Math Meth Oper Res
DOI 10.1007/s00186-012-0406-5

ORIGINAL ARTICLE

Shape-preserving dynamic programming

Yongyang Cai · Kenneth L. Judd

Abstract Dynamic programming is the essential tool in dynamic economic analysis. Problems such as portfolio allocation for individuals and optimal growth of national economies are typical examples. Numerical methods typically approximate the value function and use value function iteration to compute the value function for the optimal policy. Polynomial approximations are natural choices for approximating value functions when we know that the true value function is smooth. However, numerical value function iteration with polynomial approximations is unstable because standard methods such as interpolation and least squares fitting do not preserve shape. We introduce shape-preserving approximation methods that stabilize value function iteration, and are generally faster than previous stable methods such as piecewise linear interpolation.

Keywords Numerical dynamic programming · Shape-preserving approximation · Multi-stage decision-making problems · Value function iteration

1 Introduction

All dynamic economic problems are multi-stage decision problems, and their nonlinearities make them numerically challenging. Dynamic programming (DP) is the standard approach for any time-separable problem. If state variables and control variables are continuous, and the problem is a concave maximization problem, then its value function is continuous, concave, and often differentiable. Any numerical procedure needs to approximate the value function, but any such approximation will be imperfect since computers cannot model the entire space of continuous functions.

Y. Cai · K. L. Judd (✉)
Hoover Institution, 424 Galvez Mall, Stanford University, Stanford, CA 94305, USA
e-mail: kennethjudd@mac.com

Published online: 09 September 2012

Many DP problems are solved by value function iteration, where the period t value function is computed from the period $t+1$ value functions, and the value function is known at the terminal time T. Smooth approximation methods, such as polynomial or spline interpolation are natural in that they are smooth functions. However, numerical value function with polynomials or splines are often unstable. The key problem is that these methods aim to approximate functions in some L^2 norm. Therefore, they may violate shape properties that describe the true value function, and those shape violations lead to unstable fluctuations and significant errors in the value function iterations.

In this paper, we present a shape-preserving DP algorithm with value function iteration for solving discrete-time decision-making problems with continuous states. Essentially, we show that imposing basic shape restrictions whenever one approximates a new value function will stabilize value function iteration in many basic DP problems. 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. Sections 4 and 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

If state and control variables in a DP problem are continuous, then the value function must be approximated in some computationally tractable manner. It is common to approximate value functions with a finitely parameterized collection of functions; that is, we use some functional form $\hat{V}(x;\mathbf{c})$, where $\mathbf{c}$ is a vector of parameters, and approximate a value function, $V(x)$, with $\hat{V}(x;\mathbf{c})$ for some parameter value $\mathbf{c}$. For example, $\hat{V}$ could be a linear combination of polynomials where $\mathbf{c}$ would be the weights on polynomials. 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 (Bellman 1957).

Numerical solutions to a finite horizon DP are based on the Bellman equation:

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

where $V_t(x)$ is called the value function at time $t \le T$, the terminal value function $V_T(x)$ is given, x^+ denotes the state in the next period, where the transition depends on the current-stage state x and the action a. Furthermore, $\mathcal{D}(x,t)$ is a feasible set of a in state x, 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. [More detailed discussion of numerical DP can be found in Cai (2009), Cai and Judd (2010), Judd (1998) and Rust (2008).]

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* $\hat{V}(x; \mathbf{c}^T) \equiv V_T(x)$*. Then for* $t = T-1, T-2, \ldots, 0$*, iterate through steps 1 and 2.*

Step 1. *Maximization step. Compute*

$$v_i = \max_{a_i \in \mathcal{D}(x_i,t)} u_t(x_i, a_i) + \beta \hat{V}\left(x_i^+; \mathbf{c}^{t+1}\right)$$
$$\text{s.t.} \quad x_i^+ = g(x_i, a_i),$$

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.*

In the fitting step, 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 (Schumaker 1983), cubic splines, and B-splines. See Cai (2009) and Judd (1998) for more details.

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

$$\hat{V}(x; \mathbf{c}) = \sum_{j=0}^{n} c_j T_j\left((2x - a - b)/(b - a)\right). \tag{1}$$

We use the recursive form to evaluate Chebyshev polynomials: $T_0(z) = 1$, $T_1(z) = z$, and $T_{j+1}(z) = 2zT_j(z) - T_{j-1}(z)$, for $j = 1, 2, \ldots, n-1$, and any $z \in [-1, 1]$. 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 by the Chebyshev regression algorithm. See Cai (2009) and Judd (1998) for details on these points.

Algorithm 1 seems like a general solution of DP problems, but it has limitations. In the maximization step, a fast local optimizer will require that the objective function should be smooth and concave and the feasible constrained domain should be convex. However, while the true value function may be concave and the Lagrange data $\{(x_i, v_i) : i = 1, \ldots, m_t\}$ may be consistent with concavity, simple methods of fitting a curve to the data, like Chebyshev polynomial approximation, may produce a

nonconcave value function approximation, which in turn may lead to nonconcavity of the objective function in the maximization step and then instability in Algorithm 1.

We suspect that this explains the tendency of economists to use piecewise linear approximations of value functions since piecewise linear approximations automatically preserve shape. However, if one uses piecewise linear approximations when the true solution is a smooth function, then one needs to use many nodes to construct a good approximation, and the optimization problems in DP cannot use fast Newton-type solvers, because the piecewise linear approximation $\hat{V}(x; \mathbf{c})$ appearing in the objective function is only continuous but not differentiate at the nodes x_i and then incurs too many kinks for Newton-type optimizers.

3 Shape-preserving DP

Economics problems often have increasing and concave value functions in their utility or payoff maximization DP problems, and many operations research problems have decreasing and convex value functions in their cost minimization DP problems (see Bertsekas 2005, 2007). If the value function approximation method preserves the shape of the data, then the optimization step will be a smooth convex optimization problem for which it is easy to find the global optimum. Discretization methods and piecewise linear interpolation preserve shape but create nonsmooth problems in the optimization step. 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). 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. Cai and Judd (2012) used a shape-preserving rational function spline Hermite interpolation in numerical DP to solve a multi-stage portfolio problem with kinks.

These shape-preserving methods do not produce smooth functions, while most smooth approximation methods, such as Chebyshev interpolation, do not preserve shape. We introduce a general shape-preserving approximation method by adding shape constraints such that the shape properties still hold, and then we apply it 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, we create an optimization problem that modifies the Chebyshev coefficients so that concavity and monotonicity of the value function will be preserved. We begin with the Lagrange data $\{(x_i, v_i) : 1 \leq i \leq m\}$ generated by the maximization step of Algorithm 1. If theory tells us that the true value function is strictly increasing and concave, then add constraints to the fitting criterion that will impose shape restrictions.

Specifically, we approximate the value function using the functional form

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

where we replaced c_j in the Eq. (1) by $c_j^+ - c_j^-$ with $c_j^+, c_j^- \geq 0$. We choose some points $y_{i'}(i' = 1, \ldots, m')$, called shape nodes, and impose the requirement that $\hat{V}(x;\mathbf{c})$ satisfies the shape conditions at the shape nodes. We want to choose the parameters $\mathbf{c}$ to minimize approximation errors but also satisfy the shape conditions. We can get a perfect fit and satisfy shape conditions if we allow n to be sufficiently large, but the problem may have too many solutions. We can be sure to get a shape-preserving Chebyshev interpolant by adding enough shape-preserving constraints and using a sufficiently high degree (bigger than $(m-1)$) polynomial, but we again could have multiple solutions and end up with a more complex polynomial than necessary.

To allow for the flexibility necessary to have both interpolation and shape properties, we penalize the use of high-order polynomials. Therefore, we solve the following linear programming problem:

$$\min_{c_j, c_j^+, c_j^-} \sum_{j=0}^{m-1} \left(c_j^+ + c_j^-\right) + \sum_{j=m}^{n} (j+1-m)^2 \left(c_j^+ + c_j^-\right), \tag{2}$$

$$\text{s.t.} \quad \sum_{j=0}^{n} c_j T_j'(y_{i'}) > 0, \quad i' = 1, \ldots, m',$$

$$\sum_{j=0}^{n} c_j T_j''(y_{i'}) < 0, \quad i' = 1, \ldots, m',$$

$$\sum_{j=0}^{n} c_j T_j\left(\frac{2x_i - a - b}{b-a}\right) = v_i, \quad i = 1, \ldots, m,$$

$$c_j - \hat{c}_j = c_j^+ - c_j^-, \quad j = 0, \ldots, m-1,$$

$$c_j = c_j^+ - c_j^-, \quad j = m, \ldots, n,$$

$$c_j^+,\ c_j^- \geq 0, \quad j = 1, \ldots, n.$$

This problem includes interpolation among the constraints as well as the shape conditions, but chooses the polynomial with the smallest total weighted penalty, and is biased towards low-degree polynomials since a higher degree term is penalized more. The expression $c_j^+ - c_j^-$ represents c_j with $c_j^+, c_j^- \geq 0$, implying $|c_j| = c_j^+ + c_j^-$. The simple Chebyshev interpolation coefficients $\hat{c}_j$ give us a good initial guess. Therefore, we actually solve for the deviations of the Chebyshev coefficients from the simple Chebyshev interpolation coefficients.

The $y_{i'}$ are pre-specified shape nodes in $[-1, 1]$ for shape-preserving constraints. Usually we need $m' > m$ so that the shape preservation on the shape nodes implies

that shape is preserved everywhere. We may not know how many we need, so one must test the resulting solution on many more points, and increase the set of shape nodes if shape has not been preserved. Moreover, the interpolation constraints imply that the number of Chebyshev polynomials being used should not be smaller than the number of 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 model (2). We use the recursive formulas for evaluating the Chebyshev polynomials, $T_j(y)$, and their derivatives, $T'_j(y)$ and $T''_j(y)$ for $y \in [-1, 1]$.

4 Optimal growth example

We first illustrate our methods with a discrete-time optimal growth problem with one good and one capital stock [1]. 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.,

$$\begin{aligned} V_0(k_0) = \max_{c,l} \quad & \sum_{t=0}^{T-1} \beta^t u(c_t, l_t) + \beta^T V_T(k_T), \\ \text{s.t.} \quad & k_{t+1} = F(k_t, l_t) - c_t, \quad 0 \leq t < T, \\ & \underline{k} \leq k_t \leq \bar{k}, \quad 1 \leq t \leq T, \\ & c_t,\ l_t \geq \epsilon, \quad 0 \leq t < T, \end{aligned} \tag{3}$$

where k_t is the capital stock at time t with k_0 given, c_t is the consumption of the good, 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, $V_T(x)$ is a given terminal value function, and $u(c_t, l_t)$ is the utility function, and ϵ is a small positive number to avoid the nonpositive consumption or labor supply.

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 $\alpha = 0.25$, $\beta = 0.99$, $\gamma = 8$, $\eta = 1$, $A = (1 - \beta)/(\alpha\beta)$ and $T = 20$. Let the range of k be $[0.1, 1.9]$, i.e., $\underline{k} = 0.1$ and $\bar{k} = 1.9$. And we choose $\epsilon = 10^{-6}$ in the model (3). The production function is $f(k, l) = Ak^\alpha l^{1-\alpha}$, and the utility function is a power utility with the following form

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

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 $l_{ss} = 1$. Moreover, the utility at the steady state is 0 and then the true value function at the steady state is also 0. This normalization of the typical

[1] Please see Judd (1998) for a detailed description of this.

power utility from the economic literature not only helps avoid scaling issues but only gives us a simple criterion to check if a numerical solution is close to the true solution.

We choose the terminal value function as

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

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 Solve test problem exactly

For the finite horizon optimal growth problem (3), 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 the DP model (4) given by numerical DP algorithms because of the numerical approximation errors. But when T is large, the solution of (4) given by numerical DP algorithms is usually better than the solution of (3) 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 $T > 20$. 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 (3) given by SNOPT (Gill et al. 2005) in GAMS code (McCarl et al. 2011) as the true solutions.

4.2 DP solution

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

$$\begin{aligned} V_t(k) = \max_{c,l} \quad & u(c,l) + \beta V_{t+1}(k^+), \\ \text{s.t.} \quad & k^+ = F(k,l) - c, \\ & \underline{k} \leq k^+ \leq \bar{k}, \quad c, l \geq \epsilon, \end{aligned} \tag{4}$$

for $t < T$, where $V_T(x)$ is the previously given terminal value function. Here k is the state variable and (c, l) are the control variables.

The program code is written in GAMS, and we choose SNOPT as the nonlinear programming solver in the optimization step, and CPLEX (in the GAMS environment) as the linear programming solver for the model (2) in the fitting step. We choose

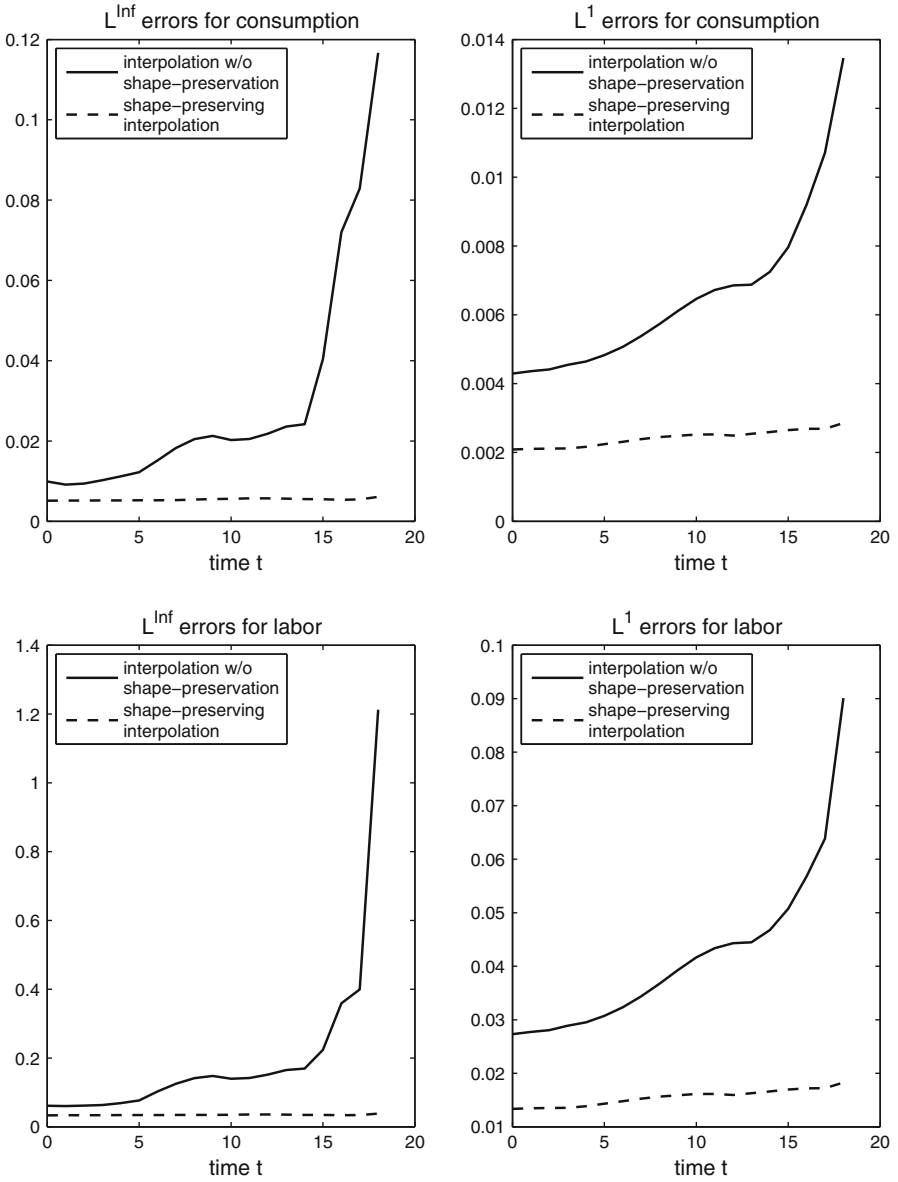

Fig. 1 Errors of numerical DP with Chebyshev interpolation with/without shape-preservation for growth problems

$m = 10$ Chebyshev interpolation nodes x_i on $[0.1, 1.9]$, and let the shape-preserving nodes $y_{i'}$ be the equally-spaced nodes in $[-1, 1]$ in the model (2).

Figure 1 shows the relative errors of optimal controls at each time in L^∞ and L^1 respectively. We use the solutions given by directly applying SNOPT in the model (3) as the true solutions. The solid lines are errors of solutions of the model (4) computed

by numerical DP algorithm with standard degree-9 Chebyshev polynomial interpolation using $m = 10$ Chebyshev interpolation nodes. The dashed lines are errors of solutions of the model (4) computed by numerical DP algorithm with shape-preserving Chebyshev polynomial interpolation with $m = 10$ Chebyshev interpolation nodes and $m' = 20$ equally-spaced nodes for shape constraints in the models (2).

From Fig. 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 Multi-stage portfolio optimization example

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 time 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 risk-free return R_f for each period. If $S_t = (S_{t,1}, \ldots, S_{t,n})^\top$ 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

$$W_{t+1} = R_f(W_t - e^\top S_t) + R^\top S_t, \tag{5}$$

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

A simple multi-stage portfolio optimization problem is to find an optimal portfolio S_t at each time t such that we have a maximal expected terminal utility, i.e.,

$$V_0(W_0) = \max_{S_t, 0 \le t < T} \mathbb{E}\{u(W_T)\},$$

where W_T is the terminal wealth derived from the recursive formula (5) with a given W_0, and u is the terminal utility function, and $\mathbb{E}\{\cdot\}$ 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 risk-free return $R_f = 1.04$, and the stock has a discrete random return

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

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 $\gamma = 2$ and $K = 0.2$ so that the terminal wealth should be always bigger than 0.2. Moreover, we assume that borrowing or shorting is not allowed in this example, i.e., $B_t \ge 0$ and $S_t \ge 0$ for all t.

5.1 Tree method

In the portfolio optimization problem, if we discretize the random returns of n stocks as $R = R^{(j)} = (R_{1,j}, \ldots, R_{n,j})$ with probability q_j for $1 \le j \le m$, then it becomes a tree. 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 time 1: $(W_{1,1}, P_{1,1})$ and $(W_{1,2}, P_{1,2})$, and four scenarios at time 2: $(W_{2,1}, P_{2,1}), \ldots, (W_{2,4}, P_{2,4})$.

In a mathematical formula, the probability of scenario k at time $t+1$ is

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

and the wealth at scenario k and time $t+1$ is

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

for $1 \le k \le m^{t+1}$ and $0 \le t < T$. Here, $W_{0,1} = W_0$ is a given initial wealth, $P_{0,1} = 1$, $\mathrm{mod}(k, m)$ is the remainder of division of k by m, and $[(k-1)/m]$ is the quotient of division of $(k-1)$ by m. The goal is to choose optimal bond allocations $B_{t,k}$ and stock allocations $S_{t,k}$ to maximize the expected terminal utility, i.e.,

$$\max \sum_{k=1}^{m^T} (P_{T,k} \cdot u(W_{T,k})). \tag{6}$$

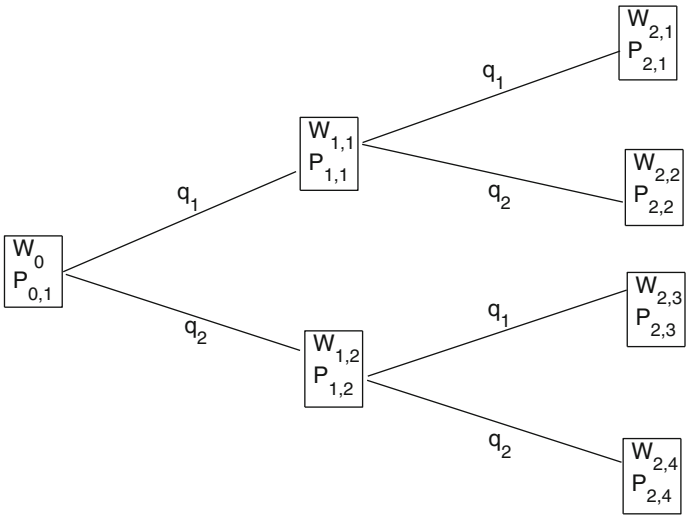

Fig. 2 A binary tree with two periods

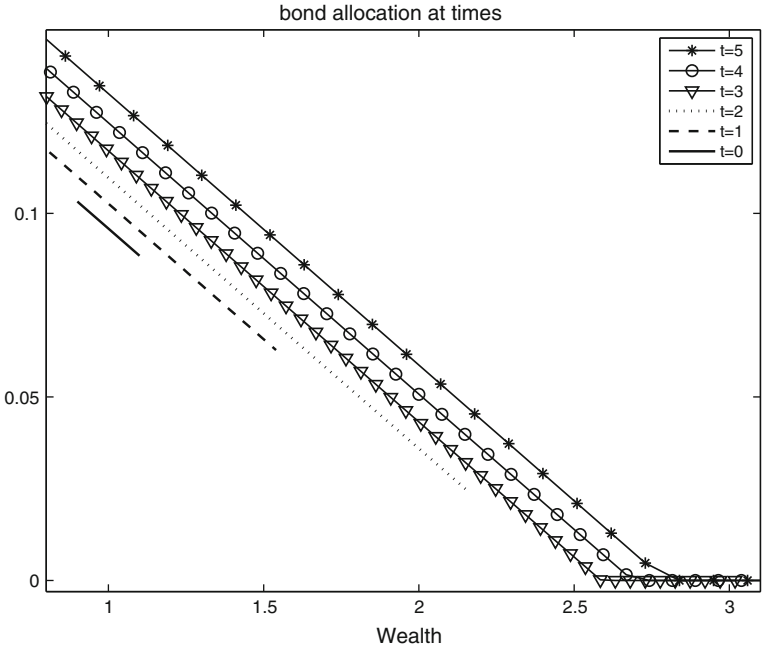

Fig. 3 Exact optimal bond allocations

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.

This tree method resembles the stochastic programming method (Birge and Louveaux 1997). But this tree method includes all possible scenarios with their assigned probabilities. 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 our DP algorithms have no such disadvantage. 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 1982) in AMPL code (Fourer et al. 1990) via the NEOS server (Czyzyk et al. 1998).

Figure 3 shows the optimal bond allocation B_t for $t = 0, 1, \ldots, 5$, computed by the tree method for the numerical example. Next, we will use the 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

$$V_t(W) = \max_{B,S} \; E\{V_{t+1}(R_f B + R^\top S)\}, \tag{7}$$
$$\text{s.t.} \quad B + e^\top S = W,$$

for $t = 0, 1, \ldots, T-1$, 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 nor borrowing bond is allowed.

Since the terminal utility function is $u(W_T) = (W_T - K)^{1-\gamma}/(1-\gamma)$, we know that the terminal wealth W_T must be always larger than K. It follows that we should have $W_t > K R_f^{t-T}$. Thus, since shorting or borrowing is not allowed and R is bounded, we choose the ranges $[\underline{W}_t, \overline{W}_t]$ for approximating value functions as

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

with a given initial wealth bound $[\underline{W}_0,\ \overline{W}_0]$, where $\varepsilon > 0$ is a small number.

Specifically, for the numerical example with $K = 0.2$, $R_f = 1.04$, $\min(R) = 0.9$ and $\max(R) = 1.4$, after we choose $\underline{W}_0 = 0.9$ and $\overline{W}_0 = 1.1$, we have

$$\begin{aligned} \left[\underline{W}_1, \ldots, \underline{W}_6\right] &= [0.81, 0.729, 0.656, 0.590, 0.531, 0.478], \\ \left[\overline{W}_1, \ldots, \overline{W}_6\right] &= [1.54, 2.156, 3.018, 4.226, 5.916, 8.282]. \end{aligned}$$

We see that the ranges are 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 times $t = 0, 1, \ldots, 5$ in Algorithm 1.

We choose $m = 10$ Chebyshev interpolation nodes in $[\underline{W}_t, \overline{W}_t]$ for earlier times $t = 0, 1, 2$, and then choose $m = 20$ Chebyshev interpolation nodes for the latter times $t = 3, 4, 5$ (having wider ranges than the earlier times). The computational results of numerical DP algorithms are given by our GAMS code. We choose SNOPT 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.

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 errors of optimal bond allocation fractions, i.e.,

$$\log_{10}\left(10^{-6} + \frac{\left|B^*_{t,\mathrm{DP}} - B^*_t\right|}{W_t}\right),$$

where B^*_t are true optimal bond allocations for the wealth W_t at time t which are given by the tree method (6), and $B^*_{t,\mathrm{DP}}$ are computed optimal bond allocation from the DP model (7) by numerical DP algorithms with Chebyshev interpolation with/without shape-preservation. The squares are errors of solutions of (7) computed by numerical DP algorithm with standard degree-9 (for earlier times $t = 0, 1, 2$) or degree-19

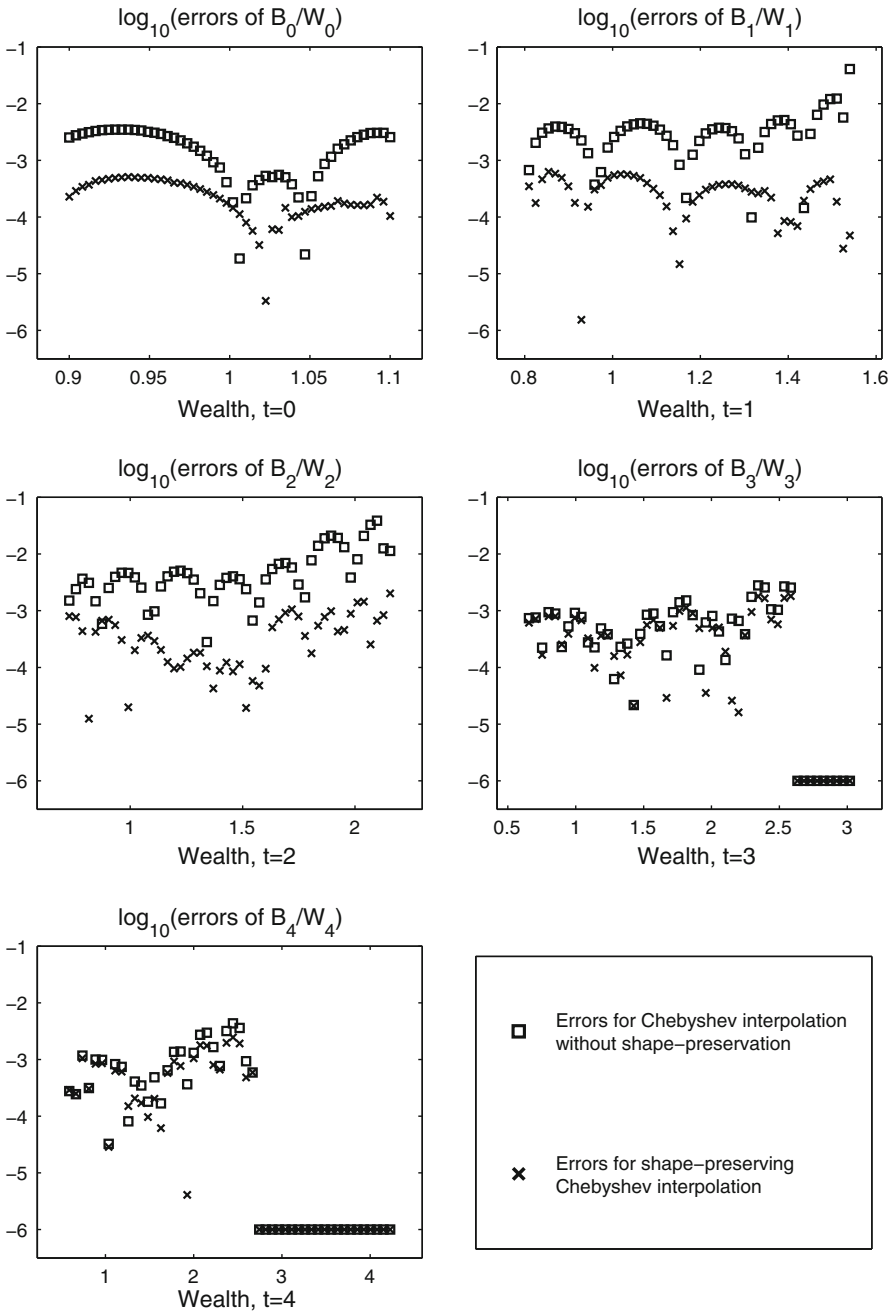

Fig. 4 Errors of optimal bond allocations from numerical DP with Chebyshev interpolation with/without shape-preservation

(for latter times $t = 3, 4, 5$) Chebyshev polynomial interpolation using $m = 10$ (for earlier times $t = 0, 1, 2$) or $m = 20$ (for latter times $t = 3, 4, 5$) Chebyshev interpolation nodes. The x-marks are errors of solutions of (7) computed by numerical DP algorithm with shape-preserving Chebyshev polynomial interpolation with $m = 10$ (for earlier times $t = 0, 1, 2$) or $m = 20$ (for latter times $t = 3, 4, 5$) Chebyshev interpolation nodes and $m' = 20$ equally-spaced nodes for shape constraints in the model (2).

From Fig. 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 times $t = 0, 1, 2, 3, 4$. This means that shape-preservation significantly improves the performance of numerical DP algorithms in stability and accuracy.

6 Conclusion

Value function iteration is the basic tool of dynamic optimization but simple methods suffer from instability or slow speed. A key reason for instability is that value function approximation methods may not preserve the shape of the true value function, and a key reason for the slow speed of other methods is the use of nonsmooth value function approximation methods. This paper presents a general shape-preserving DP algorithm with smooth approximations of the value function. It is built around a linear programming approach to the fitting step in dynamic programming, making it easy and reliable to implement.

This shape-preserving approach to DP was illustrated with two simple example, but is clearly applicable to a wide class of problems in economics, finance, and operations research where we often have shape information about the true value function. Further research will implement the same ideas to continuous-time problems, and ones with higher dimension.

Acknowledgments We gratefully acknowledge the financial support from National Science Foundation (Grant Number SES-0951576), and thank two anonymous referees for their helpful comments.

References

Bellman R (1957) Dynamic programming. Princeton University Press, Princeton
Bertsekas D (2005) Dynamic programming and optimal control, vol I. Athena Scientific, Belmont
Bertsekas D (2007) Dynamic programming and optimal control, vol II. Athena Scientific, Belmont
Birge J, Louveaux FV (1997) Introduction to stochastic programming. Springer, New York
Cai Y (2009) Dynamic programming and its application in economics and finance. PhD thesis, Stanford University
Cai Y, Judd KL (2010) Stable and efficient computational methods for dynamic programming. J Eur Econ Assoc 8(2–3):626–634
Cai Y, Judd KL (2012) Dynamic programming with shape-preserving rational spline Hermite interpolation. Econ Lett 117(1):161–164
Czyzyk J, Mesnier M, Moré J (1998) The NEOS Server. IEEE J Comput Sci Eng 5:68–75
Fourer R, Gay DM, Kernighan BW (1990) Modeling language for mathematical programming. Manag Sci 36:519–554
Gill P, Murray W, Saunders M (2005) SNOPT: an SQP algorithm for largescale constrained optimization. SIAM Rev 47(1):99–131

Judd KL (1998) Numerical methods in economics. MIT Press, Cambridge
Judd KL, Solnick A (1994) Numerical dynamic programming with shape-preserving splines. Hoover Institution, Stanford
McCarl B et al. (2011) McCarl Expanded GAMS user guide version 23.6. http://www.gams.com/mccarl/mccarlhtml/. Accessed 6 Sept 2012
Murtagh B, Saunders M (1982) A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. Math Program Study 16:84–117
Rust J (2008) Dynamic programming. In: Durlauf SN, Blume LE (eds) New Palgrave dictionary of economics. 2. Palgrave Macmillan, Basingstoke
Schumaker L (1983) On shape-preserving quadratic spline interpolation. SIAM J Numer Anal 20:854–864
Wang S-P, Judd KL (2000) Solving a savings allocation problem by numerical dynamic programming with shape-preserving interpolation. Comput Oper Res 27(5):399–408