PEKING UNIVERSITY

Simulation Methods for Optimization and Learning

Globex
July 2023

Simulation Project

Lecturer: Bernd Heidergott
Tutor: Franssen Christian

Group Members:

Chen Xi
Gu Qinglong
Wu Zhangyi
Yin Longwei
Zeng Shaoyu

Abstract

The main objective of this project is to help investors find the optimal investment strategy to maximize the total return using the Simultaneous Perturbation Stochastic Approximation Algorithm (SPSA). To be specific, the report addresses the allocation of the investment under three scenarios with a total capital of 1.

The SPSA algorithm is well suited for dealing with noisy, non-smooth optimization problems and does not require gradient information, making it efficient. A Python program is written to implement the SPSA algorithm optimization according to the settings in the problem and other relevant factors. The program was validated using human rationale and demonstrated the effectiveness of the SPSA algorithm on an investment optimization problem.

Key Words: Mathematics, SPSA, Simulation, Optimization, Self-learning Algorithm,

Independent Uniform Distribution

Table of Content

1. Introduction 4
2. Description of The Optimization Approach 5
2.1 Optimization method: SPSA 5
2.2 Coefficients in the objective function 6
2.3 Projection 6
2.4 The optimization algorithm for streaming data 8
3. Description of the program 9
3.1 Procedures 9
3.2 Estimation of coefficients 9
3.3 Parameters 10
4. Validation and verification 11
4.1 Verification of SPSA 11
4.2 Validation by using Enumeration 11
5. Result 12
5.1 Result of Problem (a) 12
5.2 Result of Problem (b) 16
5.3 Result of Problem (c) 20
6. Interpretation of Our Findings 21
7. Conclusion 22
Reference 24
Acknowledgment 24

1. Introduction

Optimizing investment strategies is critical for investors to maximize returns while controlling risk. Benefiting from the rapid availability of financial big data and the continuous improvement of machine learning, practical algorithms for optimizing investment are gaining popularity among individual and institutional investors (Moghaddam et. al, 2016). Optimization-based investment approaches can lead to higher returns compared to non-optimization approaches. Therefore, optimal investment strategies based on optimal algorithms are becoming increasingly important for investors seeking to maximize portfolio performance.

Spall (1992) first introduced Simultaneous Perturbed Stochastic Approximation (SPSA). The algorithm is a gradient-free iterative algorithm, which can be effectively used for multivariate nonlinear optimization of complex systems in the absence of an accurate system model (Abdulsadda & Iqbal, 2011).

This report refers to use the SPSA algorithm to develop an optimization tool that efficiently finds the optimal investment strategy. The python program written implements the SPSA optimizer using the required input data and outputs the optimal investment strategy for the given problem:

There is a total capital of 1 to invest across n different companies. For each company i, its market value X_i, as well as a threshold x_i are given. If the market value X_i is below the threshold x_i, then investing in that company will yield a return of 0. If the market value X_i is above the threshold x_i, then investing a fraction of the total capital in that company will yield an actual return of Y_i For company i investing a fraction p_i of the capital yields expected return

$$p_i \mathbb{E}\left[Y_i 1_{X_i \geq x_i}\right]$$

for the company i. To find the optimal investment strategy indicated to

$$max \sum_{i=1}^{3} p_i \mathbb{E}\left[Y_i 1_{X_i \geq x_i}\right]$$

such that

$$\sum_{i=1}^{n} p_i = 1 \text{ and } 0 \leq p_i \leq 1$$

Addressing the following three problems:

(a) The small investor problem: In this case, the return on investment follows an independent uniform distribution within the range of $[0, X_i]$ for $i = 1,2,3$

(b) The big investor problem: In this case, the return on investment is influenced by the specific company in which the investment is made, uniformly distributed on $[0, i + X_i p_i]$ for $i = 1,2,3$

(c) The small investor problem for real-time: In this case, the objective is to construct an algorithm that can automatically identify the optimal investment strategy based on real-time data streaming. This algorithm does not depend on the distribution of X and Y, and is robust to the changes of features of the data revealed to it.

This report conducted extensive testing of the Python program to verify that the best strategies found were meaningful and robust. By identifying the optimal fraction of investment, this report demonstrates the effectiveness of the SPSA algorithm for investment optimization and to lay the groundwork for future improvements to the optimization tool.

2. Description of The Optimization Approach

2.1 Optimization method: SPSA

We need to maximize the objective function

$$J(p_1, p_2, p_3) = \sum_{i=1}^{3} p_i \mathbb{E}\left[Y_i 1_{X_i \geq x_i}\right],$$

for $p_1, p_2, p_3 \in [0,1]$, $p_1 + p_2 + p_3 = 1$

The optimization strategy for $p = (p_1, p_2, p_3)$ is

$$p_{n+1} = p_n + \varepsilon_n \cdot \nabla J^{SPSA}(p_n),$$

where n is the number of iterations or the times of updates, ε is the step size and could be set to fixed or decreasing. In our algorithm, for decreasing type, $\varepsilon_n = \frac{1}{n+1}$ and for fixed type, $\varepsilon_n = 0.1$

The Simultaneous Perturbation Stochastic Approximation (SPSA) is applied to estimate the gradients:

$$\nabla J^{SPSA}(p_n) = \frac{J(p_n + \eta_n \Delta_n) - J(p_n - \eta_n \Delta_n)}{2\eta_n \Delta_n}$$

where η_n is set as $\frac{1}{n+1}$, $\{\Delta_n\}$ is an iid sequence and $\Delta_n(k)$ is a 3-vector with each entry be the random choice between -0.01 and 0.01.

We also consider both the batch method and the one-shot method to estimate the gradient. For the one-shot method, only one realization of Δ_n is made to evaluate the gradient. For the batch method, the gradient will be evaluated multiple times using different Δ_n each time and the estimated gradient will be the average value of evaluations.

2.2 Coefficients in the objective function

In objective function $J(p_1, p_2, p_3) = \sum_{i=1}^{3} p_i \mathbb{E}[Y_i 1_{X_i \geq x_i}]$, the coefficients of p are the expected values of the production of two variables Y_i and $1_{X_i \geq x_i}$.

In our solution, we sample these two variables and estimate the expectation value by the average of the samples. The sample procedure will be explained further in Description of the program.

2.3 Projection

In the updating process

$$\tilde{p}_{n+1} = p_n + \varepsilon_n \cdot \nabla J^{SPSA}(p_n),$$

$\tilde{p}_{n+1}$ might not be in the feasible region of the problem, which is the set

$$A = \{(p_1, p_2, p_3) |\ p_1, p_2, p_3 \geq 0,\ p_1 + p_2 + p_3 = 1\}.$$

Geometrically the region is a bounded triangular plane in 3-D space as shown in firgure 2.3.1.

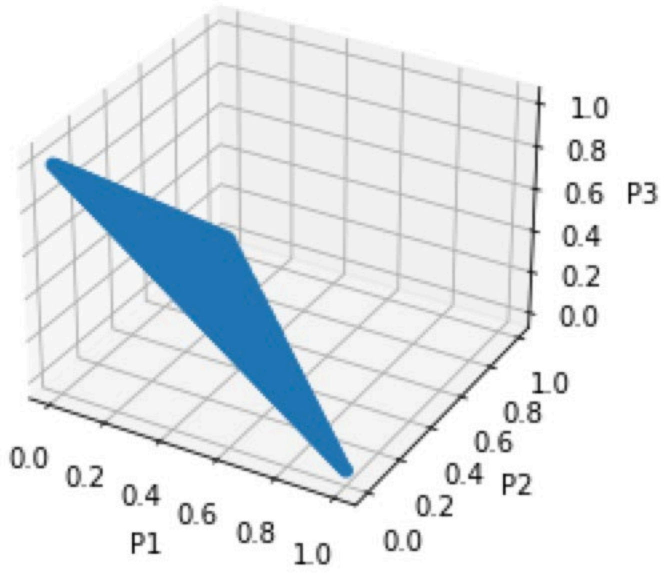


Figure 2.3.1 Bounded Region

The idea is to map the intermediate point $\tilde{p}_{n+1}$ to the nearest point p in the feasible region if it goes out of the boundary.

$$f : \mathbb{R}^3 \to A$$
$$f(\tilde{p}_{n+1}) = \underset{p \in A}{\operatorname{argmin}} \|\tilde{p}_{n+1} - p\|$$

There are different cases for projecting $\tilde{p}_{n+1}$. In some cases, $\tilde{p}_{n+1}$ will be projected to the boundary of A, while in other cases, $\tilde{p}_{n+1}$ will be projected to the inner part of A. Both cases are taken into consideration in the algorithm we designed.

After we get

$$\tilde{p}_{n+1} = p_n + \varepsilon_n \cdot \nabla J^{SPSA}(p_n),$$

we 'truncate' the negative entries of $\tilde{p}_{n+1}$ to zero since the investment p_i should be non-negative.

$$\tilde{p}_{n+1}^{(1)} = \max(\tilde{p}_{n+1}, 0)\ (elementwise)$$

Then we project $\tilde{p}_{n+1}^{(1)}$ to the point $\tilde{p}_{n+1}^{(2)}$ in the plane $p_1 + p_2 + p_3 = 1$:

$$\tilde{p}_{n+1}^{(2)}(i) = \begin{cases} \tilde{p}_{n+1}^{(1)}(i) + d \cdot \vec{n}, & p_i > 0 \\ 0, & otherwise \end{cases}$$

In this expression, $d = \frac{p_1+p_2+p_3-1}{count\_nonzero(\tilde{p}_{n+1}^{(1)})}$ is the distance between point $\tilde{p}_{n+1}^{(1)}$ and the bounded region. Function $count\_nonzero()$ counts the number of nonzero elements in the input, and $\vec{n} = (1,1,1)$.

Then we will repeat the last 2 steps until $\tilde{p}_{n+1}$ is projected in the feasible region. This is a carefully designed process convenient to implement in code and robust to various $\tilde{p}_{n+1}$. Basically, the idea is as mentioned above, to send $\tilde{p}_{n+1}$ back to the feasible region.

2.4 The optimization algorithm for streaming data

Our optimization approach seeks to adaptively allocate capital over *n* companies based on streaming data, aiming to maximize the same objective function specified above.

For the first half of iterations, X_i values are generated using:

$$X_1 \sim N(2, 1^2)$$
$$X_2 \sim N(3, 1.7^2)$$
$$X_3 \sim N(1, 1.2^2)$$

To mimic a change in the market dynamics, for the second half of iterations, X_i values are generated using:

$$X_1 \sim N(2.5, 1^2)$$
$$X_2 \sim N(4, 1^2)$$
$$X_3 \sim N(0, 1^2)$$

and Y_i are given by:

$$Y_i \sim U(0, X_i)$$

The indicator function is the same as described above.
Using the potential returns and the indicator function, we allocate capital to the company with the maximum expected return:

$$fraction\left[\underset{i}{\operatorname{argmax}}(parameter)\right] = 1$$

where parameter is the product of potential returns Y_i and the indicator function $1_{X_i \geq x_i}$, andarg $\max(parameter)$ will return the index of company whose return is the largest one.

The cumulative result, representing the evolving allocation, is updated at every step:

$$RESULT[i+1] = RESULT[i] + (fraction - RESULT[i])/(i+1)$$

This equation ensures that the allocation adaptively changes with incoming data, averaging over past decisions.

3. Description of the program

This part describes our program's procedure and explains the parameters used.

3.1 Procedures

Before running our program, some parameters could be changed to yield different outputs. This will be elaborated on later.

First, we determine the investment type as mentioned in the problem, which is either small or big, affecting the distribution of the return on investment, Y_i. More specifically, in small investor problem, Y_i are independent uniformly distributed on $[0, X_i]$, while in big investor problem, Y_i are uniformly distributed on $[0, i + X_i p_i]$. Since the interval of latter one is dynamic, the estimation of expection of coefficients in SPSA will be different. What's more, in big investor problem, such estimation is needed after every iteration.

Then, we will run the SPSA algorithm 1000 times (i.i.d. replications) to produce 1000 outputs p^* for output analysis. In each run, we will store the values of fractions, gradients, and objective functions in the whole iteration. The final result in every run will also be stored.

Last, we will make the histograms of p^* and $J(p^*)$. In order to visualise the updates of p along the iteration, we also plot this process for last relpications, showing the variations of p_n and $J(p_n)$. An auxiliary line of asymptotically result for the objective function is also plotted.

For question (c), we apply an algorithm different from (a) and (b) which is introduced in part 2.4.

3.2 Estimation of coefficients

The procedure of SPSA algorithm has already been explained in part 2 except the estimation of coefficients. We need to estimate the following

$$\mathbb{E}\left[Y_i 1_{X_i \geq x_i}\right] \text{ for } i = 1,2,3.$$

First, we use code to generate a realization of random variables X_i according to the formulas:

$$X_i = \frac{\rho V + \sqrt{1-\rho^2}\eta_i}{\max(W,1)}, \quad i = 1,2,3,$$

where, according to the background settings of the problems, we have independently distributed variables η_i, V and W:

$$\eta_i \sim N(0, i) \text{ for } i = 1,2,3, \text{ modelling the i-th company's idiosyncratic risk,}$$

$V \sim N(0,1)$, modelling the common factor that affects the economy,

$W \sim Exp(\frac{1}{0.3})$, modelling common market shocks,

and weight factor $\rho = 0.6$.

Then we can determine the value of indicator function $1_{X_i \geq x_i}$ according to the realization of (X_1, X_2, X_3):

$$1_{X_i \geq x_i} = \begin{cases} 1, & X_i \geq x_i \\ 0, & X_i < x_i \end{cases}, for\ i = 1,2,3$$

where (x_1, x_2, x_3) is the threshold of getting profit which is set to $(2,3,1)$.

Next, we generate a realization of Y_i correspond to X_i. In problem (a) we have $Y_i \sim U(0, X_i)$ and in problem (b) we have $Y_i \sim U(0, i + X_i p_i)$.

Since we get the values of Y_i and $1_{X_i \geq x_i}$, the production of them is a realization of return. In our algorithm, 1000 samples of such return are made, and we use the average value to estimate the expectation of the coefficients in the end.

3.3 Parameters

In the program, there are various parameters used in purpose of adjusting the result and applying different methods. Important parameters are listed below with explanation shown in Table 3.3.

Table 3.3.1 Description of Parameters

Name in python code	Values	Meaning and usage
STOCHASTIC	True or False	Determine whether to add normal distributed noise when evaluating objection. Set to True if noise is needed
BATCH	True of False	Determine whether to use batch method when estimate the gradient. Set to True if batch method is needed
NR_ITERATION	100	The number of iterations or updates
EPSILON_TYPE	‘fixed’ or ‘decreasing’	Determine the type of step size
NR_SAMPLES	100	The number of samples in estimation of coefficients
nn	1000, 500, or smaller	The number of replications of running the algorithm. The number of outputs.

Our program could handle various requirement flexibly by modifying the parameters above and produce different output for our analysis.

4. Validation and verification

4.1 Verification of SPSA

In SPSA, according to the work in lecture notes,

$$\mathbb{E}[\nabla J^{SPSA}(p_n)] = \nabla J(p_n) + \beta_n$$

where β_n is the bias between true gradient of objective function and our estimation.
We obeserve that

$$D^{\iota}J(p) \text{ is } \mathbf{0} \text{ for all } \iota \text{ with } |\iota| = 3.$$

So, for all n

$$||\beta_n|| \leq \eta_n^2 \delta$$

for some bound δ. To make sure the convergence of SPSA, it is required that

$$\sum_n \epsilon_n \delta \eta_n^2 < \infty$$

Since $\eta_n = \frac{1}{n+1}$ is of order $\mathcal{O}(n^{-1})$, the requirement for convergence is satisfied.

4.2 Validation by using Enumeration

To demonstrate the reliability of the SPSA algorithm in the previously mentioned investment problem, we can use enumeration method to evaluate all values of objective function with all possible fractions.

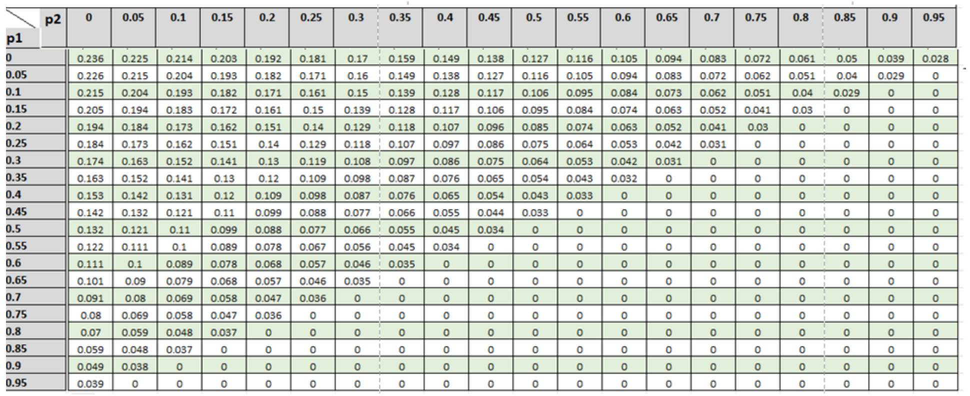
We create a list of p_1, taking 20 values uniformly from range $[0,1)$, namely $[0, 0.05, 0.1, 0.15, \dots, 0.95]$. Then a list of p2 taking values from $[0, 1 - p_1]$. Then we obtain corresponding list of p_3 by $p_3 = 1 - p_1 - p_2$. And the coefficients will be estimated accordingly. At last, we could obtain the value of objective function.

After running this algorithm multiple times, we can compare the results to see if the same company is always recognized as the most profitable choice. If the 3rd company is consistently recognized as the most profitable investment option over multiple runs, this will demonstrate the reliability and effectiveness of the SPSA algorithm in solving the investment problem.

As a result, we consistently observe from this numeration method that:

For problem (a), you can see when $p_1 = 0, p_2 = 0, p_3 = 1$, the total return is 0.236, which is the maximum value (shown in Table 4.2.1).

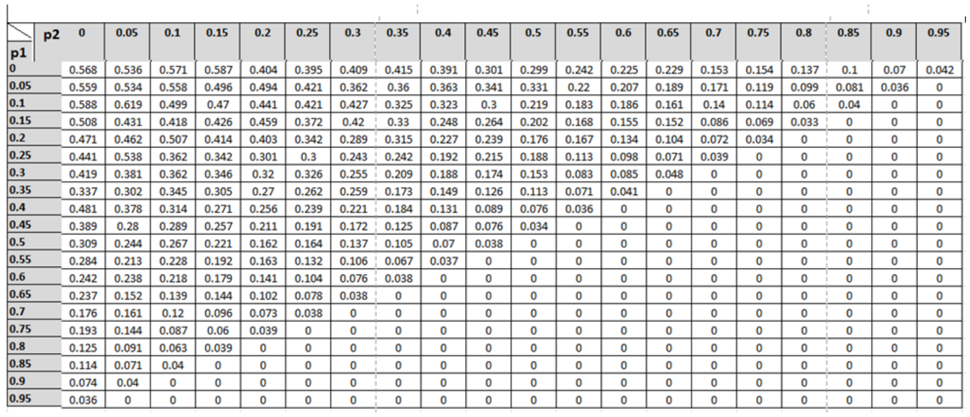
Table 4.2.1 Solution of Maximum Total Return for Problem (a) Using Enumeration Method



p1 \ p2	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
0	0.236	0.225	0.214	0.203	0.192	0.181	0.17	0.159	0.149	0.138	0.127	0.116	0.105	0.094	0.083	0.072	0.061	0.05	0.039	0.028
0.05	0.226	0.215	0.204	0.193	0.182	0.171	0.16	0.149	0.138	0.127	0.116	0.105	0.094	0.083	0.072	0.062	0.051	0.04	0.029	0
0.1	0.215	0.204	0.193	0.182	0.171	0.161	0.15	0.139	0.128	0.117	0.106	0.095	0.084	0.073	0.062	0.051	0.04	0.029	0	0
0.15	0.205	0.194	0.183	0.172	0.161	0.15	0.139	0.128	0.117	0.106	0.095	0.084	0.074	0.063	0.052	0.041	0.03	0	0	0
0.2	0.194	0.184	0.173	0.162	0.151	0.14	0.129	0.118	0.107	0.096	0.085	0.074	0.063	0.052	0.041	0.03	0	0	0	0
0.25	0.184	0.173	0.162	0.151	0.14	0.129	0.118	0.107	0.097	0.086	0.075	0.064	0.053	0.042	0.031	0	0	0	0	0
0.3	0.174	0.163	0.152	0.141	0.13	0.119	0.108	0.097	0.086	0.075	0.064	0.053	0.042	0.031	0	0	0	0	0	0
0.35	0.163	0.152	0.141	0.13	0.12	0.109	0.098	0.087	0.076	0.065	0.054	0.043	0.032	0	0	0	0	0	0	0
0.4	0.153	0.142	0.131	0.12	0.109	0.098	0.087	0.076	0.065	0.054	0.043	0.033	0	0	0	0	0	0	0	0
0.45	0.142	0.132	0.121	0.11	0.099	0.088	0.077	0.066	0.055	0.044	0.033	0	0	0	0	0	0	0	0	0
0.5	0.132	0.121	0.11	0.099	0.088	0.077	0.066	0.055	0.045	0.034	0	0	0	0	0	0	0	0	0	0
0.55	0.122	0.111	0.1	0.089	0.078	0.067	0.056	0.045	0.034	0	0	0	0	0	0	0	0	0	0	0
0.6	0.111	0.1	0.089	0.078	0.068	0.057	0.046	0.035	0	0	0	0	0	0	0	0	0	0	0	0
0.65	0.101	0.09	0.079	0.068	0.057	0.046	0.035	0	0	0	0	0	0	0	0	0	0	0	0	0
0.7	0.091	0.08	0.069	0.058	0.047	0.036	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.75	0.08	0.069	0.058	0.047	0.036	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.8	0.07	0.059	0.048	0.037	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.85	0.059	0.048	0.037	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.9	0.049	0.038	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.95	0.039	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For problem (b), you can see when $p_1 = 0, p_2 = 0, p_3 = 1$, the total return is 0.568, which is the maximum value (shown in Table 4.2.2).

Table 4.2.2 Solution of Maximum Total Return for Problem (b) Using Enumeration Method



p1 \ p2	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
0	0.568	0.536	0.571	0.587	0.404	0.395	0.409	0.415	0.391	0.301	0.299	0.242	0.225	0.229	0.153	0.154	0.137	0.1	0.07	0.042
0.05	0.559	0.534	0.558	0.496	0.494	0.421	0.362	0.36	0.363	0.341	0.331	0.22	0.207	0.189	0.171	0.119	0.099	0.081	0.036	0
0.1	0.588	0.619	0.499	0.47	0.441	0.421	0.427	0.325	0.323	0.3	0.219	0.183	0.186	0.161	0.14	0.114	0.06	0.04	0	0
0.15	0.508	0.431	0.418	0.426	0.459	0.372	0.42	0.33	0.248	0.264	0.202	0.168	0.155	0.152	0.086	0.069	0.033	0	0	0
0.2	0.471	0.462	0.507	0.414	0.403	0.342	0.289	0.315	0.227	0.239	0.176	0.167	0.134	0.104	0.072	0.034	0	0	0	0
0.25	0.441	0.538	0.362	0.342	0.301	0.3	0.243	0.242	0.192	0.215	0.188	0.113	0.098	0.071	0.039	0	0	0	0	0
0.3	0.419	0.381	0.362	0.346	0.32	0.326	0.255	0.209	0.188	0.174	0.153	0.083	0.085	0.048	0	0	0	0	0	0
0.35	0.337	0.302	0.345	0.305	0.27	0.262	0.259	0.173	0.149	0.126	0.113	0.071	0.041	0	0	0	0	0	0	0
0.4	0.481	0.378	0.314	0.271	0.256	0.239	0.221	0.184	0.131	0.089	0.076	0.036	0	0	0	0	0	0	0	0
0.45	0.389	0.28	0.289	0.257	0.211	0.191	0.172	0.125	0.087	0.076	0.034	0	0	0	0	0	0	0	0	0
0.5	0.309	0.244	0.267	0.221	0.162	0.164	0.137	0.105	0.07	0.038	0	0	0	0	0	0	0	0	0	0
0.55	0.284	0.213	0.228	0.192	0.163	0.132	0.106	0.067	0.037	0	0	0	0	0	0	0	0	0	0	0
0.6	0.242	0.238	0.218	0.179	0.141	0.104	0.076	0.038	0	0	0	0	0	0	0	0	0	0	0	0
0.65	0.237	0.152	0.139	0.144	0.102	0.078	0.038	0	0	0	0	0	0	0	0	0	0	0	0	0
0.7	0.176	0.161	0.12	0.096	0.073	0.038	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.75	0.193	0.144	0.087	0.06	0.039	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.8	0.125	0.091	0.063	0.039	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.85	0.114	0.071	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.9	0.074	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.95	0.036	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5. Result

5.1 Result of Problem (a)

For problem (a) of this project, the optimization result for fractions is

$$p_1 = 0, p_2 = 0, p_3 = 1$$

We have a few outputs based on the parameters setting on STOCHASTIC and BATCH as shown below:

The fraction of investment for each company and investment return through iteration from the last process of optimization (1000 times in total), also we include the 95% confidence interval of the investment return in our experiment.

Result 1:
The parameter setting for this result is:
STOCHASTIC = False
BATCH = False

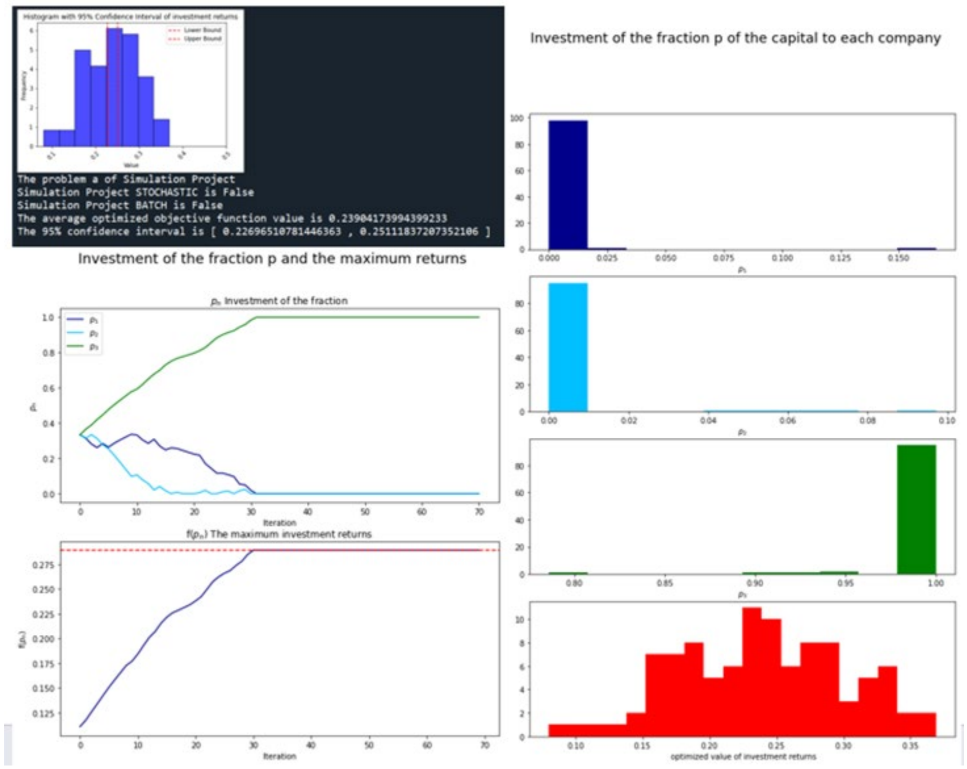
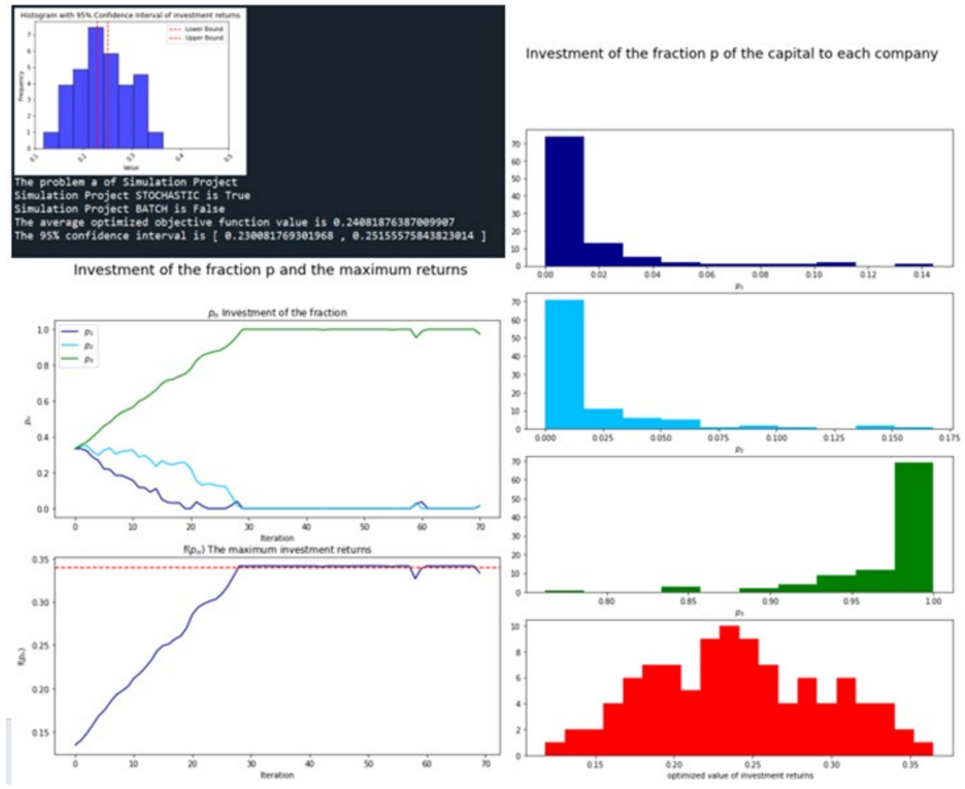

```
The problem a of Simulation Project
Simulation Project STOCHASTIC is False
Simulation Project BATCH is False
The average optimized objective function value is 0.23904173994399233
The 95% confidence interval is [ 0.22696510781446363 , 0.25111837207352106 ]
```

Figure 5.1.1 Result 1 of Problem (a)

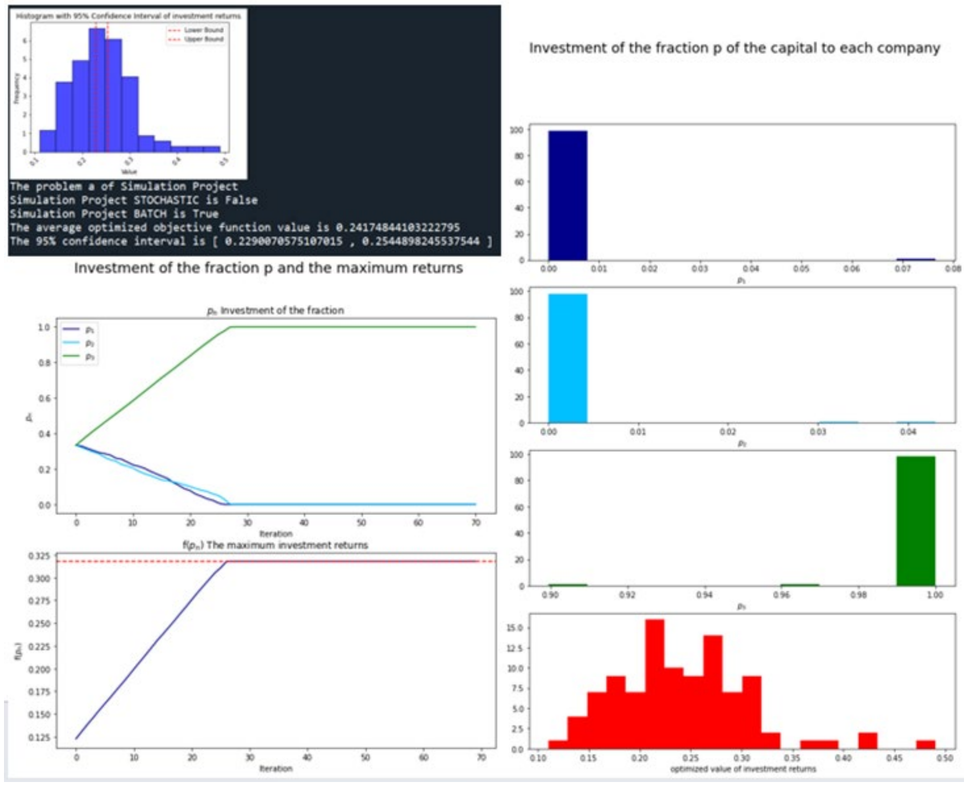
Result 2:
The parameter setting for this result is:
STOCHASTIC = True
BATCH = False



```
The problem a of Simulation Project
Simulation Project STOCHASTIC is True
Simulation Project BATCH is False
The average optimized objective function value is 0.24081876387009907
The 95% confidence interval is [ 0.230081769301968 , 0.25155575843823014 ]
```

Figure 5.1.2 Result 2 of Problem (a)

Result 3:
The parameter setting for this result is:
STOCHASTIC = False
BATCH = True



```
The problem a of Simulation Project
Simulation Project STOCHASTIC is False
Simulation Project BATCH is True
The average optimized objective function value is 0.24174844103222795
The 95% confidence interval is [ 0.2290070575107015 , 0.2544898245537544 ]
```

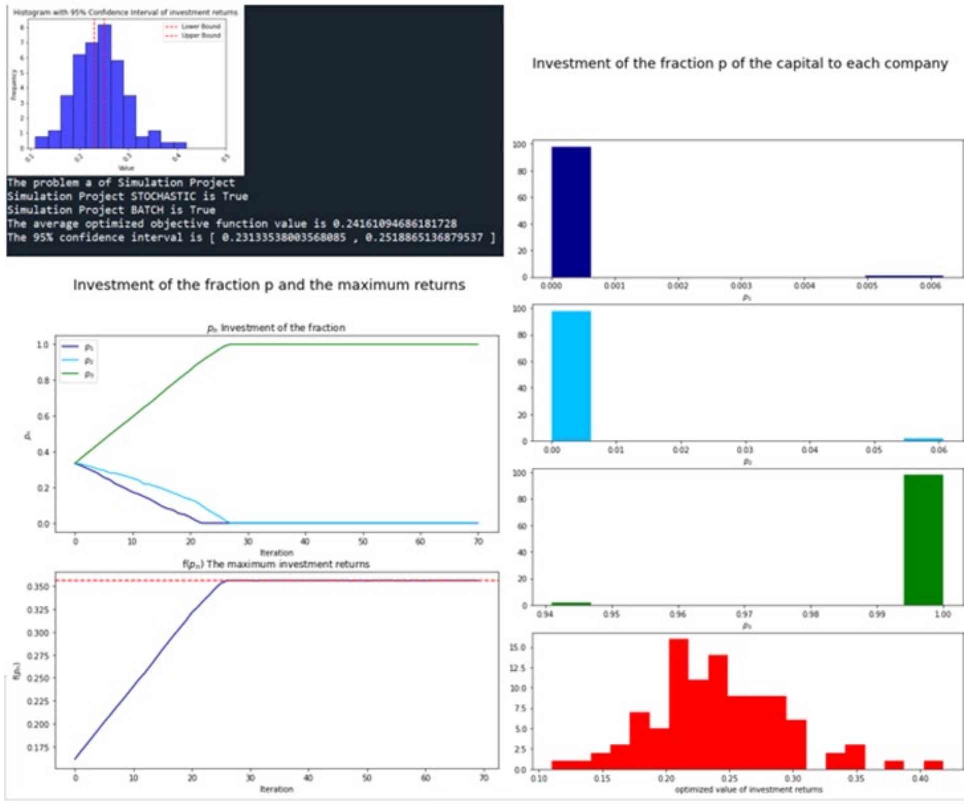
Figure 5.1.3 Result 3 of Problem (a)

Result 4:

The parameter setting for this result is:

STOCHASTIC = True

BATCH = True



```
The problem a of Simulation Project
Simulation Project STOCHASTIC is True
Simulation Project BATCH is True
The average optimized objective function value is 0.24161094686181728
The 95% confidence interval is [ 0.23133538003568085 , 0.2518865136879537 ]
```

Figure 5.1.4 Result 4 of Problem (a)

5.2 Result of Problem (b)

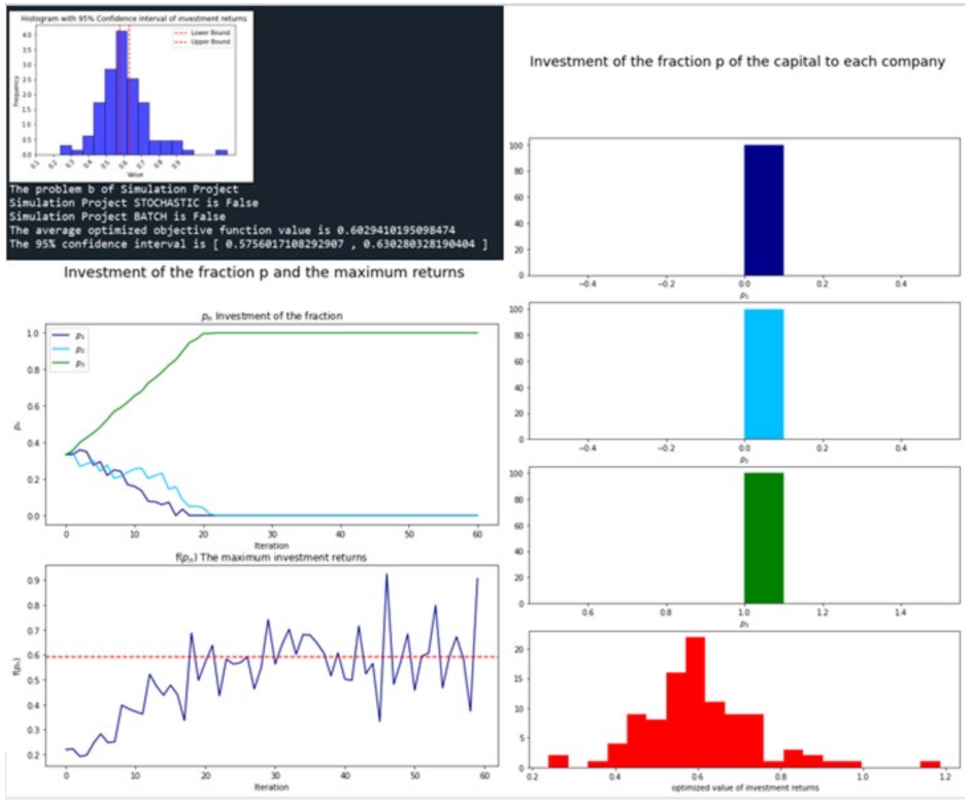
For problem (b) of this project, the optimization result for fractions is

$$p_1 = 0, p_2 = 0, p_3 = 1$$

We have a few outputs based on the parameters setting on STOCHASTIC and BATCH
as shown below:

the fraction of investment of each company and investment return through iteration from the last process of optimization (1000 times in total), also we include the 95% confidence interval of the investment return in our experiment.

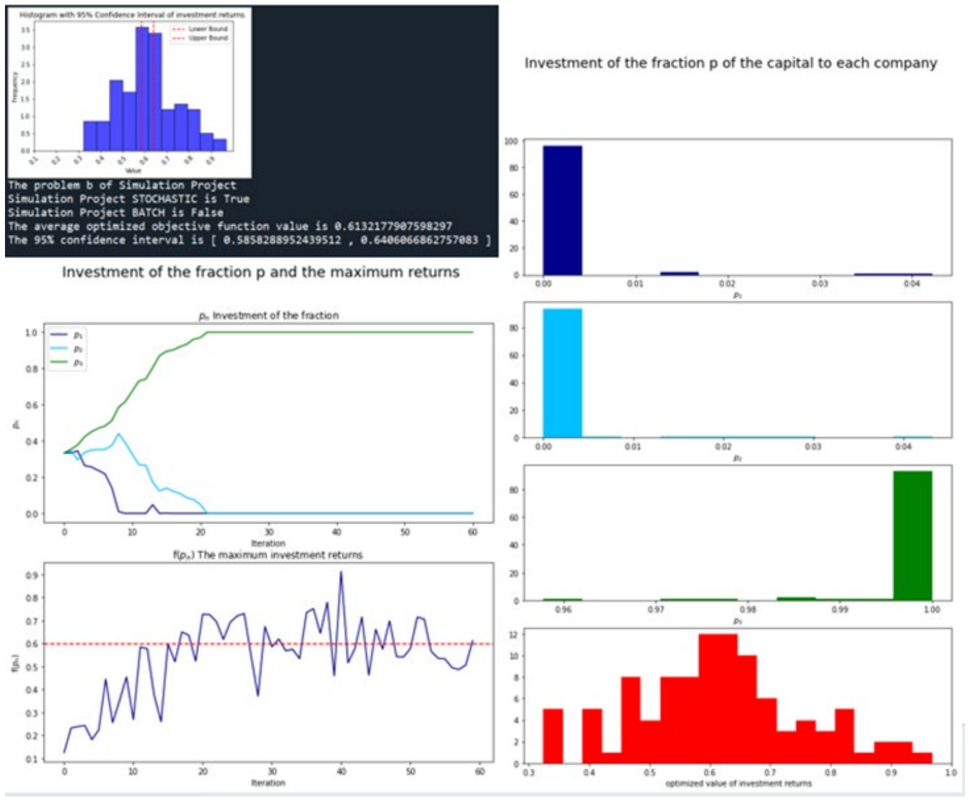
Result 1:
The parameter setting for this result is:
STOCHASTIC = False
BATCH = False



```
The problem b of Simulation Project
Simulation Project STOCHASTIC is False
Simulation Project BATCH is False
The average optimized objective function value is 0.6029410195098474
The 95% confidence interval is [ 0.5756017108292907 , 0.6302803281904O4 ]
```

Figure 5.2.1 Result 1 of Problem (b)

Result 2:
The parameter setting for this result is:
STOCHASTIC = True
BATCH = False



```
The problem b of Simulation Project
Simulation Project STOCHASTIC is True
Simulation Project BATCH is False
The average optimized objective function value is 0.6132177907598297
The 95% confidence interval is [ 0.5858288952439512 , 0.6406066862757083 ]
```

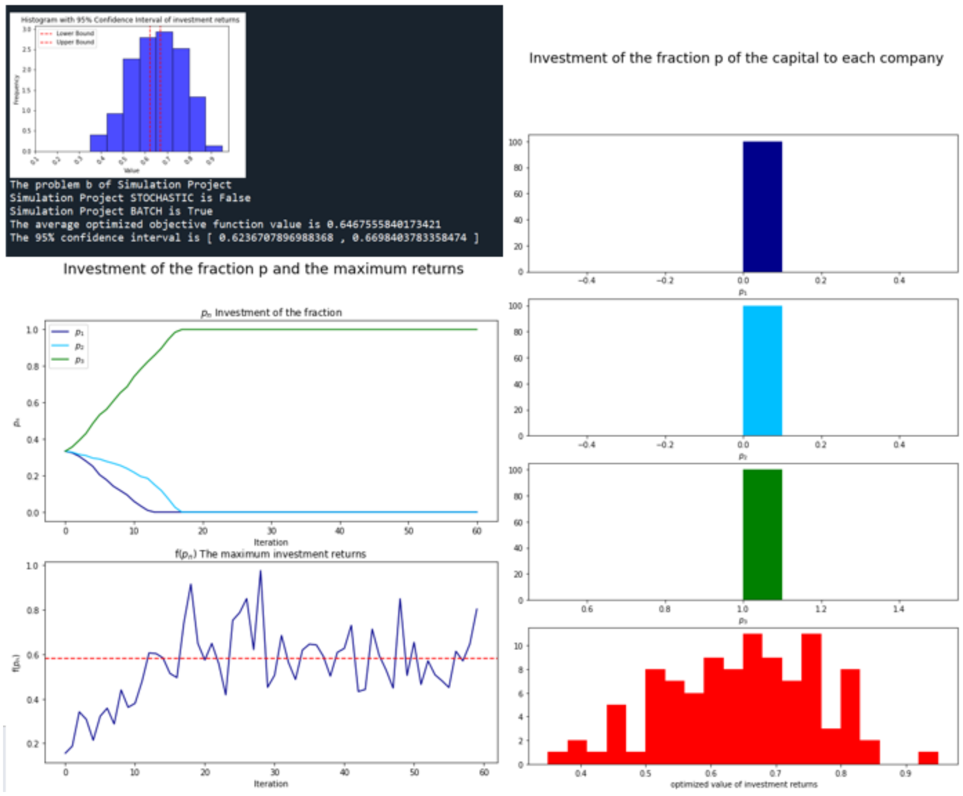
Figure 5.2.2 Result 2 of Problem (b)

Result 3:

The parameter setting for this result is:

STOCHASTIC = False

BATCH = True



```
The problem b of Simulation Project
Simulation Project STOCHASTIC is False
Simulation Project BATCH is True
The average optimized objective function value is 0.6467555840173421
The 95% confidence interval is [ 0.6236707896988368 , 0.6698403783358474 ]
```

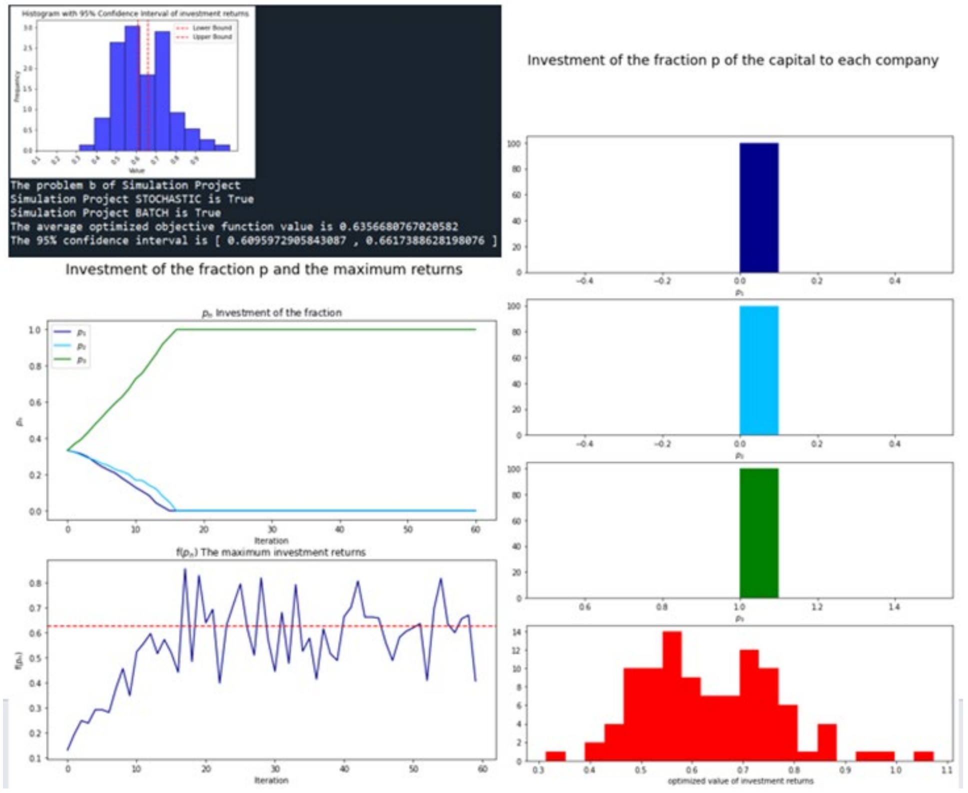
Figure 5.2.3 Result 3 of Problem (b)

Result 4:

The parameter setting for this result is:

STOCHASTIC = True

BATCH = True



```
The problem b of Simulation Project
Simulation Project STOCHASTIC is True
Simulation Project BATCH is True
The average optimized objective function value is 0.6356680767020582
The 95% confidence interval is [ 0.6095972905843087 , 0.6617388628198076 ]
```

Figure 5.2.4 Result 4 of Problem (b)

5.3 Result of Problem (c)

In our simulation, the data for company market values (X_i) and their potential returns (Y_i) were generated in two phases. In the first half of the iterations, X_i were drawn from a normal distribution with mean of (2, 3, 1) and standard deviations of (1, 1.7, 1.2). In this second half, $X_i$were drawn from a different normal distribution, with mean of (2.5, 4, 0) and standard deviations of (1, 1, 1). Y_i were uniformly distributed on [0, X_i].

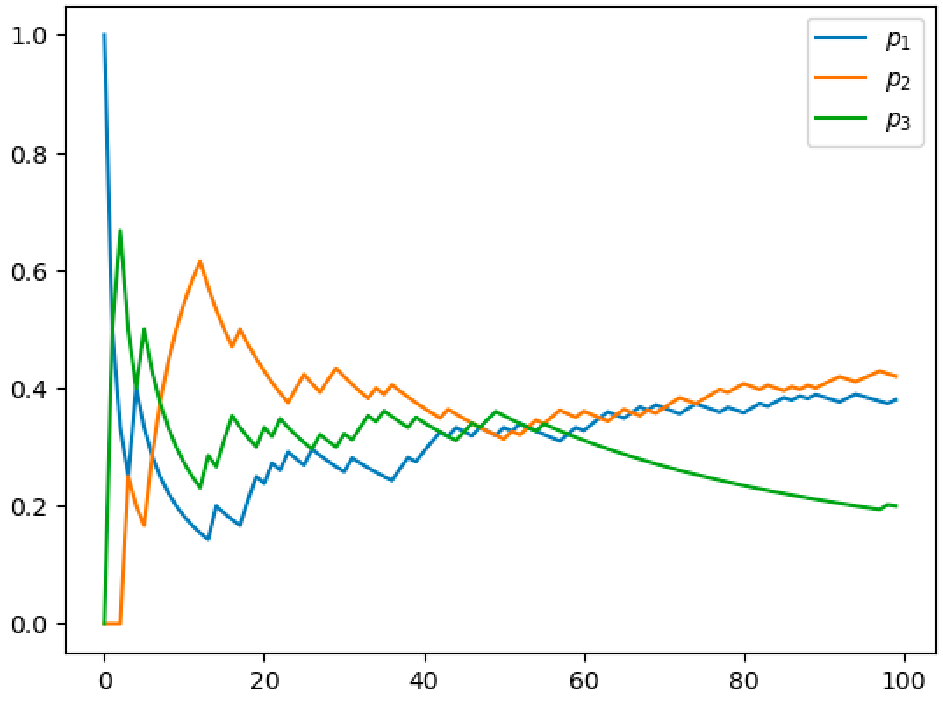


Figure 5.3.1 Result of Problem (c)

The output figure shows that after the initial iteration, the allocation for p_1 was 1, and p_2 and $p_3$were both initialized at 0. Then, the values of all three fractions fluctuated considerably and converged to certain levels respectively when the number of iterations approached the halfway mark of 50, which could be expected because the distribution of X_i in first half is similar. After it, the change in the distributions of X_i and Y_i caused another phase of value fluctuations, with p_1 and p_2 gradually increasing by a similar magnitude, and p_3 gradually decreasing, which is because the mean of X_3 is even smaller than threshold in second half.

This fluctuation and the change in trend after the 50th iteration show the adaptive nature of our algorithm. The change in the distributions of X_i and Y_i directly influenced the allocations, reinforcing the importance of responsive investment strategies in dynamic market scenarios.

6. Interpretation of Our Findings

SPSA based Python code is used to solve the investment problem where the total capital of 1 is allocated to n companies over a time period of *t* units. The objective of the investment problem is to maximize profit, and the simulation results show that in all three cases, the 3rd company has the highest possible return and can exceed the given value x_i. Therefore, investing in the 3rd company is the most profitable investment choice and the whole investment amount can be assigned to this company. Meanwhile, according to the iterative experiment, the 1st company and the 2nd company are less profitable and would be assigned null investment amounts.

In problem (a) and (b), we tested the results under different control conditions. Four images were generated based on the presence of STOCHASTIC and BATCH for each problem, all of which ultimately produced the same allocation results.

Random noise under STOCHASTIC is used to is to simulate random external factors that may affect the return on investment, bringing the algorithm closer to unpredictable market conditions. In addition, random noise can be used to test the robustness and sensitivity of an investment strategy to different market conditions, to test the ability to handle different scenarios, and to adjust the strategy accordingly. In the presence of STOCHASTIC, BATCH enables the program to get closer to the real situation and better predict possible shifting factors in dynamic environments in order to determine the most effective investment strategy.

The program solves the source error through the control of STOTIMATICS and BATCH, but still has some limitations and room for improvement. The code of the program has not yet been reasonably optimized due to the limited time. The program is time-consuming because the multiplication of iteration number, replication number, sample size and batch size is so large, which could be improved further.

For problem (c), our algorithm attempts to learn from the streaming data, respond to changes in the data stream, and update the allocation vector based on observed X_i and Y_i. How to update the allocation as new data comes in is a challenge. In our algorithm, we track the cumulative average and make allocation decisions based on it. However, such method implies that older observations impact the result as much as newer ones, which might not be an optimal solution in real-world scenarios. Giving more weight to more recent data may be beneficial.

7. Conclusion

To summarize, in problem (a) and (b), the simulation results have consistently identified the 3rd company as the most profitable option, demonstrating the effectiveness of utilizing the SPSA-based Python code to maximize profits in investment problems. The addition of random noise and BATCH under STOCHASTIC has enhanced the simulation's ability to model unpredictable market conditions and test the robustness of investment strategies. In problem (c), we simulated the dynamics of the market and designed a dynamic adaptive algorithm using both historical and current data to maximize expected returns through iterative updating.

Although there are some limitations and opportunities for improvement, the results provide valuable insights into investment allocation and highlight the potential of using the SPSA-based

Python code. Moreover, this project has showcased the capability of our solution to identify the optimal investment strategy utilizing a Stochastic Approximation (SA) algorithm. By conducting rigorous testing using manually created data, we have validated the algorithm’s effectiveness in adapting to continuous data streaming and adjusting predictions accordingly.

Reference

Abdulsadda, A. T., & Iqbal, K. (2011). An improved SPSA algorithm for system identification using fuzzy rules for training neural networks. International Journal of Automation and Computing, 8(3), 333–339. https://doi.org/10.1007/s11633-011-0589-x

H. Moghaddam, M.H. Moghaddam, M. Esfandyari, Stock market index prediction using artificial neural network, Journal of Economics, Finance and Administrative Science 21 (21) (2016) 89–93.

J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control, vol. 37, no. 3, pp. 332341, 1992.

Acknowledgment

Throughout this report, we have presented the collaborative journey undertaken by our group to address the investment problem, emphasizing the practical application of our solution in real-world scenarios. In conclusion, our holds significant potential to empower investors in navigating the complexities of the investment landscape and optimizing their financial outcomes. We would like to express our sincere gratitude to our lecturer Prof Bernd Heidergott, tutor Franssen Christian, and teaching assistant Zheng Yi for their guidance and support during the whole course period.