Zürcher Hochschule
für Angewandte Wissenschaften

A Comparison Between Traditional and Machine Learning Based Calibration of the One Factor Hull-White Model

Master's Thesis

Author:	Benedikt Grimus
Matriculation number:	1438252
Study Program:	Banking & Finance
Supervisor:	Dr. Marc Weibel

ZHAW School of Management and Law

Submitted on: 18. November 2025

Abstract

This thesis presents a novel, end-to-end machine learning framework for the calibration of the one-factor Hull-White (HW) model and provides a comprehensive comparison against the traditional Levenberg-Marquardt (LM) algorithm. The core innovation involves integrating a neural network directly with the QuantLib pricing engine. In a significant departure from antecedent methods that train networks to mimic pre-calibrated parameters, this framework forces the network to learn by minimizing the true pricing error, dynamically computing model-implied swaption volatilities and comparing them against market observations. To inform its predictions, the network leverages a rich feature set, including external market indicators and the principal components of the yield curve.

This direct optimization framework is calibrated and tested on a dataset of European at-the-money (ATM) swaptions from the Euro (EUR) market, covering the period from June 1, 2025, to August 31, 2025. The empirical results demonstrate that this approach yields a calibration tool that is superior in speed, accuracy, and robustness. Once trained, the network performs calibration in milliseconds, a computational speed-up of over 22,000 times compared to the LM optimizer, effectively shifting the computational burden to a one-time offline phase. Furthermore, it produces economically sensible and stable parameters that react logically to simulated market shocks, whereas the traditional optimizer frequently suffers from numerical instability, with key parameters collapsing under stress scenarios.

The primary bottleneck of this advanced methodology is the substantial computational effort required for training, as the integration with the non-differentiable QuantLib engine necessitates numerical gradient approximations. This hybrid approach, while powerful, represents the main trade-off for achieving a more robust and accurate calibration tool. Despite this training challenge, the findings confirm that learning to minimize pricing error directly offers significant advantages, marking a promising advancement in financial model calibration.

JEL Classification: G13, C45

Contents

Acronyms

AI Artificial Intelligence

API Application Programming Interface

ATM at-the-money

bps Basis Points

CIR Cox-Ingersoll-Ross

EUR Euro

EURIBOR Euro Interbank Offer Rate

FRA Forward Rate Agreement

GSR Gaussian Short Rate

HW Hull-White

ITM in-the-money

LIBOR London Interbank Offer Rate

LM Levenberg-Marquardt

LMM Libor Market Model

MOVE Merrill Lynch Option Volatility Estimate

MSE Mean Squared Error

NN Neural Network

OTC over-the-counter

OTM out-of-the-money

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RMSE Root Mean Squared Error

SDE Stochastic Differential Equation

SHAP Shapley Additive Explanations

tanh Hyperbolic Tangent

USD United States Dollar

VIX Volatility Index

List of Figures

List of Tables

1 Introduction

Interest rate models are a cornerstone of modern quantitative finance, providing the essential framework for pricing derivatives, managing risk, and formulating hedging strategies (Moysiadis et al., 2019). Among the various term structure models, the one-factor HW model has established itself as a benchmark due to its analytical tractability and its ability to fit the initial term structure of interest rates. Its application is widespread, from the valuation of simple options like caps and floors to more complex, path-dependent instruments such as Bermudan swaptions (Brigo & Mercurio, 2006, p. 72). However, the utility of any model is fundamentally dependent on its calibration—the process of aligning its parameters with observed market prices to ensure that it reflects current market conditions and expectations.

The calibration of the HW model, particularly its mean-reversion and volatility parameters, presents a significant challenge. Traditional techniques are well-established, relying on powerful numerical optimization algorithms like the LM method to minimize the discrepancy between model and market prices. While this approach is a cornerstone of quantitative practice, its inherent properties create challenges for modern, high-frequency applications. As iterative numerical methods, they are by design computationally demanding, requiring significant time to converge. Furthermore, the non-convex nature of the calibration problem makes them susceptible to converging to local minima, potentially yielding suboptimal parameters that do not fully capture the market landscape, particularly in volatile regimes (Vollrath & Wendland, 2009). In an increasingly fast-paced environment that demands real-time pricing and risk management, this computational latency and potential for parameter instability create a critical bottleneck.

The recent advancements in machine learning, and specifically in the field of Neural Network (NN), offer a promising alternative to address these challenges. NNs are exceptionally adept at learning complex, non-linear relationships directly from data. By training a NN on historical market data and the corresponding "true" model parameters, it is possible to create a surrogate model that can approximate the calibration function almost instantaneously. This data-driven approach has the potential to transform the calibration process from a time-consuming optimization task into a rapid inference problem, thereby offering significant gains in speed and consistency (Hernandez, 2016). The motivation for this research stems from the pressing need for more efficient and robust calibration techniques in an increasingly fast-paced financial world.

The primary aim of this thesis is to provide a comprehensive comparison between the traditional, optimization-based calibration of the one-factor HW model and a novel, machine learning-based approach. To achieve this, the research will be guided by the following key questions:

1. How does the accuracy of a NN-based calibration, measured by the model's ability to replicate market swaption prices, compare to that of the traditional LM algorithm?

2. What is the quantitative difference in computational speed between the two methods, considering both the initial setup (training) and the ongoing application (inference)?

3. How stable are the estimated model parameters over time?

To ensure a focused and rigorous analysis, this study is subject to several delimitations. The scope is confined to the one-factor HW model, calibrated using a comprehensive dataset of European ATM swaptions from the EUR market. The analysis covers a continuous time period from June 1, 2025, to August 31, 2025, providing a view of the model's performance in a recent market environment. The study does not extend to multi-factor models or other classes of interest rate derivatives.

This thesis is structured as follows. Section 2 provides a detailed background on the theoretical foundations, including swaptions, yield curve bootstrapping, and a thorough review of the HW model and its traditional calibration. Section 3 delineates the research methodology, covering the dataset, the implementation of the machine learning framework, the feature engineering process, and the metrics used for comparison. Section 4 presents the empirical results of the comparative analysis, focusing on pricing abilities, speed, and robustness. Finally, section 6 concludes the thesis by summarizing the key findings, discussing their implications for financial practitioners, and suggesting avenues for future research.

2 Background

To provide the necessary context for the comparative analysis at the heart of this thesis, this chapter establishes a comprehensive theoretical foundation. The discussion is structured to build from fundamental concepts to specific applications. It begins with an overview of swaptions, detailing their characteristics and significance in the interest rate derivatives market. Following this, the chapter explains the crucial process of bootstrapping the zero-coupon yield curve from swap rates, a foundational step for any term structure model.

With the market instruments defined, the focus shifts to the models governing their dynamics. An introduction to the theory of short-rate models provides the broader context, differentiating between equilibrium and no-arbitrage frameworks. This leads to a detailed examination of the one-factor HW model, covering its mathematical specification, analytical solutions for pricing, and the traditional calibration process via numerical optimization techniques such as the LM algorithm. Finally, the chapter introduces the core principles of the machine learning approach used in this study, outlining the foundational concepts of NNs, including their architecture and learning mechanisms like the back-propagation algorithm, which enable their use as a powerful and modern calibration tool.

2.1 Swaptions: Definition, Characteristics, and Financial Relevance

A swaption, or swap option, is a derivative contract that grants the holder the right, but not the obligation, to enter into an interest rate swap at a predetermined date in the future and under predefined terms. Specifically, a payer swaption gives the holder the right to enter into a swap as the fixed-rate payer (and floating-rate receiver), while a receiver swaption gives the right to enter as the fixed-rate receiver (and floating-rate payer). Swaptions thus represent a combination of option and swaps and are a fundamental component of the interest rate derivatives market (Brigo & Mercurio, 2006, pp. 19–20).

Swaptions can be classified by the style of exercise they permit, which significantly influences their valuation complexity and hedging strategies.

European swaptions are the most common type and allow exercise only at a single specified future date (Brigo & Mercurio, 2006, p. 19), typically coinciding with the start of the underlying swap. Their valuation is relatively tractable and often performed using closed-form solutions such as Black's formula (Black, 1976).

American swaptions allow exercise at any time up to the expiry date (Karlsson et al., 2016). This added flexibility increases the option's value but also its computational complexity. Due to the path-dependency of optimal exercise, American swaptions are typ-

ically valued using numerical methods such as lattice models (Gurrieri et al., 2009) or finite-difference (Longstaff & Schwartz, 2001) schemes.

Bermudan swaptions occupy an intermediate position, allowing the holder to exercise the option on a set of predetermined dates, often corresponding to coupon dates of the underlying swap. Bermudan swaptions are widely traded (Karlsson et al., 2016) and are especially relevant for callable and cancellable structures in interest rate risk management (Rebonato, 2004, p. 22). Valuation requires techniques that account for multiple early exercise opportunities, such as backward induction (Karlsson et al., 2016) or least-squares Monte Carlo methods (Longstaff & Schwartz, 2001).

Asian swaptions are path-dependent options where the payoff depends on the average of the underlying swap rate over a certain period rather than a single rate observation at expiry. Because of this averaging feature, their valuation requires models that account for the entire path of interest rates rather than just their terminal value. Analytical or semi-analytical approximations-such as volatility expansion techniques can be used to price them (Baaquie, 2010).

Each of these exercise styles leads to distinct pricing and risk management considerations and determines the appropriate mathematical framework and numerical method for valuation. In addition to exercise style, a swaption is characterized by several structural and market parameters that determine its valuation and risk profile:

Option Maturity (Expiry): This denotes the time until the swaption can be exercised. It reflects the horizon over which uncertainty about future interest rate movements is resolved. Longer maturities generally imply greater sensitivity to the dynamics of the underlying yield curve and the volatility of interest rates.

Underlying Swap Tenor: The tenor is the length of the interest rate swap that begins upon swaption exercise (Brigo & Mercurio, 2006, p. 19). For example, a 5yX10y swaption refers to an option that expires in five years and, if exercised, initiates a ten-year interest rate swap. The swap tenor affects the duration and convexity of the position, and thus its sensitivity to changes in the yield curve.

Strike Rate: The strike rate is the fixed interest rate agreed upon in the swaption contract. At expiry, the decision to exercise the swaption depends on the relationship between this strike and the prevailing forward swap rate for the corresponding maturity and tenor (Brigo & Mercurio, 2006, p. 239).

Premium: The premium, or market price, of the swaption reflects the cost of acquiring the optionality embedded in the contract. It is influenced by all of the above parameters, as well as prevailing interest rates, the volatility environment, and the exercise style.

Implied Volatility: Swaptions are quoted in terms of implied volatility, which is extracted from market prices using inversion techniques based on models such as Black's formula (Hohmann et al., 2015, p. 3). Implied volatility encapsulates the market's expectation of future fluctuations in interest rates and plays a central role in option pricing and risk management. Market practitioners often use a two-dimensional volatility cube, where implied volatilities vary with both option expiry and swap tenor (Suo et al., 2009, p. 1).

Swaptions serve multiple purposes in financial markets, ranging from risk management to model calibration and investment strategies. They are fundamental tools used by financial institutions, borrowers, and specialized agencies to manage future interest rate risk. In particular, Bermudan swaptions are highly relevant for hedging callable and cancellable structures in the over-the-counter (OTC) market. For example, government-sponsored mortgage agencies in the USD market often purchase European and Bermudan swaptions to hedge exposure created by callable debt funding programs (Rebonato, 2004, p. 22). By granting the holder the right, but not the obligation, to enter into an interest rate swap at a predetermined future date, swaptions create optionality that protects against adverse movements in interest rates and makes sure that a company payments at each payment date have been capped to the fixed rate (Brigo & Mercurio, 2006, p. 17). The payoff of a European swaption can also be replicated by a continuously rebalanced portfolio of zero-coupon bonds, which provides valuable insight for managing the risk of positions in the derivative itself (Brigo & Mercurio, 2006, p. 241). Additionally, swaptions are used to manage volatility risk, which can be the largest component of risk in a derivative deal (Brigo & Mercurio, 2006, p. 242).

Beyond risk management, swaptions are critical for market pricing, valuation, and calibration of interest rate models (Brigo & Mercurio, 2006, pp. 132–136). Swaption prices are essential inputs for calibrating models such as the HW model (Kladivko & Rusy, 2023) and the Libor Market Model (LMM) for instance, calibrating the LMM to the swaption market allows parameters like instantaneous correlations to be determined (Brigo & Mercurio, 2006, p. 290).

Swaptions are also used to adjust financing costs and optimize yields. Investors seeking yield enhancement and issuers in search of advantageous funding rates may sell swaption-type optionality, while swaptions or swaption-like features are frequently embedded in more complex financial products or callable structures offered to investors (Rebonato, 2004, p. 22).

Swaptions represent highly versatile derivatives within interest rate markets, serving several critical functions. Their primary application lies in risk management, where they provide market participants with effective tools to hedge against adverse interest rate movements and manage volatility exposure. Moreover, swaptions play a pivotal role in the calibration of interest rate models; market-implied volatilities from swaption prices are

essential inputs for sophisticated frameworks like the HW and LMMs. Finally, beyond their utility in hedging and valuation, these instruments are strategically employed by investors to enhance portfolio yields and by issuers to optimize the costs associated with debt financing.

2.2 Bootstrapping the Zero-Coupon Yield Curve from Swap Rates

The construction of a zero-coupon yield curve from market swap rates is an essential step in interest rate modeling. Since swap contracts are among the most liquid instruments across maturities, they serve as primary inputs for determining the risk-free term structure. The goal of the bootstrapping process is to derive discount factors $\{P(t_0,T_i)\}_{i=1}^N$ or equivalently continuously compounded zero rates $\{z(T_i)\}_{i=1}^N$ that are consistent with observed market swap rates $\{R_i\}_{i=1}^N$ (J. C. Hull, 2015, pp. 84–86).

2.2.1 Preliminaries and Definitions

Let t_0 denote the valuation date and $\{T_i\}_{i=1}^N$ the maturities of the quoted par swap rates. Each swap with maturity T_i exchanges a fixed rate R_i for a floating rate indexed to a short-term reference rate (e.g., Euro Interbank Offer Rate (EURIBOR) 6M) at payment dates $\{t_{i,k}\}_{k=1}^{n_i}$ on the fixed leg.

The discount factor $P(t_0,t)$ represents the present value at time t_0 of one unit of currency to be received at time t. The continuously compounded zero rate $z(t)$ is defined by

$$P(t_0,t) = e^{-z(t)(t-t_0)}, \tag{1}$$

which implies

$$z(t) = -\frac{1}{t-t_0}\ln P(t_0,t). \tag{2}$$

2.2.2 Valuation of a Par Swap

A standard fixed-for-floating interest rate swap can be interpreted as the difference between a fixed-rate bond and a floating-rate note. The present value of the fixed leg at inception is given by

$$PV_{\text{fixed}}(t_0) = R_i \sum_{k=1}^{n_i} \alpha_{i,k} P(t_0,t_{i,k}), \tag{3}$$

where $\alpha_{i,k}$ denotes the accrual factor for the period $[t_{i,k-1},t_{i,k}]$ under the fixed-leg day count convention.

The present value of the floating leg for a par swap (i.e., at inception) equals

$$PV_{\text{float}}(t_0) = 1 - P(t_0,T_i), \tag{4}$$

under the assumption that the first floating coupon is set at par.

At inception, the par swap has zero net present value, so that

$$PV_{\text{fixed}}(t_0) = PV_{\text{float}}(t_0), \tag{5}$$

which yields the par swap relation

$$R_i \sum_{k=1}^{n_i} \alpha_{i,k} P(t_0, t_{i,k}) = 1 - P(t_0, T_i). \tag{6}$$

2.2.3 Recursive Bootstrapping Procedure

Equation (6) establishes a nonlinear relationship between the par swap rate R_i and the discount factors $\{P(t_0, t_{i,k})\}$. If discount factors up to T_{i-1} are already known from previously bootstrapped maturities, one can rearrange (6) to isolate the unknown discount factor $P(t_0, T_i)$ as

$$P(t_0, T_i) = \frac{1 - R_i \sum_{k=1}^{n_i-1} \alpha_{i,k} P(t_0, t_{i,k})}{1 + R_i \alpha_{i,n_i}}. \tag{7}$$

The procedure thus proceeds recursively:

1. Initialize the curve with known short-term instruments (e.g., deposits or short-dated Forward Rate Agreement (FRA)s) to obtain discount factors for the first maturities.
2. For each subsequent maturity T_i, solve (7) for $P(t_0, T_i)$ using previously determined discount factors.
3. Continue until the entire maturity spectrum $\{T_i\}_{i=1}^{N}$ is covered.

Once the discount factors are determined, the continuously compounded zero rates follow from (2).

2.2.4 Interpolation and Continuity of the Term Structure

The market provides only a discrete set of maturities $\{T_i\}$. To obtain a continuous term structure, the discount factors between quoted maturities are interpolated. A common and arbitrage-free choice is to interpolate linearly in log-discount space, i.e.,

$$\ln P(t_0, t) = (1 - \lambda) \ln P(t_0, T_j) + \lambda \ln P(t_0, T_{j+1}), \quad t \in [T_j, T_{j+1}], \tag{8}$$

where

$$\lambda = \frac{t - T_j}{T_{j+1} - T_j}. \tag{9}$$

This ensures that the discount function $P(t_0,t)$ is continuous, positive, and monotonically decreasing, thereby avoiding arbitrage opportunities.

The bootstrapping method yields a set of discount factors $\{P(t_0,T_i)\}$ that are internally consistent with observed swap rates $\{R_i\}$. The corresponding zero-coupon yield curve $\{z(T_i)\}$ satisfies:

$$z(T_i) = -\frac{1}{T_i - t_0} \ln P(t_0, T_i), \tag{10}$$

providing a continuous, arbitrage-free representation of the term structure of interest rates suitable for valuation, risk management, and model calibration.

Having established how the zero-coupon yield curve can be derived from observable market instruments, we now turn to models that describe the dynamic evolution of interest rates over time: short-rate models that form the theoretical foundation for pricing and valuing interest rate derivatives.

2.3 Short Rate Models

Short-rate models are a class of term structure models that describe the evolution of interest rates by modeling the dynamics of the instantaneous short-term interest rate, denoted by r_t. These models serve as the foundational framework for pricing interest rate derivatives, particularly those with path-dependent features or early exercise options such as American-style options where traditional models like Black's formula fall short due to their static assumptions. The short rate r_t represents the risk-free rate applicable over an infinitesimally short time interval and evolves according to a Stochastic Differential Equation (SDE), typically under the risk-neutral measure. Under this framework, the present value of future cash flows is derived by discounting at the short-rate path, allowing for a consistent valuation across a variety of financial instruments (J. C. Hull, 2015, pp. 706–735).

Short-rate models are flexible in that they can be constructed either to derive the term structure as a model outcome (as in equilibrium models) or to fit the observed term structure exactly (as in no-arbitrage models). Regardless of the approach, these models typically require numerical methods for implementation, including lattice-based techniques (e.g., trinomial trees) or Monte Carlo simulation, especially in the case of American or exotic derivatives. A key aspect of the usability of the model is its calibration to market data, which involves estimating parameters - such as mean reversion speed (a) and volatility (σ)-from prices of liquid instruments such as caps and swaps. In the post-2008 financial environment, the shift towards OIS discounting has led to the need for multiple-curve modeling frameworks, often requiring the simultaneous calibration of separate short-rate processes for both the discounting curve (e.g., OIS) and the forwarding curve (e.g., London Interbank Offer Rate (LIBOR)) (J. C. Hull, 2015, pp. 706–735).

2.3.1 Equilibrium Models

Equilibrium short-rate models are constructed by specifying a stochastic process for the short-term interest rate based on underlying economic principles, such as investor preferences and macroeconomic fundamentals. Unlike arbitrage-free models, equilibrium models derive the entire term structure of interest rates as an endogenous output of the model, rather than fitting it directly to observed market data. A distinguishing feature is that the drift term in the short rate's SDE is generally independent of time, reflecting long-term economic forces rather than market-implied expectations.

One of the earliest examples, the Rendleman-Bartter model, assumes that the short rate follows a geometric Brownian motion without mean reversion, which makes it analogous to equity price modeling, but unsuitable for interest rates due to its unbounded and potentially negative outcomes (J. C. Hull, 2015, p. 708). In contrast, the Vasicek model introduces mean reversion, modeling the short rate as an Ornstein-Uhlenbeck process. Although analytically tractable and capable of capturing the tendency of interest rates to revert to a long-term mean, it allows for negative interest rates, a feature that may be undesirable in certain market environments (J. C. Hull, 2015, pp. 708–709). The Cox-Ingersoll-Ross (CIR) model addresses this limitation by specifying that the diffusion term is proportional to the square root of the short rate, thereby ensuring non-negativity under suitable parameter conditions (J. C. Hull, 2015, p. 710).

Despite their elegance and analytical appeal, equilibrium models are often limited in practical applications because they do not guarantee consistency with the observed initial term structure. As such, they are more suitable for theoretical analysis and long-term economic forecasting than for precise pricing and hedging of interest rate derivatives in the current market environment (J. C. Hull, 2015, pp. 707–714).

2.3.2 No Arbitrage Models

No-arbitrage short-rate models are designed to exactly match the observed term structure of interest rates at inception by construction. These models ensure that there are no arbitrage opportunities in the pricing of fixed-income securities and derivatives by embedding the initial yield curve as an explicit input. The drift component of the SDE of the short rate in these models is typically time dependent, calibrated such that the models' generated bond prices align with market prices at time zero. As a result, no-arbitrage models are preferred for pricing and risk management in practice, especially in environments where accuracy in capturing the term structure is essential (J. C. Hull, 2015, pp. 714–715).

The Ho-Lee model, introduced in 1986, was the first short-rate model to incorporate the no-arbitrage principle. It assumes constant volatility and introduces a time-dependent drift term to fit the initial yield curve, resulting in a normally distributed short rate. While

simple and analytically tractable, the Ho-Lee model allows for negative interest rates and constant volatility across all maturities (J. C. Hull, 2015, pp. 715–716). To address the limitations of both the Vasicek and Ho-Lee models, the HW one-factor model combines mean reversion with time-dependent drift, allowing for greater flexibility in fitting the term structure and interest rate volatility (J. C. Hull, 2015, pp. 716–718).

Further refinements include multi-factor extensions, such as the HW two-factor model, which captures changes in both the level and slope of the yield curve through the interaction of two stochastic drivers. These enhancements are especially useful in capturing the empirical behavior of interest rates and in pricing complex derivatives such as Bermudan swaptions. The requirement to calibrate these models to market-observed instruments like caps, floors, and swaptions is central to their application, and they are widely used in practice for valuation, hedging, and risk management of interest rate products (J. C. Hull, 2015, p. 719).

Building upon the no-arbitrage framework that ensures consistency with the observed term structure, it is useful to further categorize models by the statistical nature of their short-rate dynamics leading to the class of Gaussian Short Rate (GSR) models, which assume normally distributed interest-rate movements and offer valuable analytical tractability.

2.3.3 Gaussian Short Rate Models

GSR models form a specific subclass within the broader family of short rate models in which the evolution of the short-term interest rate, or a transformation thereof, is governed by a stochastic process with normally distributed increments. The defining feature of these models is the assumption that the short rate r_t (or its change) evolves under a Brownian motion dz_t with constant or time-dependent volatility, leading to normally distributed outcomes (Brigo & Mercurio, 2006, p. 53). As such, these models offer analytical tractability and closed-form solutions for bond prices and some derivative instruments. However, they also imply the theoretical possibility of negative interest rates, which may or may not be consistent with the prevailing market environment or the model's intended application (J. C. Hull, 2015, pp. 719–720).

Formally, a GSR model is characterized by a SDE of the general form:

$$dr_t = \mu(t, r_t)dt + \sigma(t)dz_t, \tag{11}$$

where $\mu(t, r_t)$ is the drift term, $\sigma(t)$ is the volatility function (possibly constant or time-dependent), and dz_t denotes the increment of a standard Wiener process under the risk-neutral measure (Brigo & Mercurio, 2006, p. 53). Because the increments of dz_t are

normally distributed, so too are the increments of r_t, provided that the transformation applied to the short rate is linear.

Prominent examples of GSR models include the Vasicek, Ho-Lee, and HW (one-factor) models, each of which was discussed in prior sections. While differing in their specification of the drift and volatility terms, all three share the core Gaussian property of allowing for symmetric, unbounded movements in the short rate. This feature enhances analytical tractability-particularly in deriving closed-form solutions for zero-coupon bond prices-but also permits negative interest rates, a theoretical artifact that gained practical relevance in post-crisis monetary regimes where nominal yields fell below zero in several developed markets.

Importantly, GSR models stand in contrast to lognormal models such as the Black-Derman-Toy and Black-Karasinski models, which restrict the short rate to remain strictly positive by modeling either the logarithm of the rate or its multiplicative structure . While these alternative models circumvent the issue of negative rates, they often sacrifice analytical tractability in favor of numerical implementation (J. C. Hull, 2015, p. 718).

This master thesis will focus in the following chapters on the one-factor HW model, a prominent GSR model that enhances the classical Vasicek formulation by introducing a time-dependent drift term to ensure an exact fit to the initial term structure of interest rates.

2.4 The One-Factor Hull-White Model

The one-factor HW model represents a pivotal advancement in interest rate modeling, combining analytical tractability, empirical flexibility, and arbitrage-free consistency. As detailed in Hull and White's seminal 1990 paper (J. Hull & White, 1990), the model extends the classical Vasicek framework by introducing time-dependent parameters, enabling exact calibration to the observed initial term structure of interest rates. As explained in section 2.3.3, the HW model falls within the class of GSR models due to its assumption of normally distributed short rate dynamics. Furthermore, as discussed in section 2.3.2, it enforces market consistency by incorporating a time-varying drift term.

The HW model offers a favorable balance between model realism and computational tractability. Its ability to fit the observed term structure exactly, model time-varying volatilities, and provide analytic solutions for a wide class of derivatives positions it as a standard tool in fixed income modeling. It is particularly well-suited for applications such as: Pricing of swaptions, caps, and callable bonds, risk management and Value-at-Risk calculations and simulation of future interest rate scenarios for balance sheet modeling. The model's allowance for negative interest rates, due to its Gaussian nature, has become a point of practical relevance in recent years, as monetary policy in several advanced

economies has driven rates below zero.

2.4.1 Mathematical Specification

The HW model is often referred to as the *extended Vasicek model*, formalized as follows under the risk-neutral measure $\mathbb{Q}$:

$$dr_t = [\theta(t) - ar_t]\,dt + \sigma(t)dz_t, \tag{12}$$

where:

Table 1: Notation for the HW Model

Symbol	Meaning
r_t	Instantaneous short rate at time t
$\theta(t)$	Deterministic function fitted to match the initial term structure
a	Constant mean reversion speed ($a > 0$)
$\sigma(t)$	Time-dependent volatility function
dz_t	Standard Brownian motion under the risk-neutral measure $\mathbb{Q}$

This formulation allows the short rate to mean-revert toward a time-varying level $\theta(t)/a$, ensuring flexibility in capturing the shape of the initial yield curve while retaining a parsimonious structure. The presence of $\sigma(t)$ permits the model to reflect changes in the term structure of interest rate volatilities, which is crucial for accurately pricing interest rate derivatives.

2.4.2 Bond Pricing and Functional Solutions

One of the key strengths of the HW model is its closed-form solution for zero-coupon bond prices. The price $P(r_t,t,T)$ of a zero-coupon bond maturing at time T, given the short rate at time t, is given by:

$$P(r_t,t,T) = \exp\left[A(t,T) - B(t,T)r_t\right], \tag{13}$$

where the deterministic functions $A(t,T)$ and $B(t,T)$ are defined as:

$$B(t,T) = \frac{1}{a}\left(1 - e^{-a(T-t)}\right), \tag{14}$$

$$A(t,T) = \int_t^T \left[\theta(s)B(s,T) - \frac{1}{2}\sigma(s)^2 B(s,T)^2\right] ds. \tag{15}$$

These solutions emerge from solving the associated partial differential equation for contingent claim prices, under the assumption of a risk-neutral market and bounded market price of risk. The functional form of $A(t,T)$ links directly to the model's ability to replicate the observed yield curve.

2.4.3 Option Pricing

A notable advantage of the HW model is its analytic tractability for European-style options on bonds, which makes it especially valuable for practical implementation (Brigo & Mercurio, 2006, p. 72). The lognormality of bond prices under the model's GSR dynamics allows for closed-form expressions for European bond option prices. For instance, a European call option on a discount bond with maturity T and exercise at $t_1 < T$ has a pricing formula structurally analogous to the Black formula, with a volatility term given by:

$$\sigma_P = \sigma(t_1)B(t_1,T), \tag{16}$$

where the bond price volatility is independent of r_t, enabling straightforward evaluation of the option price. Options on coupon-bearing instruments can be priced via decomposition into portfolios of zero-coupon bond options.

For American-style options or more complex interest-rate-contingent claims, numerical techniques such as finite-difference methods or lattice-based approaches (e.g., trinomial trees) are required to solve the underlying partial differential equation.

2.4.4 Swaption Pricing

In the one-factor HW model, European swaptions can be priced analytically using a technique known as *Jamshidian's decomposition* (Jamshidian, 1989). This method is particularly powerful for one-factor models, where all coupon bond prices at a given time are deterministic functions of the short rate.

The fundamental insight of Jamshidian's approach is that a payer (or receiver) swaption, which is an option on a portfolio of fixed payments, can be decomposed into a portfolio of options on individual zero-coupon bonds. Since the HW model provides closed-form solutions for European options on zero-coupon bonds (as discussed in section 2.4.2), this decomposition facilitates an analytical expression for the swaption price.

A European payer swaption with strike K and maturity T references a fixed-for-floating interest rate swap with fixed payments at $T_1, T_2, \ldots, T_n$. Let $P(t,T_i)$ denote the price at time t of a zero-coupon bond maturing at T_i, and let α_i denote the accrual factors (e.g.,

0.5 for semiannual payments). The value at time t of the fixed leg of the swap is:

$$\text{FixedLeg}(t) = K \sum_{i=1}^{n} \alpha_i P(t, T_i). \tag{17}$$

The value of the floating leg at time t equals one minus the price of a discount bond maturing at the swap start date T_0, assuming the swap is par at initiation. The swaption payoff at expiry T is therefore:

$$\max\left(\sum_{i=1}^{n} \alpha_i P(T, T_i)(K - S(T)), 0 \right), \tag{18}$$

where $S(T)$ is the par swap rate at time T.

Jamshidian's decomposition allows this multi-cash-flow option to be re-expressed as a portfolio of individual bond options. Specifically, there exists a unique short rate r^* such that the present value of the fixed leg equals the strike value $K \sum_{i=1}^{n} \alpha_i P(T, T_i)$. Formally, we define r^* as the root of the equation:

$$\sum_{i=1}^{n} \alpha_i P(r^*, T, T_i) = \sum_{i=1}^{n} \alpha_i P(T, T_i), \tag{19}$$

where $P(r^*, T, T_i)$ is the model-implied bond price expressed as a function of the short rate. Once r^* is found, the swaption payoff can be decomposed into a sum of bond option payoffs with the same exercise date T:

$$\text{Swaption}(t) = \sum_{i=1}^{n} \alpha_i \cdot \text{Option}(P(t, T_i), K_i), \tag{20}$$

where each $K_i = P(r^*, T, T_i)$ acts as the strike price for the bond option maturing at T_i.

In the HW model, the price of a European call option on a zero-coupon bond with maturity T_i, strike K_i, and exercise at T is given by the closed-form formula:

$$C(t) = P(t, T_i) N(d_1) - K_i P(t, T) N(d_2), \tag{21}$$

with

$$d_1 = \frac{\ln\left(\frac{P(t,T_i)}{K_i P(t,T)}\right)}{\sigma_P} + \frac{1}{2}\sigma_P, \quad d_2 = d_1 - \sigma_P, \tag{22}$$

and σ_P being the bond price volatility given by:

$$\sigma_P^2 = \int_t^T \left(\sigma(s)B(s,T_i) - \sigma(s)B(s,T)\right)^2 ds. \tag{23}$$

The full swaption price is obtained by summing the individual bond option prices across all cash flow dates.

$$\text{Swaption}(t) = \sum_{i=1}^{n} \alpha_i \left[P(t,T_i)N(d_1^{(i)}) - K_i P(t,T)N(d_2^{(i)})\right], \tag{24}$$

$$\text{with} \quad d_1^{(i)} = \frac{\ln\left(\frac{P(t,T_i)}{K_i P(t,T)}\right)}{\sigma_P^{(i)}} + \frac{1}{2}\sigma_P^{(i)}, \quad d_2^{(i)} = d_1^{(i)} - \sigma_P^{(i)}, \tag{25}$$

$$\text{and} \quad \left(\sigma_P^{(i)}\right)^2 = \int_t^T \left[\sigma(s)B(s,T_i) - \sigma(s)B(s,T)\right]^2 ds, \tag{26}$$

$$\text{where} \quad K_i = P(r^*,T,T_i), \tag{27}$$

Table 2: Notation for Swaption Pricing via Jamshidian's Decomposition

Symbol	Meaning
$P(t,T)$	Price at time t of a zero-coupon bond maturing at T
T	Expiry of the swaption
T_i	Payment date of the i-th swap cash flow ($i = 1,\ldots,n$)
α_i	Year fraction (accrual factor) between T_{i-1} and T_i
r^*	Unique short rate at expiry T solving the bond portfolio equation
K_i	Strike price of the bond option for maturity T_i, i.e., $P(r^*,T,T_i)$
$\sigma(s)$	Instantaneous volatility function of the HW model
$B(s,T)$	Sensitivity function: $B(s,T) = \frac{1-e^{-a(T-s)}}{a}$
$\sigma_P^{(i)}$	Volatility of the bond price ratio $P(t,T_i)/P(t,T)$
$d_1^{(i)}, d_2^{(i)}$	Black-type variables for bond option i
$N(\cdot)$	Standard normal cumulative distribution function

Jamshidian's decomposition offers significant computational efficiency for one-factor models by reducing a complex multidimensional option pricing problem to a sequence of univariate bond option valuations. Moreover, this approach retains full analytical tractability under the HW framework, allowing for fast and accurate pricing of European swaptions, a critical requirement in both risk management and market calibration contexts.

2.4.5 Calibration of the Hull-White Model

The calibration of the one-factor HW model plays a central role in ensuring its effectiveness for interest rate derivative pricing and risk management. As established in previous sections, the HW model provides analytical tractability and the ability to fit the initial yield curve exactly. However, to make the model reflect market-implied prices of interest rate derivatives, particularly caplets and swaptions, its dynamic parameters-mean reversion $a(t)$ and volatility $\sigma(t)$-must be appropriately calibrated.

In practical applications, $a(t)$ and $\sigma(t)$ are commonly assumed to be piecewise constant to simplify the numerical implementation (Gurrieri et al., 2009, p. 7). However, unconstrained piecewise constant forms can lead to overfitting and parameter instability (Gurrieri et al., 2009, p. 2). To overcome this, the model is typically calibrated using functional parameterizations, such as logistic forms for $a(t)$ and cubic splines for $\sigma(t)$, to ensure smoothness and stability (Gurrieri et al., 2009, p. 8).

The calibration process relies on three key inputs: the initial yield curve, which determines $\theta(t)$, market-observed prices or implied volatilities of caplets and swaptions, and a well-defined objective function, often the sum of squared differences between model and market prices or implied volatilities. In particular, European swaptions serve as the primary calibration instruments. These are priced analytically via Jamshidian's decomposition (see section 2.4.4), allowing the decomposition of the swaption payoff into a sum of bond options. The analytical tractability of this decomposition facilitates efficient model evaluation during optimization.

Three main strategies are typically employed to calibrate the HW model parameters:

1. Fixed Mean Reversion: In this approach, the mean reversion parameter $a(t)$ is fixed to a pre-determined constant value, often based on empirical considerations or desired product sensitivities (e.g., Bermudan swaptions). The volatility parameter $\sigma(t)$ is then optimized to fit a subset of swaption prices or implied volatilities. While this method simplifies the calibration and allows for fast implementation, it can be sensitive to market regime changes and may offer poor global fit if the chosen a does not reflect current market conditions (Gurrieri et al., 2009, p. 13).

2. Two-Step Estimation: This method separates the calibration of $a(t)$ and $\sigma(t)$. The mean reversion $a(t)$ is first estimated using an approximation method (e.g., the simulated method of moments approximation), which relates swaption implied volatilities in a manner independent of $\sigma(t)$. Once $a(t)$ is estimated and fixed, $\sigma(t)$ is calibrated to fit analytical swaption prices. This approach offers a balance between robustness and speed, making it suitable for environments requiring frequent recalibration (Gurrieri et al., 2009, pp. 13–14).

3. Simultaneous Optimization: This strategy involves a joint, multi-dimensional optimization of both $a(t)$ and $\sigma(t)$ to minimize the misfit between model prices and market prices of swaptions. While computationally more intensive, this method typically yields the highest fit quality across a broader range of instruments. Constraints and functional parameterizations are critical to ensure numerical stability and prevent overfitting (Gurrieri et al., 2009, p. 14).

The choice of calibration instruments significantly impacts the stability and accuracy of the resulting parameters. Two principal approaches are distinguished (Gurrieri et al., 2009, p. 16):

1. Local (Subset) Calibration: Calibration is restricted to a limited set of coterminal swaptions (e.g., those maturing at 10Y or 20Y). This reduces the dimension of the optimization problem and allows for good fits on selected instruments, but can result in instability and poor performance on non-calibrated instruments, especially if time-dependent mean reversion is used (Gurrieri et al., 2009, pp. 16–24).

2. Global (Full Matrix) Calibration: The entire swaption matrix is used in the calibration procedure. This approach provides superior fitting quality across the full market surface and stabilizes the estimation of time-dependent parameters. When functional forms for $a(t)$ and $\sigma(t)$ are used in combination with global calibration, the model can achieve robust and accurate fits without evidence of overfitting. This thesis employs this global strategy for empirical analyses (Gurrieri et al., 2009, pp. 24–29).

The actual calibration of the HW model is performed by minimizing the discrepancy between observed market data and model-implied quantities (Alaya et al., 2021, p. 2). In practice, this is typically achieved by minimizing the squared differences between market-observed implied volatilities $\hat{\sigma}_{i,j}^{\text{market}}$ and the model-implied volatilities $\hat{\sigma}_{i,j}^{\text{model}}(\theta)$, where (i, j) index the swaption expiry and tenor combinations, and θ denotes the vector of model parameters (e.g., the values of $a(t)$ and $\sigma(t)$). The optimization problem can be formulated as

$$\min_{\theta} \sum_{(i,j)\in\mathscr{I}} w_{i,j} \left(\hat{\sigma}_{i,j}^{\text{model}}(\theta) - \hat{\sigma}_{i,j}^{\text{market}} \right)^2, \tag{28}$$

where $\mathscr{I}$ denotes the set of index pairs corresponding to available market swaptions and $w_{i,j} \geq 0$ are weighting factors that may be used to emphasize certain instruments (e.g., highly liquid swaptions or those near the money) Alternatively, this objective can be defined in terms of option prices instead of implied volatilities, depending on the availability of reliable market data and numerical stability considerations. The choice of objective function should reflect both the intended use of the model and the data quality. This minimization problem can be solved by algorithms suitable for solving nonlinear least-squares

problems such as the LM algorithm ((Vollrath & Wendland, 2009)) (see section 2.4.6).

Understanding how the parameters affect model output is essential. The volatility function $\sigma(t)$ predominantly controls the *level* of the implied volatility surface (Gurrieri et al., 2009, p. 9), while the mean reversion $a(t)$ influences its *shape* (Gurrieri et al., 2009, p. 9). Specifically, increasing $a(t)$ generally leads to lower implied volatilities and a change in curvature. Accurately capturing market phenomena such as volatility humps necessitates time-dependent mean reversion.

2.4.6 Levenberg-Marquardt Algorithm

The LM algorithm is a widely used method for solving nonlinear least-squares problems, particularly effective when estimating model parameters in contexts such as the calibration of interest rate models. It aims to minimize the sum of squared residuals between observed data points and the corresponding model predictions (Marquardt, 1963).

Given a nonlinear model $f(x;\beta)$ with parameters $\beta = (\beta_1, \ldots, \beta_k)$, and observations $\mathbf{y} = (y_1, \ldots, y_n)$, the objective is to minimize the cost function

$$\Phi(\beta) = \sum_{i=1}^{n} (y_i - f(x_i;\beta))^2 = \|y - \hat{y}(\beta)\|^2. \tag{29}$$

The algorithm interpolates between two classical optimization strategies:

1. Steepest descent method: Effective when far from the minimum but may converge slowly near it.

2. Gauss-Newton method: Efficient near the minimum but may diverge in highly nonlinear regions.

The LM method combines these approaches by updating the parameter vector β iteratively. In each iteration, the nonlinear model is linearized via a first-order Taylor expansion:

$$\hat{y}(\beta + \delta) \approx \hat{y}(\beta) + J\delta, \tag{30}$$

where J is the Jacobian matrix of partial derivatives with elements $J_{ij} = \frac{\partial f(x_i;\beta)}{\partial \beta_j}$ evaluated at the current parameter vector. The correction vector δ is then determined by solving the following regularized normal equation:

$$(J^\top J + \lambda I)\delta = J^\top (y - \hat{y}(\beta)), \tag{31}$$

where λ is a damping parameter, and I is the identity matrix.

The damping parameter λ controls the balance between the Gauss-Newton and steepest descent directions. For large λ, the algorithm behaves like a steepest descent method: (λI) dominates the system matrix, ensuring stability. For small λ, the method approaches the Gauss-Newton direction: $J^\top J$ dominates, allowing for faster convergence.

The value of λ is adjusted dynamically during the iteration process: If the new parameter vector $\beta + \delta$ results in a lower cost function Φ, the step is accepted and λ is decreased. If the step does not yield improvement, λ is increased, making the algorithm more conservative.

To improve numerical stability and convergence behavior, especially when model parameters have different magnitudes, parameter scaling is often applied. This ensures that step sizes and gradients are appropriately balanced.

Due to its adaptive nature and robustness, the LM algorithm is particularly well suited for the calibration of the HW model, where the objective function is highly nonlinear and sensitive to changes in model parameters.

2.5 Neural Networks

NNs are computational models inspired by the structure and function of the human brain and its nervous system. Unlike traditional computing systems that operate through explicit programming and deterministic logic, NNs are designed to learn from data by adjusting internal parameters in response to input stimuli. This adaptive capability enables them to approximate complex, non-linear functions and has led to their widespread use in fields such as image recognition, natural language processing, and financial forecasting.

2.5.1 Types of Problems Solved by Neural Networks

NNs have proven to be powerful and versatile tools for solving a wide variety of computational problems. While they are traditionally associated with regression and classification tasks, modern neural architectures are capable of addressing a broader spectrum of problems across supervised, unsupervised, and reinforcement learning paradigms. This section provides an overview of the principal problem types solvable by NNs.

Regression Problems: In regression tasks, the goal is to predict continuous numerical values based on input features. NNs learn a mapping from input vectors to a real-valued output, typically by minimizing a loss function such as the mean squared error (Goodfellow et al., 2016, p. 99). Applications include time series forecasting, energy demand prediction, and asset price estimation.

Classification Problems: Classification involves assigning input data to one or more discrete categories (Goodfellow et al., 2016, p. 98). NNs trained for classification tasks typically use softmax activation in the output layer and categorical cross-entropy as a loss function. Common applications include image recognition, sentiment analysis, fraud detection, and medical diagnosis.

Clustering: Clustering is an unsupervised learning problem where the aim is to group similar data points based on inherent structure in the data (Aljalbout et al., 2018). While traditional algorithms like k-means are commonly used, NN-based methods-such as autoencoders, self-organizing maps, and deep clustering networks-can learn more flexible and data-adaptive representations for clustering tasks.

Dimensionality Reduction: Dimensionality reduction seeks to project high-dimensional data into a lower-dimensional space while preserving meaningful structure. NNs can perform this task using autoencoders, which compress the data into a latent representation and then reconstruct it (Hinton & Salakhutdinov, 2006). Such techniques are widely used in data visualization, noise reduction, and feature extraction.

Reinforcement Learning: In reinforcement learning, agents learn to take actions in an environment to maximize a long-term reward signal (Sutton & Barto, 2015, pp. 2–3). NNs serve as function approximators for value functions or policies in deep reinforcement learning frameworks such as Deep Q-Networks (DQNs) and Actor-Critic methods. Reinforcement learning applications include robotics control, autonomous driving, game-playing agents, and resource optimization.

Having outlined the diverse range of problems that NNs are capable of addressing, it is now essential to examine their internal structure. Understanding the architecture of NNs provides insight into how these models represent and process information to solve such complex tasks.

2.5.2 Architecture

At the fundamental level, the basic unit of a NN is referred to as a *neuron* or *node*. Each neuron receives one or more input signals, which are combined using a set of learnable parameters called *weights* (Mienye & Swart, 2024, p. 4). These weighted inputs are summed and passed through a non-linear transformation known as an *activation function*, which determines the neuron's output (Mienye & Swart, 2024, pp. 4–5). The activation function introduces non-linearity into the model, enabling the NN to learn complex patterns in the data. Commonly used activation functions in NNs include the sigmoid function, the hyperbolic tangent (*Hyperbolic Tangent (tanh)*), and the Rectified Linear Unit (ReLU). Each of these functions introduces non-linearity into the model, enabling the NN to capture complex patterns in the input data. The sigmoid and *tanh* functions are smooth, bounded,

and differentiable, making them suitable for problems requiring normalized outputs. In contrast, the ReLU function, defined as $f(x) = \max(0,x)$, is computationally efficient and mitigates the vanishing gradient problem, thereby facilitating the training of deep NNs.

NNs are typically organized into a series of layers (Mienye & Swart, 2024, p. 4):

Input Layer: The input layer is the first layer of the NN and serves solely as the interface for receiving external data. Each neuron in this layer corresponds to a single feature or variable in the input vector. Importantly, neurons in the input layer do not perform any computation; instead, they transmit the raw input signals to the subsequent layer in the NN.

Hidden Layer(s): Hidden layers are situated between the input and output layers and consist of neurons that perform intermediate computations. These layers are termed "hidden" because they are not directly observable from the input or output. The primary function of hidden layers is to transform the input data into more abstract representations through successive linear and non-linear operations. NNs with more than one hidden layer are referred to as deep NNs, which can model highly intricate data relationships. Each neuron in a hidden layer is typically fully connected to all neurons in the preceding and succeeding layers, although sparse (partially connected) architectures are also used.

Output Layer: The output layer is the final layer in the NN and produces the model's predictions. The structure and activation function of this layer are typically chosen based on the nature of the problem-for example, a softmax activation function is used for multi-class classification, while a linear activation may be used for regression tasks.

The design of a NN, such as the number of hidden layers, the number of neurons per layer, and the choice of activation functions, determines its capacity to learn and generalize from data. Proper configuration and training of these NNs are essential to achieving optimal performance on a given task (Sazli, 2006).

While the architecture defines the structural capacity of a NN, its practical effectiveness ultimately depends on how well the model's parameters are optimized. This leads to the central concept of learning, the process by which NNs iteratively adjust their weights to minimize prediction errors and improve performance.

2.5.3 Learning in Neural Networks: The Back-Propagation Algorithm

A defining characteristic of artificial NNs is their ability to *learn* and *adapt* based on interactions with their environment. In this context, learning is defined as the process through which the free parameters (i.e., synaptic weights) of the NN are adjusted to minimize a discrepancy between the actual output and the desired target output. The nature of this adjustment process depends on the learning algorithm employed.

Among the various learning algorithms, the *back-propagation algorithm* is the most widely used method for training feed-forward NNs. It operates under the framework of supervised learning, where the correct output (label) is known for each training example. The algorithm iteratively minimizes a loss function by updating weights in the direction of the negative gradient of the error.

Let $y_j(n)$ denote the actual output of neuron j at iteration n, and let d_j be the corresponding desired output. The difference between the desired and actual output is quantified using a measure known as the *error energy*. For each neuron j in the output layer, the error signal $e_j(n)$ is defined as:

$$e_j(n) = d_j - y_j(n) \tag{32}$$

The *instantaneous error energy* for neuron j at iteration n, denoted by $\varepsilon_j(n)$, is calculated as half the square of the error signal:

$$\varepsilon_j(n) = \frac{1}{2} e_j^2(n) \tag{33}$$

This squared formulation ensures that the error energy is always non-negative and penalizes larger deviations more strongly. To evaluate the total discrepancy for the entire output layer Q, the *total error energy* $\varepsilon(n)$ is computed as the sum of the individual error energies:

$$\varepsilon(n) = \sum_{j \in Q} \varepsilon_j(n) \tag{34}$$

Minimizing this total error energy is the central objective of the back-propagation learning algorithm, which iteratively adjusts the NN's synaptic weights to reduce the mismatch between predicted and target outputs. The error signal $e_j(n)$ for neuron j is defined as:

$$e_j(n) = d_j - y_j(n) \tag{35}$$

The instantaneous error energy $\varepsilon_j(n)$ associated with neuron j is then given by:

$$\varepsilon_j(n) = \frac{1}{2} e_j^2(n) \tag{36}$$

The total instantaneous error energy $\varepsilon(n)$ over all neurons $j \in Q$ in the output layer Q is computed as:

$$\varepsilon(n) = \sum_{j \in Q} \varepsilon_j(n) \tag{37}$$

To reduce the total error energy $\varepsilon(n)$, the synaptic weights $w_{ji}(n)$ are updated using the gradient descent method. The weight update rule is expressed as:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{38}$$

where η is the learning rate, a small positive constant controlling the step size.

To compute the partial derivative $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$, the chain rule is applied:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial w_{ji}(n)} \tag{39}$$

Each term in this product is derived as follows:

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{40}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{41}$$

$$\frac{\partial y_j(n)}{\partial w_{ji}(n)} = f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) \cdot y_i(n) \tag{42}$$

where $f'(\cdot)$ is the derivative of the activation function of neuron j, and $y_i(n)$ is the output of neuron i in the preceding layer.

Substituting Equations 40, 41, and 42 into equation 39 yields:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) \cdot y_i(n) \tag{43}$$

Inserting this result into the weight update rule (equation 38) gives the final expression for the synaptic weight adjustment:

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot f'\left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right) \cdot y_i(n) \tag{44}$$

This process of computing the output error, back-propagating it through the NN, and updating the weights is performed iteratively for each training sample. Over time, the NN's parameters converge such that the output increasingly approximates the desired targets, thereby enabling the NN to generalize and make accurate predictions on unseen data. This iterative optimization process is typically continued until the total error $\varepsilon(n)$ reaches an acceptably low threshold or a maximum number of training epochs is completed (Sazli,

2006).

Although the learning algorithm determines how a NN updates its internal parameters, overall performance also depends on a set of external configuration choices known as hyperparameters. Selecting these effectively is crucial for achieving stable convergence and optimal generalization, motivating the use of systematic hyperparameter optimization methods such as the Hyperband algorithm.

2.5.4 The Hyperband Algorithm for Hyperparameter Optimization

The performance of NNs depends critically on the choice of hyperparameters, such as learning rate, batch size, number of layers, or regularization strength. Since these hyperparameters are not learned from data, their optimal configuration must be determined externally through hyperparameter optimization. Traditional methods such as grid search or random search are computationally inefficient, particularly when training deep NNs. The *Hyperband* algorithm, introduced by Li et al. (2017), provides an efficient, theoretically grounded framework for hyperparameter selection based on the principle of adaptive resource allocation and early stopping.

Let $\mathscr{H}$ denote the hyperparameter space, and let $f : \mathscr{H} \to \mathbb{R}$ be a performance function (e.g., validation loss or error) that maps a hyperparameter configuration $h \in \mathscr{H}$ to its corresponding validation loss after training with a finite computational budget. The goal is to find

$$h^* = \arg\min_{h \in \mathscr{H}} f(h), \tag{45}$$

subject to a total computational budget B_{total}.

Hyperband frames this optimization as a *multi-armed bandit* problem, in which each configuration corresponds to an arm, and computational resources correspond to the number of pulls allocated to each arm. The algorithm dynamically balances *exploration* (testing many configurations) and *exploitation* (allocating more resources to promising configurations).

The core component of Hyperband is the *successive halving* procedure. Suppose n configurations are sampled uniformly at random from $\mathscr{H}$ and each is trained for an initial budget r (e.g., number of epochs or training samples). Their performances $\{f_i\}_{i=1}^n$ are evaluated, and only the top fraction $\frac{1}{\eta}$ is retained, where $\eta > 1$ is the *reduction factor*. The surviving configurations are then trained for η times the previous budget, and the process repeats. Formally, the iterative process for successive halving can be summarized

as:

$$n_j = \left\lfloor n \cdot \eta^{-j} \right\rfloor, \tag{46}$$

$$r_j = r \cdot \eta^j, \tag{47}$$

for stages $j = 0, 1, \ldots, s_{\max}$, where n_j is the number of configurations and r_j is the resource budget per configuration at stage j. At each stage, the best $\lfloor n_j/\eta \rfloor$ configurations are promoted.

While successive halving requires a predefined (n, r) pair, Hyperband systematically explores different trade-offs between the number of configurations n and the allocated resources r. Specifically, it defines multiple *brackets* indexed by $s \in \{0, 1, \ldots, s_{\max}\}$, where each bracket represents a different allocation strategy.

For each bracket s, Hyperband computes:

$$s_{\max} = \left\lfloor \log_\eta R \right\rfloor, \tag{48}$$

$$n_s = \left\lceil \frac{s_{\max} + 1}{s+1} \eta^s \right\rceil, \tag{49}$$

$$r_s = \frac{R}{\eta^s}, \tag{50}$$

where R denotes the maximum allowable resource per configuration (e.g., maximum number of training epochs).

Each bracket executes a full successive halving run with its respective (n_s, r_s) values. This design enables Hyperband to balance between two extremes:

- Wide exploration (large n_s, small r_s): testing many configurations briefly.
- Deep exploitation (small n_s, large r_s): fully training fewer configurations.

The total computational cost per bracket is approximately constant, ensuring that the overall budget B_{total} is respected:

$$B_{\text{total}} \approx (s_{\max} + 1)R. \tag{51}$$

The Hyperband procedure can be summarized as follows:

1. For each bracket $s = s_{\max}, s_{\max} - 1, \ldots, 0$:
   (a) Sample n_s configurations $\{h_{s,i}\}_{i=1}^{n_s}$ uniformly from $\mathscr{H}$.
   (b) Run successive halving with initial resource r_s and reduction factor η.
   (c) Record the best-performing configuration h_s^* in each bracket.

2. Select the globally best configuration

$$h^* = \arg\min_s f(h_s^*). \tag{52}$$

The Hyperband algorithm enjoys strong theoretical guarantees under mild regularity assumptions. Li et al. (2018) show that it achieves an asymptotically optimal trade-off between the number of configurations and resources allocated, with expected simple regret bounded by

$$\mathbb{E}[f(h^*) - f^*] = \mathscr{O}\left(\sqrt{\frac{\log n}{B_{\text{total}}}}\right), \tag{53}$$

where $f^* = \min_{h \in \mathscr{H}} f(h)$ denotes the true optimal performance. This result demonstrates that Hyperband is provably more efficient than random search and grid search in expectation.

Hyperband provides a principled and computationally efficient framework for NN hyperparameter optimization by combining random search with adaptive resource allocation. Its key strength lies in automatically balancing exploration and exploitation through multiple parallel Successive Halving procedures, resulting in near-optimal use of available computational resources without requiring manual tuning of training budgets.

2.6 Related Work

Alvarez et al. (2022) demonstrated that the prices generated by a one-factor HW model which uses parameters calibrated by a NN which was trained to learn the inverse relationship between swaption prices and the model parameters α and σ were close to the observed market prices, suggesting that the NN is an effective method for accurately calibrating the model parameters.

Hernandez (2016) investigates the use of NN as a fast alternative to traditional optimization methods for calibrating the one-factor HW model with constant parameters. The study compares the NN approach with the conventional LM optimization method, which served as the calibration benchmark. Because historical data were insufficient for direct training, the author generated a synthetic dataset of 150,000 samples using the inverse of the calibration function, effectively employing the model itself to produce realistic training data. The resulting samples captured the joint distribution of parameters, yield curves, and model errors observed in historical calibrations. A feed-forward NN was then trained to approximate the mapping between market data and model parameters, thereby shifting the computational burden to an offline training phase. Once trained, the network could perform calibrations almost instantaneously. Backtesting results demonstrated that the NN achieved a calibration accuracy comparable to that of the LM optimizer while reducing computation time by several orders of magnitude. Although performance gradually

deteriorated after six to twelve months, periodic retraining every few months effectively restored accuracy. Overall, the study showed that NNs can deliver substantial efficiency gains in interest rate model calibration without sacrificing precision.

Alaya et al. (2021) also investigate the use of NNs for the calibration of interest rate models, focusing on the G2++ model and the CIR intensity model. The paper proposes using deep calibration to improve this process, making calibration faster, more accurate, and easier to handle. Two main approaches for deep calibration are presented: Indirect deep calibration, which uses intermediate quantities like covariance matrices derived from market data, and direct deep calibration, which uses market-observable zero-coupon rate curves directly. Both methods demonstrate significant time savings, with direct deep calibration being particularly efficient due to its simplicity and lower computational costs. The paper also highlights the advantages of training on synthetic data, which allows for large datasets, stress-testing scenarios, and direct error measurement. The results show that deep calibration significantly reduces calibration time compared to classical methods, with direct deep calibration being up to 7,600 times faster. Moreover, deep calibration achieves good accuracy and is more resilient to noise in the data, outperforming traditional calibration methods in terms of global error across all parameters. The approaches also work well when applied to other models, such as the CIR intensity model. Overall, the study demonstrates that deep calibration methods offer a practical, efficient alternative for calibrating financial models, with potential applications in various financial contexts.

Moysiadis et al. (2019) focussed on the calibration of the mean reversion speed parameter in the one-factor HW model and proposed a method in which a NN is utilized to learn the future movement of interest rates based on historical data. The primary aim was to extract the mean reversion parameter from the derivative of the function learned by the NN which approximates the future rate. The approach emphasizes the robustness of the model, particularly its ability to handle market turbulence. The results demonstrate that the NN-based method delivers mean reversion values comparable to those obtained using a rolling linear regression, a standard method.

This thesis advances NN-based HW model calibration by introducing a direct, end-to-end framework. Unlike previous approaches that train networks to replicate traditional optimizers using pre-calibrated datasets, the network here is integrated with the QuantLib pricing engine and minimizes the actual pricing error between model-implied and market-observed swaption volatilities. This approach allows the network to learn parameter mappings optimal for pricing rather than mimicking an optimizer. Additionally, a flexible piecewise constant volatility structure enhances the model's ability to fit complex swaption term structures. Empirical results demonstrate improved out-of-sample pricing accuracy, validating the benefits of this direct machine learning approach.

3 Methodology

This chapter provides a detailed, step-by-step blueprint of the research design and analytical framework used to compare the traditional and machine learning-based calibration methods. The process begins with a thorough description of the dataset, which includes European ATM swaptions, the underlying EUR swap curves, and key external market indicators like the Volatility Index (VIX) and Merrill Lynch Option Volatility Estimate (MOVE) indices.

From this raw data, a sophisticated set of features is engineered through a multi-stage preprocessing pipeline. This includes bootstrapping zero-coupon curves, handling non-trading days, and using Principal Component Analysis (PCA) to distill the yield curve's primary drivers, level, slope, and curvature, into a compact and uncorrelated feature set.

The core of the methodology is the end-to-end machine learning pipeline. This section details the NN's residual architecture, the systematic hyperparameter optimization conducted via the Hyperband algorithm, and the implementation of a custom, asymmetric loss function designed to penalize the underestimation of volatility. Furthermore, it addresses the significant technical challenge of integrating the differentiable TensorFlow framework with the non-differentiable QuantLib pricing engine through the use of custom gradients and parallelized computation.

Crucially, the chapter concludes by defining the rigorous experimental protocol established for the final comparison. It details the unique "Intra-Day Hold-Out Set" framework, which was specifically designed to ensure a scientifically valid and fair out-of-sample evaluation for both the predictive NN and the in-sample LM optimizer, thereby addressing a fundamental methodological challenge in comparing these two distinct approaches.

3.1 Dataset

3.1.1 Time Period

The market data used for the calibration of the one-factor HW model covers a continuous time span of 92 calendar days, from June 1, 2025, to August 31, 2025. Observations are available for each calendar day within this period; weekends and public holidays are not excluded. All market snapshots were collected at a daily frequency, ensuring a consistent temporal resolution across the entire observation window.

3.1.2 Swaptions

The calibration is based on a comprehensive panel of European ATM payer and receiver swaptions quoted in the EUR market. Each daily market snapshot consists of a complete

normal volatility surface containing 228 distinct volatility points, derived from combinations of option maturities and underlying swap tenors.

The set of option maturities includes: 1 month, 3 months, 9 months, 1 year, 2 years, 3 years, 4 years, 5 years, 6 years, 7 years, 8 years, 9 years, 10 years, 12 years, 15 years, 20 years, 25 years, and 30 years.

The corresponding underlying swap tenors are: 1 year, 2 years, 3 years, 4 years, 5 years, 7 years, 10 years, 12 years, 15 years, 20 years, 25 years, and 30 years.

The floating leg of each underlying swap resets on a semiannual (6-month) basis. Volatilities are quoted in normal (Bachelier) terms, expressed in Basis Points (bps) of the underlying swap rate, and represent the mid-market levels (average of bid and ask quotes). Only ATM swaptions are considered in the calibration.

All swaption data were obtained from Bloomberg using the `VCUB` function with `BVOL` as the source. Due to the university's Bloomberg license restrictions, direct data downloads were not permitted; therefore, the data were captured via screenshots and subsequently converted into Excel format for processing. Bloomberg employs the EUR OIS (ESTR) discounting curve, identified as `(514) EUR OIS (ESTR)`, and the EUR swap curve `(45) Euro` for the construction of the volatility cube. This ensures internal consistency between the volatility and yield curve data used in the HW model calibration.

3.1.3 Swap Curves

The initial yield curves employed for the HW model calibration are derived from the EUR (vs. 6M EURIBOR) Swap / Projection Curve, identified in Bloomberg as `45`. This curve represents a forward (projection) curve used to generate 6-month EURIBOR forward rates for pricing and valuation. The discounting is performed using the EUR OIS (ESTR) curve (`Bloomberg ID 514`), consistent with the multi-curve framework applied in current market practice.

The market instruments used in the bootstrapping of the projection curve include:

- Cash deposits: Overnight to 12-month maturities, anchoring the short end of the curve.
- FRA: Typically spanning 1×7 to 9×15 months, bridging the short and medium-term maturities.
- Interest-rate swaps: Par fixed-vs-6M EURIBOR swaps from 1 year up to 30 years, forming the long end of the term structure.

Day-count conventions follow market standards: ACT/360 for short and medium maturities and 30U/360 for the long end. All market quotes correspond to mid (bid/ask midpoint) levels. The market data source is the Bloomberg Generic (BGN) composite, which aggregates dealer contributions to provide representative market levels.

The curve is bootstrapped sequentially from cash deposits through FRAs to interest-rate swaps, producing a continuous zero rate and forward rate term structure. The interpolation is typically performed in a log-linear fashion on either discount factors or zero rates. Within the Bloomberg volatility cube environment (`VCUB`), this curve appears as `Swap Curve (45) Euro` and provides consistent forward rate inputs for the calibration of EUR 6M swaption volatilities.

3.1.4 MOVE Index

The MOVE index represents the treasury market's implied volatility and serves as a primary external indicator of interest rate uncertainty. Daily open values of the MOVE index were collected from Yahoo Finance to capture the most up-to-date measure of market-implied interest rate volatility at the start of each trading day. By using the open rather than closing values, the NN is able to predict HW model parameters early in the day, leveraging the most recent available information without waiting for end-of-day data. The MOVE index is particularly relevant as it captures expected fluctuations in interest rates, which directly influence the calibration of the HW model.

3.1.5 Volatility Index

The VIX provides a widely recognized measure of equity market implied volatility and serves as a general gauge of market fear and risk appetite. Daily open values of the VIX were downloaded from Yahoo Finance to reflect the initial market sentiment for each trading day. High VIX levels, indicative of a "risk-off" environment, can have spillover effects on bond and swap markets, thereby indirectly affecting European interest rates. Including VIX open values in the dataset allows the NN to incorporate contemporaneous equity market volatility into the prediction of HW parameters, potentially improving model responsiveness to market stress conditions.

3.1.6 EUR/USD Exchange Rate

The EUR/USD exchange rate reflects the relative economic and monetary conditions between the Eurozone and the United States. Daily open prices were obtained from Yahoo Finance to capture the most current cross-currency market information at the start of each trading day. Large movements in the EUR/USD rate may indicate capital flows or diverging monetary policy actions between the European Central Bank and the Federal Reserve, which have direct implications for European interest rates. By including the open

EUR/USD rate in the feature set, the NN can incorporate up-to-date foreign exchange market dynamics when predicting HW model parameters.

3.2 Descriptive Analysis of Market and Model Inputs

This section provides a detailed examination of the key figures illustrating the underlying financial data and model inputs used in this study. Each figure is analyzed with respect to its economic interpretation and its implications for the modeling framework.

Figure 1: Time Series of Market-Wide Indicators (VIX, MOVE, and EUR/USD)

Figure 1 depicts the evolution of key market indicators between June and late August 2025. The overall market environment during this period is characterized by a notable decrease in volatility across major asset classes. The *VIX*, representing equity market volatility, begins near a relatively elevated level of 20 before declining sharply in late June and

stabilizing within a range of 14-17. This pattern suggests a reduction in perceived equity market risk. Similarly, the *MOVE Index*, a measure of treasury market volatility, exhibits an even more pronounced decline, from approximately 100 to below 80, indicating a substantial calming of interest rate expectations. Concurrently, the *EUR/USD* exchange rate shows a strengthening of the Euro against the United States Dollar (USD), peaking near 1.18 in mid-July before a modest correction. These co-evolving patterns reflect a broad reduction in cross-asset risk and uncertainty.

Table 3: Summary Statistics of External Market Factors

Factor	**Mean**	**Median**	**Std. Dev.**	**Skewness**	**Kurtosis**
VIX	17.05	16.71	1.89	0.89	0.34
MOVE Index	86.29	86.02	6.12	0.10	-1.10
EUR/USD Rate	1.162	1.164	0.011	-0.57	-0.54

A summary of the key external market factors over the analysis period. The positive skew in the VIX and the non-zero kurtosis across all factors suggest deviations from a normal distribution.

Table 3 provides a quantitative summary of the external market factors depicted in figure 1. The statistics confirm the visual analysis, detailing the central tendency and dispersion of each series. Notably, the VIX exhibits significant positive skewness (0.89), indicating that upward spikes in volatility are more pronounced than downward movements. Conversely, the EUR/USD rate shows a negative skew (-0.57). The kurtosis values, which deviate from the zero value of a mesokurtic distribution, provide further numerical evidence that these financial time series are not normally distributed, a crucial consideration for the modeling approach.

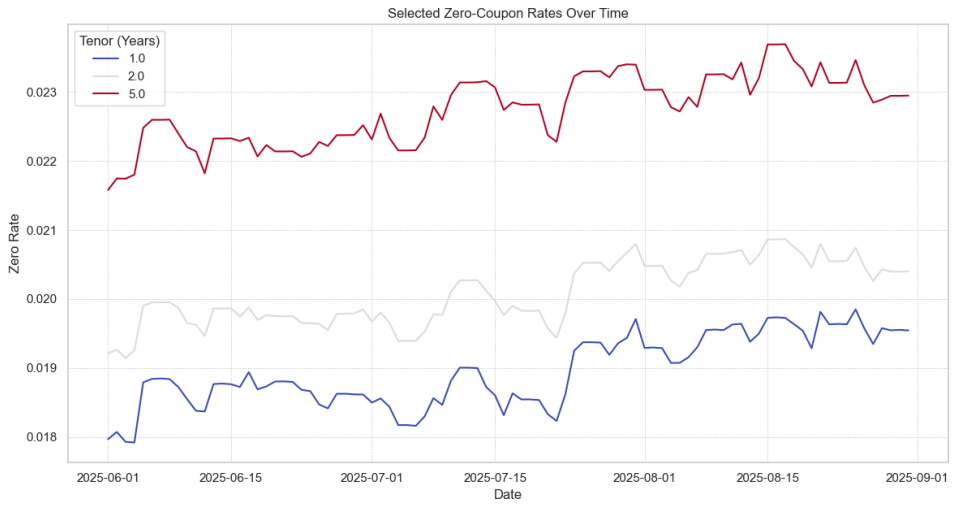

Figure 2: Yield Curve Dynamics for Selected Tenors (1Y, 2Y, 5Y)

Figure 2 illustrates the time series of yields across selected maturities. The yield curve

retains its normal, upward-sloping shape throughout the observation window. A gradual upward shift is visible across all tenors, indicating a moderate tightening of monetary conditions. The largely parallel movement across maturities implies that changes are predominantly driven by a "level" factor. Minor deviations in spreads, however, suggest that "slope" and "curvature" components contribute modestly to the curve's dynamics.

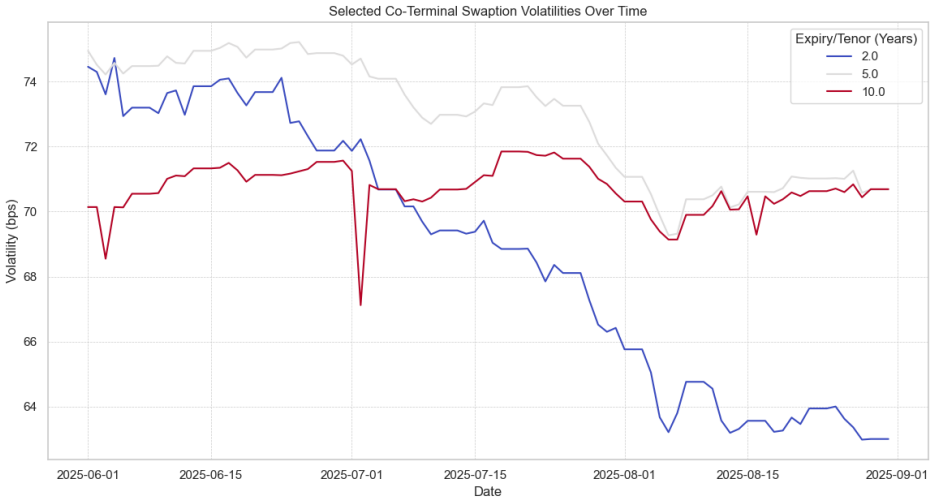

Figure 3: Time Series of Coterminal Swaption Volatilities

Figure 3 presents the evolution of the coterminal swaption volatility term structure. The curve is generally downward sloping, with shorter maturities exhibiting higher volatility levels than longer maturities. The short-term (2-year) volatility declines sharply from over 74 bps to approximately 63 bps, whereas the long-term (10-year) volatility remains comparatively stable. This differential behavior leads to a pronounced flattening of the volatility term structure-a key empirical feature that any robust model should accurately capture.

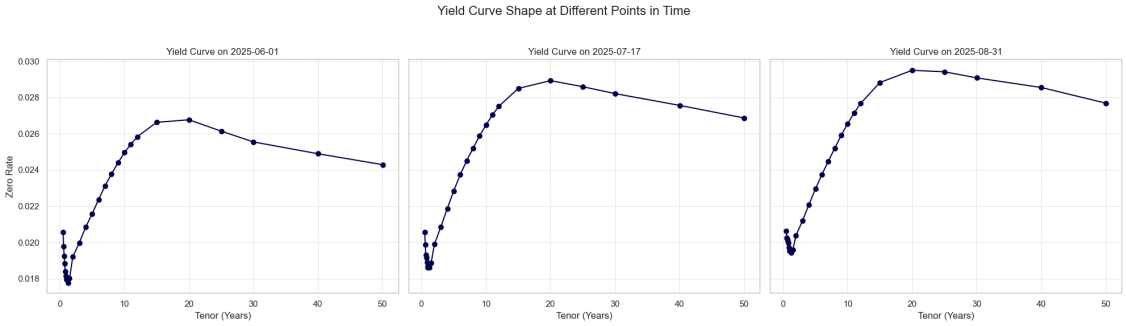

Figure 4: Yield Curve Snapshots over Time

Figure 4 visualizes static yield curve snapshots at different points in time. The yield curve consistently displays a concave, upward-sloping shape, typical for stable economic environments. The entire curve shifts upward from June to August, reinforcing the observed

trend of rising interest rates. Moreover, the curvature appears to increase slightly over time, suggesting subtle changes in the underlying term structure dynamics.

Table 4: Descriptive Statistics for Key Yield Curve Tenors

Tenor	Mean Rate (%)	Std. Dev. (%)	Skewness	Kurtosis
1-Year	1.90	0.05	-0.02	-1.10
5-Year	2.27	0.05	-0.18	-0.88
10-Year	2.62	0.06	-0.25	-1.04
30-Year	2.76	0.12	-0.29	-1.22

Descriptive statistics for selected benchmark tenors of the zero-coupon yield curve. Mean Rate and Standard Deviation are expressed in percentage points.

Table 4 quantifies the characteristics of the yield curve by presenting descriptive statistics for key benchmark tenors. The increasing mean rate with tenor numerically confirms the upward-sloping nature of the term structure. A particularly important finding is the behavior of the standard deviation, which increases from 0.05% for the 1-year rate to 0.12% for the 30-year rate. This demonstrates that the long end of the yield curve is substantially more volatile in absolute terms, a stylized fact that is further explored in figure 6.

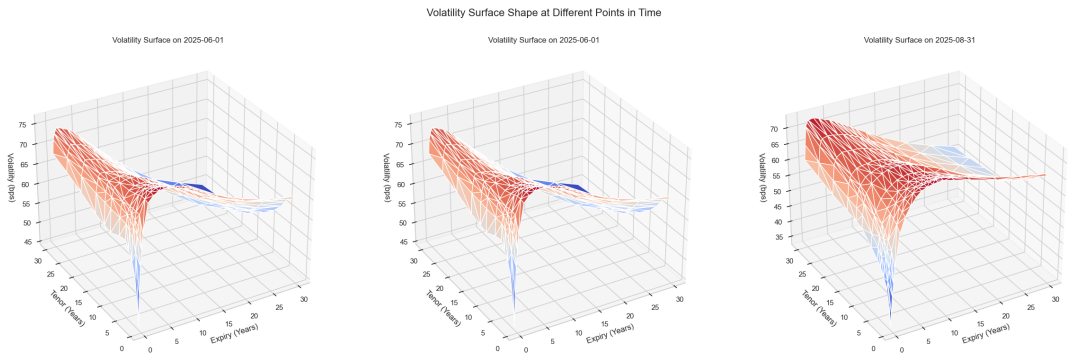

Figure 5: 3D Representation of the Swaption Volatility Surface

Figure 5 provides a three-dimensional representation of the swaption volatility surface. The surface exhibits pronounced structure, with the highest volatilities concentrated in short-expiry, short-tenor instruments. Volatility decreases along both the expiry and tenor dimensions, forming a characteristic “hump” at the short end. This structural complexity underscores the necessity of employing non-linear modeling approaches-such as NNs-to accurately capture the surface’s intricate shape and temporal dynamics.

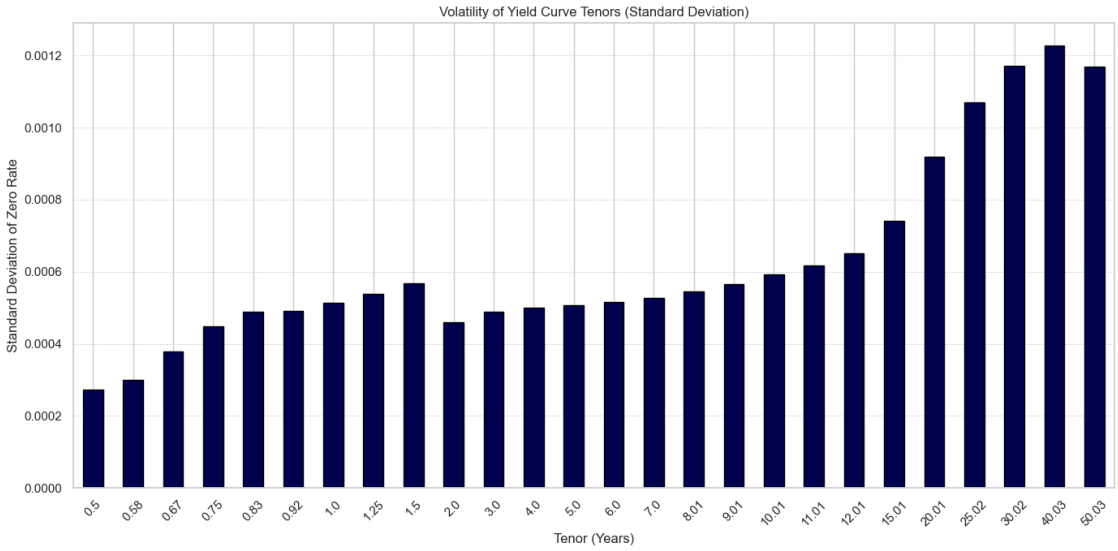

Figure 6: Standard Deviation of Daily Yield Changes by Tenor

Figure 6 shows the distribution of yield volatility across maturities. Volatility is lowest at the short end, increases moderately in the 1-2 year range, and rises significantly for longer maturities beyond 10 years. This pattern highlights that long-term yields exhibit the greatest absolute variability, a crucial consideration for risk management and model calibration.

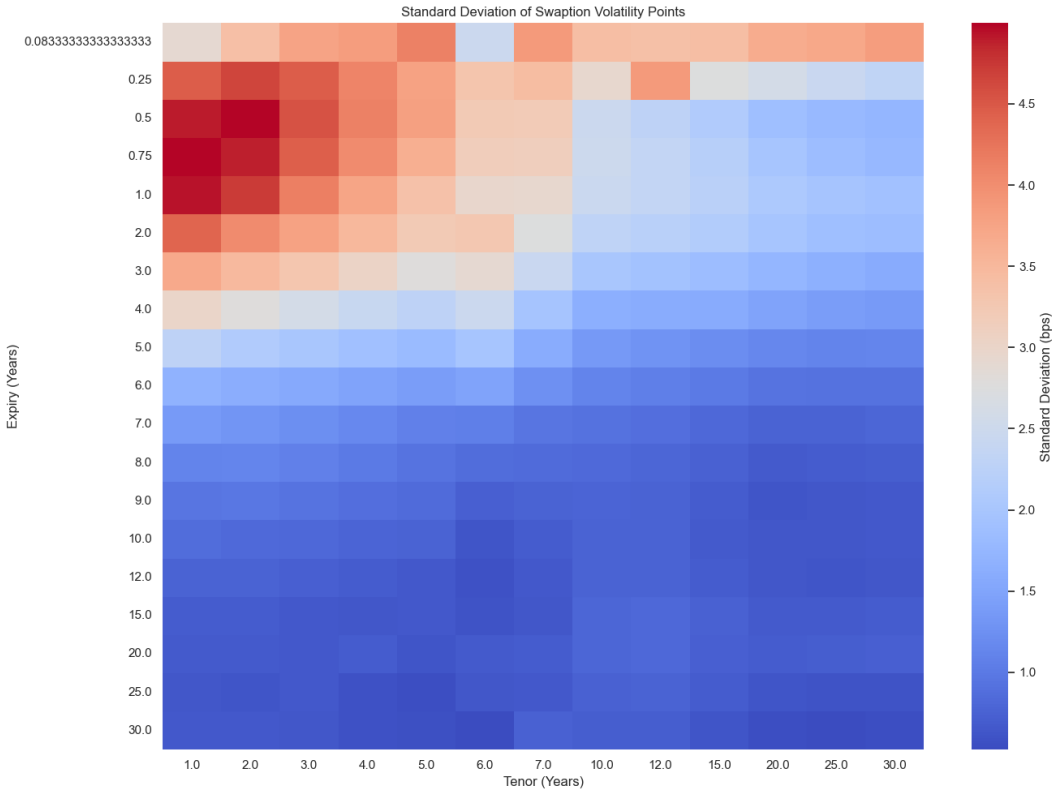

Figure 7: Heatmap of Swaption Volatility Surface Standard Deviations

Figure 7 displays the standard deviation of swaption volatilities across expiry and tenor

dimensions. The highest volatility, represented by the most intense red shading, is concentrated in the short-expiry, short-tenor region. This indicates that the front end of the surface is particularly unstable, while the long end remains relatively static. A well-performing model must therefore excel at capturing the pronounced temporal variability in this region.

Table 5: Standard Deviation of Swaption Volatilities (in bps)

Expiry	**1-Year Tenor**	**5-Year Tenor**	**10-Year Tenor**	**30-Year Tenor**
1-Year	4.93	3.37	2.45	1.92
5-Year	2.28	1.82	1.35	1.12
10-Year	0.86	0.75	0.76	0.65
30-Year	0.66	0.56	0.71	0.54

Standard deviation of daily swaption volatilities, measured in bps. The table shows the absolute volatility for key points on the expiry-tenor grid.

Table 5 provides a quantitative counterpart to the heatmap in figure 7, detailing the magnitude of instability across the volatility surface. The data confirms that the highest volatility is located at the front end, with the 1-year expiry, 1-year tenor swaption exhibiting a standard deviation of 4.93 bps. This value is nearly an order of magnitude larger than the 0.54 bps standard deviation observed for the 30-year expiry, 30-year tenor swaption. This sharp gradient in instability underscores the modeling challenge: the NN must learn to produce highly dynamic outputs for the front end of the surface while generating stable outputs for the back end.

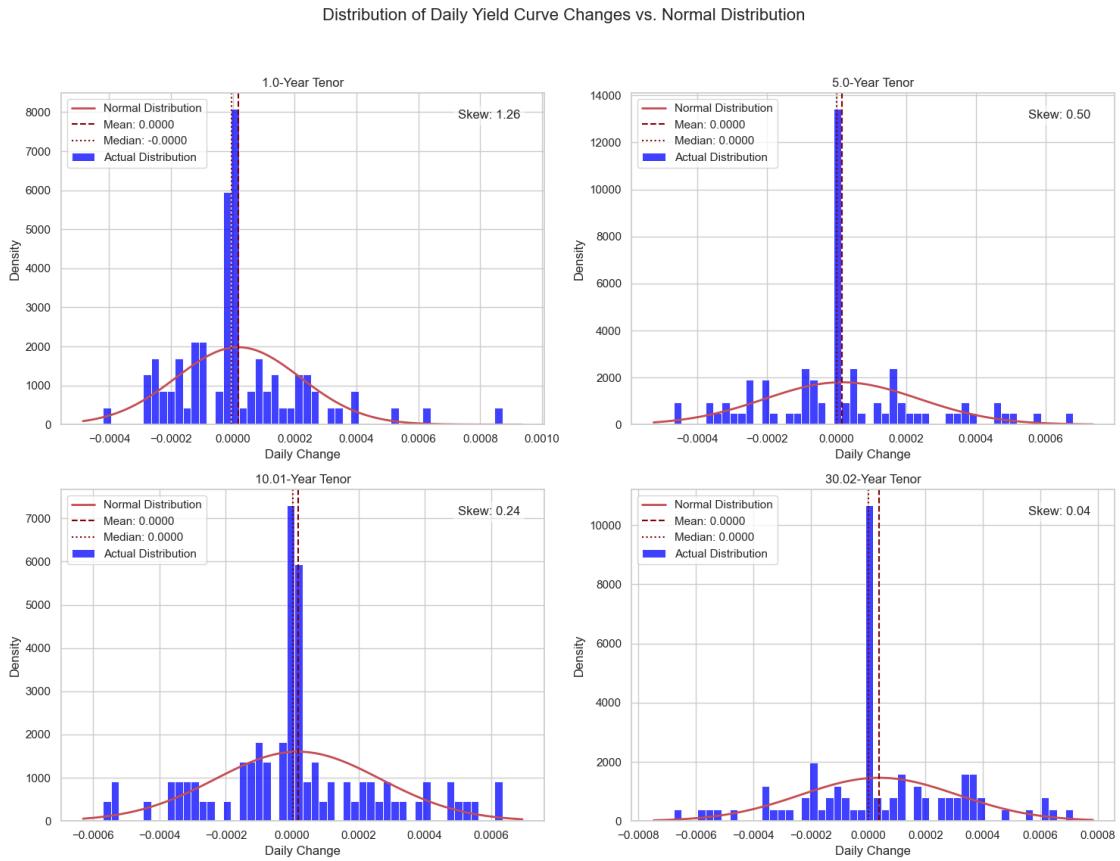

Figure 8: Histograms of Daily Yield Changes by Tenor

Figure 8 compares empirical histograms of daily interest rate changes with theoretical normal distributions. All tenors display leptokurtic distributions with fat tails, implying that extreme movements occur more frequently than predicted by Gaussian models. The 1-year tenor exhibits pronounced positive skewness (1.26), suggesting a tendency toward larger upward rate movements. Skewness gradually declines for longer maturities. These findings confirm that interest rate changes deviate substantially from normality.

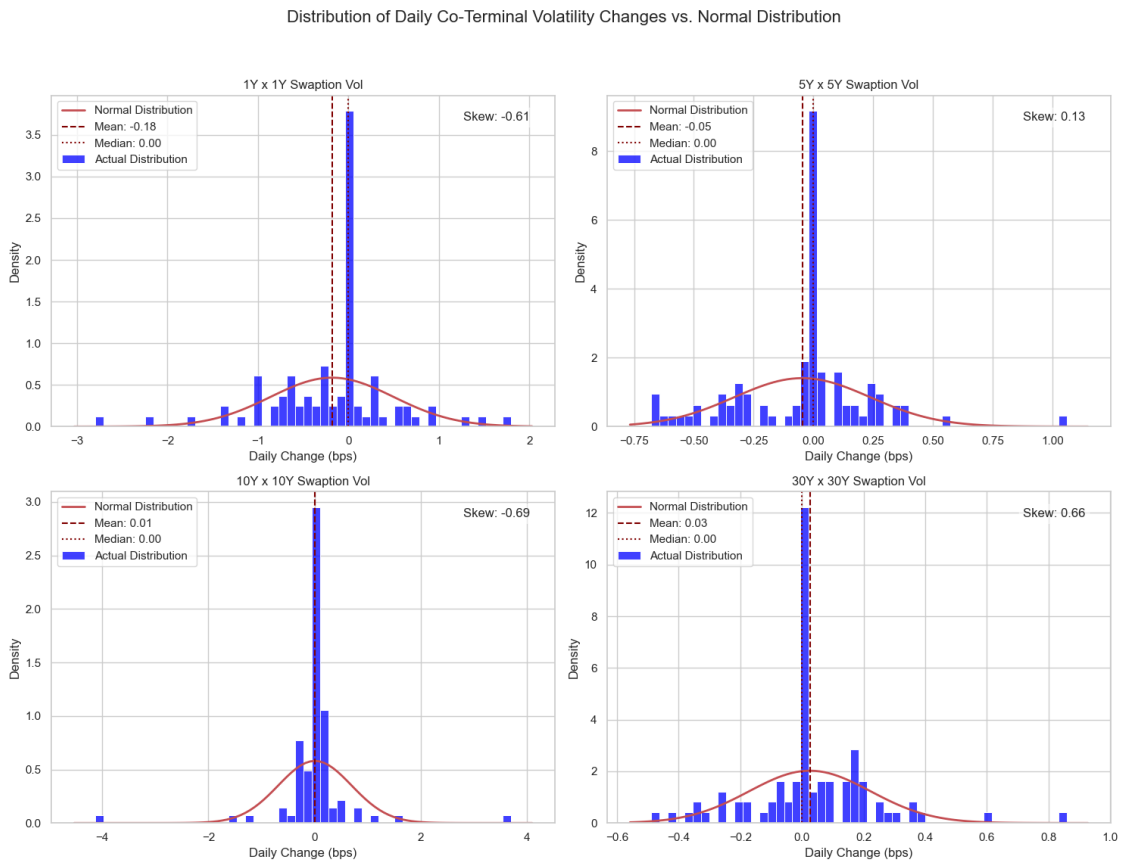

Figure 9: Histograms of Daily Swaption Volatility Changes

Figure 9 presents the distribution of daily changes in swaption volatilities. Similar to yield changes, the data exhibit significant leptokurtosis and fat tails, confirming the non-normal nature of the underlying process. A mild positive skew is particularly evident in the 5-year tenor (skew = 0.50), indicating that large volatility spikes are more frequent than declines. Together with the findings from Figure 8, this evidence supports the use of non-parametric, data-driven models-such as NNs-that can flexibly capture these empirical deviations from normality.

3.3 Data Preprocessing

3.3.1 Bootstrapping Zero-Curves

The implementation of the one-factor HW model in `QuantLib` requires zero-coupon curves as input for both discounting and forward rate generation. Consequently, the EUR swap curves obtained from Bloomberg were bootstrapped into continuous zero rate term structures following the procedure outlined in section 2.2. This ensures compatibility with the `QuantLib` framework and provides a consistent foundation for pricing swaptions and calibrating the model parameters.

3.3.2 Handling Non-Trading Days in External Data

The external market data obtained from Yahoo Finance, including the MOVE index, VIX, and EUR/USD exchange rate, is only available for trading days. In contrast, the swaption and swap curve datasets contain values for all calendar days, including weekends and public holidays. To ensure temporal consistency across all features used for HW model calibration, the external data were reindexed to cover the entire daily date range of the analysis period.

Missing values corresponding to non-trading days were filled using a combination of forward-filling and backward-filling. Forward-filling propagates the most recent available value to subsequent missing days, while backward-filling assigns the nearest subsequent available value to preceding missing days. This procedure ensures that every day within the observation window has a corresponding value for each external variable, resulting in a continuous dataset that can be used by the NN without introducing gaps or inconsistencies.

3.3.3 Feature Engineering for the Neural Network

In addition to the raw market data, several engineered features were created to enhance the NN's ability to predict HW model parameters. These features capture the shape of the yield curve, relationships between external market indicators, and underlying latent structures via dimensionality reduction. Each feature was designed with a clear economic or statistical justification for its relevance to the model parameters: the volatility σ and the mean-reversion speed a.

Features Engineered from the Yield Curve These features explicitly describe the shape and dynamics of the zero-coupon interest rate term structure. Let $\text{Rate}(T)$ denote the zero-coupon rate for a given tenor T (e.g., $\text{Rate}(10\text{Y})$ is the 10-year rate).

Slope Features:

$$\text{slope\_3m10y} = \text{Rate}(10\text{Y}) - \text{Rate}(3\text{M}) \tag{54}$$

$$\text{slope\_2y10y} = \text{Rate}(10\text{Y}) - \text{Rate}(2\text{Y}) \tag{55}$$

$$\text{slope\_5y30y} = \text{Rate}(30\text{Y}) - \text{Rate}(5\text{Y}) \tag{56}$$

A steep positive slope typically signals expectations of economic growth and future rate hikes, implying higher expected interest rate volatility (σ). Conversely, a flat or inverted

curve suggests a more stable rate environment and lower σ. Regarding mean reversion (a), a steep curve indicates strong anchoring to the long-term average, implying higher a, while an inverted curve signals weaker anchoring and lower a.

Curvature Features:

$$\text{curvature\_2y5y10y} = 2 \cdot \text{Rate}(5\text{Y}) - \text{Rate}(2\text{Y}) - \text{Rate}(10\text{Y}) \tag{57}$$

$$\text{curvature\_1y2y5y} = 2 \cdot \text{Rate}(2\text{Y}) - \text{Rate}(1\text{Y}) - \text{Rate}(5\text{Y}) \tag{58}$$

$$\text{curvature\_10y20y30y} = 2 \cdot \text{Rate}(20\text{Y}) - \text{Rate}(10\text{Y}) - \text{Rate}(30\text{Y}) \tag{59}$$

High curvature indicates significant uncertainty in medium-term rates, aiding prediction of the term structure of σ. Pronounced curvature suggests rates are expected to revert to a mean, implying higher a. Flat curvature indicates weaker mean reversion and lower a.

Features Engineered from External Market Data
Curvature-to-Slope Ratio:

$$\text{curvature\_slope\_ratio} = \frac{\text{curvature\_2y5y10y}}{\text{slope\_2y10y}} \tag{60}$$

This feature captures nuanced market regimes where slope and curvature provide complementary information. It allows the network to distinguish between different economic environments and predict both σ and a more accurately.

MOVE-to-VIX Ratio:

$$\text{MOVE\_VIX\_Ratio} = \frac{\text{MOVE\_Open}}{\text{VIX\_Open}} \tag{61}$$

A high ratio indicates stress concentrated in the bond market, signaling a higher σ specific to interest rates. It also implies uncertainty about central bank policy, potentially reducing a. This feature helps the network separate general market panics from fixed-income-specific volatility.

Features Engineered via Dimensionality Reduction Instead of feeding all individual zero-coupon rates directly into the NN, the yield curve was transformed using PCA. The reason for this choice lies in the fact that individual rates on the curve are highly correlated - a move in the 5-year rate is almost always accompanied by similar movements

in the 7-year and 10-year rates as visible in figure 10. Using these raw, correlated rates introduces several problems: multicollinearity, redundant information, and an increased risk of overfitting.

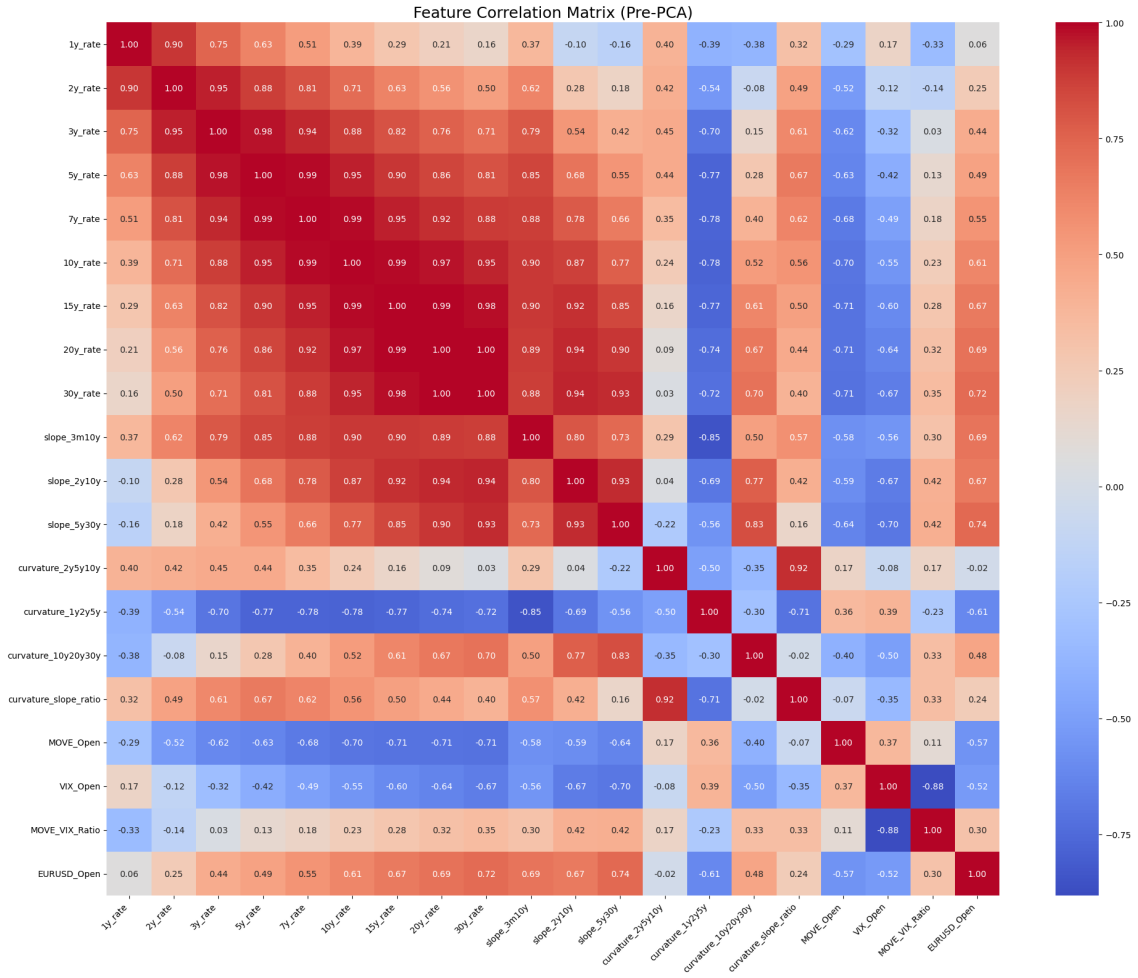

Figure 10: Correlation matrix of the engineered features before applying PCA.

Multicollinearity causes instability in the learned weights of the NN. When input features contain overlapping information, the model struggles to assign stable importance to each feature. It may assign large, offsetting weights to features that move together, resulting in unstable and unreliable predictions (Chan et al., 2022, p. 2).

Redundant information makes the learning process inefficient and typically leads to poor test performance (Sildir et al., 2020, p. 3). Most daily movements in the yield curve are dominated by a single pattern - a parallel shift. Feeding the model all the raw rates forces it to re-learn this shared structure repeatedly instead of focusing on economically meaningful dynamics such as slope or curvature changes.

High dimensionality further increases the complexity of the model, encouraging overfitting. With too many correlated features, the NN risks memorizing noise instead of generalizable relationships, which harms out-of-sample performance (Sildir et al., 2020,

p. 3).

PCA directly resolves these issues by transforming the vector of rates

$$R = [\text{Rate}(1\text{Y}), \text{Rate}(2\text{Y}), \ldots, \text{Rate}(30\text{Y})]$$

into a set of uncorrelated and economically interpretable components:

$$\text{PC\_Level} = \mathbf{w}_1 \cdot \mathbf{R} \tag{62}$$

$$\text{PC\_Slope} = \mathbf{w}_2 \cdot \mathbf{R} \tag{63}$$

$$\text{PC\_Curvature} = \mathbf{w}_3 \cdot \mathbf{R} \tag{64}$$

These three "super-features" capture nearly all the meaningful variation of the yield curve and can be interpreted as follows (Rebonato, 2018, pp. 98–107):

- PC_Level represents the overall interest rate level and captures parallel shifts. Empirically, higher rate levels are associated with higher volatility (σ).
- PC_Slope measures the steepness of the curve and reflects economic expectations, influencing both σ and a.
- PC_Curvature captures the non-linear "bow" of the curve, aiding the model in learning term-structure effects in the piecewise σ and mean-reversion dynamics α.

After applying PCA to the engineered features, the resulting correlation matrix exhibits a significant reduction in the highest correlations. The originally strong dependencies between individual yield curve rates and other features are largely removed as visible in figure 11, demonstrating that the principal components provide a set of largely uncorrelated, independent features for the NN.

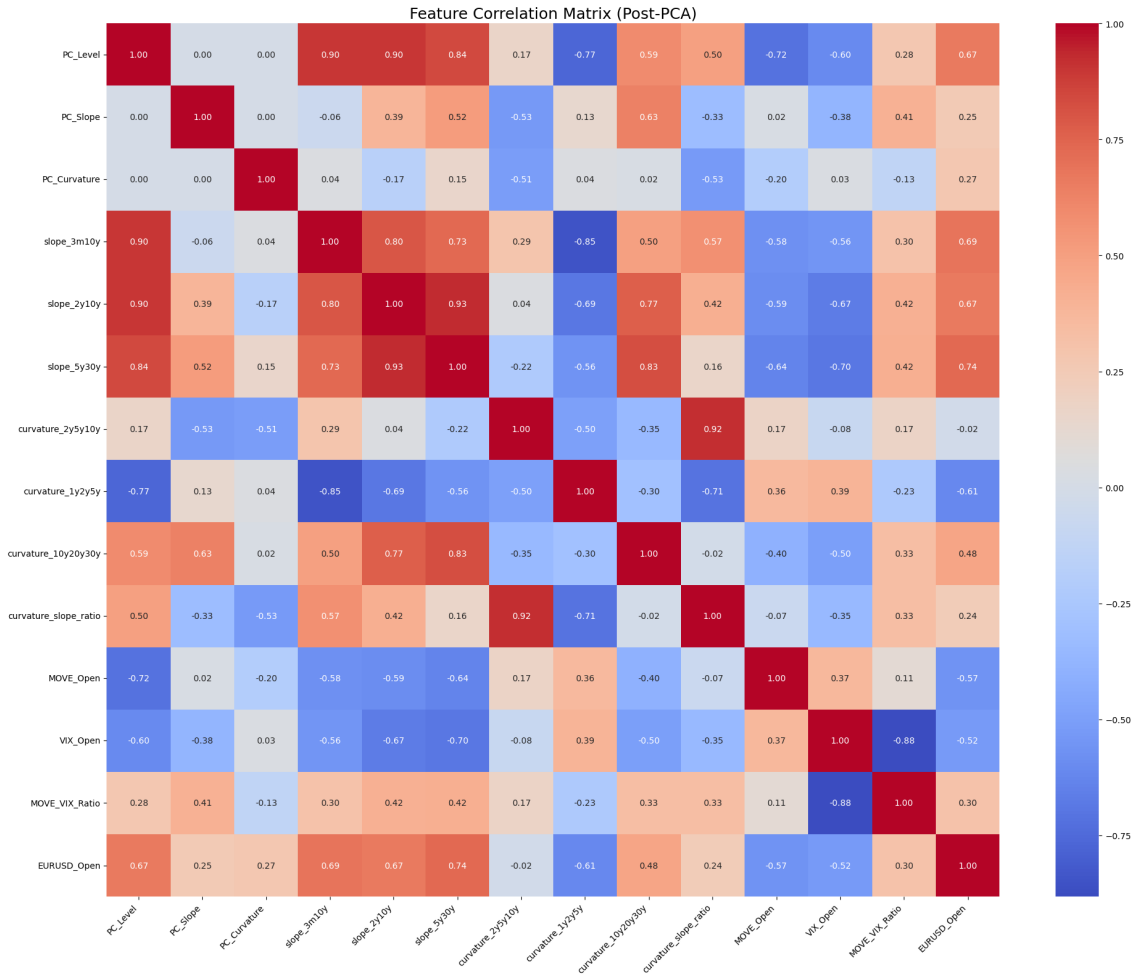

Figure 11: Correlation matrix of the engineered features after applying PCA.

By reducing the dimensionality from roughly ten correlated rate inputs to three independent components, PCA provides a stable, interpretable, and compact representation of the yield curve. This significantly improves the NN's training efficiency, robustness, and generalization performance.

3.3.4 Feature Scaling

Prior to being fed into the NN, all input features are scaled using the `StandardScaler` from the `scikit-learn` library. This process, also known as standardization or Z-score normalization, ensures that each feature has a mean of zero and a standard deviation of one (Zheng & Casari, 2018, pp. 31–32).

The scaling process consists of two main steps:

Step 1: Fit Phase (Learning from Training Data) The scaler analyzes the training data to learn the distribution of each feature. For each feature column, it calculates:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{65}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{66}$$

where x_i are the individual values of the feature and n is the number of training samples. The computed mean μ and standard deviation σ are stored in the scaler object.

Step 2: Transform Phase (Applying the Scaling) The learned parameters are used to transform all values of that feature in the training, validation, and test datasets:

$$z = \frac{x - \mu}{\sigma} \tag{67}$$

This centers the data around zero and scales it to unit variance.

From a computational perspective, normalization is necessary to manage issues arising from the inherent structure of raw data inputs. It is a technique widely used in statistics for making the units of variables comparable, which is critical when input data are expressed in very different units and exhibit disparate orders of magnitude. Without scaling, features with very large numerical values can dominate the error calculation, causing the error reduction algorithm to focus primarily on these variables while neglecting the information contained in smaller-valued features. Furthermore, when raw numerical values are extremely large, such as on the order of 10^6, their processing can generate outputs that exceed the capacity of standard computing hardware, making scaling essential to control for calculation and roundoff errors (Shanker et al., 1996).

The application of scaling directly addresses the optimization landscape of the training problem, leading to more efficient convergence. Normalizing the input is known to accelerate convergence during the optimization of linear models. Linear scale transformations, especially those that compress the search space, reduce the distance the backpropagation algorithm must cover in each iteration (Sola & Sevilla, 1997). This improved conditioning, where normalization helps ensure the covariance matrix of the layer input $\sum x$ is well-conditioned, is crucial for faster learning and aids the optimization algorithm in locating local minima (Huang et al., 2020). Consequently, an adequate normalization protocol can significantly reduce calculation and training time, with studies showing that estimation errors can be reduced by a factor of 5 to 10 and the time required to achieve such results reduced by an order of magnitude (Sola & Sevilla, 1997).

Ultimately, the primary result of standardization is an improvement in the NN's quality and training stability. Data transformation is frequently used for improved learning, and

normalization techniques are considered essential for accelerating the training and enhancing the generalization of deep NNs. NN trained on standardized data generally yield better overall results, leading to a higher classification rate and a smaller Mean Squared Error (MSE) in regression problems, where a smaller MSE indicates a higher quality solution (Shanker et al., 1996). The performance of a network, including the number of training iterations required and the final error attained, is enhanced as the input variable ranges are 'equalized' by the normalization process. This contributes to scale-invariant learning, which is important for stabilizing training and has been shown to be useful for adaptively adjusting the learning rate. In some cases, standardization is not merely a best practice but a formal prerequisite due to specific algorithm requirements (Huang et al., 2020).

By applying `StandardScaler`, the NN receives a well-conditioned, standardized input, which improves training stability, and convergence speed of the resulting model.

3.4 Hyperparameter Optimization

The predictive performance and stability of NNs depend critically on an appropriate choice of hyperparameters. To identify an optimal configuration, a structured hyperparameter optimization process was conducted using the Hyperband algorithm, as described in section 2.5.4. This algorithm efficiently allocates computational resources to explore a broad hyperparameter space while focusing on the most promising configurations.

The core settings for the Hyperband tuner were configured as follows:

- Maximum Epochs per Trial (`max_epochs`): A total of 1,000 epochs were allocated as the maximum budget for training any single hyperparameter configuration.
- Reduction Factor (`factor`): A factor of 3 was used, meaning that after each round of training within a bracket, the number of configurations is reduced by a factor of 3, retaining only the top-performing third.
- Objective: The optimization objective was the minimization of the Root Mean Squared Error (RMSE) on the validation set.

The hyperparameters tuned in this study can be divided into two categories: those defining the NN architecture and those related to the training process and loss function. Hyperparameters related to the architecture are:

- Number of Hidden Layers: This parameter determines the depth of the NN. A deeper NN has greater representational power and can capture more complex nonlinear relationships in the data, but it also increases the risk of overfitting and com-

putational cost.
Search Space: Integer between 1 and 5

- Number of Neurons per Layer: The width of each hidden layer, determined by the number of neurons, controls the capacity of the model. Larger layers allow for more detailed pattern recognition but can lead to overfitting if excessive.
  Search Space: Integer between 16 and 128, in steps of 16, for each hidden layer

- The activation function introduces non-linearity into the NN, enabling it to model complex relationships between input features and outputs.
  Search Space: Choice between `Rectified Linear Unit (ReLU)` and `tanh`.

- Dropout is a regularization technique that randomly deactivates a fraction of neurons during training to prevent overfitting by discouraging co-adaptation among neurons.
  Search Space: Boolean choice between `True` and `False`.

- Dropout Rate: If dropout is used, this parameter determines the fraction of neurons dropped during each training iteration.
  Search Space: Floating-point value between 0.1 and 0.5 (only active if `use_dropout = True`) in steps of 0.1.

Tuneable hyperparameters which cover the training and the loss functions are:

- Learning Rate: The learning rate governs the step size used by the optimizer when updating the NN weights. A high learning rate may lead to instability or divergence, whereas a very low one results in slow convergence. To capture effective values across orders of magnitude, the search was conducted on a logarithmic scale.
  Search Space: Floating-point value between 0.0001 and 0.01, sampled logarithmically.

- Underestimation Penalty: This custom hyperparameter is designed to address the asymmetric cost of forecasting errors in financial volatility modeling. Underestimating volatility poses a greater financial risk than overestimating it. Therefore, this penalty increases the loss when the model underpredicts volatility, encouraging more conservative estimates.
  Search Space: Floating-point value between 0.5 (penalty for overestimation) and 4.0 (strong penalty) in steps of 0.1.

3.5 Loss Function and Error Metric

The objective of the NN is to determine the optimal set of weights W and biases b such that the predicted HW parameters minimize the discrepancy between model-implied swaption

volatilities and their market-observed counterparts. The NN acts as a function

$$\mathrm{NN} : x \mapsto \theta(x; W, b), \tag{68}$$

where $x \in \mathbb{R}^d$ is the input feature vector containing scaled and PCA-transformed market data (yield curve levels, slopes, curvatures, etc.) for a given day, and θ denotes the vector of HW model parameters to be predicted. The NN employs a residual architecture: instead of predicting θ directly, it predicts a deviation Δz from a fixed initial guess z_0:

$$\Delta z = \mathrm{NN}(x; W, b), \quad z = z_0 + \Delta z. \tag{69}$$

The predicted parameters are obtained by applying a scaled sigmoid to ensure they remain within a predefined range [0, U]:

$$\theta(x; W, b) = U \cdot \sigma(z) = U \cdot \frac{1}{1 + e^{-z}}. \tag{70}$$

The predicted parameters θ are subsequently used within the QuantLib library as a black-box function V_{QL}, which computes model-implied volatilities $\hat{\sigma}$ for a portfolio of n swaptions given the additional market data M_j on day j:

$$\hat{\sigma} = V_{\mathrm{QL}}(\theta, M_j) = \{\hat{\sigma}_1, \hat{\sigma}_2, \ldots, \hat{\sigma}_n\}. \tag{71}$$

The predictive accuracy of the network is quantified using the RMSE, which measures the average magnitude of deviation between model-implied and market-observed volatilities, expressed in bps:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{\sigma}_i - \sigma_i)^2} \times 10000. \tag{72}$$

Table 6: Notation used in the RMSE formula

Symbol	Meaning
n	Total number of swaptions in the evaluation dataset
$\hat{\sigma}_i$	Implied volatility predicted by the HW model using the parameters generated by the NN for the i-th swaption
σ_i	Market-observed implied volatility for the i-th swaption

To align the optimization with practical calibration objectives, an asymmetric loss function is employed during training. For each day j, the loss is computed as the mean of

weighted squared errors between predicted and observed volatilities:

$$L_j(W,b) = \frac{1}{n}\sum_{i=1}^{n} w_i(\hat{\sigma}_i - \sigma_{j,i})^2, \tag{73}$$

where the weight w_i accounts for underestimation penalties:

$$w_i = \begin{cases} P, & \text{if } \hat{\sigma}_i < \sigma_{j,i} \quad \text{(underestimation)}, \\ 1, & \text{if } \hat{\sigma}_i \geq \sigma_{j,i} \quad \text{(overestimation)}. \end{cases} \tag{74}$$

Here, $P > 1$ is the underestimation penalty, treated as a hyperparameter optimized using the Hyperband algorithm. This asymmetric weighting is applied exclusively during training to encourage conservative predictions, while evaluation on validation and test sets uses the standard (unweighted) RMSE to assess predictive accuracy objectively.

The overall training objective is to identify network parameters (W^*, b^*) that minimize the expected daily loss over the distribution of training data D:

$$(W^*, b^*) = \arg\min_{W,b} \mathbb{E}_{(x_j,\sigma_j,M_j)\sim D}\left[L_j(W,b)\right]. \tag{75}$$

In practice, this expectation is approximated by iterating over the training set day by day (batches) using the stochastic optimizer Adam:

$$W_{t+1} = W_t - \eta \nabla_W L_j(W_t, b_t), \tag{76}$$

where η is the learning rate. Since V_{QL} is a non-differentiable black-box function, gradients are computed numerically via finite differences using a tf.custom_gradient implementation.

3.6 Conversion from Normal to Black Volatility Errors

In interest rate derivatives markets, two primary volatility conventions are employed for pricing and risk management: the normal (Bachelier) volatility and the lognormal (Black) volatility. The distinction between these conventions arises from the assumed stochastic dynamics of the underlying forward rate or price process. The conversion between both representations is essential when comparing model calibration errors or implied volatilities expressed under different modeling assumptions, as is the case in the present study.

Under the Bachelier (normal) model, the forward rate F_t evolves according to a normal diffusion process,

$$dF_t = \sigma_N \, dW_t, \tag{77}$$

where σ_N denotes the normal volatility and W_t is a standard Brownian motion. In this

framework, the forward rate can take both positive and negative values, making the model particularly suitable for low or negative interest rate environments. The option price under the Bachelier model is linear in volatility, and σ_N represents an absolute volatility level (for example, 0.0050 corresponds to 50 bps).

In contrast, the Black (lognormal) model assumes that the forward rate follows a lognormal diffusion process,

$$dF_t = \sigma_B F_t \, dW_t, \tag{78}$$

where σ_B represents the lognormal or Black volatility. Here, the volatility is expressed in relative terms (e.g., 0.20 corresponds to 20%), and the forward rate is strictly positive. The lognormal assumption is particularly useful when modeling multiplicative dynamics in markets with positive rates.

For ATM swaptions, where the strike K equals the forward rate F, the two pricing frameworks yield nearly equivalent option values if the volatilities satisfy the following approximate relationship (Hagan et al., 2002):

$$\sigma_N \approx \sigma_B F. \tag{79}$$

Rearranging this yields an approximation for converting normal volatility to lognormal volatility:

$$\sigma_B \approx \frac{\sigma_N}{F}. \tag{80}$$

This relationship provides an accurate first-order approximation for ATM instruments and is commonly applied in market practice for fast conversions between volatility conventions.

When assessing model calibration performance, the model error expressed in normal volatility terms, denoted by ε_N, can therefore be converted into an equivalent Black volatility error for each swaption i with forward rate F_i as

$$\varepsilon_{B,i} = \frac{\varepsilon_N}{F_i}. \tag{81}$$

This conversion enables a consistent comparison with benchmark results, such as those reported in studies or market data expressed in Black volatility terms.

However, aggregating these individual instrument errors to obtain a portfolio-level measure requires careful consideration. A simple average of volatility errors, while statistically valid, would be financially naive, as it assumes that a one-basis-point error has the same significance for every swaption. In practice, the financial impact of a volatility misestimation varies dramatically across instruments, depending on their sensitivity to volatility.

This sensitivity is precisely quantified by the instrument's vega. As established in foundational derivatives literature, the first-order approximation for the change in an option's price (ΔV) due to a change in volatility ($\Delta\sigma$) is given by the relationship $\Delta V \approx \nu \cdot \Delta\sigma$ (J. C. Hull, 2015, pp. 415-417). In the context of this analysis, the model's normal volatility error, ε_N, represents the $\Delta\sigma$ term. Consequently, the pricing error, or P&L impact, for a given swaption i is approximately $\nu_i \cdot \varepsilon_N$. The vega for swaption i is formally defined as:

$$\nu_i = \frac{\partial V_i}{\partial \sigma_N}. \tag{82}$$

In the present context, vegas are computed under the Bachelier model, consistent with the representation of the input errors in normal volatility terms. The scientific rationale for weighting individual errors by vega is grounded in the fundamental principles of differential calculus as applied to derivative pricing. A model's error in the volatility dimension, ε_N, does not translate uniformly to pricing error across all instruments. Instead, the first-order approximation of the pricing error, ΔV, is directly proportional to the volatility error, scaled by the instrument's vega: $\Delta V \approx \nu \cdot \varepsilon_N$.

Vega therefore acts as a linear sensitivity coefficient, converting an abstract error measured in basis points of volatility into a tangible error measured in monetary units. An unweighted average of volatility errors would be statistically valid but financially naive, as it would implicitly assume that a one-basis-point error on a swaption with high vega has the same financial significance as the same error on a swaption with low vega. To construct a portfolio-level error metric that accurately reflects the aggregate financial impact of individual mispricings, it is therefore necessary to weight each volatility error by its corresponding vega. This ensures the resulting metric moves beyond a simple statistical average to provide a more robust and financially meaningful assessment of the model's performance, which is essential for its utility in risk management and hedging.

Using the swaption vegas as weighting factors is the direct implementation of this principle. To obtain a single, representative portfolio-level Black volatility error, a vega-weighted average is computed across all n swaptions. This ensures that instruments with a higher price sensitivity to volatility contribute proportionally more to the aggregate measure. The aggregated Black volatility error $\bar{\varepsilon}_B$ is given by:

$$\bar{\varepsilon}_B = \frac{\sum_{i=1}^{n} \varepsilon_{B,i} \nu_i}{\sum_{i=1}^{n} \nu_i}. \tag{83}$$

Substituting the expression for $\varepsilon_{B,i}$ yields:

$$\bar{\varepsilon}_B = \frac{\sum_{i=1}^{n} \left(\frac{\varepsilon_N}{F_i} \nu_i \right)}{\sum_{i=1}^{n} \nu_i}. \tag{84}$$

Since the normal-volatility-based model error ε_N is constant across swaptions on a given

day, it can be factored out:

$$\bar{\varepsilon}_B = \varepsilon_N \frac{\sum_{i=1}^{n} \frac{\nu_i}{F_i}}{\sum_{i=1}^{n} \nu_i}. \tag{85}$$

This formula represents the vega-weighted average conversion from normal to lognormal volatility on the portfolio level. Conceptually, this approach yields a portfolio-consistent transformation of model errors that is grounded in financial reality, allowing for a robust comparison of calibration quality under different modeling frameworks.

Having established the input structure and metrics, the next step outlines the full end-to-end workflow.

3.7 Methodology Workflow

The subsequent section delineates the comprehensive, step-by-step workflow implemented for the calibration of the HW model, as depicted by the provided source code. The process is designed as an end-to-end pipeline, commencing with raw data ingestion and culminating in model evaluation, adaptive retraining, and interpretability analysis.

1. **Data Ingestion and Preprocessing:** The initial phase involves loading raw EUR swap curve data, provided in Excel format. For each historical date, the mid-market rate is computed from the bid and ask quotes. These raw swap curves are then processed and transformed into a standardized format. Subsequently, the `QuantLib` library is employed to bootstrap these swap curves, yielding the corresponding zero-coupon yield curves for each valuation date. This bootstrapping process utilizes a `PiecewiseLogLinearDiscount` methodology to construct a term structure of zero rates from the observed market swap rates.

2. **External Market Data Integration:** To enrich the feature set, external market data is loaded. This includes the MOVE index, VIX and the EUR/USD foreign exchange rate. This data is sourced via the `yfinance` library. A complete time series is ensured by forward- and back-filling any missing daily values, providing a consistent set of external features for every yield curve date.

3. **Chronological Data Splitting:** Given the time-series nature of the financial data, a strict chronological split is enforced to prevent look-ahead bias. The dataset of available daily yield curves is divided into a training set (the initial 50%), a validation set (the following 20%), and a final out-of-sample testing set (the remaining 30%). This ensures that the model is trained on past data, validated on more recent data, and finally tested on unseen future data.

4. **Feature Engineering and Dimensionality Reduction:** For each yield curve, a comprehensive feature vector is engineered. This vector includes not only the zero

rates at key tenors (1Y to 30Y) but also derived features such as various yield curve slopes (e.g., 10Y-2Y), curvatures (e.g., $(2 \times 5\text{Y}) - 2\text{Y} - 10\text{Y}$), and the external market indicators. To address the high multicollinearity inherent in yield curve rates, PCA is performed exclusively on the rate-based features of the training data. The first three principal components, are retained. This reduces dimensionality while preserving the most significant sources of variance. The complete feature set, comprising these principal components and the other engineered features, is then standardized using Scikit-learn's `StandardScaler`, which is also fitted solely on the training data.

5. **Hyperparameter Optimization:** A rigorous hyperparameter tuning process is conducted using the Hyperband algorithm, implemented via the `Keras Tuner` library. The search space includes critical model architecture parameters (number of layers, neurons per layer, activation function, dropout usage and rate), optimization parameters (learning rate), and a custom hyperparameter for an asymmetric loss function (underestimation penalty). The objective function for the tuner is the minimization of the RMSE on the validation set.

6. **Model Training and Early Stopping:** Upon identifying the optimal set of hyperparameters, the final model is retrained from scratch on the entire training set. A key architectural choice is the `ResidualParameterModel`, which does not predict the HW parameters directly but rather learns a residual correction to a predefined initial guess. Training is performed over a set number of epochs, but an early stopping mechanism is employed. The model's performance on the validation set is monitored after each epoch. If the validation RMSE does not improve for a predefined number of consecutive epochs (patience), the training is halted, and the model weights from the epoch with the lowest validation RMSE are restored. This prevents overfitting to the training data.

7. **Out-of-Sample Evaluation:** The trained model is evaluated on the unseen testing set in a forward-moving, chronological sequence. For each day in the test set, the model predicts the HW parameters, and the resulting calibration error (RMSE) is calculated.

8. **Model Explainability with Shapley Additive Explanations (SHAP):** To ensure transparency and interpretability of the final trained model, a post-hoc analysis is conducted using SHAP. A `DeepExplainer` is utilized to compute SHAP values for the model's predictions on the test set. This analysis quantifies the contribution of each input feature (e.g., PC Level, MOVE index) to the prediction of each output HW parameter. The results are visualized through summary and feature importance plots, providing crucial insights into the model's decision-making process.

3.8 Calibration Strategy

The calibration methodology implemented in this thesis aligns with the *simultaneous optimization* strategy discussed in section 2.4.5. The NN is configured to concurrently optimize all dynamic parameters of the HW model within a single, unified optimization routine.

Specifically, the approach treats the mean-reversion parameter, α, as a single constant value and the volatility parameter, $\sigma(t)$, as a piecewise-constant function with seven distinct time segments. The NN's output layer predicts the complete set of these eight parameters. During the training loop, the custom gradient calculation within the `_perform_training_step` function computes the gradients of the loss function with respect to all eight parameters simultaneously. The optimizer then updates all model weights to jointly minimize the pricing error (RMSE), thereby fitting both α and $\sigma(t)$ at the same time. This is in contrast to a two-step approach where parameters are estimated sequentially or a fixed mean-reversion approach where α is not optimized at all.

Furthermore, the calibration employs a *global (full matrix) calibration* methodology with the exception of swaptions which have a maturity and tenor of less than 2 years. This global approach provides the model with a comprehensive view of the market, enabling it to achieve a more robust and stable fit across the entire volatility surface, as recommended for empirical analyses in section 2.4.5.

3.9 Comparison of Neural Network and Levenberg-Marquardt Calibration

A robust experimental framework was designed and implemented to enable a scientifically valid comparison between the traditional LM optimization and the NN predictor. The primary objective of this framework was to address the fundamental methodological challenge of fairly comparing an in-sample fitter with an out-of-sample predictor.

The core issue arises from the differing nature of the two models. The LM algorithm operates as an in-sample fitter: for a given day, it receives the complete set of that day's market-observed swaption volatilities and determines the HW parameters (a, σ) that minimize the pricing error for this exact set. Its performance metric, typically the RMSE, is computed against the same data used for calibration, reflecting the model's goodness-of-fit. This approach is prone to overfitting, as the resulting parameters may capture noise and idiosyncrasies specific to the input data rather than generalizable features. In contrast, the NN is designed as an out-of-sample predictor. It is trained on historical data to learn a mapping from observable market features, such as yield curve shapes and volatility indices, to the corresponding optimal HW parameters. On a test day, it receives only these market features and predicts parameters that are evaluated against the previously unseen

swaption volatilities, making its error a measure of generalization ability.

To reconcile this methodological inconsistency, an *Intra-Day Hold-Out Set* framework was employed. This protocol enforces a uniform out-of-sample evaluation for both models. Each day within the test period was processed as follows. First, all valid swaptions were assembled into a master list and partitioned into calibration and hold-out sets using stratified random sampling, with the swaption expiry date as the stratification variable. Within each expiry stratum, 80% of the instruments were assigned to the calibration set and the remaining 20% to the hold-out set. Single-instrument strata were assigned to the calibration set to ensure sufficient data for LM optimization, and the random sampling was controlled by a fixed seed to guarantee reproducibility.

Once the data partitioning was completed, both models generated their HW parameters under controlled conditions. The pre-trained NN produced its parameters via a forward pass using only the day's market features, whereas the LM algorithm was restricted to the 80% calibration set and iteratively optimized the parameters to fit this subset. Wall-clock times were recorded for both processes.

Finally, both sets of parameters were evaluated on the identical 20% hold-out set. Each model was used to price the hold-out swaptions, and the RMSE between model-implied volatilities and observed market volatilities was computed. This uniform evaluation enabled a direct and fair comparison of the NN's predictive capabilities against the LM's in-sample fitting performance, effectively quantifying their relative generalization power in unseen market conditions.

It is critical to acknowledge that this study's experimental protocol evaluates the LM algorithm, an in-sample optimizer, against the NN, a predictive model, using a unified out-of-sample generalization criterion. While the LM method's primary function is to achieve a minimal error on a given set of instruments, the fundamental purpose of any model calibration is to derive parameters that accurately represent the underlying market dynamics. Consequently, a successful calibration must yield parameters capable of consistently pricing related financial instruments beyond the immediate calibration set. An optimizer that perfectly fits one portfolio of swaptions but fails to price another is of limited practical utility. The *Intra-Day Hold-Out Set* framework is explicitly designed to quantify this precise generalization capability, which is an essential, albeit implicit, requirement for any robust calibration tool.

Therefore, this comparison does not contest the LM algorithm's efficacy for pure in-sample fitting. Instead, it provides a rigorous assessment of each method's ability to produce robust, generalizable parameters suitable for broader applications in pricing and risk management.

3.10 Technical Setup and Computational Optimization

The implementation of the NN-based HW calibration framework was carried out in Python 3.12.3, employing TensorFlow 2.15 for the NN architecture and QuantLib 1.34 for the financial modeling components. All experiments were conducted on a commercially available laptop equipped with 32 GB of RAM and an AMD Ryzen 7 7840HS processor (16 cores, approximately 3.8 GHz), operating under Windows 11.

3.10.1 Integration of TensorFlow and QuantLib

A central technical challenge of this research lies in the hybrid nature of the model architecture, which combines the differentiable TensorFlow framework with the non-differentiable pricing engines of QuantLib. Since QuantLib's pricing routines are implemented in C++ and do not expose analytical gradients, the NN cannot rely on TensorFlow's automatic differentiation. To overcome this limitation, the QuantLib-based loss function was implemented within a function decorated by TensorFlow's `@tf.custom_gradient`. This mechanism allows the explicit definition of a custom backward pass, where the gradients are computed manually rather than automatically propagated by TensorFlow.

The gradients with respect to the NN outputs are approximated numerically using finite difference methods. For each model parameter, the loss function is evaluated twice - once with a positive and once with a negative perturbation - yielding an approximation of the partial derivatives. Consequently, each gradient step requires $2 \times (\text{number of parameters})$ complete QuantLib evaluations. This approach is computationally demanding but necessary to correctly propagate gradients through the non-differentiable pricing engine.

3.10.2 Parallelization and Hardware Considerations

To mitigate the computational cost associated with the numerical gradient computation, the implementation leverages Python's `concurrent.futures.ThreadPoolExecutor` to parallelize function evaluations across the 16 available CPU cores. Thread-based parallelization is preferred over process-based parallelization because it introduces less communication overhead and efficiently shares memory among threads, which is particularly advantageous for memory-intensive but moderately CPU-bound pricing routines. In practice, this parallelization leads to a near-linear reduction in computation time with respect to the number of threads, thereby rendering the training process computationally feasible.

All training was executed on the CPU. This decision was motivated by the observation that the primary computational bottleneck lies not in the NN's forward propagation-which would benefit from GPU acceleration-but in the iterative, conditional, and sequential nature of QuantLib's pricing algorithms. Such algorithms are inherently unsuited to GPU architectures, which excel in parallel matrix operations but perform poorly with branching

control flow. Re-implementing QuantLib's pricing models in a GPU-compatible framework would require prohibitive effort while yielding minimal performance benefits. Although frameworks such as *TF-Quant-Finance* aim to provide GPU-compatible financial modeling tools, they currently lack active maintenance and do not support GSR models with piecewise constant volatility-an essential feature for this research.

3.10.3 Algorithmic and Numerical Efficiency

Further acceleration was achieved through algorithmic refinements and numerical efficiency improvements. The numerical derivative used in gradient computation can be approximated either by the forward difference or the central difference scheme. The forward difference requires only one additional model evaluation per parameter and is therefore approximately twice as fast as the central difference method, albeit with slightly higher numerical noise. Given the high computational cost of each QuantLib evaluation, the forward difference scheme was selected as the default method for gradient approximation. Empirical tests confirmed that the increased noise did not significantly impair training stability or convergence, making this a worthwhile trade-off for speed.

Additionally, the precision of numerical integration within the QuantLib pricing engine was adjusted through the `pricing_engine_integration_points` parameter, which controls the number of Gaussian quadrature points used by the `Gaussian1dSwaptionEngine`. Reducing the number of integration points lowers computational cost per pricing call but may introduce marginally higher numerical error. The chosen configuration (32 integration points) represents a compromise between computational tractability and pricing stability.

3.10.4 Optimization of the Training Process

Several optimization strategies were applied to enhance convergence speed and stability. An *early stopping* criterion monitors the validation RMSE after each epoch and terminates training once no improvement is observed over a predefined patience period (20 epochs). This approach prevents overfitting and avoids unnecessary computation.

In addition, the concept of *instrument batching* was introduced to reduce the computational burden per training iteration. Rather than evaluating all available swaptions at once, the loss function is computed on randomly selected subsets of instruments, controlled by the parameter `instrument_batch_size_percentage`. This stochasticity not only accelerates each iteration but also introduces beneficial noise, which helps the optimizer escape local minima and improves generalization. Due to the small size of the dataset, all instruments were used in each batch (100%) for the final training run as well as for the hyperparameter tuning to ensure maximum information utilization.

Hyperparameter optimization was conducted using the Hyperband algorithm (see section 2.5.4). Hyperband allocates computational resources adaptively by evaluating a large number of configurations for a limited number of epochs and retaining only the most promising candidates for further training. This method significantly reduces the computational time required for hyperparameter tuning while maintaining robustness in identifying high-performing parameter configurations.

3.10.5 Workflow-Level Enhancements and Model Design

To further enhance efficiency, all input data-such as bootstrapped zero curves and volatility cubes-were preloaded into system memory prior to model training. This design choice eliminates repeated disk input/output operations, minimizing latency and enabling smoother training iterations.

Moreover, the preprocessing pipeline was designed modularly, allowing selective execution of computationally expensive routines. Boolean flags such as `PREPROCESS_CURVES` and `BOOTSTRAP_CURVES` allow bypassing redundant computations once intermediate data has been stored, thereby facilitating efficient experimentation and reproducibility during hyperparameter searches.

Finally, the NN was implemented as a residual model, predicting adjustments to an initial parameter estimate rather than the parameters directly. If the initial parameter vector is denoted by θ_{initial} and the predicted correction by $\Delta\theta_{\text{predicted}}$, the final parameter estimate is given by

$$\theta_{\text{final}} = \theta_{\text{initial}} + \Delta\theta_{\text{predicted}}. \tag{86}$$

This residual formulation simplifies the learning task, as the NN learns to approximate local corrections rather than the entire nonlinear mapping between market features and model parameters. Empirically, this leads to smoother loss surfaces, faster convergence, and improved numerical stability.

Table 7: Initial Guess Parameters for NN Calibration of the HW Model

Parameter	Initial Guess
α	0.02000
σ_1	0.00020
σ_2	0.00020
σ_3	0.00017
σ_4	0.00017
σ_5	0.00017
σ_6	0.00017
σ_7	0.00017

The initial parameter guess can be derived from various sources, such as the parameters from the previous trading day or those obtained from a traditional calibration method. This flexibility allows the NN to leverage prior knowledge and enhances its adaptability to changing market conditions. In this study, the initial guess was set to fixed values based on a calibration of the first day in the training set using the LM algorithm. Then, the values have been rounded to avoid that the NN get "stucked" in the same local minimum as the LM algorithm.

In summary, the proposed hybrid framework integrates TensorFlow's differentiable learning capabilities with QuantLib's high-fidelity financial pricing models through custom gradient definitions and parallelized numerical differentiation. Combined with algorithmic, workflow, and training-level optimizations, these measures ensure that the training process remains computationally feasible, stable, and reproducible on commercially available hardware.

3.11 Use of Artificial Intelligence

Artificial Intelligence (AI) was employed in the context of this master thesis to support the author in various auxiliary and editorial tasks. Its use was strictly limited to non-critical and non-confidential aspects of the research and writing process. The primary purpose of employing AI tools was to enhance the overall clarity, coherence, and grammatical accuracy of the text, ensuring a consistent academic writing style. Through text proposals and linguistic adjustments, the readability of the thesis was improved, allowing for a clearer communication of complex concepts to the reader. AI was additionally utilized for translation purposes between English and German, ensuring linguistic precision in both directions.

Furthermore, AI-assisted summarization techniques were employed to help the author better understand complex theoretical or methodological concepts by providing concise explanatory summaries. These summaries served solely as a comprehension aid and were not directly included in the final version of the thesis without the author's verification and adaptation.

In the technical part of the thesis, AI was used to generate and refine Python code fragments, as well as to enhance existing code written by the author in terms of readability and computational efficiency. Code completion tools were occasionally used to accelerate programming during the development process. All generated or optimized code was manually integrated into the code base following review and validation by the author. AI tools were also employed for the generation of LaTeX table code structures, while the underlying content was entirely created by the author, and for querying specific LaTeX commands to ensure accurate formatting. In addition, AI was used to generate concise and descriptive Git commit messages for efficient version control and documentation of

code changes.

At no point were personal data, business or trade secrets, or any proprietary data, particularly data downloaded from Bloomberg, processed or shared with any AI tool. The handling of such data remained entirely under the author's control and within secure environments compliant with data protection and confidentiality standards.

All textual recommendations and code suggestions provided by AI tools were thoroughly reviewed and verified by the author for technical correctness and factual accuracy. This included empirical testing of all code outputs to ensure functionality and to prevent the inclusion of any erroneous or fabricated content, often referred to as "hallucinations." The author assumes full responsibility for the correctness and validity of all content and code presented in this thesis.

AI was explicitly not used beyond the purposes listed above. In particular, it was not involved in the conceptualization of the core logic of the calibration procedure for the NN, the overall structure of the code as well as the written thesis, the scientific argumentation, or the conceptualization of figures and visualizations.

A comprehensive list of all AI tools used during the preparation of this thesis is provided in the references section 15.

4 Results

Building upon the comprehensive methodology detailed previously, this chapter presents the empirical findings of the comparative analysis. The discussion is structured to build from foundational model validation to the core performance comparison. It begins by interpreting the results of the PCA on the yield curve, validating the feature engineering process. This is followed by a detailed analysis of the hyperparameter tuning phase, which identified the optimal architecture for the NN model.

With the models established, the chapter proceeds to the central, multi-faceted comparison between the predictive NN and the traditional LM optimizer. This evaluation is structured around the key criteria of out-of-sample pricing accuracy, the stability and economic interpretability of the derived model parameters under simulated market shocks, and the computational efficiency measured by calibration runtime.

To address the common criticism of machine learning models as "black boxes", a dedicated analysis of the NN's interpretability is presented using SHAP values, revealing the key market drivers behind its predictions. Finally, the results are contextualized through a direct comparison with the benchmark study by Hernandez (2016), positioning this thesis's findings and contributions within the existing body of research.

4.1 Interpretation of Principal Component Analysis on the Yield Curve

The application of PCA to the yield curve provides a powerful framework for decomposing the complex dynamics of interest rate movements into a small number of uncorrelated factors. The empirical results of the analysis indicate that the first three principal components collectively explain 99.84% of the total variance observed in daily yield curve changes. This finding confirms that the term structure of interest rates can be effectively described by a limited set of underlying factors, each representing a distinct pattern of yield curve movement.

Table 8: Explained Variance of the Principal Components

Principal Component	Explained Variance (%)	Cumulative Variance (%)
PC1 (Level)	91.08	91.08
PC2 (Slope)	8.16	99.24
PC3 (Curvature)	0.59	99.84

The first principal component (PC1) accounts for 91.08% of the total variance and clearly emerges as the dominant driver of yield curve fluctuations. The loading plot of PC1 reveals positive loadings across all maturities, indicating that variations in this factor correspond to parallel shifts in the yield curve. In other words, changes in PC1 lead to

simultaneous increases or decreases in yields across all tenors, representing the so-called level factor. This component captures broad-based movements in the general level of interest rates and is thus associated with macroeconomic influences such as monetary policy shifts and long-term inflation expectations.

The second principal component (PC2), explaining 8.16% of the total variance, can be interpreted as the slope factor. Its loading structure is characterized by negative values for short maturities and positive values for longer maturities, reflecting an inverse relationship between short- and long-term rates. When this factor increases, the yield curve steepens-short-term yields decline while long-term yields rise. Conversely, a decrease in this component leads to a flattening of the curve. The slope factor therefore captures non-parallel shifts in the term structure and is closely related to expectations regarding future economic growth and central bank policy adjustments.

The third principal component (PC3) explains 0.59% of the total variance and represents the curvature factor. Its loading pattern exhibits a distinct hump shape, with positive loadings at the short and long ends of the maturity spectrum and negative loadings in the intermediate maturities. This configuration indicates that variations in PC3 alter the concavity of the yield curve, producing so-called butterfly movements. Such changes often occur when medium-term interest rates move differently from short- and long-term rates, providing information about market expectations of medium-term monetary policy and term premia.

The empirical findings are consistent with the theoretical and empirical literature on interest rate modeling, particularly the work of (Rebonato, 2018, pp. 98–107). The identification of level, slope, and curvature as the three principal components of the yield curve is a well-established result in fixed-income research. Their hierarchical importance-where the level factor dominates, followed by the slope and curvature factors-reflects a universal characteristic of yield curve dynamics across different markets and time periods.

The concentration of explanatory power within the first three components demonstrates the suitability of PCA for dimensionality reduction in yield curve modeling. Instead of modeling the dynamics of nine highly correlated interest rates, the analysis can be simplified to three orthogonal factors that capture nearly all relevant variation. This dimensionality reduction not only mitigates model complexity and overfitting risk but also enhances interpretability by linking statistical factors to economically meaningful concepts. Furthermore, such a factor-based representation supports practical applications in risk management, hedging, and scenario analysis, as the identified components correspond directly to the main sources of interest rate risk in the term structure.

4.2 Hyperparameter Tuning

To identify the most suitable model configuration, a comprehensive hyperparameter search was conducted using the Hyperband algorithm with a total of 1000 trials. Model performance was evaluated based on the unweighted RMSE computed on the validation dataset (*val_rmse*). For subsequent analysis, only the top 5% of all tested configurations were retained, resulting in a subset of 50 models that minimized the validation RMSE. Within this subset, Pearson correlation coefficients were calculated to quantify the linear dependencies among the numerical hyperparameters, as well as their individual relationships with the validation error. The resulting correlation matrix and correlation vector serve as diagnostic tools for assessing parameter interactions and their effects on model performance.

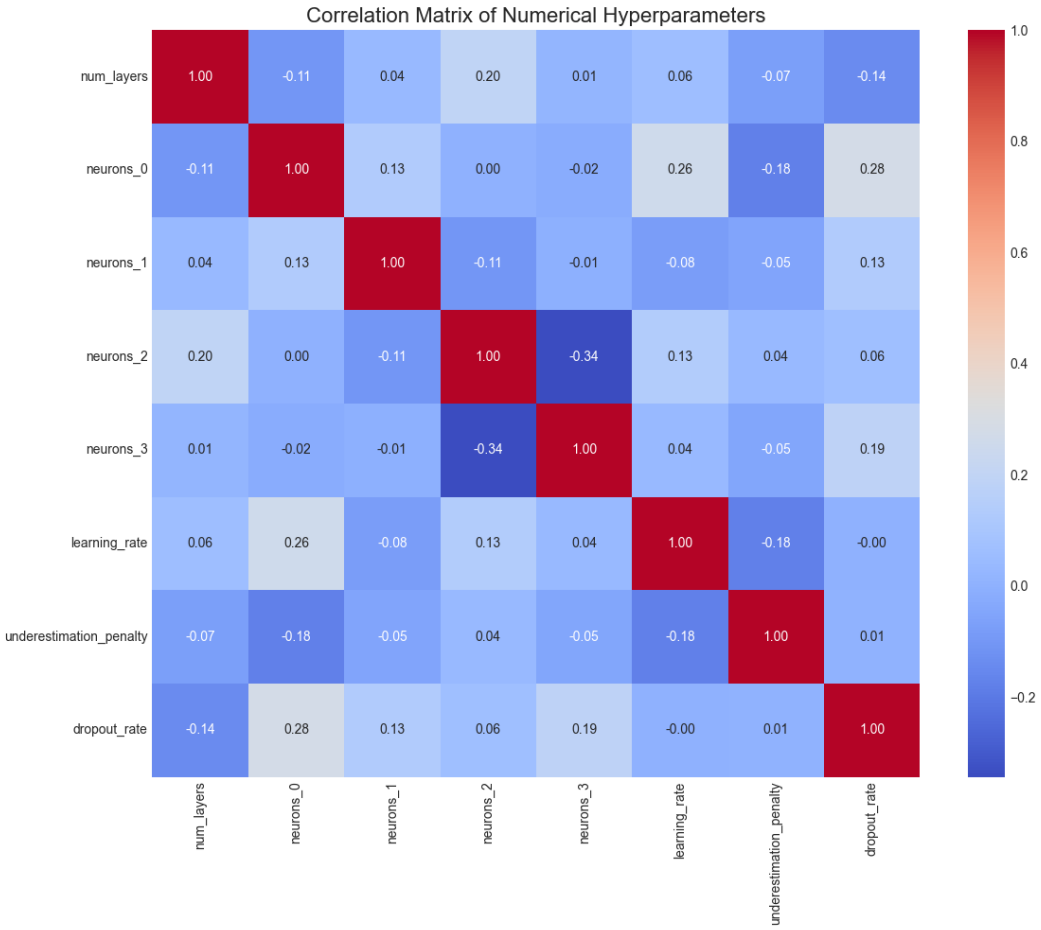

Figure 12: Correlation between the optimized numerical hyperparameters.

The correlation matrix of numerical hyperparameters in figure 12 reveals that most pairwise correlations are weak, as indicated by predominantly light blue and grey cells in the respective heatmap. This finding suggests that the search space of the best-performing models is not strongly restricted by interdependencies among parameters. Nevertheless, a moderate negative correlation of -0.34 between *neurons_2* and *neurons_3* can be observed, indicating that an increase in the capacity of the third hidden layer is often ac-

companied by a reduction in the fourth layer's size. This pattern may reflect an implicit constraint on the overall capacity distribution across deeper layers. Furthermore, weak positive correlations were detected between *neurons_0* and both *dropout_rate* (0.28) and *learning_rate* (0.26), implying that larger initial layers might benefit from slightly higher regularization and more aggressive learning rates to achieve optimal performance.

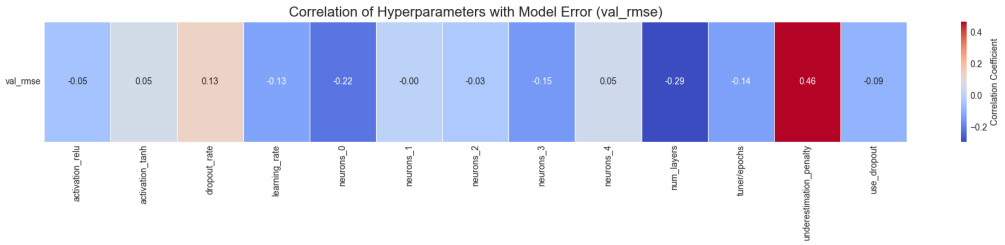

Figure 13: Correlation between the optimized numerical hyperparameters and the validation error.

The correlation analysis provided in figure 13 between hyperparameters and model error provides additional insights into performance sensitivity. Since the objective is to minimize *val_rmse*, a negative correlation indicates an improvement in performance with increasing hyperparameter values, while a positive correlation suggests performance deterioration. The strongest observed relationship is a positive correlation of 0.46 for the *underestimation_penalty*, indicating that higher values of this penalty consistently degrade performance. Consequently, the optimal configuration for this parameter likely lies near zero. In contrast, a moderate negative correlation of -0.29 between *num_layers* and error suggests that deeper network architectures tend to yield superior performance within the top-tier models. Similarly, the parameters *neurons_0* (-0.22) and *neurons_3* (-0.15) exhibit weak negative correlations with error, indicating a slight preference for wider layers at specific network depths. The regularization term *dropout_rate* shows a mild positive correlation (0.13) with error, implying that excessive dropout may hinder learning even among the best configurations. Lastly, the parameter *tuner/epochs* is negatively correlated with error (-0.14), confirming that extended training durations within the Hyperband framework generally improve convergence and validation performance.

The architecture of the best-performing model reflects these findings. It consists of five hidden layers with 112, 48, 32, 112, and 48 neurons, respectively. All layers employ the ReLU activation function. Although a dropout rate of 0.4 was specified in the search space, dropout was disabled in this configuration. The model uses a learning rate of approximately 0.0033 and applies an underestimation penalty of 1.5, indicating the use of a customized loss formulation to control prediction asymmetry. The corresponding hyperparameter configuration is summarized in table 9.

Table 9: Hyperparameter Configuration of the Best-Performing Model

Hyperparameter	Description	Value
num_layers	Number of hidden layers	5
neurons_0	Neurons in layer 1	112
neurons_1	Neurons in layer 2	48
neurons_2	Neurons in layer 3	32
neurons_3	Neurons in layer 4	112
neurons_4	Neurons in layer 5	48
activation	Activation function	ReLU
use_dropout	Dropout enabled	False
dropout_rate	Dropout rate (inactive)	0.4
learning_rate	Optimizer learning rate	0.0033
underestimation_penalty	Penalty for underestimation	1.5

The table reports the optimal hyperparameter values obtained from the Hyperband search. Dropout was disabled in the final configuration, rendering the specified dropout rate inactive. The underestimation penalty indicates the inclusion of a custom asymmetric loss component.

4.3 Out-of-Sample Performance

4.3.1 Pricing Ability

Figure 14 depicts the daily out-of-sample RMSE for the NN and the traditional LM calibration methods over the test period from early August to early September 2025. The evaluation is conducted on a hold-out set of swaptions that were excluded from the LM optimization on each respective day, ensuring an unbiased comparison between the NN's predictive capability and the LM's in-sample fitting performance as described in section 3.9.

To provide a metric that reflects the financial significance of the calibration errors, rather than just their statistical magnitude, the subsequent analysis additionally utilizes vega-weighted values. The formal procedure for this vega-based aggregation, which translates volatility errors into their approximate price impact, is specified in section 3.6.

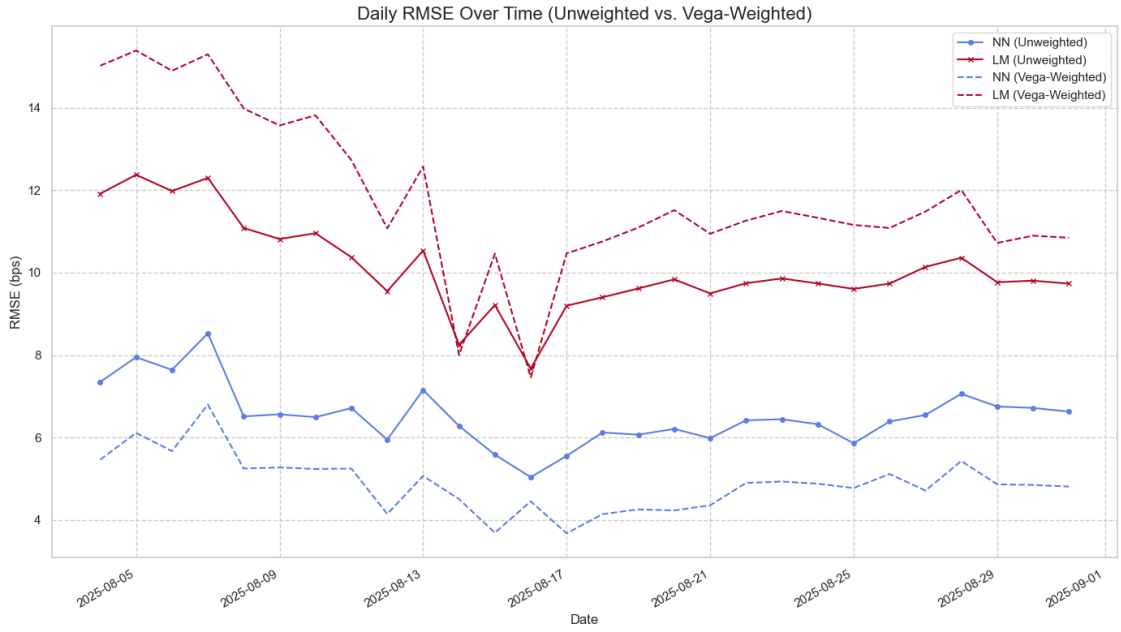

Figure 14: Comparison of daily out-of-sample RMSE between NN and LM calibration methods over the test period.

Across the entire test horizon, the NN consistently achieves superior out-of-sample accuracy relative to the LM algorithm. To provide a more nuanced evaluation beyond a simple statistical average, this analysis considers both the standard, unweighted RMSE and a vega-weighted RMSE. The latter metric assesses performance by giving more weight to errors on swaptions with higher price sensitivity to volatility, thereby reflecting financial significance. On both metrics, the NN remains persistently superior, indicating a more accurate and robust reproduction of market swaption volatilities. Table 10 summarizes the descriptive statistics for all four daily out-of-sample RMSE measures.

Table 10: Out-of-Sample RMSE Statistics (Unweighted vs. Vega-Weighted)

Method	**Mean**	**Std Dev**	**Median**	**Min**	**Max**
NN (Unweighted)	6.53	0.73	6.47	5.04	8.53
NN (Vega-Weighted)	4.89	0.68	4.87	3.67	6.80
LM (Unweighted)	10.11	1.10	9.79	7.67	12.37
LM (Vega-Weighted)	11.83	1.95	11.30	7.46	15.39

Note: The table reports summary statistics of daily out-of-sample RMSE values over the test period for both unweighted and vega-weighted metrics. All values are in basis points.

A key finding revealed by this dual-metric approach is the relationship between the two error measures for the NN. The vega-weighted RMSE (mean of 4.89 bps) is consistently and significantly lower than its unweighted counterpart (mean of 6.53 bps). This indicates that the neural network's smallest errors are concentrated on swaptions with the highest vega - typically the most liquid and financially significant instruments. The model has

implicitly learned to prioritize the accuracy of the swaptions that have the largest impact on portfolio P&L, demonstrating a level of financially-aware generalization.

In stark contrast, the LM algorithm exhibits the opposite and more problematic behavior. Its vega-weighted RMSE (mean of 11.83 bps) is significantly higher than the unweighted metric (mean of 10.11 bps). This reveals a critical weakness of the traditional optimizer: its largest pricing errors are concentrated on the most financially sensitive instruments. While attempting to find a global best-fit, the LM method struggles most with the very swaptions where accuracy is paramount, leading to a financially riskier error profile.

Finally, this analysis underscores the superior stability of the NN. This is reflected in the standard deviation of its unweighted RMSE (0.73 bps) and its even more stable vega-weighted RMSE (0.68 bps). Conversely, the financial instability of the LM method is highlighted by the substantial increase in volatility when moving from its unweighted (1.10 bps) to its vega-weighted (1.95 bps) error, confirming that its largest and most volatile errors occur on the instruments of highest financial importance.

To formally assess the statistical properties of the daily out-of-sample RMSE values for both calibration methods, a series of hypothesis tests were conducted. These include normality testing using the Shapiro-Wilk test, a non-parametric Mann-Whitney U test to examine differences in central tendency, and Levene's test to evaluate equality of variances. The results are summarized in table 11.

Table 11: Statistical Tests on Out-of-Sample RMSE Distributions

Test	**Method**	**Statistic**	**p-value**	**Interpretation**
Shapiro–Wilk	NN	0.9547	0.2588	Errors are normally distributed
Shapiro–Wilk	LM	0.9263	0.0496	Errors deviate from normality
Mann–Whitney U	NN vs. LM	3.0000	< 0.0001	Significant difference in error distributions
Levene's	NN vs. LM	1.9225	0.1713	No significant difference in variances

Note: The tests were conducted on the unweighted errors.

The Shapiro-Wilk test results indicate that the NN errors ($p = 0.2588$) do not significantly deviate from normality at any conventional significance level ($\alpha \in \{0.01, 0.05, 0.1\}$). In contrast, the LM errors ($p = 0.0496$) fail the normality assumption at $\alpha = 0.05$ and $\alpha = 0.1$, suggesting slight non-normal behavior.

Given this, the Mann-Whitney U test was used to evaluate whether the two error distributions differ in central tendency. The test statistic ($U = 3.00$, $p < 0.0001$) strongly rejects the null hypothesis across all significance levels, confirming that the NN's errors

are significantly lower than those of the LM method.

Finally, Levene's test ($p = 0.1713$) indicates no statistically significant difference in the variances of the two RMSE distributions, implying comparable dispersion. Combined, these results confirm that the NN calibration achieves a statistically and economically meaningful improvement in accuracy without introducing greater variability.

The magnitude of the observed performance difference is economically and statistically significant. On average, the NN reduces the out-of-sample pricing error by 3.58 bps compared to the LM calibration. During periods of elevated instability in the LM algorithm's performance, this gap widens up to 4.57 bps.

The mean prediction errors, defined as the difference between model-implied volatilities and observed market volatilities (Model Volatility – Market Volatility) in basis points (bps), were analyzed for the NN and LM calibration methods across various swaption expiry and tenor bins. The corresponding heatmaps in figure 15 visualize these unweighted statistical errors, where red cells indicate overestimation and blue cells indicate underestimation.

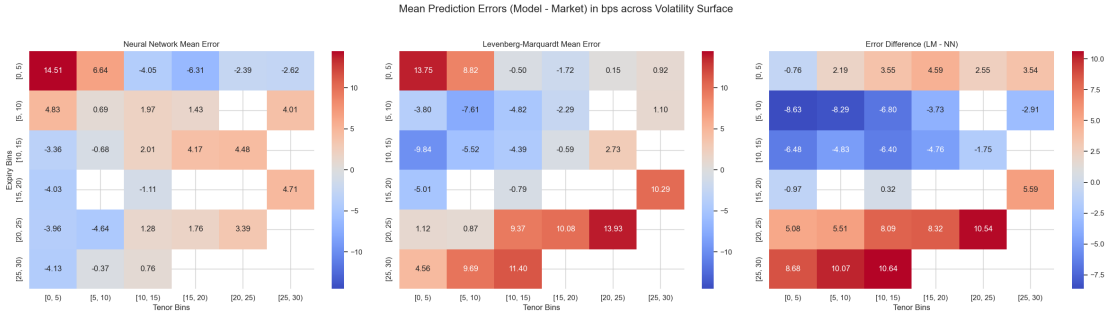

Figure 15: Heatmaps of Mean Prediction Errors for NN and LM Calibration Methods, and Their Difference.

The LM calibration exhibits pronounced, systematic pricing biases across the surface. At the short end, errors reach up to +13.75 bps, while at the long end, they are as high as +13.93 bps. Conversely, the LM model underestimates volatilities in the medium-expiry region, with errors down to -9.84 bps. This pattern demonstrates that the in-sample optimization struggles to identify a single set of parameters capable of simultaneously fitting the complex curvature of the surface. In contrast, the NN model produces a more balanced and less systematic error distribution. Although it slightly overestimates volatilities at the highly volatile short end (+14.51 bps), the errors elsewhere are smaller and more evenly distributed.

To deepen this analysis from a financial risk perspective, a second set of heatmaps was generated, as shown in figure 16. These plots visualize the *vega-weighted* mean errors, which represent the financial impact of the calibration inaccuracies. This metric highlights

the regions of the volatility surface where the model's errors would cause the largest potential P&L discrepancies.

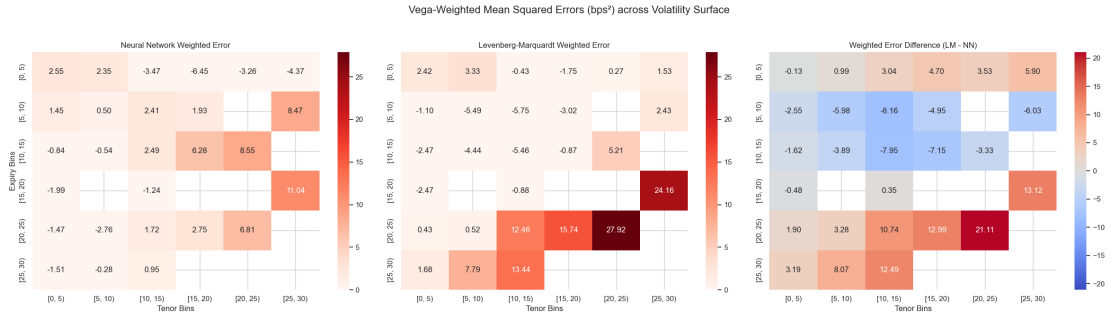

Figure 16: Heatmaps of Vega-Weighted Mean Errors for NN and LM Calibration, and Their Difference.

The vega-weighted analysis reveals a critical divergence in the models' performance. For the NN, the financial impact of its errors is relatively contained and distributed. While the largest unweighted error was at the short end, the largest weighted errors (e.g., 8.55, 11.04 bps) are shifted to the mid-to-long tenor and expiry regions. This indicates that the NN's largest statistical mispricings occur on instruments with lower financial sensitivity.

The LM model, however, displays a starkly problematic pattern. The vega-weighted heatmap shows an extreme concentration of financial error in the long-expiry, long-tenor portion of the surface, with weighted errors reaching as high as 27.92 bps. This demonstrates that the LM optimizer's largest statistical errors are systematically located in the region of the surface corresponding to high-vega, financially critical instruments. While its unweighted errors appeared more dispersed, this analysis reveals that its flaws are concentrated where they are most damaging from a risk-management perspective.

The weighted error difference heatmap (LM minus NN) confirms this conclusion unequivocally. The bright red areas in the long-expiry, long-tenor corner, with differences exceeding 21 bps, highlight the LM model's profound financial underperformance in this critical region. Conversely, the dark blue areas in the medium-expiry range show where the LM's underestimation bias has a greater financial impact than the NN's errors. Overall, the vega-weighted analysis proves that the NN not only achieves superior statistical accuracy but also produces a significantly more benign and robust financial error profile, avoiding the dangerous concentration of risk inherent in the traditional LM calibration.

The relationship between model-implied volatilities and observed market volatilities was examined on an instrument-by-instrument basis for all swaptions in the hold-out sets over the entire test period. Scatter plots (figure 17) were constructed for both the NN and LM (LM) models, with the dashed $y = x$ line representing a perfect fit, where model predictions exactly match market values. The proximity of the data points to this line provides

a direct measure of each model's out-of-sample accuracy and predictive reliability.

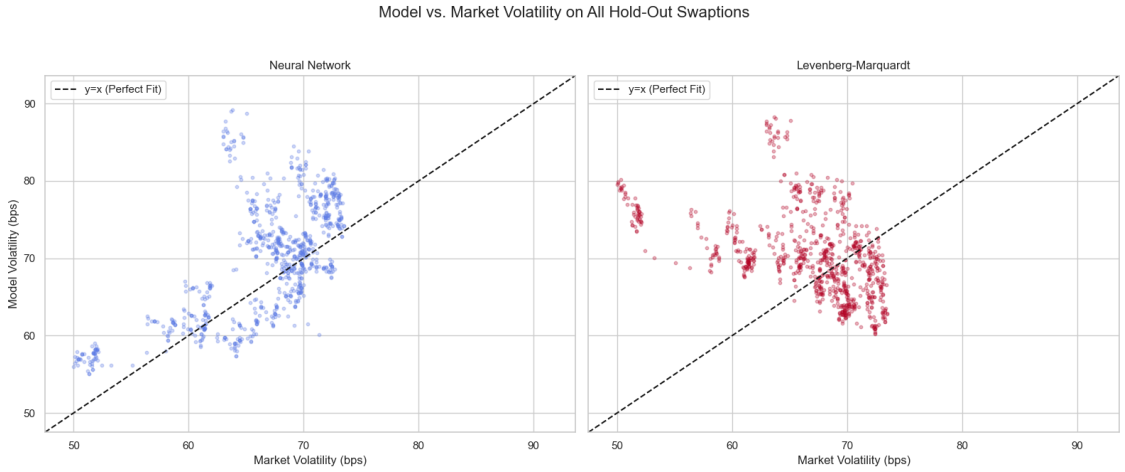

Figure 17: Scatter Plot Comparison of Model-Implied vs. Market Swaption Volatilities for NN and LM Calibration Methods.

The NN model demonstrates a markedly better goodness-of-fit. Its data points are more tightly clustered around the $y = x$ line, exhibiting a strong positive correlation with market volatilities across the full range of observed values. This indicates that the NN consistently produces volatilities that closely align with unseen market data, confirming its ability to generalize beyond the calibration set.

In contrast, the LM model exhibits substantial dispersion and a weaker correlation. Its data points form a diffuse cloud with significant vertical spread, meaning that for swaptions with similar market volatilities, the LM model can generate widely varying predicted volatilities. This highlights the method's lack of robustness and reduced reliability in out-of-sample prediction.

The LM scatter plot visually corroborates the systematic biases observed in the heatmap analysis. A majority of the points lie above the $y = x$ line, indicating a persistent tendency to overestimate volatility. Conversely, the NN errors appear more symmetrically distributed around the perfect fit line, suggesting a more balanced prediction profile with minimal systematic bias.

While the average 3.58 bps reduction in out-of-sample RMSE is statistically significant, a complete analysis requires an examination of its economic relevance. The economic significance of this accuracy improvement is amplified by the scale of modern derivatives portfolios and manifests in three critical domains: portfolio valuation, hedging efficiency, and institutional risk management.

First, the most direct monetary impact of a volatility misestimation is transmitted through a portfolio's vega. For institutional-scale trading desks, where aggregate vega is substan-

tial, a systematic pricing error of several basis points can lead to significant daily mark-to-market discrepancies. Such inaccuracies can obscure the true profitability of trading strategies and lead to flawed financial reporting. The superior accuracy of the neural network provides a more reliable basis for daily valuation and profit and loss attribution.

Second, the utility of a calibration extends beyond pricing to the calculation of risk sensitivities for hedging. An inaccurate calibration yields unreliable hedge ratios, leading to hedge slippage and unintended market exposures. Furthermore, the greater parameter stability demonstrated by the neural network translates into more stable hedge ratios over time. This reduces the need for frequent, costly portfolio rebalancing driven by model instability, thereby lowering transaction costs and improving the overall efficiency of the hedging program.

Finally, the improved accuracy has profound implications for firm-wide risk management and regulatory compliance. The parameters derived from calibration are critical inputs for stochastic risk models used to compute measures such as Value-at-Risk and to conduct stress tests. A systematic calibration error distorts the simulated profit and loss distributions, potentially leading to a material underestimation of the firm's true market risk. The observed 3.58 bps difference can be interpreted as a quantifiable measure of model risk, providing crucial information for model validation functions and for the efficient allocation of regulatory capital. In this context, the enhanced accuracy of the neural network approach is not a marginal improvement, but a meaningful contribution to financial stability and operational robustness.

4.3.2 Parameter Stability

This section examines the temporal behavior of the calibrated HW model parameters-specifically the mean-reversion rate (α) and the piecewise constant volatility term structure (σ_i)-for both the NN and LM (LM) calibration methods. The analysis is based on daily recalibrations (LM approach) and the predicted parameters (NN) over the test period, with a focus on the stability of the parameter trajectories. Parameter stability is of central importance, as erratic fluctuations can result in unstable hedging ratios, inconsistent pricing, and unreliable risk metrics.

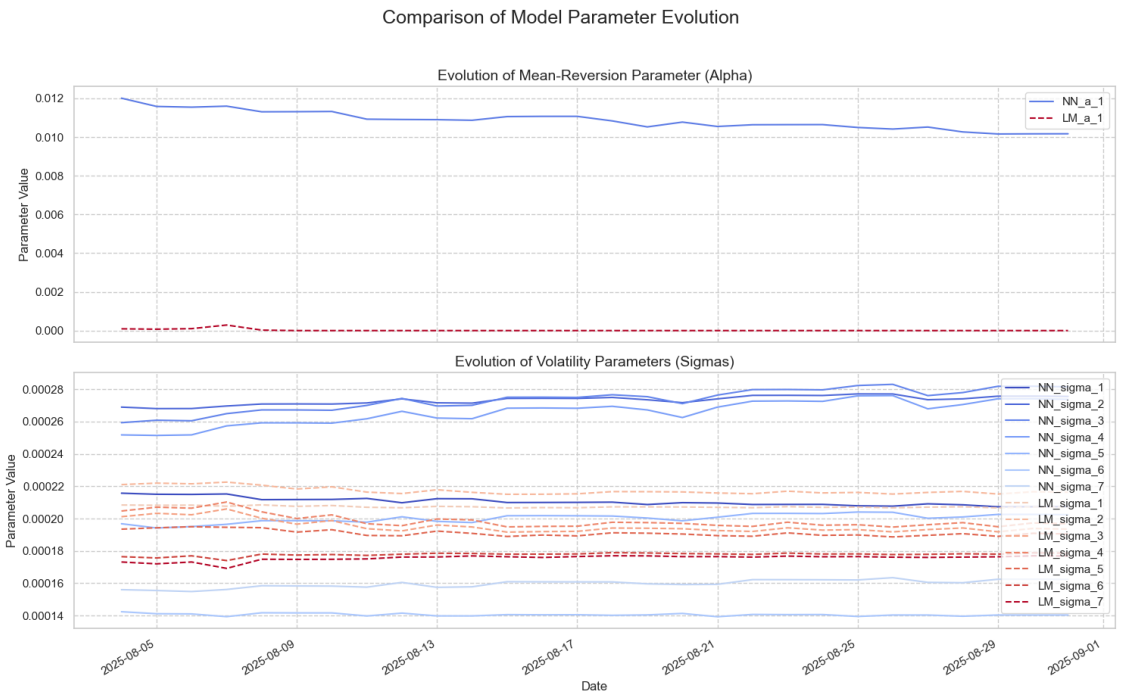

Figure 18: Parameter Evolution for NN and LM Calibration Methods.

At first inspection, both calibration techniques yield parameters that evolve smoothly over time without abrupt jumps or discontinuities. The magnitudes of the parameters appear economically plausible, with no extreme outliers observed. This visual stability suggests that both methods achieve a baseline level of temporal consistency in the estimated HW parameters.

A notable divergence arises in the behavior of the mean-reversion parameter. The LM method persistently drives α toward values near zero (mean ≈ 0.00002), with a correspondingly low standard deviation. Although statistically stable, this near-zero value effectively suppresses the mean-reversion mechanism of the model, thereby failing to capture an essential economic feature of interest rate dynamics. In contrast, the NN produces a stable and economically meaningful mean-reversion level (mean ≈ 0.0109), reflecting a more realistic adjustment speed of short rates toward their long-term equilibrium.

The volatility term structures (σ_i) estimated by the NN demonstrate a high degree of structural coherence. Across time, the piecewise volatilities retain their relative ordering, and the overall shape of the term structure evolves smoothly and predictably. This indicates that the NN calibration preserves the internal consistency of the model across maturities and dates. The resulting stability of the σ_i curves suggests that the NN effectively captures persistent features of the volatility surface while adapting flexibly to gradual market changes.

Although the LM method exhibits slightly lower parameter standard deviations on an ab-

solute scale, this apparent stability does not necessarily imply superior model robustness. The NN's moderately higher parameter variance reflects meaningful responsiveness to evolving market conditions within a coherent and economically interpretable structure. Thus, the NN's stability should be viewed as both statistical and structural-balancing smooth temporal evolution with sensitivity to genuine shifts in the underlying market environment.

4.3.3 Parameter Sensitivity Analysis

The sensitivity analysis highlights pronounced differences in the robustness and economic interpretability of the parameters estimated by the NN and the traditional LM (LM) method. Across all examined scenarios, the NN exhibits stable and economically consistent adjustments, whereas the LM method frequently produces erratic or extreme results, particularly regarding the estimation of the mean-reversion parameter. The following discussion presents a detailed examination of three distinct yield curve perturbations.

Scenario 1: Parallel Yield Curve Shift Up (+50 bps) This scenario reflects a market environment characterized by rising interest rates, such as during monetary tightening or increased inflation expectations. The NN responds by reducing the mean-reversion parameter a_1 by 22.38%. This behavior is economically intuitive: as interest rates rise, the short rate becomes less strongly attracted to its long-term mean, implying a slower reversion speed. Regarding volatility, the NN exhibits a differentiated adjustment across maturities. Short-to-medium term volatilities (σ_1 to σ_4) increase by 1.46% to 8.66%, reflecting greater short-term uncertainty, whereas long-term volatilities (σ_5 to σ_7) decrease by 3.45% to 5.47%. This pattern suggests that the network associates persistent rate hikes with a more predictable long-term environment, consistent with the stabilizing influence of central bank policy.

In contrast, the LM method produces unstable and economically implausible results. The mean-reversion parameter a_1 collapses by 99.997%, effectively eliminating mean reversion from the model. This outcome does not represent an economic insight but rather a numerical artifact, reflecting the optimizer's convergence to a degenerate local minimum. The volatility response is similarly simplistic: all parameters decrease uniformly by 2.41% to 5.60%, indicating that the LM optimizer finds a slightly smoother volatility surface that minimizes the fitting error without capturing any meaningful structural information.

Scenario 2: Parallel Yield Curve Shift Down (-50 bps) A parallel downward shift of 50 bps captures an environment of monetary easing or a flight-to-quality, where rates decline broadly across maturities. In this context, the NN increases the mean-reversion

parameter a_1 by 16.40%, which is the logical counterpart to the previous scenario. Lower rates induce a stronger pull back toward the long-term mean, consistent with market expectations of normalization. The volatility adjustments are again complex: the short-term volatilities (σ_1, σ_2) decrease, while medium- and long-term volatilities (σ_3 to σ_7) rise by up to 9.87%. This asymmetric reaction captures a common market feature, where declining rates heighten duration risk and uncertainty in the longer end of the curve.

The LM method fails to produce a meaningful calibration under this scenario. The mean-reversion parameter explodes by 303,251.39%, indicating a complete loss of numerical stability. This extreme value is economically meaningless and arises from the optimizer's sensitivity to local minima in the parameter space. The volatility adjustments are erratic, lacking any coherent term-structure pattern. While most volatilities increase, the longest-tenor parameters (σ_6, σ_7) decrease sharply by 10.45% and 17.24%, respectively. This inconsistency further demonstrates the inability of the LM approach to produce economically interpretable parameter dynamics under yield curve perturbations.

Scenario 3: Yield Curve Twist (Steepening) The final scenario models a steepening of the yield curve, representing a situation in which long-term rates rise relative to short-term rates due to improved growth or inflation expectations. The NN decreases the mean-reversion parameter a_1 moderately by 11.83%, reflecting a realistic interpretation that a steeper curve reduces the immediate mean-reverting pressure on the short rate. The network also produces a measured increase in volatility across the term structure, with changes ranging from 0.65% to 3.67%. This restrained yet systematic response indicates that the NN captures the nuanced relationship between curve shape and rate uncertainty without overfitting to transient fluctuations.

The LM method again exhibits numerical instability. The mean-reversion parameter a_1 collapses entirely, decreasing by 100.00%, which effectively removes mean reversion from the model. This recurring pattern underscores the optimizer's inability to handle non-parallel yield curve shifts in a stable manner. The volatility parameters uniformly decrease by 2.25% to 5.08%, mirroring the behavior observed in the upward shift scenario. Such uniform adjustments suggest that the LM method applies a rigid, non-adaptive response that fails to capture the dynamic interactions between curve shape and volatility, in stark contrast to the NN's flexible and economically coherent interpretation.

Overall, these scenario-based analyses demonstrate that the NN not only yields stable parameter estimates but also captures economically meaningful dynamics consistent with macro-financial reasoning. The traditional LM optimizer, by contrast, repeatedly fails to produce reliable or interpretable results, highlighting the superiority of data-driven calibration in ensuring robustness and stability under varying market conditions.

Table 12: Scenario Analysis of NN Model Parameters

	a_1		σ_1		σ_2		σ_3	
Scenario	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value
Base Case		0.01136		0.00021		0.00027		0.00026
Shift Up	-22.38	0.00881	1.46	0.00022	2.83	0.00028	4.43	0.00028
Shift Down	16.40	0.01322	-7.11	0.00020	-2.92	0.00026	6.32	0.00028
Twist	-11.83	0.01001	0.65	0.00021	1.49	0.00027	1.20	0.00027

	σ_4		σ_5		σ_6		σ_7	
Scenario	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value
Base Case		0.00026		0.00018		0.00014		0.00016
Shift Up	8.66	0.00028	-3.45	0.00018	-5.47	0.00013	-3.40	0.00015
Shift Down	2.25	0.00027	9.87	0.00020	4.22	0.00014	2.25	0.00016
Twist	3.67	0.00027	-0.48	0.00018	-2.17	0.00013	-0.02	0.00016

Table 13: Scenario Analysis of LM Model Parameters

	a_1		σ_1		σ_2		σ_3	
Scenario	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value
Base Case		0.00009		0.00021		0.00021		0.00021
Shift Up	-100.00	0.00000	-2.41	0.00020	-3.20	0.00020	-3.80	0.00020
Shift Down	303,251	0.26543	-8.52	0.00019	-1.85	0.00020	1.32	0.00021
Twist	-100.00	0.00000	-2.25	0.00020	-2.96	0.00020	-3.50	0.00020

	σ_4		σ_5		σ_6		σ_7	
Scenario	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value	% δ	Abs. Value
Base Case		0.00021		0.00022		0.00021		0.00020
Shift Up	-3.60	0.00020	-5.60	0.00020	-2.27	0.00020	-0.51	0.00020
Shift Down	3.08	0.00022	-0.67	0.00021	-10.45	0.00018	-17.24	0.00017
Twist	-3.30	0.00020	-5.08	0.00020	-2.00	0.00020	-0.36	0.00020

4.3.4 Analysis of Computational Efficiency and Practical Applicability

The computational performance of the NN and LM (LM) calibration methods was evaluated to assess their practical suitability for real-world applications. Table 14 summarizes the measured runtimes and variability for each approach, highlighting the substantial differences in computational efficiency.

Table 14: Computational Efficiency of NN and LM Calibration Methods

Method	Mean Time (s)	Std Dev Time (s)
NN	0.0041	0.0019
LM	91.87	23.52

Note: The table reports average runtime and standard deviation for daily calibrations. The NN achieves an approximate speed-up of 22,555x compared to LM.

The NN exhibits a dramatic advantage in computational speed for daily calibration tasks. With an average runtime of just 4 milliseconds per calibration, it is approximately 22,555 times faster than the traditional LM method, which requires an average of 92 seconds. This multi-order-of-magnitude improvement fundamentally transforms the operational feasibility of high-frequency calibration workflows and is consistent with the findings of Alaya et al. (2021), who demonstrated that their neural network reduced the calibration time of the G2++ model from 480.1 seconds to 0.063 seconds-an acceleration of roughly 7,620 times.

This computational advantage arises from the methodological distinction between the two approaches. The NN incurs the majority of its computational cost during an offline training phase, including a one-time hyperparameter tuning (approx. 1 week) and a subsequent model training (approx. 1 hour). Once trained, the NN performs calibration via a near-instantaneous forward pass (inference). In contrast, the LM algorithm performs a fully iterative numerical optimization online for each new market snapshot, making it substantially slower and more resource-intensive in day-to-day operations.

The initial offline cost of the NN is efficiently amortized over its operational lifetime. Optimal hyperparameters remain stable over time, requiring only periodic retraining (e.g., nightly or weekly) in approximately one hour. This allows thousands of real-time calibrations to be executed almost instantaneously, rendering the upfront investment highly cost-effective in a production environment.

The sub-second runtime of the NN makes it suitable for applications that are infeasible with the LM method, including real-time risk management, pre-trade pricing of extensive derivative portfolios, and intraday recalibration in response to evolving market conditions. The 92-second average runtime of LM constitutes a significant bottleneck, restricting its use to end-of-day or batch-processing tasks.

Beyond mean speed, the NN demonstrates highly predictable performance with a low standard deviation of execution time (0.0019 s), whereas the LM algorithm shows substantial variability (23.52 s) due to sensitivity to market-specific data and optimization

paths. This predictability is critical for integration into time-sensitive production workflows, further emphasizing the NN's suitability for operational deployment.

4.4 Interpretability of the Neural Network via SHAP Values

The interpretability of the NN was examined through SHAP values, which quantify the contribution of each input feature to the model's predictions for the HW parameters. This analysis provides insight into the internal decision-making process of the NN, allowing for a scientific understanding of how it generates parameter estimates from market data.

The SHAP analysis validates the NN as an interpretable model. The feature importance patterns reveal systematic and economically intuitive relationships, demonstrating that the model's predictions are grounded in financially relevant signals rather than being arbitrary or opaque. This dispels the perception of the network as an unexplainable "black box" and confirms that its outputs reflect meaningful market dynamics.

A key finding is the dominant influence of the long-end yield curve shape, represented by the curvature_10y20y30y feature, across nearly all output parameters, including the mean-reversion parameter (a_1) and the piecewise volatility terms (σ_i). This indicates that the NN has learned that the long-end curvature of the yield curve is critical for determining both the long-term pull of interest rates and the overall volatility term structure. This is consistent with financial intuition, as long-end curvature captures market expectations regarding long-term economic uncertainty and monetary policy.

The model further demonstrates sophistication through parameter-specific feature importance. The mean-reversion parameter (a_1) is primarily driven by short- and medium-term yield curve slopes (slope_3m10y, slope_5y30y), reflecting the theoretical notion that mean-reversion measures the speed at which interest rates revert to their long-term average. In contrast, the long-term volatility parameter (σ_7) is highly sensitive to the MOVE_VIX_Ratio, indicating that the network has learned that treasury market-implied volatility is a key determinant of long-horizon rate uncertainty.

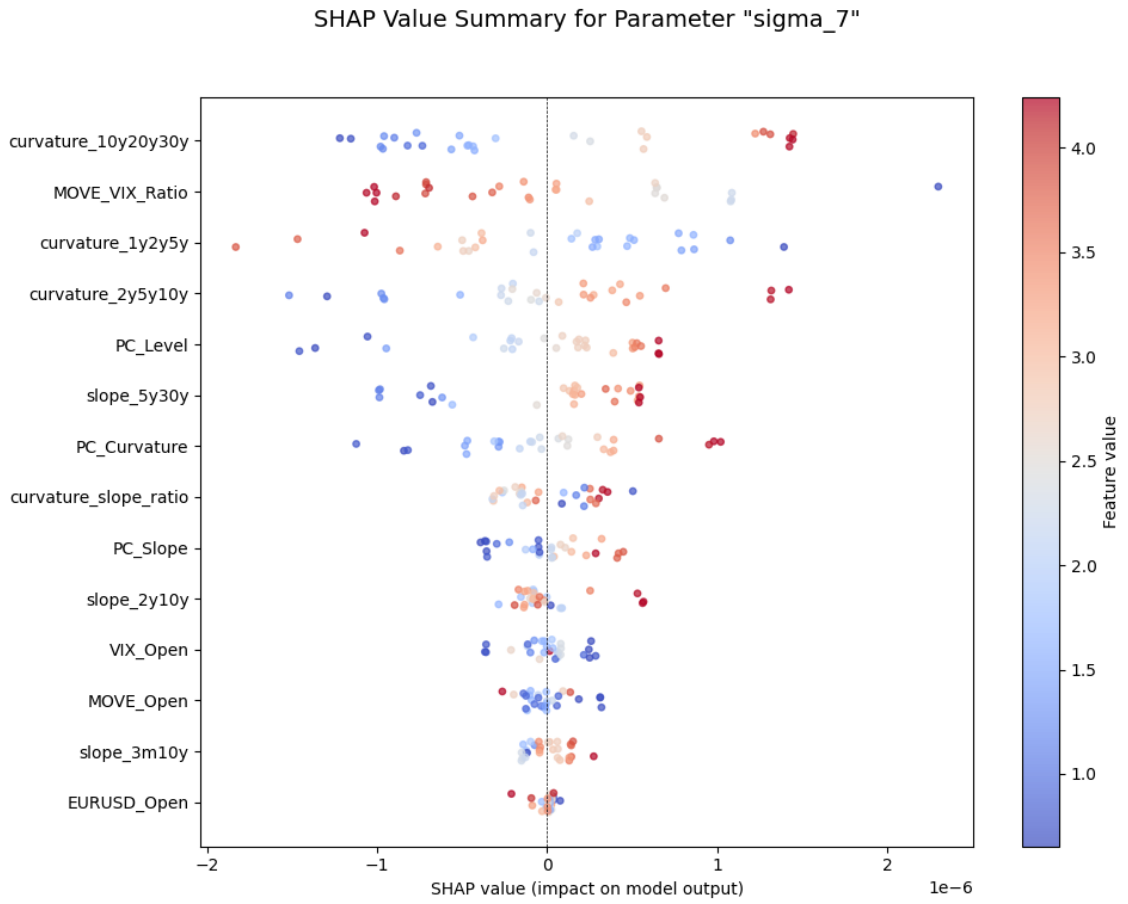

Figure 19: SHAP Summary Plot for the Long-Term Volatility Parameter σ_7.

Direct interpretation of the SHAP summary plot for σ_7 (48) reveals a counter-intuitive financial relationship learned by the model. For the MOVE_VIX_Ratio feature, the NN has identified an inverse relationship: high values of the ratio (red dots), indicating elevated Treasury volatility relative to equity volatility, lead the model to predict lower long-term volatility (σ_7), whereas low values of the ratio (blue dots) push predictions higher. This pattern may reflect a "flight to quality" effect, where extreme market stress, signaled by a high MOVE_VIX_Ratio, is associated with expectations of aggressive central bank interventions aimed at stabilizing long-term rates and suppressing volatility. The network appears to have captured that such stress episodes are often precursors to a calmer, lower-volatility regime over the long term.

This SHAP-based interpretability analysis reinforces confidence in the NN's predictions. It demonstrates that the model captures complex, non-linear relationships consistent with economic theory and market intuition, rather than fitting noise. Consequently, the NN emerges not only as a computationally efficient and accurate calibration tool but also as a robust and transparent alternative to traditional HW calibration methods.

Note: All other plots regarding the SHAP values can be found at the appendix section.

4.5 Comparison with Hernandez (2016)

Hernandez (2016) investigated the calibration of a simplified one-factor HW model characterized by a single mean-reversion parameter α and a single volatility parameter σ. His study was based on 156 GBP ATM swaptions observed between 2013 and 2016. For the estimation task, he employed a feed-forward NN with four hidden layers, trained to directly predict the parameters (α, σ) of the HW model. In contrast, the NN developed in the present thesis consists of five layers and utilizes a residual parameterization approach, in which the model learns to predict corrections to an initial guess of the parameters rather than the parameters themselves. This design choice aims to enhance numerical stability and convergence properties, particularly in non-linear calibration problems.

The methodological distinction between both approaches is substantial. Hernandez (2016) NN was trained using pre-calibrated parameter pairs obtained from a LM optimization as target variables. The LM optimizer represented the traditional calibration benchmark in his study, providing a set of optimal parameters for each observation date. Conversely, the approach presented in this thesis does not rely on such precomputed targets. Instead, the predicted parameters are directly inserted into the QuantLib pricing engine, and the NN is trained by minimizing the deviation between the model-implied and market-observed swaption volatilities. Formally, this corresponds to minimizing the loss function

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\sigma_i^{\text{model}}(\alpha(\theta), \sigma(\theta)) - \sigma_i^{\text{market}} \right)^2,$$

where σ_i^{model} and σ_i^{market} denote model-implied and market-observed volatilities, respectively, and θ represents the NN parameters (see sections 2.4.5 and 3.5). This direct loss formulation allows the network to learn parameter mappings that minimize actual pricing errors rather than reproducing the outcomes of an external optimizer. However, the approach entails higher computational costs because QuantLib's pricing routines are implemented in C++ and do not expose analytical gradients to TensorFlow's automatic differentiation, making backpropagation computationally demanding (see section 3.10.3).

Regarding input representation, Hernandez (2016) incorporated both yield curve information and swaption data into the NN's feature vector. Specifically, he employed a 6-month tenor LIBOR curve discretized at 44 maturity points, ranging from 0 days to 50 years, including tenors of 0, 1, 2, 7, and 14 days; 1–24 months; 3–10 years; and 12, 15, 20, 25, 30, and 50 years. PCA was subsequently applied to retain 99.5% of the variance. Additionally, the 156 swaption volatilities formed a major component of the input vector. In contrast, the this thesis did not use swaption volatilities as input features but instead relied on alternative market information to ensure that the model's predictive power stemmed from broader market dynamics rather than direct encoding of the target variable.

Both studies calibrated their respective models to the full swaption volatility surface. To ensure comparability, the results obtained in this thesis, originally expressed in Bachelier (normal) volatilities, were converted to Black (lognormal) volatilities and then weighted by the corresponding vegas using the approach described in section 3.6, consistent with the convention used by Hernandez (2016).

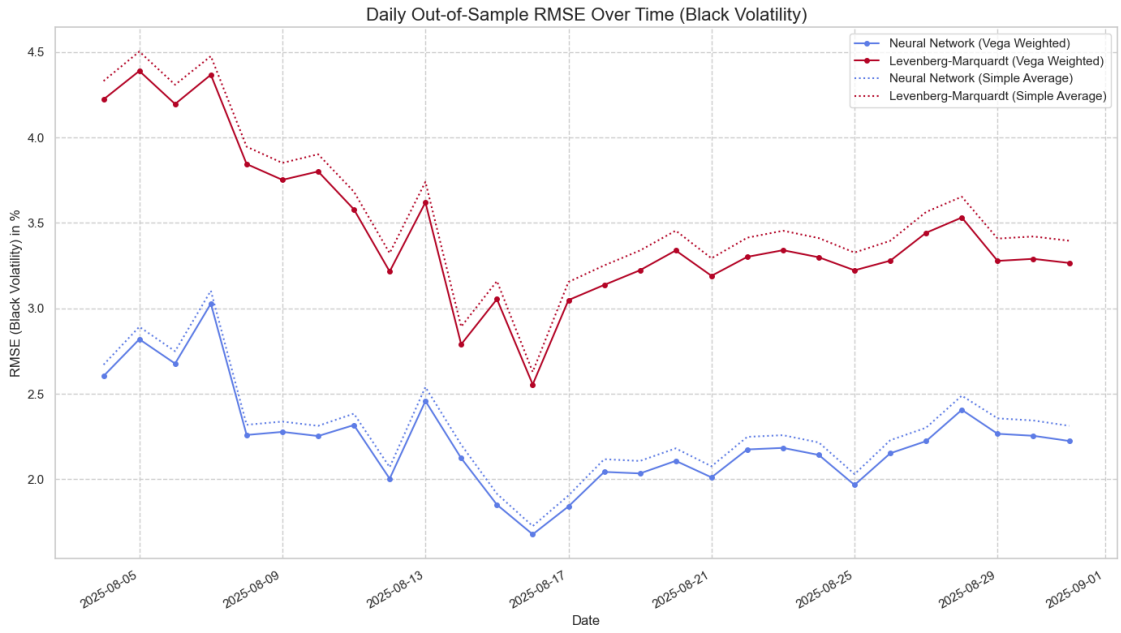

Figure 20: Average Daily RMSE in Black Volatilities for NN and LM Calibration Methods.

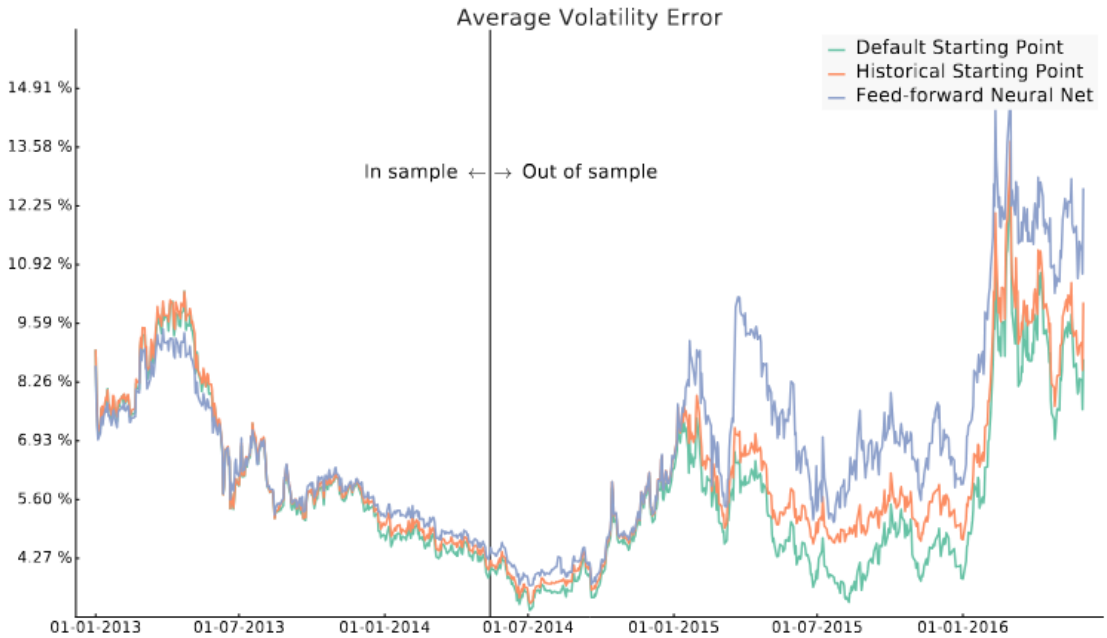

Figure 21: Hernandez's: Average Daily RMSE in Black Volatilities for NN and LM Calibration Methods. Source: (Hernandez, 2016, figure 3)

The traditional LM calibration methods in both studies exhibit comparable average rela-

tive errors of approximately 4.5%. Nevertheless, the results obtained in this thesis indicate slightly lower errors, particularly in periods when Hernandez (2016) LM-based calibration demonstrated performance deterioration, such as from January 2016 onward. This improvement can be attributed to the higher flexibility of the HW specification employed here, which allows for piecewise constant volatility parameters $\sigma(t)$ rather than a single constant parameter, thereby enabling a more accurate fit to the observed volatility surface.

The performance difference becomes more pronounced when comparing the NN-based calibration results. Hernandez (2016) NN achieved an average calibration error similar to that of his LM optimizer, approximately 4.5%, whereas the model developed in this thesis attained a significantly lower average error of around 2.5%. Both studies were conducted under relatively stable market conditions, suggesting that the superior performance of the present model is primarily due to the direct optimization of pricing errors rather than the replication of pre-calibrated parameters. By learning the end-to-end mapping from market inputs to pricing-consistent model parameters, the proposed NN achieves a better generalization to the true market dynamics underlying the volatility surface.

Beyond this headline improvement, a more granular analysis of the error metric reveals a key aspect of the model's practical performance. It was observed that the vega-weighted portfolio error for the neural network is consistently lower than its simple average error. This indicates that the model's highest accuracy is concentrated on the instruments with the highest vega, which typically correspond to the most liquid, financially significant tenors of the volatility surface. The NN has implicitly learned to prioritize the fit where it matters most for risk and P&L, relegating larger (though still relatively small) errors to the less liquid, lower-vega wings of the surface.

A further difference between the studies lies in the temporal stability of the models. Hernandez (2016) reported a degradation in performance approximately six to twelve months after training, implying limited temporal generalization. Due to the restricted testing period of only one month in this thesis, no such degradation could be observed, and further analysis over a longer horizon would be required to assess long-term stability.

It is important to note several limitations that constrain the comparability of both studies. First, Hernandez (2016) employed a simplified version of the HW model with only one mean-reversion and one volatility parameter, inherently reducing its flexibility to fit complex swaption surfaces. Second, his NN benefited from a substantially larger training dataset, as he synthetically generated approximately 150,000 samples, whereas the present thesis relied exclusively on empirical data, limiting both the temporal coverage and sample size. Furthermore, a crucial distinction lies in the evaluation protocol; this thesis enforced a strict out-of-sample test for both the NN and the LM optimizer using an intra-day hold-out set, ensuring a direct comparison of their generalization capabilities. In contrast, the benchmark in Hernandez (2016) work was the in-sample fit of the traditional

optimizer, which does not measure predictive performance on unseen data. Consequently, while the lower calibration errors observed in this work indicate improved model fit, they must be interpreted cautiously given the differences in model complexity, data volume, and testing horizon.

5 Limitations

This study is subject to several limitations arising from the model choice, dataset, methodology, evaluation framework, and practical implementation. The research exclusively focuses on the one-factor HW model. While this model serves as a standard benchmark, it has inherent limitations in capturing complex yield curve dynamics, such as twists and butterfly movements, and in fitting the complete swaption volatility surface. Consequently, the findings may not directly generalize to more sophisticated multi-factor interest rate models. Additionally, the model is calibrated solely to European ATM, leaving its ability to price out-of-the-money (OTM) or in-the-money (ITM) options and to reproduce the volatility smile or skew unevaluated.

The dataset employed introduces several constraints. The analysis covers only a short, recent period from June 1, 2025, to August 31, 2025, characterized by generally decreasing volatility and moderately rising rates. The performance across other market regimes, including periods of high volatility, financial crises, or near-zero interest rates, remains untested. Data were manually extracted from screenshots due to license restrictions, introducing potential transcription errors that would not exist with a direct Application Programming Interface (API) feed. Moreover, the use of mid-market quotes for swap rates and swaption volatilities ignores bid-ask spreads, which reflect transaction costs and liquidity, potentially affecting practical profitability and risk assessment. The NN is trained on daily snapshots, which limits its ability to capture intraday movements, restricting applicability for real-time pricing or high-frequency trading. The research is confined to the Euro market, so the conclusions may not extend to other currency markets with different structures, liquidity profiles, and central bank policies.

Methodologically, the evaluation framework itself contains significant constraints. **Perhaps the most critical limitation from a practical risk management perspective is the study's exclusive focus on pricing accuracy, while the stability and reliability of the resulting hedge parameters remain unevaluated.** A model can exhibit a low pricing error (RMSE) yet produce highly volatile hedge ratios if its calibrated parameters—and thus its derivatives with respect to market variables—fluctuate erratically over time. Such instability would render the model impractical for active hedging, as it would necessitate frequent and costly portfolio rebalancing that could erode any pricing advantages. In a production environment, predictable and stable hedge parameters are often more critical than marginal improvements in pricing accuracy. Therefore, while this thesis demonstrates the NN's superiority in replicating market prices, it does not confirm its utility as a robust tool for risk management, which remains an unaddressed question. Furthermore, the focus on RMSE as a single error metric does not capture tail risks or the distribution of errors, and the artificial hold-out set used for the LM method may slightly handicap its in-sample performance relative to real-world usage.

From a technical and practical perspective, the end-to-end training approach is computationally intensive due to numerical gradient approximation, which may have limited hyperparameter tuning and model complexity. Static hyperparameters were used, although optimal settings may vary across market regimes, and no feature selection algorithm was employed. PCA, as a linear technique, may not capture non-linear relationships fully. Moreover, the NN approach requires substantial upfront investment in data collection, feature engineering, and initial training. The long-term performance and need for retraining could not be fully assessed, and crucial aspects of model risk, governance, and regulatory considerations were not addressed. These limitations collectively suggest that while the proposed approach demonstrates promising results, its applicability and robustness in broader, real-world settings require further investigation.

6 Conclusion

This chapter synthesizes the key findings of the study, comparing the accuracy, speed, and robustness of the machine learning and traditional calibration methods. It discusses the implications for practitioners and researchers, assesses the study's limitations, and outlines directions for future research in machine learning-based financial model calibration.

The primary objective of this research was to conduct a comprehensive comparison between a traditional optimization-based calibration method, specifically the LM algorithm, and a novel, end-to-end machine learning approach for the one-factor HW interest rate model. The study demonstrates that the NN-based calibration is orders of magnitude faster than the traditional optimizer, performing in milliseconds compared to over ninety seconds for LM, representing a computational speed-up of approximately 22,555 times. This shift moves the main computational burden from ongoing, real-time application to a one-time, offline training phase.

In terms of accuracy, the NN consistently outperforms the traditional method in out-of-sample pricing, achieving a substantially lower average root mean squared error of 6.53 basis points compared to 10.11 basis points for the LM approach. The network's error profile is more balanced across the volatility surface, while the traditional method exhibits systematic biases, particularly for short-tenor and short-maturity swaptions. Moreover, the NN produces economically sensible and stable parameters that react logically to simulated market shocks, whereas the traditional optimizer frequently suffers from numerical instability, with the mean-reversion parameter collapsing or exploding under stress scenarios. Compared to the approach of Hernandez (2016), in which the network is trained on a predetermined set of optimal parameters, the end-to-end framework integrating the network directly with QuantLib's pricing engine and minimizing pricing errors proves, once trained, superior in both speed and pricing ability.

The implications of these findings are substantial. The machine learning approach is particularly well-suited for applications requiring high-frequency recalibration, including real-time risk management, pre-trade pricing for large derivative portfolios, and intraday adjustments. Its robust performance in sensitivity analyses suggests resilience to market shocks and changing regimes, whereas the LM method is more appropriate for less time-sensitive, batch-processing tasks. Practically, the speed, accuracy, and robustness of the NN enable more frequent and comprehensive risk assessments, more stable hedge ratios, and reliable pricing of complex, path-dependent derivatives.

Nonetheless, this research is subject to several limitations. The analysis was restricted to a one-factor HW model calibrated exclusively to European ATM swaptions, omitting multi-factor models, other derivative instruments, and the volatility smile or skew. The dataset covers only a short, three-month period in a single currency market, limiting general-

izability across different economic cycles and the assessment of long-term performance stability. Feature selection was not systematically performed, potentially constraining model performance. The evaluation focused solely on pricing accuracy (RMSE), without assessing the stability or reliability of hedge parameters, which are critical for practical risk management. Furthermore, the NN approach entails a substantial upfront investment in training and hyperparameter tuning, and despite SHAP-based interpretability, its "black box" nature presents challenges for model validation, governance, and regulatory approval.

Future research should extend the framework to multi-factor interest rate models like the G2++ model to capture more complex yield curve dynamics and incorporate OTM and ITM swaptions to account for the volatility smile. Evaluating hedge performance through the accuracy, stability, and P&L impact of derived hedge parameters would provide a more comprehensive assessment. Validation over longer time horizons and across diverse market conditions would provide deeper insights into the NN's behavior under varying market regimes. Systematic feature selection using techniques such as backward elimitation, could further improve model robustness. Exploring advanced NN architectures, such as recurrent networks or transformers, may enhance the capture of temporal dependencies in market data. A significant enhancement to the residual learning framework would be the adoption of a dynamic initial guess, leveraging the parameters from the previous day, to focus the learning task on modeling daily parameter dynamics rather than absolute levels, potentially improving both stability and convergence speed. Finally, re-implementing the HW swaption pricing logic in a fully GPU-accelerated and differentiable framework, eliminating numerical gradient approximations, could dramatically reduce training times, enable more extensive hyperparameter exploration, and facilitate near-real-time high-frequency recalibration on large derivative portfolios.

References

Alaya, M. B., Kebaier, A., & Sarr, D. (2021). Deep calibration of interest rates model. *arXiv preprint arXiv:2110.15133*.

Aljalbout, E., Golkov, V., Siddiqui, Y., & Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*. https://doi.org/10.48550/arxiv.1801.07648

Alvarez, H., Sacchi, R., & Señaris, T. (2022). *Hull-white calibration for swaptions using nns* (tech. rep.). Barcelona School of Economics. https://repositori-api.upf.edu/api/core/bitstreams/8bae6b11-7b9e-44a8-8b8e-6ac7ec51c920/content

Baaquie, B. E. (2010). Interest rates in quantum finance: Caps, swaptions and bond options. *Physica A: Statistical Mechanics and its Applications*, *389*(2), 296–314. https://doi.org/https://doi.org/10.1016/j.physa.2009.09.031

Black, F. (1976). The pricing of commodity contracts. *Journal of Financial Economics*, *3*(1-2), 167–179. https://doi.org/10.1016/0304-405X(76)90024-6

Brigo, D., & Mercurio, F. (2006). *Interest rate models: Theory and practice* (2nd) [1014 pages]. Springer.

Chan, J. Y.-L., Leow, S. M. H., Bea, K. T., Cheng, W. K., Phoong, S. W., Hong, Z.-W., & Chen, Y.-L. (2022). Mitigating the multicollinearity problem and its machine learning approach: A review. *Mathematics*, *10*(8). https://doi.org/10.3390/math10081283

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Gurrieri, S., Nakabayashi, M., & Wong, T. (2009). Calibration methods of hull-white model [Available at SSRN: https://ssrn.com/abstract=1514192]. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1514192

Hagan, P., Kumar, D., Lesniewski, A., & Woodward, D. (2002). Managing smile risk. *Wilmott Magazine*, *1*, 84–108.

Hernandez, A. (2016). Model calibration with neural networks. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2812140

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hohmann, D., Hörig, M., Ketterer, F., Murray, K., & Ward, R. (2015, September). The new normal: Using the right volatility quote in times of low interest rates for solvency ii risk factor modelling [Accessed: 2025-10-19]. https://web.actuaries.ie/sites/default/files/erm-resources/2079LDP_20150911.pdf

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2020). Normalization techniques in training dnns: Methodology, analysis and application. https://arxiv.org/abs/2009.12836

Hull, J., & White, A. (1990). Pricing interest-rate derivative securities. *The Review of Financial Studies*, *3*(4), 573–592. https://doi.org/10.1093/rfs/3.4.573

Hull, J. C. (2015). *Options, futures and other derivatives* (9nth). Pearson Education.

Izzo, D., Ruciński, M., & Biscani, F. (2012). The generalized island model. In F. Fernández de Vega, J. I. Hidalgo Pérez, & J. Lanchares (Eds.), *Parallel architectures and bioinspired algorithms* (pp. 151–169). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28789-3_7

Jamshidian, F. (1989). An exact bond option formula. *The Journal of Finance*, *44*(1), 205–209. https://doi.org/10.1111/j.1540-6261.1989.tb02414.x

Karlsson, P., Jain, S., & Oosterlee, C. W. (2016). Fast and accurate exercise policies for bermudan swaptions in the libor market model. *International Journal of Financial Engineering*, *03*(01), 1650005. https://doi.org/10.1142/S2424786316500055

Kladivko, K., & Rusy, T. (2023). Maximum likelihood estimation of the hull–white model. *Journal of Empirical Finance*, *70*, 227–247. https://doi.org/https://doi.org/10.1016/j.jempfin.2022.12.002

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*(1), 6765–6816. https://jmlr.org/papers/volume18/16-558/16-558.pdf

Longstaff, F. A., & Schwartz, E. S. (2001). Valuing american options by simulation: A simple least-squares approach. *The Review of Financial Studies*, *14*(1), 113–147. https://doi.org/None

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, *11*(2), 431–441. https://doi.org/10.1137/0111030

Mienye, I. D., & Swart, T. G. (2024). A comprehensive review of deep learning: Architectures, recent advances, and applications. *Information*, *15*(12). https://doi.org/10.3390/info15120755

Moysiadis, G., Anagnostou, I., & Kandhai, D. (2019). Calibrating the mean-reversion parameter in the hull-white model using nns. In *Lecture notes in computer science* (pp. 23–36). https://doi.org/10.1007/978-3-030-13463-1_2

Rebonato, R. (2004). *Interest rate option models: Understanding, analysing and using models for exotic interest rate options* [First published: 27 August 2004]. John Wiley & Sons. https://doi.org/10.1002/9781118673539

Rebonato, R. (2018). Principal components: Theory. In *Bond pricing and yield curve modeling: A structural approach* (pp. 98–107). Cambridge University Press. https://doi.org/https://doi.org/10.1017/9781316694169.006

Sazli, M. H. (2006). A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, *50*(01). https://doi.org/10.1501/commua1-2_0000000026

Shanker, M., Hu, M., & Hung, M. (1996). Effect of data standardization on neural network training. *Omega*, *24*(4), 385–397. https://doi.org/https://doi.org/10.1016/0305-0483(96)00010-2

Sildir, H., Aydin, E., & Kavzoglu, T. (2020). Design of feedforward neural networks in the classification of hyperspectral imagery using superstructural optimization. *Remote Sensing*, *12*(6). https://doi.org/10.3390/rs12060956

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *Nuclear Science, IEEE Transactions on*, *44*, 1464–1468. https://doi.org/10.1109/23.589532

Suo, W., Hull, J. C., & Daglish, T. C. (2009). Volatility surfaces: Theory, rules of thumb and empirical evidence [Accessed: 2025-10-19]. https://ssrn.com/abstract=1521882

Sutton, R. S., & Barto, A. G. (2015). *Reinforcement learning: An introduction* (2nd). MIT Press.

Vollrath, I., & Wendland, J. (2009). Calibration of interest rate and option models using differential evolution [Available at SSRN: https://ssrn.com/abstract=1367502]. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1367502

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: Principles and techniques for data scientists* (1st) [A PDF version is available at https://www.repath.in/gallery/feature_engineering_for_machine_learning.pdf]. O'Reilly Media.

Table 15: **Overview of AI Models Utilized During Thesis Preparation**

AI Model	**Citation Information**
ChatGPT (GPT-5 Model)	OpenAI, *ChatGPT (GPT-5 Model)*, 2025, Version: GPT-5, Used for text refinement including improving grammar, writing style and constiency as well as text proposals. Furtermore, it was used to create `LaTeX` table structures and or querying specific `LaTeX` commands to ensure accurate formatting, https://chat.openai.com
ChatGPT (GPT-5 Thinking Mini)	OpenAI, *ChatGPT (GPT-5 Thinking Mini Model)*, 2025, Version: GPT-5 Thinking Mini, Used for text refinement including improving grammar, writing style and constiency as well as text proposals. Furtermore, it was used to create `LaTeX` table structures and or querying specific `LaTeX` commands to ensure accurate formatting. Note this is the fallback model if the GPT-5 model free quota is exceeded, https://chat.openai.com
Gemini 2.5 Pro	Google DeepMind, *Gemini 2.5 Pro*, 2025, Version: 2.5 Pro, Used for the generation and refinement of code fragments as well as for enhancement of existing code written by the author in terms of readability and computational efficiency. Furthermore it was used to summarize complex theoretical concepts, https://gemini.google.com
Windsurf Chat (Llama 3.1 70B)	Codeium / Windsurf, *Windsurf Chat Model (based on Meta's Llama 3.1 70B)*, 2025, Version: Llama 3.1 70B, Used for coding assistance in the IDE as well as for the creation of Git commit messages, https://codeium.com/windsurf
DeepL Translator	DeepL SE, *DeepL Translator*, 2025, Version: 2025 Release, Used for high-quality translation and linguistic consistency checking, https://www.deepl.com

A Appendix

A.1 Principal Component Analysis on the Yield Curve

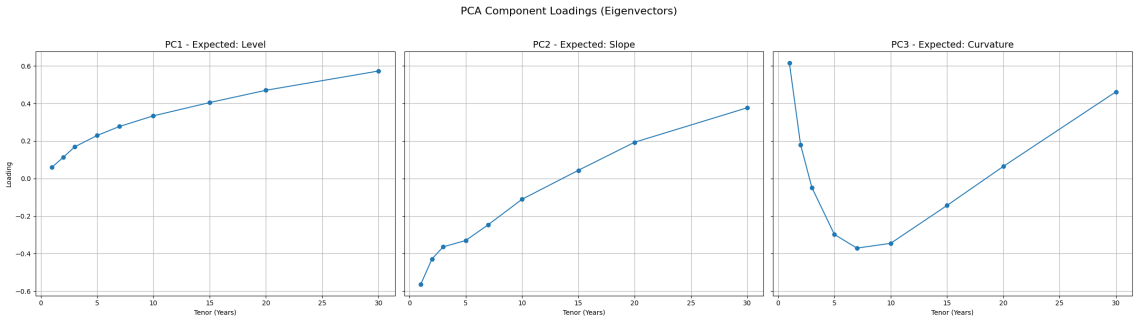

Figure 22: Component Loadings for the First Three Principal Components.

Figure 22 presents the component loadings (eigenvectors) for the first three principal components obtained from the PCA applied to the time series of zero-coupon yield curves. Each panel in the figure displays the loading values on the vertical axis against the nine yield curve maturities on the horizontal axis. The shape and sign pattern of each component provide valuable insights into the underlying yield curve movements that each factor represents.

The first principal component (PC1) exhibits uniformly positive loadings across all maturities, indicating that changes in this factor lead to parallel shifts in the yield curve. This behavior corresponds to the so-called level factor, which reflects movements in the overall level of interest rates.

The second principal component (PC2) displays negative loadings for short-term maturities and positive loadings for long-term maturities. This transition from negative to positive values indicates that changes in this factor modify the steepness of the yield curve, capturing variations in the term spread between short- and long-term rates. This component therefore represents the slope factor.

The third principal component (PC3) is characterized by a distinct hump-shaped pattern, with positive loadings at the short and long ends of the maturity spectrum and negative loadings for intermediate maturities. This configuration reflects changes in the curvature of the yield curve, often referred to as butterfly movements, in which medium-term rates move differently from short- and long-term rates.

Taken together, these three loading structures provide strong visual evidence supporting the economic interpretation of the principal components as level, slope, and curvature factors. The figure thus offers a graphical confirmation of the empirical findings discussed in the main text and highlights the fundamental drivers of yield curve dynamics.

A.2 Hyperparameter Tuning

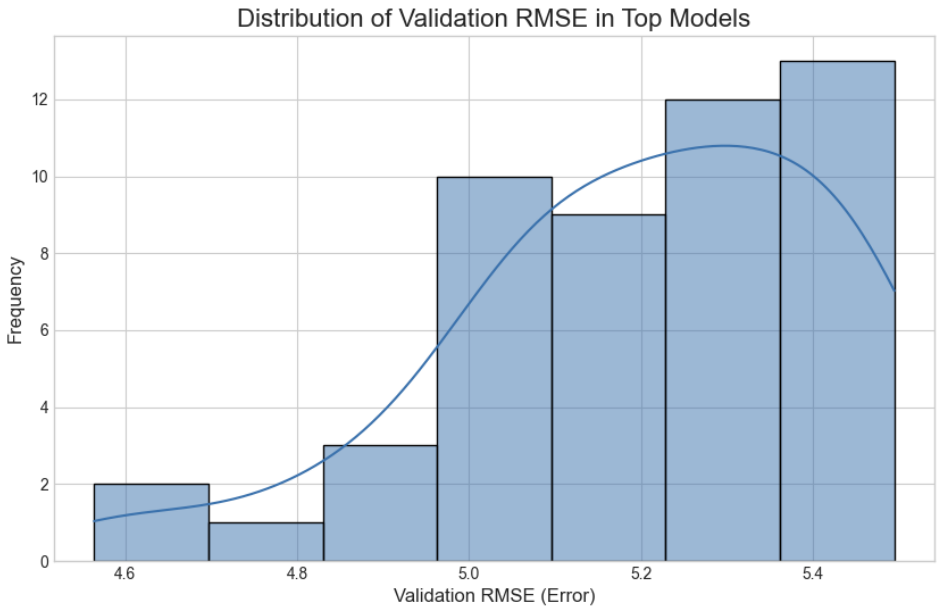

Figure 23: Distribution of Final Validation RMSE for Top 5% of NN from Hyperparameter Search.

The distribution of the final validation RMSE for the top 5% of models evaluated during the hyperparameter search is shown in figure (23), with a kernel density estimate overlaid to illustrate the shape of the distribution. Validation errors within this elite subset range from approximately 4.6 to 5.5 bps, and the distribution is roughly unimodal and bell-shaped. The highest frequency of models achieves an RMSE between 5.3 and 5.4 bps, while a tail of lower-frequency, higher-performing models extends toward an RMSE of 4.6 bps. This indicates that although multiple hyperparameter configurations can achieve high performance, the majority of well-performing models are concentrated in the 5.0-5.4 bps RMSE range. The left-hand tail represents a small number of exceptionally well-configured models, highlighting that achieving top-tier performance is sensitive and requires a precise combination of hyperparameters.

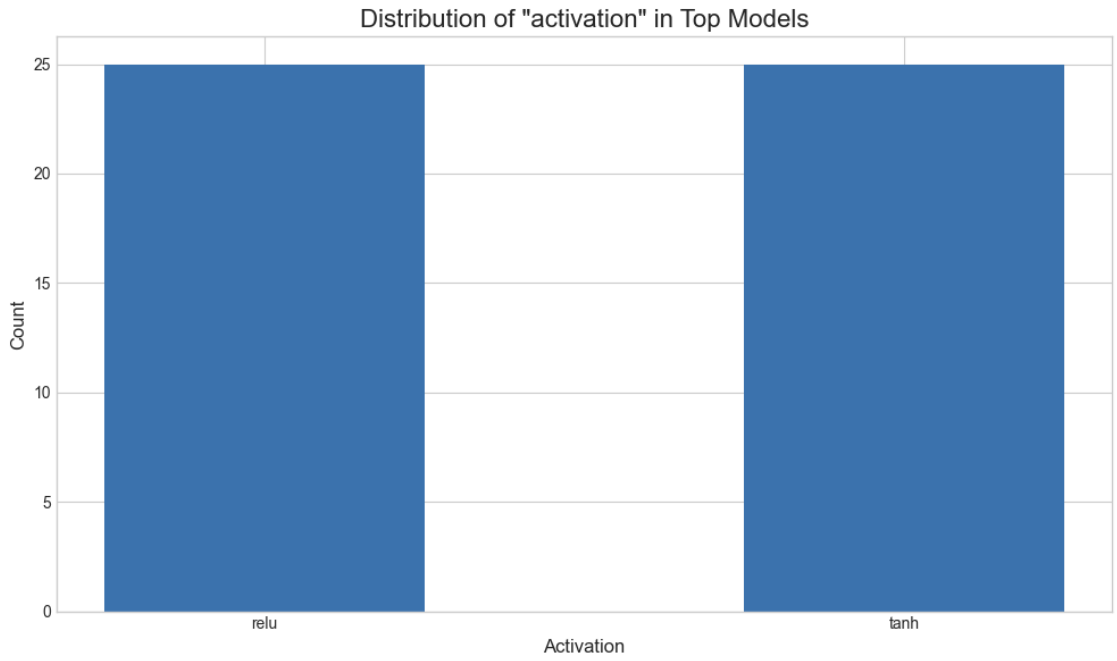

Figure 24: Distribution of Activation Functions in Top 5% of NN from Hyperparameter Search.

The distribution of activation functions across all hidden layers in the top-performing models is shown in figure 24, comparing the usage of the ReLU and tanh functions. Among the top 5% of models, the choice of activation function is evenly split, with ReLU and tanh each employed in 25 of the top 50 models. This finding indicates that, for this specific modeling task, the selection between ReLU and tanh does not significantly impact model performance. Both functions appear equally capable of enabling the network to learn the complex mapping required, suggesting that performance is more sensitive to other architectural and training hyperparameters.

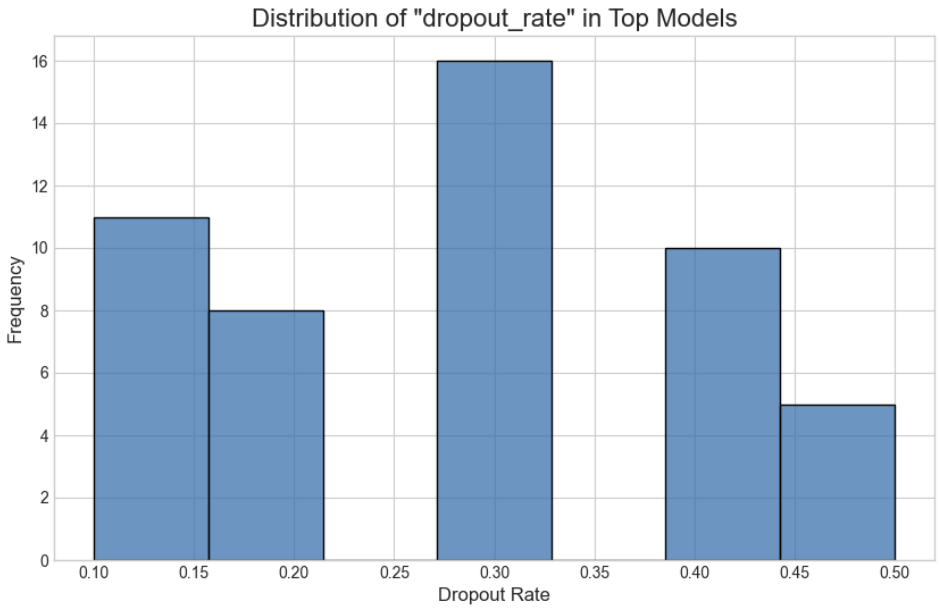

Figure 25: Distribution of Dropout Rate in Top 5% of NN from Hyperparameter Search.

The distribution of dropout rates among the top-performing models with dropout enabled is illustrated in figure 25, with the x-axis representing the fraction of neurons deactivated during training. The distribution is multi-modal, exhibiting notable peaks in the ranges of 0.10-0.15, 0.25-0.30, and 0.40-0.45, with the highest frequency occurring around 0.25-0.30. This pattern suggests that the optimal level of regularization is highly dependent on other hyperparameters, such as network depth and width. Larger or deeper networks may require more aggressive dropout, whereas smaller networks may perform better with lower dropout rates. Although the peak near 0.30 indicates a commonly effective configuration, the overall spread demonstrates that no single dropout rate serves as a universally optimal setting.

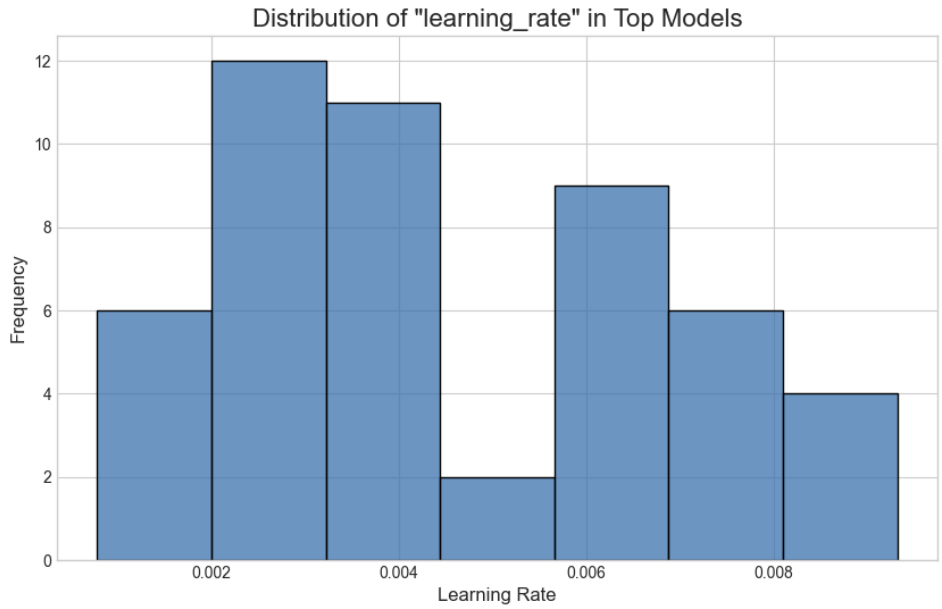

Figure 26: Distribution of Learning Rates in Top 5% of NN from Hyperparameter Search.

The distribution of learning rates among the top 5% of models is shown in figure 26, illustrating the frequency of different rates selected during the hyperparameter search. While learning rates are spread across the explored range, they are most heavily concentrated between 0.002 and 0.007, with a distinct peak in the 0.002-0.003 bin. Very high learning rates above 0.008 and very low rates below 0.002 occur infrequently. This pattern indicates that a moderately low learning rate is important for achieving optimal performance. Excessively high rates can cause unstable training and prevent convergence, whereas very low rates may result in excessively slow learning. The concentration within the moderate range suggests a well-defined "valley" in the loss landscape that can be effectively navigated with appropriately chosen learning rates.

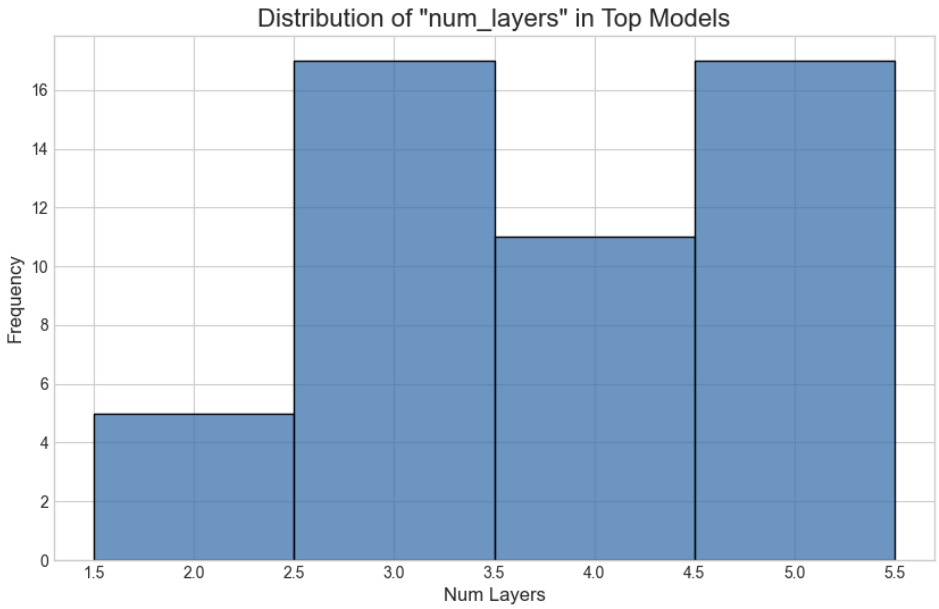

Figure 27: Distribution of Number of Hidden Layers in Top 5% of NN from Hyperparameter Search.

The distribution of the total number of hidden layers among the top 5% of models is shown in figure 27, with the x-axis representing the number of hidden layers. The distribution is strongly skewed towards deeper networks, with the most frequent configurations comprising three, four, or five hidden layers. Shallow networks with only two layers occur significantly less often among the top-performing models. This pattern indicates that network depth is a critical factor for effectively modeling the problem, suggesting that the relationships between the market data and the Hull-White parameters are hierarchical and complex, and require multiple layers of non-linear transformations to be captured accurately.

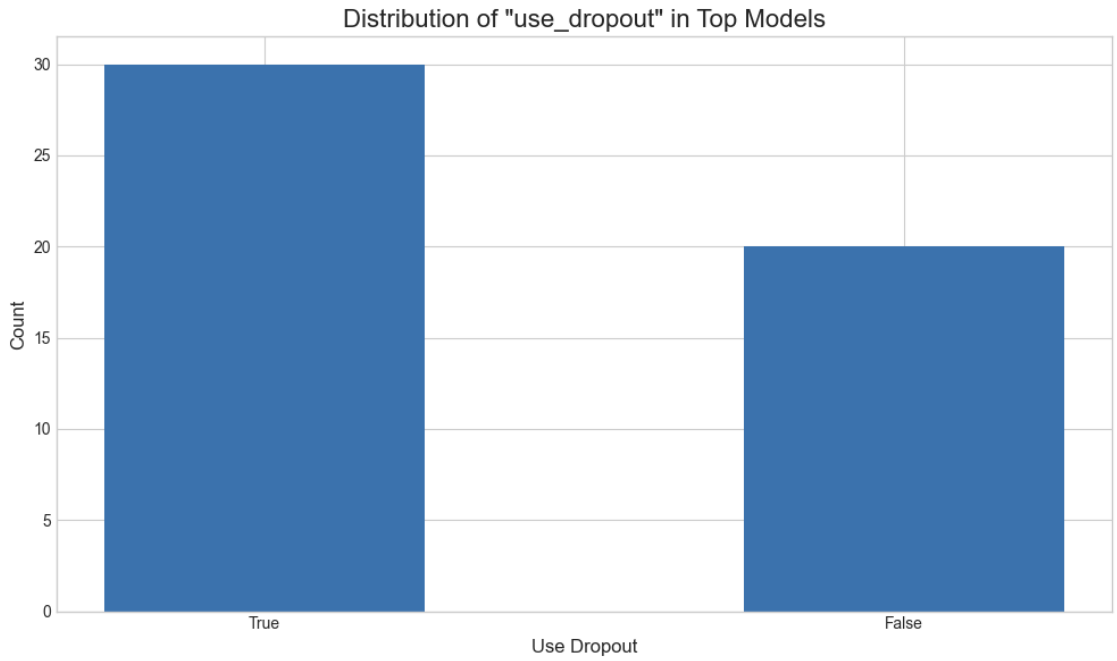

Figure 28: Distribution of Use of Dropout in Top 5% of NN from Hyperparameter Search.

The distribution of the "Use Dropout" hyperparameter among the top-performing models is presented in figure 28, indicating whether a dropout layer was enabled or disabled for regularization. Among the top 50 models, a majority (30) employed dropout, while a substantial minority (20) did not. This suggests that dropout generally contributes to improved generalization on this dataset. However, the fact that a significant portion of top models performed best without dropout indicates that its effectiveness is not universal and likely depends on the specific architecture of the network.

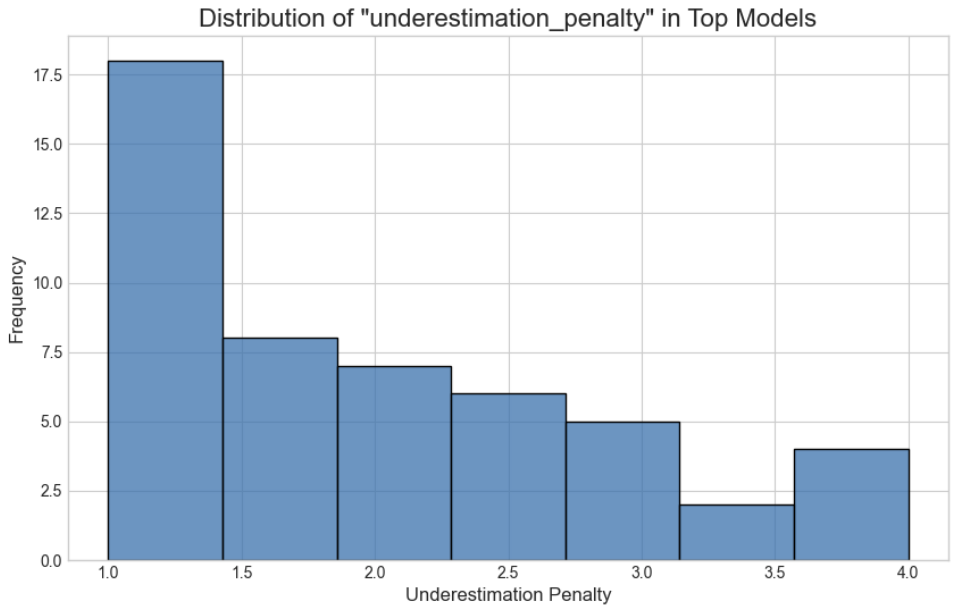

Figure 29: Distribution of Underestimation Penalty in Top 5% of NN from Hyperparameter Search.

The distribution of the custom underestimation penalty hyperparameter among the top-performing models is shown in figure 29, where a value of 1.0 represents no additional penalty. The distribution is strongly right-skewed, with the majority of the best models employing a low penalty between 1.0 and 1.5. Higher penalty values occur infrequently among top models. This indicates that imposing a large penalty for underestimation is detrimental to performance when evaluated with a symmetric metric such as RMSE. While the penalty is financially motivated, excessive conservatism introduces systematic overestimation, increasing overall error. The most effective models capture the underlying distribution accurately using a balanced or only slightly asymmetric loss function.

A.3 Final Model Training

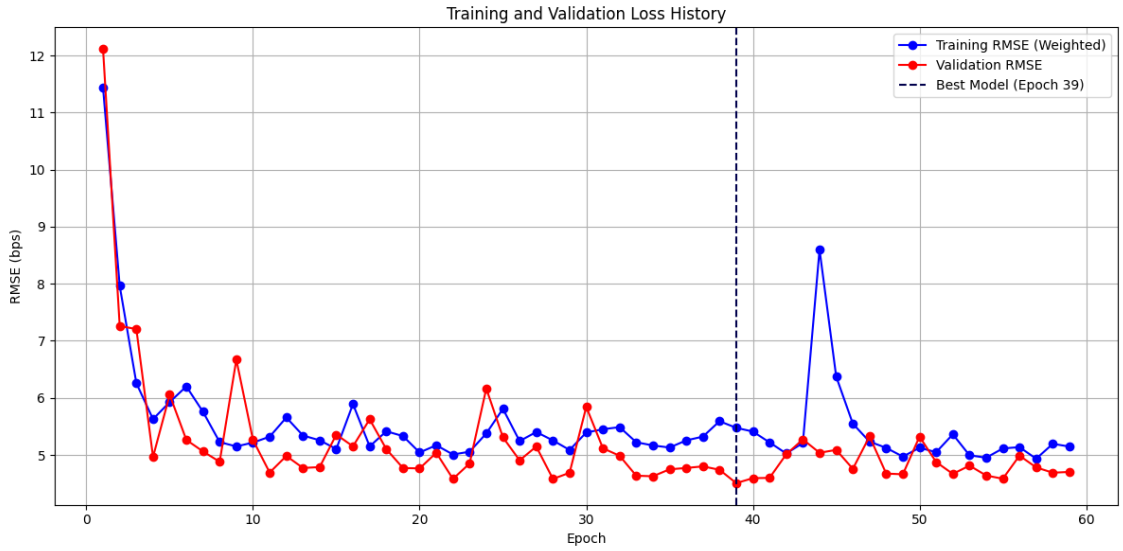

Figure 30: Training History of Final Model using Best Hyperparameters.

The training and validation loss history of the best-performing model over 59 epochs is illustrated in figure 30, where the y-axis represents the RMSE in basis points and the x-axis represents the training epoch. The blue line denotes the weighted training RMSE, incorporating an asymmetric penalty for underestimation, while the red line shows the standard, unweighted validation RMSE. A vertical dashed line at epoch 39 indicates the point at which the model achieved its lowest validation error and was selected as the final model through early stopping.

The training process exhibits several distinct phases. An initial rapid convergence occurs within the first ten epochs, during which both training and validation RMSE decrease sharply from over 12 bps to approximately 5-6 bps. This is followed by a plateau phase characterized by noisy fluctuations without a significant long-term downward trend. Throughout this phase, the validation RMSE consistently remains lower than the weighted training RMSE, reaching its global minimum at epoch 39. After this point, the validation error does not improve over the subsequent twenty epochs, triggering early stopping and restoring the model weights from epoch 39.

The learning curve provides several insights into model behavior and the optimization landscape. The steep initial decrease indicates that the network architecture and learning rate effectively capture the principal patterns in the training data. The subsequent noisy yet stable plateau reflects the complex, non-convex loss surface typical of financial model calibration, which the stochastic optimizer successfully navigates without divergence. The persistent gap between training and validation RMSE is a direct consequence of the asymmetric loss function: the elevated training loss reflects the explicit penalty for

underestimation, while the lower validation RMSE provides an unbiased measure of generalization. Finally, early stopping acts as an effective regularization mechanism, ensuring that the final model generalizes well to unseen data rather than overfitting, as evidenced by the stability of the validation curve and the model's robust predictive performance.

A.4 Methodology of the Generalized Island Model

The Generalized Island Model represents a parallelization scheme that extends the concept of isolated yet interacting populations, originally proposed within the framework of Genetic Algorithms, to a broader class of optimization algorithms. Its fundamental idea is to enhance both computational efficiency and search diversity through the coexistence of multiple, intercommunicating subpopulations referred to as islands. The approach not only accelerates computation via parallel execution but also enriches the optimization dynamics by enabling structured information exchange among subpopulations.

The model belongs to the family of coarse-grained parallelization schemes, where each island evolves largely independently, performing its own optimization process, while periodically exchanging individuals or solutions with other islands. Historically, this framework emerged in the 1980s as a natural extension of Genetic Algorithms, motivated by the need to mitigate premature convergence and improve solution diversity through distributed evolutionary processes. Subsequent research established the superiority of the island model over monolithic, global population schemes, both theoretically and empirically. Moreover, the paradigm has proven effective beyond evolutionary computation, finding applications in other metaheuristics such as Particle Swarm Optimization, where multiple swarms act as parallel islands exchanging particles at regular intervals.

The generalized island model can be applied to a broad spectrum of optimization algorithms as long as they share a compatible solution representation and can incorporate migration mechanisms. Its flexibility allows for the construction of heterogeneous archipelagos, where different islands employ distinct algorithms yet still participate in coordinated information exchange.

Formally, the model defines an archipelago $A = \langle I, T \rangle$, where $I = \{I_1, I_2, \ldots, I_n\}$ represents the set of islands (also known as demes), and T denotes the migration topology, modeled as a directed graph whose vertices correspond to the islands and whose edges specify permissible migration paths. Each island I_i is characterized by the quadruple $\langle A_i, P_i, S_i, R_i \rangle$, where A_i is the optimization algorithm employed by the island, P_i denotes its local population associated with the problem under study, S_i defines the migration-selection policy determining which individuals are sent to other islands, and R_i specifies the migration-replacement policy governing the integration of received individuals into the local population.

The operational flow of the model alternates between independent evolution and coordinated migration phases. During evolution, each island applies its optimization operator $P' \leftarrow A_i(P, \mu)$, where μ represents the migration interval. At migration epochs, each island selects a subset of individuals according to its selection policy S_i and sends them to other islands following the connectivity defined by the topology T. Upon receiving migrants, the destination islands incorporate them into their populations based on the replacement policy R_i. This decoupling ensures that the internal workings of each optimization process remain independent of the migration mechanism, facilitating modularity and scalability.

The effectiveness of the generalized island model depends on the careful configuration of several parameters. The number of islands n influences both solution diversity and robustness, as a larger number of islands can explore different regions of the search space or employ distinct parameterizations. The migration topology T governs the communication structure, with common configurations including ring, torus, and fully connected networks, each characterized by distinct information diffusion properties. The migration interval μ controls the frequency of exchanges, requiring a balance between exploration and convergence: overly frequent migrations can lead to homogenization, while infrequent ones may reduce cooperative benefits. Migration can be implemented synchronously, ensuring deterministic behavior at the cost of slower execution, or asynchronously, enhancing scalability at the expense of reproducibility. Finally, the migration-selection and migration-replacement policies S_i and R_i define which individuals are exchanged and how they are assimilated, often incorporating stochastic or elitist strategies and determining the overall migration rate.

Note: This subsection is entirely based on the description provided in (Izzo et al., 2012, pp. 151-169).

This framework was implemented to calibrate the eight parameters of a Hull-White one-factor model with piecewise-constant volatility. The number of islands was set to equal the number of available CPU cores (16), with a total population of 512 individuals distributed among them (32 individuals per population). The migration policy was configured with a unidirectional ring topology, where, at an interval of every 15 generations, the two best-performing individuals from one island migrate to the next. The replacement strategy involved removing the two worst-performing individuals in the receiving population to accommodate the migrants, a common elitist approach. Each island independently executed a genetic algorithm for a total of 150 generations. Within each island, parent selection was managed by a tournament of size three. New individuals were generated through an averaging crossover operator, and a Gaussian mutation operator was applied with a 90% probability. To balance exploration and exploitation, the mutation strength was dynamically adjusted, decaying exponentially from an initial value of 1.0 to a final value of 0.05 over the generations. Finally, an elitism count of one ensured that the best individual of each generation was preserved. The calibration was performed for a sin-

gle day, 03.08.2025 (one day before the testing set starts) to determine the initial guess parameters for the LM algorithm and concluded after 5576.09 seconds, achieving a final RMSE of 4.21 basis points.

A.5 Pricing Comparison

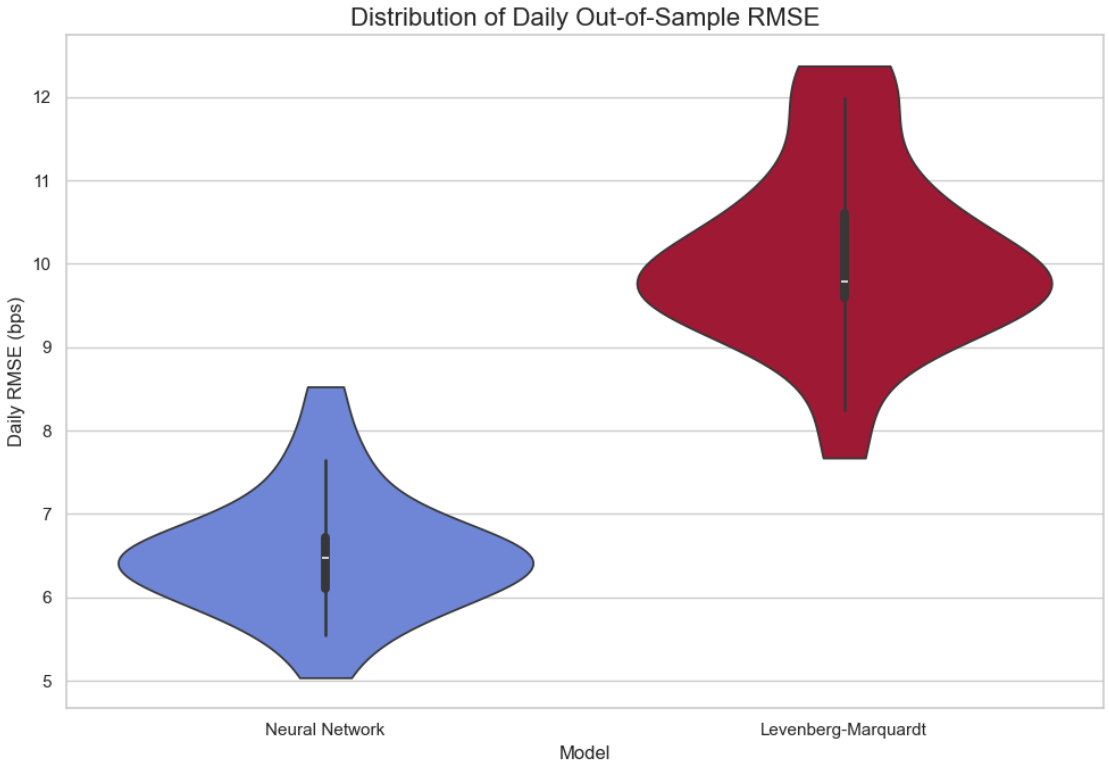

Figure 31: Distribution of RMSE between NN and LM.

The daily out-of-sample RMSE (no vega weighting was applied) for the NN and LM calibration methods is presented using a violin plot (figure 31), where the width represents the frequency of each daily error and the inner box plot indicates the median and interquartile range. A clear separation between the two methods is observed. The NN's error distribution is centered at a lower level, with a median of approximately 6.5 bps, and is considerably more compact, with most daily errors between 5.5 and 7.5 bps. In contrast, the LM method exhibits a higher median of around 9.8 bps and a substantially wider dispersion, ranging from approximately 8 to 12 bps. Notably, the 75th percentile of the NN distribution is lower than the 25th percentile of the LM distribution.

These results provide strong evidence of both the superior accuracy of the NN approach. The consistently lower RMSE indicates that the NN generalizes more effectively to unseen market data, while the narrower distribution demonstrates more reliable and predictable performance, which is essential for practical risk management. The minimal overlap between the core distributions further emphasizes that the NN's worst-day perfor-

mance typically exceeds the average performance of the LM method.

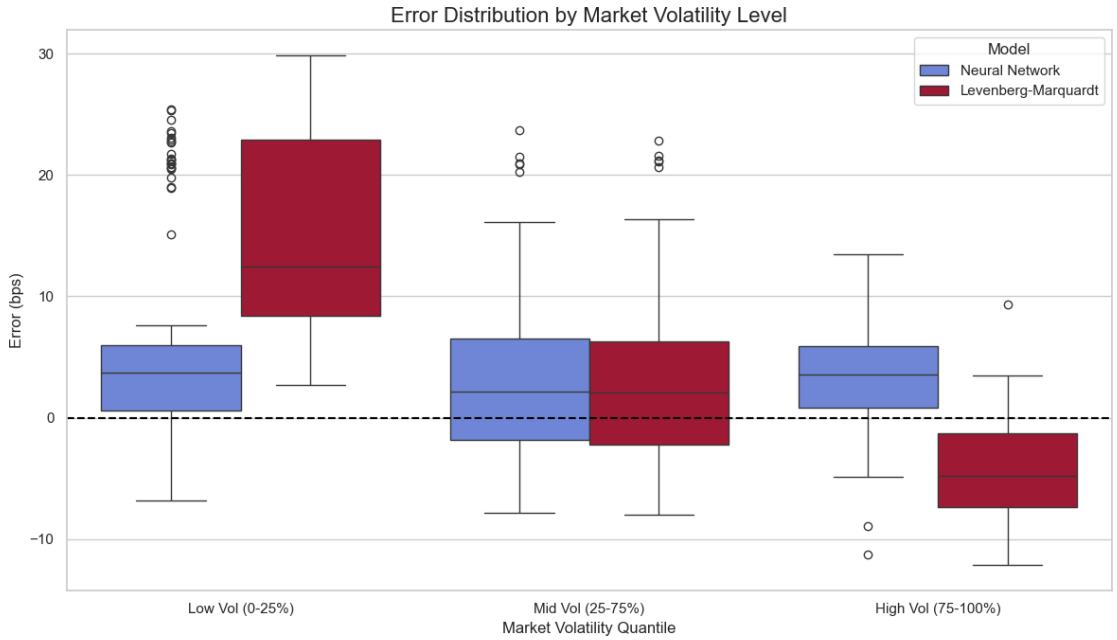

Figure 32: Distribution of RMSE between NN and LM by Swaption Volatility Quantiles.

The instrument-level pricing errors (no vega weighting was applied), defined as the difference between model-implied and market-observed volatilities, are analyzed across three market volatility quantiles—Low (0-25%), Mid (25-75%), and High (75-100%)—using a grouped box plot (figure 32), with a dashed line at zero indicating perfect predictions. The LM method exhibits large, systematic, and offsetting biases. In the low-volatility regime, it shows a strong positive bias, with a median error exceeding 10 bps, indicating overpricing, while in the high-volatility regime, the bias reverses to a median error of approximately -5 bps, indicating underpricing. In contrast, the NN maintains a consistent and less biased profile across all volatility regimes, with median errors closer to zero and substantially smaller interquartile ranges, reflecting lower prediction variance.

This analysis highlights a key limitation of the traditional LM optimizer: its inability to simultaneously fit instruments across the full volatility surface results in severe, systematic mispricings, overcompensating in one regime to correct errors in another. The NN, however, learns a more flexible and robust mapping that generalizes effectively across the volatility spectrum. Its consistently lower bias and variance demonstrate a superior capacity to capture the complex dynamics of the market, making it a more reliable tool for pricing and risk management.

A.6 SHAP Values

The SHAP feature importance and summary plots provide a detailed understanding of how individual features influence the model's predictions across different parameters. The

SHAP feature importance plot presents an aggregated view by ranking features according to their mean absolute SHAP values across all predictions. The longer a bar appears in this plot, the greater the average magnitude of that feature's contribution to the model's output, irrespective of direction. This visualization therefore highlights the features with the strongest overall predictive power. In contrast, the SHAP value summary plot (beeswarm plot) offers a granular perspective, showing the distribution of individual SHAP values for each feature across all samples. Each dot represents a single prediction, positioned horizontally according to its SHAP value, while the color encodes the original feature value from low (blue) to high (red). This dual encoding makes it possible to discern not only the importance but also the directionality and consistency of each feature's effect.

For the parameter a_1, which governs mean reversion, the feature importance plot identifies *curvature_10y20y30y*, representing the long-end curvature of the yield curve, as the most influential feature, followed by slope measures such as *slope_3m10y* and *slope_5y30y*. The summary plot shows that high curvature and steep slope values, indicated by red dots, are associated with negative SHAP values. This suggests that a pronounced long-end curvature and a steep yield curve reduce the predicted mean reversion parameter. Such a relationship aligns with financial intuition: when the market anticipates substantial future rate movements, the short rate becomes less anchored, leading to weaker mean reversion.

Across the parameters σ_1 to σ_7, which represent piecewise volatilities, a consistent pattern emerges. In nearly all cases, *curvature_10y20y30y* is the single most important predictor. This finding implies that the model perceives the SHAPe of the far end of the yield curve as the key indicator for volatility across the entire term structure. The neural network thus interprets long-end curvature as a proxy for systemic uncertainty in interest rates. At the same time, the model demonstrates adaptive behavior in weighting secondary features. For short-term volatilities (σ_1, σ_2), features describing the immediate yield curve SHAPe, such as *PC_Curvature* and slope measures, gain importance. In contrast, for long-term volatility (σ_7), the *MOVE_VIX_Ratio* becomes a critical factor, reflecting the model's capacity to incorporate cross-asset market stress into its predictions.

The SHAP beeswarm plot provides a comprehensive visual summary of these interactions. Features are ranked by their average impact, displayed along the y-axis, while the x-axis indicates how strongly a given feature value influences a specific prediction. Positive SHAP values correspond to features that increase the predicted parameter, whereas negative values indicate a downward effect. The color gradient adds a third layer of interpretation by mapping original feature values, thereby revealing relationships such as positive or negative correlations between a feature's level and its effect on the output. The density and spread of the dots further illustrate the variability and context-dependence of each feature's influence, enabling the identification of stable versus conditional effects.

When examining all predicted parameters together, the beeswarm plots reveal a consistent

and economically interpretable decision structure within the neural network. The long-end yield curve curvature emerges as the universal driver: high curvature values tend to decrease the mean reversion parameter a_1 while simultaneously increasing volatility parameters σ_i. This dual effect indicates that the model has learned to associate a more curved yield curve with heightened market uncertainty and weaker rate anchoring. Beyond this primary driver, the model differentiates between parameter-specific influences. For mean reversion, the dominant secondary features are yield curve slopes, whose high values lower a_1 in line with the expectation that steep curves imply rising future rates. For short-term volatilities, the model prioritizes immediate yield curve factors, while for long-term volatility, it emphasizes inter-market stress measures such as the *MOVE_VIX_Ratio*.

A particularly notable finding arises from the non-linear relationship between the *MOVE_VIX_Ratio* and long-term volatility σ_7. High values of this ratio, indicative of bond market stress, are often associated with negative SHAP values, implying that periods of extreme market tension can precede suppressed long-term volatility. This pattern reflects complex, regime-dependent market dynamics, such as the "flight-to-quality" phenomenon or central bank interventions following crises. Finally, the principal component features (*PC_Level*, *PC_Slope*, *PC_Curvature*) behave in accordance with financial expectations. For instance, higher *PC_Level* values, representing generally higher interest rates, correspond to higher predicted volatilities, consistent with empirical observations that volatility tends to increase in high-rate environments. These findings confirm that the model has successfully internalized financially coherent relationships and interprets engineered features in an economically meaningful manner.

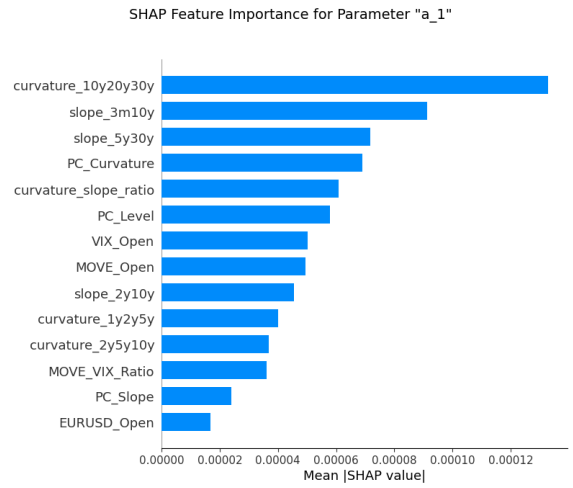

Figure 33: SHAP Feature Importance for α.

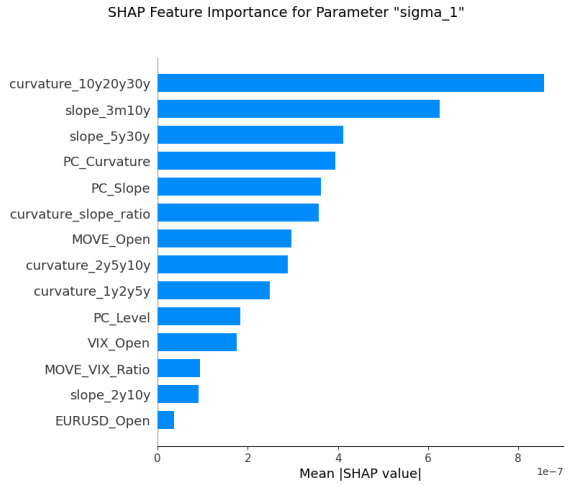

Figure 34: SHAP Feature Importance for σ_1.

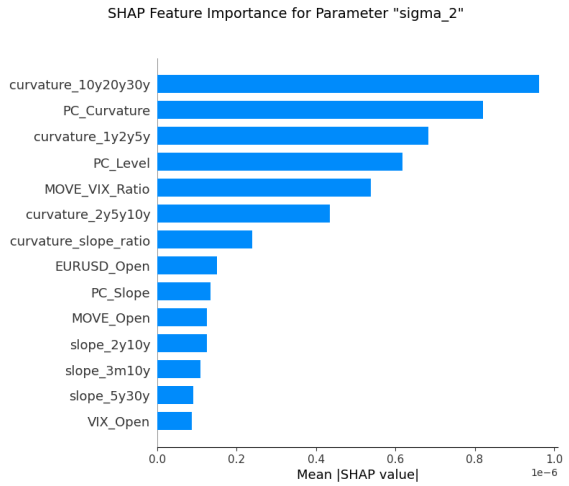

Figure 35: SHAP Feature Importance for σ_2.

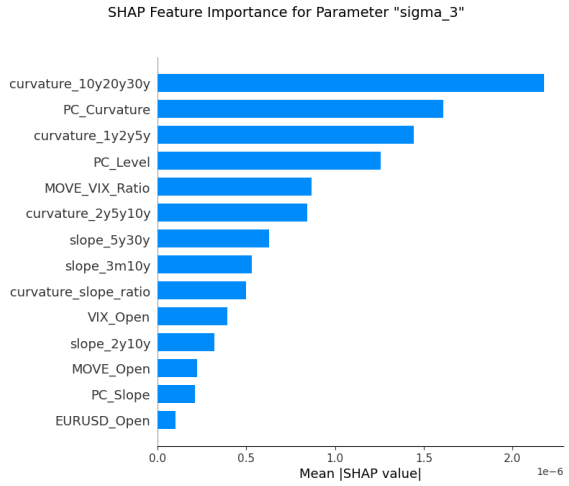

Figure 36: SHAP Feature Importance for σ_3.

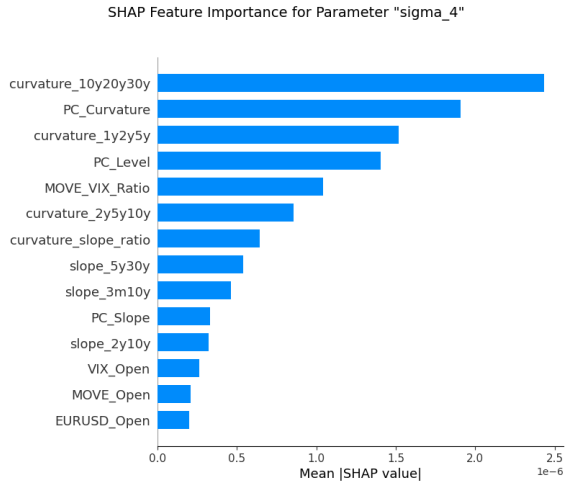

Figure 37: SHAP Feature Importance for σ_4.

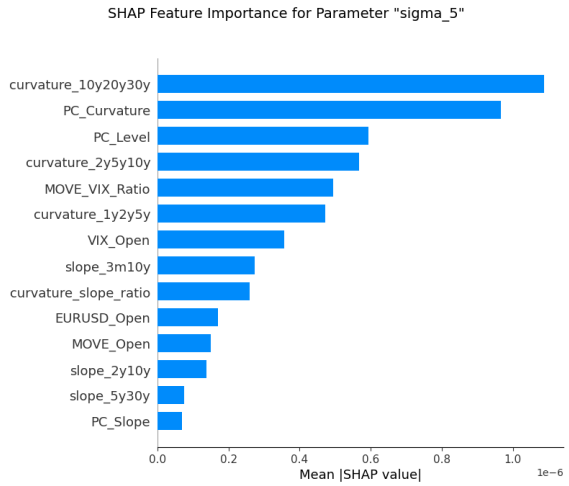

Figure 38: SHAP Feature Importance for σ_5.

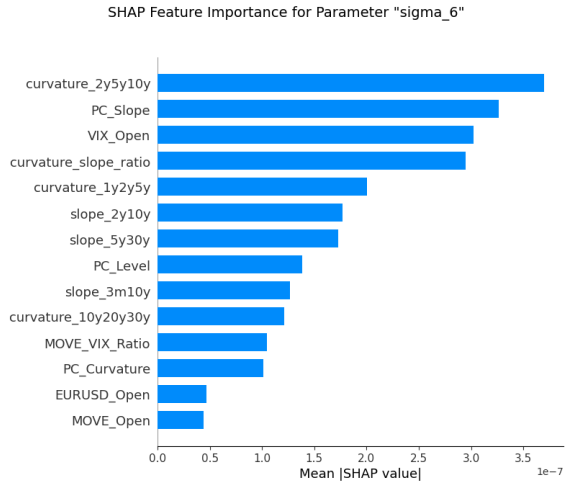

Figure 39: SHAP Feature Importance for σ_6.

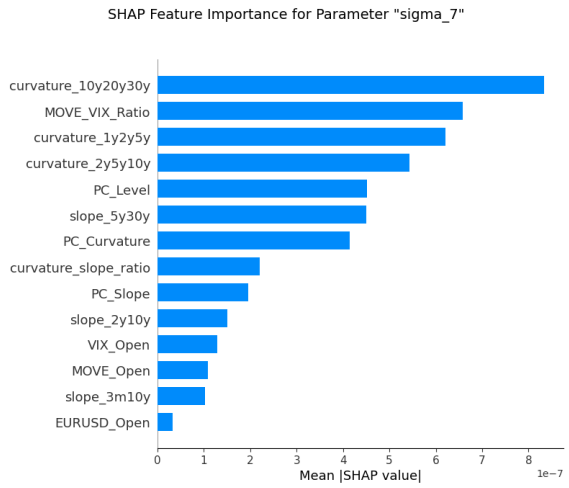

Figure 40: SHAP Feature Importance for σ_7.

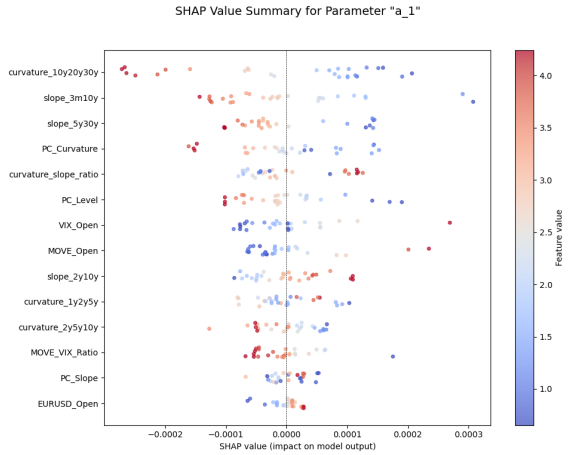

Figure 41: SHAP Feature Summary for α.

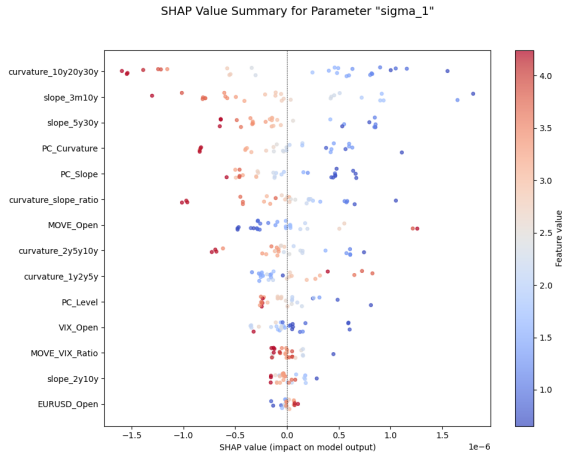

Figure 42: SHAP Feature Importance for σ_1.

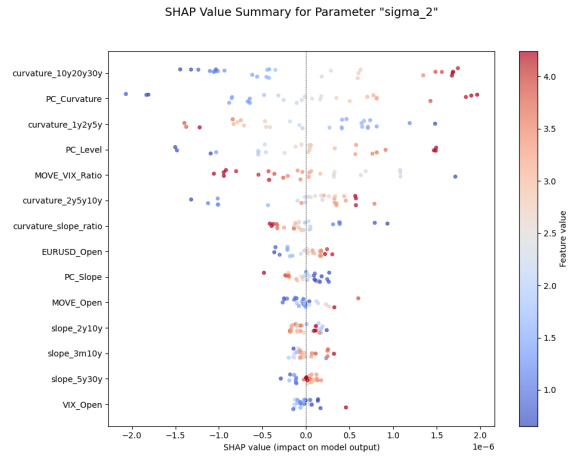

Figure 43: SHAP Feature Importance for σ_2.

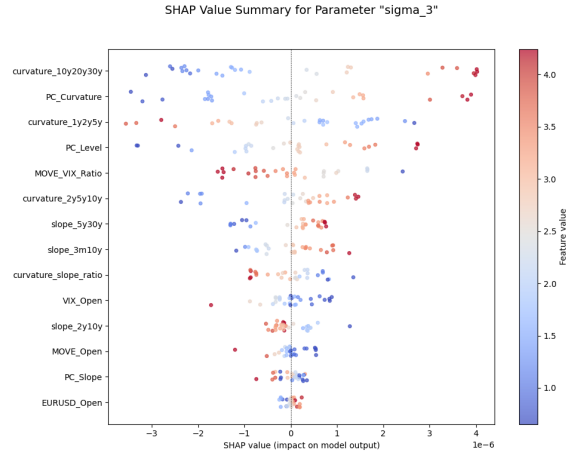

Figure 44: SHAP Feature Importance for σ_3.

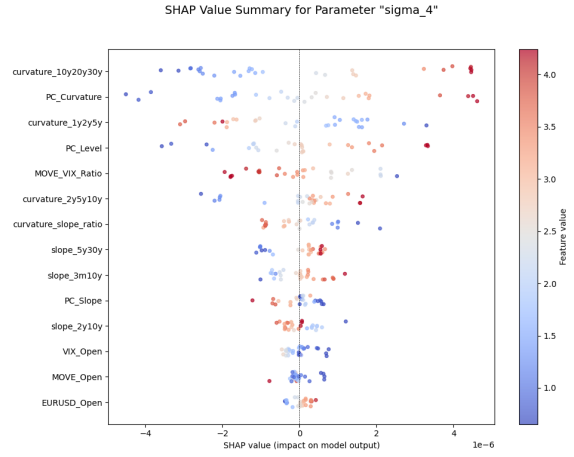

Figure 45: SHAP Feature Importance for σ_4.

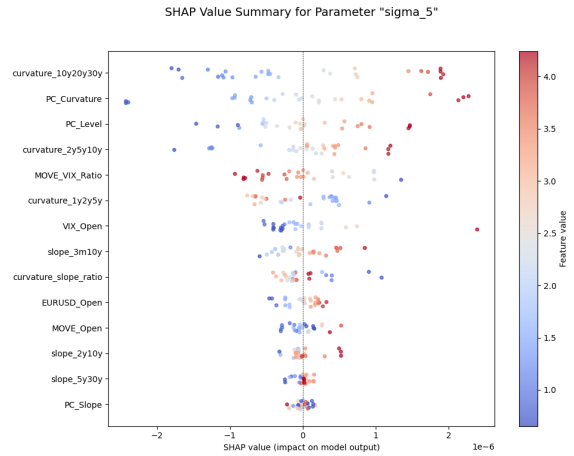

Figure 46: SHAP Feature Importance for σ_5.

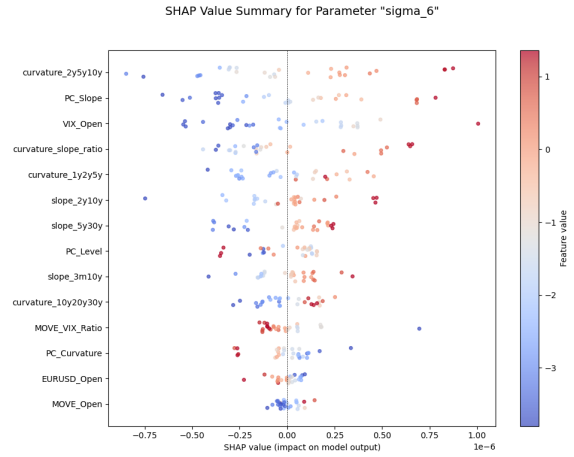

Figure 47: SHAP Feature Importance for σ_6.

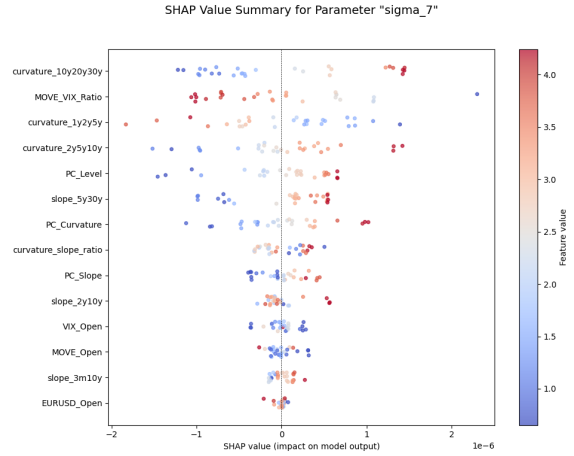

Figure 48: SHAP Feature Importance for σ_7.

A.7 Code and Data Availability

For the purposes of transparency and to facilitate further research, the complete source code implementing the methodologies, experiments, and analyses detailed within this the-

sis has been made publicly available. The project repository is hosted on GitHub and can be accessed via the following URL: https://github.com/skyi28/Calibration-of-GSR-Models-using-Neural-Networks

It is important to note that the repository contains only the source code and not the underlying market data utilized in the empirical study. The dataset, which was gathered from the Bloomberg Terminal, is not provided due to the proprietary nature of the information and the restrictive terms of the Bloomberg data license agreement, which expressly forbids the redistribution of its data.

B Affidavit

I, the undersigned, Benedikt Grimus, hereby declare under penalty of perjury that the Master's thesis entitled:

A Comparison Between Traditional and Machine Learning Based Calibration of the One Factor Hull-White Model

is my own original work and has been written by me independently. I further declare that I am the sole author of this thesis. The work submitted is the result of my own independent investigation and research. I have not used any sources or aids other than those acknowledged and cited in accordance with the academic regulations of Zürcher Hochschule für Angewandte Wissenschaften.

All passages, ideas, data, and graphics, whether quoted directly or paraphrased, that have been drawn from other sources, including but not limited to books, articles, and online resources, have been explicitly acknowledged and cited. I have made a clear distinction between my own contributions and the work of others.

I declare that I have used generative AI tools to assist in the preparation of this thesis. The specific tools and their application are detailed in the methodology section and in the reference section of this work. All outputs from these AI tools have been critically reviewed, verified, and edited by me, and I take full responsibility for the entire content of this thesis.

This thesis, either in whole or in part, has not been previously submitted for any other examination, degree, or qualification at this or any other institution.

I am fully aware of the university's policies on plagiarism and academic misconduct. I understand that any violation of these regulations will result in severe penalties, including the failure of this thesis and potential disciplinary action. I consent to my thesis being checked for plagiarism using anti-plagiarism software.

I affirm that this declaration is true and correct to the best of my knowledge and belief.

Koroni, Greece, 18th November 2025 *Benedikt Grimus*