Corvinus University of Budapest - Eötvös Loránd University

Calibrating HJM models using deep neural networks

Máté Kurucz

Thesis
Actuarial and Financial Mathematics, MSc
Quantitative Finance Specialization

Thesis supervisors:
Gergely Bence Szilágyi (Morgan Stanley), Miklós Arató (ELTE)

Budapest, 2025

Contents

Acknowledgements

I would like to thank to my thesis supervisor, Gergely Bence Szilágyi, for his academic and professional guidance over the past two semesters. His consistent availability and thought-provoking questions were invaluable throughout the process. I also thank Miklós Arató for pointing out new perspectives for this thesis, and for supervising both my BSc and MSc theses, as well as my work for the Scientific Student's Associations Conference. I greatly enjoyed working with him. I would also like to thank Csaba Kőrössy for providing the thesis topic and for his help in deepening my understanding throughout the process.

I would like to thank Tamás Szűcs for his valuable insights regarding neural network training. Last but not least, I would like to express my appreciation to my girlfriend, Anna Kelemen, for her unwavering support and motivation.

Introduction

The Heath-Jarrow-Morton (HJM) framework [17] is one of the industry standard modeling frameworks with regards to the pricing of interest rate derivatives. The flexibility of the model parameterization enables a great variety of interest rate environments to be modeled within the framework. As the generic case does not have a closed-form analytical pricing formula, pricing requires computationally heavy Monte Carlo simulation. The optimization problem solved in the calibration process involves multidimensional functions with non-trivial co-dependencies in the variables.

Given recent advances in the application of deep neural networks for effectively interpolating multidimensional complex functions, there is a potential in using deep neural networks for complex calibration problems. This can be especially well utilized in stress testing applications, where calibration under large shocks might be more challenging when applying traditional optimization-based methods, and when accuracy requirements might not be as stringent.

The primary objective of this thesis is to perform volatility calibration for at-the-money swaptions using deep neural networks under specific choices of volatility and initial forward curve parametrizations.

In the first section, we aim to introduce the HJM framework with all necessary assumptions and conditions, not excluding measurability in the multi-factor case, primarily following [17]. Subsequently, a concise overview is provided of volatility parametrizations that have emerged in the literature. [17],[35],[27],[35]. Despite the fact that our calibration efforts do not encompass stochastic volatility, we incorporate one such model to offer a broader perspective ([10]). We then address the challenge of appropriate discretization and implementation in the framework in Section 2, mainly following [16]. Our implementation is consequently benchmarked with a continuous-time setup. As the consequent pricing of fix cash flow instruments - fix coupon swap legs are no exception to this - provides a solid foundation for any HJM implementation, an articulated emphasis is placed on matching first moments of quantities like discount factors and discounted zero-coupon bonds. Throughout this thesis, our focus remains exclusively on vanilla instruments: while both a caplet and a swaption pricer were implemented, only the latter was used for calibration purposes for at-the-money instances.

The main focus in the third section is to provide background for the LIBOR transition [33],[34] and the calculation of both LIBOR and new term structure rates, such as SOFR [1],[14]. As

SOFR serves as the sole underlying rate of our calibration instruments, its calculation rules and dynamics directly influence the practical considerations for choosing an appropriate approximation theme. We consider several approximation themes and ultimately choose the most intuitive one.

In the fourth section, we aim to provide an introduction to deep neural networks to ensure sufficient background for our calibration efforts, using [18], [19],[20],[21],[24],[25].

The final section presents our numerical results. After generating swaption price maps using our HJM implementation in Python over a multidimensional parameter grid for two variants of the volatility parametrizations introduced in Section 1, we evaluate the accuracy of the trained model by predicting volatility on synthetic test data located between the original training grid points. These results demonstrate the potential of the approach not only for stress testing scenarios, but also for improved initialization of gradient-based calibration methods and, with adequate infrastructure, for independent real-time at-the-money calibration.

1. Heath-Jarrow-Morton framework

The Heath-Jarrow-Morton (later: HJM) framework belongs to the family of continuous forward rate models, which characterize the forward rate evolution by describing an entire term structure through multiple factors while avoiding arbitrage. The framework was first published in 1992 by the three authors giving their name to model in [17], and deviated from previous principles of determining solely the dynamics of the short rate in order to emulate an interest rate market.

In this section, we will derive market dynamics under the risk-neutral measure largely following the original article [17], arriving at the conclusion that in order to calibrate the model given an initial forward curve in the present, one is required to find the appropriate volatility functions and their respective parameters for the specific market. We then proceed by presenting a selection of volatility parametrizations that have emerged in the literature, without aiming for completeness.

Introduction and deriving market dynamics

The methodology in this subsection closely follows [17], and any deviations from their approach are explicitly indicated. The model is formulated within a continuously trading economy, represented by a probability space $(\Sigma, \mathcal{F}, Q)$. Market information evolves over the trading horizon $[0, \tau]$, where τ denotes the maximal maturity. The filtration capturing the flow of information is generated by a set of $n \geq 1$ independent Brownian motions $\{W_1(t), \ldots, W_n(t) : t \in [0, \tau]\}$. A fundamental assumption of the model is the existence of a continuum of default-free discount bonds $P(t, T)$, each maturing at time T, with $T \in [0, \tau]$, that are continuously traded in the market.

Definition 1.1 (Zero-coupon bond)**.** Let $P(t, T)$ denote, for all $0 \leq t \leq T \leq \tau$, the price at time t of a default-free zero-coupon bond maturing at time T, with payoff $P(T, T) = 1$ and the condition $P(t, T) > 0$.

Definition 1.2 (Instantaneous forward rate)**.** The instantaneous forward rate $f(t, T)$ at time T as observed at time t is defined by

$$f(t, T) := -\frac{\partial}{\partial T} \log\left(P(t, T)\right). \tag{1}$$

The instantaneous forward rate represents the marginal cost of borrowing (and lending) at time T given all the information in $\mathcal{F}_t$. The instantaneous forward rate $f(t,T)$ can be used to approximate the price of a zero-coupon bond over an infinitesimal maturity increment dT, under the assumption of continuous compounding. This yields the approximation:

$$P(t, T + dT) \approx P(t, T)e^{-f(t,T)dT}.$$

If the above approximation holds, then the following limit exists:

$$f(t,T) = \lim_{dT \to 0} -\frac{\log P(t, T + dT) - \log P(t,T)}{dT}.$$

In the HJM framework, and throughout this entire thesis, we assume that this limit exists for all t, T such that $0 \leq t \leq T < \tau$. This implies that the definition of the instantaneous forward rate is well-defined for all such t and T.
Solving equation (1) for the discount bond price $P(t,T)$ yields

$$P(t,T) = \exp\left(-\int_t^T f(t,s)\,ds\right), \quad \forall\, t, T : 0 \leq t \leq T \leq \tau. \tag{2}$$

Definition 1.3 (Spot rate)**.** We define the spot rate $r(t)$ at time as

$$r(t) = f(t,t), \quad \forall t : 0 \leq t \leq \tau. \tag{3}$$

The sport rate is the accumulation factor that drives the risk-free money-market account, which will serve as a numeraire for characterizing a risk-neutral measure.

Definition 1.4 (Money market account)**.** Let us denote the money market account price process by B, such that

$$B(t) = \exp\left(\int_0^r r(y)dy\right). \tag{4}$$

We have now identified the three key interest rate concepts relevant to the framework. Among these, only zero-coupon bond prices observed at valuation time $t = 0$ are directly observable in the market. The remaining two rates—the instantaneous forward rate and the short rate—can be derived from the continuum of discount bond prices. In practice, however, the modeling approach proceeds in the reverse direction: the dynamics of the forward rates are specified first, from which bond prices are then implied.

We now establish the family of instantaneous forward rate dynamics considered in the HJM

framework.

Definition 1.5 (Condition 1: Forward Rate Dynamics)**.** For fixed, but arbitrary $T \in [0, \tau]$, $f(t, T)$ satisfies the following equation:

$$f(t,T) - f(0,T) = \int_0^t \alpha(\nu, T, \omega)\, d\nu + \sum_{i=1}^{n} \int_0^t \sigma_i(\nu, T, \omega)\, dW_i(\nu) \quad \text{for all } 0 \le t \le T. \tag{5}$$

where:

(i) $\{f(0,T) : T \in [0,\tau]\}$ is a fixed, nonrandom initial forward rate curve, and $f(0,\cdot)$ is Borel measurable on $[0,\tau]$.

(ii) $\alpha : \{(t,s) : 0 \le t \le s \le T\} \times \Omega \to \mathbb{R}$ is jointly Borel measurable on $[0,T] \times [0,\tau]$, real-valued, adapted, and satisfies

$$\int_0^T |\alpha(\omega, t, T)|\, dt < +\infty \quad \text{a.e. } Q.$$

(iii) The real-valued volatility functions $\sigma_i : \{(t,s) : 0 \le t \le s \le T\} \times \Omega \to \mathbb{R}$ are jointly Borel measurable on $[0,T] \times [0,\tau]$, adapted, and satisfy

$$\int_0^T \sigma_i^2(t, T, \omega)\, dt < +\infty \quad \text{a.e. } Q, \quad \text{for } i = 1, \ldots, n.$$

The volatility coefficients, which specify sensitivity to the Brownian motions driving the information in the market, are specified up to such conditions that allow a wide range of interest rate environment dynamics to be simulated via the framework. [17]

Remark 1.6. Although the processes α and σ_i (for all $i \in \{1, \ldots, n\}$) are, in general, stochastic processes, at times we suppress the notational dependence on the underlying probability variable ω in the notation. This is justified for three reasons: first, the original article adopts the same simplification (albeit at a later stage); second, it improves notational clarity; and third—most practically—from a self-serving perspective, we shall assume deterministic volatility functions throughout the calibration exercises presented in this thesis.

After deriving the spot rate dynamics below by simple substitution, regularity conditions are imposed on the money market account, in order for it to be positive and finite almost everywhere

regardless of when we examine it in the trading interval. (5) yields

$$r(t) = f(0,t) + \int_0^t \alpha(s,t)\,ds + \sum_{i=1}^{n} \int_0^t \sigma_i(s,t)\,dW_i(s) \quad \forall t : 0 \le t \le T. \tag{6}$$

Definition 1.7 (Condition 2)**.**

$$\int_0^\tau |f(0,s)|ds < +\infty \quad \text{and} \quad \int_0^\tau \left(\int_0^t |\alpha(s,t)|ds \right) dt < +\infty \quad \text{a.e.}Q. \tag{7}$$

In order to allow for only well-behaved bond prices in the framework, some additional regularity conditions are required and introduced in [17].

Definition 1.8 (Condition 3)**.**

$$\int_0^t \left[\int_s^t \sigma_i(s,y)dy \right]^2 ds < +\infty \quad \text{a.e.}Q \quad \forall t : 0 \le t \le \tau \text{ and } \forall i \in [n]; \tag{8}$$

$$\int_0^t \left[\int_t^T \sigma_i(s,y)dy \right]^2 ds < +\infty \quad \text{a.e.}Q \quad \forall t, T : 0 \le t \le T \le \tau \text{ and } \forall i \in [n]. \tag{9}$$

Should Conditions 2 and 3 hold, the conditions of the stochastic Fubini theorem are satisfied, and we can calculate the bond price dynamics. In the first step we use (2):

$$\log(P(t,T)) = -\int_t^T f(0,y)dy - \int_t^T \left[\int_0^T \alpha(s,y)ds \right] dy - \sum_{i=1}^{n} \int_t^T \left[\int_0^T \sigma_i(s,y)dW_i(s) \right] dy = \tag{10}$$

Now let us apply the stochastic Fubini's theorem.

$$= -\int_t^T f(0,y)dy - \int_0^t \left[\int_t^T \alpha(s,y)dy \right] ds - \sum_{i=1}^{n} \int_0^t \left[\int_t^T \sigma_i(s,y)dy \right] dW_i(s) = \tag{11}$$

After equivalent transformations:

$$\begin{aligned} = &-\int_0^T f(0,y)dy - \int_0^t \left[\int_s^T \alpha(s,y)dy \right] ds - \sum_{i=1}^{n} \int_0^t \left[\int_s^T \sigma_i(s,y)dy \right] dW_i(s) \\ &+ \int_0^t f(0,y)dy + \int_0^t \left[\int_s^t \alpha(s,y)dy \right] ds + \sum_{i=1}^{n} \int_0^t \left[\int_s^t \sigma_i(s,y)dy \right] dW_i(s) \end{aligned} \tag{12}$$

Before finishing deriving $\log P(t,T)$, consider expression (6) when integrating both sides

from 0 to t with respect to $y = t$ and applying again the stochastic Fubini's theorem:

$$\int_0^t f(0,y)dy = \int_0^t r(y)dy + \int_0^t \left[\int_0^t \alpha(s,y)ds\right] dy + \sum_{i=1}^n \int_0^t \left[\int_0^t \sigma_i(s,y)dW_i(s)\right] dy \tag{13}$$

$$= \int_0^t r(y)dy + \int_0^t \left[\int_s^t \alpha(s,y)dy\right] ds + \sum_{i=1}^n \int_0^t \left[\int_s^t \sigma_i(s,y)dy\right] dW_i(s). \tag{14}$$

Since $\log P(0,T) = \int_0^T f(0,y)dy$, (12) and (14) yield:

$$\begin{aligned}\log P(t,T) = \log P(0,T) + \int_0^t r(y)dy - \int_0^t \left[\int_s^T \alpha(s,y)ds\right] dy + \\ - \sum_{i=1}^n \int_0^t \left[\int_s^T \sigma_i(s,y)dy\right] dW_i(s).\end{aligned} \tag{15}$$

As mentioned above, among the introduced processes—namely the forward rate f, the bond price P, and the short rate r—only the zero-coupon bond is directly tradable in the market. In what follows, we are interested in identifying a risk-neutral measure under which the money market account $B(t)$ serves as the numéraire, meaning that the discounted bond price $\frac{P(t,T)}{B(t)}$ is a martingale with respect to $t \in [0,T]$, for at least a finite set of maturities $\{T_1, \ldots, T_n\} \subset [0,\tau]$.

The following two notations are introduced to simplify the calculations:

$$a_i(t,T) :\equiv -\int_t^T \sigma_i(t,s)ds \quad \forall i \in [n], \tag{16}$$

$$b(t,T) = -\int_t^T \alpha(t,s)ds + \frac{1}{2}\sum_{i=1}^n a_i^2(t,T). \tag{17}$$

Equation (15) is reformulated using the newly introduced variables, in line with the methodology of [17].

$$\begin{aligned}d\log P(t,T) = \log P(0,t) + \int_0^t [r(s) + b(s,T)]ds - \frac{1}{2}\sum_{i=1}^n \int_0^t a_i^2(s,T)ds + \\ + \sum_{i=1}^n \int_0^t a_i(s,T)dW_i(s) \quad \text{a.e. } Q.\end{aligned} \tag{18}$$

Define $X(t) := \log P(t,T)$, and let us use Ito's lemma to derive the dynamics of $e^{X(t)}$.

$$\begin{aligned}dP(t,T) = de^{X(t)} &= e^{X(t)}dX(t) + e^{X(t)}d\langle X(t)\rangle \\ &= P(t,T)\left[[r(t) + b(t,T)]\,dt + \sum_{i=1}^n a_i(t,T)dW_i(t)\right] \quad \text{a.e. } Q\end{aligned} \tag{19}$$

We can now determine $\log \frac{P(t,T)}{B(t)}$.

$$\log \frac{P(t,T)}{B(t)} = \log \frac{P(0,T)}{B(0)} + \int_0^t b(s,T)ds - \frac{1}{2}\sum_{i=1}^{n}\int_0^t a_i^2(s,T)ds + \\ + \sum_{i=1}^{n}\int_0^t a_i(s,T)dW_i(s) \quad \text{a.e. } Q \tag{20}$$

In order for $\frac{P(t,T)}{B(t)}$ to be a martingale in t for a finite set of maturities after performing a change of measure to a risk-neutral one, we need the drift term to be equal to zero.

Definition 1.9 (Condition 4)**.** Let $\gamma(\omega, t, T_1, \ldots, T_n)$ be an n dimensional adapted process with $\gamma_i(t) : \Omega \times [0, S_1] \to \mathbb{R}$ coordinates a.e. Q for the set of $T_1, \ldots, T_n$ fixed maturities in $[0, \tau]$. Assume that the following conditions are satisfied for all $i \in [n]$.

- $\int_0^{T_1} \gamma_i^2(s) < \infty \quad$ a.e. Q
- $\mathbb{E}\left[\exp\{\sum_{i=1}^n \int_0^{T_1} \gamma_i(s)dW_i(s) - \frac{1}{2}\sum_{i=1}^n \int_0^{T_1} \gamma_i^2(s)ds\}\right] = 1$
- $\mathbb{E}\left[\exp\{\sum_{i=1}^n \int_0^{T_1} [a_i(s,y) + \gamma_i(s)]dW_i(s) - \frac{1}{2}\sum_{i=1}^n \int_0^{T_1} [a_i(s,y) + \gamma_i(s)ds]^2\}\right] = 1$ for $y \in \{T_1, \ldots, T_n\}$
- (Drift cancellation)

$$\begin{bmatrix} b(t,T_1) \\ \vdots \\ b(t,T_n) \end{bmatrix} - \begin{bmatrix} a_1(t,T_1) & \ldots & a_n(t,T_1) \\ & \vdots & \\ a_1(t,T_n) & \ldots & a_n(t,T_n) \end{bmatrix} \begin{bmatrix} \gamma_1(t) \\ \vdots \\ \gamma_n(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{a.e. } \lambda \times Q; \tag{21}$$

Following [17] further, we can then have the following proposition:

Proposition 1.10 (The existence and uniqueness of the martingale measure on a finite number of bonds)**.** Consider $[\alpha_1(\cdot, T_1), \ldots, \alpha_n(., T_n)]$ and $[\sigma(\cdot, T_1), \ldots, \sigma(., T_n)]$ drift and volatility vectors. If Conditions 1-3 hold for all $i \in [n]$, then Condition 4 is equivalent to the existence of a risk neutral probability measure $Q_{T_1,\ldots,T_n}$, under which the discounted bond prices with maturities $T_1, \ldots, T_n$ are all $\mathcal{F}$ – *martingales*.

Proof: We outline a sketch of the proof in one direction. Conditions 1-3 and the first assumption of Condition 4 yield that $\mathcal{E}_t(\gamma)$ is well defined, the second ensures applicability of Girsanov's theorem, allowing the appropriate change of measure by the Radon-Nykodim derivative: $\frac{Q_{T_1,\ldots,T_n}}{dQ} = \exp\{\sum_{i=1}^n \int_0^{T_1} \gamma_i(s)dW_i(s) - \frac{1}{2}\sum_{i=1}^n \int_0^{T_1} \gamma_i^2(s)ds\}$. The drift cancellation assumption

achieves that the drift term equals to zero a.e.$\lambda \times Q$.
The other direction of the proof can be done similarly, reversing the direction of the measure change. □

Remark 1.11. We note that the coordinates of γ can be interpreted as the market price of risk processes. As we shall see later, γ is not directly needed for derivative pricing in the framework.

Uniqueness is warranted by another condition, as it is written in 1.13.

Definition 1.12 (Condition 5)**.** In Condition 4 assume that the matrix $\Big((a_j(t,T_i)_{ij})\Big)$ is nonsingular a.e. $\lambda \times Q$.

Proposition 1.13 (The existence and uniqueness of the martingale measure on a finite number of bonds)**.** Consider $[\alpha_1(\cdot,T_1),\dots,\alpha_n(.,T_n)]$ and $[\sigma(\cdot,T_1),\dots,\sigma(.,T_n)]$ drift and volatility vectors. If Conditions 1-4 hold for all $i \in [n]$, then Condition 5 is equivalent to the uniqueness of the risk neutral probability measure in the previous proposition.

Now we are looking to ensure the martingale property for all discounted zero coupon bond prices by removing dependency of the market price of risk processes on the previously chosen set of maturities. As in [17], the following proposition addresses this challenge.

Proposition 1.14 (The existence and uniqueness of the martingale measure on all bonds)**.** Consider the families of drift and volatility processes $\{\alpha(\cdot,T) : T \in [0,\tau]\}$ and $\{\sigma(\cdot,T) : T \in [0,\tau]\}$. If Conditions 1-5 hold for all $i \in [n]$, then the 3 statements below are equivalent:

1. The risk neutral measure chosen for any n different maturity is the unique risk neutral measure, under which all discounted bond price process is martingale.
2. For all $i \in [n]$ $\gamma_i(t,T_1,\dots,T_n)$ is equal for $0 \le t \le \min(T_1,T_1')$, where $(T_i)_{i=1,\dots,n}$ and $(T_i')_{i=1,\dots,n}$ are different sets of maturities.
3. $\alpha(t,T) = -\sum_{i=1}^n \sigma_i(t)\Big(\gamma(t,T_1,\dots,T_n) - \int_t^T \sigma_i(t,s)ds\Big)$, with $0 \le t \le T_1 < \cdots < T_n \le \tau$ $\forall i \in [n]$ for any such set of maturities.

Proof: We outline the sketch of the proof.

- (1) $\implies$ (2): By 1.13, the risk neutral measure is unique, hence the market price of risk process is independent from the finite set of maturities used to determine it.

- (2) $\implies$ (1): The possible market price of risk processes depend on a chosen set of maturities and time. If the market price of risk process does not vary with respect to the chosen set of maturities, then the same change of measure applies to all such sets.
- (1) $\implies$ (3): We arrive at (3) by substituting previously computed values into the system of equations of market price of risks.
- (3) $\implies$ (1): Analogously to (2) $\implies$ (1).

□

Remark 1.15 (Different measure - different Brownian motions)**.** By Girsanov's theorem, we have the following Brownian motions under the risk neutral measure:

$$W_i^{Q^*}(t) = W_i(t) - \int_0^t \gamma_i(s)ds \ \forall i \in [n] \tag{22}$$

In Section 2 we will refer to these Brownian motions as $W_i^*(t)$.

Delbean and Schachermayer proved in [11] that the existence of a unique risk neutral measure is equivalent to the completeness of the market, that is, every contingent claim can be replicated by a self-financing strategy. Thus, under the risk neutral measure the following forward curve dynamics apply:

$$df(t,T) = \sum_{i=1}^{n} \left(\sigma_i(t,T) \int_t^T \sigma_i(t,s)ds \right) dt + \sum_{i=1}^{n} \sigma_i(t,T) dW_i^{Q^*}(t). \tag{23}$$

This equation shows that for calibration we solely need to determine the σ functions, the calibration of which is the purpose of this thesis.

Volatility functions

Before proceeding by the introduction to the discrete time version of the model, we highlight some existing volatility parametrizations.

- **Constant normal:**

$$\sigma_i(t,T) \equiv \sigma_i, \quad \text{where } \sigma_i \in \mathbb{R}^2. \tag{24}$$

- **Constant lognormal:** In [17] the authors use this as their first example.

$$\sigma_i(t,T) = \sigma_i \max(f(t,T),\lambda), \quad \text{where } \sigma_i, \lambda \in \mathbb{R}^2 \tag{25}$$

This way the diffusion parameters are proportional to the current forward rates levels.

- **3-factor setup** Resembling the Hull-White model, which can also be fitted into the HJM framework, the following 3-factor setup is suggested in [23]:

$$\sigma(t,T) = \begin{pmatrix} \sigma_1(t,T) = c + \mathbb{P}^{(1)}_{m_1} \exp\Big(-\lambda^1(T-t)\Big) \\ \sigma_2(t,T) = \mathbb{P}^{(2)}_{m_2} \exp\Big(-\lambda^2(T-t)\Big) \\ \sigma_3(t,T) = \mathbb{P}^{(3)}_{m_3} \exp\Big(-\lambda^3(T-t)\Big) \end{pmatrix}, \tag{26}$$

where $\mathbb{P}^{(i)}_{m_i} = a^{(i)}_1 t^1 + \cdots + a^i_{m_i} t^{m_i}$ for i=1,2,3. Ritchken and Chuang also highlighted the necessity of parameter c in [27]. They explain that usually around the shorter maturities a volatility hump can be observed and introducing this parameter enables the calibration to be more accurate.

- **Combination model by Pearson and Zhou:** In [27] the authors suggested a combinations of the ideas above and some additional ones:

$$\begin{gathered} \sigma(\cdot) = \sigma_0 f(t,T)^{\sigma_1} |f(t,T) - c|^{\sigma_2} e^{\lambda(T-t)}, \\ \text{where } \sigma_0, \sigma_1 \in \mathbb{R}^+, c, \sigma_2 \in \mathbb{R}, \text{ with } \lambda \text{ expected to be negative.} \end{gathered} \tag{27}$$

They point out the motivation behind this functional form. They want to ensure nonnegative rates while also capturing the effect of time to maturity by incorporating $T-t$. The exponents σ_1 and σ_2 aim to control skewness and convexity/concavity, respectively.

- **Forward rate spread model by Pearson and Zhou:** Also in their 1999 nonparametric analyisis [27], the authors observed volatility dependence on the forward rate spread and in [35] its incorporation into volatility modeling is suggested.

$$\sigma(\cdot) = g(f(t,T), T-t, f(t,T)^l - f(t,T)^s) \tag{28}$$

where the difference $f(t,T)^l - f(t,T)^s$ denotes a spread between a forward rate with long and short maturity.

We note that in [35] the previous parametrizations are referred to as level models.

- **Stochastic Volatility by CIR process:** For a modest sense of completeness, we present a possible choice for stochastic volatility parametrization. According to the authors of [10], introducing stochastic volatility to the HJM framework enables more accurate valuation of deep ITM and OTM interest rate options and greater flexibility of calibrating skewness of the implied volatility surface. In the framework, the forward rate volatility $\sigma(t,T)$ is modeled as a function of a finite-dimensional process S_t as follows:

$$\sigma(t,T) = \varsigma(S_t)\exp\left(\int_t^T \kappa(u)\,du\right),$$

where ς and κ are deterministic functions. This dependency introduces stochasticity in the volatility via the offset process $S_{t,\tau}$, which aggregates weighted past returns on bond prices. Let $Z_{t,t+\tau}$ denote the logarithm of the discounted bond price process, using the Musiela parametrization [9], where $T = t + \tau$ is used for maturity. Then the SDE determing $S_{t,\tau}$ is as follows:

$$S_{t,\tau} = \int_0^\infty \lambda e^{-\lambda u}(Z_{t,t+\tau} - Z_{t-u,t-u+\tau})du \tag{29}$$

Though authors introduce arbitrary number of offset processes, we restrict this example to one offset process, as the author themselves in their calculations. To characterize the dynamics of the term structure under this stochastic volatility parametrization, two auxiliary processes are defined for $0 \le \tau \in \mathbb{R}$:

$$\xi_{t,\tau} := \int_0^t \sigma^2(s,t+\tau)\,ds, \quad \xi_t := \xi_{t,0} \tag{30}$$

$$\zeta_t = \int_0^t \left(\sigma(s,t+\tau)\int_s^{t+\tau}\sigma(s,u)\,du\right)ds + \int_0^t \sigma(s,t+\tau)dW_s, \quad \zeta_t := \zeta_{t,0} \tag{31}$$

With this structure, the forward rate curve at time t is given by:

$$f(t,t+\tau) = f(0,t+\tau) + e^{-\int_t^{t+r}\kappa(u)du}\zeta_t + e^{-\int_t^{t+r}\kappa(u)du}\int_t^{t+\tau} du e^{-\int_t^u \kappa(u)du}\xi_t, \tag{32}$$

where ψ_τ and $\hat{\psi}_\tau$ are deterministic functions depending on the choice of ϕ.

This implies that the entire forward rate curve $\{f(t,t+\tau) : \tau \ge 0\}$ can be expressed in terms of the finite-dimensional vector (ν_t, η_t), provided S_t (which drives $\sigma(t,T)$) is itself Markovian.

Regarding exact function to substitute, the authors suggest constant κ and the following ς:

$$\varsigma(s) = \nu \sqrt{1 + \epsilon s^2} \wedge N. \tag{33}$$

2. Discretized HJM

In this section, we present the discretization scheme adopted to approximate the HJM framework, incorporating several ideas from [16]. We begin by deriving a discrete version of the drift term, along with a martingale correction that is deemed desirable to preserve consistency with the no-arbitrage condition. This is followed by a comparison with the continuous-time model. Finally, we describe the chosen approach for initializing the forward rate curve.

Model implementation

The most straightforward approach to simulating stochastic differential equations is the Euler–Maruyama discretization scheme. However, when examining the behavior of bond prices on a fixed time grid under the risk-neutral measure, it becomes apparent that the discretized bond prices generally fail to satisfy the martingale property. In this subsection, following [16], we provide the discretization details of our implementation. In particular, we derive the discrete drift formula that must be applied in order to preserve the desired martingale property in the discretized model.

Consider a fixed and equidistant time grid $0 = t_0 < t_1 < \cdots < t_n = T^*$, where $t_j - t_{j-1} = \Delta$ for all $j = 1, \ldots, n$. This grid represents the set of possible maturities in the simulated interest rate market. Due to the discretization over a finite set of time points, the simulation is naturally restricted to a finite time horizon. In the following, the notations f, σ, and B refer to the discretized versions of the continuous forward rate, volatility function, and money market account, respectively. The latter objects in the discrete setting are specified as follows:

$$P(t_i, t_j) = \exp\left(-\sum_{l=i}^{j-1} f(t_i, t_l)\Delta\right) \tag{34}$$

$$f(t_i, t_j) = f(t_{i-1}, t_j) + \mu(t_{i-1}, t_j)\Delta + \sum_{k=1}^{d} \sigma_k(t_{i-1}, t_j)\sqrt{\Delta} \tag{35}$$

$$B(t_j) = \exp\left(\sum_{i=0}^{j-1} f(t_i, t_i)\Delta\right) \tag{36}$$

Henceforth, we restrict our analysis to deterministic volatility functions. Under this assumption, we proceed to determine the appropriate discrete drift term by imposing the martingale property on the discounted bond prices.

$$\mathbb{E}\left[\frac{P(t_i,t_j)}{B(t_i)}|\mathcal{F}_{i-1}\right] = \mathbb{E}\left[P(t_i,t_j)\exp\left(-\sum_{k=0}^{i-1} f(t_k,t_k)\Delta\right)|Z_1,\ldots,Z_{i-1}\right] = \tag{37}$$

$$= \frac{P(t_{i-1},t_j)}{B(t_{i-1})} = P(t_{i-1},t_j)\exp\left(-\sum_{k=0}^{i-2} f(t_k,t_k)\Delta\right) \tag{38}$$

Now let us expand the zero-coupon bond price and use measurability to cancel out terms.

$$\exp\{-\sum_{k=0}^{i-1} f(t_k,t_k)\Delta\}\mathbb{E}\left[\exp\left(-\sum_{l=i}^{j-1} f(t_i,t_l)\Delta\right)|Z_1,\ldots,Z_{i-1}\right] =$$

$$= \exp\left(-\sum_{l=i-1}^{j-1} f(t_{i-1},t_l)\Delta\right)\exp\{-\sum_{k=0}^{i-2} f(t_k,t_k)\Delta\} \tag{39}$$

$$\mathbb{E}\left[\exp\left(-\sum_{l=i}^{j-1} f(t_i,t_l)\Delta\right)|Z_1,\ldots,Z_{i-1}\right] = \exp\left(-\sum_{l=i}^{j-1} f(t_{i-1},t_l)\Delta\right) \tag{40}$$

We can now substitute (35).

$$\mathbb{E}\left[\exp\left(-\sum_{l=i}^{j-1}\left(f(t_{i-1},t_l) + \mu(t_{l-1},t_j)\Delta + \sum_{k=1}^{d}\sigma_k(t_{l-1},t_j)\sqrt{\Delta}Z_l\right)\Delta\right)\middle|Z_1,\ldots,Z_{i-1}\right]$$

$$= \exp\left(-\sum_{l=i}^{j-1} f(t_{i-1},t_l)\Delta\right) \tag{41}$$

After once again canceling out terms due to measurability, and moving the drift terms to the right-hand side, we are left with the below equation:

$$\mathbb{E}\left[\exp\left(-\sum_{l=i}^{j-1}\left(\sum_{k=1}^{d}\sigma_k(t_{l-1},t_j)\sqrt{\Delta}\Delta Z_{l-1}\right)\right)|Z_1,\ldots,Z_{i-1}\right] = \exp\left(\sum_{l=i}^{j-1}\left(\mu(t_{l-1},t_j)\Delta\right)\right) \tag{42}$$

We use the fact that the terms on the left-hand side are lognormally distributed and that the factors are independent.

$$\exp\left(\sum_{k=1}^{d}\frac{1}{2}\left(\sum_{l=i}^{j-1}\sigma_k(t_{l-1},t_j)\Delta\right)^2\Delta\right) = \exp\left(\sum_{l=i}^{j-1}\left(\mu(t_{l-1},t_j)\Delta\right)\right) \tag{43}$$

This is satisfied if the following holds:

$$\mu(t_{i-1}, t_j) = \sum_{k=1}^{d} \frac{1}{2} \left(\left(\sum_{l=i}^{j} \sigma_k(t_{i-1}, t_l) \right)^2 - \left(\sum_{l=i}^{j-1} \sigma(t_{i-1}, t_l) \right)^2 \right) =$$
$$= \sum_{k=1}^{d} \left(\frac{1}{2} \sigma_k^2(t_{i-1}, t_j) + \sigma_k(t_{i-1}, t_j) \sum_{l=i}^{j-1} \sigma_k(t_{i-1}, t_l) \right). \tag{44}$$

Remark 2.1. We note that if $\Delta \to 0$, this converges to the value given by the Euler-Maruyama discretization and thus the continuous version. However, our implementation will keep Δ fixed $\frac{1}{4}$, representing a quarterly simulated term structure. This elevates the importance of keeping the martingale property, and refraining from enabling arbitrage through appoximation errors.

We also emphasize that, when targeting at-the-money (ATM) volatility calibration in this thesis, our objective is not the estimation of a continuous volatility surface. Instead, we focus on estimating a finite set of parameters that define the discretized volatility grid over the simulated time horizon.

Moment matching by pathwise adjustments

Given the limitations of computational resources, it is not feasible to work with arbitrarily large sample sizes in Monte Carlo simulations. The finite nature of this limitation inevitably introduces bias, which may compromise the martingale property of the discounted zero-coupon bond prices. To mitigate this effect, we apply moment matching by pathwise adjustments suggested by [16] aimed at better preserving the martingale structure. While the result enforced by this correction is not strictly equivalent to the true martingale property, it significantly improves pricing consistency for fixed cashflow products by reducing Monte Carlo inconsistencies arising from sample finiteness.

Consider a finite number of paths denoted by Ω, where ω denotes the individual paths with $|\Omega| = M$. The expected value in the discretized simulation reduces to a simple averaging.

$$\mathbb{E}\left[\frac{P(t_i, t_j)}{B(t_i)} \right] = \frac{\sum_{\omega \in \Omega} \exp\left(- \sum_{k=0}^{i-1} f(t_k, t_k) \Delta \right) \exp\left(- \sum_{l=i}^{j-1} f(t_i, t_l) \Delta \right)}{M} \stackrel{?}{=} P(t_0, t_j) \tag{45}$$

Let us apply the correction $c_{i,j}$ in an inductive way. Assume that we already ran the simulation and that we have the simulated forward curves f for all paths. f^* denotes the corrected paths.

At t_0 there are no corrections required, hence $f^*(t_0, \cdot) \equiv f(t_0, \cdot)$. The same applies at t_1.

Now assume that you want to modify $f(t_i, t_{j-1})$ to correct the expected value of $\frac{P(t_i,t_j)}{B(t_i)}$. In addition, assume that this expectation was already corrected for all maturities of $i, i+1, \ldots, j-1$. Now we will determine $c_{i,j-1}$, by which $f^*((t_i, t_{j-1}) = f((t_i, t_{j-1}) + c(i, j-1)$. We rearrange (45) and arrive at:

$$c_{i,j-1} = -\frac{1}{dt} \log\left(\frac{\exp\left(\sum_{k=0}^{j-1} f(0,k)dt\right)}{\frac{1}{|\Omega|}\sum_{\omega\in\Omega}\left(\sum_{k=0}^{i-1} f(k,k,\omega) + \sum_{k=i}^{j-1} f(i,k,\omega)\right)} \right). \tag{46}$$

Comparing Caplet prices in the continuous and discretized implementation

To assess whether the continuous and discretized frameworks produce caplet prices of comparable magnitude, we perform a side-by-side evaluation using matching model parameters in both settings. Since our focus lies on referencing compounded overnight rates - as explained in Section 3 - , we define the products below in order to reflect this structure. The highlight the definitions of the interest rate derivatives in scope of this thesis below following [30].

Definition 2.2 (Caplet)**.** A caplet is a European-style interest rate derivative that provides protection against increases in a compounded overnight rate. At time $T + \delta$, the holder receives the following payoff:

$$\text{Payoff} = \delta \cdot \max(L(T, T+\delta) - K,\, 0),$$

where

- δ is the accrual period (e.g., 3 months),
- $L(T, T+\delta)$ is the compounded in-arrears rate observed at the end of the period $[T, T+\delta]$,
- K is the strike rate.

Definition 2.3 (Swaption)**.** A swaption is a European-style option that grants its holder the right, but not the obligation, to enter into a fixed-for-floating interest rate swap at a future date T. In the case of a *payer swaption*, the holder pays fixed and receives a compounded floating rate (e.g., SOFR). The option is exercised (or not) at time T, but the actual cash flows occur later, based on in-arrears floating rates.

If exercised, the swap generates a sequence of cash flows at times $T_1, T_2, \ldots, T_n$, with each

payoff occurring at T_i and given by:

$$\text{Payoff}_i = \delta_i \cdot (L(T_{i-1}, T_i) - K),$$

where

- K is the fixed strike rate of the swap,
- $L(T_{i-1}, T_i)$ is the compounded overnight rate (e.g., SOFR) accrued over $[T_{i-1}, T_i]$, observed at time T_i,
- δ_i is the accrual period length for the ith payment,
- cash flows are only made if the swaption is exercised at time T.

As emphasized earlier, throughout this thesis we assume a uniform accrual period of $\delta = \frac{1}{4}$, corresponding to a quarterly time grid. Within this discretized framework, we approximate the 90-day SOFR rate observed at time t by the spot rate $r(t)$ generated under the HJM model. This simplification enables us to assess the robustness of the chosen time discretization in capturing compounded overnight rates.

Remark 2.4. We acknowledge that this approximation can be refined. A more accurate representation of the 90-day SOFR rate could be obtained by constructing a Brownian bridge between the simulated forward rates and applying daily compounding along the interpolated path. Such an approach would better reflect the realized nature of backward-looking compounded rates. Nonetheless, for the purposes of this section, we proceed with the spot rate approximation. In the following section, we address this limitation.

This means that for caplets, in our implementation of the HJM model, we have the following payoff at time T:

$$\frac{1}{4}\max(r(T) - K, 0).$$

In the continuous implementation of the model, only the simulation of two correlated normal random variables is needed to calculate the price of this derivative at $t = 0$.

Consider a one-factor, constant volatility setup under the risk neutral measure:

$$\begin{aligned} &d_t f(t,T) = \sigma^2(T-t)dt + \sigma dW^*(t) \\ &f(t,T) = f(0,T) + \sigma^2 t(T - \frac{t}{2}) + \sigma W^*(t) \implies r(T) = f(0,T) + \sigma^2\frac{T^2}{2} + \sigma W^*(T) \\ &B(T) = e^{\int_0^T r(s)ds} = e^{\int_0^T f(0,s)ds + \frac{\sigma^2 T^3}{6} + \sigma\int_0^T W^*(s)ds} \end{aligned} \tag{47}$$

Now let us determine the price of the above caplet using the martingale pricing formula:

$$\begin{aligned} PV(C) &= \frac{1}{4}\mathbb{E}_{P^*}\left[\frac{C(T)}{B(T)}\right] = \frac{1}{4}\mathbb{E}_{P^*}\left[\frac{(r(T)-K)^+}{e^{\int_0^T f(0,s)ds + \frac{\sigma^2T^3}{6} + \sigma\int_0^T W^*(s)ds}}\right] \\ &= \frac{1}{4}\mathbb{E}_{P^*}\left[\frac{(f(0,T) + \sigma^2\frac{T^2}{2} + \sigma W^*(T) - K)^+}{e^{\int_0^T f(0,s)ds + \frac{\sigma^2T^3}{6} + \sigma\int_0^T W^*(s)ds}}\right] \end{aligned} \tag{48}$$

This can be calculated via Monte Carlo-simulation. Two standard normal random variables need to be sampled in order to perform this simulation: $\frac{W^*(T)}{\sqrt{T}}$ and the standardized version of the time integral of the Brownian motion in the exponential function in $B(T)$. We apply the stochastic Fubini's theorem to gain more information about the latter.

$$\int_0^T W^*(s)ds = \int_0^t\int_0^s dW(u)ds = \int_0^t\int_u^t dsdW(u) = \int_0^t (t-u)dW(u) \tag{49}$$

This is a Wiener-integral of a bounded and deterministic real valued function, hence the centered normality. The next step is to calculate the variance. To achieve this, once again, we rely on the stochastic Fubini's theorem.

$$\begin{aligned} \mathbb{E}\left[(\int_0^t W(s)ds)^2\right] &= \mathbb{E}\left[\int_0^t\int_0^t W(u)W(s)dsdu\right] \\ &= \int_0^t\int_0^t \mathbb{E}[W(u)W(s)]dsdu \\ &= \int_0^t\int_0^t \min(u,s)dsdu \\ &= \int_0^t\int_u^t udsdu + \int_0^t\int_0^u sdsdu \\ &= \int_0^t (ut - u^2)du + \int_0^t \frac{u^2}{2}du \\ &= \frac{t^3}{2} + \frac{-t^3}{3} + \frac{t^3}{6} = \frac{t^3}{3} \end{aligned} \tag{50}$$

We have shown that $\int_0^T W^*(s)ds \sim \mathcal{N}(0, \frac{T^3}{3})$. Since $\frac{\int_0^T W^*(s)ds}{\sqrt{\frac{T^3}{3}}} \sim \mathcal{N}(0,1)$ and $\frac{W^*(T)}{\sqrt{T}} \sim \mathcal{N}(0,1)$,

the correlation of two centered variables is equal to their covariance.

$$\begin{aligned} Cov\left(\frac{\int_0^T W^*(s)ds}{\sqrt{\frac{T^3}{3}}}, \frac{W^*(T)}{\sqrt{T}}\right) &= \frac{\sqrt{3}}{T^2}\mathbb{E}\left[\int_0^T W^*(s)dsW^*(T)\right] \\ &= \frac{\sqrt{3}}{T^2}\int_0^T \mathbb{E}[W^*(s)W^*(T)]ds \\ &= \frac{\sqrt{3}}{T^2}\int_0^T sds = \frac{\sqrt{3}}{T^2}\frac{T^2}{2} = \frac{\sqrt{3}}{2} \approx 0.867 \end{aligned} \tag{51}$$

Remark 2.5. Interestingly enough, the resulting correlation is time-independent. This means that regardless of the payoff date, the same correlation is to be applied during the Monte Carlo-simulation.

The expected value above reduces to the following calculation:

$$\frac{1}{4}\mathbb{E}_{P^*}\left[\frac{(f(0,T) + \sigma^2\frac{T^2}{2} + \sigma\sqrt{T}Z_1 - K)^+}{e^{\int_0^T f(0,s)ds + \frac{\sigma^2T^3}{6} + \sigma\sqrt{\frac{T^2}{3}}Z_2}}\right], \text{ where } Corr(Z_1, Z_2) = \frac{\sqrt{3}}{2} \text{ and } Z_1, Z_2 \sim \mathcal{N}(0,1). \tag{52}$$

For simplicity and to match the least complex parametrization we will use later on, let us consider a flat yield curve and constant volatility of 50 bps in a single-factor setting. The below table contains price differences in terms of notional for ATM caplets generated with the continuous and the discretized model. We highlight that the purpose of this comparison is not to ensure complete matching between the prices. The lack of grid refinement and also moment matching contributes to the difference. The below table demonstrates that the difference is small in terms of notional, however, is noticeable in relative price terms.

Maturity	Strike & Curve level	L1 error in terms of notional	Absolute error in relative terms
2 year	4%	13bps	19%
2 years	5%	5bps	7%
2 years	6%	23bps	33%

Table 1: Caplet price comparison between discretized and continuous settings

The initial forward curve

In the Heath–Jarrow–Morton (HJM) framework, two primary inputs are required for the pricing of interest rate derivatives: the initial forward rate curve and the volatility structure. For potential volatility parametrizations, we refer the reader to Section 1. In our numerical

experiments, we employ multiple parametrizations for the initial forward curve. While the use of a constant curve would generally be considered an oversimplification—especially in practical, industry-level applications—its inclusion is justified both by precedents in the literature and by our objective to establish a proof of concept for the application of neural networks.

As we gradually move from having the least possible parameters to relatively more complicated models in the numerical experiments, we aim to adopt a forward curve model which is capable of seizing at least some properties of forward curves. Nelson and Siegel in [26] provide the following yield curve parametrization:

$$y(t) = \beta_0 + \beta_1 \frac{1 - e^{-\lambda t}}{\lambda t} + \beta_2 \left(\frac{1 - e^{-\lambda t}}{\lambda t} - e^{-\lambda t} \right). \tag{53}$$

(1) yields that $f(t) = y(t) + t\frac{d}{dt}y(t)$, hence

$$f(t) = \beta_0 + \beta_1 e^{-\lambda t} + \beta_2 \lambda t e^{-\lambda t}, \tag{54}$$

where

- β_0 controls long-term level;
- β_1 controls short-term behavior;
- β_2 controls curvature;
- λ controls decay speed.

As discussed in [32], the sole level parameter in the Nelson–Siegel model is primarily associated with the behavior of long maturities, rendering the model incapable of capturing a potential second ”hump” in the yield curve. The short-term level is governed by the sum $\beta_0 + \beta_1$. To address this limitation, Svensson proposed the introduction of a second curvature factor [31], while Björk and Christensen [8] extended the original Nelson–Siegel framework by incorporating a second slope factor. Considering the inherent flexibility of the Nelson–Siegel model and the exponentially increasing data requirements associated with introducing additional curve factors, we opt not to extend the initial forward curve parametrization beyond the standard three-factor Nelson–Siegel specification. The adequacy of this flexibility is demonstrated in the figures presented below with base parameters: $\beta_0 = 0.02, \beta_1 = -0.01, \beta_2 = 0.02$, and $\lambda = 0.5$. The figures were generated by a code written in Python.

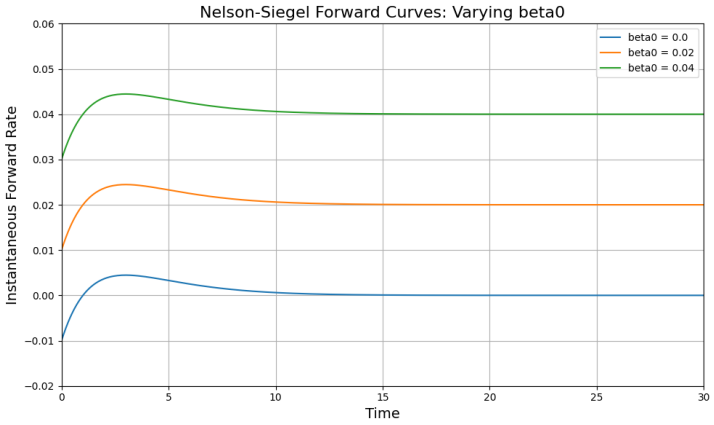

Figure 1: Effect of changing β_0 ceteris paribus

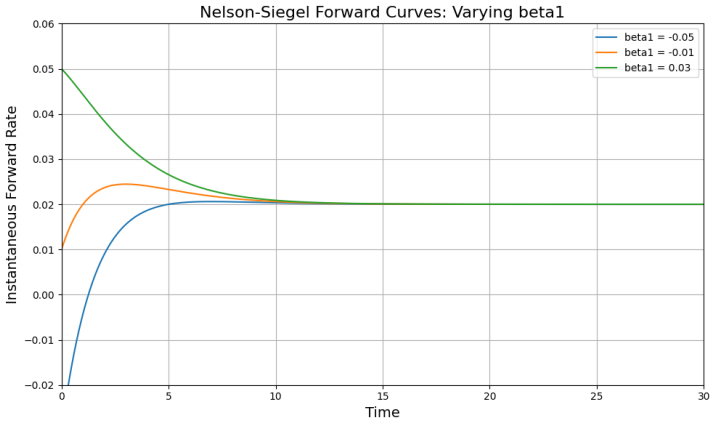

Figure 2: Effect of changing β_1 ceteris paribus

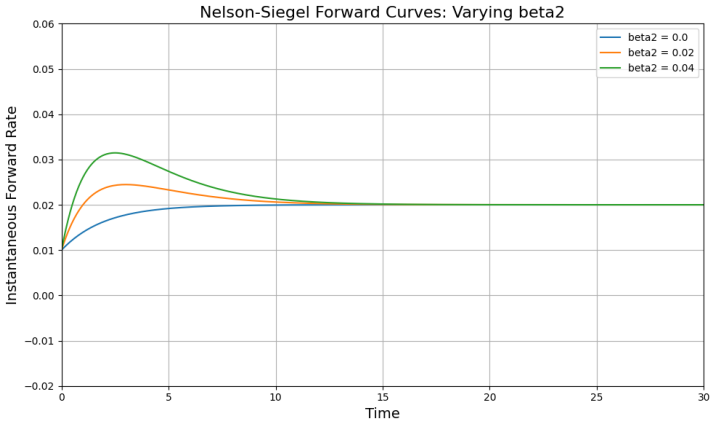

Figure 3: Effect of changing β_2 ceteris paribus

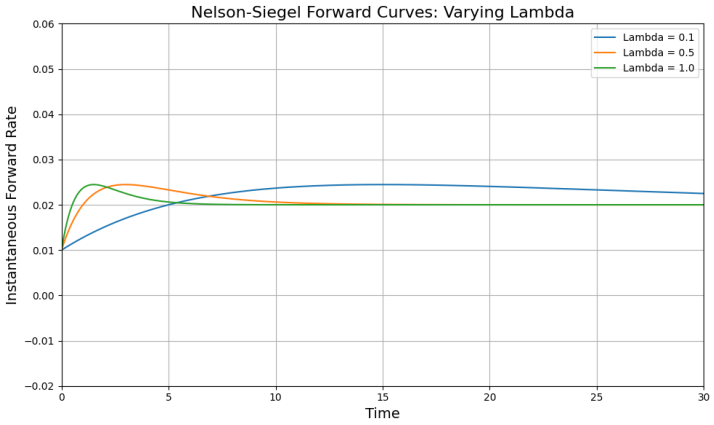

Figure 4: Effect of changing λ ceteris paribus

3. LIBOR transition and new term structures

In this section we present a concise overview about the historical calculation of LIBOR (London Inter-bank Offered Rate) and discuss some of the key motivations regarding the regulatory push towards an alternative benchmark rate. Similarly, the computation of the new term structure rates is also provided, just as our approximation approach to model the 90-day SOFR (Secured Overnight Financing Rate). Although we deem the historical background to be of limited relevance with respect to the ATM calibration process, due the official cessation of LIBOR publication on 30 September 2024 we find capturing this transition to be timely and appropriate in our thesis.

LIBOR - Calculation and Decline

As the Federal Reserve Bank of New York points out in [2], LIBOR was a series of interest rates that intended to reflect the banks' average cost of short-term wholesale unsecured borrowing, that is, the transactions used in its calculation were carried out between financial institutions, not participants in the retail market, and were not backed by collateral. As of the publication of [2] (July 2020), LIBOR was calculated for five currencies and seven tenors. Each business day, a panel of banks for each currency submitted their estimated cost to borrow. After cleaning the data by removing too high and too low percentiles, an average per tenor was calculated.

Since the financial institutions in the panels were considered to be of the safest in the market for each particular currency, LIBOR could serve as the modeling benchmark for the theoretical risk-free rate ever since the second half of the 1980s up until its cessation. With respect to interest rate products in fixed income derivative markets, LIBOR rates were of the most widely used underlying rates. It also acted as a health assessment, indicating the bank' confidence in the financial system. Due to its global importance, it needed to remain stable, credible, and representative for its usage to go undisturbed.

However, this was not always the case. In 2008 during the financial crisis and afterwards, panel banks corroborated in order to keep LIBOR fixings low and understated their borrowing costs, essentially manipulated the markets, leading to the so-called Libor scandal [34]. As a result, the regulatory landscape around Libor submissions was reshaped, banks involved were investigated and fined. The issue of relying on expert judgment persisted: after the 2008 crisis and the intro-

duction of Basel III regulations, banks turned to secured funding like repo markets (repurchase agreement), resulting in too low amount of Libor transactions for keeping it to be reliable. All these served as catalysts for the regulators to turn to new term structure rates.

SOFR - Transition and Calculation

Henceforth, we will restrict our discussion to the US markets, SOFR in particular. In the US, the Alternative Reference Rate Committee was responsible for the development of a new term structure rate [3],[4]. The main criteria for selection were benchmark quality, methodological quality, accountability, and ease of implementation. Out of multiple options, the Secured Overnight Funding Rate (SOFR) was chosen as the new reference rate, supported by the fact of driving the most significant daily volumes in the U.S. money markets. It measures the cost of borrowing cash overnight, colletarized by U.S. Treasury securities, which are generally regarded as one of the safest government bonds in the globally. Based on the FED's User Guide [1], the ISDA's (International Swaps and Derivatives Association) Compound SOFR formula utilizes the below Compound Annualized Interest:

$$\text{Compound Annualized Interest} = \left[\Pi_{b=1}^{T}\left(1 + \frac{r_b \times n_b}{N}\right)\right]\frac{N}{d_c} \tag{55}$$

Where

- T denotes the number of business days in the interest period,
- d_c denotes the number of calendar days in the interest period,
- r_b denotes the interest rate applicable on business day b
- n_b denotes the calender days from and including business day b but not including the following business day,
- N denotes the market convention of quoting the number of days in the year ($N = 360$ for U.S. money markets, this is applied in this thesis as well).

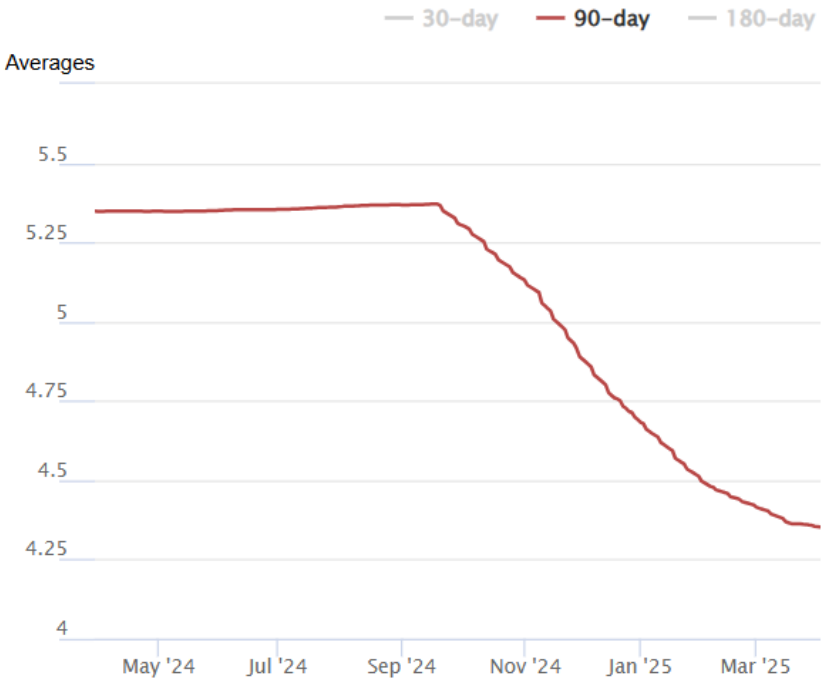

Figure 5: 90-day compounded SOFR from 1 April 2024 to 31 March. [14]

When futures or swaps/swaptions use SOFR, they allow for hedging against future interest rates by construction, since SOFR is an averaged overnight rate. Furthermore, as [1] outlines, averaging greatly contributes to smoothing out day-to-day fluctuations. These fluctuations can result from temporary, not market event related liquidity shocks or calender effects, as End-of-Month effect, that is, higher trading volume on the last day of the month has been growing over time. [13]

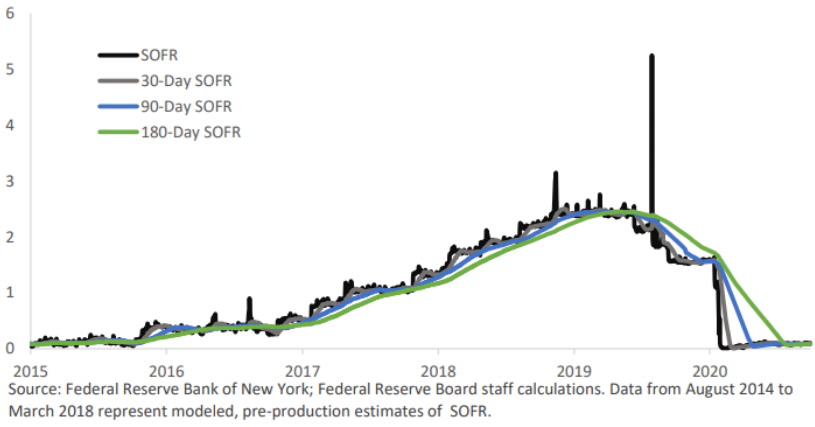

Figure 6: Smoothing effects for averaging SOFR. [1]

Approximating 90-day SOFR

One must keep in mind that our implementation of the HJM framework is restricted to a quarterly time grid. This choice is largely due to limitations in computational power.
The following list contains the approximations considered with the simplifying assumption of a 90-day quarter with notional suppression of ω. We also discuss their adequacy and the differences in price levels.

1. **End-of-period level:** As suggested above, a capacity sparing method to approximate the 90-day (not annualized) SOFR by linearly scaling the first observed overnight rate in the period.

$$\text{90-day SOFR}(t_i) \approx \frac{f(t_{i-1}, t_{i-1})}{4}. \tag{56}$$

2. **Average of Beginning-of and End-of-period levels:** Similarly to the previous idea:

$$\text{90-day SOFR}(t_i) \approx \frac{\frac{1}{2}(f(t_{i-1}, t_{i-1}) + f(t_i, t_i))}{4}. \tag{57}$$

3. **Linear compounding:** In order to account for compounding effects, linear compounding can be applied separately to each half of the accrual period.

$$\text{90-day SOFR}(t_i) \approx (1 + \frac{45}{360} f(t_{i-1}, t_{i-1}))(1 + \frac{45}{360} f(t_i, t_i)) - 1 \tag{58}$$

4. **Daily compounding with Beginning-of and End-of-period levels:** To approximate the effect of daily compounding, as discussed in Section 3, we apply compounding separately to each half of the accrual period. This is done using the forward rates observed at the beginning and end of the period.

$$\text{90-day SOFR}(t_i) \approx (1 + \frac{45}{360} f(t_{i-1}, t_{i-1}))(1 + \frac{45}{360} f(t_i, t_i)) - 1 \tag{59}$$

5. **Daily compounding with linear regression:** Along this line of reasoning, the next step is to capture the daily compounding effect, which necessitates the generation of intermediate terms. This process can be facilitated by regression-based methods, among which we opt

for a linear approach.

$$\text{90-day SOFR}(t_i) \approx \Pi_{i=0}^{89}(1 + f(t_{i-1}, t_{i-1}) + \frac{i}{89} f(t_i, t_i)) - 1 \tag{60}$$

6. **Daily compounding with the sum of linear regression and Brownian bridge of modified diffusion:** Glasserman [16] suggests the application of Brownian bridges for instances where the granularity of the given time grid proves to be insufficient.
   Assume the generating equation (35) and let $Z_1, \ldots, Z_{89}$ be independent standard normal random variables and $Z_0 \equiv 0$. Then we propose:

$$\begin{aligned}\text{90-day SOFR}(t_i) \approx \Pi_{i=0}^{89} \big(1 + f(t_{i-1}, t_{i-1}) + \frac{i}{89} f(t_i, t_i) + \\ + \left(\sum_{k=1}^{d} |\sigma_k(t_{i-1}, t_{i-1})|\right)\left(\sqrt{\frac{1}{360}}\left(\sum_{j=0}^{i} Z_i - \frac{i}{89}\sum_{j=0}^{89} Z_i\right)\right) - 1\end{aligned} \tag{61}$$

Remark 3.1. We note that these approximations easily generalize to accrual periods of different lengths.

We determine the par swap rate, also referred to as the at-the-money (ATM) swap rate, corresponding to ATM swaptions whose underlying swap commences at the option's expiry. The underlying swap structure involves the payment of the time proportional 90-day SOFR(t_i) on a quarterly basis. Given the initial forward rate curve, the time of expiry t_i, and the cash flow dates $t_{i+1}, t_{i+2}, \ldots, t_j$, we can determine the ATM swap rate $S_{t_{i+1},\ldots,t_j}$ from equality of the PV of the fixed and floating legs:

$$S_{t_{i+1},t_j} = \frac{\sum_{k=i}^{j} P(t_0, t_k)(\text{90-day SOFR}(t_k) \text{ approximation at time 0})}{\sum_{k=i}^{j} P(t_0, t_k)} \tag{62}$$

We provide formula for the case of daily compounding with linear regression.

$$S_{t_{i+1},t_j} = \frac{\sum_{k=i}^{j} P(t_0, t_k)(\Pi_{i=0}^{89}(1 + f(t_0, t_{i-1}) + \frac{i}{89} f(t_0, t_i)) - 1)}{\sum_{k=i}^{j} P(t_0, t_k)} \tag{63}$$

Remark 3.2. It is important to note that this formula also determines the relevant ATM swap rate when applying the modified Brownian bridge, due to fact that independent centered variables are used for its construction.

σ	β_0	β_1	β_2	τ	expiry	maturity	Price_1	Price_2	Price_3	Price_4	Price_5	Price_6	Largest %Diff
0.002	0.030	0.007	-0.05	1.8	8	40	93.3942	93.3199	93.6237	93.9227	25.4544	25.5983	268.9835
0.002	0.030	0.007	-0.05	1.8	20	20	76.9755	76.9100	77.1936	77.4730	79.2083	79.0995	2.9818
0.002	0.030	0.007	-0.05	1.8	40	4	20.0365	20.0165	20.0981	20.1820	22.7379	22.7646	64.2968
0.002	0.035	0.005	-0.04	2.0	8	40	95.3185	95.2557	95.6300	95.9990	28.4153	28.3749	238.3233
0.002	0.035	0.005	-0.04	2.0	20	20	74.9864	74.9258	75.2467	75.5632	66.3360	66.3717	13.9099
0.002	0.035	0.005	-0.04	2.0	40	4	19.2030	19.1845	19.2756	19.3657	28.1834	28.2112	47.0654
0.002	0.030	0.007	-0.05	1.8	8	40	243.8582	243.5024	244.3829	245.2497	160.4483	160.4543	52.8694
0.002	0.030	0.007	-0.05	1.8	20	20	191.3797	191.0160	191.8222	192.6173	194.3166	194.2512	1.7279
0.002	0.030	0.007	-0.05	1.8	40	4	48.0635	47.9483	48.1813	48.4115	59.7484	59.8527	24.8274
0.005	0.030	0.007	-0.05	1.8	8	40	222.7999	224.5024	225.4540	226.3953	142.2889	142.6284	9.1084
0.005	0.030	0.007	-0.05	1.8	20	20	174.3537	174.0406	174.8684	175.6862	126.2647	165.9248	5.8830
0.005	0.030	0.007	-0.05	1.8	40	4	46.5240	46.4093	46.6633	46.9144	55.2224	55.1185	18.9898
0.010	0.030	0.007	-0.05	1.8	8	40	486.9394	485.6502	487.6693	489.6611	395.7740	395.9597	23.7224
0.010	0.030	0.007	-0.05	1.8	20	20	400.4847	398.8933	400.9708	403.0182	403.1869	403.1472	1.4723
0.010	0.030	0.007	-0.05	1.8	40	4	98.5902	98.1024	98.7004	99.2912	110.1754	110.3447	12.4771
0.010	0.035	0.005	-0.04	2.0	8	40	453.8027	452.7400	454.9216	457.0761	369.7044	370.0051	23.6329
0.010	0.035	0.005	-0.04	2.0	20	20	359.3362	358.0829	360.1233	362.1607	352.8716	353.4923	2.6325
0.010	0.035	0.005	-0.04	2.0	40	4	93.1476	93.4126	94.0540	94.6940	102.3281	102.5245	9.2084
0.015	0.030	0.007	-0.05	1.8	8	40	723.6785	721.0146	724.4110	727.7664	632.9716	632.9156	14.9863
0.015	0.030	0.007	-0.05	1.8	20	20	570.7527	567.5992	571.0878	574.3863	576.0763	575.2818	1.4935
0.015	0.030	0.007	-0.05	1.8	40	4	151.3285	150.8138	151.2078	151.6026	163.0828	163.2580	7.8378
0.015	0.035	0.005	-0.04	2.0	8	40	706.6872	704.1474	707.9771	711.7593	621.7370	621.8038	14.6442
0.015	0.035	0.005	-0.04	2.0	20	20	533.4446	530.6441	534.0985	537.5214	527.9624	529.0691	1.8106
0.015	0.035	0.005	-0.04	2.0	40	4	136.7455	135.7493	136.8361	137.9074	145.7750	146.2662	7.7473

Table 2: Price results for swaptions with quarterly tenors price results across six SOFR-approximation schemes and percentage deviations.
Expiry and maturity given / quarter.

In Table 2, we present swaption prices obtained under different model parametrizations. For each parameter set, we calculate the percentage difference between the highest and lowest prices across the various approximation schemes.

Our results show that, under the one-factor normal volatility model combined with a Nelson-Siegel initial forward curve, the differences between the pricing approximations are indeed material. Since, heuristically, the approximation scheme based on the Brownian bridge appears the most adequate — considering that it aims to capture the inherent non-smoothness of daily SOFR rates — we choose to generate our training and validation datasets for SOFR swaption pricing using this particular approximation within our pricer. Another decision making factor is the price level of swaptions with short expiry and maturity, where we expect the effect the volatility to be lessened compared to swaptions with longer expiry. Given simulation results, we opt for the latter method consistently providing lower short expiry ATM prices for shorter maturities.

Finally, with all relevant modeling decisions available, we are able to present the swaption pricing formula within the discretized framework of our choice. Let $PV_{t_0}(S\,W_{t_i,t_j})$ denote the present value at time t_0 of a swaption with expiry t_i, and subsequent underlying swap payment dates $t_{i+1}, t_{i+2}, \ldots, t_j$. Consider a Monte Carlo simulation over N paths. For each path, we evaluate the value of the underlying swap at expiry, based on the simulated instantaneous forward curve starting from the expiry date. We then take the maximum between this value and zero, reflecting the payoff of the swaption, and discount the result back to time zero using the money market account as numeraire under the risk-neutral measure. Finally, averaging the discounted payoffs across all simulated paths yields the present value of the swaption.

$$
\begin{aligned}
PV_{t_0}(S\,W_{t_i,t_j}) = \frac{1}{N}\sum_{y=1}^{N} \exp\left(-\sum_{k=0}^{i-1} f(\omega_y, t_k, t_k)\,\Delta t\right) \times \max\Bigg(\sum_{l=i+1}^{j}\Bigg[\exp\left(-\sum_{m=i}^{l-1} f(\omega_y, t_i, t_m)\,\Delta t\right) \\
\times\Bigg(\prod_{n=0}^{89}\Bigg(1 + f(\omega_y, t_{n-1}, t_{n-1}) + \frac{n}{89}\left(f(\omega_y, t_n, t_n) - f(\omega_y, t_{n-1}, t_{n-1})\right) \\
+ \sum_{k=1}^{d} |\sigma_k(t_{n-1}, t_{n-1})| \sqrt{\frac{1}{360}}\left(\sum_{j=0}^{n} Z_j(\omega_y) - \frac{n}{89}\sum_{j=0}^{89} Z_j(\omega_y)\right)\Bigg)\Bigg) - 1\Bigg] - S_{t_i,t_j} * dt,\ 0\Bigg) \\
\times \text{Notional} \qquad (64)
\end{aligned}
$$

4. Deep neural networks

Deep neural networks

Deep neural is a subset of machine learning that utilizes multilayered neural networks to model complex patterns in data. In this section, following [24], we provide a concise overview on the particular type of deep neural networks implemented in this thesis, while not neglecting the opportunity to discuss their applicability in ATM calibration.

There are several approaches to ATM calibration for HJM models, as the literature over the years has introduced numerous methods based on gradient descent and Gauss–Newton techniques. A notable example that combines features of both is the Levenberg–Marquardt algorithm [7]. However, a major bottleneck arises when these algorithms are applied in an online calibration setting, as they require re-running Monte Carlo simulations and re-pricing the relevant calibration instruments at each iteration. Furthermore, since these algorithms are convex optimization methods, multiple initializations are necessary to mitigate the risk of convergence to local minima, further increasing computational cost.

We are, however, prepared to incur this cost — preferably by paying it up front. As discussed in [21], training deep neural networks is computationally expensive; however, model evaluation typically requires only a fraction of a second. Effective and accurate online calibration could provide significant industrial advantages, such as an increased potential for identifying arbitrage opportunities in fixed income markets and the ability to deliver faster pricing in over-the-counter markets for market makers.

Neural networks generally consist multiple layers of interconnected nodes (neurons) that process the data, each transforms it, capturing certain levels of abstraction. In our implementation Feedforward Neural Networks are in scope where data flows without cycles from the input to the output layer.

- The input layer recieves the raw data, in our case it is information about the initial forward curve and ATM swaption prices for several expiry and maturity.
- In the hidden layers multiple nodes/neurons perform certain calculation. Each recieves outputs of the nodes in the previous layer, calculates a linear combination of them, applies and activation function and adjust for bias. All parameters of the calculations are up to training via backpropogation.

- Finally, the output layer performs a linear transformation on one last occasion and thus produces the final prediction of the model.

The network architecture is fully connected, meaning each node in a given layer has weighted connections to all nodes in the subsequent layer.

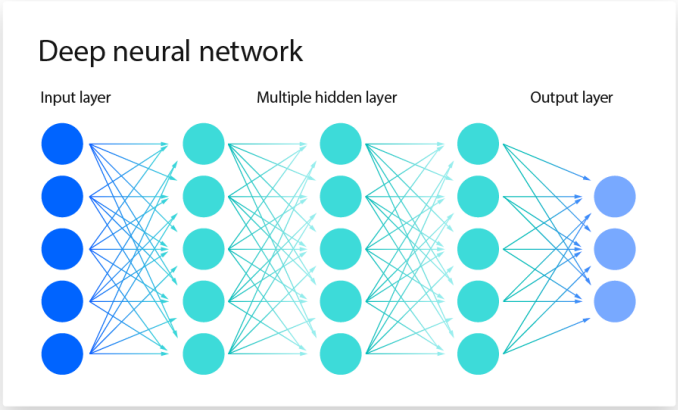

Figure 7: Architecture of a deep artificial neural network [22]

After the model generates a prediction, a loss function is computed to quantify the prediction error. The backpropagation algorithm is then applied to determine the contribution of each weight to this error by calculating gradients. These gradients are used to update the weights using a gradient descent-based optimization method — in our case, the Adam optimizer. We choose Adam for its computational efficiency and built-in bias correction [25], which can potentially offer stable training and potentially better handling of Monte Carlo noise, particularly in high-volatility regions of the training grid. The weight update process is performed once per epoch, where an epoch refers to one complete pass of the entire training dataset through the network. [12]

Although arbitrary loss functions can be defined, we adopt the Sum of Squared Error (SSE) and the Sum of Squared Relative Error (SSRE), both supplemented with L2 regularization to prevent overfitting, a concept initially introduced to statistical modeling in [18]. For activation, we choose the Swish function, defined as $\frac{x}{1+e^x}$, due to its smooth, non-monotonic nature and almost always non-zero derivative, which enables stable and effective gradient-based optimization. [28]

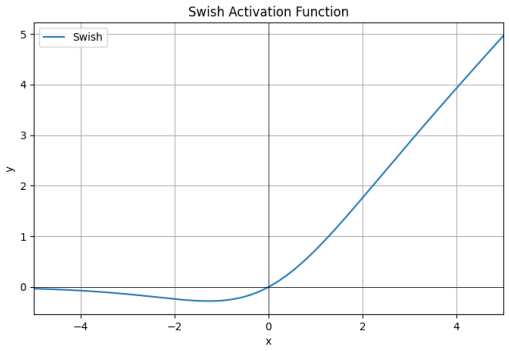

Figure 8: Swish activation function

Just like Horvath et al. in [21], we also refer to the universal approximation theorems developed by Hornik, Stinchcombe, and White [19][20], which establish that feedforward neural networks with a single hidden layer containing a finite number of neurons can approximate any continuous functions on compact subsets of $\mathbb{R}^n$, provided that suitable activation functions are used. These results greatly support the applicability of neural networks as general function approximators.

5. Numerical Expreriments - Calibration for ATM swaptions

In this section, we present the calibration results obtained from our numerical experiments. While our experiments are partly inspired by the work of Horvath et al. [21], our approach fundamentally differs in its objective. Instead of learning the mapping from model parameters to implied volatilities or option prices, as in their framework, we pursue the inverse problem: using current market data—including the full forward rate curve and prevailing at-the-money (ATM) swaption prices—we train neural networks to infer the corresponding volatility levels. The simulations are conducted on a quarterly time grid spanning a 30-year horizon. For each combination in the Cartesian product of the parameter grids defining the volatility functions and the initial forward curves, we price a predefined set of swaptions covering the maturity grid: 1Y1Y, 1Y2Y, 1Y5Y, 1Y10Y, 1Y20Y, 2Y1Y, 2Y2Y, 2Y5Y, 2Y10Y, 2Y20Y, 5Y1Y, 5Y2Y, 5Y5Y, 5Y10Y, 5Y20Y, 10Y1Y, 10Y2Y, 10Y5Y, 10Y10Y, 10Y20Y, 15Y1Y, 15Y5Y, 15Y10Y, 20Y5Y, 20Y10Y. Each underlying interest rate swap is structured to pay quarterly based on the 90-day compounded SOFR, with the first cash flow occurring one quarter after the swaption's expiry date. The 90-day comopunded SOFR is approximated via the modified Brownian bridge approach. We reiterate that our primary objective is to assess whether neural networks can effectively learn the volatility parameters of the underlying HJM models based solely on data typically available in live calibration scenarios—namely, the current forward rate curve and prevailing market prices—even in the presence of increased Monte Carlo simulation noise at higher volatility levels within the training dataset.

Constant volatility, flat initial forward curve

As this calibration setting involves the smallest number of parameters, it is expected to yield the highest prediction accuracy among the tested configurations. The training dataset was generated as the Cartesian product of the volatility grid and the flat initial forward curve levels respectively

$$[0.001, 0.002, 0.003, \ldots, 0.020] \times [1\%, 1.5\%, 2\%, \ldots, 7\%].$$

The corresponding validation dataset was constructed from the inner points of this grid, namely:

$$[0.0015, 0.0025, 0.0035, \ldots, 0.0195] \times [1.25\%, 1.75\%, 2.25\%, \ldots, 6.75\%].$$

Additional implementation details are summarized in the table below. The swaption price data

Parameter	Value
Volatility	Constant
Error Metric	SSRE
Epochs	80000
Hidden Layers	4
Neurons / Layer	100
Learning Rate	0.0001
L2 Alpha	0.0001

Table 3: Numerical experiment parameters - 1

used for both training and prediction is standardized using the mean and standard deviation computed from the training set. This normalization procedure ensures that the model focuses on relative variations in price, rather than being influenced by absolute magnitudes such as notional amounts.

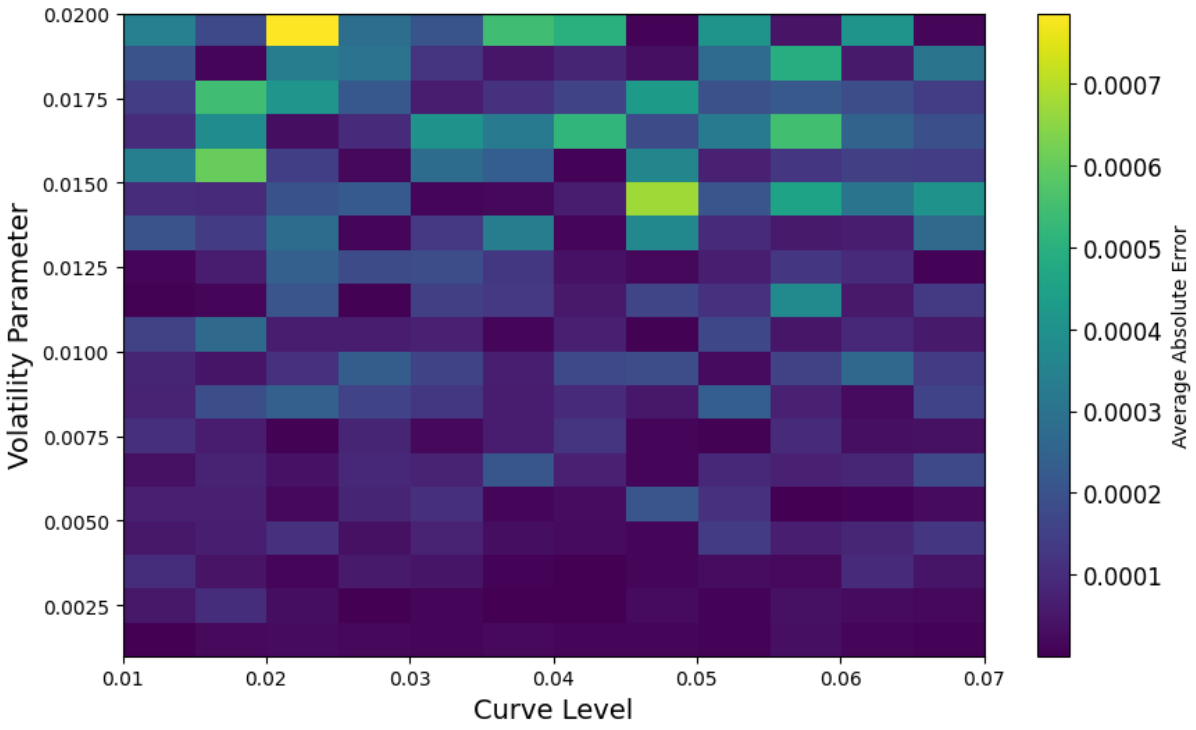

Figure 9: Validation L1 error - one factor, flat initial curve, SSRE

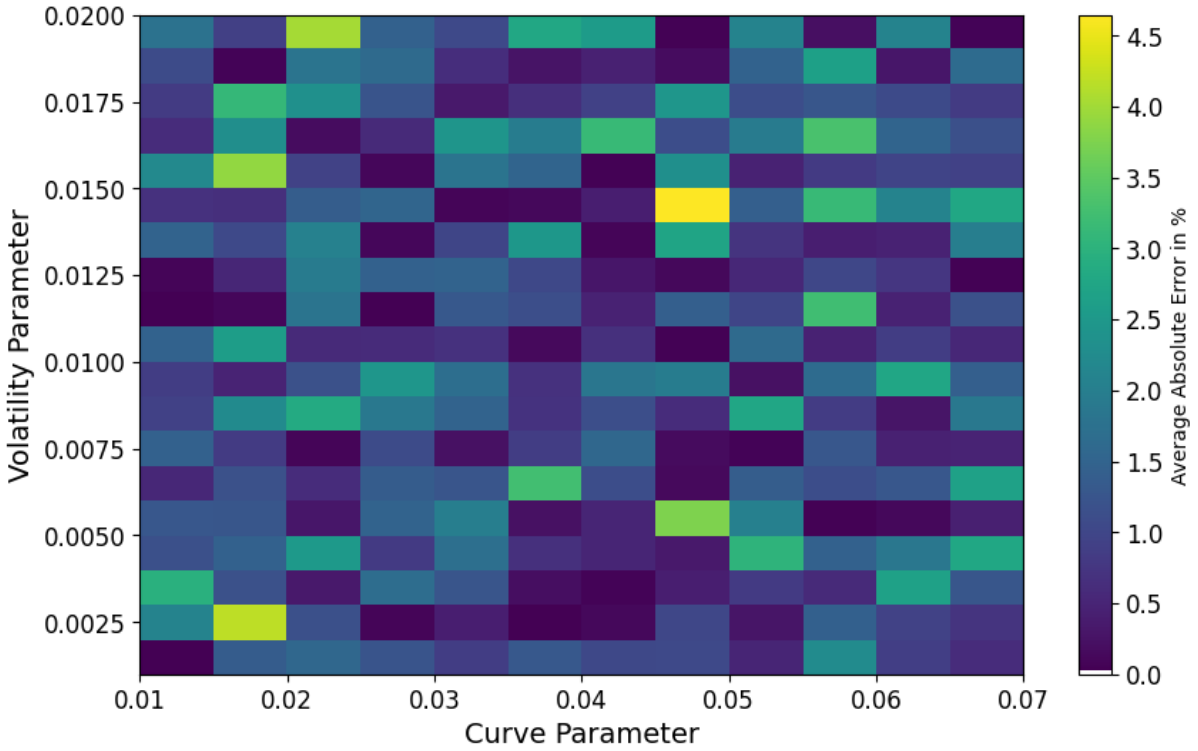

Figure 10: Validation percentage-wise error - one factor, flat initial curve, SSRE

We compare these results to the calibration where we utilize a different error function.

Parameter	**Value**
Volatility	Constant
Error Metric	SSRE
Epochs	80000
Hidden Layers	4
Neurons / Layer	100
Learning Rate	0.0001
L2 Alpha	0.0001

Table 4: Numerical experiment parameters - 2

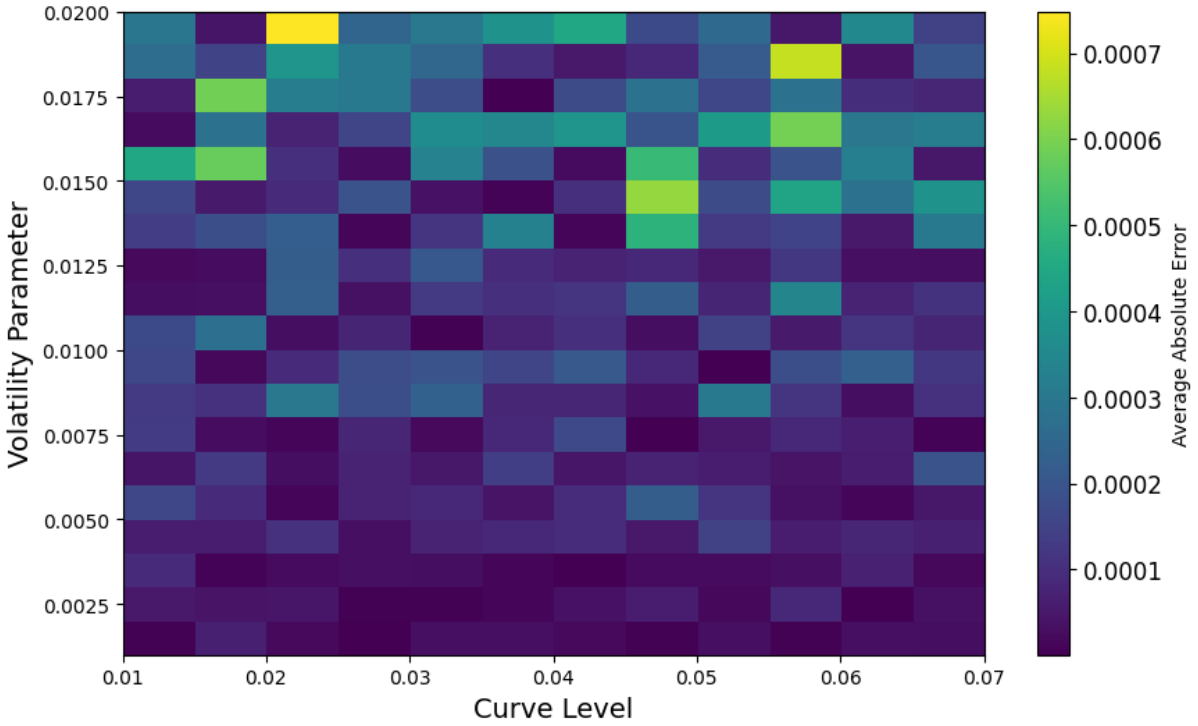

Figure 11: Validation L1 error - one factor, flat initial curve, SSE

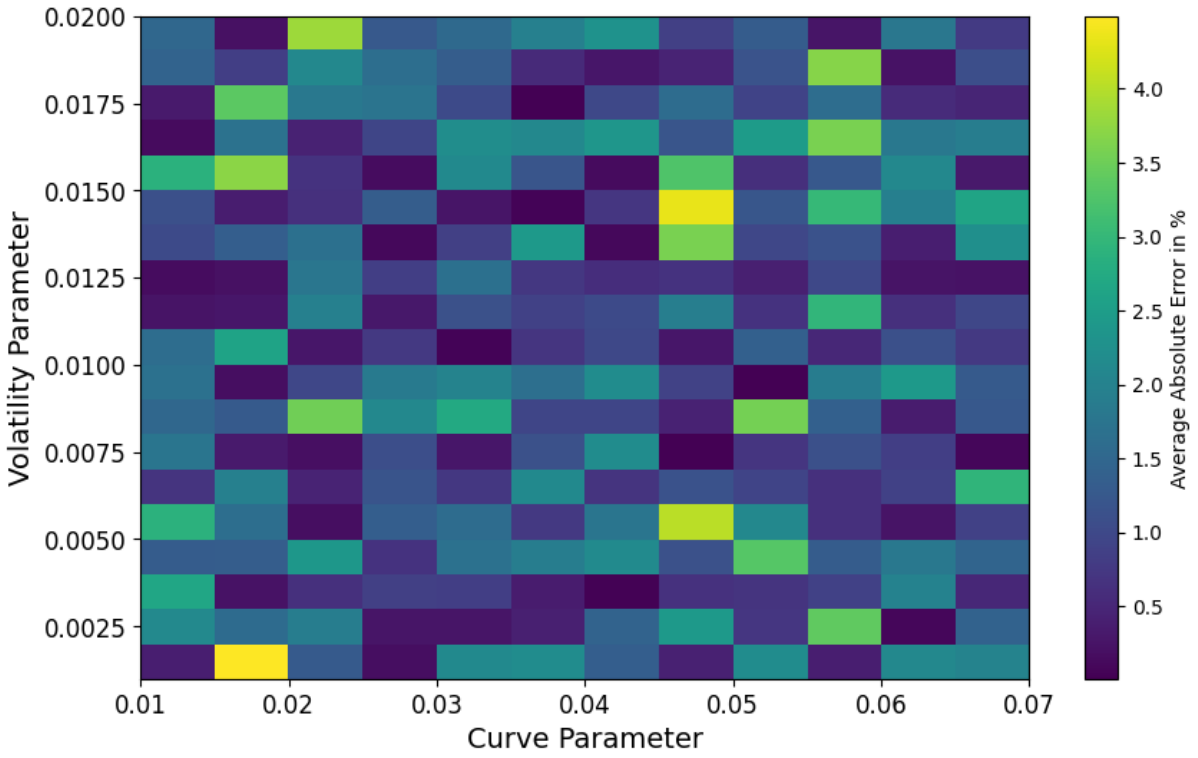

Figure 12: Validation percentage-wise error - one factor, flat initial curve, SSE

Overall, both approaches yield satisfactory results, with absolute percentage errors remain-

ing below 5%. However, since the maximum error is lower by 1 basis points in terms of the L1 norm when using the SSE loss with L2 regularization and also the percentage-wise performance is better in the high-volatility regions, we choose to adopt this configuration in subsequent training experiments.

Regarding the model's extrapolation capability, concerns about overfitting can be reasonably ruled out. By evaluating predictions at volatility levels reaching up to 125% of the maximum value observed in the training set, we assess the model's generalization performance under extrapolated conditions. This evaluation is conducted for three distinct forward curve levels. As illustrated in Figure 13, the model provides consistent and credible predictions across all scenarios. While higher prediction errors are observed at elevated volatility targets, this behavior aligns with prior expectations and reflects the increasing difficulty of accurate estimation in more extreme regimes.

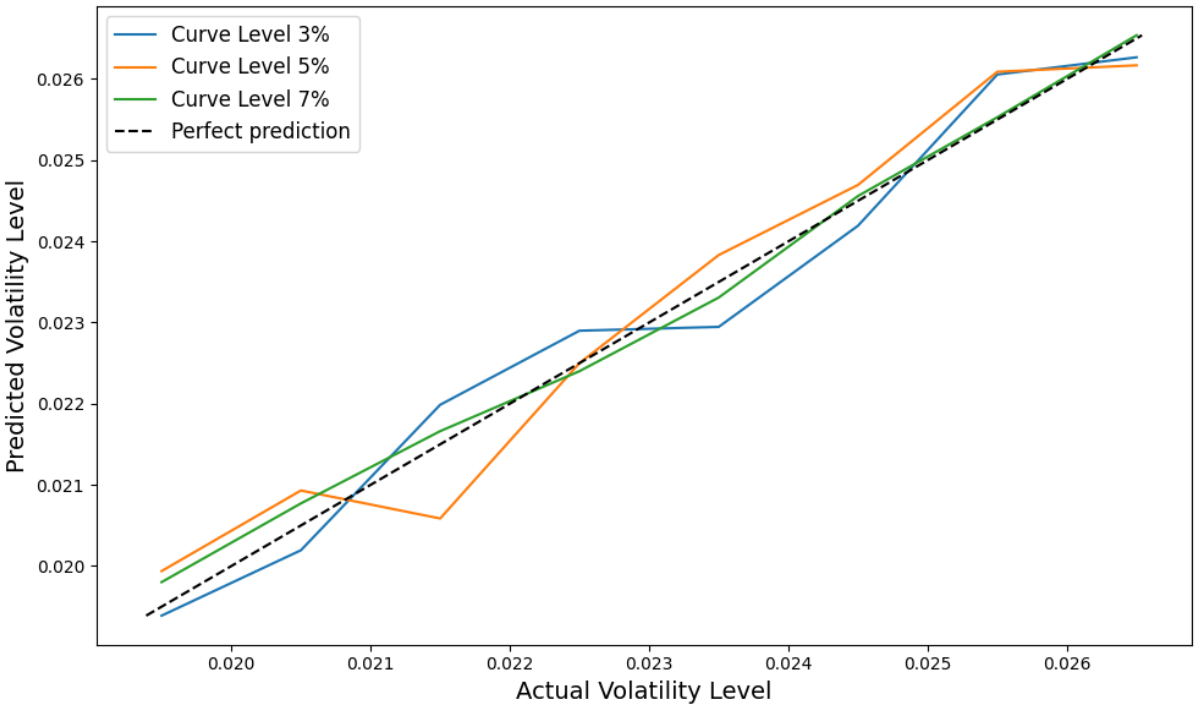

Figure 13: Extrapolation results - one factor, flat initial curve, SSE

Constant volatility, Nelson-Siegel initial forward curve

Given the effective calibration achieved under a flat initial forward curve, we proceed to evaluate the model's predictive performance using the Nelson–Siegel initial forward curve parameterization. For swaptions primarily sensitive to the long end of the forward curve, we anticipate similar price dynamics to the previous case due to the relative flatness of the initial Nelson–Siegel curve at longer maturities. Consequently, our focus lies in assessing whether the neural network can adapt to the distinct pricing behavior induced by variations at the short end of the curve.

Due to the inherent multidimensionality of the forward curve representation, we evaluate calibration accuracy with respect to each curve parameter individually. This is achieved by computing the average of the specified prediction errors within discretized parameter regions, defined by bins over the volatility and a selected curve parameter. The parameter grid employed for training was constructed as a Cartesian product of the following sets:

$$[0.01, 0.03, 0.05, \ldots, 0.019] \times [0.3, 0.5, 0.7] \times [-0.2, 0.0, 0.2] \times [-0.1, 0.0, 0.1],$$

corresponding to the volatility, $\lambda, \beta_0, \beta_1$, and β_2, respectively. The validation grid consists of the inner points of the training grid, defined as:

$$[0.02, 0.04, 0.06, \ldots, 0.018] \times [0.4, 0.6] \times [-0.1, 0.0, 0.1] \times [-0.05, 0.0, 0.05].$$

Parameter	**Value**
Volatility	Constant
Error Metric	SSE
Epochs	80000
Hidden Layers	4
Neurons / Layer	100
Learning Rate	0.0001
L2 Alpha	0.0001

Table 5: Numerical experiment parameters - 3

The results presented below demonstrate that the model is capable of capturing distinct swaption price dynamics at the short end of the time grid. Both calibration and validation are conducted on a sparse parameter grid, constrained to a specific subset of feasible initial forward curves The numerical experiment shows notable flexibility and high accuracy— even when compared to the results of the pervious experiment.. These findings support the model's poten-

tial applicability in real-world, industrial settings.

Remark 5.1. We find it important emphasizing that the increased number of parameters and the corresponding demand for a larger dataset result in noticeably slower training. However, as previously discussed, this initial computational cost is incurred only once during the offline training phase. Once trained, the model enables fast and efficient evaluation, with prediction time scaling linearly with respect to the number of model parameters. This makes the approach promising for real-time calibration to market data, provided the underlying model parameterization offers sufficient flexibility.

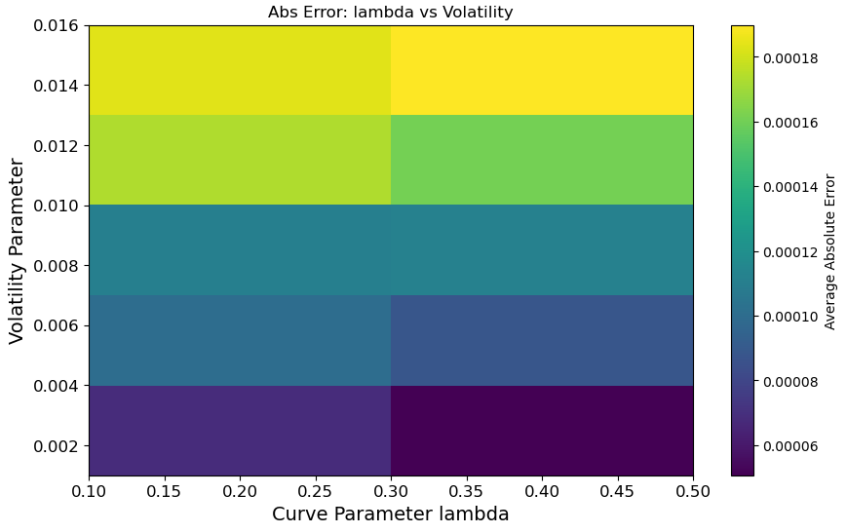

Figure 14: Average L1 Error

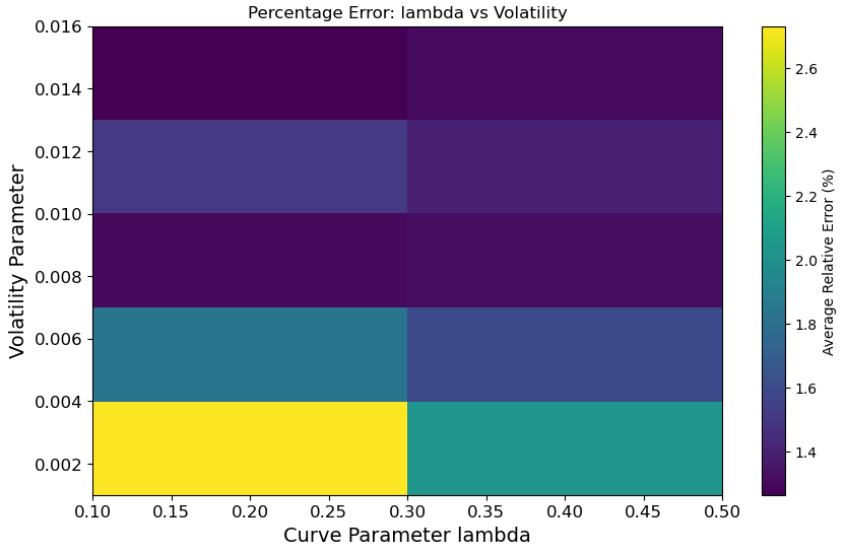

Figure 15: Mean Absolute Percentage Error

Figure 16: Calibration results w.r.t. λ - constant one factor, Nelson-Siegel initial curve, SSE

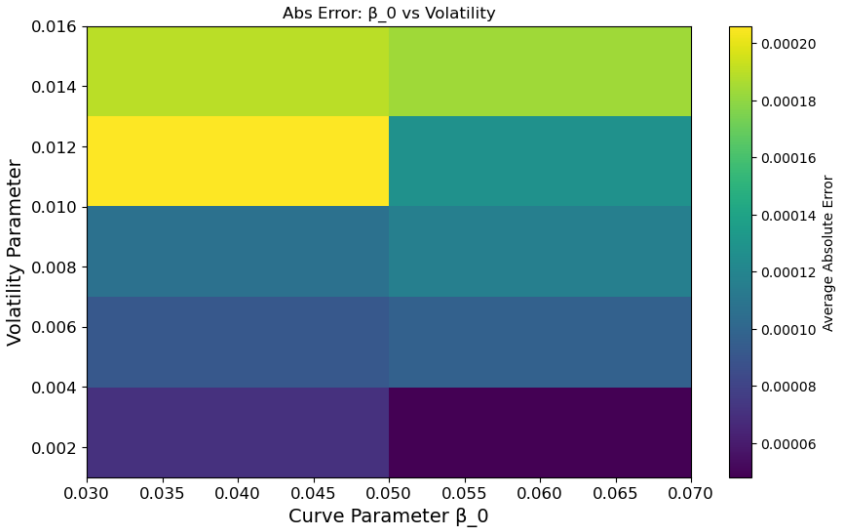

Figure 17: Average L1 Error

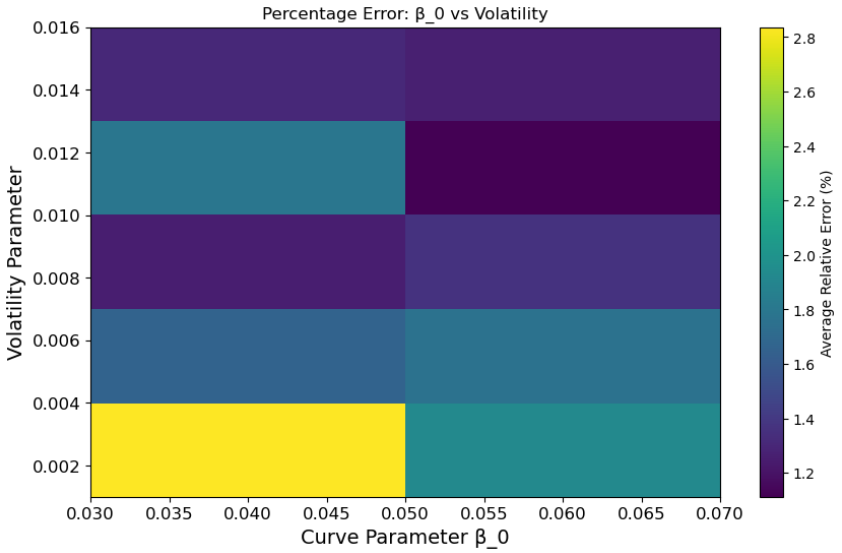

Figure 18: Mean Absolute Percentage Error

Figure 19: Calibration results w.r.t. β_0- constant one factor, Nelson-Siegel initial curve, SSE

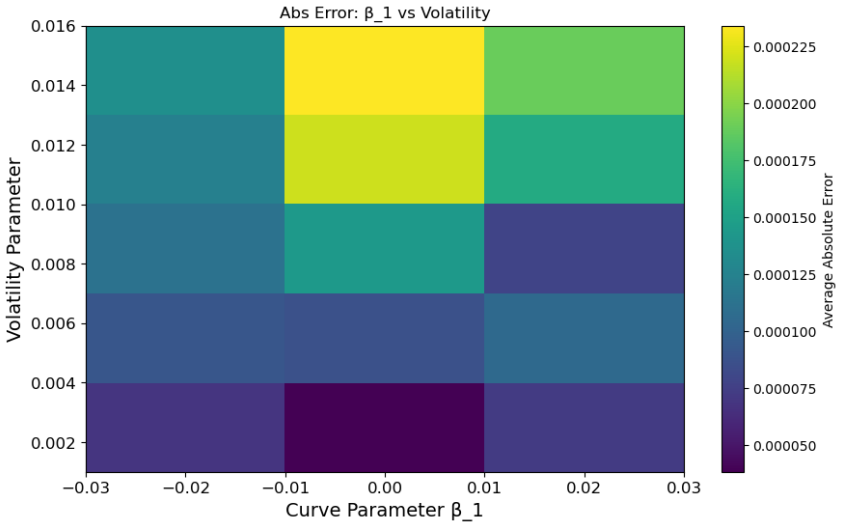

Figure 20: Average L1 Error

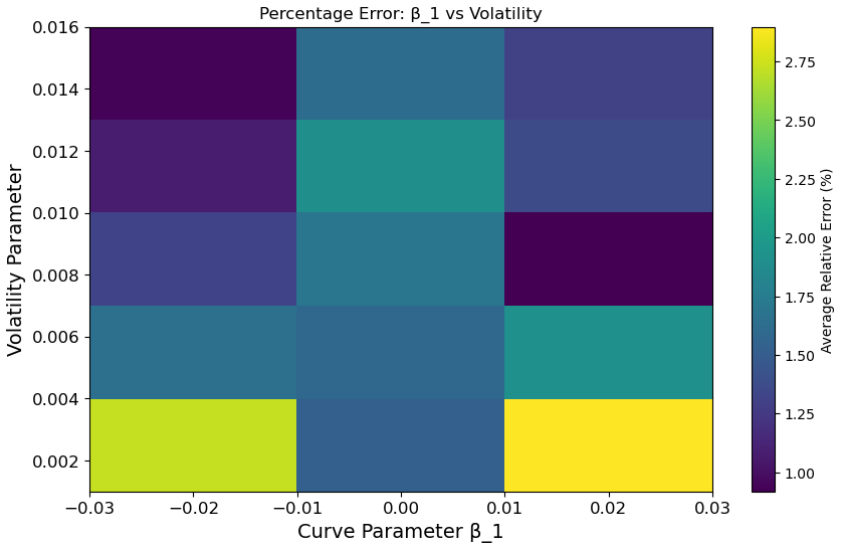

Figure 21: Mean Absolute Percentage Error

Figure 22: Calibration results w.r.t. β_1 - constant one factor, Nelson-Siegel initial curve, SSE

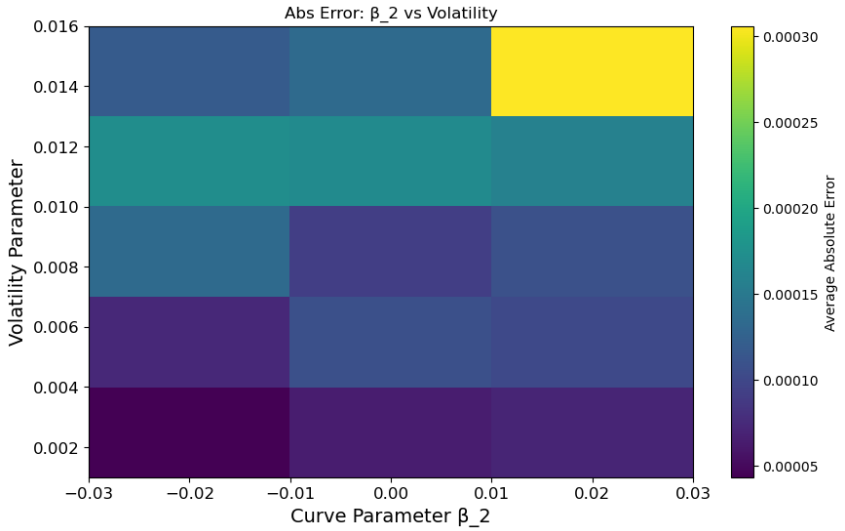

Figure 23: Average L1 Error

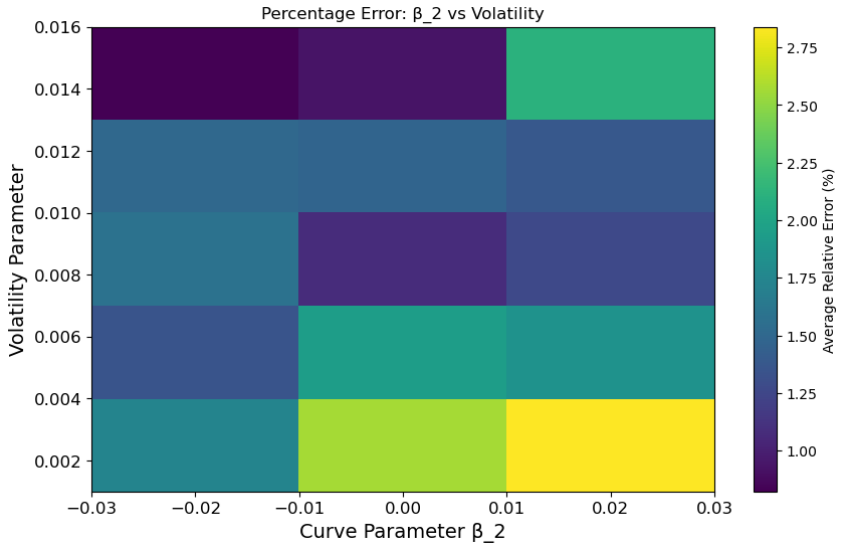

Figure 24: Mean Absolute Percentage Error

Figure 25: Calibration results w.r.t. β_2- constant one factor, Nelson-Siegel initial curve, SSE

We conjecture that, given sufficiently informative initial forward curve data in the training set, neural networks are capable of calibrating ATM volatility with relative ease. Therefore, as a proof of concept, it is sufficient to demonstrate that the network can learn the underlying volatility function under the assumption of a flat initial forward curve. Guided by this rationale, we continue our experimental investigation.

2-factor volatility, flat initial forward curve

We now investigate whether a more layered volatility structure can be effectively captured using neural networks. To extend the initial experiment with a more expressive HJM volatility model, we adopt an alternative parametrization inspired by the structure proposed in [reference]. The aim is to capture a broader range of volatility behavior while keeping the number of parameters reasonably low due to personal computing limitations.

$$\begin{aligned} \sigma_1(t) &= \alpha + \gamma \exp(-\lambda_1(T-t)) \\ \sigma_2(t) &= \gamma \exp(-\lambda_2(T-t)) \end{aligned} \tag{65}$$

We recall Andersen and Piterbarg in [5], where they explain that the correlation structure does not influence swaption data to the extent that it could be calibrated together with ATM volatilities. Hence the assumption, that we have already successfully calibrated parameters λ_1 and λ_2 to a chosen set of spread option for example. Thereby we restrict the calibration to the parameters α and γ.

As we would like to implement similar network as in the previous experiments, first, we try predicting both parameters simultaneously, and we do not succeed.

To address the calibration challenge, we introduce a two-phase learning framework. In the first phase, we train a neural network to predict the constant volatility level, denoted by α, using only the initial curve level and the swaption price map as inputs. In the second phase, a separate neural network is trained to predict the parameter γ, conditional on the previously predicted α as well as on λ_1, λ_2, and the same input features used in the first phase.

Given a set of observed market prices and an initial curve level, we first employ the α-network to produce a prediction α'. This predicted value is then passed to the second network to generate a corresponding estimate γ'. Provided that the first network has been trained to sufficient accuracy, the dominant source of error in a two-step procedure originates from the second network's prediction of γ.

We provide the details and calibration results regarding the first network.

Parameter	Value
Volatility	Constant
Error Metric	SSE
Epochs	10000
Hidden Layers	4
Neurons / Layer	50
Learning Rate	0.000001
L2 Alpha	0.001

Table 6: Numerical experiment parameters - 4 - Phase 1

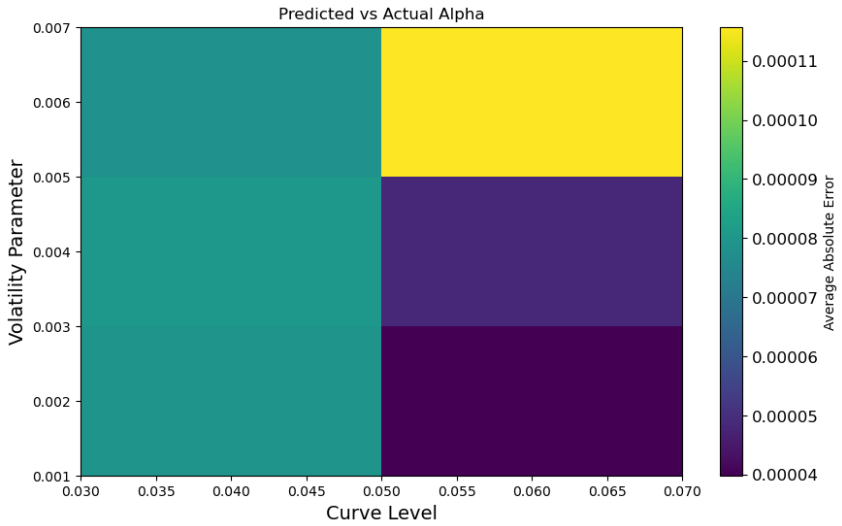

Figure 26: Average L1 Error

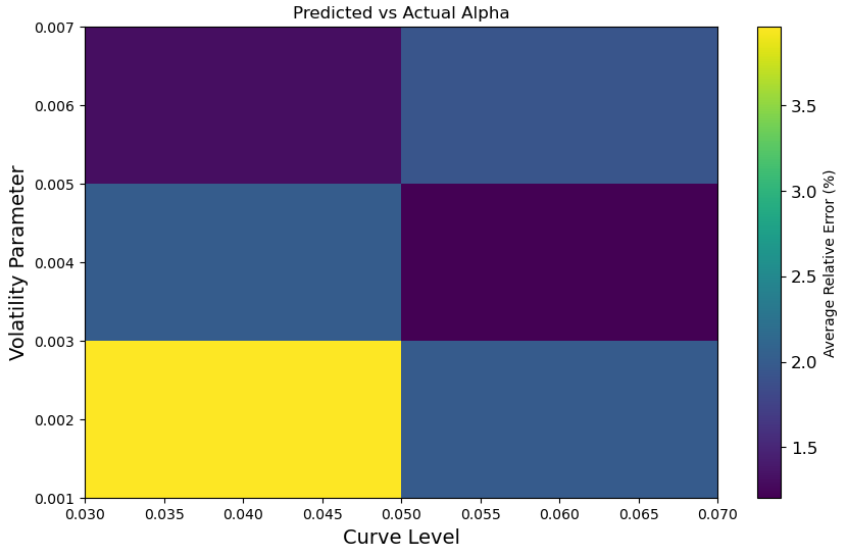

Figure 27: Mean Absolute Percentage Error

Figure 28: Calibration results w.r.t. α - two factor, flat initial curve, SSE

The accuracy of the first network is sufficient to support the testing of the second network's ability to predict γ, despite the compounded noise originating from both the Monte Carlo sim-

ulation and the first-stage predictions. However, despite experimenting with several different approaches like rescaling and feature selection, we were unable to design a network that not only achieves good training accuracy but also performs well during validation. It is worth noting that validation results were similarly unsatisfactory whether we used the predicted or the benchmark α values from the synthetic data.

As for future research directions to overcome this bottleneck, the two-phase calibration approach could be extended either by incorporating additional vanilla instruments or by increasing the size of the training dataset. We are hopeful that such extensions could improve the model's performance and predictive capabilities.

At last, we highlight the potential applicability of the approach in computationally intensive stress testing environments. Given that stress tests are required to balance a number of objectives, as outlined in [6], accuracy requirements are not always stringent. This may increase the appeal of fast calibration techniques using neural networks.

6. Conclusion

Our thesis explored the ATM calibration of HJM interest rate models using deep neural networks. We presented the theoretical foundations of the HJM framework, implemented a discretized version of the model with a focus on preserving the martingale property and theoretical expectations, and discussed the transition from LIBOR to SOFR, which is particularly relevant for the swaptions we chose to price. We also gave a concise overview on neural networks.

Our main contribution lies in demonstrating that deep neural networks can effectively learn the mapping from ATM swaption prices and initial forward curves to the volatility parameters of HJM models. Through a series of numerical experiments—starting from simple constant volatility with flat curves, up to more complex configurations involving Nelson – Siegel curves or multi-factor volatilities — we showed that the proposed neural network models achieve good accuracy, even under limited data and increased simulation noise. For parametrizations that allow distinguishing between global and local parameters within the volatility grid, we additionally proposed a two-phase calibration approach.

These results highlight the potential of neural networks in speeding up calibration tasks and enabling more efficient real-time pricing in practical applications. With proper offline training, the models are able to offer fast and accurate volatility estimates, making them a promising alternative to traditional optimization-based calibration techniques.

We note that due to the overwhelming success of the adaptability and flexibility of the HJM framework, a large number of articles cite it with relatively low degree in freedom of phrasing and notations. We hereby state that all - whether HJM related or otherwise - articles that we used are cited and used appropriately. We also opted for relatively strict follow of the original article. Any potential match in wording or notation with articles not cited are purely the result of the unity around the framework and not of plagiarism.

I would like to express my appreciation for the overall level of challenge presented by the thesis topic, it motivated me to expand my skill set — particularly in the areas of model training and general implementation.

We also note that further research opportunities remain. With sufficient computational resources, calibration of more complex model parametrizations could be explored, and benchmarking against actual market prices could be conducted. We acknowledge, however, that such data and infrastructure are typically accessible only to institutional market participants.

AI Tool Usage Declaration

I, the undersigned, Máté Kurucz, declare that during the preparation of my thesis, I used the AI-based tools listed below for the tasks specified.

Task	**Tool Used**	**Scope of Use**	**Remarks**
Language correctness	GPT	Entire thesis	Spelling check, stylistic review of certain sentences, occasional rephrasing.
Literature research	GPT	Entire thesis	Identifying and selecting relevant literature.

No other AI-based tools were used beyond those listed above.

References

[1] Alternative Reference Rates Committee. *An Updated User's Guide to SOFR*. https://www.newyorkfed.org/medialibrary/Microsites/arrc/files/2021/users-guide-to-sofr2021-update.pdf. Accessed: 2025-04-30. Feb. 2021.

[2] Alternative Reference Rates Committee. *SOFR Starter Kit: Factsheets 1*. https://www.newyorkfed.org/medialibrary/Microsites/arrc/files/2020/ARRC_Factsheet_1.pdf. Federal Reserve Bank of New York. 2020.

[3] Alternative Reference Rates Committee. *SOFR Starter Kit: Factsheets 2*. https://www.newyorkfed.org/medialibrary/Microsites/arrc/files/2020/ARRC_Factsheet_2.pdf. Federal Reserve Bank of New York. 2020.

[4] Alternative Reference Rates Committee. *SOFR Starter Kit: Factsheets 3*. Federal Reserve Bank of New York. 2020.

[5] Leif B. G. Andersen and Vladimir V. Piterbarg. *Interest Rate Modeling. Volume III: Products and Risk Management*. Atlantic Financial Press, 2010. ISBN: 0984422129.

[6] Basel Committee on Banking Supervision. *International Convergence of Capital Measurement and Capital Standards: A Revised Framework*. https://www.bis.org/publ/bcbs155.pdf. Basel II – Comprehensive Version. June 2006. (Visited on 05/13/2025).

[7] Riccardo Beltrami. "Machine learning techniques for financial time series prediction". Master's thesis. Politecnico di Milano, 2010. URL: https://www.politesi.polimi.it/retrieve/a81cb059-7f30-616b-e053-1605fe0a889a/2010_12_Beltrami.PDF.pdf.

[8] Tomas Björk and Bent Jesper Christensen. "Interest rate dynamics and consistent forward rate curves". In: *Mathematical Finance* 9.4 (1999), pp. 323–348.

[9] Andrew Brace, Dariusz Gatarek, and Marek Musiela. "The Market Model of Interest Rate Dynamics". In: *Mathematical Finance* 7.2 (1997), pp. 127–155. DOI: 10.1111/1467-9965.00028.

[10] Carl Chiarella and Oh Kang Kwon. "A Complete Markovian Stochastic Volatility Model in the HJM Framework". In: *Asia-Pacific Financial Markets* 7.4 (Sept. 2000), pp. 321–342. DOI: 10.1023/A:1010017213565.

[11] Freddy Delbaen and Walter Schachermayer. "A General Version of the Fundamental Theorem of Asset Pricing". In: *Mathematische Annalen* 300.1 (1994), pp. 463–520. DOI: 10.1007/BF01450498.

[12] European Information Technologies Certification Academy. *EITC/AI/DLTF Deep Learning with TensorFlow Study Guide*. https://eitca.org/wp-content/uploads/materials/eitca_exam_mat_eitc-ai-dltf.pdf. Accessed: 2025-05-13. 2023.

[13] Federal Reserve Bank of New York. *End-of-Month Liquidity in the Treasury Market*. Accessed: 2024-05-05. Sept. 2024. URL: https://libertystreeteconomics.newyorkfed.org/2024/09/end-of-month-liquidity-in-the-treasury-market/.

[14] Federal Reserve Bank of New York. *Secured Overnight Financing Rate Averages and Index*. https://www.newyorkfed.org/markets/reference-rates/sofr-averages-and-index. Accessed: 2025-04-30. 2025.

[15] Federal Reserve Bank of New York. *SOFR Averages and Index*. https://www.newyorkfed.org/markets/reference-rates/sofr-averages-and-index. Accessed: 2025-04-30. 2020.

[16] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Vol. 53. Stochastic Modelling and Applied Probability. New York: Springer, 2004. DOI: 10.1007/978-0-387-21617-1.

[17] F. E. H. Haug, Robert A. Jarrow, and Donald B. Morton. "Bond Pricing and the Term Structure of Interest Rates: A New Methodology for Contingent Claims Valuation". In: *The Journal of Finance* 47.2 (1992), pp. 519–547. DOI: 10.2307/2951677.

[18] Arthur E. Hoerl and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". In: *Neural Networks* 3.5 (1990), pp. 551–560.

[21] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. "Deep Learning Volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models". In: *arXiv preprint arXiv:1901.09647* (Aug. 2019). Available at https://arxiv.org/abs/1901.09647.

[22] IBM. *What is a Neural Network?* Accessed: 2025-05-11. URL: https://www.ibm.com/think/topics/neural-networks.

[23] Beyna Ingo, Carl Chiarella, and Boda Kang. "Pricing Interest Rate Derivatives in a Multifactor HJM Model with Time". In: 317 (2012). DOI: 10.2139/ssrn.2162748.

[24] Nicos B. Karayiannis. "A brief review of feed-forward neural networks". In: *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering* 50.1 (2006). https://www.researchgate.net/publication/228394623, pp. 1–20.

[25] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014). Published as a conference paper at ICLR 2015. URL: https://arxiv.org/abs/1412.6980.

[26] Charles R. Nelson and Andrew F. Siegel. "Parsimonious modeling of yield curves". In: *The Journal of Business* 60.4 (1987), pp. 473–489.

[27] Neil D. Pearson and Anjun Zhou. *A Nonparametric Analysis of the Forward Rate Volatilities*. Working Paper 99-05. Available at SSRN: https://ssrn.com/abstract=190011 or http://dx.doi.org/10.2139/ssrn.190011. Office for Futures and Options Research, 1999.

[28] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. "Swish: A Self-Gated Activation Function". In: *arXiv preprint arXiv:1710.05941* (2017). URL: https://arxiv.org/abs/1710.05941.

[29] Peter Ritchken and Iyuan Chuang. "Interest Rate Option Pricing with Volatility Humps". In: *Review of Derivatives Research* 3.3 (2000), pp. 237–262. DOI: 10.1023/A:1009690621051.

[30] Thierry Roncalli. *Handbook of Financial Risk Management*. Chapman and Hall/CRC Financial Mathematics Series. Chapman and Hall/CRC, 2020. ISBN: 9781138393906. DOI: DOI:10.1201/9781315144597.

[31] Lars E.O. Svensson. *Estimating and Interpreting Forward Interest Rates: Sweden 1992–1994*. NBER Working Paper Series 4871. National Bureau of Economic Research, 1994. DOI: 10.3386/w4871.

[32] Ranik Raaen Wahlstrøm, Florentina Paraschiv, and Michael Schürle. "A Comparative Analysis of Parsimonious Yield Curve Models with Focus on the Nelson–Siegel, Svensson and Bliss Versions". In: *Computational Economics* 59.4 (2022), pp. 967–1004. DOI: 10.1007/s10614-021-10113-w.

[33] Wikipedia contributors. *LIBOR — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Libor. Accessed: 2025-04-30. 2025.

[34] Wikipedia contributors. *Libor scandal — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Libor_scandal. Accessed: 2025-04-30. 2025.

[35] Anjun Zhou. *Modeling the Volatility of the Heath-Jarrow-Morton Model: A Multi-Factor GARCH Analysis*. Working Paper 00-05. Available at SSRN: https://ssrn.com/abstract=244521 or http://dx.doi.org/10.2139/ssrn.244521. Office for Futures and Options Research, Aug. 2000.

7. Appendix

In the following pages we provide a short summary in Hungarian.
Az alábbi szószedet tartalmazza a legfontosabb angol nyelvű szakszavak magyar nyelvű megfelelőit.

Table 7: Angol–magyar szakszószedet (válogatás)

English	**Magyar**
activation function	aktivációs függvény
bond price	kötvényárfolyam
Brownian motion	Brown-mozgás
calibration	kalibráció
caplet	egyetlen periódusra vonatkozó kamatplafon opció
compounded average interest	kamatos kamattal számított átlagos kamatláb
deep learning	mélytanulás
diversification	diverzifikáció
forward rate	forward kamatláb
initial forward curve	kezdeti forward görbe
interest rate curve	kamatlábgörbe
interest rate derivative	kamatlábra vonatkozó származtatott termék
interest rate model	kamatlábmodell
layer	réteg (neurális hálózatban)
money market account	bankbetét
market price of risk	kockázat piaci ára
neural network	mesterséges neurális hálózat
node	neuron (neurális hálók)
no-arbitrage condition	arbitrázsmentességi feltétel
numerical experiment	numerikus kísérlet
pricing function	árazó függvény
risk-neutral measure	kockázatsemleges mérték
spot rate	rövid kamat
stochastic differential equation	sztochasztikus differenciálegyenlet
swaption	kamatcserére vonatkozó opció
theorem	tétel
time discretization	idődiszkretizálás
training	tanítás (neurális hálózat)
volatility	volatilitás
weighting	súlyozás
yield curve	hozamgörbe
zero-coupon bond	kamatszelvény nélküli kötvény

Magyar nyelvű összefoglaló - HJM modellek kalibrálása mély neurális hálózatok segítségével

A dolgozat célja a Heath–Jarrow–Morton (HJM) kamatlábmodell keretrendszerében alkalmazott volatilitásfüggvények paramétereinek becslése at-the-money kalibráció során, mélytanulási módszerek segítségével. A HJM modell egyik fő előnye a rendkívüli rugalmassága, amely különösen alkalmassá teszi a pénzügyi piacokon kereskedett széleskörű kamatlábra vonatkozó származtatott termékek árazására.

A dolgozat első felében bemutatásra került a HJM modell alapfelépítése, beleértve a kockázatsemleges mérték alatti dinamikát, valamint a volatilitás szerepét a modellben. A dolgozatban több, az irodalomban fellelhető volatilitásparametrizációt mutattunk be – a teljesség igénye nélkül. Ezek közül választottunk ki néhányat a kalibrációs kísérletek során alkalmazott modellekhez. A diszkontált kötvényárak martingál mivolta nem csupán a folytonos modell bemutatásában kapott kiemelt szerepet, hanem az általunk implementált diszkretizált HJM modellben és annak a bemutatásában is hangsúlyt kapott, elősegítve a belső konzisztenciát és ezáltal az arbitrázsmentességet.

A HJM keretrendszer egy szimulációs modellt definiál, így Monte Carlo módszert alkalmazva közelítő árakat számítottunk elsősorban kamatcserére vonatkozó opciókra. Az implementáció Python nyelven történt, ennek segítségével implementáltunk swaption árazót a SOFR referencia kamatot fizető kamatcsere típusú alaptermékek esetén. Miután bemutattuk a LIBOR kivezetésének körülményeit, több approximációt is megfontoltunk a SOFR-re az általunk alkalmazott negyedéves diszkretizáció esetén, melynek eredményképp egy módosított Brown-hidas közelítést alkalmaztunk a kalibrációs kísérletek folyamán.

A dolgozat fontos saját eredménye a HJM modell neurális hálóval történő kalibrációja. Ennek során mesterséges neurális hálókat tanítottunk arra, hogy kezdeti forward görbe és swaption árak alapján visszakövetkeztesse az ismeretlen volatilitásparamétereket. Az árakat szintetikusan generáltuk különböző rácspontjain a paramétertérnek, implicit módon feltételezve a kísérlet folyamán, hogy az adott parametrizáció kellő pontossággal alkalmas közelíteni a piacot, melyben alkalmazzák. A hálózat bemeneteként a diszkrét kezdeti forward görbét meghatározó paramétereket, valamint swaption árak egy halmazát használtuk. A kimenet a volatilitásfüggvény paramétereit reprezentálta.

A predikciós eredményeink kellően pontosnak bizonyultak, így a módszer potenciálisan al-

kalmazható stressztesztelési környezetben, hagyományos optimalizációs eljárások inicializálására és ezáltal azok gyorsítására, valamint – megfelelő, a dolgozat kereteit meghaladó számítási kapacitás birtokában – önálló ATM-kalibrációs eljárásként is. Komplexebb volatilitásparametrizáció esetén az alábbi megközelítést tartjuk leginkább járhatónak: külön neurális hálózat tanítása a hosszú távú és a lokális hatást kifejtő paraméterekre. Bár összességében a lokális paraméterek esetében nem értünk el kielégítő eredményeket, bizakodunk a módszer alkalmazhatóságában.

A dolgozatban tehát sikeresen demonstráltuk mesterséges neurális hálózatok ATM-kalibrációs feladatokban való alkalmazhatóságát. További kutatási irányt jelent a bonyolultabb volatilitás parametrizációk és hálóarchitektúrák vizsgálata, különös tekintettel arra, hogy milyen szintig valósítható meg a kalibráció gyorsítása ezen a módszertani úton, valamint az, hogy tényleges piaci körülmények között, esetleges piaci inkonzisztenciák mellett milyen teljesítményre képesek ezek a megközelítések.

Python Codes

```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import openpyxl
import torch
import torch.nn as nn
from torch.optim import Adam
from tqdm import tqdm
import scipy.stats as stats
import seaborn as sns
import itertools


def hjm_sim(num_traj, num_points, dt, initial_curve, voltype, vol_params):

    # creating forward curves, initialization
    f = np.zeros((num_traj, num_points , num_points))
    f[:, 0, :] = initial_curve

    # determining vol matrix
    if voltype == 'flat':
        num_factors = 1
        volatility = np.full((num_factors, num_points, num_points), vol_params[0])

    if voltype == '2-factor':
        num_factors = 2
        c = vol_params[0]
        gamma = vol_params[1]
        lambda1 = vol_params[2]
        lambda2 = vol_params[3]
        volatility = np.zeros((num_factors, num_points, num_points))
        for j in range(num_points):
            for k in range(num_points):
                volatility[0, j, k] = c + gamma * np.exp(-lambda1 * (k - j) * dt)
                volatility[1, j, k] = gamma * np.exp(-lambda2 * (k - j) * dt)

    # forward curve evolution
    for i in range(1, num_points):

        #noise
        dz = np.random.normal(size=(num_traj, num_factors))

        #generating risk-free drift
        mu_help = np.roll(volatility[:, i-1, i:], shift = +1, axis=-1)
```

```
        mu_help[:,0] = 0
        mu = (np.cumsum(volatility[:, i - 1, i:], axis = 1)**2 - np.cumsum(mu_help, axis = 1)
            **2) * (dt**2) / 2

        #using the previous point in time
        f[:, i, i:] = f[:, i - 1, i:]

        for j in range(num_factors):

            #adding diffusion
            f[:, i, i:] += np.sqrt(dt) * volatility[j, i - 1, i:] * np.broadcast_to(dz[:, j
                ][:, np.newaxis], (num_traj, num_points - i))

            #adding drifts
            f[:, i, i:] += mu[j, :]

        #setting not valid indices zero
        f[:, i, :i] = 0

        # moment matching
        for j in range(i + 1, num_points):

            # setting zero_coupon prices at zero
            initial_bond = np.exp(np.sum(-f[0,0,:j])*dt)

            # calculating discounted bond prices
            b = np.full((num_traj), float(0))
            for k in range(i):
                b -= f[:,k,k]
            for k in range(i, j):
                b -= f[:,i,k]
            B = np.average(np.exp(b*dt))

            #adjustment
            f[:,i,j - 1] -= np.log(initial_bond/B)/dt

    return f, volatility




def nelson_siegel_curve(num_points, beta0, beta1, beta2, Lambda, dt=1/4):
    t = np.arange(num_points) * dt
    forward_rates = beta0 + beta1 * np.exp(-Lambda * t) + beta2 * Lambda * t * np.exp(-Lambda
        * t)
    return forward_rates

```

```
90
91
92  def swaption_pricer_approximation_investigation(f, dt=0.25, notional=1000000, expiry=0, tenor
        =1, maturity=4, sigma=0.01):
93      '''expiry, tenor, maturity: all need to be set in quarter'''
94
95      #money market account
96      mm = np.ones((np.shape(f)[0], expiry + 1))
97      for i in range(1, expiry + 1):
98          mm[:, i] = mm[:, i - 1] * np.exp(f[:, i, i] * dt)
99
100     #discount factors at time 0
101     df_0 = np.ones((np.shape(f)[0], expiry + tenor + maturity + 1))
102     for i in range(1, expiry + tenor + maturity + 1):
103         df_0[:, i] = df_0[:, i - 1] * np.exp(-f[:, 0, i - 1] * dt)
104
105     #discount factors at expiry
106     df_expiry = np.ones((np.shape(f)[0], tenor + maturity + 1))
107     for i in range(1, tenor + maturity + 1):
108         df_expiry[:, i] = df_expiry[:, i - 1] * np.exp(-f[:, expiry, i + expiry - 1] * dt)
109
110     payment_indices = np.arange(expiry + tenor, expiry + tenor + maturity)
111     annuity = np.sum(df_0[0, payment_indices] * dt)
112
113     floating_1 = np.zeros((np.shape(f)[0]))
114     for i in payment_indices:
115         floating_1 += df_0[:, i] * f[0, 0, i] * dt
116
117     floating_2 = np.zeros((np.shape(f)[0]))
118     for i in payment_indices:
119         floating_2 += df_0[:, i] * (f[0, 0, i-1] + f[0, 0, i]) / 2 * dt
120
121     floating_3 = np.zeros((np.shape(f)[0]))
122     for i in payment_indices:
123         floating_3 += df_0[:, i] * ((1 + f[0, 0, i-1]*45/360) * (1 + f[0, 0, i]*45/360)-1)
124
125     floating_4 = np.zeros((np.shape(f)[0]))
126     for i in payment_indices:
127         floating_4 += df_0[:, i] * (((((1 + f[0, 0, i-1]/360) ** 45) * ((1 + f[0, 0, i]/360)
                ** 45))) - 1)
128
129     floating_5 = np.zeros((np.shape(f)[0]))
130     for i in payment_indices:
131         floating_5 += df_0[:, i] * (np.prod([(1 + (f[0, 0, i-1]+(f[0, 0, i]-f[0,0,i-1])/89*i)
                /360) for i in range(0, 90)]) - 1)
132
133     floating_6 = np.zeros((np.shape(f)[0]))
```

```
    for i in payment_indices:
        floating_6 += df_0[:, i] * (np.prod([(1 + (f[0, 0, i-1]+(f[0, 0, i]-f[0,0,i-1])/89*i)
            /360) for i in range(0, 90)]) - 1)

    strike_1 = floating_1 / annuity
    strike_2 = floating_2 / annuity
    strike_3 = floating_3 / annuity
    strike_4 = floating_4 / annuity
    strike_5 = floating_5 / annuity
    strike_6 = floating_6 / annuity

    print("Strikes", strike_1[0], strike_2[0], strike_3[0], strike_4[0], strike_5[0], strike_6
        [0])

    floating_leg1 = np.zeros((np.shape(f)[0]))
    floating_leg2 = np.zeros((np.shape(f)[0]))
    floating_leg3 = np.zeros((np.shape(f)[0]))
    floating_leg4 = np.zeros((np.shape(f)[0]))
    floating_leg5 = np.zeros((np.shape(f)[0]))
    floating_leg6 = np.zeros((np.shape(f)[0]))
    fixed_leg_1 = np.zeros((np.shape(f)[0]))
    fixed_leg_2 = np.zeros((np.shape(f)[0]))
    fixed_leg_3 = np.zeros((np.shape(f)[0]))
    fixed_leg_4 = np.zeros((np.shape(f)[0]))
    fixed_leg_5 = np.zeros((np.shape(f)[0]))
    fixed_leg_6 = np.zeros((np.shape(f)[0]))

    for i in payment_indices:
        floating_leg1 += df_expiry[:, i-expiry] * notional * f[:, expiry, i] * dt

        floating_leg2 += df_expiry[:, i-expiry] * notional * (f[:, expiry, i] + f[:, expiry, i
            -1]) / 2 * dt

        floating_leg3 += df_expiry[:, i-expiry] * notional * ((1 + f[:, expiry, i-1]*45/360) *
             (1 + f[:, expiry, i]*45/360)-1)

        floating_leg4 += df_expiry[:, i-expiry] * notional * (((((1 + f[:, expiry, i-1]/360)
            ** 45) * ((1 + f[:, expiry, i]/360) ** 45))) - 1)

        floating_leg5 += df_expiry[:, i-expiry] * notional * (np.prod([(1 + (f[:, expiry, i
            -1]+(f[:, expiry, i]-f[:,expiry,i-1])/89*j)/360) for j in range(0, 90)], axis = 0)
             - 1)

        b = np.sqrt(1/360) * np.random.normal(0, 1, (np.shape(f)[0], 89))
        bb = np.cumsum(b, axis=1)
        fl_bridge = np.ones((np.shape(f)[0], 90))
        fl_bridge[:, 0] = f[:, expiry, i-1]
```

```
        for j in range(1, 90):
            fl_bridge[:, j] = f[:, expiry, i-1] + (f[:, expiry, i] - f[:, expiry, i-1]) * j /
                89 + sigma * (bb[:, j-1] - (j / 89) * b[:, -1])
        floating_leg6 += df_expiry[:, i-expiry] * notional * (np.prod([(1 + fl_bridge[:, j
            ]/360) for j in range(0, 90)], axis = 0) - 1)

        fixed_leg_1 += df_expiry[:, i-expiry] * notional * strike_1 * dt
        fixed_leg_2 += df_expiry[:, i-expiry] * notional * strike_2 * dt
        fixed_leg_3 += df_expiry[:, i-expiry] * notional * strike_3 * dt
        fixed_leg_4 += df_expiry[:, i-expiry] * notional * strike_4 * dt
        fixed_leg_5 += df_expiry[:, i-expiry] * notional * strike_5 * dt
        fixed_leg_6 += df_expiry[:, i-expiry] * notional * strike_6 * dt

    price_1 = np.mean(np.maximum(floating_leg1 - fixed_leg_1, 0) / mm[:, expiry], axis=0)
    price_2 = np.mean(np.maximum(floating_leg2 - fixed_leg_2, 0) / mm[:, expiry], axis=0)
    price_3 = np.mean(np.maximum(floating_leg3 - fixed_leg_3, 0) / mm[:, expiry], axis=0)
    price_4 = np.mean(np.maximum(floating_leg4 - fixed_leg_4, 0) / mm[:, expiry], axis=0)
    price_5 = np.mean(np.maximum(floating_leg5 - fixed_leg_5, 0) / mm[:, expiry], axis=0)
    price_6 = np.mean(np.maximum(floating_leg6 - fixed_leg_6, 0) / mm[:, expiry], axis=0)

    return price_1, price_2, price_3, price_4, price_5, price_6



def swaption_pricer_bb(f, dt=0.25, notional=100000000, expiry=0, tenor=1, maturity=4, Sigma=
    None, sw_sp={}):
    '''expiry, tenor, maturity: all need to be set in quarter; receives swaption details in
        sw_sp dictionary'''

    #money market account
    mm = np.ones((np.shape(f)[0], np.shape(f)[1]))
    for i in range(1, expiry + 1):
        mm[:, i] = mm[:, i - 1] * np.exp(f[:, i, i] * dt)

    #discount factors at time 0
    df_0 = np.ones((np.shape(f)[0], np.shape(f)[1]))
    for i in range(1, expiry + tenor + maturity + 1):
        df_0[:, i] = df_0[:, i - 1] * np.exp(-f[:, 0, i - 1] * dt)

    # List for prices
    prices = np.zeros((len(sw_sp.keys())))
    count = 0

    for item in sw_sp.keys():

        expiry = sw_sp[item]['expiry']
        tenor = sw_sp[item]['tenor']
```

```
        maturity = sw_sp[item]['maturity']

        #discount factors at expiry for specific swaption
        df_expiry = np.ones((np.shape(f)[0], tenor + maturity + 1))
        for i in range(1, tenor + maturity + 1):
            df_expiry[:, i] = df_expiry[:, i - 1] * np.exp(-f[:, expiry, i + expiry - 1] * dt)

        payment_indices = np.arange(expiry + tenor, expiry + tenor + maturity)

        # calculations supporting strike calculation: PV(float) = PV(fixed)
        floating = np.zeros((np.shape(f)[0]))
        annuity = np.sum(df_0[0, payment_indices] * dt)
        for i in payment_indices:
            floating += df_0[:, i] * (np.prod([(1 + (f[0, 0, i-1]+(f[0, 0, i]-f[0,0,i-1])/89*i
                )/360) for i in range(0, 90)]) - 1)

        strike = floating / annuity

        floating_leg = np.zeros((np.shape(f)[0]))
        fixed_leg = np.zeros((np.shape(f)[0]))

        for i in payment_indices:
            # approximating sigma via the sum of absolute volatilities for given expiry
            sigma = 0
            for j in range(np.shape(Sigma)[0]):
                sigma += np.abs(Sigma[j, expiry, i-1])

            # normal random variable for Brownian bridge
            b = np.sqrt(1/360) * np.random.normal(0, 1, (np.shape(f)[0], 89))
            bb = np.cumsum(b, axis=1)
            fl_bridge = np.ones((np.shape(f)[0], 90))
            fl_bridge[:, 0] = f[:, expiry, i-1]
            for j in range(1, 90):
                fl_bridge[:, j] = f[:, expiry, i-1] + (f[:, expiry, i] - f[:, expiry, i-1]) *
                    j / 89 + sigma * (bb[:, j-1] - (j / 89) * b[:, -1])

            floating_leg += df_expiry[:, i-expiry] * notional * (np.prod([(1 + fl_bridge[:, j
                ]/360) for j in range(0, 90)], axis = 0) - 1)
            fixed_leg += df_expiry[:, i-expiry] * notional * strike * dt

        price = np.mean(np.maximum(floating_leg - fixed_leg, 0) / mm[:, expiry], axis=0)
        prices[count] = price
        count += 1

    return prices
```

```
# Model training related code
# for data
file_path = ''
data = pd.read_excel(file_path)

# Extracting features and targets - change for specific model appropriately
vol_param = data[['vol_param']].values
curve_and_swaptions = data[['curve_param_lambda', 'curve_param_beta_0', 'curve_param_beta_1',
    'curve_param_beta_2', '1Y1Y', '1Y2Y', '1Y5Y', '1Y10Y', '1Y20Y',
                           '2Y1Y', '2Y2Y', '2Y5Y', '2Y10Y', '2Y20Y', '5Y1Y', '5Y2Y',
                           '5Y5Y', '5Y10Y', '5Y20Y', '10Y1Y', '10Y2Y',
                           '10Y5Y', '10Y10Y', '10Y20Y', '15Y1Y', '15Y5Y', '15Y10Y', '20Y5Y',
                               '20Y10Y']].values

# Convert to PyTorch tensors
features = torch.tensor(curve_and_swaptions, dtype=torch.float32)
targets = torch.tensor(vol_param, dtype=torch.float32)

# Standardize features
def standardize(data, mean=None, std=None):
    if mean is None and std is None:
        mean = torch.mean(data, dim=0)
        std = torch.std(data, dim=0)
    standardized_data = (data - mean) / std
    return standardized_data, mean, std

# Standardize training data
features_standardized_1, mean_features, std_features = standardize(features)



# Define Neural Network Model
class Model(nn.Module):
    def __init__(self, hidden):
        super().__init__()
        self.l1 = nn.Linear(29, hidden)
        self.a1 = nn.SiLU()
        self.l2 = nn.Linear(hidden, hidden)
        self.a2 = nn.SiLU()
        self.l3 = nn.Linear(hidden, hidden)
        self.a3 = nn.SiLU()
        self.l4 = nn.Linear(hidden, hidden)
        self.a4 = nn.SiLU()
        self.l5 = nn.Linear(hidden, 1)
        #self.a3 = nn.Sigmoid()

    def forward(self, x):
```

```
        x = self.a1(self.l1(x))
        x = self.a2(self.l2(x))
        x = self.a3(self.l3(x))
        x = self.a4(self.l4(x))
        x = self.l5(x)
        return x

# Loss function (Mean Squared Error with L2 Regularization)
def criterion(pred, target, weights, alpha):
    n = len(target)
    return (torch.sum(torch.abs(pred - target)**2) + alpha * torch.sum(weights**2))
""" # Alternative loss function with relative error
def criterion(pred, target, weights, alpha, eps=1e-6):
    rel_err = torch.abs(pred - target) / (target)
    return torch.sum(rel_err**2) + alpha * torch.sum(weights**2)"""

# Training Function
def train(features, targets, n_epochs, hidden, alpha, learning_rate, save_path):
    model = Model(hidden)
    optimizer = Adam(model.parameters(), lr=learning_rate)

    with tqdm(range(n_epochs)) as tepoch:
        for epoch in tepoch:
            optimizer.zero_grad()
            pred = model(features)
            weights = torch.cat([p.flatten() for p in model.parameters()])
            loss = criterion(pred, targets, weights, alpha)
            loss.backward()
            optimizer.step()
            tepoch.set_postfix(loss=loss.item())
    torch.save(model.state_dict(), save_path)
    print("Training finished and model saved.")



save_path = "...\\nn.pt"

h = 100
# Train the model
train(features_standardized_1, targets, 80000, h, 0.00001, 0.00001, save_path)

# Load Model with Safe Deserialization
torch.serialization.add_safe_globals([Model])  # Allowlist the Model class

# Initialize Model with correct architecture
model = Model(h)

```

```
# Load state_dict safely
state_dict = torch.load(save_path, map_location=torch.device('cpu'))
model.load_state_dict(state_dict)
model.eval()
```