COMP0164_Group17_Coursework

December 12, 2024

COMP0164: Digital Finance

Group Coursework

If you have set up an **Anaconda** environment, don't forget to activate it using:

conda activate digital_finance_env

Below are the imports that you will need for the coursework:

[1]:
```
import datetime
import numpy as np
import pandas as pd
import scipy as sp
import yfinance as yf
import numpy_financial as npf
import matplotlib.pyplot as plt
%matplotlib inline
```

1 Question 1: Corporate Finance [15 marks]

In this question, you are asked to evaluate business and investment ideas.

You are tasked with evaluating an investment project proposed by a company called Boogle. During the latest earnings report, Boogle's CFO announced an investment of $150 million for a new business expansion project. The project is planned to be financed with an $100 million public offering of a 10-year debt and the remainder with an equity offering. You have collected the information necessary to evaluate this project in Exhibits 1 and 2.

Exhibit 1: Relevant Information for Analysis:

Equity risk premium	4.53%
Risk-free rate of interest	3.9%
Market value of Boogle's debt	$1.0 billion
Market value of Boogle's equity	$2.6 billion
Boogle's equity beta	1.4
Boogle's before-tax cost of debt	9.2%
Corporate tax rate	37.5%

Exhibit 2: Estimated Project Financials:

	Year 1	Year 2	Year 3
Revenue	98.7	110.3	112.6
Operating Costs	32	36	38
Depreciation	16	16	16

a.) Calculate the after-tax weighted average cost of capital of Boogle prior to its new project investment. **[3 marks]**

Expected return on equity portion r_E can be calculated by:

$$r_E = r_f + \beta_E(E[r_m] - r_f)$$

where:

- r_f is the risk-free rate of interest
- β_E is the equity beta
- $E[r_m] - r_f$ can be calculated as the equity risk premium

The after-tax weighted average cost of capital (WACC) can be calculated as:

$$WACC = r_D(1 - T_c)\frac{D}{D+E} + r_E\frac{E}{D+E}$$

where:

- r_D is the before-tax cost of debt
- T_c is the corporate tax rate
- D is the market value of Boogle's debt
- E is the market value of Boogle's equity
- r_E is the expected return on equity

[2]:
```
equity_risk_premium = 4.53 / 100     # Equity risk premium
risk_free_rate = 3.9 / 100           # Risk-free rate of interest
market_value_debt = 10**9            # Market value of Boogle's debt
market_value_equity = 2.6 * 10**9    # Market value of Boogle's equity
beta_equity = 1.4                    # Boogle's equity beta
expected_return_debt = 9.2 / 100     # Before-tax cost of debt
corporate_tax_rate = 0.375           # Corporate tax rate
```

[3]:
```
def calculate_expected_return_equity(risk_free_rate, equity_beta,␣
 ↪equity_risk_premium):
    return risk_free_rate + equity_beta * equity_risk_premium

def after_tax_WACC(r_d, r_e, t_c, d, e):
    return r_d * (1 - t_c) * d / (d + e) + r_e * e / (d + e)

expected_return_equity = calculate_expected_return_equity(risk_free_rate,␣
 ↪beta_equity, equity_risk_premium)
```

```
wacc = after_tax_WACC(expected_return_debt, expected_return_equity,␣
↪corporate_tax_rate, market_value_debt, market_value_equity)
print(f"The after-tax weighted average cost of capital of Boogle is: {wacc:.
↪2%}")
```

```
The after-tax weighted average cost of capital of Boogle is: 8.99%
```

b.) Find Boogle's after-tax asset beta prior to the new project. [**2 marks**]

The following formula has been used for this question:

$$\beta_{asset} = \beta_e \cdot \frac{1}{\left[1 + (1-t) \cdot \frac{D}{E}\right]}$$

Where:

- β_e is the equity beta
- β_{asset} is the asset beta
- D is the debt
- E is the equity
- t is the tax rate

Substituting this:

$$\beta_{asset} = 1.4 \cdot \frac{1}{\left[1 + (1-0.375) \cdot \frac{1.0 \cdot 10^9}{2.6 \cdot 10^9}\right]}$$

$$\beta_{asset} = 1.1287$$

```
[4]: beta_e = 1.4
debt = 1.0*10**9
equity = 2.6*10**9
tax_rate = 0.375

def find_beta_a(beta_e, debt, equity, tax_rate):
    return beta_e * (1/(1 + debt*(1-tax_rate)/equity))

beta_a = find_beta_a(beta_e, debt, equity, tax_rate)

print(f"The after-tax asset beta prior to the new project is: {beta_a:.5}")
```

```
The after-tax asset beta prior to the new project is: 1.1287
```

c.) Assuming the new project has the same asset beta as the Boogle company in b.), find the the project equity beta. [**2 marks**]

The project equity beta \$ _{pe} \$ can be calculated from:

$$\beta_{pe} = \beta_{asset} \cdot \left[1 + (1-t) \cdot \frac{D_p}{E_p}\right]$$

where:

- D_p is the debt of the project
- E_p is the equity of the project
- t is the tax rate
- β_{asset} is the asset beta

Therefore, we have:

$$\beta_{pe} = 1.12868 \cdot \left[1 + (1 - 37.5\%) \cdot \frac{100}{50}\right] = 1.12868 \cdot 2.25 = 2.53953$$

[5]:
```
def find_beta_pe(beta_a, t, d_p, e_p):
    # Calculate the equity beta (Beta_E)
    return beta_a * (1 + (1 - t) * d_p / e_p)
# Total investment
total = 150 * 10**6
# Debt portion
d_p = 100 * 10**6
# Equity portion calculated as total investment minus debt
e_p = total - d_p
# Corporate tax rate (37.5%)
tax_rate = 0.375
# Calculate project equity beta
project_beta_pe = find_beta_pe(beta_a, tax_rate, d_p, e_p)

print(f"Project Equity Beta: {project_beta_pe:.5f}")
```

```
Project Equity Beta: 2.53953
```

d.) The formula for project after-tax free cash flow at time t is

$$FCF = (\text{Revenue} - \text{Operating Costs} - \text{Depreciation}) \times (1 - \text{Tax Rate}) + \text{Depreciation}.$$

Define a Python function to calculate the project FCFs and demonstrate that the after-tax free cash flow generated for the next three years are \$47.7 million, \$52.4 million, and \$52.6 million respectively. **[2 marks]**

[6]:
```
#Revenues are 98.7, 110.3, 112.6
#Operating Costs are 32, 36, 38
#Depreciation is 16, 16, 16
#Tax Rate is 0.375
#The initial investment is 150 million
#FCF = (Revenues - Operating Costs - Depreciation) * (1 - Tax Rate) +␣
 ↪Depreciation
#The Values from required are stored in the lists below
revenues = [98.7, 110.3, 112.6]
operating_costs = [32, 36, 38]
depreciation = [16, 16, 16]
```

```
# Append FCFs for NPV and IRR calculation
FCFs = [-150]
def FCF(revenue, operating_cost, depreciation, tax_rate):
    # Calculate the Free Cash Flow using the formula
    FCF = (revenue-operating_cost-depreciation) * (1-tax_rate)+depreciation
    return FCF #The function returns the Free Cash Flow
#The for loop is used to calculate the Free Cash Flow for each year
for i in range(len(revenues)):
    #The FCF function is called to calculate the Free Cash Flow for each year
    fcf= FCF(revenues[i], operating_costs[i], depreciation[i], 0.375)
    #The calculated FCF is appended to the FCFs list
    FCFs.append(fcf)
    #The FCF for each year is printed
    print("The FCF for year {i} is ${fcf:.1f} million".format(i=i+1, fcf=fcf))
```

```
The FCF for year 1 is $47.7 million
The FCF for year 2 is $52.4 million
The FCF for year 3 is $52.6 million
```

e.) Find the project NPV and IRR with the next three years after-tax free cash flow given in d.).[**3 marks**]

The NPV formula:

$$NPV = \sum_{t=0}^{n} \frac{FCF_t}{(1+r)^t} - C_0$$

Where:

- C_0 : the initial investment

The IRR formula:

$$0 = \sum_{t=0}^{n} \frac{FCF_t}{(1+IRR)^t}$$

[7]:
```
# Calculate the Net Present Value (NPV) using npf.npv()
npv = npf.npv(wacc, FCFs)
# Calculate the IRR using npf.irr()
irr = npf.irr(FCFs)

print("NPV of the project is: ${:.2f} million".format(npv))
print("Project IRR: {:.3%}".format(irr))
```

```
NPV of the project is: $-21.46 million
Project IRR: 0.899%
```

Assume the following is a dividend-paying stock.

1.0.1 Stock:

Atat Steel is in the steel manufacturing sector with a required rate of return of 7.35%. You estimate that if the economy is booming, the company's current annual dividend of \$0.8 per share will grow 10.5% a year for the next four years and then stabilize at a 3.5% growth rate a year indefinitely. However, if the economy falls into a recession, then Atat Steel will not likely experience the elevated 10.5% short-run growth and instead will grow by 3.5% indefinitely.

f.) Use the discount dividend method and find the current value of Atat Steel stock under both economic conditions. [**3 marks**]

The **booming economy scenario**: in order to find the dividened for the first 4 years we use Multi - Staged Dividened Discount Model with the formula

$$P_0 = \sum_{t=1}^{H} \frac{D_t}{(1+r)^t} + \frac{D_H(1+g_2)}{(r-g_2)(1+r)^H}$$

$$Dt = D(t-1) \times (1+g_1)^t$$

Where:

- P_0 : Current price of the stock
- D_0 : Current dividend
- g_1 : Short-term growth rate
- g_2: Long-term growth rate
- r: Required rate of return
- H : Number of boom years
- D_t : Dividened

The the **recession** scenario: in order to find the dividened the Constant Growth Dividened Method is used as demonstrated below.

$$P_0 = \frac{D_0 \times (1+g_2)}{(r-g_2)}$$

Where:

- P_0: Current price of the stock
- D_0: Current dividend
- g_2: Constant growth rate
- r: Required rate of return

[8]:
```
#Initialise the parameters for both scenarios
first_dividend = 0.8
required_ror = 0.0735
growth_rate = 0.035

#Function to find the dividends for the next year when there is growth
def Next_year_dividend(dividend, growth_rate):
    return dividend*(1+growth_rate)
```

```python
#Function to find the perpetual dividend
def perpetual_dividend(dividend, growth_rate, required_ror):
    return dividend*(1+growth_rate)/(required_ror-growth_rate)

#Function to find the dividends for each year during the growth period
def Dividends_growth_period(first_dividend, boom_years):
    boom_dividends = []
    for i in range(boom_years):
        #The first year dividend is calculated using the formula:
        #Dividend * (1 + g)
        if i == 0:
            next_year_dividend = Next_year_dividend(first_dividend,growth_rate_boom)
        #The subsequent year dividends are calculated using the formula:
        #Previous year Dividend * (1 + g)^the current year
        else:
            next_year_dividend = Next_year_dividend(boom_dividends[-1],growth_rate_boom)
        boom_dividends.append(next_year_dividend)
    return boom_dividends

#Function to find the NPV of the dividends during the growth period
def NPV_dividends_growth(boom_dividends, required_ror):
    boom_dividends_npv = 0
    for i in range(len(boom_dividends)):
        #The NPV of the dividends is calculated using the formula:
        #Dividend of year t / (1 + r) ^ t
        boom_dividends_npv += boom_dividends[i]/((1+required_ror)**(i+1))
    return boom_dividends_npv

#This section is for the Booming economy:

#Initialise the parameters for the booming economy
years_boom = 4
growth_rate_boom = 0.105

#Calculate the dividends for each year during the booming economy
boom_dividends = Dividends_growth_period(first_dividend,years_boom)

#Find the NPV for the first 4 years of the booming economy:
boom_dividends_npv = NPV_dividends_growth(boom_dividends,required_ror)

#Find the perpetual growth rate for the dividends during the booming economy
perpetual_boom = perpetual_dividend(boom_dividends[-1],growth_rate,required_ror)

# Calculate NPV of perpetual growth dividends during booming economy
perpetual_boom_npv = perpetual_boom/((1+required_ror)**years_boom)
```

```python
#Calculate the terminal value of the company during the booming economy
P0_boom = boom_dividends_npv + perpetual_boom_npv
print(f"Booming economy stock price: ${P0_boom:.2f}")

#This section is for the Recession economy:
P0_recession = perpetual_dividend(first_dividend,growth_rate,required_ror)
print(f"Recession economy stock price: ${P0_recession:.2f}")
```

```
Booming economy stock price: $27.59
Recession economy stock price: $21.51
```

2 Question 2: Bonds and Fixed Income [16 marks]

In this question, you are given background information on some bonds and you tasked with calculating crucial data about them.

Exhibit 3: Current Par Rates and Spot Rates:

Maturity	Par Rate (Annual)	Spot Rate (Annual)
1 year	2.50%	2.50%
2 years	3.02%	3.00%
3 years	3.53%	3.50%
4 years	3.98%	4.00%
5 years	4.42%	?

Exhibit 4: Information for Selected Bonds:

Bond Name	Maturity	Coupon (Annual)	Type of Bond
Bond A (Face value $1,000)	3 years	6.78%	Option-free
Bond B	3 years	4.30%	Callable at par on start of year 1 and tear 2
Bond C	3 years	4.30%	Putable at par on start of year 1 and year 2

Exhibit 5: Binomial Interest Rate Tree, based on an estimated interest rate volatility of 10%, where 'u' represents an up move and 'd' represents a down move:

Year 0	Year 1	Year 2
2.3400% (r)	3.5930% (ru node)	4.6540% (ruu node)
	2.9417% (rd node)	3.8043% (rud node)
		3.1254% (rdd node)

a.) Based on Exhibit 3, find the five-year spot rate. **[2 marks]**

To calculate the five-year spot rate, we use the following formula

$$P = \sum_{t=1}^{5} \frac{CF_t}{(1+S_t)^t} + \frac{P}{(1+S_5)^5}$$

Where:

- P: Price of the bond (assume $P = 100 $ for par value)
- CF_t: Annual coupon payment $C = 100 \times \text{Par Rate}$

- S_t: Known spot rates for $(t = 1, 2, 3, 4)$
- S_5: The unknown spot rate for 5 years

Steps:

$$P = \frac{CF_1}{(1+S_1)} + \frac{CF_2}{(1+S_2)^2} + \frac{CF_3}{(1+S_3)^3} + \frac{CF_4}{(1+S_4)^4} + \frac{CF_5 + P}{(1+S_5)^5}$$

$$100 = \frac{CF_1}{(1+S_1)} + \frac{CF_2}{(1+S_2)^2} + \frac{CF_3}{(1+S_3)^3} + \frac{CF_4}{(1+S_4)^4} + \frac{CF_5 + 100}{(1+S_5)^5}$$

Where:

- CF: $0.0442 \times\ 100$

$$100 = \frac{(4.42)}{(1+0.025)} + \frac{(4.42)}{(1+0.03)^2} + \frac{(4.42)}{(1+0.035)^3} + \frac{(4.42)}{(1+0.04)^4} + \frac{(4.42+100)}{(1+S_5)^5}$$

$$(1+S_5)^5 = 1.247$$

[9]:
```
#The parameters for the calculation are defined
spot_rate_list = [0.025, 0.03, 0.035, 0.04]

par_rate = 0.0442
#Assuming a face value of $100 to calculate the five-year spot rate
p = 100

def find_spot_rate(p, par_rate, spot_rate_list):
    #The spot rate for each year is calculated using the formula:
    #(p*par) / (1 + spot_rate) ^ t
    total_spot_rate_vals = 0
    for i in range(len(spot_rate_list)):
        total_spot_rate_vals += (p * par_rate)/((1 + spot_rate_list[i])**(i+1))
    #The formula (1+S5)^5 = (100+4.42)/(100 - 16.24329032372646)
    spot_rate_val_power = (p + (par_rate * p))/(p - total_spot_rate_vals)
    print(spot_rate_val_power)
    #The spot rate for the 5th year is calculated using the formula:
    #S_5 = ((100 + 4.42)/(100 - total_spot_rate_vals)) ^ (1/5) - 1
    spot_rate = spot_rate_val_power**(1/(len(spot_rate_list)+1)) - 1

    return spot_rate

five_year_spot_rate = find_spot_rate(p, par_rate, spot_rate_list)
print("The five-year spot rate is: {:.2%}".format(five_year_spot_rate))
```

```
1.2467060896206614
The five-year spot rate is: 4.51%
```

b.) Assuming the law of one price, use Exhibit 3 to calculate the compounded forward rate (annualy compounded) of a one-year loan starting in three years. **[1 mark]**

$$f_{(34)} = \frac{(1+S_4)^4}{(1+S_3)^3} - 1$$

Where:

- S_4: 4.00%
- S_3: 3.50%

$$f_{(34)} = \frac{(1+0.04)^4}{(1+0.035)^3} - 1$$

[10]:
```
#The parameters for the calculation are defined
spot_rate_year_3 = 0.035
spot_rate_year_4 = 0.04

#The spot rate for the 3rd year is 3.5%
#and the spot rate for the 4th year is 4% based on Exhibit 3
def find_forward_rate(spot_rate_year_3, spot_rate_year_4):
    #The forward rate for the 4th year is calculated using the formula:
    # (1 + spot_rate_year_4) ^ 4 / (1 + spot_rate_year_3) ^ 3 - 1
    forward_rate = ((1 + spot_rate_year_4) ** 4 / (1 + spot_rate_year_3)**3)-1
    return forward_rate

forward_rate = find_forward_rate(spot_rate_year_3, spot_rate_year_4)
print("The 1-year forward rate in 3 years is: {:.4%}".format(forward_rate))
```

```
The 1-year forward rate in 3 years is: 5.5145%
```

c.) Given spot rates for one-, two-, and three-year zero bonds, how many forward rates can be calculated? Please list the compounded forward rates (annualy compounded) that can be calculated and briefly explain your answer. [**3 marks**]

Forward Rate Formula:

$$f_{T1,T2} = \frac{(1+S_2)^{T2}}{(1+S_1)^{T1}} - 1$$

Steps to Calculate Forward Rates: 1. Forward rate for a 1-year loan starting in year 1 (in 1 year) given spot rates for year 1 and year 2 (S_1 and S_2):

$$f_{1,2} = \frac{(1+S_2)^2}{(1+S_1)} - 1$$

2. Forward rate for a 1-year loan starting in year 2 (in 2 years) given spot rates for year 2 and year 3 (S_2 and S_3):

$$f_{2,3} = \frac{(1+S_3)^3}{(1+S_2)^2} - 1$$

3. Forward rate for a 2-year loan starting in year 1 (in 1 year) given spot rates for year 1 and year 3 (S_1 and S_3):

$$f_{1,3} = (\frac{(1+S_3)^3}{(1+S_1)})^{\frac{1}{2}} - 1$$

d.) Find the yield to maturity for Bond A. [**4 marks**]

In order to calculate yield to maturity of for Bond A, the following formula is used:

$$PV = \sum_{t=1}^{n} \frac{C}{(1+y)^t} + \frac{P}{(1+y)^n}$$

Where

- PV : Present value or price of the bon
- P : Face value of the bond (In the case of Bond A = \$1000)
- C : Annual coupon payments (in the case of Bond A: $1000 \times 6.78 = \$67.8$)
- y : Yield to maturity of the bond
- n : Number of periods to maturity (In case of Bond A this is 3 years)

$$1093.15 = \frac{67.8}{(1+0.025)} + \frac{67.8}{(1+0.03)^2} + \frac{67.8}{(1+0.035)^3} + \frac{1000}{(1+0.035)^3}$$

$$1093.15 = \frac{67.8}{(1+y)} + \frac{67.8}{(1+y)^2} + \frac{67.8}{(1+y)^3} + \frac{1000}{(1+y)^3}$$

[11]:
```python
def find_ytm(price, face_value, coupon_rate, maturity):

    # Calculate the annual coupon payment
    coupon = face_value * coupon_rate

    # Construct the cash flow array:
    # Initial outflow is negative price
    # For each year until the last, just the coupon
    # On the last year, coupon plus face value
    cash_flows = [-price] + [coupon]*(maturity-1) + [coupon + face_value]

    # Use npf.irr to compute the IRR, which corresponds to the YTM
    ytm = npf.irr(cash_flows)

    return ytm

def find_PV(face_val, coupon_rate, spot_rates, maturity):

    # Calculate the annual coupon payment
    coupon = face_val * coupon_rate

    # Construct the cash flow array
    cash_flows = [coupon] * (maturity - 1) + [coupon + face_val]
```

```
    # Compute the present value by discounting each cash flow
    PV = 0
    for t, cash_flow in enumerate(cash_flows):
        PV += cash_flow / ((1 + spot_rates[t]) ** (t + 1))

    return PV

# Example usage:
face_val = 1000
coupon_rate = 0.0678
maturity = 3
spot_rate_lst = [0.025, 0.03, 0.035]

price = find_PV(face_val, coupon_rate, spot_rate_lst, maturity)
print(f"Price of Bond A using spot rates: {price:.2f}")

ytm = find_ytm(price, face_val, coupon_rate, maturity)
print(f"The Yield to Maturity is: {ytm * 100:.2f}%")
```

```
Price of Bond A using spot rates: 1093.15
The Yield to Maturity is: 3.46%
```

e.) Based on Exhibit 5, assume an equal probability of interest rate going up and down at each node. Calculate the value of Bond B and Bond C with the binomial tree model. [**3 marks**]

Values acheived for realisations is presented in this table:

Realisation	P_0	P_1	P_2
uu, u	1089.79	1030.37	1014.02
ud, u	1094.68	1035.38	1019.15
ud, d	1099.51	1040.13	1019.15
dd, d	1103.57	1044.28	1023.39

[12]:
```
# Compute up and down factors based on volatility and time step
sigma = 0.10  # Interest rate volatility
delta_t = 1   # Time step in years
u = np.exp(sigma * np.sqrt(delta_t))  # Up factor
d = 1 / u                                         # Down factor

# Equal probabilities for up and down moves
p = 0.5

# Tree rates dynamically computed
r0 = 0.0234
ru = r0 * u
rd = r0 * d
ruu = ru * u
```

```python
rud = ru * d
rdd = rd * d

tree_rates = {
    'ruu': ruu,
    'rud': rud,
    'rdd': rdd,
    'ru': ru,
    'rd': rd,
    'r0': r0
}

face_value = 1000
coupon_rate = 0.043
coupon_payment = coupon_rate * face_value
total_cashflow = face_value + coupon_payment

# Function to compute cashflows and present values
def compute_present_values(face_value, coupon, r2, r1, r0):
    # Total cashflow at maturity
    total_cf = face_value + coupon

    # Backward induction for PV calculations
    P2 = total_cf / (1 + r2)
    P1 = P2 / (1 + r1) + coupon / (1 + r1)
    P0 = P1 / (1 + r0) + coupon / (1 + r0) + coupon / (1 + r1) / (1 + r0)

    return P0, P1, P2

# Adjust for Callable and Putable Bonds
def adjust_for_callable(value, call_price):
    return min(value, call_price)

def adjust_for_putable(value, put_price):
    return max(value, put_price)

# Compute Realisations
realisation_results = {}
for scenario, rates in [("Realisation 1 (uu, u)", ('ruu', 'ru', 'r0')),
                        ("Realisation 2 (ud, u)", ('rud', 'ru', 'r0')),
                        ("Realisation 3 (ud, d)", ('rud', 'rd', 'r0')),
                        ("Realisation 4 (dd, d)", ('rdd', 'rd', 'r0'))]:
    r2, r1, r0 = (tree_rates[r] for r in rates)
    P0, P1, P2 = compute_present_values(face_value, coupon_payment, r2, r1, r0)
    realisation_results[scenario] = (P0, P1, P2)

# Print Realisations
```

```python
for scenario, values in realisation_results.items():
    print(f"{scenario}: P0 = {values[0]:.4f}, P1 = {values[1]:.4f}, P2 =
 ↪{values[2]:.4f}")

# Value of Bond B (Callable)
V_b = adjust_for_callable((face_value + coupon_payment) / (1 +
 ↪tree_rates['r0']), face_value)
print(f"\nThe Value of Bond B (Callable) is: {V_b:.3f}")

# Adjust for Putable Bond
print("\nAdjusting cashflows for Bond C (Putable):")
adjusted_values = {}
for scenario, (P0, P1, P2) in realisation_results.items():
    if scenario == "Realisation 1 (uu, u)":
        # Adjust cashflows for putable feature
        P1_adj = adjust_for_putable(total_cashflow / (1 + tree_rates['ru']) /
 ↪(1 + tree_rates['r0']) + coupon_payment / (1 + tree_rates['r0']), face_value)
        adjusted_values[scenario] = P1_adj
    else:
        adjusted_values[scenario] = P0

# Value of Bond C (Putable)
V_c = sum(adjusted_values.values()) / len(adjusted_values)
print(f"The Value of Bond C (Putable) is: {V_c:.3f}")
```

```
Realisation 1 (uu, u): P0 = 1089.7871, P1 = 1030.3721, P2 = 1014.0185
Realisation 2 (ud, u): P0 = 1094.6766, P1 = 1035.3760, P2 = 1019.1518
Realisation 3 (ud, d): P0 = 1099.5089, P1 = 1040.1290, P2 = 1019.1518
Realisation 4 (dd, d): P0 = 1103.5677, P1 = 1044.2827, P2 = 1023.3935

The Value of Bond B (Callable) is: 1000.000

Adjusting cashflows for Bond C (Putable):
The Value of Bond C (Putable) is: 1083.307
```

f.) All else being equal, explain the effect of a fall in interest rates on Bond B and Bond C. [**2 marks**]

2.0.1 Effect of a Fall in Interest Rates on Bond B and Bond C

Bond prices have an inverse relationship with interest rates; as interest rates fall, bond prices typically rise.

For Bond B (Callable): In all realisations, the price of Bond B exceeds the par value ($1000) at the start of Year 1. Hence, it will be called back at par value regardless of changes in interest rates. However, before it is called, a fall in interest rates will temporarily increase its market value, reflecting its sensitivity to rate movements until the callable feature caps its price growth.

For Bond C (Putable): The price of Bond C benefits significantly from falling interest rates, as seen in the adjusted cashflows where the bond's price exceeds the put price in all scenarios. In Realisation 1, for example, the bondholder achieves a higher return by holding the bond rather than exercising the put option. This demonstrates that the putable feature provides downside protection in rising rate environments while allowing full participation in price appreciation when rates decline, ultimately increasing the bondholder's gains.

g.) All else being equal, which bond is most likely to increase in value if interest rate volatility is 15% rather than 10%? Briefly explain your answer. (Hint: consider the value of options) [**1 mark**]

Bond C, the putable bond, is more likely to increase in value if interest rate volatility rises from 10% to 15%. This is because the value of embedded options, whether call or put, increases with higher volatility. Increased volatility raises the likelihood that the bond's market price will fall below the put price, enhancing the value of the put option for the bondholder.

While Bond B, the callable bond, also sees an increase in value with higher volatility, its price is capped due to the callable feature, limiting its upside potential. Additionally, higher volatility increases the likelihood of the issuer calling Bond B to avoid further losses, reducing its value from the bondholder's perspective. Therefore, Bond C is more sensitive to increased volatility and is likelier to experience a more significant value increase than Bond B.

3 Question 3: Forward and Futures Contracts [12 marks]

In this question you will be working with futures and forward contracts

Consider a stock that is trading at $100 today. The stock does not generate income/pay dividends. The stock is traded in a well-functioning market with no transaction costs and no restrictions on short sales. Both borrowing and lending can be done in unlimited amounts at the 2% risk-free rate.

a.) What is the difference between forward contracts and futures contracts (Answers should be no longer than 200 words) [**4 marks**]

Answer:

Forward contracts are agreements between two parties to buy or sell an asset at a future date for a price agreed upon today. This is an over-the-counter agreement so they are privately negotiated between two parties and the contract terms are customised to fit the needs. Since each contract is tailor-made, it's not standardised, giving flexibility but less liquidity. This comes with higher credit risk because there is no intermediary to guarantee the transaction. The forward contracts are mainly used for hedging.

Future contracts are standardised agreements traded on an exchange. It is marked-to-market daily, meaning gains and losses are settled daily so the credit risk is reduced. The terms of this contract are set by an exchange and the contract is fully standardised, ensuring high liquidity and ease of trading. The credit risk is reduced as mentioned previously, but also because there is a margin system in place to manage risk. Future contracts can be used for both hedging and speculative purposes.

b.) Consider a futures contract on the stock with a maturity of one year. Suppose that the futures price is currently at $97. Are the futures fairly priced? Describe an arbitrage strategy that would allow you to make a riskless profit. [**2 marks**]

No this is not fairly priced, as the future price can be expected as approximately $102. Here is the calculation.

To calculate the future value, the following formula can be used:

$$F = S \cdot e^{rT}$$

where (S) is the current price of the asset, (r) is the risk-free interest rate and (T) is the time to maturity in years.

$$F = 100 \cdot e^{0.02 \cdot 1} = 102.02$$

The expected future price is **$102.02**.

To make a riskless profit:

1. Buy the futures contract at $97, meaning we agree to buy the stock after a year for $97.
2. Short-sell the stock today at the spot price, which in this case $100.
3. Invest the proceeds ($100) from the short sale at the risk-free rate.

This will gain the total riskless profit of approximately \$5 per share. This is possible by using the \$102 which gained from investing $100atthe2\%ofrisk - freerateandspendingthatmoneytobuythestockat\$97.Therefore$:\$ 102.02 - 97 = 5.02 \$\$

The riskless profit will be **\$5.02 per share**.

c.) Same as question b) but suppose that the futures price is currently at \$103. Describe your arbitrage strategy. [**2 marks**]

Answer here:

No this is not fairly priced also since the expected future price will be approximately \$102, as calculated above.

To gain a riskless profit, this can be an arbitrage strategy:

1. Sell the futures contract at \$103, meaning we agree to sell the stock in a year at \$103.
2. Buy the stock today at the spot price, which in this case \$100.
3. Borrow money at the risk-free rate 2% to finance the stock purchase.

In this way, the total gain or loss after a year can be calculated as follows:

- Borrow \$100 today at 2% for one year. The repayment amount after one year is:

$$100 \cdot e^{0.02 \cdot 1} = 102.02$$

- At maturity:
  - Deliver the stock as per the futures contract to receive \$103.
  - Use \$102.02 to repay the loan.

The remaining profit is:

$$103 - 102.02 = 0.98$$

Therefore, the riskless profit will be **\$0.98 per share**.

d.) Consider a stock with the market value of \$76 that pays 5% dividend yield over the next year. Given the risk-free rate of return is 3%, and the agreed delivery price is \$73, what is the present value of the forward contract with one year maturity? [**2 marks**]

The following formula can be used in this situation:

The formula to calculate the future value is:

$$F = S \cdot e^{(r-q)T}$$

Where:

- S: Current stock price 76
- r: Risk-free interest rate 3%
- q: Dividend yield 5%
- T: Time to maturity in years 1

Substituting these to obtain:

$$F = 76 \cdot e^{(0.03-0.05)\cdot 1}$$
$$F = 74.49$$

Then, the following formula has been used to compute the present value:

$$PV = (F - K) \cdot e^{-rT}$$

Where:

- F: Forward price
- K: Delivery price
- r: Risk-free interest rate
- T: Time to maturity in years

Substituting these to obtain:

$$PV = (74.49 - 73) \cdot e^{-0.03\cdot 1}$$
$$PV = 1.445$$

[13]:
```
spot_price = 76
risk_free_rate = 0.03
dividend_yield = 0.05
time_to_maturity = 1
delivery_price = 73

def calculate_forward_price(spot_price, risk_free_rate, dividend_yield,␣
 ↪time_to_maturity):
    forward_price = spot_price * np.
 ↪exp((risk_free_rate-dividend_yield)*time_to_maturity)
    return forward_price

forward_price = calculate_forward_price(spot_price, risk_free_rate,␣
 ↪dividend_yield, time_to_maturity)

def calculate_pv_with_divident(forward_price, delivery_price, time_to_maturity):
    pv = (forward_price - delivery_price) * np.exp(-risk_free_rate *␣
 ↪time_to_maturity)
    return pv

print("The present value of the forward contract with one year maturity is:␣
 ↪${pv:.2f}".format(pv=calculate_pv_with_divident(forward_price,␣
 ↪delivery_price, time_to_maturity)))
```

```
The present value of the forward contract with one year maturity is: $1.45
```

e.) Consider the same conditions as in part d.), but with the dividend of $3, what is the present value of the forward contract? [**2 marks**]

Almost the same as the previous question, but in this question instead of provision of divident yield, the actual divident has been given so we decided to use them directly.

The formula to calculate the future value is:

$$F = (S - D) \cdot e^{rT}$$

Where:

- S: Current stock price
- D: Dividend
- r: Risk-free interest rate
- q: Dividend yield
- T: Time to maturity in years

Substituting these to obtain:

$$F = (76 - 3) \cdot e^{0.03 \cdot 1}$$
$$F = 75.22$$

Then, the following formula has been used to compute the present value:

$$PV = (F - K) \cdot e^{-rT}$$

Where:

- F: Forward price
- K: Delivery price
- r: Risk-free interest rate
- T: Time to maturity in years

Substituting these to obtain:

$$PV = (75.22 - 73) \cdot e^{-0.03 \cdot 1}$$
$$PV = 2.16$$

[14]:
```
# To consider the dividend, we need to substract from the spot price
dividend = 3
spot_price = 76
risk_free_rate = 0.03
dividend_yield = 0  # Since this is already given as a numerical value␣
↪(discrete dividend)
time_to_maturity = 1
delivery_price = 73
```

```
updated_spot_price = spot_price - dividend

forward_price2 = calculate_forward_price(updated_spot_price, risk_free_rate,␣
 ↪dividend_yield, time_to_maturity)
print("The present value of the forward contract with one year maturity is:␣
 ↪${pv:.2f}".format(pv=calculate_pv_with_divident(forward_price2,␣
 ↪delivery_price, time_to_maturity)))
```

```
The present value of the forward contract with one year maturity is: $2.16
```

4 Question 4: Options [16 Marks]

In this question, you will be working with options.

Suppose that you hold a long position on a European call option that has an underlying asset price of $45.34, strike price of $44, risk-free rate of 1.5%, 20% annualized volatility, and a half-a-year left to maturity. The underlying asset does not have any investment yield.

a.) Value this call option. **[1 mark]**

To estimate the price of a European call option, the Black-Scholes-Merton model provides a closed-form solution. Below are the key formulas used in the model:

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} = \frac{\ln\left(\frac{45.34}{44}\right) + \left(0.015 + \frac{0.2^2}{2}\right)0.5}{0.2\sqrt{0.5}} = 0.336$$

$$d_2 = d_1 - \sigma\sqrt{T} = 0.336 - 0.2\sqrt{0.5} = 0.2$$

$$N = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} e^{-\frac{1}{2}x^2}dx$$

$$c = S_0N(d_1) - Ke^{-rT}N(d_2) = 45.34N(0.336) - 44e^{-0.015\times0.5}N(0.2) = 3.43$$

Where:

- S_0: Current stock price $45.34
- K: Strike price $44
- r: Risk-free rate 1.5% or 0.015
- σ: Volatility 20% or 0.2
- T: Time to maturity 0.5 years
- $N()$: Cumulative distribution function of the standard normal distribution

[15]:
```
s_0 = 45.34 # Current stock price
s_p = 44    # Strike priceQD
rf = 0.015  # Risk-free rate
sigma = 0.2 # Volatility
t = 0.5     # Time to maturity

def find_option_price(s_0, K, r, sigam, T):
    # Calculate the d1 and d2 values
    d1 = (np.log(s_0/K) + (r + (sigam**2)/2) * T) / (sigam * np.sqrt(T))
    d2 = d1 - sigam * np.sqrt(T)
    # Calculate the call option price
    c = s_0 * sp.stats.norm.cdf(d1) - K * np.exp(-r * T) * sp.stats.norm.cdf(d2)
    return c

c = find_option_price(s_0, s_p, rf, sigma, t)
print(f"The price of the call option is: ${c:.2f}")
```

```
The price of the call option is: $3.43
```

b.) Based on the Black-Scholes-Merton model, describe a portfolio that replicates the call option's payoff. [**1 mark**]

To replicate the payoff of a call option at expiration, we construct a portfolio that mimics its behavior. Using the **Black-Scholes-Merton (BSM)** model, we calculate the key parameters d_1 and d_2 to determine the appropriate portfolio weights. This portfolio consists of three components: a long position in the underlying stock, a long put option with the same strike price K, and a cash position of $-K$ borrowed at the risk-free rate. By combining these components, the portfolio can perfectly replicate the call option's payoff:

- If $S_T \leq K$: 0
- If $S_T > K$: $S_T - K$

Stock:

- When $S_T \leq K$, the value of the stock is S_T.
- When $S_T > K$, the value of the stock remains S_T.

Put Option:

- When $S_T \leq K$, the put option's value is $K - S_T$, representing the payoff from exercising the option.
- When $S_T > K$, the put option's value is 0, as the option is not exercised.

Cash:

- Regardless of whether $S_T \leq K$ or $S_T > K$, the cash value is always $-K$, representing the repayment of the borrowed amount.

The portfolio payoff calculation is as follows:

- If $S_T \leq K$: Portfolio payoff $= S_T + (K - S_T) + (-K) = 0$
- If $S_T > K$: Portfolio payoff $= S_T + 0 + (-K) = S_T - K$

Using the BSM model to compute $N(d_1)$ and $N(d_2)$, the portfolio weights are:

- **Stock**: $S_0 N(d_1)$
- **Cash**: $K e^{-rT} N(d_2)$

Thus, by combining these components, the portfolio's payoff matches the call option's payoff exactly, under all scenarios at expiration.

[16]:
```
# Calculate the value of the replicating portfolio
d1 = (np.log(s_0 / s_p) + (rf + 0.5 * sigma ** 2) * t) / (sigma * np.sqrt(t))
d2 = d1 - sigma * np.sqrt(t)

# Calculate the value of the replicating portfolio
c = s_0 * sp.stats.norm.cdf(d1) - s_p * np.exp(-rf * t) * sp.stats.norm.cdf(d2)

# Calculate the stock position and borrowed cash
stock_part = s_0 * sp.stats.norm.cdf(d1)
borrowed_cash = s_p * np.exp(-rf * t) * sp.stats.norm.cdf(d2)
```

```
# Calculate the value of the replicating portfolio
portfolio_value = stock_part - borrowed_cash
print(f"Call option price: ${c:.2f}")
print(f"Replicating portfolio value: ${portfolio_value:.2f}")
print(f"Stock position: ${stock_part:.2f}")
print(f"Borrowed cash: ${borrowed_cash:.2f}")
```

```
Call option price: $3.43
Replicating portfolio value: $3.43
Stock position: $28.63
Borrowed cash: $25.20
```

The replicating portfolio value $S_0N(d_1) - Ke^{-rT}N(d_2)$ matches the European call option price, confirming that the portfolio payoff exactly replicates the call option's payoff under all expiration scenarios.

c.) Define a function to price the option with the binomial tree method. The function should take the number of steps (n) as one of the inputs. You should NOT use list comprehension in the function. [**3 marks**]

Upward and Downward Movement Factors (u **and** d**):**

$$\Delta t = \frac{T}{n}, \quad u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}} = \frac{1}{u}$$

Risk-neutral Probability (p**):**

$$p = \frac{e^{r\Delta t} - d}{u - d}$$

Stock Price at Each Node ($S_p[i]$**):**

$$S_p[0] = S_0 \cdot d^n, \quad S_p[i] = S_p[i-1] \cdot \frac{u}{d}, \quad i = 1, 2, \dots, n$$

Option Price Recursion ($f^{(j,i)}$**):**

$$f^{(j,i)} = e^{-r\Delta t} \cdot \left(pf^{(j+1,i+1)} + (1-p)f^{(j+1,i)}\right), \quad j = n-1, n-2, \dots, 0, \; i = 0, 1, \dots, j$$

[17]:
```
def binomial_tree(s_0, K, r, sigma, T, n):
    dt = T / n
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp(r * dt) - d) / (u - d)
    discount = np.exp(-r * dt)

    # Initialize the stock price tree
    s_p  =  np.zeros(n+1)
    s_p[0] = s_0 * d**n
    for i in range(1, n+1):
        s_p[i] = s_p[i-1] * u / d
```

```
    # Initialize the option value at maturity
    option = np.zeros(n+1)
    for i in range(n+1):
        option[i] = max(s_p[i] - K, 0)

    # Calculate the option price at each node
    for j in range(n-1, -1, -1):
        for i in range(j+1):
            option[i] = discount * (p * option[i+1] + (1-p) * option[i])

    return option[0]
```

d.) By setting n = 10, 50 and 100, compare and comment on the results under the two methods. **[2 marks]**

$$\Delta t = \frac{T}{n} = \frac{0.5}{n}, \quad u = e^{\sigma\sqrt{\Delta t}} = e^{0.2\sqrt{\frac{0.5}{n}}}, \quad d = e^{-\sigma\sqrt{\Delta t}} = e^{-0.2\sqrt{\frac{0.5}{n}}} = \frac{1}{u}$$

$$p = \frac{e^{r\Delta t} - d}{u - d} = \frac{e^{0.015\times\frac{0.5}{n}} - e^{-0.2\sqrt{\frac{0.5}{n}}}}{e^{0.2\sqrt{\frac{0.5}{n}}} - e^{-0.2\sqrt{\frac{0.5}{n}}}}$$

$$S_p[0] = S_0 \cdot d^n = 45.34 \cdot e^{-n\cdot 0.2\cdot\sqrt{\frac{0.5}{n}}}, \quad S_p[i] = S_p[i-1] \cdot e^{2\cdot 0.2\cdot\sqrt{\frac{0.5}{n}}}, \quad i = 1, 2, \ldots, n$$

$$f^{(j,i)} = e^{-r\Delta t} \cdot \left(pf^{(j+1,i+1)} + (1-p)f^{(j+1,i)}\right), \quad j = n-1, n-2, \ldots, 0, \ i = 0, 1, \ldots, j$$

Where: - n: Number of steps in the binomial tree $n \in (10, 50, 100)$

[18]:
```
print ("The price of the call option using the binomial option pricing model␣
↪(10 steps) is: ${option_price:.4f}".format(option_price=binomial_tree(s_0,␣
↪s_p, rf, sigma, t, 10)))
print ("The price of the call option using the binomial option pricing model␣
↪(50 steps)  is: ${option_price:.4f}".format(option_price=binomial_tree(s_0,␣
↪s_p, rf, sigma, t, 50)))
print ("The price of the call option using the binomial option pricing model␣
↪(100 steps) is: ${option_price:.4f}".format(option_price=binomial_tree(s_0,␣
↪s_p, rf, sigma, t, 100)))
```

```
The price of the call option using the binomial option pricing model (10 steps)
is: $3.4759
The price of the call option using the binomial option pricing model (50 steps)
is: $3.4369
The price of the call option using the binomial option pricing model (100 steps)
is: $3.4274
```

When buying two calls with the exercises price of x_1 and x_3 and selling two calls with the exercise price of x_2, where

$$x_2 = \frac{x_1 + x_3}{2},$$

with the same maturity for the same stock, we call it a butterfly. Consider the following call options for the stock that trades at \$57.03:

Option Name	Strike Price	Call Premium (Price)
Call Option 1	50	10
Call Option 2	55	7
Call Option 3	60	5

e.) Create a graphical representation of the butterfly strategy's payoff. **[2 marks]**

Payoff for Long Call at x_1

$$\text{Payoff}_{x_1} = \max(0, S - x_1) - p_1 = \max(0, S - 50) - 10$$

Payoff for Short 2 Calls at x_2:

$$\text{Payoff}_{x_2} = -2 \cdot [\max(0, S - x_2) - p_2] = -2 \cdot [\max(0, S - 55) - 7]$$

Payoff for Long Call at x_3:

$$\text{Payoff}_{x_3} = \max(0, S - x_3) - p_3 = \max(0, S - 60) - 5$$

Total Payoff for Butterfly Spread:

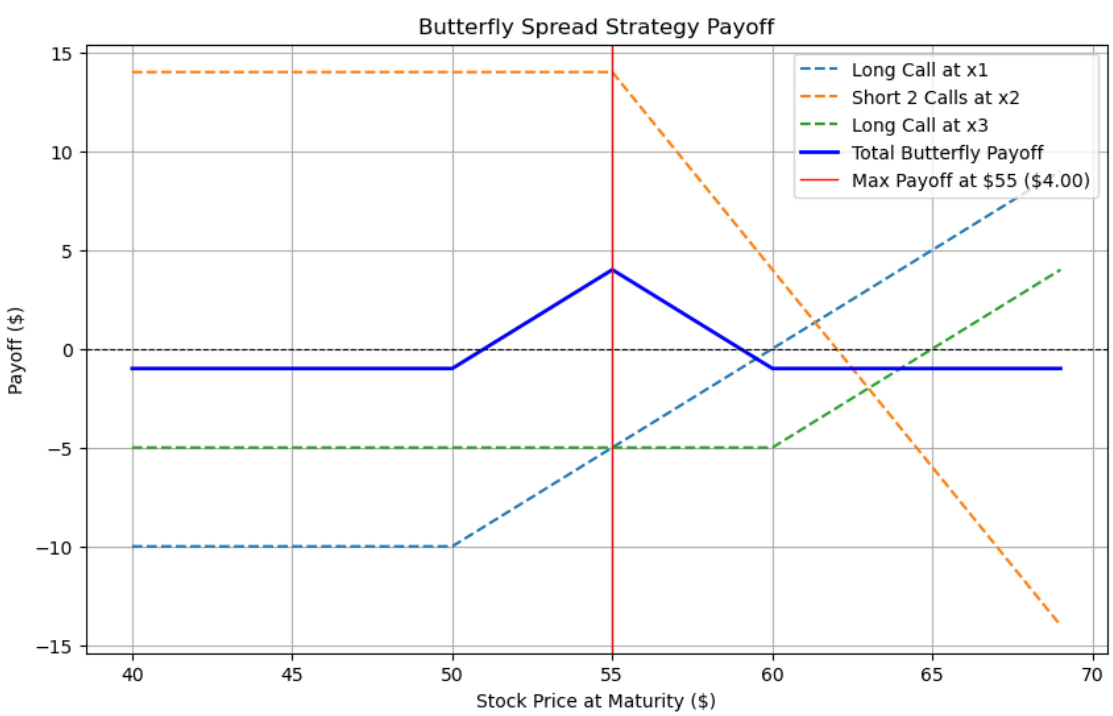
$$\text{Total Payoff} = \text{Payoff}_{x_1} + \text{Payoff}_{x_2} + \text{Payoff}_{x_3} = \max(0, S - 50) - 10 - 2 \cdot [\max(0, S - 55) - 7] + \max(0, S - 60) - 5$$

Where:

- S: Stock price at expiration $S \in [40, 69]$
- p_1, p_2, p_3: Premiums for the call options x_1, x_2, x_3

[19]:
```python
x1 = 50   # Strike price of the first call option
x2 = 55   # Strike price of the second call option
x3 = 60   # Strike price of the third call option
p1 = 10   # Premium of the first call option
p2 = 7    # Premium of the second call option
p3 = 5    # Premium of the third call option

stock_prices = np.arange(40, 70, 1) # Stock prices at maturity

payoff_x1_long = [max(0, price - x1) - p1 for price in stock_prices]
payoff_x2_short = [-2 * (max(0, price - x2) - p2) for price in stock_prices]
payoff_x3_long = [max(0, price - x3) - p3 for price in stock_prices]
# Calculate the total payoff
```

```
total_payoff = [p1 + p2 + p3 for p1, p2, p3 in
                zip(payoff_x1_long, payoff_x2_short, payoff_x3_long)]

max_payoff_index = np.argmax(total_payoff)
# Find the stock price at which the maximum payoff occurs
max_payoff_price = stock_prices[max_payoff_index]
# Find the maximum payoff value
max_payoff_value = total_payoff[max_payoff_index]

plt.figure(figsize=(10, 6))
plt.plot(stock_prices, payoff_x1_long, label='Long Call at x1', linestyle='--')
plt.plot(stock_prices, payoff_x2_short, label='Short 2 Calls at x2',␣
 ↪linestyle='--')
plt.plot(stock_prices, payoff_x3_long, label='Long Call at x3', linestyle='--')
plt.plot(stock_prices, total_payoff, label='Total Butterfly Payoff',
         color='blue', linewidth=2)
plt.axhline(0, color='black', linestyle='--', linewidth=0.8)  # Zero line
plt.axvline(x=max_payoff_price, color='red', linestyle='-', linewidth=1,
label=f'Max Payoff at \${max_payoff_price} (\${max_payoff_value:.2f})')
plt.title('Butterfly Spread Strategy Payoff')
plt.xlabel('Stock Price at Maturity ($)')
plt.ylabel('Payoff ($)')
plt.legend()
plt.grid(True)
plt.show()
```



f.) Why might an investor enter into such a strategy? **[2 marks]**

Solution:

The butterfly strategy is well-suited for investors anticipating low market volatility, as it profits most when the stock price stays near the middle strike price, $K_2 = 55$. The chart illustrates that this strategy features limited risk, with losses capped beyond the lower strike price $K_1 = 50$ and the upper strike price $K_3 = 60$. Its maximum reward occurs at K_2, where the net payoff peaks at $4.00. This cost-efficient and risk-defined strategy appeals to risk-averse investors seeking to benefit from range-bound market conditions while minimizing potential losses.

g.) Use binomial model (set some arbitrary high number of iterations) to price the european butterfly option from part e.), assuming it has 1 year to maturity, the risk-free rate is 1.5% and the annualized volatility of the underlying stock is 15%. **[5 marks]**

Butterfly Spread Payoff at Maturity (Payoff[i]):

$$\text{Payoff}[i] = \max(S_p[i] - k_1, 0) - 2 \cdot \max(S_p[i] - k_2, 0) + \max(S_p[i] - k_3, 0)$$

Where:

- $S_p[i]$: Stock price at node i
- k_1, k_2, k_3: Strike prices of the call options
- $\max(a, b)$: Maximum of a and b

Backward Recursion Using Payoff (Payoff$^{(j,i)}$):

$$\text{Payoff}^{(j,i)} = e^{-r\Delta t} \cdot \left(p \cdot \text{Payoff}^{(j+1,i+1)} + (1-p) \cdot \text{Payoff}^{(j+1,i)} \right), \quad j = n-1, n-2, \dots, 0,\ i = 0, 1, \dots, j$$

Where:

- $\Delta t = \frac{T}{n}$ is the time step
- $p = \frac{e^{r\Delta t} - d}{u - d}$ is the risk-neutral probability of an up-move in the stock price
- $u = e^{\sigma\sqrt{\Delta t}}$ and $d = \frac{1}{u}$ are the up and down factors, respectively
- r is the risk-free rate

Convergence Plot for Step Sizes:

To analyze convergence, the butterfly option price is calculated for a range of step sizes (n) and plotted.

- Set step sizes, e.g., $n = 1, 2, \dots, 1000$.
- Compute butterfly option price for each step size.
- Plot the resulting prices as a function of n to visualize convergence.

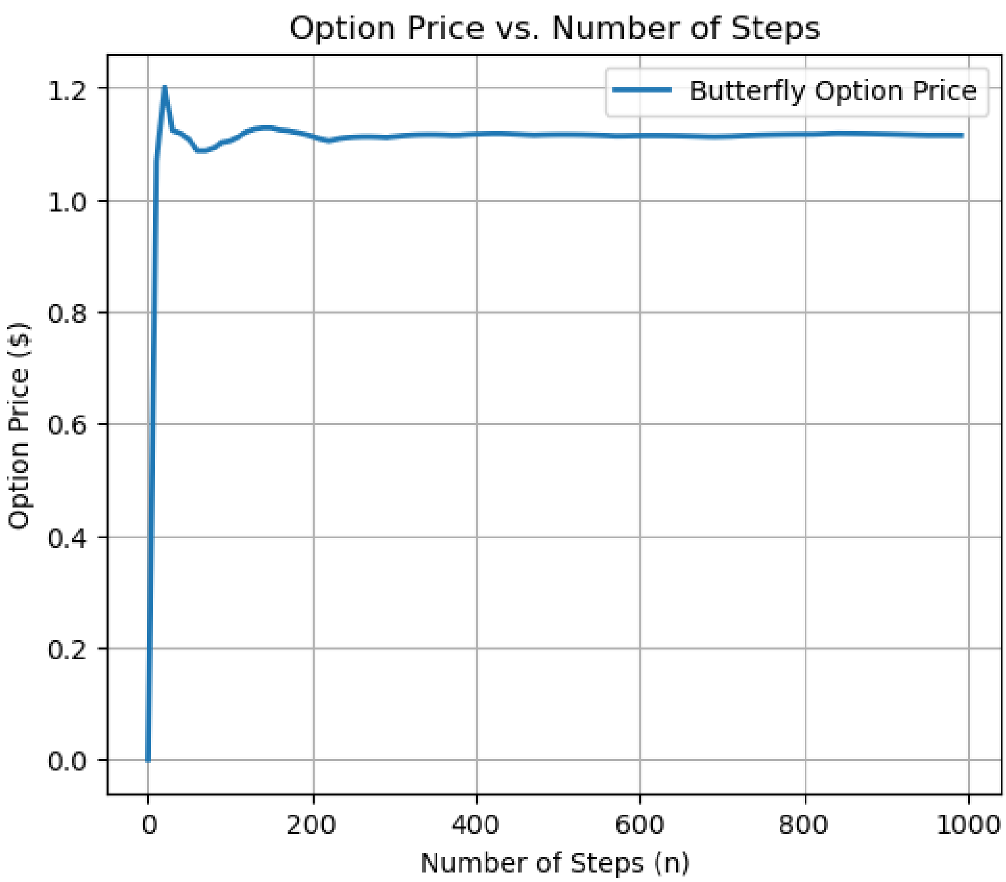
The convergence plot shows how the butterfly option price stabilizes as n increases, indicating the required step size for accurate pricing.

[20]:
```
s_0 = 57.03  # Current stock price
k1 = 50      # Strike price of the first call option
k2 = 55      # Strike price of the second call option
```

```python
k3 = 60       # Strike price of the third call option
r = 0.015     # Risk-free rate
sigma = 0.15 # Volatility
t = 1         # Time to maturity

## Function to calculate the price of the butterfly option
def binomial_butterfly(s_0, k1, k2, k3, r, sigma, T, n):
    dt = T / n
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp(r * dt) - d) / (u - d)
    discount = np.exp(-r * dt)
    stock_prices = np.zeros(n + 1)
    option_values = np.zeros(n + 1)

    for i in range(n + 1):
        stock_prices[i] = s_0 * (u ** i) * (d ** (n - i))
        sp = stock_prices[i]
        long_call1 = max(sp - k1, 0)
        short_call = -2 * max(sp - k2, 0)
        long_call2 = max(sp - k3, 0)
        option_values[i] = long_call1 + short_call + long_call2

    for j in range(n - 1, -1, -1):
        for i in range(j + 1):
            option_values[i] = discount * (p * option_values[i + 1]
                                          + (1 - p) * option_values[i])
    return option_values[0]

step_range = range(1, 1000, 10)
prices = []
for steps in step_range:
    price = binomial_butterfly(s_0, k1, k2, k3, r, sigma, t, steps)
    prices.append(price)

print("Butterfly Option Prices for Selected Steps:")
selected_indices = np.linspace(1, len(step_range) - 1, 10, dtype=int)
for idx in selected_indices:
    steps = list(step_range)[idx]
    price = prices[idx]
    print(f"n = {steps}, Butterfly Option Price = {price:.4f}")

n_400 = 400
price_400 = binomial_butterfly(s_0, k1, k2, k3, r, sigma, t, n_400)
print(f"\nThe Final Butterfly Option Price after convergence (n = {n_400}) is:
 ↪${price_400:.3f}")
plt.figure(figsize=(6, 5))
```

```python
plt.plot(step_range,
         prices, label='Butterfly Option Price', linestyle='-', linewidth=2)
plt.xlabel('Number of Steps (n)')
plt.ylabel('Option Price ($)')
plt.title('Option Price vs. Number of Steps')
plt.legend()
plt.grid(True)
plt.show()
```

```
Butterfly Option Prices for Selected Steps:
n = 11, Butterfly Option Price = 1.0682
n = 111, Butterfly Option Price = 1.1119
n = 221, Butterfly Option Price = 1.1047
n = 331, Butterfly Option Price = 1.1155
n = 441, Butterfly Option Price = 1.1171
n = 551, Butterfly Option Price = 1.1146
n = 661, Butterfly Option Price = 1.1132
n = 771, Butterfly Option Price = 1.1159
n = 881, Butterfly Option Price = 1.1173
n = 991, Butterfly Option Price = 1.1145

The Final Butterfly Option Price after convergence (n = 400) is: $1.114
```



As n (the number of steps in the binomial tree) increases, the stock price nodes become more granular, allowing the binomial model to better approximate the continuous price paths assumed in models like the Black-Scholes framework. This refinement ensures that the calculated payoff values converge to the theoretical results from the continuous-time model. For this particular strategy, as $n \to \infty$, the payoff value converges to approximately \$1.114.

5 Question 5: Case Study [14 marks] (Report on the next page)

Using the data sources and tools introduced in Workshop 3, along with any relevant material from the lectures, analyze the current state and future outlook of a selected country/economy from a macroeconomic and financial perspective. As a primary suggestion, groups may consider analyzing the United States economy, as most available free data sources focus on this country. However, groups are welcome to choose any country/economy they wish to analyze. Groups MUST include figures and/or tables from various data sources, as well as research reports or studies that justify and support the analysis conducted. The report should be approximately 600/700 words, excluding the bibliography, captions for tables and figures, and any numbers included in tables.

6 Question 6: Case Study [17 marks] (Report on the next page)

Using the data sources and tools introduced in Workshop 3, along with any relevant material from the lectures, analyze the current state and future outlook of a selected publicly traded company (stock) included in a major stock market index, such as the S&P 500 or Russell 2000, among others. The company may be located in a different country than the one chosen in Question 5. As a primary suggestion, groups may consider analyzing a company from the "Magnificent 7": Apple, Microsoft, Alphabet (Google), Amazon, Nvidia, Tesla, or Meta (Facebook). You should include, at least, a brief introduction to the company (i.e., a description of its main business) and a fundamental analysis.

Groups MUST include figures and/or tables from a variety of data sources, as well as research reports or studies that justify and support the analysis conducted. The report should be approximately 600/700 words, excluding the bibliography, captions for tables and figures, and any numbers included in tables.

You are not limited to analyzing an "excellent" publicly company/stock; you may also consider a poorly performing company whose stock price may decline, making a short position potentially profitable.

Each group, working as financial analysts, must provide a final conclusion based on their analysis by assigning a rating to the company. The rating scale is as follows: - (5) Strong Sell - (4) Sell - (3) Neutral - (2) Buy - (1) Strong Buy

Note that calculating a (target) price for the company is not required; simply provide a recommendation based on thorough research and analysis.