3. Futures

Investigate the mechanics and use of short selling S&P 500 futures contracts via the *Chicago Mercantile Exchange* (CME) as outlined in Document *A* of the case study.

The following investigation investigates futures prices of two contracts that would have been available during this time, i.e. *ESH20* and *ESM20*, with maturities in March and June, respectively. The trading prices for these contracts were sourced from BarChart Premiere historical data. A literature review (where relevant) and coding examples have been provided as part of this analysis.

Key Assumptions & limitations

- For the purposes of this assignment, we will assume that the evaluation date (i.e. the date in which Pershing Square must make a decision on their Hedging strategy) is February 21, 2020
- Purchasing of futures on the *Chicago Mercantile Exchange* (CME)
- Where discrepancies between Yahoo Finance share prices and derived share prices from *Exhibit 3* were found, the Yahoo Finance share prices were adopted. Details are outlined further in 3.1

This investigation has been divided into four sections, as listed in the table of contents below:

Table of Contents

Libraries

In [ ]:
```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import numpy as np
from datetime import datetime

import warnings
warnings.filterwarnings("ignore")
```

3.1 Hedging Position and Key details

To protect Pershing Square's portfolio from adverse market movements, they could utilise S&P 500 index futures contracts as a hedging instrument. This approach would allow the portfolio to gain protection against potential downside movements in the S&P 500

A futures contract is an agreement to buy or sell the underlying asset (in this case, the S&P 500 index) at a predetermined price on a specific date in the future. This allows the holder to lock in a price for the underlying asset, helping to protect against unfavorable price movements.

Ongoing Cashflows and Costs

Whilst there are no upfront costs or regular payments like options and Credit Default Swaps (CDS) we must consider that an initial margin and maintenance margin must be maintained over the life of ownership of the asset. A coding example calculating the initial margin of this contract is demonstrated in section 3.1. The initial margin figure is sourced from CME risk data.

Since and observation of the payoff yield an immediate positive (and never a non-negative) payoff, we did not calculate any maintenance margin requires for these contracts.

Futures Contract Payoff

The payoff of a futures contract at expiration depends on the relationship between the futures price (S_F) and the spot price of the underlying asset (S_T). If the spot price at expiration is higher than the agreed-upon futures price, the long position benefits, while the short position loses. Conversely, if the spot price is lower, the short position benefits.

The payoff of a short position futures contract is given by:

$$\text{Futures Contract Payoff} = K - S_T$$

Where:

- S_T is the spot price of the S&P 500 index at expiration.
- K is the agreed-upon futures price.

Determining the Optimal Number of Contracts

To determine the optimal number of futures contracts required to hedge the portfolio, we can use the optimal hedging ratio. The goal is to align the portfolio's exposure to market movements with the performance of the hedging instrument (the S&P 500).

The optimal number of contracts h can be calculated using the following formula:

$$h = \frac{\beta V}{F} = \bar{\rho}\frac{\bar{\sigma}_A V}{\bar{\sigma}_K F}$$

Where:

- h is the number of futures contracts to purchase or sell.
- β is the beta of the portfolio relative to the S&P 500 (this measures the portfolio's sensitivity to market movements).
- V is the value of the portfolio being hedged.
- F is the notional value of one futures contract on the S&P 500 index, calculated as the index value multiplied by the contract multiplier.
- σ_{A_t} is the standard deviation (volatility) in $A_t - A$.
- σ_{K_t} is the standard deviation (volatility) in $K_t - K$.
- ρ is the correlation between $A_t - A$ and $K_t - K$.

For the purposes of this assignment, σ_{A_t}, σ_{K_t} and ρ were calculated from historical daily returns in the prior four-year period (`HISTORICAL_PRICE_DATE`) to the evaluation date (`EVAL_DATE`).

In [ ]:
```
# Contract information
ESH20_CONTRACT_SIZE = 50
ESM20_CONTRACT_SIZE = 50
ESH20_HIST_PRICES = 'futures_data/esh20_price-history-09-24-2024.csv'
ESM20_HIST_PRICES = 'futures_data/esm20_price-history-09-24-2024.csv'
ESH20_MATURITY = '2020-03-30'
ESM20_MATURITY = '2020-06-30'
ESH20_INITIAL_MARGIN = 11_000.0
ESM20_INITIAL_MARGIN = 11_000.0

# Firm specific information
FIRM_VALUE = 7_621.28 * 1_000_000

# Contextual information
HISTORICAL_PRICE_DATE = '2016-01-04'
EVAL_DATE = '2020-02-21'
EVAL_DATE_PLUS_ONE = '2020-02-22'
EVAL_END_DATE = '2020-06-30'
```

Calculate the portfolio value, weights, historical returns and standard deviations.

This is a bit more complicated because the weighting for each firm in the portfolio varies as per the appendix provided on the assignment task sheet.

Process:

Import the historical prices for each firm, evaluate the firm value based upon the constituent assets owned in the portfolio and calculate the daily returns of the portfolio

In [ ]:
```
file_path = 'Perishing_portfolio.xlsx'
sheet_name = 'portfolio'
share_nums = {}

df = pd.read_excel(file_path, sheet_name=sheet_name)

for index, row in df.iterrows():
    ticker = row['Ticker Code']
    shares = row['Number of Shares']
    share_nums[ticker] = shares
```

```
# Also add the ^spx index
share_nums['^spx'] = 1

date_range = pd.date_range(start=HISTORICAL_PRICE_DATE, end=EVAL_END_DATE)
df = pd.DataFrame(index=date_range)

# stock prices for the evaluation period
for ticker, shares in share_nums.items():
    historical_data = yf.download(ticker, start=HISTORICAL_PRICE_DATE, end=EVAL_

    if not historical_data.empty:
        daily_values = historical_data * shares
        df[ticker] = daily_values

df = df.dropna(how='any')
df['Firm Value'] = df.loc[:, df.columns != '^spx'].sum(axis=1)
df_historic = df[df.index <= EVAL_DATE_PLUS_ONE]

df.index.name = 'Date'

p_returns = np.log(df_historic['Firm Value']).diff(1).dropna()
spx_returns = np.log(df_historic['^spx']).diff(1).dropna()

df_historic['Portfolio Returns'] = p_returns
df_historic['S&P 500 Returns'] = spx_returns
df_historic = df_historic.dropna()

# Portfolio price (we have discovered in previous sections)
p_price_eval_date = df_historic['Portfolio Returns'].loc[EVAL_DATE]

# Portfolio sigma
p_std_dev = np.std(p_returns)

# S&P 500 sigma
spx_std_dev = np.std(spx_returns)

# Safekeeping:
df_historic.to_excel('historic_portfolio_values.xlsx', sheet_name='Values')

df.head()
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	CMG	HLT	LOW	QSR	BRK-B	
2016-01-04	7.736718e+08	4.357516e+08	5.524744e+08	4.257521e+08	5.250389e+08	6.644987
2016-01-05	7.740511e+08	4.359580e+08	5.536490e+08	4.094645e+08	5.270467e+08	6.575578
2016-01-06	7.355062e+08	4.132306e+08	5.426402e+08	3.970094e+08	5.273680e+08	6.514694
2016-01-07	7.171130e+08	3.977343e+08	5.289166e+08	3.879076e+08	5.199391e+08	6.362481
2016-01-08	7.124415e+08	3.956681e+08	5.201833e+08	3.810812e+08	5.153212e+08	6.355175

Calculate optimal portfolio

> Note: The notional value and initial margin are the same as both contracts have the same contract size

In [ ]:

```
# Calculate rho
rho=np.corrcoef(df_historic['S&P 500 Returns'], df_historic['Portfolio Returns']

# Firm portfolio value and returns
beta = rho * spx_std_dev / p_std_dev

# optimal hedge ratio
spx_price_eval_date = df_historic['^spx'].loc[EVAL_DATE]
F = spx_price_eval_date * ESM20_CONTRACT_SIZE #
H = beta * FIRM_VALUE / F

# Calculate initial margin
margin_cost_esm20 = H * ESM20_CONTRACT_SIZE
margin_cost_esh20 = H * ESH20_CONTRACT_SIZE
```

Key Statistics

In [ ]:

```
stats = f'''
############ KEY STATISTICS ############
Standard deviation of portfolio from historical date ({HISTORICAL_PRICE_DATE}) t
Standard deviation of S&P 500 Sigma from historical date ({HISTORICAL_PRICE_DATE
Rho: {rho:.6f}
Beta: {beta:.6f}

Face value of futures contract on evaluation date ({EVAL_DATE}): ${F:.2f}


Optimal number of contracts to hedge firm position: {int(round(H, 0))}
(unrounded {H:.2f})

############ INITIAL MARGIN ############
Initial Margin (ESM20): {margin_cost_esm20:.2f}
Initial Margin (ESH20): {margin_cost_esh20:.2f}
```

```
'''

print(stats)
```

```
############ KEY STATISTICS ############
Standard deviation of portfolio from historical date (2016-01-04) to evaluation d
ate (2020-02-21): 0.010099
Standard deviation of S&P 500 Sigma from historical date (2016-01-04) to evaluati
on date (2020-02-21): 0.008079
Rho: 0.741110
Beta: 0.592861

Face value of futures contract on evaluation date (2020-02-21): $166887.50


Optimal number of contracts to hedge firm position: 27074
(unrounded 27074.27)

############ INITIAL MARGIN ############
Initial Margin (ESM20): 1353713.70
Initial Margin (ESH20): 1353713.70
```

3.2 (Retrospective) Key Timings

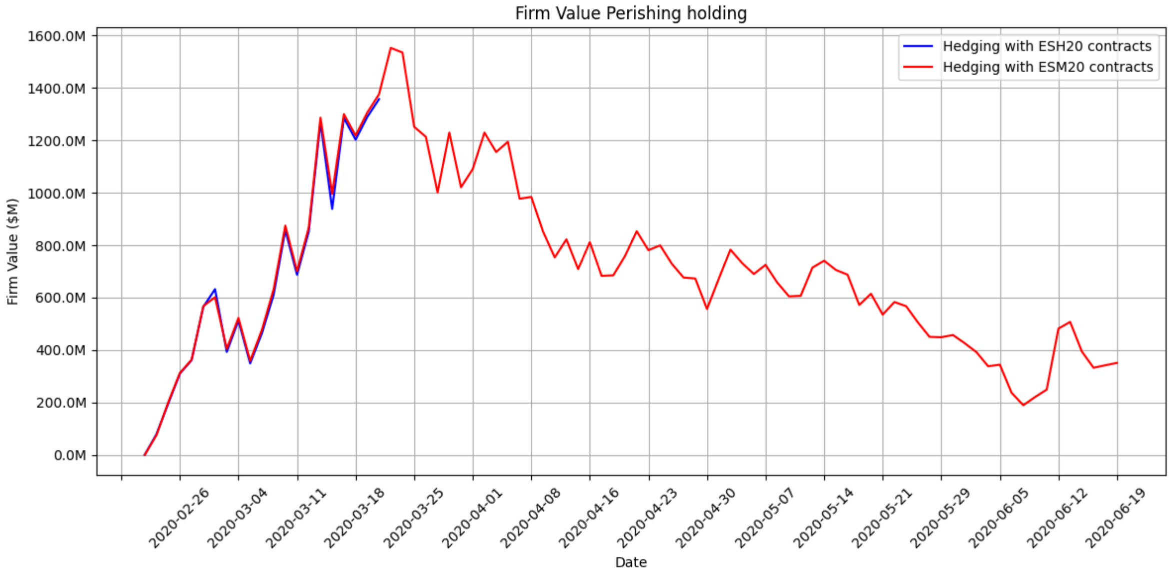
Firstly, we will graphically illustrate the payoffs of the purchasing the contracts starting from the evaluation date to the contract maturity, i.e. 31st March and 30th June.

We will characterise the best exit timing of these securities as the date which would yield the maximum payoff between the evaluation date and maturity.

In [ ]:

```
import matplotlib.ticker as ticker
# Retrieve historic futures data for ESH20 and ESM20

def generate_futures_payoffs(historical_prices_wd, contract_size):

    futures_prices_df = pd.read_csv(historical_prices_wd, index_col='Time', pars
    futures_prices_df = futures_prices_df[futures_prices_df.index >= EVAL_DATE]
    futures_prices_df.index = pd.to_datetime(futures_prices_df.index).strftime('
    futures_prices_df = futures_prices_df.sort_index()
    futures_prices_df = futures_prices_df.rename(columns={'Open': 'Price'})

    esh20_eval_date = futures_prices_df['Price'].loc[EVAL_DATE]

    futures_prices_df['Payoff from EVAL_DATE'] = (esh20_eval_date - futures_pric
    return futures_prices_df

esh20_prices_df = generate_futures_payoffs(ESH20_HIST_PRICES, ESH20_CONTRACT_SIZ
esm20_prices_df = generate_futures_payoffs(ESM20_HIST_PRICES, ESM20_CONTRACT_SIZ

plt.figure(figsize=(12, 6))
plt.plot(esh20_prices_df.index, esh20_prices_df['Payoff from EVAL_DATE'], linest
plt.plot(esm20_prices_df.index, esm20_prices_df['Payoff from EVAL_DATE'], linest
plt.title('Firm Value Perishing holding')
plt.xlabel('Date')
plt.ylabel('Firm Value ($M)')
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=5))
```

```
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: f'{x*1e-
plt.grid()
plt.legend()
plt.tight_layout()
plt.show()
```



In [ ]:

```
# The best date to close the futures contract with a reversing trade would be:
max_price_date = esm20_prices_df['Payoff from EVAL_DATE'].idxmax()

print(f'The best exit timing of the market would be {max_price_date.strftime('%Y
```

```
The best exit timing of the market would be 2020-03-23.
```

3.3 Profits

We now consider the overall profits of Pershing Square maintaining their current portfolio and the futures hedging position.

The profit diagram of their firm is illustrated is the worked coding example below.

In [ ]:

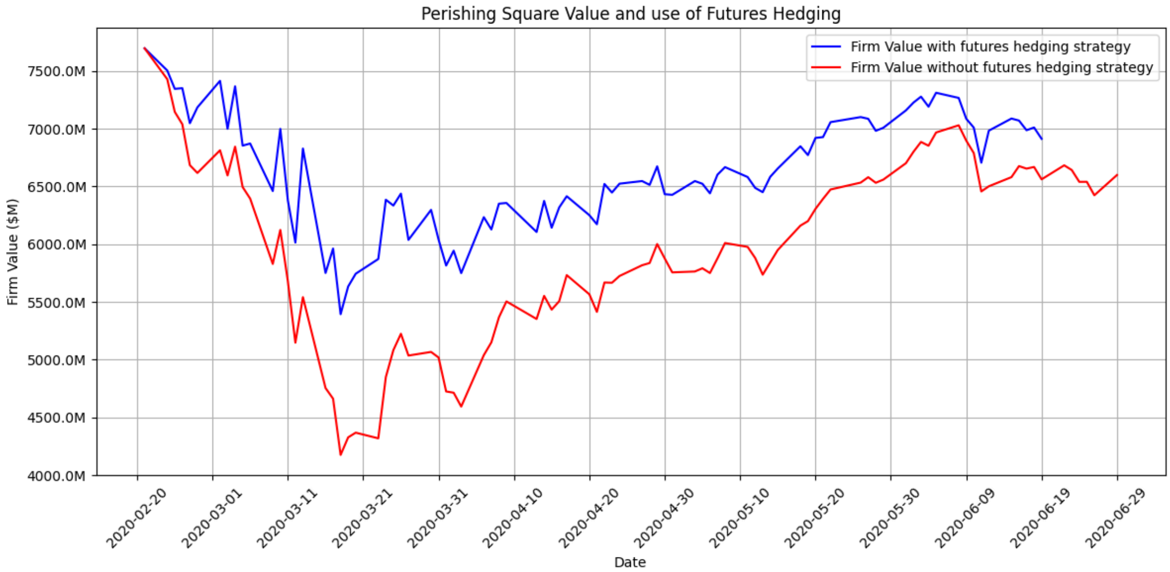
```
# Given the payoff diagram above, we will only look at the ESM20 contract since
df_profits = df[df.index >= EVAL_DATE].copy()
df_profits = df_profits[['Firm Value']]

# Shuffling because the data isn't perfect
df_profits.index = pd.to_datetime(df_profits.index).normalize()
esm20_prices_df.index = pd.to_datetime(esm20_prices_df.index).normalize()
aligned_esm20_payoff = esm20_prices_df['Payoff from EVAL_DATE'].reindex(df_profi

df_profits.loc[:, 'ESM Payoff'] = aligned_esm20_payoff

firm_value_eval_date = df_profits['Firm Value'].loc[EVAL_DATE]
df_profits['Firm Profit'] = df_profits['Firm Value'] - firm_value_eval_date
df_profits['Firm Profit With Futures Hedge'] = df_profits['ESM Payoff'] + df_prc
df_profits['Firm Value With Futures Hedge'] = df_profits['ESM Payoff'] + df_prof

df_profits.head()
```

Out[ ]:

Date	Firm Value	ESM Payoff	Firm Profit	Firm Profit With Futures Hedge	Firm Value With Futures Hedge
2020-02-21	7.696251e+09	0.000000e+00	0.000000e+00	0.000000e+00	7.696251e+09
2020-02-24	7.429246e+09	7.513111e+07	-2.670049e+08	-1.918738e+08	7.504377e+09
2020-02-25	7.145713e+09	1.996728e+08	-5.505380e+08	-3.508652e+08	7.345386e+09
2020-02-26	7.038965e+09	3.123694e+08	-6.572857e+08	-3.449162e+08	7.351335e+09
2020-02-27	6.684721e+09	3.631337e+08	-1.011530e+09	-6.483962e+08	7.047855e+09

In [ ]:

```
plt.figure(figsize=(12, 6))
plt.plot(df_profits.index, df_profits['Firm Value With Futures Hedge'], linestyl
plt.plot(df_profits.index, df_profits['Firm Value'], linestyle='-', color='r', l
plt.title('Perishing Square Value and use of Futures Hedging')
plt.xlabel('Date')
plt.ylabel('Firm Value ($M)')
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=10))
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: f'{x*1e-
plt.grid()
plt.legend()
plt.tight_layout()
plt.show()
```



3.4 Sensitivity Analysis

A futures contract can be priced using the cost of carry approach

$$K = Se^{(r+s-q)T}$$

(using **Compound Interest**)

i.e. that the futures price is dependent on the risk-free interest rate r, storage cost c and dividend yield q in some period T.

3.4.1 Forecast Changes in Cost of Carry

The following code example forecasts changes in the estimated cost of carry:

- The cost of carry is first derived as of the evaluation date (February 21, 2020).
- The payoffs are then evaluated under different cost of carry values, considering forecast periods of one week, one month, and two months into the future.

In [ ]:
```
eval_date_as_date = datetime.strptime(EVAL_DATE, '%Y-%m-%d')
esm20_maturity_as_date = datetime.strptime(ESM20_MATURITY, '%Y-%m-%d')

T = (esm20_maturity_as_date - eval_date_as_date).days / 360

K = esm20_prices_df['Price'].loc[EVAL_DATE]
S = spx_price_eval_date

cost_of_carry = np.log((K / S))* (1 / T)

print(f'K: {K} \nS: {S}\nT: {T}\nCost of Carry: {cost_of_carry:3f}')
```

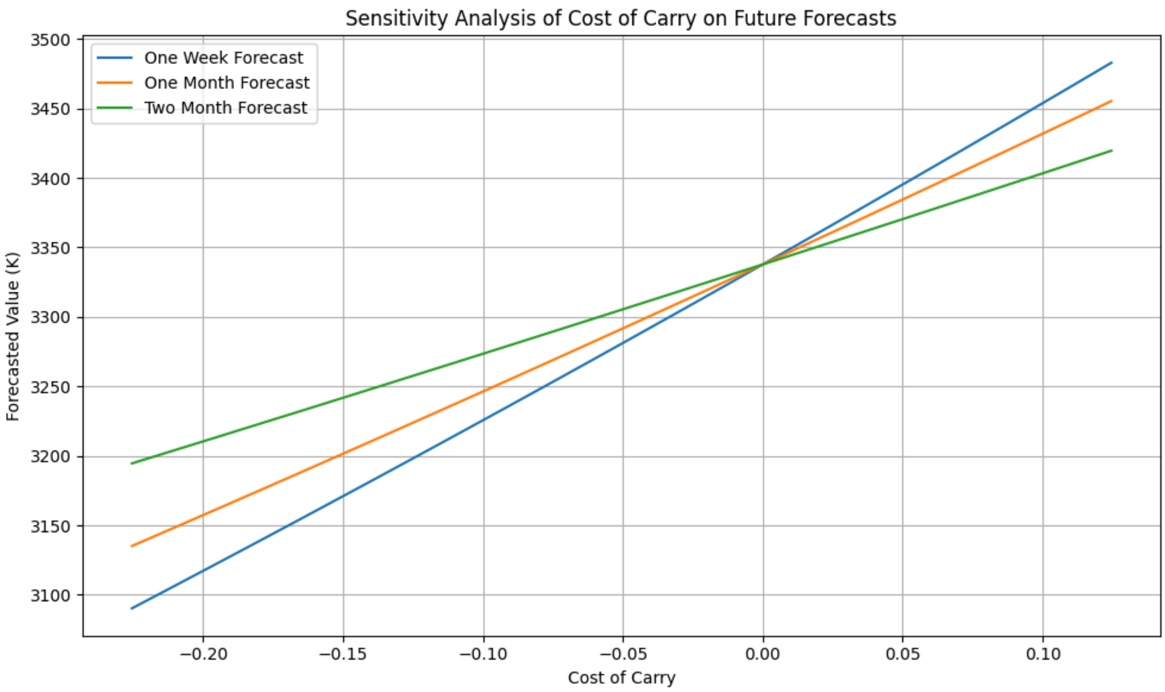
```
K: 3367.5
S: 3337.75
T: 0.3611111111111111
Cost of Carry: 0.024573
```

In [ ]:
```
min_value = cost_of_carry - 0.25
max_value = cost_of_carry + 0.1

T_after_one_week = ((esm20_maturity_as_date - eval_date_as_date).days - 7) / 360
T_after_one_month = ((esm20_maturity_as_date - eval_date_as_date).days - 30) / 3
T_after_two_months = ((esm20_maturity_as_date - eval_date_as_date).days - 60) /

sensitivity_values = np.linspace(min_value, max_value, 50)
df_sensitivity = pd.DataFrame(sensitivity_values, columns=['Cost Of Carry'])
df_sensitivity['K One Week Forecast'] = S * np.exp(df_sensitivity['Cost Of Carry
df_sensitivity['K One Month Forecast'] = S * np.exp(df_sensitivity['Cost Of Carr
df_sensitivity['K Two Month Forecast'] = S * np.exp(df_sensitivity['Cost Of Carr

df_sensitivity.head()
```

Out[ ]:

	Cost Of Carry	K One Week Forecast	K One Month Forecast	K Two Month Forecast
0	-0.225427	3090.324643	3135.154350	3194.606642
1	-0.218284	3097.875717	3141.381070	3199.046678
2	-0.211141	3105.445242	3147.620157	3203.492885
3	-0.203998	3113.033263	3153.871634	3207.945272
4	-0.196855	3120.639824	3160.135528	3212.403847

In [ ]:
```python
plt.figure(figsize=(10, 6))
plt.plot(df_sensitivity['Cost Of Carry'], df_sensitivity['K One Week Forecast'],
plt.plot(df_sensitivity['Cost Of Carry'], df_sensitivity['K One Month Forecast']
plt.plot(df_sensitivity['Cost Of Carry'], df_sensitivity['K Two Month Forecast']
plt.title('Sensitivity Analysis of Cost of Carry on Future Forecasts')
plt.xlabel('Cost of Carry')
plt.ylabel('Forecasted Value (K)')
plt.legend()
plt.grid(True)


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



3.5 Acknowledgements and Tooling

- Various tools, including GitHub, GitHub Copilot, and ChatGPT, were utilised in the development and analysis of this project.
- Portions of the code were adapted from examples provided in lectures.