Economics Letters 117 (2012) 161–164

Contents lists available at SciVerse ScienceDirect

Economics Letters

journal homepage: www.elsevier.com/locate/ecolet

Dynamic programming with shape-preserving rational spline Hermite interpolation

Yongyang Cai *, Kenneth L. Judd

Hoover Institution, 424 Galvez Mall, Stanford University, Stanford CA 94305, United States

ARTICLE INFO

Article history:
Received 23 December 2011
Received in revised form
20 April 2012
Accepted 4 May 2012
Available online 14 May 2012

JEL classification:
C6
C61
C63

Keywords:
Numerical dynamic programming
Shape-preserving approximation
Hermite interpolation
Rational function spline
Value function iteration

ABSTRACT

Numerical methods for dynamic programming often use value function iteration and interpolation. We present a novel shape-preserving rational spline approximation method that improves value function iteration in terms of both stability and accuracy compared to more common methods.

1. Introduction

Conventional dynamic programming (DP) algorithms compute a value function at a finite number of states, and then fit some function to those points to create a new approximate value function. Simple versions of this approach often have two problems. First, Cai and Judd (forthcoming) shows that value function iteration can be unstable if the approximation method does not preserve the shape of the data. Second, even if the iterations are stable, the quality of the approximations may be low. For example, a good piecewise linear approximation of a smooth value function will require many nodes. Cai and Judd (2012) shows that smooth value function approximations using both level and slope information will significantly improve the accuracy of value function iteration.

This paper introduces a novel piecewise rational function approximation that uses both level and slope data to produce C^1 shape-preserving approximations. We use a multi-stage portfolio optimization problem to show its advantages.

2. Numerical methods for dynamic programming

If state and control variables in a DP problem are continuous, value function must be approximated in some tractable manner. Value functions are typically approximated with a finitely parameterized collection of function; 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 vector $\mathbf{c}$. For example, $\hat{V}$ could be a linear combination of polynomials where $\mathbf{c}$ would be the weights. 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:

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

where $V_t(x)$ is the value function at time $t \leq T$, and the terminal value function $V_T(x)$ is given. x^+ denotes the possibly random state in the next period, where the transition depends on the current-stage state x, the action a and a serially uncorrelated random variable z_t. Furthermore, $\mathbb{E}\{\cdot \mid x, a\}$ is the conditional expectation operator, $\mathcal{D}(x, t)$ is the set of feasible values for a in state x, and

* Corresponding author. Tel.: +1 650 926 9786.
E-mail address: yycai@stanford.edu (Y. Cai).

0165-1765/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.econlet.2012.05.004

$u_t(x, a)$ is the utility function at time t. The following outlines the parametric DP method. (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, $X_t = \{x_{it}: 1 \le i \le m_t\}$ 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

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

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

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.

Note that we constrain x_i^+ to be in $[x_{t+1}^{\min}, x_{t+1}^{\max}]$. If this constraint is not imposed, then we might be extrapolating $\hat{V}$ outside of the range being approximated, which can easily lead to numerical instabilities. We choose these limits so that they seldom, if ever, bind.

3. Dynamic programming with Hermite interpolation

The conventional DP algorithm uses the data set $\{(x_i, v_i): i = 1, \ldots, m\}$, called the Lagrange data, to construct the value function approximation $\hat{V}_t(x)$. Cai and Judd (2012) introduces DP algorithms with Hermite approximation using both level and slope information, called Hermite data, in the fitting step. Computing Hermite information is almost as cheap as computing Lagrange data, but the extra information produces a more accurate approximation. Here we describe their algorithm for readers' convenience.

Algorithm 2. Numerical dynamic programming with value function iteration and Hermite approximation for finite horizon problems.

Initialization. Choose the approximation nodes, $X_t = \{x_{it}: 1 \le i \le m_t\}$ 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. For each $x_i \in X_t$, $1 \le i \le m_t$, compute

$$\begin{aligned} v_i = & \max_{a_i \in \mathcal{D}(y_i,t), y_i} u_t(y_i, a_i) + \beta \mathbb{E}\{\hat{V}(x_i^+; \mathbf{c}^{t+1})\}, \\ & \text{s.t. } x_i^+ = g(y_i, a_i, z_t), \\ & x_i - y_i = 0, \\ & x_{t+1}^{\min} \le x_i^+ \le x_{t+1}^{\max}, \end{aligned}$$

and

$$s_i = \lambda_i^*,$$

where λ_i^* is the shadow price vector of the constraint $x_i - y_i = 0$.

Step 2. Hermite 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, s_i) data.

4. Shape-preserving rational function spline Hermite interpolation

DP problems in economics problems often have increasing and concave value functions, and many cost-minimization problems in operations research have increasing and convex cost functions (see Bertsekas, 2005, 2007). Cai and Judd (forthcoming) shows that a shape-preserving polynomial approximation of the value function will stabilize value function iteration. However, their methods impose many additional shape-preserving constraints in the fitting problem and are computationally more demanding than desirable. There has been much effort developing shape-preserving and Hermite interpolation; see, for example, the survey paper in Goodman (2001). Most methods produce splines and are global, with all spline parameters depending on all the data. Judd and Solnick (1994) applied Schumaker shape-preserving polynomial splines (Schumaker, 1983) in optimal growth problems, but Schumaker splines are costly because they require creating new nodes each time a value function is constructed.

This paper presents an inexpensive shape-preserving rational function spline Hermite interpolation for a concave, monotonically increasing function. Given the Hermite data $\{(x_i, v_i, s_i) : i = 1, \ldots, m\}$, we approximate the value function on the interval $[x_i, x_{i+1}]$ with

$$\hat{V}(x; \mathbf{c}) = c_{i1} + c_{i2}(x - x_i) + \frac{c_{i3}c_{i4}(x - x_i)(x - x_{i+1})}{c_{i3}(x - x_i) + c_{i4}(x - x_{i+1})},$$

for $x \in [x_i, x_{i+1}]$, where

$$\begin{aligned} c_{i1} &= v_i, \\ c_{i2} &= \frac{v_{i+1} - v_i}{x_{i+1} - x_i}, \\ c_{i3} &= s_i - c_{i2}, \\ c_{i4} &= s_{i+1} - c_{i2}, \end{aligned}$$

for $i = 1, \ldots, m-1$. $\hat{V}(x; \mathbf{c})$ is obviously $\mathcal{C}^\infty$ on each interval (x_i, x_{i+1}), and $\mathcal{C}^1$ globally.

This is a local method because the rational function interpolant on each interval $[x_i, x_{i+1}]$ depends only on the level and slope information at the endpoints. Moreover, $\hat{V}(x; \mathbf{c})$ is shape-preserving. If the data is consistent with a concave increasing value function, i.e., $s_i > c_{i2} > s_{i+1} > 0$, then straightforward computations show that $\hat{V}'(x; \mathbf{c}) > 0$ and $\hat{V}''(x; \mathbf{c}) < 0$ for all $x \in (x_i, x_{i+1})$, that is, it is increasing and concave in the interval (x_i, x_{i+1}). It is also cheaply computed since the approximation on each interval depends solely on the data at its endpoints. This approach does not require adding new nodes to the spline, nor the determination of free parameters, features that are common in the shape-preserving polynomial spline literature.

5. Multi-stage portfolio optimization problems

We next present a numerical example of DP with the shape-preserving rational function spline Hermite interpolation to solve multi-stage portfolio optimization problems, and compare it with earlier methods. The numerical example assumes that there are one stock and one bond available for investment, and the number of investment periods is $T = 6$. For each period, 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 W_t be the total wealth, and let S_t be the amount of money invested in the stock at time t, then the amount invested in the

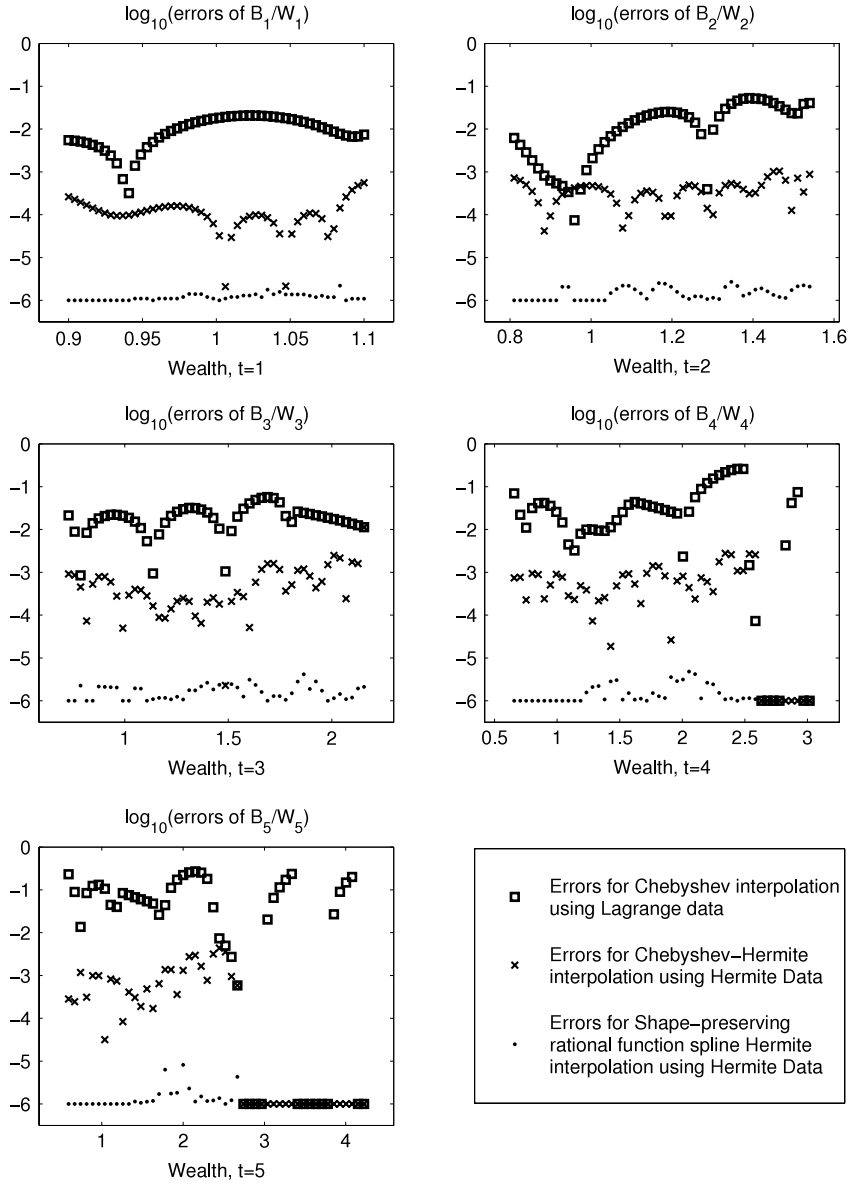

Fig. 1. Errors of optimal bond allocations from numerical DP.

bond is $B_t = W_t - S_t$, and the wealth at the next stage is

$$W_{t+1} = R_f(W_t - S_t) + RS_t,$$

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

We want to find an optimal portfolio S_t at each time t such that the expected terminal utility is maximized, i.e.,

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

where $u(W) = -(W-K)^{-1}$ with $K = 0.2$. 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.

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

$$V_t(W) = \max_{B,S \ge 0} \mathbb{E}\{V_{t+1}(R_f B + RS)\},$$
$$\text{s.t. } W - B - S = 0,$$

for $t = 0, 1, \ldots, T-1$, where W is the state variable, and B and S are the control variables, and the terminal value function is $V_T(W) = u(W)$.

The envelope theorem implies $V_t'(W) = \lambda^*(W)$, where $\lambda^*(W)$ is the shadow price for the constraint $W - B - S = 0$. From $u(W_T) = -(W_T - K)^{-1}$, we know that W_T must be always larger than K. It follows that we should have $W_t > KR_f^{t-T}$. 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

$$\underline{W}_{t+1} = \max\{\min(R)\underline{W}_t, KR_f^{t-T} + \varepsilon\},$$
$$\overline{W}_{t+1} = \max(R)\overline{W}_t,$$

with a given initial wealth bound $[\underline{W}_0, \overline{W}_0] = [0.9, 1.1]$, where $\varepsilon > 0$ is a small number.

To evaluate the accuracy of our method, we compare it to the true solution. The value function has no closed-form expression because of the borrowing constraints. An example with a closed-form solution would have been too easy for our method to solve. The borrowing constraint makes this more challenging because the bond strategy has a kink at the largest wealth where it binds. However, we can compute the true solution for any initial wealth using the tree method described in Cai and Judd (forthcoming). The tree method solves for the state-contingent values of all variables

Table 1
Errors of optimal bond allocations for various γ.

γ	Number of approximation nodes	Errors at time $t = 0$
0.5	10	0
2	10	1.1×10^{-6}
4	20	7.3×10^{-4}
	40	1.1×10^{-4}
6	20	1.7×10^{-3}
	40	3.4×10^{-4}
8	20	3.9×10^{-3}
	40	5.3×10^{-4}

at all nodes in the decision tree. We use the true solution to measure the accuracy of our DP algorithm and compare it with the accuracy of other methods. The presence of a borrowing constraint also means we should approximate the value function, which will be C^2, not the policy function which may only be C^0. Polynomial approximation theory tells us to focus on approximating the smoother function.

Fig. 1 shows relative errors for bond allocations of alternative DP algorithms:

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

where B^*_t are true optimal bond allocations from the tree method, and $B^*_{t,\mathrm{DP}}$ are computed optimal bond allocation from numerical DP algorithms, for $W_t \in [\underline{W}_t, \overline{W}_t]$. The squares are errors of solutions of Algorithm 1 with Chebyshev interpolation using Lagrange data, the x-marks are errors of Algorithm 2 with Chebyshev–Hermite interpolation using Hermite data, and the solid points are errors of Algorithm 2 with the rational function spline interpolation using Hermite data. All the computational results are given by MINOS (Murtagh and Saunders, 1982) in AMPL (Fourer et al., 1990) via the NEOS server (Czyzyk et al., 1998). For Algorithm 1 with Chebyshev interpolation or Algorithm 2 with Chebyshev–Hermite interpolation (Cai and Judd, 2012), we use $m = 10$ Chebyshev nodes and degree-9 or degree-19 Chebyshev polynomials respectively. For Algorithm 2 with the rational function spline interpolation, we use $m = 10$ equally-spaced nodes.

We see that the errors are about $O(10^{-1})$ or $O(10^{-2})$ for Chebyshev interpolation using Lagrange data, while they are about $O(10^{-3})$ or $O(10^{-4})$ for Chebyshev–Hermite interpolation using Hermite data. However, the errors of the rational function spline Hermite interpolation is always about $O(10^{-6})$, showing that it has the best performance for approximating value functions.

Table 1 lists numerical errors of optimal bond allocations from Algorithm 2 with the rational function spline interpolation, for various values of γ. We see that even for large γ, the solutions from Algorithm 2 with the rational function spline interpolation are still good.

Our new approximation method was always as fast as any of the other algorithms. Therefore, the shape-preserving rational function spline Hermite interpolation is reliable and often substantially better than other approximation methods.

6. Conclusion

This paper presents a numerical DP algorithm with a shape-preserving rational function spline Hermite interpolation. The portfolio examples indicate that this approach is reliable and accurate when solving nontrivial concave DP problems in economics. The method described here relied only on the shape properties of the problem, indicating that it can be used in a wide variety of problems in economics, finance, and operations research.

Acknowledgment

We gratefully acknowledge NSF support (SES-0951576), and we thank the anonymous referees.

References

Bellman, R., 1957. Dynamic Programming. Princeton University Press.
Bertsekas, D., 2005. Dynamic Programming and Optimal Control, vol. I. Athena Scientific.
Bertsekas, D., 2007. Dynamic Programming and Optimal Control, Vol. II. Athena Scientific.
Cai, Y., 2009. Dynamic Programming and Its Application in Economics and Finance. Ph.D. Thesis, Stanford University.
Cai, Y., Judd, K.L., 2012. Shape-preserving dynamic programming, in: Mathematical Methods of Operations Research (forthcoming).
Cai, Y., Judd, K.L., 2012. Dynamic programming with Hermite interpolation. Hoover Institution (working paper).
Cai, Y., Judd, K.L., 2010. Stable and efficient computational methods for dynamic programming. Journal of the European Economic Association 8 (2–3), 626–634.
Czyzyk, J., Mesnier, M., Moré, J., 1998. The NEOS server. IEEE Journal on Computational Science and Engineering 5, 68–75.
Fourer, R., Gay, D.M., Kernighan, B.W., 1990. Modeling language for mathematical programming. Management Science 36, 519–554.
Goodman, T.N.T., 2001. Shape preserving interpolation by curves. in: Proceedings of the 2001 International Symposium, pp. 24–35.
Judd, K., 1998. Numerical Methods in Economics. The MIT Press.
Judd, K., Solnick, A., 1994. Numerical Dynamic Programming with Shape-Preserving Splines. Hoover Institution.
Murtagh, B., Saunders, M., 1982. A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. Mathematical Programming Study 16, 84–117.
Rust, J., 2008. Dynamic programming. In: Durlauf, S.N., Blume, L.E. (Eds.), New Palgrave Dictionary of Economics, second ed. Palgrave Macmillan.
Schumaker, L., 1983. On shape-preserving quadratic spline interpolation. SIAM Journal of Numerical Analysis 20, 854–864.