ARTHUR C. DOWNING, Jr., Editor

Further Remarks on Line Segment Curve-Fitting Using Dynamic Programming*

BRIAN GLUSS

Armour Research Foundation of Illinois Institute of Technology

In a recent paper, Bellman showed how dynamic programming could be used to determine the solution to a problem previously considered by Stone. The problem comprises the determination, given N, of the N points of subdivision of a given interval (α, β) and the corresponding line segments, that give the best least squares fit to a function $g(x)$ in the interval. Bellman confined himself primarily to the analytical derivation, suggesting briefly, however, how the solution of the equation derived for each particular point of subdivision u_i could be reduced to a discrete search. In this paper, the computational procedure is considered more fully, and the similarities to some of Stone's equations are indicated. It is further shown that an equation for u_i involving no minimization may be found. In addition, it is shown how Bellman's method may be applied to the curve-fitting problem when the additional constraints are added that the ends of the line segments must be on the curve.

Introduction

Stone [1] recently posed the problem of determining the best least squares fit to a prescribed function $g(x)$ in the range (α, β) given that N points of subdivision $u_1, \cdots, u_N$ are permitted and that, within the $N + 1$ subintervals thus formed, line segments

$$y = a_j + b_j x \qquad u_{j-1} \leqq x \leqq u_j \tag{1}$$

are fitted, where $u_0 = \alpha$ and $u_{N+1} = \beta$. That is, it is required to minimize the function

$$F(a_1, a_2, \cdots, a_{N+1}; b_1, b_2, \cdots, b_{N+1}; u_1, u_2, \cdots, u_N) = \sum_{j=1}^{N+1} \int_{u_{j-1}}^{u_j} [g(x) - a_j - b_j x]^2 \, dx \tag{2}$$

over all a_j, b_j and $\alpha \leqq u_1 \leqq u_2 \leqq \cdots \leqq u_N \leqq \beta$.

* This work was supported by the Air Force System Command, U. S. Air Force, under contract AF 33(616)-8240 with the Armour Research Foundation of Illinois Institute of Technology.

Stone obtained a method of solution of the problem in a classical manner, and indicated the computational method involved, while Bellman in a later paper [2], indicated a dynamic programming [3] method of solution. We shall discuss below the computational aspects of the latter method, noting where there are similarities to some of Stone's equations, and shall demonstrate that an equation for u_j may be determined that involves an equality sign rather than a minimization. In other words, we may reduce the problem analytically to a stage where in order to determine u_j computationally, it is only necessary to solve an equation rather than minimize a function. We shall further show how dynamic programming produces computationally simple equations for the model in which the ends of the line segments are constrained to lie on the curve $g(x)$.

In order to make this discussion self-contained, it will be necessary to repeat the functional equations that Bellman derives.

Further Remarks on Bellman's Solution

If we define

$$f_N(\beta) = \min_{[a_j, b_j, u_j]} F, \tag{3}$$

then the functional equations derived in [2] are

$$f_1(\beta) = \min_{[a_1, a_2, b_1, b_2, u_1]} \left[\int_\alpha^{u_1} [g(x) - a_1 - b_1 x]^2 \, dx + \int_{u_1}^{\beta} [g(x) - a_2 - b_2 x]^2 \, dx \right] \tag{4}$$

where $\alpha \leqq u_1 \leqq \beta$, and

$$f_N(\beta) = \min_{\alpha \leqq u_N \leqq \beta} \left[\min_{[a_{N+1}, b_{N+1}]} \cdot \int_{u_N}^{\beta} [g(x) - a_{N+1} - b_{N+1} x]^2 \, dx + f_{N-1}(u_N) \right] \tag{5}$$

Least Square Equations for a_{N+1}, b_{N+1} in terms of u_N. Differentiating partially with respect to a_{N+1} and b_{N+1}, we obtain equations for a_{N+1}, b_{N+1} in terms of u_N similar to Stone's. That is,

$$\int_{u_N}^{\beta} [g(x) - a_{N+1} - b_{N+1} x] \, dx = 0, \tag{6}$$

and

$$\int_{u_N}^{\beta} x[g(x) - a_{N+1} - b_{N+1} x] \, dx = 0, \tag{7}$$

from which it follows that

$$a_{N+1} = \frac{4(\beta^2 + u_N \beta + u_N^2) I(\beta, u_N) - 6(\beta + u_N) J(\beta, u_N)}{(\beta - u_N)^3}, \tag{8}$$

and

$$b_{N+1} = \frac{6[2J(\beta, u_N) - (\beta + u_N)I(\beta, u_N)]}{(\beta - u_N)^3} \tag{9}$$

where

$$\left.\begin{aligned} I(\beta, u_N) &= \int_{u_N}^{\beta} g(x)\,dx, \\ \text{and} \qquad & \\ J(\beta, u_N) &= \int_{u_N}^{\beta} xg(x)\,dx. \end{aligned}\right\} \tag{10}$$

Equation for u_N. Differentiating equation (5) with respect to u_N, the optimal u_N is given by

$$-2\int_{u_N}^{\beta}\left[\frac{\partial a_{N+1}}{\partial u_N} + x\frac{\partial b_{N+1}}{\partial u_N}\right][g(x) - a_{N+1} - b_{N+1}x]\,dx - [g(u_N) - a_{N+1} - b_{N+1}u_N]^2 + f'_{n-1}(u_N) = 0$$

which, using equations (6) and (7), reduces to

$$[g(u_N) - a_{N+1} - b_{N+1}u_N]^2 - f'_{N-1}(u_N) = 0. \tag{11}$$

It should be pointed out, in order to be more rigorous, that the optimal u_N either satisfies equation (11) or equals α or β. Since $u_N = \alpha$ implies that using no segmentation is best, and $u_N = \beta$ that $N - 1$ segmentation points are best even if we are allowed N—and this is obviously not true since we can always improve on a given $(N - 1)$-point segmentation by splitting one of the subintervals in two, so that $f_N(\beta)$ is a monotonically decreasing function of N—hence it follows that equation (11) holds in general.

Method of Computational Solution. It is now clear how simple computationally it is to determine the solution. Given a table for the previously computed $f_{N-1}(y)$, $\alpha \leqq y \leqq \beta$, and hence a table of $f'_{N-1}(y)$, and also tables for $I(\beta, u_N)$ and $J(\beta, u_N)$—or alternatively the indefinite integrals $\int g(x)\,dx$ and $\int xg(x)\,dx$—, we solve equation (11) for u_N by successive approximation, after the expressions for a_{N+1} and b_{N+1} in equations (8) and (9) have been substituted in equation (11). From this value of u_N, we obtain a_{N+1} and b_{N+1} from equations (8) and (9), then $f_N(\beta)$ is determined by substituting for a_{N+1}, b_{N+1}, u_N in equation (5).

This procedure is used to compute successively $f_2, f_3, \cdots$, and the a_{N+1}, b_{N+1}, u_N associated with them. f_1 is first obtained by solving the five equations obtained by differentiating equation (4) partially with respect to a_1, a_2, b_1, b_2, and u_1.

Note that u_N could also have been obtained, of course, by substituting for a_{N+1} and b_{N+1} in equation (5) and making a discrete search within α, β.

Model With Additional Constraints

We shall now indicate briefly how the same dynamic programming approach may be applied to another line segmentation model which has considerable practical interest. Unlike in the previous model, where broken line segments are permitted, the constraint is added that they must be connected at the points $u_1, \cdots, u_N$, and that these meeting points must lie on the curve. That is, the line segments pass through $g(\alpha)$, $g(u_1)$, $\cdots$, $g(u_N)$, $g(\beta)$. (By exactly the same method, we could also consider the model in which the lines must meet at $u_1, \cdots, u_N$, but need not necessarily meet on the curve.)

The function it is now required to minimize is

$$F(u_j) = \sum_{j=1}^{N+1}\int_{u_{j-1}}^{u_j}\left[g(x) - \left\{\frac{(u_j - x)g(u_{j-1}) + (x - u_{j-1})g(u_j)}{u_j - u_{j-1}}\right\}\right]^2 dx, \tag{12}$$

and the general functional equation is given by

$$f_N(\beta) = \min_{\alpha \leqq u_N \leqq \beta}\left[\int_{u_N}^{\beta}\left[g(x) - \left\{\frac{(\beta - x)g(u_N) + (x - u_N)g(\beta)}{\beta - u_N}\right\}\right]^2 dx + f_{N-1}(u_N)\right], \tag{13}$$

with $f_1(\beta)$ given by

$$f_1(\beta) = \min_{\alpha \leqq u_1 \leqq \beta}\left[\int_{\alpha}^{u_1}\left[g(x) - \left\{\frac{(u_1 - x)g(\alpha) + (x - \alpha)g(u_1)}{u_1 - \alpha}\right\}\right]^2 dx + \int_{u_1}^{\beta}\left[g(x) - \left\{\frac{(\beta - x)g(u_1) + (x - u_1)g(\beta)}{\beta - u_1}\right\}\right]^2 dx\right] \tag{14}$$

As with the previous model, we now obtain by partial differentiation with respect to u_N the following equation for u_N:

$$f'_{N-1}(u_N) = 2\int_{u_N}^{\beta}\left[g(x) - \left\{\frac{(\beta - x)g(u_N) + (x - u_N)g(\beta)}{\beta - u_N}\right\}\right]\left(\frac{\beta - x}{\beta - u_N}\right) \cdot \left[g'(u_N) + \frac{g(u_N) - g(\beta)}{\beta - u_N}\right] dx. \tag{15}$$

Although at first sight this may appear to be a complicated integral, on observation it will be noted that, once again, all it involves are the functions $I(\beta, u_N)$, $J(\beta, u_N)$ and $g'(u_N)$. In fact, of course, equation (15) may be simplified still further, since the last square bracket may be taken outside the integral as it is independent of x, and then the second term in the first square bracket multiplied by $(\beta - x)/(\beta - u_N)$ may easily be integrated. Hence the equation reduces to

$$f'_{N-1}(u_N) = \frac{2}{\beta - u_N}\left[g'(u_N) + \frac{g(u_N) - g(\beta)}{\beta - u_N}\right] \cdot \left[\beta I(\beta, u_N) - J(\beta, u_N) - (\beta - u_N)^2\left(\frac{g(u_N)}{3} + \frac{g(\beta)}{6}\right)\right] \tag{16}$$

Hence, as before, given tables of $f_{N-1}(\beta)$—and hence $f'_{N-1}(\beta)$—that were computed from those of $f_{N-2}(\beta)$, equation (16) is now easily solved computationally for u_N, using successive approximation. The $f_N(\beta)$ is obtained by substituting this value of u_N in equation (13), for which we also need

$$\int_{u_N}^{\beta} [g(x)]^2\, dx = S(\beta, u_N), \text{ say.}$$

One could alternatively determine u_N with greater computer time consumption, by successive approximation after a discrete search, from equation (13) directly. The integral in this equation reduces to

$$S(\beta, u_N) - \frac{2}{\beta - u_N}\left\{[\beta g(u_N) - u_N g(\beta)]I(\beta, u_N) + [g(\beta) - g(u_N)]J(\beta, u_N) + \frac{(\beta - u_N)}{6}[g(\beta) + g(u_N)]^2\right\}. \tag{17}$$

It should finally be noted that in order to obtain $f_N(\beta)$, the ranges over which $f_1, f_2, \cdots, f_{N-1}$ must be calculated are all $[\alpha, \beta]$.

REFERENCES

1. STONE, H. Approximation of curves by line segments. *Math. Comput. 15*, 73 (Jan. 1961), 40–47.
2. BELLMAN, R. On the approximation of curves by line segments using dynamic programming. *Comm. ACM 4*, 6 (June 1961), 284.
3. BELLMAN, R. *Dynamic Programming*. Princeton University Press, 1957.

A Set of Matrices for Testing Computer Programs†

J. L. BRENNER

Stanford Research Institute, Menlo Park, California

1. Introduction

In this article, a set of matrices is defined, and the complete solution of each matrix is given. The matrices have arbitrary dimension, and a large number of parameters. By adjusting these parameters, a matrix of arbitrary dimension can be constructed which has two eigenvectors that are nearly coincident, which is deficient (has fewer than the maximum possible number of eigenvectors) or is as nearly singular as desired. Thus a set of matrices is available which forms a convenient set of check matrices for testing computer programs which purport to solve[1] or invert a matrix. The matrices of the set enjoy certain properties of regularity to which the term symmetry might well be applied; however, the matrices are not necessarily symmetric in the ordinary sense.

For selected values of the parameters, the matrices appear as variance-covariance matrices in statistics. In particular cases, the eigenvalues (but not the eigenvectors) and determinants have been found [1]. The methods of this note are shorter, more elementary, and more effective; from them, immediate insight into the matrix structure is obtained.

2. Basic Case

Let f_n be the $1 \times n$ column-vector whose entries are all 1's; f_n^* is the corresponding $n \times 1$ row-vector. For arbitrary n, k, write $J_{nk} = f_n f_k^*$; thus J_{nk} is the $k \times n$ matrix whose entries are all 1's. The matrix J_{nn} has the following properties: f_n is an eigenvector (with eigenvalue n); every vector v orthogonal to f_n $[f_n^* v = 0]$ is an eigenvector, with eigenvalue 0. This space of vectors v is spanned by

$$\{g_i\}, \quad g_i = f_n - n e_n^{\,i}, \qquad i = 1, 2, \cdots, n,$$

and has ex.gr. the first $n - 1$ of these for a basis. Here $e_n^{\,i}$ is the $1 \times n$ vector $[0, \cdots, 0, 1_i, 0, \cdots, 0]^*$. This proves the following theorem.

THEOREM 1. *By a change of basis, the matrix* $aI_n + bJ_{nn}$ *can be transformed to diag* $[a, \cdots, a, a + bn]$.

Note that the set of $n \times n$ matrices $aI_n + bJ_{nn}$ forms a semi-group, and the invertible ones form a group. The inverse of $aI_n + bJ_{nn}$ is $a^{-1}I_n - ba^{-1}(a + bn)^{-1}J_{nn}$.

3. General Case

The general matrix that can be solved by the above methods is the partitioned matrix:

$$A = \begin{bmatrix} a_1 I_{n_1} + b_{11} J_{n_1 n_1}, & b_{12} J_{n_1 n_2}, & \cdots, & b_{1t} J_{n_1 n_t} \\ b_{21} J_{n_2 n_1}, & a_2 I_{n_2} + b_{22} J_{n_2 n_2}, & \cdots, & b_{2t} J_{n_2 n_t} \\ \vdots & \vdots & & \vdots \\ b_{t1} J_{n_t n_1}, & \cdots, & & a_t I_{n_t n_t} + b_{tt} J_{n_t n_t} \end{bmatrix}$$

By choosing a suitable basis, the matrix A can be transformed into

$$B = \text{diag}[A_1, A_2 \ \cdots, A_t, A_{t+1}],$$

where for $i = 1, 2, \cdots, t$, A_i is the $n_i - 1 \times n_i - 1$ scalar

† Sponsored (in part) by Mathematics Research Center, U. S. Army, Madison, Wisconsin, under Contract No. DA-11-022-ORD-2059.

[1] I.e., find the eigenvalues and corresponding invariant subspaces.