

Financial Economics II
(FIN30160)
Dr. Yajun Xiao

Aaron Davy – 13311866
Alison Kelly – 13314261
Alison Smith – 13364011
Luke Killoran – 13434418

Table of Contents

1 Forward and Futures Pricing	2
1.1 Introduction	2
1.2 Pricing of a Forward	2
1.3 Equivalence of Forward and Futures Contracts	2
1.4 Scenarios under which Forward and Future Prices Differ	5
1.5 Forward and Future Prices under Stochastic Interest Rates	6
2 Black-Scholes Option Pricing	9
2.1 Introduction	9
2.2 Black-Scholes Pricing Model	9
2.3 The Greeks	11
2.4 Delta Hedging	13
2.5 Delta-Vega Hedging	19
3 Numerical Methods for Option Pricing	24
3.1 Binomial Model	24
3.2 Monte Carlo Simulation	25
4 Hedging Options in Discrete Time	28
4.1 Binomial Model Put Option Replication	28
4.2 Monte Carlo Simulation and Black-Scholes Pricing	29
4.3 Augmented Geometric Brownian Motion for Put Option Replication	32
Appendix A	45
Appendix B	64
References	64

Forward and Future Pricing

1.1 Introduction

Forward and future agreements are valuable to end users of products as there is no initial investment and it allows them to hedge their positions and minimise interest rate risks. If one has particular expertise, profits may be made speculating on the future price movements of underlying assets. However, there is the disadvantage that losses are often not constrained. Forwards are “over the counter” instruments synthesised by investment banks, and are generally non-standardised contracts. Futures are exchange traded contracts subject to much lower credit risk due to the Clearing House which manages credit risk exposure through demanding initial and variation margins calculated by marking-to-market.

1.2 Pricing of a Forward

While, in practice arbitrage opportunities do appear, they are transient in nature and so it is prudent when pricing, to assume they do not exist. Assuming this, and that markets behave perfectly with no transaction costs, the fair forward price F is $F = \frac{S}{d(0,T)}$. Where S is the underlying spot price and the discount $d(0,T)$ is used to find the forward price for delivery at time T .

Commonly, forwards are priced using risk-neutral pricing. This assumes that the asset price grows at the risk-free rate r_f regardless of the risk of the asset.

r_f is constant and continuously compounded, therefore:

$$F = S_0 e^{r_f T}$$

$$F = 100e^{0.02(1)}$$

$$F = \text{€}102.02$$

1.3 Equivalence of Forward and Futures Contracts

Forwards, as above, are similar to future contracts (they are both agreements between two parties to buy and sell assets at a specified price) with key differences. Futures are marked-to-market weekly (which inhibits the accumulation of losses past the point of affordability) compared to forward contracts settled at the end of the one year period. Future contract prices are determined by supply and demand, compared to forwards which are privately negotiated.

If interest rates are constant for all maturities (with assumptions as above), there should be no difference between the value of a futures contract and the equivalent forward contract.

For a stock whose returns follow Geometric Brownian Motion (GBM) such that $dS = \mu S dt + \sigma S dW$, we find using Itô's Lemma that $(dz_t = \varepsilon_t \sqrt{dt} \sim N(0, dt), \text{ and } \varepsilon_t \sim N(0,1))$:

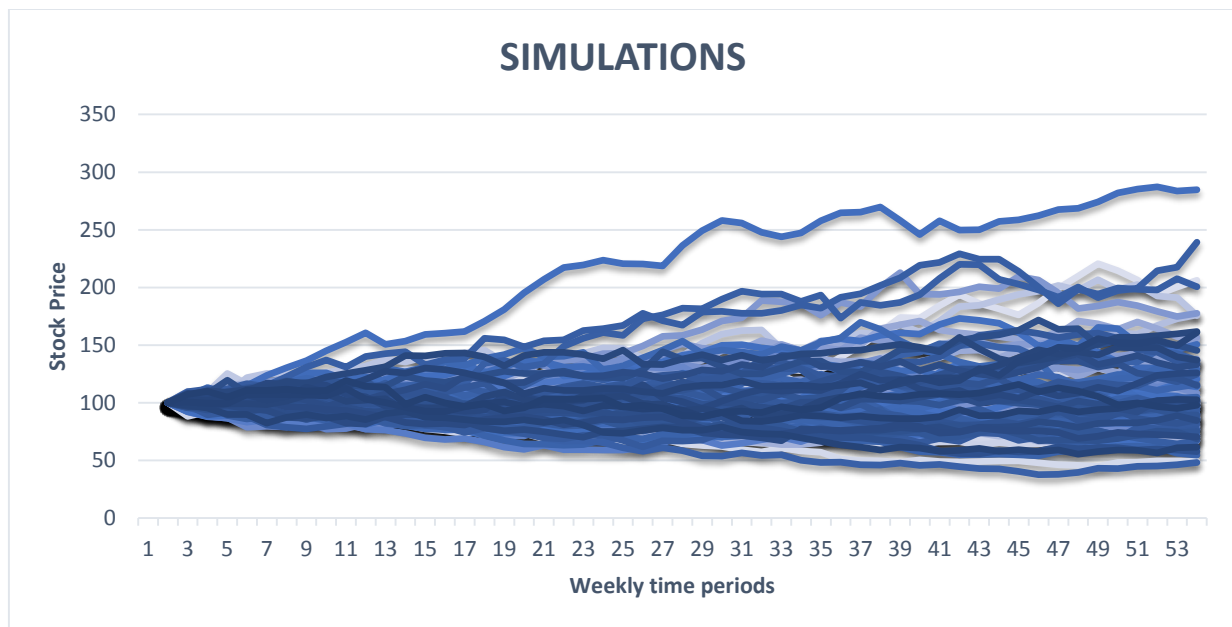
$$d \log(S) = \left[0 + \mu S \left(\frac{1}{S} \right) + \frac{1}{2} \sigma^2 S^2 \left(\frac{-1}{S^2} \right) \right] dt + \sigma S \left(\frac{1}{S} \right) dz = \left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW$$

$$\text{such that } \int_0^t d \log(S_u) = \int_0^t \left(\mu - \frac{1}{2} \sigma^2 \right) du + \int_0^t \sigma dW_u$$

$$\log(S_t) = \log(S_0) + \left(\mu - \frac{1}{2} \sigma^2 \right) t + \sigma W_t$$

$$\text{Let } v = \left(\mu - \frac{1}{2} \sigma^2 \right) \text{ then } \log(S_t) \sim N(\log(S_0) + vt, \sigma^2 t)$$

Using the formula $S_{k+1} = S_k e^{v \Delta t + \sigma \varepsilon_k \sqrt{\Delta t}} = S_k e^{v \Delta t + \sigma \varepsilon_k \sqrt{\Delta t}}$, where $\varepsilon_k \sim N(0,1)$. 100 weekly stock prices were simulated to produce the graph below (x-axis lagged by 1 weekly period). Real world weekly drift = 0.1346%, and weekly volatility = 3.47%.



To illustrate the equivalence of the forward and futures contract we set up two portfolios. Portfolio A involved going long $d(i, 1)$ futures on the underlying stock (with maturity at the end of the year) at time i , for $i = \frac{1}{52}, \dots, \frac{51}{52}, 1$. Therefore profit from this portfolio at time 1 would be

$$\sum_{i=\frac{1}{52}}^1 (F_i - F_{i-1}) e^{-r(1-i)} \prod_{j=\frac{1}{52}}^{1-i} e^{r_j \frac{1}{52}}$$

(r_j is the simulated interest rate at time j – held constant for the following week)

When settling a futures contract by marking-to-market, the difference in price at week k is $F_{\frac{k}{52}} - F_{\frac{k-1}{52}}$. This difference when positive (or negative) is placed on deposit (or financed through extra money in a margin account borrowed at the risk-free rate) until time $T = 1$ when the contract ends. This portfolio can be thought of as investing the value of the discounted change in margin account at each week, which is in essence the profit from a future.

Portfolio B simply entailed longing one forward contract with maturity 1 on the underlying stock. The profit of portfolio B at time of maturity is $S_1 - F_0$.

To prove equivalence when interest rates are constant, profit from portfolio A simplifies to equal profit of portfolio B:

$$\begin{aligned} & (F_{\frac{1}{52}} - F_0) + (F_{\frac{2}{52}} - F_{\frac{1}{52}}) + \dots + (F_{\frac{52}{52}} - F_{\frac{51}{52}}) \\ &= (F_{\frac{52}{52}} - F_0) = (S_{\frac{52}{52}} - F_0) \end{aligned}$$

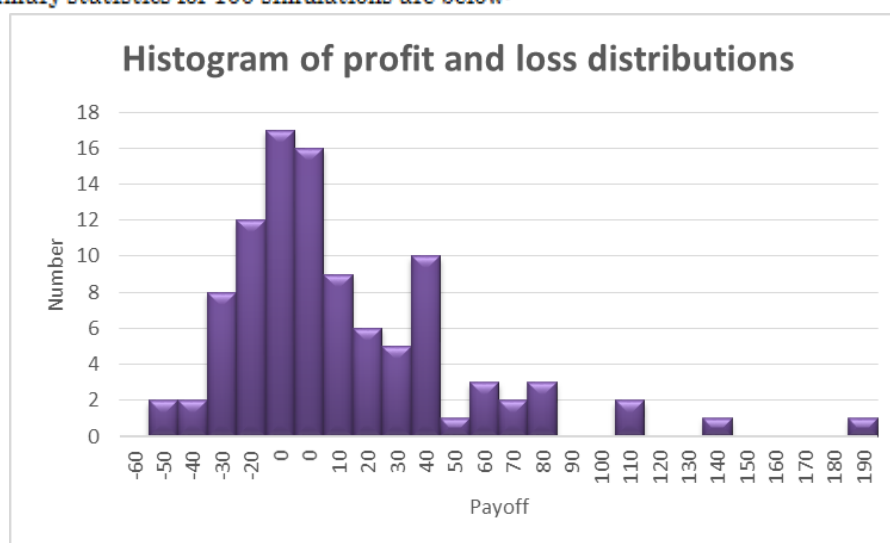
The simulated example below demonstrates this. We have weekly stock price simulated previously. $F_t = S_{(t)}e^{(1-t)}$ yields the third column of future values, where t is weekly time increments ($\frac{1}{52}$) summing to 1 full year period. At each marked-to-market period the future holder invests this discounted profit (or loss) at the fixed interest rate until time to maturity. It is shown that the total value of both portfolios at $T = 1$ is -\$8.297. Hence, the only difference between forward and futures contracts is nature of cash-flows, with forwards having a single cash flow at delivery and futures having a cash flow at the end of each weekly period.

Futures and forward cashflows

r		2%	Constant interest rates			
T		1				
t	S	F		Futures holder receives at time t	Futures holder invests cashflow at 2% to T	Forward Holder
0	100.000	102.020	F0			
0.02	109.501	111.671	F1	9.463	9.650	9.650
0.04	111.875	114.047	F2	2.331	2.377	
0.06	112.030	114.161	F3	0.112	0.114	
0.08	116.513	118.684	F4	4.441	4.523	
0.10	113.984	116.063	F5	-2.575	-2.622	
0.12	114.208	116.247	F6	0.181	0.184	
0.13	111.821	113.773	F7	-2.431	-2.473	
0.15	113.357	115.292	F8	1.493	1.519	
0.17	114.848	116.763	F9	1.447	1.471	
0.19	109.818	111.607	F10	-5.074	-5.156	
0.21	117.022	118.881	F11	7.161	7.275	
0.23	104.832	106.457	F12	-12.235	-12.424	
0.25	113.128	114.838	F13	8.256	8.380	
0.27	110.595	112.223	F14	-2.576	-2.614	
0.29	114.051	115.685	F15	3.413	3.462	
0.31	109.050	110.571	F16	-5.044	-5.115	
0.33	108.905	110.381	F17	-0.187	-0.190	
0.35	114.717	116.227	F18	5.770	5.846	
0.37	123.662	125.241	F19	8.901	9.014	
0.38	127.088	128.662	F20	3.379	3.421	
0.40	123.267	124.745	F21	-3.871	-3.917	
0.42	119.729	121.118	F22	-3.586	-3.627	
0.44	124.192	125.585	F23	4.417	4.467	
0.46	114.840	116.084	F24	-9.400	-9.501	
0.48	118.963	120.204	F25	4.078	4.121	
0.50	112.890	114.025	F26	-6.118	-6.180	
0.52	118.689	119.836	F27	5.756	5.811	
0.54	124.702	125.858	F28	5.967	6.022	
0.56	117.596	118.641	F29	-7.154	-7.217	
0.58	116.773	117.766	F30	-0.868	-0.875	
0.60	115.494	116.431	F31	-1.324	-1.335	
0.62	111.999	112.864	F32	-3.540	-3.567	
0.63	119.441	120.317	F33	7.399	7.453	
0.65	117.592	118.409	F34	-1.895	-1.908	
0.67	121.331	122.127	F35	3.694	3.718	
0.69	117.912	118.639	F36	-3.467	-3.488	
0.71	114.542	115.205	F37	-3.415	-3.435	
0.73	108.992	109.580	F38	-5.594	-5.625	
0.75	112.506	113.070	F39	3.473	3.490	
0.77	109.386	109.893	F40	-3.163	-3.178	
0.79	107.370	107.825	F41	-2.058	-2.067	

0.81	104.230	104.632	F42	-3.181	-3.193	
0.83	101.512	101.864	F43	-2.759	-2.768	
0.85	98.511	98.814	F44	-3.040	-3.049	
0.87	98.915	99.182	F45	0.366	0.367	
0.88	106.862	107.109	F46	7.909	7.927	
0.90	106.787	106.992	F47	-0.116	-0.117	
0.92	99.574	99.727	F48	-7.254	-7.265	
0.94	103.904	104.024	F49	4.292	4.297	
0.96	99.470	99.546	F50	-4.475	-4.478	
0.98	98.350	98.388	F51	-1.158	-1.158	
1.00	93.723	93.723	F52	-4.665	-4.665	-8.297
Total				-8.297	-8.297	

A histogram and summary statistics for 100 simulations are below:



Summary Statistics	Value
Median	-3.177394548
Mean	6.187770149
Minimum total value	-51.77315401
Maximum total value	184.8692879
Skew	1.735804057

Based on simulations, on average, if holder was to long the contract they would expect to accumulate a total value of €6.19 for the year. This holds for either the future or forward under equivalence and assumptions. Median value is less than mean value, which validates positive skew, and leads us to conclude a right skew distribution. This is expected from the positive drift of the GBM. As we are considering a positively skewed distribution, in general, more holders of the contracts will receive a value which is less than the mean, and in fact a negative number based on our median.

1.4 Scenarios under which Forwards and Future prices differ

When interest rates vary unpredictably and do not follow the Expectations Hypothesis, forward prices and futures prices do not necessarily equate. If spot prices and interest rates move in opposite directions (negative correlation), interest rate risk will make futures prices greater than forward prices. The opposite happens with movement in sync. This occurs because of the borrowing and lending rates that have to be made for intermediate profits and losses on a futures contract when marked-to-market.

Margin requirements are beneficial as they limit the maximum exposure to loss for futures, but those requirements, as well as price limits on futures can cause dissimilarities in valuations between the two

contracts. Existence of price limits might decrease volatility in prices making futures contracts more valuable, but they also makes futures contracts less liquid and this may make them less valuable.

Credit risk can also be a contributor to deviations between prices. Futures are standardised and traded on exchanges who essentially guarantee the contracts. It can be argued that the significantly higher default risk associated with non-standardised forward contracts should be taken into account when valuing the contract, making forwards relatively less valuable.

1.5 Forward and Futures Pricing - Stochastic Interest Rates

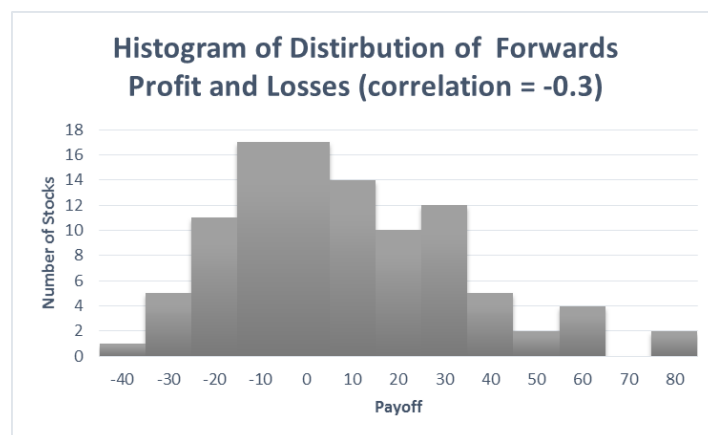
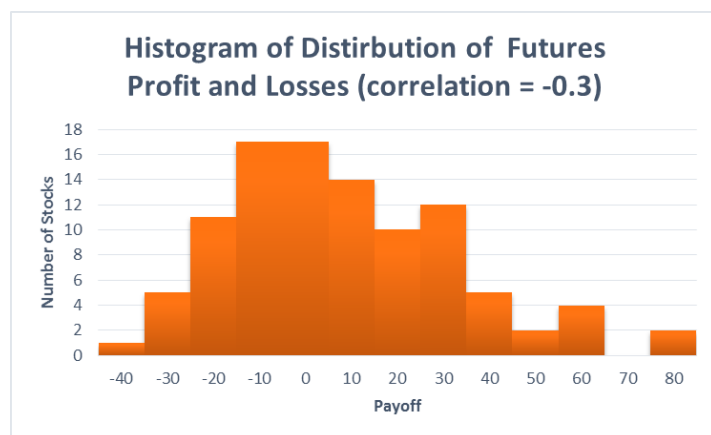
To investigate equivalence under a stochastic interest rate, we adopted the Cox–Ingersoll–Ross model, due to its simple formulation and extensive use in finance. An advantage of the Cox–Ingersoll–Ross model, is that it captures mean reversion (one of the first to do so), which is an essential characteristic of the interest rate that sets it apart from other financial instruments. This is because high interest rates tend to cause the economy to slow down and borrowers require less funds, which causes the rates to decline to the equilibrium long-term mean. When interest rates are low, funds are of high demand on the part of the borrowers so rates tend to increase again towards the long-term mean. As a result, interest rates move in a limited range, showing a tendency to revert to a long-run value. This model also, unlike the Vasicek-model, prevents the interest rate becoming negative – an undesirable feature for nominal interest rates under pre-crisis assumptions. Many other models boast this property such as the exponential Vasicek-model, Black–Derman–Toy-model and Black–Karasinski-model. However, using the Euler discretization approximation, it is possible for interest rates to become negative.

The model uses a correlation = -0.3 between interest rates and price of the underlying asset. A decreased stock price is associated with an increased interest rate. As interest rates rise, saving becomes relatively more attractive than spending and consumers demand for stocks lowers¹, hence negative correlation. Debt strain on companies also rises when interest rates fall which decreases value of future cashflows reducing demand for stocks, hence a fall in price. This scenario would mean that the value accruing when marked-to-market would be invested at a lower rate than if the interest rates were constant.

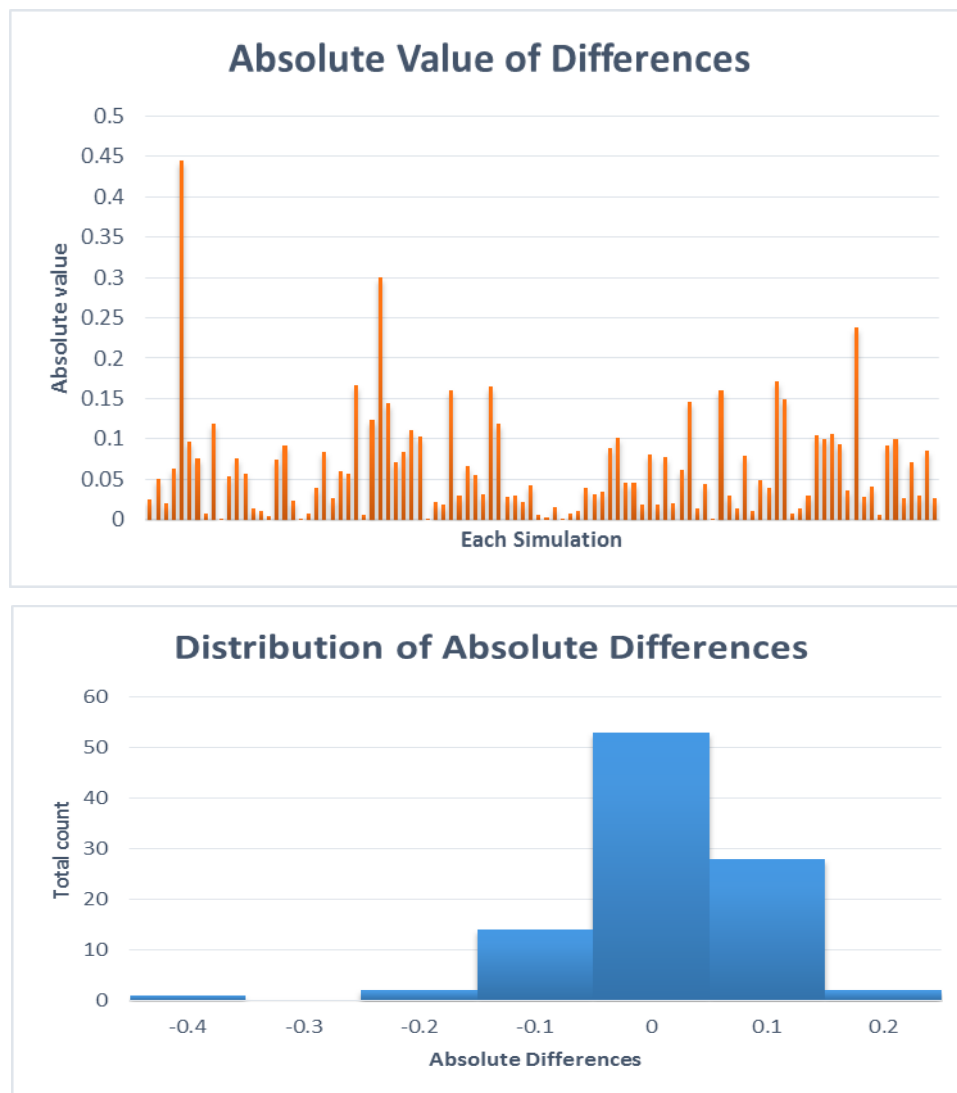
We had to choose seemingly arbitrary coefficients since we could not avail of published estimates of current market values, due to the constraint of long term interest rate being 0.02, which many papers did not have.

Below our summary statistics comparing forward and future prices, and histograms of the 100 simulated profits for each under these assumptions:

Summary Statistics	Future Value (-0.3 correlation)	Forward Value
Median	-2.55559	-2.5161
Mean	3.125167	3.168328
Minimum total value	-43.3011	-43.2831
Maximum total value	76.08206	76.24302
Skew	0.657512	0.660426



¹ <http://www.investopedia.com/investing/how-interest-rates-affect-stock-market/>



With the aid of the histograms of payoffs it is clear to see both contracts exhibit very similar statistical and distributional properties. The largest absolute difference = 0.44492. While the payoffs from forward and futures contracts are not theoretically equal (when removing the deterministic interest rate assumption) they are extremely similar, which is why theoretically they can be assumed to be equivalent.

We illustrated an example on the next page in order to contrast with the case of a deterministic interest rate. We implemented the same portfolios as before but in the case r_j , the simulated interest rate at time j , varied stochastically. In this particular case, the profit from the futures and forwards contracts were -\$7.825 and -\$7.8 respectively – a difference = -\$0.025.

Futures and forward cashflows

risk free		2%					
delta t		0.01923		Interest rates are stochastic and have a negative correlation with the underlying spot price			
T		1			Futures holder	Futures holder invests	
t	S	F	r	1/d(t,T)	receives at time t	cashflow to T	Forward Holder
0	100	102.020	0.02	1.000384689			
0.0192308	99.13745	101.101	0.020477	1.000393866	-0.901	-0.918	
0.0384615	104.20436	106.228	0.0159152	1.000306108	5.026	5.117	
0.0576923	105.54186	107.550	0.0171162	1.000329213	1.302	1.326	
0.0769231	99.69568	101.553	0.0173886	1.000334451	-5.902	-6.005	
0.0961538	98.66776	100.468	0.0171335	1.000329545	-1.069	-1.087	
0.1153846	93.75542	95.429	0.0185502	1.000356798	-4.963	-5.046	
0.1346154	98.37701	100.095	0.0193729	1.000372624	4.591	4.667	
0.1538462	98.70652	100.391	0.0235483	1.000452955	0.292	0.296	
0.1730769	102.60365	104.315	0.0249082	1.000479119	3.848	3.908	
0.1923077	103.42132	105.106	0.026441	1.000508611	0.775	0.787	
0.2115385	100.42595	102.022	0.026708	1.000513747	-3.020	-3.064	
0.2307692	100.1004	101.652	0.0243324	1.000468040	-0.362	-0.367	
0.25	100.1808	101.695	0.0205559	1.000395383	0.042	0.042	
0.2692308	97.01822	98.447	0.0223151	1.000429229	-3.200	-3.242	
0.2884615	98.68861	100.103	0.0212023	1.000407820	1.630	1.651	
0.3076923	99.9264	101.320	0.0200438	1.000385533	1.199	1.214	
0.3269231	100.27777	101.637	0.018233	1.000350696	0.313	0.317	
0.3461538	102.45915	103.808	0.0171789	1.000330418	2.145	2.170	
0.3653846	106.66511	108.028	0.0189129	1.000363776	4.174	4.221	
0.3846154	102.15121	103.416	0.0194553	1.000374211	-4.558	-4.607	
0.4038462	108.69422	109.998	0.0203474	1.000391373	6.506	6.574	
0.4230769	101.68823	102.868	0.0188692	1.000362935	-7.046	-7.117	
0.4423077	99.34947	100.464	0.0205872	1.000395986	-2.379	-2.403	
0.4615385	103.73015	104.853	0.0202612	1.000389714	4.341	4.382	
0.4807692	104.12256	105.209	0.0202867	1.000390204	0.352	0.356	
0.5	106.05383	107.120	0.0185536	1.000356863	1.891	1.907	
0.5192308	98.61023	99.563	0.020091	1.000386439	-7.490	-7.551	
0.5384615	98.77028	99.686	0.0206635	1.000397454	0.122	0.123	
0.5576923	104.13669	105.062	0.0213816	1.000411268	5.327	5.366	
0.5769231	103.23984	104.117	0.0195983	1.000376962	-0.936	-0.943	
0.5961538	102.17904	103.008	0.0170115	1.000327197	-1.101	-1.108	
0.6153846	95.23163	95.967	0.0177205	1.000340836	-6.995	-7.039	
0.6346154	90.74526	91.411	0.0204993	1.000394295	-4.527	-4.554	
0.6538462	89.72583	90.349	0.0195951	1.000376900	-1.054	-1.060	
0.6730769	93.86594	94.482	0.0166089	1.000319453	4.106	4.127	
0.6923077	94.98573	95.572	0.0169531	1.000326073	1.085	1.090	
0.7115385	96.17376	96.730	0.0164353	1.000316113	1.153	1.158	

0.7307692	98.53274	99.065	0.0167512	1.000322190	2.324	2.334
0.75	98.03806	98.529	0.0146532	1.000281832	-0.533	-0.535
0.7692308	100.25512	100.719	0.016199	1.000311568	2.182	2.190
0.7884615	101.08799	101.517	0.0174193	1.000335043	0.795	0.798
0.8076923	101.27217	101.662	0.0158751	1.000305337	0.145	0.146
0.8269231	99.46816	99.813	0.0150976	1.000290381	-1.844	-1.849
0.8461538	96.33316	96.630	0.0153191	1.000294641	-3.176	-3.183
0.8653846	96.84672	97.108	0.013372	1.000257188	0.477	0.478
0.8846154	92.48728	92.701	0.0132005	1.000253889	-4.400	-4.408
0.9038462	89.77595	89.949	0.0147496	1.000283687	-2.749	-2.753
0.9230769	85.16771	85.299	0.0163134	1.000313768	-4.645	-4.650
0.9423077	83.96674	84.064	0.017331	1.000333344	-1.234	-1.235
0.9615385	83.03391	83.098	0.0163349	1.000314182	-0.965	-0.966
0.9807692	90.29464	90.329	0.0142013	1.000273139	7.229	7.231
1	94.22016	94.220			3.891	3.891
					-7.800	
Total					-7.825	-7.800

2 Black-Scholes Option Pricing

2.1 Introduction to options

An option gives the option holder the right, not the obligation, to buy or sell a security to the option writer. There are two main types: call option which gives the holder the right to buy a security and put option which gives them the right to sell a security.

2.2 Black-Scholes Pricing Model

The Black-Scholes option pricing equation initiated the modern theory of finance, based on the no-arbitrage principle, and derived using risk-neutral pricing. The equation is based on the assumption that the price fluctuations of the underlying security can be described well by an Ito process.

The Black-Scholes equation;

Suppose that the price of a security is governed by $dS = \mu S dt + \sigma S dz$ and the interest rate is r . A derivative of this security has a price $f(S, t)$, which satisfies the partial differential equation

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial S} rS + \frac{1}{2} \frac{\partial^2 f}{\partial S^2} \sigma^2 S^2 = rf$$

It follows from this result that the price of the call and put options are given by:

$$c(S_t, t) = S_t N(d_1) - K e^{-r(T-t)} N(d_2)$$

$$p(S_t, t) = K e^{-r(T-t)} N(-d_2) - S_t N(-d_1)$$

Where $N(\cdot)$ is the cumulative standard normal distribution function and,

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T - t)}{\sigma\sqrt{T - t}}$$

$$d_2 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)(T - t)}{\sigma\sqrt{T - t}}$$

To calculate the price of the call and put option we need to evaluate d_1 and d_2 at $t = 0$.

$$S_0 = 100 \quad K = 100 \quad r = 0.02 \quad \sigma = 0.25 \quad T = 1 \quad t = 0$$

$$d_1 = \frac{\ln\left(\frac{100}{100}\right) + \left(0.02 + \frac{1}{2}(0.25)^2\right)(1 - 0)}{0.25\sqrt{1 - 0}} = 0.205$$

$$d_2 = \frac{\ln\left(\frac{100}{100}\right) + \left(0.02 - \frac{1}{2}(0.25)^2\right)(1 - 0)}{0.25\sqrt{1 - 0}} = -0.045$$

$$N(d_1) = 0.581214$$

$$N(d_2) = 0.482054$$

$$N(-d_1) = 0.418786$$

$$N(-d_2) = 0.517946$$

Price of European Call option:

$$c(S_t, t) = S_t N(d_1) - K e^{-r(T-t)} N(d_2)$$

$$c(S_0, 0) = 100(0.581214) - 100e^{-0.02(1-0)}(0.482054) = \mathbf{10.87053087}$$

Price of European Put Option:

$$p(S_t, t) = K e^{-r(T-t)} N(-d_2) - S_t N(-d_1)$$

$$p(S_0, 0) = 100e^{-0.02(1-0)}(0.517946) - 100(0.418786) = \mathbf{8.890398204}$$

The prices calculated can be verified using the Put-Call parity which states;

Let C and P be the prices of a European call and put option respectively, both with a strike price of K and both defined on the same stock with price S. Put-call parity states that

$$C - P + e^{-r(T-t)} K = S$$

$$10.87053087 - 8.890398204 + (100)(e^{-0.02(1-0)}) = 100$$

$$100 = 100$$

2.3 The Greeks

Greeks are the quantities representing the sensitivity of the price of derivatives of the underlying asset S to a unit change in some variable.

$$\text{Delta } \Delta = \frac{\partial f}{\partial S}$$

$$\text{Gamma } \Gamma = \frac{\partial^2 f}{\partial S^2}$$

$$\text{Theta } \Theta = \frac{\partial f}{\partial t}$$

$$\text{Vega } v = \frac{\partial f}{\partial \sigma}$$

$$\text{Rho } \rho = \frac{\partial f}{\partial r}$$

Delta

Delta can be used to construct portfolios that hedge against a change in the underlying stock price. It represents the amount by which the price of an option changes as compared to a unit increase in the price of a stock expressed as

$$\Delta = \frac{\partial f}{\partial S}$$

The delta of a call and put option can be calculated from the Black-Scholes formula to be

$$\Delta c = N(d_1)$$

$$\Delta p = N(d_1) - 1$$

These formulas can be used to delta hedge strategies that employ options. Using the quantities calculated previously, it can be shown for a European call and put option that:

$$\Delta c = N(d_1) = 0.581214$$

$$\Delta p = N(d_1) - 1 = -0.418786$$

This means for every unit change in the stock price, the call option price will change by \$0.581214 and the put option price will change by -\$0.418786.

Gamma

Gamma of an option is the rate of change of the option delta with respect to a change in the price of the underlying asset.

$$\Gamma = \frac{\partial^2 f}{\partial S^2}$$

Like Delta, Gamma is constantly changing, even with tiny movements of the underlying stock price.

The Gamma of a call option (put option gives same result) is given by:

$$\begin{aligned}
\Gamma &= \frac{\partial^2 f}{\partial S^2} \\
&= \frac{\partial}{\partial S} N(d_1) \\
&= N'(d_1) \frac{\partial d_1}{\partial S} \\
&= N'(d_1) \frac{\partial}{\partial S} \left(\frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \right) \\
&= \frac{N'(d_1)}{S\sigma\sqrt{T-t}}
\end{aligned}$$

We note that: $N'(d_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}d_1^2}$

$$\Gamma = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}d_1^2}}{S\sigma\sqrt{T-t}}$$

$$\Gamma = \frac{\frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}(0.205^2)}}{100(0.25)\sqrt{1-0}}$$

$$\Gamma = \frac{0.4074137467}{25}$$

$$\Gamma = 0.01629654987$$

This means for every unit change in the stock price, the delta of the call and put option will change by 0.01629654987.

Vega

The Vega of an option is the rate of change of the option price with respect to the volatility of the underlying asset.

$$v = S \sqrt{\frac{T}{2\pi}} \exp - \left(\frac{\left(\ln \frac{S}{K} + \left(r + \frac{\sigma^2}{2} \right) T \right)^2}{2\sigma^2 T} \right)$$

The option price tends to be higher when the volatility is higher. Therefore, when calculation the option price due to volatility changes, we add Vega when volatility goes up and subtract Vega when volatility falls.

In Black-Scholes d_1 notation:

$$v = S\sqrt{T-t}N'(d_1)$$

$$v = 100\sqrt{1-0}(0.4074137467)$$

$$v = 40.74137465$$

A 1% change in volatility will result in an increase of \$0.4074 in option value (call and put).

2.4 Delta-hedging

Delta-hedging is an option strategy that aims to reduce the risk associated with price movements in the underlying asset by offsetting long and short positions. Delta-hedging focuses on the obtainment of Delta neutral portfolios. However, Delta itself varies both with stock price and time, hence a portfolio that is Delta neutral initially will not remain so. It is necessary to rebalance the portfolio by changing the proportions of its securities in order to maintain neutrality.

Call option

Suppose an investor buys N shares of stock and sells one call option. The portfolio value is:

$$V_t = NS_t - c_t$$

Where f_t is the Black-Scholes call option price and S_t is the stock price at time t .

The hedge is set up as follows:

$$\frac{\partial V}{\partial S} = N - \frac{\partial c}{\partial S} = 0$$

Recall previously that

$$\Delta c = \frac{\partial c}{\partial S} = N(d_1)$$

Therefore:

$$\begin{aligned} N &= \frac{\partial c}{\partial S} \\ &= N(d_1) = \Delta c \\ N &= \Delta c \end{aligned}$$

In order for the investor to Delta hedge the call option, the number of shares of stock N should equal to Delta of the call option.

Put Option

Supposed an investor sells one put option and sells N shares of stock. The portfolio value will be as follows:

$$V_t = NS_t - p_t$$

Where p_t is the Black-Scholes put option price

The hedge will be set up as follows:

$$\frac{\partial V}{\partial S} = N - \frac{\partial p}{\partial S} = 0$$

We know what

$$\Delta p = \frac{\partial p}{\partial S} = N(d_1) - 1$$

Therefore

$$N = \frac{\partial p}{\partial S}$$

$$= N(d_1) - 1 = \Delta p$$

$$N = \Delta p$$

Straddle

Suppose the investor has written both a call and a put option simultaneously. The portfolio value will be:

$$V_t = NS_t - (c_t + p_t)$$

Let $x = c_t + p_t$

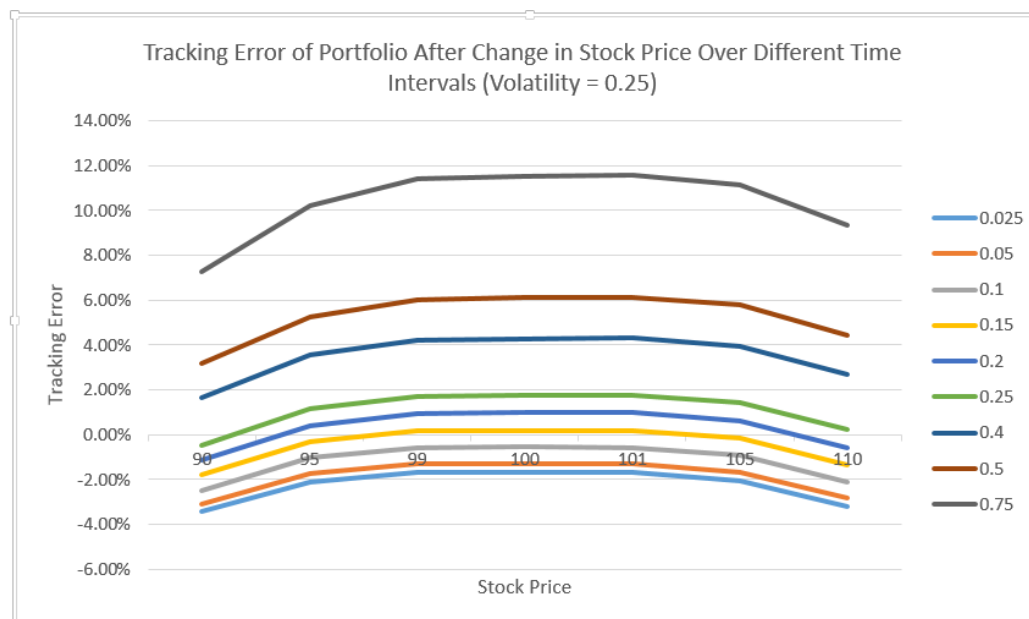
$$V_t = NS_t - x$$

$$\frac{\partial V}{\partial S} = N - \frac{\partial x}{\partial S} = 0$$

$$N = \frac{\partial x}{\partial S} = N(x) = \Delta x$$

Since the delta of the portfolio is equal to the delta of each investment $\Delta x = \Delta c + \Delta p$

Note that we set the value of the portfolios equal to 0 initially by borrowing the cost of the portfolio at the risk-free rate. So, for the call option the change in portfolio value or tracking error at time t compared with time 0 would be given by $(\Delta c)S_t - c_t - [(\Delta c)S_0 - c_0]e^{0.02t}$, and similarly for the put and straddle. We examined the tracking error or the change in portfolio value when the stock price changes for different intervals of time. The results produced the graph below.

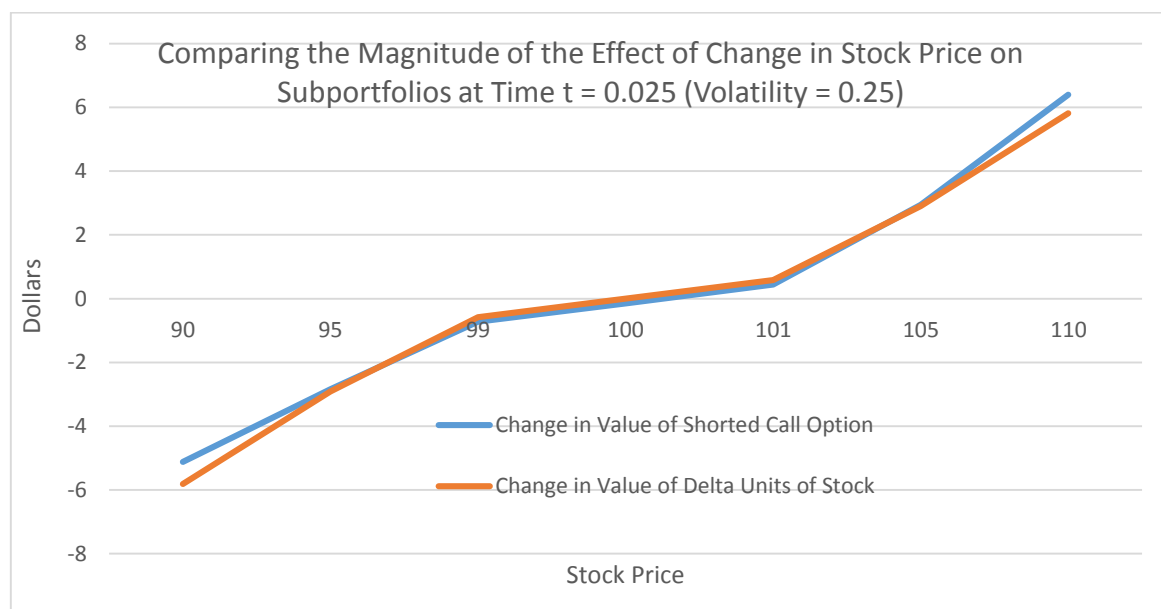


The tracking error increases as time is increased since the call option becomes less valuable the closer it is to maturity. This explains the upwards shift in the curves with time.

When the time change is small, the tracking error is negative when stock price is still valued at \$100 because the decline in call option value is less than the loss from the interest of initial portfolio cost accumulated at the risk free rate. However, when time increases by significant amount, the value of the call option is reduced so much that the portfolio value becomes very positive, despite the extra accumulation of the initial portfolio cost. This explains the tracking error even when the stock price is still valued at \$100.

The shape of all the curves is similar. When stock price decreases, value of the units of stock held decreases, but the value of the call option also decreases, so this is not enough to indicate whether the whole portfolio grows/shrinks in value. What matters here is the relative magnitude between the two decreases. As displayed in the graph below (with $t = 0.025$ arbitrarily – the argument is the same for any time difference), when stock price decreases, the relative decrease in the call option value is less than that of the decline in the value of the delta units of stock, thus, portfolio value or tracking error decreases. In the opposite way, relative increase in call option value is more than that of the incline in the value of the delta units of stock, thus, the portfolio value or tracking error again decreases. This explains the ‘U’ shape of the tracking errors with respect to stock price.

Also, as can be seen more clearly in the table presented below the graph, the value of the units of stock held change by the same proportion that the stock price changes, however, the call option value changes more for increases in the stock price than for decreases in the stock price. This explains why the tracking error or portfolio value is higher when the stock price increases than when the stock price decreases by the same proportion.



Stock Price	90	95	99	100	101	105	110
Change in Value of Call Option (\$)	-5.1179	-2.8442	-0.7188	-0.1465	0.44154	2.94599	6.39324

Below is the figures for the tracking errors under each stock price change for each time interval:

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.25 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-1.62465	-0.99234	-0.79287	-0.78389	-0.79075	-0.97035	-1.51154
0.05	-1.4735	-0.82786	-0.62166	-0.61156	-0.6175	-0.7955	-1.33879
0.1	-1.16755	-0.49406	-0.27392	-0.2615	-0.26559	-0.44051	-0.98866
0.15	-0.8565	-0.1534	0.081414	0.096228	0.09404	-0.07802	-0.63199
0.2	-0.53999	0.194696	0.445008	0.462308	0.462063	0.292612	-0.26833
0.25	-0.21767	0.550897	0.817641	0.837526	0.839272	0.67211	0.102863
0.4	0.788508	1.676474	1.999745	2.028118	2.036134	1.873031	1.267961
0.5	1.497123	2.485681	2.855135	2.889967	2.902436	2.738198	2.0956
0.75	3.434603	4.837022	5.388427	5.444734	5.469085	5.262949	4.409168

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.25 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-3.44%	-2.10%	-1.68%	-1.66%	-1.67%	-2.05%	-3.20%
0.05	-3.12%	-1.75%	-1.32%	-1.29%	-1.31%	-1.68%	-2.83%
0.1	-2.47%	-1.05%	-0.58%	-0.55%	-0.56%	-0.93%	-2.09%
0.15	-1.81%	-0.32%	0.17%	0.20%	0.20%	-0.17%	-1.34%
0.2	-1.14%	0.41%	0.94%	0.98%	0.98%	0.62%	-0.57%
0.25	-0.46%	1.17%	1.73%	1.77%	1.78%	1.42%	0.22%
0.4	1.67%	3.55%	4.23%	4.29%	4.31%	3.96%	2.68%
0.5	3.17%	5.26%	6.04%	6.12%	6.14%	5.80%	4.44%
0.75	7.27%	10.24%	11.40%	11.52%	11.57%	11.14%	9.33%

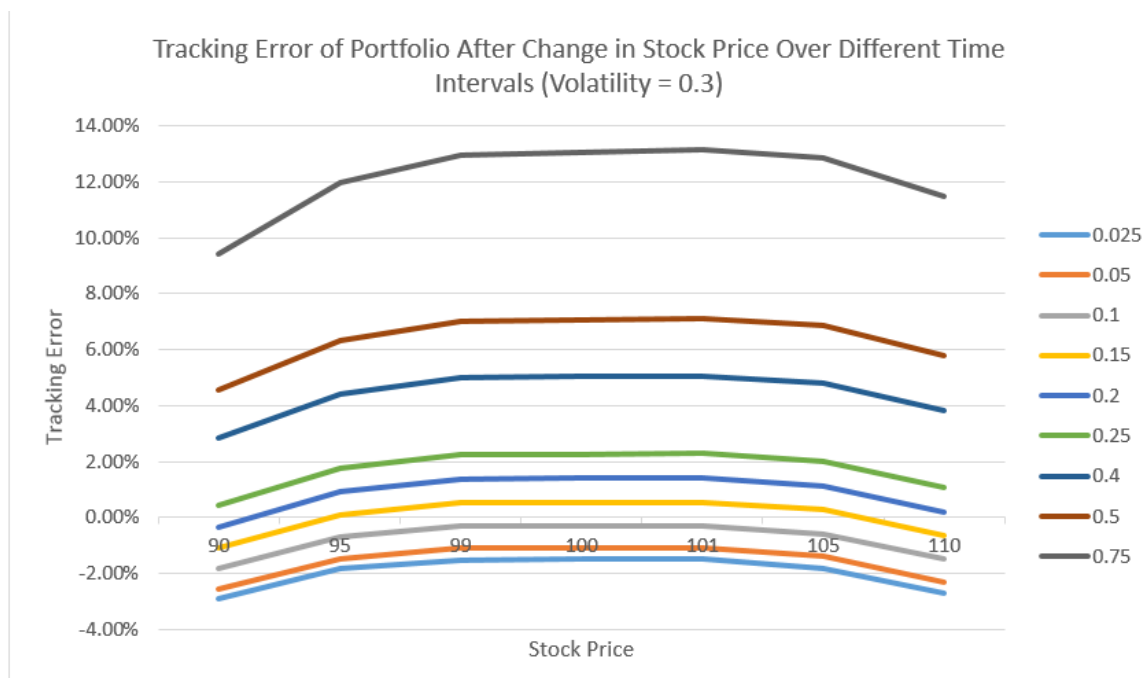
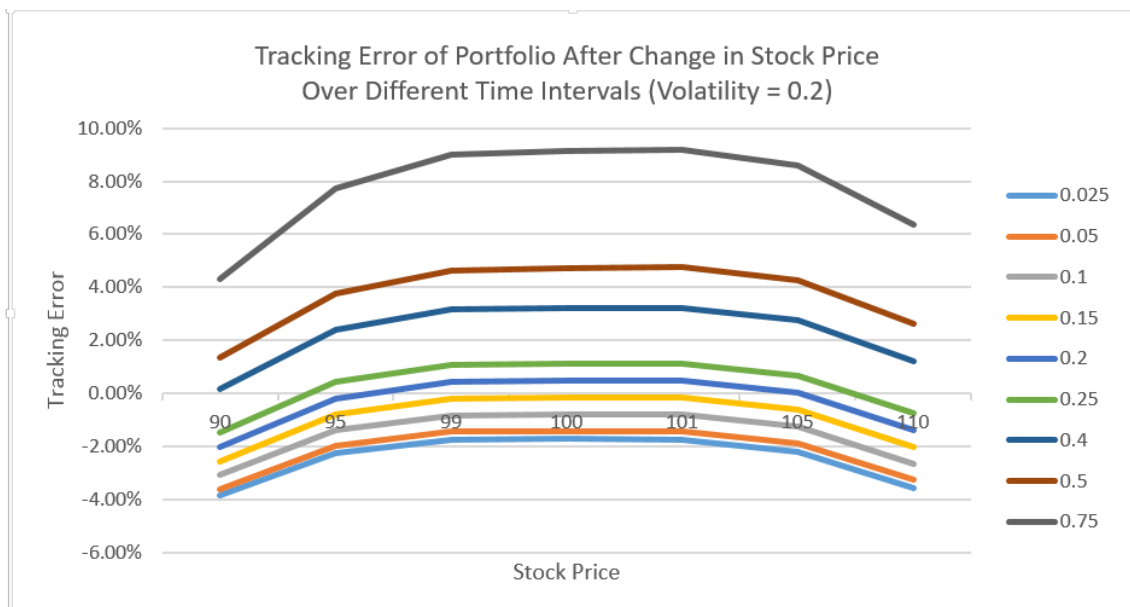
We then examined the tracking errors when volatility of stock price changes (this is both initial volatility and volatility at the end of time interval in question):

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.3 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-1.43298	-0.90528	-0.73865	-0.73097	-0.73645	-0.88526	-1.33781
0.05	-1.25762	-0.71758	-0.54443	-0.53558	-0.54008	-0.68667	-1.13996
0.1	-0.90203	-0.33624	-0.14961	-0.13837	-0.14085	-0.28307	-0.73836
0.15	-0.53956	0.053537	0.254316	0.268029	0.267603	0.129647	-0.32841
0.2	-0.1697	0.452478	0.668151	0.684427	0.686111	0.552256	0.09053
0.25	0.208116	0.861407	1.092822	1.11176	1.115609	0.985656	0.519183
0.4	1.396171	2.158606	2.443758	2.471409	2.482124	2.36199	1.872411
0.5	2.24249	3.096398	3.42497	3.459222	3.474869	3.358557	2.842316
0.75	4.624964	5.853897	6.349596	6.405537	6.434914	6.298407	5.61534

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.3 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-2.92%	-1.85%	-1.51%	-1.49%	-1.50%	-1.81%	-2.73%
0.05	-2.57%	-1.46%	-1.11%	-1.09%	-1.10%	-1.40%	-2.33%
0.1	-1.84%	-0.69%	-0.31%	-0.28%	-0.29%	-0.58%	-1.51%
0.15	-1.10%	0.11%	0.52%	0.55%	0.55%	0.26%	-0.67%
0.2	-0.35%	0.92%	1.36%	1.40%	1.40%	1.13%	0.18%
0.25	0.42%	1.76%	2.23%	2.27%	2.28%	2.01%	1.06%
0.4	2.85%	4.40%	4.99%	5.04%	5.06%	4.82%	3.82%
0.5	4.58%	6.32%	6.99%	7.06%	7.09%	6.85%	5.80%
0.75	9.44%	11.94%	12.96%	13.07%	13.13%	12.85%	11.46%

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.2 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-1.88913	-1.10183	-0.85309	-0.84213	-0.85099	-1.07569	-1.744
0.05	-1.76324	-0.96094	-0.70492	-0.69283	-0.70082	-0.92461	-1.59666
0.1	-1.50915	-0.67544	-0.4043	-0.38989	-0.39613	-0.6183	-1.29865
0.15	-1.25182	-0.3847	-0.09759	-0.08077	-0.08522	-0.30611	-0.99598
0.2	-0.99111	-0.08826	0.215754	0.235074	0.232445	0.012464	-0.68836
0.25	-0.72685	0.214348	0.536362	0.558286	0.557517	0.337963	-0.37548
0.4	0.088751	1.165388	1.549817	1.580322	1.585355	1.363024	0.598311
0.5	0.652806	1.843657	2.279657	2.316726	2.325807	2.096251	1.280848
0.75	2.123567	3.7795	4.423225	4.4824	4.501479	4.201739	3.123929

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.2 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-3.85%	-2.25%	-1.74%	-1.72%	-1.74%	-2.19%	-3.56%
0.05	-3.60%	-1.96%	-1.44%	-1.41%	-1.43%	-1.89%	-3.26%
0.1	-3.08%	-1.38%	-0.82%	-0.80%	-0.81%	-1.26%	-2.65%
0.15	-2.55%	-0.78%	-0.20%	-0.16%	-0.17%	-0.62%	-2.03%
0.2	-2.02%	-0.18%	0.44%	0.48%	0.47%	0.03%	-1.40%
0.25	-1.48%	0.44%	1.09%	1.14%	1.14%	0.69%	-0.77%
0.4	0.18%	2.38%	3.16%	3.22%	3.23%	2.78%	1.22%
0.5	1.33%	3.76%	4.65%	4.73%	4.75%	4.28%	2.61%
0.75	4.33%	7.71%	9.03%	9.15%	9.18%	8.57%	6.37%



So, it is clear from the graphs and figures that when the volatility of the stock price increases the tracking errors of the portfolios increase, and vice-versa. This is because when the stock price volatility increases the call option value becomes less sensitive to changes in price as is shown in the table below. However, with extra volatility, the call option value is much more sensitive to increases in stock price (see difference of 14.89% between 73.70% and 58.81%) than decreases in stock price (compare with difference 7.52% between corresponding figures -54.6% and -47.08%). This means that when the stock price increases the call option value grows by relatively less and thus the tracking error or portfolio value increases, and vice-versa.

Value of Call Option (\$) and Percentage Change in Value								
Volatility of Stock Price	t = 0	t = 0.025						
	100	90	95	99	100	101	105	110
0.2	8.91604	4.04749	6.15650	8.22479	8.79310	9.38121	11.92295	15.48756
0.25	10.87056	5.75263	8.02639	10.15178	10.72402	11.31209	13.81655	17.26380
0.3	12.82158	7.49592	9.89705	12.07349	12.65157	13.24282	15.73470	19.11607
0.2	0.00%	-54.60%	-30.95%	-7.75%	-1.38%	5.22%	33.72%	73.70%
0.25	0.00%	-47.08%	-26.16%	-6.61%	-1.35%	4.06%	27.10%	58.81%
0.3	0.00%	-41.54%	-22.81%	-5.83%	-1.33%	3.29%	22.72%	49.09%

Note that writing of the put option yields the exact same results and analysis since tracking errors are exactly equal to the when writing a call option. This result was expected due to put-call parity. This is also the case for the straddle whose tracking errors are exactly double those when writing the call option. This result was expected since it was equivalent to the summation of the previous two portfolios.

2.5 Delta-Vega Hedging

Suppose we wanted to hedge Delta and Vega (volatility risk). To do this we must use another option.

Let;

π be the hedged portfolio

f be the option written

g be the option hedged

ω_1 be the amount invested in stock

ω_2 be the amount invested in g

Therefore, portfolio is given by;

$$\pi = -f + \omega_1 S + \omega_2 g$$

The Delta-Vega hedge implies:

$$\frac{\partial \pi}{\partial S} = -\frac{\partial f}{\partial S} + \omega_1 + \omega_2 \frac{\partial g}{\partial S} = 0$$

$$\frac{\partial \pi}{\partial \sigma} = -\frac{\partial f}{\partial \sigma} + \omega_2 \frac{\partial g}{\partial \sigma} = 0$$

Where we assume that $\frac{\partial S}{\partial \sigma} = 0$, i.e. constant volatility. By replacing the deviations with their corresponding Greek symbols, we can find the Delta and the Vega of the portfolio.

$$\Delta \pi = -\Delta f + \omega_1 + \omega_2 \Delta g = 0$$

$$v\pi = -vf + \omega_2 vg = 0$$

$$\Rightarrow \omega_2 = \frac{vf}{vg}$$

$$\omega_1 = \Delta f - \frac{vf}{vg} \Delta g$$

Therefore, for a portfolio to be Delta-Vega hedged, you calculate the appropriate Greeks of each option and from these you can calculate the values of ω_1 and ω_2 required to be hedged.

Call Option

Suppose an investor has written a call option and wants to Delta-Vega hedge their position. The investor will need to purchase ω_2 units of the new call option g . By purchasing ω_2 units, the investor has eliminated the volatility risk and is no longer exposed to the Vega of the written call option. The investor will also want to delta hedge the written option, and to do so will purchase ω_1 units of stock.

The portfolio will have the following value;

$$\pi = -f + \omega_1 S + \omega_2 g$$

Put Option

Suppose an investor has written a put option and wants to Delta-Vega hedge their position. By purchasing ω_2 units of the new put option g the investor has eliminated the volatility risk and by selling ω_1 units of stock, the portfolio has been delta hedged. The portfolio will have the following value:

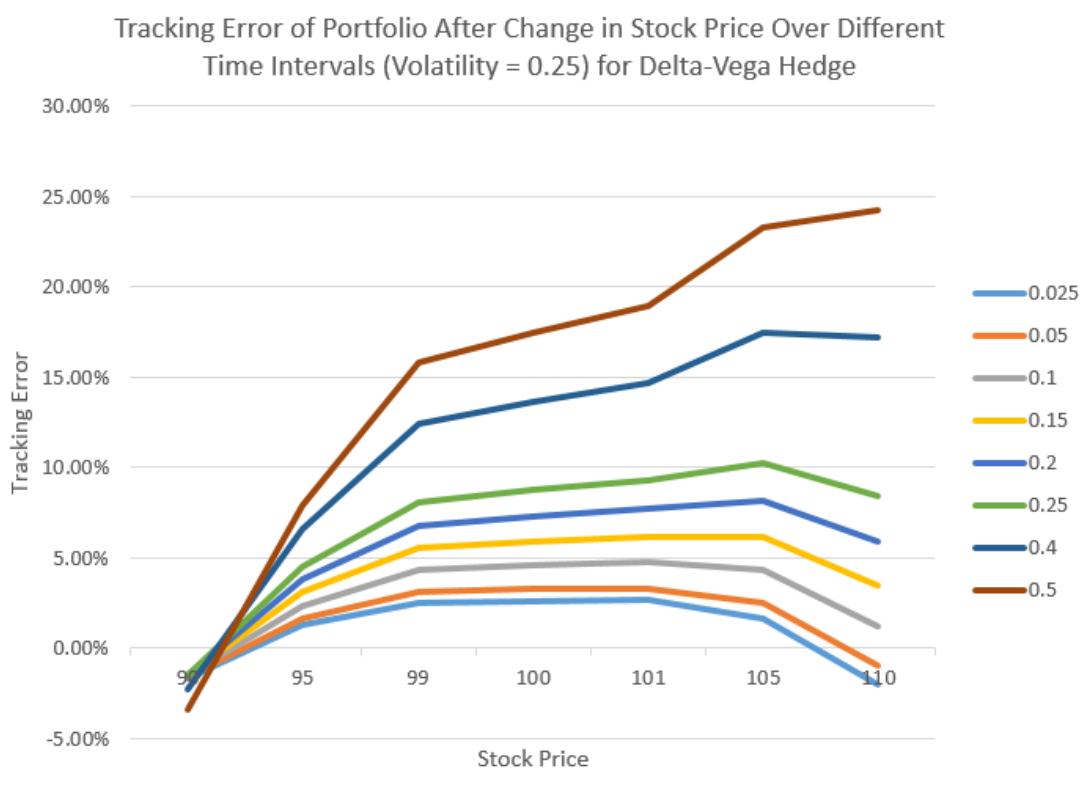
$$\pi = -f - \omega_1 S + \omega_2 g$$

Straddle

Suppose an investor has written a straddle and they now want to Delta-Vega hedge their position. As above the portfolio will be set up as follows

$$\pi = -f - \omega_1 S + \omega_2 g$$

Note that we set the value of the portfolios equal to 0 initially by borrowing the cost of the portfolio at the risk-free rate. So for the call option the change in portfolio value or tracking error at time t compared with time 0 would be given by $\left(\Delta c - \frac{v_c}{v_p} \Delta p\right) S_t - c_t + \frac{v_c}{v_p} p_t - \left[\left(\Delta c - \frac{v_c}{v_p} \Delta p\right) S_0 - c_0 + \frac{v_c}{v_p} p_0\right] e^{0.02r}$, and similarly for the put and straddle. We examined the tracking error or the change in portfolio value when the stock price changes for different intervals of time. The results produced the graph below.



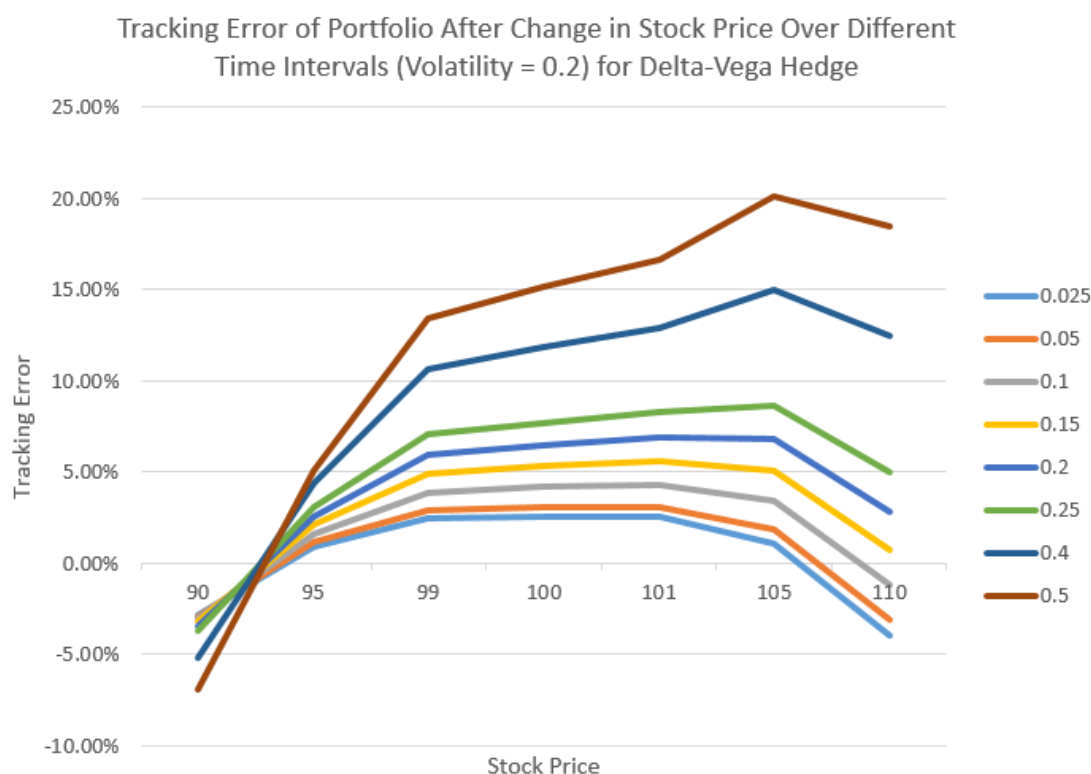
Note that over small intervals, the shape of the curve is very similar as in the delta hedging portfolio. The change in portfolio value is approximately equal to $(f_0 - f_t) + \omega_1(S_t - S_0) + \omega_2(g_t - g_0)$. This is not how we calculated the tracking error since we borrowed the initial portfolio cost from the risk-free rate, but writing the change in portfolio value this way allows for easier description of the interactions. Note that if the stock price increases $\omega_1(S_t - S_0) < 0$, $\omega_2(g_t - g_0) < 0$ and $(f_0 - f_t) < 0$, so each term is negative and the portfolio value or tracking error grows immensely. Similarly, if the stock price decreases $\omega_1(S_t - S_0) > 0$, $\omega_2(g_t - g_0) > 0$ and $(f_0 - f_t) > 0$, so each term is positive and the portfolio value or tracking error declines majorly. If this is over a larger interval of time then the differences between initial call and current call value (similarly with put) become much more pronounced. The reason why the decline in portfolio value when the stock price decreases is not as large as the growth when the stock price increases is because the amount by which put option values increase when the stock price decreases is more than the amount by which they decrease when the stock price increases by the same proportion. This opposite is true for call option values. This means that when the stock price is low, the call options do not decline much but the put options increase majorly, thereby limiting the decrease in portfolio value. Also, if the curve does not grow but instead declines when the stock price increases, this is because the initial portfolio cost accumulated forward at the risk-free rate is greater than the difference in the option values.

Below are the figures for the tracking errors under each stock price change for each time interval:

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.25 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-0.033701	0.026709	0.051469	0.053418	0.053481	0.033445	-0.040996
0.05	-0.032382	0.033848	0.063149	0.066171	0.067275	0.050991	-0.019856
0.1	-0.030359	0.048163	0.087081	0.092376	0.095681	0.087317	0.023988
0.15	-0.029277	0.062507	0.111841	0.119596	0.125281	0.125453	0.070122
0.2	-0.029306	0.076849	0.137519	0.147952	0.156223	0.165632	0.118833
0.25	-0.030655	0.091149	0.164226	0.177586	0.188684	0.208139	0.170462
0.4	-0.045620	0.133153	0.251997	0.276138	0.297615	0.353516	0.347492
0.5	-0.069443	0.159257	0.319098	0.352872	0.383582	0.471254	0.490771
0.75	-0.249761	0.196075	0.547892	0.626396	0.699338	0.922719	1.023072

Tracking Errors for a Delta-Vega Hedge with Stock Price Volatility of 0.25 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-1.67%	1.32%	2.54%	2.64%	2.64%	1.65%	-2.03%
0.05	-1.60%	1.67%	3.12%	3.27%	3.32%	2.52%	-0.98%
0.1	-1.50%	2.38%	4.30%	4.56%	4.73%	4.31%	1.19%
0.15	-1.45%	3.09%	5.53%	5.91%	6.19%	6.20%	3.47%
0.2	-1.45%	3.80%	6.80%	7.31%	7.72%	8.18%	5.87%
0.25	-1.51%	4.50%	8.12%	8.78%	9.32%	10.29%	8.42%
0.4	-2.25%	6.58%	12.45%	13.65%	14.71%	17.47%	17.17%
0.5	-3.43%	7.87%	15.77%	17.44%	18.95%	23.29%	24.25%
0.75	-12.34%	9.69%	27.07%	30.95%	34.56%	45.60%	50.56%

We then examined the tracking errors when volatility of stock price changes (this is both the initial volatility and volatility at the end of time interval in question):



Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.3 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-0.019710	0.024696	0.042649	0.044160	0.044374	0.031435	-0.018051
0.05	-0.016281	0.033057	0.054834	0.057258	0.058363	0.048719	0.002634
0.1	-0.009857	0.049904	0.079780	0.084137	0.087126	0.084444	0.045503
0.15	-0.004105	0.066913	0.105558	0.112007	0.117035	0.121865	0.090569
0.2	0.000839	0.084069	0.132256	0.140981	0.148226	0.161198	0.138110
0.25	0.004807	0.101353	0.159981	0.171194	0.180861	0.202703	0.188462
0.4	0.008401	0.153602	0.250725	0.271105	0.289674	0.343849	0.360958
0.5	-0.000069	0.188039	0.319626	0.348198	0.374698	0.457289	0.500667
0.75	-0.121764	0.257119	0.549967	0.616646	0.679670	0.887259	1.027950

Tracking Errors for a Delta-Vega Hedge with Stock Price Volatility of 0.3 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-1.27%	1.60%	2.75%	2.85%	2.87%	2.03%	-1.17%
0.05	-1.05%	2.14%	3.54%	3.70%	3.77%	3.15%	0.17%
0.1	-0.64%	3.22%	5.15%	5.43%	5.63%	5.45%	2.94%
0.15	-0.27%	4.32%	6.82%	7.23%	7.56%	7.87%	5.85%
0.2	0.05%	5.43%	8.54%	9.11%	9.57%	10.41%	8.92%
0.25	0.31%	6.55%	10.33%	11.06%	11.68%	13.09%	12.17%
0.4	0.54%	9.92%	16.19%	17.51%	18.71%	22.21%	23.32%
0.5	0.00%	12.15%	20.65%	22.49%	24.20%	29.54%	32.34%
0.75	-7.87%	16.61%	35.52%	39.83%	43.90%	57.31%	66.40%

Tracking Errors for a Delta Hedge with Stock Price Volatility of 0.2 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-0.069971	0.021446	0.059603	0.062350	0.062015	0.027186	-0.097720
0.05	-0.070996	0.027537	0.071310	0.075362	0.076275	0.045743	-0.075627
0.1	-0.073894	0.039647	0.095314	0.102129	0.105682	0.084201	-0.029840
0.15	-0.078070	0.051626	0.120176	0.129981	0.136386	0.124630	0.018280
0.2	-0.083735	0.063423	0.145995	0.159050	0.168553	0.167286	0.069010
0.25	-0.091147	0.074972	0.172890	0.189499	0.202383	0.212477	0.122677
0.4	-0.127498	0.107125	0.261644	0.291330	0.316600	0.367492	0.305606
0.5	-0.169135	0.124773	0.329989	0.371340	0.407580	0.493426	0.451924
0.75	-0.415272	0.125686	0.568153	0.663346	0.749006	0.975032	0.969507

Tracking Errors for a Delta-Vega Hedge with Stock Price Volatility of 0.2 (as a % of initial Portfolio Value)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	-2.85%	0.87%	2.43%	2.54%	2.53%	1.11%	-3.98%
0.05	-2.89%	1.12%	2.91%	3.07%	3.11%	1.86%	-3.08%
0.1	-3.01%	1.62%	3.88%	4.16%	4.31%	3.43%	-1.22%
0.15	-3.18%	2.10%	4.90%	5.30%	5.56%	5.08%	0.74%
0.2	-3.41%	2.58%	5.95%	6.48%	6.87%	6.82%	2.81%
0.25	-3.71%	3.06%	7.05%	7.72%	8.25%	8.66%	5.00%
0.4	-5.20%	4.37%	10.66%	11.87%	12.90%	14.98%	12.45%
0.5	-6.89%	5.08%	13.45%	15.13%	16.61%	20.11%	18.42%
0.75	-16.92%	5.12%	23.15%	27.03%	30.53%	39.74%	39.51%

It is clear from the graphs and figures that when the volatility of the stock price increases the tracking errors of the portfolios increase, and vice-versa. This is because when the stock price volatility increases the call option and the put option values becomes less sensitive to changes in price (as shown previously). This means when the stock price increases the changes in the put and call option values are smaller, and thus, the tracking errors are smaller.

Finally, if we compare the delta-hedge and the delta-vega hedge in absolute dollar terms we find that the delta-vega hedge performs much better for each time interval and each volatility, as is displayed in the tables below. These figures were produced by calculating for each time interval and stock price volatility: $|TrackError_D| - |TrackError_{DV}|$, so to find how close was the delta tracking error to 0 in comparison in with the delta-vega tracking error. Notice that many of figures below are positive indicating a larger tracking error for the delta hedge as the delta-vega hedge.

Difference in Absolute Tracking Errors for the Delta Hedge and the Delta-Vega Hedge with Stock Price Volatility of 0.25 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	1.590945	0.965629	0.741397	0.730477	0.737274	0.936908	1.470539
0.05	1.441114	0.794007	0.558508	0.545385	0.550227	0.744506	1.318932
0.1	1.137193	0.445898	0.186836	0.169125	0.169908	0.353194	0.964669
0.15	0.827218	0.090896	-0.030426	-0.023369	-0.031241	-0.047434	0.561872
0.2	0.510687	0.117848	0.307489	0.314356	0.305840	0.126979	0.149497
0.25	0.187018	0.459748	0.653415	0.659940	0.650588	0.463971	-0.067599
0.4	0.742889	1.543321	1.747748	1.751979	1.738518	1.519515	0.920469
0.5	1.427680	2.326424	2.536037	2.537094	2.518854	2.266944	1.604830
0.75	3.184842	4.640947	4.840535	4.818339	4.769747	4.340230	3.386096

Difference in Absolute Tracking Errors for the Delta Hedge and the Delta-Vega Hedge with Stock Price Volatility of 0.3 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	1.413265	0.880585	0.696000	0.686808	0.692075	0.853828	1.319757
0.05	1.241338	0.684524	0.489598	0.478326	0.481714	0.637947	1.137323
0.1	0.892173	0.286340	0.069831	0.054234	0.053729	0.198624	0.692852
0.15	0.535452	-0.013376	0.148758	0.156022	0.150568	0.007782	0.237838
0.2	0.168858	0.368409	0.535895	0.543446	0.537885	0.391059	-0.047579
0.25	0.203309	0.760054	0.932841	0.940566	0.934748	0.782953	0.330721
0.4	1.387770	2.005004	2.193033	2.200304	2.192450	2.018141	1.511453
0.5	2.242421	2.908359	3.105344	3.111024	3.100170	2.901268	2.341649
0.75	4.503200	5.596777	5.799630	5.788891	5.755244	5.411148	4.587390

Difference in Absolute Tracking Errors for the Delta Hedge and the Delta-Vega Hedge with Stock Price Volatility of 0.2 (in Dollars)							
	Stock Price (\$)						
Time t	90	95	99	100	101	105	110
0.025	1.819155	1.080383	0.793484	0.779783	0.788971	1.048506	1.646280
0.05	1.692248	0.933398	0.633605	0.617466	0.624544	0.878866	1.521033
0.1	1.435256	0.635797	0.308986	0.287761	0.290446	0.534100	1.268809
0.15	1.173753	0.333070	-0.022586	-0.049208	-0.051162	0.181475	0.977698
0.2	0.907376	0.024839	0.069759	0.076023	0.063891	-0.154822	0.619349
0.25	0.635708	0.139376	0.363472	0.368788	0.355134	0.125485	0.252802
0.4	-0.038747	1.058264	1.288174	1.288992	1.268756	0.995531	0.292704
0.5	0.483672	1.718885	1.949668	1.945386	1.918228	1.602825	0.828925
0.75	1.708296	3.653814	3.855072	3.819055	3.752473	3.226707	2.154423

3 Numerical Methods for Option Pricing

3.1 Binomial Model

As derived in Section 1, stocks who follow GBM such that $dS = \mu S dt + \sigma S dW$, we find using Itô's Lemma that $\log(S_t) \sim N(\log(S_0) + \nu t, \sigma^2 t)$

Under the Binomial model the stock price will be either uS or dS at the end of one small time step, where $u > R = e^{r\Delta t} > d$, and p = probability of upward move. We assumed that $d = \frac{1}{u}$ and that the stock price at time 0 is 1; in other words, $S(0) = 1$.^[2] We could scale up later. Thus,

$$E\{\log(S_1)\} = p\log(u) + (1-p)\log(d) \text{ and set } E\{\log(S_1)\} = \log(1) + \nu\Delta t = \nu\Delta t$$

$$\text{such that: } p\log(u) + (1-p)\log(d) = \nu\Delta t \quad (1)$$

$$\text{Var}\{\log(S_1)\} = p\log(u)^2 + (1-p)\log(d)^2 \text{ and set } \text{Var}\{\log(S_1)\} = \sigma^2\Delta t$$

$$\text{such that: } p\log(u)^2 + (1-p)\log(d)^2 = \sigma^2\Delta t \quad (2)$$

Solving these simultaneous equations yields: $u = e^{\sqrt{\sigma^2\Delta t + (\nu\Delta t)^2}}$ and $d = e^{-\sqrt{\sigma^2\Delta t + (\nu\Delta t)^2}}$.

Now suppose we replicated call / put option payoffs by investing x dollars in underlying stock and b dollars in the risk-free asset. Then at time 1 the following equations must hold^[3]:

$$\begin{aligned} ux + Rb &= C_u \\ ux + Rb &= C_d \end{aligned}$$

Where C_u and C_d denote the call / put option payoffs at time 1, given an upward or downward stock movement respectively. Solving these simultaneous equations yields: $x = \frac{C_u - C_d}{u - d}$ and $b = \frac{uC_d - dC_u}{R(u - d)}$, so the value of the portfolio, and thus call / put option at time 0 is:

$$x + b = \frac{1}{R} \left(\frac{R-d}{u-d} C_u - \frac{u-d}{u-d} C_d \right) = \frac{1}{R} (qC_u + (1-q)C_d) \text{ where } q = \frac{R-d}{u-d}.$$

So the option value at $t-1$ can be thought of as $C_{t-1} = \frac{1}{R} \hat{E}\{C_t\}$ which is the present value of the expectation of the possible option values at time t with risk-neutral probabilities q and $1-q$ of occurrence.

The first step we took was to forecast stock price over all possible paths in 52 weeks, by multiplying the previous step's stock price by u/d for an upward/downward movement. Then we calculated, for each branch, the value of the European call and put option payoffs at time 52, where $C_{52}^k = \max\{K - S_{52}^k, 0\}$ and $P_{52}^k = \max\{S_{52}^k - K, 0\}$. C_{52}^k and P_{52}^k are the k^{th} possible values at time 52 of the European call and put options respectively. After which, we iterated the formula working backwards through the branches to arrive at the cost of the both of the options.

The American put option could be exercised at any time step before maturity, thus the only change to the formula was that $C_{t-1}^k = \max\left\{\frac{1}{R} \hat{E}\{C_t\}, K - S_{t-1}^k\right\}$ ^[4]. If the expected present value of the put is lower than the current value, then it is discarded and the current value is the price of the put at that time step.

$$\nu = \mu - \frac{1}{2}\sigma^2 = 0.03875, \quad r = 0.02, \quad \Delta t = \frac{1}{52}, \quad R = e^{r\Delta t} = 1.000385, \quad K = 100$$

$$u = e^{\sqrt{\sigma^2\Delta t + (\nu\Delta t)^2}} = 1.035285, \quad d = \frac{1}{u} = 0.9659176$$

²http://wwwf.imperial.ac.uk/~mdavis/FDM11/BINOMIAL_AMERICAN_REV.PDF

³<http://www.uio.no/studier/emner/sv/oekonomi/ECON4510/v11/undervisningsmateriale/lect2803.pdf>

Subsequently we found prices of the options as follows:

Binomial Tree (52 Step) European Call Price = \$ 10.82542
Binomial Tree (52 Step) European Put Price = \$ 8.84529
Binomial Tree (52 Step) American Put Price = \$ 9.035699

These prices seem reasonable – we expect the European call to be worth more than the European put since the stock has a drift that is positive and the current price is the same as the strike price^[5]. Also, the American put should be worth more than the European put due to option of early exercise. We used the put-call parity formula to conduct a small check: $C - P + dK = S^{[6]}$, where d = the total discount factor over the period.

$$LHS = 10.82542 - 8.84529 + (R^{52})100 = 100 \quad \text{and} \quad RHS = 100$$

Thus, we attained the expected result.

We exported the European put tree to Excel to give an idea graphically of this process:

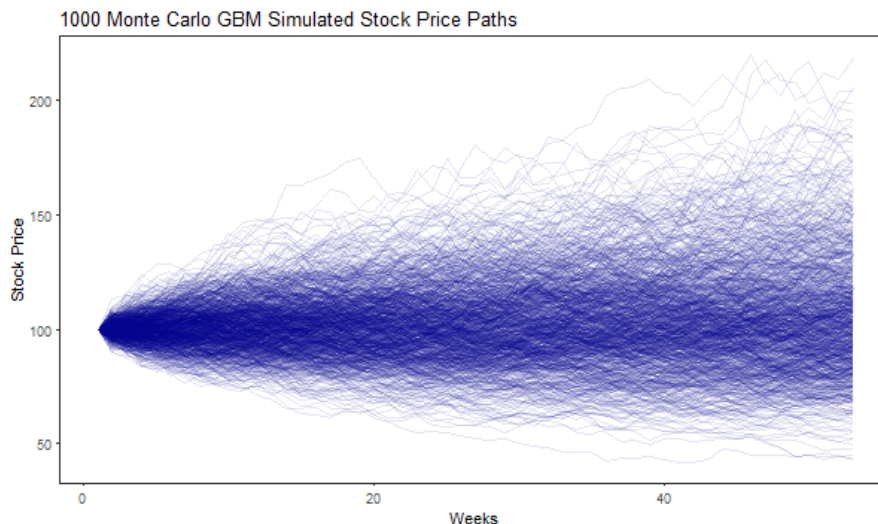
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	time 0	time 1	time 2	time 3	time 4	time 5	time 6	time 7	time 8	time 9	time 10	time 11	time 12	time 13	time 14	time 15	time 16	time 17	time 18	time 19
2	8.84529	7.385786	6.067229	4.895956	3.874541	3.001582	2.271739	1.676033	1.202407	0.8365	0.562569	0.364458	0.22653	0.134457	0.075797	0.04032	0.020084	0.009279	0.00393	0.001503
3	0	10.29344	8.693622	7.228602	5.908436	4.739626	3.724661	2.861788	2.145061	1.56469	1.107671	0.758652	0.500953	0.317634	0.192492	0.110891	0.060337	0.03077	0.014569	0.006633
4	0	0	11.88127	10.1471	8.537908	7.067255	5.745616	4.579671	3.571806	2.719868	2.017233	1.453205	1.013731	0.682379	0.441465	0.273228	0.160902	0.089583	0.046793	0.022717
5	0	0	0	13.60299	11.74408	9.996834	8.377892	6.901481	5.578527	4.4159	3.415861	2.575802	1.888334	1.341745	0.920825	0.607952	0.384368	0.231458	0.13191	0.070606
6	0	0	0	0	15.44923	13.47862	11.60332	9.842382	8.21329	6.730987	5.406901	4.248104	3.256707	2.429581	1.758467	1.230518	0.829226	0.535674	0.329948	0.192554
7	0	0	0	0	0	17.40719	15.34095	13.35128	11.45878	9.683472	8.043784	6.555449	5.230442	4.076056	3.094221	2.281208	1.627769	1.119768	0.739256	0.465888
8	0	0	0	0	0	0	19.4611	17.31765	15.2305	13.22081	11.31021	9.519796	7.86902	6.374498	5.048819	3.899508	2.92828	2.130708	1.496413	1.009796
9	0	0	0	0	0	0	0	21.59282	19.39214	17.22688	15.1178	13.08705	11.15736	9.351007	7.688596	6.187724	4.861662	3.718191	2.75876	1.978133
10	0	0	0	0	0	0	0	0	23.78268	21.54536	19.32295	17.1349	15.00279	12.94982	10.99993	9.176716	7.502056	5.994657	4.668555	3.531806
11	0	0	0	0	0	0	0	0	0	26.01043	23.75665	21.49862	19.25365	17.04175	14.88541	12.80893	10.8376	8.99648	7.308882	5.794763
12	0	0	0	0	0	0	0	0	0	0	28.25612	26.00481	23.73217	21.4528	19.18436	16.94749	14.76557	12.66416	10.67001	8.809794
13	0	0	0	0	0	0	0	0	0	0	0	30.50109	28.26912	26.00138	23.70949	21.40814	19.11528	16.85219	14.64324	12.51529
14	0	0	0	0	0	0	0	0	0	0	0	0	32.72867	30.53032	28.2847	26.00041	23.6889	21.36489	19.04661	16.75598
15	0	0	0	0	0	0	0	0	0	0	0	0	0	34.92476	32.77141	30.56226	28.3031	26.00218	23.67072	21.32337
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37.07809	34.97819	32.81675	30.5971	28.32459	26.00703
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39.18028	37.13956	35.03393	32.8648	30.63503
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	41.22562	39.24744	37.20292	35.09202
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43.21077	41.29659	39.31606
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45.13423	43.28411
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46.9959
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3.2 Monte Carlo Simulation

$$\text{from before: } \log(S_t) = \log(S_0) + \left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t$$

gives the equation: $S_{k+1} = S_k e^{v\Delta t + \sigma \varepsilon_k \sqrt{\Delta t}} = S_k e^{v\Delta t + \sigma \varepsilon_k \sqrt{\Delta t}}$, where $\varepsilon_k \sim N(0,1)$

Using this equation we simulated 1000 different stock paths over the period of a year with intermediate time steps of a week, which can be seen below.

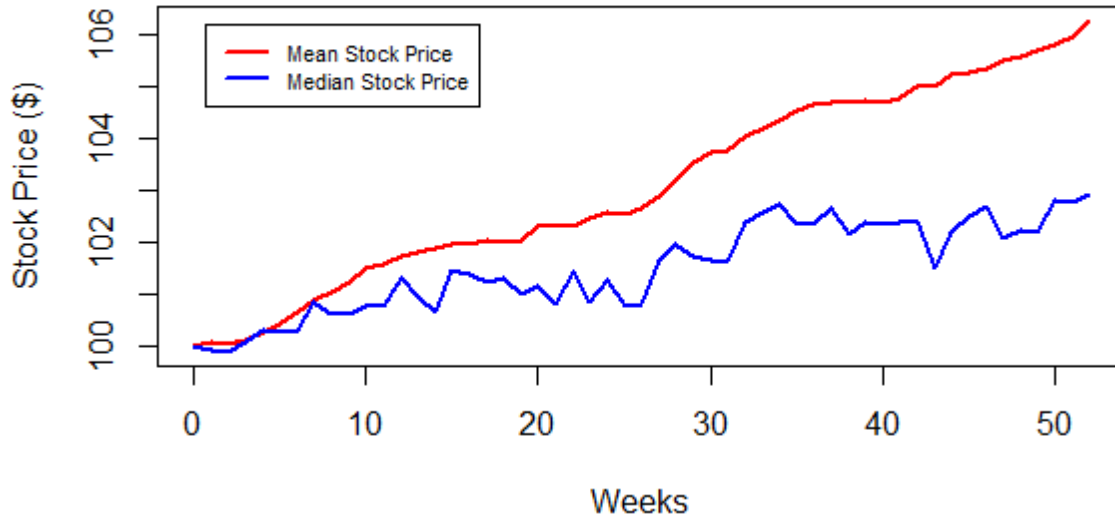


⁴<http://www.uio.no/studier/emner/sv/oekonomi/ECON4510/v11/undervisningsmateriale/lect2803.pdf>

⁵<http://www.columbia.edu/~mh2078/BlackScholesCtsTime.pdf>

⁶<https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities/put-call-options/v/put-call-parity>

Mean & Median Simulated Stock Price Paths



Then we found the price of the European call and put options:

$$C = \frac{1}{1000} \sum_{i=1}^{1000} \max \{K - S_{52}^i, 0\} \text{ and } P = \frac{1}{1000} \sum_{i=1}^{1000} \max \{S_{52}^i - K, 0\}$$

where the S_{52}^i denotes the price of the i^{th} simulated stock after 52 weeks. This attained

Monte Carlo (1000 Sims) European Call Price = \$ 13.13886

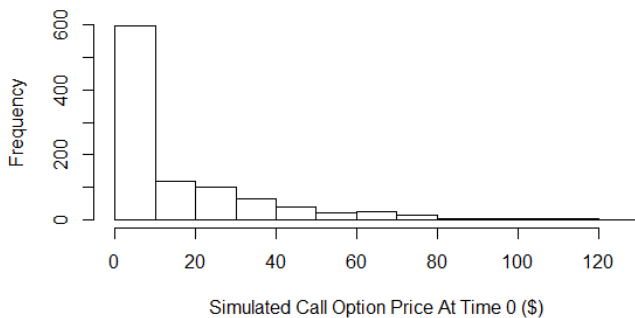
Monte Carlo (1000 Sims) European Put Price = \$ 7.323919

The Black-Scholes prices from Section 2 were:

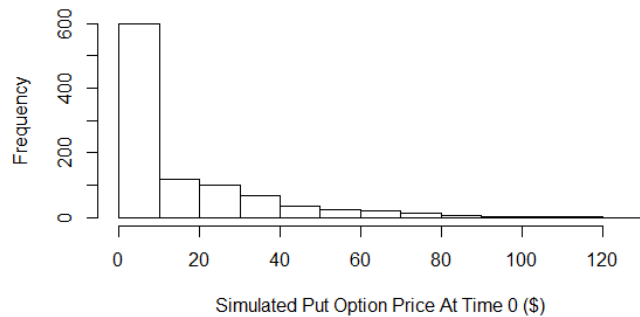
Black Scholes European Call Price = \$ 10.87056

Black Scholes European Put Price = \$ 8.890426

Histogram of Call Option Prices for 1000 Simulated Stock Paths



Histogram of Put Option Prices for 1000 Simulated Stock Paths

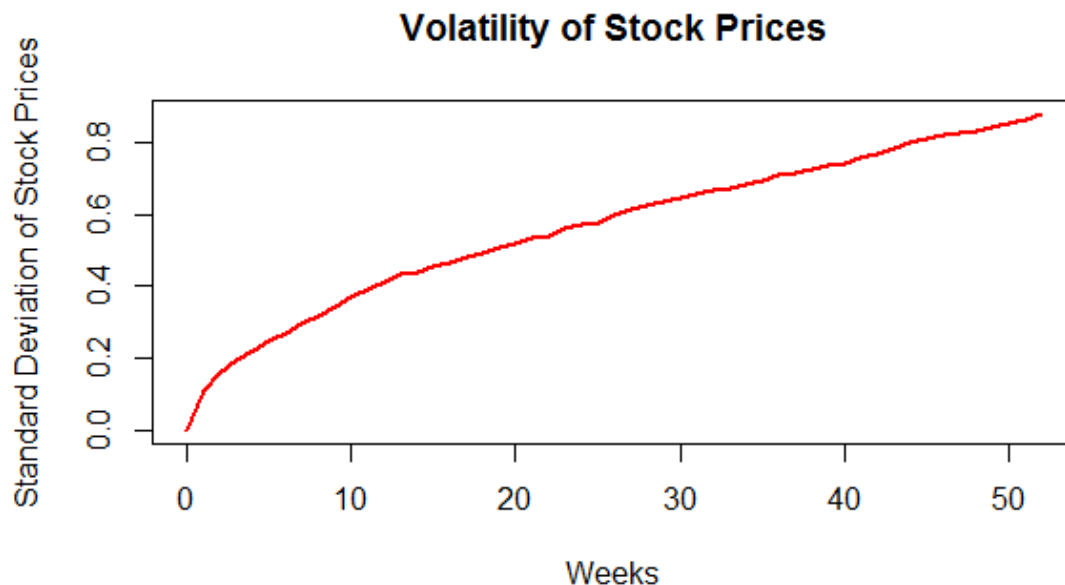


It's easy to see that the binomial tree option prices are much closer those of Black-Scholes, in comparison with the Monte-Carlo estimates—differences of \$0.04514 (0.415%) & \$0.045136 (0.508%) as opposed to \$2.2683 (20.866%) & \$1.5665 (17.620%). This is because the binomial and Black-Scholes models rely on similar assumptions and thus the binomial is, in essence, a discrete time approximation to the continuous process underlying the Black-Scholes model. The binomial model assumes that movements in the price follow a binomial distribution – as the number of trials becomes very large, this binomial distribution approaches the normal distribution assumed by Black-Scholes. When running the binomial tree with 250, and 1500 time-steps instead of 52, the prices do converge to the Black-Scholes ones.

Binomial Tree (250 Step) European Call Price = \$ 10.86115
Binomial Tree (250 Step) European Put Price = \$ 8.88102
Binomial Tree (1500 Step) European Call Price = \$ 10.86899
Binomial Tree (1500 Step) European Put Price = \$ 8.88858

With 1500 time-steps these differences shrink to \$0.00941 (0.087%) & \$0.001568 (0.018%), indicating convergence.

Another reason that Monte-Carlo fails to align with the Black-Scholes values is due to Monte-Carlo error. Two consecutive Monte-Carlo runs will yield two different option values as a result of the randomness inherent in the simulation – unlike the binomial model. This error is compounded since each new element is product of the previous elements. Thus, we would recommend running the chain multiple times and using averages. This is exhibited in the graph below which details the increasing variability between the stock price paths as time increases.



Increasing the number of intermediate time-steps would lead the Monte Carlo estimates to approach to those of the Black-Scholes. This error should shrink as the number of simulations increases. Boyle et al. (1997) and Broadie and Kaya^[6] (2006) proved the convergence of the error for Monte Carlo simulations is $\left(\frac{1}{\sqrt{n}}\right)$. Hence, this error should halve as number of paths quadruple.

Thus, assuming Black-Scholes values are the correct values, the binomial model is the much better approximation. However, in practice the issue is much more complicated. Firstly, the correctness of the Black-Scholes equation can be questioned due to unrealistic assumptions such as a constant stock volatility or its lack of versatility in dealing with more complex situations (such as stocks with dividends, American or exotic options, or a stochastic interest rate). The binomial model would be preferred to Monte-Carlo methods if the situation was simple and there exists an analytical technique for option valuing (or, in some cases, a numeric technique such as modified pricing tree) since Monte-Carlo methods will usually be too slow to be competitive. However, when the circumstances grow in complexity, binomial models will have to allow for too many possibilities and subsequently demand an extortionate amount of computing power to run. Moreover, Monte-Carlo can be used for any type of distribution, including changing and mixture distributions.

⁷http://www.columbia.edu/~mn2/broadie/Assets/broadie_kaya_exact_sim_or_2006.pdf

4 Hedging Options - Discrete Time

4.1 Binomial Model Put Option Replication

We setup a put option replicating portfolio, and used the equations derived (Section 3) to determine how much to invest in the underlying stock and the risk-free asset at each node:

$$x = \frac{C_u - C_d}{u - d} \text{ and } b = \frac{uC_d - dC_u}{R(u - d)}.$$

The corresponding portfolio value is simply $x + b$. In this case at each time t , we have;

$$x_t = -\Delta_t S_t, \text{ and } b_t = N_b^t K_t,$$

Δ_t - number of units of stock to short at time t , N_b^t - number of units of the risk-free asset (short-term US T-bills perhaps) to buy at t , and K_t is the face value of the risk-free asset at t .

The replication is perfect since x and b are determined from all the future possible outcomes (which we assume to be only an upward or downward price movement), and the put option value is derived from their expressions.

Proof:

Let C_u^t be the cost of put option at time t after an upward stock price movement from $t-1$. Let C_d^t be the cost of the put option at t after a downward stock price movement from $t-1$.

$$\text{Then } x_{t-1} = \frac{C_u^t - C_d^t}{u - d} \text{ and } b_{t-1} = \frac{uC_d^t - dC_u^t}{R(u - d)}$$

Then the replication error at time t given an upward stock price movement from time $t-1$, is given by

value of portfolio at time t just before replication – value of portfolio at time t just after replication

$$\begin{aligned} &= \Delta_{t-1} S_t + N_b^{t-1} K_t - C_u^t = \frac{x_{t-1}}{S_{t-1}} (S_t) + \frac{b_{t-1}}{K_{t-1}} (K_t) - C_u^t = x_{t-1} \left(\frac{S_t}{S_{t-1}} \right) + b_{t-1} \left(\frac{K_t}{K_{t-1}} \right) - C_u^t \\ &= x_{t-1}(u) + b_{t-1}(R) - C_u^t = \frac{C_u^t - C_d^t}{u - d}(u) + \frac{uC_d^t - dC_u^t}{R(u - d)}(R) - C_u^t \\ &= \frac{uC_u^t - uC_d^t + uC_d^t - dC_u^t}{u - d} - C_u^t = \frac{(u - d)C_u^t}{u - d} - C_u^t = C_u^t - C_u^t = 0 \end{aligned}$$

Similarly for a downward price movement from time $t-1$, we would get the replication error to be

$$x_{t-1}(d) + b_{t-1}(R) - C_d^t = \frac{dC_u^t - dC_d^t + uC_d^t - dC_u^t}{u - d} - C_d^t = \frac{(u - d)C_d^t}{u - d} - C_d^t = C_d^t - C_d^t = 0$$

We exported the differences between the tree of portfolio values just before replication and after replication (or the cost of the put option) at each time point to Excel to display equality between them. The sum of the absolute differences was $2.51021 \times 10^{-11} \approx 0$. The reason for this is that this is because R truncates some decimal places when storing values in variables and that causes slight discrepancies when calling these values and doing operations on them.

time0	time1	time2	time3	time4	time5	time6	time7	time8	time9	time10	time11	time12	time13	time14	time15	time16	time17	time18	time19
0	-1.42E-14	7.99E-15	8.88E-16	-6.22E-15	-7.99E-15	8.88E-16	-2.00E-15	-2.44E-15	-9.99E-16	1.44E-15	8.33E-16	3.61E-16	1.94E-16	-3.47E-16	4.16E-17	3.47E-17	-2.78E-17	-3.90E-17	1.08E-17
0	-1.42E-14	1.24E-14	-7.11E-15	2.66E-15	1.78E-15	7.99E-15	-5.33E-15	-3.55E-15	-3.55E-15	2.89E-15	-1.33E-15	0	-1.11E-16	-1.67E-16	4.44E-16	5.34E-16	-1.46E-16	-1.35E-16	-4.51E-17
0	0	1.07E-14	-1.78E-15	-8.88E-15	-5.33E-15	-1.42E-14	6.22E-15	-7.55E-15	6.66E-15	0	2.22E-15	-1.33E-15	1.11E-15	1.89E-15	5.55E-16	2.78E-16	2.50E-16	3.12E-16	2.43E-16
0	0	0	-5.33E-15	-1.78E-15	-3.55E-15	3.55E-15	1.78E-15	-1.07E-14	-7.11E-15	3.55E-15	4.00E-15	2.44E-15	-2.89E-15	1.67E-15	2.55E-15	5.00E-16	2.78E-16	5.83E-16	4.02E-16
0	0	0	0	-1.78E-15	1.95E-14	1.42E-14	-1.07E-14	-5.33E-15	-7.99E-15	-8.88E-16	-8.88E-16	1.33E-15	-3.55E-15	1.11E-15	-5.11E-15	2.22E-16	-5.55E-16	2.22E-16	5.83E-16
0	0	0	0	0	1.78E-14	2.49E-14	-5.33E-15	-1.78E-14	3.55E-15	8.88E-15	-1.24E-14	-1.15E-14	1.51E-14	7.99E-15	-1.78E-15	-2.22E-15	-4.44E-16	-2.22E-15	1.55E-15
0	0	0	0	0	0	2.84E-14	1.78E-14	3.55E-15	1.42E-14	1.78E-15	1.42E-14	1.87E-14	3.55E-15	-8.88E-15	2.66E-15	-4.00E-15	2.22E-15	6.44E-15	1.55E-15
0	0	0	0	0	0	0	1.42E-14	-2.13E-14	1.42E-14	-2.66E-14	7.11E-15	5.33E-15	8.88E-15	-3.55E-15	3.55E-15	-3.55E-15	4.00E-15	5.33E-15	2.66E-15
0	0	0	0	0	0	0	0	-1.42E-14	2.84E-14	1.42E-14	7.11E-15	-1.42E-14	1.07E-14	-1.24E-14	0	0	-2.04E-14	6.22E-15	-8.88E-16
0	0	0	0	0	0	0	0	0	0	2.49E-14	2.49E-14	1.78E-14	2.13E-14	4.26E-14	-3.91E-14	-1.24E-14	3.55E-15	-2.13E-14	-5.33E-15
0	0	0	0	0	0	0	0	0	0	3.20E-14	2.13E-14	-2.13E-14	-3.55E-15	2.84E-14	1.78E-14	-5.33E-15	-1.07E-14	8.88E-15	1.78E-14
0	0	0	0	0	0	0	0	0	0	0	2.13E-14	-3.55E-14	-3.55E-14	-1.78E-14	2.13E-14	2.13E-14	5.68E-14	-1.07E-14	3.20E-14
0	0	0	0	0	0	0	0	0	0	0	0	-4.26E-14	0	-7.11E-15	-2.13E-14	2.49E-14	-7.11E-15	1.07E-14	1.07E-14
0	0	0	0	0	0	0	0	0	0	0	0	0	7.11E-15	4.97E-14	2.84E-14	6.75E-14	-3.55E-15	1.78E-14	3.55E-15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.68E-14	2.84E-14	-4.26E-14	-2.49E-14	4.97E-14	-7.11E-15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.84E-14	-2.13E-14	-7.11E-15	-7.11E-15	-7.11E-15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2.13E-14	-2.84E-14	7.11E-15	4.26E-14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2.84E-14	-1.42E-14	1.42E-14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.42E-14	4.97E-14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4.97E-14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

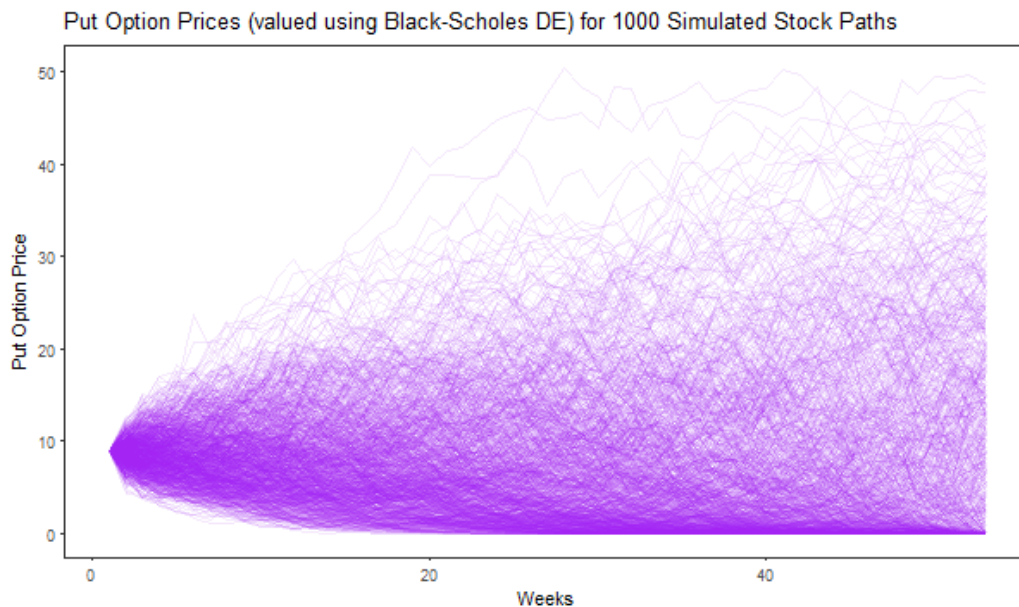
4.2 Monte-Carlo Simulation & Black-Scholes Pricing for Put Option Replication

1000 stock price paths were simulated in the same fashion as before, at each time point we priced the value of the put option. We also calculated x and b at each time point; the dollar amounts to invest in the underlying stock and risk-free asset—given by formulae below:

$$x_t = -S_t N(-d_1^t) \text{ and } b_t = 100e^{-0.02(1-t)} N(-d_2^t)$$

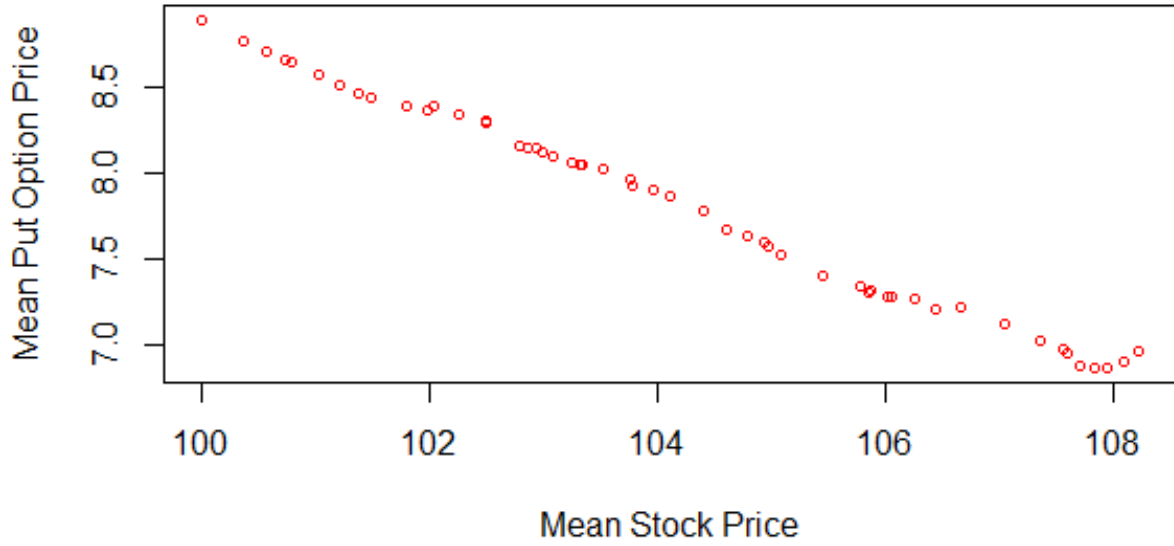
$$d_1^t = \frac{\log\left(\frac{S_t}{100}\right) + (0.02 - 0.5(0.25^2)(1-t))}{0.25\sqrt{1-t}} \text{ and } d_2^t = d_1^t - 0.25\sqrt{1-t}$$

Then the portfolio value at time t is simply $x_t + b_t$. The put option values derived from the simulated stock price paths are displayed below.

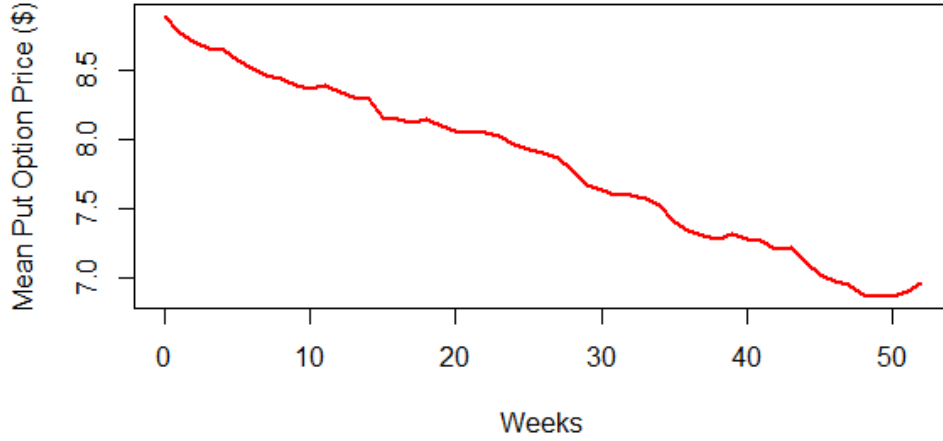


We conducted two checks to investigate whether low (and high) stock prices were indeed associated with high (and low) put option prices, and that the value of the put options decayed over time (especially since stock has a positive drift). Both of these occurred.

Mean Put Option Price (valued using Black-Scholes DE) vs Mean Stock Price



Mean Put Option Price (valued using Black-Scholes DE) for 1000 Simulated Stock Paths



$N(-d_1^t)$ represents the probability that the put option will be exercised, and $N(-d_2^t)$ denotes the probability that stock price will be above strike price at time $T = 1$ year. Thus, we are shorting the stock in a proportional amount to its expected value, and similarly we are investing in the risk-free asset proportionally to the expected present value of a zero-coupon bond paying $K = 100$ in $T - t (= 1 - t)$ years. Investing this way replicates the put option closely. However, adjusting the portfolio discretely (every week) will lead to replication errors.

The replication error at time t for a certain simulated price path is given by the value of the portfolio at time t before replication – value of portfolio at time t just after replication (or adjusting).

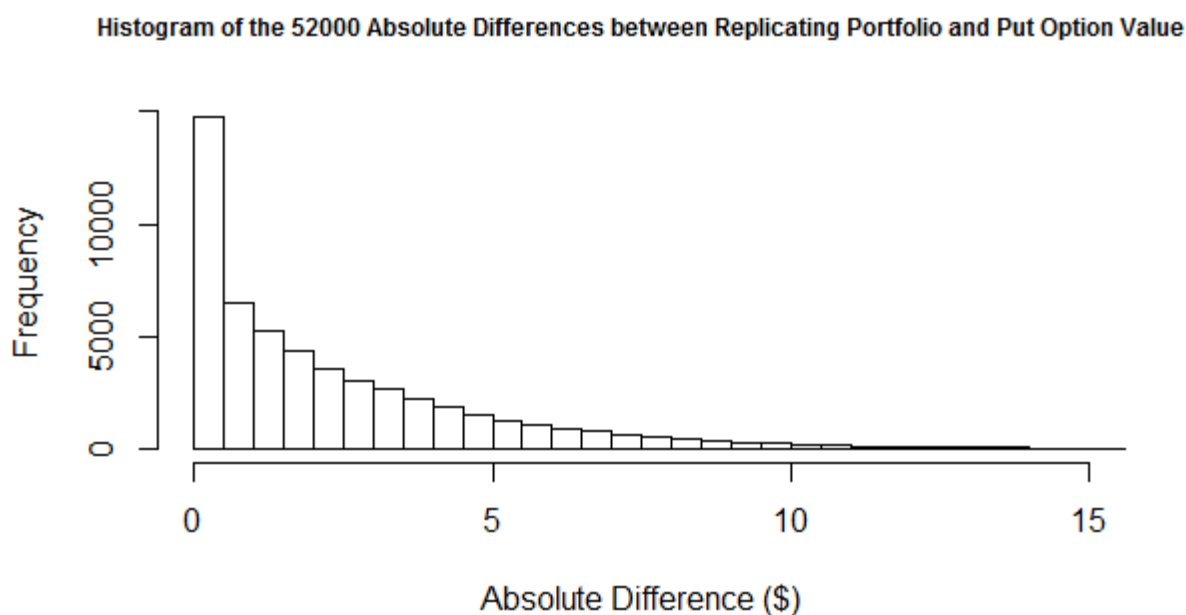
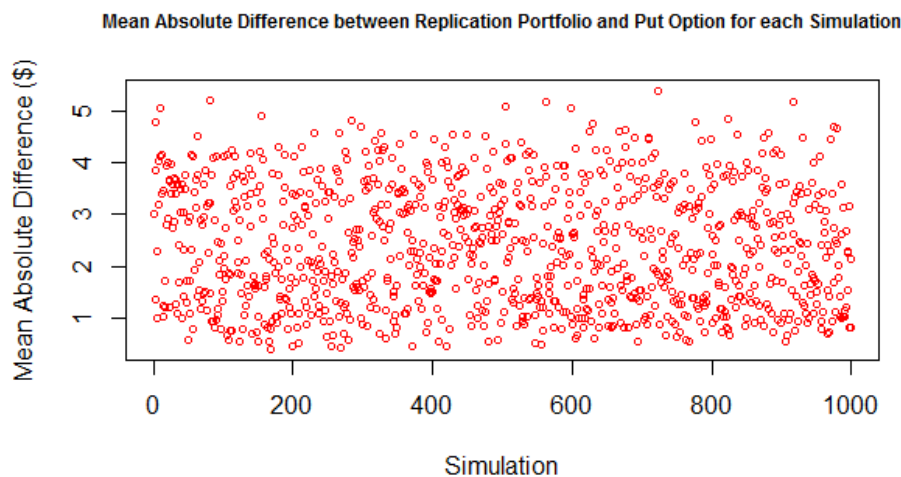
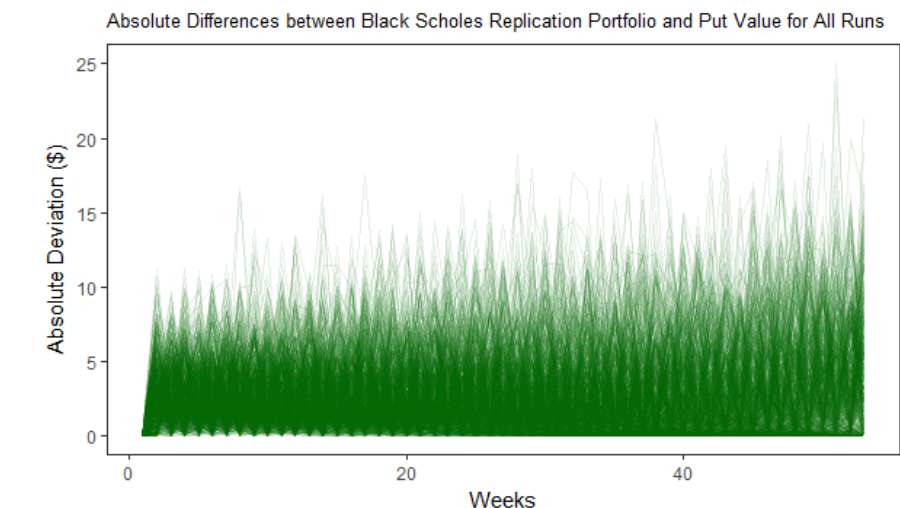
$$\begin{aligned} &= \Delta_{t-1}S_t + N_b^{t-1}K_t - (x_t - b_t) = \frac{x_{t-1}}{S_{t-1}}(S_t) + \frac{b_{t-1}}{K_{t-1}}(K_t) - (x_t - b_t) \\ &= x_{t-1}\left(\frac{S_t}{S_{t-1}}\right) + b_{t-1}(R) - (x_t - b_t) = \varepsilon_{i,t} \end{aligned}$$

The mean absolute deviation was much larger than in the binomial case with

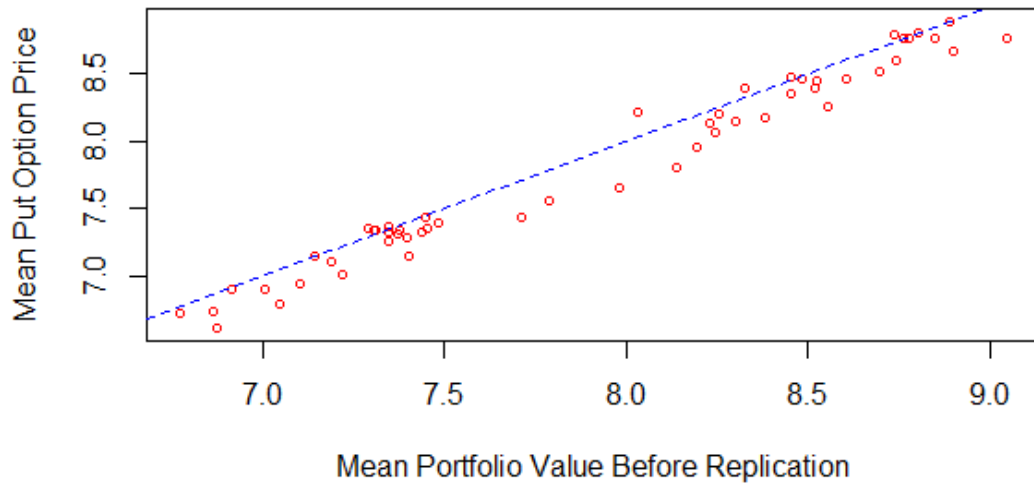
$$\frac{1}{52 * 1000} \sum_{i=1}^{1000} \sum_{j=\frac{1}{52}}^1 |\varepsilon_{i,j}| = \$2.37026$$

The graph of absolute deviations is below, along with a plot of mean absolute deviation for each simulation. There is no obvious pattern and they seem evenly clustered and spaced apart. The histogram of all absolute deviations

shows most deviations are very small, and as they gradually get larger, there are less of them. The largest absolute deviation was \$25.08572. The plot of mean portfolio values before and after replication demonstrates a linear relationship (nearly lying on line $y = x$, but often deviate). Furthermore, mean portfolio values before replication tend to be slightly lower than after, as most lie below the line.



Mean Portfolio Value Before Replication vs After Replication



4.3 Augmented GBM for Put Option Replication

The Black-Scholes model assumes that asset price follows GBM with constant volatility. However, a number of empirical studies documented systematic abnormalities and biases about the asset returns in the B-S framework. The implied volatilities from the market prices of options vary with respect to strike prices and maturities² (which collectively give rise to the volatility surface), contrary to the assumption of constant volatility. Empirical studies found that asset returns usually have a higher kurtosis and a heavier tail compared to the normal distribution which is assumed by the B-S model. Thus, we wanted to capture this extra volatility by employing a stochastic volatility (SV) model (volatility governed by a separate stochastic process). Various SV models have been studied in the literature such as Hull and White, Scott, Johnson and Shanno, Melino and Turnbull, Stein and Stein, Heston, and Nicolato and Venardos. Only the Stein and Stein model and the Heston model possess analytic closed form solutions, and are more extensively used due to their computationally efficient construct. Among these two models, Heston's model boasts advantages –it prevents negative volatility and allows nonzero correlation between asset price and its instantaneous volatility. It also models volatility as a mean-reverting process. An assumption consistent with behaviour observed in financial markets. If volatility were not mean-reverting, markets would be characterized by a considerable amount of assets with volatility exploding or going near zero. In practice, these cases are quite rare and generally short-lived. In general, Heston's model provides a versatile modelling framework that can accommodate many of the specific characteristics that are typically observed in the behaviour of financial assets. In particular, the parameter δ (volatility of the volatility) controls the kurtosis of the underlying asset return distribution, while ρ (correlation with asset price) sets its asymmetry. These benefits come at the expense of higher complexity. Compared with BSM, the implementation of the Heston model requires more sophisticated mathematics and it involves a challenging process to calibrate the model to fit market prices. The criticisms are that parameters allow the instantaneous volatility of the stock to become zero with a positive probability. From a practical point of view, the most challenging property of Heston's model is the interdependence of its parameters and the resulting inability to give these parameters a real idiosyncratic meaning. In order to take into account all the features from the volatility surface, the Heston model might be too rigid a framework – It may be necessary to add degrees of freedom to the original model.

The instantaneous variance of the underlying asset dv_t follows a Cox–Ingersoll–Ross model, and is given below. θ is the expected value of v_t as t tends to infinity, we set this, as well as v_0 , to $0.25^2 = 0.0625$ to be consistent with previous analysis. κ is the rate at which v_t reverts to θ . δ is the volatility of the volatility.

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S \text{ and } dv_t = \kappa(\theta - v_t)dt + \delta \sqrt{v_t} dW_t^v \text{ with } \text{Corr}(dW_t^S, dW_t^v) = \rho$$

$$dW_t^S = \varepsilon_t^{(1)} \sqrt{\Delta t} \text{ and } dW_t^v = \rho^2 \varepsilon_t^{(1)} \sqrt{\Delta t} + \sqrt{(1 - \rho^2)} \varepsilon_t^{(2)} \sqrt{\Delta t} \text{ where } \varepsilon_t^{(1)}, \varepsilon_t^{(2)} \sim \text{IID } N(0,1)$$

We chose δ to be 0.01 as that seemed to give enough variation to show clear differences between the stochastic volatility and constant volatility. Then we required $2\kappa\theta > \delta^2$ to ensure that instantaneous volatility could only be positive. Thus, we selected $\kappa = 1$. We set ρ to -0.3 since in equity markets volatility tends to increase when there

² <https://arxiv.org/ftp/arxiv/papers/1502/1502.02963.pdf>

are high drops in equity prices, which explains the downward skew of implied volatility³. There was two ways to draw stock prices using this model – discretized or continuous version. As mentioned in *Section 1* we had to choose seemingly arbitrary coefficients.

Discretized Version:

$$v_{t+\Delta t} = v_t + \kappa(\theta - v_t)\Delta t + \delta\sqrt{v_t}\left(\rho^2\varepsilon_t^{(1)}\sqrt{\Delta t} + \sqrt{(1-\rho^2)}\varepsilon_t^{(2)}\sqrt{\Delta t}\right)$$

$$S_{t+\Delta t} = S_t e^{\left(\mu - \frac{1}{2}v_{t+\Delta t}\right)\Delta t + \sqrt{v_{t+\Delta t}}\varepsilon_t^{(1)}\sqrt{\Delta t}}$$

Continuous Version:

Draw $S_{t+\Delta t}$ from a non-central Chi-Squared distribution with $\frac{4\kappa\theta}{\delta^2}$ degrees of freedom, and non-centrality parameter $\frac{4\kappa S_t}{(1-e^{-\kappa\Delta t})\delta^2}e^{-\kappa\Delta t}$, as time grows, this can be approximated by a Gamma distribution. This approximation isn't required since R can draw random deviates from the non-central Chi-Squared distribution directly in R. Note that we only used the discretized version.

We also wanted to introduce a stochastic interest rate, hence adopting the Vasicek model, since we had already made use of the CIR model in *Section 1*. The Vasicek model has many similar properties to the Cox–Ingersoll–Ross model except that it allows interest rates to become negative. Furthermore, dependence on a single factor greatly limits the possible shapes of the yield curve and often leads to situations where the theoretical yield curve does not correspond to the market yield curve⁴. Making the coefficients time-dependent (even one of them – usually the long term variance parameter) enables one to have an exact fit⁵. Another weakness in the model is the property that yields of all maturities are perfectly correlated. This is an unrealistic assumption regarding behaviour of yields⁶. Käppi also finds that the single factor Vasicek model fits market data poorly compared to multifactor versions⁷.

We chose $c = 0.8$ and $a = 0.1$ as that seemed to give enough variation to show clear differences between stochastic interest rate and constant interest rate. We set b and r_0 to be 0.02 for obvious reasons. We selected $\rho = -0.3$ because it made more sense for the price of the stock to decrease if interest rates increase due to more debt strain on companies (which decreases value of future cashflows), and consumers (which deflates demand)⁸. Note again choosing seemingly arbitrary coefficients since we could not avail of published estimates of current market values due to the constraint of allowing the long term interest rate to be 0.02.

Moreover, the Black-Scholes formula is valid if the stock price can only change by a small amount over a small interval of time. However, many empirical studies suggest that asset returns distribution is skewed to the left, and has a higher peak and two heavier tails than those of the normal distribution⁹. This along with the fact that the implied volatility curve is a convex curve of the strike price (and not simply constant) suggests the applicability of a jump-diffusion model. We considered two of the most well-known models: the Merton Jump-Diffusion-Model (1975), which can be seen as a foundation for jump-diffusion models, and the Kou-Double-Exponential Jump-Diffusion-Model (2002). They assume the stock follows GBM but incorporates a jump term that models the arrival of important information into the market which will have an abnormal effect on the price.

We assume that each jump in the stock price is independent of the others (not necessarily realistic but we make the assumption for modelling purposes). We also require that in a given short space of time δt , the likelihood of a jump is roughly proportional to the length of δt . The proportionality constant is denoted by λ and is called the jump intensity. If δt is taken to be small, the probability of two jumps in the interval is negligible. In fact, we assume that the number of jumps is a Poisson counting process¹⁰.

Now for a jump term the relative change in stock price at time t is given by;

³ http://www.mathnet.or.kr/mathnet/thesis_file/02_B08-109.pdf

⁴ http://www.bbk.ac.uk/ems/for_students/msc_finEng/pricing_emms014p/ab7.pdf

⁵ <https://www.doria.fi/bitstream/handle/10024/43257/nbnfi-fe200901141021.pdf?sequence=3>

⁶ <https://nccur.lib.nccu.edu.tw/bitstream/140.119/35086/6/35100306.pdf>

⁷ <https://www.doria.fi/bitstream/handle/10024/43257/nbnfi-fe200901141021.pdf?sequence=3>

⁸ <http://www.investopedia.com/investing/how-interest-rates-affect-stock-market/>

⁹ <https://brage.bibsys.no/xmlui/bitstream/handle/11250/223257/masterthesis114.pdf?sequence=1>

¹⁰ Therefore another property our counting process must satisfy is, $N_t + \delta t = N_t$ with probability $1 - \lambda\delta t - o(\delta t)$, $N_t + 1$ with probability $\lambda\delta t + o(\delta t)$ and $N_t + k$ with probability $o(\delta t)$.

$$\frac{dS_t}{S_t} = \frac{J_t S_t - S_t}{S_t} = J_t - 1.$$

Thus, both the Merton and Kou models are of the form:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + (J_t - 1)S_t dN_t \text{ where } N_t \sim \text{Poisson}(\lambda)$$

Consider the formula derived by Cont and Tankov¹¹ using Itô's formula for Jump-Diffusions:

$$df(X_t, t) = \frac{\partial f(X_t, t)}{\partial t} dt + a_t \frac{\partial f(X_t, t)}{\partial x} dt + \frac{b_t^2}{2} \frac{\partial^2 f(X_t, t)}{\partial x^2} dt + b_t \frac{\partial f(X_t, t)}{\partial x} dW_t + [f(X_{t-} + \Delta X_t) - f(X_{t-})].$$

$$d \log(S_t) = \left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW_t + [\log(J_t S_t) - \log(S_t)] = \left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW_t + \log(J_t)$$

$$S_t = S_0 e^{\left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW_t + \log(J_t)} \text{ but } \log(J_t) = \sum_{i=1}^{N_t} \log(Y_i)$$

$$\text{Thus } S_t = S_0 e^{\left[\left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW_t \right]} \prod_{i=1}^{N_t} Y_i = S_0 e^{\left[\left[\mu - \frac{1}{2} \sigma^2 \right] dt + \sigma dW_t \right]} \prod_{i=1}^{N_t} e^{V_i}$$

In Merton model:

$$\log(Y_i) \sim N(\alpha, \beta^2) \text{ so } Y_i \sim \text{lognormal}(e^{\alpha + 0.5\beta^2}, (e^{\beta^2} - 1)e^{2\alpha + \beta^2})$$

In Kou model:

$$V_i = \log(Y_i) = \begin{cases} \tau^+ & \text{with probability } p \\ -\tau^- & \text{with probability } 1-p \end{cases} \text{ where } \tau^+ \sim \text{Exp}(\eta_1) \text{ and } \tau^- \sim \text{Exp}(\eta_2)$$

In Kou's model we require $\eta_1 > 1$ and $\eta_2 > 0$ to ensure that $E\{e^{V_i}\} < \infty$ and thus $E\{S_t\} < \infty$. This essentially means that the average upward jump cannot exceed 100%, which is reasonable, because this is not observed in the stock market¹² (and average downward jump is greater than zero). In fact, we selected the daily intensity $\lambda = 0.2$, the probability of an upward jump $p = 0.52$, and the exponential parameters $\eta_1 = 100$ and $\eta_2 = 100$ (similar to suggestions here¹³) so that

$$E\{\tau^+\} = E\{\tau^-\} = 0.01, \text{ and thus } E\{Y_i\} = E\{e^{V_i}\} = E\{e^{p\tau^+ - (1-p)\tau^-}\} = M_{\tau^+}(p)M_{\tau^-}(-(1-p)) = \frac{\eta_1}{\eta_1 - p} * \frac{\eta_2}{\eta_2 + (1-p)} = \frac{100}{100 - 0.52} * \frac{100}{100 + 0.48} \approx 1.0004 \text{ and } E\{Y_i^2\} = E\{e^{2V_i}\} = \frac{\eta_1}{\eta_1 - 2p} * \frac{\eta_2}{\eta_2 + 2(1-p)} = \frac{100}{100 - 2*0.52} * \frac{100}{100 + 2*0.48} \approx 1.0009$$

so that $\text{Var}\{Y_i\} \approx 0.00005$ which seems reasonable (to see on average $0.2 * 5 = 1$ 0.04% upward jump a week in stock price a week). In order to be consistent with mean and variance of the Kou model, we set

$$\alpha \approx 0.0003749, \text{ and } \beta \approx 0.007068 \text{ so that } E\{Y_i\} = e^{\alpha + 0.5\beta^2} = 1.0004, \text{ and } \text{Var}\{Y_i\} = (e^{\beta^2} - 1)e^{2\alpha + \beta^2} = 0.00005.$$

The Merton model is useful especially for pricing options with a short time to maturity considering that this model does not incorporate the volatility clustering effect¹⁴. Although, both models retain the analytical tractability of the Black-Scholes model. In particular, in the Kou model, closed-form solutions are possible to be obtained due to the memoryless property of the double exponential distribution. Moreover, the double exponential jump-diffusion model is internally self-consistent. This means that the model is arbitrage-free and can be embedded in a rational expectations equilibrium setting. Another motivation for the Kou model comes from behavioural finance. Empirical studies show that the markets tend to have overreaction and under-reaction to outside news. The high peak of the double exponential distribution can be used to model the under-reaction, whereas the heavy tails can be used to

¹¹ <http://eprints.maths.ox.ac.uk/659/1/martin.pdf>

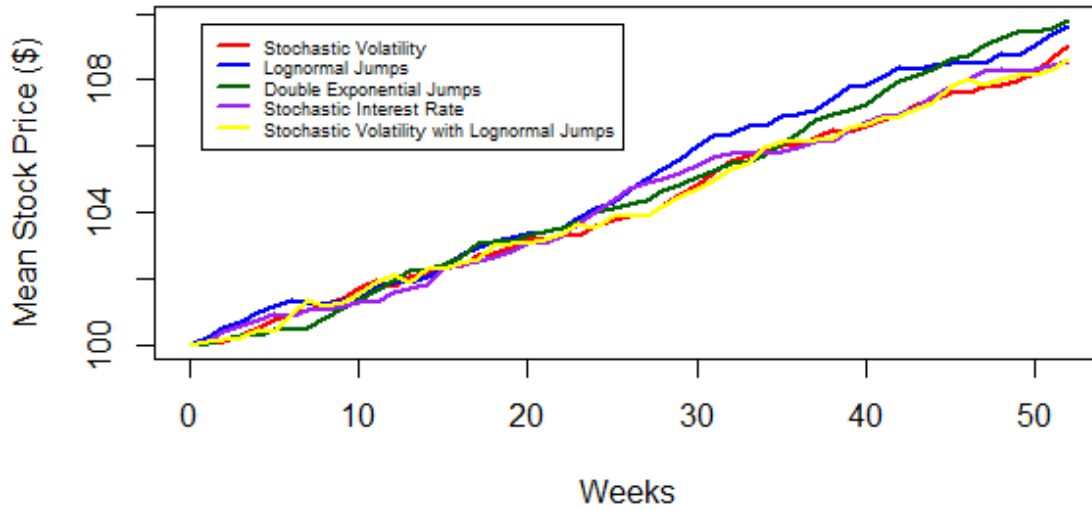
¹² <https://brage.bibsys.no/xmlui/bitstream/handle/11250/223257/masterthesis114.pdf?sequence=1>

¹³ <http://www.diva-portal.org/smash/get/diva2:343268/fulltext01.pdf>

¹⁴ <http://www.diva-portal.org/smash/get/diva2:343268/fulltext01.pdf>

model the overreaction¹⁵. Although, because the observed implied volatility surface is skewed and tends to flatten out for longer maturities, the two models abilities to produce accurate results are reduced¹⁶.

Mean Simulated Stock Price Paths



It is understandable that each of the simulated stocks tended upwards. The stochastic interest, although correlated with the stock price, did not cause an upward drift. The expected stock price with a stochastic interest rate was

$$E\{S_1\} = 100e^{0.07 - 0.5 \times 0.25^2} = \$103.95.$$

Expected final instantaneous volatility was given by¹⁷

$$E\{v_t|v_0\} = v_0e^{-\kappa t} + \theta(1 - e^{-\kappa t}) \text{ so } E\{v_1|v_0 = 0.25\} = 0.25e^{-4.55} + 0.25(1 - e^{-4.55}) = 0.25.$$

However, since the stock price follows a Heston model its expected value was given by the following series of equations¹⁸:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t} S_t dW_t^{(S)} \\ dV_t &= (aV_t + b)dt + c\sqrt{V_t} dW_t^{(V)} \end{aligned}$$

$$\mathbf{E}S_t^p = S_0^p e^{prt - \frac{\rho p}{c}(v_0 + bt)} L\left(-\frac{\rho p}{c}, \frac{\rho}{2} + \frac{\rho ap}{c} - \frac{(1 - \rho^2)p^2}{2}\right)$$

with:

$$L(p, q) = \mathbf{E} \exp\left(-pV_t - q \int_0^t V_s ds\right) = [A(p, q, t)]^{-\frac{2b}{c^2}} \exp[B(p, q, t)]$$

¹⁵ <http://www.columbia.edu/~sk75/MagSci02.pdf>

¹⁶ <http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/5375/Sideri.pdf?sequence=2>

¹⁷ https://en.wikipedia.org/wiki/Cox%E2%80%93Ingersoll%E2%80%93Ross_model

¹⁸ <http://ieeexplore.ieee.org/document/7019912/?reload=true>

$$A(p, q, t) = \frac{e^{at/2}}{P} [P \cosh(Pt/2) - a \sinh(Pt/2) + c^2 p \sinh(Pt/2)]$$

$$B(p, q, t) = -\frac{v_0}{c^2} \left\{ P + a - \frac{(P + a - c^2 p)e^{-(P-a)t/2}}{A(t)} \right\}$$

$$P = P(q) = \sqrt{a^2 + 2c^2 q}.$$

$$P = 1.002935192, A = 1.001489162, B = -1.874880459, L(30, 29.395) = 0.02387441424$$

Thus, we found for the stock following a Heston model, $E\{S_1\} = \$108.8773761$.

Expected stock price for stock following GBM but with lognormal jumps¹⁹:

$$(\lambda \text{ here is } 1 \cdot 52 = 52) \quad E\{S_1\} = 100e^{\mu - 0.5\sigma^2 - \lambda(e^{\alpha + 0.5\beta^2} - 1) + \lambda\alpha} = \$103.8156883$$

Expected stock price for stock following GBM but with double exponential jumps²⁰:

$$E\{S_1\} = 100e^{\mu - 0.5\sigma^2 - \lambda(\frac{p}{\eta_1} + \frac{1-p}{\eta_2})} = \$101.811207$$

Not all these estimates matched the end result. However, they do at least explain the upward trend that goes along with the 7% drift of the original model.

The sum of the absolute deviations for all cases is given below $\left(\frac{1}{52 \cdot 1000} \sum_{i=1}^{1000} \sum_{j=\frac{1}{52}}^1 |\varepsilon_{i,j}|\right)$

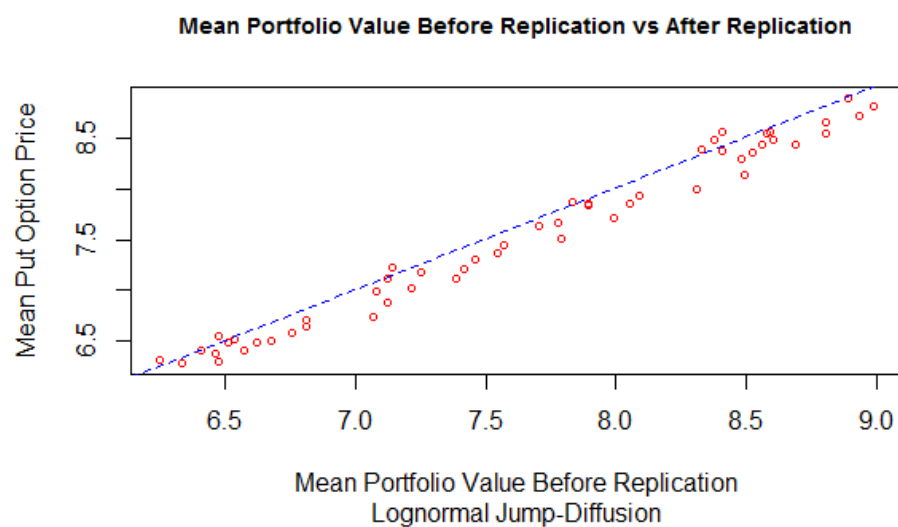
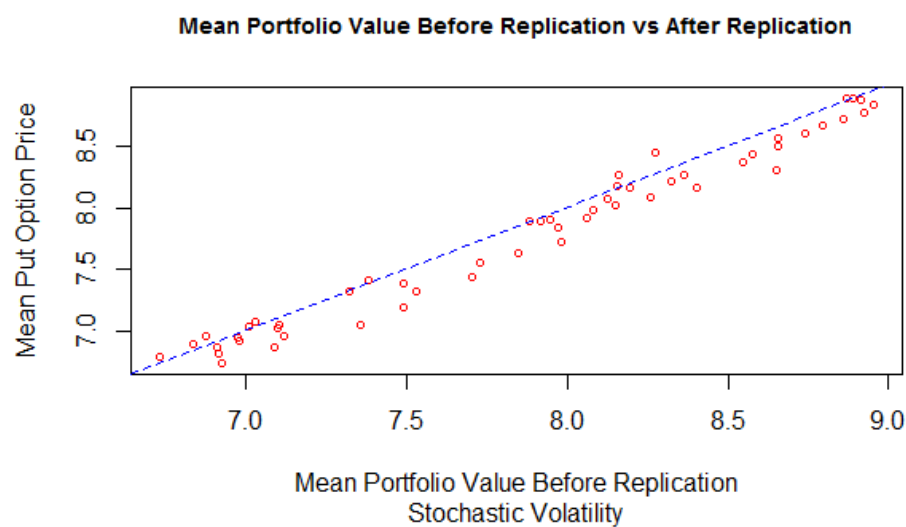
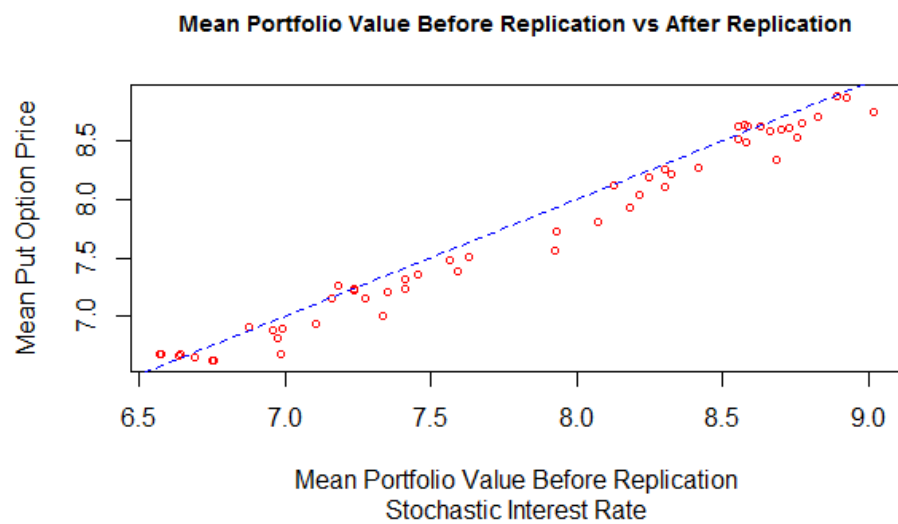
Stochastic Interest Rate	2.356471	Lognormal Jumps	2.358183
Stochastic Volatility	2.379574	Double Exponential Jumps	2.538249
		Stochastic Volatility with Lognormal Jumps	4.106620
Stock Price Volatility as measured by $\left(\frac{1}{52 \cdot 1000} \sum_{i=1}^{1000} \sum_{j=\frac{1}{52}}^1 \left \frac{S_{i,j} - S_{i-1,j}}{S_{i-1,j}} \right \right)$:			
Geometric Brownian Motion	0.02774374	Lognormal Jumps	0.02811315
Stochastic Interest Rate	0.02774374	Double Exponential Jumps	0.0297409
Stochastic Volatility	0.02773202	Stochastic Volatility with Lognormal Jumps	0.04800325

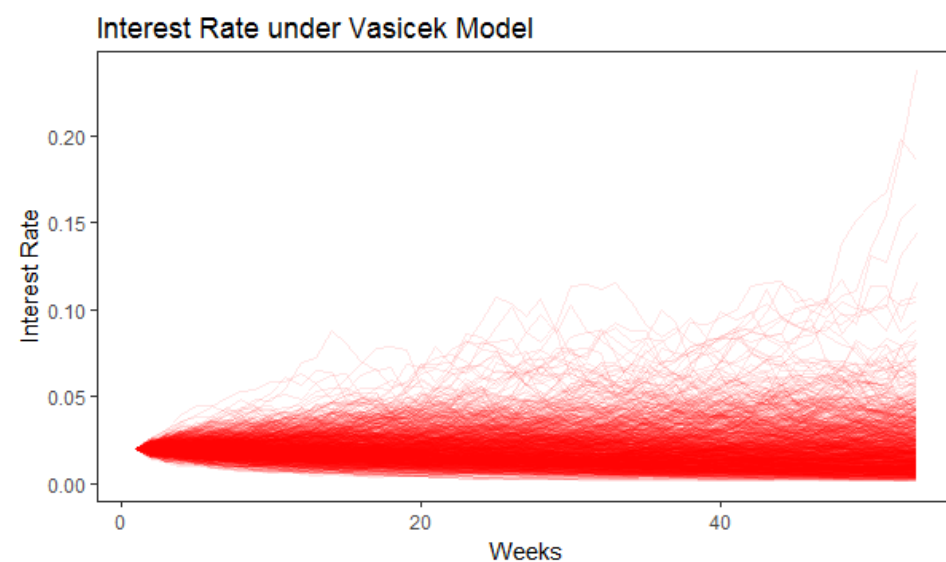
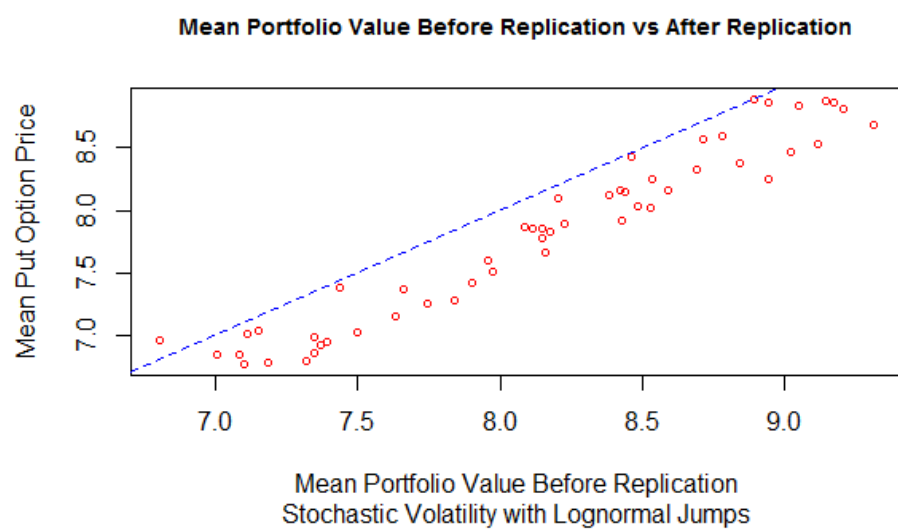
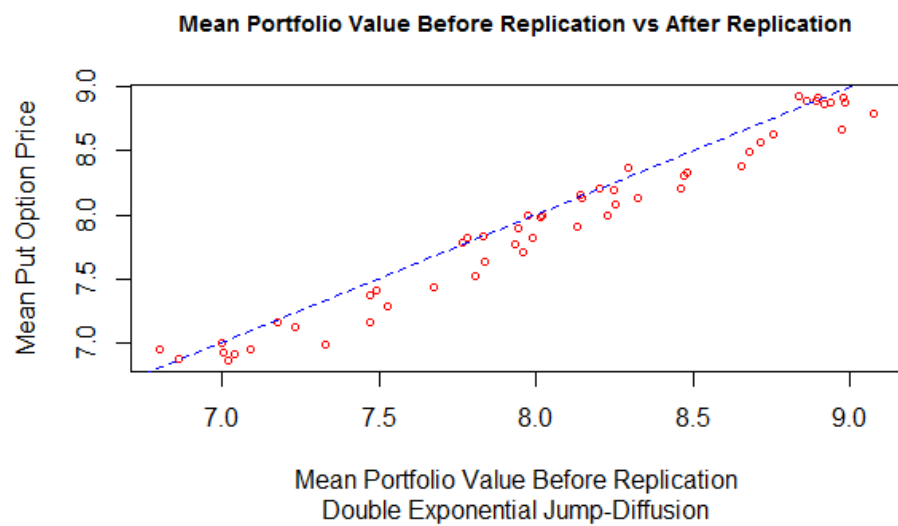
What may seem peculiar is that the mean absolute deviations were higher for the original GBM stock compared with that when interest rate is stochastic or the lognormal jump-diffusion stock. The stochastic volatility was marginally higher, with the double-exponential-jump-model higher again. The validity of these statistics are confirmed in the histograms and plots of the deviations below. These relatively low deviations are hardly due to random chance with a sample size = 52000. One thing to note, firstly, is that the simulated prices associated with the interest rates were the exact same as the original ones. The second point is that the interest rates, and jumps were selected with a relative low variance in hindsight, so the overall effect was minimal – this can be seen in the graph of interest rates, and graph of mean jumps (they do not veer much from their expected values). Moreover, this low variability is consistent with the table above which details our make-shift volatility measure.

Perhaps the jump models did not create much disorder as the jump intensity was too low – particularly with a low variance in jumps. However, as is displayed in the plots of portfolio value before and after replication, the deviations in the stochastic volatility, and even moreso, in the jump-diffusion models, tend to be much more negative than in the cases of the original GBM and stochastic interest rate. Thus, the portfolio is penalised (and certainly benefits less from chance) for being unbalanced in the face of volatility. On the whole, we would have expected the jump models to both have larger deviations than the stochastic volatility model due to changes in stock price being more abrupt and volatile. The stochastic volatility with lognormal jumps suffered much larger negative deviations than the rest, as expected. This is shown in its deviation histogram and portfolio value plot. The reason for this is its inflated variance as exhibited in the table above.

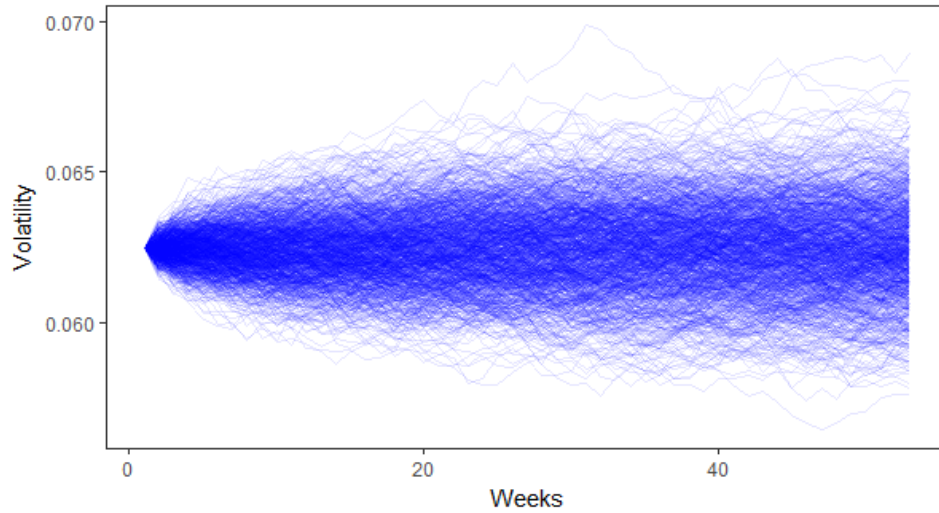
¹⁹ <http://www.maxmatsuda.com/Papers/Intro/Intro%20to%20MJD%20Matsuda.pdf>

²⁰ <http://cyrus.cob.calpoly.edu/JUMPMLE/AF-Ramezani-Zeng.pdf>

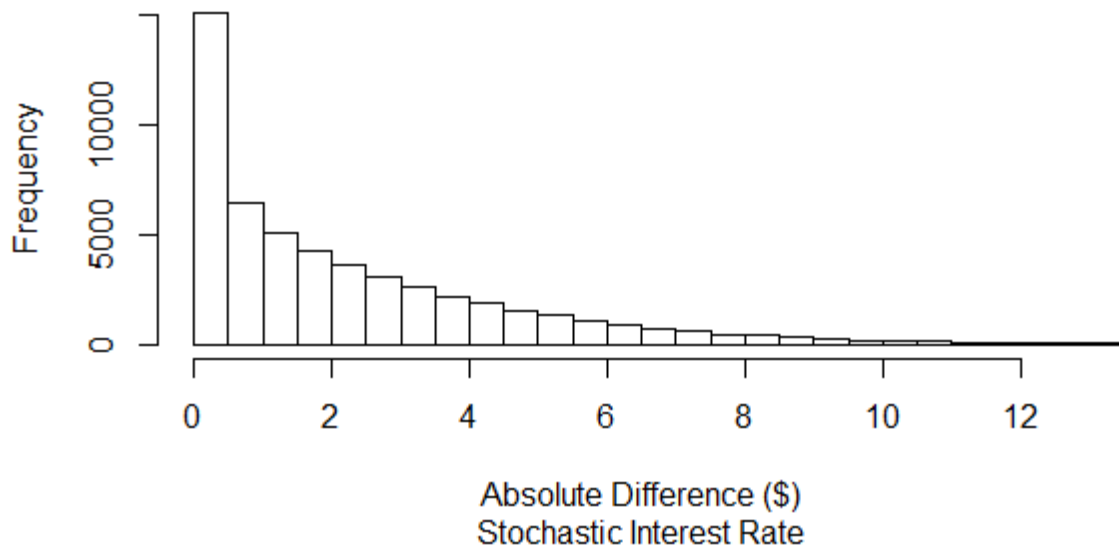




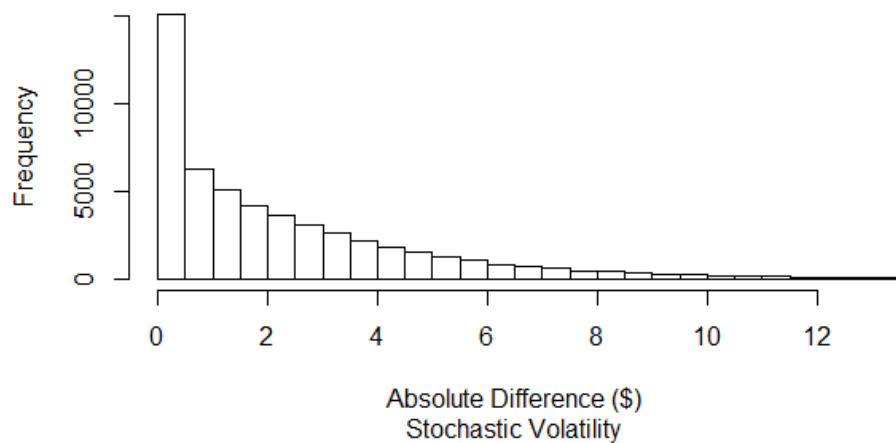
Instantaneous Volatility under Heston model



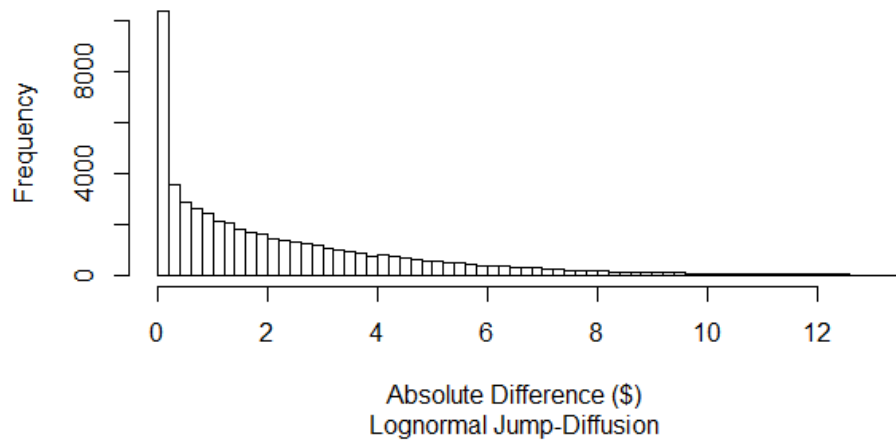
Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value



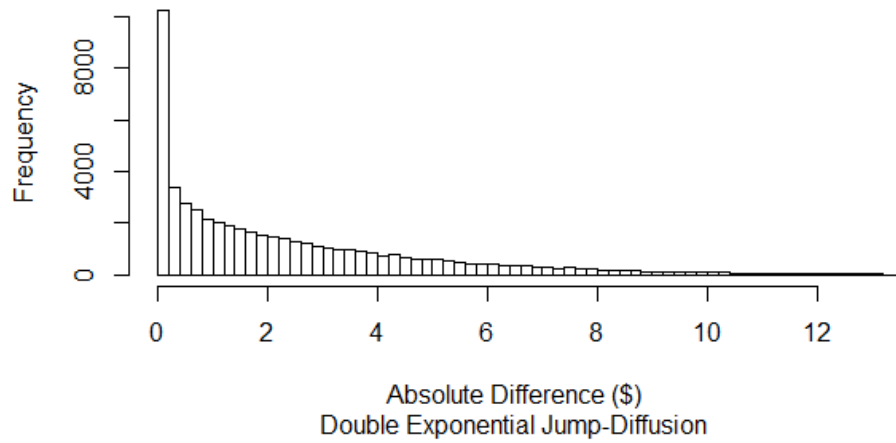
Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value



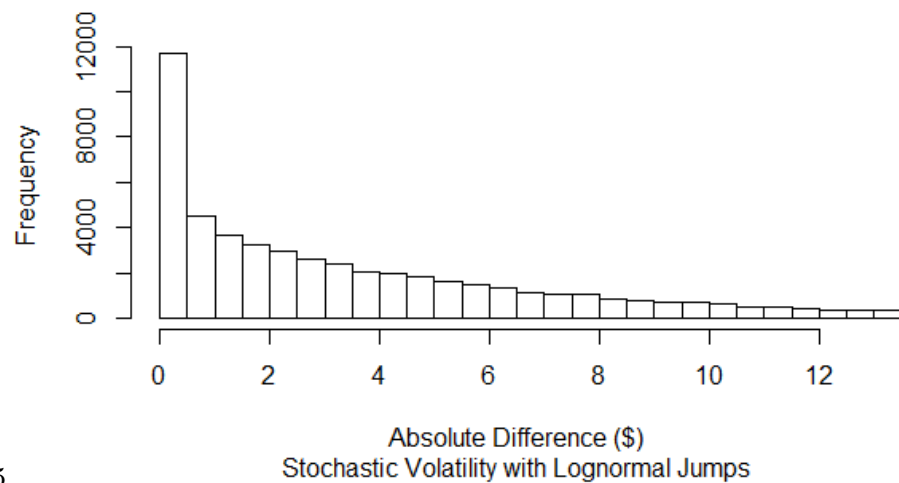
Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value

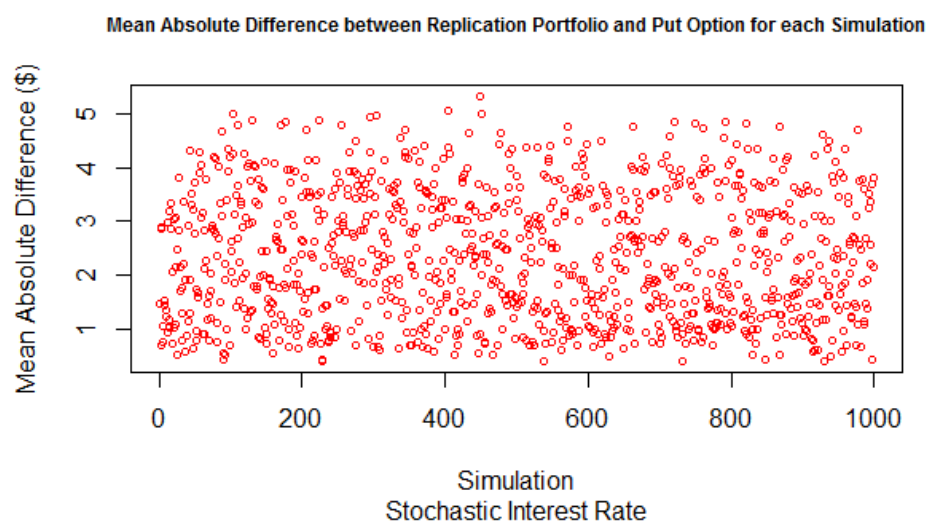
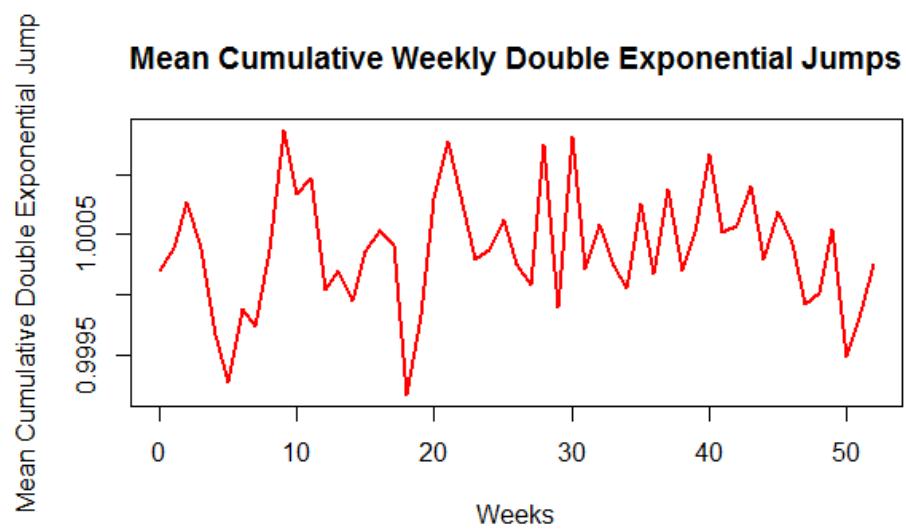
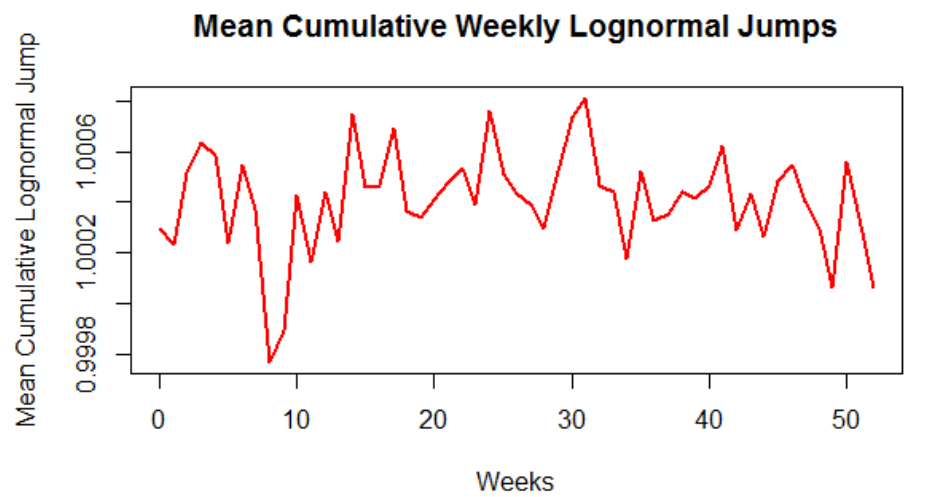


Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value

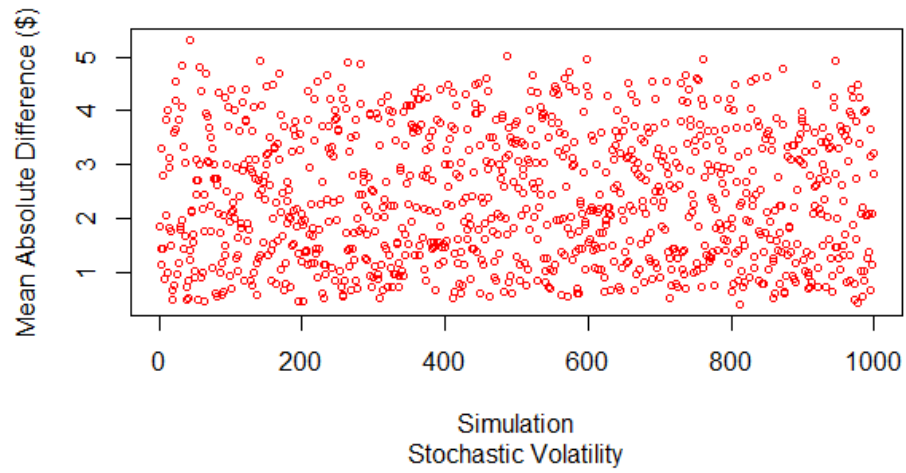


Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value

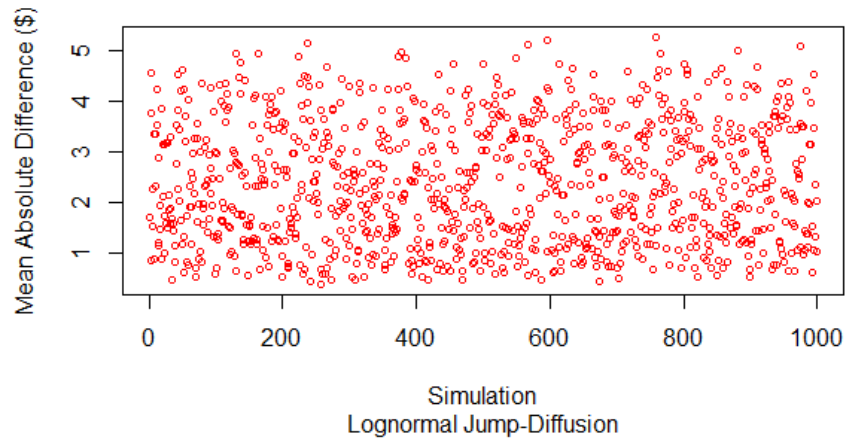




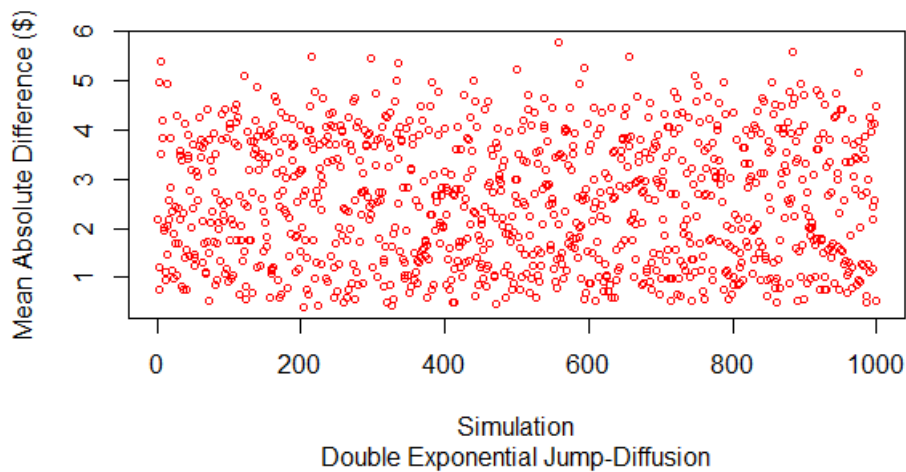
Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation



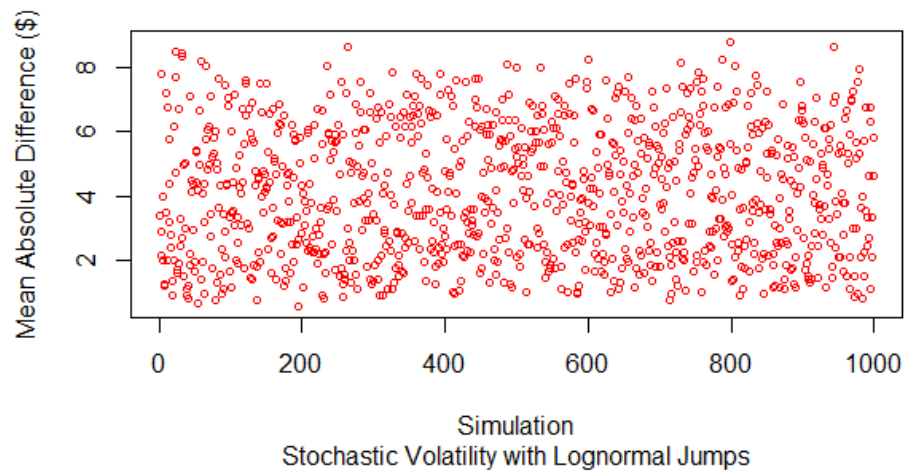
Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation



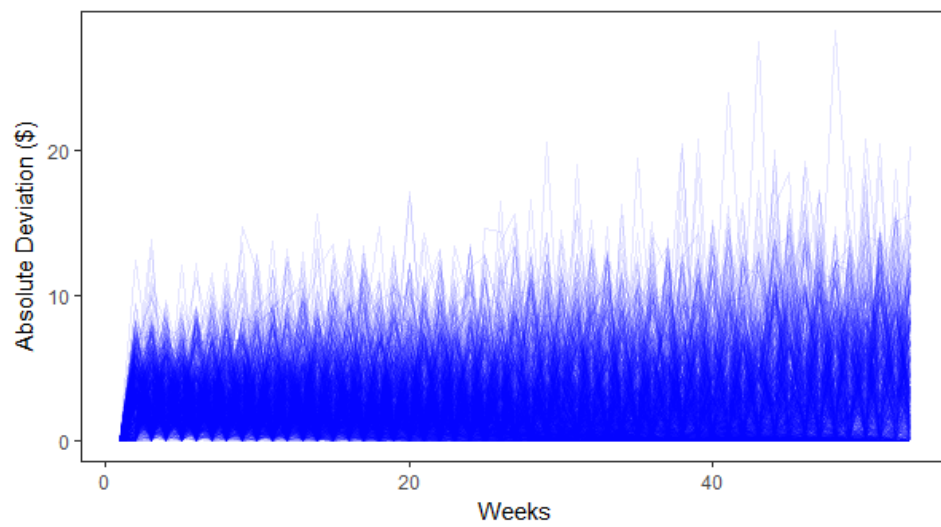
Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation



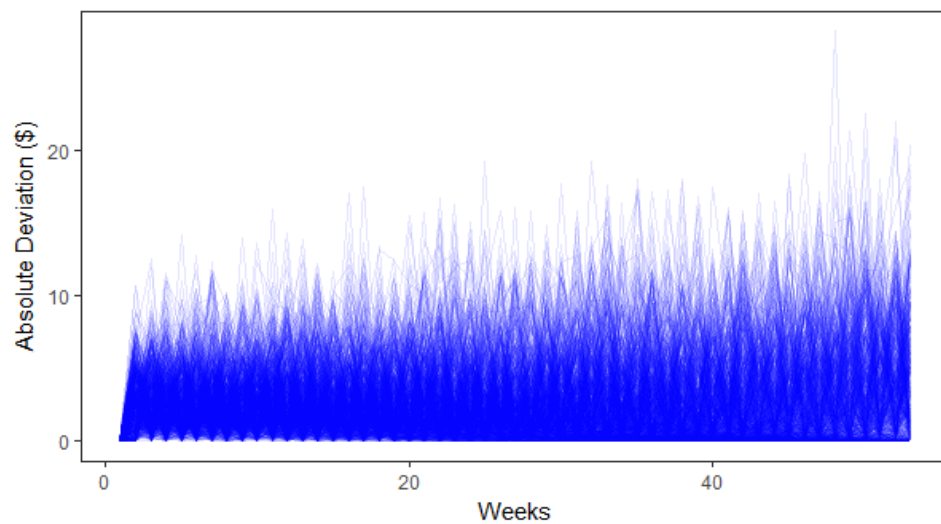
Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation



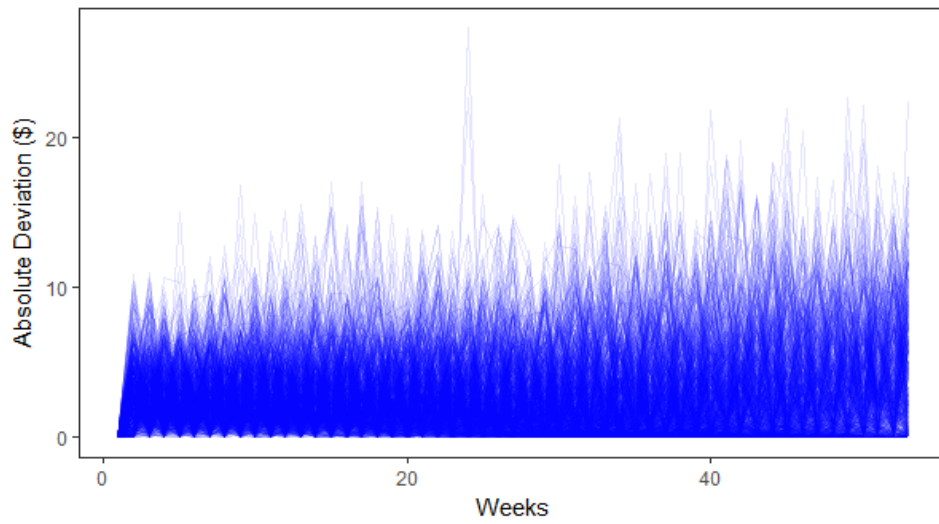
Absolute Differences (Stochastic Interest Rate)



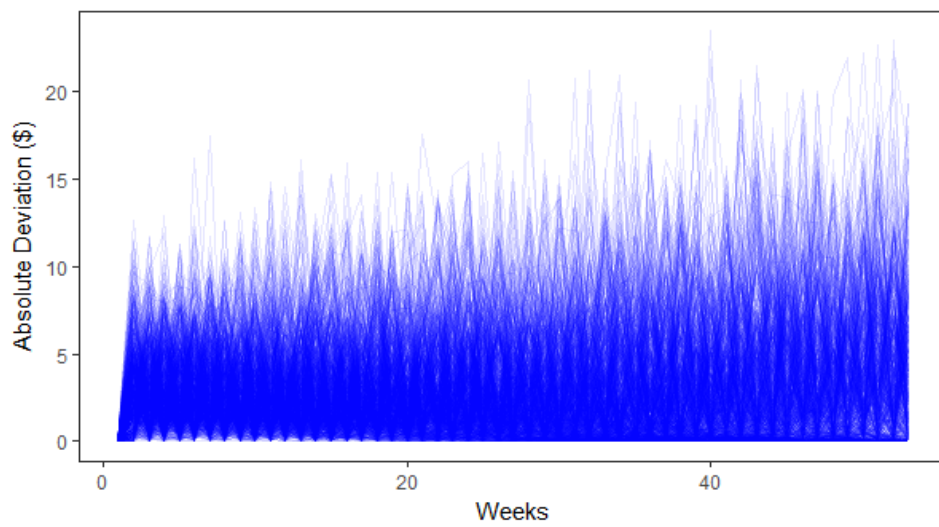
Absolute Differences (Stochastic Volatility)



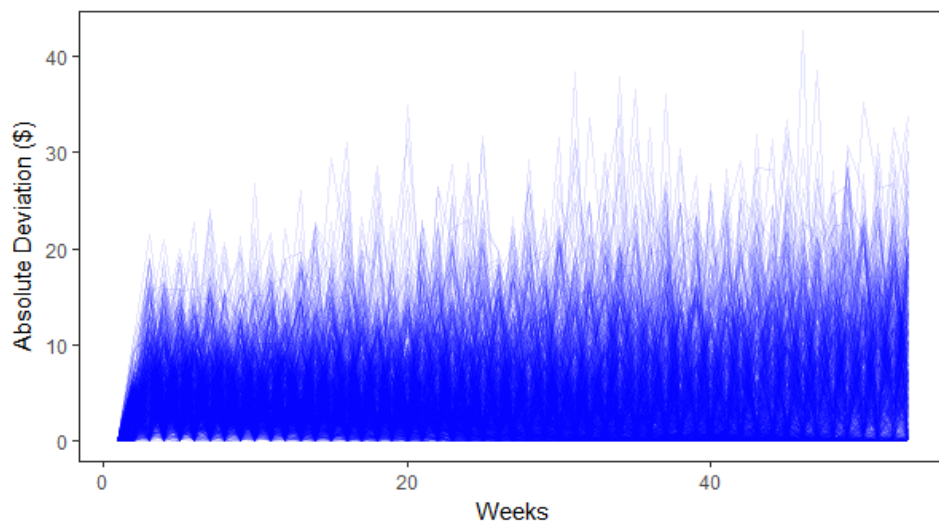
Absolute Differences (Lognormal Jump-Diffusion)



Absolute Differences (Double Exponential Jump-Diffusion)



Absolute Differences (Stochastic Volatility with Lognormal Jumps)



Appendix A

```
labexam x CIR-Ally.R x random x CIR-Ally (2).R x int_rate x CIR-Ally (3).R x CIR-Ally (1).R x sim_price x
Source on Save Run Source
1 # Simulating Stock Prices 2 ----
2 mu = 0.07
3 sigma = 0.25
4 t = 1/52
5 kappa = 0.5 # reversion to long term interest rate
6 theta = 0.02 # long term interest rate
7 eta = 0.1 # volatility of interest rate
8 rho = -0.3 # correlation
9 2*kappa*theta - eta^2 # has to greater than 0
10
11 sim_price = data.frame(matrix(nrow=1000,ncol=53))
12 sim_price[,] = 0
13 sim_price[,1] = 100 #S0
14
15 int_rate = data.frame(matrix(nrow=1000,ncol=53))
16 int_rate[,] = 0
17 int_rate[,1] = 0.02 #r0
18
19 random = data.frame(matrix(nrow=1000,ncol=53))
20 random[,] = 0
21 random[,] = rnorm(1000*53,0,1)
22
23 random2 = data.frame(matrix(nrow=1000,ncol=53))
24 random2[,] = 0
25 random2[,] = rnorm(1000*53,0,1)
26
27 random_combo = rho*random + sqrt(1-rho^2)*random2
28
29 for (i in 2:53){
30   for (j in 1:1000){
31     sim_price[j,i] = sim_price[j,i-1]*exp((mu-0.5*(sigma^2))*t+sigma*sqrt(t)*random[j,i])
32     int_rate[j,i] = int_rate[j,i-1] + kappa*(theta-int_rate[j,i-1])*t + eta*sqrt(int_rate[j,i-1])*random_combo[j,i]*sqrt(t)
33     j = j+1
34   }
35   i = i+1
36 }
37
38 write.csv(sim_price, "Excel file of Simulated Prices.csv")
39 write.csv(int_rate, "Excel file of Correlated Interest Rates.csv")
6:40 Simulating Stock Prices 2
```

```
> # Simulating Stock Prices 2 ----
> mu = 0.07
> sigma = 0.25
> t = 1/52
> kappa = 0.5 # reversion to long term interest rate
> theta = 0.02 # long term interest rate
> eta = 0.1 # volatility of interest rate
> rho = -0.3 # correlation
> 2*kappa*theta - eta^2 # has to greater than 0
[1] 0.01
>
> sim_price = data.frame(matrix(nrow=1000,ncol=53))
> sim_price[,] = 0
> sim_price[,1] = 100 #S0
>
> int_rate = data.frame(matrix(nrow=1000,ncol=53))
> int_rate[,] = 0
> int_rate[,1] = 0.02 #r0
>
> random = data.frame(matrix(nrow=1000,ncol=53))
> random[,] = 0
> random[,] = rnorm(1000*53,0,1)
>
> random2 = data.frame(matrix(nrow=1000,ncol=53))
> random2[,] = 0
> random2[,] = rnorm(1000*53,0,1)
>
> random_combo = rho*random + sqrt(1-rho^2)*random2
>
>
> for (i in 2:53){
+   for (j in 1:1000){
+     sim_price[j,i] = sim_price[j,i-1]*exp((mu-0.5*(sigma^2))*t+sigma*sqrt(t)*random[j,i])
+     int_rate[j,i] = int_rate[j,i-1] + kappa*(theta-int_rate[j,i-1])*t + eta*sqrt(int_rate[j,i-1])*random_combo[j,i]*sqrt(t)
+     j = j+1
+   }
+   i = i+1
+ }
>
> write.csv(sim_price, "Excel file of Simulated Prices.csv")
> write.csv(int_rate, "Excel file of Correlated Interest Rates.csv")
> view(sim_price)
> view(int_rate)
```

```

1 # FE Project
2 # Section 3
3 # Binomial Pricing of Options -----
4 library('ggplot2')
5 library('reshape2')
6 library('NMOF')
7 bin_price = data.frame(matrix(nrow=53,ncol=53))
8 bin_price[,] = 0
9 mu = 0.07
10 sigma = 0.25
11 v = mu - sigma^2/2
12 r = 0.02
13 t = 1/52
14 u = exp(sqrt((sigma^2)*t+(v*t)^2))
15 d = 1/u
16 p = 0.5*(1+1/sqrt((sigma^2)/((v^2)*t)+1))
17 R = exp(r*t)
18 q = (R-d)/(u-d)
19
20 bin_price[1,1]=100
21 for (k in 2:53){
22   bin_price[k,k] = 100*(d^(k-1))
23   k = k+1
24 }
25
26 for (i in 2:53){
27   for (j in 1:(i-1)){
28     bin_price[j,i] = bin_price[j,i-1]*u
29     j = j+1
30   }
31   i = i+1
32 }
33
34 ecall = data.frame(matrix(nrow=53,ncol=53))
35 ecall[,] = 0
36
37 for (k in 1:53){
38   ecall[k,53] = max(bin_price[k,53]- 100,0)
39   k = k+1
40 }
41
42 for (i in 1:52){
43   k = 53-i
44   for (j in 1:k){
45     c = (1/R)*(q*ecall[j,k+1]+(1-q)*ecall[j+1,k+1])
46     ecall[j,k] = max(c,0)
47     j = j+1
48   }
49   i = i+1
50 }
51
52 eput = data.frame(matrix(nrow=53,ncol=53))
53 eput[,] = 0
54
55 for (k in 1:53){
56   eput[k,53] = max(100-bin_price[k,53],0)
57   k = k+1
58 }
59
60 for (i in 1:52){
61   k = 53-i
62   for (j in 1:k){
63     c = (1/R)*(q*eput[j,k+1]+(1-q)*eput[j+1,k+1])
64     eput[j,k] = max(c,0)
65     j = j+1
66   }
67   i = i+1
68 }

```

```

69
70 aput = data.frame(matrix(nrow=53,ncol=53))
71 aput[,] = 0
72
73 for (k in 1:53){
74   aput[k,53] = max(100-bin_price[k,53],0)
75   k = k+1
76 }
77
78 for (i in 1:52){
79   k = 53-i
80   for (j in 1:k){
81     c = (1/R)*(q*aput[j,k+1]+(1-q)*aput[j+1,k+1])
82     d = 100-bin_price[j,k]
83     aput[j,k] = max(c,0,d)
84     j = j+1
85   }
86   i = i+1
87 }
88
89 # # More Timesteps in Binomial -----
90 # b_bin_price = data.frame(matrix(nrow=251,ncol=251))
91 # b_bin_price[,] = 0
92 # b_t = 1/250
93 # b_u = exp(sqrt((sigma^2)*b_t+(v*b_t)^2))
94 # b_d = 1/b_u
95 # b_R = exp(r*b_t)
96 # b_q = (b_R-b_d)/(b_u-b_d)
97 #
98 # b_bin_price[1,1]=100
99 # for (k in 2:251){
100 #   b_bin_price[k,k] = 100*(b_d^(k-1))
101 #   k = k+1
102 # }
103
104 #
105 # for (i in 2:251){
106 #   for (j in 1:(i-1)){
107 #     b_bin_price[j,i] = b_bin_price[j,i-1]*b_u
108 #     j = j+1
109 #   }
110 #   i = i+1
111 # }
112 #
113 # b_ecall = data.frame(matrix(nrow=251,ncol=251))
114 # b_ecall[,] = 0
115 #
116 # for (k in 1:251){
117 #   b_ecall[k,251] = max(b_bin_price[k,251]- 100,0)
118 #   k = k+1
119 # }
120 #
121 # for (i in 1:250){
122 #   k = 251-i
123 #   for (j in 1:k){
124 #     c = (1/b_R)*(b_q*b_ecall[j,k+1]+(1-b_q)*b_ecall[j+1,k+1])
125 #     b_ecall[j,k] = max(c,0)
126 #     j = j+1
127 #   }
128 #   i = i+1
129 # }
130 #
131 # b_eput = data.frame(matrix(nrow=251,ncol=251))
132 # b_eput[,] = 0
133 #
134 # for (k in 1:251){
135 #   b_eput[k,251] = max(100-b_bin_price[k,251],0)
136 #   k = k+1
137 # }

```



```

137 #
138 # for (i in 1:250){
139 #   k = 251-i
140 #   for (j in 1:k){
141 #     c = (1/b_R)*(b_q*b_eput[j,k+1]+(1-b_q)*b_eput[j+1,k+1])
142 #     b_eput[j,k] = max(c,0)
143 #     j = j+1
144 #   }
145 #   i = i+1
146 # }
147 # b_ecall[1,1]
148 # b_eput[1,1]
149 #
150 # c_bin_price = data.frame(matrix(nrow=1501,ncol=1501))
151 # c_bin_price[,] = 0
152 # c_t = 1/1500
153 # c_u = exp(sqrt((sigma^2)*c_t+(v*c_t)^2))
154 # c_d = 1/c_u
155 # c_R = exp(r*c_t)
156 # c_q = (c_R-c_d)/(c_u-c_d)
157 #
158 # c_bin_price[1,1]=100
159 # for (k in 2:1501){
160 #   c_bin_price[k,k] = 100*(c_d^(k-1))
161 #   k = k+1
162 # }
163 #
164 # for (i in 2:1501){
165 #   for (j in 1:(i-1)){
166 #     c_bin_price[j,i] = c_bin_price[j,i-1]*c_u
167 #     j = j+1
168 #   }
169 #   i = i+1
170 # }
171 "}"
172 # c_ecall = data.frame(matrix(nrow=1501,ncol=1501))
173 # c_ecall[,] = 0
174 #
175 # for (k in 1:1501){
176 #   c_ecall[k,1501] = max(c_bin_price[k,1501]- 100,0)
177 #   k = k+1
178 # }
179 #
180 # for (i in 1:1500){
181 #   k = 1501-i
182 #   for (j in 1:k){
183 #     c = (1/c_R)*(c_q*c_ecall[j,k+1]+(1-c_q)*c_ecall[j+1,k+1])
184 #     c_ecall[j,k] = max(c,0)
185 #     j = j+1
186 #   }
187 #   i = i+1
188 # }
189 #
190 # c_eput = data.frame(matrix(nrow=1501,ncol=1501))
191 # c_eput[,] = 0
192 #
193 # for (k in 1:1501){
194 #   c_eput[k,1501] = max(100-c_bin_price[k,1501],0)
195 #   k = k+1
196 # }
197 #
198 # for (i in 1:1500){
199 #   k = 1501-i
200 #   for (j in 1:k){
201 #     c = (1/c_R)*(c_q*c_eput[j,k+1]+(1-c_q)*c_eput[j+1,k+1])
202 #     c_eput[j,k] = max(c,0)
203 #     j = j+1
204 #   }
205 #   i = i+1

```

```

206 # }
207 # c_ecall[1,1]
208 # c_eput[1,1]
209
210
211 # Simulating Stock Prices ----
212 sim_price = data.frame(matrix(nrow=1000,ncol=53))
213 sim_price[,] = 0
214 sim_price[,1] = 100
215
216 random = data.frame(matrix(nrow=1000,ncol=53))
217 random[,] = 0
218 random[,] = rnorm(1000*53,0,1)
219
220 for (i in 2:53){
221   for (j in 1:1000){
222     sim_price[j,i] = sim_price[j,i-1]*exp((mu-0.5*(sigma^2))*t+sigma*sqrt(t)*random[j,i])
223     j = j+1
224   }
225   i = i+1
226 }
227
228 mean_sim_price = data.frame(matrix(nrow=1,ncol=53))
229 mean_sim_price[,] = 0
230 median_sim_price = data.frame(matrix(nrow=1,ncol=53))
231 median_sim_price[,] = 0
232 sd_sim_price = data.frame(matrix(nrow=1,ncol=53))
233 sd_sim_price[,] = 0
234
235 for (i in 1:53){
236   mean_sim_price[1,i] = mean(sim_price[,i])
237   i = i+1
238 }
239
240 g = 1:53
241 ymin = min(sd_sim_price[1,]/sqrt(1000))
242 ymax = max(sd_sim_price[1,]/sqrt(1000))
243 plot(x=(g-1),y=sd_sim_price[1,g]/sqrt(1000),xlab="weeks", ylab="Standard Deviation of Stock Prices",
244      main="Volatility of Stock Prices",type='l',
245      lwd=2,lty=1,col="red",ylim=c(ymin,ymax))
246
247
248 g = 1:53
249 ymin = min(median_sim_price[1,],mean_sim_price[1,])
250 ymax = max(median_sim_price[1,],mean_sim_price[1,])
251 plot(x=(g-1),y=mean_sim_price[1,g],xlab="weeks", ylab="Stock Price ($)",
252      main="Mean & Median Simulated Stock Price Paths",type='l',
253      lwd=2,lty=1,col="red",ylim=c(ymin,ymax))
254 lines(x=(g-1),y=median_sim_price[1,g],lwd=2,col="blue")
255 labels <- c("Mean Stock Price","Median Stock Price")
256 legend("topleft", inset=.05,labels,lwd=2, lty=c(1,1), col=c("red","blue"), cex = 0.7)
257
258 m_sim_price = sim_price
259 rownames(m_sim_price) = paste("trial", seq(1000), sep="")
260 colnames(m_sim_price) = paste("time", seq(53), sep="")
261 dat = as.data.frame(m_sim_price)
262 dat$trial = rownames(dat)
263 mdat = melt(dat, id.vars="trial")
264 mdat$time = as.numeric(gsub("time", "", mdat$variable))
265 p1 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
266   ggtitle("1000 Monte Carlo GBM Simulated Stock Price Paths") +
267   labs(x="weeks",y="Stock Price") +
268   theme_bw() +
269   <

```

```

268 theme_bw() +
269 theme(panel.grid=element_blank()) +
270 geom_line(size=0.2, alpha=0.1, col="darkblue")|
271
272 # Monte Carlo Pricing of Options -----
273 ecall2 = data.frame(matrix(nrow=1000,ncol=53))
274 ecall2[,] = 0
275 eput2 = data.frame(matrix(nrow=1000,ncol=53))
276 eput2[,] = 0
277
278 for (k in 1:1000){
279   for (j in 1:53){
280     ecall2[k,j] = max(0,sim_price[k,j]-100*(1/R^(53-j)))*(1/R^(j-1))
281     eput2[k,j] = max(0,100*(1/R^(53-j))-sim_price[k,j])*(1/R^(j-1))
282     j = j+1
283   }
284   k = k+1
285 }
286
287 hist(ecall2[,53],xlab="Simulated Call Option Price At Time 0 ($)", ylab="Frequency",
288      main="Histogram of Call Option Prices for 1000 Simulated Stock Paths",cex.main=0.95)
289 hist(eput2[,53],xlab="Simulated Put Option Price At Time 0 ($)", ylab="Frequency",
290      main="Histogram of Put Option Prices for 1000 Simulated Stock Paths",cex.main=0.95)
291
292 # check using call to put parity
293 check = ecall[1,1] - eput[1,1] + 100*(1/(R^52))
294 checkm = data.frame(matrix(nrow=1000,ncol=1))
295 checkm[,] = 0
296 checkm[,1] = ecall2[,53] - eput2[,53] + 100*(1/(R^52))
297 sum(checkm[,1])
298
299 BS.eput <- function(St,Tminust,K,int_rate){
300   <

```

```

298
299 BS.eput <- function(St,Tminust,K,int_rate){
300   d1 = (log(St/K)+(int_rate+0.5*(sigma^2))*(Tminust/52))/(sigma*sqrt(Tminust/52))
301   d2 = d1 - sigma*sqrt(Tminust/52)
302   bs_eput = K*(exp(int_rate*(-Tminust/52)))*pnorm(-d2,0,1) - St*pnorm(-d1,0,1)
303   bs_eput
304 }
305
306 BS.ecall <- function(St,Tminust,K,int_rate){
307   d1 = (log(St/K)+(int_rate+0.5*(sigma^2))*(Tminust/52))/(sigma*sqrt(Tminust/52))
308   d2 = d1 - sigma*sqrt(Tminust/52)
309   bs_ecall = St*pnorm(d1,0,1) - K*(exp(int_rate*(-Tminust/52)))*pnorm(d2,0,1)
310   bs_ecall
311 }
312
313 d1.find <- function(St,Tminust,K,int_rate){
314   d1 = (log(St/K)+(int_rate+0.5*(sigma^2))*(Tminust/52))/(sigma*sqrt(Tminust/52))
315   d1
316 }
317
318 d2.find <- function(St,Tminust,K,int_rate){
319   d2 = (log(St/K)+(int_rate-0.5*(sigma^2))*(Tminust/52))/(sigma*sqrt(Tminust/52))
320   d2
321 }
322
323
324
325 d1.find(100,52,100,0.02)
326 aput[1,1]
327 bs_ecall = BS.ecall(100,52,100,0.02)
328 bs_eput = BS.eput(100,52,100,0.02)
329 bs_ecall
330 bs_eput
331 ecall[1,1]

```

```

332 eput[1,1]
333 mean(ecall2[,53])
334 mean(eput2[,53])
335 sd(ecall2[,53])/sqrt(1000)
336 sd(eput2[,53])/sqrt(1000)
337 sd(ecall2[,53])
338 sd(eput2[,53])
339 sd(ecall2[,10])
340 sd(eput2[,10])
341 median(ecall2[,53])
342 median(eput2[,53])
343
344 # Section 4
345 # Binomial Hedging -----
346 x = data.frame(matrix(nrow=53,ncol=53))
347 x[,] = 0
348 b = data.frame(matrix(nrow=53,ncol=53))
349 b[,] = 0
350 bin_port_value = data.frame(matrix(nrow=53,ncol=53))
351 bin_port_value[,] = 0
352 bin_difference = data.frame(matrix(nrow=53,ncol=53))
353 bin_difference[,] = 0
354 bin_before_rep = data.frame(matrix(nrow=53,ncol=53))
355 bin_before_rep[,] = 0
356
357 for (i in 1:52){
358   k = 53-i
359   for (j in 1:k){
360     x[j,k] = (1/(u-d))*(eput[j,k+1]-eput[j+1,k+1])
361     b[j,k] = (1/(R*(u-d)))*(u*eput[j+1,k+1]-d*eput[j,k+1])
362     bin_port_value[j,k] = x[j,k]+b[j,k]
363     j = j+1
364   }
365   i = i+1
366 }
367 <

```

```

368 for (j in 1:52){
369   bin_port_value[j,53] = x[j,52]*u+b[j,52]*R
370   j = j+1
371 }
372 bin_port_value[53,53] = x[52,52]*d+b[52,52]*R
373
374 for(k in 2:53){
375   bin_before_rep[k,k] = x[k-1,k-1]*d+b[k-1,k-1]*R
376   k = k+1
377 }
378
379 for (i in 1:52){
380   for (j in 1:i){
381     bin_before_rep[j,i+1] = x[j,i]*u+b[j,i]*R
382     j = j+1
383   }
384   i = i+1
385 }
386
387 bin_before_rep[1,1]=eput[1,1]
388 bin_difference = bin_before_rep-eput
389 sum(abs(bin_difference))
390 sum(abs(bin_port_value - eput))
391
392 # write.csv(bin_port_value, "test2.csv")
393 write.csv(bin_difference, "test3.csv")
394
395 # Black-Scholes & Monte Carlo Hedging -----
396 bs_eput = data.frame(matrix(nrow=1000,ncol=53))
397 bs_eput[,] = 0
398 x2 = data.frame(matrix(nrow=1000,ncol=53))
399 x2[,] = 0
400 b2 = data.frame(matrix(nrow=1000,ncol=53))
401 b2[,] = 0

```

```

400 b2 = data.frame(matrix(nrow=1000,ncol=53))
401 b2[,] = 0
402 sim_port_value = data.frame(matrix(nrow=1000,ncol=53))
403 sim_port_value[,] = 0
404 sim_value_before_rep = data.frame(matrix(nrow=1000,ncol=53))
405 sim_value_before_rep[,] = 0
406 sim_diff = data.frame(matrix(nrow=1000,ncol=53))
407 sim_diff[,] = 0
408
409 for (i in 1:52){
410   for (j in 1:1000){
411     bs_eput[j,i] = BS.eput(sim_price[j,i],53-i,100,0.02)
412     x2[j,i] = 100*(1/R^(53-i))*pnorm(-d2.find(sim_price[j,i],53-i,100,0.02),0,1)
413     b2[j,i] = -sim_price[j,i]*pnorm(-d1.find(sim_price[j,i],53-i,100,0.02),0,1)
414     sim_port_value[j,i] = x2[j,i] + b2[j,i]
415     j = j+1
416   }
417   i = i+1
418 }
419
420 sim_value_before_rep[,1] = bs_eput[,1]
421
422 for (i in 2:53){
423   for (j in 1:1000){
424     sim_value_before_rep[j,i] = x2[j,i-1]*(sim_price[j,i]/sim_price[j,i-1]) + b2[j,i-1]*R
425     j = j+1
426   }
427   i = i+1
428 }
429

```

Source on Save

```

430 for (k in 1:1000){
431   bs_eput[k,53] = max(0,(100-sim_price[k,53]))
432   sim_port_value[k,53] = max(0,(100-sim_price[k,53]))
433   k = k+1
434 }
435
436 sim_diff = abs(sim_value_before_rep - bs_eput)
437 sum(sim_diff)/(52*1000)
438
439 mean_rep_value = data.frame(matrix(nrow=1,ncol=53))
440 mean_rep_value[,] = 0
441 for (i in 1:1000){
442   mean_rep_value[1,i] = mean(sim_value_before_rep[,i])
443   i = i+1
444 }
445
446 mean_diff = data.frame(matrix(nrow=1000,ncol=1))
447 mean_diff[,] = 0
448 sd_diff = data.frame(matrix(nrow=1,ncol=1000))
449 sd_diff[,] = 0
450
451 for (i in 1:1000){
452   mean_diff[i,1] = sum(sim_diff[i,])/52
453   i = i+1
454 }
455
456 mean_put_price = data.frame(matrix(nrow=1,ncol=53))
457 mean_put_price[,] = 0
458 median_put_price = data.frame(matrix(nrow=1,ncol=53))
459 median_put_price[,] = 0
460 sd_put_price = data.frame(matrix(nrow=1,ncol=53))
461 sd_put_price[,] = 0
462

```

```

463 for (i in 1:53){
464   mean_put_price[1,i] = mean(bs_eput[,i])
465   median_put_price[1,i] = median(bs_eput[,i])
466   sd_put_price[1,i] = sd(bs_eput[,i])
467   i = i+1
468 }
469
470 g = 1:1000
471 f = xy.coords(x=g,y=mean_diff[g,1])
472 plot(f,xlab="Simulation", ylab="Mean Absolute Difference ($)",cex.main = 0.75,
473      main="Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation",pch=1,cex=0.7,lwd=1.8,col='red')
474
475 m_sim_diff = sim_diff
476 rownames(m_sim_diff) = paste("trial", seq(1000), sep="")
477 colnames(m_sim_diff) = paste("time", seq(53), sep="")
478 dat = as.data.frame(m_sim_diff)
479 dat$trial = rownames(dat)
480 mdat = melt(dat, id.vars="trial")
481 mdat$time = as.numeric(gsub("time", "", mdat$variable))
482 p3 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
483   ggtitle("Absolute Differences between Black Scholes Replication Portfolio and Put Value for All Runs") +
484   labs(x="weeks",y="Absolute Deviation ($)") +
485   theme_bw() +
486   theme(plot.title = element_text(size=10)) +
487   theme(panel.grid=element_blank()) +
488   geom_line(size=0.1, alpha=0.1, col="darkgreen")
489
490 f = xy.coords(x=mean_rep_value[1,],y=mean_put_price[1,])
491 plot(f,xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price",cex.main = 0.85,
492      main="Mean Portfolio Value Before Replication vs After Replication",pch=1,cex=0.7,lwd=1.8,col='red')
493 abline(0,1,col="blue",lwd=1,lty=2)
494
495 test <- unlist(sim_diff)
496 hist(test,xlab="Absolute Difference ($)", ylab="Frequency",
497      main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option value",
498      cex.main=0.75,breaks=50,xlim=c(0,15))
499
500
501
502
503 # # Graphs regarding simulated put price -----
504 # g = 1:53
505 # ymin = min(mean_put_price[1,])
506 # ymax = max(mean_put_price[1,])
507 # plot(x=(g-1),y=mean_put_price[1,g] ,xlab="weeks", ylab="Mean Put Option Price ($)",
508 #      main="Mean Put Option Price (valued using Black-Scholes DE) for 1000 simulated Stock Paths",type='l',
509 #      lwd=2,lty=1,col="red",ylim=c(ymin,ymax),cex.main=0.8)
510 #
511 #
512 #
513 # g = 1:53
514 # plot(x=(g-1),y=sd_put_price[1,g] ,xlab="weeks", ylab="Put option standard deviation",
515 #      main="Put option volatility for 1000 simulated Stock Paths",type='l',
516 #      lwd=2,lty=1,col="red")
517 #
518 # f = xy.coords(x=mean_sim_price[1,],y=mean_put_price[1,])
519 # plot(f,xlab="Mean Stock Price", ylab="Mean Put Option Price",cex.main = 0.85,
520 #      main="Mean Put Option Price (valued using Black-Scholes DE) vs Mean Stock Price",pch=1,cex=0.7,lwd=1.8,col='red')
521 #
522 # f = xy.coords(x=median_sim_price[1,],y=median_put_price[1,])
523 # plot(f,xlab="Median Stock Price", ylab="Median Put Option Price",
524 #      main="Median Put Option Price vs Median Stock Price",pch=1,cex=0.7,lwd=1.8,col='red')
525 #

```



```

526 # m_bs_eput = bs_eput
527 # rownames(m_bs_eput) = paste("trial", seq(1000), sep="")
528 # colnames(m_bs_eput) = paste("time", seq(53), sep="")
529 # dat = as.data.frame(m_bs_eput)
530 # dat$trial = rownames(dat)
531 # mdat = melt(dat, id.vars="trial")
532 # mdat$time = as.numeric(gsub("time", "", mdat$variable))
533 # p2 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
534 #   ggtitle("Put Option Prices (valued using Black-Scholes DE) for 1000 simulated Stock Paths") +
535 #   labs(x="weeks", y="Put Option Price") +
536 #   theme_bw() +
537 #   theme(panel.grid=element_blank()) +
538 #   geom_line(size=0.2, alpha=0.1, col="purple")
539 #
540 # hist(bs_eput[,53], xlab="Simulated Put Option Price At Time of Maturity", ylab="Frequency",
541 #      main="Histogram of Put Option Prices (valued using Black-Scholes DE) for 1000 Simulated Stock Paths",
542 #      cex.main=0.75)
543
544 # Simulating Stock Prices 2 ----
545 kappa = 1
546 theta = 0.25^2
547 eta = 0.01 #0.05
548 rho = -0.3
549 2*kappa*theta - eta^2 # has to be greater than 0
550
551 # c = ((eta^2)*(1-exp(-kappa)))/(4*kappa)
552 # d = (4*kappa*theta)/(eta^2)
553 # lamb = ((4*kappa*exp(-kappa))*100)/((eta^2)*(1-exp(-kappa)))
554 # H_sim_price[,53]=c*rchisq(n=1000,df=d,ncp=lamb)
555
556 a = 0.1
557 b = 0.02
558 eta2 = 0.8
559 rho2 = -0.3
560
561
562 lambda = 1
563 logmu = 0.0003749406333
564 logsigma = 0.007068152235
565
566 H_sim_price = data.frame(matrix(nrow=1000, ncol=53))
567 H_sim_price[,] = 0
568 H_sim_price[,1] = 100
569
570
571 H_sim_price2 = data.frame(matrix(nrow=1000, ncol=53))
572 H_sim_price2[,] = 0
573 H_sim_price2[,1] = 100
574
575 H_sim_price3 = data.frame(matrix(nrow=1000, ncol=53))
576 H_sim_price3[,] = 0
577 H_sim_price3[,1] = 100
578
579 H_sim_price4 = data.frame(matrix(nrow=1000, ncol=53))
580 H_sim_price4[,] = 0
581 H_sim_price4[,1] = 100
582
583 H_vol = data.frame(matrix(nrow=1000, ncol=53))
584 H_vol[,] = 0
585 H_vol[,1] = 0.0625
586
587 H_vol2 = data.frame(matrix(nrow=1000, ncol=53))
588 H_vol2[,] = 0
589 H_vol2[,1] = 0.0625
590
591 H_random = data.frame(matrix(nrow=1000, ncol=53))
592 H_random[,] = 0
593 H_random[,] = rnorm(1000*53, 0, 1)
594

```

```

594
595 H_random2 = data.frame(matrix(nrow=1000,ncol=53))
596 H_random2[,] = 0
597 H_random2[,] = rnorm(1000*53,0,1)
598
599 H_random_combo = data.frame(matrix(nrow=1000,ncol=53))
600 H_random_combo[,] = 0
601
602 H_random_combo = rho*H_random + sqrt(1-rho^2)*H_random2
603
604 H_random3 = data.frame(matrix(nrow=1000,ncol=53))
605 H_random3[,] = 0
606 H_random3[,] = rnorm(1000*53,0,1)
607
608 H_random4 = data.frame(matrix(nrow=1000,ncol=53))
609 H_random4[,] = 0
610 H_random4[,] = rnorm(1000*53,0,1)
611
612 H_random5 = data.frame(matrix(nrow=1000,ncol=53))
613 H_random5[,] = 0
614 H_random5[,] = rnorm(1000*53,0,1)
615
616 H_random6 = data.frame(matrix(nrow=1000,ncol=53))
617 H_random6[,] = 0
618 H_random6[,] = rnorm(1000*53,0,1)
619
620 H_random7 = data.frame(matrix(nrow=1000,ncol=53))
621 H_random7[,] = 0
622 H_random7[,] = rnorm(1000*53,0,1)
623
624 H_random_combo2 = data.frame(matrix(nrow=1000,ncol=53))
625 H_random_combo2[,] = 0
626 H_random_combo3 = data.frame(matrix(nrow=1000,ncol=53))
627 H_random_combo3[,] = 0
628
...
628
629 H_random_combo2 = rho2*H_random + sqrt(1-rho2^2)*H_random4
630
631 H_random_combo3 = rho*H_random6 + sqrt(1-rho^2)*H_random7
632
633 H_poisson = data.frame(matrix(nrow=1000,ncol=53))
634 H_poisson[,] = 0
635 H_poisson[,] = rpois(1000*53,lambda)
636
637 H_cum_lognormal = data.frame(matrix(nrow=1000,ncol=53))
638 H_cum_lognormal[,] = 0
639
640 H_cum_doubleexp = data.frame(matrix(nrow=1000,ncol=53))
641 H_cum_doubleexp[,] = 0
642
643 double.exp <- function(p,n1,n2,numb){
644   if (numb==0) return(1)
645   answer = 1
646   for (l in 1:numb){
647     prob = runif(1,0,1)
648     if (prob<p) { temp = rexp(1,n1)}
649     else {temp = -rexp(1,n2)}
650     answer = answer*exp(temp)
651     l = l+1
652   }
653   answer
654 }
655
656
657 for (i in 1:53){
658   for (j in 1:1000){
659     H_cum_lognormal[j,i] = prod(rlnorm(H_poisson[j,i],logmu,logsigma))
660     H_cum_doubleexp[j,i] = double.exp(0.52,100,100,H_poisson[j,i])
661     j = j+1
662   }
663 }

```



```

662 }
663 i = i+1
664 }
665
666 H_int_rate = data.frame(matrix(nrow=1000,ncol=53))
667 H_int_rate[,] = 0
668 H_int_rate[,1] = 0.02
669
670
671
672
673 for (i in 2:53){
674   for (j in 1:1000){
675     #H_vol[j,i] = H_vol[j,i-1]*exp((1/H_vol[j,i-1])*(kappa*(theta-H_vol[j,i-1])-0.5*eta^2)*t+(eta/sqrt(H_vol[j,i-1]))*H_random_combo[j,i]*sqrt(t))
676     H_vol[j,i] = H_vol[j,i-1] + kappa*(theta-H_vol[j,i-1])*t + eta*sqrt(H_vol[j,i-1])*H_random_combo[j,i]*sqrt(t)
677     H_sim_price[j,i] = H_sim_price[j,i-1]*exp((mu-0.5*H_vol[j,i-1])*t+sqrt(H_vol[j,i-1])*sqrt(t)*H_random[j,i])
678     #H_vol2[j,i] = H_vol2[j,i-1]*exp((1/H_vol2[j,i-1])*(kappa*(theta-H_vol2[j,i-1])-0.5*eta^2)*t+(eta/sqrt(H_vol2[j,i-1]))*H_random_combo3[j,i]*sqrt(t))
679     H_vol2[j,i] = H_vol2[j,i-1] + kappa*(theta-H_vol2[j,i-1])*t + eta*sqrt(H_vol2[j,i-1])*H_random_combo3[j,i]*sqrt(t)
680     H_sim_price2[j,i] = H_sim_price[j,i-1]*H_cum_lognormal[j,i]*exp((mu-0.5*H_vol2[j,i-1])*t+sqrt(H_vol2[j,i-1])*sqrt(t)*H_random6[j,i])
681     H_int_rate[j,i] = H_int_rate[j,i-1] + a*(b-H_int_rate[j,i-1])*t + eta2*H_int_rate[j,i-1]*H_random_combo2[j,i]*sqrt(t)
682     H_sim_price3[j,i] = H_sim_price3[j,i-1]*H_cum_lognormal[j,i]*exp((mu-0.5*(sigma^2))*t+sigma*sqrt(t)*H_random3[j,i])
683     H_sim_price4[j,i] = H_sim_price4[j,i-1]*H_cum_doubleexp[j,i]*exp((mu-0.5*(sigma^2))*t+sigma*sqrt(t)*H_random5[j,i])
684     j = j+1
685   }
686   i = i+1
687 }
688
689
690 # Black-Scholes & Monte Carlo Hedging 2 -----
691 H_bs_eput = data.frame(matrix(nrow=1000,ncol=53))
692 H_bs_eput[,] = 0
693 H_x = data.frame(matrix(nrow=1000,ncol=53))
694 H_x[,] = 0
695 H_b = data.frame(matrix(nrow=1000,ncol=53))
696 H_b[,] = 0
697
698
699
700
701
702 H_bs_eput2 = data.frame(matrix(nrow=1000,ncol=53))
703 H_bs_eput2[,] = 0
704 H_x2 = data.frame(matrix(nrow=1000,ncol=53))
705 H_x2[,] = 0
706 H_b2 = data.frame(matrix(nrow=1000,ncol=53))
707 H_b2[,] = 0
708 H_sim_diff2 = data.frame(matrix(nrow=1000,ncol=53))
709 H_sim_diff2[,] = 0
710 H_sim_value_before_rep2 = data.frame(matrix(nrow=1000,ncol=53))
711 H_sim_value_before_rep2[,] = 0
712
713 H_bs_eput3 = data.frame(matrix(nrow=1000,ncol=53))
714 H_bs_eput3[,] = 0
715 H_x3 = data.frame(matrix(nrow=1000,ncol=53))
716 H_x3[,] = 0
717 H_b3 = data.frame(matrix(nrow=1000,ncol=53))
718 H_b3[,] = 0
719 H_sim_diff3 = data.frame(matrix(nrow=1000,ncol=53))
720 H_sim_diff3[,] = 0
721 H_sim_value_before_rep3 = data.frame(matrix(nrow=1000,ncol=53))
722 H_sim_value_before_rep3[,] = 0
723
724 H_bs_eput4 = data.frame(matrix(nrow=1000,ncol=53))
725 H_bs_eput4[,] = 0
726 H_x4 = data.frame(matrix(nrow=1000,ncol=53))
727 H_x4[,] = 0
728 H_b4 = data.frame(matrix(nrow=1000,ncol=53))
729 H_b4[,] = 0
730 H_sim_diff4 = data.frame(matrix(nrow=1000,ncol=53))
731 H_sim_diff4[,] = 0
732 H_sim_value_before_rep4 = data.frame(matrix(nrow=1000,ncol=53))

```

```


732 H_sim_value_before_rep4 = data.frame(matrix(nrow=1000,ncol=53))
733 H_sim_value_before_rep4[,] = 0
734
735 H_bs_eput5 = data.frame(matrix(nrow=1000,ncol=53))
736 H_bs_eput5[,] = 0
737 H_x5 = data.frame(matrix(nrow=1000,ncol=53))
738 H_x5[,] = 0
739 H_b5 = data.frame(matrix(nrow=1000,ncol=53))
740 H_b5[,] = 0
741 H_sim_diff5 = data.frame(matrix(nrow=1000,ncol=53))
742 H_sim_diff5[,] = 0
743 H_sim_value_before_rep5 = data.frame(matrix(nrow=1000,ncol=53))
744 H_sim_value_before_rep5[,] = 0
745
746 for (i in 1:52){
747   for (j in 1:1000){
748     H_bs_eput[j,i] = BS.eput(H_sim_price[j,i],53-i,100,0.02)
749     H_x[j,i] = 100*(1/R^(53-i))*pnorm(-d2.find(H_sim_price[j,i],53-i,100,0.02),0,1)
750     H_b[j,i] = -H_sim_price[j,i]*pnorm(-d1.find(H_sim_price[j,i],53-i,100,0.02),0,1)
751
752     H_bs_eput2[j,i] = BS.eput(H_sim_price2[j,i],53-i,100,0.02)
753     H_x2[j,i] = 100*(1/R^(53-i))*pnorm(-d2.find(H_sim_price2[j,i],53-i,100,0.02),0,1)
754     H_b2[j,i] = -H_sim_price2[j,i]*pnorm(-d1.find(H_sim_price2[j,i],53-i,100,0.02),0,1)
755
756     H_bs_eput3[j,i] = BS.eput(H_sim_price3[j,i],53-i,100,0.02)
757     H_x3[j,i] = 100*(1/R^(53-i))*pnorm(-d2.find(H_sim_price3[j,i],53-i,100,0.02),0,1)
758     H_b3[j,i] = -H_sim_price3[j,i]*pnorm(-d1.find(H_sim_price3[j,i],53-i,100,0.02),0,1)
759
760     # Interest Rates
761     H_bs_eput4[j,i] = BS.eput(sim_price[j,i],53-i,100,H_int_rate[j,i])
762     H_x4[j,i] = 100*(1/exp(H_int_rate[j,i]*(53-i)/52))*pnorm(-d2.find(sim_price[j,i],53-i,100,H_int_rate[j,i]),0,1)
763     H_b4[j,i] = -sim_price[j,i]*pnorm(-d1.find(sim_price[j,i],53-i,100,H_int_rate[j,i]),0,1)
764
765     H_bs_eput5[j,i] = BS.eput(H_sim_price4[j,i],53-i,100,0.02)
766     H_x5[j,i] = 100*(1/R^(53-i))*pnorm(-d2.find(H_sim_price4[j,i],53-i,100,0.02),0,1)
767     H_b5[j,i] = -H_sim_price4[j,i]*pnorm(-d1.find(H_sim_price4[j,i],53-i,100,0.02),0,1)
768   }
769   j = j+1
770 }
771 i = i+1
772 }
773
774 for (k in 1:1000){
775   H_bs_eput[k,53] = max(0,100-H_sim_price[k,53])
776   H_bs_eput2[k,53] = max(0,100-H_sim_price2[k,53])
777   H_bs_eput3[k,53] = max(0,100-H_sim_price3[k,53])
778   H_bs_eput4[k,53] = max(0,100-sim_price[k,53])
779   H_bs_eput5[k,53] = max(0,100-H_sim_price4[k,53])
780   k = k+1
781 }
782
783 H_sim_value_before_rep[,1] = H_bs_eput[,1]
784 H_sim_value_before_rep2[,1] = H_bs_eput2[,1]
785 H_sim_value_before_rep3[,1] = H_bs_eput3[,1]
786 H_sim_value_before_rep4[,1] = H_bs_eput4[,1]
787 H_sim_value_before_rep5[,1] = H_bs_eput5[,1]
788
789 for (i in 2:53){
790   for (j in 1:1000){
791     H_sim_value_before_rep[j,i] = H_x[j,i-1]*(H_sim_price[j,i]/H_sim_price[j,i-1]) + H_b[j,i-1]*R
792     H_sim_value_before_rep2[j,i] = H_x2[j,i-1]*(H_sim_price2[j,i]/H_sim_price2[j,i-1]) + H_b2[j,i-1]*R
793     H_sim_value_before_rep3[j,i] = H_x3[j,i-1]*(H_sim_price3[j,i]/H_sim_price3[j,i-1]) + H_b3[j,i-1]*R
794     H_sim_value_before_rep4[j,i] = H_x4[j,i-1]*(sim_price[j,i]/sim_price[j,i-1]) + H_b4[j,i-1]*exp(H_int_rate[j,i-1]*t)
795     H_sim_value_before_rep5[j,i] = H_x5[j,i-1]*(H_sim_price4[j,i]/H_sim_price4[j,i-1]) + H_b5[j,i-1]*R
796     j = j+1
797   }
798   i = i+1
799 }
800
801 H_sim_diff = abs(H_sim_value_before_rep - H_bs_eput)
802 H_sim_diff2 = abs(H_sim_value_before_rep2 - H_bs_eput2)

```

```

801
802 H_sim_diff = abs(H_sim_value_before_rep - H_bs_eput)
803 H_sim_diff2 = abs(H_sim_value_before_rep2 - H_bs_eput2)
804 H_sim_diff3 = abs(H_sim_value_before_rep3 - H_bs_eput3)
805 H_sim_diff4 = abs(H_sim_value_before_rep4 - H_bs_eput4)
806 H_sim_diff5 = abs(H_sim_value_before_rep5 - H_bs_eput5)
807 sum(H_sim_diff)/(52*1000)
808 sum(H_sim_diff2)/(52*1000)
809 sum(H_sim_diff3)/(52*1000)
810 sum(H_sim_diff4)/(52*1000)
811 sum(H_sim_diff5)/(52*1000)
812
813
814
815 H_mean_rep_value = data.frame(matrix(nrow=1,ncol=53))
816 H_mean_rep_value[,] = 0
817 H_mean_rep_value2 = data.frame(matrix(nrow=1,ncol=53))
818 H_mean_rep_value2[,] = 0
819 H_mean_rep_value3 = data.frame(matrix(nrow=1,ncol=53))
820 H_mean_rep_value3[,] = 0
821 H_mean_rep_value4 = data.frame(matrix(nrow=1,ncol=53))
822 H_mean_rep_value4[,] = 0
823 H_mean_rep_value5 = data.frame(matrix(nrow=1,ncol=53))
824 H_mean_rep_value5[,] = 0
825
826 for (i in 1:1000){
827   H_mean_rep_value[i,1] = mean(H_sim_value_before_rep[,i])
828   H_mean_rep_value2[i,1] = mean(H_sim_value_before_rep2[,i])
829   H_mean_rep_value3[i,1] = mean(H_sim_value_before_rep3[,i])
830   H_mean_rep_value4[i,1] = mean(H_sim_value_before_rep4[,i])
831   H_mean_rep_value5[i,1] = mean(H_sim_value_before_rep5[,i])
832   i = i+1
833 }
834
835 H_mean_diff = data.frame(matrix(nrow=1000,ncol=1))
836 H_mean_diff[,] = 0
837 H_mean_diff2 = data.frame(matrix(nrow=1000,ncol=1))
838 H_mean_diff2[,] = 0
839 H_mean_diff3 = data.frame(matrix(nrow=1000,ncol=1))
840 H_mean_diff3[,] = 0
841 H_mean_diff4 = data.frame(matrix(nrow=1000,ncol=1))
842 H_mean_diff4[,] = 0
843 H_mean_diff5 = data.frame(matrix(nrow=1000,ncol=1))
844 H_mean_diff5[,] = 0
845
846 for (i in 1:1000){
847   H_mean_diff[i,1] = sum(H_sim_diff[i,])/52
848   H_mean_diff2[i,1] = sum(H_sim_diff2[i,])/52
849   H_mean_diff3[i,1] = sum(H_sim_diff3[i,])/52
850   H_mean_diff4[i,1] = sum(H_sim_diff4[i,])/52
851   H_mean_diff5[i,1] = sum(H_sim_diff5[i,])/52
852   i = i+1
853 }
854
855 H_mean_put_price = data.frame(matrix(nrow=1,ncol=53))
856 H_mean_put_price[,] = 0
857 H_mean_put_price2 = data.frame(matrix(nrow=1,ncol=53))
858 H_mean_put_price2[,] = 0
859 H_mean_put_price3 = data.frame(matrix(nrow=1,ncol=53))
860 H_mean_put_price3[,] = 0
861 H_mean_put_price4 = data.frame(matrix(nrow=1,ncol=53))
862 H_mean_put_price4[,] = 0
863 H_mean_put_price5 = data.frame(matrix(nrow=1,ncol=53))
864 H_mean_put_price5[,] = 0
865
866 for (i in 1:53){
867   H_mean_put_price[1,i] = mean(H_bs_eput[,i])
868   H_mean_put_price2[1,i] = mean(H_bs_eput2[,i])
869   H_mean_put_price3[1,i] = mean(H_bs_eput3[,i])
870   H_mean_put_price4[1,i] = mean(H_bs_eput4[,i])

```

70:49  Simulating Stock Prices ↕

```

869 H_mean_put_price3[1,i] = mean(H_bs_eput3[,i])
870 H_mean_put_price4[1,i] = mean(H_bs_eput4[,i])
871 H_mean_put_price5[1,i] = mean(H_bs_eput5[,i])
872 i = i+1
873 }
874
875 H_mean_sim_price = data.frame(matrix(nrow=1,ncol=53))
876 H_mean_sim_price[,] = 0
877 H_mean_sim_price2 = data.frame(matrix(nrow=1,ncol=53))
878 H_mean_sim_price2[,] = 0
879 H_mean_sim_price3 = data.frame(matrix(nrow=1,ncol=53))
880 H_mean_sim_price3[,] = 0
881 H_mean_sim_price4 = data.frame(matrix(nrow=1,ncol=53))
882 H_mean_sim_price4[,] = 0
883
884 for (i in 1:53){
885   H_mean_sim_price[1,i] = mean(H_sim_price[,i])
886   H_mean_sim_price2[1,i] = mean(H_sim_price2[,i])
887   H_mean_sim_price3[1,i] = mean(H_sim_price3[,i])
888   H_mean_sim_price4[1,i] = mean(H_sim_price4[,i])
889   i = i+1
890 }
891
892
893 g = 1:53
894 ymin = min(H_mean_sim_price[1,],H_mean_sim_price2[1,],H_mean_sim_price3[1,],
895            H_mean_sim_price4[1,],mean_sim_price[1,])
896 ymax = max(H_mean_sim_price[1,],H_mean_sim_price2[1,],H_mean_sim_price3[1,],
897            H_mean_sim_price4[1,],mean_sim_price[1,])
898 plot(x=(g-1),y=H_mean_sim_price[1,g],xlab="weeks",ylab="Mean Stock Price ($)",
899      main="Mean Simulated Stock Price Paths",type='l',
900      lwd=2,lty=1,col="red",ylim=c(ymin,ymax))
901 lines(x=(g-1),y=H_mean_sim_price3[1,g],lwd=2,col="blue")
902 lines(x=(g-1),y=H_mean_sim_price4[1,g],lwd=2,col="darkgreen")
903 lines(x=(g-1),y=mean_sim_price[1,g],lwd=2,col="purple")
904 lines(x=(g-1),y=H_mean_sim_price2[1,g],lwd=2,col="yellow")

```

```

903 lines(x=(g-1),y=mean_sim_price[1,g],lwd=2,col="purple")
904 lines(x=(g-1),y=H_mean_sim_price2[1,g],lwd=2,col="yellow")
905 labels <- c("Stochastic Volatility",
906            "Lognormal Jumps","Double Exponential Jumps","Stochastic Interest Rate","Stochastic volatility with Lognormal Jumps")
907 legend("topleft",inset=.05,labels,lwd=2,lty=c(1,1,1,1,1),col=c("red","blue","darkgreen","purple","yellow"),cex = 0.55)
908
909 m_H_vol = H_vol
910 rownames(m_H_vol) = paste("trial", seq(1000), sep="")
911 colnames(m_H_vol) = paste("time", seq(53), sep="")
912 dat = as.data.frame(m_H_vol)
913 dat$trial = rownames(dat)
914 mdat = melt(dat, id.vars="trial")
915 mdat$time = as.numeric(gsub("time", "", mdat$variable))
916 pvol = ggplot(mdat, aes(x=time, y=value, group=trial)) +
917   ggtitle(expression("Instantaneous volatility under Heston model",
918                      "Stochastic volatility")) +
919   labs(x="weeks",y="volatility") +
920   theme_bw() +
921   theme(panel.grid=element_blank()) +
922   geom_line(size=0.2, alpha=0.1, col="blue")
923
924 m_H_int_rate = H_int_rate
925 rownames(m_H_int_rate) = paste("trial", seq(1000), sep="")
926 colnames(m_H_int_rate) = paste("time", seq(53), sep="")
927 dat = as.data.frame(m_H_int_rate)
928 dat$trial = rownames(dat)
929 mdat = melt(dat, id.vars="trial")
930 mdat$time = as.numeric(gsub("time", "", mdat$variable))
931 pir = ggplot(mdat, aes(x=time, y=value, group=trial)) +
932   ggtitle("Interest Rate under Vasicek Model") +
933   labs(x="weeks",y="Interest Rate") +
934   theme_bw() +
935   theme(panel.grid=element_blank()) +
936   geom_line(size=0.2, alpha=0.1, col="red")
937

```

```

937
938 H_mean_vol = data.frame(matrix(nrow=1,ncol=53))
939 H_mean_vol[,] = 0
940 H_mean_int_rate = data.frame(matrix(nrow=1,ncol=53))
941 H_mean_int_rate[,] = 0
942 H_mean_log = data.frame(matrix(nrow=1,ncol=53))
943 H_mean_log[,] = 0
944 H_mean_dexp = data.frame(matrix(nrow=1,ncol=53))
945 H_mean_dexp[,] = 0
946
947 for (i in 1:53){
948   H_mean_vol[1,i] = mean(H_vol[,i])
949   H_mean_int_rate[1,i] = mean(H_int_rate[,i])
950   H_mean_log[1,i] = mean(H_cum_lognormal[,i])
951   H_mean_dexp[1,i] = mean(H_cum_doubleexp[,i])
952   i = i+1
953 }
954
955 g = 1:53
956 plot(x=(g-1),y=H_mean_int_rate[1,g] ,xlab="weeks", ylab="Mean Interest Rate",
957      main="Mean Interest Rate under Vasicek Model",type='l',
958      lwd=2,lty=1,col="red")
959
960 g = 1:53
961 plot(x=(g-1),y=H_mean_vol[1,g] ,xlab="weeks", ylab="Mean volatility",
962      sub = "Stochastic volatility",
963      main="Mean Instantaneous volatility under Heston model",type='l',
964      lwd=2,lty=1,col="red")
965
966 g = 1:53
967 plot(x=(g-1),y=H_mean_log[1,g] ,xlab="weeks", ylab="Mean Cumulative Lognormal Jump",
968      main="Mean Cumulative Weekly Lognormal Jumps",type='l',
969      lwd=2,lty=1,col="red")
970
971 g = 1:53
972 plot(x=(g-1),y=H_mean_dexp[1,g] ,xlab="weeks", ylab="Mean Cumulative Double Exponential Jump",
973      main="Mean Cumulative Weekly Double Exponential Jumps",type='l',
974      lwd=2,lty=1,col="red")
975
976 f = xy.coords(x=H_mean_sim_price[1,],y=H_mean_vol[1,])
977 plot(f,xlab="Mean Stock Price", ylab="Mean Instantaneous volatility",
978      main="Mean Put Option Price vs Instantaneous volatility",
979      sub = "Stochastic volatility",
980      pch=1,cex=0.7,lwd=1.8,col='red')
981
982 f = xy.coords(x=mean_sim_price[1,],y=H_mean_int_rate[1,])
983 plot(f,xlab="Mean Stock Price", ylab="Mean Interest Rate",
984      main="Mean Put Option Price vs Mean Interest Rate",
985      pch=1,cex=0.7,lwd=1.8,col='red')
986
987 # Last Graphs 5 sections 4 each = 20 -----
988
989 #1
990 g = 1:1000
991 f = xy.coords(x=g,y=H_mean_diff[g,1])
992 plot(f,xlab="simulation", ylab="Mean Absolute Difference ($)",cex.main = 0.75,
993      main="Mean Absolute Difference between Replication Portfolio and Put Option for each simulation",
994      sub= "Stochastic volatility",
995      pch=1,cex=0.7,lwd=1.8,col='red')
996
997 m_H_sim_diff = H_sim_diff
998 rownames(m_H_sim_diff) = paste("trial", seq(1000), sep="")
999 colnames(m_H_sim_diff) = paste("time", seq(53), sep="")
1000 dat = as.data.frame(m_H_sim_diff)
1001 dat$trial = rownames(dat)
1002 mdat = melt(dat, id.vars="trial")
1003 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1004 a1 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1005   qqttitle(expression("Absolute Differences (Stochastic volatility)",
70:49

```

Simulating Stock Prices ↕


```

1003 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1004 a1 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1005   ggtitle(expression("Absolute Differences (Stochastic volatility)",
1006     "Stochastic volatility")) +
1007   labs(x="weeks", y="Absolute Deviation ($)") +
1008   theme_bw() +
1009   theme(plot.title = element_text(size=10)) +
1010   theme(panel.grid=element_blank()) +
1011   geom_line(size=0.1, alpha=0.1, col="blue")
1012
1013 f = xy.coords(x=H_mean_rep_value[,1], y=H_mean_put_price[,1])
1014 plot(f, xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price", cex.main = 0.85,
1015   sub= "Stochastic volatility",
1016   main="Mean Portfolio Value Before Replication vs After Replication", pch=1, cex=0.7, lwd=1.8, col='red')
1017 abline(0,1,col="blue",lwd=1, lty=2)
1018
1019 test <- unlist(H_sim_diff)
1020 hist(test, xlab="Absolute Difference ($)", ylab="Frequency",
1021   sub= "Stochastic volatility",
1022   main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value",
1023   cex.main=0.75, breaks=100, xlim=c(0,13))
1024
1025 #2
1026 g = 1:1000
1027 f = xy.coords(x=g, y=H_mean_diff2[g,1])
1028 plot(f, xlab="Simulation", ylab="Mean Absolute Difference ($)", cex.main = 0.75,
1029   main="Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation",
1030   sub= "Stochastic volatility with Lognormal Jumps",
1031   pch=1, cex=0.7, lwd=1.8, col='red')
1032
1033 m_H_sim_diff2 = H_sim_diff2
1034 rownames(m_H_sim_diff2) = paste("trial", seq(1000), sep="")
1035 colnames(m_H_sim_diff2) = paste("time", seq(53), sep="")
1036 dat = as.data.frame(m_H_sim_diff2)
1037 dat$trial = rownames(dat)
1038 mdat = melt(dat, id.vars="trial")
1039
270:49 Simulating Stock Prices

```

```

1038 mdat = melt(dat, id.vars="trial")
1039 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1040 a2 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1041   ggtitle(expression("Absolute Differences (Stochastic volatility with Lognormal Jumps)",
1042     "Stochastic volatility with Lognormal Jumps")) +
1043   labs(x="weeks", y="Absolute Deviation ($)") +
1044   theme_bw() +
1045   theme(plot.title = element_text(size=10)) +
1046   theme(panel.grid=element_blank()) +
1047   geom_line(size=0.1, alpha=0.1, col="blue")
1048
1049 f = xy.coords(x=H_mean_rep_value2[,1], y=H_mean_put_price2[,1])
1050 plot(f, xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price", cex.main = 0.85,
1051   sub= "Stochastic volatility with Lognormal Jumps",
1052   main="Mean Portfolio Value Before Replication vs After Replication", pch=1, cex=0.7, lwd=1.8, col='red')
1053 abline(0,1,col="blue",lwd=1, lty=2)
1054
1055 test <- unlist(H_sim_diff2)
1056 hist(test, xlab="Absolute Difference ($)", ylab="Frequency",
1057   sub= "Stochastic volatility with Lognormal Jumps",
1058   main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option Value",
1059   cex.main=0.75, breaks=100, xlim=c(0,13))
1060
1061 #3
1062 g = 1:1000
1063 f = xy.coords(x=g, y=H_mean_diff3[g,1])
1064 plot(f, xlab="Simulation", ylab="Mean Absolute Difference ($)", cex.main = 0.75,
1065   main="Mean Absolute Difference between Replication Portfolio and Put Option for each Simulation",
1066   sub= "Lognormal Jump-Diffusion",
1067   pch=1, cex=0.7, lwd=1.8, col='red')
1068
1069 m_H_sim_diff3 = H_sim_diff3
1070 rownames(m_H_sim_diff3) = paste("trial", seq(1000), sep="")
1071 colnames(m_H_sim_diff3) = paste("time", seq(53), sep="")
1072 dat = as.data.frame(m_H_sim_diff3)
1073 dat$trial = rownames(dat)
1074
170:49 Simulating Stock Prices

```

```

1073 dat$trial = rownames(dat)
1074 mdat = melt(dat, id.vars="trial")
1075 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1076 a3 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1077   ggtitle(expression("Absolute Differences (Lognormal Jump-Diffusion)",
1078     "Lognormal Jump-Diffusion")) +
1079   labs(x="weeks", y="Absolute Deviation ($)") +
1080   theme_bw() +
1081   theme(plot.title = element_text(size=10)) +
1082   theme(panel.grid=element_blank()) +
1083   geom_line(size=0.1, alpha=0.1, col="blue")
1084
1085 f = xy.coords(x=H_mean_rep_value3[,1], y=H_mean_put_price3[,1])
1086 plot(f, xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price", cex.main = 0.85,
1087   sub= "Lognormal Jump-Diffusion",
1088   main="Mean Portfolio Value Before Replication vs After Replication", pch=1, cex=0.7, lwd=1.8, col='red')
1089 abline(0,1,col="blue",lwd=1, lty=2)
1090
1091 test <- unlist(H_sim_diff3)
1092 hist(test, xlab="Absolute Difference ($)", ylab="Frequency",
1093   sub= "Lognormal Jump-Diffusion",
1094   main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option value",
1095   cex.main=0.75, breaks=100, xlim=c(0,13))
1096
1097 #4
1098 g = 1:1000
1099 f = xy.coords(x=g, y=H_mean_diff5[g,1])
1100 plot(f, xlab="Simulation", ylab="Mean Absolute Difference ($)", cex.main = 0.75,
1101   main="Mean Absolute Difference between Replication Portfolio and Put Option for each simulation",
1102   sub= "Double Exponential Jump-Diffusion",
1103   pch=1, cex=0.7, lwd=1.8, col='red')
1104
1105 m_H_sim_diff5 = H_sim_diff5
1106 rownames(m_H_sim_diff5) = paste("trial", seq(1000), sep="")
1107 colnames(m_H_sim_diff5) = paste("time", seq(53), sep="")
1108 dat = as.data.frame(m_H_sim_diff5)
1109
1107 colnames(m_H_sim_diff5) = paste("time", seq(53), sep="")
1108 dat = as.data.frame(m_H_sim_diff5)
1109 dat$trial = rownames(dat)
1110 mdat = melt(dat, id.vars="trial")
1111 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1112 a4 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1113   ggtitle(expression("Absolute Differences (Double Exponential Jump-Diffusion)",
1114     "Double Exponential Jump-Diffusion")) +
1115   labs(x="weeks", y="Absolute Deviation ($)") +
1116   theme_bw() +
1117   theme(plot.title = element_text(size=10)) +
1118   theme(panel.grid=element_blank()) +
1119   geom_line(size=0.1, alpha=0.1, col="blue")
1120
1121 f = xy.coords(x=H_mean_rep_value5[,1], y=H_mean_put_price5[,1])
1122 plot(f, xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price", cex.main = 0.85,
1123   sub= "Double Exponential Jump-Diffusion",
1124   main="Mean Portfolio Value Before Replication vs After Replication", pch=1, cex=0.7, lwd=1.8, col='red')
1125 abline(0,1,col="blue",lwd=1, lty=2)
1126
1127 test <- unlist(H_sim_diff5)
1128 hist(test, xlab="Absolute Difference ($)", ylab="Frequency",
1129   sub= "Double Exponential Jump-Diffusion",
1130   main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option value",
1131   cex.main=0.75, breaks=100, xlim=c(0,13))
1132
1133 #5
1134 g = 1:1000
1135 f = xy.coords(x=g, y=H_mean_diff4[g,1])
1136 plot(f, xlab="Simulation", ylab="Mean Absolute Difference ($)", cex.main = 0.75,
1137   main="Mean Absolute Difference between Replication Portfolio and Put Option for each simulation",
1138   sub= "Stochastic Interest Rate",
1139   pch=1, cex=0.7, lwd=1.8, col='red')
1140

```

```



1141 m_H_sim_diff4 = H_sim_diff4
1142 rownames(m_H_sim_diff4) = paste("trial", seq(1000), sep="")
1143 colnames(m_H_sim_diff4) = paste("time", seq(53), sep="")
1144 dat = as.data.frame(m_H_sim_diff4)
1145 dat$trial = rownames(dat)
1146 mdat = melt(dat, id.vars="trial")
1147 mdat$time = as.numeric(gsub("time", "", mdat$variable))
1148 a5 = ggplot(mdat, aes(x=time, y=value, group=trial)) +
1149   ggtitle(expression("Absolute Differences (Stochastic Interest Rate) ",
1150     "Stochastic Interest Rate")) +
1151     labs(x="Weeks",y="Absolute Deviation ($)") +
1152     theme_bw() +
1153     theme(plot.title = element_text(size=10)) +
1154     theme(panel.grid=element_blank()) +
1155     geom_line(size=0.1, alpha=0.1, col="blue")
1156
1157 f = xy.coords(x=H_mean_rep_value4[,1],y=H_mean_put_price4[,1])
1158 plot(f,xlab="Mean Portfolio Value Before Replication", ylab="Mean Put Option Price",cex.main = 0.85,
1159   sub= "Stochastic Interest Rate",
1160   main="Mean Portfolio Value Before Replication vs After Replication",pch=1,cex=0.7,lwd=1.8,col='red')
1161 abline(0,1,col="blue",lwd=1,pty=2)
1162
1163 test <- unlist(H_sim_diff4)
1164 hist(test,xlab="Absolute Difference ($)", ylab="Frequency",
1165   sub= "Stochastic Interest Rate",
1166   main="Histogram of the 52000 Absolute Differences between Replicating Portfolio and Put Option value",
1167   cex.main=0.75,breaks=100,xlim=c(0,13))
1168
1169
1170
1171 H_vol_price = data.frame(matrix(nrow=1000,ncol=52))
1172 H_vol_price[,] = 0
1173 H_vol_price2 = data.frame(matrix(nrow=1000,ncol=52))
1174 H_vol_price2[,] = 0

```

```

1171 H_vol_price = data.frame(matrix(nrow=1000,ncol=52))
1172 H_vol_price[,] = 0
1173 H_vol_price2 = data.frame(matrix(nrow=1000,ncol=52))
1174 H_vol_price2[,] = 0
1175 H_vol_price3 = data.frame(matrix(nrow=1000,ncol=52))
1176 H_vol_price3[,] = 0
1177 H_vol_price4 = data.frame(matrix(nrow=1000,ncol=52))
1178 H_vol_price4[,] = 0
1179 H_vol_price5 = data.frame(matrix(nrow=1000,ncol=52))
1180 H_vol_price5[,] = 0
1181
1182 for (i in 2:53){
1183   for (j in 1:1000){
1184     H_vol_price[j,i-1] = abs((sim_price[j,i]-sim_price[j,i-1])/sim_price[j,i-1])
1185     H_vol_price2[j,i-1] = abs((H_sim_price[j,i]-H_sim_price[j,i-1])/H_sim_price[j,i-1])
1186     H_vol_price3[j,i-1] = abs((H_sim_price3[j,i]-H_sim_price3[j,i-1])/H_sim_price3[j,i-1])
1187     H_vol_price4[j,i-1] = abs((H_sim_price4[j,i]-H_sim_price4[j,i-1])/H_sim_price4[j,i-1])
1188     H_vol_price5[j,i-1] = abs((H_sim_price2[j,i]-H_sim_price2[j,i-1])/H_sim_price2[j,i-1])
1189     j = j+1
1190   }
1191   i = i+1
1192 }
1193
1194 #this is to show volatility is not constant
1195 sum(H_vol_price)/(52*1000)
1196 sum(H_vol_price2)/(52*1000)
1197 sum(H_vol_price3)/(52*1000)
1198 sum(H_vol_price4)/(52*1000)
1199 sum(H_vol_price5)/(52*1000)

```

270:49  Simulating Stock Prices 

Console

Appendix B

References

<https://www.berryfx.com/advantages-disadvantages-forward-contracts-currency-options/>
[Accessed on February 18th]

<http://www.timworrall.com/fin-40008/forwards.pdf>
[Accessed on February 18th]

<https://www.quora.com/What-is-marked-to-market-in-future-contract>
[Accessed on February 18th]

<http://people.stern.nyu.edu/adamodar/pdfiles/valn2ed/ch34.pdf>
[Accessed on February 20th]

<http://www.investopedia.com/exam-guide/cfa-level-1/derivatives/fundamental-differences-between-futures-forwards.asp>
[Accessed on February 20th]

<https://tradingsim.com/blog/4-key-differences-between-futures-and-forward-contracts/>
[Accessed on February 20th]

<http://janroman.dhis.org/finance/Books%20Notes%20Thesises%20etc/Shrive%20Finance/chap31.pdf>
[Accessed on February 24th]

<http://www.investopedia.com/investing/how-interest-rates-affect-stock-market/>
[Accessed on February 24th]

<https://www.forbes.com/sites/gurufocus/2016/07/25/what-will-happen-to-the-stock-market-when-interest-rates-rise-may-surprise-you/#5510fcb37e63>
[Accessed on February 24th]

http://wwwf.imperial.ac.uk/~mdavis/FDM11/BINOMIAL_AMERICAN_REV.PDF
[Accessed on February 19th]

<http://www.uio.no/studier/emner/sv/oekonomi/ECON4510/v11/undervisningsmateriale/lect2803.pdf>
[Accessed on February 19th]

<http://www.columbia.edu/~mh2078/BlackScholesCtsTime.pdf>
[Accessed on February 23rd]

<https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities/put-call-options/v/put-call-parity>
[Accessed on February 23rd]

http://www.columbia.edu/~mn2/broadie/Assets/broadie_kaya_exact_sim_or_2006.pdf
[Accessed on February 26th]