Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2016

Project Title:	**Refining Black-Scholes Heuristics Using Online Learning**
Student:	**Alexandra Rouhana**
CID:	**00736752**
Course:	**EEE4**
Project Supervisor:	**Dr Andras Gyorgy**
Second Marker:	**Dr Dan Goodman**

Abstract

This project explores the possibility of pricing options using minimax algorithms. The main emphasis of this project is to show how differences in the implementation of this algorithm give results that diverge or converge to the ones given by the Black-Scholes option pricing model. In order to do so, a game between the Investor and the Market was simulated. This game has been divided into multiple trading periods during which both players select in turn their future actions from a predefined set. For the Investor, finding the best move essentially comes to minimize his losses knowing that the adversary, the Market will want to maximize them. By varying the and the quantization used, the results obtained reflect real market dynamics, such as the volatility smile, absent from Black-Scholes model which considers constant volatility for the underlying.

Acknowledgements

First of all, I would like to express my gratitude towards Dr Andras Gyorgy who accepted to supervise this project. I would like to thank him for the enthusiasm he showed as well as the support and help he never failed to provide.

This project would have never been possible without the people I worked with during my internship at the financial software company, Murex. I would like to thank Francois Bouchart for teaching me the essentials of option pricing and for his patience and generosity. I would also like to thank Matthieu Avanthey for welcoming me in the FOX team and giving me this amazing opportunity.

Last but not the least, I would like to thank my family and friends for their support throughout these four past years. I would like to thank Quentin McGaw, a fellow Imperial student, for these very helpful advises and his encouragements. I would like to thank my father, Khalil Rouhana, who was always there to discuss and help me clarify many aspects of this project. I would also like to thank my entire family, including my mother Christiane Rouhana and my brother, Roger Rouhana for the patience, kindness and support they brought me.

Contents

1 Introduction **5**
- 1.1 Introduction to Option Pricing Models 5
- 1.2 A change of perspective 6
- 1.3 Project Objectives 7

2 Overview of Option Pricing Models **8**
- 2.1 The Risk-Neutrality Measure 8
  - 2.1.1 Modelling Volatility 9

3 The minimax hedging game in continuous markets **16**
  - 3.0.1 Background 16

4 Design and Analysis **21**
  - 4.0.1 Building a K-ary Tree 21
  - 4.0.2 CFR and CFR+ 22
  - 4.0.3 Dynamic State Map Algorithm 24
  - 4.0.4 Design of the Minimax Algorithm 24

5 Implementation **27**
  - 5.0.1 Implementing the Code in Python 2.7 27
  - 5.0.2 Quantization 29
- 5.1 Black-Scholes and Implied volatility 30
  - 5.1.1 Back-Scholes closed form equations. 30
  - 5.1.2 Implied Volatility 30

6 Testing the minimax Implementation **32**

7 Results **34**
7.0.1 Effect of parameters *var_budg* and ζ. 34
7.0.2 Memory usage and Timing 36
7.1 Relating the Minimax to the Black-Scholes. 39
7.1.1 Replicating the Black-Scholes 39
7.1.2 Replicating the Smile 43

8 Conclusion **44**
8.1 Evaluation 44
8.2 Further works 45
8.3 Final Remarks 45

Glossary

Option Financial contract that gives the owner the right to buy or sell an asset an agreed price (also known as the strike price) after a given time (maturity).

Strike Price at which the underlying will be bought or sold under the option contract.

Maturity Time after which the underlying will be bought or sold under the option contract.

Underlying Asset that is being sold or bought under the option agreement.

European Option A European option can only be exercised at maturity. The underlying can only be sold or bought at maturity (Hull, 2007).

American Option In the case of an American option, the exercise can be done at any time before the end of the contract (Hull, 2007).

Put Option The put option gives the right to the owner to sell the underlying.

Call Option The call option gives the right to the owner to buy the underlying (Hull, 2007).

"In the money" An option is said to be "In the money" if it is generating a non zero payoff. In the case of a call option this results in the value of the stock being higher than the value of the strike (Hull, 2007).

"Out of the money" An option is said to be "Out of the money" if it is generating a zero payoff. In the case of a call option this is equal to the value of the stock being lower than the value of the strike (Hull, 2007).

Spot Price The Spot price represents the current price of the underlying. citephull.

Characteristic equation The characteristic equation of function is its fourier transform.

Greek Greeks in the context of financial options represent the sensitivity of the option price to changes in underlying parameters.

Chapter 1

Introduction

1.1 Introduction to Option Pricing Models

After the financial crash of 2008, many financial models were questioned and blamed for the crisis (Harford, a). Among these, the very famous Black-Scholes model published in 1973 by Fischer Black and Myron Scholes which establishes the relationship between the value of an Option and the price of its Underlying (Harford, a). From large airline hedging potential losses due to increasing oil prices to hedge funds speculating on foreign exchange rates, options are used by many different players of our economy. But estimating the value of these options can be quite challenging as they depend on an underlying that varies randomly with time.

As will be explained in later sections of this report, the price of an option is determined by its expected final payoff. The owner of an option will encounter profit if the underlying follows his/her expectation but might also face losses if the Spot Price price goes in the opposite direction. The Black-Scholes model suggests the existence of a perfect hedging strategy that eliminates all risk associated to the option. This strategy consists in continuously investing in the underlying by an amount proportional to the change in the option value. The market price of an option is actually determined by the final value of this Black- Scholes hedging strategy .

Like many financial models, the Black-Scholes has been extensively used without real consideration of its underlying assumptions. The Black-Scholes model assumes that the the underlying asset follows a Gaussian motion and has a constant volatility. In the Black-Scholes world, extreme movements in the underlying price are therefore very unlikely. It also assumes that that continuous trade is possible. (Hull, 2007) In other words, this implies that the owner of an option can trade with an unlimited capital and that his transactions are instantaneous.

Over the years, traders have developed heuristics based on the Black-Scholes model but challenging its assumptions (Haug and Taleb, 2010). These new models differ from each other by their way of representing the volatility of the underlying asset. Two models will be explored in this report: the local volatility model which assumes that the volatility is a deterministic function of time and the underlying asset as well as the stochastic volatility model that represents volatility as a random variable. (Ruf, 2012)

1.2 A change of perspective

A growing groups of computer scientists and engineers are applying robotics techniques, such as online learning, to these financial puzzles. The main motivation behind this project is to look at the Black-Scholes hedging problem through the scope of a game between two players: the Investor who is continuously investing money in the underlying and Nature representing the union of all the other market players.

The "hedging game" is defined on n trading periods. In each period, Nature decides on the new value of the underlying and the Investor invests in the asset accordingly.The aim of the Investor is to make the best possible move ensuring that no other decisions could have resulted in a more favorable outcome. The best possible strategy will therefore minimize regret. For the investor, finding the best move essential comes to trying to minimize his losses knowing that the adversary, Nature will want to maximize them.

Previous works have shown that under certain constraints on nature's behavior, the solution of this minimax optimization problem converges to the

one given by the Black-Scholes (Abernethy et al., 2013), as the number of trading periods goes to inifinity. In this project, a Python implementation of the game described in Abernethy et al. (2013) has been implemented.The scope of this project is only limited to European call options.

1.3 Project Objectives

The main ambition of this project is to provide a deeper understanding of the conditions under which the Black-Scholes model is minimax optimal. Another important goal of this project is to evaluate whether or not the minimax algorithm will give results closer to real market dynamics, under different conditions. In order to achieve these objectives, the following deliverables are required:

- A review of the existing option pricing models. The main aim of this chapter is to provide an understanding of real market dynamics and how they have been modeled mathematically.
- An implementation of the minimax algorithm in Python 2.7.
- Run simulations on varying program parameters and discretization to understand under what condition will convergence with Black-Scholes model.

The following report contains the following sections: the next section contains detailed analysis of the existing option pricing models. The third chapter will includes a definition of the implemented minimax problem and a review of the paper (Abernethy et al., 2013). After that, a detailed explanation of the software implementation of the minimax algorithm as well as well as testing and evaluation of the overall algorithm implementation.

Chapter 2

Overview of Option Pricing Models

2.1 The Risk-Neutrality Measure

A *risk-neutral* investor is concerned solely by the expected return of its investment regardless of its risk. An investment with a return of 800$ with a probability of 50% is equivalent to one with a return 400$ with a probability of 100% for this type of investor. This concept of risk-neutrality can be also be applied to option pricing.

The Black-Scholes model shows the existence of a perfect hedging strategy that eliminates all risk associated to the option.(Hull, 2007) By following this assumption, the buyer of an option will be ready to pay a premium for the option that corresponds to the *present value* of the expected payoff regardless of the risk of the asset.(Kjellin and Lovgren, 2006)
Under the Risk-Neutrality measure, the price of an option, $\mathbb{P}$, is given by:

$$V = \exp(-rT)\mathbb{E}(g(S_t)) \quad \text{with} \quad g(S_t) = \max(S_t - K, 0) \tag{2.1}$$

In this project we consider that the interest rate, r is constant over time.The function g represents the payoff of the option in function of the random variable S_t, the price of the underlying.K is strike price of the function.

2.1.1 Modelling Volatility

The Black-Scholes option pricing model.

Under Black-Scholes assumption, the underlying A follows a geometric brownian motion S such that

$$dS_t = \mu S_t + \sigma S_t dW_t \quad with \quad W_t \sim N(0,1) \quad and \quad W_t(0) = 0 \tag{2.2}$$

where W_t is a Brownian motion, and μ ('drift') and σ ('the percentage volatility') do not vary with time (Abernethy et al., 2013). The previous stochastic differential equation admits as a solution,

$$S_t = \exp\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t \tag{2.3}$$

S_t follows a log-normal distribution with,

$$E(S_t) = S_0 e^{\mu t}, Var(S_t) = S_0^2 \exp^{2\mu t}\left(e^{\sigma^2 t} - 1\right), \tag{2.4}$$

In order to hedge the risk of the option, the investor will invest the amount $\Delta(S_t, t)$ in the asset A.The return on this investment will therefore be given by,

$$\Delta(S_t, t)\frac{dS_t}{S(t)} \tag{2.5}$$

The ideal hedging strategy should enable us to cancel out completely the infinitesimal change in the value of the option. If V is the value of the option then,

$$\partial V(S_t, t) = \Delta(S_t, t)\frac{dS_t}{S(t)} \tag{2.6}$$

Since we know that V depends on time and on the random variable S_t and if we neglect the effect of interest rate, we can use Ito's Lemma to express $\partial V(S_t, t)$,

$$\partial V(S_t, t) = \frac{\partial V}{\partial S}dS + \frac{dV}{\partial t}dt + \sigma^2\frac{1}{2}V^2\frac{\partial^2 V}{\partial S^2} \tag{2.7}$$

According to Black-Scholes, the amount invested $\Delta(S_t, t)$ to perfectly hedge the option is given by

$$\Delta(S_t, t) = S_t\frac{\partial V}{\partial S_t} \tag{2.8}$$

Equation 2.6 will therefore become,

$$\partial V(S_t, t) = \frac{\partial V}{\partial S_t} dS_t \tag{2.9}$$

Substituting equation 2.9 into equation 2.7,

$$\frac{dV}{\partial t} dt + \sigma^2 \frac{1}{2} V^2 \frac{\partial^2 V}{\partial S^2} = 0 \tag{2.10}$$

We can see that equation 8 is a non stochastic PDE that can be solved explicitly with the following solution (Abernethy et al., 2013),

$$\boxed{V(S_t, t) = \mathbb{E}_y[g_{EC}(S \exp(Y))] \quad with \quad Y \sim N(\frac{1}{2}\sigma^2(T-t), \sigma^2(T-t))} \tag{2.11}$$

Black-Scholes Equation Closed-Loop Form for European Calls

The Black-Scholes admits the folowing closed form expression for the value of the option:

$$V(S, \sigma, t) = S_0 P_1 - K P_2 \tag{2.12}$$

$$P_1 = N(d_1) \quad \text{and} \quad P_2 = N(d_2) \quad \text{with} \quad N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-x^2/2} \tag{2.13}$$

$$d_1 = \frac{1}{\sigma\sqrt{T-t}} [\ln(\frac{S}{K}) + (\frac{\sigma^2}{2})(T-t)] \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{T-t} \tag{2.14}$$

The Smile Problem.

The implied volatility corresponds to the volatility of the underlying asset given by the 'inverted' Black-Scholes equation given the quoted market value of the option, the quoted value of the underlying,the strike price and the maturity of the option. The Black-Scholes option prices can be expressed by the function $\mathbb{P}(S, T, K, \sigma)$. If $\mathbb{P}$ and § are quoted on the market as $\mathbb{P}_0$ and S_0 then we can solve the following equation to find the implied volatility σ, with $\mathbb{P}$ as the current price of the option, S_0 the spot price of the underlying, T the maturity of the option, K the strike and σ the volatility.

$$V_0 = V(S_0, T, K, \sigma) \tag{2.15}$$

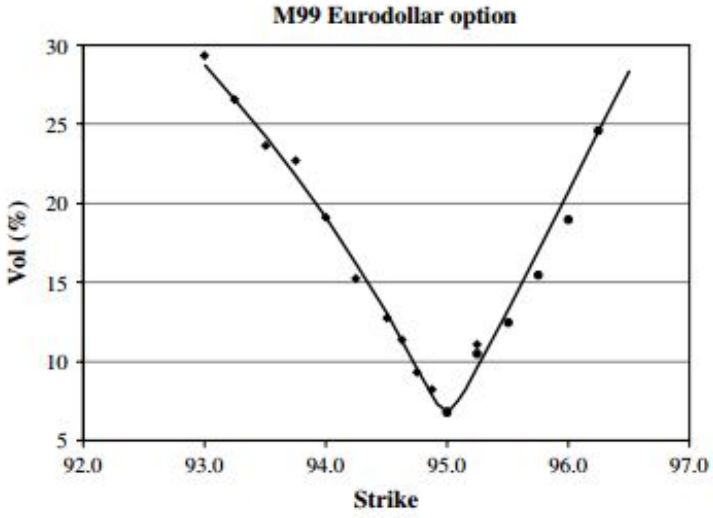

Figure 2.1: Implied volatility for a 1999 Euro-Dollar European option (Hagan et al., 2007).

$$\text{With} \quad \frac{\partial V}{\partial \sigma} > 0 \quad \text{(cf eq. 2.17)}, \tag{2.16}$$

(Hull, 2007, see chapter 17, section 8)
Since the function $V(\sigma)$ is *strictly increasing* as seen in equation 2.19,there will be a unique solution for σ.

In theory, the volatility of the underlying is a constant asset. In reality, the implied volatility will vary for options of different prices, maturity and strike (Hagan et al., 2007). Figure 1 shows the implied volatility expressed in terms of the strike for a 1999 Euro-Dollar European option. The implied volatility reached a minimum for a strike at 95 and increases for higher and lower strikes. This specific shape of the implied volatility curve is known as the *volatility smile*.

The existence of the *smile* shows that the Black-Scholes model is built on the false assumption that the volatility of the underlying asset is constant.The models described in the next sections of this report have been developed taking into consideration the *smile* dynamics.

The Local Volatility Model

The local volatility model was developed in 1994 by Bruno Dupire. As in the Black-Scholes model it assumes that the underlying follows a diffusion process with a Brownian motion. However, it also supposes that the volatility varies with time. Actually, volatility is defined as a deterministic function of the stock S and time t.

$$dS_t = \mu S_t \partial t + \sigma(S_t, t) S_t dW_t \tag{2.17}$$

The Local Volatility function can actually be represented by a 2D surface depending on the value of S and t.
The volatility is actually computed from real market data (Ye, 2011).Indeed, the function is built from real market price for various maturity and strikes. $\sigma(t, S)$ is then obtained by interpolation. This is shown by equation 2.18,

$$\sigma(T, K)^2 = \frac{1}{K^2} \frac{2\frac{\partial \mathbb{P}}{\partial T} + 2\mu K \frac{\partial \mathbb{P}}{\partial K}}{\frac{\partial^2 \mathbb{P}}{\partial K^2}} \tag{2.18}$$

As can be seen, the Dupire local volatility requires the derivatives $\frac{\partial p}{\partial T}$, $\frac{\partial p}{\partial K}$. However, since only prices of options for specific strikes and maturity are available on the market, the surface $\mathbb{P}(T, K)$ is not continuous.These derivatives have to be obtained through numerical differentiation.However, equation 2.18 shows that the second degree derivative of the price in function of strike is scaled by the square of the strike. Therefore, small errors in the estimation of the derivative in denominator will therefore result in large error in the local volatility (van der Kamp, 2009), which is a great source of instability.

The local volatility model gives prices which correspond to the actual market prices. However, it does not follow the market dynamics. According to the local volatility, the price of the asset decreases as the volatility increases and increases as the volatility decreases. In reality, the implied volatility of the asset move in the same direction as its price (Dupire, 1994). Stochastic volatility models are actually better suited to model market dynamics. (Hagan et al., 2007)(Berestycki et al., 2004).

Stochastic Volatility Models

Stochastic Volatility models assume that the volatility of the underlying asset follows, similarly to the underlying asset, a Brownian diffusion equation. Several types of stochastic volatility models exist(Berestycki et al., 2004).In this project, only the Heston model will be explored. According to this model, the volatility of the underlying follows a Cox, Ingersoll and Ross diffusion (Kjellin and Lovgren, 2006)(Heston, 1993).

$$\partial S_t = \mu S_t \partial t + \sqrt{\mathcal{V}_t} S_t \partial W_t^{(1)} \tag{2.19}$$

$$\partial \mathcal{V}_t = \kappa(\theta - \mathcal{V}_t)\partial t + \eta\sqrt{\mathcal{V}_t}\partial W_t^{(2)} \tag{2.20}$$

$$\tag{2.21}$$

κ is the speed at which the variance $\mathcal{V}_t$ converges to its long run average θ if $\kappa > 0$,η corresponds to the volatility of the variance. The Heston model has two main advantages:

First Similarly to the Black-Scholes, it admits a closed form formula for pricing of European Options (Heston, 1993).It is therefore easier to implement.

Second It reproduces the smile/skewness dynamics available on the market (Agnieszka Janek, 2010).

Heston Model Closed Form Formula

Very similarly to the Black-Scholes, the Heston model follows the probability differential equation (2.7) which assumes the existence of a risk-less hedging strategy. The proposed solution to differential equation 2.7 has the following form (Heston, 1993):

$$\mathbb{P}(S, \sigma, t) = S_0 P_1 - K \exp(-rT) P_2 \tag{2.22}$$

With S_0 being the spot price at expiry, P_1 represents the option delta [1] and P_2 the probability that the option expires in the money. Determining a closed form for P_1 and P_2 cannot be done directly unlike in the case of the Black-Scholes model.

[1]The delta of an option corresponds to the derivative $\frac{\partial V}{\partial S}$. It shows how the price of the option varies with the value of the underlying

However, they both admit a Characteristic equation that satisfies the differential equation 2.7. The characteristic equation is given by the following laborious equations, (Heston, 1993):

$$\phi_{ln(S_t)}(w) = \exp[(C(t,w))\theta + D(t,w)V_0 + iw\ln(S_0\exp(rt))]$$
$$C(t,w) = a\left[r_- t - \frac{2}{\eta^2}\ln\left(\frac{1-g\exp(-ht)}{1-g}\right)\right]$$
$$D(t,w) = r_-\frac{1-\exp(-ht)}{1-g\exp(-ht)}$$
$$r_\pm = \frac{\beta \pm h}{\eta^2} \quad ; \quad h = \sqrt{\beta^2 - 4\alpha\gamma}$$
$$g = \frac{r_-}{r_+}$$
$$\alpha = -\frac{w^2}{2} - \frac{iw}{2} \quad ; \quad \beta = \kappa - \rho\eta iw \quad ; \quad \gamma = \frac{\eta^2}{2}$$

The probabilities P_1 and P_2 can be found by taking the inverse Fourier transform of the characteristic function.

$$P_1 = \frac{1}{2} + \frac{1}{\pi}\int_0^{\inf} Re\left[\frac{\exp(-iw\ln(K))\phi_{\ln(S)}(w-i)}{iw\phi_{\ln(S)}(-i)}\right]\partial w$$
$$P_2 = \frac{1}{2} + \frac{1}{\pi}\int_0^{\inf} Re\left[\frac{\exp(-iw\ln(K))\phi_{\ln(S)}(w)}{iw}\right]\partial w$$

Figure 2.2: This table compares the two models seen previously according to three criteria: ease of implementation, market dynamics and real market data.

	Local Volatility	**Stochastic Volatility**
Ease of Implementation	*The local volatility is not easy to implement since it requires interpolations and extrapolations from real market data and there is no closed form. Since the Local Volatility model does not admit any closed loop, it has to be implemented using Monte Carlo simulations.*	*The stochastic volatility model admits closed form formulas for European calls. It is therefore easier to implement.*
Reflects market dynamics.	*The local volatility model does not reflect market dynamics.*	*The stochastic volatility model reflects very well the smile dynamics.*
Matches real market data.	*Since the local volatlity function is built from real data, it matches very well real market prices.*	*This model does not match real market data as well as the local volatility.*

Summary

Option pricing models have become more realistic in reflecting the randomness of the underlying. This evolution has been done by modifying the Black-Scholes model and redefining the properties of the underlying's volatility.

The Black-Scholes model does not reflect real market dynamics, such as the smile. Accrording to Haug and Taleb (2010), this is due to the effect of time discretization and the impossibility of instantaneous trading in real life which contradicts one of Black-Scholes main assumption (cf. Intro.).Evaluating whether or not the effect of discretization on the minimax value could reflect smile dynamics is one of the ambition of this project.

Chapter 3

The minimax hedging game in continuous markets

3.0.1 Background

Online learning techniques can be applied to option hedging in order to find the most optimal strategy that can be used by the owner of the option. Our problem can be seen as a problem between two players: Nature and the Investor. The game is divided into n trading periods that stop at maturity T and each period i will follow the following sequence:

- Investor invests Δ_i dollars in the underlying asset
- Nature selects price changes r_i and updates the price of the asset to $S_{i-1}(1 + r_i)$
- Investor receives profit $\Delta_i r_i$

If $g(S_t)$ is the payoff of the option at time t given the price S_t of the underlying, we know that the final cost of the option will be given by its final payoff,

$$g(S_T) = g(So \prod_1^n (1 + r_i)) \tag{3.1}$$

The aim of the investor is therefore to minimize the following function with respect to Δ_i ,

$$g(So \prod_1^n (1 + r_i)) - \Delta_i r_i \tag{3.2}$$

The role of Nature will be to maximize this function.

Defining the minimax problem

The minimax problem solved has the following constraints:

- TotVarConstraint: The total price fluctuation is bounded by a constant c such that $\sum_{i=1}^{n} {r_i}^2 \leq c$
- JumpConstraint: Every single jump is bounded by the constant ζ such that $|ri| \leq \zeta$

This method does not assume that the asset follows a geometric brownian motion. However, according to the constraints of our minimax problem, the total price fluctuation (TotVarConstraint) which is an estimator of the variance is bounded. The JumpConstraint indicates that the price of the asset cannot suffer from extreme fluctuations. These two constraints are in line with the Black-Scholes framework which assumes that the undelying is very unlikely to suffer from very large price fluctuations over time and that the standard deviation will never vary over time.

The value of our game will therefore be expressed as a function of ζ, c and the underlying price as follows:

$$V(S, c, n, \zeta) = \min_{\Delta} \max_{r} -\Delta r + V(S(1 + r_m), c - r^2, n - 1, \zeta) \tag{3.3}$$

According to Abernethy et al. (2013), as n goes to infinity, the solution to this minimax problem will tend towards the one given by the Black-Scholes Hedging strategy.

Theorem 1.

$$\boxed{\lim_{n\to\infty} V(S, c, n, \zeta) = U(S, c) = \mathbb{E}_y[g_{E_C}(S \exp(Y))]} \tag{3.4}$$

$$\textit{with} \quad Y \sim N\big(\frac{1}{2}c^2(T-t), c^2(T-t)\big) \tag{3.5}$$

It has therefore been proven that the worst case behavior of the underlying, with the given constraints, converges to a log-normal distribution.Theorem 1 therefore confirms the BS assumptions (Hull, 2007).

The proof of theorem 1 is done by showing that the upper and lower bound of V as n tends to infinity corresponds to the BS solution given by $U(S, c)$. The following theorem shows that $U(S, c)$ is the lower bound of $V(S, c, n)$.

Theorem 2.

$$\lim_{n\to\infty} V(S, c, n, \zeta) \geq U(S, c) \tag{3.6}$$

(Abernethy et al., 2013).

Lemma 1.

$$\lim_{n\to\infty} \prod_{1}^{n}(1 + r_i) = G(c) \quad \textit{with } \log(G(c)) \sim N(-\frac{1}{2}c, c) \tag{3.7}$$

According to Lemma 1,

$$\lim_{n\to\infty} \mathbb{E}\big(g(S_0 \prod_{1}^{n}(1 + r_i))\big) = \mathbb{E}(SG(c)) = U(S, c) \tag{3.8}$$

In order to prove theorem 2, lemma 1 is required. This lemma is proved in Appenfix A of the paper (Abernethy et al., 2013)

Proof.

$$V(S, c, n, \zeta) = \min_{\Delta_1} \max_{r_1} ... \min_{\Delta_n} \max_{r_n} \quad g(So \prod_{1}^{n}(1 + r_i)) - \sum_{i=1}^{n} \Delta_i r_i$$

$$V(S, c, n, \zeta) \geq \min_{\Delta_1} \max_{r_1} ... \min_{\Delta_n} \quad \mathbb{E}\big(g(So \prod_{1}^{n}(1 + r_i)) - \sum_{i=1}^{n} \Delta_i r_i\big)$$

$$= \mathbb{E}(g(So \prod_{i=1}^{n}(1 + r_i)) = \mathbb{E}(g(SG(c))$$

□

The following theorem shows that $U(S, c)$ is the upper bound of $V(S, c, n)$.

Theorem 3.

$$\lim_{n\to\infty} V(S, c, n, \zeta) \leq U(S, c) \tag{3.9}$$

Proof. In order to prove (4), the following inequality is demonstrated

$$V(S, c, n, \zeta) \leq U(S, c) + (18c + \frac{8}{\sqrt{2\pi}})K\zeta^{1/4} \tag{3.10}$$

The payoff g is assumed to be Lipschitz continuous in K (Abernethy et al., 2013). A Taylor expansion is performed of U around $U(S, c)$, at time m for some $m \geq 1$ for which $V(S, c, m)$ is a good approximation of U(S,c). At time m, the value of the stock becomes $S(1 + r_m)$ and the TotVarConstraint becomes $c - r_m^2$

$$\begin{aligned}
&U(S + Sr_m; c - r_m^2) \\
&= U(S, c) + r_m S\frac{\partial U(S, c)}{\partial S} - r_m^2\frac{\partial U(S, c)}{\partial c} + \frac{1}{2}r_m^2 S^2\frac{\partial U(S, c)}{\partial S}^2 + O(r_m^3) \\
&= U(S, c) + r_m S\frac{\partial U(S, c)}{\partial S} + O(r_m^3)
\end{aligned}$$

The last line can be derived under the assumption that the Investor is investing an amount given by

$$\Delta(S, c) = S\frac{\partial U(S, c)}{\partial S} \tag{3.11}$$

where c is the value of the remaining fluctuation budget (see section 2.1.2 The Black-Scholes option pricing model). If we go back to the definition of the value of Minimax game,

$$\begin{aligned}
&V(S, c, m, \zeta) = \min_{\Delta}\max_{r_m} -\Delta r_m + V(S(1 + r_m), c - r^m, m - 1, \zeta) \\
&\leq \max_{r_m} -r_m S\frac{\partial U(S, c)}{\partial S} + V(S(1 + r_m), c - r^m, m - 1, \zeta) \\
&= \max_{r_m} -r_m S\frac{\partial U(S, c)}{\partial S} + U(S(1 + r_m), c - r^m, m - 1, \zeta) + (approx.terms) \\
&= U(S, c) + O(r_m^3) + (\text{approx. terms})
\end{aligned}$$

From appendix A in Abernethy et al. (2013), if we consider α_n as the sequence containing the cumulative error of each trading period, then

$$V(S, c, n, \zeta) \leq U(S, c) + \alpha_n$$
$$\alpha_n \leq (18c + \frac{8}{\sqrt{2\pi}})LK\zeta^{1/4}$$

with α_n going to zero as n goes to infinity (Abernethy et al., 2013). □

Summary

This section has covered the main mathematical theorems in the paper that serves as the starting point of the minimax implementation of this project. The main result we would like to validate through simulations is the following: *the minimax value will tend to the black-scholes price as the number of trading period becomes large enough.* The following section will discuss the software implementation of the minimax problem formulated in equation 3.3.

Chapter 4

Design and Analysis

4.0.1 Building a K-ary Tree

In order to find the optimal strategy for the investor, it is necessary to simulate a certain number of scenarios, by varying both the market fluctuations, r_i and the investments of the investor, Δ_i and observing the payoff and the variance budget. The most explicit way of solving this problem is to create a tree simulating all the possible combinations of market fluctuations and investments.Such a structure is shown in figure 4.1. However, as the following example shows, such a tree would be impossible to implement.

In the following minimalist case, the investor owns a 1 month option on 1 stock at 25$. If we assume that the amount invested is bounded between -25$ and 25$. By adopting a discretization 2$ each possible investment, the number of possible investments is equal to 25. If we consider the set Δ containing all possible investments, we can write $|\Delta| = 25$. Equally, if the maximum jump allowed,ζ, in the price of the stocks is 5$, the market will have 5 possibilities to choose from. In that case, if the set containing all possible market jumps is called r, $|r| = 10$ If the option has 1 month maturity with 2 days interval between each trading the number of trading periods T is given by T=15. One integer has a size of 4 Bytes. Considering that the RAM has 4GB (2^{32} bytes) of storage area, the maximum number of nodes than can be stored and computed by the algorithm is 2^{30} bytes.The resulting space complexity is $O((|\Delta||R|)^T) = (250)^{15} > 2^{30}$. Such a a tree would therefore be impossible to compute due to lack of available memory storage.

Several alternative techniques have been considered for the implementation of the Minimax algorithm. The following section gives an overview of the different possibilities and justifies the choices made for the actual implementation.

4.0.2 CFR and CFR+

Defining CFR/CFR+ in the general context.

Counter factual regret minimization (CFR) is an algorithm where for each information set $\mathcal{I}$ (or state), the player chooses the most optimal action by solving a regret-minimizing algorithm. Each game can be seen as a tree, where each terminal leaf is denoted by z. For each information set $\mathcal{I}$ and instant time t, the probability σ of choosing an action a minimizing regret (Michael Bowling), is given by

$$\sigma_i^t(\mathcal{I}, a) = \begin{cases} \frac{1}{|A|} & \text{if} \quad \sum_{a' \in A} R_i^{t-1}(\mathcal{I}, a,) < 0 \\ \frac{R_i^{t-1}(I,a)}{\sum_{a' \in A} R_i^{t-1}(I,a')} & \text{otherwise} \end{cases}$$

In this case, $\mathcal{A}$ represents the set of possible actions the player can chose from. The probability σ is therefore determined by the regret of a up to $t-1$, divided by the sum of regrets for all actions in $\mathcal{A}$.

The counter factual regret is computed in the following manner (Michael Bowling),

$$R_i^{t-1}(I, a) \equiv \sum_{j=1}^{t-1} v_i^{\sigma}(a) - \sum_{j=1}^{t-1} \sum_{a' \in A} v_i^{\sigma}(a')\sigma_i(a') \tag{4.1}$$

This represents the accumulated counter factual value that arises from choosing action a rather than the expected value from the strategy actually used. The counter factual value v_i is defined below,

$$v_i^{\sigma}(I, a) = \sum_{z \in Z} u_i(z)\pi_{-i}^{\sigma}(z)\pi_i^{\sigma}(z) \tag{4.2}$$

The probability of player i reaching the leaf z with strategy profile σ is given by $\pi_i^{\sigma}(z)$. Similarly, $\pi_{-i}^{\sigma}(z)$ represents the contribution of the other player to reaching this leaf. The utility u_i is the payoff of the strategy.

CFR+ is an improved version of the CFR. In the case of CFR+, the counter factual regret is thresholded at zero whenever it is negative. This enables a more efficient algorithm since, if the regret is positive at instant t it will not be canceled later on by negative regret and will contribute more to the overall accumulated regret (Michael Bowling).

CFR+ for option pricing.

CFR+ can have many applications including finding optimal strategy for poker games (Michael Bowling). However, it is possible to map this problem to the one tackled here. Indeed, in the scope of this project, the CFR+ algorithm would include two opposing players, the investor and the market. In the case of the investor, the set of actions A is equal to the previously defined set Δ and for the market, A would be equal to r. Each time t will correspond to a trading period. The final payoff u_i is the price of the option determined by the accumulated negative payoff determined as follow,

$$\sum_{i=0}^{t} -\Delta_i r_i \tag{4.3}$$

Each Information set would defined by one information only, the variance budget defined below. The variance budget will limit the set A of the market player to actions guaranteeing a strictly positive variance budget.

$$\sum_{i=0}^{t} c - {r_i}^2 \tag{4.4}$$

The CFR and CFR+ algorithms are adapted to the problem faced. However, they require a large development time which could not be afforded in the time line of this project. An algorithm which is simpler to implement is described in the next section.

4.0.3 Dynamic State Map Algorithm

Building the state map

Similarly to the CFR+ algorithm, this implementation admits a state description which reduces greatly the number of redundant states (see figure 4.2). Each state at trading period n contains the following information:

- The variance budget up to n, determined in 3.4.
- The negative payoff of the strategy up to n, given in 3.3.

At each trading period, each state includes a unique combination of these two elements. If we consider the example introduces in 1.1.1, the resulting time complexity becomes linear, $O(\Delta * R * S * T) = (250) * 15 < 2^{30}$. The following pseudo-code has been implemented in order to build the state map:

Algorithm 1: Building State Map

```
for depth in range(1,T) do
    for state in depth-1 do
        for r_i in r do
            for Δ_i in Δ do
                if c-r_i^2 > 0 then
                    var_budg = state → var_budg − r_i^2;
                    payoff = state → payoff − d_i * r_i;
                    new  state = State(c, payoff);
                    if new_state not in depth then
                        add new_state to depth
```

4.0.4 Design of the Minimax Algorithm

The minimax algorithm follows a backward path and tries to recreate the tree from the given state map (see figure 4.1 and 4.2). The algorithm starts at the last depth of the state map. It then finds the states in these depth with the same originating state and the same previous Δ investment and adds them

to a list called *same_delta*. This list is computed for each possible Δ.The maximum of this list *same_delta* is found and added to another list called m. After all the possible Δ investments have been considered,the minimum of m is calculated. This minimum then becomes the utility of originating state or "parent" in the previous depth. The procedure is repeated until the root of the tree is reached. For more clarity, the pseudo-code of the algorithm is included below.

Algorithm 2: Minimax Procedure

```
depth = last trading period (Initialization)
while depth > 0 do
    for each parent in depth-1 do
        Find in depth states with same parent;
        Add to same_parent list
        same_parent=empty;
        for each state in same_parent do
            same_delta = empty;
            for each Δ do
                Find states with same Δ ;
                Add to list same_delta;
                Get maximum of same_delta;
                Add maximum to list m;
            Find minimum of m;
            Set utility of parent state to minimum of m ;
    depth = depth-1;
```

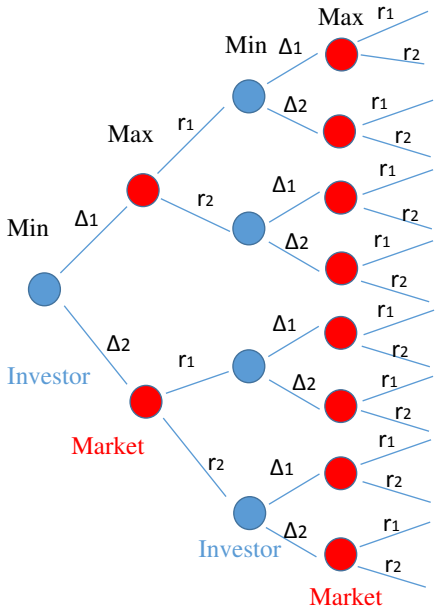

Figure 4.1: Minimax Game Tree between the Market and the Investor.

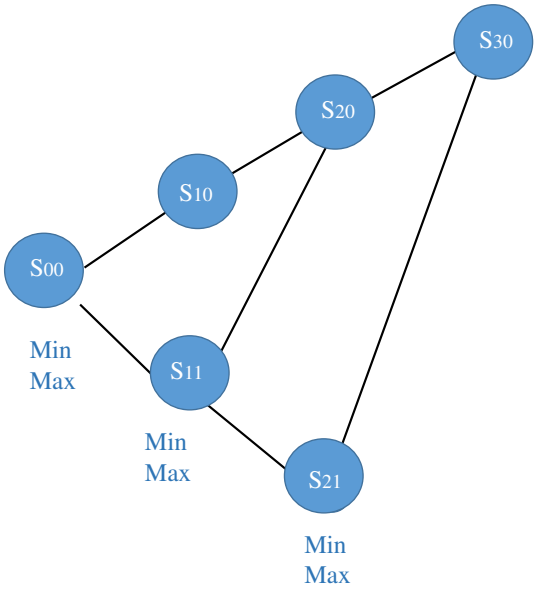

Figure 4.2: State Map reduction of the game.

Chapter 5

Implementation

5.0.1 Implementing the Code in Python 2.7

The Algorithm has been written in Python 2.7. This programming language has been chosen for its ease of use (student had prior experience in this language) as well as the availability of very efficient mathematical libraries such as *numpy, scipy* and *matplotlib* that were very useful to the implementation of the Black-Scholes model as well as the implied volatility calculation.

Python also offers a large variety of powerful data structures and classes. The state map (cf. algorithm 1) is stored in a large list of n sets. The implementation of sets has enabled to reduce the number of computation greatly. Indeed, sets eliminate duplicate elements automatically. This allows to delete redundancies without the computational cost of a "for loop" checking whether or not the state exists in this depth. Sets are based on hash tables.A table of size k with n keys has $O(1+n/k)$ complexity which averages to $O(1)$. Finding an element in a python set has therefore a time complexity of $O(1)$ (Harford, b) on average while the same operation performed with a "for loop" has $O(n)$ complexity.

Python presents another advantage. Classes and object oriented programming can be implemented with ease. In the minimax code, a class called *State* was created in order to store information for each state. Each set contains an object of a class called State that includes the following information:

- The variance budget defined in 3.4 which is the first State Information

,

- The utility defined in 3.3 which is the second State Information ,
- The last delta action taken by the investor before reaching the state, which is a necessary additional Information to recreate the tree and solve the minimax problem (see Algortihm 2),
- An ID that corresponds to the parent of the state ,
- A boolean that indicates whether or not the state has a child.

In order to eliminate states with the same utility and variance budget (but different in term of delta action,parent id and "with child" boolean) it was necessary to overwrite the hash properties, equality properties within the class. This can be done with the following lines of codes,

Listing 5.1: Overwrite set properties in class object

```
1  class State(object):
2      def __init__(self, var_budg, utility, delta, parent
           = None, with_child=False):
3          self.var_budg = var_budg #State information
4          self.utility = utility #State information
5          self.delta = delta #Additional information
6          self.parent = parent #Additional information
7          self.with_child= with_child  #Additional
               information
8      def __eq__(self, other):
9          return other.var_budg == self.var_budg and
               other.utility == self.utility
10     def __hash__(self):
11         return hash(str(self.var_budg) + str(self.
               utility))
```

The functions in line 8 redefines the equality property used by the set so that states with the same variance budget and utility are considered to be equal regardless of the other elements in the object. In line 9, the defined function recreates a new hash that depends only on the state information. Two elements with the same state information would therefore return the same hash and be treated as equal.

5.0.2 Quantization

Different type of Quantization have been applied to the data processed by the algorithm.

State discretization

The state discretization expresses the degree of difference between each state. A state discretization of 0 means that two states with utility and variance budget that are equal once they are rounded to the unit will be considered as equal. Similarly, a state discretization of 1 indicates that two states equal when rounde two the tenth are actually equal.

Delta discretization

The delta discretization expresses the degree of precision with which the investor can chose an action. The number of possible actions is bounded by the value of the initial stock price. It is important to note that the delta value can be positive or negative as the buyer can buy (negative Δ) or sell (positive Δ) the underlying. The smaller the value of the state discretization will be the more possible moves there will be for the investor.

Market Jump discretization

This is very similar to the the Delta discretization. The total number of possible jumps for the market player will be bounded by the jump constraint zeta. As mentioned before, the smaller the degree of discretization the larger the number of possible moves for the market.
One of the main challenges of this implementation is to find the right compromise between increasing the accuracy of the minimax value by implementing very small discretization while maintaining a reasonable running time and memory usage.

5.1 Black-Scholes and Implied volatility

5.1.1 Back-Scholes closed form equations.

One of the main ambition of this project is to compare the value of the strategy computed with the minimax algorithm and then compare it to the results given by the Black-Scholes option pricing model. Implementing the Black-Scholes model in closed form was therefore necessary.
Since the scope of this project is limited to European calls which admit a closed form under the Black-Scholes. The implementation of the Black-Scholes model has been done easily. The normal cumulative distribution function necessary to implement the closed form equations (cf. equations 2.12 in chapter 2) has been implemented with the *scipy.stats* tool.

5.1.2 Implied Volatility

In order to compute the implied volatility, Newton's Algorithm for non linear equations was implemented. As was seen in chapter 2 ("The Smile Problem"), given $\mathbb{P}(S, T, K, \sigma)$. If $\mathbb{P}$ and S are quoted on the market as $\mathbb{P}_0$ and S_0, the implied volatility σ can be found by solving the following equation ,

$$V_0 = V(S_0, T, K, \sigma) \tag{5.1}$$

Newton's algorithm is given by the following recursion:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5.2}$$

The algorithm applied to this specific case becomes:

$$\sigma_{n+1} = \sigma_n - \frac{V(\sigma_n)}{V'(\sigma_n)} \tag{5.3}$$

The algorithm will iterate until the black-scholes price converges to the price given to the algorithm.

In that case $V'(\sigma_n)$ corresponds to the Vega Greek that measures the option price sensitivity to the underlying's volatility. This is a derivative that can be expressed in the following manner,

$$\text{Vega} = \frac{\partial V}{\partial \sigma} \quad \text{if we derive equation 2.13} \quad , \text{Vega} = S\sqrt{T}N'(d_1) \tag{5.4}$$

$$\text{with} \quad N'(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \tag{5.5}$$

The pseudo-code for the Newton recursion is seen below,

Algorithm 3: Compute Implied volatility with Newton Algorithm

```
Function ImpliedVolatility(Target,S,K,T)
    sigma = 0.4
    for iteration in range(1,maximum iteration) do
        get BSprice for given sigma;
        get Vega for given sigma ;
        if target - BSprice < error then
            return sigma ;
        else
            sigma = sigma - BSprice/vega ;
```

Chapter 6

Testing the minimax Implementation

In order to make sure that the value given by the algorithm is accurate, small test functions have been implemented and used during the development of the minimax algorithm. These tests where implemented for two specific small trees which were first solved manually. The first tree consisted in a two trading period game where $\Delta = [1, 2]$ and $r = [-1, 1]$. This tree is showed in figure 6.1. In the second case, the tree included three trading periods for which $\Delta = [1, 2]$ and $r = [1]$.

In addition to testing that the final result matched, it was also necessary to test that the intermediary steps of the algorithm were also coherent by implementing small test functions. In the Minimax algorithm,

1. The minimax algorithm follows a backward path during which the value of the utility at each parent node is overwritten (see Figure 6.1). The test implemented verified that this utility was overwritten by the correct value

2. We also checked whether the lists *same_parent*, *same_delta* contain respectively objects with the same *parent_id* element and *delta* element.

In both cases, the minimax value given by the algorithm matched the one found manually. Testing this algorithm for small discretization or for a greater of trading periods is quite challenging since it requires solving very large tree by hand, which past a certain number of states become extremely

difficult. However if the algorithm works in these two previous cases, there is no reason why on larger scale trees. We can safely assume that the recursion loop implemented in the minimax algorithm outputs accurate results,

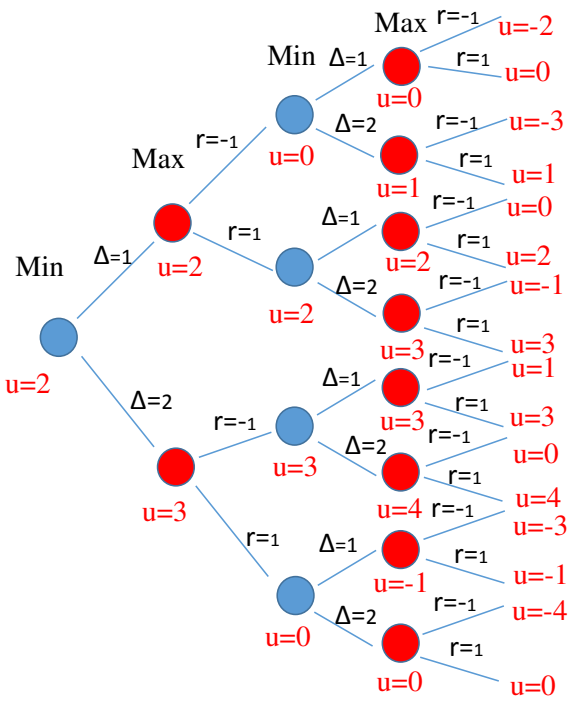

Figure 6.1: This figure shows the utility at each node of the tree after the the backward path has been generated.

Chapter 7

Results

7.0.1 Effect of parameters *var_budg* and ζ.

In the paper written by Abernethy et al. (2013), the variance budget and the jump constraint ζ were assumed to be reasonable approximations of the Black-Scholes constant volatility. Before evaluating whether this assumption is reasonable or not, it is necessary to evaluate what are the exact effect of these parameters on the final value of the algorithm.

Figure 7.2 and 7.3 show how the minimax value and the total number of states generated evolve as the variance budget increases. As expected, both of these elements increase as *var_budg* increases. Indeed, if the variance budget is small, very quickly, the total possible moves of the market will be reduced to the small jumps that will keep variance budget strictly higher than 0. The market contribution to the accumulated negative payoff (see equation 3.3) will become very small very quickly. A smaller variance budget will therefore imply a smaller minimax value. This can also explain why the number of states increases as the variance budget increases. More states will be generated since more of them will satisfy the positive variance budget constraint.
The result of this simulation are expected and show that the variance budget constraint has been correctly implemented.

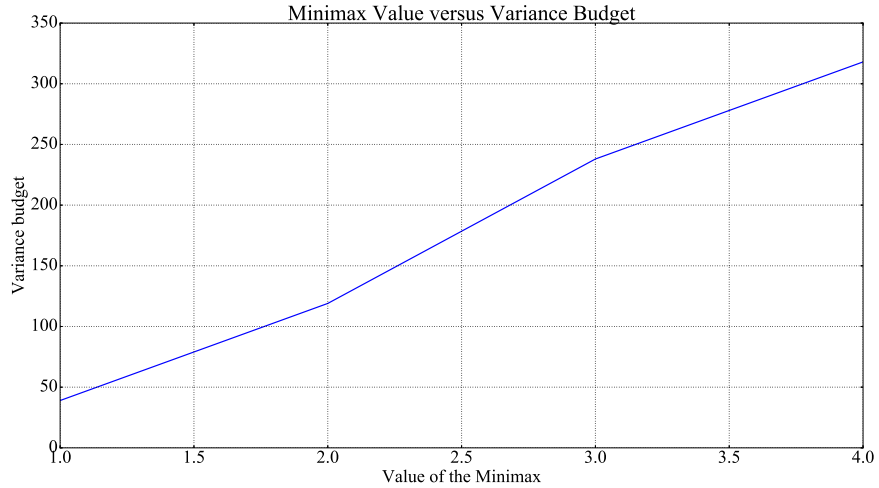

Figure 7.1: Effect of the variance budget on the minimax value

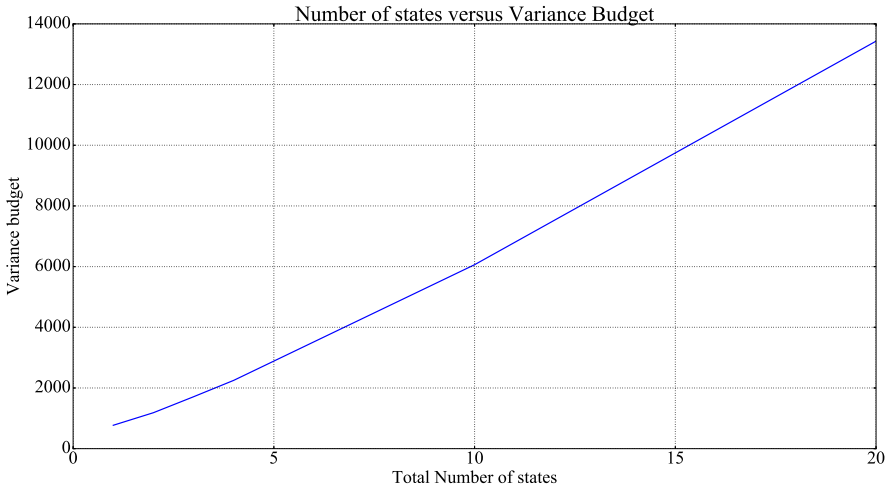

Figure 7.2: Effect of the variance budget on the total number of states

Figure 5.4 shows how the jump constraint zeta affects the final minimax value for different variance budgets. As can be seen, the value of the game increases linearly as the value of zeta increases before reaching a peak and falling again. This is to be expected. Large market jumps will have large contribution to the to total accumulated payoff but also on the variance budget. Therefore, very quickly, the variance budget will get close to zero and the set of possible market jumps will become small. This is why the

accumulated negative payoff will stagnate at a certain value for large values of zeta. As exepected, if the variance budget is increased the minimax value will reach its peak for larger value of zeta.

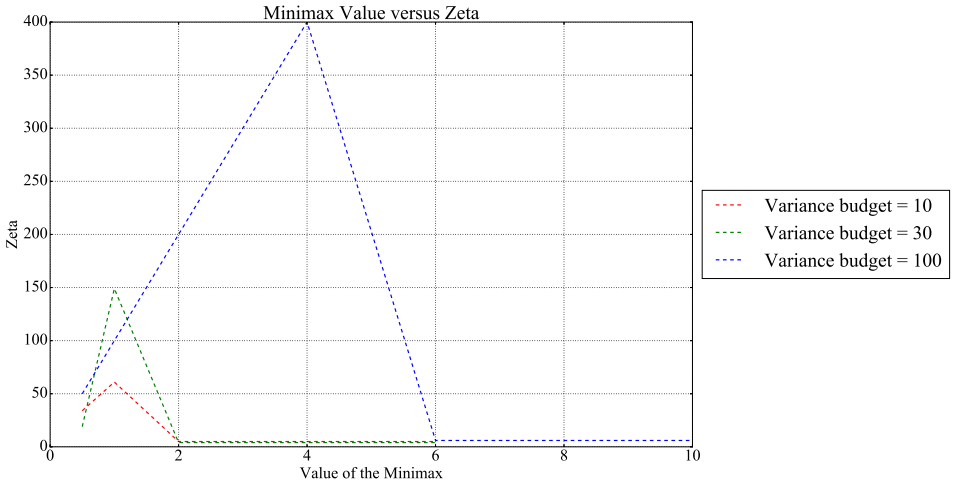

Figure 7.3: Effect of zeta on the minimax value for variance budget of 10,30,100.

7.0.2 Memory usage and Timing

Figure 7.4 shows how the number of states increases with depth. In that specific case the number of trading periods is 10 and the market has 6 possible moves and the investor has 10 possibilities at each trading period. The increase in number of states is very quick for small depth and decreases for larger depth. This is due to the fact that for larger depth, the variance budget decreases and the set of possible market moves becomes smaller.This is why the number of created states reduces. The same progression is observed for the time required to generate each depth. As the depth increases, the number of operations performed to calculate the payoff will increase at a slower rate.

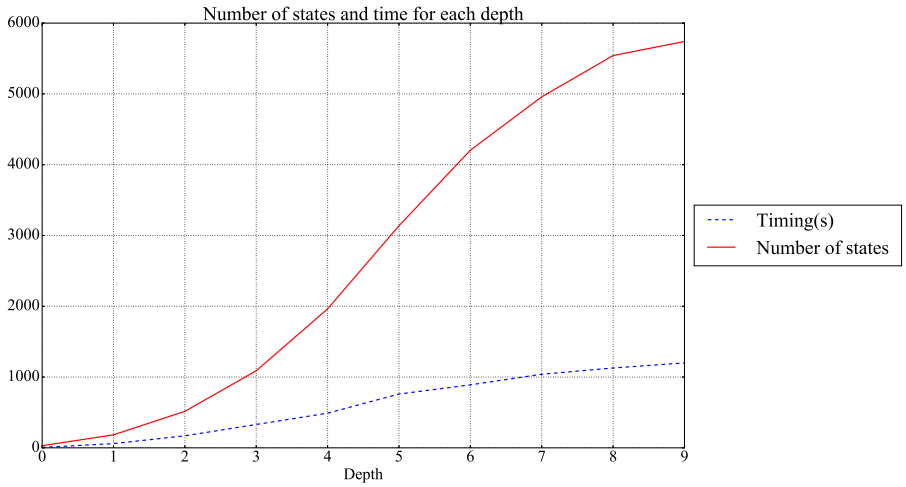

Figure 7.4: Evolution of the number of states and timing with the depth.

The memory usage has been monitored thanks to the python library *guppy* and the function *hpy* which returns the total memory required by the script. As can be seen, the memory usage and the timing increases linearly with the number of states generated.Figure 7.5 shows that the number of states that can be computed before reaching $2^{30}kB$ (see chapter 4) is very big. Figure 5.6 compares the time taken by the function taken *build_state_map* with the timing of the function $Minimax$. The most costly operation in terms of time is the creation of the state map. This is to be expected since while generating the depth, the state map algorthm considers all possible combinations of Δ_i and r_i, although it only adds a fraction of them to the depth. The minimax algorithm therefore operates on a smaller data set.

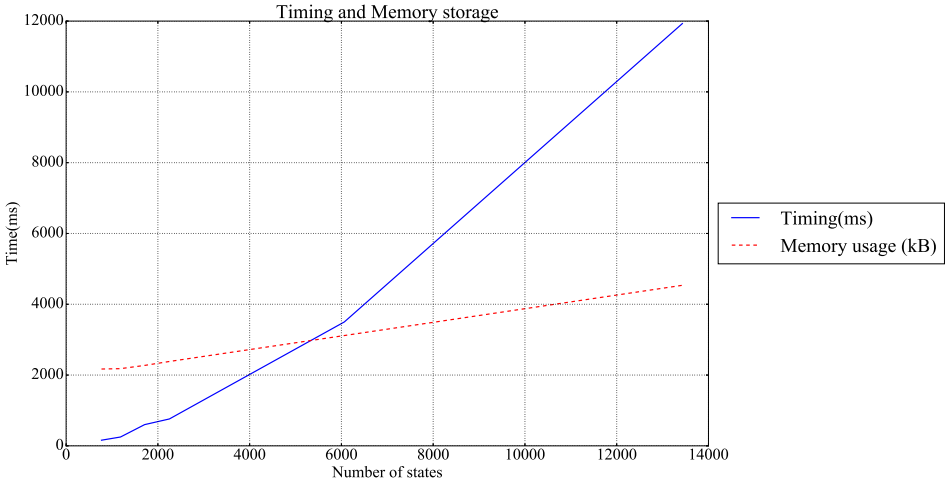

Figure 7.5: Timing and Memory in function of the number of states

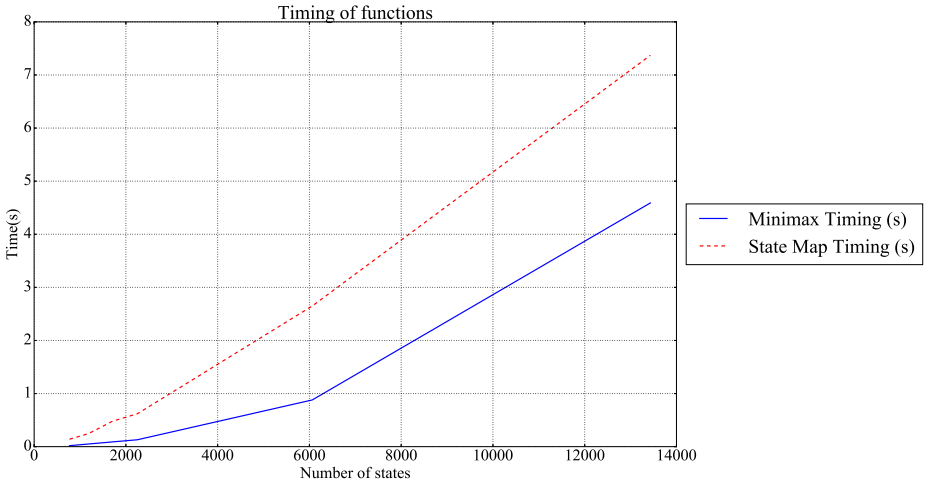

Figure 7.6: Timing for building the state map and solving the minimax

7.1 Relating the Minimax to the Black-Scholes.

7.1.1 Replicating the Black-Scholes

The paper "How to Hedge an Option Against an Adversary" shows that if the number of trading periods tend to infinity, the value of the game will tend to the Black-Scholes result. There will therefore be a minimum number of trading periods for which the value of the minimax will converge to the value given by the Black-Scholes closed form equation.

As seen in chapter 2, the Black-Scholes model takes as input parameters the underlying, the strike, the maturity and the volatility. When trying to replicate the Black-Scholes results by the minimax, the value of the variance budget and of zeta giving matching Black-Scholes price for a given volatility can be set by trial and error. However, the choice of the correct variance budget can be facilitated by the following fact: the minimax value grows almost linearly with the variance budget (see figure 7.12). Therefore, knowing this relationship enables to tune easily the minimax model to obtain the wanted result.

The minimax value was computed for several trading periods. It seems that convergence with the Black-Scholes solution, for an option of 10 days, is observed when the number of trading periods is higher than 100. This can be shown in figure 5.7 where the minimax value is computed for an option with a strike value of 4 and an initial stock value of 5, a variance budget of 2 and a zeta value of 2.

Figure 5.9 and Figure 5.10 shows the result for a higher Black-Scholes volatility. In order to find a matching price, the variance budget was increased to 200. In that case we can see that convergence occurs for more than 250 trading periods. This is to be expected since the initial game for a small number of trading periods is higher than for a smaller variance budget. This explains why,the minimax value requires more trading periods to converge.

For both cases, the state discretization has been changed. Figure 5.7 and 5.8 shows the effect of state discretization on the convergence of the minimax value to the Black-Scholes price. In figure 5.7, the state discretization applied is set 1 while it is equal to 2 in figure 5.8. The higher the state dis-

cretization the smaller is the error between the two results given by the two algorithms. The same phenomenon is observed for a higher Black-Scholes volatility.

Recovering the Black-Scholes result requires a great number of trades. In the previous simulations, the traders is assumed to have performed more than 10 trades each day. Indeed, the considered option only covered a maturity of 10 days. Therefore, the number of trades and of states required to achieve results close to the black-scholes will therefore become very important for longer maturity options. This will greatly impact timing and memory usage of the system.

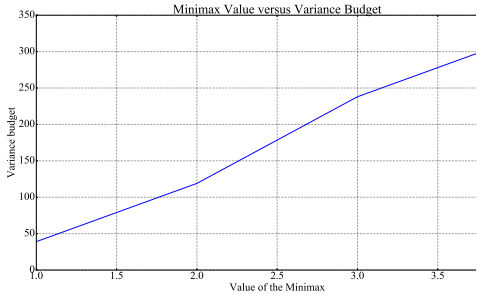

Figure 7.7: Effect of variance budget on minimax value

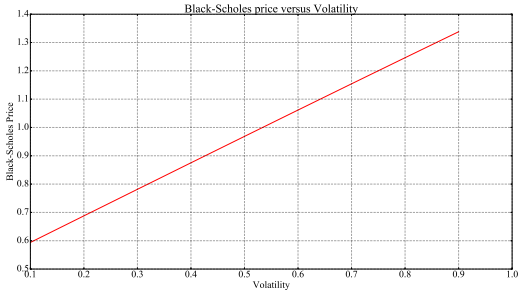

Figure 7.8: Black Scholes price versus volatility

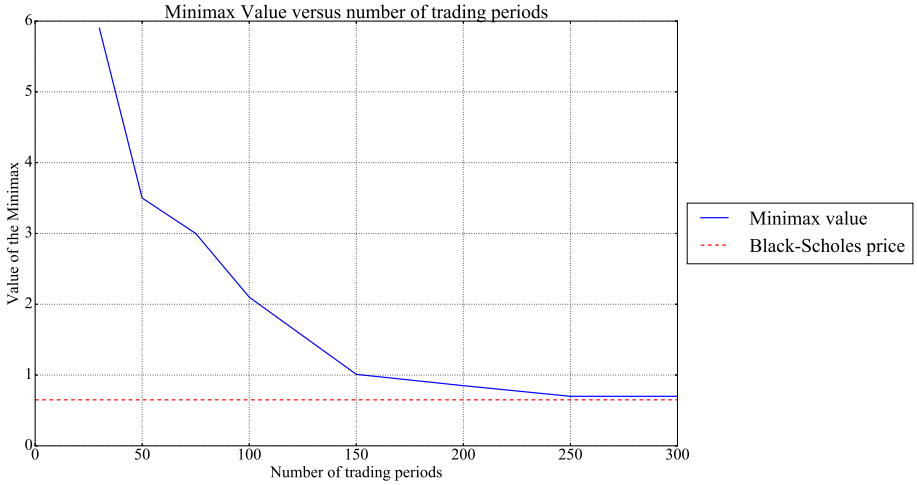

Figure 7.9: The blue line shows the minimax value versus the number of trading periods for state discretization of 1 and variance budget of 100, strike of 4 and initial stock price of 5. The red line indicates the corresponding Black Scholes price with standard deviation of 0.5.

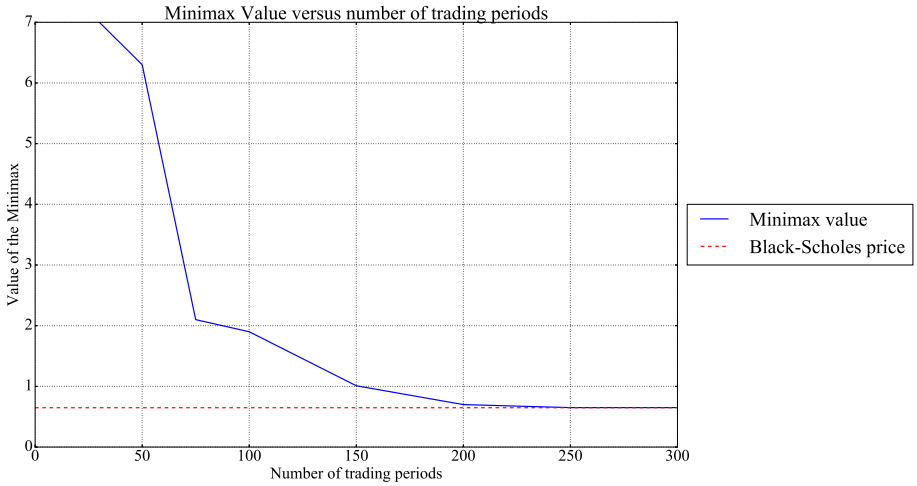

Figure 7.10: The blue line shows the minimax value versus the number of trading periods for state discretization of 2 and variance budget of 100, strike of 4 and initial stock price of 5. The red line indicates the corresponding Black Scholes price with standard deviation of 0.5.

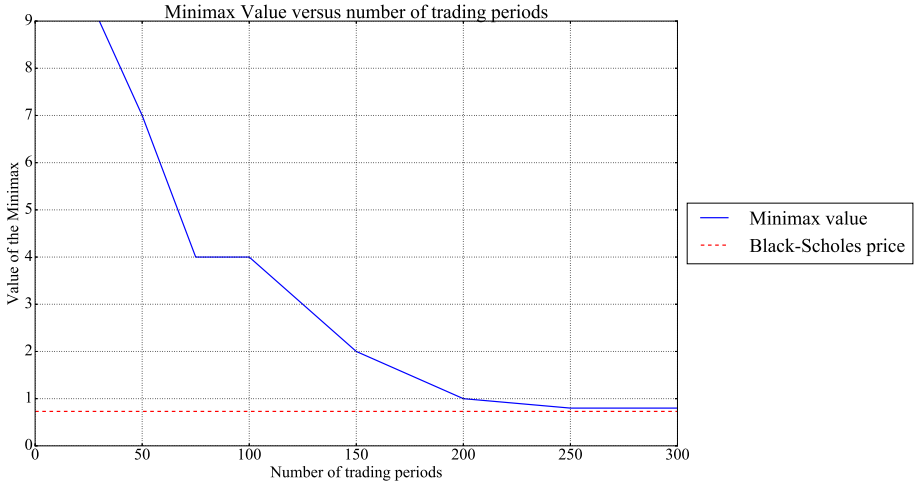

Figure 7.11: The blue line shows the minimax value versus the number of trading periods for state discretization of 1 and variance budget of 200, strike of 4 and initial stock price of 5. The red line indicates the corresponding Black Scholes price with standard deviation of 0.8.

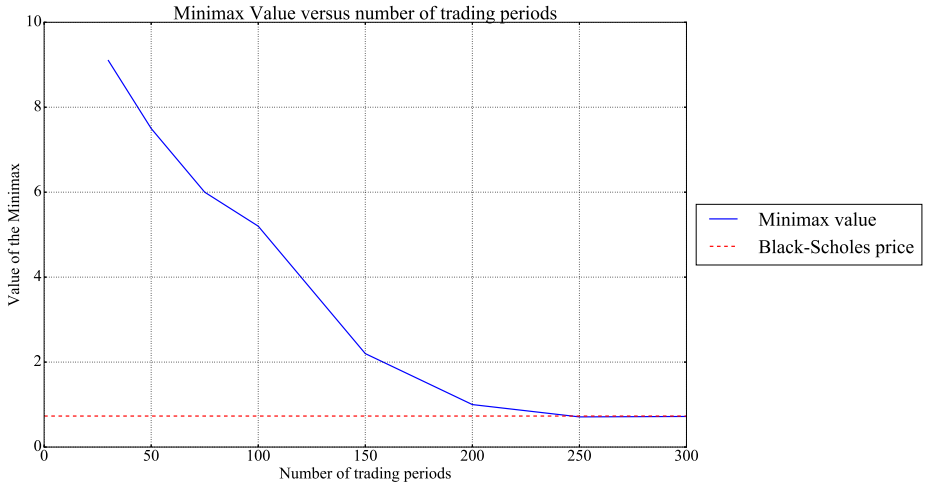

Figure 7.12: The blue line shows the minimax value versus the number of trading periods for state discretization of 2 and variance budget of 200, strike of 4 and initial stock price of 5. The red line indicates the corresponding Black Scholes price with standard deviation of 0.8.

7.1.2 Replicating the Smile

The implied volatility of the Black-Scholes described in chapter 2 can be obtained by solving a non-linear equation using a Newton recursion (see chapter 5). In these simulations, the value of the minimax have been computed for varying strikes and matched to the corresponding Black-Scholes implied volatility.

As seen before, the minimax value will converge to the value of the Black-Scholes for at least 100 trading periods. Figure 6.10 shows the result of simulations realized on the same option as in section 5.2.1. In that case however, the number of trading periods is equal to 50. By varying strike values, it seems that the implied volatility increases as the option becomes more and more out of the money or in the money. The shape observed seems to be reflecting the dynamics of the smile.

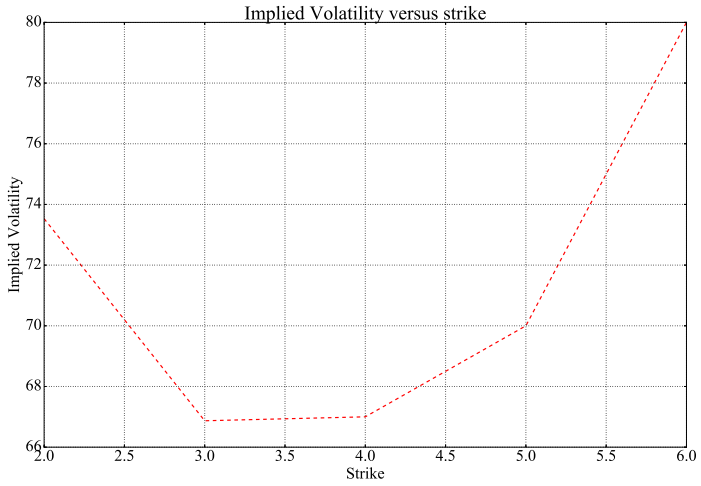

Figure 7.13: Implied volatility in function of the strike.

Chapter 8

Conclusion

8.1 Evaluation

The results obtained are quite interesting as they confirm the fact that the Black-Scholes assumption of constant volatility holds only if the number of transactions is very large. The trader must trade continuously to achieve a risk less option. However, as the number of trading periods decreases, not only will the minimax value diverge from the Black-Scholes model, it will also contradict one of its main assumption by reflecting smile dynamics. This phenomenon has also been described in Haug and Taleb (2010).

Overall, the formulation of minimax problem which has been implemented correctly mimics the Black Scholes model. This is partly due to the choice the market constraints. Indeed, as observed in in figures 7.7 and 7.8, the effect of the variance budget on the minimax value is very similar to the effect of the volatility on the Black Scholes price.

The main requirements of this project was to implement the minimax algorithm described in Abernethy et al. (2013) and find under which condition the algorithm converges to the Black-Scholes value. Finding the conditions under which smile dynamics would be reflected by the algorithm results was another important objective. These two elements have been achieved although timing was an important limitation of the implementation.

8.2 Further works

Starting point of a new proof.

The background research performed in chapter 2 was aimed at giving an overall understanding on the world of option pricing and set the rest of the work to be done in a precise context. However, the mathematical knowledge included in this section could potentially be the starting point of a new proof, similar to the one in Abernethy et al. (2013) which aims at solving the same minimax problem with different constraints. Instead of having a final solution converging to the Black-Scholes equation, it could converge to the local volatility model or the Heston volatility model. For example, modifying the TotVarConstraint by a constraint that varies smoothly and admits a lower bound might be a good attempt to reproduce the Heston volatility model and obtain solutions that no longer follow the Black-Scholes but are in line with the stochastic model. Such a constraint can have the following aspect,

$$c_2 < r_i - r_{i+1}^2 < c_1. \tag{8.1}$$

In other words, chapter 2 could be used to unveil mathematical hints that could eventually be the starting point of a proof showing the convergence of the minimax solution to another more realistic model.

Implementation of new algorithm

As said previously, timing represents an important limitation in the minimax implementation. Running the algorithm for a high number of trading periods and for small discretization would be very time costly. An implementation of the CFR/CFR+ algorithm would be necessary to test convergence properties on longer maturity options. It would also be very intersting to test other online learning algorithm and see which performs best on the given problem.

8.3 Final Remarks

The code for this project can be found on my github repository at the following address: `https://github.com/rouhanaa/Hedging-with-minimax`.

Bibliography

Jacob Abernethy, Peter L Bartlett, Rafael, M. Frongillo, and Andre Wibisono. How to hedge an option against an adversary: Black-scholes pricing is minimax optimal. 2013.

Rafal Weron Uwe Wystup Agnieszka Janek, Tino Kluge. Fx smile in the heston model. 2010.

Henri Berestycki, Jerome Busca, and Igor Florent. Computing the implied volatility in stochastic volatility models. 2004.

Bruno Dupire. Pricing with a smile. 1994.

Patrick S Hagan, Deep Kumar, Andrew S. Lesniewski, and Diana E. Woodward. Managing smile risk. 2007.

Tim Harford. Black-scholes: The maths formula linked to the financial crash. a.

Tim Harford. Timecomplexity. b.

Espen Gaarder Haug and Nassim Nicholas Taleb. Option traders use (very) sophisticated heuristics, never the black–scholes–merton formula. 2010.

Steven L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. 1993.

John C Hull. *Option, Futures and Derivatives*. Pearson, 2007.

Rickard Kjellin and Gustav Lovgren. Option pricing under stochastic volatility. 2006.

Michael Johanson Oskari Tammelin Michael Bowling, * Neil Burch. Supplementary materials for heads-up limit hold'em poker is solved.

Johannes Ruf. Local and stochastic volatility. 2012.

Roel van der Kamp. Local volatility modelling. 2009.

Hui Ye. A comparison of local volatility and implied volatility. 2011.