POLITECNICO DI TORINO

Master's Degree
in Mathematical Engineering

Master's Degree thesis

A survey of scenario generation methods for asset-liability management

Supervisor
prof. Paolo Brandimarte

Candidate
Linda Terzi

March 2023

Alla mia famiglia

Summary

Asset-liability management (ALM) attempts to find the optimal investment strategy under uncertainty in both the asset and liability streams. The ALM problem can be of interest to both private investors and banks or insurance companies. In fact, while a private investor may need to consider also the liabilities, as he may wish to plan his personal investments accounting for future consumption decisions that he has already planned; on the other hand an insurance company or a bank must take liabilities into consideration as they have payments to customers and employees that must be satisfied.

In order to obtain reliable results in this portfolio optimization problem, it's important to focus on the scenario generation. A good scenario generation method allows the investor, not only to trust the output, but also to have a saving in terms of calculation power.

In the present work we study the impact of different scenario generation methods on the results of ALM problem in stochastic programming. Firstly we investigate the best ALM problem formulation for the survey by studying the sensibility of a more basic formulation, with different assumptions on the asset returns distribution and only one deterministic liability, and a more complete formulation in which we add the transaction costs, we consider the asset prices, random liabilities and more decision variables. Finally we focus on the comparison between the results obtained with different scenario tree structures and different scenario generation methods: Monte Carlo, Antithetic Sampling, Control Variates and Quasi Monte Carlo with low discrepancy sequences. The comparison is made, besides looking at the quality and stability of the solution, also solving the problem for many periods in order to choose as better scenario generation method the one that allows the investor to cover a long time period with his current investment strategy decision.

Acknowledgements

Per la stesura dell'elaborato di tesi ringrazio il Professore Paolo Brandimarte, con i suoi corsi mi ha fatto appassionare all'ambito della ricerca operativa e ha saputo trasmettere nelle sue lezioni l'aspetto più applicativo degli argomenti trattati, senza mai trascurarne il lato teorico e accademico.
Questo elaborato di tesi segna la chiusura di un capitolo molto importante nella mia vita, quello universitario al Politecnico di Torino. Il mio percorso, come tutti i percorsi, non è stato sempre lineare e a volte sembrava davvero essere infito, poi, come tutti i percorsi, appena si conclude, sembra sia durato un attimo.
In primis voglio ringraziare la mia famiglia per essermi sempre stata vicina, un particolare grazie va ai miei genitori Monia e Valerio, per avermi consigliata e supportata in ogni mia scelta ed essere sempre stati un esempio per me.
Grazie anche alla mia famiglia torinese (ma reggiana), l'Asse Trapani-Beaulard, per essere sempre stati a condividere con me gioie e dolori del Poli, ma soprattutto per le cene di Natale, le tigellate e tutte le cene pazzesche insieme; senza di voi Torino per me non sarebbe stata casa.
Un grande grazie per tutte le gite, le serate e i momenti di leggerezza a tutti i miei amici dell'atletica emiliano-romagnola. Al gruppo marcia di Torino, nonostante le sveglie improponibili della domenica dei primi anni, grazie per aver cercato di farmi sembrare un'atleta anche a Torino. Grazie Noemi e Gioia per esserci sempre state. Grazie Michele per avermi supportata (sempre a modo tuo) in questa magistrale e per tutte le cene in videochiamata solo per fare due chiacchiere.
Infine un enorme grazie a Robbi, per la pazienza, il supporto e per essere stato la mia componente razionale nei momenti di sconforto e stanchezza in questi cinque anni; grazie per aver creduto in questo percorso come se fosse stato il tuo, sei il miglior teammate che avrei potuto desiderare.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Everyone has some plans for his future, from individual to multinational company, from a family-run business to a country, every reality has a future to face and one of the main objectives is to live it at its best.

In planning the future one of the most important aspects is the financial one. Every plans has costs and in order to be successful it is crucial to face them adequately.

The Asset-Liability Management (ALM) problem aims to find the best asset allocation to implement in the present in order to be able to face future liabilities. Therefore this is the problem that may help us, as private investors or as company, to manage our present wealth in order to bear the costs in a time horizon even long. Since we can't know the exact time horizon we need, it is important to solve the problem considering a time period of few years. At the horizon we require to have paid all the liabilities that we have encountered in the meantime and still have enough wealth to face other years like those simulated in the ALM problem solved. A solution that satisfies both, is considered a good ones.

Unfortunately the ALM problem deals with a lot of uncertainty, so we have to make some assumptions in order to solve it. In the present thesis we assume to know the asset returns distribution and how asset prices move, in order to generate scenarios for the resolution of the problem. We also suppose that we can exactly model the risk attitude and the satisfaction of the investor with an utility function.

In particular we focus on the impact of different scenario generation methods on the solution, in fact a good scenarios generation allows the investor to trust the solution with a reasonable computational cost.

The goal of the thesis is to make a survey on the impact of different scenario generation methods in the ALM problem solution in stochastic programming. The methods studied are: Monte Carlo, Antithetic Sampling, Control Variates and Quasi Monte Carlo with low discrepancy sequences. For all the methods we investigate also the solution sensitivity of different shapes of the scenario tree underlying the methods.

The thesis is structured as follow: in Chapter 2 we present the problem from a generic point of view and its two main mathematical formulations developed in the thesis with the assumptions made; in Chapter 3 we recall the theory of the stochastic programming; in Chapter 4 we present from a theoretical perspective the scenario generation methods

developed and investigated; in Chapter 5 we solve and comment a numerical example; finally in Chapter 6 we present the conclusions of the study and some possible implementation and future works.

Chapter 2

Asset liability modeling

For private and institutional investors, a joint characterisation of risk factors affecting both assets and liabilities is needed. The control of interest rate and liquidity risks is the traditional purpose of Asset Liability Management (ALM). Asset-liability management (ALM) refers to buying and selling securities (assets) to meet current and future payments (liabilities) under uncertainty associated usually, but not exclusively, with interest-rate movements. It is practiced for pension-fund management and in the banking and the insurance industries primarily. [1] There are different sets of decision variables that may be used in portfolio optimization. In simple models, the asset weights are considered, i.e., the fraction of wealth that we allocate to each asset. However, in an asset–liability model there is a need to generate cash flows matching liabilities, which leads to a different set of decision variables, such as how many units of each asset we hold, i.e., the actual number of stock shares or bonds. [2]

Based upon the time horizon over which the asset-liability optimization decision is to be modeled and the conditions under which it is to be modeled, we can categorize the ALM models into four basic categories:

1. Single-period static models.

2. Single-period stochastic models.

3. Multi-period static models.

4. Multi-period stochastic models.

These models represent an extension of risk measures and ALM goals from single period settings to multi-period settings. We can summarize the models cited in Figure 2.1.

In this work we deal with multi-period stochastic ALM model and we use stochastic programming, that describes uncertainty by discretizing time. Moreover, the tree structure of stochastic programming is quite helpful in modelling uncertainty. [3]

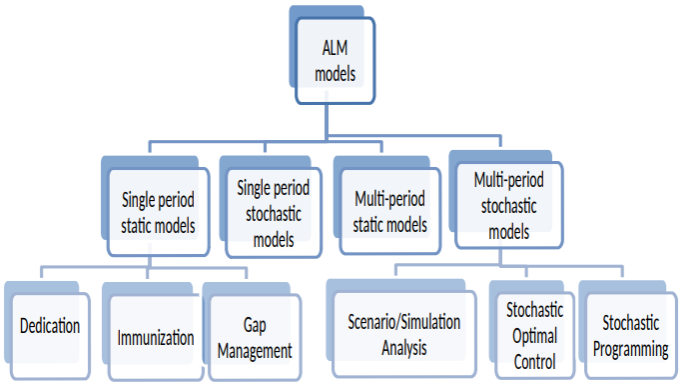

Figure 2.1: ALM model classification. [3]

2.1 General problem formulation

We suppose to optimize a portfolio with n risky assets. The goal is to maximize the expected utility of terminal wealth, we consider the risk aversion by taking a concave utility function. Time is discrete and the rebalancing is made in the set of times $\{0, 1, ..., T-1\}$ where T is the time horizon. We don't allow short sales and as we said before the decision variables of this simple problem are the asset weights.

By defining :

- x : asset weights,
- u : utility function,
- w_T : terminal wealth;

we can write a naive form of the optimization problem faced by the investor as

$$max_{x \geq 0} E[u(w_T)].$$

2.2 Utility function

In economics, utility represents the satisfaction or pleasure that consumers receive for consuming a good or service. Utility function aims to measure consumers' preferences for a set of goods and services and it is widely used in rational choice theory to analyze human behavior. [4]

In particular we refer to the Von Neumann–Morgenstern utility function. Its definition comes from the homonymous utility theorem in the field of decision theory. The Von Neumann–Morgenstern (VNM) utility theorem shows that, under certain axioms of rational behavior, a decision-maker faced with risky (probabilistic) outcomes of different choices will behave as if he or she is maximizing the expected value of some function defined over

the potential outcomes at some specified point in the future. This function is known as the Von Neumann–Morgenstern utility function.

In this work the previously defined utility function is named $u(\cdot)$ and its argument named w represents the amount of wealth.

The Von Neumann–Morgenstern expected utility is a particularly simple form of expected utility functional, as it maps random variables to the real line, but it is only justified by specific hypotheses on the preference relationship that it represents and it is defined as

$$U(w) = E[u(w)]. \tag{2.1}$$

A Von Neumann–Morgenstern utility function $u(w)$ represents an investor's attitude towards risk, where the risk behavior is determined through the absolute and relative risk aversion coefficients, define as

$$R_u^a(w) = -\frac{u''(w)}{u'(w)}, \quad R_u^r(w) = -w\frac{u''(w)}{u'(w)}. \tag{2.2}$$

The more concave the utility function, i.e., the larger $u''(w)$ in absolute value, the larger the risk aversion.

Definition 1 *A function $f(\cdot)$ is* concave *if the function $-f(\cdot)$ is convex.*

Therefore, it is not the absolute value of the function but how strongly it bends that determines differences in investor behaviour. A linear utility function representing risk-neutral behavior would exhibit an absolute risk aversion coefficient of zero. Concave utility functions, representing risk-averse behavior, exhibit positive absolute risk aversion. When modeling investor choice, we are interested in how the risk aversion changes with wealth. Moreover for financial problems, it is reasonable to assume that utility $u(\cdot)$ is a strictly increasing function, since we prefer more wealth to less. Formally, this property is referred to as non-satiation.

The coefficients of absolute and relative aversion may be decreasing, constant, or increasing with respect to their argument. Hence, utility functions may belong to one of the following families:

- Decreasing, constant, increasing absolute risk aversion, denoted by DARA, CARA, and IARA, respectively;
- Decreasing, constant, increasing relative risk aversion, denoted by DRRA, CRRA, and IRRA, respectively.

Some typical utility functions are:

- logarithmic utility: $u(w) = log(w)$,
- power utility: $u(w) = -\frac{w^{1-\gamma}-1}{1-\gamma}$, $\gamma > 1$,
- quadratic utility: $u(w) = w - \frac{\lambda}{2}w^2$,
- exponential utility: $u(w) = -e^{-\alpha w}$, $\alpha > 0$;

in particular we can notice that the logarithmic utility is a limit case of the family of power utility functions.

The properties of the most common utility functions just mentioned are summarized in the table below:

	$R_u^a(w)$	$R_u^r(w)$	**Properties**
$log(w)$	$\frac{1}{w}$	1	DARA, CRRA
$-e^{-\alpha w}$	α	αw	CARA
$w - \frac{\lambda}{2}w^2$	$\frac{\lambda}{1-\lambda w}$	$\frac{\lambda w}{1-\lambda w}$	IARA

Table 2.1: Properties of most common utility functions.

The last line in the Table 2.1 implies, for instance, that an investor becomes more risk-averse if his wealth increases, which is usually considered at odds with standard investors' behavior. [2][5]

In the next sections we outline in details the utility functions chosen for basic and extended formulations of the problem.

2.3 Basic formulation

The first formulation of the ALM problem, that we call *basic formulation*, considers stochastic returns r of a set $\mathcal{I}$ of assets and a unique and deterministic liability L. The utility function $u(\cdot)$ considered faces zero when the wealth w is equal to the liability.
In the problem we consider the multiplicative gain $R = r + 1$ instead of the returns for each asset directly.

The problem develops in a time horizon from $t = 0$ to $t = T$, where T is the time at which we have to pay the liability while $t = 0$ represents the present, so the moment when we have to allocate the initial wealth in the assets. Moreover in any time instant we have to sample the realization of the asset returns with a multistage scenario tree. The node sequences that describe the paths from the root node to each leaf nodes of the tree represent scenarios of the problem. We distinguish the nodes of the tree in three subsets: root node n_0, intermediate nodes $\mathcal{T}$ and leaf nodes $\mathcal{S}$. The entire set of nodes $\mathcal{N}$ is the union without overlapping of the three subset just mentioned.
We assume that in the leaf nodes no allocation is needed because at time T we only pay the liability.

2.3.1 Problem formulation

The decision variable x is the fraction of wealth that is allocated to each asset, and we assume to have an initial wealth equal to W_0 that we want allocate entirely; so the constraint on the initial wealth is formulated as

$$\sum_{i \in \mathcal{I}} x_{i,n_0} = W_0. \tag{2.3}$$

Rebalancing is allowed, and necessary, only in the intermediate nodes

$$\sum_{i\in\mathcal{I}} R_{i,n}x_{i,a(n)} = \sum_{i\in\mathcal{I}} x_{i,n} \quad \forall n \in \mathcal{T}, \tag{2.4}$$

where $a(n)$ is the antecedent node of node n.
As we said before, in leaf nodes we have to meet the liability and to measure the residual wealth, so the corresponding constraint is

$$\sum_{i\in\mathcal{I}} R_{i,s}x_{i,a(s)} - L = w^s \quad \forall s \in \mathcal{S}, \tag{2.5}$$

where w^s is the terminal wealth in scenario s. Underlying that the goal of the problem is to maximize the terminal expected utility with the certainty to meet the liability, we assign to each scenario s a probability p_s that is the probability conditioned to the node sequence that determines s. Therefore the objective function results

$$\sum_{s\in S} p_s w^s. \tag{2.6}$$

Finally we can write the optimization problem:

$$\begin{aligned} \max \quad & (2.6) \\ s.t. \quad & (2.3), \\ & (2.4), \\ & (2.5) \\ & x_{i,n}, w^s \geq 0. \end{aligned} \tag{2.7}$$

[2]

2.3.2 Variants of the basic problem

In this work we present three variants of the basic problem.

First variant

The first variant presented leads to a myopic formulation.
A piecewise linear utility function is considered where the utility faces zero as the liability is met, underlying the need to satisfy the payment. We have a slope q multiplying the positive wealth w_+ and a slope r multiplying the negative wealth w_-, as we can see in Figure 2.2 taken from [2]. Therefore the utility in the object function is:

$$u(w) = \begin{cases} qw & w \geq 0 \\ rw & w < 0 \end{cases} \tag{2.8}$$

in order to be able to write it in the optimization model we can refer to the formulation

$$u(w) = qw_+ - rw_-. \tag{2.9}$$

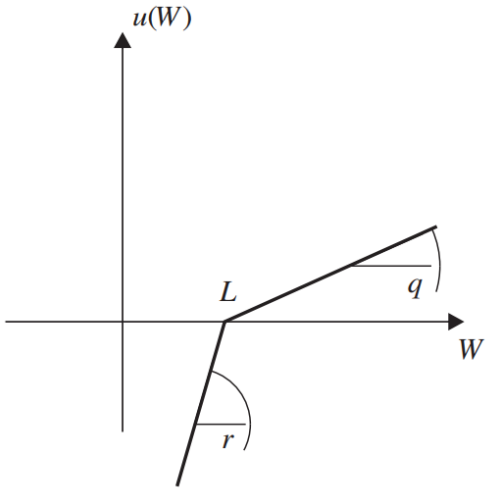

Figure 2.2: A piecewise linear utility function. [2]

To better catch risk aversion of our user we take $q < r$. The function is concave and leads a Linear Programming problem, but is also a myopic problem because it might imply "local" risk neutrality. [2]

In this case we consider the asset returns distributed according to a normal multivariate with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. By considering the matrix of asset returns $\mathbf{r}$ we have

$$\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.10}$$

where $\mathcal{N}$ represents the normal distribution.

Second variant

In the second variant the investor has a quadratic utility function:

$$u(w) = w - bw^2, \quad b > 0, \tag{2.11}$$

which implies that absolute risk aversion is increasing. It seems reasonable because an investor that gains a lot of wealth might be more afraid to loose it.

The asset returns are assumed to be normally distributed as in the first variant presented.

Third variant

The third variant considers asset returns distributed according to a log-normal distribution instead of a normal one and a power utility function.

Definition 2 *A positive random variable x is* log-normally distributed *(i.e. $X \sim Lognormal(\mu_x, \sigma_x^2)$) if the natural logarithm of X is normally distributed with mean μ and variance σ^2:*

$$\ln(X) \sim \mathcal{N}(\mu, \sigma^2).$$

In particular we consider the *multivariate log-normal* distribution for the asset returns, consistently with the choice to consider a multivariate normal distribution in the first variant of the problem.
We name $\overline{\mathbf{r}}$ the returns log-normally distributed.
Previously we supposed that $\mathbf{r} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, then $\overline{r}_i = \exp(r_i)$ has a multivariate log-normal distribution; given that the exponential is applied elementwise to the random vector $\mathbf{r}$, the mean of $\overline{\mathbf{r}}$ is

$$E[\overline{\mathbf{r}}]_i = e^{\mu_i + \frac{1}{2}\Sigma_{ii}}, \tag{2.12}$$

and the variance is

$$Var[\overline{\mathbf{r}}]_{ij} = e^{\mu_i + \mu_j + \frac{1}{2}(\Sigma_{ii} + \Sigma_{jj})}(e^{\Sigma_{ij}} - 1). \tag{2.13}$$

The relation between normal and log-normal distributions is well summarized in Figure 2.3 (imagine By StijnDeVuyst - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53714670) where X represents our r and Y represents $\overline{r}$.

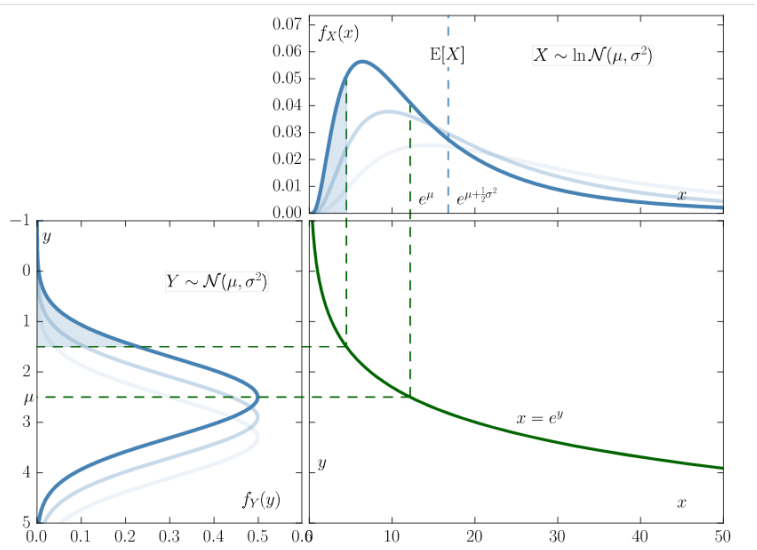

Figure 2.3: Relation between normal and log-normal distribution.

Moreover the power utility function considered is:

$$u(w) = \frac{w^{1-\gamma} - 1}{1 - \gamma}, \tag{2.14}$$

in this situation we are interest in meeting the liability so we don't care to have less utility as the wealth increase so the choice seems reasonable.

2.4 Extended formulation

The second formulation of ALM problem proposed in this work, which we refer to as *extended formulation*, is a generalization of the *basic formulation* of the problem presented

before. In this case we consider the asset prices P instead of their returns and we introduce the transition costs. Liabilities now are assumed stochastic and we have a liability L^n for all nodes $n \in \mathcal{N}$ in the scenario tree.

As we did in the previous section we index the assets by $i \in \mathcal{I}$; we consider the time instants set $t = 0, \dots, T$ in which our problem develops; we use the same scenario tree structure to capture the stochastic nature of the problem and we assume that each scenario $s \in \mathcal{S}$ has probability π^s.

2.4.1 Problem formulation

In the present problem formulation we don't consider the amount of wealth allocated to each asset but we want to have more details by considering the amount of wealth owned after the rebalancing, the amount of wealth sold and the amount bought for each asset. We define respectively:

- x_i^n : amount of asset i owned at node n, after the rebalancing;
- y_i^n : amount of asset i sold at node n;
- z_i^n : amount of asset i purchased at node n;

These decision variables are defined only on trading nodes $n \in \mathcal{N} \backslash \mathcal{S}$, as we assume that the rebalancing is not allowed in the terminal time instant.
Moreover we define W^s the wealth at the terminal node s.
Another difference with respect to the *basic formulation* is the need to have an initial amount for each asset $i \in \mathcal{I}$ at the root node n_0, that we call $h_i^{n_0}$.

On the base of these definitions and on the concepts explained in the previous sections, we can define the structure of the optimization problem as follow:

$$\max \quad \sum_{s \in \mathcal{S}} \pi^s u(W^s) \tag{2.15}$$

$$s.t. \quad x_i^{n_0} = h_i^{n_0} + z_i^{n_0} - y_i^{n_0}, \qquad \forall i \in \mathcal{I} \tag{2.16}$$

$$x_i^n = x_i^{a(n)} + z_i^n - y_i^n, \qquad \forall i \in \mathcal{I}, \forall n \in \mathcal{T} \tag{2.17}$$

$$(1-c) \sum_{i \in \mathcal{I}} P_i^n y_i^n - (1+c) \sum_{i \in I} P_i^n z_i^n = L^n, \qquad \forall n \in \mathcal{N} \setminus \mathcal{S} \tag{2.18}$$

$$W^s = (1-c) \sum_{i \in \mathcal{I}} P_i^s x_i^{a(s)} - L^s, \qquad \forall s \in \mathcal{S} \tag{2.19}$$

$$x_i^n, y_i^n, z_i^n, W^s \geq 0. \tag{2.20}$$

Where $u(\cdot)$ is a nonlinear utility function.

It is important to notice that we are interest in the wealth allocation in the root node, as the role of terminal utility is just to ensure that we are left in a "good position" at the end of the planning horizon, in order to avoid nasty end-of-horizon effects. [2]

2.4.2 Variants of the implemented problem

For the *extended formulation* of the ALM problem we assume asset prices follow a geometric Brownian motion and we simulate the sample paths using Monte Carlo methods.

In the different versions of the problem thus formulated we change only utility function.

First variant

In order to make a comparison with the results obtained with the *basic formulation* and in order to see the impact of transaction costs, in the first variant we solve the *extended formulation* of the problem by taking the piecewise linear utility function 2.9.

Second variant

In the second variant we consider the he quadratic utility function, that is the most frequently used in the financial economics to describe investor behaviour:

$$u(w) = w - bw^2, \quad b > 0. \tag{2.21}$$

As shown in section 2.2, the absolute risk aversion function demonstrates that the quadratic utility function exhibits increasing absolute risk aversion. Thus, the quadratic function is consistent with investors who reduce the nominal amount invested in riskier assets as their wealth increases. By definition, a quadratic utility function must exhibit increasing relative risk aversion. [6]

Chapter 3

Stochastic programming

Stochastic programming is a good modelling paradigm for asset and liability management problems. It incorporates multiple correlated sources of risk for both the asset and liability side, considers a long time horizon perspective and allows for dynamic portfolio rebalancing.

Asset and liability management problem deals with uncertainty. A general approach is to assign to the unknown parameters a probability distribution. Mathematical programming models for dealing with uncertainty are known as *stochastic programs*. [5]

3.1 Stochastic optimization problem

We introduce stochastic optimization by considering a two-stage problem. In the first stage a decision is made not knowing what will happen in the future, but knowing the probability of different events. After the first stage decision, the random event occurs and uncertainty is resolved. Then, in the second stage, further decisions are made that can depend on what has happened. This is often called a two-stage stochastic problem with recourse. The word 'recourse' here refers to what needs to be done at the second stage as a result of the random event. [7]

3.2 General formulation

The most general formulation of stochastic programs is the recourse model, which is the one usually used for financial applications, an it is the combination of the anticipative and the adaptive models. [5]

3.2.1 Anticipative model

Suppose that a decision x must be made in an uncertain world where the uncertainty source is represents by the random vector $\boldsymbol{\omega}$.

In anticipative models feasibility is expressed in terms of probabilistic (or chance) constraints. A reliability level α, where $0 < \alpha < 1$ is specified and constraints are expressed in the form

$$P\{\boldsymbol{\omega}|f_j(x,\boldsymbol{\omega}) = 0, \quad j = 0,\dots,n\} \geq \alpha, \tag{3.1}$$

An *anticipative model* selects a strategy that leads to some desirable characteristics of the constraint and objective functionals under the realizations of the random vector. [5]

3.2.2 Adaptive model

In an adaptive model observations related to the uncertainty source are known before the decision x is made. It is clear that observations provide only partial information about the random variables because otherwise the model would simply wait to observe the values of the random variables and the decision x would be the solution of a deterministic mathematical program. In contrast to this situation we have the other extreme where all observations are made after the decision x has been made, and the model becomes anticipative.

In this case the decisions x depend only on the events that could be observed so x is termed $\mathcal{A}$-*adapted* or $\mathcal{A}$-*measurable*, where $\mathcal{A}$ be the collection of all the relevant information that could become available by making an observation. [5]

3.2.3 Recourse model

The recourse problem combines the anticipative and adaptive models in a common mathematical framework. The problem seeks a policy that not only anticipates future observations but also takes into account that observations are made about uncertainty, and thus can adapt by taking recourse decisions. In our case, the asset allocation is decided considering both future movements of asset prices (anticipation) and that the allocation will be rebalanced as prices change (adaptation). [5]

We can construct the stochastic programming formulation by starting from the standard form of the LP model,

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{aligned} \tag{3.2}$$

then we assume stochastic data and obtain the following stochastic model:

$$\begin{aligned} \min \quad & \mathbf{c}(\omega)^{\mathrm{T}}\mathbf{x} \\ s.t. \quad & \mathbf{A}(\omega)\mathbf{x} = \mathbf{b}(\omega), \quad a.s. \\ & \mathbf{x} \geq 0. \end{aligned} \tag{3.3}$$

Clearly, this is not feasible and the constraints must be relaxed. For example we present the case of inequality constraints:

$$\mathbf{A}(\omega)\mathbf{x} \geq \mathbf{b}(\omega), a.s., \tag{3.4}$$

that leads to a feasible solution.
In order to restore feasibility we introduce decisions that are made after realization of uncertainty: *recourse* actions.

We now formalize a two-stage LP model with recourse as:

- The first-stage decision $\mathbf{x} \geq 0$ must satisfy immediate constraints $\mathbf{Ax} = \mathbf{b}$ and incur an immediate (first-stage) cost $\mathbf{c}^{\mathrm{T}}\mathbf{x}$.
- At the second stage, a random scenario ω occurs, associated with random data. Given this information, a second-stage decision (recourse action) $\mathbf{y}(\omega) \geq 0$ is made.
- The second-stage decisions are related to the first-stage decision by interstage constraints $\mathbf{Wy}(\omega) + \mathbf{T}(\omega)\mathbf{x} = \mathbf{h}(\omega)$.
- The second stage decision incurs a cost $\mathbf{q}(\omega)^{\mathrm{T}}\mathbf{y}(\omega)$, which is random from the viewpoint of the root node.
- The overall objective is to minimize the sum of the first-stage cost and the expected value of second-stage cost.

Then we obtain the optimization model:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} + E_{\omega}[\mathbf{q}(\omega)^{\mathrm{T}}\mathbf{y}(\omega)] \\ s.t \quad & \mathbf{Ax} = \mathbf{b}, \\ & \mathbf{Wy}(\omega) + \mathbf{T}(\omega)\mathbf{x} = \mathbf{h}(\omega), \quad a.s. \\ & \mathbf{x}, \mathbf{y}(\omega) \geq 0. \end{aligned}$$

When the recourse matrix $\mathbf{W}$ is deterministic, as above, we have a fixed recourse problem. The more general case with a random recourse matrix $\mathbf{W}(\omega)$ may present additional difficulties.

We now introduce a recourse function $\mathcal{Q}(x)$ and we rewrite the model in a deterministic formulation:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathcal{Q}(x) \\ s.t. \quad & \mathbf{Ax} = \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{aligned} \tag{3.5}$$

where the recourse function is defined as follow:

$$\mathcal{Q}(x) = E_{\xi}[Q(\mathbf{x}, \boldsymbol{\xi}(\omega))], \tag{3.6}$$

and

$$Q(\mathbf{x}, \boldsymbol{\xi}(\omega)) = \min_{y}\{\mathbf{q}(\boldsymbol{\xi}(\omega))^{\mathrm{T}}\mathbf{y} | \mathbf{Wy} = \mathbf{h}(\boldsymbol{\xi}(\omega)) - \mathbf{T}(\boldsymbol{\xi}(\omega))\mathbf{x}, \mathbf{y} \geq 0\}. \tag{3.7}$$

This formulation shows that stochastic linear programming is, in general, a nonlinear programming problem.

For computational purposes, we may represent uncertainty by a discrete probability distribution, resulting in a scenario tree, where ω_s is the outcome corresponding to scenario s. Each scenario $s \in \mathcal{S}$ is associated with a probability π_s and a set of scenario-dependent random data, as shown in the following LP:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} + \sum_{s\in\mathcal{S}} \pi_s \mathbf{q}_s^{\mathrm{T}}\mathbf{y}_s \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{W}\mathbf{y}_s + \mathbf{T}_s\mathbf{x} = \mathbf{h}_s, \quad \forall s \in \mathcal{S} \\ & \mathbf{x}, \mathbf{y}_s \geq 0. \end{aligned} \tag{3.8}$$

This is a plain LP. [2]

3.2.4 Multistage model

The recourse problem is not restricted to the two-stage formulation. It is possible that observations are made at T different stages and are captured in the information sets $\{\mathcal{A}_t\}_{t=1}^{T}$ with $\mathcal{A}_1 \subset \mathcal{A}_2 \subset \cdots \subset \mathcal{A}_T$. Stages correspond to time instances when some information is revealed and a decision can be made.

A multistage stochastic program with recourse will have a recourse problem at stage τ conditioned on the information provided by $\mathcal{A}_\tau$, which includes all information provided by the information sets $\mathcal{A}_t$, for $t = 1, 2, ..., \tau$. The program also anticipates the information in $\mathcal{A}_t$, for $t = \tau + 1, \ldots, T$.

The multistage program, which extends the two-stage model, is formulated as a nested optimization problem. [5]

3.3 Scenario tree

The key ingredient in stochastic programming models is the scenario tree, and the quality of the solution depends critically on how well uncertainty is captured. [2]

Traditional stochastic programming uses scenario trees (Figure 3.1 as an example) to represent possible future events. Thus, the emphasis is on obtaining a good first-stage solution rather than obtaining an entire accurate policy. [5]

The root node in the scenario tree represents the present and is immediately observable from available information. The nodes further down the tree stand for events that are conditional on outcomes at prior stages. The arcs linking the nodes represent various realizations of the uncertain variables. Ideally, the generated set of realizations constitutes the whole universe of possible outcomes of the random variable. [8]

The first decision to take is the shape of the scenario tree, i.e., the branching factor which is applied at each node. A typical approach is to have a larger branching factor at early stages, as representing uncertainty there accurately may be more important, an example is represented in Figure 3.2.

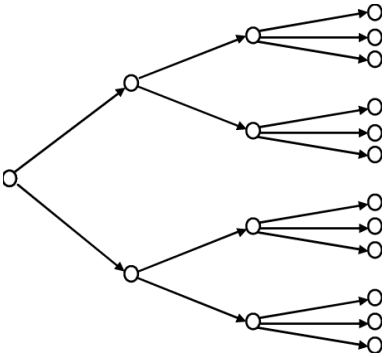

Figure 3.1: Example of a scenario tree.

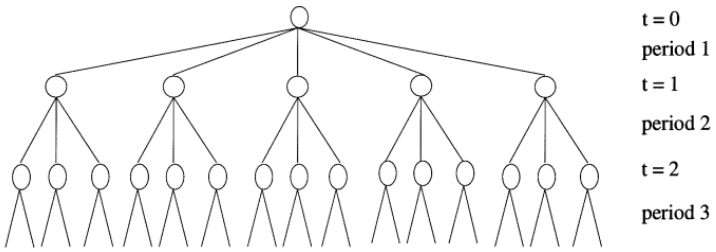

Figure 3.2: A scenario tree. [9]

Given a scenario tree structure, we have to decide which outcomes we should associate to nodes in the tree, and possibly the (conditional) probabilities associated to each branch in the tree.

The techniques used in this work for sampling for scenario tree generation are:

- Naive Monte Carlo sampling. In this case, the probability distribution for future nodes branching from current node is uniform.
- Antithetic sampling, in the case of symmetric distributions, leads to a sample that matches odd moments of the underlying density.
- Low-discrepancy sequences.

All these techniques will be discussed in the following chapter.

The considerations we have done so far apply to a generic stochastic program. When we deal with an application in finance, there is still another issue: arbitrage.
We have an arbitrage opportunity if there exists a portfolio which is guaranteed to have a non-negative value at the end of the holding period in any scenario, but which has a negative value at the beginning.
Sensible scenarios should not only reflect the information we have, but they should also rule out arbitrage opportunities. [10]

Chapter 4

Scenario generation

Stochastic ALM models are based on scenario generation. A scenario is a forecasted unique state realized by randomly changing variables of the model during the planned investment period. [11]

4.1 Sample path generation

Any variable whose value changes over time in an uncertain way is said to follow a stochastic process. Stochastic processes can be classified as discrete time or continuous time. A discrete-time stochastic process is one where the value of the variable can change only at certain fixed points in time, whereas a continuous-time stochastic process is one where changes can take place at any time. Stochastic processes can also be classified as continuous variable or discrete variable. In a continuous-variable process, the underlying variable can take any value within a certain range, whereas in a discrete variable process, only certain discrete values are possible.[12]

Many models of interest in financial engineering can be represented by continuous-time stochastic differential equations.
Therefore we introduce the sample path generation for the asset prices of the portfolio considered.

4.1.1 Wiener Process

The Wiener process is a particular type of Markov stochastic process with a mean change of zero and a variance rate of 1.0 per year; where a Markov process is a particular type of stochastic process where only the current value of a variable is relevant for predicting the future.
The Wiener process is sometimes referred to as Brownian motion.

Expressed formally, a variable z follows a Wiener process if it has the following two properties:

Property 3 *The change Δz during a small period of time Δt is*

$$\Delta z = \epsilon\sqrt{\Delta t}, \tag{4.1}$$

where ϵ has a standard normal distribution $\phi(0,1)$.

Property 4 *The values of Δz for any two different short intervals of time, Δt, are independent.*

It follows from the first property that Δt itself has a normal distribution with mean equals to zero and standard deviation equals to $\sqrt{\Delta t}$. [12]

We can formally define the Wiener process as follow. [13]

Definition 5 *A stochastic process Z is called a* Wiener process *if the following conditions hold.*

1. $Z_0 = 0$.
2. *The process W has independent increments, i.e. if $r < s \leq t < u$ then $Z_u - Z_t$ and $Z_s - Z_r$ are independent random variables.*
3. *For $s < t$ the random variable $Z_t - Z_s$ has Gaussian distribution $\mathcal{N}(0, t-s)$.*
4. *Z has continuous trajectories.*

Generalized Wiener Process

The mean change per unit time for a stochastic process is known as the *drift rate* and the variance per unit time is known as the *variance rate*.
A generalized Wiener process for a variable x can be defined in terms of dz as

$$dx = adt + bdz, \tag{4.2}$$

where a and b are constants.
Similarly to what we have seen for a Wiener process, the change in the value of x in any time interval T is normally distributed with mean equals to aT and a variance equals to b^2T (so a standard deviation equals to $b\sqrt{T}$).

To summarize, the generalized Wiener process given in equation (4.2) has an expected drift rate (i.e., average drift per unit of time) of a and a variance rate (i.e., variance per unit of time) of b^2. [12]

4.1.2 Itô Process

A further type of stochastic process, known as an Itô process, can be defined. This is a generalized Wiener process in which the parameters a and b are functions of the value of the underlying variable x and time t.

An Itô process can therefore be written as

$$dx = a(x,t)dt + b(x,t)dz. \tag{4.3}$$

Both the expected drift rate and variance rate of an Itô process are liable to change over time. [12]

4.1.3 Stock Price

If S is the stock price at time t, then the expected drift rate in S should be assumed to be μS for some constant parameter μ. This means that in a short interval of time, Δt, the expected increase in S is $\mu S \Delta t$. The parameter μ is the expected rate of return on the stock.

A reasonable assumption is that the variability of the return in a short period of time, Δt, is the same regardless of the stock price. This suggests that the standard deviation of the change in a short period of time Δt should be proportional to the stock price and leads to the model

$$dS = \mu S dt + \sigma S dz, \tag{4.4}$$

where the variable μ is the stock's expected rate of return and the variable σ is the volatility of the stock price.

The model in equation (4.4) is said Geometric Brownian Motion, that is the stochastic process assumed in this work for asset prices. Under this process the return to the holder of the stock in a small period of time is normally distributed and the returns in two non-overlapping periods are independent.[12]

Proposition 6 *The solution to the equation*

$$\begin{aligned} dS &= \alpha S dt + \sigma S dW, \\ S_0 &= s_0, \end{aligned} \tag{4.5}$$

is given by

$$S(t) = s_0 \cdot \exp\left\{\left(\alpha - \frac{\sigma^2}{2}\right)t + \sigma W(t)\right\}. \tag{4.6}$$

4.1.4 Itô's Lemma

Suppose that the value of a variable x follows the Itô process

$$dx = a(x,t)dt + b(x,t)dz, \tag{4.7}$$

where dz is a Wiener process and a and b are functions of x and t. The variable x has a drift rate of a and a variance rate of b^2. Itô's lemma shows that a function G of x and t follows the process:

$$dG = \left(\frac{\partial G}{\partial x}a + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial x^2}b^2\right)dt + \frac{\partial G}{\partial x}bdz, \tag{4.8}$$

where dz is the same Wiener process as in equation (4.7).

From the equation (4.4) for the process of the stock price, it follows that the process followed by a function G of S and t is

$$dG = \left(\frac{\partial G}{\partial S}\mu S + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial G}{\partial S}\sigma S dz. \tag{4.9}$$

A key point is that the Wiener process dz underlying the stochastic process for the variable is exactly the same as the Wiener process underlying the stochastic process for the function of the variable. Both are subject to the same underlying source of uncertainty. [12]

4.1.5 Black-Scholes-Merton Model

We cite the Black-Scholes-Merton differential equation for the sake of completeness but we doesn't derive it.

The Black–Scholes–Merton model is based on the geometric Brownian motion assumption and it is possible derive the following differential equation:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2} = rf. \tag{4.10}$$

It has many solutions, corresponding to all the different derivatives that can be defined with S as the underlying variable. [12]

4.2 Monte Carlo sampling

The traditional method to generate a scenario tree for ALM is through Monte Carlo sampling, with uniformly distributed pseudo-random numbers transformed appropriately into a target distribution.

As Monte Carlo is based on the volume of a set distribution for the definition of probability measure, an obvious way to deal with this problem is to increase the number of nodes in the randomly sampled event tree. However, the stochastic program might become computationally intractable due to the tree's exponential growth rate. This hypothesis is supported, not only by the law of large numbers that guarantees the convergence to a correct value as the number of draws increase, but also by the central limit theorem that offers information about the error magnitude after a finite number of simulations. [14]

4.2.1 Structure of a Monte Carlo simulation

We can think about Monte Carlo simulator as a function mapping a stream of pseudo-random numbers, i.e., a sequence of i.i.d. variables with uniform distribution on the unit interval (0,1), into a set of estimates of interest. Then, random numbers are transformed in order to create a sample from the probability distribution of interest.

We can represent a typical Monte Carlo simulation as a sequence of building blocks.

1. Generation of a stream of $\{U_i\}$ a sequence of i.i.d. variables uniformly distributed on the unit interval (0,1). These pseudo-random numbers are generated by an algorithm, that ensures us the characteristics of repeatability and efficiency.

2. Transformation of the stream of uniform random numbers into a sequence of random variates with the desired distribution, resulting in a stream $\{X_j\}$. We use a different subscript, j rather than i, to stress the fact that more than one random number might be needed to generate one realization of the random variate of interest.

3. Simulation model, we represent and use domain-specific dynamics in order to generate a sample path. This is highly problem dependent. In particular in this work we generate a sample path of asset class prices.

4. Collection of all outcomes in order to come up with an estimate.

The list above points out quite clearly that a typical Monte Carlo simulation is, at least conceptually, a function mapping a vector of n uniform random numbers into a numeric output. [15]

4.2.2 Pseudo-random numbers generators

A pseudo-random number generator is a tool to generate a stream of numbers on the unit interval. We briefly present some pseudo-random generators.

Linear congruential generators

The first method presented to generate $U(0,1)$ random numbers is the *linear congruential generator* (LCG). This approach was actually implemented in many generators included in commercial software, and it is still possible to choose an LCG. However, LCGs are not considered state-of-the-art anymore and should not be used for high-quality Monte Carlo experiments.

An LCG generates a sequence of non-negative integer numbers Z_i, which are transformed into uniform random numbers. The sequence of integer numbers starts from a user-selected seed Z_0; then, given the previous integer number i, the next number in the sequence is generated as follows:

$$Z_i = (aZ_{i-1} + c) \mod m, \tag{4.11}$$

where a (the multiplier), c (the shift), and m (the modulus) are properly chosen integer numbers and "mod" denotes the remainder of integer division. Finally, the integer number Z_i is transformed into a uniform number

$$U_i = \frac{Z_i}{m}. \tag{4.12}$$

The principal limitation of this method is its periodicity. In fact we can observe that we can generate at most m distinct integers numbers Z_i, and whenever we repeat a previously generated number, the sequence will be repeated as well.

By setting the increment $c = 0$ we obtain the *multiplicative* LCG, that is the base of alternative and more sophisticated generators.

In the simple LCG case, the states are just integer numbers and the output space is the unit interval. More recent generators use more complex state spaces.

- *Combined generators.* The idea is to combine multiple generators that, taken individually, would have quite some weaknesses, but improve a lot when combined. For example the output of a generator that combines three LCGs X_i, Y_i and Z_i is

$$U_i = \left(\frac{X_i}{m_1} + \frac{Y_i}{m_2} + \frac{Z_i}{m_3}\right) \mod 1. \tag{4.13}$$

- *Multiple recursive generators.* The idea is to use a richer state space $\mathcal{S}$. They have the form is:

$$X_i = (a_1 X_{i-1} + a_2 X_{i-2} + \cdots + a_k X_{i-k}) \mod m. \tag{4.14}$$

  With $m = 2$ we have the class of generators called *linear feedback shift registers* and their output function maps a sequence of bits into a number in the unit interval:

$$U_i = \sum_{j=1}^{L} X_{is+j-1} 2^{-j}, \tag{4.15}$$

  where s and L are positive integers representing the step size and word length, respectively.

[15]

Inverse transform method

Suppose that we are given the CDF $F_X(x) = P\{X \leq x\}$, and that we want to generate random variates distributed according to $F_X(\cdot)$. If we are able to invert $F_X(\cdot)$ easily, then we may apply the following inverse transform method:

1. Draw a random number $U \sim \mathcal{U}(0,1)$.
2. Return $X = F_X^{-1}(U)$.

It is easy to see that the random variate X generated by this method is actually characterized by the CDF $F_X(\cdot)$:

$$P\{X \leq x\} = P\{F_X^{-1}(U) \leq x\} = P\{U \leq F_X(x)\} = F_X(x), \tag{4.16}$$

where we have used the monotonicity of F_X and the fact that U is uniformly distributed. [15]

Acceptance-rejection method

Suppose that we must generate random variates according to a PDF $f_X(x)$, and that the difficulty in inverting the corresponding CDF makes the inverse transform method unattractive.
Assume that we can find a function $t(x)$ such that

$$t(x) \geq f_X(x), \quad \forall x \in I, \tag{4.17}$$

where I is the support of $f_X(\cdot)$. The function $t(\cdot)$ is not a probability density, but the related function $g(x) = t(x)/c$ is, provided that we choose

$$c = \int_I t(x)dx. \tag{4.18}$$

If the distribution $g(\cdot)$ is easy to sample from, it can be shown that the following acceptance-rejection method generates a random variate X distributed according to the density $f_X(\cdot)$:

1. Generate $Y \sim g$.
2. Generate $U \sim \mathcal{U}(0,1)$, independent of Y.
3. If $U \leq f_X(Y)/t(Y)$, return $X = Y$; otherwise, repeat the procedure.

The PDF $f_X(\cdot)$ is called the target density, and the "easy" density $g(\cdot)$ is called the instrumental or candidate density. If the support I is bounded, a natural choice of instrumental density is the uniform distribution on I, and we may set

$$c = \max_{x \in I} f_X(x). \tag{4.19}$$

[15]

4.2.3 Normal variates generation

Sampling from a normal distribution is a common need in financial applications. Therefore we now introduce the most relevant generation methods for the purpose of this work: the ones for the normal variates.

Recall first that if $X \sim \mathcal{N}(0,1)$, then $\mu + \sigma X \sim \mathcal{N}(\mu, \sigma^2)$; hence, we just need a method for generating a stream of independent standard normal variables.

At this purpose, there is a set of ad hoc methods.

Box-Muller method

The Box-Muller method takes advantage of the representation of points on the plane by polar coordinates. Consider two independent variables $X, Y \sim \mathcal{N}(0,1)$, and let (R, θ) be the polar coordinates of the point of Cartesian coordinates (X, Y) in the plane, so that

$$d = R^2 = X^2 + Y^2, \quad \theta = \tan^{-1} Y/X. \tag{4.20}$$

Given independence, the joint density of X and Y is

$$f(x,y) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \frac{1}{\sqrt{2\pi}} e^{-y^2/2} = \frac{1}{2\pi} e^{-(x^2+y^2)/2} = \frac{1}{2\pi} e^{-d/2}. \tag{4.21}$$

The last expression looks like a product of an exponential density for d and a uniform distribution; the term $1/2\pi$ may be interpreted as the uniform distribution for the angle $\theta \in (0,2\pi)$. To properly express the density in terms of (d, θ), we should take the Jacobian J of the transformation from (x, y) to (d, θ) into account. After some calculation it can be concluded that $J = 2$, and the correct density in the alternative coordinates is

$$f(d,\theta) = \frac{1}{2} \frac{1}{2\pi} e^{-d/2}. \tag{4.22}$$

Therefore we may generate R^2 as an exponential variable with mean $2, \theta$ as a uniformly distributed angle, and then transform back into Cartesian coordinates to obtain two independent standard normal variates. The Box-Muller algorithm may be implemented as follows:

1. Generate two independent uniform variates $U_1, U_2 \sim \mathcal{U}(0,1)$.
2. Set $R^2 = -2 \log U_1$ and $\theta = 2\pi U_2$.
3. Return $X = R \cos\theta, Y = R \sin\theta$.

In practice, this algorithm may be improved by avoiding the costly evaluation of trigonometric functions and integrating the Box-Muller approach with the rejection approach. The idea results in the polar rejection method. [15]

Multivariate normal distribution sampling

In many financial applications one has to generate variates according to a multivariate normal distribution with expected value array $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Firstly we have to find the Cholesky factor for Σ, i.e., an upper triangular matrix $\mathbf{U}$ such that $\boldsymbol{\Sigma} = \mathbf{U}^{\mathrm{T}}\mathbf{U}$. Then we apply the following algorithm:

1. Generate n independent standard normal variates $Z_1, \ldots, Z_n \sim \mathcal{N}(0,1)$.
2. Return $\mathbf{X} = \boldsymbol{\mu} + \mathbf{U}^{\mathrm{T}}\mathbf{Z}$, where $\mathbf{Z} = [Z_1, \ldots, Z_n]^{\mathrm{T}}$.

We should note that Cholesky decomposition is not the only one that we can use. An alternative is the square root matrix $\boldsymbol{\Sigma}^{1/2}$, i.e., a symmetric matrix such that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{\Sigma}^{1/2}$. [15]

4.2.4 Path generation

Let us consider the stochastic process S_t describing a stock price that we model, as we said in Section 4.1.2, with an Itô stochastic differential equation:

$$dS_t = a(S_t, t)dt + b(S_t, t)dW_t, \tag{4.23}$$

where the function $a(S_t, t)$ defines the drift component, the function $b(S_t, t)$ defines the volatility component, and dW_t is the differential of a standard Wiener process.
We know that the standard Wiener process is a Gaussian process with stationary and independent increments such that

$$W_{t+s} - W_t \sim \mathcal{N}(0, s). \tag{4.24}$$

Sampling paths of a standard Wiener process with any time discretization involves no more than the generation of a stream of independent standard normals.

When faced with the more general Ito process of Eq. (4.23), we discretize using the Euler scheme as follow.

$$\delta S_t = S_{t+\delta t} - S_t = a(S_t, t)\delta t + b(S_t, t)\sqrt{\delta t}\epsilon. \tag{4.25}$$

For instance, consider the geometric Brownian motion described by

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{4.26}$$

and the Euler scheme yields

$$S_{t+\delta t} = (1 + \mu\delta t)S_t + \sigma S_t\sqrt{\delta t}\epsilon. \tag{4.27}$$

This is easy to implement, however the marginal distribution of each variable S_t is normal, whereas we know from Section 4.1.2 that it should be log-normal. This is not quite a negligible issue when we model stock prices that are supposed not to get negative.

The value obtained by straightforward application of the Euler scheme may be considered as a predictor

$$\hat{S}_{t+\delta t} = S_t + a(S_t, t)\delta t + b(S_t, t)\sqrt{\delta t}\epsilon. \tag{4.28}$$

[15]

geometric Brownian motion simulation

We can rewrite the Geometric Brownian Motion

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{4.29}$$

using Itô's lemma, as

$$d\log S_t = \left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t. \tag{4.30}$$

Taking exponentials to get rid of the logarithm yields

$$S_t = S_0 \exp\left(\nu t + \sigma \int_0^t dW(\tau)\right), \tag{4.31}$$

where $\nu = \mu - \sigma^2/2$. Since the integral of a standard Wiener process is normally distributed, from a computational point of view we may rewrite the last expression as

$$S_t = S_0 e^{\nu t + \sigma\sqrt{t}\epsilon}, \quad \epsilon \sim \mathcal{N}(0,1), \tag{4.32}$$

which shows log-normality of prices. From a path generation point of view, the last expression is used as

$$S_{t+\delta t} = S_t \exp(\nu \delta t + \sigma\sqrt{\delta t}\epsilon), \tag{4.33}$$

from which it is easy to generate sample paths of GBMs. [15]

This last equation is the one implemented one in this work.

4.3 Variance reduction methods

The variance reduction strategies aim at improving the efficiency of Monte Carlo methods without introducing any bias. In this Chapter we present the variance reduction methods used in the numerical example, illustrated in the last chapter.

4.3.1 Antithetic sampling

The antithetic sampling approach is very easy to apply and does not rely on any deep knowledge about the specific problem we are tackling. In crude Monte Carlo, we generate a sample consisting of independent observations, but certain correlation may be helpful.

Consider a sequence of paired replications $(X_1^{(i)}, X_2^{(i)}), i = 1, \ldots, n$:

$$\begin{pmatrix} X_1^{(1)} \\ X_2^{(1)} \end{pmatrix}, \begin{pmatrix} X_1^{(2)} \\ X_2^{(2)} \end{pmatrix}, \ldots, \begin{pmatrix} X_1^{(n)} \\ X_2^{(n)} \end{pmatrix}. \tag{4.34}$$

We allow for correlation within each pair, but observations are independent.

This means that $X_j^{(i_1)}$ and $X_k^{(i_2)}$ are independent however we choose $j, k = 1,2$, provided $i_1 \neq i_2$. Thus, the pair-avaraged observations

$$X^{(i)} = \frac{X_1^{(i)} + X_2^{(i)}}{2} \tag{4.35}$$

are independent and can be used to build a confidence interval as usual. Considering the sample mean $\overline{X}(n)$ based on the pair-averaged observations $X^{(i)}$, its variance is given by

$$Var[\overline{X}(n)] = \frac{Var(X)}{2n}\Big(1 + \rho(X_1, X_2)\Big). \tag{4.36}$$

In Eq. (4.36) we can see the potential role of correlation; in order to reduce the variance of the sample mean, we should take negatively correlated replications within each pair. This is easily achieved by using a random number sequence $\{U_k\}$ for the first replication in each pair, and then $\{1^{-}U_k\}$ in the second one. Since the input streams are negatively correlated, we hope that the output observations will be, too. This is true if we require a monotonic relationship between them.

In the normal variates case, we may simply generate a sequence $\epsilon_i \sim \mathcal{N}(0,1)$ and use the sequence $-\epsilon_i$ for the antithetic sample. [15]

4.3.2 Control variates

Antithetic sampling is a very simple techniques that, provided the monotonicity assumption is valid, do not require much knowledge about the systems we are simulating. Better results might be obtained by taking advantage of deeper, domain-specific knowledge.

Suppose that we want to estimate $\theta = E[X]$, and that there is another random variable Y, with a known expected value ν, which is correlated with X. The variable Y is called the control variate. The correlation Y may be exploited by adopting the controlled estimator

$$X_C = X + c(Y - \nu), \tag{4.37}$$

where c is a parameter that we must choose.
It easy to see that

$$\begin{aligned} &E[X_C] = \theta, \\ &Var(X_C) = Var(X) + c^2 Var(Y) + 2cCov(X,Y). \end{aligned} \tag{4.38}$$

The first formula shows that the controlled estimator is an unbiased estimator of θ. The second formula suggests that we could even minimize the variance by choosing the optimal value for c:

$$c^* = -\frac{Cov(X,Y)}{Var(Y)}, \tag{4.39}$$

in which case we find

$$\frac{Var(X_C^*}{Var(X)} = 1 - \rho_{XY}^2, \tag{4.40}$$

where ρ_{XY} is the correlation between X and Y.

In practice, the optimal value of c must be estimated, since $Cov(X,Y)$ and possibly $Var(Y)$ are not known. This may be accomplished by running a set of pilot replications to estimate them, the pilot replications should be discarded in order to avoid some bias in the estimate of θ. [15]

The control variates used in this work for the path generation is the sum of prices. In fact the expected value of the sum of the stock prices Y, as defined in Section 4.1.2, is (under the risk-neutral measure):

$$E[Y] = E\left[\sum_{i=0}^{N} S(t_i)\right] = \sum_{i=0}^{N} S(0)e^{ri\delta t} = S(0)\frac{1 - e^{r(N+1)\delta t}}{1 - e^{r\delta t}}. \tag{4.41}$$

4.4 Low-Discrepancy Sequences

In the sections above we deal with the generation of pseudo-random numbers, with some strategies for the variance reduction useful in financial and economic applications.

The importance to underline that the numbers generated with the previous methods are pseudo-random will be more clear in this section, that aim to present the generation methods of quasi-random numbers.

In order to start introducing this topic, it is useful to remember that the aim of a Monte Carlo simulation is actually to estimate a multidimensional integral on the unit hypercube:

$$\int_{(0,1)^m} h(\mathbf{u}) d\mathbf{u}. \tag{4.42}$$

We need a stream of i.i.d. random numbers to fill the integration domain in a satisfactory manner. When a regular grid derived from classical product rules for numerical integration is not feasible, we may fill it by random numbers, but we could also use alternative deterministic sequences that are evenly distributed. This last concept is the base of quasi-random numbers. However, we need to clarify what a "good coverage" is, which in turn requires a well-defined way to measure its quality.

Assume that we want to generate a sequence $\mathcal{L}_N$ of N points in I^m,

$$\mathcal{L}_N = (x^{(1)}, x^{(2)}, \ldots, x^{(N)}),$$

in order to cover the domain in a satisfactory way. If the points are well distributed, the number of them included in any subset G of I^m should be roughly proportional to its volume $\text{vol}(G)$.
Given a vector $x = [x_1, x_2, \ldots, x_m]^{\mathrm{T}}$, consider the rectangular subset G_x defined as

$$G_x = [0, x_1) \times [0, x_2) \times \cdots \times [0, x_m), \tag{4.43}$$

which has volume $x_1 x_2 \cdots x_m$. Denoting $S(\mathcal{L}, G)$ a function counting the number of points in the sequence $\mathcal{L}$ that are contained in a subset $G \subset I^m$, a possible way to define discrepancy is

$$D^*(\mathcal{L}_N) = D*(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}) = \sup_{x \in I^m} \left| S(\mathcal{L}_N, G_{\mathbf{x}}) - N x_1 x_2 \cdots x_m \right|. \tag{4.44}$$

To be precise, this is called the star discrepancy, and it measures the worst-case distance between the ideal and the actual number of points included in a rectangle of the type specified by Eq. (4.43).

Some theoretical results suggest that low-discrepancy sequences may perform better than pseudo-random sequences obtained through an LCG or its variations. From the theory of confidence intervals, we know that the estimation error with Monte Carlo sampling is

$$O\left(\frac{1}{\sqrt{N}}\right), \tag{4.45}$$

where N is the sample size. With certain point sequences, it can be shown that the error is

$$O\left(\frac{(\log N)^m}{N}\right), \tag{4.46}$$

where m is the dimension of the space in which we are integrating.

It is conjectured that the expression in Eq. (4.46) is a lower bound on what can be achieved, and the term "low-discrepancy sequence" is used when referring to sequences achieving this bound. [15]

In the following, we illustrate the basic ideas behind the two of them used in this work, namely, Halton and Sobol sequences.

4.4.1 Van der Corput sequence

Defining b a prime number we can represent an integer number n in a base b by

$$n = (\cdots d_4 d_3 d_2 d_1 d_0)_b,$$

then if we reflect the digits and add a radix point, we obtain a number h in the unit interval of the form

$$h = (0.d_0 d_1 d_2 d_3 d_4 \cdots)_b.$$

More formally if we rewrite the integer number n as

$$n = \sum_{k=0}^{m} d_k b^k,$$

the nth number in the Van der Corput sequence with base b is

$$V(n, b) = \sum_{k=0}^{m} d_k b^{-(k+1)}. \tag{4.47}$$

[15]

4.4.2 Halton sequences

An **Halton sequence** in multiple dimensions is obtained by associating a Van der Corput sequence with each dimension, using prime numbers for each base.

For example an Halton sequence in dimension 3 will have the base $b = 2$ associated with dimension 1, base $b = 3$ with dimension 2 and base $b = 5$ with dimension 3.

It can be shown that Van der Corput sequence with high base features long monotonic subsequences, and this results in a very bad coverage of the unit square when we use Halton sequences with high bases associated with high dimensions. [15]

4.4.3 Sobol sequences

To avoid the problem in high-dimensional spaces faced by the Halton sequences, we can think to use only Van der Corput sequences with low basis.

In the Sobol sequences only the smallest base, $b = 2$, is used. In particular, suitable permutations of the corresponding Van der Corput sequence are associated with each dimension.

Chapter 5

Numerical example

The aim of the work is to investigate the more suitable scenario generation method for the ALM problem solved. In order to do so we have to compare the results of the optimization problem starting from different scenario. As we said at the beginning, we want to be able to meet the liabilities and to have some extra utility at the end of the planned horizon.

In this chapter we present the problem with a numerical example in order take conclusions based on numerical results. The problem is performed with MATLAB R2022b and a 16Gb RAM.

In Sections 2.3 and 2.4 we present the two formulations of the ALM problem from a generic and theoretical point of view. It should be noted that we use the same variable names as Sections 2.3 and 2.4.

5.1 Data

Firstly we have to choose the data for portfolio optimization.

For the sake of simplicity the portfolio is assumed to be composed by asset classes and not on specific actions.

In particular we take eight asset classes, four equities:

- World equity,
- Eurozone equity,
- Emerging Market equity,
- Asia Pacific (ex Japan) equity;

and four bonds:

- Global aggregate,
- Eurozone governative,

- Emerging Market hard currency bond,
- Asia Pacific aggregate.

For each of these has been taken the historical series of prices of the last 10 years, more precisely the working days from december 2012 to december 2022. The historical series were taken from different free databases on Bloomberg and Yahoo Finance.

From the historical prices we compute the historical returns, in particular we use the exchange rate to compute all asset prices in Euro and on these prices we compute the returns as

$$r_t = \log\left(\frac{P_t}{P_{(t-1)}}\right), \tag{5.1}$$

where r_t is the return at day t and $P_T, P_{(t-1)}$ are the prices in days $t, t-1$ respectively.

We have to notice that the historical period chosen contains two unpredictable world crises like the pandemic from Covid-19 of 2020 and the conflict in Ukrainian soil of 2022; both of these two had a strong impact on the asset prices as we can see in Figures (5.1) and (5.2)

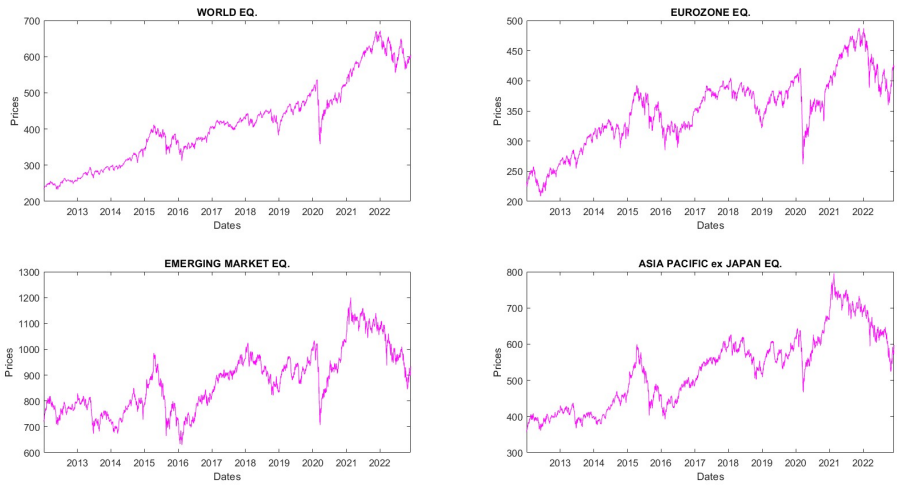

Figure 5.1: Historical series of equity prices.

The prices are firstly saved in an Excel file and the returns are computed in Excel directly; then both are read in MATLAB for the computations.

5.2 Scenario Tree Construction

The basis of all generations of scenarios made in this work is the scenario tree.

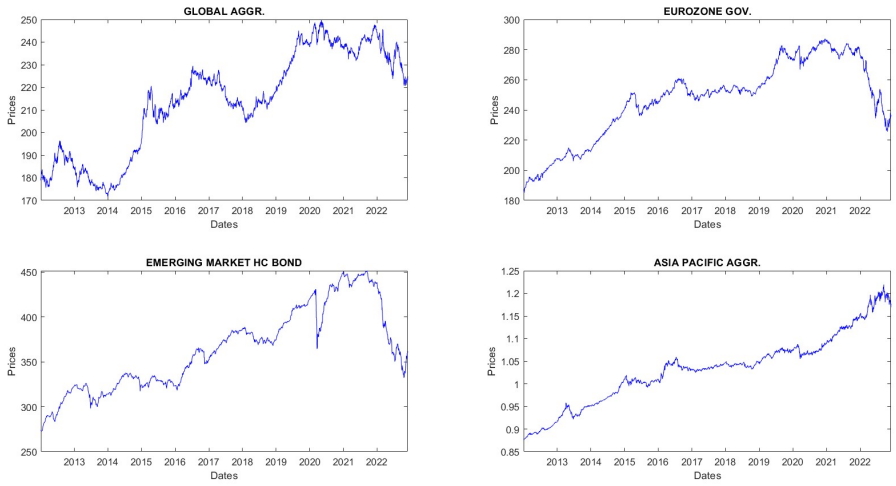

Figure 5.2: Historical series of bond prices.

In order to perform the simulations on the scenario tree we firstly construct a function that built a tree given a branching factor `branching_factor` and a time horizon `T`. To investigate the importance of the shape of the tree in the scenario generation we also add the possibility to choose between an *uniform* shape, i.e. at each time step we have the same branching factor set by the user, or a *dense* shape, which uses the branching factor chosen by the user in the root node, `branching_factor`-1 in the first step and so on until `branching_factor`=2, at that point `branching_factor` remains equal to two until the set horizon.

The function, named `tree_construction`, gives as output:

- `NodalPartitionMat`: matrix in which each column corresponds to a time step and each row represents the node sequence from root to leaf nodes that defines a scenario;
- `antecedentNodes`: vector in which each entry is the antecedent node of the corresponding position, i.e. `antecedentNodes(i)=j` means that `j` is the antecedent node of node `i`.
- `numberNodes`: total number of nodes of scenario tree.
- `nodes`: matrix $T \times 2$ in which each row has two values corresponding to the first and the last node of a time step.

From the `NodalPartitionMat` matrix is possible to extract all the subsets of nodes used in the optimization problems so we doesn't need to save them in other variables.

For all the problems performed we construct the tree and then we sample the values simulated with the different scenario generation methods. The idea is that equity asset

classes and bond asset classes are independent, so we generate them separately and we "recombine" the values on the scenario tree. This is important to underline because explain the reason why at first we consider equities and bonds separately. In every simulation a seed is set in order to satisfy the repeatability requirement.

5.3 Basic formulation

In the base formulation we solve the optimization problem with the specific optimization toolbox functions. The input data are the returns of the asset classes considered, in particular we want to solve the problem on the future, so we use the historical means `mu_e, mu_b` and the historical variances `sigma_e, sigma_b` to simulate future returns on a scenario tree.

5.3.1 First variant: piecewise linear utility and normally distributed returns

Scenario generation

Firstly we have to generate scenarios in order to perform and solve the optimization problem. In this case we assume that all the asset classes have returns normally distributed, in particular we consider a multivariate normal distribution both on equities and bonds.

By considering for the tree the parameters:

- branching factor: `branching_factor` = 3,
- time horizon: `T` = 5;

we generate the scenario with Monte Carlo sampling, using the build-in MATLAB function `mvnrnd`, and with Antithetic Sampling, using a function built ad hoc named `MultiNormAS`.

In `MultiNormAS` we apply the antithetic sampling technique to the multivariate normal distribution sampling. We want to sample from a multivariate normal distribution with parameters μ and $\mathbf{\Sigma}$,the MC algorithm is:

1. Generate n independent standard normal variates $Z_1, \ldots, Z_n \sim \mathcal{N}(0,1)$.
2. Return $\mathbf{X} = \mu + \mathbf{U}^{\mathtt{T}}\mathbf{Z}$, where $\mathbf{Z} = [Z_1, \ldots, Z_n]^{\mathtt{T}}$.

Where $\mathbf{U}$ is computed with the Cholesky decomposition of $\mathbf{\Sigma}$ In the function `MultiNormAS` we compute X_1 as in the second step of the algorithm above, $X_2 = \mu - U^{\mathtt{T}}Z$ and we consider as the final output $X = \frac{X_1+X_2}{2}$, as required by the variance reduction technique of Antithetical Sampling.

We can do a first observation on the results of the two simulations by looking the confidence intervals for the mean and the standard deviation for each asset class in the two methods. The confidence intervals for the mean are:

```
ci_mc_mean =                          ci_as_mean =

   -0.0011    0.0007                    -0.0004    0.0011
   -0.0016    0.0007                    -0.0010    0.0008
   -0.0016    0.0005                    -0.0003    0.0013
   -0.0015    0.0007                    -0.0003    0.0013
    0.0000    0.0008                    -0.0000    0.0005
    0.0000    0.0006                    -0.0000    0.0003
   -0.0000    0.0005                    -0.0000    0.0004
   -0.0001    0.0002                    -0.0001    0.0002
```

The confidence intervals for the standard deviation are:

```
ci_mc_sd =                            ci_as_sd =

    0.0080    0.0093                    0.0064    0.0074
    0.0103    0.0119                    0.0079    0.0092
    0.0096    0.0111                    0.0072    0.0084
    0.0102    0.0118                    0.0073    0.0084
    0.0034    0.0040                    0.0024    0.0028
    0.0027    0.0031                    0.0017    0.0019
    0.0027    0.0031                    0.0019    0.0022
    0.0017    0.0019                    0.0012    0.0014
```

where `_mc` and `_as` meaning Monte Carlo and Antithetic Sampling respectively.
As we can notice confidence intervals for the AS method are smaller than the MC ones, as we expected from theory.

Problem formulation

The mathematical problem solved is the following

$$
\begin{aligned}
\max \quad & \sum_{s\in\mathcal{S}} p_s(qw_+ - rw_-) \\
s.t. \quad & \sum_{i\in\mathcal{I}} x_{i,n_0} = W_0, \\
& \sum_{i\in\mathcal{I}} R_{i,n}x_{i,a(n)} = \sum_{i\in\mathcal{I}} x_{i,n} \quad \forall n \in \mathcal{T}, \\
& \sum_{i\in\mathcal{I}} R_{i,s}x_{i,a(s)} - L = w_+ - w_- \quad \forall s \in \mathcal{S}, \\
& x_{i,n}, w_+, w_- \geq 0.
\end{aligned}
\tag{5.2}
$$

Where the parameter are set as $q = 1, r = 2.5 \quad W_0 = 55000, L = 80000$.

The problem structure in MATLAB is constructed in a function named `probALM`. This function returns the optimization problem with the input data:

- `returns`: a matrix with return simulations, each column corresponds to an asset class and each row to a realization;
- `q`: surplus coefficient value;
- `r`: shortfall coefficient value;
- `L`: liability to meet at the time horizon;
- `W0`: initial wealth value;
- `NodalPartitionMat`;
- `antecedentNodes`.

We first set the returns generated with MC and then with AS and we solve the problem with the built-in function `solve` and observe the value at the optimum of x, the amount allocated in each asset class.

In particular we are interest in what we should do in $t = 0$, in the root node of the tree, therefore we look the first column of x that represents the initial allocation of wealth.

Results

The results obtained from the previous steps are the following.
Using the MC method we have:

```
>> solutionALM.x(:,1)

ans =
            0
            0
            0
            0
            0
        55000
            0
            0
```

Using the AS:

```
>> solutionALM.x(:,1)

ans =
            0
            0
            0
            0
            0
        55000
            0
            0
```

The results are the same, and we can notice that the problem solution is not diversified at all. In fact in the solution we allocate all the initial wealth in an unique asset class.

The reason could be the piecewise linear utility function, that can't capture the risk well, or the scenarios, that create some arbitrage opportunity. The first hypothesis will be investigated in Case 5.3.2, while for the second one we could generate more scenarios by using a tree with an higher branching factor and a *dense* shape.

Therefore we consider the tree the parameters:

- branching factor: `branching_factor` $= 6$,
- time horizon: `T` $= 5$;

now we have 1237 nodes while before they were 364.
Unfortunately from the results we still see a non-diversified initial allocation, both for MC and AS simulations.

We conclude that this problem is too simply to be able to capture the diversification risk.

5.3.2 Second variant: quadratic utility and normally distributed returns

Scenario generation

In this case we have the same assumptions on the return distribution of the first variant (5.3.1).

Therefore we have the same methods and implementation than the first variant (5.3.1).

Problem formulation

The mathematical problem is the following:

$$\begin{aligned} \max \quad & \sum_{s\in\mathcal{S}} p_s(w_s - bw_s^2) \\ s.t. \quad & \sum_{i\in\mathcal{I}} x_{i,n_0} = W_0, \\ & \sum_{i\in\mathcal{I}} R_{i,n}x_{i,a(n)} = \sum_{i\in\mathcal{I}} x_{i,n} \quad \forall n \in \mathcal{T}, \\ & \sum_{i\in\mathcal{I}} R_{i,s}x_{i,a(s)} - L = w_s \quad \forall s \in \mathcal{S}, \\ & x_{i,n}, w_s \geq 0. \end{aligned} \tag{5.3}$$

Where the parameters are set as $b = \frac{1}{80000}$ $W_0 = 55000, L = 80000$.

The problem structure in MATLAB is constructed in a function named `probALM_quad`. This function returns the optimization problem, the input data are the same as the `probALM` function, they only differ in the utility function parameters, instead of `q, r` here we have `b` that is passed by the user.

As in the first variant (5.3.1) we solve the problem using the function `solve` and we observe the first column of x.

Results

We start using a small tree with branching factor equal to 3 and a time horizon of 5 years. In this case we can see that both for MC and AS simulations the solver couldn't find a optimal solution that satisfy the short sell constraint.
The output is

```
No feasible solution found.

quadprog stopped because it was unable to find a point that satisfies
the constraints within the value of the constraint tolerance.

ans =

       16212.49
       -2969.92
       -3672.77
        7039.37
        -578.65
       -2020.28
       -2793.96
       16406.64
```

By considering a more branched tree we can see that the solution change but we face the same problem. We set the branching factor equal to 8 with the same time horizon and a *dense* shaped tree; the results are similar but we report the ones for AS as an example:

```
ans =

       16212.49
       -2969.92
       -3672.77
        7039.37
        -578.65
       -2020.28
       -2793.96
       16406.64
```

We conclude that this problem has too hard constraints, that the solver can't satisfy. The problem is too simply with these formulation and assumptions, so we can't use these

results to make some conclusion on the scenario generation method, that we postpone to the comment on the results of the cases of implemented problem.

5.3.3 Third variant: power utility and log-normally distributed returns

Scenario generation

In this case we assume the returns to be log-normally distributed.
We consider firstly a small tree with a *uniform* shape and in a second moment a more branched tree with a *dense* shape. In particular the parameters chosen are summarized as:

	Tree1	Tree2
Branching factor	2	4
Time horizon	5	5
Shape	`'uniform'`	`'dense'`

Table 5.1: Summary of the parameters chosen for the two trees to compare.

On these trees we simulate the returns log-normally distributed with the built-in function `lognrnd`, that takes in input the mean and the standard deviation of the starting normal distribution. For consistency with what has been done in variants (5.3.1) and (5.3.2), we use the mean and the standard deviation computed with historical data.

Problem formulation

The mathematical problem to be solved is the following:

$$\begin{aligned} \max \quad & \sum_{s\in\mathcal{S}} p_s \left(\frac{w_s^{1-\gamma} - 1}{1-\gamma} \right) \\ s.t. \quad & \sum_{i\in\mathcal{I}} x_{i,n_0} = W_0, \\ & \sum_{i\in\mathcal{I}} R_{i,n} x_{i,a(n)} = \sum_{i\in\mathcal{I}} x_{i,n} \quad \forall n \in \mathcal{T}, \\ & \sum_{i\in\mathcal{I}} R_{i,s} x_{i,a(s)} - L = w_s \quad \forall s \in \mathcal{S}, \\ & x_{i,n}, w_s \geq 0. \end{aligned} \tag{5.4}$$

Where we set the parameter as $\gamma = 2.5 \quad W_0 = 55000, L = 80000$.

The problem structure in MATLAB is constructed in a function named `probALM_pw`. This function returns the optimization problem, the input data are the same as the `probALM` function, they only differ in the utility function parameters, instead of `q, r` here we have `gamma` that is given by the user.

In this case we have a nonlinear utility function, so we have to set an initial value in order to solve the optimization problem with `solve` function. The initial point, named `x0`, is a struct variable with the same fields as the optimization variables with their initial values. Initial point is arbitrarily chosen with initial value of the wealth in the root node equal to W_0 and the value of x equal to $W_0/8$ for each asset class in the root node and zeros for all the other nodes and asset classes.

Results

The output of the previous steps is the following.
Using "Tree1" we have the initial wealth allocation:

```
ans =

   1.0e+04 *

    0.0011
    0.0015
    0.0012
    5.4909
    0.0011
    0.0020
    0.0010
    0.0012
```

while with "Tree2" we have:

```
ans =

   1.0e+04 *

    0.0010
    0.0016
    0.0013
    0.0010
    0.0011
    4.2887
    0.0013
    1.2040
```

In both outputs we have the warning from the solver to have find a local minimum.

We can notice that in this case there is diversification, even if it is not diversified as we hope because it isn't the optimal solution, it is an appreciable result.

5.3.4 Conclusions and comments

We can conclude that the *basic formulation* of the problem is not good because it is too simplified with respect to the reality.

These assumptions and simplifications lead the problem to be locally risk neutral (as the utility used in the first variant (5.3.1)) or too restrictive in the constraints (as in the second variant (5.3.2)). A good interpretation is the last variant studied (5.3.3) in which we assume a log-normal distribution for the returns and a power utility function. In this case even if the results are more acceptable than the previous cases, they are too far from to be applicable.

For the results obtained it is not possible to study the impact of different scenario generation methods.

In order to do so we have to consider a more elaborated problem with more realistic assumptions, as we do in the next section with the *extended formulation* of the problem.

5.4 Extended formulation

The problems solved are two. We cite them with the mathematical problem formulation and the assumption used. In all of these statement the prices are simulated following a GBM and the liabilities are sampled from a uniform distribution $\mathcal{U}(60000,80000)$.

5.4.1 Scenario generation

The scenario tree is constructed as in the *basic formulation* of the problem, so we refer to scenario trees generated as explained in the previous section.

In this formulation of the ALM problem we use the simulations of the asset class prices instead of their returns.

The assumption is that the prices following a Geometric Brownian Motion. The simulation is done building a function; as we want to study different scenario generation methods we built different functions, each for every method.

For example, the function that uses the Monte Carlo sampling is named `PriceAssetMC` and it receives in input:

- `s0`: initial price values of the asset classes;
- `T`: time horizon;
- `mu`: annualized mean of historical data;
- `sigma`: annualized standard deviation of historical data;
- `numSteps`: number of steps in the sample path;
- `numRep`: number of simulation for each asset class.

These input are almost the same for all functions.

The method, and the corresponding function implemented, used in this case are:

- Monte Carlo: `PriceAssetMC`,
- Antithetic Sampling: `PriceAssetAS`,
- Control Variates: `PriceAssetMCCV`,
- Quasi Monte Carlo: `PriceAssetQuasiMC`;

while the first two functions have exactly the same input data, listed before, the third and the fourth ones have some differences.

For the function `PriceAssetMCCV` implemented with Control Variates we need one more input than the first two functions, that is the number of pilot replication named `numR_test`.

The last function `PriceAssetQuasiMC`, implemented with the generation of quasi-random number instead of pseudo-random number, needs two more input than `PriceAssetMC` and `PriceAssetAS`. In fact we can chose which kind of low discrepancy sequence to implement between Sobol and Halton and we have to set the number of initial points in sequence to omit.

The construction of both Sobol and Halton sequence is done using build-in functions `sobolset` and `haltonset` respectively.

We assume that the final values of the asset paths are the prices searched.
In functions `PriceAssetMC`, `PriceAssetAS` and `PriceAssetQuasiMC` we compute the sample paths for $t = T$ that we can see in Figures (5.4) and (5.3) below for all the asset classes considered. The prices at time step t simulated is the last value of the sample path with time horizon t.

We can see that there is a slightly difference in the sample paths generated with pseudo-random and quasi-random numbers, while the variance reduction of Antithetic Sampling gives very different sample paths in which the variance reduction is visible.

In this section we have also to simulate the liability to meet at each time steps. We assume that in the nodes of the same time step there is the same liability to meet. The sampling of the liability from a uniform density on the interval [60000,80000] is done with the build-in function of MATLAB.

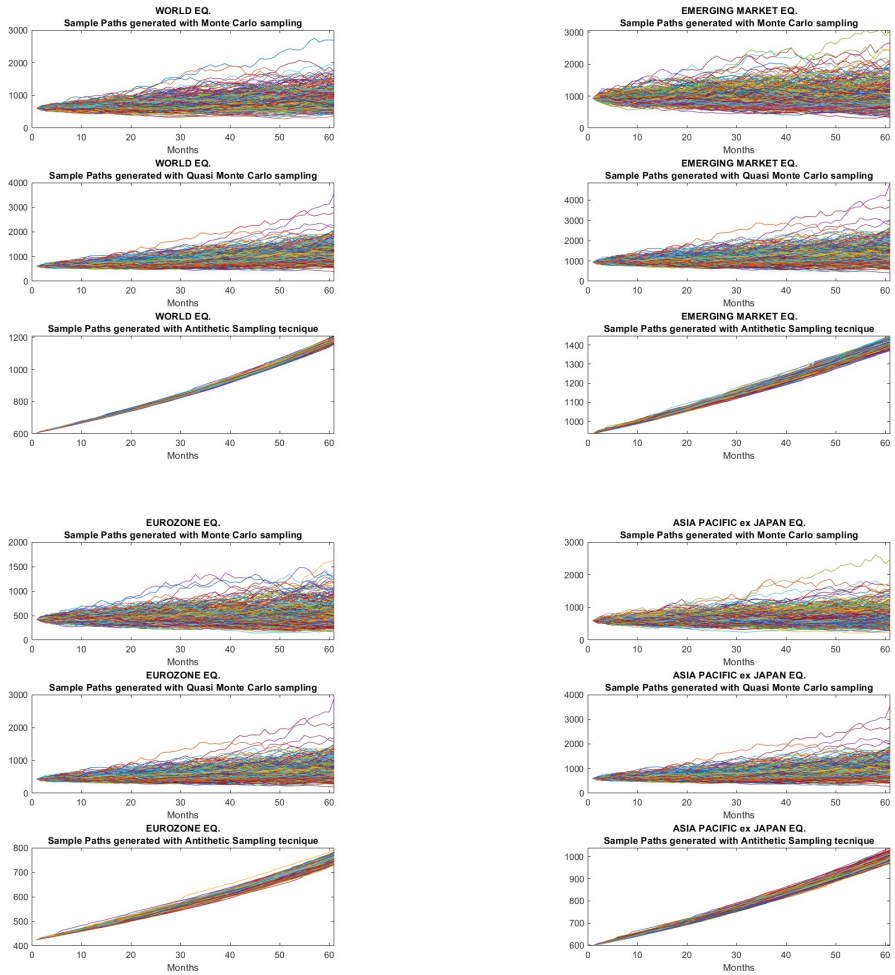

Figure 5.3: Sample paths in the different methods implemented for the equity asset classes.

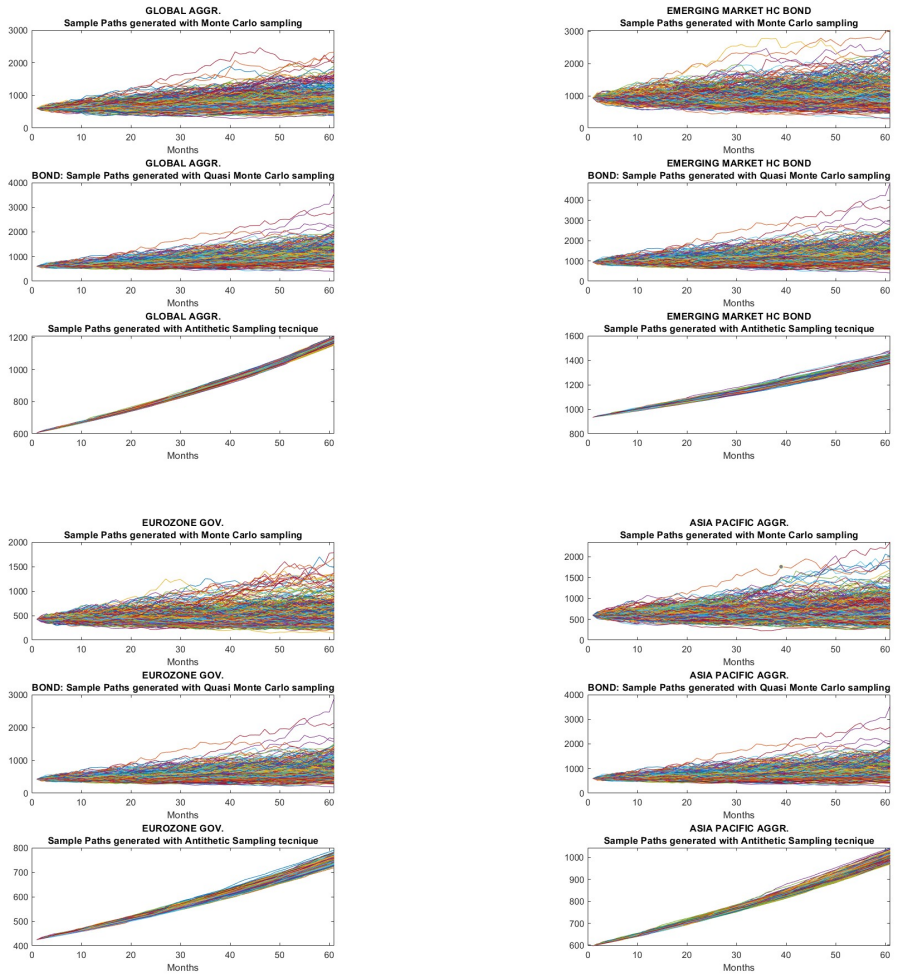

Figure 5.4: Sample paths in the different methods implemented for the bond asset classes.

5.4.2 First variant: piecewise linear utility

Problem formulation

The mathematical problem solved is as following:

$$
\begin{aligned}
\max \quad & \sum_{s\in\mathcal{S}} p_s(qw_+ - rw_-) \\
s.t. \quad & x_i^{n_0} = h_i^{n_0} + z_i^{n_0} - y_i^{n_0}, && \forall i \in \mathcal{I} \\
& x_i^n = x_i^{a(n)} + z_i^n - y_i^n, && \forall i \in \mathcal{I}, \forall n \in \mathcal{T} \\
& (1-c)\sum_{i\in\mathcal{I}} P_i^n y_i^n - (1+c)\sum_{i\in I} P_i^n z_i^n = L^n, && \forall n \in \mathcal{N}\setminus\mathcal{S} \\
& w_+ - w_- = (1-c)\sum_{i\in\mathcal{I}} P_i^s x_i^{a(s)} - L^s, && \forall s \in \mathcal{S} \\
& x_i^n, y_i^n, z_i^n, w_+, w_- \geq 0.
\end{aligned}
\tag{5.5}
$$

Where the parameters are chosen as $q = 1, r = 2.5 \quad c = 0.05, h_i^{n_0} = 6875 (= 55000/8) \quad \forall i \in \mathcal{I}$.

By solving this problem we can investigate the impact of considering a more complete formulation.

Firstly we generate the scenarios on a small tree with a *uniform* shape with `branching_factor = 3` and `T = 5`, then we consider a more branched tree with parameters `branching_factor = 6`, `T = 5` and a *dense* shape.

Results

Tree with `uniform` shape, `branching_factor = 3` and `T = 5`

As we said in the previous section for the *basic formulation* of the problem, we are interested in the allocation of the wealth at the root node, so we observe the first column of `x`, that in this case represents the wealth owned for each asset class after the rebalancing. In order to know the strategy to implement we have to look the values of the first columns of the decision variables `z` and `y`, to know the amounts to sell and to buy for each asset class. In any case, the variable `x` gives us the idea of the diversification of the strategy, so it is a good indicator.

We perform the resolution of the problem, using the build-in function `solve`, for all the four methods implemented.
The results are displayed following the order:

1. Monte Carlo,
2. Antithetic Sampling,
3. Control Variates,
4. Quasi Monte Carlo;

and they are reported below:

```
first_choice_pl =

   1.0e+04 *

         0    4.9377         0    1.5864
    3.9026    0.0535         0    0.6875
         0         0         0         0
    0.6875         0    6.7252    0.6875
    0.0109         0         0    0.6875
    0.6875         0         0    0.6875
    0.6875         0         0    0.6875
    0.6875         0         0    0.6875
```

We can notice that the worst result is obtained with the Control Variates method because there's no diversification. Antithetic Sampling performs slightly better while

Monte Carlo and Quasi Monte Carlo methods give nearly results and they seem reasonable.

The more interesting result is the first column, the one corresponding to the Monte Carlo method. In the *basic formulation* we have noticed that a piecewise linear function leads to be locally risk neutral and as a consequence we obtained a non-diversified allocation. In this formulation we have a diversified portfolio, meaning that the risk is better captured.

Then we display the value of the objective function, the terminal expected utility, in the different methods.

```
ans =

   1.0e+08 *

    1.5498    0.6315    1.8668    0.6925
```

In all the cases we still have some wealth at the end of the time period, that was one of our goals.

We use these results to repeat the optimization problem in order to see if we can roll the horizon and cover a longer time period.

We consider to repeat our problem 10 times, so we want to see if the problem is able to cover a time period of 50 years. This is done by repeating the problem formulation 10 times, each of them we set the value of `h` equals to the last value of `x` of the previous iteration and we simulate different prices over the same tree structure.

Therefore the idea is to solve the problem several times, each of them starting with the residual wealth of the previous iteration. This is done using the amount holding for each asset after the rebalancing (`x`) at the end of the time horizon as the initial holding for each asset in the root node for the next iteration.

A the end of the 10 cycles the values of the objective function in the four different methods are:

```
ans =

   1.0e+10 *

    0.1054    0.0058    2.8025    0.0067
```

We can notice that both with Antithetic Sampling and Quasi Monte Carlo methods we obtain the almost same expected wealth, and the Monte Carlo sampling seems to be the better method with these assumptions and objective function.

Tree with `dense` shape, `branching_factor = 6` and `T = 5`

We report the values of the objective function for all the four methods at the end of the 10 periods cycle.

```
ans =

   1.0e+11 *

    3.6818    0.0006    0.0952    0.0006
```

With initial allocation in the root note of an initial wealth equal to 55000 equally distributed in the asset classes, after the first rebalancing we hold:

```
first_choice_pl =

   1.0e+04 *

         0    4.5584    5.0153    3.9315
         0    0.6875    0.0060    0.6875
         0         0         0         0
         0         0         0    0.6875
    0.6875         0         0         0
    0.6875         0         0         0
    2.8126         0         0         0
    0.6875         0         0    0.0144
```

We can notice that with a more branched tree we obtain a final expected wealth higher than with the problem performed on a smaller tree in MC and CV methods, while the others are similar but smaller; however we don't have relevant differences in the holdings. The more appreciable difference is the diversification in the Control Variates method, that we don't have with the small tree.

Trees comparison

Now we can think how well the uncertainty is captured by the different sampling methods considered, taking trees with different branching factors, therefore taking more or less branched trees that theoretically represent better or worse the uncertainty.
The impact of the branching factor on the results can be seen by observing how the value of the objective function at the optimum changes. We fix the time horizon `T = 5` and we prove different values of the branching factor.

Branch Factor	Monte Carlo	Antithetic Sampling	Control Variates	Quasi MC
`B = 2`	1.4577	0.6295	1.8410	0.6365
`B = 3`	1.5498	0.6315	1.8668	0.6925
`B = 4`	2.8125	0.6319	1.6135	0.6785
`B = 5`	2.0965	0.6315	1.3333	0.6776
`B = 6`	1.5986	0.6287	1.4063	0.6810

Table 5.2: Summary of the results of the optimization problem with piecewise linear utility. Results are expressed in hundreds of millions.

From Table 5.2 we can see that Antithetic Sampling and Quasi Monte Carlo methods produce the more stable solutions over the number of branches for nodes, so even with different number of nodes we have similar solutions meaning that these two methods sample well even with low number nodes.

Out-of-sample tests

"How good is the solution found?" in order to answer to this question we have to implement the solutions obtained above with different price simulation methods. Therefore we perform an out-of-sample test for each solution obtained in Table 5.2. We sample with each method studied a new stream of prices, independent from the one used to solve the optimization problem, and we perform the solution obtained on these new prices.
In few words, we plug the first-stage solution obtained above into several second-stage problems, each one corresponding to an observation of the risk factors.

In the following table we report the value of the objective function after the out-of-sample test.

Branch Factor	Monte Carlo	Antithetic Sampling	Control Variates	Quasi MC
`B = 2`	0.2378	0.5331	0.6310	0.5364
`B = 3`	0.3384	0.4298	1.0670	0.4416
`B = 4`	1.2351	0.6329	0.8436	0.5478
`B = 5`	1.0118	0.6080	0.5908	0.6359
`B = 6`	0.4712	0.6301	0.8697	0.5593

Table 5.3: Summary of the results of the out-of-sample tests of optimization problem with piecewise linear utility. Results are expressed in hundreds of millions.

From Table 5.5 we can see that for high number of nodes the Antithetic Sampling has the better results. In fact with this method we have results near to each other.

5.4.3 Second variant: quadratic utility

Problem formulation

The mathematical problem solved is as following:

$$
\begin{aligned}
\max \quad & \sum_{s\in\mathcal{S}} p_s(w_s - bw_s^2) \\
s.t. \quad & x_i^{n_0} = h_i^{n_0} + z_i^{n_0} - y_i^{n_0}, && \forall i \in \mathcal{I} \\
& x_i^n = x_i^{a(n)} + z_i^n - y_i^n, && \forall i \in \mathcal{I}, \forall n \in \mathcal{T} \\
& (1-c)\sum_{i\in\mathcal{I}} P_i^n y_i^n - (1+c)\sum_{i\in I} P_i^n z_i^n = L^n, && \forall n \in \mathcal{N}\setminus\mathcal{S} \\
& w_s = (1-c)\sum_{i\in\mathcal{I}} P_i^s x_i^{a(s)} - L^s, && \forall s \in \mathcal{S} \\
& x_i^n, y_i^n, z_i^n, w_s \geq 0.
\end{aligned}
\tag{5.6}
$$

Where the parameters are chosen as $b = 1/80000 \quad c = 0.05, h_i^{n_0} = 6875 \quad \forall i \in \mathcal{I}$.

As for the previous section the solution of this problem we can investigate the impact of considering a more complete formulation, with the comparison of the solution obtained with the *base* formulation.

Firstly we generate the scenario on a small tree with a *uniform* shape with `branching_factor = 3` and `T = 5`, then we consider a more branched tree with parameters `branching_factor = 8` and `T = 5` and a *dense* shape.

Results

Tree with `uniform` shape, `branching_factor = 3` and `T = 5`
This problem could lead to negative objective function final values.

In some sense the quadratic utility function is stricter than the piecewise linear one.

The holdings after the rebalancing at the root node are:

```
first_choice_quad =

   1.0e+04 *

    0.2634    2.7409    0.0478    1.3710
    3.1398    1.1396    0.1447    0.7022
    0.1431    0.4247    0.0364    0.3759
    0.4509    0.6213    5.9961    0.5294
    0.3045    0.1098    0.0829    0.4525
    0.8484    0.1489    0.0453    0.5721
    0.3780    0.0663    0.0261    0.4990
    0.6002    0.1234    0.0322    0.6946
```

This result is what we expected in terms of diversification for all the methods.

However in Quasi MC method, by looking the values of the wealth in each scenario at the end of the 10 periods we can observe that there are six negative values, in such scenarios we fail. In the others method we don't have such deal.

The values of the objective function in the four methods, after 10 periods, are:

```
ans =

   1.0e+03 *

    7.4852    0.6923    2.0875    4.5924
```

we can notice that the values of the expected wealth after the tenth iteration on a 5 years time horizon are quite similar for all the method.

Tree with `dense` shape, `branching_factor = 6` and `T = 5`
The results obtained after 10 periods cycle of the problem, are near to the ones obtained with the smaller scenario tree structure.

```
ans =

   1.0e+03 *

    6.3937    0.4420    1.4683    4.4887
```

also the initial holdings after rebalancing are similar

```
first_choice_quad =

   1.0e+04 *

    0.0368    3.0662    4.1957    2.4436
    0.0805    1.0669    0.5305    0.7236
    0.1589    0.3042    0.1340    0.2959
    0.2157    0.6086    0.0820    0.5935
    0.3298    0.0690    0.0186    0.2305
    0.6742    0.0803    0.0182    0.3577
    2.4160    0.0625    0.0197    0.2096
    0.3008    0.0732    0.0091    0.4380
```

we can conclude that for this kind of objective function the tree structure underlying the scenario generation is not very impacting on the results.

However this tree is slightly better than the smaller one because in this case we don't have failures in any methods.

Trees comparison

As for the first variant, we can think how well the uncertainty is captured by the different sampling methods considered, taking trees with different branching factors.
We fix the time horizon `T = 5` and we prove the same different values of the branching factor as in previous variant.

Branch Factor	Monte Carlo	Antithetic Sampling	Control Variates	Quasi MC
`B = 2`	2.0000	2.0000	2.0000	1.9784
`B = 3`	1.9998	2.0000	2.0000	1.9585
`B = 4`	1.9992	2.0000	1.9998	1.9441
`B = 5`	1.9979	2.0000	1.9995	1.9436
`B = 6`	1.9961	2.0000	1.9992	1.9416

Table 5.4: Summary of the results of the optimization problem with quadratic utility. Results are expressed in tens of hundreds.

From Table 5.2 we can see that Antithetic Sampling has the best performance, however all the methods performs very well with this utility function.

Out-of-sample tests

Finally we perform an out-of-sample test for each solution obtained in Table 5.4.

In the following table we report the value of the objective function after the out-of-sample test.

Branch Factor	Monte Carlo	Antithetic Sampling	Control Variates	Quasi MC
`B = 2`	-0.0011	-0.0000	-0.0341	-0.0000
`B = 3`	-0.0210	-0.0000	-0.0607	-0.0031
`B = 4`	-0.5423	-0.0000	-0.6509	-0.0155
`B = 5`	-0.2788	-0.0007	-0.9323	-0.0128
`B = 6`	-2.2171	-0.0052	-0.6545	-0.0290

Table 5.5: Summary of the results of the out-of-sample tests of optimization problem with piecewise linear utility. Results are expressed in hundreds of millions.

From Table 5.5 we can see that Antithetic Sampling has the best results. Also Quasi Monte Carlo performs well.

5.4.4 Conclusions and comments

The results obtained considering the quadratic utility are better than the ones obtained with the piecewise linear utility function for all the methods implemented.

In fact portfolios are more diversified with the quadratic utility and the two cases required similar computational times, so we can conclude that the second case both with the smaller and the larger tree has a better behaviour than the first case in all its variants.

On the other hand the formulation with the piecewise linear utility function seems to be more responsive as scenario tree structure changes, so the results obtained in the first formulation are interesting as well. In this case we have that overall the best results are obtained with Quasi Monte Carlo method, as they are diversified in the root node and similar to each other even with few nodes, finally it has great results in the out-of-sample tests.

We conclude that with a piecewise linear utility function the Quasi Monte Carlo is the best method, while with a quadratic utility function is Antithetic Sampling. Given that Quasi MC performs well even with the quadratic utility, we can conclude that overall is the best method.

Chapter 6

Conclusions and possible implementations

In the present work we want to investigate the impact on the ALM problem solution in stochastic programming of the use of scenarios generated with methods Monte Carlo, Antithetic Sampling, Control Variates and Quasi Monte Carlo with low discrepancy sequences.

At this purpose we study:

- the importance of the mathematical formulation of the problem, by studying the quality and stability of the results of a basic and an extended formulation of the problem;
- the impact of the structure of the scenario tree underlying the problem and scenario generation methods;
- the impact of a piecewise linear utility and a quadratic one, by comparing the results of problems stated with both utility functions;
- the impact of scenario generation methods by observing the quality and stability of results obtained and running the problem on more periods in order to see to see how "conservative" is the solution obtained.

We conclude that it is important to consider a more extended formulation of the problem while a more branched tree doesn't provide relevant improvements on the solution quality.

As we expected a quadratic utility gives appreciable results with a more branched tree, on the other hand the piecewise linear utility function seems to be more responsive as scenario tree changes, therefore it is more interesting for the purpose of the work.

Finally we conclude that variance reduction methods as Anthitetic Sampling and Control Variates doesn't improve the performance of the problem resolution in the piecewise linear utility case, while taking Quasi Monte Carlo simulations gives the best results overall.

The present work can be further developed considering a more complete formulation, for example including contributions in a pension fund case, or taking into account other methods like importance sampling or stratified sampling.

Bibliography

[1] ManMohan S. Sodhi. «LP Modeling for Asset-Liability Management: A Survey of Choices and Simplifications». In: *Operations Research* 53.2 (2005), pp. 181–196. DOI: 10.1287/opre.1040.0185. eprint: https://doi.org/10.1287/opre.1040.0185. URL: https://doi.org/10.1287/opre.1040.0185 (cit. on p. 13).

[2] Paolo Brandimarte. *An Introduction to Financial Markets: A Quantitative Approach*. John Wiley & Sons, 2017 (cit. on pp. 13, 16–18, 20, 26).

[3] Dilawar Ahmad Bhat. «A review of asset liability management models». In: (2020) (cit. on pp. 13, 14).

[4] CAROLINE BANTON. *Asset/Liability Management: Definition, Meaning, and Strategies*. URL: https://www.investopedia.com/terms/a/asset-liabilitymanagement.asp (cit. on p. 14).

[5] Stavros A Zenios and William T Ziemba. *Handbook of Asset and Liability Management: Applications and case studies*. Elsevier, 2007 (cit. on pp. 16, 23, 24, 26).

[6] *Quadratic Utility*. URL: https://www.d42.com/portfolio/analysis/quadratic-utility (cit. on p. 21).

[7] E.J. Anderson. *Business Risk Management: Models and Analysis*. Nov. 2013, pp. 1–367. ISBN: 9781118349465. DOI: 10.1002/9781118749388 (cit. on p. 23).

[8] Giorgio Consigli, Daniel Kuhn, and Paolo Brandimarte, eds. *Optimal Financial Decision Making under Uncertainty*. International Series in Operations Research and Management Science 978-3-319-41613-7. Springer, Dec. 2017. ISBN: ARRAY(0x58a79930). DOI: 10.1007/978-3-319-41613-7. URL: https://ideas.repec.org/b/spr/isorms/978-3-319-41613-7.html (cit. on p. 26).

[9] Kjetil Høyland and Stein W Wallace. «Generating scenario trees for multistage decision problems». In: *Management science* 47.2 (2001), pp. 295–307 (cit. on p. 27).

[10] P. Brandimarte. *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction*. Statistics in Practice. Wiley, 2013. ISBN: 9781118625576. URL: https://books.google.it/books?id=iH9ltZtOsM4C (cit. on p. 27).

[11] Francesca Maggioni and Stein W Wallace. «Analyzing the quality of the expected value solution in stochastic programming». In: *Annals of Operations Research* 200 (2012), pp. 37–54 (cit. on p. 29).

[12] John C Hull. *Options, futures, and other derivatives / John C. Hull*. eng. 8th global ed.. Boston [etc.: Pearson, 2012. ISBN: 978-0-273-75907-2 (cit. on pp. 29–32).

[13] Tomas Bjork. *Arbitrage Theory in Continuous Time*. 3rd ed. Oxford University Press, 2009. URL: https://EconPapers.repec.org/RePEc:oxp:obooks:9780199574742 (cit. on p. 30).

[14] Alan Delgado de Oliveira, Tiago Pascoal Filomena, and Marcelo Brutti Righi. «Performance comparison of scenario-generation methods applied to a stochastic optimization asset-liability management model». In: *Pesquisa Operacional* 38 (2018), pp. 53–72 (cit. on p. 32).

[15] P. Brandimarte. *Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics*. Wiley Handbooks in Financial Engineering and Econometrics. Wiley, 2014. ISBN: 9781118594513. URL: https://books.google.it/books?id=d-yuBgAAQBAJ (cit. on pp. 33–39, 41).