Shape-Preserving Dynamic Programming

Kenneth Judd and Yongyang Cai

July 20, 2011

1 Introduction

The multi-stage decision-making problems are numerically challenging. When the problems are time-separable, dynamic programming (DP) is a popular method to solve them. 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. Discretization of state variables can approximate the value functions, but it is very time consuming to get a good approximation. Polynomial or spline approximation will save much time, but computational errors may accumulate through the value function iterations. However, if the value functions are concave and their approximation preserves the concavity, then computational error accumulation problem could be solved, see Santos and Vigo-Aguiar (1998), and Maldonado and Svaiter (2001).

In this paper, we present a shape-preserving DP algorithm with value function iteration for solving the discrete-time decision-making problems with continuous states. The paper is constructed as follows. Section 2 introduces the parametric DP algorithm and describes numerical methods in the algorithm. Section 3 presents the shape-preserving DP algorithm with value function iteration. Section 4 and Section 5 give some numerical examples for optimal growth problems and multi-stage portfolio optimization problems respectively to show the stability and accuracy of the shape-preserving DP.

2 Numerical Methods for DP

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, $V(x) \approx \hat{V}(x; \mathbf{c})$, where $\mathbf{c}$ is a vector of 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 the vector of parameters, $\mathbf{c}$, such that $\hat{V}(x; \mathbf{c})$ approximately satisfies the Bellman equation. Numerical DP with value function iteration can solve the Bellman equation approximately (see Judd (1998)).

A general DP model is based on the Bellman equation:

$$\begin{aligned} V_t(x) = \max_{a \in \mathcal{D}(x,t)} \quad & u_t(x,a) + \beta E\{V_{t+1}(x^+) \mid x, a\}, \\ \text{s.t.} \quad & x^+ = g(x,a), \end{aligned}$$

where $V_t(x)$ is called the value function at stage t, x^+ is the next-stage state (may be random) conditional on the current-stage state x and the action a, $\mathcal{D}(x,t)$ is a feasible set of a, and $u_t(x,a)$ is the utility function at time t. The following is the algorithm of parametric DP with value function iteration for finite horizon problems.

Algorithm 1. *Numerical Dynamic Programming with Value Function Iteration for Finite Horizon Problems*

Initialization. Choose the approximation nodes, $X_t = \{x_{it} : 1 \leq i \leq m_t\}$ for every $t < T$, and choose a functional form for $\hat{V}(x; \mathbf{c})$. Let $\hat{V}(x; \mathbf{c}^T) \equiv u_T(x)$. Then for $t = T-1, T-2, \ldots, 0$, iterate through steps 1 and 2.

Step 1. Maximization step. Compute

$$\begin{aligned} v_i = \max_{a_i \in \mathcal{D}(x_i,t)} \quad & u_t(x_i,a_i) + \beta E\{\hat{V}(x_i^+; \mathbf{c}^{t+1}) \mid x_i, a_i\} \\ \text{s.t.} \quad & x_i^+ = g(x_i,a_i), \end{aligned}$$

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

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

There are three main components in numerical DP: optimization, approximation, and numerical integration. In the following we focus on discussing approximation and omit the introduction of optimization and numerical integration.

A linear approximation scheme consists of two parts: basis functions and approximation nodes. Approximation methods can be classified as either spectral methods or finite element methods. A spectral method uses globally nonzero basis functions $\phi_j(x)$ and defines $\hat{V}(x;\mathbf{c}) = \sum_{j=0}^{n} c_j\phi_j(x)$ to be the degree n approximation. In our examples, we use Chebyshev polynomial interpolation, which is a spectral method. In contrast, a finite element method uses locally basis functions $\phi_j(x)$ that are nonzero over sub-domains of the approximation domain. Examples of finite element methods include piecewise linear interpolation, Schumaker interpolation, cubic splines, and B-splines. See Judd (1998), Cai (2009), and Cai and Judd (2010) for more details.

Piecewise linear interpolation is a common way applied by many scholars because of its simplicity and shape-preservation. But piecewise linear interpolation has its disadvantage: it is a challenge for optimization software to find the optimal solution in the maximization step of the numerical DP algorithm, because the approximation $\hat{V}(x)$ is only continuous but not differentiate at the nodes x_i.

Here we give a brief introduction of Chebyshev polynomials. Chebyshev basis polynomials on $[-1,1]$ are defined as $T_j(x) = \cos(j\cos^{-1}(x))$, while general Chebyshev basis polynomials on $[a,b]$ are defined as $T_j((2x-a-b)/(b-a))$ for $j = 0,1,2,\ldots$. The degree n Chebyshev polynomial approximation for $V(x)$ is

$$\hat{V}(x;\mathbf{c}) = \sum_{j=0}^{n} c_j T_j\left((2x-a-b)/(b-a)\right).$$

With a set of Chebyshev nodes x_i and the Lagrange data set $\{(x_i, v_i) : i = 1,\ldots,m\}$, the coefficients c_j can be calculated easily by Chebyshev regression algorithm (see Judd (1998) and Cai (2009)).

3 Shape-preserving DP

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. Discretization method and piecewise linear interpolation method preserve the shape property. Another shape-preserving method is the so-called Schumaker shape-preserving interpolation method (Schumaker (1983)). A revised version of Schumaker interpolation is given in Cai (2009) and Cai and Judd (2011). Fiorot and Tabka (1991), and Steven Pruess (1993) gave some other shape-preserving splines approximation methods. Judd and Solnick (1994) discussed some theoretical properties of the shape-preserving splines in numerical DP and applied them in optimal growth problems. Wang and Judd (2000) applied a bivariate shape-preserving spline interpolation method in numerical DP to solve a savings allocation problem.

For a univariate approximation problem, Schumaker interpolation preserves the shape properties including monotonicity and concavity. But many approximation methods such as Chebyshev interpolation/approximation do not have the shape-preserving property. So we can use a shape-preserving approximation by adding shape constraints such that the shape properties still hold.

Here we will extend them to a more general shape-preserving approximation method and its application in numerical DP.

3.1 Shape-preserving Chebyshev Interpolation

One problem for Chebyshev interpolation 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 some pre-specified nodes.

For the univariate Chebyshev polynomial approximation on $[a,b]$, we have $\hat{V}(x;\mathbf{c}) = \sum_{j=0}^{n} c_j T_j(z)$ with $z = \frac{2x-a-b}{b-a}$, so

$$\hat{V}'(x;\mathbf{c}) = \frac{2}{b-a}\sum_{j=0}^{n} c_j T_j'(z),$$

and

$$\hat{V}''(x;\mathbf{c}) = \frac{4}{(b-a)^2}\sum_{j=0}^{n} c_j T_j''(z).$$

Therefore, given the Lagrange data $\{(x_i, v_i) : 1 \le i \le m\}$ generated by the maximization step of Algorithm 1, if we know that the value function is strictly increasing and concave, then in the fitting step of Algorithm 1, we could get a shape-preserving Chebyshev interpolation by solving the following problem:

$$\begin{aligned}
\min_{c_j} \quad & \sum_{j=0}^{n} \frac{1}{(j+1)^2}|c_j|, \\
\text{s.t.} \quad & \sum_{j=0}^{n} c_j T_j'(y_i) > 0, \quad i = 1,\ldots,m', \\
& \sum_{j=0}^{n} c_j T_j''(y_i) < 0, \quad i = 1,\ldots,m', \\
& \sum_{j=0}^{n} c_j T_j(z_i) = v_i, \quad i = 1,\ldots,m,
\end{aligned}$$

for a degree-n Chebyshev interpolation on $[a, b]$. In the model, $z_i = (2x_i - a - b)/(b - a)$ $(i = 1, \ldots, m)$ are the interpolation nodes in $[-1, 1]$, and y_i $(i = 1, \ldots, m')$ are pre-specified nodes in $[-1, 1]$ for shape-preserving constraints.

To cancel the absolute operator in the above model, we replace c_j by $c_j^+ - c_j^-$ with $c_j^+, c_j^- \geq 0$ such that $|c_j| = c_j^+ + c_j^-$. Thus, the above model becomes a linear programming (LP) problem:

$$\begin{aligned} \min_{c_j} \quad & \sum_{j=0}^{n} \frac{1}{(j+1)^2}(c_j^+ + c_j^-), \\ \text{s.t.} \quad & \sum_{j=0}^{n} c_j T_j'(y_i) > 0, \quad i = 1, \ldots, m', \\ & \sum_{j=0}^{n} c_j T_j''(y_i) < 0, \quad i = 1, \ldots, m', \\ & \sum_{j=0}^{n} c_j T_j(z_i) = v_i, \quad i = 1, \ldots, m, \\ & c_j = c_j^+ - c_j^-, \quad j = 1, \ldots, n, \\ & c_j^+ \geq 0, \quad c_j^- \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{1}$$

In our numerical examples in Sections 4 and 5, we let z_i be the Chebyshev interpolation nodes in $[-1, 1]$ (i.e., $z_i = -\cos((2i - 1)\pi/(2m))$), and let y_i be the equally-spaced nodes in $[-1, 1]$.

Usually we should let $m' > m$ in order that the approximation function have the shape-preservation on enough nodes. Moreover, the number of basis terms of Chebyshev polynomial should not be smaller than the number of Chebyshev interpolation nodes. That is, we should let $n \geq m-1$ so that both m interpolation equality constraints and $2m'$ shape-preserving constraints could hold in the above model. In fact, most of coefficient of basis terms with a degree higher than $m - 1$ are 0 or close to 0 in most cases, so it will have almost the same computation time with the standard Chebyshev interpolation in the objective function of the maximization step of numerical DP algorithm.

From $T_j(y) = \cos(j \cos^{-1}(y))$, we have

$$T_j'(y) = \frac{j \sin(j \cos^{-1}(y))}{\sqrt{1 - y^2}}$$

and

$$T_j''(y) = \frac{y T_j'(y) - j^2 T_j(y)}{1 - y^2},$$

for any $y \in (-1,1)$, while $T_j'(1) = j^2$, $T_j'(-1) = (-1)^{j+1}j^2$, $T_j''(1) = (j^4 - j^2)/3$ and $T_j''(-1) = (-1)^j(j^4 - j^2)/3$, for $j = 0, 1, 2, \ldots$.

However, we can also use the following recursive formulas to evaluate $T_j(y)$, $T_j'(y)$ and $T_j''(y)$ for any $y \in [-1,1]$:

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

and

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

and

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

4 Examples for Optimal Growth Problems

We first illustrate our methods with a discrete-time optimal growth problem with one good and one capital stock. It is to find the optimal consumption function and the optimal labor supply function such that the total utility over the T-horizon time is maximal, i.e.,

$$\begin{aligned}
V_0(k_0) = \max_{c,l} \quad & \sum_{t=0}^{T-1} \beta^t u(c_t, l_t) + \beta^T u_T(k_T), \\
\text{s.t.} \quad & k_{t+1} = F(k_t, l_t) - c_t, \quad 0 \le t < T, \\
& \underline{k} \le k_t \le \bar{k}, \quad 1 \le t \le T, \\
& c_t \ge \epsilon, \ l_t \ge \epsilon, \quad 0 \le t < T,
\end{aligned} \tag{2}$$

where k_t is the capital stock at time t with k_0 given, c_t is the consumption, l_t is the labor supply, $\underline{k}$ and $\bar{k}$ are given lower and upper bound of k_t, β is the discount factor, $F(k,l) = k + f(k,l)$ with $f(k_t, l_t)$ the aggregate net production function, and $u(c_t, l_t)$ is the utility function, where c_t is consumption of the good, and l_t

is the labor supply, and ϵ is a small positive number to avoid the nonpositive consumption or labor supply. This objective function is time-separable.

We use the following numerical examples of the finite horizon optimal growth model to illustrate the importance of the shape-preserving property. In the following examples, we let $\alpha = 0.25$, $\beta = 0.95$, $\gamma = 8$, $\eta = 1$, $A = (1-\beta)/(\alpha\beta)$ and $T = 50$. Let the range of k be $[0.1, 10]$, i.e., $\underline{k} = 0.1$ and $\bar{k} = 10$. And we choose $\epsilon = 10^{-6}$ in the model (2). The production function is $f(k,l) = Ak^{\alpha}l^{1-\alpha}$, and the terminal value function is

$$u_T(k) = \frac{u(f(k,1),1)}{1-\beta}.$$

We see that the terminal value function is smooth and concave, and the optimal controls will not be binding at least at the next-to-the-last stage $t = T-1$. Thus, it is supposed that polynomial approximation methods could approximate the value functions well. However, in our numerical examples in this section, we found that shape-preservation is still very important in numerical DP: shape-preserving Chebyshev interpolation has more stable and accurate solutions than the one without shape-preservation.

4.1 Scaling Technique

A typical utility function in optimal growth problems is a power utility with the following form

$$u(c,l) = \frac{c^{1-\gamma}}{1-\gamma} - B\frac{l^{1+\eta}}{1+\eta}$$

where $B = (1-\alpha)A^{1-\gamma}$, while the aggregate net production function is $f(k,l) = Ak^{\alpha}l^{1-\alpha}$ with $A = (1-\beta)/(\alpha\beta)$. Thus the steady state of the infinite horizon deterministic optimal growth problems is $k_{ss} = 1$ while the optimal consumption and the optimal labor supply at k_{ss} are respectively $c_{ss} = A$ and $l_{ss} = 1$.

However, when β is close to 1 and γ is large, A will be small and the optimal consumption at various levels of capital k will also be small, such that the utility function will have a large magnitude. We know a large magnitude in computation will often incur numerical errors in computation and optimization. An appropriate scaling will improve the computational accuracy, while the scaling does not affect the optimal solutions.

In the following finite horizon optimal growth examples, we will choose a scaled power utility function

$$u(c,l) = \frac{(c/A)^{1-\gamma} - 1}{1-\gamma} - (1-\alpha)\frac{l^{1+\eta} - 1}{1+\eta}.$$

We know this scaling will have the same optimal solutions for consumption and labor supply. And this scaling technique is very helpful in getting good solutions from numerical DP.

4.2 Solve Exactly with Large-Scale Optimizers

For the finite horizon optimal growth problem (2), when T is small, we can use a good large-scale optimization package to solve the problem directly, and its solution could be better than the solution of (3) given by numerical DP algorithms because of the numerical approximation errors. But when T is large, the solution of (3) given by numerical DP algorithms is usually better than the solution of (2) given by a large-scale numerical optimization package directly. In addition, if the problem becomes stochastic, i.e., the value function form becomes $V_t(x, \theta_t)$ where θ_t is a discrete time Markov chain, then it usually becomes infeasible for an optimization package to solve the stochastic problem directly with high accuracy when $T > 10$. But numerical DP algorithms can still solve it well, see Cai (2009).

In the examples of this section, we choose to solve finite horizon deterministic optimal growth problems with $T \leq 100$, so we will use the solutions of the model (2) given by SNOPT (Gill, Murray and Saunders (2005)) in GAMS code as the "true" solutions.

The scaling technique discussed in the above section is also very helpful in getting good solutions from large-scale optimizers when γ and T is large. For example, in our numerical example with $\gamma = 8$ and $T = 50$, when SNOPT is applied as the large-scale optimizer, the solution without using the scaling technique is not as good as the one using the scaling technique.

4.3 DP Solution

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

$$\begin{aligned} V_t(k) &= \max_{c,l} \quad u(c,l) + \beta V_{t+1}(k^+), \\ & \quad\ \text{s.t.} \quad k^+ = F(k,l) - c, \\ & \qquad\qquad \underline{k} \leq k^+ \leq \bar{k}, \quad c \geq \epsilon, \ l \geq \epsilon, \end{aligned} \tag{3}$$

which is the Bellman (1957) equation. Here k is the state variable and (c, l) are the control variables, and $V_T(k) = u_T(k)$.

The program code is written in GAMS, and we choose SNOPT (Gill, Murray and Saunders (2005)) as the optimization solver. We choose $m = 40$ Chebyshev interpolation nodes on $[0.1, 10]$.

Figure 1 show the relative errors of optimal controls at each stage in L^∞ and L^1 respectively. We use the solutions given by directly applying SNOPT in the model (2) as the "true" solutions. The solid lines are errors of solutions given by numerical DP algorithm with standard degree-39 Chebyshev polynomial interpolation using $m = 40$ Chebyshev interpolation nodes in the model (3). The dashed lines are errors of solutions given by numerical DP algorithm with shape-preserving Chebyshev polynomial interpolation with $m = 40$ Chebyshev interpolation nodes and $m' = 100$ equally-spaced nodes for shape constraints from the models (3) and (1).

From Figure 1, we see that the solid lines for optimal controls are much higher than the corresponding dashed lines at the first steps, and then they are close each other. This means that the shape-preservation really helps a lot in obtaining more stable and accurate solutions in numerical DP algorithms.

5 Example for Multi-stage Portfolio Optimization Problems

We also illustrate our methods with a multi-stage portfolio optimization problem. 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 the 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 $S_t = (S_{t1}, \ldots, S_{tn})^\top$ is a vector of money invested in the n risky assets at time t, then money invested in the riskless asset is $B_t = W_t - e^\top S_t$, where e is a column vector of 1s. Thus, the wealth at the next stage is

$$W_{t+1} = R_f(W_t - e^\top S_t) + R^\top S_t, \tag{4}$$

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

A simple multi-stage portfolio optimization problem is to find an optimal portfolio S_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 (4) with a given W_0, and u is the terminal utility function, and $E\{\cdot\}$ is the expectation operator.

In this section, we present a numerical example with one stock and one bond available for investment. We assume that the number of periods is $T = 6$, the bond has a riskfree return $R_f = 1.04$, and the stock has a discrete random return

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

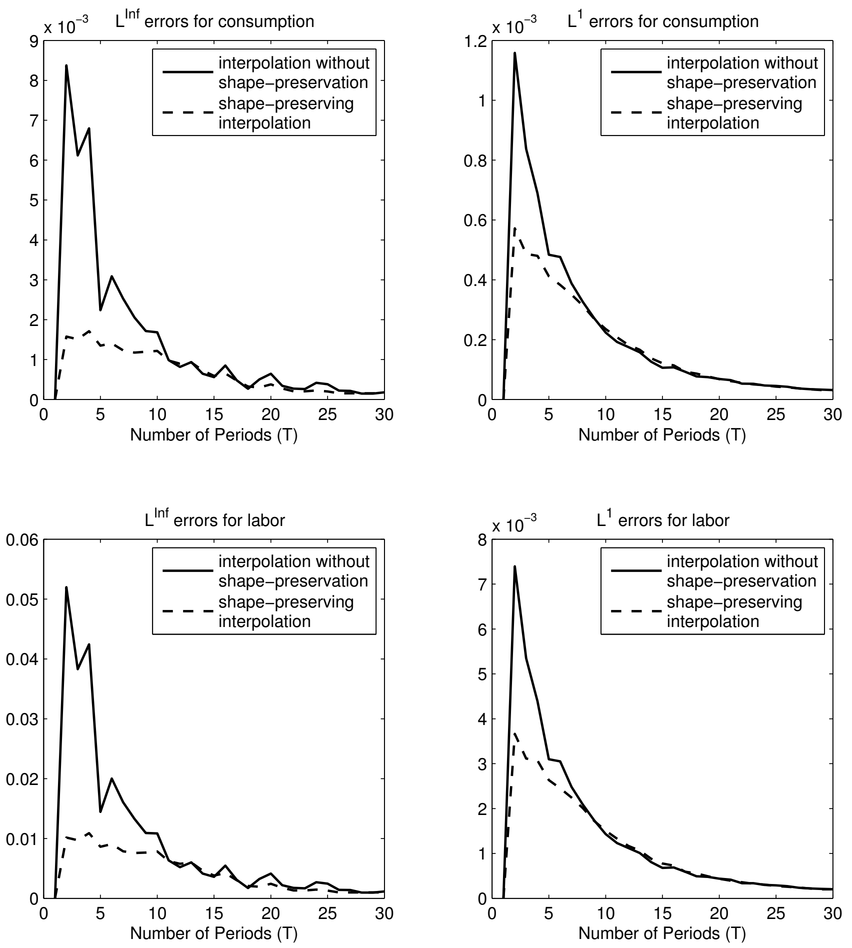


Figure 1: Errors of Numerical DP with Chebyshev interpolation with/without shape-preservation

Let the range of initial wealth W_0 as $[0.9, 1.1]$. The terminal utility function is

$$u(W) = \frac{(W-K)^{1-\gamma}}{1-\gamma}$$

with $\gamma = 4$ and $K = 0.4$ so that the terminal wealth should be always bigger than 0.4. Moreover, we assume that borrowing or shorting is not allowed in this example, i.e., $B_t \geq 0$ and $S_t \geq 0$ for all t.

5.1 Tree Method

In the portfolio optimization problem, if we discretize the random returns of n stocks as $R = R^{(j)} = (R_{1,j}, \dots, R_{n,j})$ with probability q_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

$$P_{t+1,k} = P_{t,[(k-1)/m]+1} q_{\text{mod}(k,m)+1},$$

is the probability of scenario k at stage $t+1$, and

$$W_{t+1,k} = W_{t,[(k-1)/m]+1}(R_f B_{t,[(k-1)/m]+1} + \sum_{i=1}^{n} R_{i,\text{mod}(k,m)+1} S_{i,t,[(k-1)/m]+1}),$$

is the wealth at scenario k and stage $t+1$, for $1 \leq k \leq m^{t+1}$ and $0 \leq t < T$. Here, $W_{0,1} = W_0$ is a given initial wealth, $P_{0,1} = 1$, $\text{mod}(k,m)$ is the remainder of division of k by m, and $[(k-1)/m]$ is the quotient of division of $(k-1)$ by m. We should add $B_{t,k} \geq 0$ and $S_{t,k} \geq 0$ for all t, k as bound constraints in the tree model, if neither shorting stock or borrowing bond is allowed.

Figure 2 shows one simple tree with $m = 2$ and $T = 2$ for a portfolio with one bond and one stock ($n = 1$). The stock's random return has a probability q_1 to have a return $R_{1,1}$, and the probability $q_2 = 1 - q_1$ to have a return $R_{1,2}$. So there are two scenarios at stage 1: $(W_{1,1}, P_{1,1})$ and $(W_{1,2}, P_{1,2})$, and four scenarios at stage 2: $(W_{2,1}, P_{2,1})$, ..., $(W_{2,4}, P_{2,4})$.

The disadvantage of the tree method is that when m or T is large, the problem size will exponentially increase and it will be a big challenge for an optimizer to find an accurate solution. But the disadvantage will disappear if we use DP algorithms to solve the problem. Since the numerical example in this section is not large for the above tree model, the exact optimal allocations can be calculated by the tree model and MINOS optimization package (Murtagh and Saunders (1978)) in AMPL code.

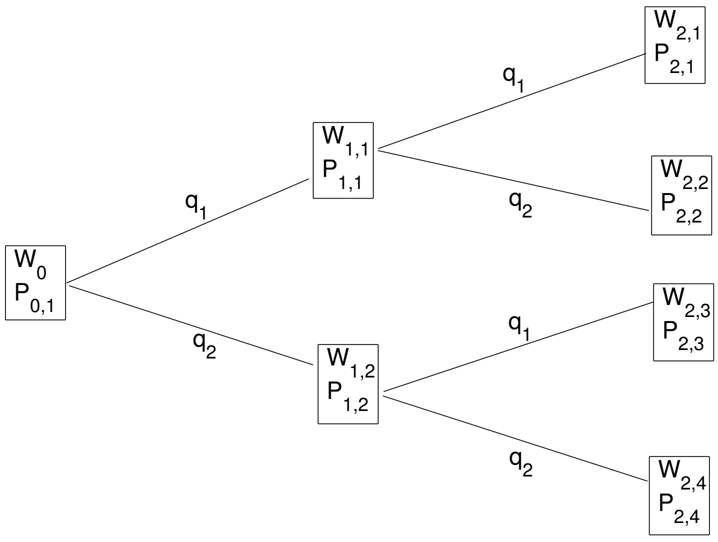


Figure 2: A binary tree with two periods

Figure 3 shows the optimal bond allocation B_t and stock allocation S_t, for $t = 0, 1, \dots, 5$, computed by the tree method for the numerical example.

Next, we will use the exact optimal allocations computed by the tree method to test stability and accuracy of our shape-preserving algorithms.

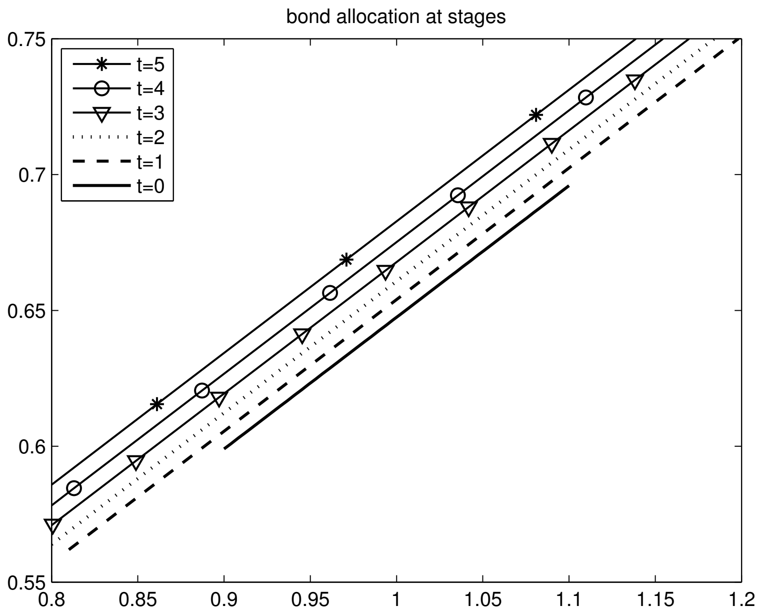
5.2 DP Solution

The DP model of this multi-stage portfolio optimization problem is

$$\begin{aligned} V_t(W) &= \max_{B,S} \quad E\{V_{t+1}(R_f B + R^\top S)\}, \\ & \qquad \text{s.t.} \quad B + e^\top S = W, \end{aligned}$$

for $t = 0, 1, \dots, T-1$, where W is the state variable and S is the control variable vector, and the terminal value function is $V_T(W) = u(W)$. We should add $B \geq 0$ and $S \geq 0$ as bound constraints in the above DP model, if neither shorting stock or borrowing bond is allowed.

Since the terminal utility function is $u(W) = (W-K)^{1-\gamma}/(1-\gamma)$, this implies that there should be no possibility to let the terminal wealth W_T be lower than K. It follows that the lower bound of wealth at stage t should be not less than KR_f^{t-T}. Thus, since we do not allow shorting or borrowing and R is bounded in



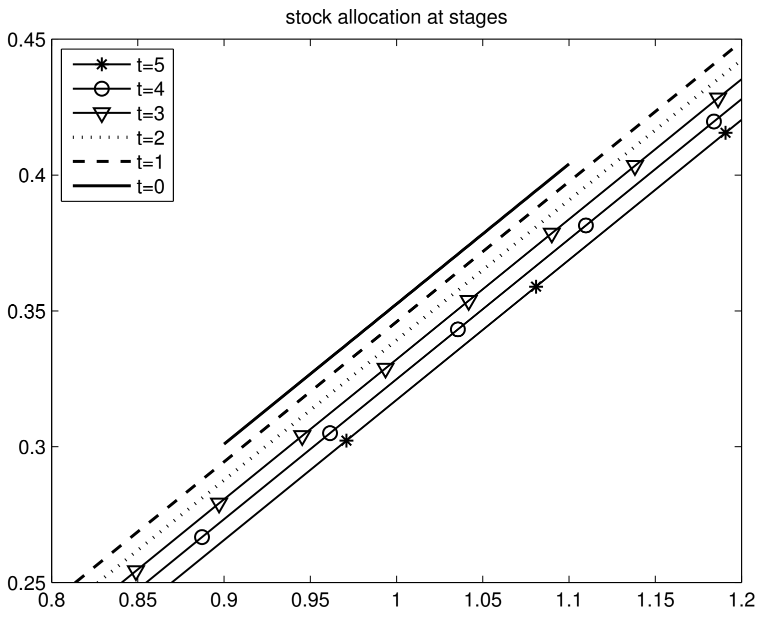


Figure 3: Exact optimal allocations

this example, the ranges $[\underline{W}_t, \overline{W}_t]$ can be computed in an iterative way:

$$\begin{aligned}
\underline{W}_{t+1} &= \max\left\{\min(R)\underline{W}_t, KR_f^{t-T}\right\} \\
\overline{W}_{t+1} &= \max(R)\bar{W}_t,
\end{aligned}$$

with a given initial wealth bound $[\underline{W}_0,\ \overline{W}_0]$.

Specifically, for the numerical example with $K = 0.4$, $R_f = 1.04$, $\min(R) = 0.9$ and $\max(R) = 1.4$, after we choose $\underline{W}_0 = 0.9$ and $\overline{W}_0 = 1.1$, we have

$$\begin{aligned}
[\underline{W}_1, \overline{W}_1] &= [0.81, 1.54], \\
[\underline{W}_2, \overline{W}_2] &= [0.729, 2.156], \\
[\underline{W}_3, \overline{W}_3] &= [0.656, 3.018], \\
[\underline{W}_4, \overline{W}_4] &= [0.59, 4.226], \\
[\underline{W}_5, \overline{W}_5] &= [0.531, 5.916], \\
[\underline{W}_6, \overline{W}_6] &= [0.478, 8.282].
\end{aligned}$$

We see that the range is expanding exponentially along time t. If we use a fixed range along time t in our numerical DP algorithms, then it will definitely reduce the accuracy of solutions. So here we choose the above ranges at stages $t = 0, 1, \ldots, 5$ in both the maximization step and the fitting step of Algorithm 1.

We choose $m = 30$ Chebyshev interpolation nodes in $[\underline{W}_t, \overline{W}_t]$ for each stage $t = 0, 1, \ldots, 5$. The computational results of numerical DP algorithms are given by our GAMS code. We choose SNOPT (Gill, Murray and Saunders (2005)) as the optimization solver of the nonlinear programming problems in the maximization step of Algorithm 1, and CPLEX as the solver of the linear programming problems for shape-preserving Chebyshev interpolation in the fitting step of Algorithm 1. And the ranges $[\underline{W}_t, \overline{W}_t]$ are given in the previous iterative way.

Figure 4 shows the relative errors of numerical DP algorithms with Chebyshev interpolation with/without shape-preservation. The vertical axis values of plotted x-marks and squares for their corresponding horizontal axis value W_t (wealth), are given as log of relative errors, i.e.,

$$\log_{10}\left(\frac{\left|S^*_{t,DP} - S^*_t\right|}{|S^*_t|}\right),$$

where S^*_t are true optimal stock allocations for the wealth W_t at stage t which are given by the tree method, and $S^*_{t,DP}$ are computed optimal stock allocation from numerical DP algorithms with Chebyshev interpolation with/without shape-preservation. The squares are errors of solutions given by numerical DP algorithm with standard degree-29 Chebyshev polynomial interpolation using $m = 30$

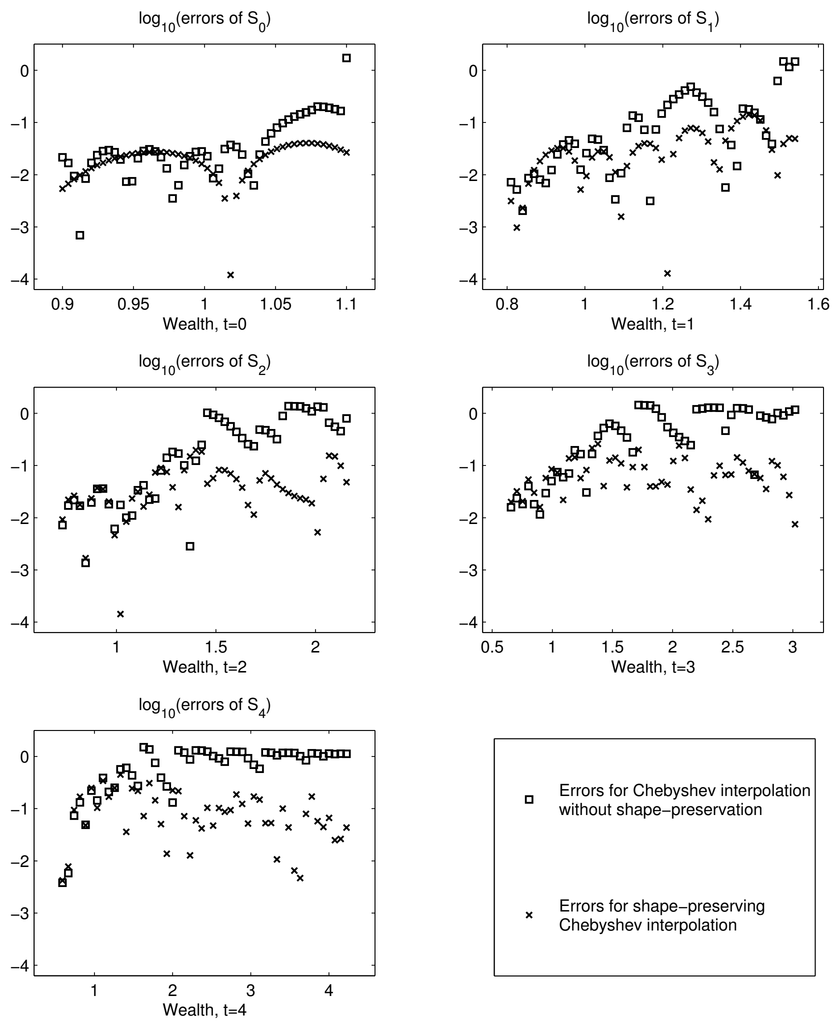


Figure 4: Relative Errors of Optimal Stock Allocations from Numerical DP with Chebyshev interpolation with/without shape-preservation

Chebyshev interpolation nodes in the model (3). The x-marks are errors of solutions given by numerical DP algorithm with shape-preserving Chebyshev polynomial interpolation with $m = 30$ Chebyshev interpolation nodes and $m' = 100$ equally-spaced nodes for shape constraints from the models (3) and (1).

From Figure 4, we see that most of x-marks (errors for shape-preserving Chebyshev interpolation) have higher accuracy than their corresponding squares (errors for standard Chebyshev interpolation), at all stages $t = 0, 1, 2, 3, 4$.

This means that the shape-preservation really helps a lot in obtaining more stable and accurate solutions in numerical DP algorithms.

6 Conclusion

This paper presents a general shape-preserving DP algorithm and shows that shape-preserving property of the value function approximation is critical in stability and accuracy of numerical dynamic programming.

References

[1] Bellman, Richard (1957). *Dynamic Programming.* Princeton University Press.

[2] Byrd, Richard H., and Jorge Nocedal and Richard A. Waltz (2006). "KNITRO: an integrated package for nonlinear optimization".

[3] Cai, Yongyang (2009). *Dynamic Programming and Its Application in Economics and Finance.* PhD thesis, Stanford University.

[4] Cai, Yongyang, and Kenneth Judd (2010). "Stable and efficient computational methods for dynamic programming". *Journal of the European Economic Association*, Vol. 8, No. 2-3, 626–634.

[5] Cai, Yongyang, and Kenneth Judd (2011). "Dynamic programming with Hermite information".

[6] Fiorot, J.C., and J. Tabka (1991). "Shape-preserving C^2 cubic polynomial interpolating splines". *Mathematics of Computation,* 57(195), 291–298.

[7] Gill, Philip, Walter Murray, Michael Saunders, and Margaret Wright (1994). "User's Guide for NPSOL 5.0: a Fortran Package for Nonlinear Programming". Technical report, SOL, Stanford University.

[8] Judd, Kenneth (1998). *Numerical Methods in Economics.* The MIT Press.

[9] Judd, Kenneth, and Andrew Solnick (1994). "Numerical dynamic programming with shape-preserving splines".

[10] Maldonado, Wilfredo L., and Benar Fux Svaiter (2001). "On the accuracy of the estimated policy function using the Bellman contraction method". *Economics Bulletin*, 3(15), 1–8.

[11] Murtagh, Bruce, and Michael Saunders (1978). "Large-scale linearly constrained optimization". *Mathematical Programming*, 14, 41–72.

[12] Pruess, Steven (1993). "Shape-preserving C^2 cubic spline interpolation". *IMA Journal of Numerical Analysis*, 13, 493–507.

[13] Rust, John (2008). "Dynamic Programming". In: *New Palgrave Dictionary of Economics*, ed. by Steven N. Durlauf and Lawrence E. Blume. Palgrave Macmillan, second edition.

[14] Santos, Manuel S., and Jesus Vigo-Aguiar (1998). "Analysis of a numerical dynamic programming algorithm applied to economic models". *Econometrica*, 66(2), 409–426.

[15] Schumaker, Larry (1983). "On Shape-Preserving Quadratic Spline Interpolation". *SIAM Journal of Numerical Analysis*, 20, 854–864.

[16] Wang, Sheng-Pen, and Kenneth L. Judd (2000). "Solving a savings allocation problem by numerical dynamic programming with shape-preserving interpolation". *Computers & Operations Research*, 27(5), 399–408.