London Metropolitan University

A Java-Designed Six-Model European Asian Call Equity Option Application and an Analysis of the Usefulness of these Models

Plamen Stilyianov

Date Submitted: 19th September 2005.

A dissertation submitted in partial fulfilment of the requirements for the Degree of MSc in Financial Markets with Information Systems.

Acknowledgement

I would like to thank my tutor Brian A. Eales for the invaluable advice and support he gave me during the writing of this thesis.

Abstract

In this thesis I develop and discuss a calculator application to value Asian options and compare these to European call options and their hedge sensitivities. I have used four different models, namely, the Standard Vanilla Model, Arithmetic Model, Geometric Model and the Arithmetic Model with a Geometric Variate. The application interface has been designed in such a way as to allow an end-user to compare the option value and the corresponding hedge values of the various models. Using the standard market model assumption of a Black-Scholes environment with a log-normal stock value and deterministic interest rate, I have produced a variety of models using Java code to value arithmetic and geometric average rate options.

The distribution of arithmetic averages is unknown, which complicates the pricing of Asian options based on an arithmetic average and so a variety of approximations have been produced over the years in order to approximate the distribution, and therefore effectively value an Asian option. The arithmetic models used here are those by Turnbull and Wakeman (1991) and Levy (1992), geometric models are those by Kemna and Vorst (1990) and Curran (1992), the Arithmetic Model with Control Variate used is the Monte Carlo Simulation by Clewlow and Strickland (2004), and the Standard Vanilla Model is that by Black and Scholes. I use a Monte Carlo Valuation for European Arithmetic Asian Call Option with Geometric Asian Call Option Control Variate by Clewlow and Strickland (2004) as a benchmark.

Note

The application has been tested against models from Global Derivatives (website), Derivative Models (website) and Bloomberg by using data from Bloomberg for the Asian Options of IBM LN* stock**.

*IBN LN is the London Stock Exchange stock of the US-based IT company. It was chosen due to its high liquidity and its relatively high value means that the differences in values of the options in the models are clear.

**I use American terms throughout this thesis, so here stock is used rather than the British term share

1 Introduction

There has been a huge growth of financial derivatives markets during the over the last thirty years or so. Investors now rely on custom engineered products for hedging and managing their exposure to the markets which are usually structured to the needs of clients by investment banks. Of utmost importance is the accurate pricing of financial derivatives and since the invention of the pricing equation for plain vanilla European options by Black and Scholes there has been a wealth of publications from authors striving for analytic or numeric algorithms for valuing evermore complex and exotic options.

Exotic options have recently emerged as an important instrument for risk management. The increased interest in exotic option structures reflects the growth of the derivatives generally. It also reflects the increased demand for highly customised risk management/ hedging structures.

The importance of exotic options derives from a number of factors.

- Exotic options expand the available range of the risk management/risk transfer opportunities.
- They create unique valuation, pricing and hedging problems that provide additional insights into the pricing and hedging of the options generally.
- Furthermore, they provide insight into risk dimensions such as correlation/covariance and uncertain cash flow hedging.

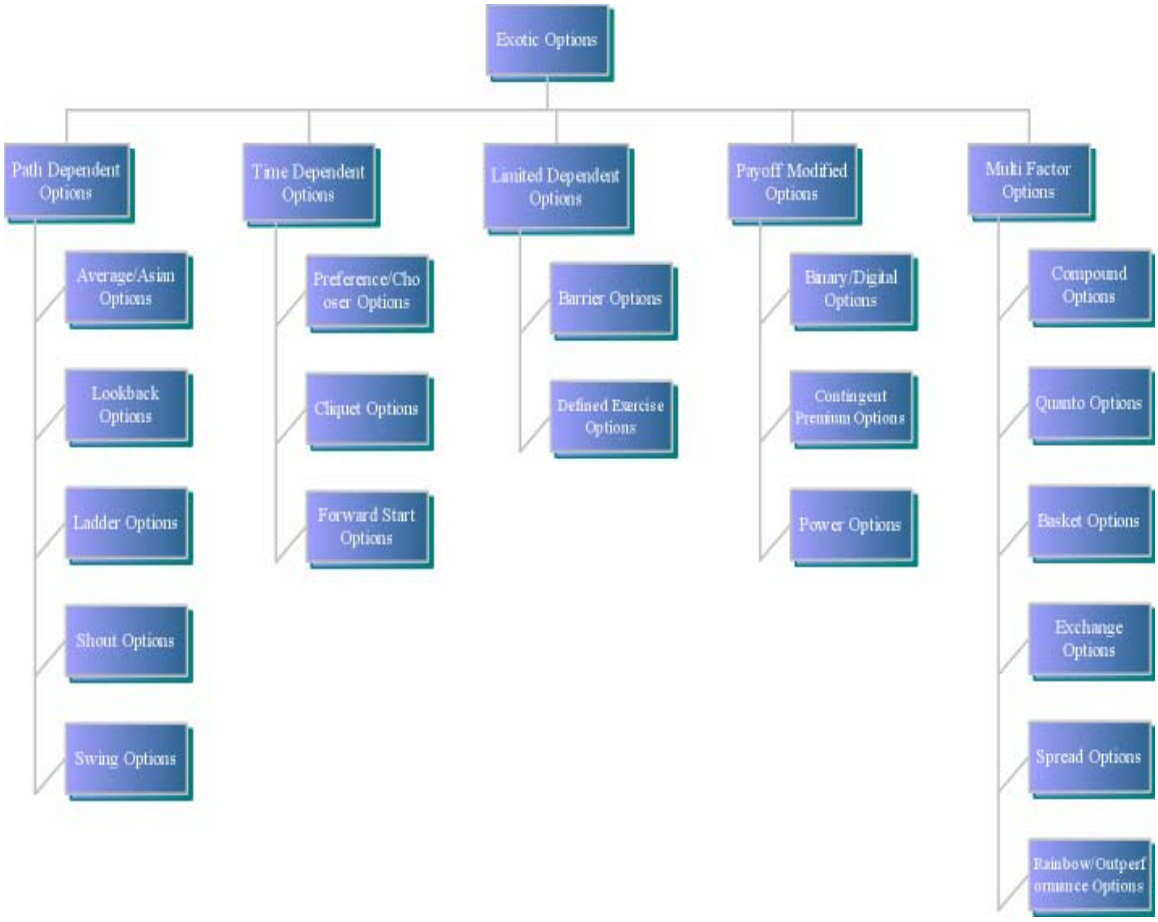

Figure 1.1 Exotic Options – Hierarchy.

The category of exotic options which are investigated here are path dependent Average/Asian options. These options are issued on the underlying asset over a pre-specified period and are known as average rate options. Asian options are used within the foreign exchange, interest rate and commodity markets, which Milevsky and Posner (1998) have estimated as being in the order of $5-$10 billion in outstanding balances. Within the foreign exchange market, it is possible to cover a series of foreign currency flows with a single hedging instrument whose exercise is cash-settled; for commodities such as oils and metals average value options reduce the risk of value manipulation of the underlying asset when close to expiration, hence both the investor and the issuer can obtain protection from fluctuations in the market; and in commodity-linked bond markets Asian options allow the investor to share in the wealth of the company in which they have invested, which also allows the issuer of the bond not to be subjected to a higher maturity date value which may occur after having months of low values, also as Nielsen and Sandmann (1998) comment that Asian options can also be used as part of complex financial contracts and strategies, such as retirement plans, life insurance, or corporate financing.

Asian Options are cheaper than Standard Vanilla Options, and their averaging features make them a preferred method for hedging needs. I have investigated five samples for ITM, ATM, OTM of an IBM LN 90 day Stock Option, which are shown in Figures 1.2 and 1.3. and confirm this pricing difference with data taken from Bloomberg for IBM LN stock in Appendix A. The valuation of the European call option has been done by using the Black-Scholes model, while the Asian option has been valued by the Arithmetic Model with Geometric Variate used in the Monte Carlo Simulation.

S	K	T	r	d	sig	MCCV	BS
4493	4473	0,24658	0.04314	0.0087	0.21087	125.48	216.35
4493	4483	0,24658	0.04314	0.0087	0.21087	120.22	211.17
4493	**4493**	**0,24658**	**0.04314**	**0.0087**	**0.21087**	**115.10**	**206.06**
4493	4503	0,24658	0.04314	0.0087	0.21087	110.13	201.04
4493	4513	0,24658	0.04314	0.0087	0.21087	105.30	196.11

Table 1.1 IBM LN Stock.

S - Initial Stock Value, K - Strike price, T- Time to maturity, r - Risk Free Rate, D - Dividend Yield, sig -Volatilities, BS - Black and Scholes. MCCV - Monte Carlo with Control Variate.

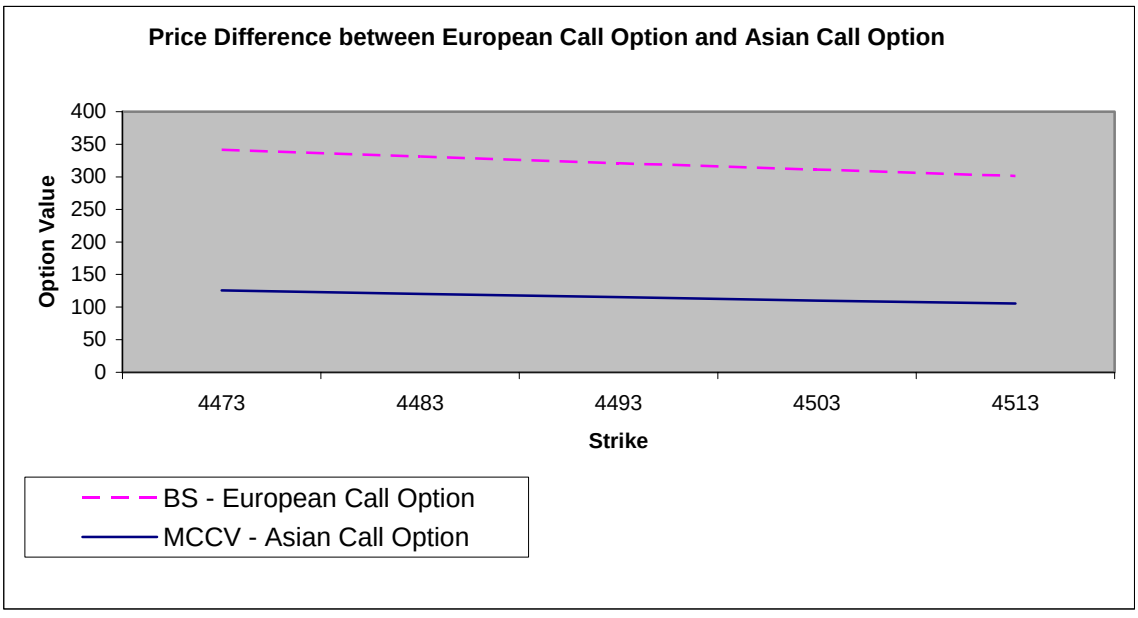

Figure 1.2 Asian and European Call Option Value Comparison of IBM LN Stock.

Models	European Call Option	Asian Call Option
Enhanced Discrete Dividends	206.3707	
Trinomial	206.1733	
Black-Scholes	205.9299	
Arithmetic		115.7449
Geometric		113.3578

Table 1.2 Asian and European Call Option Value Comparison of IBM LN Stock. (Data taken from Bloomberg) (see Appendix A).

Since the topic of Asian Options is very broad in this thesis I address only the valuing of European Asian Call Options. Hedge sensitivities are also shown for only information purposes as at the time of writing I could not find a reliable source for arithmetic or geometric calculation of 'the Greeks' (Hedge Sensitivities). The formulas used for computation of the Asian hedge sensitivities has been obtained through discussion forums (Global-Derivatives.com and Wilmott.com) which means they are strictly proprietary models and therefore I can claim no responsibility for their accuracy (except for the Standard Vanilla options). The aim of this work is to research, discuss, and implement an application for the pricing of Asian and European call options with which to provide the end-user with a quick and easy access to the calculated option value and hedge sensitivities of the models. The application has been developed with Java Swing J2SE 1.50. For testing the accuracy of the application models I have used data of IBM LN stock and calculation models from Bloomberg for Arithmetic Averaging, Geometric Averaging, Trinomial and Black-Scholes option value models (see Appendix A). (See Appendix C for the Java source code).

2 Definitions of Derivative Terms

In this chapter I provide a general description of the derivatives and the models which have been used throughout this work. Also, I describe the payout of a standard European call option and an Asian Average Rate Option which is essential to an understanding of the valuing of Asian options discussed in later chapters.

2.1 Basic Options

Options come in both exchange-based and OTC varieties and there are two types of option classes available: calls and puts. Buying an option bestows on the purchaser (the holder) a privilege, and this privilege is the right but not the obligation to buy (call) or sell (put) a specified underlying

security at a specified value (strike or exercise value) on a specified future date (European-style options) or on or before specified future date (American-style options) .

2.2 Path Dependent Options

Path dependent options are a type of exotic options. In conventional options, the value of the option is based on the value of the underlying asset at expiry of the option. This is the case, particularly with European options. The path of the underlying asset value can be used to determine the payoff or the structure of the option. There are a number of variations on the structures outlined. These represent structural changes to the basic elements of the primary path dependent options. Path dependency in option structure is a source of value to the option users and the demand for path dependent structures is driven by a number of factors, including:

- The nature of the underling asset or liability allows a path dependent hedge.
- The fact that the value of the option will rarely be at its maximum level at the maturity of the option.

Figure 1.1 sets out the major types of path dependent options. Path dependent exotic options include:

- Average rate(or Asian) options
- Average rate strike options
- Look-back options
- Ladder options
- Shout options
- Swing options

2.3 European Option Pricing

The Black-Scholes formula framework has been used for pricing the standard vanilla options, where the underlying asset St (spot value) (which has a log-normal distribution) follows a geometric Brownian motion:

$$dS = \mu dt + \sigma dSd \qquad (2.1)$$

Here, the first component says that during a small interval of time *dt* the average return on the asset is μ*dt*, this is deterministic and the parameter μ is known as the drift. Added to the drift term is a random component made up of a change, *dz*, in a random variable *z*, and a parameter σ, which is generally referred to as the volatility of the asset.

The Black-Scholes model is based on the following assumptions:

1. No dividends are paid out on the underlying stock during the option life.
2. The option can only be exercised at expiry (European characteristics).
3. Efficient markets (Market movements cannot be predicted).
4. Commissions are non-existent.
5. Interest rates do not change over the life of the option (and are known).
6. Stock returns follow a log-normal distribution.

The basic inputs to value a European option on a non-dividend paying stock is as follows:

S - Underlying stock value
X - Strike price
r - Risk free rate of interest
V - Volatility
T-t - Time to maturity

We can then apply these input variables into the following set of equations to derive the value for a European call option on a non-dividend stock:

$$c_t = SN(d_1) - Xe^{-r(T-t)}N(d_2) \qquad (2.2)$$

And for a European put option on a non-dividend stock:

$$p_t = Xe^{-r(T-t)}N(-d_2) - SN(-d_1) \qquad (2.3)$$

Where N(d1) and N(d2) are the cumulative normal distribution functions for d1 and d2, which are defined as:

$$d_1 = \frac{\ln(\frac{S_t}{X}) + (r + 0.5\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \quad (2.4)$$

$$d_2 = \frac{\ln(\frac{S_t}{X}) + (r - 0.5\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \quad (2.5)$$

d2 can be further simplified as:

$$d_2 = d_1 - \sigma\sqrt{T-t} \quad (2.6)$$

By means of substitution.

In order to compute the cumulative normal distribution function, we can consider the partial derivative of N(x).

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-z^2}{2}} dz \quad (2.7)$$

We then apply the terms d_1 and d_2 to the equation and we obtain the solutions to the terms (as defined above). We will shortly discuss the Partial differential equations resulting in the Black-Scholes equation and its 'Greeks'.

2.4 Asian Option Pricing

A number of variations to the basic average rate options exist, including:

- Differed average rate options.
- Double average rate options.
- Cumulative average rate options.
- Average strike rate options.

2.4.1 Average Calculation

Asian options are broadly segregated into the following categories; arithmetic average Asian and geometric average Asian, and both of these forms can be averaged on a weighted average basis, whereby a given weight is applied to each stock being averaged. This can be useful for attaining an average on a sample with a highly skewed sample population.

For the discrete cases, there are n values that have been observed at times

$0 \leq t_1 < t_2 < t_3 < \ldots < t_n \leq T$, and tend to be in equidistant periods, such as daily, weekly or monthly averages over one year.

- Discrete Arithmetic Average:

$$A = \frac{1}{n}\sum_{n=1}^{n} S_{t_i} \tag{2.8}$$

- Continuous Arithmetic Average:

$$A = \frac{1}{T-t}\int_{t}^{T} \ln(Su)du \tag{2.9}$$

- Discrete Geometric Average:

$$G = \left(\prod_{i=1}^{n} St_i\right)^{\frac{1}{n}} \tag{2.10}$$

- Continuous Geometric Average:

$$G = e^{\left(\frac{1}{T-t}\int_{t}^{T} \ln(Su)du\right)} \tag{2.11}$$

2.4.2 Option Valuation

The pricing/valuation of average rate options must incorporate the following factors in practice:

- Path dependency of the average asset value, the type of average (arithmetic or geometric), and the nature of the distribution of the average value.
- The averaging process is not continuous in practice. The average value in practice must be calculated on a discrete basis (daily etc).
- The average process may not be over the full term of the option.

An Asian option with average A, strike K and maturity T has the payoff given by:

- Fixed Strike Asian Call Option:

$$\max(A - K, 0) \tag{2.12}$$

- Floating Strike Asian Call Option:

$$\max(S_T - A, 0) \tag{2.13}$$

I have focused exclusively on the fixed strike Asian call option throughout this work.

3 Average Rate (Asian) Options

The average rate option is a path dependent option. Average rate options are also referred to as Asian options. The average rate option is similar to a conventional option. The major difference with conventional options is the applicable payoff at maturity or exercise. Under a conventional option, the payoff is determined as the difference between the strike price/rate of the option and the spot value rate at maturity or exercise date. Under an average rate option however, the payoff is determined as the difference between the strike price/rate of the option and the average spot rate over a specified period (usually the life of the option).

The major features of average rate options include:

- The ability to hedge a series of regular or irregular cash flows with option payoffs being based on the average value/rate rather than a single final value/rate observation. There is an effective reduction in premium cost relative to using a series of options to hedge this type of exposure.
- A lower premium cost relative to comparable conventional options.

Average rate options are used in all asset classes and are used extensively in foreign exchange and commodity markets where the average structure facilitates hedging of series of regular cash flows. An average is less volatile than the underlying asset itself and will lower the value of an average-rate option compared with similar conventional options. If the option is based on an average, an attempt to manipulate the asset value just before expiration will normally have little or no effect on the option's value. Asian options should therefore be of particular interest in the markets for thinly traded assets.

The average can be defined as an arithmetic or geometric average, which is calculated using continuous or discrete observations. The payout is the difference between this average and a pre-defined strike price. It is possible to have the so-called average strike Asian options whose payoffs are represented by the difference between the last asset value and one of the above averages.

Another class of average rate options is the flexible Asian options which allocate different weights to value observations and are now being extensively used. The majority, however, are European style options based on un-weighted arithmetic averages of the underlying asset. The main difference between arithmetic averages and geometric ones in the case of options is that the latter are log-normally distributed, while the former are not. In the standard Black-Scholes model all security values are log-normally distributed, so for a geometric Asian option, the product of the

correlated log-normal random variables is also log-normally distributed and this results in a log-normal state value density function that means that risk neutral expectations can be used to obtain the analytic no arbitrage value of the option and this is the reason why geometric Asian options can be relatively easy to value in a Black-Scholes environment. However, arithmetic Asian options use a sum of correlated log-normal random variables, which is unfortunately not log-normally distributed; hence there is no closed form probability density function. Since no general analytical solution for the value of such options is known, various techniques have been developed to analyse arithmetic average Asian options. The general models used can be described as follows:

3.1 Arithmetic Average–Rate Models

It is not possible to find a closed-form solution for the valuation of the options on an arithmetic average as when the asset is assumed to be log-normally distributed, the arithmetic average will not itself have a log-normal distribution. Arithmetic average-rate options can be valued by analytical approximation. The methods for arithmetic average-rate calculations I have used are as:

1. Turnbull and Wakeman Approximation

Turnbull and Wakeman base their approximations on a moment matching technique using the Edgeworth expansion. The moment matching technique is very accurate for short term maturities, but less accurate for longer maturities.

2. Levy's Approximation

The Levy approximation is slightly more accurate than the Turnbull and Wakeman approximation. The main difference between the Levy and the Turnbull and Wakeman approximations are that the Turnbull and Wakeman approximation use an Edgeworth series expansion, where they have overlooked the fact that when only the first two moments are taken into the approximation, the log-normal assumption is still accurate enough for pricing, which means that higher moments in the expansion are not required.

3.2 Geometric Average–Rate Models

If the underlying asset is assumed to be log-normally distributed, the geometric average of the asset will itself be log-normally distributed. The methods for geometric average-rate calculations I have used are as;

1. Kemna and Vorst

The Kemna and Vorst method is to value arithmetic Asian options using a geometric average, similar to a control variate technique, the method relies on adjusting the strike price by the difference obtained between the expectation of the arithmetic and geometric averages.

2. Curran

Curran has developed an approximation method for pricing Asian options based on the geometric conditioning approach.

3.3 Arithmetic Model with Control Variate

1. Monte Carlo

The particular implementation of a Monte Carlo Simulation I use is described in Clewlow and Strickland (1998). This model uses the value of a geometric Asian option as a control variate against which the arithmetic Asian value is then valued. Therefore, what is actually simulated is the difference between the two option types.

4 Models and Analysis

4.1 Monte Carlo Simulation Model

In the introduction of this thesis (2.1), I mentioned that there is a vast amount of literature on the pricing of Asian options. For the most important ones, solutions have been found numerically or analytically. I use the Monte Carlo Model developed by Clewlow and Strickland (2004).

In recent years, the complexity of simulation methods has increased considerably, in a effort for greater accuracy and speed. The Monte Carlo method in particular can be applied for a variety of purposes: valuation of securities, estimation of their sensitivities, assessment of the hedging performance, risk analysis, stress testing, amongst others. One of the most important parts in a Monte Carlo Simulation is the random number generator, which is needed in order to produce independent and identically distributed numbers given by u_1, u_2, . . . In this model, there is a requirement for a uniform distribution on [0, 1]. There are several known approaches in which the random number generator can be created, but my chosen approach is to use a method where the initial seed is fixed. This will enable the simulation to generate the same set of random numbers, hence allowing the user to compare the simulations with the various analytic methods implemented. If the random number generator does not generate random numbers from that initial seed, then the paths will be different every time the same simulation is run, hence the option value will vary every time we run the simulation. For example, this would have no obvious use in industry, since a trader would not want a variety of option values for a particular option.

From the literature it can be seen that for an Asian option, the best Monte Carlo Simulation uses a polar reject method in conjunction with the Box-Muller method where a geometric average is used as a control variate. The geometric Asian option makes a good static hedge style control variate for the Arithmetic Asian option, whilst the Box-Muller method is a faster method when compared to other inversion methods which are used to transform our uniformly distributed numbers into a set of normally distributed numbers.

4.1.1 Pseudo-Random Number Generator

Pseudo-random generators are deterministic algorithms that produce numbers with certain distribution properties. The numbers have to be independent and identically distributed random variables. The Math.random() method is the standard pseudo-random number generator for J2SE 1.5. The Math.random() generates numbers which are uniformly distributed. Using the cumulative density function which has values between 0 and 1, it is possible to convert this into random numbers of any chosen distribution. The distribution requirement is that of a normal distribution. Other techniques available for this conversion are the Box-Muller and the Polar Rejection methods. They both use two U(0, 1) pseudo-random numbers and convert them to normal N(0, 1). In order to produce more accurate results, I shall be using the Random utility API class with initial fixed seed as −1, so the Random class method nextDouble() will be generating random numbers from the class instance Random (-1), hence starting the sequence from the same seed will always return the same sequence, which allows us to compare all the different analytic methods of pricing against the same standard. This random number generator is not periodic; hence sequences will not repeat themselves.

4.1.2 Polar Rejection Box-Muller Method

To transform the set of uniformly distributed numbers into normally distributed numbers, the Box-Muller method is used.

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy \qquad (4.1)$$

This method is considered as the fastest method for transformations. This considers two uniform deviates u_1 and u_2 and two quantities y_1 and y_2 given by:

$$y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2 \qquad (4.2)$$

$$y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2 \qquad (4.3)$$

The two uniform deviates u_1 and u_2 can be taken as arbitrary points u_1 and u_2 on a unit circle, so that $R^2 = \upsilon_1^2 + \upsilon_2^2$ is the uniform deviate, which is used instead of u_1, hence (u_1, u_2) can be used

as the angle with respect to the υ_1 axis in place of the angle $2\pi x_2$. Therefore, the sine and cosine in the above equations can now be expressed as $\frac{\upsilon_1}{\sqrt{R^2}}$ and $\frac{\upsilon_2}{\sqrt{R^2}}$, hence there is no need to call upon the trigonometric functions within J2SE.

4.1.3 Arithmetic Average Model

In this model I value a European Arithmetic Asian (average value) call option. This option pays the difference, if positive, between the arithmetic average of the asset value A_T and the strike price K at the maturity date T. The arithmetic average is taken on a set of observations (fixings) of the asset value S_{t_i} (which we assume follows GBM) at the dates t_i ; $i = 1,\ldots, N$

$$A_T = \frac{1}{N}\sum_{n=1}^{n} S_{t_i} \qquad (4.4)$$

Thus the pay-off at the maturity date is $\max(0, A_T - K)$

4.1.4 Geometric Control Variate

The control variate technique can be used to find more accurate analytical solutions to a derivative value if there is a similar derivative with a known analytic solution. With this in mind, MCS is then undertaken on the two derivatives in parallel.

Given the value of the geometric Asian, the arithmetic Asian can be valued by considering the equation:

$$G_T = \left(\prod_{i=1}^{N} S_{t_i}\right)^{\frac{1}{N}} \qquad (4.5)$$

Since the geometric average is essentially the product of log-normally distributed variables then it is also log-normally distributed, Therefore the value of the geometric Asian call option is given by the modified Black-Scholes formula:

$$C_{GEOMETRIC\_ASIAN} = \exp(-rT)\left(\exp(a+\frac{1}{2}b)N(x) - KN(x-\sqrt{b})\right) \qquad (4.6)$$

Where

$$a = \ln(G_t) + \frac{N-m}{N}(\ln(S) + v(t_{m+1} - t) + \frac{1}{2}v(T - t_{m+1})) \quad (4.7)$$

$$b = \frac{(N-m)^2}{N^2}(\sigma^2(t_{m+1} - t) + \frac{\sigma^2(t - t_{m+1})}{6N^2}(N-m)(2(N-m)-1) \quad (4.8)$$

$$v = r - \delta - \frac{1}{2}\sigma^2 \quad (4.9)$$

$$x = \frac{a - \ln(K) + b}{\sqrt{b}} \quad (4.10)$$

where G_t is the current geometric average and m is the last known fixing.

4.1.5 Simulation

In my Monte Carlo Model I simulate the difference between the arithmetic and geometric Asian options or a hedge portfolio which is long one arithmetic Asian and short one geometric Asian option. This is much faster than using delta of the geometric Asian option to generate a delta hedge control variate because we do not have to compute the delta at every time step and it is equivalent to a continuous delta hedge. This increases the efficiency of the simulation significantly because these constants are used for every time step for every simulation.

4.1.6 Calibration and Optimisation Techniques

In this model I use two approaches to optimise the running time and the accuracy of the Monte Carlo Application.

- Adjusting the number of simulations.

Simulations	Time Step	Option Value	Running Time
10	10	124.0565	VG
100	10	124.0565	VG
1000	10	124.0565	VG
10000	10	124.0565	VG
100000	10	124.0565	G
1000000	10	124.0565	G

Table 4.1 Monte Carlo output with fixed time steps.
VG - very good time, G – good time

The results from Table 4.1 show that with the increase of the simulations, the option value is not changed, and so the method for optimisation with adjustments of simulation is rejected.

-
- Adjusting the time step.

Simulations	Time Step	Option Value	Running Time
10	10	124.0565	VG
10	100	115.9152	VG
10	**1000**	**115.1004**	**VG**
10	10000	115.0190	VG
10	100000	115.0108	G
10	1000000	115.0100	NG
Simulations	Time Step	Option Value	Running Time
100	10	124.0565	VG
100	100	115.9152	VG
100	**1000**	**115.1004**	**VG**
100	10000	115.0190	NG
100	100000	115.0108	NG
100	1000000	n/a	NG
Simulations	Time Step	Option Value	Running Time
1000	10	124.0565	VG
1000	100	115.9152	VG
1000	**1000**	**115.1004**	**G**
1000	10000	115.0190	G
1000	100000	115.0108	NG
1000	1000000	n/a	NG

Table 4.2 Monte Carlo with fixed number of simulations.
VG - very good time, G – good time, NG – no good.

The results from Table 4.2 show that there is a significant difference in the option value when the time steps are changed. The option value falls from £124.065 to £115.0108, so it converges at around £115 with an increase in the time steps.

The data used in the simulation comes from the stock of IBM LN, sourced from Bloomberg and has the following specifics.

S = 4493, K = 4493, r = 4.314%, d = 0.87%, σ = 21.097%, T = 0.24658 - 90 days.
Bloomberg Option Valuation (Appendix A).
Arithmetic Average (£) = 115.7449.
Geometric Average (£) = 113.3578.

We can see from the results above that the optimisation makes sense for calibration values of **simulations = 100**, **time step = 1000** where the running time is very good, and the option value is 115.1004. My model's option value of £115.1004 is very similar to the Bloomberg Arithmetic Average Option value of £115.7449 and so I conclude that the benchmark model is acceptable.

4.2 Turnbull and Wakeman Model

The model below is based on the work of Turnbull and Wakeman (1991). The approximation adjusts the mean and variance so that they are consistent with the exact moments of the arithmetic average. The adjusted mean, b_A, and variance, σ_A^2, are then used as input in the Generalised Black-Scholes model by Black and Scholes (1993) in this case I have replaced the Black-Scholes Model with the model by Black described by Haug (1998).

$$c_{TW} \approx e^{-rT_2}(M_1 N(d_1) - KN(d_2)) \text{ - Black'76} \qquad (4.11)$$

$$c_{TW} \approx Se^{(b-r)T_2} N(d_1) - Ke^{-rT_2} N(d_2) \qquad \text{- Gen. Black and Scholes} \qquad (4.12)$$

$$d_1 = \frac{\ln(\frac{S}{K}) + (b_A + 0.5\frac{\sigma_A^2}{2})T_2}{\sigma_A\sqrt{T_2}} \qquad (4.13)$$

$$d_2 = d_1 - \sigma_A\sqrt{T_2} \qquad (4.14)$$

where T_2 is the time remaining until maturity. For averaging options which have already begun their averaging period T_2 is simply T (the original time to maturity), if the averaging period has not yet begun, then T_2 is T - τ

The adjusted volatility and dividend yield are given as:

$$\sigma_A = \sqrt{\frac{\ln(M_2)}{T} - 2b_A} \tag{4.15}$$

$$b_A = \frac{\ln(M_1)}{T} \tag{4.16}$$

To generalise the equations, I assume that the averaging period has not yet begun and give the first and second moments as:

$$M_1 = \frac{e^{(r-D)T} - e^{(r-D)\tau}}{(r-D)(T-\tau)} \tag{4.17}$$

$$M_2 = \frac{2e^{(2(r-D)+\sigma^2)T}S}{(r-D+\sigma^2)(2r-2q+\sigma^2)T^2} + \frac{2S^2}{(r-D)T^2}\left(\frac{1}{2(r-D)+\sigma^2} - \frac{e^{(r-D)T}}{r-D+\sigma^2}\right) \tag{4.18}$$

If the averaging period has already begun, the strike price must be adjusted accordingly:

$$K_A = \frac{T}{T_2}K - \frac{(T-T_2)}{T_2}S_{Avg} \tag{4.19}$$

where T is the original time to maturity, T_2 the remaining time to maturity, K is the original strike price and S_{avg} is the average asset value. Haug (1998) notes that if r = D, the formula will not generate a solution.

4.2.1 Results

In this section I investigate how the change of the Asian option value responds to the change of the strike price within the bounds of 10% and 30% volatility, and also how the value responds to the length of time of the maturity. All of the results are based on the Asian option of IBM LN stock provided by Bloomberg. The methods used here are based on the Monte Carlo Model with Control Variate (my benchmark model) against the Turnbull-Wakeman Model.

The following abbreviations are used:
S - Spot Value
K - Strike price
T - Time to Maturity
r - Risk Free Rate
d - Dividend
sig - Volatility
MCCV - Monte Carlo with Control Variate Model
TW - Turnbull and Wakeman Model
OTM - out of money
ATM - at the money
ITM - in the money

- Volatility Criteria

S	K	T	r	d	sig	MCCV	TW
4493	4443	0,24658	0.04314	0.0087	0.1	91.4300	92.0941
4493	4453	0,24658	0.04314	0.0087	0.1	84.6678	85.3077
4493	4463	0,24658	0.04314	0.0087	0.1	78.1862	78.8007
4493	4473	0,24658	0.04314	0.0087	0.1	71.9932	72.5817
4493	4483	0,24658	0.04314	0.0087	0.1	66.0957	66.6574
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.1	**60.4987**	**61.0330**
4493	4503	0,24660	0.04314	0.0087	0.1	55.2054	55.7118
4493	4513	0,24661	0.04314	0.0087	0.1	50.2174	50.6957
4493	4523	0,24662	0.04314	0.0087	0.1	45.5346	45.9834
4493	4533	0,24663	0.04314	0.0087	0.1	41.1553	41.5739

Table 4.3 Comparison of the Asian Option Value of Monte Carlo and Turnbull-Wakeman Models with a volatility of 10%.

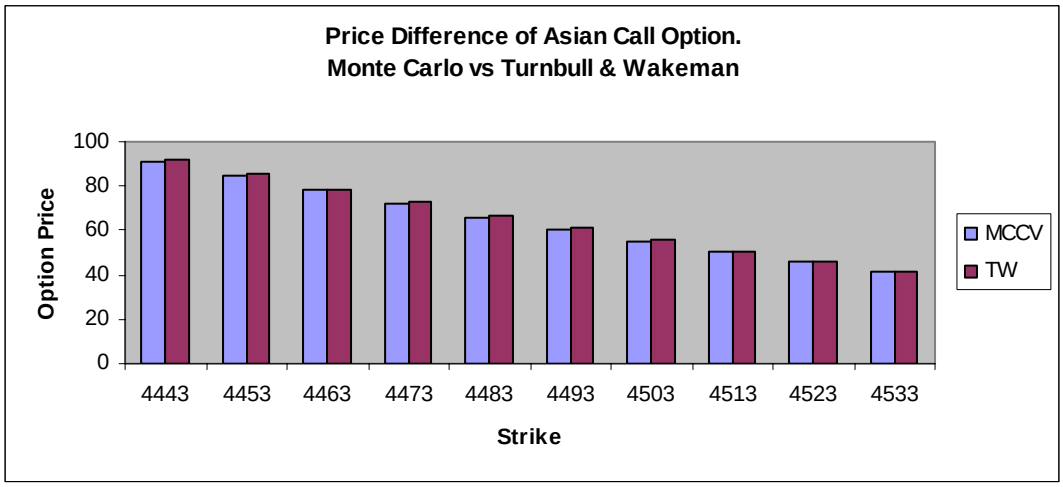

Figure 4.1 Comparison of the Asian Option Value of Monte Carlo and Turnbull-Wakeman Models with a volatility of 10%.

In Table 4.3 and Figure 4.1 it can be seen that the change of the option values from ITM to OTM of the MCCV and TW models decreases, and eventually they will converge to the point of zero when the option will expire as worthless. I test the option values of the models at ATM with a volatility of 10% and it can be seen from the results that there is a negative difference between the models of 60.4987 – 61.0330 = ± 0.5343.

S	K	T	r	d	sig	MCCV	TW
4493	4443	0,24658	0.04314	0.0087	0.3	183.9030	188.9004
4493	4453	0,24658	0.04314	0.0087	0.3	178.5548	183.4689
4493	4463	0,24658	0.04314	0.0087	0.3	173.3092	178.1397
4493	4473	0,24658	0.04314	0.0087	0.3	168.1662	172.9129
4493	4483	0,24658	0.04314	0.0087	0.3	163.1257	167.7883
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.3	**158.1877**	**162.7659**
4493	4503	0,24660	0.04314	0.0087	0.3	153.3518	157.8455
4493	4513	0,24661	0.04314	0.0087	0.3	148.6179	153.0266
4493	4523	0,24662	0.04314	0.0087	0.3	143.9856	148.3088
4493	4533	0,24663	0.04314	0.0087	0.3	139.4545	143.6919

Table 4.4 Comparison of the Asian Option Value between Monte Carlo and Turnbull-Wakeman Models with a volatility of 30%.

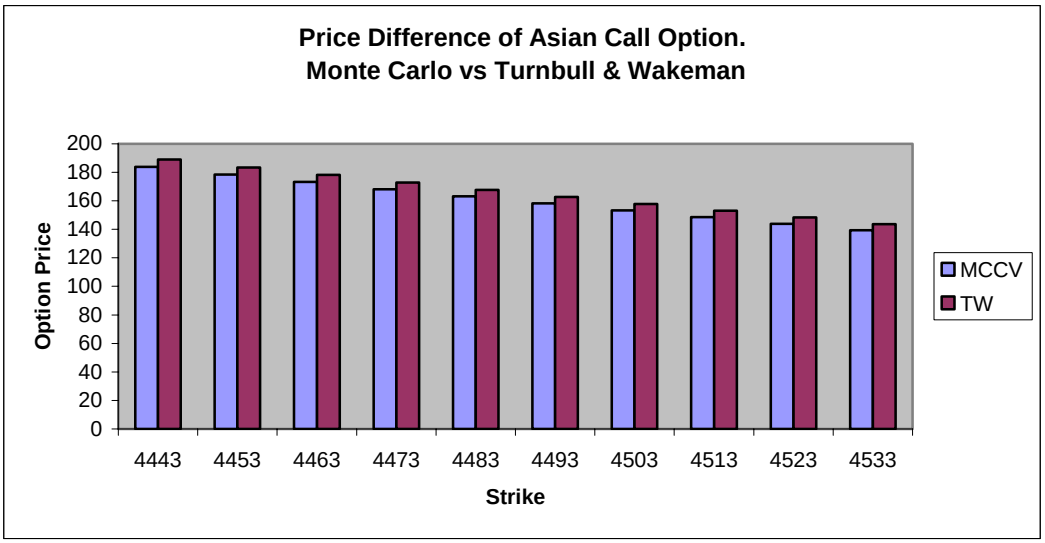

Figure 4.2 Comparison of the Asian Option Value between Monte Carlo and Turnbull-Wakeman Models with a volatility of 30%.

I use a volatility of 30% when the option is ATM and keep the rest of the parameters the same, and also compare this model with the earlier test with a 10% volatility. From the Table 4.4 and Figure 4.2 it can be seen that the difference in the option values of the models has increased with the increase of the volatility from 10% to 30%. The change in the option value at ATM of the models is 158.1877 – 162.7659 = ±4.5782. I conclude from this analysis that with an increase in the volatility the difference in the option values of the models also increases, in this case volatility from 10% to 30% shifts the value difference of the models from 0.5343 to 4.5782 and therefore it can be said that there is a positive relationship between the volatility and the difference in the option values of the MCCV and TW models.

- Time to Maturity Criteria

Here, I test the value of the ATM Asian Call option of IBM LN stock in response to changes in the time to maturity (I give the time to maturity in days). The original data of IBM LN stock is provided from Bloomberg and is for a period of 90 days maturity.

S	K	T	days	r	d	sig	MCCV	TW
4493	4493	0.0027	1	0.04314	0.0087	0.21087	11.4343	11.4501
4493	4493	0.0540	20	0.04314	0.0087	0.21087	52.3261	52.7680
4493	**4493**	**0.2465**	**90**	**0.04314**	**0.0087**	**0.21087**	**115.0809**	**117.3511**
4493	4493	0.4931	180	0.04314	0.0087	0.21087	165.8300	170.6582
4493	4493	0.7397	270	0.04314	0.0087	0.21087	205.5590	213.0963
4493	4493	0.9863	360	0.04314	0.0087	0.21087	239.4046	249.7632
4493	4493	1.2328	450	0.04314	0.0087	0.21087	269.3637	282.6315
4493	4493	1.4794	540	0.04314	0.0087	0.21087	296.4998	312.7519

Table 4.5 Comparison of the Asian Option Values between Monte Carlo and Turnbull- Wakeman Models with changes in the time to maturity.

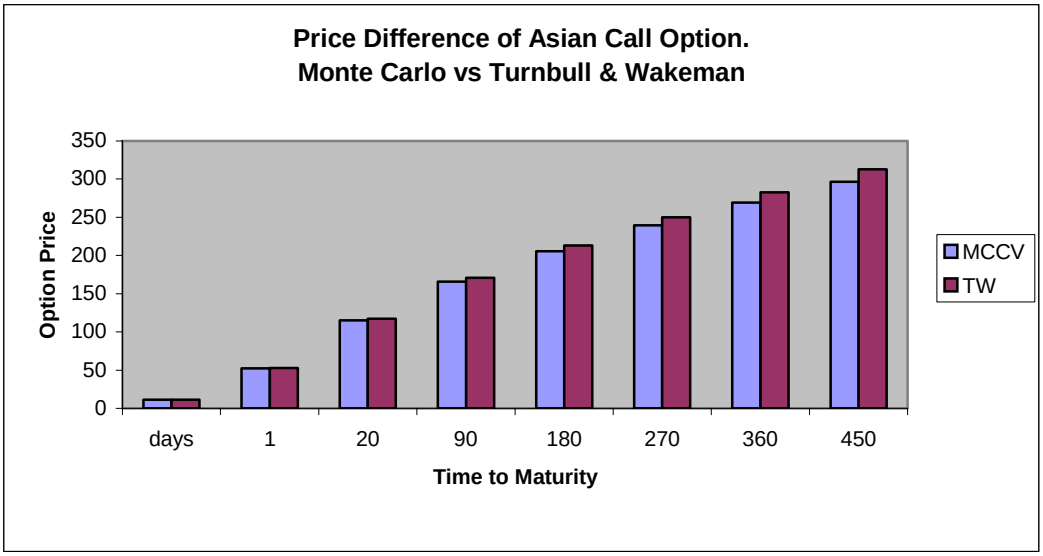

Figure 4.3 Comparison of the Asian Option Values between Monte Carlo and Turnbull-Wakeman Models with changes in the length of time to maturity.

Table 4.5 and Figure 4.3 show that with an increase of the time to maturity the difference in the value of the option between the MCCV and TW models has also increased. So our conclusion is that there is a positive relationship between the maturity and the difference of the option value between the models.

The above models show the following results for IBM LN stock; MCCV = 115.0808 and TW = 117.3511, so the difference is 2.3431 in favour of TW and I conclude that the TW model is a little overvalued compared to that of the MCCV.

4.3 Levy Model

The Levy approximation is slightly more accurate than the Turnbull and Wakeman approximation. The main difference between the Levy and the Turnbull and Wakeman approximations are that the Turnbull and Wakeman approximation use an Edgeworth series expansion, where they have overlooked the fact that when only the first two moments are taken into the approximation, the log-normal assumption is still accurate enough for pricing, which means that higher moments in the expansion are not required. A Wilkinson approximation is used by Levy in order to find a closed form analytical approximation for the Asian option. Levy (1992) explains that the main advantage of the approach is that it avoids the need to adopt time consuming numerical procedures.

The Wilkinson approximation is a procedure that has been utilised in communication engineering. Also, the Wilkinson approximation provides a quick and accurate method for valuing average rate options, and ensures that the put-call parity is maintained. Levy (1992) describes the Wilkinson method in detail, where the method approximates the mean $\alpha(t)$ and the variance $\upsilon(t)^2$ as functions of moments, by using moment generating functions.

Levy puts forward another analytical approximation which is suggested to give more accurate results than the TW approximation. I explore the differences later.

The approximation to a call is given as:

$$c_{Levy} \approx S_A N(d_1) - K_A e^{-rT} N(d_2) \qquad (4.20)$$

and through put-call parity, the value of a put is obtained:

$$p_{Levy} \approx c_{Levy} - S_A + K_A e^{-rT_2} \qquad (4.21)$$

Where

$$d_1 = \frac{1}{\sqrt{K}}\left[\frac{\ln(L)}{2} - \ln(K_A)\right] \tag{4.22}$$

$$d_2 = d_1 - \sqrt{K} \tag{4.23}$$

and

$$S_A = \frac{S}{(r-D)T}(e^{-DT_2} - e^{-rT_2}) \tag{4.24}$$

$$K_A = K - S_{Avg}\frac{T-T_2}{T} \tag{4.25}$$

$$K = \ln(L) - 2[rT_2 + \ln(S_A)] \tag{4.26}$$

$$L = \frac{M}{T^2} \tag{4.27}$$

$$M = \frac{2S^2}{r-D+\sigma^2}\left\{\frac{e^{(2(r-D)+\sigma^2)T_2}-1}{2(r-D)+\sigma^2}\right\} - \frac{e^{(r-d)T_2}-1}{r-D} \tag{4.28}$$

where the variables are the same as defined under the TW approximation.

4.3.1 Results

Here, I discuss how the change of the Asian option value will respond to the change of the strike price within the bounds of 10% and 30% volatility, and also how the value will respond with the length of time to the maturity. The results are based on the Asian option of IBM LN stock provided by Bloomberg. The methods used here are based on the Monte Carlo Model with Control Variate which is our benchmark model against the Levy model.

The following abbreviations are used:
S - Spot Value
K - Strike price
T - Time to Maturity
r - Risk Free Rate
d - Dividend
sig - Volatility
MCCV - Monte Carlo with Control Variate Model
Levy - Levy Model
OTM - out of money
ATM - at the money
ITM - in the money

- **Volatility Criteria**

S	K	T	r	d	sig	MCCV	Levy
4493	4443	0,24658	0.04314	0.0087	0.1	91.4300	92.0941
4493	4453	0,24658	0.04314	0.0087	0.1	84.6678	85.3077
4493	4463	0,24658	0.04314	0.0087	0.1	78.1862	78.8007
4493	4473	0,24658	0.04314	0.0087	0.1	71.9932	72.5817
4493	4483	0,24658	0.04314	0.0087	0.1	66.0957	66.6574
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.1	**60.4987**	**61.0330**
4493	4503	0,24660	0.04314	0.0087	0.1	55.2054	55.7118
4493	4513	0,24661	0.04314	0.0087	0.1	50.2174	50.6957
4493	4523	0,24662	0.04314	0.0087	0.1	45.5346	45.9834
4493	4533	0,24663	0.04314	0.0087	0.1	41.1553	41.5739

Table 4.6 Comparison of the Asian Option Value between Monte Carlo and Levy Models with a volatility of 10%.

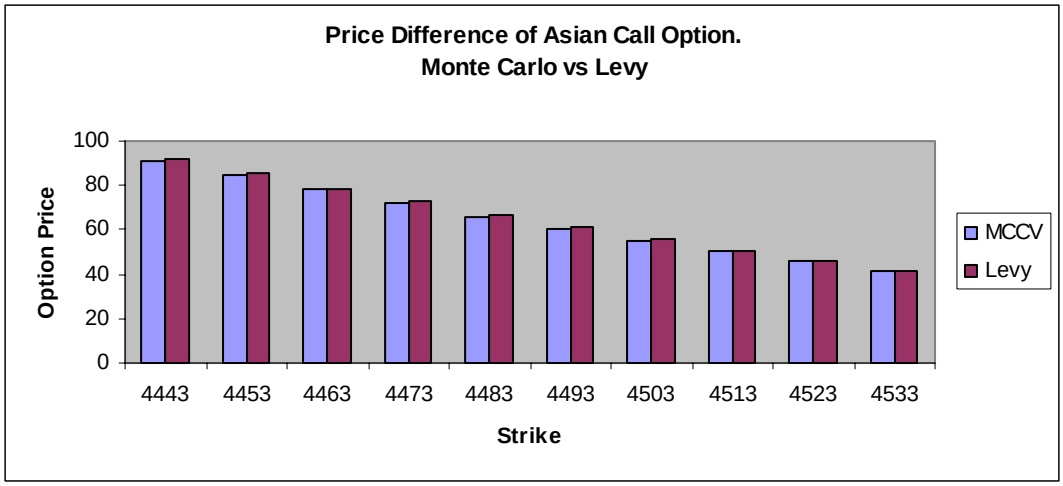

Figure 4.4 Comparison of the Asian Option Value between Monte Carlo and Levy Models with a volatility of 10%.

Table 4.6 and Figure 4.4 show that the change in the option values from ITM to OTM in the MCCV and Levy Models decreases and eventually they will converge to the point of zero when the option will expire as worthless. I test the option values of the models at ATM with a volatility of 10%. It can be seen from the results that there is some negative difference between the models of 60.4987 – 61.0330 = ± 0.5343.

S	K	T	r	d	sig	MCCV	Levy
4493	4443	0,24658	0.04314	0.0087	0.3	183.9030	188.9004
4493	4453	0,24658	0.04314	0.0087	0.3	178.5548	183.4689
4493	4463	0,24658	0.04314	0.0087	0.3	173.3092	178.1397
4493	4473	0,24658	0.04314	0.0087	0.3	168.1662	172.9129
4493	4483	0,24658	0.04314	0.0087	0.3	163.1257	167.7883
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.3	**158.1877**	**162.7659**
4493	4503	0,24660	0.04314	0.0087	0.3	153.3518	157.8455
4493	4513	0,24661	0.04314	0.0087	0.3	148.6179	153.0266
4493	4523	0,24662	0.04314	0.0087	0.3	143.9856	148.3088
4493	4533	0,24663	0.04314	0.0087	0.3	139.4545	143.6919

Table 4.7 Comparison of the Asian Option Value between Monte Carlo and Levy Models with a volatility of 30%.

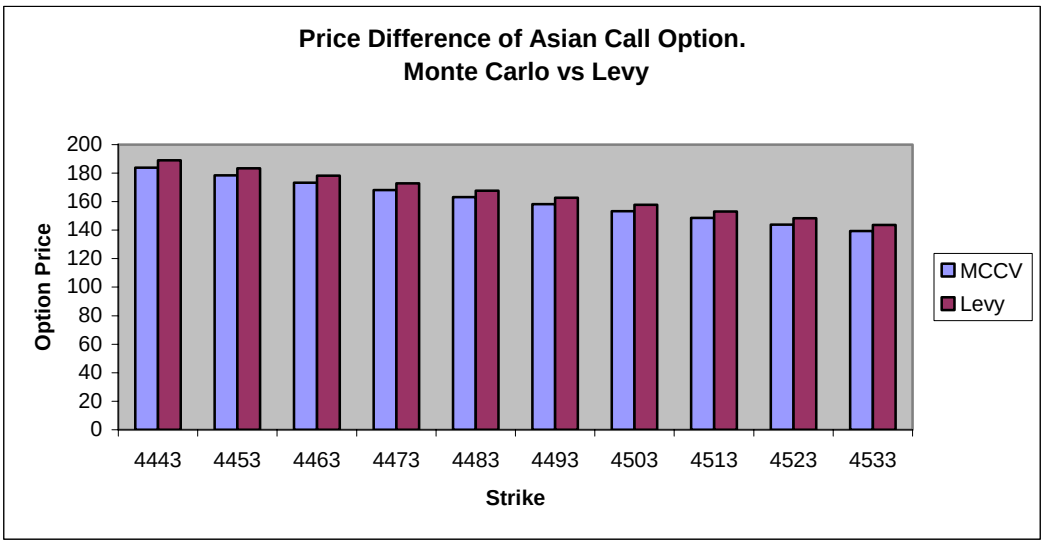

Figure 4.5 Comparison of the Asian Option Value between Monte Levy Models with a volatility of 30%.

Here, I use a volatility of 30% when the option is ATM and keep the rest of the parameters the same, and also compare it to the above mentioned earlier test with a volatility of 10%. From the Table 4.7 and Figure 4.5 it can be seen that the difference in the option value between the models increases with an increase of the volatility from 10% to 30%. The change of the option value at ATM of the models is 158.1877 – 162.7659 = ±4.5782. I conclude from this analysis that with an increase of the volatility the difference of the option value between the models also increases, in this case volatility from 10% to 30% shifts the value difference of the models from 0.5343 to 4.5782. So, it can be seen that there is a positive relationship between volatility and the difference of the option value between the MCCV and Levy models.

- Maturity Criteria

Here, I test the change in the value of an ATM Asian Call Option of IBM LN stock with a change in the time to maturity. (The maturity has been given in days). The original data of IBM LN stock is provided from Bloomberg and is for a period of 90 days maturity.

S	K	T	days	r	d	sig	MCCV	Levy
4493	4493	0.0027	1	0.04314	0.0087	0.21087	11.4343	11.4492
4493	4493	0.0540	20	0.04314	0.0087	0.21087	52.3261	52.7680
4493	**4493**	**0.2465**	**90**	**0.04314**	**0.0087**	**0.21087**	**115.0809**	**117.3511**
4493	4493	0.4931	180	0.04314	0.0087	0.21087	165.8300	170.6582
4493	4493	0.7397	270	0.04314	0.0087	0.21087	205.5590	213.0963
4493	4493	0.9863	360	0.04314	0.0087	0.21087	239.4046	249.7632
4493	4493	1.2328	450	0.04314	0.0087	0.21087	269.3637	282.6315
4493	4493	1.4794	540	0.04314	0.0087	0.21087	296.4998	312.7519

Table 4.8 Comparison of the Asian Option Values between Monte Carlo and Levy Models with changes in the length of time to maturity.

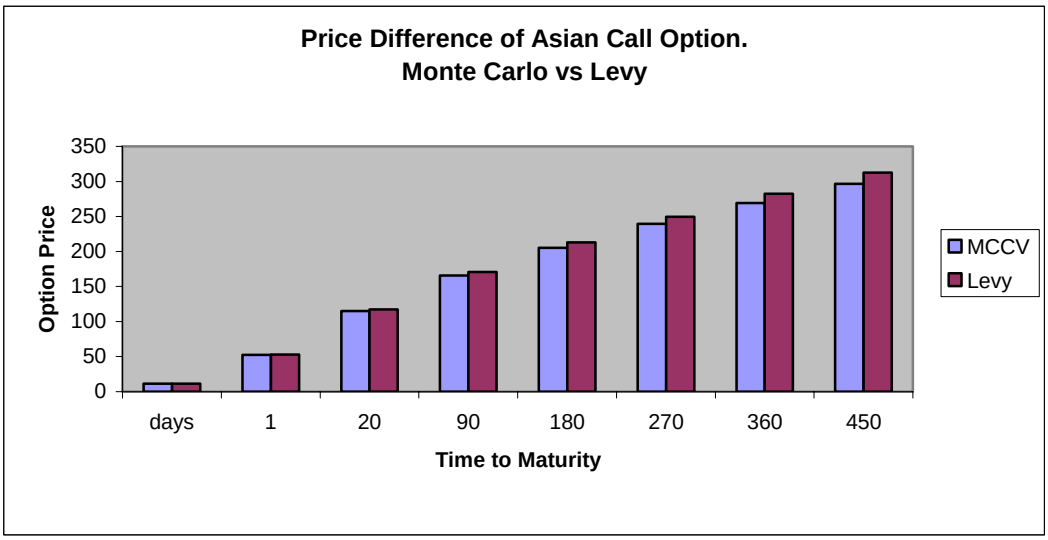

Figure 4.6 Comparison of the Asian Option Values between Monte Carlo and Levy Models with changes in the length of time to maturity.

Table 4.8 and Figure 4.6 show that with an increase in the time to maturity the difference in the value of the options of the MCCV and TW Models has also increased. I include that there is a positive relationship between the maturity and the difference in the option values of the models.

The models show the following results for IBM LN stock MCCV = 115.0808 and Levy = 117.3511, so the difference is 2.3431 is in favour of Levy, and I conclude that the Levy Model is overvalued compared to the MCCV Model.

4.4 Kemna and Vorst Model

Kemna and Vorst (1990) proposed a closed form pricing solution to geometric averaging options by altering the volatility and the cost of carry term. Geometric averaging options can be valued via a closed form analytic solution as the geometric average of the underlying values follows a log-normal distribution as well, whereas under average rate options, this condition collapses.

The solution to the geometric averaging Asian call is given as:

$$C_G = Se^{(b-r)(T-t)}N(d_1) - Ke^{-r(T-t)}N(d_2) \tag{4.29}$$

Where N(x) is the cumulative normal distribution function of:

$$d_1 = \frac{\ln(\frac{S}{K}) + (b + 0.5\sigma_A^2)T}{\sigma^2\sqrt{T}} \tag{4.30}$$

$$d_2 = \frac{\ln(\frac{S}{K}) + (b - 0.5\sigma_A^2)T}{\sigma^2\sqrt{T}} \tag{4.31}$$

which can be simplified to:

$$d_2 = d_1 - \sigma_A\sqrt{T} \tag{4.32}$$

The adjusted volatility and dividend yield are given as:

$$\sigma_A = \frac{\sigma}{\sqrt{3}} \tag{4.33}$$

Where σ is the observed volatility, r is the risk free rate of interest and D is the dividend yield.

$$b_A = \frac{1}{2}\left(r - D - \frac{\sigma^2}{6}\right) \tag{4.34}$$

4.4.1 Results

Here, I examine how the change in the value of the Asian option responds to a change in the strike price within the bounds of a volatility of 10% and 30%, and also how the value changes with the length of time to maturity. The results are based on the Asian call option of IBM LN stock provided by Bloomberg. The methods used here are based on the Monte Carlo Model with Control Variate which is my benchmark model against the Kemna and Vorst model.

The following abbreviations are used:

S - Spot Value
K - Strike price
T - Time to Maturity
r - Risk Free Rate
d - Dividend
sig - Volatility
MCCV - Monte Carlo with Control Variate Model
Vorst - Kemna and Vorst Model
OTM - out of money
ATM - at the money
ITM - in the money

- **Volatility Criteria**

Here, I use a volatility of 10% to compare the change of the option value and later we will also examine the value's respond to the volatility 30%.

K	T	r	d	sig	MCCV	Vorst
4443	0,24658	0.04314	0.0087	0.1	91.4300	91.3836
4453	0,24658	0.04314	0.0087	0.1	84.6678	84.6206
4463	0,24658	0.04314	0.0087	0.1	78.1862	78.1383
4473	0,24658	0.04314	0.0087	0.1	71.9932	71.9450
4483	0,24658	0.04314	0.0087	0.1	66.0957	66.0472
4493	**0,24659**	**0.04314**	**0.0087**	0.1	**60.4987**	**60.4502**
4503	0,24660	0.04314	0.0087	0.1	55.2054	55.1571
4513	0,24661	0.04314	0.0087	0.1	50.2174	50.1694
4523	0,24662	0.04314	0.0087	0.1	45.5346	45.4859
4533	0,24663	0.04314	0.0087	0.1	41.1553	41.1053

Table 4.9 Comparison of the Asian Option Value of the Monte Carlo and Kemna and Vorst Models with a volatility of 10%.

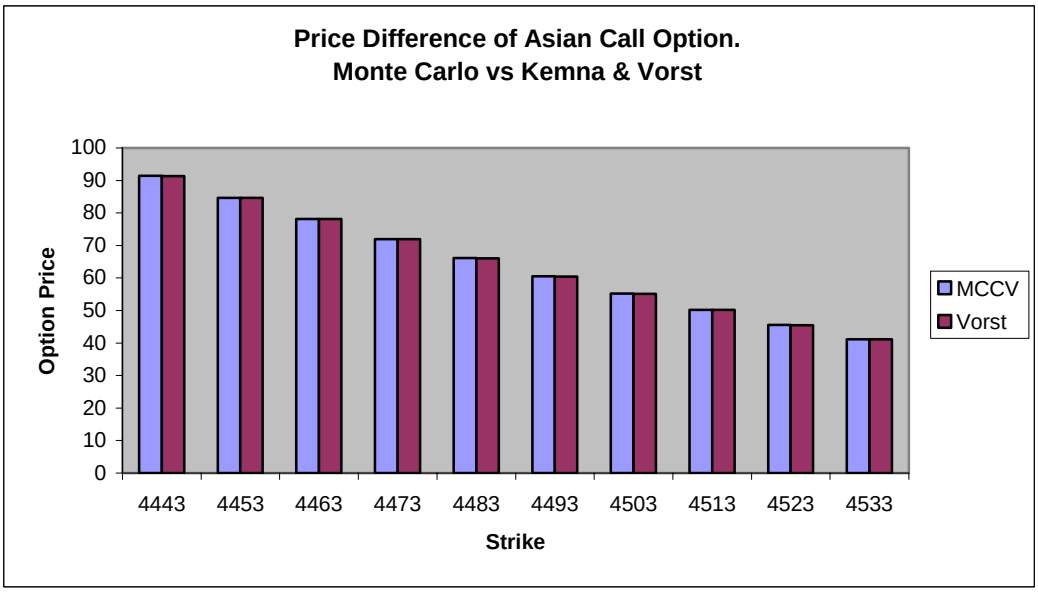

Figure 4.7 Comparison of the Asian Option Value of Monte Carlo with the Kemna and Vorst Models at a volatility of 10%.

In Table 4.9 and Figure 4.7 it can be seen the change of the option values from ITM to OTM of the MCCV and Kemna and Vorst models has decreased slightly, and eventually they will converge to the point of zero when the option will expire as worthless. I test the option values of the models at ATM with a volatility of 10%. It can be seen from the results that there is a small positive difference between the models of 60.4987 – 60.4502 = ± 0.00485

K	T	r	d	sig	MCCV	Vorst
4443	0,24658	0.04314	0.0087	0.3	183.9030	183.7797
4453	0,24658	0.04314	0.0087	0.3	178.5548	178.4311
4463	0,24658	0.04314	0.0087	0.3	173.3092	173.1852
4473	0,24658	0.04314	0.0087	0.3	168.1662	172.9129
4483	0,24658	0.04314	0.0087	0.3	163.1257	163.0014
4493	**0,24659**	**0.04314**	**0.0087**	0.3	**158.1877**	**158.0632**
4503	0,24660	0.04314	0.0087	0.3	153.3518	153.2270
4513	0,24661	0.04314	0.0087	0.3	148.6179	148.4924
4523	0,24662	0.04314	0.0087	0.3	143.9856	143.8591
4533	0,24663	0.04314	0.0087	0.3	139.4545	139.3269

Table 4.10 Comparison of the Asian Option Values of the Monte Carlo and Kemna and Vorst Models at a volatility of 30%.

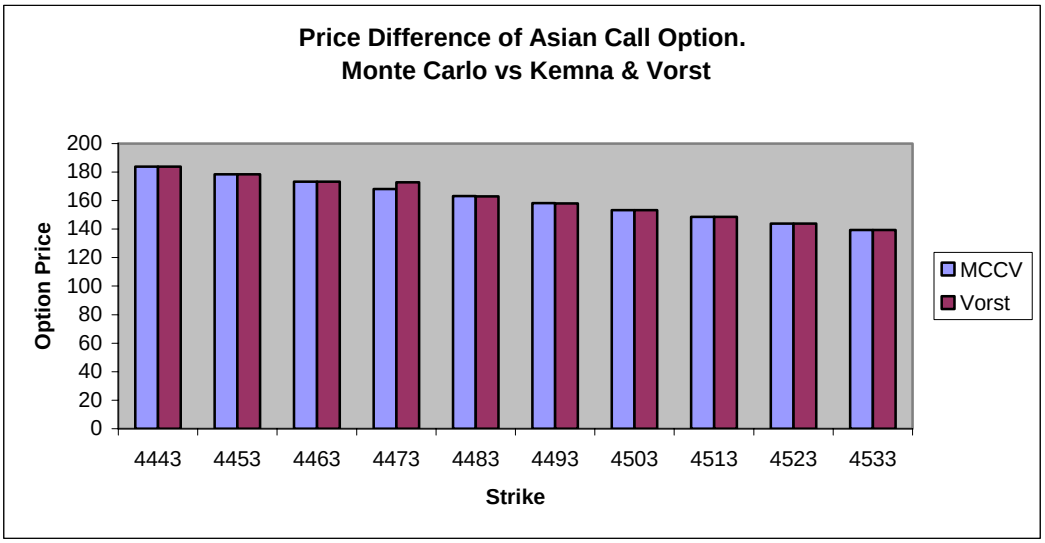

Figure 4.8 Comparison of the Asian Option Value of Monte Levy with the Kemna and Vorst Models at a volatility of 30%.

Here, I investigate an ATM option at a volatility of 30%, and keep the rest of the parameters the same, we compare it to the above test with 10% volatility. From the Table 4.10 and Figure 4.8 it can be seen that the difference of the option value between the models increases with an increase of the volatility from 10% to 30%. The change of the option value at ATM between the models is 158.1877 – 158.0632 = ±0.7068. I conclude that with an increase in the volatility there is also an

increase in the difference in the option values of the models. In this case an increase in the volatility from 10% to 30% leads to a difference in values of 0.00485 to 0.7068. It can be seen that there is a positive relationship between volatility and the difference in option values of the MCCV and Levy models.

- Maturity Criteria

Here, I test the change in the value of the ATM Asian Call option of IBM LN stock with a change in the length of time to maturity. The maturity has been given in days. The original data of IBM LN stock is provided from Bloomberg and is for a maturity of 90 days.

K	T	days	r	d	sig	MCCV	Vorst
4493	0.0027	1	0.04314	0.0087	0.21087	11.4343	11.4256
4493	0.0540	20	0.04314	0.0087	0.21087	52.3261	52.2860
4493	**0.2465**	**90**	**0.04314**	**0.0087**	**0.21087**	**115.0809**	**114.9903**
4493	0.4931	180	0.04314	0.0087	0.21087	165.8300	165.6969
4493	0.7397	270	0.04314	0.0087	0.21087	205.5590	205.3914
4493	0.9863	360	0.04314	0.0087	0.21087	239.4046	239.2069
4493	1.2328	450	0.04314	0.0087	0.21087	269.3637	269.1386
4493	1.4794	540	0.04314	0.0087	0.21087	296.4998	296.2495

Table 4.11 Comparison of the Value of Asian Option of Monte Carlo and Kemna and Vorst Models with changes in the length of time to maturity.

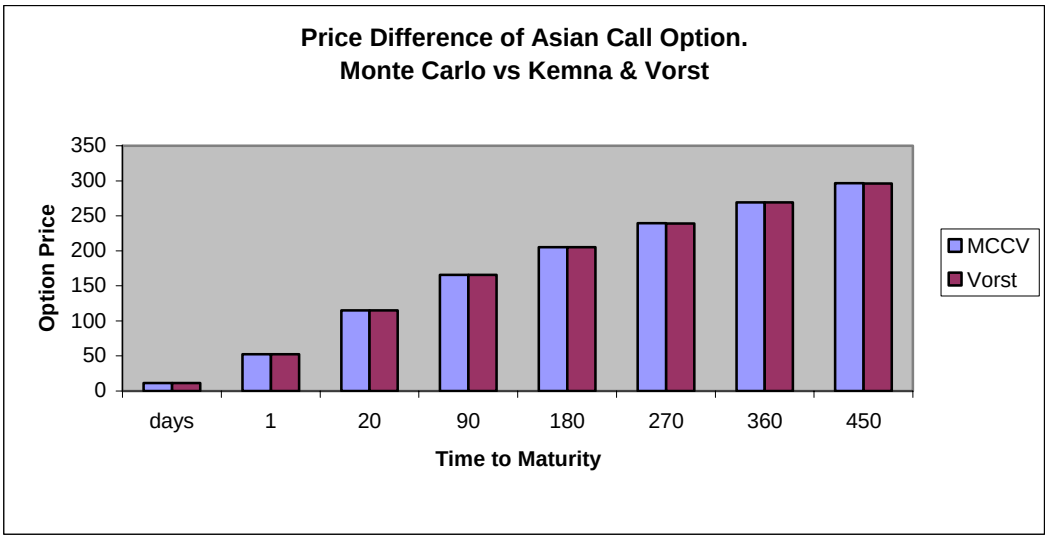

Figure 4.9 Comparison of the value of Asian Options of Monte Carlo and Kemna and Vorst Models with changes in the length of time to maturity.

Table 4.11 and Figure 4.9 show that with an increase in the time to maturity the difference in the value of the option between the MCCV and Kemna and Vorst Models has also increased. It can be seen that there is a positive relationship between the length of time to maturity and the difference in the value of the options in the models.

The above models show the following results for IBM LN stock: MCCV = 115.0809 and Levy = 114.9903, so the difference is 0.00906 in favour of the MCCV Model, and therefore I conclude that the Kemna and Vorst Model is undervalued relative to that of the Monte Carlo.

4.5 Curran

Curran (1992) developed an approximation method for valuing Asian options based on the geometric conditioning approach and he claims that his method is more accurate than other closed-form approximations. He provides a conditioning approach to compute the expected discounted risk neutral payoff. The integration is done across all possible geometric mean values, for which the probability density is obtained; hence the expected Asian payoff that is conditioned on that geometric mean value can be computed. The resulting integration provides a Black-Scholes-type formula. In order to calculate the value of the Asian option, I have used the formulas provided by Curran (1992), with the following notations as:

S - Initial asset value
K - Strike price of option
r - Risk-free interest rate
b - Cost of carry
T - Time to expiration in years
t_1 - Time to first averaging point
Δt - Time between averaging points
n - Number of averaging points
σ - Volatility of asset
N(x) - The cumulative normal distribution function

$$\mu_i = \ln(S) + (b - \frac{\sigma^2}{2})t_i \tag{4.35}$$

$$\sigma_i = \sqrt{\sigma^2[t_1 + (i-1)\Delta t]} \tag{4.36}$$

$$\sigma_{xi} = \sigma^2\{t_1 + \Delta t[(i-1) - \frac{i(i-1)}{2n}]\} \tag{4.37}$$

$$\mu = \ln(S) + (b - \frac{\sigma^2}{2})[t_1 + (n-1)\frac{\Delta t}{2}] \tag{4.38}$$

$$\sigma_x = \sqrt{\sigma^2[t_1 + \Delta t(n-1)\frac{(2n-1)}{6n}]} \tag{4.39}$$

$$\hat{K} = 2K - \frac{1}{n}\sum_{i=1}^{n} e^{(\mu i + \frac{\sigma_i(\ln(K)-\mu)}{\sigma_x^2} + \frac{\sigma_i^2 - \frac{\sigma_{xi}^2}{\sigma_x^2}}{2})} \quad (4.40)$$

4.5.1 Results

Here, I examine how the change in the value of the Asian option value responds to a change in the strike price within the bounds of a 10% and 30% volatility, and also how the value changes with the change in the time to maturity. The results are based on the Asian call option of IBM LN stock provided by Bloomberg. This is based on the Monte Carlo Model with Control Variate which is our benchmark model against Curran's model.

The following abbreviations are used:

S - Spot Value
K - Strike price
T - Time to Maturity
r - Risk Free Rate
d - Dividend
sig - Volatility
MCCV - Monte Carlo with Control Variate Model
Curran - Curran Model
OTM - out of money
ATM - at the money
ITM - in the money

- **Volatility Criteria**

S	K	T	r	d	sig	MCCV	Curran
4493	4443	0,24658	0.04314	0.0087	0.1	91.4300	91.9145
4493	4453	0,24658	0.04314	0.0087	0.1	84.6678	85.1281
4493	4463	0,24658	0.04314	0.0087	0.1	78.1862	78.6229
4493	4473	0,24658	0.04314	0.0087	0.1	71.9932	72.4073
4493	4483	0,24658	0.04314	0.0087	0.1	66.0957	66.4880
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.1	**60.4987**	**60.8701**
4493	4503	0,24660	0.04314	0.0087	0.1	55.2054	55.5568
4493	4513	0,24661	0.04314	0.0087	0.1	50.2174	50.5498
4493	4523	0,24662	0.04314	0.0087	0.1	45.5346	45.8478
4493	4533	0,24663	0.04314	0.0087	0.1	41.1553	41.4490

Table 4.12 Comparison of the Value of an Asian Option of the Monte Carlo and Curran models with a volatility of 10%.

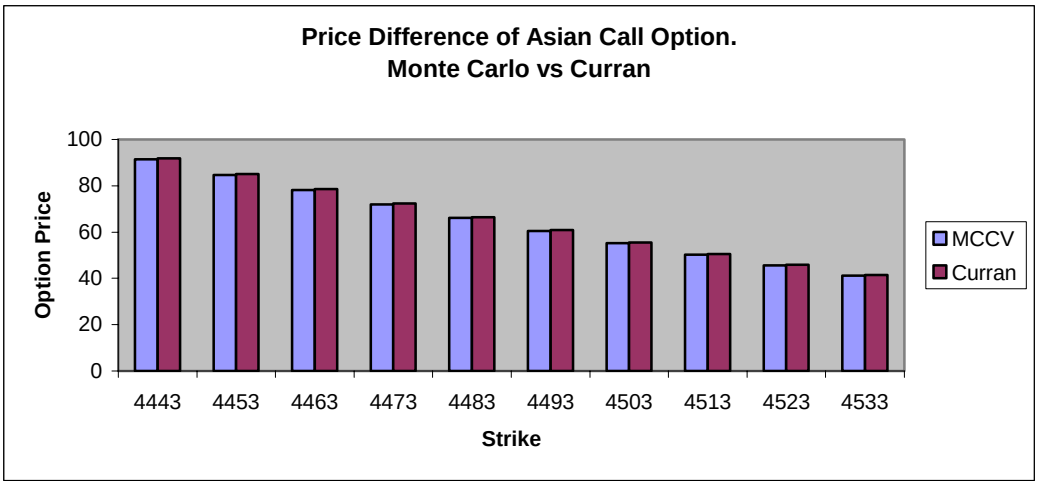

Figure 4.10 Comparison of the Value of an Asian Option of the Monte Carlo and Curran Models with a volatility of 10%.

In Table 4.12 and Figure 4.10 it can be seen that the change in the option values from ITM to OTM of the MCCV and Curran models has decreased, and eventually they will converge to the point of zero when the option will expire as worthless. Here, I test the values of the models at ATM with a volatility of 10%. It can be seen from the results that there is a negative difference between the models of 60.4987– 60.8701= ± 0.3714.

S	K	T	r	d	sig	MCCV	Curran
4493	4443	0,24658	0.04314	0.0087	0.3	183.9030	187.4938
4493	4453	0,24658	0.04314	0.0087	0.3	178.5548	182.0891
4493	4463	0,24658	0.04314	0.0087	0.3	173.3092	176.7877
4493	4473	0,24658	0.04314	0.0087	0.3	168.1662	171.5896
4493	4483	0,24658	0.04314	0.0087	0.3	163.1257	166.4948
4493	**4493**	**0,24659**	**0.04314**	**0.0087**	0.3	**158.1877**	**161.5030**
4493	4503	0,24660	0.04314	0.0087	0.3	153.3518	156.6141
4493	4513	0,24661	0.04314	0.0087	0.3	148.6179	151.8278
4493	4523	0,24662	0.04314	0.0087	0.3	143.9856	147.1436
4493	4533	0,24663	0.04314	0.0087	0.3	139.4545	142.5610

Table 4.13 Comparison of the Value of Asian Option Value using the Monte Carlo and Curran Models with a volatility of 30%.

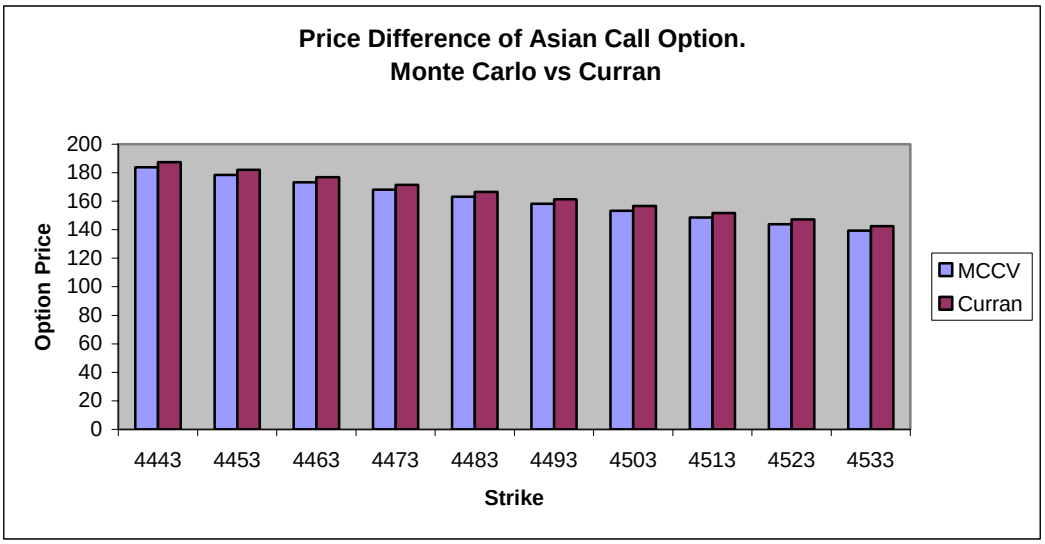

Figure 4.11 Comparison of the value of an Asian option using the Monte Carlo and Curran Models with a volatility of 30%.

Using a volatility of 30% for an ATM option, and keeping the rest of the parameters the same to compare it to the above test with 10% volatility. In Table 4.13 and Figure 4.11 it can be seen that the difference in the values of the options of the models increases with an increase of the volatility

from 10% to 30%. The difference in the change of the option value at ATM of the models is 158.1877– 161.5030 = ±3.3153. I conclude from this that with an increase of the volatility there is also an increase in difference in the option values of the models. In this case an increase in the volatility from 10% to 30% leads to the value difference from 0.3714 to 3.3153 which is significant. It can be said that there is a positive relationship between volatility and the option value difference of the MCCV and Curran models.

- Maturity Criteria

Here, I test the change in the ATM Asian Call option of IBM LN stock with the change in the length of time to maturity. The maturity has been given in days. The original data of IBM LN stock was obtained from Bloomberg and is for a maturity of 90 days.

S	K	T	days	r	d	sig	MCCV	Curran
4493	4493	0.0027	1	0.04314	0.0087	0.21087	11.4343	11.4153
4493	4493	0.0540	20	0.04314	0.0087	0.21087	52.3261	52.5901
4493	**4493**	**0.2465**	**90**	**0.04314**	**0.0087**	**0.21087**	**115.0809**	**116.745**
4493	4493	0.4931	180	0.04314	0.0087	0.21087	165.8300	169.4007
4493	4493	0.7397	270	0.04314	0.0087	0.21087	205.5590	211.0747
4493	4493	0.9863	360	0.04314	0.0087	0.21087	239.4046	246.8816
4493	4493	1.2328	450	0.04314	0.0087	0.21087	269.3637	278.8077
4493	4493	1.4794	540	0.04314	0.0087	0.21087	296.4998	307.9131

Table 4.14 Comparison of the Values of an Asian Option of Monte Carlo and Curran Models with changes in the length of time to maturity.

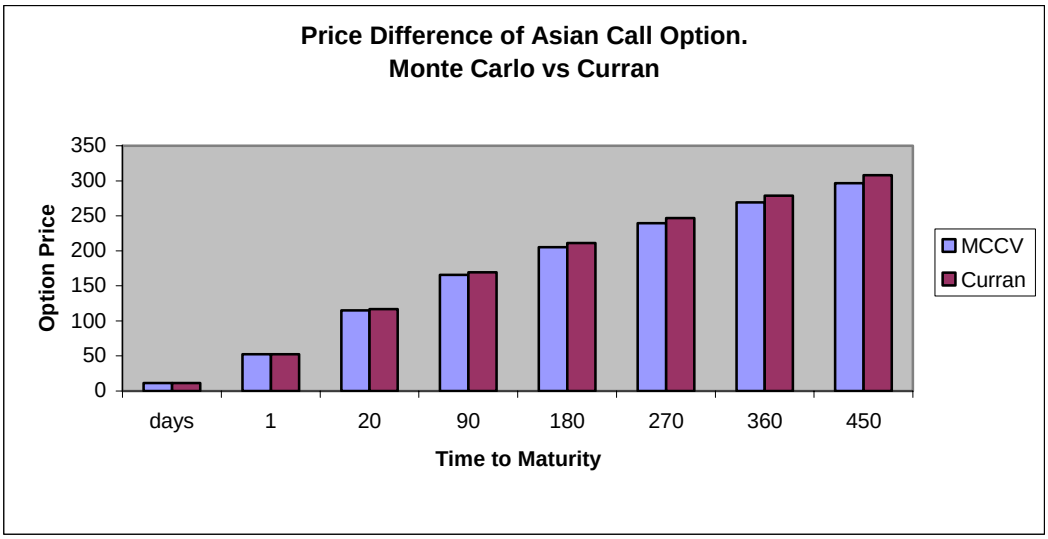

Figure 4.12 Comparison of the Value of an Asian Option of Monte Carlo and Curran Models with changes in length of time to maturity.

Table 4.14 and Figure 4.12 show that with an increase in length of the time to maturity the difference in the value of the option using the MCCV and Curran models has also increased. Therefore, I conclude that there is a positive relationship between the maturity and the value of the option in the models.

The above models show the following results for IBM LN stock: MCCV = 115.0809 and Curran = 116.7449, so the difference is 1.664 in favour of Curran, so I conclude that Curran Model is overvalued relative to the that of the Monte Carlo.

5 A Comparison of the Models

In this chapter I compare the option values computed from the different models comment on the findings and secondly to verify the accuracy of the application models with external models from Bloomberg, Global Derivatives and Derivatives Models.

The four models being:

1. Standard Vanilla Model
   - Back and Scholes

2. Arithmetic Model
   - Turnbull and Wakeman
   - Levy

3. Geometric Model
   - Kemna and Vorst
   - Curran

4. Arithmetic Model with Geometric Variate
   - Monte Carlo

5.1 Assessment of the Application Models

In my analysis I used the Asian option of IBM stock and kept the original contract specifications of the option but have used different strike prices to calculate its value using different models.

K	T	r	d	sig	MCCV	TW	Levy	Vorst	Curran	B&S
4443	0,24658	0.04314	0.0087	0.21087	142.1385	144.7019	144.7019	142.0492	144.0004	232.4176
4453	0,24658	0.04314	0.0087	0.21087	136.4415	138.9472	138.9472	136.3517	138.2625	226.9793
4463	0,24658	0.04314	0.0087	0.21087	130.8887	133.3363	133.3363	130.7986	132.6696	221.6250
4473	0,24658	0.04314	0.0087	0.21087	125.4808	127.8699	127.8699	125.3904	127.2222	216.3549
4483	0,24658	0.04314	0.0087	0.21087	120.2180	122.5482	122.5482	120.1275	121.9207	211.1688
4493	**0,24659**	**0.0431**	**0.0087**	**0.21087**	**115.1004**	**117.3715**	**117.3715**	**115.0099**	**116.7651**	**206.0667**
4503	0,24660	0.04314	0.0087	0.21087	110.1282	112.3398	112.3398	110.0377	111.7554	201.0485
4513	0,24661	0.04314	0.0087	0.21087	105.3009	107.4528	107.4528	105.2100	106.8913	196.1141
4523	0,24662	0.04314	0.0087	0.21087	100.6183	102.7095	102.7095	100.5264	102.1723	191.2630
4533	0,24663	0.04314	0.0087	0.21087	96.0797	98.1100	98.1100	95.9867	97.5973	186.4950

Table 5.1 Value Difference of European Call Options computed from the Models used in the Application.

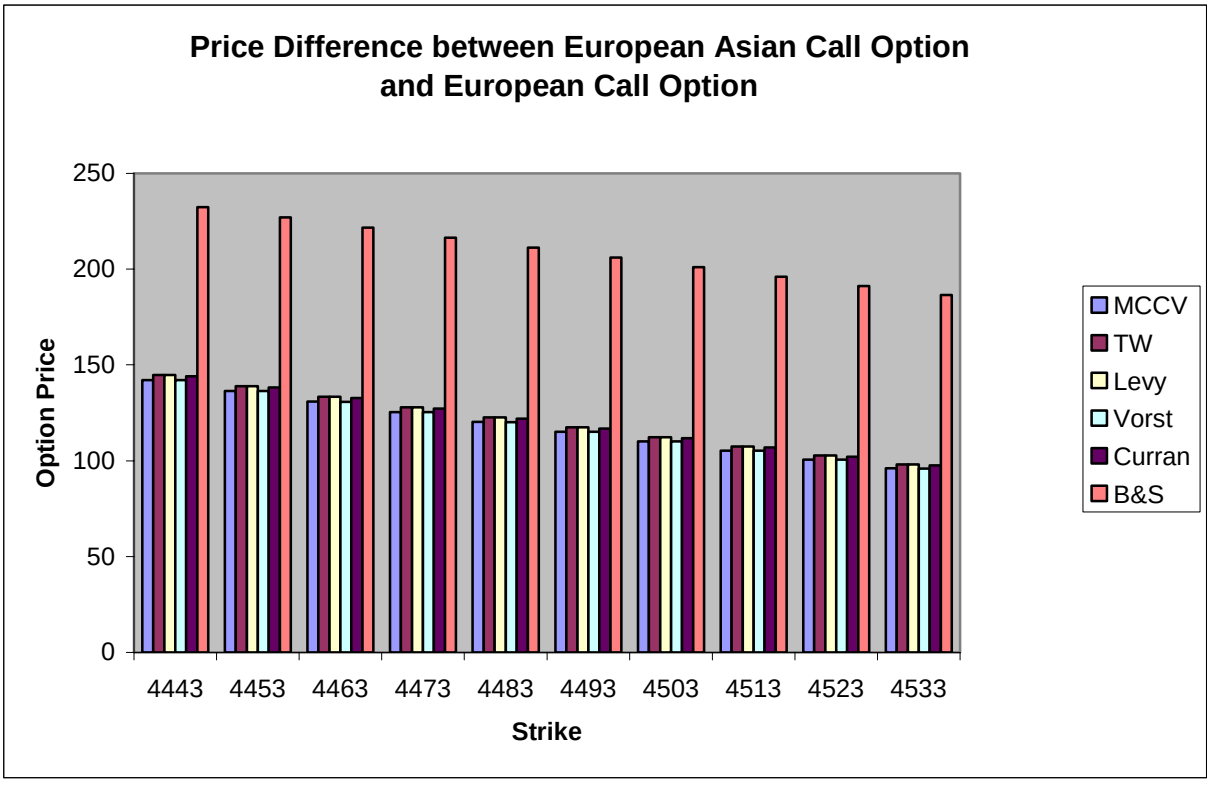

Figure 5.1 Value Difference of European Call Options computed from the Models used in the Application.

Table 5.1 and Figure 5.1 show the comparison of the option values of the models. I have used different strikes of ITM, ATM and OTM for the IBM LN option, as I wanted to trace the option value difference at different level of the strike price, The results show that the option value change is highest at ATM and lowest at OTM, which make sense because the OTM options converge to the point around zero. It can be seen that there is an obvious value difference between European Call Option and European Asian Call Option at all different levels of the strike price. The aim of this analysis is to discover how the value of Asian options behaves at different strike levels and to compare the calculations of the different exotic type models. First, it can be seen that the Arithmetic models (TW, Levy) have very close option values at all levels. Secondly, it can be seen that the Exotic Geometric models (Vorst and Curran) always produce small differences in the calculating of the option value. The analysis focuses in calculating the option value mainly at ATM option value level where it is also the quoted value by Bloomberg. The average value of the Arithmetic models is around £117.37, the Geometric average value is in the bounds of ££115.00 and £116.76 and the Arithmetic Model with Control Variate (Monte Carlo) is around £155.1. I have used the Monte Carlo Model as a benchmark and the closest value to it is provided by the Geometric model of Vorst which gave a value of £115.0099. However, this does not mean that the Monte Carlo Model produces the correct market value for this exotic Asian option.

I find that the Monte Carlo Model has an advantage over the other models when there is a varying strike price, since it outperforms all the other models when the option is in and out of the money. To analyse the influence of the volatility on the models used in my application I use the ATM European Asian Call Option of IBM LN stock with a volatility range of between 1% and 50%.

S	K	T	r	d	sig	MCCV	TW	Levy	Vorst	Curran	B&S
4493	4493	0,24659	0.04314	0.0087	0.0050	18.9355	18.9322	18.9322	18.9165	19.0049	37.9134
4493	4493	0,24659	0.04314	0.0087	0.0100	19.3172	19.3211	19.3211	19.2983	19.3832	38.3080
4493	4493	0,24659	0.04314	0.0087	0.0500	35.9319	36.0785	36.0785	35.9019	36.0367	65.7372
4493	4493	0,24659	0.04314	0.0087	0.1000	60.4987	61.0330	61.0330	60.4502	60.8701	108.6778
4493	4493	0,24659	0.04314	0.0087	0.1500	85.1946	86.3590	86.3590	85.1273	86.0353	152.4476
4493	4493	0,24659	0.04314	0.0087	0.2000	109.7417	111.7857	111.7857	109.6553	111.2387	196.4109
4493	4493	0,24659	0.04314	0.0087	0.2500	134.0802	137.2599	137.2599	133.9748	136.4068	240.4331
4493	4493	0,24659	0.04314	0.0087	0.3000	158.1877	162.7659	162.7659	158.0632	161.5030	284.4642
4493	4493	0,24659	0.04314	0.0087	0.3500	182.0523	188.2982	188.2982	181.9087	186.5010	328.4791
4493	4493	0,24659	0.04314	0.0087	0.4000	205.6664	213.8553	213.8553	205.5035	211.3772	372.4617
4493	4493	0,24659	0.04314	0.0087	0.4500	229.0240	239.4377	239.4377	228.8417	236.1094	416.4002
4493	4493	0,24659	0.04314	0.0087	0.5000	252.1202	265.0467	265.0467	251.9185	260.6749	460.2850

Table 5.2 Value Difference of European Call Options with a different range of volatility computed from the models used in the Application.

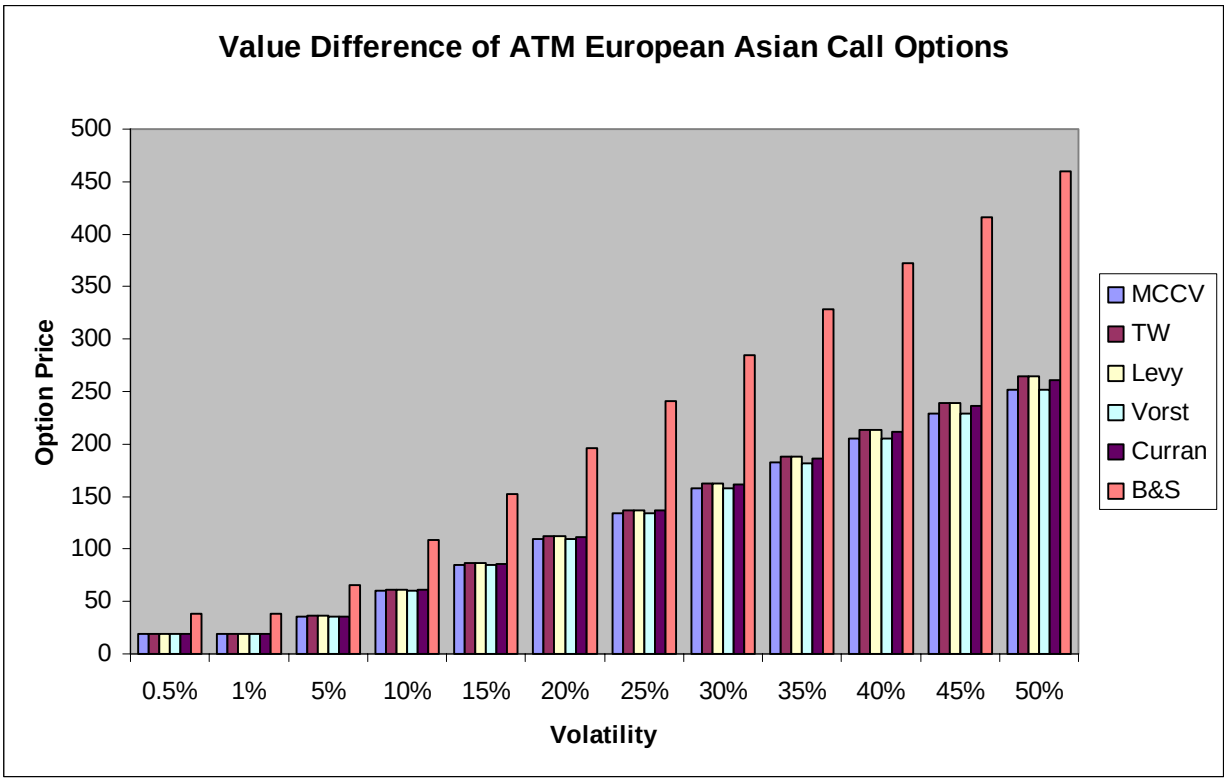

Figure 5.2 Value difference of European Call Options with a different range of volatility computed from the models used in the Application.

Table 5.2 and Figure 5.2 show that the option values of the models are almost equal at a volatility of 1%, but as the volatility starts increasing with every 5% step the option values produced by the models vary to an ever greater percent. I conclude that a higher level of volatility increases the difference in the option value of the models, and that the Kemna and Vorst model is the least sensitive to changes in volatility.

S	K	T	days	r	d	sig	MCCV	TW	Levy	Vorst	Curran
4493	4493	0.0027	1	0.04314	0.0087	0.21087	11.4343	11.4501	11.4492	11.4256	11.4153
4493	4493	0.0540	20	0.04314	0.0087	0.21087	52.3261	52.7680	52.7680	52.2860	52.5901
4493	4493	0.2465	90	0.04314	0.0087	0.21087	115.0809	117.3511	117.3511	114.9903	116.7449
4493	4493	0.4931	180	0.04314	0.0087	0.21087	165.8300	170.6582	170.6582	165.6969	169.4007
4493	4493	0.7397	270	0.04314	0.0087	0.21087	205.5590	213.0963	213.0963	205.3914	211.0747
4493	**4493**	**0.9863**	360	**0.04314**	**0.0087**	**0.21087**	**239.4046**	**249.7632**	**249.7632**	**239.2069**	**246.8816**
4493	4493	1.2328	450	0.04314	0.0087	0.21087	269.3637	282.6315	282.6315	269.1386	278.8077
4493	4493	1.4794	540	0.04314	0.0087	0.21087	296.4998	312.7519	312.7519	296.2495	307.9131

Figure 5.3 Value Difference of European Call Options with a different range of maturities computed from the models used in the Application.

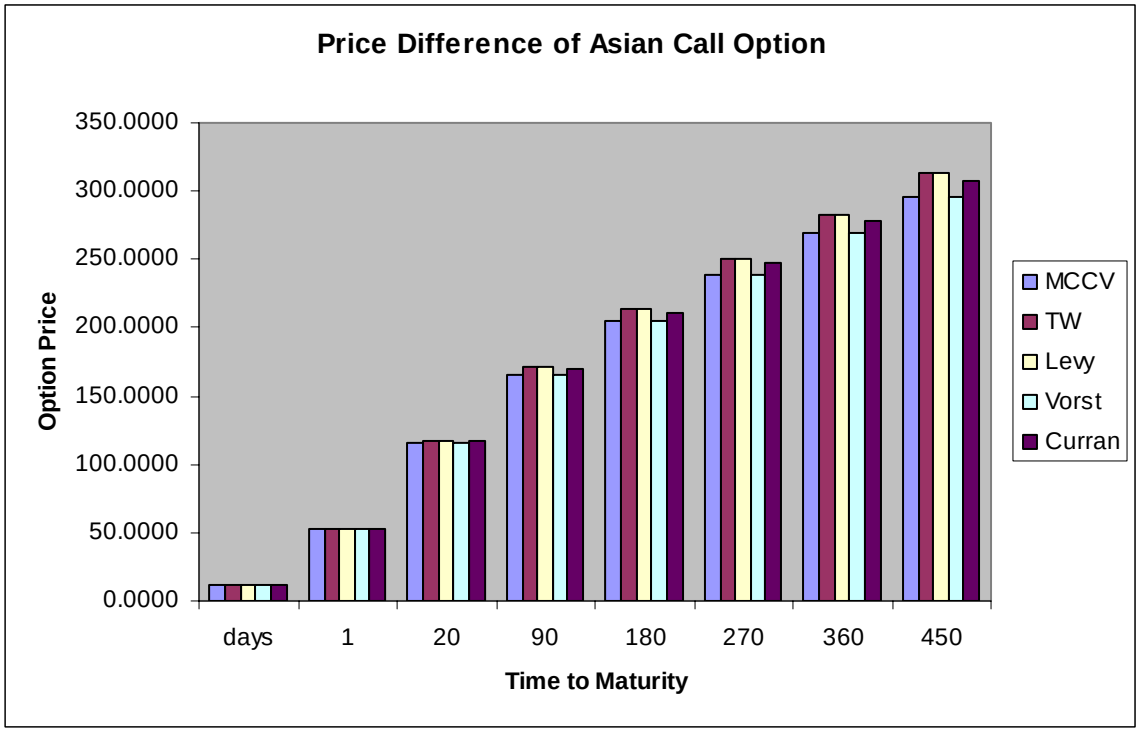

Table 5.3 Value Difference of European Call Options with a different range of maturities computed from the models used in the Application.

In Table 5.2 and Figure 5.2 it can be seen that the option values of the models are almost equal at 1 day to maturity, but as the maturity increases the option values in the models vary. I conclude that the higher the level of maturity the greater the difference in the option value of the models, and that the Monte Carlo model is the least sensitive to changes to maturity.

5.2 Verification of Application Models

Here, I investigate the accuracy of the models (see Appendix B for references).

The option specifics are as:

S	K	T	r	d	sig
4493	4493	0,24659	0.04314	0.0087	0.21087

Table 5.4 IBM LN contract specifications.

5.2.1 Standard Vanilla Option Model

Tested only for European call options.

Models	Option Value
Application(B&S) – Black and Scholes	**206.0667**
Bloomberg (B&S))- Black and Scholes	205.9299
Bloomberg (EDD) - Enhanced Discrete Dividends	206.3707
Bloomberg (Tri) - Trinomial	206.1733

Table 5.5 European Call Option on IBM LN stock.

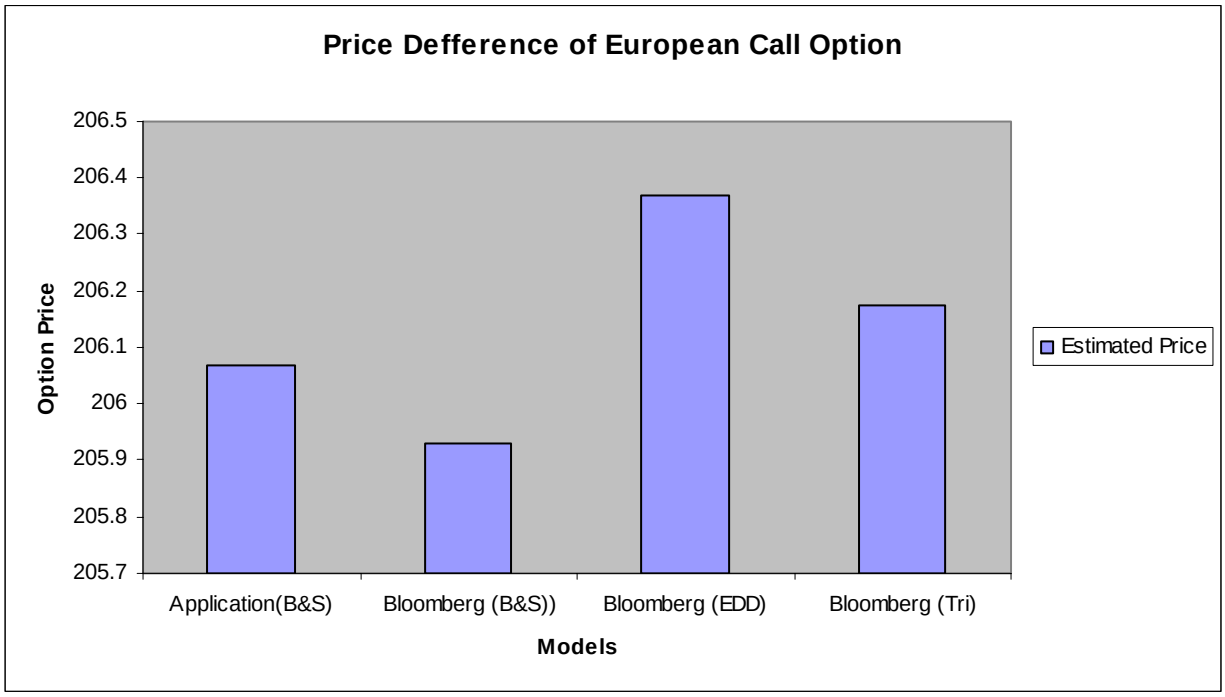

Figure 5.4 European Call Option for IBM LN stock.

Table 5.5 and Figure 5.4 show that Bloomberg quotes three different values for a standard option as there is a small difference in calculating even a simple standard option using different models. Comparing my application's Standard Vanilla Model to that by Bloomberg the values are similar at £206.0667 to Bloomberg's trinomial £206.1733. Therefore I conclude that my application of the Standard Vanilla model is reasonably accurate.

5.2.2 Exotic Asian Option Models

Although I have used three different sources to test the accuracy of the models used in my application I have no proof of their accuracy of the three sources; Bloomberg, Global Derivatives and Derivatives Models, though Bloomberg, as used by so many market practitioners would suffer great reputational damage if it were frequently wrong.

Models	Option Value
MCCV	**115.1004**
Bloomberg Arithmetic	115.7449
Bloomberg Geometric	113.3578
TW	**117.3715**
Global Derivatives (TW)	117.1504381
Derivatives Models (TW)	117.37146
Levy	**117.3715**
Global Derivatives (Levy)	117.1503322
Derivatives Models (Levy)	117.37146
Vorst	**115.0099**
Global Derivatives (Vorst)	115.0087
Derivatives Models (Vorst)	115.0099
Curran	**116.7651**

Table 5.6 European Asian Call Option of IBM LN stock.

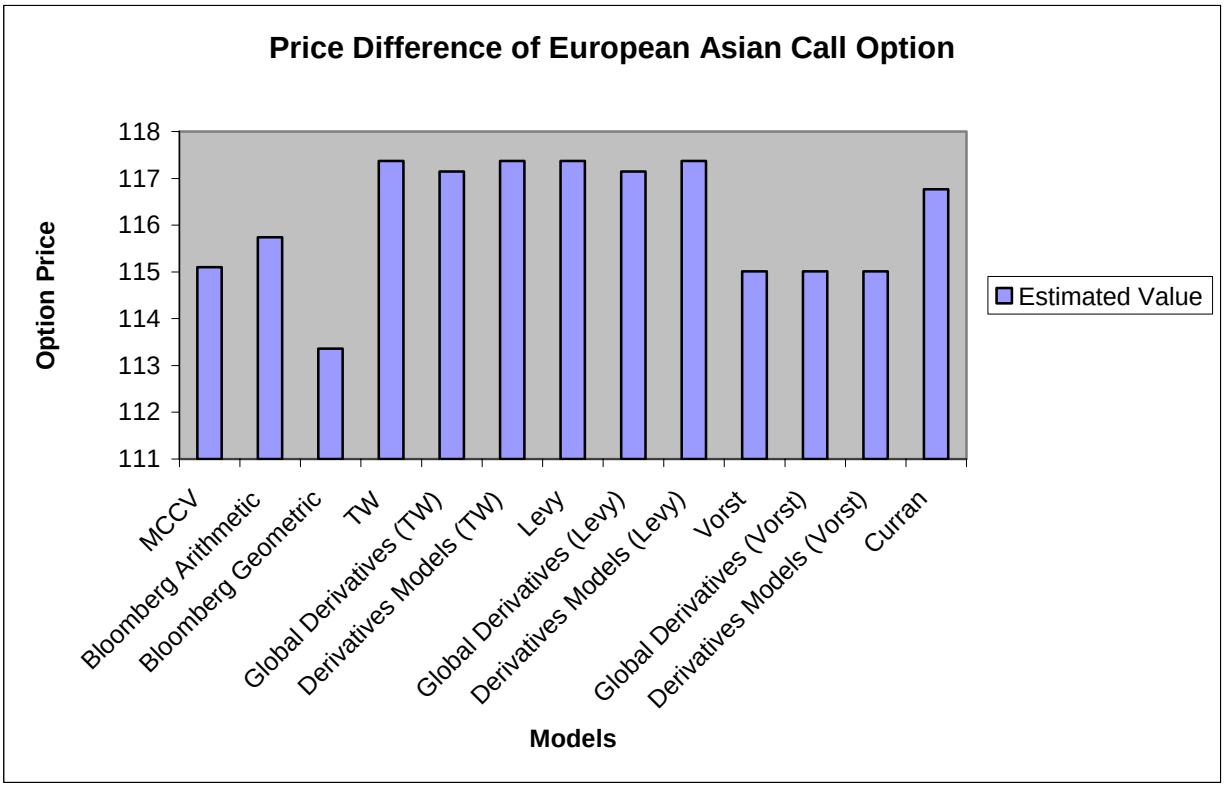

Figure 5.5 European Asian Call Option of IBM LN stock.

In Table 5.6 and Figure 5.5, I distinguish three different 'sets' of option values, at points around £117, £115 and the rest. £117 is the value according to the Turnbull and Wakeman Model. The Levy Model is also around £117. The Kemna and Vorst models give a value of £115. I believe that the TW, Levy and Vorst models are fairly accurate. Curran's option value is £116 and has not been confirmed by any source. The Monte Carlo Model's option value is £115 which is very close to that of the Vorst model. Bloomberg quotes two different values for geometric and arithmetic options, £113 and £115.7 respectively. Bloomberg's Geometric option value has not been confirmed by any of the models I have used, but the Bloomberg's arithmetic value of £115 has been confirmed by the Monte Carlo Model and the Kemna and Vorst model. The dilemma here is that the Geometric Kemna and Vorst Model corresponds to the Bloomberg's arithmetic value but the geometric values of Bloomberg are substantially different from the results of the other geometric models. According to the Bloomberg their Asian Option calculator is based on the Kemna and Vorst Model. I conclude that there will always be a difference in the valuing of options by the models, but only a small one.

6 Hedge sensitivities

As previously mentioned Asian hedge sensitivities are shown here for information purposes only as the data at the time of writing this thesis has not been validated. The 'Greek' parameters of the Asian options have different characterises relative to those of vanilla options. The hedge parameters differ when applied to Asian options due to the effect of the range of the observed values (as opposed to a single observed value). Both the total number of observations and the observational frequency serve to determine how 'the Greeks' operate. The parameters are also influenced by the number of values that have already been set relative to the number of values that have still to be set.

Hedging these options depends to some extend upon the time to maturity, in as much as the hedge parameters vary according to whether t is prior to the pricing period or within it. When t is prior to the averaging period, the option is obviously dependent upon a full set of forward values, whereas during the value period, the option is dependent upon an ever-decreasing set of forward values, as the average-contributing values are established.

The Hedge sensitivity measures are as follows:

1. Delta (δ)
   The delta of the option measures how fast an option prior change with the value of their underlying asset.
2. Gamma (γ)
   The gamma of the option measures how fast the option's delta changes with the value of its underlying asset.
3. Theta (θ)
   The theta of the option measures the sensitivity of its value with respect to time to maturity.
4. Vega (v)
   The vega of the option measures how fast an option value changes with its volatility
5. Rho (ρ)
   The rho of the option measures the sensitivity of the option's value with respect to the fluctuation of interest rate.

Nowadays all trading and risk management by financial institutions using derivatives utilise these measures.

7 Design and Implementation of the Application

Here, I describe the procedures and methods used in the design and the implementation of the Asian Option Calculator application. (For referenced diagrams see Appendix B).

7.1 The Application's Design

The design stage of the application consists of two phases; Interface design and class diagram design.

7.1.1 Interface Design

The interface design of the application has been done using the SmartDraw 7 software application, and helped me to visualise the look of the end product.

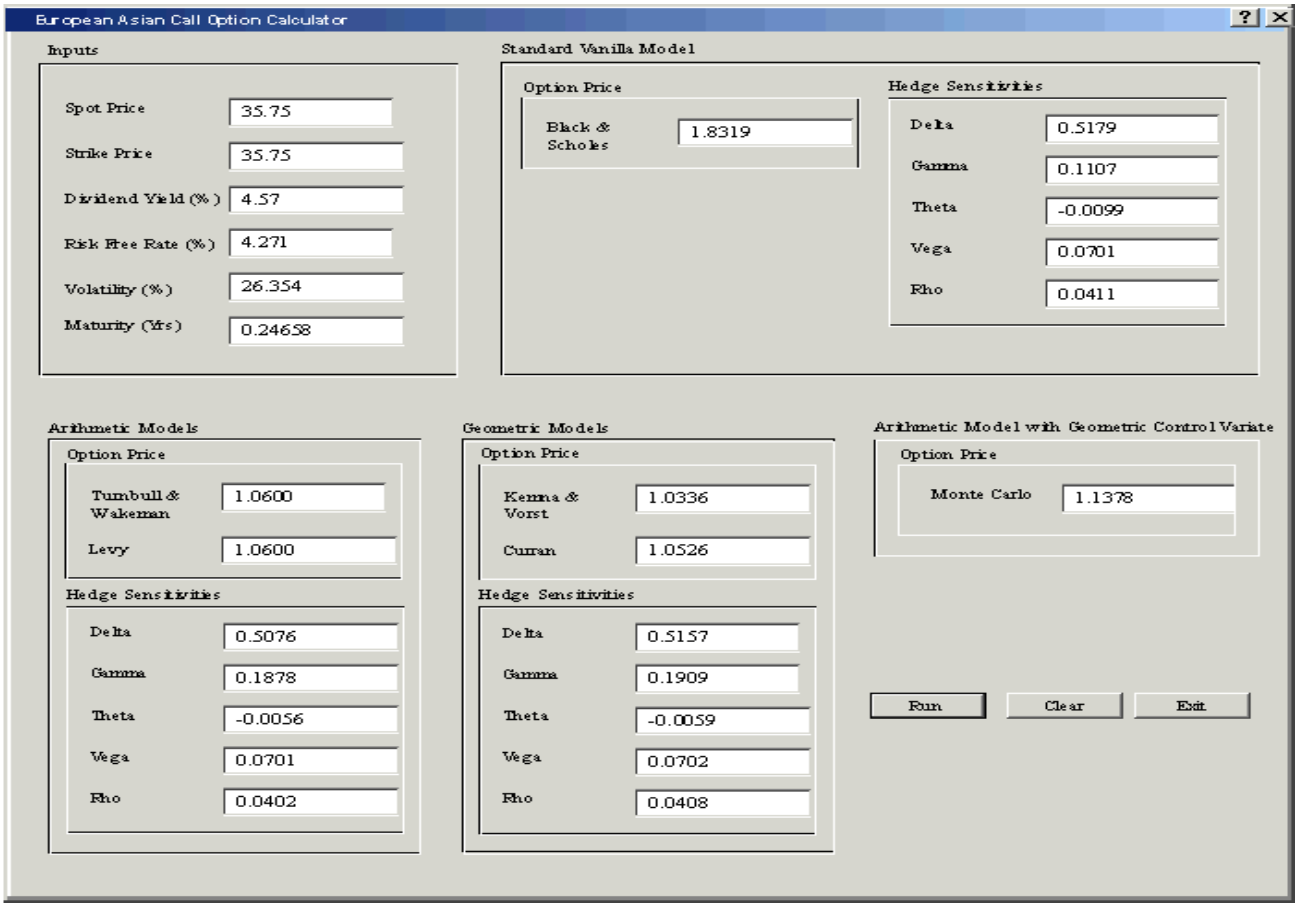

Figure 7.1 Prototype of the Application.

7.1.2 Class Diagram Design

The Class Diagram design aided me in implementing the earlier discussed models into Java object -oriented language code. I also used Oracle JDeveloper and Borland JBuilder application tools to design the classes into UML (Unified Modeling Language) diagrams.

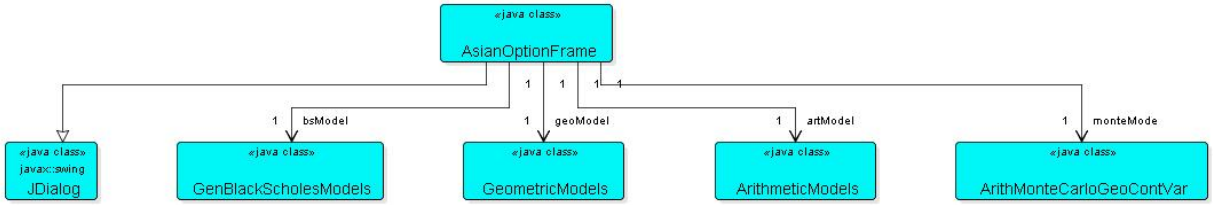

Figure 7.2 Compact UML view of the Asian Option Calculator application.

7.2 Implementation of the Application

As can be seen from the prototype interface that the model has been divided into four parts; Arithmetic models, Geometric models, Arithmetic model with Geometric Control Variate and Standard Vanilla model. In this section I discuss only the model's implementation part of the classes, there is a detailed class diagram of the interface design called AsianOptionFrame (Appendix B).

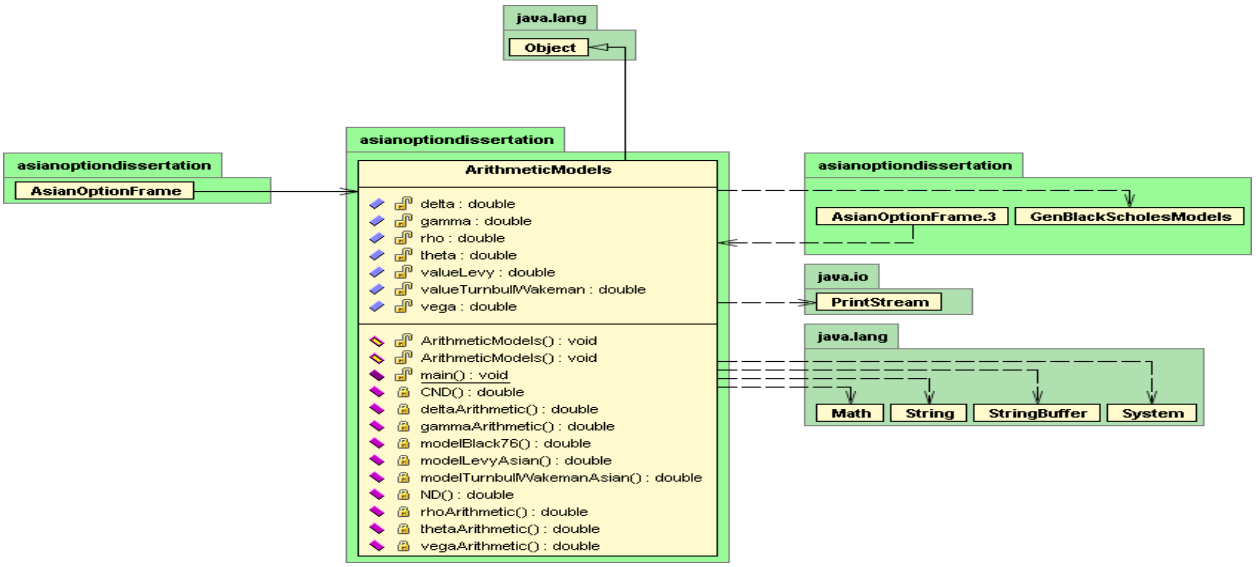

Figure 7.3 Arithmetic Models class diagram showing implementation of the arithmetic models which I have already discussed including the Turnbull and Wakeman and Levy models.

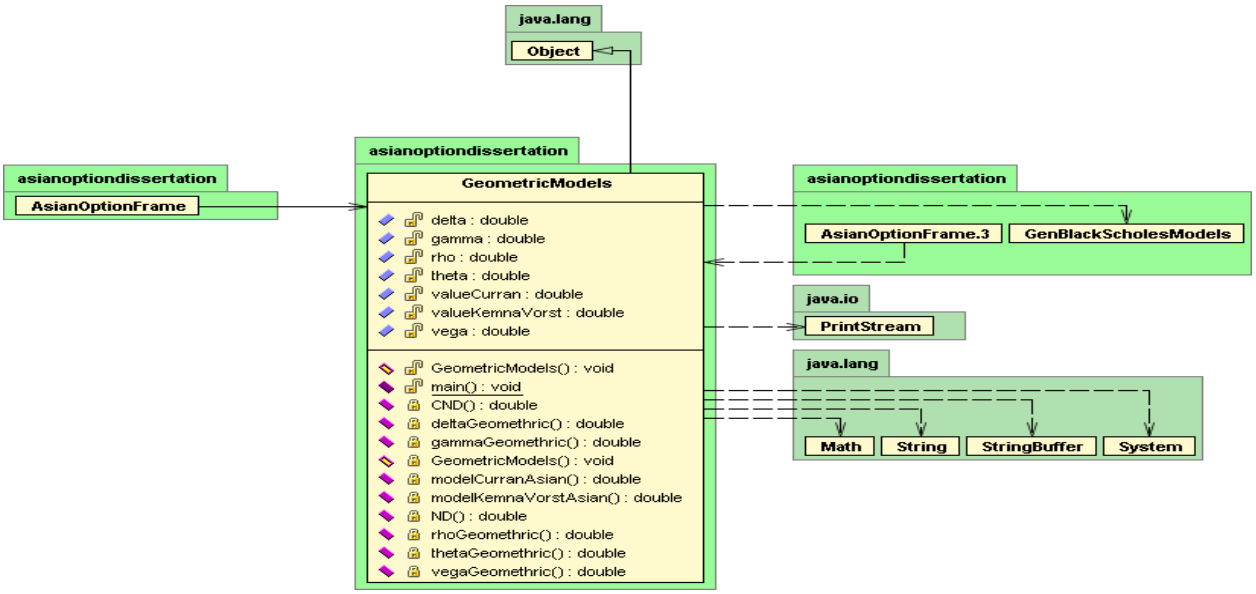

Figure 7.4 Geometric Models Class Diagram and shows my implementation of the Kemna and Vorst Model and the Curran Model.

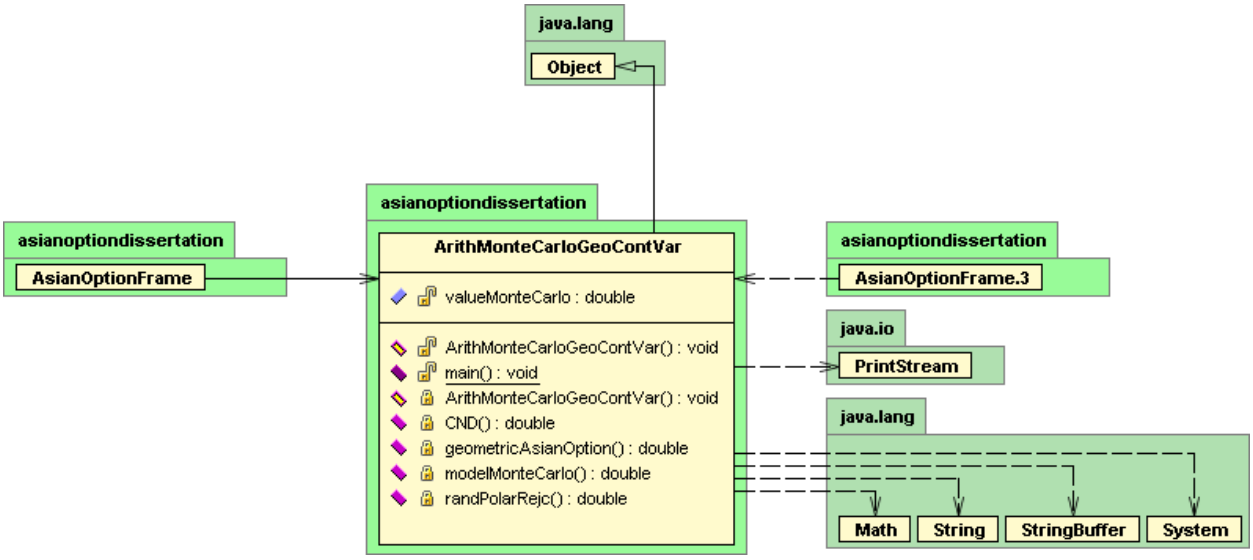

Figure 7.5 Arithmetic Model with Control Variate Class Diagram and shows my implementation of the Monte Carlo Model, which was the benchmark.

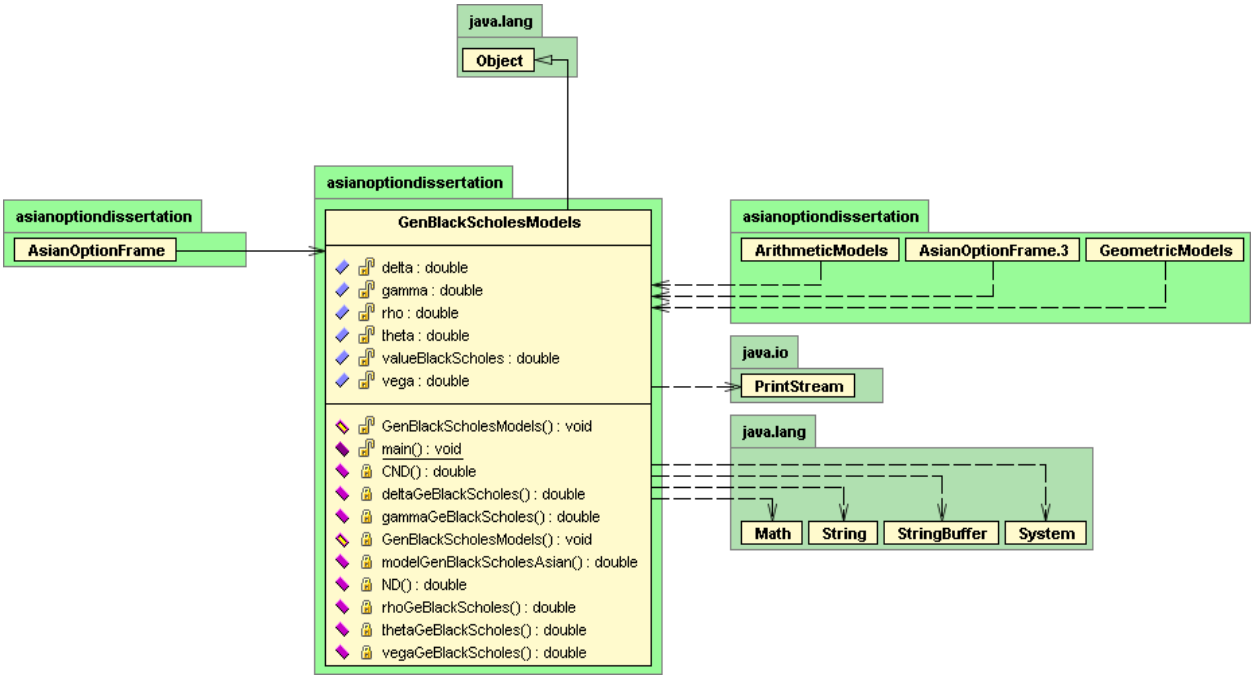

Figure 7.6 Standard Vanilla Model Class Diagram and shows my implementation of the Back and Scholes Model.

The implementation of the application models has been developed in java code. For visual reasons I have shown the implementation with the class diagrams, which allows for a better understanding of the class interaction. The application tools used for the development of the java code are IntelliJ IDEA 5.0 and Borland JBuilder 2005. I have used IntelliJ for development of the core model classes as it has better re-factoring features and also used JBuilder for designing the Swing interface part of the application and I developed the executable jar file using Exe4j software application.

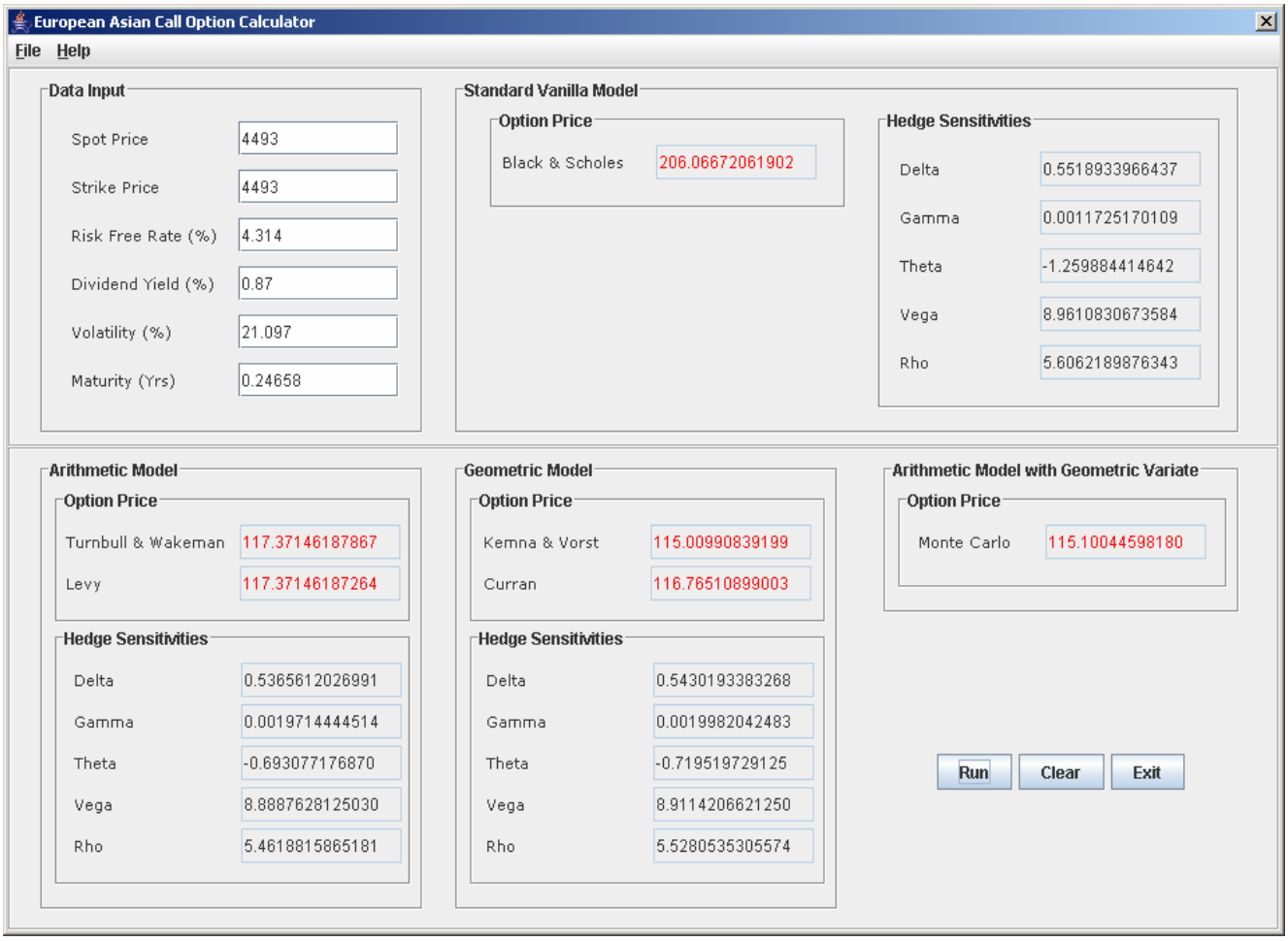

Figure 7.7 Java Swing application of the European Asian Call Option Calculator. It shows the interface of the application that I used to input data on IBM LN taken from Bloomberg.

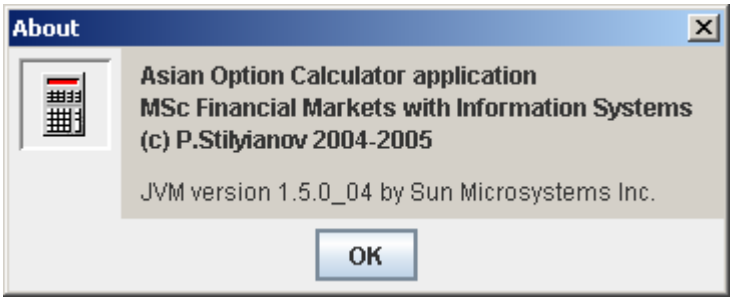

Figure 7.8 Copyright© screen

** JVM version 1.5.0 atleast required - from Sun Systems web site http://www.java.com/en/index.jsp

8 Conclusion

In my analysis I discovered that no one model outperformed the others in all circumstances and therefore I not able to recommend one particular model for the valuing of Asian Options. My analysis leads me to highlight the great importance of volatility in the valuing of options as this can not be observed in the market and therefore it is more likely to be a source of error than the other parameters. I find that the Kemna and Vorst Model is the least sensitive to volatility and also that it has the least sensitivity to maturity. As far as the parameter strike price is concerned. for a varying strike, the best model to use is the Monte Carlo Model, since it outperforms all the other models when the option is deep in and out of the money.

Although I chose the Monte Carlo Control Variate Simulation as a benchmark, its use in the real world may not be practical as running the required large number of simulations and large number of time steps that it requires is very time consuming.

I believe that the simplified analytic approximation model described by Levy, which depends on only the first two moments of the approximated distribution, is sufficient and accurate enough to provide reliable valuations in the context of many scenarios. I found that nothing was gained by the addition of the higher terms as presented in the model by Turnbull and Wakeman. This result is particularly interesting given the high levels of volatility and the long time to maturity against which the models were tested.

The inclusion of the Plain Vanilla European Option in the results serves to demonstrate how Asian options can be more appealing to an investor who is very adverse to volatility exposure. The values of the Asian options were noticeably cheaper than the European equivalents in most scenarios. This result though is already a well-established fact.

Bibliography

Black, F. & Scholes, M. (1973), The Pricing Of Options And Corporate Liabilities, Journal of Political Economy, 81(3), 637-654.

Clewlow, L., & Strickland, C. (2004), Implementing Derivative Models, Wiley Series In Financial Engineering.

Curran, M. (1992), Beyond Average Intelligence, Risk 5(10), 60.[9] Curran, M. (1994), Valuing Asian And Portfolio Options By Conditioning On The Geometric Mean, Management Science, 40(12), 1705-1711.

Kemna, A. G. Z. & Vorst, A. C. F. (1990), A Pricing Method for Options Based on Average Asset Values, Journal of Banking and Finance, 14, 113-129.

Levy, E. (1992), Pricing European Average Rate Currency Options, Journal of International Money and Finance, 14, 474-491.

Levy, E. & Turnbull, S. (1992), Average Intelligence, Risk 5(2), 53-58.

Milevsky, M. & Posner, S. (1998), Asian Options, The Sum Of Lognormals, And The Reciprocal Gamma Distribution, Journal of Financial and Quantitative Analysis, 3(3).

Turnbull, S. & Wakeman, L. (1991), A Quick Algorithm For Pricing European Average Options, The Journal of Financial and Quantitative Analysis, 26(3), 377-389.

Vorst, A. (1992), Prices And Hedge Ratios Of Average Exchange Rate Options, International Review Of Financial Analysis, 1 179-193.

Web Sites:
Derivatives Models - http://www.derivativesmodels.com/
Global Derivatives - http://www.global-derivatives.com/

Forums:
http://www.global-derivatives.com/
http://www.wilmott.com/

Appendices

Appendix A

Data and Models

Figure 1.1 Asian Option Valuations - Arithmetic

Figure 1.2 Asian Option Valuations - Geometric

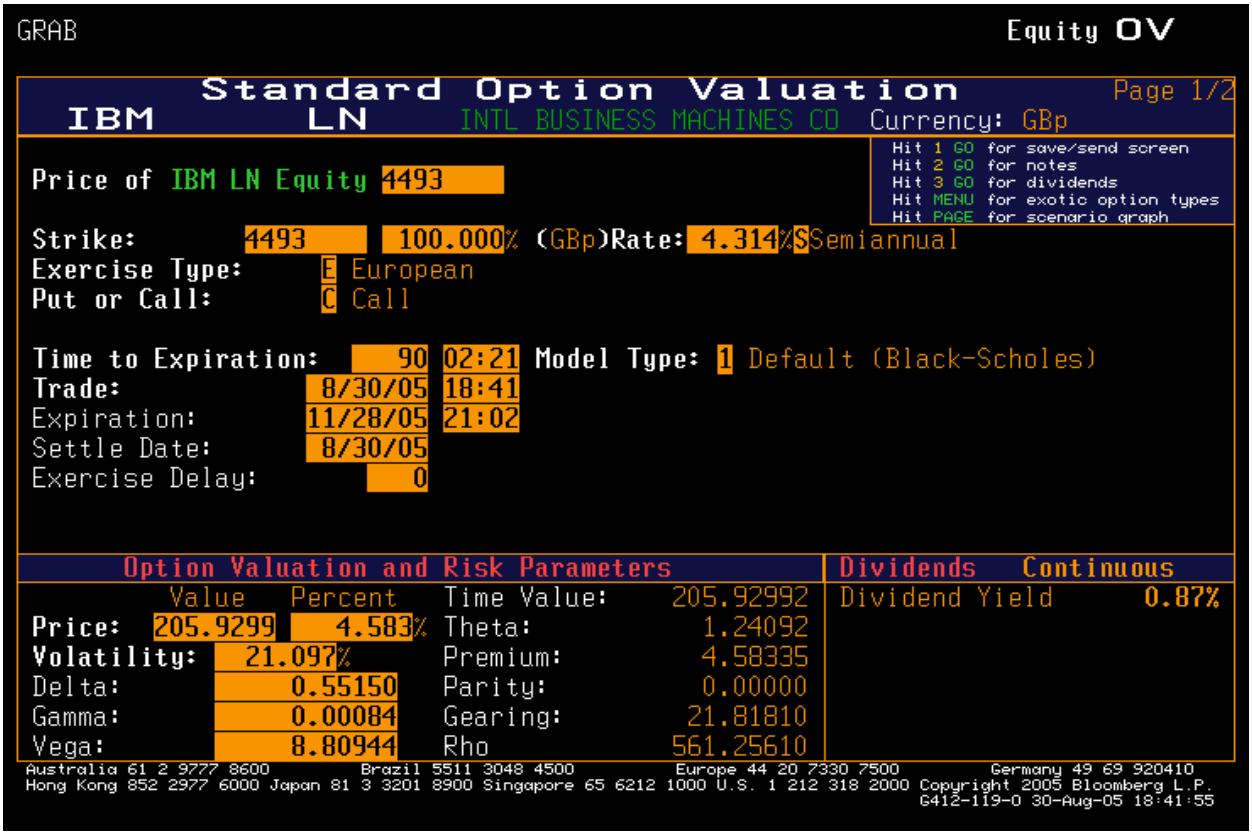

Figure 1.3 Standard Option Valuations - Black and Scholes

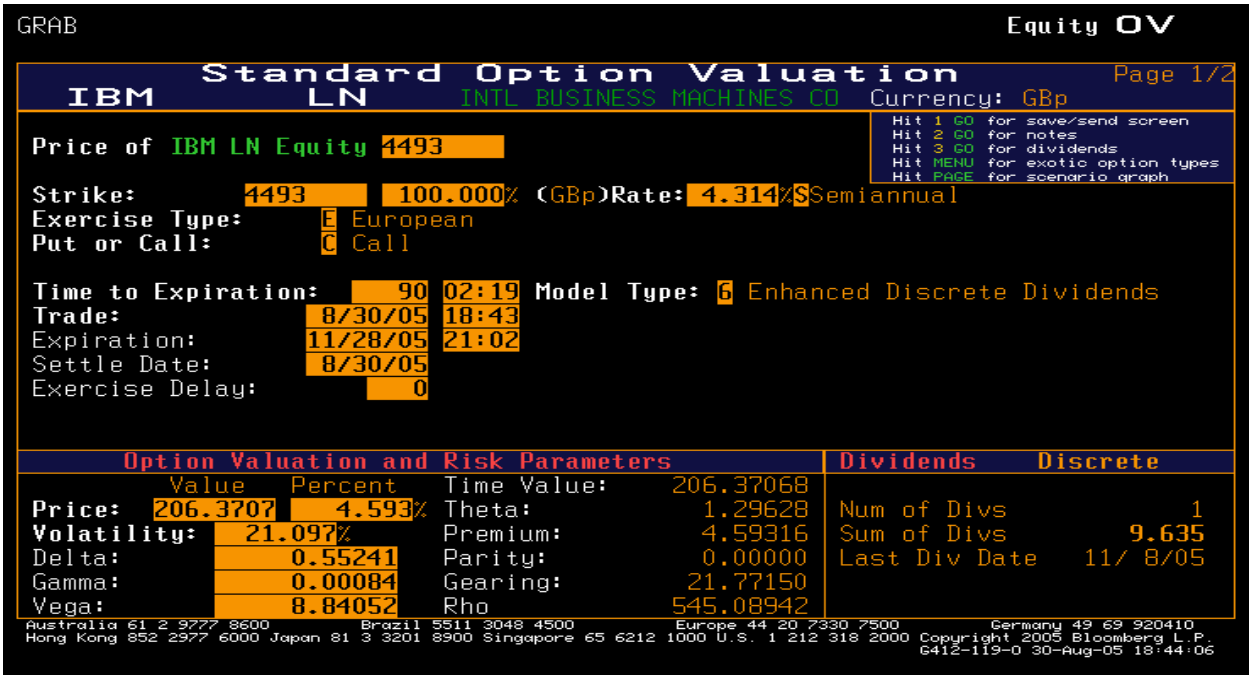

Figure 1.4 Standard Option Valuations - Enhanced Discrete Dividents

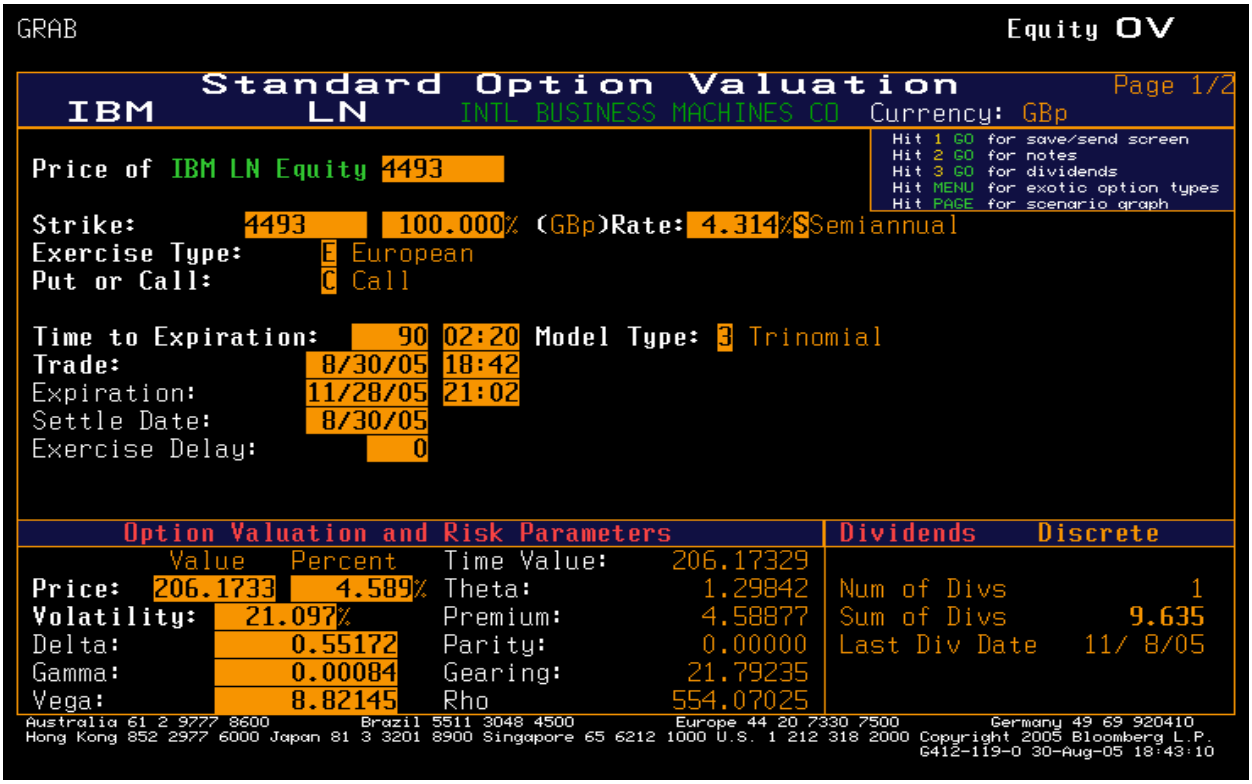

Figure 1.5 Standard Option Valuations – Trinomial

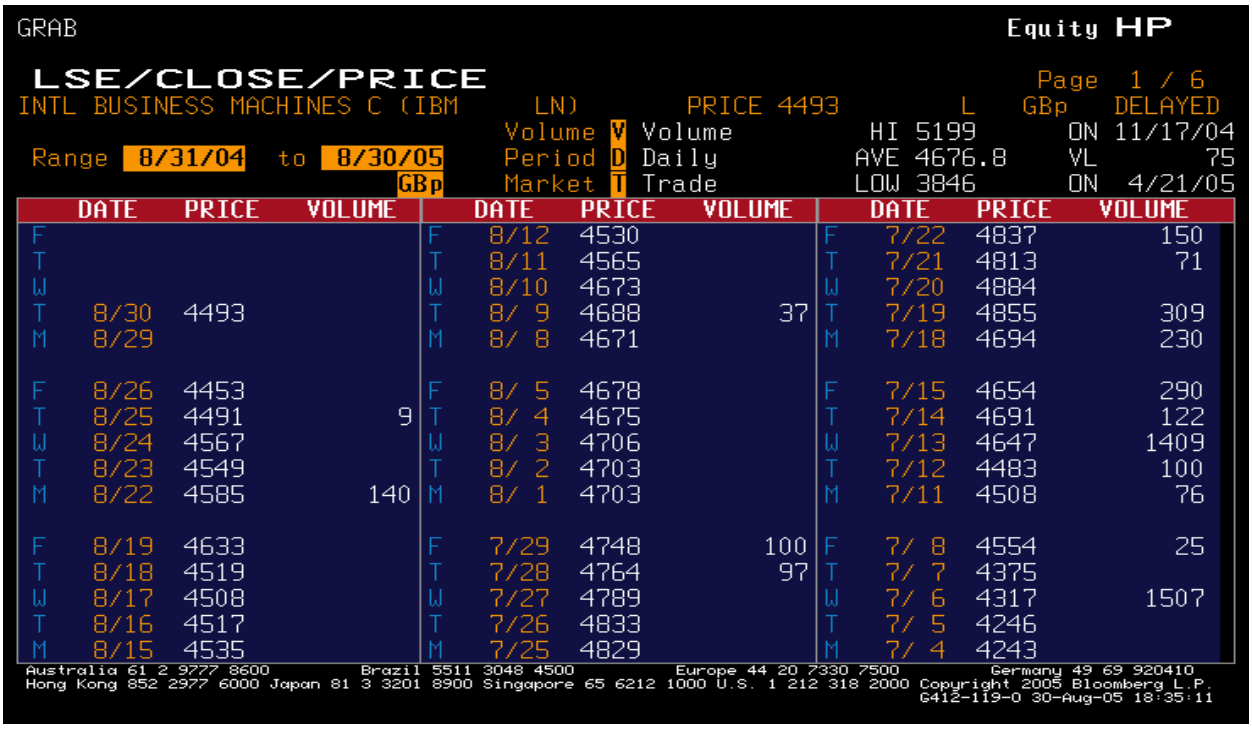

Figure 1.6 Closing Prices

Asian Options
Average Rate Analytical Approximations (1991, 1992)

Inputs

- Terms to Change are those highlighted in blue below

Option Type	Call
Time to Averaging Start	**0.000 yrs**
Time to Maturity	**0.246 yrs**
Remaining Time to Maturity	**0.246 yrs**
Stock Price	**4,493.000**
Strike Price	**4,493.000**
Average Price	**4,493.000**
Risk Free Rate	**4.31%**
Observed Dividend Yield	**0.87%**
Observed Volatility	**21.10%**
Additional Inputs	
Number of Averaging Points	**4**
Time to First Averaging Point	0.000 yrs
Time Between Averaging Points	**0.062**

Outputs

Option	Price
Turnbull-Wakeman Approximation	
Call Option	**117.150438**
Levy Approximation	
Call Option	**117.150332**
Curran's Approximation	
Call Option	**0.1307**

Note: Curran's Approximation Incomplete

http://www.global-derivatives.com
Kevin Cheng (Jun 23, 2003)
The sheet is protected in order to prevent calculated cells from being mistakenly altered
For unprotected versions, please email **info@global-derivatives.com**

Figure 1.6 Global Derivatives - Average Rate Analytical Approximations

Asian Options (Geometric Closed Form)
Kemna & Vorst (1990)

Inputs

- Terms to Change are those highlighted in blue below

Select Option Type	Call
Start Date	**30-Aug-05**
Maturity Date	**28-Nov-05**
Stock Price	**4,493.000**
Strike Price	**4,493.000**
Dividend Yield	**0.87%**
Risk Free Rate	**4.31%**
Volatility	**21.10%**
Time to Maturity	0.247 yrs

Outputs

Option	Price
Call Option	**115.0087**

http://www.global-derivatives.com
Kevin Cheng (Jun 30, 2003)
The sheet is protected in order to prevent calculated cells from being mistakenly altered
For unprotected versions, please email **info@global-derivatives.com**

Figure 1.7 Global Derivatives – Geometric Closed Form

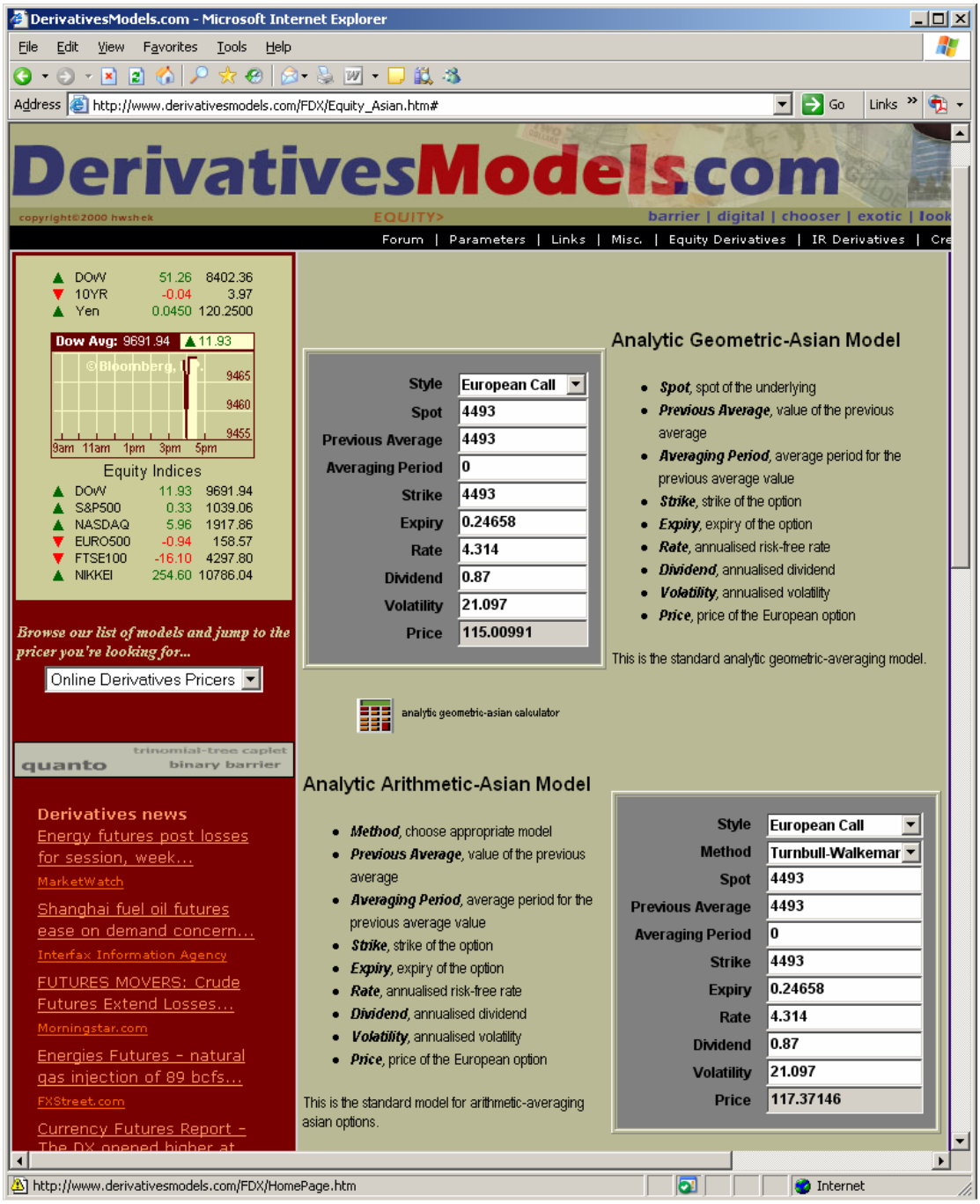

Figure 1.8 Derivatives Models – Turnbull and Wakeman

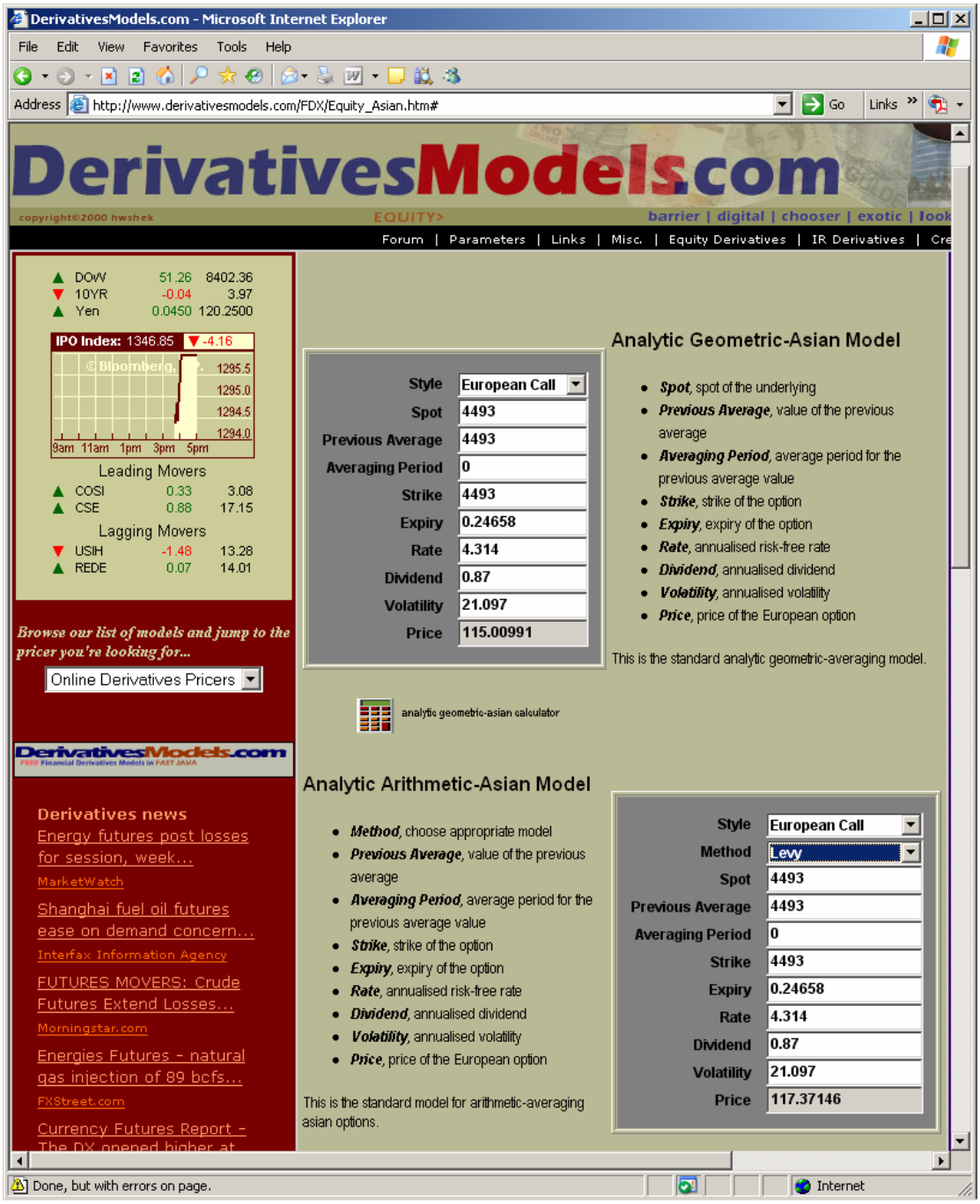

Figure 1.9 Derivatives Models – Levy

Appendix B

Design and Implementation

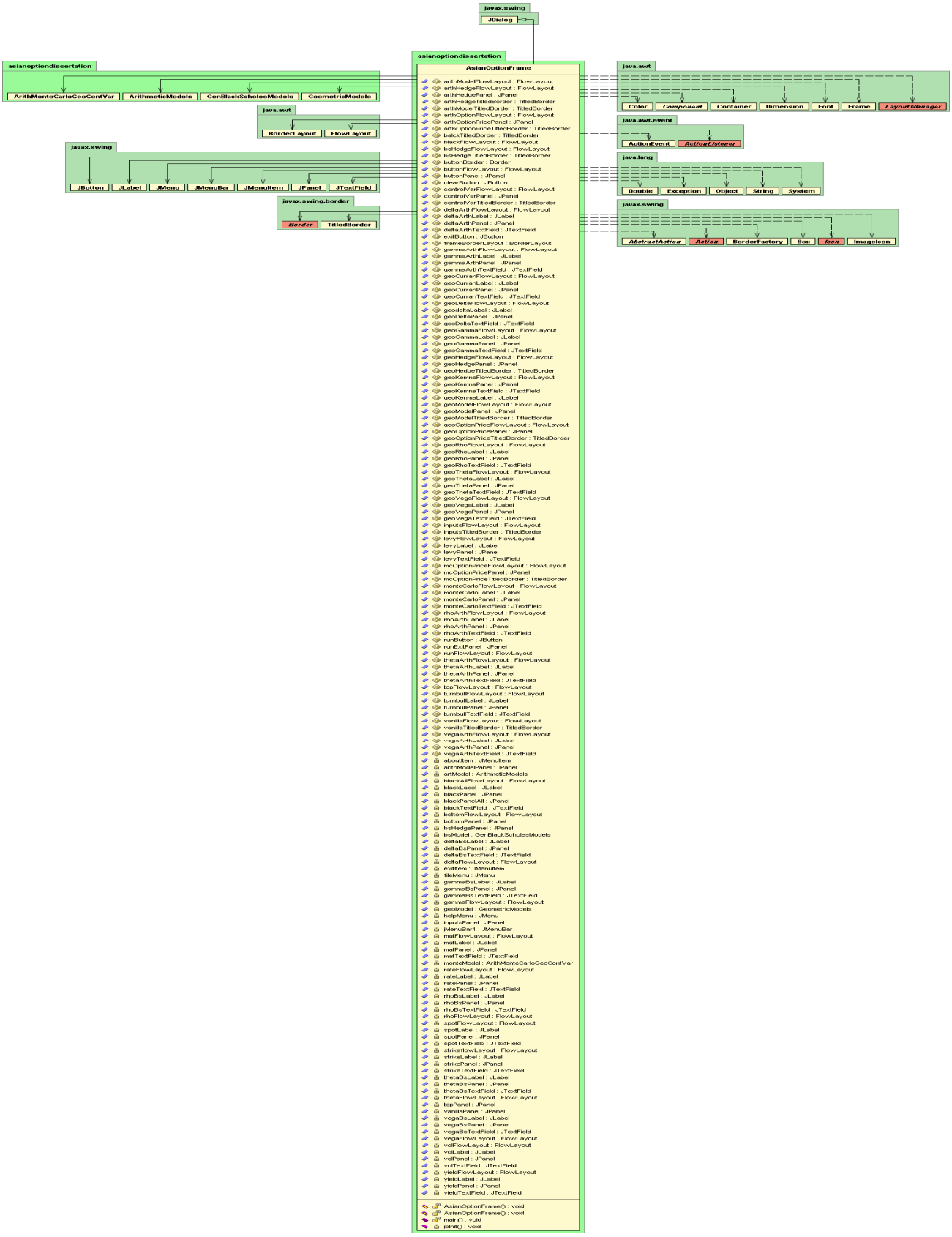

Figure 2.0 Detail Class Diagram of AsianOptionFrame class

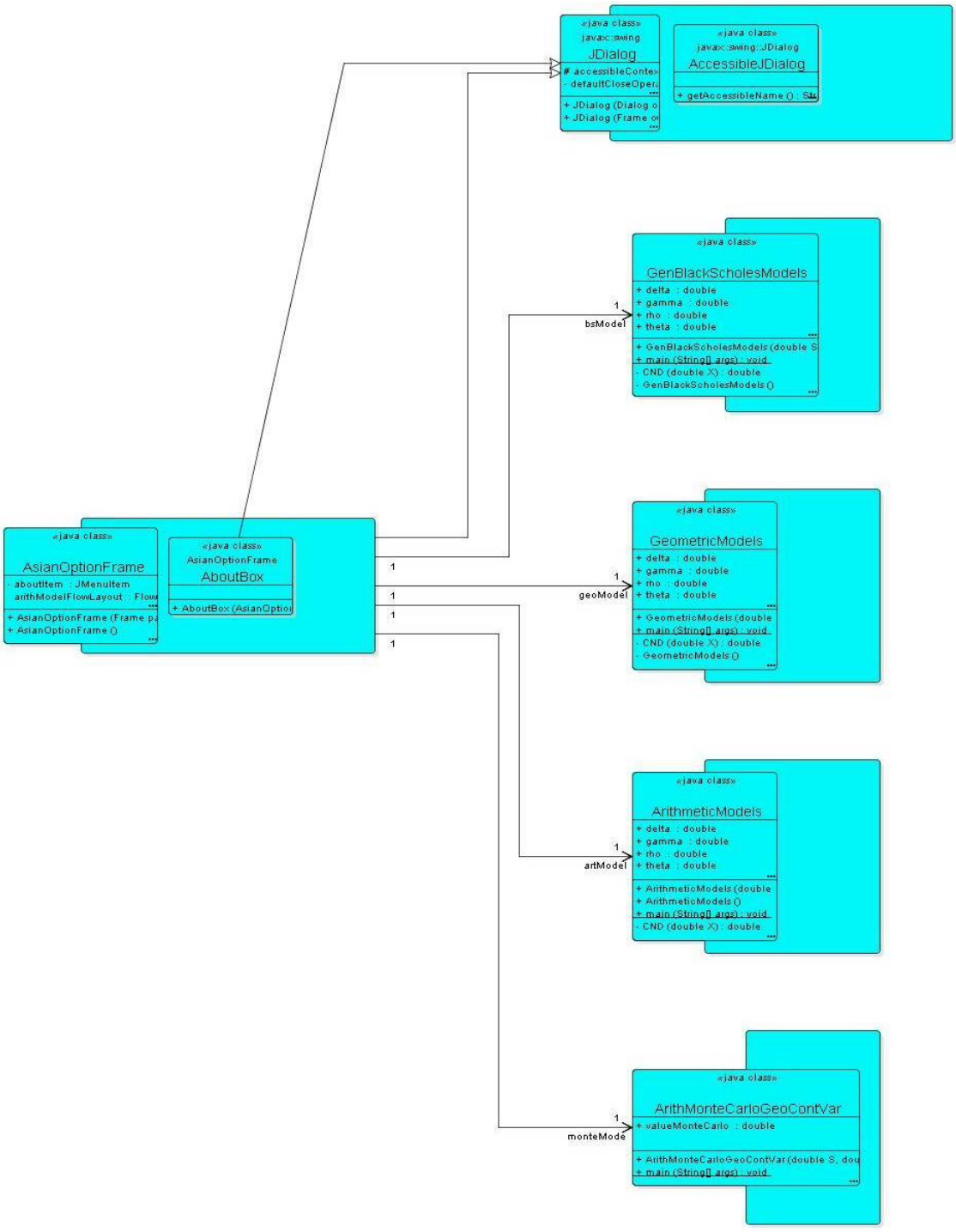

Figure 2.1 Detail Class Diagram of Asian Option Calculator

Appendix C

Source Code

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.Font;
import java.awt.Dimension;
import javax.swing.BorderFactory;

/**
 * Created by IntelliJ IDEA.
 * Author: Plamen Stilyianov
 * Title: MSc Financial Markets with Information Systems
 * Date: 02-Sep-2005
 * Time: 00:42:00
 * To change this template use File | Settings | File Templates.
 */


public class AsianOptionFrame extends JDialog {
   private JPanel topPanel = new JPanel();

   // Import algorithms
   private ArithMonteCarloGeoContVar monteModel;
   private GeometricModels geoModel;
   private GenBlackScholesModels bsModel;
   private ArithmeticModels artModel;


   // Menue bar
   private JMenuBar jMenuBar1 = new JMenuBar();
   private JMenu fileMenu = new JMenu();
   private JMenuItem exitItem = new JMenuItem();
   private JMenu helpMenu = new JMenu();
   private JMenuItem aboutItem = new JMenuItem();

   //Data Input
   TitledBorder inputsTitledBorder = new TitledBorder(BorderFactory.
         createEtchedBorder(Color.white, new Color(142, 142, 142)), "Data Input");
   private JPanel inputsPanel = new JPanel();
   private JPanel spotPanel = new JPanel();
   private JLabel spotLabel = new JLabel("Spot Price");
   private JTextField spotTextField = new JTextField("4493");
   private JPanel strikePanel = new JPanel();
   private JLabel strikeLabel = new JLabel("Strike Price");
   private JTextField strikeTextField = new JTextField("4493");
   private JPanel yieldPanel = new JPanel();
   private JLabel yieldLabel = new JLabel("Dividend Yield (%)");
   private JTextField yieldTextField = new JTextField("0.87");
   private JPanel ratePanel = new JPanel();
   private JLabel rateLabel = new JLabel("Risk Free Rate (%)");
   private JTextField rateTextField = new JTextField("4.314");
   private JPanel volPanel = new JPanel();
   private JLabel volLabel = new JLabel("Volatility (%)");
   private JTextField volTextField = new JTextField("21.097");
   private JPanel matPanel = new JPanel();
   private JLabel matLabel = new JLabel("Maturity (Yrs)");
   private JTextField matTextField = new JTextField("0.24658");

   Border buttonBorder = new TitledBorder(BorderFactory.createEtchedBorder(Color.
         white, new Color(134, 134, 134)), "");

   // Standard Vanilla Model
```

```
TitledBorder vanillaTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Standard Vanilla Model");
private JPanel vanillaPanel = new JPanel();
/ / Black Option Price
TitledBorder balckTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)), "Option Price");
private JPanel blackPanel = new JPanel();
private JPanel blackPanelAll = new JPanel();
private JLabel blackLabel = new JLabel("Black & Scholes");
private JTextField blackTextField = new JTextField();
/ / Hedge Sensitivities
TitledBorder bsHedgeTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Hedge Sensitivities");
private JPanel bsHedgePanel = new JPanel();
private JPanel deltaBsPanel = new JPanel();
private JLabel deltaBsLabel = new JLabel("Delta");
private JTextField deltaBsTextField = new JTextField();
private JPanel gammaBsPanel = new JPanel();
private JLabel gammaBsLabel = new JLabel("Gamma");
private JTextField gammaBsTextField = new JTextField();
private JPanel thetaBsPanel = new JPanel();
private JLabel thetaBsLabel = new JLabel("Theta");
private JTextField thetaBsTextField = new JTextField();
private JPanel vegaBsPanel = new JPanel();
private JLabel vegaBsLabel = new JLabel("Vega");
private JTextField vegaBsTextField = new JTextField();
private JPanel rhoBsPanel = new JPanel();
private JLabel rhoBsLabel = new JLabel("Rho");
private JTextField rhoBsTextField = new JTextField();
private FlowLayout spotFlowLayout = new FlowLayout();
private FlowLayout strikeflowLayout = new FlowLayout();
private FlowLayout yieldFlowLayout = new FlowLayout();
private FlowLayout volFlowLayout = new FlowLayout();
private FlowLayout rateFlowLayout = new FlowLayout();
private FlowLayout matFlowLayout = new FlowLayout();
private FlowLayout deltaFlowLayout = new FlowLayout();
private FlowLayout gammaFlowLayout = new FlowLayout();
private FlowLayout thetaFlowLayout = new FlowLayout();
private FlowLayout vegaFlowLayout = new FlowLayout();
private FlowLayout rhoFlowLayout = new FlowLayout();
private FlowLayout bottomFlowLayout = new FlowLayout();
private FlowLayout blackAllFlowLayout = new FlowLayout();

FlowLayout inputsFlowLayout = new FlowLayout();
FlowLayout bsHedgeFlowLayout = new FlowLayout();
FlowLayout topFlowLayout = new FlowLayout();
FlowLayout vanillaFlowLayout = new FlowLayout();
FlowLayout blackFlowLayout = new FlowLayout();

/ / Bottom Panel
TitledBorder arthModelTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Arithmetic Model");
TitledBorder arthOptionPriceTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)), "Option Price");
TitledBorder arthHedgeTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Hedge Sensitivities");

TitledBorder geoModelTitledBorder = new TitledBorder(BorderFactory.
```

```
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Geometric Model");
TitledBorder geoOptionPriceTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)), "Option Price");
TitledBorder geoHedgeTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Hedge Sensitivities");

TitledBorder controlVarTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)),
      "Arithmetic Model with Geometric Variate");
TitledBorder mcOptionPriceTitledBorder = new TitledBorder(BorderFactory.
      createEtchedBorder(Color.white, new Color(142, 142, 142)), "Option Price");

private JPanel bottomPanel = new JPanel();
private JPanel arithModelPanel = new JPanel();
FlowLayout arithModelFlowLayout = new FlowLayout();
JPanel arthOptionPricePanel = new JPanel();
JPanel turnbullPanel = new JPanel();
JLabel turnbullLabel = new JLabel();
JTextField turnbullTextField = new JTextField();
JPanel gammaArthPanel = new JPanel();
JLabel gammaArthLabel = new JLabel("Delta");
JTextField gammaArthTextField = new JTextField();
JPanel arthHedgePanel = new JPanel();
JPanel deltaArthPanel = new JPanel();
JLabel deltaArthLabel = new JLabel("Gamma");
JTextField deltaArthTextField = new JTextField();
JPanel thetaArthPanel = new JPanel();
JLabel thetaArthLabel = new JLabel("Theta");
JTextField thetaArthTextField = new JTextField();
JPanel vegaArthPanel = new JPanel();
JLabel vegaArthLabel = new JLabel("Vega");
JTextField vegaArthTextField = new JTextField();
JPanel rhoArthPanel = new JPanel();
JLabel rhoArthLabel = new JLabel("Rho");
JTextField rhoArthTextField = new JTextField();
JPanel levyPanel = new JPanel();
JLabel levyLabel = new JLabel();
JTextField levyTextField = new JTextField();
BorderLayout frameBorderLayout = new BorderLayout();
FlowLayout arthOptionFlowLayout = new FlowLayout();
FlowLayout levyFlowLayout = new FlowLayout();
FlowLayout turnbullFlowLayout = new FlowLayout();
FlowLayout arthHedgeFlowLayout = new FlowLayout();
FlowLayout deltaArthFlowLayout = new FlowLayout();
FlowLayout gammaArthFlowLayout = new FlowLayout();
FlowLayout thetaArthFlowLayout = new FlowLayout();
FlowLayout vegaArthFlowLayout = new FlowLayout();
FlowLayout rhoArthFlowLayout = new FlowLayout();
JPanel geoModelPanel = new JPanel();
JPanel geoOptionPricePanel = new JPanel();
JPanel geoKemnaPanel = new JPanel();
JLabel geoKenmaLabel = new JLabel();
JTextField geoKemnaTextField = new JTextField();
JPanel geoCurranPanel = new JPanel();
JLabel geoCurranLabel = new JLabel();
JTextField geoCurranTextField = new JTextField();
JPanel geoHedgePanel = new JPanel();
JPanel geoDeltaPanel = new JPanel();
JLabel geodeltaLabel = new JLabel("Delta");
JTextField geoDeltaTextField = new JTextField();
```

```
JPanel geoGammaPanel = new JPanel();
JLabel geoGammaLabel = new JLabel("Gamma");
JTextField geoGammaTextField = new JTextField();
JPanel geoThetaPanel = new JPanel();
JLabel geoThetaLabel = new JLabel("Theta");
JTextField geoThetaTextField = new JTextField();
JPanel geoVegaPanel = new JPanel();
JLabel geoVegaLabel = new JLabel("Vega");
JTextField geoVegaTextField = new JTextField();
JPanel geoRhoPanel = new JPanel();
JLabel geoRhoLabel = new JLabel("Rho");
JTextField geoRhoTextField = new JTextField();
FlowLayout geoModelFlowLayout = new FlowLayout();
FlowLayout geoOptionPriceFlowLayout = new FlowLayout();
FlowLayout geoKemnaFlowLayout = new FlowLayout();
FlowLayout geoCurranFlowLayout = new FlowLayout();
FlowLayout geoHedgeFlowLayout = new FlowLayout();
FlowLayout geoDeltaFlowLayout = new FlowLayout();
FlowLayout geoGammaFlowLayout = new FlowLayout();
FlowLayout geoThetaFlowLayout = new FlowLayout();
FlowLayout geoVegaFlowLayout = new FlowLayout();
FlowLayout geoRhoFlowLayout = new FlowLayout();
JPanel controlVarPanel = new JPanel();
JPanel mcOptionPricePanel = new JPanel();
JPanel monteCarloPanel = new JPanel();
JLabel monteCarloLabel = new JLabel();
JTextField monteCarloTextField = new JTextField();
FlowLayout controlVarFlowLayout = new FlowLayout();
FlowLayout mcOptionPriceFlowLayout = new FlowLayout();
FlowLayout monteCarloFlowLayout = new FlowLayout();
JPanel buttonPanel = new JPanel();
JPanel runExitPanel = new JPanel();
FlowLayout buttonFlowLayout = new FlowLayout();
JButton runButton = new JButton();
JButton clearButton = new JButton();
JButton exitButton = new JButton();
FlowLayout runFlowLayout = new FlowLayout();

public AsianOptionFrame() {
  this(null, "", false);
}

/**
 * @param modal
 * @param title
 * @param parent
 */
public AsianOptionFrame(Frame parent, String title, boolean modal) {
  super(parent, title, modal);
  try {
    jbInit();
  }
  catch (Exception e) {
    e.printStackTrace();
  }

}

private void jbInit() throws Exception {
  this.setSize(new Dimension(950, 630));
  this.getContentPane().setLayout(frameBorderLayout);
  this.setTitle("European Asian Call Option Calculator");
```

```
this.setJMenuBar(jMenuBar1);
spotLabel.setOpaque(true);
strikeLabel.setOpaque(true);
rateLabel.setOpaque(true);
yieldLabel.setOpaque(true);
volLabel.setOpaque(true);
matLabel.setOpaque(true);
blackLabel.setOpaque(true);
deltaBsLabel.setOpaque(true);
gammaBsLabel.setOpaque(true);
thetaBsLabel.setOpaque(true);
vegaBsLabel.setOpaque(true);
rhoBsLabel.setOpaque(true);
turnbullLabel.setOpaque(true);
levyLabel.setOpaque(true);
deltaArthLabel.setOpaque(true);
gammaArthLabel.setOpaque(true);
thetaArthLabel.setOpaque(true);
vegaArthLabel.setOpaque(true);
rhoArthLabel.setOpaque(true);
geoKenmaLabel.setOpaque(true);
geoCurranLabel.setOpaque(true);
geodeltaLabel.setOpaque(true);
geoGammaLabel.setOpaque(true);
geoThetaLabel.setOpaque(true);
geoVegaLabel.setOpaque(true);
geoRhoLabel.setOpaque(true);
monteCarloLabel.setOpaque(true);

jMenuBar1.add(fileMenu);
jMenuBar1.add(helpMenu);
fileMenu.setMnemonic('F');
fileMenu.setText("File");
exitItem.setText("Exit");
exitItem.setMnemonic('X');
fileMenu.add(exitItem);
exitItem.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    System.exit(0);
  }
});
Action actionAbout = new AbstractAction("About",
    new ImageIcon("About.gif")) {
  public void actionPerformed(ActionEvent e) {
    AboutBox dlg = new AboutBox(AsianOptionFrame.this);
    dlg.setVisible(true);
  }
};
helpMenu.setMnemonic('H');
helpMenu.setText("Help");
aboutItem.setMnemonic('A');
//aboutItem.setText("About");
aboutItem = helpMenu.add(actionAbout);
helpMenu.add(aboutItem);


arthOptionFlowLayout.setVgap(0);
arthHedgeFlowLayout.setVgap(0);
arithModelFlowLayout.setVgap(0);
arthOptionPricePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
arithModelPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
vanillaFlowLayout.setHgap(20);
```

```
geoModelPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoModelPanel.setBorder(geoModelTitledBorder);
geoModelPanel.setPreferredSize(new Dimension(290, 330));
geoModelPanel.setLayout(geoModelFlowLayout);
geoOptionPricePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoOptionPricePanel.setBorder(geoOptionPriceTitledBorder);
geoOptionPricePanel.setPreferredSize(new Dimension(270, 100));
geoOptionPricePanel.setLayout(geoOptionPriceFlowLayout);
geoKemnaPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoKemnaPanel.setPreferredSize(new Dimension(250, 30));
geoKemnaPanel.setLayout(geoKemnaFlowLayout);
geoKenmaLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoKenmaLabel.setPreferredSize(new Dimension(125, 25));
geoKenmaLabel.setToolTipText("");
geoKenmaLabel.setText("Kemna & Vorst");
geoKemnaTextField.setPreferredSize(new Dimension(120, 25));
geoKemnaTextField.setEditable(false);
geoCurranPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoCurranPanel.setPreferredSize(new Dimension(250, 30));
geoCurranPanel.setLayout(geoCurranFlowLayout);
geoCurranLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoCurranLabel.setPreferredSize(new Dimension(125, 25));
geoCurranLabel.setText("Curran");
geoCurranTextField.setPreferredSize(new Dimension(120, 25));
geoCurranTextField.setEditable(false);
geoHedgePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoHedgePanel.setBorder(geoHedgeTitledBorder);
geoHedgePanel.setPreferredSize(new Dimension(270, 190));
geoHedgePanel.setLayout(geoHedgeFlowLayout);
geoDeltaPanel.setPreferredSize(new Dimension(250, 30));
geoDeltaPanel.setLayout(geoDeltaFlowLayout);
geodeltaLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geodeltaLabel.setPreferredSize(new Dimension(120, 25));
geodeltaLabel.setText("Delta");
geoDeltaTextField.setPreferredSize(new Dimension(120, 25));
geoDeltaTextField.setEditable(false);
geoGammaPanel.setPreferredSize(new Dimension(250, 30));
geoGammaPanel.setLayout(geoGammaFlowLayout);
geoGammaLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoGammaLabel.setPreferredSize(new Dimension(120, 25));
geoGammaLabel.setText("Gamma");
geoGammaTextField.setPreferredSize(new Dimension(120, 25));
geoGammaTextField.setEditable(false);
geoThetaPanel.setPreferredSize(new Dimension(250, 30));
geoThetaPanel.setLayout(geoThetaFlowLayout);
geoThetaLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoThetaLabel.setPreferredSize(new Dimension(120, 25));
geoThetaTextField.setPreferredSize(new Dimension(120, 25));
geoThetaTextField.setEditable(false);
geoVegaPanel.setPreferredSize(new Dimension(250, 30));
geoVegaPanel.setLayout(geoVegaFlowLayout);
geoVegaLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoVegaLabel.setPreferredSize(new Dimension(120, 25));
geoVegaTextField.setPreferredSize(new Dimension(120, 25));
geoVegaTextField.setEditable(false);
geoRhoPanel.setPreferredSize(new Dimension(250, 30));
geoRhoPanel.setLayout(geoRhoFlowLayout);
geoRhoLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
geoRhoLabel.setPreferredSize(new Dimension(120, 25));
geoRhoTextField.setPreferredSize(new Dimension(120, 25));
geoRhoTextField.setEditable(false);
geoModelFlowLayout.setAlignment(FlowLayout.LEFT);
```

```
geoModelFlowLayout.setVgap(0);
geoKemnaFlowLayout.setHgap(0);
geoCurranFlowLayout.setHgap(0);
geoOptionPriceFlowLayout.setHgap(0);
geoOptionPriceFlowLayout.setVgap(0);
geoHedgeFlowLayout.setAlignment(FlowLayout.LEFT);
geoHedgeFlowLayout.setVgap(0);
vanillaPanel.setMinimumSize(new Dimension(450, 95));
controlVarPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
controlVarPanel.setBorder(controlVarTitledBorder);
controlVarPanel.setPreferredSize(new Dimension(270, 115));
controlVarPanel.setLayout(controlVarFlowLayout);
mcOptionPricePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
mcOptionPricePanel.setBorder(mcOptionPriceTitledBorder);
mcOptionPricePanel.setPreferredSize(new Dimension(250, 75));
mcOptionPricePanel.setLayout(mcOptionPriceFlowLayout);
monteCarloPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
monteCarloPanel.setPreferredSize(new Dimension(230, 30));
monteCarloPanel.setLayout(monteCarloFlowLayout);
monteCarloLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
monteCarloLabel.setPreferredSize(new Dimension(90, 25));
monteCarloLabel.setText("Monte Carlo");
monteCarloTextField.setPreferredSize(new Dimension(120, 25));
monteCarloTextField.setEditable(false);
controlVarFlowLayout.setAlignment(FlowLayout.LEFT);
controlVarFlowLayout.setVgap(0);
mcOptionPriceFlowLayout.setHgap(0);
mcOptionPriceFlowLayout.setVgap(0);
buttonPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
buttonPanel.setBorder(null);
buttonPanel.setPreferredSize(new Dimension(290, 330));
buttonPanel.setLayout(buttonFlowLayout);
runExitPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
runExitPanel.setBorder(null);
runExitPanel.setPreferredSize(new Dimension(250, 210));
runExitPanel.setLayout(runFlowLayout);
buttonFlowLayout.setHgap(0);
buttonFlowLayout.setVgap(0);
bottomPanel.setBorder(BorderFactory.createEtchedBorder());
topPanel.setBorder(BorderFactory.createEtchedBorder());
blackFlowLayout.setAlignment(FlowLayout.LEFT);
blackAllFlowLayout.setAlignment(FlowLayout.LEFT);
blackTextField.setEditable(false);
runButton.setText("Run");
clearButton.setText("Clear");
exitButton.setText("Exit");
runFlowLayout.setVgap(100);
thetaBsTextField.setEditable(false);
deltaBsTextField.setEditable(false);
gammaBsTextField.setEditable(false);
vegaBsTextField.setEditable(false);
rhoBsTextField.setEditable(false);
rhoArthTextField.setEditable(false);
vegaArthTextField.setEditable(false);
thetaArthTextField.setEditable(false);
gammaArthTextField.setEditable(false);
deltaArthTextField.setEditable(false);
levyTextField.setEditable(false);
turnbullTextField.setEditable(false);
this.getContentPane().add(topPanel, java.awt.BorderLayout.NORTH);
this.getContentPane().add(bottomPanel, java.awt.BorderLayout.CENTER);
```

```
/* ================= Top Panel ============================== */
topPanel.setLayout(topFlowLayout);
topPanel.setPreferredSize(new Dimension(950, 275));

inputsPanel.setFont(new Font("Verdana", 0, 11));
inputsPanel.setAlignmentX((float) 0.0);
inputsPanel.setAlignmentY((float) 0.0);
inputsPanel.setPreferredSize(new Dimension(290, 260));
spotPanel.setLayout(spotFlowLayout);
spotPanel.setPreferredSize(new Dimension(250, 30));
spotLabel.setPreferredSize(new Dimension(120, 25));
spotLabel.setFont(new Font("Verdana", 0, 11));
spotTextField.setPreferredSize(new Dimension(120, 25));
strikePanel.setLayout(strikeflowLayout);
strikePanel.setPreferredSize(new Dimension(250, 30));
strikeLabel.setPreferredSize(new Dimension(120, 25));
strikeLabel.setFont(new Font("Verdana", 0, 11));
strikeTextField.setPreferredSize(new Dimension(120, 25));
yieldPanel.setLayout(yieldFlowLayout);
yieldPanel.setPreferredSize(new Dimension(250, 30));
yieldLabel.setPreferredSize(new Dimension(120, 25));
yieldLabel.setFont(new Font("Verdana", 0, 11));
yieldTextField.setPreferredSize(new Dimension(120, 25));
ratePanel.setLayout(rateFlowLayout);
ratePanel.setPreferredSize(new Dimension(250, 30));
rateLabel.setPreferredSize(new Dimension(120, 25));
rateLabel.setFont(new Font("Verdana", 0, 11));
rateTextField.setPreferredSize(new Dimension(120, 25));
volPanel.setLayout(volFlowLayout);
volPanel.setPreferredSize(new Dimension(250, 30));
volLabel.setPreferredSize(new Dimension(120, 25));
volLabel.setFont(new Font("Verdana", 0, 11));
volTextField.setPreferredSize(new Dimension(120, 25));
matPanel.setLayout(matFlowLayout);
matPanel.setPreferredSize(new Dimension(250, 30));
matLabel.setPreferredSize(new Dimension(120, 25));
matLabel.setFont(new Font("Verdana", 0, 11));
matTextField.setPreferredSize(new Dimension(120, 25));

vanillaPanel.setLayout(vanillaFlowLayout);
vanillaPanel.setFont(new Font("Verdana", 0, 11));
vanillaPanel.setPreferredSize(new Dimension(590, 260));

blackPanel.setLayout(blackFlowLayout);
blackPanel.setPreferredSize(new Dimension(270, 220));
blackPanelAll.setFont(new Font("Verdana", 0, 11));
blackPanelAll.setPreferredSize(new Dimension(270, 75));
blackPanelAll.setLayout(blackAllFlowLayout);
blackLabel.setPreferredSize(new Dimension(110, 25));
blackLabel.setFont(new Font("Verdana", 0, 11));
blackTextField.setPreferredSize(new Dimension(120, 25));
bsHedgePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));

bsHedgePanel.setPreferredSize(new Dimension(260, 220));
deltaBsPanel.setLayout(deltaFlowLayout);
deltaBsPanel.setPreferredSize(new Dimension(230, 30));
deltaBsLabel.setPreferredSize(new Dimension(100, 25));
deltaBsLabel.setFont(new Font("Verdana", 0, 11));
deltaBsTextField.setPreferredSize(new Dimension(120, 25));
gammaBsPanel.setLayout(gammaFlowLayout);
gammaBsPanel.setPreferredSize(new Dimension(230, 30));
gammaBsLabel.setPreferredSize(new Dimension(100, 25));
```

```
gammaBsLabel.setFont(new Font("Verdana", 0, 11));
gammaBsTextField.setPreferredSize(new Dimension(120, 25));
thetaBsPanel.setLayout(thetaFlowLayout);
thetaBsPanel.setPreferredSize(new Dimension(230, 30));
thetaBsLabel.setPreferredSize(new Dimension(100, 25));
thetaBsLabel.setFont(new Font("Verdana", 0, 11));
thetaBsTextField.setPreferredSize(new Dimension(120, 25));
vegaBsPanel.setLayout(vegaFlowLayout);
vegaBsPanel.setPreferredSize(new Dimension(230, 30));
vegaBsLabel.setPreferredSize(new Dimension(100, 25));
vegaBsLabel.setFont(new Font("Verdana", 0, 11));
vegaBsTextField.setPreferredSize(new Dimension(120, 25));
rhoBsPanel.setLayout(rhoFlowLayout);
rhoBsPanel.setPreferredSize(new Dimension(230, 30));
rhoBsLabel.setPreferredSize(new Dimension(100, 25));
rhoBsLabel.setFont(new Font("Verdana", 0, 11));
rhoBsTextField.setPreferredSize(new Dimension(120, 25));

spotFlowLayout.setAlignment(FlowLayout.LEFT);
strikeflowLayout.setAlignment(FlowLayout.LEFT);
yieldFlowLayout.setAlignment(FlowLayout.LEFT);
rateFlowLayout.setAlignment(FlowLayout.LEFT);
volFlowLayout.setAlignment(FlowLayout.LEFT);
matFlowLayout.setAlignment(FlowLayout.LEFT);
vanillaFlowLayout.setAlignment(FlowLayout.LEFT);
vanillaFlowLayout.setVgap(0);
blackFlowLayout.setHgap(0);
blackFlowLayout.setVgap(0);
topFlowLayout.setAlignment(FlowLayout.LEFT);
topFlowLayout.setHgap(20);
inputsPanel.add(spotPanel);
spotPanel.add(spotLabel);
spotPanel.add(spotTextField);
inputsPanel.add(strikePanel);
strikePanel.add(strikeLabel);
strikePanel.add(strikeTextField);
inputsPanel.add(ratePanel);
yieldPanel.add(yieldLabel);
yieldPanel.add(yieldTextField);
inputsPanel.add(yieldPanel);
ratePanel.add(rateLabel);
ratePanel.add(rateTextField);
inputsPanel.add(volPanel);
volPanel.add(volLabel);
volPanel.add(volTextField);
inputsPanel.add(matPanel);
matPanel.add(matLabel);
matPanel.add(matTextField);
topPanel.add(inputsPanel);
topPanel.add(vanillaPanel);
inputsPanel.setLayout(inputsFlowLayout);
inputsPanel.setBorder(inputsTitledBorder);

//Standard Vanilla Model
vanillaPanel.setBorder(vanillaTitledBorder); //Option Price
blackPanelAll.setBorder(balckTitledBorder);
blackPanelAll.add(blackLabel);
blackPanelAll.add(blackTextField);
vanillaPanel.add(blackPanel);
blackPanel.add(blackPanelAll);
vanillaPanel.add(bsHedgePanel);
blackPanel.add(Box.createVerticalGlue());
```

```
//Hedge Sensitivities
bsHedgePanel.setBorder(bsHedgeTitledBorder);
bsHedgePanel.setLayout(bsHedgeFlowLayout);
deltaBsPanel.add(deltaBsLabel);
deltaBsPanel.add(deltaBsTextField);
bsHedgePanel.add(deltaBsPanel);
bsHedgePanel.add(gammaBsPanel);
bsHedgePanel.add(thetaBsPanel);
gammaBsPanel.add(gammaBsLabel);
gammaBsPanel.add(gammaBsTextField);
thetaBsPanel.add(thetaBsLabel);
thetaBsPanel.add(thetaBsTextField);
bsHedgePanel.add(vegaBsPanel);
vegaBsPanel.add(vegaBsLabel);
vegaBsPanel.add(vegaBsTextField);
bsHedgePanel.add(rhoBsPanel);
rhoBsPanel.add(rhoBsLabel);
rhoBsPanel.add(rhoBsTextField);

/* ================= Bottom Panel ============================== */
// Bottom Panel
bottomPanel.setLayout(bottomFlowLayout);
bottomPanel.setPreferredSize(new Dimension(950, 350));
bottomFlowLayout.setAlignment(FlowLayout.LEFT);
bottomFlowLayout.setHgap(20);
arithModelPanel.setLayout(arithModelFlowLayout);
arithModelPanel.setBorder(arthModelTitledBorder);
arithModelPanel.setPreferredSize(new Dimension(290, 330));
arthOptionPricePanel.setBorder(arthOptionPriceTitledBorder);
arthOptionPricePanel.setPreferredSize(new Dimension(270, 100));
arthOptionPricePanel.setLayout(arthOptionFlowLayout);
turnbullPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
turnbullPanel.setPreferredSize(new Dimension(250, 30));
turnbullPanel.setLayout(turnbullFlowLayout);
turnbullLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
turnbullLabel.setPreferredSize(new Dimension(130, 25));
turnbullLabel.setToolTipText("");
turnbullLabel.setText("Turnbull & Wakeman");
turnbullTextField.setPreferredSize(new Dimension(120, 25));
arthHedgePanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
arthHedgePanel.setBorder(arthHedgeTitledBorder);
arthHedgePanel.setPreferredSize(new Dimension(270, 190));
arthHedgePanel.setLayout(arthHedgeFlowLayout);
deltaArthPanel.setPreferredSize(new Dimension(250, 30));
deltaArthPanel.setLayout(deltaArthFlowLayout);
deltaArthLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
deltaArthLabel.setPreferredSize(new Dimension(120, 25));
deltaArthLabel.setText("Delta");
deltaArthTextField.setPreferredSize(new Dimension(120, 25));
gammaArthPanel.setPreferredSize(new Dimension(250, 30));
gammaArthPanel.setLayout(gammaArthFlowLayout);
gammaArthLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
gammaArthLabel.setPreferredSize(new Dimension(120, 25));
gammaArthLabel.setText("Gamma");
gammaArthTextField.setPreferredSize(new Dimension(120, 25));
thetaArthPanel.setPreferredSize(new Dimension(250, 30));
thetaArthPanel.setLayout(thetaArthFlowLayout);
thetaArthLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
thetaArthLabel.setPreferredSize(new Dimension(120, 25));
thetaArthTextField.setPreferredSize(new Dimension(120, 25));
vegaArthPanel.setPreferredSize(new Dimension(250, 30));
```

```
vegaArthPanel.setLayout(vegaArthFlowLayout);
vegaArthLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
vegaArthLabel.setPreferredSize(new Dimension(120, 25));
vegaArthTextField.setPreferredSize(new Dimension(120, 25));
rhoArthPanel.setPreferredSize(new Dimension(250, 30));
rhoArthPanel.setLayout(rhoArthFlowLayout);
rhoArthLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
rhoArthLabel.setPreferredSize(new Dimension(120, 25));
rhoArthTextField.setPreferredSize(new Dimension(120, 25));
levyPanel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
levyPanel.setPreferredSize(new Dimension(250, 30));
levyPanel.setLayout(levyFlowLayout);
levyLabel.setFont(new java.awt.Font("Verdana", Font.PLAIN, 11));
levyLabel.setPreferredSize(new Dimension(130, 25));
levyLabel.setText("Levy");
levyTextField.setPreferredSize(new Dimension(120, 25));
arithModelFlowLayout.setAlignment(FlowLayout.LEFT);
arthHedgeFlowLayout.setAlignment(FlowLayout.LEFT);
arthOptionFlowLayout.setHgap(0);
turnbullFlowLayout.setHgap(0);
levyFlowLayout.setAlignment(FlowLayout.LEFT);
levyFlowLayout.setHgap(0);
deltaArthFlowLayout.setAlignment(FlowLayout.LEFT);
gammaArthFlowLayout.setAlignment(FlowLayout.LEFT);
thetaArthFlowLayout.setAlignment(FlowLayout.LEFT);
vegaArthFlowLayout.setAlignment(FlowLayout.LEFT);
rhoArthFlowLayout.setAlignment(FlowLayout.LEFT);
bottomPanel.add(arithModelPanel);
bottomPanel.add(geoModelPanel);
bottomPanel.add(buttonPanel);
geoRhoPanel.add(geoRhoLabel);
geoRhoPanel.add(geoRhoTextField);
geoVegaPanel.add(geoVegaLabel);
geoVegaPanel.add(geoVegaTextField);
geoHedgePanel.add(geoDeltaPanel);
geoHedgePanel.add(geoGammaPanel);
geoHedgePanel.add(geoThetaPanel);
geoHedgePanel.add(geoVegaPanel);
geoHedgePanel.add(geoRhoPanel);
geoThetaPanel.add(geoThetaLabel);
geoThetaPanel.add(geoThetaTextField);
geoGammaPanel.add(geoGammaLabel);
geoGammaPanel.add(geoGammaTextField);
geoDeltaPanel.add(geodeltaLabel);
geoDeltaPanel.add(geoDeltaTextField);
geoCurranPanel.add(geoCurranLabel);
geoCurranPanel.add(geoCurranTextField);
geoOptionPricePanel.add(geoKemnaPanel);
geoOptionPricePanel.add(geoCurranPanel);
geoModelPanel.add(geoOptionPricePanel);
geoModelPanel.add(geoHedgePanel);
geoKemnaPanel.add(geoKenmaLabel);
geoKemnaPanel.add(geoKemnaTextField);
rhoArthPanel.add(rhoArthLabel);
rhoArthPanel.add(rhoArthTextField);
vegaArthPanel.add(vegaArthLabel);
vegaArthPanel.add(vegaArthTextField);
arthHedgePanel.add(deltaArthPanel);
arthHedgePanel.add(gammaArthPanel);
arthHedgePanel.add(thetaArthPanel);
arthHedgePanel.add(vegaArthPanel);
arthHedgePanel.add(rhoArthPanel);
```

```
    thetaArthPanel.add(thetaArthLabel);
    thetaArthPanel.add(thetaArthTextField);
    gammaArthPanel.add(gammaArthLabel);
    gammaArthPanel.add(gammaArthTextField);
    deltaArthPanel.add(deltaArthLabel);
    deltaArthPanel.add(deltaArthTextField);
    levyPanel.add(levyLabel);
    levyPanel.add(levyTextField);
    arithModelPanel.add(arthOptionPricePanel);
    arithModelPanel.add(arthHedgePanel);
    turnbullPanel.add(turnbullLabel);
    turnbullPanel.add(turnbullTextField);
    arthOptionPricePanel.add(turnbullPanel);
    arthOptionPricePanel.add(levyPanel);
    monteCarloPanel.add(monteCarloLabel);
    monteCarloPanel.add(monteCarloTextField);
    buttonPanel.add(controlVarPanel);
    controlVarPanel.add(mcOptionPricePanel);
    mcOptionPricePanel.add(monteCarloPanel);
    buttonPanel.add(runExitPanel, null);
    runExitPanel.add(runButton);
    runExitPanel.add(clearButton);
    runExitPanel.add(exitButton);

    runButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {

        Double spPrice = new Double(spotTextField.getText());
        Double stPrice = new Double(strikeTextField.getText());
        Double rate = new Double(rateTextField.getText());
        Double div = new Double(yieldTextField.getText());
        Double vol = new Double(volTextField.getText());
        Double yrs = new Double(matTextField.getText());

        monteModel = new ArithMonteCarloGeoContVar(spPrice, stPrice, rate / 100, div / 100,
vol / 100, yrs);
        geoModel = new GeometricModels(spPrice, stPrice, rate / 100, div / 100, vol / 100,
yrs);
        bsModel = new GenBlackScholesModels(spPrice, stPrice, rate / 100, div / 100, vol /
100, yrs);
        artModel = new ArithmeticModels(spPrice, stPrice, rate / 100, div / 100, vol / 100,
yrs);

        //Black & Schles
        blackTextField.setText(String.valueOf(bsModel.valueBlackScholes).substring(0, 15));
        blackTextField.setForeground(Color.red);
        deltaBsTextField.setText(String.valueOf(bsModel.delta).substring(0, 15));
        gammaBsTextField.setText(String.valueOf(bsModel.gamma).substring(0, 15));
        thetaBsTextField.setText(String.valueOf(bsModel.theta).substring(0, 15));
        vegaBsTextField.setText(String.valueOf(bsModel.vega).substring(0, 15));
        rhoBsTextField.setText(String.valueOf(bsModel.rho).substring(0, 15));
        //Arithmetic Model
        turnbullTextField.setText(String.valueOf(artModel.valueTurnbullWakeman).substring(0,
15));
        turnbullTextField.setForeground(Color.red);
        levyTextField.setText(String.valueOf(artModel.valueLevy).substring(0, 15));
        levyTextField.setForeground(Color.red);
        deltaArthTextField.setText(String.valueOf(artModel.delta).substring(0, 15));
        gammaArthTextField.setText(String.valueOf(artModel.gamma).substring(0, 15));
        thetaArthTextField.setText(String.valueOf(artModel.theta).substring(0, 15));
        vegaArthTextField.setText(String.valueOf(artModel.vega).substring(0, 15));
        rhoArthTextField.setText(String.valueOf(artModel.rho).substring(0, 15));
```

```
            //Geometric Model
            geoKemnaTextField.setText(String.valueOf(geoModel.valueKemnaVorst).substring(0,
15));
            geoKemnaTextField.setForeground(Color.red);
            geoCurranTextField.setText(String.valueOf(geoModel.valueCurran).substring(0, 15));
            geoCurranTextField.setForeground(Color.red);
            geoDeltaTextField.setText(String.valueOf(geoModel.delta).substring(0, 15));
            geoGammaTextField.setText(String.valueOf(geoModel.gamma).substring(0, 15));
            geoThetaTextField.setText(String.valueOf(geoModel.theta).substring(0, 15));
            geoVegaTextField.setText(String.valueOf(geoModel.vega).substring(0, 15));
            geoRhoTextField.setText(String.valueOf(geoModel.rho).substring(0, 15));
            //Monte Carlo
          monteCarloTextField.setText(String.valueOf(monteModel.valueMonteCarlo).substring(0,
15));
            monteCarloTextField.setForeground(Color.red);

          }
        });

        clearButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            //Data Input
            spotTextField.setText("");
            strikeTextField.setText("");
            yieldTextField.setText("");
            rateTextField.setText("");
            volTextField.setText("");
            matTextField.setText("");
            //Black & Schles
            blackTextField.setText("");
            deltaBsTextField.setText("");
            gammaBsTextField.setText("");
            thetaBsTextField.setText("");
            vegaBsTextField.setText("");
            rhoBsTextField.setText("");
            //Arithmetic Model
            turnbullTextField.setText("");
            levyTextField.setText("");
            deltaArthTextField.setText("");
            gammaArthTextField.setText("");
            thetaArthTextField.setText("");
            vegaArthTextField.setText("");
            rhoArthTextField.setText("");
            //Geometric Model
            geoKemnaTextField.setText("");
            geoCurranTextField.setText("");
            geoDeltaTextField.setText("");
            geoGammaTextField.setText("");
            geoThetaTextField.setText("");
            geoVegaTextField.setText("");
            geoRhoTextField.setText("");
            //Monte Carlo
            monteCarloTextField.setText("");
          }
        });

        exitButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            System.exit(0);
          }
        });
      }
```

```
public static void main(String[] args) {
   AsianOptionFrame dialog = new AsianOptionFrame();
   dialog.pack();
   dialog.setVisible(true);
   // System.exit(0);
}

class AboutBox extends JDialog {

   public AboutBox(AsianOptionFrame owner) {
      super(owner, "About", true);
      this.setSize(new Dimension(300, 200));
      this.setBackground(super.getBackground());
      JLabel lbl = new JLabel(new ImageIcon("AsianCalculator.gif"));
      JPanel p = new JPanel();
      Border b1 = new BevelBorder(BevelBorder.LOWERED);
      Border b2 = new EmptyBorder(5, 5, 5, 5);
      lbl.setBorder(new CompoundBorder(b1, b2));
      p.add(lbl);
      getContentPane().add(p, BorderLayout.WEST);

      String message = "Asian Optian Calculator application\n" +
                "MSc Financial Markets with Information Systems \n" +
                "(c) P.Stilyianov 2004-2005";
      JTextArea txt = new JTextArea(message);
      txt.setBorder(new EmptyBorder(5, 10, 5, 10));
      txt.setFont(new Font("Helvetica", Font.BOLD, 12));
      txt.setEditable(false);
      txt.setBackground(owner.getBackground());
      p = new JPanel();
      p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
      p.add(txt);

      message = "JVM version " + System.getProperty("java.version") +
           " by " + System.getProperty("java.vendor");
      txt = new JTextArea(message);
      txt.setBorder(new EmptyBorder(5, 10, 5, 10));
      txt.setFont(new Font("Arial", Font.PLAIN, 12));
      txt.setEditable(false);
      txt.setLineWrap(true);
      txt.setWrapStyleWord(true);
      txt.setBackground(getBackground());
      p.add(txt);

      getContentPane().add(p, BorderLayout.CENTER);

      final JButton btOK = new JButton("OK");
      ActionListener lst = new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            dispose();
         }
      };
      btOK.addActionListener(lst);
      p = new JPanel();
      p.add(btOK);
      getRootPane().setDefaultButton(btOK);
      getRootPane().registerKeyboardAction(lst,
           KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
           JComponent.WHEN_IN_FOCUSED_WINDOW);
      getContentPane().add(p, BorderLayout.SOUTH);
```

```
        // That will transer focus from first component upon dialog's show
        WindowListener wl = new WindowAdapter() {
          public void windowOpened(WindowEvent e) {
            btOK.requestFocus();
          }
        };
        addWindowListener(wl);

        pack();
        setResizable(false);
        setLocationRelativeTo(owner);
      }
    }}
import java.util.Random;

/**
 * Created by IntelliJ IDEA.
 * Author: Plamen Stilyianov
 * Title: MSc Financial Markets with Information Systems
 * Date: 02-Sep-2005
 * Time: 00:42:00
 * To change this template use File | Settings | File Templates.
 */
public class ArithMonteCarloGeoContVar {
  // An European Arithmetic Asian Option with a Geometric Asian Option Control Variate

  public double valueMonteCarlo = 0.0;

  private ArithMonteCarloGeoContVar() {
  }

  public ArithMonteCarloGeoContVar(double S, double K, double r, double div, double sig, double
T) {
    double M = 100;   // Simulations
    double N = 1000;    // Time step

    valueMonteCarlo = modelMonteCarlo(S, K, r, div, sig, T, M, N);

  }

  private double modelMonteCarlo(double S, double K, double r, double div, double sig, double T,
double M, double N){

    // Constant
    double dt = T / N;   //  Time step
    double nudt = (r - div - 0.5 * sig * sig) * dt;    //
    double sigsdt = sig * Math.sqrt(dt);   //

    double delta = 0.0;
    double St = S;
    double sumSt = 0.0;
    long productSt = 1;
    double rndNum = 0.0;
    double arithAvgOpt = 0.0;
    double geomAvgOpt = 0.0;
    double diffOVal = 0.0;
    double diffOValSum = 0.0;
    double diffOValSumSq = 0.0;
    double powProd = 1 / N;

    for (int j = 0; j < M; j++) {
```

```
        St = S;
        sumSt = 0.0;
        productSt = 1;

        for (int i = 0; i < N; i++) {
           rndNum = randPolarRejc();
           St = St * Math.exp(nudt + (sigsdt * rndNum));
           sumSt = sumSt + St;
           productSt = (long) (productSt* St);
        }

        arithAvgOpt = sumSt / N;
        geomAvgOpt = Math.pow(productSt, powProd);
        diffOVal = Math.max(0.0, arithAvgOpt - K) - Math.max(0.0, geomAvgOpt - K);
        diffOValSum = diffOValSum + diffOVal;
        diffOValSumSq = diffOValSumSq + Math.pow(diffOVal, 2);
        delta = delta + arithAvgOpt;

     }

     double portfolioValue = (diffOValSum / M) * Math.exp(-r * T);
     double vega = (diffOValSum / 2 * sig) * Math.exp(-r * T);
     double sdDev = Math.sqrt(diffOValSumSq - diffOValSum * (diffOValSum / M)) * Math.exp((-
2 * r * T) / (M - 1));
     double sdErr = sdDev * Math.sqrt(M);
     double callValue = portfolioValue + geometricAsianOption(S, K, r, div, sig, T, N, nudt, dt);
     System.out.println("callValue: "+callValue);
     return callValue;
   }

   private double randPolarRejc() {
     double randmVar;
     double x1 = 0.0;
     double x2 = 0.0;
     double w = 0.0;
     double u1 = 0.0;
     double u2 = 0.0;
     boolean varAvailable = false;

     if (varAvailable) {
        varAvailable = false;
        randmVar = x2;
     } else {
        varAvailable = true;
        do {
          u1 = new Random(-1).nextDouble() * 2 - 1;
          //u1 = Math.random() * 2 - 1;  //Pick two uniform numbers in the square_ranging
                              // from -1 to +1 in each direction
          u2 = new Random(-1).nextDouble() * 2 - 1;
         // u2 = Math.random() * 2 - 1;
           w = u1 * u1 + u2 * u2;    // See if these u1 and u2 are in the unit_circle by letting w =
R^2
        } while (w >= 1 || w == 0);    //If they are not, we try again

        double c = Math.sqrt(-2 * Math.log(w) / w);
        //Box Muller transformation to get two normal
        // deviates (where we_return one and save the other for the next time)
        x1 = c * u1;
        x2 = c * u2;
        randmVar = x1;
     }
```

```
    return randmVar;
  }

  private double geometricAsianOption(double S, double K, double r, double div,
                        double sig, double T, double N, double nudt, double dt) {
    double optionValue = 0.0;
    double ga_v = r - div - 0.5 * Math.pow(sig, 2);
    double ga_a = Math.log(S) + nudt + 0.5 * ga_v * (T - dt);
    double ga_b = Math.pow(sig, 2) * dt + Math.pow(sig, 2) * (T - dt) * N * (2 * N - 1) / (6 *
Math.pow(N, 2));
    double ga_x = (ga_a - Math.log(K) + ga_b) / Math.sqrt(ga_b);
    optionValue = Math.exp(-r * T) * ((Math.exp(ga_a + 0.5 * ga_b) * CND(ga_x)) - K *
CND(ga_x - Math.sqrt(ga_b)));

    return optionValue;

  }

  private double CND(double x) {

    //NORMSDIST INTEGRAL VARIABLES
    double b1 = 0.319381530;
    double b2 = -0.356563782;
    double b3 = 1.781477937;
    double b4 = -1.821255978;
    double b5 = 1.330274429;
    double p = 0.2316419;

    double t = 0.0;
    double zx = 0.0;
    double px = 0.0;

    t = 1 / (1 + (p * x));
    zx = (1 / (Math.sqrt(2 * Math.PI))) * Math.exp(- Math.pow(x, 2) / 2);
    px = 1 - zx * (b1 * t + b2 * (Math.pow(t, 2)) + b3 * (Math.pow(t, 3)) + b4 * (Math.pow(t,
4)) + b5
         * (Math.pow(t, 5)));

    return px;
  }


  public static void main(String [] args) {

    //test initial parameters
     double S = 4493;    // Stock price
     double K = 4493;    // strike price
     double T = 0.24658;      // Maturity date
     double sig = 0.21097;  // Volatility
     double r = 0.04314;   // Interest rate
     double div = 0.0087; // Divident yield
     //double M = 100;    // Simulations
     //double N = 100;     // Time step

    //double S = 100;    // Stock price
    //double K = 100;    // strike price
    //double T = 1;      // Maturity date
    //double sig = 0.20;  // Volatility
    //double r = 0.05;   // Interest rate
    //double div = 0; // Divident yield

    ArithMonteCarloGeoContVar mcTest = new ArithMonteCarloGeoContVar(S, K, r, div, sig, T);
```

```
      System.out.println("call_value: " + mcTest.valueMonteCarlo);


   }
}


/**
 * Created by IntelliJ IDEA.
 * Author: Plamen Stilyianov
 * Title: MSc Financial Markets with Information Systems
 * Date: 01-Sep-2005
 * Time: 12:01:58
 * To change this template use File | Settings | File Templates.
 */
public class ArithmeticModels {

   //greeks
   public double delta, gamma, vega, rho, theta = 0.0;
   public double valueLevy, valueTurnbullWakeman = 0.0;

   public ArithmeticModels() {
   }

   public ArithmeticModels(double S, double K, double r, double div, double sig, double T) {

      valueTurnbullWakeman = modelTurnbullWakemanAsian(S, K, r, div, sig, T);
      valueLevy = modelLevyAsian(S, K, r, div, sig, T);
      delta = deltaArithmetic(S, K, T, r, div, sig);
      gamma = gammaArithmetic(S, K, T, r, div, sig);
      theta = thetaArithmetic(S, K, T, r, div, sig) / 365;
      vega = vegaArithmetic(S, K, T, r, div, sig) / 100;
      rho = rhoArithmetic(S, K, T, r, div, sig) / 100;
   }

   private double modelTurnbullWakemanAsian(double S, double K, double r, double div, double
sig, double T) {

      //constant
      double b = r - div;   // cost of carry rate
      double tau = 0.0;   // time to the begining of the average period

      double m1 = 0.0;
      double m2 = 0.0;

      double optionValue = 0.0;
      double x = 0.0;
      m1 = ((Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau))) * S;
      m2 = Math.pow(S, 2) * 2 * Math.exp((2 * b + Math.pow(sig, 2)) * T) / ((b + Math.pow(sig,
2))
            * (2 * b + Math.pow(sig, 2)) * Math.pow(T - tau, 2))
             + (Math.pow(S, 2) * 2 * Math.exp((2 * b + Math.pow(sig, 2)) * tau) / (b * Math.pow(T
- tau, 2)))
            * (1 / (2 * b + Math.pow(sig, 2)) - (Math.exp(b * (T - tau))) / (b + Math.pow(sig, 2)));
      System.out.println("M:" + m2);
      double modVol = Math.sqrt((1 / T) * Math.log(m2 / Math.pow(m1, 2))) * 100;

      optionValue = modelBlack76(K, m1, modVol, T, r);
      return optionValue;
   }
```

```
  private double modelBlack76(double strike, double m1, double modVol, double yrs, double r) {
    double optionPrice = 0.0;
    double d1 = 0.0;
    double d2 = 0.0;
    double nd1 = 0.0;
    double nd2 = 0.0;

    d1 = (Math.log(m1 / strike) + Math.pow((modVol / 100), 2) * (yrs / 2)) / ((modVol / 100)
* Math.sqrt(yrs));
    d2 = d1 - (modVol / 100) * Math.sqrt(yrs);

    nd1 = CND(d1);
    nd2 = CND(d2);

    optionPrice = Math.exp(-r * yrs) * (m1 * nd1 - strike * nd2);
    return optionPrice;
  }

  private double modelLevyAsian(double S, double K, double r, double div, double sig, double T)
{
    double valueLevy = 0.0;
    double se = 0.0;
    double m = 0.0;
    double d = 0.0;
    double sv = 0.0;
    double xStar = 0.0;
    double d1, d2 = 0.0;
    double b = r - div;   / / cost of carry rate
    double t2 = T;   / / remaining time to maturity
    double sa = S; / / Arithmetic Average price

    se = S / (T * b) * (Math.exp((b - r) * t2) - Math.exp(-r * t2));

    m = 2 * Math.pow(S, 2) / (b + Math.pow(sig, 2)) * ((Math.exp((2 * b + Math.pow(sig, 2))
                          * t2) - 1) / (2 * b + Math.pow(sig, 2)) - (Math.exp(b * t2) - 1) / b);
    d = m / Math.pow(T, 2);
    sv = Math.log(d) - 2 * (r * t2 + Math.log(se));
    xStar = K - ((T - t2) / T) * sa;
    d1 = 1 / Math.sqrt(sv) * (Math.log(d) / 2 - Math.log(xStar));
    d2 = d1 - Math.sqrt(sv);

    valueLevy = se * CND(d1) - xStar * Math.exp(-r * t2) * CND(d2);
    return valueLevy;
  }

  private double CND(double X) {
    double L, K, w;
    double a1 = 0.31938153, a2 = -0.356563782, a3 = 1.781477937, a4 = -1.821255978, a5 =
1.330274429;

    L = Math.abs(X);
    K = 1.0 / (1.0 + 0.2316419 * L);
    w = 1.0 - 1.0 / Math.sqrt(2.0 * Math.PI) * Math.exp(-L * L / 2) * (a1 * K + a2 * K * K + a3
                          * Math.pow(K, 3) + a4 * Math.pow(K, 4) + a5 * Math.pow(K, 5));

    if (X < 0.0) {
      w = 1.0 - w;
    }
    return w;
  }

  private double ND(double X) {
```

```
    double w;

    w = Math.exp(Math.pow(-X, 2) / 2) / Math.sqrt(2 * Math.PI);

    return w;
  }

  / / GREEKS


  private double deltaArithmetic(double S, double K, double T, double r, double div, double sig) {
    double deltaAr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1 = 0.0;

    m1 = (Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau));

    m2 = (2 * Math.exp((2 * b + Math.pow(sig, 2)) * T)) / ((b + Math.pow(sig, 2)) * (2 * b +
Math.pow(sig, 2))
        * Math.pow((T - tau), 2))
      + (2 * Math.exp((2 * b + Math.pow(sig, 2)) * tau) / (b * Math.pow((T - tau), 2)))
        * (1 / (2 * b + Math.pow(sig, 2)) - (Math.exp(b * (T - tau)) / (b + Math.pow(sig, 2))));

    System.out.println("M:" + m2);
    ba = Math.log(m1) / T;

    va = Math.sqrt(Math.log(m2) / T - 2 * ba);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));

    deltaAr = Math.exp((ba - r) * T) * CND(d1);

    return deltaAr;
  }

  private double gammaArithmetic(double S, double K, double T, double r, double div, double sig)
{
    double gammaAr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1 = 0.0;

    m1 = (Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau));

    m2 = 2 * Math.exp((2 * b + Math.pow(sig, 2)) * T) / ((b + Math.pow(sig, 2))
        * (2 * b + Math.pow(sig, 2)) * Math.pow((T - tau), 2))
      + 2 * Math.exp((2 * b + Math.pow(sig, 2)) * tau) / (b * Math.pow((T - tau), 2))
        * (1 / (2 * b + Math.pow(sig, 2)) - Math.exp(b * (T - tau)) / (b + Math.pow(sig, 2)));
    System.out.println("M:" + m2);
    ba = Math.log(m1) / T;

    va = Math.sqrt(Math.log(m2) / T - 2 * ba);
```

```
    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));

    gammaAr = Math.exp((ba - r) * T) * CND(d1) / (S * va * Math.sqrt(T));

    return gammaAr;
  }

  private double thetaArithmetic(double S, double K, double T, double r, double div, double sig) {
    double thetaAr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;

    m1 = (Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau));

    m2 = 2 * Math.exp((2 * b + Math.pow(sig, 2)) * T) / ((b + Math.pow(sig, 2)) * (2 * b +
Math.pow(sig, 2))
        * Math.pow((T - tau), 2))
      + 2 * Math.exp((2 * b + Math.pow(sig, 2)) * tau) / (b * Math.pow((T - tau), 2))
        * (1 / (2 * b + Math.pow(sig, 2)) - Math.exp(b * (T - tau)) / (b + Math.pow(sig, 2)));
    System.out.println("M:" + m2);
    ba = Math.log(m1) / T;

    va = Math.sqrt(Math.log(m2) / T - 2 * ba);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    d2 = d1 - (sig * Math.sqrt(T));

    thetaAr = -(S * Math.exp((ba - r) * T) * ND(d1) * va) / (2 * Math.sqrt(T)) - (r * K *
Math.exp(-r * T)
        * CND(d2)) - ((ba - r) * S * CND(d1) * Math.exp((ba - r) * T));
    return thetaAr;
  }

  private double vegaArithmetic(double S, double K, double T, double r, double div, double sig) {
    double vegaAr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;

    m1 = (Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau));

    m2 = 2 * Math.exp((2 * b + Math.pow(sig, 2)) * T) / ((b + Math.pow(sig, 2)) * (2 * b +
Math.pow(sig, 2))
        * Math.pow((T - tau), 2))
      + 2 * Math.exp((2 * b + Math.pow(sig, 2)) * tau) / (b * Math.pow((T - tau), 2))
        * (1 / (2 * b + Math.pow(sig, 2)) - Math.exp(b * (T - tau)) / (b + Math.pow(sig, 2)));
    System.out.println("M:" + m2);
    ba = Math.log(m1) / T;
    va = Math.sqrt(Math.log(m2) / T - 2 * ba);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    // d2 = d1 - (sig*Math.sqrt(T));
```

```
    vegaAr = S * Math.exp((ba - r) * T) * ND(d1) * Math.sqrt(T);
    return vegaAr;
  }

  private double rhoArithmetic(double S, double K, double T, double r, double div, double sig) {
    double rhoAr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;

    m1 = (Math.exp(b * T) - Math.exp(b * tau)) / (b * (T - tau));

    m2 = 2 * Math.exp((2 * b + Math.pow(sig, 2)) * T) / ((b + Math.pow(sig, 2)) * (2 * b +
Math.pow(sig, 2))
        * Math.pow((T - tau), 2)) + 2 * Math.exp((2 * b + Math.pow(sig, 2))
        * tau) / (b * Math.pow((T - tau), 2))
        * (1 / (2 * b + Math.pow(sig, 2)) - Math.exp(b * (T - tau)) / (b + Math.pow(sig, 2)));
    System.out.println("M:" + m2);

    ba = Math.log(m1) / T;
    va = Math.sqrt(Math.log(m2) / T - 2 * ba);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    d2 = d1 - (sig * Math.sqrt(T));

    if (b != 0)
      rhoAr = T * K * Math.exp(-r * T) * CND(d2);
    else {
      rhoAr = -T * new GenBlackScholesModels(S, K, r, div, sig, T).valueBlackScholes;
    }


    return rhoAr;
  }

  public static void main(String [] args) {

    //initial parameters
    double S = 4493;    // Stock price
    double K = 4493;    // strike price
    double T = 0.24658;     // Maturity date
    double sig = 0.21097;  // Volatility
    double r = 0.04314;   // Interest rate
    double div = 0.0087; // Divident yield

    //double S = 35.75;    // Stock price
    //double K = 35.75;    // strike price
    //double T = 0.24658;     // Maturity date
    //double sig = 0.26354;  // Volatility
    //double r = 0.04271;   // Interest rate
    //double div = 0.0457; // Divident yield

    ArithmeticModels arTest = new ArithmeticModels(S, K, r, div, sig, T);

    System.out.println("turnbullWakeman: " + arTest.valueTurnbullWakeman);
    System.out.println("levy: " + arTest.valueLevy);
    System.out.println("delta: " + arTest.delta);
    System.out.println("gamma: " + arTest.gamma);
```

```
      System.out.println("theta: " + arTest.theta);
      System.out.println("vega: " + arTest.vega);
      System.out.println("rho: " + arTest.rho);
    }
}


/**
 * Created by IntelliJ IDEA.
 * Author: Plamen Stilyianov
 * Title: MSc Financial Markets with Information Systems
 * Date: 01-Sep-2005
 * Time: 22:02:08
 * To change this template use File | Settings | File Templates.
 */
public class GeometricModels {

    //greeks
    public double delta, gamma, vega, rho, theta = 0.0;
    public double valueKemnaVorst, valueCurran = 0.0;

    private GeometricModels() {
    }

    public GeometricModels(double S, double K, double r, double div, double sig, double T) {
      valueKemnaVorst = modelKemnaVorstAsian(S, K, r, div, sig, T);
      valueCurran = modelCurranAsian(S, K, r, div, sig, T);
      delta = deltaGeomethric(S, K, T, r, div, sig);
      gamma = gammaGeomethric(S, K, T, r, div, sig);
      theta = thetaGeomethric(S, K, T, r, div, sig) / 365;
      vega = vegaGeomethric(S, K, T, r, div, sig) / 100;
      rho = rhoGeomethric(S, K, T, r, div, sig) / 100;
    }

    private double modelKemnaVorstAsian(double S, double K, double r, double div, double sig,
double T) {
      double valueKemnaVorst = 0.0;

      double b = r - div;   // cost of carry rate
      double siga = 0.0;    // adjusted volatility
      double d1, d2 = 0.0;
      double t = 0.0;

      b = 0.5 * (r - div - Math.pow(sig, 2) / 6);

      siga = sig / Math.sqrt(3);

      d1 = (Math.log(S / K) + (b + 0.5 * Math.pow(siga, 2)) * T) / (siga * Math.sqrt(T));

      d2 = (Math.log(S / K) + (b - 0.5 * Math.pow(siga, 2)) * T) / (siga * Math.sqrt(T));

      valueKemnaVorst = S * Math.exp((b - r) * (T - t)) * CND(d1) - K * Math.exp(-r * (T - t)) *
CND(d2);

      return valueKemnaVorst;
    }

    private double modelCurranAsian(double S, double K, double r, double div, double sig, double
T) {

      double n = 260;
      double t1 = 0;
```

```
    double dt = T / n;
    double mu = Math.log(S) + ((r - div) - (0.5 * Math.pow(sig, 2))) * (t1 * dt + (n - 1) * 0.5 *
dt);
    double vx = sig * Math.sqrt(t1 * dt + dt * (n - 1) * (2 * n - 1) / (6 * n));

    double curran = 0.0;
    double K1 = 0.0;
    double d1 = 0.0;
    double[] ti = new double[(int) n + 1];
    double[] vi = new double[(int) n + 1];
    double[] mui = new double[(int) n + 1];
    double[] vxi = new double[(int) n + 1];

    for (int i = 1; i <= n; i++) {
      ti[i] = (i + t1) * dt;
      vi[i] = Math.sqrt(sig * sig * (t1 * dt + (i - 1) * dt));
      mui[i] = Math.log(S) + (r - div - 0.5 * sig * sig) * ti[i];
      vxi[i] = sig * sig * (t1 * dt + dt * ((i - 1) - i * (i - 1) / (2 * n)));
    }

    double summ2 = 0.0;

    for (int i = 1; i <= n; i++) {
      summ2 = summ2 + (1 / n) * Math.exp(mui[i] + (vxi[i] / (Math.pow(vx, 2))) *
(Math.log(K) - mu)
          + 0.5 * (Math.pow(vx, 2) + (Math.pow(vxi[i], 2) / Math.pow(vx, 2))));
    }

    K1 = 2 * K - summ2;
    d1 = (mu - Math.log(K1)) / vx;

    double summ1 = 0;

    for (int i = 1; i <= n; i++) {
      summ1 = summ1 + Math.exp(mui[i] + 0.5 * (Math.pow(vi[i], 2))) * CND(d1 + vxi[i] /
vx);
    }

    curran = Math.exp(-r * T) * ((1 / n) * summ1 - K * CND(d1));
    return curran;
  }

  private double CND(double X) {
    double L, K, w;
    double a1 = 0.31938153, a2 = -0.356563782, a3 = 1.781477937, a4 = -1.821255978, a5 =
1.330274429;

    L = Math.abs(X);
    K = 1.0 / (1.0 + 0.2316419 * L);
    w = 1.0 - 1.0 / Math.sqrt(2.0 * Math.PI) * Math.exp(-L * L / 2) * (a1 * K + a2 * K * K + a3 *
Math.pow(K, 3)
        + a4 * Math.pow(K, 4) + a5 * Math.pow(K, 5));

    if (X < 0.0) {
      w = 1.0 - w;
    }
    return w;
  }

  private double ND(double X) {
    double w;
```

```
    w = Math.exp(Math.pow(-X, 2) / 2) / Math.sqrt(2 * Math.PI);

    return w;
  }

  / / GREEKS

  private double deltaGeomethric(double S, double K, double T, double r, double div, double sig)
{
    double deltaGr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1 = 0.0;


    ba = 0.5 * (b + Math.pow(sig, 2) / 6);
    va = sig / Math.sqrt(3);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));

    deltaGr = Math.exp((ba - r) * T) * CND(d1);

    return deltaGr;
  }

  private double gammaGeomethric(double S, double K, double T, double r, double div, double
sig) {
    double gammaGr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1 = 0.0;


    ba = 0.5 * (b + Math.pow(sig, 2) / 6);
    va = sig / Math.sqrt(3);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));

    gammaGr = Math.exp((ba - r) * T) * CND(d1) / (S * va * Math.sqrt(T));

    return gammaGr;
  }

  private double thetaGeomethric(double S, double K, double T, double r, double div, double sig)
{
    double thetaGr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;
```

```
    ba = 0.5 * (b + Math.pow(sig, 2) / 6);
    va = sig / Math.sqrt(3);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    d2 = d1 - (sig * Math.sqrt(T));

    thetaGr = -(S * Math.exp((ba - r) * T) * ND(d1) * va) / (2 * Math.sqrt(T)) - ((ba - r) * S *
CND(d1)
                        * Math.exp((ba - r) * T)) - (r * K * Math.exp(-r * T) * CND(d2));

    return thetaGr;
  }

  private double vegaGeomethric(double S, double K, double T, double r, double div, double sig)
{
    double vegaGr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;


    ba = 0.5 * (b + Math.pow(sig, 2) / 6);
    va = sig / Math.sqrt(3);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    // d2 = d1 - (sig*Math.sqrt(T));

    vegaGr = S * Math.exp((ba - r) * T) * ND(d1) * Math.sqrt(T);

    return vegaGr;
  }

  private double rhoGeomethric(double S, double K, double T, double r, double div, double sig) {
    double rhoGr = 0.0;
    double m1 = 0.0;
    double m2 = 0.0;
    double ba = 0.0;
    double b = r - div;
    double tau = 0.0;
    double va = 0.0;
    double d1, d2 = 0.0;


    ba = 0.5 * (b + Math.pow(sig, 2) / 6);
    va = sig / Math.sqrt(3);

    d1 = (Math.log(S / K) + (ba + Math.pow(va, 2) / 2) * T) / (va * Math.sqrt(T));
    d2 = d1 - (sig * Math.sqrt(T));

    if (b != 0)
      rhoGr = T * K * Math.exp(-r * T) * CND(d2);
    else {
      rhoGr = -T * new GenBlackScholesModels(S, K, r, div, sig, T).valueBlackScholes;
    }

    return rhoGr;
  }
```

```
    public static void main(String [] args) {

        //initial parameters
        //   double S = 4493;    // Stock price
        //   double K = 4493;    // strike price
        //   double T = 0.24658;     // Maturity date
        //   double sig = 0.21097;  // Volatility
        //   double r = 0.04314;   // Interest rate
        //   double div = 0.0087; // Divident yield

        double S = 35.75;    // Stock price
        double K = 35.75;    // strike price
        double T = 0.24658;     // Maturity date
        double sig = 0.26354;  // Volatility
        double r = 0.04271;   // Interest rate
        double div = 0.0457; // Divident yield

        GeometricModels gmTest = new GeometricModels(S, K, r, div, sig, T);

        System.out.println("curranOptionValue: " + gmTest.valueCurran);
        System.out.println("kemnaVorstOptionValue: " + gmTest.valueKemnaVorst);
        System.out.println("geometricDelta: " + gmTest.delta);
        System.out.println("geometricGamma: " + gmTest.gamma);
        System.out.println("geometricTheta: " + gmTest.theta);
        System.out.println("geometricVega: " + gmTest.vega);
        System.out.println("geometricRho: " + gmTest.rho);
    }
}


/**
 * Created by IntelliJ IDEA.
 * Author: Plamen Stilyianov
 * Title: MSc Financial Markets with Information Systems
 * Date: 02-Sep-2005
 * Time: 00:10:17
 * To change this template use File | Settings | File Templates.
 */
public class GenBlackScholesModels {

    //greeks
    public double delta, gamma, vega, rho, theta = 0.0;
    public double valueBlackScholes = 0.0;

    private GenBlackScholesModels() {
    }

    public GenBlackScholesModels(double S, double K, double r, double div, double sig, double T) {
        valueBlackScholes = modelGenBlackScholesAsian(S, K, r, div, sig, T);
        delta = deltaGeBlackScholes(S, K, T, r, div, sig);
        gamma = gammaGeBlackScholes(S, K, T, r, div, sig);
        theta = thetaGeBlackScholes(S, K, T, r, div, sig) / 365;
        vega = vegaGeBlackScholes(S, K, T, r, div, sig) / 100;
        rho = rhoGeBlackScholes(S, K, T, r, div, sig) / 100;
    }

    private double modelGenBlackScholesAsian(double S, double K, double r, double div, double
sig, double T) {
        double valueBS = 0.0;
        double d1, d2 = 0.0;
```

```
      double b = r - div;

      d1 = (Math.log(S / K) + (b + 0.5 * Math.pow(sig, 2)) * T) / (sig * Math.sqrt(T));
      d2 = d1 - (sig * Math.sqrt(T));

      valueBS = (S * Math.exp((b - r) * T) * CND(d1)) - (K * Math.exp(-r * T) * CND(d2));

      return valueBS;
   }

   private double CND(double X) {
      double L, K, w;
      double a1 = 0.31938153, a2 = -0.356563782, a3 = 1.781477937, a4 = -1.821255978, a5 =
1.330274429;

      L = Math.abs(X);
      K = 1.0 / (1.0 + 0.2316419 * L);
      w = 1.0 - 1.0 / Math.sqrt(2.0 * Math.PI) * Math.exp(-L * L / 2) * (a1 * K + a2 * K * K + a3 *
Math.pow(K, 3)
            + a4 * Math.pow(K, 4) + a5 * Math.pow(K, 5));

      if (X < 0.0) {
         w = 1.0 - w;
      }
      return w;
   }

   private double ND(double X) {
      double w;

      w = Math.exp(Math.pow(-X, 2) / 2) / Math.sqrt(2 * Math.PI);

      return w;
   }

   / / GREEKS

   private double deltaGeBlackScholes(double S, double K, double T, double r, double div, double
sig) {
      double deltaGeBS = 0.0;
      double d1 = 0.0;
      double b = r - div;

      d1 = (Math.log(S / K) + (b + Math.pow(sig, 2) / 2) * T) / (sig * Math.sqrt(T));

      deltaGeBS = Math.exp((b - r) * T) * CND(d1);

      return deltaGeBS;
   }

   private double gammaGeBlackScholes(double S, double K, double T, double r, double div,
double sig) {
      double gammaGeBS = 0.0;
      double d1 = 0.0;
      double b = r - div;

      d1 = (Math.log(S / K) + (b + Math.pow(sig, 2) / 2) * T) / (sig * Math.sqrt(T));

      gammaGeBS = Math.exp((b - r) * T) * CND(d1) / (S * sig * Math.sqrt(T));

      return gammaGeBS;
   }
```

```
   private double thetaGeBlackScholes(double S, double K, double T, double r, double div, double
sig) {
      double thetaGeBS = 0.0;
      double d1, d2 = 0.0;
      double b = r - div;

      d1 = (Math.log(S / K) + (b + Math.pow(sig, 2) / 2) * T) / (sig * Math.sqrt(T));
      d2 = d1 - (sig * Math.sqrt(T));

      thetaGeBS = -(S * Math.exp((b - r) * T) * ND(d1) * sig) / (2 * Math.sqrt(T)) - (r * K *
Math.exp(-r * T)
                                  * CND(d2))- ((b - r) * S * CND(d1) * Math.exp((b - r) * T));
      //thetaGeBS = -(sig*S*ND(d1)*Math.exp(-div*T)) / (2*Math.sqrt(T)) -
(r*K*CND(d2)*Math.exp(-r*T))
      // + (div*S*CND(d1)*Math.exp(-div*T));

      return thetaGeBS;
   }

   private double vegaGeBlackScholes(double S, double K, double T, double r, double div, double
sig) {
      double vegaGr = 0.0;
      double d1 = 0.0;
      double b = r - div;

      d1 = (Math.log(S / K) + (b + Math.pow(sig, 2) / 2) * T) / (sig * Math.sqrt(T));

      vegaGr = S * Math.exp((b - r) * T) * ND(d1) * Math.sqrt(T);

      return vegaGr;
   }

   private double rhoGeBlackScholes(double S, double K, double T, double r, double div, double
sig) {
      double rhoGeBS = 0.0;
      double d1, d2 = 0.0;
      double b = r - div;

      d1 = (Math.log(S / K) + (b + Math.pow(sig, 2) / 2) * T) / (sig * Math.sqrt(T));
      d2 = d1 - (sig * Math.sqrt(T));

      if (b != 0)
         rhoGeBS = T * K * Math.exp(-r * T) * CND(d2);
      else
         rhoGeBS = -T * modelGenBlackScholesAsian(S, K, r, div, sig, T);
      return rhoGeBS;
   }

   public static void main(String [] args) {

      //initial parameters
      // double S = 4493;    // Stock price
      // double K = 4493;    // strike price
      // double T = 0.24658;     // Maturity date
      // double sig = 0.21097;  // Volatility
      // double r = 0.04314;   // Interest rate
      //  double div = 0.0087; // Divident yield

      double S = 35.75;    // Stock price
      double K = 35.75;    // strike price
      double T = 0.24658;     // Maturity date
```

```
        double sig = 0.26354;  // Volatility
        double r = 0.04271;   // Interest rate
        double div = 0.0457; // Divident yield

        GenBlackScholesModels bsTest = new GenBlackScholesModels(S, K, r, div, sig, T);

        System.out.println("blackScholesOptionValue: " + bsTest.valueBlackScholes);
        System.out.println("Black Scholes delta: " + bsTest.delta);
        System.out.println("Black Scholes gamma: " + bsTest.gamma);
        System.out.println("Black Scholes theta: " + bsTest.theta);
        System.out.println("Black Scholes vega: " + bsTest.vega);
        System.out.println("Black Scholes rho: " + bsTest.rho);
    }
}
```