

Homework 2, MATH 5010

Zongyi Liu

Feb 28, 2025

Question 1

What is the difference between historic and implied volatilities? Which one is higher for Microsoft stock MSFT? Go to the Bloomberg terminal and type: `MSFT <Equity> HIVG <Go>`. Print the output.

Answer

The Historical Volatility is also called Realized Volatility or Statistical Volatility, and it measures the actual volatility of an asset's price over a specific past period. It is calculated using historical price data (e.g., daily, weekly, or monthly returns).

Historic Volatility is typically calculated as the standard deviation of the asset's logarithmic returns over a given time frame. The formula is:

$$\text{Historic Volatility} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (r_i - \bar{r})^2}$$

where:

- r_i : logarithmic return at time i ,
- \bar{r} : average return over the period,
- N : number of observations.

As for the Implied Volatility (IV), it is a forward-looking measure of volatility derived from the market price of an option. It represents the market's expectation of future volatility over the option's life.

Implied Volatility is not directly calculated but is inferred using an options pricing model (e.g. the Black-Scholes model). It is the value of volatility that, when plugged into the model, matches the market price of the option. The Black-Scholes formula for a call option is:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2)$$

where:

- C : market price of the call option,
- S_0 : current price of the underlying asset,
- K : strike price,
- r : risk-free rate,
- T : time to expiration,
- $N(d)$: cumulative distribution function of the standard normal distribution,
- d_1 and d_2 : functions of volatility (σ).

Implied Volatility is the value of σ that satisfies this equation.

Finally, for their relationship, over time, implied volatility tends to converge toward historic volatility as the future becomes the past. Implied volatility can be higher or lower than historic volatility depending on market expectations (e.g., during periods of uncertainty or calm).

We can use a table to illustrate this:

Aspect	Historic Volatility	Implied Volatility
Time Frame	Based on past price movements.	Based on future expectations.
Data Used	Historical price data.	Market price of options.
Calculation	Calculated using statistical methods (e.g., standard deviation).	Derived from options pricing models (e.g., Black-Scholes).
Purpose	Measures past price fluctuations.	Predicts future price fluctuations.
Market Sentiment	Does not reflect market sentiment.	Reflects market sentiment and expectations.
Use in Trading	Used for risk analysis and benchmarking.	Used for options pricing and trading strategies.

Here is the plot provided by Bloomberg Terminal, and the timespan I set was Sept 16, 2020 to Feb 20, 2025.

Figure 1: The Results after using MSFT <Enquiry> HIVG <GO>



Take the date Feb 28, 2025 as the last date for consideration, for a 30-days range, the historical volatility (HV) is 25.99%, which is higher than the implied volatility (IV), which was around 24.51%. The exact data can be found here.

However, if we reconsider it in a large time span, the overall Historical Volatility is smaller than the Implied Volatility. It can be easily seen from the graph, and we can also scrutinize the 180-day-data.

Figure 2: A Simple Plot Showing HV and IV of MSFT



Question 2

What is the volatility smile? Using Bloomberg terminal function SKEW (CLJ5 Comdty SKEW <Go>) plot the implied volatility smiles for TYH5 Comdty, CLJ5 Comdty, NGJ25 Comdty, ESH5 Index, SPY Equity. What are these securities? For description type CLJ5 Comdty DES. Compare their smiles. Submit printouts.

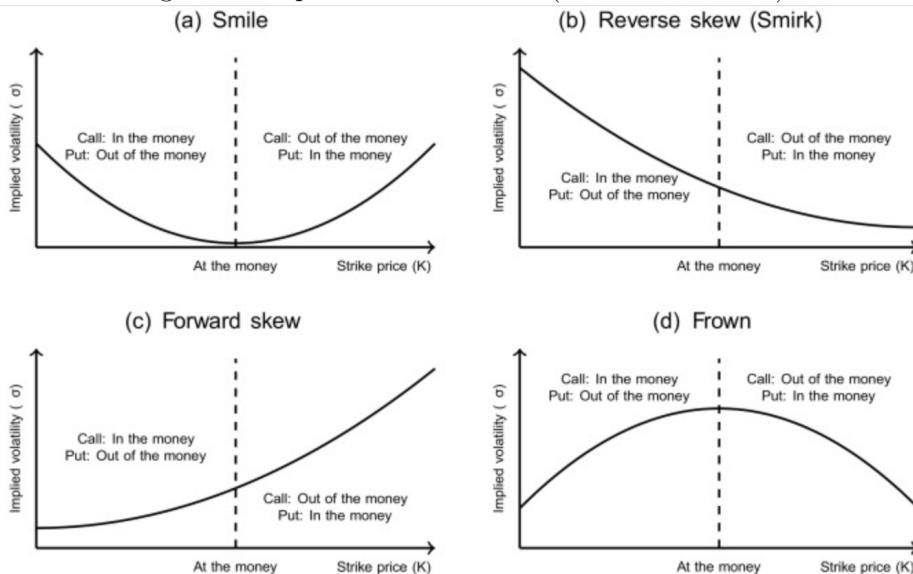
Answer

In the Textbook, the author defines Volatility Smile as "A plot of the implied volatility of an option with a certain life as a function of its strike price." (Hull, 409)

In my understanding, Volatility Smile is a pattern observed in the implied volatility of options across different strike prices, where implied volatility is higher for both out-of-the-money (OTM) puts and calls compared to at-the-money (ATM) options, creating a "smile" shape when plotted. This phenomenon contradicts the assumption of constant volatility in models like Black-Scholes and arises because markets often price in higher uncertainty for extreme price movements (both up and down).

The x-axis is Strike Price, whereas the y-axis is Implied Volatility.

Figure 3: Simple Illustration of VS (cred to Soini et al.)



In Bloomberg, they represent:

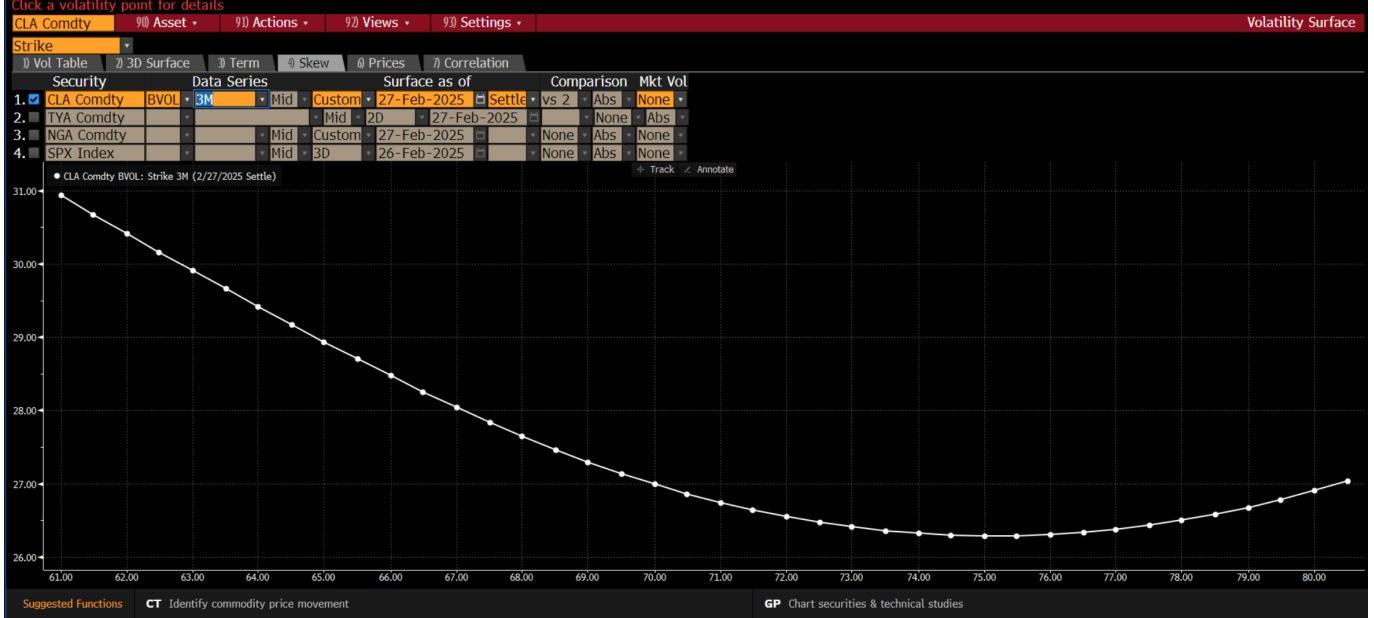
- TYH5 Comdty: U.S. 10-Year Treasury Futures Contract ([March 2015](#)).
- CLJ5 Comdty: Crude Oil Futures ([May 2015](#)).
- NGJ25 Comdty: Natural Gas Futures ([January 2025](#)).
- ESH5 Index: ([S&P 500 E-mini Futures \(March 2015\)](#).
- SPY Equity: SPDR S&P 500 ETF Trust.

Here is the graph of volatility smile in Bloomberg Terminal. The model I set was **Strike** instead of **Delta** or so, because in **Delta** mode, there are only four comparable niches, and the data for **ESH5 Index** is not available to show their volatility smiles.

Take **Strike** as the x-axis, the Volatility Surface has a Smirk Shape, or Reverse Skew for CLJ5 Comdty, and it reflects a market expectation of greater downside risk than upside potential. It is characterized by higher implied volatility for OTM puts compared to OTM calls and is often driven by bearish sentiment, economic uncertainty, or fear of sharp declines. In such scenarios, OTM puts become more expensive due to higher implied volatility, while OTM calls are relatively cheaper. Traders may use strategies like put spreads or risk reversals to capitalize on this skew, while investors might hedge against potential price declines.

For both TYH5 Comdty and NGJ5 Comdty, we have a shape as Forward Skew, meaning its implied volatility is higher for out-of-the-money (OTM) calls (higher strike prices) compared to out-of-the-money (OTM) puts (lower

Figure 4: Volatility Surface of CLJ5 Comdty



strike prices). This creates an upward-sloping asymmetry in the volatility curve. This pattern indicates that the market expects greater upside potential (price increases) than downside risk (price decreases), often driven by bullish sentiment, anticipated positive events (e.g., earnings reports, supply constraints), or speculative demand. In such scenarios, OTM calls become more expensive due to higher implied volatility, while OTM puts are relatively cheaper. Traders may use strategies like call spreads or risk reversals to capitalize on this skew, while investors might hedge against potential price increases.

Figure 5: Volatility Surface of TYH5 Comdty

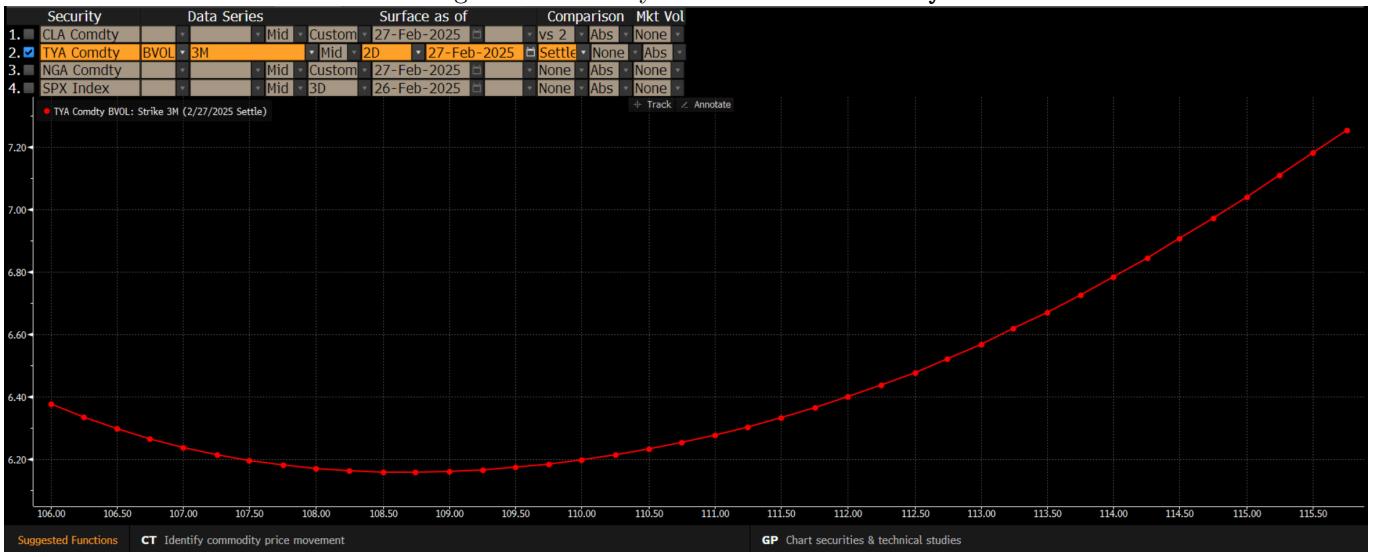
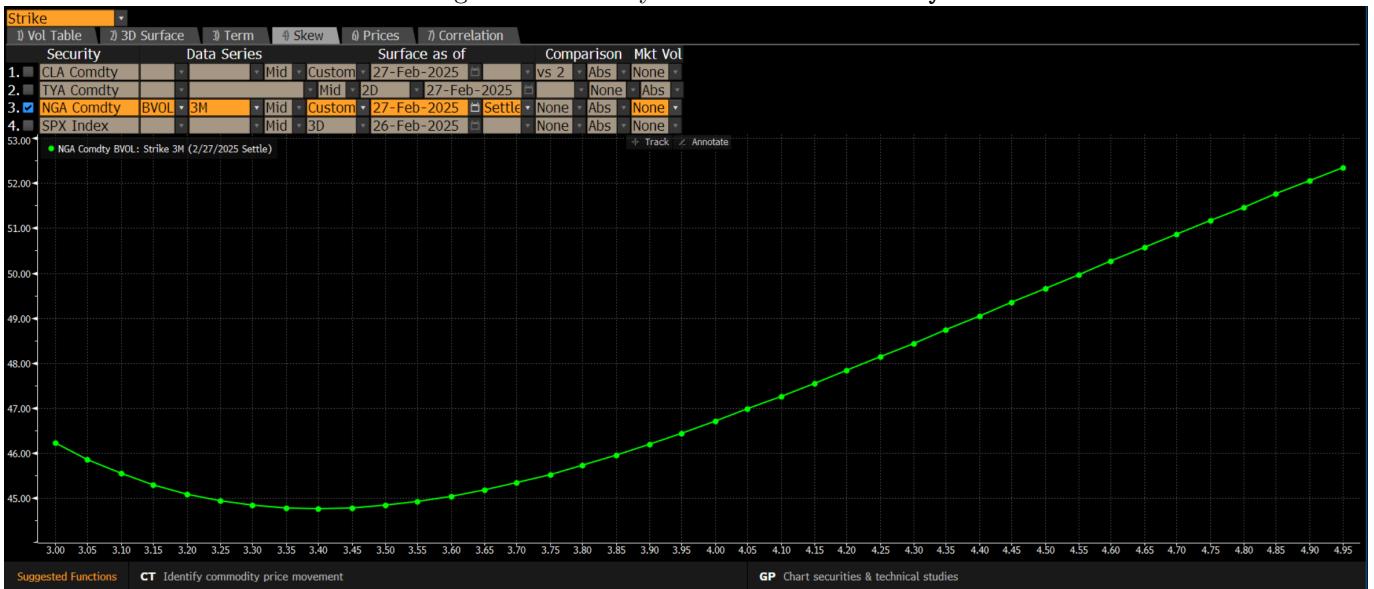


Figure 6: Volatility Surface of NGJ5 Comdty



Here we get a downward sloping graph of SPY Equity and ESH5 Index; typically a downward-sloping volatility surface in the 3-month range indicates that implied volatility decreases as strike prices increase, reflecting a negative volatility skew. This pattern often signals bearish market sentiment, with investors more concerned about downside risks (e.g., potential price drops) than upside potential. Higher implied volatilities for lower strike prices (out-of-the-money puts) suggest increased demand for downside protection, possibly due to market stress, upcoming events like earnings announcements, or historical patterns of sharp declines. For traders, this skew implies that put options are more expensive, while call options are cheaper, influencing strategies such as put spreads or risk reversals.

Figure 7: Volatility Surface of SPY Equity

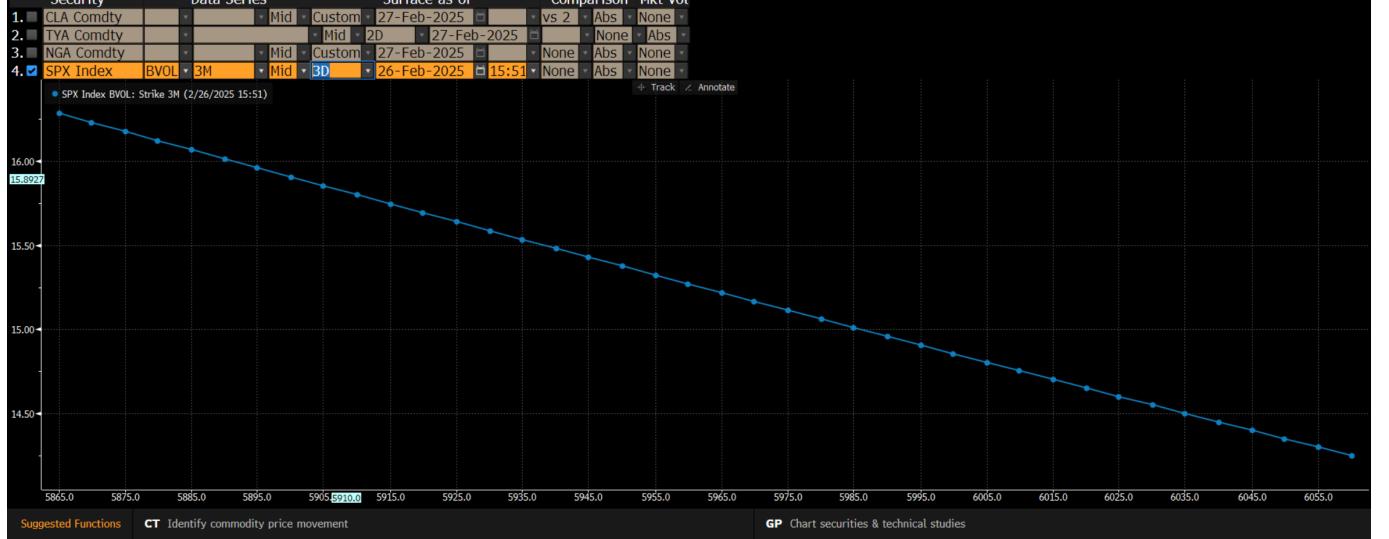
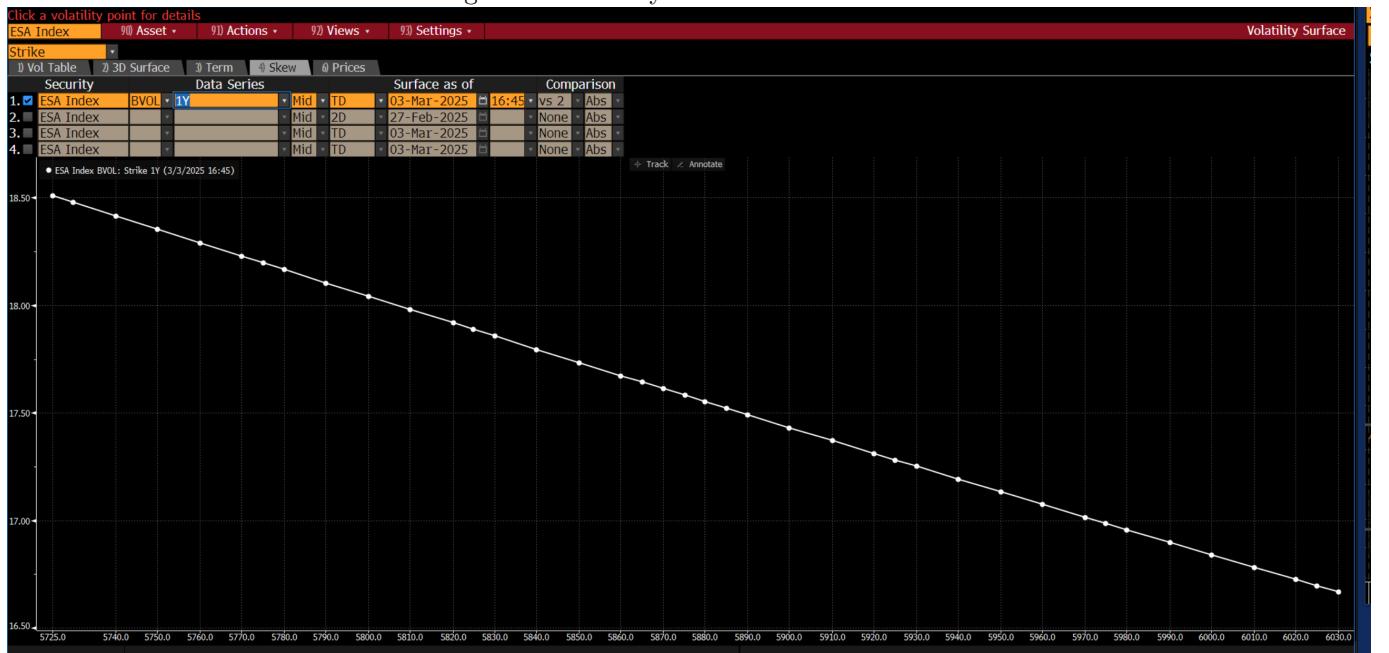


Figure 8: Volatility Surface of ESH5 Index



Question 3

Suppose the stock price is 70, the risk-free rate is 5% continuously compounded. What is the price of a 1 year call struck at 70 if the volatility is 0. How would you hedge the call. Check your answer with the option calculator making volatility smaller and smaller.

Answer

Here I used the integration of Fast Fourier Transformation (FFT) and Fractional Fast Fourier Transformation (FrFFT) for this problem; there are 4 methods to be chosen, GBM, Heston, Variance Gamma, and Black-Scholes. Here I used Black-Scholes.

If $\sigma = 0$ and the continuously compounded rate is 0, the stock price is deterministic and grows at rate r . In one year, it is thus worth $70 \cdot e^{0.05} \approx 73.5889$. The strike is $K = 70$. Our payoff is thus 3.588 . Discounting at rate r , we get as today's fair option price $3.588 \cdot e^{-0.05} \approx 3.413$. When I set $r = 0.05$, $q = 0$, $\sigma = 0$, the printout is:

```
=====
```

Model is BlackScholes

```
-----
```

Option via Integration: for strike 70.0000 the option premium is 3.4155

FrFFT execution time was 0.0215478

If $\sigma = 0$ and the continuously compounded rate is 0.03, the stock price is worth $70 \cdot e^{0.05-0.02} \approx 71.4141$. The strike is $K = 70$. Our payoff is thus 1.414 . Discounting at rate $r-q$, we get as today's fair option price $1.414 \cdot e^{-0.02} \approx 1.3861$. When I set $r = 0.05$, $q = 0.03$, $\sigma = 0$, the printout is:

```
=====
```

Model is BlackScholes

```
-----
```

Option via Integration: for strike 70.0000 the option premium is 1.3610

FrFFT execution time was 0.0130410

My python codes are as follows:

```
1 import warnings
2 warnings.filterwarnings("ignore")
3
4
5 import numpy as np
6 import cmath
7 import math
8 import time
9
10 # Fixed Parameters
11 S0 = 70
12 K = 70
13 k = math.log(K)
14 r = 0.05
15 q = 0.03
16
17 # Parameters for FFT and FrFFT
18
19 n_FFT = 9
20 N_FFT = 2**n_FFT
21
22 n_FrFFT = 9
23 N_FrFFT = 2**n_FrFFT
24
25 N = 1000 # it doesn't impact the numerical results if the code is correct
26
27 #step-size
28 eta = 0.25
29 # damping factor
30 alpha = 0.4
31
32 # step-size in log strike space
```

```

33 lda_FFT = (2*math.pi/N_FFT)/eta # lda is fixed under FFT
34 lda_FrFFT = 0.001 # lda is an adjustable parameter under FrFFT,
35
36
37 #Choice of beta
38 beta = np.log(S0)-N*lda_FFT/2
39 #beta = np.log(S0)-N*lda_FrFFT/2
40 #beta = np.log(K)
41
42 #model-specific Parameters
43 model = 'BlackScholes'
44
45 params = []
46 if (model == 'GBM'):
47
48     sig = 0.3
49     params.append(sig);
50
51 elif model == 'BlackScholes':
52     sigma = 0 # Volatility
53     params.append(sigma)
54
55 elif (model == 'VG'):
56
57     sig = 0.3
58     nu = 0.5
59     theta = -0.4
60     #
61     params.append(sig);
62     params.append(nu);
63     params.append(theta);
64
65
66 elif (model == 'Heston'):
67
68     kappa = 2.0
69     theta = 0.05
70     sig = 0.30
71     rho = -0.70
72     v0 = 0.04
73     #
74     params.append(kappa)
75     params.append(theta)
76     params.append(sig)
77     params.append(rho)
78     params.append(v0)
79
80 def generic_CF(u, params, S0, r, q, T, model):
81
82 if (model == 'GBM'):
83
84     sig = params[0]
85     mu = np.log(S0) + (r-q-sig**2/2)*T
86     a = sig*np.sqrt(T)
87     phi = np.exp(1j*mu*u-(a*u)**2/2)
88
89 elif (model == 'BlackScholes'):
90     sigma = params[0] # Volatility
91
92     mu = np.log(S0) + (r - q - 0.5 * sigma**2) * T # Drift
93     a = sigma * np.sqrt(T) # Standard deviation over the maturity period
94     phi = np.exp(1j * mu * u - 0.5 * a**2 * u**2) # Characteristic function for Black-

```

```

Scholes
96
97
98 elif(model == 'Heston'):
99
100 kappa = params[0]
101 theta = params[1]
102 sigma = params[2]
103 rho = params[3]
104 v0 = params[4]
105
106 tmp = (kappa-1j*rho*sigma*u)
107 g = np.sqrt((sigma**2)*(u**2+1j*u)+tmp**2)
108
109 pow1 = 2*kappa*theta/(sigma**2)
110
111 numer1 = (kappa*theta*T*tmp)/(sigma**2) + 1j*u*T*r + 1j*u*math.log(S0)
112 log_denum1 = pow1 * np.log(np.cosh(g*T/2)+(tmp/g)*np.sinh(g*T/2))
113 tmp2 = ((u*u+1j*u)*v0)/(g/np.tanh(g*T/2)+tmp)
114 log_phi = numer1 - log_denum1 - tmp2
115 phi = np.exp(log_phi)
116
117 #g = np.sqrt((kappa-1j*rho*sigma*u)**2+(u*u+1j*u)*sigma*sigma)
118 #beta = kappa-rho*sigma*1j*u
119 #tmp = g*T/2
120
121 #temp1 = 1j*(np.log(S0)+(r-q)*T)*u + kappa*theta*T*beta/(sigma*sigma)
122 #temp2 = -(u*u+1j*u)*v0/(g/np.tanh(tmp)+beta)
123 #temp3 = (2*kappa*theta/(sigma*sigma))*np.log(np.cosh(tmp)+(beta/g)*np.sinh(tmp))
124
125 #phi = np.exp(temp1+temp2-temp3);
126
127
128 elif (model == 'VG'):
129
130 sigma = params[0];
131 nu = params[1];
132 theta = params[2];
133
134 if (nu == 0):
135 mu = np.log(S0) + (r-q - theta - 0.5*sigma**2)*T
136 phi = np.exp(1j*u*mu) * np.exp((1j*theta*u-0.5*sigma**2*u**2)*T)
137 else:
138 mu = np.log(S0) + (r-q + np.log(1-theta*nu-0.5*sigma**2*nu)/nu)*T
139 phi = np.exp(1j*u*mu)*((1-1j*nu*theta*u+0.5*nu*sigma**2*u**2)**(-T/nu))
140
141 return phi
142 def evaluateIntegral(params, S0, K, r, q, T, alpha, eta, N, model):
143
144 # Just one strike at a time
145 # no need for Fast Fourier Transform
146
147 # discount factor
148 df = math.exp(-r*T)
149
150 sum1 = 0
151 for j in range(N):
152 nuJ = j*eta
153 psi_nuJ = df*generic_CF(nuJ-(alpha+1)*1j, params, S0, r, q, T, model)/((alpha + 1j*
154 nuJ)*(alpha+1+1j*nuJ))
155 if j == 0:
156 wJ = (eta/2)
157 else:

```

```

157 wJ = eta
158 sum1 += np.exp(-1j*nuJ*k)*psi_nuJ*wJ
159
160 cT_k = (np.exp(-alpha*k)/math.pi)*sum1
161
162 return np.real(cT_k)
163
164 def genericFFT(params, S0, K, r, q, T, alpha, eta, n, model):
165
166 N = 2**n
167
168 # step-size in log strike space
169 lda = (2*np.pi/N)/eta
170
171 #Choice of beta
172 #beta = np.log(S0)-N*lda/2
173 #beta = np.log(K)
174
175 # forming vector x and strikes km for m=1,...,N
176 km = np.zeros((N))
177 xX = np.zeros((N))
178
179 # discount factor
180 df = math.exp(-r*T)
181
182 nuJ = np.arange(N)*eta
183 psi_nuJ = generic_CF(nuJ-(alpha+1)*1j, params, S0, r, q, T, model)/((alpha + 1j*nuJ)*(alpha+1+1j*nuJ))
184
185 for j in range(N):
186 km[j] = beta+j*lda
187 if j == 0:
188 wJ = (eta/2)
189 else:
190 wJ = eta
191
192 xX[j] = np.exp(-1j*beta*nuJ[j])*df*psi_nuJ[j]*wJ
193
194 yY = np.fft.fft(xX)
195 cT_km = np.zeros((N))
196 for i in range(N):
197 multiplier = np.exp(-alpha*km[i])/math.pi
198 cT_km[i] = multiplier*np.real(yY[i])
199
200 return km, cT_km
201
202 def genericFrFFT(params, S0, K, r, q, T, alpha, eta, n, lda, model):
203
204 N = 2**n
205 gamma = eta*lda/(2*math.pi)
206
207 #Choice of beta
208 #beta = np.log(S0)-N*lda/2
209 beta = np.log(K)
210
211 # initialize x, y, z, and cT_km
212 km = np.zeros((N))
213 x = np.zeros((N))
214 y = np.zeros((2*N), dtype=np.complex128)
215 z = np.zeros((2*N), dtype=np.complex128)
216 cT_km = np.zeros((N))
217
218 # discount factor

```

```

219 df = math.exp(-r*T)
220
221 # compute x
222 nuJ = np.arange(N)*eta
223 psi_nuJ = generic_CF(nuJ-(alpha+1)*1j, params, S0, r, q, T, model)/((alpha + 1j*nuJ)
224 *(alpha+1+1j*nuJ))
225
226 for j in range(N):
227 km[j] = beta+j*lida
228 if j == 0:
229 wJ = (eta/2)
230 else:
231 wJ = eta
232 x[j] = np.exp(-1j*beta*nuJ[j])*df*psi_nuJ[j]*wJ
233
234 # set up y
235 for i in range(N):
236 y[i] = np.exp(-1j*math.pi*gamma*i**2)*x[i]
237 y[N:] = 0
238
239 # set up z
240 for i in range(N):
241 z[i] = np.exp(1j*math.pi*gamma*i**2)
242 z[N:] = z[:N][::-1]
243
244 # compute xi_hat
245 xi_hat = np.fft.ifft(np.fft.fft(y) * np.fft.fft(z))
246
247 # compute call prices
248 for i in range(N):
249 cT_km[i] = np.exp(-alpha*(beta + i*lida))/math.pi * (np.exp(-1j*math.pi*gamma*i**2)*
250 xi_hat[i]).real
251
252 return km, cT_km
253
254 print('')
255 print('=====')
256 print('Model is %s' % model)
257 print('-----')
258
259 T = 1
260
261 # FFT
262 print('')
263 start_time = time.time()
264 km, cT_km = genericFFT(params, S0, K, r, q, T, alpha, eta, n_FFT, model)
265 #cT_k = cT_km[0]
266 cT_k = np.interp(k, km, cT_km)
267
268 elapsed_time = time.time() - start_time
269
270 #cT_k = np.interp(np.log(k), km, cT_km)
271 print("Option via FFT: for strike %s the option premium is %6.4f" % (np.exp(k), cT_k))
272
273 print("Option via FFT: for strike %s the option premium is %6.4f" % (np.exp(k),
274 cT_km[0]))
275 print('FFT execution time was %0.7f' % elapsed_time)
276
277 # FrFFT
278 print('')
279 start_time = time.time()
280 km, cT_km = genericFrFFT(params, S0, K, r, q, T, alpha, eta, n_FrFFT, lda_FrFFT,
281 model)

```

```

277 # cT_k = cT_km[0]
278 cT_k = np.interp(k, km, cT_km)
279
280 elapsed_time = time.time() - start_time
281
282 # cT_k = np.interp(np.log(), km, cT_km)
283 print("Option via FrFFT: for strike %s the option premium is %6.4f" % (np.exp(k),
284   cT_k))
285 #print("Option via FFT: for strike %s the option premium is %6.4f" % (np.exp(k),
286   cT_km[0]))
287 print('FrFFT execution time was %0.7f' % elapsed_time)
288
289 # Integral
290 print(' ')
291 start_time = time.time()
292 cT_k = evaluateIntegral(params, S0, K, r, q, T, alpha, eta, N, model)
293 elapsed_time = time.time() - start_time
294 print("Option via Integration: for strike %s the option premium is %6.4f" % (np.exp(k),
295   cT_k))
296 print('Evaluation of integral time was %0.7f' % elapsed_time)

```

Question 4

Explain why an American option on a stock paying continuous dividend yield is always worth as much as its intrinsic value. Give a numerical example of a situation when European option is worth less than intrinsic value. (Give the numerical value of stock price, strike price, time to expiration, etc.)

Answer

Concept

Intrinsic value of a call option is the difference between the current price of the underlying asset and the strike price of the option, if the underlying asset's price is above the strike price. If the underlying asset's price is below the strike price, the call option has no intrinsic value. Thus it can be expressed as:

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

Here we have S to be the current stock price, and K to be the strike price

If the stock pays a continuous dividend yield, the stock price is expected to decrease as dividends are paid. This increases the incentive to exercise the call option early to avoid losing value from future dividends. Since the American option can be exercised early, the holder will exercise it if the intrinsic value exceeds the option's continuation value (the value of holding the option). Therefore, the American option is always worth at least its intrinsic value.

Numerical Example

For a the numerical example, we can set

- Stock price: $S = 100$
- Strike price: $K = 90$
- Time to expiration: 1 year
- Risk-free interest rate: 5%
- Continuous dividend yield: 3%
- Volatility: 20%

Using the Black-Scholes formula, the European call option price is approximately:

$$C_{\text{European}} = 12.03$$

Calculation of C_{European}

We have the Black-Scholes Formula:

$$C = Se^{-qt}N(d_1) - Ke^{-rt}N(d_2)$$

Where d_1 and d_2 can be calculated as:

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

$$d_2 = d_1 - \sigma\sqrt{t}$$

Given what we previously set: $S = 100$, $K = 90$, $r = 0.05$, $q = 0.03$, $\sigma = 0.20$, $t = 1$:

$$d_1 = \frac{\ln(100/90) + (0.05 - 0.03 + 0.5(0.2^2))(1)}{0.2\sqrt{1}} = \frac{\ln(1.1111) + (0.02 + 0.04)}{0.2} = \frac{0.1054 + 0.04}{0.2} = \frac{0.1454}{0.2} = 0.727$$

$$d_2 = 0.727 - 0.2 = 0.527$$

Then we can find $N(d_1)$ and $N(d_2)$ using standard normal distribution tables:

$$N(0.727) \approx 0.7673$$

$$N(0.527) \approx 0.7014$$

Finally we plug them back into the Black-Scholes Formula

$$C = 100e^{-0.03(1)}(0.7673) - 90e^{-0.05(1)}(0.7014) = 100(0.9704)(0.7673) - 90(0.9512)(0.7014)$$

$$C = 74.43 - 60.91 \approx 12.03$$

The intrinsic value is:

$$S - K = 100 - 90 = 10$$

Here, the European option is worth 12.03, which is greater than the intrinsic value because there is still time value. Then we modify the example:

- Stock price: $S = 100$
- Strike price: $K = 110$ (out of the money)
- Time to expiration: 1 month
- Dividend yield: 3%
- Volatility: 10%

Using Black-Scholes and similar calculation as previous step:

$$C_{\text{European}} \approx 0.25$$

But the intrinsic value is:

$$\max(100 - 110, 0) = 0$$

In this case, the European option is worth less than its intrinsic value, because there is no early exercise opportunity, and the option is out of the money.

Question 5

Explain the European call-put parity argument. Why it can not be used for American options

Answer

The Put-Call Parity is a fundamental relationship between the prices of European call and put options on the same underlying asset, with the same strike price and expiration date. It defines a relationship between the price of a European call option and European put option, both with the identical strike price and expiry, namely that a portfolio of a long call option and a short put option is equivalent to (and hence has the same value as) a single forward contract at this strike price and expiry

The formula is:

$$C - P = Se^{-qt} - Ke^{-rt}$$

- C : Price of the European call option
- P : Price of the European put option
- S : Current stock price
- K : Strike price
- r : Risk-free interest rate
- q : Continuous dividend yield
- t : Time to expiration
- e^{-qt} : Discounted stock price due to dividend payments
- e^{-rt} : Discounted strike price due to risk-free rate

The parity works well for European options because they can only be exercised at expiration. The key idea behind this formula is that a portfolio consisting of a long European call option and a short European put option is equivalent to holding the underlying stock while borrowing the present value of the strike price.

Both portfolios will yield the same payoff at expiration, which is why they must have the same value before expiration (by the no-arbitrage principle).

However, the Put-Call Parity does not directly apply to American options because:

- American options can be exercised at any time before expiration, which adds an additional layer of complexity. This early exercise possibility makes the payoff structure of American options different from European options.
- For American call options on dividend-paying stocks, early exercise is often optimal just before the ex-dividend date (to avoid losing dividends). This means the American call option might be more valuable than the European call option.
- Unlike European options, American calls and puts do not have a fixed parity relationship because the option holder's decision to exercise early can affect the option's price.
- The strict no-arbitrage condition that holds for European options does not necessarily apply to American options due to the early exercise feature. This creates pricing discrepancies that do not align with put-call parity.

A Numerical Example

Suppose we take stock price $S = 100$, strike price $K = 95$, time to expiration as 6 months, risk-free rate as 5%, dividend yield as 2%, European call price 8 and European put price to be 3.

$$8 - 3 = 100e^{-0.02(0.5)} - 95e^{-0.05(0.5)}$$

$$5 = 100(0.9900) - 95(0.9753)$$

$$5 = 99 - 92.65$$

$$5 = 6.35$$

There is a small difference due to rounding, but the parity approximately holds.

Question 6

Calculate the implied volatility of Microsoft stock using March 2025 calls expiring March 21, 2025 with strike 410 and with strike 420. Get the quotes from any data provider, for example, finance.yahoo.com and use all other necessary data. On Bloomberg type MSFT Equity CALL (page down as needed).

Answer

I used python to calculate the implied volatility. Currently the risk-free rate is 4.24% as I got from the source. And the current market price for strike 410 is \$3.04 and 420 \$1.21 separately. Then use the Black-Scholes model to recalculate the implied volatility:

```
1 import numpy as np
2 from scipy.stats import norm
3 from scipy.optimize import brentq
4
5 # Black-Scholes formula for European Call Option
6 def black_scholes_call(S, K, t, r, sigma):
7     d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * t) / (sigma * np.sqrt(t))
8     d2 = d1 - sigma * np.sqrt(t)
9     call_price = S * norm.cdf(d1) - K * np.exp(-r * t) * norm.cdf(d2)
10    return call_price
11
12 # Function to compute Implied Volatility
13 def implied_volatility_call(S, K, t, r, market_price):
14     # Brent's method to solve for implied volatility (IV)
15     result = brentq(lambda x: black_scholes_call(S, K, t, r, x) - market_price, 1e-6, 5)
16     return result
17
18 # Example: Calculating IV for Microsoft (MSFT)
19 S = 394.9 # Current Stock Price (MSFT)
20 K_410 = 410 # Strike Price 410
21 K_420 = 420 # Strike Price 420
22 t = 0.05 # Time to Expiration in years (March 21, 2025)
23 r = 0.0424 # Risk-free rate (4.24%)
24 market_price_410 = 3.04
25 market_price_420 = 1.21
26
27 # Compute Implied Volatility for both options
28 iv_410 = implied_volatility_call(S, K_410, t, r, market_price_410)
29 iv_420 = implied_volatility_call(S, K_420, t, r, market_price_420)
30
31 # Display the results
32 print(f"Implied Volatility for strike 410: {iv_410*100:.2f}%")
33 print(f"Implied Volatility for strike 420: {iv_420*100:.2f}%)
```

And the printout is:

Implied Volatility for strike 410: 23.10%

Implied Volatility for strike 420: 22.61%

This can be reconfirmed with the data from Yahoo Finance:

Figure 9: Strike Price 410, Expire Date 2025-03-21

Calls										
Contract Name	Last Trade Date (EST)	Expire Date	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
MSFT250307C00410000	3/3/2025 12:06 PM	2025-03-07	0.47	0.47	0.48	-0.40	-45.98%	4,358	6,556	23.98%
MSFT250314C00410000	3/3/2025 12:05 PM	2025-03-14	1.77	1.75	1.79	-1.23	-41.00%	927	2,513	23.47%
MSFT250321C00410000	3/3/2025 12:01 PM	2025-03-21	3.02	2.98	3.05	-0.78	-20.53%	590	7,464	23.33%
MSFT250328C00410000	3/3/2025 11:54 AM	2025-03-28	4.17	4.05	4.15	-1.04	-19.96%	243	860	23.12%
MSFT250404C00410000	3/3/2025 11:54 AM	2025-04-04	5.32	5.15	5.30	-1.20	-18.40%	91	423	23.32%

Figure 10: Strike Price 420, Expire Date 2025-03-21

Calls										
Contract Name	Last Trade Date (EST)	Expire Date	Last Price	Bid	Ask	Change	% Change	Volume	Open Interest	Implied Volatility
MSFT250307C00420000	3/3/2025 12:00 PM	2025-03-07	0.11	0.10	0.11	-0.08	-44.44%	2,693	9,199	26.17%
MSFT250314C00420000	3/3/2025 12:03 PM	2025-03-14	0.58	0.54	0.56	-0.30	-34.09%	458	2,513	23.15%
MSFT250321C00420000	3/3/2025 12:04 PM	2025-03-21	1.21	1.18	1.22	-0.39	-24.38%	1,305	23,553	22.57%
MSFT250328C00420000	3/3/2025 12:00 PM	2025-03-28	1.94	1.86	1.94	-0.52	-21.14%	286	2,729	22.33%
MSFT250404C00420000	3/3/2025 12:01 PM	2025-04-04	2.73	2.67	2.74	-0.80	-22.66%	79	547	22.42%

Question 7

Download Excel options model with VBA code from the courseworks . Save as a source. Open in Excel. Modify the Black model for futures to Black-Scholes model for stocks paying dividends at rate q (like in the Hull's book). Make the necessary changes in Visual Basic code. Use Excel help or consult TA's if you do not know what to do. Check that code works and submit the code printout.

Answer

Figure 11: The Printout of the B-S Model Re-implemented in Excel

A	B	C	D	E	F	G	H	I	J	K	L	M	
1													
2													
3	Black-Scholes Model												
4	Future Price	Strike Price	Time To Expiry	Volatility	Risk Free Rate	Dividend	Call/Put	Option Price	Delta	Gamma	Theta	Vega	Rho
5	125.000000	242.000000	4.000000	0.360000	0.070000	0.034000	c	13.59616035	0.360346294	0.004158241	-0.006602331	0.935604123	1.028781626
6	125.000000	242.000000	4.000000	0.360000	0.070000	0.034000	p	87.39049672	-0.639653706	0.004158241	-0.041678979	0.935604123	-6.287204991
7													
8													
9													
10													
11													
12													
13													

Question 8

Download Excel Brownian Motion model from the courseworks. Save as a source. Open in Excel. Modify it to Geometric Brownian motion with growth rate $\mu = 0.04$, volatility $\sigma = 0.20$ and 250 trajectories. Submit excel formulas printout. Geometric Brownian motion starts with positive X_0 so you must change a starting value from 0 to a positive number that you may choose (you can choose 1, 100 or other positive number).

Answer

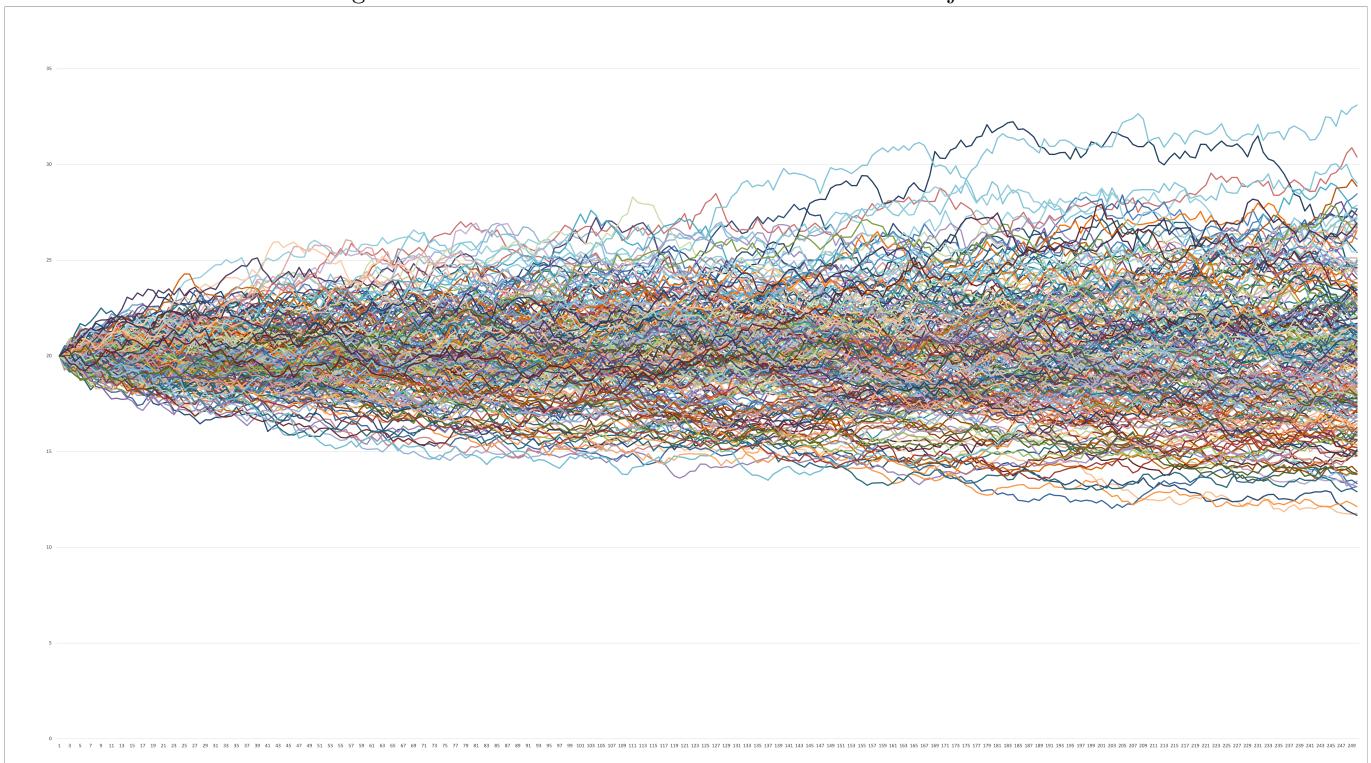
The number for X_0 I chose was 20, I firstly put several parameters in the A column (for time step, it is 250).

Figure 12: Settings in my Excel

A	B	C	D	E	F	G
1	Growth Rate	Time	1	2	3	4
2		0.04 Trac 1	20	19.82077317	19.54878179	19.43056657
3	Volatility	Trac 2	20	19.65231767	19.9391074	20.203435
4		0.2 Trac 3	20	20.14019724	19.95438317	19.9537862
5	Initial Value	Trac 4	20	19.92875626	19.8819482	19.5855312
6		20 Trac 5	20	19.88932391	20.11611406	19.41433972
7	Num of Trajectories	Trac 6	20	20.38122799	20.34695327	20.71934936
8		250 Trac 7	20	20.08282482	19.96230487	19.99068615
9	Time Steps	Trac 8	20	20.11316508	20.37823955	20.01866757
10		250 Trac 9	20	19.87982115	19.70980455	19.64561904
11		Trac 10	20	20.14935925	20.04433286	19.50385732
12		Trac 11	20	20.10647884	20.30952821	20.26797587
13		Trac 12	20	20.11397681	20.43318372	20.64539772
14		Trac 13	20	20.0551359	20.15186673	19.8341527
15		Trac 14	20	19.653336	19.69143688	19.35136806
16		Trac 15	20	19.8889085	19.64843999	19.58420851
17		Trac 16	20	19.85366181	19.6011544	19.48604228
18		Trac 17	20	20.07770051	20.30712651	20.54204803
19		Trac 18	20	19.87541073	19.77064086	19.52381052
20		Trac 19	20	19.86821155	19.95924734	20.28252577
21		Trac 20	20	20.1672283	20.1779516	20.54243567
22		Trac 21	20	20.28688007	20.14439704	19.73582509

In D2, I set the formula to be $=C2*EXP((\$A\$2-0.5*\$A\$4^2)*(1/250)+\$A\$4*SQRT(1/250)*NORMSINV(RAND()))$. The final printout is:

Figure 13: Geometric Brownian Motion with 250 Trajectories



Out of interest, I redid it in R:

```
1 X0 <- 20
2 mu <- 0.04
3 sigma <- 0.20
4 n <- 250
5 T <- 1
6 N <- 250
7 dt <- T / N
8
9 # Generate time vector
10 time <- seq(0, T, by = dt)
11
12 # Initialize matrix to store trajectories
13 X <- matrix(NA, nrow = N + 1, ncol = n)
14 X[1, ] <- X0
15
16 # Simulate GBM
17 set.seed(123) # For reproducibility
18 for (i in 1:n) {
19   for (j in 2:(N + 1)) {
20     dW <- rnorm(1, mean = 0, sd = sqrt(dt))
21     X[j, i] <- X[j - 1, i] * exp((mu - 0.5 * sigma^2) * dt + sigma * dW)
22   }
23 }
24
25 # Convert to data frame for plotting
26 X_df <- as.data.frame(X)
27 X_df$time <- time
28 X_long <- melt(X_df, id.vars = "time", variable.name = "trajectory", value.name = "price")
29
30 # Plot the trajectories
31 ggplot(X_long, aes(x = time, y = price, group = trajectory)) +
32   geom_line(alpha = 0.5) +
33   labs(
34     x = "Time",
35     y = "Price") +
36   theme_minimal()
```

One interesting thing I noticed is that the set of trajectories tend to move upward, that's because there exists a drift term μ , which is typically positive in financial implications (and also in this case). Thus we can notice as time goes on, the trajectories get out of 30 in both plots.

Figure 14: Geometric Brownian Motion with 250 Trajectories Generated in R

