STABLE AND EFFICIENT COMPUTATIONAL METHODS FOR DYNAMIC PROGRAMMING

Yongyang Cai
Stanford University

Kenneth L. Judd
Hoover Institution

Abstract
Dynamic programming is the foundation of dynamic economic analysis and often requires numerical solution methods. Standard methods are either slow or unstable. These instabilities are avoided when one uses modern methods from numerical optimization and approximation. Furthermore, large dynamic programming problems can be solved by using modern parallel computing architectures. (JEL: K23, L26, L51)

1. Introduction

Dynamic programming (DP) is the foundation of dynamic economic analysis. Numerical solutions are often the only way to solve DP problems but most methods are either slow, particularly for multidimensional problems, or are unreliable and numerically unstable. In this paper, we discuss some new ideas that stabilize numerical methods for dynamic programming, use information more efficiently, and exploit modern computing environments to make possible the solution of large DP problems. We illustrate these ideas with a basic dynamic portfolio problem. This paper gives a short overview of our recent work on DP. For a more extensive description of this work, see Cai (2009).

DP is based on the observation by Bellman (1957) that "[a]n 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." This reduces multi-period sequential decision problems to a sequence of one-period optimal control problems. We examine only finite horizon problems in this paper but the numerical ideas also apply to infinite-horizon problems. The recursive principle of DP implies that the solution can be characterized by value functions, $V_t(x)$, defined by the Bellman equation

The editor in charge of this paper was George Marios Angeletos.

Acknowledgments: We acknowledge the comments of the referee and editor of JEEA, and the participants of the August 2009 conference of the EEA.

E-mail addresses: Cai: yycai@stanford.edu; Judd: kennethjudd@mac.com

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

for $t = 0, 1, \dots, T-1$, where T is finite, x is the state and a is the action choice (x and a can be vectors), $u_t(x,a)$ is the utility function at time $t < T$, and $u_T(x) = V_T(x)$ is the terminal utility function, β is the discount factor, $D_t(x)$ is a feasible set of a_t at time t, x^+ is the (possibly random) state at time $t+1$ if the state is x at time t and action a is chosen at time t, and $E\{\cdot\}$ is the expectation operator.

This paper describes the critical numerical challenges that must be met by methods that aim to solve for the value functions. We give a quick overview of those methods, describe our new methods, and illustrate them for a simple portfolio example.

2. A Portfolio Example

We illustrate our methods with a dynamic portfolio problem with transaction costs. We assume that an investor begins with some wealth, invests it in several assets, and manages it so as to maximize the expected utility of wealth at time T. We assume a power utility function for terminal wealth,

$$V_T(W) = W^{1-\gamma}/(1-\gamma),$$

where $\gamma > 0$ and $\gamma \neq 1$. Let $R = (R_1, \dots, R_n)$ be the random one-period return of n risky assets, and R_f be the return of the riskless asset. The portfolio share for asset i at the beginning of a period is denoted x_i. The state variables are the wealth W and allocations $x = (x_1, \dots, x_n)^\top$ invested in the risky assets at the beginning of a period. The difference between wealth and the wealth invested in stocks is invested in bonds. We assume a proportional transaction cost τ for all sales and purchases of the risky assets. Let $\delta_i^+ W$ denote the amount of asset i purchased, expressed as a fraction of wealth, and let $\delta_i^- W$ denote the amount sold, where $\delta^+, \delta^- \geq 0$. Let e denote the column vector where $e_i = 1$ for all i.

The DP problem becomes

$$V_t(W,x) = \max_{\delta^+, \delta^- \geq 0} E\{V_{t+1}(W^+, x^+)\},$$

where $X_i^+ \equiv R_i(x_i + \delta_i^+ - \delta_i^-)W$ is time $t+1$ wealth in asset i, $m \equiv e^\top(\delta^+ - \delta^- + \tau(\delta^+ + \delta^-))$ where mW is the change in bond holding, $W^+ \equiv e^\top X^+ + R_f(1 - e^\top x - m)W$ is time $t+1$ wealth, and $x^+ \equiv X^+/W^+$ is the vector of risky asset allocations. Given the isoelasticity of V_T, we know that $V_t(W,x) = W^{1-\gamma} g_t(x)$, for some functions $g_t(x)$, $t = 0, 1, \dots, T-1$. The numerical task reduces to approximating the $g_t(x)$ functions.

3. Numerical Methods for DP

Multi-stage decision-making problems are numerically challenging. Judd (1998) and Rust (2008) review numerical methods used in economics for solving DP problems. In this section, we present the basic approach for parametric DP with value function iteration. These algorithms adopt the backward recursion approach, but are generally applied only to very low dimensional problems because of the "curse of dimensionality."

If state and control variables are continuous, and if the value function is continuous, it is natural to use a continuous approximation for the value functions. Because computers cannot model the entire space of continuous functions, we must instead use a finitely parameterizable collection of functions to approximate value functions, $V(x) \approx \hat{V}(x; c)$, where c is a vector of parameters. There are many options for the functional form $\hat{V}$ but we focus on using linear combinations of polynomials. Algorithm 1 describes the standard approach.

Algorithm 1. (Value Function Iteration (VFI) for Finite Horizon Problems)

Initialization. Choose the approximation nodes, $X = \{x_i : 1 \leq i \leq m\}$, and a functional form for $\hat{V}(x; c)$. Let $\hat{V}(x; c_T) \equiv u_T(x)$. Then for $t = T-1, T-2, \ldots, 0$, iterate through steps 1 and 2.

Step 1. Maximization step. Compute

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

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

Step 2. Fitting step. Using an appropriate approximation method, compute the c_t such that $\hat{V}(x; c_t)$ approximates the (x_i, v_i) data; that is, choose c_t so that $\hat{V}(x_i; c_t) \approx v_i$.

Algorithm 1 includes three types of numerical problems. First, we need to solve a maximization problem at each node x_i. Second, the evaluation of the objective requires us to compute an expectation. Third, we need to efficiently take the data and compute the best fit for the new value function. The challenge is not only to use good numerical methods for each of these steps but also to choose methods that are compatible with each other and jointly lead to efficient algorithms. The next section describes these choices in more detail.

4. Tools from Numerical Analysis

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

4.1. Optimization

For each VFI, the most time-consuming part is the optimization step. There are m optimization tasks, one for each node. If the number of VFIs is T, then the total number of optimization tasks is $T \times m$. All these optimization tasks are usually small problems with a low number of control variables.

If the value function approximation is not smooth, then the objective function of the optimization problem in the maximization step is not smooth, forcing us to use methods that can solve non-smooth problems. If the value function approximation is smooth, we can use Newton's method and related methods for constrained nonlinear optimization problems, which have a locally quadratic convergence rate.

We used NPSOL (see Gill et al. 1994), a set of Fortran subroutines for minimizing a smooth function subject to linear and nonlinear constraints. The NPSOL libraries may be called from a driver program in Fortran, C/C++, or MATLAB. NPSOL is an appropriate optimization solver for DP applications in economics and finance, since the optimization tasks in numerical DP are small-size smooth problems.

4.2. Numerical Integration

In the objective function of the Bellman equation, we often need to compute the conditional expectation of $V(x^+ \mid x, a)$. One naive way is to apply Monte Carlo methods to compute the expectation. However, one must use large Monte Carlo samples to get accurate solutions to the optimization problems. When the integrand is smooth, it is much more efficient to use Gaussian quadrature.

Our portfolio example assumes log normal returns. If Y is log normal, then $Y = e^X$ where $X \sim \mathcal{N}(\mu, \sigma^2)$, and we use the formula

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

where ω_i and x_i are the weights and nodes for the degree-$(2n-1)$ Gauss-Hermite quadrature rule. If X is multivariate normal, $\mathcal{N}(\mu, \Sigma)$, we can use a product Gauss-Hermite rule after using a Cholesky decomposition of Σ.

4.3. Traditional Approximation Methods

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 $\varphi_j(x)$ and defines $\hat{V}(x; c) = \sum_{j=0}^{n} c_j \varphi_j(x)$ to be the degree-n

approximation. We use tensor and complete Chebyshev approximation, which are spectral methods. In contrast, a finite element method uses locally basis functions $\varphi_j(x)$ that are nonzero over sub-domains of the approximation domain. Examples of finite element methods include piecewise linear interpolation, Schumaker shape-preserving interpolation, cubic splines, and B-splines. Traditional methods take the (x_i, v_i) data generated by the optimization step and chooses c_t by either interpolation or least squares regression.

We prefer Chebyshev polynomials when the value function is smooth. Chebyshev polynomials on $[-1, 1]$ are defined as $T_j(x) = \cos(j \cos^{-1}(x))$, and general Chebyshev polynomials on $[a, b]$ are defined as $T_j((2x - a - b)/(b - a))$ for $j = 0, 1, 2, \ldots$. Let $T_\alpha(x)$ denote the tensor product $T_{\alpha_1}(x_1) \cdots T_{\alpha_d}(x_d)$ for $x \in [-1, 1]^d$. Then the degree-n tensor Chebyshev approximation of $V(x)$ is

$$\hat{V}(x; c) = \sum_{0 \le \alpha_i \le n, 1 \le i \le d} c_\alpha T_\alpha(x).$$

The degree-n complete Chebyshev approximation for $V(x)$ is

$$\hat{V}(x; c) = \sum_{0 \le |\alpha| \le n} c_\alpha T_\alpha(x),$$

where $|\alpha|$ denotes $\sum_{i=1}^{d} \alpha_i$ for the nonnegative integer vector $\alpha = (\alpha_1, \ldots, \alpha_d)$. Computing a degree-$n$ complete Chebyshev polynomial is about $d!$ faster than computing a degree-n tensor Chebyshev polynomial, while the precision of approximation is almost the same for $n \ge 6$ usually in practice.

These three components of a DP method interact in many ways. For example, the approximation step can use Newton-style methods and be much faster if value function approximations are smooth. In the next sections, we describe two ways that enhance the usual methods. We first improve the approximation step with smooth shape-preserving methods. We also implement parallel methods. These two elements produce numerical methods that are stable, efficient, and able to solve large problems.

5. Shape Preservation

We often know that the unknown function is monotone and/or concave or convex. For our portfolio problem, we know that the value function is monotone and concave in wealth. The problem with ordinary methods—such as polynomials and splines combined with interpolation or regression—in the fitting step is that the result may be nonconcave or nonmonotone even if the data are consistent with monotonicity and concavity. As discussed in Judd (1998), this can lead to unstable value function iterations (VFIs). VFI works only if the fitting step preserves critical shape information. Discretization and piecewise linear interpolation preserve monotonicity and concavity. However, they make the objective function in the maximization step nondifferentiable, which forces one to use slow methods in the optimization step. Other shape-preserving methods such as the Schumaker method (1983) and the method used in Wang and Judd (2000) are only C^1 approximations of the value function, thereby slowing convergence of Newton-style optimization methods.

We use Chebyshev polynomials and least-squares approximation with shape constraints to guarantee shape preservation. The solution to the quadratic programming problem

$$\begin{aligned} \min_{c} \quad & \sum_{i=1}^{m} (\hat{V}(x_i; c) - v_i)^2 \\ \text{subject to} \quad & \hat{V}'(z_j; c) > 0, \ \hat{V}''(z_j; c) < 0, \ j = 1, \ldots, k, \end{aligned}$$

will produce a concave and monotone $\hat{V}(x; c)$, if we use a sufficiently dense set of points z_j to check shape. This method guarantees shape preservation but at a cost of allowing approximation errors $\hat{V}(x_i; c) - v_i$. Fortunately, these errors can be eliminated by using more basis functions in the approximation. Essentially, we have an underidentified constrained least-squares method. The possibility of multiple solutions is not important since our aim is only to produce some function that fits the data and has the desired global shape properties.

6. Parallelization

Parallelization allows researchers to solve huge problems and is the foundation of modern scientific computation. Our work shows that parallelization can also be used effectively in solving DP problems. The key fact is that at each maximization step, there are many independent optimization problems, one for each x_i. In our portfolio problems there are often thousands of such independent problems, and future problems will easily have millions of independent problems. We used the Condor system, a high-throughput computing framework, at the University of Wisconsin for our problems. We used the Master-Worker tool where a master processor allocates the individual optimization problems over a cluster of workstations. Parallelization of our portfolio problems is not trivial. We need to package collections of maximization problems to minimize the time spent on communication between master and the workers, and we need to avoid parallel blocks where the system needs to wait for a slow worker to complete a job.

We easily found methods for allocating work that made good use of parallelism. In one example we assumed six risky assets plus a bond, and solved a six-period problem with transaction costs. We used five Chebyshev nodes in each dimension, resulting in $5^6 = 15{,}625$ maximization problems in each VFI. We used 200 workers, and split the work into 3,125 tasks, each task solving five maximization problems. The six VFIs solved in less than 100 minutes, and the overall parallel performance was 87%, implying that the solution time was 174 times faster than a single computer. Many other problems in Cai (2009) attained 90–95% efficiency. Simple modifications of the parallel structure to minimize parallel blocks will improve parallel efficiency and allow us to efficiently use larger collections of computers.

7. Dynamic Portfolio Example

We next present an application of our algorithms to a dynamic portfolio problem. We assume three stocks and one bond, where the stock returns are log-normally distributed, $\log(R) \sim \mathcal{N}(\mu, \Sigma)$, $\mu = (0.0572, 0.0638, 0.07)$, and

$$\Sigma = \begin{bmatrix} 0.0256 & 0.00576 & 0.00288 \\ 0.00576 & 0.0324 & 0.0090432 \\ 0.00288 & 0.0090432 & 0.04 \end{bmatrix}.$$

We assume a power terminal value function with $\gamma = 3.5$, a transaction cost of $\tau = 0.01$ and $R_f = 1.0408$. We use the degree-7 complete Chebyshev approximation method and a multi-dimensional product Gauss-Hermite quadrature rule with nine nodes in each dimension to compute expectations. We assume a $T = 6$ investment horizon.

The key property of the solution is a no-trade region Ω_t for $t = 0, \ldots, 5$. When $x_t \in \Omega_t$, the investor will not trade at all, and when $x_t \notin \Omega_t$, the investor will trade to some point on the boundary of Ω_t. Because the value function has the form $W^{1-\gamma} g_t(x)$, the optimal portfolio rules and the "no-trade" regions Ω_t are independent of W. Figure 1 shows the no-trade regions for periods $t = 0, 1, 4, 5$. We see that the no-trade region grows as t approaches T. This corresponds to the intuition that an investor is not as likely to adjust his portfolio if he has to pay transaction costs and the holding time is short.

To the best of our knowledge, this is the first method to solve problems with more than three risky assets and more than a few periods. See Constantinides (1976), Janecek and Shreve (2004), and Muthuraman and Kumar (2006) for examples of portfolio choice analyses in the face of transaction costs.

8. Conclusion

Numerical DP can be made far more effective when we use modern methods from the numerical optimization and numerical approximation literature, and

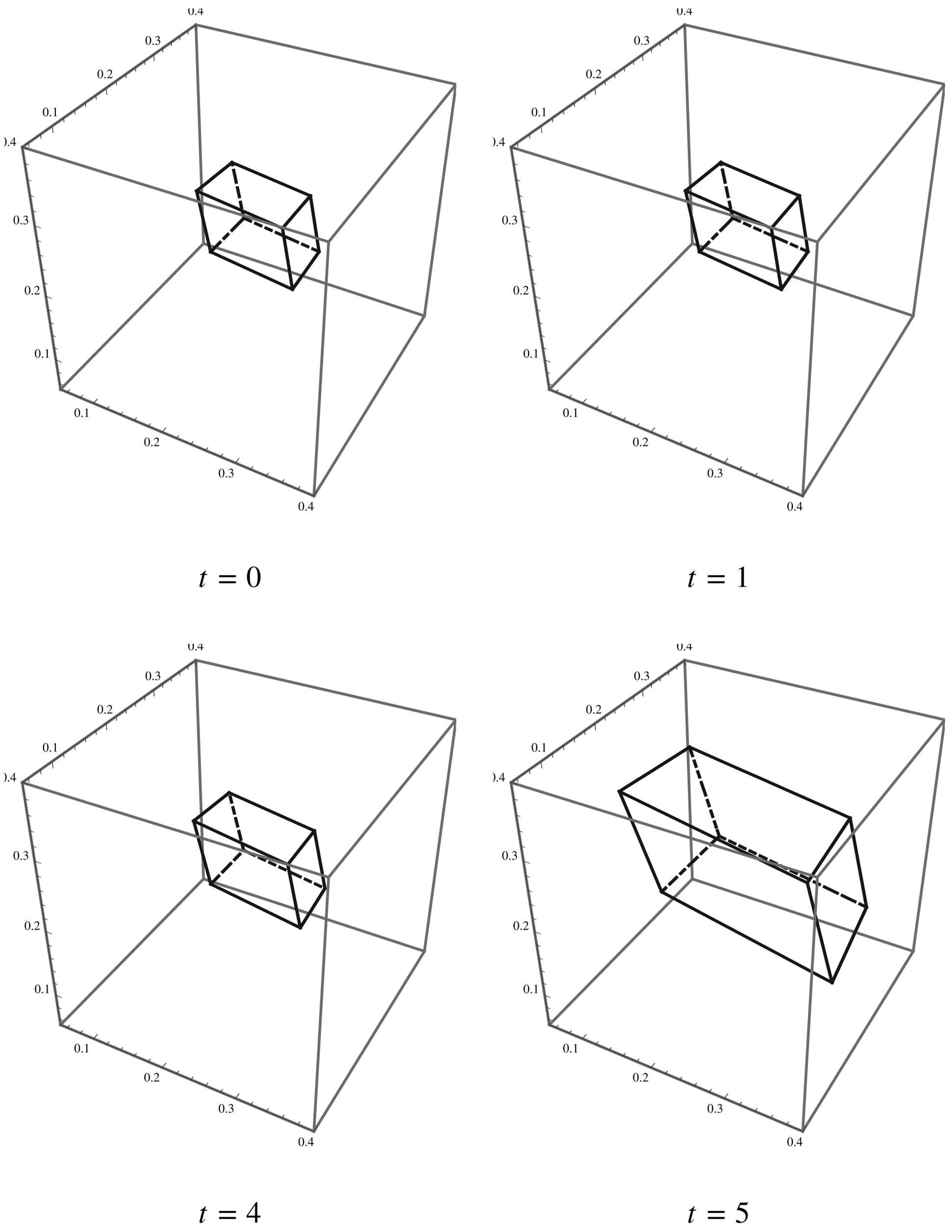

Figure 1. No-trade regions for three correlated stocks with log-normal returns and one bond.

when we use modern parallel computing systems. This paper has given just a few examples of how these ideas can be used in economics problems. See Cai (2009) for a more complete description of the methods described above and for many more examples.

References

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

Cai, Yongyang (2009). "Dynamic Programming and Its Application in Economics and Finance." Ph.D. thesis, Stanford University.

Constantinides, George (1976). "Optimal Portfolio Revision with Proportional Transaction Costs: Extension to HARA Utility Functions and Exogenous Deterministic Income." *Management Science*, 22, 921–923.

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.

Janecek, Karel, and Steven Shreve (2004). "Asymptotic Analysis for Optimal Investment and Consumption with Transaction Costs." *Finance and Stochastics*, 8, 181–206.

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

Muthuraman, Kumar, and Sunil Kumar (2006). "Multidimensional Portfolio Optimization with Proportional Transaction Costs." *Mathematical Finance*, 16, 301–335.

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

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

Wang, Sheng-Pen, and Kenneth Judd (2000). "Solving a Savings Allocation Problem by Numerical Dynamic Programming with Shape-Preserving Interpolation." *Computers & Operations Research*, 27, 399–408.