Dynamic Programming for Portfolio Problems

Yongyang Cai, iCME, Stanford University

Dr. Kenneth L. Judd, Hoover Institution

September 4, 2009

This talk is based on the PhD thesis of Yongyang Cai. I have sent this to you since it will form the basis for my talk.

Please excuse the preliminary nature of this draft.

Any comments will be welcomed.

DYNAMIC PROGRAMMING AND ITS APPLICATION IN ECONOMICS AND FINANCE

A DISSERTATION
SUBMITTED TO THE INSTITUTE FOR
COMPUTATIONAL AND MATHEMATICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Yongyang Cai
June 2009

Abstract

Multistage decision problems are numerically challenging. Typically the work to solve such problems is an exponential function both with respect to the number of stages and the number of decision parameters. The problems have been researched extensively and a wide variety of methods to solve them have been proposed. Inevitably all methods are limited in the size of problem they can solve. The purpose of our work is to develop a new more ecient algorithm and one that is suitable to run on parallel architectures and in so doing extend signicantly the size of problems that are tractable.

We present a numerical dynamic programming algorithm that has three components: optimization, approximation and integration. A key feature of the approximation methods we use is to preserve mathematical features such as convexity and dierentiability, which enables us to use powerful optimization methods. To illustrate the eciency of the new method we present extensive results on optimal growth models and on dynamic portfolio problems obtained from implementation of the algorithm designed to run on the Condor Master-Worker system.

Contents

List of Tables

List of Figures

Chapter 1

Optimal Decision-Making Problem

In the real world, we often encounter a lot of optimal decision-making problems, which are also called optimal control problems. In the following, two simple examples are given, the first one is a finite horizon dynamic asset allocation problem arising in finance, and the second one is an infinite horizon deterministic optimal growth model arising in economics.

1.1 Dynamic Portfolio Problem

A dynamic portfolio problem is a multi-stage asset allocation problem, which is very popular in asset management.

Let W_t be an amount of money planned to be invested at stage t. Assume that available assets for trading are n stocks and a bond, where stocks have a random return vector $R = (R_1, \ldots, R_n)$ and the bond has a riskfree return R_f for each period. If $x_t W_t = (x_{t1}, \ldots, x_{tn})^\top W_t$ is a vector of money invested in n risky assets at time t, then money invested in the riskless asset is $(1 - e^\top x_t)W_t$ where e is a column vector with 1 everywhere. Thus, wealth at next stage is

$$W_{t+1} = W_t(R_f(1 - e^\top x_t) + R^\top x_t), \tag{1.1}$$

for $t = 0, 1, \ldots, T-1$.

A simple dynamic asset allocation problem is to find an optimal portfolio x_t at each stage t such that we have a maximal expected terminal utility, i.e.,

$$V_0(W_0) = \max_{x_t, 0 \le t < T} E[u(W_T)],$$

where W_T is the terminal wealth derived from the recursive formula (1.1) with give W_0, and u is the terminal utility function.

1.2 Deterministic Optimal Growth Model

A simplest infinite-horizon economic problem is a discrete-time deterministic optimal growth model with one good and one capital stock, which is also called as a wealth accumulation problem. It is to find an optimal consumption decision such that total utility over an infinite-horizon time is maximized, i.e.,

$$\begin{aligned} V(k_0) = \max_{c_t} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t) \\ \text{s.t.} \quad & k_{t+1} = F(k_t) - c_t, \quad t \geq 0, \end{aligned}$$

where k_t is the capital stock at time t with k_0 given, c_t is the consumption, β is the discounter factor, $F(k) = k + f(k)$ with $f(k_t)$ the aggregate net production function, $u(\cdot)$ is the utility function, and $V(\cdot)$ is called as the value function.

Chapter 2

A Review of Dynamic Programming Theory

The previous two models are two kinds of sequential decision-making problems, which consist of a state process initialized at some given initial state, a sequence of decisions in choice sets, a sequence of transition probabilities of states, and a utility function u (Rust [47]). Dynamic programming (DP), also known as backward induction, is a recursive method to solve these sequential decision problems. It can be applied in both discrete time and continuous time settings. When there is no close-formed solution for a continuous time problem, the most commonly used numerical method is to approximate the continuous time problem by the discrete time version with small time intervals between successive decisions, and then to solve the approximate discrete time problem by using discrete time DP methods. This is practical because the solutions of the discrete time DP problems will converge to the continuous time solution under some general conditions, as the time intervals tend to zero (Kushner [30]). In this monograph, we will omit the continuous time problems and focus on discrete time DP problems.

In Bellman [3], the common structure underlying sequential decision problems is shown and DP is applied to solve a wide class of sequential decision problems. In any DP problems, there are two key variables: a state variable and an action variable (the action is also called decision variable or control variable, and the variables can be vectors). The optimal decision is a function dependent on the state variable and time t. The function is called the policy function in Bellman [3]. The equivalence between the original sequential decision problems and their corresponding DP problems is called the principle of optimality, which is described in Bellman [3] as: "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision". Thus, the multi-stage or infinite-period sequential decision problems are reduced to a sequence of one-period optimal control problems.

2.1 General Model

The finite horizon optimal decision-making problems can be expressed in the following general model:

$$V_t(x_t) = \max_{a_s \in D(x_s,s), s \geq t} \sum_{s=t}^{T-1} \beta^{s-t} E[u_s(x_s, a_s) \mid x_t] + \beta^{T-t} E[u_T(x_T) \mid x_t],$$

for $t = 0, 1, \ldots, T$, where T is finite, x_t is the state process with initial state x_0, and a_t is the action variable (x_t and a_t can be vectors), $u_t(x, a)$ is a utility function at time $t < T$ and $u_T(x)$ is the terminal utility function, β is the discount factor ($0 < \beta \leq 1$), $D(x_t, t)$ is a feasible set of a_t, $E(\cdot)$ is the expectation operator.

The DP model for this kind of finite horizon problems is the basic Bellman equation,

$$V_t(x) = \max_{a \in D(x,t)} \quad u_t(x, a) + \beta E\{V_{t+1}(x^+) \mid x, a\},$$

where $V_t(x)$ is called the value function at stage t with $V_T(x) = u_T(x)$, x^+ is the next-stage state at $t + 1$ conditional on the current-stage state x and the action a at t. From this model, we know that there are three main components in the numerical DP method: numerical optimization, numerical approximation and numerical integration.

The infinite horizon optimal decision-making problems have the following general form:

$$V(x_0) = \max_{a_t \in D(x_t)} \sum_{t=0}^{\infty} \beta^t E[u(x_t, a_t)],$$

where x_t is the state process with initial state x_0, and $0 \leq \beta < 1$.

The DP model for this kind of infinite horizon problems is

$$V(x) = \max_{a \in D(x)} \quad u(x, a) + \beta E\{V(x^+) \mid x, a\},$$

where $V(x)$ is called the value function, x^+ is the next-stage state conditional on the current-stage state x and the action a.

2.2 Examples

For the previous dynamic asset allocation model, its value functions satisfy the Bellman equation,

$$V_t(W_t) = \max_{x_t} \quad E\{V_{t+1}(W_t(R_f(1 - e^\top x_t) + R^\top x_t))\},$$

where W_t is the state variable and x_t is the control variable vector, for $t = 0, 1, \ldots, T - 1$, and the terminal value function is $V_T(W) = u(W)$.

For the previous deterministic optimal growth model, its value function satisfies the Bellman

equation,

$$V(k) = \max_{c} \quad u(c) + \beta V(F(k) - c),$$

where k is the state variable, and c is the control variable.

2.3 Theoretical Results of Dynamic Programming

Let Γ denote the Bellman operator,

$$\Gamma(f)(x) = \max_{a \in D(x)} \quad u(x,a) + \beta E\{f(x^+) \mid x, a\},$$

in which we suppose that f is a continuous and bounded function on a state space S, $u(x,a)$ is a continuous function of (x,a), the state space S is compact, the constraint sets $D(x)$ are compact for each $x \in S$, $\beta \in (0,1)$, and $E\{f(x^+) \mid x, a\}$ is a continuous function of (x,a). It is well known that Γ satisfies the contraction property, i.e.,

$$\|\Gamma(f) - \Gamma(g)\|_\infty \leq \beta \|f - g\|_\infty,$$

for any continuous and bounded functions f and g on the compact state space S. By the contraction mapping theorem, we know that there is a unique fixed point of the operator Γ. That is, the Bellman equation $\Gamma(f) = f$ has a unique solution V under the above assumptions. See the proof in Blackwell [5] and Denardo [14].

For infinite horizon problems, by the contraction property of Γ, it is well known that for any bounded and continuous function V_0,

$$\lim_{n \to \infty} \Gamma^n(V_0) = V^*,$$

where V^* is the fixed point of Γ, and Γ^n denotes n successive iterations of the Bellman operator Γ. Being motivated by the contraction property, value function iteration $V_{n+1} = \Gamma V_n$, is the simplest numerical procedure for finding V^*. V_n is called the n-th iterated value function.

Under regularity conditions and some general assumptions, Santos and Vigo-Aguiar [49] gave a numerical error bound for the converged value function from value function iteration for an infinite horizon problem,

$$\|V^* - \hat{V}_\varepsilon^{\Delta x}\| \leq \frac{\|\nabla^2 V^*\|}{2(1-\beta)} (\|\Delta x\|)^2 + \frac{\varepsilon}{1-\beta},$$

where β is the discount factor, $V^*(\cdot)$ is the exact value function (the fixed point of Γ), Δx is the mesh size in the finite element method of Santos and Vigo-Aguiar [49], $\hat{V}_\varepsilon^{\Delta x}(\cdot)$ is the converged estimated value function with a stopping criterion ε (i.e., $\|L\Gamma(\hat{V}_\varepsilon^{\Delta x}) - \hat{V}_\varepsilon^{\Delta x}\| \leq \varepsilon$, where L is the finite element approximation operator).

The above numerical error bound formula is extended to non-expansive approximation operator L (i.e., $\|Lf - Lg\| \leq \|f - g\|$ for any bounded Borel measurable functions f and g on the compact

space S) by Stachurski [51],

$$\|V^* - V_\pi\| \leq \frac{2}{1-\beta}\left(\beta\|\hat{V}_N - \hat{V}_{N-1}\| + \|L\Gamma(\hat{V}_N) - \Gamma(\hat{V}_N)\|\right),$$

or

$$\|V^* - V_\pi\| \leq \frac{2}{(1-\beta)^2}\left(\beta\|\hat{V}_N - \hat{V}_{N-1}\| + \|LV^* - V^*\|\right),$$

where $\hat{V}_N$ is the N-th iterated value function, π is the $\hat{V}_N$-greedy policy function (i.e., $\pi(x) = \arg\max_{a\in D(x)} \{u(x,a) + \beta E[\hat{V}_N(x^+) \mid x, a]\}$), $V_\pi(x)$ is the value following the policy π when starting at initial state x (i.e., $V_\pi(x) = \sum_{t=0}^{\infty} \beta^t E[u_t(x_t, a_t)]$).

Maldonado [35] showed that if the utility function is strongly concave, then the distance between the optimal policy function and the solution of the n-th iteration of the Bellman equation for an infinite horizon problem is lower than a constant times the square root of the distance between the value function and the n-th iteration of the Bellman operator Γ from any initial function V_0. That is,

$$\|\pi^* - \pi\| \leq c\|V^* - \hat{V}_N\|^{1/2},$$

where $c > 0$ is a constant, $\hat{V}_N$ is the N-th iterated value function, π is the $\hat{V}_N$-greedy policy function, and π^* is the optimal policy function.

Chapter 3

Numerical Methods for Dynamic Programming

When state variables and control variables are continuous, in general situations, the expectation operator and the maximum operator in dynamic programming (DP) models can only be computed approximately. Moreover, each iterated value function cannot be calculated and represented exactly by Computers. Therefore, some kinds of approximation must be done. We separate them into three parts: numerical integration, numerical optimization, and numerical approximation for functions.

3.1 Discretize the State Space

The simplest DP problems have a finite horizon T and a finite set of states, $X = \{x_i : 1 \leq i \leq m\}$. That is,

$$V_t(x_i) = \max_{a_i \in D(x_i)} \quad u_t(x_i, a_i) + \beta E V_{t+1}(x_i^+),$$

where $V_T(x) = u_T(x)$, a_i is the control variable, and $x_i^+ \in X$ is the next-stage state conditional on the current-stage state x_i, for $1 \leq i \leq m$, $0 \leq t < T$.

For an infinite horizon problem with a finite set of states, $X = \{x_i : 1 \leq i \leq m\}$, the above model becomes

$$V(x_i) = \max_{a_i \in D(x_i)} \quad u(x_i, a_i) + \beta E V(x_i^+),$$

or simply denoted as $V = \Gamma V$. We know that value function iteration $V_{n+1} = \Gamma V_n$ with an initial guess V_0 can be applied to find the solution V of the Bellman equation. Moreover, we could use acceleration methods such as policy function iteration, Gaussian iteration method and upwind Gauss-Seidel method, etc. We will not discuss them in details here, detailed discussion of them can be seen in Judd [28].

To solve a DP problem with a continuous state $x \in S$ approximately, the simplest numerical procedure is discretization method, which replaces the state space S by a finite set $X = \{x_i : 1 \leq i \leq$

$m\}$, of permissible values. Then we can apply the above value function iteration method for this new discrete-state DP problem. Discretization method essentially approximates V with a step function, since it implicitly treats any state between x_i and x_{i+1} as either x_i or x_{i+1}.

Discretization method is often inefficient since a large number of discrete states are required to get a well-approximated solution. In order to get a solution within $O(\varepsilon)$ error bound for a problem with n continuous states, we often need $O((1/\varepsilon)^n)$ points. So it becomes impractical for larger problems with a few state variables, because of the "curse of dimensionality". For problems with continuous state variables and smooth value functions, there are many alternatives to discretization, such as piecewise linear approximation, polynomial approximation, spline approximation, and so on. All these methods need some parameters to represent approximated functions. DP with these methods are referred as parametric dynamic programming algorithms.

3.2 Parametric Dynamic Programming

In DP problems, if state variables and control variables are continuous such that value functions are also continuous, then we have to use some approximation for the value functions, since computers cannot model the entire space of continuous functions. We focus on using a finitely parameterizable collection of functions to approximate value functions, $V(x) \approx \hat{V}(x; b)$, where b are parameters. The functional form $\hat{V}$ may be a linear combination of polynomials, or it may represent a rational function or neural network representation, or it may be some other parameterization specially designed for the problem. After the functional form is fixed, we focus on finding parameters b such that $\hat{V}(x; b)$ approximately satisfies the Bellman equation. The algorithm of the numerical DP with value function iteration can solve the Bellman equation approximately. See Judd [28].

Algorithm 3.1. *Numerical Dynamic Programming with Value Function Iteration for Finite Horizon Problems*

Initialization. Choose the approximation nodes, $X_t = \{x_{it} : 1 \le i \le m_t\}$, and choose functional form for $\hat{V}(x; b^t)$ for every $t < T$. Let $t = T - 1$ and $\hat{V}(x; b^T) = V_T(x) = u_T(x)$.

Step 1. Maximization step. Compute

$$v_i = \max_{a_i \in D(x_i,t)} u_t(x_i, a_i) + \beta E\{\hat{V}(x_i^+; b^{t+1}) \mid x_i, a_i\},$$

for each $x_i \in X_t$, $1 \le i \le m_t$.

Step 2. Fitting step. Using an appropriate approximation method, compute the b^t such that $\hat{V}(x; b^t)$ approximates (x_i, v_i) data.

Step 3. If $t = 0$ STOP; else $t \to t - 1$ and go to step 1.

Operations of expectation (numerical integration), numerical optimization and numerical fitting in the above algorithm have three numerical (absolute) errors, $\varepsilon_1, \varepsilon_2, \varepsilon_3$, in each iteration ($\varepsilon_2$ may be large if numerical optimization solver returns a local optimizer which is not the global one). Then

$$\begin{aligned} -\epsilon + \beta E\{\hat{V}(x_*^+; b^t) - V_{t+1}(x_*^+) \mid x, a^*\} &\leq \hat{V}(x; b^t) - V_t(x) \\ &\leq \epsilon + \beta E\{\hat{V}(x_*^+; b^t) - V_{t+1}(x_*^+) \mid x, a^*\}, \end{aligned}$$

where $\epsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$, a^* is the optimal action and x_*^+ is the next-stage state conditional on the current-stage state x and action a^*. So for a finite horizon problem, its numerical error bound for each $t < T$ is

$$\|\hat{V}(x; b^t) - V_t(x)\| \leq \begin{cases} \frac{\epsilon(1-\beta^{T-t})}{1-\beta}, & \text{when } \beta < 1, \\ (T-t)\epsilon, & \text{when } \beta = 1. \end{cases}$$

Remark 3.1. *For finite horizon problems, if you want to get a value function within ε accuracy, then we should have $\epsilon < (1-\beta)\varepsilon$ when $\beta < 1$, and $\epsilon < \varepsilon/T$ when $\beta = 1$.*

The following is the algorithm of parametric DP with value function iteration for infinite horizon problems:

Algorithm 3.2. *Numerical Dynamic Programming with Value Function Iteration for Infinite Horizon Problems*

Initialization. Choose the approximation grid, $X = \{x_i : 1 \leq i \leq m\}$, and choose functional form for $\hat{V}(x; b)$. Make initial guess $\hat{V}(x; b^0)$ and choose stopping criterion $\varepsilon > 0$. Let $t = 0$.

Step 1. Maximization step. Compute

$$v_i = \max_{a_i \in D(x_i)} u(x_i, a_i) + \beta E\{\hat{V}(x_i^+; b^t) \mid x_i, a_i\},$$

for each $x_i \in X$, $1 \leq i \leq m$.

Step 2. Fitting step. Using the appropriate approximation method, compute the b^{t+1} such that $\hat{V}(x; b^{t+1})$ approximates (x_i, v_i) data.

Step 3. If $\|\hat{V}(x; b^{t+1}) - \hat{V}(x; b^t)\| < \varepsilon$, STOP; else $t \to t+1$ and go to step 1.

Remark 3.2. *For infinite horizon problems, when β is close to 1, if you want to get a value function within ε accuracy, then the stopping criterion should be $\|\hat{V}(x; b^{t+1}) - \hat{V}(x; b^t)\| < (1-\beta)\varepsilon$ and $\epsilon < (1-\beta)^2\varepsilon$ (see Stachurski [51]).*

3.3 Algorithm Architecture

In the maximization step of the DP algorithm architecture, the utility function $u(x, a)$, law of motion from x to x^+, the constraints, and numerical integration of the next-stage value function are defined as basic subroutines so that it is easy to change from one DP problem into another DP problem by changing these basic subroutines.

Since $\hat{V}(x; b)$ are the approximation function with given values at finite nodes, the approximation is only good at a finite range. That is, the state variable must be in a finite range $[x^t_{\min}, x^t_{\max}]$ which may be dependent on time t. Thus, in the constraints, we should have the restriction $x^+_i \in [x^{t+1}_{\min}, x^{t+1}_{\max}]$. For infinite horizon problem, the range should be invariant $[x_{\min}, x_{\max}]$, and for all $x \in [x_{\min}, x_{\max}]$, we should have $x^+ \in [x_{\min}, x_{\max}]$. Thus, if the steady state x^{ss} exists then x^{ss} should be inside of $[x_{\min}, x_{\max}]$.

Another issue is that all the functions should be defined everywhere, as the optimizer may touch the outside of the feasible domain. Otherwise, some optimizers may fail to continue the optimization process. For example, if a function $f(x)$ is defined for positive x, then we could extend it to non-positive region as a linear extrapolation

$$\widehat{f}(x) = \begin{cases} f(x), & \text{if } x \geq \epsilon, \\ f(\epsilon) + f'(\epsilon)(x - \epsilon), & \text{if } x < \epsilon, \end{cases}$$

or a quadratic extrapolation

$$\widehat{f}(x) = \begin{cases} f(x), & \text{if } x \geq \epsilon, \\ f(\epsilon) + f'(\epsilon)(x - \epsilon) + \frac{1}{2} f''(\epsilon)(x - \epsilon)^2, & \text{if } x < \epsilon. \end{cases}$$

If f is a smooth function, the quadratic extrapolation formula makes $\widehat{f} \in C^2$ such that it would be great for optimizers to use non-zero second derivatives (the value of second derivative of f is a denominator in the iteration step of Newton method), while the linear extrapolation formula makes $f''(x) = 0$ for $x < \epsilon$ and $f''(\epsilon)$ even fails to exist. But the linear extrapolation formula has its advantage when $|f''(x)| \to \infty$ as $x \downarrow 0$, such as $f(x) = \log(x)$ or $f(x) = x^\alpha$ with $\alpha < 2$. In such a situation, the quadratic extrapolation may overflow if ϵ is chosen to be very small.

In the maximization step, we need care about the choice of the optimization solver. Since usually each maximization step is an optimization problem with small size (just several unknowns and constraints), the best choice may be NPSOL. If the objective function is non-differentiable, then Nelder-Mead method may be a good choice while there is no constraint. But if there are some constraints, Nelder-Mead method is often not a good alternative. Instead, we could use convex optimization techniques discussed in Grand *et al.* [23] and Mattingley and Boyd [36], if the optimization problem is a convex one.

The stopping criterion for the optimization solver is another issue. If it is too loose, then the converged solution might not approximate the true one enough such that the concavity and monotonicity

properties of the value function may be lost even if we are using a shape-preserving approximation method in the fitting step; but if it is too strict, then it will take too many times and may be unnecessary because the small errors may be recovered in the following value iterations for infinite horizon problems.

A good initial guess of optimal control for each optimization problem in the maximization step is also an important issue. The intuitive choice is to use the optimal values of choice variables in the last iteration of the DP procedure. But in the first iterations, the optimal values of choice variables in the last node may be a better choice, this could be done via "Warm Start" option in the optimization solver.

In the fitting step of the algorithm, various approximation methods are given as separate modules, so the user can switch from one approximation method to another approximation method. In the current version, the following approximation methods are provided: piecewise linear interpolation method, (tensor/complete) Chebyshev polynomial approximation method, Schumaker shape-preserving spline interpolation method, Cubic spline method, and B-spline method. These methods will be described in details in section 4.2.

In the step 3, for an infinite horizon problem, we will calculate the values of $\hat{V}(x; b^{t+1})$ and $\hat{V}(x; b^t)$ at some other state grid nodes $\tilde{X} = \{\tilde{x}_i : 1 \leq i \leq \tilde{m}\}$, then get the approximation of $\|\hat{V}(x; b^{t+1}) - \hat{V}(x; b^t)\|$. Since $\hat{V}(x; b^t)$ may be close to 0 or quite large at some nodes, we use $\max\{|\hat{V}(\tilde{x}_i; b^{t+1}) - \hat{V}(\tilde{x}_i; b^t)|/(1 + |\hat{V}(\tilde{x}_i; b^t)|)\}$ as the adjusted norm $\|\hat{V}(x; b^{t+1}) - \hat{V}(x; b^t)\|$.

The version of DP method with value function iteration for these multidimensional problems with both continuous and discrete states is given in Algorithm 3.3. In the algorithm, we denote d as the dimension for the continuous states x and k as the dimension for discrete states θ, i.e., $x \in \mathbb{R}^d$ and $\theta \in \Theta = \{\theta_j = (\theta_{j1}, \ldots, \theta_{jk}) : 1 \leq j \leq N\}$. The transition probabilities from θ_i to θ_j for $1 \leq i, j \leq N$ are given. For convenience, we also give the form of model,

$$V_0(x, \theta) = \max_{a_t \in D(x_t, \theta_t, t)} \sum_{t=0}^{T} \beta^t E\{u_t(x_t, \theta_t, a_t)\},$$

where $(x_t, \theta_t$ is the state process with initial state $(x_0, \theta_0) = (x, \theta)$. Its corresponding DP model is

$$V_t(x, \theta) = \max_{a \in D(x, \theta, t)} \quad u_t(x, \theta, a) + \beta E\{V_{t+1}(x^+, \theta^+) \mid x, \theta, a\},$$

where (x^+, θ^+) is the next-stage state conditional on the current-stage state (x, θ).

Algorithm 3.3. *Parametric Dynamic Programming with Value Function Iteration for Problems with Multidimensional Continuous and Discrete States*

Initialization. Given a finite set of $\theta \in \Theta = \{\theta_j = (\theta_{j1}, \ldots, \theta_{jk}) : 1 \leq j \leq N\}$ and the transition probabilities from θ_i to θ_j for $1 \leq i, j \leq N$. Choose functional form for $\hat{V}(x, \theta; b)$ with $x \in \mathbb{R}^d$ for all $\theta \in \Theta$, and choose the approximation grid, $X_t = \{x_i^t = (x_{i1}^t, \ldots, x_{id}^t) : 1 \leq i \leq m_t\}$. Make initial guess $\hat{V}(x, \theta; b^0)$ and choose stopping criterion $\varepsilon > 0$ for an infinite horizon problem, or let $\hat{V}(x, \theta; b^0) = V_T(x, \theta) := u_T(x, \theta)$ for a finite horizon problem.

Step 1. Maximization step. Compute

$$v_{ij} = \max_{a_{ij} \in D(x_i, \theta_j, t)} u_t(x_i, \theta_j, a_{ij}) + \beta E\{\hat{V}(x_i^+, \theta_j^+; b^t) \mid x_i, \theta_j, a_{ij}\},$$

for each $x_i \in X_t$ and $\theta_j \in \Theta$, $1 \le i \le m_t$, $1 \le j \le N$.

Step 2. Fitting step. Using the appropriate approximation method, compute the b_j^{t+1}, such that $\hat{V}(x, \theta_j; b_j^{t+1})$ approximates $\{(x_{ij}, v_{ij})\colon 1 \le i \le m_t\}$ data for each $1 \le j \le N$.

Step 3. If $t = T - 1$ for a finite horizon problem, or if $\max\{e_j\} < \varepsilon$ for an infinite horizon problem, where $e_j = \|\hat{V}(x, \theta_j; b_j^{t+1}) - \hat{V}(x, \theta_j; b_j^t)\|$ for $1 \le j \le N$, STOP; else go to step 1.

Remark 3.3. *The conditional expectation operator for the discrete state θ may have the sparsity property. Foe example, some conditional probabilities from θ_j to θ_k may be 0, i.e., $\Pr(\theta_j^+ = \theta_k) = 0$. If so, then there is no need to calculate $\hat{V}(x, \theta_k; b^t)$ when computing $E\{\hat{V}(x_i^+, \theta_j^+; b^t) \mid x_i, \theta_j, a_{ij}\}$.*

Chapter 4

Tools from Numerical Analysis

The previous chapter outlined the basic numerical challenges. In this chapter, we review the tools from numerical analysis that we use to produce stable and efficient algorithms. There are three main components in numerical dynamic programming (DP): optimization, approximation and numerical integration.

4.1 Optimization

For each value function iteration, the most time-consuming part is the optimization step. Moreover, there are m optimization tasks if the number of approximation nodes is m, all these m optimization tasks are required to give their global optimizers. If the number of value function iterations is N, then the total number of optimization tasks is $N \times m$. Thus, an efficient and stable optimization step is very important in numerical DP. Luckily, all these optimization tasks are usually small-size problems with a small number of control variables. Moreover, they are easily parallelized in each value function iteration. More detailed discussion and application about parallelization will be shown in chapter 8.

If value function is only piecewise linear, then the objective function of the optimization problem in the maximization step is not smooth, such that we can only use Nelder-Mead-like methods to solve the optimization problem unless it can be transformed into a linear programming (LP) problem. But a Nelder-Mead-like method is usually very slow, which makes it inefficient in solving optimization problems.

If value function is smooth, we can use Newton's method and related methods for constrained nonlinear optimization problems. Newton-type methods are very credited by the property of a locally quadratic convergence rate to local optimizer. A detailed discussion of these methods can be seen in Gill, Murray and Wright [21, 22], and Murray and Prieto [40]. There are a lot of optimization software packages available to solve these constrained nonlinear optimization problems, such as SNOPT (see Gill, Murray and Saunders [18, 19]), KNITRO (see Byrd *et al.* [7]), and so on. These solvers can be also available for usage in NEOS server website: `http://neos.mcs.anl.gov`, after coding in AMPL or GAMS.

In our Fortran codes designed for various DP problems, we use NPSOL Fortran package (see Gill, Murray, Saunders and Wright [20]), which is also similar to NPOPT, a part of SNOPT package for small dense problems. NPSOL is a set of Fortran subroutines for minimizing a smooth function subject to linear and nonlinear constraints. NPSOL uses the popular sequential quadratic programming (SQP) algorithm in which each search direction is the solution of a quadratic programming (QP) subproblem. Since NPSOL requires relatively few evaluations of the objective or constraint functions, it is especially effective if the problem functions are expensive to evaluate. Hence, NPSOL is the most appropriate optimization solver for our DP application in economics and finance, since the optimization tasks in numerical DP are small-size smooth problems with relatively expensive objective or constraints evaluation due to the "curse of dimensionality" for value function approximation and its numerical integration.

4.2 Approximation Methods

In an approximation scheme, it consists of two aspects: basis functions and approximation nodes. Approximation nodes can be chosen as uniformly spaced nodes, Chebyshev nodes, or some other specified nodes. From the viewpoint of basis functions, approximation methods can be classified as either spectral methods or finite element methods. A spectral method uses globally nonzero basis functions $\phi_j(x)$ such that $\hat{V}(x) = \sum_{j=0}^{m-1} c_j \phi_j(x)$. In this section, we present the examples of spectral methods such as ordinary polynomial approximation method, and (tensor/complete) Chebyshev polynomial approximation method. In contrast, a finite element method uses locally nonzero basis functions $\phi_j(x)$ which are nonzero over sub-domains of the the approximation domain. The examples of finite element methods include piecewise linear interpolation method, Schumaker shape-preserving spline method interpolation method, Cubic spline method, B-spline method.

4.2.1 Piecewise Linear Interpolation

The simplest continuous state value function approximation is the piecewise linear interpolation method. That is, in the fitting step of parametric DP algorithms with value function iteration, we approximate $V(x)$ by

$$\hat{V}(x;\lambda) = v_i + \frac{v_{i+1} - v_i}{x_{i+1} - x_i}(x - x_i) = \lambda_{i0} + \lambda_{i1} x, \qquad \text{if } \; x_i \le x \le x_{i+1},$$

where $\lambda_{i0} = (x_{i+1} v_i - x_i v_{i+1})/(x_{i+1} - x_i)$ and $\lambda_{i1} = (v_{i+1} - v_i)/(x_{i+1} - x_i)$, for $i = 1, \ldots, m$. This interpolation obviously preserves monotonicity and concavity. But it is not differentiable. So the problem with linear interpolation is that it makes the maximization step less efficient. The kinks in a linear interpolant will generally produce kinks in the objective of the maximization step, forcing us to use slower optimization algorithms.

4.2.2 Ordinary Polynomial Approximation

The most naive global approximation method is ordinary polynomial interpolation method. That is, given a Lagrange data points set $\{(x_i, v_i) : i = 1, \ldots, n\}$, we find a polynomial $\hat{V}(x)$ such that $\hat{V}(x_i) = v_i$ for $i = 1, \ldots, n$.

For univariate problems, this function $\hat{V}(x)$ is a degree-$(n-1)$ polynomial. That is, we find the polynomial coefficients c_i such that $\hat{V}(x) = \sum_{i=0}^{n-1} c_i x^i$. These coefficients can be theoretically computed by a linear system $Ac = v$, where $c = (c_0, c_1, \ldots, c_{n-1})^\top$, $v = (v_1, \ldots, v_n)^\top$, and

$$A = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix},$$

which is called the Vandermonde matrix. From the matrix computation theorem, we know there is a unique solution to the linear system $Ac = v$, if the points $x_1, \ldots, x_n$ are distinct.

However, to solve the linear system $Ac = v$, it will have a time cost $O(m^3)$ in general. Moreover, the conditional number of A is generally large. In practice, there is a simple way to find the coefficients c by using the Lagrange interpolation polynomial

$$p(x) = \sum_{i=1}^{n} v_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j},$$

which interpolates the Lagrange data, i.e., $p(x_i) = v_i$ for $i = 1, \ldots, n$. Therefore, c_i is equal to the combination of the coefficients of x^i in $p(x)$.

If this is a multivariate problem or we want to use a smaller degree polynomial to approximate the function, we want to find a polynomial approximation $\hat{V}(x) = \sum_{j=0}^{m-1} c_j \phi_j(x)$ with $m \leq n$, where $\phi_j(x)$ are the basis monomials here ($\phi_0(x) = 1$ and in univariate case $\phi_j(x) = x^j$), such that c is the minimizer of the problem

$$\min \sum_{i=1}^{n} \left(v_i - \sum_{j=0}^{m-1} c_j \phi_j(x) \right)^2.$$

Let

$$\Phi = \begin{bmatrix} 1 & \phi_1(x_1) & \cdots & \phi_{m-1}(x_1) \\ 1 & \phi_1(x_2) & \cdots & \phi_{m-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_n) & \cdots & \phi_{m-1}(x_n) \end{bmatrix}.$$

Then the minimizer is

$$c = (\Phi^\top \Phi)^{-1} \Phi^\top v,$$

which is equivalent to the interpolation when $m = n$ and Φ is nonsingular.

In some cases, we may wish to find a polynomial that fits both function values v_i and its first

derivatives s_i at specified points x_i. That is, we want to find a polynomial $\hat{V}(x) = \sum_{j=0}^{m-1} c_j \phi_j(x)$ such that

$$\begin{aligned} \sum_{j=0}^{m-1} c_j \phi_j(x_i) &= v_i, \quad i = 1, \ldots, n, \\ \sum_{j=0}^{m-1} c_j \nabla \phi_j(x_i) &= s_i, \quad i = 1, \ldots, n. \end{aligned}$$

Since there are $(d+1)n$ conditions (d is the dimension of x_i), we should let $m = (d+1)n$ to satisfy the conditions. This is called as the Hermite interpolation.

4.2.3 Orthogonal Polynomials

In the above Lagrange polynomial approximation method, we need to solve the linear system $c = (\Phi^\top \Phi)^{-1} \Phi^\top v$, which will have a time cost $O(m^3)$ and a large conditional number of $\Phi^\top \Phi$ in general. Thus the polynomial approximation method is not practical.

Moreover, the above Lagrange polynomial interpolation does not always work even on smooth and good-shaped functions. In Judd [28], there is an example $f(x) = (1 + x^2)^{-1}$ over the interval [-5,5] to show the nonconvergence of interpolation, i.e., the degree $n - 1$ interpolation at n uniformly spaced points will be worse as n is larger.

Therefore, if we choose basis functions and approximation nodes such that Φ is an orthogonal matrix, then $\Phi^\top \Phi$ is a diagonal matrix, and it makes the computation of $c = (\Phi^\top \Phi)^{-1} \Phi^\top v$ trivial. This leads to the orthogonal polynomial ideas.

From another point of view, if we think the space of smooth functions is spanned by the monomials x^j, then the polynomial approximation method is to use the monomials as basis functions to construct the approximation of functions. Thus from the basic vector space theory, orthogonal base will be better, i.e., we need to construct the orthogonal polynomials.

At first, in the space of smooth functions, we define the weighted inner product as

$$< f, g >= \int_a^b f(x) g(x) w(x) dx,$$

where $w(x)$ is a weighting function on $[a, b]$ that is positive almost everywhere and has a finite integral on $[a, b]$. Thus, we say that two functions f and g are orthogonal with respect to the weighting function $w(x)$ if $< f, g >= 0$. Moreover, the family of polynomials $\{\phi_j(x)\}$ is called mutually orthogonal polynomials with respect to the weighting function $w(x)$ if $< \phi_i, \phi_j >= 0$ for any $i \neq j$.

Given a weighting function and a family of basis functions, we can use the Gram-Schmidt procedure to generate the orthogonal base. By choosing monomials as the family of basis functions and different weighting function $w(x)$, there are several famous families of orthogonal polynomials such as Legendre polynomials on $[-1, 1]$ with $w(x) = 1$, Chebyshev polynomials on $[-1, 1]$ with $w(x) = \left(1 - x^2\right)^{-1/2}$, Laguerre polynomials on $[0, \infty)$ with $w(x) = e^{-x}$, Hermite polynomials on $(-\infty, \infty)$ with $w(x) = e^{-x^2}$, and so on. For more detailed discussion, see Kenneth Judd [28].

4.2.4 Chebyshev Polynomial Approximation

In section 4.2.2, we approximate functions in the L^2 form to minimize $\sum_{i=1}^{n}(v_i - \hat{V}(x_i))^2$ or minimize $\|V - \hat{V}\|_2$ where $\hat{V}(x) = \sum_{j=0}^{m-1} c_j \phi_j(x)$. But the convergence in L^2 can not guarantee that the approximation is close to the function everywhere, i.e., the approximation could be far away from the function at some individual points while the L^2 error is small.

L^∞ Approximation

We want to find a sequence of polynomials, $\{\hat{V}_n(x)\}$, such that

$$\lim_{n\to\infty} \|V - \hat{V}_n\|_\infty = 0,$$

where $\|V - \hat{V}_n\|_\infty$ is the L^∞ norm, i.e., $\|V - \hat{V}_n\|_\infty = \sup_{x\in[a,b]} |V(x) - \hat{V}_n(x)|$. $\hat{V}_n$ is called as the L^∞ approximation or uniform approximation of V. The Weierstrass Theorem provides strong motivation to approximate continuous functions by using a sequence of polynomials.

Theorem 4.1. *(Weierstrass Theorem) If the function $f \in C^k[a,b]$, then there exists a sequence of polynomials, p_n, where the degree of p_n is n, such that*

$$\lim_{n\to\infty} \|f^{(l)} - p_n^{(l)}\|_\infty = 0,$$

for any $0 \le l \le k$, where $f^{(l)}$ and $p_n^{(l)}$ means the l-th derivative of f and p_n.

One example of such a sequence of polynomials is the Bernstein polynomials on $[0,1]$,

$$p_n(x) = \sum_{i=0}^{n} \binom{n}{i} f\left(\frac{i}{n}\right) x^i (1-x)^{n-i}.$$

But the Weierstrass Theorem does not tell us how to find a good polynomial approximation with the lowest degree to achieve a required level of accuracy. The following theorem tells us that among all degree-n polynomials there is a polynomial to best approximate the function uniformly.

Theorem 4.2. *(Equioscillation Theorem) If $f \in C[a,b]$, then there is a unique polynomial of degree n, $q_n^*(x)$, such that*

$$\|f - q_n^*\|_\infty = \inf_{\deg(q)\le n} \|f - q\|_\infty.$$

Moreover, for this q_n^*, there are at least $n+2$ points $a \le x_0 < x_1 < \cdots < x_{n+1} \le b$ such that

$$f(x_j) - q_n^*(x_j) = m(-1)^j \inf_{\deg(q)\le n} \|f - q\|_\infty,$$

for $j = 0, \ldots, n+1$, where $m = 1$ or $m = -1$.

The equioscillation theorem tells us the existence of $q_n^*(x)$, but not how to find it. In practice, it can be hard to find it. In the next subsection, we present the Chebyshev least square approximation method that also does very well in the uniform approximation.

Chebyshev Polynomial

Chebyshev polynomials on $[-1,1]$ are defined as $T_j(x) = \cos(j\cos^{-1}(x))$, while general Chebyshev polynomials on $[a,b]$ are defined as $T_j((2x-a-b)/(b-a))$ for $j = 0,1,2,\ldots$. These polynomials are orthogonal under the weighted inner product: $< f,g >= \int_a^b f(x)g(x)w(x)dx$ with the weighting function $w(x) = \left(1-\left(\frac{2x-a-b}{b-a}\right)^2\right)^{-1/2}$. The polynomials $T_j(x)$ on $[-1,1]$ can be recursively evaluated:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{j+1}(x) &= 2xT_j(x) - T_{j-1}(x), \quad j = 1,2,\ldots. \end{aligned}$$

Chebyshev Least Square Approximation

Using the above orthogonal polynomials, we have the least square polynomial approximation of V with respect to the weighting function

$$w(x) = \left(1-\left(\frac{2x-a-b}{b-a}\right)^2\right)^{-1/2},$$

i.e., a degree-n polynomial $\hat{V}_n(x)$, such that $\hat{V}_n(x)$ solves

$$\min_{\deg(\hat{V})\le n} \int_a^b (V(x)-\hat{V}_n(x))^2 w(x)dx.$$

Thus, we know that the least square degree-n polynomial approximation $\hat{V}_n(x)$ on $[-1,1]$ has the form

$$\hat{V}_n(x) = \frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j(x),$$

where

$$c_j = \frac{2}{\pi}\int_{-1}^{1} \frac{V(x)T_j(x)}{\sqrt{1-x^2}}dx, \quad j = 0,1,\ldots,n,$$

are the Chebyshev least square coefficients. It is difficult to compute the coefficients because the above integral generally does not have an analytic solutions, even if we know the explicit form of V.

But if we know the values of V at some specific nodes, then we can approximate V by interpolation method. That is, we find a function $\hat{V}$ such that $\hat{V}(x_i) = V(x_i)$ at the given nodes x_i, $i = 1,\ldots,m$. One simple scheme is ordinary polynomial interpolation. But this does not always work even on well-behaved, simple functions. Practically, Chebyshev interpolation $\hat{V}(x) = c_0/2 + \sum_{j=1}^{m-1} c_j T_j(x)$ will work well for smooth functions on $[-1,1]$. And by the minimax property of Chebyshev polynomials,

we could choose the Chebyshev interpolation nodes on $[-1,1]$: $x_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right)$, $i = 1,\ldots,m$, so that the interpolation error bound is minimized.

By the above coefficient formula of the least square degree-n polynomial approximation, we have

$$\begin{aligned} c_j &= \frac{2}{\pi}\int_{-1}^{1}\frac{V(x)T_j(x)}{\sqrt{1-x^2}}dx \\ &= \frac{2}{\pi}\int_{0}^{\pi} V(\cos(\theta))T_j(\cos(\theta))d\theta \\ &\approx \frac{2}{\pi}\frac{\pi}{m}\sum_{i=1}^{m} V\left(\cos\left(\frac{(2i-1)\pi}{2m}\right)\right)T_j\left(\cos\left(\frac{(2i-1)\pi}{2m}\right)\right) \\ &= \frac{2}{m}\sum_{i=1}^{m} V(z_i)T_j(z_i) \\ &= \frac{2}{m}\sum_{i=1}^{m} v_i T_j(z_i), \end{aligned}$$

for $j = 0,1,\ldots,n$.

Chebyshev Regression Algorithm

Numerically, by adapting Chebyshev least squares approximation and Chebyshev interpolation ideas, we can have a general Chebyshev regression algorithm. That is, we compute the degree n truncation of the degree $m-1$ interpolation formula by dropping the high-degree terms (degree $n+1$ through $m-1$ terms).

Algorithm 4.1. *Chebyshev Regression Algorithm*

Objective: Given $V(x)$ on $[a,b]$, choose m nodes to construct a degree $n < m$ Chebyshev polynomial approximation $\hat{V}_n(x)$ for $V(x)$.

Step 1. Choose the m Chebyshev nodes on $[-1,1]$:

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right), \quad i = 1,\ldots,m.$$

Step 2. Adjust the nodes to $[a,b]$:

$$x_i = (z_i+1)(b-a)/2 + a, \quad i = 1,\ldots,m.$$

Step 3. Evaluate $v_i = V(x_i)$ for $i = 1,\ldots,m$.

Step 4. Compute Chebyshev coefficients:

$$c_j = \frac{2}{m}\sum_{i=1}^{m} v_i T_j(z_i), \qquad j = 0,\ldots,n.$$

Then the Chebyshev approximation for $V(x)$, $x \in [a, b]$ is

$$\hat{V}_n(x) = \frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j \left(\frac{2x - a - b}{b - a} \right).$$

We can verify that if $n = m - 1$, then

$$\hat{V}_n(x_i) = \frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j (z_i) = v_i.$$

The reader can refer to Judd [28] for detailed discussion.

In fact, by choosing the Chebyshev polynomials as basis functions and the Chebyshev nodes as approximation nodes, we have the property

$$\sum_{i=1}^{m} T_j(z_i)T_k(z_i) = \begin{cases} 0, & \text{if } i \neq j, \\ m, & \text{if } i = j = 0, \\ m/2, & \text{if } 0 < i = j \leq n. \end{cases}$$

This property makes the matrix Φ orthogonal in the Lagrange approximation method, such that we can have the simple formula in the above step 4 to compute the Chebyshev coefficients.

Expanded Chebyshev Polynomials

For Chebyshev polynomial approximation method, it has poor approximation on the neighbor of end points, so that the Maximization step might not get a good solution. To solve this problem, we use the expanded Chebyshev polynomials. That is, we use the Chebyshev polynomial approximation over $[a - \delta_1, b + \delta_2]$ instead of $[a, b]$, such that $x_i = (z_i + 1)((b + \delta_2) - (a - \delta_1))/2 + (a - \delta_1)$ are the adjusted m Chebyshev nodes on $[a - \delta_1, b + \delta_2]$ with $x_1 = a$ and $x_m = b$, where $z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right)$ are the Chebyshev nodes on $[-1, 1]$. Solving the linear system, we have $\delta_1 = \delta_2 = \frac{z_1+1}{-2z_1}(b - a)$. In fact, when $a = -1$ and $b = 1$, $x_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right)\sec(\frac{\pi}{2m})$, which are called the expanded Chebyshev array (see Rivlin [45]).

Another benefit of expanded Chebyshev approximation method is that we do not need to worry that the approximation function may be infeasible. The infeasibility may happen in the ordinary Chebyshev approximation method: if we use $T_j(x) = \cos(j \cos^{-1}(x))$ directly in the numerical computation instead of the recursion formula (this happens in AMPL codes), then numerically x might be chosen in $[-1 - \epsilon, -1)$ or $(1, 1 + \epsilon]$ within the tolerance ϵ even if there is a constraint $x \in [-1, 1]$, but this make $\cos^{-1}(x)$ infeasible.

The following is the expanded Chebyshev polynomial approximation method:

Algorithm 4.2. *Expanded Chebyshev Regression Algorithm*

Objective: Given $V(x)$ on $[a, b]$, choose m nodes to construct a degree $n < m$ expanded Chebyshev polynomial approximation $\hat{V}_n(x)$ for $V(x)$.

Step 1. Choose the m Chebyshev nodes on $[-1,1]$:

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right), \quad i = 1,\ldots,m.$$

Step 2. Adjust the nodes to $[a,b]$:

$$x_i = (z_i+1)(b-a+2\delta)/2 + a - \delta, \quad i = 1,\ldots,m.$$

where $\delta = \frac{z_1+1}{-2z_1}(b-a)$.

Step 3. Evaluate $v_i = V(x_i)$ for $i = 1,\ldots,m$.

Step 4. Compute Chebyshev coefficients:

$$c_j = \frac{2}{m}\sum_{i=1}^{m} v_i T_j(z_i), \quad j = 0,\ldots,n.$$

Then the Chebyshev approximation for $V(x)$, $x \in [a,b]$ is

$$\hat{V}_n(x) = \frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j\left(\frac{2x-a-b}{b-a+2\delta}\right).$$

We can verify that if $n = m-1$, then

$$\hat{V}_n(x_i) = \frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j(z_i) = v_i.$$

Multidimensional Tensor Chebyshev Approximation

In a d-dimensional approximation problem, let $a = (a_1,\ldots,a_d)$ and $b = (b_1,\ldots,b_d)$ with $b_i > a_i$ for $i = 1,\ldots,d$. Let $x = (x_1,\ldots,x_d)$ with $x_i \in [a_i,b_i]$ for $i = 1,\ldots,d$. For simplicity, we denote this as $x \in [a,b]$. Let $\alpha = (\alpha_1,\ldots,\alpha_d)$ be a vector of nonnegative integers. Let $T_\alpha(z)$ denote the tensor product $T_{\alpha_1}(z_1)\cdots T_{\alpha_d}(z_d)$ for $z = (z_1,\ldots,z_d) \in [-1,1]^d$. Let $(2x-a-b)./(b-a)$ denote the vector $(\frac{2x_1-a_1-b_1}{b_1-a_1},\ldots,\frac{2x_d-a_d-b_d}{b_d-a_d})$. Then the degree n tensor Chebyshev approximation for $V(x)$ is

$$\hat{V}_n(x) = \sum_{0\le\alpha_i\le n, 1\le i\le d} c_\alpha T_\alpha\left((2x-a-b)./(b-a)\right).$$

Let us denote the d-dimensional Chebyshev interpolation nodes as $z^{(k)} = (z_1^{(k_1)},\ldots,z_d^{(k_d)}) \in [-1,1]^d$ where $z_i^{(k_i)} = -\cos\left(\frac{(2k_i-1)\pi}{2m}\right)$ for $k_i = 1,\ldots,m$, and $i = 1,\ldots,d$. Let $x^{(k)} = (x_1^{(k_1)},\ldots,x_d^{(k_d)}) = (z^{(k)}+1).*(b-a)/2 + a$, i.e., $x_i^{(k_i)} = (z_i^{(k_i)}+1)(b_i-a_i)/2 + a_i$, $i = 1,\ldots,d$. Let $v_{(k)} = V(x^{(k)})$ be

given. Then the coefficients of the degree n tensor Chebyshev approximation are evaluated as

$$c_\alpha = \frac{2^{\tilde{d}}}{m^d} \sum_{1\le k_i\le m, 1\le i\le d} v_{(k)} T_\alpha(z^{(k)}),$$

where $\tilde{d} = \sum_{i=1}^d 1_{\alpha_i>0}$ with $1_{\alpha_i>0}$ as the indicator

$$1_{\alpha_i>0} = \begin{cases} 1, & \text{if } \alpha_i > 0, \\ 0, & \text{if } \alpha_i = 0, \end{cases}$$

for all nonnegative integer vectors α with $0 \le \alpha_i \le n$.

Multidimensional Complete Chebyshev Approximation

Using the previous notations, the degree n complete Chebyshev approximation for $V(x)$ is

$$\hat{V}_n(x) = \sum_{0\le|\alpha|\le n, 1\le i\le d} c_\alpha T_\alpha\left((2x-a-b)./(b-a)\right),$$

where $|\alpha|$ denotes $\sum_{i=1}^d \alpha_i$ for the nonnegative integer vector $\alpha = (\alpha_1, \ldots, \alpha_d)$. The coefficients of the degree n complete Chebyshev approximation are the same with the coefficients of the degree n tensor Chebyshev approximation for the terms $|\alpha| \le n$. We know the number of terms with $0 \le |\alpha| = \sum_{i=1}^d \alpha_i \le n$ is $\binom{n+d}{d}$. So the complexity of computation of a degree n complete Chebyshev polynomial is about $\binom{n+d}{d}/(n+1)^d \approx 1/d!$ of complexity of computation of a degree n tensor Chebyshev polynomial, while the precision of approximation is almost the same for $n \ge 6$ usually in practice.

4.2.5 Cubic Splines

Since cubic splines are C^2, many people prefer to apply cubic splines for approximation of smooth functions. In Judd [28], there is a detailed discussion for it.

Suppose that the Lagrange data set is (x_i, v_i), we want to construct a spline, $s(x)$, such that $s(x_i) = v_i$, $i = 0, \ldots, n$. On each interval $[x_i, x_{i+1}]$, $s(x)$ will be a cubic polynomial: $a_i + b_i x + c_i x^2 + d_i x^3$. The interpolating conditions plus C^2 continuity at the interior nodes implies $4n-2$ conditions:

$$\begin{aligned}
&v_i = a_i + b_i x_i + c_i x_i^2 + d_i x_i^3, \quad i = 1, \ldots, n, \\
&v_i = a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3, \quad i = 0, \ldots, n-1, \\
&b_i + 2c_i x_i + 3d_i x_i^2 = b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2, \quad i = 1, \ldots, n-1, \\
&2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i, \quad i = 1, \ldots, n-1,
\end{aligned}$$

while there are $4n$ unknowns. This leaves us two conditions short of fixing the unknown coefficients. Various cubic splines are differentiated by the two additional conditions imposed. For example, the natural spline imposes $s''(x_0) = 0 = s''(x_n)$; the Hermite spline imposes $s'(x_0) = s_0$ and $s'(x_n) = s_n$

for some given s_0 and s_n as the estimated slopes at the end points; the secant Hermite spline imposes $s'(x_0) = (v_1 - v_0)/(x_1 - x_0)$ and $s'(x_n) = (v_n - v_{n-1})/(x_n - x_{n-1})$; the not-a-knot spline imposes that $d_0 = d_1$ and $d_{n-1} = d_n$, i.e., the third derivative of the spline is continuous at the second and next-to-last breakpoint. These two additional conditions have much more effects in the maximization step than the approximation step.

4.2.6 B-Splines

Suppose that we want to construct order k splines (C^{k-2} function) on $[a, b]$, and we have a grid of knots at $x_{-k} < \cdots < x_{-1} < x_0 < \cdots < x_{n+k}$, where $x_0 = a$ and $x_n = b$. The B-splines form a basis for splines. The B^k-spline can be defined by the recursive relation:

$$B_i^0(x) = \begin{cases} 1, & \text{if } x_i \le x < x_{i+1}, \\ 0, & \text{if } x < x_i \text{ or } x \ge x_{i+1}, \end{cases} \qquad i = -k, \ldots, n+k-1,$$

$$B_i^1(x) = \begin{cases} \frac{x - x_i}{x_{i+1} - x_i}, & \text{if } x_i \le x < x_{i+1}, \\ \frac{x_{i+2} - x}{x_{i+2} - x_{i+1}}, & \text{if } x_{i+1} \le x < x_{i+2}, \\ 0, & \text{if } x < x_i \text{ or } x \ge x_{i+2}, \end{cases} \qquad i = -k, \ldots, n+k-2,$$

$$B_i^k(x) = \left(\frac{x - x_i}{x_{i+k} - x_i} \right) B_i^{k-1}(x) + \left(\frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} \right) B_{i+1}^{k-1}(x), \quad i = -k, \ldots, n-1.$$

Using these B-splines basis, we can approximate a function $V(x)$ by an order k splines $\hat{V}_k(x)$:

$$\hat{V}_k(x) = \sum_{j=1}^{k} c_{ij} B_i^j(x), \qquad \text{for } x_i \le x < x_{i+1},\ i = 0, \ldots, n-1,$$

such that $\hat{V}_k(x_i) = V(x_i)$ for $i = 0, \ldots, n-1$, and $\hat{V}_k(x) \in C^{k-2}[a, b]$.

Multidimensional B-Splines

In a d-dimensional approximation problem, let $m = (m_1, \ldots, m_d)$ and $\alpha = (\alpha_1, \ldots, \alpha_d)$ be vectors of nonnegative integers. Let $B_m^{(\alpha)}(x)$ denote the tensor product $B_{m_1}^{(\alpha_1)}(x_1) \cdots B_{m_d}^{(\alpha_d)}(x_d)$ for $x = (x_1, \ldots, x_d)$ in $[a, b] \subset \mathbb{R}^d$.

Denote the d-dimensional interpolation nodes as $x^{(m)} = (x_1^{(m_1)}, \ldots, x_d^{(m_d)}) \in [a, b]$ for $m_j = 0, \ldots, n$ and $j = 1, \ldots, d$. Denote $m + 1$ as $(m_1 + 1, \ldots, m_d + 1)$. Then the order k tensor B-spline approximation for $V(x)$ is

$$\hat{V}_k(x) = \sum_{1 \le \alpha_j \le k, 1 \le j \le d} c_{m,\alpha} B_m^{\alpha}(x), \quad \text{if } x^{(m)} \le x < x^{(m+1)},$$

for $m_j = 0, \ldots, n-1$, $j = 1, \ldots, d$, such that $\hat{V}_k(x^{(m)}) = V(x^{(m)})$ for all m, and $\hat{V}_k(x) \in C^{k-2}[a, b]$.

4.3 Numerical Integration

In the objective function of the Bellman equation, we often need to compute the conditional expectation of $V(x^+ \mid x, a)$. When the random variable is continuous, we have to use numerical integration method to compute the expectation.

One naive way is to apply Monte Carlo or pseudo Monte Carlo methods to compute the expectation. By the center limit theorem in statistics, the numerical error of the integration computed by (pseudo) Monte Carlo methods has a distribution which is close to normal distribution. So there is no bound for the numerical error occurred by (pseudo) Monte Carlo methods. Moreover in the optimization problem, it often needs hundreds or thousands of evaluations of the objective functions. These imply that once one evaluation of the objective function has a big numerical error, the previous iterations to solve the optimization problem may make no sense. Therefore, the iterations may never converge to the optimal solution. So it is not practical to apply (pseudo) Monte Carlo methods in the optimization problem generally, unless the stopping criterion of the optimization problem is set very loosely.

Therefore, it will be good to have a numerical integration method with a bounded numerical error. Here we present several common numerical integration methods.

4.3.1 Gauss-Hermite Quadrature

In the expectation operator of the objective function of the Bellman equation, if the random variable has a normal distribution, then it will be good to apply the Gauss-Hermite quadrature formula to compute the numerical integration. That is, if we want to compute $E(f(Y))$ where Y has a distribution $N(\mu, \sigma^2)$, then

$$\begin{aligned} E(f(Y)) &= (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{\infty} f(y) e^{-(y-\mu)^2/(2\sigma^2)} dy \\ &= (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{\infty} f(\sqrt{2}\,\sigma\, x + \mu) e^{-x^2} \sqrt{2}\sigma dx \\ &\doteq \pi^{-\frac{1}{2}} \sum_{i=1}^{n} \omega_i f(\sqrt{2}\sigma x_i + \mu), \end{aligned}$$

where the ω_i and x_i are the Gauss-Hermite quadrature weights and nodes over $(-\infty, \infty)$. We list some of ω_i and x_i in Table 4.1.

If Y is log normal, i.e., $\log(Y)$ has a distribution $N(\mu, \sigma^2)$, then we can assume that $Y = e^X$ where $X \sim N(\mu, \sigma^2)$, thus

$$E(f(Y)) = E(f(e^X)) \doteq \pi^{-\frac{1}{2}} \sum_{i=1}^{n} \omega_i f\left(e^{\sqrt{2}\sigma x_i + \mu}\right).$$

In portfolio problems, asset random returns are often assumed to be log-normal, such that the Gauss-Hermite quadrature rule can be applied to compute expectations. For example, suppose that an investor plans to invest 1 dollar, and there are two assets available for investment: one bond with a

Table 4.1: Gauss-Hermite quadrature

N	x_i	ω_i	N	x_i	ω_i
2	0.7071067811	0.8862269254	7	0.2651961356(1)	0.9717812450(−3)
3	0.1224744871(1)	0.2954089751		0.1673551628(1)	0.5451558281(−1)
	0.0000000000	0.1181635900(1)		0.8162878828	0.4256072526
				0.0000000000	0.8102646175
4	0.1650680123(1)	0.8131283544(-1)			
	0.5246476232	0.8049140900	10	0.3436159118(1)	0.7640432855(−5)
				0.2532731674(1)	0.1343645746(−2)
5	0.2020182870(1)	0.1995324205(-1)		0.1756683649(1)	0.3387439445(−1)
	0.9585724646	0.3936193231		0.1036610829(1)	0.2401386110
	0.0000000000	0.9453087204		0.3429013272	0.6108626337

Note: $a(k)$ means $a \times 10^k$. A (x, ω) entry for N means that $\pm x$ are quadrature nodes in the N-point formula, and each gets weight ω. Source: Stroud and Secrest [52].

Table 4.2: Errors in computing $U(0)$ of (4.1) with Gauss-Hermite quadrature

γ	0.5	1.1	2.0	3.0	4.0	5.0	10.0
$U(0)$	2.0404	−9.9601	−0.96079	−0.46156	−0.29564	−0.21304	−0.07752
$n = 3$	1(−10)	5(−10)	5(−11)	2(−11)	2(−11)	1(−11)	4(−12)
$n = 5$	1(−9)	5(−9)	5(−10)	2(−10)	2(−10)	1(−10)	4(−11)
$n = 7$	1(−9)	6(−9)	6(−10)	3(−10)	2(−10)	1(−10)	5(−11)
$n = 9$	1(−9)	5(−9)	5(−10)	2(−10)	2(−10)	1(−10)	4(−11)
$n = 11$	2(−9)	1(−8)	1(−9)	5(−10)	3(−10)	2(−10)	8(−11)

Note: $a(k)$ means $a \times 10^k$.

riskless annual return $R_f = e^r$ with an interest rate $r = 0.04$, and one stock with a random log-normal annual return R with $\log(R) \sim N(\mu, \sigma^2)$. If he invests x in the stock, and the remaining $(1 - x)$ in the bond, then his wealth after one year becomes $W = xR + (1 - x)R_f$. If the investor consumes all the wealth W and his utility is $u(W) = W^{1-\gamma}/(1 - \gamma)$, then his expected utility is

$$U(x) = (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{+\infty} u(xe^z + (1 - x)R_f)e^{-(z-\mu)^2/(2\sigma^2)}dz. \tag{4.1}$$

When the investor invests all of his money into the bond, i.e., $x = 0$, we have $U(0) = u(R_f) = R_f^{1-\gamma}/(1 - \gamma)$. When the investor invests all of his money into the stock, i.e., $x = 1$, we have $U(1) = E[u(R)] = \exp((1-\gamma)\mu + (1-\gamma)^2\sigma^2/2)/(1-\gamma)$. Table 4.2 and 4.3 display the absolute errors of the Gauss-Hermite quadrature rules applied to (4.1) for various values of n (number of Gauss-Hermite quadrature nodes) and γ when $x = 0$ and $x = 1$ respectively. Here we choose $\mu = 0.07$ and $\sigma = 0.2$.

In comparison with Monte Carlo integration methods, Table 4.4 provides the absolute errors of Monte Carlo integration methods applied to (4.1) for various values of n (number of Monte Carlo simulations) and γ when $x = 1$, $\mu = 0.07$, and $\sigma = 0.2$.

Table 4.3: Errors in computing $U(1)$ of (4.1) with Gauss-Hermite quadrature

γ	0.5	1.1	2.0	3.0	4.0	5.0	10.0
$U(1)$	2.0816	−9.9322	−0.95123	−0.47088	−0.32348	−0.2602	−0.29903
$n = 3$	2(−8)	5(−10)	5(−7)	2(−5)	1(−4)	5(−4)	4(−2)
$n = 5$	1(−9)	5(−9)	5(−10)	2(−9)	6(−8)	8(−7)	2(−3)
$n = 7$	1(−9)	6(−9)	5(−10)	2(−10)	2(−11)	8(−10)	3(−5)
$n = 9$	1(−9)	5(−9)	6(−10)	4(−10)	4(−10)	5(−10)	3(−7)
$n = 11$	2(−9)	1(−8)	1(−9)	4(−10)	1(−10)	1(−9)	1(−7)

Note: $a(k)$ means $a \times 10^k$.

Table 4.4: Errors in computing $U(1)$ of (4.1) with Monte Carlo method

γ	0.5	1.1	2.0	3.0	4.0	5.0	10.0
$U(1)$	2.0816	−9.9322	−0.95123	−0.47088	−0.32348	−0.2602	−0.29903
$n = 1(3)$	9(−3)	9(−3)	2(−3)	3(−3)	2(−3)	3(−3)	1(−2)
$n = 1(4)$	9(−4)	1(−4)	2(−3)	7(−5)	5(−3)	9(−4)	1(−2)
$n = 1(5)$	6(−4)	3(−4)	2(−4)	6(−4)	4(−4)	1(−4)	1(−3)
$n = 1(6)$	2(−4)	3(−6)	2(−4)	2(−4)	3(−4)	2(−4)	9(−4)

Note: $a(k)$ means $a \times 10^k$.

In next example, we assume that the riskless annual return of the bond is $R_f = 1 + r$ with an interest rate r, and the stock has a random normal annual return $R \sim N(1 + \mu, \sigma^2)$. Assume that the investor has an initial wealth W_0, and he invests xW_0 in the stock and the remaining $(1 - x)W_0$ in the bond. If we assume that the investor consumes all the wealth $W_1 = W_0(xR + (1 - x)R_f)$ at the end of one year with an exponential utility $u(W) = -e^{-\lambda W}$ for a constant absolute risk aversion coefficient $\lambda > 0$, then his expected utility is

$$\begin{aligned} U(x) &= (2\pi\sigma^2)^{-1/2} \int_{-\infty}^{+\infty} -e^{-\lambda W_0(1+xz+(1-x)r)} e^{-(z-\mu)^2/(2\sigma^2)} dz \\ &= -\exp(-\lambda W_0(1 + r + x(\mu - r)) + (\lambda x W_0 \sigma)^2/2). \end{aligned} \tag{4.2}$$

Table 4.5 and 4.6 display the relative errors of Gauss-Hermite quadrature rules and Monte Carlo methods, respectively, for various values of n (number of Gauss-Hermite quadrature nodes) and λ when $x = 0.5$, $r = 0.04$, $\mu = 0.07$, and $\sigma = 0.2$.

Table 4.2, 4.3, 4.4, 4.5 and 4.6 tell us that Gauss-Hermite quadrature rules work very well with high accuracy and small numbers of function evaluation, while Monte Carlo methods have much lower accuracy even with very large numbers of function evaluation.

Table 4.5: Errors in computing $U(0.5)$ of (4.2) with Gauss-Hermite quadrature

λ	0.1	0.5	1.0	2.0	5.0	10.0
$U(0.5)$	−0.89992	−0.59082	−0.34994	−0.12369	−0.0057994	−4.3186(−5)
$n = 3$	5(−11)	8(−11)	8(−9)	5(−7)	1(−4)	6(−3)
$n = 5$	5(−10)	5(−10)	5(−10)	5(−10)	3(−8)	3(−5)
$n = 7$	6(−10)	6(−10)	6(−10)	6(−10)	2(−10)	5(−8)
$n = 9$	5(−10)	5(−10)	5(−10)	6(−10)	1(−9)	3(−9)
$n = 11$	1(−9)	1(−9)	1(−9)	1(−9)	4(−10)	2(−8)

Note: $a(k)$ means $a \times 10^k$.

Table 4.6: Errors in computing $U(0.5)$ of (4.2) with Monte Carlo method

λ	0.1	0.5	1.0	2.0	5.0	10.0
$U(0.5)$	-0.89992	-0.59082	-0.34994	-0.12369	-0.0057994	$-4.3186(-5)$
$n = 1(3)$	$3(-5)$	$1(-3)$	$5(-4)$	$4(-3)$	$3(-3)$	$3(-2)$
$n = 1(4)$	$9(-5)$	$4(-4)$	$1(-3)$	$8(-4)$	$5(-3)$	$1(-2)$
$n = 1(5)$	$5(-6)$	$9(-5)$	$2(-5)$	$1(-4)$	$4(-5)$	$4(-3)$
$n = 1(6)$	$5(-6)$	$5(-6)$	$2(-5)$	$3(-4)$	$1(-3)$	$2(-3)$

Note: $a(k)$ means $a \times 10^k$.

Table 4.7: Gauss-Legendre quadrature

N	x_i	ω_i	N	x_i	ω_i
2	0.5773502691	0.1000000000(1)	7	0.9491079123	0.1294849661
				0.7415311855	0.2797053914
3	0.7745966692	0.5555555555		0.4058451513	0.3818300505
	0.0000000000	0.8888888888		0.0000000000	0.4179591836
4	0.8611363115	0.3478548451	10	0.9739065285	0.6667134430(−1)
	0.3399810435	0.6521451548		0.8650633666	0.1494513491
				0.6794095682	0.2190863625
5	0.9061798459	0.2369268850		0.4333953941	0.2692667193
	0.5384693101	0.4786286704		0.1488743389	0.2955242247
	0.0000000000	0.5688888888			

Note: $a(k)$ means $a \times 10^k$. A (x, ω) entry for N means that $\pm x$ are quadrature nodes in the N-point formula, and each gets weight ω. Source: Stroud and Secrest [52].

4.3.2 Gauss-Legendre Quadrature

In the expectation operator of the objective function of the Bellman equation, if the random variable has a uniform distribution, then it will be good to apply the Gauss-Legendre quadrature formula to compute the numerical integration. That is, if we want to compute $E(f(Y))$ where Y has a uniform distribution over $[a, b]$, then

$$E(f(Y)) = \int_a^b f(y)\frac{1}{b-a}\,dy \doteq \frac{1}{2}\sum_{i=1}^{n}\omega_i f\left(\frac{(x_i+1)(b-a)}{2} + a\right),$$

where the ω_i and x_i are the Gauss-Legendre quadrature weights and nodes over $[-1, 1]$. We list some of ω_i and x_i in Table 4.7.

4.3.3 Gauss-Laguerre Quadrature

In the expectation operator of the objective function of the Bellman equation, if the random variable has an exponential distribution, then it will be good to apply the Gauss-Laguerre quadrature formula to compute the numerical integration. That is, if we want to compute $E(f(Y))$ where Y has a exponential distribution over $[0, \infty)$ with a rate λ, then

$$E(f(Y)) = \int_0^\infty f(y)\lambda e^{-\lambda y}dy \doteq \sum_{i=1}^{n}\omega_i f(x_i/\lambda),$$

Table 4.8: Gauss-Laguerre quadrature

N	x_i	ω_i	N	x_i	ω_i
2	0.5857864376	0.8535533905	7	0.1930436765	0.4093189517
	0.3414213562(1)	0.1464466094		0.1026664895(1)	0.4218312778
				0.2567876744(1)	0.1471263486
3	0.4157745567	0.7110930099		0.4900353084(1)	0.2063351446(−1)
	0.2294280360(1)	0.2785177335		0.8182153444(1)	0.1074010143(−2)
	0.6289945082(1)	0.1038925650(−1)		0.1273418029(2)	0.1586546434(−4)
				0.1939572786(2)	0.3170315478(−7)
4	0.3225476896	0.6031541043			
	0.1745761101(1)	0.3574186924	10	0.1377934705	0.3084411157
	0.4536620296(1)	0.3888790851(−1)		0.7294545495	0.4011199291
	0.9395070912(1)	0.5392947055(−3)		0.1808342901(1)	0.2180682876
				0.3401433697(1)	0.6208745609(−1)
5	0.2635603197	0.5217556105		0.5552496140(1)	0.9501516975(−2)
	0.1413403059(1)	0.3986668110		0.8330152746(1)	0.7530083885(−3)
	0.3596425771(1)	0.7594244968(−1)		0.1184378583(2)	0.2825923349(−4)
	0.7085810005(1)	0.3611758679(−2)		0.1627925783(2)	0.4249313984(−6)
	0.1264080084(2)	0.2336997238(−4)		0.2199658581(2)	0.1839564823(−8)
				0.2992069701(2)	0.9911827219(−12)

Note: $a(k)$ means $a \times 10^k$. A (x, ω) entry for N means that $\pm x$ are quadrature nodes in the N-point formula, and each gets weight ω. Source: Stroud and Secrest [52].

where the ω_i and x_i are the Gauss-Laguerre quadrature weights and nodes over $[0, \infty)$. We list some of ω_i and x_i in Table 4.8.

4.3.4 General Quadrature Formula

In the expectation operator of the objective function of the Bellman equation, if the random variable has a probability distribution which is not one of normal distribution, log normal distribution, uniform distribution, or exponential distribution, then there are some other ways to give a good quadrature formula.

The first way is to give a specific formula for such a specific distribution, like Gauss-Hermite formula for normal distribution, Gauss-Legendre formula for uniform distribution, and Gauss-Laguerre formula for exponential distribution.

The second way is to try to transform this specific distribution into one kind of distribution among normal distribution, uniform distribution and exponential distribution, like that log normal distribution can be transformed into a normal distribution.

The third way is the following method:

If the domain of the random variable is finite, then we can use the Gauss-Legendre formula. That

is, if the random variable Y has a density function $p(y)$ over $[a, b]$, then

$$\begin{aligned} E[f(Y)] &= \int_a^b f(y)p(y)dy \\ &\doteq \frac{1}{2}\sum_{i=1}^{n} \omega_i f\left(\frac{(x_i+1)(b-a)}{2}+a\right) p\left(\frac{(x_i+1)(b-a)}{2}+a\right), \end{aligned}$$

where the ω_i and x_i are the Gauss-Legendre quadrature weights and nodes over $[-1, 1]$.

If the domain of the random variable is $(-\infty, +\infty)$, then we can use the Gauss-Hermite formula. That is, if the random variable Y has a density function $p(y)$ over $(-\infty, +\infty)$, then

$$\begin{aligned} E[f(Y)] &= \int_{-\infty}^{+\infty} f(y)p(y)dy \\ &= \int_{-\infty}^{+\infty} \left(f(x)p(x)\sqrt{2\pi\sigma^2}e^{(x-\mu)^2/(2\sigma^2)}\right) \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/(2\sigma^2)}dx \\ &\doteq \sqrt{2}\sigma\sum_{i=1}^{n} \omega_i f\left(\sqrt{2}\sigma x_i+\mu\right) p\left(\sqrt{2}\sigma x_i+\mu\right) e^{x_i^2}, \end{aligned}$$

where the ω_i and x_i are the Gauss-Hermite quadrature weights and nodes over $(-\infty, +\infty)$, and μ and σ are chosen such that $p(x)$ is close to the normal density function $\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/(2\sigma^2)}$.

If the domain of the random variable is $[a, +\infty)$, then we can use the Gauss-Laguerre formula. That is, if the random variable Y has a density function $p(y)$ over $[a, +\infty)$, then

$$\begin{aligned} E[f(Y)] &= \int_a^{+\infty} f(y)p(y)dy \\ &= \int_0^{+\infty} \left(f(x+a)p(x+a)e^{\lambda x}/\lambda\right) \lambda e^{-\lambda x}dx \\ &\doteq \frac{1}{\lambda}\sum_{i=1}^{n} \omega_i f\left(x_i/\lambda+a\right) p\left(x_i/\lambda+a\right) e^{x_i}, \end{aligned}$$

where the ω_i and x_i are the Gauss-Laguerre quadrature weights and nodes over $[0, +\infty)$, and λ is chosen such that $p(x+a)$ is close to the exponential density function $\lambda e^{-\lambda x}$.

If the domain of the random variable is $(-\infty, b]$, then we can use the Gauss-Laguerre formula too. That is, if the random variable Y has a density function $p(y)$ over $(-\infty, b]$, then

$$\begin{aligned} E[f(Y)] &= \int_{-\infty}^{b} f(y)p(y)dy \\ &= \int_0^{+\infty} \left(f(-x+b)p(-x+b)e^{\lambda x}/\lambda\right) \lambda e^{-\lambda x}dx \\ &\doteq \frac{1}{\lambda}\sum_{i=1}^{n} \omega_i f\left(-x_i/\lambda+b\right) p\left(-x_i/\lambda+b\right) e^{x_i}, \end{aligned}$$

where the ω_i and x_i are the Gauss-Laguerre quadrature weights and nodes over $[0, +\infty)$, and λ is

chosen such that $p(-x+b)$ is close to the exponential density function $\lambda e^{-\lambda x}$.

4.3.5 Multidimensional Integration

If we want to compute a multidimensional integration, when the dimension is low, then we can apply the product rule to do it. For example, suppose that we want to compute $E(f(X))$ where X is a random vector with multivariate uniform distribution over $[-1,1]^d$, then

$$\begin{aligned} E(f(X)) &= \int_{[-1,1]^d} f(x)\ dx \\ &\doteq \frac{1}{2^d}\sum_{i_1=1}^{n}\cdots\sum_{i_d=1}^{n}\omega_{i_1}\cdots\omega_{i_d} f\left(x_{i_1},\cdots,x_{i_d}\right) \end{aligned}$$

where the ω_i and x_i are the Gauss-Legendre quadrature weights and nodes over $[-1,1]$.

Suppose that we want to compute $E(f(X))$ where X is a random vector with multivariate normal distribution $N(\mu,\Sigma)$ over $(-\infty,+\infty)^d$, where μ is the mean column vector, and Σ is the covariance matrix, then we could do the Cholesky factorization at first, i.e., find a lower triangular matrix L such that $\Sigma = LL^\top$. This is feasible as Σ must be a semi-positive definite matrix from the covariance property. Thus,

$$\begin{aligned} E(f(X)) &= \left((2\pi)^d\det(\Sigma)\right)^{-1/2}\int_{R^d} f(y)e^{-(y-\mu)^\top\Sigma^{-1}(y-\mu)/2}dy \\ &= \left((2\pi)^d\det(L)^2\right)^{-1/2}\int_{R^d} f\left(\sqrt{2}Lx+\mu\right)e^{-x^\top x}2^{d/2}\det(L)dx \\ &\doteq \pi^{-\frac{d}{2}}\sum_{i_1=1}^{n}\cdots\sum_{i_d=1}^{n}\omega_{i_1}\cdots\omega_{i_d} f(\sqrt{2}l_{11}x_{i_1}+\mu_1, \\ &\quad \sqrt{2}(l_{21}x_{i_1}+l_{22}x_{i_2})+\mu_2,\cdots,\sqrt{2}(\sum_{j=1}^{d}l_{dj}x_{i_j})+\mu_d), \end{aligned}$$

where the ω_i and x_i are the Gauss-Hermite quadrature weights and nodes over $(-\infty,\infty)$, and l_{ij} is the (i,j)-element of L, and $\det(\cdot)$ means the matrix determinant operator.

While the dimension is high, the product rule becomes infeasible because it will use n^d points and associated weights which is called the "curse of dimensionality". But we can use monomial formulas which is a nonproduct approach. A degree-k monomial formula uses N points x^i and associated weights ω_i such that the $\sum_{i=1}^{N}\omega_i f(x^i)$ is equal to the exact integration of f if f is a polynomial with a degree k or lower. The reader can refer to Judd [28] and Stroud [52] for the detailed formulas.

The monomial formulas are especially useful in the integration operator of the DP process if we use a low degree Chebyshev polynomial approximation. In that case, we try to compute the integration $E\left(\hat{V}(x^+ \mid x)\right)$ in the objection function, where $\hat{V}(x)$ is a degree-k Chebyshev polynomial. Thus if we can have a degree-k monomial quadrature formula, then the calculation of the integration will be exact.

Sometimes the dimension of the multivariate integration could be reduced, even multivariate integration could be transformed into univariate integration. For example, suppose that we have a normal random vector $X \sim N(\mu, \Sigma)$ in $\mathbb{R}^d$, and we want to compute $E[f(X)]$ while the multivariate function f has the form $f(x) = g\left(\sum_{i=1}^d a_i x_i\right)$ for some constants a_i and $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. From the property of normal random vector, $Y = \sum_{i=1}^d a_i X_i$ has a normal distribution with mean $\sum_{i=1}^d a_i \mu_i$ and variance $a^\top \Sigma a$. Thus $E[f(X)] = E[g(Y)]$, which is a univariate integration and can be estimated by the Gauss-Hermite quadrature rule. This technique will be applied in the portfolio optimization problems.

If the dimension of the multivariate integration is high, and it can not be reduced, then we may have to apply Monte-Carlo-like method. When we applied the Monte-Carlo-like method in the DP scheme, it is often called as stochastic DP method.

4.3.6 Estimation of Parameters and Distribution

In the previous sections, we always assume that the probability distribution, discrete state values and their transition probability matrix are given. For example, in the portfolio optimization problem, we often assume that the return of the asset has a normal or log-normal distribution, the parameters including the interest rate, the mean return and the standard deviation of the assets are given at first. And if we let these parameters follow discrete Markov processes, then we also assume that their discrete values and transition probability matrix are known in advance. Moreover, if we let these parameters follow continuous stochastic processes, then some new parameters defining the stochastic processes should be given at first.

But in the real life, these information cannot be seen usually. So we have to use some ways to estimate them.

For example, for the return of an asset, we can use the historical return data of this asset to plot out its histogram, then use this historical distribution as the estimation of the distribution of the return. When the histogram is close to a normal distribution, we may assume that the return of this asset has a normal distribution.

The next step is to estimate the mean and standard deviation of the return. One way is to use the historical mean and historical standard deviation as the estimation. The second way is to apply some autoregressive methods such as ARIMA and GARCH to estimate them. The third way is to use some regression methods with some economic factors as prediction variables to estimate the mean and standard deviation. The fourth way is to use maximum likelihood method to estimate them. For the standard deviation, it may be better to use an implied volatility of this asset as the estimation.

If we assume that the mean and standard deviation follow discrete Markov processes, then we can use Hidden Markov Model (HMM) plus Expectation-Maximum (EM) algorithm to estimate their discrete values and transition probability matrix.

Now if we have multiple assets, the distribution of each asset return is estimated and the correlation matrix is estimated too, then we may want to know the multivariate distribution of the returns of these assets. One way is to use copula methods. One common copula is Gaussian copula which

assumes the multivariate distribution is normal if the marginal distributions are normal.

Chapter 5

Optimal Growth Problems

There are plenty of applications of dynamic programming (DP) method in economics. In this chapter we present the application in the optimal growth models.

5.1 Deterministic Optimal Growth Problems

The simplest infinite-horizon economic problem is the discrete-time optimal growth model with one good and one capital stock, which is a deterministic model. It is to find the optimal consumption function such that the total utility over the infinite-horizon time is maximal, i.e.,

$$\begin{aligned} V(k_0) = \max_{c,l} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t) \\ \text{s.t.} \quad & k_{t+1} = F(k_t) - c_t, \quad t \geq 0, \end{aligned}$$

where k_t is the capital stock at time t with k_0 given, c_t is the consumption, β is the discounter factor, $F(k) = k + f(k)$ with $f(k_t)$ the aggregate net production function, and $u(c_t)$ is the utility function. This objective function is time-separable.

If we add a new control variable l which is the labor supply in the above model, it becomes

$$\begin{aligned} V(k_0) = \max_{c,l} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t, l_t) \\ \text{s.t.} \quad & k_{t+1} = F(k_t, l_t) - c_t, \quad t \geq 0, \end{aligned}$$

where $F(k, l) = k + f(k, l)$ with $f(k_t, l_t)$ the aggregate net production function, and $u(c_t, l_t)$ is the utility function. This objective function is still time-separable.

After adding an irreversibility constraint, the problem becomes

$$\begin{aligned} V(k_0) = \max_{c,l} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t, l_t) \\ \text{s.t.} \quad & k_{t+1} = F(k_t, l_t) - c_t, \\ & k_{t+1} \geq (1-\delta)k_t, \quad t \geq 0, \end{aligned}$$

where δ is the depreciation rate.

5.2 Initialization Step in Dynamic Programming

The DP version of the discrete-time optimal growth problem is

$$V(k) = \max_{c,l} \quad u(c,l) + \beta V(F(k,l) - c)$$

which is the Bellman equation. From the above equation, the steady states k^* and its corresponding optimal control variables (c^*, l^*) should satisfy the following equations:

$$\begin{aligned} & k^* = F(k^*, l^*) - c^*, \\ & u_l(c^*, l^*) + \beta V'(k^*) F_l(k^*, l^*) = 0, \\ & u_c(c^*, l^*) - \beta V'(k^*) = 0, \\ & V'(k^*) = \beta V'(k^*) F_k(k^*, l^*). \end{aligned}$$

They can be simplified as

$$\begin{cases} k^* = F(k^*, l^*) - c^*, \\ u_l(c^*, l^*) + u_c(c^*, l^*) F_l(k^*, l^*) = 0, \\ \beta F_k(k^*, l^*) = 1. \end{cases} \tag{5.1}$$

In our numerical examples, we will choose two different initial guess, one good choice is $\hat{V}(k; b^0) = u(F(k, l^*) - k, l^*)/(1-\beta)$, the another is $\hat{V}(k; b^0) = 0$.

5.3 Maximization Step in Dynamic Programming

The most time-consuming part of the parametric DP algorithm is the maximization step. The maximization step for the growth model is

$$v_i = \max_{c_i, l_i} \; u(c_i, l_i) + \beta \hat{V}(F(k_i, l_i) - c_i; b^t),$$

or

$$v_i = \max_{k_i^+, l_i} \; u(F(k_i, l_i) - k_i^+, l_i) + \beta \hat{V}(k_i^+; b^t),$$

or

$$\begin{aligned} v_i &= \max_{k_i^+, c_i, l_i} u(c_i, l_i) + \beta \hat{V}(k_i^+; b^t) \\ &\quad \text{s.t.} \quad k_i^+ = F(k_i, l_i) - c_i, \end{aligned}$$

for $k_i \in K$. These three models are identical theoretically, but may produce different solutions numerically. We can run the computation one node by one node, or only once in block form like

$$\max_{c,l} \sum_{i=1}^{m} \left[u(c_i, l_i) + \beta \hat{V}(F(k_i, l_i) - c_i; b^t) \right].$$

In the maximization step, the concavity of the objective function is a very valuable property. If the maximization step is a concave problem, then the global maximum is the unique local maximum and easy to find. Now, if $u(c, l)$ is concave on (c, l), and $F(k, l)$ is concave on l, then Hessian matrix of $u(F(k, l) - k^+, l)$ for variables k^+ and l is

$$M = \begin{bmatrix} u_{cc} & -u_{cc}F_l - u_{cl} \\ -u_{cc}F_l - u_{cl} & u_{cc}F_l^2 + 2u_{cl}F_l + u_{ll} + u_c F_{ll} \end{bmatrix}.$$

We have

$$\begin{aligned} M_{11} &= u_{cc} < 0, \\ M_{22} &= u_{cc}F_l^2 + 2u_{cl}F_l + u_{ll} + u_c F_{ll} \le 2(-\sqrt{u_{cc}u_{ll}} + u_{cl})F_l + u_c F_{ll} < 0, \\ \det(M) &= u_{cc}u_{ll} - u_{cl}^2 + u_c u_{cc} F_{ll} > 0, \end{aligned}$$

as u_{cc}, u_{ll}, F_{ll} are negative, u_c, F_l are positive, and $\det \begin{pmatrix} u_{cc} & u_{cl} \\ u_{cl} & u_{ll} \end{pmatrix} = u_{cc}u_{ll} - u_{cl}^2 > 0$, from the concavity of u and F. Then M is a negative definite matrix. Thus, if $\hat{V}(k^+; b^t)$ is concave on k^+ too, then the objective function of the second model of the maximization step must be concave. Similarly, we can show that if $\hat{V}(k^+; b^t)$ is concave on k^+ and $F(k, l)$ is concave on l, then $\hat{V}(F(k, l) - c; b^t)$ is concave on (c, l) such that the objective function of the first model of the maximization step must be concave if $u(c, l)$ is concave on (c, l) too. For the objective function of the third model of the maximization step, obviously it is concave on (k^+, c, l) if $\hat{V}(k^+; b^t)$ is concave on k^+ and $u(c, l)$ is concave on (c, l). Therefore, if we choose a concave initial guess and $u(c, l)$ is concave on (c, l), and we can use some shape-preserving approximation method in the fitting step, then every maximization step in the numerical DP algorithm will be a concave problem and easy to be solved theoretically.

In our numerical examples, the aggregate net production function $f(k, l)$ is chosen as $f(k, l) = Ak^\alpha l^{1-\alpha}$ with $\alpha = 0.25$ and $A = (1 - \beta)/(\alpha\beta)$, and utility functions are chosen as $u(c, l) = \log(c) + B\log(3 - l)$ or $u(c, l) = c^{1-\gamma}/(1 - \gamma) - Bl^{1+\eta}/(1 + \eta)$ (it could be $u(c, l) = \log(c) + \log(1 - l)$ or $u(c, l) = c^{1-\gamma}/(1 - \gamma) - l^{1+\eta}/(1 + \eta)$), where $\gamma > 0$ and $\eta > 0$ are two input parameters, B is chosen such that $k^* = 1$ and $l^* = 1$, i.e., $B = 2(1 - \alpha)$ for the logarithm utility or $B = (1 - \alpha)A^{1-\gamma}$ for the

power utility. From numerical point of view, to let the functions feasible in the process of solving the maximization step, we have to extend the definition of logarithm and power functions to nonnegative region, and add some constraints, such as $c > 0$ and $0 < l < 3$ in the logarithm utility.

The order to solve the optimal problems for the approximation nodes $\{k_1, \ldots, k_m\}$ in one maximization step is important too. The intuitive choice is to solve the problems from k_1 to k_m (here we assume that $k_1 < k_2 < \cdots < k_m$). But in fact, the order reverse to the order of the last maximization step is better. That is, if the order of last maximization step is from k_1 to k_m, then this step's order is from k_m to k_1. This will benefit us from the "warm start" option provided by the optimization solvers.

Another order to speed up the solving process of the maximization step is to start at the node k^* and the node nearest to k^* (denoted as k_i) if k^* is not a node (we could use (k^*, c^*, l^*) as the initial guess of (k^+, c, l) for this node), and then to proceed from k_i to k_m and from k_i to k_1 separately with "warm start" option.

Gauss-Seidel method may be another speeding-up technique.

5.4 Stochastic Optimal Growth Problems

We consider the stochastic optimal growth model now. Let θ denote the current productivity level and $f(k, l, \theta)$ denote net income. Define $F(k, l, \theta) = k + f(k, l, \theta)$, and assume that θ follows $\theta_{t+1} = g(\theta_t, \varepsilon_t)$ where ε_t are i.i.d. disturbances. Then the infinite-horizon discrete-time optimization problem becomes

$$\begin{aligned} V(k_0, \theta_0) = \max_{k,c,l} \quad & E\left\{\sum_{t=0}^{\infty} \beta^t u(c_t, l_t)\right\} \\ \text{s.t.} \quad & k_{t+1} = F(k_t, l_t, \theta_t) - c_t, \\ & \theta_{t+1} = g(\theta_t, \varepsilon_t), \quad t \geq 0, \end{aligned}$$

where x_0 and θ_0 are given. The θ has many economic interpretations. In the life-cycle interpretation, θ is a state variable that may affect either asset income, labor income, or both. In the monopolist interpretation, θ may reflect shocks to costs, demand, or both.

Its DP formulation is

$$V(k, \theta) = \max_{c,l} \quad u(c, l) + \beta E\{V(F(k, l, \theta) - c, \theta^+) \mid \theta\},$$

where θ^+ is next period's θ realization.

In the above model, k_{t+1} is a deterministic variable which is fully dependent on k_t, l_t, θ_t and c_t.

But we can extend it to a stochastic capital stock case:

$$\begin{aligned} V(k_0,\theta_0) = \max_{k,c,l} \quad & E\left\{\sum_{t=0}^{\infty}\beta^t u(c_t,l_t)\right\} \\ \text{s.t.} \quad & k_{t+1} = F(k_t,l_t,\theta_t) - c_t + \epsilon_t, \\ & \theta_{t+1} = g(\theta_t,\varepsilon_t), \quad t \geq 0, \end{aligned}$$

where ϵ_t are i.i.d. disturbances, and independent of ε_t. Its DP formulation is

$$\begin{aligned} V(k,\theta) = \max_{c,l} \quad & u(c,l) + \beta E\{V(k^+,\theta^+) \mid \theta\} \\ \text{s.t.} \quad & k^+ = F(k,l,\theta) - c + \epsilon, \\ & \theta^+ = g(\theta,\varepsilon). \end{aligned}$$

5.5 Multi-Dimensional Optimal Growth Problems

An n-dimensional discrete-time optimal growth problem is

$$\begin{aligned} V(k_0) \quad = \quad \max_{k,c,l} \quad & \sum_{t=0}^{\infty}\beta^t u(c_t,l_t,k_{t+1}) \\ \text{s.t.} \quad & k^+_{i,t+1} = F(k_{it},l_{it}) - c_{it}, \quad i = 1,\ldots,n, \quad t \geq 0, \end{aligned}$$

where the utility function can be chosen as

$$u(c,l,k) = \sum_{i=1}^{n}\left[c_i^{1-\gamma}/(1-\gamma) - Bl_i^{1+\eta}/(1+\eta)\right] + \sum_{i\neq j}\mu_{ij}(k_i-k_j)^2,$$

for n-dimensional vectors c, l and k.

Its DP formulation is

$$\begin{aligned} V(k) \quad = \quad \max_{k^+,c,l} \quad & u(c,l,k^+) + \beta V(k^+) \\ \text{s.t.} \quad & k_i^+ = F(k_i,l_i) - c_i, \quad i = 1,\ldots,n. \end{aligned}$$

In multi-dimensional DP, multiple loop sequence is important by using last node's solution, or combining last node's solution and last solution of the same node. For example, in two-dimensional case, we should loop from left to right in the first line of nodes, then from right to left in the second line of nodes, and then from left to right in the third line of nodes, continue this sequence until the end line. This is more important if we are using "Warm Start" option.

Using complete Chebyshev polynomials in multi-dimensional numerical DP algorithms will save time in computing with almost same accuracy by comparison with tensor Chebyshev polynomials.

5.6 Multi-Dimensional Stochastic Optimal Growth Problems

We consider the multi-dimensional stochastic optimal growth model now. Let θ denote the current productivity levels and $f(k,l,\theta)$ denote net income. Define $F(k,l,\theta) = k + f(k,l,\theta)$, (e.g., $f(k,l,\theta) = \theta A k^{\alpha} l^{1-\alpha}$), and assume that θ follows $\theta_{t+1} = g(\theta_t, \varepsilon_t)$ where ε_t are i.i.d. disturbances. Then the infinite-horizon discrete-time optimization problem becomes

$$\begin{aligned} V(k_0,\theta_0) = \max_{k,c,l} \quad & E\left\{\sum_{t=0}^{\infty} \beta^t u(c_t, l_t, k_{t+1})\right\} \\ \text{s.t.} \quad & k_{t+1} = F(k_t, l_t, \theta_t) - c_t + \epsilon_t, \\ & \theta_{t+1} = g(\theta_t, \varepsilon_t), \quad t \geq 0, \end{aligned}$$

where k_0 and θ_0 are given. Here k_t, l_t and c_t are multi-dimensional vectors, θ may be a random variable or random vector.

Its DP formulation is

$$\begin{aligned} V(k,\theta) \quad = \quad & \max_{k^+,c,l} \quad u(c,l,k^+) + \beta E[V(k^+,\theta^+) \mid \theta] \\ & \text{s.t.} \quad k^+ = F(k,l,\theta) - c + \epsilon, \end{aligned}$$

where θ^+ is next period's θ realization.

5.6.1 Numerical Examples

We use one numerical example to show the stability and convergence of numerical DP for the above infinite-horizon multi-dimensional stochastic optimal growth problem. We let k, θ, k^+, θ^+, c and l be 2-dimensional vectors, $\epsilon \equiv 0$, $\beta = 0.8$, $[\underline{k}, \bar{k}] = [0.5, 3.0]^2$, $F(k,l,\theta) = k + \theta A k^{\alpha} l^{1-\alpha}$ with $\alpha = 0.25$, and $A = (1-\beta)/(\alpha\beta) = 1$, and

$$u(c,l,k^+) = \sum_{i=1}^{2} \left[c_i^{1-\gamma}/(1-\gamma) - B l_i^{1+\eta}/(1+\eta) \right] + \mu(k_1^+ - k_2^+)^2,$$

with $\gamma = 2$, $\eta = 1$, $\mu_{ij} = 0$, and $B = (1-\alpha)A^{1-\gamma} = 0.75$.

Here we let θ_1^+ and θ_2^+ be independent, and assume that the possible values of θ_i, θ_i^+ are $a_1 = 0.9$ and $a_2 = 1.1$, and the probability transition matrix from θ_i to θ_i^+ is

$$P = \begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix},$$

for each $i = 1, 2$. That is,

$$\Pr[\theta^+ = (a_{j_1}, a_{j_2}) \mid \theta = (a_{i_1}, a_{i_2})] = P_{i_1,j_1} P_{i_2,j_2},$$

where P_{i_d,j_d} is the (i_d, j_d) element of P, for any $i_d, j_d = 1, 2$, $d = 1, 2$.

Therefore,

$$E\{V(k^+, \theta^+) \mid k, \theta = (a_{i_1}, a_{i_2}), c, l\} = \sum_{j_1,j_2=1}^{2} P_{i_1,j_1} P_{i_2,j_2} \cdot V(k_1^+, k_2^+, a_{j_1}, a_{j_2}),$$

where $k_d^+ = F(k_d, l_d, a_{i_d}) - c_d$, for any $i_d = 1, 2$, $d = 1, 2$.

For this example, we use value function iteration method while the continuous function approximation is complete degree-9 Chebyshev polynomial approximation method with 10^2 Chebyshev nodes for continuous state variables, and the optimizer is NPSOL, and the initial value function is 0 everywhere. The stopping criterion for the parametric value function iteration (VFI) method is

$$D_t := \max \frac{|\hat{V}(k, \theta; b^{t+1}) - \hat{V}(k, \theta; b^t)|}{1 + |\hat{V}(k, \theta; b^t)|} < \varepsilon,$$

for $\varepsilon = 10^{-5}$, where b^t is the Chebyshev polynomial coefficient vector at the t-th VFI, and D_t is called as the relative change of $\hat{V}(k, \theta; b^{t+1})$ from $\hat{V}(k, \theta; b^t)$.

The parametric value function iteration method converges at the 48-th iteration with a decreasing process D_t, shown in Figure 5.1.

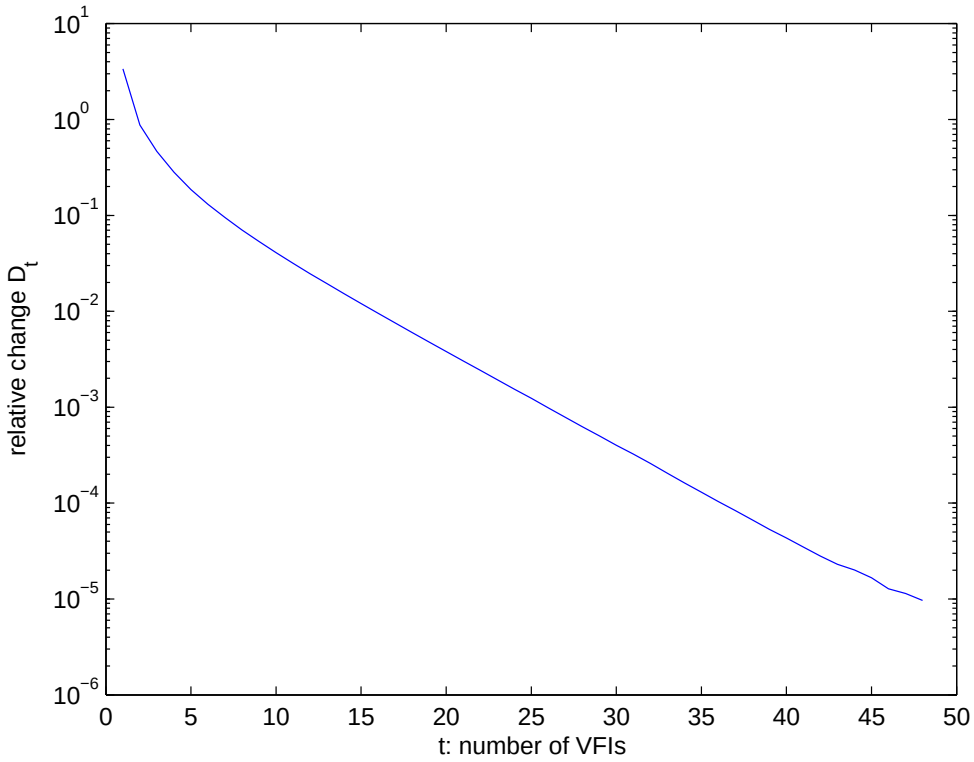


Figure 5.1: Relative change of VFI for growth problems

Chapter 6

Shape Preserving Approximation Method

In economics and finance, many DP models have the monotone and/or concave/convex value functions such that objective functions in their optimization models preserve the shape property theoretically. So if we can have the shape-preserving value function approximation in the fitting step, then it will be very helpful to get good optimal solutions as the local optimizer will be also the global optimizer for convex optimization problems. One good shape-preserving method is the so-called Schumaker shape-preserving interpolation method. J.C.Fiorot and J. Tabka [16], and Steven Pruess [44] give some other shape-preserving splines approximation methods.

6.1 Schumaker Shape-Preserving Spline Method

Here we present the shape-preserving quadratic spline of Schumaker [50] which produces a smooth function which both interpolates data and preserves some shape. We first examine the Hermite interpolation version and then discuss the Lagrange version.

Let us consider the shape-preservation problem on a single interval $[x_1, x_2]$. The basic Hermite problem on the interval takes the data v_1, v_2, s_1, s_2, and constructs a piecewise-quadratic function $s \in C^1[x_1, x_2]$ such that

$$s(x_i) = v_i, \quad s'(x_i) = s_i, \quad i = 1, 2.$$

Here is the interpolation algorithm.

Algorithm 6.1. *Schumaker Shape-Preserving Interpolation*

Step 1. Compute $\delta = (v_2 - v_1)/(x_2 - x_1)$. If $(s_1 + s_2)/2 = \delta$, then

$$s(x) = v_1 + s_1(x - x_1) + \frac{(s_2 - s_1)(x - x_1)^2}{2(x_2 - x_1)},$$

and STOP.

Step 2. If $(s_1-\delta)(s_2-\delta)\geq 0$, set $\xi=(x_1+x_2)/2$; Else if $|s_2-\delta|<|s_1-\delta|$, then let

$$\bar{\xi}=x_1+\frac{2(x_2-x_1)(s_2-\delta)}{(s_2-s_1)},$$

and let $\xi=(x_1+\bar{\xi})/2$; Else let

$$\underline{\xi}=x_2+\frac{2(x_2-x_1)(s_1-\delta)}{(s_2-s_1)},$$

and let $\xi=(x_2+\underline{\xi})/2$. Then

$$s(x)=\begin{cases} v_1+s_1(x-x_1)+C_1(x-x_1)^2, & x\in[x_1,\xi],\\ A_2+\bar{s}(x-\xi)+C_2(x-\xi)^2, & x\in[\xi,x_2],\end{cases}$$

where $C_1=(\bar{s}-s_1)/(2a)$, $A_2=v_1+as_1+a^2C_1$, $C_2=(s_2-\bar{s})/(2b)$, $\bar{s}=[2(v_2-v_1)-(as_1+bs_2)]/(x_2-x_1)$, $a=\xi-x_1$, and $b=x_2-\xi$.

The reader can refer to Judd [28] and Larry Schumaker [50] for detailed discussion.

Notice that a and b are used as denominators for C_1 and C_2. This may give rise to the problem when a or b is close to 0, i.e., ξ is close to x_1 or x_2. Practically, this problem is often encountered in our numerical examples. Numerically, we should not use $a=\xi-x_1$ or $b=x_2-\xi$ to compute a or b if ξ is close to x_1 or x_2.

Here we will propose a new version of the algorithm. From the step 2 of the algorithm, we know that if $(s_1-\delta)(s_2-\delta)<0$ and $|s_2-\delta|<|s_1-\delta|$ then

$$\xi=(x_1+\bar{\xi})/2=x_1+\frac{(x_2-x_1)(s_2-\delta)}{(s_2-s_1)},$$

else if $|s_2-\delta|\geq|s_1-\delta|$ then

$$\xi=(x_2+\underline{\xi})/2=x_2+\frac{(x_2-x_1)(s_1-\delta)}{(s_2-s_1)}.$$

Note that

$$x_1+\frac{(x_2-x_1)(s_2-\delta)}{(s_2-s_1)}=x_2+\frac{(x_2-x_1)(s_1-\delta)}{(s_2-s_1)},$$

we just need to set

$$\xi=x_1+\frac{(x_2-x_1)(s_2-\delta)}{(s_2-s_1)},$$

if $(s_1-\delta)(s_2-\delta)<0$. This saves the distinction of computing ξ under the comparison between $|s_2-\delta|$ and $|s_1-\delta|$ in the original algorithm 6.1. Thus,

$$a=\xi-x_1=(s_2-\delta)/\lambda,\quad b=x_2-\xi=(\delta-s_1)/\lambda,$$

where $\lambda = (s_2 - s_1)/(x_2 - x_1)$. It follows that

$$\begin{aligned} \bar{s} &= \frac{2(v_2 - v_1) - (as_1 + bs_2)}{(x_2 - x_1)} \\ &= 2\delta - \frac{(s_2 - \delta)}{\lambda}\frac{s_1}{(x_2 - x_1)} - \frac{(\delta - s_1)}{\lambda}\frac{s_2}{(x_2 - x_1)} \\ &= 2\delta - \frac{(s_2 - \delta)s_1 + (\delta - s_1)s_2}{(s_2 - s_1)} = 2\delta - \delta = \delta, \end{aligned}$$

by $\delta = (v_2 - v_1)/(x_2 - x_1)$ if $(s_1 - \delta)(s_2 - \delta) < 0$. There are several issues we still need to worry about numerically: $s_1 \simeq s_2$, $s_1 \approx \delta$, or $s_2 \approx \delta$. The case $s_1 \approx s_2$ happens practically when the shape of the function is very close to a straight line in this interval. By replacing the condition $(s_1 + s_2)/2 = \delta$ in the step 1 into $|(s_1 + s_2)/2 - \delta| < \epsilon$ for some given tolerance $\epsilon > 0$, these cases disappear.

By $C_1 = (\bar{s} - s_1)/(2a)$, we have

$$A_2 = v_1 + as_1 + a^2 C_1 = v_1 + a(s_1 + \bar{s})/2.$$

Moreover, if $(s_1 - \delta)(s_2 - \delta) \geq 0$, then from $\xi = (x_1 + x_2)/2$ we have $a = b = (x_2 - x_1)/2$, and then

$$\bar{s} = \frac{2(v_2 - v_1) - (as_1 + bs_2)}{(x_2 - x_1)} = 2\delta - \frac{s_1 + s_2}{2}.$$

Algorithm 6.2. *Revised Schumaker Shape-Preserving Interpolation*

Step 1. Compute $\delta = (v_2 - v_1)/(x_2 - x_1)$. If $|(s_1 + s_2)/2 - \delta| < \epsilon$, then

$$s(x) = v_1 + s_1(x - x_1) + \frac{(s_2 - s_1)(x - x_1)^2}{2(x_2 - x_1)},$$

and STOP.

Step 2. If $(s_1 - \delta)(s_2 - \delta) \geq 0$, set

$$\xi = (x_1 + x_2)/2, \;\; a = b = \xi - x_1, \;\; \bar{s} = 2\delta - \frac{s_1 + s_2}{2}.$$

Else let

$$\lambda = \frac{s_2 - s_1}{x_2 - x_1}, \;\; a = (s_2 - \delta)/\lambda, \;\; b = (\delta - s_1)/\lambda, \;\; \xi = x_1 + a, \;\; \bar{s} = \delta.$$

Then

$$s(x) = \begin{cases} v_1 + s_1(x - x_1) + C_1(x - x_1)^2, & x \in [x_1, \xi], \\ A_2 + \bar{s}(x - \xi) + C_2(x - \xi)^2, & x \in [\xi, x_2], \end{cases}$$

where $C_1 = (\bar{s} - s_1)/(2a)$, $A_2 = v_1 + a(s_1 + \bar{s})/2$, and $C_2 = (s_2 - \bar{s})/(2b)$.

This revised algorithm not only has less computation, but also is more accurate than the original version.

Now we consider a general interpolation problem. If we have Hermite data $\{(x_i, v_i, s_i) : i = 1, \ldots, n\}$, we then apply the shape-preserving interpolant algorithm to each interval to find $\xi_i \in [x_i, x_{i+1}]$. If we have Lagrange data, $\{(x_i, v_i) : i = 1, \ldots, n\}$, we must first add estimates of the slopes and then proceed as we do with Hermite data. Schumaker suggests the following formulas for estimating slopes s_1 through s_n:

$$\begin{aligned} L &= \left[(x_{i+1} - x_i)^2 + (v_{i+1} - v_i)^2\right]^{1/2}, \quad \delta_i = \frac{v_{i+1} - v_i}{x_{i+1} - x_i}, \quad i = 1, \ldots, n-1, \\ s_i &= \begin{cases} \frac{L_{i-1}\delta_{i-1} + L_i\delta_i}{L_{i-1} + L_i}, & \text{if} \quad \delta_{i-1}\delta_i > 0, \\ 0, & \text{if} \quad \delta_{i-1}\delta_i \le 0, \end{cases} \quad i = 2, \ldots, n-1, \\ s_1 &= \frac{3\delta_1 - s_2}{2}, \quad s_n = \frac{3\delta_{n-1} - s_{n-1}}{2}. \end{aligned}$$

But the estimation might not preserve the monotonicity. Aatos Lahtinen [31, 32] suggested a variation version to avoid the problem.

Since Schumaker method is only a quadratic splines approximation, it is only C^1 and not smooth enough for some optimization solvers such that we need some more care in the maximization step if we use Schumaker approximation in the fitting step of the DP procedure. When the value function is concave or convex in DP, if we can compute the values (x_i, v_i, s_i) approximately enough, then we should have $(s_1 - \delta)(s_2 - \delta) < 0$. If not, we need to modify the data set instead of keeping these data and using the branch for $(s_1 - \delta)(s_2 - \delta) \ge 0$.

6.2 Shape Correction and Preserving

For a univariate approximation problem, Schumaker interpolation method preserves the shape properties including monotonicity and concavity. But many approximation methods such as Chebyshev approximation do not have the shape-preserving property, so we can use least-square-error approximation with shape constraints to guarantee the shape-preservation partially. For example, if we know that the value function is strictly increasing and concave, then we could try to find the optimal parameters b in the following model:

$$\begin{aligned} \min_b \ & \sum_{i=1}^{n} \left(\hat{V}(x_i; b) - v_i\right)^2 \\ \text{s.t.} \ & \hat{V}'(x_i; b) > 0, \quad i = 1, \ldots, n, \\ & \hat{V}''(x_i; b) < 0, \quad i = 1, \ldots, n. \end{aligned}$$

But sometimes this model may return some binding solutions numerically, which we do not hope to happen.

Before doing the shape-preserving fitting, we should modify the values of v_i such that the discrete set of v_i have the same properties of value function. For example, if value function is strictly increasing, then we should have $v_i < v_{i+1}$, which can not be guaranteed by the optimization solvers due to the

numerical errors (e.g., the optimization solver returns a local optimizer which is not global). This step is called shape correction. To do it, we could modify the optimization step in the DP method for finite horizon problems as

$$\begin{aligned} \max_{a_i \in D(x_i,t)} \ & \sum_{i=1}^{m} v_i \\ \text{s.t.} \quad & v_i = u_t(x_i, a_i) + \beta E\{\hat{V}(x_i^+; b^{t+1}) \mid x_i, a_i\}, \quad i = 1, \ldots, m, \\ & v_i < v_{i+1}, \quad i = 1, \ldots, m-1. \end{aligned}$$

But sometimes this model may return some binding solutions numerically, e.g., $v_{i-1} < v_i \approx v_{i+1} < v_{i+2}$ for some i. In this case, we should modify v_i as $(v_{i-1} + v_{i+1})/2$, or v_{i+1} as $(v_i + v_{i+2})/2$. If the linear modification is not good for the approximation method, then we could use quadratic modification, i.e., use three points (x_{i-1}, v_{i-1}), (x_i, v_i), (x_{i+2}, v_{i+2}) to generate a quadratic function $q(x)$, and then modify v_{i+1} as $q(x_{i+1})$.

6.3 Shape-preserving Chebyshev Approximation

One problem for Chebyshev approximation is the absence of shape-preservation in the algorithm. To solve this, one way is to modify the Chebyshev coefficients such that the concavity and monotonicity of the value function can be preserved at the interpolation nodes. That is, we could solve a new least square problem which has a quadratic objective and linear inequality constraints:

$$\begin{aligned} \min_{c_j} \quad & \sum_{i=1}^{m} \left(\frac{1}{2} c_0 + \sum_{j=1}^{n} c_j T_j \left(\frac{2x_i - a - b}{b - a} \right) - v_i \right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{n} c_j T_j' \left(\frac{2x_i - a - b}{b - a} \right) > 0, \\ & \sum_{j=1}^{n} c_j T_j'' \left(\frac{2x_i - a - b}{b - a} \right) < 0, \quad i = 1, \ldots, m. \end{aligned}$$

We can use the following recursive formula to evaluate $T_j'(z_i)$ and $T_j''(z_i)$ for $z_i = \frac{2x_i - a - b}{b - a}$, $i = 1, \ldots, m$:

$$\begin{aligned} T_0'(x) &= 0, \\ T_1'(x) &= 1, \\ T_{j+1}'(x) &= 2T_j(x) + 2xT_j'(x) - T_{j-1}'(x), \quad j = 1, 2, \ldots, \end{aligned}$$

and

$$\begin{aligned} T_0''(x) &= 0, \\ T_1''(x) &= 0, \\ T_{j+1}''(x) &= 4T_j'(x) + 2xT_j''(x) - T_{j-1}''(x), \quad j = 1, 2, \ldots. \end{aligned}$$

This least square solution may not preserve the shape along the whole domain of x, but sometimes it helps in the DP procedure. Intuitively, the Chebyshev coefficients may be a good initial guess, but in fact sometimes the coefficients in the last iteration of the DP procedure may be better.

When slopes of the value function, v_i', are given by the Envelope theorem for the Bellman equation, we can modify the model as

$$\begin{aligned} \min_{c_j} \quad & \sum_{i=1}^{m} \left(\frac{1}{2}c_0 + \sum_{j=1}^{n} c_j T_j \left(\frac{2x_i - a - b}{b - a} \right) - v_i \right)^2 + \\ & \lambda \sum_{i=1}^{m} \left(\frac{2}{b-a} \sum_{j=1}^{n} c_j T_j' \left(\frac{2x_i - a - b}{b - a} \right) - v_i' \right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{n} c_j T_j'' \left(\frac{2x_i - a - b}{b - a} \right) < 0, \quad i = 1, \ldots, m, \end{aligned}$$

where λ is some given parameter.

6.3.1 Numerical Example

We use one numerical example of the deterministic optimal growth model to illustrate the necessarity of the shape-preserving property. The DP version of the deterministic optimal growth model is

$$V(k) = \max_{c,l} \quad u(c, l) + \beta V(F(k, l) - c),$$

where k is the capital state variable, $c > 0$ is the consumption decision, $l > 0$ is the labor control variable, β is the discount factor, u is a utility function, $F(k, l) = k + f(k, l)$ while $f(k, l) = Ak^{\alpha}l^{1-\alpha}$ is the production function.

In this example, we let $\alpha = 0.25$, $\beta = 0.9$, $A = (1 - \beta)/(\alpha\beta) = 4/9$, $u(c, l) = c^{1-\gamma}/(1 - \gamma) - Bl^{1+\eta}/(1 + \eta)$ with $\gamma = 4$, $\eta = 1$ and $B = (1 - \alpha)A^{1-\gamma}$. According to the system of equations (5.1), we know that the steady state is $k^* = 1$ while the corresponding optimal labor is $l^* = 1$ and optimal consumption is $c^* = f(k^*, l^*) = 4/9$. Moreover, $V(k^*) = u(c^*, l^*)/(1 - \beta) = -80.6836$.

Let the range of k be chosen as $[0.1, 2]$. We use the value function iteration method. The initial guess of the value function is 0 everywhere. The stopping rule for the DP method is $\|V_{n+1} - V_n\| < 10^{-6}$. The approximation method is chosen as the degree-40 expanded Chebyshev polynomial interpolation method with 41 Chebyshev nodes in $[0.1, 2]$. The optimization solver is chosen as KNITRO and the program code is written in AMPL.

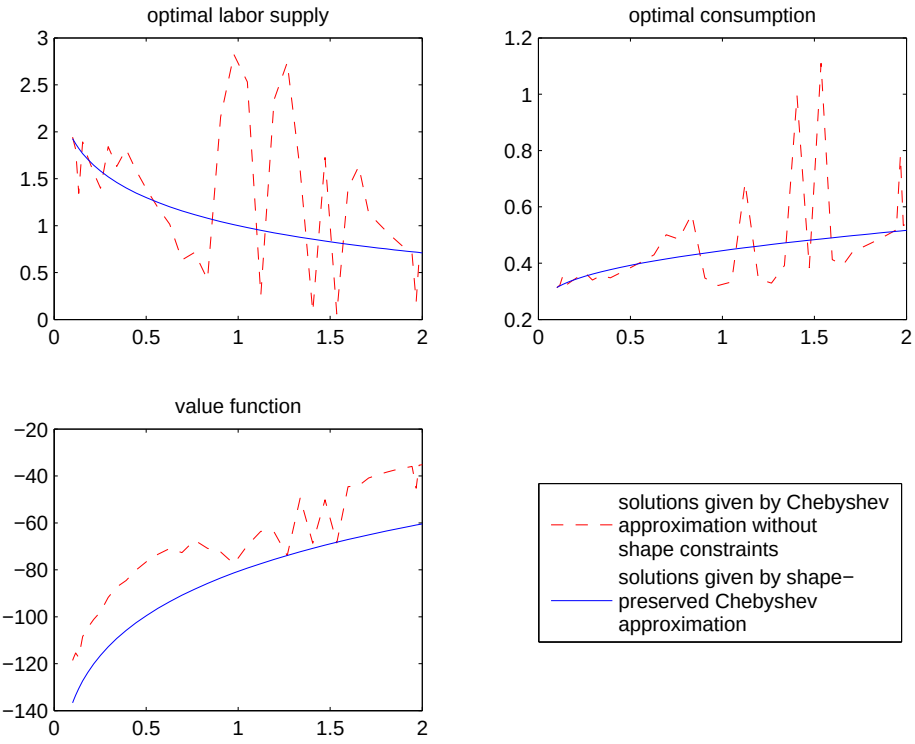


Figure 6.1: Numerical DP with shape-preserving Chebyshev approximation

At first, we try the standard Chebyshev polynomial interpolation method without shape constraints. After 634 iterations, the value function iteration method converges. When the capital state is $k^* = 1$, the corresponding computed optimal $\hat{l}^* = 3.08035$, $\hat{c}^* = 0.241709$ and $\hat{V}(1) = -74.0949$. All of these are far away from the exact optimal labor, consumption and value at the steady state. The approximated optimal control functions and value functions are shown with dashed lines in Figure 6.1. We see that all three functions have many large wiggles.

Next, we try the shape-preserving Chebyshev polynomial approximation method with positive gradient and negative Hessian constraints. The value function iteration method converges at the 78-th step. When the capital state is $k^* = 1$, the corresponding computed optimal $\hat{l}^* = 0.999998$, $\hat{c}^* = 0.444445$ and $\hat{V}(1) = -80.6836$. All of these are very close to the exact optimal labor, consumption and value at the steady state. The approximated optimal control functions and value functions are shown with solid lines in Figure 6.1. We see that all three functions are smooth, monotone and concave.

Chapter 7

Dynamic Programming with Hermite Interpolation

The conventional dynamic programming (DP) algorithm uses the maximization step to compute

$$v_i = V_t(x_i) = \max_{a_i \in D(x_i)} \; u(x_i, a_i) + \beta E\{V_{t+1}(x_i^+) \mid x_i, a_i\},$$

for each pre-specified node x_i, $i = 1, \ldots, m$. Then it applies the Lagrange data set $\{(x_i, v_i) : i = 1, \ldots, m\}$ in the fitting step to construct the approximated value function $\hat{V}_t(x)$. If the fitting step uses a Hermite interpolation method requiring slope information at nodes x_i of $V_t(x)$, such as Schumaker interpolation method, then it seems that we have to estimate the slopes, s_i, by use of finite difference methods. But if we can get the slope information directly, then it will save computation time and make the function approximation more accurate, such that the numerical DP algorithm with Hermite interpolation will be more efficient and accurate.

7.1 Hermite Interpolation

In the approximation methods, we need to know the levels v_i and/or the slopes s_i at nodes x_i, for $i = 1, \ldots, m$. If the Hermite data set $\{(x_i, v_i, s_i) : i = 1, \ldots, m\}$ is available, then the Hermite interpolation method can be applied in the fitting step of the numerical DP algorithm.

The following envelope theorem tells us how to calculate the first derivative of a function which is defined by a maximization operator.

Theorem 7.1. *(Envelope theorem) Let*

$$\begin{aligned} V(x) \quad &= \quad \max_{y} \; f(x, y) \\ &\qquad \text{s.t. } g(x, y) = 0. \end{aligned} \tag{7.1}$$

Suppose that $y^*(x)$ is the optimizer of (7.1), and that $\lambda^*(x)$ is the corresponding shadow price vector. Then

$$\frac{dV(x)}{dx} = \frac{\partial f}{\partial x}(x, y^*(x)) + \lambda^*(x)^\top \frac{\partial g}{\partial x}(x, y^*(x)).$$

Thus, it is not necessary to compute $dy^*(x)/dx$ term in order to get $dV(x)/dx$. If there is an inequality constraint in (7.1), $h(x, y) \geq 0$, we can simply add a slack variable s to transform it into an equality constraints, $h(x, y) - s = 0$, then use the envelope theorem to compute $dV(x)/dx$.

Since the problem (7.1) could have many other equivalent forms, sometimes we can choose an alternative form such that $dV(x)/dx$ can have a much simpler computation. This will be discussed in the next sections.

7.2 Hermite Interpolation in Optimal Growth Models

The DP model for the optimal growth model is

$$V_t(k) = \max_{c,l} \; u(c, l) + \beta V_{t+1}(F(k, l) - c),$$

for some utility function u and production function F. In the maximization step of the numerical DP method, after using the envelope theorem, we have

$$s_i = V_t'(k_i) = \beta V_{t+1}'(k_i^+) F_k(k_i, l_i),$$

where $k_i^+ = F(k_i, l_i) - c_i$ for optimal c_i and l_i given k_i.

In the maximization step of the numerical DP method, we can have the second equivalent model

$$V_t(k) = \max_{k^+,l} \; u(F(k, l) - k^+, l) + \beta V_{t+1}(k^+),$$

by using k^+ and l as control variables, while the first model uses c and l as control variables. By the envelope theorem, we have

$$s_i = V_t'(k_i) = u_c(c_i, l_i) F_k(k_i, l_i),$$

where $c_i = F(k_i, l_i) - k_i^+$ for optimal k_i^+ and l_i given k_i.

The first model we have to compute the derivative of V_{t+1}, and the second model we have to compute the partial derivative of u. Now we have the third equivalent model

$$\begin{aligned} V_t(k) \quad = \quad & \max_{k^+,l,c} \; u(c, l) + \beta V_{t+1}(k^+), \\ & \text{s.t.} \quad F(k, l) - k^+ - c = 0, \end{aligned}$$

by using k^+, l, and c as control variables. By the envelope theorem, we have

$$s_i = V_t'(k_i) = \lambda_i F_k(k_i, l_i),$$

where λ_i is the Lagrange multiplier of the constraint $F(k,l) - k^+ - c = 0$ given k_i. Theoretically, we can show that $\lambda_i = u_c(c_i, l_i) = \beta V'_{t+1}(k_i^+)$ for optimal k_i^+, l_i and c_i given k_i, but this multiplier λ_i is a direct output of the optimization solvers, so we do not need to calculate it.

Similarly, if there is the irreversibility constraint $k^+ \geq (1-\delta)k$ with δ as a depreciation rate, then

$$s_i = V'_t(k_i) = \lambda_i F_k(k_i, l_i) - (1-\delta)\mu_i,$$

where μ_i is the Lagrange multiplier for the irreversibility constraint, which is also available in the output information of the optimization solvers.

7.3 Hermite Interpolation in Multi-period Asset Allocation Models

Here we come to see the application of Hermite interpolation method in one simple multi-period portfolio optimization model. Assume an investor has an initial wealth W_0 and want to invest it into n stocks with a risky random return vector R and one bond with a riskless return R_f, he can reallocate the portfolio at stage $t = 0, 1, \ldots, T-1$, and his objective is to maximize the expected utility of terminal wealth. Then the DP model for this problem is

$$V_t(W_t) = \max_{X_t} \; E\left[V_{t+1}(R_f(W_t - e^\top X_t) + R^\top X_t) \mid W_t\right],$$

for $0 \leq t < T$, where X_t is the amount of dollars invested in the risky assets, while $V_T(W) = u(W)$ for some utility function u. Here we assume that there is no transaction cost for simplicity.

Theoretically, this model is equivalent to

$$\begin{aligned} V_t(W_t) \quad = \quad & \max_{B_t, X_t} \quad E\left[V_{t+1}\left(R_f B_t + R^\top X_t\right) \mid W_t\right] \\ & \text{s.t.} \quad B_t + e^\top X_t = W_t, \end{aligned}$$

where B_t is the amount of dollars invested in the bond.

But numerically there are some difference. By the envelope theorem for the first model, we have

$$V'_t(W_t) = R_f E\left[V'_{t+1}\left(W^*_{t+1}\right) \mid W_t\right],$$

where $W^*_{t+1} = R_f W_t + (R - R_f)^\top X^*_t$, while X^*_t is the optimal allocation vector for the risky assets at time t. This implies that we have to calculate the derivatives of V_{t+1} and then compute the expectation which may take a lot of time for high-dimensional integration. But using the envelope theorem for the second model, we have

$$V'_t(W_t) = -\lambda,$$

where λ is the multiplier for the constraint $B_t + e^\top X_t = W_t$, which is an output of optimization solvers.

If we do not allow borrowing money or shorting stocks, i.e., $B_t \geq 0$ and $X_t \geq 0$, then we just need to add them into the second model, and we can still show that $V_t'(W_t) = -\lambda$.

7.4 Numerical Example

We use a simple numerical example to show the improvement by Hermite interpolation. In the above portfolio optimization problem, we assume that the number of risky asset is $n = 1$, the number of periods is $T = 6$, the riskfree return $R_f = 1.04$, and

$$R = \begin{cases} 0.9, & \text{with probability } 1/2, \\ 1.4, & \text{with probability } 1/2. \end{cases}$$

Let the range of W_0 as $[0.9, 1.1]$, and the utility function is $u(W) = -(W - 0.2)^{-1}$.

By use of the tree method (discussed in section 9.1), we calculate the exact optimal allocations and value functions and show them in Figure 7.1, for $t = 0, 1, \ldots, 5$.

At first, we try Schumaker shape-preserving interpolation method with $N = 30$ nodes, we get the results shown in Figure 7.2. We see that the results are not good in comparison with the exact solutions, particularly for the bond allocation.

Now we come to see the solutions given by Schumaker plus Hermite interpolation method with $N = 30$ nodes, shown in Figure 7.3. We see that the solutions are much closer to the exact solutions. So the Hermite interpolation really helps to improve the accuracy of the solutions.

We also try other approximation methods, such as Chebyshev interpolation method, cubic B-spline interpolation method, natural cubic spline interpolation, not-a-knot cubic spline interpolation method, and so on. We found that the Schumaker plus Hermite interpolation provided the best solutions (cubic B-spline interpolation method also gives good solutions, but is still less accurate than Schumaker plus Hermite interpolation method). We show results from Chebyshev interpolation method in Figure 7.4, cubic B-spline interpolation method in Figure 7.5, and not-a-knot cubic spline interpolation method in Figure 7.6. We also tried natural cubic spline interpolation method, which results are worse than not-a-knot cubic spline interpolation method, so its figure is omitted here.

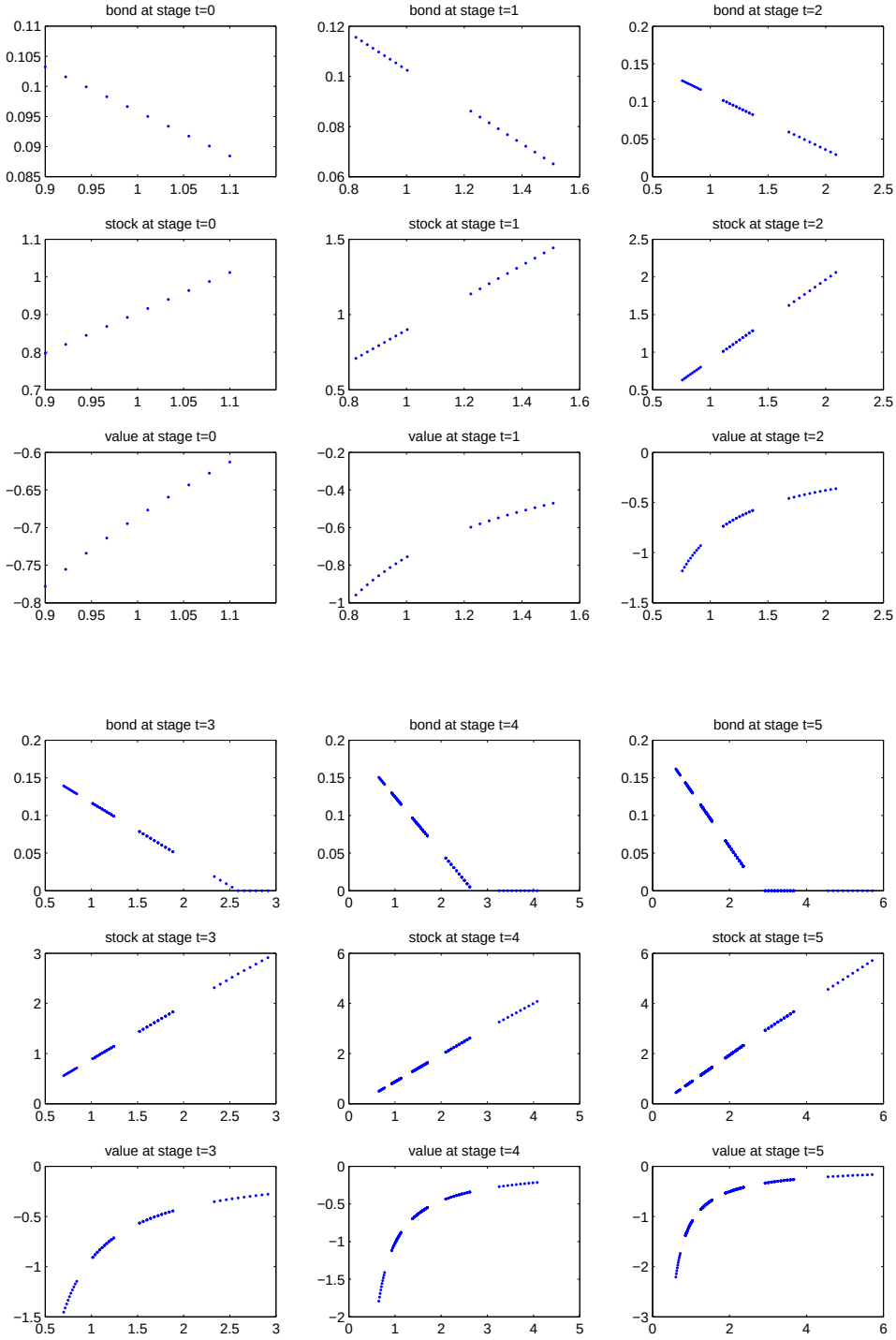


Figure 7.1: Exact optimal allocation and value functions

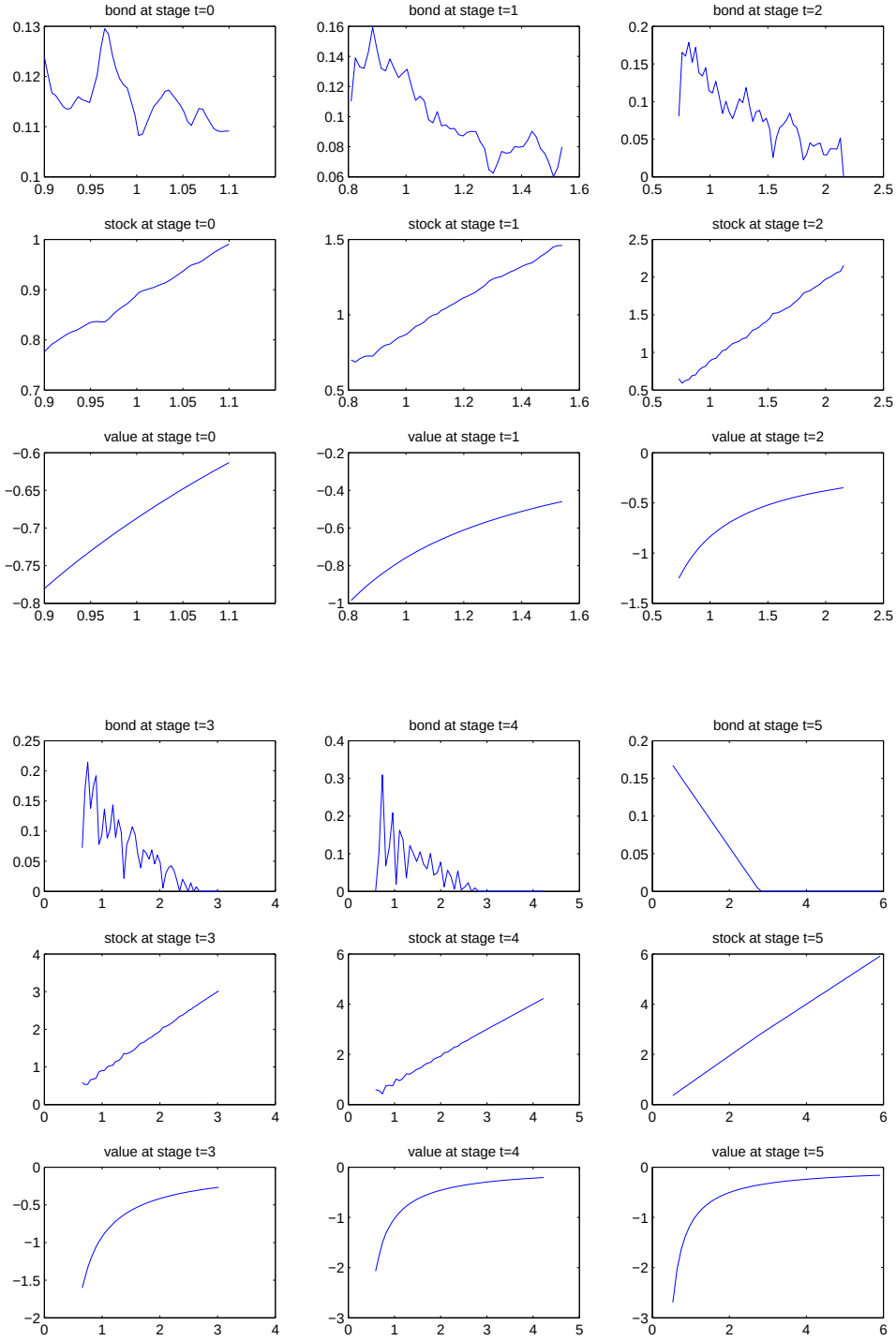


Figure 7.2: Schumaker interpolation

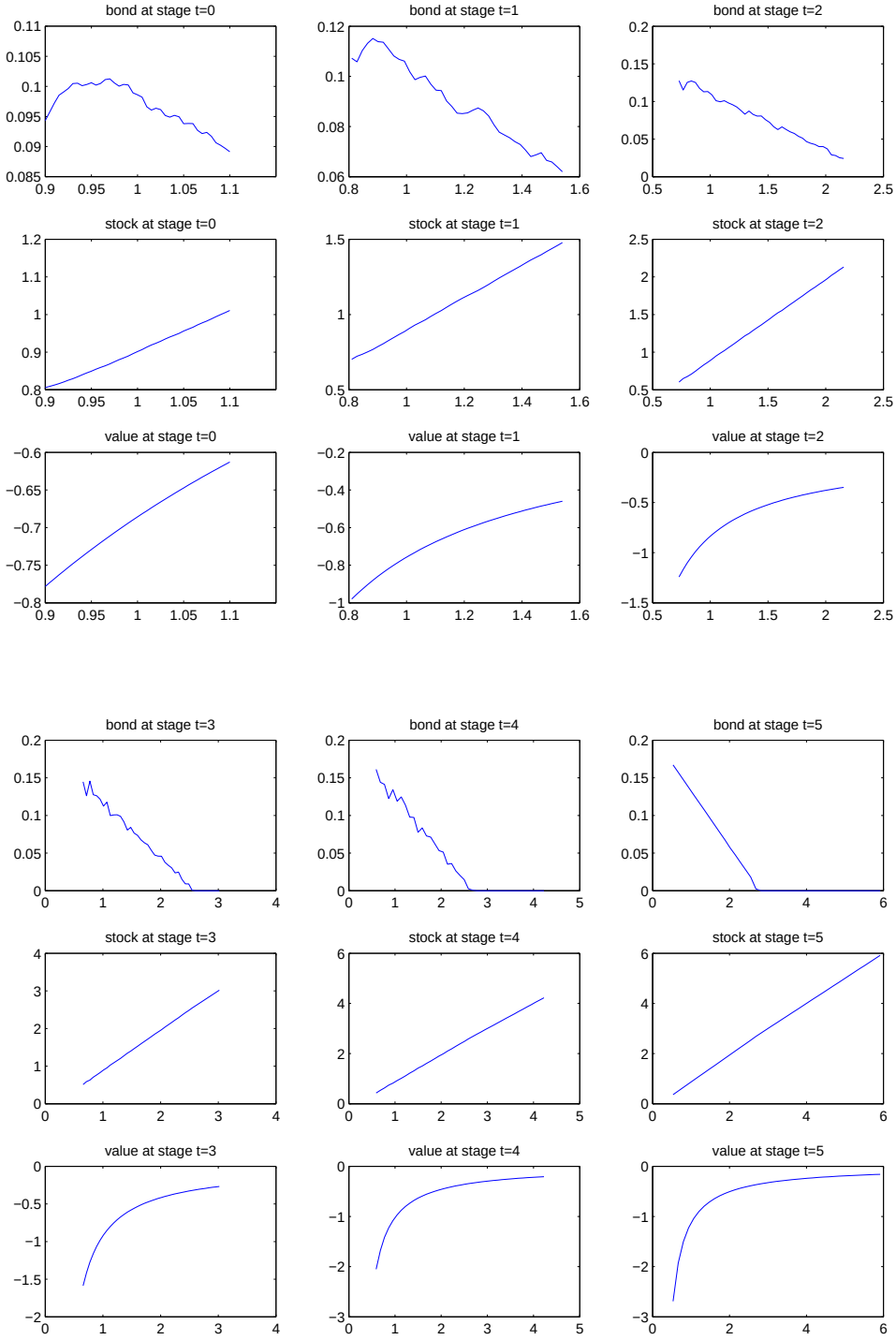


Figure 7.3: Schumaker plus Hermite interpolation

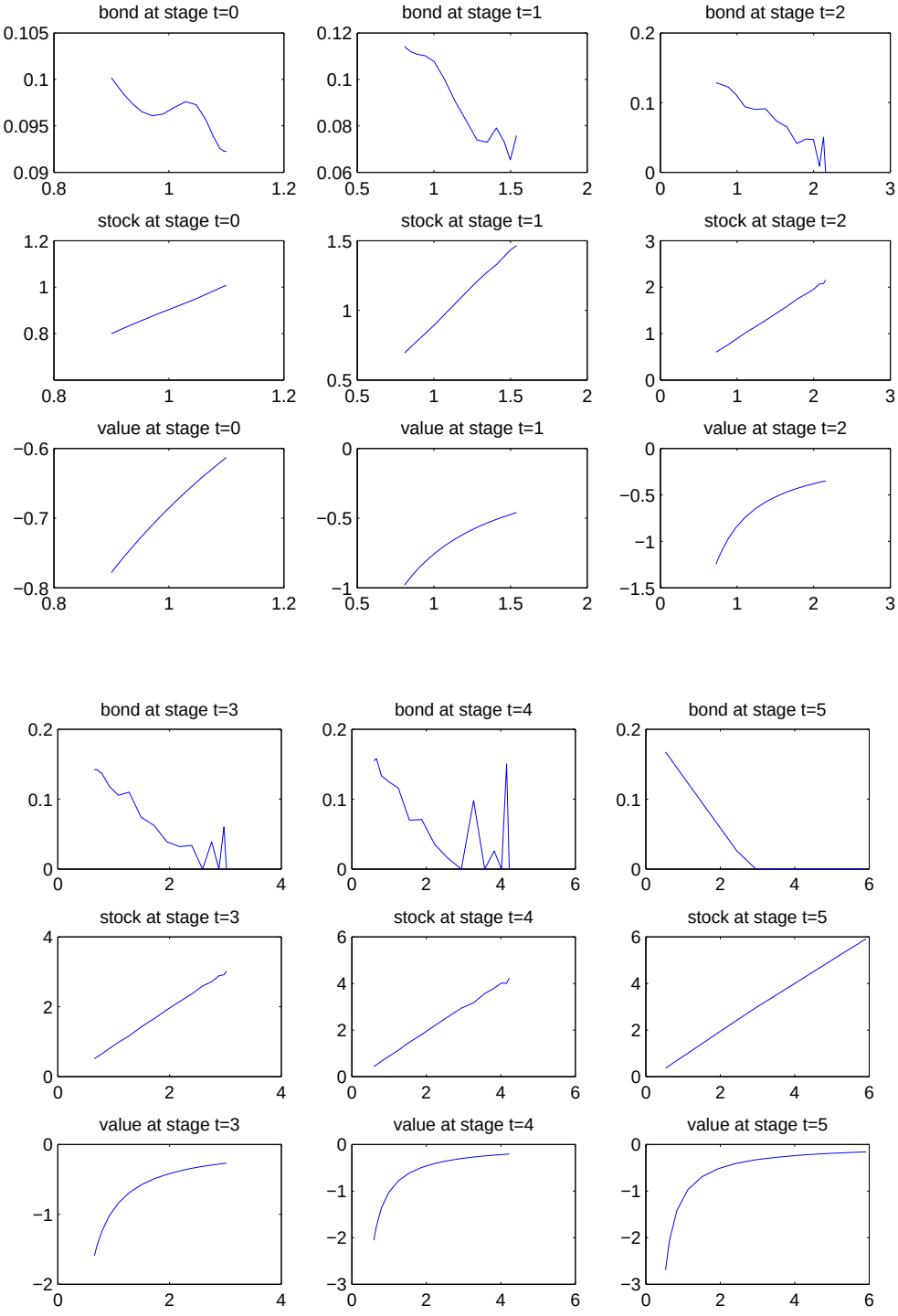


Figure 7.4: Chebyshev interpolation, $N = 16$, deg$= 15$

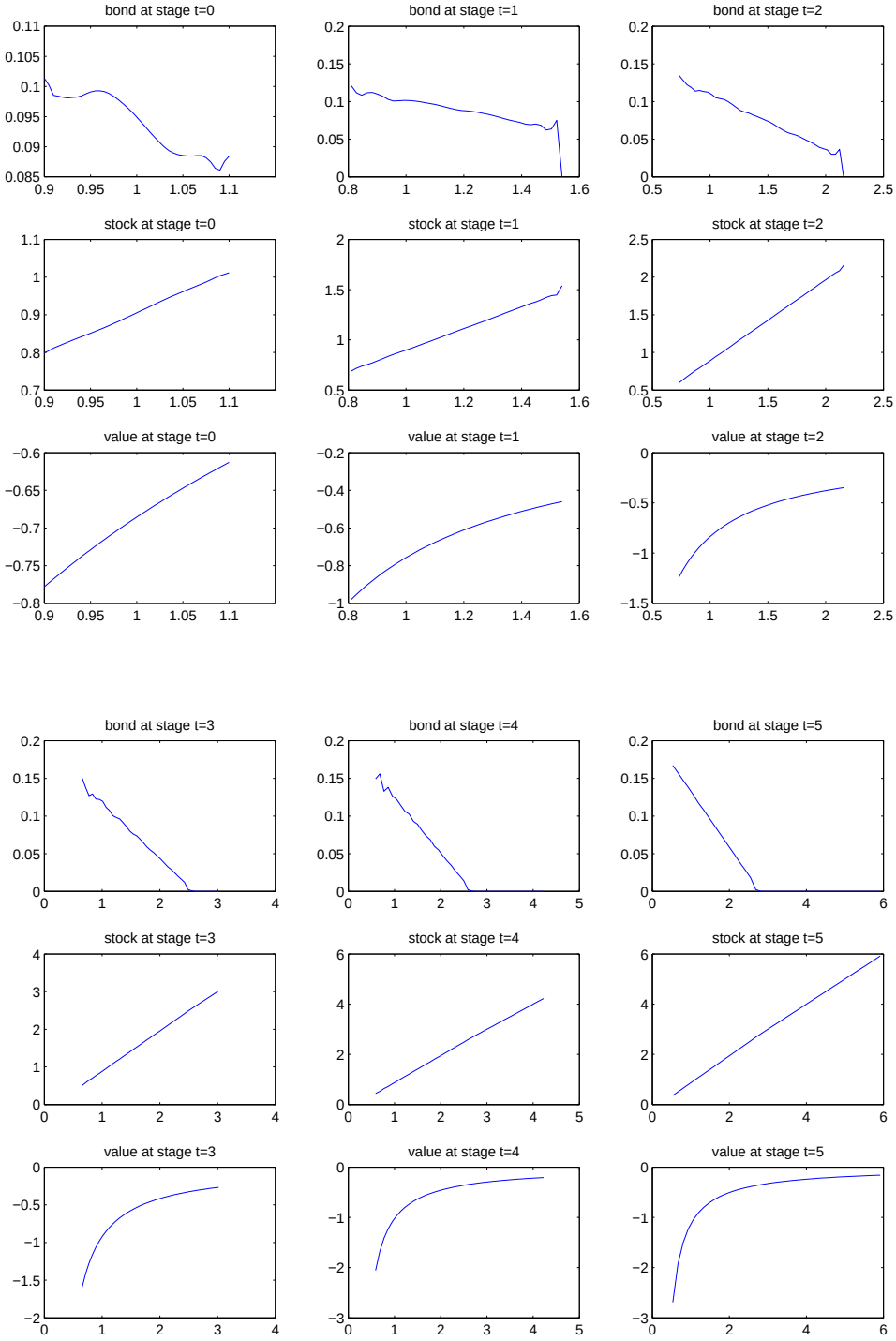


Figure 7.5: Cubic B-spline interpolation, $N = 30$

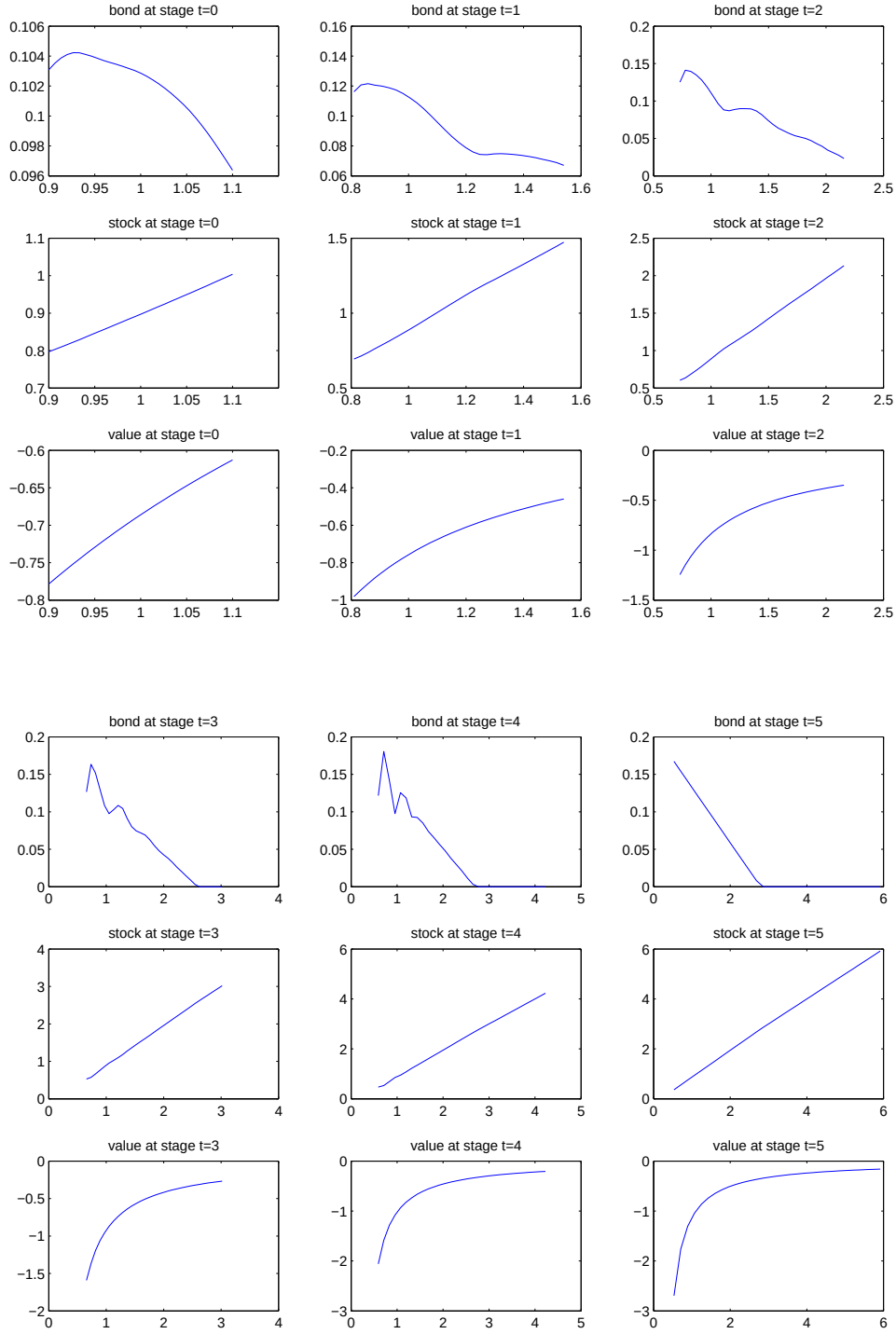


Figure 7.6: Not-a-knot cubic spline interpolation, $N = 30$

Chapter 8

Parallelization

From the numerical dynamic programming (NDP) algorithms 3.1 and 3.2, we see that it is common that NDP requires weeks or months of computation to solve high-dimensional problems, due to the "curse of dimensionality" of the number of optimization problems in each maximization step. It is natural to do parallelization for the maximization steps to break the "curse of dimensionality". In the next section, we developed parallel numerical DP algorithms under the Condor system to solve high-dimensional problems.

The Condor system is a high-throughput computing open-source software framework for distributed parallelization of computationally intensive tasks on a farm of computers, which is freely available for use. It is developed by the Condor team at the University of Wisconsin-Madison. Here we focused the Condor Master-Worker (MW) system for our parallel numerical DP algorithms. The Condor MW system consists of two entities: a master process and a cluster of worker processes. The master process manages basically for decomposing the problem into small tasks, queueing and distributing the tasks among the worker processes, and collecting the results from the workers. The workers' execution is a simple cycle: receive a task from the master, do the task, and then send back its result to the master. In MW applications, Condor acts as a resource management tool for allocating and managing the requested computers available in the pool of machines. A file-based, remote I/O scheme can be used as the message passing mechanism between the master and the workers.

It is easy to program in the master-worker paradigm, as the user can circumvent the typical parallel programming hurdles, such as load balancing, termination detection, and algorithm control information distributing among compute nodes. Moreover, the computation in the master-worker paradigm is fault-tolerant: if a worker fails in executing a portion of the computation, the master simply distributes the portion of the computation to another worker, which can be an additional worker available in the pool of computers. Furthermore, the user can request any number of workers, which is independent of the number of tasks. Thus, if the user requested m workers and there are $n \times m$ tasks, then the fast computers will get more tasks than the slow machines, such that the parallel block problem can almost disappear if n is not small.

8.1 Parallel NDP Algorithms under the Condor MW System

We applied the NDP algorithms with value function iteration (VFI) for problems with multidimensional continuous and discrete states in the Condor MW system. The model is

$$V_t(x,\theta) = \max_{a \in D(x,\theta,t)} \; u_t(x,\theta,a) + \beta E\{V_{t+1}(x^+,\theta^+) \mid x,\theta,a\},$$

for continuous state vector $x \in \mathbb{R}^d$ and discrete state vector $\theta \in \Theta = \{\theta_j = (\theta_{j1},\ldots,\theta_{jk}) : 1 \le j \le N\}$.

The following is the algorithm architecture for the master process:

Algorithm 8.1. *Master Parametric Dynamic Programming with Value Function Iteration for Problems with Multidimensional Continuous and Discrete States*

Initialization. Set up $\hat{V}(x,\theta;b^0)$ and initial guesses of actions a, for all $\theta \in \Theta = \{\theta_j = (\theta_{j1},\ldots,\theta_{jk}) : 1 \le j \le N\}$. Choose the approximation nodes, $X_t = \{x_i^t = (x_{i1}^t,\ldots,x_{id}^t) : 1 \le i \le m_t\}$.

Step 1. Separate the maximization step into N tasks, one task per $\theta_j \in \Theta = \{\theta_j = (\theta_{j1},\ldots,\theta_{jk}) : 1 \le j \le N\}$. Each task contains the parameters b^t, and initial guesses of actions a for all $x_i \in X_t$ with a given θ_j. Then send these tasks to the workers.

Step 2. Wait until all tasks are done by the workers. Then collect the parameters b_j^{t+1} and optimal actions a_{ij}^* from the workers, for $1 \le i \le m_t$ and $1 \le j \le N$.

Step 4. If $t = T-1$ for a finite horizon problem, or if $\|\hat{V}(x,\theta;b^{t+1}) - \hat{V}(x,\theta;b^t)\| < \varepsilon$ for an infinite horizon problem, STOP; else go to step 1.

The following is the algorithm architecture for the workers:

Algorithm 8.2. *Worker Algorithm for Problems with Multidimensional Continuous and Discrete States*

Step 1. Get the parameters b^t and initial guesses of a for one given θ_j from the master.

Step 2. For this given θ_j, compute

$$v_{ij} = \max_{a_{ij} \in D(x_i,\theta_j,t)} \; u(x_i,\theta_j,a_{ij}) + \beta E\{\hat{V}(x_i^+,\theta_j^+;b^t) \mid x_i,\theta_j,a_{ij}\},$$

for each $x_i \in X_t$, $1 \le i \le m_t$.

Step 3. Using an appropriate approximation method, compute the b_j^{t+1}, such that $\hat{V}(x,\theta_j;b_j^{t+1})$ approximates $\{(x_{ij},v_{ij})\colon 1 \le i \le m_t\}$.

Step 4. Send b_j^{t+1} and optimal actions a_{ij}^* for $1 \le i \le m_t$, to the master.

When there are too many nodes for continuous states, i.e., when m_t is large, it will be better to break the task for one θ_j into subtasks. Since partial number of nodes x_i for one given θ_j can not construct the approximation over the full range $[x_{\min}, x_{\max}]$, the workers can not do the step 3 and 4 together with step 1 and 2 in the above algorithm 8.2. If it is quick to compute b_j^{t+1} in the fitting step (e.g., Chebyshev polynomial approximation using Chebyshev regression algorithm), then we can just let the master do the step. Otherwise, the master process has to send two different kind of tasks to the workers. The first kind of tasks will let the workers calculate v_{ij} for some given nodes x_i and one θ_j. Then, after collecting v_{ij} for all nodes x_i and one θ_j, the master process sends the second kind of tasks to the worker for computing b_j^{t+1}. Therefore, the master algorithm will be changed into the following version:

Algorithm 8.3. *Master Parametric Dynamic Programming with Value Function Iteration for Problems with Multidimensional Continuous and Discrete States*

Initialization. Set up $\hat{V}(x, \theta; b^0)$ and initial guesses of actions a, for all $\theta \in \Theta = \{\theta_j = (\theta_{j1}, \ldots, \theta_{jk}) : 1 \le j \le N\}$. Choose the approximation grid, $X_t = \{x_i^t = (x_{i1}^t, \ldots, x_{id}^t)\colon\ i = 1, \ldots, m_t\}$.

Step 1. Separate X_t into D disjoint subsets with almost equal sizes: $X_{t1}, \ldots, X_{tD}$, and separate the maximization step into $N \times D$ first kind of tasks, one task per (X_{td}, θ_j), for $d = 1, \ldots, D$ and $j = 1, \ldots, N$. Each task contains the parameters b^t, and initial guesses of actions a for nodes in X_{td} with a given θ_j. Then send these tasks to to the workers.

Step 2. Wait until all the first kind of tasks are done by the workers. Then collect all v_{ij} and optimal actions a_{ij}^* from the workers, for $1 \le i \le m_t$, $1 \le j \le N$.

Step 3. Separate the fitting step into N second kind of tasks, one task per $\theta_j \in \Theta$. Each task contains v_{ij} for all $1 \le i \le m_t$ and one θ_j. Then send these tasks to the workers.

Step 4. Wait until all the second kind of tasks are done by the workers. Then collect the parameters b_j^{t+1} from the workers, for all $1 \le j \le N$.

Step 5. If $t = T - 1$ for a finite horizon problem, or if $\|\hat{V}(x, \theta; b^{t+1}) - \hat{V}(x, \theta; b^t)\| < \varepsilon$ for an infinite horizon problem, STOP; else go to step 1.

Notice that in the step 4 of the above algorithm, if b_j^{t+1} is quick to be computed, then we could just let the master compute them, and cancel the second kind of tasks for the workers.

The worker algorithm will be separated into two kind of versions for two kinds of tasks:

Algorithm 8.4. *Worker Algorithm for the First Kind of Tasks for Problems with Multidimensional Continuous and Discrete States*

Step 1. Get the parameters b^t and initial guesses of a_{ij} for all nodes $x_i \in X_{td}$'s and one given θ_j from the master process.

Step 2. For this given θ_j, compute

$$v_{ij} = \max_{a_{ij} \in D(x_i, \theta_j, t)} u(x_i, \theta_j, a_{ij}) + \beta E\{\hat{V}(x_i^+, \theta_j^+; b^t) \mid x_i, \theta_j, a_{ij}\},$$

for all $x_i \in X_{td}$.

Step 3. Send v_{ij}^* for these given $x_i \in X_{td}$ and θ_j, to the master process.

Algorithm 8.5. *Worker Algorithm for the Second Kind of Tasks for Problems with Multidimensional Continuous and Discrete States*

Step 1. Get v_{ij} for all nodes $x_i \in X_t$ and one given θ_j from the master process.

Step 2. Using an appropriate approximation method, compute the b_j^{t+1}, such that $\hat{V}(x, \theta_j; b_j^{t+1})$ approximates $\{(x_{ij}, v_{ij})\colon 1 \le i \le m_t\}$.

Step 3. Send b_j^{t+1} to the master process.

We should note that the above algorithms have the parallel block problem, i.e., if all but one task is still in running and it runs very slowly, then we have to wait until it finishes and then go to next value function iteration. But we are able to minimize the parallel block problem by decomposing the big tasks into more smaller tasks, unless the communication between the master and the workers may take too much time, which is unusual in our application.

We use two numerical examples to illustrate the efficiency of the parallel numerical DP algorithms under Condor system.

8.2 Parallelization in Optimal Growth Problems

In chapter 5, we discussed the following DP model for multi-dimensional stochastic optimal growth problems:

$$\begin{aligned} V(k, \theta) \quad = \quad & \max_{k^+, c, l} \quad u(c, l, k^+) + \beta E\{V(k^+, \theta^+) \mid k, \theta, c, l\} \\ & \text{s.t.} \quad k^+ = F(k, l, \theta) - c + \epsilon, \\ & \qquad\quad c > 0, \quad l > 0, \quad k^+ \in [\underline{k}, \bar{k}], \end{aligned}$$

In our first example, we come to see the application of parallelization of numerical DP for the above model. We let k, θ, k^+, θ^+, c, l and ϵ be 4-dimensional vectors, $\beta = 0.8$, $[\underline{k}, \bar{k}] = [0.5, 3.0]^4$, $F(k, l, \theta) = k + \theta A k^\alpha l^{1-\alpha}$ with $\alpha = 0.25$, and $A = (1 - \beta)/(\alpha\beta) = 1$, and

$$u(c, l, k^+) = \sum_{i=1}^{4} \left[c_i^{1-\gamma}/(1-\gamma) - B l_i^{1+\eta}/(1+\eta) \right] + \sum_{i \neq j} \mu_{ij} (k_i^+ - k_j^+)^2,$$

with $\gamma = 2$, $\eta = 1$, $\mu_{ij} = 0$, and $B = (1-\alpha)A^{1-\gamma} = 0.75$.

Here we let $\theta_1^+, \ldots, \theta_4^+$ be independent each other, and assume that the possible values of θ_i, θ_i^+ are

$$a_1 = 0.7,\ a_2 = 0.85,\ a_3 = 0.95,\ a_4 = 1.05,\ a_5 = 1.15,\ a_6 = 1.3,$$

and the probability transition matrix from θ_i to θ_i^+ is

$$P = \begin{bmatrix} 0.75 & 0.25 & 0 & 0 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0.5 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 & 0.75 \end{bmatrix},$$

for each $i = 1, \ldots, 4$. That is,

$$\Pr[\theta^+ = (a_{j_1}, \ldots, a_{j_4}) \mid \theta = (a_{i_1}, \ldots, a_{i_4})] = P_{i_1,j_1} P_{i_2,j_2} P_{i_3,j_3} P_{i_4,j_4},$$

where P_{i_d,j_d} is the (i_d, j_d) element of P, for any $i_d, j_d = 1, \ldots, 6$, $d = 1, \ldots, 4$.

In addition, we assume that $\epsilon_1, \ldots, \epsilon_4$ are i.i.d., and each ϵ_i has 3 discrete values:

$$\delta_1 = -0.001,\ \delta_2 = 0.0,\ \delta_3 = 0.001,$$

while their probabilities are $q_1 = 0.25$, $q_2 = 0.5$ and $q_3 = 0.25$, respectively. That is,

$$\Pr[\epsilon = (\delta_{n_1}, \ldots, \delta_{n_4})] = q_{n_1} q_{n_2} q_{n_3} q_{n_4},$$

for any $n_d = 1, 2, 3$, $d = 1, \ldots, 4$. Moreover, $\epsilon_1, \ldots, \epsilon_4$ are assumed to be independent of $\theta_1^+, \ldots, \theta_4^+$.

Therefore,

$$\begin{aligned} & E\{V(k^+, \theta^+) \mid k, \theta = (a_{i_1}, \ldots, a_{i_4}), c, l\} \\ = & \sum_{n_1,n_2,n_3,n_4=1}^{3} q_{n_1} q_{n_2} q_{n_3} q_{n_4} \sum_{j_1,j_2,j_3,j_4=1}^{6} P_{i_1,j_1} P_{i_2,j_2} P_{i_3,j_3} P_{i_4,j_4} \cdot \\ & V(\hat{k}_1^+ + \delta_{n_1}, \ldots, \hat{k}_4^+ + \delta_{n_4}, a_{j_1}, \ldots, a_{j_4}). \end{aligned}$$

where $\hat{k}_d^+ = F(k_d, l_d, a_{i_d}) - c_d$, for any $i_d = 1, \ldots, 6$, $d = 1, \ldots, 4$.

For this example, we use value function iteration method while the continuous function approximation is complete degree-3 Chebyshev polynomial approximation method with 4^4 Chebyshev nodes for continuous state variables, and the optimizer is NPSOL, and the initial value function is 0 everywhere. Since the number of possible values of θ_i is 6 for $i = 1, \ldots, 4$, the total number of tasks for one value function iteration is $6^4 = 1296$. Under Condor system, we assign 25 workers to do this parallel work by using the master algorithm 8.1 and the worker algorithm 8.2. After running 3 value function

Table 8.1: Parallel efficiency for various number of worker processors

# Worker processors	Parallel efficiency	Average task CPU time (minute)	Total wall clock time (hour)
25	93.56%	21	65
54	93.46%	25	33
100	86.73%	25	19

iterations (VFI), we have the following statistic table:

Wall clock time for all 3 VFIs	235,364 seconds
Wall clock time for 1st VFI	42,034 seconds
Wall clock time for 2nd VFI	103,871 seconds
Wall clock time for 3rd VFI	89,458 seconds
Total time workers were up (alive)	5,354,677 seconds
Total cpu time used by all workers	4,887,384 seconds
Minimum task cpu time	557 seconds
Maximum task cpu time	4,196 seconds
Mean uptime for the workers	214,187 seconds
Standard deviation uptime for the workers	4,175 seconds
Mean cpu time for the workers	195,495 seconds
Standard deviation cpu time for the workers	7,362 seconds
Number of (different) workers	25
Average Number Present Workers	22.75
Overall Parallel Performance	93.56%

Notice that the overall parallel performance is 93.56%, so the parallel efficiency of our parallel numerical DP method is very high for this example.

Moreover, the L^∞ norm of relative change of value functions at 1st, 2nd and 3rd VFI are respectively 8.259158, 1.548019, 0.6354596, which is decreasing. So it means that the value function iteration method is converging.

Table 8.1 gives the parallel efficiency with various number of worker processors for this optimal growth model.

8.3 Parallelization in Dynamic Portfolio Problems

In this section, we come to see the application of parallelization in a multi-period asset allocation problem model. An investor wants to invest his money into n uncorrelated stocks and a bond with a riskless return R_f for one year. The random annual return of stock i, R_i, is assumed to have a log-normal distribution, i.e., $\log(R_i) \sim N(\mu_i - \frac{\sigma_i^2}{2},\ \sigma_i^2)$, for $i = 1, \ldots, n$. The investor would like to consume all the wealth at the end of T years with a power utility function $u(W) = W^{1-\gamma}/(1-\gamma)$. At

the beginning of every year, he has a chance to re-balance his portfolio with a proportional transaction cost rate τ for buying or selling stocks. From section 10.2, we know that the value function of the problem's DP model is

$$V_t(W_t, x_t) = W_t^{1-\gamma} \cdot g_t(x_t),$$

where W_t and x_t are respectively wealth and allocation fractions of stocks rightly before re-balancing at stage $t = 0, 1, \ldots, T$, and

$$\begin{aligned} g_t(x_t) \quad = \quad & \max_{\delta_t^+, \delta_t^-} \quad E\left[\Pi_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1})\right] \\ & \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ & \qquad\; s_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-), \\ & \qquad\; \Pi_{t+1} = e^\top s_{t+1} + R_f(1 - e^\top x_t - m_t), \\ & \qquad\; x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ & \qquad\; \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

for $t = 0, 1, \ldots, T-1$, while $g_T(x) = 1/(1-\gamma)$.

In our numerical examples, we choose the following parameters and methods:

Number of periods (years)	$T = 6$
interest rate	$r = 0.04$
drift of stocks	$\mu_i = 0.07$, $i = 1, \ldots, n$
standard deviation of stocks	$\sigma_i = 0.2$, $i = 1, \ldots, n$
proportional transaction cost ratio	$\tau = 0.002$
terminal utility	$u(W) = W^{1-\gamma}/(1-\gamma)$
quadrature rule	Gauss-Hermite quadrature rule with 5 nodes in one dimension
function approximation method	degree-4 complete Chebyshev polynomial with 5 nodes in one dimension
optimization solver	NPSOL

To realize the importance of parallelization to high-dimensional dynamic portfolio problems, let us at first take a look at the growth rate of run times for various numbers of stocks $n = 3, 4, 5$ and $\gamma = 3$ on a single machine.

In Table 8.2, we see that the growth rate is about 60 times per stock. And we know that it could be larger if we increase the number of Gauss-Hermite nodes or the degree of Chebyshev polynomials.

Now we come to use parallelization algorithms to solve the high-dimensional dynamic portfolio problems. Since there is no discrete state variables in this model, we just need the master algorithm 8.3 and the worker algorithm 8.4. In this example, we choose a dynamic portfolio problem with $n = 6$ stocks and one riskless bond, and we let the relative risk aversion coefficient $\gamma = 3.5$. Since the number

Table 8.2: Run times of NDP for portfolio problems on a single machine

	$n = 3$	$n = 4$	$n = 5$
Run time of 1st VFI	0	6	196
Run time of 2nd VFI	2	91	5,554
Run time of 3rd VFI	1	96	5,926
Run time of 4th VFI	1	62	3,619
Run time of 5th VFI	1	62	3,616
Run time of 6th VFI	1	65	3,764
Total run time of 6 VFIs	6	382	22,675

Note: the time unit is second. They run on a Mac with a 2.2 GHz Intel Core 2 Duo processor and 2 GB memory.
The objective function of 1st VFI is independent of x_T, so it is much faster than other VFIs.

of Chebyshev nodes is 5 for each dimension, the total number of tasks for one value function iteration could be up to $5^n = 15625$, here we set it as 3125 such that each task contains 5 nodes. Under Condor system, we assign 200 workers to do this parallel work.

Wall clock time for all 6 VFIs	5,624 seconds
Wall clock time for 1st VFI	484 seconds
Wall clock time for 2nd VFI	1,451 seconds
Wall clock time for 3rd VFI	1,026 seconds
Wall clock time for 4th VFI	898 seconds
Wall clock time for 5th VFI	886 seconds
Wall clock time for 6th VFI	878 seconds
Total time workers were up (alive)	1,061,396 seconds
Total cpu time used by all workers	892,446 seconds
Minimum task cpu time	2 seconds
Maximum task cpu time	395 seconds
Mean uptime for the workers	5,307 seconds
Standard deviation uptime for the workers	136 seconds
Mean cpu time for the workers	4,462 seconds
Standard deviation cpu time for the workers	355 seconds
Number of (different) workers	200
Average Number Present Workers	189
Overall Parallel Performance	87.2%

Chapter 9

Dynamic Portfolio Problems

There are plenty of applications of dynamic programming (DP) method in finance. In this chapter we present the application in dynamic portfolio problems.

Dynamic portfolio problems are multi-stage asset allocation problems, which are popular in asset management. Assume that n risky assets ("stocks") and/or a riskless asset ("bank account" paying a fixed interest rate r) are available for trading. Adjustments of the assets are made through N stages to maximize the investor's expected utility of terminal wealth. Assume that T is the terminal time and in $[0, T]$ there are $N+1$ stages at times: $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$ with $t_{j+1} - t_j = \Delta t_j$, and the portfolio could be re-allocated on the stages $t_0, \ldots, t_{N-1}$. We always assume that the stocks may be bought or sold in arbitrary amounts (not necessarily integral number of shares, and they will be bounded if there are "no-shorting" or "no-borrowing" constraints.).

If we let the annual risk-free interest rate be r, then the risk-free return over the period (t_j, t_{j+1}) is $R_f = e^{r\Delta t_j}$ for a continuously compound rate, or $R_f = 1 + r\Delta t_j$ for a discretely compound rate. In practice, if borrowing cash is allowed, then the interest rate for borrowing cash from banks should be higher than the interest rate from saving cash in the banks. But here we assume that they are same for simplicity.

Let $S_t = (S_{t1}, \ldots, S_{tn})^\top$ denote the price vector of risky assets at time $t \in [0, T]$, and let their random asset returns be $R^j = (R_1^j, \ldots, R_n^j)^\top$ over the period (t_j, t_{j+1}) for $j = 0, 1, \ldots, N-1$, where $x^\top$ denotes the transpose of a vector x. In a multi-stage portfolio problem, the distribution of the random asset returns are given or derived from discretizing continuous time models. In continuous time world, we often assume that S_t follows a n-dimensional geometric Brownian motion with a constant drift vector $\mu = (\mu_1, \ldots, \mu_n)^\top$ and volatility vector $\sigma = (\sigma_1, \ldots, \sigma_n)^\top$, i.e,

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t,$$

where z_t is an n-dimensional Brownian motion with a correlation matrix Σ. Here $\frac{dS_t}{S_t}$ denotes $(\frac{dS_{t1}}{S_{t1}}, \ldots, \frac{dS_{tn}}{S_{tn}})^\top$, and σdz_t denotes $(\sigma_1 dz_{t1}, \ldots, \sigma_n dz_{tn})^\top$, similar notations are used later.

By discretizing the stochastic process directly, we get

$$\frac{\Delta S_t}{S_t} = \mu\Delta t + \sigma\sqrt{\Delta t}z_1, \tag{9.1}$$

where z_1 is an n-dimensional normal random variable vector while its covariance matrix is its correlation matrix Σ. That is, $S_{t+\Delta t} = RS_t$, while the return

$$R \sim N(1 + \mu\Delta t, (\Lambda\Sigma\Lambda)\Delta t),$$

where Λ =diag$(\sigma_1, \ldots, \sigma_n)$ is a diagonal matrix with diagonal elements $(\sigma_1, \ldots, \sigma_n)$, and $1 + \mu\Delta t$ denotes $(1 + \mu_1\Delta t, \ldots, 1 + \mu_n\Delta t)$. Since the return R usually should be positive, but the normal distribution may have negative values, so we adjust the distribution of R such that the adjusted probability distribution has the following property:

$$\begin{aligned}
\Pr(R_i &< \varepsilon) = 0, \\
\Pr(R_i &= \varepsilon) = \Phi\left(\frac{\varepsilon - (1 + \mu_i\Delta t)}{\sigma_i\sqrt{\Delta t}}\right), \\
\Pr(R_i &< y) = \Phi\left(\frac{y - (1 + \mu_i\Delta t)}{\sigma_i\sqrt{\Delta t}}\right), \text{ if } y > \varepsilon,
\end{aligned}$$

for some small positive constant $\varepsilon < \mu_i\Delta t$ and all $i = 1, \ldots, n$, where $\Phi(x)$ denotes the standard normal cumulative probability distribution. Since usually $\sigma_i \leq 0.25$, then $\Phi\left(\frac{\varepsilon-(1+\mu_i\Delta t)}{\sigma_i\sqrt{\Delta t}}\right) \leq \Phi(-4/\sqrt{\Delta t})$ which is close to 0 when $\Delta t \leq 1$. So this adjustation has a very little effect on the distribution. Later we will say that this R has a truncated multivariate normal distribution.

By applying the Ito's formula, the asset price process S_t has another equivalent stochastic differential equation:

$$d(\log(S_t)) = (\mu - \frac{\sigma^2}{2})dt + \sigma dz_t,$$

where σ^2 denotes $(\sigma_1^2, \ldots, \sigma_n^2)$. By discretizing the stochastic differential equation, we have

$$\Delta(\log(S_t)) = (\mu - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}z_1. \tag{9.2}$$

That is, $S_{t+\Delta t} = RS_t$, while the log-return

$$\log(R) = \Delta(\log(S_t)) \sim N((\mu - \frac{\sigma^2}{2})\Delta t, (\Lambda\Sigma\Lambda)\Delta t).$$

For simplicity, in this chapter we always assume that $\Delta t_j = \Delta t = 1$, and the one-period risk-free return is R_f, and the one-period risky asset return $R^j = R = (R_1, \ldots, R_n)^\top$ has same distribution, for $j = 0, 1, \ldots, N-1$, unless it is specifically defined.

Let W_t denote the wealth at time $t = 0, 1, \ldots, T$, and let the investment fractions in the risky assets after reallocation be $x_t = (x_{t1}, \ldots, x_{tn})^\top$ with $e^\top x_t = \sum_{i=1}^n x_{ti} \leq 1$, where e is the vector with 1 everywhere. Then the investment fraction in the riskless asset is $(1 - e^\top x_t)$. At time $t+1$, the wealth

becomes

$$\begin{aligned} W_{t+1} &= W_t(R_f(1 - e^\top x_t) + R^\top x_t) \\ &= W_t(R_f + (R - R_f)^\top x_t), \end{aligned}$$

for $t = 0, 1, \ldots, T-1$. If there is no riskless asset, then $W_{t+1} = W_t(R^\top x_t)$, while x_t should satisfy $e^\top x_t = 1$.

A simple dynamic portfolio optimization problem is to find optimal portfolio at each stage such that we have a maximal expected utility of the terminal wealth, i.e.,

$$\max \ E[u(W_T)], \tag{9.3}$$

where u is the given utility function (e.g., an exponential utility $u(W) = -\exp(-\lambda W)$ with $\lambda > 0$, which has an absolute risk aversion coefficient $\lambda = -u''(W)/u'(W)$, or a power utility $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma > 0$ and $\gamma \neq 1$, or the log utility $u(W) = \log(W)$ for $\gamma = 1$, which has a relative risk aversion coefficient $\gamma = -Wu''(W)/u'(W)$).

There are several ways to solve this multi-stage portfolio problem: (1) tree method if the random returns are discrete; (2) stochastic programming (SP); (3) dynamic programming (DP).

9.1 Tree Method

For the dynamic portfolio optimization problem 9.3, if we discretize the random returns of n risky assets as $R = R^{(j)} = (R_{1,j}, \ldots, R_{n,j})$ with probability p_j for $1 \leq j \leq m$, then it becomes a tree model:

$$\max \ \sum_{k=1}^{m^T} P_{T,k} u(W_{T,k}),$$

where

$$P_{t+1,k} = P_{t,(k-1)/m+1} p_{(k \bmod m)+1}, \quad P_{0,j} = 1,$$

and

$$W_{t+1,k} = W_{t,(k-1)/m+1}(R_f(1 - e^\top x_{t,(k-1)/m+1}) + \sum_{i=1}^{n} R_{i,(k \bmod m)+1} x_{i,t,(k-1)/m+1}),$$

for $1 \leq k \leq m^{t+1}$ and $0 \leq t < T$.

The disadvantage of tree method is that when m or T is large, the problem size will be exponentially increased, such that it will be a big challenge for an optimizer to get accurate solution.

9.2 Stochastic Programming Model

For the dynamic portfolio optimization problem 9.3, the stochastic programming model is

$$\max \ \frac{1}{M}\sum_{k=1}^{M} u(W_{T,k}),$$

where $M = \prod_{t=0}^{T-1} m_t$, and m_t is the number of scenarios in the period t given the previous scenario, and

$$W_{t+1,k} = W_{t,(k-1)/m_t+1}(R_f(1 - e^\top x_{t,(k-1)/m_t+1}) + \sum_{i=1}^{n} R_{i,t,k} x_{i,t,(k-1)/m_t+1}),$$

where $(R_{1,t,k}, \ldots, R_{n,t,k})$ is the k-th sample of random return over the period $(t, t+1)$, for $1 \le k \le \prod_{i=0}^{t} m_i$ and $0 \le t \le T-1$.

In order to get a good estimation of optimal asset allocation, stochastic programming method often has to generate or simulate a huge amount of scenarios $O(M)$, while the number of unknowns are $O(n\prod_{t=0}^{T-2} m_t)$, such that it will be a big challenge for an optimizer to get good solution. Often stochastic programming has to use some decomposition techniques (such as Benders decomposition, Dantzig-Wolfe decomposition, etc.) and variance reduction techniques. See Infanger [25] and Collumb [9].

The disadvantage of stochastic programming is the huge size of optimization problem and uncertainty of scenarios. A typical scenario tree was represented by $(50 \times 50 \times 40)$ scenarios leading to $100,000$ scenarios.

9.3 Dynamic Programming Model

For the dynamic portfolio optimization problem 9.3, the DP model is

$$\begin{aligned} V_t(W_t) &= \max \ E\left[V_{t+1}(W_{t+1}) \mid W_t\right] \\ &= \max_{x_t} \ E\left[V_{t+1}\left(W_t(R_f(1 - e^\top x_t) + R^\top x_t\right)\right], \end{aligned} \tag{9.4}$$

for $0 \le t < T$, while $V_T(W) = u(W)$. The function $V_t(W)$ is called as the value function at time t, which is also called as indirect utility function or derived utility function. If there is no riskless asset, then $W_{t+1} = W_t(R^\top x_t)$ and we just need to cancel the R_f term in the Bellman equation, while x_t should satisfy $e^\top x_t = 1$.

Many researchers attacked the dynamic portfolio problems by using the DP method. Most research discussed the problems with only 2 or 3 assets for standard utility functions (CARA, CRRA, or HARA). Gupta [24] studied the problem with behavior utilities. Wang [53] applied suboptimal solution methods based on approximate value iteration to solve the dynamic portfolio problem without transaction costs. His primary innovation is the use of mean-variance portfolio selection methods to find suboptimal portfolios instead of optimal portfolios. Unfortunately, the solutions from this method

cannot guarantee to be close to the optimal solutions. Moreover, since he did not apply efficient, accurate and stable approximation method and integration method in the value function iteration, such that his method is inefficient and the solutions may be inaccurate.

9.4 Numerical Dynamic Programming

If we use some determined quadrature formulas to estimate the expectation in the objective function of the DP model, we call it as the numerical DP method.

9.4.1 Integration for Truncated Normal Random Returns

In the above DP model for discrete-time optimal asset allocation problem, we know the approximation for the value functions is one-dimensional problem, the optimization problem has a size $O(n)$, but the integration operator of the expected value at next stage, $E[V_{t+1}(W_{t+1}) \mid W_t]$, is a n-dimensional problem in the standard way which suffers the well-known "curse of dimensionality", as there are n random variables: $R_1, \ldots, R_n$.

However, this "curse of dimensionality" can be solved, if $R = (R_1, \cdots, R_n)^\top \in \mathbb{R}^n$ is assumed to be a random vector with a truncated multivariate normal distribution over one period $(t, t+\Delta t)$. For example, if we assume that the risky asset price vector S_t is an n-dimensional geometric Brownian motion:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t,$$

where $\mu = (\mu_1, \cdots, \mu_n)^\top$ is the drift, $\sigma = (\sigma_1, \ldots, \sigma_n)$ is the volatility, z_t is an n-dimensional Brownian motion with a correlation matrix Σ, then we can discretize the stochastic process

$$\frac{\Delta S_t}{S_t} = \mu \Delta t + \sigma \sqrt{\Delta t} z_1,$$

such that the asset return over the period $[t, t+\Delta t)$, $R = S_{t+\Delta t}/S_t$, is normal with mean $(1+\mu\Delta t)$ and variance matrix $(\Lambda\Sigma\Lambda)\Delta t$, where $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_n)$. Then we adjust R such that $P(R_i < \varepsilon) = 0$ for some small positive number ε and all $i = 1, \ldots, n$.

In the following, we assume that $\Delta t = 1$ for simplicity, and R is a random vector with a truncated multivariate normal distribution with mean $1+\mu$ and variance matrix $\Lambda\Sigma\Lambda$, where $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_n)$ is the standard deviation vector of R, and Σ is the correlation matrix of R. Let $R_f = 1 + r$ be the riskfree return.

Let the risky asset allocation fraction $x_t = (x_{t1}, \ldots, x_{tn})^\top$, and let W_t be the wealth at stage t, then

$$W_{t+1} = W_t((R - R_f)^\top x_t + R_f),$$

Since R is a random vector with a truncated multivariate normal distribution, and W_{t+1} is a linear combination of R_i, from the property of normal random vector, we know that W_{t+1} is close to be

normal with mean $W_t(1+\hat{\mu}(x_t))$ and standard deviation $\hat{\sigma}(x_t)W_t$, where

$$\begin{cases} \hat{\mu}(x_t) = (\mu - r)^\top x_t + r, \\ \hat{\sigma}(x_t) = \sqrt{x_t^\top \Lambda\Sigma\Lambda x_t}. \end{cases} \tag{9.5}$$

If there is no riskless asset, then $\hat{\mu}(x_t) = \mu^\top x_t$, while x_t should satisfy $e^\top x_t = 1$.

Thus $E[V_{t+1}(W_{t+1}) \mid W_t]$ is transformed into a univariate integration approximately and we can apply one-dimensional Gauss-Hermite quadrature to estimate it. The numerical DP model is

$$V_t(W_t) = \pi^{-\frac{1}{2}} \max_{x_t} \sum_{i=1}^{m} \omega_i V_{t+1}(W_t(1+\hat{\mu}(x_t) + \sqrt{2}\hat{\sigma}(x_t)q_i)),$$

where the ω_i and q_i are the Gauss-Hermite quadrature weights and nodes over $(-\infty, \infty)$, for $i = 1, \ldots, m$. This model is called as numerical DP model with normal random return of portfolio.

When m is big enough, $1 + \hat{\mu}(x_t) + \sqrt{2}\hat{\sigma}(x_t)q_i$ could be less than 0 such that W_{t+1} may be less than 0. But if the terminal utility function is a CRRA utility, then we should have to let $W_t > 0$. One way to solve this drawback is to choose an appropriate m, which has no problem usually when $\Delta t \leq 1$.

If we do not allow shorting stocks or borrowing money, then we just need to add the constraints: $x_t \geq 0$ and $1 - e^\top x_t \geq 0$.

9.4.2 Nonlinear Change of Variables for States

Numerically, to approximate the value function, we need to have a lower bound and an upper bound at each stage for the state variable W. When we do not allow shorting stocks or borrowing cash, the lower bound of W must be bigger than 0. In the asset allocation problem, we often add the mean return constraint $\hat{\mu}(x_t) \geq \underline{\mu}$ and the standard deviation constraint $\hat{\sigma}(x_t) \leq \bar{\sigma}$ for some given $\underline{\mu}$ and $\bar{\sigma}$. For simplicity, here we let $\Delta t = 1$.

If the range of W_t at the time t is $[L_t, U_t]$, and we use the m Gauss-Hermite quadrature nodes: $q_1, \ldots, q_m$ in the numerical DP model with normal random return of portfolio, then the lower bound of W_{t+1} is

$$\begin{aligned} L_{t+1} \quad = \quad & L_t \cdot \min_{x_t} \ (1 + \hat{\mu}(x_t) + \sqrt{2}\hat{\sigma}(x_t) \min_{1\leq i\leq m} (q_i)) \\ & \text{s.t.} \quad \hat{\mu}(x_t) \geq \underline{\mu},\ \hat{\sigma}(x_t) \leq \bar{\sigma}, \\ & \qquad x_t \geq 0,\ 1 - e^\top x_t \geq 0, \end{aligned}$$

and the upper bound of W_{t+1} is

$$\begin{aligned} U_{t+1} \quad = \quad & U_t \cdot \max_{x_t} \ (1 + \hat{\mu}(x_t) + \sqrt{2}\hat{\sigma}(x_t) \max_{1\leq i\leq m} (q_i)) \\ & \text{s.t.} \quad \hat{\mu}(x_t) \geq \underline{\mu},\ \hat{\sigma}(x_t) \leq \bar{\sigma}, \\ & \qquad x_t \geq 0,\ 1 - e^\top x_t \geq 0. \end{aligned}$$

Note that the number of Gauss-Hermite quadrature nodes m should not be chosen too large such that $L_{t+1} \leq 0$.

By observing the formulas of L_{t+1} and U_{t+1}, we know that the range of W is exponentially expanding over the time. For example, if we have only one stock asset and its one-period return has a normal distribution with mean $(1+\mu)$ and standard deviation σ, then $L_{t+1} = L_t(1+\mu+\sqrt{2}\sigma \min(q_i)) = L_0(1+\mu+\sqrt{2}\sigma \min(q_i))^{t+1}$, and $U_{t+1} = U_t(1+\mu+\sqrt{2}\sigma \max(q_i)) = U_0(1+\mu+\sqrt{2}\sigma \max(q_i))^{t+1}$. When we set $L_0 = 0.9$ and $U_0 = 1.1$, $\mu = 0.07$, $\sigma = 0.2$ and the number of Gauss-Hermite nodes $m = 10$, we have

$$\begin{aligned}
[L_0, U_0] &= [0.9, 1.1], \\
[L_1, U_1] &= [0.088, 2.246], \\
[L_2, U_2] &= [0.00866, 4.586], \\
[L_3, U_3] &= [0.00085, 9.36], \\
[L_4, U_4] &= [8 \times 10^{-5}, 19], \\
[L_5, U_5] &= [8 \times 10^{-6}, 39].
\end{aligned}$$

If we choose the utility function $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma > 1$ (γ could be dependent on W), then $u(W) \downarrow -\infty$ as $W \downarrow 0$, and $u(W) \to 0$ as $W \to +\infty$. Then the value function at each stage will be steep at the nearby of small lower bound and flat at the nearby of large upper bound on its domain. So it will be hard to approximate the value function well directly on the state variable W at a wide range with small lower bound and large upper bound. Thus, to approximate the value function accurately, approximation nodes should be assigned in such a way that they are denser when they are closer to the small lower bound, while they are sparser when they are closer to the nearby of large upper bound.

To solve the above problem, we could change the state variable from W to $w = \log(W)$. Thus the range of w is just linearly expanding over the time. For the above example, the ranges of w over the times are

$$\begin{aligned}
[l_0, u_0] &= [-0.1, 0.095], \\
[l_1, u_1] &= [-2.4, 0.8], \\
[l_2, u_2] &= [-4.7, 1.5], \\
[l_3, u_3] &= [-7.1, 2.2], \\
[l_4, u_4] &= [-9.4, 2.95], \\
[l_5, u_5] &= [-11.7, 3.66],
\end{aligned}$$

and the new value function is $v(w) = v(\log(W)) = V(W)$.

Besides the nonlinear change of state variable from W to $w = \log(W)$, we could also apply the change of value function form from $V(W)$ to $\log(-v(w))$ (here we assume that the utility function

$u(W) < 0$ for all $W > 0$, such as $u(W) = W^{1-\gamma}/(1-\gamma)$ for $\gamma > 1$). That is, if we approximate the value function using $V(W) = \sum_{i=1}^{n} c_i\phi_i(W)$, then now we use

$$\log(-v(w)) = \sum_{i=1}^{n} c_i\phi_i(w),$$

i.e., $v(w) = -\exp(\sum_{i=1}^{n} c_i\phi_i(w))$.

When we allow shorting stocks or borrowing cash, there may be no bound for next stage's wealth. But for the lower bound, it is natural to impose the constraint $W_{t+1} > 0$. So the range of W will be $(0, \infty)$ in every stage. Now we could use a rational change of variable from W to $w = (W-a)/(W+a)$ for some $a > 0$. Thus the range of w is $(-1, 1)$ in every stage. This rational change is a one-to-one mapping while the inverse function is $W = a(1+w)/(1-w)$. Using this transformation, the value function will be good at the nearby of $W = a$.

9.4.3 Integration for Log Normal Random Returns

When we assume that R is normal, it makes that the asset price could be less than 0. So we often assume that the asset log returns over $[t, t+\Delta t)$, $\log(R) = (\log(R_1), \cdots, \log(R_n))^\top \in \mathbb{R}^n$ ("log" denotes the natural logarithm), has a multivariate normal random distribution $N((\mu - \frac{\sigma^2}{2})\Delta t, (\Lambda\Sigma\Lambda)\Delta t)$ in $\mathbb{R}^n$, where $\mu = (\mu_1, \cdots, \mu_n)^\top$ is the drift, $\sigma = (\sigma_1, \cdots, \sigma_n)^\top$ is the volatility, and Σ is the correlation matrix of the asset log returns, and $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_n)$.

Since the correlation matrix is positive definite, we can apply Cholesky factorization to Σ such that $\Sigma = LL^\top$, where $L = (L_{ij})_{n\times n}$ is a lower triangular matrix. Then

$$\begin{aligned}
\log(R_1) &= (\mu_1 - \frac{\sigma_1^2}{2})\Delta t + (L_{11}z_1)\sigma_1\sqrt{\Delta t}, \\
\log(R_2) &= (\mu_2 - \frac{\sigma_2^2}{2})\Delta t + (L_{21}z_1 + L_{22}z_2)\sigma_2\sqrt{\Delta t}, \\
&\vdots \\
\log(R_n) &= (\mu_n - \frac{\sigma_n^2}{2})\Delta t + (L_{n1}z_1 + L_{n2}z_2 + \cdots + L_{nn}z_n)\sigma_n\sqrt{\Delta t},
\end{aligned}$$

where z_i are independent standard normal random variables. So

$$R_i = \exp\left((\mu_i - \frac{\sigma_i^2}{2})\Delta t + \sigma_i\sqrt{\Delta t}\sum_{j=1}^{i} L_{ij}z_j\right).$$

For simplicity, we let $\Delta t = 1$ in the following.

Let the risky asset allocation fraction $x_t = (x_{t1}, \ldots, x_{tn})^\top$, and let W_t be the wealth at stage t,

then

$$\begin{aligned} W_{t+1} &= W_t(R_f(1-e^\top x_t)+R^\top x_t) \\ &= W_t(R_f(1-e^\top x_t)+\sum_{i=1}^{n}\exp\left((\mu_i-\frac{\sigma_i^2}{2})+\sigma_i\sum_{j=1}^{i}L_{ij}z_j\right)x_{ti}. \end{aligned}$$

Therefore, we can apply product Gauss-Hermite quadrature to estimate the conditional expectation of $V_{t+1}(W_{t+1})$ while W_t, x_t are given. The numerical DP model is

$$\begin{aligned} V_t(W_t) = \pi^{-\frac{n}{2}} \quad \max \quad & \sum_{k_1,\ldots,k_n=1}^{m}\omega_{k_1}\cdots\omega_{k_n}V_{t+1}(W_t(R_f(1-e^\top x_t)+ \\ & \sum_{i=1}^{n}\exp\left(\mu_i-\frac{\sigma_i^2}{2}+\sqrt{2}\sigma_i\sum_{j=1}^{i}q_{k_j}L_{ij}\right)x_{ti})), \end{aligned}$$

where the ω_k and q_k are the Gauss-Hermite quadrature weights and nodes over $(-\infty,\infty)$, for $k = 1,\ldots,m$. If there is no riskless asset, then we just need to cancel the R_f term in the formula for $V_t(W_t)$, and add a constraint $e^\top x_t = 1$.

Similarly with the normal random return case, if we do not allow shorting, we can also use the log transformation of variable $w = \log(W)$ and the transformation of value function

$$\log(-v(w)) = \log(-V(W)) = \sum_{i=1}^{n} c_i\phi_i(w).$$

And if we allow shorting stocks or borrowing cash, we just need to add the constraints: $x_t \geq 0$ and $1 - e^\top x_t \geq 0$, and we can also use the rational change of variable $w = \frac{W-a}{W+a}$ for some $a > 0$.

9.4.4 CARA Utility and Portfolio Problems

In portfolio problems, there are a lot of alternatives for utility functions, such as CRRA utility ($u(W) = W^{1-\gamma}/(1-\gamma)$ for $\gamma > 0$ and $\gamma \neq 1$, or $u(W) = \log(W)$), CARA utility ($u(W) = -\exp(-\lambda W)/\lambda$ for $\lambda > 0$), and HARA (hyperbolic absolute risk aversion) utility ($u(W)$ satisfying $-u'(W)/u''(W) = \eta + W/\gamma$), and so on. In fact, the HARA utility can be expressed explicitly as

$$u(W) = \begin{cases} a(W-c)^{1-\gamma}/(1-\gamma)+b, & \text{if } \gamma \neq 1,\infty, \\ a\log(W-c)+b, & \text{if } \gamma = 1, \\ -a\exp(-\lambda W)/\lambda + b, & \text{if } \gamma = \infty, \end{cases}$$

for some constants $a > 0$, b and c (Usually we just let $a = 1$ and $b = 0$). So the CRRA utility class and the CARA utility class are just subsets of the HARA utility class. See Merton [38], Rubinstein [46], etc.

When the terminal utility function is a CARA utility $u(W) = -e^{-\lambda W}$ with a constant absolute risk aversion coefficient $\lambda > 0$, the optimal solutions and value functions can be solved explicitly for

the DP model (9.4) if the random return is assumed to be normal.

Assume that n stocks are available in the portfolio, and these stocks have a normal return vector $R \sim N(1+\mu, \Lambda\Sigma\Lambda)$, where $\mu = (\mu_1, \ldots, \mu_n)^\top$ is the mean net return, Λ =diag$(\sigma_1, \ldots, \sigma_n)$ is the diagonalized matrix of standard deviation $\sigma = (\sigma_1, \ldots, \sigma_n)^\top$, and Σ is the correlation matrix of stocks. If we invest $x = (x_1, \ldots, x_n)^\top$ fractions of initial wealth W_0 into these stocks, then $x^\top R$ is also normal with mean $(1 + x^\top\mu)$ and variance $x^\top\Lambda\Sigma\Lambda x$. Similarly with equation 4.2 for the case with one stock and one bond, if we also invest the remaining $(1 - e'x)W_0$ into a bond with a riskless return $R_f = 1 + r$, then the expected exponential utility of wealth of the portfolio at the end of one year is

$$\begin{aligned} U(x, W_0) &= (2\pi(x^\top\Lambda\Sigma\Lambda x))^{-1/2} \int_{\infty}^{+\infty} -e^{-\lambda W_0(1+z+(1-e^\top x)r)} e^{-(z-x^\top\mu)^2/(2(x^\top\Lambda\Sigma\Lambda x))} dz \\ &= -\exp(-\lambda W_0(1 + r + x^\top(\mu - r)) + (\lambda W_0)^2(x^\top\Lambda\Sigma\Lambda x)/2). \end{aligned}$$

Thus, the optimal allocation of this one-period portfolio problem satisfies the following KKT conditions:

$$\mu - r = \lambda W_0\Lambda\Sigma\Lambda x.$$

That is, the optimal allocation fraction vector for stocks is

$$x^* = (\Lambda\Sigma\Lambda)^{-1}(\mu - r)/(\lambda W_0).$$

So x^*W_0 is a constant vector, meaning that the optimal amount vector of dollars invested in stocks is a constant vector independent of the total amount of wealth.

Moreover, the value function at initial stage for this one-period portfolio problem is

$$V_0(W_0) = U(x^*, W_0) = -\alpha e^{-\lambda(1+r)W_0},$$

where

$$\alpha = \exp(-(\mu - r)^\top(\Lambda\Sigma\Lambda)^{-1}(\mu - r)/2).$$

Recursively, we can derive that for the T-period DP model (9.4),

$$V_t(W_t) = -\alpha^{T-t} \exp(-\lambda(1 + r)^{T-t} W_t),$$

and the optimal allocation fraction vector for stocks is

$$x_t^* = (\Lambda\Sigma\Lambda)^{-1}(\mu - r)/(\lambda(1 + r)^{T-t-1} W_t),$$

for $t = 0, 1, \ldots, T - 1$.

9.4.5 CRRA Utility and Portfolio Problems

The most common utility function in the economics and finance is CRRA utility. Let us assume that the terminal utility function has the form yielding a constant relative risk aversion (CRRA) coefficient $\gamma > 0$, i.e., the utility function is $u(W) = \frac{W^{1-\gamma}}{1-\gamma}$ for $\gamma > 0$ and $\gamma \neq 1$, or $u(W) = \log(W)$ for $\gamma = 1$. Let $R = (R_1, \ldots, R_n)^\top$ be the random return of risky assets, and R_f is the return of the riskless asset, over a period. Let the allocation fractions for the risky assets be $x = (x_1, \ldots, x_n)^\top$, and let $x_i \geq l_i$ for some constant l_i, $i = 1, \ldots, n$.

Assume that $u(W) = \frac{W^{1-\gamma}}{1-\gamma}$ for $\gamma > 0$ and $\gamma \neq 1$, then we can show that $V_t(W) = \beta^{T-t} W^{1-\gamma}/(1-\gamma)$ for some constant $\beta > 0$. Let the value function at stage $t+1$ be assumed to be $V_{t+1}(W) = \alpha_{t+1} W^{1-\gamma}/(1-\gamma)$, where α_{t+1} is a given constant.

In the following, we come to show that $V_t(W) = \alpha_t W^{1-\gamma}/(1-\gamma)$ for some constant α_t and the optimal x_t is independent of W.

Under the constraints $x_t \geq l$ and $1 - e^\top x_t \geq 0$, the optimal x_t satisfies the KKT equations

$$
\begin{aligned}
& E\left[\left(R_f\left(1 - e^\top x_t\right) + R^\top x_t\right)^{-\gamma}(R_k - R_f)\right] - \lambda_k + \varsigma = 0, \quad k = 1, \ldots, n, \\
& \lambda_k (x_{tk} - l_k) = 0, \quad k = 1, \ldots, n, \\
& \varsigma\left(1 - e^\top x_t\right) = 0,
\end{aligned}
$$

and an implicit constraint $R_f\left(1 - e^\top x_t\right) + R^\top x_t > 0$. Without loss of generality, let us assume that $x_{ti} > l_i$ for $i = 1, \ldots, m$, and $x_{tj} = l_j$ for $j = m+1, \ldots, n$, then $\lambda_1 = \lambda_2 = \cdots = \lambda_m = 0$.

Now if $1 - \sum_{i=1}^k x_i > 0$, then $\varsigma = 0$, this implies that

$$
E\left[\left(R_f\left(1 - \sum_{i=1}^m x_{ti}\right) + \sum_{i=1}^m R_i x_{ti}\right)^{-\gamma}(R_k - R_f)\right] = 0, \tag{9.6}
$$

for $k = 1, \ldots, m$, which is a system of m equations with m unknowns.

Otherwise, we have $1 - \sum_{i=1}^m x_i = 0$, then

$$
E\left[\left(\sum_{i=1}^m R_i x_i\right)^{-\gamma}(R_k - R_f)\right] + \varsigma = 0,
$$

for $k = 1, \ldots, m$. This follows that

$$
E\left[\left(\sum_{i=1}^m R_i x_i\right)^{-\gamma}(R_k - R_m)\right] = 0, \tag{9.7}
$$

for $k = 1, \ldots, m$, which is also a system of m equations with m unknowns..

So we showed that under the constraints $x_t \geq l$ and $1 - e^\top x_t \geq 0$, the optimal allocation fractions are still independent of wealth W_t and α_{t+1}, no matter which distribution R has, if we assume that $V_{t+1}(W) = \alpha_{t+1} W^{1-\gamma}/(1-\gamma)$.

If there is no riskless asset or we cancel the constraint $1-e^\top x_t \geq 0$, we can use a similar reasoning to derive the same conclusion that the optimal allocation fractions are independent of wealth W_t.

Moreover, the value function at stage t is

$$\begin{aligned} V_t(W) &= E\left[\alpha_{t+1}(W\left(R_f(1-e^\top x_t^*)+R^\top x_t^*\right))^{1-\gamma}/(1-\gamma)\right] \\ &= \alpha_{t+1}W^{1-\gamma}E\left[\left(R_f\left(1-e^\top x_t^*\right)+R^\top x_t^*\right)^{1-\gamma}/(1-\gamma)\right] \\ &= (\alpha_{t+1}\beta_t)W^{1-\gamma}/(1-\gamma) \\ &= \alpha_t W^{1-\gamma}/(1-\gamma), \end{aligned}$$

where $\beta_t = E\left[\left(R_f\left(1-e^\top x_t^*\right)+R^\top x_t^*\right)^{1-\gamma}\right]$ is a constant as the optimal allocation fractions x_t^* are independent of W, so that $\alpha_t = \alpha_{t+\Delta t}\beta_t$ is also a constant.

From the above discussion, if the value function at the end stage T is $V_T(W) = u(W) = \frac{W^{1-\gamma}}{1-\gamma}$ with $\gamma > 0$ and $\gamma \neq 1$, then at every stage $t = 0, 1, \ldots, T-1$, we have

$$V_t(W) = \left(\prod_{j=t}^{T-1}\beta_j\right)W^{1-\gamma}/(1-\gamma).$$

Moreover, if R_f and the distribution of R are independent of t, then the optimal allocation fractions x_t^* are constant along with W and t, and the value function has the form

$$V_t(W) = \beta^{T-t}W^{1-\gamma}/(1-\gamma),$$

for any $t = 0, 1, \ldots, T$, while

$$\beta = E\left[\left(R_f\left(1-e^\top x^*\right)+R^\top x^*\right)^{1-\gamma}\right]$$

is a constant, where R_f is one-period riskless return and R is one-period risky asset return. If there is no riskless asset, then we just need to cancel the R_f term in the formula for β, while x^* should satisfy $e^\top x^* = 1$.

When the terminal utility function is $u(W) = \log(W)$ for the constant relative risk aversion coefficient $\gamma = 1$, the above KKT conditions still hold, thus if we assume that $V_{t+1}(W) = \log(W)+\xi_{t+1}$ with a non-random ξ_{t+1} independent of W, then we have

$$V_t(W) = \log(W) + (\eta_t + \xi_{t+1}) = \log(W) + \xi_t,$$

with $\eta_t = E\left[\log\left(R_f\left(1-e^\top x_t^*\right)+R^\top x_t^*\right)\right]$. Thus, since $V_T(W) = u(W) = \log(W)$, we have

$$V_t(W) = \log(W) + \sum_{j=t}^{T-1}\eta_j.$$

Moreover, if R_f and the distribution of R are independent of t, then the optimal allocation fractions

x_t^* are constant along with W and t, and the value function has the form

$$V_t(W) = \log(W) + (T-t)\eta,$$

for $t = 0, 1, \ldots, T-1$, while

$$\eta = E\left[\log\left(R_f\left(1 - e^\top x^*\right) + R^\top x^*\right)\right]$$

is a constant, where R_f is one-period riskless return and R is one-period risky asset return. If there is no riskless asset, then we just need to cancel the R_f term in the formula for η, while x^* should satisfy $e^\top x^* = 1$.

Since the above results are independent of the distribution of R, it is simple to extend the above results to the case when the re-allocation times $0 = t_0 < t_1 < \cdots < t_N = T$ are not equally spaced, and/or R_f and the distribution of R change at each stage t_i So we showed that myopia is optimal for CRRA utility. That is, the investor bases each period's decision on that period's wealth and probability distribution of random returns, while the objective is to maximize the expected utility of wealth at the end of that period, where the utility function is the same CRRA utility in the multiperiod problem. This result was also showed in Mossin [39], Samuelson [48], and Merton [37, 38]. Here we extended the conclusion to the case without riskless asset and the cases with some shorting or borrowing constraints.

The above results can be extended to the continuous-time investment problem with no-shorting and no-borrowing constraints and/or no-riskless-asset case, and the explicit solutions for the optimal portfolio rules and explicit value function can be derived, see Cai [8].

9.4.6 Numerical Examples

In order to determine that our numerical DP is robust and the portfolio policy obtained from it is accurate, we use a typical investment example to illustrate it. An investor has a current wealth of one million dollars and plans to invest it in next 20 years. The investor wants to invest in the portfolio of 5 assets: US stocks, International stocks, Corporate bonds, Government bonds, and Cash. The risk-free interest rate of the Cash asset is $r = 0.05$. The continuous time stochastic process of these 4 risky asset prices $S_t = (S_{t1}, \ldots, S_{t4})$ is assumed to be an 4-dimensional geometric Brownian motion:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t,$$

where z_t is an 4-dimensional Brownian motion with a correlation matrix Σ. The drift μ, standard deviation σ and Σ are given in the following table (the data are from Infanger [26]).

drifts μ and standard deviations σ

	US stocks	Int stocks	Corp bnd	Gvnt bnd
Mean	0.1080	0.1037	0.09	0.079
Std	0.1572	0.1675	0.0657	0.0489

correlation matrix Σ

	US stocks	Int stocks	Corp bnd	Gvnt bnd
US stocks	1.00	0.601	0.247	0.062
Int stocks	0.601	1.00	0.125	0.027
Corp bnd	0.247	0.125	1.00	0.883
Gvnt bnd	0.062	0.027	0.883	1.00

We assume that there is no transaction to reallocate portfolio, and shorting stocks or borrowing cash is not allowed. At the beginning of each year, the investor has a chance to re-allocate the portfolio. So we choose $\Delta t = 1$ year, and the investor problem becomes a 20-period portfolio optimization problem. All numerical solutions in this section are given by AMPL programs and KNITRO optimizer.

Example for Normal Return

In this example, we assume that the investor has a constant relative risk aversion coefficient $\gamma = 2$, i.e., the terminal utility function is $u(W) = -W^{-1}$. Since S_t is a geometric Brownian motion, by discretizing it directly as the equation (9.1), we know that one-year random return vector R could be assumed to have a truncated multivariate normal distribution with mean $(1 + \mu)$ and variance matrix $\Lambda\Sigma\Lambda$, where Λ =diag$(\sigma_1, \ldots, \sigma_4)$. Let x_t be the portfolio allocation at stage t, then from the discussion in section 9.4.1, we know that W_{t+1} is close to be normal with mean $(1 + \hat{\mu}(x_t))$ and standard deviation $\hat{\sigma}(x_t)$, where $\hat{\mu}(x_t)$ and $\hat{\sigma}(x_t)$ are given in the equation (9.5).

Let the range of initial wealth be $[0.9, 1.1]$ millions. Let us require that $\hat{\mu}(x_t) \geq \underline{\mu} = 0.09$, and $\hat{\sigma}(x_t) \leq \bar{\sigma} = 0.1$. If we use the 9-node Gauss-Hermite quadrature formula, then the range of wealth at the end time $T = 20$ could be chosen as $[0.000115, 7211]$ millions. By use of a degree-9 Chebyshev polynomial interpolation method and the techniques discussed in section 9.4.1 and 9.4.2, we found that at any stage $t = 0, 1, \ldots, 19$, the computed optimal risky asset allocation fraction x^* are almost constant for any wealth in its given range at stage t: $x^* \approx (0.3858, 0.1370, 0.4772, 0)$, while the bound of their numerical oscillation is only about 0.00001.

Moreover, the value function at time $t = 0$ is approximated by

$$\hat{V}_0(W) \approx -\exp(-1.75032 - \log(W)) = -0.173718W^{-1},$$

which is very close to the true value function

$$V_0(W) = -\beta^{20}W^{-1} \approx 0.173709W^{-1},$$

Table 9.1: Optimal allocation under power utility and log-normal returns, $t = 19$

Wealth	US stocks	Int stocks	Corp bnd	Gvnt bnd	Cash	Value
0.0002	1	0	0	0	0	−4565.8
0.001	0.3592	0.1337	0.5071	0.0000	0.0000	−913.16
0.01	0.3592	0.1337	0.5071	0.0000	0.0000	−91.316
0.1	0.3592	0.1337	0.5071	0.0000	0.0000	−9.1316
1	0.3592	0.1337	0.5071	0.0000	0.0000	−0.91316
10	0.3592	0.1337	0.5071	0.0000	0.0000	−0.09132
100	0.3592	0.1337	0.5071	0.0000	0.0000	−0.00913
1000	0.3592	0.1337	0.5071	0.0000	0.0000	−0.00091
8000	0.0088	0.0087	0.0342	0.0792	0.8656	−0.00012

as

$$\beta = E\left[\left(R_f(1 - e^\top x^*) + R^\top x^*\right)^{-1}\right] \approx 0.9162018,$$

where $R_f = 1 + r = 1.05$.

In addition, the mean return of the computed optimal portfolio is $\hat{\mu}(x^*) \approx 0.0988 > \underline{\mu}$, and its standard deviation is $\hat{\sigma}(x^*) \approx 0.0893 < \bar{\sigma}$. So we verified that the constraints for $\hat{\mu}(x_t) \geq \underline{\mu} = 0.09$ and $\hat{\sigma}(x_t) \leq \bar{\sigma} = 0.1$ hold and are not binding.

Example for Log-Normal Return

In this example, we still assume that the investor has a constant relative risk aversion coefficient $\gamma = 2$, i.e., the terminal utility function is $u(W) = -W^{-1}$. However, by discretizing $\log(S_t)$ as the equation (9.2), we assume that one-year random return vector R has a log-normal distribution with mean $(\mu - \frac{\sigma^2}{2})$ and covariance matrix $\Lambda\Sigma\Lambda$, where $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_4)$.

Let the range of initial wealth be $[0.9, 1.1]$ millions and we use the product Gauss-Hermite quadrature formula with 5-node in each dimension. In order to avoid overflow or underflow, we assume that the value at $W_t \leq \exp(-9)$ is equal to the value at $W_t = \exp(-9)$, and the value at $W_t \geq \exp(9)$ is equal to the value at $W_t = \exp(9)$. Thus the range of wealth at any time t could be a subset of $[\exp(-9), \exp(9)] = [0.00012, 8103]$ millions. By use of a degree-9 Chebyshev polynomial interpolation method and the techniques discussed in section 9.4.3 and 9.4.2, we found that at any stage $t = 0, 1, \ldots, 19$, for any wealth in $[0.1, 10]$ millions, the computed optimal asset allocation fraction x^* is close to a constant vector: $x^* \approx (0.36, 0.134, 0.51, 0)$, while the bound of their numerical oscillation is about 0.001. Table 9.1, 9.2 and 9.3 respectively give optimal asset allocation fractions at time $t = 19$, $t = 9$, $t = 0$.

Table 9.2: Optimal allocation under power utility and log-normal returns, $t = 9$

Wealth	US stocks	Int stocks	Corp bnd	Gvnt bnd	Cash	Value
0.005	0.3596	0.1337	0.5067	0.0000	0.0000	−73.6104
0.01	0.3593	0.1337	0.5070	0.0000	0.0000	−36.804
0.1	0.3593	0.1337	0.5070	0.0000	0.0000	−3.68162
0.5	0.3597	0.1337	0.5065	0.0000	0.0000	−0.73816
1	0.3593	0.1337	0.5070	0.0000	0.0000	−0.36878
5	0.3580	0.1335	0.5084	0.0000	0.0000	−0.07334
10	0.3582	0.1336	0.5082	0.0000	0.0000	−0.03665
100	0.3621	0.1340	0.5038	0.0000	0.0000	−0.00371
1000	0.3397	0.1315	0.5287	0.0000	0.0000	−0.00036

Table 9.3: Optimal allocation under power utility and log-normal returns, $t = 0$

Wealth	US stocks	Int stocks	Corp bnd	Gvnt bnd	Cash	Value
0.9	0.3587	0.1336	0.5077	0.0000	0.0000	−0.18049
0.95	0.3586	0.1336	0.5078	0.0000	0.0000	−0.17096
1	0.3586	0.1336	0.5078	0.0000	0.0000	−0.16238
1.05	0.3585	0.1336	0.5079	0.0000	0.0000	−0.15462
1.1	0.3585	0.1336	0.5079	0.0000	0.0000	−0.14756

Moreover, the value function at time $t = 0$ is approximated by

$$\hat{V}_0(W) \approx -\exp(-1.81782 - 1.00384 \log(W)) = -0.1624 W^{-1.00384},$$

which is very close to the true value function

$$V_0(W) = -\beta^{20} W^{-1} \approx 0.1625 W^{-1},$$

as

$$\beta = E\left[\left(R_f + \sum_{i=1}^{4} (R_i - R_f) x_i^*\right)^{-1}\right] \approx 0.9131586,$$

where $R_f = e^r = 1.0513$.

9.4.7 Dynamic Programming vs Stochastic Programming

By comparison with stochastic programming (SP), dynamic programming (DP) have the following advantages:

(1) DP can solve the infinite horizon or large-number-of-period problems, while SP can only deal with small-number-of-period problems.

(2) DP has much-smaller-size optimization problems, and runs much faster than SP.

(3) DP has stable solutions, while SP has solutions with only $(1 - \alpha)$ confidence level.

(4) DP can get solutions for a range of the initial wealth, while SP could only get solutions for one given wealth, and will have to do another time-consuming non-stable optimization process for another initial wealth.

(5) DP has much more accurate solutions, while SP has low computation precision and the sum of utilities in all scenarios may overflow.

The most of previous research on DP focuses on the discretization method or piecewise linear interpolation method. But the discretization method performs poorly because of it needs a huge amount of nodes such that it can get the desired precision, and the piecewise linear interpolation method performs poorly because of it is non-differentiable on the joint nodes such that the optimizer may give non-optimal solutions. However, after we applied those Chebyshev polynomials or spline methods, these disadvantages disappear, such that DP method is very good in practice.

Many SP scholars may argue that DP method suffers from the "curse of dimensionality". But in fact, for the dynamic asset allocation problem without transaction cost, there is no such problem. In the numerical approximation part, the number of state is only one (which is wealth), and in the numerical integration part, DP method gives alternatives to avoid the "curse of dimensionality": (i) (pseudo) Monte Carlo method for very high dimensional integration; (ii) deterministic numerical integration method for low or medium-dimensional integration, which are much faster, much stabler and much more accurate than (pseudo) Monte Carlo method, while SP has only "(pseudo) Monte Carlo" option.

Many scholars prefer to SP method because they thought that DP cannot deal with transaction cost well. But in fact, transaction costs can be handled in DP method, and even better than SP in view of accuracy. This is going to be further discussed in chapter 10.

Many programmers prefer SP method because they think that DP method is more complicated to program, as it includes three major parts: numerical approximation, numerical integration and numerical optimization. But in practice, SP method has to deal with large-scale numerical optimization by using some decomposition techniques and requiring enormous attention for large-scale nonlinear optimization problems, and has to consider the simulation process very carefully, and has to apply some variance reduction techniques which are necessary to get a good solution. Therefore, now it is hard to say which method is more complicated to program, after we give a clear and detailed discussion on DP method.

Numerical Examples

Here we illustrate the difference between numerical DP and SP with some simple few-period portfolio examples with one risky stock and one bond with a riskless interest rate r. Assume that the risky stock price S_t follows a stochastic differential equation:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dz_t,$$

where z_t is a standard Brownian motion. By discretizing the stochastic process of S_t, we know that the stock return of one period could be assumed to be log-normal or normal.

In the following examples, numerical DP will use degree-9 Chebyshev polynomial interpolation method and Gauss-Hermite quadrature rule with 9 nodes.

Log-Normal Return and Power Utility

The first example is a very simple one-period portfolio problem with the above stock and bond, while the terminal utility is assumed to be a power utility function $u(W) = \frac{W^{1-\gamma}}{1-\gamma}$ with $\gamma > 0$ and $\gamma \neq 1$, and the stock return of one period with length Δt, R, is assumed to be log-normal, i.e., $\log(R) \sim N(\mu - \frac{\sigma^2}{2})\Delta t, \sigma^2 \Delta t)$. In the continuous time optimal portfolio problem of Merton [37], if the stock price follows a geometric Brownian motion and the terminal utility function is assumed to be $u(W) = \frac{W^{1-\gamma}}{1-\gamma}$, then under some general assumptions, the continuous time optimal portfolio allocation fraction is

$$x^* = (\mu - r)/(\gamma\sigma^2),$$

which is the well-known Merton's ratio.

Since our log-normal return assumption is derived from discretization of the stochastic process of the stock price, from the discussion in section 9.4.5, we know that the optimal allocation fraction x^* for the discrete time model (9.4) should be also close to the Merton's ratio, i.e., $x^* = (\mu - r)/(\gamma\sigma^2)$, if $x^* \in (0, 1)$.

In this one-period example, we let $r = 0.04$, $\mu = 0.108$, $\sigma = 0.1572$, $\gamma = 3$, and let the initial wealth be $W_0 = 1$. Then we know the optimal allocation fraction $x^* = (\mu - r)/(\gamma\sigma^2) = 0.9172391$ for the stock, while the optimal allocation fraction for the risk-free bond is $1 - x^* = 0.0827609$ for the continuous time model.

Table 9.4 display the solutions of numerical dynamic programming (NDP) and stochastic programming (SP) for this simple example with various $\Delta t = 0.01, 0.1, 1$.

We see that NDP performs very well as its optimal allocation fraction is very close to the Merton's ratio in each table. But SP method gets a mean solution close to the Merton's ratio with a small standard deviation only when $\Delta t = 0.1, 1$ and the number of scenarios is 10000.

The reason why SP does not work when $\Delta t = 0.01$ is that in the model the objective function

$$f(x, W_0) = E(u(W_0(xR + (1 - x)R_f)))$$

is flat on x, where $R_f = e^{r\Delta t}$ and $\log(R) \sim N((\mu - \frac{\sigma^2}{2})\Delta t,\ \sigma^2\Delta t)$. The following table lists the objective values $f(x, 1)$ when $x = x^*, 1, 0$:

	$f(x^*, 1)$	$f(1, 1)$	$f(0, 1)$
$\Delta t = 0.01$	-0.499289	-0.5007098	-0.499600
$\Delta t = 0.1$	-0.492932	-0.5071438	-0.496016
$\Delta t = 1$	-0.433651	-0.5762102	-0.461558

In the above table, $f(1, 1)$ and $f(0, 1)$ are computed from the formulas

$$\begin{aligned} f(1,1) &= E(u(R)) = E(\exp((1-\gamma)X))/(1-\gamma) \\ &= \exp((-\mu + \gamma\sigma^2/2)(1-\gamma)\Delta t)/(1-\gamma), \end{aligned}$$

Table 9.4: Solution of NDP and SP for 1-period problem with a power utility

		Optimal stock fraction
$\Delta t = 0.01$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.916964$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.9361$ $sd(x^*_{SP10000}) = 0.1115$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.5446$ $sd(x^*_{SP1000}) = 0.4156$
$\Delta t = 0.1$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.917287$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.912$ $sd(x^*_{SP10000}) = 0.062$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.9109$ $sd(x^*_{SP1000}) = 0.1036$
$\Delta t = 1$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.918011$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.9253$ $sd(x^*_{SP10000}) = 0.0124$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.8669$ $sd(x^*_{SP1000}) = 0.0443$

Note: $\bar{\cdot}$ means the average of the 10 solutions.
$sd(\cdot)$ means the estimated standard deviation of the 10 solutions.

Table 9.5: Solution of NDP and SP for 2-period problem with a power utility

	Optimal stock fraction
NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.91801$
SP with 200×200 scenarios (10 runs)	$\bar{x}^*_{SP,200\times200} = 0.9611$ $sd(x^*_{SP,200\times200}) = 0.0747$
SP with 40×40 scenarios (10 runs)	$\bar{x}^*_{SP,40\times40} = 0.7971$ $sd(x^*_{SP,40\times40}) = 0.2164$

Note: The solutions are the allocation fractions at stage 0.

and $f(0,1) = u(R_f)$.

The Monte Carlo property tells us that the relative error of Monte Carlo integration is $O(1/\sqrt{N})$ where N is the number of scenarios. Thus when the objective function is flat within a relative change $O(\varepsilon)$, the number of scenarios should be about $[1/\varepsilon^2]$. The comparison between Gauss-Hermite quadrature rules and Monte Carlo methods for one-period investment problems with log-normal random return and power utility functions are also shown in Table 4.2, 4.3, and 4.4.

For multi-period asset allocation problems, the same inaccuracy problem of SP occurs. For example, if we modify the above a single period into 2-period or 3-period with each period $\Delta t = 1$, then the theoretical analysis tells us the optimal allocation fraction is still the above $x^* = 0.9172391$ for any period and any wealth. Table 9.5 and 9.6 list the solutions given by NDP and SP when $\Delta t = 1$ for 2-period and 3-period problems respectively.

Table 9.6: Solution of NDP and SP for 3-period problem with a power utility

	Optimal stock fraction
NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.91801$
SP with $40 \times 40 \times 40$ scenarios (10 runs)	$\bar{x}^*_{SP,40\times40\times40} = 0.7538$ $sd(x^*_{SP,40\times40\times40}) = 0.237$

Note: The solutions are the allocation fractions at stage 0.

From the results in Table 9.4 $\sim$ 9.6, we see that NDP works very well, its solutions are same across the time, while SP does not work well in the 2-period problem and becomes much worse in the 3-period problem with $40 \times 40 \times 40$ scenarios.

Normal Return and Exponential Utility

The next example is a very simple one-period portfolio problem with the previous stock and bond, while the terminal utility is assumed to be an exponential utility function $u(W) = -e^{-\lambda W}$ with $\lambda > 0$, and the stock return of one period with length Δt, R, is assumed to be normal, i.e., $R \sim N(\mu\Delta t, \sigma^2\Delta t)$.

From the discussion in section 9.4.4, we know that the optimal allocation fraction x^*_t of the stock for the DP model (9.4) with a period length Δt is

$$x^*_t = (\mu - r)/(\lambda(1 + r\Delta t)^{T-t-1}\sigma^2 W_t),$$

for $t = 0, 1, \ldots, T-1$.

In this one-period example, we let $r = 0.04$, $\mu = 0.07$, $\sigma = 0.2$, $\lambda = 1$, and let the initial wealth be $W_0 = 1$, while $T = 1$. Then we know the optimal allocation $x^*_0 W_0 = (\mu - r)/(\lambda\sigma^2) = 0.75$ for the stock, while the optimal allocation fraction for the risk-free bond is $(1 - x^*)W_0 = 0.25$.

Table 9.7 displays the solutions of NDP and SP for this simple example with various $\Delta t = 0.01, 0.1, 1$.

We see that NDP performs very well as its optimal allocation is very close to the exact optimal allocation $x^*_0 W_0 = 0.75$. But SP gets a close mean solution with a small standard deviation only when $\Delta t = 1$ and the number of scenarios is 10000.

The comparison between Gauss-Hermite quadrature rules and Monte Carlo methods for one-period investment problems with normal random return and exponential utility functions are also shown in Table 4.5 and 4.6.

For multi-period asset allocation problems, the same inaccuracy problem of SP occurs. For example, if we modify the above a single period into 2-period or 3-period with each period $\Delta t = 1$, then the theoretical analysis tells us the optimal allocation fraction is still the above x^*_t for stage t. That is, for the 2-period problem, the optimal allocation at stage 0 is

$$x^*_0 W_0 = (\mu - r)/(\lambda(1 + r)\sigma^2) = 0.721154,$$

Table 9.7: Solution of NDP and SP for 1-period problem with a CARA utility

		Optimal stock allocation
$\Delta t = 0.01$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.742715$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.6723$ $sd(x^*_{SP10000}) = 0.3802$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.6081$ $sd(x^*_{SP1000}) = 0.4345$
$\Delta t = 0.1$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.749964$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.6751$ $sd(x^*_{SP10000}) = 0.1707$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.6356$ $sd(x^*_{SP1000}) = 0.3778$
$\Delta t = 1$	NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.749996$
	SP with 10000 scenarios (10 runs)	$\bar{x}^*_{SP10000} = 0.7591$ $sd(x^*_{SP10000}) = 0.0569$
	SP with 1000 scenarios (10 runs)	$\bar{x}^*_{SP1000} = 0.7357$ $sd(x^*_{SP1000}) = 0.1800$

Note: $\bar{\cdot}$ means the average of the 10 solutions.
$sd(\cdot)$ means the estimated standard deviation of the 10 solutions.

Table 9.8: Solution of NDP and SP for 2-period problem with a CARA utility

	Optimal stock allocation
NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.720661$
SP with 200×200 scenarios (10 runs)	$\bar{x}^*_{SP,200\times200} = 0.7314$ $sd(x^*_{SP,200\times200}) = 0.2676$
SP with 40×40 scenarios (10 runs)	$\bar{x}^*_{SP,40\times40} = 0.7289$ $sd(x^*_{SP,40\times40}) = 0.3414$

Note: The solutions are the allocation fractions at stage 0.

and for the 3-period problem, the optimal allocation at stage 0 is

$$x_0^* W_0 = (\mu - r)/(\lambda(1+r)^2\sigma^2) = 0.693417.$$

Table 9.8 and 9.9 list the solutions given by NDP and SP when $\Delta t = 1$ for 2-period and 3-period problems respectively.

From the results in Table 9.7 $\sim$ 9.9, we see that NDP works very well, while SP gives a mean solution close to the exact optimal allocation with high standard deviation in the 2-period problem and becomes much worse in the 3-period problem with $40 \times 40 \times 40$ scenarios.

Table 9.9: Solution of NDP and SP for 3-period problem with a CARA utility

	Optimal stock allocation
NDP with 9-node Gauss-Hermite quadrature	$x^*_{NDP} = 0.691153$
SP with $40 \times 40 \times 40$ scenarios (10 runs)	$\bar{x}^*_{SP,40\times40\times40} = 0.5946$ $sd(x^*_{SP,40\times40\times40}) = 0.3764$

Note: The solutions are the allocation fractions at stage 0.

In this section, all NDP results are given by AMPL programs and KNITRO optimizer, and all SP results are given by GAMS programs and NEOS CONOPT optimizer (`http://neos.mcs.anl.gov`). We also try other optimizers such as SNOPT, KNITRO for SP, all results are similar.

Since there is no difference between SP and stochastic DP for a single period asset allocation problem, we see that the same inaccuracy problem occurs in the stochastic DP. Moreover, for multi-period asset allocation problem, since stochastic DP uses (pseudo) Monte Carlo method to estimate the expectation of next-stage value function, the same inaccuracy problem occurs again.

Apart from the inaccuracy occurred by Monte Carlo method, the large-scaliness of optimization problem in SP will also make its solution inaccurate.

9.5 Dynamic Portfolio Optimization Problem with Consumption

Let W_t be the wealth, and let x_t be the allocation fractions in n risky assets after consumption and reallocation at time t. If we consume $C_t = c_t W_t$, then

$$W_{t+1} = W_t(1 - c_t)(R_f(1 - e^\top x_t) + R^\top x_t),$$

for $t = 0, 1, \ldots, T-1$. If there is no riskless asset, then $W_{t+1} = W_t(1 - c_t)(R^\top x_t)$, while x_t satisfies $e^\top x_t = 1$.

The dynamic portfolio optimization problem is to find optimal portfolio and consumption decision at each stage such that we have a maximal expected total utility, i.e.,

$$\max \ \beta^T E[u(W_T)] + \sum_{t=0}^{T-1} \beta^t E[u(C_t)],$$

where u is the given utility function, β is the discount factor.

The DP model for this problem is

$$V_t(W_t) = \max \ u(C_t) + \beta E\left[V_{t+1}(W_{t+1}) \mid W_t\right],$$

for $t = 0, 1, \ldots, T-1$, while $V_T(W) = u(W)$.

9.5.1 CRRA Utility and Investment-Consumption Problem

When the utility function in the investment-consumption problem is a CRRA utility with a constant relative risk aversion coefficient $\gamma > 0$, Samuelson [48] told us that the optimal portfolio rule is independent of wealth W at each stage and independent of all consumption-saving decisions, and the optimal consumption decisions are proportional to the wealth at each stage. Merton [37] gave the explicit optimal portfolio rule for continuous-time investment-consumption problem, and it was extended in Merton [38] for HARA utility functions.

Here we extend the conclusion to the case with no-shorting and no-borrowing constraints, while riskless asset may or may not exist. Assume that the wealth at stage t before consumption is W_t, the return vector of all available assets is R (R_f will be an element of R if the riskless asset is available), and the portfolio allocation fraction vector after consumption and reallocation at stage t is $x_t \geq 0$, and the consumption at stage t is $C_t = c_t W_t$. Then the next-stage wealth is $W_{t+1} = W_t(1-c_t)(R^\top x_t)$ while $1 - e^\top x_t = 0$.

When $u(W) = W^{1-\gamma}/(1-\gamma)$ for $\gamma > 0$ and $\gamma \neq 1$, since $u(C_t) = W_t^{1-\gamma} u(c_t)$ and $W_{t+1} = W_t(1-c_t)(R^\top x_t)$, we can show that

$$V_t(W_t) = \alpha_t W_t^{1-\gamma},$$

where

$$\begin{aligned} \alpha_t &= \max_{c_t, x_t} \; u(c_t) + \beta\alpha_{t+1}(1-c_t)^{1-\gamma} E\left[(R^\top x_t)^{1-\gamma}\right] \\ &\quad \text{s.t.} \quad x_t \geq 0, \;\; 1 - e^\top x_t = 0, \end{aligned}$$

for $t = 0, 1, \ldots, T-1$, and $\alpha_T = 1/(1-\gamma)$. From the above optimization problem, we see that the optimal portfolio rules x_t^* are independent of W_t, c_t and time t (that is, $x_t^* \equiv x^*$), and the optimal consumption c_t^* is independent of W_t and satisfies the following KKT condition:

$$(c_t^*)^{-\gamma} = \beta\alpha_{t+1}(1-\gamma)(1-c_t^*)^{-\gamma} E[(R^\top x^*)^{1-\gamma}],$$

which implies that

$$c_t^* = \left[1 + \left(\beta\alpha_{t+1}(1-\gamma)E[(R^\top x^*)^{1-\gamma}]\right)^{1/\gamma}\right]^{-1}.$$

Notice that the above formula for c_t^* is same in principle with the formula given in Samuelson [48] (The original equation (24) in Samuelson [48] was $a_1 = [(1+r^*)/(1+\rho)]^{1/\gamma - 1}$, but it should be understood as $a_1 = [(1+r^*)^\gamma/(1+\rho)]^{1/(\gamma-1)}$ by adding parentheses. And the γ in Samuelson [48] is same with $1-\gamma$ here, and $\beta = 1/(1+\rho)$, $(1+r^*)^\gamma$ in Samuelson [48] is same with $E[(R^\top x^*)^{1-\gamma}]$ here). For x^*, it satisfies the same equations (9.6) or (9.7) in principle, which are different with those in Samuelson [48]. Thus,

$$\alpha_t = u(c_t^*) + \beta\alpha_{t+1}(1-c_t^*)^{1-\gamma} E\left[(R^\top x^*)^{1-\gamma}\right],$$

which is dependent on T, for $t = 0, 1, \ldots, T-1$.

For the infinite-horizon problem ($T = \infty$), since $x_t^* \equiv x^*$ are independent of W_t, c_t and time t and $V_t(W_t) = \alpha_t W_t^{1-\gamma}$, if there exists a stationary consumptions and a stationary value function, i.e., $\alpha_t \equiv \alpha$, $c_t^* \equiv c^*$ for all t, then we have

$$\begin{cases} c^* = \left[1 + \left(\beta\alpha(1-\gamma)E[(R^\top x^*)^{1-\gamma}]\right)^{1/\gamma}\right]^{-1}, \\ \alpha = u(c^*) + \beta\alpha(1-c^*)^{1-\gamma}E\left[(R^\top x^*)^{1-\gamma}\right]. \end{cases}$$

Solving the equations, we get

$$\begin{cases} c^* = 1 - \left(\beta E[(R^\top x^*)^{1-\gamma}]\right)^{1/\gamma} \\ \alpha = (c^*)^{-\gamma}/(1-\gamma) \end{cases}.$$

That is, the value function for the infinite-horizon problem with $u(W) = W^{1-\gamma}/(1-\gamma)$ is

$$V(W) = (c^*)^{-\gamma}u(W),$$

where $c^* = 1 - \left(\beta E[(R^\top x^*)^{1-\gamma}]\right)^{1/\gamma}$ is the optimal consumption fraction of wealth, x^* is the optimal portfolio rule, at each stage. Notice that we should have $c^* > 0$, so we should have $\beta E[(R^\top x^*)^{1-\gamma}] < 1$.

When $u(W) = \log(W)$, we can show that

$$V_t(W_t) = \xi_t + \eta_t \log(W_t),$$

where

$$\begin{aligned} \xi_t &= \beta\xi_{t+1} + \max_{c_t, x_t} \log(c_t) + \beta\eta_{t+1}(\log(1-c_t) + E\left[\log(R^\top x_t)\right]) \\ &\quad \text{s.t.} \quad x_t \geq 0, \ 1 - e^\top x_t = 0, \end{aligned}$$

and $\eta_t = 1 + \beta\eta_{t+1}$, for $t = 0, 1, \ldots, T-1$, while $\xi_T = 0$ and $\eta_T = 1$. In fact, from $\eta_t = 1 + \beta\eta_{t+1}$ and $\eta_T = 1$, we have

$$\eta_t = (1 - \beta^{T-t+1})/(1-\beta),$$

when $\beta \neq 1$, or $\eta_t = T - t + 1$ when $\beta = 1$, for $t = 0, 1, \ldots, T$. Moreover, from the iterative relation of ξ_t, we see that the optimal portfolio rules x_t^* are independent of W_t, c_t and time t (that is, $x_t^* = x^*$, while x^* satisfies the equations (9.6) or (9.7) in principle), and the optimal consumption c_t^* is independent of W_t and x_t, and

$$c_t^* = 1/(1 + \beta\eta_{t+1}) = (1-\beta)/(1-\beta^{T-t+1}),$$

when $\beta \neq 1$, or $c_t^* = 1/(T-t+1)$ when $\beta = 1$, for $t = 0, 1, \ldots, T-1$. Thus, the iterative relation for ξ_t is

$$\xi_t = \beta\xi_{t+1} + \log(c_t^*) + \beta\eta_{t+1}(\log(1-c_t^*) + E\left[\log(R^\top x^*)\right]).$$

For the infinite-horizon problem with $0 < \beta < 1$, if there exists a stationary consumptions and

a stationary value function, i.e., $\xi_t = \xi$, $\eta_t = \eta$, and $c_t^* = c_t$ for all t, then we have $\eta = 1/(1-\beta)$, $c^* = 1-\beta$ (which is equal to the optimal consumption decision in the degenerate case for the power utility functions with $\gamma = 1$), and

$$\xi = \beta\xi + \log(c^*) + \beta\eta(\log(1-c^*) + E\left[\log(R^\top x^*)\right]),$$

which implies that

$$\xi = \left(\log(1-\beta) + \frac{\beta}{1-\beta}(\log(\beta) + E\left[\log(R^\top x^*)\right])\right)/(1-\beta).$$

That is, the value function for the infinite-horizon problem with $u(W) = \log(W)$ is

$$V(W) = \xi + \log(W)/(1-\beta),$$

where ξ is given in the previous equation in which x^* is the optimal portfolio rule, and the optimal consumption fraction of wealth at each stage is $c^* = 1-\beta$.

The above results can be extended to the continuous-time investment-consumption problem with no-shorting and no-borrowing constraints and/or no-riskless-asset case, and the explicit solutions for the optimal portfolio rules and optimal consumption decisions and explicit value function can be derived, see Cai [8].

9.6 Dynamic Portfolio Optimization Problem with Cash Flow

In some cases, there will be deposit or withdrawal at each stage $t = 0, 1, \ldots, T-1$. Let D_t denote the possible cash flow over the time $t = 0, \ldots, T-1$. If we use wealth W_t, which is the amount of money right before the cash flow D_t, as the state variables, then the model becomes

$$V_t(W_t) = \max_{x_t} \; E\left[V_{t+1}((W_t + D_t)(R_f(1 - e^\top x_t) + R^\top x_t))\right].$$

When there is no riskless asset, we just need to cancel the R_f term in the Bellman equation, and add a constraint $e^\top x_t = 1$.

When we use numerical DP method to solve this kind of problems, the techniques discussed in section 9.4.2 can be applied. Moreover, if the random return vector R is assumed to have a truncated multivariate normal distribution or a multivariate log-normal distribution, then the techniques discussed in section 9.4.1 or 9.4.3 can also be applied to solve this kind of problems.

Chapter 10

Portfolio with Transaction Costs

In the previous chapter, we discussed the dynamic portfolio problems without transaction costs, and the application of numerical DP methods for solving the problems. In this chapter, we consider the transaction cost into the DP models in the previous chapter, then the problems have multiple dimensional approximation.

10.1 DP Model for Portfolio Problems with Transaction Costs

For the dynamic portfolio problems with transaction costs, we can choose the state variables as the wealth W_t and allocation fractions $x_t = (x_{t1}, \ldots, x_{tn})^\top$ invested in the risky assets. Here W_t and x_t are the values rightly before re-allocation at time t. Thus, the DP model becomes

$$\begin{aligned} V_t(W_t, x_t) \quad = \quad & \max_{\Delta_t} \quad E\left[V_{t+1}(W_{t+1}, x_{t+1})\right] \\ & \text{s.t.} \quad e^\top(\Delta_t + f(\Delta_t)) = M_t, \\ & \qquad X_{t+1} = R. * (x_t W_t + \Delta_t), \\ & \qquad W_{t+1} = e^\top X_{t+1} + R_f((1 - e^\top x_t)W_t - M_t), \\ & \qquad x_{t+1} = X_{t+1}/W_{t+1}, \end{aligned}$$

where R_f is the riskfree return, $R = (R_1, \ldots, R_n)^\top$ is the random return vector of the risky assets, X_t is the vector of the amount of dollars invested in the risky assets, Δ_t is the vector of amount of dollars with which buying or selling the risky assets, $.*$ is elementwise product, e is a column vector with 1 everywhere, and $f(\Delta_{ti})$ is the transaction cost function for buying or selling part of stock i with amount of Δ_{ti} dollars. We could let $f(\Delta_{ti}) = \tau\Delta_{ti}^2$, or $f(\Delta_{ti}) = \tau|\Delta_{ti}|$, or the hyperbolic form, for some constant $\tau > 0$. The terminal value function is $V_T(W, x) = u(W)$ for some given utility function u. Sometimes, the terminal value function is chosen as $V_T(W, x) = u(W - \tau e^\top f(xW))$, if we assume that all risky assets have to be converted into the riskless asset before consumption. Later, we just assume that $V_T(W, x) = u(W)$ for simplicity.

If we do not allow shorting stocks or borrowing cash, then we just need to add the constraints: $x_t W_t + \Delta_t \geq 0$ and $(1 - e^\top x_t)W_t \geq M_t$. And the range of x_t and x_{t+1} is $[0,1]^n$ while $e^\top x_{t+1} \leq 1$.

In the optimal asset allocation problem, R is often assumed to be log-normal and correlated. Assume that the random log-returns of the assets, $\log(R) = (\log(R_1), \cdots, \log(R_n))^\top \in \mathbb{R}^n$, have a multivariate normal distribution $N((\mu - \frac{\sigma^2}{2})\Delta t, (\Lambda\Sigma\Lambda)\Delta t)$ in $\mathbb{R}^n$, where Δt is the length of a period, $\mu = (\mu_1, \cdots, \mu_n)^\top$ is the drift, $\sigma = (\sigma_1, \cdots, \sigma_n)^\top$ is the volatility, and Σ is the correlation matrix of the log-returns, and $\Lambda =$diag$(\sigma_1, \ldots, \sigma_n)$.

Since the correlation matrix is positive definite, we can apply Cholesky factorization to Σ such that $\Sigma = LL^\top$, where $L = (L_{ij})_{n\times n}$ is a lower triangular matrix. Then

$$\log(R_i) = (\mu_i - \frac{\sigma_i^2}{2})\Delta t + \sigma_i\sqrt{\Delta t}\sum_{j=1}^{i} L_{ij} z_j,$$

where z_i are independent standard normal random variables, for $i = 1, \ldots, n$. For simplicity, let $\Delta t = 1$. Thus,

$$\begin{aligned} W_{t+1} &= R_f(W_t(1 - e^\top x_t) - M_t) + R^\top(x_t W_t + \Delta_t) \\ &= R_f(W_t(1 - e^\top x_t) - M_t) + \sum_{i=1}^{n} \exp\left(\mu_i - \frac{\sigma_i^2}{2} + \sigma_i \sum_{j=1}^{i} L_{ij} z_j\right)(x_{ti} W_t + \Delta_{ti}), \end{aligned}$$

where $R_f = e^r$ for the continuously compound interest rate r.

Therefore, we can apply product Gauss-Hermite quadrature to estimate the conditional expectation of $V_{t+1}(W_{t+1}, x_{t+1})$ while W_t, x_t are given. The numerical DP model becomes

$$\begin{aligned} V_t(W_t, x_t) = \max_{\Delta_t} \ & \pi^{-\frac{n}{2}} \sum_{k_1,\ldots,k_n=1}^{m} \omega_{k_1} \cdots \omega_{k_n} V_{t+1}(W_{t+1,k_1,\ldots,k_n}, x_{t+1,k_1,\ldots,k_n}) \\ \text{s.t.} \quad & e^\top(\Delta_t + f(\Delta_t)) = M_t, \\ & W_{t+1,k_1,\ldots,k_n} = R_f(W_t(1 - e^\top x_t) - M_t) + \sum_{j=1}^{n} X_{t+1,j,k_1,\ldots,k_i}, \\ & X_{t+1,i,k_1,\ldots,k_i} = \exp\left(\mu_i - \frac{\sigma_i^2}{2} + \sigma_i\sqrt{2}\sum_{j=1}^{i} q_{k_j} L_{ij}\right)(x_{ti} W_t + \Delta_{ti}), \\ & x_{t+1,i,k_1,\ldots,k_i} = X_{t+1,i,k_1,\ldots,k_i} / W_{t+1,k_1,\ldots,k_n}, \quad i = 1, \ldots, n, \end{aligned}$$

where $x_{t+1,k_1,\ldots,k_n} = (x_{t+1,1,k_1}, x_{t+1,2,k_1,k_2}, \ldots, x_{t+1,n,k_1,\ldots,k_n})^\top$, $V_T(W, x) = u(W)$, the ω_k and q_k are the Gauss-Hermite quadrature weights and nodes over $(-\infty, \infty)$, for $k = 1, \ldots, m$. When there is no riskless asset, we just need to cancel the R_f term and replace M_t by 0 in the constraints, while we should have $e^\top x_t = 1$ (the state variable vector $x_t = (x_{t1}, \ldots, x_{tn})$ should be changed as $(x_{t1}, \ldots, x_{t,n-1})$, and there is same cutoff in $x_{t+1,k_1,\ldots,k_n}$).

10.2 Proportional Transaction Cost and CRRA Utility

In economics and finance, we usually assume a CRRA utility function and a proportional transaction cost, i.e., $u(W) = W^{1-\gamma}/(1-\gamma)$ for some constant $\gamma > 0$ and $\gamma \neq 1$, or $u(W) = \log(W)$ for $\gamma = 1$, and $f(\Delta) = \tau|\Delta|$ for some constant $\tau > 0$. By let $\Delta = (\delta^+ - \delta^-)W$ with $\delta^+, \delta^- \geq 0$, we have $|\Delta| = (\delta^+ + \delta^-)W$. Let r be the risk-free rate.

The model becomes

$$\begin{aligned} V_t(W_t, x_t) &= \max_{\delta_t^+, \delta_t^-} E\left[V_{t+1}(W_{t+1}, x_{t+1})\right] \\ &\quad \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad X_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-)W_t, \\ &\qquad W_{t+1} = e^\top X_{t+1} + R_f(1 - e^\top x_t - m_t)W_t, \\ &\qquad x_{t+1} = X_{t+1}./W_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

where the terminal value function is $V_T(W, x) = u(W)$.

Thus, for $u(W) = W^{1-\gamma}/(1-\gamma)$, if we assume that $V_{t+1}(W_{t+1}, x_{t+1}) = W_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1})$, then

$$\begin{aligned} V_t(W_t, x_t) &= \max_{\delta_t^+, \delta_t^-} E\left[W_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1})\right] \\ &\quad \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad s_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_f(1 - e^\top x_t - m_t), \\ &\qquad W_{t+1} = \Pi_{t+1} W_t, \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0. \end{aligned}$$

By substituting W_{t+1} by $\Pi_{t+1}W_t$ in the objective function, we get $V_t(W_t, x_t) = W_t^{1-\gamma} \cdot g_t(x_t)$, where

$$\begin{aligned} g_t(x_t) &= \max_{\delta_t^+, \delta_t^-} E\left[\Pi_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1})\right] \\ &\quad \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad s_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_f(1 - e^\top x_t - m_t), \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0. \end{aligned}$$

Therefore, by induction, from $V_T(W,x) = u(W) = W^{1-\gamma} \cdot 1/(1-\gamma)$, we showed that

$$V_t(W_t, x_t) = W_t^{1-\gamma} \cdot g_t(x_t),$$

for any time $t = 0, 1, \ldots, T$, while $g_t(x)$ has the iterative formula given in the above optimization problem and $g_T(x) = 1/(1-\gamma)$ (or $g_T(x) = (1 - \tau e^\top |x|)^{1-\gamma}/(1-\gamma)$ when we assume that all risky assets have to be converted into the riskless asset before consumption), if we assume a proportional transaction cost and a power utility $u(W) = W^{1-\gamma}/(1-\gamma)$ with a constant relative risk aversion coefficient $\gamma > 0$ and $\gamma \neq 1$.

For $u(W) = \log(W)$, we can also show by induction that

$$V_t(W_t, x_t) = \log(W_t) + \psi_t(x_t),$$

where

$$\begin{aligned}
\psi_t(x_t) \quad = \quad & \max_{\delta_t^+, \delta_t^-} \quad E\left[\log(\Pi_{t+1}) + \psi_{t+1}(x_{t+1})\right] \\
& \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\
& \qquad\quad s_{t+1} = R_t . * (x_t + \delta_t^+ - \delta_t^-), \\
& \qquad\quad \Pi_{t+1} = e^\top s_{t+1} + R_f(1 - e^\top x_t - m_t), \\
& \qquad\quad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\
& \qquad\quad \delta_t^+ \geq 0,\ \delta_t^- \geq 0,
\end{aligned}$$

while $\psi_T(x) = 0$ (or $\psi_T(x) = \log(1 - \tau e^\top |x|)$, when we assume that all risky assets have to be converted into the riskless asset before consumption).

When there is no riskless asset, we just need to cancel the R_f term and replace m_t by 0 in the above models for g_t or ψ_t, while we should have $e^\top x_t = 1$ (the state variable vector $x_t = (x_{t1}, \ldots, x_{tn})$ should be changed as $(x_{t1}, \ldots, x_{t,n-1})$, and there is same cutoff in x_{t+1}).

If we do not allow shorting stocks or borrowing cash, then the range of x_t is $[0,1]^n$, and in the models we just need to add the constraints: $x_t + \delta_t^+ - \delta_t^- \geq 0$ and $m_t \leq 1 - e^\top x_t$, such that $x_{t+1} \in [0,1]^n$ and $e^\top x_{t+1} \leq 1$. And we still have the property of separation of W and x in the value functions $V(W,x)$. In fact, in the above model, since variables $m_t, s_{t+1}, \Pi_{t+1}, x_{t+1}$ will be substituted, so the control variables are only δ_t^+, δ_t^-, and the constraints are only $m_t \leq 1 - e^\top x_t$, $x_t + \delta_t^+ - \delta_t^- \geq 0$, $\delta_t^+ \geq 0$ and $\delta_t^- \geq 0$, which are linear, as $m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)) = (1+\tau)e^\top \delta_t^+ + (-1+\tau)e^\top \delta_t^-$ is also linear on δ_t^+ and δ_t^-. Thus, it is a model with $2n$ control variables with $2n$ bound constraints and $(n+1)$ linear constraints, where n is the number of risky assets.

In fact, if there is neither $L_i > 0$ nor $U_i > 0$ such that $L_i \leq R_i \leq U_i$ for a risky asset i (e.g., R_i is log-normal), called by no-boundness of returns, then we have $\Pr(\Pi_t \leq 0) > 0$ when $x_{ti} + \delta_{ti}^+ - \delta_{ti}^- < 0$. This follows that for the CRRA utility functions, the optimal solution must have $x_{ti} + (\delta_{ti}^+)^* - (\delta_{ti}^-)^* \geq 0$ unless the asset i can be replicated by other assets which is a degenerate case. Since the expectation

of next-time value function is computed numerically by numerical DP, we must add the constraint $x_{ti} + \delta_{ti}^+ - \delta_{ti}^- \geq 0$ to avoid an unreasonable approximation solution. If the no-boundness of returns applies for all risky assets, then for the CRRA utility functions, we must have both "no-shorting" and "no-borrowing" constraints: $x_t + \delta_t^+ - \delta_t^- \geq 0$ and $m_t \leq 1 - e^\top x_t$.

We know that there will be a "no-trade" region Ω_t for any $t = 0, 1, \ldots, T-1$. When $x_t \in \Omega_t$, the investor will not trade at all, and when $x_t \notin \Omega_t$, the investor will buy or sell the risky assets such that $(x_t + \delta^+ - \delta^-)/(1 - \tau e^\top(\delta^+ + \delta^-)) \in \partial\Omega_t$ with the smallest transaction costs, where $\partial\Omega_t$ denotes the boundaries of Ω_t. That is, the "no-trade" region Ω_t is defined as

$$\Omega_t = \{x_t : \ (\delta_t^+)^* = (\delta_t^-)^* = 0\},$$

where $(\delta_t^+)^*, (\delta_t^-)^*$ are the optimal controls for the given x_t. See Kamin [29], Constantinides [10, 11, 12], Davis and Norman [13], Muthuraman and Kumar [41], and so on.

Abrams and Karmarkar [1] showed that the "no-trade" region is a connected set and that it is a cone when the utility function is assumed to be positively homogeneous (a function $u(x)$ is positively homogeneous if there exists a positively value function $g(x)$ such that $u(ax) = g(a)u(x)$ for any $a > 0$). Moreover, in the case of proportional transaction costs and concave utility functions, the "no-trade" region can take on many forms ranging from a simple halfline to a nonconvex set. So we should use numerical methods to compute the "no-trade" region.

When we allow shorting stocks and borrowing cash, the domain of x_t could be $(-\infty, +\infty)$, such that it is hard to approximate $g_t(x_t)$ well. In the cases when the "no-trade" region $\Omega_t \subset [0,1]^n$ at each stage t, for the random returns $R = (R_1, \ldots, R_n)^\top$ and the riskfree return R_f, if at each stage t we can have $R^\top y + R_f(1 - e^\top y) \geq R_i y_i$ almost surely for all $1 \leq i \leq n$ and $y = (y_1, \ldots, y_n)^\top \in \partial\Omega_t$, then we can set the domain of x_t as $[0,1]^n$ for each stage t. For example, if $(R_1, \ldots, R_n)$ are i.i.d. random variables with a lower bound $L > 0$ (and we should have $L < R_f$ for no-arbitrage), and let $y_i \leq a$ for a minimal constant a and all $y = (y_1, \ldots, y_n)^\top \in \partial\Omega_t$, then when $a \leq 1/(n - (n-1)L/R_f)$, we can set the domain of x_t as $[0,1]^n$ and require $x_t + \delta_t^+ - \delta_t^- \geq 0$ and $m_t \leq n - e^\top x_t$ for each stage t.

From the separability of W and x, we see that the optimal portfolio rules are independent of wealth W_t. Thus the "no-trade" region Ω_t are also independent of W_t, for the CRRA utility functions.

10.3 Numerical Examples

In this section, we will give several numerical examples, in which the number of risky assets $n = 3, 6$, and there is one riskless asset (called as bond later) available with return $R_f = e^r$, the terminal value function is $u(W) = W^{1-\gamma}/(1-\gamma)$, and the transaction cost function is $f(\Delta) = \tau|\Delta|$ with $\tau > 0$ for buying or selling Δ amount of money of risky assets.

To the best of our knowledge, when the number of correlated risky assets $n \geq 4$ and the number of periods $T \geq 6$, this is the first time to explicitly give a good numerical solution with transaction costs and non-quadratic utility functions. When $n = 1$, the problem has been well studied, see Zabel

[54], Constantinides [10, 12], Gennotte and Jung [17], Boyle and Lin [6], and so on. Kamin [29] considered the case without riskless asset and $n = 2$. Constantinides [11] and Abrams and Karmarkar [1] established some properties of the "no-trade" region for multiple assets, but numerical examples are for $n = 1$.

In the continuous-time version, there are a lot of papers about the portfolio optimization problem with transaction costs when $n \leq 2$, see Davis and Norman [13], Duffie and Sun [15], Akian [2], Janecek and Shreve [27], Liu [33], and so on. Muthuraman and Kumar [41, 42] gave numerical examples with $n \leq 3$. In Muthuraman and Zha [43], they provided a computational scheme by combining simulation with the boundary update procedure while the objective is to maximize the long-term expected growth rate, and presented some computational results with $n \geq 3$. But since they applied simulation into the computational scheme, the accuracy of solutions can be guaranteed, just like what we showed in the numerical examples for the comparison between SP and numerical DP.

By using the numerical DP method and the techniques discussed in section 10.2, we computed the "no-trade" regions for each stage in the following examples. In the numerical DP, we applied the NPSOL optimization package (see Gill, Murray, Saunders, and Wright [20]) for the numerical optimization part. In these examples, we assume that the risky asset return vector R is log-normal, i.e., $\log(R) \sim N((\mu - \sigma^2/2), \Lambda\Sigma\Lambda)$, where $\mu = (\mu_1, \ldots, \mu_n)^\top$, $\sigma = (\sigma_1, \ldots, \sigma_n)$, $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_n)$, and Σ is the correlation matrix of $\log(R)$.

10.3.1 Three Stocks with Log-Normal Returns and One Bond

The first example assumes that three stock returns R_1, R_2, R_3 are independently and identically distributed, and they are log-normal, i.e., $\log(R_i) \sim N((\mu_i - \sigma_i^2/2), \sigma_i^2)$, where $\mu_i = 0.07$, $\sigma_i = 0.2$, for $i = 1, 2, 3$. Let $T = 6$, $\gamma = 3.5$, $\tau = 0.01$, and $r = 0.04$. In the numerical DP, we applied the degree-7 complete Chebyshev approximation method and multi-dimensional product Gauss-Hermite quadrature rule with 9 nodes in each dimension. Our numerical results showed that the "no-trade" regions are close to cubes after canceling small perturbations, and

$$\begin{aligned}
\Omega_5 &\approx [0.139, 0.286]^3, \\
\Omega_4 &\approx [0.166, 0.254]^3, \\
\Omega_3 &\approx [0.1645, 0.2522]^3, \\
\Omega_2 &\approx [0.1639, 0.2516]^3, \\
\Omega_1 &\approx [0.1636, 0.2513]^3, \\
\Omega_0 &\approx [0.1635, 0.2512]^3.
\end{aligned}$$

So we see that the "no-trade" region shrinks and rapidly converges to the infinite horizon limit as the time to the terminal time increases. This property was firstly observed in Gennotte and Jung [17] for the case with only one risky asset and one riskless asset. Moreover, Merton's ratio, which is $(\Lambda\Sigma\Lambda)^{-1}(\mu - r)/\gamma = (0.214, 0.214, 0.214)^\top$, is also located inside of Ω_t, for all t.

The second example assumes that three stock returns are correlated with log-normal distribution,

$\log(R) \sim N((\mu - \sigma^2/2), \Lambda\Sigma\Lambda)$, where $R = (R_1, R_2, R_3)$, $\mu = (0.07, 0.08, 0.09)^\top$, $\sigma = (0.16, 0.18, 0.2)^\top$, Λ is the diagonalized matrix of σ, and

$$\Sigma = \begin{bmatrix} 1 & 0.2 & 0.1 \\ 0.2 & 1 & 0.314 \\ 0.1 & 0.314 & 1 \end{bmatrix}.$$

Let $T = 6$, $\gamma = 3.5$, $\tau = 0.01$ and $r = 0.04$. In the numerical DP, we applied the NPSOL optimization package, the degree-7 complete Chebyshev approximation method and multi-dimensional product Gauss-Hermite quadrature rule with 9 nodes in each dimension. Our numerical results gave us the "no-trade" regions Ω_t for $t = 0, 1, 2, 3, 4, 5$, shown in Figure 10.1.

We see again that the "no-trade" region shrinks and rapidly converges to the infinite horizon limit as the time to the terminal time increases.

10.3.2 Six Stocks with Log-Normal Returns and One Bond

In this example, we assumes that six stock returns $R_1, \ldots, R_6$ are independently and identically distributed, and they are log-normal, i.e., $\log(R_i) \sim N((\mu_i - \sigma_i^2/2), \sigma_i^2)$, where $\mu_i = 0.07$, $\sigma_i = 0.2$, for $i = 1, \ldots, 6$. Let $T = 6$, $\gamma = 3.5$, $\tau = 0.002$, and $r = 0.05$. In the numerical DP, we applied the degree-4 complete Chebyshev approximation method and multi-dimensional product Gauss-Hermite quadrature rule with 5 nodes in each dimension. Our numerical results showed that the "no-trade" regions are close to cubes, and

$$\begin{aligned} \Omega_5 &\approx [0.1259, 0.1552]^6, \\ \Omega_4 &\approx [0.1270, 0.1531]^6, \\ \Omega_t &\approx [0.1269, 0.1530]^6, \ t = 3, 2, 1, 0. \end{aligned}$$

So we see again that the "no-trade" region shrinks and rapidly converges to the infinite horizon limit as the time to the terminal increases. Moreover, Merton's ratio, which is $(\Lambda\Sigma\Lambda)^{-1}(\mu - r)/\gamma = 0.143 \cdot (1, 1, 1, 1, 1, 1)^\top$, is also located inside of Ω_t, for all t.

10.3.3 Two Stocks with Uniform Returns and One Bond

In this example, we assumes that two stock returns R_1, R_2 are independently and identically distributed, and they are uniform on $[0.87, 1.27]$. Let $T = 6$, $\gamma = 2.5$, $\tau = 0.005$, and $r = 0.05$. In the numerical DP, we applied the degree-9 complete Chebyshev approximation method and multi-dimensional product Gauss-Legendre quadrature rule with 9 nodes in each dimension. Our numerical results gave us the "no-trade" regions Ω_t for $t = 0, 1, 2, 3, 4, 5$, shown in Figure 10.2. The circle point in the region Ω_t is the optimal allocation ratio when we assume that there is no transaction cost.

Notice that all the elements of points of the "no-trade" regions are bigger than 0.5, indicating that the corresponding bond fractions are negative. That is, in the regions, the bond is shorted.

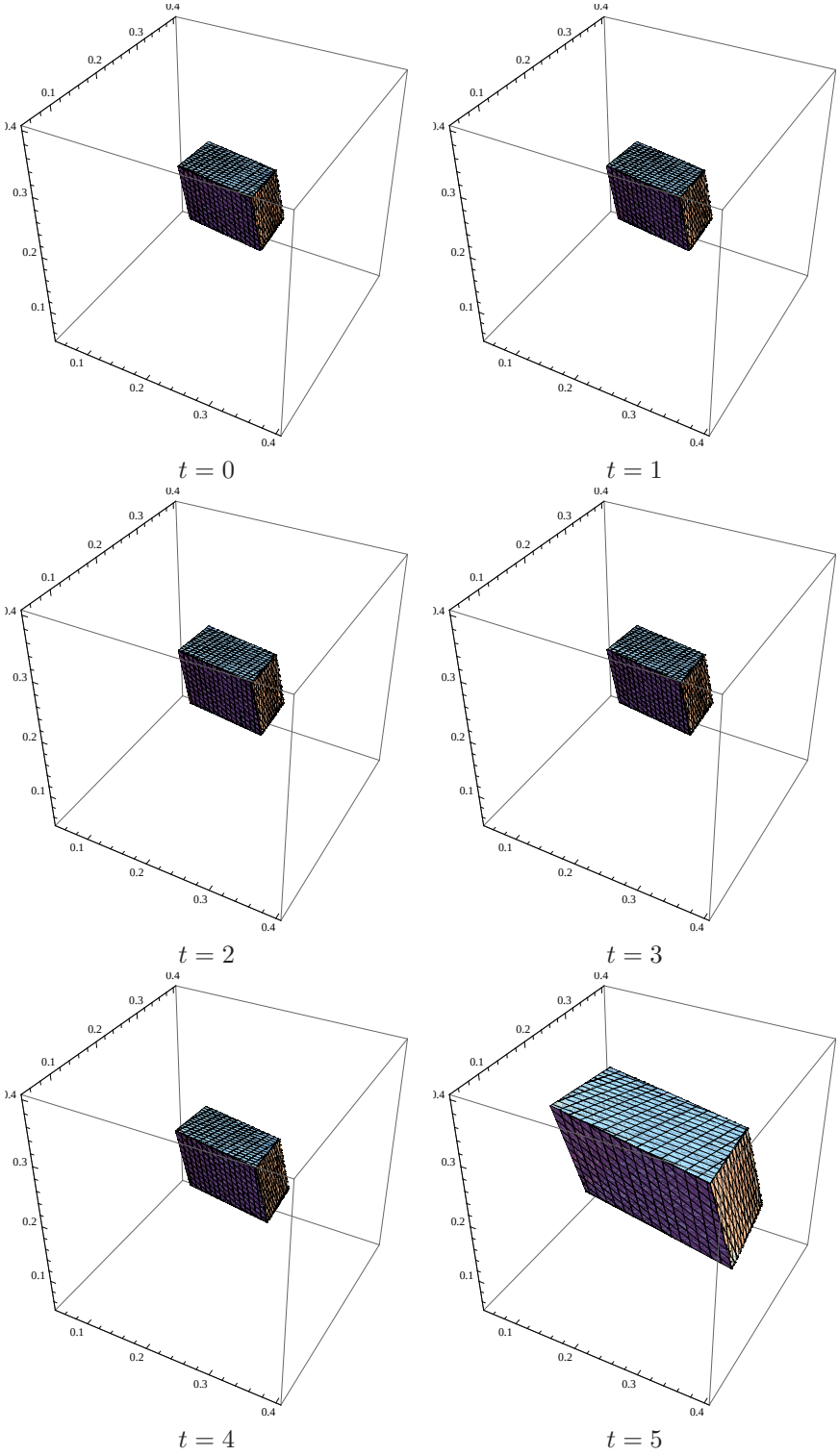
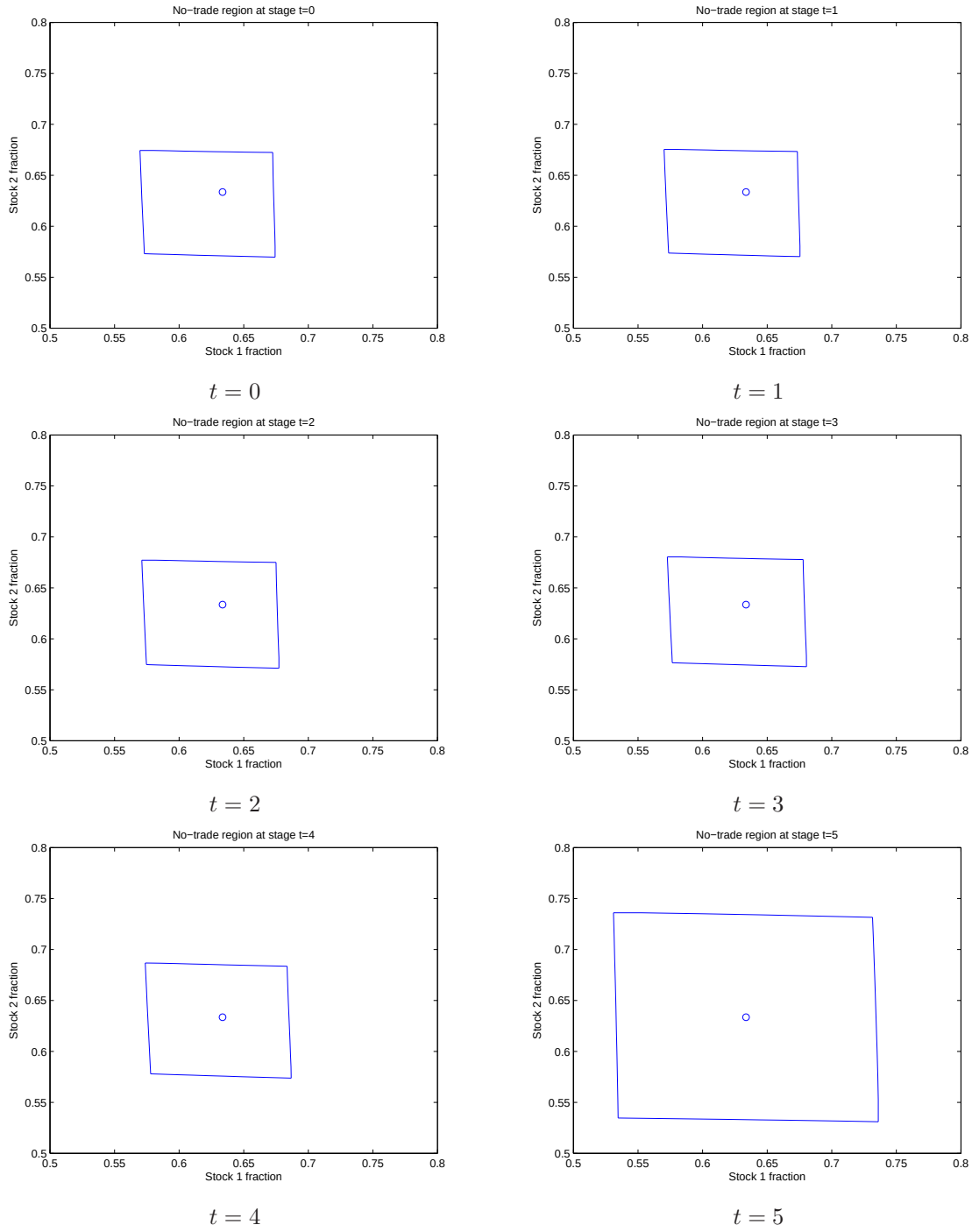


Figure 10.1: No-trade regions for 3 correlated stocks with log-normal returns and 1 bond



$t = 0$ $t = 1$ $t = 2$ $t = 3$ $t = 4$ $t = 5$

Figure 10.2: No-trade regions for 2 stocks with uniform returns and 1 bond

10.3.4 Two Stocks with Discrete Returns and One Bond

In this example, we assumes that two stock returns R_1, R_2 are discrete with the following joint probability mass function:

$$\begin{aligned}
\Pr(R_1 = 0.85, R_2 = 0.88) &= 0.08, \\
\Pr(R_1 = 0.85, R_2 = 1.06) &= 0.1, \\
\Pr(R_1 = 1.08, R_2 = 0.88) &= 0.12, \\
\Pr(R_1 = 1.08, R_2 = 1.06) &= 0.4, \\
\Pr(R_1 = 1.08, R_2 = 1.20) &= 0.12, \\
\Pr(R_1 = 1.25, R_2 = 1.06) &= 0.1, \\
\Pr(R_1 = 1.25, R_2 = 1.20) &= 0.08.
\end{aligned}$$

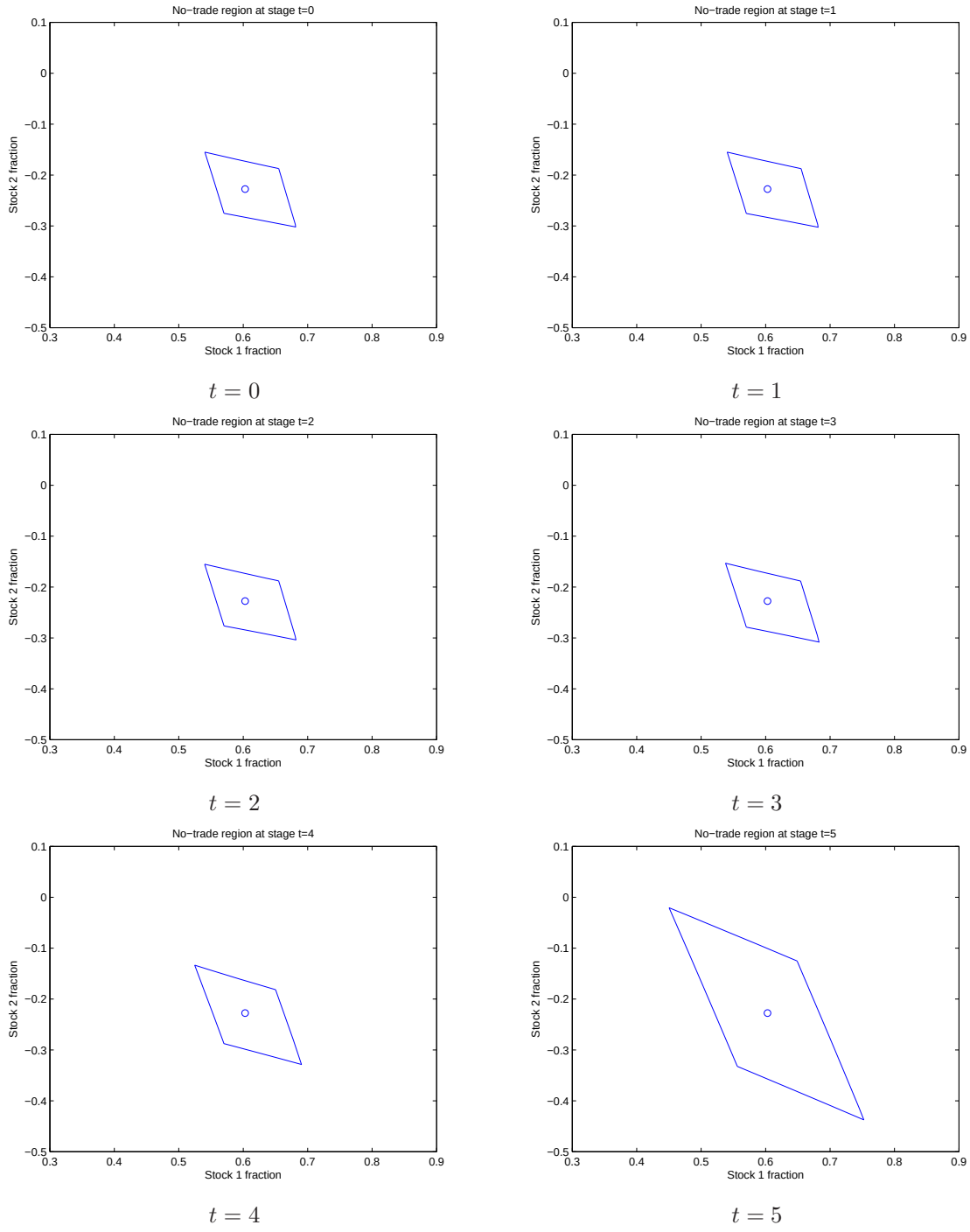
Let $T = 6$, $\gamma = 2.5$, $\tau = 0.003$, and $r = 0.05$. In the numerical DP, we applied the degree-9 complete Chebyshev approximation method. Our numerical results gave us the "no-trade" regions Ω_t for $t = 0, 1, 2, 3, 4, 5$, shown in Figure 10.3. The circle point in the region Ω_t is the optimal allocation ratio when we assume that there is no transaction cost.

Notice that all the second elements of points of the "no-trade" regions are negative, indicating that the second stock is shorted in the region.

10.4 Asset Returns with Stochastic Mean and Covariance

In the previous sections, we always assume that the asset returns are normal or log-normal random variables with fixed interest rate, fixed mean return vector and fixed covariance matrix along the time. But in the real life models, the interest rate r_t, and drift vector μ_t and covariance matrix are stochastic. Let all these parameters be denoted as a vector θ_t at time t, and they could be discrete Markov chains with a given transition probability matrix from previous stage to current stage, or continuously distributed conditional on their previous-stage values.

Assume that the end time utility function is a CRRA utility with a constant relative risk aversion coefficient $\gamma > 0$, and the transaction cost of buying or selling Δ amount of money of one stock is proportional with ratio τ. By choosing wealth W_t, allocation fractions x_t and parameters θ_t as the



$t = 0$ $t = 1$

$t = 2$ $t = 3$

$t = 4$ $t = 5$

Figure 10.3: No-trade regions for 2 stocks with discrete returns and 1 bond

state variables, the DP model becomes

$$\begin{aligned} V_t(W_t, x_t, \theta_t) &= \max_{\delta_t^+, \delta_t^-} E\left[V_{t+1}(W_{t+1}, x_{t+1}, \theta_{t+1}) \mid W_t, x_t, \theta_t\right] \\ &\quad \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad X_{t+1} = R_t . * (x_t + \delta_t^+ - \delta_t^-)W_t, \\ &\qquad W_{t+1} = e^\top X_{t+1} + R_{tf}(1 - e^\top x_t - m_t)W_t, \\ &\qquad x_{t+1} = X_{t+1}/W_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

where the terminal value function is $V_T(W, x, \theta) = u(W)$. Here R_{tf} is the riskfree return and R_t is the random return of risky assets in the period $[t, t+1)$, and they are dependent on the parameter θ_t.

When the utility function is $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma > 0$ and $\gamma \neq 1$, if we assume that $V_{t+1}(W_{t+1}, x_{t+1}, \theta_{t+1}) = W_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1}, \theta_{t+1})$, then we have $V_t(W_t, x_t, \theta_t) = W_t^{1-\gamma} \cdot g_t(x_t, \theta_t)$ where

$$\begin{aligned} g_t(x_t, \theta_t) &= \max_{\delta_t^+, \delta_t^-} E_R\left[\Pi_{t+1}^{1-\gamma} \cdot E_\theta[g_{t+1}(x_{t+1}, \theta_{t+1}) \mid \theta_t, R_t]\right] \\ &\quad \text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad s_{t+1} = R_t . * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_{tf}(1 - e^\top x_t - m_t), \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

in which we applied the tower property of expectation operators.

Therefore, by induction, from $V_T(W, x, \theta) = u(W) = W^{1-\gamma} \cdot 1/(1-\gamma)$, we know that

$$V_t(W_t, x_t, \theta_t) = W_t^{1-\gamma} \cdot g_t(x_t, \theta_t),$$

for any time $t = 0, 1, \ldots, T$, while $g_t(x, \theta)$ has the iterative formula given in the above optimization problem and $g_T(x, \theta) = 1/(1-\gamma)$, if we assume a power utility and a proportional transaction cost.

For $u(W) = \log(W)$, we can also show by induction that

$$V_t(W_t, x_t, \theta_t) = \log(W_t) + \psi_t(x_t, \theta_t),$$

where

$$\begin{aligned}\psi_t(x_t,\theta_t) &= \max_{\delta_t^+,\delta_t^-} E_R\left[\log(\Pi_{t+1}) + E_\theta[\psi_{t+1}(x_{t+1},\theta_{t+1}) \mid \theta_t, R_t]\right] \\ &\text{s.t.} \quad m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)), \\ &\qquad s_{t+1} = R_t . * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_{tf}(1 - e^\top x_t - m_t), \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0,\end{aligned}$$

with $\psi_T(x,\theta) = 0$.

When there is no riskless asset, we just need to cancel the R_{tf} term and replace m_t by 0 in the above models for g_t or ψ_t, while we should have $e^\top x_t = 1$ (the state variable vector $x_t = (x_{t1},\ldots,x_{tn})$ should be changed as $(x_{t1},\ldots,x_{t,n-1})$, and there is same cutoff in x_{t+1}).

If we do not allow shorting stocks or borrowing cash, then the range of x_t is $[0,1]^n$, and in the models we just need to add the constraints: $x_t+\delta_t^+-\delta_t^- \geq 0$ and $m_t \leq 1-e^\top x_t$, such that $x_{t+1} \in [0,1]^n$ and $e^\top x_{t+1} \leq 1$. And we still have the property of separation of W and (x,θ) in the value functions $V(W,x,\theta)$. In fact, in the above model, since variables $m_t, s_{t+1}, \Pi_{t+1}, x_{t+1}$ will be substituted, so the control variables are only δ_t^+, δ_t^-, and the constraints are only $m_t \leq 1-e^\top x_t$, $x_t+\delta_t^+-\delta_t^- \geq 0$, $\delta_t^+ \geq 0$ and $\delta_t^- \geq 0$, which are linear, as $m_t = e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)) = (1+\tau)e^\top\delta_t^+ + (-1+\tau)e^\top\delta_t^-$ is also linear on δ_t^+ and δ_t^-. Thus, it is a model with $2n$ control variables with $2n$ bound constraints and $(n+1)$ linear constraints, where n is the number of risky assets.

10.4.1 Numerical Example

In this example, we assume that the interest rate r_t and the covariance matrix of stocks are constant along the time, but all drift terms of stocks are discrete Markov chains and independent each other. Assume that the assets available for trading include one bond with a constant interest rate $r = 0.04$, and 2 stocks with independent log-normal annual returns $N(\mu_i - \sigma_i^2/2, \sigma_i^2)$ for $i = 1,2$. Let $\sigma_1 = \sigma_2 = 0.2$, and $\mu_i = 0.06$ or 0.08 with the following transition probability matrix

$$\begin{bmatrix} 0.75 & 0.25 \\ 0.25 & 0.75 \end{bmatrix},$$

for $i = 1,2$, while μ_1 is independent of μ_2. Let the terminal time be $T = 6$ years while there is a reallocation chance at the beginning of each year with a proportional transaction cost ratio $\tau = 0.01$ for buying or selling stocks. The terminal utility function is $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma = 3$.

The numerical solutions are given by using the product Gauss-Hermite quadrature rule with 9 nodes in each dimension, and the degree-9 complete Chebyshev polynomial approximation method with 10 Chebyshev nodes in each dimension, Figure 10.4 displays the no-trade regions for four possible discrete states of (μ_1,μ_2) at each stage $t = 0,1,\ldots,5$. The top-right squares are the no-trade regions

for the state $(\mu_1, \mu_2) = (0.08, 0.08)$, and the bottom-left squares are the no-trade regions for the state $(\mu_1, \mu_2) = (0.06, 0.06)$, and the top-left and the bottom-right squares are respectively the no-trade regions for the state $(\mu_1, \mu_2) = (0.06, 0.08)$ and $(\mu_1, \mu_2) = (0.08, 0.06)$. The circle points inside the squares are the optimal allocation fractions given the discrete states (μ_1, μ_2) when we assume that there is no transaction cost in trading stocks.

10.5 Portfolio with Transaction Costs and Consumption

At the time rightly before reallocation time t, let W_t be the wealth, and let x_t be the allocation fractions of the wealth in n risky assets. The dynamic portfolio optimization problem is to find optimal portfolio and consumption decision C_t at each stage t such that we have a maximal expected total utility, i.e.,

$$\max \ \beta^T E[u(W_T)] + \sum_{t=0}^{T-1} \beta^t E[u(C_t)],$$

where u is the given utility function, β is the discount factor.

By using W_t and x_t as state variables, the DP model becomes

$$\begin{aligned} V_t(W_t, x_t) \quad = \quad & \max_{C_t, \Delta_t} \ u(C_t) + \beta E\left[V_{t+1}(W_{t+1}, x_{t+1})\right] \\ & \text{s.t.} \quad e^\top(\Delta_t + f(\Delta_t)) = M_t, \\ & \qquad X_{t+1} = R. * (x_t W_t + \Delta_t), \\ & \qquad W_{t+1} = e^\top X_{t+1} + R_f(W_t(1 - e^\top x_t) - C_t - M_t), \\ & \qquad x_{t+1} = X_{t+1}/W_{t+1}, \end{aligned}$$

where R_f is the riskfree return, $R = (R_1, \ldots, R_n)^\top$ is the random return vector of the risky assets, X_{t+1} is the vector of the amount of dollars in the risky assets at time rightly before $(t+1)$, Δ_t is the vector of amount of dollars with which buying or selling the risky assets, $.*$ is elementwise product, e is a column vector with 1 everywhere, and $f(\Delta_{ti})$ is the transaction cost function for buying or selling part of stock i with amount of Δ_{ti} dollars. We could let $f(\Delta_{ti}) = \tau\Delta_{ti}^2$, or $f(\Delta_{ti}) = \tau|\Delta_{ti}|$, or the hyperbolic form, for some constant $\tau > 0$. The terminal value function is $V_T(W, x) = u(W)$ (or $V_T(W, x) = u(W - \tau e^\top f(xW))$ when we assume that all risky assets have to be converted into the riskless asset before consumption) for some given utility function u.

If we do not allow shorting stocks or borrowing cash, then we just need to add the constraints: $x_t W_t + \Delta_t \geq 0$ and $W_t(1 - e^\top x_t) \geq M_t + C_t$. And the range of x_t and x_{t+1} is $[0, 1]^n$ while $e^\top x_{t+1} \leq 1$.

Assume that the utility function $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma > 0$ and $\gamma \neq 1$, and the transaction costs in buying or selling Δ amount of dollars of a stock is $\tau|\Delta|$. Then it can be shown that

$$V_t(W_t, x_t) = W_t^{1-\gamma} \cdot g_t(x_t),$$

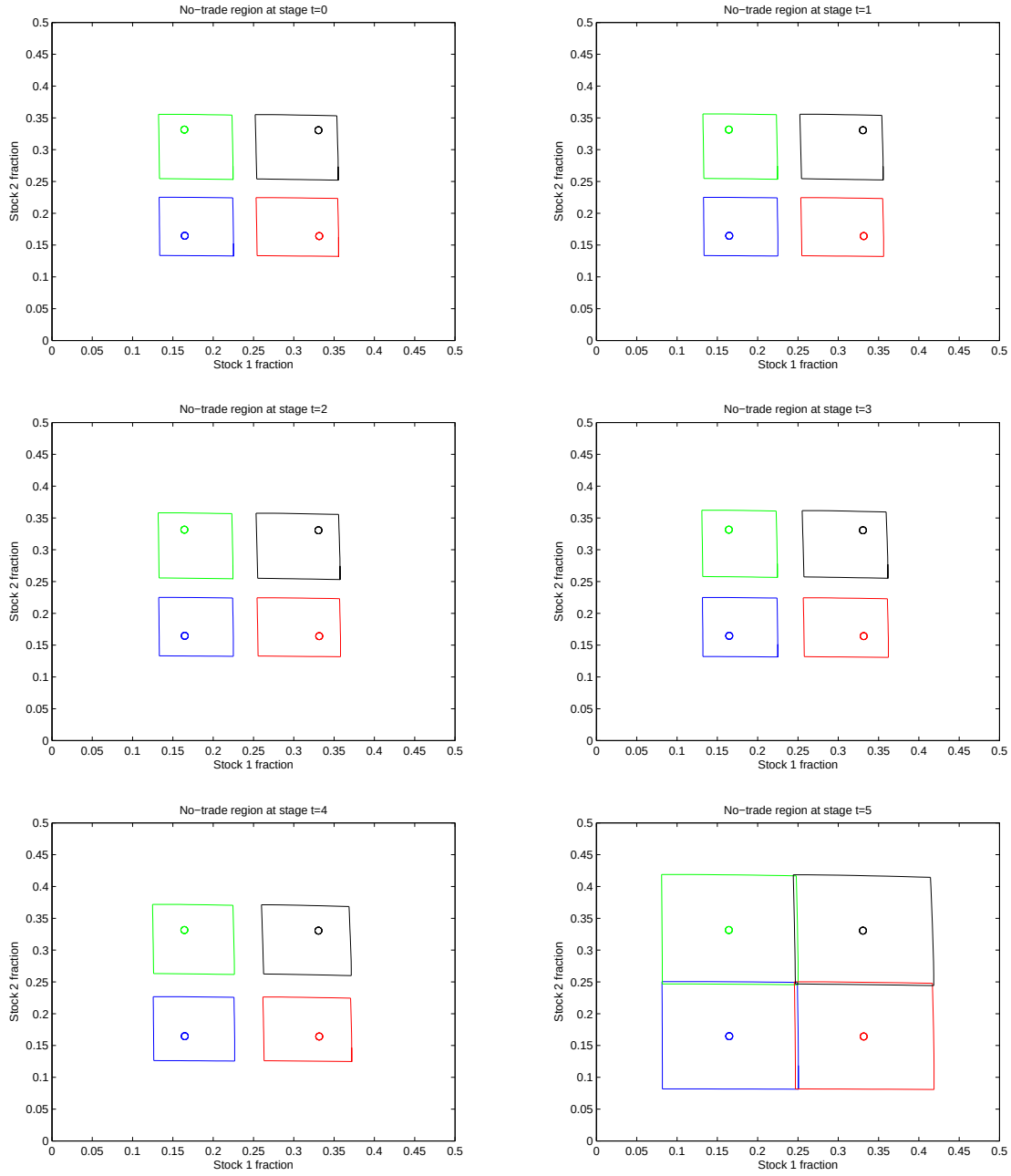


Figure 10.4: No-trade regions for 2 stocks with stochastic μ and 1 bond

where

$$\begin{aligned} g_t(x_t) &= \max_{c_t,\delta_t^+,\delta_t^-} u(c_t) + \beta E\left[\Pi_{t+1}^{1-\gamma} \cdot g_{t+1}(x_{t+1})\right] \\ &\quad \text{s.t.} \quad e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)) = m_t, \\ &\qquad s_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_f((1 - e^\top x_t) - c_t - m_t), \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

while $g_T(x) = 1/(1-\gamma)$ (or $g_T(x) = (1 - \tau e^\top |x|)^{1-\gamma}/(1-\gamma)$, when we assume that all risky assets have to be converted into the riskless asset before consumption).

If the utility function is $u(W) = \log(W)$, and there are proportional transaction costs in buying or selling stocks. Then we can show that

$$V_t(W_t, x_t) = \eta_t \log(W_t) + \psi_t(x_t),$$

where $\eta_t = 1 + \beta\eta_{t+1}$ with $\eta_T = 1$ (which implies that $\eta_t = (1 - \beta^{T-t+1})/(1-\beta)$), and

$$\begin{aligned} \psi_t(x_t) &= \max_{c_t,\delta_t^+,\delta_t^-} \log(c_t) + \beta E\left[\eta_{t+1}\log(\Pi_{t+1}) + \psi_{t+1}(x_{t+1})\right] \\ &\quad \text{s.t.} \quad e^\top(\delta_t^+ - \delta_t^- + \tau(\delta_t^+ + \delta_t^-)) = m_t, \\ &\qquad s_{t+1} = R. * (x_t + \delta_t^+ - \delta_t^-), \\ &\qquad \Pi_{t+1} = e^\top s_{t+1} + R_f((1 - e^\top x_t) - c_t - m_t), \\ &\qquad x_{t+1} = s_{t+1}/\Pi_{t+1}, \\ &\qquad \delta_t^+ \geq 0,\ \delta_t^- \geq 0, \end{aligned}$$

while $\psi_T(x) = 0$ (or $\psi_T(x) = \log(1 - \tau e^\top |x|)$, when we assume that all risky assets have to be converted into the riskless asset before consumption).

When there is no riskless asset, we just need to cancel the R_f term and replace m_t by $-c_t$ in the above models for g_t or ψ_t, while we should have $e^\top x_t = 1$ (the state variable vector $x_t = (x_{t1}, \ldots, x_{tn})$ should be changed as $(x_{t1}, \ldots, x_{t,n-1})$, and there is same cutoff in x_{t+1}).

From the separability of W and x, we see that the optimal portfolio rules are independent of wealth W_t. Thus the "no-trade" regions Ω_t are also independent of W_t, for the CRRA utility functions. Here the "no-trade" region Ω_t is defined as

$$\Omega_t = \{x_t/(1 - c_t^*) : \ (\delta_t^+)^* = (\delta_t^-)^* = 0\},$$

where $c_t^*, (\delta_t^+)^*, (\delta_t^-)^*$ are the optimal controls for the given x_t.

10.5.1 Numerical Examples

In this section, we will give several numerical examples, in which the number of risky assets is $n = 3$, and there is one riskless asset (called as bond later) available with return $R_f = e^r$, the terminal value function is $u(W) = W^{1-\gamma}/(1-\gamma)$, and the transaction cost function is $f(\Delta) = \tau|\Delta|$ with $\tau = 0.01$ for buying or selling Δ amount of money of risky assets.

By using the numerical DP method and the techniques discussed in section 10.5, we computed the "no-trade" regions for each stage in the following examples, when $n = 3$. In the numerical DP, we applied the NPSOL optimization package (see Gill, Murray, Saunders, and Wright [20]). In these examples, we assume that the risky asset return vector R is log-normal, i.e., $\log(R) \sim N((\mu - \sigma^2/2), \Lambda\Sigma\Lambda)$, where $\mu = (\mu_1, \ldots, \mu_n)^\top$, $\sigma = (\sigma_1, \ldots, \sigma_n)$, $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_n)$, and Σ is the correlation matrix of $\log(R)$.

The first example assumes that three stock returns are independently and identically distributed. Let $T = 50$, $\gamma = 3$, $\beta = 0.95$, $r = 0.04$, $\mu = (0.07, 0.07, 0.07)^\top$, $\sigma = (0.2, 0.2, 0.2)^\top$, and $\Sigma = \text{diag}(1, 1, 1)$. In the numerical DP, we applied the degree-9 complete Chebyshev approximation method and multi-dimensional product Gauss-Hermite quadrature rule with 9 nodes in each dimension. Our numerical results showed that the "no-trade" regions are close to cubes after canceling small perturbations, and

$$\begin{aligned}
\Omega_{49} &\approx [0.16, 0.33]^3, \\
\Omega_{48} &\approx [0.14, 0.24]^3, \ldots, \\
\Omega_t &\approx [0.19, 0.27]^3, \text{ for } t = 0, \ldots, 15, \\
c^*_{49} &\approx 0.51,\ c^*_{48} \approx 0.35,\ c^*_{47} \approx 0.27, \\
c^*_{46} &\approx 0.22,\ c^*_{47} \approx 0.19,\ c^*_{46} \approx 0.17, \ldots, \\
c^*_2 &\approx 0.054,\ c^*_1 \approx 0.0535,\ c^*_0 \approx 0.053.
\end{aligned}$$

So we see that the "no-trade" region and the optimal consumption decision converges to their infinite horizon limit as the time to the terminal time increases. From the discussion in section 9.5.1, we know that the optimal consumption decision for the infinite-horizon problem without transaction costs is

$$c^* = 1 - \left(\beta E[(R^\top x^* + R_f(1 - e^\top x^*))^{1-\gamma}]\right)^{1/\gamma} \approx 0.05,$$

where

$$x^* = (\Lambda\Sigma\Lambda)^{-1}(\mu - r)/\gamma = (0.25, 0.25, 0.25)^\top$$

is the Merton's ratio locating inside of converged "no-trade" region $\Omega \approx [0.19, 0.27]^3$. We see that the optimal consumption decision for the infinite-horizon problem with transaction costs is close to and a bit larger than the one without transaction costs.

The second example assumes that three stock returns are correlated. Let $T = 6$, $\gamma = 3$, $r = 0.04$,

$\mu = (0.07, 0.07, 0.07)^\top$, $\sigma = (0.2, 0.2, 0.2)^\top$, and

$$\Sigma = \begin{bmatrix} 1 & 0.2 & 0.2 \\ 0.2 & 1 & 0.04 \\ 0.2 & 0.04 & 1 \end{bmatrix}.$$

Our numerical results gave us the "no-trade" regions Ω_t for $t = 0, 1, 2, 3, 4, 5$, shown in Figure 10.5.

The third example assumes the same parameters with the above second example except that the correlation matrix is changed as

$$\Sigma = \begin{bmatrix} 1 & 0.4 & 0.4 \\ 0.4 & 1 & 0.16 \\ 0.4 & 0.16 & 1 \end{bmatrix}.$$

Our numerical results gave us the "no-trade" regions Ω_t for $t = 0, 1, 2, 3, 4, 5$, shown in Figure 10.6.

Notice that the faces of the "no-trade" regions seem to be flat, but in fact there are small perturbation on the faces, which might be due to numerical errors or the possibility that the exact "no-trade" region might have curvy faces.

10.6 Portfolio with Transaction Costs, Consumption and Wages

At the time rightly before reallocation time t, let W_t be the wealth of the portfolio, and let x_t be the allocation fractions of the wealth in n risky assets, while the fraction in the riskless asset is $(1 - e^\top x_t)$. Assume that the investor will receive wages w_t at each stage t ($w_t = 0$ for $t > T_R$, where T_R is the retire time), and w_t is a discrete Markov chain. The dynamic portfolio optimization problem is to find optimal portfolio and consumption decision C_t at each stage t such that we have a maximal expected total utility, i.e.,

$$\max \ \beta^T E[u(W_T)] + \sum_{t=0}^{T-1} \beta^t E[u(C_t)],$$

where u is the given utility function, β is the discount factor.

By using W_t, x_t, w_t as state variables, the DP model becomes

$$\begin{aligned} V_t(W_t, x_t, w_t) \quad = \quad & \max_{C_t, \Delta_t} \ u(C_t) + \beta E\left[V_{t+1}(W_{t+1}, x_{t+1}, w_{t+1})\right] \\ & \text{s.t.} \quad e^\top(\Delta_t + f(\Delta_t)) = M_t, \\ & \qquad X_{t+1} = R. * (x_t W_t + \Delta_t), \\ & \qquad W_{t+1} = e^\top X_{t+1} + R_f(W_t(1 - e^\top x_t) + w_t - C_t - M_t), \\ & \qquad x_{t+1} = X_{t+1}/W_{t+1}, \end{aligned}$$

where R_f is the riskfree return, $R = (R_1, \ldots, R_n)^\top$ is the random return vector of the risky assets, X_{t+1} is the vector of the amount of dollars in the risky assets at time rightly before $(t+1)$, Δ_t is the vector of amount of dollars with which buying or selling the risky assets, $.*$ is elementwise product, e is a column

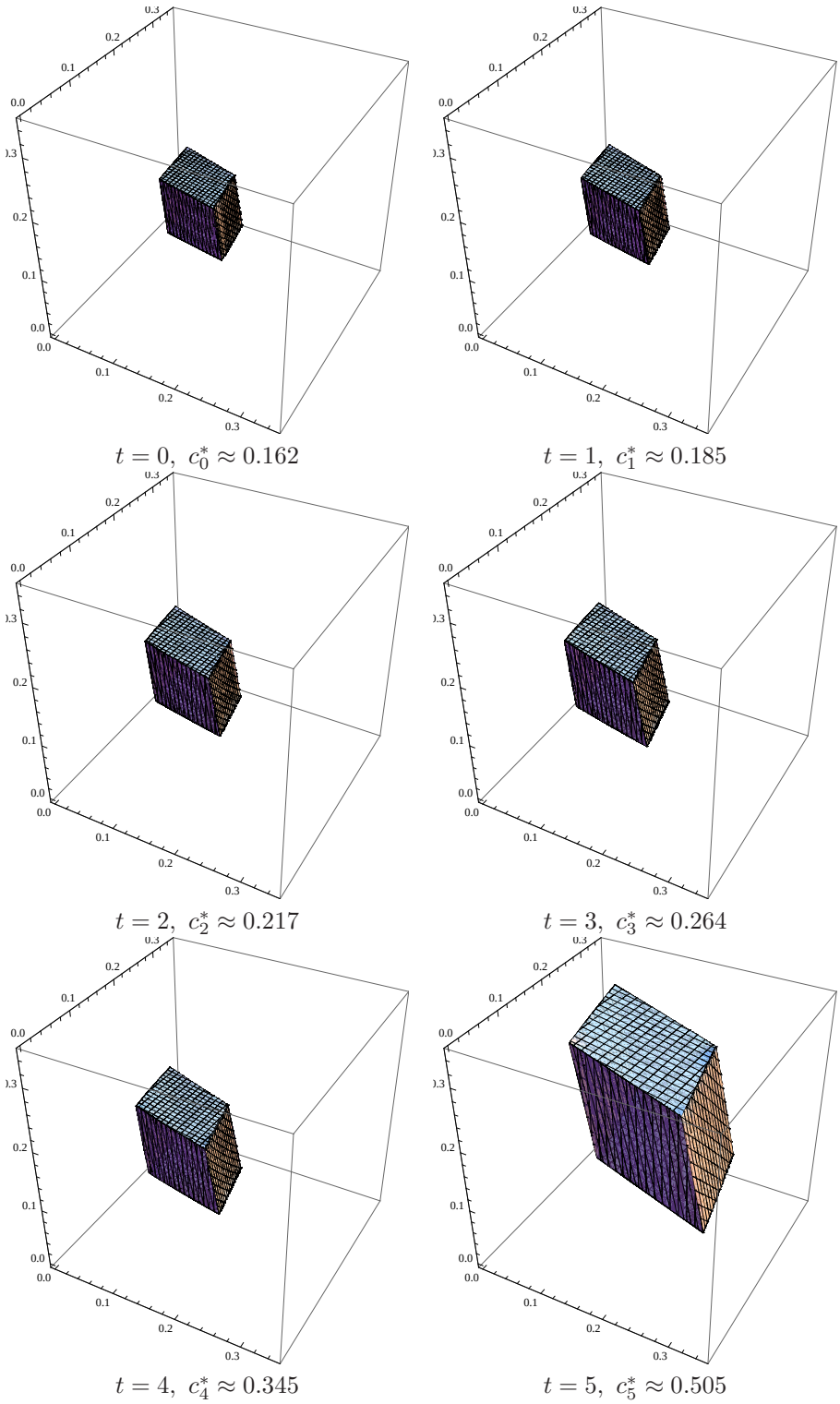
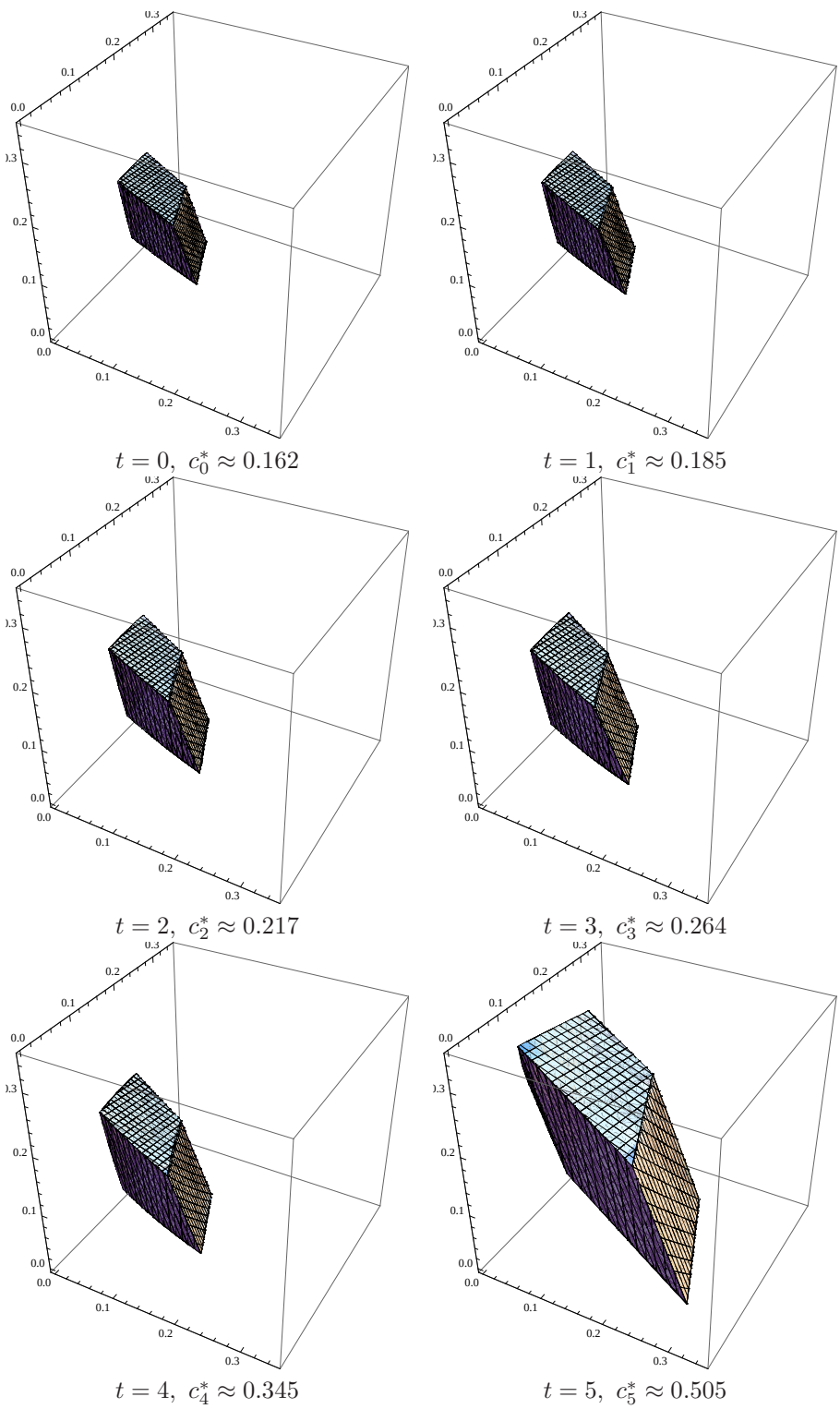


Figure 10.5: No-trade regions for 3 stocks and 1 bond with consumption (correlation: $\Sigma_{12} = \Sigma_{13} = 0.2$, $\Sigma_{23} = 0.04$)



$t = 0,\ c_0^* \approx 0.162$ $t = 1,\ c_1^* \approx 0.185$

$t = 2,\ c_2^* \approx 0.217$ $t = 3,\ c_3^* \approx 0.264$

$t = 4,\ c_4^* \approx 0.345$ $t = 5,\ c_5^* \approx 0.505$

Figure 10.6: No-trade regions for 3 stocks and 1 bond with consumption (correlation: $\Sigma_{12} = \Sigma_{13} = 0.4$, $\Sigma_{23} = 0.16$)

vector with 1 everywhere, and $f(\Delta_{ti})$ is the transaction cost function for buying or selling part of stock i with amount of Δ_{ti} dollars. We could let $f(\Delta_{ti}) = \tau\Delta_{ti}^2$, or $f(\Delta_{ti}) = \tau|\Delta_{ti}|$, or the hyperbolic form, for some constant $\tau > 0$. The terminal value function is $V_T(W_T, x_T, w_T) = u(W_T + w_T - \tau e^\top f(x_T W_T))$ for some given utility function u. In the model, only C_t and Δ_t is chosen as control variables, while M_t, X_{t+1}, W_{t+1} and x_{t+1} will be substituted.

If we do not allow shorting stocks or borrowing cash, then we just need to add the constraints: $x_t W_t + \Delta_t \geq 0$ and $W_t(1 - e^\top x_t) + w_t \geq M_t + C_t$. And the range of x_t and x_{t+1} is $[0, 1]^n$ while $e^\top x_{t+1} \leq 1$.

When there is no riskless asset, we just need to cancel the R_f term and replace m_t by $-c_t$ in the above models for g_t or ψ_t, while we should have $e^\top x_t = 1$ (the state variable vector $x_t = (x_{t1}, \ldots, x_{tn})$ should be changed as $(x_{t1}, \ldots, x_{t,n-1})$, and there is same cutoff in x_{t+1}).

Chapter 11

Portfolio with Options and Transaction Costs

In many cases, we would like to reduce or control the risk and amount of loss in the investment strategies. One way is to add put options into the set of portfolio assets as a hedging strategy. A put option is a financial contract that gives its holder the right to sell a certain amount of an underlying security at a specified price (called as strike price) within a specified time (called as time to maturity, or expiration time).

For simplicity, here we just assume that there is only one stock S_t and one put option P_t with an expiration time T and a strike price K based on this stock. Here P_t is denoted as the stochastic process of the price of a put option contract that gives its holder the right to sell one share of of the underlying stock at the strike price within the expiration time. It is simple to extend it to the case with multiple risky assets and/or multiple put/call options with different expiration times and/or strike prices, or some other derivatives.

11.1 DP Models for Portfolio Problems with Options

Assume that there is a put option with an expiration time T, and it is available for trading at each stage $t = 0, 1, \ldots, T$, with a price process P_t, while its underlying stock price process is S_t. At the time rightly before reallocation time t, assume that W_t is the wealth, X_t is the amount of money invested in the stock, and Y_t is the amount of money invested in the option. Since the price of the option is dependent on the stock price, we should add S_t as a state variable. Thus we will use W_t, S_t, X_t, Y_t as

state variables in the DP method. Thus the DP model is

$$\begin{aligned} V_t(W_t, S_t, X_t, Y_t) &= \max_{\Delta X_t, \Delta Y_t} E\left[V_{t+\Delta t}(W_{t+\Delta t}, S_{t+\Delta t}, X_{t+\Delta t}, Y_{t+\Delta t})\right] \\ &\quad \text{s.t.} \quad S_{t+\Delta t} = S_t R, \\ &\qquad X_{t+\Delta t} = R(X_t + \Delta X_t), \\ &\qquad Y_{t+\Delta t} = (Y_t + \Delta Y_t) P_{t+\Delta t}/P_t, \\ &\qquad M_t = \Delta X_t + \Delta Y_t + \tau_1 f(\Delta X_t) + \tau_2 f(\Delta Y_t), \\ &\qquad W_{t+\Delta t} = R_f(W_t - X_t - Y_t - M_t) + X_{t+\Delta t} + Y_{t+\Delta t}, \end{aligned}$$

where R is the random return of the stock in one period with length Δt, R_f is the riskfree return, ΔX_t is the amount of dollars for buying or selling the stock, ΔY_t is the amount of dollars for buying or selling the option, $f(.)$ is the transaction cost function, τ_1 and τ_2 are respectively the transaction cost ratios for buying or selling the stock and the option. The terminal value function is $V_T(W, S, X, Y) = u(W)$ (or $V_T(W, S, X, Y) = u(W - \tau_1 f(X) - \tau_2 f(Y))$ when we assume that all risky assets have to be converted into the riskless asset before consumption) for some given utility function u.

If we do not allow shorting in stock, option or bond, then we just need to add the constraints: $X_{t+\Delta t} \geq 0$, $Y_{t+\Delta t} \geq 0$ and $W_t - X_t - Y_t \geq M_t$, and the domain of (X_t, Y_t) can be set as $[0, W_t]^2$.

If the utility function is a CRRA utility with relative risk aversion coefficient $\gamma > 0$, and there are proportional transaction costs in buying or selling the stock or the option with the ratio τ, i.e., $f(x) = \tau|x|$, then let $\Delta X_t = W_t(\delta_{tx}^+ - \delta_{tx}^-)$ and $\Delta Y_t = W_t(\delta_{ty}^+ - \delta_{ty}^-)$ with $\delta_{tx}^+, \delta_{ty}^- \geq 0$ such that $f(\Delta X_t) = W_t(\delta_{tx}^+ + \delta_{tx}^-)$ and $f(\Delta Y_t) = W_t(\delta_{ty}^+ + \delta_{ty}^-)$. Let $x_t = X_t/W_t$ and $y_t = Y_t/W_t$. By using W_t, S_t, x_t, y_t as state variables, we can separate W_t and (S_t, x_t, y_t) in the value function $V_t(W_t, S_t, x_t, y_t)$.

When $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma > 0$ and $\gamma \neq 1$, we have

$$V_t(W_t, S_t, x_t, y_t) = W_t^{1-\gamma} \cdot g_t(S_t, x_t, y_t),$$

where

$$\begin{aligned} g_t(S_t, x_t, y_t) &= \max_{\delta_{tx}^+, \delta_{tx}^-, \delta_{ty}^+, \delta_{ty}^-} E\left[\Pi_{t+\Delta t}^{1-\gamma} \cdot g_{t+\Delta t}(S_{t+\Delta t}, x_{t+\Delta t}, y_{t+\Delta t})\right] \\ &\quad \text{s.t.} \quad S_{t+\Delta t} = S_t R, \\ &\qquad \xi_{t+\Delta t} = R(x_t + \delta_{tx}^+ - \delta_{tx}^-), \\ &\qquad \eta_{t+\Delta t} = (y_t + \delta_{ty}^+ - \delta_{ty}^-) P_{t+\Delta t}/P_t, \\ &\qquad m_t = \delta_{tx}^+ - \delta_{tx}^- + \delta_{ty}^+ - \delta_{ty}^- + \tau_1(\delta_{tx}^+ + \delta_{tx}^-) + \tau_2(\delta_{ty}^+ + \delta_{ty}^-), \\ &\qquad \Pi_{t+\Delta t} = R_f(1 - x_t - y_t - m_t) + \xi_{t+\Delta t} + \eta_{t+\Delta t}, \\ &\qquad x_{t+\Delta t} = \xi_{t+\Delta t}/\Pi_{t+\Delta t}, \\ &\qquad y_{t+\Delta t} = \eta_{t+\Delta t}/\Pi_{t+\Delta t}, \\ &\qquad \delta_{tx}^+, \delta_{tx}^-, \delta_{ty}^+, \delta_{ty}^- \geq 0, \end{aligned}$$

with $g_T(S,x,y) = 1/(1-\gamma)$ (or $g_T(x) = (1-\tau_1|x|-\tau_2|y|)^{1-\gamma}/(1-\gamma)$ when we assume that all risky assets have to be converted into the riskless asset before consumption).

When $u(W) = \log(W)$, we have

$$V_t(W_t, S_t, x_t, y_t) = \log(W_t) + \psi_t(S_t, x_t, y_t),$$

where

$$\begin{aligned}
\psi_t(S_t,x_t,y_t) \quad = \quad & \max_{\delta^+_{tx},\delta^-_{tx},\delta^+_{ty},\delta^-_{ty}} E\left[\log(\Pi_{t+\Delta t}) + \psi_{t+\Delta t}(S_{t+\Delta t}, x_{t+\Delta t}, y_{t+\Delta t})\right] \\
& \text{s.t.} \quad S_{t+\Delta t} = S_t R, \\
& \qquad \xi_{t+\Delta t} = R(x_t + \delta^+_{tx} - \delta^-_{tx}), \\
& \qquad \eta_{t+\Delta t} = (y_t + \delta^+_{ty} - \delta^-_{ty}) P_{t+\Delta t}/P_t, \\
& \qquad m_t = \delta^+_{tx} - \delta^-_{tx} + \delta^+_{ty} - \delta^-_{ty} + \tau_1(\delta^+_{tx} + \delta^-_{tx}) + \tau_2(\delta^+_{ty} + \delta^-_{ty}), \\
& \qquad \Pi_{t+\Delta t} = R_f(1 - x_t - y_t - m_t) + \xi_{t+\Delta t} + \eta_{t+\Delta t}, \\
& \qquad x_{t+\Delta t} = \xi_{t+\Delta t}/\Pi_{t+\Delta t}, \\
& \qquad y_{t+\Delta t} = \eta_{t+\Delta t}/\Pi_{t+\Delta t}, \\
& \qquad \delta^+_{tx}, \delta^-_{tx}, \delta^+_{ty}, \delta^-_{ty} \geq 0,
\end{aligned}$$

with $\psi_T(S,x,y) = 0$ (or $\psi_T(S,x,y) = \log(1-\tau_1|x|-\tau_2|y|)$ when we assume that all risky assets have to be converted into the riskless asset before consumption).

If we do not allow shorting in stock, option or bond, then we just need to add the constraints: $x_{t+\Delta t} \geq 0$, $y_{t+\Delta t} \geq 0$ and $1 - x_t - y_t \geq m_t$, and the domain of (x_t, y_t) can be set as $[0,1]^2$.

When there is no riskless asset, we just need to cancel the R_f term and replace m_t by 0 in the above models for g_t or ψ_t, while we should have $x_t + y_t = 1$ (the state variable y_t could be cancelled) in the case with only one stock and one derivative.

It is simple to extend the above models to include consumption decision at each stage, just like what we did in section 10.5.

11.2 Pricing Formulas for Put Option

Let the expiration time of the put option be T, and let its strike price be K. Then the payoff of the put option at the expiration time T is $\max(K - S_T, 0)$, where S_T is the price of the underlying stock at time T. We assume that the price S_t of the underlying asset follows a geometric Brownian motion with constant drift μ and volatility σ, the risk-free interest rate is r, and all other Black and Scholes [4] assumptions hold. Then the Black-Scholes formula gives us the risk-neutral price for the European put option at time $t < T$:

$$P_t = Ke^{-r(T-t)}\Phi(-d_2) - S_t\Phi(-d_1),$$

where

$$\begin{aligned} d_1 &= \frac{\log(S_t/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \\ d_2 &= d_1 - \sigma\sqrt{T-t}, \end{aligned}$$

and $\Phi(x)$ denotes the standard cumulative normal probability distribution. When we use the Black-Scholes formula as the pricing formula of the option, we should also assume that the one-period log-return of the underlying stock $\log(R) \sim N((\mu - \frac{\sigma^2}{2})\Delta t, \sigma^2\Delta t)$ for the consistency, where Δt is the time length of one period.

The Black-Scholes pricing formula of the European put option is dependent on the Black and Scholes [4] assumptions, for example, there are no transaction costs in buying or selling the stock or the option, and the stock can be traded continuously. But in our portfolio model, we assume that there are transaction costs in buying or selling the stock or the option, and the stock can only be traded at stages.

To solve the above conflict, we choose the binomial lattice method for pricing the put option numerically, see Luenberger [34]. The binomial lattice method can also be applied for pricing American-type options, while the Black-Scholes formula is only for European-type options.

In the binomial lattice model, we set h as a subperiod length and N as the number of subperiods such that $Nh = \Delta t$, where Δt is the length of one period in the multi-period portfolio optimization model. If the price is known as S at the beginning of a subperiod, the price at the beginning of the next subperiod is Su or Sd, with $u > 1$ and $0 < d < 1$. The probability of the upward movement is p, and the probability of the downward movement is $1 - p$. To match the log normal assumption of the stock return, we assume that

$$\begin{cases} p = \frac{1}{2} + \frac{\mu - \sigma^2/2}{2\sigma}\sqrt{h}, \\ u = \exp(\sigma\sqrt{h}), \\ d = \exp(-\sigma\sqrt{h}), \end{cases} \tag{11.1}$$

such that the expected growth rate of $\log(S)$ in the binomial model converges to $(\mu - \sigma^2/2)$, and the variance of the rate converges to σ^2, as h goes to zero. Let r be the riskfree interest rate, such that the riskfree return of one subperiod is $\exp(rh)$. Thus, the value of one subperiod put option on the stock governed by the binomial lattice is given as

$$P = \exp(-rh)(qP_u + (1-q)P_d),$$

where P_u and P_d are respectively the values at the upward branch and the downward branch at the end of this subperiod, q is the risk-neutral probability with $q = (\exp(rh) - d)/(u - d)$. Since the payoff of the put option at the expiration time T is $P_T = \max(K - S_T, 0)$, we can compute the price of the put option at any binomial nodes by applying the above backward iterative relation.

From the above binomial model, we know that the stock return of one period, R, has the following

probability distribution:

$$\Pr(R = u^k d^{N-k}) = \frac{N!}{k!(N-k)!} p^k (1-p)^{N-k},$$

for $k = 0, 1, \ldots, N$. And the riskfree return of one period is $R_f = (\exp(rh))^N = \exp(r\Delta t)$.

11.3 Numerical Examples

In all the examples in this section, we assume that the available assets for trading are one bond with a riskless annual rate $r = 0.04$, one stock S_t with a binormal return and a proportional transaction cost ratio τ_1, and one put option P_t with an expiration time T and a strike price K based on the stock and a proportional transaction cost ratio τ_2. The terminal value function is $u(W) = W^{1-\gamma}/(1-\gamma)$ with $\gamma = 3$. The length of one period is $\Delta t = 1/12$ year, i.e., one month. We apply the binomial lattice model (11.1) with $h = 1/120$, $\mu = 0.07$ and $\sigma = 0.2$. So the number of subperiods is $N = \Delta t/h = 10$. We assume that shorting the stock, the option or the bond are forbidden.

In the numerical DP method, we use the degree-11 complete Chebyshev polynomial approximation method with 12 Chebyshev nodes in each dimension of continuous states: (x_t, y_t), where x_t and y_t are respectively the fractions of wealth invested in the stock and the option at the time rightly before the reallocation time t. The optimization software is chosen as NPSOL.

In the first example, we let the transaction cost ratios $\tau_1 = \tau_2 = 0.01$, the strike price $K = S_0$, and the expiration time $T = 0.5$, so this is a 6-period portfolio optimization problem with an at-the-money put option. The domain of (x_t, y_t) is set as $[0, 1] \times [0, 0.2]$.

In Figure 11.1, we give the "no-trade" region and the optimal allocation fractions at stage 0 when the initial allocation has no options.

Since the payoff of put option at the expiration time is $\max(K - S_T, 0)$, we know that the put option is negatively and highly correlated to the underlying stock. The high correlation of these two risky assets implies that the objective function of the optimization problem is flat and the "no-trade" region should be a strip which is close to a line and the strip should be tilted toward one axis. The negative correlation implies that the strip should be tilted with a positive slope. All these properties can be observed from Figure 11.1: the strip's width is about 0.018, and its length is about 0.8, and its slope is about 0.09.

The dotted arrows in Figure 11.1 show the paths from the initial allocations without option to the optimal allocations. We see that when the initial stock allocation fraction is between 40% and 90% (and the initial option allocation is 0), the optimal allocation is to keep the stock and use cash to buy some of the put option.

For example, if the total amount of wealth is one million dollars at stage 0, and the current stock price is $S_0 = 100$ dollars, all the wealth is invested in the stock initially (there are 10000 shares), then the optimal transaction is to sell 744 shares of the stock to get 73657 dollars after paying 744 dollars for transaction costs, and then pay 54220 dollars to buy the put option with a contract size of 11727 shares (as the price of the put option is $0.0462351K$ and $S_0 = K$) after paying 542 dollars

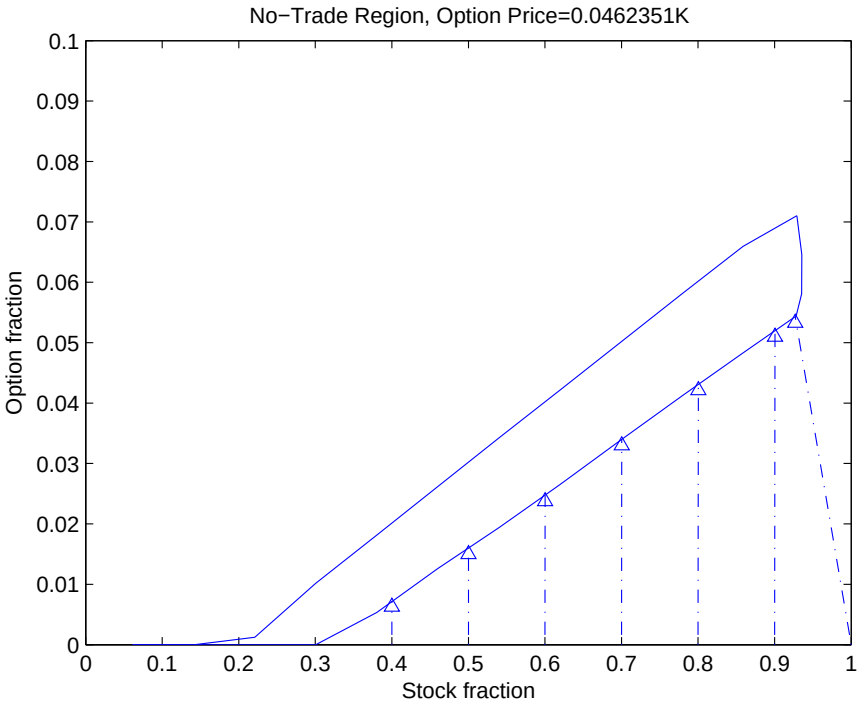


Figure 11.1: No-trade region for 1 stock, 1 put option, 1 bond

for transaction costs. So the total amount of transaction costs is $744 + 542 = 1286$ dollars, and the left amount of cash is $73657 - 54220 - 542 = 18895$ dollars, which will be invested in the bond. Thus, after all transactions, we have $10000 - 744 = 9256$ shares of the stock, the put option with a contract size of 11727 shares, and the bond worth of 18895 dollars, and the new total amount of wealth is $10^6 - 1286 = 998714$ dollars. That is, the optimal allocation fractions among the new total amount of wealth are, respectively, $925600/998714 \approx 92.68\%$ for the stock, $54220/998714 \approx 5.43\%$ for the put option, and $18895/998714 \approx 1.89\%$ for the bond.

Figure 11.2 tells us that the put option contributes to the value functions for the above 6-period dynamic portfolio problem with $\tau_1 = 0.01$, $K = S_0$ and $T = 0.5$ year (i.e., 6 periods). We see that all three value functions for the dynamic portfolio problem with the put option are bigger than the value function for the portfolio problem that has no options available for trading. Moreover, if the transaction cost ratio for the put option is larger, then the corresponding value function $V_0(W, S, x, y)$ at $y = 0$ for the portfolio problem with the put option is closer to the value function for the portfolio problem without options.

11.3.1 No-Trade Regions for Various Transaction Cost Ratios

In this subsection, we let the expiration time $T = 6$ months and the strike price is $K = S_0$, so this is a 6-period portfolio optimization problem with an at-the-money put option, and we come to observe the effect of various transaction cost ratios: $\tau_1 = 0.01, 0.005, 0.002$, and $\tau_2 = 2\tau_1, \tau_1, \tau_1/2$. See the figures

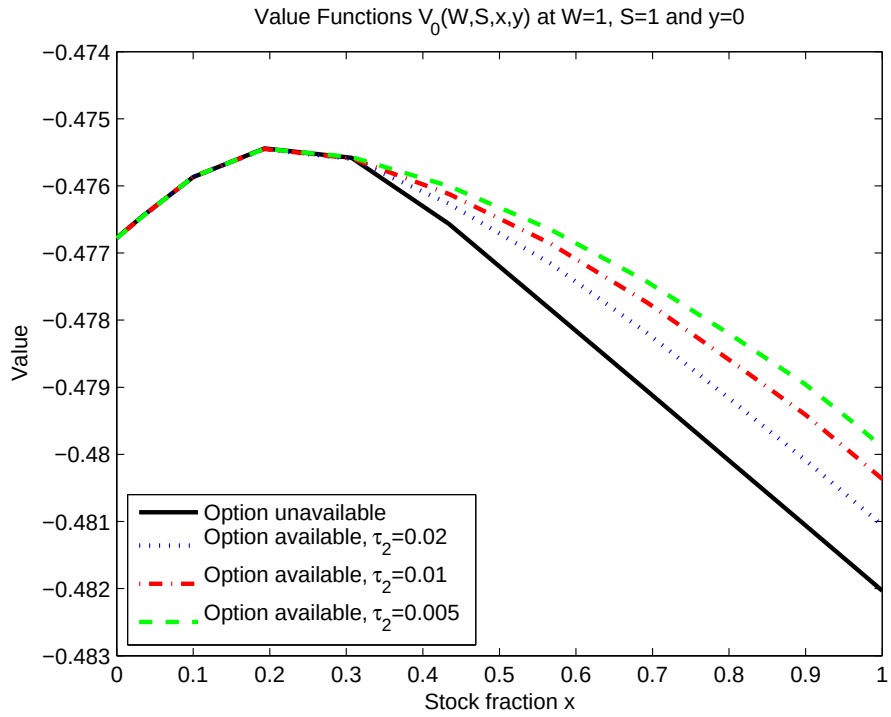


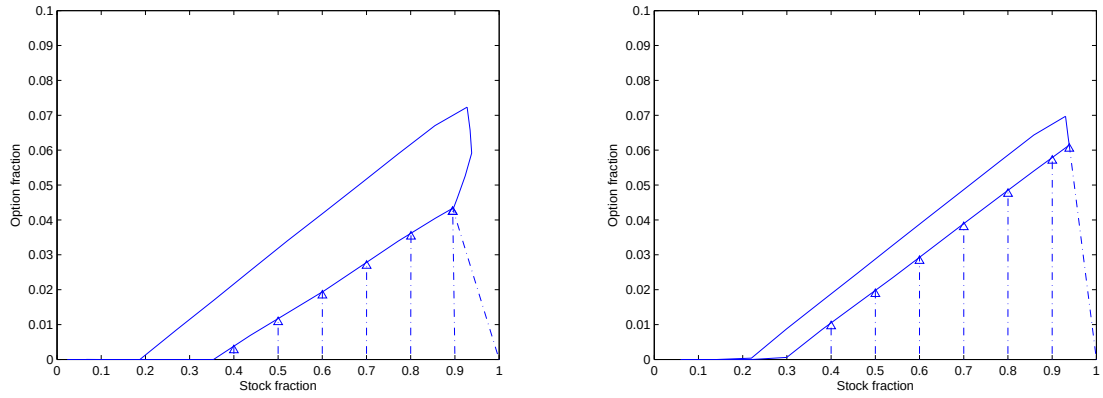
Figure 11.2: Value functions with/without options

11.3 and 11.4. The "no-trade" region for the case $\tau_1 = \tau_2 = 0.01$ has been shown in Figure 11.1, so we omit it here. Since the values of τ_1 and τ_2 has no impact on our option pricing formula, our put option prices for various τ_1 and τ_2 are same with its price when $\tau_1 = \tau_2 = 0.01$, which is $0.0462351K$. In the cases with $\tau_1 = 0.01, 0.005$ and $\tau_2 = 2\tau_1, \tau_1, \tau_1/2$, the domain of (x_t, y_t) is set as $[0,1] \times [0, 0.2]$. And in the cases with $\tau_1 = 0.002$ and $\tau_2 = 2\tau_1, \tau_1, \tau_1/2$, the domain of (x_t, y_t) is set as $[0,1] \times [0, 0.1]$.

We see that the "no-trade" region becomes thinner as τ_1 and/or τ_2 is smaller, and it becomes shorter as τ_1 is smaller, which match the intuition, and in fact the "no-trade" region will converge to a point as $\tau_1 \to 0$ and $\tau_2 \to 0$. Notice that in the cases when $\tau_2 = 2\tau_1 = 0.01$ or $\tau_2 = 2\tau_1 = 0.004$, if the initial allocation for stock is bigger than 33% or 30% respectively, and there is no allocation for option initially, then the optimal transaction is to sell stocks and put all the earned cash into the bond (do not buy the put option) such that the optimal allocation for stock is 33% or 30% respectively among the wealth after transactions.

11.3.2 No-Trade Regions for Various Expiration Times

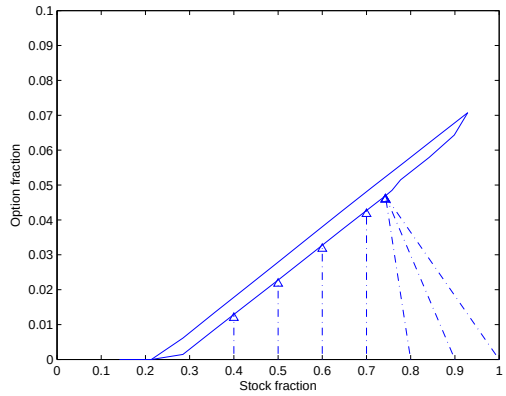
We let $\tau_1 = \tau_2 = 0.01$, $K = S_0$ in this subsection. And we choose various expiration times $T = j$ months, , which implies j-period portfolio allocation problems with an at-the-money put option, for $j = 1, \ldots, 6$. See Figure 11.5. The "no-trade" region for the case $T = 6$ month is omitted here as



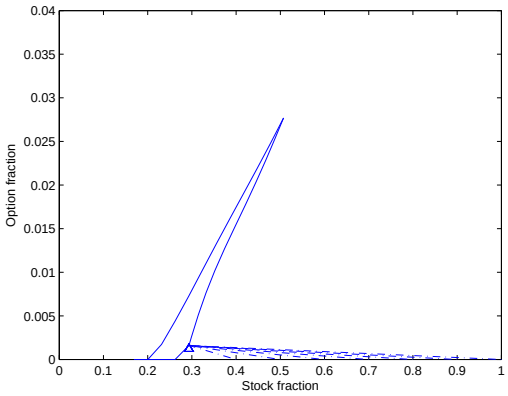
$\tau_1 = 0.01,\ \tau_2 = 0.02$ $\tau_1 = 0.01,\ \tau_2 = 0.005$

$\tau_1 = 0.005,\ \tau_2 = 0.01$ $\tau_1 = 0.005,\ \tau_2 = 0.005$

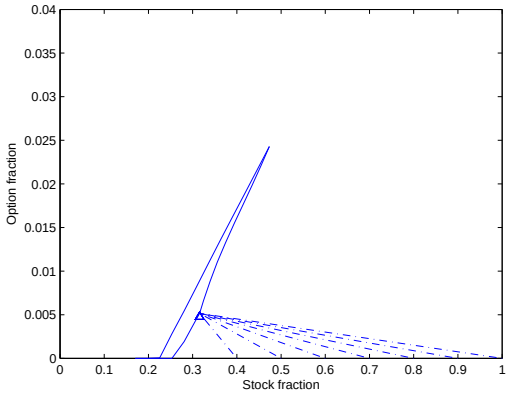
Figure 11.3: No-trade regions for 1 stock, 1 put option, 1 bond (various transaction cost ratios, part 1)



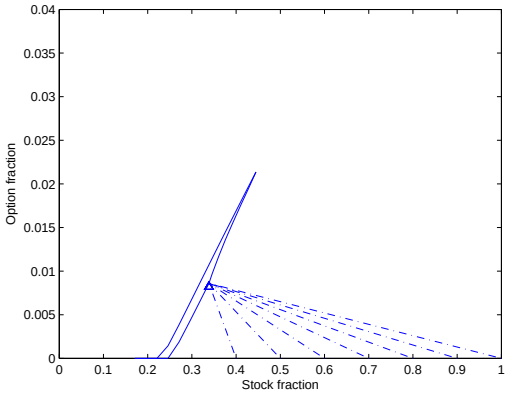
$\tau_1 = 0.005,\ \tau_2 = 0.0025$



$\tau_1 = 0.002,\ \tau_2 = 0.004$



$\tau_1 = 0.002,\ \tau_2 = 0.002$



$\tau_1 = 0.002,\ \tau_2 = 0.001$

Figure 11.4: No-trade regions for 1 stock, 1 put option, 1 bond (various transaction cost ratios, part 2)

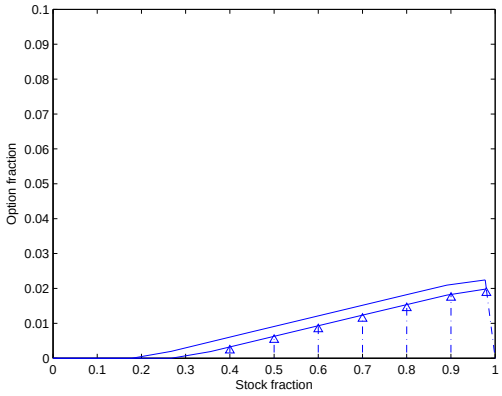
it has been shown in the first example of this section (Figure 11.1). The domain of (x_t, y_t) is set as $[0, 1] \times [0, 0.2]$ for all the expiration times.

We see that the "no-trade" region strip becomes thinner and more tilted toward the horizon axis when the time to maturity is smaller.

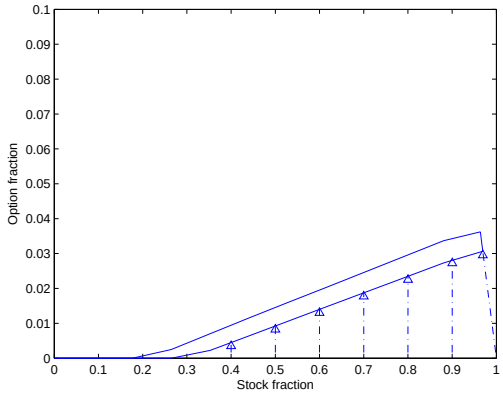
11.3.3 No-Trade Regions for Various Strike Prices

We let $\tau_1 = \tau_2 = 0.01$, the expiration time is $T = 6$ months, which implies 6-period portfolio problems with a put option. And we choose one of three kinds of European put options: out-of-the-money one with $K = 0.8S_0$, at-the-money one with $K = S_0$, and in-the-money one with $K = 1.2S_0$. See Figure 11.6. The at-the-money case has been shown in the first example of this section (Figure 11.1), so we omit it here. The domain of (x_t, y_t) is set as $[0, 1] \times [0, 0.1]$ for the case with $K = 0.8S_0$, while it is $[0, 1] \times [0, 0.5]$ for the case with $K = 1.2S_0$,.

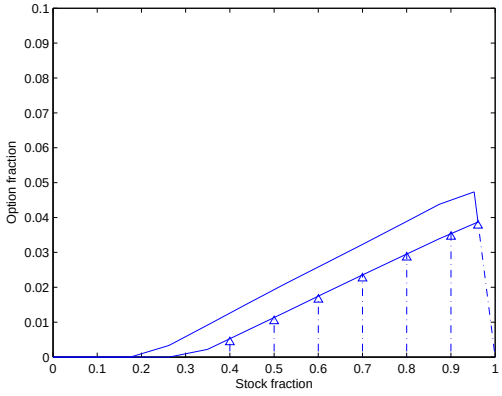
Notice that the range of the vertical axis is different in the above figures. We see that the "no-trade" region strip becomes thinner and more tilted toward the horizon axis as the strike price goes lower.



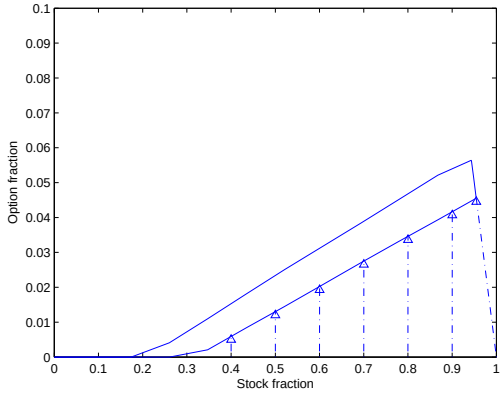
$T = 1$ month



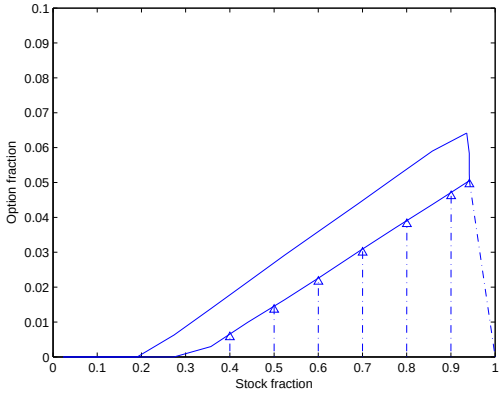
$T = 2$ months



$T = 3$ months

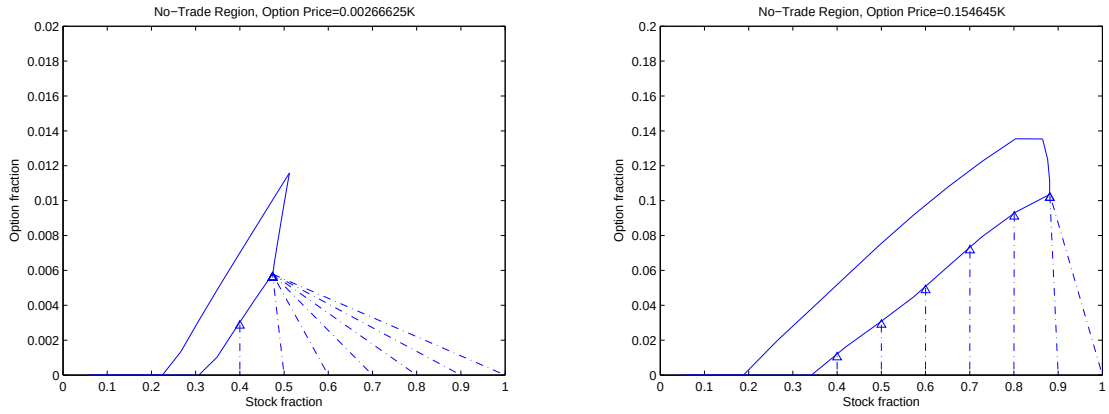


$T = 4$ months



$T = 5$ months

Figure 11.5: No-trade regions for 1 stock, 1 put option, 1 bond (various expiration times)



$K = 0.8S_0$ $K = 1.2S_0$

Figure 11.6: No-trade regions for 1 stock, 1 put option, 1 bond (various strike prices)

Bibliography

[1] Robert A. Abrams and Uday S. Karmarkar. Optimal multiperiod investment-consumption policies. *Econometrica*, 48(2):333–353, 1980.

[2] Marianne Akian, Josi Luis Menaldi, and Agnis Sulem. On an investment-consumption model with transaction costs. *SIAM Journal of Control and Optimization*, 34(1):329–364, 1996.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[4] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[5] D. Blackwell. Discounted dynamic programming. *Annals of Mathematical Statistics*, 36:226–235, 1965.

[6] Phelim P. Boyle and Xiaodong Lin. Optimal portfolio selection with transaction costs. *North American Actuarial Journal*, 1(2):27–39, 1997.

[7] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. *KNITRO: an integrated package for nonlinear optimization*, 2006.

[8] Yongyang Cai. Explicit solution for continuous-time portfolio selection under constraints. Working paper.

[9] Alexis Collomb. *Approximate value iteration approaches to constrained dynamic portfolio problems*. PhD thesis, MIT, 2004.

[10] G. M. Constantinides. Optimal portfolio revision with proportional transaction costs: extension to HARA utility functions and exogenous deterministic income. *Management Science*, 22(8):921–923, 1976.

[11] G. M. Constantinides. Multiperiod consumption and investment behavior with convex transaction costs. *Management Science*, 25:1127–1137, 1979.

[12] G. M. Constantinides. Capital market equilibrium with transaction costs. *J. Polit. Econ.*, 94(4):842–862, 1986.

[13] M. H. A. Davis and A. R. Norman. Portfolio selection with transaction costs. *Mathematics of Operations Research*, 15(4):676–713, 1990.

[14] E. Denardo. Contraction mappings underlying the theory of dynamic programming. *SIAM Review*, 9:165–177, 1967.

[15] Darrell Duffie and Tong sheng Sun. Transactions costs and portfolio choice in a discrete-continuous-time setting. *Journal of Economic Dynamics and Control*, 14:35–51, 1990.

[16] J.C. Fiorot and J. Tabka. Shape-preserving C^2 cubic polynomials interpolating splines. *Mathematics of Computation*, 57(195):291–298, 1991.

[17] Gerard Gennotte and Alan Jung. Investment strategies under transaction costs: the finite horizon case. *Management Science*, 40(3):385–404, 1994.

[18] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[19] Philip E. Gill, Walter Murray, and Michael A. Saunders. *User's guide for SNOPT version 7: software for large-scale nonlinear programming*, 2007.

[20] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for NPSOL 5.0: a Fortran package for nonlinear programming. Technical report, SOL, 1994.

[21] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[22] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization*. Addison-Wesley, 1990.

[23] M. Grant, S. Boyd, and Y. Ye. *CVX: Matlab software for disciplined convex programming*, 2006. Available at `http://www.stanford.edu/boyd/cvx`.

[24] Aparna Gupta. *Optimum asset allocation with behavior utilities: a plan for acquiring and consuming retirement funds*. PhD thesis, Stanford University, 2000.

[25] Gerd Infanger. *Planning under Uncertainty: Solving Large-Scale Stochastic Linear Programs*. Personal Transformation Press, 1994.

[26] Gerd Infanger. *Dynamic asset allocation using a stochastic dynamic programming approach. In: Handbook of Asset and Liability Management*, volume 1, chapter 5. North Holland, 2006.

[27] Karel Janecek and Steven E. Shreve. Asymptotic analysis for optimal investment and consumption with transaction costs. *Finance Stochast.*, 8(2):181–206, 2004.

[28] Kenneth L. Judd. *Numerical Methods in Economics*. The MIT Press, 1998.

[29] Jules H. Kamin. Optimal portfolio revision with a proportional transaction cost. *Management Science*, 21(11):1263–1271, 1975.

[30] H.J. Kushner. Numerical methods for stochastic control problems in continuous time. *SIAM Journal on Control and Optimization*, 28:999–1048, 1990.

[31] Aatos Lahtinen. Shape preserving interpolation by quadratic splines. *Journal of Computational and Applied Mathematics*, 29:15–24, 1990.

[32] Aatos Lahtinen. On the choice of parameters in shape-preserving quadratic spline interpolation. *Journal of Computational and Applied Mathematics*, 39:109–113, 1992.

[33] H. Liu. Optimal consumption and investment with transaction costs and multiple risky assets. *Journal of Finance*, 59:289–338, 2004.

[34] David Luenberger. *Investment Science*. Oxford University Pressy, 1997.

[35] Wilfredo Leiva Maldonado and Benar Fux Svaiter. On the accuracy of the estimated policy function using the Bellman contraction method. *Economics Bulletin*, 3(15):1–8, 2001.

[36] J. Mattingley and S. Boyd. *CVXMOD - Convex optimization software in python*, 2008. Available at http://cvxmod.net.

[37] Robert Merton. Lifetime portfolio selection under uncertainty: the continuous time case. *The Review of Economics and Statistics*, 51(3):247–257, 1969.

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

[39] Jan Mossin. Optimal multiperiod portfolio policies. *The Journal of Business*, 41(2):215–229, 1968.

[40] Walter Murray and Francisco J. Prieto. A sequential quadratic programming algorithm using an incomplete solution of the subproblem. *SIAM Journal on Optimization*, 5(3):590–640, 1995.

[41] Kumar Muthuraman and Sunil Kumar. Multidimensional portfolio optimization with proportional transaction costs. *Mathematical Finance*, 16(2):301–335, 2006.

[42] Kumar Muthuraman and Sunil Kumar. Solving free-boundary problems with applications in finance. *Foundations and Trends in Stochastic Systems*, 1(4):259–341, 2008.

[43] Kumar Muthuraman and Haining Zha. Simulation-based portfolio optimization for large portfolios with transaction costs. *Mathematical Finance*, 18(1):115–134, 2008.

[44] Steven Pruess. Shape-preserving C^2 cubic spline interpolation. *IMA Journal of Numerical Analysis*, 13:493–507, 1993.

[45] T.J. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*. New York: Wiley-Interscience, 1990.

[46] Mark Rubinstein. A strong case for generalized logarithmic utility model as the premier model of financial markets. *The Journal of Finance*, 31(2):551–571, 1976.

[47] John Rust. *Dynamic programming. In: New Palgrave Dictionary of Economics, ed. by Steven N. Durlauf and Lawrence E. Blume*. Palgrave Macmillan, second edition, 2008.

[48] Paul Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *The Review of Economics and Statistics*, 51(3):239–246, 1969.

[49] Manuel S. Santos and Jesus Vigo-Aguiar. Analysis of a numerical dynamic programming algorithm applied to economic models. *Econometrica*, 66(2):409–426, 1998.

[50] Larry L. Schumaker. On shape-preserving quadratic spline interpolation. *SIAM Journal of Numerical Analysis*, 20:854–864, 1983.

[51] John Stachurski. Continuous state dynamic programming via nonexpansive approximation. *Comput Econ*, 31:141–160, 2008.

[52] A. Stroud and D. Secrest. *Gaussian Quadrature Formulas*. NJ: Prentice Hall, Englewood Cliffs, 1966.

[53] Alexander Wang. *Approximate value iteration approaches to constrained dynamic portfolio problems*. PhD thesis, MIT, 2004.

[54] Edward Zabel. Consumer choice, portfolio decisions, and transaction costs. *Econometrica*, 41(2):321–335, 1973.