UNIVERSITY OF COPENHAGEN

MSc. Mathematics-Economics

Your Neural Network is Your Net Worth

Valuation and Hedging of Interest Rate Derivatives with Automatic Adjoint Differentiation and Differential Machine Learning

Andreas Axel
Nicklas Kenno Hansen
Sebastian Valdemar Hansen

Supervisor: David Glavind Skovmand
Date: December 20th 2023

Abstract

In this thesis, we have succesfully implemented a flexible and scalable setup for performing efficient valuation and accurate hedging of interest rate derivatives by Monte Carlo simulation instrumented with Automatic Adjoint Differentiation. Our main contribution demonstrates effective training of Differential Machine Learning models for a variety of interest rate products, including linear, European, path-dependent, and callable contracts. This is carried out within the classic Vasicek model, and the General Multi-Factor Stochastic Volatility Model by Trolle and Schwartz. Our findings reveal within the Vasicek model that Differential Regression perform almost on par with Differential Neural Networks. However, when dealing with the more complex multi-factor stochastic volatility model, it becomes evident that only the Differential Neural Networks are capable of consistently fitting stable and accurate estimators.

Contents

1 Introduction

Recent interest rate hikes have increased volatility in fixed-income markets highlighting the need for a robust and efficient pricing and risk management setup for interest rate products.

In this thesis, we investigate how Differential Machine Learning [SH20] can be applied in the context of pricing interest rate derivatives and calculating risk sensitivities used for hedging. The central concept ensuring that this is viable in practice is training on path-wise samples of both discounted payoffs from Monte Carlo simulations and the corresponding sensitivities with respect to the input arguments by utilizing Automatic Adjoint Differentiation. As a proof of concept, we apply the techniques in a simple, yet classical Vasicek short-rate model. We show how Differential Regression and Differential Neural Networks can be used for standard interest rate products such as caps and European swaptions, but also more complex products with e.g. path-dependency as the Barrier option, or early-callability as the Bermudan. In a dynamically hedging context, we show how Differential Machine Learning can estimate risk sensitivities that can be used to immunize trading risks for a wide selection of products. Similarly, we examine how well Differential Machine Learning can be used for decomposing risk sensitivities for popular trading strategies such as a *butterfly spread* or a *straddle* (among others) to get a clear picture of how different movements in underlyings can affect a trader's PnL or *Net Worth*.

We introduce the model from [TS06] which is interesting since it features stochastic volatility and multiple risk-driving factors, allowing for a more general and rich market description. We apply Differential Machine Learning once again to learn prices and risk figures in a setting where the yield curve is not able to capture the entire volatility. As far as we know, Differential Machine Learning has not previously been applied to term-structure / rate-curve models with multiple factors and stochastic volatility and our work thus makes a fulfilling contribution to the existing literature.

We have tried to keep the project as self-contained as possible including implementation details and considerations. To accompany the results reported here, we invested significant amount of effort in producing a flexible and efficient environment for our

Monte Carlo engine, that tries to resemble a "near" production-level setup inspired by [Sav19]. The code for our engine can be found in our repository along with code for producing every single result and figure reported in this project. All in all, this adds up to about 25,000 lines of code (including a wide selection of tests, ensuring the quality of our implementation).

1.1 Disposition

Below follows a brief description of what, we cover in the different sections within this thesis.

In section 2 we introduce a wide selection of (more or less common) interest rate derivatives such as swaps, caps, and swaptions, and how their cashflows are structured. We use this to define a general setup to determine arbitrage-free prices from cashflow definitions.

Section 3 gives an introduction to the Vasicek single-factor short-rate model and focuses in particular on deriving distributional properties for the short rate and how bond prices can be derived within the model relying on its affine properties.

Section 4 dives into Monte Carlo methods with outset in well-known theory. For this, we present some of the main techniques for solving problems of probabilistic nature through simulation in form of Monte Carlo integration and how these can be used to approximate the value and risk of financial claims. Furthermore, we describe a variety of practical considerations about how to efficiently implement a scalable Monte Carlo framework by segregating model and product objects in the engine and code. Additionally we cover some variance reduction techniques and the least-squares-method which allow us to transform callable products into path-dependent ones.

Section 5 motivates the concept of Automatic Differentiation and how it can be used in finding sensitivities of interest rate products. This concept plays particularly well together with Monte Carlo methods to produce pathwise prices *and* differentials in constant time. We also present methods for handling discontinuities in payoff functions by adding smoothing and fuzzy logic to the product specification.

Section 6 introduces the concept of Differential Machine Learning which relies on learning from not only pathwise discounted payoffs, but also differential labels in the form of pathwise sensitivities. We consider two different specifications, namely Differential Regression and Differential Neural Networks. Both use differential regularization which improves the accuracy of their predictions and requires fewer training samples compared to their traditional counterparts.

Section 7 puts all of the previous sections together by showcasing the performance of the differential machine learning estimators in a variety of experiments. We compare the models against each other and their traditional counterparts in terms of estimating the value and sensitivity functions of the interest rate derivatives from section 2. Furthermore, we compare the performance of the estimators when they are used to hedge the products and replicate their payoff functions.

Section 8 provides an introduction to the general framework of Heath-Jarrow-Morton for how the entire forward rate curve is modelled. In particular we investigate instances of the model which results in a Markovian property, that comes with a great practical importance.

In section 9 we introduce a stochastic volatility multi-factor model, which is an extension of the Heath-Jarrow-Morton framework that possess a Markovian structure. We derive expressions for the instantaneous forward rate and the zero-coupon bond prices. Furthermore, we show how they depend on a finite set of state variables. In practice we simulate these from an Euler scheme and show how the model can be used to price interest rate derivatives using both Monte Carlo approximation and a semi-analytical pricing formula.

In section 10 we use Differential Machine Learning to price and hedge interest rate derivatives in the more complex stochastic volatility model. In this case, we find the Differential Neural Network particularly robust for estimating prices and risk sensitivities.

Finally, we summarize our findings and provide a conclusion for our thesis in section

11, while also highlights a few interesting topics that could be subject to further considerations and research.

1.2 Market Description

Before proceeding to the different interest rate products and derivatives, we first provide a brief overview of the market we consider in this thesis.

A simple approach for modelling *one* interest rate could be to apply the machinery we know very well from Black-Scholes. Here we model an arbitrage-free evolution of the future price of an asset given today's observed price. This can constitute a perfectly good model, but the issue is, we obtain just *one* interest rate. There is certainly applications for this, but if we wish to model the entire curve and be consistent across maturities, this procedure falls short. When modelling interest rate products, we typically do not have *one* underlying asset so to say, we have a collection of all rates for all maturities that is prevailing at a given time point. The underlying asset of interest thus becomes a whole curve.

A classical first approach for modelling the entire curve is through short rate models. Here the short rate drives the complete curve. In particular we will dive into the Vasicek model where we assume the short rate to follow an Ornstein-Uhlenbeck process and deduce curve dynamics from this. This approach, however, also has its short comings of e.g. not always being flexible enough to be calibrated to an observed initial term structure.

A more general framework was introduced by Heath-Jarrow-Morton for modelling the entire instantaneous forward rate curve directly. By this approach, we take today's curve as given and model its future evolution for all terms simultaneously. We will investigate this framework later with a particular interest in the special cases where HJM models has a Markovian structure.

All in all, for an interest rate model to be of any real practical relevance it has to model rates of all maturities, what we also interpret as the entire yield curve.

1.2.1 Discounting and Interest Rates

One of the most fundamental concepts within finance is the *time value of money* captured by the phrase "a dollar today is worth more than a dollar tomorrow". In the context of asset pricing, the time value of money is reflected by discounting expected future cashflows to present value. The net present value (NPV) of an asset is then the collection of its discounted expected future cashflows.

In order to carry out the discounting, we let $P(t,T)$ denote the time t value of a *zero-coupon bond* (ZCB or T-bond) with maturity T. ZCBs are riskless in the sense that they are guaranteed to pay \$1 at maturity without coupon payments during their lifetime. Hence, they express the time-t value of having a dollar at time T and can therefore also be used as *discount factors* of other assets' cashflows. For this reason, we notice that the function $T \mapsto P(t,T)$ is called the *discount curve* or *zero-coupon curve*. Knowing this curve of discount factors we can effectively discount payments on all future dates which eventually determines the price of asset entitled to these. Besides this description of the ZCBs, we also assume that

- $P(T,T) = 1, \ \forall\, T \geq 0$,
- $P(t,T)$ is differentiable in T.

As ZCBs determines present value of cashflows they play a crucial role in asset pricing. Naturally it then becomes a central question to valuing these bonds as they will be indispensable in further asset pricing analysis.

If the short rate is assumed to be deterministic, we note that $P(t,T), \ t > 0$ is actually already known today at time 0. This becomes clear from making the following two self-financing transactions: At time 0, we buy 1 T-bond at the cost of $P(0,T)$ and sell $P(0,T)/P(0,t)$ units of t-bonds. At time t, we must pay $P(0,T)/P(0,t)$, while we and are guaranteed to receive \$1 at time T. To finance the intermediate time period, we can sell 1 T-bond for the price of $P(t,T)$, which must cost exactly

$$P(t,T) = \frac{P(0,T)}{P(0,t)}, \tag{1.1}$$

in order to not impose any arbitrage. However when short rates are stochastic, similar derivations are not attainable. Instead under imposed no-arbitrage arguments and the existence of a risk-neutral measure these can be determined from interest rate modelling of the short rate, $r(t)$, by the formula

$$P(t,T) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-\int_t^T r(s)ds}\right]. \tag{1.2}$$

In this thesis, we limit ourselves to the problems related to pricing and risk management of different interest rate derivatives. For the risk management, we focus on calculating risk sensitivities and constructing hedging portfolios. Thus, we now briefly introduce some of the relevant rates and their notation.

Instead of selling and rolling over zero coupon bonds in order to obtain financing, we use the *(simple continuously compounded) forward rate*, $F(t;T,T+\delta)$. It is the rate, contracted at time t, at which we can borrow funds over the accrual / coverage period $[T;\ T+\delta]$. To be consistent with the price of the zero coupon rates, the following equivalent relation must hold

$$1+\delta F(t;T,T+\delta) = \frac{1}{P(T,T+\delta)} \quad \Leftrightarrow \quad F(t;T,T+\delta) = \frac{1}{\delta}\left(\frac{P(t,T)}{P(t,T+\delta)} - 1\right). \tag{1.3}$$

Historically, this could for instance have been a 3-month forward xIBOR (Interbank Offered Rate), but since the LIBOR scandal and the following transition has been made towards alternative reference rates such as the *Secured Overnight Financing Rate* (SOFR), *Euro Short-Term Rate* (ESTR), or *Overnight Index Swap* (OIS) rate.

If the forward rate is contracted at the start of the accrual period, then we refer to it as the *simple spot rate* which is denoted as

$$F(T,T+\delta) := F(T;T,T+\delta) = \frac{1}{\delta}\left(\frac{1}{P(T,T+\delta)} - 1\right).$$

Furthermore, we also introduce the *instantaneous forward rate* with maturity T as

$$f(t,T) = -\frac{\partial \log P(t,T)}{\partial T}, \tag{1.4}$$

where the function $T \mapsto f(t,T)$ is called the *instantaneous forward curve*. Using the assumption that $P(T,T) = 1$ in combination with equation (1.4), the zero coupon bond prices can be expressed in terms of the instantaneous forward rate, i.e.

$$P(t,T) = e^{-\int_t^T f(t,u)du}. \tag{1.5}$$

In the limit of the short end of the instantaneous forward curve, we have the definition of the *(risk-free) instantaneous short rate* given by

$$r(t) = f(t,t).$$

This rate is particular important for defining the *money market account*[1], B. This can be derived by the following arguments: The return of investing one dollar today over the period $[0, \Delta t]$ is given by

$$\frac{1}{P(0,\Delta t)} = e^{\int_0^{\Delta t} f(0,u)du}.$$

We can approximate the exponential function (as a function of Δt) using it's Taylor series around 0, that is

$$\begin{aligned} e^{\int_0^{\Delta t} f(0,u)du} &= 1 + f(0,0)e^{\int_0^0 f(0,u)du}\Delta t + o(\Delta t) \\ &= 1 + r(0)\Delta t + o(\Delta t). \end{aligned}$$

If we reinvest the amount *instantaneously* at time Δt in a new ZCB with expiry $2\Delta t$, then we get the combined return of investments at time $2\Delta t$ to be

$$\frac{1}{P(0,\Delta t)}\frac{1}{P(\Delta t, 2\Delta t)} = (1 + r(0)\Delta t)(1 + r(\Delta t)\Delta t) + o(\Delta t).$$

Following this strategy of consecutively rolling over our returns in new ZCBs leads to what we conceptually think of as the money-market or bank account in the limit. The bank account $B(t)$ is then the asset which at time t grows by $r(t)$. Hence,

$$B(t + \Delta t) = B(t)(1 + r(t)\Delta t) + o(\Delta t).$$

[1] Also referred to as the *bank account* and *risk-free asset*.

And in the limit $\Delta t \to 0$ this converges towards the process

$$dB(t) = r(t)B(t)dt, \quad B(0) = 1.$$

The explicit solution for this ordinary differential equation is given by the well known expression

$$B(t) = e^{\int_0^t r(s)ds}.$$

The bank account is important for relating amounts available at different future times together. If we want to have one dollar in the bank account at time T, we need to invest

$$\frac{B(t)}{B(T)} = e^{-\int_t^T r(s)ds} \tag{1.6}$$

in the bank account at time $t \leq T$. This gives us another discount factor that is stochastic and unknown all the way up until time T since it depends on the evolution of the short rate. Finally, we note that the bank account and the short rate is closely connected to the discount factor given by $P(t,T)$ which in turn is already known at time t. This can be established under the risk-neutral probability measure where $P(t,T)$ becomes the conditional expectation[2] of (1.6).

$$P(t,T) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-\int_t^T r(s)ds}\right] = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\right]. \tag{1.7}$$

In the following section we build upon the zero coupon bond by introducing some linear products and different dervivatives on them.

[2]Throughout this thesis we work under the assumption that the market is friction-less (i.e. all assets are continuously traded at all sizes both long and short and without any transaction costs), complete and free of arbitrages. Thus we consider the equivalent "risk-neutral" measure, $\mathbb{Q} \sim \mathbb{P}$, which ensures all prices are martingales when discounted by the bank account (numeraire). Likewise, $\mathbb{P}$ denotes the "real world" measure under which market observables are actually quoted and traded.

2 Interest Rate Products

Prior to detailing the framework of this project, we first present a range of interest rate derivatives that will be featured in the subsequent sections. Among these products, some are primarily employed for illustrative and explanatory purposes, whereas others are chosen with the purpose of demonstrating the versatility and/or limitations of our framework and techniques.

2.1 Swaps

Interest rate swaps (or swaps) defines a financial contract between to parties to exchange cashflows on a predetermined set of fixings dates. One party agrees to pay a predetermined fixed rate on a notional principal over the specified time horizon. In return they receive a floating rate payment on the same notional over the same time period from the other party.

Swaps were developed to increase financial flexibility from exploiting different borrowing markets available to different companies. To clarify why swaps are useful we present a simple example below.

Example 2.1. Consider two companies A and B that have different borrowing opportunities for fixed and floating rates; Company A is borrowing with a fixed rate of $5\frac{1}{2}\%$ for five years, but is alternatively able to borrow at a floating rate of the LIBOR + $\frac{1}{2}\%$. On the contrary, company B is borrowing at a floating rate equal to the LIBOR + 1%, but able to alternatively borrow at a fixed rate $6\frac{1}{2}\%$ for five years.

Agreeing to swap cash flows could make both companies better off and also allowing an institution, I, to benefit from mediating the contract as follows:

- Company A pays LIBOR to the intermediary and receives fixed at $5\frac{3}{16}\%$ (that is a *receiver swap*). Company A now pays the LIBOR rate + $5\frac{1}{2}\%$ but receives $5\frac{3}{16}\%$ which amounts to the LIBOR rate + $\frac{5}{16}\%$. This is compared to the LIBOR rate + $\frac{1}{2}\%$.
- Company B pays the intermediary a fixed rate at $5\frac{5}{16}\%$ and receives the LIBOR rate (that is a *payer swap*) which in total amounts to paying $6\frac{5}{16}\%$ instead of $6\frac{1}{2}\%$.

- Intermediary then receives a fixed rate of $\frac{1}{8}\%$.

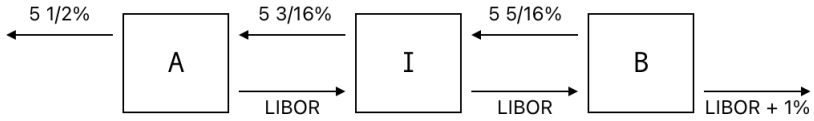


Figure 1: Example of an interest rate swap contract between two companies and an intermediary

Figure 1 above, summarizes the example and shows how all 3 parties can potentially be better off by entering the swap contract. ◦

From the example above, we can further formalize the definition of a swap contract. A *payer swap* contract - which obligates the holder to pay the fixed rate - needs to hold the following specifications:

- A tenor structure, $T_0 < T_1 < \cdots < T_n$, that is an increasing sequence of future dates. We will without loss of generality also assume that the fixing dates are equidistant, i.e. $T_i - T_{i-1} = \delta$.
- The fixed rate K to be payed by the holder,
- And a notional value N.

Cashflows take place at the coupon / fixing dates $T_1, ..., T_n$, where the holder of the contract pays a fixed rate K on the notional δN and receives a floating rate $F(T_{i-1}, T_i)$ payment on the notional δN. This amounts to the net cash flow at time T_i:

$$(F(T_{i-1}, T_i) - K)\delta N.$$

Note for practical purposes that the last fixing date is T_{n-1} such that the last settlement is made at time T_n. By definition of the prevailing simple market interest rate, the floating rate coupon payments can be rewritten as:

$$F(T_{i-1}, T_i)\delta N = \left(\frac{1}{P(T_{i-1}, T_i)} - 1\right) N.$$

We can replicate this time-T_i payoff from investing at time-t in N T_{i-1}-bonds, and with the payoff of N at time T_{i-1} we buy $\frac{N}{P(T_{i-1},T_i)}$ amount of T_i-bonds. This pays $\frac{N}{P(T_{i-1},T_i)}$

at time-T_i and constitutes a zero net investment. The time-t value of the floating rate coupon payment at time T_i is therefore given as the difference between receiving the floating coupon payment and paying the fixed rate payment, i.e.

$$(P(t,T_{i-1}) - P(t,T_i))N.$$

Inserting this in the above, we can express the time-t value of the *payer swap* cashflow at time T_i as

$$N(P(t,T_{i-1}) - P(t,T_i) - K\delta P(t,T_i)).$$

Summing over the cash flows from $T_1, ..., T_n$, we see that the total value $\Pi_p(t)$ of the payer swap at time $t \leq T_0$ is given by

$$\Pi_p(t) = N \sum_{i=1}^{n} \left(P(t,T_{i-1}) - P(t,T_i) - K\delta P(t,T_i)\right). \tag{2.1}$$

Further simplification gives our final expression for the value of the payer swap at time t, namely

$$\Pi_p(t) = N \left(P(t,T_0) - P(t,T_n) - K\delta \sum_{i=1}^{n} P(t,T_i) \right). \tag{2.2}$$

It is customary to trade interest rate swaps as contracts with zero net-preset-value when entered. This is achieved by determining the fixed rate K which ensures that the value of both the payer and receiver swap is zero, i.e., $\Pi_p(t) = \Pi_r(t) = 0$. We see that this level must be equal to below quantity and is defined as the *par swap rate* $S(t,T_0,T_n)$ at time $t \leq T_0$

$$K = \frac{P(t,T_0) - P(t,T_n)}{\delta \sum_{i=1}^{n} P(t,T_i)} =: S(t,T_0,T_n).$$

Inserting this, we can verify that the zero net-present-value condition holds true

$$\begin{aligned}
\Pi_p(t) &= N \left(P(t,T_0) - P(t,T_n) - K\delta \sum_{i=1}^{n} P(t,T_i) \right) \\
&= N \left(P(t,T_0) - P(t,T_n) - \frac{P(t,T_0) - P(t,T_n)}{\delta \sum_{i=1}^{n} P(t,T_i)} \delta \sum_{i=1}^{n} P(t,T_i) \right) \\
&= 0 = \Pi_r(t).
\end{aligned}$$

As a final rewriting, we can also use the definition of the simple forward rate, $F(t, T_{i-1}, T_i)$, express the payer swap's value a combination of discounted forward rates over the swap rate. That is, we first write the simple forward rate as

$$\delta F(t, T_{i-1}, T_i) P(t, T_i) = P(t, T_{i-1}) - P(t, T_i)$$

which, we can insert into equation (2.1) to alternatively express the time $t \leq T_0$-value as

$$\Pi_p(t) = N\delta \sum_{i=1}^{n} P(t, T_i)(F(t; T_{i-1}, T_i) - K). \tag{2.3}$$

An important observation to make on the current value of the swap rate is that it is not affected by the dynamics of interest rates, but only the current levels. Stated differently, we see from above equation (2.3) that we can determine the time-t value completely using only observations of the current term structure - hence the swap is a *linear product*.

We see from equation (2.3) that the swap rate can also be expressed as the weighted average of simple forward rates

$$S(t, T_0, T_n) = \sum_{i=1}^{n} \omega_i(t) F(t; T_{i-1}, T_i), \quad \omega_i(t) = \frac{P(t, T_i)}{\sum_{j=1}^{n} P(t, T_j)}. \tag{2.4}$$

We note that these weights are stochastic processes over time, t.

Finally, we end the introduction of interest rate swap contracts by summarizing: Swaps are financial products, which are traded other-the-counter (OTC). The contracts require a bilateral agreement of the par swap rate, a notional amount, a set of fixing dates, and payer/receiver positions. These markets are highly liquid and interest rate swaps make up majority of the global OTC market with a daily turnover of almost 4.5 trillion USD as of April 2022.[3]

[3] https://www.bis.org/statistics/rpfx22_ir.pdf

2.2 Caplets

A caplet is an interest derivative where the underlying is the continuously compounded simple forward rate F. The payoff for a given reset date T, settlement date $T+\delta$, and a strike rate K is

$$\delta \left(F(T, T+\delta) - K\right)^+ .$$

Hence, it is mirroring the payout structure of a European call option. Caps are formed from a series (also called a *strip*) of caplets. Or more precisely, by arranging future dates $T_0 < T_1 < \ldots T_n$, at equal intervals where $T_i - T_{i-1} = \delta$, resulting in a cash flow of

$$\sum_{i=0}^{n-1} \delta \left(F(T_i, T_{i+1}) - K\right)^+ .$$

We use the naming convention as a 1Y3M caplet, indicating an expiry of $T = 1$ year and a coverage period of $\delta = 3M$. Similarly, a 1Y6Y3M cap refers to the aggregate of caplets spanning a total of 5 years (from year 1 to 6) with quarterly reset dates. The price of a caplet at time t, denoted as $Cpl(t; T, T+\delta)$ for a reset date T and settlement date $T+\delta$ leads to a time-t price of a Cap with maturity T_n being

$$Cap(t; T_0, T_n) = \sum_{i=0}^{n-1} Cpl(t; T_i, T_{i+1}).$$

A cap primarily serves as a hedging tool against rising interest rates with several optionalities - one for each underlying caplet. It functions by setting a maximum limit on the interest rate on a floating rate liability, thereby providing a safeguard against interest rate volatility, which is extensively used by large financial institutions for mitigating risk. In terms of pricing, the initial value of a caplet, denoted $Cpl(0; T, T+\delta)$, is determined through risk-neutral valuation, i.e.

$$Cpl(0; T, T+\delta) = \mathbb{E}^{\mathbb{Q}} \left[e^{-\int_0^T r(s)ds} \delta \left(F(T, T+\delta) - K\right)^+ \right].$$

This reflects the present value of risk-neutral expected future payoff. The caplet is termed *at-the-money* when its strike rate aligns with the prevailing par swap rate i.e. $K = S(0, T_0, T_n)$. The risk-neutral caplet price can also be expressed in a different way. By

inserting the definition of the compounded forward rate, we can rewrite to get

$$\begin{aligned}
Cpl(0;T,T+\delta) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\delta\left(F(T,T+\delta)-K\right)^+\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\delta\left(\frac{1}{\delta}\left\{\frac{1}{P(T,T+\delta)}-1\right\}-K\right)^+\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\left(\frac{1}{P(T,T+\delta)}-1-K\delta\right)^+\right] \\
&= \frac{\bar{K}}{\bar{K}}\mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\left(\frac{1}{P(T,T+\delta)}-\frac{1}{\bar{K}}\right)^+\right], \quad \bar{K}=\frac{1}{1+\delta K} \\
&= \frac{1}{\bar{K}}\mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\left(\bar{K}-P(T,T+\delta)\right)^+\right]. \qquad (2.5)
\end{aligned}$$

This demonstrates that the price of a caplet on a forward rate is analogous to the price structure of a European put option on a zero-coupon bond with a modified strike. This structural framework becomes beneficial in the hedge experiments conducted later in section 7.

2.3 Swaptions

2.3.1 European

Options on interest rate swaps are called *swaptions*. A European payer swaption gives the right, but not the obligation to enter into a payer swap at the swaption expiry. This will usually coincide with the first reset date of the underlying swap.
We recall, that the value of a payer swap with fixed rate K at the first reset date T_0 is given by (2.1) as

$$\Pi_p(T_0,K) = N\sum_{i=1}^{n} P(T_0,T_i)\delta(F(T_0;T_{i-1},T_i)-K).$$

In the swaption contract the holder will choose to exercise only if this value is positive. This is determined by the swaption payoff at expiry which is given by

$$N\left(\sum_{i=1}^{n} P(T_0,T_i)\delta(F(T_0;T_{i-1},T_i)-K)\right)^+ .. \qquad (2.6)$$

We term an European payer swaption with expiry in $T = 1$ year, with underlying swap maturing 5 years after expiry with coverage $\delta = 3M$, as an $1Y6Y3M$ European payer swaption. Notice an important distinction to caps/floors type interest-rate derivatives. We saw above how these can be decomposed into the sum of the underlying caplets/floorlets where each term depends on a single forward rate. Because of the $(\,\cdot\,)^+$-operator it is not possible to decompose (2.6) into more elementary products. That means that the swaption has fewer optionalities compared to the caps but also that we have to account for the stochastic dependency between different forward rates when valuating swaptions. The value at time $t \leq T_0$ is then given by

$$V_p^{EU}(t) := Swpt(t) = \mathbb{E}_t^{\mathbb{Q}}\left[Ne^{\int_0^{T_0} r(s)ds}\left(\sum_{i=1}^{n} P(T_0,T_i)\delta(F(T_0;T_{i-1},T_i)-K)\right)^+\right]. \quad (2.7)$$

Since we can express the swap rate as a weighted average of simple forward rates from equation (2.4)m we can rewrite the above time-t value in a more compact form, namely

$$Swpt(t) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{\int_0^{T_0} r(s)ds}A(T_0,T_n)(S(T_0,T_n)-K)^+\right] \quad (2.8)$$

where we use the definition of the *annuity factor* given by

$$A(T_0,T_n) := \sum_{i=1}^{n} \delta_i P(T_0,T_i). \quad (2.9)$$

Notice that the annuity factor $A(T_0,T_n) > 0$ which means a payers swaption can be seen as being similar to a call option on the swap rate itself.

2.3.2 Bermudan

A Bermudan swaption gives the holder the right, but not the obligation, to enter into a swap contract on a discrete set of exercise dates $\tau_1, ..., \tau_k$, whereas a regular swaption only give this right on a single date (often the first fixing date).

A typical formulation of the Bermudan swaption contract assigns the same end date T_n to all the underlying swaps (in that case we say they are *co-terminal*). Notice that in contrast to typical callable equity products, the underlying swap is different for each exercise time. Formally the value of a co-terminal Bermudan payer swaption[4], if exercised at time τ, is the value of the underlying payer swap at this time, i.e.

$$V_{Berm}(\tau) = \Pi_p(\tau) = \sum_{\tau \leq T_i \leq T_n} N \cdot P(\tau, T_i) \cdot \delta \cdot (F(\tau, T_i, T_n) - K)\,.$$

With this formulation in mind, it is natural to think of a Bermudan swaption as a collection of European co-terminal swaptions with the same strike, where only one of them can be exercised. This insight gives us also an immediate lower bound for the Bermudan swaption. It must simply be worth at least as much as any of the constituent European swaptions (with a weak equality because of the optionality). Taking the price of the most expensive European option spanned by a given Bermudan tenor structure, must then serve as a lower bound of the Bermudan swaption.
Following the same logic, a simple upper bound for the Bermudan swaption will be to buy each of the constituent European swaptions, whereas you will only have the optionality to enter into one of these by holding the Bermudan. Summarizing, we have the following bounds

$$\max_k \{Swpt(t, T_k, T_n)\} \leq V_{Berm}(t) \leq \sum_k Swpt(t, T_k, T_n).$$

Unlike European products, the Bermudans are callable and can thus not be priced using the traditional Risk-Neutral Pricing Theorem. Instead the problem is formulated as an optimal stopping problem, where the holder of the option is assumed to exercise at the optimal stopping time τ^*. This is discussed further in a general context within subsection

[4]The holder of a payer (receiver) Bermudan swaption on an exercise time, t, pays the fixed (floating) leg of the swap.

4.6 where we formulate an algorithm for solving this problem.

2.3.3 Basket

A different kind of product that will be investigated is a basket swaption. It has some features which are similar to the classical European swaption, however, instead of having only a single swap as its underlying, it has an entire collection swaps as its underlying - each with a potential different weight - which is know as the *basket*. The optionality is on the entire basket, and the basket option thus allows an investor to bet on the relative performance of the different underlying swaps without having any exposure to the downside expect for the initial premium of the option. This is achieved by carefully choosing the right combination of weights in the underlying swaps. The price of an European basket swaption is given by

$$V^{EU}_{Basket}(t) = \mathbb{E}^{\mathbb{Q}}_t \left[\frac{B(t)}{B(T)} \left(\sum_i w_i \Pi_i(T) - K_{Basket} \right)^+ \right],$$

where the weights, w_i, are used to determine both the relative weight of each of the swaps and whether the it is a payer or receiver swap (through their sign). The strike of the basket K_{Basket} is used to determine mow much the value of the basket must increase in order to be in-the-money. Finally, as the underlying swaps are often correlated, the valuation of these requires a framework that can capture the joint distribution.

2.3.4 Barrier

A third extension of the European swaption is the barrier swaption. To be more explicit, we consider the particular case of an up-and-out swaption. The barrier options have a predetermined set of observation (or monitorization) times where, $\{\tau\}$, the contract's payoff is contingent on whether or not it has breached above the barrier, $B > 0$. The monitorization times are often either only at the time of expiration, $\{\tau\} = \{T\}$, a discrete set of dates $\{\tau\} = \{\tau_1, \ldots \tau_n\}$, or the entire continuum of times from today to the expiry, $\{\tau\} = [t, T]$. The payoff of the barrier is similar to the European swaption if the barrier B has not been breach at any of the monitorization times, i.e.

$$\Pi_p(t) < B, \quad \forall t \in \{\tau\}. \tag{2.10}$$

However, if the barrier is breached at any of the observations times then the payoff is zero [5]. The risk-neutral price of the payer barrier swaption at time $t \leq T$ is thus

$$V^{Barr}(t) = \mathbb{E}_t^{\mathbb{Q}} \left[e^{-\int_t^T r(s)ds} \left(\Pi_p(T_0, K)\right)^+ \mathbb{1}_{\{\Pi_p(t) \leq B,\, \forall t \in \{\tau\}\}} \right].$$

We emphasize that the barrier options are in general *path-dependent* which make the evaluation more difficult that European options. Luckily, this is easily solved by our Monte Carlo engine which we present in further detail in section 4.

[5] In practice, there exist many alternative rules in the case the barrier has been breached for determining the payoff. Other common alternatives include giving a fraction fixed of proportional amount to either the underlying or the original premium of the Barrier option. This amount is known as the *rebate* as if often denoted R. We will, however, without the loss of generality assume that the rebate is always zero, i.e. $R = 0$.

3 The Vasicek Model

The term structure of interest rates represents the interrelationship between interest rates of different maturities. It provides insights into future market expectations and economic conditions and is, commonly visualized through the yield curve. Traditional interest rate modeling focused on the one-dimensional behavior of the instantaneous short rate process, denoted as r. Directly modeling these dynamics is beneficial as it facilitates the explicit calculation of essential financial metrics, such as bonds and different rates. These can be derived through no-arbitrage principles as expected functions of the short rate process. The existence of a risk-neutral measure, $\mathbb{Q}$, implies that the price of a contingent claim at time t, V_t, with payoff at time T, Φ_T is

$$V_t = \mathbb{E}_t^{\mathbb{Q}} \left[e^{-\int_t^T r(s)ds} \Phi_T \right].$$

Clearly from equation (1.2) identifying the distribution of $\exp\left\{-\int_t^T r(s)ds\right\}$ enables the computation of bond prices, from which various rates and fixed income contracts can be derived - either directly by rewriting or through risk neutral pricing. The entire yield curve becomes attainable through the distributional properties of the short rate.

3.1 Short-rate Dynamics

To derive such distributional properties requires a particular specification of the short-rate dynamics. Generally we assume the instantaneous short rate dynamics evolves under the real-world measure $\mathbb{P}$ as

$$dr(t) = \mu(t, r(t))dt + \sigma(t, r(t))dW^{\mathbb{P}}(t), \tag{3.1}$$

where $\mu(t, r(t))$ and $\sigma(t, r(t))$ denote the drift and diffusion (or volatility) processes of the short rate, respectively, and $W^{\mathbb{P}}(t)$ is a Brownian motion under the real-world measure. One can show the existence of a process, λ, such that if the bond price behaves according to the short rate

$$dP(t,T) = \mu^*(t, r(t))dt + \sigma^*(t, r(t))dW^{\mathbb{P}}(t),$$

then

$$\lambda(t) = \frac{\mu^*(t, r(t)) - r(t)P(t,T)}{\sigma^*(t,T)}.$$

Here μ^* and σ^* are again the drift and volatility but this time for bond price. Additionally an equivalent probability measure $\mathbb{Q} \sim \mathbb{P}$ is established through the Radon-Nikodym derivative

$$L(t) = \frac{d\mathbb{Q}}{d\mathbb{P}} = e^{-\frac{1}{2}\int_0^t \lambda^2(s)ds - \int_0^t \lambda(s)dW^{\mathbb{P}}(s)}.$$

Utilizing the Girsanov theorem, the relationship between the measures is established as

$$dW^{\mathbb{P}}(t) = dW^{\mathbb{Q}}(t) - \lambda(t)dt.$$

Consequently, the short rate under $\mathbb{Q}$ evolves according to the following dynamics

$$\begin{aligned} dr(t) &= \mu(t, r(t))dt + \sigma(t, r(t))dW^{\mathbb{P}}(t) \\ &= \mu(t, r(t))dt + \sigma(t, r(t))\left(dW^{\mathbb{Q}}(t) - \lambda(t)dt\right) \\ &= \left[\mu(t, r(t)) - \lambda(t)\sigma(t, r(t))\right]dt + \sigma(t, r(t))dW^{\mathbb{Q}}(t). \end{aligned}$$

Transitioning from the real-world measure to the risk-neutral measure is crucial for derivative pricing as it ensures arbitrage-free (and potentially unique) prices due to First and Second Fundamental Theorem of Asset Pricing. This transition is characterized by the market's specification of λ which is referred to as the *market price of risk*. Here r denotes the instantaneous return on a risk-free investment, thus $(\mu - r)/\sigma$ is the risk adjusted excess return on an investment, also known as the *sharpe ratio*. Often, only the risk-neutral dynamics are stated, such that the choice of λ is working implicitly in the background. An exemplification could be that the short rate develops according to the SDE

$$dr(t) = \left[ab - (a + \lambda\sigma)r(t)\right]dt + \sigma dW^{\mathbb{P}}(t), \quad r(0) = r_0.$$

In order to change to the risk-neutral measure, a particular choice of the market price of risk is required. By assuming it to have the functional form $\lambda(t) = \lambda r(t)$ the dynamics under $\mathbb{Q}$ becomes

$$dr(t) = a\left[b - r(t)\right]dt + \sigma dW^{\mathbb{Q}}(t), \quad r(0) = r_0. \tag{3.2}$$

When the short rate evolves according to (3.2) the model is characterized as the *Vasicek Model*. The interpretation of the time-homogeneous parameters are rather simple; b is the long-term level of the short rate, a denotes the *mean-reversion rate* and dictates the speed at which the short-rate can be expected to revert to its long-term level, and σ represents (constant) instantaneous volatility. The Vasicek model is often mentioned as a pioneering cornerstone in the field of interest modelling, setting a precedent for subsequent interest rate models with its mean-reverting feature and analytical tractability.

3.2 Distributional Properties and Bond Pricing

The analytical tractability arises from its distribution properties. Using the transformation $f(r,t) = re^{at}$ and applying Itô's lemma, we get

$$\begin{aligned} df(r,t) &= ar(t)e^{at}dt + e^{at}dr(t) \\ &= ar(t)e^{at}dt + e^{at}\left\{a\left[b - r(t)\right]dt + \sigma dW^{\mathbb{Q}}\right\} \\ &= abe^{at}dt + \sigma e^{at}dW^{\mathbb{Q}}(t). \end{aligned}$$

By integrating from 0 to t yields

$$r(t)e^{at} = r(0) + \int_0^t abe^{as}ds + \int_0^t \sigma e^{as}dW^{\mathbb{Q}}(s).$$

This lets us rewrite and solve for $r(t)$ as a function of the current state, $r(0)$, and the development of the Wiener process, i.e.

$$\begin{aligned} r(t) &= r(0)e^{-at} + b\left[e^{a(s-t)}\right]_{s=0}^{s=t} + \sigma\int_0^t e^{a(s-t)}dW^{\mathbb{Q}}(s) \\ &= r(0)e^{-at} + b\left(1 - e^{-at}\right) + \sigma\int_0^t e^{a(s-t)}dW^{\mathbb{Q}}(s). \end{aligned} \tag{3.3}$$

From this equation it is evident that the short rate is normally distributed with mean and variance given by

$$\mathbb{E}^Q\left[r(t)\right] = r(0)e^{-at} + b\left(1 - e^{-at}\right), \quad \mathbb{V}ar^Q\left[r(t)\right] = \frac{\sigma^2}{2a}\left(1 - e^{-2at}\right). \tag{3.4}$$

This implies a positive probability of achieving negative interest rates. This is not a "bug" but a "feature". However, for risk-free bond pricing, the model's log-normal distribution properties are advantageous. Bonds can be priced either through distribution properties in expectation or using the model's property as part of the affine term structure (ATS) family.

The Vasicek model provides an ATS if the functions $A(t,T)$ and $B(t,T)$ satisfy certain ordinary differential equations (ODEs), known as Riccati equations:

$$\partial_t A(t,T) = \frac{1}{2}\sigma^2 B^2(t,T) - abB(t,T), \qquad A(T,T) = 0 \tag{3.5}$$

$$\partial_t B(t,T) = aB(t,T) - 1, \qquad B(T,T) = 0. \tag{3.6}$$

Solving these equations, one can derive explicit expressions for $A(t,T)$ and $B(t,T)$, enabling the computation of bond prices through

$$P(t,T) = e^{-A(t,T)-B(t,T)r(t)}. \tag{3.7}$$

This method provides an exact pricing formula, a significant advantage of the Vasicek model. The solutions to the Ricatti equations are derived in appendix A resulting in

$$B(t,T) = \frac{1-e^{-a(T-t)}}{a},$$
$$A(t,T) = \left(b - \frac{\sigma^2}{2a^2}\right)[(T-t) - B(t,T)] + \frac{\sigma^2}{4a}B^2(t,T).$$

Here A is clearly expressed in terms of B. Beyond showing that a closed-form bond pricing is indeed possible, the model's yield curve shapes follows certain characteristics. By defining the yield as $y(t,T) := -\log(P(t,T)/(T-t)$, we notice a finite limit exists

$$y_\infty = \lim_{T\to\infty} y(t,T) = b - \frac{\sigma^2}{2a^2}.$$

Ultimately the model provides one of three different shapes for the yield curve depending on the parameters. Whenever these scenarios occur can be seen mathematically by carrying out cumbersome computations. We do not present the calculations here but instead summarize the results according to [AP10a] as follows:

The yield curve is

- *Upward sloping* (normal): $y(t,T)$ increases as $T \to \infty$ when $r < y_\infty - \frac{\sigma^2}{4a^2}$.
- *Downward sloping* (inverted): $y(t,T)$ decreases as $T \to \infty$ when $r > b$.
- *Otherwise* (humped shaped): $y(t,T)$ increases and then flattens in all other cases.

Despite the model's capability to estimate three distinct curves, it is widely recognized that it fails to calibrate accurately to market-observed yield curves. This limitation stems from the model's rigid assumptions and lack of flexibility. However, its analytical tractability is invaluable for visualization purposes and in introducing novel concepts, aligning with the objectives of this project.

4 Monte Carlo Methods

Monte Carlo methods are a class of algorithms where realizations of random variables with known probability distribution are repeatedly sampled from a sample space, $\omega \in \Omega$, in order to numerically estimate results for problems with a probabilistic nature.

In the field of mathematical finance, Monte Carlo methods are often used to solve problems involving the distributional properties of dynamic stochastic processes within a specified model. Common applications include examples such as pricing, hedging and risk management of financial products and their derivatives. Monte Carlo methods are traditionally preferred for valuing options with a complex payoff structure, where a closed-form price solution does not exists. We introduce below the general framework and extend the methodology later on to obtain not only price estimates, but also sensitivities.

4.1 Design Pattern - Segregation of Products and Models

In order to establish a general framework for accessing these problems with Monte Carlo methods, we must first generalize the notion of a financial product. We do this with a high degree of abstraction to be able to consider any product that can be formulated in terms of it's cashflows. This design pattern is inspired by the approach taken in [Sav19]. As described in section 2, products are essentially a collection of cashflows at predetermined times. Formally, we denote the set of cashflows for a product as

$$\{CF(t_i),\ 0 \leq t_i \leq T\}\,,$$

where t_i denotes each of the payment dates and T denotes the last date on the product's timeline, which we denote as TL. Typically, if there are no deferred payments, then T is the expiry or maturity of the product. Otherwise, if there are deferred payments, then T would most likely be the time of the last payment. The cashflows may be deterministic or stochastic depending on the specific product. In the stochastic case, they have either a linear, optional, or exotic dependency on some set of specific market variables. Finally, the cashflows may furthermore not necessarily take place at the same time as they are determined, i.e. become measurable. As such the cashflows, in general, depend both on time and state.

The union of the cashflows' payment times and the times at which they become measurable are referred to as the product's *event dates*. In the context of the Monte Carlo methods the event dates play an important role as these are exactly the times at which the model must sample something. For this, we note the important and useful segregation between models and products when dealing with the above mentioned financial problems:

- The product is responsible for determining *when* and *what* is sampled in each scenario generated. This also ensure that the necessary market variables are available for determining the cashflows needed in the valuation.
- The model is in turn responsible for *how* the market variables are generated on the product's event dates in each scenario from the sample space.

We implement our Monte Carlo engine with this design pattern noting that the term *scenario* is a collection of market variables and discount factors sampled at certain times.

This design pattern also constitutes a natural link to the financial theory outlined in the former sections. More specifically, Monte Carlo methods in the setup described here relies on the Law of Large Numbers and Central Limit Theorem to solve the pricing problem governed by the Risk Neutral Pricing Theorem. To see this, we recall its statement specifying that the price, V, of a financial claim, is the expected value of the claim's numeraire deflated (i.e. discounted) cashflows. Often, the starting point is using the Risk Neutral Pricing Theorem under the risk neutral measure, $\mathbb{Q}$, where the numeraire is the bank account, B. Depending on the product it may be beneficial for a number of reasons to change measure which alters both the measure under which the stochastic processes live but also the numeraire. Handling this falls within the responsibility of the model and its implementation when using the segregation above. For some products a change of measure can impact the computation time quite a bit which we will revert back to in later sections.

4.2 Monte Carlo Integration

We let the market observables which the product's cashflow depend on, be denoted as $\boldsymbol{x}(\omega) := \{x_1(\omega), ..., x_n(\omega)\}$ for a given state $\omega \in \Omega$. Hence emphasizing we are modelling

the market variables as (stochastic) processes as functions of the states from the sample space $\omega \in \Omega$. Furthermore, to simplify notation, we collect the cashflows and discounting in the *discounted payoff function* denoted as $g(\boldsymbol{x})$. We can turn back to the Risk Neutral Pricing Theorem and use that the measure-theoretical definition of an expected value is an integral[6] over the state space to express the value of the claim as

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\sum_{t \le t_i \le T} \frac{B(t)}{B(t_i)} CF(t_i; \boldsymbol{x}_{t_i})\right] = \mathbb{E}_t^{\mathbb{Q}}\left[g(\boldsymbol{x})\right] = \int_A g(\boldsymbol{x}(\omega))\, d\mathbb{Q}(\omega), \quad \forall A \in \mathcal{F}_t, \quad (4.1)$$

where t_i denotes the times where payments are made and $\boldsymbol{x}_{t_i}$ denotes the market observables up until time t_i corresponding to some state $\omega \in \mathcal{F}_{t_i}$.

When $\Omega \subseteq \mathbb{R}^k$, $k \in \mathbb{N}$, the above problem can be approximated by Monte Carlo integration where the expectation in the form of its integral is essentially approximated by discretization. In order to obtain an approximation of the distribution the simulation algorithm starts from independent and identically distributed uniformly sampled random numbers, $U \sim \mathcal{U}[0,1]^k$, which can be one-dimensional but also multi-dimensional depending on the problem. These random variables are then transformed as needed through the simulation algorithm in order to obtain the desired distribution. Exactly how such an algorithm is chosen is discussed further in subsection 4.3. Also, in this project we simply take the uniformly distributed random variables for granted and note that they are easily transformed into Gaussian i.i.d. random variables through an approximation of the inverse normal distribution, $Z = N^{-1}(U)$, $U \sim \mathcal{U}[0,1]$. Again, the simulation algorithm implemented for the model is responsible for transforming the Gaussian random variables into realizations/samples of the market variables collected into a scenario for each path, which eventually form the distribution that enables us to obtain the Monte Carlo estimate.

In the most naive form $N \in \mathbb{N}$ realizations are sampled i.i.d. to calculate an the Monte Carlo estimator, $\hat{V}$, which is defined as follows

$$V = \int_\Omega g(\boldsymbol{x}(\omega)) d\mathbb{Q}(\omega) \approx \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}_i) =: \hat{V}. \quad (4.2)$$

[6] In the discrete case, the integral can be replaced by its discrete version $\sum_{\omega\in\Omega} g(\boldsymbol{x})(\omega)\mathbb{Q}(\omega)$.

The last equation emphasizes that the Monte Carlo estimator in its simplest form is a just an arithmetic average taken over the realized discounted payoffs from the sampled market variables. Furthermore, the Strong Law of Large Numbers ensures that the Monte Carlo estimator convergences almost surely with probability 1 to the analytical expectation, that is

$$\hat{V} \xrightarrow{a.s.} V \quad , \quad N \longrightarrow \infty. \tag{4.3}$$

However, as simulations are generally computational expensive, any practical usage requires not only convergence in the limit but also sufficiently precision within a finite number of samples. Proposition 4.1 below shows the convergence order for the variance of the Monte Carlo estimator and its standard error.

Proposition 4.1 (Convergence order of the Monte Carlo estimator)**.** The convergence order of the variance and standard error of the Monte Carlo estimator are respectively proportional to N^{-1} and $N^{-1/2}$, where $N \in \mathbb{N}$ denotes the number of sampled realizations. That is

$$\operatorname{var}\left(\hat{V}\right) = \mathcal{O}\left(N^{-1}\right) \quad , \quad \operatorname{SE}\left(\hat{V}\right) = \mathcal{O}\left(N^{-1/2}\right). \tag{4.4}$$

Proof. The result follows straight forward from the definition of the Monte Carlo estimator by using the properties of variance and the i.i.d. property of the samples:

$$\begin{aligned}\operatorname{var}\left(\hat{V}\right) &= \operatorname{var}\left(\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}_i)\right) = \frac{1}{N^2}\operatorname{var}\left(\sum_{i=1}^{N} g(\boldsymbol{x}_i)\right) \\ &\overset{i.i.d.}{=} \frac{1}{N^2}\sum_{i=1}^{N}\operatorname{var}\left(g(\boldsymbol{x}_i)\right) = \frac{1}{N}\operatorname{var}\left(g(\boldsymbol{x}_i)\right),\end{aligned}$$

$$\begin{aligned}\operatorname{SE}\left(\hat{V}\right) = \operatorname{SD}\left(\hat{V}-V\right) &= \left[\operatorname{var}\left(\hat{V}-V\right)\right]^{1/2} = \left[\operatorname{var}\left(\frac{1}{N^2}\sum_{i=1}^{N} g(\boldsymbol{x}_i) - \mathbb{E}\left[g(\boldsymbol{x})\right]\right)\right]^{1/2} \\ &= \left[\operatorname{var}\left(\frac{1}{N^2}\sum_{i=1}^{N} g(\boldsymbol{x}_i) - 0\right)\right]^{1/2} \overset{i.i.d.}{=} \left[\frac{1}{N^2}\sum_{i=1}^{N}\operatorname{var}\left(g(\boldsymbol{x}_i)\right)\right]^{1/2} = \frac{1}{N^{1/2}}\operatorname{var}\left(g(\boldsymbol{x}_i)\right).\end{aligned}$$

□

This proposition highlights one of the major weaknesses of Monte Carlo methods as it shows that in order to reduce the error by half, we need to increase the number of sim-

ulations four-fold. Depending on the model specification and the financial product of interest, one may need to perform another discretization in the time dimension - which further increases the computational cost - in order to obtain the realization of the underlying stochastic processes. In these situations a choice has to be made whether to simulate the realizations path- or step-wise. As discussed in [Sav19] path-wise implementations has a number of performance advantages such as the allowance for vectorization of computations across paths and several techniques for variance reductions. However, a step-wise implementation may also give rise to difficulties for path-dependent products and differentiation algorithms such as AAD which we will introduce in later section 5.

4.3 Putting the Model in the Engine - From SDE to Samples

In order to make practical use of the Monte Carlo methods, a specification of how the realizations are drawn / sampled is required. That is, we need to specify, how to simulate in practice. There exists several different approaches and their applicability depend partly on the model specification and the financial products of interest. The model specification determines the distribution of the scenarios / events in the sample space Ω while the financial products determine the value of the products in different states. From a computational viewpoint it is thus the model's responsibility to draw samples in states that are compatible with the products of interest.

In general an analytical expression for the distribution or transition probabilities may not be available, just as they may be numerical unstable from a computational point of view. Furthermore, the financial products may have several complicating features such as the possibility of early exercises, path-dependency (such as barriers), multiple strikes, multiple underlying assets, and more. All of this complicates the computations and are factors we need to consider when performing the Monte Carlo simulations.

For our purpose, we consider models that can be defined from the solution $X(t)$ to a set of $n \in \mathbb{N}$ stochastic differential equations (SDE) of the form

$$dX(t) = \mu(t, X(t))dt + \sigma(t, X(t))dW(t), \quad 0 \leq t \leq T \tag{4.5}$$

where W is a m-dimensional standard Wiener process with independent components. Furthermore, $\mu : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is known as the drift function while $\sigma : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the diffusion function. The processes X may not necessarily itself be market observable variables, $\boldsymbol{x}$, but are instead the so-called *state variables* which trough a functional mapping

$$\zeta : \{X_t, 0 \leq t \leq T\} \to \{\boldsymbol{x}_t, t \in TL\}$$

determine the market accordingly to the model specification. If the model directly specifies the dynamics of the market variables then ζ is simply the identity mapping. In general, we will refrain from focusing too much on the difference in our notation in order to avoid unnecessary clutter as the distinction between state and market variables should always be clear from the context. Instead we simply note that one has to specify such a mapping when implementing the model which specifically for this project involve deriving the bond-reconstruction formula as interest rate derivatives are considered.

We are however still interested in knowing how to specify the evolution of the stochastic diffusion processes - regardless of whether it is market observables or observable state variables - in equation (4.5) that are being sampled by the model. For this, we turn to one of the most straightforward and explicit simulation schemes for SDEs of this type, namely the *Euler–Maruyama method* (also simply referred to as Euler discretization). Given a discretized timeline from t_0 to T of the form $0 = t_0 < t_1 < ... < t_M = T$ and the initial value of the state variables $X(t_0)$, each of the $i = 1, ..., n$ elements the process X evolves recursively according to

$$X^i(t_{k+1}) = X^i(t_k) + \mu^i(t_k, X_{t_k})\Delta t_k + \sum_{j=1}^{m} \sigma^{ij}(t_k, X_{t_k})\sqrt{\Delta t_k} Z_k^j, \quad k = 0, ..., M-1 \quad (4.6)$$

where $\Delta t_j := t_{k+1} - t_k$ and Z_k is an m-dimension vector of independent standard Gaussian variables. The strength of the Euler discretization scheme is it is easy to implement. On the contrary it comes with a relatively slow order of convergence. Even in the 1-dimensional case under slightly more strict conditions[7] of μ and σ, the strong order of convergence of the Euler discretization for SDEs is of order $\mathcal{O}(\Delta t^{1/2})$ while the weak order of convergence is of order $\mathcal{O}(\Delta t^1)$ [Sey17], [Gla03].

[7] Such as Lipschitz condition, at most linear growth and sufficient smoothness

Figure 2 below summarizes all the components necessary for assembling the Monte Carlo engine where we easily can interchange different products and models independently of each other. After the user has configured the arguments required in the basic Monte Carlo algorithm `mcBase` which include the product `prd`, the model `mdl`, the random number generator `rng`, and the number of paths `N`, the steps are as follows:

(1) The product is defined by its timeline and definition line (*"defline"*) which specifies when and what is sampled. The timeline and *"defline"* are copied into the model.

(2) The model instructs the RNG how many discretizations are needed.

(3) The user specifies how many paths should be simulated.

(4) The RNG samples the random variables and passes these to the model. The model then simulates the paths.

(5) The product's discounted payoff function is evaluated on the samples.

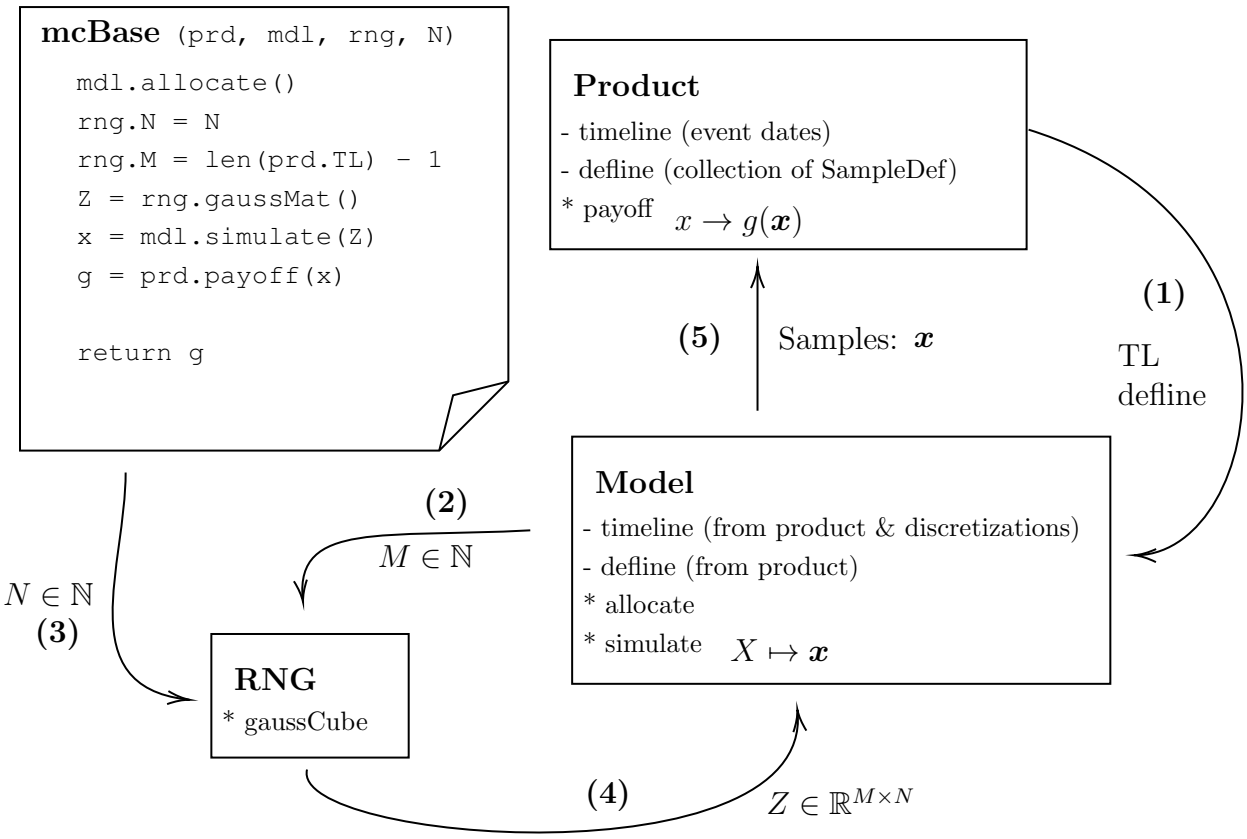


Figure 2: Overview of the Monte Carlo engine and the basic Monte Carlo algorithm `mcBase`*. The arrows represents data flow. A dash (-) indicates an attribute of the object class whereas a star (*) represents callable methods.*

There exists several other simulation schemes based on discretization which have been proposed for improving the convergence order, some of which require more conditions to be imposed upon the SDEs in equation (4.5). As an example, the Euler discretiation can be viewed in analogy of a first order Taylor expansion. Increasing the order of the expansion by Ito's lemma leads to Milstein's scheme that additioanlly requires the derivatives of σ which e.g. can be found by Runge-Kutta's method. This scheme is computationally and mathematically more complex[8] but improves the order of strong convergence to $\mathcal{O}(\Delta t^1)$ while its weak order of convergence stays unchanged from the simple Euler

[8]In multi-dimensional cases Milstein's scheme require simulation from the distribution of *Levy area* terms [Gla03].

discretization with an order of $\mathcal{O}(\Delta t^1)$ [Sey17], [Gla03].

As computational resources are scarce, it is evident that an optimal implementation seeks to avoid unnecessary repetition of similar computations and unnecessary memory allocation. In our implementation, we have sought to avoid sub-optimal routines, but we do not dwell too much on these problems as they are more related to the studies of computer science. Other kinds of optimization may also be considered such as implementing an efficient random number generator (RNG) for sampling first the uniformly distributed random variables, U that are transformed into the Gaussian random variables, Z. There are many different algorithms both of these problems such as Latin hypercube variance reduction and Sobol's quasi-random number generator as discussed in [Gla03] and [Sav19] but again we leave such considerations for potential future improvements in a low-level implementation. We do however consider other kinds optimization techniques and methods of variance reduction as outlined in subsections 4.4 and 4.5.

4.3.1 Simulating Correlated Random Variables

In applications where random variables are required to depend on each other we have to revisit the simulation step. Suppose we need to simulate N correlated Gaussian variables with given mean μ and correlation structure by the covariance matrix Σ, that is $X \sim \mathcal{N}(\mu, \Sigma)$. We can then break this simulation up in two steps.

First, we simulate the uncorrelated Gaussians $Z \sim \mathcal{N}(0, I)$, where I is the identity matrix. If we then consider X as a transformation of Z given by $X = \mu + LZ$, we must have that this is also Gaussian with mean μ and covariance matrix $LL^\top$, see [Sey17] for reference.

Secondly, we can then find the matrix L by a factorization method. A popular choice for this is Cholesky decomposition that factorizes L as a lower-triangular matrix. If we then make sure to do the decomposition as $LL^\top = \Sigma$, we can compute our correlated Gaussians by $X = \mu + LZ$.

This is in particular useful when simulating correlated Brownian motions. Consider for

simplicity the two-dimensional case where W_1, W_2 are two Brownian motions with the correlation $\langle W_1, W_2 \rangle = \rho$. The covariance matrix thus becomes symmetric and positive definite:

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

Namely in the case of positive definiteness we find the Cholesky method applicable to decompose the covariance matrix. By following the outline above, and making use of the property, that Brownian motions have Gaussian increments, we generate a random vector $Z \sim \mathcal{N}(0, I)$, do a Cholesky decomposition of the covariance matrix $\Sigma = LL^\top$, and then obtain our correlated random vector $X = LZ \sim \mathcal{N}(0, \Sigma)$.

4.4 Aligning Deferred and Multiple Payments

The following results are presented in the context of risk neutral pricing to be specific on how we can apply it in our case. None the less, they hold true in general for any pair of equivalent martingale measure and numeraire for a measurable variable and not just under the $\mathbb{Q}$-measure with B as numeraire and a cashflow as random variable.

Proposition 4.2 (Deferred payments valued back to the time of measurability)**.** Let $t < T < T + \delta$ be three distinct times and let $CF(T + \delta, \boldsymbol{x}_T)$ be the cashflow that becomes known, i.e. measurable, at time T but paid at a deferred time $T + \delta$. The value of this cashflow at time t is

$$V_t = \mathbb{E}_t^{\mathbb{Q}} \left[\frac{B(t)}{B(T+\delta)} CF(T+\delta, \boldsymbol{x}_T) \right] = \mathbb{E}_t^{\mathbb{Q}} \left[\frac{B(t)}{B(T)} CF(T+\delta, \boldsymbol{x}_T) P(T, T+\delta) \right].$$

Proof. The results follows immediately from the Tower Property of conditional expectation as $\mathcal{F}_t \subseteq \mathcal{F}_T$

$$\begin{aligned} V_t &= \mathbb{E}_t^{\mathbb{Q}} \left[\frac{B(t)}{B(T+\delta)} CF(T+\delta, \boldsymbol{x}_T) \right] = \mathbb{E}_t^{\mathbb{Q}} \left[\mathbb{E}_T^{\mathbb{Q}} \left[\frac{B(t)}{B(T+\delta)} CF(T+\delta, \boldsymbol{x}_T) \right] \right] \\ &= \mathbb{E}_t^{\mathbb{Q}} \left[\frac{B(t)}{B(T)} CF(T+\delta, \boldsymbol{x}_T) \mathbb{E}_T^{\mathbb{Q}} \left[\frac{B(T)}{B(T+\delta)} \right] \right] = \mathbb{E}_t^{\mathbb{Q}} \left[\frac{B(t)}{B(T)} CF(T+\delta, \boldsymbol{x}_T) P(T, T+\delta) \right]. \end{aligned}$$

□

As stated in [BM06] this result essentially formalizes the intuitive economic principle

that the current value of receiving a payment of a known amount in the future can be modelled equivalently as the present value of such an amount. The second result is more useful for products that have payments at different times as it formalizes that receiving an amount that is known and paid today can be modelled equivalently as receiving the forward value of such an amount at a future date corresponding to the forward maturity.

Proposition 4.3 (Payment valued by pushing it forward)**.** Let $t < T < T + \delta$ be three distinct times, and let $CF(T, \boldsymbol{x}_T)$ be the cashflow that become known, i.e. measurable, and paid at time T then the value of such a cashflow can equivalently be expressed by the identity

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}CF(T, \boldsymbol{x}_T)\right] = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T+\delta)}\frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}\right].$$

Proof. As $\mathcal{F}_t \subseteq \mathcal{F}_T$ the results follows immediately from the Tower Property of conditional expectation

$$\begin{aligned}
V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}CF(T, \boldsymbol{x}_T)\right] &= \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}P(T, T+\delta)\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}\mathbb{E}_T^{\mathbb{Q}}\left[\frac{B(T)}{B(T+\delta)}\right]\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[\mathbb{E}_T^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}\frac{B(T)}{B(T+\delta)}\right]\right] \\
&= \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T+\delta)}\frac{CF(T, \boldsymbol{x}_T)}{P(T, T+\delta)}\right].
\end{aligned}$$

□

Phrased differently the former result gave a way of bringing promised payments from the deferred time to the time of measurability while the latter result allows us to push all payments forward in time to a common future date. Given the segregation between the model and products in our design pattern, implementing these details are fairly simple as we only need to slightly modify the specification of the products.

It should also be noted that despite their simplicity, these result are very useful for modelling certain products such as interest rate caps and floors. In fact, the combination of these two results allow us to efficiently value the product under one common measure: The terminal (forward) measure. The reason for this is that the first result allows us to

model the products as if they did not have deferred payments while the latter allows us to consider the payments at a common forward date, namely on or after the last payment date of the product. In the context of implementing a Monte Carlo engine the results, when combined, also give rise to simpler or faster simulations as all the market observables can be sampled at the time each cashflow become measurable and later valued at a common future date. Choosing a suitable measure for the problem at hand can generally reduce the variance of the Monte Carlo estimation which is briefly discussed in the following subsection.

4.5 Variance Reduction Methods

We consider two simple methods that can be implemented with relative ease in our existing Monte Carlo engine that, with an acceptable amount of additional computations, can significantly reduce the overall variance of the Monte Carlo estimator.

4.5.1 Change of Measure

As we consider problems related to the pricing of interest rate derivatives, the Monte Carlo estimator, which estimates the expectation from the Risk Neutral Pricing Theorem, needs to deal with the dependency between the numeraire and payoff function. For interest rates derivatives these often have relatively high correlation which in some cases result in estimates with large variance and slow convergence. Instead, one can solve the same problem but under a different measure. For interest rate derivatives this measure is often either a terminal/forward measure or the swap measure. The former changes the pricing problem of each cashflow to

$$V_t = \mathbb{E}_t^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}CF(T,\boldsymbol{x}_T)\right] = P(t,T)\mathbb{E}_t^{\mathbb{Q}^T}\left[CF(T,\boldsymbol{x}_T)\right], \tag{4.7}$$

where it is evident that the discounting can be taken independently outside of the expectation under the forward measure. This result naturally also holds true for products with multiple cashflows by the linearity of the expectation. We will revisit this method in later section 8.3.

4.5.2 Antithetic Variates

Another simple, yet effective, technique for reducing the variance of the Monte Carlo estimator is by using Antithetic Variates (AV) which is applied to the RNG. It works by using that the fact, that if a random variable is standard uniformly distributed, $U \sim \mathcal{U}[0,1]$, then so is $1-U$. Using the inverse distribution function (assuming it exists) on both of these random variables gives two samples from a desired distribution. Obviously, these variables are identically distribution but not independent. Specifically, we can get two standard Gaussian random variables by applying the inverse standard normal distribution to get $Z = N^{-1}(U)$ and $-Z = N^{-1}(1-U)$. These have the same size as the distribution is symmetric around the origin.

By sampling M standard uniformly random variables, $\{U_1, ..., U_M\}$, and applying the inverse standard normal distribution function to each of these, we get a set of i.i.d. standard Gaussian random variables, $\{Z_1, ..., Z_M\}$, required to sample one scenario in the Monte Carlo simulation. By using the antithetic sampling as described above, we can then easily get another path from $\{-Z_1, ..., -Z_M\}$ which are i.i.d. too. First and foremost, this saves us mostly the computational cost from processing the sampling of the random variables, but we can also reuse many of our calculations if we implement the model carefully as outlined in [Sav19].

Letting $(Y_j, \tilde{Y}_j),\ j = 1, ..., M$ denote the pair of payoffs generated from each original M and the corresponding antithetic paths, we note that the pairs are i.i.d. We can thereby calculate the antithetic estimate over $2N$ total paths (N original and N antithetic) as

$$V^{AV} := \frac{1}{2N}\left(\sum_{i=1}^{N} Y_i + \sum_{i=1}^{N} \tilde{Y}_i\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_i + \tilde{Y}_i}{2}\right).$$

The main interest in using antithetic variates is to reduce the variance of the Monte Carlo estimator without increasing the computational cost. Therefore, the first concern is to ensure that the estimator does not give worse estimates by sampling the antithetic paths in addition to only sampling the original ones. This is the essence of the following

result.

Proposition 4.4. (Additional sampling with AV does not increase variance) Sampling N paths from antithetic variates in addition to the original N paths does not result in larger variance of the Monte Carlo estimator:

$$\text{var}\left(V^{AV}\right) \leq \text{var}\left(\hat{V}\right).$$

Proof. By the definition of the antithetic variate estimator we need to show that

$$\text{var}\left(V^{AV}\right) = \text{var}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_i + \tilde{Y}_i}{2}\right)\right) \leq \text{var}\left(\frac{1}{N}\sum_{i=1}^{N} Y_i\right) = \text{var}\left(\hat{V}\right).$$

Using our former observation that both the pairs of samples within the AV estimator are i.i.d., and the samples within the original Monte Carlo estimators are i.i.d., and basic properties of the variance we see that this inequality holds if

$$\frac{1}{4}\text{var}\left(Y_i + \tilde{Y}_i\right) < \text{var}\left(Y_i\right).$$

We can now rewrite the left hand side by using Y_i and $\tilde{Y}_i$ are identically distributed to get

$$\frac{1}{4}\text{var}\left(Y_i + \tilde{Y}_i\right) = \frac{1}{4}\text{var}\left(Y_i\right) + \frac{1}{4}\text{var}\left(\tilde{Y}_i\right) + \frac{2}{4}\text{cov}\left(Y_i, \tilde{Y}_i\right) = \frac{1}{2}\text{var}\left(Y_i\right) + \frac{1}{2}\text{cov}\left(Y_i, \tilde{Y}_i\right).$$

We now use that $\left|\text{cov}\left(Y_i, \tilde{Y}_i\right)\right| \leq \frac{1}{2}\left(\text{var}\left(Y_i\right) + \text{var}\left(\tilde{Y}_i\right)\right)$ to get the desired result:

$$\text{var}\left(Y_i + \tilde{Y}_i\right) = \frac{1}{2}\text{var}\left(Y_i\right) + \frac{1}{2}\text{cov}\left(Y_i, \tilde{Y}_i\right) \leq \frac{1}{2}\text{var}\left(Y_i\right) + \frac{1}{2}\left(\text{var}\left(Y_i\right) + \text{var}\left(\tilde{Y}_i\right)\right) = \text{var}\left(Y_i\right).$$

Finally, we note that the inequality used on the covariance is easily seen by first writing the variance of a sum of two random variables (in this case Y_i and $\tilde{Y}_i$) as

$$\begin{aligned} & \text{var}\left(Y_i + \tilde{Y}_i\right) = \text{var}\left(Y_i\right) + \text{var}\left(\tilde{Y}_i\right) + 2\text{cov}\left(Y_i, \tilde{Y}_i\right) \\ \Leftrightarrow \quad & \text{cov}\left(Y_i, \tilde{Y}_i\right) = \frac{1}{2}\left[\text{var}\left(Y_i + \tilde{Y}_i\right) - \text{var}\left(Y_i\right) - \text{var}\left(\tilde{Y}_i\right)\right]. \end{aligned}$$

By using the fact that the variance cannot be negative, $\text{var}\left(Y_i\right), \text{var}\left(\tilde{Y}_i\right) \geq 0$, we get the

inequality

$$|\text{cov}\left(Y_i, \tilde{Y}_i\right)| \geq \text{cov}\left(Y_i, \tilde{Y}_i\right) \geq \frac{1}{2}\text{var}\left(Y_i + \tilde{Y}_i\right).$$

□

This result is not surprising since it relies on sampling the antithetic paths in addition to the original ones. However, in any practical implementation, we are also concerned with the computational costs. It should be evident that if the model is implemented efficiently then sampling the N additional paths with antithetic variates should require at most the same costs as sampling N additional original paths. The following result gives a condition which, when satisfied, ensures that sampling half of the paths with antithetic variates lowers the variance compared to sampling them in the original manner.

Proposition 4.5 (AV Variance reduction condition)**.** Sampling with Antithetic Variance (strictly) reduces the variance of the Monte Carlo estimator if

$$\text{cov}\left(Y_i, \tilde{Y}_i\right) < 0. \tag{4.8}$$

Proof. This result follows by the same outline of argumentation as we used in above proposition. Our goal is to find a condition for which the Monte Carlo estimator with antithetic variates has lower variance than the original Monte Carlo estimator with the same total number of simulations. By the definition of the antithetic variate estimator we need to ensure that

$$\text{var}\left(V^{AV}\right) = \text{var}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_i + \tilde{Y}_i}{2}\right)\right) < \text{var}\left(\frac{1}{2N}\sum_{i=1}^{2N} Y_i\right) = \text{var}\left(\hat{V}\right).$$

Using that the samples within both estimators are i.i.d. and basic properties of the variance statistic, we can rewrite this condition into the simpler expression

$$\text{var}\left(Y_i + \tilde{Y}_i\right) < 2\text{var}\left(Y_i\right).$$

Now, we rewrite the expression on the left hand side, using that Y_i and $\tilde{Y}_i$ are identically distributed. Thus,

$$\text{var}\left(Y_i + \tilde{Y}_i\right) = \text{var}\left(Y_i\right) + \text{var}\left(\tilde{Y}_i\right) + 2\text{cov}\left(Y_i, \tilde{Y}_i\right) = 2\text{var}\left(Y_i\right) + 2\text{cov}\left(Y_i, \tilde{Y}_i\right).$$

Inserting this into the former expression and simplifying gives the stated condition. □

4.6 Least Squares Method (LSM)

So far we have mostly considered how to handle problems related to the Risk Neutral Pricing Theorem in its simplest form. It naturally works well for linear products despite not being needed in this case, as it would be a waste of computational resources. Likewise, it is perfectly capable of handling European contracts with a single option of callability at the time of expiration. However, for European contracts Monte Carlo methods are often not needed or preferred if the problem is of low dimensionality, i.e. when the model has few factors and the product does not have more than a couple of underlyings. For these simple cases other methods such as Finite Difference or Fourier / Laplace transformations are often the weapon of choice. Monte Carlo methods are instead used for more complex models with several stochastic drivers, or when the products' payoff structure become more exotic. The latter is for instance the case when there are multiple underlying assets or path-depend features such as barriers or Asian options. These products are not a problem for our Monte Carlo engine and fits well within the setup summarized in figure 2. However, some products have other features that we need to handle slightly differently. This includes e.g. derivatives that gives the holder the right to early callability (possibly on multiple dates), either in the form of exercising or cancelling payments. In the previous section 2.3.2 we introduced the Bermudan-style callability that typically has at least one pre-defined early exercise date prior to maturity, or it could be the American option with continuous callability. For these products, the traditional Risk Neutral Pricing Theorem is extended to an optimal stopping problem

$$V_t = \sup_{t \le \tau \le T} \mathbb{E}_t^{\mathbb{Q}} \left[g(\boldsymbol{x}_\tau) \right] = \sup_{t \le \tau \le T} \mathbb{E}_t^{\mathbb{Q}} \left[\sum_{t \le t_i \le \tau} \frac{B(t)}{B(t_i)} CF(t_i, \boldsymbol{x}_{t_i}) \right],$$

where option holder exercises at the optimal stopping time τ^* according to an optimal exercise strategy. This strategy is often found or approximated through dynamic programming.

Fortunately, we can easily extend our Monte Carlo engine to handle these products by turning them from callable into path-dependent problems. This is exactly the job of

the Least Squares Method (LSM) or Least Squares Monte Carlo (LSMC) presented by Longstaff and Schwartz [LS01] where the optimal exercise strategy is approximated by training an estimator, often in the form of an Ordinary Least Squares (OLS) regressor, recursively backwards over all the exercise times. The method is split into two simulations:

- The *pre-simulations* which is a set of $n \in \mathbb{N}$ scenarios used to fit / train the regression. Essentially giving us a proxy or estimator for the optimal exercise strategy.
- The *main simulations* which is a set of $N \in \mathbb{N}$ scenarios which uses the trained regressor to give the estimated optimal exercise times. Then the N simulations are used similar to the previous subsections to evaluate the Monte Carlo estimator.

The regression is performed cross-sectionally over paths by using either the state variables or a set of market variables as covariates and the continuations value as its response variables. The algorithm is initialized by first evaluating the payoff at the last time point and then performing the recursion backwards until a set of coefficients / parameters have been obtained for all exercise times. In the main simulation the optimal exercise strategy is then to exercise whenever the exercise value exceeds the expected continuation value from the prediction of the regressor. In practical settings, such as in our implementation, we would typically choose $n < N$ to learn a (rough) sketch of the early exercise boundary, of course depending on how small n is set. This is often a good prioritization of computational resources, since fitting the regressor is the computational-bottleneck in this algorithm.

In principle any estimator could be used in above algorithm. But as usual one needs to weight the pros and cons in terms of accuracy, stability, computational resources, and speed. For our purpose we wish to use a pathwise differentiation method called AAD (this is covered in below section 5) which requires an estimator in the LSM-algorithm that is sufficiently differentiable. Because AAD can be computationally expensive - especially in terms of memory - one may need to consider optimization of the LSM implementation accordingly through the use of check-pointing as described in [HS17]. Taking these considerations into account, we choose to implement LSM with OLS regressors. One of the main advantages of this class of estimators is their closed form solution for minimizing

the Mean Squarred Error (MSE) which gives the Maximum Likelihood Estimator (MLE) under the assumption that the error terms (residuals), ε, are normally distributed with mean zero. That means in a practical setup we can find the OLS estimator from matrix manipulations which can be computed fast.

More specifically at each exercise time in the recursion, we fit an OLS-regressor as follows: Let $\mathbf{X} \in \mathbb{R}^{n \times k}$ denote the matrix of k *raw* covariates (state or market variables) for each of the n pre-simulation paths at some time t_i and $\boldsymbol{y} \in \mathbb{R}^n$ a column vector of the corresponding response variables (continuation values). That is, we have collected the observations in a row for each path. Then we transform the raw covariates into our actual covariates with a set of $B \in \mathbb{N}$ basis functions ϕ, i.e.

$$\phi := \phi(\mathbf{X}) = \{\phi_1(\mathbf{X}), ..., \phi_B(\mathbf{X})\} \in \mathbb{R}^{n \times B}.$$

Using this, we can express the regression equations of the linear model with matrix notation[9] as

$$\boldsymbol{y} = \phi\beta + \varepsilon.$$

We can then minimize the MSE which is a convex optimization problem by zeroing the gradient from differentiating with respect to the weight coefficients and solving for them. That is, we consider the problem

$$\min_{\beta} \mathrm{MSE}(\beta \mid \boldsymbol{y}, \phi) = \frac{1}{n} \min_{\beta} \|\boldsymbol{y} - \phi\beta\|_2^2, \quad \beta \in \mathbb{R}^B$$

which has the optimal solution in the form a vector of coefficients, $\hat{\beta}$, given by the normal equation

$$\hat{\beta} = \left(\phi^\top \phi\right)^{-1} \phi \boldsymbol{y}.$$

Repeating this over all but the last exercise times leaves us with a set of weight coefficient vectors which can then be loaded and used for predicting the continuation value in the main simulation.

The specific choice of basic functions can have some effect but from our former work

[9] Meaning that for each of the observations, i, we have in vector notation that $\boldsymbol{y}_i = \phi_i^\top \beta + \varepsilon_i$.

in [HHA23], we found that using polynomials with an intercept (also called bias or constant) feature together with interaction terms of lower order monomials was performing very well and similar to using Laguerre polynomials as original proposed by Longstaff and Schwartz [LS01]. For this reason, we implement a classical polynomial regressor but as within everything else in our Monte Carlo engine, it can easily be interchanged with some other estimator.

4.6.1 Speed Up and Stability with Singular Value Decompostion (SVD)

We now have everything needed to run the LSM-algorithm as a module inside our Monte-Carlo engine, to be used in the valuation of early-callable products. However, we note that the inversion performed in the normal equation can be numerical unstable - if it exists at all - especially if the covariates (columns) are close to being colinear. To tackle this, we make a few reformulations which improves the stability and performance, both in terms of memory and processing power required, without decreasing the fit or accuracy of the predictions. This is done by using Singular Value Decomposition (SVD) several times in the steps outlined above. First and foremost, we ensure that we are using the *pseudo-inverse*, which is computed using SVD, in the normal equation since a standard matrix inversion will often not exist - either because the matrix is not a square or does not have full rank. Secondly, we use SVD on the covariates themselves. That is

$$USV^\top = \phi,$$

where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{B\times B}$ are orthonormal matrices, and $S = \text{diag}(s_1, ..., s_n) \in \mathbb{R}^{n\times B}$ is a diagonal matrix with non-negative and decreasing singular values are $s_1 \geq ... \geq s_n \geq 0$. To further improve the performance in the form of lower memory consumption and overall faster processing, we can perform *reduced* or *economic* SVD. To do this, we let $r = \text{rank}(\phi)$ denote the rank of the covariate matrix[10] and make a cutoff by partitioning the columns of U and V such that $U = [U_1, U_2]$ and $V = [V_1, V_2]$ where $U_1 = [u_1, ..., u_r]$ and $V_1 = [v_1, ..., v_r]$ each have r columns. Finally, defining $S_r := \text{diag} := (s_1, ..., s_r)$ and

[10] Which is easy to find by using that s_r is the smallest singular value strictly greater than zero. I.e. $s_{r+1} = ... = s_n = 0$.

using matrix-block notation we get the reduced SVD of ϕ

$$\phi = USV^\top = [U_1, U_2] \begin{bmatrix} S_r & 0 \\ 0 & 0 \end{bmatrix} [V_1, V_2]^\top = U_1 S_r V_1^\top = \sum_{i=1}^{r} u_i v_i^\top s_i.$$

This decomposition obviously requires much less memory than the standard SVD where the number of pre-simulations needed often is much larger than the rank of the covariates, i.e. $k \gg r$. Using reduced SVD on ϕ in the normal equation gives

$$\begin{aligned}
\hat{\beta} &= \left[\phi^\top \phi\right]^{-1} \phi^\top \boldsymbol{y} \\
&= \left[\left(U_1 S_r V_1^\top\right)^\top \left(U_1 S_r V_1^\top\right)\right]^{-1} \left(U_1 S_r V_1^\top\right)^\top \boldsymbol{y} \\
&= \left[\left(V_1 S_r^\top U_1^\top\right) \left(U_1 S_r V_1^\top\right)\right]^{-1} \left(V_1 S_r^\top U_1^\top\right) \boldsymbol{y} \\
&= \left[V_1 S_r^\top I S_r V_1^\top\right]^{-1} \left(V_1 S_r^\top U_1^\top\right) \boldsymbol{y} \\
&= V_1 S_r^{-1} S_r^{-1} V_1^\top \left(V_1 S_r^\top U_1^\top\right) \boldsymbol{y} \\
&= V_1 S_r^{-1} S_r^{-1} I S_r^\top U_1^\top \boldsymbol{y} \\
&= V_1 S_r^{-1} U_1^\top \boldsymbol{y}
\end{aligned}$$

where we first used that $S_r = S_r^\top$ as S_r is diagonal and therefore also symmetric, and finally also that $V_1^\top V_1 = I$ and $U_1^\top U_1 = I$ as they are both semi-orthonormal.

As a final minor improvement we fit the regression by only using in-the-money (ITM) paths in the regression step as proposed in the original work by Longstaff and Schwartz [LS01]. It is not always the case that there are any or enough paths in the training step to ensure that the regression is stable. Therefore, we implement this feature in a slightly modified edition by first choosing the ITM paths and then resolving to also include some out-the-money (OTM) paths until the number of covariance reaches a specified threshold.

5 Automatic Differentiation

Computing risk sensitivities in great numbers for many different scenarios, and in a reasonable amount of time, remains a central challenge across trading floors and risk management practices all over the financial world. Old-school methods such as bump-and-revalue are slow and computationally heavy. Thus, deeming them incapable of meeting the growing demand for fast, detailed, and accurate risk reporting by both market practitioners, management, and financial regulators. Especially if we have to combine a Monte Carlo simulation with bump-and-revalue in different directions to get a full risk picture. This requires a full iteration for each price and sensitivity we wish to obtain. In some cases, for certain exotic products, Monte Carlo methods are the only viable choice for valuing such claims. The idea presented in [BG96] is to approximate differentials by taking a pathwise approach. This is very intuitively combined with Monte Carlo simulation, where we are used to estimate the price of a particular claim V_t by sampling a number of future states with outset in today's market $\boldsymbol{x}_T|\boldsymbol{x}_t$ and computing payoffs $g(\boldsymbol{x}_T)$ for the N paths:

$$V_t = \mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T)\,|\boldsymbol{x}_t\right] \approx \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}^i) := \hat{V}_t.$$

By denoting model parameters by $\boldsymbol{a}$ we can obtain the differentials of the average payoff among N paths as the average of the differentials over each path (assuming sufficiently smooth payoff conditions):

$$\begin{aligned}\bar{\boldsymbol{a}} := \frac{\partial}{\partial \boldsymbol{a}}\hat{V} &\approx \frac{\partial}{\partial \boldsymbol{a}}\left(\frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}^i)\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{a}}\left(g(\boldsymbol{x}^i)\right). \end{aligned} \tag{5.1}$$

The risk sensitivities thus becomes pathwise quantities which are well suited for our Monte Carlo engine as outlined in the previous section 4.3. The question now becomes, how can we efficiently compute the pathwise differentials in (5.1). It is showcased in [GG06] how to efficiently implement the calculation of (5.1) by rearranging the order of computations appropriately by linear algebra operations and making use of adjoints methods. This idea is extended in [Cap11] to a *automatic adjoint differentiation* (AAD) framework. By

the observation, that the pathwise estimator in (5.1) is in fact the Jacobian of the payoff function with respect to $\boldsymbol{a}$. AAD provides a general approach for calculating derivatives which we will expand on below.

5.1 Forward and Backward Automatic Differentiation

Automatic differentiation is usually divided into two methods depending on the calculation order; namely *forward* and *backward* referring to which way you traverse the decomposition of the calculation at hand. Forward auto-differentiation computes analytical derivatives but are linear in calculation time (similar, but often significantly faster, compared to a standard bump-and-revalue approach). This makes the forward mode less interesting from a practical point of view where the purpose is implementation in a (large-scale) production setup, where sensitivities has to be estimated in great numbers. Automatic backward differentiation on the other hand is able to produce differentials in constant time and is the real key described in [Sav19] and [Cap11] to unlocking new realms of speed in computing sensitivities.

In a simple setting, we wish to calculate the derivative of the composite function $(f \circ g)(x)$ with respect to the input x. We split the function evaluation into steps using below notation:

$$\begin{aligned} y = f(g(x)) = f(g(\omega_0)) = f(\omega_1) = \omega_2, \\ \omega_0 := x, \\ \omega_1 := g(\omega_0), \\ \omega_2 := f(\omega_1) = y. \end{aligned}$$

If we apply the chain rule to the composite function we can then decompose the calculations into

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial \omega_1}\frac{\partial \omega_1}{\partial x} = \frac{\partial f(\omega_1)}{\partial \omega_1}\frac{\partial g(\omega_0)}{\partial x}.$$

We now have a choice of where we wish to start the chain of calculations. In fact this is a complicated minimization problem of number arithmetic operations needed to compute the full Jacobian matrix of a given function, if we wish to take all ways of traversing the

chain rule into account. We consider only the two cases as follows: If we do as above by starting from the inside with $\partial\omega_1/\partial x$ and working our way out, this is called *forward* mode or accumulation. Here the method is to fix the independent variable and compute each of the following expressions recursively from that. This means, we will need the *seed* $\partial x/\partial x = 1$ and work our way out from that. This defines the recursive relation

$$\begin{aligned}\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial \omega_1}\frac{\partial \omega_1}{\partial x} \\ &= \frac{\partial y}{\partial \omega_1}\left(\frac{\partial \omega_1}{\partial \omega_0}\frac{\partial \omega_0}{\partial x}\right) \\ &= \frac{\partial y}{\partial \omega_1}\left(\frac{\partial \omega_1}{\partial \omega_0}\cdot 1\right),\end{aligned}$$

where $\omega_0 = x$.

If we conversely start from the outside by computing $\partial y/\partial\omega_1$ and traverses to the inside this is called *backward* or *adjoint* mode. The method instead fixes the dependent variable and computes derivatives with respect to each of the following expressions. Here we will have to provide the seed $\partial y/\partial y = 1$. In turn this gives us another recursive relation:

$$\begin{aligned}\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial \omega_1}\frac{\partial \omega_1}{\partial x} \\ &= \left(\frac{\partial y}{\partial \omega_2}\frac{\partial \omega_2}{\partial \omega_1}\right)\frac{\partial \omega_1}{\partial x} \\ &= \left(1\cdot\frac{\partial \omega_2}{\partial \omega_1}\right)\frac{\partial \omega_1}{\partial x}\end{aligned}$$

where $\omega_2 = y$.

Comparing our two methods against each other in the simple setting of $f : \mathbb{R} \to \mathbb{R}$, we see there's no difference in the amount of computations needed. However generalizing to $f : \mathbb{R}^n \to \mathbb{R}^m$, we would need to initialize the forward mode with n seeds (one for each independent variable $x_1, ..., x_n$) and perform a full evaluation (a *sweep*) for each. In the reverse mode we would then provide m seeds (one for each of the output dimensions partial derivative) and then perform a sweep for each of these. This shows us when each of the algorithms is to be preferred to another.

From a practical viewpoint, we note that both automatic forward and backward differentiation can be implemented through *operation overload*. For automatic forward

differentiation the partial derivatives are accumulated throughout the calculation graph (more on this in the next subsection) and can for instance be implemented by using using *dual numbers*. Modern, more advanced implementations are capable of tracking entire arrays of dual numbers simultaneously, which significantly improves the computational performance in terms of speed but comes with an additional cost of increased memory consumption. On the contrary, backward automatic differentiation instead just tracks the calculations performed by storing these in memory on a *tape*. This allows for several optimizations of different specific problems - especially in terms of reducing the memory required. For instance, it is possible to store intermediate results on the tape (also called *check pointing*) once they are calculated and then wiping the redundant operations from the tape. A specific application of using check-pointing to solve a financial problem is presented in the paper *LSM Reloaded* [HS17]. Here the LSM algorithm is efficiently differentiated by combining the automatic differentiation of the Monte Carlo simulation algorithm with an analytical differentiation of the SVD-regression through check-pointing.

5.2 Adjoint Differentiation for Interest Rate Derivatives

As we have seen above automatic differentiation breaks the computation of derivatives down to basic *adjoint* calculations. Adjoint calculations offer an efficient way to compute all derivatives of inputs simultaneously, making use of the practical observation that the sensitivities of a given calculation share common factors due to the chain rule for derivatives.

Consider e.g. the time-0 Black price of a caplet fixing at time-T with accrual period δ:

$$Cpl(0,T,T+\delta) = \delta P(0,T)\left(F(0,T,T+\delta)\Phi(d_1) - K\Phi(d_2)\right),$$

where

$$d_{1,2} = \frac{\log\left(\frac{F(0,T,T+\delta)}{K}\right) \pm \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}.$$

To compute this price it requires several partial computations or *operations*, which can be represented as a composite function. Figure 3 below gives a visual representation of the eight operations involved in computing the Black price. It is a calculation graph

which keeps track of the step-wise operations and their computational order from the input arguments (parameters and market variables) required to calculate the Black price in the end.

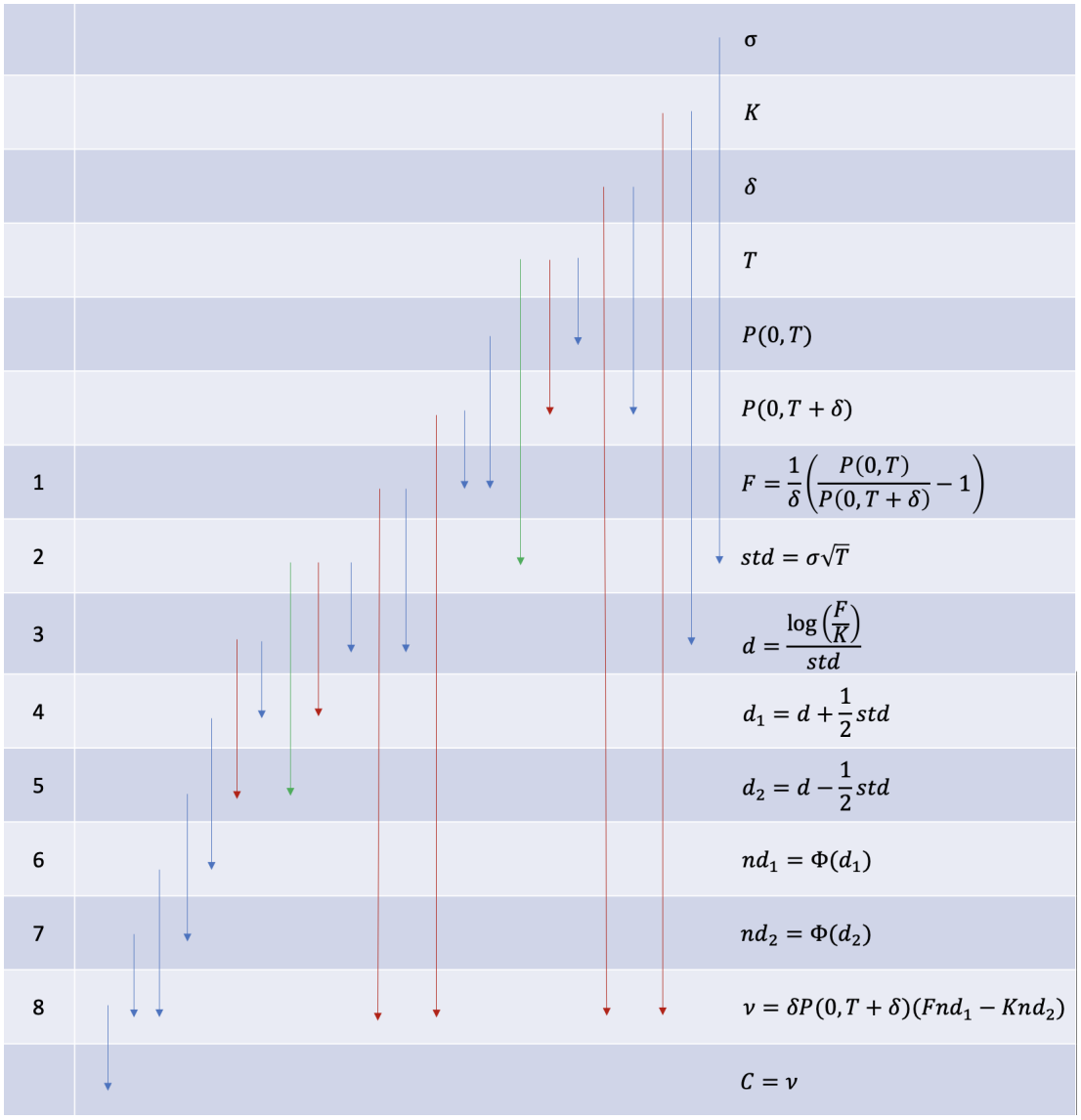


Figure 3: Visualization of using AD on black-formula of caplet. The colored arrows refers to how many dependencies each element has.

By keeping track of these elementary calculations - e.g. on the tape - we can efficiently reuse computations in the sensitivity analysis. We denote the derivative of the final results to a input parameter as the *adjoint*. As an example, we could be interested in calculating the sensitivity of the Black price with respect to the forward volatility:

$$\bar{\sigma} := \frac{\partial C}{\partial std}\frac{\partial std}{\partial \sigma} := \frac{\partial std}{\partial \sigma}\overline{std}$$

This gives us some sort of computational order or relationship between the adjoints. we can compute $\bar{\sigma}$ after we know $\overline{std}$, this makes the adjoint $\bar{\sigma}$ a function of $\overline{std}$. Notice this is in the reverse order, of what you would need to evaluate the expressions themselves. To compute $\overline{std}$ we take a look back at the graph in figure 3 and see there's three branches connected this: d, d_1, d_2. By applying the chain rule once again we have

$$\overline{std} = -\frac{d}{std}\bar{d} + \frac{1}{2}\bar{d_1} - \frac{1}{2}\ \bar{\mathrm{d}}_2$$

Now we have a way to compute $\overline{std}$ as a function of the adjoints $\bar{d}, \bar{d_1}, \bar{d_2}$. We can continue this pattern:

$$\begin{aligned}
\overline{std} &= -\frac{d}{std}\bar{d} + \frac{1}{2}\bar{d_1} - \frac{1}{2}\ \bar{\mathrm{d}}_2 \\
&= -\frac{d}{std}\left(\bar{d_1} + \ \bar{\mathrm{d}}_2\right) + \frac{1}{2}\bar{d_1} - \frac{1}{2}\ \bar{\mathrm{d}}_2 \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\bar{d_1} - \left(\frac{1}{2} + \frac{d}{std}\right)\bar{d_2} \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\varphi(d_1)\overline{nd_1} - \left(\frac{1}{2} + \frac{d}{std}\right)\varphi(d_2)\overline{nd_2} \\
&= \left(\frac{1}{2} - \frac{d}{std}\right)\varphi(d_1)\delta P(0, T+\delta)F\bar{v} + \left(\frac{1}{2} + \frac{d}{std}\right)\varphi(d_2)\delta P(0, T+\delta)K\bar{v},
\end{aligned}$$

where the adjoint of v is 1. This might seem tedious at first glance, but the efficiency really comes into play when calculating other derivatives. We can then reuse the overlapping computations and thus be able to calculate sensitivities by a few number of operations. For illustrative purposes we look at the adjoint of K:

$$\begin{aligned}
\bar{K} &:= \frac{\partial d}{\partial K}\bar{d} + \frac{\partial v}{\partial K}\bar{v} \\
&= -\frac{1}{K \cdot std}\bar{d} - \delta P(0, T+\delta)nd_2
\end{aligned}$$

We see that in the calculations of the derivatives above, we can reuse the computation of the adjoint $\overline{std}$ in both cases. By this insight, we can break the calculation of all the derivative sensitivities down to calculating the adjoints of all the inputs instead.

5.3 Payoff Smoothing and Fuzzy logic

All types of differentiation algorithms have difficulties handling discontinuities in e.g. payoff functions or caused by conditional code implementation (this is called control flow). This is due to the obvious fact that this is in nature not differentiable. Such cases must be handled with care to ensure that the differentiation implementation acts as we would expect. If this is not dealt with properly, we cannot expect to correctly price or calculate sensitivities of digitals, barrier-options, etc. But this is not a problem restricted to AD and it is common to get around the issue by approximating the discontinuities by close continuous functions instead, which is known as *smoothing* algorithms. The benefit of payoff smoothing is that we can rewrite the payoff itself such that we do not have to edit the model. Of course this comes with the requirement of certain domain knowledge about the specific product, since we have to identify and smooth all possible discontinuities ourselves. In practice, however, this is not a large issue for most products as strikes, barriers and alike are explicitly defined in the product definition.

5.3.1 Linear Smoothing of Digital Options

A digital (also referred to as binary or *all-or-nothing*) call option pays a fixed coupon C if the underlying X at maturity T_0 is above the predetermined strike level K, that is

$$g^{\text{Dig}}(X(T_0)) = C \cdot \mathbb{1}_{\{X(T_0)>K\}}.$$

The payoff from the digital call can be approximated as a limit of a call spread, i.e. the difference between holding a European call with strike $K - \varepsilon/2$ and selling a European call with strike $K + \varepsilon/2$ where we let $\varepsilon \to 0$ (here we use the central difference). We denote the payoff of this spread as g^{sprd} and notices it replicates the digital exactly in the limit as $\varepsilon \to 0$ (with a coupon of $C = 1$ without loss of generality):

$$\begin{aligned}\lim_{\varepsilon\to 0} g^{\text{sprd}}(X(T_0)) &= \lim_{\varepsilon\to 0} \frac{(X(T_0) - (K - \varepsilon/2))^+ - (X(T_0) - (K + \varepsilon/2))^+}{\varepsilon} \\ &= \mathbb{1}_{\{X(T_0)>K\}} = g^{\text{Dig}}(X(T_0))\end{aligned}$$

We may also notice that the limit above is exactly how we would express the (negative) derivative analytically of a European call with respect to the strike level K. This gives us

a way to approximate the derivative of a continuous quantity instead as shown in figure 4 below.

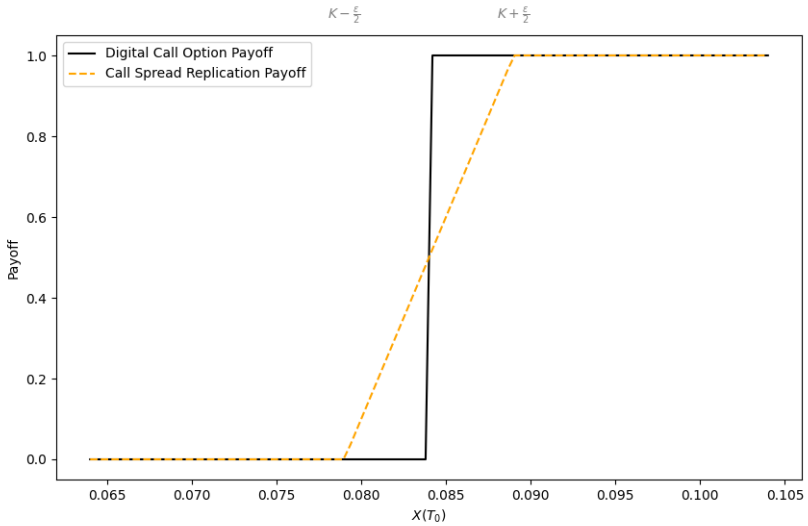


Figure 4: Approximating the digital payoff using a call spread. The idea is then to let $\varepsilon \to 0$.

As such we do not have to do any alterations to the model and all the modification can be implemented in the product specification. We simply price a claim that has a smooth payoff that is equal to a claim with a non-smooth payoff in the limit. This also comes with the interpretation that we spread the notional of the digital contract evenly across strikes between the interval $(K - \varepsilon/2,\ K + \varepsilon/2)$.

5.3.2 Linear Smoothing of Barrier Options

Smoothing a barrier option can be done similar to the idea we showed above for the digital by spreading the notional of the original barrier B across the interval $(B-\varepsilon/2, B+\varepsilon/2)$. For an *Up-and-out* barrier, we would then determine the part of the original notional that has been *knocked-out*, depending on how far the underlying ends up inside $(B - \varepsilon/2, B + \varepsilon/2)$. By only knocking out a fraction of the total notional for each path, we are essentially handing the discretization error that we make. This, the key difference to above case is the frequency with which, the barrier breach is monitored. If the barrier breach is only observed at maturity, it is equivalent to being long a call with the same strike and being short a digital with strike equal to the barrier level. If we however have

a discretely or continuously monitored barrier, the barrier then becomes a *path dependent* product.

The smoothing spread $(B - \varepsilon/2, B + \varepsilon/2)$ is implemented such that the option completely "dies" above the level $B + \varepsilon/2$ and fully "lives" below the level $B - \varepsilon/2$. In between these points, the option loses a part of the notional. The amount is determined at each monitorization point and its size is found by interpolating between 100% of the notional at the upper level, $B + \varepsilon/2$, and 0% of the notional at the lower level, $B - \varepsilon/2$. This procedure is repeated throughout the entire lifetime of the barrier for all monitorization points with the remaining notional. Hence, we are essentially allowing for the possibility if the underlying remains between the spread for a prolonged time then the notional is essentially "chipped away" by small losses over time.

We visualize the smoothing techniques below for a caplet with an *Up-and-out* barrier that is monitored in the two different ways below in figure 5. In the discretely monitored case, we show an example where the barrier is partly breach at a single observation time point. It reaches 75% of the interval and thus we knock off this part of the original notional of the option by defining an *alive* coefficient to scale the notional according to the interpolation mentioned above

$$a(x; B, \varepsilon) := \frac{B + \frac{\varepsilon}{2} - x}{\varepsilon}$$

where the argument x is the underlying assets.

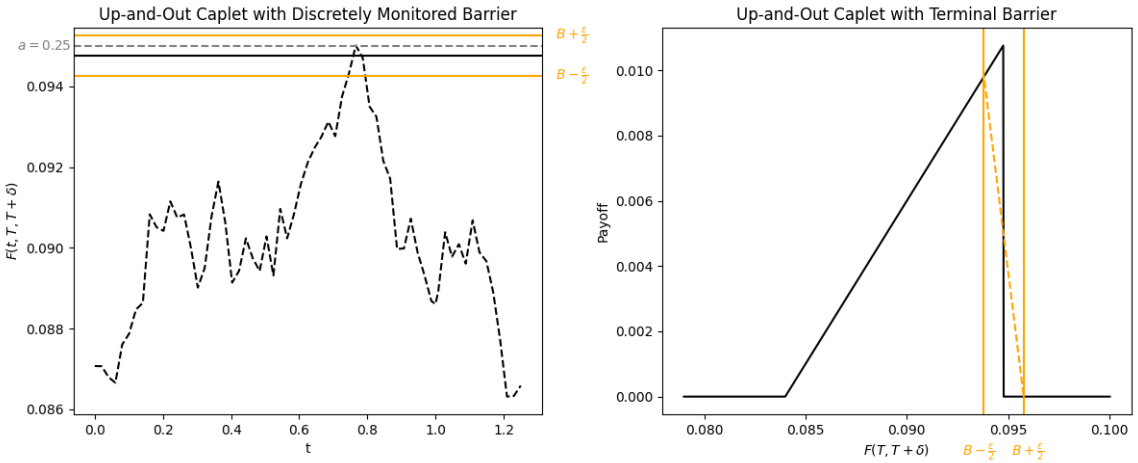


Figure 5: Smoothing techniques for Barrier options.

For the other case where the barrier is only applied at the expiry (terminal time), we can visualize the smoothing in a similar manner by modifying the payoff function directly.

5.3.3 Fuzzy Logic and Sigmoid Smoothing

Another approach to take for handling discontinuities caused by conditional code, i.e. `if else`-statements is to implement what is called *fuzzy logic*. The intuition of fuzzy logic is a concept for transforming classic "hard" logical conditions (either strictly `true` or strictly `false`) into a "soft" logic which determines to what *degree* the condition is true and to what degree it is false. The linear smoothing applied above already falls within the domain of fuzzy logic but the concept has a larger general reach and is often applied in production for financial pricing and risk engines [Sav16].

The transformation made by the fuzzy logic induces a choice of the so-called *fuzzification* which specifies how we interpret the underlying's degree of fulfilling the condition. Turning back to digital options, we can evaluate to what degree the underlying X has breached the strike level K by applying a smoothing around K. Instead of using a call spread, which is applied linearly over some interval, we can transform the payoff function over the entire domain of the underlying by applying a sigmoid function. This converts the payoff from being strictly binary into also having the possibilities of having values between 0 and 1. By introducing a scaling factor a and shifting the function towards b

we can modify the sigmoid function to resemble the digital payoff as

$$f(x) = \frac{1}{1 + e^{-a(x-b)}}.$$

This is a smooth function for which we can vary the steepness of by changing a. In figure 6 below, we have shown the effect of varying a when it comes to replicating the payoff of the digital option.

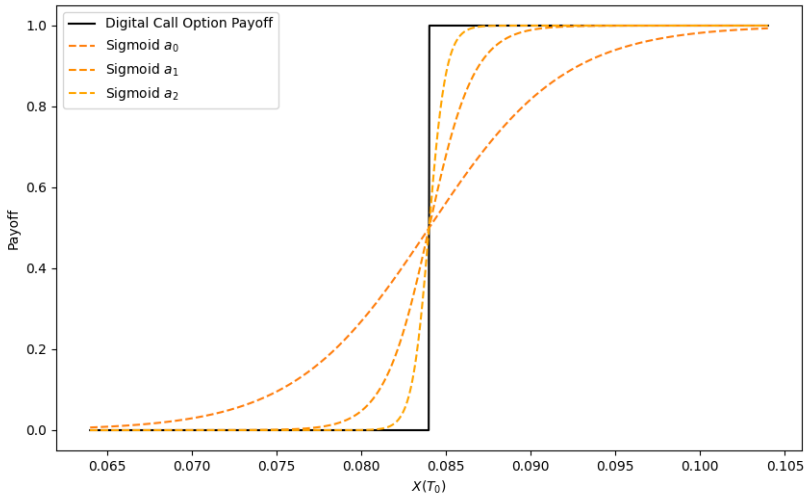


Figure 6: Transforming the binary logical expression $\mathbb{1}_{\{X(T_0)>K\}}$ using a modified sigmoid function with increasing scaling factor $a_0 < a_1 < a_2$. This applies directly in the payoff for the digital call option.

A similar smoothing by the sigmoid function can also easily be implemented for the barrier option. We close this section about smoothing and fuzzy logic by noticing that the many more variants of smoothing exists. Furthermore, the exact choice of smoothing and fuzzification can be difficult to determine a priori as the optimal choices of these depends both on the product, the model, and the discretization in the simulation scheme. Thus, it will most likely require manual tinkering by trail and error if the model and product are sufficiently complex. We will investigate this in more detail in section 7.

5.4 Relying on a Machine Learning Library

The challenging task of computing sensitivities, or derivatives, in large scale is also faced in the field of machine learning. Deep neural networks are constructed by a great number of parameters that has to be learned in somewhat reasonable time. Briefly speaking, a

neural network is defined by a set of equations arranged in a graph, where each equation has a number of parameters that needs to be estimated or tuned. We will elaborate further on the details and theory of neural networks in below section 6.3. The procedure for learning or calibrating these parameters is usually known as *back-propagation* and consists of minimizing a specified loss function, that gives some error measure between targets and predictors obtained by evaluating the network's predictions on the corresponding inputs. A neural network algorithm thus have to efficiently compute both its predictions and the derivatives of the error function. In practice this is achieved by implementing adjoint differentiation in the back-propagation step to have errors as a function of the weights. The resulting derivatives - often in the form of a gradient, a Jacobian or a Hessian - is typically used to calibrate the network's parameters through some variation of stochastic gradient decent.

In this project we have chosen to work with PyTorch which is a popular open-source machine learning library available for `Python` with an underlying `C++` implementation. Most importantly this library provides a powerful adjoint differentiation setup that automatically computes gradients of it's own multi-dimensional structures called tensors. Thus, the adjoint calculations we covered above will be calculated "behind the scenes" and the adjoint code itself, i.e. the computational graph, will be generated by `PyTorch` at run time.

`PyTorch`'s gradient calculations is performed via it's Autograd automatic differentiation engine. This engine stores a recording of data and executed operations when evaluating an expression (such as our caplet price from above) in what is called a *directed acyclic graph* (DAG) similar to the high level calculation graph we provided in figure 3, but representing solely the mathematical operations involved. Relying on the theory on adjoint differentiation above, the engine is able to compute the gradients efficiently. Finally, we also note that the `Autograd` package has its own rules to apply when encountering non-differentiable functions[11] which we need to be aware of in our implementation.

[11] `https://pytorch.org/docs/stable/notes/autograd.html`

6 Differential Machine Learning

In this section, we introduce the concepts and theory related to Differential Machine Learning (DiffML) which at its core aims to solve the problems related to the Risk Neutral Pricing Theorem. The procedure and notation relies on our previous work [HHA23] and mainly focuses on Differential Regression and Differential Neural Networks. Before getting to the details about Differential Machine Learning, we first provide some motivation and context for looking at these methods compared to more classical approaches.

One of the most common approaches to solve the problems related to the Risk Neutral Pricing Theorem is to specify a model in the form of a diffusion process as stated in the form of an SDE similar to equation (4.5). In doing so, one hopes to be able to determine the value and hedge coefficients for the products of interest with sufficient accuracy. Assuming that the model is Markovian given the information contained in the current market state, $\boldsymbol{x}_t \in \mathbb{R}^n$, the target is to find a closed-formed deterministic function, h, of the market state[12], $h(\boldsymbol{x}_t)$. Stated mathematically, the first objective is to find the true pricing function h which satisfy

$$V_t = \mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \mathcal{F}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \boldsymbol{x}_t\right] = h(\boldsymbol{x}_t). \tag{6.1}$$

Likewise, by taking the partial derivative wrt. any argument, $\boldsymbol{a}$, gives the second objective function which is to find a function for the hedge coefficients[13]

$$\frac{\partial V_t}{\partial \boldsymbol{a}} = \frac{\partial}{\partial \boldsymbol{a}}\mathbb{E}^{\mathbb{Q}}\left[g(\boldsymbol{x}_T) \mid \boldsymbol{x}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{\partial g(\boldsymbol{x}_T)}{\partial \boldsymbol{a}} \mid \boldsymbol{x}_t\right] = \frac{\partial h(\boldsymbol{x}_t)}{\partial \boldsymbol{a}}. \tag{6.2}$$

In general $\boldsymbol{a}$ is a vector and the above expression is thus a Jacobian matrix. Furthermore, the expression is not just limited to derivatives wrt. market observables ($\boldsymbol{a} = \boldsymbol{x}$) which

[12] Formally, this means that $\mathcal{F}_t$ is the *natural filtration* generated with respect to $\boldsymbol{x}_t$. If we make the regression the state variables instead of the market variables (which is sufficient for pricing but makes the estimation of hedge-coefficients more difficult) then this is obviously satisfied for the models that we consider.

[13] The interchanging of the expectation (an integral) and the derivative holds under sufficient regularization conditions - such as Leibniz integral rule - which are satisfied for the vast majority of payoff functions and stochastic processes which describe a financial market (see [CL14] and [Pit03]). For some financial products, that are either not continuous or not sufficiently "smooth", extensions may be required, e.g. by Fuzzy Logic or more formally by using functional Ito calculus [Dup19] or Malliavin calculus [Sav16].

give the hedge coefficient. It can also be either model parameters or state variables which can be used for addressing the calibration problem of the model.

In some of the classical models and for vanilla products one might be able to find closed or semi-closed form solutions to the above problems. However, as the products and models become more complex, we resort to numerical methods and find an estimator such as the Monte Carlo estimator as we have generally described in section 4.

Given that h and its derivative are deterministic functions, albeit unknown, of the current market state they can be approximated by an estimator $\hat{h}$ and its derivative. Therefore, the aim is to identify an estimator, $\hat{h}$, by machine learning that can yield accurate approximations of h and its derivative, simultaneously. This is exactly what Differential Machine Learning tries to accomplish without requiring too much training data. As we shall see, it does so by training on simulated discounted payoffs and their pathwise differentials from a (calibrated) model and then enforcing additional structure in the form of regularization from learning an estimator that cannot only replicate the value function but also its derivatives.

Performing Monte Carlo simulation with AAD as described in section 4 and 5, respectively, gives us a dataset with $N \in \mathbb{N}$ samples to train the differential machine learning model on. More specifically, we again let $\mathbf{X} \in \mathbb{R}^{N \times n}$ denote our input features in the form of current market observables, $\boldsymbol{y} \in \mathbb{R}^N$ the corresponding pathwise discounted payoffs obtained from the simulation, and $\mathbf{Z} \in \mathbb{R}^{N \times n}$ the pathwise differentials which we obtain through AAD. This approach results in a distinctive data structure for the i'th observation / row, wherein

$$\mathbf{X}^{(i)} \in \mathbb{R}^n, \qquad \boldsymbol{y}^{(i)} \in \mathbb{R}, \qquad \mathbf{Z}^{(i)} = \frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{X}^{(i)}} \in \mathbb{R}^n. \tag{6.3}$$

We emphasize, that the j'th column of $\mathbf{Z}$ corresponds to the derivative of $\boldsymbol{y}$ with respect to the j'th column in $\mathbf{X}$, i.e. $\mathbf{Z}_j = \partial \boldsymbol{y} / \partial \mathbf{X}_j$.

6.1 Generating Training Data

The accuracy of a model depends of course on the quality of the inputs it is given. This means, it is crucial to establish an accurate way of producing sample data and in particular path-wise differentials. We have a solid setup for this utilizing our Monte Carlo engine instrumented with AAD as we covered in above section 5. If we let $\boldsymbol{P}$ denote the vector of model parameters, AAD can be used to produce parameter sensitivities for respectively market observables (as input features), and parameter sensitivities for discounted payoff labels by viewing these as functions of the paremeter vector:

$$\frac{\partial \boldsymbol{X}^{(i)}}{\partial \boldsymbol{P}}, \quad \frac{\partial \boldsymbol{y}^{(i)}}{\partial \boldsymbol{P}}$$

To relate these quantities to market sensitivities we rely on a trick pointed out in [Lin13] to apply the chain-rule on the payoff function:

$$\frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{P}} = \frac{\partial \boldsymbol{y}^{(i)}(\mathbf{P})}{\partial \boldsymbol{X}^{(i)}(\mathbf{P})}^{\top} \frac{\partial \boldsymbol{X}^{(i)}(\mathbf{P})}{\partial \mathbf{P}}.$$

By rearranging, we can obtain the market sensitivities

$$\boldsymbol{Z}^{(i)} \equiv \frac{\partial \boldsymbol{y}^{(i)}(\mathbf{P})}{\partial \boldsymbol{X}^{(i)}(\mathbf{P})}^{\top} = \left(\frac{\partial \boldsymbol{X}^{(i)}(\mathbf{P})}{\partial \mathbf{P}}^{\top} \right)^{-1} \frac{\partial \boldsymbol{y}^{(i)}}{\partial \mathbf{P}}. \tag{6.4}$$

This method forms the foundation for calculating sensitivities, allowing conversion from model parameter sensitivities to market sensitivities. A graphical presentation of the mechanism is shown in figure 7. This method allow us to compose training samples fast, by relying on the speed of AAD for derivative sensitivities with respect to model parameters and then by simple matrix operations we can obtain market sensitivities.

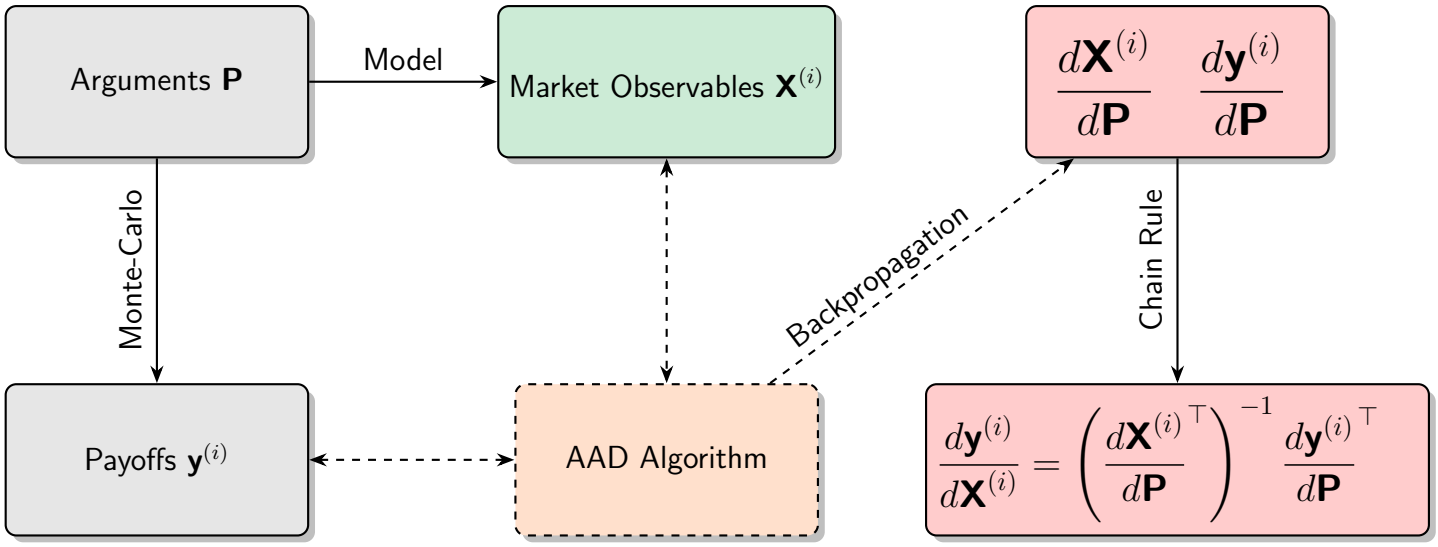


Figure 7: Overview of training data generation using AAD. Dashed lines indicate calculations carried out by the AAD algorithm.

6.2 Differential Regression

Inspired by Ridge- and Lasso-regression which adds regularization by penalizing large coefficients through a modification in the objective function, we can in a similar manner enforce differential regularization to OLS regression. To achieve this, we start similar to subsection 4.6 by considering a set of $B_r \in \mathbb{N}$ differentiable basis functions

$$\psi := \psi(\mathbf{X}) \equiv [\psi_1(\mathbf{X}), \ldots, \psi_{B_r}(\mathbf{X})] \in \mathbb{R}^{N \times B_r}.$$

An obvious choice is for the basis to use polynomials where we collect the weight coefficients, in a vector $\beta \in \mathbb{R}^{B_r}$. Without any further modification, we are within the theory of classical OLS-regression where the regression is fitted only with simulated discounted pathwise payoffs as the dependent variables and the transformed market observables as the covariates. We recall that the objective function for this classical regression is then to minimize the mean squarred error (MSE), i.e.

$$\min_{\beta} \mathrm{MSE}(\beta \mid \boldsymbol{y}, \psi) = \min_{\beta} \frac{1}{N} \|\boldsymbol{y} - \psi\beta\|^2, \quad \beta \in \mathbb{R}^{B_r}$$

The optimal choice of the estimated weights are the normal equation for regression

$$\hat{\beta} = \left[\psi^\top \psi\right]^{-1} \left[\psi^\top \boldsymbol{y}\right].$$

Same methodology is applied within differential regression, but instead of only predicting payoffs / prices with $\boldsymbol{y} \approx \psi(\mathbf{X})\hat{\beta}$, we also make the addition of estimating the sensitivities $\mathbf{Z}_j \approx D\psi_j\hat{\beta}$, where $D\psi_j := \partial\psi(\mathbf{X})_j/\partial\mathbf{X}_j$ denotes the matrix of derivatives of the basis functions ψ wrt. the j'th covariate $\mathbf{X}_j$.

In order for the objective function to take both the payoffs and sensitivities into account we extend the MSE with and additional terms that penalizes deviations in the fitted derivatives. In opposition to Ridge- and Lasso-regression which normally exhibit the bias-variance tradeoff, this approach still an unbiased estimator as it simply enforces more structure in the fit. The weight coefficients are now found by solving the adjusted MSE

$$\min_\beta \left\{ \frac{1}{N}\|\boldsymbol{y} - \psi\beta\|^2 + \sum_j \frac{1}{N}\alpha_j \|\mathbf{Z}_j - D\psi_j\beta\|^2 \right\} = \min_\beta \left\{ \text{MSE} + \sum_j \frac{1}{N}\alpha_j \|\mathbf{Z}_j - D\psi_j\beta\|^2 \right\},$$

where α_j determines the regularization enforced on each sensitivity relative to minimizing the MSE. One choice is $\alpha_j = \frac{\|\boldsymbol{y}\|}{\|\mathbf{Z}_j\|}$ ensures that the terms in the objective function have the same magnitude.

To find the optimal weights for the regression coefficient, β , we zero the gradient of the objective function wrt. β and solve for the estimate $\hat{\beta}$ which gives the adjusted normal equation:

$$\begin{aligned}
& -\frac{2}{N}\left(\psi^\top\psi\beta - \psi^\top\boldsymbol{y} + \sum_j \alpha_j \left(D\psi_j^\top D\psi_j\beta - D\psi_j \boldsymbol{Z}_j\right)\right) = 0 \\
\Leftrightarrow \quad & \left(\psi^\top\psi + \sum_j \alpha_j D\psi_j^\top D\psi_j\right)\beta = \psi^\top\boldsymbol{y} + \sum_j \alpha_j D\psi_j^\top \mathbf{Z}_j \\
\Leftrightarrow \quad & \hat{\beta} = \left(\psi^\top\psi + \sum_j \alpha_j D\psi_j^\top D\psi_j\right)^{-1}\left(\psi^\top\boldsymbol{y} + \sum_j \alpha_j D\psi_j^\top \mathbf{Z}_j\right).
\end{aligned}$$

Relating this adjusted normal equation to equation (6.1) and (6.2), we now have a method for finding an unbiased estimator for both the functions of both the price and sensitivities. Hence, this approach also have computational advantages compared to other kinds of regression as we solve multiple problems at once while not relying on cross-validation.

Furthermore, we also see that when using differential regression, we are faced with the same issue as discussed in subsection 4.6 in terms of numerical stability and existence of the inverse matrices. To overcome these issues we can again utilize the SVD of $\psi(\mathbf{X})$ in finding the pseudo-inverse and also for improving both the computational performance in the form of much lower memory requirements and improving the stability by avoid (near) multi-colinear covariates.

Finally, as we specifically implement differential polynomial regression, we can choose to include an additional covariate for the intercept and also include interactions of lower order monomials when there are multiple *raw* covariates ($n > 1$). Including these interactions comes with a trade-off as it increases the number of parameters exponentially with the degree of the polynomial which requires exponential more training data to avoid numerical instabilities even when using SVD.

6.3 Differential Neural Network

A feedforward neural network model is usually conceptualized as a function mapping input features $\mathbf{X}$ to the corresponding predicted labels $\boldsymbol{y}$. This relationship is expressed as

$$\boldsymbol{y} = G_L(\mathbf{X}) + \varepsilon,$$

where $G_L(x)$ denotes a deep neural network comprising L layers, and ε represents independent and identically distributed (iid) noise. The architecture of the deep neural is a composition of sequential transformations. Each layer, ℓ, in the network is defined by a specific weight matrix $w_\ell \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$, a bias vector $b_\ell \in \mathbb{R}^{n_\ell}$, and an activation function $g_\ell : \mathbb{R} \mapsto \mathbb{R}$. Here n_ℓ signifies the dimensionality of the ℓ'th layer. The transition from

the ℓ'th to the $\ell+1$'th layer is characterized by the transformation

$$\nu_{\ell+1} = g_\ell(\nu_l)w_l + b_l,$$

where $\nu_\ell \in \mathbb{R}^{n_\ell}$ is the row vector of pre-activation values of layer ℓ, named units (or neurons). Thus the deep neural network can be formulated as a composition of these transformations, represented as

$$G_L(x) = \nu_L \circ \nu_{L-1} \circ \cdots \circ \nu_0. \tag{6.5}$$

Within the framework of this discussion, it is relevant to reference the *universal approximation theorem* and its relevance in learning the estimator, $\hat{h}$, from the covariates as elaborated in previous sections. The theorem, while presented in various forms, consistently arrives at a similar conclusion: For any given continuous function, regardless of its complexity, there exists a corresponding estimator in the form of a feedforward neural network capable of approximating this function with an arbitrary level of precision. Remarkably, this approximation can be achieved even with a neural network consisting of only a single hidden layer, given a sufficient choice of units. This is true irrespective of the chosen activation function or the nature of the input data. However, while this theorem provides an assurance of the theoretical existence of a neural network capable of fitting any continuous function, it does not offer guidance of the specific construction of such a network. Nor does it guarantee the feasibility of training it within a practical timeframe. Consequently, in practical applications, a trade-off between the efficiency (speed) of training and the accuracy (precision) of the approximation often arises.

Proceeding from equation (6.5), the network can be represented recursively by the feedforward equations, presenting the architecture in a detailed manner, closely aligned with computational implementation.

$$\begin{aligned} \nu_0 &= x, \\ \nu_\ell &= g_\ell(\nu_{\ell-1})w_\ell + b_\ell, \quad \ell = 1, ..., L \\ y &= \nu_L. \end{aligned} \tag{6.6}$$

Typically, feedforward networks utilize backpropagation techniques to refine the selection of weights and biases within a given cost function, a process that involves calculating derivatives. The derivatives of the predicted output $y = \nu_L$ wrt. the input variables $x = \nu_0$ represent the price sensitivities to the state variables. By systematically differentiating the feedforward equations in reverse order, it is possbile to define a secondary feedforward network using adjoints

$$\begin{aligned} \bar{\nu}_L &= \bar{y} = 1, \\ \bar{\nu}_{\ell-1} &= (\bar{\nu}_\ell w_\ell^\intercal) \otimes g'_\ell(\bar{\nu}_{\ell-1}), \quad \ell = L, ..., 1 \\ \bar{x} &= \bar{\nu}_0 \end{aligned} \tag{6.7}$$

where $\otimes$ is elementwise multiplication. This network accepts input of $\bar{y}$ and the units $\nu_0, \ldots, \nu_L$ to produce $\bar{x}$. Therefor it shares the same weight matrices and bias vectors as the feedforward equations in (6.6). Notably, extending this explicit backpropagation approach to various network architectures is essentially AAD.

6.3.1 Twin Network

Drawing inspiration from Savine & Huge's work in [SH20], we closely follow their implementation to combine the feedforward network outlined in (6.6) with the explicit backpropagation network from (6.7). This composition defines the *twin network*. The rationale behind this integration is to merge the processes of price approximation and risk sensitivity calculations within a singular network framework, while also consolidating the computation associated with the shared weights.

The twin network effectively doubles the depth compared to the original forward network. Despite this increase in computational complexity, the network is capable of simultaneously providing predictions of prices and estimations of risk sensitivities. It is important to note from the second network that the adjoint units are activated using the derivatives g'_ℓ of the original activation function g_ℓ. This necessitates a careful selection of the activation function, ensuring that it is at least twice differentiable to accommodate these requirements.

The twin network is exemplified for a network with $L = 3$ layers and dimensions $n = n_0 = 1$, $n_1 = 4$, $n_2 = 3$, $n_3 = 1$, as depicted in the figure 8. This illustration effectively demonstrates the manner in which the weight calculations from the initial network are utilized within the subsequent network.

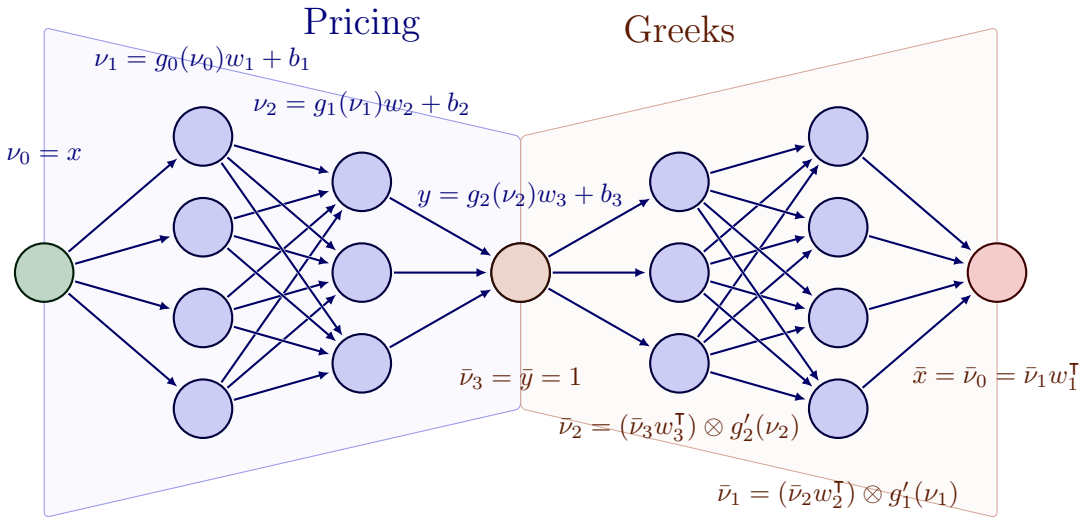


Figure 8: Twin network

The approximation of (6.1) is $\hat{h}(\mathbf{x}; w_\ell, b_\ell; \ell = 1, ..., L)$. In this model, weights and biases are determined based on a training dataset that includes differential labels, state and price data, represented as in (6.3). The optimization process involves defining a loss function, denoted by $\mathcal{L}$, which the weights and biases aim to minimize

$$\{w_\ell, b_\ell\}_{\ell=1,...,L} = \arg\min \mathcal{L}\left(\{w_\ell, b_\ell\}_{\ell=1,...,L}\right).$$

As discussed by Savine & Huge in [SH20], optimal numerical results are achieved using a combined loss function that accounts for both value and derivative errors:

$$\mathcal{L}\left(\{w_\ell, b_\ell\}_{\ell=1,...,L}\right) = a\mathcal{L}_{\text{val}} + b\mathcal{L}_{\text{diff}}. \tag{6.8}$$

This method, which minimizes two components, is similar to classic regularization techniques like ridge regression.

The first component, focusing on payoff learning, involves minimizing $\mathcal{L}_{\text{val}} = \text{MSE}$, which measures the discrepancy between the predicted values ν_L and the actual payoff labels

$\mathbf{y}$. Backpropagation through the second network isn't necessary for calculating this loss term, as it doesn't influence the MSE.
The second component, concentrating on learning from pathwise differentials, uses $\mathcal{L}_{\text{diff}}$ to measure the variance between the differential labels $\mathbf{Z}$ and the predicted labels $\bar{\nu}_0$. This is implemented as a weighted average of derivative errors, $\overline{MSE}$, i.e.

$$\mathcal{L}_{\text{diff}} = \frac{\sum_j \lambda_j^2 \overline{MSE}_j}{n},$$

where n is the number of differentials. This step necessitates full evaluation of the twin network to compute $\bar{\nu}_0$.

Implementing pathwise differentials in a neural network effectively requires careful consideration and justification of various parameter choices, which are addressed from a numerical standpoint in the subsequent section.

7 Price and Sensitivity Estimations

In this section we put the different parts of the machinery together, which we have covered up until this point. We will show how seamless it becomes to generate data via our Monte Carlo engine equipped with an AAD algorithm to efficiently produce path-wise training data to be used for our Differential Machine Learning methods to learn price and sensitivity curves. This can be done for a wide range of products and instruments (which we do below), as long as we can model the cashflow. We apply the estimation techniques in delta hedging experiments to showcase how well the methods performs in a dynamical trading environment.

7.1 Differential Regression

For each simulated path of state variables (for now, just the short rate), deltas are calculated as described in figure 7 to acquire pathwise differentials for differential machine learning. Figure 9 demonstrates the approximation of a caplet's pricing function, h, and delta via differential regression, with an expiry of $T = 1Y$.

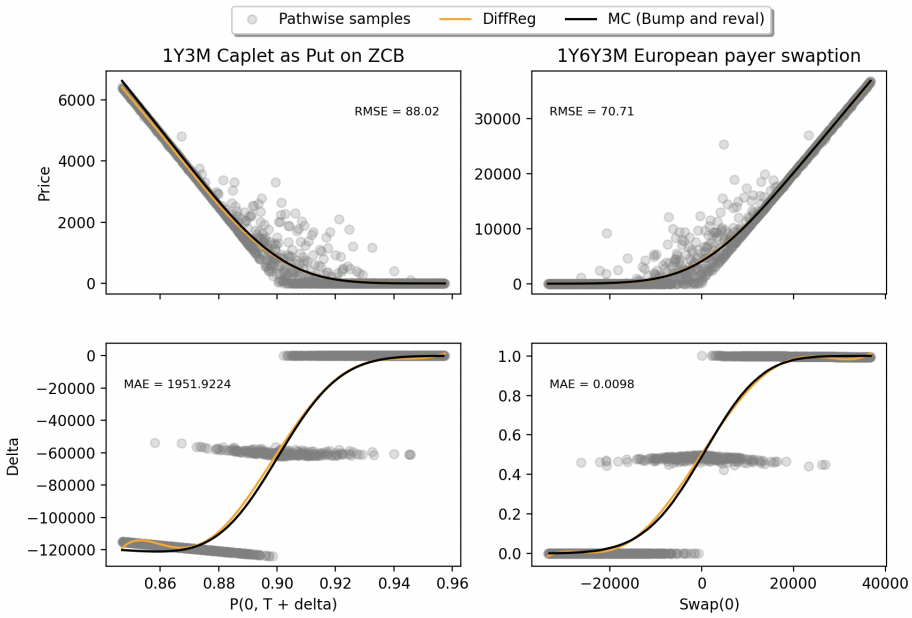


Figure 9: Price (upper panel) and delta (lower panel) estimation for a 1Y3M caplet on zero-coupon bond (left panel) and an 1Y6Y3M European swaption (right panel), by differential regression using polynomial degree = 7 and regularization $\alpha = 1$. The model is trained on 1,024 AV training samples, with strike/swap rate $K = 8.92\%$ and notional $N = 1,000,000$.

The training sample's cloud formation results from antithetic-variates, and choices like

polynomial order and sample size are arbitrary, requiring further investigation below. We set the differential regularization at $\alpha = 1$, based on numerical tests from [HHA23]. The caplet delta relates to zero-coupon-bonds, as indicated by the y-axis values. The true price and delta function are based on the bump-and-revalue method, bumping the short rate by 1 bp. Despite training on only $N_{train} = 1,024$ samples, the model's estimation of the pricing- and delta function appears promising.

Evaluating performance based on a snapshot of price and delta estimation can be a bit misleading. Instead we consider a more real-life delta-hedging experiment. Performing hedge experiments yields not just the hedge-error, but also implicitly evaluates the effectiveness of underlying mechanisms as the derivative approaches expiry. The strategy is as follows;

Example 7.1. At the initial time $t = t_0 = 0$ we sell a derivative for the price $V(0)$, and establish a position of $\hat{\Delta}(0)$ in the underlying, $U(0)$, for instance the swap or zero-coupon bond. This action is guided by our estimator $\hat{\Delta}(0)$ prediction of the "true" $\Delta(0)$. The difference between the price of the derivative and the delta-position is invested in a risk-free asset, to accrue interest. Initially, our balance is $b(0) = V(0) - \hat{\Delta}(0) \cdot U(0)$. Then, at each discrete time step $\Delta t = M/T$, continuing until the option's exercise or expiration at times $t_1, ..., t_M = T$ (where $M = 50$), the procedure is following algorithm 1.

Algorithm 1 Delta Hedge Vasicek

$t = 1$;
while $t < M$ **do**
  $r(t) \leftarrow r(t-1) + dr(t)$;
  $U(t),\ B(t) \leftarrow r(t)$;
  $V(t) \leftarrow \hat{\Delta}(t-1) \cdot U(t) + b(t) \cdot B(t)$;
  $\hat{\Delta}(t) \leftarrow U(t)$;
  $b(t) \leftarrow V(t),\ b(t),\ B(t)$;
  if exercised before expiration **then**
    break
  else
    $t + 1$
  end if
end while

When the counterparty has exercised the option or it expires, at time τ, we match the

payoff, $\Phi(\tau)$, against the value of the hedge-portfolio to compute profit-and-loss

$$PnL(\tau) = P(0,\tau)\left[V(\tau) - \Phi(\tau)\right].$$

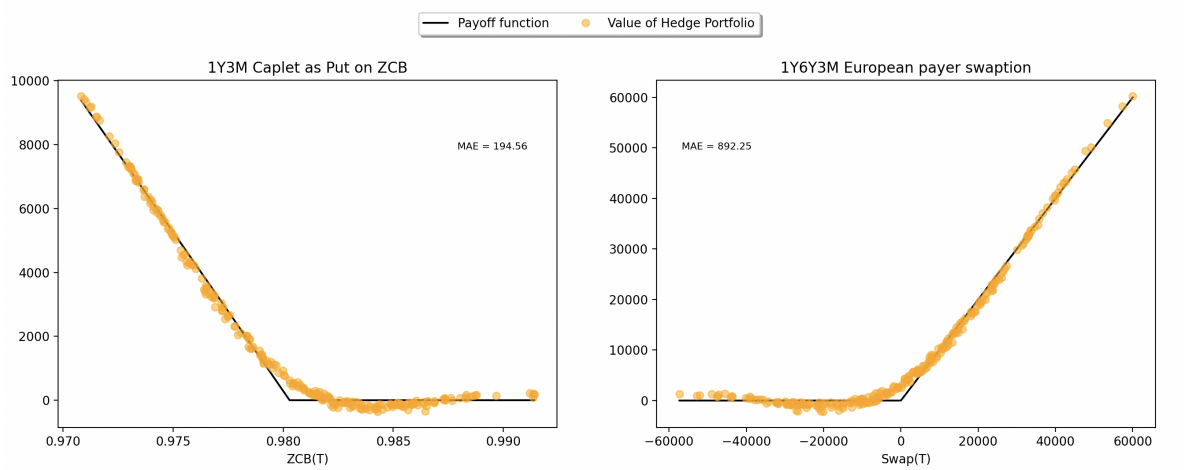


Figure 10: Hedge error of delta hedging a 1Y3M caplet as put on a zero-coupon bond with strike rate $K = 8.71\%$ and a 1Y6Y3M European payer swaption with swap rate $K = 7.88\%$ using differential regression, both with notional $N = 1,000,000$. The model is in both cases trained on 1,024 AV samples with using 7 degree polynomial and regularization $\alpha = 1$ and the portfolio is re-balanced 250 times through the options' lifespan of 1 year.

A visualization of the hedge experiment is provided in figure 10 for a caplet and a European swaption, when hedging once every day over the year. We expect the model to be versatile no matter the state of the market, hence we have dispersed an uniformly grid of short rates. Generally the hedge replicates the payoff well, but shows a systematic difficulty fitting the at-the-money payoff. The complexity of fitting isn't evident from figure 9, but the challenges intensifies as the option approaches expiry ($T \to 0$), as depicted in figure 11. As the option gets closer to expiry, the curve's slope sharpens around at-the-money positions while remaining constant elsewhere. This poses a challenge: the polynomial overly adapts to linear segments, sacrificing flexibility needed elsewhere. Alternative strategies, like sampling from varied distributions or modifying grid size, are options. Yet, to hedge multiple spots without preset distribution knowledge, we continue training on uniformly distributed inputs:

$$\boldsymbol{x} \sim U\left(a,b\right), \quad a < b.$$

Here, a and b span the full spot grid range. To prevent asymptotic issues in our experiments, we ensure a sufficiently wide grid, avoiding extrapolation problems — critical since polynomial regression is unreliable for extrapolation. Figure 11 demonstrates the trade-off: prioritizing quality can reduce accuracy in the core domain.

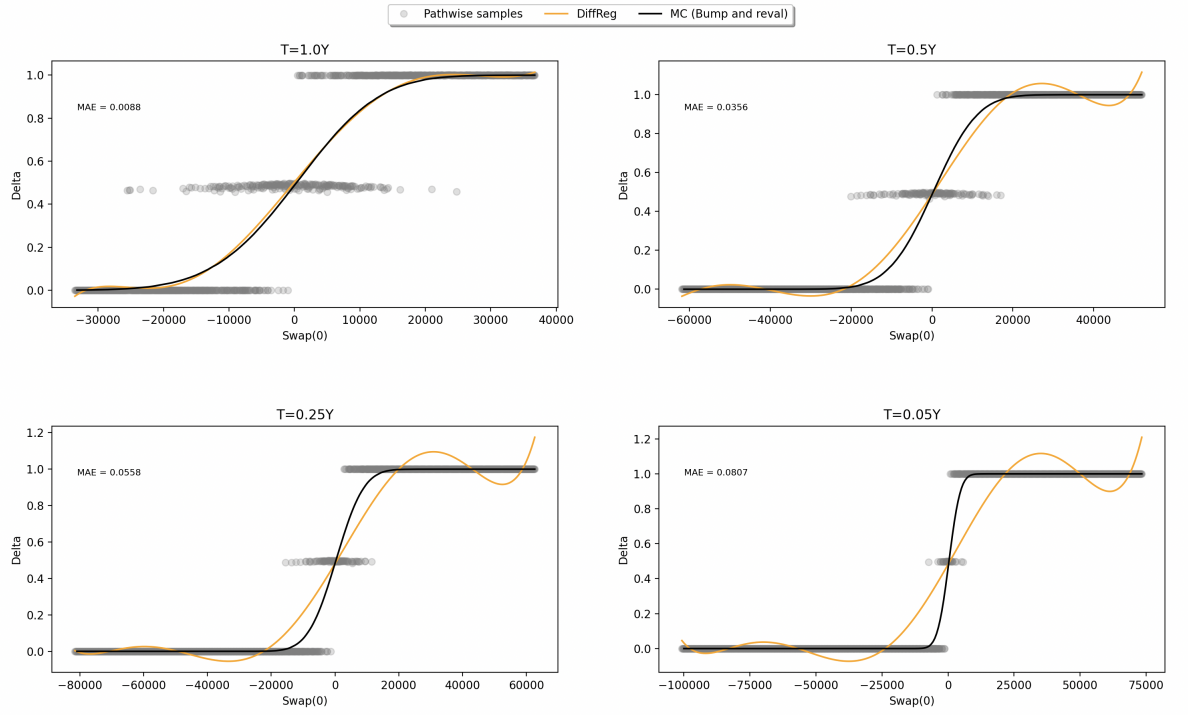


Figure 11: Delta estimation for a 1Y6Y3M European payer swaption until expiry, by 7'th order differential regression. The model is trained on 1,024 AV training samples, with strike/swap rate $K = 8.982\%$ and notional $N = 1,000,000$.

The choice of polynomial order and training sample size remains unoptimized for performance enhancement or hedge error reduction. Experimenting with different hedge frequencies and training sample sizes, and analyzing their effects on hedge error standard deviations, reveals potential for fine-tuning. This is illustrated in Figure 12, where increasing polynomial orders correlate with decreased hedge error. Remarkably, the hedge error stabilizes with a minimal number of samples, as seen in the right panel. Balancing computational complexity and precision, a 9th order polynomial fit with 1,024 training samples seems to be on the safe side.

As covered in section 4, the convergence order of the standard error of the Monte Carlo estimator are $\mathcal{O}(N^{-0.5})$ which implies a convergence order of -0.5. This convergence order seems present for especially the higher-order polynomials, with decreasing effect hedging more than once every other day on a one-year contract.

Practically, the figure shows that daily hedging of a 1 million notional European payer swaption using a 9th order polynomial differential regression with 1,024 training samples results in a PnL deviation around $e^{6.0} \approx 403$. Relative to the initial swaption price of $V(0) = 11,142$, this equates to a hedging error of about 3.7%, indicating pretty good hedge performance.

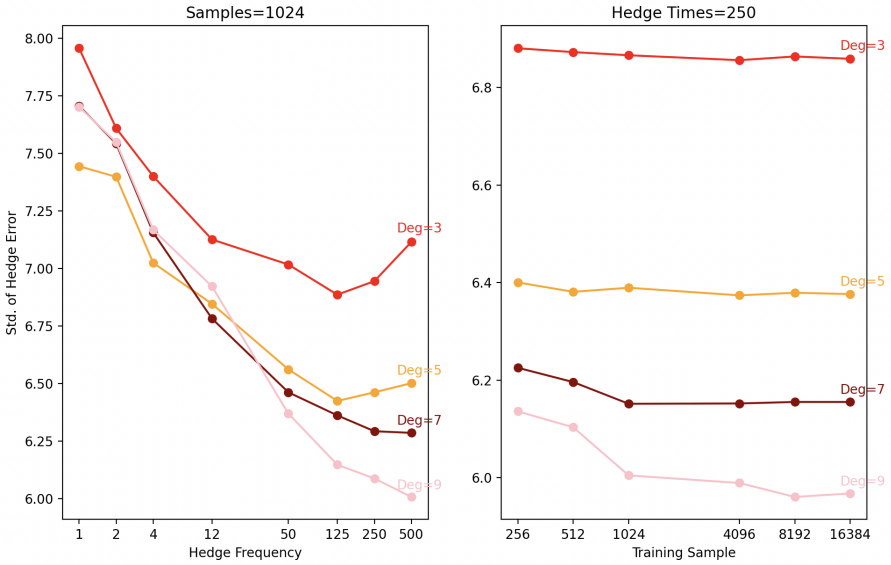


Figure 12: Convergence plots of hedge error for a 1Y6Y3M European payer swaption with strike rate $K = 8.71\%$ for different polynomials including interactions on log-log scale. Simulated from single spot value.

In conclusion our preferred approach is using 1,024 training samples for a 9'th order polynomial. However, this is only in the context of vanilla derivatives. The impact on performance when dealing with more complex derivatives in various aspects is explored below.

To enhance complexity and test the models further, one strategy is to introduce more non-differentiable points. An example is estimating the price and delta of an European barrier swaption as depicted in figure 13. We have applied smoothing techniques and from section 5.3.2 and 5.3.3 for AAD compatibility, as shown in the figure. The choice of smoothing method appears less vital, with sigmoid functions offering only minor enhancements. While pricing functions are generally accurately estimated, fitting the delta near the barrier remains challenging, owing to a scarcity of path-dependent training samples

that surpass the barrier.

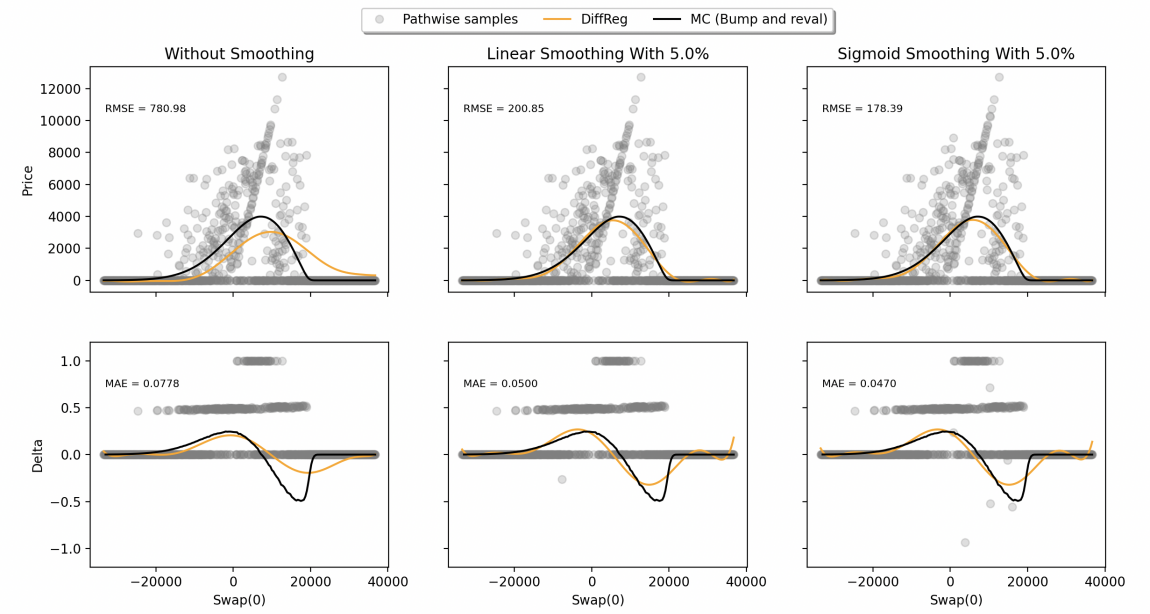


Figure 13: Delta and price estimation of a 1Y3M European up-and-out barrier swaption using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. The barrier is $H = 20,000$ with smoothing interval $\pm 5\%$ of H. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate 8.71%. The MC price and delta are estimated without any smoothing.

The difficulty in accurately estimating price and delta for barrier swaptions becomes more acute as the option nears expiry, as illustrated in figure 14. Approaching expiry ($T \to 0$), the underlying swap's value and the price slope both increase, leading the delta function towards $\pm\infty$. The uniform distribution of training labels across the interval treats deep in-the-money and out-of-the-money observations equally with those near the barrier, resulting in suboptimal fitting. This underlines a limitation in differential regression, indicating a potential need for alternative sampling methods.

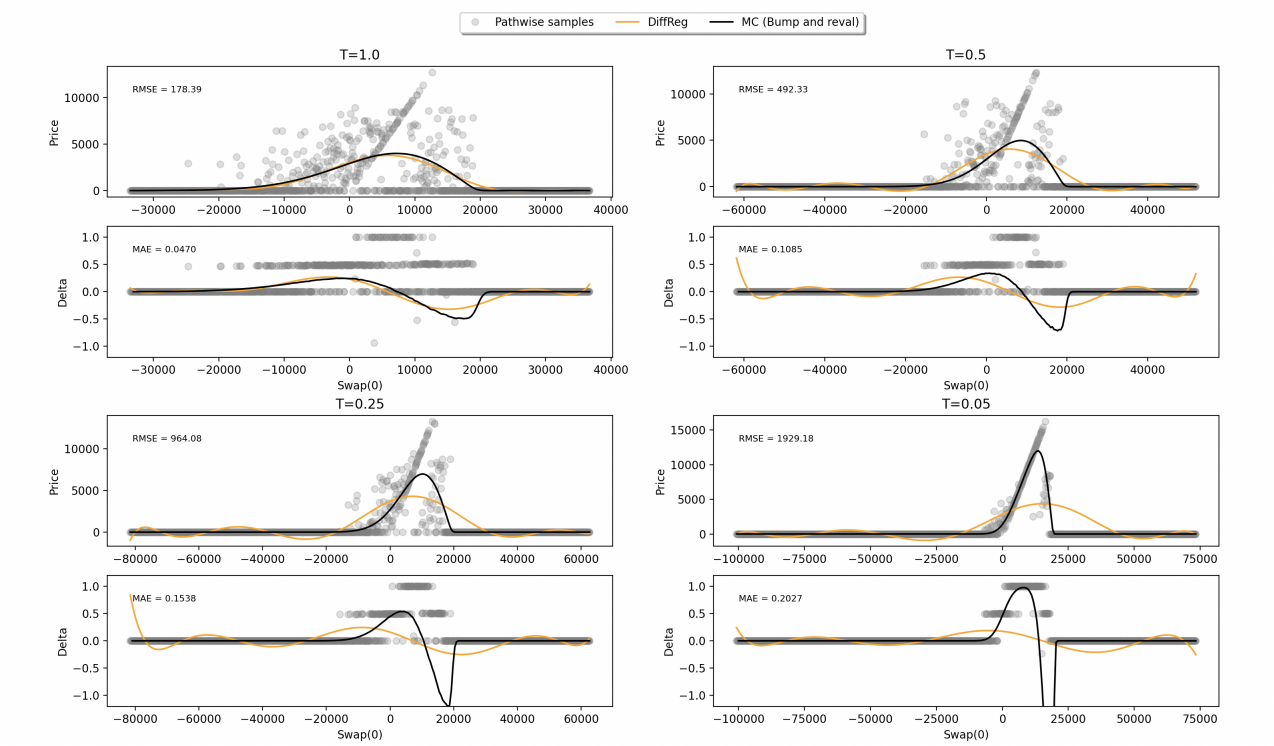


Figure 14: Delta and price estimation for varying maturities of European up-and-out barrier swaptions using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. The barrier level is $B = 20,000$ with smoothing interval $\pm 5\%$ of H using sigmoid function. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate 8.71%. The MC price and delta are estimated without any smoothing.

Monte Carlo methods are also widely used for basket options, where the underlying is a collection of market observables. A simple case is a European basket swaption as specified in subsection 2.3.3, where the underlying is a basket of swaps. In cases when the model is complex (unlike the Vasicek model) or the number of underlying assets is large, a standard Monte Carlo implementation is relatively slow and generally undesirable for risk applications like evaluating the pricing function in various market scenarios to estimate Value-at-Risk. However, when integrating differential Machine Learning, particularly differential regression, enhances efficiency. It requires fewer Monte-Carlo simulation and an evaluation of the modified normal equation to fit the regression. The regression covariates can either be the state variables, swap rates, simple forward rates, the underlying swap contracts, or the basket itself. To accurately fit both the pricing function and hedge coefficients, underlying swaps are the preferred choice. In the Vasicek model, due to the highly (almost perfectly) correlation between swaps, both classical and differential regression accurately estimate the pricing function as illustrated in 15 below. For precise hedge

coefficient estimates of each underlying swap, differential regularization is necessary. In cases where correlation is not perfect, classical regression breaks with just a few underlying assets while differential regression performs well up until around 7-10 underlying assets as demonstrated in [SH20]. However, in our Vasicek model experiment, differential regression doesn't show a systematic advantage over classical regression with an adequate number of training samples, around 1,024, as shown in the following example.

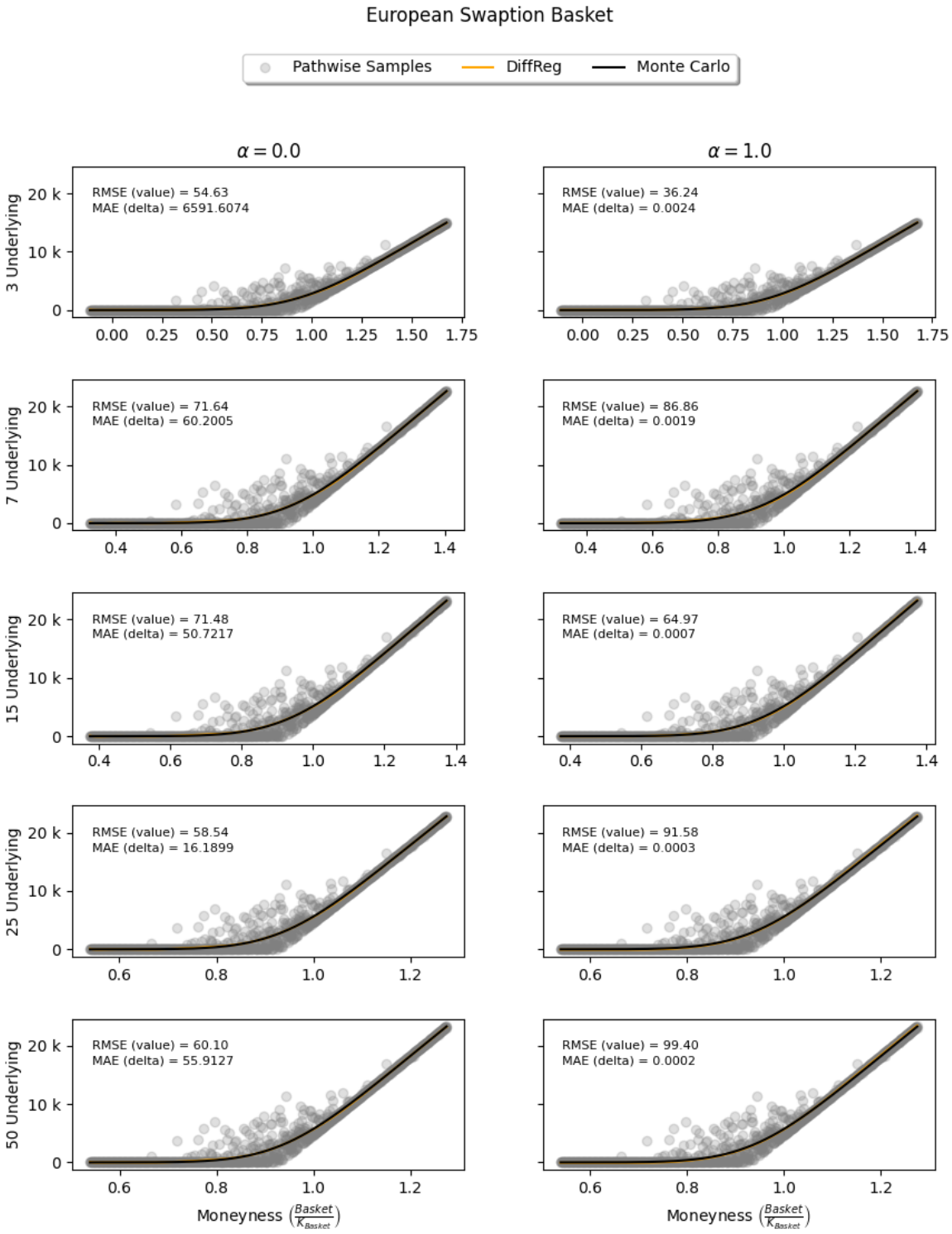


Figure 15: Value of European Basket Swaption with $1Y$ *to expiry. The last fixing date for each the underlying swaps are randomly chosen as the expiry plus* $T_n \in \{3M, 6M, 1Y, 2Y, 5Y\}$*, the fixing rates are sampled from* $\mathcal{N}(0.06\ ;\ 0.02)$*, and the weights are first randomly sampled from* $w \in \mathcal{U}(0\ ;\ 1)$ *and then scaled such that* $\sum_i w_i = 1$*. All swaps have 4 fixings per year,* $\delta = 3M$*, and a notional of* $1,000,000$*.*

Another natural extension from the single European swaption is entire portfolios of European swaptions. Our Monte Carlo framework is adept at combining several products into a portfolio, enabling simultaneous sampling. Applying this approach to the European swaption portfolio, treating it as one product inspired by [Sav19]. A notable subset of European portfolios consists of various option strategies designed to achieve specific risk exposures while mitigating others. A trader might aim for delta-neutrality, but still have exposure to changes in volatility. Some strategies, like a *calendar spread*, involves buying and selling options with different maturities, but others are composed entirely of options that expire on the same day. Generally, and particularly for these strategies, it's advantageous to express the strategy's value as a function of the underlying swap rate.

Below we demonstrate that differential regression is capable of fitting both the value and delta function of the strategies as a function of the underlying swap rate. As an example, figure 16 shows that differential regression accurately fits the price and delta of a *butterfly spread* very well. A butterfly spread is the composition of 4 European payer swaptions:

- Long 1 European payer swaption at a low fix rate (6.0%),
- Short 2 European payer swaptions at a mid fix rate (7.5%),
- Long 1 European payer swaption at a high fix rate (9.0%).

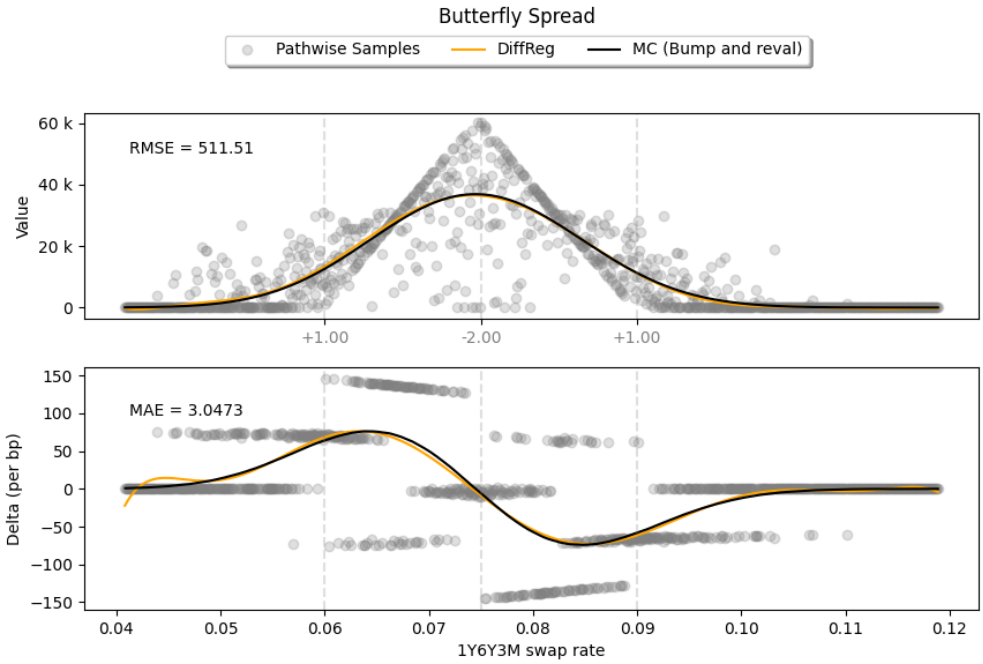


Figure 16: Butterfly spread value and sensitivity functions over different swap rates. The underlying swaptions all expiry at $1Y$*, and the underlying swaps fixes 4 times per year (*$\delta = 0.25$*) from* $1Y, ..., 6Y$*, each with a notional of* $1,000,000$*. The fix rates are* 6.0%*,* 7.5%*, and* 9.0% *The differential regressor is trained on* $1,024$ *samples with AV reduction. The differential regularization is set to* $\alpha = 1.0$*, and the polynomial has* 9 *degrees.*

The bottom plot in figure 16 displays the overall sensitivity of the butterfly spread's value wrt. changes in the underlying swap rate. However, it is important to note the swap rate is an market observable it is not a tradeable instrument. For hedging a portfolio of European swaptions, we are interested in determining hedge-coefficients. This requires a slight modification to our estimation approach. Instead of training on the discounted payoffs, $g(\boldsymbol{x})$, we can use the discounted cashflows, $\frac{B(t)}{B(T_i)}CF(T_i, \boldsymbol{x}_{T_i})$, as our $\boldsymbol{y}$-labels. This alteration decomposes the payoffs and risk factors of the portfolio.

In figure 17, we present this decomposition for the butterfly spread, providing a clearer understanding of the interplay between various products in the portfolio.

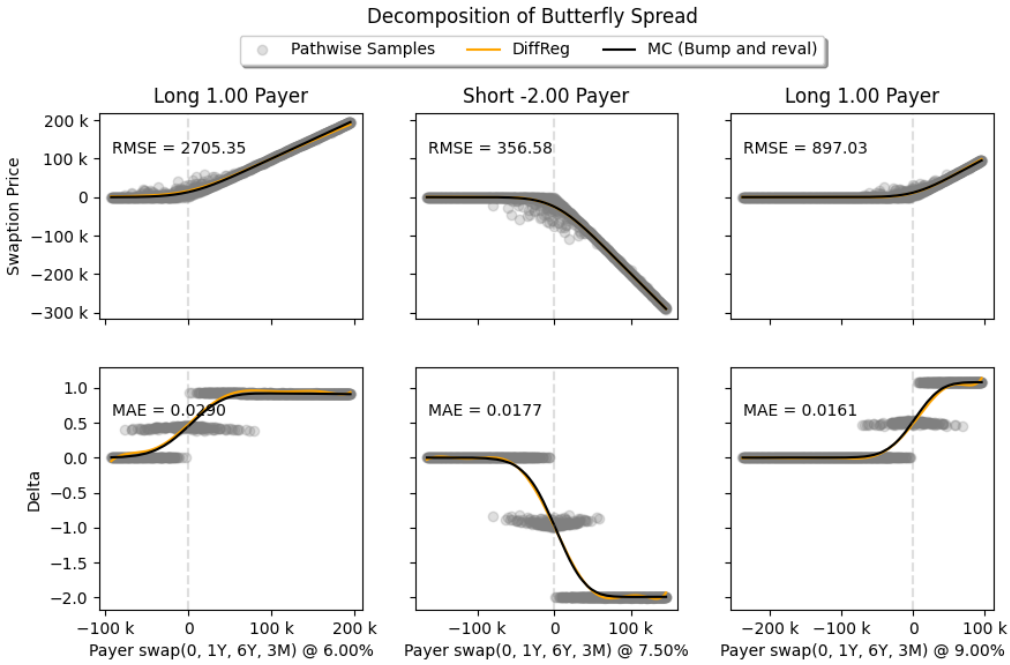
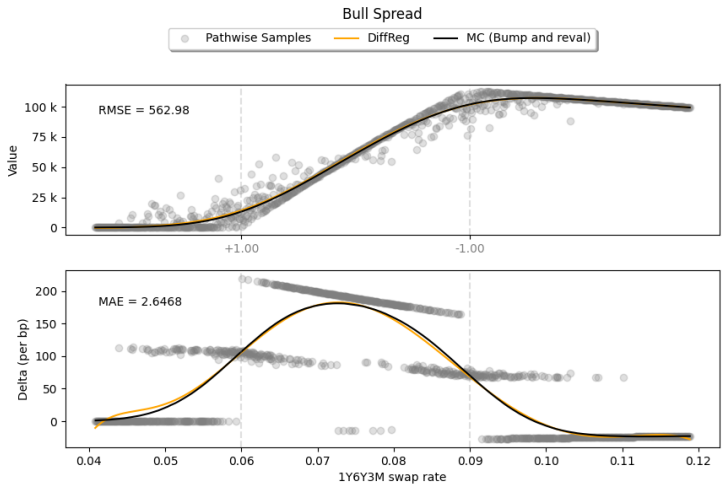


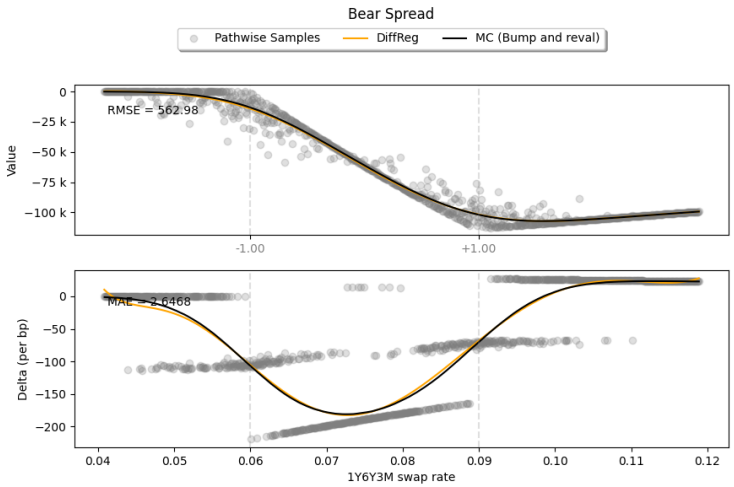
Figure 17: Decomposition of Butterfly spread. Settings are the same as in figure 16.

In summary, we have succesfully determined both the value and risk sensitivity of the entire strategy wrt. the swap rate, and the decomposition of all the individual swaptions in terms of their value functions and deltas wrt. their respective underlying swaps. The computational cost of getting these results, requiring just a single Monte Carlo simulation with relative few paths (e.g. 1,024) and some additional evaluations of the modified normal equation.

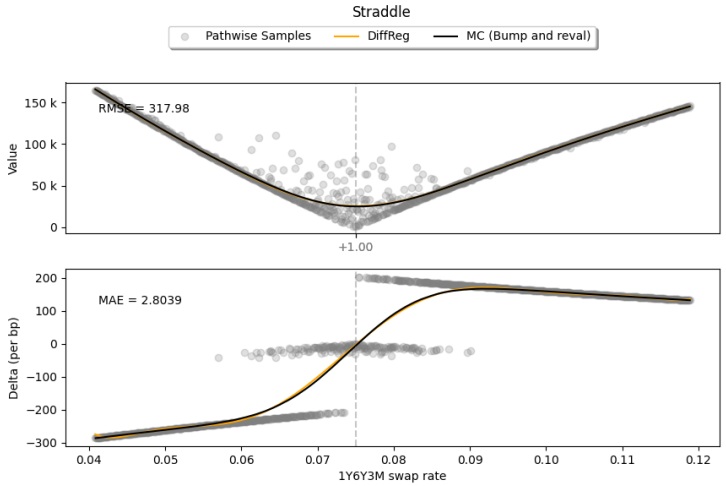
To demonstrate the effectiveness of this method beyond just butterfly spreads, we have replicated the experiment with several other common option strategies. The results of these additional tests are visualized in figure 18, showcasing the method's applicability across a range of different scenarios.



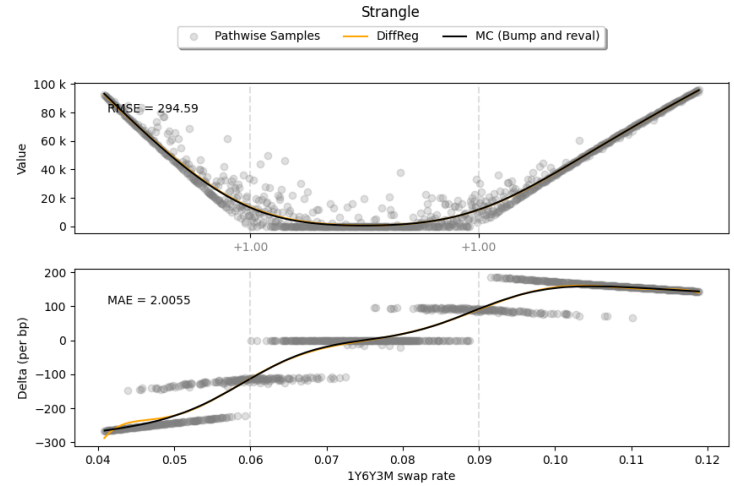
(a) Bull Spread (long 1 payer swaption @ 6.0%. and short 1 payer swaption @ 9.0%.)



(b) Bear Spread (short 1 payer swaption @ 6.0%. and long 1 payer swaption @ 9.0%)



(c) Straddle (long 1 payer swaption @ 7.5% and long 1 receiver swaption @ 7.5%)



(d) Strangle (long 1 payer swaption @ 6.0% and long 1 receiver swaption @ 9.0%.)

Figure 18: Common option strategies as a function of the underlying swap rate. Settings are as in figure 16.

As mentioned previously in subsection 2.3.2 and 4.6 Bermudan options are callable derivatives that, once the optimal exercise strategy has been determined e.g. via the LSM-algorithm, can be modelled as path-dependent products. Similar to a European swaption basket or a portfolio of European swaptions, a co-terminal Bermudan swaption also has several underlying market variables, typically different swaps. Consequently, the differential regressor encounters similar challenges in fitting the delta function, particularly as the expiry time diminishes.

Additionally, since the true optimal stopping rule is only estimated (and not known) using the LSM-algorithm, there is an inherent source of error. The result of conducting the delta-hedge experiment for a co-terminal Bermudan payer swaption with exercise dates in 1, 2, or 5 years on a 10-year swap is shown below in figure 19. The figure uses different colors to represent the exercise decisions of the counterparty according to the LSM-derived stopping rule.

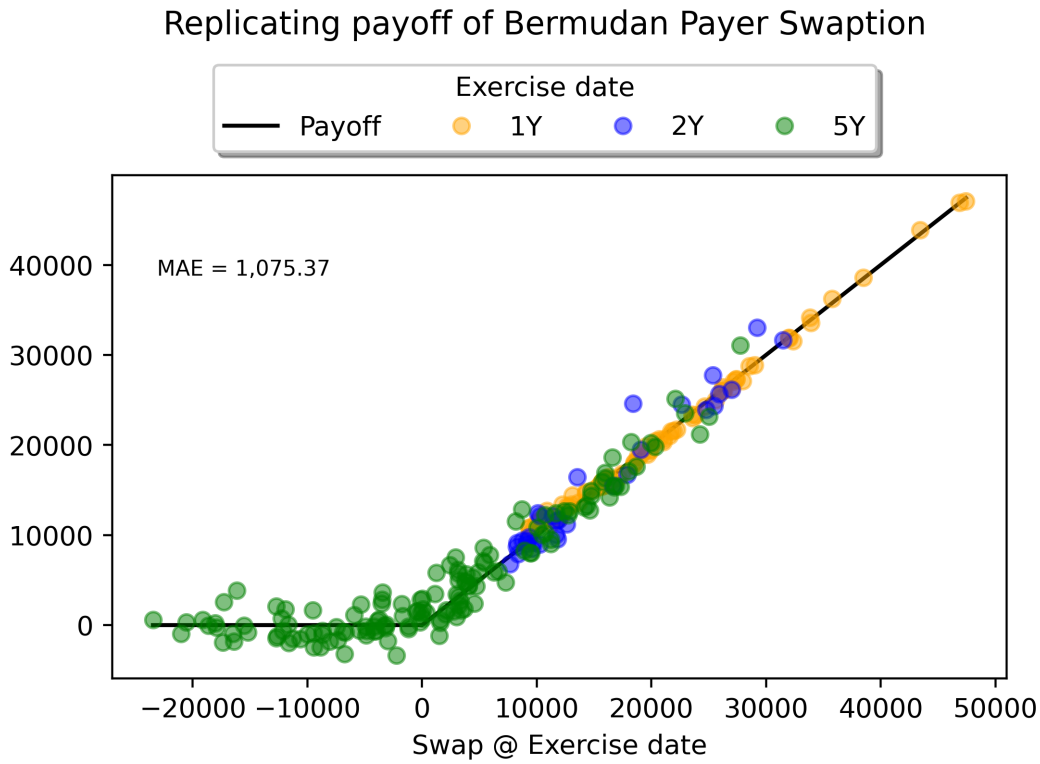


Figure 19: Replication of payoff function for a co-terminal Bermudan Payer Swaption using differential regression trained on 1,024 training samples with 9 degrees, interactions, and $\alpha = 1.0$ for regularization. Exercise dates are 1Y, 2Y, 5Y, maturity of the underlying payer swaps is 10Y, the strike rate is 7.03%, and the notional is 1,000,000. The hedge frequency is 100 times per year. The exercise condition is proxied using LSM with $n = 5,000$ pre-simulations.

Differential regression generally demonstrates its utility in pricing and risk estimation for products without complex payoff structures or excessively steep sensitivities. To address these more challenging scenarios, we explore how the differential neural network handles these issues in the subsequent section.

7.2 Differential Neural Network

Our implementation of the differential neural network rely on the notebook DifferentialMLTF2 accompanying [SH20].

The performance of the neural network depends heavily on the choice of activation function. We have selected *SoftPlus* for two key reasons. Firstly it meets the requirement of being at least twice-differentiable

$$\text{SoftPlus}(x) = \log(1 + e^x), \quad \text{SoftPlus}'(x) = \text{Sigmoid}(x) = (1 + e^{-x})^{-1},$$

with well-defined second order derivative. Secondly, SoftPlus resembles a hockey stick shape, similar to option payoff functions, making it suitable for learning from payoffs. Regarding balancing costs in equation (6.8) between values and derivatives, we calculate coefficients using the principle that derivative errors should match value error in magnitude, meaning

$$a = \frac{1}{1+\lambda}, \quad b = \frac{\lambda}{1+\lambda}.$$

Our neural network is a basic feedforward type with four hidden layers of 20 units each, but the architecture can be more complex. There's a trade-off between complexity for better estimations and computational costs. For optimization, we use the ADAM optimizer, generally considered best practice. The optimizer's role is threefold: it decides if a stationary point is reached, influences which stationary point is reached, and affects the rate of convergence by dictating the learning rate.

Normalizing data in neural network training is crucial, mainly because it balances the scales of loss terms $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{diff}}$ Additionally, due to the sensitivity of neural networks to weight initialization in non-convex optimization, our use of the Xavier-Glorot initialization in TensorFlow assumes centered, orthonormal inputs. Thus, standardized data

mapping

$$x \mapsto \frac{x - \mu_x}{\sigma_x} = \tilde{x},$$

significantly enhances training performance. The symbols μ_x and σ_x represent the sample mean and standard deviation of the input features and output labels $x = \{\mathbf{X}, \mathbf{y}\}$. As a consequence is the j'th vector of derivatives standardized differentiable labels scaled according to

$$\tilde{\mathbf{Z}}_j = \frac{\partial \tilde{\mathbf{y}}}{\partial \tilde{\mathbf{X}}_j} = \frac{\partial(\mathbf{y} - \mu_{\mathbf{y}})/\sigma_{\mathbf{y}}}{\partial(\mathbf{X}_j - \mu_{\mathbf{X}_j})/\sigma_{\mathbf{X}_j}} = \frac{\sigma_{\mathbf{X}_j}}{\sigma_{\mathbf{y}}} \frac{\partial \mathbf{y}}{\partial \mathbf{X}_j} = \frac{\sigma_{\mathbf{X}_j}}{\sigma_{\mathbf{y}}} \mathbf{Z}_j.$$

The model's predicted prices and derivatives must be re-scaled to their original scale.

Figure 20 illustrates the superiority of differential learning over standard deep learning for arbitrarily chosen parameters. Here, the standard neural network struggles especially to capture the shape of the delta curve, while the differential network exhibits significantly improve curve-fitting performance. This indicates the advantage of differential machine learning.

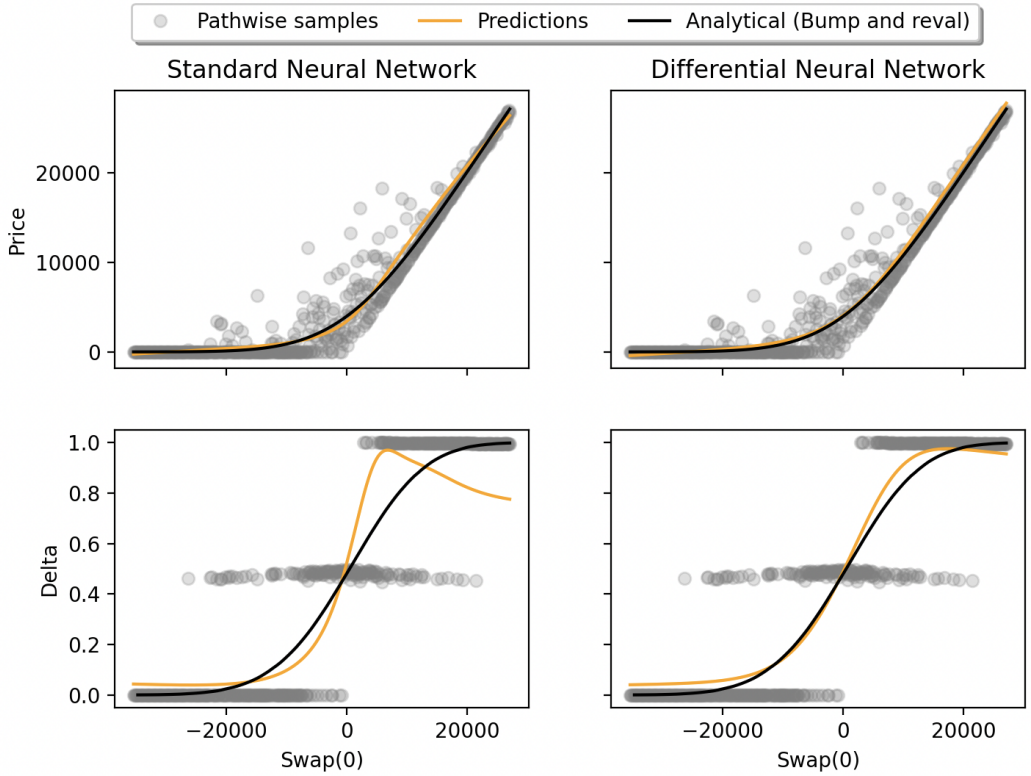


Figure 20: Price and delta estimation for 1Y6Y3M Europoean payer swaption by Standard ($\lambda = 0$) and Differential ($\lambda = 1$) Neural Network. The model is trained on 1,024 AV training samples, with swap strike rate $K = 8.982\%$ and notional $N = 1,000,000$. NN specifications; epocs = 50, batches per epoch = 16, min batch sz = 1,024, layers = 4 with hidden units = 20 in each.

Generally neural networks can obtain greater accuracy than differential regression. We noted previously in figure 11 regression was not capable of fitting the delta function very close to expiry. Similar experiment is shown in figure 21. Clearly this advocates of using neural networks compared to regression, as we can fit the entire delta function for different maturities.

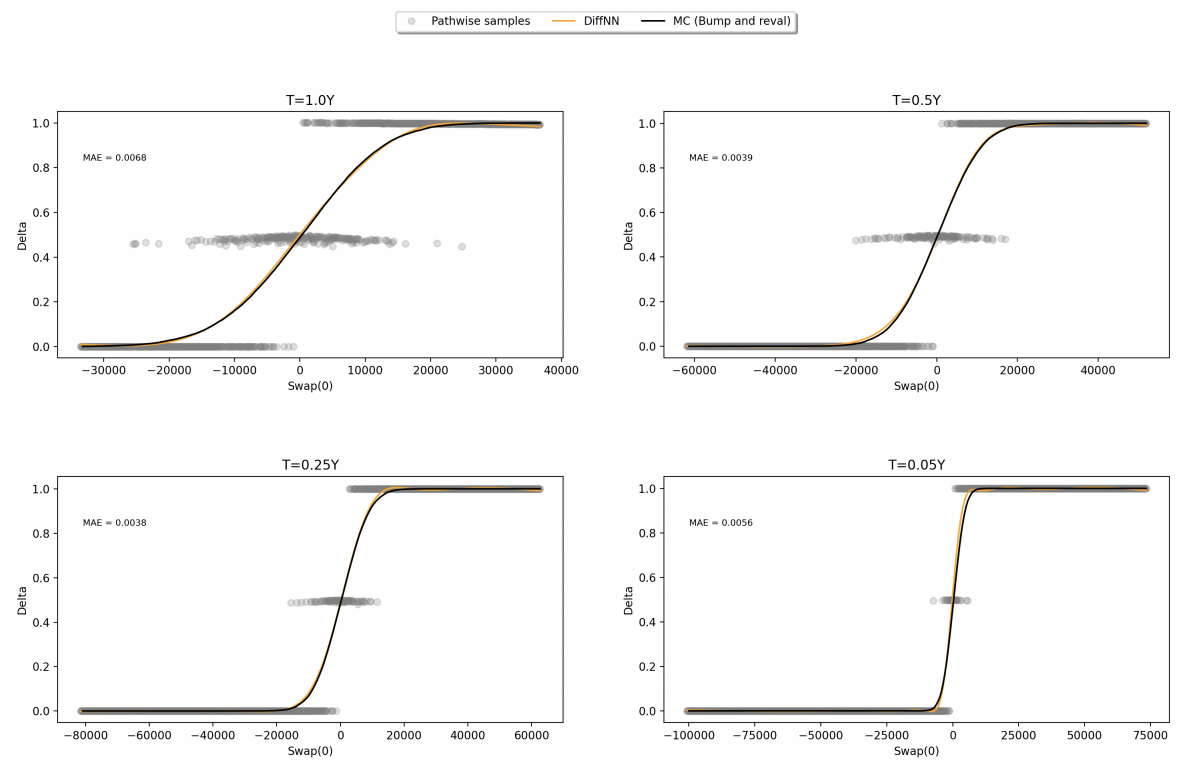


Figure 21: Delta estimation for a 1Y6Y3M European payer swaption until expiry by differential Neural Network. The model is trained on 4096 AV training samples, with swap strike rate $K = 8.71\%$ and notional $N = 1,000,000$. NN specifications; epochs = 500, layers = 4 with hidden units = 20 in each.

Conducting the delta hedge experiment really showcases the performance of differential NNs. From figure 22 we obtain a much better replication of the caplet and the swaption compared to differential regression. It is even unclear whether the error is due to wrong prediction, discretization or monte-carlo estimation.

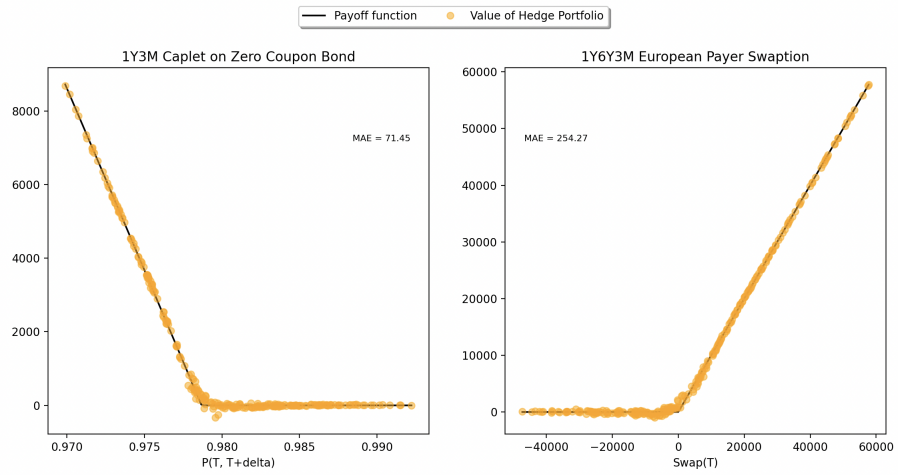


Figure 22: Delta hedge every 2.5 half day of a 1Y3M Caplet on a ZCB and a 1Y6Y3M European payer swaption by differential Neural Network. The model is trained on 4096 AV training samples, with swap strike rate $K = 8.71\%$ and notional $N = 1,000,000$. NN specifications; epochs = 250, layers = 4 with hidden units = 20 in each.

While neural networks can yield satisfactory results, performance is not guaranteed and depends heavily on the architectural choices and hyperparameters used in crafting a neural network estimator. Rather than exploring every possible parameter combination, which would be overly extensive, we focus on outlining the most relevant parameter variations based on our experience; training samples and batch sizes. The impact of these parameters is demonstrated in figure 23, where the standard deviation of the hedge error in a delta hedge is used as a measure. The choice of number and neurons and layers of course also effects the performance of the model. We try to keep the network a reasonable size for best comparison with the regression method and as investigated in [HHA23], having 4 layers with 20 neurons in each, seems adequate for our purposes.

Notably, neural networks augmented with differentiable labels surpass standard networks. It indicates that performance gains are driven by training samples, but also on the right balance of batch size to training data size. The figure even suggests if batch size proportionality is not chosen correctly, like 25% of samples, satisfactory performance is not obtainable. A batch size proportional to 50% of training data size proves to yield the smallest hedge error. In order to choose the particular training size of either 1,024 or 4,096 samples, one has to consider the trade-off between accuracy and the increased time complexity with larger training sizes. We favor the smallest viable training sample. In comparison to 12, the neural network is indeed capable of reducing the hedge error more

than differential regression. A hedge error of 5.5 in the figure, corresponds to a price adjusted hedge error of around 2.2%.

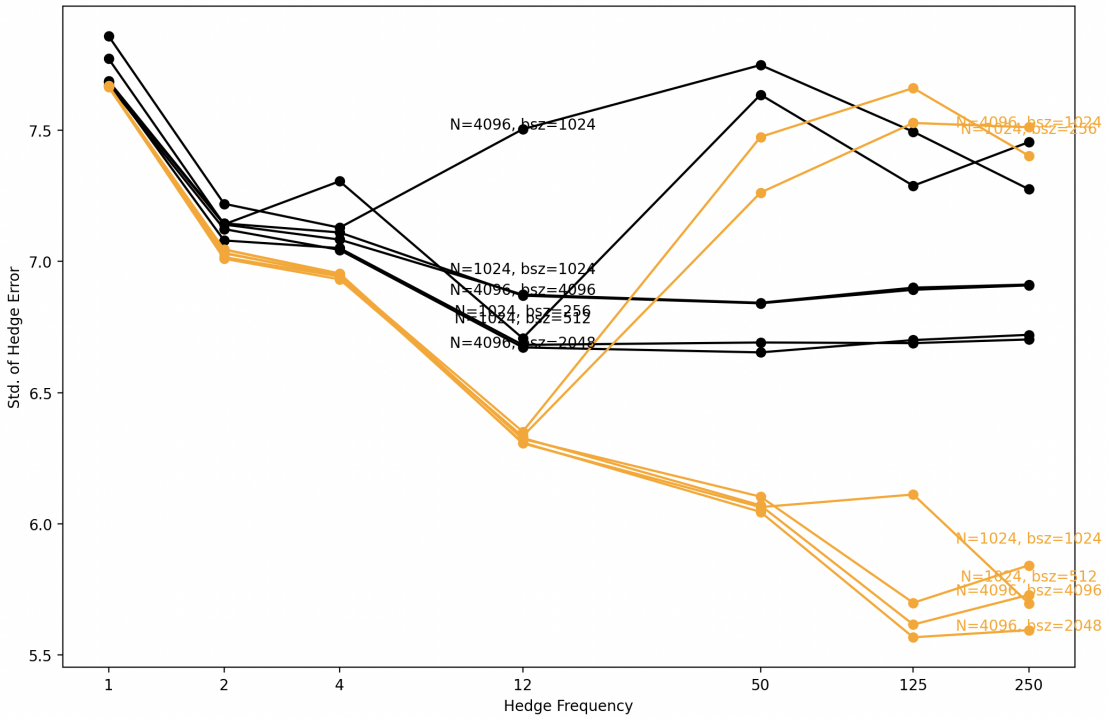


Figure 23: Convergence plot of hedge error for a 1Y6Y3M European payer swaption for standard-(black) and differential neural network (orange) on log-log scale. Simulated from single spot value with strike rate $K = 8.71\%$, varying training samples, $N = \{1024, 4096\}$, batch sizes $\{25\%, 50\%, 100\%\}$ of N and hedge frequency per year.

Beyond vanilla products, neural networks exhibit different behaviours compared to differential regression when applied to exotic derivatives. For example the challenges highlighted in figure 14 are not as problematic for neural networks as evidenced in below figure 24. The neural network seems to capture non-differentiable tendencies in a manner that regression cannot. Despite the difficulty in sampling training points around the barrier, the network appears to approximate the function adequately with relatively few samples (16,384).

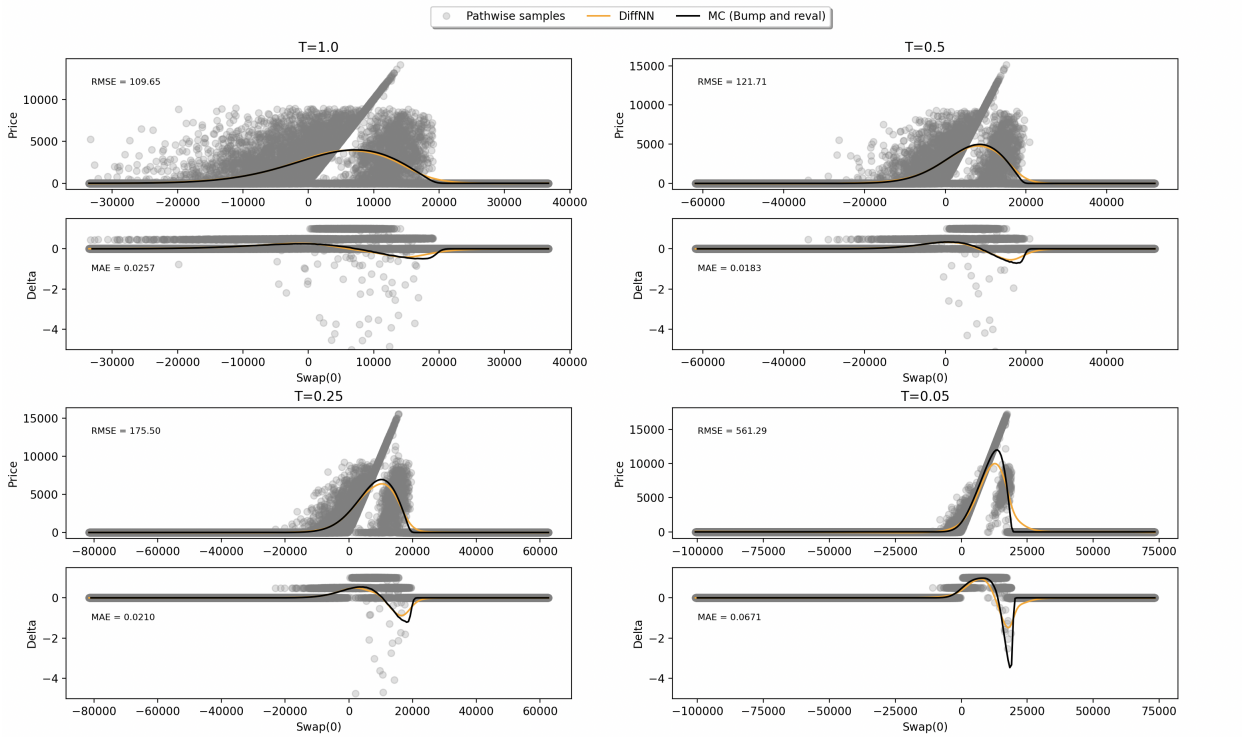


Figure 24: Delta and price estimation for varying maturities of European up-and-out barrier swaptions using differential neural network trained on 16,384 training samples. The barrier is $B = 20,000$ with smoothing interval $\pm 5\%$ of B using sigmoid function. The swaption is monitored every 2.5 day, with a notional of 1,000,000 and strike rate $K = 8.71\%$. The MC price and delta are estimated without any smoothing.

7.3 Considerations

This section has demonstrated the interaction of the Vasicek short rate model with the differential machine learning framework. Showcasing its effectiveness not just in pricing vanilla and exotic interest rate derivatives, but also in enabling more real-world hedge experiments. The importance of training on differentiable labels is evident for estimating risk accurately both for regression and neural networks. The comparison between the two methods is based on the models selected from earlier sections;

- Differential regression with 9'th order polynomial.
- Differential neural network with 50% batch ratio, 250 epochs[14], 4 layers with 20 units in each.

In terms of accuracy, table 1 presents a comparison of these methods for delta hedging a

[14]see figure 39 in appendix B for justification.

1Y3M European caplet, a 1Y6Y3M European payer swaption and a Bermudan swaption with exercise dates 1Y, 2Y, 3Y and 10Y maturity of the underlying payer swap. The experiment is initiated from different states and trained on 1,024 training samples for comparison. The neural network, despite the limited sample size, outperforms differential regression in accuracy.

Product	**Diff. Reg**	**Diff. NN**
Caplet	178.28	61.02
Eur Swaption	347.29	240.41
Ber Swaption	2222.68	1624.51

Table 1: Accuracy comparison of standard deviation of hedge error.

However accuracy is just one dimensionality the frameworks can be compared within. Another dimensionality of interest for overall performance measurement is time complexity. For industrial implementations it is of huge importance, when calculating millions of trades. Table 2 compares the computational time of price estimation for 1,024 trades measured in milliseconds. Differential regression is significantly faster, at least 750 times quicker than differential neural network.

Product	**Diff. Reg**	**Diff. NN**
Caplet	1.429	1310.269
Eur Swaption	1.598	1327.414
Ber Swaption	1.792	1349.301

Table 2: Time complexity comparison of price estimation in milliseconds.

The preferred method depends on the specific context. For estimating the price and risk of numerous derivatives simultaneously, differential regression is much faster and sufficiently accurate. If the products have differentiable structures, the neural network's performance gains do not justify its computational demands. However, for exotic products, neural networks capture nuances that regression misses, beneficial for pricing, risk management, and hedging. Usually more complicated products are traded in much smaller magnitudes, hence the computational speed is of less importance under these circumstances.

8 Heath-Jarrow-Morton Framework

In the previous sections we have build a strong foundation for pricing and computing risk sensitivities for a wide range of interest rate products. The combination of a flexible Monte Carlo backbone with a fast risk propagation allow us to easily expand to models with higher complexity. As illustrated in 2, we can easily interchange the model when valuing a certain interest rate product. So far, we have considered the Vasicek single-factor short rate model as a proof of concept - a model that has numerous limitations in terms of real world applications. Furthermore, we have showed in this context how efficient differential machine learning approaches can estimate prices and risk figures, and thus be used in a hedging context. We wish to extend this concept and showcase the performance in a more complex setting, where the current model state is not just completely given by the short rate itself. We will introduce such a model in section 9, but before doing so, we lay the foundations of its framework, where we model the entire forward rate *curve* in doing so we introduce some well-established results in the Heath-Jarrow-Morton (HJM) framework.

8.1 Modelling Forward Rate Dynamics

Before the emergence of the HJM framework in the early 1990s, the landscape of interest rate modelling had mostly revolved around short-rate models. These are known as diffusion short-rate models, where the short rate follows an Itô process

$$dr(t) = \mu\left(t, r(t)\right) + \sigma\left(t, r(t)\right) dW(t).$$

By this approach, short-rate models are based on modelling a *single* short-term interest rate. This constitutes a framework that is often not flexible enough to accurately capture the entire term structure [Fil09], and challenging to calibrate to market observables. They can be used to model a single rate relatively well, but capturing the dynamics across many different maturities seems as a fundamental challenge.

HJM models, on the other hand, are designed to directly model the entire term structure of interest rates, providing a more comprehensive framework. This is achieved by modelling the instantaneous forward rate curve instead of only the short rate. This is quite a

leap in model ambitions, since this means the framework has to describe the evolution of the entire continuum of maturities T on (instantaneous) forward rates $f(\cdot, T)$ taking offset in a known initial condition $f(0, T)$. This means that HJM models are automatically calibrated to the initial term structure / bond prices, and also comes with the advantages that we describe both the current yield curve shape (or accept it) and the expected future evolution.

While there are multiple advantages of modelling forward rates, it also come with some pitfalls. Theoretically complexities such as ensuring the model is tractable and arbitrage-free will be considered below. Empirically calibration and computational complexity contains challenges, especially for higher dimensions, that needs be handled in a cunning way. Before demonstrating the frameworks flexibility and a particular choice of model, a more formal introduction is required and thus presented below.

The forward rate dynamics under $\mathbb{P}$ is in the very general HJM framework defined as

$$f(t,T) = f(0,T) + \int_0^t \mu(s,T)ds + \int_0^t \sigma(s,T)dW(s), \tag{8.1}$$

where $\mu(\omega, t, T) \in \mathbb{R}$ is the drift and $\sigma(\omega, t, T) \in \mathbb{R}^N$ is a volatility function of a finite number of N stochastic processes $W_1, \ldots, W_N$. To ensure model consistency and tractability the following conditions of the functions are crucial:

1. The mappings $(\omega, s) \mapsto \mu(\omega, s)$ and $(\omega, s) \mapsto \sigma(\omega, s)$ are $\mathcal{B}[0,T] \otimes \mathbb{F}_t$-measurable $\forall s \in [0,T]$ and $\mu(t,T) = \sigma(t,T) = 0$ for $t \geq T$.
2. $\int_0^T \int_0^T |\mu(s,t)|dsdt < \infty \; \forall T \geq 0$.
3. $\sup_{s,t \leq T} ||\sigma(s,t)|| < \infty \; \forall \, T$.

Property 2. is important for modelling the forward rate process as the pricing of zero-coupon bonds (ZCB) satisfy

$$P(t,T) = e^{-\int_t^T f(t,u)du}, \tag{8.2}$$

and thus requires μ to be twice integratable. Additionally the third property imply ω-by-ω non-explosiveness, meaning the model does not predict extremely large and unrealistic

values. The relation between the short-rate and forward-rate is given by

$$r(t) = f(t,t) = f(0,t) + \int_0^t \mu(s,t)ds + \int_0^t \sigma(s,t)dW(s), \tag{8.3}$$

demonstrating the benefit of modelling the entire forward curve instead of the short-rate. Another important part of the framework is how zero coupon bonds are priced, as they are used for both discounting and pricing of many rates and interest-rate derivatives. The following results thus illustrates how the HJM setting is used for ZCB pricing.

Lemma 8.1. For some given maturity T, the ZCB price follows a Itô dynamics of the form

$$P(t,T) = P(0,T) + \int_0^t P(s,T)\left[r(s) + b(s,T)\right]ds + \int_0^t P(s,T)\nu(s,T)dW(s) \quad \forall t \leq T, \tag{8.4}$$

where the bond volatility is

$$\nu(s,T) = -\int_s^T \sigma(s,u)du$$

and

$$b(s,T) = -\int_s^T \mu(s,u)du + \frac{1}{2}||\nu(s,T)||^2.$$

Proof. See appendix C. □

Even though this demonstrates how the underlying factors of the forward rate drives the ZCB bond price, it is at least as much of interest considering the discounted price process. This process will prove to be useful in order to ensure absence of arbitrage. Corollary 8.1 reveal the behaviour of the discounted price process over time.

Corollary 8.1. For $t \leq T$ the discounted bond price process is

$$\frac{P(t,T)}{B(t)} = P(0,T) + \int_0^t \frac{P(s,T)}{B(s)}b(s,T)ds + \int_0^t \frac{P(s,T)}{B(s)}\nu(s,T)dW(s). \tag{8.5}$$

Proof. We recall that the dynamics of the money market account $B(t) = B(0)e^{\int_0^t r(s)ds}$ can be written as

$$dB(t) = r(t)B(t)dt.$$

We apply this along with (C.3) and the Itô chain-rule on the discounted bond price process to obtain:

$$\begin{aligned} d\left(\frac{P(t,T)}{B(t)}\right) &= \frac{1}{B(t)}dP(t,T) - \frac{P(t,T)}{B(t)^2}dB(t) \\ &= \frac{P(t,T)}{B(t)}\left([r(t)+b(t,T)]\,dt + \nu(t,T)dW(t)\right) - \frac{P(t,T)}{B(t)}r(t)dt \\ &= \frac{P(t,T)}{B(t)}\left[b(t,T)dt + \nu(t,T)dW(t)\right]. \end{aligned} \tag{8.6}$$

Which in turns gives us (8.5) by rewriting the dynamics to integral form.

□

Ensuring arbitrage-free conditions in the model is of course a central consideration, otherwise arbitrage-opportunities will be exploited that undermines the principle of efficient markets. This can be guaranteed by enforcing a drift restriction that is demonstrated in theorem 8.1 below. We assume the existence of a risk neutral measure $\mathbb{Q} \sim \mathbb{P}$ defined by the market price of risk for the bond market γ, such that $W^Q(t) = W(t) - \int_0^t \gamma(s)^\top ds$ is the Girsanov transformed Brownian motion.

Theorem 8.1 (Drift Condition)**.** $\mathbb{Q}$ is an equivalent local martingale measure (ELMM) if and only if the T-bond drift

$$b(t,T) = -\nu(t,T)\gamma(t)^\top \ \forall t \geq T,$$

where ν is the T-bond volatility. If so, (8.1) becomes

$$f(t,T) = f(0,T) + \int_0^t \left(\sigma(s,T)\int_s^T \sigma(s,u)^\top du\right)ds + \int_0^t \sigma(s,T)dW^Q(s),$$

and the discounted T-bond price satisfies

$$\frac{P(t,T)}{B(t)} = P(0,T)\mathcal{E}_t\left(\nu(\cdot,T)\bullet W^Q\right), \ \forall t \leq T, \tag{8.7}$$

where

$$\mathcal{E}_t\left(\nu(\cdot,T)\bullet W^Q\right) = e^{\nu(\cdot,T)\bullet W^Q - \frac{1}{2}\langle \nu(\cdot,T)\bullet W^Q,\, \nu(\cdot,T)\bullet W^Q\rangle_t} = e^{\int_0^t \nu(s,T)dW^{\mathbb{Q}}(s) - \frac{1}{2}\int_0^t ||\nu(s,T)||^2 ds}$$

is the stochastic exponential of the Itô process.

Proof. See appendix C. □

The drift condition ensures the bond price process to be a martingale and is the key result for ensuring the absence of arbitrage when modelling interest rates in the HJM framework. Accordingly one of the features of HJM is the distribution of $f(t,T)$ and $P(t,T)$ only depends on $\sigma(t,T)$ under $\mathbb{Q}$ and not $\mu(t,T)$ from $\mathbb{P}$ which is similar to the Black-Scholes model.

In order to have proper discounting under the $\mathbb{Q}$-measure and numeriare pricing, we need the risk free short rate. In the HJM framework the short rate dynamics can be expressed via the proposition below.

Proposition 8.1. The short rate process in the HJM framework is an Itô process of the form

$$r(t) = r(0) + \int_0^t \zeta(u)du + \int_0^t \sigma(u,u)dW(u), \tag{8.8}$$

where

$$\zeta(u) = a(u,u) + \partial_u f(0,u) + \int_0^u \partial_u \alpha(s,u)ds + \int_0^u \partial_u \sigma(s,u)dW(s).$$

Proof. See appendix C. □

8.2 Markovian Heath-Jarrow-Morton Models

The HJM model prescribes a straightforward framework to specify an arbitrage free term structure - all we have to do, is specify the volatility structure. But what seems conceivably simple is in fact a very rich class that shows to be *infinite* dimensional, even in one-factor form.
In general we see from theorem 8.1 that the HJM model requires the whole forward curve as a state variable at time t which makes the Markov dimension of the model infinite. This is even independent of the dimension of the Brownian motion and for deterministic forward rate volatility.

We consider the formulation of the instantaneous forward rate for a couple of maturities

$T_1 < T_2$ to show the HJM setup is in general *non-Markov* for a deterministic volatility structure $\sigma(s,t)$ and one-dimensional Brownian motion:

$$f(t,T_1) = f(0,T_1) + \int_0^t \sigma(s,T_1) \int_s^{T_1} \sigma(s,u)duds + \int_0^t \sigma(s,T_1)dW(s)$$
$$f(t,T_2) = f(0,T_2) + \int_0^t \sigma(s,T_2) \int_s^{T_2} \sigma(s,u)duds + \int_0^t \sigma(s,T_2)dW(s)$$

Notice when we assume a deterministic $\sigma(s,t)$ the first two terms is also deterministic, and the stochastic integrals are Gaussian with mean zero (integral over the centered, pairwise independently Gaussian increments). In this setting for the system to be Markov we would need the existence of a deterministic function α such that

$$f(t,T_2) = \alpha(f(t,T_1),t,T_1,T_2).$$

If we have such relationship, all the forward rates at time t are deduced from one another, and we would be able to describe the state of the model at time t with a state variable of a certain dimension, i.e. making the system Markovian.
Since the stochastic integrals aggregates the random increments from the Brownian motion with different weights $\sigma(s,T_1)$ and $\sigma(s,T_2)$, it makes the problem of finding a function α over-determined. That means the model cannot be Markov in a finite set of state variables. This is of course not very convenient to work with from a computational perspective and it was what motivated Cheyette (1992) [Che92] to impose a certain *separable* restriction on the forward volatility structure.

Chiarella & Kwon (2001) [CK01] shows in a multi-dimensional setting, and Andreasen (2001) [And01] in one-dimension, how to restrict the volatility structure to achieve a finite dimensional, Markovian specification of the HJM framework. This is achieved by first defining the deterministic function $\kappa(t) \in \mathbb{R}^d$ and the progressive process $\eta(t) \in \mathbb{R}^{d\times d}$ to ensure that the volatility is separable, and then being able to represent the dynamics of the forward rate by a Markov system involving a finite set of state variables. If we define

the functions

$$\sigma(s,t) = (\sigma_1(s,t), ..., \sigma_d(s,t)), \tag{8.9}$$

$$\sigma_i(s,t) = \eta_i(s)\, e^{\int_s^t \kappa_i(u)du}, \quad i = 1, .., d \tag{8.10}$$

such that $\eta_i(s)$ is a function of calendar time and $e^{\int_s^t \kappa_i(u)du}$ is a function of maturity, we see initially that the t-derivative of σ_i is equal to

$$\frac{\partial \sigma_i(s,t)}{\partial t} = -\kappa_i(t)\sigma_i(s,t),$$

and $\sigma_i(t,t) = \eta_i(t)$. By proposition 8.1 we can then write the short rate process as

$$dr(t) = \zeta^{\mathbb{Q}}(t)dt + \sum_{i=1}^{d} \eta_i(t) dW^{\mathbb{Q}}(t), \tag{8.11}$$

$$\begin{aligned}\zeta^{\mathbb{Q}}(t) &= \partial_t f(0,t) + \sum_{i=1}^{d} \int_0^t \sigma_i(s,t)^2 ds - \sum_{i=1}^{d} \kappa_i(t) \int_0^t \sigma_i^{\mathbb{Q}}(s,t) ds - \sum_{i=1}^{d} \kappa_i(t) \int_0^t \sigma_i(s,t) dW_i^{\mathbb{Q}}(s) \\ &= \partial_t f(0,t) + \sum_{i=1}^{d} \left(\int_0^t \sigma_i(s,t)^2 ds - \kappa_i(t) \left[\int_0^t \sigma_i^{\mathbb{Q}}(s,t) ds + \int_0^t \sigma_i(s,t) dW_i^{\mathbb{Q}}(s) \right] \right)\end{aligned}$$

where $\sigma_i^{\mathbb{Q}}(s,t) := \sigma(s,t) \int_s^t \sigma(s,u)du$. Introducing the state variables $x(t) = (x_1(t), ..., x_d(t))$ and $y(t) = (y_1(t), ..., y_d(t))$ to argue for "markovianity". We define these as follows:

$$x_i(t) = \int_0^t \sigma_i^{\mathbb{Q}}(s,t) ds + \int_0^t \sigma_i(s,t) dW_i^{\mathbb{Q}}(s), \tag{8.12}$$

$$y_i(t) = \int_0^t \sigma_i(s,t)^2 ds. \tag{8.13}$$

By applying Leibniz rule[15] we obtain the dynamics of x_i, i.e.

$$\begin{aligned}
x_i(t) &= \int_0^t \sigma_i^{\mathbb{Q}}(s,t)ds + \int_0^t \sigma_i(s,t)dW_i^{\mathbb{Q}}(s) \\
&= \left[\sigma_i^{\mathbb{Q}}(t,t) + \int_0^t \partial_t\sigma_i^{\mathbb{Q}}(s,t)ds\right]dt + \sigma_i(t,t)dW(t) + \left(\int_0^t \partial_t\sigma_i(s,t)dW(s)\right)dt \\
&= \left[\sigma_i^{\mathbb{Q}}(t,t) + \int_0^t \left(\partial_t\sigma_i(s,t)\int_s^t \sigma_i(s,u)du\right)ds\right]dt + \sigma_i(t,t)dW(t) + \left(\int_0^t \partial_t\sigma_i(s,t)dW(s)\right)dt \\
&= \left[\eta_i(t) - \int_0^t \left(\kappa_i(t)\sigma_i(s,t)\int_s^t \sigma_i(s,u)du\right)ds\right]dt + \eta_i(t)dW(t) - \left(\int_0^t \kappa_i(t)\sigma_i(s,t)dW(s)\right)dt \\
&= \left[\eta_i(t) - \kappa_i(t)\left(\int_0^t \sigma_i^{\mathbb{Q}}(s,t)ds + \int_0^t \sigma_i(s,t)dW(s)\right)\right]dt + \eta_i(t)dW(t) \\
&= \left(\eta_i(t) - \kappa_i(t)x_i(t)\right)dt + \eta_i(t)dW(t).
\end{aligned}$$

Likewise, we get that the dynamics of y_i by following the same approach to be

$$\begin{aligned}
d(y_i(t)) &= \sigma_i^2(t,t)dt + \int_0^t \partial_t\sigma_i^2(s,t)ds \\
&= \eta_i^2(t)dt + \int_0^t 2(\partial_t\sigma_i(s,t))\sigma_i(s,t)ds \\
&= \eta_i^2(t)dt - \left(2\int_0^t \kappa_i(t)\sigma_i^2(s,t)ds\right)dt \\
&= \eta_i^2(t)dt - \left(2\kappa_i(t)\int_0^t \sigma_i^2(s,t)ds\right)dt \\
&= \left(\eta_i^2(t) - 2\kappa_i(t)y_i(t)\right)dt.
\end{aligned}$$

Notice that this process is of finite variation since it consists of only a drift-term. Inserting these terms in the above expression for the short rate gives

$$dr(t) = \partial_t f(0,t) + \sum_{i=1}^d \left(y_i(t) - \kappa_i(t)x_i(t)\right) + \sum_{i=1}^d \eta_i(t)dW_i^{\mathbb{Q}}(t). \tag{8.14}$$

If $\eta_i(t) = \eta_i(t, x(t), y(t))\ \forall i$, i.e. η_i is a function of the current state then we see this forms a Markov system that is sufficient to describe any point on the yield curve. We

[15] Leibniz Rule: $d\left(\int_0^t f(\omega,s,t)ds\right) = \left(f(\omega,t,t) + \int_0^t \partial_t f(\omega,s,t)ds\right)dt$.
Leibniz rule for stochastic integrals: $d\left(\int_0^t f(\omega,s,t)dW(s)\right) = f(\omega,t,t)dW(t) + \left(\int_0^t \partial_t f(\omega,s,t)dW(s)\right)dt$ with the restrictions that f and its derivatives are square integrable.

can further rewrite the short rate as

$$r(t) = f(0,t) + \sum_{i=1}^{d} x_i(t)$$

to see that the short rate itself is only driven by the state variable $x(t)$. But note that we still need $y(t)$ to be able to describe the current state since it is still working in the background as it shows up in the forward rate and in the zero coupon bond price.

Chiarella & Kwon (2001) [CK01] further express the zero coupon bond price (in essence the term structure) in terms of a finite set of state variables. This gives the backbone in for pricing interest rate products - the bond reconstruction formula - as stated below in theorem 8.2.

Theorem 8.2. The Bond reconstruction formula in [CK01] is given by

$$P(t,T) = \frac{P(0,T)}{P(0,t)} \exp\left(- \sum_{i=1}^{d} \gamma_i(t,T)^2 y_i(t) - \sum_{i=1}^{d} \gamma_i(t,T) x_i(t) \right),$$

where $\gamma_i(t,T) = \int_t^T \exp\left(-\int_t^u \kappa_i(s)ds\right) du$.

A similar formula is derived later in section 9 where we consider the general multi-factor stochastic volatility model by Trolle and Schwartz [TS06].

8.3 Forward Measures and Change of Numeraire

In subsection 4.5 we briefly introduced the forward measure and change of numeraire as a method for reducing the variance of the Monte Carlo simulation. In general change of measure and numeraire is of course much more general and and are especially useful in interest rate modelling. More specifically, changing numeraire means replacing the (risk-free) bank account as numeraire by another traded asset. This proves to be useful for option pricing purposes and we will below derive an explicit option price formula for Gaussian HJM models.

We start by taking outset in the general risk neutral pricing formula which gives the price of a derivative with payoff function Φ on a single market variables, X, with a single

payoff at time T as

$$\pi(t) = \mathbb{E}^{\mathbb{Q}}\left[\frac{B(t)}{B(T)}\Phi(X_T)\big|\mathcal{F}_t\right]. \tag{8.15}$$

As mentioned in the Monte Carlo section, solving this expectation requires the derivation or numeric calculation of a double integral for the joint distribution of $B(T)^{-1}$ and X. Since we are in general interested in pricing interest rate sensitive claims independence of $B(T)^{-1}$ and X seems as an unrealistic assumption to make. However, under the T-forward measure denoted by $\mathbb{Q}^T$ the numeraire - in the form of a zero coupon bond - is measurable at the valuation time t and not only at the expiry T. Thus, allowing us to move the numeraire outside of the expectation.

Let us first introduce what is meant by the *T-forward measure*. For a particular $T > 0$ we can define an equivalent probability measure $\mathbb{Q}^T \sim \mathbb{Q}$ on the filtration $\mathcal{F}_T$ using the Radon-Nikodym derivative. By defining the stochastic likelihood ratio process L as

$$L := \frac{1}{P(0,T)B(T)}$$

and observing it satisfies $L > 0$ since $P(0,T)B(T) > 0$. Furthermore, by using that the discounted time-0 value of the ZCB price is $P(0,T) = \mathbb{E}^{\mathbb{Q}}\left[\frac{1}{B(T)}\right]$, we have that

$$\mathbb{E}^{\mathbb{Q}}[L] = \mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{P(T,T)}{P(0,T)B(T)}\right] = 1.$$

This qualifies L as the Radon-Nikodym derivative of $\mathbb{Q}^T$ with respect to $\mathbb{Q}$ and we can define the equivalent probability measure $\mathbb{Q}^T$ from

$$\frac{d\mathbb{Q}^T}{d\mathbb{Q}} = \frac{1}{P(0,T)B(T)}. \tag{8.16}$$

This allows us to change measure of the expectation as

$$\mathbb{E}^{\mathbb{Q}}[XL] = \mathbb{E}^{\mathbb{Q}^T}[X].$$

We notice that from theorem 8.1 we have the following rewriting

$$\begin{aligned} P(0,T)\mathcal{E}_t\left(\nu(\cdot,T)\bullet W^Q\right) &= \frac{P(t,T)}{B(t)} \\ \Leftrightarrow \mathcal{E}_t\left(\nu(\cdot,T)\bullet W^Q\right) &= \frac{P(t,T)}{P(0,T)B(t)} = \mathbb{E}^{\mathbb{Q}}\left[\frac{P(T,T)}{P(0,T)B(T)}\Big|\mathcal{F}_t\right] \\ &= \mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}\Big|\mathcal{F}_t\right] = \mathbb{E}^{\mathbb{Q}}\left[L|\mathcal{F}_t\right] = \frac{d\mathbb{Q}^T}{d\mathbb{Q}}\Big|\mathcal{F}_t. \end{aligned} \tag{8.17}$$

By then applying Girsanov's Change of Measure theorem we get the $\mathbb{Q}^T$-Brownian motion

$$W^T(t) = W^{\mathbb{Q}}(t) - \int_0^t \nu(s,T)^\top ds, \quad t \leq T. \tag{8.18}$$

From this we make the observation that if $\nu(s,T) = -\int_0^T \sigma(t,u)du$ is deterministic, then the bond prices will be log-normal under $\mathbb{Q}$ but also under any forward measure $\mathbb{Q}^T$. We will revisit this property below. Specifically for zero coupon bonds, we have the following results presented in lemma 8.2.

What is crucial in this setting is that assets discounted with $P(t,T)$ as numeraire becomes martingales under the forward measure $\mathbb{Q}^T$.

Lemma 8.2. For any $S > 0$, the T-bond discounted S-bond price process, that is $\frac{P(t,S)}{P(t,T)}$, is a $\mathbb{Q}^T$-martingale with

$$\frac{P(t,S)}{P(t,T)} = \frac{P(0,S)}{P(0,T)}\mathcal{E}_t\left(\sigma_{S,T}\bullet W^T\right)$$

using the definition $\sigma_{S,T}(t) := \nu(t,S) - \nu(t,T) = \int_S^T \sigma(t,u)du$. Furthermore, the T- and S-forward measures are related by

$$\frac{d\mathbb{Q}^S}{d\mathbb{Q}^T}\Big|_{\mathcal{F}_t} = \frac{P(t,S)P(0,T)}{P(t,T)P(0,S)} = \mathcal{E}_t\left(\sigma_{S,T}\bullet W^T\right)$$

Proof. Consider $u \leq t \leq S \wedge T$. We see from applying (abstract) Bayes' rule that the

price process $P(t,S)/P(t,T)$ is a $\mathbb{Q}^T$-martingale:

$$\begin{aligned}\mathbb{E}^{\mathbb{Q}^T}\left[\frac{P(t,S)}{P(t,T)}\middle|\mathcal{F}_u\right] &= \frac{\mathbb{E}^{\mathbb{Q}}\left[L\frac{P(t,S)}{P(t,T)}\middle|\mathcal{F}_u\right]}{\mathbb{E}^{\mathbb{Q}}\left[L|\mathcal{F}_u\right]} = \frac{\mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}\frac{P(t,S)}{P(t,T)}\middle|\mathcal{F}_u\right]}{\mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}|\mathcal{F}_u\right]} \\ &= \frac{\mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}\frac{P(t,S)}{P(t,T)}\middle|\mathcal{F}_u\right]}{\frac{P(u,T)}{P(0,T)B(u)}} = \frac{\mathbb{E}^{\mathbb{Q}}\left[\frac{P(t,T)}{P(0,T)B(t)}\frac{P(t,S)}{P(t,T)}\middle|\mathcal{F}_u\right]}{\frac{P(u,T)}{P(0,T)B(u)}} \\ &= \frac{\frac{P(u,S)}{B(u)}}{\frac{P(u,T)}{B(u)}} = \frac{P(u,S)}{P(u,T)},\end{aligned}$$

using expression (8.17) and the tower property. Following the same reasoning as in the proof of theorem 8.1 we can express the price process using the stochastic exponential. The last statement follows from the Radon-Nikodym property $\frac{dQ^S}{dQ^T} = \frac{dQ^S}{dQ}\frac{dQ}{dQ^T}$.

□

Turning back to the pricing equation in (8.15), we can now perform a change of measure from the risk neutral, $\mathbb{Q}$, to the forward measure, $\mathbb{Q}^T$, which effectively means that $B(T)^{-1}$ and $X(T)$ becomes independent under the expectation as claimed earlier. We now show how this is carried out for a general T-claim X which we assume satisfies the condition

$$\mathbb{E}^{\mathbb{Q}}\left[|X|/B(T)\right] < \infty.$$

First, by applying(*abstract*) Bayes' Rule, we get an auxiliary result stating that

$$\mathbb{E}^{\mathbb{Q}^T}\left[|X|\right] = \frac{\mathbb{E}^{\mathbb{Q}}\left[L|X|\right]}{\mathbb{E}^{\mathbb{Q}}\left[L\right]} = \mathbb{E}^{\mathbb{Q}}\left[\frac{|X|}{P(0,T)B(T)}\right] < \infty.$$

Secondly, from applying the abstract Bayes' Rule to (8.15) and multiplying with $1 = P(0,T)/P(0,T)$ which is time-0 measurable, we can carry out all of the necessary calcu-

lations as follows

$$\begin{aligned}
\pi(t) &= B(t)\mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B(T)}\middle|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\mathbb{E}^{\mathbb{Q}}\left[\frac{X}{P(0,T)B(T)}\middle|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\mathbb{E}^{\mathbb{Q}}\left[XL\middle|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\mathbb{E}^{\mathbb{Q}^T}\left[X\middle|\mathcal{F}_t\right]\mathbb{E}^{\mathbb{Q}}\left[L|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\,\mathbb{E}^{\mathbb{Q}^T}\left[X\middle|\mathcal{F}_t\right]\,\mathbb{E}^{\mathbb{Q}}\left[\frac{1}{P(0,T)B(T)}\middle|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\,\mathbb{E}^{\mathbb{Q}^T}\left[X\middle|\mathcal{F}_t\right]\,\frac{1}{P(0,T)}\mathbb{E}^{\mathbb{Q}}\left[\frac{P(T,T)}{B(T)}\middle|\mathcal{F}_t\right] \\
&= P(0,T)B(t)\,\mathbb{E}^{\mathbb{Q}^T}\left[X\middle|\mathcal{F}_t\right]\,\frac{P(t,T)}{P(0,T)B(t)} \\
&= P(t,T)\mathbb{E}^{\mathbb{Q}^T}\left[X|\mathcal{F}_t\right]. \qquad (8.19)
\end{aligned}$$

This expression is much more preferred exactly because of independence under the forward measure. This means we now only have to the integral over a single stochastic variable in order to evaluate the expectation. This is naturally a great computational saving in a Monte Carlo setup as stated in subsection 4.4. Implementing this in practice, requires that we model the diffusion process under the forward measure which in most cases are easily obtained by an application of Girsanov's theorem. We also note the useful fact in equation (8.19) that $P(t,T)$ is an observable quantity at time t and does not have to be computed inside the simulation for each path.

8.4 Vasicek Specification

As mentioned above, the HJM framework is a very general setup. The generality is also what can be problematic when it comes to practical implementations. Often it is required to make special assumptions on the volatility structure $\sigma(t,T)$ to impose a finite-dimensional Markovian representation of the forward rate curve, as we showed in above subsection 8.2. Many popular interest rate models can be shown to be a certain instance of a HJM model by choosing the volatility process careful enough. In fact, any arbitrage-free interest rate model with the filtration generated exclusively by Brownian motions is a special case of an HJM model [AP10a]. We will show below, how this can

be done for the Vasicek model. Recall the short rate dynamic in the Vasicek model is defined as

$$dr(t) = a\left[b - r(t)\right]dt + \sigma dW^{\mathbb{Q}}.$$

The short rate process in HJM (by using the drift condition) is given by equation (8.3) to be

$$r(t) = f(t,t) = f(0,t) + \int_0^t \left(\sigma(s,t)\int_s^t \sigma(s,u)^\top du\right) ds + \int_0^t \sigma(s,t) dW^{\mathbb{Q}}(s). \tag{8.20}$$

By defining the volatility process $\sigma(s,t)$ as

$$\sigma(s,t) := \sigma e^{-a(t-s)}, \tag{8.21}$$

and imposing the following restriction on the initial instantaneous forward curve $f(0,t)$

$$f(0,t) = r(0)e^{-at} + b\left(1 - e^{-at}\right) - \int_0^t \left(\sigma(s,t)\int_s^t \sigma(s,u)^\top du\right) ds,$$

we see that, we can rewrite in order to get the Vasicek model formulation. Explicitly inserting $\sigma(s,t)$ in the above expression, we get

$$\begin{aligned}
f(0,t) &= r(0)e^{-at} + b\left(1-e^{-at}\right) - \int_0^t \left(\sigma e^{-a(t-s)} \int_s^t \sigma e^{-a(u-s)} du\right) ds \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \sigma^2 \int_0^t e^{-a(t-s)} \left[\frac{1}{a} e^{-a(u-s)}\right]_{u=s}^{u=t} ds \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \sigma^2 \int_0^t e^{-a(t-s)} \left[\frac{1}{a}\left(e^{-a(t-s)} - 1\right)\right] ds \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \frac{\sigma^2}{a} \int_0^t e^{-2a(t-s)} - e^{-a(t-s)} ds \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \frac{\sigma^2}{a} \left[\frac{e^{-2a(t-s)}}{2a} - \frac{e^{-a(t-s)}}{a}\right]_{s=0}^{s=t} \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \frac{\sigma^2}{a} \left[\left(\frac{1}{2a} - \frac{1}{a}\right) + \left(\frac{e^{-at}}{a} - \frac{e^{-2at}}{2a}\right)\right] \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \frac{\sigma^2}{a^2} \left[\left(\frac{1}{2} - 1\right) + \left(e^{-at} - \frac{e^{-2at}}{2}\right)\right] \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) + \frac{\sigma^2}{a^2} \left[-\frac{1}{2}\left(1 - 2e^{-2at} + e^{-at}\right)\right] \\
&= r(0)e^{-at} + b\left(1-e^{-at}\right) - \frac{\sigma^2}{2a^2}\left(1-e^{-at}\right)^2 . \qquad (8.22)
\end{aligned}$$

By inserting this restriction on $f(0,t)$ back into (8.20) gives us the same solution for the short rate, as we had in equation (3.3) in section 3. From this equation (8.22) can then

be used to calculate ZCB prices using the relation

$$
\begin{aligned}
&P(0,t) \\
&= \exp\left(-\int_0^t f(0,s)ds\right) \\
&= \exp\left(-\int_0^t r(0)e^{-as} + b\left(1-e^{-as}\right) - \frac{\sigma^2}{2a^2}\left(1-e^{-as}\right)^2 ds\right) \\
&= \exp\left(-r(0)\int_0^t e^{-as}ds - b\int_0^t \left(1-e^{-as}\right)ds + \frac{\sigma^2}{2a^2}\int_0^t \left(1-e^{-as}\right)^2 ds\right) \\
&= \exp\left(-r(0)\left[\frac{-e^{-as}}{a}\right]_{s=0}^{s=t} - b\left[s - \frac{-e^{-as}}{a}\right]_{s=0}^{s=t} + \frac{\sigma^2}{2a^2}\left[s - \frac{e^{-2as}}{2a} + 2\frac{e^{-as}}{a}\right]_{s=0}^{s=t}\right) \\
&= \exp\left(\frac{r(0)}{a}\left[e^{-at}-1\right] - b\left[\left(t + \frac{e^{-at}}{a}\right) - \left(0 + \frac{1}{a}\right)\right]\right. \\
&\qquad \left. + \frac{\sigma^2}{2a^2}\left[\left(t - \frac{e^{-2at}}{2a} + 2\frac{e^{-at}}{a}\right) - \left(0 - \frac{1}{2a} + \frac{2}{a}\right)\right]\right) \\
&= \exp\left(\frac{r(0)}{a}\left[e^{-at}-1\right] - \frac{b}{a}\left[at + e^{-at} - 1\right] + \frac{\sigma^2}{2a^3}\left[at - \frac{e^{-2at}}{2} + 2e^{-at} + \frac{1}{2} - 2\right]\right) \\
&= \exp\left(\frac{(r(0)-b)\left(e^{-at}-1\right)}{a} - bt + \frac{\sigma^2 t}{2a^2} + \frac{\sigma^2}{4a^3}\left[4e^{-at} - e^{-2at} - 3\right]\right). \qquad (8.23)
\end{aligned}
$$

This result is equivalent to equation (3.7). From equation (8.23), we note the Markovian structure of the Vasicek model becomes quite clear: The zero-coupon bond price (or the instantaneous forward rate) and thus the yield curve can be completely determined by today's given state $r(0)$ and deterministic expressions.

8.4.1 Caplet Pricing

In the setting of Vasicek we can derive an explicit analytical expression for many different European options - we choose the caplet price as an example. This is of course preferred in practice, but we can also use it as a sanity check for our Monte Carlo engine. We showed in equation 2.5 how the caplet can be viewed as a scaled European put option on the zero-coupon bond (irrespective of the model choice). Thus, we can rewrite this

expression using forward measures which gives

$$\begin{aligned}
Cpl(0;T,T+\delta) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&\quad - \frac{1}{\bar{K}}\mathbb{E}^{\mathbb{Q}}\left[e^{-\int_0^T r(s)ds}P(T,T+\delta)\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&= \mathbb{E}^{\mathbb{Q}^T}\left[P(0,T)B(T)e^{-\int_0^T r(s)ds}\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&\quad - \frac{1}{\bar{K}}\mathbb{E}^{\mathbb{Q}^{T+\delta}}\left[P(0,T+\delta)B(T+\delta)e^{-\int_0^T r(s)ds}P(T,T+\delta)\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&= P(0,T)\mathbb{E}^{\mathbb{Q}^T}\left[\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&\quad - \frac{1}{\bar{K}}P(0,T+\delta)\mathbb{E}^{\mathbb{Q}^{T+\delta}}\left[\frac{P(T,T+\delta)}{P(T,T+\delta)}\mathbb{1}\left\{\bar{K} > P(T,T+\delta)\right\}\right] \\
&= P(0,T)\mathbb{Q}^T\left[\bar{K} > P(T,T+\delta)\right] \\
&\quad - P(0,T+\delta)\frac{1}{\bar{K}}\mathbb{Q}^{T+\delta}\left(\bar{K} > P(T,T+\delta)\right). \qquad (8.24)
\end{aligned}$$

In the simple setting of the Vasicek model we have a constant volatility structure given by (8.21). Hence, as we argued via Girsanov's theorem in (8.18) this means that bond prices will be log-normally distributed under both $\mathbb{Q}^T$ and $\mathbb{Q}^{T+\delta}$. From lemma 8.2 we have that

$$\frac{P(T,T+\delta)}{P(T,T)} = \frac{P(0,T+\delta)}{P(0,T)}\mathcal{E}_T\left(\sigma_{T+\delta,T} \bullet W^T\right),$$

where $\sigma_{S,T}(t) = \int_S^T \sigma(t,u)du$. Writing the above expression out we see that

$$\frac{P(T,T+\delta)}{P(T,T)} = \frac{P(0,T+\delta)}{P(0,T)}\exp\left\{\int_0^T \sigma_{T,T+\delta}(s)dW^T(s) - \frac{1}{2}\int_0^T \|\sigma_{T+\delta,T}(t)\|^2 ds\right\}$$

is log-normal. On the contrary $X := \log\left(\frac{P(T,T+\delta)}{P(T,T)}\right)$ must be normal with the following moments:

$$\mathcal{N}\left(\log\left(\frac{P(0,T+\delta)}{P(0,T)}\right) - \frac{1}{2}\int_0^T \|\sigma_{T,T+\delta}(t)\|^2 ds;\ \int_0^T \|\sigma_{T+\delta,T}(s)\|^2 dW^T(s)\right). \qquad (8.25)$$

We can thus calculate the probabilities in the above expression (8.24) for the caplet price under the respective forward measures to be

$$\begin{aligned}\mathbb{Q}^T\left(\bar{K} > P(T,T+\delta)\right) &= \mathbb{Q}^T\left(\bar{K} > \frac{P(T,T+\delta)}{P(T,T)}\right) \\ &= \Phi(\bar{K} > e^X) \\ &= \Phi\left(\frac{\log(\bar{K}) - \mathbb{E}^{\mathbb{Q}^T}[X]}{\sqrt{\mathrm{var}^{\mathbb{Q}^T}(X)}}\right). \end{aligned} \tag{8.26}$$

We follow the same procedure for $\mathbb{Q}^{T+\delta}$ by again using lemma 8.2 to write

$$\frac{1}{P(T,T+\delta)} = \frac{P(T,T)}{P(T,T+\delta)} = \frac{P(0,T)}{P(0,T+\delta)}\mathcal{E}_T\left(\sigma_{T,T+\delta} \bullet W^{T+\delta}\right),$$

which is log-normal. As before $Y := \log\left(\frac{1}{P(T,T+\delta)}\right)$ must be normal with the following moments:

$$\mathcal{N}\left(\log\left(\frac{P(0,T)}{P(0,T+\delta)}\right) - \frac{1}{2}\int_0^T \|\sigma_{T,T+\delta}(t)\|^2 ds;\ \int_0^T \|\sigma_{T+\delta,T}(s)\|^2 dW^T(s)\right). \tag{8.27}$$

Then similar to above we can express the probability as

$$\mathbb{Q}^{T+\delta}\left(\bar{K} > P(T,T+\delta)\right) = \mathbb{Q}^{T+\delta}(\bar{K} > e^{-Y}) = \Phi\left(\frac{\log(\bar{K}) - \mathbb{E}^{\mathbb{Q}^T}[Y]}{\sqrt{\mathrm{var}^{\mathbb{Q}^T}(Y)}}\right) =: \Phi(d_2). \tag{8.28}$$

Combining the pricing equation with expression (8.26) and (8.28) we obtain the caplet price

$$\begin{aligned} Cpl(0;T,T+\delta) &= P(0,T)\mathbb{Q}^T\left(\bar{K} > P(T,T+\delta)\right) \\ &\quad - P(0,T+\delta)\frac{1}{\bar{K}}\mathbb{Q}^{T+\delta}\left(\bar{K} > P(T,T+\delta)\right) \\ &= P(0,T)\Phi(d_1) - \frac{P(0,T+\delta)}{\bar{K}}\Phi(d_2). \end{aligned} \tag{8.29}$$

Here the terms d_1 and d_2 are given by

$$d_1 = \frac{\log\left(\frac{P(0,T)\bar{K}}{P(0,T+\delta)}\right) + \frac{1}{2}\int_0^T \|\sigma_{T+\delta,T}(s)\|^2 ds}{\sqrt{\int_0^T \|\sigma_{T+\delta,T}(s)\|^2 ds}},$$
$$d_2 = d_1 - \sqrt{\int_0^T \|\sigma_{T+\delta,T}(s)\|^2 ds}.$$

These are obtained from inserting the moments (8.25) and (8.27) into (8.26) and (8.28), respectively. This particular nice formula is widely known as Black's formula.

In the setting of the Vasicek model, we can express this formula using the volatility structure from (8.21) to be

$$\begin{aligned}\sigma_{S,T}(s) &= \int_S^T \sigma e^{-a(u-s)} du = \left[-\frac{\sigma}{a} e^{-a(u-s)}\right]_{u=S}^{u=T} \\ &= \frac{\sigma}{a}\left(e^{-a(S-s)} - e^{-a(T-s)}\right).\end{aligned}$$

Here it is useful to notice that $\|\sigma_{T+\delta,T}(s)\|^2 = \|\sigma_{T,T+\delta}(s)\|^2$ since

$$\begin{aligned}\|\sigma_{T+\delta,T}(s)\|^2 &= \left|\left|\frac{\sigma}{a}\left(e^{-a(T+\delta-s)} - e^{-a(T-s)}\right)\right|\right|^2 \\ &= \frac{\sigma^2}{a^2}\left(e^{-2a(T+\delta-s)} + e^{-2a(T-s)} - 2e^{-a(2T+\delta-2s)}\right) \\ &= \|\sigma_{T,T+\delta}(s)\|^2.\end{aligned}$$

Evaluating the integrals in (8.29) results in

$$\begin{aligned}\int_0^T \|\sigma_{T+\delta,T}(s)\|^2 ds &= \int_0^T \frac{\sigma^2}{a^2}\left(e^{-2a(T+\delta-s)} + e^{-2a(T-s)} - 2e^{-a(2T+\delta-2s)}\right) ds \\ &= \frac{\sigma^2}{a^2}\left(\left[\frac{1}{2a}e^{-2a(T-s)}\right]_{s=0}^{s=T} + \left[\frac{1}{2a}e^{-2a(T+\delta-s)}\right]_{s=0}^{s=T} - 2\left[\frac{1}{2a}e^{-a(2T+\delta-2s)}\right]_{s=0}^{s=T}\right) \\ &= \frac{\sigma^2}{2a^3}\left(1 - e^{-2aT} + e^{-2a\delta} - e^{-2a(T+\delta)} - 2\left(e^{-a\delta} - e^{-a(2T+\delta)}\right)\right).\end{aligned}$$

This gives us an equation for direct computation of the Vasicek-caplet from the model parameters which was what we wanted.

The Vasicek model is of course a very simple specification of the HJM framework and as discussed earlier it of little practical use in today's markets. However, we build upon many of the same principle for more sophisticated and useful models. This is covered in the next section where we consider a specification of the HJM framework that is both markovian due to its separability in the volatility function, and has a multi-factor, stochastic volatility specification. Thus, we gain all of the neat properties from the HJM framework - such as the guarantee of absence of arbitrage - for free while maintaining a very general and flexible model specifiation.

9 Stochastic Volatility

So far our projected has demonstrated effectiveness in estimating pricing and hedging interest rate derivatives in the Vasicek model. However, the Vasicek model is generally considered somewhat limited in its ability to fitting actual market states, due to its lack of flexibility. To adress this, we introduced the HJM framework, which is a much broader framework containing many different classes of term structure models. With the aim of showcasing our framework of combining Monte Carlo methods with AAD in a more adaptable and market relevant context. To achieve this we introduce in this section the General Multi-Factor Stochastic Volatility Model by Trolle and Schwartz [TS06]. The model offers sufficient flexibility for describing a large verity of market regimes while remaining tractable.

9.1 Introductory Remarks

In single-factor models where volatility is determined by the state variable, like in the Vasicek model, all volatility is explained solely through the randomness of the state variable. To make this point clear it's instructive to revisit the zero-coupon bond price in the Vasicek context. Being part of the ATS framework, the zero-coupon bond price is a deterministic function of the state variable $r(t)$. With reference to equation (3.7) we showed

$$P(t,T) = e^{-A(t,T)-B(t,T)r(t)},$$

where the functions A, B satisfies the Ricatti equations in (3.5) - (3.6). Because of this feature, all sources of randomness and risk is thus accounted for in the zero-coupon bond.

Empirical studies, like those in [TS06] and [CCDG05], indicate the need to incorporate additional sources of randomness into the modelling of interest rate volatilities. Models that allow the volatility to be driven by a separate Brownian motion are typically referred to as *stochastic volatility* models. These models come with the benefit of better fitting market implied volatility smiles, as they can account for varying implied volatilities on options across different strikes and expiration dates.

A simple extension of the Vasicek model to include a stochastic variance process is carried

out in the Fong-Vasicek (FV) model, resembling a variant of the Heston model:

$$dr(t) = a(b - r(t))dt + \sqrt{z(t)}dW_1^{\mathbb{Q}}(t),$$
$$dz(t) = \kappa(\theta - z(t))dt + \eta\sqrt{z(t)}dW_2^{\mathbb{Q}}(t),$$

where $\langle dW_1^{\mathbb{Q}}(t), dW_2^{\mathbb{Q}}(t)\rangle = \rho dt$ and $a, b, \kappa, \theta, \eta \geq 0$. In this extension, zero-coupon bond prices also satisfies a similar ATS structure. As detailed in [AP10b], the ZCB price satisfies the equation

$$P(t,T) = e^{A(t,T)+r(t)B(t,T)+z(t)C(t,T)},$$

where A, B, C again are deterministic functions. Thus $P(t,T)$ is a deterministic function of the two state variables $r(t)$ and $z(t)$. This leads to the same conclusion as before: exposures to both $r(t)$ and $z(t)$ can be hedged by taking positions in two zero-coupon bonds with different maturities.

A common distinction in stochastic volatilty models is between *spanned* and *unspanned* stochastic volatility. *Spanned* stochastic volatility occurs if trading in zero-coupon bonds can hedge against risk from a stochastic volatility variable. In this scenario, the variable is said to be spanned by the discount curve. If all stochastic variables are spanned by the discount curve, as in above FV model, we can in fact infer the levels of those from two points on the yield curve and inverting the system. Moves in e.g. implied volatility for a certain interest rate product would also be present in the yield curve. This means effectively, that we can hedge our vega risk by investing in zero-coupon bonds.

Conversely, *unspanned* stochastic volatility refers on the other hand to the case, where volatilities of discount bond prices depends on a vector of random state variables $(z_1(t), ..., z_n(t))$ that are not included in the state variables used in the reconstruction formula for the discount curve. This feature, or version of stochastic volatility, seems to be better backed empirically as seen in [CCDG05]. These models are usually referred to as *Unspanned* stochastic volatility models or *true* stochastic volatility models. [And06] defines this feature more formally. Assuming the zero-coupon bond price is described by the process:

$$\frac{dP(t,T)}{P(t,T)} = r(t)dt + a(t,T)^{\top}dW^{Q}(t),$$

where a is some vector-valued function, a *true* stochastic volatility model must then have the property, that there exists some factor z and at least one maturity $\mathcal{T}$ such that:

$$\frac{\partial a(t,\mathcal{T})}{\partial z} \neq 0 \quad \text{and} \quad \frac{\partial P(t,T)}{\partial z} = 0, \forall T. \tag{9.1}$$

This implied the existence of a risk factor in the model, that is not captured, or *spanned*, by the zero-coupon bond.

9.2 A General Multi-Factor Stochastic Volatility Model

We consider the general multi-factor stochastic volatility model by [TS06] to describe the interest rate term structure, by extending the Heath-Jarrow-Morton framework we described in section 8. For the rest of this work we will refer to the model by *TS-MFSV*. The model extends the HJM framework by introducing a stochastic variance process $\nu(t)$ to scale the volatility structure $\sigma_{f,i}(t,T)$, encapsulating the concept of *stochastic volatility*. Additionally the model describes the term structure using multiple factors, which is a feature that is widely acknowledged as required to capture the entire dynamics of the term structure, as highlighted in [RS91].

We let the time-t instantaneous forward rate be represented by $f(t,T)$ and the *TS-MFSV* model defines the dynamics as follows:

$$\begin{aligned} df(t,T) &= \mu_f(t,T)dt + \sum_{i=1}^{N} \sigma_{f,i}(t,T)\sqrt{\nu_i(t)}dW_i^{\mathbb{Q}}(t), \\ d\nu_i(t) &= \kappa_i(\theta_i - \nu_i(t))dt + \sigma_i\sqrt{\nu_i(t)}\left(\rho_i dW_i^{\mathbb{Q}}(t) + \sqrt{1-\rho_i^2}dZ_i^{\mathbb{Q}}\right), \end{aligned} \tag{9.2}$$

for $i = 1, ..., N$ where $W_i^{\mathbb{Q}}(t)$ and $Z_i^{\mathbb{Q}}(t)$ denote independent standard Wiener processes under the risk-neutral measure $\mathbb{Q}$. We have N factors driving the forward rates and $N \times 2$ stochastic drivers. Additionally, a notable aspect is that stochastic volatility is imperfectly correlated with the forward rates, a feature which is typically not considered in HJM models. The model has some degree of *unspanned* stochastic volatility with the exception of the cases when perfect correlation (i.e. $\rho = \{1, -1\}$).

The model bears great resemblance with the Heston model [Hes93] and the intuition of the variance parameters are quite similar:
σ_i controls the volatility of the variance process and affects in essence the curvature of the volatility smile. κ_i is the mean reversion parameter, it controls the speed of which deviations are pulled back towards the long-term mean-reversion level θ_i. Note that increasing κ_i reduces the long-term variance of ν_i, consequently limit the effects of stochastic variance process on the volatility smile for longer-dated maturities.

This model exhibits an important feature of describing the dynamics of the forward rate curve using a finite number of state variables similar to what we described in section 8.2 and the focus of the paper [CK01]. This aspect is highly practical for efficient Monte Carlo simulation and pricing interest rate derivatives.

Furthermore, the model facilitates semi-analytical pricing formula for caplets/floorlets as zero-coupon bond options using Fourier inversion similar to the derivations from the Heston model. It's also possible to derive solutions for swaption prices relying on the stochastic duration approximation by [Mun99] for multi-factor models. These pricing formulas are of course useful for unit testing a Monte Carlo engine, as we have done below, but can also be useful for calibration purposes. We will however not be considering calibration within this project, but rely on the one performed in the paper [TS06].

The TS-MFSV model has a wide range useful features that extend beyond the scope of our project, offering potential for further considerations. For instance, one ability of the model is to fit implied cap skews with the dynamics of implied volatilities. Another feature is applied in the context of estimating the value of swaptions *relative* to caps. This can be used e.g. in statistical arbitrage contexts or to evaluate what is the cheapest way to hedge an interest rate risk. Additionally, the model shows a good forecasting performance of interest rates and interest rate derivatives. This can be useful in Value-At-Risk calculations for portfolios consisting of interest rates derivatives.

9.2.1 Deriving the Instantaneous Forward Rate Curve

Using the HJM drift condition from 8.1 to ensure absence of arbitrage we know the drift term in equation (9.2) must be given by:

$$\mu_f(t,T) = \sum_{i=1}^{N} \nu_i(t)\sigma_{f,i}(t,T) \int_t^T \sigma_{f,i}(t,u)du, \tag{9.3}$$

and the dynamics of $f(t,T)$ under risk-neutral measure is completely determined by the initial forward rate curve, the forward rate volatility structure and the volatility state variables. The forward rate volatility structure is

$$\sigma_{f,i}(t,T) := (\alpha_{0,i} + \alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}. \tag{9.4}$$

This structure has separability in calendar and maturity time as we argued in section 8.2, meaning it has a Markovian term structure. This comes with a great practical importance when it comes to simulating from the model which we will elaborate further on below. The forward rate volatility function is visualized in below figure 25 for a specific set of parameters according to table 3 for the model with $N = 3$.

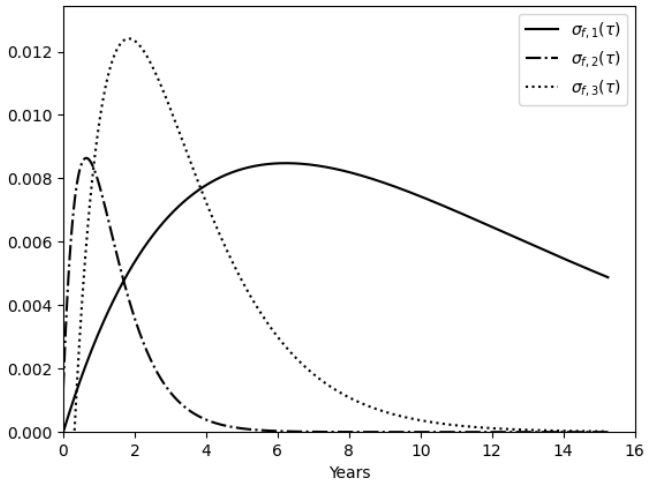


Figure 25: Forward rate volatility functions for $N = 3$ and parameter setting according to table 3. We observe they are hump shaped, while $\sigma_{f,1}$ affects the entire forward rate curve, $\sigma_{f,2}$ affects the short end and $\sigma_{f,3}$ affects mainly the intermediate part.

To derive a solution for the instantaneous forward rate curve we can insert the volatility

structure (9.4) into the drift condition (9.3) to obtain the drift of the forward rate

$$\mu_f(t,T) = \sum_{i=1}^{N} \nu_i(t)\Big[\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i(T-t)}-e^{-2\gamma_i(T-t)}\right) - \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t)e^{-2\gamma_i(T-t)} + \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(T-t)\left(e^{-\gamma_i(T-t)}-e^{-2\gamma_i(T-t)}\right) - \frac{\alpha_{1,i}^2}{\gamma_i}(T-t)^2e^{-2\gamma_i(T-t)}\Big].$$

This is in turn inserted to the model dynamics from equation (9.2) and solving the stochastic integrals with the specific choice of volatility structure from equation (9.4) gives us a solution for the instantaneous forward rate

$$f(t,T) = f(0,T) + \sum_{i=1}^{N}\mathcal{B}_{x_i}(T-t)x_i(t) + \sum_{i=1}^{N}\sum_{j=1}^{6}\mathcal{B}_{\phi_{j,i}}(T-t)\phi_{j,i}(t), \tag{9.5}$$

which is described by a finite number of state variables according to

$$\begin{aligned}
x_i(t) &= \int_0^t \sqrt{\nu_i(s)}e^{-\gamma_i(t-s)}dW_i^{\mathbb{Q}}(s),\\
\phi_{1,i}(t) &= \int_0^t \sqrt{\nu_i(s)}(t-s)e^{-\gamma_i(t-s)}dW_i^{\mathbb{Q}}(s),\\
\phi_{2,i}(t) &= \int_0^t \nu_i(s)e^{-\gamma_i(t-s)}ds,\\
\phi_{3,i}(t) &= \int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds,\\
\phi_{4,i}(t) &= \int_0^t \nu_i(s)(t-s)e^{-\gamma_i(t-s)}ds,\\
\phi_{5,i}(t) &= \int_0^t \nu_i(s)(t-s)e^{-2\gamma_i(t-s)}ds,\\
\phi_{6,i}(t) &= \int_0^t \nu_i(s)(t-s)^2e^{-2\gamma_i(t-s)}ds.
\end{aligned}$$

This representation of the forward rate exhibits the Markovian structure, where the dynamics of the forward curve can be described by a finite set of $N \times 8$ state variables. In above formulation we only have $x_i(t), \phi_{1,i}(t), ..., \phi_{6,i}(t)$ appearing, where $\nu_i(t)$ is an integrated part of the system. This is a similar structure to what is described in [CK01]

and [Che92]. In (9.5) we have defined the functions

$$\mathcal{B}_{x_i}(\tau) = (\alpha_{0,i} + \alpha_{1,i}\tau)e^{-\gamma_i\tau},$$
$$\mathcal{B}_{\phi_{1,i}}(\tau) = \alpha_{1,i}e^{-\gamma_i\tau},$$
$$\mathcal{B}_{\phi_{2,i}}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(\alpha_{0,i} + \alpha_{1,i}(T-t)),$$
$$\mathcal{B}_{\phi_{3,i}}(\tau) = -\left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right) + \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma} + 2\alpha_{0,i}\right)(T-t) + \frac{\alpha_{1,i}^2}{\gamma}(T-t)^2\right)e^{-2\gamma_i(T-t)},$$
$$\mathcal{B}_{\phi_{4,i}}(\tau) = \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)e^{-\gamma(T-t)},$$
$$\mathcal{B}_{\phi_{5,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i} + 2\alpha_{0,i} + 2\alpha_{1,i}(T-t)\right)e^{-2\gamma_i(T-t)},$$
$$\mathcal{B}_{\phi_{6,i}}(\tau) = -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}.$$

The closer details on deriving these expressions can be found in the appendix D which also derives the state variable dynamics which is needed when simulating. The $\phi_{1,i}(t), ..., \phi_{6,i}(t)$ variables are mainly established for ease of notation and to illustrate the Markovian structure of the forward rate. In particular, they have no stochastic terms and thus serve solely as *auxiliary* state variables, reflecting the path movements caused by the stochastic drivers in $x_i(t)$ and $\nu_i(t)$. Due to the Markovian property we can simulate the forward rate (9.5) from a simple Euler scheme. An example of this simulation is visualized in figure 26 with more practical details and considerations in section 9.3.

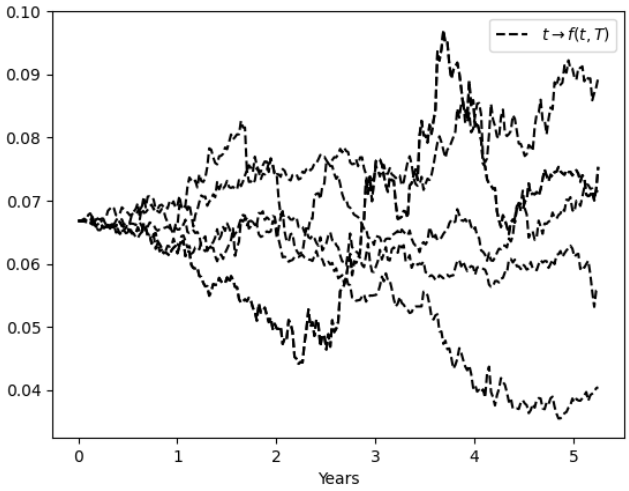


Figure 26: Five sample paths of the instantaneous forward rate according to eq. (9.5) prevailing at time t with maturity $T = 5.25y$ for a model instantiation with factor dimension $N = 3$ and parameter setting according to table 3 and setting the level $f(0,T) = 0.0668$.

9.2.2 A Bond Reconstruction Formula

Indeed, having an analytical expression for the zero-coupon bond price is highly beneficial. We derived in section 8.2 a bond reconstruction formula for an HJM model instance. We can find a similar expression within this model by inserting the solution for the forward rate from equation (9.5) into the relation 1.5

$$\begin{aligned} P(t,T) &= \exp\left\{-\int_t^T f(t,u)du\right\} \\ &= \exp\left\{-\int_t^T f(0,u) - \int_t^T \sum_{i=1}^{N} \mathcal{B}_{x_i}(u-t)x_i(t) - \int_t^T \sum_{i=1}^{N}\sum_{j=1}^{6} \mathcal{B}_{\phi_{j,i}}(u-t)\phi_{j,i}(t)du\right\}. \end{aligned} \tag{9.6}$$

By inserting the function definitions, working out the integrals in the exponential functions and rewriting, we obtain a zero-coupon bond reconstruction formula as

$$\begin{aligned} P(t,T) &= \exp\left\{-\int_t^T f(t,u)du\right\} \\ &= \frac{P(0,t)}{P(0,t)} \exp\left\{\sum_{i=1}^{N} B_{x_i}(T-t)x_i(t) + \sum_{i=1}^{N}\sum_{j=1}^{6} B_{\phi_{j,i}}(T-t)\phi_{j,i}(t)\right\} \end{aligned} \tag{9.7}$$

Using the same definitions of the state variables as in the forward rate (9.5), but defining the new functions:

$$
\begin{aligned}
B_{x_i}(\tau) &= \frac{\alpha_{1,i}}{\gamma_i}\left(\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right)+\tau e^{-\gamma_i\tau}\right),\\
B_{\phi_{1,i}}(\tau) &= \frac{\alpha_{1,i}}{\gamma_i}\left(e^{-\gamma_i\tau}-1\right),\\
B_{\phi_{2,i}}(\tau) &= \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right)+\tau e^{-\gamma_i\tau}\right),\\
B_{\phi_{3,i}}(\tau) &= -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{2\gamma_i^2}+\frac{\alpha_{0,i}}{\gamma_i}+\frac{\alpha_{0,i}^2}{2\alpha_{1,i}}\right)\left(e^{-2\gamma_i\tau}-1\right)+\left(\frac{\alpha_{1,i}}{\gamma_i}+\alpha_{0,i}\right)\tau e^{-2\gamma_i\tau}+\frac{\alpha_{1,i}}{2}\tau^2 e^{-2\gamma_i\tau}\right),\\
B_{\phi_{4,i}}(\tau) &= \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right),\\
B_{\phi_{5,i}}(\tau) &= -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{\gamma_i}+\alpha_{0,i}\right)\left(e^{-2\gamma_i\tau}-1\right)+\alpha_{1,i}\tau e^{-2\gamma_i\tau}\right),\\
B_{\phi_{6,i}}(\tau) &= -\frac{1}{2}\left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(e^{-2\gamma_i\tau}-1\right).
\end{aligned}
$$

The details of the proof can be found in appendix E.

Moreover by applying Ito's lemma to the bond price $P(t,T) = e^{-\int_t^T f(t,u)du}$, we can derive the dynamics of the zero-coupon bond price. This approach is similar to what we demonstrated in the general Heath-Jarrow-Morton (HJM) context with lemma 8.1, and specifically in equation (C.3)

$$
\frac{dP(t,T)}{P(t,T)} = r(t)dt + \sum_i^N B_{x_i}(T-t)\sqrt{\nu_i(t)}dW_i^{\mathbb{Q}}(t). \tag{9.8}
$$

Where the dynamic of the variance process carries along

$$
d\nu_i(t) = \kappa_i(\theta_i - \nu_i(t))dt + \sigma_i\sqrt{\nu_i(t)}\left(\rho_i dW_i^{\mathbb{Q}}(t) + \sqrt{1-\rho_i^2}dZ_i^{\mathbb{Q}}\right).
$$

This further demonstrates that changes in bond prices are solely driven by the $dW_i^{\mathbb{Q}}(t)$-evolution and, to some extend, uncorrelated with changes in instantaneous variance, depending on the ρ_i parameters. We observe in fact that the term structure is unaffected by the drift and (to some extend) the diffusion from the variance process. In that sense, volatility cannot be derived from the term structure alone, but requires information

contained in derivatives. This justifies to some extend the need for interest rate derivatives as pointed out in [CCDG05] to complete fixed income markets.
Simulating the zero-coupon bond price process is just as easy as the forward rate since the Markovian property carries on. We have visualized below how an initial term structure could look like, when using the parameter configuration from table 3 for $N = 3$.

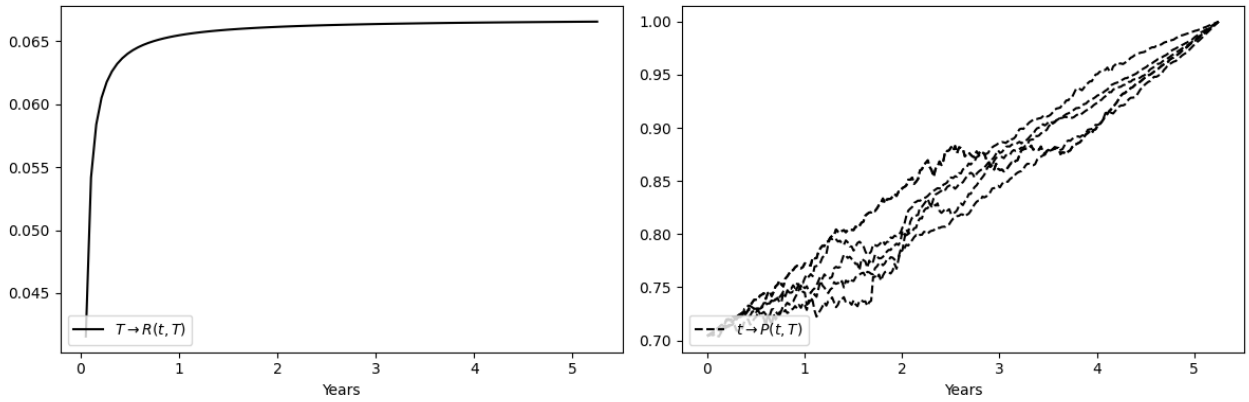


Figure 27: Left plot shows the term structure in terms of zero-coupon bond yields prevailing at time $t = 0$ for various maturity horizons T. Right plot visualizes five sample paths of the zero-coupon bond evolution through time t with fixed maturity $T = 5.25y$, as according to equation (9.7). Both plots stems from a model instantiation with factor dimension $N = 3$ with parameter setting according to table 3 and providing the level $P(0,T)/P(0,t) = e^{-0.0668(T-t)}$.

9.2.3 Semi Analytical Bond Option Pricing

For pricing bond options with a closed form solution, we can resort to the method developed by [CDG03], extending the Fourier transform methodology to a HJM context, by introducing the transform:

$$\psi(u,t,T_0,T_1) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-\int_t^{T_0} r(s)ds} e^{u\log(P(T_0,T_1))}\right].$$

It can be shown, that this transform has an exponentially affine solution (this is done in proposition 2 of [TS06], but is not our focus here):

$$\begin{aligned}&\psi(u,t,T_0,T_1) \qquad (9.9)\\ &= \exp\left(M(T_0-t) + \sum_t^N N_i(T_0-t)\nu_i(t) + u\log(P(t,T_1)) + (1-u)\log(P(t,T_0))\right)\end{aligned}$$

where $M(\tau)$ and $N_i(\tau)$ solve the system of ODEs

$$\frac{dM(\tau)}{d\tau} = \sum_{i=1}^{N} N_i(\tau)\kappa_i\theta_i, \tag{9.10}$$

$$\begin{aligned}\frac{dN_i(\tau)}{d\tau} &= N_i(\tau)(-\kappa_i + \sigma_i\rho_i(uB_{x_i}(T_1 - T_0 + \tau) + (1-u)B_{x_i}(\tau))) + \frac{1}{2}N_i(\tau)^2\sigma_i^2 \\ &+ \frac{1}{2}(u^2 - u)B_{x_i}(T_1 - T_0 + \tau)^2 + \frac{1}{2}((1-u)^2 - (1-u))B_{x_i}(\tau)^2 \\ &+ u(1-u)B_{x_i}(T_1 - T_0 + \tau)B_{x_i}(\tau).\end{aligned} \tag{9.11}$$

By applying the Fourier inversion theorem as in [CDG03] to equation (9.9), we can price options where the underlying is the zero-coupon bond. The time t price of a European put option with expiry T_0 and strike K on a zero-coupon bond maturing at T_1 is then given by:

$$\mathcal{P}(t, T_0, T_1, K) = KG_{0,1}(\log K) - G_{1,1}(\log K) \tag{9.12}$$

where $G_{a,b}(\cdot)$ is the Gil-Pelaez formula

$$G_{a,b}(y) = \frac{\psi(a, t, T_0, T_1)}{2} - \frac{1}{\pi}\int_0^\infty \frac{\mathrm{Im}\left[\psi(a + iub, t, T_0, T_1)e^{-iuy}\right]}{u}du \tag{9.13}$$

in which i denote the unit imaginary number $i = \sqrt{-1}$. This formula is particularly useful for interest rate products that can be modelled as European style options with the zero-coupon bond as underlying. Instances of such products are caplets and floorlets. In equation (2.5) we showed how a caplet can be be valued as a scaled European put option on a zero-coupon bond. This is of course extendable to caps and floors, since these are just portfolios of caplets and floorlets. Therefor, above formula provides *semi*-analytical pricing on those claims.

The *semi*-analytical is in this context referring to the requirement of implementing computationally heavy numerical routines to evaluate the Gil-Pelaez formula twice. For each of these instances, we have to solve the system of ODEs given by above equations (9.10)-(9.11) for each discretization step we take in the numerical integration. The practical details for how we compute this is elaborated on below.

9.3 Implementation and Practical Considerations

This subsection is dedicated to the practical details and choices for numerical methods we have resorted to. Firstly, to simulate the state variables of the model as they evolve over time, and secondly to compute the analytical pricing formula (9.12). Monte Carlo simulation has become our backbone for efficient pricing and risk computations using the differential machine learning methods and this will of course also be described below.

9.3.1 Monte Carlo Simulation

We have intentionally build our code library very generic to handle different models with the purpose of resembling a "near" production Monte Carlo setup. The flexibility makes it possible to reuse components responsible for product implementation, random number generation and the Monte Carlo framework. The model module can easily be swapped with an implementation of the *TS-MFSV* model as visualized in figure 2.

In the Vasicek model we had a single variable to describe the current state entirely, namely the short rate itself, driven by one stochastic driver. This Markov property is also shared with the *TS-MFSV* model, but we have in this setting $N \times 8$ state variables: $\nu_i(t), x_i(t), \phi_{1,i}(t), ..., \phi_{6,i}(t)$. As a result of this property, much of the practicalities are already dealt with, when simulating from the model. The random number generator provides us with the Gaussian increments needed for the risk driving processes. These are sampled from a multivariate standard Gaussian distribution with dimensions determined from the number of paths and factor dimension $N \times 2$ derived from the model's number of risk driving processes.

We use an Euler scheme to discretize the SDEs of the state variables from equations (D.9)-(D.15). The formulations are proven rigorously in appendix D. Simulation procedure of the state variables follows the procedure described in section 4.3, where this scheme generalizes easily to multiple dimensions. We note however, that the model has a correlation structure between the Brownian motions driving the instantaneous forward rate dynamic and the instantaneous variance process in (9.2). A way to impose this correlation can be done via Cholesky decomposition as we described in section 4.3.1. We choose this decomposition method due to its simplicity and seems best practice for mod-

elling correlated Brownian motions.

When simulating the variance process from a Euler scheme we also face the possibility of $\nu_i(t)$ becoming negative. This is even the case when the Feller condition[16] is met for the calibrated parameters to make sure that $\nu_i(t)$ does not attain its boundary value (i.e. 0). If the generated Brownian motion increment is negative enough such that

$$\kappa_i(\theta_i - \nu_i(t_h))\Delta_h + \sigma_i\sqrt{\nu_i(t_h)\Delta_h}W^{\mathbb{Q}}_{v,i}(t_h) < 0$$

we obtain $\nu_i(t_{h+1}) < 0$. The probability of $\nu_i(t_{h+1}) < 0$ becoming negative in a simple Euler scheme is given by:

$$\mathbb{P}(\nu_i(t_h) < 0) = \mathcal{N}\left(\frac{-(1-\kappa_i\Delta_h)\nu_i(t_h) - \kappa_i\theta_i\Delta_h}{\sigma_i\sqrt{\nu_i(t_n)\Delta_h}}\right)$$

To circumvent the computation of $\sqrt{\nu_i(t)}$ for negative values, a straightforward approach is to take the absolute value $|\nu_i(t)|$. This approach has been implementation, primarily because of its ease of integration with an Euler scheme. The disadvantages of this method is that it introduces a bias to some extend for certain paths. In our numerical tests we find that this has a limited effect on pricing accuracy. There exists other methods to avoid introducing this bias such as approximating the distribution of $\nu_i(t)$ by a log-normal scheme or by a transformed volatility scheme to avoid calculating the problematic term $\sqrt{\nu_i(t)}$ entirely. Both of these (and further) are considered in [Zhu10].

In this work the focus is on pricing and hedging performance using differential machine learning, rather developing a calibration / estimation routine ourselves. There is however useful connections between calibrating a model and powerful computation of risk sensitivities that is mentioned in sources like [Sav19]. We thus rely on the parametric estimation performed in [TS06] and reported in their tables 1 and 2. Their estimation procedure is carried out by running a maximum-likelihood procedure of an (extended) Kalman filter, where a measurement equation describes how the observable quantities as the term structure and derivatives prices are inferred by a latent state vector (i.e. a

[16] This is pointed out in [Zhu10] for the Heston model and in the *TS-MFSV* model means the condition $2\kappa_i\theta_i \geq \sigma_i^2$ is satisfied.

vector of the model's state variables) allowing for measurement errors.

The data set used in the estimation consists of weekly observations (as Friday closing mid-quotes) of LIBOR/swap term structures and log-normal implied ATMF swaption and cap volatilities from August 21, 1998 until July 8, 2005. The LIBOR/swap term structures consist of LIBOR rates with maturities of 3mth, 6mth and 9mth and swap rates with maturities 1yr, 2yr, 3yr, 5yr, 7yr, 10yr and 15yr. The caps have length 1yr, 2yr, 3yr, 4yr, 5yr, 7yr and 10yr.

The model is estimated for different numbers of risk driving processes. In our implementations we have used the estimated model for respectively $N = 1$ (one term structure factor and one volatility factor) and for $N = 3$ (three term structure factors and three volatility factors). We have restated the parameter estimates for these model instances in below table 3.

	$\boldsymbol{N=1}$	$\boldsymbol{N=3}$		
	$i=1$	$i=1$	$i=2$	$i=3$
κ_i	0.0553	0.2169	0.5214	0.8340
σ_i	0.3325	0.6586	1.0212	1.2915
$\alpha_{0,i}$	0.0045	0.0000	0.0014	-0.0085
$\alpha_{1,i}$	0.0131	0.0037	0.0320	0.0272
γ_i	0.3341	0.1605	1.4515	0.6569
ρ_i	0.4615	0.0035	0.0011	0.6951
θ_i	0.7542	1.4235	0.7880	1.2602

Table 3: Parameter estimates cf. [TS06].

To provide an intuition on how the model state is evolving, we illustrate below one sample path wrt. the state variables. The path is generated by initializing the model with factor dimension $N = 3$ and using the parameter estimates reported in table 3

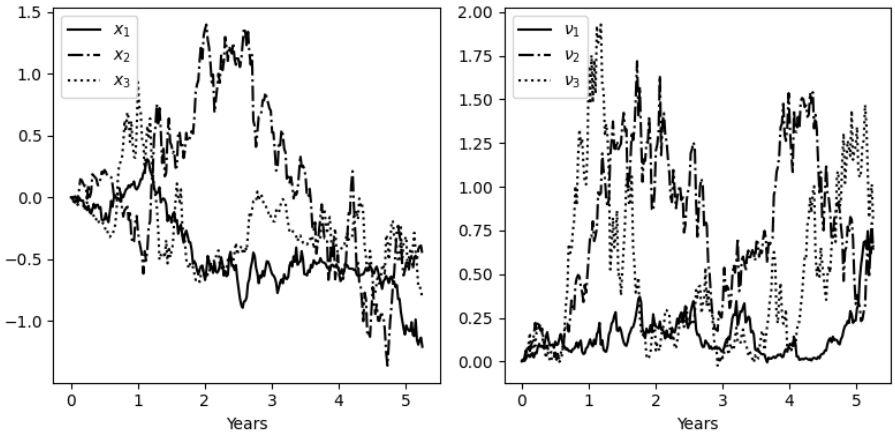


Figure 28: A single sample path of the risk driven state variables $x_i(t), \nu_i(t)$ for $i = 1, 2, 3$.

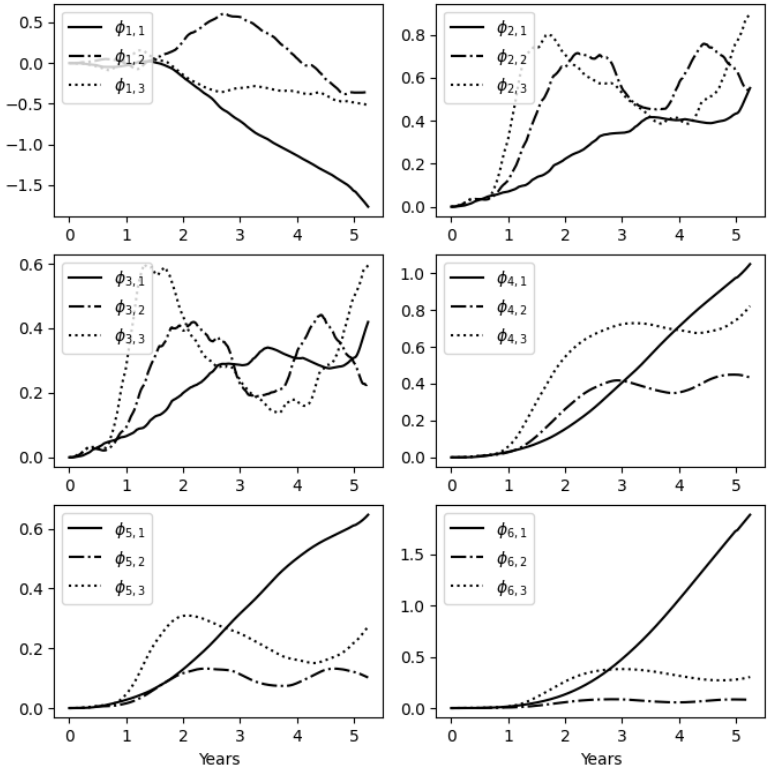


Figure 29: A single sample path of the deterministic state variables $\phi_{j,i}(t)$ derived from $x_i(t), \nu_i(t)$ for $j = 1, .., 6$ and $i = 1, 2, 3$.

9.3.2 Numerical Approximation of the Analytical Bond Option Price

Computing the analytical form in (9.12) requires several numerical approximation techniques. We solve the system of ODEs by (9.10) - (9.11) using a fourth-order Runge-Kutta algorithm. This has to be solved for each integration point of the integral in the Gil-

Pelaez formula (9.13). We discretize the integral by 50 steps using a Trapezoidal quadrature and truncating the integral at 8000. The approach is similar to [TS06], except they use a Gauss-Legendre quadrature instead. These routines are implemented with relative ease and have shown reasonable accuracy in our applications. Unfortunately, the combined numerical approximation of (9.12) becomes computationally heavy. One potential solution is the cumulant expansion approach as suggested in [CDG03]. The idea behind this approach is to approximate the characteristic function (which in our case would be (9.9)) using information about the distribution provided by a few moments/cumulants.

The semi analytical price formula is compared below to a Monte Carlo estimate for a caplet with fixed tenor $\delta = 3M$, varying strike and maturity. Generally, there is a close match between the Monte Carlo estimations and the analytical prices with some discrepancy deep in-the-money for longer maturities.

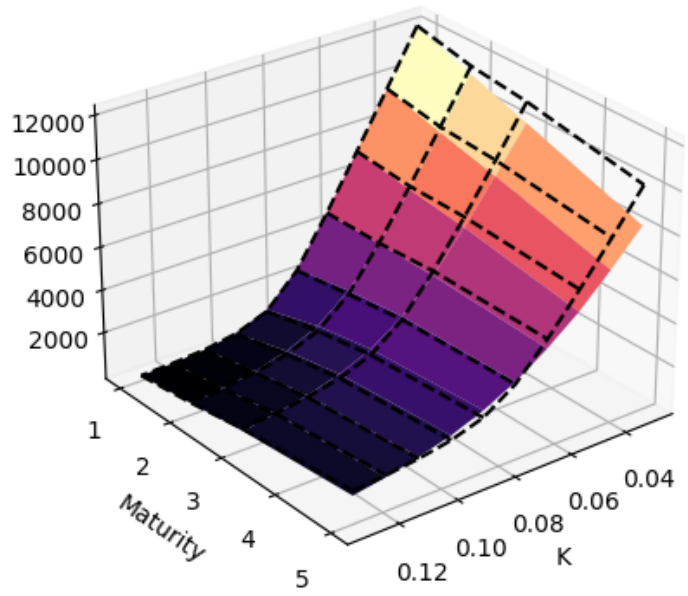


Figure 30: Price difference for Monte Carlo estimation (coloured surface) against the analytical price (9.12) (black wireframe) for a caplet with $3M$ tenor, varying strike K and time to maturity T. Factor dimension $N = 1$ with parameter setting given by table 3. Monte Carlo estimate with $51,200$ paths and 100 discretization steps per maturity year.

Having introduced the TS-MFSV model, we now turn our attention to examining its compatibility and effectiveness within the differential machine learning framework.

10 Differential ML in Stochastic Volatility Markets

In this section, we consider some of the same experiments presented earlier in section 7 but under the TS-MFSV from [TS06]. We emphasize this is not a simple extension as fitting an estimator with Differential Machine Learning is not an exact science and often requires the utilization of different tricks and extensions in order to achieve satisfactory results. Besides the tuning of hyperparameters and choice of basis- / activation-functions, the extensions and tricks also include topics related to: The domain specific knowledge about the specific products at hand, controlling the asymptotic behavior of the estimator, and taking advantages of different sampling techniques. Some of these concepts are covered in [Sav19] where most are loosely described on a high conceptual level. To be more specific, some of the techniques involve:

- Dimension reduction, e.g. through differential principal component analysis (PCA), or transformation from state variables and curves to the underlying market variables and their volatility processes.

- Asymptotic control through more frequent sampling of extreme scenarios (also related to importance sampling), and extrapolation of the estimator. The former can for instance be implemented by having a "burn-in phase" with higher volatility prior to sampling the training samples from the correct distribution. Likewise, an example of the latter is realized by using that many products (such as most options) are linear in the limit and therefore converge to their intrinsic value - i.e. the payoff and thus also value converges to the same as a corresponding forward without any optionality.

Although, these concepts are fairly simple in principle, they can be difficult to implement in practice. For instance, applying dimension reduction through (differential) PCA removes the possibility of directly extracting the Greeks / hedge coefficients through AAD and back-propagation. Likewise, sampling from a different distribution, even if it is only slightly changed through a burn-in phase with larger volatility, or weighting the training samples more around tricky and important areas such as the strike, often comes with a trade-off where the remaining space become more difficult to fit simultaneously.

When working with a multi-factor stochastic volatility model, we clearly need to incorporate much of the information from many of the state variables in order to achieve good results - especially, the ones related to the volatility are important. In this project, we take the estimation or calibration of the model parameters for granted but note that in practice the algorithm for fitting the parameters can often also be differentiated through AAD. This has several advantages as many popular algorithms used for model fitting requires the derivative of some order[17] for the optimization with respect to model parameters. By further utilizing the implicit function theorem and the chain-rule on the optimization algorithm and the pricing engine, one can obtain sensitives with respect to both model parameters and market variables [SH20], [GLP21]. We leave this as an extension for further implementation of our framework and engine. Instead, we now turn to the problems of obtaining estimators for the valuation function and their sensitivities with respect to the underlying and volatility.

10.1 Pricing and Sensitivities

The first objective is to fit an estimator that is capable of replicating the pricing function. We therefore start by considering the simplest case with $N = 1$ dimension. That is, we have one Wiener process for the state variable x and another, correlated one, for ν. Our model specification uses the parameters listed in table 3 except when stated otherwise.

To get some dispersion, we perform a "burn-in" pre-simulation simulation from time $t_{-1} = 0.0$ to $t_0 = 1.0$ with higher volatility. From here, we reinitialize the model in the obtained states and sample our pathwise training samples. To get the test samples, we do the same procedure but without increasing the volatility. The result is shown in figure 31 below where the sample space of the underlying asset and the volatility process is shown for an arbitrary product.

[17] Often the Jacobian or Hessian.

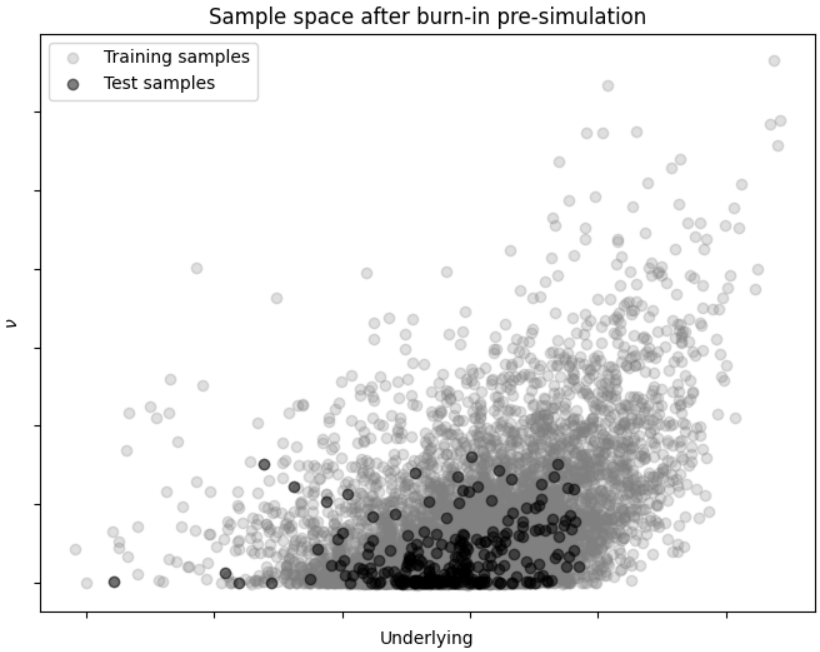


Figure 31: Sample space after burn-in pre-simulation from time t_{-1} to t_0.

Intuitively, this sampling technique is generally promising but there may be some issues regarding the few test samples that are scattered around the lower left corner. We could possibly detect these cases by using statistical measures such as the bi-variate Gaussian maximum log-likelihood or an algorithm for our outlier detection such as k Nearest Neighbour clustering or isolation forest. For the detected extreme test cases [SH20] suggest a couple of methods, including running a full Monte Carlo simulation with AAD for evaluating both the price and Greeks or relying on product specific knowledge such as the cases where value of the product is close its intrinsic value. In this project, we do not implement such techniques in order to properly demonstrate the capabilities and limitations of differential machine learning.

As the state variables are generally difficult to interpret the model, we show test states of the model captured by the Instantaneous forward and zero coupon bond curves along with the density of the Instantaneous risk free short rate in figure 32 below.

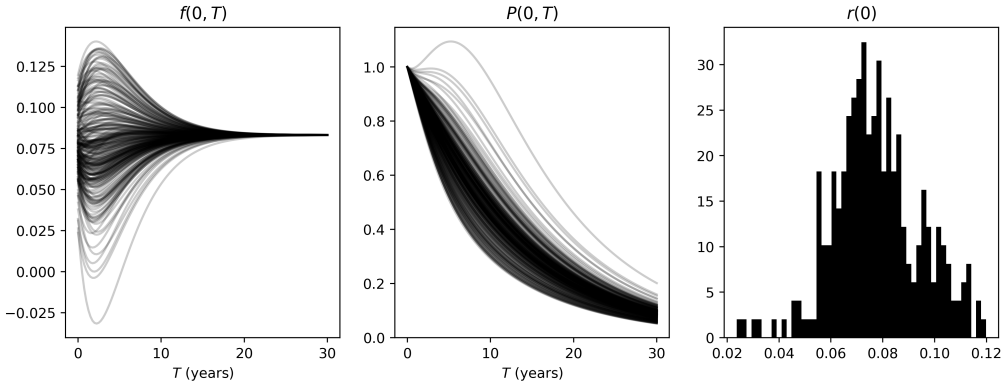


Figure 32: Summary of states for test cases after burn-in pre-simulations. Left: Instantaneous forward curves. Mid: Zero coupon bond curves. Right: Density of instantaneous risk free short rate.

In the following, we present the results of using Differential Regression and Differential Neural Networks for estimating the value and Greeks for some of the more common products under the stochastic volatility model. The resulting prices are compared to using full Monte Carlo simulations with 50,000 paths and antithetic variates.

10.1.1 AAD to Generate Training Data

When using a multi-factor model there is often different choices for how to represent the current state. It can be the state variables directly, which in the TS-MFSV model could be the entire vector/matrix of $(x(t), \nu(t), \phi_1(t), \phi_2(t), ..., \phi_6(t))^\top$. As we argued above, this information could also be filtered by running a (differential) PCA analysis. It could also be argued, that since the filtration is generated purely by the Brownian motions entering $x_i(t)$ and $\nu_i(t)$ these could be sufficient the describe the current state. How we represent the current state is important for how we build the Differential Machine Learning training data.

For a general option that has underlying with value $U(t)$ at time-t we choose the pair $(U(t), \nu(t)_1, ..., \nu_N(t))$ to form the set of training input features $\boldsymbol{X}$. We have to include $\nu(t)$ in this set, since the stochastic volatility is unspanned by the yield curve (which the underlying most likely is derived from) and thus make sure, we have a full picture of the state of the model.

We let the set of training labels $\boldsymbol{y}$ be the price of the interest rate product we are interested in. The set of differential labels $\boldsymbol{Z}$ is then given by the price differentials with respect to the underlying and the variance process. We have a solid setup for learning from these

training sets, the question now becomes to efficiently generating them. For this purpose we rely on our AAD machinery from 5 to generate fast and accurate differentials. This process is best visualized in the flow chart 7 given in section 6.1.

10.1.2 Caplet

We estimate price, *"delta"* and *"vega"* using respectively Differential Regression and Differential Neural Network and compare the two methods against each other. In this context we define *"delta"* as the option price's sensitivity with respect to the underlying, i.e. the zero-coupon bond and *"vega"* as the option price's sensitivity to the variance process $\nu(t)$ as a way to measure how the option price reacts to changes in volatility. A shift in $\nu(t)$ affects mainly short-term maturities and *"vega"* can in this case not be directly thought of, as in e.g. Vasicek or Black-Scholes where we have constant volatility (both in calendar and maturity time). If we were to conduct a full investigation of how prices reacts to fundamental shifts in the whole volatility structure, we would also have to consider the sensitivity with respect to the long term mean, θ. This is considered e.g. in [Zhu10], but our scope for below will just be the instantaneous variance. This sensitivity is relevant in a hedging context if we wish to immunize a portfolio against movements in the variance process. This is expanded on in below section 10.2.

We consider a Caplet with notional $N = 1,000,000$ that has 1 year to maturity with 3 months tenor and strike rate of 7%. First we report the price estimate in figure 33 for the differential regression method against a Monte Carlo estimation with 50,000 paths. We observe a general agreement between the two methods with an RMSE score of approximately 787 over a test range of moneyness and conclude an acceptable fit. A comparison is brought in figure 34 that shows a slightly lower RMSE score of approximately 725 across the same test range. From looking at the price estimation performance only, it seems the two models are performing equally well.

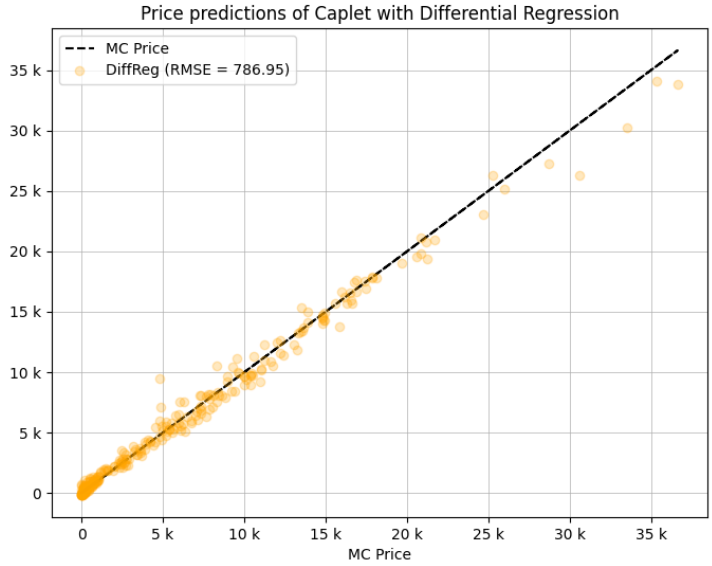


Figure 33: Comparison of price predictions for a 1Y3M Caplet with notional $N = 1,000,000$ and strike $K = 7\%$ using Differential Regression and Monte Carlo price estimates from $50,000$ paths. The Differential Regressor is trained on 8,192 training samples with 9 degrees including interaction terms, and using $\alpha = 1.0$ for regularization.

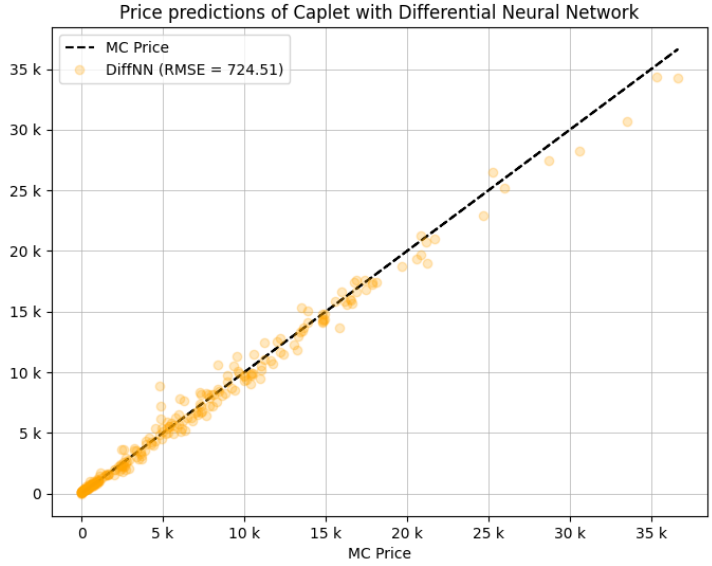
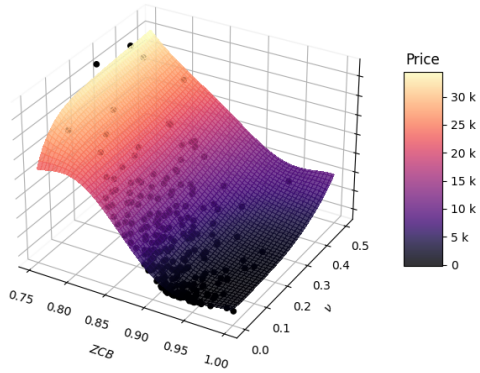


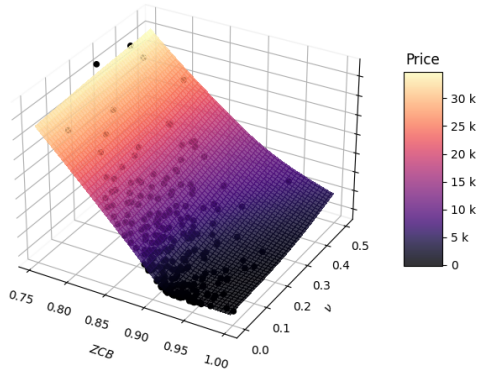
Figure 34: Comparison of price predictions for a 1Y3M Caplet with notional $N = 1,000,000$ and strike $K = 7\%$ using Differential Regression and Monte Carlo price estimates from $50,000$ paths. The exercise date is $1Y$. The Differential Neural Network has 4 hidden layers, each with 20 nodes, is trained on 8,192 training samples with 250 epochs. The activation functions used are softplus and sigmoid. For regularization we use $\lambda = 1.0$.

One thing is an accurate price point estimate, another thing is, how well-behaved and sta-

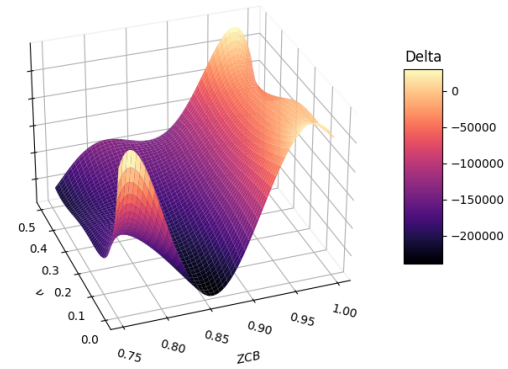
ble is our estimate across a range of moneyness and also values of instantaneous variance $\nu(t)$. In figure 35a and 35b shows a comparison between a price surface using a differential regression method versus a neural network estimation. Here it is clearly visible that the neural network estimation is more well-behaved in the edge cases deep-in-the-money and for high volatility areas where the sample space is more sparse. A similar conclusion can be made from figures 35c and 35d, where the *"delta"* estimate is shown. The regression model has again difficulties estimating the surface in the same areas. This is tendency is emphasized even more in the lower pane in figure 35e and 35f where the *"vega"* estimates are blowing up in the edge cases which is not the case for the neural network method. This method seems capable of producing a much more stable estimation surface, both when it comes to prices and also (selected) sensitivities.



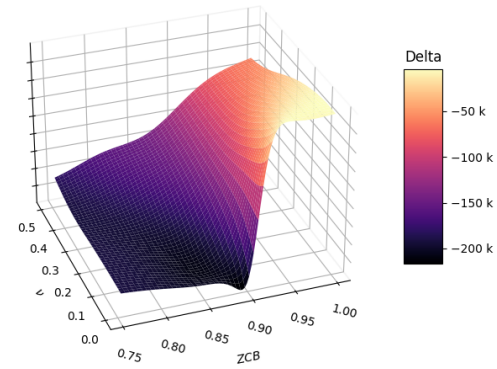
(a) Price prediction with Differential Regression



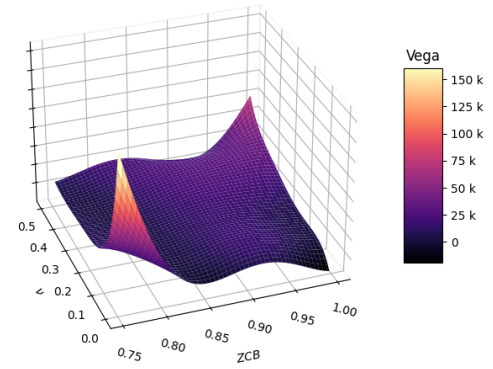
(b) Price prediction with Differential Neural Network



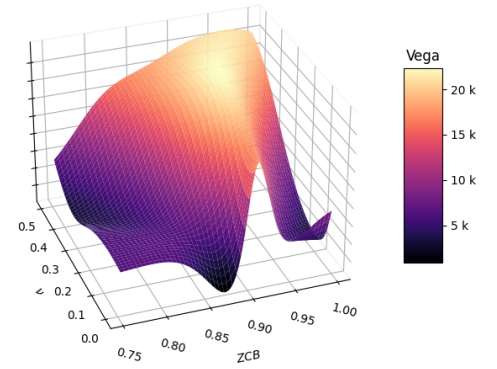
(c) Delta prediction with Differential Regression



(d) Delta prediction with Differential Neural Network



(e) Vega prediction with Differential Regression



(f) Price prediction with Differential Neural Network

Figure 35: Price (top), delta (mid), and vega (bottom) predictions with Differential Regression (left) and Differential Neural Network (right) of 1Y3M Caplet with notional $N = 1,000,000$ with strike $K = 7\%$. The Differential Regressor is trained on 8,192 training samples with 9 degrees and interaction terms. The Differential Neural Network which has 4 hidden layers, each with 20 nodes, is trained on 8,192 training samples with 250 epochs. The activation functions used are softplus and sigmoid. For regularization we use $\alpha = \lambda = 1.0$.

10.1.3 Swaption

We perform a similar investigation by performing the experiment on a European payer swaption that expires in 1 year with the right to enter into the underlying payer swap, which has a notional of $N = 1,000,000$, a fixing rate of 8.41%, and a maturity 5 years later with quarterly fixings. At first sight, the Differential Regressor outperforms the Differential Neural Network in teams of estimating the price as seen in figure 36 and 37 below where the former has an RMSE of 4,978 while the latter has an RMSE of 5,609. However, had we instead used fewer basis function - either by not including interactions or choosing a lower degree for the polynomial - then the Neural Network would perform better according to our experiments. Our experiments also did not encourage the change of any of the hyper-parameters or the activation functions used for the Differential Neural Network.

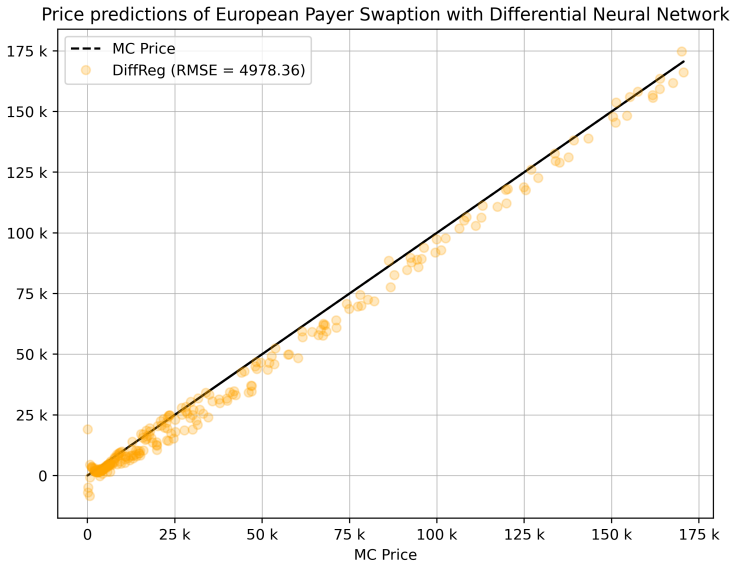


Figure 36: Comparison of price predictions for a European Payer Swaption using Differential Regression and Monte Carlo price estimates from 50,000 paths. The exercise date is 1Y. The underlying payer swap has a notional of $N = 1,000,000$, a fixing rate at 8.41 pct., quarterly fixings, $\delta = 0.25$, with the first fixing date being at the exercise date and the last being 5 years later. The Differential Regressor is trained on 1,024 training samples with 5 degrees, interactions, and using $\alpha = 1.0$ for regularization.

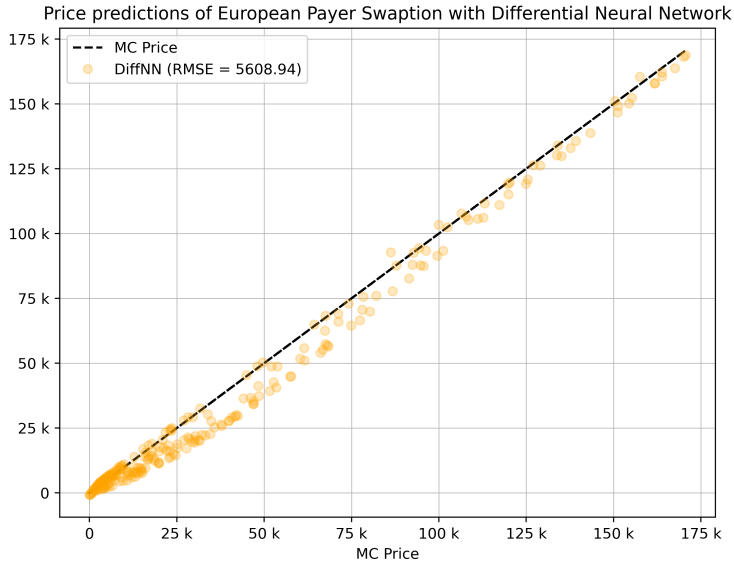
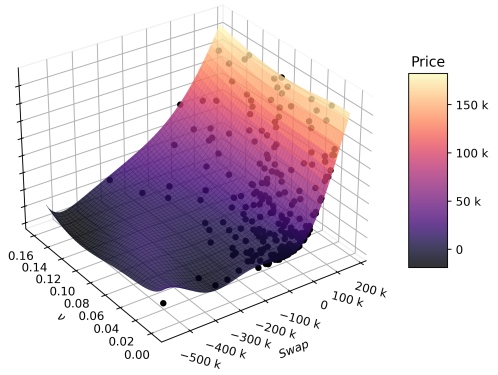
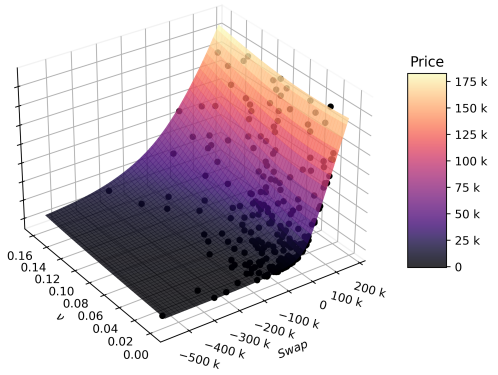


Figure 37: Comparison of price predictions for a European Payer Swaption using Differential Neural Network and Monte Carlo price estimates from $50,000$ *paths. The exercise date is* $1Y$*. The underlying payer swap has a notional of* $N = 1,000,000$*, a fixing rate at* 8.41 *pct., quarterly fixings,* $\delta = 0.25$*, with the first fixing date being at the exercise date and the last being 5 years later. The Differential Neural Network has 4 hidden layers, each with 20 nodes, is trained on 8,192 training samples with 250 epochs. The activation functions used are softplus and sigmoid. For regularization we use* $\lambda = 1.0$*.*

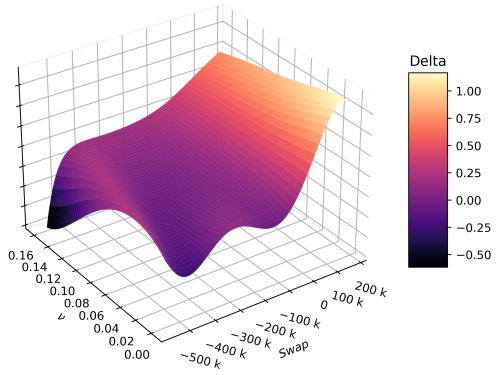
Ideally, we are not only interest in the pricing function to be accurate for specific test cases but also for the pricing function to be well behaved and show great stability. Furthermore, once we have sold the derivative, we would also need the Greeks to determine the hedge coefficients needed to replicate the payoff by dynamically trading in the risk-free asset, the underlying, and another European payer swaption with the same maturity and a different strike. In the sub-figures shown below in figure 38, we compare performance of the Differential Regression and the Differential Neural Network by showing their predictions for not only the pricing function but also the sensitivities of the price wrt. the underlying and the volatility state variable. This is done for a grid of combinations of the underlying swap spot price and the current level of the volatility state variable. The grid of combinations is a superset of the test dataset and thus we also show prices obtained by performing full Monte Carlo simulation on the test cases in the top sub-figures.



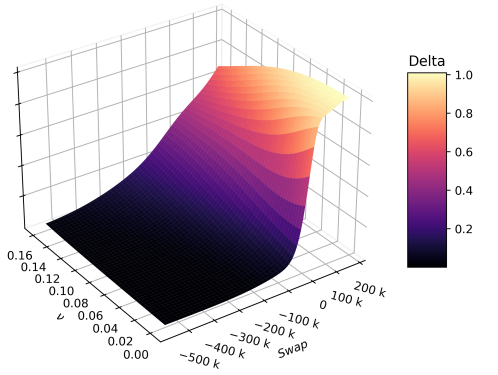
(a) Price prediction with Differential Regression



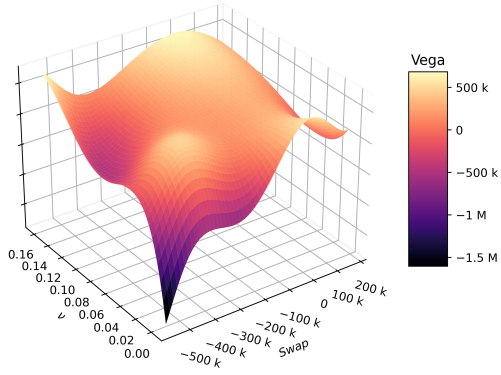
(b) Price prediction with Differential Neural Network



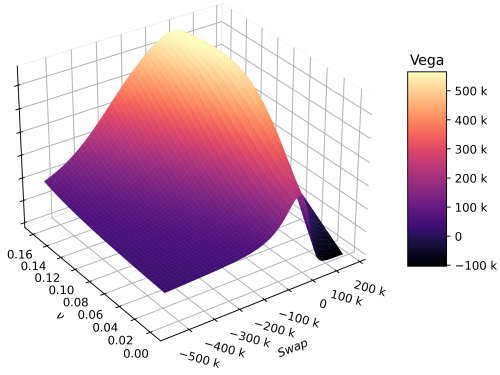
(c) Delta prediction with Differential Regression



(d) Delta prediction with Differential Neural Network



(e) Vega prediction with Differential Regression



(f) Vega prediction with Differential Neural Network

Figure 38: Price (top), delta (mid), and vega (bottom) predictions with Differential Regression (left) and Differential Neural Network (right) of European Payer Swaption. The exercise date is $1Y$. The underlying payer swap has a notional of $N = 1,000,000$, a fixing rate at 8.41%, quarterly fixings, $\delta = 0.25$, with the first fixing date being at the exercise date and the last being 5 years later. The Differential Regression has 9 degrees with interactions between the lower monomials. The Differential Neural Network has 4 hidden layers, each with 20 nodes, and is trained on 250 epochs. The activation functions used are softplus and sigmoid. Both estimators are trained on 8,192 training samples. For regularization we use $\alpha = \lambda = 1.0$.

Comparing figure 38a and 38b, we see that the overall prediction of the prices are almost identical between the two estimators. However, the Differential Neural Network is a bit more well behaved for the cases where the volatility state variable and the spot price of swap are both low. This phenomenon is seen even more clearly when comparing figure 38c and 38d, where the former is not capable of fitting the entire delta curve in a way that we would expect. More specifically, we note that as opposed to the Differential Neural Network, the Differential Regressor predicts negative delta for a few areas and also shows a "hump" shape. Finally, we can compare figure 38e and 38f where the Differential regressor behaves relatively irregularly. In some of the edge-cases it even predicts "vegas" that are very negative while also having several unexpected local maxima. On the contrary, the Differential Neural Network is able of predict "vegas" that have a clear structure as a function of the spot price of the underlying swap and the instantaneous volatility. This shape - despite having a few areas with negative "vega" - coincides with the general established theory for option Greeks.

10.2 Dynamically Hedging

After pricing a financial product and hopefully selling it to a counterpart, we hope to be able to hedge the risk from being short this asset. As mentioned earlier, we might be able to buy an identical product elsewhere from a reasonable price or make a portfolio that replicates the payoff function. However, if the market is not providing us with such possibilities then we would once again be faced with the problem of constructing a self-financing portfolio which we can dynamically rebalance in order to mitigate any risk from the market movements. As opposed to the experiments shown in section 7 where we considered a market governed by the Vasicek model and therefore was able to construct such a self-financing portfolio by delta-hedging, we are now considering a much more realistic model where we also have to make additional considerations for dealing with the risk originating from the stochastic volatility.

More specifically, in the simple setting of the Vasicek model, we showed that by dynamically trading in the underlying asset we can hedge our delta risk originating from selling various types of interest rate derivatives. But in the setting of [TS06]'s model described in the previous section 9, or any other *true* stochastic volatility model, we have

to immunize our trading risks in other ways than purely trading in zero-coupon bonds, since the interest rate derivative will not be spanned entirely by the discount curve. Depending on the number of sources of risk in the model, we have to include additional assets in our hedge to complete the model. Restating the model equations (9.2):

$$\begin{aligned} df(t,T) &= \mu_f(t,T)dt + \sum_{i=1}^{N} \sigma_{f,i}(t,T)\sqrt{\nu_i(t)}dW_i^{\mathbb{Q}}(t), \\ d\nu_i(t) &= \kappa_i(\theta_i - \nu_i(t))dt + \sigma_i\sqrt{\nu_i(t)}\left(\rho_i dW_i^{\mathbb{Q}}(t) + \sqrt{1-\rho_i^2}dZ_i^{\mathbb{Q}}\right), \end{aligned}$$

we emphasize that we have $N \times 2$ risk driving processes, which means we also need $N \times 2$ assets to complete the market following since all other assets can be replicated by linear combinations of these (*spanned*). We wish to show below how a hedge portfolio can be constructed in the presence of stochastic volatility. The procedure is much alike [Gat06]. Let us consider the case where we have sold a single caplet, C at a certain strike K, (although this is not practised in real world markets) and we would have to set up a hedge for this position. If we view the caplet as a put option on the zero-coupon bond and let [TS06] describe the market, we have for the case $N = 1$ by (9.7) a description of the market:

$$\begin{aligned} dP(t,T) &= r(t)P(t,T)dt + B_x(T-t)\sqrt{\nu(t)}P(t,T)dW^{\mathbb{Q}}, \\ d\nu(t) &= \kappa(\theta - \nu(t))dt + \sigma\sqrt{\nu(t)}\left(\rho dW^{\mathbb{Q}}(t) + \sqrt{1-\rho^2}dZ^{\mathbb{Q}}\right), \end{aligned} \tag{10.1}$$

We can then form a riskless portfolio consisting of the option we sold, an amount Δ_1 in the underlying asset $P(t,T)$ and additionally the amount Δ_2 in another another option F on the same underlying. We view the option prices as functions of $t, P(t,T)$ and $\nu(t)$ such that our total holdings is:

$$\Pi(t,P(t,T),\nu(t)) = C(t,P(t,T),\nu(t)) - \Delta_1 P(t,T) - \Delta_2 F(t,P(t,T),\nu(t))$$

We impose a self-financing condition for this strategy such that:

$$d\Pi = dC - \Delta_1 dP - \Delta_2 dF. \tag{10.2}$$

We simplify the notation used in equation (10.1) by introducing the functions

$$\begin{aligned} f(\nu(t)) &:= B_x(T-t)\sqrt{\nu(t)}, \\ a(\nu(t)) &:= \kappa(\theta - \nu(t)), \\ b(\nu(t)) &:= \sigma\sqrt{\nu(t)}, \end{aligned}$$

such that we can write:

$$\begin{aligned} dP(t,T) &= r(t)P(t,T)dt + f(\nu(t))P(t,T)dW^{\mathbb{Q}}, \\ d\nu(t) &= a(\nu(t))dt + b(\nu(t))\left(\rho dW^{\mathbb{Q}}(t) + \sqrt{1-\rho^2}dZ^{\mathbb{Q}}\right). \end{aligned}$$

We apply Ito's lemma to the portfolio in equation (10.2) and use the notation above to obtain the portfolio dynamics:

$$\begin{aligned} d\Pi = &\left(\frac{\partial C}{\partial t} + \frac{1}{2}f(\nu)^2P^2\frac{\partial^2 C}{\partial P^2} + \frac{1}{2}b^2\frac{\partial^2 C}{\partial \nu^2} + \rho f(\nu)Pb\frac{\partial^2 C}{\partial P \partial \nu}\right)dt \\ &+ \frac{\partial C}{\partial P}dP + \frac{\partial C}{\partial \nu}d\nu \\ &- \Delta_1 dP \\ &- \Delta_2\left[\left(\frac{\partial F}{\partial t} + \frac{1}{2}f(\nu)^2P^2\frac{\partial^2 F}{\partial P^2} + \frac{1}{2}b^2\frac{\partial^2 F}{\partial \nu^2} + \rho f(\nu)Pb\frac{\partial^2 F}{\partial P \partial \nu}\right)dt + \frac{\partial F}{\partial P}dP + \frac{\partial F}{\partial \nu}d\nu\right] \\ = &\left(\frac{\partial C}{\partial t} + \frac{1}{2}f(\nu)^2P^2\frac{\partial^2 C}{\partial P^2} + \frac{1}{2}b^2\frac{\partial^2 C}{\partial \nu^2} + \rho f(\nu)Pb\frac{\partial^2 C}{\partial P \partial \nu}\right)dt \\ &- \Delta_2\left(\frac{\partial F}{\partial t} + \frac{1}{2}f(\nu)^2P^2\frac{\partial^2 F}{\partial P^2} + \frac{1}{2}b^2\frac{\partial^2 F}{\partial \nu^2} + \rho f(\nu)Pb\frac{\partial^2 F}{\partial P \partial \nu}\right)dt \\ &+ \left(\frac{\partial C}{\partial P} - \Delta_2\frac{\partial F}{\partial P} - \Delta_1\right)dP \\ &+ \left(\frac{\partial C}{\partial \nu} - \Delta_2\frac{\partial F}{\partial \nu}\right)d\nu \end{aligned}$$

To immunize our portfolio against movements in P and ν, and thus make it riskless, we see that we must eliminate the dP and $d\nu$ terms by ensuring:

$$\begin{cases} \dfrac{\partial C}{\partial P} - \Delta_2\dfrac{\partial F}{\partial P} - \Delta_1 &= 0, \\ \dfrac{\partial C}{\partial \nu} - \Delta_2\dfrac{\partial F}{\partial \nu} &= 0. \end{cases}$$

Solving this system of equations gives us the quantities:

$$\Delta_2 = \frac{\partial C}{\partial v}\left(\frac{\partial F}{\partial v}\right)^{-1}, \tag{10.3}$$

$$\Delta_1 = \frac{\partial C}{\partial P} - \frac{\partial F}{\partial P}\Delta_2. \tag{10.4}$$

These equations gives us a theoretical way of finding the hedging coefficients. However, using the estimators obtained from applying differential machine learning, we are only guaranteed accurate results under expectation. As mentioned earlier, the universal approximation theorem guarantees the existence of an estimator that is capable of replicating the results of the true expectation. However, as we do not have unlimited data or training time available, we must of course find a reasonable approach to find a good estimator. We have chosen to rely on differential regularization in the machine learning algorithms to improve the rate of convergence. As we visualized in figures 35 and 38 we could find generally good and stable estimators, especially using the differential neural network methods, but still with some uncertainty attached. Under the Vasicek model, where we only needed to predict delta, we could still achieve reasonable and satisfactory results for all of the products considered. However, as equation (10.3) and (10.4) requires performing division of sensitivities, we experience in practice this calculation to be unstable which results in poor estimates for the hedge coefficients. This is most likely caused in areas where the sensitivities are close to zero and certain paths "blow up". In a similar hedging experiment as we did for Vasicek, we only have to be off once, for the total hedge PnL to misalign with the true value of the product we were trying to hedge.

11 Conclusion

In this concluding section, we summarize our findings and discuss potential further research upon the results and framework presented in this thesis.

In section 1, 2, and 3 we have outlined well-established theory and results covering a large range of interest rate products and derivatives. These sections also introduced the renowned Vasicek model, which served for carrying out several simulation experiments conducted in the subsequent parts of our study.

In section 4 we have described a general and versatile framework for Monte Carlo simulation which acts as our primary pricing and risk engine throughout the project. Our flexible design choice by segregating models from products has allowed us to interchange all of the engine's components independently of each other. This modular approach enhances the frameworks ability to cover all of the products from section 2 and models with relative ease, while keeping the possibility of adding many more in the future. In this context, we have covered how to price linear products, European options, and path dependent products by linking the classical Risk Neutral Pricing Theorem to the practical implementation of the Monte Carlo engine. Furthermore, we have instrumented the Monte Carlo engine with an efficient implementation of the Least Squares Method, allowing us to model callable products as if they were path-dependent. Finally, we have implemented different techniques for optimizing the computational performance by relying on the Singular Value Decomposition in the regression steps and employing common variance reduction techniques.

Moving to section 5, we have presented the theory for performing Automatic Differentiation for financial problems. In doing so, we have discussed how to handle discontinuous functions through fuzzy logic and smoothing of the payoff functions. We have also discussed the practical aspects of implementing automatic differentiation in large scale setups by utilizing operation overload, tapes, directed acyclic graphs, and more. Our implementation relies on PyTorch which in combination with the Monte Carlo engine enabled us to simulate pathwise samples in the form of discounted payoffs and their sensitivities with respect to market observables and model parameters. These have successfully been used

to train the Differential Machine Learning models in the form of Differential Regression and Differential Neural Networks which are presented theoretically in section 6. These methods rely on additional differential labels as their loss functions are penalized through a differential regularization.

In section 7, we have demonstrated the effectiveness of our implementations to provide fast and accurate estimators of not only the pricing function, but also the sensitivities across various products. We found in the setting of the Vasicek model, the accuracy of the Differential Regression almost matches Differential Neural Network for simpler products. A notable advantage of Differential Regression is its significantly lower computational demand. However regression falls short when the complexity of products increases, requiring the need for Differential Neural Networks that efficiently calculates price and risk factors, although they do so with greater computational requirement.

Further, in section 8 we introduced the broad HJM Framework for which we later consider the General Multi-Factor Stochastic Volatility Model by Trolle and Schwartz covered in our section 9. The model, which we have implemented in our framework, is known for its generality in many different areas of interest rate and term structure modelling while maintaining many neat properties such as semi-analytical solutions of European products and being Markovian. Thus, it is a remarkable candidate for scaling our experiments to something that closer resembles actual market conditions.

Finally, we have carried out our experiments with the more advanced model in section 10, where our results have shown that both the Differential Regression and the Differential Neural Networks are capable of fitting estimators that closely replicates the pricing function. On the contrary, when it comes to estimating Greeks, we find the Differential Neural Network model produces the most accurate and stable estimates.

11.1 Extensions and Further Research

We showed in section 6 how market sensitivities can be obtained from parameter sensitivities. A way to enhance this computation is by extending our setup to include information obtained from running a calibration procedure. From a gradient based optimization

method, it is possible to extract sensitivities of the model parameters with respect to market observables. As such, when the model is optimized, we can store this sensitivity (vector) giving us half of the computations in the visualized graph 7. This idea is e.g. covered in [GLP21].

Another extension, that could fit well into our framework from section 9 is considering Differential Principal Component Analysis (PCA) which is covered in [SH20] and [HS21]. This method performs dimension reduction by transforming state variables to a reduced state space. As opposed to classical PCA, which only takes the state and/or market variables as its input, Differential PCA also includes information from the pathwise differential labels in the form of discounted payoffs' sensitivities wrt. the input arguments.

Furthermore, [SH20] also covers *Advanced Asymptotic Control* to identify edge-case samples in the training set based on extreme inputs. As we have discussed earlier, there are several methods for handling the edge cases, including - but not limited to - assigning them greater weights in the loss-function (i.e. treating them as soft constraints), relying on domain specific knowledge about the concrete product at hand, or performing a large or full Monte Carlo valuation.

References

[And01] J. Andreasen. Turbo charging the cheyette model. *Bank of America*, 2001.

[And06] J. Andreasen. Stochastic volatility for real. *Bank of America*, 2006.

[AP10a] L. Andersen and V. Piterbarg. *Interest Rate Modelling*, volume I. Atlantic Financial Press, 2010.

[AP10b] L. Andersen and V. Piterbarg. *Interest Rate Modelling*, volume II. Atlantic Financial Press, 2010.

[BG96] M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management Science*, 42:269 – 285, 1996.

[BM06] D. Brigo and F. Mercurio. *Interest Rate Models - Theory and Practice With Smile, Inflation and Credit*. Springer, 2 edition, 2006.

[Cap11] L. Capriotti. Fast greeks by algorithmic differentiation. *The Journal of Computational Finance*, 14:3–35, 2011.

[CCDG05] J. Casassus, P. Collin-Dufresne, and B. Goldstein. Unspanned stochastic volatility and fixed income derivatives pricing. *Journal of Banking and Finance 29: 2723 - 2749*, 2005.

[CDG03] P. Collin-Dufresne and B. Goldstein. Generalizing the affine framework to hjm and random field models. *Working paper, U.C. Berkeley.*, 2003.

[Che92] O. Cheyette. Markov representation of the heath-jarrow-morton model. 1992.

[CK01] C. Chiarella and O. Kwon. Classes of interest rate models under the hjm framework. *Asia-Pacifc Financial Markets 8: 1-22*, 2001.

[CL14] Nan Chen and Yanchu Liu. American option sensitivities estimation via a generalized infinitesimal pertubation analysis approach. *Operations Research*, 2014.

[Dup19] B. Dupire. Functional itô calculus. *Quantitative Finance*, 2019.

[Fil09] D. Filipovic. *Term-Structure Models A Graduate Course*. Springer-Verlag Berlin Heidelberg, 1 edition, 2009.

[Gat06] J. Gatheral. *The Volatility Surface: a Practitioner's Guide*. WILEY, 2006.

[GG06] M. B. Giles and P. Glasserman. Smoking adjoints: fast evaluation of greeks in monte carlo calculations. 2006.

[Gla03] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2003.

[GLP21] D. Goloubentsev, E. Lakshtanov, and V. Piterbarg. Automatic implicit function theorem. *SSRN*, 2021.

[Hes93] S. Heston. A closed form solution for options with stochastic volatility. *Review of Financial Studies*, 6:327–343, 1993.

[HHA23] N. Hansen, S. Hansen, and A. Axel. American delta force. *University of Copenhagen*, 2023.

[HS17] B. N. Huge and A. Savine. Lsm reloaded - differentiate xva on your ipad mini. *SSRN*, 2017.

[HS21] B. Huge and A. Savine. Axes that matter: Pca with a difference. *Risk Magazine*, 2021.

[Lin13] M. Linderstrøm. *Fixed Income Derivatives Lecture Notes*. 3 edition, 2013.

[LS01] F. Longstaff and E. Schwartz. Valuing american options by simulation: A simple least-squares approach, the review of financial studies. 2001.

[Mun99] C. Munk. Stochastic duration and fast coupon bond option pricing in multi-factor models,. *Review of Derivatives Research*, 3:157–181, 1999.

[Pit03] V. Pitervarg. Computing deltas of callable libor exotics in forward libor models. 2003.

[RS91] Litterman R. and J. Scheinkman. Common factors affecting bond returns. *Journal of Fixed Income (June)*, 1991.

[Sav16] A. Savine. Fuzzy logic for financial derivatives. Global Derivatives, 2016.

[Sav19] A. Savine. *Modern Computational Finance - AAD and Parallel Simulation*. WILEY, 2019.

[Sey17] R. Seydel. *Tools for computational Finance*. Springer, 6 edition, 2017.

[SH20] A. Savine and B. Huge. Differential machine learning. 2020.

[TS06] A. Trolle and E. Schwartz. A general stochastic volatility model for the pricing and forecasting of interest rate derivatives. *National Bureau of Economic Research*, 2006.

[Zhu10] J. Zhu. *Applications of Fourier Transform to Smile Modeling - Theory and Implemention*. Springer, 2 edition, 2010.

A Vasicek ATS - Solution to the Ricatti equations

To solve the Ricatti equations 3.5 and 3.6, we consider the derivative

$$\begin{aligned}\frac{\partial}{\partial t}\left[B(t,T)e^{-at}\right] &= \partial_t B(t,T)e^{-at} - aB(t,T)e^{-at} \\ &= \left(aB(t,T) - 1\right)e^{-at} - aB(t,T)e^{-at} \\ &= -e^{-at},\end{aligned}$$

where equation (3.6) was applied in the second equality. As a result of integration and utilizing the terminal condition, we get

$$\begin{aligned}\int_t^T \frac{\partial}{\partial s}\left[B(s,T)e^{-as}\right]ds &= \int_t^T e^{-as}ds \\ \Leftrightarrow \quad -B(t,T)e^{-at} &= -\frac{1}{a}\left[e^{-aT} - e^{-at}\right] \\ \Leftrightarrow \quad B(t,T) &= \frac{1-e^{-a(T-t)}}{a}.\end{aligned}$$

For deriving A, we note that $A(t,T) = A(T,T) - \int_t^T \partial_s A(s,T)ds$. Hence it is straightforward by inserting B in (3.5) and working out the relatively simple algebra

$$
\begin{aligned}
A(t,T) &= A(T,T) - \int_t^T \partial_s A(s,T)ds \\
&= ab\int_t^T B(s,T)ds + \frac{1}{2}\sigma^2 \int_t^T B^2(s,T)ds \\
&= b\int_t^T \left(1 - e^{-a(T-s)}\right) - \frac{\sigma^2}{2a^2}\int_t^T \left(1 + e^{-2a(T-s)} - 2e^{-a(T-s)}\right) ds \\
&= b\,(T-t) - b\left(\frac{1-e^{-a(T-t)}}{a}\right) - \frac{\sigma^2}{2a^2}\left[(T-t) + \left(\frac{1-e^{-2a(T-t)}}{2a}\right) - 2\left(\frac{1-e^{-a(T-t)}}{a}\right)\right] \\
&= b\left[(T-t) - B(t,T)\right] - \frac{\sigma^2}{4a^3}\left[2a(T-t) + 1 - e^{-2a(T-t)} - 4 + 4e^{-a(T-t)}\right] \\
&= b\left[(T-t) - B(t,T)\right] - \frac{\sigma^2}{4a^3}\left[2a(T-t) - \left(1 + e^{-2a(T-t)} - 2e^{-a(T-t)}\right) - 2 + 2e^{-a(T-t)}\right] \\
&= b\left[(T-t) - B(t,T)\right] - \frac{\sigma^2}{2a^2}(T-t) - \frac{\sigma^2}{4a^3}\left(-2 + 2e^{-a(T-t)}\right) + \frac{\sigma^2}{4a}\left(\frac{1 + e^{-2a(T-t)} - 2e^{-a(T-t)}}{a^2}\right) \\
&= b\left[(T-t) - B(t,T)\right] - \frac{\sigma^2}{2a^2}(T-t) + \frac{\sigma^2}{2a^2}\left(\frac{1-e^{-a(T-t)}}{a}\right) + \frac{\sigma^2}{4a}\left(\frac{1-e^{-a(T-t)}}{a}\right)^2 \\
&= b\left[(T-t) - B(t,T)\right] - \frac{\sigma^2}{2a^2}\left[(T-t) - B(t,T)\right] + \frac{\sigma^2}{4a}B^2(t,T) \\
&= \left(b - \frac{\sigma^2}{2a^2}\right)\left[(T-t) - B(t,T)\right] + \frac{\sigma^2}{4a}B^2(t,T).
\end{aligned}
$$

B Hedge Experiments

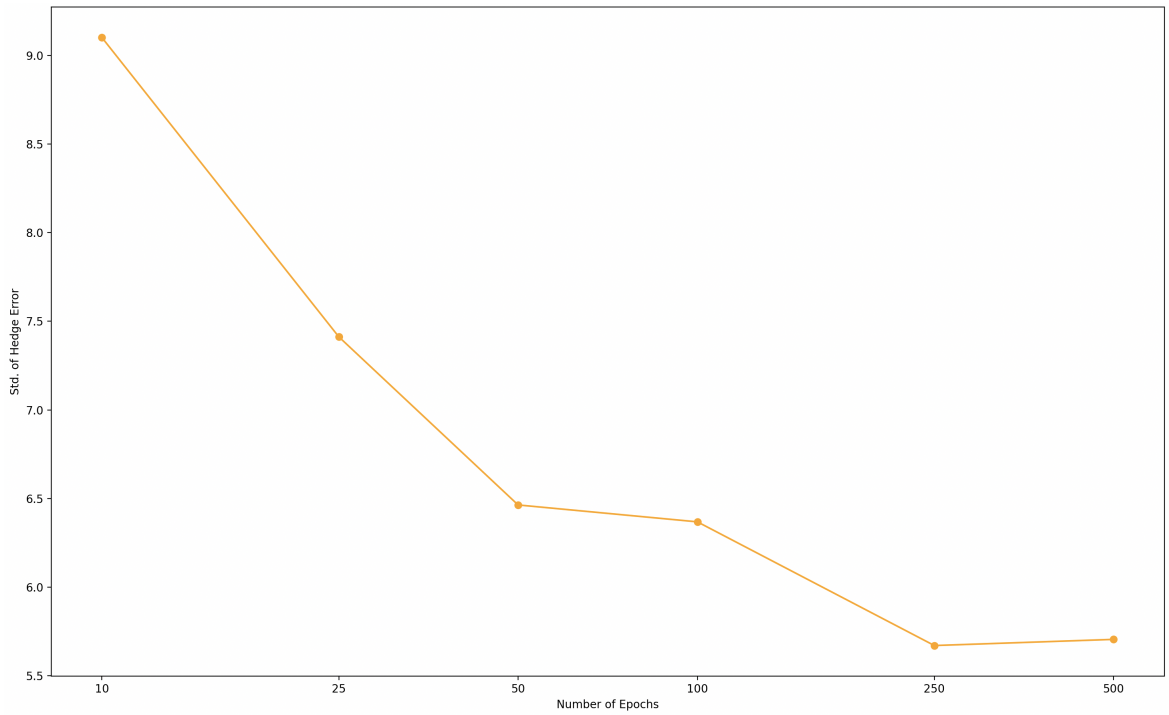


Figure 39: Convergence plot of hedge error for a 1Y6Y3M European payer swaption for differential neural network on log-log scale when varying number of epochs. Simulated from single spot value with strike rate $K = 0.0871$*. The network specifications are; training samples* $N = 1024$*, batch size,* $bsz = \{12.5\%, 25\%, 50\%, 100\%\}$ *of* N *and hedge frequency is every second day a year (*125*).*

C HJM proofs

Lemma C.1. For some given maturity T, the ZCB price follows a Itô dynamics of the form

$$P(t,T) = P(0,T) + \int_0^t P(s,T)\left[r(s) + b(s,T)\right]ds + \int_0^t P(s,T)\nu(s,T)dW(s) \quad \forall t \leq T, \tag{C.1}$$

where the bond volatility is

$$\nu(s,T) = -\int_s^T \sigma(s,u)du$$

and

$$b(s,T) = -\int_s^T \mu(s,u)du + \frac{1}{2}||\nu(s,T)||^2.$$

Proof. Utilizing (8.2) and Fubini for stochastic integrals as shown in [Fil09]

$$\begin{aligned}
\log P(t,T) &= -\int_t^T f(t,u)du \\
&\overset{(8.1)}{=} -\int_t^T f(0,u)du - \int_t^T\int_0^t \mu(s,u)dsdu - \int_t^T\int_0^t \sigma(s,u)dW(s)du \\
&\overset{Fub.}{=} -\int_t^T f(0,u)du - \int_0^t\int_t^T \mu(s,u)duds - \int_0^t\int_t^T \sigma(s,u)dudW(s) \\
&= -\int_0^T f(0,u)du - \int_0^t\int_s^T \mu(s,u)duds - \int_0^t\int_s^T \sigma(s,u)dudW(s) \\
&\quad + \int_0^t f(0,u)du + \int_0^t\int_s^t \mu(s,u)duds + \int_0^t\int_s^t \sigma(s,u)dudW(s) \\
&\overset{Fub.}{=} -\int_0^T f(0,u)du + \int_0^t \left(b(s,T) - \frac{1}{2}||\nu(s,T)||^2\right)ds + \int_0^t \nu(s,T)dW(s) \\
&\quad + \int_0^t f(0,u)du + \int_0^t\int_0^u \mu(s,u)dsdu + \int_0^t\int_0^u \sigma(s,u)dW(s)du \\
&= -\int_0^T f(0,u)du + \int_0^t \left(b(s,T) - \frac{1}{2}||\nu(s,T)||^2\right)ds + \int_0^t \nu(s,T)dW(s) \\
&\quad + \int_0^t \underbrace{\left(f(0,u) + \int_0^u \mu(s,u)ds + \int_0^u \sigma(s,u)dW(s)\right)}_{r(u)}du \\
&\overset{(8.2)}{=} \log P(0,T) + \int_0^t \left(r(s) + b(s,T) - \frac{1}{2}||\nu(s,T)||^2\right)ds + \int_0^t \nu(s,T)dW(s).
\end{aligned}$$

Or on dynamic form

$$d\left(\log P(t,T)\right) = \left(r(t) + b(t,T) - \frac{1}{2}||\nu(t,T)||^2\right)dt + \nu(t,T)dW(t). \tag{C.2}$$

By considering the ZCB as $P(t,T) = e^{\log P(t,T)}$ we can obtain the dynamic equation by applying Itô

$$\begin{aligned}
d\left(P(t,T)\right) =& d\left(e^{\log P(t,T)}\right) \\
=& e^{\log P(t,T)} d\left(\log P(t,T)\right) + \frac{1}{2} e^{\log P(t,T)} \left(d \log P(t,T)\right)^2 \\
=& P(t,T)\left[\left(r(t) + b(t,T) - \frac{1}{2}||\nu(t,T)||^2\right)dt + \nu(t,T)dW(t)\right] + \frac{1}{2}P(t,T)||\nu(t,T)||^2 dt \\
=& P(t,T)\left(r(t) + b(t,T)\right)dt + P(t,T)\nu(t,T)dW(t).
\end{aligned} \tag{C.3}$$

By rewriting on integral form, (8.4) follows.

□

Theorem C.1 (Drift Condition)**.** $\mathbb{Q}$ is an equivalent local martingale measure (ELMM) if and only if the T-bond drift

$$b(t,T) = -\nu(t,T)\gamma(t)^\top \ \forall t \geq T,$$

where ν is the T-bond volatility. If so, (8.1) becomes

$$f(t,T) = f(0,T) + \int_0^t \left(\sigma(s,T)\int_s^T \sigma(s,u)^\top du\right)ds + \int_0^t \sigma(s,T)dW^Q(s),$$

and the discounted T-bond price satisfies

$$\frac{P(t,T)}{B(t)} = P(0,T)\mathcal{E}_t\left(\nu(\cdot,T) \bullet W^Q\right), \ \forall t \leq T, \tag{C.4}$$

where

$$\mathcal{E}_t\left(\nu(\cdot,T) \bullet W^Q\right) = e^{\nu(\cdot,T)\bullet W^Q - \frac{1}{2}\langle \nu(\cdot,T)\bullet W^Q,\, \nu(\cdot,T)\bullet W^Q\rangle_t} = e^{\int_0^t \nu(s,T)dW^{\mathbb{Q}}(s) - \frac{1}{2}\int_0^t ||\nu(s,T)||^2 ds}$$

is the stochastic exponential of the Itô process.

Proof. Switching from $\mathbb{P}$ to the risk-neutral measure $\mathbb{Q}$, the dynamics of the discounted

price process are

$$\frac{d\left(P(t,T)/B(t)\right)}{P(t,T)/B(t)} = b(t,T)dt + \nu(t,T)dW(t) = \left[b(t,T) + \nu(t,T)\gamma(t)^\top\right]dt + \nu(t,T)dW^{\mathbb{Q}}. \tag{C.5}$$

Clearly the process is driftless and thus a martingale when $b(t,T) = -\nu(t,T)\gamma(t)^\top$. Consequently the relation hold

$$-\int_s^T \mu(s,u)du + \frac{1}{2}||\nu(s,T)||^2 = b(s,T) = -\nu(t,T)\gamma(t)^\top.$$

Solving for the drift yields

$$\begin{aligned}
\mu(t,T) &= \left[\partial_T \nu(t,T)\right]\nu(t,T)^\top - \partial_T \nu(t,T)\gamma(t)^\top \\
&= (-\sigma(t,T))\left(-\int_s^T \sigma(s,u)du\right) - (-sigma(t,T))\,\gamma(t)^\top \\
&= \sigma(t,T)\int_t^T \sigma(t,u)du - \sigma(t,T)\gamma(t)^\top.
\end{aligned}$$

By inserting this expression of the drift into (8.1) and change measure, the instantaneous forward rate becomes

$$\begin{aligned}
f(t,T) &= f(0,t) + \int_0^t \left(\sigma(s,T)\int_s^T \sigma(s,u)^\top du - \sigma(s,T)\gamma(s)\right)ds + \int_0^t \sigma(0,T)dW(s) \\
&= f(0,t) + \int_0^t \left(\sigma(s,T)\int_s^T \sigma(s,u)^\top du\right)ds + \int_0^t \sigma(s,T)dW^{\mathbb{Q}}(s),
\end{aligned}$$

as desired. To establish (8.7), we restrict the drift to be of the required form, thus (C.5) becomes

$$d\left(\frac{P(t,T)}{B(t)}\right) = \frac{P(t,T)}{B(t)}\nu(t,T)dW^{\mathbb{Q}}(t) = \frac{P(t,T)}{B(t)}d\left(\nu(\cdot,T)\bullet W^{\mathbb{Q}}\right).$$

Applying this together with Itô on the log-transformation yields

$$\begin{aligned}
d\left(\log\frac{P(t,T)}{B(t)}\right) &= \frac{1}{P(t,T)/B(t)}d\left(\frac{P(t,T)}{B(t)}\right) - \frac{1}{2}\frac{1}{\left(P(t,T)/B(t)\right)^2}\left(d\left(\frac{P(t,T)}{B(t)}\right)\right)^2 \\
&= \nu(t,T)dW^{\mathbb{Q}}(t) - \frac{1}{2}||\nu(t,T)||^2 dt.
\end{aligned}$$

Writing on integral form and isolate proves the theorem

$$\begin{aligned}
\log \frac{P(t,T)}{B(t)} &= \log \frac{P(0,T)}{B(0)} + \int_0^t \nu(s,T) dW^{\mathbb{Q}}(s) - \frac{1}{2}\int_0^t ||\nu(t,T)||^2 ds \\
\Leftrightarrow \quad \frac{P(t,T)}{B(t)} &= \frac{P(0,T)}{B(0)} e^{\int_0^t \nu(s,T) dW^{\mathbb{Q}}(s) - \frac{1}{2}\int_0^t ||\nu(t,T)||^2 ds} \\
&= P(0,T)\mathcal{E}_t\left(\nu(\cdot,T) \bullet W^{\mathbb{Q}}\right).
\end{aligned}$$

□

Proposition C.1. The short rate process in the HJM framework is an Itô process of the form

$$r(t) = r(0) + \int_0^t \zeta(u) du + \int_0^t \sigma(u,u) dW(u), \tag{C.6}$$

where

$$\zeta(u) = a(u,u) + \partial_u f(0,u) + \int_0^u \partial_u \alpha(s,u) ds + \int_0^u \partial_u \sigma(s,u) dW(s).$$

Proof. By (8.3) the short rate dynamics:

$$\begin{aligned}
r(t) = f(t,t) &= f(0,t) + \int_0^t \sigma^{\mathbb{Q}}(s,t) ds + \int_0^t \sigma(s,t) dW^{\mathbb{Q}}(s), \\
\sigma^{\mathbb{Q}}(s,t) &= \sigma(s,t) \int_s^t \sigma(s,u)^t op du
\end{aligned}$$

By applying Itô

$$dr(t) = \left(\partial_t f(0,t) + \partial_t \int_0^t \sigma^{\mathbb{Q}}(s,t) ds\right) dt + d\left[\int_0^t \sigma(s,t) dW^{\mathbb{Q}}(t)\right]$$

From which we observe by applying Leibniz rule:

$$\begin{aligned}
\partial_t \int_0^t \sigma^{\mathbb{Q}}(s,t) ds &= \sigma^{\mathbb{Q}}(t,t) + \int_0^t \partial_t \sigma^{\mathbb{Q}}(s,t) ds \\
&= 0 + \int_0^t \left(\partial_t \sigma(s,t) \int_0^t \sigma(s,u) du + \sigma(s,t)\sigma(s,t)^\top\right) ds
\end{aligned}$$

and further application of the Leibniz rule:

$$d\left(\int_0^t \sigma(s,t)dW^{\mathbb{Q}}\right) = \sigma(t,t)dW^{\mathbb{Q}}(t) + \left(\int_0^t \partial_t\sigma(s,t)dW^{\mathbb{Q}}(s)\right)dt,$$

and inserting these expression back into above gives us the short rate drift:

$$dr(t) = \left(\int_0^t \partial_T\sigma(s,t)dW^{\mathbb{Q}}(s) + \int_0^t \sigma(s,t)\sigma(s,t)^\top\right)$$

□

D Deriving the Instantaneous Forward Rate

By inserting (9.4) into the drift condition (9.3) we obtain:

$$\mu_f(t,T) = \sum_{i=1}^{N} \nu_i(t)(\alpha_{0,i} + \alpha_{1,i}(T-t))e^{-\gamma_i(T-t)} \int_t^T (\alpha_{0,i} + \alpha_{1,i}(u-t))e^{-\gamma_i(u-t)}du \quad \text{(D.1)}$$

We consider first the integral part. This can be solved by integrating by parts:

$$\begin{aligned}\int_t^T (\alpha_{0,i} + \alpha_{1,i}(u-t))e^{-\gamma_i(u-t)}du &= \left[\frac{-1}{\gamma_i^2}\left(e^{-\gamma(u-t)}\left(\alpha_{0,i}\gamma_i + \alpha_{1,i}\gamma_i u + \alpha_{1,i} - \alpha_{1,i}\gamma_i t\right)\right)\right]_{u=t}^{u=T} \\ &= \frac{1}{\gamma_i^2}\left[(\alpha_{0,i}\gamma_i + \alpha_{1,i}) - e^{-\gamma(T-t)}(\alpha_{1,i} + \alpha_{1,i}\gamma_i(T-t) + \alpha_{0,i}\gamma_i)\right]\end{aligned}$$

Inserting this expression back into the rewritten drift condition from (D.1) and evaluating the last product terms

$$(\alpha_{0,i} + \alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}\frac{1}{\gamma_i^2}\left[(\alpha_{0,i}\gamma_i + \alpha_{1,i}) - e^{-\gamma(T-t)}(\alpha_{1,i} + \alpha_{1,i}\gamma_i(T-t) + \alpha_{0,i}\gamma_i)\right]$$

$$\begin{aligned}&= \frac{1}{\gamma_i^2}\Big[e^{-\gamma_i(T-t)}(\alpha_{0,i} + \alpha_{1,i}(T-t))(\alpha_{0,i}\gamma_i + \alpha_{1,i}) \\ &\quad - e^{-2\gamma_i(T-t)}(\alpha_{0,i} + \alpha_{1,i}(T-t))(\alpha_{1,i} + \alpha_{1,i}\gamma_i(T-t) + \alpha_{0,i}\gamma_i)\Big]\end{aligned}$$

$$\begin{aligned}&= \frac{1}{\gamma_i^2}\Big[e^{-\gamma_i(T-t)}\left\{\alpha_{0,i}^2\gamma_i + \alpha_{0,i}\alpha_{1,i}\gamma_i(T-t) + \alpha_{0,i}\alpha_{1,i} + \alpha_{1,i}^2(T-t)\right\} \\ &\quad - e^{-2\gamma_i(T-t)}\left\{\alpha_{1,i}\alpha_{0,i} + \alpha_{1,i}^2\gamma_i(T-t)^2 + \alpha_{1,i}^2(T-t) + \alpha_{0,i}^2\gamma_i + 2\alpha_{0,i}\alpha_{1,i}\gamma_i(T-t)\right\}\Big]\end{aligned}$$

$$\begin{aligned}&= \frac{1}{\gamma_i^2}\Big[(\alpha_{1,i}\alpha_{0,i} + \alpha_{0,i}^2\gamma_i)\left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)}\right) \\ &\quad + (\alpha_{1,i}^2 + \alpha_{0,i}\alpha_{1,i}\gamma_i)(T-t)\left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)}\right) - \alpha_{0,i}\alpha_{1,i}\gamma_i(T-t) - e^{-2\gamma_i(T-t)} \\ &\quad + (\alpha_{1,i}^2\gamma_i + \alpha_{1,i}^2)(T-t)^2\left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)}\right) - \alpha_{1,i}^2\gamma_i(T-t)^2e^{-\gamma_i(T-t)}\Big]\end{aligned}$$

$$\begin{aligned}&= \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)}\right) - \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t)e^{-2\gamma_i(T-t)} \\ &\quad + \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(T-t)\left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)}\right) - \frac{\alpha_{1,i}^2}{\gamma_i}(T-t)^2e^{-2\gamma_i(T-t)}.\end{aligned}$$

We insert this expression back into (D.1) to obtain the drift of the forward rate as:

$$\mu_f(t,T) = \sum_{i=1}^{N} \nu_i(t) \Big[\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) \left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)} \right) - \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} (T-t) e^{-2\gamma_i(T-t)} + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-t) \left(e^{-\gamma_i(T-t)} - e^{-2\gamma_i(T-t)} \right) - \frac{\alpha_{1,i}^2}{\gamma_i} (T-t)^2 e^{-2\gamma_i(T-t)} \Big]. \tag{D.2}$$

To show the forward rate curve is Markov in a finite set of state variables we plug our solution of μ_f into the model dynamics (9.2) and solve the stochastic integrals:

$$f(t,T) = f(0,T) + \int_0^t \mu_f(s,T) ds + \sum_{i=1}^{N} \int_0^t \sigma_{f,i}(s,T) \sqrt{\nu_i(s)} dW_i^Q(s). \tag{D.3}$$

We work this out in two parts for simplicity. We start off with the integral of μ_f where we can interchange the integration and summation order in the finite sum:

$$\begin{aligned}
\int_0^t \mu_f(s,T) ds &= \int_0^t \mu_f(s,T) ds \\
&= \int_0^t \sum_{i=1}^{N} \nu_i(s) \Big[\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) - \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} (T-s) e^{-2\gamma_i(T-s)} \\
&\quad + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-s) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) - \frac{\alpha_{1,i}^2}{\gamma_i} (T-s)^2 e^{-2\gamma_i(T-s)} \Big] ds \\
&= \sum_{i=1}^{N} \int_0^t \nu_i(s) \Big[\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) - \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} (T-s) e^{-2\gamma_i(T-s)} \\
&\quad + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-s) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) - \frac{\alpha_{1,i}^2}{\gamma_i} (T-s)^2 e^{-2\gamma_i(T-s)} \Big] ds \\
&= \sum_{i=1}^{N} \Bigg[\int_0^t \nu_i(s) \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) ds && \text{(D.4)} \\
&\quad - \int_0^t \nu_i(s) \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} (T-s) e^{-2\gamma_i(T-s)} ds && \text{(D.5)} \\
&\quad + \int_0^t \nu_i(s) \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-s) \left(e^{-\gamma_i(T-s)} - e^{-2\gamma_i(T-s)} \right) ds && \text{(D.6)} \\
&\quad - \int_0^t \nu_i(s) \frac{\alpha_{1,i}^2}{\gamma_i} (T-s)^2 e^{-2\gamma_i(T-s)} ds \Bigg] && \text{(D.7)}
\end{aligned}$$

Further breaking this down to the constituent terms we get:

$$
\begin{aligned}
\text{(D.4)} &= \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left\{e^{-\gamma(T-t)}\int_0^t \nu_i(s)e^{-\gamma_i(t-s)}ds - e^{-2\gamma(T-t)}\int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds\right\} \\
&= \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left\{e^{-\gamma(T-t)}\phi_{2,i}(t) - e^{-2\gamma(T-t)}\phi_{3,i}(t)\right\}, \\
\text{(D.5)} &= -\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t)e^{-2\gamma_i(T-t)}\left\{\int_0^t \nu_i(s)e^{\gamma_i}(t-s)ds + \int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds\right\} \\
&= -\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t)e^{-2\gamma_i(T-t)}\left\{\phi_{3,i}(t)+\phi_{5,i}(t)\right\}, \\
\text{(D.6)} &= \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left\{(T-t)\int_0^t \nu_i(s)\left(e^{-\gamma_i(T-s)}-e^{-2\gamma_i(T-s)}\right)ds\right. \\
&\qquad\qquad \left. + \int_0^t \nu_i(s)(t-s)\left(e^{-\gamma_i(T-s)}-e^{-2\gamma_i(T-s)}\right)ds\right\} \\
&= \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(T-t)\left(e^{-\gamma_i(T-t)}\int_0^t \nu_i(s)e^{-\gamma_i(t-s)}ds - e^{-2\gamma_i(T-t)}\int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds\right) \\
&\quad + \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i(T-t)}\int_0^t \nu_i(s)(t-s)e^{-\gamma_i(t-s)}ds - e^{-2\gamma_i(T-t)}\int_0^t \nu_i(s)(t-s)e^{-2\gamma_i(t-s)}ds\right) \\
&= \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(T-t)\left(e^{-\gamma_i(T-t)}\phi_{2,i}(t) - e^{-2\gamma_i(T-t)}\phi_{3,i}(t)\right) \\
&\quad + \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i(T-t)}\phi_{4,i}(t) - e^{-2\gamma_i(T-t)}\phi_{5,i}(t)\right), \\
\text{(D.7)} &= -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}\int_0^t \nu_i(s)\left((T-t)+(t-s)\right)^2 e^{-2\gamma_i(t-s)}ds \\
&= -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}\left\{(T-t)^2\int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds + \int_0^t \nu_i(s)(t-s)^2e^{-2\gamma_i(t-s)}ds\right. \\
&\qquad\qquad \left. + 2(T-t)\int_0^t \nu_i(s)(t-s)e^{-2\gamma_i(t-s)}ds\right\} \\
&= -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}\left\{(T-t)^2\phi_{3,i}(t)+\phi_{6,i}(t)+2(T-t)\phi_{5,i}(t)\right\}.
\end{aligned}
$$

Putting these terms together back in the integral of μ_f we can then write this as:

$$\begin{aligned}
\int_0^t \mu_f(s,T)ds = \sum_{i=1}^{N} \Bigg\{ & \left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-t) \right) e^{-\gamma_i(T-t)} \phi_{2,i}(t) \\
& - \left[\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (T-t) + \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t) + \frac{\alpha_{1,i}^2}{\gamma}(T-t)^2 \right] e^{-2\gamma_i(T-t)} \phi_{3,i}(t) \\
& + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) e^{-\gamma(T-t)} \phi_{4,i}(t) \\
& - \left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}(T-t) + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) + 2\frac{\alpha_{1,i}^2}{\gamma_i}(T-t) \right) e^{-2\gamma_i(T-t)} \phi_{5,i}(t) \\
& - \frac{\alpha_{1,i}^2}{\gamma_i} e^{-2\gamma_i(T-t)} \phi_{6,i}(t) \Bigg\} \\
= \sum_{i=1}^{N} \Bigg\{ & \left(\frac{\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) (\alpha_{0,i} + \alpha_{1,i}(T-t)) \right) e^{-\gamma_i(T-t)} \phi_{2,i}(t) \\
& - \left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i} \left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) + \frac{\alpha_{1,i}}{\gamma_i} \left(\frac{\alpha_{1,i}}{\gamma} + 2\alpha_{0,i} \right) (T-t) + \frac{\alpha_{1,i}^2}{\gamma}(T-t)^2 \right) e^{-2\gamma_i(T-t)} \phi_{3,i}(t) \\
& + \frac{\alpha_{1,i}^2}{\gamma_i} \left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}} \right) e^{-\gamma(T-t)} \phi_{4,i}(t) \\
& - \frac{\alpha_{1,i}}{\gamma_i} \left(\frac{\alpha_{1,i}}{\gamma_i} + 2\alpha_{0,i} + 2\alpha_{1,i}(T-t) \right) e^{-2\gamma_i(T-t)} \phi_{5,i}(t) \\
& - \frac{\alpha_{1,i}^2}{\gamma_i} e^{-2\gamma_i(T-t)} \phi_{6,i}(t) \Bigg\}.
\end{aligned}$$

This gives us an expression for the integral of μ_f to work with which depend on the *ad hoc* defined variables $\phi_{2,i}(t), ..., \phi_{6,i}(t)$. We will show below what the idea is behind

introducing these, but first we have to work a bit on the last term in (D.3):

$$\begin{aligned}
\sum_{i=1}^{N}\int_0^t \sigma_{f,i}(s,T)\sqrt{\nu_i(s)}dW_i^Q(s) &= \sum_{i=1}^{N}\int_0^t (\alpha_{0,i}+\alpha_{1,i}(T-s))e^{-\gamma(T-s)}\sqrt{\nu_i(s)}dW_i^Q(s)\\
&= \sum_{i=1}^{N}\int_0^t (\alpha_{0,i}+\alpha_{1,i}(T-t)+\alpha_{1,i}(t-s))e^{-\gamma(T-s)}\sqrt{\nu_i(s)}dW_i^Q(s)\\
&= \sum_{i=1}^{N}\Bigg\{(\alpha_{0,i}+\alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}\int_0^t \sqrt{\nu_i(s)}e^{-\gamma_i(t-s)}dW_i^Q(s)\\
&\qquad + \alpha_{1,i}e^{-\gamma_i(T-t)}\int_0^t \sqrt{\nu_i(s)}(t-s)e^{-\gamma_i(t-s)}dW_i^Q(s)\Bigg\}\\
&= \sum_{i=1}^{N}(\alpha_{0,i}+\alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}x_i(t)+\alpha_{1,i}e^{-\gamma_i(T-t)}\phi_{1,i}(t),
\end{aligned}$$

where we introduced the additional variables $x_i(t)$ and $\phi_{1,i}(t)$. Inserting these two terms back in (D.3) we obtain finally:

$$\begin{aligned}
f(t,T) &= f(0,T)+\sum_{i=1}^{N}(\alpha_{0,i}+\alpha_{1,i}(T-t))e^{-\gamma_i(T-t)}x_i(t)\\
&\quad+\sum_{i=1}^{N}\Bigg\{\alpha_{1,i}e^{-\gamma_i(T-t)}\phi_{1,i}(t)\\
&\quad+\left(\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(\alpha_{0,i}+\alpha_{1,i}(T-t))\right)e^{-\gamma_i(T-t)}\phi_{2,i}(t)\\
&\quad-\left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)+\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma}+2\alpha_{0,i}\right)(T-t)+\frac{\alpha_{1,i}^2}{\gamma}(T-t)^2\right)e^{-2\gamma_i(T-t)}\phi_{3,i}(t)\\
&\quad+\frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)e^{-\gamma(T-t)}\phi_{4,i}(t)\\
&\quad-\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i}+2\alpha_{0,i}+2\alpha_{1,i}(T-t)\right)e^{-2\gamma_i(T-t)}\phi_{5,i}(t)\\
&\quad-\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}\phi_{6,i}(t)\Bigg\}\\
&\equiv f(0,T)+\sum_{i=1}^{N}\mathcal{B}_{x_i}(T-t)x_i(t)+\sum_{i=1}^{N}\sum_{j=1}^{6}\mathcal{B}_{\phi_{j,i}}(T-t)\phi_{j,i}(t). \qquad \text{(D.8)}
\end{aligned}$$

by defining the functions:

$$\mathcal{B}_{x_i}(\tau) = (\alpha_{0,i} + \alpha_{1,i}\tau)e^{-\gamma_i\tau},$$
$$\mathcal{B}_{\phi_{1,i}}(\tau) = \alpha_{1,i}e^{-\gamma_i\tau},$$
$$\mathcal{B}_{\phi_{2,i}}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)(\alpha_{0,i} + \alpha_{1,i}(T-t)),$$
$$\mathcal{B}_{\phi_{3,i}}(\tau) = -\left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right) + \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma} + 2\alpha_{0,i}\right)(T-t) + \frac{\alpha_{1,i}^2}{\gamma}(T-t)^2\right)e^{-2\gamma_i(T-t)},$$
$$\mathcal{B}_{\phi_{4,i}}(\tau) = \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)e^{-\gamma(T-t)},$$
$$\mathcal{B}_{\phi_{5,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i} + 2\alpha_{0,i} + 2\alpha_{1,i}(T-t)\right)e^{-2\gamma_i(T-t)},$$
$$\mathcal{B}_{\phi_{6,i}}(\tau) = -\frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}.$$

We restate the state variables as we defined ad hoc in the above:

$$\begin{aligned}
x_i(t) &= \int_0^t \sqrt{\nu_i(s)}e^{-\gamma_i(t-s)}dW_i^Q(s),\\
\phi_{1,i}(t) &= \int_0^t \sqrt{\nu_i(s)}(t-s)e^{-\gamma_i(t-s)}dW_i^Q(s),\\
\phi_{2,i}(t) &= \int_0^t \nu_i(s)e^{-\gamma_i(t-s)}ds,\\
\phi_{3,i}(t) &= \int_0^t \nu_i(s)e^{-2\gamma_i(t-s)}ds,\\
\phi_{4,i}(t) &= \int_0^t \nu_i(s)(t-s)e^{-\gamma_i(t-s)}ds,\\
\phi_{5,i}(t) &= \int_0^t \nu_i(s)(t-s)e^{-2\gamma_i(t-s)}ds,\\
\phi_{6,i}(t) &= \int_0^t \nu_i(s)(t-s)^2e^{-2\gamma_i(t-s)}ds.
\end{aligned}$$

To obtain the dynamics of these state variables we note, that if we define the function $g(t) = \int_0^t f(s,t)dW(s)$ for a sufficiently smooth function f, we can show the following

relation by an application of stochastic Fubini:

$$\begin{aligned}
g(t) - g(0) &= \int_0^t f(s,t)dW(s) - \int_0^t f(s,s)dW(s) + \int_0^t f(t,t)dW(t) \\
&= \int_0^t \left(f(s,t) - f(s,s)\right) dW(s) + \int_0^t f(u,u)dW(u) \\
&= \int_0^t + \int_s^t \frac{\partial}{\partial u} f(s,u)dudW(s) + \int_0^t f(u,u)dW(u) \\
&= \int_0^t \int_0^u \frac{\partial}{\partial u} f(s,u)dW(s)du + \int_0^t f(u,u)dW(u) \\
\Rightarrow g(t) &= g(0) + \int_0^t \int_0^u \frac{\partial}{\partial u} f(s,u)dW(s)du + \int_0^t f(u,u)dW(u)
\end{aligned}$$

From the integral representation, we can rewrite to the stochastic differential equations to obtain the dynamics. Applying this result to the first two of above definitions of the state variables (since these are stochastic integrals), gives us:

$$\begin{aligned}
dx_i(t) &= -\gamma_i \left(\int_0^t \sqrt{\nu_i(s)} e^{-\gamma_i(t-s)} dW_i^Q(s) \right) dt + \sqrt{\nu_i(t)} dW_i^{\mathbb{Q}}(t) \qquad \text{(D.9)} \\
&= -\gamma_i x_i dt + \sqrt{\nu_i(t)} dW_i^{\mathbb{Q}}(t), \\
d\phi_{1,i}(t) &= \left(-\gamma_i \int_0^t \sqrt{\nu_i(s)} (t-s) e^{-\gamma_i(t-s)} dW_i^Q(s) + \int_0^t \sqrt{\nu_i(s)} e^{-\gamma_i(t-s)} dW_i^Q(s) \right) dt \\
&= (x_i(t) - \gamma_i \phi_{1,i}(t)) dt \qquad \text{(D.10)}
\end{aligned}$$

And Ito's lemma and Leibniz rule gives us the rest:

$$d\phi_{2,i}(t) = \left(\nu_i(t) - \gamma_i \int_0^t \nu_i(s) e^{-\gamma_i(t-s)} ds\right) dt = (\nu_i(t) - \gamma_i \phi_{2,i}(t)) dt, \tag{D.11}$$

$$d\phi_{3,i}(t) = \left(\nu_i(t) - 2\gamma_i \int_0^t \nu_i(s) e^{-2\gamma_i(t-s)} ds\right) dt = (\nu_i(t) - 2\gamma_i \phi_{3,i}(t)) dt, \tag{D.12}$$

$$\begin{aligned} d\phi_{4,i}(t) &= \left(\int_0^t \nu_i(s) e^{-\gamma_i(t-s)} ds - \gamma_i \int_0^t \nu_i(s)(t-s) e^{-\gamma_i(t-s)} ds\right) dt \\ &= (\phi_{2,i}(t) - \gamma_i \phi_{4,i}(t)) dt, \end{aligned} \tag{D.13}$$

$$\begin{aligned} d\phi_{5,i}(t) &= \left(\int_0^t \nu_i(s) e^{-2\gamma_i(t-s)} ds - 2\gamma_i \int_0^t \nu_i(s)(t-s) e^{-2\gamma_i(t-s)} ds\right) dt \\ &= (\phi_{3,i}(t) - 2\gamma_i \phi_{5,i}(t)) dt, \end{aligned} \tag{D.14}$$

$$\begin{aligned} d\phi_{6,i}(t) &= \left(2\int_0^t \nu_i(s)(t-s) e^{-2\gamma_i(t-s)} ds - 2\gamma_i \int_0^t \nu_i(s)(t-s)^2 e^{-2\gamma_i(t-s)} ds\right) dt \\ &= (2\phi_{5,i}(t) - 2\gamma_i \phi_{6,i}(t)) dt. \end{aligned} \tag{D.15}$$

E Zero-Coupon Bond Price Reconstruction Formula

We consider the expression (9.6) term by term. We see, that the first one rewrites to:

$$\exp\left\{-\int_t^T f(0,u)du\right\} = \exp\left\{\int_0^t f(0,u)du - \int_0^T f(0,u)du\right\} = \frac{P(0,T)}{P(0,t)}.$$

And the integral in the second term amounts to

$$-\int_t^T \sum_{i=1}^N \mathcal{B}_{x_i}(u-t)x_i(t)du = \sum_{i=1}^N -x_i(t)\int_t^T \mathcal{B}_{x_i}(u-t)du,$$

where we focus only on the integration part:

$$\begin{aligned}
-\int_t^T \mathcal{B}_{x_i}(u-t)du &= -\int_t^T (\alpha_{0,i}+\alpha_{1,i}(u-t))e^{-\gamma_i(u-t)}du \\
&= -\alpha_{0,i}e^{\gamma_i t}\int_t^T e^{-\gamma_i(u-t)}du + \alpha_{1,i}e^{\gamma_i t}\left(\int_t^T ue^{-\gamma_i u}du - t\int_t^T e^{-\gamma_i u}du\right) \\
&= \alpha_{0,i}e^{\gamma_i t}\left[\frac{e^{-\gamma_i u}}{\gamma_i}\right]_t^T - \alpha_{1,i}e^{\gamma_i t}\left(\left[\frac{-e^{-\gamma_i u}(\gamma_i u+1)}{\gamma_i^2}\right]_t^T + t\left[\frac{e^{-\gamma_i u}}{\gamma_i}\right]_t^T\right) \\
&= \frac{\alpha_{0,i}}{\gamma_i}\left(e^{-\gamma(T-t)}-1\right) - \alpha_{1,i}e^{\gamma_i t}\left(\left(\frac{e^{-\gamma_i t}(\gamma_i t+1)}{\gamma_i^2} - \frac{e^{-\gamma_i T}(\gamma_i T+1)}{\gamma_i^2}\right) + \frac{t}{\gamma_i}\left(e^{-\gamma_i T}-e^{-\gamma_i t}\right)\right) \\
&= \frac{\alpha_{0,i}}{\gamma_i}\left(e^{-\gamma(T-t)}-1\right) - \alpha_{1,i}\left(\frac{\gamma_i t+1}{\gamma_i^2} - \frac{e^{-\gamma_i(T-t)}(\gamma_i T+1)}{\gamma_i^2} + t\frac{e^{-\gamma_i(T-t)}-1}{\gamma_i}\right) \\
&= \frac{\alpha_{0,i}}{\gamma_i}\left(e^{-\gamma(T-t)}-1\right) + \frac{\alpha_{1,i}}{\gamma_i}\left((T-t)e^{-\gamma_i(T-t)} + \frac{1}{\gamma_i}e^{-\gamma_i(T-t)} - \frac{1}{\gamma_i}\right) \\
&= \left(\frac{\alpha_{1,i}}{\gamma_i^2}+\frac{\alpha_{0,1}}{\gamma_i}\right)\left(e^{-\gamma_i\tau}-1\right) + \frac{\alpha_{1,i}\tau}{\gamma_i}e^{-\gamma_i\tau} \\
&= \frac{\alpha_{1,i}}{\gamma_i}\left(\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,1}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right) + \tau e^{-\gamma_i\tau}\right) \equiv B_{x_i}(\tau).
\end{aligned}$$

And finally, we solve the integral in the last term by splitting further down into six parts:

$$\begin{aligned}
&-\int_t^T \sum_{i=1}^N \sum_{j=1}^6 \mathcal{B}_{\phi_{j,i}}(u-t)\phi_{j,i}(t)du \\
&\qquad = \sum_{i}^N -\phi_{1,i}(t)\int_t^T \mathcal{B}_{\phi_{1,i}}(u-t)du - \ldots - \phi_{6,i}(t)\int_t^T \mathcal{B}_{\phi_{6,i}}(u-t)du
\end{aligned}$$

where

$$-\int_t^T \mathcal{B}_{\phi_{1,i}}(u-t)du = -\alpha_{1,i}e^{\gamma_i t}\int_t^T e^{-\gamma_i u}du = \alpha_{1,i}\left(e^{-\gamma_i\tau}-1\right) \equiv B_{\phi_{1,i}}(\tau).$$

and

$$\begin{aligned}
-\int_t^T \mathcal{B}_{\phi_{2,i}}(u-t)du &= -\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\int_t^T(\alpha_{0,i}+\alpha_{1,i}(u-t))e^{-\gamma_i(u-t)}du \\
&= \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)B_{x_i}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\frac{\alpha_{1,i}}{\gamma_i}\left(\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,1}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right)+\tau e^{-\gamma_i\tau}\right) \\
&= \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right)+\tau e^{-\gamma_i\tau}\right) =: B_{\phi_{2,i}}(\tau)
\end{aligned}$$

and

$$\begin{aligned}
&-\int_t^T \mathcal{B}_{\phi_{3,i}}(u-t)du \\
&= \int_t^T\left(\frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)+\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma}+2\alpha_{0,i}\right)(u-t)+\frac{\alpha_{1,i}^2}{\gamma}(u-t)^2\right)e^{-2\gamma_i(u-t)}du \\
&= \frac{\alpha_{0,i}\alpha_{1,i}}{\gamma_i}\left(\frac{1}{\gamma}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\int_t^T e^{-2\gamma_i(u-t)}du \\
&\quad+\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma}+2\alpha_{0,i}\right)\int_t^T(u-t)e^{-2\gamma_i(u-t)}du+\frac{\alpha_{1,i}^2}{\gamma}\int_t^T(u-t)^2e^{-2\gamma_i(u-t)}du \\
&= \frac{\alpha_{0,i}\alpha_{1,i}}{2\gamma_i^2}\left(\frac{1}{\gamma}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left[1-e^{-2\gamma_i(T-t)}\right] \\
&\quad+\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma}+2\alpha_{0,i}\right)\left[\frac{2t\gamma_i+1}{4\gamma_i^2}-\frac{t}{2\gamma_i}-\frac{(2T\gamma_i+1)e^{-2\gamma_i(T-t)}}{4\gamma_i^2}+te^{-2\gamma_i(T-t)}\right] \\
&\quad+\frac{\alpha_{1,i}^2}{\gamma}\left[\frac{2t\gamma_i(t\gamma_i+1)+1}{4\gamma_i^3}-\frac{(2T\gamma_i(T\gamma_i+1)+1)e^{-2\gamma_i(T-t)}}{4\gamma_i^3}+\frac{t^2}{2\gamma_i}\left(1-e^{-2\gamma_i(T-t)}\right)\right. \\
&\quad\left.-t\left(\frac{2t\gamma_i+1}{2\gamma_i^2}-\frac{(2T\gamma_i+1)e^{-2\gamma_i(T-t)}}{2\gamma_i^2}\right)\right] \\
&= -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{2\gamma_i^2}+\frac{\alpha_{0,i}}{\gamma_i}+\frac{\alpha_{0,i}^2}{2\alpha_{1,i}}\right)\left(e^{-2\gamma_i\tau}-1\right)+\left(\frac{\alpha_{1,i}}{\gamma_i}+\alpha_{0,i}\right)\tau e^{-2\gamma_i\tau}+\frac{\alpha_{1,i}}{2}\tau^2e^{-2\gamma_i\tau}\right) \\
&\equiv B_{\phi_{3,i}}(\tau).
\end{aligned}$$

and

$$
\begin{aligned}
-\int_t^T \mathcal{B}_{\phi_{4,i}}(u-t)du &= -\int_t^T \frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)e^{-\gamma(T-t)}du \\
&= -\frac{\alpha_{1,i}^2}{\gamma_i}\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\int_t^T e^{-\gamma(T-t)}du = \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i}+\frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau}-1\right) \equiv B_{\phi_{4,i}}(\tau)
\end{aligned}
$$

and

$$
\begin{aligned}
&-\int_t^T \mathcal{B}_{\phi_{5,i}}(u-t)du = \int_t^T \frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i}+2\alpha_{0,i}+2\alpha_{1,i}(u-t)\right)e^{-2\gamma_i(u-t)}du \\
&= \frac{\alpha_{1,i}}{\gamma_i}e^{2\gamma_i t}\left(\frac{\alpha_{1,i}}{\gamma_i}\int_t^T e^{-2\gamma_i u}du + 2\alpha_{0,i}\int_t^T e^{-2\gamma_i u}du + 2\alpha_{1,i}\int_t^T ue^{-2\gamma_i u}du - 2\alpha_{1,0}t\int_t^T e^{-2\gamma_i u}du\right) \\
&= \frac{\alpha_{1,i}}{\gamma_i}e^{2\gamma_i t}\left[\frac{-\alpha_{1,i}}{2\gamma_i^2}\left(e^{-2\gamma_i T}-e^{-2\gamma_i t}\right)-\frac{\alpha_{0,i}}{\gamma_i}\left(e^{-2\gamma_i T}-e^{-2\gamma_i t}\right)\right. \\
&\qquad \left.+\frac{\alpha_{1,i}}{2\gamma_i^2}\left((2t\gamma_i+1)e^{-2\gamma_i t}-(2T\gamma_i+1)e^{-2\gamma_i T}\right)-\frac{\alpha_{1,i}t}{\gamma_i}\left(e^{-2\gamma_i T}-e^{-2\gamma_i t}\right)\right] \\
&= \frac{\alpha_{1,i}}{\gamma_i}\left[\frac{-\alpha_{1,i}}{2\gamma_i^2}\left(e^{-2\gamma_i\tau}-1\right)-\frac{\alpha_{0,i}}{\gamma_i}\left(e^{-2\gamma_i\tau}-1\right)\right. \\
&\qquad \left.+\frac{\alpha_{1,i}}{2\gamma_i^2}\left((2t\gamma_i+1)-(2T\gamma_i+1)e^{-2\gamma_i\tau}\right)-\frac{\alpha_{1,i}t}{\gamma_i}\left(e^{-2\gamma_i\tau}-1\right)\right] \\
&= -\frac{\alpha_{1,i}}{\gamma_i}\left(\frac{\alpha_{1,i}}{\gamma_i^2}\left(e^{-2\gamma_i\tau}-1\right)-\frac{\alpha_{0,i}}{\gamma_i}\left(e^{-2\gamma_i\tau}-1\right)+\frac{\alpha_{1,i}}{\gamma_i}\tau e^{-2\gamma_i\tau}\right) \\
&= -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{\gamma_i}+\alpha_{0,i}\right)\left(e^{-2\gamma_i\tau}-1\right)+\alpha_{1,i}\tau e^{-2\gamma_i\tau}\right) \equiv B_{\phi_{5,i}}(\tau),
\end{aligned}
$$

and

$$
\begin{aligned}
-\int_t^T \mathcal{B}_{\phi_{6,i}}(u-t)du &= \int_t^T \frac{\alpha_{1,i}^2}{\gamma_i}e^{-2\gamma_i(T-t)}du \\
&= \frac{\alpha_{1,i}^2}{\gamma_i}\frac{1}{2\gamma_i}\left(1-e^{-2\gamma_i\tau}\right) = -\frac{1}{2}\left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(e^{-2\gamma_i\tau}-1\right) \equiv B_{\phi_{6,i}}(\tau).
\end{aligned}
$$

Putting everything together gives us the zero-coupon bond price as

$$
\begin{aligned}
P(t,T) &= \exp\left\{-\int_t^T f(t,u)du\right\} \\
&= \frac{P(0,t)}{P(0,t)}\exp\left\{\sum_{i=1}^N B_{x_i}(T-t)x_i(t)+\sum_{i=1}^N\sum_{j=1}^6 B_{\phi_{j,i}}(T-t)\phi_{j,i}(t)\right\} \qquad \text{(E.1)}
\end{aligned}
$$

Using the same definitions of the state variables as in the forward rate (D.8) but defining the new functions:

$$B_{x_i}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right) + \tau e^{-\gamma_i\tau}\right),$$

$$B_{\phi_{1,i}}(\tau) = \frac{\alpha_{1,i}}{\gamma_i}\left(e^{-\gamma_i\tau} - 1\right),$$

$$B_{\phi_{2,i}}(\tau) = \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right) + \tau e^{-\gamma_i\tau}\right),$$

$$B_{\phi_{3,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{2\gamma_i^2} + \frac{\alpha_{0,i}}{\gamma_i} + \frac{\alpha_{0,i}^2}{2\alpha_{1,i}}\right)\left(e^{-2\gamma_i\tau} - 1\right) + \left(\frac{\alpha_{1,i}}{\gamma_i} + \alpha_{0,i}\right)\tau e^{-2\gamma_i\tau} + \frac{\alpha_{1,i}}{2}\tau^2 e^{-2\gamma_i\tau}\right),$$

$$B_{\phi_{4,i}}(\tau) = \left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(\frac{1}{\gamma_i} + \frac{\alpha_{0,i}}{\alpha_{1,i}}\right)\left(e^{-\gamma_i\tau} - 1\right),$$

$$B_{\phi_{5,i}}(\tau) = -\frac{\alpha_{1,i}}{\gamma_i^2}\left(\left(\frac{\alpha_{1,i}}{\gamma_i} + \alpha_{0,i}\right)\left(e^{-2\gamma_i\tau} - 1\right) + \alpha_{1,i}\tau e^{-2\gamma_i\tau}\right),$$

$$B_{\phi_{6,i}}(\tau) = -\frac{1}{2}\left(\frac{\alpha_{1,i}}{\gamma_i}\right)^2\left(e^{-2\gamma_i\tau} - 1\right).$$