

Sinan_Bassam_02066528

by Bassam SINAN

Submission date: 06-Sep-2022 04:09PM (UTC+0100)

Submission ID: 185750743

File name: Sinan_Bassam_02066528.pdf (5.04M)

Word count: 18862

Character count: 88020

**Imperial College
London**

IMPERIAL COLLEGE LONDON

DEPARTMENT OF MATHEMATICS

**Deep hedging of Autocallables with
rough Bergomi model**

Author: Bassam SINAN (CID: 02066528)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2021-2022

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

I would like to thank my industry supervisor, Gordon Lee, for his time, precious remarks and guidance throughout the thesis. I would also like to thank my academic supervisor Dr. Antoine Jacquier for his help, especially regarding the volatility models and my tutor, Dr Mikko Pakkanen, who has vastly contributed to my understanding of deep hedging.

Last but not least, I would like to thank my family for their support along the way.

Abstract

"Greek Hedging is a legacy approach". This approach once justified by lack of data and computational power is starting to fade away as we enter the era of machine learning. Greek Hedging, almost exclusively used in many financial institutions to quantify risk is a consequence of conventional pricing models. However, the GFC aftermath gave birth to a new idea which is indifference pricing, a framework under which the price of an instrument is given by the hedge the position that the trader will implement. This approach being model-free essentially allows one to both price and hedge without having to rely on any kind of pricing models.

In 2018, Buehler et al. [7] publish a paper in which they show how training neural networks alongside indifference pricing can be used to price any portfolio of liabilities in a tractable model which is able to incorporate transaction costs. This ability to price any kind of liabilities is a breakthrough as we do not need some convoluted analytical pricing model for the valuation of complex instruments, especially when there is no consensus on the pricing.

Such an instrument is for instance, an autocallable which is one of the most popular structured products in the world. In essence, the holder sells a barrier in exchange for coupons in the future. The key issue with the product is that they are notoriously hard to hedge as the gamma (change of delta to the spot) can be very high around the barriers and hence hard to manage. There are many models in this space like Heston, Local volatility, etc. and the emergence of Rough Volatility models such as rough Bergomi have promised a more parsimonious volatility model that looks more like the real world.

Keywords: Rough Bergomi, Neural Networks, Long Short Term Memory, Volatility Skew, Autocallables, Deep Hedging

Contents

1	Theoretical prerequisites: the rough Bergomi model	9
1.1	The history of volatility	9
1.2	Fractional Brownian Motion	11
1.3	Bergomi model	12
1.3.1	Motivations	12
1.3.2	Dynamics for the Bergomi model	12
1.4	Rough Bergomi model	13
1.4.1	Deriving the rough Bergomi model under the \mathbb{P} measure	13
1.4.2	Pricing under measure \mathbb{Q}	14
1.4.3	Dynamics of the rough Bergomi model under the \mathbb{Q} measure	15
1.5	Simulating the Volterra Process: the hybrid scheme	15
1.5.1	Calibration	17
2	Problem setting and utility indifference pricing	19
2.1	Market setting in discrete time	19
2.2	Convex Risk measure	20
2.3	Indifference Pricing	20
2.4	Exponential Utility Indifference Pricing	21
3	Artificial networks: FNN and LTSM	23
3.1	Introduction	23
3.2	Feedforward neural networks (FFN)	24
3.2.1	Architecture	24
3.2.2	Activation functions	25
3.2.3	Universal approximation theorem	25
3.2.4	Loss functions	27
3.2.5	Minibatch	27
3.2.6	Epochs	27
3.2.7	Stochastic Gradient Descent	27
3.2.8	Backpropagation	28
3.3	Long short-term memory (LTSM)	29
3.3.1	Generic Recurrent Neural Networks	30
3.3.2	LSTM	31
4	Description of Autocallables and their use	33
4.1	Mechanisms and features	33
4.1.1	Coupon and Redemption	33
4.1.2	Barrier observation	34
4.1.3	Adding downside with a Down-and-in Put	34
4.1.4	Product Payoff with possible scenarios	34
4.2	Pricing Approach	35
4.2.1	Monte Carlo	36
4.2.2	Static hedging: breaking down the product into a combination of options	36
4.3	Hedging issues	37
4.3.1	Downside: replicated by a DIP	37
4.3.2	Upside: replicated by a strip of digital calls	37

5	Deep Hedging Autocallables: numerical results	40
5.1	Model setting	40
5.1.1	Neural Networks	40
5.1.2	Product specification	40
5.2	The hedging strategy	41
5.2.1	Handling corner cases	41
5.2.2	Comparing the calibrated models	45
A	Deep Hedging under Neural Networks	49
A.1	FNN hedging strategy	50
A.1.1	Evolution of the hedging ratio across varying spot prices	50
A.1.2	FNN: Evolution of the hedging ratio over the realized paths of two scenarios	53
A.2	LSTM	56
A.2.1	Evolution of the hedging ratio across varying spot prices	56
A.2.2	Evolution of the hedging ratio over the realized paths of two scenarios	59
	Bibliography	63

List of Figures

1.1	Average Implied volatility profile of SP500 options as March 1999 demonstrating this notion of volatility cube/surface	10
1.2	Implied volatility levels as of September 2, 2022 after market close, extracted from Apple call option prices, expiring September 23, 2022. As we can see, implied volatility decreases with strike and displays a skewed shape, hence the name "skew".	10
1.3	S&P 500 Implied volatility taken from market call prices (mid) as of 2 September 2022 after market close. Parameters are summarized in table 1.1.	18
3.1	The history of Neural Networks over the last decades, source: http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html	23
3.2	Graphical representation of a neural network with $r = 3, I = d_0 = 4, d_1 = 6, d_2 = 5$ and $O = d_3 = 3$ (source from Deep Learning lecture notes).	25
3.3	Non-exhaustive list of one-dimensional activation functions (adapted from Wikipedia).	26
3.4	Non-exhaustive list of multi-dimensional activation functions (adapted from Wikipedia).	27
3.5	FFN feed the information in one straight line, but in RNN information is fed like in a loop which enables the model to remember past predictions [14, Figure 2, page 6].	30
3.6	In a standard RNN (top image), the repeating module contains a single layer whereas in a LSTM (bottom), it contains four interacting layers. Source: Christopher Olah, used with permission.	32
4.1	Autocall payoff at maturity with Protection Barrier = Coupon Barrier = 70, Autocall Barrier 120, Coupon level = 20. This figure clearly shows that the final Payoff can be broken down into a combination of a DIP to model the downside and a digital call to model the upside with the coupon being paid.	35
4.2	Price and delta for a short European put with $r = 0, \sigma = 0.3, T = 1, K = 1$ and price and delta for a short DIP with the same specifications and a barrier $B_{DIP} = 0.5$.	37
4.3	Price and delta for a long European Digital Call with $r = 0, \sigma = 0.1, B_{Digital} = 1.3$ and price and delta for a call spread with the same specifications first strike $k_1 = 1.28$ and second strike $k_2 = 1.32$ with notional = $\frac{1}{width}$ where width = $k_2 - k_1$. In the top figure, the option has maturity left $T=1$ and in the bottom figure, it has maturity left $T=0.1$	38
4.4	Payoff for a long European Digital Call with barrier $H = 1$ and payoff for a Digital Call spread with first strike $k_1 = 0.95$ and second strike $k_2 = 1$ with notional = $\frac{1}{k_2 - k_1} = 20$	38
4.5	Vega for a long European Digital Call with $r = 0, \sigma = 0.1, T = 0.1, B_{Digital} = 1.3$. The right figure is for an option with expiry $T=1$, the left is with expiry $T=0.1$	39
5.1	Deep hedging ratio δ_{70} under FNN across different times and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Black-Scholes, Heston and Rough-Bergomi. The figures showcase how δ_t evolves with spot price. Autocallable with $B_{AC} = 1.3$ and $B_{DIP} = 0.7$	43
5.2	In the top row, Realized paths are plotted for the three pricing models over scenario B. In the middle row, the hedging strategy is implemented respectively for each model under FNN and in the last row, the hedging strategy is implemented under LSTM.	44

5.3	Evolution of the deep hedging ratio δ_t under FNN across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 10$ under our three calibrated models.	46
5.4	In the top row, Realized paths are plotted for the three pricing models over scenario A. In the bottom row, the hedging strategy is implemented respectively for each model under LSTM and $\lambda = 10$	47
5.5	In the top row, Realized paths are plotted for the three pricing models over scenario B. In the bottom row, the hedging strategy is implemented respectively for each model under LSTM and $\lambda = 10$	47
A.1	Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Black-Scholes.	50
A.2	Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Heston.	51
A.3	Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Rough Bergomi.	52
A.4	Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Black-Scholes.	53
A.5	Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Heston.	54
A.6	Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Rough-Bergomi.	55
A.7	Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Black-Scholes.	56
A.8	Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Heston.	57
A.9	Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Rough Bergomi.	58
A.10	Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Black-Scholes.	59
A.11	Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Heston.	60
A.12	Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Rough-Bergomi.	61

List of Tables

1.1	Parameters fitted respectively for each plot in fig. 1.3 as of 2 September 2022 after market close.	18
5.1	Comparing indifference pricing for the three different models under FNN and LSTM against pricing the autocall under standard Monte-Carlo simulations.	42
5.2	Comparing indifference pricing this time for our three calibrated models under FNN and LSTM against pricing the autocall under standard Monte-Carlo simulations.	45

Introduction

In 2014, Gatheral et al. [15] postulate that "Volatility is rough". This observation motivated by Comte and Renault's work [9] on the Fractional Stochastic Volatility model would then give birth to a range of new volatility models attempting to capture this so-called "roughness". One of them is the rough Bergomi model as introduced by Bayer et al. [3] in 2016. This model has gained a lot of traction over the past few years as it is able to accurately capture implied volatility surfaces consistent with market data. Unfortunately, the initial simulation approach they suggest is slow which makes the model less tractable. It was later refined in Bennedsen et al. [4] who propose a method called the "Hybrid Scheme" which greatly improves the simulation time. Thanks to its improvement, we are now able to study the model's in-depth and investigate its applications to various products available within the financial space.

One particular product is of interest to us: the autocallable usually abbreviated as autocall. The autocall is one of the most popular structured products in the world and to this day offers no consensus on its valuation and its hedging. To this end, we are keen on investigating rough Bergomi against other famous model such as the widely known Black Scholes and against Heston, which is the model we will consider as our benchmark.

Unfortunately, as we just hinted, autocallables are very hard to value in practice. Effectively, its complex path-dependence generates issues when it comes to pricing and hedging. Essentially, the autocall is a product that contrary to other more classic (or certain exotic) options has the special feature of early redemption. In other words, under certain conditions, the instrument can terminate before it reaches maturity. This varying maturity feature has created many issues when using standard methods to value the product. With that in mind, we believe that straying from conventional valuation methods should lead to interesting results.

As a matter of fact, traditional valuation models are commonly used to price all kinds of products, usually doing so under the risk-neutral measure. However, in practice, markets are subject to various market imperfections such as liquidity constraints or transactions costs - which is not captured by those pricing models. These imperfections lead traders to manually adjust their hedges which does not always lead to the most optimal hedges. In this search for optimality, Halperin [17] propose a

where this model doesn't make any guesses or assumptions about the structure of the data it is given.

In 2018, Buehler et al. [7] publish a paper in which they show how training neural networks alongside indifference pricing can be used to price any portfolio of liabilities in a tractable model which is able to incorporate transaction costs. This ability to price any kind of liabilities is a breakthrough as we do not need some convoluted analytical pricing model to price complicated instruments, especially where there is no consensus on the pricing. What's more, this approach is able to encode the trader's risk preferences according to some optimal constraints relevant to our problem.

The goal of this thesis is to combine deep hedging alongside indifference pricing with a pricing model such as rough Bergomi to simulate stock prices and see empirically how the valuation and hedging behaves. We will specifically discuss in this thesis how to formalize this idea.

In this thesis, we shall first motivate the rough Bergomi model and provide the theoretical prerequisites in chapter 1. We will show how the model is constructed and calibrate it using real market data with SP 500 calls. In chapter 2, we detail our market setting rigorously and define

indifference pricing and its framework. In chapter 3, we present two types Artificial Neural Networks, Feedforward Neural Networks and a specific case of Recurrent Network, the Long Short Term Memory. These two types of Neural Networks will be used in our training. Concretely, we will compare our results under the two models and see which would be suitable to handle our task. Then, in chapter 4, we discuss Autocallables and the issues in their pricing and hedging. Finally, we present the results in chapter 5 under three different models: Black-Scholes, rough Bergomi and Heston.

Chapter 1

Theoretical prerequisites: the rough Bergomi model

1.1 The history of volatility

Volatility is in essence a key parameter to consider when modelling derivatives. But the term volatility is commonly abused and could mean a variety of things, from historical/realized to implied volatility, and from present to forward implied volatility, these notions are all different and one needs to apprehend which notion is discussed in a given context. One stylized fact observed from European options is that implied volatility, in the sense that it is the volatility extracted from market prices under a given model, is not constant over time and strikes. In equity markets, it is known as skew as implied volatility resembles a skew as shown in fig. 1.2. This notion gave birth to the volatility surface¹ as demonstrated in [10, Page 48]) in fig. 1.1.

In 1973, Black and Scholes [31] publish what will be known as the most famous model to price options. This model, although very tractable which helped tremendously increase its appeal, relies on assumptions that are not compatible with markets. One of them being that volatility is constant but as we saw, the implied volatility surface in markets destroys this assumption. To overcome this challenge, Dupire et al. [13] proposed to model the volatility as a function of time and spot price. This approach known as local volatility gained a lot of traction, proving easy to calibrate.

However, because the volatility is a deterministic function of time and spot price, this model completely fails to price options such as forward starting options and cliquets which are sensitive to forward volatility. This was discussed by Mazzon and Pascucci [25] and Wilmott [34] who show that in a local volatility model, forward skews are typically flat: therefore pricing cliquets or other options with exposure to forward skew will result in a typically lower price than those involving stochastic models and will almost surely misprice the option. The flat volatility coming from the fact that because of the dependency of the volatility on the underlying price, the spot moving higher has a higher probability i.e as time goes by, the volatilities and the skew will eventually go down hence leading to a flat forward volatility.

Therefore, in the attempt to capture the randomness of the volatility, stochastic volatility models have been popularized and are nowadays used both by practitioners and academics. In those models, volatility is neither constant nor deterministic but instead, it is a stochastic process. Among the many models, Heston [18] is popular for its tractability; providing a closed-form solution for the European call.

Unfortunately, Heston just like the other stochastic volatility models fail to replicate the observed skew and other volatility shapes observed in real markets. This disadvantage led academics and practitioners to look at volatility with a more granular approach. One model that followed was proposed by Bergomi [5]. In Bergomi's model, forward variance is a stochastic process and volatility skew was accurately captured for the case of SP 500 and other indices. Following this observation, Bayer et al. [3] have introduced rough Bergomi (rBergomi) a model that is driven by fractional

¹Also sometimes called cube due to its cubic shape as demonstrated in the Figure

Brownian motions and presented as an extension of the Bergomi model in the way the forward variance is captured.

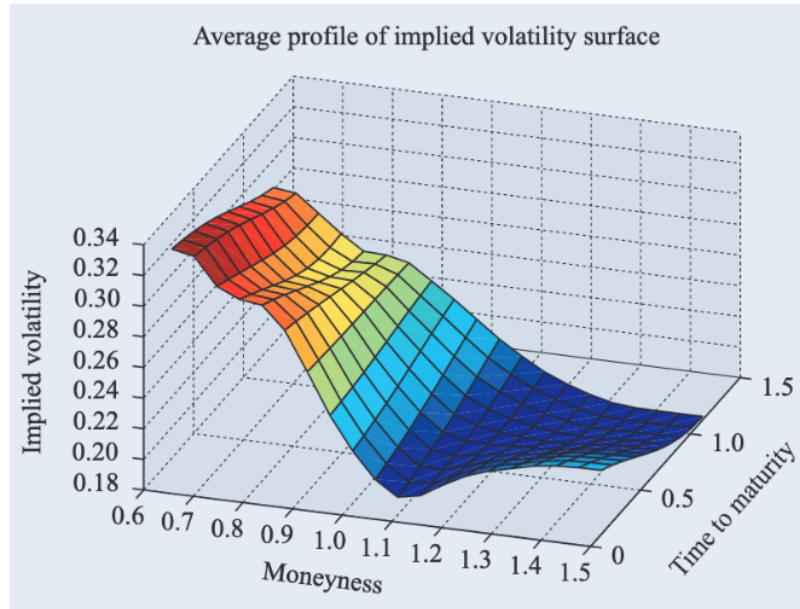


Figure 1.1: Average Implied volatility profile of SP500 options as March 1999 demonstrating this notion of volatility cube/surface

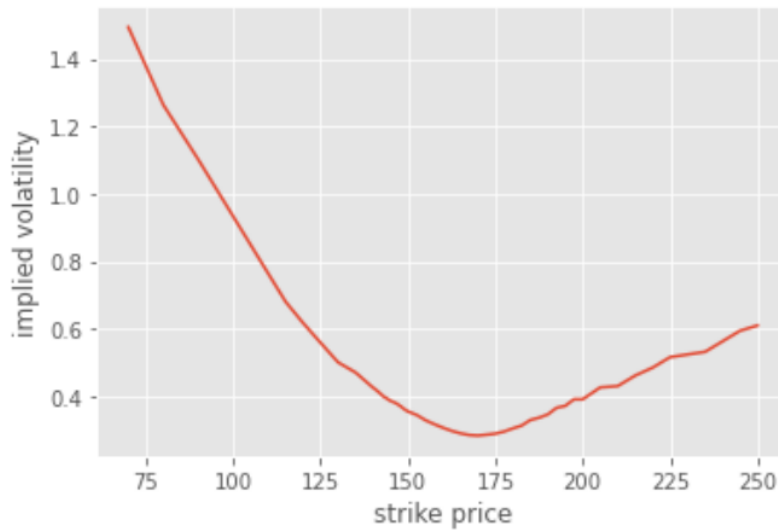


Figure 1.2: Implied volatility levels as of September 2, 2022 after market close, extracted from Apple call option prices, expiring September 23, 2022. As we can see, implied volatility decreases with strike and displays a skewed shape, hence the name "skew".

1.2 Fractional Brownian Motion

In 1968, Mandelbrot and Van Ness [24] introduce for the first time the concept of fractional Brownian motions. In their initial paper, they introduce this process as a sum of integrals of Brownian motions. Throughout this paper, we consider a probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t \geq 0}, \mathbb{P})$ where \mathcal{F} is the natural filtration generated by a standard Brownian motion.

Definition 1.2.1 (Fractional Brownian Motion). A fractional Brownian motion (fBm) is a Gaussian process $\{W_t^H, t \geq 0\}$ with mean zero and autocovariance function:

$$\mathbb{E}(W_t^H W_s^H) = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H}), \quad t, s \in \mathbb{R}. \quad (1.2.1)$$

where $H \in (0,1)$ is called the Hurst parameter or the Hurst index.

Remark 1.2.2. For $H = \frac{1}{2}$, the covariance function reads:

$$\mathbb{E}(W_t^{\frac{1}{2}} W_s^{\frac{1}{2}}) = \min(t, s)$$

which then simplifies to a *standard Brownian Motion*.

Proposition 1.2.3 (Stationary increments). *Consider the process $Y_t = W_{t+s}^H - W_s^H, t \geq 0$. A direct consequence of its definition as per eq. (1.2.1) is that Y and W^H have same covariance. Because both processes are Gaussian (by linearity), then they are equal in distribution and:*

$$Y_t = W_{t+s}^H - W_s^H \stackrel{d}{=} W_t^H.$$

Proposition 1.2.4 (Self-similarity). *Consider the process $Z_t = W_{at}^H, t \geq 0$ for some fixed $a > 0$. Using again the same arguments as in proposition 1.2.3, we obtain the equality in distribution:*

$$Z_t = W_{at}^H \stackrel{d}{=} a^H W_t^H$$

Proposition 1.2.5 (Correlation of the increments). *Let $(W_t^H)_{t \geq 0}$ be a fBm with Hurst index $H \in (0,1)$. Then its increments are:*

- (i) *Positively correlated for $H > \frac{1}{2}$.*
- (ii) *Independent for $H = \frac{1}{2}$ and simplifies to a standard Brownian motion.*
- (iii) *Negatively correlated for $H < \frac{1}{2}$.*

In particular for $H > \frac{1}{2}$, W^H exhibits long-range dependence:

$$\sum_{n=1}^{\infty} \mathbb{E}[W_1^H (W_{n+1}^H - W_n^H)] = \infty$$

Remark 1.2.6. As it is pointed out in [32], if $H < \frac{1}{2}$, the fBm is "counterpersistent" in the sense that it is likely to decrease in the future if it was increasing in the past and vice versa. On the other hand, if $H > \frac{1}{2}$, then the fBM is "persistent", it will likely keep trending in the same direction as previous values.

Proposition 1.2.7. *Let $(W_t^H)_{t \geq 0}$ be a fBm with Hurst index $H \in (0,1) \setminus \{\frac{1}{2}\}$. Then W^H is not a semi-martingale nor it is a Markov Process.*

Remark 1.2.8. A direct consequence from proposition 1.2.7 is that we cannot apply classic analytical pricing methods such as PDEs or Fourier transform to models involving fBm.

Theorem 1.2.9 (Kolmogorov continuity theorem and Hölder continuity). *Let $X : (X_t)_{t \geq 0}$ be a stochastic process. If there exist positive constants α, β, K such that:*

$$\mathbb{E}[|X_t - X_s|^\alpha] \leq K|t-s|^{1+\beta}$$

Then there exists a modification \tilde{X} of X that is a continuous process i.e a process $\tilde{X} : (\tilde{X}_t)_{t \geq 0}$ such that:

- \tilde{X} is sample-continuous
- $\mathbb{P}(X_t = \tilde{X}_t) = 1$

In particular, \tilde{X} is γ -Hölder continuous for every $\gamma \in \left(0, \frac{\beta}{\alpha}\right)$.

Corollary 1.2.10 (Continuity of fBm). *The fBm W^H admits continuous modifications and for any $\gamma \in (0, H)$ this modification is γ -Hölder continuous on each finite interval.*

1.3 Bergomi model

1.3.1 Motivations

Pricing instruments depending on forward volatility such as cliquets or forward starting options requires to have a model that captures it. A stylized fact from markets is that vanilla options tend to exhibit a non-constant-shaped implied volatility slope. In the equity market, implied volatility is characterized by a skewed shape for example. Hence vanilla options have exposure to this skew and this additional risk is usually exhibited when trading spreads. The implied volatility term-structure is hence used to trade the options under Black Scholes. But how does one get that for forward starting options? The same approach leads us to incorporate the market implied forward volatility term-structure and this risk is known as forward skew.

However, because the forward option market is usually less liquid than the vanilla one, it is not always possible to have a complete surface for all strikes at a given maturity T for strikes at date t_1 which complicates the pricing of those instruments under the Black Scholes framework.²

Bergomi has introduced in [5] a model that overcomes this problem by capturing forward variance.

1.3.2 Dynamics for the Bergomi model

Consider the forward variance given by dynamics:

$$d\xi^T(t) = \omega(T-t)\xi^T(t)dW_t^{\mathbb{Q}}$$

where W_t is a \mathbb{Q} -Brownian Motion and ω is in the form $\omega(\tau) = \omega e^{-k\tau}$. For simplicity, we will drop the \mathbb{Q} notation but unless specified otherwise, the Brownian Motion is associated to the \mathbb{Q} measure.

Applying Ito's lemma to $d\xi^T(t)$ and integrating between 0 and t then yields:

$$\xi^T(t) = \xi^T(0) \exp \left\{ \omega e^{-k(T-t)} X_t - \frac{1}{2} \omega^2 e^{-2k(T-t)} \mathbb{E}[X_t^2] \right\} \quad (1.3.1)$$

where $X = (X_t)_{t \geq 0}$ is an Ornstein-Uhlenbeck process of the form:

$$dX_t = -kX_t dW_t + dW_t, \quad X_0 = 0$$

which solution is well-known and given by:

$$X_t = \int_0^t e^{-k(T-s)} dW_s$$

. To compute X 's second moment, we use Ito's isometry which yields:

$$\begin{aligned} \mathbb{E}(X_t^2) &= \int_0^t e^{-2k(T-s)} ds \\ &= \frac{1 - e^{-2kt}}{2k} \end{aligned}$$

²There exist standardized cliquets for which one can obtain some market consensus data from which one can extrapolate a fairly accurate idea of where the market is pricing forward skew

Bergomi's one-factor model is then similar to the GBM but with a stochastic forward volatility. It is completely characterized by the following dynamics:

$$\begin{aligned} dS_t &= S_t(r - q)dt + S_t\sqrt{\xi_t(t)}dZ_t, & S_0 > 0, \\ d\xi_t(T) &= \omega(T - t)\xi_t(T)dW_t \\ d\langle Z, W \rangle_t &= \rho dt, \end{aligned} \quad (1.3.2)$$

In the N-factor Bergomi model, the forward variance reads:

$$\xi^T(t) = \xi^T(0) \mathcal{E} \left(\sum_{i=1}^N \eta_i \int_0^t e^{-\kappa_i(T-s)} dW_s^i \right) \quad (1.3.3)$$

where \mathcal{E} is the stochastic exponential and is given by $\mathcal{E}(X_t) = \exp\left(X_t - X_0 - \frac{1}{2}\langle X \rangle_t\right)$ where X is assumed to be a continuous semimartingale.

1.4 Rough Bergomi model

In 2014, Gatheral et al. [15, Section 3.1, page 14] observed that the logarithm of the realized variance behaves similarly to a fBm. To capture this property in a model, Bayer et al. [3] have introduced rough Bergomi as an extension of the Bergomi model in the way the forward variance is captured.

We will go through the derivation of their model in this section.

1.4.1 Deriving the rough Bergomi model under the \mathbb{P} measure

First, consider the Mandelbrot and Van Ness representation as we have introduced in section 1.2:

$$W_t^H = C_H \left(\int_{-\infty}^t \frac{dW_s^{\mathbb{P}}}{(t-s)^{-\alpha}} - \int_{-\infty}^0 \frac{dW_s^{\mathbb{P}}}{(s)^{-\alpha}} \right) \quad (1.4.1)$$

where $\alpha = H - \frac{1}{2}$ and where we choose $C_H = \sqrt{\frac{2H\Gamma(3/2-H)}{\Gamma(H+1/2)\Gamma(2-2H)}}$ such that it leads to the same covariance we used to define a fBm in eq. (1.2.1).

Gatheral et al. [15, Section 3.1, page 14] found that increments of the logarithm realized volatility were proportional to the increments of fBm:

$$\log \sigma_{t+\Delta} - \log \sigma_t = \nu(W_{t+\Delta}^H - W_t^H), \nu > 0$$

This finding leads us to study the log variance difference:

$$\begin{aligned} \log v_u - \log v_t &= 2\nu C_H \left(\int_{-\infty}^u \frac{dW_s^{\mathbb{P}}}{(u-s)^{-\alpha}} - \int_{-\infty}^t \frac{dW_s^{\mathbb{P}}}{(t-s)^{-\alpha}} \right) \\ &= 2\nu C_H \left(\int_t^u \frac{dW_s^{\mathbb{P}}}{(u-s)^{-\alpha}} + \int_{-\infty}^t \left[\frac{1}{(u-s)^{-\alpha}} - \frac{1}{(t-s)^{-\alpha}} \right] dW_s^{\mathbb{P}} \right) \\ &= 2\nu C_H (M_t(u) + Z_t(u)) \end{aligned}$$

Additionally, we introduce $\tilde{W}_t^{\mathbb{P}}(u)$ known as a Volterra process:

$$\tilde{W}_t^{\mathbb{P}}(u) := \sqrt{2H} \int_t^u \frac{dW_s^{\mathbb{P}}}{(u-s)^{-\alpha}} \quad (1.4.2)$$

In the above, $Z_t(u)$ is \mathcal{F}_t -measurable, $M_t(u)$ and $\tilde{W}_t^{\mathbb{P}}(u)$ are independent of \mathcal{F}_t . Furthermore, $M_t(u) \sim \mathcal{N}\left(0, \frac{(u-t)^{2H}}{2H}\right)$ and $\tilde{W}_t^{\mathbb{P}}(u) \sim \mathcal{N}\left(0, (u-t)^{2H}\right)$.

Then introducing $\eta = \frac{2\nu C_H}{\sqrt{2H}}$ we can write the realized variance as:

$$v_u = v_t \exp \left\{ \eta \tilde{W}_t^{\mathbb{P}}(u) + 2\nu C_H Z_t(u) \right\}$$

Using well-known properties of normal and lognormal random variables, we may write:

$$\begin{aligned} \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] &= v_t \exp \left\{ \frac{1}{2} \eta^2 \mathbb{E}[|\tilde{W}_t|^2] + 2\nu C_H Z_t(u) \right\} \\ &= v_t \exp \left\{ \frac{1}{2} \eta^2 (u-t)^{2H} + 2\nu C_H Z_t(u) \right\} \end{aligned}$$

We can then rewrite the realized variance in terms of the stochastic exponential:

$$v_u = \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E} \left(\eta \tilde{W}_t^{\mathbb{P}}(u) \right) \quad (1.4.3)$$

Then under the physical measure \mathbb{P} , the model dynamics are given by:

$$\begin{aligned} dS_u &= S_u(\mu_u du + \sqrt{v_u} dZ_u^{\mathbb{P}}), \quad S_0 > 0, \\ v_u &= v_t \exp \left\{ \eta \tilde{W}_t^{\mathbb{P}}(u) + 2\nu C_H Z_t(u) \right\} \\ d\langle Z^{\mathbb{P}}, W^{\mathbb{P}} \rangle_t &= \rho dt, \end{aligned} \quad (1.4.4)$$

1.4.2 Pricing under measure \mathbb{Q}

Having modeled eq. (1.4.4) under the real-world measure \mathbb{P} , we wish now to derive it under the Equivalent Martingale Measure (EMM) \mathbb{Q} , the measure under which a derivative price is equal to the expected value of discounted future cash flows. A direct consequence is that under this equivalent measure \mathbb{Q} , the discounted asset price is a martingale. Setting interest rates to 0 without loss of generality:

$$\frac{dS_u}{S_u} = \sqrt{v_u} dZ_u^{\mathbb{Q}}.$$

where $dZ_u^{\mathbb{Q}}$ is obtained by Girsanov's theorem:

$$dZ_u^{\mathbb{Q}} = dZ_u^{\mathbb{P}} + \frac{\mu_u}{\sqrt{v_u}} du, \quad t \leq u \leq T. \quad (1.4.5)$$

Recalling the process we introduced in eq. (1.4.2) and the model we obtained under the \mathbb{P} measure in eq. (1.4.4), we can rewrite $W^{\mathbb{P}}$:

$$dW_u^{\mathbb{P}} = \rho dZ_u^{\mathbb{P}} + \bar{\rho} d\bar{Z}_u^{\mathbb{P}} \quad (1.4.6)$$

where $(Z^{\mathbb{P}}, \bar{Z}^{\mathbb{P}})$ are two independent standard Brownian motions and the choice $\bar{\rho} = 1 - \rho^2$ ensures independence of the two previous Brownian motions. \bar{Z} is now the last process for which we haven't got an expression under the measure \mathbb{Q} . A prompt change of measure yields:

$$d\bar{Z}_u^{\mathbb{Q}} = d\bar{Z}_u^{\mathbb{P}} + \gamma_u du \quad (1.4.7)$$

where γ is a suitable adapted process referred to as the market price of volatility risk.

Rewriting eq. (1.4.6) using eq. (1.4.5) and eq. (1.4.7) reads:

$$dW_u^{\mathbb{Q}} = dW_u^{\mathbb{P}} + (\rho\mu_u/\sqrt{v_u} + \bar{\rho}\gamma_u) du \quad (1.4.8)$$

which can be rewritten as:

$$dW_s^{\mathbb{P}} = dW_s^{\mathbb{Q}} + \lambda_s ds$$

Now, that we have obtained an expression of $W^{\mathbb{Q}}$, we can rewrite the realized variance in eq. (1.4.3) under the \mathbb{Q} measure:

$$\begin{aligned}
v_u &= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E} \left(\eta \tilde{W}_t^{\mathbb{P}}(u) \right) \\
&= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E} \left(\eta \sqrt{2H} \int_t^u \frac{dW_s^{\mathbb{P}}}{(u-s)^\gamma} \right) \\
&= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \exp \left(\eta \sqrt{2H} \int_t^u \frac{dW_s^{\mathbb{P}}}{(u-s)^\gamma} - \frac{\eta^2}{2} (u-t)^{2H} \right) \\
&= \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \mathcal{E} \left(\eta \tilde{W}_t^{\mathbb{Q}}(u) \right) \exp \left(\int_t^u \frac{\lambda_s}{(u-s)^\gamma} ds \right) \\
&= \xi_t(u) \mathcal{E} \left(\eta \tilde{W}_t^{\mathbb{Q}}(u) \right)
\end{aligned}$$

where the forward variance

$$\xi_t(u) = \mathbb{E}^{\mathbb{P}}[v_u | \mathcal{F}_t] \exp \left(\int_t^u \frac{\lambda_s}{(u-s)^\gamma} ds \right)$$

depends on the past values of the Brownian motion up to time t unlike the one in the Bergomi model given in eq. (1.3.1). This specification in the rough Bergomi model enables in theory to correctly capture the evolution of the variance swap curve.

1.4.3 Dynamics of the rough Bergomi model under the \mathbb{Q} measure

To summarize, the rough Bergomi model is completely characterized by:

$$S_t = S_0 \exp \left\{ \int_0^t \sqrt{v_u} dZ_u^{\mathbb{Q}} - \frac{1}{2} \int_0^t v_u du \right\} \quad (1.4.9)$$

$$v_t = \xi_u(0) \exp \left\{ \eta \tilde{W}_t^{\mathbb{Q}} - \frac{\eta^2}{2} t^{2\alpha+1} \right\} \quad (1.4.10)$$

$$\tilde{W}_t^{\mathbb{Q}} = \sqrt{2\alpha+1} \int_0^t (t-s)^\alpha dW_s^{\mathbb{Q}}, \quad (1.4.11)$$

$$dZ_t^{\mathbb{Q}} = \rho dW_t^{\mathbb{Q}} + \sqrt{1-\rho^2} d\tilde{W}_t^{\mathbb{Q}} \quad (1.4.12)$$

where $\alpha = H - \frac{1}{2} \in (-\frac{1}{2}, \frac{1}{2}) \setminus \{0\}$, H being the Hurst exponent of the fBm as per definition 1.2.1.

Bayer et al. [3, page 18-21] have demonstrated that this model is able to fit implied volatility skews very accurately, which is substantial progress compared to traditional stochastic volatility models. As a matter of fact, it is well known that models used in the pricing of exotic options like Heston or SABR struggle to replicate observed volatility surfaces implied from european options.

However, tractability can be a main issue for this model as methods like PDEs or Fourier transforms cannot be applied as we mentioned in remark 1.2.8. Effectively, the initial approach proposed by Bayer et al. [3, Section 4, page 14-16] to simulate the Volterra process $\tilde{W}_t^{\mathbb{Q}}$ is based on a Cholesky decomposition and is of order $\mathcal{O}(n^3)$. This approach can hence prove to be numerically slow. The next section is dedicated to a new approach called the hybrid scheme which aims to simulate the Volterra process with another approach that is numerically faster.

1.5 Simulating the Volterra Process: the hybrid scheme

Recently, Bennedsen et al. [4] have proposed the hybrid scheme and its turbocharged version implementation [26] is as of today the fastest technique to simulate the Volterra process $\tilde{W}_t^{\mathbb{Q}}$. It is of order $\mathcal{O}(n \log n)$ which is a substantial progress from the Cholesky decomposition method of order $\mathcal{O}(n^3)$. To simulate the asset, we will hence use this last implementation.³

We briefly sketch the fundamental principles of this scheme and the reader may refer to Bennedsen et al. [4] for further details.

³which can be found here: https://github.com/ryanmccrickerd/rough_bergomi

We consider a Brownian semistationary process (BSS), introduced in [2], of the form:

$$X_t = \int_{-\infty}^t g(t-s)\sigma_s dW_s \quad (1.5.1)$$

where W is a standard Brownian Motion, $g : (0, \infty) \rightarrow [0, \infty[$ and $\sigma = (\sigma_t)_{t \in \mathbb{R}}$ is an $(\mathcal{F}_t)_{t \in \mathbb{R}}$ -predictable process with locally bounded trajectories.

To insure eq. (1.5.1) is well-defined, we assume that g and σ are measurable functions that are square-integrable.

When for some $\alpha \in (-\frac{1}{2}, \frac{1}{2}) \setminus \{0\}$, g behaves like a power-law near zero, then the trajectories of X behave like the trajectories of a Brownian Motion with Hurst index $H = \alpha + \frac{1}{2} \in (0, 1) \setminus \{\frac{1}{2}\}$.

The following assumptions are made on the function g :

- (i) For some $\alpha \in (-\frac{1}{2}, \frac{1}{2}) \setminus \{0\}$,

$$g(x) = x^\alpha L_g(x), x \in (0, 1],$$

where $L_g : (0, 1] \rightarrow [0, \infty)$ is continuously differentiable slowly varying at 0 and bounded away from 0. Moreover, there exists a constant $C > 0$ such that the derivative L'_g satisfies:

$$|L_g(x)| \leq C(1+x^{-1}), x \in (0, 1],$$

- (ii) g is continuously differentiable on $(0, \infty)$. Its derivative is monotonic and is square integrable between $(1, \infty)$.

- (iii) For some $\beta \in (-\infty, -\frac{1}{2})$,

$$g(x) = \mathcal{O}(x^\beta), \quad x \rightarrow \infty$$

These conditions ensure that g is square-integrable.

We consider now a truncated version of eq. (1.5.1) with the integral starting at 0 as we work with processes in which the time grid starts at 0. Processes of this form are called Truncated Brownian semistationary (TBSS) process:

$$Y_t = \int_0^t g(t-s)\sigma_s dW_s \quad (1.5.2)$$

where g, σ, W are defined as previously.

We define the hybrid scheme to discretize Y_t in its integral representation. Consider the grid $\mathcal{G}_t^n := \{0, \frac{1}{n}, \dots, \frac{[nt]}{n}\}$. Then if the volatility process σ is kept constant in each discretization cell, we can write:

$$Y_t = \sum_{k=1}^{[nt]} \int_{t-\frac{k}{n}}^{t-\frac{k-1}{n}} g(t-s)\sigma_s dW_s \simeq \sum_{k=1}^{[nt]} \sigma_{t-\frac{k}{n}} \int_{t-\frac{k}{n}}^{t-\frac{k-1}{n}} g(t-s) dW_s := Y_t^n$$

- If k is small in some sense, for instance define some $\kappa > 0$ such that $k < \kappa$ then we can approximate g as:

$$g(t-s) \simeq (t-s)^\alpha L_g\left(\frac{k}{n}\right), \quad t-s \in \left[\frac{k-1}{n}, \frac{k}{n}\right] \setminus \{0\}$$

- If k is large then we can approximate g :

$$g(t-s) \simeq g\left(\frac{b_k}{n}\right), \quad t-s \in \left[\frac{k-1}{n}, \frac{k}{n}\right]$$

In particular, the discretization MSE is minimized when [4, Proposition 2.8, page 942]:

$$b_k^* = \left(\frac{k^{\alpha+1} - (k-1)^{\alpha+1}}{\alpha+1} \right)^{\frac{1}{\alpha}}, \quad k \geq \kappa+1$$

Finally splitting Y_t^n between the values up to κ and larger yields:

$$Y_t^n := \check{Y}_t^n + \hat{Y}_t^n \tag{1.5.3}$$

where:

$$\begin{aligned} \check{Y}_t^n &:= \sum_{k=1}^{\min(\lfloor nt \rfloor, \kappa)} L_g\left(\frac{k}{n}\right) \sigma_{t-\frac{k}{n}} \int_{t-\frac{k}{n}}^{t-\frac{k-1}{n}} (t-s)^\alpha dW_s, \\ \hat{Y}_t^n &:= \sum_{k=\kappa+1}^{\lfloor nt \rfloor} g\left(\frac{b_k}{n}\right) \sigma_{t-\frac{k}{n}} W_{t-\frac{k-1}{n}} - W_{t-\frac{k}{n}}. \end{aligned}$$

We have now derived the hybrid scheme developed by Bennedsen et al.

To see how this is connected to fractional Brownian motions and ultimately to the rough Bergomi model, consider now the Volterra process: $W_t^\alpha := \sqrt{2\alpha+1} \int_0^t (t-u)^\alpha dW_u^1$.

To simulate this process, we can use the first-order simulation ($\kappa = 1$) of the hybrid-scheme we just outlined. Indeed, we can approximate Y^α as:

$$Y_{\frac{t}{n}}^\alpha \simeq \tilde{Y}_{\frac{t}{n}}^\alpha := \sqrt{2\alpha+1} \left(\int_{\frac{t-1}{n}}^{\frac{t}{n}} \left(\frac{t}{n} - s\right)^\alpha dW_s + \sum_{k=2}^t \left(\frac{b_k}{n}\right)^\alpha W_{t-\frac{k-1}{n}} - W_{t-\frac{k}{n}} \right)$$

which is just Y_t^n in eq. (1.5.3) with: $L_g(\cdot) = 1, \sigma(\cdot) = 1, g(u) = u^\alpha, u > 0$. In particular, $\alpha = H - \frac{1}{2} \in (-\frac{1}{2}, \frac{1}{2}) \setminus \{0\}$.

Hence, we just showed that Y^α can be approximated as a TBSS which we can simulate under the hybrid scheme and achieve a performance of order $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^3)$.

1.5.1 Calibration

One strength of the rough Bergomi model is the few number of parameters one needs to calibrate to the market. As a matter of fact, there are 4 free parameters which are H, η, ρ, ξ . In particular, ξ being the forward variance is theoretically observable in the market which reduces the amount of parameters to 3. Indeed, as there are actively traded variance swaps, one would observe the market variance swap curve and calibrate accordingly. In practice however, finding variance swap curve data is not as easy and for this reason, so we also need to estimate ξ . For our estimation, we assume that the forward variance curve is constant i.e $\xi_t(0) = \xi(0)$.

As we can see on fig. 1.3, the rough Bergomi model accurately fits the IV profile of SP 500 calls for different maturities. We have summarized the parameters fitted for each figure in table 1.1.

We also note that the optimal parameters for $H/\alpha, \rho$ and η seem to roughly stay in the same range even for increasing maturities. In [3, Section 5.2, page 18-20], Bayer et al. find that fitting S&P options as of February 4, 2010 with parameters $H = 0.05, \rho = -0.9, \xi = 2.3$ provides an accurate fit for expiries ranging from T=0.041 to 2.88 (years). These parameters are quite close to ours which provides some intuition regarding the parameter calibration. In particular, we note that that ξ fixes the initial forward variance and that H/α controls the decay of the term structure of the ATM volatility skew.

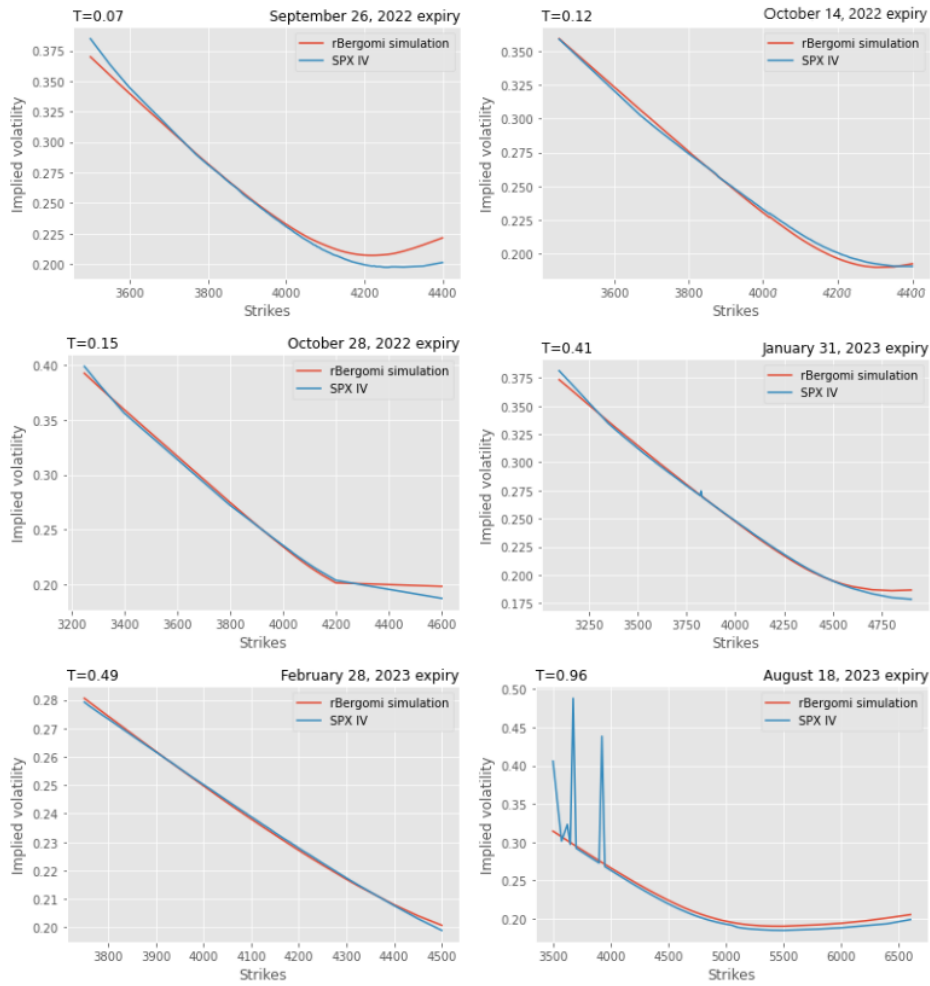


Figure 1.3: S&P 500 Implied volatility taken from market call prices (mid) as of 2 September 2022 after market close. Parameters are summarized in table 1.1.

Expiry	T	H/ α	ρ	ξ	η
September 26, 2022	0.06	0.09/-0.41	-0.99	0.090	2.10
October 14, 2022	0.11	0.10/-0.40	-0.99	0.085	2.08
October 28, 2022	0.15	0.10/-0.40	-0.99	0.090	2.13
January 31, 2023	0.41	0.10/-0.40	-0.99	0.105	2.11
February 28, 2023	0.49	0.10/-0.40	-0.97	0.104	2.11
August 18, 2023	0.96	0.10/-0.40	-0.95	0.130	2.08

Table 1.1: Parameters fitted respectively for each plot in fig. 1.3 as of 2 September 2022 after market close.

Chapter 2

Problem setting and utility indifference pricing

Pricing instruments is important and hedging them is probably as much. Usually, analytical models are developed in a way that they rely on risk factors that influence the price of an instrument. Sometimes, these models need to be calibrated to the market and one has to reverse engineer said implicit risk factors such as implied volatility even though they are not state variables in the model. In section 1.1, we have discussed the many limitations that models offer when pricing financial instruments.

In the attempt to overcome these limitations, a new approach gained traction which states that the price at which a contingent claim should be sold is equal to the cheapest hedge that the trader will implement in order to hedge the claim. Formalized, this concept is called *indifference pricing*, it is the price the hedger of the claim needs to charge to be indifferent between taking the position and not taking it. In this chapter, we shall formalize this idea.

2.1 Market setting in discrete time

First, let us define our market setting. We consider a discrete-time market with horizon $T \in \mathbb{N}$ in which we assume that the market is made up of $d \in \mathbb{N}$ risky assets, which prices are denoted by $S := (S_t)_{t=0,1,\dots,T}$ with $S_t = (S_{t,1}, \dots, S_{t,d})$ and assume that they form an adapted, non-negative stochastic process on a filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ although, we are not required to make assumptions on the equivalent martingale measure. Our portfolio of claims that we aim to hedge is denoted by a random variable Z which is \mathcal{F}_T measurable. Z , is referred to in the finance literature as the contingent claim and in our setting, it represents the autocallable that we aim to hedge with this thesis. It is important to note that this approach is model-free and it will not require any pricing model, or any sensitivity-approach such as computing greeks.

To hedge the claim Z , we trade in S using an \mathbb{R}^d -valued \mathcal{F} -adapted stochastic process $\delta := (\delta_t)_{t=0,1,\dots,T-1}$ with $\delta_t = (\delta_{t,1}, \dots, \delta_{t,d})$.

Here δ_t^i is the agent's position in asset i at time t . Because the hedging strategy is self-financed, we may need some cash at inception $p_0 \in \mathbb{R}$ which can be interpreted as the price of the claim Z , sold at inception. Additionally, we assume that for any trade size $x \in \mathbb{R}$, trading incurs proportional transaction cost $k_i|x|S_t^i$ for some constant $k_i \in \mathbb{R}_+$. Thus, the strategy's terminal wealth $V_T(\delta)$ is given by:

$$V_T(\delta) = (\delta \cdot S)_T - C_T(\delta)$$

where

$$(\delta \cdot S)_T = \sum_{t=1}^T \delta_t(S_t - S_{t-1})$$

represents the replicating strategy's P&L and:

$$C_T(\delta) = \sum_{i=1}^d k_i \left(|\delta_{0,i}| S_{0,i} + \sum_{t=2}^T |\delta_{t-1,i} - \delta_{t-2,i}| S_{t-1,i} + |\delta_{T,i}| S_{T,i} \right)$$

is the total cost incurred at the liquidation of the self-financing strategy δ . It is important to note that Z is not necessarily a function of the terminal asset, S_T but can be a function of the entire path. Hence, this method provides a systematic approach to valuing and hedging complicated payoffs such as autocallables without having to delve into complicated pricing models.

Finally, our terminal P&L is given by:

$$P\&L(Z, p_0, \delta) := p_0 + V_T - Z$$

The agents seek to optimize their PL with respect to δ according to their risk preferences. Define a loss function $\ell: \mathbb{R} \rightarrow \mathbb{R}$ that models the agents risk preferences, their goal is to ultimately minimize their loss at maturity:

$$\inf_{\delta \in \mathcal{H}} \mathbb{E}[\ell(-Z + p_0 + (\delta \cdot S)_T - C_T(\delta))] \quad (2.1.1)$$

We shall discuss which loss function will be used to model risk preferences in section 2.4. Furthermore, we will discuss in greater detail how we wish to tackle this minimizing problem with Neural Networks that we will present in chapter 3.

2.2 Convex Risk measure

In essence, in a complete and arbitrage-free market, for our claim Z , there exists a unique replication strategy δ and a fair price $p_0 \in \mathbb{R}$ such that $-Z + p_0 + (\delta \cdot S)_T - C_T(\delta) = 0$ holds \mathbb{P} -a.s. However in practice, transaction costs and other market frictions are inherent to markets which nullifies the previous assertion. To overcome this issue, we need to define an acceptable price such that the overall position becomes acceptable in light of the various costs and constraints. This optimization problem is formulated through the use of convex risk measures which we recall their definition next.

Definition 2.2.1 (Convex Risk Measure). A convex risk measure $\rho: \mathbb{X} \rightarrow \mathbb{R}$ which satisfies the following for each $X, Y \in \mathbb{X}$:

- (i) **Decreasing Monotonicity:** If $X \leq Y$, then $\rho(X) \geq \rho(Y)$.
A portfolio with a better position has less risk.
- (ii) **Convexity:** $\rho(\lambda X + (1 - \lambda)Y) \leq \lambda \rho(X) + (1 - \lambda)\rho(Y)$ for $0 \leq \lambda \leq 1$.
The risk of two Portfolios added together is less or equal to two separate portfolios risk added together (Diversification).
- (iii) **Translation invariance:** If $m \in \mathbb{R}$, then $\rho(X + m) = \rho(X) - m$.
Adding cash to a Portfolio will reduce its risk by as much.

2.3 Indifference Pricing

Let $\rho: \mathbb{X} \rightarrow \mathbb{R}$ be a convex risk measure and let \mathcal{H} be the set of restricted trading strategies such as liquidity, asset availability or trading restrictions, we write the optimization problem:

$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta)) \quad (2.3.1)$$

The following proposition found in [7, Section 3, Proposition 3.2, p.7] justifies the use of convex measures.

Proposition 2.3.1. π is monotone decreasing and cash-invariant. If C_T and \mathcal{H} are convex, then π is convex and is hence a convex risk measure.

We define an optimal hedging strategy as a minimizer $\delta \in \mathcal{H}$ of eq. (2.3.1) As Buehler et al. [7] have shown, indifference pricing can be used alongside deep hedging.

Defining the indifference price $p(Z)$ as the hedger of the claim Z needs to charge to be indifferent between position $-Z$ and not taking the position (i.e 0). The indifference price $p(Z)$ is then solution to $\pi(-Z + p_0) = \pi(0)$ and by translation invariance, $p_0 = p(Z)$. In short, the indifference price can be expressed as the difference between the optimization problem with claim Z and claimless, i.e:

$$p(Z) = \pi(-Z) - \pi(0) \quad (2.3.2)$$

Remark 2.3.2. Under no transaction costs and no trading constraints, p coincides with the price of a replicating portfolio (if it exists).

2.4 Exponential Utility Indifference Pricing

As we have seen in eq. (2.1.1), we wish to find some loss function, $\ell: \mathbb{R} \rightarrow \mathbb{R}$ that encodes the agents risk preferences. A number of candidates can be used such as the quadratic loss $\ell(x) = x^2$ or absolute loss $\ell(x) = |x|$. Utility functions are a popular set of functions to tackle this problem as they provide a parameter to tune risk preferences accordingly, they have been covered extensively in the literature (e.g [33], [11]). Suppose our loss function to be modeled according to some utility function $U: \mathbb{R} \rightarrow \mathbb{R}$ such that $\ell(X) = -U(x)$. U is assumed to be strictly concave and increasing so that $-U$ becomes strictly convex and decreasing.

Exponential utility is a special case within utility functions that renders indifference pricing convenient to use as we will show in this section. The exponential utility function is given by:

$$U_\lambda(x) := -\exp(-\lambda x), \quad x \in \mathbb{R} \quad (2.4.1)$$

where $\lambda > 0$ is the risk-aversion parameter. Then denoting $q := q(Z) \in \mathbb{R}$ the indifferent price of the claim Z under the exponential utility, q must solve:

$$\begin{aligned} \sup_{\delta \in \mathcal{H}} \mathbb{E}[U((\delta \cdot S)_T - C_T(\delta))] &= \sup_{\delta \in \mathcal{H}} \mathbb{E}[U(q - Z + (\delta \cdot S)_T - C_T(\delta))] \\ \sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda((\delta \cdot S)_T - C_T(\delta)))] &= \sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda(q - Z + (\delta \cdot S)_T - C_T(\delta)))] \\ \exp(\lambda q) &= \frac{\sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda(-Z + (\delta \cdot S)_T - C_T(\delta)))]}{\sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda((\delta \cdot S)_T - C_T(\delta)))]} \\ q &= \frac{1}{\lambda} \log \left(\frac{\sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda(-Z + (\delta \cdot S)_T - C_T(\delta)))]}{\sup_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda((\delta \cdot S)_T - C_T(\delta)))]} \right) \end{aligned} \quad (2.4.2)$$

The purpose of this section is to show that q coincides with p as defined in eq. (2.3.2). As we have obtained an explicit expression for q , consider the entropic risk measure ρ :

$$\rho(X) = \frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda X)]$$

Then going from eq. (2.3.2):

$$\begin{aligned}
p(Z) &= \pi(-Z) - \pi(0) \\
&= \inf_{\delta \in \mathcal{H}} \rho(-Z + (\delta \cdot S)_T) - \inf_{\delta \in \mathcal{H}} \rho((\delta \cdot S)_T) \\
&= \frac{1}{\lambda} \log \left(\frac{\inf_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda(-Z + (\delta \cdot S)_T))]}{\inf_{\delta \in \mathcal{H}} \mathbb{E}[\exp(-\lambda(\delta \cdot S)_T)]} \right) \\
&= \frac{1}{\lambda} \log \left(\frac{\sup_{\delta \in \mathcal{H}} \mathbb{E}[U(-Z + (\delta \cdot S)_T)]}{\sup_{\delta \in \mathcal{H}} \mathbb{E}[U(\delta \cdot S)_T]} \right) = q
\end{aligned}$$

We have now proven that p coincides with q .

This last equation tells us that to find the indifference price, we need to solve the hedging problem with utility function U with claim Z and claimless. Hence, in the case of deep learning, we would need to train the network twice with and without claim Z . However, in practice, we find that the optimum strategy without claim Z reduces to 0 i.e not hedging the absence of claim, which seems intuitive. This entails that concretely, we only need to train the network once.

In the context of exponential utility, the indifference price is then given by the estimator:

$$\hat{p} = \frac{1}{\lambda} \log \left(\frac{\frac{1}{N} \sum_{i=1}^N U_{\lambda}(-Z + (\delta \cdot S)_T)}{\frac{1}{N} \sum_{i=1}^N U_{\lambda}(\delta \cdot S)_T} \right) \quad (2.4.3)$$

Chapter 3

Artificial networks: FNN and LSTM

3.1 Introduction

Deep Learning is a branch of machine learning that uses artificial neural networks (ANN) to solve some optimization problem. ANNs have gained a lot of traction in the past decades, namely due to the increasing computational power at hand. Despite the numerous research papers that were published and the progress that was made over the last decades, neural networks were still not as much of a popular research topic¹ in the 90s. The late 2000s and early 2010s saw important breakthrough in the fields of speech and image recognition with Krizhevsky et al. [22] and Hinton et al. [19] that vastly popularized neural networks as a cross-industry application.

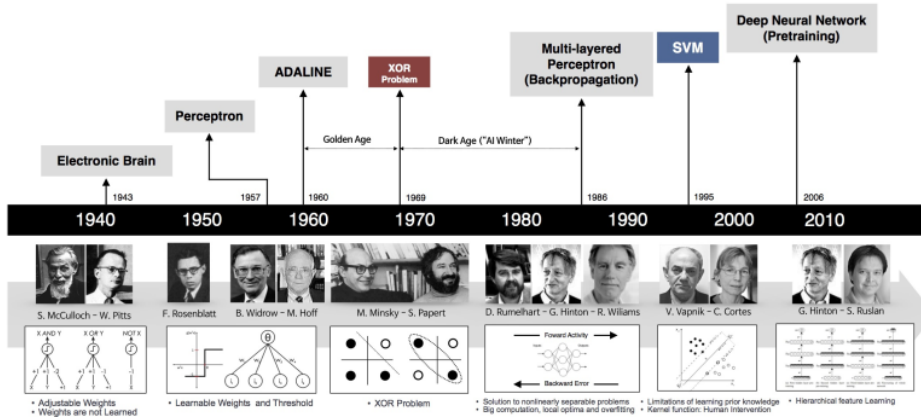


Figure 3.1: The history of Neural Networks over the last decades, source: http://beamlab.org/deeplearning/2017/02/23/deep_learning_101_part1.html

Informally, the idea behind deep learning - in this case feedforward neural networks - is to design a function $f = (f_1, \dots, f_O): \mathbb{R}^I \rightarrow \mathbb{R}^O$ that converts $I \in \mathbb{N}$ inputs, into $O \in \mathbb{N}$ outputs in some optimal way. This is known as the optimization problem and in the case of hedging, we call it the hedging problem.

The optimization problem is quantified through a loss function ℓ in the sense that we optimize the loss function according to some conditions relevant to our problem.

¹at least, not as popular as now

A famous example is the regression problem where given $\mathbf{x} = (x_i^n), n \in \mathbb{N}, i \in I$ samples of our input variables - also known as "features" - we aim to predict $\mathbf{y} = (y^n), n \in \mathbb{N}$ "labels", hence we try to match the output of \mathbf{f} such that it is as close as possible from labels \mathbf{y} .

Usually, in this regression problem, we choose squared loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$ and we seek to minimize the average loss, known as empirical risk:

$$\mathcal{L}(\mathbf{f}) := \frac{1}{N} \sum_{i=1}^N \ell(\mathbf{f}(\mathbf{x}^i), \mathbf{y}^i)$$

In this chapter, we shall formalize these ideas and we will present two models involving neural networks, Feedforward neural networks and Long Short Term memory.

3.2 Feedforward neural networks (FFN)

This section is inspired from Deep Learning lecture notes [27]. We present in this section the building blocks of Feedforward neural networks.

3.2.1 Architecture

Definition 3.2.1 (Feedforward neural network). A function $\mathbf{f} : \mathbb{R}^I \rightarrow \mathbb{R}^O$ is a feedforward neural network (FNN) with $r - 1 \in \{0, 1, \dots\}$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i -th hidden layer for any $i = 1, \dots, r - 1$, and activation functions $\sigma_i : \mathbb{R}^{d_i} \rightarrow \mathbb{R}^{d_i}, i = 1, \dots, r$, where $d_r := O$, if

$$\mathbf{f} = \sigma_r \circ \mathbf{L}_r \circ \dots \circ \sigma_1 \circ \mathbf{L}_1,$$

where $\mathbf{L}_i : \mathbb{R}^{d_{i-1}} \rightarrow \mathbb{R}^{d_i}$, for any $i = 1, \dots, r$, is an affine function

$$\mathbf{L}_i(\mathbf{x}) := W^i \mathbf{x} + \mathbf{b}^i, \quad \mathbf{x} \in \mathbb{R}^{d_{i-1}},$$

parameterised by weight matrix $W^i = [W_{j,k}^i]_{j=1, \dots, d_i, k=1, \dots, d_{i-1}} \in \mathbb{R}^{d_i \times d_{i-1}}$ and bias vector $\mathbf{b}^i = (b_1^i, \dots, b_{d_i}^i) \in \mathbb{R}^{d_i}$, with $d_0 := I$. We shall denote the class of such functions \mathbf{f} by

$$\mathcal{N}_r(I, d_1, \dots, d_{r-1}, O; \sigma_1, \dots, \sigma_r).$$

If $\sigma_i(\mathbf{x}) = (g(x_1), \dots, g(x_{d_i})), \mathbf{x} = (x_1, \dots, x_{d_i}) \in \mathbb{R}^{d_i}$, for some $g : \mathbb{R} \rightarrow \mathbb{R}$, we write g in place of σ_i .

The architecture of the FNN is completely characterized by:

- the hyperparameters: r, d_1, \dots, d_r
- the (actual) parameters: weights W^1, \dots, W^r and biases $\mathbf{b}^1, \dots, \mathbf{b}^r$,
- the activation functions: $\sigma_1, \dots, \sigma_r$

Remark 3.2.2. It is key to consider composition since building functions this way will generate **non linearity** which is central for neural networks. A short intuitive illustration to understand this concept is to consider a FNN with multiple layers, endowed with a linear activation function, then a FNN would just behave like a single-layered FNN since the sum of linear functions is essentially another linear function².

Proposition 3.2.3 (Number of parameters). *The number of parameters that characterise $\mathbf{f} \in \mathcal{N}_r(I, d_1, \dots, d_{r-1}, O; \sigma_1, \dots, \sigma_r)$ is (assuming that $\sigma_1, \dots, \sigma_r$ involve no additional parameters):*

$$\sum_{i=1}^r (d_{i-1} + 1) d_i$$

²See for further details: <https://stackoverflow.com/questions/9782071/why-must-a-nonlinear-activation-function-be-used-in-a-backpropagation-neural-net>

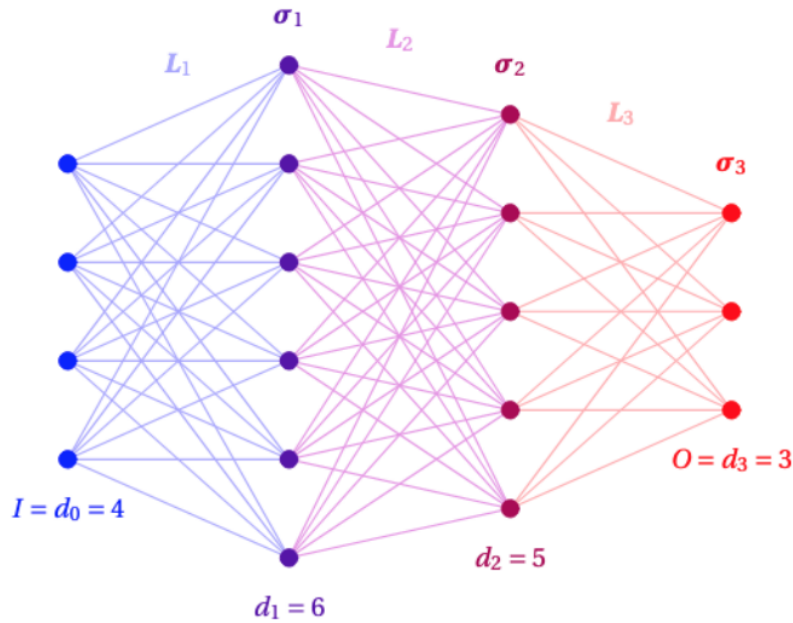


Figure 3.2: Graphical representation of a neural network with $r = 3$, $I = d_0 = 4$, $d_1 = 6$, $d_2 = 5$ and $O = d_3 = 3$ (source from Deep Learning lecture notes).

3.2.2 Activation functions

As introduced in definition 3.2.1, the activation function σ_i of the i -th layer is applied to the i -th affine function L_i which is in turn applied to the one plus i -th layer until this process reaches the last layer. As we've discussed previously, non-linearity is key for neural networks and hence most activation functions are non-linear (again, linear activation functions would only achieve linearity).

We can break down activation functions into two types: **one-dimensional and multi-dimensional ones**. Provided are some examples of common activation function, respectively as per fig. 3.3 and fig. 3.4.

3.2.3 Universal approximation theorem

Before delving into the technicalities of feedforward networks, we will first justify their use. The universal approximation theorem states that any "reasonable" function can be approximated by a suitable neural network. Let us formalize this idea.

First, we recall some theoretical prerequisites from topology. Let $K \subset \mathbb{R}^I$ be compact and we shall remember that K is compact if it is both closed and bounded. For any measure $f: \mathbb{R}^I \rightarrow \mathbb{R}$, the sup norm reads:

$$\|f\|_{sup,K} := \sup_{x \in K} |f(x)|$$

and, for any $p \geq 1$, the L^p norm reads:

$$\|f\|_{L^p(K)} := \left(\int_K |f(x)|^p dx \right)^{\frac{1}{p}}$$

Furthermore, we denote by $L^p(K, \mathbb{R})$ the class of measurable functions $f: K \rightarrow \mathbb{R}$ such that $\|f\|_{L^p(K)} < \infty$. The following theorem is adapted from Leshno et al. [23, Theorem 1 and Proposition 1].










Activation	Definition	Derivative	Range	Smoothness
 Identity (Id)	$g(x) = x$	$g'(x) = 1$	\mathbb{R}	C^∞
 Sigmoid (Logistic, σ)	$g(x) = \frac{1}{1 + e^{-x}}$	$g'(x) = g(x)(1 - g(x))$	$(0, 1)$	C^∞
 Heaviside (H)	$g(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	$g'(x) = 0, x \neq 0$	$\{0, 1\}$	none
 Hyperbolic tangent (tanh)	$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$g'(x) = 1 - g(x)^2$	$(-1, 1)$	C^∞
 Rectified linear unit (ReLU)	$g(x) = \max\{x, 0\}$	$g'(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$	$[0, \infty)$	C
 Parametric rectified linear unit (PReLU)	$g(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$ $\alpha > 0$	$g'(x) = \begin{cases} \alpha, & x < 0 \\ 1, & x > 0 \end{cases}$	\mathbb{R}	C
 Exponential linear unit (ELU)	$g(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$ $\alpha > 0$	$g'(x) = \begin{cases} g(x) + \alpha, & x < 0 \\ 1, & x > 0 \end{cases}$	$(-\alpha, \infty)$	$C^1, \alpha = 1$ $C, \alpha \neq 1$
 Softplus	$g(x) = \log(1 + e^x)$	$g'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$	C^∞
 Gaussian	$g(x) = e^{-x^2}$	$g'(x) = -2xg(x)$	$(0, 1]$	C^∞

Figure 3.3: Non-exhaustive list of one-dimensional activation functions (adapted from Wikipedia).

Theorem 3.2.4 (Universal approximation theorem). *Let $g: \mathbb{R} \rightarrow \mathbb{R}$ be a measurable function such that:*

- (a) g is not a polynomial function,
- (b) g is bounded on any finite interval,
- (c) the closure of the set of all discontinuity points of g in \mathbb{R} has zero Lebesgue measure.

Moreover, let $K \subset \mathbb{R}^I$ be compact and $\epsilon > 0$.

- (i) For any $u \in C(K, \mathbb{R})$, there exist $d \in \mathbb{N}$ and $f \in \mathcal{N}_2(I, d, 1; g, Id)$ such that:

$$\|u - f\|_{sup, K} \leq \epsilon$$

- (ii) Let $p \geq 1$. For any $v \in L^p(K, \mathbb{R})$, there exist $d \in \mathbb{N}$ and $h \in \mathcal{N}_2(I, d, 1; g, Id)$ such that

$$\|v - h\|_{L^p(K)} \leq \epsilon$$

Remark 3.2.5. It is important to note that theorem 3.2.4 holds only for networks with a single hidden layer. In practice, this theorem is usually extended to deeper networks.

Remark 3.2.6. Another important observation is that theorem 3.2.4 does not inform on how neural networks f and h look like, it only guarantees their existence. Similarly, it does not inform on how many hidden units the network should have, hence the parameter tuning of the network is up to the discretion of the tuner.

Activation	Definition	Derivative	Range	Smoothness
Softmax	$g_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^d e^{x_j}}, i = 1, \dots, d$	$\frac{\partial g_i(\mathbf{x})}{\partial x_j} = \begin{cases} g_i(\mathbf{x})(1 - g_i(\mathbf{x})), & i = j \\ -g_i(\mathbf{x})g_j(\mathbf{x}), & i \neq j \end{cases}$	$(0, 1)^d$	C^∞
Maxout	$g(\mathbf{x}) = \max\{x_1, \dots, x_d\}$	$\frac{\partial g(\mathbf{x})}{\partial x_i} = \begin{cases} 1, & x_i > \max_{j \neq i} x_j \\ 0, & x_i < \max_{j \neq i} x_j \end{cases}$	\mathbb{R}	C

Figure 3.4: Non-exhaustive list of multi-dimensional activation functions (adapted from Wikipedia).

3.2.4 Loss functions

To quantify and minimize the risk, we use a loss function defined as:

Definition 3.2.7.

$$\ell: \mathbb{R}^O \rightarrow \mathbb{R}^O$$

Given input $\mathbf{x} \in \mathbb{R}^I$ and output value $y \in \mathbb{R}^O$, if \mathbf{x} and y are a realisation of a joint random vector (\mathbf{X}, \mathbf{Y}) with \mathbf{X} random input vector and \mathbf{Y} random output vector, then when the distribution is known, we seek optimal f by minimizing risk

$$\mathbb{E}[\ell(f(\mathbf{X}), \mathbf{Y})]$$

In practice, the distribution is unknown hence the use of the empirical risk instead:

$$\mathcal{L}(f) := \frac{1}{N} \sum_{i=1}^N \ell(f(\mathbf{x}^i), \mathbf{y}^i) \quad (3.2.1)$$

3.2.5 Minibatch

Instead of using all samples to compute $\mathcal{L}(f)$, we use minibatch, a randomly drawn subset of samples. This helps tremendously to cope with large numbers of samples.

Consider the FNN $f_\theta := f \in \mathcal{N}_r(I, d_1, \dots, d_{r-1}, O)$. parametrised by the weight and bias vector $\theta := (W^1, \dots, W^r; \mathbf{b}^1, \dots, \mathbf{b}^r)$

Then, instead of considering eq. (3.2.1), we define minibatch risk as a function of θ .

$$\mathcal{L}_B(\theta) := \frac{1}{\#B} \sum_{i \in B} \ell(f_\theta(\mathbf{x}^i), \mathbf{y}^i) \quad (3.2.2)$$

where \mathcal{L}_B averages over any subset, minibatch, $B \subset \{1, \dots, N\}$ of samples.

Remark 3.2.8 (Batch Size). The number of samples in a minibatch is called the batch size.

3.2.6 Epochs

The number of times the network passes through the entire dataset is called epochs. Each epoch is divided into smaller datasets: minibatches. The tuning of epochs is somewhat complicated. On the one hand, an excess of epochs can produce overfitting which defeats the purpose of neural networks. On the other hand, a shortage of epochs will cause the model to be underfitting which means that the model hasn't learned enough. Hence, finding the right number of epochs is essential to training the network.

3.2.7 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the most common method when it comes to minimizing the empirical risk. Optimizing a function by seeking its minimum leads us to naturally

consider deriving the function. Let us consider first Ordinary Gradient Descent to introduce the concept:

Consider first the problem of minimising a generic differentiable objective function $F : \mathbb{R} \rightarrow \mathbb{R}^d$.

The natural approach is to compute the Hessian of F but in practice, it is not always feasible. However, we can compute an approximation using the differential equation:

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla F(\mathbf{x}(t)), \quad t > 0 \quad (3.2.3)$$

with initial condition $\mathbf{x}(0) \in \mathbb{R}^d$. This equation defines the so-called gradient flow $(\mathbf{x}(t))_{t \geq 0}$ of F (Santambrogio [30]). Under certain assumptions on F , which guarantee the existence of a unique minimiser and are stronger than mere convexity, it can be shown that $\mathbf{x}(t)$ tends to the minimiser as $t \rightarrow \infty$ ([30, Remark 2.1]). We can use this result by discretizing eq. (3.2.4) which reads:

$$\frac{\mathbf{x}(t + \eta) - \mathbf{x}(t)}{\eta} \simeq -\nabla F(\mathbf{x}(t)), \quad t > 0 \quad (3.2.4)$$

where ν is the increment amplitude, which is called the **learning rate** in machine learning. Rewriting the previous equation reads:

$$\mathbf{x}(t + \eta) \simeq \mathbf{x}(t) - \eta \nabla F(\mathbf{x}(t)), \quad t > 0$$

This Euler approximation motivates (ordinary) gradient descent, which is an iterative algorithm that progressively seeks a minimiser with gradient updates given some initial condition \mathbf{x}_0

$$\mathbf{x}_{new} := \mathbf{x}_{old} - \eta \nabla F(\mathbf{x}_{old}(t)), \quad t > 0 \quad (3.2.5)$$

We have derived with eq. (3.2.5) an algorithm that seeks to minimize our generic differentiable function F .

Now going back to our FNN, f_θ , that we defined in subsection 3.2.5. While we could think to use this algorithm to minimize $\mathcal{L}(f_\theta)$ as defined in eq. (3.2.1), it can be in practice computationally costly with large datasets, while gradient descent applied to $\mathcal{L}(f_\theta)$ may also lead to an overfitted network f_θ . To circumvent this limitation, we introduce **stochastic gradient descent** (SGD).

In SGD, we randomly split the training data into minibatches that are used successively to compute gradient updates. As outlined previously, after each minibatch, the output is fed to the next minibatch as input and the the parameter vector θ is updated each time:

$$\theta_i := \theta_{i-1} - \eta \nabla_{\theta} \mathcal{L}_{B_i}(\theta_{i-1}), \quad i = 1, \dots, k, \quad (3.2.6)$$

where $\mathcal{L}_{B_i}(\theta)$ is the minibatch empirical risk corresponding to minibatch B_i defined in eq. (3.2.2).

This procedure is then repeated over the number of epochs defined in the beginning, with new minibatches, while initialising with the last value of the previous epoch.

3.2.8 Backpropagation

As we've outlined in the previous section, being able to compute the gradient is essential to SGD and while there are a number of techniques to perform this computation, they often display limitations that we do not want in deep learning. The arguably most common technique is finite differences which is a linear approximation of the derivative over a small step size $\Delta > 0$:

$$F'(x) = \frac{F(x + \frac{1}{2}\Delta) - F(x - \frac{1}{2}\Delta)}{\Delta}$$

However, as one might have already noticed, this approximation is purely linear and does suit highly non-linear functions, often the case in deep learning.

To compute the gradient of a FNN, we will use backpropagation instead. Recall the FNN $f_\theta \in \mathcal{N}_r(I, d_1, \dots, d_{r-1}, O, \sigma_1, \dots, \sigma_r)$. Suppose that we are training this FNN by SGD and assume for simplicity that σ_i is the component-wise application of a one-dimensional activation function $g_i: \mathbb{R} \rightarrow \mathbb{R}$, for any $i = 1, \dots, r$.

As we want to minimize eq. (3.2.2) and because of linearity of the gradient operator, we will only need to study the minimization of the loss function $\ell(\mathbf{f}_\theta(\mathbf{x}^i), \mathbf{y}^i)$.

We introduce below some helpful recursive notation. For $x \in \mathbb{R}^I$,

$$\begin{aligned} \mathbf{z}^i &= (z_1^i, \dots, z_{d_i}^i) := \mathbf{L}_i(\mathbf{a}^{i-1}) = W^i \mathbf{a}^{i-1} + b_i, & i = 1, \dots, r, \\ \mathbf{a}^i &= (a_1^i, \dots, a_{d_i}^i) := \mathbf{g}_i(\mathbf{z}^i), & i = 1, \dots, r, \\ a^0 &:= \mathbf{x}, \end{aligned}$$

so then $\mathbf{f}_\theta = \mathbf{a}^r$ and $\ell(\mathbf{f}_\theta(\mathbf{x}^i)) = \ell(\mathbf{a}^r, \mathbf{y})$ and introduced the adjoint $\delta^i = (\delta_1^i, \dots, \delta_{d_i}^i) \in \mathbb{R}^{d_i}$ by:

$$\delta_i^j := \frac{\partial \ell}{\partial z_{d_i}^i}, \quad j = 1, \dots, d_i,$$

for any $i = 1, \dots, r$.

Remark 3.2.9. The chain rule is central to backpropagation. Consider differentiable $G: \mathbb{R}^d \rightarrow \mathbb{R}$ and $\mathbf{F} = (F_1, \dots, F_d): \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$ and define $H = G \circ \mathbf{F}: \mathbb{R}^{d'} \rightarrow \mathbb{R}$

The chain rule states:

$$\frac{\partial H}{\partial x_i}(\mathbf{x}) = \sum_{j=1}^d \frac{\partial G}{\partial y_j}(\mathbf{y}) \frac{\partial F_j}{\partial x_i}(\mathbf{x}), \quad \mathbf{x} = (x_1, \dots, x_{d'}), \quad \mathbf{y} = (y_1, \dots, y_d),$$

Proposition 3.2.10. *Using the chain rule and the adjoints we introduced earlier, we can derive a backward recursive procedure for the components of $\nabla_\theta \ell(\mathbf{f}_\theta(\mathbf{x}), \mathbf{y})$:*

$$\begin{aligned} \delta^r &= \mathbf{g}'_r(\mathbf{z}^r) \odot \nabla_{\hat{\mathbf{y}}} \ell(\mathbf{a}^r, \mathbf{y}) \\ \delta^i &= \mathbf{g}'_i(\mathbf{z}^i) \odot (W^i + 1)' \delta^{i+1}, & i = 1, \dots, r-1, \\ \frac{\partial \ell}{\partial b_j^i} &= \delta_j^i, & i = 1, \dots, r, j = 1, \dots, d_i, \\ \frac{\partial \ell}{\partial W_{j,k}^i} &= \delta_j^i a_k^{i-1}, & i = 1, \dots, r, j = 1, \dots, d_i, k = 1, \dots, d_{i-1}, \end{aligned}$$

where \odot stands for the component-wise Hadamard product of vectors.

3.3 Long short-term memory (LSTM)

Long short-term memory (LSTM) is a special case of Recurrent Neural Networks (RNN). RNN are known - unlike FNN - to remember their input. They are hence more viable solutions for path dependence than FNN, as we will show later for the case of autocallables.

In 1986, Rumelhart et al. [29] publish an article on a new learning procedure which will give birth to RNN like we know them today³.

³although it is important to note that this is a living theory that is still evolving today as we know it

3.3.1 Generic Recurrent Neural Networks

RNN are derived from FNN. Unlike the latter, where the information moves in a straight line with the output being exclusively fed as input to the next layer until the last layer is reached, RNN save the output of the previous layer and feeds it back into the model (the information does not go in one direction only).

Because of this feature, the algorithm is able to retain the information learned by the model and can continue to learn to achieve the correct prediction during backpropagation. fig. 3.5 illustrates this mechanism with the information stored and then fed back like in a loop.

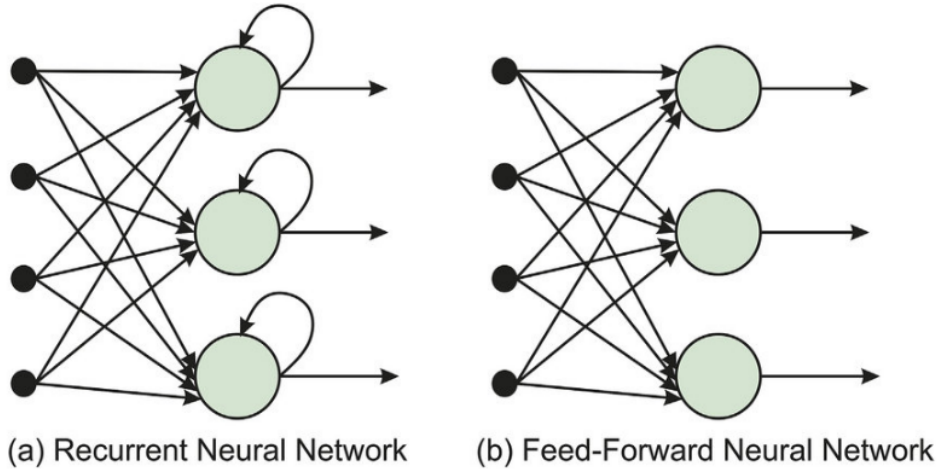


Figure 3.5: FFN feed the information in one straight line, but in RNN information is fed like in a loop which enables the model to remember past predictions [14, Figure 2, page 6].

However, RNN are subject to one major issue which is referred to in machine learning as *Vanishing Gradient*⁴.

Vanishing Gradient occurs during gradient descent when backpropagation is used to compute the gradient. Recall from eq. (3.2.6) that the parameter vector θ is updated after each minibatch. However, in some instances, the gradient will be very small⁵, preventing the weights to be updated. And because the chain rule is used in backpropagation to compute the gradient, this problem is exaggerated. Indeed, in an N-layer network, to compute the gradient of the early layers, N small numbers will be multiplied together which will essentially cause the gradient to vanish.

In the case of RNN, as the information is fed back to the model, this causes multiplications of gradients but because the gradient is already very small, this causes the gradient to become even smaller.

We provide an example to illustrate this idea adapted from [28].

Consider the function $f_{\theta} := f \in \mathcal{N}_r(I, d_1, \dots, d_{r-1}, O)$ parametrised by vector $\theta := (W^1, \dots, W^r; \mathbf{b}^1, \dots, \mathbf{b}^r)$ just like we defined previously for a FNN but this time let f_{θ} be the function we want to learn through our RNN algorithm.

Define $y_i \in \mathbb{R}^d$ and $x_i \in \mathbb{R}^d$ to be respectively the output and the input of the layer. We provide a stylized description of a generic RNN through the recursive equation: $y_i = f_{\theta}(x_{i-1}, y_{i-1})$.

⁴exploding gradient produces the same issues as well and it is caused by the same reasons as Vanishing Gradient
⁵hence the "vanishing" qualifier

To compute the gradient, we take the differential of y :

$$\begin{aligned}
dy &= \nabla_{\theta} f_{\theta}(y_{i-1}, x_i) d\theta + \nabla_y f_{\theta}(y_{i-1}, x_i) dy_{i-1} \\
&= \nabla_{\theta} f_{\theta}(y_{i-1}, x_i) d\theta + \nabla_y f_{\theta}(y_{i-1}, x_i) (\nabla_{\theta} f_{\theta}(y_{i-2}, x_{i-1}) d\theta + \nabla_y f_{\theta}(y_{i-2}, x_{i-1}) dy_{i-2}) \\
&= \nabla_{\theta} f_{\theta}(y_{i-1}, x_i) d\theta + \nabla_y f_{\theta}(y_{i-1}, x_i) (\nabla_{\theta} f_{\theta}(y_{i-2}, x_{i-1}) d\theta + \nabla_y f_{\theta}(y_{i-2}, x_{i-1}) (\nabla_{\theta} f_{\theta}(y_{i-3}, x_{i-2}) d\theta + \\
&\quad \nabla_y f_{\theta}(y_{i-3}, x_{i-2}) dy_{i-3}) \\
&= (\nabla_{\theta} f_{\theta}(y_{i-1}, x_i) + \underbrace{\nabla_y f_{\theta}(y_{i-1}, x_i) \nabla_y f_{\theta}(y_{i-2}, x_{i-1}) \cdots \nabla_y f_{\theta}(y_1, x_2)}_{i-1 \text{ times}}) d\theta + \dots
\end{aligned}$$

It is straightforward to see that successive multiplication of gradients will only accentuate the problem when the gradient is small, causing it to eventually vanish.

3.3.2 LSTM

Vanishing gradient has been well documented in the machine-learning literature and the LSTM algorithm has been designed to circumvent this issue. We will present now its fundamental mechanisms.

LSTM is a special case of RNN, and we specify here the architecture based on the work of Hochreiter and Schmidhuber [20]. While RNN might be able to handle short term memory, as the gap between saved information and the new one increases, so does the inability of RNN to connect the information. The LSTM architecture has been designed to handle long-range dependence and thus complicated path-dependence and it is perhaps right now the most popular RNN. It has produced very good results in multidisciplinary tasks such as speech and handwriting recognition [16, Section 10.10.1].

RNN and LSTM differ only through the number of layers present in the repeating module as shown in fig. 3.6.

Define $y_i \in \mathbb{R}^c$ and $x_i \in \mathbb{R}^d$ to be respectively the output and the input of the layer. The LSTM layer can be summarized with the recursive equation:

$$(y_t, C_t) = f_{\theta_{\text{LSTM}}}(x_{t-1}, y_{t-1}, C_{t-1}) \quad (3.3.1)$$

where $C_t \in \mathbb{R}^c$ is the cell state vector at time t with $c \in \mathbb{N}$ units and θ_{LSTM} is the parameter of the LSTM layer. We sketch in what follows the basic mechanism of the LSTM layer, more details on the algorithm and the LSTM layer can be found with Goodfellow et al. [16, Section 10.10.1] and Hochreiter and Schmidhuber [20].

- (i) **Forget gate layer:** Given y_{t-1} and x_t , the forget gate f_t outputs a number between 0 and 1 to cell C_{t-1} :

$$f_t = \sigma_{\text{sig}}(L_f(y_{t-1}, x_t))$$

where L_f is the affine function parametrised by weights and biases, introduced in definition 3.2.1 and σ_{sig} is the sigmoid activation function shown in fig. 3.3.

- (ii) **Deciding which information is stored in the cell state:** Given y_{t-1} and x_t , the input gate i_t outputs a number between 0 and 1 and a new vector, \tilde{C}_t is created that outputs a number between -1 and 1.

$$i_t = \sigma_{\text{sig}}(L_i(y_{t-1}, x_t))$$

$$\tilde{C}_t = \tanh(L_C(y_{t-1}, x_t))$$

where \tanh is the hyperbolic tangent activation function shown in fig. 3.3.

- (iii) **Updating the cell state C_t :** Then, the cell state C_t is updated by combining the old and new information.

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t$$

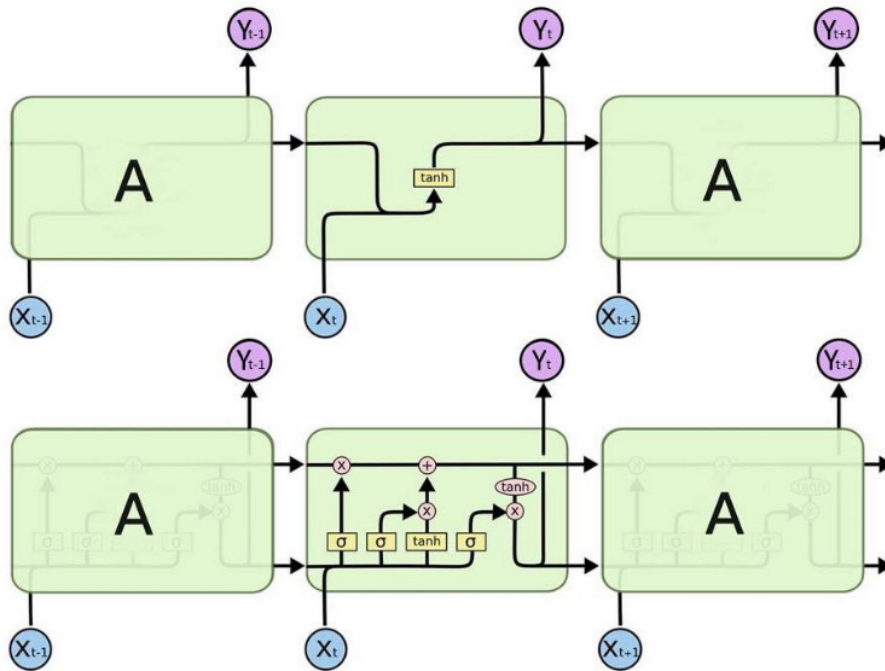


Figure 3.6: In a standard RNN (top image), the repeating module contains a single layer whereas in a LSTM (bottom), it contains four interacting layers. Source: Christopher Olah, used with permission.

- (iv) **What information to output:** Finally, the output is filtered by an output gate which returns a value between 0 and 1 and the output produced by the output gate is then multiplied by a tanh function that bounds values between -1 and 1.

$$o_t = \sigma_{sig}(L_o(y_{t-1}, x_t))$$

$$y_t = o_t \circ \tanh(C_t)$$

Chapter 4

Description of Autocallables and their use

Autocallables - also referred to as Autocalls - are very popular structured products that provide a redemption feature meaning investors will recover their notional once the underlying breaches a given Autocallable Barrier.

There are several variants for Autocallables - each displaying this redemption mechanism - which provide additional features, whether it is a memory coupon, worst/best of options etc. For the sake of this thesis, we will focus on a typical Autocallable with a downside consisting of a down-and-in put whose barrier is observed continuously. In the first section, we will cover the basic mechanism of Autocalls and then focus on this example in detail.

4.1 Mechanisms and features

4.1.1 Coupon and Redemption

In the case of an Autocall, there are two features which are shared by (almost) all the different variants and that are Redemption and Coupon. As such, in this section we will first describe those mechanisms:

An Autocall pays a periodic coupon to the investor as long as the underlying price is above the Coupon Barrier. On the other hand, the product will be redeemed to the investor if the underlying breaches the Autocall Barrier.

Denote the Coupon Barrier, B_C and Autocall Barrier, B_{AC} . Formally, we define the Coupon and Redemption using the same notations as per [6, Chapter 12.1, page 187] :

$$\begin{cases} Coupon(t_i) = Notional \cdot C \cdot \mathbf{1}_{\{Ret(t_i) \geq B_C\}} \cdot \mathbf{1}_{\{\max_{j=1, \dots, t-1} Ret(t_j) < B_{AC}\}} \\ Redemption(t_i) = Notional \cdot C \cdot \mathbf{1}_{\{Ret(t_i) \geq B_{AC}\}} \cdot \mathbf{1}_{\{\max_{j=1, \dots, t-1} Ret(t_j) < B_{AC}\}} \end{cases} \quad (4.1.1)$$

where $t_i, (i=1, 2, \dots, n)$ represents each observation date. C is the Coupon Level expressed as the coupon in terms of the percentage of the notional and $Ret(t_i) := S(t_i)/S(0)$ is the underlying (where $S(t_i)$ denotes the spot price at time t_i) return between time t_i and time $t_0 = 0$.

In plain english, a Coupon will paid to the investor on each Observation if the Underlying price is above the Coupon Barrier and if the product hasn't autocalled on the said Observation date or before.

Similarly, a product has not autocalled on date t_i if the underlying price hasn't breached the Autocall Barrier on date t_i as well as on each observable date before.

Remark 4.1.1. Some autocallables have the mention "Reverse Convertible". What it means is that they provide a guaranteed Coupon i.e the Coupon Barrier is set to 0, hence the Autocall Reverse Convertible will pay a Coupon on every Observation date provided that it hasn't autocalled.

4.1.2 Barrier observation

There are 4 main types of barriers.

- American/Continuous: The barrier is observed continuously. Hence, if the underlying price breaches (i.e goes below) the barrier at any given time, then the barrier is breached.
- European: The barrier is observed at maturity. Hence, only if the underlying price breaches (i.e goes below) the barrier at maturity, then the barrier is breached.
- Daily/Close-to-close: The barrier is observed daily. Sometimes, it is also referred as close-to-close since the barrier is observed daily on the close. Hence, if the underlying price breaches (i.e goes below) the barrier at the close during the lifetime of the product, then the barrier is breached.
- Periodic: The barrier is observed every period. The most popular periodic observations are quarterly i.e the barrier is observed every 3 months, semi-annually and yearly. Usually, they are observed on the close.

For example, on June 07 2022, the S&P 500 (^GSPC ticker in Yahoo Finance) closed at 4,160.68. Hence, if a barrier is observed daily/Close-to-close on the S&P 500, then the underlying would have breached this barrier as long as the barrier was above or equal to 4,160.68.

4.1.3 Adding downside with a Down-and-in Put

As one might have already realized from the specifications in eq. (4.1.1), this product doesn't have any downside since nothing is paid in the worst-case scenario. Hence, an Autocall with no downside might pay very low coupons or even sometimes no Coupon at all. In practice, investors seek high level of coupons and as such they are ready to give up capital protection in exchange for higher coupons. This usually translates for the investor into being short a Down-and-in Put (DIP). As one might recall, a DIP is a Put with a Barrier, where the latter can be observed in a certain number of ways which we outlined in subsection 4.1.2.

In the case of a European Barrier (Barrier is observed at maturity), the investor is short a DIP with maturity equal to the product maturity. This entails that if the underlying price at maturity ends up below the DIP Barrier, then the investor's capital is no longer protected as he will not fully retrieve the notional they invested. Instead, they will receive the performance of the asset (meaning that if the asset ends up at 0, then the investor's money will be completely lost). In subsection 4.1.4, we carry out an in-depth analysis of the payoff in all possible scenarios with this DIP feature.

4.1.4 Product Payoff with possible scenarios

Early redemption

- On each observation date¹, if the underlying price is equal or superior to the Autocall Barrier on a given observation date, then the product is redeemed and the investor receives:

$$\text{Final Payoff} = 100\% + \text{Coupon}$$

- Else, product continues and:
 - If the underlying price is equal to or superior to the Coupon Barrier:

$$\text{Intermediary Payoff} = \text{Coupon}$$

- Else:

$$\text{Intermediary Payoff} = 0$$

¹Again can be continuous, daily, quarterly etc.

Redemption at maturity (in case of no early redemption)

- If the underlying closes at or above the Protection Barrier, redemption at:
 - If the underlying closes at or above the Coupon Barrier, redemption at:

$$\text{Final Payoff} = 100\% + \text{Coupon}$$

- If the underlying closes below the Coupon Barrier, redemption at:

$$\text{Final Payoff} = 100\%$$

- If the underlying closes below the Protection Barrier, redemption at:
 - If the underlying closes at or above the Coupon Barrier, redemption at:

$$\text{Redemption} = \text{Notional} \cdot \text{Ret}(t_{final}) + \text{Coupon}$$

- If the underlying closes below the Coupon Barrier, redemption at:

$$\text{Redemption} = \text{Notional} \cdot \text{Ret}(t_{final})$$

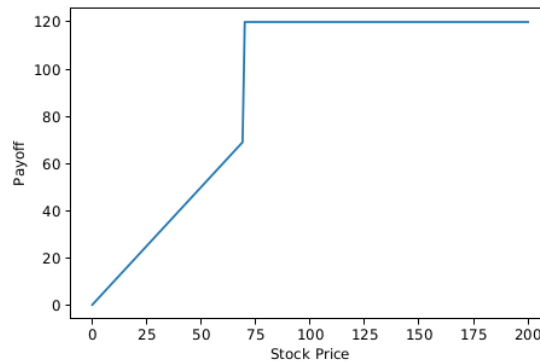


Figure 4.1: Autocall payoff at maturity with Protection Barrier = Coupon Barrier = 70, Autocall Barrier 120, Coupon level = 20. This figure clearly shows that the final Payoff can be broken down into a combination of a DIP to model the downside and a digital call to model the upside with the coupon being paid.

4.2 Pricing Approach

The Autocallability feature automatically redeems the product if the underlying breaches the Autocall Barrier which implies that this product has varying maturity. To model this property, different approaches have been used in the industry. Deng et al. in [12] have studied the PDE approach where they used finite difference to approximate the derivatives involved in the Black-Scholes PDE. Alm et al. in [1] derive a Monte Carlo algorithm that greatly reduces variance using one-step survival techniques.

While these methods provide interesting approaches to value the autocallable, there is no consensus on how this popular product should be priced. For this reason, indifference pricing using neural networks would present itself as a new valuation method while being able to easily incorporate market frictions. However, we are still interested in having a benchmark to value these instruments and to so we will be valuing Autocalls using standard Monte Carlo techniques as we outline in this section thereafter.

4.2.1 Monte Carlo

A standard way to price products is to use Monte Carlo Simulation. Essentially, we can summarize the pricing method as follows:

- (i) Simulate price paths under a given model.
- (ii) For each simulation, compute the Cash Flows perceived by investing in this product.
- (iii) Discount the Cash Flows and take the average across all simulations.
- (iv) The product value at time 0 is this average.

Hence to value the autocallable numerically, we use the same approach. This is summarized in algorithm 1.

Algorithm 1 Monte Carlo Algorithm to price a generic autocall under a given model

```

T ← number of steps
N ← number of simulations
Generate stock path:  $S_t^n, t \in \{0, T\}, n \in \{0, N\}$ 
for  $n = 1, 2, \dots, N$  do
   $V_n \leftarrow 0$ 
  for  $t = 0, 1, \dots, T$  do
     $DF(r, t) \leftarrow e^{-r \cdot t}$  ▷ Discount Factor
    if  $t \leq (T - 1)$  then ▷ During Product lifetime
      if  $B_C < S_t^n < B_{AC}$  then ▷ Product pays a Coupon and does not Autocall
         $V_n \leftarrow V_n + C \cdot S_0$ 
         $V_n \leftarrow V_n \cdot DF(r, t)$  ▷ Discounting CF
      else if  $S_t^n \leq B_C$  then ▷ Product does not Pay a Coupon
         $V_n \leftarrow V_n + 0$ 
      else if  $B_{AC} \leq S_t^n$  then ▷ Product has Autocalled and pays a final Coupon
         $V_n \leftarrow V_n + S_0 \cdot C$ 
         $V_n \leftarrow V_n \cdot DF(r, t)$ 
        break
      end if
    else ▷ At Maturity, if no early redemption
       $V_n \leftarrow V_n + S_0 - \max(K - S_t^n, 0) \cdot \mathbf{1}_{\{S_t^n \leq B_C\}} + \mathbf{1}_{\{S_t^n > B_C\}} \cdot C \cdot S_0$ 
       $V_n \leftarrow V_n \cdot DF(r, t)$ 
    end if
  end for
end for
return  $\hat{V}_0 = \frac{V_1 + \dots + V_N}{N}$ 

```

4.2.2 Static hedging: breaking down the product into a combination of options

Static hedging can also be used to value a portfolio of derivatives, and has been advocated in Carr and Madan's famous paper on valuing variance swaps and its extensions [8]. A static hedge in essence does not produce perfect replication but it can provide a good and fast approximation. In the following, we study an example of a generic autocallable with periodic coupons and downside as we presented in subsection 4.1.3. The AC BRC can be broken down into a sum of options and a Zero Coupon Bond. Concretely, the investor is:

- Short a Down-and-in Put (DIP) with Strike = 100% on the downside.
- Long a Zero Coupon Bond that pays a guaranteed Coupon at maturity.
- Long a strip of conditional digital calls to model the Coupon and Autocallable feature.

Remark 4.2.1. As long as this product doesn't cross the DIP Barrier, this product is 100% capital protected.

4.3 Hedging issues

As we hinted in the thesis introduction, autocallables are in fact challenging product to hedge as greeks around barriers usually explode or change signs. We shall use the replication derived in subsection 4.2.2 to highlight these issues.

4.3.1 Downside: replicated by a DIP

First consider the downside represented by a short DIP. We provide in fig. 4.2 the delta associated with the position. As we can see, the short put has a delta that jumps and in this case - although it can attain even higher values for lower barriers and other option parameters. Effectively, the delta reaches gradually double the value of a vanilla put and then instantly goes to 1 once the DIP barrier is breached at 0.5.

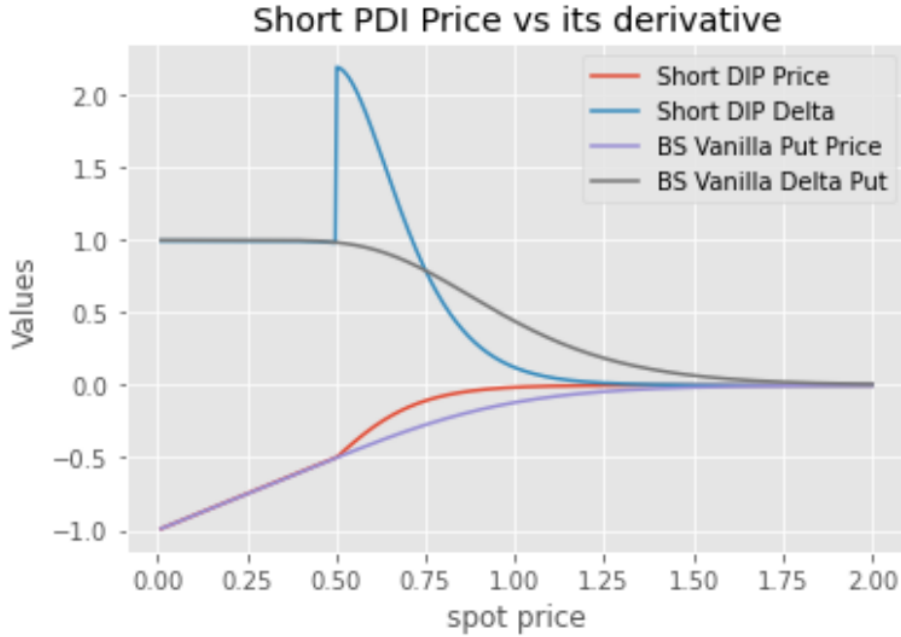


Figure 4.2: Price and delta for a short European put with $r = 0, \sigma = 0.3, T = 1, K = 1$ and price and delta for a short DIP with the same specifications and a barrier $B_{DIP} = 0.5$.

4.3.2 Upside: replicated by a strip of digital calls

Second, consider the upside, modeled by a strip of conditional digital calls to model the coupon and early redemption feature. We provide in fig. 4.3 the price and delta for a European digital call under Black-Scholes. As one can see in our example, the delta of the digital call quickly accumulates a lot of leverage around the barrier. Effectively, the hedger of the option has to be short more than the amount of notional he has on the position. Once the barrier is breached, the delta falls very quickly so the hedger has to unwind quickly all the delta accumulated. This is even more accentuated when the option gets closer to expiry where the delta explodes even more around the barrier.

The underlying reason why the delta reaches values as high is because of the discontinuity of the payoff of the digital call which is either 0 or 1, depending whether the spot price is lower or higher than the barrier. One way to smooth out the payoff is to consider a call spread. Indeed, consider

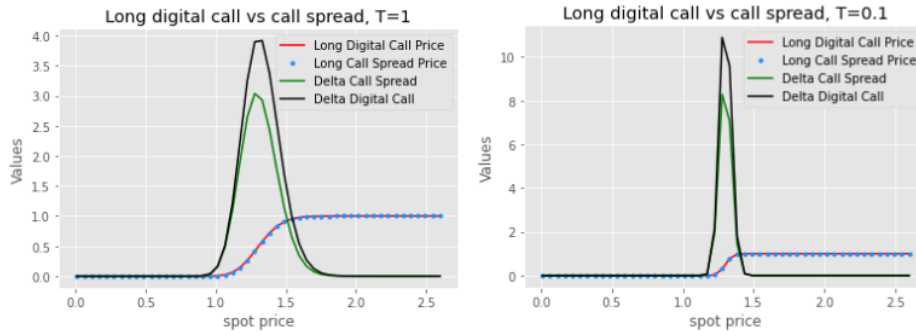


Figure 4.3: Price and delta for a long European Digital Call with $r = 0, \sigma = 0.1, B_{\text{Digital}} = 1.3$ and price and delta for a call spread with the same specifications first strike $k_1 = 1.28$ and second strike $k_2 = 1.32$ with notional $= \frac{1}{\text{width}}$ where $\text{width} = k_2 - k_1$. In the top figure, the option has maturity left $T=1$ and in the bottom figure, it has maturity left $T=0.1$.

the digital call payoff:

$$\begin{aligned} \text{Payoff Digital Call} &= \mathbf{1}_{\{S_T > H\}} \\ &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \{(S_T + \epsilon - H)^+ - (S_T - H)^+\} \end{aligned} \quad (4.3.1)$$

where H is the Barrier of the digital call.

To approximate the digital call payoff, we may find a small ϵ such that the call spread payoff roughly matches the digital call payoff such as in fig. 4.4. The difference between the two strikes (the ϵ in eq. (4.3.1)) is called the barrier width and it is the notional that the seller of the option of the position will have on his call spread position. If the barrier width is 0.05 for instance, then the seller of the option of the position will be effectively long 20 call spreads. The larger the width, the more conservative the seller of the option of the option is.

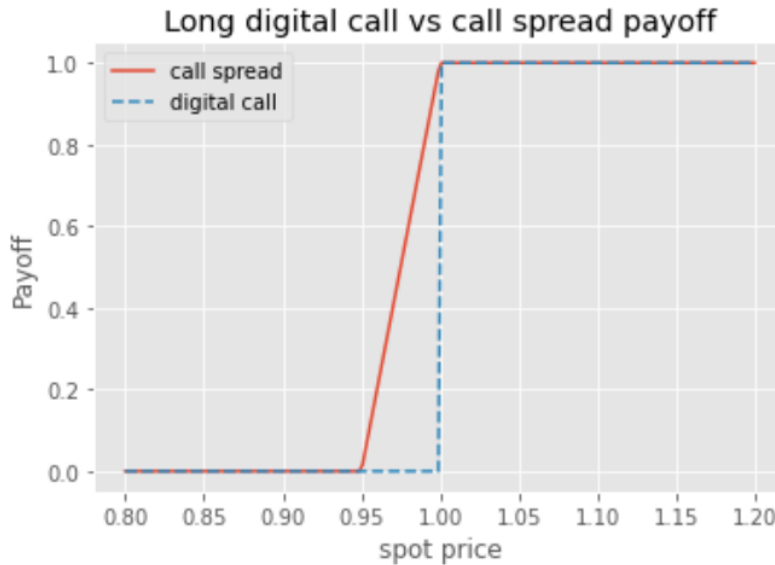


Figure 4.4: Payoff for a long European Digital Call with barrier $H = 1$ and payoff for a Digital Call spread with first strike $k_1 = 0.95$ and second strike $k_2 = 1$ with notional $= \frac{1}{k_2 - k_1} = 20$.

While this is an improvement as shown in fig. 4.3 with a delta that is lower with a call spread than the digital call, it is however still very high. In our example, when the expiry is in $T=0.1$ years, then the seller of the option is effectively short 8 times the notional around the barrier. This behavior around the barrier is also observed for the vega which is also a very important risk factor in the case of the digital call. As shown in fig. 4.5, the vega changes sign before and after the barrier which means that the trader might have to take a completely opposite position as the one they had before. This is even more accentuated as the option gets closer to its expiry.

Traditionally, the autocallable is hedged as a combination of options where the hedge is implemented through the greeks. However, as we have just seen, the behavior of barrier and digital options around the barrier using a greek approach is very challenging and requires to actively monitor the portfolio of options for each barrier and important risk factor.

Hedging the autocallable with the deep hedging method alongside indifference pricing would entail that the hedging is not an approximation anymore. In other words, the seller of the option no longer has to monitor the barriers for each risk factor. In addition, it is straightforward able to incorporate market frictions such as transaction costs inherent to markets. What's more, because the hedging is done through neural networks, theoretically, the selling/buying of derivatives would be fully automated. In the next section, we will discuss our results obtained from deep hedging.

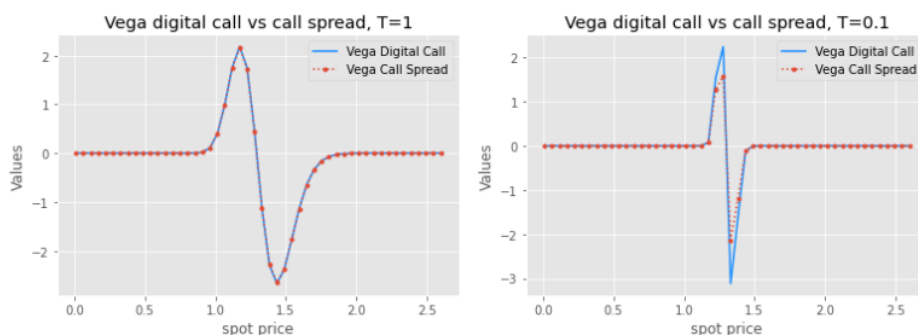


Figure 4.5: Vega for a long European Digital Call with $r = 0, \sigma = 0.1, T = 0.1, B_{\text{Digital}} = 1.3$. The right figure is for an option with expiry $T=1$, the left is with expiry $T=0.1$.

Chapter 5

Deep Hedging Autocallables: numerical results

5.1 Model setting

5.1.1 Neural Networks

In hidden layers, ReLU (as shown in fig. 3.3) has become the default activation function. We decide to choose hence ReLU for the hidden layers and Softplus (shown in fig. 3.3) for the output activation function as a smoothed version of ReLU and as a function that is not bounded on the upside. Concretely, f has architecture:

$$f \in \mathcal{N}_4(2, 100, 100, 100, 1; \text{ReLU}, \text{ReLU}, \text{ReLU}, \text{Softplus}).$$

Although this approach is in essence model-free, the number of price paths required substantially exceeds the available data in the market. To this end, we simulate price paths S_1, \dots, S_N , $N \in \mathbb{N}$.

Then, we train f using the exponential utility function for the loss function and specify four levels of risk aversion λ and four levels of transaction costs k , respectively $\lambda \in \{1, 5, 10, 15\}$ and $k \in \{0, 0.05\%, 0.5\%, 5\%\}$.

The tuning of the hyperparameters is not an exact science and after testing different parametrizations, we decide to train the FNN with 30 epochs and 100 for the minibatch size.

In the case of the LSTM, we specify a single LSTM layer with one-dimensional input with $c = 20$ units in the LSTM cell where Softplus is the activation function. Then, we train the network over different learning rates. Unlike the FNN, the LSTM converges slower and separating the training into two different learning rates ensures faster convergence of the LSTM network. Concretely:

- (i) First, we train the network with 10 epochs and mini-batch size 5000. We run ADAM with a large learning rate=0.01.
- (ii) Second, we train the network 10 times with 10 epochs and mini-batch size 5000. However, this time we run ADAM with a smaller learning rate=0.001.

Both Neural Networks were implemented in Google Colaboratory with a GPU accelerator. We provide code snippets to reproduce the results in github: <https://github.com/BassamSINAN/Deep-hedging-of-Autocallables-with-rough-Bergomi-model>.

5.1.2 Product specification

As for our claim Z , we aim to hedge a simple autocall with specifications:

- Autocall Barrier with continuous observations: $B_{AC} = 1.3$
- DIP Barrier with continuous observations: $B_{DIP} = 0.7$

- Product struck at the money: $K = S_0 = 1$
- Maturity: $T = 1$ year
- Coupons are paid either at maturity if the product doesn't autocall during the lifetime of the product, or paid when the product autocalls

Mathematically, the payoff can be written as:

$$Z = \underbrace{\mathbf{1}_{AC}(S_0 + c)}_{(*)} + \underbrace{(1 - \mathbf{1}_{AC})((S_0 + c) - (K - S_T)^+ \mathbf{1}_{DIP})}_{(**)} \quad (5.1.1)$$

where

$$\mathbf{1}_{AC} = \begin{cases} 1, & \text{if } \sup_{t \in [0, T]} S_t \geq B_{AC}, \\ 0, & \text{otherwise} \end{cases}$$

is the indicator function that outputs 1 if the product has autocalled and 0 otherwise and

$$\mathbf{1}_{DIP} = \begin{cases} 1, & \text{if } \inf_{t \in [0, T]} S_t \leq B_{DIP}, \\ 0, & \text{otherwise} \end{cases}$$

is the indicator functions that outputs 1 if the product has crossed the DIP Barrier and 0 otherwise.

If the product has autocalled, then the product will pay $S_0 + c$ i.e the investors will get their notional back as a well as a coupon on top given by $(*)$ in eq. (5.1.1). However, if the product hasn't autocalled $(**)$ during the lifetime of the product, then there are two cases to consider. Either the DIP Barrier hasn't been breached during the lifetime of the product so the product will pay at maturity $S_0 + c$. The second case is when the product has breached the barrier, so it will pay $S_0 + c$ minus the negative performance of the product. For example, if the barrier has been breached and the asset is down 30% at maturity, then the product will pay 70% back plus c .

5.2 The hedging strategy

In order to compare the hedge under different model settings, we simulate S under Black-Scholes, Heston and Rough Bergomi.

In subsection 5.2.1, we aim to investigate which ANN between FNN and LSTM is able to best capture the right hedging strategy. We also investigate how risk aversion and transaction costs affect the hedging. Then in subsection 5.2.2, we study the hedging under calibrated models.

5.2.1 Handling corner cases

Our goal is to assess whether the two ANNs provide a sensible hedging strategy. With this in mind, *we fix toy parameters for each pricing model such that we are able to study corner cases.*

For each model, we simulate the stock price path with:

$$S_0 = 1 \text{ (values were normalized), } T = 1, \quad r = 0, \\ N = 100,000, \quad \text{steps} = 100$$

Rough Bergomi

In order to simulate the asset price under the rough Bergomi model, we refer to section 1.5 in which we detail how we use the hybrid scheme to simulate the associated Volterra process. We use numerical values:

$$\xi = 0.6, \quad \rho = -0.8, \quad \alpha = -0.43 \text{ (or } H = 0.07), \quad \eta = 1.9$$

Heston

In the Heston model, the stock price is governed by the following set of stochastic differential equation:

$$\begin{aligned} dS_t^H &= S_t^H \sqrt{V_t} dW_t^1, & S_0 &= s > 0, \\ dV_t &= \kappa(\theta - V_t)dt + \sigma_V \sqrt{V_t} dW_t^2, & V_0 &= v_0 > 0, \\ d\langle W^1, W^2 \rangle_t &= \rho dt, \end{aligned} \quad (5.2.1)$$

where $\kappa, \sigma_V, \theta, v_0, s > 0$ and the correlation parameter ρ lies in $[-1, 1]$. In eq. (5.2.1), the process $(V_t)_{t \geq 0}$ represents the instantaneous variance (squared volatility) of the underlying stock price S^H and W^1, W^2 are two Brownian Motion with correlation ρ . κ represents the mean-reversion rate i.e the rate at which V_t reverts towards θ which represents the long-run average variance of the price. Furthermore, v_0 is the initial variance and σ_V is the volatility of the square root of the variance process.

To simulate the stock price, we discretize the previous equations and use:

$$\kappa = 3, \quad \theta = 0.04, \quad v_0 = \sigma^2, \quad \xi = 0.6, \quad \rho = -0.8$$

Black-Scholes

First, we recall the stochastic equation governing the Stock Price in the continuous Black-Scholes setting:

$$S_t^{BS} = S_0 \exp(\mu t + \sigma W_t), \quad t \in [0, 1]$$

where $W = (W)_{t \in [0, 1]}$ is a standard Brownian Motion. To simulate the stock price, we discretize the previous equation and use:

$$\sigma = 0.5, \quad \mu = -\frac{1}{2}\sigma^2$$

Numerical Results

As we recall from eq. (2.4.3), we can obtain prices for both FNN and LSTM networks. We compile prices in table 5.1 for both networks under different risk aversions and transaction costs. As we expect, prices increase with risk aversions and transaction costs. We also notice that prices under LSTM are lower to FNN which is not intuitive at first glance. We will need to delve into more details to understand why this is the case.

Parameters		Black-Scholes		Heston		Rough Bergomi	
Risk Aversion	Transaction costs	FNN	LSTM	FNN	LSTM	FNN	LSTM
$\lambda = 1$	k=0.00%	0.87189	0.87089	0.93429	0.93388	0.84793	0.84501
	k=0.05%	0.87378	0.87189	0.93579	0.93497	0.84974	0.84495
	k=0.50%	0.88317	0.87774	0.94222	0.94062	0.85887	0.85062
	k=5.00%	0.89125	0.89125	0.94700	0.94700	0.87623	0.87623
$\lambda = 5$	k=0.00%	0.88883	0.87839	0.94184	0.94081	0.89020	0.86591
	k=0.05%	0.89093	0.88107	0.94363	0.94161	0.89089	0.86823
	k=0.50%	0.90634	0.88758	0.95597	0.95012	0.90189	0.87636
	k=5.00%	0.94604	0.93082	0.98166	0.98168	0.94656	0.92172
$\lambda = 10$	k=0.00%	0.91666	0.88888	0.95370	0.94660	0.93303	0.90008
	k=0.05%	0.91811	0.89407	0.95619	0.95110	0.93425	0.89406
	k=0.50%	0.93300	0.89983	0.96858	0.96045	0.94342	0.90786
	k=5.00%	0.97702	0.95130	1.00022	0.99482	0.97998	0.95010
$\lambda = 15$	k=0.00%	0.94156	0.91127	1.00893	0.96123	0.96175	0.98801
	k=0.05%	0.94527	0.91020	0.98044	0.95892	0.96079	0.98011
	k=0.50%	0.95566	0.91149	1.00880	0.97054	0.96842	0.99089
	k=5.00%	0.99666	0.96953	1.00893	1.00443	0.99487	0.98849
Analytical Price(Monte-Carlo)		0.87152		0.92954		0.84575	

Table 5.1: Comparing indifference pricing for the three different models under FNN and LSTM against pricing the autocal under standard Monte-Carlo simulations.

Evolution of the hedging ratio across varying levels of spot price

Next, we plot the deep hedging ratio prediction δ_{70} for both networks for across spot price $s \in]0, 2]$ for different level of transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ and for $\lambda = 10$ under Black-Scholes, Heston and Rough-Bergomi. The various figures showcase how δ_t evolves with spot price.

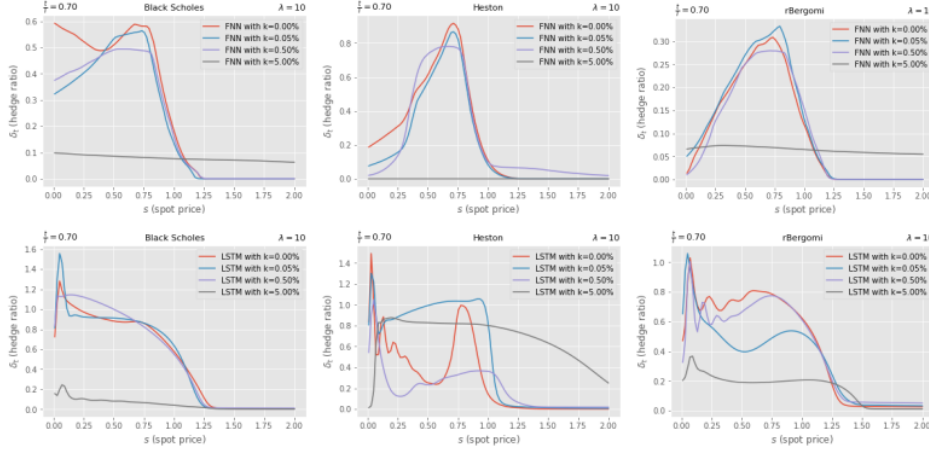


Figure 5.1: Deep hedging ratio δ_{70} under FNN across different times and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Black-Scholes, Heston and Rough-Bergomi. The figures showcase how δ_t evolves with spot price. Autocallable with $B_{AC} = 1.3$ and $B_{DIP} = 0.7$.

Figure 5.1 tells us that the hedging ratio seems to decrease quickly from $s=0.7$, which is equal to the DIP barrier, and reaches 0 when the spot price gets close to the autocall barrier equal to 1.3. Indeed, as we recall, the autocall knocks out as the autocall barrier is breached which entails that the hedge is liquidated. However, the behavior preceding the DIP barrier is less straightforward. In FNN, the hedging ratio seems to attain its peak as spot prices get close to the DIP barrier displaying a bell curve. The hedging strategy seems to decrease to low values as spot prices get closer to 0. However, in LSTM, while the curvature of the bell found in FNN is somewhat present, the peak this time is attained for low spot prices, with a hedging ratio that can exceed 1.

In particular, fig. A.1, fig. A.2, fig. A.3 for FNN and fig. A.7, fig. A.8, fig. A.9 for LSTM given in the appendix exhibit the evolution of the hedging ratio under different risk aversion parameters *ceteris paribus*. For FNN, we observe that a higher lambda will increase the curvature of the bell curve we observed earlier. For LSTM, this is not as blatant but it does seem that increasing risk aversion decreases the smoothness of the hedge, especially around low spot prices and around the DIP Barrier.

Furthermore, as we recall in subsection 4.2.2, we can approximately replicate the downside of the autocallable with a short DIP option and the upside with being long a strip of digital calls. We recognize in fig. 5.1 under LSTM for Black Scholes, Heston and rBergomi somewhat the shape of the delta of the short DIP (shown in fig. 4.2) represented by a bell curve which is surely accentuated by the delta of the digital call (shown in fig. 4.3). This bell curve is somewhat mitigated by the model used and its parameters, for instance in our simulation with Heston, we can clearly see it. In Black Scholes, it is much smoother.

Evolution of the hedging ratio across two scenarios

To understand this behavior, we plot realized paths for two scenarios A and B where in the first one, the autocall barrier is breached and in the second one it is not. We then study the hedging strategy under the two neural networks and we decide to stick with $\lambda = 10$ as results are more conclusive under this risk aversion parameter. This is shown in fig. 5.2 and full results can be found in appendix A.1.2 and appendix A.2.2 with all levels of λ .

We observe that once the Autocallable barrier is breached, the FNN hedging ratio does not necessarily go to 0. This is contrary to our expectations as we predicted that the hedging ratio would decrease as the spot price gets closer to the Autocall Barrier since the product would then knock out. This is however observed under LSTM which as we outlined in section 3.3 has the ability to handle long-range dependence. This makes it a better candidate than FNN for handling path-dependence.

This example highlights the power that the LSTM offers to price and hedge claims that may or may not be path dependent. While there is no universal consensus on the analytical models that are used to value complicated path dependent claims, this approach under LSTM provides a systematic and tractable way to do so while allowing easily to add market frictions such as transaction costs. Horvath et al. [21, Section 1] have shown for the case of the Up-and-out call that the LSTM seemed to accurately capture the hedging ratio and handle the instrument's path dependence. We have here extended this result to a more complicated path dependent payoff, doing so under the rough Bergomi model which captures more realistically the underlying path.

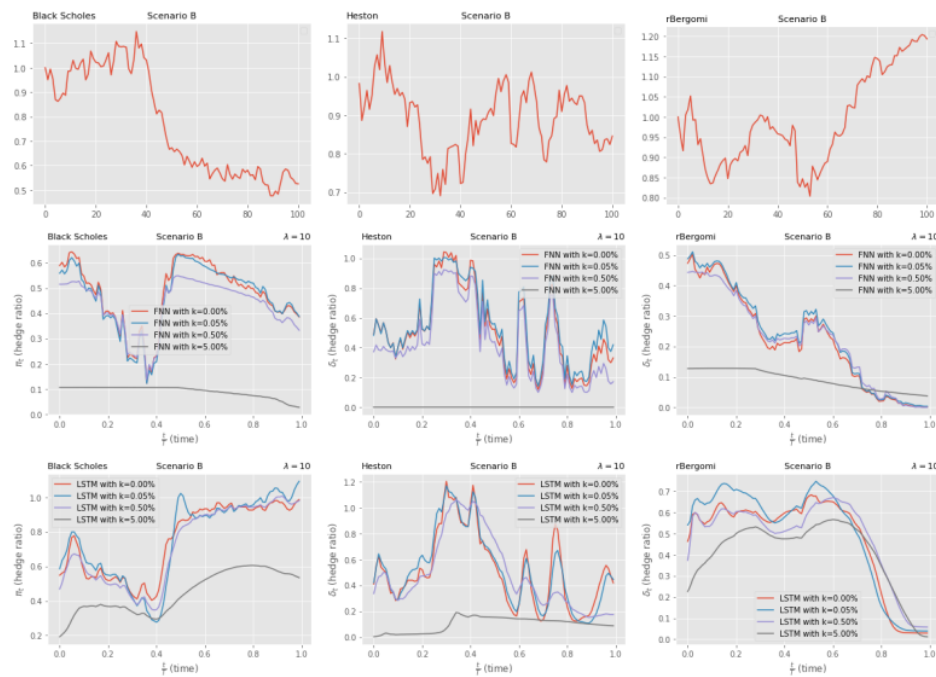


Figure 5.2: In the top row, Realized paths are plotted for the three pricing models over scenario B. In the middle row, the hedging strategy is implemented respectively for each model under FNN and in the last row, the hedging strategy is implemented under LSTM.

5.2.2 Comparing the calibrated models

We now proceed to compare the models in such a way that they are calibrated to real data - the SP 500. Products parameters detailed in subsection 5.1.2 are kept the same besides the Autocall Barrier which we adjust to 1.2 instead of 1.3.

Rough Bergomi

To calibrate rough Bergomi, we use values such that they correspond to a maturity with T=1 year obtained in our calibration in subsection 1.5.1. Concretely, we use:

$$\xi = 0.13, \quad \rho = -0.95, \quad \alpha = -0.40 \text{ (or } H = 0.1), \quad \eta = 2.08$$

Heston

We use numerical values:

$$\kappa = 6.17, \quad \theta = 0.03, \quad v_0 = 0.04, \quad \sigma_V = 0.45, \quad \rho = -0.78$$

where the parameters above were chosen such that they empirically maximize the fit to the S&P 500 observed implied volatility shape in the same vein of the calibration we carried out in subsection 1.5.1.

Black-Scholes

We use numerical values adjusted to our calibrated Heston model:

$$\sigma = \sqrt{v_0}, \quad \mu = -\frac{1}{2}\sigma^2$$

Numerical Results

We now compile prices in table 5.2 for our calibrated models following our previous specifications.

Parameters		Black-Scholes		Heston		Rough Bergomi	
Risk Aversion	Transaction costs	FNN	LSTM	FNN	LSTM	FNN	LSTM
$\lambda = 1$	k=0.00%	1.00629	1.00642	1.01325	1.01325	0.94453	0.94363
	k=0.05%	1.00801	1.00718	1.01513	1.01401	0.94552	0.94443
	k=0.50%	1.00902	1.00902	1.01523	1.01523	0.95219	0.95018
	k=5.00%	1.00902	1.0090	1.01523	1.01523	0.95766	0.95766
$\lambda = 5$	k=0.00%	1.00791	1.00859	1.01551	1.01569	0.95524	0.95196
	k=0.05%	1.00983	1.01734	1.01731	1.01630	0.95671	0.95262
	k=0.50%	1.01659	1.01419	1.02120	1.01956	0.96687	0.95958
	k=5.00%	1.01746	1.01746	1.02120	1.02120	0.99089	0.99090
$\lambda = 10$	k=0.00%	1.0107	1.01083	1.01714	1.0177	0.97453	0.96658
	k=0.05%	1.01148	1.01155	1.01917	1.01871	0.97245	0.96018
	k=0.50%	1.01999	1.01727	1.02461	1.02461	0.98280	0.96819
	k=5.00%	1.02231	1.02231	1.02461	1.02461	1.00634	1.00237
$\lambda = 15$	k=0.00%	1.01095	1.01293	1.01938	1.02565	0.98518	0.97946
	k=0.05%	1.01338	1.01356	1.02014	1.02191	1.01335	0.97436
	k=0.50%	1.02191	1.01909	1.02517	1.02382	1.01335	1.00921
	k=5.00%	1.02464	1.39917	1.02624	1.02624	1.01333	1.01025
Analytical Price(Monte-Carlo)		1.00468		1.01040		0.94430	

Table 5.2: Comparing indifference pricing this time for our three calibrated models under FNN and LSTM against pricing the autocall under standard Monte-Carlo simulations.

In fig. 5.3, we have plotted the hedging for our three calibrated models with $\lambda = 10$ and under LSTM as we found in the previous section that best results were achieved through these specifications.

We observe that Black Scholes and Heston exhibit a similar hedging profile while rough Bergomi seems to display a much smoother profile. This can also be implicitly deduced from the prices we obtained in table 5.2 as Black Scholes and Heston exhibited a similar range of prices while rough Bergomi shows much lower prices. As we remember that the indifference pricing can be perceived as the price of the hedge, we expected this to be reflected in the hedging strategy.

Finally, we have plotted for each model in fig. 5.4 and fig. 5.5 the hedging strategy over two scenarios. Just like before, scenario A describes a realized path that breaches the autocall barrier. In B, the barrier is not breached. Although the realized paths are different between each model, we carefully picked realized paths that are fairly similar.

In scenario A, the hedging strategy profile under the three models looks fairly similar. Although one may note that the hedging ratio is bigger for rough Bergomi which may arise for the calibrations.

In scenario B, the hedging strategy is again somehow similar between Black Scholes and Heston. However for rough Bergomi, the stock price plunges rapidly below the DIP Barrier (0.7) which induces a rapid soars in the hedging ratio.

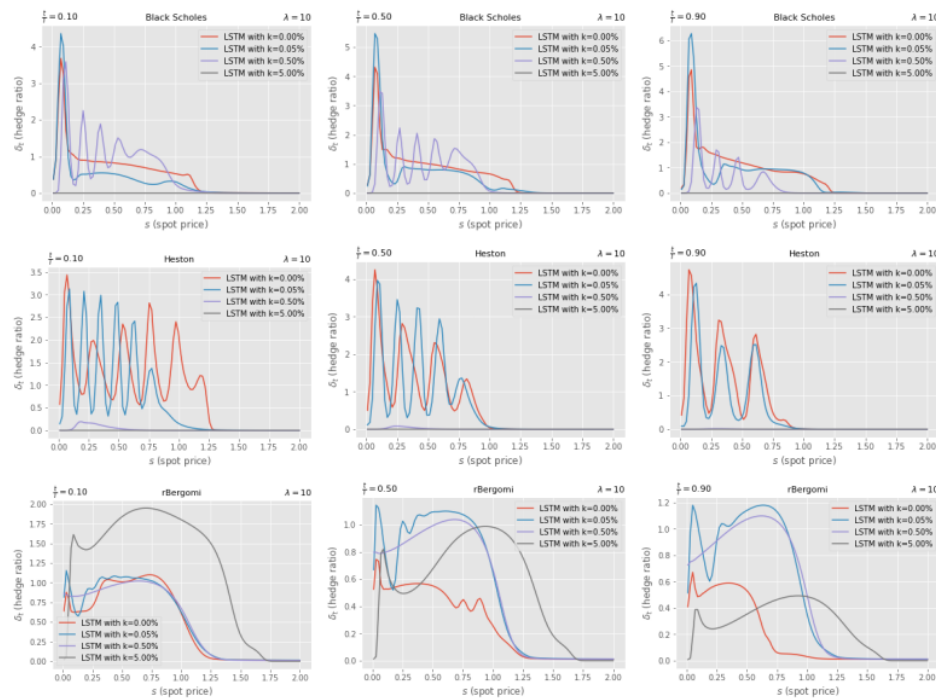


Figure 5.3: Evolution of the deep hedging ratio δ_t under FNN across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 10$ under our three calibrated models.

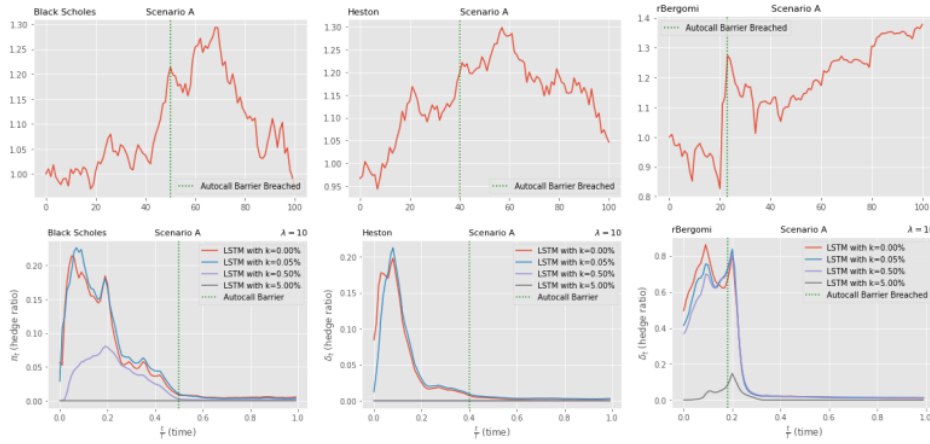


Figure 5.4: In the top row, Realized paths are plotted for the three pricing models over scenario A. In the bottom row, the hedging strategy is implemented respectively for each model under LSTM and $\lambda = 10$.

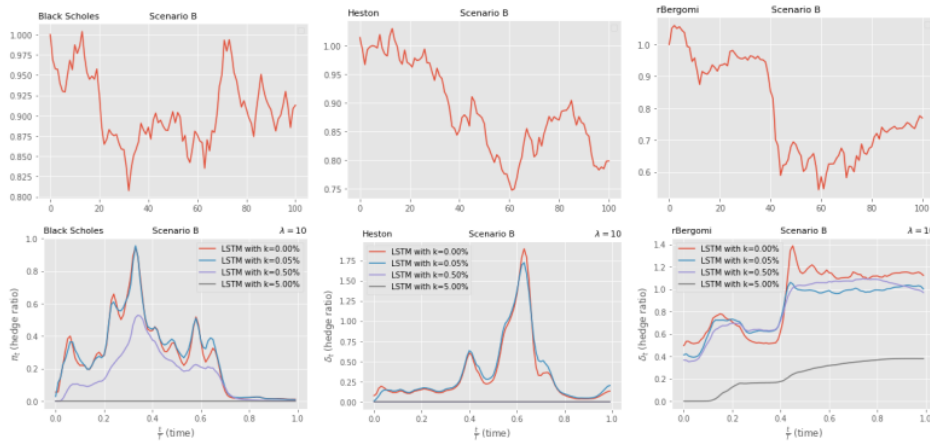


Figure 5.5: In the top row, Realized paths are plotted for the three pricing models over scenario B. In the bottom row, the hedging strategy is implemented respectively for each model under LSTM and $\lambda = 10$.

Conclusion

In this thesis, we have used state-of-the art techniques to tackle an interesting problem: valuing and hedging autocallables. We first looked at rough Bergomi, a very recent and promising model to capture a more realistic stock price profile. We then discussed a very recent and fast simulation method, the hybrid scheme which we used to simulate stock prices. We then used indifference pricing to value the Autocallable according to some criteria such as transaction costs or risk preferences.

In our study, we found that the LSTM is an adapted neural network to value complex path-dependent products such as the autocall. We also compared our calibrated rough Bergomi model against industry benchmarks such as Heston and Black Scholes and found that the hedging strategy profile seemed to be fairly similar, although smoother.

Finally, we conclude by stating that our approach has enabled us to easily price and hedge an instrument over which there is no consensus. We also were able to easily incorporate market imperfections such as transactions costs which are inherent to market. This is today not easily done even with the best analytical models.

In essence, this approach should be model free and data driven. An exciting continuation of this research would be now to apply this framework to real data obtained from traded autocallables which we did not have access to and assess its viability.

Appendix A

Deep Hedging under Neural Networks

A.1 FNN hedging strategy

A.1.1 Evolution of the hedging ratio across varying spot prices

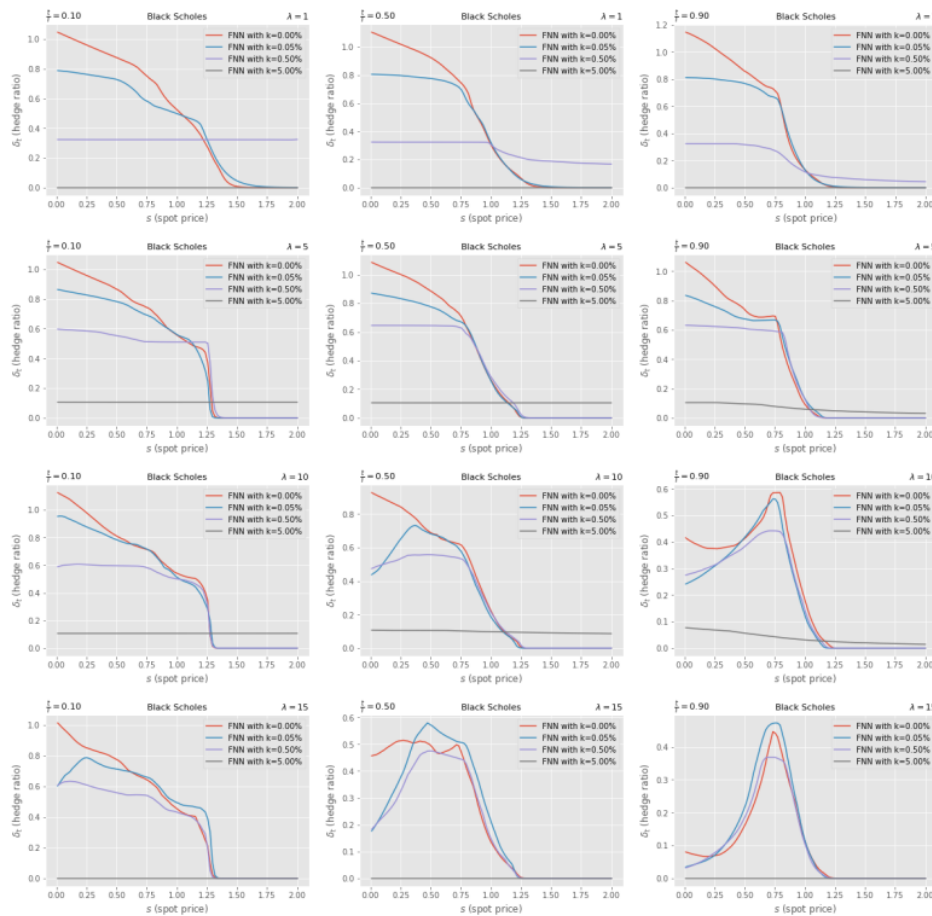


Figure A.1: Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{0.10, 0.50, 0.90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Black-Scholes.

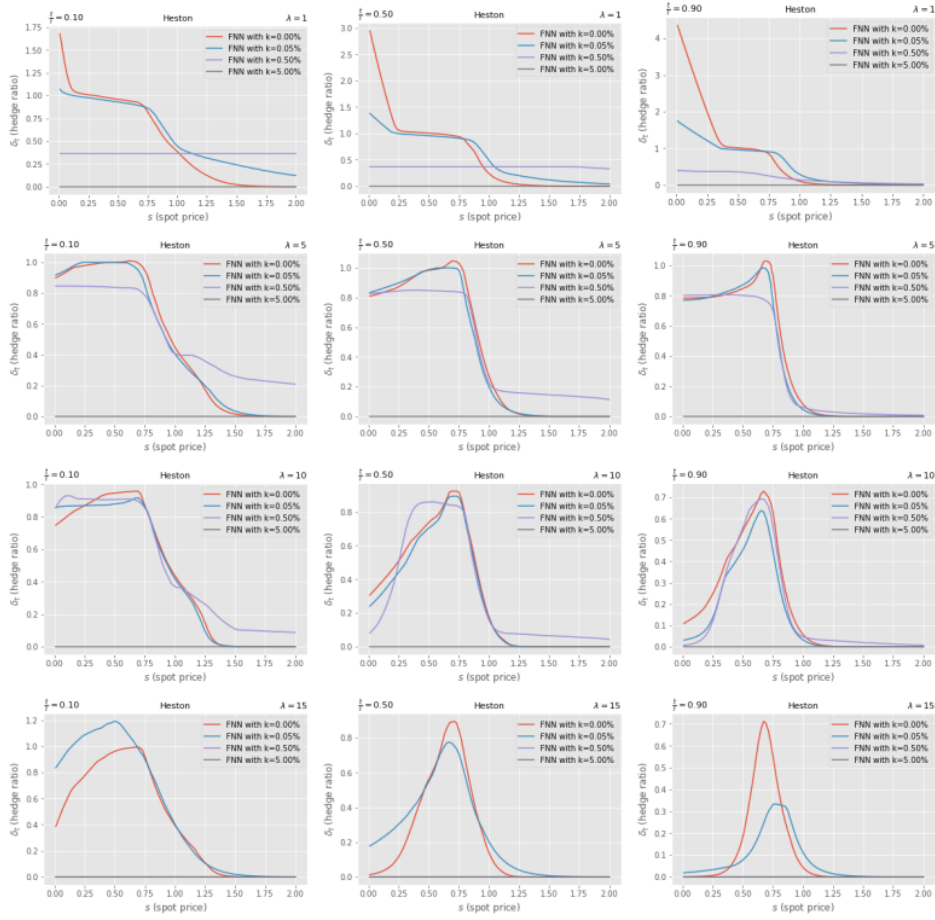


Figure A.2: Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{0.10, 0.50, 0.90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Heston.

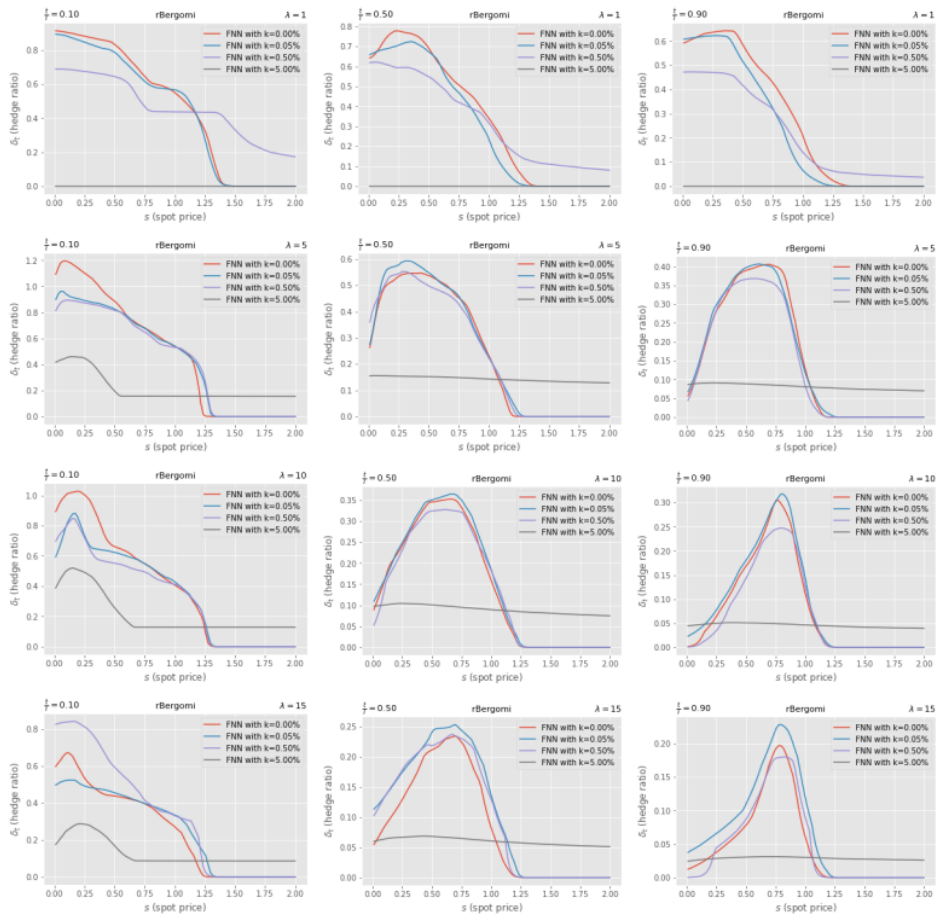


Figure A.3: Evolution of deep hedging ratio δ_t under FNN across different times $t \in \{0.10, 0.50, 0.90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Rough Bergomi.

A.1.2 FNN: Evolution of the hedging ratio over the realized paths of two scenarios

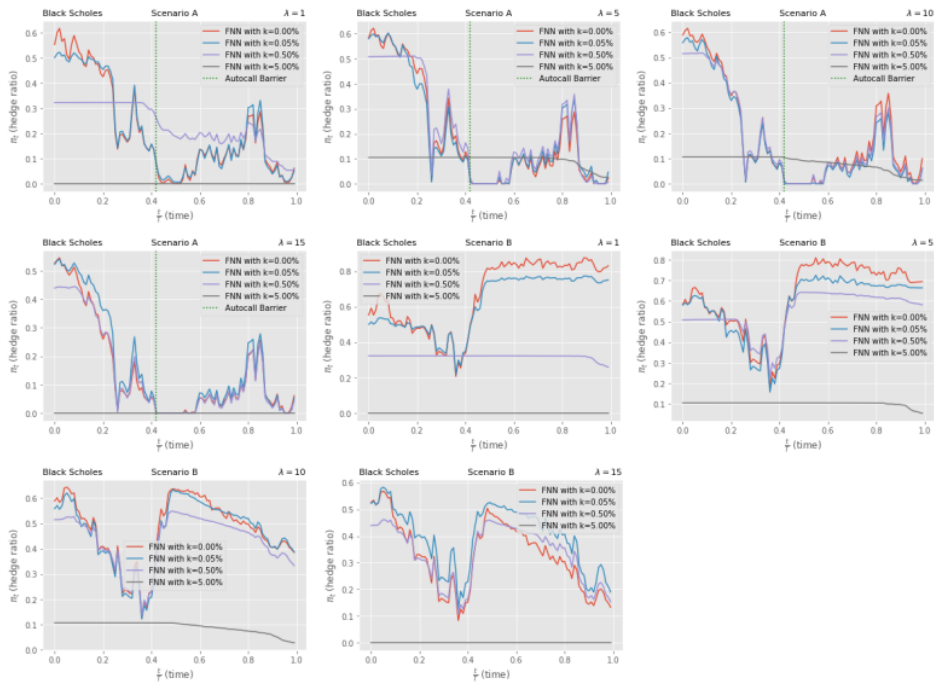


Figure A.4: Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Black-Scholes.

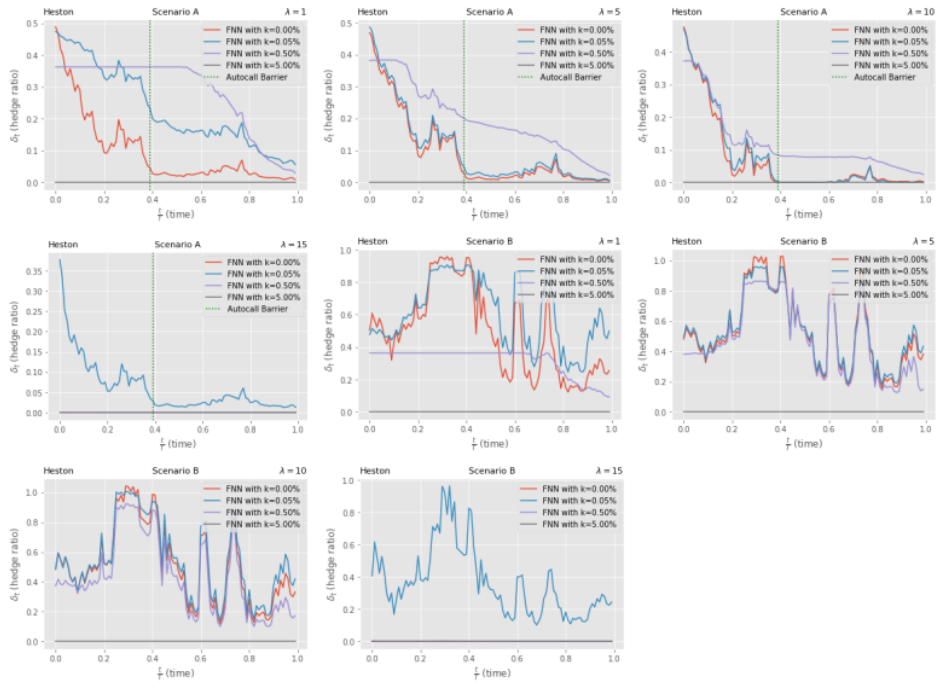


Figure A.5: Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Heston.

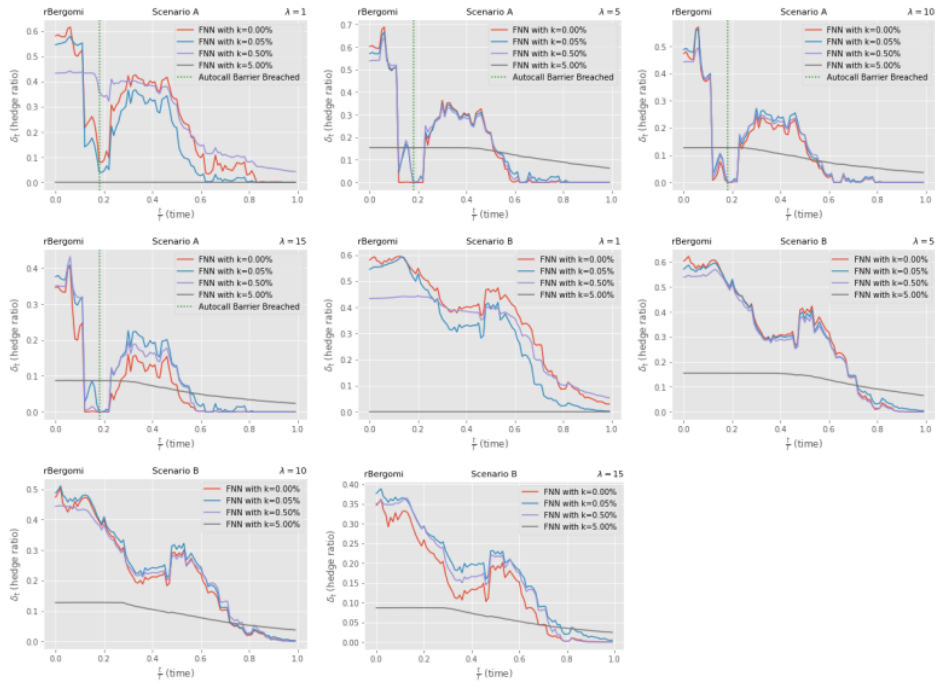


Figure A.6: Evolution of deep hedging ratio δ_t under FNN over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda \in \{1, 5, 10, 15\}$ under Rough-Bergomi.

A.2 LSTM

A.2.1 Evolution of the hedging ratio across varying spot prices

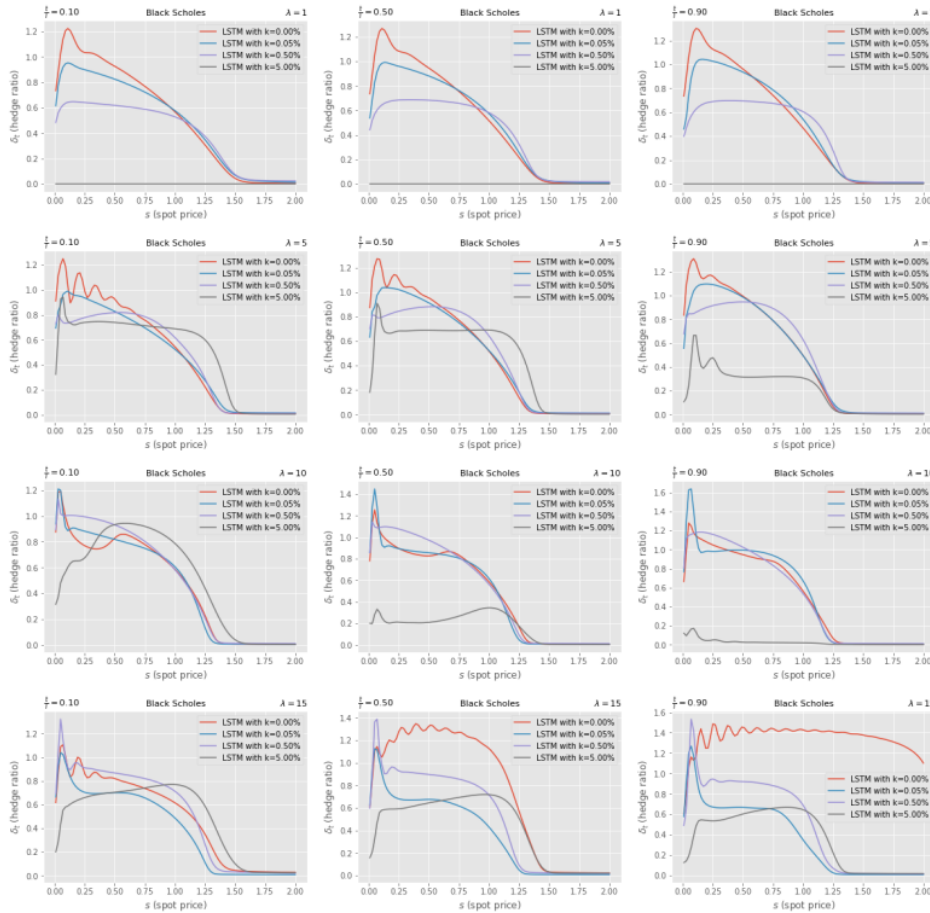


Figure A.7: Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Black-Scholes.

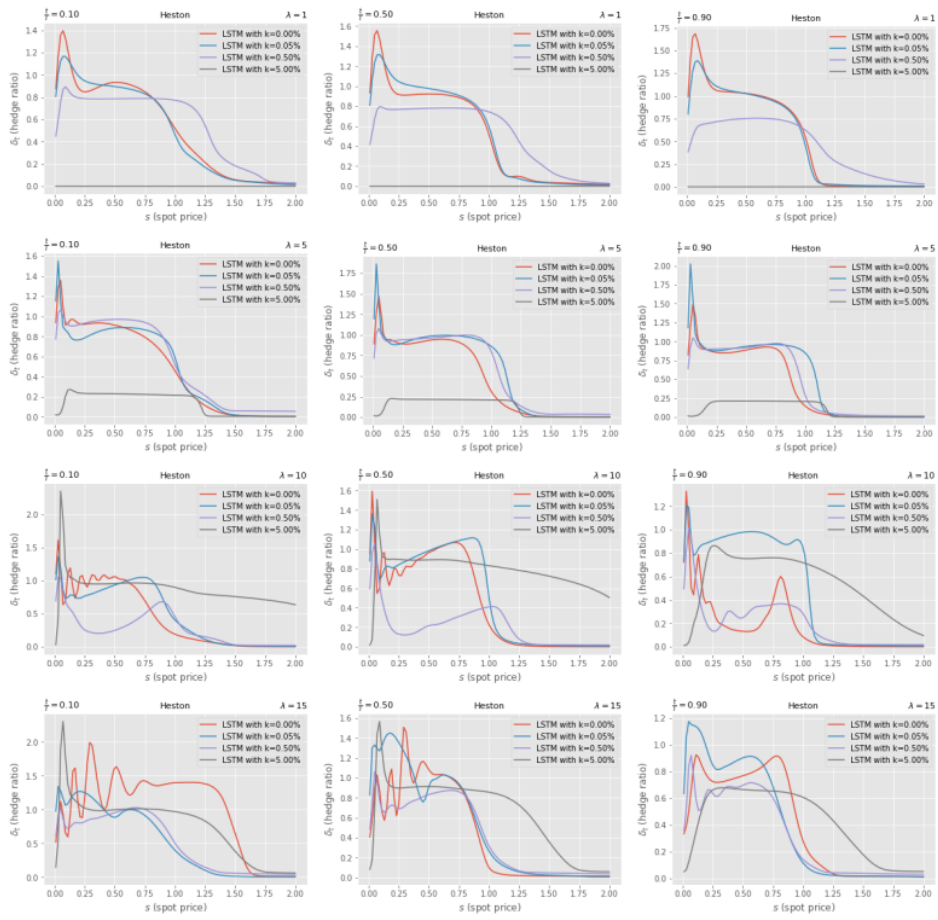


Figure A.8: Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{10, 50, 90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Heston.

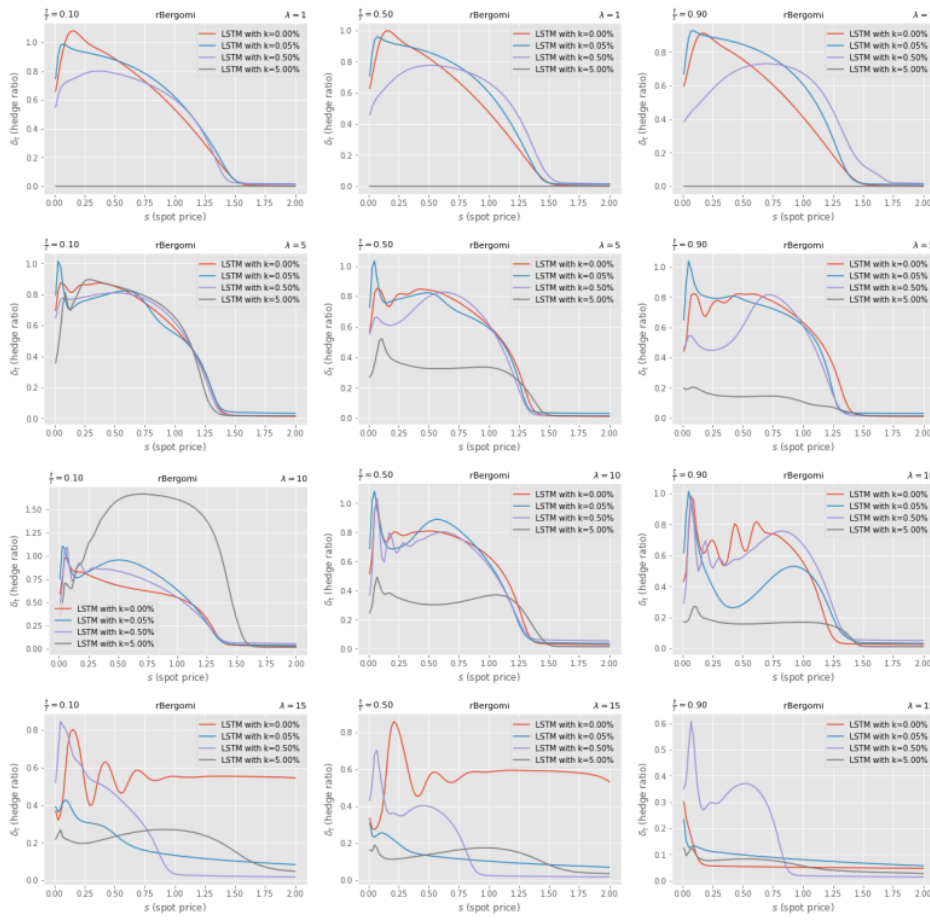


Figure A.9: Evolution of deep hedging ratio δ_t under LSTM across different times $t \in \{0.10, 0.50, 0.90\}$ and transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 1, 5, 10, 15$ under Rough Bergomi.

A.2.2 Evolution of the hedging ratio over the realized paths of two scenarios

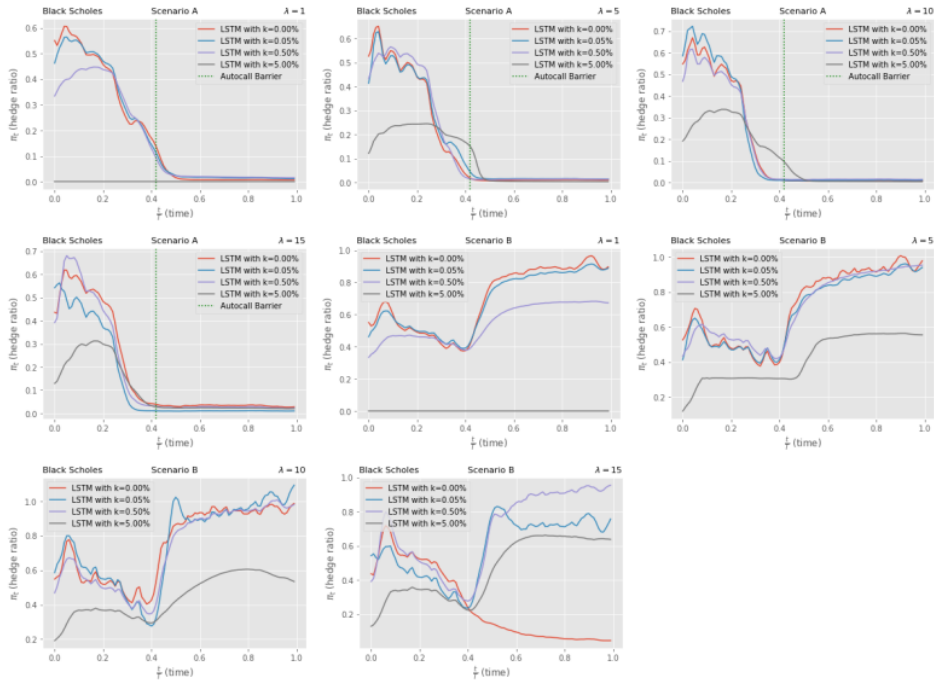


Figure A.10: Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Black-Scholes.

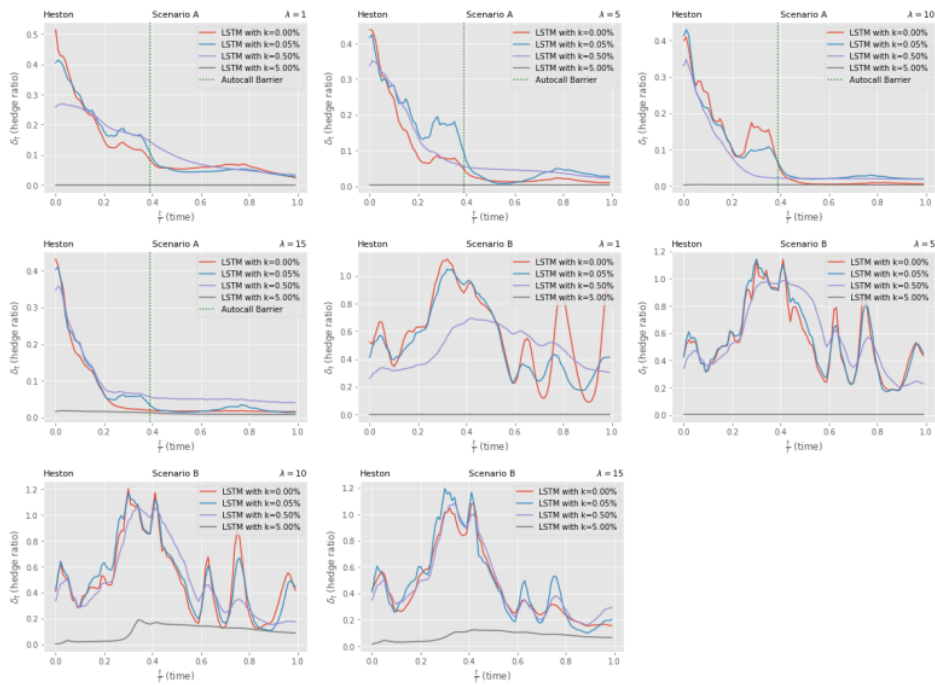


Figure A.11: Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Heston.

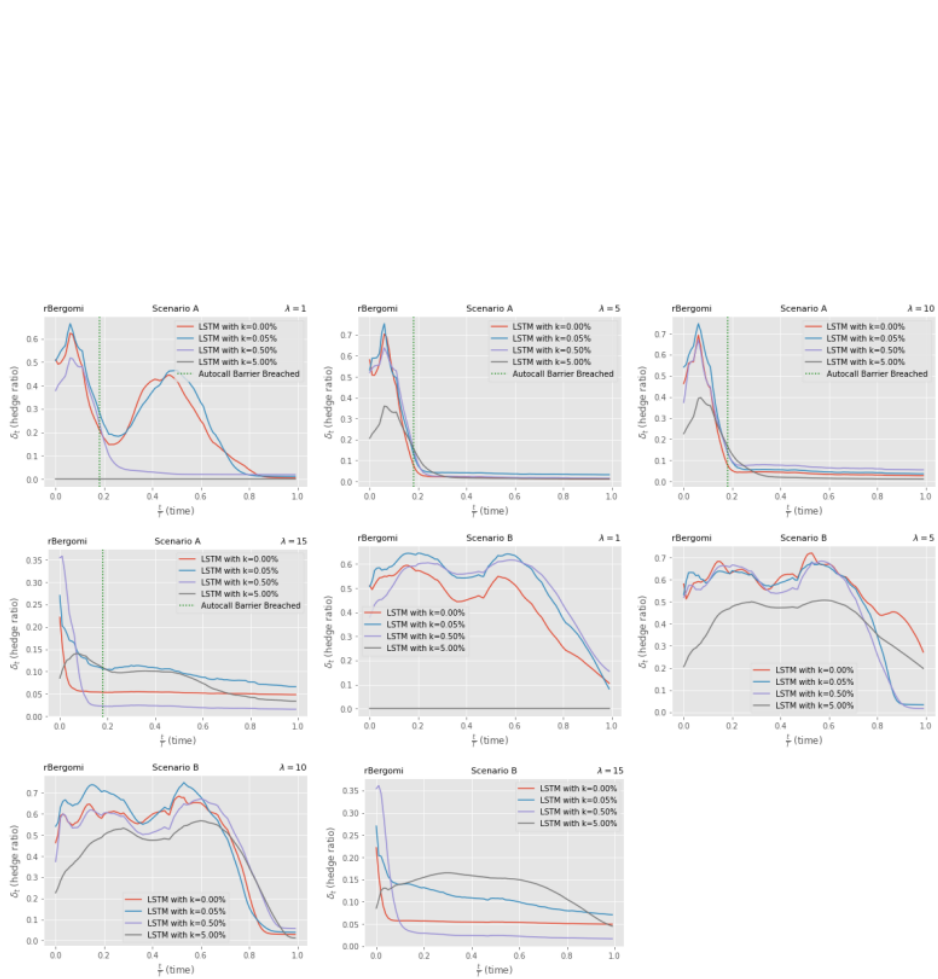


Figure A.12: Evolution of deep hedging ratio δ_t under LSTM over realized paths of Scenario A and B, for transaction costs $k \in \{0.00, 0.05\%, 0.50\%, 5.00\%\}$ for $\lambda = 15$ under Rough-Bergomi.

Bibliography

- [1] Thomas Alm, Bastian Harrach, Daphne Harrach, and Marco Keller. A monte carlo pricing algorithm for autocallables that allows for stable differentiation. *Journal of Computational Finance*, 17(1), 2013.
- [2] Ole E Barndorff-Nielsen and Jürgen Schmiegel. Brownian semistationary processes and volatility/intermittency. *Advanced financial modelling*, 8:1–26, 2009.
- [3] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.
- [4] Mikkel Bennedsen, Asger Lunde, and Mikko S Pakkanen. Hybrid scheme for brownian semistationary processes. *Finance and Stochastics*, 21(4):931–965, 2017.
- [5] Lorenzo Bergomi. Smile dynamics ii. *Available at SSRN 1493308*, 2005.
- [6] Mohamed Bouzoubaa and Adel Osseiran. *Exotic options and hybrids: A guide to structuring, pricing and trading*. John Wiley & Sons, 2010.
- [7] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.
- [8] Peter Carr and Dilip Madan. Towards a theory of volatility trading. *Option Pricing, Interest Rates and Risk Management, Handbooks in Mathematical Finance*, 22(7):458–476, 2001.
- [9] Fabienne Comte and Eric Renault. Long memory in continuous-time stochastic volatility models. *Mathematical finance*, 8(4):291–323, 1998.
- [10] Rama Cont and José Da Fonseca. Dynamics of implied volatility surfaces. *Quantitative finance*, 2(1):45, 2002.
- [11] Mark HA Davis, Vassilios G Panas, and Thaleia Zariphopoulou. European option pricing with transaction costs. *SIAM Journal on Control and Optimization*, 31(2):470–493, 1993.
- [12] Geng Deng, Joshua Mallett, and Craig McCann. Modeling autocallable structured products. In *Derivatives and Hedge Funds*, pages 323–344. Springer, 2016.
- [13] Bruno Dupire et al. Pricing with a smile. *Risk*, 7(1):18–20, 1994.
- [14] Ashkan Eliasy and Justyna Przychodzen. The role of ai in capital structure to enhance corporate funding strategies. *Array*, 6:100017, 2020.
- [15] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative finance*, 18(6):933–949, 2018.
- [16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [17] Igor Halperin. Qlbs: Q-learner in the black-scholes (-merton) worlds. *The Journal of Derivatives*, 28(1):99–122, 2020.
- [18] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993.
- [19] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

- [20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [21] B. Horvath, A. Muguruza Gonzalez, and M. S. Pakkanen. Machine learning and data sciences for financial markets: A guide to contemporary practices, cambridge university press, to appear. *Data-centric methods*. In A. Capponi and C.-A. Lehalle, 2022.
- [22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [23] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [24] Benoit B Mandelbrot and John W Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM review*, 10(4):422–437, 1968.
- [25] Andrea Mazzon and Andrea Pascucci. The forward smile in local-stochastic volatility models. *Journal of Computational Finance*, Forthcoming, 2016.
- [26] Ryan McCrickerd and Mikko S Pakkanen. Turbocharging monte carlo pricing for the rough bergomi model. *Quantitative Finance*, 18(11):1877–1886, 2018.
- [27] Mikko Pakkanen. Lecture notes in deep learning, 2021.
- [28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.
- [29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [30] Filippo Santambrogio. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017.
- [31] Myron Scholes and Fischer Black. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.
- [32] Georgiy Shevchenko. Fractional brownian motion in a nutshell. *arXiv preprint arXiv:1406.1956*, 2014.
- [33] A Elizabeth Whalley and Paul Wilmott. An asymptotic analysis of an optimal hedging model for option pricing with transaction costs. *Mathematical Finance*, 7(3):307–324, 1997.
- [34] Paul Wilmott. Cliquet options and volatility models. *The best of Wilmott*, page 379, 2002.