```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import scipy.stats as stats
```


Assig

In the last c… …udied derivatives, using the Black-Scholes model to value optio… …ent, we will study the issue of *dynamic hedging*, that is, consistentl… …shares to offset our derivatives position according to $\Delta = \partial V / \partial S$.

Quick Review

Prior to expiration, with time T left until expiration, the option has a value which we can find with the Black-Scholes formula:

$$c = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

if S_0 is a geometric Brownian motion. We use Φ to denote the cumulative normal distribution with zero mean and unit variance. The parameters d_1, d_2 are:

$$d_1 = \frac{\log S_0/K + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} = -\frac{\log K/S_0 - \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \quad (1)$$

$$d_2 = d_1 - \sigma\sqrt{T}. \quad (2)$$

Recall:

- S_0 - the current stock price.
- K - the strike price or exercise price of the call option.
- σ - the volatility of the underlying GMB.
- T - the time until expiration.
- r - the risk-free interest rate.

Put options are then valued according to put-call parity,

$$p = c - S_0 + Ke^{-rT}.$$

In Python, we can define the functions `call_value()` and `put_value()` as we have done in class:

```
In [2]: def call_value(S0, K, sigma, T, r):
            d1 = (np.log(1.*S0/K) + T*(r+0.5*sigma*sigma)) / (sigma*np.sqrt(T
        ))
            d2 = d1 - sigma*np.sqrt(T)

            return S0*stats.norm.cdf(d1) - K*np.exp(-r*T)*stats.norm.cdf(d2)

        def put_value(S0, K, sigma, T, r):
            return call_value(S0, K, sigma, T, r) - S0 + K*np.exp(-r*T)
```

Problem 1

The value of a European style call option is

$$c = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

where

$$d_1 = \frac{\log S_0/K + \left(r + \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} \quad (3)$$

$$d_2 = d_1 - \sigma\sqrt{T}. \quad (4)$$

The option's delta is $\Delta = \partial c/\partial S = \Phi(d_1)$.

In our discussion of the Black-Scholes model, we have seen that Δ is the number of shares the seller of a call options need to buy to hedge themselves, i.e., make their position risk-free for an instant.

Since c and S, the value of the call option and the stock price change over time in relation to each other, Δ is not constant.

See the following realization below of a call option with parameters that reflect the stock market of April 2023:

- $S_0 = 4100.$
- $K = 4100.$
- $\sigma = 24.0\%.$
- $T = 1$ year.
- $r = 4.6\%.$

This option is *at the money* at the time of the sale, i.e., the stock price is equal to the exercise price. An option seller can sell such a call option for the following amount:

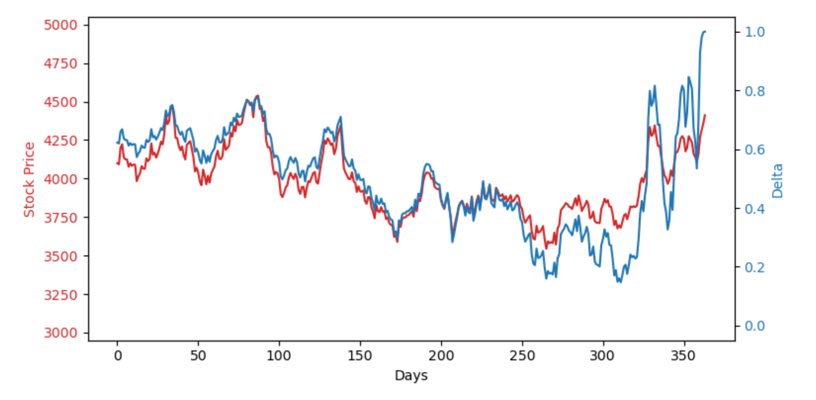
```
In [3]: S0 = 4100
        K = 4100
        sigma = 0.24
        T = 1
        r = 0.046

        print("Call can be sold for {:.2f}".format(call_value(S0, K, sigma, T
        , r)))
```

```
Call can be sold for 481.95
```

Note that below we use 365 days instead of 252 trading days because options are priced to include things that happen on weekends or holidays too, as they are affected by the final price. A simplification is to thus trade all days the same in our random walk, whether they are trading days or weekends and holidays.

We simulate the stock price until the day before expiration.

```
In [4]: np.random.seed(587)

        daily_sigma = sigma / np.sqrt(365)
        daily_r = r / 365

        days = 365

        Z = np.random.normal(0, 1, days)
        S = S0 * np.cumprod(1 + daily_r + daily_sigma*Z[1:])
        S = np.insert(S, 0, S0)


        def compute_delta(S, K, sigma, T, r):
            d1 = (np.log(1.*S/K) + T*(r+0.5*sigma*sigma)) / (sigma*np.sqrt(T)
        )
            return stats.norm.cdf(d1)


        time_steps = range(days-1)
        delta = [compute_delta(S, K, sigma, (days-t)/days, r)
                 for S, t in zip(S[:-1], time_steps)]


        fig, ax1 = plt.subplots(figsize=(8, 4))

        color = 'tab:red'
        ax1.set_xlabel('Days')
        ax1.set_ylabel('Stock Price', color=color)
        ax1.plot(time_steps, S[:-1], color=color)
        ax1.tick_params(axis='y', labelcolor=color)
        ax1.set_ylim(2950, 5050)

        # Create a second set of axis overlayed on the original one.
        ax2 = ax1.twinx()

        color = 'tab:blue'
        ax2.set_ylabel('Delta', color=color)
        ax2.plot(time_steps, delta, color=color)
        ax2.tick_params(axis='y', labelcolor=color)
        ax2.set_ylim(-0.05, 1.05)

        fig.tight_layout()
        plt.show()
```



The option starts well in the money, and the option's Δ is around 0.6, meaning that an option seller would need to buy 0.6 shares for every call they have sold. As the value of the underlying shares falls to around 3600 after half a year, the Δ drops accordingly to 0.3, meaning that an option seller would need to own 0.3 shares for every call they have sold. Note that, with a long time until expiration, the Δ closely tracks the stock price, as convexity and time decay are minor factors.

At around the 270-day mark, the shares start increasing in price slowly but steadily. To some extent, the Δ of the call option increases, as well. However, as the stock price of roughly 3800 is still significantly below the strike price of 4100, the Δ drops when the stock price pulls back a bit or remains flat for some price. Here we see the time decay of the option value!

Nearing expiry, the shares are well above 4000, and thus closing in on the strike price. Since it is not unlikely that the option ends *in the money*, the Δ becomes much more volatile. Here we see the convexity of the option taking over.

Eventually, the stock price ends at 4411.19 one day before expiration, meaning that the option is well in the money and the Δ approaches 1, as it is almost certain that this option ends in the money on the final day.

In general, we observe that the option seller has to sell their shares into falling stock prices (lower stock prices imply lower Δ) and buy shares into rising stock prices (higher stock prices imply higher Δ); this is the cost of Δ-hedging.

In the following, we will explore the cost of this Δ-hedging. We start by setting up a data frame, and you will be guided through the individual steps in the subparts of the problem.

The data frame `df` contains in the column *Delta* the values for Δ over time in our simulation above. Note that the option starts with $\Delta = 0.622353$, and on day 1 it drops to $\Delta = 0.619323$.

We need to buy 0.644323 shares on day 0 to balance the Δ of the shares that trade at 4100.

```
In [5]: df = pd.DataFrame(data=delta, columns=["Delta"])
        df.loc[:, "S"] = S[:-1]
        df.loc[0, "Transaction"] = df.loc[0, "Delta"]
        df.head()
```

Out[5]:

	Delta	S	Transaction
0	0.622353	4100.000000	0.622353
1	0.619323	4092.600810	NaN
2	0.659939	4200.806296	NaN
3	0.667929	4223.148785	NaN
4	0.636542	4138.617181	NaN

Part a)

On day 1, we need to reduce our position in the shares to adjust to the new Δ, selling some number at the new price of 4092.60; similarly on day 2 and 3 at the new prices of 4200.80 and 4223.15, we need to buy some shares as the Δ has gone up. On day 4, we need to sell some shares at 4138.62.

Compute the transactions necessary in each time step for the entire data frame by finding the change in Δ and fill the column *Transaction* with this info.

```
In [ ]:
```

Part b)

Compute the cost for each transaction and keep track of the cumulative cost. Store the info in the columns *Cost* and *Acc_Cost*.

```
In [ ]:
```

Part c)

To buy the shares, the option seller needs to borrow money. Use our interest rate r, stored as `r` from the previous problem to compute the interest expense each day due on the cumulative cost of the share purchases in column *Agg_Cost*. Then find the cumulative interest expense. Store these values in the columns *Int_Expense* and *Acc_Int_Expense*, respectively.

```
In [ ]:
```

Part d)

Compute the profit or loss for the option seller.

- The option seller has received the premium plus interest, $c\exp[rT]$.
- The option seller sells their shares, if any, for K if the option is exercised or S_T if the option is not exercised. Mathematically, the option seller receives $\min(S_T, K)$.
- The option seller has incurred costs building their position to Δ-hedge.
- The option seller pays the accumulated interest expense.

```
In [ ]:
```

Part e)

Study five different scenarios to find the profit or loss of Δ-hedging. Plot the stock price and the option's Δ over time. Then comment on the results for the P/L for the option seller.

1. The share price jumps to 4150 on day 1 and remains there until the option expires.
2. The share price remains at 4100 until the option expires, not moving on any day.
3. The share price goes from 4100 to 4150 at a constant rate, i.e., by an equal number of points each day.
4. The share price increases by 5 points each day for the first 200 days, then it drops by 5 points each day for the remaining days.
5. The share price drops by 2% on the first day, recovers by 2% on the second day, and it continues to alternate between -2% and +2%.

```
In [ ]:
```