HEC MONTRÉAL
École affiliée à l'Université de Montréal

Solving Optimal Portfolio Choice Problems with Predictable Returns by Dynamic Programming Methods

par
Siyang Wu

Thèse présentée en vue de l'obtention du grade de Ph.D. en administration
(option Ingénierie financière)

le 1 juin 2018

HEC MONTRÉAL
École affiliée à l'Université de Montréal

Cette thèse intitulée :

Solving Optimal Portfolio Choice Problems with Predictable Returns by Dynamic Programming Methods

Présentée par :

Siyang Wu

a été évaluée par un jury composé des personnes suivantes :

Résumé

Dans cette thèse, nous étudions les problèmes de choix de portefeuille optimal en temps discret, donc dans une marché incomplète. Nous donnons une introduction générale dans la partie I.

Dans la partie II, nous développons un algorithme de programmation dynamique vers l'avant pour résoudre les problèmes de choix de portefeuille optimal. Cette méthode itérative est basée sur des simulations et offre ainsi une grande flexibilité pour la modélisation des rendements des actifs. Cette méthode avancée est vers l'avant dans le sens que le choix de la décision dans chaque scénario est effectué depuis l'instant 0 jusqu' la fin de l'horizon, par opposition aux méthodes traditionnelles de programmation dynamique par l'arrière (BDP). Notre méthode réduit considérablement le fardeau de calcul des méthodes typiques de BDP car il n'y a pas besoin de discrétiser l'espace d'état ni l'espace de décision/action. L'algorithme résultant est adapté aux problèmes où le nombre d'actifs dans le pool d'investissement est important. En outre, il existe des indications selon lesquelles la méthode pourrait être naturellement étendue pour inclure d'autres caractéristiques telles que les coûts de transaction et la consommation intertemporelle.

Dans la troisième partie de cette thèse, nous développons un algorithme de programmation dynamique par l'arrière pour résoudre le problème de choix de portefeuille optimal dans le contexte des rendements non-gaussiens (la distribution de Johnson-S_U), une caractéristique prédominante des actifs financiers. Nos tests numériques avec des données historiques suggèrent que les politiques optimales sont significativement différentes selon les deux hypothèses (la distribution de Johnson-S_U et la distribution gaussienne) pour les rendements d'acitif. Notre contribution principale est d'explorer comment les rendements journaliers non-gaussiens peuvent modifier la décision d'un investisseur typique lorsque les rendements sont prévisibles.

Mots clés: choix de portefeuille optimal, sélection optimale de portefeuille, programmation dynamique approximative, distribution Johnson-S_U, VAR(1), CRRA, programmation dynamique vers l'avant.

Méthodes de recherches: analyse numérique, recherche quantitative.

Abstract

In this thesis, we study optimal portfolio choice problems in discrete time, thus in an incomplete market economy. We give a general introduction in Part I.

In Part II, we develop a forward dynamic programming algorithm to solve optimal portfolio choice problems. This iterative method is based on simulations and thus offers great flexibility for the modelling of assets' returns. It is a forward method in that the choice of decision for any scenario is effectively done from time 0 to the end of horizon, in opposition to traditional backward dynamic programming (BDP) methods. Our method reduces significantly the computational burden of typical BDP methods since there is no need to discretize the state space nor the decision/action space. The resulting algorithm is suitable for problems where the number of assets in the investment pool is large. Furthermore, there are indications that the method could naturally be extended to include other features such as transaction costs and inter-temporal consumption.

In Part III of this thesis, we develop a backward dynamic programming algorithm to solve the optimal portfolio choice problem in the context of non Gaussian returns, a prevalent feature of financial assets. Our numerical tests with historical data suggest that the assumption of Johnson-S_U distribution leads to quite different optimal policies than the case with Gaussian returns. Our major contribution is to explore how non-Gaussian log-returns can alter a typical investor's decision when returns are predictable.

Key Words: optimal portfolio choice, optimal portfolio selection, approximate dynamic programming, Johnson-S_U distribution, vector auto-regressive, CRRA, forward dynamic programming.

Research methods: numerical analysis, quantitative research.

Contents

List of Figures

List of Tables

List of Notation

T	:	investment horizon, at which date we evaluate the utility of final wealth
N_a	:	number of risky assets
$\mathcal{A}$	:	the set of available risky asset, $\lvert\mathcal{A}\rvert = n$
$x_{t,i}$	:	proportional allocation (or portfolio weights) in asset i at time t for the time period $[t, t+1)$, $\boldsymbol{x}_t = (x_{t,i})_{i\in\mathcal{A}}$
W_t	:	total wealth at the beginning of the period $[t, t+1]$
$u(\cdot)$	:	utility function
R_f	:	risk-free return
$R^g_{t,i}$	:	gross return during period $[t-1, t]$ of risky asset i
$R_{t,i}$	:	excess return (over risk-free rate) during period $[t-1, t]$ of risky asset i, defined as the difference between the return on asset a and the risk-free rate, $\boldsymbol{R}_t = [R_{t,1}, \ldots, R_{t,N_a}]$
c_t	:	consumption for period $[t, t+1)$
$r^e_{t,i}$	:	**log** excess return of risky assets in period [t-1,t) where $R_{t,i} = e^{r^e_{t,i}} - 1$, and $\boldsymbol{r^e_t} = [r^e_{1,t},\ \ldots,\ r^e_{N_a,t}]$
δ_t	:	the log dividend yield of a stock index
$\mathcal{D}_t$	:	grid for the dividend variable δ_t
$\mathcal{R}_n$	:	the real coordinate space of n dimension
$\boldsymbol{y}$	:	the state vector of the restricted VAR(1) including assets' returns and the log dividend yield
m_q, p_q	:	Gauss-Hermite quadrature nodes and weights, $q = 1, \ldots, Q$
Σ	:	covariance matrix of $\boldsymbol{y}$
Ω	:	Cholesky decomposition of Σ
$\boldsymbol{C_z}$	:	correlation matrix of Gaussian noises
Λ	:	Cholesky decomposition of $\boldsymbol{C_z}$
$\boldsymbol{C_u}$	:	correlation matrix of Johnson-S_U noises
$\mathbf{z}_t$	:	exogenous state variables in period $[t-1, t]$

$V_t(\cdot)$	:	(pre-decision) value function at time t
$V_t^x(\cdot)$	:	post-decision value function at time t
$\widetilde{V}_t(\cdot)$	:	approximation of post-decision value function
S_t	:	pre-decision state variable of the dynamic program
S_t^x	:	post-decision state variable of the dynamic program
$\mathcal{G}_t$	:	the set of grids for state variables at time t

List of Abbreviations

The following is a reference list of abbreviations in this thesis.

BDP	Backward dynamic programming
FDP	Forward dynamic programming
OPCP	Optimal portfolio choice problem
CRRA	Constant relative risk aversion
iid	Independent and identically distributed
GH	Gauss-Hermite quadrature

Remerciements

Tout d'abord, je voudrais exprimer ma sincère gratitude à mon directeur de thèse, le professeur Michel Denault, pour m'avoir initié à la recherche, pour le soutien continu de mon doctorat et de mes recherches, pour sa patience, sa motivation, son enthousiasme connaissance. Ses conseils m'ont aidé tout au long de la recherche et de l'écriture de cette thèse.

Je voudrais remercier mon codirecteur de thèse, le professeur Jean-Guy Simonato, pour ses encouragements, ses commentaires perspicaces, ses conseils et ses questions difficiles.

Mes sincères remerciements vont également à l'IFM2, pour les soutiens financiers. Je remercie mes collègues de KPMG pour leurs encouragements et leurs accommodements. De plus, je remercie tous mes amis de Montréal pour tout le plaisir que nous avons eu dans le pass.

Dernier point mais non le moindre, je voudrais remercier ma famille: ma mère, Yu Wu, pour m'avoir donné naissance à la première place. Avec mon beau-père Yuxin Shu, ils m'ont soutenu spirituellement tout au long de mes années de doctorat. Mention spéciale Cheng Jiang pour son soutien et sa compréhension durant la dernière année de mon PhD. Ce sont les personnes les plus importantes dans mon monde et je leur dédie cette thèse.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my thesis director Prof. Michel Denault for enlightening me the first glance of research, for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to thank my thesis co-director Prof. Jean-Guy Simonato, for his encouragement, insightful comments and guidance, and hard questions.

My sincere thanks also goes to IFM2, for the financial supports. I thank my colleagues at KPMG, for their encouragement and accommodation. Also I thank all my friends in Montréal for all the fun we have had in the past.

Last but not the least, I would like to thank my family: my mom, Yu Wu , for giving birth to me at the first place. Together with my stepfather Yuxin Shu, they supported me spiritually throughout my PhD years. Special mention to Cheng Jiang for being so supportive and comprehensive during the last year of my PhD. They are the most important people in my world and I dedicate this thesis to them.

Part I

General Introduction

The problem of portfolio selection and management with rebalancing has been intensively studied in the financial economics literature after the pioneering works of [40, 41, 42] and [53]. This problem is a mathematical framework for assembling a portfolio of assets such that the investor's objective is attained according to a certain criterion. Typically the criterion must take both assets' returns and risks into account while the objective is to maximize the expected utility of final wealth after a number of time periods. The optimal policy to this problem strikes a balance between risk and reward and both in the present and the future. The investor benefits from a properly structured investment plan based on his/her knowledge from the past. This problem has many names in the literature, such as "modern portfolio theory", "optimal portfolio choice", "portfolio selection", "optimal portfolio allocation", "portfolio optimization" or "intertemporal portfolio choice" to name a few. While there is no consensus for the nomenclature, we use "optimal portfolio choice problem" (OPCP) for the purpose of this thesis.

Recent academic research on OPCP has focused on the identification of key aspects of real-word OPCPs and to understand both qualitatively and quantitatively their role in the optimal investment decisions of individuals and institutions. To find realistic solutions to OPCPs, a critical step is to relate the theoretical formulation of the problem and its solution to the data. There are several ways to accomplish this task, yet no single approach has emerged as a clear favourite. Each approach has its advantages and disadvantages, and an approach favoured in one context is often less attractive in another.

An OPCP can be formulated either in continuous or discrete time. In the continuous time formulation, it is assumed that the portfolio can be balanced at every single instant. Its advantage is the analytical tractability – closed-form solutions can be derived by the use of stochastic calculus for problems which are analytically intractable in discrete time. However, the real world seems to be more in concordance with a discrete time modelling because a portfolio cannot be rebalanced at every instant of time. Even though the continuous-time Bellman equation is the limit of its discrete-time counterpart, the solutions to the two problems are different. The reason is that the continuous-time optimal policies are not necessarily admissible in discrete time due to possible negative wealth [6]. One may argue that continuous time modelling can be approximated by trading at extremely high frequency, but the resulting transaction costs due to liquidity problems (or market depth) may be exceptionally large thus considerably degrading a trading strategy's performance.

In the discrete time formulation, portfolio weights are rebalanced only at a finite number of decision moments. At each decision moment, the optimal allocation depends in general on the level of state variables (e.g. amount of wealth, past asset returns, etc.) at the beginning of current time period. Asset returns within the current time period are observed, resulting in a new amount of wealth for the next decision moment. This process is repeated until the end of the investment horizon. A favourite method for solving an OPCP in discrete time is dynamic programming, a method which takes future decision-making into account while making decisions for the present. In traditional dynamic programming, optimal policies are calculated backward: the last period decision rule, contingent on available wealth and the realizations of all previous periods' asset returns, is devised in advance using the terminal condition; then the next-to-last period's decision rule is devised, taking into account how the results of this period will influence the final period's decisions; and so forth backward in time. This procedure quickly becomes very complex if there are more than a few time periods or more than a few assets.

In this thesis, we focus on two methods to solve OPCPs in the discrete time formulation with finite investment horizon and where the investor's preferences are characterized by the CRRA utility function.

In Part II, we develop a forward dynamic programming (FDP) algorithm. This is a forward method in that the choice of decision for any scenario is effectively done from time 0 to the end of horizon, in opposition to traditional backward dynamic programming (BDP). The resulting algorithm is iterative and based on simulations and thus offers a great flexibility for modelling assets' returns. Value function approximations are used to make decisions in every iteration. In the first iteration, decisions can be very far from the optimal ones since the initial approximations are arbitrarily selected and thus represent a poor description of the true value functions. Even so, the information collected within the current iteration will be used to correct and improve the quality of these approximations. The updated approximated value functions are used in the next iteration for decision-makings; and so on back and forth until the approximations are stable or the convergence of optimal weights is obtained.

Compared to BDP, the FDP offers comparable precision in portfolio weights while reducing the computational burden by avoiding the trouble of discretizing in both the state and action space. Our method dissociates the decision process from the return generation process. In other words, the simulation-based algorithm remains the same regardless of the underlying return distribution. Assets' returns are considered as exogenously given and fed to the algorithm as input. The whole decision-making algorithm is independent from the way that assets' return are simulated. Finally, there are indications that the method could naturally be extended to include other features such as transaction costs and inter-temporal consumption.

In Part III, we follow the main stream of methodology in the literature – backward dynamic programming (BDP). We put the commonly used Gaussian distribution for assets' return into question and develop a quasi-analytic BDP method through the use of Gauss-Hermite quadrature for OPCPs where returns are predictable and follow a restricted vector auto-regressive (VAR) process with Johnson-S_U noises.

Parts II and III of this thesis each opens with an introduction and literature review specific to that part in order to highlight our motivations, followed by descriptive sections of the methods, sections presenting numerical results, and a discussion of managerial impacts. The final part of this thesis concludes and outlines our contributions to the literature.

Part II

Optimal Portfolio Choice Problems with Forward Dynamic Programming

In Part II of this thesis, we present our forward dynamic programming approach to solve OPCPs where the portfolio may be rebalanced at finite decision moments. The objective of our representative investor is to maximize the utility of his final wealth at the end of the investment horizon. In this simulation-based iterative approach, decisions are made from the starting point forth and updates of value functions are completed backward in time. The algorithm relies on appropriate value function approximations and a reliable updating procedure. It starts with a relatively poor estimation of value functions and learns from trial-and-error through realistic simulations of risky assets' returns. The information acquired is used to correct the estimated value function. The algorithm stops when the pre-defined criteria are satisfied.

This part is structured as follow. Chapter 1 consists of a literature review on OPCPs in discrete time formulation. This chapter also brings out the motivation of our FDP approach. Chapter 2 sets up the theoretical formulation of multi-period OPCPs with utility maximization in discrete time. The section opens with arbitrary utility functions and then focuses on the particular case of CRRA utility function. Chapter 3 is dedicated to our forward approach. It begins by establishing the theoretical foundation of the FDP before presenting the very algorithm and various discussions on the algorithm's parameters. Numerical examples are included in Chapter 4 to illustrate the performance and computation time of FDP compared to the classical BDP approach by Gauss-Hermite quadrature. This section also contains a sensitivity analysis of FDP algorithm parameters. Finally, Chapter 5 summarizes the principle features of the FDP approach and highlights our contributions to the literature.

Chapter 1

Literature Review for Part II

The first section is devoted to a brief review of methods used in the literature to solve OPCPs with a focus on the intertemporal expected utility maximization. The second section examines the existing literature of forward dynamic programming (FDP), also known as *approximate dynamic programming* which finds its origin in *reinforcement learning* or *machine learning*. The last section consists of a review of applications of FDP on finance related fields.

1.1 The Optimal Portfolio Choice Problem in Discrete Time

A first approach to formulate the optimal portfolio choice problems in discrete time is the mean-variance paradigm of [40]. This paradigm, capable of capturing the two fundamental aspects of portfolio choice – diversification and the trade-off between expected return and risk – is currently the most common formulation in industry. However, despite its simplicity and analytical form, this paradigm has its limits. For instance, it is a myopic single-period problem while most investment problems involve longer horizons with intermediate portfolio rebalancing. Furthermore, the mean-variance formulation, only representing the special case of quadratic utility, ignores any preferences toward higher-order return moments [6]. This ignorance is not desirable according to recent literature. For example, [27] show that systematic skewness of asset returns is economically important and recommands a risk premium of 3.6% on average per year. Similar conclusions were drawn in [44], where the authors find that empirically investors sacrifice mean-variance efficiency for higher skewness exposure.

The limits of the mean-variance framework lead to the second formulation, the intertemporal expected utility maximization. Its dynamic nature suggests the use of dynamic programming to solve multi-period OPCPs in discrete time. Unfortunately, closed-form solutions are available only for a few special cases. The recent literature has been focused on the development of various numerical and approximate methods to incorporate realistic features of the investor's preferences and return dynamics. Some authors assume simple, sometimes even unrealistic, return distributions or particular utility function forms, and perform various expansions of the value function for which the problem can be solved analytically. For example, [12] log-linearize the first-order conditions (FOCs) and budget constraint. [33] identify the equilibrium in multi-agent economies by asymptotic analysis and assuming power utilities. [20] substitute a general polynomial form for the value function into the FOCs to solve for optimal controls. Other authors prefer the discretization approach which requires firstly discretizing over the state space, and evaluating the value function at each point on the grid by a choice of quadrature integration [2], simulations [3], binomial discretizations [19], nonparametric regressions [5], or Taylor expansion [25, 26]. Then the value function over the entire state space is approximated by interpolation and the dynamic optimization can be solved by backward recursion.

However, the discretization approach suffers from the so-called "curse of dimensionality" when the problem involves more than a few number of state variables. Recent research has been focused on the development of simple, precise, and efficient algorithms to deal with potentially large-scale problems with path-dependency or non-stationary dynamics. For this purpose, the simulation-and-regression method, pioneered by [34] for pricing American-style options, has lately received particular attention from for example [7, 22, 21]. The idea is to repeat a two-step procedure beginning from the last period T where we know the exact form of the objective function. These algorithms involve simulating a set of sample paths of state variables in a first step, and then regressing the realized values over a set of basis functions to obtain approximate value functions. The algorithm in [7] relies on the maximization of a Taylor expansion of the investor's expected utility where conditional moments are computed with least squares regression of the realized utility and its derivatives on basis functions. [22, 21] propose to regress over the decision space as well (also known as "Q-learning"). This approach is interesting since it avoids the overhead work associated with deriving the first four derivatives of the value function, unlike the Taylor series based approach. Moreover, it can also handle functions that are not differentiable. The authors show with numerical examples that only relatively few discretization points on the decision space are needed to obtain good precision.

In Part II of this thesis, we develop an algorithm following the idea of regression over the decision space, similar to in [22] except for solving the problems with forward instead of backward recursion. The next subsection provides a brief review on the development of forward dynamic programming and its applications.

1.2 The Origin of Forward Dynamic Programming

The ideas and techniques of forward dynamic programming find their roots in reinforcement learning (or machine learning) from the optimal control community or neuro dynamic programming from the artificial intelligence community. [4] treated the theoretical aspects in the language of control theory. Later, [55] describe the field from the perspective of artificial intelligence and computer science community. And recently, [51] tells the story to the operational research community. This family of algorithms uses forward iterations to solve a forward version of Bellman's equation. Although many forward algorithms are highly related to their backward twin, for many problems the forward approach is more convenient or sometimes even the only feasible one, especially when the final state set is unknown. Scientists from different communities appear to be working in different worlds and have their own terminology unique to themselves for the same technique. The book of [51] does mention some connections between those related fields.

As their backward twin, FDP algorithms are often used to solve a Markov decision process (MDP) which are used in a wide area of disciplines including robotics, automated control, economics, finance and manufacturing. A core body of research on Markov decision processes results from the book of [28]. In the context of solving a MDP, a typical FDP algorithm starts from time zero. The decision maker chooses an action based on all available information he possesses up to that instant and from his current state. Then he receives an observation of the exogenous environment including a random realization (which leads the system transit to the next state), and a reward (which is then used to update his beliefs on the environment based on which next actions will be taken). It is worth to note that a key advantage of FDP algorithms is that the knowledge of explicit specification of the transition probabilities is not required. In contrast, this knowledge is required with BDP algorithms. In other words, if transition probabilities are unknown in a particular problem, as long as they can be observed or accessed through an exogenous simulator, forward algorithms can be applied. FDP algorithms are also suitable for on-line problems where decisions are taken in a serial fashion or in the order that the input is fed to the algorithm, without having the entire input available from the start.[1]

The idea of proceeding forward can be further combined with function approximations to address problems with a very large number of states. Popular choices include piecewise linear approximations, basis function approximations, etc. Common FDP algorithms include for example Q-learning, temporal difference learning. There exists a rich literature for FDP algorithms[2]. For example, in the control community, [48] study policy evaluation algorithms using linear function approximation for an infinite-horizon problem. In the operational research community, [14] examine the portfolio selection problem in the mean-variance framework and provide an algorithm using Bayesian inference to accommodate qualitative input about expected returns. [52] use basis functions to approximate value of states and a Bayesian belief structure to represent uncertainty about unknown values, based on the knowledge gradient concept from the optimal learning

[1]When the algorithm's performance is compared to that where an agent acts optimally from the beginning (an *offline* algorithm), the difference in performance gives rise to the notion of regret.

[2]We refer the readers to [55, 50, 51] for a more complete introduction and available FDP algorithms.

literature.

In summary, two features make FDP algorithms powerful: the use of samples to optimize performance and the use of function approximations to deal with large-scale environments.

1.3 Solving OPCPs by FDP: Motivations

In Part II of this thesis, we propose a FDP algorithm to solve multi-period OPCPs in discrete time within the maximization of expected utility of an investor's final wealth. Decisions are taken progressively forward in time while updates of approximate value function are completed backwardly. The decision process is totally independent from the simulation process of exogenous state variables.

Our approach bypasses the need of any form of expansion, thus can handle more general form of utility functions. The simulation-based recursive algorithm does not require any discretization neither on the state space nor on the decision space. It is therefore a potential candidate algorithm for large-scale problems. Moreover, our numerical results show that the approach offers better precision within less computation time as compared to a BDP approach with discretization and Gauss-Hermite quadrature.

The rest of Part II is organized as follows. Chapter 2 describes the dynamics of the returns and portfolio problem examined in this study. Chapter 3 is the core chapter of this part. It presents the detailed FDP approach. This chapter opens with an introduction (Subsection 3.1) presenting a scketch and an illustrative example of the FDP method, followed by a description of the theoretical framework (Section 3.2). Section 3.3 offers algorithms in the general case (3.3.1) and in the particular case where the investor's preferences can be characterised by the CRRA utility function (3.3.3 for interdependant assets' returns and 3.3.4 for iid assets' returns). This core section ends with a discussion of parameter selection. Chapter 4 examines numerical results, and Chapter 5 concludes.

Chapter 2

The Optimal Portfolio Choice Problem

This chapter first describes the general formulation of an OPCP with a generic utility function before presenting the set-up of the particular case studied in this project i.e. with the CRRA utility. Problem set-ups for two specific dynamics of assets' returns (iid and VAR(1)) within the CRRA utility framework are presented.

2.1 General Formulation

The investment opportunity set consists of N_a risky assets and a risk-free asset (cash). The return of the risk-free asset is denoted R_f which is assumed to be known and constant over time. The excess return of risky assets are denoted by $\boldsymbol{R}_t = (\hat{R}_{t,1}, \ldots, \hat{R}_{t,N_a})$, where $\hat{R}_{t,k}$, for $k = 1, \ldots, N_a$ is the excess return over the risk-free asset of asset i for the period $(t-1, t]$. We define the excess return as the difference between the gross return of risky assets ($\boldsymbol{R}_t^g$) and the risk-free asset: $\boldsymbol{R}_t := \boldsymbol{R}_t^g - R_f$. We will allow the possibility that assets' returns exhibit some degree of predictability. To model this, let $\boldsymbol{z}_t$ be a vector of exogenous information. One can view $\boldsymbol{z}_t$ as a vector of observable market state variables influencing risky assets' returns $\boldsymbol{R}_{t+1}$.

Our investor is provided with initial wealth W_0 at time $t = 0$ and wishes to maximize the expected utility of his terminal wealth at date T through a series of portfolio rebalance strategies among risky assets and the risk-free security at times $t = 0, 1, \ldots, T-1$. There are no transaction costs nor taxes, and no intermediate consumption. With these assumptions, the investor's OPCP can be expressed as

$$V_0(W_0, \boldsymbol{z}_0) = \max_{\{\boldsymbol{x}_t\}_{t=0}^{T-1}} \mathbb{E}_0 \left[u(W_T) \right], \tag{2.1}$$

with the following self-financing constraints at all t:

$$W_{t+1} = W_t \left(\boldsymbol{x}_t \boldsymbol{R}_{t+1}(\boldsymbol{z}_t) + R_f \right), \tag{2.2}$$

where $\boldsymbol{x}_t$ is the vector of percentage allocation of the investor's wealth in risky assets at time t, $\mathbb{E}_0$ denotes the expectation conditional on the information available at time $t = 0$, which is in this case, the initial wealth W_0 and the initial exogenous states value $\boldsymbol{z}_0$, and $u(\cdot)$ is a non-decreasing concave utility function describing the investor's risk preferences. In the previous equation, asset returns are expressed as a function of the exogenous state variables to emphasize the inherent dependence of $\boldsymbol{R}_{t+1}$ on $\boldsymbol{z}_t$.

At each rebalancing moment t, an investor allocates his total wealth W_t into the set of available assets by deciding the percentage amount $x_{t,i}$ to invest (the decision variable) in asset i. The vector $\boldsymbol{x}_t = [x_{t,1}, \ldots, x_{t,N_a}]$, where $\boldsymbol{x}_t^T \mathbb{I} \leq 1$ is the portfolio allocation in risky assets immediately after decisions at time t are taken but before new returns for the next period $t+1$ become available.

According to the Bellman principle, solving the equation (2.1) above is equivalent to solving the so-called Bellman equation

$$V_t(\boldsymbol{S}_t) = \max_{\boldsymbol{x}_t} \mathbb{E}_t \left[V_{t+1}(\boldsymbol{S}_{t+1}) \right], \tag{2.3}$$

subject to the self-financing constraint (2.2) where $\boldsymbol{S}_t = (\boldsymbol{z}_t, W_t)$ is the vector of state variables of this dynamic program problem. The Bellman equation (2.3) defines the function V_t which represents the best value one can achieve from time t to the end of the investment horizon given the current state $\boldsymbol{S}_t$. The value function at the end of the investment horizon is assumed to be the utility of terminal wealth[1]: $V_T(\boldsymbol{S}_T) = V_T(\boldsymbol{z}_T, W_T) = u(W_T), \forall \boldsymbol{z}_T$.

Conditional expectations in (2.3) are taken over the random asset returns $\boldsymbol{R}_{t+1}$ and the next period information state $\boldsymbol{z}_{t+1}$. We assume all expectations are well defined and there is no opportunity of arbitrage in the market, a sufficient condition for the OPCP to have a unique solution (see for example the "Portfolio Choice Theorem" in Section 9.7 of [35]). Following the Proposition 2.1 in [10], the value function $V_t(\boldsymbol{z}_t, \boldsymbol{x}_t, W_t)$ is non decreasing in total wealth W_t and jointly concave in $(\boldsymbol{x}_t, W_t)$ for any market state $\boldsymbol{z}_t$. Therefore, the optimization problem (2.3) is convex, as the maximization of a concave function over a convex set.

The next section presents the particular case studied in this project where the investor's risk preferences can be characterized by the constant relative risk aversion (CRRA) utility function.

2.2 OPCPs with CRRA Utility

Consider the constant relative risk aversion (CRRA) utility function:

$$u(W_T) = \frac{W_T^{1-\gamma}}{1-\gamma}, \gamma > 1.$$

It is well known in the OPCP literature, e.g. [7, 25], that the homotheticity of CRRA utility implies the fact that optimal portfolio weights are independent of the level of wealth. This property of the CRRA utility function allows to remove the variable W_t

[1]We could have instead defined the final wealth in terms of the liquidation value of the portfolio, including the transaction costs associated with liquidation.

from the list of state variables, yielding the *reduced* Bellman equation

$$\mathcal{V}_t(\boldsymbol{z}_t) = \max_{\boldsymbol{x}_t} \mathbb{E}_t \left[(R_f + \boldsymbol{x}_t \boldsymbol{R}_{t+1}(\boldsymbol{z}_t))^{1-\gamma} \mathcal{V}_{t+1}(\boldsymbol{z}_{t+1}) \right], \tag{2.4}$$

with terminal condition $\mathcal{V}_T(\boldsymbol{z}_T) = 1/(1-\gamma), \forall \boldsymbol{z}_T$. In equation (2.4), it is assumed that assets' returns at time $t+1$ can be predicted by the vector of information $\boldsymbol{z}_t$. Two specific return dynamics will be studied in this project. In the first case, assets' returns follow a vector autoregressive (VAR) process of order 1, and in the second case assets' returns are assumed to be independent and identically distributed (iid). The next two subsections are dedicated to the description of the problem setups with these two specific return dynamics respectively.

2.2.1 Predictable Asset Returns

Let's first consider a problem similar to the one examined in [56, 26] and in [21]. In this problem, it is assumed that the excess returns are predictable with a dynamics given by the following 1^{st} order vector autoregressive process VAR(1):

$$\boldsymbol{R}_{t+1} = A_0 + A_1 \boldsymbol{R}_t + \boldsymbol{\xi_{t+1}},$$

where A_0 and A_1 are matrices of size N_a-by-1 and N_a-by-N_a respectively for constant parameters, and $\boldsymbol{\xi_{t+1}}$ is a vector of Gaussian error terms with constant covariance matrix Σ. This model is the multidimensional version of the AR(1) process and it captures linear interdependencies among asset returns. The exogenous state variables $\boldsymbol{z}_t$ introduced in the last section are the 1-period lagged returns $\boldsymbol{R}_t$, which influence assets' returns in the next period $\boldsymbol{R}_{t+1}$ through the VAR(1) dynamic.

The Bellman equation (2.4) for the OPCP with CRRA utility and VAR(1) return dynamic can be expressed as

$$\mathcal{V}_t(\boldsymbol{R}_t) = \max_{\boldsymbol{x}_t} \mathbb{E}_t \left[(R_f + \boldsymbol{x}_t \boldsymbol{R}_{t+1}(\boldsymbol{R}_t))^{1-\gamma} \mathcal{V}_{t+1}(\boldsymbol{R}_{t+1}(\boldsymbol{R}_t)) \right] \tag{2.5}$$

under the self-financing constraint (2.2) and with the terminal condition

$$\mathcal{V}_T(\boldsymbol{R}_T) = 1/(1-\gamma), \forall \boldsymbol{R}_T.$$

2.2.2 IID Assets Returns

Now let's consider the case as studied by [25] where risky assets' returns are assumed to be independent and identically distributed from one period to another. In this case, no exogenous state variable need to be modelled because returns are independent thus cannot be predicted. The Bellman equation (2.4) simplifies to

$$\mathcal{V}_t = \max_{\boldsymbol{x}_t} \mathbb{E}_t \left[\left(R_f + \boldsymbol{x}_t \boldsymbol{R}_{t+1} \right)^{1-\gamma} \mathcal{V}_{t+1} \right] \tag{2.6}$$

with constraint (2.2) and terminal condition $\mathcal{V}_T = 1/(1-\gamma)$. Notice that the value function at time t is independent of the physical state of the system and simplifies to a constant regardless the value of assets' returns or wealth level. As demonstrated in [25], an important feature of OPCPs with CRRA utility function and iid asset returns is that the multi-period problem reduces to a series of single period problems and the percentage allocations in risky assets are the same for all periods.

Chapter 3

Solving Portfolio Choice Problem Using FDP

This chapter presents our forward dynamic programming algorithm. In Section 3.1, we first motivate our forward approach by an example which illustrates the "curse of dimensionality" suffered by usual BDP approaches. Then in the same section, we provide a sketch of our proposed forward algorithm. Next, Section 3.2 presents the notions of post-decision state variables and post-decision value functions, which are one of the core parts in our forward algorithm. It will be shown that, through the use of post-decision value functions around post-decision state variables, the order of maximization and conditional expectation can be inverted in the usual Bellman equation, which further gives rise to the forward approach. Pseudocode of forward algorithms for the general case and two specific cases are presented in Section 3.3. Finally implementation details and parameter tuning are discussed in Section 3.4.

3.1 Introduction and Overview

The dynamic nature of multi-period OPCPs suggests the use of dynamic programming to solve it. DP is a well-known method widely used in many fields and relies on the Bellman principle. In a typical DP algorithm, the solution is obtained through backward recursion on a grid of state variables[1]. These BDP algorithms suffer from the well-known "curse of dimensionality" due to the need of discretization. As an illustration, assume there are 3 exogenous state variables ($\boldsymbol{z}_t$ is a 3-by-1 vector). If the domain of each state variable and of total wealth are discretized into a grid with 20 points in each dimension (which is a coarse grid), the state space would consist of $20^{3+1} = 160,000$ points in total. This means that, at each rebalancing moment t, we need to solve the optimization problem in (2.3) $160,000$ times. And in addition, for each of the possible states, we repeatedly have to compute numerically the conditional expectation of random returns in (2.3) with computationally expansive numerical methods[2].

As the BDP approach becomes very inefficient when the dimension of state variables exceeds a few, we propose a forward version of dynamic programming to solve OPCPs. The forward dynamic programming (FDP), springing from Artificial Intelligence or Reinforcement Learning, is a simulation-based algorithm and does not require any discretization. It is thus naturally a candidate for high-dimensional problems. In our proposed approach, the decision process is totally independent from the simulation process of exogenous state variables, which offers possibilities of easy extensions to include interesting features of assets' returns or the investor's risk preferences.

The recursive FDP algorithm we proposed in this project is composed of two passes – forward and backward as sketched in Figures 3.1 and 3.2. The rest of the current section draws an overview of our approach without getting into any technical details. Our intention here is to introduce the general idea instead of providing thorough explanation about each step in our algorithm which will be found in later sections of this chapter.

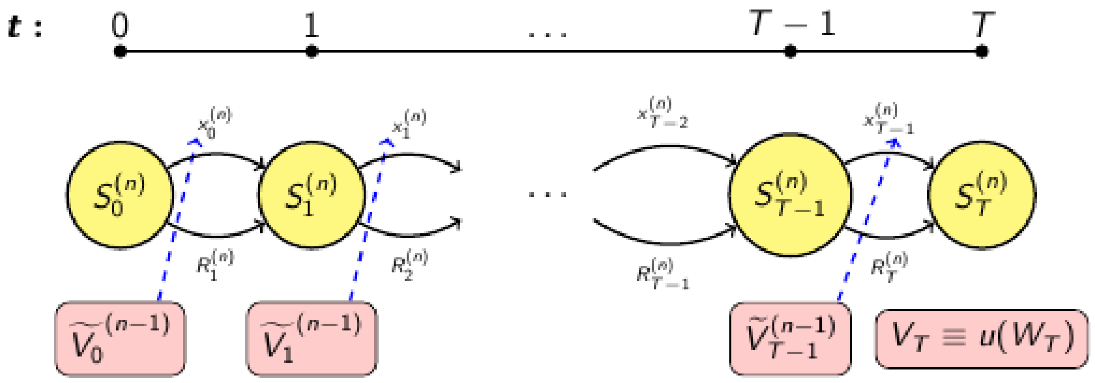
Our algorithm starts with a forward pass as illustrated in Figure 3.1. The number inside the parenthesis stands for the iteration number. Within iteration n, decisions are taken forward in time at each step $t = 0, 1, \ldots, T-1$ based on a set of value function approximations $\widetilde{V}_t^{(n)}(\boldsymbol{S}_t; \boldsymbol{x}_t)$ which represents one's best knowledge about the system. The value functions and its approximations usually depend on the state variable $\boldsymbol{S}_t$ and the decision variable $\boldsymbol{x}_t$. It is the modeller's responsibility to choose the appropriate form of approximations based on problem-specific features such as the utility function and the dynamics of the state variables. Examples of popular approximations include look-up tables, polynomial approximations, or neural network. Our choice of approximation will

[1]Please refer to Part III where we developed a BDP algorithm with GH quadrature to study OPCOs with non-Gaussian assets' returns. A literature review of BDP methods can also be found in this part.

[2]Of course this can be avoided if there exist analytical expressions for the conditional expectation, which is the case only in few particular cases for return distribution.

be presented later in section 3.3; in the current section, we invite the reader to visualise the value function approximations in an arbitrary form or just think of an approximation of his choice. In the rest of this thesis and where the context allows, we will omit the state variable inside the parenthesis and use $\widetilde{V}_t^{(n)}$ to denote the value function approximations within the n-th iteration.

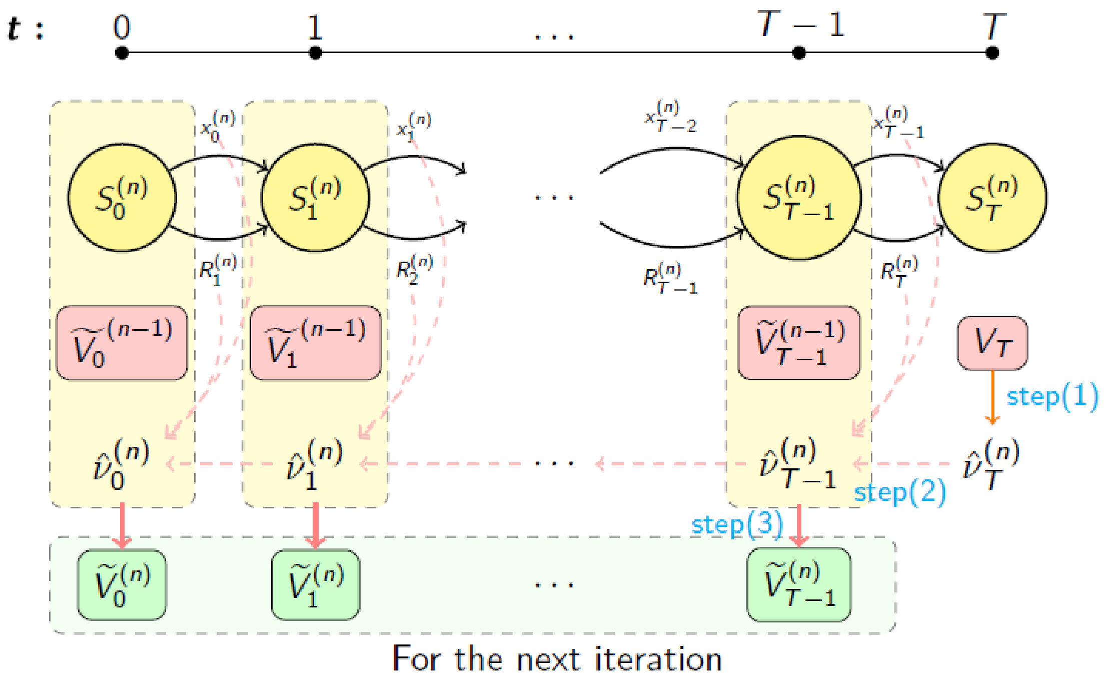
Figure 3.1: FDP algorithm illustration: forward pass



After the time-t decision is taken based on the value function approximations from the previous iteration $\widetilde{V}_t(\cdot)^{(n-1)}$ – a strategy called "exploitation" – a realization of the exogenous state variables for the next period $(t, t+1]$ is observed or simulated from an external simulator. The observed or simulated $\boldsymbol{z}_{t+1}^{(n)}$ produces a realization for assets' returns $\boldsymbol{R}_{t+1}^{(n)}(\boldsymbol{z}_{t+1}^{(n)})$, leading the system to the next state $\boldsymbol{S}_{t+1}^{(n)}$. Recall the state of the system is composed of the vector of exogenous variables which influence future assets' returns and the level of total wealth $\boldsymbol{S}_t^{(n)} := (\boldsymbol{z}_t^{(n)}, W_t^{(n)})$. This very procedure is repeated until the reach of the terminal time T, where the exact form of the value function $V_T(\cdot)$ is known thanks to the terminal condition.

Next, the algorithm enters a backward pass to update value function approximations as illustrated by Figure 3.2. Starting from the terminal time T, the realized value on current path is computed by $\hat{\nu}_T^{(n)} = V_T(\boldsymbol{S}_T^{(n)})$ (see step(1)). Then in step(2), the time $T-1$ realized value $\hat{\nu}_{T-1}^{(n)}$ is calculated backward in time by the Bellman principle and using the decision $\boldsymbol{x}_{T-1}^{(n)}$ made previously in the forward path, the simulated assets' returns $\boldsymbol{R}_T^{(n)}$ as well as the time T realized value $\hat{\nu}_T^{(n)}$ which is just obtained in step(1). Note that the time $T-1$ realized value is in fact a function of the decision $\boldsymbol{x}_{T-1}^{(n)}$, the state variable $\boldsymbol{S}_T^{(n)}$ and the next period realized value $\hat{\nu}_T^{(n)}$. We denote this relation by $\hat{\nu}_{T-1}^{(n)} = h(\boldsymbol{x}_{T-1}^{(n)}, \boldsymbol{S}_T^{(n)}, \hat{\nu}_T^{(n)})$, where $h(\cdot)$ is a generic function linking $\hat{\nu}_{T-1}^{(n)}$ to its arguments. The actual form of $h(\cdot)$ is problem specific since it depends on the underlying utility function and the dynamics of state variables. For the moment, we consider the most general case and do not specify $h(\cdot)$. Two particular cases with CRRA utility will be described in sections 3.3.3 and 3.3.4. Once $\hat{\nu}_{T-1}^{(n)}$ is obtained, it is used to update the value function approximations at $T-1$ (see step(3)). The procedure is continued by repeating steps $(2)-(3)$ for every decision moment t until time zero. The set of updated approximate value functions will be used in the next iteration for the decision-making processes in the forward path.

Figure 3.2: FDP algorithm illustration: backward pass



It is possible that the reader finds the description of our forward approach too general. This is normal at this point since the current section is intended to only provide an overall view of our forward approach without going into details. A detailed algorithm will be presented in Section 3.3.1 with a general form of utility function and arbitrary approximate value functions. Algorithms will be provided for two particular return dynamics with CRRA utility function in sections 3.3.3 and 3.3.4. Finally, Section 3.4 will cover a discussion on parameter tuning as well as implementation details. In preparation for these sections presenting the detailed algorithms, the next section introduces first the notion of post-decision, which is a key concept in our forward approach.

3.2 The Notion of Post-Decision

The term of post-decision state, essentially used by W. Powell and his co-authors [50, 51, 46, 52] may first appear unfamiliar to the readers, yet it is not a brand new notion. In fact, the post-decision state is equivalent to the state-action pair or augmented state in the reinforcement learning literature. It represents the "state" of the system once a decision $\boldsymbol{x}_t$ is taken but before the arrival of new exogenous information $\boldsymbol{z}_{t+1}$ (hence the name *post*-decision). The following definition states the post-decision state variables in the context of general OPCP formulation.

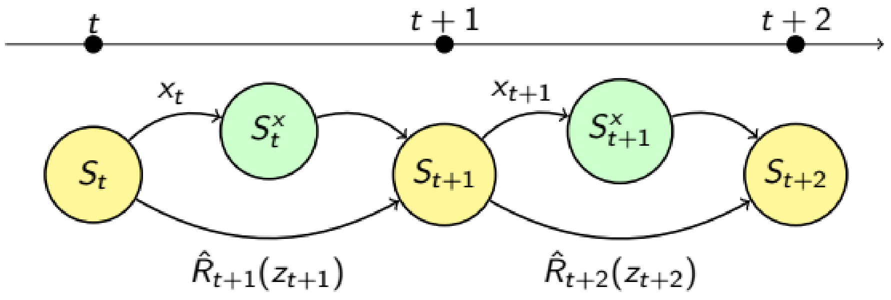
Definition 3.2.1. *The post-decision state variable is defined as the triplet of exogenous variables representing market information, the decision vector, and the endogenous state variable representing the total wealth at time t:*

$$\boldsymbol{S}_t^x \equiv (\boldsymbol{z}_t, \boldsymbol{x}_t, W_t). \tag{3.1}$$

The superscript x in previous definition indicates that the system is in the state after a decision x is made. The usual definition of state $\boldsymbol{S}_t = (\boldsymbol{z}_t, W_t)$ in BDP can be considered as the pre-decision state. The relation between post-decision states and pre-decision states is depicted by Figure 3.3: suppose that the system is in a pre-decision state $\boldsymbol{S}_t$ at time t. From this state, assuming the decision $\boldsymbol{x}_t$ is taken, then the system transits to the time-t post-decision state $\boldsymbol{S}_t^x$. Next, realizations of the exogenous market variables $\boldsymbol{z}_{t+1}$ and thus asset returns $\boldsymbol{R}_{t+1}$ for the period $(t, t+1]$ are observed or simulated, which leads the system to the next pre-decision state $\boldsymbol{S}_{t+1}$ according to the state transition equation (2.2).

Compared to Figures 3.1 and 3.2, the system transition within each time period is intentionally separated into 2 steps: first from $\boldsymbol{S}_t$ to $\boldsymbol{S}_t^x$ and then from $\boldsymbol{S}_t^x$ to $\boldsymbol{S}_{t+1}$. From Figure 3.3, it can be noticed that the portfolio composition is changed by new decisions $\boldsymbol{x}_t$ at time t, but the total wealth W_t remains unchanged. On the other side, the arrival of new market information $\boldsymbol{z}_{t+1}$ generates new assets' returns $\boldsymbol{R}_{t+1}$ which change the

Figure 3.3: Illustration of the *pre-post-pre* transition process



total wealth from W_t to W_{t+1} but not the portfolio composition $\boldsymbol{x}_t$. One should be aware that the post-decision states do not exist in reality and cannot be observed in a physical system; they are only auxiliary tools to illustrate the logic behind our forward approach.

Now we define the post-decision value function around the post-decision state:

Definition 3.2.2. *The post-decision value function at time t is defined as the conditional expectation of the value function at time $t+1$ given that the system is in the post-decision state $\boldsymbol{S}_t^x$:*

$$V_t^x(\boldsymbol{S}_t^x) \equiv \mathbb{E}_t\left[V_{t+1}(\boldsymbol{S}_{T+1})|\boldsymbol{S}_t^x\right]. \tag{3.2}$$

where $\boldsymbol{S}_t^x$ is defined in (3.1).

The post-decision value function is equivalent to the Q-factor in Q-learning **(author?)** [50]. In a similar way that the post-decision state is related to the pre-decision state, the post-decision value function is linked to the pre-decision value function in the following way:

$$V_t(\boldsymbol{S}_t) = \max_{\boldsymbol{x}_t} V_t^x(\boldsymbol{S}_t^x), \tag{3.3}$$

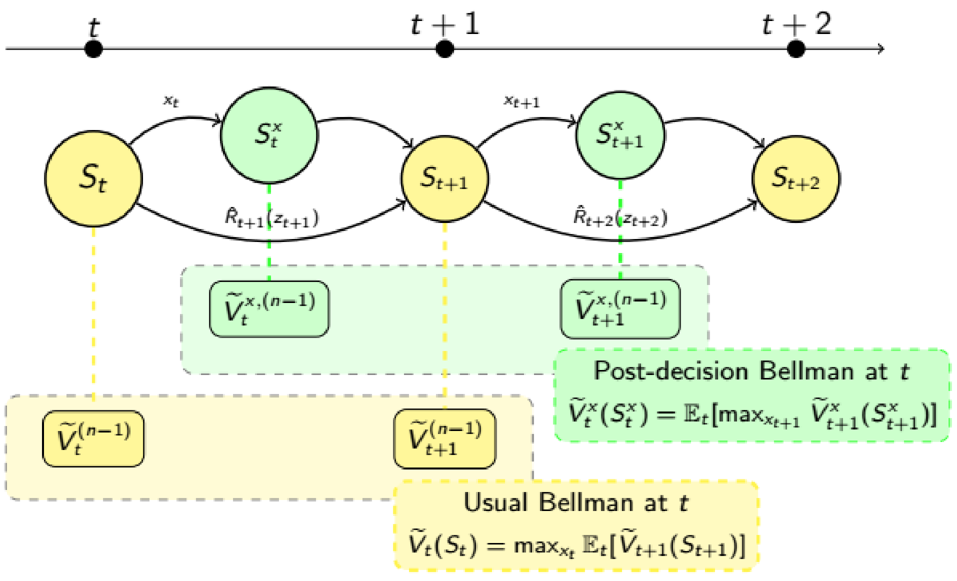
Equation (3.3) indicates an important distinction between the pre-decision and post-decision value functions: by construction, the post-decision value function represents the expected value at time t by taking a specific decision, which might or might not be optimal, whereas the pre-decision value function is the consequence of taking the **best** decision at time t due to the maximisation. In other words, the post-decision value function is decision-dependent, while the pre-decision value function is the "best" value of post-decision value function by taking the most appropriate decision.

Substituting equation (3.3) into equation (3.2) but for time $t+1$, we obtain the recursive Bellman equation around the post-decision state at time t:

$$V_t^x(\boldsymbol{S}_t^x) = \mathbb{E}_t\left[\max_{\boldsymbol{x}_{t+1}} V_{t+1}^x(\boldsymbol{S}_{t+1}^x)|\boldsymbol{S}_t^x\right]. \tag{3.4}$$

There are two key distinctions between the post-decision version (3.4) and the usual version of Bellman equation which is given in equation (2.3). Firstly, as illustrated by Figure 3.4, the post-decision Bellman equation establishes a recursive relation around the post-decision states, while the usual Bellman equation is around the pre-decision states. Secondly, in the post-decision version, the conditional expectation and maximisation operators are inverted compared to the usual version.

Figure 3.4: Illustration of the Post-decision Bellman Equation



t
$t+1$
$t+2$
x_t
x_{t+1}
S_t
S_t^x
S_{t+1}
S_{t+1}^x
S_{t+2}
$\hat{R}_{t+1}(z_{t+1})$
$\hat{R}_{t+2}(z_{t+2})$
$\widetilde{V}_t^{x,(n-1)}$
$\widetilde{V}_{t+1}^{x,(n-1)}$
$\widetilde{V}_t^{(n-1)}$
$\widetilde{V}_{t+1}^{(n-1)}$
Post-decision Bellman at t
$\widetilde{V}_t^x(S_t^x) = \mathbb{E}_t[\max_{x_{t+1}} \widetilde{V}_{t+1}^x(S_{t+1}^x)]$
Usual Bellman at t
$\widetilde{V}_t(S_t) = \max_{x_t} \mathbb{E}_t[\widetilde{V}_{t+1}(S_{t+1})]$

One notable implication of this inversion is that the maximisation problem in (3.4) is deterministic rather than stochastic as it is the case in the usual Bellman equation (2.3). As a matter of fact, it can be considered that the maximisation in (2.3) is made at the beginning of the period $[t, t+1)$, **before** new returns become available. Since assets' returns are assumed to be predicted by the exogenous market variables $\boldsymbol{z}_t$, the optimisation problem must take the conditional distribution of $\boldsymbol{z}_{t+1}$ given $\boldsymbol{z}_t$ into consideration, and thus is stochastic even if the functional form of $V_t(\boldsymbol{S}_t)$ is known. The situation is slightly different in the post-decision Bellman equation. Due to the inversion, the maximisation operation is done before the conditional expectation. Moreover, this maximisation problem can be considered to be made at the beginning of the period $[t+1, t+2)$ and thus **after** assets' returns $\boldsymbol{R}_{t+1}$ become available. Therefore, if $V^x_{t+1}(\boldsymbol{S}^x_{t+1})$ is known, this optimisation problem is deterministic, which can still be quite a challenging problem, but there is an extensive pool of efficient tools available to find a solution.

The next section presents three FDP algorithms. The first one is for the most general case with arbitrary choices of utility function and of approximated post-decision value function. The other two algorithms are designed for two particular cases of return dynamics with CRRA utility. In the two particular cases, post-decision value functions are approximated by a set of polynomial bases.

3.3 FDP algorithms

3.3.1 The Basic FDP Algorithm

In Section 3.1, we explained the general idea of our FDP algorithm without getting into technical details. This explanation was, however, in the language of the usual definition of state variables and value functions since the notion of post-decision had not been introduced yet. In our actual algorithm, the forward and backward passes are performed using the concept of post-decision value functions around post-decision states such as defined in Section 3.2. The goal is to obtain reliable estimates of these value functions for all post-decision states in order to make decisions optimally. However, if the state space is large (e.g. continuous state variables), it would be impossible to have an exact representation of the optimal value being in each state over the entire state space. A common technique is to approximate the post-decision value function by a certain functional form. We denote the approximation of $V^x_t(\boldsymbol{S}^x_t)$ introduced on page 22 by $\widetilde{V}^x_t(\boldsymbol{S}^x_t),\ t = 0, \dots, T-1$. A series of strategies for this purpose includes linear approximation, separable piecewise linear and concave approximation, and general linear regression models with non-linear base functions. The selection of function approximations should be problem-specific and based on the principle of parsimony.

Two important notational remarks merit some attention. First, we will omit the functions' arguments in the rest of this thesis if the context allows no ambiguity. Thus

$\widetilde{V}^{(n)}$ stands for $\widetilde{V}^{(n)}(S_t^{(n)})$ and $\widetilde{V}^{x,(n)}$ stands for $\widetilde{V}^{x,(n)}(S_t^{x,(n)})$. Second, notations with the superscript $^{(n)}$ denotes the sample realisation on the n-th path, while those without $^{(n)}$ has a general meaning. As an example, S_t^x represents the general post-decision state variable at time t. It is a random variable thus its value is arbitrary. While $S_t^{x,(n)}$ refers to the realisation of S_t^x on the n-th path. Its value is known after the time-t decision is made.

Figures 3.5 and 3.6 illustrate the forward and backward pass in our FDP algorithm with approximated post-decision value functions around post-decision states. Compared to Figures 3.1 and 3.2, the idea is the same except that a post-decision state $S_t^{x,(n)}$ is added between $S_t^{(n)}$ and $S_{t+1}^{(n)}$ and that the approximations are with respect to the post-decision value functions $\widetilde{V}^{x,(n)}$ rather than the pre-decision value functions $\widetilde{V}^{(n)}$.

In these figures, we suppose the exogenous state variables are the lagged assets' returns themselves $\boldsymbol{z}_t \equiv \boldsymbol{R}_t$. Within each iteration n, the forward pass (Figure 3.5) starts from $t = 0$. Given an initial state $S_0^{(n)}$, a decision $\boldsymbol{x}_0^{(n)}$ is made by maximizing the current post-decision value function $\widetilde{V}_0^{x,(n-1)}$ over the admissible decision space. This decision brings the system to the post-decision state $S_0^{x,(n)}$. Then a path of assets' returns $\boldsymbol{R}_1^{(n)}$ is simulated and the system transits to the next pre-decision state $S_1^{(n)}$. The procedure is repeated until the end of horizon T.

Each pre-decision state $S_t^{(n)}$ is related to a pre-decision value function $\widetilde{V}_t^{(n-1)}$, and each post-decision state $S_t^{(n)}$ is related to a post-decision value function $\widetilde{V}_t^{x,(n-1)}$. Moreover, pre-decision and post-decision value functions are tied to each other. For example, the two items embraced by the bracket ① are linked in the following way:

$$①: \quad \widetilde{V}_0^{(n-1)} = \max_{x_0} \widetilde{V}_0^{x,(n-1)} \tag{3.5}$$

which is in fact equation (3.3) for $t = 0$. Similarly, the two items inside bracket ② are linked by:

$$②: \quad \widetilde{V}_0^{x,(n-1)}(S_0^{x,(n)}) = \mathbb{E}[\widetilde{V}_1^{(n-1)}(S_1^{(n)})|S_0^{x,(n)}] \tag{3.6}$$

which is actually the definition of the post-decision value function (3.2) for $t = 0$. The expectation in the above equation is conditional on $S_0^{x,(n)}$ because asset returns $\boldsymbol{R}_1^{(n)}$ are not known yet at node $S_0^{x,(n)}$ (see Figure 3.5). Note that replacing $\widetilde{V}_0^{x,(n-1)}$ by ② in ① yields the usual Bellman equation for $t = 0$.

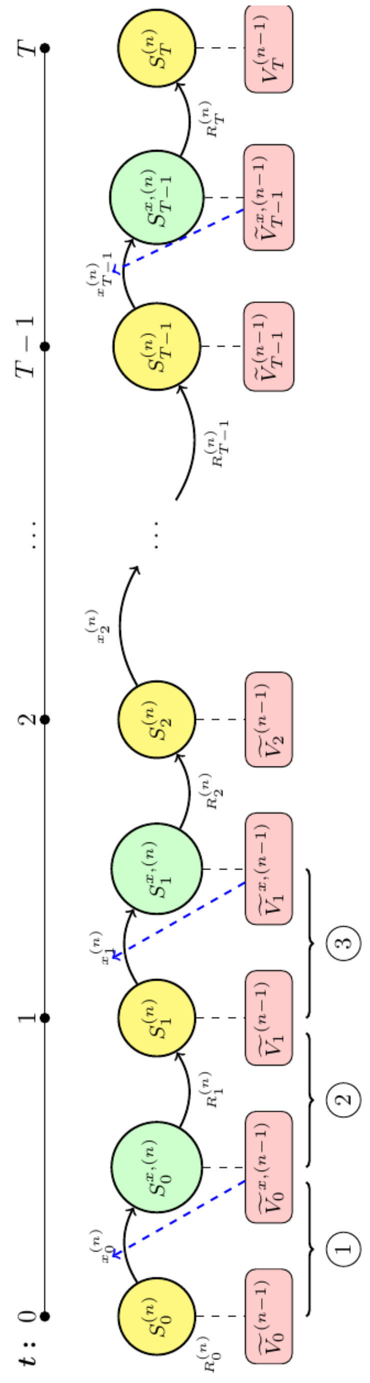


Figure 3.5: Illustration of the Forward Pass around Post-decision State

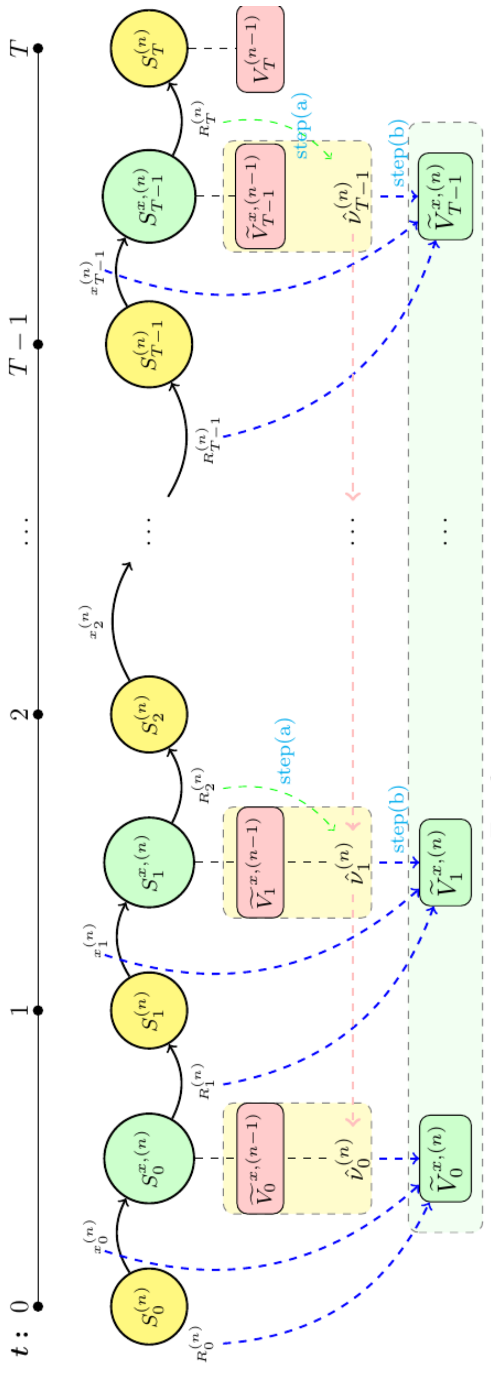


Figure 3.6: Illustration of the Backward Pass around Post-decision State

Finally, the relation of the two items inside $\textcircled{3}$ is given by:

$$\textcircled{3}: \quad \widetilde{V}_1^{(n-1)}(S_1^{(n)}) = \max_{x_1} \widetilde{V}_1^{x,(n-1)}(S_1^{x,(n)}) \tag{3.7}$$

which is in fact equation (3.3) for $t = 1$. Also note that replacing $\widetilde{V}_1^{(n-1)}$ by $\textcircled{3}$ in $\textcircled{2}$ yields the post-decision Bellman equation for $t = 0$.

In the backward pass (Figure 3.6), at each decision moment t starting from $t = T-1$ until $t = 0$, realized values $\hat{\nu}_t^{(n)}$ on the current path are first calculated in step(a), and value function approximations are updated in step(b) around the post-decision states $S_t^{x,(n)} = (\boldsymbol{R}_t^{(n)}, \boldsymbol{x}_t^{(n)}, W_t^{(n)})$ rather than around the pre-decision states $S_t^{(n)}$. In other words, updates around post-decision states involve the decision variable $\boldsymbol{x}_t$ in addition to the exogenous state variable $\boldsymbol{R}_t$ and the total wealth W_t. It is worth to point out that the value function at $t = T$ is known and given by the utility of final wealth:

$$V_T(S_T) = u(W_T), \qquad S_T = (\boldsymbol{R}_T, W_T). \tag{3.8}$$

Thus the time $T-1$ realized value equals to:

$$\hat{\nu}_{T-1} = \mathbb{E}[u(W_T)]$$

In the case that only one path of assets' returns is simulated, the expectation operator is unnecessary and we have:

$$\hat{\nu}_{T-1}^{(n)} = u(W_T^{(n)}).$$

Realized values for $t < T-1$ are obtained by the post-decision Bellman equation (3.4) and the expectation operator is omitted if only one path is simulated. Therefore, we have:

$$\hat{\nu}_t^{(n)} = \max_{x_{t+1}} \widetilde{V}_{t+1}^{x,(n)}, \quad 0 \le t < T-1$$

Following the literature (see for example Section 10.2.5 in [51]), the quantity $\hat{\nu}_t^{(n)}$ is a valid, unbiased estimate of the value of being in state $S_t^{x,(n)}$ and following the policy produced by $\widetilde{V}^{x,(n-1)}$. Also notice that, since the update procedure is completed backward, at the moment of calculating $\hat{\nu}_t^{(n)}$, the value function from the next period $\widetilde{V}_{t+1}^{x,(n-1)}$ has already been updated to $\widetilde{V}_{t+1}^{x,(n)}$. Therefore, we use the latest approximation $\widetilde{V}_{t+1}^{x,(n)}$ to obtain $\hat{\nu}_t^{(n)}$. Then the quantities $\hat{\nu}_t^{(n)}$ are used to update the value function approximation at time t according to the pre-selected updating rule to get the new approximation

$\widetilde{V}_t^{x,(n)}$. The algorithm subsequently steps back one period to $t-1$ and the same steps repeat until it reaches $t=0$.

Figure 3.7 presents our FDP algorithm in the general case where we don't yet impose a particular form for the utility function, value function approximations and update rules. The main purpose of this recursive FDP algorithm is to "learn and update" iterative approximations for the post-decision value functions through trial-and-errors at each decision moment t. The following parameters must be set before starting the algorithm:

- $\widetilde{V}_t^{x,(0)}(\boldsymbol{S}_t^x)$, $\forall t = 0, 1, \ldots, T-1$: initial guesses of the approximated value functions;
- N: total number of iterations (or stopping rules);
- *Update*$(\cdot)$: a technique to update the value function approximations.

The proposed basic FDP algorithm consists of several iterations of three main steps excluding the initialization step where an initial (pre-decision) state $\boldsymbol{S}_0^{(n)} = (\boldsymbol{z}_0^{(n)}, W_0^{(n)})$ is chosen and the iteration counter is set to $n=1$. Within each iteration, the first main step is to simulate a path for the exogenous state variables $\boldsymbol{Z}^{(n)} = \{\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}, \ldots, \boldsymbol{z}_T^{(n)}\}$.

Then the algorithm enters the second main step – the forward pass as depicted by Figure 3.5 – where one has in hand the current set of approximations $\widetilde{V}^{x,(n-1)}(\cdot)$ obtained from previous iteration $n-1$. Starting from $t=0$, decisions $\boldsymbol{x}_t^{(n)}$ are made based on these approximations by solving the deterministic optimisation problem (3.9). These decisions $\boldsymbol{x}_t^{(n)}$ bring the system to the time-t post-decision state $\boldsymbol{S}_t^{x,(n)} = \left(\boldsymbol{z}_t^{(n)}, \boldsymbol{x}_t^{(n)}, W_t^{(n)}\right)$. Then assets' returns for the next period $t+1$ are derived from $\boldsymbol{z}_{t+1}^{(n)}$ which is part of the simulated path of exogenous variables obtained in the simulation step. With the time $t+1$ assets' returns, the total wealth evolves according to the self-financing constraint (2.2). This procedure is repeated until $T-1$, where the post-decision value function is given by the expected utility of final wealth.

The third and final step is a backward pass to calculate realized values $\hat{\nu}_t^{(n)}$ before updating value function approximations, as illustrated by Figure 3.6. Starting from $t = T-1$, the realized values $\hat{\nu}_t^{(n)}$ are first computed recursively by equation (3.10), and then plugged into the pre-determined update rule to produce a new set of value function approximations $\widetilde{V}_t^{x,(n)}$, which will be used in the next iteration. It is worth noting that equation (3.10) is in fact a simplified version of the actual recursive equation which is

Basic Algorithm

```
1. Initialization.
    1.1 Initialize the pre-decision state $\boldsymbol{S}_0^{x,(n)}$ at $t=0$ for all iterations $n$.
    1.2 Set $n=1$.
2. Simulation step: simulate a path of the exogenous state variables which
   determine asset returns: $\boldsymbol{Z}^{(n)} = \{\boldsymbol{z}_1^{(n)}, \boldsymbol{z}_2^{(n)}, \dots, \boldsymbol{z}_T^{(n)}\}$.
3. Forward pass: for $t=\{0,1,\dots,T-1\}$, do:
    3.1 Decision step: Determine the optimal decision at $t$ using current approximations
        $\widetilde{V}_t^{x,(n-1)}$ by solving:
            $$\boldsymbol{x}_t^{(n)} = \arg\max_{\boldsymbol{x}_t} \widetilde{V}_t^{x,(n-1)}(\boldsymbol{S}_t^{(n)}, \boldsymbol{x}_t), \quad (3.9)$$
        where $\boldsymbol{S}_t^{(n)} = (\boldsymbol{z}_t^{(n)}, W_t^{(n)})$.
    3.2 Save the realized values of the triplet $\boldsymbol{S}_t^{x,(n)} \equiv \left(\boldsymbol{z}_t^{(n)}, W_t^{(n)}, \boldsymbol{x}_t^{(n)}\right)$.
    3.3 Calculate $W_{t+1}^{(n)}$ by the transition function: $W_{t+1}^{(n)} = W_t^{(n)}\left(\boldsymbol{x}_t^{(n)}\boldsymbol{R}_{t+1}^{(n)} + R_f\right)$, where
        $\boldsymbol{R}_{t+1}^{(n)}$ are obtained from $\boldsymbol{z}_{t+1}^{(n)}$.
    3.4 If $t<T-1, t=t+1$, go to step 3.1; else end loop on $t$.
    3.5 At the end of the forward pass, one should have in hand the n-th realisation of
        the triplets $\boldsymbol{S}_t^{x,(n)} \equiv \left(\boldsymbol{z}_t^{(n)}, W_t^{(n)}, \boldsymbol{x}_t^{(n)}\right)$, at all decision moments $t=0,\dots,T-1$.
        These triplets will be used in the backard pass to update value function
        approximations.
4. Backward pass: for $t=\{T-1,\dots,0\}$, do:
    4.1 Calculate the time-$t$ realized value on current path using the
        following recursive equation:
            $$\hat{\nu}_t^{(n)} = \begin{cases} u(W_T^{(n)}) & \text{if } t=T-1, \\ \max_{\boldsymbol{x}_{t+1}} \widetilde{V}_{t+1}^{x,(n)}(\boldsymbol{S}_{t+1}^{x,(n)}) & \text{if } t<T-1 \end{cases} \quad (3.10)$$
    4.2 Update the value function approximation using the triplet $\boldsymbol{S}_t^{x,(n)}$ and the realized
        value on current path $\hat{\nu}_t^{(n)}$ by the pre-chosen update rule:
            $$\widetilde{V}_t^{x,(n)}(\boldsymbol{S}_t^{x,(n)}) = Update(\widetilde{V}_t^{x,(n-1)}(\boldsymbol{S}_t^{x,(n)}), \hat{\nu}_t^{(n)}).$$
    4.3 If $t>0, t=t-1$, go to step 4.1; else end loop on $t$.
5. If $n<N, n=n+1$, go to step 2; else return the approximations $\widetilde{V}_t^{x,(N)}(\cdot), \forall t$.
```

Figure 3.7: Basic FDP algorithm with arbitrary choices of utility function and approximate post-decision value functions.

given below:

$$\hat{\nu}_t^{(n)} = \begin{cases} \mathbb{E}_t[u(W_T^{(n)})] & \text{if } t=T-1, \\ \mathbb{E}_t[\max_{\boldsymbol{x}_{t+1}} \widetilde{V}_{t+1}^{x,(n)}(\boldsymbol{S}_t^{x,(n)})] & \text{if } t<T-1 \end{cases} \quad (3.11)$$

Compared to equation (3.11), the expectation operators are removed in (3.10) since only one path is simulated within each iteration. In practice, the actual performance of the FDP algorithm depends on the quality of the input signal (or measurement) sequences $\hat{\nu}_t^{(n)}$. If there is too much noise in the input signals, the approximations may actually not converge to the true value functions. To decrease input noises and to prevent brutal oscillation of the estimates of $\hat{\nu}_t^{(n)}$, we could use equation (3.11) instead of (3.10) to calculate the realized value in the backward pass. The expectation can be computed either by closed formula (if available), quadrature methods (see [31]) or Monte Carlo simulation (which is in fact what we suggest in Section 3.3.3).

This three-step procedure is repeated until the maximum number of iteration is attained or the stopping criteria are satisfied. The proposed FDP algorithm is similar to the double-pass algorithm for finite horizon (Figure 10.5) in [51]. The main difference is that the double-pass algorithm in [51] proposes only to update the value function approximation at $t = 0$ and thus the optimal policy is updated only at $t = 0$ through one maximization. While in our algorithm, value function approximations at all decision moments $t = 0, \ldots, T-1$ are updated and consequently the optimal policies at all decision moments are updated. The next section presents a simplified 2-period example to illustrate the basic FDP algorithm.

3.3.2 An Illustrative Example for the General FDP Algorithm

To better illustrate the general algorithm, consider the following 2-period example with a single risky asset and a risk free asset. The risk free rate is 2%, meaning $R_f = 1.02$. At each decision moment, the investor has two choices: $x_t \in \{0, 100\%\},\ t = 0, 1$, which means investing either none or all of his/her wealth in the risky asset. The exogenous variable can take two values in each period: $z_t \in \{-0.5, 0.5\},\ t = 1, 2$. The asset returns are influenced by the exogenous state variable in the following way:

$$R_t = \begin{cases} 0.1, & \text{if } z_t = -0.5 \\ -0.1, & \text{if } z_t = 0.5 \end{cases}$$

It is worth to emphasize that the type of notations without the superscript (n) such as $\boldsymbol{z}_t$ stands for either a state or a decision *variable*, whose value is unknown. Once the iteration number (n) is added as a superscript, notations such as $\boldsymbol{z}_t^{(n)}$ denote one *realization* (sample path) of the corresponding variable $\boldsymbol{z}_t$ and thus its value becomes known and deterministic, i.e. no more uncertainty.

Before launching the algorithm, one has to decide how to update value function approximations. The update rule should be problem specific. In the current example, we use a simple average rule for illustration purpose and refer the readers to Section 3.4 for more discussions on this subject.

At the beginning of the n^{th} iteration, the initial state is set to $z_0^{(n)} = 0$ and $W_0^{(n)} = 1$. Next, a random path of the exogenous variable is generated: $Z^{(n)} = \{z_1^{(n)},\ z_2^{(n)}\} = \{-0.5. 0.5\}$, which yields the following return path: $R_1^{(n)} = 0.1,\ R_2^{(n)} = -0.1$.

Then the forward pass starts: based on the set of value function approximations obtained in iteration $n-1$, decisions are made forward in time. The value function approximations can be very simple such as a lookup table representation or rather complex. Let's assume a lookup table representation in the current example and suppose that $V_0^{(n-1)}$ is given by

$$\widetilde{V}_0^{x,(n-1)}(z_0, x_0, W_0) = \begin{cases} 4, & \texttt{if } (z_0, x_0, W_0) = (0, 0\%, 1) \\ 5, & \texttt{if } (z_0, x_0, W_0) = (0, 100\%, 1) \\ 10, & \texttt{otherwise.} \end{cases}$$

Therefore, the optimal decision at $t = 0$ for this iteration will be $x_0^{(n)} = 100\%$ since $V_0^{x,(n-1)}(0, 0\%, 1) = 4 < V_0^{x,(n-1)}(0, 100\%, 1) = 5$. The decision $x_0^{(n)}$ changes the total wealth to $W_1^{(n)}$ according to the transition function (2.2):

$$W_1^{(n)} = W_0^{(n)}(R_f + x_0^{(n)} R_1^{(n)}) = 1 * (1.02 + 100\% * 0.1) = 1.12$$

The procedure is very the same for $t = 1$. Suppose the current approximation of $V_1^{(n-1)}$ tells us:

$$V_1^{x,(n-1)}(z_1, x_1, W_1) = \begin{cases} 3, & \text{if } (z_1, x_1, W_1) = (-0.5, 0\%, 1.12) \\ 2, & \text{if } (z_1, x_1, W_1) = (-0.5, 100\%, 1.12) \\ 1, & \text{otherwise.} \end{cases}$$

So the optimal decision at $t = 1$ is $x_1^{(n)} = 0$ because $V_1^{x,(n-1)}(-0.5, 0\%, 1.12) = 3 > V_1^{x,(n-1)}(-0.5, 100\%, 1.12) = 2$. Accordingly, the final wealth at $t = T = 2$ on current path is given by

$$W_2^{(n)} = W_1^{(n)}(R_f + x_1^{(n)} R_2^{(n)}) = 1.12 * (1.02 + 0\% * (-0.1)) = 1.1424.$$

The last step is the backward pass to first calculate the realized value on current path and then use them to update backwardly value function approximations. The backward pass starts from $t = T - 1 = 1$ by either calculating $\hat{\nu}_1^{(n)}$ in its full version according to equation (3.11) or by approximating it according to equation (3.10). Note that, in either way, the actual calculation of $\hat{\nu}_1^{(n)}$ depends on the underlying utility function. In this illustrative example, we assume an arbitrary utility function which gives $\hat{\nu}_1^{(n)} = u(W_2^{(n)}) = u(1.1424) = 11$. This value is then used to update $V_1^{x,(n-1)}(\boldsymbol{S}_1^x)$ by the pre-determined updating rule:

$$V_1^{x,(n)}(\boldsymbol{S}_1^x) = \begin{cases} \frac{V_1^{x,(n-1)}(\boldsymbol{S}_1^x)+\hat{\nu}_1^{(n)}}{2} = \frac{3+11}{2} = 7, & \text{if } \boldsymbol{S}_1^x = (z_1^{(n)}, x_1^{(n)}, W_1^{(n)}) = (-0.5, 0\%, 1.12), \\ V_1^{x,(n-1)}(\boldsymbol{S}_1^x), & \text{otherwise.} \end{cases}$$

Notice that when a lookup table representation is used to approximate value functions, only values related to states visited by the simulated paths are updated. Values of unvisited states remain unchanged.

Similarly, the update at $t = 0$ is completed by first computing $\hat{\nu}_0^{(n)}$ either by equation (3.10) or its full version (3.11), which again depends on the underlying utility function. Again, we adopt the simplified version (3.10), thus $\hat{\nu}_0^{(n)} = \max_{\boldsymbol{x}_1} \widetilde{V}_1^{x,(n)}(\boldsymbol{S}_1^{x,(n)}) = 7$, and the new set of approximations at $t = 0$ is given by:

$$V_0^{x,(n)}(\boldsymbol{S}_0^x) = \begin{cases} \frac{V_0^{x,(n-1)}(\boldsymbol{S}_0^x)+\hat{\nu}_0^{(n)}}{2} = \frac{5+7}{2} = 6, & \texttt{if } \boldsymbol{S}_0^x = (z_0^{(n)}, x_0^{(n)}, W_0^{(n)}) = (0, 100\%, 1), \\ V_0^{x,(n-1)}(\boldsymbol{S}_0^x), & \texttt{otherwise} \end{cases}$$

The same steps are repeated until the maximum number of iterations is attained or the stopping criteria are satisfied. Note that when a lookup table representation is used as approximation, only values of the visited states are updated. If a state is never visited, its corresponding value will not be updated and thus stays at the initial guess. For this reason, the choice of initial guesses often has significant impact on the algorithm's performance. Bad initial guesses might bias the decision making process, leading to suboptimal policies. Therefore, value function approximations should be initiated with care and caution.

A useful approach to reduce the impact of initial guesses on performance is to introduce the $\epsilon - greedy$ strategy: with probability ϵ, choose a decision at random; and with probability $1 - \epsilon$, choose a decision as usual, i.e. by solving equation (3.9) using the old set of approximations. In the literature, the former way is known as *exploration* and the latter is known as *exploitation*. Exploring new states from time to time can greatly enhance the algorithm's estimating behaviour, while on the other side too much exploration may also deteriorate the convergence speed, especially when the decision space is large. This is the so called the *exploration-exploitation dilemma* which will be further discussed in Section 3.4.

FDP algorithms can substantially change flavour according to the choices of functional approximation and the update procedure. The following two sections focus on the particular cases of OPCPs with CRRA utility, where post-decision value functions are approximated by polynomial bases and coefficients are updated by regression and exponential smoothing. Other possible update techniques are discussed in Section 3.4.

3.3.3 FDP Algorithm with CRRA Utility, Predictable Returns and Polynomial Bases

Consider the problem described in Section 2.2.1 where the utility function is characterized by CRRA, assets' returns follow a VAR(1) process and the bases are polynomials. As discussed previously, with the CRRA utility function, optimal policies do not depend on the level of total wealth which is thus removed from the list of state variables. The VAR(1) dynamic implies that the exogenous state variables are the one-period lagged assets' returns. Therefore, the pre-decision and post-decision state variables reduce to $\boldsymbol{S}_t = \boldsymbol{R}_t$, and $\boldsymbol{S}_t^x = (\boldsymbol{R}_t, \boldsymbol{x}_t)$ respectively. The post-decision version of Bellman equation on the reduced[3] value function is given by

$$\mathcal{V}_t^x(\boldsymbol{S}_t^x) = \mathbb{E}_t\left[(R_f + \boldsymbol{x}_t\boldsymbol{R}_{t+1})^{1-\gamma}\max_{\boldsymbol{x}_{t+1}}\mathcal{V}_{t+1}^x(\boldsymbol{S}_{t+1}^x)|\boldsymbol{S}_t^x\right], \tag{3.12}$$

with the terminal condition $\mathcal{V}_{T-1}^x(\boldsymbol{S}_{T-1}^x) = \mathbb{E}_{T-1}\left[(R_f + \boldsymbol{x}_{T-1}\boldsymbol{R}_T)^{1-\gamma}/(1-\gamma)\right]$.

From this section forth, we use polynomial bases to approximate post-decision value functions around post-decision states. Precisely, we consider the linear regression model:

$$\mathcal{V}_t^x(\boldsymbol{S}_t^x) \approx \widetilde{\mathcal{V}}_t^x(\boldsymbol{S}_t^x) := \sum_{p=1}^{P}\theta_{t,p}\phi_p(\boldsymbol{S}_t^x) = \boldsymbol{\theta}_t'\boldsymbol{\Phi}(\boldsymbol{S}_t^x), \quad t = 0,\ldots,T-1. \tag{3.13}$$

where $\phi_p(\boldsymbol{S}_t^x) : \mathcal{S}_t^x \mapsto \mathbb{R}$ is the p^{th} base function whose value depends on the post-

[3]Recall that the term "reduced" refers to the fact that the value functions are independent of total wealth.

decision state variable $\boldsymbol{S}_t^x$ and $\theta_{t,p}$ is its associated parameter or coefficient which will be estimated by regression. It is worth to point out that this model is linear with respect to the coefficients $\boldsymbol{\theta}_t$, but the base ϕ_p themselves can be non-linear regarding to state variables $\boldsymbol{S}_t^x$. Using this form of representation, the post-decision value functions $\mathcal{V}_t^x(\boldsymbol{S}_t^x)$ are "summarized" by the coefficients $\boldsymbol{\theta}_t$. Our parametric value function approximation (VFA) reduces the problem of finding the optimal value for each state to a much simpler problem of estimating P parameters $\theta_{t,p}$ at each time step. It is assumed that the base functions ϕ_p are time-invariant (and thus are not indexed by t), while their associated coefficients $\theta_{t,p}$ are time-dependent. In this way, the form of the function approximations is preserved while their value can evolve across time steps. This time-variant feature is typical for finite horizon problems, in contrast to infinite horizon problems where the system is assumed to be in a *steady-state* and thus the coefficients $\boldsymbol{\theta_t}$ are time-independent. The non-stationary property of finite horizon problems causes additional difficulties, because the number of coefficients to be estimated is $P*T$ instead of P as it is the case for an infinite horizon problem. Moreover, estimation errors might accumulate quickly across time periods.

There are many candidates for the basis functions $\phi_p(\boldsymbol{S}_t^x)$. The best choice varies from one problem to another, and there is no general rule (except for theoretical analyses which require some additional conditions). The best practice is to exploit as much as possible problem-specific characteristics and to select base functions that reflect most these characteristics. In the CRRA utility case, we adopt linear, quadratic and cross-product terms of decisions and cross-product terms of decisions-returns as well as the constant term (please refer to Chapter 4 for implementation details).

The selected updating rule is the exponential smoothing with a stepsize λ:

$$\boldsymbol{\theta}_t^{(n+1)} = \lambda \boldsymbol{\theta}_t^{(n)} + (1-\lambda)\boldsymbol{\theta}_t^M.$$

The parameter λ is also known as the "forgetting factor". It controls the smoothing or "forgetting" speed. By varying the value of λ, one adjusts the weights attributed to current approximations $\boldsymbol{\theta}_t^{(n)}$ and new information obtained in the last iteration $\boldsymbol{\theta}_t^M$. Large values of λ means more confidence on current approximations. We refer the readers to Section 3.4 for a discussion on the selection of stepsize.

Having these approximated value functions and updating rule in hand, we propose an algorithm for the case with CRRA utility and VAR(1) return dynamic as shown in

Figure 3.8. This algorithm represents a refinement of the basic algorithm in Figure 3.7. Recall that our basic algorithm is inspired by the double-pass algorithm for approximate policy iteration (Figure 10.5 on page 392) in [51]. The main difference between the double-pass algorithm and our basic algorithm is that the former only updates value function approximations at $t = 0$, while the latter performs updates at all decision moments $t = 0, \dots, T-1$.

The algorithm in Figure 3.8 is almost the same as the general algorithm, except for two major variations which need further explications. These two variations, to the best of our knowledge, have not been applied in the context of solving OPCP by FDP, and hence represent our major innovations to the literature. The first variation is within the forward pass: an inner loop on m is added within each iteration n. The objective of this inner loop is to build M pairs of $(\boldsymbol{x}_t^{(m,n)}, \boldsymbol{R}_t^{(m,n)})$ which are used later in the regression of the backward pass. In this inner loop, the current set of approximations is evaluated M times with additional simulated paths before any update is made. In other words, the coefficients $\boldsymbol{\theta}_t^{(n-1)}$, which summarize the approximated value functions after $n-1$ iterations, are kept unchanged within the n^{th} iteration and are used to make decisions on all the M additional paths, just as depicted in the decision step 3.1.1. The parameter M can thus be viewed as the number of points in the regression. Note that, by adding this inner loop, the resulting complete algorithm consists of $N * M$ forward passes and N backward passes (each involving M paths), compared to the general algorithm which includes N forward passes and N backward passes.

The second variation is in step 4.1.1.2 of the backward pass, where a sub-simulation step is introduced to estimate the realized value on current path $\hat{\nu}_t^{(m,n)}$ which is in fact a conditional expectation. This step aims to prevent big swings in the realized value $\hat{\nu}_t^{(m,n)}$ from one simulated path to another, and thus to insure faster convergence and more accurate estimations. Other techniques, such as quadrature methods, can also be used here as an alternative way to estimate the conditional expectation. We adopt the Monte Carlo simulation method because it is robust, efficient and can be easily extended to multidimensional cases.

Algorithm – VAR(1) Asset Returns

```
1. Initialization.
    1.1 Initialize $\boldsymbol{S}_0^{(m,n)}$ for all iterations $m, n$.
    1.2 Make an initial guess for the coefficients $\boldsymbol{\theta}_t^{(0)} = \mathbf{0}$.
    1.3 Set $n = 1$.
2. Simulation step: simulate $M$ paths of asset returns:
       $\{\boldsymbol{R}_1^{(m,n)}, \boldsymbol{R}_2^{(m,n)}, \ldots, \boldsymbol{R}_T^{(m,n)}\},\ m = 1, \ldots, M.$
3. Forward pass: set $m = 1$, then do:
    3.1 Set $t = 0$, and do:
      3.1.1 Decision step: Determine the optimal decision at time $t$ using the $\epsilon$-greedy strategy:
            with probability $\epsilon$, choose a decision at random; with probability $1 - \epsilon$, choose a decision
            optimally by solving:
            $\boldsymbol{x}_t^{(m,n)} = \arg\max_{\boldsymbol{x}_t} \boldsymbol{\theta}_t^{(n-1)} \boldsymbol{\Phi}(\boldsymbol{x}_t, \boldsymbol{R}_t^{(m,n)})$
      3.1.2 Save the pair $(\boldsymbol{x}_t^{(m,n)}, \boldsymbol{R}_t^{(m,n)})$.
      3.1.3 If $t < T - 1$, $t = t + 1$, go to step 3.1.1; else end loop on $t$.
    3.2 If $m < M$, $m = m + 1$, go to step 3.1); else end loop on $m$.
    3.3 At the end of the forward pass, one should have in hand $M$ realisations of the triplets
        $\boldsymbol{S}_t^{x,(n)} \equiv \left(\boldsymbol{z}_t^{(n)}, W_t^{(n)}, \boldsymbol{x}_t^{(n)}\right)$, at all decision moments $t = 0, \ldots, T - 1$. These triplets will be
        used in the backard pass to update value function approximations.
4. Backward pass: For $t = \{T - 1, \ldots, 0\}$, do:
    4.1 Set $m = 1$ and do:
      4.1.1 For the path $(m, n)$ obtained in the forward pass, calculate the realized value $\hat{\nu}_t^{(m,n)}$ by
            estimating the time-$t$ conditional expectation:
        4.1.1.1 Sub-simulation step: given the time-$t$ return $\boldsymbol{R}_t^{(m,n)}$, simulate $Nsub$ return paths for
                time $t + 1$ according to the VAR(1) dynamic: $\left\{\boldsymbol{R}_{t+1}^{(m,n,k)}\right\}_{k=1}^{Nsub}$.
        4.1.1.2 Calculate the realized value on path $(m, n)$ by:
                $$\hat{\nu}_t^{(m,n)} = \frac{1}{Nsub} \sum_{k=1}^{Nsub} \left(R_f + \boldsymbol{x}_t^{(m,n)} \boldsymbol{R}_{t+1}^{m,n,k}\right)^{1-\gamma} f(\boldsymbol{R}_{t+1}^{(m,n,k)}), \qquad (3.14)$$
                where the function $f(\cdot)$ is defined by:
                $$f(\boldsymbol{R}_{t+1}^{(m,n,k)}) = \begin{cases} \frac{1}{1-\gamma}, & \text{if } t = T - 1, \\ \max_{\boldsymbol{x}_{t+1}} \boldsymbol{\theta}_{t+1}^{(n)} \boldsymbol{\Phi}(\boldsymbol{x}_{t+1}, \boldsymbol{R}_{t+1}^{(m,n,k)}), & \text{otherwise} \end{cases} \qquad (3.15)$$
      4.1.2 If $m < M, m = m + 1$, go to step 4.1.1; else end loop on $m$.
    4.2 Calculate the values of base functions on all paths: $\mathbf{X} := \{\Phi(\boldsymbol{x}_t^{(m,n)}, \boldsymbol{R}_t^{(m,n)})\}_{m=1}^{M}$.
    4.3 Regress the vector of realized values $\mathbf{y} := \{\hat{\nu}_t^{(m,n)}\}_{m=1}^{M}$ over $\mathbf{X}$ to obtain the
        estimated coefficients within the n-th iteration: $\boldsymbol{\theta}_t^{M} = \mathbf{X} \backslash \mathbf{y}$.
    4.4 Update the coefficients via exponential smoothing: $\boldsymbol{\theta}_t^{(n)} = \lambda \boldsymbol{\theta}_t^{(n-1)} + (1 - \lambda) \boldsymbol{\theta}_t^{M}$.
    4.5 If $t > 0, t = t - 1$, go to step 4.1; else end loop on $t$.
5. If $n < N, n = n + 1$, go to step 2; else return the coefficients $\boldsymbol{\theta}_t^{(N)}, \forall t$.
```

Figure 3.8: FDP algorithm with CRRA utility function and assets' returns follow the VAR(1) process as given in Section 2.2.1.

3.3.4 FDP Algorithm with CRRA Utility and IID Returns

Now let's consider the problem formulation described in Section 2.2.2 where assets' returns are assumed to be IID. In this particular case, the pre-decision value function at each time step does not depend on any state variable and thus reduces to a constant. The post-decision state consists of only the decision vector: $\boldsymbol{S}_t^x := \boldsymbol{x}_t,\ t = 0, \ldots, T-1$, and the post-decision Bellman equation is given by:

$$\widetilde{V}_t^x(\boldsymbol{x}_t) = \mathbb{E}_t \left[(R_f + \boldsymbol{x}_t \boldsymbol{R}_{t+1})^{1-\gamma} \max_{\boldsymbol{x}_{t+1}} \widetilde{V}_{t+1}^x(\boldsymbol{x}_{t+1}) \right], \tag{3.16}$$

with the terminal condition

$$\widetilde{V}_{T-1}^x(\boldsymbol{x}_{T-1}) = \mathbb{E}[\mathcal{V}_T] = \mathbb{E}_{T-1} \left[\frac{(R_f + \boldsymbol{x}_{T-1}\boldsymbol{R}_T)^{1-\gamma}}{1-\gamma} \right], \ \forall \boldsymbol{x}_{T-1}.$$

Figure 3.9 provides an algorithm for the IID-return case. Compared to the algorithm in previous section (Figure 3.8), the only adjustment is in equation (3.17), the calculation of $\hat{\nu}_t^{(m,n)}$. Note in this case of IID assets' returns, the expression of bases $\boldsymbol{\Phi}(\cdot)$ only has one argument which is the decision vector. In other words, the decision does not depend on the previous asset returns. Thus the deterministic maximisation problem $\max_{\boldsymbol{x}_{t+1}} \boldsymbol{\theta}_{t+1}^{(n-1)} \boldsymbol{\Phi}(\boldsymbol{x}_{t+1})$ need be solved only once at each time t for all return paths in the sub-simulation step.

Algorithm – IID Asset Returns

```
1. Complete steps 1 through 2.2 - a.1) of the algorithm in Figure 3.8.
2. In Step 4.1.1.2 of the backward pass, use instead the following recursive equation to
   calculate the realized value on path $(m,n)$:
```

$$\hat{\nu}_t^{(m,n)} = \frac{1}{Nsub} \sum_{k=1}^{Nsub} \left(R_f + \boldsymbol{x}_t^{(m,n)} \boldsymbol{R}_{t+1}^{m,n,k} \right)^{1-\gamma} \max_{\boldsymbol{x}_{t+1}} \boldsymbol{\theta}_{t+1}^{(n)} \boldsymbol{\Phi}(\boldsymbol{x}_{t+1}) \tag{3.17}$$

```
3. Complete the other steps of the algorithm in Figure 3.8.
```

Figure 3.9: FDP algorithm with CRRA utility function and iid assets' returns as described in Section 2.2.2.

3.4 Algorithm Tuning and Parameter Selection

Forward algorithms are usually sensitive to the choice of algorithmic parameters such as $N, M, Nsub, \lambda$ and ϵ. Ill-chosen parameters may lead to undesirable or unstable estimation results. Unfortunately, universal rules do not generally exist in the literature for parameter selection so the choice is often heuristic. This section is devoted to a discussion on parameter selection. Sensitivity analyses with numerical examples are presented in Section 4.1.2.

Number of iteration N

The choice of total number of iterations N is usually straightforward and relatively easy. If historical data is used, N is limited to the maximum number of data points available. Otherwise, if a simulator is used to simulate exogenous state variables, N is not bounded upward. The easiest and most naive method is to choose an arbitrary N. The idea is to fix N to a sufficiently large number, and "hopefully" convergence can be achieved within those iterations but nothing is guaranteed. An alternative way widely used in operational research is to stop the algorithm once the pre-selected criteria are met. Examples of such criteria include

1. Stop criterion on optimal weights: stop if $\left\|\boldsymbol{x}_t^{(n)} - \boldsymbol{x}_t^{(n-1)}\right\| < \psi$
2. Stop criterion on approximation coefficients: stop if $\left\|\boldsymbol{\theta}_t^{(n)} - \boldsymbol{\theta}_t^{(n-1)}\right\| < \psi$
3. Stop criterion on approximated value functions: stop if $\left\|V_t^{(n)} - V_t^{(n-1)}\right\| < \psi$

where $||\cdot||$ is the l_p-norm from the vector space of its argument to $\mathbb{R}$ and ψ is the predetermined tolerance level.

In this thesis, we use a variant of the stopping criterion on optimal weights, i.e. criterion (1). In fact, at the end of the $n-th$ iteration, we calculate the average weights of the last 10 iterations $\boldsymbol{x}_t^{(n,10)}$ and of the 20 last iterations $\boldsymbol{x}_t^{(n,20)}$ in additional to $\boldsymbol{x}_t^{(n)}$. The algorithm is stopped when

1.a) $\left\|\boldsymbol{x}_t^{(n)} - \boldsymbol{x}_t^{(n-1)}\right\| < \psi$,

1.b) $\left\|\boldsymbol{x}_t^{(n,20)} - \boldsymbol{x}_t^{(n)}\right\| < \psi$, and

1.c) $\left\|\boldsymbol{x}_t^{(n,10)} - \boldsymbol{x}_t^{(n)}\right\| < \psi$.

Sub-simulation parameter $Nsub$

In the algorithms presented in Sections 3.3.3 and 3.3.4, Monte Carlo simulation is used to improve the quality of the input signal (or measurement) sequences $\hat{\nu}_t^{(m,n)}$. In fact, the following expectation

$$\hat{\nu}_t^{(m,n)} = \mathbb{E}\left[\left(R_f + \boldsymbol{x}_t^{(m,n)}\mathbf{R}_{t+1}^{(m,n)}\right)^{1-\gamma} \max_{\boldsymbol{x}_{t+1}} \widetilde{\mathcal{V}}_{t+1}^{x,(n-1)}(\boldsymbol{S}_{t+1}^x)\right]. \tag{3.18}$$

is approximated by first performing a sub-simulation of size K for $\mathbf{R}_{t+1}^{(m,n,k)}$ given $\mathbf{R}_t$, $k = 1, \dots, K$ and then computing the average realized value by equation (3.14) which is repeated below:

$$\hat{\nu}_t^{(m,n)} = \frac{1}{Nsub}\sum_{k=1}^{Nsub}\left(R_f + \boldsymbol{x}_t^{(m,n)}\boldsymbol{R}_{t+1}^{(m,n,k)}\right)^{1-\gamma} \max_{\boldsymbol{x}_{t+1}} \boldsymbol{\theta}_{t+1}^{(n)}\boldsymbol{\Phi}(\boldsymbol{x}_{t+1}, \boldsymbol{R}_{t+1}^{(m,n,k)})$$

Since the goal of this sub-simulation is to approximate the expectation in (3.18), increasing the number of simulated paths $Nsub$ improves the approximation's quality. Moreover, variance reduction techniques can be employed to obtain more accurate results with less numerical efforts. Our numerical results suggest that $Nsub$ should increase when there are more risky assets. And by setting $Nsub \in [20, 200]$ and combining variance reducing techniques such as antithetic variables and sample moment matching, is sufficient to assure convergence.

Number of inner iteration M

The parameter M controls the number of sub-iterations in order to evaluate a set of value function approximations before any update is made. It also corresponds to the number of data points in the linear regression within each iteration n to obtain a new vector of coefficients $\boldsymbol{\theta}_t^M$. Therefore, M should be large enough to make the regression estimates statistical significant. A rule of thumb in statistics for the minimum data size is to add 10 points for each additional coefficient to be estimated. Our numerical tests agree with this empirical rule. Hence we suggest to set $M = 10 * P \pm M_0$ where P is the number of bases or the number of coefficients to be estimated and $M_0 \in [0, 20]$ is an integer to allow a certain degree of flexibility.

Stepsize λ

The coefficients $\boldsymbol{\theta}_t$ are updated by exponential smoothing with a stepsize λ. The choice of λ is often crucial for the algorithm's convergence. Inappropriate choices have led many to conclude that their algorithm does not work at all[4]. Empirical studies suggest that choosing $\lambda \in [0.9, 0.99]$ works fine.

In this thesis, we adopt a two-stage constant stepsize rule, which means

$$\lambda = \begin{cases} \lambda_1, & \text{if } 0 < n \le N_\lambda, \\ \lambda_2, & \text{if } n > N_\lambda, \end{cases}$$

where $0.9 \le \lambda_1 \le \lambda_2 < 1$ are constant stepsizes in the first and second stage respectively, and N_λ is the number of iterations in the first stage, an integer not too large (≤ 50). The intuition behind this strategy is to rely less on earlier approximations when the algorithm starts since the initial guess may contain little useful information. Then as iterations advance, approximations improve with trials and errors, and we consequently adjust the stepsize upward to put more weights on existing approximations (i.e. what we already know). When $\lambda_1 = \lambda_2$, the 2-stage rule reduces evidently to the usual constant stepsize rule.

In the literature, adaptive stepsize and stochastic stepsizes rules have been developed to deal with more complex problems[5]. The main idea is to try to find a balance between noise and the change in the signal by adjusting to the data in a way that keeps the stepsize larger when the parameter being estimated is still changing quickly. In this thesis, only constant stepsizes are considered since they behave quite well and we want to keep the FDP algorithm to its simplest form, although incorporating adaptive stepsizes into the proposed FDP algorithm is a natural extension of our works and is one of our future research directions.

Exploration percentage ϵ

An important question for a typical FDP algorithm is the *"exploitation-exploration dilemma"*: in order to act near optimally, the agent must choose between what he knows

[4]Reference:[51], page 426.
[5]See for example Chapter 11 in [51].

and gets something close to what he expects (exploitation) and something he is not sure about and possibly learns more (exploration). Taking an action because it appears to be the most appropriate way is called *"exploitation"*. However, remember that the state space might be so big that there might be states with better values which would never be visited if we always do what we think the best. Sometimes, we need to try something new in the hope to explore more valuable states. This is called *"exploration"*, i.e. take an action by exploring a new option even it is suboptimal.

Finding a good balance between exploration and exploitation is a challenge. On one hand, it can cost time (and money in some cases) to visit a new state; on the other hand, we may have doubt about how well we know the value of being in a state and whether a decision will help us learn this value better. Sometimes a simple heuristic rule like pure exploration or pure exploitation works, while other times more sophisticated technique such as *Bayesian* belief structures must be used in order to obtain convergence. It is thus the modeller's responsibility to seek for the problem-specific features and adopt the appropriate strategy.

We employ in this paper the ϵ-greedy exploration strategy, i.e. with probability ϵ we explore by taking decision randomly, and with probability $1-\epsilon$ we make decision optimally by exploiting what we already know. The intuition is that we force the algorithm to explore to a certain degree, while the rest of iterations focus on states appearing to be the most valuable. This strategy is well-known and widely-employed in the literature. For example, [47] propose to use a pure exploitation strategy ($\epsilon = 0\%$) in their algorithm with piecewise linear function approximations to solve mutual fund cash balance problem and lagged asset acquisition problem. The convergence of their algorithm is demonstrated in [46].

Similar to the stepsize λ, we allow a 2-stage ϵ-greedy exploration. Precisely,

$$\epsilon = \begin{cases} \epsilon_1, & \text{if } 0 < n \leq N_\epsilon, \\ \epsilon_2, & \text{if } N_\epsilon < n, \end{cases}$$

where $0 \leq \epsilon_1 \leq \epsilon_2 \leq 1$, and N_ϵ is the number of iterations in the first stage for ϵ. Obviously N_ϵ can be set to N_λ for simplicity. And letting $\epsilon_1 = \epsilon_2$, one finds the standard ϵ-greedy exploration rule.

An interesting variation suggested by [50] is to allow the percentage of exploration ϵ

to decrease with the number of iterations, for example

$$\epsilon^n(s) = \frac{c}{N^n(s)},$$

where $0 < c < 1$ is a tuning parameter and $N^n(s)$ is the number of times state s has been visited by iteration n. This variation is useful when the approximation is poor when the algorithm starts, thus to explore more at the first stage. It is expected that the approximations improve along with iterations so that one could reduce the exploration and rely on the most recent approximation (i.e. to use more exploitation). However, this variation has a danger: if exploration percentage decreases faster than the approximations converge to the "real" solution, the algorithm could diverge and fail to find the optimal solution. It is thus important to control the exploration decreasing speed, which is done by varying the constant c in the formula above. Yet the best value of c to be used is a tricky question and only can be found by try-and-error in problem specific practices.

Chapter 4

Numerical Results

This chapter compares, through numerical tests, the performance (optimal weights and certainty equivalent rates) of our FDP algorithm to a BDP algorithm (presented in Appendix C) which discretizes the state space and uses Gauss-Hermite quadrature (Appendix A) to compute conditional expectations. We start by considering the simplest case with one single risky asset whose returns are IID (Sections 4.1.1 and 4.1.2), followed by a little more complex case with three risky assets but still IID returns (Section 4.1.3). Next, cases with predictable returns are studied in Section 4.2.

In all our numerical examples, short selling and borrowing at risk-free rate are forbidden in all risky assets, thus the percentage allocations in asset i are bounded by $[0, 1]$ and the sum of all allocations cannot exceed 1. The self-financing constraint is imposed, and thus total wealth evolves according to (2.2). Note that, due to this constraint, the allocation in risk-free asset need not to be modelled explicitly and is given by the balance of the total wealth (equals to 1 in the CRRA utility case) minus holdings in all risky assets.

4.1 Test 1: CRRA Utility with IID Returns

When assets' returns are IID, we use polynomials up to order 2 of the decision variables to approximate the post-decision value functions. As an example, for the case with 3 risky assets, the approximation of the time-t post-decision value function is given by

$$\begin{aligned}\widetilde{V}_t^x(\boldsymbol{x}_t) =& \theta_{t,1} + \sum_{i=1}^{3}\left(\theta_{t,i+1}x_{t,i} + \theta_{t,i+4}x_{t,i}^2 + \sum_{j>i}\theta_{t,5+i+j}x_{t,i}x_{t,j}\right)\\ =& \boldsymbol{\theta}_t^{\prime}\boldsymbol{\Phi}(\boldsymbol{x}_t),\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{\theta}_t &= [\theta_{t,1},\ \ldots,\ \theta_{t,10}]'\\ \boldsymbol{\Phi}(\boldsymbol{x}_t) &= [1,\ x_{t,1},\ x_{t,2},\ x_{t,3},\ x_{t,1}^2,\ x_{t,2}^2,\ x_{t,3}^2,\ x_{t,1}x_{t,2},\ x_{t,1}x_{t,3},\ x_{t,2}x_{t,3}]'.\end{aligned}$$

In addition to the comparison of optimal weights, for multi-asset cases, we also compare certainty equivalent rates. In fact, it is more common to compare certainty equivalent rates in a multidimensional environment (rather than only comparing optimal weights themselves) because policies which appear quite different in asset weights may lead to similar certainty equivalent rates.

The certainty equivalent is a guaranteed return that someone would accept rather than taking a chance on a possibly higher, but uncertain, return. It represents the guaranteed amount of cash that would yield the same exact expected utility as a given risky asset with absolute certainty.

Following [26], we evaluate the certainty equivalent rates (CER) resulting from a certain decision policy through Monte Carlo simulation.

Specifically, we simulate a sample of I return paths according to assets' return distribution $\boldsymbol{r}_t \sim \mathcal{N}(\mu, \boldsymbol{\Sigma})$. For each simulated path i, we obtain the associated terminal wealth $\hat{W}_{T,i}$ and utility $u(\hat{W}_{T,i})$. Then the expected utility of terminal wealth is estimated by the sample mean of realized utility across all simulated paths:

$$EU_{policy} = \frac{1}{I}\sum_{i=1}^{I} u(\hat{W}_{T,i}), \tag{4.1}$$

where the policy could be FDP or BDP. The estimated certainty equivalent associated with the expected utility can be calculated by $CE_{policy} = \big(u^{-1}(EU_{policy})\big)$. The estimated CER over T periods related to the selected policy is then obtained by the certainty equivalent minus 1:

$$CER_{policy} = CE_{policy} - 1 = \left(u^{-1}(EU_{policy})\right) - 1. \tag{4.2}$$

4.1.1 Single Risky Asset - IID Returns

We start by applying the algorithm in Section 3.3.4 to the single-period portfolio choice problem between one risky asset and a risk-free asset of a CRRA investor with $\gamma = 5$ and a holding period of one month. This simple example has more pedagogic value than practical value. We first illustrate that the FDP policy converges to the BDP policy. Then we analyse the sensitivity of FDP policies to the algorithm parameters such as $M, Nsub, \lambda$ and ϵ.

Following the example studied in [7], the constant risk free rate is 6% per year and the stock excess returns are IID log-normal with a mean and volatility which match the sample moments[1]: mean = 0.73% and standard deviation = 4.42%.

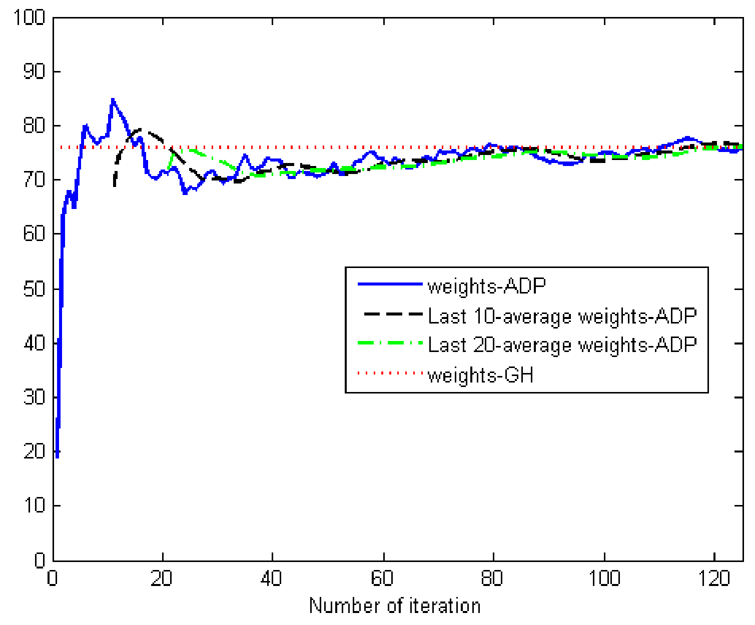
In this test, we set the number of inner loop, the number of sub-simulation, the stepsize in both phases, and the percentage of exploration in both phases to respectively $M = 100, Nsub = 100, \lambda_1 = \lambda_2 = 0.99, \epsilon_1 = \epsilon_2 = 100\%$. The tolerance parameter for stopping criteria (1.a) – 1.c) on page 38) is set to $\psi = 0.001$. The initial values for the coefficients $\boldsymbol{\theta}_t$ of value function approximations are set to 0.

Figure 4.1a shows the evolution of optimal allocations at $t = 0$ with respect to the number of iteration. The dotted line is the BDP policy $x_0^{BDP} = 76.01\%$. The solid line represents the optimal weights by FDP after iteration n, and the dashed line and the dotted-dashed line are respectively average weights of the last 10 and 20 iterations.

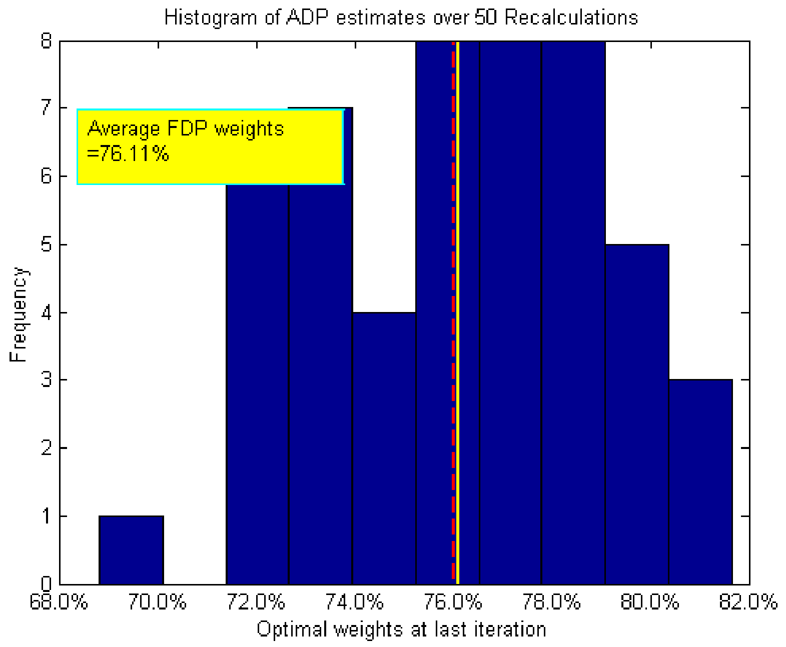
In this test, the algorithm stopped after $N = 125$ iterations. It can be observed from the figure that the FDP policy start to converge to the BDP policy after about 15 iterations, and it stabilizes around the BDP solution after about 50 iterations. The FDP weight at the last iteration $x_0^{(125)} = 76.02\%$ when the algorithm stopped is very close to the BDP weight.

It is worth to notice that, due to its simulation-based nature, the estimates of the

[1]Sample moments are derived from historical data on the value-weighted CRSP index from January 1986 to December 1995 by [7].



(a) Evolution of optimal weights at $t = 0$ as the number of iteration increases.



(b) Histogram of FDP optimal weights at $t = 0$ with 50 recalculations.

Figure 4.1: FDP results with CRRA utility single asset single period.

FDP algorithm vary from one test to another even using the same algorithm parameters. In fact, the performance of the FDP algorithm depends strongly on the "training data set". To illustrate this variability, we repeat the algorithm $K = 50$ times with the same parameters except that N is fixed to 200 instead of using the stopping criteria. Figure 4.1b presents the histogram of optimal weights at the last iteration $x_0^{(200),k}$, $k = 1, 2, \ldots, K$. The dashed line is the BDP solution and the solid line is the average of $x_0^{(200),k}$. Although there is some degree of deviation from one test to another, repeating the FDP algorithms several times leads to estimates very close to the conventional BDP solution.

4.1.2 Parameter Sensitivity

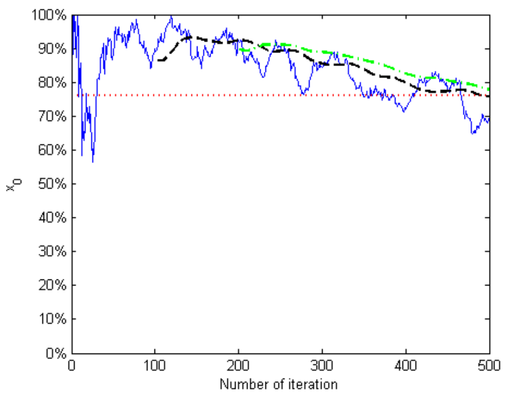
The performance of the FDP relies on appropriate choices of the algorithmic parameters:

- M: the number of inner loop which can also be viewed as the number of points in regression,
- $Nsub$: the number of paths in the sub-simulation to calculate the realized values on current path,
- λ: the stepsize, and
- ϵ: the exploration percentage.

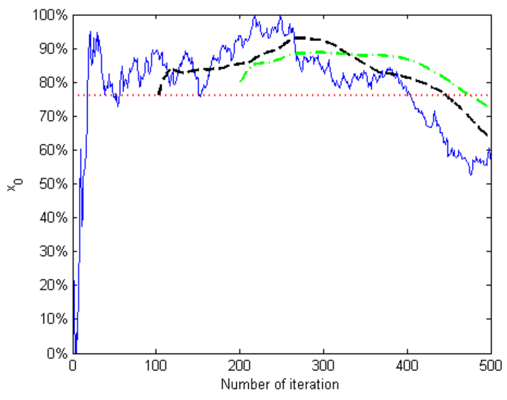
We use the simple single period and single asset example to study the sensitivity of the FDP policy with respect to these parameters. This study provides, at least qualitatively, a view on how each parameter impacts the behaviour of the forward algorithm.

Let's first examine M and $Nsub$ together since the joint effect of M and $Nsub$ determines the convergence of the FDP algorithm. We run the FDP algorithm several times by varying the value of M and $Nsub$ while keeping other parameters constant. Figure 4.2 summarizes results by plotting the dynamics of FDP estimates for optimal weights at $t = 0$ as the number of iteration increases. The three sub-figures on the left are obtained with $M = 1000$ and those on the right are results with $M = 10000$. The three lines correspond to experiences with $Nsub = 20$ and 200 respectively.

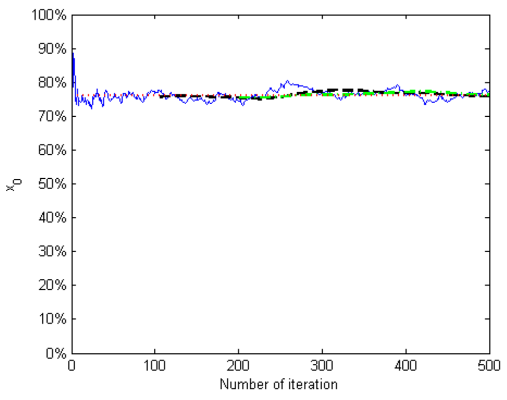
The FDP algorithm performs as expected: as M and $Nsub$ increase, the FDP estimates stabilize around the BDP solution within less iterations. Generally speaking, for the same value of M, the convergence is better when more paths are used in the



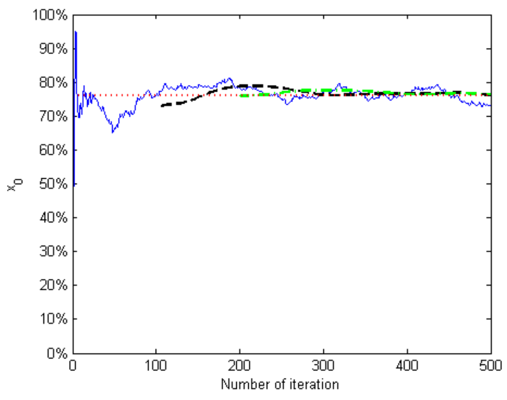
(a) $M = 1000, Nsub = 20$



(b) $M = 10000, Nsub = 20$



(c) $M = 1000, Nsub = 200$



(d) $M = 10000, Nsub = 200$

Figure 4.2: Sensitivity study of M and $Nsub$ with $N = 500, \lambda = 0.99, \epsilon = 100\%$.

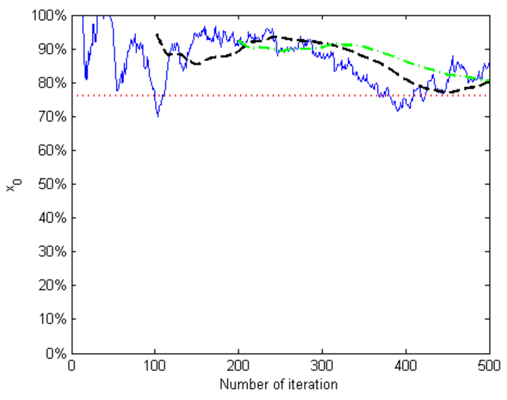
sub-simulation, i.e. larger $Nsub$. The same conclusion can be drawn for M. However, larger M and $Nsub$ requires more computation time. As a rule of thumb, we set

- $M = \min(100;\ 10*P\pm 10)$, where P is the number of coefficients θ_t to be estimated at each time step t,
- $Nsub \in [20, 200]$.

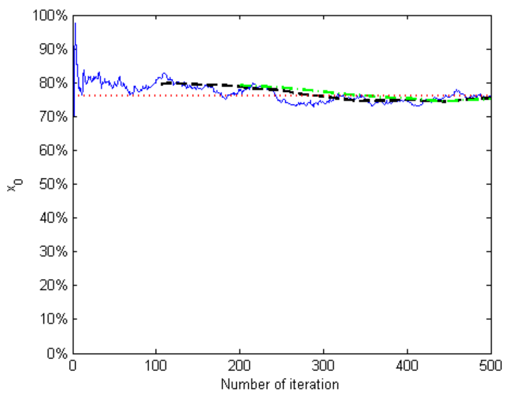
The next study is with respect to the exploration percentage ϵ. Figure 4.3 shows the dynamics of x_0 for three levels of exploration $\epsilon = 10\%, 50\%, 90\%$ while setting $N = 500, M = 100$ and $\lambda = 0.99$. The sub-figures on the left column are obtained with $Nsub = 20$ while those on the right are with $Nsub = 200$. In all the sub-figures, the dotted line stands for the BDP solution, the solid line represents the FDP estimates, and the dashed line and the dash-dotted line represent the average FDP estimates of the last 100 and 200 iterations respectively.

Figure 4.3 gives two interesting observations. First, when looking at the two sub-figures within the same line (i.e. tests with the same exploration level), it can be seen that FDP estimates are less volatile in the right sub-figure and they converge more quickly to the red dotted line which is the BDP solution. Therefore, it can be concluded that, regardless the exploration level, better convergence is achieved when more paths are used in the sub-simulation. The second observation is that the proposed FDP algorithm is robust to the exploration level if other parameters are appropriately tuned. In fact, when comparing the three figures in the same column (i.e. tests with the same $Nsub$), similar convergence patterns are observed for different levels of exploration ϵ. In this thesis, we set the exploration percentage in both phases to $\epsilon_1 = \epsilon_2 = 100\%$ in the following numerical examples.

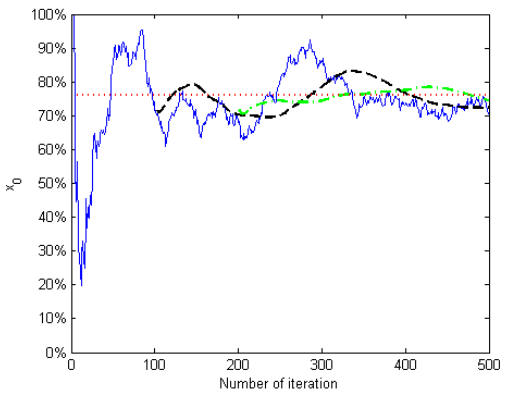
Finally, a similar study is performed to examine the sensitivity to the value of stepsize λ. Three values of $\lambda \in \{0.9, 0.95, 0.99\}$ are considered and other algorithmic parameters are set to $N = 500, M = 100, Nsub = 200, \epsilon = 100\%$. Figure 4.4 presents the test results. As anticipated, the FDP estimates are more volatile for smaller values of λ. Volatility is a desired feature in first iterations when the algorithm starts, because it reduces the risk that estimates are “absorbed” in a suboptimal state due to inaccurate value function approximations in early stage. However, as the iterations progress, the approximations become more and more reliable. It is thus necessary to stop the algorithm from “trusting” new information a lot and put more weight on existing approximations. These observations support the use of 2-phase strategy, where iterations are divided into



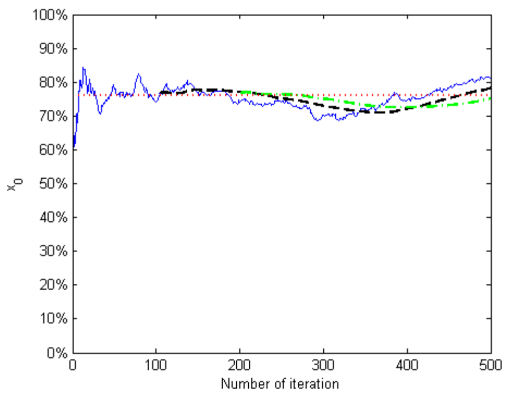
(a) $Nsub = 20, \epsilon = 10\%$



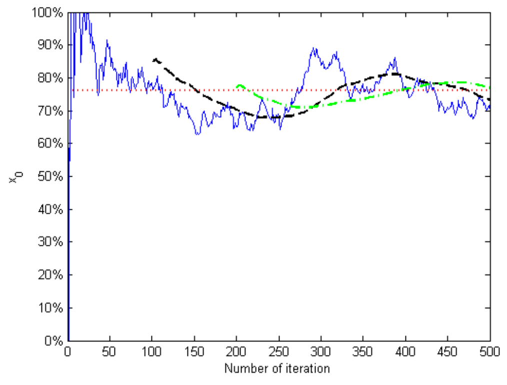
(b) $Nsub = 200, \epsilon = 10\%$



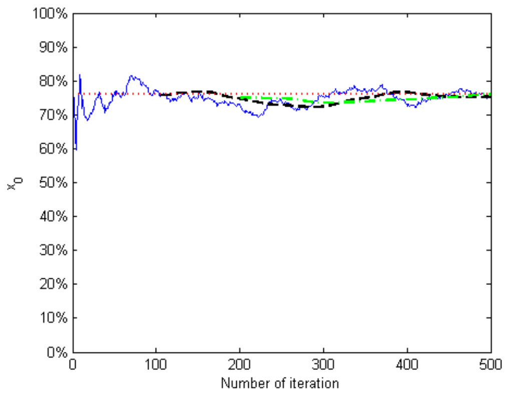
(c) $Nsub = 20, \epsilon = 50\%$



(d) $Nsub = 200, \epsilon = 50\%$

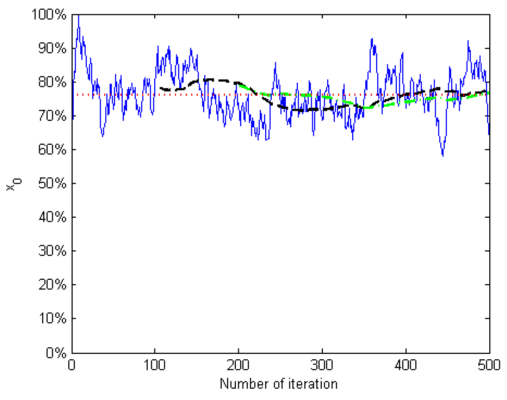


(e) $Nsub = 20, \epsilon = 90\%$

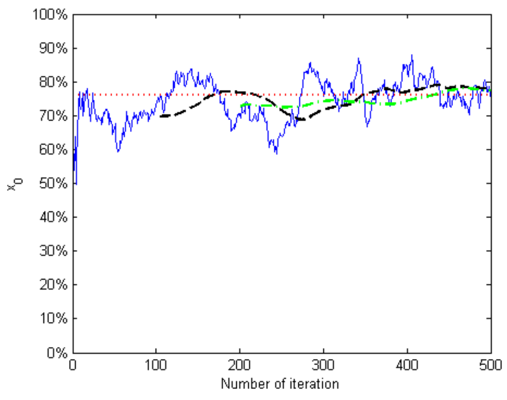


(f) $Nsub = 200, \epsilon = 90\%$

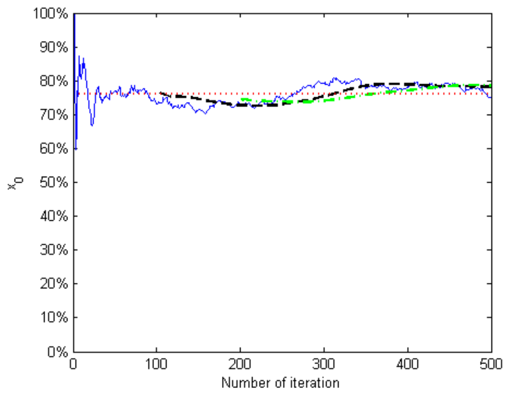
Figure 4.3: Sensitivity study of the percentage of exploration ϵ with $N = 500, M = 100, \lambda = 0.99$.



(a) $\lambda = 0.9$



(b) $\lambda = 0.95$



(c) $\lambda = 0.99$

Figure 4.4: Sensitivity study of the stepsize λ with $N = 500, M = 100, Nsub = 200, \epsilon = 100\%$.

two part and a different stepsize is used in each part[2]. This strategy will be used in examples with predictable returns in Section 4.2, where we set $\lambda_1 \in [0.9, 0.95]$ in the first phase, and $\lambda_2 = 0.99$ in the second phase.

4.1.3 Multiple Risky Assets - IID Returns

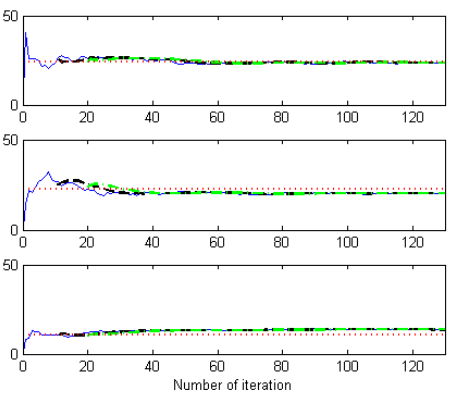
Now we consider the static OPCP of a CRRA investor examined in the section 2.3 of [26] where the investment opportunity set consists of the risk-free asset (cash) and three index for the United States, Europe, and Asia-Pacific respectively, obtained from Morgan Stanley Capital International (MSCI)-Barra. In this particular example, it is assumed that log excess returns, defined by $r_t := \log(R_t/R_f)$ where R_t is the excess returns, are normally distributed. Using a time series from December 1969 to July 2006, the estimated annual mean and covariance matrix of the log excess returns on the three index are

$$\mu = \begin{bmatrix} 0.0530 \\ 0.0620 \\ 0.0570 \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} 0.0263 & & \\ 0.0219 & 0.0324 & \\ 0.0183 & 0.0282 & 0.0714 \end{bmatrix}.$$

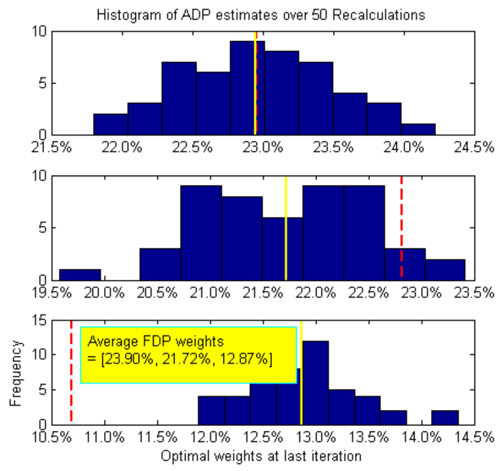
Applying the algorithm in Section 3.3.4 with $\gamma = 5, M = 100, Nsub = 200, \epsilon = 100\%$ and $\lambda = 0.99$, we obtain results as shown in Figure 4.5a. The dotted lines stand for the BDP policy: $\boldsymbol{x}_0 = [23.91\%, 22.82\%, 10.70\%]'$. The solid line is the FDP estimates and the dashed line and the dash-dotted line represent the average FDP estimates of the last 10 and 20 iterations respectively. The algorithm stopped after $N = 130$ iterations because the stopping criteria 1.a)-1.c) are fulfilled with $\psi = 0.001$. The FDP estimates at the final iteration are $\boldsymbol{x}_0^{(130)} = [24.25\%, 20.55\%, 12.95\%]'$, which is close to but still different from the BDP solution.

Since the FDP estimates vary from one test to another due to the dependence on the underlying datasets used to train the algorithm (as discussed in Section 4.1.1), we repeated the same test $K = 50$ times by fixing $N = 200$. Figure 4.5b presents the histogram of optimal weights at the last iteration $x_0^{(200),k}$, $k = 1, 2, \ldots, K$. The dashed line is the BDP solution and the solid line is the average of $x_0^{(200),k}$. Again, as in the single asset case, although there is some degree of deviation from one test to another, repeating the algorithm several times and taking the average can reduce the variability

[2]Please refer to Section 3.4 on page 40 for details.



(a) Optimal weights



(b) Histogram of $\boldsymbol{x}_0$ with 50 recalculations.

Figure 4.5: FDP estimates with CRRA utility and 3 risky assets, single period, $\gamma = 5, N = 100, M = 400, Nsub = 200, \epsilon = 100\%$ and $\lambda = 0.99$.

in FDP estimates.

It is possible to put the FDP's performance into question by looking at the histogram (Figure 4.5b) since the estimates for optimal weights in the 2nd and the 3rd assets are still distant from the BDP solution even averaging over 50 recalculations. In fact, the deviation from the BDP solution is not worrisome because in a multi-dimensional optimization, more than one local optimum may exist. Therefore, it is more common to compare certainty equivalent rates rather than comparing optimal weights themselves because policies which appear quite different in asset weights may lead to similar certainty equivalent rates.

We evaluate the certainty equivalent rates (CER) resulting from the BDP and from the FDP optimal policies at each iteration through Monte Carlo simulation as described on page 43 with a sample of $I = 10,000$ return paths. As illustrated by Figure 4.6, the CER of FDP (the solid line) increases quickly within first iterations and converges to the CER of the BDP policy (the dotted line)[3] which is $CER_{BDP} = 735.79$ basis points (bps). The CER of the FDP policy at the last iteration is $CER_{FDP} = 734.86$ bps. The difference in CERs between BDP and FDP policies is less than 1 bps when the FDP algorithm stopped despite the fact that their optimal weights are not exactly the same.

[3]Certainty equivalent rates are obtained with the same set of $100,000$ paths for BDP and FDP policies.

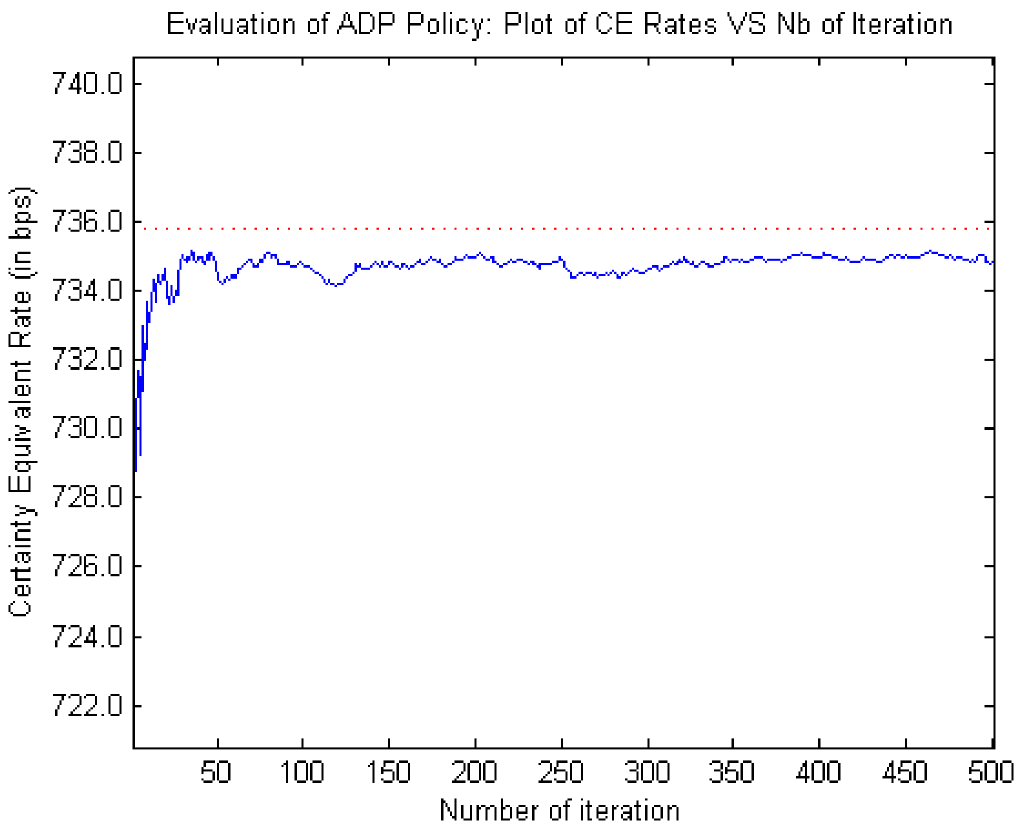


Figure 4.6: Certainty equivalent rates with CRRA utility and 3 risky assets, single period, $\gamma = 5, N = 100, M = 400, Nsub = 200, \epsilon = 100\%$ and $\lambda = 0.99$.

4.1.4 Multiple periods

As demonstrated in [25, 26], an important feature of OPCPs with CRRA utility function is that, when asset returns are iid, the multi-period problem reduces to a series of single period problems and the percentage in risky asset is the same for all periods. For this reason, rather than presenting multi-period results of cases with iid returns, we will study the performance of FDP in a multi-period environment with more complex dynamics for assets' returns in the following section.

4.2 Test 2: CRRA Utility with Predictable Asset Returns

This section examines the performance of the proposed forward algorithm within a multi-period environment where assets' returns can be predicted by the one-period lagged returns. We consider the custom-made VAR(1) dynamic used in [21] to test the robustness of an OPCP when several state variables are present:

$$\boldsymbol{R}_{t+1} = A_0 + A_1\boldsymbol{R}_t + \boldsymbol{\xi_{t+1}}, \tag{4.3}$$

where $\boldsymbol{\xi_{t+1}}$ is a vector of Gaussian error terms with zero mean and constant covariance matrix Σ. The vector A_0 is the drift term and A_1 is a time-invariant $N_a \times N_a$ matrix of parameters. The values of A_0, A_1 and Σ are given in Appendix B for the cases with 1, 2 and 3 risky assets.

This VAR(1) dynamics is interesting since it induces substantial differences between the single period and multi-period portfolio allocations. A restricted version of the VAR(1) return dynamic is studied by [12? , 36, 56, 25] and in [22]. In the restricted version, it is assumed that all asset returns can be predicted by one single exogenous variable which is the log dividend yield. Compared to the restricted VAR(1), the complete version studied in this part of the thesis allows for more complex interdependency structure among asset returns. More importantly, it shows the ability of the FDP algorithm to handle environments with more than one exogenous variable.

For the VAR(1) case, we use linear, quadratic and cross-product terms of decisions and cross-product terms of decisions-returns as well as the constant term to construct value function approximations. For instance, if there are three risky assets $N_a = 3$, the approximation of value function at each time step is represented by 19 basis $\phi_{t,p}$ and their associated coefficients $\theta_{t,p}$:

$$\widetilde{\mathcal{V}}_t^x(\boldsymbol{S}_t^x) = \boldsymbol{\theta}_t^{'}\boldsymbol{\Phi}_t, \tag{4.4}$$

where

$$\begin{aligned}
\boldsymbol{\theta}_t &= [\theta_{t,1},\ \ldots,\ \theta_{t,19}]', \\
\boldsymbol{\Phi}(\boldsymbol{x}_t) &= [1,\ x_{t,1},\ x_{t,2},\ x_{t,3},\ x_{t,1}^2,\ x_{t,2}^2,\ x_{t,3}^2, \\
&\quad x_{t,1}x_{t,2},\ x_{t,1}x_{t,3},\ x_{t,2}x_{t,3},\ x_{t,1}R_{t,1},\ x_{t,1}R_{t,2},\ x_{t,1}R_{t,3}, \\
&\quad x_{t,2}R_{t,1},\ x_{t,2}R_{t,2},\ x_{t,2}R_{t,3},\ x_{t,3}R_{t,1},\ x_{t,3}R_{t,2},\ x_{t,3}R_{t,3}]'.
\end{aligned}$$

4.2.1 Optimal Weights and Certainty Equivalent Rate Comparison

Using the VAR(1) dynamic, we studied the performance of our FDP algorithm for a multi-period OPCP with 1, 2 and 3 risky assets respectively and compared to solutions obtained by a BDP approach using discretization and a generalisation of the GH quadrature as described in Appendix C.

For this multi-period OPCP, we performed tests with three investment horizons $T = 2,\ 6,\ 12$ and with three levels of risk aversion $\gamma = 2,\ 5,\ 10$, Then we noticed that:

a) the FDP approach is robust to the risk aversion level. It behaves similarly for various levels of risk aversion and offers comparable optimal weights and certainty equivalent rate to the BDP approach. More importantly, for multi-asset cases ($N_a = 2,\ 3$) the optimal policies produced by the FDP yield higher certainty equivalent rates than the BDP approach;

b) the FDP performance is robust with respect to the investment horizon. Longer horizon requires more computation time for both approaches. Our tests suggest that the BDP approach is faster for shorter horizon $T = 1,\ 2$ in all cases (single-asset and multi-asset), while the FDP dominates in computation time for longer horizon, especially in the multi-asset cases.

For illustration purpose, we set the investment horizon to $T = 6$ (medium term) and the relative coefficient of risk aversion equals to $\gamma = 5$. The next step is to select parameters for both the BDP and FDP approaches.

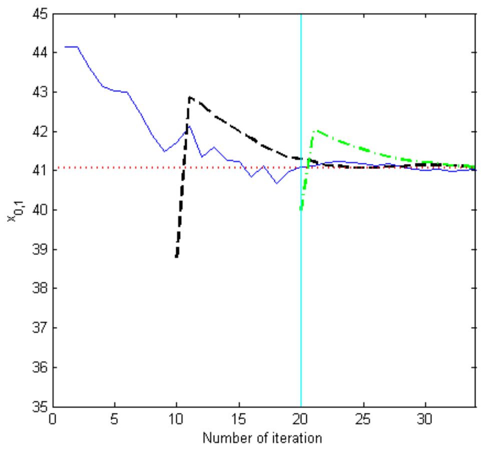
For the BDP approach, we will show the case where the state variables (lagged asset returns) are discretized into 20 points in each dimension and 10 nodes are used in GH quadrature. Tests with 10, 30, 40 discretization points are performed and we observed that a grid with 10 points is not enough to provide good precision for a medium term problem ($T = 6$), while grids with 30 or 40 points do not improve significantly the precision in term of optimal weights and certainty equivalent rates but consume too much time. The choice for number of nodes to use in the GH quadrature is quite common in the literature for Gaussian distributed variables. We performed tests by setting the number of nodes to 10, 20, 30, 40 and do not see significant variation in optimal weights nor certainty equivalent rates.

The FDP algorithmic parameters selected in our numerical tests are presented in Table 4.1 for cases with 1, 2, 3 risky assets. Some of them are slightly different when more assets (such more exogenous state variables) are involved. Note that we employed a pure exploration strategy in both phases $\epsilon = \epsilon_1 = \epsilon_2 = 100\%$. Our numerical tests suggest that other values of ϵ can also lead to convergence if the parameters $Nsub, M$ and λ are appropriately tuned. A pure exploration strategy is selected here from a time efficient point of view: in the current OPCP set-up, our simulation process (Matlab codes) is faster than the optimisation process (Matlab codes).

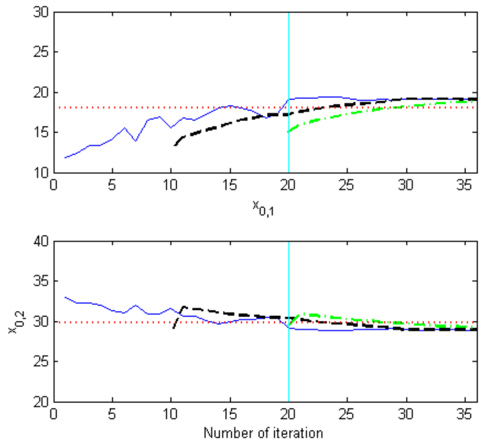
Table 4.1: FDP parameters for OPCP with VAR(1) return process

Parameter Name	Symbol	1 Asset	2 Assets	3 Assets
Nb. of iterations in inner loop	M	40	100	400
Nb. of paths in sub-simulation	$Nsub$	40	80	80
Stepsize (phase 1)	λ_1	0.9	0.9	0.9
Stepsize (phase 2)	λ_2	0.99	0.99	0.99
Nb. of iterations in phase 1	N_λ	20	20	20
Exploration percentage	$\epsilon_1 = \epsilon_2$	100%	100%	100%
Tolerance for stopping criteria	ψ	0.001	0.001	0.005

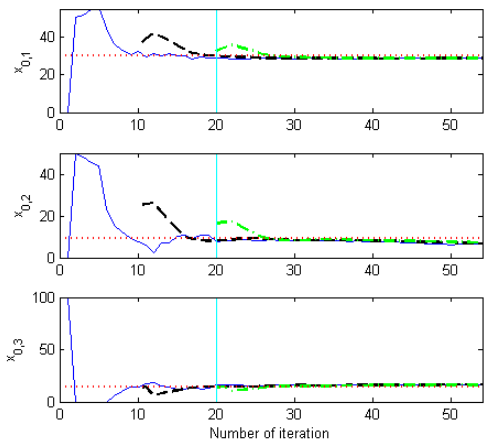
Using these parameters, we performed numerical tests with one to three risky assets $N_a = 1,\ 2,\ 3$. Figure 4.7 illustrates the evolution of FDP weights at $t = 0$ (the solid line) regarding the number of iterations. The dashed line and the dash-dotted line are respectively average weights of the last 10 and 20 iterations. This figure suggests that the FDP weights converge quickly to the BDP solution, which is indicated by the dotted



(a) $N_a = 1$



(b) $N_a = 2$



(c) $N_a = 3$

Figure 4.7: Comparison of FDP weights and BDP weights for VAR(1) returns.

line. Comparative results of optimal weights at $t = 0$ and computational time[4] as well as certainty equivalent rates are shown in Table 4.2. The certainty equivalent rates are obtained through Monte Carlo simulation such as described on page 44. Specifically, we simulate a sample of $I = 5000$ return paths according to the VAR(1) dynamic with antithetic variables for variance reduction, and computed CER_{BDP} and CER_{FDP} by equations (4.1 – 4.2). The FDP policy evaluated is the one obtained at the last iteration when the algorithm is terminated by stopping criteria 1.a)-1.c). The last line of Table 4.2 shows the number of iterations performed (N) before these criteria are fulfilled in each test.

Table 4.2: Comparative results of FDP and BDP algorithms for VAR(1) returns with $1, 2$ and 3 risky assets and $T = 6$

	1 Asset		2 Assets		3 Assets	
	BDP	FDP	BDP	FDP	BDP	FDP
$x_{0,1}$	41.06%	41.02%	18.04%	14.23%	30.07%	28.82%
$x_{0,2}$	-	-	29.84%	31.46%	9.29%	6.75%
$x_{0,3}$	-	-	-	-	14.28%	16.46%
Time (sec)	3.80	3.93	116.62	134.41	5,086.94	2,002.32
N	-	34	-	36	-	54
CER	373.75	373.09	443.56	447.57	464.31	477.41

From results presented in Table 4.2, several interesting observations can be made. Three of them are highlighted below. First of all, for the case of single risky asset, the FDP approach yields optimal weights and estimated CER very close to those obtained by the BDP approach within comparable time[5].

Secondly and more importantly, for multi-asset cases, the FDP approach produces policies which lead to higher estimated CER than the BDP approach and within less computational time. Actually, the estimated CER related to FDP policies CER_{FDP} exceeds the one related to BDP policies by about 4 bps for the 2 risky assets case and 13 bps for the 3 risky assets case.

Thirdly, our FDP approach has an advantage over the BDP approach regarding the calculation time: as the number of risky assets N_a increases, the FDP computational time increases much less dramatically compared to the BDP computational time. As

[4]Computation time is obtained on a personal laptop with Intel Core i3-2350M CPU @ 2.30 GHz and 8.00 GB RAM. No parallel computing.

[5]Certainty equivalent rates are obtained by evaluating the FDP and BDP policies using the same sample.

an illustration, when N_a passes from 2 to 3, the BDP run time jumps from 116 seconds to 5,087 seconds. This phenomena is the well-known *curse of dimensionality* due to the need of discretizing over the state space as discussed in the beginning of Section 3.1. On the other side, the increase of FDP run time is relatively moderate – from 134 seconds to 2,002 seconds, which represents only about 40% of the BDP run time. Therefore, the FDP approach can be potentially more suitable for OPCPs with three or more exogenous state variables.

Studies of parameters' sensitivity are also conducted under this example with VAR(1) return process. We obtain similar conclusions as in the IID case which are summarized below[6]:

- M: the number of iterations in the inner loop, which can also be viewed as the number of points available for regression. Must be large enough to ensure statistical significance. A rule of thumb is to set M approximatively equal to $\min(100; 10*P)$, where P is the number of coefficients θ_t to be estimated at each time step t.
- $Nsub$: the number of paths in the sub-simulation. Larger values of $Nsub$ decrease noises in input signals (realized values), and thus improve the algorithm's estimation performance. We propose to set $Nsub \in [40, 200]$. Qualitatively, $Nsub$ should increase as the number of risky assets N_a increases, but the relation does not seem to be strictly proportional. Larger values of $Nsub$ require more computational time.
- λ: the stepsize. We adopted the 2-phase stepsize rule in numerical tests of the current section. We tested $\lambda_1 = [0.9,\ 0.95,\ 0.99], \lambda_2 = [0.9,\ 0.95,\ 0.99]$. Our numerical tests suggest that $\lambda_1 = 0.9$ and $\lambda_2 = 0.99$ works the best. Other combinations also lead to convergence (in terms of optimal weights) but require more iterations.
- ϵ: the percentage of exploration. Our numerical tests suggest that the impact of this parameter is not material if other parameters $(M, Nsub, \lambda)$ are properly tuned. Since exploitation is more time consuming, we choose a pure exploration strategy.

[6]Details results and graphs are available upon demande.

Chapter 5

Conclusion on Part II

We propose a recursive forward approach for dynamic programming that relies on simulation and regression with respect to both state variables and decision variables. The tools used in our approach are simple and well researched: Monte Carlo simulation, ordinary least-squares, continuous optimization, approximation by polynomial basis, and exponential smoothing. The forward approach, however, is little known in the area of portfolio optimization. To the best of our knowledge, our study is one of the first explorations to apply forward algorithms for a variety of portfolio choice problems. Implemented with care, the algorithm is very flexible, robust, and its performance degrades at a much slower rate than a standard quadrature approach, when the number of dimension increases. It is worth noting that, the simulation process is independent from the decision process. Even Monte Carlo simulation is used in our algorithm, the external simulator need not to be explicitly modelled: it can be either black-boxed or bootstrapping with historical data. Moreover, the computation time of the forward approach increases much less quickly compared to a standard BDP approach with quadrature, and thus provide a possible way to handle, at least partially, the curses of dimensionality.

Part III

Optimal Portfolio Choice Problems with Predictable Johnson-S_U Returns

Chapter 6

Literature Review for Part III

Multi-period dynamic OPCPs with predictable returns have been intensively studied in recent literature. One of the main streams in the literature, pioneered by [43] and followed by for example [17], [18], focuses on developing models in continuous time. The main advantage of continuous time modelling is its tractability, which means that, under usual conditions and assumptions, solutions to the OPCP exist in closed-form. However, the real world seems to be more in concordance with a discrete time modelling because a portfolio cannot be rebalanced continuously at every instant of time. One may argue that continuous time modelling can be approximated by trading at extremely high frequency, but the resulting transaction costs due to liquidity problems (or market depth) may be exceptionally large thus deteriorating considerably a trading strategy's performance.

In the discrete modelling stream of OPCPs, following the mean-variance paradigm of [40] which is a single-period set up, a number of papers address the multi-period OPCP where an investor has the option to rebalance dynamically his portfolio allocation when future returns of risky assets can be predicted. The predictability of asset's returns can change considerably the optimal allocation in a multi-period setting, compared to a single-period case. [32], [8] and [3] explore the impacts of ignoring the predictability in a multi-period setting by comparing a myopic policy (where one optimizes only for the following period) and a dynamic policy (where one optimizes until the end of planning horizon). [13] use log-linear approximations to obtain a closed-form solution to the multi-period portfolio choice problem with return predictability and intermediate consumption;

[38] examine the same problem but with transaction costs. A key motivation of taking predictability into consideration is to achieve a better allocation when one has more information about the future. Furthermore, empirical studies suggest that asset's returns exhibit mean-reversion. Thus a multi-period setting, which takes long term developments of asset's returns into account, may help to smooth optimal allocations over time.

But are asset's returns really predictable? Numerous recent empirical works show that stock returns exhibit a certain predictability by a set of lagged instruments. For example, [11] finds that the risk premium on stocks move closely together with those on 20-year T-bonds. [23] find that the dividend yield, a quantity that is related to the default spread and moves in a similar way with long-term business conditions, forecasts future stock returns.

Therefore, we choose a restricted vector auto-regressive (VAR) process for asset returns as discussed by [32], [13], [3], [38], [37], [57], [25], [6] and [39]. This model assumes that asset's returns, as well as the log-dividend yield, are both forecast by the one period lagged log-dividend yield. It is preferred by the literature due to its simplicity and ability to include a certain degree of predictability. Standard analysis of OPCPs with the restricted VAR(1) model is based on the assumption of normal innovations. This assumption is convenient since it reduces computation burdens in finding optimal solutions to the OPCP thanks to some nice properties of Gaussian distribution. In particular, Gaussian distribution allows one to compute expectations in Bellman equation efficiently by quadrature methods such as Gauss-Hermite quadrature ([31]). However, Gaussian distribution is not always suitable for financial time series modelling due to its inability to capture the typical empirical features of stock returns such as leptokurtic and heavy tails, especially when data frequency is high (monthly, daily or intra-day returns). A long list of papers has established that many financial assets exhibit clearly non-Gaussian log-returns. Among others, [24], [9], [1] demonstrate that distributions of hedge funds returns have a negative skewness and an excess kurtosis larger than zero using monthly data.

These drawbacks of Gaussian distribution lead us to choose an alternative distribution for log-returns – the Johnson-S_U. This distribution with four-parameters, first investigated in 1949 by [29], is proposed as a continuous transformation of a standard Normal variable. [15] and [54] both examine GARCH processes with Johnson-$S_U$distribution, and find that the Johnson-$S_U$distribution outperforms the normal and student-t distributions in describing the shape and extreme tails of daily returns. In fact, the Johnson distribution family can calibrate independently the four first moments “while disregarding potentially insignificant moments of order higher than 5”, as remarked by [49] and highlighted in [45].

We improve the restricted VAR(1) model by replacing Gaussian innovations by multivariate Johnson-$S_U$innovations, which will be presented in Chapter 7, and solve multi-period OPCPs with CRRA utility by backward dynamic programming using Gauss-Hermite quadrature to calculate expectations, which will be presented in Chapter 8. Numerical experiments are presented in Chapter 10, where we compare policies resulting from Gaussian and Johnson-$S_U$distributions with one and two risky assets, as well as with intermediate consumption. Section 10.4 compares performance of Johnson and Gaussian policies by first conducting a scenario analysis in subsection 10.4.1 and then performing out-of-sample tests in Subsection 10.4.2. Chapter11 concludes. Our major contribution is to explore how non-Gaussian log-returns can alter a typical investor’s decision when returns are predictable.

Chapter 7

Market Model – Restricted VAR

This chapter describes our market model, a restricted vector auto-regressive (VAR) process for log-excess returns and log-dividend yield. We first define the general process with arbitrary hypothesis on assets' return distribution. Then sections 7.2 and 7.3 present special cases with Gaussian and Johnson-S_U noises respectively. In these two sections, we indicate how Gauss-Hermite quadrature can be used to compute conditional expectations efficiently.

7.1 Restricted vector auto-regressive process

The market model studied in this chapter has been used extensively in the portfolio choice literature (see the introduction). Let R_f denote the constant risk free rate of return at time t and $R^g_{i,t}$ the one period **gross return** for asset i at time t. The **excess return** and **log excess return** for asset i are then defined respectively by $R_{i,t} := R^g_{i,t} - R_f$, and $r^e_{i,t} = \ln(1 + R_{i,t})$ respectively. The model for the dynamic of log excess returns of N_a risky assets is given by:

$$r^e_{i,t+1} = \alpha_i + \beta_i \delta_t + \sigma_i \nu_{i,t+1}, \qquad i = 1, \ldots, N_a \tag{7.1}$$

where α_i, β_i and $\sigma_i > 0$ are constant parameters, and the $\nu_{i,t}$ for $i = 1, \ldots, N_a$ are zero-mean, unit-variance random noises with correlations ρ_{ij}. Here δ_t is the log dividend yield of a stock index, defined as

$$\delta_t = \ln(1 + D_t),$$

where D_t is the dividend-to-price ratio of a stock index (dividend yield of a stock index), which is computed as the sum of the past 12 months dividends divided by the current level of the index (we refer to [6] for this definition). It is assumed that the dividend yield[1] has an AR(1) dynamics, written as:

$$\delta_{t+1} = \alpha_\delta + \beta_\delta \delta_t + \sigma_\delta \nu_{\delta,t+1}$$

where α_δ, β_δ and $\sigma_\delta > 0$, are constant parameters and $\nu_{\delta,t+1}$ is a zero-mean, unit-variance error term. Stacking up the above $N_a + 1 = n$ equations, and dropping the time subscripts for notational convenience, we can write the resulting system in matrix form as

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\sigma_\nu}\ \boldsymbol{\nu} \tag{7.2}$$

[1]Note that for the sake of simplicity, we shall refer to “log excess returns” and “log dividend yield” respectively as “returns” ad “dividend yield” (or simply “dividend”).

where $\mathbf{y}$, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are the following $n \times 1$ vectors

$$\mathbf{y} = \begin{bmatrix} r^e_{1,t+1} \\ \vdots \\ r^e_{N,t+1} \\ \delta_{t+1} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \alpha_1 + \beta_1 \ \delta_t \\ \vdots \\ \alpha_N + \beta_N \ \delta_t \\ \alpha_\delta + \beta_\delta \ \delta_t \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_{1,t+1} \\ \vdots \\ \nu_{N,t+1} \\ \nu_{\delta,t+1} \end{bmatrix}$$

and $\boldsymbol{\sigma}_{\boldsymbol{\nu}}$ is a $n \times n$ diagonal matrix of standard error parameters, and where $\boldsymbol{\nu}$ has correlation matrix $\mathbf{C}_{\boldsymbol{\nu}}$. It is worth noting that the distribution of the noise vector $\boldsymbol{\nu}$ has not been identified yet. In the next two sections, we consider the two cases where the noise vector $\boldsymbol{\nu}$ follows a Gaussian distribution and a Johnson-$S_U$distribution, and how Gauss-Hermite quadrature can be applied.

7.2 Gaussian noises

In the literature, it is usually assumed that the elements of the noise vector are jointly Gaussian-distributed $\boldsymbol{\nu} \sim \mathbf{z}$, where $\mathbf{z}$ is multivariate Gaussian with zero-mean and correlation matrix $\mathbf{C}_\mathbf{z}$. The density function of this standard normal vector can be written as:

$$f(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{C}_\mathbf{z})}} \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{C}_\mathbf{z}^{-1}\mathbf{z}\right). \tag{7.3}$$

Using equation (7.2) and a multivariate change of variable[2], the density of $\mathbf{y}$ is given by

$$f(\mathbf{y}) = f(\mathbf{z}) \times \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right|, \tag{7.4}$$

where $\frac{\partial \mathbf{z}}{\partial \mathbf{y}}$ is the Jacobian matrix, $\det(\cdot)$ is the determinant of a matrix and $|a|$ is the absolute value of a. Equation (7.2) implies $\mathbf{z} = \boldsymbol{\sigma}_\mathbf{z}^{-1}(\mathbf{y} - \boldsymbol{\mu})$, thus $\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \boldsymbol{\sigma}_\mathbf{z}^{-1}$. By defining $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_\mathbf{z}\mathbf{C}_\mathbf{z}\boldsymbol{\sigma}'_\mathbf{z}$ and by replacing $\frac{\partial \mathbf{z}}{\partial \mathbf{y}}$ and (7.3) in (7.4), we obtain the density function of

[2]This is the technique *integration by substitution*, which can be found in any textbook on Calculus such as [31], Section 7.2 and [16], Chapter IV.

y

$$f(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{C_z}) \det(\boldsymbol{\sigma}_\mathbf{z})}} \exp\left(-\frac{1}{2}\left(\boldsymbol{\sigma}_\mathbf{z}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)' \mathbf{C}_\mathbf{z}^{-1}\left(\boldsymbol{\sigma}_\mathbf{z}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right)\right)$$

$$= \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right).$$

Thus the vector $\mathbf{y}$ is multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. We can write the expectation of a continuous function $g(\mathbf{y})$ of $\mathbf{y}$ as:

$$E[g(\mathbf{y})] = \int_{\mathcal{R}_n} g(\mathbf{y}) f(\mathbf{y})\, d\mathbf{y},$$

$$= \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}} \int_{\mathcal{R}_n} g(\mathbf{y}) \exp\left(-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right) d\mathbf{y}. \tag{7.5}$$

As discussed in [31] (section 7.2), this last integral can be approximated by Gauss-Hermite (GH) quadrature. To do this, we define the change of variable

$$\mathbf{x} = \boldsymbol{\Omega}^{-1}(\mathbf{y}-\boldsymbol{\mu})/\sqrt{2} \quad \longleftrightarrow \quad \mathbf{y} = \sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu} \tag{7.6}$$

where $\boldsymbol{\Omega}$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$ with:

$$\boldsymbol{\Sigma} = \boldsymbol{\Omega}\boldsymbol{\Omega}' \quad \text{and} \quad \boldsymbol{\Sigma}^{-1} = \left(\boldsymbol{\Omega}^{-1}\right)' \boldsymbol{\Omega}^{-1}.$$

The vector $\mathbf{x}$ is a zero-mean and unit-variance multivariate Gaussian variable within-dependant elements. The change of variable in (7.6) puts equation (7.5) ready for GH quadrature integration:

$$\begin{aligned}
E\left[g\left(\mathbf{y}\right)\right] = & \frac{1}{\sqrt{(2\pi)^n \det\left(\boldsymbol{\Sigma}\right)}} \times \\
& \int_{\mathcal{R}_n} g\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu}\right) \exp\left(-\frac{1}{2}\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu}\right)' \boldsymbol{\Sigma}^{-1}\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu}\right)\right)\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)\right| d\mathbf{x}, \\
= & \frac{1}{\sqrt{(2\pi)^n \det\left(\boldsymbol{\Sigma}\right)}} \int_{\mathcal{R}_n} g\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu}\right) \exp\left(-\mathbf{x}'\mathbf{x}\right) \sqrt{2^n}\left|\det\left(\boldsymbol{\Omega}\right)\right| d\mathbf{x}, \\
= & \frac{\left|\det\left(\boldsymbol{\Omega}\right)\right|}{\sqrt{\pi^n \det\left(\boldsymbol{\Sigma}\right)}} \int_{\mathcal{R}_n} g\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{x}+\boldsymbol{\mu}\right) \exp\left(-\mathbf{x}'\mathbf{x}\right) d\mathbf{x}, \\
\approx & \frac{\left|\det\left(\boldsymbol{\Omega}\right)\right|}{\sqrt{\pi^n \det\left(\boldsymbol{\Sigma}\right)}} \sum_{k_1=1}^{Q_1} \cdots \sum_{k_n=1}^{Q_n} g\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{m}+\boldsymbol{\mu}\right)\ p_{k_1} \ldots p_{k_n},
\end{aligned} \tag{7.7}$$

where $\mathbf{m} = [m_{k_1}\ , \ \ldots\ , \ m_{k_n}]'$ is a vector of GH nodes for all n dimensions with m_{k_i} being one of the Q_i nodes in the i^{th} dimension and p_{k_i} being the corresponding weights ($k_i = 1, \ldots, Q_i$). The second line follows by the fact that (due to the change of variable in (7.6))

$$\left|\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right| = \begin{vmatrix} \frac{\partial y_1}{\partial x_1} & 0 & 0 & 0 \\ 0 & \frac{\partial y_2}{\partial x_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{\partial y_n}{\partial x_n} \end{vmatrix} = \sqrt{2^n}\left|\det\left(\boldsymbol{\Omega}\right)\right|.$$

7.3 Johnson-S_U noises

It is well known by practitioners and researchers who have handled financial market data that asset returns do not behave according to the Gaussian distribution. Thus "the use of Gaussian models when the asset return distributions are not normal could lead to a wrong choice of portfolio, the underestimation of extreme losses or mispriced derivative products" (stated by [30]). Consequently, non-Gaussian models and models based on processes with jumps have gained popularity among financial market practitioners.

In this section, in order to introduce non-normality, we assume that the noise vector in equation (7.2) follows a multivariate Johnson-$S_U$distribution, which means that $\boldsymbol{\nu} \sim \mathbf{u}$ and

$$\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\sigma}_{\mathbf{u}}\mathbf{u}$$

where the elements of $\mathbf{u}$ are standard JJohnson-$S_U$random variables with zero-mean, unit-variance and correlation matrix $\mathbf{C_u}$. These standard Johnson-$S_U$random variables are continuous transformation of $\mathbf{C_z}$-correlated standard normal random variables z_i

$$\begin{aligned} & u_i = c_i + d_i \ \sinh\left(\frac{z_i - a_i}{b_i}\right) \\ \longleftrightarrow \quad & z_i = a_i + b_i \ \sinh^{-1}\left(\frac{u_i - c_i}{d_i}\right) \quad i = 1, \ldots, n \end{aligned} \tag{7.8}$$

where a_i , $b_i > 0, c_i, d_i > 0$ are the four parameters characterizing a Johnson-$S_U$distribution, and $\boldsymbol{\sigma}_{\mathbf{u}}$ is a diagonal matrix of standard error parameters. The z_i are standard normal random variables with correlations given by $\mathbf{C_z}$. The covariance matrix of $\mathbf{y}$ is given by $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_{\mathbf{u}}\mathbf{C_u}\boldsymbol{\sigma}_{\mathbf{u}}$. It is important to notice that the correlation between z_i and z_j is different from the correlation between u_i and u_j, i.e. $\mathbf{C_z} \neq \mathbf{C_u}$, because of the non linear transformation in (7.8). Appendix D shows how to derive ρ_{z_i,z_j} (the correlation between normal variables) from ρ_{u_i,u_j} (the correlation between Johnson-$S_U$variables) and vice-versa.

Using transformation (7.8), the relationship between y_i and z_i can be written as

$$\begin{aligned} & y_i = \mu_i + \sigma_{u,i}\left(c_i + d_i \ \sinh\left(\frac{z_i - a_i}{b_i}\right)\right) \\ \longleftrightarrow \quad & z_i = a_i + b_i \ \sinh^{-1}\left(\frac{\left(\frac{y_i - \mu_i}{\sigma_{u,i}}\right) - c_i}{d_i}\right) \quad i = 1, \ldots, n \end{aligned} \tag{7.9}$$

which can be generically written as the following multivariate mapping:

$$\mathbf{y} = \mathbf{h}(\mathbf{z}) \quad \longleftrightarrow \quad \mathbf{z} = \mathbf{h}^{-1}(\mathbf{y}), \tag{7.10}$$

where $\mathbf{h}(\mathbf{z}) = [h_1(z_1) \ , \ \ldots \ , \ h_n(z_n)]'$, with $h_i(z_i) = \mu_i + \sigma_{u,i}\left(c_i + d_i \ \sinh\left(\frac{z_i - a_i}{b_i}\right)\right)$ for $i = 1, \ldots, n$, and $\mathbf{h^{-1}}(\mathbf{y}) = [h_1^{-1}(y_1) \ , \ \ldots \ , \ h_n^{-1}(y_n)]'$.

Similarly as in section 7.2, the density function of $\mathbf{y}$ can be obtained by the multivariate change of variable as described in equation (7.9). The resulting density function can be written as

$$
\begin{aligned}
f(\mathbf{y}) &= f(\mathbf{z}) \times \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right|, \\
&= \frac{1}{\sqrt{\det(\mathbf{C_z})(2\pi)^n}} \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{C}_\mathbf{z}^{-1}\mathbf{z}\right) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right|, \\
&= \frac{1}{\sqrt{\det(\mathbf{C_z})(2\pi)^n}} \exp\left(-\frac{1}{2}\mathbf{h}^{-1}(\mathbf{y})'\,\mathbf{C}_\mathbf{z}^{-1}\mathbf{h}^{-1}(\mathbf{y})\right) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right|,
\end{aligned}
$$

where $f(\mathbf{z})$ is given by (7.3) and $\dfrac{\partial \mathbf{z}}{\partial \mathbf{y}}$ is the diagonal Jacobian matrix.

Given this density function of $\mathbf{y}$, one can compute the expectation of an arbitrary function $g(\mathbf{y})$:

$$
\begin{aligned}
E[g(\mathbf{y})] &= \int_{\mathcal{R}_n} g(\mathbf{y})\, f(\mathbf{y})\, d\mathbf{y}, \\
&= \frac{1}{\sqrt{\det(\mathbf{C_z})(2\pi)^n}} \int_{\mathcal{R}_n} g(\mathbf{y}) \ \exp\left(-\frac{1}{2}\mathbf{h}^{-1}(\mathbf{y})'\,\mathbf{C}_\mathbf{z}^{-1}\mathbf{h}^{-1}(\mathbf{y})\right) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right| d\mathbf{y}.
\end{aligned}
$$

The second line in previous equation is obtained by simply replacing the expression for the density of $\mathbf{y}$. Since $\mathbf{y} = \mathbf{h}(\mathbf{z})$ and $h^{-1}(h(\mathbf{z})) = \mathbf{z}$ as defined previously in equation (7.10), we have

$$
\begin{aligned}
E[g(\mathbf{y})] &= \frac{1}{\sqrt{\det(\mathbf{C_z})(2\pi)^n}} \int_{\mathcal{R}_n} g(\mathbf{h}(\mathbf{z})) \ \left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{z}}\right)\right| \ \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{C}_\mathbf{z}^{-1}\mathbf{z}\right) \left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right| d\mathbf{z}, \\
&= \frac{1}{\sqrt{\det(\mathbf{C_z})(2\pi)^n}} \int_{\mathcal{R}_n} g(\mathbf{h}(\mathbf{z})) \ \exp\left(-\frac{1}{2}\mathbf{z}'\mathbf{C}_\mathbf{z}^{-1}\mathbf{z}\right) d\mathbf{z} \qquad (7.11)
\end{aligned}
$$

since $\left|\det\left(\dfrac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)\right| \times \left|\det\left(\dfrac{\partial \mathbf{y}}{\partial \mathbf{z}}\right)\right| = 1.$

Equation (7.11) can be reorganized in order to use GH quadrature integration by defining a vector $\mathbf{x}$ as

$$
\mathbf{x} = \boldsymbol{\Lambda}^{-1}\mathbf{z}/\sqrt{2} \quad \longleftrightarrow \quad \mathbf{z} = \sqrt{2}\boldsymbol{\Lambda}\mathbf{x}
$$

where $\mathbf{C_z} = \mathbf{\Lambda\Lambda}'$ and $\mathbf{C_z}^{-1} = \left(\mathbf{\Lambda}^{-1}\right)' \mathbf{\Lambda}^{-1}$, which yields the following expression, that is amenable to Gauss-Hermite quadrature approximation

$$
\begin{aligned}
E\left[g\left(\mathbf{y}\right)\right] = & \frac{1}{\sqrt{\det\left(\mathbf{C_z}\right)\left(2\pi\right)^n}} \\
& \int_{\mathcal{R}_n} g\left(\mathbf{h}\left(\sqrt{2}\mathbf{\Lambda x}\right)\right)\exp\left(-\frac{1}{2}\left(\sqrt{2}\mathbf{\Lambda x}\right)'\mathbf{C_z}^{-1}\left(\sqrt{2}\mathbf{\Lambda x}\right)\right)\left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right| d\mathbf{x} \\
= & \frac{1}{\sqrt{\det\left(\mathbf{C_z}\right)\left(2\pi\right)^n}}\int_{\mathcal{R}_n} g\left(\mathbf{h}\left(\sqrt{2}\mathbf{\Lambda x}\right)\right)\exp\left(-\mathbf{xx}\right)\left|\det\left(\mathbf{\Lambda}\right)\right| 2^{n/2} d\mathbf{x} \\
= & \frac{\left|\det\left(\mathbf{\Lambda}\right)\right|}{\sqrt{\det\left(\mathbf{C_z}\right)\pi^n}}\int_{\mathcal{R}_n} g\left(\mathbf{h}\left(\sqrt{2}\mathbf{\Lambda x}\right)\right)\exp\left(-\mathbf{xx}\right) d\mathbf{x} \\
\approx & \frac{\left|\det\left(\mathbf{\Lambda}\right)\right|}{\sqrt{\det\left(\mathbf{C_z}\right)\pi^n}}\sum_{k_1=1}^{Q_1}\cdots\sum_{k_n=1}^{Q_n} g\left(\mathbf{h}\left(\sqrt{2}\Lambda\mathbf{m}\right)\right)\times p_{k_1}\ldots p_{k_n} \qquad (7.12)
\end{aligned}
$$

where $n = N+1$ with N being the number of risky assets in the portfolio; $\mathbf{h}$ is defined in equation (7.10); and as in section 7.2, $\mathbf{m} = [m_{k_1}\ ,\ \ldots\ ,\ m_{k_n}]'$ is a vector for GH nodes in all n dimensions with m_{k_i} being one of the Q_i GH nodes in the i^{th} dimension and p_{k_i} being its corresponding weight, $i = 1, \ldots, n$.

Chapter 8

A Gauss-Hermite Algorithm for a Non-Gaussian OPCP

8.1 Problem formulation with CRRA utility function

We now turn our attention to optimal portfolio choice problems. Precisely, we study optimal portfolio allocations of an investor whose risk aversion can be described by the constant relative risk aversion (CRRA) utility function for the market model discussed in previous section. Mathematically, we want to maximize the expected utility of the final wealth by choosing the optimal asset allocations over an investment horizon of T periods

$$\max_{\boldsymbol{x}_0,\dots,\boldsymbol{x}_{T-1}} \mathbb{E}[u(W_T)], \tag{8.1}$$

where the wealth transits according to $W_{t+1} = W_t\left(R_f + \boldsymbol{x}_t(e^{\mathbf{r}^e_{t+1}} - \mathbb{I})\right)$. Here $\mathbf{0} \leq \boldsymbol{x}_t \leq \mathbb{I}$ is the vector of optimal weights in risky assets for period $[t, t+1), t = 0, \dots, T-1$; $\mathbb{I}$ is the vector of ones the appropriate size and $u(W_t) = \frac{W_t^{1-\gamma}}{1-\gamma}$ is the CRRA utility function, and its parameter γ controls the degree of risk aversion: the larger is its value, the greater is the risk aversion.

This optimization problem can be solved by dynamic programming. Define the value function by

$$V_t(W_t, \mathbf{y}_t) := \max_{\boldsymbol{x}_t,\dots,\boldsymbol{x}_{T-1}} \mathbb{E}[u(W_T)],$$

where the vector $\mathbf{y_t}$ contains returns of risky assets and dividend yield: $\mathbf{y_t} = [r^e_{t,1}, \ldots, r^e_{t,N}, \delta_t]'$. The value function V_t is usually interpreted as the *best possible value* one can achieve from a state $(W_t, \mathbf{y_t})$ until the end of planning horizon T by always acting optimally. The associated Bellman equation is given by (see e.g. [25]):

$$V_t(W_t, \mathbf{y}_t) = \max_{\boldsymbol{x}_t} \mathbb{E}\left[\left(W_t\left(R_f + \boldsymbol{x}_t(e^{\mathbf{r}^e_{t+1}} - 1)\right)\right)^{1-\gamma} V_{t+1}(\mathbf{y}_{t+1})\right]. \tag{8.2}$$

Assuming that the investor liquidates all assets at the end of investment horizon T, the terminal conditions are $V_T(W_T, \mathbf{y}_T) = \dfrac{W_t^{1-\gamma}}{1-\gamma}, \forall \mathbf{y}_T$ and $\forall W_T$. The goal of the PCP is then to find $V_0(W_0, \mathbf{y}_0)$ and more importantly, the optimal allocation $\boldsymbol{x}_0$ that goes with it.

An important property of CRRA utility function is that optimal decisions are independent on wealth W_t. Thus we can always normalize the wealth W_t to 1 in the above Bellman equation without loss of generality.

Also, one characteristic of our market model presented earlier is that the asset's return, $r_{i,t}$, depend only on the one-period-lagged dividend yield (δ_{t-1}). Thus the sole state variable in equation (8.2) is in fact the dividend yield δ_t. The Bellman equation (8.2) thus amounts to the following *reduced version*:

$$\mathcal{V}_t(\delta_t) = \max_{\boldsymbol{x}_t} \mathbb{E}\left[\left(R_f + \boldsymbol{x}_t(e^{\mathbf{r}^e_{t+1}} - 1)\right)^{1-\gamma} \mathcal{V}_{t+1}(\delta_{t+1})\right], \tag{8.3}$$

with terminal condition $\mathcal{V}_T(\delta_T) = 1/(1-\gamma), \forall \delta_T$.

8.2 Gauss-Hermite solutions with Gaussian and Johnson-S_U noises

The expectation in this *reduced Bellman equation* (8.3) can be computed by Gauss-Hermite quadrature approximation using equations (7.7) in section 7.2 or equation (7.12) in section 7.3 respectively for the Gaussian or Johnson case. Precisely, define

$$g(\mathbf{y_{t+1}}, \boldsymbol{x}_t) := \left(R_f + \boldsymbol{x}_t(e^{\mathbf{r^e_{t+1}}} - 1)\right)^{1-\gamma} \mathcal{V}_{t+1}(\delta_{t+1}).$$

Then for the Gaussian case, equation (8.3) can be approximated by

$$\begin{aligned} V_t(\delta_t) &= \max_{\boldsymbol{x}_t} \mathbb{E}\left[g(\mathbf{y_{t+1}}, \boldsymbol{x}_t)\right], \\ &\approx \max_{\boldsymbol{x}_t} \frac{|\det(\boldsymbol{\Omega})|}{\sqrt{\det(\boldsymbol{\Sigma})\pi^n}} \sum_{k_1=1}^{Q} \cdots \sum_{k_n=1}^{Q} p_{k_1} \dots p_{k_n} g\left(\sqrt{2}\boldsymbol{\Omega}\mathbf{m}+\boldsymbol{\mu}\right), \\ &= \max_{\boldsymbol{x}_t} \frac{|\det(\boldsymbol{\Omega})|}{\sqrt{\det(\boldsymbol{\Sigma})\pi^n}} \sum_{k_1=1}^{Q} \cdots \sum_{k_n=1}^{Q} p_{k_1} \dots p_{k_n} \left(R_f + \boldsymbol{x}_t(e^{\mathbf{r^e_{t+1}}} - 1)\right)^{1-\gamma} \mathcal{V}_{t+1}(\delta_{t+1}), \end{aligned} \tag{8.4}$$

where $\boldsymbol{\Sigma} = \boldsymbol{\sigma_z}\mathbf{C_z}\boldsymbol{\sigma_z'}$, $\boldsymbol{\Sigma} = \boldsymbol{\Omega}\boldsymbol{\Omega}'$ with $\boldsymbol{\Omega}_i$ being the i^{th} line of $\boldsymbol{\Omega}$, and

$$r^e_{i,t+1} = \alpha_i + \beta_i\delta_t + \sigma_{u,i}\sqrt{2}\boldsymbol{\Omega_i}\mathbf{m},\ i = 1, \dots, N \tag{8.5}$$

$$\delta_{t+1} = \alpha_n + \beta_n\delta_t + \sigma_{u,n}\sqrt{2}\boldsymbol{\Omega_n}\mathbf{m}. \tag{8.6}$$

And for the Johnson case, let the function $\mathbf{h}(\cdot)$ being defined as in section 7.3

$$h_i(z_i) = \mu_i + \sigma_{u,i}\left(c_i + d_i\ \sinh\left(\frac{z_i - a_i}{b_i}\right)\right) \quad \text{and} \quad \mathbf{h}(\mathbf{z}) = [h_1(z_1)\ ,\ \dots\ ,\ h_n(z_n)]'\ ,$$

then an approximation of equation (8.3) is given by

$$\begin{aligned} V_t(\delta_t) &= \max_{\boldsymbol{x}_t} \mathbb{E}\left[g(\mathbf{y_{t+1}}, \boldsymbol{x}_t)\right], \\ &\approx \max_{\omega_t} \frac{|\det(\boldsymbol{\Lambda})|}{\sqrt{\det(\boldsymbol{C_z})\pi^n}} \sum_{k_1=1}^{Q} \cdots \sum_{k_n=1}^{Q} p_{k_1} \dots p_{k_n} g\left(\mathbf{h}\left(\sqrt{2}\boldsymbol{\Lambda}\mathbf{m}\right)\right), \\ &= \max_{\omega_t} \frac{|\det(\boldsymbol{\Lambda})|}{\sqrt{\det(\boldsymbol{C_z})\pi^n}} \sum_{k_1=1}^{Q} \cdots \sum_{k_n=1}^{Q} p_{k_1} \dots p_{k_n} \left(R_f + \boldsymbol{x}_t(e^{\mathbf{r^e_{t+1}}} - 1)\right)^{1-\gamma} \mathcal{V}_{t+1}(\delta_{t+1}), \end{aligned} \tag{8.7}$$

where $\boldsymbol{C_z} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}'$ with $\boldsymbol{\Lambda}_i$ being the i^{th} line of $\boldsymbol{\Lambda}$, and for $i = 1, \dots, N$:

$$r^e_{i,t+1} = h_i(\sqrt{2}\boldsymbol{\Lambda}_i\mathbf{m}) = \alpha_i + \beta_i\delta_t + \sigma_{u,i}\left(c_i + d_i \sinh\left(\frac{\sqrt{2}\boldsymbol{\Lambda}_i\mathbf{m} - a_i}{b_i}\right)\right), \tag{8.8}$$

$$\delta_{t+1} = h_n(\sqrt{2}\boldsymbol{\Lambda}_n\mathbf{m}) = \alpha_n + \beta_n\delta_t + \sigma_{u,n}\left(c_n + d_n \sinh\left(\frac{\sqrt{2}\boldsymbol{\Lambda}_n\mathbf{m} - a_n}{b_n}\right)\right). \tag{8.9}$$

In equations (8.4) to (8.9), $\mathbf{m} = [m_{k_1}\ , \dots\ ,\ m_{k_n}]'$ is the vector of standard Gauss-Hermite nodes and p_{k_i} is the associated weight as defined previously in sections 7.2 and 7.3.

8.3 Backward dynamic programming algorithm

Backward dynamic programming (DP) is used to solve equations (8.4) for Gaussian case, or (8.7) for Johnson case. The method is explained in details in an algorithm, described in Figure 8.1. We consider for simplicity the case with one risky asset, but the algorithm can easily be extended to cases with multiple risky assets. The general idea of this algorithm is, beginning from the final stage and going backward, to discretize the state variable δ_t into grids $\mathcal{D}_t$ at each time step, and compute the associated value function $\mathcal{V}_t(\delta_t)$ for each point on the grid using linear interpolation when it's necessary.

More precisely, steps 1 to 3 are preparation steps for the backward recursion. In step 1, we calculate the 2-dimensional GH nodes and their corresponding weights. These quantities are used repetitively in step 4 but need only to be computed once in advance. In step 2, the state variable (δ_t) is discretized at each time stage. Appendix F presents several discretization methods. In step 3, the backward recursion is initialized at the final stage to the terminal condition.

Step 4 is the core step of Algorithm 1, where the value function is computed for each point of state variable on the grid and at each time stage. Linear interpolation is used when necessary. Specifically, we follow the technique used in [25]: when an interpolation is required, firstly the value functions $\mathcal{V}_t$ are transformed to their certain equivalent (CE) functions CE_t which is defined by $CE_t(\delta_t) = u^{-1}(\mathcal{V}_t(\delta_t)) = \Big[(1-\gamma)\mathcal{V}_t(\delta_t)\Big]^{\frac{1}{1-\gamma}}$. Then linear interpolations are applied to $CE_t(\delta_t)$. Finally, we transform the interpolated value (in certainty equivalent) back to its corresponding value in value function. This transformation aims to increase the computational precision, since CE functions are much more linear than the value function itself in general.

Algorithm

```
Preparation steps:
  a) Compute the correlated nodes for Johnson distribution and their
     corresponding weights in 2-dimension:
      a)1. Choose $Q$, the number of nodes used in Gauss-Hermite quadrature.
           Determine the GH nodes $\mathbf{m}$ and their corresponding weights $\mathbf{p}$.
      a)2. Compute the Cartesian product of $\mathbf{m}$ and $\mathbf{p}$: $M = \mathbf{m} \times \mathbf{m}$ and $P = \mathbf{p} \times \mathbf{p}$.
      a)3. Compute $Prod_P$ by multiplying elements in each line of $P$.
  b) Discretize the state variable $\delta_t$ at each time step: $\mathcal{D}_t := \{\delta_t\}_{n=1}^{N_g}$, $\forall t =$
     $1, \ldots, T$, where $N_g$ is the number of discretization points.
  c) Initialize $V_T(\delta_T) = \frac{1}{1-\gamma}, \forall \delta_T \in \mathcal{D}_T$.
Solve the *reduced* Bellman equation for each time $t$ and discretized dividend
point:
  d) For $t = T-1, \ldots, 0$, do
      d)1. For all $\delta_t^{(n)} \in \mathcal{D}_t, n = 1, \ldots, N_g$, do
          d)1.1. Using $M$ and $Prod_p$ found in step 1, compute $r_{t+1}^{(n,k)}$ and $\delta_{t+1}^{(n,k)}$ by
                 equations (8.5) and (8.6) (Gaussian case), or by equations (8.8)
                 and (8.9) (Johnson case).
          d)1.2. Optimize over $\boldsymbol{x}_t$:
                 $$\mathcal{V}_t(\delta_t^{(n)}) = \max_{\boldsymbol{x}_t \in [0,1]} \Psi \sum_{k=1}^{Q^2} Prod_P(k) \left(R_f + \boldsymbol{x}_t(e^{r_{t+1}^{(n,k)}} - 1)\right)^{1-\gamma} \mathcal{V}_{t+1}(\delta_{t+1}^{(n,k)}),$$
                 where $\Psi$ is defined by
                 $$\Psi := \begin{cases} |\det(\boldsymbol{\Omega})|/\sqrt{\det(\boldsymbol{\Sigma})\pi^2}, & \text{for the Gaussian case} \\ |\det(\boldsymbol{\Lambda})|/\sqrt{\det(\boldsymbol{C_z})\pi^2}, & \text{for the Johnson case} \end{cases}$$
                 Note that the quantities $\mathcal{V}_{t+1}(\delta_{t+1}^{(n,k)})$ are obtained by linear
                 interpolation using the grid $\mathcal{D}_{t+1}$ and $\{\mathcal{V}_{t+1}(\delta_{t+1}^{(n)})\}_{j=1}^{N_g}$.
                 Let $\boldsymbol{x}_t^*(\delta_t^{(n)})$ denotes the optimal solution.
          d)1.3. If $n < N_g$, $n = n+1$, and go to step d)1.; else return $\{\mathcal{V}_t(\delta_t^{(n)})\}_{n=1}^{N_g}$
                 and $\{\boldsymbol{x}_t^*(\delta_t^{(n)})\}_{n=1}^{N_g}$.
      d)2. If $t > 0$, $t = t-1$, and go to step d); else return $\mathcal{V}_0(\delta_0)$ and $\boldsymbol{x}_0^*(\delta_0)$.
```

Figure 8.1: A backward DP algorithm for a multi-period OPCP with CRRA utility. Singly risky asset with log-normal returns which follow a restricted AR(1) dynamic.

Chapter 9

Optimal Portfolio Choices with Intermediate Consumption

9.1 Problem formulation

This section generalizes the multi-period OPCP with CRRA utility discussed previously to include inter-temporal consumption. Following the framework in [6] section 2.4.2, the discrete-time problem with time-separable CRRA utility of consumption is

$$V(t, W_t, \delta_t) = \max_{\{\boldsymbol{x}_\mathbf{s}, c_s\}_{s=t}^{T-1}} \mathbb{E}_t \left[\sum_{s=t}^{T} \frac{(c_s W_s)^{1-\gamma}}{1-\gamma} \right], \tag{9.1}$$

subject to the budget constraint:

$$W_{s+1} = (1 - c_s) W_s \left(R_f + \boldsymbol{x}_\mathbf{s} (e^{\mathbf{r}^\mathbf{e}_{\mathbf{s+1}}} - 1) \right),$$

where c_t is the proportion of total wealth used for consumption at time t, and other variables are defined as before. The above problem is also subject to the no-bankruptcy constraint $W_s \geq 0$, and the terminal condition $c_T = 1$. Due to the homogeneity of CRRA utilities with respect to wealth and following steps similar to the case without consumption, the Bellman equation in this case is given by:

$$\frac{1}{1-\gamma}\phi_t(\delta_t) = \max_{\boldsymbol{x}_t, c_t} \left[\frac{c_t^{1-\gamma}}{1-\gamma} + \mathbb{E}\left[\left((1-c_t)(R_f + \boldsymbol{x}_t(e^{r^e_{t+1}} - 1)) \right)^{1-\gamma} \frac{\phi_{t+1}(\delta_{t+1})}{1-\gamma} \right] \right]. \tag{9.2}$$

Also similar to the case without intermediate consumption, the value function at time t, $\phi_t(\delta_t)$, depends only on the sole state variable δ_t, and is linked to the consumption by

$$\phi_t(\delta_t) = \begin{cases} c_t^{-\gamma} & \text{if } \gamma > 0 \text{ and } \gamma \neq 1; \\ 1 & \text{if } \gamma = 1. \end{cases}$$

This explicit form of the value function implies that it can be obtained as a by-product of the optimization problem provided by the consumption policy. Furthermore, as highlighted by [6], with CRRA utility, the portfolio and consumption choices turn out to be sequential: "Because the value function is homothetic in wealth and the consumption choice c_t only scales the investable wealth $(1 - c_t)W_t$, the FOCs for the portfolio weights $\boldsymbol{x}_t$ are independent of c_t." Consequently, at each decision moment t, the optimal portfolio allocation is first determined ignoring consumption, and then the consumption choice is made given the optimal weights.

9.2 Algorithm with intermediate consumption

Figure 9.1 illustrates an algorithm for the PCP with inter-temporal consumption and one risky asset. This algorithm is almost identical to Algorithm 1 (steps 1 to 4.1.3 are the same as Algorithm 1), except here we have one extra step 4.1.4 where optimal consumption (in proportion) is calculated given the optimal weight in risky asset. Steps 4.1.3 and 4.1.4 reflect the sequential computation for optimal portfolio and consumption choices with CRRA utilities.

Algorithm

```
  Perform the steps of Algorithm 1 in Figure 8.1.
 4.1.4 Optimize over $c_t \in [0,1]$:
           $V_t(\delta_t^{(n)}) = \max_{c_t} \frac{c_t^{1-\gamma}}{1-\gamma} + (1-c_t) f(\boldsymbol{x}_t^*(\delta_t^{(n)})),$
     and let $c_t^*(\delta_t^{(n)})$ be the solution.
```

Figure 9.1: A backward DP algorithm for multi-period OPCPs with 1 risky asset and consumption, CRRA utility function.

Chapter 10

Numerical Experiments

10.1 Data Description

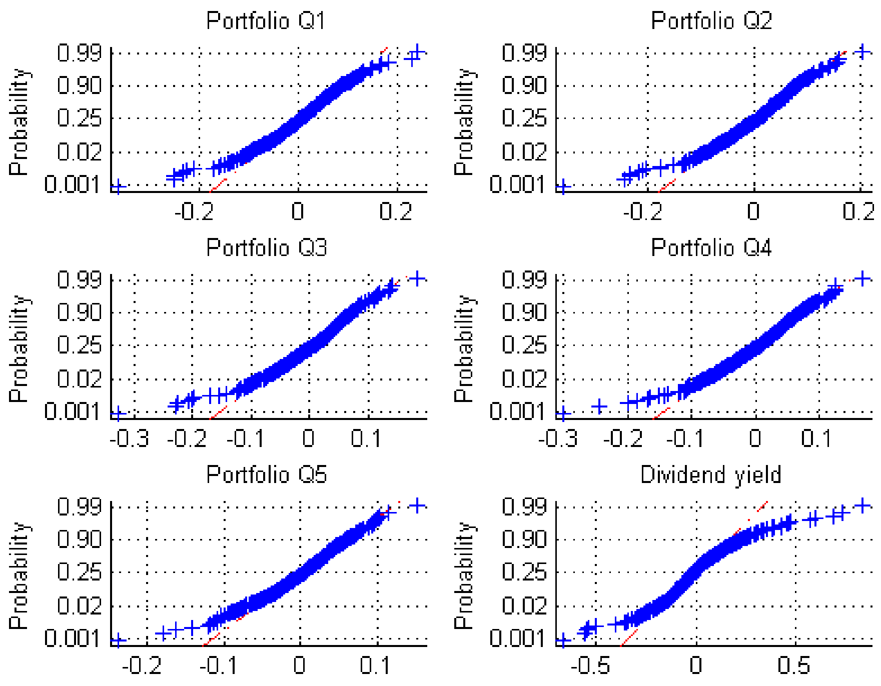
Monthly returns of size portfolios studied in this paper are obtained from French's website, where one can download data of "five quintiles size portfolios constructed using the June market equity and NYSE breakpoints at the end of each June"[1]. Dividend yields are constructed using data from WRDS database. Precisely, one first downloads directly from WRDS monthly returns including and excluding dividend of the CRSP index (AMEX/NASDAQ/NYSE combined index). We then compute the monthly dividend return by taking difference between returns including and excluding dividend payment. For each month in the sample period, multiplying the dividend return by the index's market value at the beginning of that month yields the dividend distribution in dollar for that month. Finally, dividend yields are computed as the sum of all dividend distribution (in dollar) over last 12 months divided by the index's market value of the current month. The study period spans from January 1960 to December 2013. We normalize dividend yields for this period by subtracting its average and dividing by its standard deviation.

[1]Description from French's website:
http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_port_form_sz.html

10.2 Parameter Estimation

Before estimating the parameters of the restricted VAR process, we regress the time series of each quintile portfolio's excess returns on (the time series of) the 1-period lagged dividend yield. We then test the Gaussian distribution assumption by plotting residuals obtained from previous regressions against theoretical probabilities from a normal distribution, as illustrated in Figure 10.1. If residuals are normally distributed, the plots will be linear (a straight line with slope of 45 degree as indicated by the dashed line), other distribution types will introduce curvature in the plot. Strong non linearities are observed on all plots, especially for the right tail side of the dividend yield, and for size portfolios on the left tail side.

Figure 10.1: Normal probability plots of residuals obtained from regressions of size portfolios' excess returns and of log-dividend yield on the one-period lagged log-dividend yield.



The parameters are estimated by Maximum Log-Likelihood (MLL) procedures which are described in Appendix E. Precisely, in our market model the joint dynamic of asset returns and log-dividend yield is given by a restricted VAR process of type:

$$\mathbf{y_{t+1}} = \boldsymbol{\alpha} + \boldsymbol{\beta}\delta_t + \boldsymbol{\sigma}\boldsymbol{\nu}_{t+1}. \tag{10.1}$$

The estimated parameters for all five quintile portfolios and dividend yield, by assuming the noise term $\boldsymbol{\nu}_t$ are either Johnson-$S_U$or Gaussian distributed, are shown respectively in Tables 10.2 and 10.1. Standard errors of estimated parameters are reported in parenthesis. A superscript $*$, $**$ or $***$ indicates respectively that the corresponding estimated parameter is significant at confidence level of 90%, 95% or 99%.

Tables 10.2 and 10.1 also report the results of significance tests for α_i and β_i. These test results suggest strong serial correlation in the dividend yield series in both Gaussian and Johnson cases, while the relationships between portfolios' returns and the lagged dividend yield are only significant at lower confidence levels.

Table 10.1: ML estimation for the restricted VAR(1) parameters with Gaussian noises.

	PF Q1	PF Q2	PF Q3	PF Q4	PF Q5	Dividend
LLH	867.5536	908.1319	964.7856	1005.0001	1118.0713	315.2865
α_i	0.0054**	0.0057**	0.0057**	0.0054**	0.0038**	-0.0020
	(0.0026)	(0.0025)	(0.0023)	(0.0022)	(0.0018)	(0.0079)
β_i	0.0027	0.0040*	0.0037*	0.0029	0.0021	0.9893***
	(0.0023)	(0.0022)	(0.0020)	(0.0018)	(0.0015)	(0.0059)
$\sigma_{z,i}$	0.0633***	0.0595***	0.0545***	0.0512***	0.0430***	0.1486***
	(0.0012)	(0.0011)	(0.0010)	(0.0010)	(0.0009)	(0.0015)
			Cz			
	1					
	0.9649	1				
	0.9282	0.9806	1			
	0.8860	0.9489	0.9775	1		
	0.7503	0.8276	0.8763	0.9216	1	
	-0.7576	-0.8150	-0.8448	-0.8723	-0.8896	1

Table 10.2: ML estimation for the restricted VAR(1) parameters with Johnson-S_U noises.

	PF Q1	PF Q2	PF Q3	PF Q4	PF Q5	Dividend
LLH	900.2177	940.2779	996.9832	1033.4106	1141.6324	388.0529
α_i	0.0055**	0.0057**	0.0057***	0.0054***	0.0038**	-0.0005
	(0.0025)	(0.0023)	(0.0021)	(0.0020)	(0.0017)	(0.0067)
β_i	0.0029	0.0038**	0.0035**	0.0029*	0.0016	0.9758***
	(0.0021)	(0.0019)	(0.0018)	(0.0017)	(0.0014)	(0.0046)
$\sigma_{u,i}$	0.0632***	0.0592***	0.0542***	0.0509***	0.0431***	0.1559***
	(0.0028)	(0.0023)	(0.0021)	(0.0019)	(0.0018)	(0.0116)
a_i	0.3632*	0.7356***	0.8737***	0.7365***	0.5933***	-0.4253***
	(0.1467)	(0.2207)	(0.2546)	(0.2574)	(0.2188)	(0.0992)
b_i	1.6053***	1.8838***	1.9647***	1.9619***	1.8636***	1.1715***
	(0.1876)	(0.2475)	(0.2669)	(0.2826)	(0.2767)	(0.1046)
c_i	0.3511	0.6892	0.8067	0.6941	0.5663	-0.3822
d_i	1.2670	1.4946	1.5424	1.5861	1.5144	0.7156
	Correlation matrix among Johnson innovations (C_u)					
	1					
	0.9649	1				
	0.9282	0.9806	1			
	0.8860	0.9489	0.9775	1		
	0.7502	0.8275	0.8763	0.9215	1	
	-0.7547	-0.8114	-0.8410	-0.8687	-0.8848	1
	Correlation matrix among Gaussian innovations (C_z)					
	1					
	0.9628	1				
	0.9234	0.9794	1			
	0.8795	0.9465	0.9767	1		
	0.7388	0.8199	0.8714	0.9188	1	
	-0.7444	-0.8077	-0.8413	-0.8721	-0.8913	1

10.3 Optimal weights with predictable returns

10.3.1 One risky asset

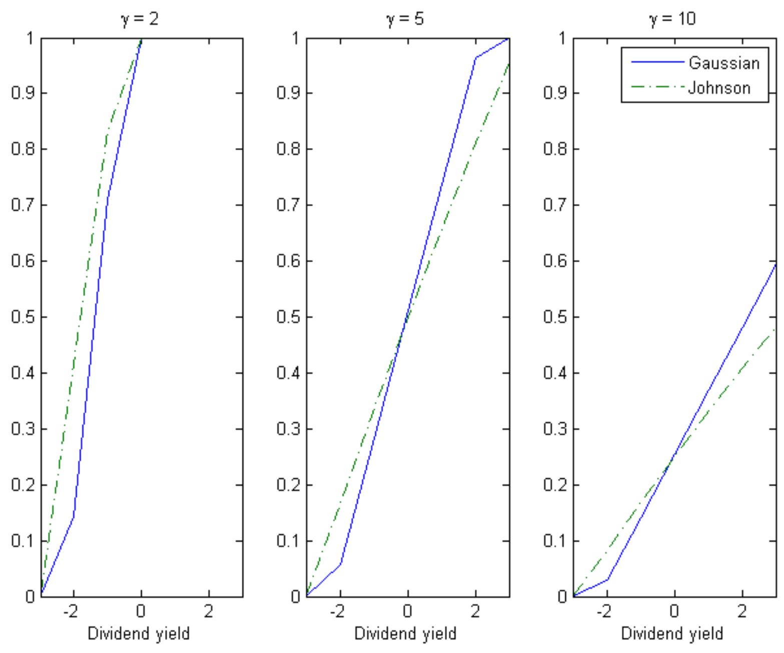
We first consider the case of one single risky asset with predictable returns by log-dividend yield. The joint dynamic of asset return and log-dividend is described by the restricted VAR discussed previously in section 7. We use estimated parameters from Table 10.2 where columns 1 to 5 represent respectively the first to the fifth quintile portfolio, and the last column is for log-dividend yield. For example, if the only risky asset in our portfolio is the 1st quintile size portfolio, we take parameters in the first and last columns of Table 10.2. We use our recursive algorithm described in Figure 8.1 to compute optimal weights, under both Gaussian and Johnson assumptions, for different values of risk aversion γ, investment horizon T and initial state (log-dividend yield) value δ_0. Ten (10) nodes are used for Gauss Hermite quadrature in each dimension.

Figure 10.2 shows, for the single-period case (one month, $T = 1$), optimal weights at $t = 0$ in the risky asset. We only present results with the fifth quintile portfolio in this paper because results for other quintile portfolios are quite similar. It can be observed that for the single-period case, Johnson and Gaussian assumptions predict significantly different optimal weights, especially for highly positive values of initial dividend yield and higher values for the risk aversion coefficient γ: as the initial dividend yield increases, the difference becomes more pronounced, and this difference is bigger for larger values of γ.

Figure 10.3 illustrates how optimal allocation in risky asset at the initial period varies as the investment horizon becomes longer. Three (3) different values for initial dividend are considered $\delta_0 \in \{-2, 0, 2\}$. Since the dividend yields are normalized, these values correspond respectively to -2 times its standard deviation, the mean and +2 times the standard deviation. We present results with the fifth quintile portfolio as in the case for single-period. Results for other portfolios are close to those of the fifth quintile portfolio.

Some interesting remarks can be drawn from Figure 10.3. Firstly, under both Gaussian and Johnson assumptions, the optimal allocation to risky asset increases as planning horizon becomes longer, but the increase is less important and more stable for the Johnson case compared to the Gaussian case. Secondly, differences in optimal allocations between Gaussian and Johnson policies are noticeable for all levels of initial dividend

Figure 10.2: Optimal allocation in risky asset as a function of initial dividend yield for different values of γ with one period and one risky asset, $R_f = 1.0028$.



yield δ_0 considered, and the differences are larger for higher levels of δ_0. Table 10.3, which lists optimal allocations in risky asset at $t = 0$ for both policies and for different initial dividend yield, helps to better visualize previous remarks.

Figure 10.3: Optimal fraction in risky asset at $t = 0$ for different length of investment horizon T (in year) for Gaussian and Johnson-S_U assumptions. One risky asset (the fifth quintile portfolio), $\gamma = 10, R_f = 1.0028$, initial state $\delta_0 = -2$ (left), $\delta_0 = 0$ (midddle) and $\delta_0 = 2$ (right).

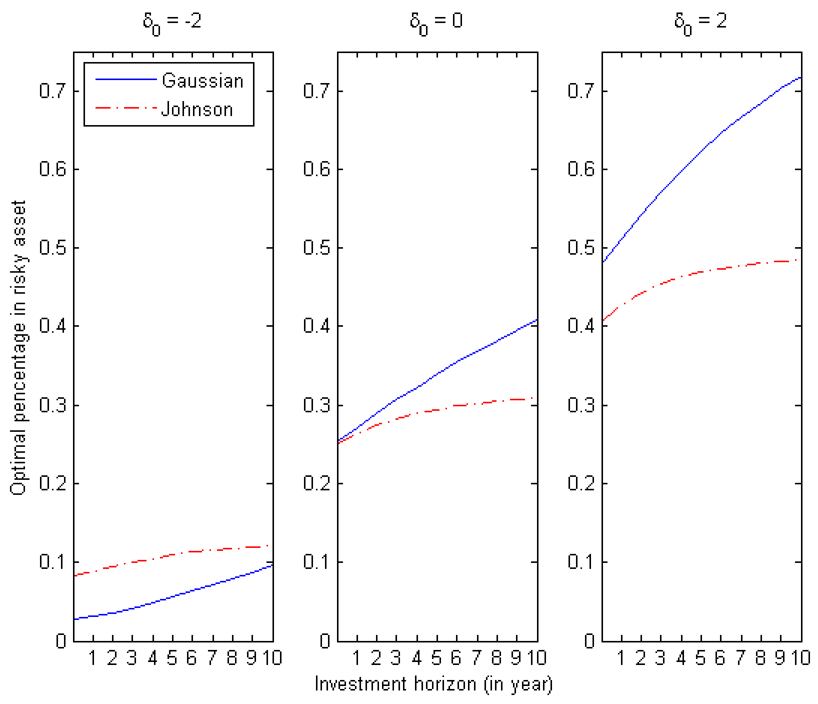


Table 10.3: Optimal allocations in risky asset at $t = 0$ for different inverstment horizons. One risky asset, $R_f = 1.0028$.

δ_0	$T = 12$						$T = 120$					
	$\gamma = 2$		$\gamma = 5$		$\gamma = 10$		$\gamma = 2$		$\gamma = 5$		$\gamma = 10$	
	John	Gaus	John	Gaus	John	Gaus	John	Gaus	John	Gaus	John	Gaus
-2.0	0.4291	0.1500	0.1757	0.0619	0.0886	0.0313	0.4959	0.2674	0.2338	0.1727	0.1228	0.0985
-1.5	0.6447	0.4423	0.2641	0.1810	0.1331	0.0912	0.7160	0.5834	0.3277	0.3209	0.1705	0.1765
-1.0	0.8552	0.7348	0.3512	0.3005	0.1770	0.1513	0.9269	0.8936	0.4199	0.4697	0.2176	0.2549
-0.5	1.0000	1.0000	0.4370	0.4200	0.2204	0.2114	1.0000	1.0000	0.5106	0.6182	0.2639	0.3338
0.0	1.0000	1.0000	0.5214	0.5395	0.2632	0.2715	1.0000	1.0000	0.6002	0.7651	0.3099	0.4131
0.5	1.0000	1.0000	0.6044	0.6590	0.3054	0.3317	1.0000	1.0000	0.6883	0.9081	0.3552	0.4922
1.0	1.0000	1.0000	0.6858	0.7785	0.3469	0.3918	1.0000	1.0000	0.7747	1.0000	0.3999	0.5715
1.5	1.0000	1.0000	0.7657	0.8978	0.3878	0.4520	1.0000	1.0000	0.8590	1.0000	0.4438	0.6505
2.0	1.0000	1.0000	0.8440	1.0000	0.4280	0.5122	1.0000	1.0000	0.9420	1.0000	0.4873	0.7296

It is important to highlight that there are two sources for the differences between Gaussian and Johnson optimal allocations. The first and more intuitive source is the difference in the assumption of underlying distribution. The other one comes from differences in the ML estimates. By comparing Tables 10.2 and 10.1, the "common" parameters (α_i, β_i and σ_i) are slightly different when changing from one distribution assumption to another. Thus the difference in portfolio allocation may be not only due to different distribution assumption, but also to differences in the estimation of "common" parameters.

To isolate the impact of the distribution assumption, we next perform a controlled study. The controlled Gaussian case is similar to the Gaussian case except that rather than using the ML estimations assuming a Gaussian distribution as given by Table 10.1, we use the estimated parameters assuming a Johnson-$S_U$distribution (only α_i, β_i and σ_i) as given by Table 10.2. In other words, in the *controlled Gaussian case* we vary only the distribution assumption while keeping the parameter estimations constant. Table 10.4 compares optimal allocations in the risky asset for Johnson, Gaussian assumption and the controlled case, for various investment horizons and risk aversion levels when initial dividend yield $\delta_0 = 0$. The risky asset chosen to produce results in Table 10.4 is again the fifth quintile portfolio to keep consistency with our results presented previously.

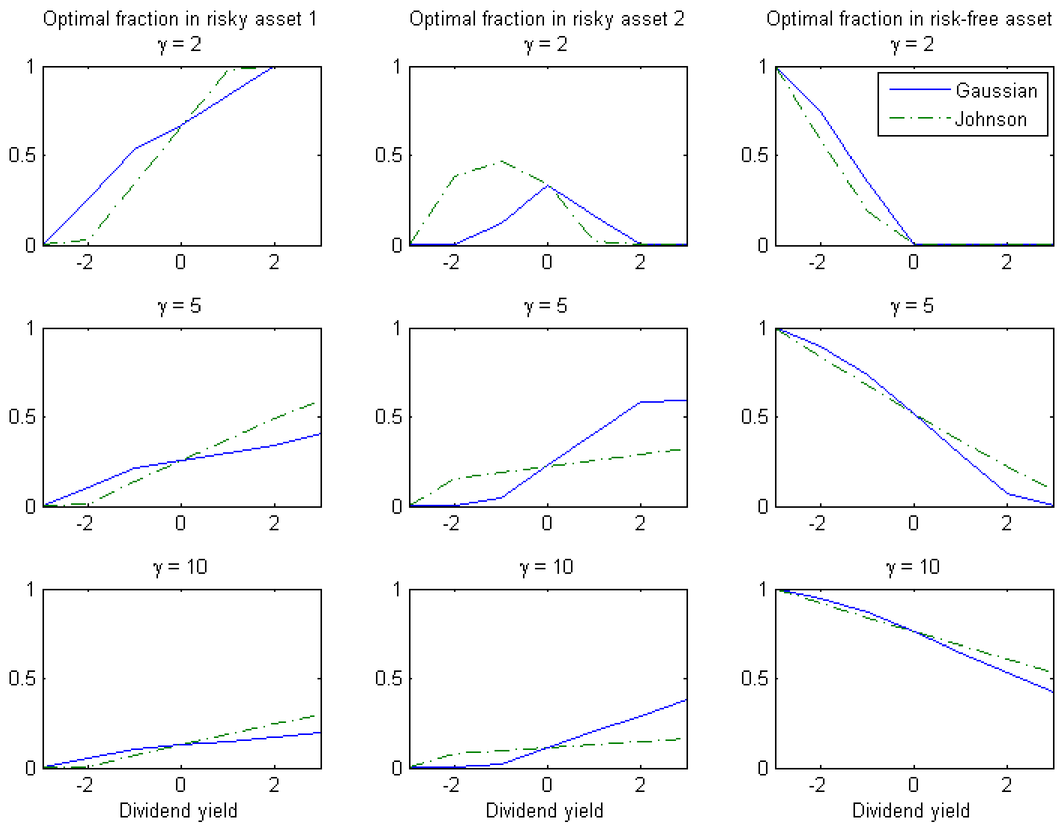
Table 10.4: Controlled experiment for impacts of distributional assumption for different T and γ with 1 risky asset including dividend yield. $R_f = 1.0028$, $\delta_0 = 0$

	$\gamma = 2$			$\gamma = 5$			$\gamma = 10$		
T	Johnson	Gaussian	Gaussian	Johnson	Gaussian	Gaussian	Johnson	Gaussian	Gaussian
(Year)		(controlled)			(controlled)			(controlled)	
1/12	1.0000	1.0000	1.0000	0.5001	0.5082	0.5107	0.2510	0.2540	0.2553
1	1.0000	1.0000	1.0000	0.5212	0.5294	0.5394	0.2630	0.2660	0.2715
2	1.0000	1.0000	1.0000	0.5396	0.5482	0.5700	0.2737	0.2767	0.2890
3	1.0000	1.0000	1.0000	0.5543	0.5633	0.5994	0.2823	0.2854	0.3061
4	1.0000	1.0000	1.0000	0.5658	0.5754	0.6275	0.2891	0.2924	0.3227
5	1.0000	1.0000	1.0000	0.5749	0.5848	0.6541	0.2945	0.2980	0.3387
6	1.0000	1.0000	1.0000	0.5820	0.5923	0.6793	0.2988	0.3024	0.3540
7	1.0000	1.0000	1.0000	0.5876	0.5982	0.7028	0.3022	0.3059	0.3685
8	1.0000	1.0000	1.0000	0.5920	0.6029	0.7245	0.3048	0.3087	0.3824
9	1.0000	1.0000	1.0000	0.5954	0.6065	0.7446	0.3070	0.3109	0.3955
10	1.0000	1.0000	1.0000	0.5981	0.6093	0.7629	0.3086	0.3127	0.4078

10.3.2 Two risky assets with predictable returns

We now examine the case with two or more risky assets. We only report in this section results with the first and fifth quintile size portfolios as risky assets. Figure 10.4 illustrates how optimal weights in risky assets (the 1st and 2nd columns) and in risk-free asset (the 3rd column) change with initial dividend yield for a single-period ($T = 1$) investor. Three levels of risk aversion are considered.

Figure 10.4: Single-period optimal fraction in risky assets and in risk-free asset for various initial dividend yield when error term is Gaussian and Johnson-$S_U$distributed. Results with three levels of risk aversion are illustrated: $\gamma = 2, 5, 10$, $Rf = 1.0028$.



At least two remarks should be made from Figure 10.4. First of all, by comparing across three levels of risk aversion (comparison by line), the proportion in risky assets decreases for higher risk aversion levels, which is coherent with the design of CRRA utility.

Table 10.5: Comparison of Johnson and Gaussian policies for various risk aversion levels and initial dividend yield. Single period case

	Low $\gamma = 2$		Medium $\gamma = 5$		High $\gamma = 10$	
	Johnson	Gaussian	Johnson	Gaussian	Johnson	Gaussian
$\delta_0 < 0$	Q5	Q1	Q5	Q1	Q5	Q1
$\delta_0 = 0$	Q1	Q1	$\approx$	$\approx$	$\approx$	$\approx$
$\delta_0 > 0$	Q1	Q1	Q1	Q5	Q1	Q5

Second, by comparing across risky assets (sub-figures in the 1st (for portfolio Q1) and 2nd columns (for portfolio Q5)), we notice the Gaussian and Johnson policies allocate wealth among risky assets in a different way. Table 10.5 gives a clearer idea of asset preference (larger allocation) of Johnson and Gaussian policies. At least three factors influence allocation to risky assets: initial dividend yield, risk aversion and asset specific characteristics (mean, variance, skewness and kurtosis). In our example of this section, portfolio Q1 has a higher average, higher variance, slightly higher Sharpe Ratio, higher kurtosis and more skewed to the left than portfolio Q5 (see Table 10.6).

Table 10.6: Descriptive statistics of portfolio returns and dividend yields.

	PF Q1	PF Q2	PF Q3	PF Q4	PF Q5	Dividend
mean	0.0054	0.0057	0.0056	0.0053	0.0037	-0.0000
s.d	0.0634	0.0596	0.0546	0.0513	0.0431	1.0000
skewness	-0.6371	-0.8324	-0.8567	-0.7892	-0.6190	0.2971
kurtosis	6.0720	6.1601	6.1231	5.9893	5.1544	2.4515
S.R.	0.0852	0.0951	0.1025	0.1029	0.0848	-0.0000

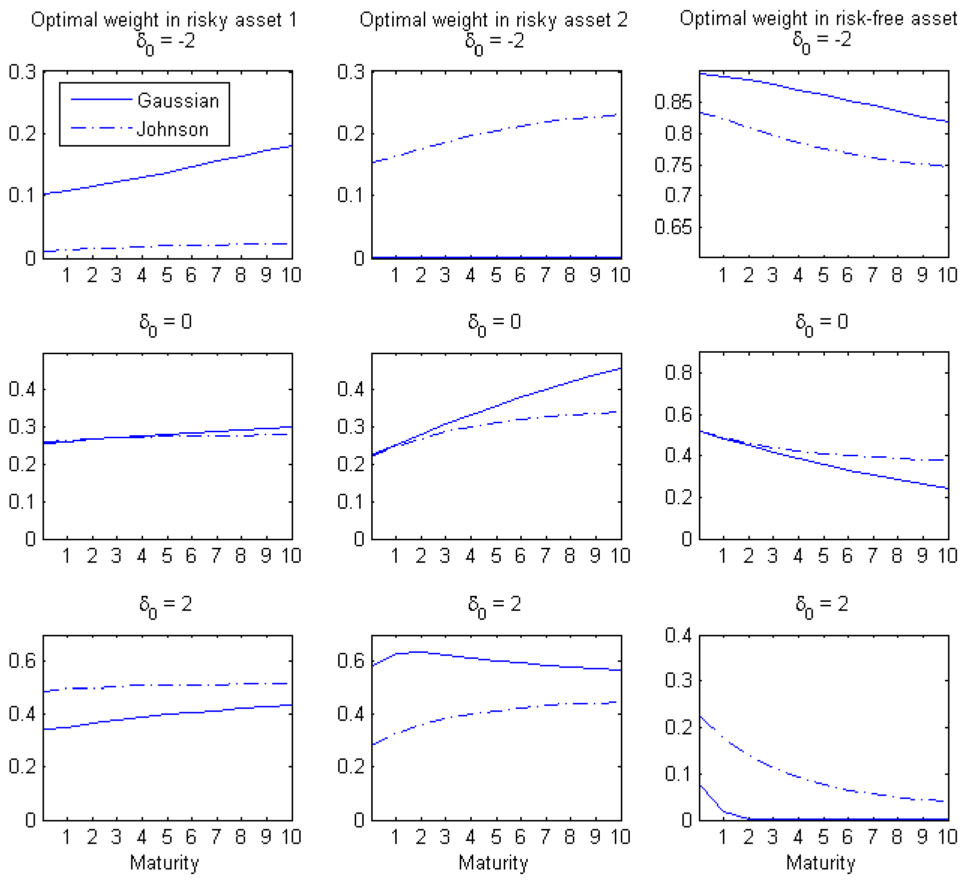
When δ_0 is low, the Gaussian policy invests more in portfolio Q1 because it has the higher Sharp Ratio, while the Johnson policy does the inverse and prefers portfolio Q5. One possible explanation is that when δ_0 is strongly negative and according to our market model (the restricted VAR(1)), the possibility to have a strongly negative return during the following period is higher for assets more skewed to the left (portfolio Q1). Thus the Johnson policy judges the situation as too risky and allocates more wealth to the "safer" asset (portfolio Q5), even when the investor is willing to take more risk (small values of γ).

When δ_0 equals to zero, both policies favor portfolio Q1 for an investor willing to take more risk, and assign similar weights to both assets for an investors more averse to risk.

When the initial dividend yield (δ_0) is high, the possibility to have a strongly positive return during the following period is higher for assets with higher kurtosis, thus the Johnson policy puts more wealth in portfolio Q1 regardless of the risk aversion level. As for the Gaussian policy, it prefers portfolio Q5 due to a lower variance, except when the risk aversion level is low.

We next examine the multi-period case with two risky assets. Figure 10.5 shows optimal weights in risky and risk-free assets. The risky assets used to produce this figure are again the 1^{st} and 5^{th} quintile portfolios. Three levels of initial dividend yield are considered, $\delta_0 \in \{-2, 0, 2\}$ which corresponds respectively to -2 times of its standard deviation, the mean and 2 times of its standard deviation. According to this figure, the preference between risky assets is in accordance with our previous analysis for the single-period case. Another remark is that both Gaussian and Johnson policies allocate less wealth to the risk-free asset as investment horizon becomes longer, a fact consistent with the existing literature. Yet, the risk-free allocation decreases faster for the Gaussian policy compared to its Johnson twin.

Figure 10.5: Multi-period optimal fraction in risky assets and in risk-free asset for various investment horizon T (in month) with Gaussian and Johnson assumptions. Medium risk aversion $\gamma = 5$ and $Rf = 1.0028$.



10.3.3 Optimal weights with inter-temporal consumption

We next study the case with one risky asset and adding inter-temporal consumption. The approach illustrated in this subsection can be easily extended to cases with multiple risky assets. Figure 10.6 illustrates the optimal fraction of total wealth in consumption (left) and in risky asset (right) at $t = 0$ as a function of initial dividend yield, for both Gaussian and Johnson error terms with an investment horizon of 10 years (120 months $T = 120$). We consider three different relative risk aversion levels: low $\gamma = 2$ (solid line), median $\gamma = 5$ (dashed line) and high $\gamma = 10$ (dotted line). Parameter values are taken from Table 10.2 with $R_f = 1.0028$. We take the five quintile size portfolios, one at a time, as the only risky asset. Results are similar for all five quintile portfolios and we only report those with the fifth quintile portfolio to stay in line with our previous results. For a clearer illustration of these results, Table 10.7 lists the numbers used to plot Figure 10.6. Comparing optimal allocations in Table 10.7 to those in Table 10.3, we find that when consumption is taken into account, both Johnson and Gaussian policies put less wealth in risky asset.

Several remarks should be made about Figure 10.6. First, for an investor with fixed risk aversion, the proportions allocated to consumption and to the risky asset increase with initial dividend yield. Second, for different investors with different risk aversions, the ones willing to take more risk put more wealth into risky asset, while the proportion of consumption remains almost the same. Thirdly, the Johnson policy seems to be more bullish when initial dividend yield level is low, thus puts more allocations to consumption and risky asset than Gaussian policy. In contrary, when initial dividend level rises, Johnson policy becomes more bearish than Gaussian policy.

Figure 10.6: Optimal fraction in consumption and in risky asset at $t = 0$ for different initial dividend yield when error term is Gaussian- and Johnson-S_U-distributed with 10 years of investment horizon, and three risk aversion level $\gamma = 2, 5, 10.$, $R_f = 1.0028$

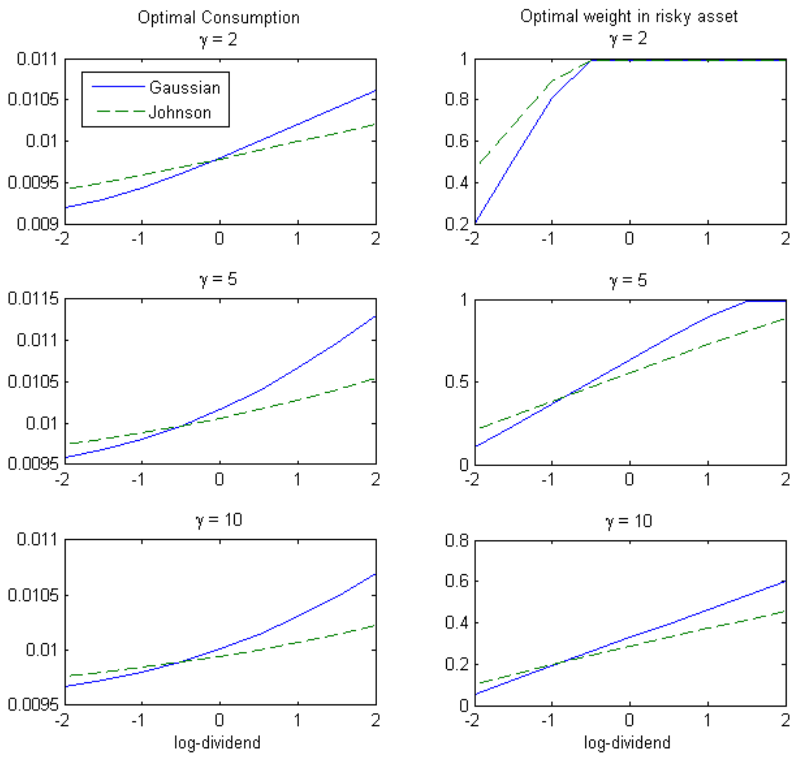


Table 10.7: Optimal allocations to consumption and risky asset for the 5th quintile portfolio with $T = 120$ (10 years), $R_f = 1.0028$.

	$\gamma = 2$				$\gamma = 5$				$\gamma = 10$			
	Johnson		Gaussian		Johnson		Gaussian		Johnson		Gaussian	
δ_0	ω_0	c_0	ω_0	c_0	ω_0	c_0	ω_0	c_0	ω_0	c_0	ω_0	c_0
-2.0	0.4581	0.0094	0.1985	0.0092	0.2013	0.0097	0.1019	0.0096	0.1038	0.0098	0.0550	0.0097
-1.5	0.6719	0.0095	0.4999	0.0093	0.2911	0.0098	0.2330	0.0097	0.1493	0.0098	0.1222	0.0097
-1.0	0.8797	0.0096	0.7981	0.0094	0.3796	0.0099	0.3646	0.0098	0.1943	0.0098	0.1899	0.0098
-0.5	0.9807	0.0097	0.9808	0.0096	0.4667	0.0100	0.4961	0.0100	0.2386	0.0099	0.2575	0.0099
0.0	0.9805	0.0098	0.9805	0.0098	0.5523	0.0101	0.6270	0.0102	0.2824	0.0099	0.3250	0.0100
0.5	0.9803	0.0099	0.9801	0.0100	0.6364	0.0102	0.7566	0.0104	0.3256	0.0100	0.3923	0.0101
1.0	0.9801	0.0100	0.9797	0.0102	0.7189	0.0103	0.8837	0.0107	0.3680	0.0101	0.4593	0.0103
1.5	0.9799	0.0101	0.9793	0.0104	0.7998	0.0104	0.9782	0.0110	0.4098	0.0101	0.5260	0.0105
2.0	0.9797	0.0102	0.9789	0.0106	0.8789	0.0105	0.9775	0.0113	0.4509	0.0102	0.5924	0.0107

10.4 Policy Performance Analysis

From the previous section, it is easy to see that Johnson and Gaussian assumption obtain different portfolio weights, especially when the number of risky assets exceeds one and when the investment horizon becomes longer. Then the question is which assumption/policy results in better performance? This section is devoted to answering this question by evaluating and comparing Gaussian and Johnson-S_u policies according to two axis, which are presented in two sub-sections. The first axis (subsection 10.4.1) consists of simulation studies based on arbitrarily chosen but realistic, parameter values, and the second one (in subsection 10.4.2) is composed of out-of-sample tests using empirical data collected from the market.

10.4.1 Robustness with respect to a wrong distributional assumption

In this subsection, we perform scenario analysis to examine the impact of an incorrect distribution assumption on portfolio performance. We first create two scenarios where the "real" distribution is respectively Johnson-S_u and Gaussian, and in each scenario compute the loss in annualized certainty equivalent rate by applying optimal policies resulted from the native distribution (the same as the distribution in the scenario) and the foreign distribution (different from what is assumed in that scenario).

To be precise, we first compute our optimal policies by assuming the underlying return distribution is respectively Gaussian or Johnson-S_u, using fictive but realistic parameters for both Gaussian and Johnson-S_u distributions[2] (see Appendix G for details about values of the fictive parameters used in the simulation study). Denote these policies respectively by π_G and π_J.

In a second step, we assume the "real" environment is generated by a Gaussian or Johnson-S_u distribution and simulate returns of the risky asset and dividend yield according to the restricted VAR process with the "real" distribution. Next we apply π_G and π_J to these simulated paths. Thus four different scenarios are created, namely the Gaussian policy applied to a Gaussian environment (π_G^G), the Gaussian policy applied to a Johnson environment (π_G^J), the Johnson policy applied to a Johnson environment (π_J^J) and the Johnson policy applied to a Gaussian environment (π_J^G).

[2]The value of parameters used in our simulation study are closed to those calibrated according to real data.

Finally, we compute and compare the annualized certainty equivalent rate, which can be interpreted as the certain return that an investor would accept rather than taking an uncertain but potentially higher return, of the average final utility over all paths obtained in all scenarios. For example, for scenario π_G^G (Gaussian policy applied to a Gaussian environment), the average final utility is given by

$$\overline{u}_T = \frac{1}{N_{sim}} \sum_{n=1}^{N_{sim}} \frac{\left(W_T^{(n)}\right)^{1-\gamma}}{1-\gamma},$$

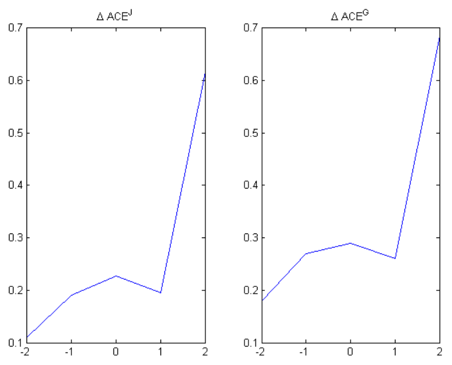
where T is the planning horizon (in months), $W_T^{(n)} = \Pi_{t=0}^{T-1}(R_f + \omega_t^{(n)}(e^{r_{t+1}^{(n)}} - 1))$, and $\omega_t^{(n)}(t = 0, \dots, T-1)$ is the optimal weight given by policy π_G. The annualized certainty equivalent rate for this scenario, ACE_G^G, is obtained by

$$\overline{u}_T = \frac{(1 + ACE_G^G)^{\frac{T(1-\gamma)}{12}}}{1-\gamma} \qquad \longleftrightarrow \qquad ACE_G^G = ((1-\gamma)\overline{u}_T)^{\frac{12}{T(1-\gamma)}} .$$

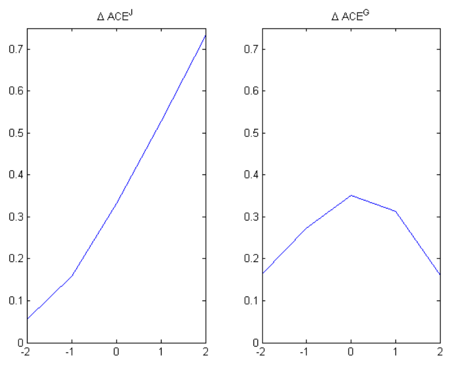
For presentation clarity, we name a policy the "native policy" if it is computed with the same distribution assumption as in the simulation, otherwise it is named as the "foreign policy". We compare the ACE across the native policy and the foreign policy within each environment by taking their difference. Precisely, we obtain the difference between ACE when the real environment is Gaussian $\Delta ACE^G = ACE_G^G - ACE_J^G$, and when the real environment is Johnson $\Delta ACE^J = ACE_J^J - ACE_G^J$. Recall that the subscript stands for the assumption based on which the policy is computed, and the superscript stands for the real environment according to which paths are simulated. It is expected that both ΔACE^G and ΔACE^J to be positive since in each case, the native policy should perform better than the foreign policy.

Figure 10.7 illustrates ΔACE^G and ΔACE^J in basis point for three different degrees of risk aversion and various initial dividend yield δ_0 with a planning horizon $T = 120$ (10 years). We observe that the native policy always dominates the foreign policy in term of higher annualized certainty equivalent, and this observation is valid for other lengths of investment horizon, initial dividend yield and risk aversion levels.

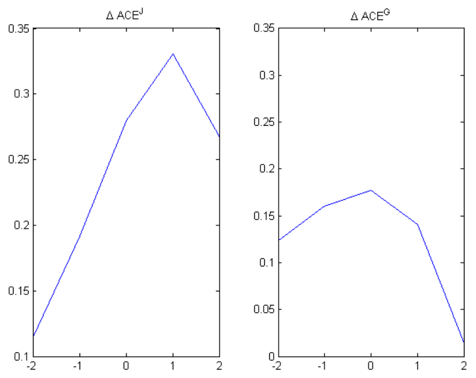
Figure 10.7: Differences of annualized certainty equivalent rate (in bps) by scenario analysis for three different degrees of risk aversion and various initial dividend yield δ_0 with planning horizon $T = 120$, $R_f = 1.0028$, 100000 simulation paths.



(a) $\gamma = 2$



(b) $\gamma = 5$



(c) $\gamma = 10$

10.4.2 Out of Sample Performance

We test the out-of-sample (OOS) performance of policies produced with Gaussian and Johnson assumptions. The last 120 months of our sample period (2003/12 – 2013/12) are used for OOS performance tests. Three types of tests are conducted, all assuming an initial wealth $W_0 = 1$. Note the OOS period includes the recent financial crisis 2008 – 2009.

In the first test, we fix the investment horizon $T = 6$ (one half year) and divide the OOS period (120 months) into 20 blocks ($120/6 = 20$). At the beginning of each block, we compute the optimal policies for Gaussian and Johnson assumptions with parameters estimated with all available data up to that moment. These policies are implemented until the end of the block. Using empirical data and by adopting the Johnson or Gaussian policies, we compute the "realized" final wealth for each block, which are illustrated in Figure 10.8 together with the mean final wealth of all blocks.

The second test is very similar to the first one, except that we re-optimize at each month. That-is-to-say, for each block of T months, we compute the optimal policies at $t = 0$, and hold it for one month. Then we re-compute the optimal policies at $t = 1$ with a investment horizon of $T - 1$, and hold it for another month. The procedures are repeated until the end of investment horizon. Figure 10.9 compares the realized final wealth at the end of each block together with the average final wealth over all blocks for Johnson and Gaussian policies of this test.

In the third test, we proceed as in the second one, except fixing the investment horizon always to $T = 6$ and re-optimize at the beginning of each month. Thus at each (re-)optimization, only the policy for the first period is implemented, and re-balancing is made at each period until the end of our out-of-sample period (120 months). Figure 10.10 compares the wealth at each period calculated using empirical returns and by implementing Johnson and Gaussian policies, together with the average wealth over the whole OOS period.

From the three tests and figures, it can be observed that the Johnson policy outperforms the Gaussian policy during market disturbance, while the inverse is true during market recovery and expansion time. This observation may suggest the evidence of a regime switching dynamic in the market.

Figure 10.8: Out of sample test (I) performance for each of five quintile portfolios with Gaussian and Johnson assumptions. $W_0 = 1$

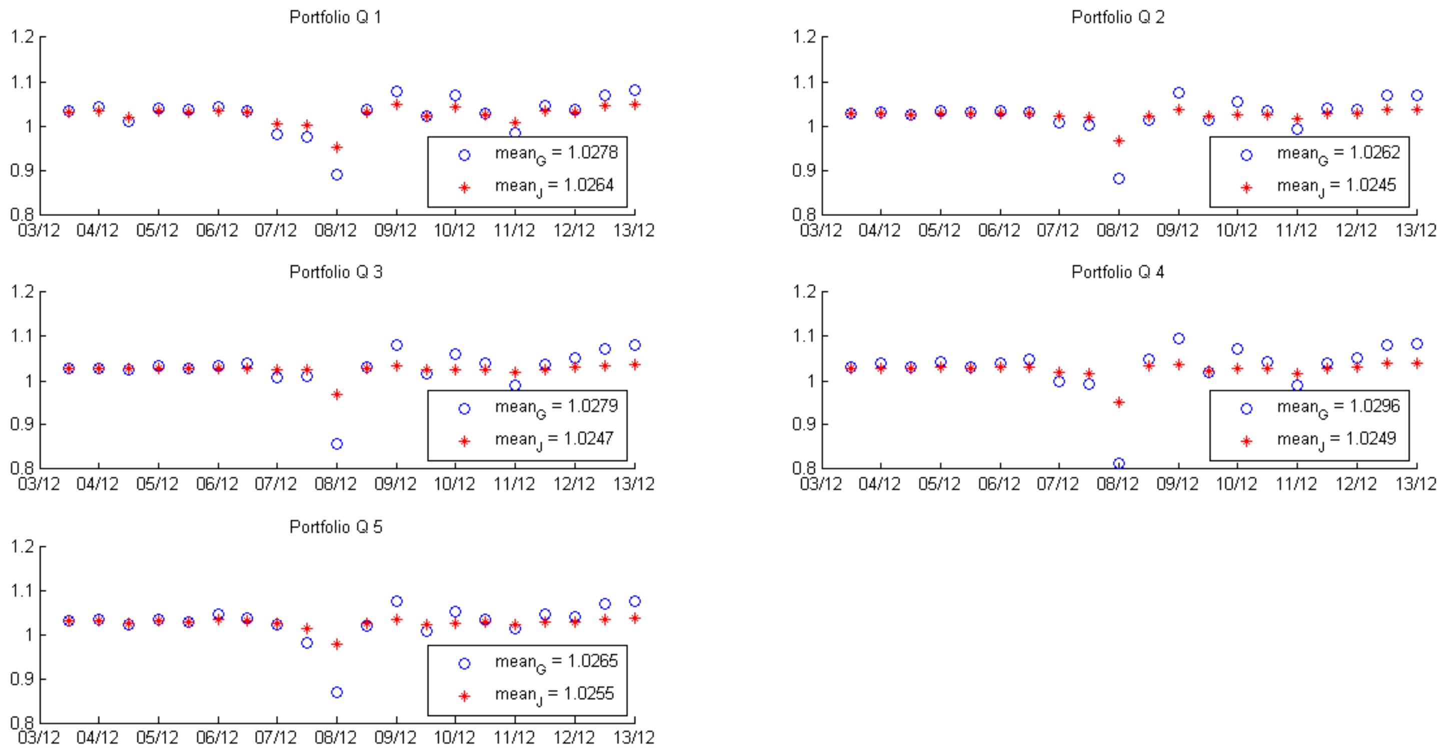


Figure 10.9: Out of sample test (II) performance for each of five quintile portfolios with Gaussian and Johnson assumptions. $W_0 = 1$

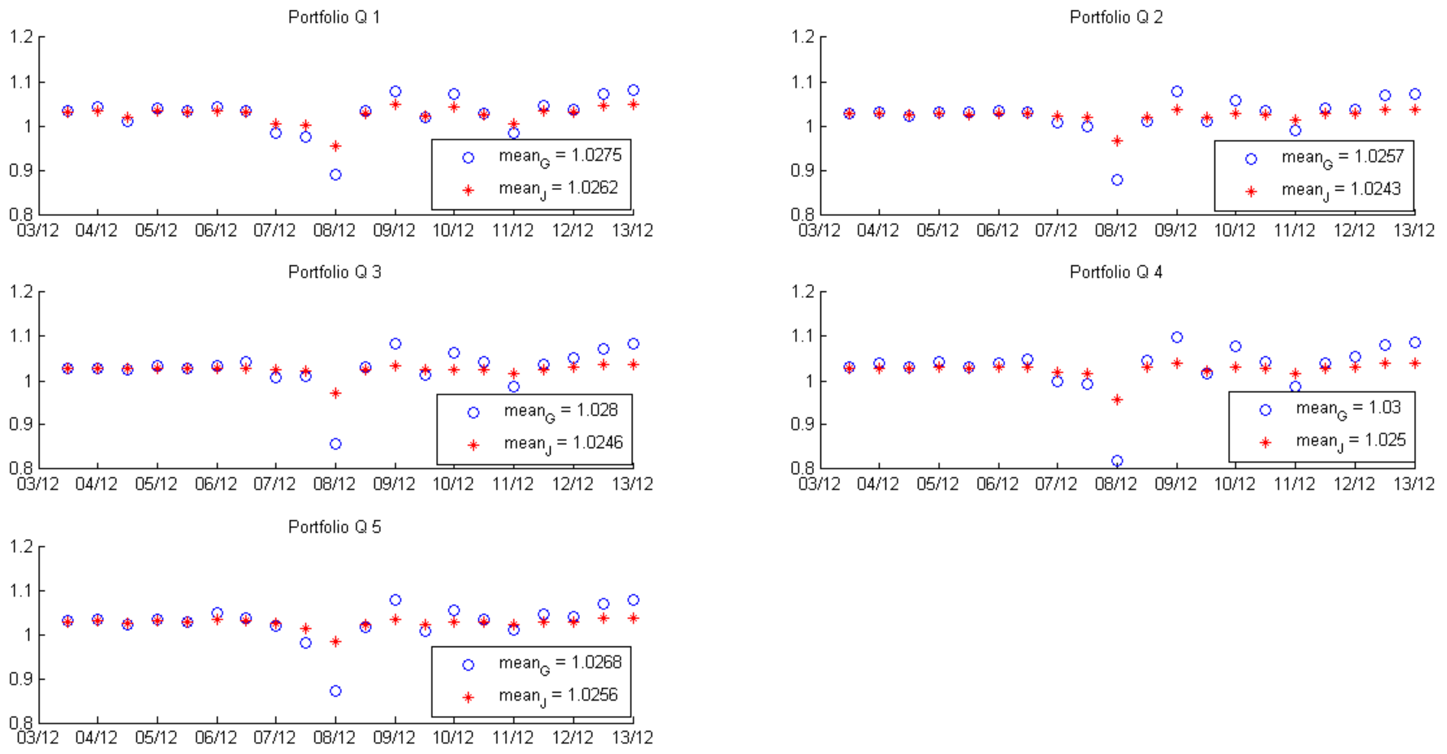
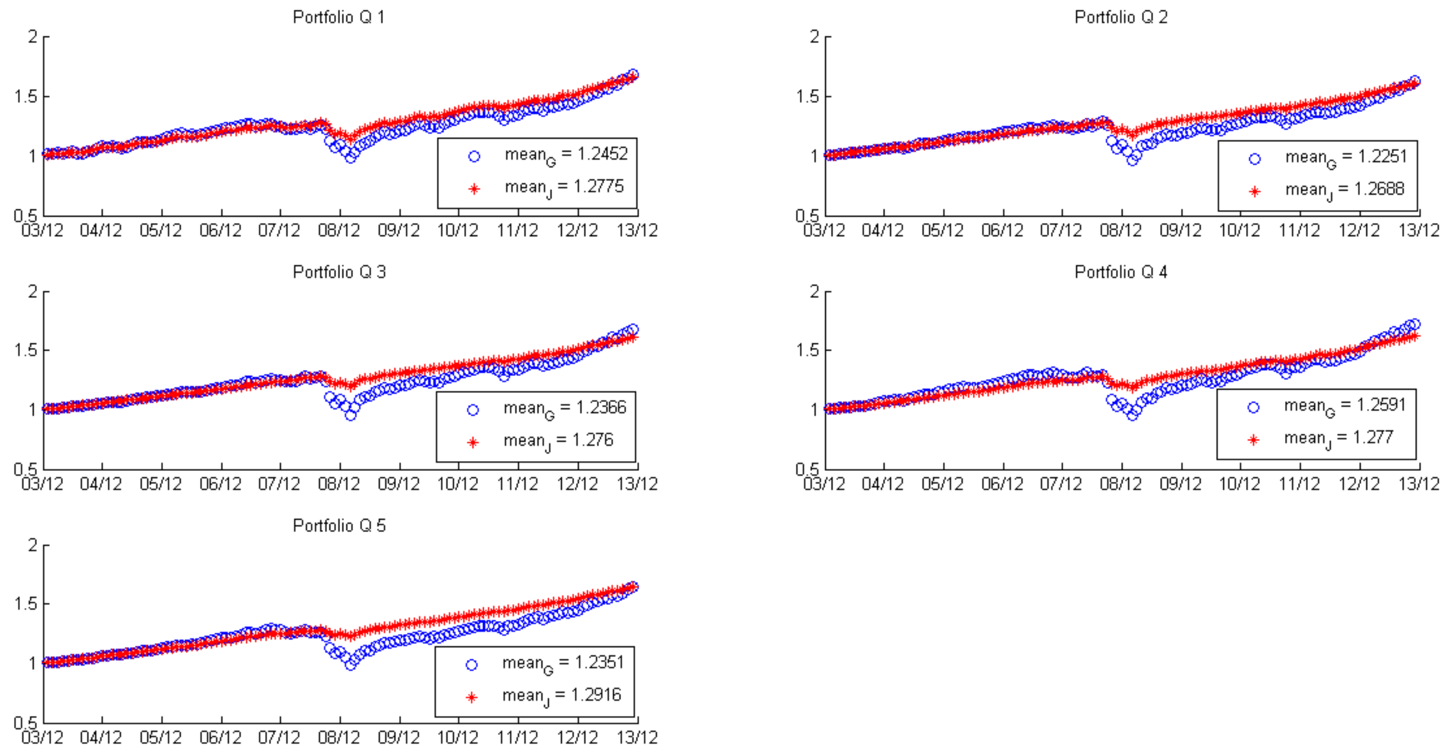


Figure 10.10: Out of sample test (III) performance for each of five quintile portfolios with Gaussian and Johnson assumptions. $W_0 = 1$



Chapter 11

Conclusion on Part III

In this part, we have examined multi-period portfolio choice problems of an investor with CRRA utility function in an incomplete market with Gaussian and Johnson-$S_U$returns. Our market model is a restricted VAR(1) process. We use dynamic programming to find optimal solutions and use Gauss-Hermite quadrature to compute efficiently conditional expectations.

Our numerical results suggest that 1) there are noticeble differences in optimal allocations between Johnson and Gaussian policies; 2) the Johnson policy is more conservative than the Gaussian policy; and 3) the distributional assumption is important, especially during market turbulence. Consequently, we suggest to use Johnson returns in financial modeling during difficult market periods.

Part IV

General Conclusion

This thesis studies methods to solve optimal portfolio choice problems by dynamic programming. Part II proposes a forward version of DP. The recursive algorithm is based on simulation of asset returns and regression of realized values on basis functions depending on decision and state variables. Compared to a standard BDP method with discretization and GH quadrature to compute optimal portfolios, our method offers similar or even better performance in terms of certainty equivalent rates. In addition, our approach is much more time-efficient as the number of assets increases. It is thus a possible way to partially handle the curse of dimensionality in dynamic programming.

Part III presents a standard backward DP technique with Gauss-Hermite quadrature. We apply this technique to an environment where risky assets' returns follow a restricted VAR dynamic with Johnson-S_U noises. Our numerical tests using historical data suggest that the optimal policy in the Johnson-S_U return environment differs from the Gaussian environment. In period of market crises, the Johnson-S_U distribution seems to be a more appropriate assumption for fund managers.

Bibliography

[1] Noël Amenc, Susan Curtis, and Lionel Martellini. The alpha and omega of hedge fund performance measurement. *Unpublished Work*, 2003.

[2] Pierluigi Balduzzi and Anthony W. Lynch. Transaction costs and predictability: some utility cost calculations. *Journal of Financial Economics*, 52(1):47 – 78, 1999.

[3] Nicholas Barberis. Investing for the long run when returns are predictable. *Journal of Finance*, 55:225–264, 2000.

[4] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1st edition, 1996.

[5] Michael W. Brandt. Estimating portfolio and consumption choice: A conditional euler equations approach. *The Journal of Finance*, 54(5):1609–1645, 1999.

[6] Michael W. Brandt. Chapter 5 - portfolio choice problems. In *Handbook of Financial Econometrics: Tools and Techniques*, volume 1 of *Handbooks in Finance*, pages 269 – 336. North-Holland, 2010.

[7] Michael W. Brandt, Amit Goyal, Pedro Santa-Clara, and Jonathan R. Stroud. A simulation approach to dynamic portfolio choice with an application to learning about return predictability. *The Review of Financial Studies*, 18(3):831–873, 2005.

[8] Michael J. Brennan, Eduardo S. Schwartz, and Ronald Lagnado. Strategic asset allocation. *Journal of Economic Dynamics and Control*, 21:1377–1403, 1997.

[9] Chris Brooks and Harry M. Kat. The statistical properties of hedge fund index returns and their implications for investors. *Journal of Alternative Investmentsl*, 5:26–44, 2002.

[10] David B. Brown and James E. Smith. Dynamic portfolio optimization with transaction costs: Heuristics and dual bounds. *Management Science*, 57(10):1752–1770, 2011.

[11] John Y. Campbell. Stock returns and the term structure. *Journal of Financial Economics*, 18, 1987.

[12] John Y. Campbell and Luis M. Viceira. Consumption and portfolio decisions when expected returns are time varying. *Quarterly Journal of Economics*, 114:433 – 495, 1999.

[13] John Y. Campbell and Luis M. Viceira. Consumption and portfolio decisions when expected returns are time varying. *The Quarterly Journal of Economics*, 114,2:433–495, 1999.

[14] Anant Chiarawongse, Seksan Kiatsupaibul, Sunti Tirapat, and Benjamin Van Roy. Portfolio selection with qualitative input. *Journal of Banking & Finance*, 36(2):489 – 496, 2012.

[15] Pilsun Choi and Kiseok Nam. Asymmetric and leptokurtic distribution for heteroscedastic asset returns: The s_u-normal distribution. *Journal of Empirical Finance*, 15(1):41 – 63, 2008.

[16] Richard Courant. *Differential and Integral Calculus*. Number v. 1. John Wiley & Sons, 1988.

[17] John C. Cox and Chi-Fu Huang. Optimal consumption and portfolio policies when assets prices follow a diffusion process. *Journal of Economic Theory*, 49:33–83, 1989.

[18] Jaksa Cvitanic and Ioannis Karatzas. Contingent claim valuation and hedging with constrained portfolio. 65:13–33, 1996.

[19] Robert M. Dammon, Chester S. Spatt, and Harold H. Zhang. Optimal consumption and investment with capital gains taxes. *Review of Financial Studies*, 14(3):583–616, 2001.

[20] Sanjiv Ranjan Das and Rangarajan K. Sundaram. An approximation algorithm for optimal consumption/investment problems. *Intelligent Systems in Accounting, Finance & Management*, 11(2):55–69, 2002.

[21] Michel Denault, Eric Delage, and Jean-Guy Simonato. Dynamic portfolio choice: a simulation-and-regression approach. *Optimization and Engineering*, 18(2):369–406, 2017.

[22] Michel Denault and Jean-Guy Simonato. Dynamic portfolio choices by simulation-and-regression: Revisiting the issue of value function vs portfolio weight recursions. *Computers & Operations Research*, 79:174–189, March 2017.

[23] Eugene F. Fama and Kenneth R. French. Business conditions and expected returns on stocks and bonds. *Journal of Financial Economics*, 25, 1989.

[24] William Fung and David A. Hsieh. Empirical characteristics of dynamic trading strategies : the case of hedge funds. *Review of Financial Studies*, 10, 2001.

[25] Lorenzo Garlappi and Georgios Skoulakis. Numerical solutions to dynamic portfolio problems: The case for value function iteration using taylor approximation. *Computational Economics*, 33(2):193–207, 2009.

[26] Lorenzo Garlappi and Georgios Skoulakis. Solving consumption and portfolio choice problems: The state variable decomposition method. *Review of Financial Studies*, 23(9):3346 – 3400, 2010.

[27] Campbell R. Harvey and Akhtar Siddique. Conditional skewness in asset pricing tests. *The Journal of Finance*, 55(3):1263–1295, 2000.

[28] Ronald A. Howard. *Dynamic Programming and Markov Processes.* Technology Press of the Massachusetts Institute of Technology, 1960.

[29] Norman L. Johnson. Systems of frequency curves generated by methods of translation,. *Biometrika*, 36:149–176, 1949.

[30] Eric Jondeau, Ser-Huang Poon, and Michael Rockinger. *Financial Modeling Under Non-Gaussian Distributions.* London: Springer, 2007.

[31] Kenneth L. Judd. *Numerical Methods in Economics.* MIT, Press, Cambridge, MA, 1998.

[32] Shmuel Kandel and Robert F. Stambaugh. On the predictability of stock returns: An asset allocation perspective. *Journal of Finance*, 51:385–424, 1996.

[33] Leonid Kogan and Raman Uppal. Risk aversion and optimal portfolio policies in partial and general equilibrium economies. November 2001.

[34] Francis A. Longstaff and Eduardo S. Schwartz. Valuing american options by simulation: a simple least-squares approach. *Review of Financial studies*, 14(1):113–147, 2001.

[35] David G. Luenberger. *Investment Science*. Oxford University Press, 1997.

[36] Anthony W. Lynch. Portfolio choice and equity characteristics: Characterizing the hedging demands induced by return predictability. *Journal of Financial Economics*, 62(1):67–130, 2001.

[37] Anthony W. Lynch. Portfolio choice and equity characteristics: Characterizing the hedging demands induced by return predictability. *Journal of Financial Economics*, 62:67–130, 2002.

[38] Anthony W. Lynch and Pierluigi Balduzzi. Predictability and transaction costs: The impact on relalancing rules and behavior. *Journal of Finance*, 55:2285–2310, 2000.

[39] Anthony W. Lynch and Sinan Tan. Multiple risky assets, transaction costs, and return predictability: Allocation rules and implications for u.s. investors. *Journal of Financial and Quantitative Analysis*, 45:1015–1053, 2010.

[40] Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[41] Robert C. Merton. Lifetime portfolio selection under uncertainty:the continuous time case. *The Review of Economics and Statistics*, 51:pp. 247–257, 1969.

[42] Robert C. Merton. Optimum consumption and portfolio rules in a continuous-time model. *Journal of Economic Theory*, 3:373–413, 1971.

[43] Robert C. Merton. Theory of finance from the perspective of continuous time,. *Journal of Financial and Quantitative Analysis*, 7:1851–1872, 1975.

[44] Todd Mitton and Keith Vorkink. Equilibrium underdiversification and the preference for skewness. *Review of Financial Studies*, 20(4):1255–1288, 2007.

[45] Naguez Naceur and Jean-Luc Prigent. Optimal portfolio positioning within generalized johnson distributions. *Working paper*, 2014.

[46] Juliana Nascimento and Warren Powell. An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Mathematics of Operations Research*, 34(1):210 – 237, 2009.

[47] Juliana Nascimento and Warren Powell. Dynamic programming models and algorithms for the mutual fund cash balance problem. *MANAGEMENT SCIENCE*, 56(5):801–815, 2010.

[48] A. Nedic and Dimitri P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13:79–110, 2003.

[49] Alexander Passow. Omega portfolio construction with johnson distributions. *FAME Research Paper n.120*, 2004.

[50] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics), 1st edition.* Wiley-Interscience, 2007.

[51] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics), 2nd edition.* Wiley-Interscience, 2011.

[52] Ilya O. Ryzhov and Warren B. Powell. Bayesian active learning with basis functions. pages 143–150, 2011.

[53] Paul A. Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *Review of Economics and Statistics*, 51:239 – 246, 1969.

[54] Jean-Guy Simonato. GARCH Processes with Skewed and Leptokurtic Innovations: Revisiting the Johnson Su Case. *Finance Research Letters*, Vol. 9, no 4, December 2012.

[55] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, 1st edition, 1998.

[56] Jules H. van Binsbergen and Michael W. Brandt. Solving dynamic portfolio choice problems by recursing on optimized portfolio weights or on the value function? *Computational Economics*, 29(3):355–367, 2007.

[57] Jessica A. Wachter. Portfolio and consumption decisions under mean-reverting returns: An exact solution for complete markets. *Journal of Financial and Quantitative Analysis*, 37:63–91, 2002.

Appendices

Appendix A

Gauss-Hermite Quadrature

In numerical analysis, a quadrature rule, which is usually stated as a weighted sum of function values evaluated at specified points within the domain of integration, is used to approximate the definite integral of a certain function. A Gaussian quadrature rule seeks to obtain the best estimate of an integral by carefully picking n optimal abscissas $x_i, i = 1, \ldots, n$. This class of quadrature rules, named after Carl Friedrich Gauss, yields exact results for polynomials up to $2n-1$ degree.

The Gauss-Hermite quadrature is a Gaussian quadrature over $(-\infty, \infty)$ with weighting function $w(x) = e^{-x^2}$. It approximates the value of integrals of the following kind:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i),$$

where n is the number of nodes x_i used. The nodes are roots of the Hermite polynomials $H_n(x), i = 1, 2, \ldots, n$, which can be determined recursively by

$$H_0(x) = 1, \qquad H_1(x) = 2x, \qquad H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x),$$

and the associated weights ω_i are given by

$$\omega_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}.$$

For further details on Gaussian quadrature methods, we refer to the Chapter 7 in [31].

Appendix B

Parameters of the VAR(1) Dynamic Studied in Part I

This appendix presents the custom-built parameters of the VAR(1) dynamic studied in the FDP part for cases with 1, 2 and 3 risky assets. The VAR(1) dynamic is given by:

$$\boldsymbol{R}_{t+1} = A_0 + A_1 \boldsymbol{R}_t + \boldsymbol{\xi_{t+1}}, \tag{B.1}$$

where $\boldsymbol{\xi_{t+1}}$ is a vector of Gaussian error terms with zero mean and constant covariance matrix Σ. The parameter matrix A_0, A_1 and Σ are given by

- **One risky asset:**

$$A_0 = 0.0055, \quad A_1 = 0.2967, \quad \Sigma = 0.0021.$$

- **Two risky assets:**

$$A_0 = \begin{bmatrix} 0.0055 \\ 0.0059 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0.1989 & -0.1436 \\ 0.1285 & 0.0809 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.0021 & 0.0018 \\ 0.0018 & 0.0025 \end{bmatrix}.$$

- **Three risky assets:**

$$A_0 = \begin{bmatrix} 0.0062 \\ 0.0054 \\ 0.0059 \end{bmatrix}, A_1 = \begin{bmatrix} 0.1959 & 0.0099 & -0.1492 \\ 0.0236 & 0.1253 & -0.0195 \\ 0.1150 & 0.0450 & 0.0556 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0022 & 0.0017 & 0.0018 \\ 0.0017 & 0.0019 & 0.0019 \\ 0.0018 & 0.0019 & 0.0025 \end{bmatrix}.$$

Appendix C

BDP Algorithm

In the numerical examples of Part II, a BDP approach is used as benchmark for our FDP approach. This appendix illustrates this BDP algorithm with the CRRA utility function and one risky asset whose returns are Gaussian and follow a AR(1) process and given by:

$$\hat{R}_{t+1} = A_0 + A_1 \hat{R}_t + \xi_{t+1}, \tag{C.1}$$

where $\xi_{t+1} \sim \mathcal{N}(0, \sigma_\xi)$ is a Gaussian-distributed noise term. The BDP algorithm presented here discretizes the state space and uses Gauss-Hermite quadrature to compute conditional expectations.We refer to Appendix F for the discretization method and to Appendix A for the derivation of Gauss-Hermite quadrature formula within the context of the restricted VAR(1) dynamic.

BDP Algorithm - CRRA - AR(1) Returns

```
Preparation steps:
  a) Choose $Q$, the number of nodes used in Gauss-Hermite quadrature.
     Determine the vector of GH nodes $\mathbf{m} = [m_1, \dots, m_k, \dots, m_Q]$ and their
     corresponding weights $\mathbf{p} = [p_1, \dots, p_k, \dots, p_Q]$.
  b) Discretize the state variable $\hat{R}_t$ at each time step: $\mathcal{G}_t := \{R_t^j\}_{j=1}^{N_g}$,
     $\forall t = 1, \dots, T$, where $N_g$ is the number of discretization points.
  c) Initialize $V_T(R_T^j) = \frac{1}{1-\gamma}$, $\forall R_T^j \in \mathcal{G}_T$.
Solve the *reduced* Bellman equation for each time $t$ and discretized point for
asset returns:
  d) For $t = T-1, \dots, 0$, do
     d)1. For all $R_t^j \in \mathcal{G}_t, j = 1, \dots, N_g$, do
        d)1.1. Using $\mathbf{m}$ and $\mathbf{p}$ determined in step 1, compute $R_{t+1}^{(j,k)}$ by :
               $$R_{t+1}^{(j,k)} = A_0 + A_1 R_t^j + \sigma_\xi \sqrt{2} m_k$$
        d)1.2. Optimize over $x_t$:
               $$V_t(R_t^j) = \max_{x_t \in [0,1]} \frac{1}{\sqrt{\pi}} \sum_{k=1}^{Q} p_k \left(R_f + x_t R_{t+1}^{(j,k)}\right)^{1-\gamma} \mathcal{V}_{t+1}(R_{t+1}^{(j,k)}),$$
               where $\mathcal{V}_{t+1}(R_{t+1}^{(j,k)})$ is obtained by linear interpolation using the
               grid $\mathcal{G}_{t+1}$ and $\{\mathcal{V}_{t+1}(R_{t+1}^j)\}_{j=1}^{N_g}$.
               Let $x_t^*(R_t^j)$ denote the optimal solution.
        d)1.3. If $j < N_g, j = j+1$ and go to step d)1.; else return $\{\mathcal{V}_t(R_t^j)\}_{j=1}^{N_g}$ and
               $\{x_t^*(R_t^j)\}_{j=1}^{N_g}$.
     d)2. If $t > 0, t = t-1$ and go to step d); else return $\mathcal{V}_0(R_0^j)$ and $x_0^*(R_0^j)$.
```

Figure C.1: A backward DP algorithm for a multi-period OPCP with 1 risky asset whose returns are Gaussian and follow an AR(1) process, CRRA utility.

Appendix D

Correlation between two Johnson S_u random variables

From Choi and Nam (2008), the correlation ρ_{u_i,u_j} between two Johnson S_u random variables with parameters a_i, b_i, a_j and b_j can be computed with

$$\rho_{u_i,u_j} = \phi \times \left[\begin{array}{c} \frac{1}{2} e^{\rho_{z_i,z_j} v_i v_j} \cosh\left(-\Gamma_i - \Gamma_j\right) \\ -\frac{1}{2} e^{-\rho_{z_i,z_j} v_i v_j} \cosh\left(-\Gamma_i + \Gamma_j\right) - \sinh\left(-\Gamma_i\right) \sinh\left(-\Gamma_j\right) \end{array} \right]$$

where ρ_{z_i,z_j} is the correlation between the standard normal random variates and

$$\phi = \frac{\exp\left(\frac{1}{2}\left(v_i^2 + v_j^2\right)\right)}{\sqrt{V_i}\sqrt{V_j}}$$

with $v_i = \frac{1}{b_i}$, $\Gamma_i = \frac{a_i}{b_i}$ and $V_i = \frac{1}{2}\left(w - 1\right)\left(w \cosh\left(2\Gamma\right) + 1\right)$.

Appendix E

Maximum Log-Likelihood Estimation

In this section, we derive the Maximum Log-Likelihood (MLL) function to estimate the parameters in the restricted VAR(1) model with Johnson-S_u noise terms. The MLL function of the same model with Gaussian noises is standard and can be performed using any computing software such as MATLAB.

Let's denote the vector containing return of the risky asset and the log-dividend yield at time t by $\mathbf{y_t} := [\mathbf{r_t}, \delta_t]'$. The i-th component of $\mathbf{y_t}$ evolves according to:

$$y_{i,t} = \alpha_i + \beta_i \delta_{t-1} + \epsilon_{i,t}, \qquad i = 1, \ldots, N+1$$

where N is the number of risky assets, the residuals $\epsilon_{i,t} = \sigma_{u,i} u_{i,t}$ and $u_{i,t}$ is a standard Johnson-S_u variable given by

$$u_{i,t} = \frac{\sinh\left(\frac{z_{i,t}-a_i}{b_i}\right) - M_y}{\sqrt{V_y}},$$

with $M_y = -\omega^{1/2}\sinh(\Omega), V_y = 0.5(\omega - 1)(\omega \cosh(2\Omega) + 1), \omega = e^{(1/b_i)^2}, \Omega = a_i/b_i$, and $z_{i,t}$ is a standard Normal random error.

We can now express the standard Normal error $z_{i,t}$ as a function of the residues $\epsilon_{i,t}$

(and thus $y_{i,t}$):

$$\begin{aligned} z_{i,t} &= a_i + b_i \sinh^{-1}(M_y + \sqrt{V_y}\epsilon_{i,t}/\sigma_{u,i}), \\ \frac{\partial z_{i,t}}{\partial \epsilon_{i,t}} &= \frac{b_i\sqrt{V_y}}{\sigma_{u,i}\sqrt{1+\big(M_y+\sqrt{V_y}\epsilon_{i,t}/\sigma_{u,i}\big)^2}}, \end{aligned}$$

where $\epsilon_{i,t} = y_{i,t} - (\alpha_i + \beta_i\delta_{t-1})$.

The density function of $\epsilon_{i,t}$ and its log-likelihood function are given by:

$$\begin{aligned} f(\epsilon_{i,t}) &= f(z_{i,t})\left|\frac{\partial z_{i,t}}{\partial \epsilon_{i,t}}\right| \\ &= \frac{1}{2\pi}\exp\left\{-\frac{1}{2}\left(a_i + b_i\sinh^{-1}(M_y + \frac{\sqrt{V_y}\epsilon_{i,t}}{\sigma_{u,i}})\right)^2\right\} \\ &\qquad \times \frac{b_i\sqrt{V_y}}{\sigma_{u,i}}\frac{1}{\sqrt{1+\left(M_y+\frac{\sqrt{V_y}\epsilon_{i,t}}{\sigma_{u,i}}\right)^2}}, \qquad \text{(E.1)} \\ ll(\epsilon_{i,t}) &= \log\left(f(\epsilon_{i,t})\right) \\ &= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\left(a_i + b_i\sinh^{-1}(M_y + \frac{\sqrt{V_y}\epsilon_{i,t}}{\sigma_{u,i}})\right)^2 + \log(b_i) \\ &\qquad + \frac{1}{2}\log(V_y) - \log(\sigma_{u,i}) - \frac{1}{2}\log\left(1+\left(M_y+\frac{\sqrt{V_y}\epsilon_{i,t}}{\sigma_{u,i}}\right)^2\right). \qquad \text{(E.2)} \end{aligned}$$

Similarly, by the relation $\epsilon_{i,t} = y_{i,t} - (\alpha_i + \beta_i\delta_{t-1})$, and thus $\partial\epsilon_{i,t}/\partial y_{i,t} = 1$, which gives the log-likelihood function of $y_{i,t}$:

$$\begin{aligned} ll(y_{i,t}) &= \log\Big(f(y_{i,t})\Big) = \log\left(f(\epsilon_{i,t})\left|\frac{\partial \epsilon_{i,t}}{\partial y_{i,t}}\right|\right) \\ &= -\frac{1}{2}\log(2\pi) - \frac{1}{2}\left(a_i + b_i\sinh^{-1}(M_y + \frac{\sqrt{V_y}(y_{i,t}-\alpha_i-\beta_i\delta_{t-1})}{\sigma_{u,i}})\right)^2 \\ &\qquad + \log(b_i) + \frac{1}{2}\log(V_y) - \log(\sigma_{u,i}) \\ &\qquad - \frac{1}{2}\log\left(1+\left(M_y+\frac{\sqrt{V_y}(y_{i,t}-\alpha_i-\beta_i\delta_{t-1})}{\sigma_{u,i}}\right)^2\right). \qquad \text{(E.3)} \end{aligned}$$

Appendix F

Discretization Methods

There are various ways to discretize the state variable δ_t into grids $\mathcal{D}_t$. It is important to use the appropriate method depending on the problem-specific characteristics, because an improperly chosen grid may strongly deteriorate the algorithm's performance. This appendix proposes several discretization methods all suitable for the OPCP presented previously.

The first approach fixes the maximum and minimum points on the grid at each time step at the conditional expectation plus α_u times or minus α_l times of the conditional standard deviation, i.e. the maximum and minimum on $\mathcal{D}_t$ are respectively given by $\mathbb{E}[\delta_t|\delta_{t-1}] + \alpha_u \sigma(\delta_t|\delta_{t-1})$ and $\mathbb{E}[\delta_t|\delta_{t-1}] - \alpha_l \sigma(\delta_t|\delta_{t-1})$.

The second method simulates N_{sim} trajectories for dividend yield with the estimated parameters and fixes the maximum and minimum points to respectively the 99th and the 1st simulated percentiles (or alternatively to the maximal and minimal simulated values). We refer this grid as the *simulated grid*.

A third method is to use a grid determined implicitly by nodes used in Gauss-Hemite interpolation. This grid has the advantage to avoid extrapolations in the computations of $V_t(\delta_t)$. More precisely, let $\delta_t^m \in \mathcal{D}_t$ be an arbitrary point on the grid at time stage t. As discussed in section 8: given a value of δ_t^m, all values of δ_{t+1} needed (also referred as "implied" dividend-nodes) to compute $V_t(\delta_t^m)$ by GH quadrature can be determined by equation (8.6) for the Gaussian case or by equation (8.9) for the Johnson case. Denote this set of "implied" nodes by $\{\delta_{t+1}|\delta_t^m\}$. We repeat the same procedures for all points

δ_t^m on grid $\mathcal{D}_t$. We obtain N_g sets of "implied" nodes $\{\delta_{t+1}|\delta_t^m\},\ m = 1,\ \ldots,\ N_g$ where N_g is the number of discretization points. The maximum and minimum points of $\mathcal{D}_{t+1}$ are then given respectively by the maximum and minimum of all sets of "implied" nodes:

$$\delta_{t+1}^{\max} := \max_{m=1,\ \ldots,\ N_g} \{\delta_{t+1}|\delta_t^m\}, \qquad \delta_{t+1}^{\min} := \min_{m=1,\ \ldots,\ N_g} \{\delta_{t+1}|\delta_t^m\}.$$

And finally we discretize the range between $[\delta_{t+1}^{\min}, \delta_{t+1}^{\max}]$ into $N_g - 1$ intervals to obtain the grid for time stage $t + 1$. The same steps are repeated until the final stage T is attained. We refer this grid as the *implicit grid*.

An important issue when employing the *implicit grid* is that the range of $\mathcal{D}_t$ increases extremely quickly with t, especially for the Johnson case. We list maximum and minimum at various t for both Gaussian and Johnson distribution in Table F.1 assuming initial state $\delta_0 = 1$. If a wide range (minimum – maximum) is discretized into equally spaced intervals to form $\mathcal{D}_t$, most of points on the grid would be associated to optimal allocations of either 0% or 100%, which gives little information, while only a small part of $\mathcal{D}_t$ is associated to optimal allocations between 0% and 100% (due to our no short sale and no borrowing constraints), which is nonetheless the useful part.

Table F.1: Range of $\mathcal{D}_t$ for Gaussian and Johnson cases ($\delta_0 = 1$).

	Gaussian		Johnson	
t	min	max	min	max
1	-0.01	1.99	-84.3	160.5151
3	-1.99	3.92	-251.9	473.7110
5	-3.94	5.81	-415.3	779.2857
10	-8.58	10.33	-806.8	1511.1
20	-17.85	18.58	-1521.0	2846.2
40	-31.17	32.33	-2710.6	5070.1
60	-42.20	43.08	-3640.5	6808.4

To solve the previous issue, one can increase the number of discretization points on the grid, but the computational burden also increases exponentially. Another solution is to refine partially the grid $\mathcal{D}_t$ while keeping the number of points relatively small. That is to say, the minimum (min) and maximum (max) points on $\mathcal{D}_t$ are as given by the *implicit grid*. Then we choose a range $[lb, ub]$, $\min < lb < ub < \max$ to be refined and discretize it into $N_{ref} - 1$ equally spaced intervals, where $N_{ref} < N_g$ is a user-chosen parameter representing the number of points on the refined part. Next, the ranges $[\min, lb]$ and

$[ub, \max]$ are discretized into $N_{sub} := (N_g - N_{ref})/2 - 1$ equally spaced intervals. Note that attentions must be paid while choosing N_g and N_{ref} so that N_{sub} is an integer. And finally, the three sub-grids are assembled together to produce a new grid $\mathcal{D}_t^{ref}$. We refer this grid as the *hybrid grid*.

Appendix G

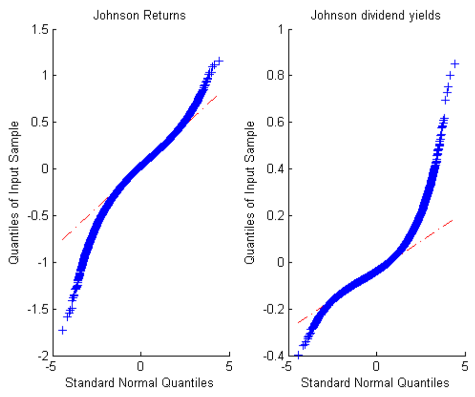
Values of parameter used in our simulation study

Table G.1 presents values of parameters used for Gaussian and Johnson-S_u distributions in our simulation study of section 10.4.1. The choice of these values are heuristic but somehow realistic. Figure G.1 shows qq-plots for two simulated single-period samples (Johnson and Gaussian) containing $100,000$ paths each, with initial dividend yield $\delta_0 = 0$. The Johnson sample manifests clearly heavier tails than the Gaussian sample.

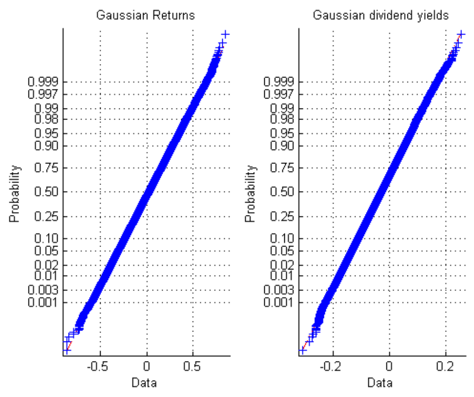
Table G.1: Parameters used in simulation study.

	Johnson		Gaussian	
	Risky	Dividend	Risky	Dividend
α	0.02	-0.03	0.02	-0.03
β	0.015	0.975	0.015	0.975
σ_u	0.2	0.0632	0.2	0.0632
a	0.5	-0.76	n.a.	n.a.
b	2	1.52	n.a.	n.a.
c	0.4855	-0.6764	n.a.	n.a.
d	1.6961	1.0454	n.a.	n.a.
ρ_u	-0.85		n.a.	n.a.
ρ_z	-0.8674		-0.85	

Figure G.1: Simulated samples with parameters in Table G.1. 100,000 paths, $\delta_0 = 0$ and single period.



(a) Simulated Johnson paths



(b) Simulated Gaussian paths