CF969-7-SP-CO and CF969-7-PT-CO
Big Data for Computational Finance Assignment1

Melek Kuru 2315873

March 2024

TASK 1

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np
import random
import matplotlib.pyplot as plt
from math import sqrt
random.seed(2315873)
n = 8
d1 = 7
d2 = 3
dummystep = 10*d1 + d2

for _ in range(dummystep):
    dummy = random.uniform(0, 1)

Corr = np.array([[0]*n for _ in range(n)], dtype=float)
for i in range(n):
    for j in range(n):
        Corr[i][j] = (-1)**abs(i-j)/(abs(i-j)+1)

ssigma = np.array([[0]*1 for _ in range(n)], dtype=float)
mmu = np.array([[0]*1 for _ in range(n)], dtype=float)
ssigma[0] = 2
mmu[0] = 3

for i in range(n-1):
    ssigma[i+1] = ssigma[i] + 2*random.uniform(0, 1)
    mmu[i+1] = mmu[i] + 1

ddiag = np.array([[0]*n for _ in range(n)], dtype=float)
np.fill_diagonal(ddiag, ssigma)
C2 = np.matmul(np.matmul(ddiag, Corr), ddiag)
C = 0.5*(C2 + C2.T)

m = gp.Model('portfolio')
```

```python

x = m.addMVar(n, lb=0.0, ub=1.0, vtype=GRB.CONTINUOUS, name='x')

portfolio_risk = x @ C @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

m.addConstr(x.sum() == 1, 'budget')

m.write('portfolio_selection_optimization.lp')

optimal_allocations = []

target_returns = np.arange(3.00, 9.25, 0.25)

for target_return in target_returns:

    target_return_constraint = m.addConstr(mmu.T @ x == target_return,
         'target_return')


    m.optimize()
    if m.status == GRB.OPTIMAL:
        optimal_allocation = x.X
        optimal_allocations.append(optimal_allocation)
        print(f"Target return: {target_return}, Optimal Allocation: {
            optimal_allocation}")
    m.remove(target_return_constraint)
    standard_deviations = []
expected_returns = []

for optimal_allocation in optimal_allocations:
    std_dev = sqrt(np.dot(optimal_allocation, np.dot(C,
        optimal_allocation)))
    exp_return = np.dot(mmu.flatten(), optimal_allocation)

    standard_deviations.append(std_dev)
    expected_returns.append(exp_return)

print("Standard Deviations List : \n\n", standard_deviations)
print("\n")
print("Expected Returns List : \n\n", expected_returns)
min_risk_index = np.argmin(standard_deviations)
minrisk_volatility = standard_deviations[min_risk_index]
minrisk_return = expected_returns[min_risk_index]
fig, ax = plt.subplots(figsize=(10, 8))
ax.scatter(standard_deviations, expected_returns, color='black', label
    ='Portfolio Optimization Results')
ax.scatter(minrisk_volatility, minrisk_return, color='yellow', label='
    Minimum Risk Portfolio')
ax.annotate('Minimum Risk Portfolio', (minrisk_volatility,
    minrisk_return),
             xytext=(-50, 30), textcoords='offset points',
                arrowprops=dict(facecolor='black', arrowstyle='->'))
ax.set_xlabel('Standard Deviation ')
ax.set_ylabel('Expected Return ')
ax.set_title('Portfolio Optimization Results')
ax.legend()
ax.grid(True)
plt.show()
```

Listing 1: Python Code for Portfolio Optimization - Task 1

EXPLANATION OF TASK1:

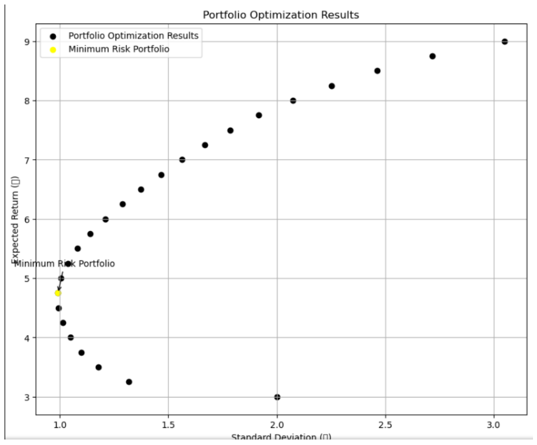

Figure 1: Output of Task 1

Based on Markowitz's contemporary portfolio theory, the task entails optimizing investment portfolios to achieve minimal risk for desired returns. It uses the covariance matrix to understand asset interrelationships and the standard deviation to calculate risk. The weighted average of the returns on each individual asset is used to determine the overall returns of efficient portfolios, which maximize returns for a given level of risk or minimize risk for a given return.

OBSERVATIONS:

- The portfolios plotted start at around a standard deviation of 1.0 for the lowest expected return of 3.0. As the expected return increases to 9.0, the associated risk also increases, with standard deviation reaching up to about 3.0.

- The Minimum Risk Portfolio is highlighted with a yellow dot, located at approximately a standard deviation of 1.0 and an expected return near 5.0. This specific portfolio is the optimal choice for investors seeking the lowest risk, as it offers the smallest standard deviation for the given set of portfolios.

- The expected return rises as we move rightward along the horizontal axis (increasing standard deviation), illuminating the fundamental trade-off in investing: higher returns are typically accompanied by higher risk.

- The way that the portfolios (represented by black dots) progress seems to indicate an efficiency frontier—an upward-sloping curve that shows how risk and return are related. This curve indicates that a portfolio is efficient if it offers the highest expected return for a given degree of risk.
- The scatter plot shows a clear positive correlation between risk and expected return. The Minimum Risk Portfolio is clearly the outlier on the bottom left, indicating its unique position in offering the lowest risk.
- The variance in expected returns for similar levels of risk suggests that there are multiple portfolio options available for an investor, depending on their risk-return preference.

CONCLUSION:
The thorough analysis supports the principles of portfolio optimization, according to which higher expected returns are typically associated with higher risk. The Minimum Risk Portfolio, which stands out in the scatter plot, is an important idea in portfolio management since it is the lowest volatility investment option. Investors can select portfolios based on their risk appetite by choosing from those above and to the right of the Minimum Risk Portfolio, which offers a trade-off between increased risk and possibly higher returns.

TASK 2

```python
m = gp.Model('portfolio')

x = m.addMVar(n, lb=0.0, ub=1.0, vtype=GRB.CONTINUOUS, name='x')

portfolio_risk = x @ C @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

m.addConstr(x.sum() <= 1, 'modified_budget')

optimal_allocations = []
target_returns = np.arange(3.00, 9.25, 0.25)

for target_return in target_returns:
    target_return_constraint = m.addConstr(mmu.T @ x == target_return,
        'target_return')
    m.optimize()
    if m.status == GRB.OPTIMAL:
        optimal_allocation = x.X
        optimal_allocations.append(optimal_allocation)
    m.remove(target_return_constraint)

standard_deviations = []
expected_returns = []
for optimal_allocation in optimal_allocations:
    std_dev = sqrt(np.dot(optimal_allocation, np.dot(C,
       optimal_allocation)))
    exp_return = np.dot(mmu.flatten(), optimal_allocation)
    standard_deviations.append(std_dev)
    expected_returns.append(exp_return)
    min_risk_index = np.argmin(standard_deviations)
minrisk_volatility = standard_deviations[min_risk_index]
minrisk_return = expected_returns[min_risk_index]

fig, ax = plt.subplots(figsize=(10, 8))
ax.scatter(standard_deviations, expected_returns, color='black', label='Portfolio Optimization Results')
ax.scatter(minrisk_volatility, minrisk_return, color='yellow', label='Minimum Risk Portfolio')
ax.annotate('Minimum Risk Portfolio', (minrisk_volatility, minrisk_return), xytext=(-50, 30), textcoords='offset points', arrowprops=dict(facecolor='black', arrowstyle='->'))
ax.set_xlabel('Standard Deviation ')
ax.set_ylabel('Expected Return ')
ax.set_title('Portfolio Optimization Results with Modified Budget Constraint')
ax.legend()
ax.grid(True)
plt.show()
```

Listing 2: Python Code for Portfolio Optimization - Task 2

EXPLANATION OF TASK 2:

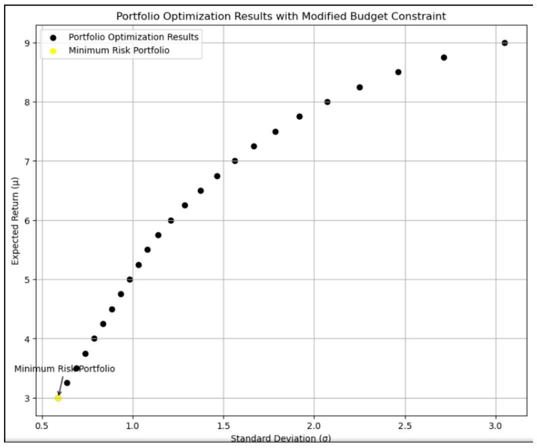

Figure 2: Output of Task 2

In Task 2, the investment strategy is adjusted to permit partial capital investments, which is similar to holding cash or a risk-free asset with no return. This allows for the uninvestment of some portfolio components by changing the constraint from a rigid budget equality to a flexible inequality. By incorporating the concept of the Capital Asset Pricing Model (CAPM), this approach expands the risk-return options available to investors by facilitating the creation of a capital market line that represents effective mixes of risky and risk-free assets.

OBSERVATIONS:

- Similar to Task 1, Task 2's scatter plot displays a notable extension towards the lower end of the risk axis. This represents portfolios in which a portion of capital may be left uninvested, hence lowering overall risk.
- The Minimum Risk Portfolio for Task 2, highlighted in yellow, is positioned at an even lower risk than the Minimum Risk Portfolio for Task 1, which is indicated by a shift towards the left on the scatter plot.
- The portfolios begin at a standard deviation significantly closer to zero, which suggests that the portfolio could hold a substantial amount of the capital uninvested, essentially lowering the portfolio's overall risk.
- Task 2's efficient frontier starts at the Minimum Risk Portfolio and extends upwards to the right, just like in Task 1, but the introduction of a risk-free option bends the frontier downwards towards the risk axis, showing that lower risk levels are now achievable.
- The range of standard deviations in Task 2 is extended towards the lower end, starting near 0.5, compared to the previous task that started around 1.0. This change is a direct result of the modified budget constraint.
- The range of expected returns remains approximately the same, from 3.0 to 9.0, showing that the investment opportunity set still offers the same upside potential.
- The Minimum Risk Portfolio has a lower standard deviation in Task 2 compared to Task 1, reflecting the presence of the cash or risk-free asset in the portfolio.

CONCLUSION:
The optimization results for Task 2 demonstrate that investors can achieve a lower-risk portfolio by easing the budget constraint to allow for uninvested capital, which is essentially a risk-free asset. This gives portfolio construction a crucial new dimension by highlighting the significance of risk aversion and liquidity preference.

TASK 3

```
m = gp.Model('portfolio')

x = m.addMVar(n, lb=0.0, ub=1.0, vtype=GRB.CONTINUOUS, name='x')

portfolio_risk = x @ C @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

m.addConstr(x.sum() == 1, 'budget')

target_returns = np.arange(3.00, 9.25, 0.25)
optimal_allocations = []

for target_return in target_returns:
```

```python
    target_return_constraint = m.addConstr(mmu.T @ x >= target_return, 'target_return')
    m.optimize()

    if m.status == GRB.OPTIMAL:
        optimal_allocation = x.X
        optimal_allocations.append(optimal_allocation)

    m.remove(target_return_constraint)
standard_deviations = [sqrt(np.dot(optimal_allocation, np.dot(C, optimal_allocation))) for optimal_allocation in optimal_allocations]
expected_returns = [np.dot(mmu.flatten(), optimal_allocation) for optimal_allocation in optimal_allocations]
min_risk_index = np.argmin(standard_deviations)
fig, ax = plt.subplots(figsize=(10, 8))
ax.scatter(standard_deviations, expected_returns, color='black', label='Portfolio Optimization Results')
ax.scatter(standard_deviations[min_risk_index], expected_returns[min_risk_index], color='yellow', label='Minimum Risk Portfolio')
ax.annotate('Minimum Risk Portfolio', (standard_deviations[min_risk_index], expected_returns[min_risk_index]), xytext=(-50, 30), textcoords='offset points', arrowprops=dict(facecolor='black', arrowstyle='->'))
ax.set_xlabel('Standard Deviation ')
ax.set_ylabel('Expected Return ')
ax.set_title('Modified Portfolio Optimization Results')
ax.legend()
ax.grid(True)
plt.show()
```

Listing 3: Python Code for Portfolio Optimization - Task 3

EXPLANATION OF TASK 3:

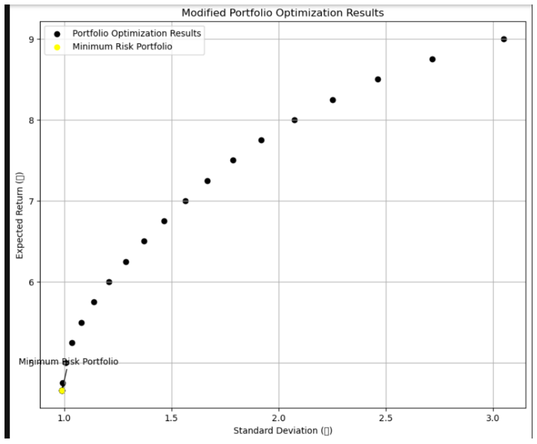

Figure 3: Output of Task 3

Portfolios can now aim for a minimum expected return target instead of a specific one, with the possibility of exceeding it if risk constraints allow. This change is in line with common investment strategies that strike a balance between pursuing additional returns and obligations or goals. It also accommodates investors who have a minimum return requirement but are open to higher risk and returns.
OBSERVATIONS:

- The portfolios represented in the scatter plot generally display a higher range of expected returns compared to Task 1, as indicated by the minimum expected return constraint being set to greater than or equal to the target return r.

- The Minimum Risk Portfolio is marked with a yellow dot and is positioned with the lowest risk among all portfolios that meet or exceed the minimum expected return constraint.

- The range of standard deviations seems to be similar to that of Task 1; however, there are fewer portfolios with very low risk and high return, as the constraint allows for portfolios to focus more on maximizing return once the minimum return has been met.

- It is notable that the minimum standard deviation value for the Minimum Risk Portfolio appears similar to that of Task 1, hovering around 1.0. However, due to the change in the expected return constraint, the associated returns for similar risk levels may be higher.

- The highest expected return values seem to be similar to those in Task 1, again reaching around 9.0, but they may be achieved with a broader range of risk (standard deviation) levels since the optimization is not constrained to achieve exactly those returns.

CONCLUSION:
The modification in Task 3 allows for portfolios that meet or exceed a certain level of expected return, rather than targeting a specific return. This adjustment aligns with investment strategies that prioritize reaching a certain return threshold while also trying to capture higher returns when they can be achieved without exceeding a set risk level.

TASK 4

```
m = gp.Model('portfolio')

x = m.addMVar(n, lb=-GRB.INFINITY, ub=1.0, vtype=GRB.CONTINUOUS, name=
    'x')


portfolio_risk = x @ C @ x
m.setObjective(portfolio_risk, GRB.MINIMIZE)

m.addConstr(x.sum() == 1, 'budget')

m.write('portfolio_selection_optimization.lp')
```

```python

optimal_allocations = []

target_returns = np.arange(3.00, 9.25, 0.25)

for target_return in target_returns:

    target_return_constraint = m.addConstr(mmu.T @ x == target_return,
         'target_return')

    m.optimize()

    if m.status == GRB.OPTIMAL:

        optimal_allocation = x.X

        optimal_allocations.append(optimal_allocation)

        print(f"Target return: {target_return}, Optimal Allocation: {
            optimal_allocation}")

    m.remove(target_return_constraint)


standard_deviations = []
expected_returns = []

for optimal_allocation in optimal_allocations:
    std_dev = sqrt(np.dot(optimal_allocation, np.dot(C,
        optimal_allocation)))
    exp_return = np.dot(mmu.flatten(), optimal_allocation)

    standard_deviations.append(std_dev)
    expected_returns.append(exp_return)

print("Standard Deviations List: \n\n", standard_deviations)
print("\n")
print("Expected Returns List : \n\n", expected_returns)
import matplotlib.pyplot as plt
import numpy as np


min_risk_index = np.argmin(standard_deviations)
minrisk_volatility = standard_deviations[min_risk_index]
minrisk_return = expected_returns[min_risk_index]

fig, ax = plt.subplots(figsize=(10, 8))

ax.scatter(standard_deviations, expected_returns, color='black', label
    ='Portfolio Optimization Results')

```

```
ax.scatter(minrisk_volatility, minrisk_return, color='yellow', label='
    Minimum Risk Portfolio')
ax.annotate('Minimum Risk Portfolio', (minrisk_volatility,
    minrisk_return),
             xytext=(-50, 30), textcoords='offset points',
             arrowprops=dict(facecolor='black', arrowstyle='->'))

ax.set_xlabel('Standard Deviation ')
ax.set_ylabel('Expected Return ')
ax.set_title('Portfolio Optimization Results')
ax.legend()
ax.grid(True)
plt.show()
```

Listing 4: Python Code for Portfolio Optimization - Task 4

EXPLANATIONS OF TASK 4:

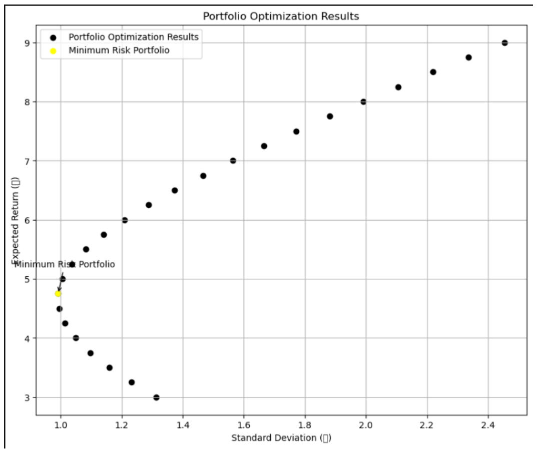

Figure 4: Output of Task 4

To model short selling, one can allow the asset weights in a portfolio to be negative. In short selling, investors sell securities they do not own in the hopes of buying them back at a lower price. With the elimination of the positive weight restriction, portfolio managers now have more strategic options. For example, they can take advantage of asset declines to hedge against losses. In certain market circumstances, this could theoretically lower risk and boost returns.

OBSERVATIONS:

- The plot shows portfolios with expected returns similar to those in Task 1, but with a wider range of standard deviations. This implies that short selling has enabled some portfolios to achieve similar returns with potentially less risk, and vice versa.
- The Minimum Risk Portfolio, marked with a yellow dot, could potentially have a lower standard deviation compared to the same portfolio in Task 1. This is because short selling can be used to hedge against risk.

- The Minimum Risk Portfolio's position in the scatter plot indicates a possible decrease in risk due to short selling, with the lowest risk portfolio now situated at a lower standard deviation than the one in Task 1.

- Some portfolios are achieving expected returns at different levels of risk than those possible without short selling, as indicated by the spread of the portfolios on the scatter plot.

CONCLUSION:

The ability to take short positions in a portfolio allows for greater flexibility in managing risk and seeking returns. The observed dispersion in the Task 4 results suggests that some portfolios may benefit from short selling to achieve a more favorable risk-return profile.