Parameter Estimation for Black-Scholes Equation

Peter Gross
Advisor: Dr. Jialing Dai

Final Report
URA Spring 2006

Abstract

The Black-Scholes equation is a hallmark of mathematical finance, and any study of this growing field would be incomplete without having seen and understood the logic behind this equation. The initial focus of this paper will be to explore the arguments leading to the equation and the financial background necessary to understand the arguments. The problem of estimating the only parameter which is not observable directly in the market, the volatility, is then tackled through two different methods: historical volatility and implied volatility. The goal is then to determine the "best" way to estimate volatility, by comparing the theoretical price the equation predicts with the actual price in the market, based on data from the Chicago Board Options Exchange on six selected stock options.

1 Introduction

The Black-Scholes equation was first presented in [6]. It was for this work that Scholes received the Nobel Prize of Economics in 1997 (Black had passed away two years earlier). The problem presented in this paper was one of finding the "fair" value of a stock option. What a stock option actually is, and what exactly fair means in this context, will be discussed later on. The equation is widely seen to have paved the way for an influx of mathematical sophistication into certain aspects of finance. Furthermore, it has spawned the field of financial engineering, which is concerned with the design of financial contracts and the pricing of derivatives. The pricing formula for a put option is shown below

$$\begin{aligned} P(S,T,K,r,\sigma) &= Ke^{-rT}N(-d_2) - SN(-d_1) \\ d_1 &= \frac{\ln(S/K) + (r+\sigma^2/2)T}{\sigma\sqrt{T}},\ d_2 = d_1 - \sigma\sqrt{T} \end{aligned} \tag{1}$$

Here, N is the cumulative standard normal distribution function, r is the current risk-free rate of return, S is the current stock price, K is the strike price, T is the time until expiration in years, and σ is the (annualized) volatility of the stock.

Section 2 will discuss the necessary finance background for understanding this project. It will also introduce the reader to the type of economic reasoning which - apart from a good mathematics background - is valuable to know for study of the field of mathematical finance. Section 3 will contain an introduction to the relevant mathematical concepts, which are valuable in understanding the equation and some of which will be used in the derivation of the pricing equation in Section 4. Section 5 will discuss how the data was obtained, and give a general overview of how to work with the data base. Since volatility is the only parameter in the equation which is not directly observable in the market, this paper will discuss different ways of estimating it. Volatility can be estimated using historical data, by calculating it from historical prices. One goal of this paper will be to vary the time scale one takes into account, i.e. to change the time windows from which prices are used to calculate the volatility. In section 6, we will then look at which time scale seems to be optimal in the sense of approximating the option price as near as possible empirically, based on eight stocks. Section 7 will then investigate a second way of estimating volatility, by looking at so-called implied volatility. To calculate this, we need to take the option price on a specific day, and then find out which volatility would produce this option price, given the other known parameters in the model.

Implied volatility is often interpreted as representing what investors think volatility is at any given point of time. Since we cannot explicitly solve the equation for σ, we need to rely on numerical techniques, which we borrowed from [3], to approximate implied volatility. We will then determine this measure of volatility at different points of time, and look at how it compares directly to historical volatility up to that point. The goodness of fit for this - as in the case of historical volatility - will be the amount to which the calculated option prices deviate from the actual prices in the market.

In the conclusion of the paper we will discuss some pitfalls of the Black-Scholes model, which have been already identified by the literature and lead to developments of new models. We will also check if our analysis corroborates these flaws, and what assumptions specifically have to be revised, based on our results.

2 Finance Background

2.1 Definitions and basic Facts

2.1.1 Basic Terminology

In what follows, we will introduce the reader to the concepts and terminology used in finance, as far as it is important for this paper. A *derivative* is a financial instrument whose value depends on the value of some underlying variable. 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 of a derivative is that its value is well-defined, given the value of the underlying variable. Among

these derivatives, the regular stock option is the one we will discuss 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 conferred 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, while 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.

2.1.2 Value of an 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 $\max(0, K - S)$, or $\max(0, S - K)$, for the case of a put and call, respectively. To see this for the case of the put, suppose that $K > S$ on the expiration date. Then, if I am the holder of a put option, I can can buy the stock for S, and then exercise the option to sell it for K, in which case I will have realized a profit of $K - S$. On the other hand, if $K \leq S$, I would not want to exercise my option (remember, the option gives the holder the *right* to exercise it, not the obligation), since I would lose money if I bought the stock and then used the option to sell it for a higher price. This verifies the formula for the case of the put, and the call can be verified analogously. Apart from its *intrinsic* value, which is described above, an option is also said to have *time value*, which at any time before the expiration date comes into play. This reflects the fact that future movements in the underlying stock can favorably influence the options intrinsic value, so time value can be seen as the possibility of future gains (or losses) of the stock positively influencing the value of the call (put). 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.

2.1.3 Some Strategies involving Options

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. Another motive for buying an option might be speculative in nature, where for instance an investor is betting on a stock going up, say from \$60 to \$80, in the next three months. He might then go ahead and buy 100 stocks for \$6000, and realize a gain of \$20 per \$60 invested, or a total of \$200. However,

he might also buy a call option with a strike price of \$60 for a 100 shares, which is going to cost less than buying 100 shares, and then exercise these calls on expiration date, realizing a profit of \$20 on each call, which cost less than buying the actual stock. We see that the *yield*, the profit realized divided by the money invested, is higher when using the option than when buying the stock. The option is therefore said to have *leverage*. On the other hand, the speculator is also not exposing himself to losses on the stock, if his prediction of the stock would turn out to be false, since the worst he can do by buying the options is lose the money he paid for the option, since if the stock will go below \$60, he will simply not exercise the option. If he holds the stock itself, he can lose significantly, since the stock might go down, and with it his investment. These two examples serve to illustrate some of the reasons why options are being used by investors. A more comprehensive account of strategies involving options is given in [2].

2.1.4 Some practical facts and assumptions

Options are traded on different exchanges, such CBOE, AMEX, and PACIFEX. The data used for this paper is from trades made at the CBOE. Most options have expiration dates about 3, 6, and 9 months from when the option was written, and they mostly expire the Saturday after the third Friday of a given month. When the option is first written, the strike prices are cluttered around the current price of the underlying stock, mostly in increments of \$2.50, \$5, or \$10. While in the theoretical model there is only one price, in reality there is always a bid-ask spread in the price of the option, meaning that there is a difference in price if someone buys an option as opposed to selling it. The price at which an individual investor can buy an option is always slightly higher than the price at which one can sell an option, representing transaction costs of market makers, i.e. those who facilitate the trades. This will be familiar to anyone who has traveled abroad, and exchanged currency. For instance, I might exchange a dollar to 0.80 euros, and then if I exchange it back right away, I will receive less than my original dollar. This is the transaction cost that was incurred, and which the exchange booth will charge to make its income, similar to the income market makers make with bid-ask spreads on option prices. We will ignore these transaction costs, and work with the assumption that there is always just one price, regardless of if we buy or sell an option. The exact way we then determine the price of the option in the market is laid out in Section 5. We also do not take account of tax considerations, or dividends paid on stocks, where the latter is possible due to our sample selection of stocks which did not pay dividends in the time we are going to look at them. Much analysis is done this way in mathematical finance, and can be defended by drawing the analogy with classical mechanics in an idealized setting, where frictions are not present. When working scientifically, one always has to start out with a simple model, to become familiar with the way it works, and then see how additional restrictions influence the initial conclusions.

Another thing that any person interested in studying option prices empiri-

cally should watch out for when dealing with data is the issue of stock splits. Sometimes a company decides to split m of its current shares into n new shares, each of which will then have a value of $P\frac{m}{n}$, where P is the current stock price. Since this is just a change in the units and not in value, the value of the options one holds should also stay the same. For this reason, the option contract is then changed by changing the strike price to $\frac{m}{n}$ of its former value, and changing the volume of shares the contract is good for to $\frac{n}{m}$ of the former volume. This should then be accounted for in the data analysis, although it is probably easiest to select periods without stock splits if possible.

2.2 Pricing considerations

2.2.1 Rational 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. Although American options are going to be analyzed in the data, due to our selection process these are only options that do not pay dividends and hence work very similar to European options in the sense that it is not optimal to exercise them before expiration date.

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

$$C \leq S,$$

where C is the price of the call and S is the price of the stock. To see this, suppose C were greater than S. Then a 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

$$P \leq K,$$

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,

$$P \leq Ke^{-rT},$$

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 rational lower pricing bound for an 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 merely 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 $\max(S, \quad K)$. 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, $C + Ke^{-rT} \geq S$. Recalling the definition of the value of a put, we realize that it's always nonnegative, since it's the maximum of the difference between stock and strike price and zero. Putting this fact together with the last inequality yields

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

The final bound we want to derive is the lower bound for the European put. For this, consider two portfolios again, the first one containing a put and

a share, the second one cash worth Ke^{-rT}. On the day of expiration, the first portfolio will then be worth

$$\max(K - S, 0) + S = \max(K, S),$$

while the second one will be worth K. Therefore, the first one is always worth at least as much as the second one in the absence of arbitrage, so that

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

remembering that the option's value is always nonnegative.

2.2.2 Put-Call Parity

The final task of this section will be to derive the so-called 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. 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 an 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 $\max(S, \quad K)$. 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, which means that

$$C + Ke^{-rT} = P + S.$$

Note that in this formula we are stretching the notation a bit by making S be the stock price at the time at which we want to price the option, and not at expiration date as it is used for above.

3 Mathematics Background

3.1 Random Variables

The background necessary to read this paper is basic probability and second semester calculus. The reader unfamiliar with probability is referred to [1] for an excellent introduction to the field. Some more advanced concepts will be discussed, but for the most part are self-contained. Since the price of an option depends on the path the underlying stock takes, we will discuss ways of making

sense of the random nature of the stock price, and how to take each individual stock's characteristics into consideration when pricing the option on it. First of all, it will be useful to make a couple of definitions.

A *random variable* X is a function from a sample space Ω to the set of real numbers, $\mathbb{R}$ (or in more general cases, to $\mathbb{R}^n$) A particular value of X is called a *realization* of the random variable. A natural way to subdivide the set of all random variables is into discrete and continuous ones (there are actually also mixed random variables, which are discrete for certain values, and continuous for others, so our subdivision is not a partition). A *discrete* random variable is one in which can only take on a denumerable amount (either finite or countably infinite) of values, meaning that the cardinality of $X(\Omega)$ is at most countably infinite. *Continuous* random variables are those whose range is a non-empty interval of real numbers, which is of course uncountable infinite. We will mostly work with discrete random variables in the beginning.

Let P be a probability measure on Ω. A *probability measure* is a function on a sample space[1], such that the following conditions are satisfied:

(1) $\forall \omega \in \Omega, P(\omega) \geq 0$.

(2) $P(\Omega) = 1$

(3) For any countable, pairwise disjoint collection of sets $\{A_i\}_{i\in\Lambda} \subseteq \Omega$, $P(\bigcup_{i\in\Lambda} A_i) = \sum_{i\in\Lambda} P(A_i)$.

With this in mind, we can now define the probability distribution of a discrete random variable as

$$p_x = P(X = x) := P(X^{-1}(x)).$$

The probability that the random variable takes on value x is given by the probability of the event that is mapped to x by X. We will call $P(X = x)$ the *probability mass function* for X. To see why this is a quite natural definition, consider Ω to consist of all possible outcomes when a fair coin is flipped twice, i.e. $\Omega = \{HH, HT, TH, TT\}$ Then, we can define X to associate with each event in this space the number of heads in the event. Then, keeping in mind that each outcome is equally likely, we have that

$$P(X = 2) = P(X^{-1}(2)) = P(HH) = \frac{1}{4}.$$

[1]Strictly speaking, the function is defined over a collection of subsets of the sample space, which satisfies certain properties, a so-called σ-algebra. If one follows this definition, the first condition about P would have to change to $P(\{\omega\}) \geq 0$. We will relax this requirement, since we are mostly dealing with discrete random variables, in which case the σ-algebra actually is equal to the powerset of Ω, so that we can use events instead of the sets containing the events. We will therefore use slightly abusive notation by just writing $P(\omega)$ instead of $P(\{\omega\})$

3.2 Stochastic Processes

3.2.1 Definitions

A *stochastic process* $\{X_t\}_{t\in T}$ is a family of random variables, indexed by a parameter t, which runs over an index set T. One simple example for a stochastic process would be the action of throwing a die five times, and record in each individual trial the number of dots. For any particular sequence of realizations of the random variable, we use the term *sample path*. So, for instance, {1,4,2,3,6} and {2,6,3,4,2} are both sample paths of the stochastic process just described. Two common classifications of stochastic processes are based upon distinctions about the *state space*, which is the range of the random variables, and the *index set* T. We say that a process is a *discrete state* process or a *continuous state* process, if its state space is discrete or continuous, respectively. Similar distinctions are made with respect to the index set. A special case, which is often used in practice, is that of setting $T = [0, \infty]$, since then the stochastic process can be representing a continuously occurring random phenomenon, such as the diffusion of particles in an aqueous solution, or the price of a stock. We see that our example of throwing a die five times is a discrete-state, discrete-time process, since there are finitely many outcomes possible in each trial, and the number of trials is finite as well (they could actually both be denumerable, so long as they are not uncountably infinite, they are discrete).

There are some further properties of stochastic processes that appear in numerous applications, so that it is useful to look at them.

1) If for any $t_1 < t_2 < t_3$, $X_{t_2} - X_{t_1}, X_{t_3} - X_{t_2}$, are independent, then X_t is said to be a process with *independent increments*.
2) If for any t_i the distribution of $X_{t_i+h} - X_{t_i}$ depends only on h, then X_t is said to have *stationary increments*.
3) X_t is said to have the *Markov property* if given X_t and $s > t$, X_s is independent of X_u, for all $u < t$.

The first property basically says that the amount the random variable changed (or the sample path taken) over some time interval does not affect the probability of the change in any other ensuing, disjoint time interval. The second property is best explained with an example. If we assume for the moment being that the stochastic process describing the path of a stock satisfies this property, then we can expect to see the same change in stock price over a fixed time interval, say a day, no matter if we choose to look at the market today, in a week, in a month or in a year. The only thing that influences the distribution of change is the length of the time interval, regardless of which time we start looking at the price. The third property is the most important property in this context. It basically states that all that matters for the future path of the stochastic process is the current state, and we can discard all states before this moment in making statements about the future. In a sense, the Markov property expresses the notion of the process having no memory. One such example

is the drawing balls of different color from an urn with replacement. Every time we draw a ball, the probability of drawing a certain color stays the same, since we put the ball we drew before back into the urn. The urn hence does not "remember" what we did before, since the situation is always the same in every trial. In essence, this is one of the assumptions we will be making about the path of the stock price. At any given point of time, past developments should not really matter since only current information should have influence on the price. The standard academic finance argument for this is that if there were some past patterns in the market that people would use to predict the future, then if there were some real gain from this, many investors would start using this procedure, which would influence the price and ruin the entire prediction, making it useless. There is however controversy as to how much past patterns can really say about the future, and the entire field of technical analysis makes use of techniques that exploit patterns. The usefulness of this area is disputed, and not the topic of this paper. For the purpose of deriving the Black-Scholes model, this assumption will be used, since it is a reasonable starting point.

3.2.2 Markov Chains

We will now look at a specific discrete time, discrete state stochastic process known as a *Markov Chain* (there are also continuous time Markov Chains, but they have to be discrete-state to be called chain). We will take the index set T to be equal to the natural numbers. We will also call each value of the index set a *period*. The Markov Chain is characterized by - as the name suggests - the Markov property described above. A Markov Chain is said to be in *state* i at period t, if $X_t = i$. Since by the Markov property, the probability of changing to state j is only dependent on it being in state i right now, we can define $p_{ij}^t := P(X_{t+1} = j | X_t = i)$. We will further restrict our analysis by looking at *stationary* Markov chains, those whose probability does not depend on which period we are in. We are therefore dropping the superscript from the definition above, and now referring to p_{ij} as the transition probability from state i to state j.

To make these definitions clearer, consider a simple example from genetics. Suppose that we have a population of $2N$ genes, which are either type A, or type B genes. Now, at any given time period, there will be j genes of type A, and $1 - j$ genes of type B. The genes are now competing with each other, and the only way the evolutionary forces are at work in this model is by mere quantity. In particular, each gene in the next generation will be an A type with probability $p_j = \frac{j}{2N}$, and a B type with probability $q_j = 1 - p_j$. Since each gene changes according to a Bernoulli distribution with the given probability, and there will be 2N such trials in each period, we can see that the transition probabilities should be binomially distributed. In particular,

$$p_{jk} = f(k, 2N, p_j) = \binom{2N}{k} p_j^k (1 - p_j)^{2N-k}, \quad k \in \{1, 2, ..., 2N\}$$

where f is the probability mass function for the binomial distribution.

3.2.3 Binomial Price Process

We are now in the position to describe the model which will lay the groundwork for the derivation of the Black-Scholes equation. It was first proposed to simplify the derivation of the equation in 1979, by Cox, Ross, and Rubinstein (see [5]). This section is largely adopted from [4]. We assume that we are operating in an N-period economy, where in each period either a "good" or "bad" state materializes. Hence, we can model this by letting $\Omega' = \{0, 1\}$, and further require that $P(0) = q$, and $P(1) = 1 - q$, where $0 \leq q \leq 1$. Bad in this case is associated with 0, and good with 1. We let $\Omega = (\Omega')^N$ be the sample space describing the outcome in each period. Then we can define the (random) projection mapping

$$\pi_t : \Omega \to \{0, 1\}$$

by $\pi_t(\omega) = \omega_t$, where ω_t is the state of the economy in period t. The function then just picks the tth coordinate from the random vector $\omega = (\omega_1, \omega_2, ..., \omega_N)$. It is also of interest to us to measure the number of good and bad states that the economy was in at any given time t. To do this, we define

$$n_t(\omega) = \sum_{i=1}^{t} \pi_t(\omega)$$

The interpretation of this is that for any particular sample path ω, $n_t(\omega)$ will give us the number of periods up to time t in which the economy was good, and similarly $t - n_t(\omega)$ will give us the number of periods in which the economy turned out bad. Since for each i, n_i is a Bernoulli random variable (only two outcomes possible), we can see that n_t, as the sum of independent Bernoulli random variables, is binomially distributed. Therefore, similar to the genetic example before, the distribution function is given by

$$P(n_t = k) = \binom{t}{k} q^k (1 - q)^{t-k}, \quad k \in \{1, 2, ..., N\}$$

The next step is to relate the stock price to these states of the economy. Suppose that initially the stock price is S_0. Then, in period 1, the stock price is either uS_0, or dS_0, where $u > d > 0$ are the returns on the stock. The first case occurs when the economy is in a good state, and the second if the economy is in a bad state. We need to make d greater than zero to ensure that the stock is always strictly positive, since a zero or negative price would make no sense. We further assume that in any period s thereafter, we have the same u and d, so that $S_{t+1} = uS_t$, or $S_{t+1} = dS_t$, for each t, depending again on the state of the economy. Therefore, the price of the stock at time t given any particular $\omega \in \Omega$ is

$$S_t(\omega) = S_0 u^{n_t(\omega)} d^{t - n_t(\omega)} \tag{2}$$

If we divide by S_0, and take the log of both sides, we get

$$\ln\left(\frac{S_t}{S_0}\right) = n_t(\omega)\ln(u) + (t - n_t(\omega))\ln(d) = n_t(\omega)\ln\left(\frac{u}{d}\right) + t\ln(d). \quad (3)$$

We see that the logarithm of the ratio of the price at time t and time 0 is a linear function of a binomial random variable, and hence itself binomial. This will lead us to introduce in the next section the model of geometric Brownian Motion, for the stochastic process governing the price, which is obtained by taking the limit as the time interval between price changes approaches 0. Although the exact derivation of this will not be shown, anyone who has taken some statistics should be familiar with the binomial approximation to the Normal distribution. The idea here is similar, in that the process described above is binomial, and hence approximately normal, so that in the limit the logarithm of the ratio of successive prices is normally distributed. The term geometric comes from the fact that we are considering the logarithm of the ratio of successive prices, the so-called *log-return*, or log of percentage change over some period, instead of the prices themselves.

Now, one might ask why assuming that the price of a stock goes up or down with the same probability and by the same percentage amount in any given period is realistic. This is definitely not true if we let the period be a day, or even a week. However, we never said how long the period would really be. If we think of the period as consisting of a second, then this might be a pretty realistic approximation. If we want certain continuity assumptions, which lead to the Brownian Motion, then we want the stock to only undergo very small changes in very small time intervals, and not have a jump discontinuity. There are models for the case that the stock jumps from time to time, but we will not discuss them here, because the Black-Scholes model does not build on this assumption.

3.2.4 Brownian Motion

We now state the properties that the stochastic process governing the log-returns of the stock price is assumed to have in the Black-Scholes model, and which is the limiting case of the binomial process described before. The Brownian Motion $\{W_t\}$ is a continuous time, continuous-state, stochastic process such that the following conditions hold:

1) $\{W_t\}$ has the independent increments property
2) For each $W_{t_r}, W_{t_s}, r < s, W_{t_s} - W_{t_r} \sim N(0, s - r)$,
where $N(\mu, \sigma^2)$ is the Normal distribution with mean μ and variance σ^2.

4 The Black-Scholes Equation

4.1 Example of Hedging Portfolio

We will now present one way of deriving the Black-Scholes equation, via a binomial pricing process, which is based on the binomial price model described in the last section. This approach was first taken in [5], and represents the clearest way of deriving the equation. The presentation will closely follow the original paper. We assume that there is one risk-free interest rate at which we can borrow or lend money, and that there is one period left until the expiration date of a call option on the stock. Now, suppose we write 3 calls with strike price \$100 at C each, buy 2 shares of the stock at \$100, and borrow \$50 at an interest rate of 25%. Suppose further that the stock either doubles or halves in value at the end of the period. Then, if the stock goes up to \$200, the calls we wrote are going to be exercised, so that we lose 3(\$200 - \$100) = \$300. We also have to pay back the \$50 we borrowed, plus the interest of \$25 accumulated. Finally, since the stocks we had went up to \$200, we have \$400 in stocks. Adding all of these together yields \$25. Similarly, if the stock went down to \$50, we have \$100 in stocks, the calls will not be exercised, and we owe \$75 principal and interest of the money we borrowed. This again amounts to \$25. So, no matter what happens, we will end up with \$25 at the next period. Since we require there to be no risk-less arbitrage opportunities, the cost of setting up this portfolio must equal its profit, so that we require that $3C - 200 + 50 = 25$, and then $C = 58.33$.

We hence have determined the *fair value* of the option, by requiring that it be the value that equals its expected payoff. The interesting thing to note is that we have determined this merely by knowing the interest rate, the underlying stock price, the strike price, and the range of values that the stock can take on after one period. There was no mention anywhere of the probability of the stock going up or down, which is somewhat surprising. We only had to know the possible values the stock could take on after this period, which is what the volatility in a way will tell us later on in the Black-Scholes equation.

4.2 Binomial Option Pricing Model

We assume that the stock price changes from period to period according to the multiplicative binomial process in (2). We further assume that the risk-free return per period (the interest rate in one period plus 1), r_1, will be less than u, and greater than d. This reflects the risk premium on a stock price increase compared to a risk-free bond, during a good economy, and on the other hand the downward risk exposure that a stock brings with it in a bad state. We can now model the price the call should have, C, based on the two possible outcomes after one period, $C_u = \max(0, uS - K)$, which we suppose happens with probability q, and $C_d = \max(0, dS - K)$, which happens with probability $1 - q$. We also set up a portfolio with an amount of Δ stocks of price S in the first period, and B bonds. The value of this portfolio is then in the second period is

either $\Delta uS + r_1 B$ or $\Delta dS + r_1 B$, with probability q and $1-q$, respectively. Let us now choose Δ and B in such a way to equate the end of period payoffs from the call with the end of period values of the portfolio, i.e.

$$\Delta uS + r_1 B = C_u, \text{ and } \Delta dS + r_1 B = C_d$$

Solving for the two variables yields

$$\Delta = \frac{C_u - C_d}{(u-d)S}, \quad B = \frac{uC_d - dC_u}{(u-d)r_1}$$

A portfolio which is chosen in such a way is known as a *hedging portfolio*. C is not allowed to be less than $\Delta S + B$, since if that were the case, we would be able to make a risk-less profit by selling the call and buying the portfolio, since Δ and B were chosen in a way to replicate the returns of the call in the second period, so that we could gain the difference in price between the portfolio and the call without losing the entitlement to the same return as the call had. The current value of the call can also not be greater than the portfolio. In the case of the call being worth 0 this is immediately clear from the non-negativity of the portfolio. If, on the other hand, it were true that $\Delta S + B < S - K$, then if we set the price of the call between these two values, other investors would buy our call and exercise it immediately, thereby making arbitrage profits, which we assume do not exist. We see therefore, that the value of the call in the first period has to equal the value of the hedging portfolio, i.e.

$$C = \Delta S + B = \frac{[pC_u + (1-p)C_d]}{r_1} \tag{4}$$
$$p = \frac{r_1 - d}{u - d}, \quad 1 - p = \frac{u - r_1}{u - d}$$

Notice again how the probability of the stock going up or down does not appear in the equation for the value of the call. Another nice feature of introducing p is that it has the properties of probability, since it is always between 0 and 1. To see this, notice that $r_1 > d$, so that $r_1 - d > 0$. Similarly, $u - d > 0$, so p as the ratio of these two values is certain non-negative. To see that it is less than 1, we note that since $u > r_1$, it follows that $u - d > r_1 - d$, and hence since $r_1 - d$ is strictly greater than zero, $p = \frac{r_1 - d}{u - d} < 1$.

We now go over to a three period model, where we are interested again in pricing the call in period 0, and it expires in period 2. The stock can take on 3 possible values, $u^2 S$, duS, or $d^2 S$. the value of the call at expiration date is then again the maximum of zero and each of these values, depending on the state of the economy that materialized during period 1 and period 2. We can now derive recursively the value at which the option should be priced in the first period, by starting at the two possible outcomes in the first period, and asking ourselves what it should be there. Applying (4) to each of these cases, we arrive at

$$C_u = [pC_{uu} + (1-p)C_{ud}] \tag{5}$$
$$C_d = [pC_{du} + (1-p)C_{dd}]$$

Taking these values as given, we can apply (4) once more to figure out the value of C,

$$C = [p^2C_{uu} + 2p(1-p)C_{ud} + (1-p)^2C_{dd}]/r_1^2. \tag{6}$$

We recognize the inside of the parentheses to be a binomial expansion. Exploiting the pattern that emerges when we increase the number of periods, we are able to write down the general formula for the value of a call at period 0 that expires n periods from now as

$$C = [\sum_{k=0}^{n} \binom{n}{k} p^k(1-p)^{n-k} \max(0, u^k d^{n-k} S - K)]/r_1^n \tag{7}$$

This can be further simplified and brought into a form that resembles the Black-Scholes very much. Let $b = \min\{a \in \mathbb{N} | \quad u^a d^{n-a} S > K\}$. Then b is the smallest number of positive states of the economy necessary for the option to be in-the-money at expiration date. Then for every $k < b$, $\quad max(u^j d^{n-j} S - K, 0) = 0$, and for every $k \geq b$, $\quad max(u^j d^{n-j} S - K, 0) = u^j d^{n-j} S - K$, by the way b was defined. Since all the periods up to the bth are then zero in the sum in (7), it simplifies to

$$C = [\sum_{k=b}^{n} \binom{n}{k} p^k(1-p)^{n-k} (u^k d^{n-k} S - K)]/r_1^n \tag{8}$$

If $b > n$, then the option is clearly worthless, since then the minimum amounts of good states in the economy necessary to bring the stock up to a value that will make the option worth exercising is larger than the number of periods, n. Separating (8) some more we get

$$C = S[\sum_{k=b}^{n} \binom{n}{k} p^k(1-p)^{n-k} \left(\frac{u^k d^{n-k}}{r_1^n}\right)] - Kr_1^{-n}[\sum_{k=b}^{n} \binom{n}{k} p^k(1-p)^{n-k}] \tag{9}$$

Finally, by introducing a variable which again behaves like a probability, namely

$$p' := \frac{u}{r_1} p, \quad \text{and } 1 - p' := \frac{d}{r_1}(1-p),$$

we can write (8) as

$$C = S\Phi(b, n, p') - Kr_1^{-n}\Phi(b, n, p), \tag{10}$$

where Φ stands for the complementary binomial probability mass function. In the language of binomial random variables, the first term is described to be the

probability that in n periods there will be at least b successes , given n total periods, and "probability" p' of success in any given period (it is not actually a probability, but it behaves like one). This is the celebrated Binomial Option Pricing Formula, and it looks a lot like the Black-Scholes equation already. This in turn will now be obtained by a limiting procedure, where the number of periods goes to infinity. [2]

4.3 Convergence to Black-Scholes Equation

We have now arrived at the point where we take the binomial pricing formula, and show that it will have the Black-Scholes Equation as a limiting case. We will now verify - using the fact that a standardized binomial random variable approaches the Normal random variable in distribution as the number of trials approaches infinity - that the distribution of log-returns in the binomial model actually converges to the normal distribution in the limit, namely that as $n \to \infty$,

$$\Phi(b, n, p') \to N(d_1), \quad \Phi(b, n, p) \to N(d_2). \tag{11}$$

which will yield the Black-Scholes equation for a call option. The put option formula, which we stated in the beginning of the paper, can then be obtained via the put-call parity relationship in section 2.2.2.

We proceed by showing the convergence of the second term only, since the work necessary to show convergence of the first term is very similar. The complementary binomial probability distribution, $\Phi(b, n, p)$, gives the probability that given n total periods, the number of upward moves of the stock is greater than or equal to b, where p is the probability of the stock moving upward. Recalling that n_t is the number of upward moves of the stocks in n periods, we can write the following expression:

$$1 - \Phi(b, n, p) = Pr(n_t < b - 1) \tag{12}$$

$$= Pr\left(\frac{(n_t - np)}{\sqrt{np(1-p)}} < \frac{(b - 1 - np)}{\sqrt{np(1-p)}}\right) \tag{13}$$

If we let S_0 and S_t denote the the asset price now and n periods later, then we can take the expectation and variance of equation (3) to get

$$E[\ln\left(\frac{S_t}{S_0}\right)] = n\left(p \ln\left(\frac{u}{d}\right) + \ln(d)\right) \tag{14}$$

$$Var[\ln\left(\frac{S_t}{S_0}\right)] = np(1-p)\left(\ln\left(\frac{u}{d}\right)\right)^2 \tag{15}$$

[2] One should of course think of these periods to subdividing a finite time interval, namely the one between the time one wants to price the option at, and the expiration date of the option. The limiting process is hence one of infinite subdivision of a finite interval, not one of going indefinitely into the future, keeping the subintervals constant, which would make no sense in this context.

We now require that this mean and variance should in the limit, as n approaches infinity, agree, since they are both describing the same underlying process of the log-returns of stock prices. Hence,

$$\lim_{n\to\infty} n\left(p\ln\left(\frac{u}{d}\right)+\ln(d)\right) = (r-\frac{\sigma^2}{2})(ndt) \tag{16}$$

$$\lim_{n\to\infty} np(1-p)\left(\ln\left(\frac{u}{d}\right)\right)^2 = \sigma^2(ndt) \tag{17}$$

where dt stands for the length of the period, and ndt gives the time until expiration date of the option. In the original equation, (1), $T = ndt$ To see where these values came from, see the following derivation.

$$\begin{aligned} & d_2 \\ &= d_1 - \sigma\sqrt{ndt} \\ &= \frac{\ln(S/K)+(r+\sigma^2/2)ndt}{\sigma\sqrt{ndt}} - \sigma\sqrt{ndt} \\ &= \frac{\ln(S/K)+(r-\sigma^2/2)ndt}{\sigma\sqrt{ndt}} \end{aligned}$$

We hence see that since this is the standardized value of the log-return, the mean is $(r-\sigma^2/2)ndt$, and the variance is $\sigma^2(ndt)$, since it is the square of the standard deviation, which is found in the denominator of the above expression. By definition, a is the smallest non-negative integer such that $b \geq \frac{\ln\left(\frac{K}{Sd^n}\right)}{\ln\left(\frac{u}{d}\right)}$. As a consequence, there exist an ϵ in $(0,1]$ such that

$$k-1 = \frac{\ln\left(\frac{K}{Sd^n}\right)}{\ln\left(\frac{u}{d}\right)} - \epsilon.$$

Plugging this into equation (13) yields

$$Pr(n_t < b-1) = Pr\left(\frac{(n_t-np)}{\sqrt{np(1-p)}} < \frac{\ln\left(\frac{K}{S}\right) - n\left(p\ln\left(\frac{u}{d}\right)+\ln(d)\right) - \epsilon\ln\left(\frac{u}{d}\right)}{\sqrt{np(1-p)}\ln\left(\frac{u}{d}\right)}\right) \tag{18}$$

Now, taking the limit as n approaches infinity, and utilizing the fact that the binomial distribution converges to the Normal distribution in the limit, we get

$$\lim_{n\to\infty}(Pr(n_t < b-1)) = N\left(\frac{\ln\left(\frac{K}{S}\right) - \left(r-\frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}\right), \tag{19}$$

which is exactly the term in the Black-Scholes equation when taking the complement and using the symmetry property of Normal Distribution. The second convergence can be shown similarly, so this completes our derivation of Black-Scholes option pricing formula via the Binomial Pricing Formula.

5 Data Collection and Methodology

5.1 General Comments on Data Selection

I was using the Berkeley Options Database for the period from January 1989 to December 1993, which was made available to me through the Finance Department at the University of Arizona. It contains all option trades and quotes recorded at the Chicago Board Options Exchange between those dates, with data about volume of trade, price of option, price of underlying stock, strike price, expiration date, and a time stamp that reveals at which time, up to the second, the trade was made. For quotes, the bid and ask spread is revealed. The exact expiration date can be determined by looking at the the third Friday of a given expiration month. The program *Date.java* I have written, which can be found in the appendix, will determine this automatically via date functions. Since all of the options traded on American exchanges are American options, meaning that they can be exercised before expiration date, and the Black-Scholes model is actually for valuation of European Options, it was necessary to select stocks which did not pay any dividends between 1989 and 1993 (or at least over the life of a particular option during that period). If a stock pays no dividend, then the incentive to exercise early will vanish, due to the time value of the option, i.e. the possibility of favorable future developments. The arguments laid out before for the pricing of an European option should technically hold for non-dividend paying stocks. The first thing, however, that I had to determine to make the data useful, was which options were actually continuously traded on the exchange. Prof. Lamoureux provided me with a list of options that were traded on January 3rd, 1989, and then with another list of options that were traded on December 30, 1993. With the help of a small program which determined which ticker symbols were present at both dates (the number of symbols was around 200 and 600, respectively, making it too tedious to do this manually), I arrived at a list of about 150 options.

The next task was to find out which of these 150 options on different stocks paid no dividends. I started out typing in ticker symbols at Yahoo! Finance, and looking at historical prices, which has a feature to just display stock splits and dividends. After a fair amount of symbols had not even shown up on Yahoo, I began to wonder, researched a bit online, and found out that the symbols on the options do not necessarily match the symbols on the stock. So, I used the Options Clearing Corporation's web page to translate the option ticker symbol into the underlying stock symbol, which still did not work for some symbols. This was partly because some companies had gone bankrupt since then, and hence were neither listed in the Option Clearing Corporation, or Yahoo. Sometimes, mergers between companies made the historical data irretrievable, at least through Yahoo, and sometimes new companies had taken older companies' ticker symbols over, which led to some confusion. After going through all of the symbols, I found six stocks, which actually did not pay dividends. The companies I have selected are FedEx Corporation, Computer Sciences Corporation, Forest Laboratories Inc., Oracle Corp., National Semiconductor Corp.,

and Molex Corporation. Some of these companies did not have any information about dividends listed on Yahoo, so I looked up dividend information about these up in the Daily Investor's Guide in the library.

5.2 Data Filtering

I was now ready to extract the relevant data from this huge database. Since I wanted as much data as possible for each of these options, I chose to initially filter all the trades of the eight stocks selected for the entire four years, and then later write a program that would filter among this reduced data set with more refined criteria. Since this was a task involving probably more than trillions of lines of data, I opted against writing a program in Java, which would have run very slowly, and instead used the grep command in Unix to extract all the lines containing the option symbols of interest into separate files. A sample line of the database is shown below

1CSC890106143110 3 04500004000000504710

Here, the 1 indicates a Trade, as opposed to a quote, the CSC denotes the underlying stock (Computer Sciences Corporation), 890106 is the date, 143110 means it was traded at 2:31 pm and 10 seconds (the time is military time), the 3 denotes the expiration Month (March), the next empty space means that it is a call, whereas in the case of a put there would be a -, the 04500 denotes the strike price ($45), the 00400 denotes the price at which the option was traded ($4), 00005 is the quantity of options traded (5), and 04710 is the price of the underlying asset ($47.10). Notice that the price of the option is already standardized to be the per-unit price, so that we don't have to divide it by the number of options traded. We notice that the price of the option, $4, reflects the value of immediate exercise, $47.10 - $45 = $2.10, i.e. the intrinsic value, plus the time value of $1.90. Detailed information about working with the Berkeley Option database can be found in [7].

5.3 Methodology

There are now several ways we could proceed based on the data we have. Since in many cases there is more than one trade of a certain option on a given day, we decided to look at the last trade on any given day, as the option price on that day. In accordance with this, we also chose to consider the corresponding asset price to be the one we use to calculate the historical volatility. It is important to be consistent here, since we initially thought of taking the closing prices from Yahoo! Finance, but realized that the prices there sometimes differed from the prices at which the option trades were made. This could in extreme cases lead to arbitrage opportunities, which actually did not really exist, since there might be a time difference between the closing price of the stock traded on the NYSE or NASDAQ, and the last trade of the option on that stock made on the Chicago Board Options Exchange. I have therefore written a program,

FormatOptionData.java, which can be found in the appendix, to filter out the last trades on each day the option was traded, for an option of pre-specified strike price and expiration date, and convert it into a more readable format. To determine the risk-free rate, we will look up the (annualized) interest rate for a 3 month Treasury Bond. This is the best proxy for the risk-free rate, since there is virtually no risk of the government defaulting at any given time, so that the return is guaranteed.

6 Estimates of Historical Volatility

6.1 Definition and Practical Problems

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 is given by

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

where $u_i = \ln\left(\frac{S_i}{S_{i-1}}\right)$, $\bar{u}$ is the sample average of all u_i, S_i is the stock price in period i, and τ is the total length of each period in years ([2]). The numerator is readily seen to be the sample standard deviations of the u_i, the log-returns of the stock, and the denominator is a scaling factor to make the estimate be one of yearly volatility. Since the the prices were taken from the last daily trade of the corresponding option, we have price data for every day on which a trade of an option occurred. The problem with determining τ now 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 $\frac{1}{365}$. We start out with the second assumption, since this seems to be a more elegant formulation, 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 on weekends.

6.2 Tradeoffs in determining time scale

The standard error of this estimate,

$$SE_{\hat{\sigma}} \approx \frac{\hat{\sigma}}{\sqrt{2n}},$$

tells us that the precision of our estimate will generally go up as n increases. We might now be tempted to take into account as much past data as we can possibly gather. The tradeoff, however, is that volatility in reality changes over time, so that we do not want to take into account too much past data. Our

goals is therefore to determine how many past periods (days) of data we want to take into account to arrive an estimate as precise as possible for the volatility. Since we do not know the real volatility, how can we measure the precision of our estimate? Our metric for this will be the squared relative deviation,

$$M(\hat{P}) = \frac{(\hat{P} - P)^2}{P},$$

where $\hat{P}$ is the price the Black-Scholes equation yields when we plug in our estimated volatility, and P is the actual option price. We do this by fixing a day, probably in the middle of the option's life, and looking at the last trade of the day, thereby determining the option price, P. We then determine historical volatility for different time windows, by looking at prices from the past up to the date that we fixed, and determine which one gives us the best estimate. We would furthermore like to look at the consistency of the estimates, in the sense of them going into a certain direction if we increase the time interval, and if this trend also holds up when we compare the direction across the different stocks. We will see this in the results in a graph. To get some more information from the results, we also define the *residual* to be the difference between the predicted price and the actual price, the *squared residual* to be the square of this number, and the *relative residual* to be the residual divided by the actual price. The relative residual will serve as a measure of how much the predicted value of the option differed from the empirical value, in percentage terms. The relative squared residual will however be the variable which will be regressed on the time window. The time window will be changed in increments of 7 days, so the first historical volatility calculation will be based on closing prices of the stock 7 days before the pricing date of the option up to the pricing date, the second one 14 days before until the pricing date, and so on. To hold other variables constant, and allow some between-stocks comparisons, we will estimate volatility for the same pricing date, for each stock involved[3]. I found the price of put options on the stocks in the table below, for which the true option price, the expiration date, the stock price, the interest rate (which is of course the same across all stocks), and the strike price is shown.

[3]In theory, because of the stationary increments property of the Brownian Motion that is assumed to describe the stocks path, this should actually not matter. However, in reality this would possibly make a difference, since the stocks probably do not follow a Brownian Motion process exactly. We try to minimize nuisance in our study by not analyzing volatility of different stocks over different periods, since there might actually be some periods in the market where overall volatility is higher, or sudden changes occur. If this is the case, then we want this underlying noise to be present for all stocks, so that we can still make some comparisons among them, and attribute differences to the stocks, and not to the underlying conditions in the market

Summary of underlying Stocks of Options (All priced on 4/10/1992)

Stock	StockPrice	StrikePrice	ExpirationMonth	OptionPrice	InterestRate
CSC	$63.50	$65.00	9	$5.25	3.65%
FDX	$46.10	$45.00	7	$2.50	3.65%
FRX	$34.40	$35.00	8	$2.63	3.65%
MOQ	$32.40	$30.00	5	$.31	3.65%
NSM	$8.70	$10.00	9	$1.63	3.65%
ORQ	$13.20	$15.00	9	$2.75	3.65%

6.3 Results

I have included the capability to estimate historical volatility, according to the above formula, into the program *UserInterface.java* I am using. What I hoped to find is some kind of point of maximum efficiency for estimates of volatility, marking the point at which the cost of changing volatility outweighs the increase in precision gained by taking more time into account. This would require volatility to be at least "locally" constant, i.e. within some small time interval around a fixed date. In all this, one can however never forget, that by using the Black-Scholes as a measure of the precision of my estimates, I am actually testing the joint hypothesis that other assumptions of the Black-Scholes, such as no-arbitrage, and complete markets, are true, and that varying the time-scale makes a difference in estimating volatility. Unfortunately, there seems to be no better way of doing this, since most more advanced models do not have closed-form solutions, and therefore are not computationally easy to implement as proxies for the accuracy of volatility estimates. A summary of the results for historical volatility is shown in the table below.

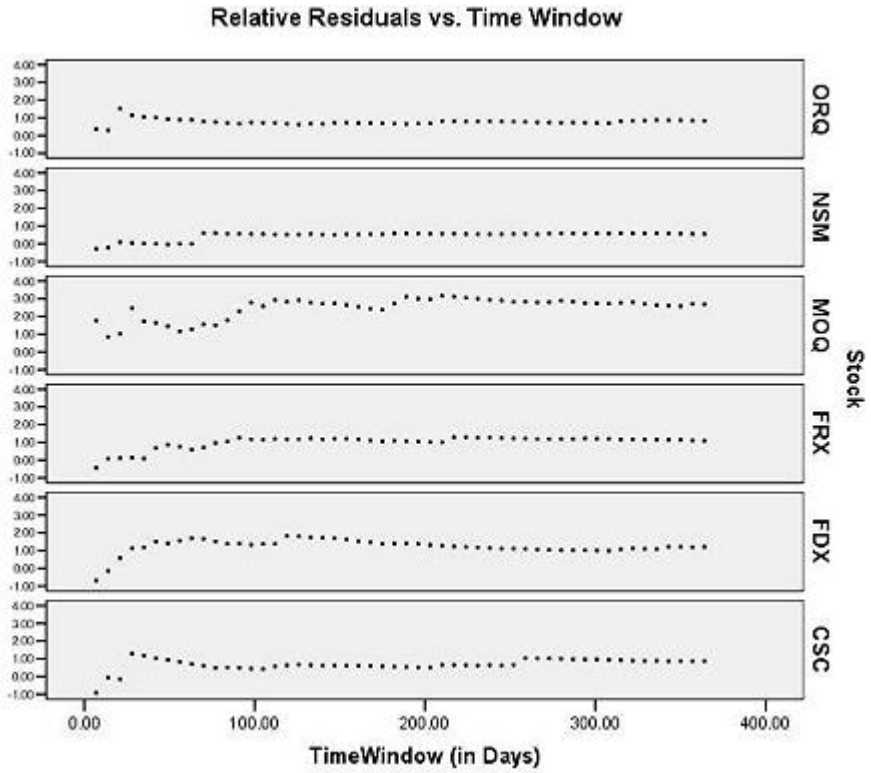
Descriptive Statistics for Historical Volatility Results (N =312)

	Minimum	Maximum	Mean	Std. Deviation
Volatility	.00	1.76	.7054	.24666
Residual	-4.79	6.80	2.1746	1.46669
RelativeResidual	-.91	3.16	1.1131	.76835
SquaredResidual	.00	46.30	6.8733	7.25930
SquaredRelativeResidual	.00	8.82	2.4482	1.76062

The volatility is not very meaningful in this case, since the summary here is aggregated across all stocks, and the stocks have different volatilities. Looking at the Residual, we see that using historical volatility, the highest an option is overvalued is by $6.80, and the lowest one is undervalued by is $4.79. On average, using historical volatility (disregarding the time window chosen), the

stock option is overvalued by $2.18. This persistent upward bias will show up later on in a plot of volatility versus time window chosen. The relative residual reveals an average overpricing by 111%, with a low of 91% underpricing, and including a case where an option was overvalued by 316%. The squared versions are mainly measures which ignore the direction of the pricing bias, and the relative squared deviation will be used in a regression analysis to determine how long of a time window one should use to minimize the pricing bias.

Instead of displaying a graph of how volatility depends on the length of the time window chosen, we instead opted to present the relative residual as a function of the time window, since for all the stocks we selected, since they are virtually at the money, the price the Black-Scholes equation predicts is an increasing, linear function of the volatility, so that the relative residual, being again a linear function of the predicted price, will reflect the calculated volatilities, since if these increase, so does the relative residual. It is also more meaningful to display these, since they represent the percentage by which the option was under or overpriced relative to the true price.



In each case, there is a persistent overpricing bias, which seems to stabilize itself after one takes a time window of more than a hundred days (remember, this means that historical volatility is calculated hundred days before 4/10/1992 up until that date, and then plugged into the Black-Scholes equation with the other values observed from the market). It also seems to be a common pattern, that taking into account two or three weeks (one interval denotes seven days) seems to be the best one can do with historical volatility. In some cases, such as MOQ and ORQ, big spikes in volatility are observable, which are most likely due to sudden changes in the stock price in the time window chosen. This shows the fundamental tradeoff in historical volatility, namely that in short time intervals the sampling error will be larger than in longer ones, so local outliers in log-returns will distort the volatility we arrive at. On the other hand, taking into account more data seems to produce an upward pricing bias.

To make our casual observations more precise, we will now regress the squared relative deviation on the time window chosen. One problem associated with this, which we have not taken into account, is that there is serial autocorrelation present in our data, since we are successively enlarging our time window, and thereby taking into account the same data, plus a new batch of seven days, in every sample point. In other words, the time windows are overlapping, since we always calculate historical volatility up to the 10th of April, 1992. We have calculated the Durbin-Watson statistic for this purpose, and concluded that there is positive autocorrelation present, since it is less than 1.5, meaning that successive values of calculated volatilities are correlated with each other. The linear regression we use hence loses its efficiency criterion, but it still gives us unbiased estimates of the parameters of the coefficient in the equation. We hence proceed with the regular regression shown below.

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.117[a]	.014	.011	1.75125	.238

a. Predictors: (Constant), TimeWindow

b. Dependent Variable: SquaredRelativeResidual

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients		
		B	Std. Error	Beta	t	Sig.
1	(Constant)	2.084	.201		10.357	.000
	TimeWindow	.002	.001	.117	2.082	.038

a. Dependent Variable: SquaredRelativeResidual

The effect that the length of the Time Window chosen has on the squared relative residual is significant at the 5% level. Since the coefficient is positive,

albeit only very slightly so, we conclude that increasing the time window will increase the pricing bias, so that a very short time window should be chosen to minimize the pricing bias (we want the squared relative residual to ideally be zero, in which case there is no bias). Putting these facts together with our observations earlier, we conclude that taking into account between 2 and 3 weeks, in the absence of unusual spikes, will lead to the best estimates of historical volatility, as we measure it.

7 Estimates of Implied Volatility

7.1 Calculation of implied volatility

The second major step of this project is to determine estimates of volatility based on implied volatility. Since implied volatility is a point estimate, we will determine implied volatility for several days before the day we price the option on, and then plug this value into the Black-Scholes equation with the other observed market values from the table earlier. This can get circular, since if we just take the value of implied volatility on the day we price want to price the option, we will recover the same price when plugging it back into the equation. This is why it makes sense to look at implied volatility the day before, two days before, a week before, and so on. Calculating implied volatility will require implementing some numerical procedures to interpolate the volatilities implied in a price of the option on a certain day, since the Black-Scholes equation is not explicitly solvable for volatility. We hence use the Newton-Raphson method to find the volatility on any given day, given the short term interest rate, the stock price, the option price, the strike price and the time to expiration on that date. For details about the algorithm, see [3]. The algorithm is coded into the program *UserInterface.java*, which can be seen in the appendix.

7.2 Results

All the residuals are defined the same way as in the historical volatility case. What we are changing now to find the optimal estimate is the day on which we elicit implied volatility from the closing trade of the option we want to price, with the same expiration date and strike price. We then go back and plug this value into the equation for pricing it on 4/10/1992. We define *DaysBeforePricing* - as the name suggests - to be the days before 4/10/1992, from which we take the implied volatility. Let us consider an example to make the procedure clearer. Let us say we take CSC to be the stock, with strike price \$65 and expiration month September. We now look at implied volatility on April 2nd, 1992. Let us suppose that the stock price is \$60, the short term interest rate is 3%, the option price is \$5.00 on that day. Then we use these values, and see - with the help of the Newton-Raphson algorithm - which value of volatility would lead to the observed option price. This is our implied volatility, which we then again plug back into the Black-Scholes equation, but this time with the observed

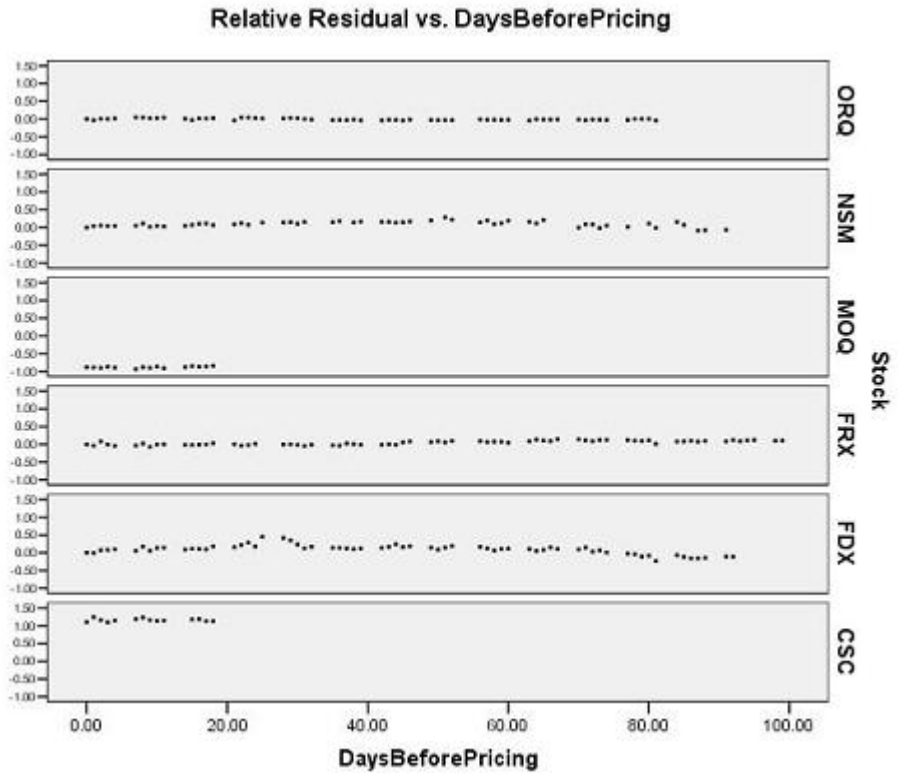
market values on April 10th, which are shown in the table of stocks earlier in the historical volatility section. This is then our predicted price, which we compare to the real price via our residuals. Let us now look at the summary statistics for implied volatility, and the associated residuals of our predicted prices and real prices.

Descriptive Statistics for Implied Volatility Results (N=280)

	Minimum	Maximum	Mean	Std. Deviation
Volatility	.220254	.647817	.40368793	.091919065
Residual	-2.486199	3.100420	.13384132	.867188535
RelativeResidual	-.94	1.24	.0628	.34015
SquaredResidual	.000000	9.612604	.76724368	2.141335445
SquaredRelativeResidual	.000000	3.846042	.30181645	.841730742

Although these results are aggregated over all stocks and all days, the volatility is moving in a well-defined band between about 20 and 60 per cent, with a mean of 40. This is in contrast with the more wildly varying values we calculated for historical volatility, ranging from 0 to 170, with a mean of 70. The standard deviation is also significantly lower than in the case of historical volatility, suggesting a more precise estimate. Looking at the residuals, we notice that on average the option is only overpriced by \$0.13, compared to \$2.17 in the historical volatility case. The outliers are nearer towards the mean as well. A similar picture emerges when looking at the relative deviations, where on average the option price was only overpriced by about 6.3%, whereas in the historical volatility case the overpricing was an average 111%. The largest overestimate of the option price is 114%, barely above the average by which historical overestimated. Looking at these results, we see that implied volatility seems to be clearly superior to historical volatility in terms of approximating true volatility, at least insofar as it replicates the actual price in the market much nearer when using the Black-Scholes model.

Before making further comments on this, let us look at a graph of how the relative residual changes as we go back from the day we price the option on to elicit implied volatility. Notice that for MOQ and CSC, there are not many days available, since there were not any trades of the options with the necessary expiration date and strike price more than 20 days before April 10th. For the most part, it is quite striking how stable the estimates are over time, and how near to 0 they are. The only anomalies are MOQ, which seems persistently underpriced using implied volatility, and CSC, which is constantly overpriced, with no trend downwards noticeable. For MOQ, the reverse trend is observable, where persistent overpricing is the case. Unlike in the historical volatility case, there is no clear trend distinguishable here, so that we have to turn to regression to determine if going back further leads to better estimates or not.



Serial Autocorrelation, as shown present by the Durbin-Watson statistic below - again limits the efficiency of our tests, and the strength of our conclusions, but we nonetheless proceed to the regression results. Notice, however, that this correlation over time is actually a reason to doubt the independence assumption made on the process governing stock price, since if the past is strongly correlated with the presence, then independence is clearly violated. There is much debate about whether this assumption can hold or not, and we merely mention the issue here, since it seems to show up in the data. On the other hand, the autocorrelation is not surprising, given the stability of our residuals over time, since if the values were staying the same all the time, then we would have perfect serial autocorrelation.

According to these results, which are significant at the 1% level, the pricing bias goes down if we go back in time more. In fact, if this model were the true model, we would have to calculate implied volatility 69 days before the day we price, in order to make the predicted price match the true price. However, the effect of the two outlier stocks, MOQ and CSC, might be weighing in a lot,

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.373[a]	.139	.136	.782226896	.142

a. Predictors: (Constant), DaysBeforePricing

b. Dependent Variable: SquaredRelativeResidual

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.757	.082		9.191	.000
	DaysBeforePricing	-.011	.002	-.373	-6.713	.000

a. Dependent Variable: SquaredRelativeResidual

which is why we run the regression again below, censoring these two stocks, to see if clearer results are available.

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.025	.008		3.954	.000
	DaysBeforePricing	2.89E-005	.000	.015	.239	.811

a. Dependent Variable: SquaredRelativeResidual

This change actually makes our prior inference highly insignificant, since now we have to fail to reject the hypothesis that there is no effect of how many days before the date of pricing we take implied volatility from on the relative residual. Although this is a rather unsatisfactory answer, if we base our conclusions on the graph earlier, we might as well conclude that it is best to choose values from the last few days, since they yield precise and consistent estimates.

8 Conclusions

We finish this paper by a discussion of our results. When using historical volatility to estimate true volatility, there is a significant bias towards overpricing the option price when using the Black-Scholes equation, with an average of over 100% deviation from the real price. When changing the time window we take into account, in the absence of large spikes, the first two or three weeks before the date we are pricing the option on seem to be the optimal choice. In contrast to this, implied volatility yields astonishingly precise and consistent

estimates for volatility, with the exception of two significant outliers. The average overpricing is a mere 6% of the original option price. However, there is no clear trend as to when one should look at implied volatility, since it is nearly constant across weeks, sometimes even months. For this reason, it seems best to stay within a week of the date we are interested in pricing the option on, since from visual inspection this seems to be a stable period. However, our conclusions about the precision of implied volatility is seriously weakened by the two outliers, MOQ and CSC, since they persistently were over and underpriced, respectively. Surveying the empirical literature on implied volatility and looking at larger samples of stocks might clear up these issues more and yield stronger conclusions. Comparing both estimators directly also turns out to be problematic, since if we wanted to compare the means of relative residuals, for instance, we would violate the independence condition of the two samples. There might furthermore be problems involved with comparing implied and historical volatility directly, since they are calculated very differently. Lastly, the issue of serial autocorrelation in both samples would have to be dealt with more formally, since more advanced regression models are available to arrive at more valid conclusions.

Acknowledgements

I thank Jialing Dai for advising me on this project, and spending countless hours discussing the mathematical, and statistical concepts involved in this project with me. I also thank Robert Indik for his advice on how to structure the report. I am very grateful to Duncan Buell, the donor of the Lusk Scholarship, which I received to work on this project, and without which this would have not been possible. I also am indebted to Chris Lamoureux, the Department Head of the Finance Department at the University of Arizona, for making the Berkeley Option Database accessible to me, and Andrew Zhang, the "data czar" of the Finance Department, for helping me locate and gather the necessary data.

References

[1] Chung, Kai Lai. Elementary Probability Theory with Stochastic Processes. 3rd edition. Springer (1974).

[2] Hull, John. Options, Futures, and Other Derivatives. 5th edition. Prentice Hall (2002): 151-252.

[3] Kwok, Yue-Kuen. Mathematical Models of Financial Derivatives. Springer Finance (1998): 62-64.

[4] Medina, Pablo Koch; Sandro, Merino. Mathematical Finance and Probability. Birkhauser Verlag (2003): 201-216.

[5] Rubinstein, Mark, Ross, Stephen A, and Cox, John C. Option Pricing: A Simplified Approach. http://www.in-the-money.com/artandpap/Option Pricing - A Simplified Approach.doc.

[6] Scholes, Myron, Black Fischer. The Pricing of Options and Corporate Liabilities. The Journal of Political Economy Vol. 81 (May - June 1973): 637 - 654.

[7] The Berkeley Options Database User's guide. The Berkeley Options Database. Institute for Business and Economic Research #1922. University of California, Berkeley. http://www.in-the-money.com/pages/bodbguide.htm.

Appendix

UserInterface.java

```
/*
 * Author: Peter Gross
 * This program is a user interface for calculating implied
 * and historical volatility
 */

import java.io.File; import java.io.FileNotFoundException; import
java.io.FileWriter; import java.io.IOException; import
java.io.PrintWriter; import java.util.ArrayList; import
java.util.Scanner;

public class UserInterface {

    public static void main(String[] args) throws FileNotFoundException {
        startInterface();
    }

    // This function starts the interface
    public static void startInterface() throws FileNotFoundException {

        Scanner in = new Scanner(System.in);
        String input;

        System.out.println();
        System.out.println("Please select Option");
        System.out.println("1> Calculate Historical Volatility");
        System.out.println("2> Calculate Implied Volatility");
        System.out.println("3> Exit");
        System.out.print(":> ");

        input = in.next();

        if (input.equals("3"))
            return;
        else {
            if (input.equals("1")) {
                historicalVolatility();
                startInterface();
            } else if (input.equals("2")) {
                impliedVolatility();
                startInterface();
            } else if (input.compareTo("3") > 0 || input.compareTo("0") < 0) {
                System.out.println("Invalid Entry");
```

```
            startInterface();
        }
    }
}

// This function handles implied volatility calculations
public static void impliedVolatility() throws FileNotFoundException {

    Scanner key = new Scanner(System.in);

    System.out.print("Please enter symbol of stock: ");
    String stock = key.next();

    String fileName = stock.toUpperCase() + "closingprices";

    Scanner in = new Scanner(System.in);

    ArrayList<Date> dates = new ArrayList<Date>();
    ArrayList<Double> rates = new ArrayList<Double>();

    // First input interest rates from file called Rates3Months
    // Format needs to be DD/MM/YYYY/Rate in each line
    Scanner file = new Scanner(new File("Rates3Months"));

    while (file.hasNext()) {

        Scanner temp = new Scanner(file.nextLine());
        temp.useDelimiter("/");

        int month = temp.nextInt();
        int day = temp.nextInt();
        int year = temp.nextInt() % 100;

        rates.add(temp.nextDouble() / 100.0);

        dates.add(new Date(year, month, day));

    }

    // Now input market data from file
    ArrayList<entry> entries = new ArrayList<entry>(UserInterface
            .inputMarketData(fileName));

    ArrayList<Double> impliedVolatilities = new ArrayList<Double>();
    ArrayList<Double> impliedVolatilityPrices = new ArrayList<Double>();

    double rate;

    // Enter closing stock price, short term interest rate and date on day
    // to be priced on here
    double assetPriceNow = 16.2;
    double rateNow = .0365;
    Date dateToBePriced = new Date(93, 4, 12);

    // Number of iterations for Newton-Raphson Algorithm
    int steps = 100;

    int[] dates2 = new int[entries.size()];

    // Find implied volatilities for market data
    for (int j = 0; j < entries.size(); j++) {

        entry temp = entries.get(j);

        // Find interest rate associated with given date
        int k = 0;
        while (dates.get(k).returnDate() != temp.getDate()) {
            k++;
```

```
            }

            rate = rates.get(k);

            // Now find time left till expiration date
            double timeLeft = Date.daysBetween(new Date(temp.year, temp.month,
                    temp.day), Date.findExpDate(temp.year, temp.expMonth)) * 1.0 / 365;
            double timeLeftNow = Date.daysBetween(dateToBePriced, Date
                    .findExpDate(temp.year, temp.expMonth)) * 1.0 / 365;

            // Find Implied Volatilities
            impliedVolatilities.add(findImpliedVolatility(temp.assetPrice,
                    temp.optionPrice, rate, temp.strikePrice, timeLeft, steps));
            dates2[j] = temp.getDate();

            // Find predicted prices, given implied volatilities
            impliedVolatilityPrices.add(UserInterface.BlackScholesPrice(
                    assetPriceNow, temp.strikePrice, rateNow, timeLeftNow,
                    impliedVolatilities.get(j), true));
        }

        // Prepare results for output
        ArrayList<String> lines = new ArrayList<String>();

        lines.add("Stock\tImplied Volatility\tDate\tPredicted Price");

        for (int l = 0; l < impliedVolatilities.size(); l++) {
            lines.add(stock + "\t" + impliedVolatilities.get(l) + "\t"
                    + dates2[l] + "\t" + impliedVolatilityPrices.get(l));
        }

        UserInterface.output(stock + "impliedVolatility", lines.size(), lines);

    }

    // This function contains an interface to determine historical
    // volatility
    private static void historicalVolatility() throws FileNotFoundException {

        Scanner in = new Scanner(System.in);

        Date datePriced;
        Date expDate;

        int expMonth;
        int interval;
        int times;
        double timeLeft;
        double rate = .0365; // Need to specify interest rate on date to be
                             // priced
        double strike;
        double assetPrice = 46.1; // Also need to specify asset price on date
                                  // to be priced
        boolean put = true;

        String fileName;
        String stockSymbol;
        String outputFileName;
        String optionType;

        int temp[] = new int[3];
        double volatilities[];
        Date[] startDate;
        double prices[];
        ArrayList<String> lines = new ArrayList<String>();

        System.out.println();
        System.out.println("Please specify parameters");
```

```
System.out.print("Stock symbol: ");
stockSymbol = in.nextLine();

System.out.print("Expiration Month: ");
expMonth = in.nextInt();

System.out.print("Strike Price: ");
strike = in.nextDouble();

System.out.println("Date to be priced : ");
System.out.print("yy: ");
temp[0] = in.nextInt();

System.out.print("mm: ");
temp[1] = in.nextInt();

System.out.print("dd: ");
temp[2] = in.nextInt();

datePriced = new Date(temp[0], temp[1], temp[2]);

System.out.print("Put ? <y/n>: ");
if (in.next().equals("n"))
    put = false;

System.out
        .print("Select interval in days to calculate historical volatility: ");
interval = in.nextInt();

System.out.print("Select number of times to go back: ");
times = in.nextInt();

// Find expiration date
expDate = Date.findExpDate(temp[0], expMonth);

// Find number of days left to expiration
timeLeft = Date.daysBetween(datePriced, expDate);

// Convert to years
timeLeft /= 365;

// Name of Input File
fileName = stockSymbol;

if (put)
    optionType = "PUT";
else
    optionType = "CALL";

// Name of Output File
outputFileName = "HISTVOL" + stockSymbol + optionType + "EXPIRES"
        + expDate.year + expDate.month + "STRIKEPRICE" + strike;

// Find the volatilities
volatilities = new double[times];

startDate = new Date[times];
prices = new double[times];

for (int i = 0; i < times; i++) {
    startDate[i] = Date.findDateNDaysLess((i + 1) * interval,
            datePriced);

    volatilities[i] = findHistoricalVolatility(startDate[i].year,
            startDate[i].month, startDate[i].day, datePriced.year,
            datePriced.month, datePriced.day, fileName);
```

```
            prices[i] = BlackScholesPrice(assetPrice, strike, rate, timeLeft,
                    volatilities[i], put);
        }

        lines.add("Volatility               Days    Price ");
        for (int j = 0; j < times; j++)
            lines.add("" + volatilities[j] + "\t "
                    + Date.daysBetween(startDate[j], datePriced) + "\t "
                    + prices[j]);

        output(outputFileName, lines.size(), lines);

    }

    // Returns historical volatility of asset price based on start and end
    // dates from file
    public static double findHistoricalVolatility(int startYear,
            int startMonth, int startDay, int endYear, int endMonth,
            int endDay, String fileName) throws FileNotFoundException {

        // File input variable
        Scanner file = new Scanner(new File(fileName));

        // Variable to store entries that are relevant
        ArrayList<entry> entries = new ArrayList<entry>();

        // Temporary variables
        entry temp1, temp2;
        String s1, s2;

        // count keeps track of how many closing prices are generated
        int count = 0;

        // Loop through file
        while (file.hasNext()) {
            temp1 = new entry(file.nextLine());
            temp2 = new entry(file.nextLine());

            s1 = temp1.date();
            s2 = temp2.date();

            // Makes sure only closing prices are used and prices are in the
            // range of interest
            if (!s1.equals(s2)
                    && temp1.getDate() >= (startYear * 10000 + startMonth * 100 + startDay)
                    && temp1.getDate() <= (endYear * 10000 + endMonth * 100 + endDay)) {
                entries.add(temp1);
                count++;
            }

        }

        // Variables to calculate volatility
        double[] logReturn = new double[count];
        double sum = 0;
        double sumOfSquares = 0;
        double average = 0;
        double volatility = 0;

        // Convert asset prices into log returns
        for (int i = 0; i < count - 1; i++)
            logReturn[i] = Math.log(entries.get(i + 1).assetPrice
                    / entries.get(i).assetPrice);

        // Sum log returns
        for (int i = 0; i < count; i++)
            sum += logReturn[i];
```

```
    // average of log returns
    average = sum / (count);

    // Find sum of squares of log returns
    for (int i = 0; i < count; i++)
        sumOfSquares += Math.pow(logReturn[i] - average, 2);

    // Find yearly volatility estimate
    volatility = Math.sqrt((1.0 / (count - 1)) * sumOfSquares)
            / Math.sqrt(1.0 / 365);

    return volatility;

}

// This function returns the Black Scholes price of an option
public static double BlackScholesPrice(double assetPrice,
        double strikePrice, double interestRate, double timeToExpiration,
        double volatility, boolean put) {

    double price = 0;

    double d1 = (Math.log((double) assetPrice / strikePrice) + (interestRate +
                (volatility * volatility) / 2.0)
            * timeToExpiration)
            / (volatility * Math.sqrt(timeToExpiration));

    double d2 = (d1 - volatility * Math.sqrt(timeToExpiration));

    if (put)

        price = strikePrice * Math.exp(-interestRate * timeToExpiration)
                * NormalDist(-d2) - assetPrice * NormalDist(-d1);

    return price;

}

// Polynomial approximation to Normal Distribution
public static double NormalDist(double d) {

    double approximation = 0;
    double k = 1 / (1 + 0.2316419 * d);
    double a1 = 0.319381530;
    double a2 = -0.356563782;
    double a3 = 1.781477937;
    double a4 = -1.821255978;
    double a5 = 1.330274429;

    if (d >= 0)
        approximation = 1
                - ((1.0 / Math.sqrt(2 * Math.PI)) * Math
                        .exp((-(d * d)) / 2.0))
                * (a1 * k + a2 * k * k + a3 * k * k * k + a4 * k * k * k
                        * k + a5 * k * k * k * k * k);

    else {
        k = 1 / (1 + 0.2316419 * -d);
        approximation = 1 - (1 - ((1.0 / Math.sqrt(2 * Math.PI)) * Math
                .exp((-(d * d)) / 2.0))
                * (a1 * k + a2 * k * k + a3 * k * k * k + a4 * k * k * k
                        * k + a5 * k * k * k * k * k));
    }

    return approximation;

}
```

```
// This function writes an ArrayList<String> into a file named
// outputFileName, for the first count
// elements of the ArrayList
public static void output(String outputFileName, int count,
        ArrayList<String> data) {

    {
        FileWriter writer = null;
        try {
            writer = new FileWriter(outputFileName);
        } catch (IOException ioe) {
            System.out.println("Could not create the new file: " + ioe);
        }

        PrintWriter diskFile = new PrintWriter(writer);

        for (int i = 0; i < count; i++) {
            diskFile.println(data.get(i));
        }
        diskFile.close();
    }
}

// This function reads a tab-delimited input file containing market
// data for implied volatilities
// The format of the file has to be as follows:
// STOCK YY MM DD TIME EXP TYPE STRIKE TRADE ASSET
// each line in the file represents an entry
public static ArrayList<entry> inputMarketData(String fileName)
        throws FileNotFoundException {

    // This ArrayList will save the entries
    ArrayList entries = new ArrayList<entry>();

    // Open the file
    Scanner file = new Scanner(new File(fileName));

    // Count variable to keep track of ArrayList
    int i = 0;

    // Skip first (title) line
    file.nextLine();

    // Fill ArrayList with entries from file
    while (file.hasNext()) {
        entries.add(i, new entry(file.nextLine(), true));
        i++;
    }

    return entries;

}

// This method estimates implied volatility using the Newton-Raphson method
// with steps iterations
public static double findImpliedVolatility(double assetPrice,
        double optionPrice, double interestRate, double strikePrice,
        double timeToExpiration, int steps) {

    // Initial estimate for volatility
    double sigmaInitial = Math.sqrt(Math.abs(2
            / timeToExpiration
            * (Math.log(strikePrice / assetPrice) + interestRate
                    * timeToExpiration)));

    // SigmaN is nth estimate
    double sigmaN = sigmaInitial;
```

```
        for (int i = 0; i < steps; i++) {

            sigmaN = sigmaN
                    - (BlackScholesPrice(assetPrice, strikePrice, interestRate,
                            timeToExpiration, sigmaN, true) - optionPrice)
                    / Vega(assetPrice, optionPrice, interestRate, strikePrice,
                            sigmaN, timeToExpiration);

        }
        return sigmaN;

    }

    // Partial Derivative of Black-Scholes with respect to volatility
    public static double Vega(double assetPrice, double optionPrice,
            double interestRate, double strikePrice, double volatility,
            double timeToExpiration) {

        double d1 = (Math.log((double) assetPrice / strikePrice) +
                (interestRate + (volatility * volatility) / 2.0)
                * timeToExpiration)
                / (volatility * Math.sqrt(timeToExpiration));

        double vega = (strikePrice * Math.sqrt(timeToExpiration) * Math.exp(-d1
                * d1 / 2))
                / Math.sqrt(2 * Math.PI);

        return vega;
    }

}
```

Date.java

```
/*
 * Author: Peter Gross
 * This class handels date arithmetic, which is necessary for the
 * historical volatility calculations in UserInterface. Note that it is only
 * meant to handle dates between January 1, 1989 and December 31, 1993, since that
 * is the range of dates that my part of the database covers.
 */

public class Date {

    static final int[] years = { 89, 90, 91, 92, 93 };

    static final boolean[] leap = { false, false, false, true, false };

    static final int[] daysInMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
            30, 31 };

    int year;

    int month;

    int day;

    boolean isLeap;

    public Date(int initYear, int initMonth, int initDay) {

        year = initYear;
        month = initMonth;
        day = initDay;
        isLeap = leap[year - 89];

    }
```

```
// Increment one day
public void incrementDays() {

    if (month == 2) {
        if (isLeap) {
            if (day < 29)
                day++;
            else {
                day = 1;
                month++;
            }
        }

        else {
            if (day < 28)
                day++;
            else {
                day = 1;
                month++;
            }
        }

    } else if (month != 12 && month != 2) {
        if (day < daysInMonth[month - 1])
            day++;
        else {
            month++;
            day = 1;
        }
    }

    else {
        if (day < daysInMonth[11])
            day++;
        else {
            month = 1;
            year++;
            isLeap = leap[year - 89];
            day = 1;
        }
    }
}

// Increment n days
public void incrementDays(int n) {

    for (int i = 0; i < n; i++)
        incrementDays();

}

// Find the day that is n days before Date date
public static Date findDateNDaysLess(int n, Date date) {

    Date firstDate = new Date(89, 1, 1);

    while (daysBetween(firstDate, date) != n)
        firstDate.incrementDays();

    return firstDate;

}

// Return date as integer (YYMMDD)
public int returnDate() {

    return year * 10000 + month * 100 + day;
}
```

```
    // Find the days between start and end
    public static int daysBetween(Date start, Date end) {

        int daysBetween = 0;
        Date temp = new Date(start.year, start.month, start.day);

        while (temp.day != end.day || temp.month != end.month
                || temp.year != end.year) {
            daysBetween++;
            temp.incrementDays();
        }

        return daysBetween;

    }

    // This function returns the third Friday of a given month and given year
    // This is important since options expire the third Friday of a given month
    public static Date findExpDate(int expYear, int expMonth) {

        // This is the first Friday in the data set
        Date firstFriday = new Date(89, 1, 6);

        int daysBetween = 0;
        Date currentDate = firstFriday;

        // Increment through until month and year are reached
        while (currentDate.month != expMonth || currentDate.year != expYear) {
            currentDate.incrementDays();
            daysBetween++;
        }

        // Return third Friday of given Month as Date object
        return new Date(expYear, expMonth, 14 + (8 - daysBetween % 7));
    }
}
```

FormatOptionData.java

```
/*
 * Author: Peter Gross
 * This program converts data from the Berkeley Option Database into a
 * more readable format for further inspection.
 * For a given year, expiration month, strike price, option type, and
 * ticker symbol it will write a file with all trades of the particular option
 * from the beginning of the year through the expiration month. Files from
 * Berkeley Option Database need to be in same folder, and named after their
 * ticker symbol
 */

import java.io.File; import java.io.FileNotFoundException; import
java.io.FileWriter; import java.io.IOException; import
java.io.PrintWriter; import java.util.ArrayList; import
java.util.Scanner;

public class FormatOptionData {

    public static void main(String[] args) throws FileNotFoundException {

        ArrayList<entry> lines = new ArrayList<entry>();

        int year = 93; // Year
        boolean put = true; // Set to false if option is call
        int expiration = 8; // Expiration month
        double strike = 15; // Strike price
        String optionType;
```

```
        // Set option type
        if (put)
            optionType = "PUT";
        else
            optionType = "CALL";

        String fileName = "ITQ"; // Type in ticker symbol here

        // Create output file
        String outputFileName = fileName.substring(0, 3) + optionType
                + "EXPIRES" + year + expiration + "STRIKEPRICE" + strike;

        // Read File

        Scanner file = new Scanner(new File(fileName));

        lines.ensureCapacity(10000000);

        long count = 0;
        entry temp;

        while (file.hasNext()) {
            temp = new entry(file.nextLine());

            if (temp.expMonth == expiration && temp.year == year
                    && temp.isPut == put && temp.strikePrice == strike
                    && temp.month < expiration + 1 && temp.recordType == 2) {
                lines.add(temp);
                count++;
            }

            // Write file

            output(outputFileName, (int) count, lines);

        }

    }

    // This function writes an ArrayList<entry> into a file named
    // outputFileName, for the first count
    // elements of the ArrayList
    public static void output(String outputFileName, int count,
            ArrayList<entry> data) {

        FileWriter writer = null;
        try {
            writer = new FileWriter(outputFileName);
        } catch (IOException ioe) {
            System.out.println("Could not create the new file: " + ioe);
        }

        PrintWriter diskFile = new PrintWriter(writer);

        for (int i = 0; i < count; i++) {
            diskFile.println(data.get(i).outputAll());
        }
        diskFile.close();
    }
}
```

entry.java

```
/*
 * Author: Peter Gross
 * This class stores information from the Berkeley Option Database and
 * allows manipulation of this information.
 */
```

```
import java.util.Scanner;

public class entry {

    public int year, month, day, time; // Time and date of transaction

    public int recordType; // Number indicating type of transaction (Trade,
                           // Quote, ...)

    public String symbol; // Ticker symbol

    public int expMonth; // Expiration Month of Option

    public boolean isPut = false; // Put

    public boolean isCall = false; // Call

    public double strikePrice; // Strike Price of Option (in dollars)

    public double optionPrice; // Price traded at (in dollars)

    public double assetPrice; // Price of underlying stock (in dollars)

    // This constructor converts a line in the database to an entry
    // for the purpose of finding historical volatility
    public entry(String input) {

        if (input.charAt(0) == ' ')
            recordType = Integer.parseInt(input.substring(1, 2));
        else
            recordType = Integer.parseInt(input.substring(0, 2));

        symbol = input.substring(2, 5);
        year = Integer.parseInt(input.substring(5, 7));
        month = Integer.parseInt(input.substring(7, 9));
        day = Integer.parseInt(input.substring(9, 11));
        time = Integer.parseInt(input.substring(11, 17));

        if (input.charAt(17) == ' ')
            expMonth = Integer.parseInt(input.substring(18, 19));
        else
            expMonth = Integer.parseInt(input.substring(17, 19));

        if (input.charAt(19) == '-')
            isPut = true;
        else
            isCall = true;

        strikePrice = Double.parseDouble(input.substring(20, 25)) / 100;
        optionPrice = Double.parseDouble(input.substring(25, 30)) / 100;
        assetPrice = Double.parseDouble(input.substring(35, 40)) / 100;
    }

    // Overloaded constructor for input file requirements for implied volatility
    // calculations in UserInterface
    public entry(String input, boolean market) {

        if (market) {

            Scanner in = new Scanner(input);
            in.useDelimiter("\t");

            symbol = in.next();

            // RecordType not present in input file for implied volatility
            recordType = 0;
```

```
            year = in.nextInt();
            month = in.nextInt();
            day = in.nextInt();
            time = in.nextInt();
            expMonth = in.nextInt();
            if (in.next() == "Put")
                isPut = true;
            else
                isCall = true;

            strikePrice = in.nextDouble();
            optionPrice = in.nextDouble();
            assetPrice = in.nextDouble();

        }

    }

    // Returns a string which summarizes all relevant data in a readable manner
    public String outputAll() {

        String type;

        // convert integers and doubles to strings
        String exp = "" + expMonth;
        String timeStr = "" + time;
        String dayStr = "" + day;
        String optionStr = "" + optionPrice;
        String monthStr = "" + month;

        if (isPut)
            type = "Put ";
        else
            type = "Call";

        if (expMonth < 10) {
            exp = "0" + expMonth;
        }
        if (time < 100000)
            timeStr = "0" + timeStr;
        if (day < 10)
            dayStr = "0" + dayStr;

        if ((optionPrice / 0.1) % 1 == 0)
            optionStr = optionStr + "0";

        if (optionPrice < 10)
            optionStr = "0" + optionStr;
        if (month < 10)
            monthStr = "0" + monthStr;

        return symbol + " " + year + " " + monthStr + " " + dayStr + " "
                + timeStr + " " + exp + "  " + type + " " + strikePrice + "
"
                + optionStr + " " + assetPrice;
    }

    // Returns date as a string
    public String date() {
        return "" + year + month + day;
    }

    // Returns date as an integer (YYMMDD)
    public int getDate() {
        return year * 10000 + month * 100 + day;
    }

}
```