FTEC 2101 Report

Background and Introduction:

This study centers on the application of diverse optimization techniques in addressing the Mean-Variance Portfolio Optimization problem. The array of methods examined include Unconstrained Optimization with Gradient Descent methods, Second-Order Cone Programming, Linear Programming, and Mixed Integer Programming. These techniques are instrumental in identifying the optimal asset allocation within a portfolio, subject to an array of constraints encompassing budgetary limitations, risk tolerance levels, and diversification mandates. By facilitating a systematic exploration of the risk-return trade-off, these optimization methodologies empower investors to delineate efficient portfolios, which align most effectively with their investment predilections and strategic objectives.

Model and Theory:

Task 1:
The problem we handled is a simplified mean-variance Portfolio Optimization problem:

$$\begin{gathered} minimize \quad \lambda p^T \Sigma p - p^T \overline{r} \\ s.t.\ 1^T p = B \end{gathered}$$

where Σ is the covariance matrix with dimension n*n, $\overline{r}$ is the expected return vector with dimension n*1. We can derivate the KKT conditions for this problem as

$$p_\lambda^* = \frac{1}{2\lambda}\Sigma^{-1}\left(\overline{r} + \frac{2\lambda B - r_1}{r_2}1\right), \text{where } r_1 = 1^T\Sigma^{-1}\overline{r}\,, r_2 = 1^T\Sigma^{-1}1.$$

For proving it, we first derivate the Lagrangian Function:

$$L(p,\mu) = \lambda p^T \Sigma p - p^{T\overline{r}} + (1^T p - B).$$

And the KKT conditions will be:

1. $\nabla_p(p^*, \mu^*) = 2\lambda\Sigma p^* - \overline{r} + \mu^* 1 = 0$
2. $\mu^*(1^T p^* - B) = 0$
3. $1^T p^* - B = 0$

From condition 3, we can get: $p^* = \frac{B}{1^T}$.

From condition 1, we get: $\mu^* 1^T = \overline{r} - 2\lambda\Sigma P^*$.

where $\mu^* = \frac{\overline{r} - 2\lambda\Sigma P^*}{1^T}$

$$= \frac{\overline{r} - 2\lambda \sum \frac{B}{1^T}}{1^T} * \frac{\Sigma^{-1}1}{\Sigma^{-1}1}$$

$$= \frac{\overline{r}\Sigma^{-1}1 - 2\lambda B}{1^T\Sigma^{-1}1} = \frac{r_1 - 2\lambda B}{r_2}$$

Then, substituting it back to condition 1 for finding the optimal portfolio:

$$\begin{aligned} p_\lambda^* &= \frac{\overline{r} - \mu^* 1^T}{2\lambda\Sigma} \\ &= \frac{1}{2\lambda}\Sigma^{-1}(\overline{r} + \frac{2\lambda B - r_1}{r_2}1) \end{aligned}$$

For task 1: b, we need to calculate the range of λ such that $\overline{r}p^* \geq R_d$.

$$\bar{r}^T p_\lambda^* \geqslant R_d$$

$$\Rightarrow R_d - \bar{r}^T p_\lambda^* \leq 0$$

$$\Rightarrow R_d - \bar{r}^T \cdot \frac{1}{2\lambda}\Sigma^{-1}(\bar{r} + \frac{2\lambda B - r_1}{r_2}1) \leq 0$$

$$\Rightarrow R_d - \left(\frac{\bar{r}^T\Sigma^{-1}}{2\lambda} \times \bar{r} + \frac{\bar{r}^T\Sigma^{-1}}{2\lambda} \times \left(\frac{2\lambda B}{r_2}1 - \frac{r_1}{r_2}1\right)\right) \leq 0$$

$$\Rightarrow R_d - \left(\frac{\bar{r}^T\Sigma^{-1}\bar{r}}{2\lambda} + \frac{\bar{r}^T\Sigma^{-1}B}{r_2}1 - \frac{\bar{r}^T\Sigma^{-1}}{2\lambda} \cdot \frac{r_1}{r_2}1\right) \leq 0$$

$$\Rightarrow R_d - \left(\frac{\bar{r}^T\Sigma^{-1}}{2\lambda} \cdot (\bar{r} - \frac{r_1}{r_2}1) + \frac{\bar{r}^T\Sigma^{-1}B}{r_2}1\right) \leq 0$$

$$\Rightarrow R_d - \frac{\bar{r}^T\Sigma^{-1}B}{r_2}1 \leq \frac{\bar{r}^T\Sigma^{-1}}{2\lambda}(\bar{r} - \frac{r_1}{r_2}1)$$

$$\Rightarrow \frac{R_d - \frac{\bar{r}^T\Sigma^{-1}B}{r_2}1}{(\bar{r} - \frac{r_1}{r_2}1)(\bar{r}^T\Sigma^{-1})} \leq \frac{1}{2\lambda}$$

So, we get $0 \geq \lambda \geq \frac{(\overline{r}\Sigma^{-1})\left(\overline{r} - \frac{r_1}{r_2}1\right)}{2\left(R_d - \frac{\overline{r}^T\Sigma^{-1}B}{r_2}1\right)}$.

Task2:

The previously discussed function primarily addresses the total risk associated with a portfolio. However, in practical terms, upside risk is not detrimental to investors, as it arises when the actual return surpasses the expected one. Therefore, in Task 2 and Task 3, our focus shifts towards minimizing downside risk, and we subsequently transform the problem into a Mixed Integer Programming problem. This approach allows for a more nuanced consideration of risk dynamics and facilitates better alignment with investor preferences.

The problem is:

$$\sum_{t=1}^{T} \mathbb{1}\{\boldsymbol{p}^\top(\bar{\boldsymbol{r}} - \boldsymbol{R}_t) > 0\}$$

$\sum_{i=1}^n p_i \bar{r}_i \geq R_{\mathsf{d}}.$

$\sum_{i=1}^n p_i \leq B.$

$p_i \geq 0,\ i = 1, ..., n.$

The main task here is the conversion the 1{} function.

Let $z_i = \mathbb{1}\{p^T(\bar{r} - R_t) > 0\}$, $z_i \in \{0, 1\}$ where $i = 1, \cdots, T$, $t = 1, \cdots, T$

If $p^T(\bar{r} - R_t) > 0$, $z_i = 1$ else $z_i = 0$.

if $z = 0$, then $p^T(\bar{r} - R_t) \leq 0$ else unbounded

$p^T(\bar{r} - R_t) - Mz_i \leq 0$

if $z = 1$, then $p^i(\bar{r}_i - R_i) > 0$ else unbounded

$p^T(\bar{r} - R_t) > -(1 - z_i)M$

Rewrite the problem:

$$\min \sum_{i=1}^{T} z_i$$

s.t.

$p^T(\bar{r} - R_j) - Mz_i \leq 0$, $i = 1, \cdots, T$

$p^T(\bar{r} - R_i) > -(1 - z_i)M$, $i = 1, \cdots, T$

$\sum_{i=1}^n p_i \bar{r}_i \geqslant R_d$

$\sum_{i=1}^n p_i \leq B$

$p_i \geqslant 0$, $i = 1, \cdots, n$

$z_i \in \{0, 1\}$, $i = 1, \cdots, n$

Task3:

Since the MIP problem is very computational when T is large. So, we introduce an approximated objective function with the following ridge function and then convert it into standard form linear program problem:

$$\min \frac{1}{T}\sum_{t=1}^{T} \max\{p^T\bar{r} - p^T R_t , 0\}$$

$$\begin{aligned} \text{s.t.}\quad & \sum_{i=1}^{n} p_i \bar{r}_i \geqslant R_d \\ & \left|\sum_{i=1}^{n} p_i\right| \leq B \\ & p_i \geqslant 0 , \ i = 1, \cdots, n \end{aligned}$$

We need to introduce new variables, z_i, where $z_i = \max\,(p^T\bar{r} - p^T R_i)$, i = 1,…,T, and handle the absolute value sign.

After transformation, it becomes:

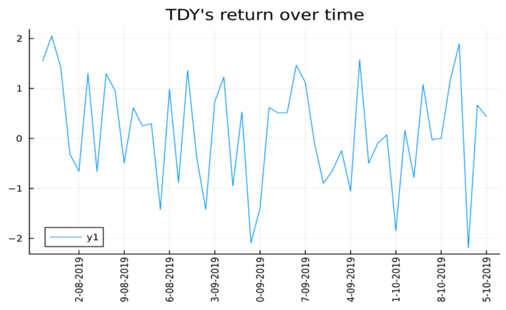
$$\max -\frac{1}{T}\sum_{i=1}^{T} z_i$$

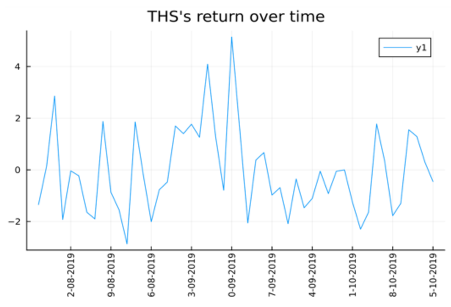
$$\begin{aligned} \text{s.t.}\quad & z_t \geqslant p^T\bar{r} - p^T R_t , \ t = 1, \cdots, T \\ & \sum_{i=1}^{n} p_i \bar{r}_i \geqslant R_d \\ & \sum_{i=1}^{n} p_i \leq B \\ & -\sum_{i=1}^{n} p_i \leq B \\ & p_i \geqslant 0 , \ i = 1, \cdots, n \\ & z_t \geqslant 0 , \ t = 1, \cdots, T \end{aligned}$$

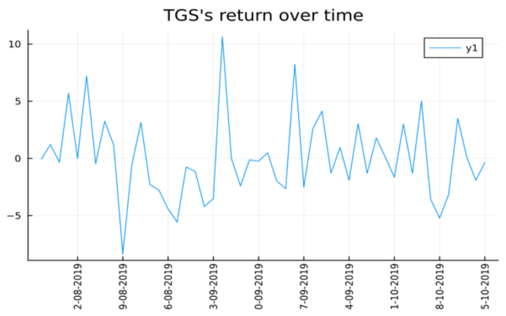
Experiments:

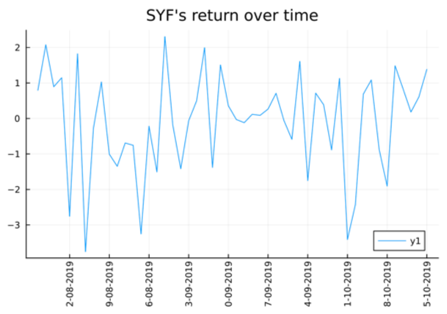
Task4:

Upon visualizing the daily returns of the meta dataset, we can glean significant insights about it. The dataset comprises fifty days of time-series data related to stock returns, spanning from February 8, 2019, to May 10, 2019. The columns represent different stocks (n), while the rows correspond to distinct dates (T). This comprehensive dataset offers a valuable foundation for further analysis and portfolio optimization.







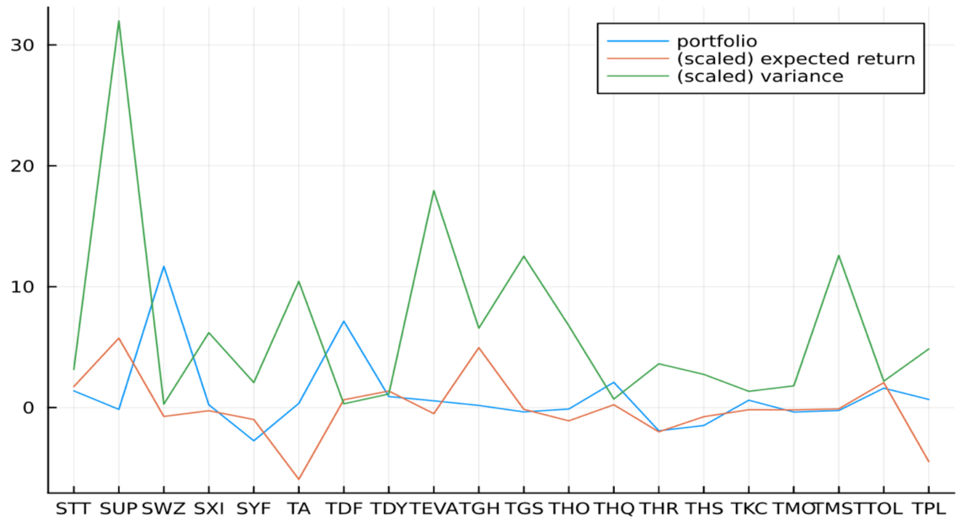


Task5:

After implementing the closed form solution found in Task 1 with B = 20, Rd = max {2, sum (bar - R)}, lambda = 0.5. We obtain a 20 elements vector that stores the optimal portfolio weight.

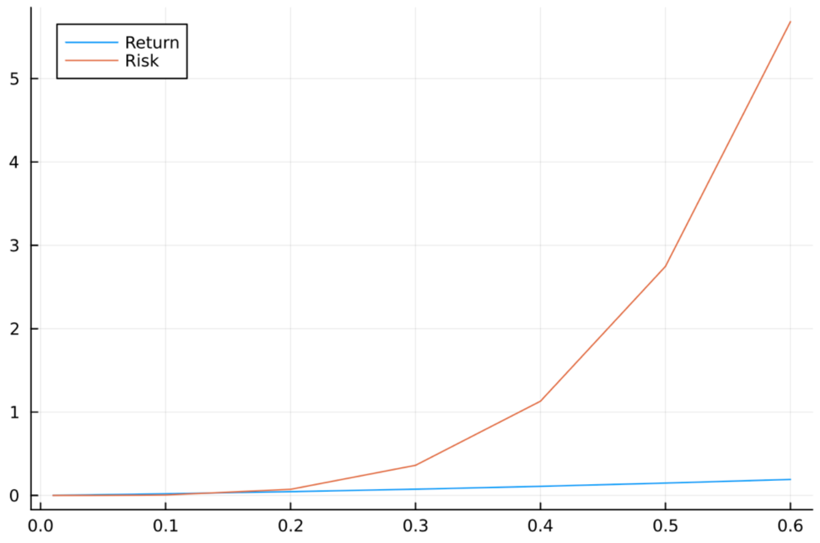
```
  1.3799960465968677
 -0.1484580624902313
 11.688294402479036
  0.22724443223662696
 -2.7459922362294
  0.3450656388818665
  7.144594062928471
  0.9110483613216331
  0.5583059533431871
  0.17701420709112284
 -0.3622506078287646
 -0.12000017029958071
  2.0876789060038896
 -1.922241709499803
 -1.4846890356658327
  0.6078160874284902
 -0.37319371300928267
 -0.24508111148636907
  1.6039643217893567
  0.6708842264087115
```
.

We can observe this portfolio, the weight can be positive or negative (buy or sell the stock). And the sum of portfolio_opt is 20.



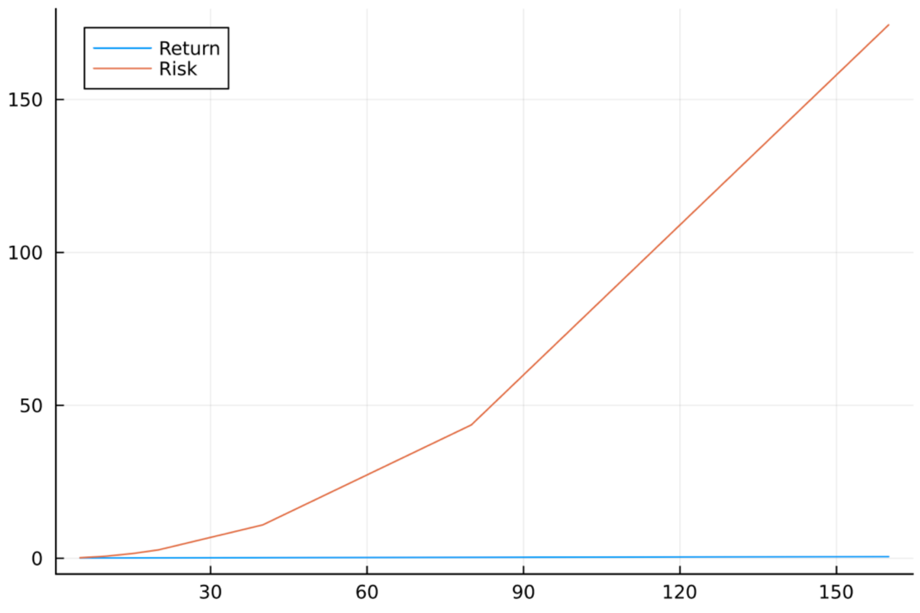
We are also able to plot the portfolio on a graph with scaled expected return and variance for the entire stock set. This strategy predominantly favors purchasing SUP and TGH while substantially selling TA and TPL.
Prior to adjusting the parameters, we computed the extreme value of lambda from Task 1b, where l_exme = 0.1019979556. This value will be instrumental in further refining our portfolio optimization approach.

Then we try different value of lambda and observe to the effects on the optimal portfolio.



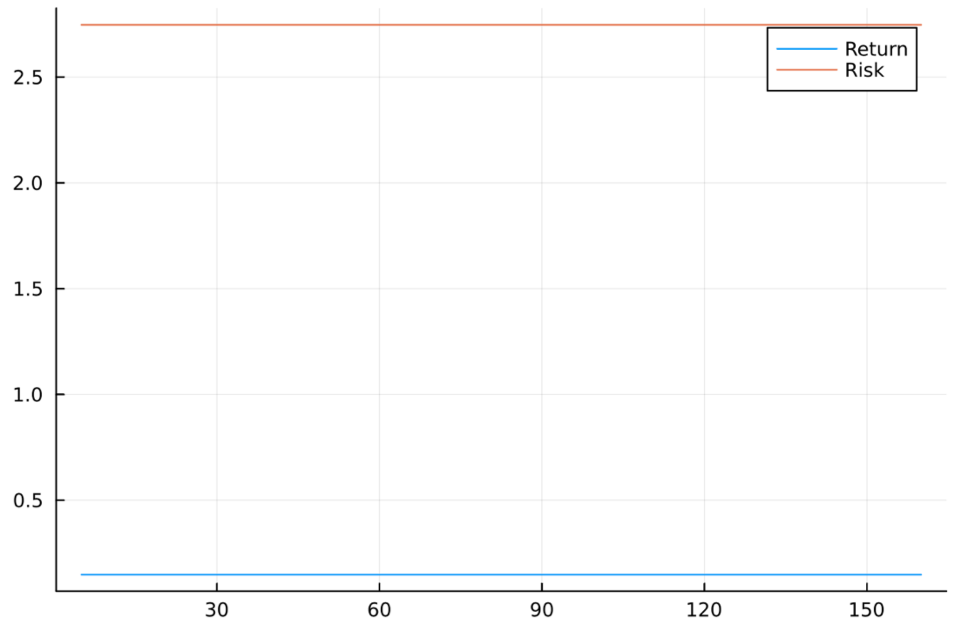
I chose the range of lambda from 0.01 to 0.6. It is evident that both return and risk are positively proportional to lambda; as lambda increases, the risk increases exponentially, while the return experiences a comparatively modest increase.

Next, we examine the relationship between the portfolio and budget, B.



Next, we examine the relationship between the portfolio and budget, B. I chose the range of B from 5 to 160. We can observe that the Risk curve moves upward and the Return curve forms a straight line. Thus, the budget is positively proportional to the risk and irrelevant to the return.

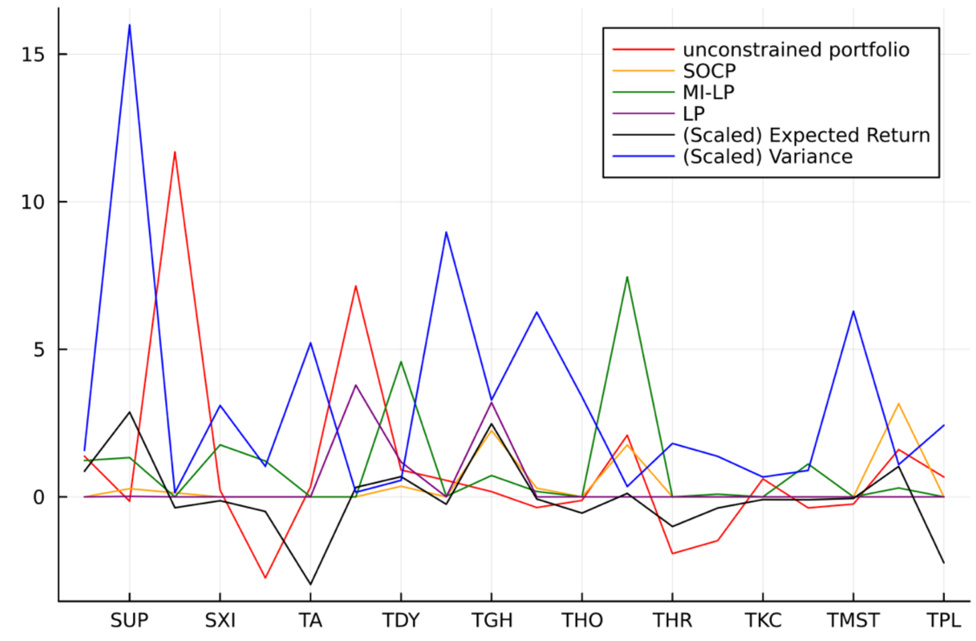
Finally, we plot the portfolio with different values of Rd.



It is clear that the required return on the portfolio is uncorrelated, as the change in Rd has no effect on both return and risk (two straight lines).

Task 6:

We need to implement SOCP program and mixed-integer program and linear program respectively. For SOCP, the key point is to reformulates the objective function $\frac{1}{2}p^T\Sigma p$ to a second-order cone, $\frac{1}{2}p^T A^T Ap$ where $A^T A = \Sigma$., then $t \geq sqrt(p^T \Sigma p) = norm(Ap)$. It can be decomposed using various methods. In this task, I used Cholesky decomposition. After implementing all three types of programs, we can compare the portfolios found by plotting them and calculating their performance on the test set.



According to the figure, all portfolios except for the unconstrained portfolio do not contain negative weights since we added non-negative constraints to them. Moreover, MIP and SOCP create more complex portfolios than the linear program.

```
Perfor_Opt = performance_testdata( path_subgroup, portfolio_opt[:,1]
```

```
Sharpe Ratio = 0.02343358099992304, Prob. of Downside Risk Violation =
0.4849246231155779, Amt. of Downside Risk Violation = 0.070167252527602
7, Return = 0.004655109825124278, Portfo Value = 19.999999999999996
```

```
Perfor_SOCP = performance_testdata( path_subgroup, (JuMP.value.(x_so
```

```
Sharpe Ratio = 0.051189809978188436, Prob. of Downside Risk Violation =
0.5414572864321608, Amt. of Downside Risk Violation = 0.060790655352844
364, Return = 0.007643299331346252, Portfo Value = 8.233157652168526
```

```
Perfor_IP = performance_testdata( path_subgroup, JuMP.value.(x_milp)
```

```
Sharpe Ratio = 0.022988276721965968, Prob. of Downside Risk Violation =
0.5201005025125628, Amt. of Downside Risk Violation = 0.096999192103489
08, Return = 0.0058911741857432705, Portfo Value = 20.0
```

```
Perfor_LP = performance_testdata( path_subgroup, JuMP.value.(x_lp) ,
```

```
Sharpe Ratio = 0.07241984008144643, Prob. of Downside Risk Violation =
0.5703517587939698, Amt. of Downside Risk Violation = 0.052668829169622
84, Return = 0.009120936663323025, Portfo Value = 8.183052008898677
```

Surprisingly, the linear program's performance on the test set achieves the highest Sharpe ratio of around 0.0724 and the lowest amount of downside risk violation at approximately 0.053. In contrast, the integer program performs the worst on this specific data set, only obtaining a Sharpe ratio of around 0.023.

Competitive Task:

In this task, our objective is to design a customized optimizer for solving large-scale portfolio problems. For gradient descent algorithms, it is preferable to have a function that can be differentiated everywhere. Since the max function cannot be differentiated at point 0, we use the Huber function as an approximation. The Huber function provides a smooth transition between linear and quadratic regions, making it suitable for optimization algorithms that rely on gradient information. By employing the Huber function, we can effectively handle large-scale portfolio optimization problems while maintaining the differentiability of the objective function.

$$h_\delta(x) = \begin{cases} \frac{x^2}{2\delta}, & \text{if } 0 \le x \le \delta, \\ x - \frac{1}{2}\delta, & \text{if } x > \delta, \\ 0, & \text{if } x < 0, \end{cases}$$

And we modify the optimization problem using penalty method, adding constraints to the objective function by imposing a cost to infeasible points. Then the objective function becomes:

$$\begin{aligned}\min_{\boldsymbol{p}\in\mathbb{R}^n} \quad & \sum_{t=1}^{T} h_\delta(\boldsymbol{p}^\top\bar{\boldsymbol{r}} - \boldsymbol{p}^\top\boldsymbol{R}_t) + \gamma f_1(\boldsymbol{p}) + \upsilon f_2(\boldsymbol{p}) \\ \text{s.t.} \quad & 0 \le p_i \le 1,\ i = 1, \ldots, n,\end{aligned}$$

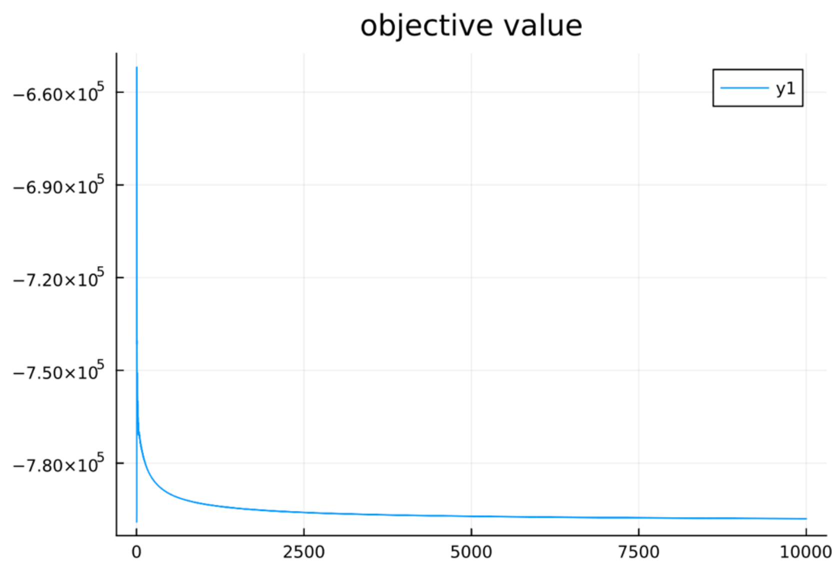
The derivative function of $f_1()$ and $f_2()$ are constant if they are the same constraint for Task 2,3. So it should not affect the convergence of the objective function whatever γ, ν are. And the derivative of Huber function will simply be programed by:

```
function derivative_huber_fct(x, delta)
    if 0 <= x <= delta
        return x / delta
    elseif x > delta
        return 1
    else
        return 0
    end
end
```

In this task, we utilize the information obtained to compute the gradient of the objective function at any given portfolio points.

To implement this function, we need to break it down into separate parts. Generally, there are two approximation methods for the max function: the Huber function and the sigmoid function. There are also two gradient descent methods: the projected gradient descent and the conditional gradient method. Additionally, we test standardization and normalization for the gradient. After implementing these functions, we can assess the performance of the algorithm.

Projected-gradient descending method using Huber loss:
In this case, we employ the Huber loss function to approximate the max function and project the updated portfolio to a reasonable region from -10 to 10 with 10,000 iterations. We find that standardization significantly improves convergence by applying a standard function before updating. The algorithm converges rapidly, reaching convergence at around 1,000 iterations. The performance of this algorithm is generally good, although the probability of downside risk violation is higher than expected.



```
downsiderisk( x_gd, T, stocks_retur_full,bar_R_full )
```

```
Prob. of Downside Risk Violation = 0.6708385481852316, Amt. of Downside
Risk Violation = 7.563042108442833e-5

1×2 Matrix{Float64}:
 0.670839  7.56304e-5
```

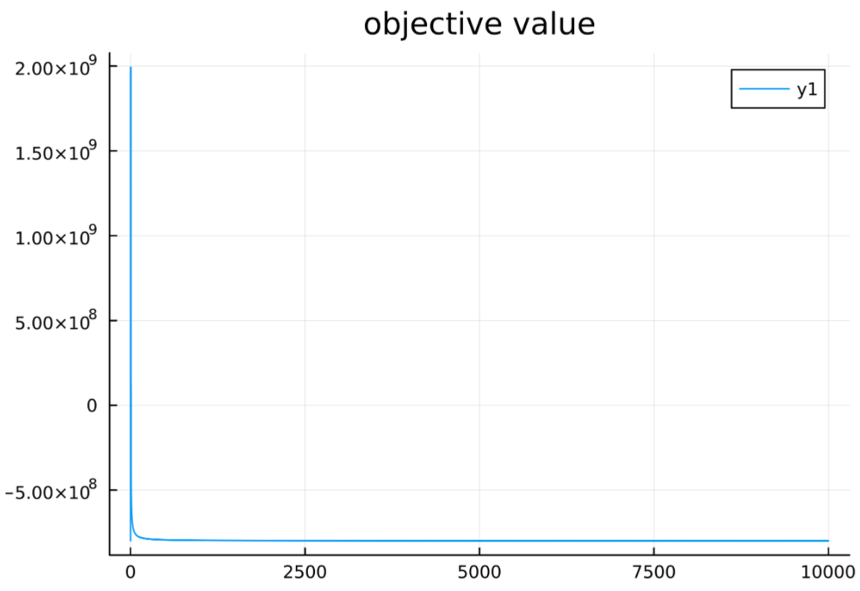
```
perfor_GD = performance_testdata( "./ftec_project_files/", x_gd , 44
```

```
Sharpe Ratio = -0.08504524198726834, Prob. of Downside Risk Violation =
0.5615212527964206, Amt. of Downside Risk Violation = 0.000259839425036
2843, Return = -4.9077509932355576e-5, Portfo Value = 0.001250952923226
6258
```

Conditional gradient with Huber loss function:
Similar to the previous algorithm, we replace the projection method with the conditional gradient method. The update in each iteration changes accordingly to:

$$\boldsymbol{a}^{(k)} = \arg\min_{\boldsymbol{a}\in X} \ \boldsymbol{a}^\top \nabla F(\boldsymbol{x}^{(k)})$$
$$\boldsymbol{x}^{(k+1)} = (1 - \tfrac{2}{k+1})\boldsymbol{x}^{(k)} + \tfrac{2}{k+1}\boldsymbol{a}^{(k)}$$

After 10000 iterations, we get:



We can see that conditional decent converges even faster than projection decent method. It converges immediately.

```
downsiderisk( x_gd, T, stocks_retur_full,bar_R_full )
```

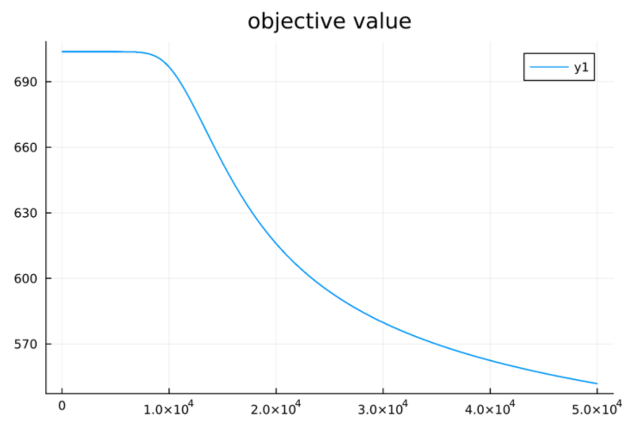
```
Prob. of Downside Risk Violation = 0.4793491864831039, Amt. of Downside
Risk Violation = 0.0011303779232325936

1×2 Matrix{Float64}:
 0.479349  0.00113038
```

```
perfor_GD = performance_testdata( "./ftec_project_files/", x_gd , 44
```

```
Sharpe Ratio = -0.0006024585054972592, Prob. of Downside Risk Violation
= 0.4742729306487696, Amt. of Downside Risk Violation = 0.0011643920798
242822, Return = -1.8592059986810802e-6, Portfo Value = 0.3494650534946
495
```

we observe that the conditional descent converges even faster than the projected gradient descent method. It converges immediately and obtains a lower probability of downside risk violation at about 0.47. However, the Sharpe ratio is not as good as the previous one.

Projected-gradient descending method using Sigmoid loss:
Instead of using the Huber function, we replace it with the Sigmoid function combined with



the projection method.

We observe that this algorithm converges extremely slowly, not fully converging even after

```
downsiderisk( x_gd, T, stocks_retur_full,bar_R_full )
```

```
Prob. of Downside Risk Violation = 0.22778473091364204, Amt. of Downsid
e Risk Violation = 0.08556660421925164

1×2 Matrix{Float64}:
 0.227785  0.0855666
```

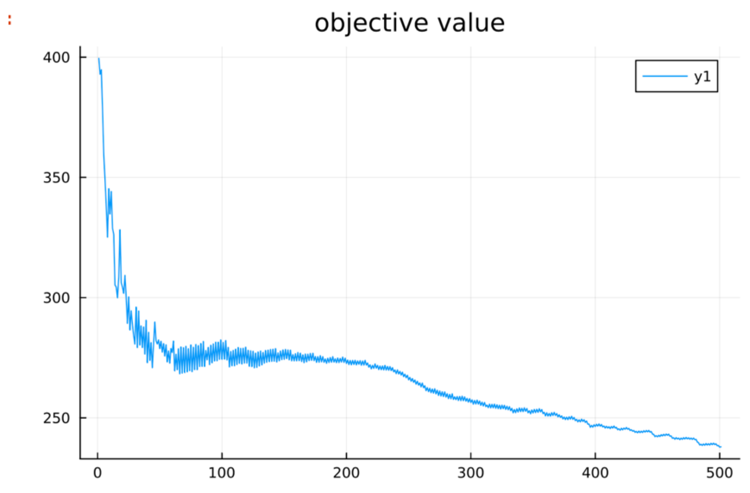
```
formance_testdata( "./ftec_project_files/", x_gd , 447, bar_R_full );
```

```
Sharpe Ratio = 0.001886079360915339, Prob. of Downside Risk Violation =
0.46308724832214765, Amt. of Downside Risk Violation = 0.08258721102507
735, Return = 0.0004473201916991042, Portfo Value = 17.801754245754243
```

50,000 iterations.

However, the performance is still slightly better than the Huber-based algorithms.

(Selected as final solution) Conditional descent method using Sigmoid loss:



```
Prob. of Downside Risk Violation = 0.2428035043804756, Amt. of Downside
Risk Violation = 0.06307570563708542
```

```
1×2 Matrix{Float64}:
 0.242804  0.0630757
```

```
perfor_GD = performance_testdata( "./ftec_project_files/", x_gd , 447
```

```
Sharpe Ratio = 0.013811533529527296, Prob. of Downside Risk Violation =
0.4899328859060403, Amt. of Downside Risk Violation = 0.062936808206012
34, Return = 0.002496478415952306, Portfo Value = 13.718339321357284
```

As the graph shows, the sigmoid function as an approximation of the max function, combined with the conditional descent approach, obtains the best result among all other methods. Consequently, this algorithm is selected as the final solution for Task 8.

Conclusion:

Findings:
In this project, we explored the mean-variance Portfolio Optimization problem using various optimization methods, including Unconstrained optimization with descending methods, Second-Order Cone Programming, Linear programming, and mixed integer programming. Our goal was to determine the optimal weights for each asset in the portfolio, subject to constraints like budget, risk tolerance, and diversification requirements. We analyzed the trade-offs between risk and return and assessed the performance of each optimization method on a given dataset.

Discussion and Improvements:
While our findings provide valuable insights into portfolio optimization, there is room for further improvement in various aspects of the formulation and implementation:

1. Alternative approximation methods:
Exploring other approximation methods could potentially improve the performance and convergence of these algorithms. For example, we could utilize the softmax function as an alternative to the max function. The softmax function assigns a probability to each element in a set while emphasizing the largest values, making it a suitable candidate for our optimization problem.

2. Hyperparameter tuning:

Instead of manual tuning hyperparameter using experience, we can search the best hyperparameter wisely and wider. For instance, we could use cross-validation or other advanced techniques to find the optimal values for lambda and budget that maximize our portfolio's Sharpe ratio or minimize downside risk.

3. Additional constraints:

We can incorporate other real-world constraints, such as transaction costs, taxes, or sector exposure limits, to make the optimization problem more representative of actual investment scenarios. For example, we could include a constraint that limits the total transaction costs associated with rebalancing the portfolio or a constraint that ensures the portfolio's exposure to a particular sector does not exceed a certain threshold.

4. Ensemble methods:

Combining the strengths of different optimization methods through ensemble techniques could lead to improved performance and more robust portfolio optimization solutions. For example, we could create an ensemble of portfolios generated using different optimization methods and assign weights to each portfolio based on their individual performance. This approach could potentially capture the best aspects of each method, leading to better overall performance and risk management.