JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 54, 402–407 (1976)

Splines and Efficiency in Dynamic Programming

JAMES W. DANIEL*

Departments of Mathematics and of Computer Sciences, and Center for Numerical Analysis, The University of Texas at Austin, Austin, Texas

Submitted by Masanao Aoki

It is shown how one can use splines, represented in the B-spline basis, to reduce the difficulties of large storage requirements in dynamic programming via approximations to the minimum-return function without the inefficiency associated with using polynomials to the same end.

1. INTRODUCTION

A commonly discussed technique (Bellman [2], Bellman–Dreyfus [3], Bellman–Kalaba [4], Bellman–Kalaba–Kotkin [5], Peterson [14]), for reducing storage requirements in the fundamental recursions for the minimum-return function in dynamic programming is to approximate this function by linear combinations of a relatively small number of fixed functions. To describe the approach simply, we suppose that we are using dynamic programming to solve a control problem and that the state variable x is one-dimensional and has been normalized so that $|x| \leqslant 1$; the problem we are treating could be, for example, that of minimizing $\Phi[x(T)]$ subject to $dx/dt = G(x, y)$ on $0 \leqslant t \leqslant 1$, $x(0) = a$, with y restricted say to $c \leqslant y \leqslant d$. Using a discrete time-step of Δ, the recurrence formula of dynamic programming becomes (Peterson [14])

$$f_{k+1}(x) \equiv \min_y \{f_k[x + G(x, y)\,\Delta]\}, \qquad \text{for all } x \text{ in } [-1, 1], \tag{1.1}$$

with

$$f_1(x) \equiv \min_y \{\Phi[x + G(x, y)\,\Delta]\}. \tag{1.2}$$

The storage problem here is that we must store $f_k(x)$ at an enormous number of gridpoints x; the situation is even worse when x is multidimensional.

* Research supported in part by the Office of Naval Research under Contract N00014-67-A-0126-0015, NRO-044-425; reproduction in whole or in part is permitted for any purposes of the United States Government.

A standard way of reducing the storage requirements is to approximate each minimum-return function f_k as a linear combination of *fixed* functions $\{\varphi_i\}_1^M$, that is,

$$f_k(x) \sim \sum_{i=1}^{M} a_{ik}\varphi_i(x). \tag{1.3}$$

We then only need store the M coefficients $\{a_{ik}\}_{i=1}^M$ to represent $f_k(x)$ for all x; if M is not terribly large, then this results in a considerable storage saving. We do not address ourselves here to fundamentally different approaches for saving storage such as those expounded in Larson [13], and in Jacobson–Mayne [12]; it should be noted, however, that even those techniques require the approximation and representation of functions, so that the considerations of this present note are still relevant.

Returning again to the approximation method of Eq. (1.3), we remark that the storage savings clearly depend essentially only on M rather than on the form of the particular basis functions $\{\varphi_i\}$ being used. How then should we choose among the various $\{\varphi_i\}$? Clearly, we should consider questions of the approximation accuracy of the linear combinations of the $\{\varphi_i\}$, and we should consider the effort involved in computing with such approximations. Therefore, in this paper, we address ourselves to these questions of accuracy and efficiency. We use these two criteria for comparing the method based on spline approximations with the standard one (Bellman–Kalaba [4], Bellman–Kalaba–Kotkin [5], Peterson [14]) using polynomial approximations.

Before we proceed to this, we state in general how one uses this general approach to approximate the fundamental recursion given by Eqs. (1.1) and (1.2). We choose to approximate via interpolation at given points $\{x_i\}_1^M$. It is quite possible to allow the interpolation points $\{x_i\}_1^M$ and even the basis functions $\{\varphi_i\}_1^M$ to vary, for example with the stage number k or the neighborhood of x-values in which one is evaluating the return function; such an extension allows the consideration of the common practice of using an interpolation polynomial of moderate degree determined only by "local" data (Larson [13]), a technique which saves storage but does not lead to a very smooth approximation, unlike the spline methods (to be discussed), which are just as efficient. For notational convenience we do not explicitly indicate this possible variability of $\{x_i, \varphi_i\}_1^M$; no essential changes are introduced in our analysis by this simplification. The general process to be considered can be outlined as follows:

Step 1. Perform whatever initial computations can be used to simplify later steps.

Step 2. Compute $\hat{f}_1(x_i) = \min_y\{\Phi[x_i + G(x_i, y)\,\Delta]\}$, $1 \leqslant i \leqslant M$.

Step 3. Set $k = 1$.

Step 4. Compute the coefficients $\{a_{jk}\}$ so that $\sum_{j=1}^{M} a_{jk}\varphi_j(x_i) = \tilde{f}_k(x_i)$, $1 \leqslant i \leqslant M$.

Step 5. Compute $\tilde{f}_{k+1}(x_i) = \min_y\{\sum_{j=1}^{M} a_{jk}\varphi_j[x_i + G(x_i, y)\Delta]\}$, $1 \leqslant i \leqslant M$.

Step 6. Increase k by 1 and return to Step 4.

The above algorithm is, of course, terminated when $k + 1 = N = 1/\Delta$.

2. Polynomial Approximation

It is widely recommended (Bellman–Kalaba–Kotkin [5], Peterson [14], Bellman [2]), that one choose an orthonormal set of functions $\{\varphi_i\}$ such as trigonometric functions or classical orthogonal polynomials. To be precise, we follow the suggestion of Peterson [14] and assume that φ_i is a classical orthogonal polynomial of degree $i - 1$, such as

$$\varphi_i(x) = T_{i-1}(x) \equiv \cos[(i - 1) \arccos x], \tag{2.1}$$

the classical Chebyshev polynomials. Thus, each f_k is approximated by a polynomial of degree $M - 1$. If f_k has a bounded rth derivative, then f_k can be approximated by such polynomials to degree $\mathcal{O}(1/M^r)$ as M increases; this answers the question of the accuracy attainable by this choice of $\{\varphi_i\}$ so we turn our attention to matters of efficiency.

By choosing the interpolation points $\{x_i\}_1^M$ as the zeros of φ_{M+1}, for example, $x_i = \cos((2i - 1)\pi/2M)$ in the case of Eq. (2.1), we can exploit the orthogonality of the $\{\varphi_i\}$ in the algorithm of Section 1. In particular, the computation of the a_{jk} in Step 4 reduces to

$$a_{jk} = \beta_j \sum_{i=1}^{M} \tilde{f}_k(x_i)\, \varphi_j(x_i), \qquad 1 \leqslant j \leqslant M, \tag{2.2}$$

for some constants β_j depending only on the set $\{\varphi_i\}$. Thus, all the coefficients a_{jk}, $1 \leqslant j \leqslant M$, for fixed k, can be evaluated in about M^2 multiplications and M^2 additions.

Whatever method used to implement Step 5, certainly, for each y we must evaluate $\sum_{j=1}^{M} a_{jk}\varphi_j[x_i + G(x_i, y)\,\Delta]$. By careful exploitation of the usual threeterm recursion among orthogonal polynomials (Cheney [8], Clenshaw [9]), we can evaluate this expression in about M multiplications and M additions for each i and each y. To be explicit, if we have to evaluate at Q different points y, Step 5 will require about QM^2 multiplications and QM^2 additions.

Since we see that the costs in Step 2 are independent of $\{\varphi_i\}$, and since the costs in Step 1 for this case are just the evaluations of $\{x_i\}$, we find that our total costs for each value of k, $1 \leqslant k \leqslant N - 1$, is about $(Q + 1)\, M^2$

multiplications and additions. Since generally Q is much greater than one, the following is valid:

> Degree M polynomial approximation costs on the order of NQM^2 operations and yields $\mathcal{O}(1/M^r)$ accuracy for return functions having a bounded rth derivative. (2.3)

3. Spline Approximation

We now consider approximation by *splines of order r* (or degree $<r$) *having joints at the interpolation points* $\{x_i\}_1^M$. For convenience we assume that the x_i are equally spaced in $[-1, 1]$, so that $x_i = -1 + (i-1)h$ with $h \equiv 2/(M-1)$. Then such a spline S_r is an $(r-2)$-times continuously differentiable function on $[-1, 1]$, which equals a polynomial of degree not greater than $(r-1)$ in each interval (x_i, x_{i+1}) for $1 \leqslant i \leqslant M-1$. For information on splines, see, for example, Ahlberg, *et al.* [1], Greville [11], Schoenberg [15], and Schultz [16]. If f_k has a bounded rth derivative, then it can be approximated by such a spline S_r to an accuracy of $\mathcal{O}(1/M^r)$ just as for polynomials; since the dimension of this spline space is $M + r - 2$, and we do not usually take r to be large, we see that splines of order r give essentially the same approximation power as polynomials with roughly the same number of free parameters. Much more importantly, however, we can represent each spline as a linear combination of $M + r - 2$ special *B-splines* φ_i having the property that they vanish everywhere outside the interval $[x_i, x_{i+r}]$. Thus, in evaluating any function $\sum a_j\varphi_j(x)$, no more than r different B-splines φ_j can be nonzero at each x; since each B-spline can be efficiently evaluated (de Boor [6, 7], Cox [10]) at a cost independent of M, we see that the cost of evaluating $\sum a_j\varphi_j(x)$ is a small constant C independent of M. One can show that C is roughly $(3/2)\,r^2$ (de Boor [6, 7]). Therefore, in Step 5 of the algorithm in Section 1, we need perform only CMQ operations as opposed to M^2Q for polynomials.

To perform Step 4, we need to be able to interpolate by splines in the B-spline representation, that is, to solve $\sum_j a_j\varphi_j(x_i) = f(x_i)$ for the a_j. Actually, one usually interpolates at an additional $r-2$ points so as to determine uniquely the spline; this does not disturb the structure of the system of equations. Because of the small supports of the B-splines, we see that this system of equations is essentially r-banded, and hence, can be decomposed into a product of banded lower- and upper-triangular matrices in the order of M operations. Once we do this decomposition in Step 1, we can solve the interpolation problems of Step 4 in the order of M operations using this decomposition.

In summary, we see that there is a small integer C such that:

> Degree r spline approximation with M joints costs on the order of $CNQM$ operations and yields $\mathcal{O}(1/M^r)$ accuracy for return functions having a bounded rth derivative. (3.1)

This should be compared with Eq. (2.3), where we see that polynomial approximation with the same accuracy and the same storage requirements involves more work by a *factor* of roughly M, certainly a significant increase for even moderate M. Clearly, practitioners should seriously consider using spline approximation.

4. Higher Dimensions

Similar approximations can be employed in higher dimensions by using tensor products: In two dimensions, for example, we could approximate via $f(u, v) \sim \sum_{i,j} a_{ij}\varphi_i(u)\,\varphi_j(v)$. In p dimensions, one sees that polynomial approximation costs roughly QNM^{2p} compared with $CQNM^p$ for spline approximation.

References

1. J. H. Ahlberg, E. N. Nilson and J. L. Walsh, "The Theory of Splines and Their Applications," Academic Press, New York, 1967.
2. R. Bellman, "Introduction to the Mathematical Theory of Control Processes, Vol. II. Nonlinear Processes," Academic Press, New York, 1971.
3. R. Bellman and S. Dreyfus, "Applied Dynamic Programming," Princeton Univ. Press. Princeton, N. J., 1962.
4. R. Bellman and R. Kalaba, Reduction of dimensionality, dynamic programming, and control processes, P-1964, The RAND Corporation, Santa Monica, California, 1960.
5. R. Bellman, R. Kalaba and B. Kotkin, Polynomial approximation, a new computational technique in dynamic programming: allocation processes, *Math. Comp.* **17** (1973), 155–161.
6. C. de Boor, Subroutine package for calculating with B-splines, Los Alamos Scientific Lab. Report LA-4728-MS, 1971.
7. C. de Boor, On calculating with B-splines, *J. Approximation Theory* **6** (1972), 50–62.
8. E. W. Cheney, "Introduction to Approximation Theory," McGraw-Hill, New York, 1966.
9. C. W. Clenshaw, Chebyshev series for mathematical functions, Nat. Phys. Lab. Math. Tables, Vol. 5, Teddington, England, 1971.
10. M. G. Cox, The numerical evaluation of B-splines, Nat. Phys. Lab. Report DNAC4, Teddington, England, 1971.

11. T. N. E. GREVILLE, Ed., "Theory and Applications of Spline Functions," Academic Press, New York, 1969.
12. D. H. JACOBSON AND D. Q. MAYNE, "Differential Dynamic Programming," American Elsevier, New York, 1970.
13. R. E. LARSON, "State Increment Dynamic Programming," American Elsevier, New York, 1968.
14. E. L. PETERSON, Optimum multivariable control, *in* "Applied Combinatorial Mathematics," (E. F. Beckenbach, Ed.), pp. 253–283. Wiley, New York, 1964.
15. I. J. SCHOENBERG, "Approximations with Special Emphasis on Spline Functions," Academic Press, New York, 1970.
16. M. H. SCHULTZ, "Spline Analysis," Prentice–Hall, Englewood Cliffs, New Jersey, 1973.