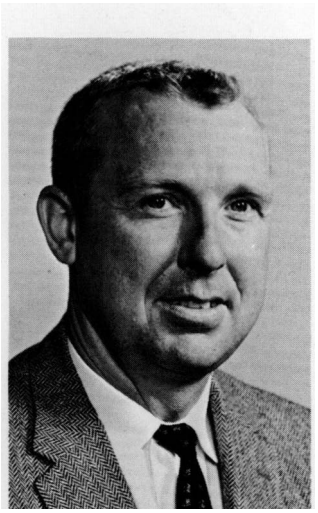
Computer generation of arbitrary functions



by L. E. FOGARTY
University of Michigan
Ann Arbor, Michigan

Preparation of this report was supported under Contract NASr 54(06).

L. E. FOGARTY was born in Minnesota in 1920, and grew up near Bozeman, Montana where he graduated from Montana State College in 1940 with a degree in Electrical Engineering. That same year he joined NACA (the forerunner of NASA) and worked as a wind-tunnel engineer at Langley Field, Virginia until 1944, when he joined the Navy. The Navy also had him working as a wind-tunnel engineer, at the David Taylor Model Basin in Washington, D.C., where he remained after his discharge from the Navy until 1947.

In 1950 he received a PhD from Cornell University in Aeronautical Engineering, and after a short stay at North American Aviation, joined Link Aviation in Binghamton, New York. At Link he was concerned with flight simulation as Supervisor of the Flight Mechanics Department and later as Chief Engineer.

In 1959 he joined the University of Michigan as lecturer in Aerospace Engineering, then went with Applied Dynamics, where he served as a vice-president in '63 and '64. In 1965 he returned to the University of Michigan as a lecturer and research associate.

Dr. Fogarty continues to serve as a consultant to Link and to Applied Dynamics. His major interests are in flight simulation, space vehicle mechanics, and analog computer applications.

He is an Associate Fellow of the AIAA and is a member of SCi.

AUTHOR'S NOTE:

This presentation is intended to be mainly tutorial, rather than a report on original research — I don't claim priority on any of it except possibly the use of multi-dimensional elementary functions (the hyperboloids shown in Figure 13). I'm not even sure that someone else hasn't used them before, although I've never seen them. The series expansion technique described in the last part of the paper is old. The reference by Silverburg and Pike is the earliest description I have found, but Clymer tells me that he used it years ago. Probably this is another instance where proper credit can't be given because of antiquity.

INTRODUCTION

One of the pressing problems of computer technology is that of generating an arbitrary specified output as a function of given input variables. Examples of such problems are:

1. Given an input variable which represents altitude, h, generate an output which represents atmospheric density, $\rho(h)$.
2. Given input variables α, M, which represent angle of attack and Mach number, generate an output which represents drag coefficient C_D, as a function of α and M.

In these examples, ρ and C_D, although known functions of the input variables h, α, and M, are not specified in terms of mathematical formulas, hence are "arbitrary" functions of the input variables. Such functions seldom are given as continuous functions of the input variables, but rather are specified at a number of discrete values of them. They must be estimated by means of an interpolation formula at intermediate values.

The problem considered here is:

Given a data matrix which represents the values of a function $F(x_i)$ for given values of n input variables x_i, generate a continuous output, $F(x_i)$, using prescribed interpolation rules for determining F at intermediate values of the x_i.

Because of the generality of the problem, it is necessary to eliminate most of the many possible schemes from consideration. General methods depend on series expansion in terms of prescribed continuous functions (reference 1). Suitable expansion series are Fourier series, power series, etc., each of which has advantages in particular cases. In the general case, when the output functional form is not known ahead of time, one cannot always select the best type of series.

THEORY OF THE ONE-DIMENSIONAL INTERPOLATION SERIES (reference 2)

It is desired to represent an arbitrary function by superposing a series of elementary functions which can be generated easily on a computer. For analysis purposes, we will start with the elementary function known as the unit impulse, or Dirac delta function. Physically we can think of the unit impulse, $\delta_0(x)$, as a thin pulse of unit area and width approaching zero located "at" $x = 0$.

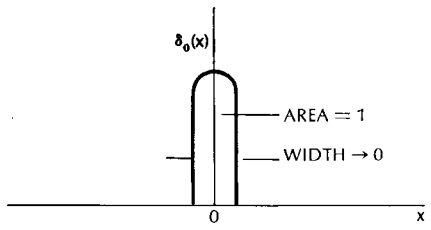


Figure 1 – Unit impulse, $\delta_0(x)$

Formally, we define the unit impulse from:

$$\delta_0(\xi - x) = 0 \qquad \xi \neq x \tag{1}$$

$$\lim_{\epsilon \to 0} \int_{x-\epsilon}^{x+\epsilon} \delta_0(\xi - x)d\xi = 1 \tag{2}$$

Additional higher order elementary functions are obtained by repeated integrations of the unit impulse. For example:

$$\int_{a<0}^{x} \delta_0(\xi)d\xi = \delta_1(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x > 0 \end{cases} \tag{3}$$

$\delta_1(x)$ is the unit step function located at $x = 0$

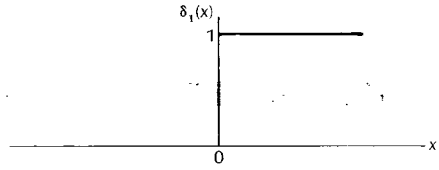


Figure 2 – Unit step, $\delta_1(x)$

Also the integral of the unit step function, $\delta_1(x)$, is the unit ramp function, $\delta_2(x)$; the integral of the unit ramp is a "unit parabola," $\delta_3(x)$, etc.

$$\int_{a<0}^{x} \delta_1(\xi)d\xi = \delta_2(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases} \tag{4}$$

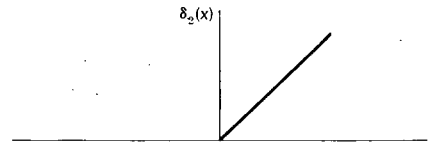


Figure 3 – Unit ramp, $\delta_2(x)$

$$\int_{a<0}^{x} \delta_2(\xi)d\xi = \delta_3(x) = \begin{cases} 0 \text{ for } x < 0 \\ \frac{x^2}{2} \text{ for } x \geq 0 \end{cases} \tag{5}$$

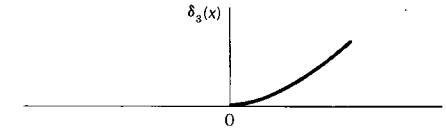


Figure 4 – Unit parabola, $\delta_3(x)$

From the properties of the unit impulse defined by equations (1) and (2) it is readily shown that, for $a < x < b$,

$$f(x) = \int_a^b f(\xi)\delta_0(\xi - x)\,d\xi = -\int_a^b f(\xi)\delta_0(x - \xi)\,d\xi \tag{6}$$

Since we wish to approximate $f(x)$ in terms of quantities evaluated at the beginning of the interval, we choose the second form of equation (6) and integrate by parts to obtain:

$$f(x) = f(a) + \int_a^b f'(\xi)\delta_1(x - \xi)\,d\xi \tag{7}$$

Representing the integral in (7) by a finite sum, with the interval from a to b divided, for simplicity, into n equal increments of length h:

$$\begin{aligned} f(x) \approx\ & f(a) + hf'(a)\delta_1(x - a - h) \\ & + hf'(a + h)\delta_1(x - a - 2h) + \ldots + \\ & + hf'(a + nh - h)\delta_1(x - b) \end{aligned} \tag{8}$$

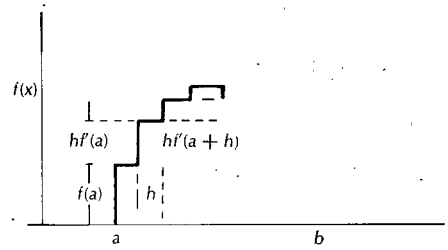


Figure 5 – Zero-order interpolation

Equation (8) is a series representation of $f(x)$ which corresponds to the assumption that $f(x)$ is constant in each subinterval of length h. Series (8) will be named the "zero-order interpolation series."

In equation (7) we integrate by parts again to obtain:

$$f(x) = f(a) + f'(a)(x - a) + \int_a^b f''(\xi)\delta_2(x - \xi)d\xi \tag{9}$$

Representing the integral in (9) by a finite sum as before,

$$f(x) \approx f(a) + f'(a)(x-a) + hf''(a)\delta_2(x-a-h) + hf''(a+h)\delta_2(x-a-2h) + \ldots + hf''(a+nh-h)\delta_2(x-b) \quad (10)$$

The functions δ_2 are unit ramps, hence the individual terms of the series (10) represent slope increments of magnitude hf''.

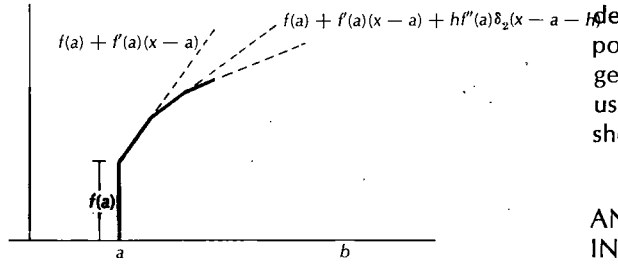


Figure 6 – First-order interpolation series representation of $f(x)$

The series (10) corresponds to the assumption that the slope of $f(x)$ is constant in each subinterval, h. Linear interpolation is performed between values of $f(x)$ specified at the ends of the intervals, $x = a + kh$, $k = 0, 1, \ldots n$. Series (10) will be named the "first-order interpolation series."

It is obvious that we can repeat, as many times as required, the operations of integrating by parts and representing the resulting integral by a finite sum. In each case one assumes the existence of all derivatives of $f(x)$ on (a, b) through the order one higher than the order of the resulting interpolation series. The mth order series is:

$$f(x) \approx f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2!} + \ldots + f^m(a)\frac{(x-a)^m}{m!} + hf^{m+1}(a)\delta_{m+1}(x-a-h) + \ldots + hf^{m+1}(a+nh-h)\delta_{m+1}(x-b) \quad (11)$$

The mth order interpolation series representation of $f(x)$ consists of the first $(m+1)$ terms of the Taylor series together with an n-term approximation of the integral remainder of the Taylor series. Note that, in accordance with the mean value theorem, a better representation of $f(x)$ is obtained when the derivative terms, hf^{m+1}, are evaluated at an intermediate point on each interval, rather than at the beginning as shown in equation (11).

The zero-order series (8) approximates a digital function generator without interpolation between data points. The first-order series (10) which performs linear interpolation between data points is the one which best represents the common analog diode function generator.

The second-order series,

$$f(x) \approx f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2} + hf'''(a)\delta_3(x-a-h) + \ldots \quad (12)$$

provides parabolic interpolation between data points. This series is not commonly used but it would seem to have good promise for representing physical functions which are well-behaved, that is, functions for which the higher-order derivatives are small and the Taylor series converges rapidly.

The utility of the analytical representation of $f(x)$, as in series (11), lies in the ready availability of error estimates for each series and the opportunity for analytical manipulation.

Note that equal increments of x were used in series (11) only for simplicity. The form of the series is unaltered by the use of unequal increments, and the size of the $(m+1)^{th}$ derivative indicates the proper spacing of the "break points" for the elementary functions. For example, to generate a parabola using the functions δ_2, as in the widely used diode quarter-square multiplier, the break points should be spaced evenly since f'' is constant in this case.

ANALOG COMPUTER GENERATION OF THE INTERPOLATION SERIES

To reduce confusion, we define the m^{th}-order interpolation function:

$$g^m{}_i(x) \equiv \delta_{m+1}(x - x_i)$$

For example,

$$g^1{}_i(x) = \delta_2(x - x_i)$$

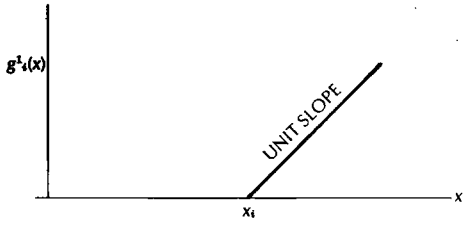


Figure 7 – First-order interpolation function, $g^1{}_i(x)$

Then the first-order series is written, for $0 \leq x \leq x_n$: $(x_0 = 0)$

$$f(x) \approx f(0) + a_0 g^1{}_0(x) + a_1 g^1{}_1(x) + + \cdots + a_{n-1} g^1{}_{n-1}(x) \quad (13)$$

The second-order series is

$$f(x) \approx f(0) + a_0 x + b_0 g^2{}_0(x) + b_1 g^2{}_1(x) + \cdots + b_{n-1} g^2{}_{n-1}(x) \quad (14)$$

An analog computer circuit for generating several functions of x, using the first-order interpolation series, is shown in figure 8. Once the first-order interpolation functions, $g^1{}_i$, are generated, then any number of arbitrary functions of x can be generated by summing the $g^1{}_i$ multiplied by suitable constants. The resulting straight-line approximations of the $F_i(x)$ all will have the same break points, but the breakpoints are otherwise arbitrary and can be selected for best fit of the family of functions to be generated.

Note that the second-order interpolation functions are sections of parabolas, the square of the first-order func-

tions. Therefore, the circuit for generating the second-order series is similar to that for the first-order series except that the $g^1{}_i$ are squared before being applied to the coefficient potentiometers, and an additional slope term is needed in the final summation.

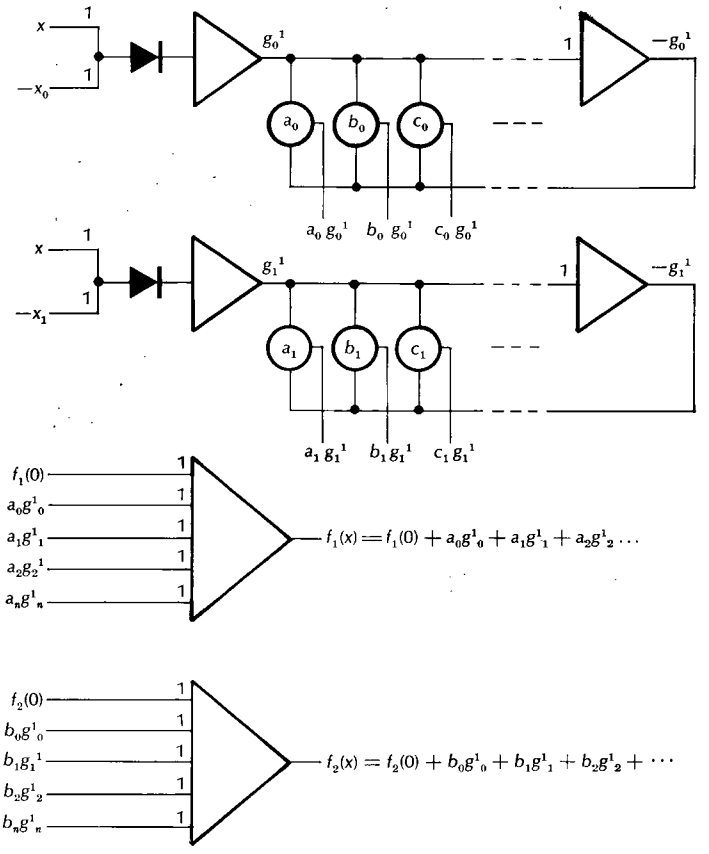


Figure 8 – Computer circuit for several functions of x using first-order interpolation functions, $g^1{}_i(x)$

GENERALIZED INTERPOLATION FUNCTIONS OF ORDER ONE

We are led, by consideration of scaling problems and function generator setup procedure, to consider a modification of the interpolation functions. In designing the actual computer setup of a general-purpose function generator based on the interpolation series, we encounter the occasional requirement for large slopes of some $F_j(x)$, hence large values of the coefficients a_{ij} in figure 8. We therefore consider the generalized interpolation functions $h^1{}_i$, shown in figure 9.

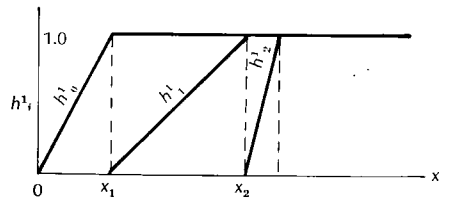


Figure 9 – Reflexed first-order interpolation functions

The functions $h_i(x)$ are defined as:

$$h^1{}_i(x) = 0 \text{ for } x \leq x_i$$
$$h^1{}_i(x) = m_i(x - x_i) \text{ for } x_i \leq x \leq x_{i+1}$$
$$h^1{}_i(x) = 1 \text{ for } x \geq x_{i+1} \qquad (15)$$

The functions $h^1{}_i(x)$ are easily generated on the computer by means of the circuitry shown in figure 10. Use of the reflexed interpolation functions eliminates the slope limitation of $F(x)$ encountered with the $g^1{}_i(x)$, since the circuitry shown in figure 10 can generate $h^1{}_i(x)$ with nearly infinite slope. Further, the $h^1{}_i(x)$ always range from 0 to 1; hence scaling problems associated with the interpolation functions are eliminated. This advantage is of particular importance in generation of functions of several variables.

The generalized interpolation functions of higher order obviously are reflexed parabolas, cubic curves, etc.

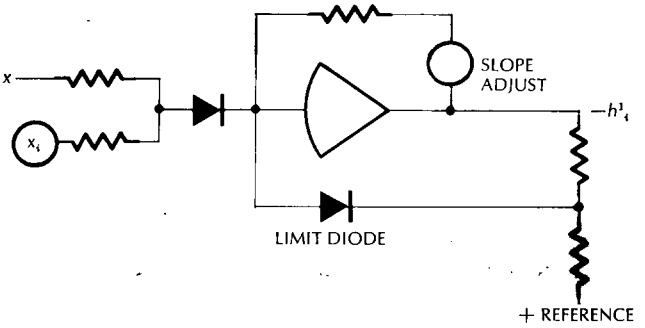


Figure 10 – Circuit for generation of reflexed first-order interpolation function

FUNCTIONS OF TWO OR MORE VARIABLES

Functions of two variables are generated by a double series related to series 8, 10, or 12. Consider the function of two variables shown in figure 11.

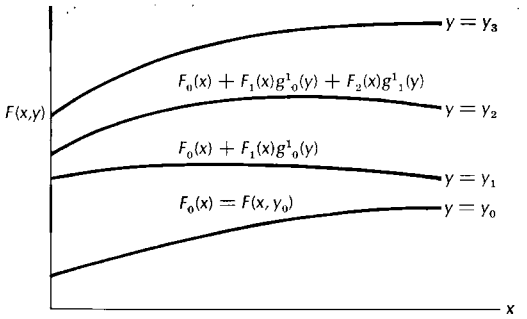


Figure 11 – Typical function of two variables

We can generate $F(x, y)$ with linear interpolation between values of y by means of the series:

$$F(x, y) = F_0(x) + F_1(x)g_0{}^1(y) + F_2(x)g_1{}^1(y) + F_3(x)g_2{}^1(y) + \ldots \qquad (16)$$

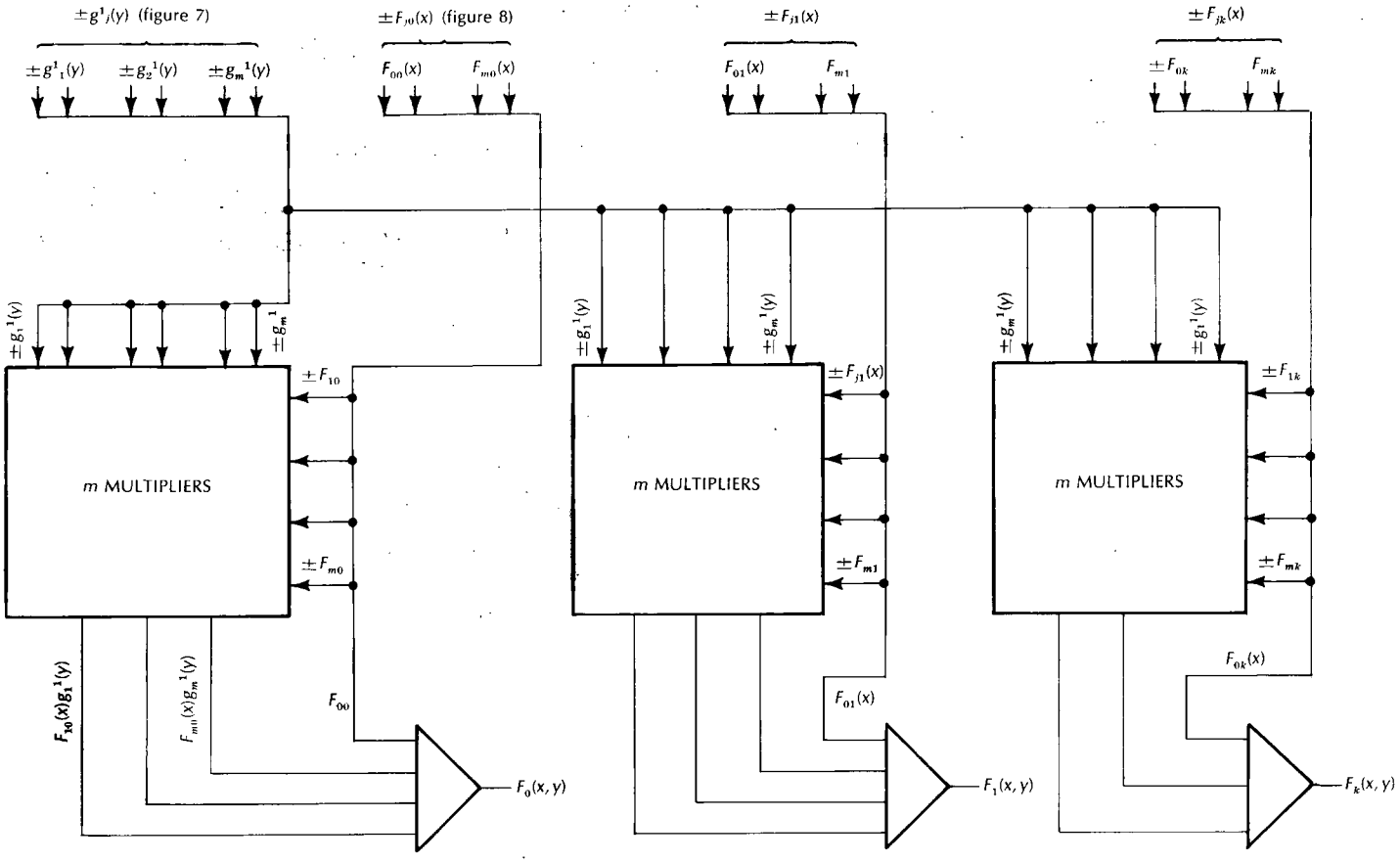


Figure 12 – Circuit for generation of k functions of two variables

The functions $F_j(x)$ are generated by the method described earlier for functions of a single variable.

Because of multiplier scaling problems, in many cases it is preferable to use reflex functions, $h_j^1(y)$, as shown in figure 9, and to represent $F(x, y)$ by means of the series:

$$F(x, y) = \sum_j F_j(x) h_j^1(y) \tag{17}$$

A block diagram for generating several functions of x and y, $F_k(x, y)$, is shown in figure 12.

TWO-DIMENSIONAL INTERPOLATING FUNCTIONS

The two-dimensional series (16) can be expanded in the alternate form:

$$F(x, y) = \sum_j F_j(x) g_j^1(y) = \sum_j \sum_i a_{ij} g_i^1(x) g_j^1(y) \tag{18}$$

$$F(x, y) = \sum_i \sum_j a_{ij} g_{ij}^1(x, y) \tag{19}$$

where

$$g_{ij}^1(x, y) = g_i^1(x) g_j^1(y) \tag{20}$$

Or, using reflexed functions,

$$F(x, y) = \sum_i \sum_j a_{ij} h_{ij}^1(x, y) \tag{21}$$

The functions $h_{ij}^1(x, y)$ are limited hyperboloids of the form shown in figure 13.

It is obvious that series (17) and the series (21) are equivalent. However, the generation of the two series on the

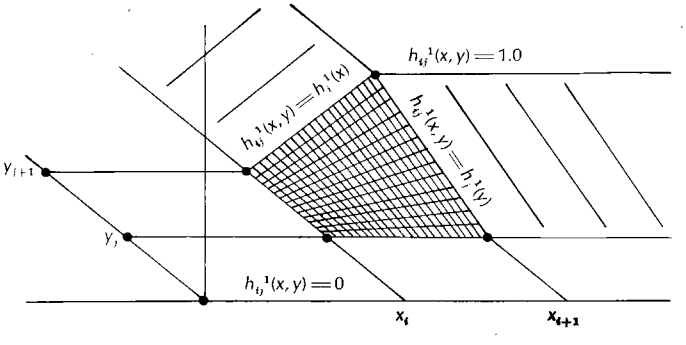


Figure 13 – The two-dimensional interpolation function of first order, $h_{ij}^1(x, y)$

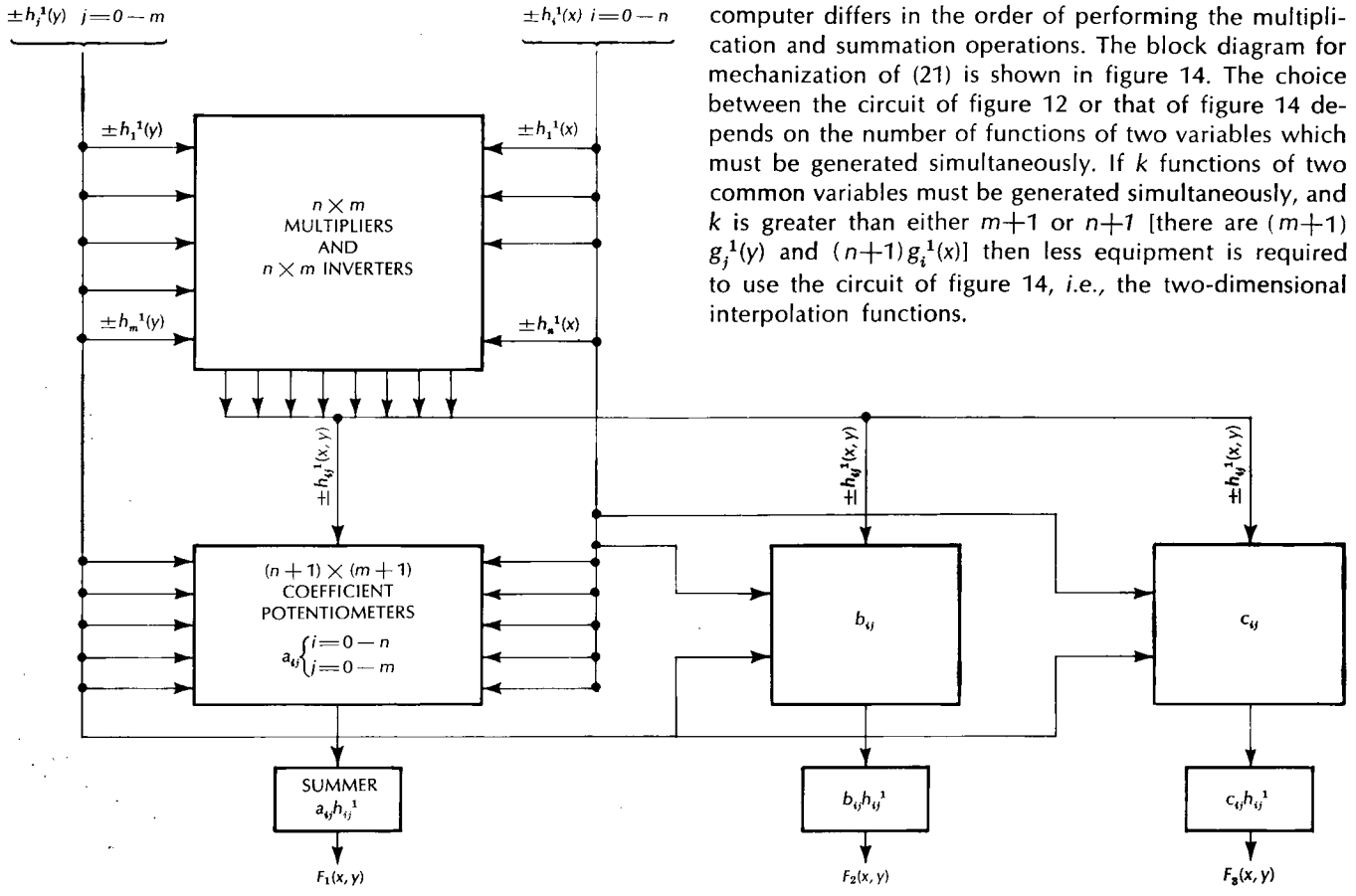


Figure 14 – Generation of two-dimensional functions using two-dimensional interpolation functions

computer differs in the order of performing the multiplication and summation operations. The block diagram for mechanization of (21) is shown in figure 14. The choice between the circuit of figure 12 or that of figure 14 depends on the number of functions of two variables which must be generated simultaneously. If k functions of two common variables must be generated simultaneously, and k is greater than either $m+1$ or $n+1$ [there are $(m+1)$ $g_j^1(y)$ and $(n+1)g_i^1(x)$] then less equipment is required to use the circuit of figure 14, *i.e.*, the two-dimensional interpolation functions.

FUNCTIONS OF MORE THAN TWO VARIABLES

The processes indicated by series (17) and (21) can be extended to any number of dimensions. For example, in analogy to (17), for three dimensions:

$$F(x, y, z) = \sum_k F_k(x, y) g_k^1(z) \tag{23}$$

$$= \sum_k \sum_j F_{jk}(x) g_j^1(y) g_k^1(z) \tag{24}$$

$$= \sum_k \sum_j \sum_i a_{ijk} g_i^1(x) g_j^1(y) g_k^1(z) \tag{25}$$

Expansion of the circuits of figure 12 or 14 to generate series (23) is straightforward. It is readily apparent that, when multidimensional functions must be generated, the quantities of equipment required can become very large indeed. If the same number of data divisions is used on each axis, the number of data points increases exponentially (*e.g.*, 10 data points for a one-dimensional data matrix, 100 data points for a 10×10 two-dimensional matrix, 1000 data points for a $10 \times 10 \times 10$ three-dimensional matrix, etc.). The quantity of equipment required tends to increase exponentially, also. One is therefore led to give serious consideration to any method which can decrease the number of required data divisions on each axis, since the equipment required tends to decrease exponentially. For example, if one can halve the data divisions required along each axis, the function generation equipment needed for a three-dimensional function should decrease by over 80%. While we do not at this time have sufficient experience to make a valid comparison, there are indications that use of the second-order interpolation series (14) rather than the first-order series can provide just such substantial savings in many cases where physical data are represented.

SYMMETRICAL FUNCTIONS

A great many functions are symmetrical about one or more axes. When this is the case, the required data divisions per axis of symmetry can be reduced by a factor of two by use of the simple absolute value circuit, shown in figure 15, on the input for each axis of symmetry.

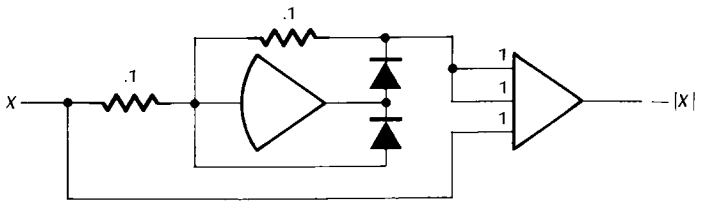


Figure 15 – Absolute value circuit

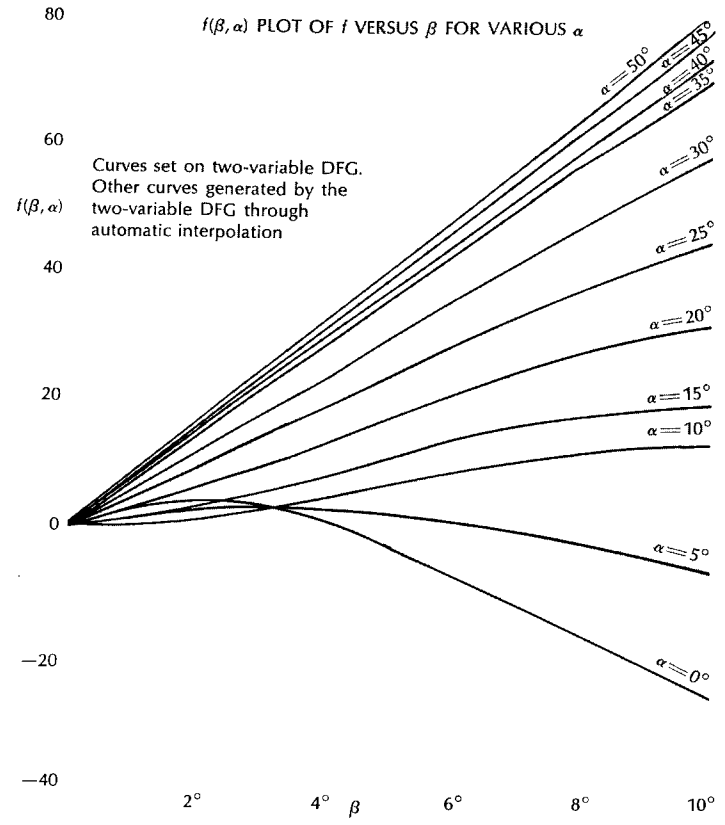


Figure 16 — Two-variable function generation $f(\beta, \alpha)$ vs. β for various α

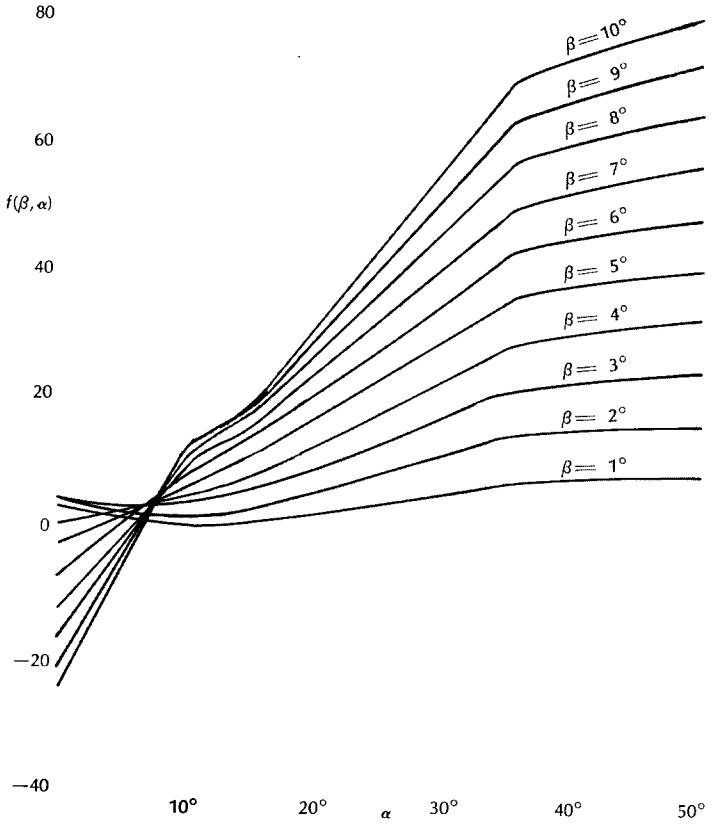


Figure 17 — Two-variable function generation $f(\beta, \alpha)$. Cross-plot of f versus α for various β.

EXPERIMENTAL RESULTS

Figure 16 shows a function of two variables, $f(\beta, \alpha)$, generated by means of the two-variable series (17).

Figure 17 shows a cross-plot made by holding β constant and sweeping through the range of α.

Figure 18 is a contour map of constant $f(\beta, \alpha)$. It was made by sweeping through the range of β and using the error between the generated $f(\beta, \alpha)$ and the desired constant value of f to drive the value of α.

APPROXIMATION OF FUNCTIONS OF SEVERAL VARIABLES BY MEANS OF ONE-DIMENSIONAL FUNCTIONS (Reference 3)

The scheme presented in the first eight sections reproduces $F(x, y)$ exactly at each of the data points x_i, y_j and provides intermediate values by means of a selected interpolation formula. The quantities of equipment required for functions of several variables can become very large if there are many data points.

In order to reduce the required quantity of equipment, we are led to consider approximating the multidimensional function by means of a series of functions of the individual variables. For computer purposes, a simple series approximation of $F(x, y)$ is:

$$F(x, y) = K + F_1(x) + G_1(y) + F_2(x)G_2(y) + F_3(x)G_3(y) + \ldots \tag{26}$$

$F(x, y)$ is tabulated as an $n \times m$ data matrix $F(x_i, y_j) i = 1 \ldots n, j = 1 \ldots m$.

Since we wish to terminate the series (26) as soon as acceptable accuracy is attained, we minimize the error of approximation term-by-term.

Because some portions of the data matrix may be of greater importance than others, *i.e.*, the required accuracy of the approximation may not be uniform over x and y, we weigh the errors differently in different portions of the field, x, y. Therefore we determine $F_i(x)$, $G_j(y)$ to minimize the weighted squared error term-by-term.

$$\varepsilon^2 = \sum_i \sum_j W_{ij} \Big[F(x_i, y_j) - K - F_1(x_i) - G_1(y_j) - F_2(x_i)G_2(y_j) - \ldots \Big]^2 \tag{27}$$

The squared error using only the first term is:

$$\varepsilon_0^{\,2} = \sum_i \sum_j W_{ij} \Big[F(x_i, y_j) - K \Big]^2 \tag{28}$$

and this is minimized when

$$\frac{\partial}{\partial K}(\varepsilon_0^{\,2}) = 0$$

$$0 = -2 \left\{ \sum_i \sum_j W_{ij} \Big[F(x_i, y_j) - K \Big] \right\}$$

or

$$K = \frac{\sum_i \sum_j W_{ij} F(x_i, y_j)}{\sum_i \sum_j W_{ij}} \tag{29}$$

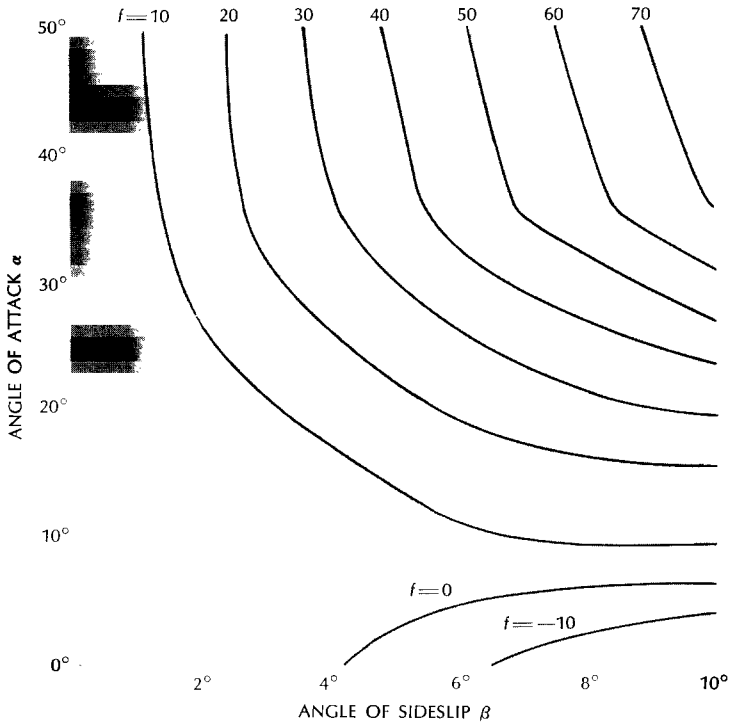


Figure 18 – Two-variable function generation $f(\alpha, \beta)$. Contours of constant f.

Thus K is the weighted mean value of $F(x_i, y_j)$.

Designating $F(x_i, y_j) - K = R_1(x_i, y_j)$, we now minimize

$$\varepsilon_1^2 = \sum_i \sum_j W_{ij} \left[R_1(x_i, y_j) - F_1(x_i) - G_1(y_j) \right]^2 \quad (30)$$

Consider, for example, the determination of the value of $F_1(x_i)$. Call this a_i. Then we wish to determine a_i such that

$$\varepsilon_{1i}^2 = \sum_j W_{ij} \left[R_1(x_i, y_j) - a_i - G_1(y_j) \right]^2 \quad (31)$$

is minimized. This will be true when

$$\frac{\partial}{\partial a_i} \varepsilon_{1i}^2 = -2 \sum_j W_{ij} \left[R_1(x_i, y_j) - a_i - G_1(y_j) \right] = 0$$

i.e.,

$$a_i = \frac{\sum_j W_{ij} \left[R_1(x_i, y_j) - G_1(y_j) \right]}{\sum_j W_{ij}} \quad (32)$$

Similarly, if the value of $G_1(y_j)$ at $y = y_j$ is b_j, then

$$b_j = \frac{\sum_i W_{ij} \left[R_1(x_i, y_j) - F_1(x_i) \right]}{\sum_i W_{ij}} \quad (33)$$

Since $F_1(x_i)$ and $G_1(y_j)$ are interdependent, it is necessary to determine the a_i and b_j by an iterative procedure. Convergence of the iteration has not been investigated but convergence has been obtained in all cases so far checked by starting with $G_1(y_j) = 0$.

Designating

$$F(x_i, y_j) - K - F_1(x_i) - G_1(y_j) = R_2(x_i, y_j) \quad (34)$$

we now minimize

$$\varepsilon_2^2 = \sum_i \sum_j W_{ij} \left[R_2(x_i, y_j) - F_2(x_i) G_2(y_j) \right]^2 \quad (35)$$

Proceeding as before, we let a_{2i} be the value of F_2 at $x = x_i$ and seek to determine a_{2i} such that ε_{2i}^2 is minimized.

Setting the derivative with respect to a_{2i} equal to zero we obtain

$$a_{2i} = \frac{\sum_j W_{ij} G_2(y_j) R_2(x_i, y_j)}{\sum_j W_{ij} \left[G_2(y_j) \right]^2} \quad (36)$$

Similarly, if b_{2j} is the value of $G_2(y)$ at $y = y_j$ then

$$b_{2j} = \frac{\sum_i W_{ij} F_2(x_i) R_2(x_i, y_j)}{\sum_i W_{ij} \left[F_2(x_i) \right]^2} \quad (37)$$

Again an iterative procedure is required and convergence has been obtained in all cases tried so far, starting with $G_2(y_j) = 1$.

The procedure for obtaining additional terms in the series (26) is the same. For a sizeable data matrix, the required calculations are too lengthy to perform by hand, so they have been programmed for a digital computer. For comparison purposes, the function of figure 16 was tabulated (table I) and reproduced on an analog computer as

$$F(x_i, y_j) = K + F_1(x) + G_1(y) + F_2(x) G_2(y) \quad (38)$$

Table I – Function of two variables, $F(\alpha, \beta)$, $\alpha = y_j$, $\beta = x_i$

$\beta = x_i$ \ $\alpha = y_j$	0°	10°	15°	35°	50°
0°	0	0	0	0	0
1°	3.0	0.1	.9	6.9	8.0
2°	3.9	1.0	2.9	14.2	16.3
3°	3.7	2.6	5.2	21.2	24.4
4°	1.5	5.0	8.0	28.3	32.3
5°	— 2.8	6.9	10.8	36.0	40.5
6°	— 7.1	8.4	13.7	42.8	48.4
7°	—12.0	10.4	15.9	50.0	56.2
8°	—16.8	12.1	17.8	57.0	64.2
9°	—21.6	13.0	19.0	63.3	72.5
10°	—26.7	13.0	19.2	70.2	80.6

All points were weighted equally. The analog results are shown in figure 19 and the digital computer print out in tables II-V. Tables III and V show the ideal residual errors, $R_2(x_i, y_j)$ and $R_3(x_i, y_j)$, where

$$R_2(x_i, y_j) = F(x_i, y_j) - K - F_1(x_i) - G_1(y_j) - F_2(x_i)G_2(y_j) \quad (39)$$

$$R_3(x_i, y_j) = R_2(x_i, y_j) - F_3(x_i)G_3(y_j) \quad (40)$$

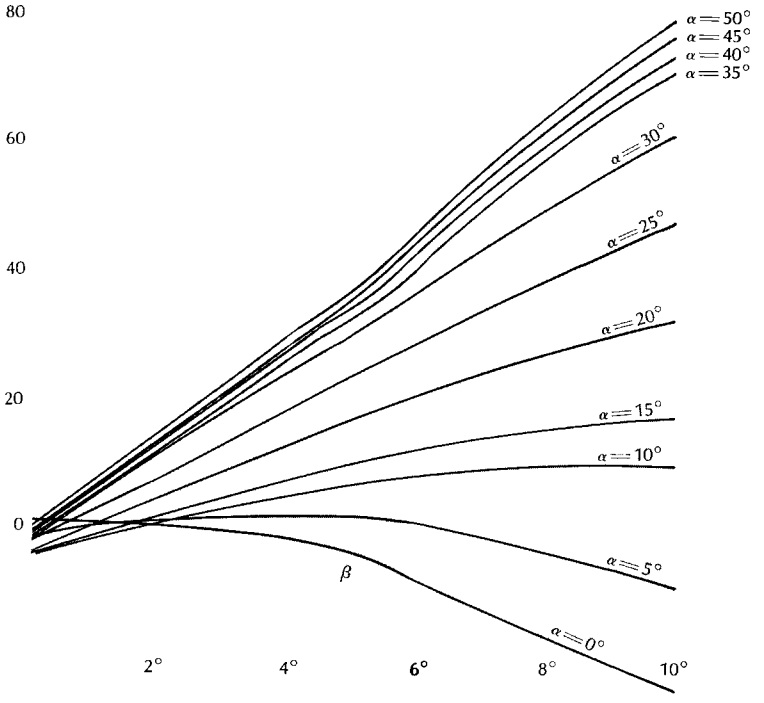


Figure 19 – Two-variable function, $f(\beta, \alpha)$.
$f(\beta, \alpha) = K + f_1(\beta) + g_1(\alpha) + f_2(\beta)g_2(\alpha)$.

Table II – Computer print out of

$$F(x_i, y_j) = K + K_1F_1(x_i) + K_2F_2(y_j) + K_3F_3(x_i)F_4(x_j) + \cdots + k_nF_n(x_i)F_m(y_j) + E(x_i, y_j)$$

$K = +0.17169\,E + 02$ $(K = 17.169)$

$K_{01} = +0.17169\,E + 02$

x_i	$F_1(x_i)$
01	$-.10000\,E + 01$
02	$-.77984\,E + 00$
03	$-.55385\,E + 00$
04	$-.33485\,E + 00$
05	$-.12517\,E + 00$
06	$+.64704\,E - 01$
07	$+.23711\,E + 00$
08	$+.40369\,E + 00$
09	$+.56444\,E + 00$
10	$+.70306\,E + 00$
11	$+.82071\,E + 00$

$K_{02} = +.23978\,E + 02$

y_j	$F_2(y_j)$
01	$-.10000\,E + 01$
02	$-.44116\,E + 00$
03	$-.28609\,E + 00$
04	$+.76221\,E + 00$
05	$+.96504\,E + 00$

$K_{03} = +.33586\,E + 02$

x_i	$F_3(x_i)$
1	$-.81593\,E + 00$
2	$-.71045\,E + 00$
3	$-.57337\,E + 00$
4	$-.42867\,E + 00$
5	$-.26073\,E + 00$
6	$-.52328\,E - 01$
7	$+.14594\,E + 00$
8	$+.35493\,E + 00$
9	$+.56468\,E + 00$
10	$+.77593\,E + 00$
11	$+.10000\,E + 01$

y_j	$F_4(y)$
1	$-.10000\,E + 01$
2	$-.25327\,E + 00$
3	$-.15167\,E + 00$
4	$+.62462\,E + 00$
5	$+.78032\,E + 00$

Table III – Error table

$E_3 = F(x_i, y_j) - [K + K_1F_1(x_i) + K_2F_2(y_j) + K_3F_3(x_i)F_4(y_j)]$

x_i \ y_j	1	2	3	4	5
1	$-.34259\,E + 01$	$+.36377\,E + 01$	$+.27036\,E + 01$	$-.11593\,E + 01$	$-.17561\,E + 01$
2	$-.66323\,E + 00$	$+.85488\,E + 00$	$+.36094\,E + 00$	$-.25211\,E + 00$	$-.30050\,E + 00$
3	$+.96089\,E + 00$	$-.95905\,E + 00$	$-.82076\,E + 00$	$+.29208\,E + 00$	$+.52683\,E + 00$
4	$+.18609\,E + 01$	$-.18882\,E + 01$	$-.15436\,E + 01$	$+.49645\,E + 00$	$+.10745\,E + 01$
5	$+.17012\,E + 01$	$-.16597\,E + 01$	$-.14882\,E + 01$	$+.47341\,E + 00$	$+.97325\,E + 00$
6	$+.11407\,E + 01$	$-.12469\,E + 01$	$-.88656\,E + 00$	$+.54140\,E + 00$	$+.45141\,E + 00$
7	$+.53979\,E + 00$	$-.10204\,E + 01$	$+.63424\,E - 01$	$+.22200\,E + 00$	$+.19518\,E + 00$
8	$-.20094\,E + 00$	$-.10266\,E + 00$	$+.46804\,E + 00$	$+.17765\,E + 00$	$-.34207\,E + 00$
9	$-.71646\,E + 00$	$+.62148\,E + 00$	$+.67648\,E + 00$	$+.17550\,E - 01$	$-.59901\,E + 00$
10	$-.80133\,E + 00$	$+.93843\,E + 00$	$+.57260\,E + 00$	$-.49420\,E + 00$	$-.21549\,E + 00$
11	$-.39561\,E + 00$	$+.82444\,E + 00$	$-.10598\,E + 00$	$-.31491\,E + 00$	$-.79536\,E - 02$

Table IV – Second product, $K_5F_5(x_i)F_6(y_j)$

$K_5 = +.36383\,E + 01$ $(K_5 = 3.6383)$

x_i	$F_5(x_i)$
1	$-0.10000\,E + 01$
2	$-0.19552\,E + 00$
3	$+0.27965\,E + 00$
4	$+0.54177\,E + 00$
5	$+0.49451\,E + 00$
6	$+0.33380\,E + 00$
7	$+0.16317\,E + 00$
8	$-0.54143\,E - 01$
9	$-0.20676\,E + 00$
10	$-0.23556\,E + 00$
11	$-0.12092\,E + 00$

y_j	$F_6(y_j)$
1	$+0.94167\,E + 00$
2	$-0.10000\,E + 01$
3	$-0.74157\,E + 00$
4	$+0.30988\,E + 00$
5	$+0.49002\,E + 00$

Table V – Error table

$E_5 = F(x_i, y_j) - [K + K_1F_1(x_i) + K_2F_2(y_j) + K_3F_3(x_i)F_4(y_j) + K_5F_5(x_i)F_6(y_j)]$

x_i \ y_j	1	2	3	4	5
1	$+.24319\,E - 03$	$-.06234\,E - 03$	$+.55437\,E - 02$	$-.31871\,E - 01$	$+.26717\,E - 01$
2	$+.66345\,E - 02$	$+.14353\,E + 00$	$-.16658\,E + 00$	$-.31676\,E - 01$	$+.48081\,E - 01$
3	$+.27882\,E - 02$	$+.58404\,E - 01$	$-.66243\,E - 01$	$-.23214\,E - 01$	$+.28258\,E - 01$
4	$+.47059\,E - 02$	$+.82950\,E - 01$	$-.81910\,E - 01$	$-.11437\,E + 00$	$+.10861\,E + 00$
5	$+.69408\,E - 02$	$+.13952\,E + 00$	$-.15395\,E + 00$	$-.84133\,E - 01$	$+.91611\,E - 01$
6	$-.29480\,E - 02$	$-.32467\,E - 01$	$+.14055\,E - 01$	$+.16505\,E + 00$	$-.14370\,E + 00$
7	$-.19243\,E - 01$	$-.42674\,E + 00$	$+.50367\,E + 00$	$+.38038\,E - 01$	$-.95721\,E - 01$
8	$-.15440\,E - 01$	$-.29965\,E + 00$	$+.32196\,E + 00$	$+.23870\,E + 00$	$-.24554\,E + 00$
9	$-.80758\,E - 02$	$-.13079\,E + 00$	$+.11862\,E + 00$	$+.25066\,E + 00$	$-.23039\,E + 00$
10	$+.57073\,E - 02$	$+.81402\,E - 01$	$-.62950\,E - 01$	$-.22862\,E + 00$	$+.20447\,E + 00$
11	$+.18664\,E - 01$	$+.38450\,E + 00$	$-.43222\,E + 00$	$-.17858\,E + 00$	$+.20762\,E + 00$

DISCUSSION

The error tables, III and V, show that using only four functions of the two variables α and β, one should be able to reproduce the function $f(\beta, \alpha)$ with a maximum error of about 4%. Addition of a second product to the series as shown in (26) further reduces the theoretical error to well under 1%. Practically, it is found that the actual errors are somewhat greater than the theoretical errors, due to inaccuracy in setting the single variable diode function generators, etc. Nevertheless, the scheme works well for most data from physical systems and is a reasonable solution of a difficult problem in many cases. The computations for factoring the data matrix into series form are fairly lengthy if performed by hand, so they have been programmed for solution on a digital computer. Running time on the small PDP-8 computer was under 30 seconds, even though the iterative calculations were arbitrarily repeated 200 times.

FUNCTIONS OF MORE THAN TWO VARIABLES

Extension of the series (26) to more than two variables is obvious, but again it is found that the quantity of equipment needed tends to increase exponentially. Frequently the data are given for only a small number of values of a third variable, and in this case series (26) can be combined with (23) to good advantage.

MODIFICATION OF THE APPROXIMATING SERIES

Inspection of the functions $F_1(x)$, $G_1(y)$, $F_2(x)$, $G_2(y)$ used in producing figure 19 shows that there is a substantially linear trend to each of these functions. For ease of computer realization, probably one should modify the series (26) to be of the form:

$$F(x_i, y_j) = K_1 + K_2x + K_3y + K_4(xy) + K_5F_1(x)G_1(y) + \cdots \tag{41}$$

Determination of the unknown coefficients and functions in (41) is similar to the procedure for (26) and should provide maximum accuracy with the least amount of set-up time.

For the series (41) we obtain:

$$K_1 = \frac{\sum_i \sum_j W_{ij}F(x_i, y_j)}{\sum_i \sum_j W_{ij}} \tag{42}$$

$$K_2 = \frac{\sum_i \sum_j W_{ij}\,[R_1(x_i, y_j) - K_3y_j]}{\sum_i \sum_j W_{ij}x_i} \tag{43}$$

$$K_3 = \frac{\sum_i \sum_j W_{ij}[R_1(x_i, y_j) - K_2x_i]}{\sum_i \sum_j W_{ij}y_j} \tag{44}$$

$$K_4 = \frac{\sum_i \sum_j W_{ij}\,[R_2(x_i, y_j)]\,[x_iy_j]}{\sum_i \sum_j W_{ij}(x_i, y_j)^2} \tag{45}$$

where

$$R_1(x_i, y_j) = F(x_i, y_j) - K_1 \tag{46}$$

$$R_2(x_i, y_j) = R_1(x_i, y_j) - K_2x_i - K_3y_j \tag{47}$$

CONCLUSION

Two schemes for generating functions of several variables have been described. Both schemes have been shown to be practical, but both have limitations. The first scheme is capable (theoretically) of reproducing a data matrix without error but the quantities of equipment required may become very large. The second scheme provides a series approximation of the data matrix which usually converges very rapidly, but requires precomputation to determine the terms of the series. In many cases, this is not a serious penalty to pay in view of the modest amount of equipment required to obtain acceptable accuracy.

ACKNOWLEDGMENT

The digital computer program for generating the approximating series (26) was designed by Mr. John Brussolo of Applied Dynamics, Inc., who also performed all of the digital computer runs.

REFERENCES

1 H D HUSKEY G A KORN
Computer handbook section 3 part 3
Contains a bibliography of function generation
McGraw-Hill Book Company New York 1962

2 C LANCZOS
Linear differential operators section 6.2

3 E W PIKE T R SILVERBERG
Designing mechanical computers
Machine Design July 1962