathbf{Z}^T\mathbf{Z} + \lambda I_p\right)^{-1}\mathbf{Z}^T\mathbf{Z}$$

And depending on the situations in which $n << p$, or $p << n$, we can use one of the two previous expressions; the one resulting in the lowest number operations. In the sequel of the chapter, we will use a ReLU activation function, which is:

$$g : x \mapsto max(x, 0)$$

In the case where g is derivable (for example when $g : x \mapsto x$ or $g : x \mapsto sigmoid(x) = \frac{1}{1+exp(-x)}$ or $g : x \mapsto tanh(x)$), it is possible to obtain a sensitivity (at the first order) of the response to a change in the covariates. Indeed, in this case, we have for a fixed (tied to the observations) $i \in \{1, \ldots, n\}$:

$$y_i = \sum_{j=1}^{p} \alpha_j X_i^{(j)} + \sum_{k=1}^{L} \gamma_k g(X_i^T W^{(k)})$$

Where $W^{(k)}$ is the k^{th} column of W among L. And for a fixed (tied to the covariates) integer $j_0 \in \{1, \ldots, p\}$, and $u := X_i^T W^{(k)}$, we have the following sensitivity of y_i:

$$\begin{aligned}\frac{\partial y_i}{\partial X_i^{(j_0)}} &= \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k \frac{\partial g}{\partial u}(u)\frac{\partial u}{\partial X_i^{(j_0)}}(u) \\ &= \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k W_{j_0}^{(k)} \frac{\partial g}{\partial u}(u)\end{aligned}$$

For $g : x \mapsto x$ for example, this leads to:

$$\frac{\partial y_i}{\partial X_i^{(j_0)}} = \alpha_{j_0} + \sum_{k=1}^{L} \gamma_k W_{j_0}^{(k)}$$

C.2.2 Examples on data from [Sap+14]

In order to present the performances of the model described in section C.2.2, we use four simulated randomized data sets, from [Sap+14], $SLC14_1$ and $SLC14_2$ with 2 different seeds each. These datasets are available in `R` package `caret`.

$SLC14_1$ consists of n rows, where n is a function parameter, and 20 columns. Its construction starts with the simulation of a $n \times 20$ matrix `dat`, containing random Gaussian numbers, with mean equal to zero and variance equal to 9. Each row of `dat` contains:

$$\mathbf{x} = (x_1, x_2, \ldots, x_{20})$$

The following transformation is applied to obtain each column of $SLC14_1$

$$\begin{aligned} \mathbf{x} \quad \mapsto \quad & x_1 + sin(x_2) + log(|x_3|) + x_4^2 + x_5 \times x_6 + I(x_7 \times x_8 \times x_9 < 0) + I(x_{10} > 0) \\ + \quad & x_{11} \times I(x_{11} > 0) + \sqrt{(}|x_{12}|) + cos(x_{13}) + 2 \times x_{14} + |x_{15}| \\ + \quad & I(x_{16} < -1) + x_{17} \times I(x_{17} < -1) - 2 \times x_{18} - x_{19} \times x_{20} + \epsilon \end{aligned}$$

where $\epsilon \sim \mathcal{N}(0, 9)$. The same type of idea is applied for obtaining the dataset $SLC14_2$. The construction of $SLC14_2$ begins with a matrix `dat` having n rows and 200 columns. The distribution of the random numbers within `dat` is a Gaussian distribution with mean equal to 0, and variance equal to 16, and the transformation applied to the columns of `dat` is:

$$\mathbf{x} \mapsto -1 + log(|x_1|) + \ldots + log(|x_{200}|) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 25)$.

For each seed, `123` and `456`, and $n = 250$, we separate the datasets $SLC14_1$ and $SLC14_2$ into a training/testing test, and a validation set. The training/testing set contains 70% of the whole data (175 observations), and the test set contains the remaining 30% of the data (75 observations).

A 5-fold cross-validation repeated 10 times is applied to the training/testing set, in order to choose some *optimal* hyperparameters for the BQRVL model (the number of nodes in the hidden layer and the regularization parameter λ).

The BQRVFL model with these *optimal* hyperparameters is applied to predict values on the validation set. In figures C.1, C.2, C.3, and C.4 we present the validation sets' observations (in black), along with the out-of-sample predictions on the validation set (in blue), and the 80%, 95% confidence intervals around the predictions (gray shaded regions).

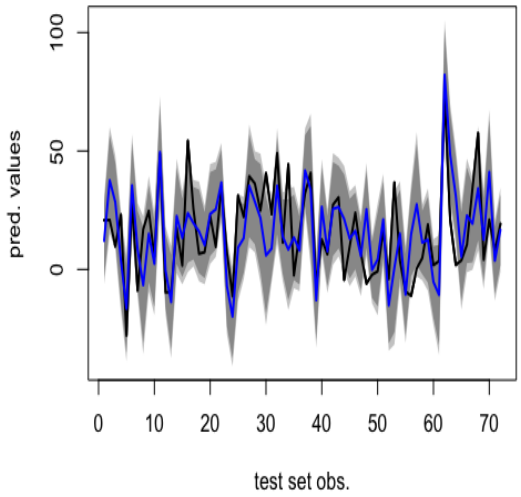

Fig. C.1.: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.1 (with `seed = 123`)

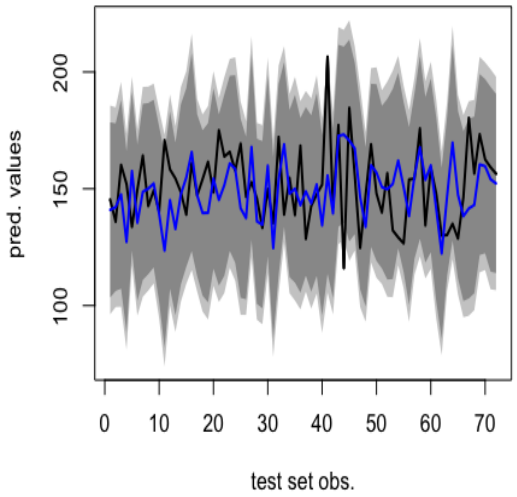

Fig. C.2.: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.2 (with `seed = 123`)

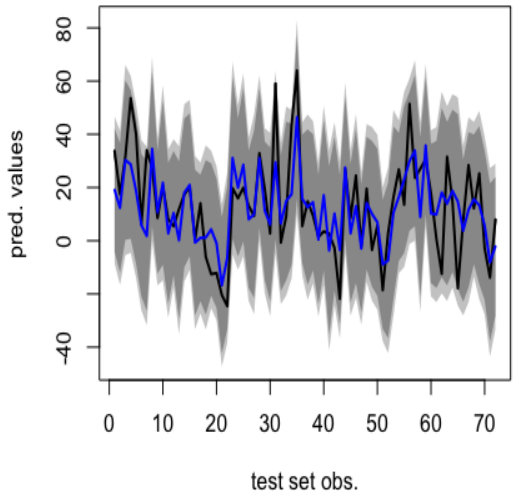

Fig. C.3.: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.1 (with `seed = 456`)

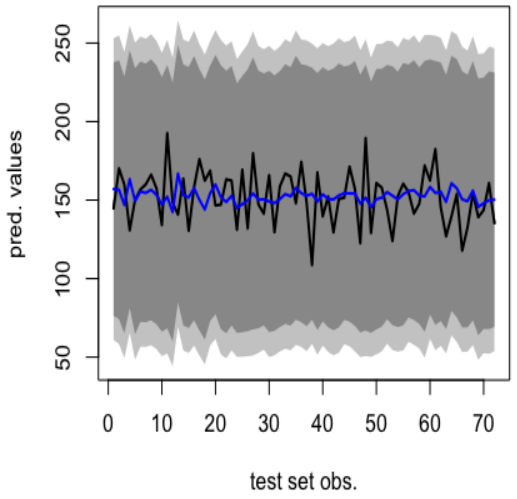

Fig. C.4.: Out-of-sample predictions and 80%, 95% confidence intervals for SLC14.2 (with `seed = 456`)

We observe that the confidence intervals often contain the validation set value, with a few exceptions. The following table C.1 contain the *optimal* (based on a grid search) parameters found for each dataset, with a ReLU activation function for the BQRVFL, and the proportions of points of the validation set contained into the 80%, 95% confidence intervals:

Tab. C.1.: optimal hyperparameters found for the BQRVFL on $SLC14_1$ and $SLC14_2$

Dataset	Number of nodes	λ	Prop at 95%	Prop at 80%
$SLC14_1$ - `seed = 123`	975	1.53	86.11%	77.77%
$SLC14_1$ - `seed = 456`	700	132.19	100.00%	97.22%
$SLC14_2$ - `seed = 123`	825	20	97.22%	94.44%
$SLC14_2$ - `seed = 456`	650	1788.65	100.00%	100.00%

In the next section, C.3, we are going to use this model as a workhorse for Bayesian optimization.

C.3 Bayesian optimization of black box functions

C.3.1 Description of the method

The optimization problem is about finding $\mathbf{x}^* \in \mathcal{C} \subseteq \mathbb{R}^k$, so that:

$$\mathbf{x}^* = ArgMin_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x})$$

f is the objective function, whose evaluations are expensive to calculate, and gradients are not necessarily available in closed-form. Methods based on gradient descent, or requiring to evaluate f several times will hence be inefficient for carrying out this task.

Bayesian optimization ([Moc+78] and [Jon01]) will minimize an alternative, cheaper function called the acquisition function, instead of minimizing f directly. The acquisition function is based on the uncertainty around the predictions of a surrogate machine learning model, trying to approximate f. Here, we consider only two types of acquisition functions. If we denote by $\widetilde{f}(\mathbf{x}, \theta)$, the prediction obtained with the surrogate model (whose distribution depends on θ) on a point $\mathbf{x} \in \mathcal{C}$, they are defined as:

- The **Upper Confidence Bound** (UCB) acquisition function, (actually a Lower Confidence Bound in the case of minimization). The idea is to minimize:

  $$a_{UCB}(\mathbf{x}; \mu, \sigma) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x})$$

  κ is a tuning parameter, designed to balance between exploration and exploitation (more details on this). With $\kappa = 1.96$ for example, this looks like the

lower confidence bound of $\widetilde{f}(\mathbf{x}, \theta)$, at a risk level of 5%, when the surrogate model is $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$.

- The **Expected Improvement** (EI) acquisition function. If f^* is the current minimum value found after a few evaluations of f, we would like to maximize the expected improvement of the surrogate model's predictions over f^* :

$$a_{EI}(\mathbf{x}; \theta) = \mathbb{E}\left[max(\widetilde{f}(\mathbf{x}, \theta) - f^*, 0)\right]$$

In the case where the posterior distribution of the surrogate model is Gaussian, $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$, the acquisition function with Expected Improvement can be re-written as:

$$a_{EI}(\mathbf{x}) = \sigma(\mathbf{x})\left(\mu(\mathbf{x})\Phi\left(\gamma(\mathbf{x})\right) + \Phi^{'}\left(\gamma(\mathbf{x})\right)\right)$$

Where Φ is the probability distribution function of a $\mathcal{N}(0, 1)$, and $\Phi^{'}$ is its density function. We assume in the sequel of the paper that the predictions of all the surrogate models are drawn from a multivariate Gaussian distribution.

The whole optimization process that we use is iterative and is described below, with the total number of iterations denoted as nb_{iter}:

1. We start with nb_{init} points randomly sampled in $\mathcal{C}$, for which some evaluations of f have been obtained. With these points, we construct a training data set for the surrogate model:

$$\mathcal{D} = \{(\mathbf{x}_1, f\left(\mathbf{x}_1\right)), \ldots, (\mathbf{x}_{nb_{init}}, f(\mathbf{x}_{nb_{init}}))\}$$

2. We train the surrogate model on $\mathcal{D}$, with a fixed, estimated parameter $\hat{\theta}$ on which its distribution depends, and search for the next point to be evaluated for f, as:

$$\mathbf{x}_{next} = ArgMax_{\mathbf{x}\in\mathcal{C}}\, a_{EI}(\mathbf{x}; \hat{\theta})$$

or

$$\mathbf{x}_{next} = ArgMin_{\mathbf{x}\in\mathcal{C}}\, a_{UCB}(\mathbf{x}; \hat{\theta})$$

If $(\mathbf{x}_{next}, f(\mathbf{x}_{next}))$ is already found in $\mathcal{D}$, then $\mathbf{x}_{next}$ is picked randomly in $\mathcal{C}$.

3. $\mathcal{D}$ is enriched with $(\mathbf{x}_{next}, f(\mathbf{x}_{next}))$ and we return to point 2., until the budget of iterations, nb_{iter}, is reached.

For the BQRVFL (or any other machine learning model considered as a surrogate model for this purpose), an important point to consider in step 2. is the choice of θ, which is also a problem of hyperparameters' selection.

In the specific case of the BQRVFL, there are two hyperparameters for the surrogate model: the number of nodes in the hidden layer, and the regularization parameter σ^2. The choice of the activation function or the type of simulation for the nodes in the hidden layer is limited here to the ReLU activation function, and a Sobol sequence.

Since it is fast to compute for these type of models and we would like to avoid a costly hyperparameters' search at this point, we choose to minimize the Generalized Cross Validation (GCV) error (see [Gol+79]) on the dataset $\mathcal{D}$ constructed at step 1. (in order to obtain $\hat{\theta}$). The GCV error is a convenient approximation to the Leave-One-Out cross validation error, and is defined as:

$$GCV = \frac{1}{nb_{init}} \sum_{i=1}^{nb_{init}} \left(\frac{y_i - \hat{y}_i}{1 - \frac{tr(\mathbf{S})}{nb_{init}}} \right)^2 \tag{C.1}$$

where nb_{init} is the number of observations in the response variable

$$y = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{nb_{init}}))^T$$

$\mathbf{S}$ is a smoother matrix verifying $\hat{y} = \mathbf{S}y$, and $tr(\mathbf{S})$ is the trace of matrix $\mathbf{S}$. In the case of BQRVFL, y contains the observed values of the response y, $\mathbf{S} = \left(\mathbf{Z}^T\mathbf{Z} + \lambda I_p\right)^{-1}\mathbf{Z}^T$, and $\hat{y}$ contains the model's fitted values.

Having obtained $\hat{\theta}$, the *optimal* θ in point 2., searching for the next point to be evaluated is done with a gradient based optimization (cite nlminb PORT routine) of $\mathbf{x} \mapsto a_{EI}\left(\mathbf{x}; \hat{\theta}\right)$ and $\mathbf{x} \mapsto a_{UCB}\left(\mathbf{x}; \hat{\theta}\right)$.

C.3.2 Use of the BQRVFL for Bayesian optimization

We now illustrate the BQRFL model's behaviour on a simple example of Bayesian optimization. The objective function to be minimized is the rescaled Branin function, which is widely used for testing Bayesian optimization frameworks, and notably in (cite Picheny et al. (2012), Picheny, V., Wagner, T., & Ginsbourger, D. (2012). A benchmark of kriging-based infill criteria for noisy optimization). This function is defined for $\mathbf{x} = (x_1, x_2) \in [0, 1]^2$ as:

$$\mathbf{x} \mapsto \frac{1}{51.95}\left[\left(\bar{x}_2 - \frac{5.1\bar{x}_2}{4\pi^2} + \frac{5\bar{x}_1}{\pi} - 6\right)^2 + \left(10 - \frac{10}{8\pi}\right) cos(\bar{x}_1) - 44.81\right] \tag{C.2}$$

where $\bar{x}_1 = 15x_1 - 5$ $\bar{x}_2 = 15x_2$. The rescaled Branin function is presented below, in figures C.5 and C.6.

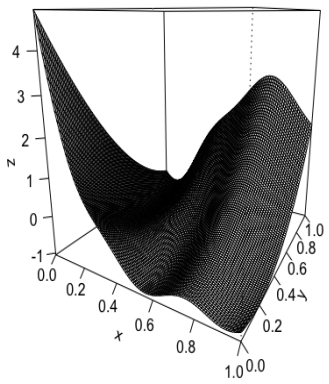

Fig. C.5.: Rescaled Branin function perspective plot

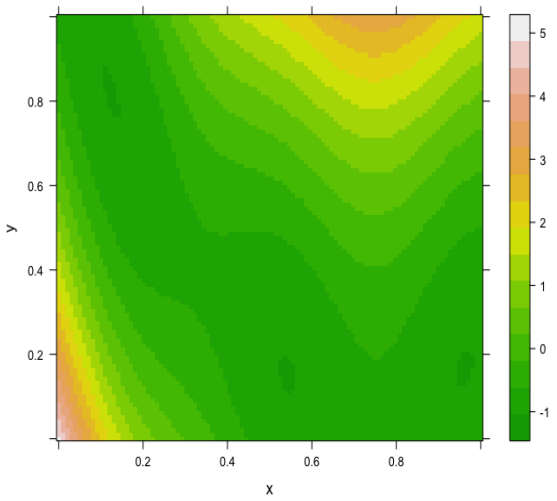

Fig. C.6.: Rescaled Branin function level plot

The global minimum value of the function is found (by using a multi-start gradient-based optimization) at $\mathbf{x} = (0.5427728, 0.1516667)$, and is equal to -1.047394.

Below, in figures C.7 and C.8 (and tables C.2 and C.3), we present the results obtained by using the BQRVFL model, and the Random Forest model as workhorses for the Bayesian optimization of the rescaled Branin function. In appendix C.5.1, we also present the evolution of the algorithm for the BQRVFL, as more $\mathbf{x}_{next}$ points (chosen by maximizing $a_{EI}(., \hat{\theta})$) are added to $\mathcal{D}$.

For figures C.7 and C.8 (and tables C.2 and C.3), we use $nb_{init} = 100$ and a total budget of 100 iterations. The algorithm is stochastic, because the nb_{init} points are picked randomly in $\mathcal{C}$ for each choice of `seed` (each restart of the algorithm).

Tab. C.2.: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Expected Improvement) with BQRVFL

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.024	-0.986	-0.986	-0.963	-0.812	0.046
50	-1.047	-1.027	-1.027	-1.027	-1.027	-1.010	**0.008**
100	-1.047	-1.027	-1.027	-1.029	-1.027	-1.025	**0.006**

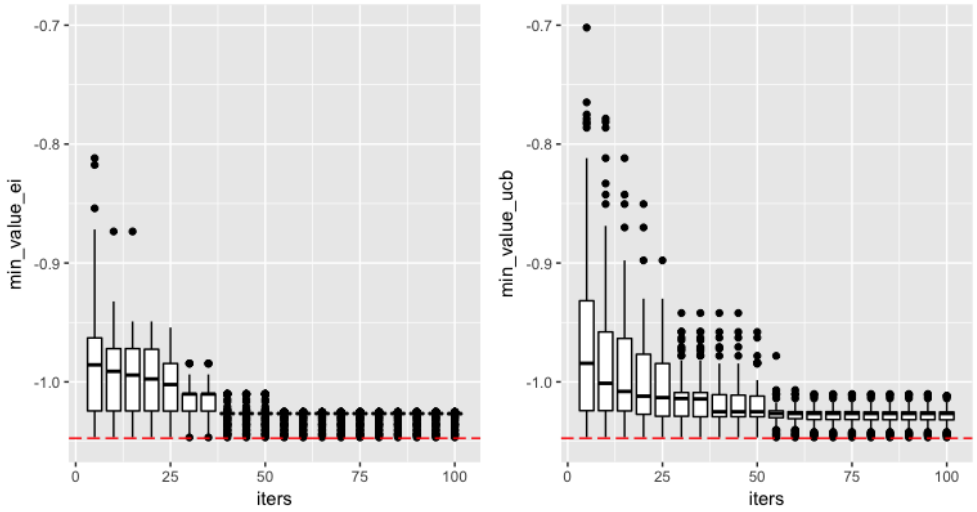

Fig. C.7.: Minimum value for the rescaled Branin function, as a function of the number of iterations with BQRVFL. **Left:** With Expected Improvement (EI) **Right:** With Upper Confidence Bound (UCB) - BQRVFL

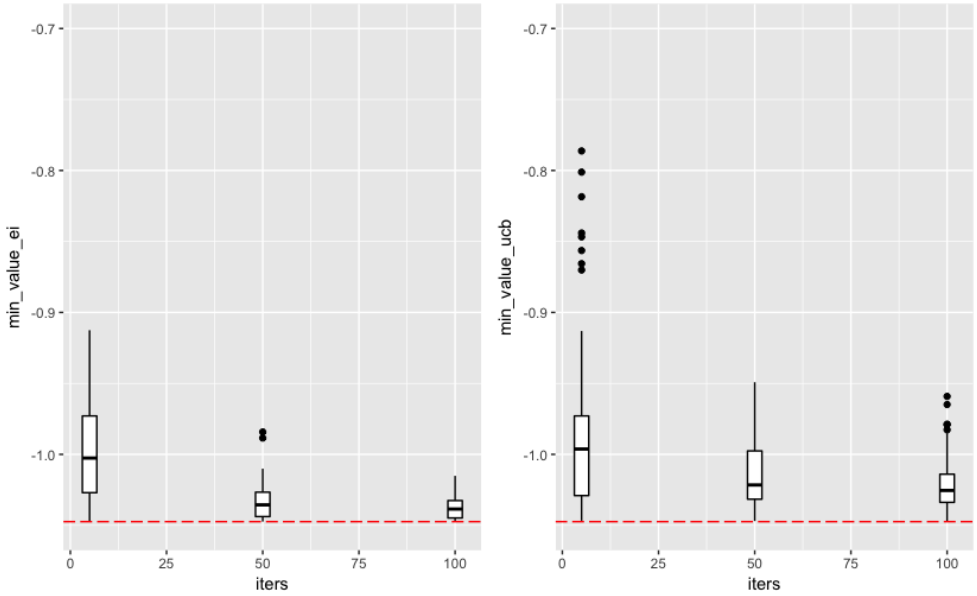

Fig. C.8.: Minimum value for the rescaled Branin function, as a function of the number of iterations. **Left:** With Expected Improvement (EI) **Right:** With Upper Confidence Bound (UCB) - Random Forest

Tab. C.3.: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Upper Confidence Bound) with BQRVFL

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.024	-0.984	-0.963	-0.932	-0.702	0.079
50	-1.047	-1.029	-1.025	-1.020	-1.012	-0.958	0.017
100	-1.047	-1.032	-1.027	-1.028	-1.025	-1.012	0.008

Tab. C.4.: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Expected Improvement) with Random Forest

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.027	**-1.003**	**-0.999**	-0.973	-0.912	**0.033**
50	-1.047	-1.044	**-1.036**	**-1.033**	-1.027	-0.984	0.012
100	-1.047	-1.045	**-1.039**	**-1.038**	-1.033	-1.015	0.008

Tab. C.5.: Summary of minimum values found for the rescaled Branin function, as a function of the number of iterations (with Upper Confidence Bound) with Random Forest

Nb. of iterations	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
5	-1.047	-1.029	-0.996	-0.989	-0.973	-0.786	0.055
50	-1.047	-1.032	-1.022	-1.015	-0.998	-0.949	0.023
100	-1.047	-1.034	-1.025	-1.022	-1.014	-0.959	0.019

For 100 seeds, and a fixed number of iterations nb_{iter}, we rerun the procedure described before in section C.3.1, 100 times. For the BQRVFL, nb_{iter} goes from 5 to 100 with steps of 5, and for the Random Forest with confidence intervals, $nb_{iter} \in \{5, 50, 100\}$. A further tuning of the Random Forest hyperparameters, or of the parameter κ, or a different choice for nb_{init}, or activation function for the BQRVFL, would of course lead to different results.

We observe that pretty good results are obtained with the Random Forest for the mean and the median value. But generally with a higher variance on the minimum value obtained than BQRVFL. Another interesting information obtained through these simulations is the number of times (over $nb_{iter} = 100$ iterations) $\mathbf{x}_{next}$ has been picked randomly in $\mathcal{C}$ at Step 2. (when $\mathbf{x}_{next}$ is already in $\mathcal{D}$).

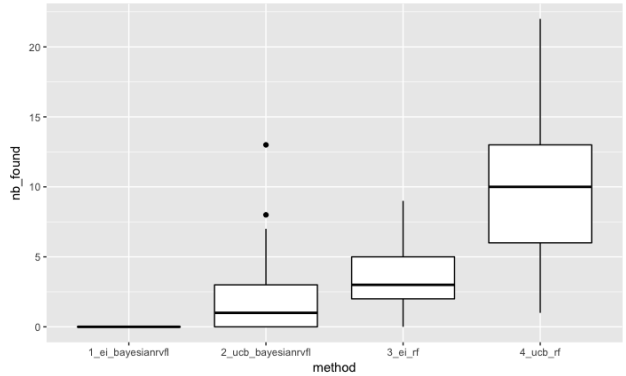

Fig. C.9.: Distribution of the number of times a simulation of $\mathbf{x}_{next} \in \mathcal{C}$ has been required at step 2.

Tab. C.6.: Distribution of the number of times a simulation of $\mathbf{x}_{next} \in \mathcal{C}$ has been required at step 2.

Method	Min	1st Qrt	Median	Mean	3rd Qrt	Max	Std. Dev
BQRVFL with EI	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
BQRVFL with UCB	0.00	0.00	1.00	1.96	3.00	13.00	2.22
RF with EI	0.00	2.00	3.00	3.66	5.00	9.00	1.98
RF with UCB	1.00	6.00	10.00	9.98	13.00	22.00	4.54

In table C.6 and figure C.9, we observe that on choosing *good* next points at step 2., BQRVFL generally performs better than the other models. Indeed with BQRVFL and Expected Improvement acquisition function for example, the next point to be evaluated is never obtained by simulation of $\mathbf{x}_{next} \in \mathcal{C}$.

A good place to apply Bayesian optimization, is to the choice of hyperparameters of machine learning algorithms. The cross-validation function, which is the objective function of the hyperparameters, is indeed expensive to train/predict over the cross-validation resamples.

C.3.3 Example of models based on Dynamic Nelson Siegel and Kernel Regularized Least Squares

Description of the models

In the previous chapters, we introduced the RVFL, the KRLS, and the DNS-RVFL, DNS-KRLS models. Here, we compare all these models by using the Bayesian optimization algorithm described in this chapter.

Examples of minimization of the out out-of-sample RMSE

Tab. C.7.: out-of-sample RMSE for models based on the DNS framework

Horizon	NS-Gaussian	NS-Matérn 3/2	NS-Matérn 5/2	NS-RVFL
12	0.6584	0.6590	0.6652	**0.5483**
18	0.8356	0.8356	0.8356	**0.7078**
24	1.0130	1.0130	1.0130	**0.8564**
30	1.1607	1.1607	1.1607	**1.0289**
36	1.2863	1.2863	1.2863	**1.1405**

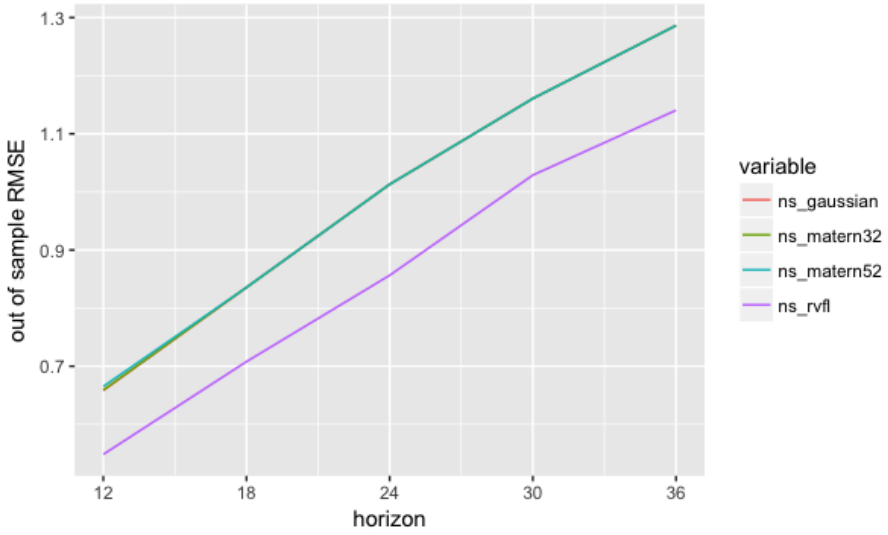

Fig. C.10.: out-of-sample RMSE for models based on the DNS framework

Tab. C.8.: average out-of-sample RMSE for data-driven KRLS models

Horizon	Gaussian	Matérn 3/2	Matérn 5/2	RVFL
12	0.6392	**0.5168**	0.5798	0.5540
18	0.7466	**0.6591**	0.7394	0.7032
24	0.8885	**0.7716**	0.8965	0.8191
30	1.0095	**0.8844**	1.0083	0.9535
36	1.1544	**1.0099**	1.1396	1.0646

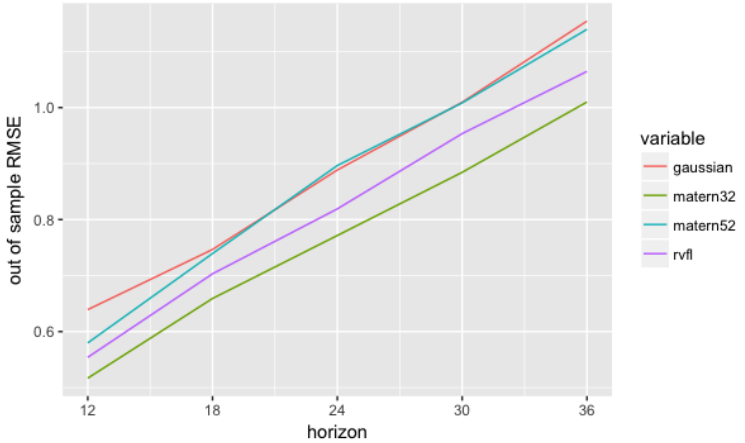

Fig. C.11.: average out-of-sample RMSE for data-driven models

Details of the distribution of errors are provided in the appendix C.5.2 (boxplots and summaries).

C.4 Conclusion

In this chapter, we presented a quasi-randomized functional link neural networks (BQRVFL) model, by linking it to a Bayesian linear regression framework, and obtained formulas for the confidence intervals around its predictions. The BQRVFL model that we constructed is then used as a workhorse for Bayesian optimization, and notably to find *optimal* hyperparameters for all the forecasting models described in the previous chapters.

In general, we observe that the *best* results are obtained in the DNS-KRLS framework by the RVFL (applied to the level, slope and curvature). And in the KRLS framework, the kernel Matérn 3/2 performs the best. This kernel also performs the best overall, in terms of average out-of-sample RMSE. However, it is worth mentioning, as in previous chapters, that the cost of computing the kernel is a quadratic function of the number of input spot rates.

Figure (C.12) illustrates this fact very well, with 100 repeats at each time to maturity (see appendix C.5.2 for details on the timings). It compares the KRLS and RVFL models timings on training and forecasting horizons equal to 12 months and 36 months, and respectively 15, 22 and 30 observed time to maturities. The RVFL timings remain relatively stable as the number of time to maturities increase, whereas the KRLS timings increase at a fast pace in the same situation.

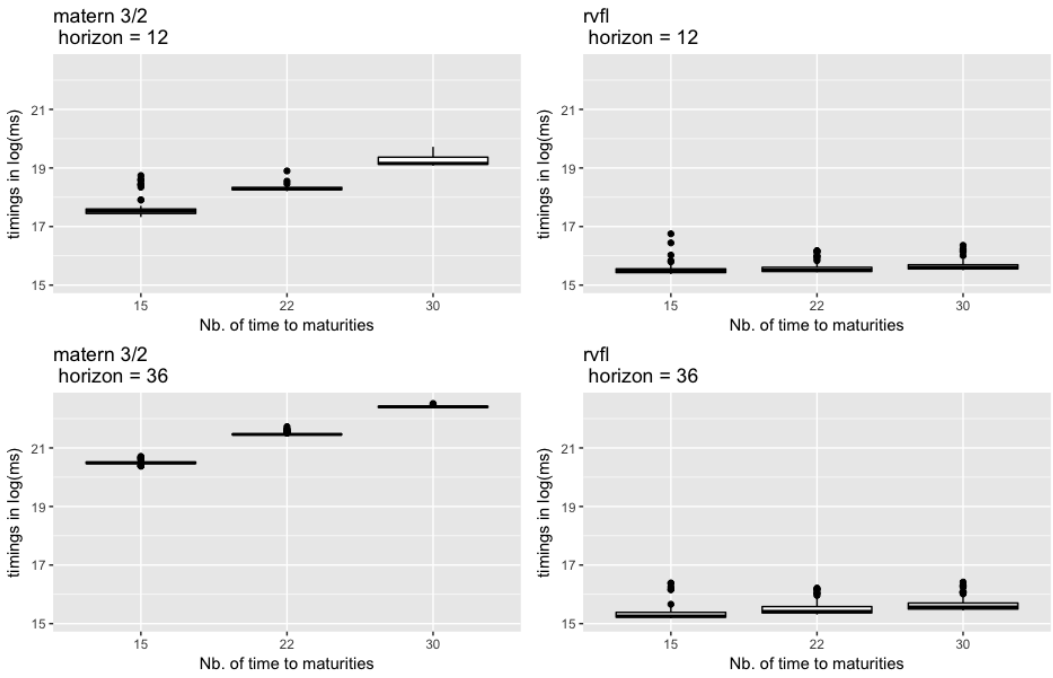

Fig. C.12.: timings of KRLS vs timings of RVFL

C.5 Appendix

C.5.1 Evolution of the algorithm (choices of $\mathbf{x}_{next}$)

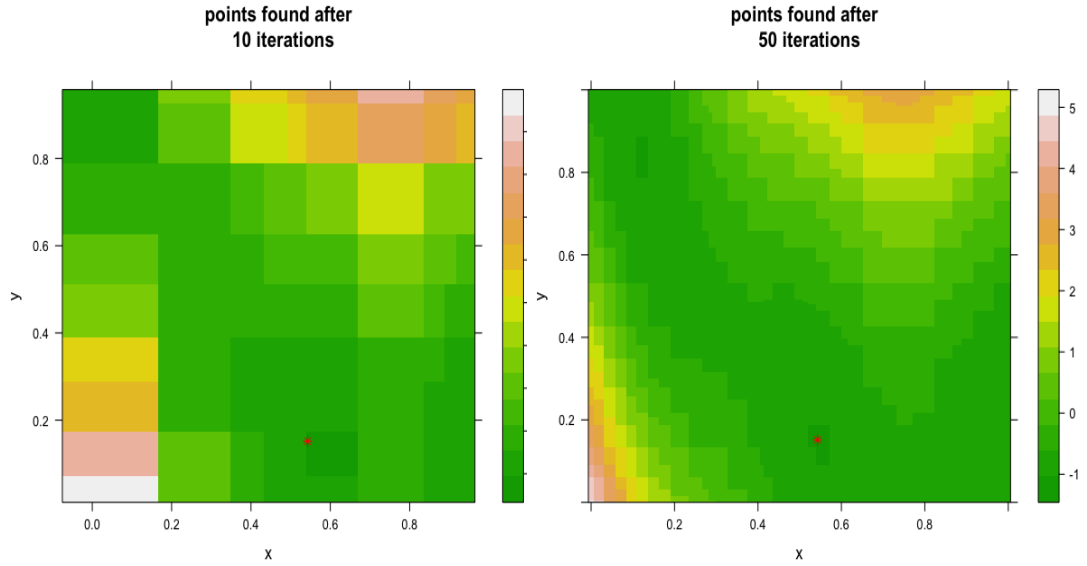

Fig. C.13.: Points found by the algorithm after 10 iterations

Fig. C.14.: Points found by the algorithm after 50 iterations

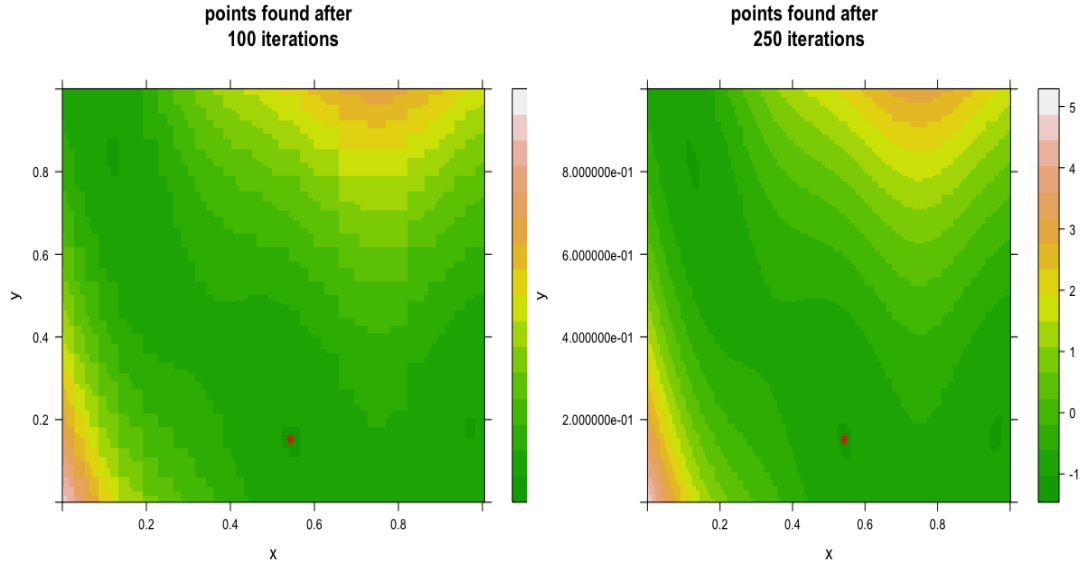

Fig. C.15.: Points found by the algorithm after 100 iterations

Fig. C.16.: Points found by the algorithm after 250 iterations

C.5.2 Timings in log(ms)

matern 3/2 - horizon = 12

```
 15              22              30
 Min.   :17.33   Min.   :18.21   Min.   :19.08
 1st Qu.:17.45   1st Qu.:18.26   1st Qu.:19.13
 Median :17.52   Median :18.28   Median :19.16
 Mean   :17.60   Mean   :18.30   Mean   :19.23
 3rd Qu.:17.60   3rd Qu.:18.34   3rd Qu.:19.37
 Max.   :18.74   Max.   :18.90   Max.   :19.72
```

rvfl - horizon = 12

```
15              22              30
 Min.   :15.37   Min.   :15.44   Min.   :15.51
 1st Qu.:15.43   1st Qu.:15.47   1st Qu.:15.55
 Median :15.48   Median :15.51   Median :15.61
 Mean   :15.53   Mean   :15.58   Mean   :15.66
 3rd Qu.:15.56   3rd Qu.:15.61   3rd Qu.:15.70
 Max.   :16.75   Max.   :16.18   Max.   :16.36
```

matern 3/2 - horizon = 36

```
  15              22              30
 Min.   :20.38   Min.   :21.43   Min.   :22.36
 1st Qu.:20.47   1st Qu.:21.45   1st Qu.:22.39
 Median :20.49   Median :21.46   Median :22.40
 Mean   :20.50   Mean   :21.47   Mean   :22.40
 3rd Qu.:20.52   3rd Qu.:21.47   3rd Qu.:22.42
 Max.   :20.71   Max.   :21.73   Max.   :22.52
```

rvfl - horizon = 36

```
15              22              30
 Min.   :15.18   Min.   :15.31   Min.   :15.45
 1st Qu.:15.22   1st Qu.:15.37   1st Qu.:15.50
 Median :15.26   Median :15.42   Median :15.57
 Mean   :15.35   Mean   :15.51   Mean   :15.65
 3rd Qu.:15.38   3rd Qu.:15.58   3rd Qu.:15.70
 Max.   :16.39   Max.   :16.21   Max.   :16.42
```

Conclusion

D

Bibliography

[AB13] Ferdinando M Ametrano and Marco Bianchetti. „Everything you always wanted to know about multiple interest rate curve bootstrapping but were afraid to ask“. In: *Available at SSRN 2219548* (2013) (cit. on pp. 4–6, 11, 18, 24, 38).

[And07] Leif Andersen. „Discount curve construction with tension splines“. In: *Review of Derivatives Research* 10.3 (2007), pp. 227–267 (cit. on pp. 6, 13, 18, 20, 22, 36).

[AP10] Leif BG Andersen and Vladimir V Piterbarg. *Interest rate modeling*. Atlantic Financial Press, 2010 (cit. on pp. 6, 12, 18, 19, 36).

[AW14] Monther Alhamdoosh and Dianhui Wang. „Fast decorrelated neural network ensembles with random weights“. In: *Information Sciences* 264 (2014), pp. 104–117 (cit. on p. 75).

[BC99] Tomas Björk and Bent Jesper Christensen. „Interest rate dynamics and consistent forward rate curves“. In: *Mathematical Finance* 9 (1999), pp. 323–348 (cit. on pp. 6, 11).

[Ber+15] Christoph Bergmeir, Rob J Hyndman, Bonsoo Koo, et al. „A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction“. In: *Monash University, Department of Econometrics and Business Statistics, Tech. Rep.* (2015) (cit. on pp. 51, 82, 103).

[Bli97] Robert R Bliss. „Testing Term Structure Estimation Methods“. In: *Advances in Futures and Options Research* 9 (1997), pp. 197–231 (cit. on p. 13).

[Bon+15] F Bonnin, F Combes, F Planchet, and M Tammar. „Un modèle de projection pour des contrats de retraite dans le cadre de l'ORSA“. In: *Bulletin Français d'Actuariat* 14 (2015), pp. 107–129 (cit. on p. 39).

[BP70] George EP Box and David A Pierce. „Distribution of residual autocorrelations in autoregressive-integrated moving average time series models“. In: *Journal of the American statistical Association* 65.332 (1970), pp. 1509–1526 (cit. on p. 81).

[Bre01] Leo Breiman. „Random forests“. In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 74).

[Bre96a] Leo Breiman. „Bagging predictors“. In: *Machine learning* 24.2 (1996), pp. 123–140 (cit. on p. 69).

[Bre96b] Leo Breiman. „Stacked regressions“. In: *Machine learning* 24.1 (1996), pp. 49–64 (cit. on pp. 77, 78).

[BT97] Phelim P Boyle and Ken Seng Tan. „Quasi-Monte Carlo methods“. In: *International AFIR Colloquium Proceedings, Australia*. Vol. 1. 1997, pp. 1–24 (cit. on pp. 44, 115).

[BY03] Peter Bühlmann and Bin Yu. „Boosting with the L 2 loss: regression and classification“. In: *Journal of the American Statistical Association* 98.462 (2003), pp. 324–339 (cit. on p. 69).

[Car98] Rich Caruana. „Multitask learning“. In: *Learning to learn*. Springer, 1998, pp. 95–133 (cit. on p. 46).

[CC10] CFOForum and CROForum. „QIS 5 Technical Specification Risk-free interest rates“. In: (2010) (cit. on p. 4).

[Cha+92] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. „Forecasting the behavior of multivariate time series using neural networks“. In: *Neural networks* 5.6 (1992), pp. 961–970 (cit. on p. 39).

[Chr+11] Jens HE Christensen, Francis X Diebold, and Glenn D Rudebusch. „The affine arbitrage-free class of Nelson–Siegel term structure models“. In: *Journal of Econometrics* 164.1 (2011), pp. 4–20 (cit. on pp. 5, 16).

[Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009 (cit. on p. 48).

[DC10] Satchidananda Dehuri and Sung-Bae Cho. „A comprehensive survey on functional link neural networks and an adaptive PSO–BP learning for CFLNN“. In: *Neural Computing and Applications* 19.2 (2010), pp. 187–205 (cit. on pp. 40, 113).

[DL06] Francis X Diebold and Canlin Li. „Forecasting the term structure of government bond yields“. In: *Journal of econometrics* 130.2 (2006), pp. 337–364 (cit. on pp. 4, 6, 16, 28, 40, 49, 51, 89, 94, 114).

[DR13] Francis X Diebold and Glenn D Rudebusch. *Yield Curve Modeling and Forecasting: The Dynamic Nelson-Siegel Approach*. Princeton University Press, 2013 (cit. on pp. 49, 50).

[DS15] Christophe Dutang and Petr Savicky. *randtoolbox: Generating and Testing Random Numbers*. R package version 1.17. 2015.

[Efr92] Bradley Efron. „Bootstrap methods: another look at the jackknife“. In: *Breakthroughs in statistics*. Springer, 1992, pp. 569–593 (cit. on p. 73).

[EIO15] EIOPA. „Technical documentation of the methodology to derive EIOPA's risk- free interest rate term structures“. In: *https://eiopa.europa.eu/Publications* (2015) (cit. on pp. 4, 5).

[ET86] Bradley Efron and Robert Tibshirani. „Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy“. In: *Statistical science* (1986), pp. 54–75 (cit. on p. 73).

[Ext+16] Peter Exterkate, Patrick JF Groenen, Christiaan Heij, and Dick van Dijk. „Nonlinear forecasting with many predictors using kernel ridge regression“. In: *International Journal of Forecasting* 32.3 (2016), pp. 736–753 (cit. on pp. 39, 89, 91, 96, 99).

[Fer+17] Jeremy Ferwerda, Jens Hainmueller, Chad J Hazlett, et al. „Kernel-Based Regularized Least Squares in R (KRLS) and Stata (krls)“. In: *Journal of Statistical Software* 79.i03 (2017) (cit. on pp. 89, 91, 93).

[Fri01] Jerome H Friedman. „Greedy function approximation: a gradient boosting machine“. In: *Annals of statistics* (2001), pp. 1189–1232 (cit. on p. 69).

[Gol+79] Gene H Golub, Michael Heath, and Grace Wahba. „Generalized cross-validation as a method for choosing a good ridge parameter“. In: *Technometrics* 21.2 (1979), pp. 215–223 (cit. on p. 121).

[Gre00] William H Greene. „Econometric analysis 4th edition“. In: *International edition, New Jersey: Prentice Hall* (2000) (cit. on p. 78).

[Gre03] William H Greene. „Econometric analysis, 5th“. In: *Ed.. Upper Saddle River, NJ* (2003), pp. 89–140 (cit. on pp. 79, 80).

[GVL12] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012 (cit. on pp. 90, 115).

[HH93] Andrew C Harvey and Andrew C Harvey. *Time series models*. Vol. 2. Harvester Wheatsheaf New York, 1993 (cit. on p. 81).

[HK08] Robin Hyndman and Yeasmin Khandakar. „Automatic time series forecasting: The forecast Package for R“. In: *Journal of Statistical Software* 27.3 (2008), pp. 1–22 (cit. on pp. 29, 51, 103).

[HK70] Arthur E Hoerl and Robert W Kennard. „Ridge regression: Biased estimation for nonorthogonal problems“. In: *Technometrics* 12.1 (1970), pp. 55–67 (cit. on pp. 43, 78, 90).

[Hof+08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. „Kernel methods in machine learning“. In: *The annals of statistics* (2008), pp. 1171–1220 (cit. on p. 92).

[Hot+10] Torsten Hothorn, Peter Bühlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. „Model-based boosting 2.0“. In: *Journal of Machine Learning Research* 11.Aug (2010), pp. 2109–2113 (cit. on pp. 69, 71, 76).

[HP81] J Michael Harrison and Stanley R Pliska. „Martingales and stochastic integrals in the theory of continuous trading“. In: *Stochastic processes and their applications* 11.3 (1981), pp. 215–260 (cit. on p. 3).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 39).

[HU07] Rob J Hyndman and Md Shahid Ullah. „Robust forecasting of mortality and fertility rates: a functional data approach“. In: *Computational Statistics & Data Analysis* 51.10 (2007), pp. 4942–4956 (cit. on pp. 6, 16).

[HW06] Patrick S Hagan and Graeme West. „Interpolation methods for curve construction“. In: *Applied Mathematical Finance* 13.2 (2006), pp. 89–129 (cit. on pp. 6, 18, 19, 22, 24, 37).

[HW12] John Hull and Alan White. „The FVA debate“. In: *Risk magazine* 25.8 (2012), pp. 83–85 (cit. on p. 4).

[HW90] J. Hull and A. White. „Pricing interest-rate derivative securities“. In: *The Review of Financial Studies* 3.4 (1990), pp. 573–592 (cit. on p. 7).

[JC14] Hainmueller Jens and Hazlett Chad. „Kernel regularized least squares: Reducing misspecification bias with a flexible ad interpretable machine learning approach“. In: *Political Analysis* 22.2 (2014), pp. 143–68 (cit. on pp. 89, 91).

[JK08] S Joe and F Kuo. *Notes on generating Sobol sequences*. `http://web.maths.unsw.edu.au/~fkuo/sobol/joe-kuo-notes.pdf`. 2008 (cit. on pp. 44, 45).

[Jon01] Donald R Jones. „A taxonomy of global optimization methods based on response surfaces“. In: *Journal of global optimization* 21.4 (2001), pp. 345–383 (cit. on pp. 114, 119).

[LB78] Greta M Ljung and George EP Box. „On a measure of lack of fit in time series models“. In: *Biometrika* 65.2 (1978), pp. 297–303 (cit. on p. 81).

[LH74] CL Lawson and RJ Hanson. „Solving least squares problems, SIAM, Philadelphia, PA, 1995“. In: *First published by Prentice-Hall* (1974) (cit. on p. 78).

[LH95] Charles L Lawson and Richard J Hanson. *Solving least squares problems*. Vol. 15. Siam, 1995 (cit. on p. 78).

[Lit+91] Robert B Litterman, José Scheinkman, and Laurence Weiss. „Volatility and the yield curve“. In: *The Journal of Fixed Income* 1.1 (1991), pp. 49–53 (cit. on pp. 4, 16).

[Lüt05] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005 (cit. on pp. 45, 51, 95).

[Moc+78] J Mockus, V Tiesis, and A Zilinskas. „Toward global optimization, volume 2, chapter bayesian methods for seeking the extremum“. In: (1978) (cit. on pp. 114, 119).

[Mou+18] Thierry Moudiki, Frédéric Planchet, and Areski Cousin. „Multiple Time Series Forecasting Using Quasi-Randomized Functional Link Neural Networks“. In: *Risks* 6.1 (2018), p. 22 (cit. on pp. 69, 70, 114).

[Nie92] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992 (cit. on pp. 45, 115).

[NS87] Charles R Nelson and Andrew F Siegel. „Parsimonious modeling of yield curves“. In: *Journal of business* (1987), pp. 473–489 (cit. on pp. 4, 5, 28, 40, 49, 57, 89).

[Pan12] Alan Pankratz. *Forecasting with dynamic regression models*. Vol. 935. John Wiley & Sons, 2012 (cit. on pp. 40, 49).

[Pao+94] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. „Learning and generalization characteristics of the random vector functional-link net“. In: *Neurocomputing* 6.2 (1994), pp. 163–180 (cit. on pp. 39, 113).

[Pen55] Roger Penrose. „A generalized inverse for matrices“. In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 51. 03. Cambridge Univ Press. 1955, pp. 406–413 (cit. on p. 48).

[Pfa+08] Bernhard Pfaff et al. „VAR, SVAR and SVEC models: Implementation within R package vars“. In: *Journal of Statistical Software* 27.4 (2008), pp. 1–32 (cit. on p. 51).

[RD91] James O Ramsay and CJ Dalzell. „Some tools for functional data analysis“. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1991), pp. 539–572 (cit. on pp. 6, 16).

[Ren+16] Ye Ren, PN Suganthan, N Srikanth, and Gehan Amaratunga. „Random vector functional link network for short-term electricity load demand forecasting“. In: *Information Sciences* 367 (2016), pp. 1078–1093 (cit. on p. 40).

[Ros96] Bruce E Rosen. „Ensemble learning using decorrelated neural networks“. In: *Connection science* 8.3-4 (1996), pp. 373–384 (cit. on pp. 74, 75).

[RS05] JO Ramsay and BW Silverman. „Springer Series in Statistics“. In: *Functional data analysis*. Springer, 2005 (cit. on pp. 6, 16, 17).

[Rum+88] DE Rumelhart, GE Hinton, and RJ Williams. „Learning internal representations by error propagation“. In: *Neurocomputing: foundations of research*. MIT Press. 1988, pp. 673–695 (cit. on p. 39).

[RW06] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006 (cit. on pp. 92, 114, 115).

[Sap+14] Stephanie Sapp, Mark J van der Laan, and John Canny. „Subsemble: an ensemble method for combining subset-specific algorithm fits“. In: *Journal of applied statistics* 41.6 (2014), pp. 1247–1259 (cit. on pp. 114, 117).

[Sch+92] Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin. „Feedforward neural networks with random weights“. In: *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*. IEEE. 1992, pp. 1–4 (cit. on pp. 39, 113).

[Sno+12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. „Practical bayesian optimization of machine learning algorithms“. In: *Advances in neural information processing systems*. 2012, pp. 2951–2959 (cit. on p. 114).

[Sve94] Lars EO Svensson. *Estimating and interpreting forward interest rates: Sweden 1992-1994*. Tech. rep. National Bureau of Economic Research, 1994 (cit. on pp. 4, 28).

[SW01] A Smith and T Wilson. „Fitting yield curves with long term constraints“. In: *London: Bacon and Woodrow* (2001) (cit. on pp. 5, 6, 18–20).

[Wag+14] Stefan Wager, Trevor Hastie, and Bradley Efron. „Confidence intervals for random forests: The jackknife and the infinitesimal jackknife“. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1625–1651 (cit. on p. 114).

[Wel10] Max Welling. „The kalman filter“. In: *Lecture Note* (2010) (cit. on pp. 90, 115).

[Whi12] Richard White. „Multiple Curve Construction“. In: *Quantitative research, OpenGamma* (2012) (cit. on pp. 5, 11).

[Wic16] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2016.

[Wol92] David H Wolpert. „Stacked generalization“. In: *Neural networks* 5.2 (1992), pp. 241–259 (cit. on pp. 69, 77).

[ZS16] Le Zhang and PN Suganthan. „A comprehensive evaluation of random vector functional link networks“. In: *Information Sciences* 367 (2016), pp. 1094–1105 (cit. on pp. 40, 43).

List of Figures

List of Tables

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

You can put your declaration here, to declare that you have completed your work solely and only with the help of the references you mentioned.

City, August 26, 2015

Ricardo Langner