The University of Texas at Austin
McCombs School of Business

RM 294: OPTIMIZATION
PROJECT 2 REPORT
INTEGER PROGRAMMING

Submitted by -
Ronak Goyal (rg49394)
Sarthak Shivnani (ss223347)
Aryan Shah (ahs2388)
Brinda Asuri (bva246)

Table of Contents

1. INTRODUCTION

An index fund is a type of mutual fund or exchange-traded fund (ETF) designed to replicate or track the performance of a specific financial market index, such as the NASDAQ-100, the Standard & Poor's 500 Index (S&P 500), or similar benchmarks. In theory, an index fund could be constructed by purchasing all constituent stocks of the target index, weighted in accordance with the index's composition. However, this approach is often impractical due to the high number of small positions required, leading to substantial transaction and rebalancing costs, and potential liquidity challenges. Consequently, constructing an index fund with a subset of q stocks, where q is significantly smaller than the total number of stocks in the index (n), is a more efficient strategy. In this project, we aim to create a smaller, optimized portfolio that closely tracks the performance of the NASDAQ-100 index while mitigating the logistical and cost-related drawbacks of full replication.

1.1 Objective

The primary objective of this project is to design and implement an index fund that closely tracks the NASDAQ-100 index using a significantly smaller number of stocks (m) than the total components in the index (n). This involves creating an integer program to select precisely m stocks based on a similarity matrix (ρ), where ρij represents the similarity between stocks i and j, typically measured through correlation or similar metrics. Following stock selection, a linear program will be used to determine the optimal weight for each chosen stock in the portfolio. The performance of this index fund will then be evaluated against the NASDAQ-100 index using out-of-sample data, with assessments conducted across various values of m to explore the impact of diversification. Ultimately, this project seeks to balance accuracy, cost-effectiveness, and diversification, optimizing the construction of a simplified index fund that effectively tracks the NASDAQ-100 index.

2. DATA PREPARATION

The dataset comprises two CSV files containing daily prices for a market index and its component stocks over two consecutive years. Each file is organized with dates in the first column, followed by the index price, and subsequent columns containing prices of individual component stocks. Data from the first year will be used for portfolio construction, while data from the second year will be used to evaluate the performance of the constructed portfolios. This project framework is designed to be adaptable, allowing for the methodology's application to various market indexes and components. This section establishes the foundation for subsequent analyses on stock selection, weight allocation, and portfolio performance.

2.1 Data Cleaning

We Took the following steps to prepare the data for analysis-

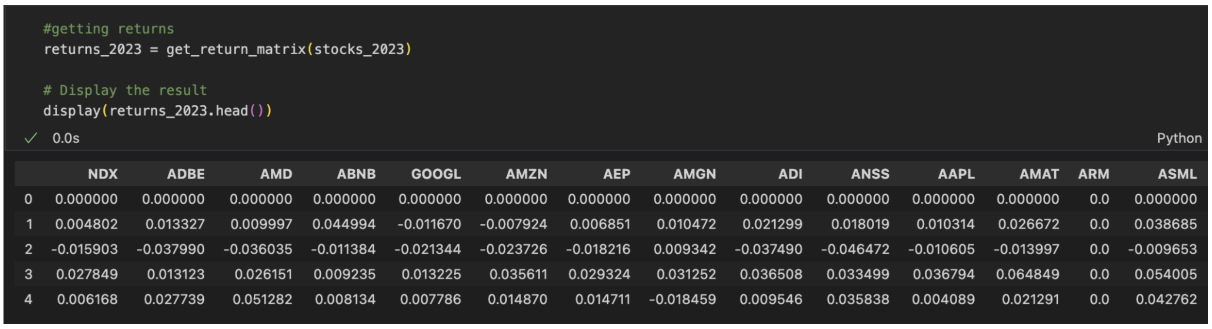
- *Imputed missing index values*- Both the 2023 and 2024 datasets contained missing index values, impacting the accuracy of our analysis. To address this, we manually imputed the missing indices by referencing each day's closing index values obtained from Yahoo Finance.
- *Imputing zeros for missing values-* The stock Arm was listed on September 14, 2023, leading to absent stock price data in the 2023 dataset for dates prior to its listing. In the

absence of historical prices, these missing values were imputed as zero to maintain dataset continuity.

- *Handling remaining missing values-* We applied linear interpolation first and then used forward fill and backward fill to address any additional missing values in the dataset.

2.2 Data Modeling

- *Return Matrix-* To capture daily fluctuations in stock performance, we calculated daily stock returns, establishing a consistent metric for comparison over time. These returns enable us to analyze each stock's behaviour relative to the index, which is crucial in selecting stocks that represent our fund accurately. Focusing on returns rather than raw prices allow us to standardize performance across varying price levels, ensuring a closer alignment of our portfolio with the index. This method provides a strong basis for evaluating and adjusting our fund to closely mirror the index's movements.

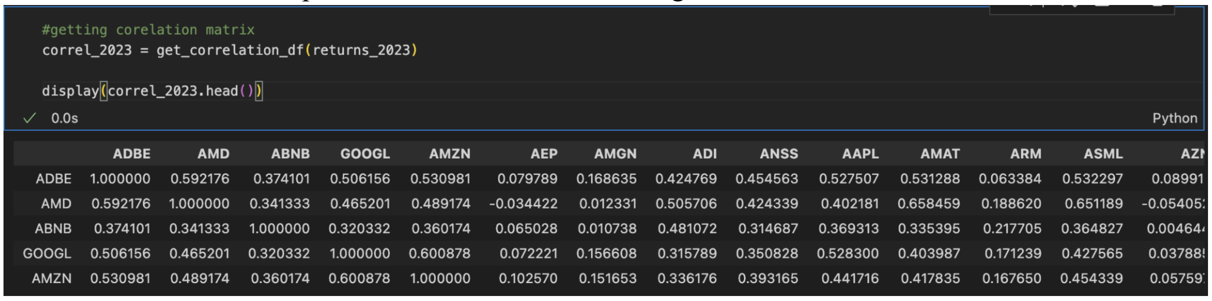


```
#getting returns
returns_2023 = get_return_matrix(stocks_2023)

# Display the result
display(returns_2023.head())
```

✓ 0.0s Python

	NDX	ADBE	AMD	ABNB	GOOGL	AMZN	AEP	AMGN	ADI	ANSS	AAPL	AMAT	ARM	ASML
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000
1	0.004802	0.013327	0.009997	0.044994	-0.011670	-0.007924	0.006851	0.010472	0.021299	0.018019	0.010314	0.026672	0.0	0.038685
2	-0.015903	-0.037990	-0.036035	-0.011384	-0.021344	-0.023726	-0.018216	0.009342	-0.037490	-0.046472	-0.010605	-0.013997	0.0	-0.009653
3	0.027849	0.013123	0.026151	0.009235	0.013225	0.035611	0.029324	0.031252	0.036508	0.033499	0.036794	0.064849	0.0	0.054005
4	0.006168	0.027739	0.051282	0.008134	0.007786	0.014870	0.014711	-0.018459	0.009546	0.035838	0.004089	0.021291	0.0	0.042762

- *Correlation Matrix Calculation-* We start by calculating the returns correlation matrix to understand the relationships between various stocks, using Pearson correlation.



```
#getting corelation matrix
correl_2023 = get_correlation_df(returns_2023)

display(correl_2023.head())
```

✓ 0.0s Python

	ADBE	AMD	ABNB	GOOGL	AMZN	AEP	AMGN	ADI	ANSS	AAPL	AMAT	ARM	ASML	AZ
ADBE	1.000000	0.592176	0.374101	0.506156	0.530981	0.079789	0.168635	0.424769	0.454563	0.527507	0.531288	0.063384	0.532297	0.08991
AMD	0.592176	1.000000	0.341333	0.465201	0.489174	-0.034422	0.012331	0.505706	0.424339	0.402181	0.658459	0.188620	0.651189	-0.05405
ABNB	0.374101	0.341333	1.000000	0.320332	0.360174	0.065028	0.010738	0.481072	0.314687	0.369313	0.335395	0.217705	0.364827	0.00464
GOOGL	0.506156	0.465201	0.320332	1.000000	0.600878	0.072221	0.156608	0.315789	0.350828	0.528300	0.403987	0.171239	0.427565	0.03788
AMZN	0.530981	0.489174	0.360174	0.600878	1.000000	0.102570	0.151653	0.336176	0.393165	0.441716	0.417835	0.167650	0.454339	0.05759

3. PORTFOLIO CONSTRUCTION USING LINEAR PROGRAMMING

3.1 Approach

In the phase of stock selection and weight allocation, our primary goal was to construct an index fund that closely mirrors the NASDAQ-100 index, using a simplified subset of its component stocks. This involved a two-step optimization process: Integer Programming (IP) for selecting stocks and Linear Programming (LP) for allocating weights. During the IP phase, we focused on identifying an optimal subset of NASDAQ-100 stocks to include in the portfolio, using a 'similarity matrix' to capture relationships among stocks. By maximizing similarity between chosen stocks and their benchmarks, we aimed to achieve a well-diversified portfolio. In the LP phase, we optimized the weights of these selected stocks, incorporating their historical returns relative to the NASDAQ-100 index. A transformation technique was applied to manage the non-linearity from absolute values in the linear program, enabling us to calculate weights that closely align with the index's returns. The results from this stage provided insights into the portfolio’s tracking accuracy for both in-sample (2023) and out-of-sample (2024) periods.

3.2 Stock Selection

Objective

The primary objective of the stock selection process is to maximize the similarity between a selected subset of m stocks and the entire index of n stocks. This objective is represented by an equation where ρi,j denotes the correlation between stocks in the constructed fund and the index, and xi,j is a binary variable indicating whether a stock from the index is included in the fund.

$$\max_{x,y} \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} x_{ij}$$

Decision Variables

In the code, the decision variables are designed to model the selection and mapping of elements for constructing a subset that effectively represents a larger set, such as in portfolio or index fund creation:

1. *Selection Variable***:** A binary vector indicating which elements from the full set are chosen for the subset.
   - o Each entry in the vector can be 1 (if the element is selected) or 0 (if not selected).
   - o This variable constrains the model to select a predefined number of elements that best represent the larger set.
2. *Linking Variable:* A binary matrix that models the association between each element in the larger set and each element in the subset.
   - o For each pair of elements, a value of 1 indicates that an element in the subset is the best representative for an element in the larger set, while 0 indicates no association.
   - o This variable allows the model to establish optimal mappings between elements of the subset and the larger set based on specified criteria, such as similarity or correlation.

Together, these decision variables enable the model to select an optimal subset and define representative links to closely approximate the characteristics of the larger set.

Constraints

To achieve this objective, three key constraints have been established:

- *Constraint 1:* Limits the number of elements from set I (the index) that can be mapped to set F (the fund) to a maximum of q (or m), ensuring that only a selected subset of stocks from the index is included in the fund.

$$s.t. \sum_{j=1}^{n} y_j = m.$$

- *Constraint 2*: Enforces that each element in set I maps uniquely to a single element in set F, establishing a one-to-one relationship between index and fund stocks. This ensures that each stock in the index corresponds to only one stock in the fund.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad for\ i = 1,2,\dots,n$$

- *Constraint 3:* Stipulates that if an element from set I is mapped to an element in set F (i.e., an index stock is included in the fund), the chosen stock in F must indeed be part of the fund. This guarantees that selected index stocks are incorporated into the fund portfolio.

$$x_{ij} \leq y_j \qquad for\ i,j = 1,2,\dots,n$$

$$x_{ij}, y_j \in \{0,1\}$$

3.3 Portfolio Weights Calculation

Objective

The primary objective of the portfolio weight calculation is to determine the optimal allocation for each selected stock within the fund, thereby ensuring that the portfolio's weighted average returns closely mirror those of the target index. This approach seeks to achieve accurate index tracking while maintaining a fully invested portfolio. Following the selection of constituent stocks for the fund, the subsequent step involves calculating precise weight allocations for each stock. This optimized allocation is structured to align the portfolio's performance with that of the index as closely as possible.

$$\min_{w} \sum_{t=1}^{T} \left| q_t - \sum_{i=1}^{m} w_i r_{it} \right|$$

Decision Variables

- *Selection Variable:* A binary variable indicating whether an element from the full set is chosen for inclusion in the subset.
    - Each entry in this binary vector takes a value of 1 if the element is selected, or 0 if it is not.
    - This variable restricts the model to select a specified number of elements that best represent the larger set.
- *Linking Variable*: A binary matrix that models the association between each element in the larger set and each element in the subset.
    - For each pair of elements, a value of 1 indicates that an element in the subset is chosen as the best representative for an element in the larger set, while 0 indicates no association.
    - This variable enables the model to establish optimal mappings between the elements in the subset and the larger set based on specific similarity criteria.
- *Weight Variable*: A continuous variable representing the proportion allocated to each element in the subset.
    - This variable determines the optimal allocation of each selected element, ensuring that the subset's weighted average aligns with the characteristics of the larger set.
    - Constraints ensure that these weights are non-negative and sum to a specified total, maintaining a fully allocated subset.

These decision variables enable the model to select an optimal subset, establish representational links, and allocate weights to approximate the characteristics of the larger set effectively.

Constraints

To maintain the integrity of this allocation, two essential constraints need to be satisfied.

- *Constraint 1*-the total sum of all weight allocations must equal 1, ensuring that the entire portfolio allocation is fully utilized.

$$s.t. \sum_{i=1}^{m} w_i = 1$$

- *Constraint 2*- each weight should lie between 0 and 1, indicating that each stock's allocation is non-negative and does not exceed 100%.

$$w_i \geq 0.$$

3.4 Portfolio Analysis for Different 'M' Values

We've observed that a fund constructed with only 5 stocks yields satisfactory performance, but we now aim to explore possibilities for further enhancement. To achieve this, we'll revisit the stock selection process, adjusting the number of stocks included in the fund, represented by 'm.' Users can customize the range of 'm' values to be tested, with intervals of 10 between each. This analysis spans a wide range of 'm' values, from 10 to 100. By systematically varying 'm,' our goal is to identify the ideal balance between portfolio size and tracking performance. This approach will help us determine whether adding more stocks leads to diminishing returns in tracking accuracy. The insights gained for each 'm' value will clarify the trade-offs between portfolio complexity and tracking precision.

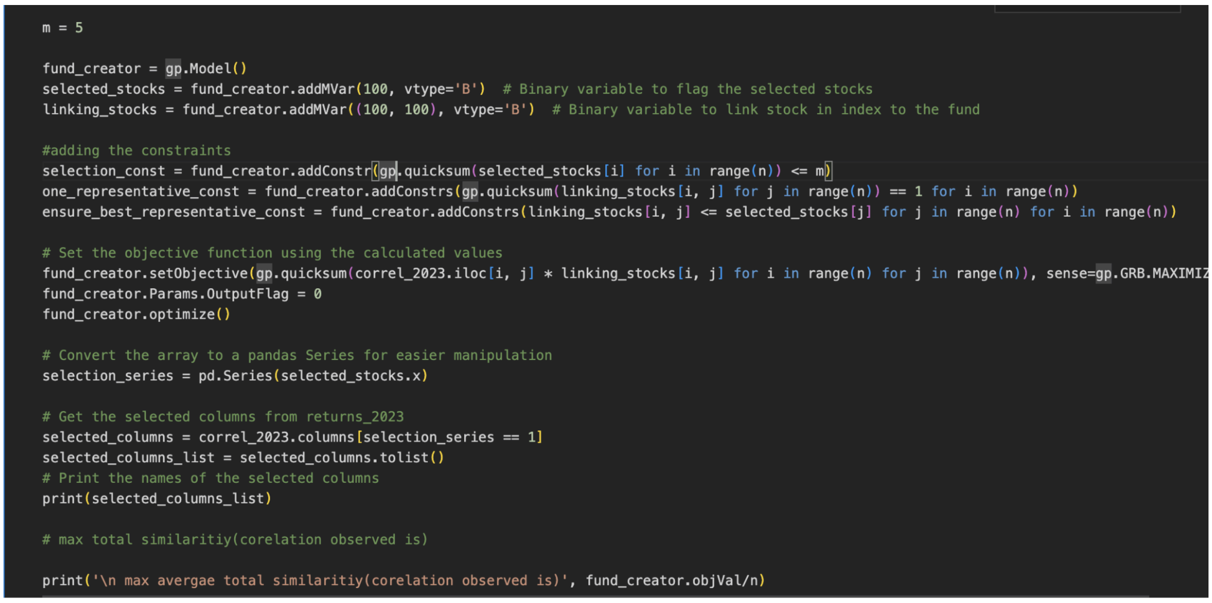
3.4.1 Find the best 5 stocks to include in your portfolio and the weights of those 5 stocks, using the 2023 data. How well does this portfolio track the index in 2024?

An analysis of 2023 data was conducted to identify an optimal subset of five stocks that could closely track the NASDAQ-100 index. The selected stocks—INTU, NXPI, HON, PEP, and SNPS—were chosen due to their high representational correlation with the NASDAQ-100. Each of these stocks serves as a significant representative within the index, with INTU, for instance, representing 28 times, NXPI 24 times, HON 22 times, PEP 14 times, and SNPS 12 times. This selection ensures that the subset captures key dynamics of the overall index.

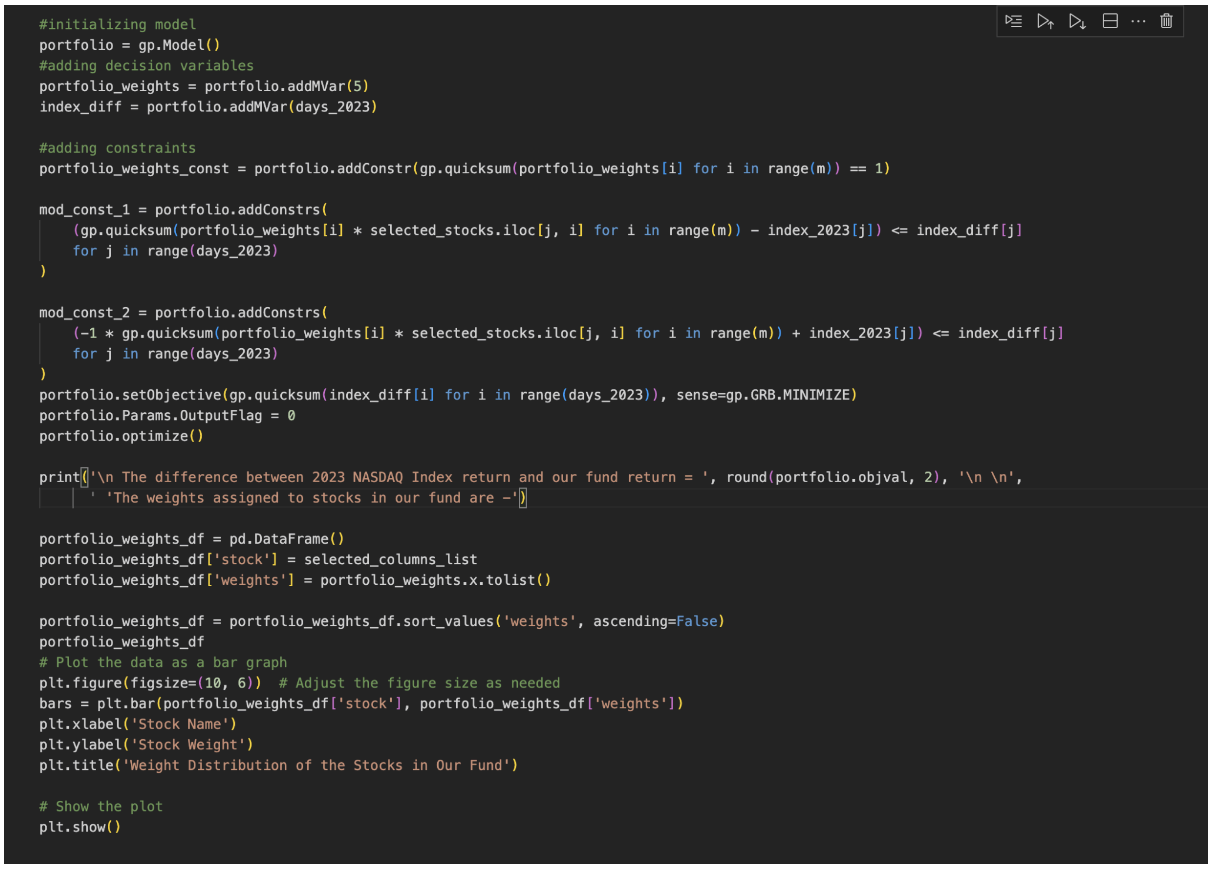
The portfolio was then constructed by assigning optimal weight allocations to each selected stock, aiming to align the portfolio's weighted returns with those of the index. Performance testing on 2024 data indicated that the portfolio tracked the index with high accuracy, showing a training error of 1.11 for 2023 and a testing error of 1.15 for 2024. These low tracking error values suggest that the portfolio effectively mirrors the NASDAQ-100's return patterns.

Visual analysis reinforces this result. The line plot of daily returns for 2024 shows a close alignment between the NASDAQ-100 index and the portfolio, suggesting effective tracking performance. Additionally, the bar chart illustrating each stock's representational frequency supports their selection, as each stock makes a significant contribution to the index's behavior.

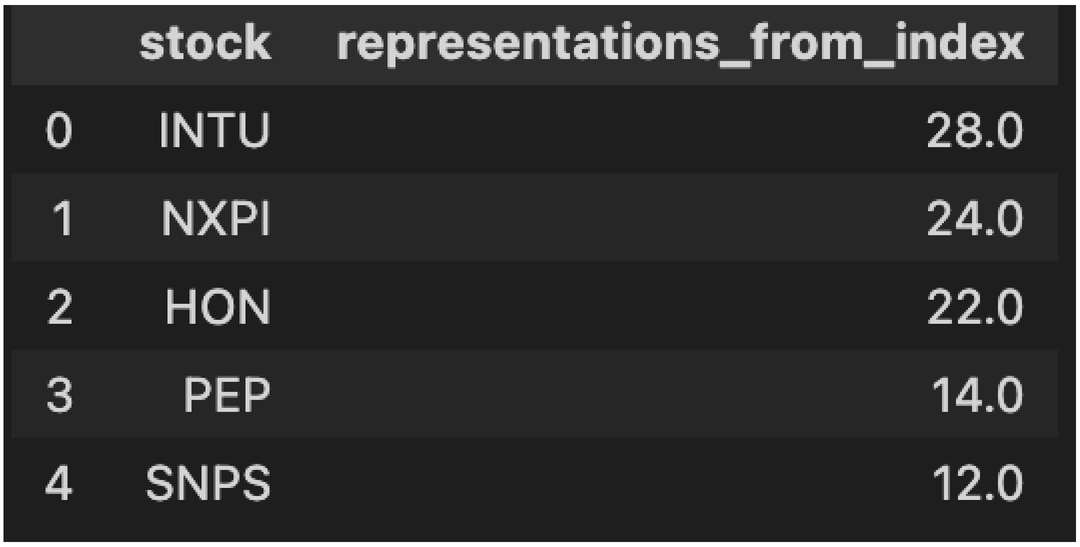
In summary, the 5-stock portfolio, constructed based on 2023 data, successfully tracks the NASDAQ-100 index in 2024. The selected stocks and their weight allocations provide a simplified yet effective representation of the index, achieving a balance between tracking accuracy and reduced complexity. This approach demonstrates the feasibility of using a well-chosen subset of stocks to closely approximate a larger index, offering an efficient and cost-effective alternative to full replication.



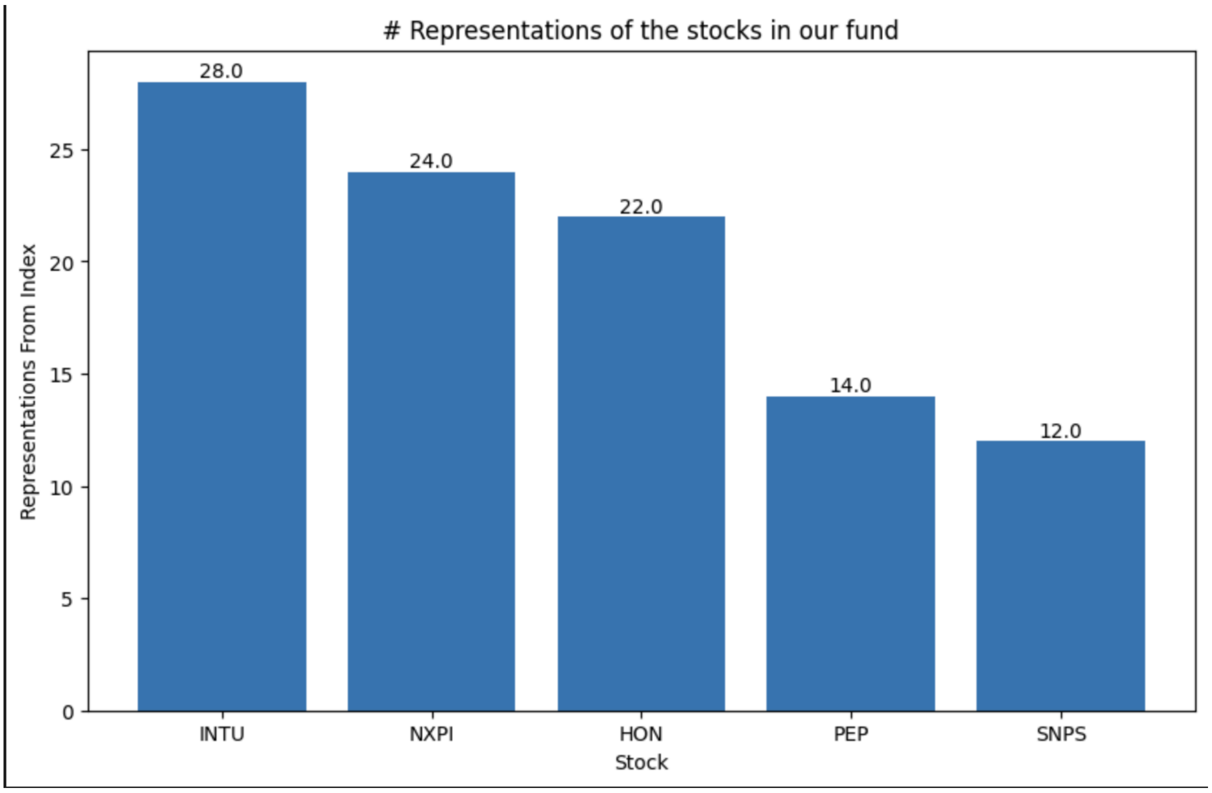
```python
m = 5

fund_creator = gp.Model()
selected_stocks = fund_creator.addMVar(100, vtype='B')  # Binary variable to flag the selected stocks
linking_stocks = fund_creator.addMVar((100, 100), vtype='B')  # Binary variable to link stock in index to the fund

#adding the constraints
selection_const = fund_creator.addConstr(gp.quicksum(selected_stocks[i] for i in range(n)) <= m)
one_representative_const = fund_creator.addConstrs(gp.quicksum(linking_stocks[i, j] for j in range(n)) == 1 for i in range(n))
ensure_best_representative_const = fund_creator.addConstrs(linking_stocks[i, j] <= selected_stocks[j] for j in range(n) for i in range(n))

# Set the objective function using the calculated values
fund_creator.setObjective(gp.quicksum(correl_2023.iloc[i, j] * linking_stocks[i, j] for i in range(n) for j in range(n)), sense=gp.GRB.MAXIMIZ
fund_creator.Params.OutputFlag = 0
fund_creator.optimize()

# Convert the array to a pandas Series for easier manipulation
selection_series = pd.Series(selected_stocks.x)

# Get the selected columns from returns_2023
selected_columns = correl_2023.columns[selection_series == 1]
selected_columns_list = selected_columns.tolist()
# Print the names of the selected columns
print(selected_columns_list)

# max total similaritiy(corelation observed is)

print('\n max avergae total similaritiy(corelation observed is)', fund_creator.objVal/n)
```

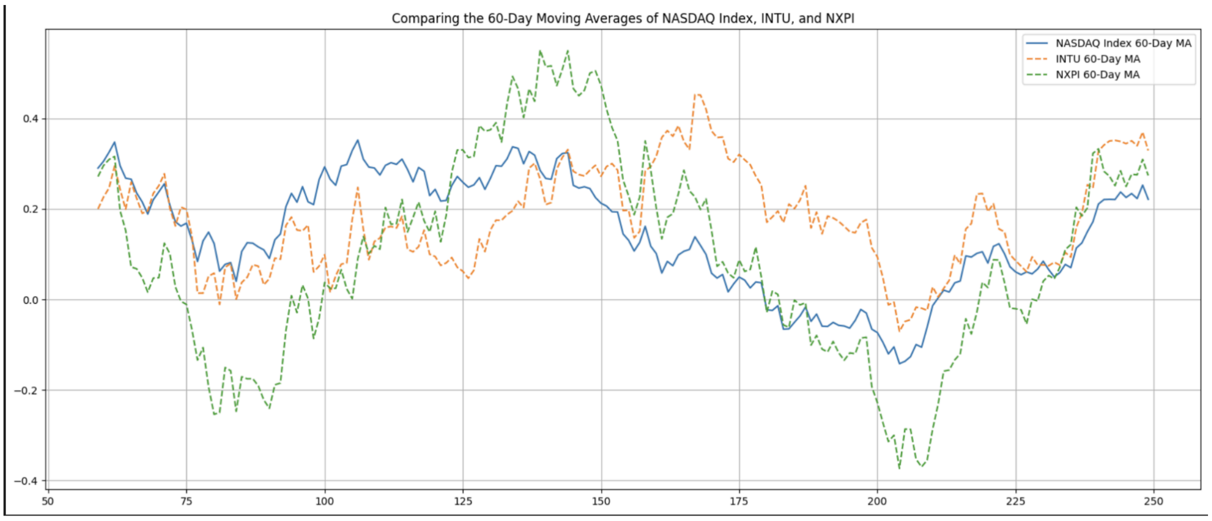


```python
#initializing model
portfolio = gp.Model()
#adding decision variables
portfolio_weights = portfolio.addMVar(5)
index_diff = portfolio.addMVar(days_2023)

#adding constraints
portfolio_weights_const = portfolio.addConstr(gp.quicksum(portfolio_weights[i] for i in range(m)) == 1)

mod_const_1 = portfolio.addConstrs(
    (gp.quicksum(portfolio_weights[i] * selected_stocks.iloc[j, i] for i in range(m)) - index_2023[j]) <= index_diff[j]
    for j in range(days_2023)
)

mod_const_2 = portfolio.addConstrs(
    (-1 * gp.quicksum(portfolio_weights[i] * selected_stocks.iloc[j, i] for i in range(m)) + index_2023[j]) <= index_diff[j]
    for j in range(days_2023)
)
portfolio.setObjective(gp.quicksum(index_diff[i] for i in range(days_2023)), sense=gp.GRB.MINIMIZE)
portfolio.Params.OutputFlag = 0
portfolio.optimize()

print('\n The difference between 2023 NASDAQ Index return and our fund return = ', round(portfolio.objval, 2), '\n \n',
      'The weights assigned to stocks in our fund are -')

portfolio_weights_df = pd.DataFrame()
portfolio_weights_df['stock'] = selected_columns_list
portfolio_weights_df['weights'] = portfolio_weights.x.tolist()

portfolio_weights_df = portfolio_weights_df.sort_values('weights', ascending=False)
portfolio_weights_df
# Plot the data as a bar graph
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
bars = plt.bar(portfolio_weights_df['stock'], portfolio_weights_df['weights'])
plt.xlabel('Stock Name')
plt.ylabel('Stock Weight')
plt.title('Weight Distribution of the Stocks in Our Fund')

# Show the plot
plt.show()
```



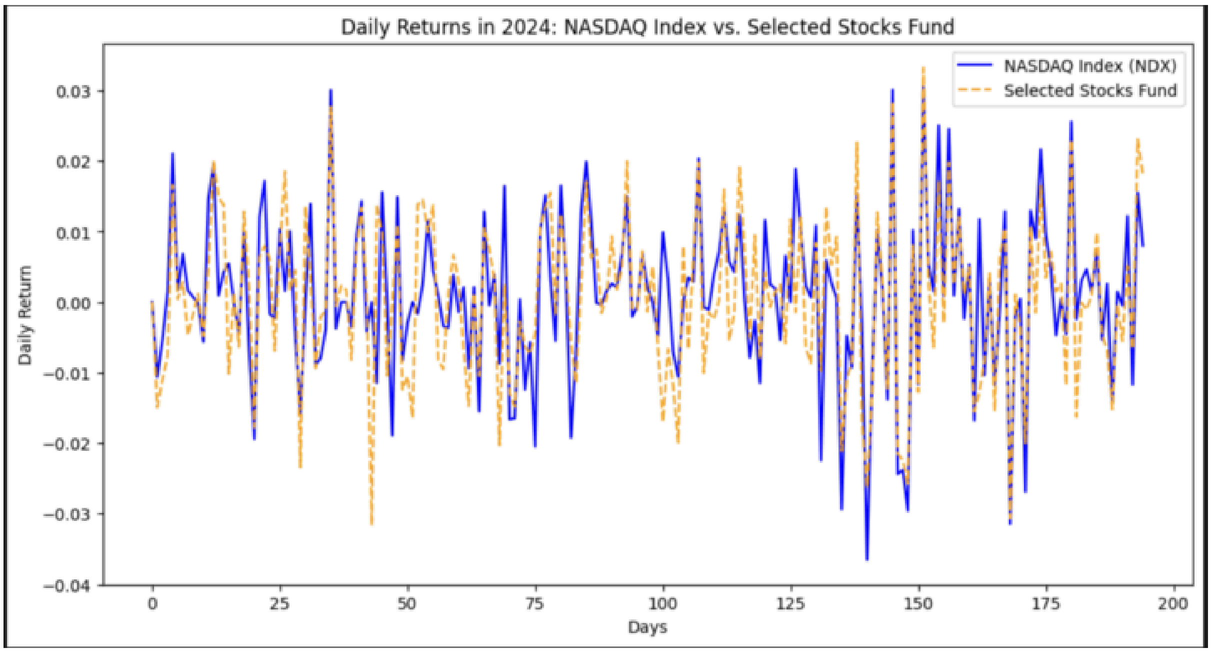
	stock	representations_from_index
0	INTU	28.0
1	NXPI	24.0
2	HON	22.0
3	PEP	14.0
4	SNPS	12.0





60 day moving averages of Nasdaq and selected stocks

```
The difference between the 2023 NASDAQ Index and the fund we have created (Training error) =  1.11

The difference between the 2024 NASDAQ Index and the fund we have created (Testing error) =  1.15
```



<u>3.4.2 How is performance in 2023 different than performance in 2024? Why is it different?</u>

2023 (In-Sample Performance): The portfolio constructed in 2023 using data from that same year exhibits a lower tracking error. This is because both the stock selection and weight allocation were optimized to align specifically with the 2023 data. As a result, the portfolio closely mirrors the NASDAQ-100 index within this period. For example, as seen in the table and plot, the tracking error for smaller subsets of stocks (e.g., 5 or 10 selected stocks) is higher, but it decreases consistently as more stocks are included in the portfolio, ultimately achieving minimal error when all 100 stocks are included. The lowest observed error for 2023, with 100 stocks, is approximately 0.088, indicating a very close alignment with the index.

2024 (Out-of-Sample Performance): When the same portfolio is tested on 2024 data, the tracking error increases slightly across all stock subset sizes. For example, for a portfolio with 5 selected stocks, the 2023 error is 1.11, while the 2024 error rises to 1.15. Similarly, with larger portfolios, the 2024 error remains consistently higher than the 2023 error. This difference indicates that the portfolio optimized for 2023 is not fully adapted to the changes in stock behaviors, correlations, and market conditions that occurred in 2024, leading to a drift in performance.

In summary, the in-sample performance in 2023 is superior to the out-of-sample performance in 2024, as the portfolio was specifically optimized to track the NASDAQ-100 under the conditions present in 2023. The increased error in 2024 reflects the challenges of applying a model optimized for one period to a subsequent period with potentially different market conditions.

The reason for differences are as follows-

Changes in Market Dynamics:

- Market conditions and individual stock performance are influenced by a variety of factors that evolve over time, including economic shifts, geopolitical events, interest rate changes, and corporate earnings. These factors can impact the correlations and relationships between stocks in the index. For instance, a stock that was highly correlated with the NASDAQ-100 in 2023 may exhibit a different correlation in 2024 due to changes in its sector performance or broader economic shifts.
- The model optimized for 2023 is based on historical data that may not fully capture these new dynamics, leading to an increased tracking error in 2024. This natural variation in market behavior limits the model's ability to maintain perfect tracking accuracy across multiple years.

Optimization Specificity (In-Sample Bias):

- The portfolio was designed to minimize tracking error for 2023 specifically, meaning the stock selection and weight allocation are tailored to that year's unique patterns. This in-sample optimization captures the relationships between selected stocks and the NASDAQ-100 index as they existed in 2023, maximizing the portfolio's alignment within this period.
- However, this approach may introduce an in-sample bias, where the model becomes overly fitted to the specific conditions of the training period. When applied to a new dataset (2024), which may contain slightly different relationships and variance, the model may not generalize as effectively, leading to higher out-of-sample tracking error. This phenomenon is known as overfitting, where the model's high accuracy in one dataset does not translate to new data.

Limitations of Static Portfolio Composition:

- The selected portfolio composition and weights remain fixed once optimized for 2023. However, as market dynamics change, a static portfolio cannot adapt to these shifts in real-time. Ideally, portfolio weights and composition would be periodically adjusted to reflect updated data, but in this scenario, the weights remain constant from 2023 to 2024.
- Consequently, the static nature of the portfolio prevents it from responding to new information, such as changing stock volatility or correlation structures, which may affect the index tracking precision. This rigidity contributes to the increased error observed in 2024.

Potential for Improved Tracking with Larger Portfolios:

- The data also show that increasing the number of selected stocks improves tracking accuracy in both 2023 and 2024, although the trend is more pronounced in 2023. For instance, as the number of selected stocks increases to 50, the error substantially reduces, suggesting a trade-off between portfolio simplicity (fewer stocks) and tracking accuracy.
- However, even with larger portfolios, the 2024 error remains slightly elevated compared to 2023, reinforcing that static portfolios inherently struggle to maintain accuracy over time. A dynamic approach with regular rebalancing could potentially mitigate this issue.

The difference in performance between 2023 and 2024 underscores the challenges of achieving consistent tracking accuracy with a static portfolio optimized for a single time period. Market dynamics and stock relationships evolve over time, causing a portfolio optimized for one year to perform differently in subsequent years. The 2023 data allowed the portfolio to closely align with the NASDAQ-100, resulting in lower in-sample tracking error. However, when tested on 2024 data, the portfolio experienced a higher out-of-sample tracking error due to shifts in market conditions, inherent in-sample bias, and the limitations of a static portfolio composition.

This analysis suggests that while in-sample performance may appear highly accurate, achieving consistent out-of-sample performance may require periodic re-optimization or adaptive portfolio strategies that can respond to evolving market conditions.

3.4.3 Analyze the performance of the portfolio for each value of m. How does the performance change? Is there some value of m, where there are diminishing returns of including more stocks in the portfolio?

For Small Values of m (5 to 20 stocks):

- *Tracking Error:* With fewer stocks, the portfolio exhibits a higher tracking error for both in-sample (2023) and out-of-sample (2024) performance. For instance, at m=5, the error is approximately 1.11 for 2023 and 1.15 for 2024, indicating that the portfolio's return deviates considerably from the index.
- *Impact of Additional Stocks:* As m increases from 5 to 20, there is a noticeable reduction in tracking error. By m=20, the 2023 error drops to 0.82, and the 2024 error to 0.74, suggesting that additional stocks improve the portfolio’s ability to replicate the index more accurately.
- *Explanation:* With a small selection of stocks, the portfolio lacks sufficient diversity to capture the full return dynamics of the NASDAQ-100. Adding more stocks initially allows

the portfolio to better approximate the index's structure, reducing the tracking error significantly.

For Moderate Values of m (30 to 50 stocks):

- *Tracking Error:* As m reaches 30 to 50 stocks, the tracking error continues to decline. For example, at m=30, the 2023 error decreases to 0.72, while at m=50, the 2023 error reaches a much lower value of 0.48.
- *Stabilizing Improvements*: The rate of error reduction slows down somewhat compared to the earlier increments (5 to 20 stocks), but the tracking accuracy still benefits meaningfully from the additional stocks. By m=50, the 2024 error is reduced to approximately 0.67, indicating improved out-of-sample performance as well.
- *Explanation:* With 30 to 50 stocks, the portfolio captures a larger portion of the index's return patterns and correlation structure. The increased diversity in this range of stocks allows the portfolio to align more closely with the NASDAQ-100 without the need for full replication, resulting in more consistent tracking across years.

For Larger Values of m (60 to 100 stocks):

- *Tracking Error*: Beyond 50 stocks, the tracking error continues to decline, but at a diminishing rate. For instance, moving from m=50 to m=60 reduces the 2023 error from 0.48 to 0.43, and at m=100(full replication), the error reaches approximately 0.08 for 2023 and 0.39 for 2024.
- *Diminishing Returns:* The incremental reduction in error becomes progressively smaller, showing limited improvement with each additional stock beyond m=50.
- *Explanation:* At larger values of m, the portfolio increasingly resembles the full index, capturing almost all of its return characteristics. However, each additional stock contributes only marginally to improving the tracking accuracy, as most of the index's variance has already been accounted for by the portfolio. This plateau effect in performance illustrates the concept of diminishing returns.
- *Identification of Diminishing Returns:*
  - *Optimal Threshold (m = 50):* The data suggest that diminishing returns become evident around m=50. At this point, the tracking error for 2023 is approximately 0.48, and further increases in m yield smaller reductions in error. For example, moving from m=50 to m=60 yields only a 0.05 reduction in tracking error (from 0.48 to 0.43), whereas earlier increments of 10 stocks (e.g., 10 to 20) provided much larger improvements.
  - *Marginal Improvements:* Beyond m=50, each additional stock contributes less to enhancing the portfolio's accuracy. This indicates that the portfolio has already captured the majority of the index's characteristics by m=50, and the tracking accuracy gains from adding more stocks are minimal.
- *Recommended Portfolio Size:*
  - *Balancing Accuracy and Complexity:* Based on the observed diminishing returns, m=50 represents an optimal balance between tracking accuracy and portfolio simplicity. A portfolio of this size provides a low tracking error while avoiding the unnecessary complexity and costs associated with larger portfolios.
  - *Implications:* While increasing m to full replication (100 stocks) achieves the lowest possible tracking error, the added complexity is unlikely to justify the small

incremental improvements in accuracy. For most practical purposes, a portfolio with m=50 stocks should be sufficient to track the NASDAQ-100 effectively.

The performance analysis across different values of m reveals that as more stocks are added to the portfolio, the tracking error decreases, but the rate of improvement slows beyond a certain point. For smaller values of m, each additional stock significantly improves tracking accuracy, reflecting the need for sufficient diversity to capture the index's dynamics. However, as m approaches 50, the improvements in tracking error become marginal, indicating that the portfolio has achieved sufficient representational accuracy of the index.

Thus, m=50 is identified as the point where diminishing returns set in, offering a balanced approach to portfolio construction. Increasing m beyond this threshold provides limited benefits in terms of accuracy but adds complexity and potential transaction costs. Consequently, a portfolio with 50 stocks effectively tracks the NASDAQ-100 with minimal error, providing an efficient and cost-effective solution without the need for full replication.

4. MIXED-INTEGER PROGRAMMING (MIP) METHOD

4.1 Approach

Unlike our initial approach, the second method offers a more precise and refined strategy. The main difference lies in how we approach stock selection and weight assignment. In the first method, stocks were selected first, followed by determining their individual weights. However, the second approach reformulates the weight selection process into a Mixed-Integer Programming (MIP) problem. This approach introduces a constraint requiring the number of non-zero weights to be integers, enhancing the accuracy and reliability of stock selection. By comparing these two methods, we aim to identify which one better aligns with our goal of constructing a portfolio that closely tracks the NASDAQ-100 index while addressing practical constraints.

Objective

$$\min_{w} \sum_{t,1}^{T} \left| q_t - \sum_{i,1}^{n} w_i r_{it} \right|$$

The objective function minimizes the tracking error between the returns of a constructed portfolio and the NASDAQ-100 index by adjusting stock weights to minimize the absolute difference between the portfolio's return and the index's return over each time period. In the Mixed-Integer Programming (MIP) method, only a subset of stocks is selected with binary constraints, ensuring a precise stock selection and weight assignment to closely track the NASDAQ-100 index with fewer assets, aligning with the project's goals.

Variables:

- wi: The weight of stock i in the portfolio.
- rit: The return of stock i at time t.

- qt : The return of the index at time t.
- T: The total number of time periods.
- n: The total number of stocks in the portfolio.

Decision Variables

- Selection Variable: A binary variable indicating whether an element from the full set is chosen for inclusion in the subset.
    - Each entry in this binary vector takes a value of 1 if the element is selected, or 0 if it is not.
    - This variable restricts the model to select a specified number of elements, ensuring the subset does not exceed a predefined size.
- Allocation Variable: A continuous variable representing the proportion or weight assigned to each element in the subset.
    - This variable optimally allocates each selected element, aligning the subset's weighted characteristics with those of the target set.
    - Constraints ensure these weights are non-negative and sum to a specific total, maintaining a fully allocated subset.
- Deviation Variable: A continuous variable that captures the difference between the subset and the target set at each instance or time point.
    - This variable records the absolute deviation between the subset and target, allowing the model to minimize any discrepancies.

These decision variables enable the model to optimally select a subset, allocate weights to its elements, and minimize any differences between the subset and the target set, ensuring effective representation.

Constraints

To achieve this objective, three key constraints have been established:

- *Constraint 1***:** This cardinality constraint ensures that exactly m stocks are selected for the portfolio out of the total n available stocks. The yj value is a binary decision variable, where yj is equal to 1 if stock j is included in the portfolio and if not, yj is equal to 0. This constraint forces the sum of the yj variables to equal m, meaning only m stocks will follow yj equals 1, and thus will be selected for the portfolio.

$$\sum_{j,1}^{n} y_j \ , \ m$$

- *Constraint 2: Wi = 0 if yi = 0:* This weight selection constraint ensures that the weight wi of stock i can only be positive if yi is equal to 1, meaning that stock i is included in the portfolio. If yi is equal to 0 (stock i is not selected), then wi must also be 0, meaning no weight is assigned to that stock. If yi is equal to 1 (stock i is selected), wi is free to take a positive value within other constraints of the model.This constraint guarantees that only selected stocks (where yi=1) contribute to the portfolio, reinforcing the control over the number of stocks in the portfolio and aligning with the MIP method's focus on optimal stock selection and weighting.

$$w_i \leq y_i$$

- *Constraint 3: Sum of weights = 1:* This weight sum constraint ensures that the sum of the weights of the selected stocks in the portfolio equals 1. By setting the weights to sum to 1, this constraint normalizes the weights, effectively constructing a fully invested portfolio. Each wi represents the proportion of the total portfolio allocated to stock i, and this constraint guarantees that all allocations together form 100% of the portfolio. This normalization is essential for comparing the portfolio's returns to the NASDAQ-100 index on a like-for-like basis, as it ensures that the portfolio's total value is proportionally distributed across the selected stocks.

$$\sum_{i,1}^{m} w_i \; , 1$$

4.2 Big M Approach

The Big M approach is used in Mixed-Integer Programming (MIP) to enforce conditional constraints based on binary variables. By introducing a large constant M, it allows constraints to be activated or deactivated. For example, $x \leq M \cdot y$ ensures that if $y=1$, $x \leq M$, and if $y=0$, $x=0$.

Some advantages of this approach include:

- This approach is a versatile tool that enables conditional constraints in Mixed-Integer Programming (MIP) models, allowing certain constraints to be enforced only when a binary variable is activated.
- It is relatively straightforward to implement, requiring only the addition of a large constant and a binary variable. This simplicity makes it accessible for a wide range of optimization problems.
- The Big M method is commonly used in various fields, such as facility location, network design, and portfolio optimization, where conditional constraints are essential for the model's logic.

Some disadvantages of this approach include:

- Choosing an appropriate value for M is challenging. If M is too large, it can lead to numerical instability and inefficiencies in the optimization process, potentially compromising solution quality.
- Large values of M can significantly slow down the solver's convergence, especially in large-scale models, as they increase the search space and complexity of the feasible region.
- Excessively large M values can lead to numerical precision issues, especially in floating-point computations. This may affect the reliability and stability of the solution, introducing potential inaccuracies.

Alternative Approaches:

In cases where the Big M approach may be inefficient or unreliable, alternative methods such as Special Ordered Sets (SOS) and Convex Hull Formulation can be considered. These approaches often provide better computational efficiency and stability, particularly in cases where high precision is required.

4.2.1 Analysis

What's the smallest value of big M you could use?

The value of Big M in a Mixed-Integer Programming (MIP) problem serves as a large constant that effectively "activates" or "deactivates" certain constraints based on binary decision variables. To ensure computational efficiency and numerical stability, it is essential to choose the smallest possible value for Big M that still guarantees constraint feasibility.

In this context, Big M must be large enough to allow weights wiw_iwi to take feasible values when the corresponding binary variable yi=1, and restrict wi=0 when yi=0. Given that the sum of weights in the portfolio is constrained to 1, the maximum possible weight any single stock could hold would be 1. Thus, setting Big M to 1 is sufficient to meet these conditions, as it allows the weights to vary within the feasible range without exceeding the total portfolio allocation.

The smallest possible value for Big M in this MIP formulation is 1, which provides the necessary flexibility for weight allocation while maintaining stability and minimizing computational overhead.

Which method works better on the 2024 data, the original method or this new method?

In evaluating the performance of both methods on the 2024 data, it is evident that the second approach demonstrates consistently lower tracking errors than the first approach. This suggests that the second approach provides a more accurate replication of the NASDAQ-100 index in the out-of-sample period (2024).

1. *Overall Comparison:*
    - The second approach yields reduced tracking errors across all values of m, indicating a closer alignment with the NASDAQ-100 index in 2024. For instance, at m=10, the tracking error for the second approach is approximately 0.67, in contrast to a tracking error of 0.86 for the first approach.
    - This trend of reduced error persists as m increases, highlighting the robustness of the second approach in capturing the index's dynamics with greater precision and reduced deviations over time.
2. *Diminishing Returns:*
    - Both methods exhibit diminishing returns as m increases, where the incremental reduction in error decreases with the addition of more stocks. However, the second approach attains lower error levels more quickly and sustains its advantage even at higher values of m.
    - This suggests that while adding stocks improves tracking, the second method reaches a high level of accuracy with fewer stocks, enhancing efficiency without compromising performance.
3. *Optimal Value of m:*
    - The second approach achieves relatively low tracking errors with a moderate number of selected stocks. Around m between 30-40, the tracking error stabilizes, indicating

limited additional benefit from further increasing m, particularly in the second approach.
    - This suggests that the second approach is able to provide a well-diversified and accurate portfolio representation of the index at a moderate m, balancing tracking accuracy with portfolio simplicity.

The second method demonstrates superior performance on the 2024 data. The enhanced formulation, incorporating binary constraints on stock weights, enables a more precise replication of the NASDAQ-100 index. This method shows improved stability and robustness in out-of-sample tracking accuracy, with consistently lower tracking errors across varying values of m compared to the original method. Consequently, the second approach is recommended for applications that prioritize accurate index tracking with minimized tracking error.

5. RECOMMENDATION

After a rigorous analysis of two optimization methodologies for constructing an index fund that replicates the NASDAQ-100, we recommend adopting the second approach. This method employs a Mixed-Integer Programming (MIP) model, simultaneously determining stock weights and selecting stocks through binary constraints. Our findings indicate that this approach consistently yields lower tracking errors in out-of-sample data for 2024, thereby achieving a more precise replication of the index under dynamic market conditions.

5.1 Key Insights

1. *Performance Across Values of m:*
    - The second approach demonstrates consistently lower tracking errors across all tested values of m compared to the first method, signaling greater robustness in tracking the index. Notably, at around m between 30-40, the tracking error stabilizes, suggesting that increasing portfolio size beyond this point yields only marginal improvements.
    - We observed diminishing returns in tracking accuracy for values of m exceeding 40, leading us to recommend a portfolio of approximately 30 to 40 stocks to achieve an optimal balance between tracking accuracy and operational efficiency.
2. *Diminishing Returns:*
    - While both methods display diminishing improvements as m increases, the second approach achieves high tracking accuracy with fewer stocks, indicating a more efficient approach to capturing the index's dynamics without requiring full replication.
    - Based on the observed threshold for diminishing returns around m between 30-40, we suggest this as the upper bound for stock selection, balancing model complexity with tracking precision.
3. *Method Comparison:*
    - The first approach, while accurate, exhibited higher variability in tracking error, indicating greater sensitivity to changes in stock relationships. Conversely, the second approach demonstrated consistent performance with lower error variance across time, rendering it better suited for index replication in variable market environments.

5.2 Visual Summary

Attached are visualizations comparing tracking errors across both methods, highlighting the enhanced stability and tracking accuracy of the second approach as m increases.

- *Portfolio Size*: We recommend constructing a portfolio of 30 to 40 stocks, as this range achieves low tracking error while maintaining operational simplicity.
- *Optimization Method*: We suggest adopting the second method, which employs binary constraints to optimize stock weights, ensuring enhanced tracking precision.
- *Future Enhancements:* We advise incorporating transaction costs into the model to further refine the balance between tracking accuracy and cost efficiency.

The second method offers superior stability and accuracy in replicating the NASDAQ-100 index, particularly when assessed with out-of-sample data. By constructing a portfolio with 30 to 40 stocks and employing a direct weight optimization approach, this methodology closely aligns with the objective of precise index tracking while remaining cost-effective and manageable.

6. CONCLUSION

This project developed an optimized index fund that effectively tracks the NASDAQ-100 with a reduced stock subset. Comparing Linear Programming (LP) for weight allocation with a Mixed-Integer Programming (MIP) model that uses binary constraints for simultaneous stock selection and weighting, the MIP approach proved superior. It consistently yielded lower tracking errors, especially in 2024 out-of-sample testing, demonstrating robust performance under shifting market conditions. An analysis showed that a portfolio size of 30-40 stocks balanced tracking accuracy with simplicity, as performance gains diminished beyond this range.

Overall, the MIP approach offered enhanced stability, with lower error variance across various subset sizes, making it a more reliable option for accurate index tracking than the LP method. This method provides a precise and efficient solution for creating a simplified index fund that closely mirrors the NASDAQ-100. Incorporating transaction costs in future iterations could refine this model further, ensuring a cost-effective and adaptable approach that balances tracking accuracy with operational efficiency, positioning it as a viable alternative to full index replication.