THE UNIVERSITY OF TEXAS AT AUSTIN

McCOMBS SCHOOL OF BUSINESS

RM 294 - Optimization I
Dr. Daniel Mitchell

Project 2 - Integer Programming

Group 15
Kavya Angara (ka32577)
Srividya Rayaprolu (lr34488)
Pratik Gawli (pbg397)
Amrit Sandhu (ams22664)

Table of Contents

1. Introduction

With the unlimited information present today, investors should find efficient and smart ways to maximize returns while trying to minimize initial outlay and risk. Passive investing and active investing are two such contrasting strategies for putting money to work in markets. Active investing generally looks to beat the benchmark whereas passive investing aims to duplicate its performance.

“Indexing” is a form of passive fund management. Instead of actively selecting securities to invest in and planning when to acquire and sell them, a fund manager develops a portfolio whose holdings reflect those of a specific index. The theory is that by closely matching the index's profile—the stock market overall or a significant portion of it—the fund will match its performance. One example of a fund that tracks the NASDAQ-100 index is the QQQ Index Fund.

Since indexes are hypothetical portfolios, they can often follow unrealistic assumptions such as continuous rebalancing. This makes it difficult for fund managers to mimic the holdings of an index in real-life since frequent rebalancing has significant trading costs attached to it. We are proposing an alternative investment strategy that does away with frequent rebalancing requirements but can still closely approximate the returns of indexes.

To achieve this we wrote two optimization programs that try to best mimic the returns of an index by buying only a limited number (m) of the index’s holdings in various proportions.

We test our program by attempting to track the NASDAQ-100 index. We used 2019 data for fitting our optimization models and will perform our testing against 2020 (out-of-sample) data.

2. Method 1 - IP Optimization

2.1 Approach

Our investment strategy for method one follows two steps to determine the portfolio holdings:

1. **Stock selection**: Deciding which ‘m’ stocks to buy.
2. **Portfolio weights calculation**: Deciding how much of each stock to buy

2.1.1 Stock Selection

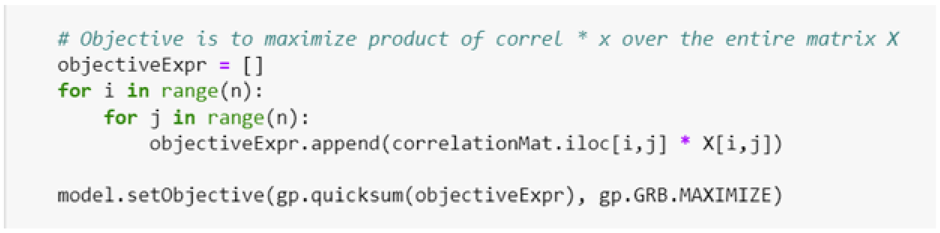
Step 1

We will first solve an integer problem where we will select m stocks out of n stocks for the portfolio we would like to create. To achieve the same, we created a correlation matrix, which will be used to represent correlation between the stock. Each element ρij in the correlation matrix will tell us the correlation between stock i and stock j.

As we began formulating our integer program, we created a matrix X of size n^2 where n is the number of all stocks and we created a vector of binary variables called y of size n. Considering X_{ij} as an element from X matrix, X_{ij} will tell us whether stock i is the best representative of stock j or not. The element y_j in y vector will tell us whether stock j is being considered in our index fund or not.

The objective vector was identified:

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



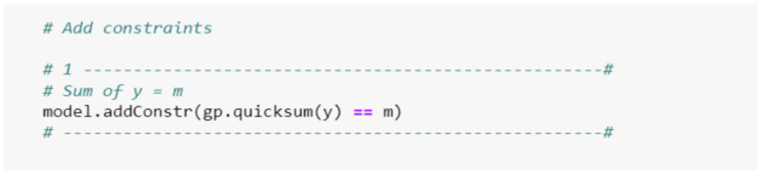
```
# Objective is to maximize product of correl * x over the entire matrix X
objectiveExpr = []
for i in range(n):
    for j in range(n):
        objectiveExpr.append(correlationMat.iloc[i,j] * X[i,j])

model.setObjective(gp.quicksum(objectiveExpr), gp.GRB.MAXIMIZE)
```

And the constraints identified are as follows:

1. The number of stocks to be selected in the index fund should be exactly ‘m’. This is to align with our goal of investing in the fewest number of stocks possible while maximizing return on investment.

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

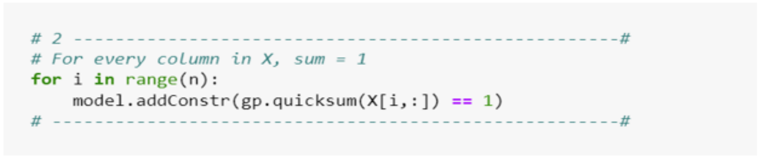


```
# Add constraints

# 1 ----------------------------------------------------#
# Sum of y = m
model.addConstr(gp.quicksum(y) == m)
# -----------------------------------------------------#
```

2. Each stock from the NASDAQ-100 index is exactly represented once in our index fund. This is in alignment with our goal of diversification of our index fund.

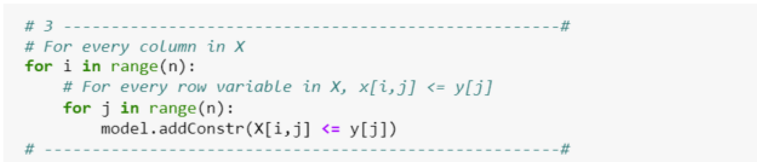
$$\sum_{j=1}^{n} x_{ij} = 1 \quad for\ i = 1,2,\ldots,n$$



```
# 2 -----------------------------------------------------#
# For every column in X, sum = 1
for i in range(n):
    model.addConstr(gp.quicksum(X[i,:]) == 1)
# ------------------------------------------------------#
```

3. Stock i from NASDAQ-100 index is best represented by Stock j if and only if Stock j is in our index fund

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



```
# 3 ---------------------------------------------------#
# For every column in X
for i in range(n):
    # For every row variable in X, x[i,j] <= y[j]
    for j in range(n):
        model.addConstr(X[i,j] <= y[j])
# ----------------------------------------------------#
```

We are using daily prices of index and the component stocks of the NASDAQ-100 in 2019 as input to construct our portfolio.

2.1.2 Calculating Portfolio weights

Step 2

The next step is to calculate the portfolio weights. We will solve this problem using Integer programming. The objective here is to minimize the return of the stock and the return of the index as closely as possible.

We can formulate it as below :

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

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

$$w_i \geq 0.$$

We can see from above that the minimization objective is non linear, hence will need to translate it to a linear objective function.

The new objective would be:

$$min \sum_{t=1}^{T} Z_t$$

```
# Objective is to minimize sum of z
model.setObjective(gp.quicksum(z), gp.GRB.MINIMIZE)
```

The identified constraints are:

1. Sum of the weights should be equal to 1.

$$\sum_{i=1}^{m} w_i = 1$$

```
# Weights sum to 1
model.addConstr(gp.quicksum(w) == 1)
```

2. We minimize the sum of absolute value of the differences in portfolio returns and index returns for over the entire time period.

$$Z_i \geq \sum_{j=0}^{m} w_j . r_j - q$$

$$for\ i = 0 \rightarrow T$$

$$Z_i \geq q - \sum_{j=0}^{m} w_j . r_j$$

$$for\ i = 0 \rightarrow T$$

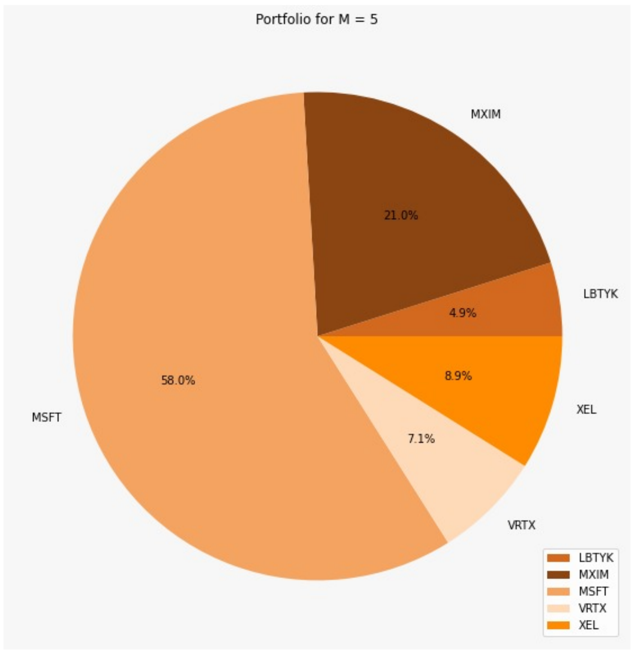
```
for i in range(len(returnsDf)):
    # For every row in returnsDf, sum of w * returns = z
    totalReturns = []
    for j in range(len(returnsDf.columns)):
        totalReturns.append(w[j] * returnsDf.iloc[i,j])

    indexReturn = indexDf.iloc[i]
    model.addConstr(gp.quicksum(totalReturns) - indexReturn <= z[i])
    model.addConstr(- gp.quicksum(totalReturns) + indexReturn <= z[i])
```

2.2 Constructing the Portfolio

Let us start by understanding the portfolio built using m=5.

We ran our optimization algorithm by setting m=5 i.e. selecting 5 stocks to represent all the stocks available in the NASDAQ-100. The output of stock selection and corresponding weightage is shown in the pie chart below:



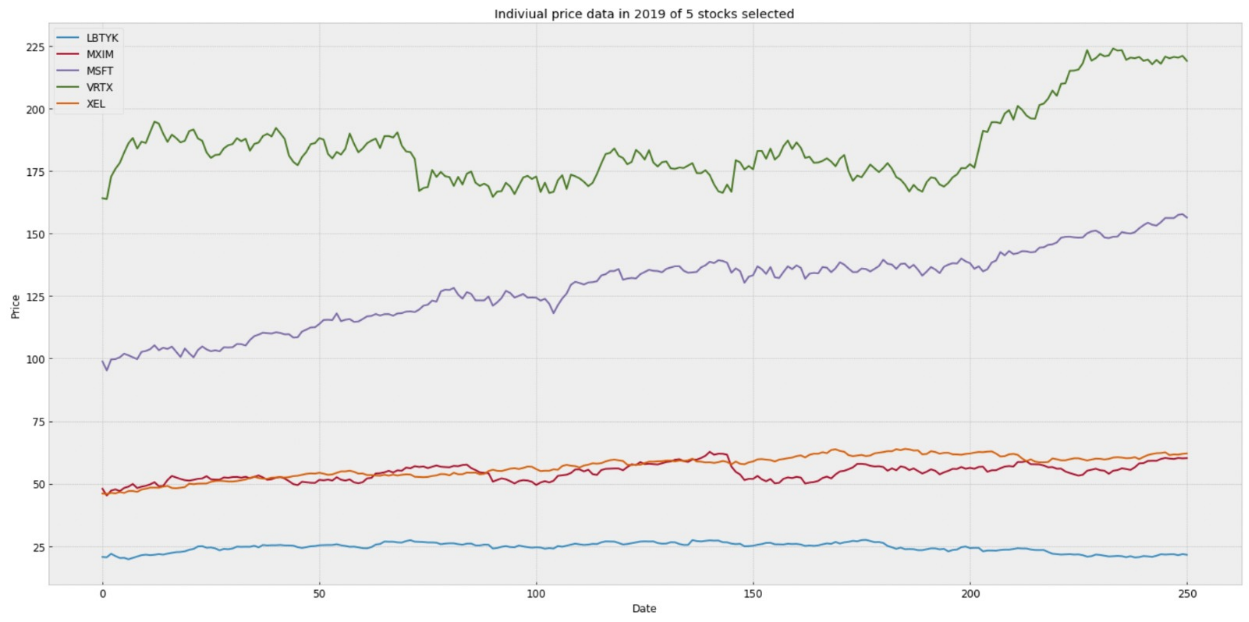
The details of the selected stocks are:

Stock	Ticker	Sector
Liberty Global	LBTYK	Telecommunication, Mass Media
Xcel Energy	XEL	Utilities
Maxim Integrated Products	MXIM	Electronics
Microsoft	MSFT	Information Technology
Vertex Pharmaceuticals	VRTX	Health Care

As we observe, the 5 selected stocks in our portfolio represent 5 different industries and the portfolio created is very diversified which is as desired since we would like our Index fund to be representative of all the industries present in NASDAQ 100 for accurate index tracking.

The industry with the maximum weightage i.e. 58% is Information technology which is in line with the top holdings in the NASDAQ-100 and utilities being the second largest industry with top holdings.

The plot shown below can be used to visualize the price fluctuations of the 5 stocks in our index fund for the year 2019 (i.e. 250 days):



Similarly, we constructed portfolios by changing the number of desired stocks in our index fund by running the optimization algorithm for the below values of m:

$$\mathbf{m \in \{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \}}$$

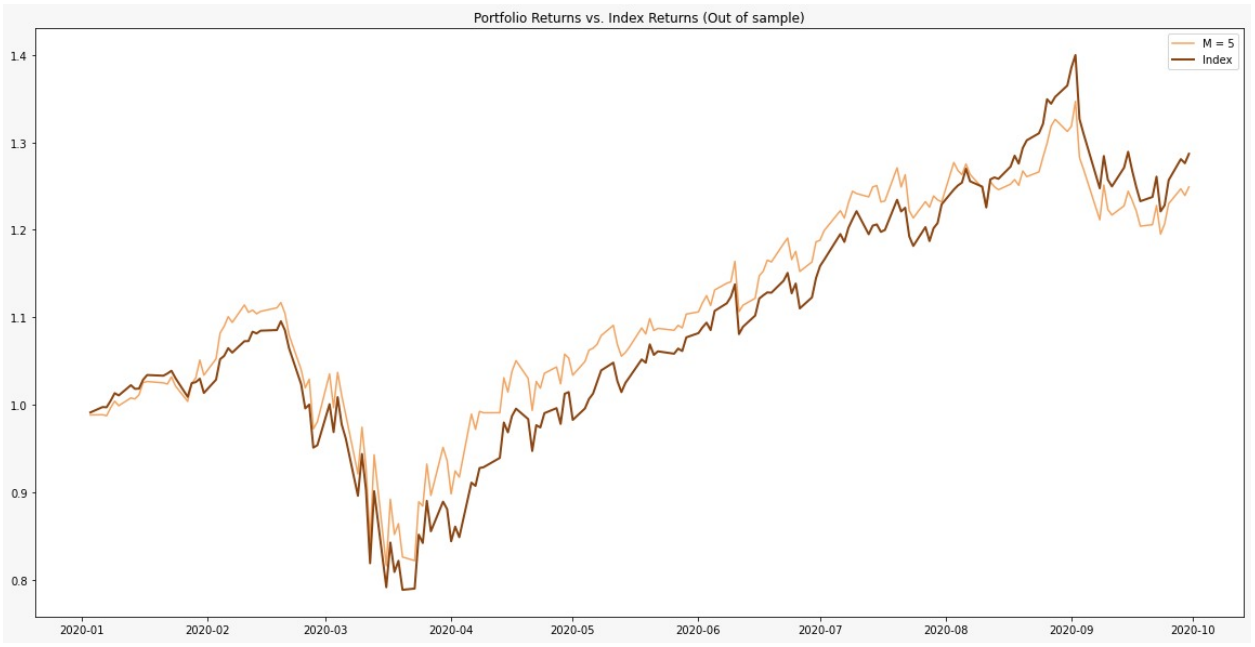
We observed that our program created portfolios with similar diversification across industries as the index for all values of m.

2.3 Evaluation of Portfolio

Now that we’ve taken a look at how the portfolio is constituted, let's evaluate the portfolio’s tracking accuracy by looking at various metrics.

We start our portfolio evaluation by looking at the portfolio for m = 5 stocks.

The below graph shows side-by-side comparison of portfolio returns vs. index returns for out-of-sample 2020 data against the output for our optimization program for m=5:

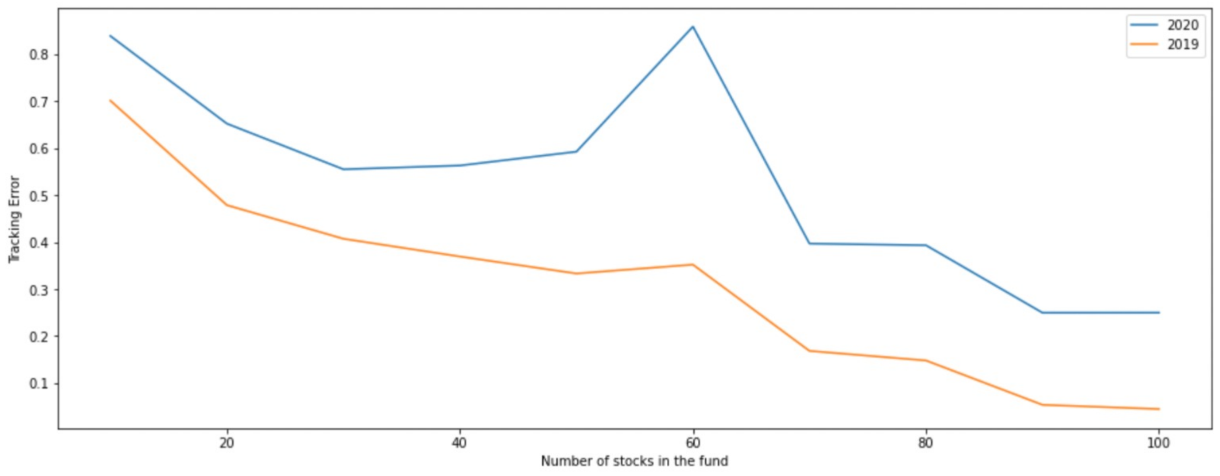


To see how well the portfolio created with m=5 tracks the NASDAQ-100, we calculated the sum of absolute error as below for out-of-sample data i.e. 2020:

$$\sum_{t=1}^{T} \left| q_t - \sum_{i=1}^{5} w_i r_{it} \right|$$

This resulted in an error as ~0.87.

Similarly, to evaluate the performance for each value of m, we calculated the sum of tracking error for in-sample data i.e. 2019 and out-of-sample data i.e. 2020 and the graph below represents the same:

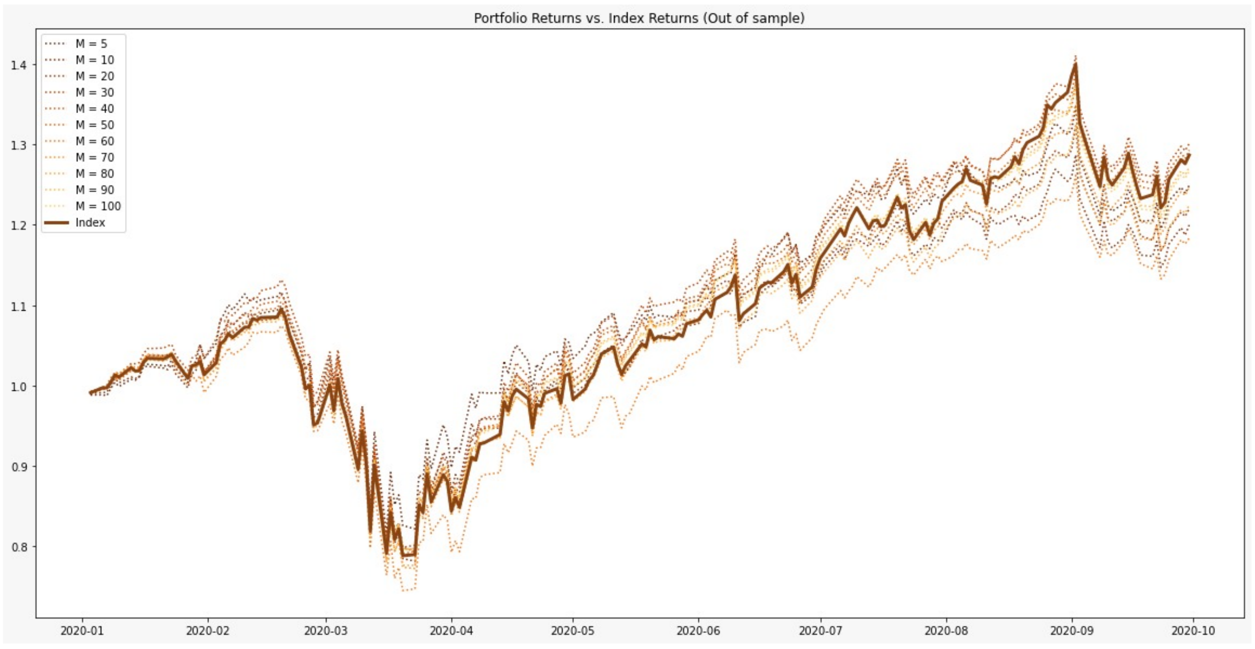


The performance of the portfolio improves as we increase the value of m i.e. we include more and more in our index fund.

We did not find any value of m beyond which there are diminishing returns on adding more stocks in the portfolio. This is true for in-sample data performance as well as out-sample data performance.

From the tracking error for both years, we observed that 2019's error rate is comparatively lower as compared to 2020's. This is as expected as the portfolios were built by using the weights from 2019 data as training data.

The below graph shows side-by-side comparison of portfolio returns vs. index returns for out-of-sample 2020 data against the output for our optimization program for all values of m:



3. Method 2 - MIP Optimization

3.1 Approach

In this approach, we did not formulate a separate stock selection Integer Programming problem instead we formulated one mixed integer programming problem (MIP). This MIP constraints the number of non-zero weights to be an integer.

We leveraged the weight selection problem used in method 1 by replacing 'm' with 'n', where 'm' was the number of stocks we selected for our index fund and 'n' was the total number of stocks.

In MIP, we optimized weights for all 'n' stocks. Then we defined y vector in which each element yi will tell us whether stock i is being considered or not.

The objective can be formulated as below:

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

```
# Objective is to minimize sum of z
model.setObjective(gp.quicksum(z), gp.GRB.MINIMIZE)
```

And the constraints identified are as follows:

1. To mimic the absolute value function, set z to be equal to the absolute value of differences in index and portfolio returns

$$Z_i \geq \sum_{j=0}^{m} w_j . r_j - q$$
$$for\ i = 0 \to T$$

$$Z_i \geq q - \sum_{j=0}^{m} w_j . r_j$$
$$for\ i = 0 \to T$$

```
for i in range(len(returnsDf)):
    # For every row in returnsDf, sum of w * returns = z
    totalReturns = []
    for j in range(len(returnsDf.columns)):
        totalReturns.append(w[j] * returnsDf.iloc[i,j])

    indexReturn = indexDf.iloc[i]
    model.addConstr(gp.quicksum(totalReturns) - indexReturn <= z[i])
    model.addConstr(- gp.quicksum(totalReturns) + indexReturn <= z[i])
```

2. This is the big M constraint, which limits the weight to zero if it not one of the ‘m’ chosen stocks

$$w_i \leq M.y_i \qquad for\ i = 0\ to\ n$$

```python
# Add big-M constraints
M = 1
for i in range(len(returnsDf.columns)):
    model.addConstr(w[i] <= M * y[i])
```

3. We set a total number of non-zero stocks to m.

$$\sum_{i=0}^{n} y_i = m$$

```python
# Sum of y = m
model.addConstr(gp.quicksum(y) == m)
```

4. The sum of portfolio weights has to be 1.

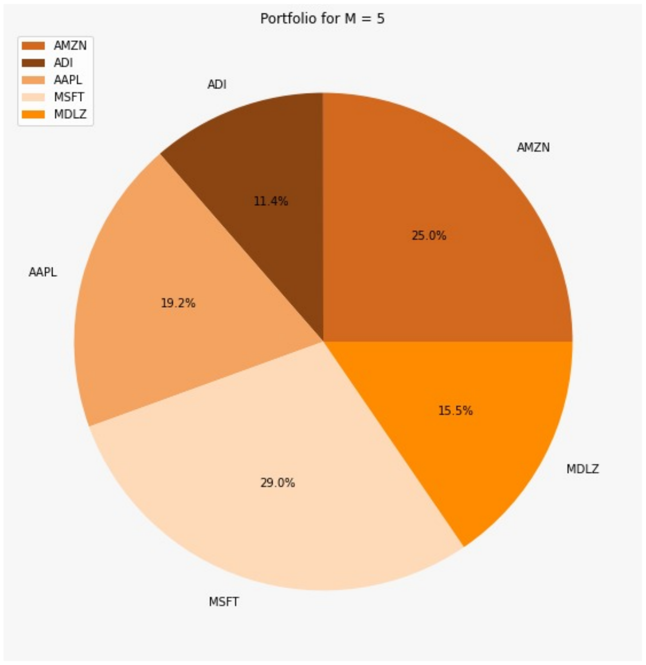
$$\sum_{i=0}^{n} w_i = 1$$

```python
# Weights sum to 1
model.addConstr(gp.quicksum(w) == 1)
```

3.2 Evaluation of Portfolio

Let us start by evaluating the portfolio built using m=5.

In this method, we constructed the portfolio by setting up the number of non zero weights to five (m = 5) i.e. selecting 5 stocks to represent all the stocks available in the NASDAQ-100. The output of stock selection and corresponding weightage is shown in the pie chart below:

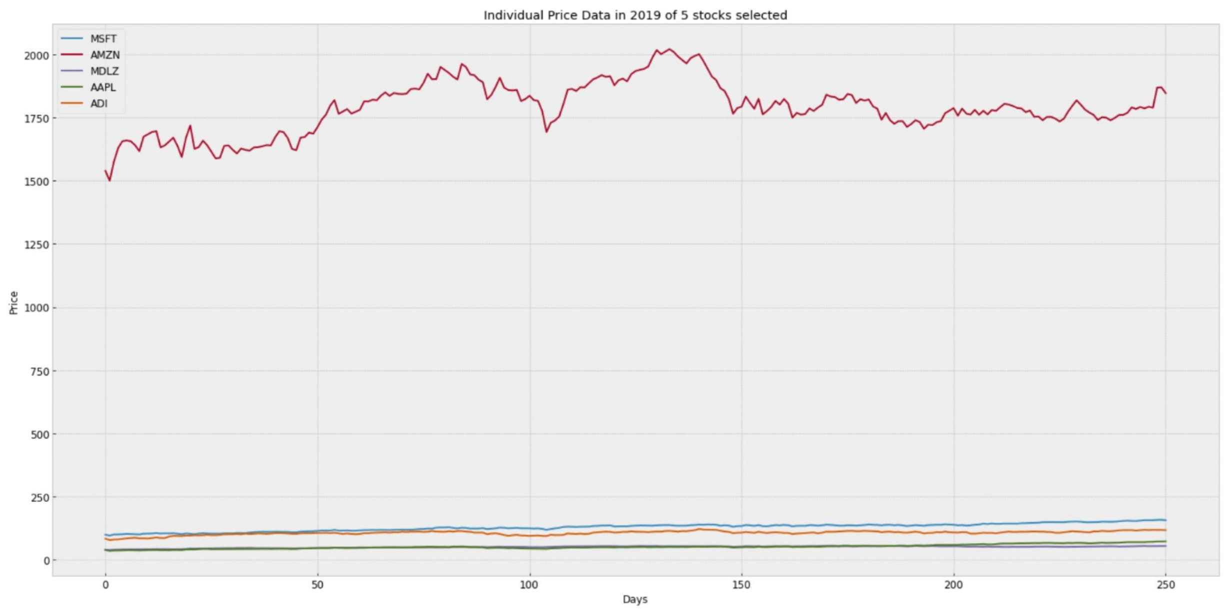


The details of the selected stocks are:

Stock	Ticker	Sector
Microsoft	MSFT	Information Technology
Amazon	AMZN	Ecommerce, Cloud, AI
Mondelez	MDLZ	Food and beverages
Apple	APPL	Technology
Adler	ADL	Real estate

As we observe, the 5 selected stocks in our portfolio represent top holdings in the NASDAQ-100. 3 of the stocks selected by our optimization program exist in the top 5 holdings NASDAQ-100.

The plot shown below can be used to visualize the price fluctuations of the 5 stocks in our index fund for the year 2019 (i.e. 250 days):



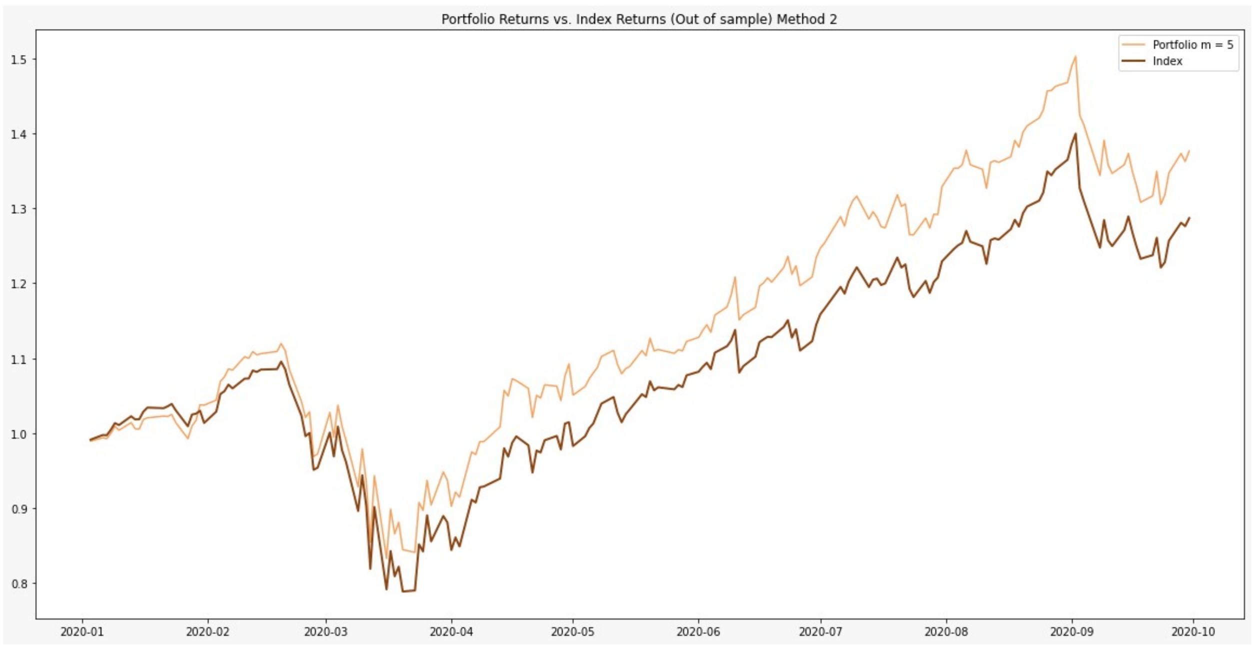
Similarly, we constructed portfolios by changing the number of desired stocks in our index fund by running the optimization algorithm for the below values of m:

$$\mathbf{m \in \{ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 \}}$$

We were able to observe that the output of the optimization program was able to create similar portfolios for each value of m as that of NASDAQ-100 top holdings instead of focusing on diversifying the portfolio like method 1.

To evaluate the performance of a portfolio created using the number of stocks in our index fund as m=5, we compared the results for our portfolio against the index of NASDAQ-100 for 2020 data.

The below graph shows side-by-side comparison of portfolio returns vs. index returns for out-of-sample 2020 data against the output for our optimization program for m=5:

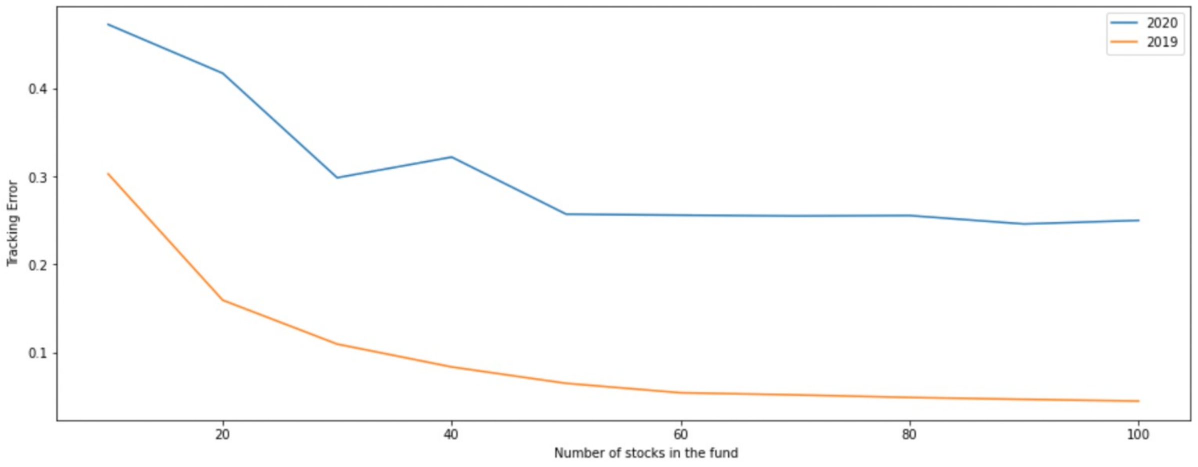


To see how well the portfolio created with m=5 tracks the NASDAQ-100, we calculated the sum of absolute error as below for out-of-sample data i.e. 2020:

$$\sum_{t=1}^{T} \left| q_t - \sum_{i=1}^{5} w_i r_{it} \right|$$

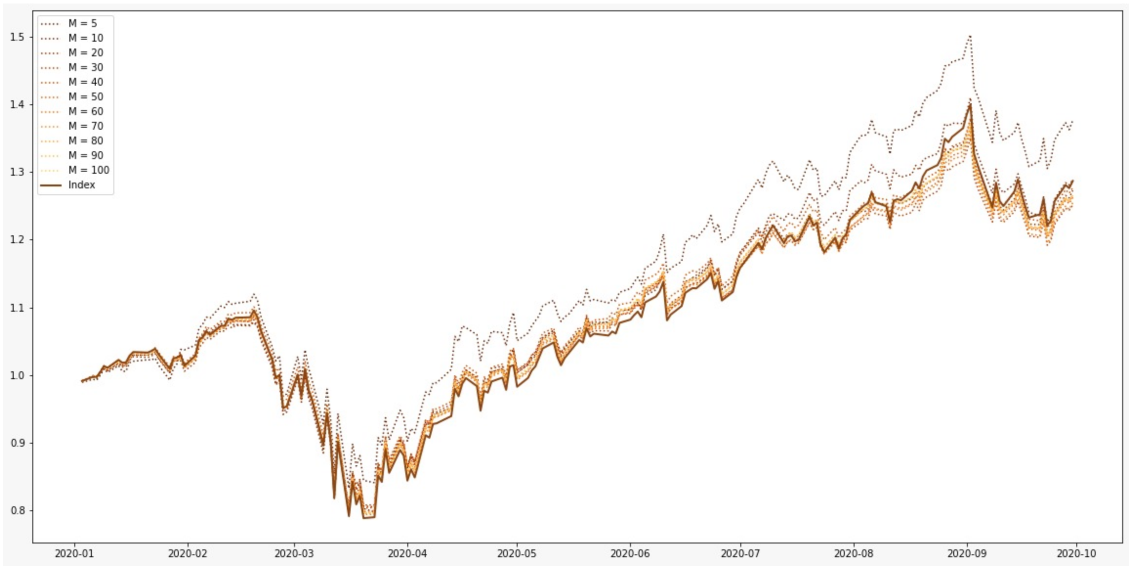
This resulted in an error as ~0.591.

Similarly, to evaluate the performance for each value of m, we calculated the sum of tracking error for in-sample data i.e. 2019 and out-of-sample data i.e. 2020 and the graph below represents the same:



The performance of the portfolio improves as we increase the value of m i.e. we include more and more in our index fund. But we can clearly observe that after m=60, the tracking error stays nearly constant for both in-sample data i.e. 2019 and out-sample i.e. 2020.

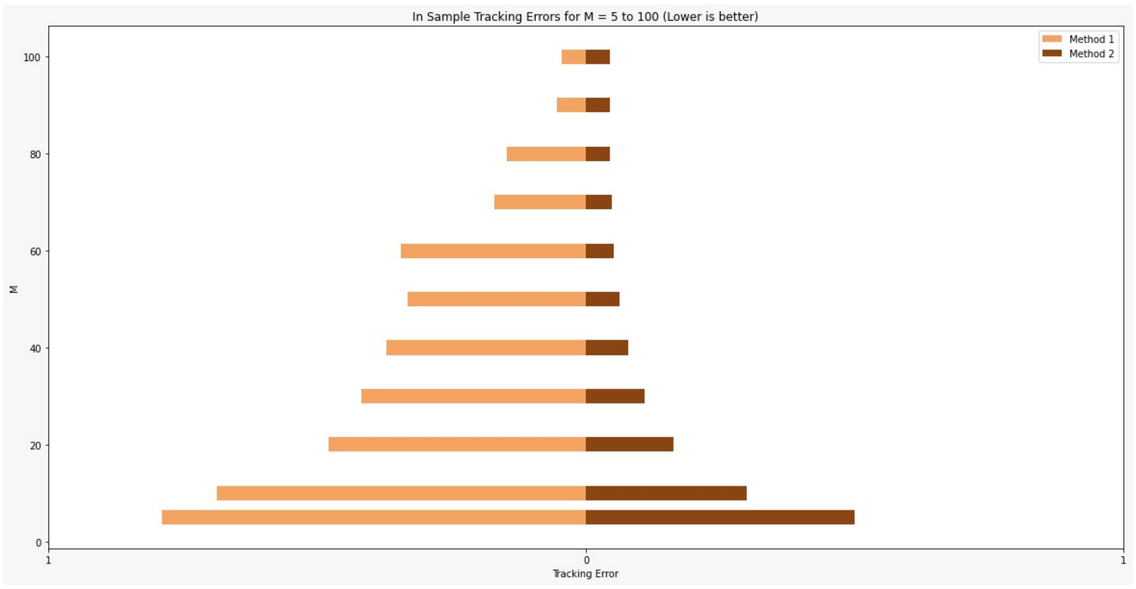
From the tracking error for both years, we observed that 2019's error rate is comparatively lower as compared to 2020's. This is as expected as the portfolios were built by using the weights from 2019 data as training data.



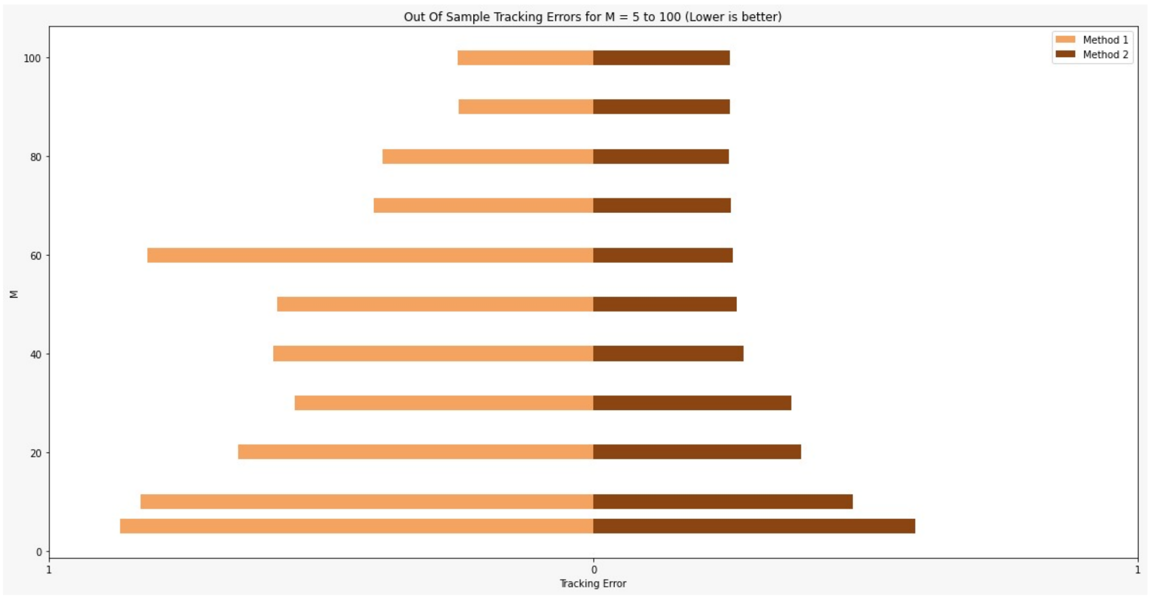
4. Conclusion and Recommendations

Using two methods, we built an index fund that replicates the NASDAQ 100. We observe a correlation of 99.71% between our recommended portfolio (m = 60; method 2) and the index, which we believe to be reasonably accurate and worth the cost and complexity saving achieved when compared to actively rebalancing to match the index.

Comparison of Method 1 vs. Method 2 for In-sample i.e. 2019 *(in terms of tracking error)*



Comparison of Method 1 vs. Method 2 for Out-of-sample i.e. 2020 *(in terms of tracking error)*



The second method is substantially improved for all ‘m’ values when compared to method one because our primary goal is to replicate the NASDAQ 100 while investing in the fewest amount of stocks possible.

Secondly, the first method yields the smallest absolute tracking error for m = 100, which is at odds with our goal of replicating NASDAQ 100 while owning the fewest stocks possible. On the other hand, the second method provides close to the smallest absolute tracking error at m = 60 and the error remains almost constant until m = 100. As a result, we can mimic NASDAQ 100 with just **60 stocks** (instead of 100 stocks identified in method 1), which lowers the cost of transactions drastically.

Lastly, as previously indicated, we also note that in the second method , where m=5, three of the top five holdings in the NASDAQ-100 are included in our index fund. However, in the first method, we only see one of the top five NASDAQ-100 holdings.This supports our claim that the second method more accurately replicates the NASDAQ 100.