FTEC2101/ESTR2520 Optimization Methods **Spring 2021**

Project Specification – Portfolio Optimization in Practice

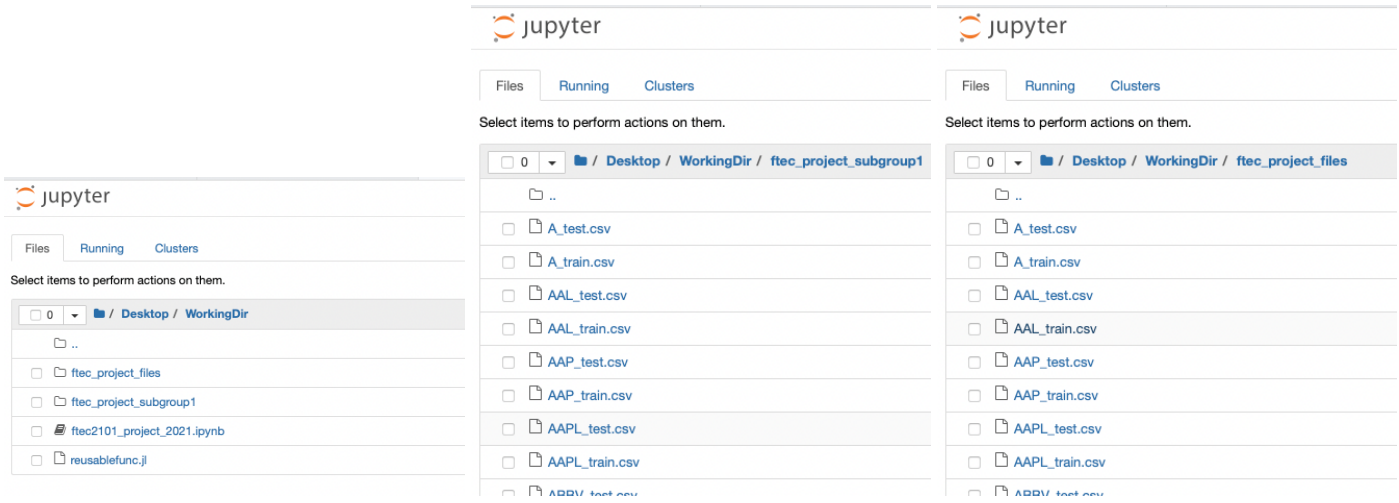
Last Updated: April 10, 2021, Deadline: May 14, 2021, 23:59 (HKT)

Thus far we have learnt a number of optimization methods, ranging from the simplex method for linear programming, modeling techniques for integer programming, gradient/Newton methods for unconstrained optimization, KKT conditions, SOCP formulation, etc.. Besides the theories, it is important to remember that optimization methods are practical tools for solving real world problems. This is the aim of the current project to highlight on the latter aspect.

This project implements practical solutions for portfolio optimization using `Julia`[1]. We will explore various aspects of portfolio design and more importantly, *implement* these ideas on a real-world dataset.

Dataset & Folder Setting In this project, we will use a dataset gathered from the S&P 500 market with the stock prices of 471 stocks between 2013 and 2018. To prepare for the project, please retrieve the Jupyter notebook template `ftec2101_project_2021.ipynb`, a helper function `reusablefunc.jl`, and the archives `ftec_project_files.zip`, `ftec_project_subgroupi.zip` (where $i \in \{1, ..., 5\}$ – depending on your assigned subgroup).

Move the files to a working directory of your choice, and unzip the archives. Your working directory shall have the following content:



The folders `ftec_project_subgroupi`, `ftec_project_files` contain essentially the same content of the stock data of 20 stocks and 471 stocks, respectively, with the latter containing the complete dataset that includes the former. Furthermore, for each stock, the dataset is split a training dataset and testing dataset, as follows:

- `*_train.csv` – stock prices for *training*, taken from Feb, 2013 to Apr, 2016.
- `*_test.csv` – stock prices for *testing*, taken from Apr, 2016 to Feb, 2018.

[1] As an alternative, you are welcomed to use `Python` with optimization modeling packages supporting SOCP and MI-NLP such as `cvxpy`. However, you have to notify the instructor about the choice and the plan on how you wish to accomplish the project in the latter case on or before May 1, 2021, i.e., two weeks before the deadline.

1.1 Compulsory Tasks (50%)

Suppose that there are n stocks in the market that can be invested. We begin our study by considering the (simplified) Markowitz's mean-variance Portfolio Optimization problem:

$$\begin{aligned} \min_{\boldsymbol{p}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{p}^\top \boldsymbol{\Sigma} \boldsymbol{p} \\ \text{s.t.} \quad & \mathbf{1}^\top \boldsymbol{p} = B,\ \bar{\boldsymbol{r}}^\top \boldsymbol{p} = R_{\mathsf{d}}, \end{aligned} \tag{1.1}$$

where B is the fixed budget, R_{d} is the *fixed* desired return, and $\mathbf{1}$ is an all-one vector. We have defined the covariance matrix and expected reward vector respectively as:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & \rho_{22} & & \rho_{2n} \\ \vdots & & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \cdots & \rho_{nn} \end{pmatrix}, \quad \bar{\boldsymbol{r}} = \begin{pmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \vdots \\ \bar{r}_n \end{pmatrix} \tag{1.2}$$

such that for any $i, j = 1, ..., n$, it holds

$$\bar{r}_i = \mathbb{E}_t[R_i(t)], \quad \rho_{ij} = \mathbb{E}_t[(R_i(t) - \bar{r}_i)(R_j(t) - \bar{r}_j)], \quad R_i(t) = \frac{\text{closing price on day } t - \text{opening price on day } t}{\text{opening price on day } t}, \tag{1.3}$$

i.e., $R_i(t)$ is the return of stock i on day t. Notice that (1.1) is a constrained optimization problem, which can be easily shown to be a convex optimization since $\boldsymbol{\Sigma}$ must be a PSD matrix.

Theory & Modeling

Task 1: (5%)

Consider the portfolio optimization given in (1.1). Answer the following questions:

(a) Derive the KKT conditions for (1.1) and show that the optimal solution for (1.1) is:

$$\boldsymbol{p}^\star = \frac{\boldsymbol{\Sigma}^{-1}\Big\{(\mathrm{r}_0 B - \mathrm{r}_1 R_{\mathsf{d}})\mathbf{1} + (\mathrm{r}_2 R_{\mathsf{d}} - \mathrm{r}_1 B)\bar{\boldsymbol{r}}\Big\}}{\mathrm{r}_0\mathrm{r}_2 - \mathrm{r}_1^2},$$

where

$$\mathrm{r}_0 = \bar{\boldsymbol{r}}^\top \boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{r}}, \quad \mathrm{r}_1 = \mathbf{1}^\top \boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{r}}, \quad \mathrm{r}_2 = \mathbf{1}^\top \boldsymbol{\Sigma}^{-1}\mathbf{1}.$$

(b) Consider the special case that $n = 2$ (with only 2 stocks) and the covariance/expected return be given by

$$\bar{\boldsymbol{r}} = \begin{pmatrix} \bar{r}_1 \\ 1 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Denote the solution found in part (b) as $\boldsymbol{p}^\star = (p_1^\star, p_2^\star)$. Comment on how the optimal portfolio will change if (i) $\bar{r}_1$ increases, and/or (ii) R_{d} increases. *Remark:* You may assume $R_{\mathsf{d}} > B$ and $\bar{r}_1 \neq 1$.

The answer to part (b) should confirm some intuition that you might have for the portfolio design. In the following, we shall extend the framework to take care of some realistic aspects of portfolio optimization.

Suppose that you are given a portfolio $\boldsymbol{w} \in \mathbb{R}^n$ such that w_i is the amount of stock i that you are holding. Our objective is to determine $\boldsymbol{x} \in \mathbb{R}^n$ such that if $x_i > 0$, then we invest more on stock i (i.e., longing); and vice versa for $x_i < 0$ (i.e., shorting). In other words, the vector

$$\boldsymbol{p} = \boldsymbol{w} + \boldsymbol{x}$$

represents the new portfolio where the *risk* and *expected return* should be calculated from.

In practice, a transaction cost (e.g., stamp duty, administration fees, etc.) is imposed every time a stock is traded. For a given $\boldsymbol{x}$, the total transaction cost can be modeled as:

$$\sum_{i=1}^{n} \tilde{\phi}(x_i) \quad \text{where} \quad \tilde{\phi}(x_i) = \begin{cases} c, & \text{if } x_i \neq 0, \\ 0, & \text{if } x_i = 0, \end{cases} \tag{1.4}$$

for some $c > 0$. In this model, a flat transaction cost of $\$c$ is incurred if we trade any share of stock i.

Task 2: (5%)

Formulate the portfolio optimization problem as a mixed-integer program (in fact, a mixed-integer non-linear program, MI-NLP) to determine $\boldsymbol{x} \in \mathbb{R}^n$ with the following requirements:

- The objective is to minimize the risk (variance) associated with the new portfolio, i.e.,

  $$(\boldsymbol{x} + \boldsymbol{w})^\top \boldsymbol{\Sigma} (\boldsymbol{x} + \boldsymbol{w})$$

- It is required that the expected return is above or equal a given threshold $\$R_{\mathtt{d}}$, i.e.,

  $$\textstyle\sum_{i=1}^{n} (x_i + w_i)\bar{r}_i \geq R_{\mathtt{d}}.$$

- The sum of *net change in portfolio value* and *transaction cost*, given by $\tilde{\phi}(\cdot)$ in the above, must be less than or equal to the budget $\$B$, i.e.,

  $$\textstyle\sum_{i=1}^{n} \{x_i + \tilde{\phi}(x_i)\} \leq B,$$

  where $\tilde{\phi}(\cdot)$ is defined in (1.4).

- The changes to the invested amount on *each stock* must be less than or equal $\$M$, i.e.,

  $$|x_i| \leq M, \ i = 1, ..., n.$$

- The new portfolio must be non-negative, i.e.,

  $$x_i + w_i \geq 0, \ i = 1, ..., n.$$

For ease of implementation in the latter task, you may ensure that only linear constraints are found in the formulated MI-NLP, while the objective function can be nonlinear (actually quadratic).

Another transaction cost model is: for some $a > 0$,

$$\sum_{i=1}^{n} \phi(x_i) \quad \text{where} \quad \phi(x_i) = a x_i^2 \tag{1.5}$$

which can be viewed as an approximation of the flat transaction cost model $\tilde{\phi}(\cdot)$ in (1.4).

Task 3: (5%)

(a) Similar to Task 2, formulate a nonlinear program to determine $\boldsymbol{x} \in \mathbb{R}^n$ with the following requirements:

- The objective is to minimize the risk (variance) associated with the new portfolio.
- It is required that the expected return is above or equal a given threshold as $\$R_{\mathtt{d}}$.
- The sum of *net change in portfolio value* and *transaction cost*, given by $\phi(\cdot)$ in the above, must be less than or equal to the budget $\$B$, i.e.,

$$\textstyle\sum_{i=1}^{n}\{x_i + \phi(x_i)\} \leq B,$$

  where $\phi(\cdot)$ is defined in (1.5).
- The changes to the invested amount on *each stock* must be less than or equal $\$M$.
- The new portfolio must be non-negative.

(b) Rewrite the above optimization problem as an SOCP. You may adopt the factorization of the positive definite matrix as $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}^{1/2})^\top \boldsymbol{\Sigma}^{1/2}$.

Computational The optimization problems formulated in Task 2 & 3 **do not** admit a closed form solution. We next focus on solving these portfolio optimization problems numerically on computers.

Data Preprocessing. The required data – expected return $\bar{r}_i$, and covariance between stocks $\boldsymbol{\Sigma} = [\rho_{ij}]_{n\times n}$ – can be *approximated* from historical data as:

$$\bar{r}_i \approx \frac{1}{T}\sum_{t=1}^{T} R_i(t), \quad \rho_{ij} \approx \frac{1}{T}\sum_{t=1}^{T}\Big(R_i(t) - \frac{1}{T}\sum_{t'=1}^{T} R_i(t')\Big)\Big(R_j(t) - \frac{1}{T}\sum_{t'=1}^{T} R_j(t')\Big), \quad \forall\ i,j = 1,...,n. \tag{1.6}$$

With a slight abuse of notations, we shall refer the approximated expected return and covariance by the same symbols $\bar{r}_i = \frac{1}{T}\sum_{t=1}^{T} R_i(t)$ and $\boldsymbol{\Sigma} = [\rho_{ij}]_{n\times n}$.

In the following tasks, we focus on a small set of $n = 20$ stocks (from `ftec_project_subgroup` i):

Task 4: Warmup Exercise (5%)

(a) Inspect the dataset, e.g., by plotting the daily return over time of around 3-4 stocks of your choice.

Remark: The program template has provided the relevant helper codes for this task.

The program template has also calculated the approximated expected return and covariance for you.

Portfolio Optimization. In the following tasks, you will implement the different formulations of portfolio optimization that you have completed in Task 1–3. We will implement these optimization problems into Julia and evaluate the performance of your portfolio.

Task 5: Closed Form Solution with Simple Portfolio Optimization (10%)

(a) Implement the closed form solution found in Task 1 with the given training data. You may take the parameters

$$B = 20,\ R_{\mathsf{d}} = 1.01 \sum_{i=1}^{n} \bar{r}_i.$$

Comment on the portfolio found using this approach.

(b) Plot the portfolio across different stocks and compare the portfolio profile with (i) the expected returns $\bar{r}_i$, (ii) the variance of each stock, i.e., $\hat{\rho}_{i,i}$. Comment on the trends found. *Hint: You may need to scale up/down the expected returns and variances to make the plots more readable.*

(c) Try different values for the desired return R_{d}, budget B. Comment on their effects on the optimal portfolio.

Suppose the given portfolio is $w_i = 1$, $i = 1, ..., n$. We now implement the two formulations in Tasks 2–3.

Task 6: Optimization-based Formulation (15%)

(a) Implement the mixed-integer nonlinear program from Task 2 with the following parameters:

$$M = 20,\ B = 20,\ c = 2,\ R_{\mathsf{d}} = 1.01 \sum_{i=1}^{n} \bar{r}_i,\ \boldsymbol{w} = \mathbf{1},$$

where $\mathbf{1}$ indicates an all-one vector. You may use the solver `Juniper` (with `Ipopt`) in `JuMP` for tackling the MI-NLP.

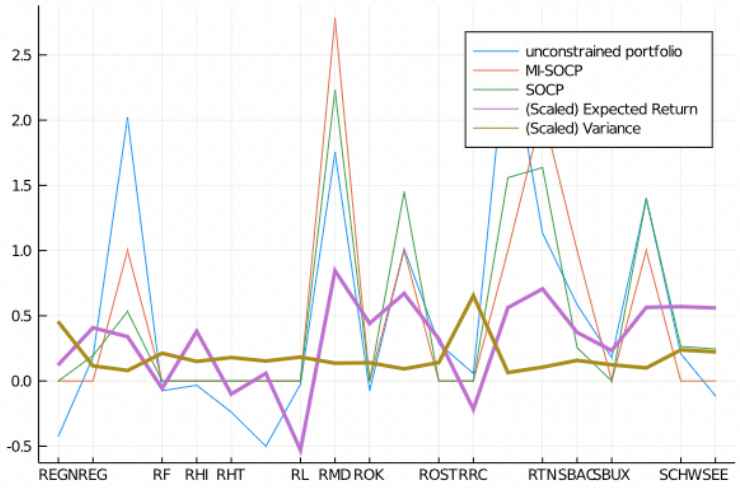
(b) Implement the SOCP from Task 3 with the following parameters:

$$M = 20,\ B = 20,\ a = 2,\ R_{\mathsf{d}} = 1.01 \sum_{i=1}^{n} \bar{r}_i,\ \boldsymbol{w} = \mathbf{1}.$$

You may use the solver `ECOS` in `JuMP` for tackling the SOCP.

(c) Plot and compare the portfolios found using the approaches from Task 1, 2, and 3. Comment on the differences between the solutions obtained.

Notice that the final plot in part c) of the above may look like:



which shows the portfolio optimized for each stock, compared with the expected return and variance/risk.

The performance of a portfolio can be measured by the following *Sharpe Ratio*[2] [Fabozzi et al., 2007]:

$$\text{Sharpe ratio} = \frac{\bar{\boldsymbol{r}}_{\text{test}}^{\top}\boldsymbol{p}}{\sqrt{\boldsymbol{p}^{\top}\boldsymbol{\Sigma}_{\text{test}}\boldsymbol{p}}}$$

notice that the expected return $\bar{\boldsymbol{r}}_{\text{test}}$ and covariance $\boldsymbol{\Sigma}_{\text{test}}$ shall be calculated using similar formula as (1.6) but with the *testing* data that are not seen during the portfolio optimization. Notice that if $\boldsymbol{p} = \boldsymbol{0}$, then we define $\text{Sharpe ratio} = 0$. For your convenience, we have prepared a function that are included with the preloaded file `reusablefunc.jl` and it can be called as `sharpe_ratio(path, x, w, c)`, with the inputs as

- `path` – the relative path in which the testing data are stored.
- `x`, `w`, `c` – the change in portfolio $\boldsymbol{x}$ and the original portfolio $\boldsymbol{w}$, and the flat cost $\$c$ for each transaction.

The function will report (i) the Sharpe ratio of the portfolio $\boldsymbol{p} = \boldsymbol{w} + \boldsymbol{x}$, (ii) the expected return, (iii) the total transaction cost, and (iv) the total portfolio value.

Task 7: Evaluate the Portfolio Performance on Testing Set (5%)

Compare the solutions found in Task 5 and 6 under the given parameters, in terms of the Sharpe ratio, expected return, and the total transaction cost. Comment on the differences.

Remark: Note that ideally, the MI-NLP formulation shall yield the best balance between total cost and Sharpe ratio as it is taking the flat cost model into account directly.

For example, you may observe the following output using the helper code provided:

```
In [40]: sharpe_IP = sharpe_ratio( path_subgroup, JuMP.value.(x_mip), ones(n), 2 );
         Sharpe Ratio = 0.09796420037619875, Return = 0.005377281279348732, Tx Cost = 30, Portfo Value = 9.849134464463981

In [41]: sharpe_SOCP = sharpe_ratio( path_subgroup, JuMP.value.(x_socp), ones(n) , 2 );
         Sharpe Ratio = 0.07702244117000454, Return = 0.004190053139613171, Tx Cost = 40, Portfo Value = 9.76288367358269

In [42]: sharpe_Opt = sharpe_ratio( path_subgroup, portfolio_opt, zeros(n), 2 );
         Sharpe Ratio = 0.0589573629618575, Return = 0.0034910503428275215, Tx Cost = 40, Portfo Value = 10.000000000000004
```

For further readings on modeling transaction cost in Portfolio optimization, please refer to the article [Lobo et al., 2007].

1.2 Competitive Tasks (30%)

The goal of this competitive task is to implement your own (approximate) solver to the portfolio optimization problem, without relying on JuMP and its optimizers such as ECOS, Juniper. To motivate, we observe that while optimization packages such as JuMP are very convenient to use, yet these solvers are limited by their high complexity for large-scale problems when $n \gg 1$.

To prepare for this task, we adopt two modifications to search for a portfolio with the *best performance* and *lowest cost*, as follows:

[2]Note that we have assumed that the risk free asset has zero return.

- From Task 3, you should have formulated a portfolio optimization problem in the following form (before you transform the problem into an SOCP):

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \quad & \frac{1}{2}(\boldsymbol{x}+\boldsymbol{w})^\top \boldsymbol{\Sigma}(\boldsymbol{x}+\boldsymbol{w}) \\ \text{s.t.} \quad & f_1(\boldsymbol{x}) \le 0, \quad f_2(\boldsymbol{x}) \le 0, \\ & l_i \le x_i \le u_i, \ i=1,...,n, \end{aligned}$$

  where f_1, f_2 are functions of the portfolio $\boldsymbol{x} \in \mathbb{R}^n$ and l_i, u_i are known constants that depend on w_i, M (notice that from Task 2/3, we have $x_i + w_i \ge 0, \ |x_i| \le M$). A heuristic for solving the above problem is to consider the following approximation:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \quad & \frac{1}{2}(\boldsymbol{x}+\boldsymbol{w})^\top \boldsymbol{\Sigma}(\boldsymbol{x}+\boldsymbol{w}) + \lambda f_1(\boldsymbol{x}) + \upsilon f_2(\boldsymbol{x}) \\ \text{s.t.} \quad & l_i \le x_i \le u_i, \ i=1,...,n, \end{aligned} \tag{1.7}$$

  for some $\lambda, \upsilon > 0$.

  For your information, this method is called the penalty method which adds constraints to the objective function by imposing a cost to infeasible points [cf. This idea is related to the Big-M simplex method]. For instance, since we desired that $f_1(\boldsymbol{x}) \le 0$ in the original formulation, the approximation shall impose a positive cost if $f_1(\boldsymbol{x}) > 0$ [recall that $\lambda > 0$]. Moreover, intuitively increasing λ (resp. υ) ensures that the constraint $f_1(\boldsymbol{x}) \le 0$ (resp. $f_2(\boldsymbol{x}) \le 0$) is more strictly enforced.

- The second modification is to adopt an alternatve transaction cost model to (1.4) or (1.5) with the Huber function, parameterized by $\delta, a > 0$, as:

$$\sum_{i=1}^n \phi_{\mathsf{h},\delta}(x_i) \quad \text{where} \quad \phi_{\mathsf{h},\delta}(x_i) = \begin{cases} \dfrac{a}{2\delta}x_i^2, & \text{if } |x_i| \le \delta, \\ a\big(|x_i| - \frac{1}{2}\delta\big), & \text{if } |x_i| > \delta, \end{cases} \tag{1.8}$$

  which yields a better approximation than (1.5) for the transaction cost model in (1.4).

Finally, observe that (1.7) is now an optimization problem with a 'simple' constraint:

$$X = \{\boldsymbol{x} \in \mathbb{R}^n : l_i \le x_i \le u_i, \ i=1,...,n\},$$

where X is a convex set, and the projection into this set can be easily calculated. In other words, with a differentiable and convex f_1, f_2, it is easy to apply methods such as projected gradient, conditional gradient methods to solve (1.7); see Appendix A. This is the goal of the main task in this section:

Task 8: Customized Solver for Portfolio Optimization (30%)

Using the complete dataset with the training data from $n = 471$ stocks stored in `/ftec_project_files/`. Implement a projection-based iterative algorithm to tackle (1.7) with the Huber transaction cost model (1.8) which approximates the SOCP in Task 3 with the settings:

$$M = 20, \ B = 20, \ a = 2, \ R_{\mathsf{d}} = 1.01\textstyle\sum_{i=1}^n \bar{r}_i, \ \boldsymbol{w} = \mathbf{1}.$$

Notice that the budget constraint B and expected return goal R_{d} are irrelevant in this task as the additive constants in the constraints shall have no effects on (1.7). As a recommendation, you can apply the projected gradient descent method using a constant step size.

You may need to tune the parameters $\upsilon, \lambda, a, \delta$, as well as the step size, to achieve the best result. Let $\boldsymbol{x}^\star$

be the solution found using your algorithm, to 'sparsify' your solution (and thus reduce the transaction cost), you may also apply the post-processing step:

$$x_i' = \begin{cases} x_i^\star, & \text{if } |x_i^\star| \geq \delta, \\ 0, & \text{if } |x_i^\star| < \delta. \end{cases}, \quad \forall\, i = 1, ..., n,$$

and output $\boldsymbol{x}'$ in the above as the optimized change in portfolio (this postprocessing step has been written for you in the helper's code).

For the iterative algorithm applied, you are required to initialize by $\boldsymbol{x}^0 = \mathbf{0}$ and the algorithm has to be run for at least 10^4 iterations. You are encouraged to try more than one algorithm (which will be counted towards the 'innovation' section in the Report assessment).

Assessment You will receive a maximum of **10%** for implementing at least one numerical algorithm (e.g., projected gradient) **correctly**, together with (i) plotting the trajectory of the algorithm is show that the objective value in (1.7) is decreasing and providing comments on the algorithm(s) implemented, (ii) providing the derivations on why the implemented algorithm can be used to solve (1.7).

The remainder of your marks will be calculated according to the following formula evaluated using the testing set data:

$$\begin{aligned} \text{Score} = 10\% \times \frac{\min\{3, \text{Your Sharpe Ratio}\}}{\min\{3, \text{Highest Sharpe Ratio}\}} + 5\% \times \frac{\min\{1, \text{Your Return}\}}{\min\{1, \text{Highest Return}\}} \\ + 5\% \times \frac{\max\{200, \text{Lowest Tx. Cost}\}}{\max\{200, \text{Your Tx. Cost}\}}. \end{aligned} \tag{1.9}$$

The portfolio value is evaluated as $\sum_{i=1}^n (x_i + p_i)$. In the above, the highest Sharpe ratio (resp. lowest cost, etc.) are calculated as the highest one (resp. lowest one) among the class of FTEC2101[3]. Notice that the transaction cost in the above is calculated using the flat cost model (1.4) with $c = 2$. They can be calculated by the `sharpe_ratio` function described before, as done in the helper's code, e.g.,

```
In [32]: sharpe_PPGD = sharpe_ratio( "./ftec_project_files/", x_custom_pp, ones(n) , 2 )
         Sharpe Ratio = 0.061660369284455854, Return = 0.1432734933667747, Tx Cost = 0, Portfo Value = 471.0
Out[32]: 0.061660369284455854
```

If you have tried more than one algorithm and/or more than one set of parameters, you can select **only one set of portfolio** for the competition. Please indicate which set of portfolio is selected in your report and include that in the submission of your program files. Moreover, please observe the following rules:

- You are **not allowed** to directly optimize on the testing set data. In other words, your iterative algorithm should not 'touch' any data file with the suffix `_test.csv` as you are not supposed to see the 'future' when investing. Your score in (1.9) will be set to zero if we detect such 'cheating' behavior. However, you can apply the function `sharpe_ratio` to estimate your performance as many times as you like.

[3]These constants will be calculated separately among the ESTR2520 students.

1.3 Report (20%)

You are required to compile a project report with answers to the questions posed in Task 1 to Task 8. You may structure the report according to the order of the tasks, for instance:

1. **Background and Introduction** — In this section, you can briefly introduce the problem, e.g., explaining the goal of portfolio design, discussing the role of optimization methods in portfolio design.
2. **Model and Theory** — In this section, you can discuss how the portfolio design problem is modeled as optimization problems. More specifically,
   – You shall begin by discussing the formulation (1.1) and then answer the questions in **Task 1**.
   – Next, you can describe the optimization models with transaction costs and then answer **Tasks 2 & 3**.
3. **Experiments** — In this section, you describe the experiments conducted to test your formulation, i.e.,
   – You shall first describe the dataset by answering **Task 4**. In addition, it is helpful to describe a few meta-data regarding the dataset, e.g., from when to when the stock price data is collected.
   – Then, you can describe the experiments for each of the 3 formulations, by answering **Tasks 5 & 6**.
   – Finally, you can compare the formulations by answering **Task 7**.
4. **Competitive Task** — In this section, you describe the custom solver you built to solve the large-scale portfolio design problem, i.e.,
   – You shall first describe the approximation technique as laid out in the discussion of (1.7), (1.8).
   – Then, you shall describe the iterative algorithm you have derived in **Task 8**.
   – Apply the iterative algorithm on the complete training dataset and show the objective value vs. iteration number. Discuss whether the algorithm converges and report on the performance of the designed portfolio (e.g., the Sharpe ratio, etc..).
5. **Conclusions** — In this section, you shall summarize the findings in the project, and discuss various aspects that can be improved with the formulation, etc..

Throughout the report, please feel free to write your answer which involves equations (e.g., Task 1-3) on a paper and scan it to your Word/PDF report as a figure. For Task 4 to 8, please include all the plots and comments as requested. For Task 8, please indicate the *Sharpe ratio*, *Return*, *Transaction Cost*, *Value of Portfolio* of **the selected solution**.

The program code has to be submitted separately. However, you are welcomed to use excerpts from the program codes if you find that it is helpful for explaining some of the concepts.

Lastly, you are welcomed to use online resources when preparing the project. However, you must give **proper references** for every sources that are not your original creation.

Assessment Here is a breakdown of the assessment metric for the report writing component.

- (10%) *Report Writing*: A project report shall be readable to a person with knowledge in optimization (e.g., your classmates in FTEC2101/ESTR2520). Make sure that your report is written with clarity, and more importantly, using your own language!

- (10%) *Innovation*: You can get innovation marks if you include extra experiments, presentations, etc.. that are relevant to the project (with sufficient explanations); see Appendix A for some recommendations.

1.4 Submission

This is an individual project. While discussions regarding *how* to solve the problems is encouraged, students should answer the problems on their own (just like your HWs). The deadline of submission is May 14, 2021, 23:59 (HK time). Please submit a single zip file with the following content:

- Your Project Report in PDF format.
- Your Program Codes [either in Jupyter notebook (`.ipynb`), or Julia code (`.jl`)].

In addition, all project reports shall be submitted to VeriGuide for plagiarism check.

A Additional Information

Several tips for implementing the MI-NLP, SOCP, etc.. have been included with the the template program. However, please be reminded that it is not necessary to follow all the tips therein.

Suggestions for Extensions — The below are only suggestions for the extensions. You are more than welcomed to propose new ideas!

- *Algorithm Aspect* – For Task 8, with the constraint structure given, the recommended algorithms are *projected gradient descent (PGD) method* and *conditional gradient (CG) method*, which are described as follows. For solving a general optimization:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; F(\boldsymbol{x}) \;\; \text{s.t.} \;\; l_i \le x_i \le u_i, \;\; i = 1, ..., n. \tag{1.10}$$

Let $X = \{\boldsymbol{x} \in \mathbb{R}^n \;:\; l_i \le x_i \le u_i, \;\; i = 1, ..., n\}$, these algorithms can be described as

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ with $l_i \le x_i^{(0)} \le u_i$ for all i, constant step size $\gamma > 0$, max. iteration number K_{max}.

For $k = 0, ..., K_{max}$

$\quad \boldsymbol{x}^{(k+1)} = \text{Proj}_X\big\{\boldsymbol{x}^{(k)} - \gamma \nabla F(\boldsymbol{x}^{(k)})\big\}$

End For

PGD Method

Input: $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ with $l_i \le x_i^{(0)} \le u_i$ for all i, max. iteration number K_{max}.

For $k = 0, ..., K_{max}$

$\quad \boldsymbol{a}^{(k)} = \arg\min_{\boldsymbol{a}\in X} \;\; \boldsymbol{a}^\top \nabla F(\boldsymbol{x}^{(k)})$

$\quad \boldsymbol{x}^{(k+1)} = (1 - \frac{2}{k+1})\boldsymbol{x}^{(k)} + \frac{2}{k+1}\boldsymbol{a}^{(k)}$

End For

CG Method

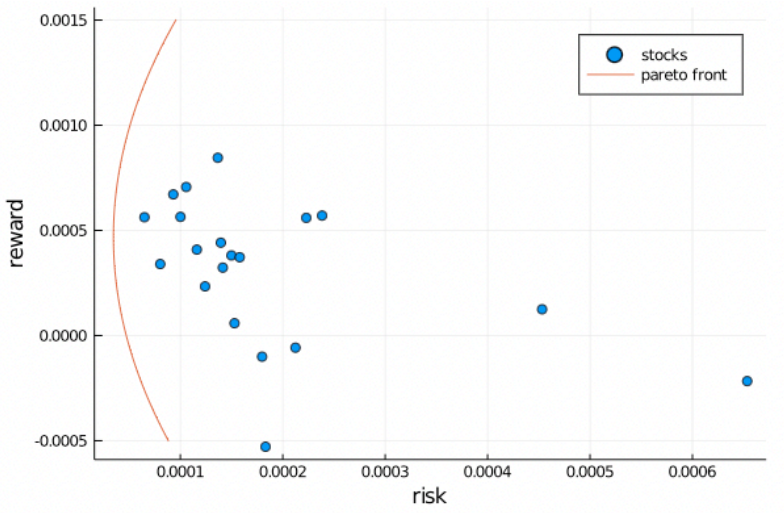
Moreover, the projection/CG operators can be easily computed: for any $\boldsymbol{g} \in \mathbb{R}^n$, we have

$$\boldsymbol{g}^{PG} = \text{Proj}_X(\boldsymbol{g}), \;\; \text{where} \;\; [\boldsymbol{g}^{PG}]_i = \begin{cases} g_i, & \text{if } g_i \in [l_i, u_i], \\ l_i, & \text{if } g_i < l_i, \\ u_i, & \text{if } g_i > u_i. \end{cases}$$

$$\boldsymbol{g}^{CG} = \arg\min_{\boldsymbol{a}\in X} \ \boldsymbol{a}^\top \boldsymbol{g}, \quad \text{where} \quad [\boldsymbol{g}^{CG}]_i = \begin{cases} u_i, & \text{if } g_i < 0, \\ l_i, & \text{if } g_i \geq 0. \end{cases}$$

It should not be very hard to program the above into a Julia function. Nevertheless, you are more than welcomed to explore the use of other iterative algorithms for solving the optimization at hand.

- *Presentation Aspect* – The comparison of the (unconstrained) optimal portfolio against the return/risk profile as done in Task 5 (b) is related to the *Efficient/Pareto Frontier* of an investment portfolio. To investigate the latter, you can plot the risk of the optimal portfolio (in Task 1) as a function of the desired reward, e.g., as follows



Importantly, this efficient frontier is found to be always 'enclosing' the reward/risk of each stock covered in the portfolio; see `https://en.wikipedia.org/wiki/Efficient_frontier` for further reference.

References

F. J. Fabozzi, P. N. Kolm, D. A. Pachamanova, and S. M. Focardi. *Robust portfolio optimization and management*. John Wiley & Sons, 2007.

M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research*, 152(1):341–365, 2007.