FUNCTIONAL APPROXIMATIONS AND DYNAMIC PROGRAMMING

Richard Bellman
Stuart Dreyfus

P-1176

Revised 4-28-59

SUMMARY

In this note we wish to indicate some ways in which the theory of approximation can be used to increase the range of present day computers. Although we are primarily interested in applying these techniques to the functional equations occurring in the theory of dynamic programming, it should be noted that these same methods are applicable, and even more readily, to the classical functional equations of mathematical physics.

What we wish to do is to trade additional computing time, which is expensive, for additional memory capacity, which does not exist.

FUNCTIONAL APPROXIMATIONS AND DYNAMIC PROGRAMMING

Richard Bellman
Stuart Dreyfus

1. Introduction

In this note we wish to indicate some ways in which the theory of approximation can be used to increase the range of present day computers. Although we are primarily interested in applying these techniques to the functional equations occurring in the theory of dynamic programming, [1], it should be noted that these same methods are applicable, and even more readily, to the classical functional equations of mathematical physics.

What we wish to do is to trade additional computing time, which is expensive, for additional memory capacity, which does not exist.

2. Dimensionality Difficulties

A typical problem arising in the theory of control processes is that of maximizing a functional of the form

$$(1) \qquad J(y) = \int_0^T g(x,y)dt,$$

where x and y are N-dimensional vectors related by the differential equation

$$(2) \qquad \frac{dx}{dt} = h(x,y), \quad x(0) = c.$$

As we have discussed elsewhere at some length, [2], [3],

questions of this nature, although nominally within the domain of the calculus of variations, in actuality cannot be reduced to the point of numerical solution by means of classical techniques.

Writing

$$\text{(3)} \qquad \max_{y} J(y) = f(c,T),$$

the theory of dynamic programming replaces the foregoing variational problem by that of solving the nonlinear partial differential equation

$$\text{(4)} \qquad \frac{\partial f}{\partial T} = \max_{v} \left[g(c,v) + (h(c,v), \partial f/\partial c) \right],$$

$$f(c,0) = 0,$$

where

$$\text{(5)} \qquad \frac{\partial f}{\partial c} = \left(\frac{\partial f}{\partial c_1}, \frac{\partial f}{\partial c_2}, \ldots, \frac{\partial f}{\partial c_N} \right).$$

For computational purposes, it is often convenient to use the approximate difference equation

$$\text{(6)} \qquad f(c,T + \Delta) = \max_{v} \left[g(c,v)\Delta + f(c + h(c,v)\Delta, T) \right].$$

In terms of the capacities of modern computers, we have an extremely efficient algorithm if $N = 1$, a scalar problem, and a feasible algorithm if $N = 2$. If $N = 3$ or more, we face fast memory difficulties if we attempt to proceed in a

routine fashion.

The reason for this is the following. To store a function of N variables in the usual way, we tabulate the values of the function at a set of lattice points within the domain of interest. If there are M different possible values of c_1, of c_2, and so on, the total number of grid points will be M^N. For $M = 100$, and $N = 3$, this yields a quantity outside of present capabilities.

In various papers, [4], [5], we have indicated some methods which enable us to circumvent these difficulties. These methods combine analytic devices with the method of successive approximation. In this paper, we wish to present a new method, based upon approximation techniques, which appears to have wide applicability.

3. One-dimensional Case

In order to illustrate the application of the method in its simplest form, let us consider the problem of determining the sequence of functions $\{f_n(c)\}$, $n = 1,2,\ldots,$ given by the recurrence relation

$$f_1(c) = \max_v g(c,v),$$

$$f_{n+1}(c) = \max_v \left[g(c,v) + f_n(h(c,v))\right]. \tag{1}$$

Let us suppose that c takes values only over $[-1, 1]$ and that the function $h(c,v)$ similarly assumes values over this

interval for all c-values and all permissible v-values.

The standard approach involves a grid of values in the interval $[-1, 1]$ where the number of grid-points depends upon the accuracy that we desire. Let us proceed in a different manner. In place of considering that the function is determined by the set of grid-points, we shall consider the function to be determined by a Fourier expansion in terms of a suitable orthonormal set. For the interval $[-1, 1]$, a convenient set is the set of normalized Legendre polynomials. Thus, for some fixed value R, we write

$$(2) \qquad f_n(c) = \sum_{k=0}^{R} a_{k,n} P_k(c).$$

This is, of course, an approximation comparable with that of using a finite number of grid-points.

We could envisage using a power series expansion rather than an expansion in terms of orthogonal functions. Generally speaking, an orthogonal expansion is to be preferred, both on the grounds of accuracy and the grounds of ease of determination of the coefficients. As far as the calculation of the values of $P_k(c)$ are concerned, the simple three-term recurrence relations connecting the successive members of the sequence of Legendre polynomials make the computation of these values not much more difficult than that of the calculation of the powers c^k.

The function $f_n(c)$ is now replaced, as far as storage in the computer is concerned, by the set of coefficients

$$(3) \qquad A_n = \left[a_{0,n}, a_{1,n}, \ldots, a_{R,n}\right].$$

Whenever we wish a value of $f_n(c)$, we compute it by means of the approximate relation in (2). Consequently, in computing the values of $f_{n+1}(c)$ by way of (1), we require only the storage of the set of values given by A_n.

There is, however, a difficulty in this approach. The coefficients are determined by the relations

$$(4) \qquad a_{k,n} = \int_{-1}^{1} f_n(c)P_k(c)dc.$$

How are we going to calculate them? If we replace the integral by a simple Riemann sum, we are led back to the necessity for tabulating the values of $f_n(c)$ at a set of grid-points.

It is here that we invoke the theory of mechanical quadrature. In place of evaluating the integrals in (4) by means of Riemann sums, we use an interpolation formula of the following form:

$$(5) \qquad \int_{-1}^{1} f_n(c)P_k(c)dc = \sum_{j=1}^{S} a_j f_n(c_j)P_k(c_j),$$

where the a_j and c_j are carefully chosen.

If the quantities c_j are chosen to be the zeros of the Legendre polynomial of degree S, and the coefficients a_j are chosen to be the Christoffel numbers, the formula in (5) is exact if the integrand $f_n(c)P_k(c)$ is a polynomial of degree $2S - 1$.

Since the quantities a_j and c_j are tabulated, and in any case, could easily be determined in advance, the amount of effort entailed depends upon the choice of S. Since our original approximation, contained in (2), is equivalent to that of assuming that $f_n(c)$ is a polynomial of degree R, it would be reasonable to take S equal to R. However, there is no necessity for doing this.

If we use the equation in (5) to compute the values of the coefficient set A_n, it is clear that at each stage we require only the values $f_n(c_j)$, $j = 1,2,\ldots,R$.

The computation then proceeds in the following fashion. Given the values of $f_n(c_j)$, we compute the coefficient set A_n. Using these coefficients we can determine the values of $f_n(h(c,v))$ which occur in (1) in the course of computing the new values $f_{n+1}(c_j)$.

4. Discussion

Apart from the fixed set of instructions, and the values such as $a_j P_k(c_j)$ which are determined at the beginning of the process, we require the set $A_n = [a_{0,n}, a_{1,n}, \ldots, a_{R,n}]$ to be retained in the fast memory at the n-th stage. This is a set of $(R + 1)$ values.

On the other hand, a grid size of δ over $[-1, 1]$ would require $1/\delta$ values. In one dimension, the difference between $R + 1$ and $1/\delta$ is not particularly important, and the great amount of additional computation required by the method described above can more than outweigh this advantage.

Consider, however, the two-dimensional case. The straightforward approach based upon a grid size of δ in the c_1 and c_2 intervals requires $(1/\delta)^2$ values of $f_n(c_1,c_2)$. On the other hand, if we set

$$(1) \qquad f_n(c_1,c_2) = \sum_{i,j=0}^{R} a_{ij,n} P_i(c_1) P_j(c_2),$$

and proceed as above, we require only $R(R+1)/2$ values, the coefficients $a_{ij,n}$, $i,j = 0,1,\ldots,R$.

Proceeding to three dimensions, we compare $(1/\delta)^3$ and $R(R+1)(R+2)/6$, the number of coefficients $a_{ijk,n}$. Let us use some typical values of δ and R and compare the values, say $\delta = .01$ and $R = 5,10$.

Dimension	$(1/\delta)^N$	R = 5	R = 10
1	100	6	11
2	10^4	21	55
3	10^6	56	286
4	10^8	126	1001
5	10^{10}	252	3003

We see that variational problems involving four and five state variables which are completely untouchable by direct methods are within the scope of the method we have outlined. Combining this method with the Lagrange multiplier technique, and the method of successive approximations, we have a way of attacking previously impregnable problems.

5. A Numerical Example

Consider the problem of determining the sequence $\{v_n\}$ so as to minimize the function

$$(1) \qquad F_N = \sum_{k=1}^{N} u_k^2 + \lambda \sum_{k=1}^{N} v_k^2,$$

where

$$(2) \quad (a) \quad u_{n+1} = 2u_n - u_n^2 + v_n, \quad u_0 = c,$$

(b) v_n must be chosen so as to keep u_{n+1} within the interval $[-1, 1]$.

Introducing for $N \geq 1$ and $-1 \leq c \leq 1$, the function

$$(3) \qquad f_N(c) = \operatorname*{Min}_{\{v\}} F_N,$$

we derive the recurrence relation

$$(4) \qquad f_N(c) = \operatorname*{Min}_{v} \left[c^2 + \lambda v^2 + f_{N-1}(2c - c^2 + v) \right], \quad N \geq 2,$$

$$f_1(c) = c^2.$$

This yields a simple computational determination of the sequence $\{f_N(c)\}$.

As a test of the foregoing method, this sequence was first determined by the usual method based upon a grid of values over $[-1, 1]$, and then following the procedure described above, using the approximation

$$(5) \qquad f_N(c) = \sum_{k=0}^{6} a_{k,N} P_k(c),$$

and a value of $S = 14$ in (3.5).

Let $\{f_k(c)\}$ denote the sequence determined using the grid, and $\{f_k^*(c)\}$ the sequence obtained via Legendre polynomials. A comparison of values is given below for $k = 6$.

c	$f_6^*(c)$	$f_6(c)$
1.0	1.782	1.77
.8	1.370	1.36
.2	.153	.145
0.0	.006	0.0
− .2	.202	.20
− .8	4.876	4.89
−1.0	8.666	8.67

As can be seen, the agreement is quite good.

REFERENCES

1. R. Bellman, Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.

2. R. Bellman, "On the application of the theory of dynamic programming to the study of control processes," Proc. Symposium on Control Processes, Polytechnic Inst. of Brooklyn, April, 1956, pp. 199-213.

3. R. Bellman and R. Kalaba, On Adaptive Control Processes, The RAND Corporation, Paper P-1610, February 23, 1959.

4. R. Bellman, "Dynamic programming and Lagrange multipliers," Proc. Nat. Acad. Sci. USA, vol. 42, 1956, pp. 767-769.

5. R. Bellman, "Some new techniques in the dynamic programming solution of variational problems," Q. Appl. Math., vol. 16, 1958, pp. 295-305.