Department of Electrical and Computer Engineering, University of Thessaly Volos.

Θεωρητική και πειραματική μελέτη παράλληλων χρηματοοικονομικών συστημάτων

Theoretical and experimental study of parallel financial systems

Diploma thesis by Nikou Petros

Supervisors:
Associate Professor Panagiota Tsompanopoulou

Professor Panagiotis Mpozanis

Volos, September 2016

Department of Electrical and Computer Engineering, University of Thessaly Volos.

Θεωρητική και πειραματική μελέτη παράλληλων χρηματοοικονομικών συστημάτων

Theoretical and experimental study of parallel financial systems

Diploma thesis by Nikou Petros

Supervisors:
Associate Professor Panagiota Tsompanopoulou

Professor Panagiotis Mpozanis

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την

…......................................	....
Παναγιώτα Τσομπανοπούλου	Παναγιώτης Μποζάνης
Επίκουρη Καθηγήτρια Π.Θ.	Καθηγητής Π.Θ

....

ΠΕΤΡΟΣ ΝΙΚΟΥ

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστημίου Θεσσαλίας

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου σε όλους όσους με βοήθησαν στην περάτωση της παρούσας διπλωματικής εργασίας και κυρίως την επιβλέπουσα καθηγήτρια μου κα. Παναγιώτα Τσομπανοπούλου για την συνεχή και σημαντική καθοδήγησή της καθ ‘όλη την διάρκεια της πτυχιακής.

Θα ήθελα να ευχαριστήσω επίσης την οικογένεια και τους φίλους μου για την άνευ όρων υποστήριξη τους σε όλη την διάρκεια της φοίτησής μου.

Περίληψη

Σκοπός αυτής της πτυχιακής είναι να επεξηγήσει το οικονομικό μοντέλο τιμολόγησης των stock options των Black-Scholes-Merton γνωστό ως Black-Scholes. Για την πλήρη κατανόηση της formula παρουσιάζεται μια διαδικτυακή χρηματιστηριακή εφαρμογή σε Java η οποία έχει την δυνατότητα να πραγματοποιεί ταυτόχρονα downloading και parsing τιμές μετοχών απο το Google Finance. Στη συνέχεια με βάση αυτές τις τιμές μπορούμε να τιμολογήσουμε μέσω της εφαρμογής μια option για οποιαδήποτε μετοχή. Στο 6^{ο} κεφάλαιο καθώς και παράρτημα παρουσιάζεται μια πιο λεπτομερής ανάλυση για την θεωρία της εφαρμογής με σκοπό ο αναγνώστης να συλλάβει πλήρως την ιδέα της εφαρμογής και πως αυτή γίνεται implemented. Η formula καθώς και η εξίσωση των Black-Scholes αποτελεί το σήμα κατατεθέν στο τομέα των οικονομικών μαθηματικών, καθώς κάθε μελέτη σε αυτόν το τομέα θα ήταν ανολοκλήρωτη χωρίς την κατανόηση της λογικής πίσω από αυτή την εξίσωση. Το μοναδικό πρόβλημα έγκειται στον υπολογισμό μιας παραμέτρου της εξίσωσης, αυτής της μεταβλητότητας (Volatility) η οποία δεν είναι άμεσα παρατηρήσιμη στην αγορά. Το πρόβλημα αυτό αντιμετωπίζεται με 2 διαφορετικές μεθόδους: 1ον την ιστορική μεταβλητότητα και 2ον την τεκμαρτή μεταβλητότητα που θα δούμε παρακάτω. Επιπλέον θα παρατεθούν τρόποι βελτίωσης του αλγορίθμου με μείωση του μέσου χρόνου parsing και downloading.

Acknowledgements

I would like to express my sincere gratitude to all the people who helped me with the completion of this thesis and foremost my thesis advisor Professor Panagiota Tsompanopoulou, for her constant guidance and for providing me precious advice and suggestions.
Last but not least I would to thank my family and friends for their unconditional support and encouragement during all these years of my studies.

Petros Nikou
Volos 2016

Abstract

The purpose of this thesis is primarily to study the Black-Scholes-Merton formula or just Black-Scholes in a detailed and comprehensible way and then to simulate a stock application by presenting an online Java application able of downloading and parsing concurrently in real time, stock prices from Google Finance using threads. In order to fully understand the Black-Scholes formula there is a Black-Scholes calculator implemented also in Java for pricing options. In both chapter 6 and the appendix there is a detailed report about the application, which will guide the reader step by step in order to fully conceive the whole concept of the application and how is implemented. Black-Scholes formula is a hallmark of mathematical finance and any study of this field would be incomplete without having understood the logic behind this equation. The main problem is estimating the only parameter, which is not directly observable in the market, the volatility. However this problem is tackled through two different methods: historical volatility and implied volatility. Moreover due to the fact that the concurrent parsing algorithm has a lot of room of improvement, there will be introduced some ways in order to decrease the average parsing time.

Contents

Thesis Structure

Section I offers a quick introduction to the Black-Scholes formula by describing it's parameters and the reason for it's existence.

Section II introduces the definition of the derivative and then what exactly is a *stock option*, which basically is another type of derivative. Moreover there will be a detailed description of each of the five parameters of the formula in order for the reader to fully understand them.

In Section III there will be introduced some of the basic *strategies* involving options in a detailed way especially a complex strategy known as iron condor.

Section IV will offer a detailed description of the *Greeks, which* are essential and crucial in strategies involving options. Section V will present the application and how it is implemented. Moreover in order for the reader to fully understand concurrent programming we offer a comprehensive description of the *Executors* class.

In section VI conclusions and future work of this thesis are being quoted.

Last but not least on the appendix some screenshots and a further description of the application will be quoted.

Chapter 1

Introduction

The Black-Scholes-Merton formula was first presented in [1]. Only Scholes and Merton received the Nobel Prize in 1997 (Black had passed away two years ago, in 1995). The problem that was presented in [1] was finding the "fair" value of a stock option, what a stock option really is and what exactly "fair" means in this context. The equation basically gave birth to the field of financial engineering, which is concerned with the design of financial contracts and the pricing of derivatives. The value of a call option for a non dividend-paying stock is shown below:

$$C(S, K, T, r, \sigma) = N(d_1)S - N(d_2)Ke^{-r(T-t)} \quad (1)$$

$$d1 = \frac{ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \quad (2)$$

$$d2 = d1 - \sigma\sqrt{T} \quad (3)$$

where:

N: cumulative standard normal distribution

S: current stock Price

K: strike price

T-t : the time until expiration in years with t = 0

r: current risk-free rate of return

σ: is the (annualized) volatility of the stock

for a put option it would be:

$$P(S, K, T, r, \sigma) = N(-d_2)Ke^{-r(T-t)} - N(-d_1)S = Ke^{-r(T-t)} - S + C(S, K, T, r, \sigma) \quad (4)$$

The above pricing formula for a put option reveals the so called the Put-Call parity which will allow us to determine the value of the call, given the value of the put, requiring that they both are options on the same stock, with the same strike price and expiration date. Put-Call parity will be discussed in the section V in a more detailed way.

Chapter 2

Definitions and basic terminology

In this section, we describe the concepts and terminology used in finance considering this paper. A ***derivative*** is a financial instrument whose value depends on the value of some underlying variable. Specifically a financial instrument is a monetary contract between two parties, which can be traded, created, modified and settled. It can be cash (currency), evidence of an ownership interest in a entity (share), or a contractual right to receive or deliver cash (bond). In most cases this is the price of a stock on a certain date, but it could also depend on some interest rate, or even on something more unusual like the amount of rainfall in a certain week. The important feature is that its value is well defined [7], given the value of the underlying variable. Among all these derivatives the stock option is the one presented here in detail. An *option* gives the holder the right, but not the obligation to sell or buy a stock at a certain price, the strike price, on a pre-specified date, the expiration date. If the right to buy the stock is bestowed upon the holder, then this is called a *call option*, or simply *call*, while if the right to sell is conferred, it is called a ***put option***, or ***put***. Another distinction is made between ***European*** and ***American*** options, where the first one only allows the option to be exercised on the expiration date, while the latter allows the holder to exercise the option any time up to the expiration date. Exercising the put here refers to the act of buying the stock and using the option to sell the stock at a higher price. On the other hand, exercising the call consists of using the option to buy the stock at a lower price than the current market price, and then sell it at the market price, which will be higher.

2.1 Value of an Option

The value of an option is the amount of money, which will receive the holder of a put or a call option on the expiration date, if he decides to exercise the option. If on the expiration date the price of the stock is S, and the strike price is K, then the value of the option is $\max(0, S - K)$, or $\max(0, K - S)$, for the case of a call and put, respectively. To see this for the case of the put, suppose that $K > S$ on the expiration date. Then, if the holder of a put option, can buy the stock for S, and then exercise the option to sell it for K, in which case he/she will have realized a profit of $K-S$. Similarly for a call option he/she will have realized a profit of S-K, of course if $S<K$ he/she would not exercise the call option because it would be unprofitable. Although K-S and S-K are the profits for the case of a call and put respectively, the above are the maximum *theoretical* profits because in K-S and S-K we have to also subtract the

price of the *premium*, which is calculated by the Black-Scholes formula. So the real profits would be K-S-P(S, K, T, r, σ) and S-K-C(S, K, T, r, σ) of a call and put option respectively.

2.2 Time Value and Intrinsic Value

Option premiums feature two basic components, the *intrinsic value* and the *time value.* The *intrinsic value* is the difference between the underlying and the strike price of a stock. Specifically, the intrinsic value for a call option is equal to the underlying price minus the strike price; for a put option the intrinsic value is the strike price minus the underlying price.

Intrinsic Value(Call) = Underlying price – Strike Price (5)

Intrinsic Value(Put) = Strike Price – Underlying Price (6)

By definition the only options that have intrinsic are those that are *in-the-money.* For call options, *in-the-money* refers to options where the strike price is less than the current underlying stock price. A put option is in-the-money if the strike price is greater than the underlying price of the stock. To recap an option is also said to be *at-the-money* if the intrinsic value is zero, *in-the-money* if the intrinsic value is greater than zero, and *out-of-the-money* if the intrinsic value is less than zero. Time value explains why an out of the money option still is traded before expiration date, because the further out-of-the-money an option is, the lower its market price(option premium).

In-the-money(Call) = Strike Price < Underlying Price (7)

In-the-money(Put) = Strike Price > Underlying Price (8)

Any premium that is in excess of the option's intrinsic value is referred to as *time value.* For example, let's assume a call option has a total premium of $9.00. If the option has an intrinsic value of $7.00, its time value would be equal to $2.00($9.00 – $7.00 = $2.00).

Time Value = Option premium – Intrinsic Value (9)

In general, the more time to expiration, the greater the time value of the option. It represents the amount of time that the option position has to become more profitable due to favorable move in the underlying price. Usually, investors are willing to pay a higher premium for more time, since time increases the likelihood that the position can become more profitable. Time value decreases over time and *decays* to zero at expiration. This phenomenon is known as *time decay.* Because the market price of at-the-money and out-of-the-money options is made up from time value only, we can conclude that time value of options declines the further out of-the-money they are (with other parameters being equal). This is valid for both Call and Put options.

2.3 Risk-Free Rate

The ***risk-free rate*** of return is the theoretical rate of return of an investment with zero risk. ***The risk-free rate*** represents the interest an investor would expect from an absolutely risk-free investment over a specified period of time. In theory, the ***risk-free rate*** is the minimum return an investor expects [10][5] for any investment because he will not accept additional risk unless the potential rate of return is greater than the risk-free rate. In practice, however, the risk-free rate does not exist because even the safest investments carry a very small amount of risk. Thus, the interest rate on a three-month U.S. Treasury Bill is often used as the risk-free rate for U.S.-based investors.

2.4 Historical Volatility

Volatility of a stock is defined to be the standard deviation of log-returns of the stock. The log-returns are the logarithms of the ratio of successive prices. The Black-Scholes model assumes a constant volatility, and one way to estimate this is to use historical volatility as an estimator. If we have price data from n + 1 periods (in our case days), then the estimate for historical volatility [7] is given by:

$$\hat{\sigma} = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(u_i - \bar{u})^2}}{\sqrt{\tau}} \quad (10)$$

where:

$$u_i = ln\left(\frac{S_i}{S_{i-1}}\right) \quad (11)$$

$\bar{u}$: The sample average of all u_i

S_i: The stock price in period i

τ: The total length of each period in years.

The problem with determining τ, is that there are only 252 days in the year on which trades actually take place (since there are weekends and holidays), but 365 days in a typical year. We could let τ equal $\frac{1}{252}$ or $\frac{1}{365}$. The second assumption is more elegant, especially considering the fact that another term in the Black-Scholes formula, the term which represents time discounting at the risk-free rate of interest, r, is at work continuously, even the weekends.

2.5 Implied Volatility

Opposed to the historical volatility which is *backwards-looking* implied volatility is *forward-looking.* It is a parameter part of an option pricing model, such as the Black-Scholes model, that gives the market price of an option. Implied volatility is a measure of the estimation of future variability [15] for the asset underlying the option contract. Since implied volatility is *forward-looking*, it helps to gauge the sentiment about the volatility of a stock or the market. However, it does not forecast the *direction* in which an option will be headed. Implied volatility is not directly observable, so it needs to be solved [4] using five other inputs of the model: the market price of the option, the underlying stock price S, the strike price K, the time to expiration τ and the risk-free rate. The implied volatility is calculated by taking the market price of the option also known as *market's belief*, entering it into the B-S formula and back solving [6] for the value of volatility.

The Black-Scholes model makes a number of assumptions that may not always be correct. The model assumes volatility is constant, when in reality it is often moving. The model further assumes efficient markets are based on a *random walk* of asset prices. The Black-Scholes model is limited to European style options that can only be exercised on the last day as opposed to American style options that can be exercised at any time before expiration. The shortcomings of the Black-Scholes method have led some to place more importance on historical volatility as opposed to implied volatility. Historical volatility is the realized volatility of the underlying asset over a previous time period. It is determined by measuring the standard deviation of the underlying asset from the mean during that time period. The standard deviation is a statistical measure of the variability of price changes from the mean price change. This differs from the implied volatility determined by the Black-Scholes method, as it is based on the actual volatility of the underlying asset. However, using historical volatility also has some drawbacks. Volatility shifts as markets go through different regimes. Thus, historical volatility may not be an accurate measure of future volatility. One way to use implied volatility is to compare it with historical volatility. For example let's assume that the implied volatility of a stock was 54.1% on June 29, 2016 at 12:13 p.m. Looking back over the past 30 days, the historical volatility is calculated let's say to be 19.35%. Comparing this to the current implied volatility, it should alert a trader that there might be an event that can affect the stock price significantly. This could be a news event significant enough to elevate the historical volatility relative to the historical volatility for the past 30 days. The equation below represents a way to calculate implied volatility.

$$\sigma = \sqrt{\frac{ln\left(K/_{S}\right) - r(T - t)}{(T - t)(E_{\Gamma} + \frac{3}{2})}} \quad (12)$$

The parameters S, K, r ,T have been already mentioned above. The parameter E_{Γ} is given by the equation:

$$E_{\Gamma} = E_{\xi} - 2 \quad (13)$$

where:

$$E_{\xi} = -\frac{1}{\tau^2}(lnS - lnK) + \beta \quad (14)$$

$$\tau \equiv \sigma\sqrt{T - t} \quad (15)$$

$$\beta \equiv \frac{1}{2} - \frac{r}{\sigma^2} \quad (16)$$

2.6 Options vs Stocks

Investors and traders undertake option trading either to hedge open positions (for example, buying puts to hedge a long position or buying calls to hedge a short position), or to speculate on likely price movements of an underlying asset. The biggest benefit of using options is that of ***leverage.*** For example, let's say an investor has $1200 to invest. The investor is very bullish in the short term on, for example, Apple which we assume is trading at $100 and can buy a maximum of 12 shares of Apple (excluding commissions for simplicity). Apple has also three-month calls with a strike price $105 available for $3. Instead the investor buys four call option contracts (again ignoring commissions) where, each contract is equal to 100 shares. Shortly before the call options expire, suppose Apple is trading at $113 and the calls are trading at $8. The investor decides to exercise the calls. Here's how the return on investment stacks up in each case:

- Outright purchase of Apple shares: Profit = (113 – 100) x 12 shares = $156. The return will be $\frac{(113-100)}{100} * 100 = 13\%$
- Purchase of 4 call option contracts. Profit = 8 x 100 x 4 – (3 x 100 x 4) = $2000. The return will be $\frac{(3200-1200)}{1200} * 100 = 166,7\%$

The risk with buying the calls rather than the shares is that if Apple had not traded above $105, the calls would have expired worthless and we would experience a loss of $1200 (option premiums). In fact Apple would have had to trade at $108 ($105 strike price + $3 premium paid), or about 9% higher from its price when the calls were purchased, in order for the trade to break even.

Chapter 3

Strategies involving options

3.1 Married Put

An option can be seen as an insurance contract, where one party wants to insure, or *hedge*, a certain position in the market up to some expiration date in the future. One possible strategy is to insure against downward losses on a stock by buying a put on the stock at a certain strike price, so that if the price of the stock went below the strike price one can always recover the losses by exercising the put. Obviously this strategy functions like an insurance policy, and is known as *married put.* The monthly return can be calculated as in [8].

3.2 Equity Collar

In order to understand the Equity collar properly we have to introduce what a *covered* call is. Writing an option refers to the act of selling an option. When someone writes ("sells") an option he/she must deliver to the buyer a specified number of shares if the option is exercised. The writer has an obligation to perform a *duty* while the buyer has the option to *take action*. There are two general types of option writing: *covered* and *naked*.

3.2.1 Covered Call

In a *covered* call, the option writer ("seller") already owns the underlying trading instrument and wishes to make extra money from the position. He/she can write ("sell") an option based on the expectation that the underlying's price will move in a particular way. The buyer pays the writer a premium in exchange for writing an option. If the option trades at a value that benefits the buyer the seller is obligated to hand over the shares. If the option expires at a value that does not benefit the buyer, the seller retains the original shares. If the option writer does not own the underlying instrument, it is said to be *naked* option. This is more risky than writing a covered call since the writer is still obligated to produce the specified number of shares of the particular contract (without owning them already). The example below offers a great understanding of how an Equity collar works.

Let's consider an investor who owns one hundred shares of a stock with underlying price of \$5. An investor could construct a collar by buying one put with a strike price of \$3 and selling one call with a strike price of \$7. *The collar* would ensure that the gain on the portfolio will be no higher than \$2 and the loss will be no worse that \$2. There are three possible scenarios when the option expire:

- Scenario 1: If the stock price is above the \$7, then the person who bought the call from the investor will exercise the purchased call; the investor effectively sells the shares at the \$7 strike price. This would lock in a \$2/share profit for the investor. He *only* makes a \$2 profit(minus fees), no matter high the share price goes. For example, if the stock price goes up to \$11, the buyer of the call will exercise the option and the investor will sell the shares that he bought at \$5 for \$11, for a \$6 profit, but must then pay out \$11 – \$7 = \$4, making his profit only \$2/share (\$6 - \$4). The premium paid for the put must then be subtracted from this \$2 profit to calculate the total return [8] on this investment.
- Scenario 2: If the stock price drops below the \$3 strike price on the put then the investor may exercise the put and the person who sold it is forced to buy the investor's 100 shares at \$3. The investor loses \$2/share but can lose only \$2 (plus fees) now matter how low the price of the stock goes. For example, if the stock price falls to \$1 then the investor exercises the put and has \$2 profit/share. The value of the investor's stock has fallen by \$5 – \$1 = \$4. The call expires worthless (since the buyer does not exercise it) and the total net loss is \$2 – \$4 = -\$2/share. The premium received for the call must then be added to reduce this \$2 loss to calculate the total return on this investment.
- Scenario 3: If the stock price is between the two strike prices for example at \$4, on the expiry date both options expire unexercised and the investor is left with 100 shares whose value is stock price (x100), plus the cash gained from selling the call option, minus the price paid to buy the put option, minus fees.

One source of risk is counterparty risk. If the stock price expires below the \$3 floor then the counterparty may default on the put contract, thus creating the potential for losses up to the full value of the stock (plus fees).

3.3 Long Straddle

A long straddle options strategy as in [16] is when an investor purchases both a call and put option with the same strike price, underlying asset and expiration date simultaneously. An investor will often use this strategy when he or she believes the price of the underlying asset will move significantly, but is *unsure* of which direction the move will take. This strategy allows the investor to maintain unlimited gains, while the loss is limited to the cost of both options contracts.

3.4 Long Strangle

A strangle is an options strategy [13] where the investor holds a position in both a call and put with different strike prices, but with the same maturity and underlying asset.

This options strategy is profitable only if there are large movements in the price of the underlying asset, but we are unsure of which way that price movement will be. A strangle is generally less expensive than a straddle as in the latter the contracts are purchased *out-of-the-money.* For example let's say that a stock is trading at $50 a share. To employ the strangle option strategy a trader enters into two options positions, one call and one put. Say a $55 call trades at $3 and a $45 put trades at $2.85. Let's assume that we buy one call contract of $300 ($3 per option x 100 shares) and one put contract of $285 for a total cost of $585. If the price of the stock stays between $45 and $55 until expiration the loss will be $585. The trader will make money if the price of the stock starts to move outside of the range. Say that the price of the stock ends up at $35. The call option will expire worthless and the loss will be $300. The put option however, has gained considerable value, it is worth $715 ($1000 - $285). So the total gain we have made is $415 ($715 - $300).

3.5 Iron Condor

Most investments are made with the expectation that the price will go up. Some are made with the expectation that the price will move down. Unfortunately, it is often the case that the price doesn't do a whole lot of moving at all. Wouldn't it be nice if you could make money when the markets didn't move? Well, you can. This is the beauty of options, and more specifically of the strategy known as the *iron condor.* Iron condors sound complicated, and they do take some time to learn, but they are a good way to make consistent profits. In fact, some very profitable traders exclusively use iron condors. There are two ways of looking at it. The first is as a pair of strangles one short and one long, at outer strikes. The other way of looking at it is as two credit spreads: a call credit spread above the market and a put credit spread below the market. It is these two "wings" that give the iron condor its name. These can be placed quite far from where the market is now, but the strict definition involves consecutive strike prices.

3.5.1 What is a credit spread

A credit spread is the difference in yield between a U.S. Treasury bond and a debt security with the same maturity but of lesser quality. A credit spread can also refer to an options strategy where a high premium option is sold and a low premium option is bought on the same underlying security like in [12]. This creates the credit, with the hope that both options expire worthless, allowing you to keep that credit. As long as the underlying does not cross over the strike price of the closer option, we get to keep the full credit. Credit spreads between U.S. Treasuries and other bond issuances are measured in basis points, with a 1% difference in yield equal to a spread of 100 basis points. For example with S&P500 at $2150, one might buy the August $2220 call option (black dot below point 4 on the chart below) for $2.20 and sell the August 2205 call (black dot above point 3) for $4.20. This would produce a credit of $2 in our account. This transaction does require a maintenance margin. The broker will only ask that we have cash or securities in our account equal to the difference between the two strikes minus the credit we received. The margin requirement for 1 SNP short call and 1 SNP long call will be found by taking the difference of the short and the long call and multiply by the number of contracts and the multiplier (100) because every contract is equal to 100 options. So $2220 – $2150 = $15, thus the maintenance margin = (15-2)*1*1*100 = $1300 for each spread. If the marker closes below $2205, we keep the $200 for a 15% return.

To create a full iron condor all we need to do is add the credit put spread in a similar manner. Buy the August $2065 put (black dot below point 1) for $5.50 and sell the September the September $2080 (black dot above point 2) for $6.50, for another $1 of credit. Here the maintenance margin is $1400 with $100 credit (for each spread). Now we have a full iron condor. If the market stays between $2080 and $2205 we will keep the full credit, which is now $300. The requirement will be $2,700. Our potential return is 11.1% for less than two months. Because this does not presently meet the Securities and Exchange Commission's strict definition of an iron condor, we will be required to have the margin on both sides.

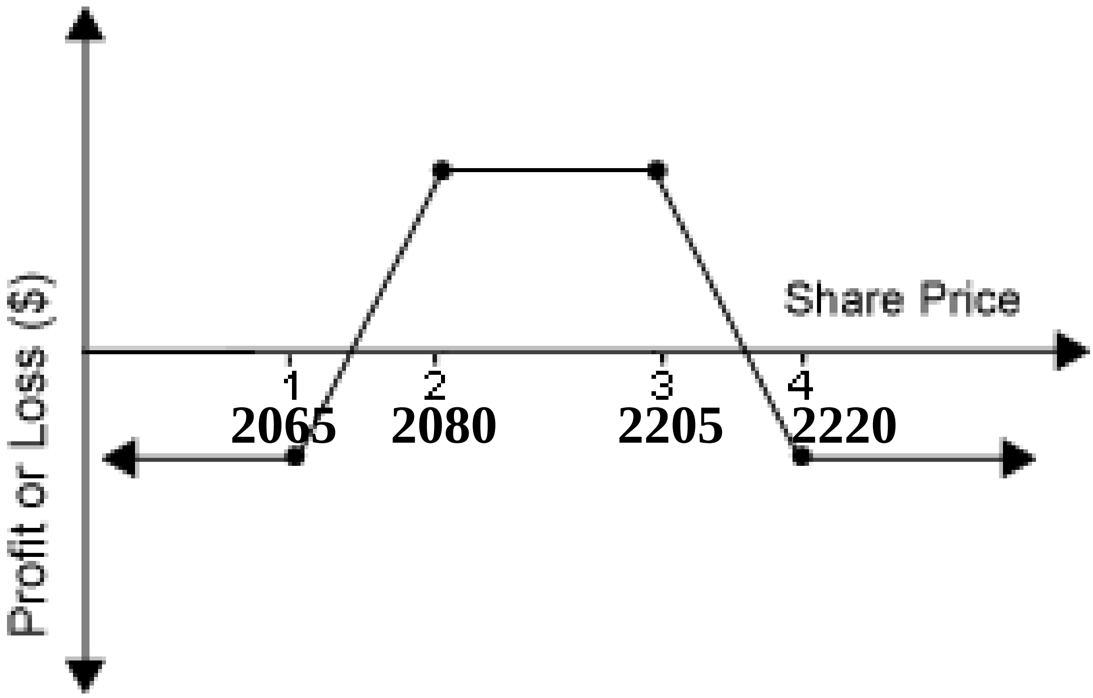

Figure 1: Iron condor chart.

Chapter 4

The Greeks

The Greeks measure the sensitivity of the value of a derivative or a portfolio to changes in parameter value(s) [11] while holding the other parameters fixed. They are partial derivatives of the price with respect to the parameter values. One Greek, “gamma” (as well as others not listed here) is a partial derivative of another Greek, "delta" in this case. The Greeks are important not only in the mathematical theory of finance, but also for those actively trading. Financial institutions will typically set (risk) limit values for each of the Greeks that their traders must not exceed. Delta is the most important Greek since this usually confers the largest risk. Many traders will zero their delta at the end of the day if they are speculating and following a delta-neutral hedging approach as defined by Black–Scholes.
The Greeks for Black–Scholes are given in closed form below. They can be obtained by differentiation of the Black–Scholes formula above:

		Calls	Puts
Delta (δ)	$\frac{\partial C}{\partial S}$	$N(d1)$	$N(d1) - 1$
Gamma (γ)	$\frac{\partial^2 C}{DS^2}$	$\frac{N'(d1)}{S\sigma\sqrt{T-t}}$	
Vega	$\frac{\partial C}{\partial \sigma}$	$SN'(d1)\sqrt{T-t}$	
Theta (θ)	$\frac{\partial C}{\partial t}$	$-\frac{SN'(d1)\sigma}{2\sqrt{T-t}} - rKe^{-r(T-t)}N(d2)$	$-\frac{SN'(d1)\sigma}{2\sqrt{T-t}} + rKe^{-r(T-t)}N(-d2)$
Rho (ρ)	$\frac{\partial C}{\partial r}$	$K(T-t)e^{-r(T-t))}N(d1)$	$-K(T-t)e^{-r(T-t))}N(-d2)$

Table 1: Black-Scholes Greeks Calculation

4.1 Delta:

It is the percentage an option will increase or decrease in value in relation to the underlying stock. For example a delta of .60 or 60% means the option will move or change in value equal to 60% of the underlying stock's price change, which means a $1.00 rise in the stock should a 60-cent rise in the option premium. If the stock fell by $1.00, the option should decrease by 60-cents. Moreover the *delta* will change (either increase or decrease), in general based on how *in-the-money* or *out-of-the-money* your option becomes.

For example if a stock is trading at $85 and you had a $95 *out-of-the-money* call option with 4 months of time on it; that option might have a delta of .41 or 41%. Let's also say that the call option was priced $6. If the stock increased by $10, this means that the out-of-the-money call option with a *delta* of 41% would've increased by $4.1. So as our options gets further in-the-money the delta will increase up to 100%, likewise the option gets further out-of-the-money the delta will decrease. Delta values can also be negative. Put options deltas always range from -1 to 0. For example if a put option has a delta of -0.23, then if the price of the underlying asset will increase by $1, the price of the put option will decrease by 23-cents.

4.2 Gamma

Gamma is one of the most obscure Greeks. *Delta*, *Vega* and *Theta* get the most attention, but Gamma has important implications for risk, in options strategies. First, though let's review what gamma represents. *Gamma* represents the rate of change of *Delta*. But since *Delta* is not fixed and will increase or decrease as mentioned above, it needs its own measure, which is *Gamma*. When we incorporate a *Gamma* risk analysis into our trading we learn that two deltas of equal size may not be equal in outcome. The *Delta* with the higher *Gamma* will have a higher risk (and potential reward of course) because given an unfavorable move of the underlying; the *Delta* with the higher *Gamma* will exhibit a larger adverse change.

Figure 1 reveals that the highest *Gammas* are always found on *at-the-money* options with the January 110 call showing a Gamma of 5.58. The same can be seen for the 110 puts. The risk/reward resulting from changed in *Delta* are highest at this point. In terms of position a seller of put options would face a negative *Gamma* (all selling strategies have negative *Gammas*) and the buyer of puts would acquire a negative *Gamma* (all buying strategies have negative *Gammas*). But all Gamma values are positive because the values change in the same direction as *Delta.* Signs change with positions or strategies because higher *Gammas* mean greater potential loss for sellers and for buyers greater potential gain.

Options	JAN <20>			FEB <48>			APR <111>			JUL <202>		
130.0 calls	0.04	0.14	0.30	0.13	0.64	2.48	0.88	1.43	13.0	2.22	1.50	21.9
125.0 calls	0.08	0.73	1.99	0.35	1.52	7.58	1.63	1.90	20.9	3.30	1.71	29.3
120.0 calls	0.30	2.37	8.78	0.98	2.68	17.9	2.82	2.27	30.9	4.91	1.85	37.8
115.0 calls	1.1[illegible]	4.74	26.0	2.24	3.60	33.7	4.61	2.45	42.6	6.84	1.89	46.9
110.0 call>	[illegible].10	5.58	52.8	4.41	3.74	52.4	7.00	2.40	54.7	9.26	1.82	56.1
Out of the money Gammas	6.36	3.96	77.0	7.67	3.10	69.7	10.00	2.13	66.2	12.26	1.66	64.9
	10.58	1.99	90.6	11.56	2.14	82.6	13.55	1.74	76.0	15.56	1.43	72.8
	15.30	0.83	96.5	15.97	1.30	90.7	17.56	1.31	83.7	19.27	1.18	79.6
90.0 calls	20.29	0.31	98.8	20.60	0.71	95.4	21.86	0.93	89.3	23.27	0.93	85.0
130.0 puts	20.20	0.00	-100	20.05	0.00	-100	20.40	1.43	-87.0	21.20	1.50	-78.1
125.0 puts	15.07	0.00	-100	15.20	1.52	-92.4	15.94	1.90	-79.2	16.98	1.71	-70.7
120.0 puts	10.[illegible]9	2.37	-91.2	10.77	2.68	-82.1	11.98	2.27	-69.1	13.47	1.85	-62.2
115.0 puts	5.9[illegible]	4.74	-74.0	7.04	3.60	-66.3	8.74	2.45	-57.4	10.40	1.89	-53.1
110.0 puts>	2.90	5.58	-47.2	4.24	3.74	-47.6	6.10	2.40	-45.3	7.84	1.82	-43.9
105.0 puts	1.20	3.96	-23.1	2.32	3.10	-30.3	4.10	2.13	-33.8	5.78	1.66	-35.1
100.0 puts	0.43	1.99	-9.45	1.22	2.14	-17.4	2.72	1.74	-24.0	4.19	1.43	-27.2
95.0 puts	0.15	0.83	-3.47	0.60	1.30	-9.27	1.75	1.31	-16.3	3.03	1.18	-20.4

Figure 2: IBM options Gamma Values. Values taken on Dec. 29, 2007.

4.3 Vega

Vega is a measure of the impact of changes in the underlying volatility on the option price. Specifically, the *Vega* of an option expresses the change in the price of the option for every 1% change in the volatility. Options tend to be more expensive when volatility is higher. Thus, whenever volatility goes up, the price of the option goes up and vice versa. Therefore, when calculating the new option price due to volatility changes we, add the *Vega* when the volatility goes up but subtract it when volatility drops. For example a stock is trading at $46 in July and an August $50 call is selling for $2. Let's assume that the *Vega* of the option is $0.15 and that the underlying volatility is 25%. If the underlying volatility increased by 1% to 26%, then the price of the option should rise to $2 + $0.15 = $2.15. However if the volatility had gone to down by 3% to 22% then option price should be $2 – (3 x 0,15) = $1,55. The more time remaining to time expiration, the higher the *Vega*. This makes sense as *time value* makes up a larger proportion of the for longer term options and it is the *time value* that is sensitive to changes in volatility.

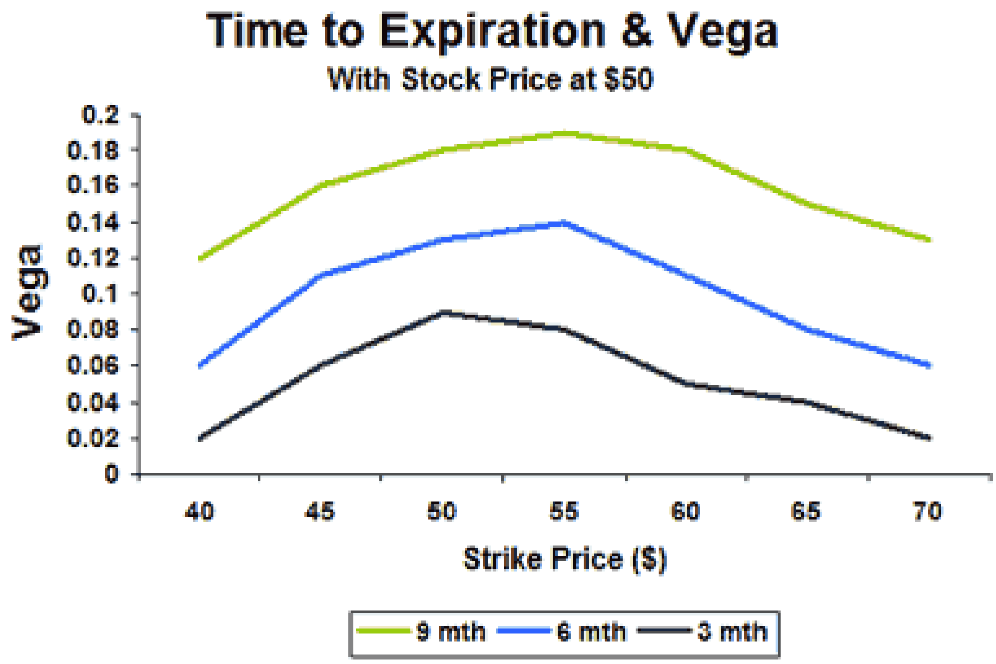

Figure 3: The behavior of the Vega of options at various strikes expiring in 3, 6 and 9 months time when the stock is currently trading at $50

4.4 Theta

The *Theta* is a measurement of the option's time decay. The theta measures the rate at which options lose their value, specifically the time value, as the expiration date draws nearer. Generally expressed as a negative number, the theta of an option reflects the amount by which the option's value will decrease every day. For example a call option with a current price of $3 and a theta of -0.05 will experience a drop in price of $0.05 per day. So in two days time, the price of the option should fall to $2.90. Longer-term options have theta of almost 0 as they do not have lose value on daily basis. Theta is higher for shorter-term options, especially ***at-the-money*** options. This is pretty obvious as such options have the highest *time value* and thus have more premium to lose each day. In contrast, theta goes up dramatically as options near expiration as time decay is at its greatest during that period. In general options of volatility stocks have higher *Theta* than low volatility stocks. This is because time value premium on these options are higher and so they have more to lose per day.

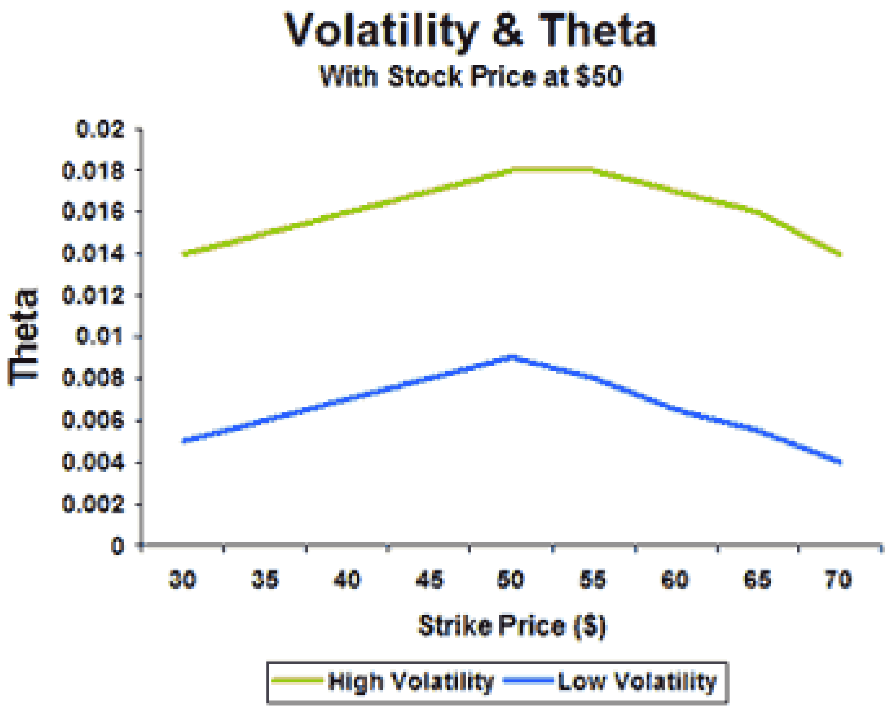

Figure 4: High volatility options have higher Theta than low volatility options.

4.5 Rho

Rho is the rate at which the price of an option changes relative to a change in the risk-free rate of interest. Rho measures the sensitivity of an option or options portfolio to a change in interest rate. For example, if an option or option portfolio has a rho of 1, then for every percentage-point increase in interest-rates the value of the option increases 1%. So let's assume that a call option is priced at $4 and has a rho of 0.25. If the risk-free rate rises let's say from 3% to %4, the value of the call option would rise from $4 to $4.25.

Chapter 5

Pricing Boundaries

We will now discuss rational pricing boundaries for European option prices. These are upper and lower bounds for the price of an option, and they represent the range of values the option price can assume if we require there to be no arbitrage opportunities. The reason we discuss this for European, and not for American options, is because European options can only be exercised on expiration date, and not before. American options additionally contain the problem of *optimal time of exercise*, although in the absence of dividends this will be the expiration date due to the possibility of favorable developments in the stock in the future. In the case of a dividend it might be more profitable to exercise a call option early, since then the dividend can be earned with the stock in possession. A second reason for discussing European options is that the Black-Scholes equation actually is a model for European options only.

Arbitrage is the possibility to realize *risk-less* profits without any initial investment. These are mostly possible if there is a discrepancy in value between two identical investments, in which case the arbitrageur can buy the cheaper one, and sell it for the higher price. It is assumed for the analysis that this is not possible for a longer period of time. The standard argument for this is that investors will quickly notice these opportunities and take advantage of them, in which case they will disappear quickly. Competition crowds out arbitrage opportunities so that in most cases equilibrium prices should prevail. The no- arbitrage arguments involved in the following paragraph will nicely illustrate the nature of economic reasoning useful for understanding the arguments later on in the derivation of Black-Scholes.

The first such rational pricing considerations is given by the upper bound on the value of a call, namely

$$C \leq S \quad (17)$$

where C is the price of the call option and S the price of the stock as mentioned above. To understand this, suppose C was greater than S. Then the holder of the call could realize a risk- less profit by buying the stock for S, and then selling the option for C. Another way of seeing this is that it should not be more expensive to buy the right to buy a stock than it is to buy the stock itself. On the other hand, we have

$$P \leq K \quad (18)$$

where P is the price of the put and K is the strike price. If it were the case that K is less than P, then an arbitrageur could easily realize a profit by writing a put, and investing the profit at the risk-free rate. He would then have max(P − K, P)e^{-rT} , where T is the time until expiration since the only thing he committed to is possibly buying the stock for K at some future time, and even that only if the option holder would actually exercise the call. By requiring that there are no such arbitrage opportunities, we actually arrive at the slightly stronger condition,

$$P \leq Ke^{-rT} \quad (19)$$

since in the absence of arbitrage opportunities, the present discounted value of the strike price should exceed the price of the put. The present discounted value in this case is the amount necessary to invest in bonds at the present time that would yield K at expiration date, T years from now, if continuously compounded with annual rate r.

Now, to prove what the lower pricing bound for a European call is, let us assume that there are two portfolios, one containing a call with strike price K and Ke^{-rT} in cash, the other containing one share of the stock. Assume we are T years away from expiration date. Then, at expiration date, if the stock price is greater than the strike price, then we exercise our call and realize a profit of S − K, where S is the stock price on the expiration day. Our cash has however changed to K in the meanwhile, because of the interest accumulated, and our portfolio leaves us with S. On the other hand, if the stock price is less than the strike price, we do not exercise the option, and we are left with the cash. We see that the first portfolio hence yields max(S, K). The second portfolio just yield S, the price of the stock at expiration date. We hence see that the first portfolio is always at least as large as the second, so that in the absence of arbitrage, $S \leq C + Ke^{-rT}(20)$. Putting this fact together with the last inequality yields

$$C \geq \max(S - Ke^{-rT}, 0) \quad (21)$$

Similarly, the lower pricing bound for a European put will be

$$P \geq \max(Ke^{-rT} - S, 0) \quad (22),$$

remembering that the value of an option is always nonnegative.

5.1 Put-Call Parity

In this section we will derive the so called put-call parity, which will allow us to determine the value of the call, given the value of the put and vice versa, requiring that they both are options on the same stock, with the same strike price and expiration date. It will prove very useful in the derivation of the Black-Scholes equation, since we will only have to derive it for one option type, and can then use the put-call parity to make a couple of algebraic manipulations and arrive at the formula for the second option type. In addition, we can focus on one specific type of option in our empirical analysis later on, since we could get essentially the same results for the other type by using the parity relationship. Suppose again that we have two portfolios, the first one containing a European call option and cash worth Ke^{-rT} and the second one containing a put and a share. The same logic from before applied here will yield that on expiration date, both portfolios are worth max(S, K). Since these options are both European, they can only be exercised on this date anyways, so that both portfolios actually have the same value on any day during which they exist. So, the formula of the put-call parity is given below:

$$C + Ke^{-rT} = P + S \quad (23)$$

Chapter 6

Black-Scholes implementation

6.1 Application description

6.1.1 Needs and Solutions

The basic *need* of an application that broadcasts stock prices is primarily how fast is able of transmitting them. High frequency trading firms also known as HTFs need the price changes of a stock extremely fast. One way to achieve this some HFTs firms are placed very close the exchanges in order to receive the stock prices faster than the others. These prices are travelling with the speed of light, so it is all about distance. But since it is impossible all the hedge funds and HTFs to be placed near the exchanges, another way of receiving stock price changes very fast is through smart parallel algorithms and huge distributed systems. In our case, since we can't have an immediate access to an exchange, the stock prices are parsed concurrently from Google Finance using threads.

6.1.2 User interface

After studying the Black-Scholes formula and some of the strategies that can be applied with options, we are going to offer a detailed description of the Java network application. Basically, simulates an environment of a multithreaded stock application that helps us to implement the Black-Scholes formula. The application features two windows:

- The first window simulates an environment of a stock application by showing the stock prices of ten different stocks (we can add as many stocks as we want due to scalability of the application). The stocks prices are being downloaded and parsed concurrently from Google Finance using the *ExecutorsService* which is described below. This window has meaning from 4:30 pm to 11:00 pm EEST due to the seven hour difference that Athens is ahead of New York. At 9:30 am EDT the New York stock exchange also known as NYSE, opens and stock prices are being downloaded, parsed and printed randomly in the JTextfields due to the fact that threads are executed in a different order every time. The prices printed in the JTextfield are in complete synchronization with the ones in Google Finance, actually in most of the time this parsing algorithm which will be given in the appendix, downloads the prices faster than they change in Google Finance. Therefore in this window we see the stock prices move faster than they do in Google Finance, but this has it's drawbacks also. Because in this application we create a URL connection every time that we download a stock price, in the end we send hundreds of requests in a very short period of time. As result if our application runs for a large period of time we get a 504 connection error. Ten out of nine stocks are some of the largest and most established companies in the S&P100. The tenth stock is the index S&P100.

- The second window is a Black-Scholes calculator in order to price an option premium (call or put) for these ten stocks or for any option we would wish. In this window we enter the five parameters of the Black-Scholes formula (stock price S, strike price K, time to expiration T, volatility σ, and the risk-free rate r) and can we calculate the option premium of our preferred stock. If we don't choose any of the existing stocks or the selection "Any" and press the call or put button the application is going to throw an exception. The main advantage of this application is that we can watch a stock price directly from the first window and then use it as input in the Black-Scholes formula. Both risk-free rate and volatility fields are auto filled, the first with the value of 4% and the latter with the implied volatility taken from iVolatility.com. Of course we can enter any value we wish in both fields. Along with the option premium (call or put) the *Greeks* are also being calculated, which are key elements as mentioned above for planning our option strategy.

6.1.3 UML class diagrams

The application consists of two classes. The first class is responsible for the number of threads that we would like to create but mainly for the assignment of each thread to the task of downloading and parsing a stock price. The second class puts in place the parsing algorithm and also creates the two windows that were mentioned above. The figure below offers a detailed representation of both classes.

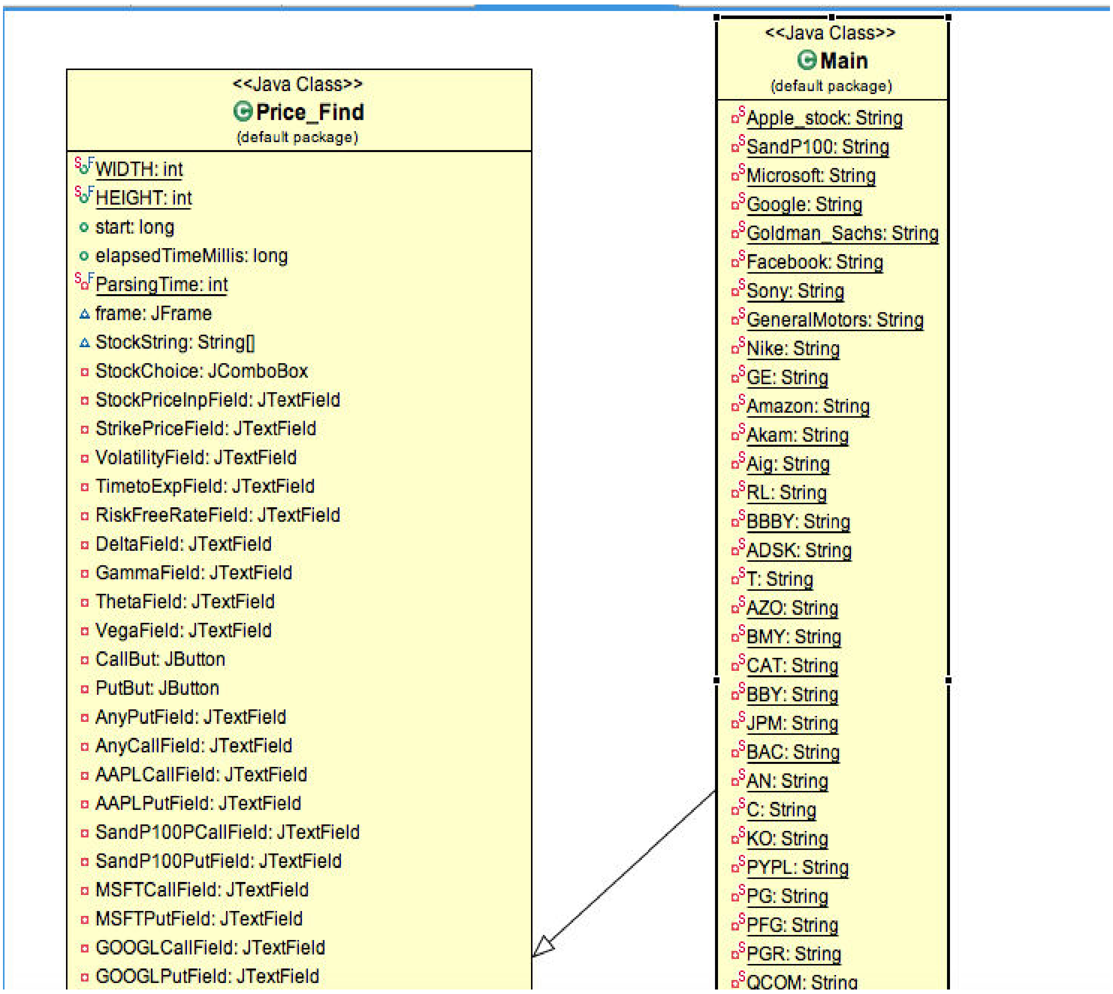

Figure 5: UML class diagram of the application. Due to the big amount of instance variables that each class features, only some of them are presented in above. Both classes have over one hundred instance variables.

6.2.1 Executors

The Concurrency API [14] introduces the concept of an ExecutorService as a higher level replacement for working with threads directly. Executors are capable of running asynchronous tasks and typically manage a pool of threads, so we don't have to create new threads manually. All threads of the internal pool will be reused under the hood for revenant tasks, so we can run as many concurrent tasks as we want throughout the life-cycle of our application with a single executor service. The class Executors provides convenient factory methods for creating different kinds of executor services. Due to the fact that the Java process never stops, Executors have to be stopped explicitly; otherwise they keep listening for new tasks. An ExecutorService provides two methods for that purpose: shutdown() waits for currently running tasks to finish while shutdowNow() interrupts all running tasks and shut the executor down immediately.

6.2.2 Implementing Executors

In our simulator we use an executor with a thread pool of size ten (as mentioned above we can add as many stocks as we want). Every thread is *assigned* with the task of downloading and parsing the price of each stock. Therefore there is a 1:1 *assignment* between stocks and threads in this application. Due to the fact that our application sends a great amount of requests in the Google Finance site as mentioned above, a constant must be used as a *time limit* which is being assigned with the milliseconds that our application will parse and download the stock prices in order to avoid getting a 504 connection error. As a result the executor shuts down softly by waiting a certain amount of time for termination of currently running tasks. After a maximum of our time limit(parsing time) the executor finally shuts down by interrupting all running tasks.

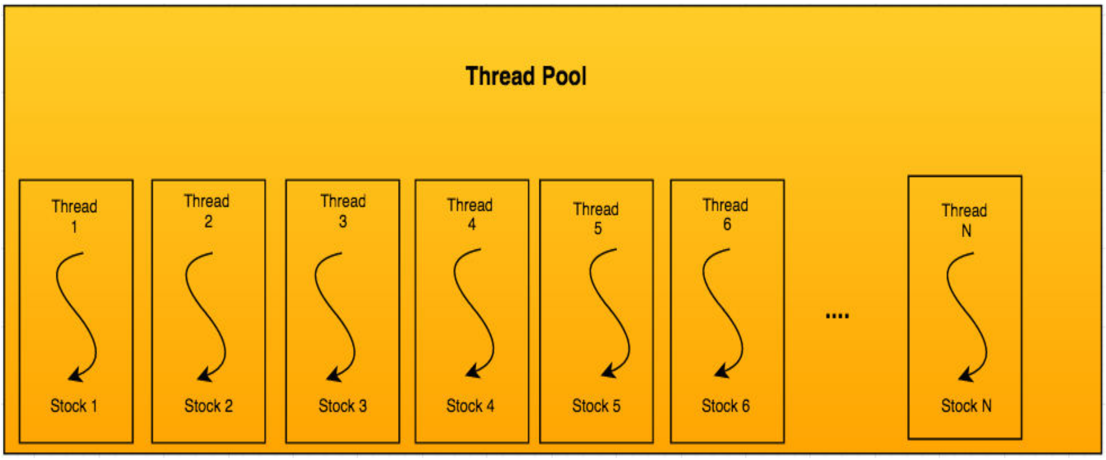

Figure 5: Executors thread pool. In our example there is a 1:1 assignment between threads and stocks, which means each thread is responsible for parsing one stock price.

6.2.3 Parsing Algorithm

In order to parse and download a stock price from the html source code of the Google Finance site we need three identifiers.

- Name of the stock,
- spanid of the stock
- and the URL link of the stock

The name of the stock let's assume for Apple is necessary in order to save in a text file the whole html code of a stock from Google Finance. The spanid of the stock is an identifier, which is essential in locating the stock price in the html code. Id is actually an attribute of the html that provides a unique identifier for an element within the document. Therefore the value of the stock is something unique in the html code of each stock. Last but not least the URL link of the stock is the key element in order to initiate a URL connection for each stock. The parsing algorithm is given in the following code.

6.3.1 Parsing Time

Next we quote a figure that describes the relation between the parsing time of all ten stocks and the time of the day when the stock market of the New York is open. We notice that half an hour after and half an hour before, the market opens and closes respectively, the algorithm spends the most time in paring the stock prices, because in both the first and the latter hours of the day there is a lot of traffic in the Google Finance site due to the fact that there is a lot of price movement. Thus the busier the site is the more time to parse the stock prices. This figure is a result from a four day testing with over 12 hours of parsing. In that four day period the parsing was taking place different hours of the day in order to have a proper calculation of the average parsing throughout the day. The characteristics of the MacBook Pro that the tests were taken place are:

- CPU: 2.3 GHz Intel core i5 with 3MB shared L3 cache
- 4 threads (two real two virtual)
- RAM: 4 GB 1333 MHz DDR3
- HDD: 320GB 5400 rpm Serial ATA
- Internet connection: 4 Mbps

and runs OSX Yosemite 10.10.5

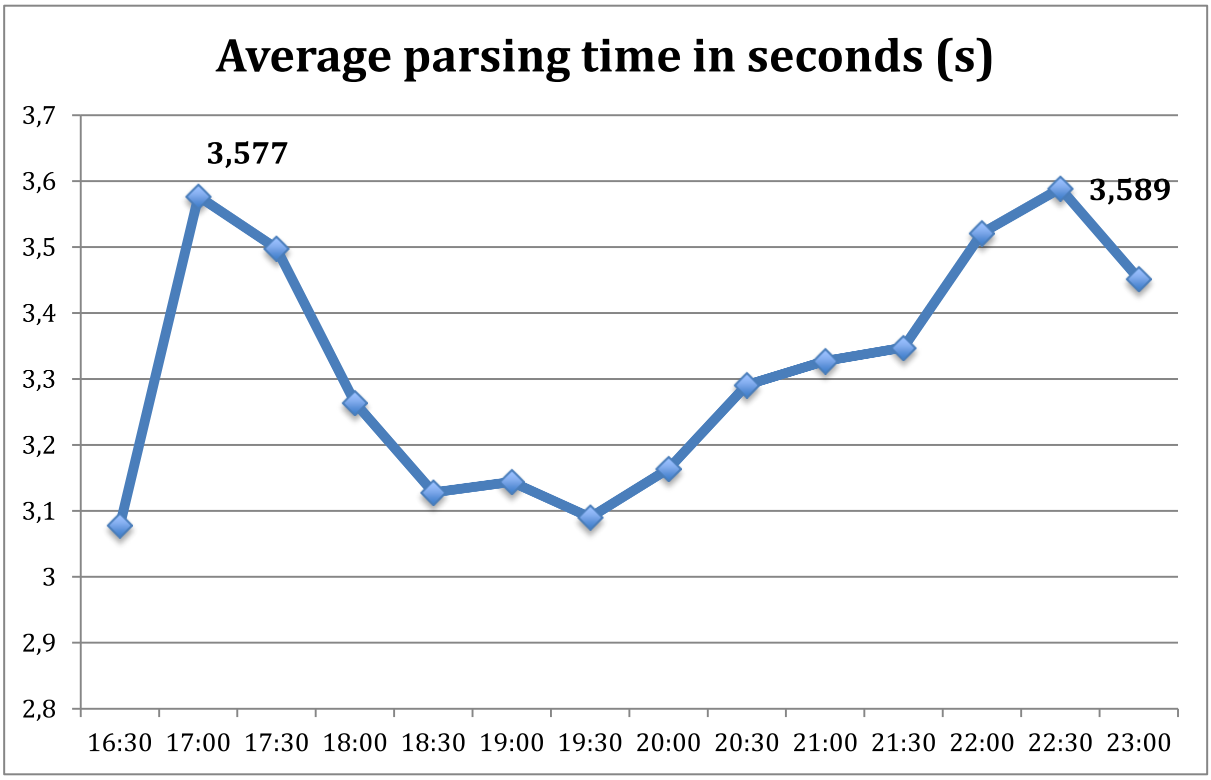

Figure 6: Average parsing time of ten stock prices from 16:30 to 23:00 EEST (from the time the NYSE opens till it closes).

6.3.2 Concurrent vs Iterative

Along with the concurrent parsing algorithm we have also implemented the iterative in order to compare the results and calculate the percentage of how faster the concurrent parsing algorithm but mainly the speedup. After repeated tests on a four-day of testing as mentioned above, the average parsing time is 5578 milliseconds. In this case the concurrent parsing algorithm is faster by: $\frac{|3311-5578|}{3311} x\ 100 = 68.4\%$. In addition the speedup is $\frac{T_1}{T_p} = \frac{5578}{3311} = 1.68$

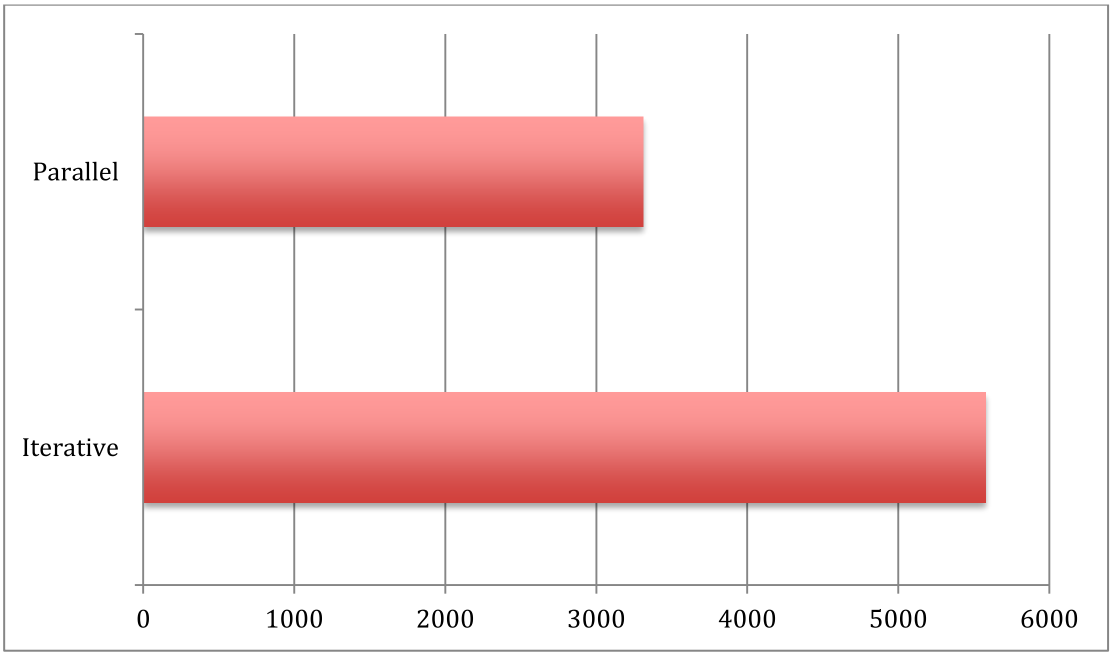

Figure 7: Average parsing time of ten stocks using iterative and concurrent parsing algorithm. As seen on the chart the concurrent algorithm is 68.4% faster than the iterative.

6.3.3 Scalability and results

In order to test the algorithm in a larger scale a Virtual Machine has been set up. The VM features:

- 24 vCPUs Intel Xeon E5-2630 0 @ 2.30 GHz
- 4 cores each CPU
- 8 threads each vCPU

The algorithm has been tested for parsing 10, 20, 40, 50 and 100 stock prices. The results are shown below.

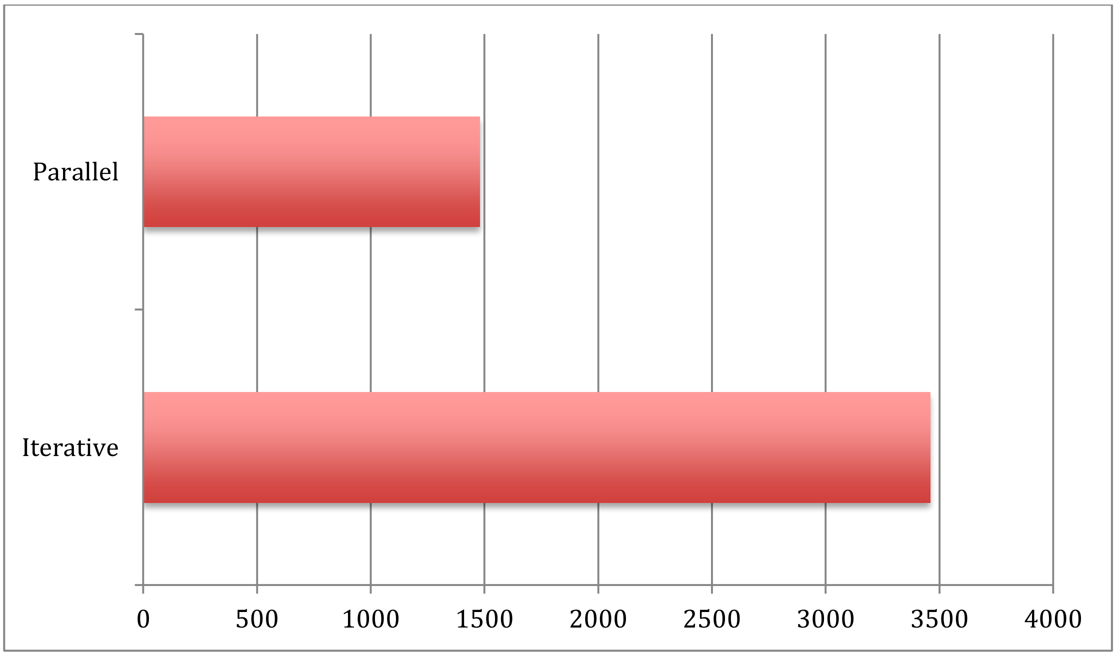

Figure 8: Average parsing time of 10 stocks of iterative and concurrent parsing algorithm using VM (virtual machine). The parallel algorithm is 133% faster than the iterative. Speedup $= \frac{T_1}{T_p} = \frac{3459.2}{1487.5} = 2.33$

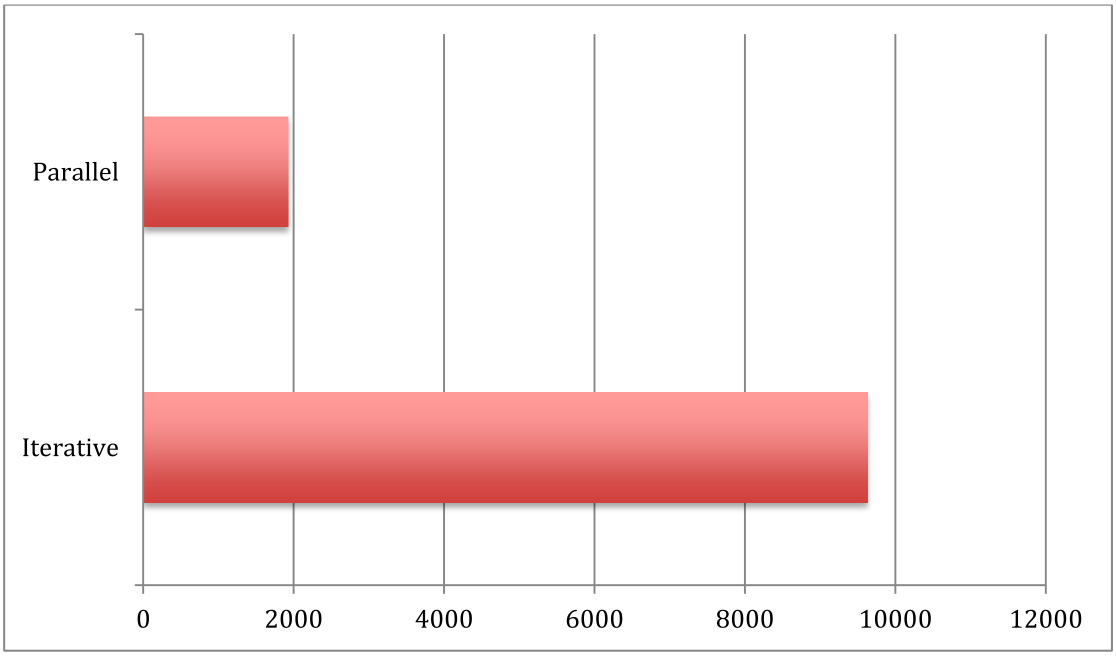

Figure 9: Average parsing time of 20 stocks of iterative and concurrent parsing algorithm using VM. The parallel algorithm is 400.4% faster than the iterative. Speedup $= \frac{T_1}{T_p} = \frac{9633}{1931} = 4.98$

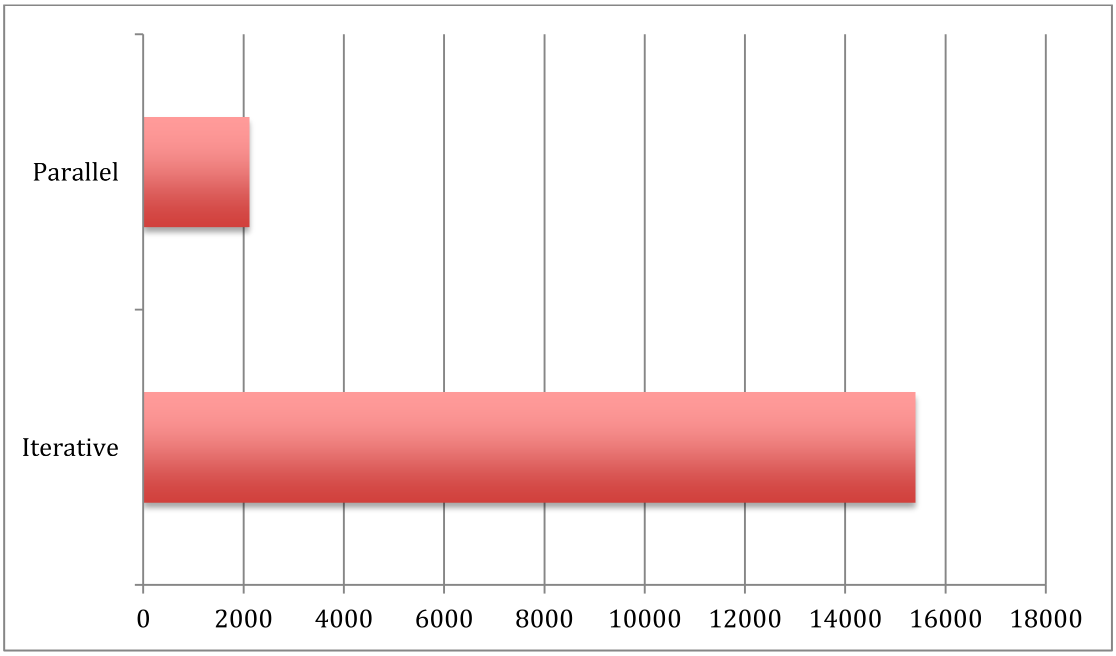

Figure 10: Average parsing time of 40 stocks of iterative and concurrent parsing algorithm using VM. The parallel algorithm is 663.16% faster than the iterative. Speedup $= \frac{T_1}{T_p} = \frac{15400}{2016.33} = 7.63$

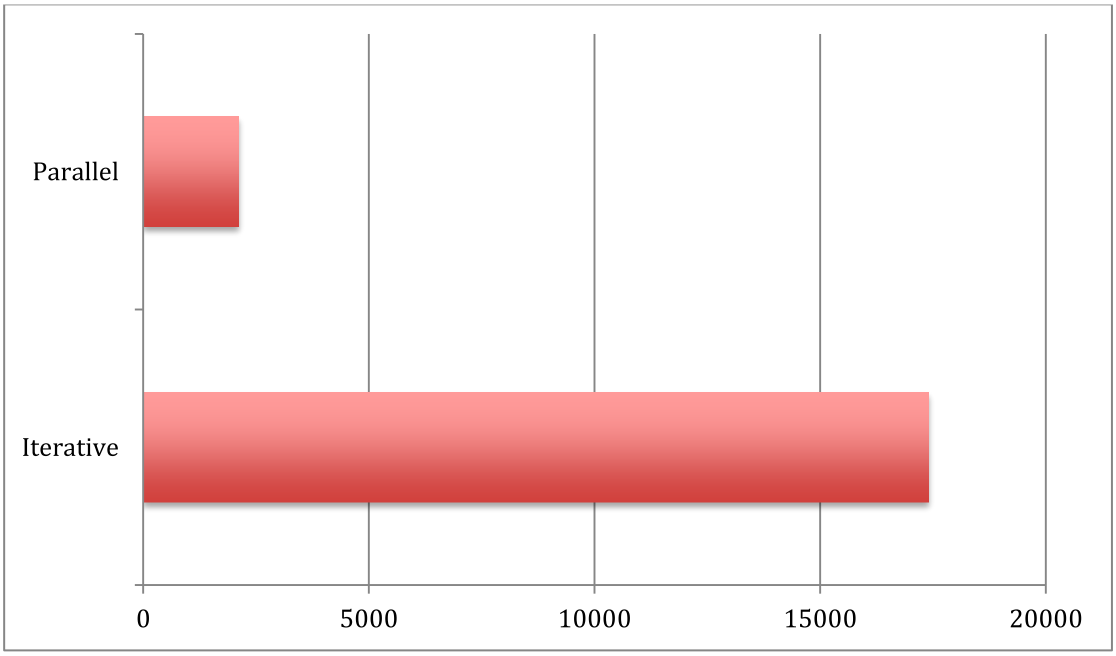

Figure 11: Average parsing time of 50 stocks of iterative and concurrent parsing algorithm using VM (virtual machine). The parallel algorithm is 722.9% faster than the iterative. Speedup $= \frac{T_1}{T_p} = \frac{17406}{2115} = 8.22$

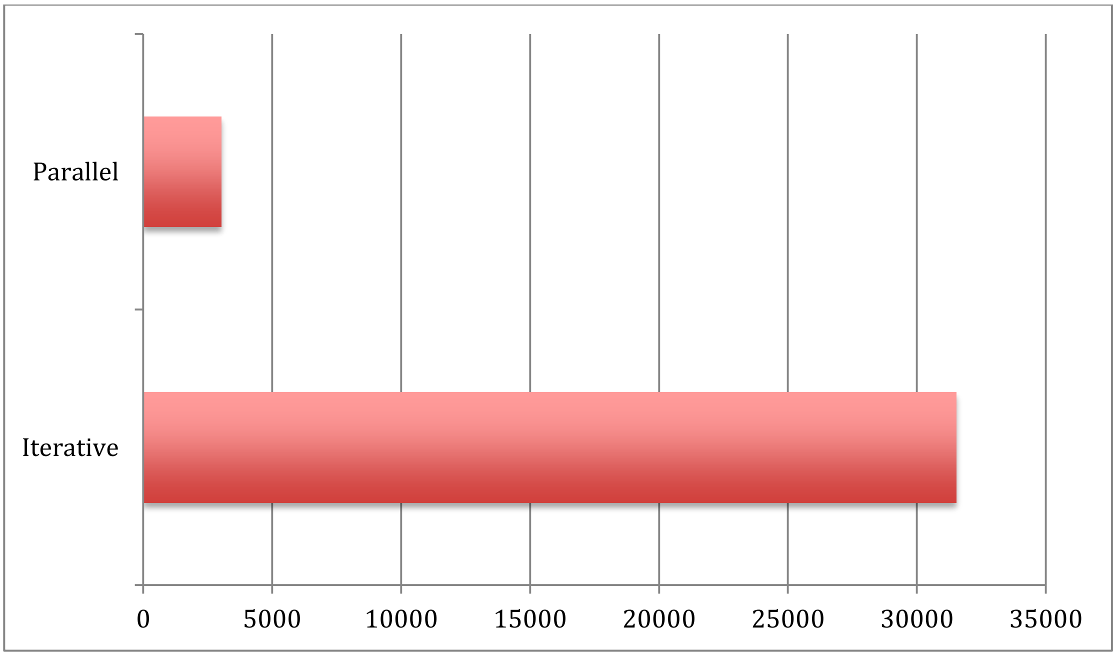

Figure 12: Average parsing time of 100 stocks of iterative and concurrent parsing algorithm using VM (virtual machine). The parallel algorithm is 941.1% faster than the iterative. Speedup $= \frac{T_1}{T_p} = \frac{31531}{3029} = 10.41$

It is obvious that the parallel algorithm is capable of scaling in parsing, up to a few thousands of stocks. When the parallel parsing algorithm is being applied to a larger number of stocks it is also suitably efficient as it was.

6.4 Bottleneck

A network bottleneck refers to a discrete condition in which data flow is limited by computer or network resources.

Although the network requests are being sent concurrently, the Network interface card (NIC) will empty its buffer sequentially. The bottleneck isn't in the cores but in the NIC.

Figure 13: An analog example of a bottleneck. In our case the cars represent the network requests and the bottle the NIC buffer respectively.

Although the NIC buffer empties sequentially, the parallel parsing algorithm is still way faster and more efficient than the iterative. And the reason is that the iterative isn't able of holding the NIC's buffers full. As a result the iterative algorithm can't take advantage of the network's full potential, while the parallel sends all requests concurrently ensuring that the NIC's buffer stays full.

Chapter 7

Conclusions-Future Work

7.1 Conclusions and results

The purpose of this thesis was dual. Firstly to study the Black-Scholes formula and some basic strategies that are being applied but primarily to present an online application in order to fully understand the formula on a practical level. The application is able of parsing stock prices from Google Finance concurrently using threads. After countless hours of parsing stock prices we 've reached the conclusions:

- It doesn't matter only how fast the Internet connection is in order to parse a stock price because that is only half the job of the parsing algorithm. The other half on the algorithm is based on how fast and efficient the processor and the hard drive of a system are. In order to decrease the average parsing time and scale the algorithm we need faster Internet connection and faster processor. This algorithm could be tested in a system in other parallel systems so that we could have a better picture of it's potential so as to examine how much we could decrease the average parsing time.
- Due to the fact that the Network interface card (NIC), empties its buffer sequentially, there have to be a machine with multiple NIC in order for the parallel algorithm to reach its full potential. So it should be recommended a cluster of as many machines as possible. On each machine have a few threads (just to ensure the NIC's buffer stays full).

Moreover as mentioned above the hard disk drive is also a crucial factor for improving this algorithm. This is because when the whole html source code is being saved from Google finance to our file system as a txt file. Then all the characters are being assigned to an array, which is basically ***a copy and paste*** process that requires some milliseconds. Thus, the faster the HDD (higher RPM or an SSD) is, the faster the parsing algorithm can become. Of course there is a bound on how much we can decrease the average parsing time.

References

[1] F. Black and M. Scholes, "The Pricing of Options and Corporate Liabilities", *Journal of Political Economy*, vol. 81, no. 3, pp. 637-654, 1973.

[2] T. Heimer and S. Arend, "The genesis of the Black-Scholes option pricing formula",*Frankfurt School – Working Paper Series*, vol. 98, pp. 11-12, 2008.

[3] E. Turner, "M.S. Joshi. The Concepts and Practice of Mathematical Finance", Chicago, 2006.

[4] S. Nickolas, "What is an option's implied volatility and how is it calculated? "Investopedia", *Investopedia*, 2015. [Online]. Available: http://www.investopedia.com/ask/answers/032515/what-options-implied-volatility-and-how-it-calculated.asp

[5] A. Damodaran, "What is the Riskfree Rate? A Search for the Basic Building Block", *SSRN Electronic Journal*, pp. 5-7.

[6] P. Jacquinot and N. Sukhomlin, "A direct formulation of implied volatility in the BlackScholes model", *Journal of Economics and International Finance*, vol. 2, no. 6, pp. 95-101, 2010.

[7] P. Gross, "Parameter Estimation for Black-Scholes Equation", URA, Chicago, 2006.

[8] M. Hemler and T. Miller Jr, "The Performance of Options-Based Investment Strategies: Evidence for Individual Stocks During 2003–2013", 2015.

[9] C. McKhann, "Options Trading With The Iron Condor | Investopedia", *Investopedia*, 2006. [Online]. Available: http://www.investopedia.com/articles/optioninvestor/06/ironcondor.asp.

[10] "Risk-Free Rate Of Return Definition | Investopedia", *Investopedia*, 2003. [Online]. Available: http://www.investopedia.com/terms/r/risk-freerate.asp.

[11] "Greeks Definition | Investopedia", *Investopedia*, 2003. [Online]. Available: http://www.investopedia.com/terms/g/greeks.asp.

[12] "Credit spread Definition | Investopedia", *Investopedia*, 2003. [Online]. Available: http://www.investopedia.com/terms/c/creditspread.asp.

[13] "Strangle Definition | Investopedia", *Investopedia*, 2003. [Online]. Available: http://www.investopedia.com/terms/s/strangle.asp.

[14] "Executors (The Java™ Tutorials > Essential Classes > Concurrency)", *Docs.oracle.com*, 2016. [Online]. Available: https://docs.oracle.com/javase/tutorial/essential/concurrency/executors.html.

[15] "Implied Volatility (IV) Definition | Investopedia", *Investopedia*, 2003. [Online]. Available: http://www.investopedia.com/terms/i/iv.asp.

[16] "Long Straddle Definition | Investopedia", *Investopedia*, 2006.[Online]. Available: http://www.investopedia.com/terms/l/longstraddle.asp.

Appendix

Further Application description

As mentioned above, in the appendix section there will be some further application description. In order to have a better understanding of the application we will offer an example including some screenshots. Let's assume that we want to see how the stock market is going today by watching the stock prices and then plan a strategy involving options. The first step would be to get the most recent market stock prices.

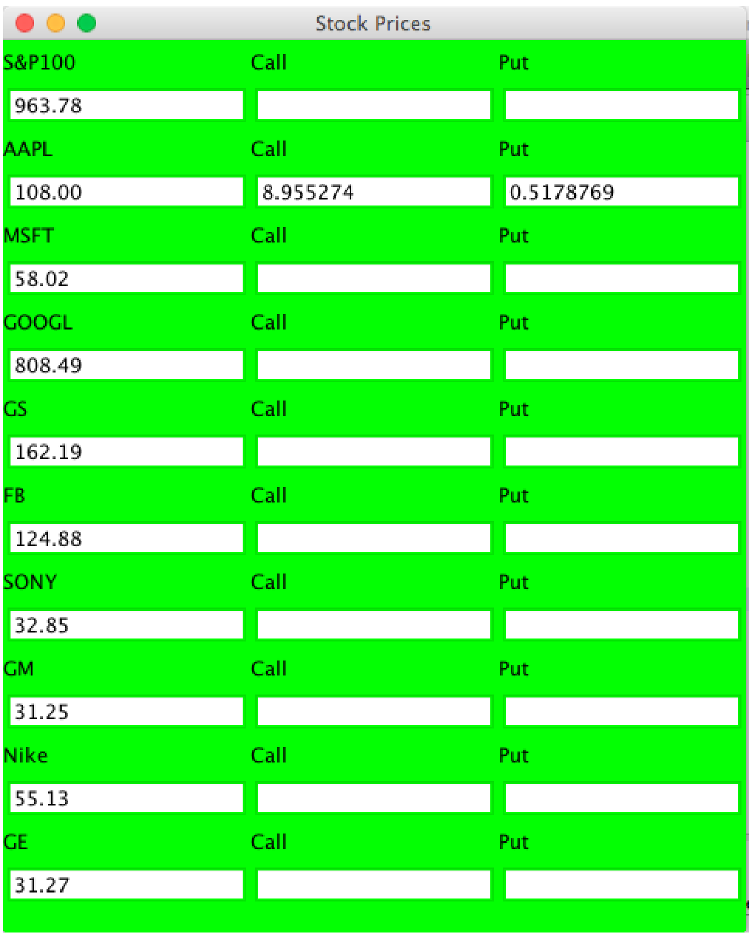

Figure 14: The concurrent parsing algorithm in action. The prices printed in the fields are in total synchronization with the prices at Google Finance.

Now that we know the current market price of the stock, we want to calculate the option premium as well as the Greeks in order to plan a strategy involving option. Here comes the second window to below in order to calculate the premium (put or call). Below we can see an example for Apple which goes by the name as AAPL. The current stock price is $108 on 10th of August. Let's choose an *in-the-money* option with a strike price K = $100, time to expiration = 40 days, implied volatility equal to σ = 22.23% (1 month ago taken from iVolatility) and a risk-free rate of 4%. As we can see above on figure 7 the option premium for a call priced at $8.95 while the put at $0.51 because an out-of-the-money money. Along with the option premium the key elements for planning our strategy known as *the Greeks* are also being calculated. If we select the option *Any* then the option premiums are printed under the call and put buttons respectively.

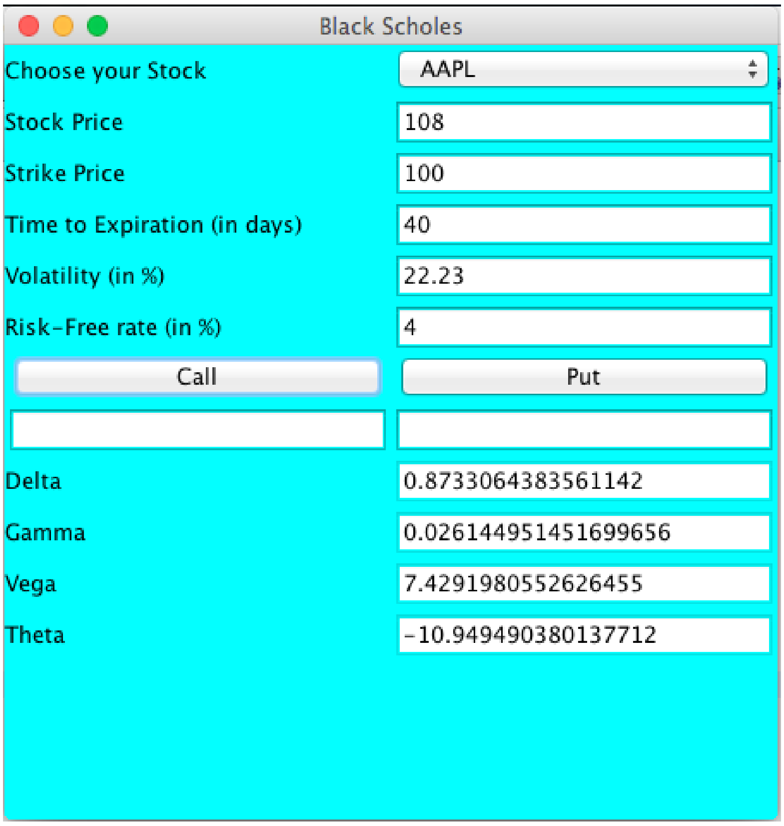

Figure 15: AAPL example for pricing a premium and calculating the Greeks using Black-Scholes formula.

In what follows the reader can examine the concurrent parsing algorithm and how the threads are being initiated.

Parsing Algorithm: Concurrent downloading and parsing of a stock price from Google Finance for a thread pool of size N.

```
GoogleFinanceUrl = new URL(thePath);

GoogleFinanceUrc  = GoogleFinanceUrl.openConnection();
GoogleFinanceinp = GoogleFinanceUrc.getInputStream();
output = new PrintWriter(new FileOutputStream(Out));
while( (c = GoogleFinanceinp.read()) != -1) {
  output.print((char) c);
}
output.close();
GoogleFinanceinp.close();
f1 = new File(Out);
l = f1.length();
char[] GoogleFnResidentFile = new char[(int) l];
i = 0;
c = 0;
l = 0;
GoogleFninp = new FileInputStream(Out);
while( (c = GoogleFninp.read()) != -1 ) {
 GoogleFnResidentFile[i] = (char) c;
  i++;
}
GoogleFninp.close();
i = 0;
c = 0;
GoogleFnWholeFile = new String(GoogleFnResidentFile);
StartingPoint = GoogleFnWholeFile.indexOf(spanId, 0);
EndingPoint =
GoogleFnWholeFile.indexOf("</span>",StartingPoint);
StocksPrice = GoogleFnWholeFile.substring(StartingPoint +
spanId.length(), EndingPoint);
```

where the thePath is the Url link in order to start a URL connection in Google finance for each stock.

Parsing Algorithm initiation: The code below initiates the pool of N threads by assigning in each thread the parsing of each stockPrice.

```
for (int i = 0; i<ThreadNum; i++) {
final Future<String> price  = (Future<String>) threadPool.submit(new
Price_Find(Urls[i], Outs[i], SpanIds[i] ) );
}
```

By the time the loop will have finished each thread is assigned with the task of downloading and parsing the stock price from Google Finance. As seen in the above code the function *submit* is responsible for initiating the threadPool. The three identifiers which are described above are the parameters passed in the function *submit.*