J. P. Morgan Quant Mentorship Program 2022

Shruthi Rao

B.Tech (Batch 2024) r.shruthi@iitg.ac.in IIT Guwahati

Electronics and Communication Engineering (Major)
Computer Science & Engineering (Minor)

Contents

1 Case Study A: Derivatives

1.1 Question 1: Continuous Compounding

The concept of compound interest is that interest is added back to the principal sum so that interest is gained on that already-accumulated interest during the next compounding period.
The formula for compound interest is:

$$A = P(1 + r/n)^{nt}$$

Where,
A = the future value of the investment/loan, including interest
P = the principal investment amount (the initial deposit or loan amount)
r = the annual interest rate (decimal)
n = the number of times that interest is compounded per unit t
t = the time the money is invested or borrowed for

We are given the amount \$10,1000, at an interest rate of 5% per annum.
We can say that,
P = \$10,000 and
$r = 5/100 = 0.05$(decimal)
n will be the number of times principle is compounded per year.

1.1.1 Part A

We need to find the Amount if the principle was compounded semi-annually, i.e. every 6 months, after 10 years. The principle is compounded every 6 months, therefore,
$n = 2$ (it is compounded twice a year)
$t = 10$
Calculating the Amount,

$$\begin{aligned} A &= \$10,000(1 + \frac{0.05}{2})^{2*10} \\ &= \$16,386.16 \end{aligned}$$

1.1.2 Part B

If the invested amount was compounded weekly, the value of n changes.
The number of weeks in a year = 52. Therefore, $n = 52$
$t = 10$(it remains the same, i.e. 10 years)

$$\begin{aligned} A &= \$10,000(1 + \frac{0.05}{52})^{52*10} \\ &= \$16,483.25 \end{aligned}$$

1.1.3 Part C

If the invested amount was compounded daily, the value of n changes.
The number of days in a year = 365. Therefore, $n = 365$
$t = 10$(it remains the same, i.e. 10 years)

$$\begin{aligned} A &= \$10,000(1 + \frac{0.05}{365})^{365*10} \\ &= \$16,486.648 \end{aligned}$$

1.1.4 Part D

If the invested amount was compounded continuously, the value of n changes.
Frequency of compounding approaches infinity when it is compounded continuously.
Therefore, value of n approaches infinity. $t = 10$(it remains the same, i.e. 10 years)
Hint given:

$$\lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = e^x$$

Using the Hint, We can conclude that:

$$\lim_{n\to\infty}\left(1 + \frac{r}{n}\right)^{nt} = e^{rt}$$

If the principle is compounded continuously, we can rewrite the formula as:

$$\begin{aligned} A &= \lim_{n\to\infty} P\left(1+\frac{r}{n}\right)^{nt} \\ &= P\lim_{n\to\infty}\left(1+\frac{r}{n}\right)^{nt} \\ &= Pe^{rt} \end{aligned}$$

Where,
A = Amount or final value of the investment including the interest
P = Principle or the amount invested initially
r = the annual interest rate (decimal)
t = the time the money is invested or borrowed for

Plugging in the values from part A:

$$\begin{aligned} A &= Pe^{rt} \\ &= \$10,100e^{0.05*10} \\ &= \$16,487.21 \end{aligned}$$

1.2 Question 2: Call/Put Option Pricing

1.2.1 Part A

For a stock currently trading at \$S,
We assume that for the call option, strike is \$K, time of expiry is T years and r is the annual "risk-free" interest rate.

Using the result we obtained in part D of question 1, we can think of the strike as the amount in the future, compounded continuously after time 'T' with an annual rate T. Let's find the principle if this were the case, i.e. the present value of the strike price:

$$A = Pe^{rt}$$
$$K = Pe^{rT}$$
$$P = \frac{K}{e^{rT}}$$

In call option, we bet on the price of the stock going up, hence the amount we'd be willing to pay today is nothing but the difference between current trading value and present value of strike price, which is nothing but S-P since we want to minimize our loss and maximise the profit.
Therefore, the amount we would be willing to pay today, i.e. the Call Option Price today is:

$$\text{Call Price today} = Max\{(S-P), 0\}$$
$$C_0 = Max\{(S - \frac{K}{e^{rT}}), 0\}$$
$$C_0 = Max\{(S - Ke^{-rT}), 0\}$$

We can also look at this in the following manner, at any time t, between present$(t = 0)$ and maturity time T, let the price of stock at that time be S_t, then the Call Price at time t can be found as follows:
We want the price of the stock to be greater than the strike price at

that moment, to ensure profit, therefore:

$$\begin{aligned} S_t &\geq Ke^{-r(T-t)} \\ 0 &\leq S_t - Ke^{-r(T-t)} \\ C_t &= Max\{(S_t - Ke^{-r(T-t))}, 0\} \end{aligned}$$

1.2.2 Part B

For the same parameters in Part A, we repeat the analysis for a put option instead of a call option. In a call option, we bet on the price of a stock going up whereas for a put option we bet on the price going down.
The analysis for put is exactly the opposite of that of call option. Therefore, the maximum amount we'd be willing to pay today, i.e Put price is:

$$\begin{aligned} \text{Put option price today} &= Max\{(P - S), 0\} \\ P_0 &= Max\{(\frac{K}{e^{rT}} - S), 0\} \\ P_0 &= Max\{(Ke^{-rT} - S), 0\} \end{aligned}$$

Using the same analysis like part A, price of put option at any time will be given by:

$$P_t = Max\{(Ke^{-r(T-t)} - S_t), 0\}$$

1.3 Question 3: BSM Pricing

In Question 2, we have considered 'S' as the present price of the stock, so we may call it S_0 and we define S_t as the value or selling price of stock at time 't'.
In the Black Scholes Merton model, we also consider volatality of the stock's price as a factor into the pricing of call/put options.
We can write the price of the call option as:

$$
\begin{aligned}
C\left(S_t, t\right) &= N\left(d_1\right) S_t - N\left(d_2\right) K e^{-r(T-t)} \\
d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\
d_2 &= d_1 - \sigma\sqrt{T-t} \\
P\left(S_t, t\right) &= K e^{-r(T-t)} - S_t + C\left(S_t, t\right) \\
&= N\left(-d_2\right) K e^{-r(T-t)} - N\left(-d_1\right) S_t
\end{aligned}
$$

where,
• $C\left(S_t, t\right)$ - BSM price of a call option at time t and stock price S_t (in S)
• $P\left(S_e, t\right)$ - BSM price of a put option at time t and stock price S_c (in S)
• S_t - Stock price at time t (in S)
• σ - Stock volatility (absolute, not percent)
• K - Option strike price (in S)
• T-Time to maturity (in years)
• N - CDF of the standard normal distribution
• r-annual risk frec rate of interest (absolute, not percent)

1.3.1 Part A

We want to find the call and put pricing when t tends to T, i.e. when we are very close to expiry.
For **call option** at maturity:
Case 1: When $S_T >$ K, i.e, if the stock price at maturity is greater than the pre-defined strike price,

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

As t $\rightarrow$ T, the second term inside the bracket becomes 0 since the degree of numerator is greater than the denominator in the second term.
As $S_T >$ K, the first term tends to infinity due to the denominator, and hence we conclude that as t $\rightarrow$ T, $d_1 \rightarrow +\infty$.

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

In d_2, the second term reduces to 0 as t $\rightarrow$ T, so we can conclude that $d_2 \rightarrow +\infty$.
We are given that N is the CDF of the standard normal distribution and the formula for call price is:

$$C(S_t, t) = N(d_1)S_t - N(d_2)Ke^{-r(T-t)}$$

In the above equation, the term $e^{-r(T-t)}$ reduces to 1 as t $\rightarrow$ T and the terms as N is the CDF of the standard normal distribution function whose variables in this case tend to infinity,
$N_1 \rightarrow 1$, because $d_1 \rightarrow +\infty$ and $N_2 \rightarrow 1$ because $d_2 \rightarrow +\infty$, using the property of cumulative distribution function:

$$\lim_{x\rightarrow+\infty} F_X(x) = 1$$

where $F_x(x)$ represents the CDF of the variable x.
Hence, the price of call option at the time of maturity reduces to:

$$C(S_T, T) = S_T - K$$

Case 2: When $S_T <$ K, i.e, if the stock price at maturity is less than the pre-defined strike price, the second term inside the bracket is 0, same as the previous case but the first part is negative and hence, in this case, $d_1 \to -\infty$. Just like e proved in case 1, we can prove that in this case, $d_2 \to -\infty$. Using the property of CDF

$$\lim_{x \to -\infty} F_X(x) = 0$$

We can conclude that when $S_T <$ K, the price of call option is 0.

$$C(S_T, T) = 0$$

Combining both the cases, we may write that:

$$C(S_T, T) = Max\{(S_T - K), 0\}$$

For **put option** pricing, we are given that:

$$P(S_t, t) = Ke^{-r(T-t)} - S_t + C(S_t, t)$$

Case 1: When $S_T >$ K, i.e, if the stock price at maturity is greater than the pre-defined strike price, $C(S_T, T) = S_T - K$, so the put price according to the equation when t $\to$ T

$$\begin{aligned} P(S_T, T) &= K - S_T + S_T - K \\ &= 0 \end{aligned}$$

Case 2: When $S_T <$ K, i.e, if the stock price at maturity is less than the pre-defined strike price, $C(S_T, T) = 0$, so the put price according to the equation when t $\to$ T

$$\begin{aligned} P(S_T, T) &= K - S_T + 0 \\ &= K - S_T \end{aligned}$$

Combining both the cases, we may write that:

$$P(S_T, T) = Max\{(K - S_T), 0\}$$

From the conclusion we arrived at in Question 2, we may have written the call option and put option price as:

$$\begin{aligned} C_T &= Max\{(S_T - K), 0\} \\ P_T &= Max\{(K - S_T), 0\} \end{aligned}$$

We see that these formulae agree with the formulae we derived in the previous question, without inculcating σ and we may hence conclude that σ won't be able to affect call/put option prices by much as we get close to expiry.

1.3.2 Part B

We are asked to plot the BSM price of a call option C, with stock price S_t as the x-axis with the given parameters:
Range of S_t = \$1 to \$100
K = \$50
r = 0.12p.a.
σ = 0.30
T = 1 year
I have plotted the graph using matplotlib(for both part b and c), which takes t as the input and shows the graph. The code has been attached with the file.

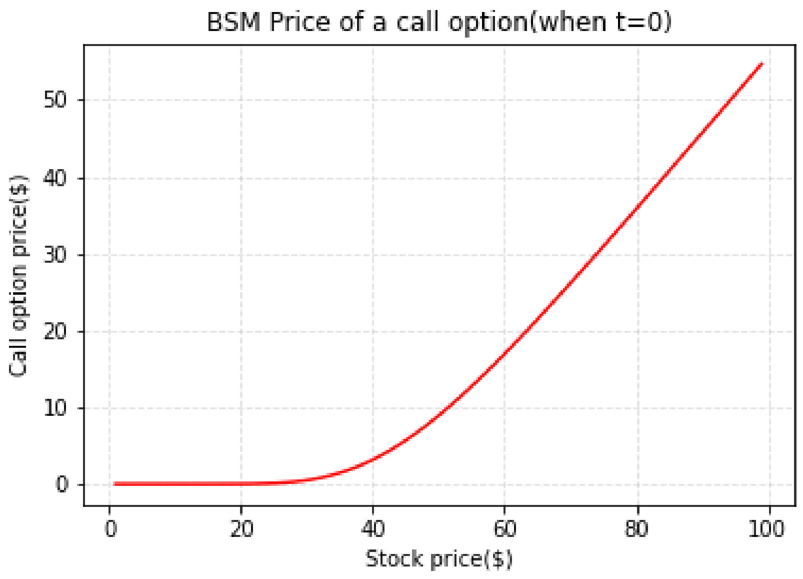


Figure 1: **Case1**: Variation of Call Option when t=0

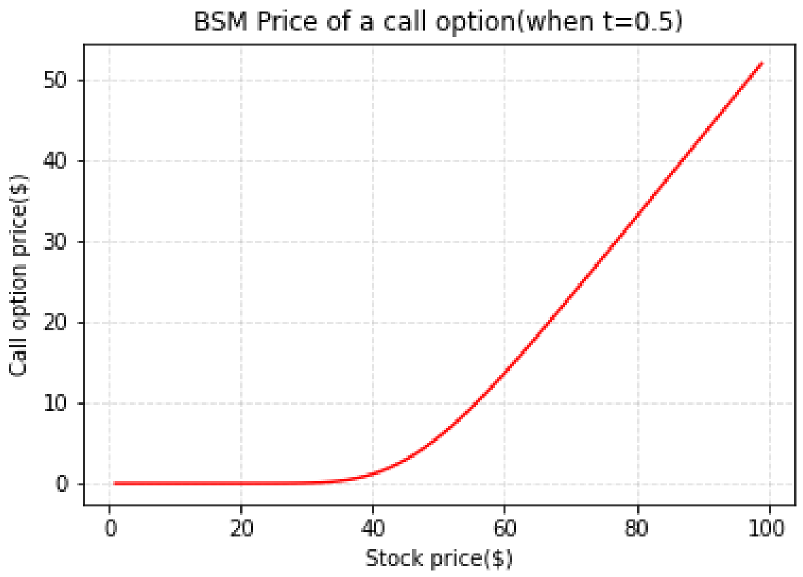


Figure 2: **Case2**: Variation of Call Option when t=0.5

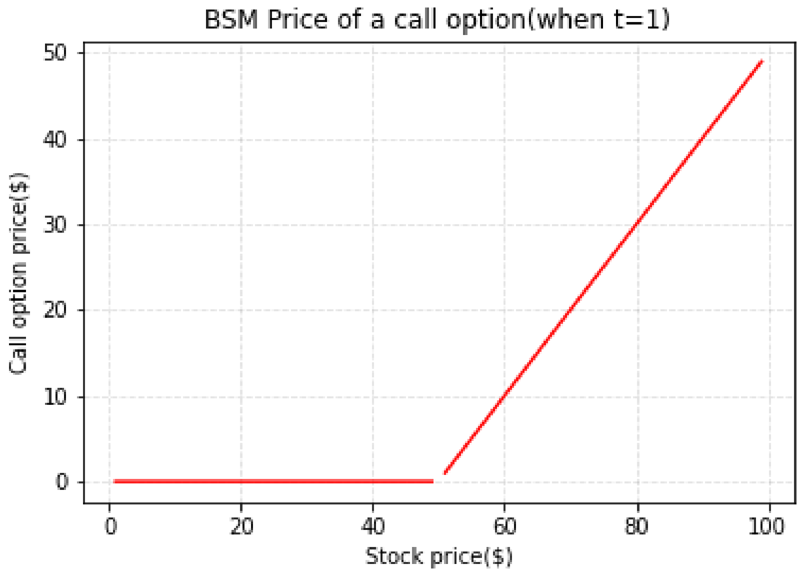


Figure 3: **Case3**: Variation of Call Option when t=1

1.3.3 Part C

We need to plot the graphs for the same parameters but for put option.
The function for call and put Option is not defined when S_t = K in case 3 for both parts when $t = T$, hence it breaks at S_t = 50\$.

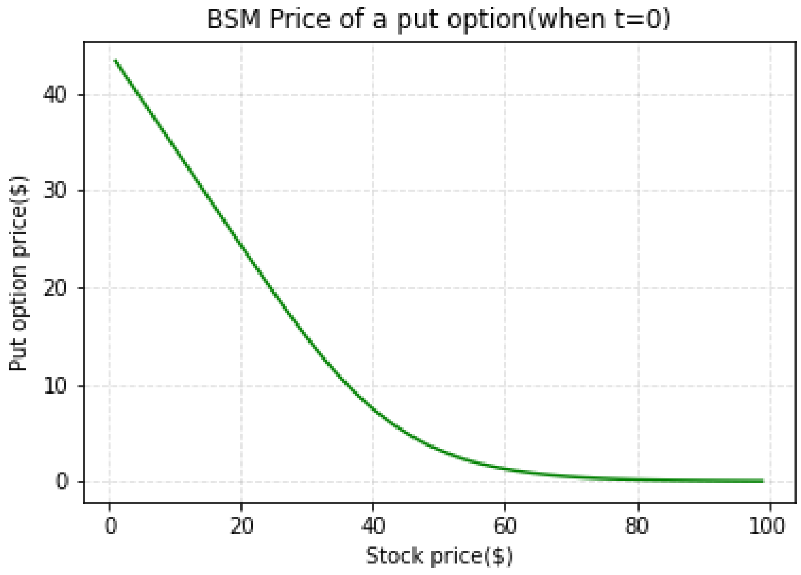


Figure 4: **Case1**: Variation of Put Option when t=0

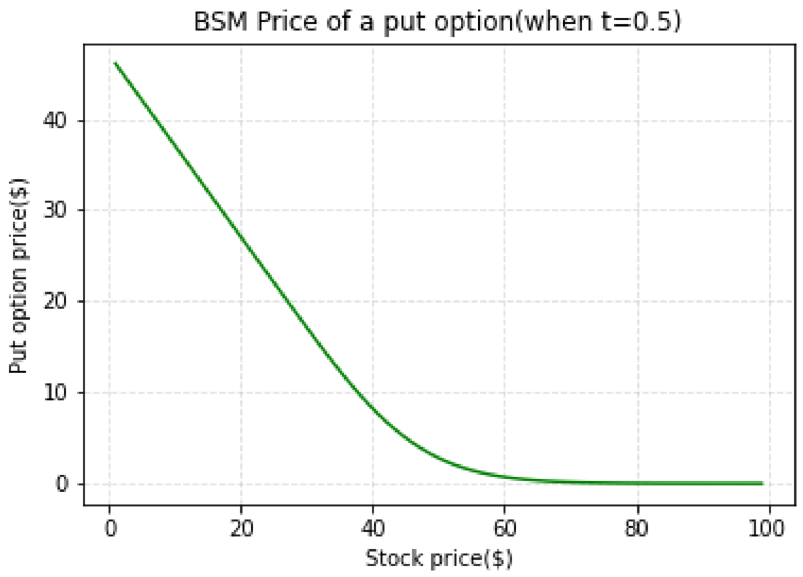


Figure 5: **Case2**: Variation of Put Option when t=0.5

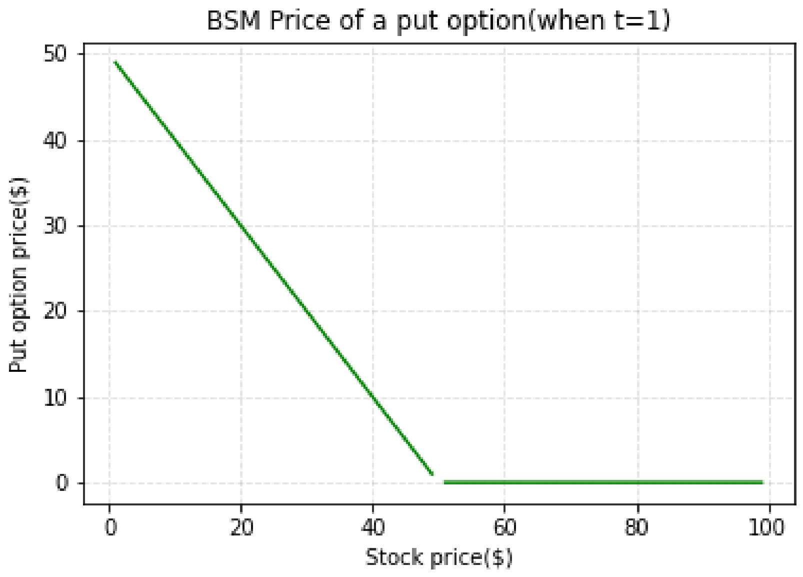


Figure 6: **Case3**: Variation of Put Option when t=1

Description of the code file:

```
#Library for plotting
import matplotlib.pyplot as plt
#Library for mathematical calculations
import numpy as np
#Library used for calculating CDF
import scipy as sp
#For standard normal variable
from scipy.stats import norm
    #given values are allotted to the variable
    sigma = 0.3 #given values are allotted to the variable
    T = 1
    r = 0.12
    K = 50
    #Value of t is given as the input
    t = float(input())
    #Equations are defined as variables as given in the question
    St = np.array(range(1,100)) #
    d1 = (np.log(St/K) + (r +
         (sigma*sigma)/2)*(T-t))/(sigma*np.sqrt(T-t))
    d2 = d1 - sigma*np.sqrt(T-t)
    N_d1 = sp.stats.norm.cdf(d1)
    N_d2 = sp.stats.norm.cdf(d2)
    C = N_d1*St - N_d2*K*np.exp(-r*(T-t))
    P = K*np.exp(-r*(T-t)) - St + C
    #Creating the plot
    plt.plot(St,P, color = "red")
    #Adding a title
    plt.title('BSM Price of a call option(when t=0)')
    #Give labels to x and y axes
    plt.xlabel('Stock price($)')
    plt.ylabel('Call option price($)')
    #create a grid
    plt.grid(alpha=.4,linestyle='--')
    #show the plot
    plt.show()
```

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "*pythonSHRUTHI_RAO_A_Q3_B_MAIN.py*" to run the program for part B.
- Run Command "*pythonSHRUTHI_RAO_A_Q3_C_MAIN.py*"

to run the program for part C.

- Type in the value of t(0 or 0.5 or 1)as needed.

1.4 Question 4: Delta Calculation

Delta is defined as the rate of change of the option price with respect to the price of the underlying asset. It is the slope of the curve that relates the option price to the underlying asset price.

1.4.1 Part A

We need to analytically calculate the delta of call and put options in terms of t, S_t, T, σ, K with the BSM formulae given in the previous question.
Delta for Call option is nothing but $\frac{\partial C(S_t,t)}{\partial S}$

$$\begin{aligned}
C(S_t, t) &= N(d_1)S_t - N(d_2)Ke^{-r(T-t)} \\
d_1 &= \frac{1}{\sigma\sqrt{T-t}}[\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)] \\
d_2 &= d_1 - \sigma\sqrt{T-t} \\
C(S_t, t) &= N(d_1)S_t - N(d_1 - \sigma\sqrt{T-t})Ke^{-r(T-t)} \\
\frac{\partial C(S_t, t)}{\partial S} &= \frac{\partial}{\partial S}(N(d_1)S_t - N(d_1 - \sigma\sqrt{T-t})Ke^{-r(T-t)}) \\
&= \frac{\partial}{\partial S}(SN(d_1)) - \frac{\partial}{\partial S}(Ke^{-r(T-t)}N(d_1 - \sigma\sqrt{T-t})) \\
&= (N(d_1)\frac{\partial}{\partial S}S + S\frac{\partial}{\partial S}N(d_1)) - Ke^{-r(T-t)}\frac{\partial}{\partial S}(N(d_1 - \sigma\sqrt{T-t})) \\
&= (N(d_1) + Sn(d_1)\frac{\partial}{\partial S}(d_1)) - Ke^{-r(T-t)}n(d_1 - \sigma\sqrt{T-t})\frac{\partial}{\partial S}(d_1 - \sigma\sqrt{T-t}) \\
&= N(d_1) + Sn(d_1)\frac{\partial(d_1)}{\partial S} - Ke^{-r(T-t)}n(d_1 - \sigma\sqrt{T-t})\frac{\partial(d_1)}{\partial S}
\end{aligned}$$

where n(x) represents the probability distribution of standard normal variable.
Using the following formula, we try to simplify $n(d_1 - \sigma\sqrt{T-t})$

$$\begin{aligned}
n(x) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \\
n(d_1 - \sigma\sqrt{T-t}) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{(d_1-\sigma\sqrt{T-t})^2}{2}} \\
&= \frac{1}{\sqrt{2\pi}} e^{-\frac{d_1^2}{2} - \frac{\sigma^2(T-t)}{2} + d_1\sigma\sqrt{T-t}} \\
&= n(d_1) e^{-\frac{\sigma^2(T-t)}{2} + [\frac{1}{\sigma\sqrt{T-t}}\{\ln(\frac{S_t}{K}) + (r+\frac{\sigma^2}{2})(T-t)\}]\sigma\sqrt{T-t}} \\
&= n(d_1) e^{\ln(\frac{S_t}{K}) + r(T-t)} \\
&= n(d_1) \frac{S_t}{K} e^{r(T-t)}
\end{aligned}$$

Substituting this is the equation obtained earlier, all the terms except $N(d_1)$ get cancelled and we obtain:

$$\frac{\partial C(S_t,t)}{\partial S} = N(d_1)$$

Calculating Delta for Put Option: We know that price of put option is given by:

$$P(S_t,t) = Ke^{-r(T-t)} - S_t + C(S_t,t)$$

Delta for Call option is nothing but $\frac{\partial P(S_t,t)}{\partial S}$

$$\begin{aligned}
\frac{\partial P(S_t,t)}{\partial S} &= \frac{\partial (Ke^{-r(T-t)} - S_t + C(S_t,t))}{\partial S_t} \\
&= -\frac{\partial S_t}{\partial S_t} + \frac{\partial C(S_t,t)}{\partial S_t} \\
&= -1 + N(d_1)
\end{aligned}$$

Summarizing the above equations, we have:

$$\begin{aligned}
\text{delta}_{Calloption} &= N(d_1) \\
\text{delta}_{Putoption} &= N(d_1) - 1
\end{aligned}$$

We want delta in terms of t, S_t, K and r, substituting the value of d_1, we have:

$$\text{delta}_{Calloption} = N(\frac{1}{\sigma\sqrt{T-t}}[\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)])$$
$$\text{delta}_{Putoption} = N(\frac{1}{\sigma\sqrt{T-t}}[\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(T-t)]) - 1$$

Where N(x) represents the CDF of a standard normal distribution. We want to calculate the deltas for the six scenarios in the previous question. The parameters are:
S_t = \$125
K = \$50
r = 0.12p.a.
σ = 0.30
T = 1 year
I have solved this question programatically, where the value of delta for both Call and Put Option is printed out when t is given as the input.
When $t = 0$:
$delta_{calloption}$ = 0.9998435037893203

$delta_{putoption}$ = -0.00015649621067970187
When $t = 0.5$:
$delta_{calloption}$ = 0.9999987513171391

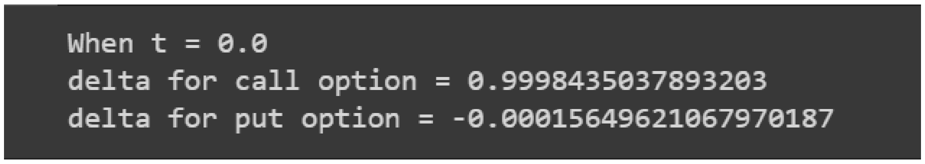
$delta_{putoption}$ = -1.248682860888195x10^{-06}
When $t = 1$:
$delta_{calloption}$ = 1

$delta_{putoption}$ = 0

Description of the code file:

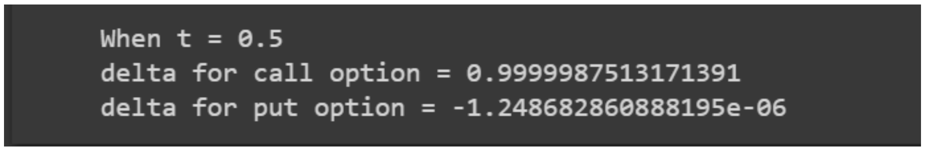
```
#Library for mathematical calculations
import numpy as np
#Library used for calculating CDF
import scipy as sp
#For standard normal variable
from scipy.stats import norm
    #Alloting values according to the question
    sigma = 0.3
    T = 1
    r = 0.12
    K = 50
    #t is given as the input
    t = float(input())
    St = 125
    d1 = (np.log(St/K) + (r +
         (sigma*sigma)/2)*(T-t))/(sigma*np.sqrt(T-t))
    d2 = d1 - sigma*np.sqrt(T-t)
    N_d1 = sp.stats.norm.cdf(d1)
    N_d2 = sp.stats.norm.cdf(d2)
    C = N_d1*St - N_d2*K*np.exp(-r*(T-t))
    P = K*np.exp(-r*(T-t)) - St + C
    delta_C = N_d1
    delta_P = N_d1 - 1
    #Printing values for corresponding t
    print('When t =', t)
    print('delta for call option =', delta_C)
    print('delta for put option =', delta_P)
```

Outputs of the files:



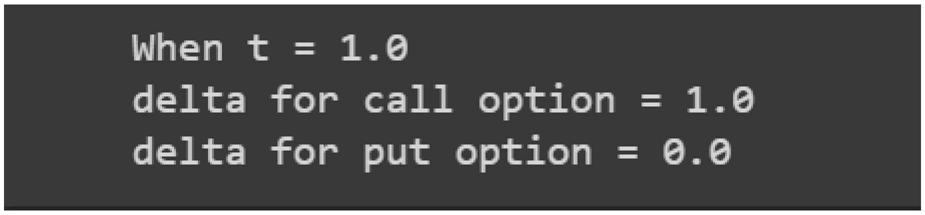
```
When t = 0.0
delta for call option = 0.9998435037893203
delta for put option = -0.00015649621067970187
```

Figure 7: **Case1**: Delta when t=0



```
When t = 0.5
delta for call option = 0.9999987513171391
delta for put option = -1.248682860888195e-06
```

Figure 8: **Case2**: Delta when t=0.5



```
When t = 1.0
delta for call option = 1.0
delta for put option = 0.0
```

Figure 9: **Case3**: Delta when t=1

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "python SHRUTHI_RAO_A_Q4_A_MAIN.py" to run the program.
- Type in the value of t(0 or 0.5 or 1) for which you want to calculate the delta.

1.4.2 Part B

We want to calculate delta as the slope of the plots we made in Question 3 when $S_t = \$125$.
I have solved this question programmatically, which takes t as the input and gives delta for both call and put options as the outputs.
The output is as follows:
When $t = 0$:
$delta_{calloption}$ = 0.9998432248412712

$delta_{putoption}$ = -0.00015677515872880576
When $t = 0.5$:
$delta_{calloption}$ = 0.9999987441614167

$delta_{putoption}$ = -1.255838583347213x10^{-06}
When $t = 1$:
$delta_{calloption}$ = 1

$delta_{putoption}$ = 0
Outputs of the files:

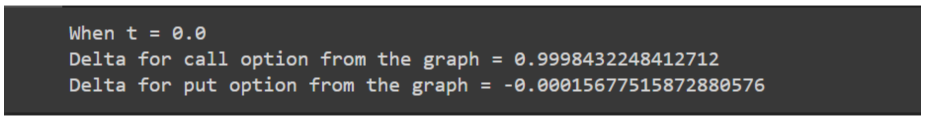

```
    When t = 0.0
    Delta for call option from the graph = 0.9998432248412712
    Delta for put option from the graph = -0.00015677515872880576
```

Figure 10: **Case1**: Delta from slope when t=0

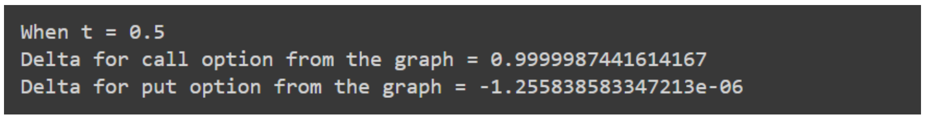

```
When t = 0.5
Delta for call option from the graph = 0.9999987441614167
Delta for put option from the graph = -1.255838583347213e-06
```

Figure 11: **Case2**: Delta from slope when t=0.5

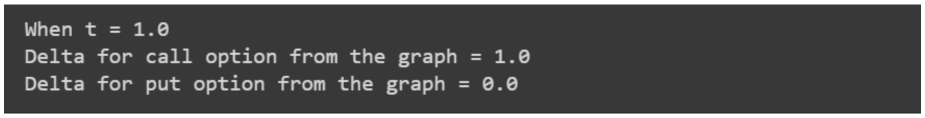

```
When t = 1.0
Delta for call option from the graph = 1.0
Delta for put option from the graph = 0.0
```

Figure 12: **Case3**: Delta from slope when t=1

Description of the code file:

```
#Library for mathematical calculations
import numpy as np
#Library used for calculating CDF
import scipy as sp
#For standard normal variable
from scipy.stats import norm
    #Alloting values according to the question
    sigma = 0.3
    T = 1
    r = 0.12
    K = 50
    #t is given as the input
    t = float(input())
    St = np.array(range(1,150))
    d1 = (np.log(St/K) + (r +
         (sigma*sigma)/2)*(T-t))/(sigma*np.sqrt(T-t))
    d2 = d1 - sigma*np.sqrt(T-t)
    N_d1 = sp.stats.norm.cdf(d1)
    N_d2 = sp.stats.norm.cdf(d2)
    C = N_d1*St - N_d2*K*np.exp(-r*(T-t))
    P = K*np.exp(-r*(T-t)) - St + C
    #Printing values for corresponding t.
    #We want the value at St = 125$, which is at the 124th position in
         the array
    print('When t =', t)
    print('Delta for call option from the graph =', np.gradient(C)[124])
    print('Delta for put option from the graph =', np.gradient(P)[124])
```

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "python SHRUTHI_RAO_A_Q4_B_MAIN.py" to run the program.
- Type in the value of t(0 or 0.5 or 1) for which you want to calculate the delta.

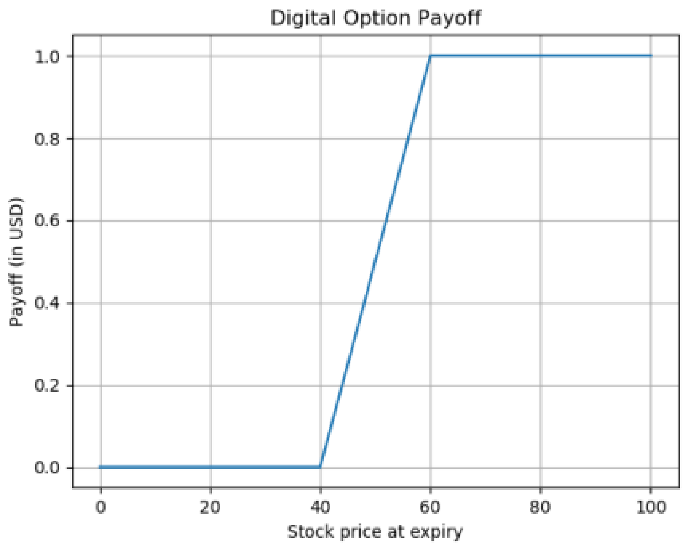
Conclusion: We see that the result of Part A and Part B are very close(precise upto 6 decimal places) for case 1 and 2, but not exactly the same.

1.5 Question 5: Delta Hedging

From the previous question, we derived that the delta (δ) of a stock option is the ratio of the change in the price of the stock option to the change in the price of the underlying stock.
It is the **number of units of the stock we should hold** for each option shorted in order to create what is called a "riskless portfolio".
Delta hedging: The construction of a riskless portfolio (by being long/short in the underlying stock by a certain amount : long if delta is –ve, short if delta is +ve).
We know that the delta of a call option is positive (option price increases with increase in stock price), whereas the delta of a put option is negative (option price decreases with increase in stock price).
To calculate the riskless portfolios of a complex derivative payoff, we break it down into a linear combination of calls and puts, and hedge the weighted sum of their deltas.

1.5.1 Part A

Digital option: The payoff for a new derivative called Digital Option is given down below.



We can write the payoff function as:

$$0 \text{when} S_T < 40$$

$$\frac{(S_T - 40)}{20} \text{when} 40 < S_T < 60$$

$$1 \text{when} 60 < S_T$$

We know that for a call option with strike price K, the price of call option is given by: $C = Max\{(S_T - K), 0\}$.
Hence, we can say that for a call option with strike price 40$, we can write the price of stock as $C_{K=40} = Max\{(S_T - 40), 0\}$ and for a call option with strike price 60$, we can write the price as $C_{K=60} = Max\{(S_T - 60), 0\}$.
When $S_T > 60$, the payoff function:

$$\begin{aligned}
\text{payoff for}(S_T > 60) &= 1 \\
&= 20/20 \\
&= \frac{60 - 40}{20} \\
&= \frac{S_T - 40 - (S_T - 60)}{20} \\
\text{payoff for}(40 < S_T < 60) &= \frac{(S_T - 40)}{20} \\
\text{payoff for}(S_T < 40) &= 0 \\
\text{payoff combined} &= \frac{Max\{(S_T - 40), 0\} - Max\{(S_T - 60), 0\}}{20} \\
&= \frac{C_{K=40} - C_{K=60}}{20}
\end{aligned}$$

This equation means that 1 unit of digital option corresponds to 0.05(i.e.1/20) of the price of buying a call option at lower strike and selling a call option at higher strike. Hence we conclude that the given portfolio of digital option can be written as a linear combination of call options(namely buying 0.05 units of a call option with

strike of \$40 and selling 0.05 units of a call option with strike of \$60).
Suppose we are given the price of an option as a linear combination of 'n' call options, we can write the total payoff as:

$$C = a_1C_1 + a_2C_2 + ... + a_nC_n$$

We can write the delta of this function as:

$$\text{Delta of the function C} = \frac{\partial C}{\partial S}$$
$$\Delta = \frac{\partial}{\partial S}(a_1C_1 + a_2C_2 + ... + a_nC_n)$$
$$\Delta = a_1\frac{\partial C_1}{\partial S} + a_{@}\frac{\partial C_2}{\partial S} + ... + a_n\frac{\partial C_n}{\partial S}$$

Hence, we can calculate delta as the linear combination of the delta combinations from the above method and delta hedging can be done by using the final value of delta we get after the calculations.
For our problem, we may write the delta for the combined equation as:

$$\text{Delta} = \frac{\text{(delta for call option when K=40) - (delta for call option when K=60)}}{20}$$
$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$

We are given that we have started with 'n' units of the option. As we have broken it down into call option, we assume that we start with 'n' units of call option as specified. For delta hedging, we simply multiply the delta for the combined equation by 'n' and we will hedge Δn units of stock. We short Δn units of the stock when $\Delta >$ 0 and we long Δn units of the stock when $\Delta < 0$
We are given the parameters:
r = 0.12p.a.
σ = 0.30
T = 1 year
We need to create riskless portfolios at t=0 for 3 cases.

Case 1: $S_t = \$30$

$$\begin{aligned}
\Delta &= \frac{\text{(delta for call option when K=40) - (delta for call option when K=60)}}{20} \\
\Delta &= \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20} \\
&= \frac{N(-0.40894) - N(-1.76049)}{20} \\
&= \frac{0.34129 - 0.03916}{20} \\
&= 0.0151
\end{aligned}$$

Here, we should short(sell) 0.151n units of the stock.
Case 2: $S_t = \$50$

$$\begin{aligned}
\Delta &= \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20} \\
&= \frac{0.90213 - 0.476978}{20} \\
&= 0.0212578
\end{aligned}$$

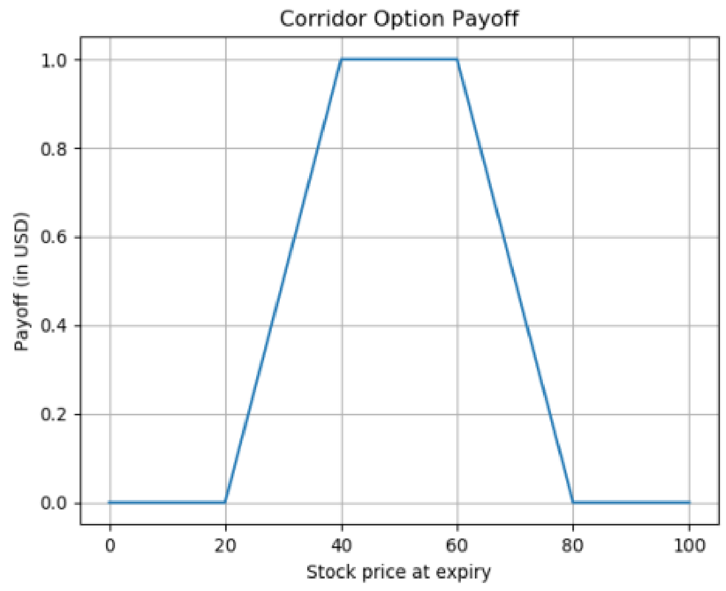
Here, we should short(sell) 0.0212n units of the stock.
Case 3: $S_t = \$70$

$$\begin{aligned}
\Delta &= \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20} \\
&= \frac{0.99214 - 0.85629}{20} \\
&= 0.00679
\end{aligned}$$

Here, we should short(sell) 0.00769n units of the stock.

1.5.2 Part B

We are given the payoff vs S_T graph of a different derivative option called a Corridor option.



We write the pay-off function for this graph as:

$$\begin{aligned}
\text{payoff for}(80 < S_T) &= 0 \\
\text{payoff for}(60 < S_T < 80) &= \frac{(S_T - 80)}{20} \\
\text{payoff for}(40 < S_T < 60) &= 1 \\
\text{payoff for}(S_T < 20) &= 0
\end{aligned}$$

$$\begin{aligned}
\text{payoff} &= \frac{Max\{(S_T - 20), 0\} - Max\{(S_T - 40), 0\} - Max\{(S_T - 60), 0\} + Max\{(S_T - 80), 0}{20} \\
&= \frac{C_{K=20} - C_{K=40} - C_{K=60} + C_{K=80}}{20}
\end{aligned}$$

Hence we conclude that the given portfolio of digital option can be written as a linear combination of call options(namely buying 0.05 units of a call option with strike of \$20 and \$80 and selling 0.05 units of a call option with strike of \$40 and \$60.)
r = 0.12p.a.
σ = 0.30
T = 1 year
We need to create riskless portfolios at t=0 for 5 cases. Therefore, we calculate delta and multiply it by n to create a riskless portfolio.
Case 1: S_t = \$10

$$\begin{aligned}\Delta &= \frac{N(d_1)_{K=20} - N(d_1)_{K=40} - N(d_1)_{K=60} + N(d_1)_{K=80}}{20} \\ &= 0.00196\end{aligned}$$

Here, we should short(sell) 0.00196n units of the stock.
Case 2: S_t = \$30

$$\begin{aligned}\Delta &= \frac{N(d_1)_{K=20} - N(d_1)_{K=40} - N(d_1)_{K=60} + N(d_1)_{K=80}}{20} \\ &= 0.02971\end{aligned}$$

Here, we should short(sell) 0.02971n units of the stock.
Case 3: S_t = \$50

$$\begin{aligned}\Delta &= \frac{N(d_1)_{K=20} - N(d_1)_{K=40} - N(d_1)_{K=60} + N(d_1)_{K=80}}{20} \\ &= -0.01123\end{aligned}$$

Here, we should long(buy) 0.01123n units of the stock.
Case 4: S_t = \$70

$$\begin{aligned}\Delta &= \frac{N(d_1)_{K=20} - N(d_1)_{K=40} - N(d_1)_{K=60} + N(d_1)_{K=80}}{20} \\ &= -0.0153\end{aligned}$$

Here, we should long(buy) 0.0153n units of the stock.

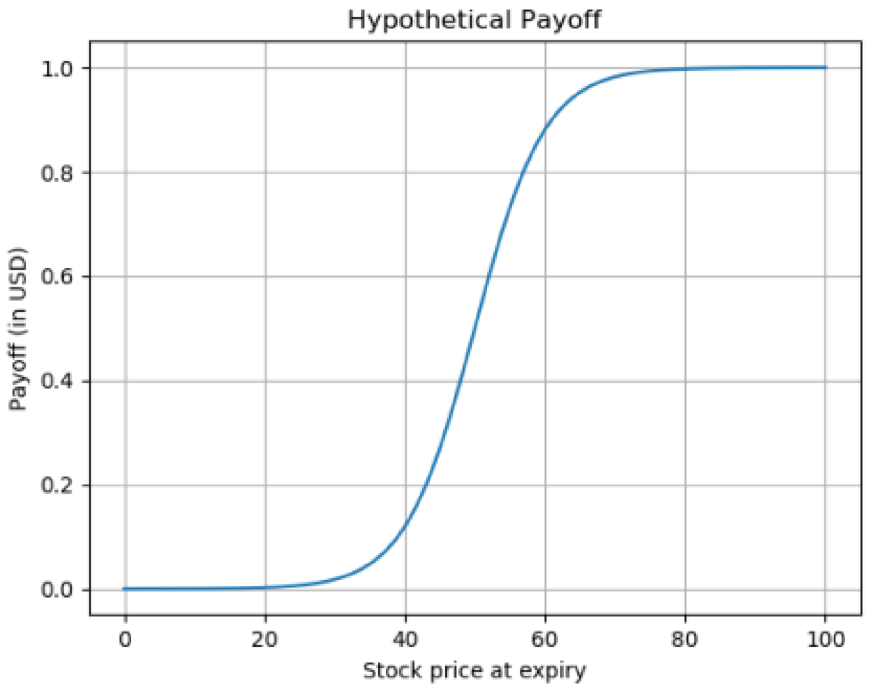
Case 5: $S_t = \$90$

$$\Delta = \frac{N(d_1)_{K=20} - N(d_1)_{K=40} - N(d_1)_{K=60} + N(d_1)_{K=80}}{20}$$
$$= -0.00719$$

Here, we should long(buy) 0.00719n units of the stock.

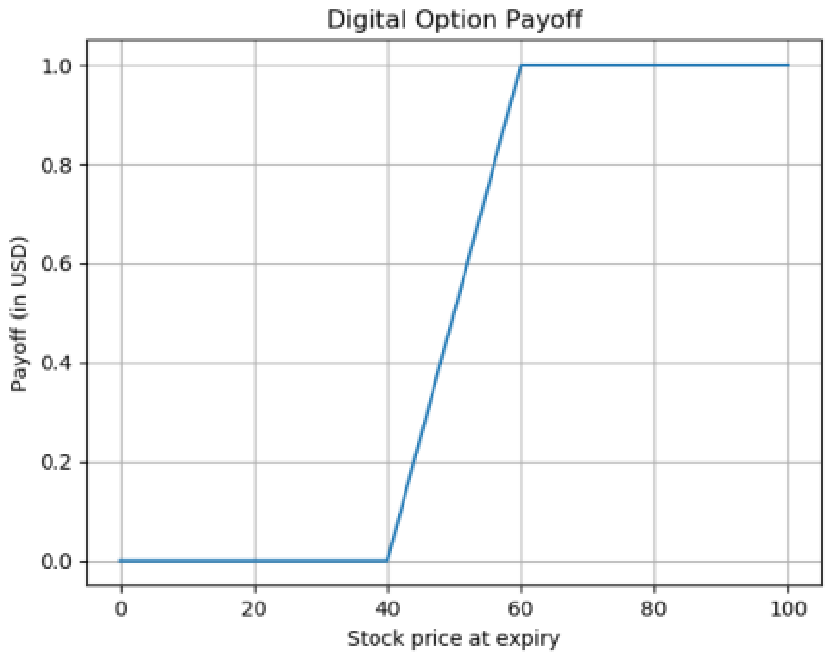
1.5.3 Part C

We are given a hypothetical payoff graph where payoff vs S_T is not linear. We are asked to describe a general purpose strategy/technique/hypothesis to create riskless portfolios for such a payoff for any given S_0.



As this graph is not linear, it cannot be broken down into equivalent linear combination of call and put options. Hence, we cannot

exactly calculate delta hedging by adding the weighted sum of corresponding deltas. The best approach towards solving this would be approximating it to a linear function, i.e., replacing the rounded corners of the curve with pointed ones so that it resembles a linear graph.
If we try to approximate the given hypothetical curve, by sharpening the corners, it will look like the curve in part A, which can be broken down into linear combination of call and put options that we have already calculated.
After sharpening the curve, it will look something like:



For this curve, for a given S_0 like we have calculated in part A, the delta is:

$$\Delta = \frac{N(d_1)_{S0,K=40} - N(d_1)_{S0,K=60}}{20}$$

Hence we can assume that the delta for the hypothetical curve is approximately the same.

Testing the strategy: Creating a riskless portfolio for the given parameters for t=0
r = 0.12p.a.
σ = 0.30
T = 1 year
Case 1: S_t = \$10

$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$
$$= 1.1689191606832365\text{x}10^{-06}$$

Here, we should short(sell) 0.00196n units of the stock.
Case 2: S_t = \$30

$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$
$$= 0.0151$$

Here, we should short(sell) 0.151n units of the stock
Case 3: S_t = \$50

$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$
$$= 0.0212578$$

Here, we should short(sell) 0.0212n units of the stock.
Case 4: S_t = \$70

$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$
$$= 0.00679$$

Here, we should short(sell) 0.00679n units of the stock.
Case 5: S_t = \$90

$$\Delta = \frac{N(d_1)_{K=40} - N(d_1)_{K=60}}{20}$$
$$= 0.00140$$

Here, we should short(sell) 0.00140n units of the stock.

2 Case Study B

2.1 Question 1: Circles?

2.1.1 Part A

We are given the following figure, three circles of equal radius 'r' arranged side by side inside a rectangle of length '$6r$' and breadth '$2r$'. A diagonal is drawn in the rectangle. Area 'A' is the bounded region to the left of the first circle, above the intersecting diagonal shaded with solid horizontal lines. Area 'B' is the area between the first circle and the diagonal shaded with dotted lines.
First, we will consider a general case for 'n' circles and then plug in the value as $n = 3$ for part A.

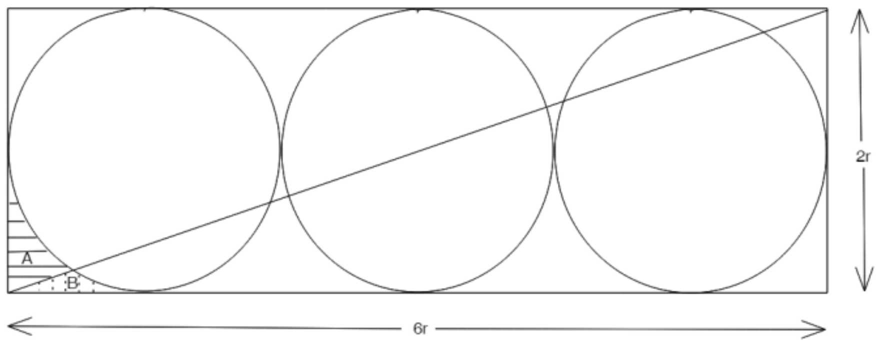
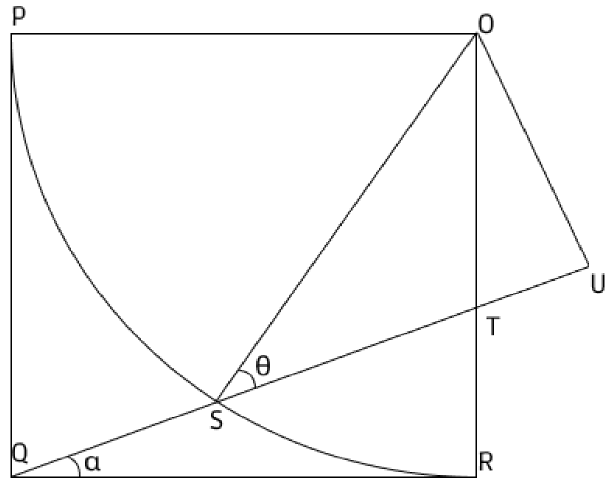


Figure 13: **Given figure for n=3 circles**

Let us consider the small square part of the circle which includes the shaded region. When we zoom in, the diagram will look something like:



Here, we can clearly see that tanα = 1/n
$\angle\ \alpha = tan^{-1}$(1/n)

Now, if we consider Q as the origin, co-ordinates of O will be (r,r) and the equation of the diagonal, i.e. line OT will be y = (1/n)x.

From co-ordinate geometry, we know that he distance between point $P = (x_0, y_0)$ and line $L : ax + by + c = 0$ is

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

Using this, we plug in the value of O(r,r) and the equation of line y - (1/n)x = 0 and find the distance OU:

$$\begin{aligned} OU &= \frac{|r - r/n|}{\sqrt{1^2 + (1/n)^2}} \\ &= \frac{(n-1)r/n}{\sqrt{n^2+1}/n} \\ &= \frac{(n-1)r}{\sqrt{n^2+1}} \end{aligned}$$

Hence, we conclude that,

$$\begin{aligned} sin\theta &= \frac{OU}{SO} \\ &= \frac{(n-1)r/\sqrt{n^2+1}}{r} \\ &= \frac{(n-1)}{\sqrt{n^2+1}} \end{aligned}$$

Hence, $\theta = sin^{-1}\frac{(n-1)}{\sqrt{n^2+1}}$
Since $RT = rtan\alpha$,
$RT = r/$n, and $OT = r$ - $r/$n = (n-1)$r/$n.
We can also see from the figure that,
$\angle\ SOT = \pi/2$ - θ - α

Area of region D is nothing but the area of triangle SOT. Therefore, we can write that:

$$\begin{aligned} \text{Area of region D} &= \frac{1}{2}(SO)(OT)sin(\angle SOT) \\ [D] &= \frac{1}{2}(r)(\frac{(n-1)r}{n})sin(\frac{\pi}{2}-\theta-\alpha) \\ [D] &= \frac{(n-1)cos(\theta+\alpha)r^2}{2n} \end{aligned}$$

Area of region C**:**

$$\begin{aligned} \text{Area of region C} &= \text{Area of sector} - \text{area of region D} \\ &= (\frac{\pi/2-\theta-\alpha}{2\pi})\pi r^2 - [D] \end{aligned}$$

Area of region B**:**

$$\begin{aligned} \text{Area of region B} &= \text{Area of triangle RUQ} - \text{area of region C} \\ &= \frac{1}{2}(r)(\frac{r}{3}) - [C] \\ &= \frac{r^2}{6} - [C] \end{aligned}$$

Area of region A:

Area of region A = Area of square $OPQR$ – area of a quarter POR – Area of region B

$$[A] = r^2 - \frac{\pi r^2}{4} - [B]$$

We are given that:

$$AreaRatio = \frac{\text{Area of A}}{\text{Area of A + Area of B}}$$

The code attached to this report returns the Area Ratio depending on the value of n set before compilation. The following is the output when n is set as 3 Therefore, using the result obtained by the code, **AreaRatio for n=3 is 0.707432**.

Description of the code file:

```
//Main functions of the program:
double area_of_D(double n, double r, double theta, double alpha) //To
    calculate area of region D

double area_of_C(double n,double r,double theta,double alpha) //To
    calculate area of region C

double area_of_B(double n,double r,double theta,double alpha) //To
    calculate area of region B

double area_of_A(double n,double r,double theta,double alpha) //To
    calculate area of region A

int main() // main() function to run the program and call other
    functions.
```

Instructions to run the code:

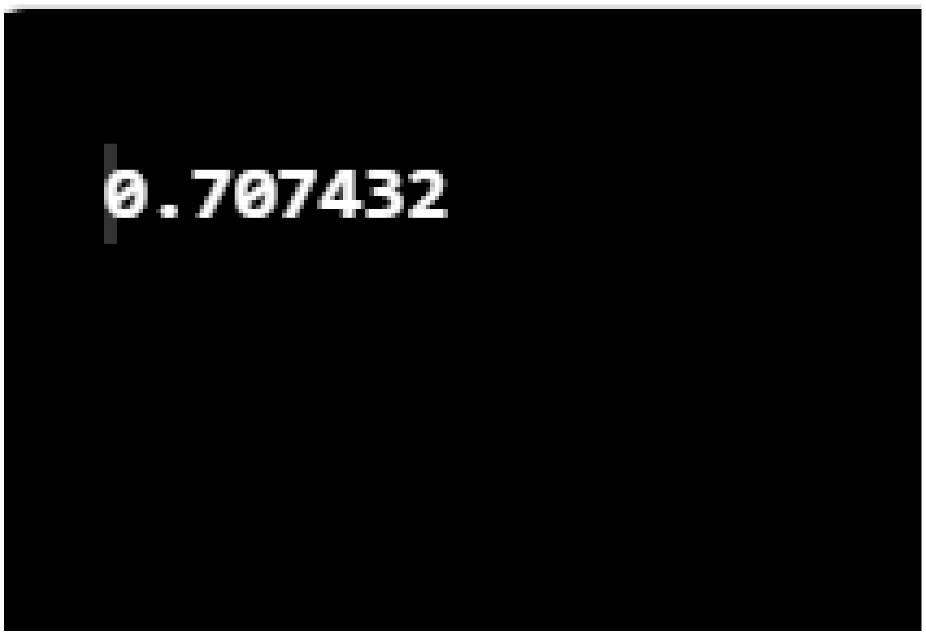
•To compile, type command:

```
g++ SHRUTHI_RAO_B_Q1_A_MAIN.cpp -o SHRUTHI_RAO_B_Q1_A_MAIN
```

•To execute, type command:

```
SHRUTHI_RAO_B_Q1_A_MAIN
```

Output of the code:



```
0.707432
```

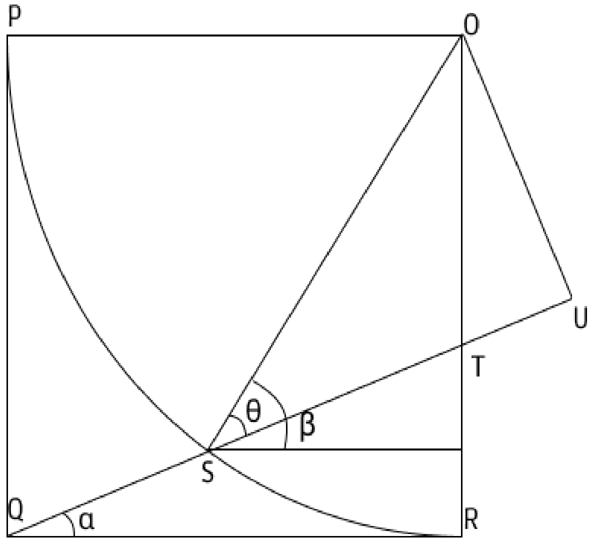
Figure 14: **Area Ratio output**

2.1.2 Part B

We want to find the range of Area Ratio.

$$\text{Area Ratio} = \frac{\text{Area of A}}{\text{Area of A + Area of B}}$$

First, we will show that the area ratio is an increasing function. The denominator in the Area ratio is equal to $r^2 - \pi r^2/4$. It is not affected by the slope of the diagonal or by n.



As we can clearly see, β increases from $\pi/4$ to $\pi/2$ as n increases. Now, using this and using parametric substitution, the area of the region B comes out as:

$$\text{Area of B} = \frac{r^2(1 - sin(\beta) - cos(\beta) + \pi/2 - \beta)}{2}$$

Differentiating the area with respect to β, we get:

$$\frac{d(Area)}{d\beta} = \frac{r^2(sin\beta - cos\beta - 1)}{2}$$

As

$$sin(\beta) - cos(\beta) - 1 \leq 0$$

for all β between $\pi/4$ and $\pi/2$, we can say that Area of region B is a strictly decreasing function.
Hence, we can conclude that Area of region A is a strictly increasing function.
When n=1,
Area of A = Area of B,
Area Ratio = 0.5
To find the upper limit of the Area Ratio, we find the ratio when n tends to infinity. We use Sandwich Lemma to prove this.
From the figure, we can see that the area of region B is always less than the area of triangle RUQ.

$$0 \leq \text{Area of B} \leq \frac{r^2}{2n}$$
$$0 \leq \lim_{n\to\infty} \text{Area of B} \leq \lim_{n\to\infty} \frac{r^2}{2n}$$
$$0 \leq \lim_{n\to\infty} \text{Area of B} \leq 0$$

Hence we can conclude the fact that when n tends to infinity, Area of B tends to zero and Area Ratio tends to 1.
Hence, the range of Area Ratio is:

$$\textbf{Area Ratio} \in [0.5, 1)$$

2.1.3 Part C

We want to find the minimum value of 'n' for which the AreaRatio is greater than or equal to the given percentage.
We consider the calculations we performed in part A for n circles and write a code which gives us the value of required 'n' as the output.
I have solved this question programmatically, using a c++ code which takes percentage as the input and gives the required 'n' as the output, i.e. minimum n for which AreaRatio is greater than or equal to the given percentage.
The values of 'n' obtained by the code are:

- a.50%
  n = 1
- b.70%
  n = 3
- c.90%
  n = 15
- c.99.9%
  n = 2240
- c.99.99%
  n = 23012
- c.100%
  n = N.A.

We see that for 100%, we do not obtain any output, this is because Area Ratio never reaches 1 (or 100%) which we also proved in partB and also because the code gets into an infinite loop, so we don't get an output or we get a timeout in the command prompt.

Description of the code file:

```
//We use the same functions we used in part A to calculate the areas of
    the individual regions and add a while loop in the main() function
    and we have changed the value of n.
while( Area_ratio*100 < Percentage){
       n++;
       q = (n-1)/sqrt(n*n +1);
       w = 1/sqrt(n*n +1);
       theta = asin(q);
       alpha = asin(w);
       total_area_of_shaded_region = r*r - M_PI*r*r/4;
       Area_ratio = area_of_A(n, r, theta,
           alpha)/total_area_of_shaded_region;
   } //Gives us the minimum value of n required
```

Instructions to run the code:

•To compile, type command:

```
g++ SHRUTHI_RAO_B_Q1_C_MAIN.cpp -o SHRUTHI_RAO_B_Q1_C_MAIN
```

•To execute, type command:

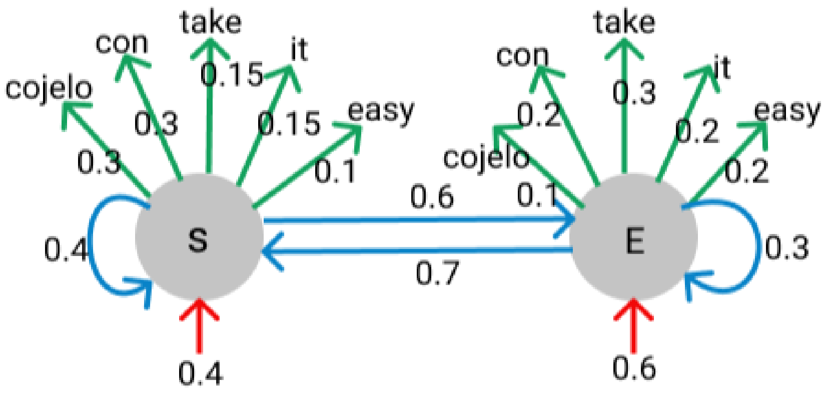
```
SHRUTHI_RAO_B_Q1_A_MAIN
```

•Type in the percentage without the symbol, example if the given percentage is 50%, type in 50.

2.2 Question 2: Keep Calm and Carry on...

We want to design a around language switching, assuming we know the model parameters and the final sentence, we want to derive an algorithm to guess the most likely language transition states.
We are given the following Language Model Parameters:
Vocabulary = (”cojelo”, ”con”, ”take”, “it”, “easy”)
Languages = (”E”, ”S”)
Sentence start probability :
Ps[‘E’] = 0.6
Ps[‘S’] = 0.4
Transition probability: the probability of the language of the next word given the current language.
Pt[‘S’|‘E’] = 0.7
Pt[‘E’|‘E’] = 0.3
Pt[‘S’|‘S’] = 0.4
Pt[‘E’|‘S’] = 0.6
Emission probability :the probability of emitting a word in the vocabulary given the language of the speaker.
Pe[‘cojelo’|‘E’] = 0.1
Pe[‘con’|‘E’] = 0.2
Pe[‘take’|‘E’] = 0.3
Pe[‘it’|’E’] = 0.2
Pe[‘easy’|‘E’] = 0.2
Pe[‘cojelo’|‘S’] = 0.3
Pe[‘con’|‘S’] = 0.3
Pe[‘take’|‘S’] = 0.15
Pe[‘it’|‘S’] = 0.15
Pe[‘easy’|‘S’] = 0.1

For this, we design a state diagram, which will look something like:



To find the probability of a given language string, since it is a unigram model, we can simply multiply the probability of all the given words. We start by taking the first language and multiply it by the corresponding sentence start probability and proceed to multiply it with the required transition probabilities and finally we multiply it by the corresponding emission probabilities of the required words given the language(s).

2.2.1 Part A

We are given 5 strings and we need to find the most likely language string and it's corresponding probability. I have solved the problem programmatically, the code takes the string as the input and calculates the probability of all the possible language strings possible for given string and gives the most likely string as the output along with the corresponding probability(rounded off upto 5 decimal places).

The solution to each of the 4 problems is:

- a. Cojelo Con Take It Easy
  S S E S E : 0.00003
- b. Con Take It Easy
  S E S E : 0.00027
- c. Easy Con
  E S : 0.02520
- d. Cojelo Easy Take It
  S E S E : 0.00018

Description of the code file:

```
//Main functions used in the code:
void init (void)
//used to redirect the words to an array later using a map

void rec (int index, bool flag, int size, double temp)
//creating a recursive function that calculates the probability of each
    language string
//index represents the position of the word, flag represents the
    language(E for true and S for false)
//size represents the number of words and temp will give the temporary
    probability

int main ()//used to call the functions and get the final output needed
```

Instructions to run the code:

•To compile, type command:

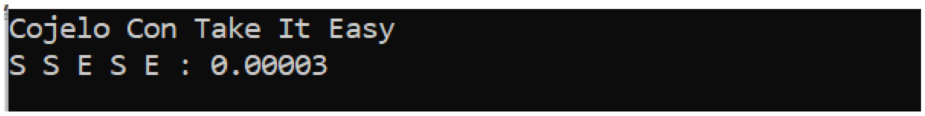
```
g++ SHRUTHI_RAO_B_Q2_A_MAIN.cpp -o SHRUTHI_RAO_B_Q2_A_MAIN
```

•To execute, type command:

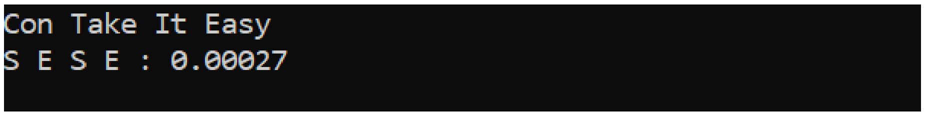
```
SHRUTHI_RAO_B_Q2_A_MAIN
```

•Type the string as the input as given in the question, with spaces.(Example: for a, type "Cojelo Con Take It Easy" without the quotes to get the required output)

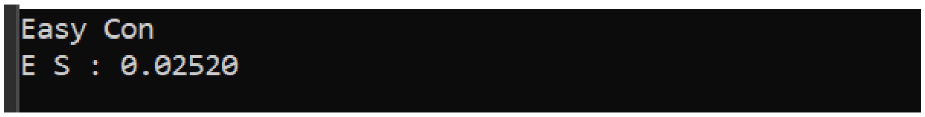
Output of the code:



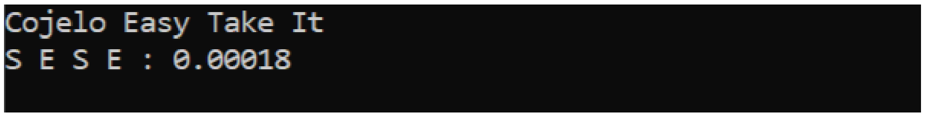
```
Cojelo Con Take It Easy
S S E S E : 0.00003
```



```
Con Take It Easy
S E S E : 0.00027
```



```
Easy Con
E S : 0.02520
```



```
Cojelo Easy Take It
S E S E : 0.00018
```

2.2.2 Part B

We are asked to write a program which takes a string as an input and gives the most likely language string along with the corresponding probability as the output.
I have written the code in c++ along with comments to explain the code wherever needed. I have used the same code to find the probability and language string in part A and B.

Description of the code file:

Same as Part A

Instructions to run the code:

•To compile, type command:

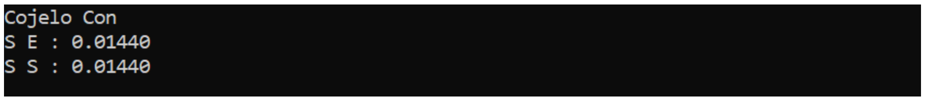
```
g++ SHRUTHI_RAO_B_Q2_B_MAIN.cpp -o SHRUTHI_RAO_B_Q2_B_MAIN
```

•To execute, type command:

```
SHRUTHI_RAO_B_Q2_B_MAIN
```

•Type the string as the input as given in the question, with spaces.(Example: type "Cojelo Con" without the quotes to get the required output)

Output of the code:



```
Cojelo Con
S E : 0.01440
S S : 0.01440
```