AAEC/E229

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

PIECEWISE-LINEAR APPROXIMATION TO CONTINUOUS FUNCTIONS

by

A.ISAACS

June 1972

ISBN 0 642 99476 5

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

PIECEWISE-LINEAR APPROXIMATION TO CONTINUOUS FUNCTIONS

by

A. ISAACS

ABSTRACT

Methods are examined for finding an optimal least-squares approximation to a continuous function on a finite interval by a continuous piecewise-linear function with a specified number of linear segments. A practical method and a corresponding computer program are presented which use the solution of the simpler problem where the approximating function is not constrained to be continuous. The topic arises in connection with the use of diode function generators to represent functions on an analogue computer.

National Library of Australia card number and ISBN 0 642 99476 5

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

ANALOG COMPUTERS; ERRORS; FORTRAN; FUNCTIONS; HILBERT SPACE; IBM COMPUTERS; INTEGRAL EQUATIONS; LEAST SQUARE FIT; MATHEMATICS; PROGRAMMING; RELIABILITY

CONTENTS

1. INTRODUCTION

This report concerns methods for finding an optimal least-squares approximation to a continuous function on a finite interval by a continuous piecewise-linear function with a specified number of linear segments. A precise statement of the problem and a review of various possible methods of solution are given in Sections 1 - 5; a practical method of solution is then given. Appendices follow which include details of the use of a computer program written to carry out this method and results from the program.

The topic arises in connection with the representation of functions on an analogue computer using diode function generators, which set up continuous piecewise-linear functions with up to some maximum number of segments. Since the noise level of the apparatus increases as the number of segments is increased, an acceptable approximation is required with as few segments as possible. Thus for any given number of segments we require a near optimal approximation.

In many ways the problem is similar to a pattern-recognition problem. We require an approximation method that will recognise any large peak in the graph of the function ($f(x)$) to be approximated, except that when fitting a limited number of segments it is the 'overall shape' which is important. Direct methods of approaching the problem, namely solution of the normal equations or use of a dynamic programming search, are not practically useful as they stand.

The main point of the method to be presented in this report is the use of a simpler related problem, namely the problem of approximating a continuous function by a piecewise-linear function which is not required to be continuous, to obtain an initial approximation which may then be used with one of the methods mentioned above.

Except for the paper by Gluss (1964), the literature on this problem is concerned with approximate methods which in general produce near-optimal approximations only if $f(x)$ has a simple form.

Norkin (1962) takes the simplest approach using equally spaced break points (the points where the slope of the approximation changes) and Hilbert space theory. Vansteenkiste (1967) improves on this using the curvature of $f(x)$ to choose the break points.

Phillips (1968) presents a method using the L_∞ norm assuming that $f(x)$ has no point of inflexion in the range considered. Hamer (1956) and Ream (1959) have presented methods for both the L_∞ and L_2 norms but assume that $f(x)$ has no point of inflexion and that where each segment is fitted, $f(x)$ will be approximately a quadratic polynomial.

The method of Gluss (1964) has none of the above restrictions; the break points are chosen by fitting a step function to the first derivative of $f(x)$. However this method does not have the advantages, discussed in Section 6, of our method. Gluss' method is described in more detail in Appendix A.

Stone (1961) and Bellman (1964) consider the simpler approximation problem in which the segments are independent, Stone using the normal equations and Bellman using a dynamic programming search. We examine Bellman's method in more detail in Section 5.

2. PRECISE STATEMENT OF THE PROBLEM

Let $A_c(n)$ denote the set of all continuous piecewise-linear functions defined on $[0,1]$ which have n distinct break points. The general form of a function $h(x) \in A_c(n)$ is as follows:

$$h(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\, x - \frac{y_{i+1}x_i - y_i x_{i+1}}{x_{i+1} - x_i} \quad \text{if } x_i \leq x \leq x_{i+1} \qquad i = 0,1,\ldots,n \qquad \ldots(1)$$

where $0 = x_o < x_1 < x_2 < \ldots < x_n < x_{n+1} = 1$.

The break points are $x_1, x_2, \ldots, x_n$ and the function consists of n+1 line segments

$$\ell_i(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\, x - \frac{y_{i+1}x_i - y_i x_{i+1}}{x_{i+1} - x_i} \quad \left(x_i \leq x \leq x_{i+1}\right) \qquad \ldots(2)$$

$$= \ell_i(x_i, x_{i+1}, y_i, y_{i+1}; x) \qquad i = 0,1,2,\ldots,n$$

which meet at the break points, that is which satisfy

$$\ell_i(x_{i+1}) = \ell_{i+1}(x_{i+1}) \qquad i = 0,1,\ldots,n-1. \qquad \ldots(3)$$

Thus $h(x)$ is obtained by joining the points $(0,y_o)$, $(x_1,y_1),\ldots,(x_n,y_n)$, $(1,y_{n+1})$ with straight lines $\ell_i(x)$, $i = 0,1,\ldots,n$. We use the above notation throughout.

The problem is, given a function $f(x)$ which is continuous on $[0,1]$, to choose a function $h(x) \in A_c(n)$ which minimises some norm of $f-h$

$$N(f-h) = N(x_1,x_2,\ldots,x_n,y_o,y_1,\ldots,y_{n+1}) \quad .$$

The norm which we will consider is the least-squares norm. The square of this norm is given by

$$L_2(f-h) = L_2(x_1,x_2,\ldots,x_n,y_o,y_1,\ldots,y_{n+1})$$

$$= \int_0^1 \left[f(x) - h(x) \right]^2 dx \qquad \ldots(4)$$

$$= \sum_{i=o}^{n} \int_{x_i}^{x_{i+1}} \left[f(x) - \ell_i(x) \right]^2 dx \quad .$$

We generalise this norm by including a weighting function $w(x)$ defined and continuous on $[0,1]$ and such that $w(x) \geq 0$ for all $x \in [0,1]$ and $\int_0^1 w(x)dx = 1$. Hence we wish to minimise

$$L_2(f-h) = \int_0^1 w(x) \left[f(x) - h(x)\right]^2 dx$$

$$= \sum_{i=o}^{n} \int_{x_i}^{x_{i+1}} w(x) \left[f(x) - \ell_i(x)\right]^2 dx. \qquad \ldots(5)$$

Throughout this report 'approximation' will be taken to mean 'line segment approximation'. Thus we will refer to the above problem as the 'continuous approximation problem'.

We also consider a simpler auxiliary problem - that of an approximation with the segments independent of one another (that is, the segments are not constrained to meet at the break points and so the approximation need not be continuous). We will refer to this problem as the 'approximation problem with independent segments' or as the 'independent approximation problem'.

For this problem the segments have the form

$$\ell_i(x) = \ell_i(x_i, x_{i+1}, y_i^{(1)}, y_i^{(2)}; x)$$

$$= \frac{y_i^{(2)} - y_i^{(1)}}{x_{i+1} - x_i} \; x - \frac{y_i^{(2)} x_i - y_i^{(1)} x_{i+1}}{x_{i+1} - x_i}$$

$$(x_i \leq x \leq x_{i+1}) \quad . \qquad \ldots(6)$$

Thus

$$\ell_i(x_i) = y_i^{(1)}, \quad \ell_i(x_{i+1}) = y_i^{(2)}$$

and in general $y_i^{(2)} \neq y_{i+1}^{(1)}$.

An independent approximation is thus a piecewise-linear function defined on the interval [0,1] with n distinct breakpoints between 0 and 1. The space of all such functions will be denoted by A(n).

Two other norms which might be used are the L_1 and L_∞ norms given by

$$L_1(f-h) = \int_0^1 w(x) \; | \; f(x) - h(x) \; | \; dx \qquad \ldots(7)$$

and

$$L_\infty(f-h) = \max_{0 \leq x \leq 1} \; (w(x) \; | \; f(x) - h(x) \; |) \quad . \qquad \ldots(8)$$

However we restrict ourselves to using the L_2 norm. Because of its analytical properties, this norm is the easiest to work with. It is a continuous function of each of its 2n+2 variables and is infinitely differentiable with respect to them; it is the norm in a Hilbert space and the theory of approximation in a Hilbert space is available to us (see Section 4); its properties as an integral make it suitable for the use of a dynamic programming approach (see Section 5). Moreover, for the application we have in mind here, there is little difference between the use of the various norms.

3. THE NORMAL EQUATIONS

We now examine the straightforward approach of solving the normal equations:

$$\frac{\partial L}{\partial x_i} = 0 \qquad i = 1,2,\ldots,n$$

$$\frac{\partial L}{\partial y_i} = 0 \qquad i = 0,1,\ldots,n+1 \qquad \ldots(9)$$

We derive the equations and then consider the practical aspects of their solution. We need the following result from calculus.

Let $k(u,x)$ be a real-valued function of two variables which is continuous and differentiable with respect to each of its variables. Let

$$g(u) = \int_a^u k(u,x)dx$$

where 'a' is a constant. Then

$$g'(u_o) = \int_a^{u_o} \frac{\partial k(u_o,x)}{\partial u}\, dx + k(u_o,u_o) \qquad \ldots(10)$$

where $g'(u_o)$ and $\frac{\partial k(u_o,x)}{\partial u}$ denote derivatives with respect to u evaluated at the point $u=u_o$.

If we apply this result and use the fact that $\ell_i(x_{i+1}) = \ell_{i+1}(x_{i+1})$ for $i = 0,1,\ldots,n-1$ we obtain the following equations:

$$\left.\begin{aligned} &\int_{x_{i-1}}^{x_i} w(x)\,[f(x) - \ell_{i-1}(x)]\,\frac{\partial \ell_{i-1}(x)}{\partial x_i}\,dx - \\ &\quad - \int_{x_i}^{x_{i+1}} w(x)\,[f(x) - \ell_i(x)]\,\frac{\partial \ell_i(x)}{\partial x_i}\,dx = 0 \end{aligned}\right| \quad \text{for } i = 1,2,\ldots,n$$

$$\left.\begin{aligned} &\int_{x_{i-1}}^{x_i} w(x)\,[f(x) - \ell_{i-1}(x)]\,\frac{\partial \ell_{i-1}(x)}{\partial y_i}\,dx + \\ &\quad + \int_{x_i}^{x_{i+1}} w(x)\,[f(x) - \ell_i(x)]\,\frac{\partial \ell_i(x)}{\partial y_i}\,dx = 0 \end{aligned}\right| \quad \text{for } i = 1,2,\ldots,n \qquad \ldots(11)$$

$$\int_{x_o}^{x_1} w(x)\,[f(x) - \ell_o(x)]\,\frac{\partial \ell_o(x)}{\partial y_o}\,dx = 0$$

$$\int_{x_n}^{x_{n+1}} w(x) \quad [f(x) - \ell_n(x)] \quad \frac{\partial \ell_n(x)}{\partial y_{n+1}} \quad dx = 0 \quad . \qquad \Bigg] \quad \ldots(11)$$

We consider briefly some practical computational aspects of these equations and of the use of Newton's method to solve them.

Now

$$\left.\begin{aligned}
&\frac{\partial \ell_i(x)}{\partial x_i} = \frac{y_{i+1} - y_i}{(x_{i+1} - x_i)^2}\, x - \frac{y_{i+1} - y_i}{(x_{i+1} - x_i)^2}\, x_{i+1} \quad , \\
&\qquad\qquad i = 1,2,\ldots,n \\
&\frac{\partial \ell_{i-1}(x)}{\partial x_i} = -\frac{y_i - y_{i-1}}{(x_i - x_{i-1})^2}\, x + \frac{y_i - y_{i-1}}{(x_i - x_{i-1})^2}\, x_{i-1} \quad , \\
&\qquad\qquad i = 1,2,\ldots,n \\
&\frac{\partial \ell_i(x)}{\partial y_i} = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad , \\
&\qquad\qquad i = 0,1,\ldots,n \\
\text{and} \quad &\frac{\partial \ell_{i-1}(x)}{\partial y_i} = \frac{x - x_{i-1}}{x_i - x_{i-1}} \\
&\qquad\qquad i = 1,2,\ldots,n
\end{aligned}\right\} \quad \ldots(12)$$

which are all linear in x. Hence equations (11) will involve only the integrals

$$\int w(x)\ f(x)\ x\ dx, \int w(x)\ f(x)\ dx$$

and $$\int w(x)\ x^2\ dx, \int w(x) x dx, \int w(x)\ dx \ .$$

To solve equations (11) in a computer program we would require the values of these integrals between arbitrary limits in the interval [0,1]. Thus in practice we would require accurate tabulation of the cumulative integrals over this range and would also require an accurate interpolation formula to obtain values at points not tabulated.

If we used Newton's method to solve the normal equations, we could differentiate (11) again to obtain the Jacobian matrix; however this could be approximated using finite differences. We could also make use of the following information concerning the Jacobian matrix:

$$\left.\begin{aligned} \frac{\partial^2 L}{\partial x_i \partial x_j} &= \frac{\partial^2 L}{\partial x_j \partial x_i} \\ \frac{\partial^2 L}{\partial x_i \partial y_j} &= \frac{\partial^2 L}{\partial y_j \partial x_i} \\ \frac{\partial^2 L}{\partial y_i \partial y_j} &= \frac{\partial^2 L}{\partial y_j \partial y_i} \end{aligned}\right] \quad \ldots(13)$$

(Hence the Jacobian matrix will be symmetrical)

and from (11):

$$\left.\begin{aligned} \frac{\partial^2 L}{\partial x_i \partial x_j} &= 0 \quad \text{if } j < i-1 \text{ or } j > i+1 \\ \frac{\partial^2 L}{\partial x_i \partial y_j} &= 0 \quad \text{if } j < i-1 \text{ or } j > i+1 \\ \frac{\partial^2 L}{\partial y_i \partial y_j} &= 0 \quad \text{if } j < i-1 \text{ or } j > i+1 \quad . \end{aligned}\right] \quad \ldots(14)$$

The normal equations are, in general, very complex and if twenty segments were being fitted (the maximum number that we expect would be used) they would consist of 42 equations in 42 unknowns. In all but the simplest cases we would need a good initial approximation to their solution in order to avoid loss of accuracy, lack of convergence, only to a local minimum, and to avoid the use of excessive computer time.

Convergence to a local minimum rather than an absolute minimum is likely to occur if $f(x)$ has several turning points and points of inflexion. In Appendix B we give a detailed example, fitting two segments to a comparatively simple function, which illustrates the possibility of this behaviour.

As it turns out, the initial approximation which we will obtain (see Section 5) will normally be an excellent approximation by itself and there will be no need to proceed with the solution of the normal equations as outlined above. Moreover those cases where this initial approximation

will not be adequate will certainly involve complicated functions $f(x)$ and thus will be cases where the above method would not work without first in some way obtaining a good initial approximation.

4. CONTINUOUS APPROXIMATION WITH FIXED BREAK POINTS

If the n break points $x_1, x_2, \ldots, x_n$ are fixed, then the best least-squares approximation to $f(x)$ with these break points is obtained using Hilbert space theory (see for example Rudin (1966)).

Let H be the Hilbert space $L^2[0,1]$ consisting of all integrable functions g defined on the interval $[0,1]$ such that

$$\left.\begin{aligned} &\int_0^1 w(x)\,[g(x)]^2 dx < \infty \text{ , with inner product} \\ &(g_1, g_2) = \int_0^1 w(x) g_1(x) g_2(x)\,dx \end{aligned}\right] \qquad \ldots(15)$$

and norm

$$\| g \| = (g,g)^{\frac{1}{2}} \qquad \ldots(16)$$

which is the L_2 norm.

We now define the subspace M of H, which will consist of those elements of $A_c(n)$ which have break points $x_1, x_2, \ldots, x_n$, that is

$$M = \{g \in H : g \text{ is continuous and } g \text{ is linear between } x_i \text{ and } x_{i+1},\ i = 0,1,\ldots,n\} . \qquad \ldots(17)$$

M is of dimension n+2.

The function $f(x)$ that we wish to approximate is assumed to be continuous and hence is an element of H. Hilbert space theory asserts that there is a unique element $h \in M$ such that $\| f - h \|$ is a minimum, that is such that

$$g \in M,\ g \neq h \Rightarrow \| f - g \| > \| f - h \| .$$

This element h is the approximating function we seek.

Moreover the theory asserts that if $e_0, e_1, \ldots, e_{n+1}$ is an orthonormal basis for M then

$$h(x) = \sum_{i=0}^{n+1} (f, e_i)\, e_i(x) \qquad \ldots(18)$$

The following n+2 functions form a basis for M:

$$u_o(x) = \begin{cases} \dfrac{x_1 - x}{x_1} & \text{if } x_o \leq x \leq x_1 \\ o & \text{otherwise} \end{cases}$$

$$u_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{if } x_{i-1} \leq x < x_i \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & \text{if } x_i \leq x < x_{i+1} \quad \text{for } i = 1,2,\ldots n \\ o & \text{otherwise} \end{cases} \qquad \ldots(19)$$

$$u_{n+1}(x) = \begin{cases} \dfrac{x - x_n}{x_{n+1} - x_n} & \text{if } x_n \leq x \leq x_{n+1} \\ o & \text{otherwise} \end{cases}$$

From this basis we obtain an orthonormal basis $e_o, e_1, \ldots, e_{n+1}$ using the Gram-Schmidt process:

$$e_o(x) = \frac{u_o(x)}{\|u_o\|} \qquad \ldots(20)$$

and

$$e_i(x) = \frac{e_i'(x)}{\|e_i'\|} \qquad i = 1,2,\ldots,n+1 \qquad \ldots(21)$$

where

$$e_i'(x) = u_i(x) - \sum_{j=0}^{i-1} (u_i, e_j) e_j(x) \quad . \qquad \ldots(22)$$

The required approximation is now found from equation (18).

This method is a standard approach to least-squares approximation problems. It is discussed by Norkin (1962) (for the case where the break points are equally spaced) and is used by Vansteenkiste (1967).

5. DYNAMIC PROGRAMMING APPROACH

We consider briefly the dynamic programming approach to the line segment approximation problem firstly for the continuous case and then for the case where the segments are independent.

Dynamic programming (see for example Bellman (1957) is concerned with multistage decision problems which satisfy the following conditions:

(i) We have a system characterised at any stage by a finite set of parameters, the state variables.

(ii) At each stage of the process we have a choice of a number of decisions.

(iii) The effect of a decision is a transformation of the state variables.

(iv) The past history of the system is of no importance in determining future actions.

(v) The purpose of the process is to maximise (or, in our case, minimise) some function of the state variables.

We call this function the criterion function and call any allowable sequence of decisions a policy. An optimal policy is one which maximises (minimises) the value of the criterion function. The following intuitive principle now applies and is basic to dynamic programming.

> Principle of Optimality: An 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. ...(23)

This principle leads to efficient search algorithms (involving a recurrence relation) for finding an optimal policy in particular cases.

Consider now the continuous approximation problem as stated in Section 2. We outline a dynamic programming algorithm for its solution.

Let

$$h_i(u,v) = \min_{\substack{x_1,x_2,\ldots,x_{i-1} \\ y_0,y_1,\ldots,y_{i-1}}} \left(\sum_{j=0}^{i-1} \int_{x_j}^{x_{j+1}} w(x) \, [f(x) - \ell_j(x)]^2 \, dx \right) \qquad \ldots(24)$$

given that $x_i = u$, $y_i = v$.

(This is the minimum value of the square of the norm from an optimal approximation with i segments which is constrained to end at the point (u,v).)

Now we put

$$g(x_i,y_i,x_{i+1},y_{i+1}) = \int_{x_i}^{x_{i+1}} w(x)\ [f(x) - \ell_i(x)]^2 dx \qquad ...(25)$$

and, using (23), we obtain the recurrence relation

$$h_{N+1}(u,v) = \min_{x_N,y_N} \left[h_N(x_N,y_N) + g(x_N,y_N,u,v) \right]$$

$$N = 1,2,\ldots,n-1 \qquad ...(26)$$

where

$$h_1(u,v) = \min_{y_o} \left[g(o,y_o,u,v) \right] . \qquad ...(27)$$

The minimum value of the square of the L_2-norm is given by

$$\min L(x_1,\ldots,x_n,y_o,\ldots,y_{n+1}) = \min_{u,v} \left[h_n(u,v) + \min_{y_{n+1}} g(u,v,1,y_{n+1}) \right]$$

$$...(28)$$

it being noted that the optimal value of y_{n+1} is given in terms of u and v. This leads to a search algorithm where we search over a two-dimensional mesh. The number of combinations of points searched is of the order of $(n-1)m^2$, where m is the number of mesh points.

This method is not useful in practice. Since the segments must join mesh points, the mesh must be very fine - we would need of the order of 1000 mesh points (30 x 30 for example) before we could expect to obtain an acceptable approximation. With 100 mesh points an efficient computer program took between three and four minutes to carry out the search. Thus to search 1000 mesh points would take at least five hours. To obtain an approximation with 1 per cent accuracy in each of the x and y directions would require 50 x 50 = 2500 mesh points.

(Note the efficiency of the dynamic programming approach: with $n = 10$, $m = 100$ there are of the order of $\binom{m}{n} = \frac{m!}{(m-n)!n!} = O(10^{10})$ possible combinations of points, while the dynamic programming search considers only of the order of $(n-1)m^2 = O(10^5)$ combinations.)

We now consider the approximation problem with independent segments. This is almost identical to the above except that the functional equations will involve functions of one independent variable rather than two and the search algorithm will involve searching only over a one-dimensional mesh. We follow the paper by Bellman (1964).

For $0 \leq x_i < x_{i+1} \leq 1$ denote by $\hat{\ell}_i(x)$ the line segment which minimises

$$\int_{x_i}^{x_{i+1}} w(x)\ [f(x) - \ell_i(x)\ ]^2 dx \quad .$$

We wish to choose n break points $x_1, x_2, \ldots, x_n$ (where $0 = x_o < x_1 < \ldots < x_n < x_{n+1} = 1$) to minimise

$$L(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} w(x)\ [f(x) - \hat{\ell}_i(x)\ ]^2 dx \quad . \qquad \ldots(29)$$

Note that $\hat{\ell}_i(x)$ is completely specified in terms of x_i, x_{i+1} and may be found using calculus.

Let

$$h_N(b) = \min_{x_1 \ldots, x_{N-1}} L(x_1, x_2, \ldots x_{N-1}) \text{ given } x_N = b \quad , \qquad \ldots(30)$$

$$h_1(b) = \int_{x_o}^{b} w(x)\ [f(x) - \hat{\ell}_o(x)\ ]^2 dx \qquad \ldots(31)$$

and we may obtain the recurrence relation

$$h_{N+1}(b) = \min_{x_N} \left[h_N(x_N) + \int_{x_N}^{b} w(x)\ [f(x) - \hat{\ell}_N(x)\ ]^2 dx \right] \qquad \ldots(32)$$

$$N = 1, 2, \ldots, n$$

with

$$\min_{x_1, \ldots, x_n} L(x_1, x_2, \ldots, x_n) = h_{n+1}(1) \quad . \qquad \ldots(33)$$

Equations (32) and (33) may also be obtained using the principle of optimality. They are particularly simple because the square of the norm, being an integral, splits up into a sum of the squares of norms over the separate intervals (x_o, x_1), $(x_1, x_2), \ldots, (x_n, x_{n+1})$.

Equations (31), (32) and (33) constitute an efficient and practical search algorithm for finding an optimal independent approximation. We search over a one-dimensional mesh between $x = 0$ and $x = 1$. Again the number of combinations searched is of the order of $(n-1)m^2$ where m is the number of mesh points, but now the accuracy obtained using 100 mesh points is quite acceptable.

6. PRACTICAL APPROACH

None of the methods of solution considered in the previous sections is really satisfactory. The dynamic programming method requires too many mesh points, and hence too much computation, to obtain sufficient accuracy. In solving the normal equations we may have troubles with lack of convergence or convergence to a local minimum. It is clear that it is best if we first obtain a good approximate solution.

To obtain this we use the dynamic programming search to solve the simpler approximation problem with independent segments (Bellman 1964).

This initial step has the following advantages:

(i) Intuitively, the break points for this case should not differ very much from those for the continuous case. In both we are finding an optimal splitting of $f(x)$ into $n+1$ 'linear' segments.

(ii) We are able to find accurately the best approximation with independent segments even when the function $f(x)$ is very complicated (see examples in Appendix D).

(iii) We obtain a lower limit for the value of the least-squares norm that may be obtained with a continuous approximation.

(iv) If the discrepancies at the break points are small then we know that there will be little difference between this solution and the best continuous approximation. Hence in particular cases we will often know that our approximate solution is a good one. This case will arise especially when $f(x)$ is particularly simple (especially if $f''(x) \neq 0$, $0 \leq x \leq 1$) or when the number of segments is large.

Having found the best approximation with independent segments we take the break points found and use Hilbert space theory to find the best continuous approximation with these break points. This now gives us our initial approximate solution to the continuous line segment approximation problem.

As pointed out in Section 3, in most practical cases this approximation will be adequate and we need proceed no further. However, if the exact solution is required we could proceed to solve the normal equations using Newton's method (Section 3) and the above initial approximation.

A computer program has been written in FORTRAN to produce the initial approximate solution as outlined above. Instructions for using the

program are contained in Appendix C and results from test runs are shown in Figures 2 - 5.

7. CONCLUSION

The approach presented in this report provides a useful, accurate and widely applicable method for the continuous line segment approximation of functions.

Unlike previous methods, this reliably approximates very complicated functions and gives an indication of the possible deviation from an optimal approximation. We obtain a good lower bound for the value of the least-squares error of an optimal approximation and hence in most cases we will know that the approximation produced is very close to optimal. The reliability is due to the fact that the preliminary approximation with independent segments is always optimal (to within the accuracy of the search mesh).

The FORTRAN computer program using this method demonstrates the above features and the computational feasibility and economy of the approach. It will fit up to twenty segments to a continuous function with a maximum run time (on an IBM 360/50I) of three minutes. In practice the only inaccuracies introduced are due to the use of 1000 point Simpson's rule integration, but this is very accurate, even with functions which are not smooth. Thus, the program will produce good approximations to piecewise differentiable functions, although with less accuracy and reliability than for smooth functions.

8. ACKNOWLEDGEMENTS

The author gratefully acknowledges the assistance of Mr. E. Corran (who first suggested the topic), Mr P. Gaal (who carried out the initial literature search) and Mr J. Barry.

9. REFERENCES

Bellman, R. (1957). - Dynamic Programming, Princeton University Press, Princeton, N.J.

Bellman, R. (1964). - 'On the approximation of curves by line segments using dynamic programming', Comm. A.C.M. Vol. 4, No. 6, p. 284.

Gluss, B. (1964). - 'An alternative method for continuous line segment curve-fitting', Inform. Control, 7, pp. 200-206.

Hamer, H. (1956). - 'Optimum linear-segment function generation', Trans. AIEE Pt. 1, Vol. 75, pp. 518-520.

Norkin, K.B. (1962). - 'Automatic adjustment of a universal function generator with a piecewise-linear approximation', Automation and Remote Control, 23, pp. 1263-1270.

Phillips, G.M. (1968). - 'Algorithms for piecewise straight line approximations', Computer Journal, Vol. 11, pp. 211-212.

Ream, N. (1959). - 'Approximation errors in diode function generators', J. Electronics and Control, Vol. 7, pp. 83-96.

Rudin, W. (1966). - Real and Complex Analysis, McGraw-Hill, N.Y.

Stone, H. (1961). - 'Approximation of curves by line segments', Math. Comp., Vol. 15, No. 73, pp. 40-47.

Vansteenkiste, Gh.J. (1967). - 'Optimal generation of arbitrary functions', Annales de l'Association pour le Calcul analogique, No. 3, pp. 142-146.

APPENDIX A

The Method of Gluss (1964)

Instead of minimising

$$\sum_{i=1}^{n+1} \int_{x_{i-1}}^{x_i} [f(x)-a_i x-b_i]^2 \, dx$$

Gluss minimises

$$\sum_{i=1}^{n+1} \int_{x_{i-1}}^{x_i} [f'(x)-a_i]^2 \, dx \quad .$$

That is, he finds the best least-squares approximation to the first derivative $f'(x)$ by a step function

$$g(x) = a_i \text{ for } x_{i-1} < x < x_i \qquad i=1,2,\ldots,n+1 \; .$$

This problem is similar to the independent approximation problem and can be solved using a simple dynamic programming search. Gluss now obtains a class of continuous line-segment approximations to $f(x)$, having the break points found above, by integrating $g(x)$

$$G(x;c) = \int_{x_o}^{x} g(t)\,dt$$

$$= a_i x - a_i x_{i-1} + c \quad \text{for } x_{i-1} < x < x_i$$

$$i=1,2,\ldots,n+1 \quad .$$

He now uses elementary calculus to minimise

$$\int_{x_o}^{x_{n+1}} [f(x)-G(x;c)]^2 \, dx$$

with respect to c. Intuitively, this should be a reasonable continuous line-segment approximation to $f(x)$, especially if the graph of $f(x)$ is fairly simple.

Another approach which could be taken is to use Gluss' method to find the break points $x_1, x_2, \ldots x_n$ and then use Hilbert space theory to fit the best approximation with these break points.

Gluss' method will be acceptable in many simple practical cases, but it does not have the advantages of the present method. It will be of less use if the graph of $f(x)$ is 'spiky' - for example, too much weight would be given to a very small peak where the gradient is very large.

APPENDIX B

As an example of a local minimum in the norm function

$$L_2(f-h) = L(x_1,\ldots,x_n,y_o,y_1,\ldots,y_{n+1}),$$

Consider the function

$$f(x) = \exp[-10(x-0.5)^2] + \tfrac{1}{2}\exp[-10(x+0.5)^2].$$

When fitting this with two segments, the norm function is a function of four variables

$$L_2(f-h) = L(x_1,y_o,y_1,y_2)$$

and the normal equations will consist of four equations in four unknowns We plot the function

$$F(x_1) = \min_{y_o,y_1,y_2} L(x_1,y_o,y_1,y_2)$$

to give a function of one variable. That is, for each choice of the break point x_1 we fit the best two-segment approximation $h(x)$ and compute the least-squares error $L_2(f-h)$.

This function $F(x_1)$ has two local minima, one of which is also the absolute minimum giving the best choice of x_1. If $F(x_1)$ has a minimum when $x_1 = x_1^1$ and the corresponding values of y_o,y_1,y_2 are y_o^1,y_1^1,y_2^1, then $L(x_1,y_o,y_1,y_2)$ will have a local minimum in the neighbourhood of $(x_1^1,y_o^1,y_1^1,y_2^1)$. Thus the function of four variables has at least two local minima, one of which is the absolute minimum of the function. Figure 1 shows the function $F(x_1)$ and the two approximations - the first from the local minimum of $F(x_1)$ and the second from the absolute minimum. It is clear that in this case the use of Newton's method to solve the normal equations may easily lead to an approximation which is not optimal.

APPENDIX C

LISA SUBROUTINE PACKAGE

This appendix describes the use of the collection of FORTRAN subroutines (referred to as the LISA subroutine package) which produces the approximate solution to the continuous line segment approximation problem as outlined in Section 6.

(i) General:

The LISA package was written for use on an IBM 360/50 computer with 512K bytes of core storage. All floating point variables and constants are double precision.

(ii) Calling Sequence:

To the user the package appears to be a single subroutine (subroutine LISA) which has no argument list and is called as follows:

```
CALL LISA
```

from a mainline program.

(iii) Input:

Input is via formatted read statements and one NAMELIST read statement contained in subroutine LISA. The entire input is handled by the following statements:

```
      IMPLICIT REAL*8(A-H,Ø-Z)
      DIMENSIØN HEAD(20),XFN(200),YFN(200)
      READ(1,1006)HEAD
 1006 FØRMAT(10A8)
      NAMELIST/NLIST/NSEG,XMIN,XMAX,NPTS,IPLOT,IBKPTS
      IPLØT=1
      IBKPTS=0
      READ(1,NLIST)
      IF(NPTS.GT.0)READ(1,1005)(XFN(I),YFN(I),I=1,NPTS)
 1005 FØRMAT(2D20.10)
      NS1=NSEG+1
      IF(IBKPTS.NE.1) GØ TØ 30
      READ(1,1002)(XB(I),I=2,NSEG)
 1002 FØRMAT(10D8.4)
      XB(1)=XMIN
      XB(NS1)=XMAX
```

Appendix C (cont.)

```
      IXB(1)=1
      IXB(NS1)=101
      DØ 30 I=2,NSEG
      X=(XB(I)-XMIN)*1.D2/(XMAX-XMIN)+0.5D0
      IXB(I)=IDINT(X)+1
   30 CØNTINUE
```

The input variables are as follows:

HEAD: This specifies 160 characters for a two-line heading. (The first two data cards will appear as a heading at the top of each page of output.)

NSEG: The number of line segments required in the approximation ($2 \leq NSEG \leq 20$).

XMIN,XMAX: These specify the range of X ($XMIN \leq X \leq XMAX$) over which it is required to approximate the function f(x).

NPTS: If NPTS=0 the program assumes that the function f(x) is specified in a function subprogram, REAL FUNCTIØN FN*8(X).

If NPTS > 0 the program assumes that the function is tabulated at NPTS points and reads the next NPTS data cards as the tabulation of the function.

NPTS must lie in the range $0 \leq NPTS \leq 200$ and has a default value of 0.

IPLØT: Specifies whether or not plots of the results are required. If IPLØT=0, no plots are produced, otherwise, results are plotted. IPLØT has a default value of 1. If plots are required the necessary JCL cards must accompany the job (see examples below).

IBKPTS: Specifies whether or not break points are to be read in.

If IBKPTS=1 the program assumes that break points are supplied and that an approximation is required using Hilbert space theory with these given break points.

Appendix C (cont.)

If NPTS > 0 the package reads in the next NPTS data cards as the tabulations of the function FN(X). These cards must be punched with pairs (XFN(I), YFN(I)) of real numbers specifying the X-value and corresponding function value using the format (2D20.10). The cards must be ordered in increasing values of X, and the values must satisfy

XFN(1)≦XMIN<XMAX≦XFN(NPTS)

(iv) Function Subprogram FN(X):

If NPTS=0 a function subprogram

```
REAL FUNCTIØN FN*8(X)
```

must be supplied to specify f(x). The name must be FN and the subprogram must be in double precision. For example to specify $f(x) = e^{-2x}/x$ we could have the following:

```
REAL FUNCTIØN FN*8(X)
IMPLICIT REAL*8(A-H,Ø-Z)
FN=(DEXP(-2.DO*X))/X
RETURN
END
```

Note that the double precision form of library functions must be used (e.g. DEXP rather than EXP).

If NPTS > 0 and the function f(x) is tabulated, the LISA package fits a cubic spline function to the points and uses this as the function FN(X) throughout the calculations.

(v) Weighting Function:

A weighting function may also be specified in a supplied function subprogram. This must be a continuous function defined and positive in the range XMIN ≦ X ≦ XMAX. The function must have the name WT, and will be specified in a double precision subprogram as for FN(X) above. For example, to specify $w(x) = \frac{1}{|f(x)|}$ we might have:

```
REAL FUNCTIØN WT*8(X)
IMPLICITY REAL*8(A-H,Ø-Z)
F=FN(X)
AF=DABS(F)
IF(AF.LE.1.D-2)AF=1.D-2
WT=1.DO/AF
RETURN
END
```

Appendix C (cont.)

If no function WT(X) is supplied then it is assumed that WT(X) =1.D0 for all values of X.

(vi) Output:

Firstly the parameters and tabulation of the function FN(X) (if NPTS > 0) are printed out. Then follows six pages number 1-6 which present the results of three stages of the program.

Pages 1 and 2 present the results of the dynamic programming search with segments not necessarily meeting. Results are scaled to fit in the square $-1 \leq X \leq 1$, $-1 \leq Y \leq 1$ and are printed out followed by the corresponding scale factors. There follows the same results scaled by standard scale factors (to allow comparison of the three sets of output).

The least-squares error is minimised in this and the results are the best to within the error of the basic search mesh. These results supply a lower bound for the least-squares error that may be achieved with the segments meeting and the discrepancies at the break points give an indication of the accuracy of the final result (which is on pages 5 and 6 of the output).

The break points found in the first part of the program are kept in the next two stages. Pages 3 and 4 merely average the results of the first stage (i.e. average the y-values at the break points) the results being presented in the same format as Pages 1 and 2.

Pages 5 and 6 present the results of fitting the best least-squares continuous approximation with the above fixed break points using Hilbert space theory. Again format is as above.

The results of Pages 1 and 2 and Pages 5 and 6 are plotted on two plots, labelled PLØT 1 and PLØT 2 respectively (if IPLOT=1).

(vii) Scaling:

The function values are scaled in the range $XMIN \leq X \leq XMAX$ so that $\max|f(x)| = 1$. The range of X is then scaled so that $\max(|XMIN|, |XMAX|) = 1$. Hence the scaled function and scaled range fit in the region $-1 \leq x \leq 1$, $-1 \leq y \leq 1$. The scale factors used here are the standard scale factors in all computation and are used when calculating the standardised error criteria (see (iii)).

Appendix C (cont.)

Once an approximation has been found it is scaled down as above if the maximum absolute value of the approximating function is greater than one. The composite scaling factor is then used when tabulating and plotting results so that all values produced are between -1 and +1.

The weighting function is scaled so that the integral over the scaled range is XMAX-XMIN, i.e. if XMIN, XMAX are scaled values, then after scaling

$$\int_{XMIN}^{XMAX} w(x)\,dx = XMAX-XMIN$$

(viii) Sample Card Decks:

(a) 10 segments, XMIN = -1.D0, XMAX = 1.D0, function subprogram supplied, weighting function specified and plots required.

```
// EXEC BUFFPRØG,PRG=AEPLØT                (required if plotting)
// EXEC FØRTGCLG,PARM.LKED=MAP
// FORT.SYSIN DD *
      ....
      ....
      CALL LISA                           (Main program)
      ....
      ....
      END
      REAL FUNCTIØN FN*8(X)
      ....
      ....                                (This subprogram must
      ....                                 precede LISA deck)
      END
      REAL FUNCTIØN WT*8(X)
      ....                                (This subprogram must
      ....                                 precede LISA deck)
      END
```

Appendix C (cont.)

```
SUBROUTINE LISA
....                                        (LISA card deck)
....
END
/*
//GØ.AEPLØT DD SYSOUT=C                     (Must be included if
                                             plotting is required)
//GØ.SYSIN DD *
Heading .....................................
Heading .....................................
bϵNLIST NSEG=10,XMIN=-1.D0,XMAX=1.D0,ϵEND      (b = "blank")
/*
```

(b) 10 segments, XMIN= -1, XMAX = 1, function tabulated and plots required; e.g. NPTS = 51.

This is as above except that there is no user-supplied subprogram REAL FUNCTIØN FN*8(X) included at the front of the deck, and the input must be as follows:

```
//GØ.SYSIN DD *
Heading .....................................
Heading .....................................
bϵNLIST NSEG=10,XMIN=-1.D0,XMAX=1.D0,NPTS=51,ϵEND
```

```
x(1)    f(1)
x(2)    f(2)
 .       .
 .       .
 .       .
x(51)   f(51)
```

51 cards containing tabulated function, format (2D20.10), $x(1) < x(2) < \ldots < x(51)$

```
/*
```

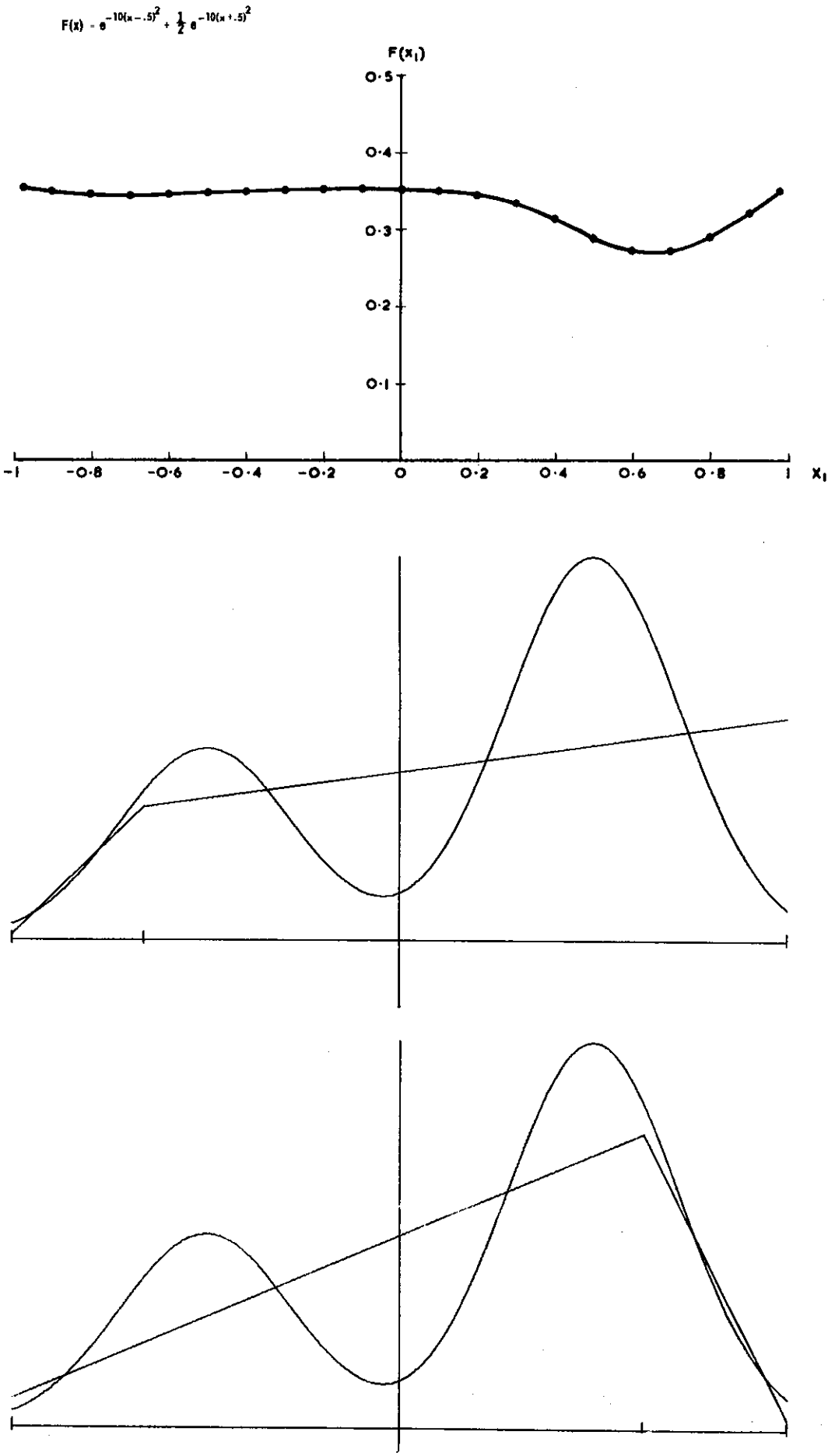
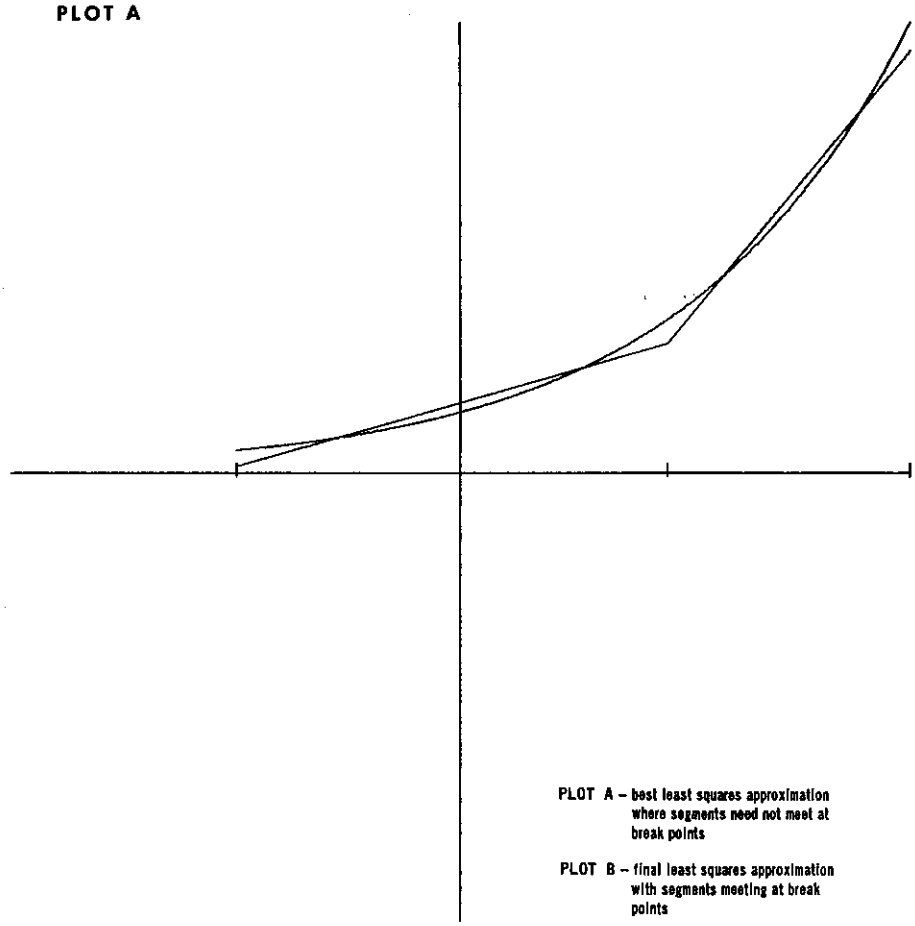


FIGURE 1 EXAMPLE OF LOCAL MINIMUM IN GRAPH OF MINIMUM NORM, AND RESULTING APPROXIMATIONS

PLOT A



PLOT A – best least squares approximation where segments need not meet at break points

PLOT B – final least squares approximation with segments meeting at break points

PLOT B

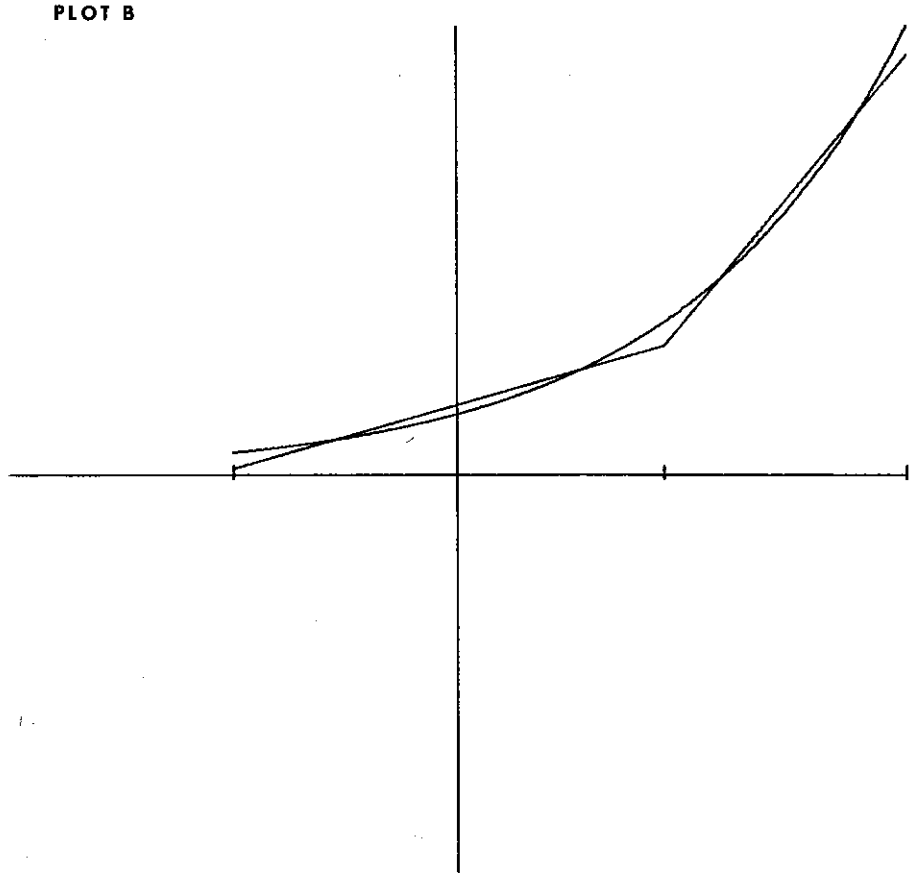
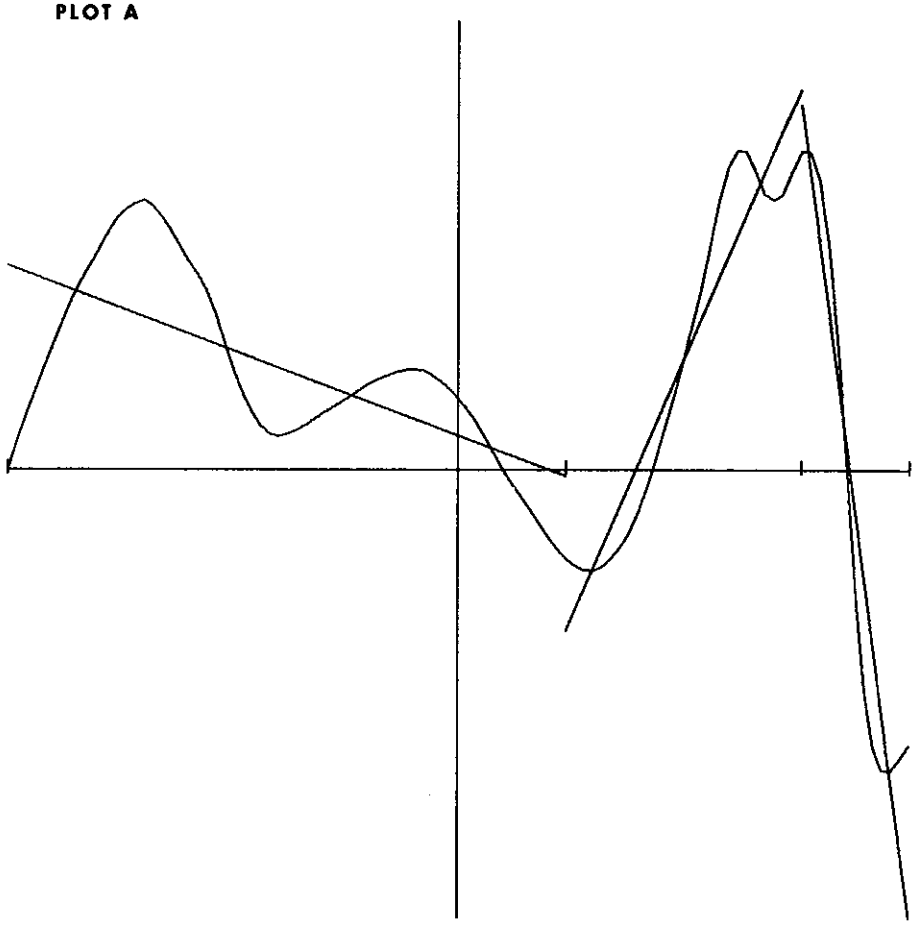


FIGURE 2 FN(X) = EXP(X)

NSEG = 2, XMIN = -1.D0,XMAX=2.D0,NPTS=0,IPLOT =1

EXECUTION TIME = 33 SECONDS

PLOT A



PLOT A – best least squares approximation where segments need not meet at break points

PLOT B – final least squares approximation with segments meeting at break points

PLOT B

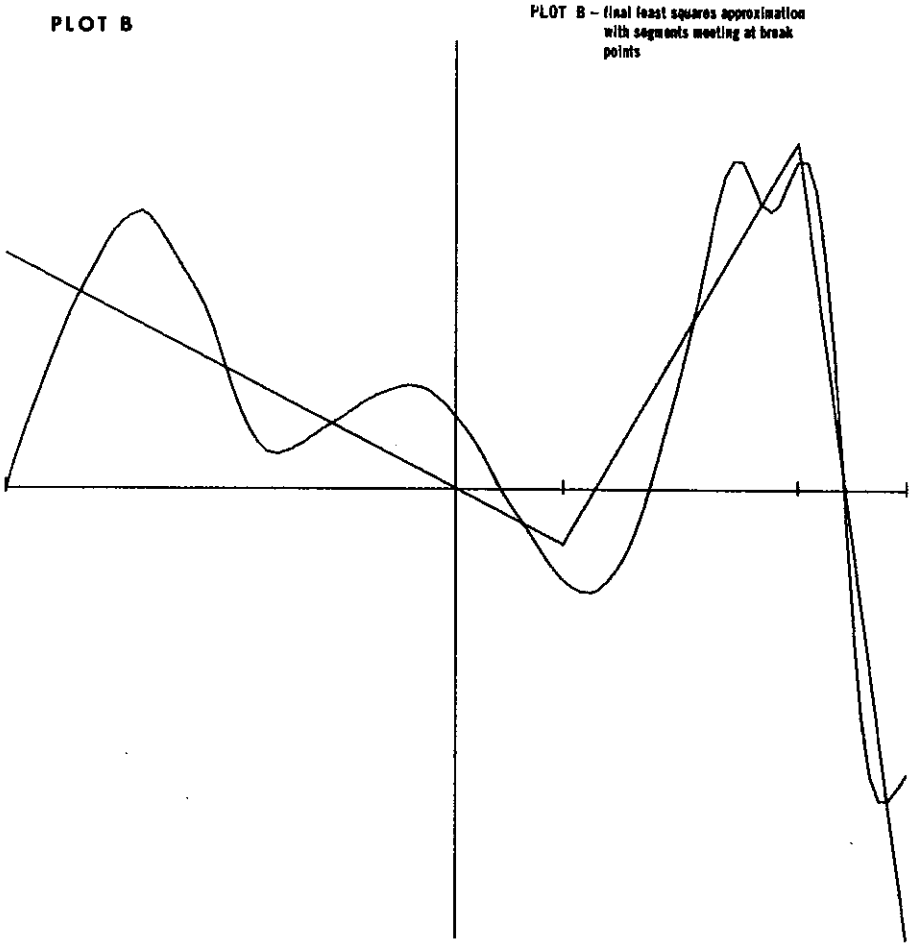
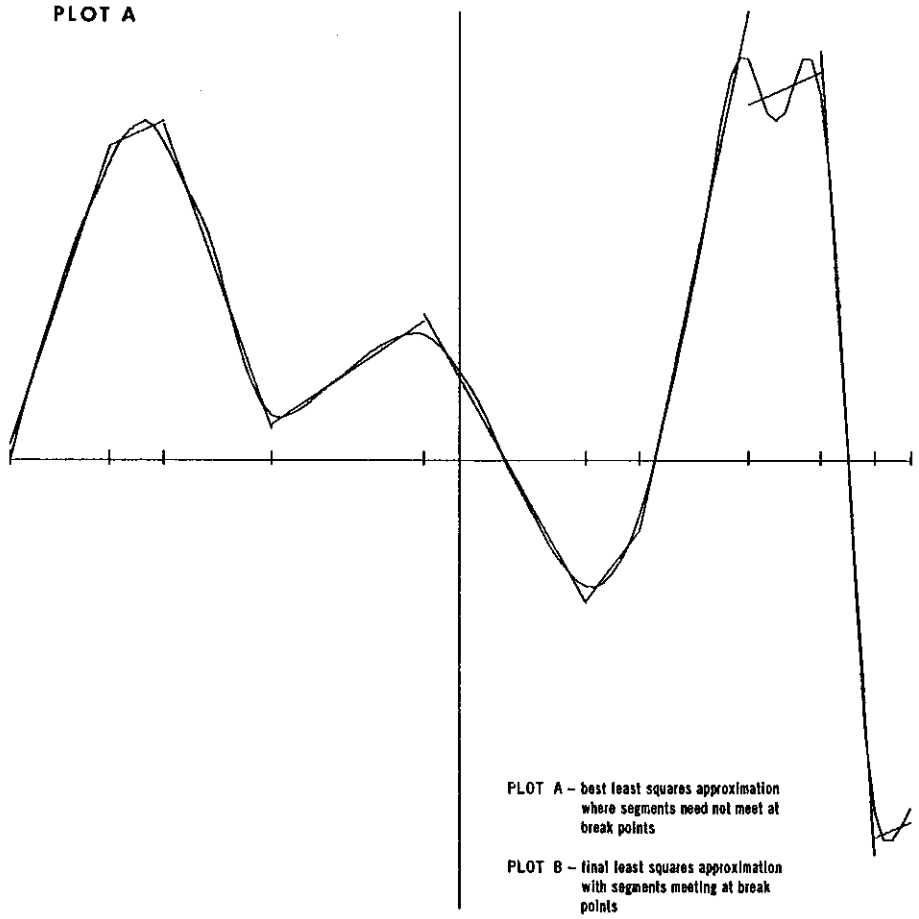


FIGURE 3 TABULATED FUNCTION

NSEG-3, XMIN- –1.D0, XMAX-1.D0, NPTS-51, IPLOT-1

EXECUTION TIME - 51 SECONDS



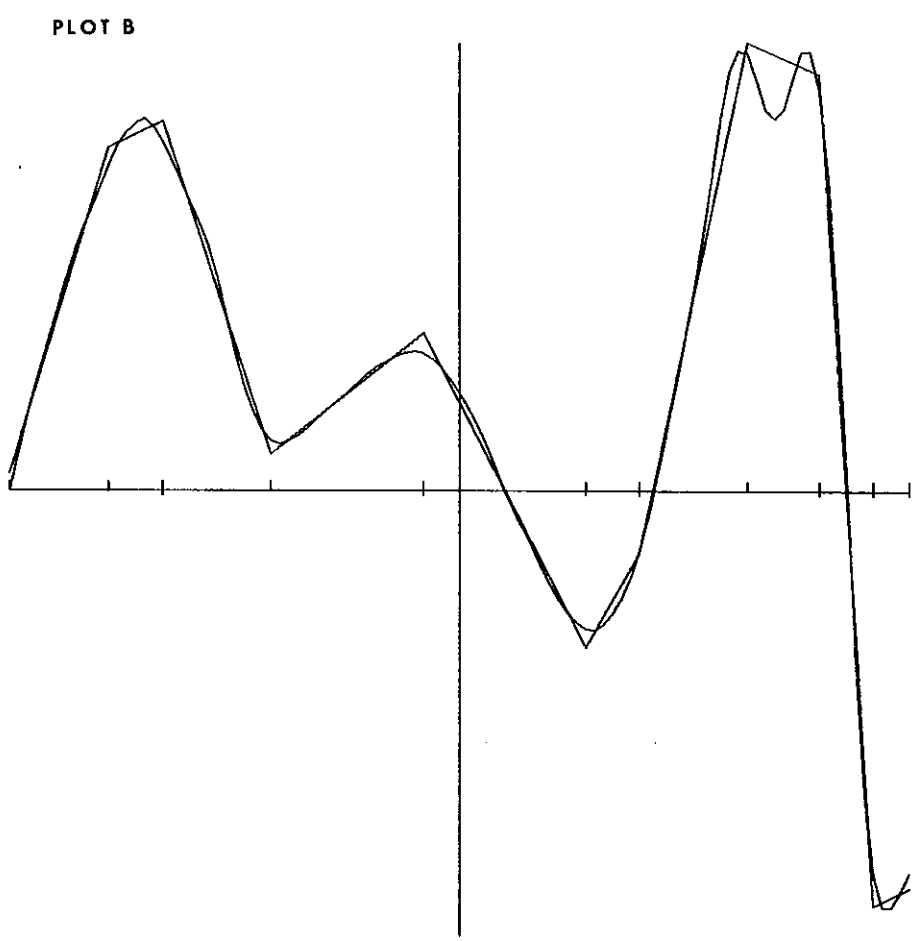
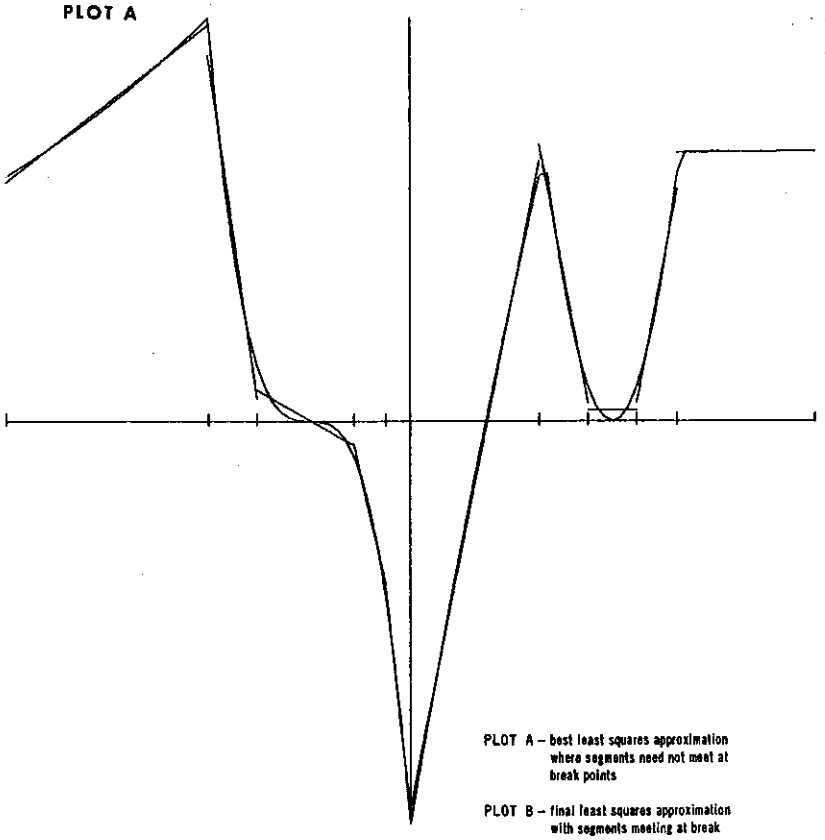


FIGURE 4 TABULATED FUNCTION

NSEG=10, IMIN= –1.D0, IMAX=1.D0, NPTS=51, IPLØT=1

PLOT A



PLOT A – best least squares approximation where segments need not meet at break points

PLOT B – final least squares approximation with segments meeting at break points

PLOT B

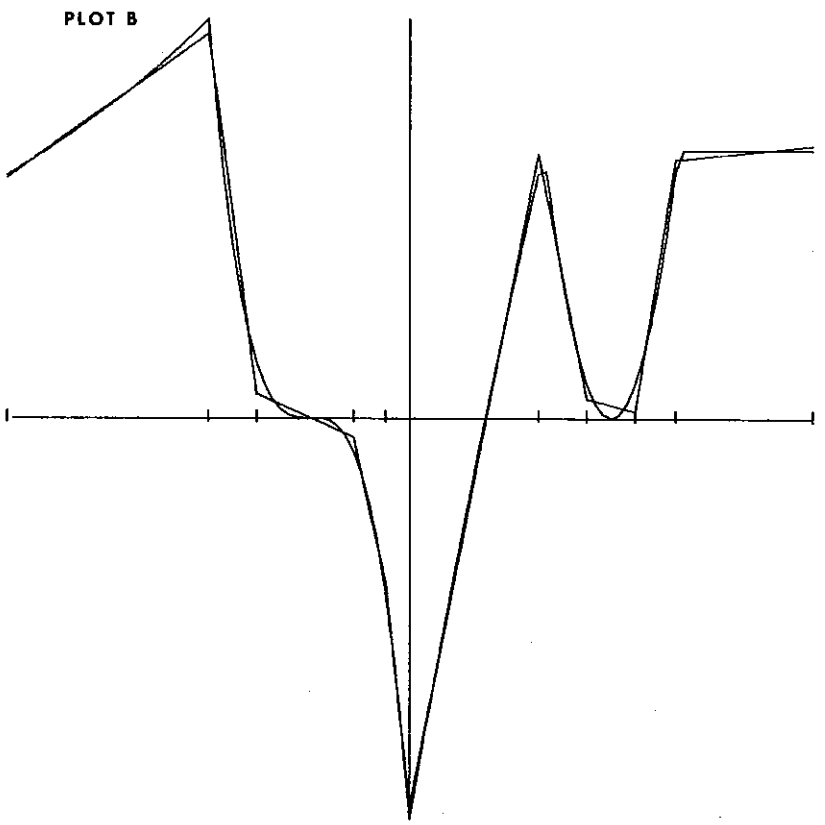


FIGURE 5

$$FN(X) = \begin{cases} EXP(X + 0.5) & -1 \le x \le -0.5 \\ -6x^3 - 48x^2 - 12x - 1 & -0.5 \le x \le 0 \\ \dfrac{5 \, LN(x + 1)}{3 \times 0.2877} - 1 & 0 < x < 1/3 \\ 24x^2 - 24x + 6 & 1/3 \le x \le 2/3 \\ 2/3 & 2/3 \le x \le 1 \end{cases}$$

NSEG = 10, XMIN = -1.00, XMAX = 1.00, NPTS = 0, IPLOT = 1

EXECUTION TIME = 113 SECONDS