1963

Applications of real polynomials of binary variables

Wendell Beck Sander
Iowa State University

SANDER, Wendell Beck, 1935-
APPLICATIONS OF REAL POLYNOMIALS OF
BINARY VARIABLES.

Iowa State University of Science and Technology
Ph.D., 1963
Engineering, electrical

APPLICATIONS OF REAL POLYNOMIALS OF BINARY VARIABLES

by

Wendell Beck Sander

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of

The Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Electrical Engineering

Approved:

Signatures have been redacted for privacy.

Iowa State University
Of Science and Technology
Ames, Iowa

1963

TABLE OF CONTENTS

I. INTRODUCTION

This dissertation develops a mathematical tool and shows how this tool can be used for analysis and synthesis in a variety of areas.

As digital and analog computer technology has progressed there has been increasing interest in the interface between analog systems (or analog nature itself) and discrete digital processors. Special coding schemes such as reflected binary codes and weighted 4-bit codes for the integers 0-9 have been investigated (1,26,29). Methods for translating from digital to analog and analog to digital forms have been extensively pursued (21) including some investigation into decoding from digital form to an analog representation of a function of the digital form (7,16). These interface problems have been investigated largely on an intuitive or cut-and-try basis.

The defining relations describing the operation of digital processors are nearly always in terms of the symbolic logic known as Boolean algebra invented by George Boole (3) (1815-1864) and extended by such people as Whitehead and Russell (30). This algebra is applicable to two-level or binary operations which led Shannon (27) to apply the algebra to relay and switching networks which are basically two-level in nature. Since Shannon's early work there has been extensive study of Boolean algebra and its applications.

A relatively new field that is rather loosely connected with computer technology is the field of pattern recognition. Many papers have been written in this field in recent years (2,14,17,19,31). Most of these schemes consider the pattern to be a pattern of white and black areas from which a set of "characteristics" of the pattern are derived. The decision (usually linear) as to what pattern is being observed is based on the relative values of the characteristics. The fundamental problem in all such schemes is to choose a good set of characteristics. Since most schemes translate the pattern into a finite two-valued two-dimensional matrix a secondary problem exists in getting high resolution without undue complication. Since the patterns are observed as white and black only, multitone (black-grey-white) patterns cannot be handled effectively.

This paper investigates the application of real polynomials of binary variables to the above areas. Coleman (6) makes use of a real polynomial approach in the design of core logic which is essentially a linearly separable function problem. He considers only the case of a two-valued function and only fitting "complete" functions (functions specified for all possible values of the variables) with an orthogonal set of variables. This dissertation is much more general in approach, allowing arbitrary functions of the binary variables and considering incomplete functions.

II. REAL POLYNOMIALS OF BINARY VARIABLES

A. Arbitrary Functions

<u>Definition 1</u>:

A two-valued variable is a variable x_j that can take on one of only two finite values x_j^1 and x_j^2 where $x_j^1 \neq x_j^2$. x_j^1 and x_j^2 are both real.

<u>Definition 2</u>:

A real polynomial of n two-valued variables is a function f such that

$$\begin{aligned} f(x_1,x_2,x_3,\ldots,x_n) = {} & k_0+k_1x_1+k_2x_2+\ldots+k_nx_n \\ & + k_{12}x_1x_2+k_{13}x_1x_3+\ldots \\ & + k_{123\ldots n}x_1x_2\ldots x_n \end{aligned} \tag{1}$$

where $x_1,x_2,\ldots,x_n$ are two-valued variables, k_0, k_1, $k_2,\ldots$, $k_{12\ldots n}$ are constant coefficients of the polynomial, and the indicated operations are real multiplication and real addition.

<u>Definition 3</u>:

A complete function of n two-valued variables is a function defined for all possible combinations of values of the n variables. An incomplete function is a function of two-valued variables that is not complete.

Several observations can be made at this point. The domain of a complete function can be visualized as an n-dimensional hyper-rectangle with each vertex of the hyper-rectangle corresponding to a point of definition of the function. Thus the domain is a set of 2^n finite points and the function takes on a finite set of values. This paper is concerned only with functions that have finite values at all points.

Definition 4:

A set of binary variables is a set $x_1, x_2, \ldots, x_n$ of two-valued variables such that $x_1^1 = x_2^1 = x_3^1 = \ldots = x_n^1$ and $x_1^2 = x_2^2 = x_3^2 = \ldots = x_n^2$.

Definition 5:

A real polynomial of binary variables is a real polynomial of a set of two-valued variables where the set of two-valued variables is a set of binary variables.

Any function of two-valued variables can be represented by a finite table listing the possible combinations of values that the variables $\{x_j\}$ take on and the value of the function for each point. An example of such a table for a complete function of three two-valued variables is shown in Table 1.

Table 1. General function of three two-valued variables

x_3	x_2	x_1	$f(x_1,x_2,x_3)$
x_3^1	x_2^1	x_1^1	y_0
x_3^1	x_2^1	x_1^2	y_1
x_3^1	x_2^2	x_1^1	y_2
x_3^1	x_2^2	x_1^2	y_3
x_3^2	x_2^1	x_1^1	y_4
x_3^2	x_2^1	x_1^2	y_5
x_3^2	x_2^2	x_1^1	y_6
x_3^2	x_2^2	x_1^2	y_7

B. The 0,1 Variable

Definition 6:

The variable z_j is a two-valued variable such that $z_j^1 = 0$ and $z_j^2 = 1$.

It follows directly that a set of variables $\{z_j\}$ is a set of binary variables.

Definition 7:

The negation of a two-valued variable x_j is denoted by $\overline{x_j}$ and is defined such that $\overline{x_j^1} = x_j^2$ and $\overline{x_j^2} = x_j^1$.

Theorem 1: $\overline{z_j} = 1 - z_j$

Proof: We have $x_j^1 = 0$ and $z_j^2 = 1$ then

$$1 - z_j^1 = 1 - 0 = 1 = z_j^2 = \overline{z_j^1}$$

and

$$1 - z_j^2 = 1 - 1 = 0 = z_j^1 = \overline{z_j^2}$$

A function of three z variables is represented in Table 2.

Table 2. General function of z_1, z_2, z_3

z_3	z_2	z_1	$f(z_1,z_2,z_3)$
0	0	0	y_0
0	0	1	y_1
0	1	0	y_2
0	1	1	y_3
1	0	0	y_4
1	0	1	y_5
1	1	0	y_6
1	1	1	y_7

We can now proceed with two theorems that will allow us to find a polynomial in $\{z_j\}$ directly.

Theorem 2: Given any finite complete function f of n variables $\{z_j\}$ this function can be written as

$$f(z_1,z_2,\ldots,z_n) = (y_0\bar{z}_1\bar{z}_2\ldots\bar{z}_n) + (y_1z_1\bar{z}_2\ldots\bar{z}_n) + (y_2\bar{z}_1z_2\ldots\bar{z}_n) + \ldots + (y_2z_1z_2\ldots z_n). \tag{2}$$

Proof: Consider the k th point $z_{ak},z_{bk},\ldots,z_{jk},z_{j+1k},\ldots,z_{nk}$ where the values of $z_{ak},z_{bk},\ldots,z_{jk}$ are all 0 and the values of $z_{j+1k},\ldots,z_{nk}$ are all 1. Then consider the 2^n combinations of real products of z and $\bar{z}$ variables such that none of the n variables appear both true and negated but all variables $\{z_j\}$ appear either true or negated. The only one of these products that is not zero for the k th point is the product $\bar{z}_a\bar{z}_b\ldots\bar{z}_jz_{j+1}\ldots z_n$ and the product at that point is one. Therefore, the only term of the function 2 that is non-zero is the term $c\bar{z}_a\bar{z}_b\ldots\bar{z}_jz_{j+1}\ldots z_n$ where c is the coefficient of this term and we have

$$f(z_a=0,\ z_b=0,\ldots,\ z_j=0,\ z_{j+1}=1,\ldots,\ z_n=1)$$

$$= c\bar{z}_{ak}\bar{z}_{bk}\ldots\bar{z}_{jk}z_{j+1k}\ldots z_{nk} = c$$

Q.E.D.

Theorem 3: Any arbitrary finite complete function f of n variables $\{z_j\}$ can be uniquely represented as a polynomial in $\{z_j\}$.

Proof: From Theorem 1

$$\bar{z}_j = 1 - z_j$$

Therefore 2 can be written as

$$f(z_1,z_2,\ldots,z_n) = [y_0(1-z_1)(1-z_2)\ldots(1-z_n)] + [y_1z_1(1-z_2)\ldots(1-z_n)] + \ldots$$

$$+ [y_{2^n-1} z_1 z_2 \cdots z_n]$$

which is a real polynomial in the binary variables $z_1, z_2, \ldots, z_n$, and can be reduced to the form of 1.

The variables $\{z_j\}$ are convenient because all functions of variables $\{z_j\}$ may be represented as polynomials and these polynomials may be readily derived from a function table.

C. Change of Variables

<u>Theorem 4</u>: Any function of n variables $\{z_j\}$ can be transformed to a function of n two-valued variables $\{x_j\}$ by the transformation

$$z_j = \frac{x_j - x_j^1}{x_j^2 - x_j^1} \tag{3}$$

and any function of n variables $\{x_j\}$ can be transformed to a function of n variables $\{z_j\}$ by the transformation

$$x_j = z_j(x_j^2 - x_j^1) + x_j^1 \tag{4}$$

The proof follows directly from substituting for x_j in 3 and for z_j in 4 and from $x_j^1 \neq x_j^2$.

<u>Theorem 5</u>: Any finite function of n two-valued variables $\{x_j\}$ can be uniquely written as a real polynomial in $\{x_j\}$.

<u>Proof</u>: Given the function it can be transformed to a function of variables $\{z_j\}$ by Theorem 4 and then written as a polynomial in $\{z_j\}$. The function can then be transformed back to the variables $\{x_j\}$ by the linear transformation 3 resulting in a polynomial in $\{x_j\}$.

D. Binary Functions

It is of interest to consider the special case of two-valued functions of two-valued variables, that is, functions which take on one of only two possible values. These special functions can be used to describe switching networks and computer circuits much as Boolean algebra is used.

In Boolean algebra it is customary to denote the two values of a binary variable as 0 and 1. Although this choice is arbitrary and was probably chosen for convenience, it leads to direct equivalence between a real polynomial form and a Boolean polynomial form.

Given a binary function of n variables $\{z_j\}$ that takes on one of the two values 0 and 1; the functional form of 2 is identical in form to the P-term canonical form (5) of the Boolean representation of the function, as defined by the function table, with logical negation analogous to variable negation of Definition 7, logical inclusive OR analogous to the real sum, and logical product analogous to the real product.

Note that the function table of such a function in real variables is indeed identical to a truth table of Boolean logic.

E. The Orthogonal Variable

Definition 8:

The variable v_j is a binary variable such that $v_j^1 = - 1$ and $v_j^2 = + 1$.

It can be shown (15,17) that a set of variables v_j form an orthogonal set (A proof by mathematical induction is given in Appendix A.). That is, given variables $v_1,v_2,\ldots,v_n$ and all possible products of these variables $v_1v_2,v_1v_3,\ldots,v_1v_n,\ldots,v_1v_2\ldots v_n$ the variables and all products as listed in a function table are mutually orthogonal and are all orthogonal to a constant in the sense that

$$\sum_{k=1}^{2^n} v_{jk}v_{jk} = 2^n$$

and

$$\sum_{k=1}^{2^n} f_{jk} = 0$$

where k is an index of rows of the function table. Thus, all terms of a real polynomial in $\{v_j\}$ are mutually orthogonal.

An algorithm for direct computation of the coefficients of the polynomial in $\{v_j\}$ is developed in Appendix B.

The coefficient $c_{ij\ldots m}$ of the term $v_i v_j \ldots v_m$ of the v polynomial is

$$c_{ij\ldots m} = \frac{1}{2^n} \sum_{k=1}^{2^n} v_{ik}b_{jk}\cdots v_{mk}y_k \tag{5}$$

and the constant term c_0 is

$$c_0 = \frac{1}{2^n} \sum_{k=1}^{2^n} y_k \,. \tag{6}$$

These coefficients can also be derived from the theory of orthogonal polynomials (8) and Fourier series (6). It is interesting that both orthogonal polynomials and Fourier series reduce to the same system in two-valued variables. Table 3 is a function table illustrating a particular function and sets of variables in $\{z_j\}$ and $\{v_j\}$ including the products of the variables $\{v_j\}$.

Table 3. Example of a particular function in $\{z_j\}$ and $\{v_j\}$.

z_3	z_2	z_1	v_3	v_2	v_1	v_3v_2	v_3v_1	v_2v_1	$v_3v_2v_1$	y
0	0	0	-1	-1	-1	+1	+1	+1	-1	2
0	0	1	-1	-1	+1	+1	-1	-1	+1	1
0	1	0	-1	+1	-1	-1	+1	-1	+1	3
0	1	1	-1	+1	+1	-1	-1	+1	-1	1
1	0	0	+1	-1	-1	-1	-1	+1	+1	2
1	0	1	+1	-1	+1	-1	+1	-1	-1	2
1	1	0	+1	+1	-1	+1	-1	-1	-1	3
1	1	1	+1	+1	+1	+1	+1	+1	+1	4

Notice that the variables $\{v_j\}$ and their products are indeed mutually orthogonal and each column v_j contains an equal number of +1 and -1 terms. For Table 3 the polynomials in $\{z_j\}$ and $\{v_j\}$ will now be developed.

The polynomial of the form 2 may be written directly.

$$y=2\bar{z}_1\bar{z}_2\bar{z}_3+z_1\bar{z}_2\bar{z}_3+3\bar{z}_1z_2\bar{z}_3+z_1z_2\bar{z}_3+2\bar{z}_1\bar{z}_2z_3+2z_1\bar{z}_2z_3+3\bar{z}_1z_2z_3+4z_1z_2z_3 .$$

Substituting $\bar{z}_j = 1-z_j$ gives

$$y=2(1-z_1)(1-z_2)(1-z_3)+z_1(1-z_2)(1-z_3)+3(1-z_1)z_2(1-z_3)+z_1z_2(1-z_3)$$

$$+2(1-z_1)(1-z_2)\,z_3+2z_1(1-z_2)z_3+3(1-z_1)z_2z_3+4z_1z_2z_3$$

which reduces to

$$y=2-z_1+z_2-z_1z_2+z_1z_3+2z_1z_2z_3 . \qquad (7)$$

The coefficients of the polynomial in $\{v_j\}$ are

$$c_0 = \frac{1}{8}\sum_{k=1}^{8} y_k = \frac{18}{8} \quad ; \quad c_1 = \frac{1}{8}\sum_{k=1}^{8} v_{1k}y_k = -\frac{2}{8}$$

$$c_2 = \frac{1}{8}\sum_{k=1}^{8} v_{2k}y_k = \frac{4}{8} \quad ; \quad c_3 = \frac{1}{8}\sum_{k=1}^{8} v_{3k}y_k = \frac{4}{8}$$

$$c_{12} = \frac{1}{8}\sum_{k=1}^{8} v_{1k}v_{2k}y_k = 0 \quad ; \quad c_{13}\ \frac{1}{8}\sum_{k=1}^{8} v_{1k}v_{3k}y_k = \frac{4}{8}$$

$$c_{23} = \frac{1}{8}\sum_{k=1}^{8} v_{2k}v_{3k}y_k = \frac{2}{8} \quad ; \quad c_{123} = \frac{1}{8}\sum_{k=1}^{8} v_{1k}v_{2k}v_{3k}y_k = \frac{2}{8} \ .$$

Thus the polynomial of Table 3 in $\{v_j\}$ is

$$y = \frac{1}{4}(9-v_1+2v_2+2v_3+2v_1v_3+2v_1v_3+v_2v_3+v_1v_2v_3) \quad . \tag{8}$$

The equivalence of 7 and 8 can be checked by the transformation $z = \frac{v+1}{2}$ applied to 7. This gives

$$y = 2 - \frac{(v_1+1)}{2} + \frac{(v_2+1)}{2} - \frac{(v_1+1)(v_2+1)}{4} + \frac{(v_1+1)(v_3+1)}{4} + 2\,\frac{(v_1+1)(v_2+1)(v_3+1)}{8}$$

$$= 2 - \frac{1}{2}v_1 - \frac{1}{2} + \frac{1}{2}v_2 + \frac{1}{2} - \frac{1}{4}v_1v_2 - \frac{1}{4}v_1 - \frac{1}{4}v_2 - \frac{1}{4} + \frac{1}{4}v_1v_3 + \frac{1}{4}v_1 + \frac{1}{4}v_3$$

$$+ \frac{1}{4} + \frac{1}{4}v_1v_2v_3 + \frac{1}{4}v_1v_2 + \frac{1}{4}v_1v_3 + \frac{1}{4}v_2v_3 + \frac{1}{4}v_1 \quad \frac{1}{4}v_2 + \frac{1}{4}v_3 + \frac{1}{4}$$

$$= \frac{1}{4}(9-v_1+2v_2+2v_3+2v_1v_3+v_2v_3-v_1v_2v_3).$$

Thus the two polynomials are equivalent.

F. Approximation and Least Squares Fitting

In some cases it is desirable to find an approximate function of n two-valued variables that fits a given function to a required degree of accuracy. For example, a complete polynomial in 10 variables could have up to 2^{10} or 1024 terms. This can be unrealistic and unnecessary in many applications.

A common and very desirable method of curve fitting is the method of least squares. If we denote the approximated value of the function by $\hat{y}_k$ and the true value by y_k then the coefficients of the least squares polynomial are the polynomial coefficients that give a minimum

$$\sum_{k=1}^{2^n} (y_k - \hat{y}_k)^2 . \tag{9}$$

The orthogonal variables $\{v_j\}$ are particularly convenient in least squares approximation of complete functions of two-valued variables due to the following two theorems.

<u>Theorem 6</u>: Given any finite complete function f of binary variables written as a real polynomial P in the binary variables $\{v_j\}$; the approximate polynomial P formed by deleting one or more of the terms of P is the least squares best fitting polynomial in the remaining polynomial terms of P.

<u>Proof</u>: Identify each of the possible polynomial terms of P by a set of variables $p_1,p_2,\ldots,p_j,\ldots,p_{2^n}$ arranged such that the p_{j+1} to p_{2^n} terms are those to be deleted. The corresponding coefficients are $c_1,c_2,\ldots,c_{2^n}$. The exact function values y_k and the approximate function values $\hat{y}_k$ can be written as

$$y_k = c_1p_{1k}+c_2p_{2k}+\ldots+c_{2^n}p_{2^n k} \tag{10}$$

$$\hat{y}_k = c_1'p_{1k}+c_2'p_{2k}+\ldots+c_j'p_{jk}$$

where $c_1',c_2',\ldots,c_j'$ are the least squares best fit polynomial coefficients for the polynomial terms $p_1,p_2,\ldots,p_j$.

The squared error E is

$$E = \sum_{k=1}^{2^n} (y_k - \hat{y}_k)^2$$

$$= \sum_{k=1}^{2^n} (y_k - c_1' p_{1k} - c_2' p_{2k} - \cdots - c_j' p_{jk})^2$$

$$= \sum_{k=1}^{2^n} (y - \sum_{i=1}^{j} c_i' p_{ik})^2$$

Differentiating E with respect to $c_1', c_2', \ldots, c_j'$ and setting the result equal to zero yields a system of j equations of the form

$$\frac{\partial E}{\partial c_L'} = \sum_{k=1}^{2^n} 2(-y_k p_{Lk} + p_{Lk} \sum_{i=1}^{2^n} c_i p_{ik}) = 0$$

or

$$\sum_{k=1}^{2^n} p_{Lk} \sum_{i=1}^{j} c_i' p_{ik} = \sum_{k=1}^{2^n} y_k p_{Lk}$$

Rearranging the summation on the left yields

$$\sum_{i=1}^{j} c_i' \sum_{k=1}^{2^n} p_{ik} p_{Lk} = \sum_{k=1}^{2^n} y_k p_{Lk} \tag{11}$$

But from the orthogonality shown in Appendix A every term

$\sum_{k=1}^{2^n} p_{ik} p_{Lk} = 0$ except for i=L and $\sum_{k=1}^{2^n} p_{Lk} p_{Lk} = 2^n$.

Therefore 11 reduces to

$$2^n c_L' = \sum_{k=1}^{2^n} y_k p_{Lk}$$

or

$$c_L' = \frac{1}{2^n}\sum_{k=1}^{2^n} y_k p_{Lk}$$

which from 5 is the coefficient of the p_L term of the original polynomial P. Thus

$$c_L' = c_L \qquad \text{Q.E.D.}$$

<u>Theorem 7</u>: Given any finite complete function f of binary variables written as a polynomial P in $\{v_j\}$ and approximated by deleting terms $c_{j+1}p_{j+1}, c_{j+2}p_{j+2}, \ldots, c_{2^n}p_{2^n}$ of 10; the mean square error E of the approximation is

$$\begin{aligned} E &= c_{j+1}^2 + c_{j+2}^2 + \ldots + c_{2^n}^2 \\ &= \sum_{k=1}^{2^n} y_k^2 - c_1^2 - c_2^2 - \ldots - c_j^2 \\ &= \sum_{k=1}^{2^n} y_k^2 - \sum_{k=1}^{2^n} \hat{y}_k^2 \end{aligned} \tag{12}$$

<u>Proof</u>: Squaring equation 10 for any given point k gives

$$y_k^2 = (c_1p_{1k} + c_2p_{2k} + \ldots + c_jp_{jk} + c_{j+1}c_{j+1k} + \ldots + c_{2^n}c_{2^n k})^2 \tag{13}$$

Squaring and summing 13 over all points gives

$$\sum_{k=1}^{2^n} y_k^2 = \sum_{k=1}^{2^n} \left(c_1p_{1k}\sum_{k=1}^{2^n} c_ip_{ik} + c_2p_{2k}\sum_{k=1}^{2^n} c_ip_{ik} + \ldots + c_{2^n}p_{2^n k}\sum_{k=1}^{2^n} c_ip_{ik}\right)$$

$$\sum_{k=1}^{2^n} c_1p_{1k}\sum_{k=1}^{2^n} c_ip_{ik} + \sum_{k=1}^{2^n} c_2p_{2k}\sum_{k=1}^{2^n} c_ip_{ik} + \ldots + \sum_{k=1}^{2^n} c_{2^n}p_{2^n k} = \sum_{k=1}^{2^n} c_ip_{ik}$$

Rearranging the summations gives

$$\sum_{k=1}^{2^n} y_k^2 = \sum_{k=1}^{2^n} c_1 c_i \sum_{k=1}^{2^n} p_{1k} p_{ik} + \sum_{k=1}^{2^n} c_2 c_i \sum_{k=1}^{2^n} p_{2k} p_{ik}$$

$$+ \ldots + \sum_{k=1}^{2^n} c_{2^n} c_i \sum_{k=1}^{2^n} p_{2^n k}\, p_{ik}$$

But

$$\sum_{k=1}^{2^n} p_{1k} p_{ik} = \begin{cases} 2^n & ; \; i=1 \\ 0 & ; \; i \neq 1 \end{cases} \quad , \text{etc.}$$

Thus,

$$\sum_{k=1}^{2^n} y_k^2 = c_1^2 + c_2^2 + \ldots + c_{2^n}^2 \tag{14}$$

In similar fashion

$$\hat{y}_k = (c_1 p_{1k} + c_2 p_{2k} + \ldots + c_j p_{jk})$$

Then

$$\sum_{k=1}^{2^n} \hat{y}_k^2 = c_1^2 + c_2^2 + \ldots + c_j^2 \tag{15}$$

Now

$$E = \sum_{k=1}^{2^n} (y_k - \hat{y}_k)^2$$

and from Theorem 6

$$E = \sum_{k=1}^{2^n} (c_1 p_{1k} + c_2 p_{2k} + \ldots + c_j p_{jk} + c_{j+1} p_{j+1k}$$

$$+\dots+c_{2^n}p_{2^n k} - c_1p_{1k} - c_2p_{2k} - \dots - c_jp_{jk})^2$$

$$= \sum_{k=1}^{2^n} (c_{j+1}p_{j+1k} + c_{j+2}p_{j+2k} + \dots + c_{2^n}p_{2^n k})^2$$

which proceeding as from 13 to 14 yields

$$E = c_{j+1}^2 + c_{j+2}^2 + \dots + c_{2^n}^2$$

and from 14 gives

$$E = \sum_{k=1}^{2^n} y_k^2 - c_1^2 - c_2^2 - \dots - c_j^2$$

and from 15 gives

$$E = \sum_{k=1}^{2^n} y_k^2 - \sum_{k=1}^{2^n} \hat{y}_k^2$$

Theorems 6 and 7 show that a complete function of binary variables can be approximated by a least squares best fit by writing the function as a polynomial in $\{v_j\}$ and dropping terms. The approximation error can be readily found by equation 12.

Although Theorem 6 gives a least square best fit for polynomials in $\{v_j\}$ the results can be extended to certain important cases of polynomials of any two-valued variables by Theorem 8.

Definition 9:

A real polynomial of two-valued variables $\{x_j\}$ is of order o if there are no terms of the polynomial involving more than o variables of $\{x_j\}$.

Theorem 8: Given a finite complete function of two-valued variables $\{x_j\}$ the least squares best fitting polynomial of a form allowing all terms of order less than o and any or all of the terms of order o may be found by finding the least squares best fitting polynomial in $\{v_j\}$ with the corresponding terms (i.e. allowing all terms of order less than o and the same terms of order o with x_j replaced by v_j) and applying the translation

$$x_j = \frac{1}{2}[v_j(x_{2j}-x_{1j}) + x_{1j} + x_{2j}] \tag{16}$$

Proof: Since the transformation 16 is a simple translation and constant expansion, every term of order o in the approximating polynomial in $\{v_j\}$ will give no terms higher than o in the transformed polynomial and only these terms of order o in $\{x_j\}$ corresponding to the terms of order o in $\{v_j\}$. Therefore, the transformation of the approximating polynomial in $\{v_j\}$ allowing a set of terms as described in the theorem will give an approximating polynomial in $\{x_j\}$ allowing the corresponding terms.

The proof goes now by contradiction. Assume that there exists an approximating polynomial P_x in $\{x_j\}$ allowing a given set of terms of order o but none higher that is a better least squares fit than found by the procedure of the theorem. Then the P_x can be transformed to a polynomial P_v in $\{v_j\}$ by the transformation

$$v_j = \frac{2x_j - x_{1j} - x_{2j}}{x_{2j} - x_{1j}} \tag{17}$$

P_v will have no terms of order greater than o and only terms of order o corresponding to terms of order o in P_x. But the approximation in $\{v_j\}$ of the theorem is the best least squares fit in those terms, therefore, P_v cannot be better, contradicting the assumption of this proof.

The type of truncation implied by Theorem 8 is very useful since well-behaved functions tend to have the smallest coefficients on the highest order terms.

G. Incomplete Functions

The discussion up to now has concerned complete functions, that is, functions that are defined for every possible combination of variables. A set of n two-valued variables can take on exactly 2^n possible combinations or points in n-space so that a domain is inherently implied. However, a particular function may simply be not defined for some of these possible points. Table 4 shows a partially specified or incomplete function of 3 variables z_1,z_2,z_3 for which only five of the eight points are defined.

Table 4. An incomplete function of z_1,z_2,z_3

z_3	z_2	z_1	$f(z_1,z_2,z_3)$
0	0	0	y_0
0	1	0	y_2
0	1	1	y_3
1	0	0	y_4
1	1	1	y_7

There are an infinite number of polynomials in z_1,z_2,z_3 that will pass through these points each giving a different set of values to the undefined points. Thus, there is no satisfactory single polynomial that can properly represent such a function. There are a couple of polynomials that are of interest related to this problem. One is a simple polynomial passing through the points and the other is the least squares best fitting polynomial.

A simple polynomial can be written for an incomplete function that is independent of the undefined points. This is shown in general in Appendix C but a simple illustration will suffice to demonstrate what happens.

Consider the three variable function table of Table 5. All function values at defined points are symbolized by y and at undefined points by u.

Table 5. A complete table of an incomplete function

z_3	z_2	z_1	$f(z_1,z_2,z_3)$
0	0	0	y_0
0	0	1	u_1 (undefined)
0	1	0	y_2
0	1	1	y_3
1	0	0	y_4
1	0	1	u_5 (undefined)
1	1	0	u_6 (undefined)
1	1	1	y_7

A function in z and $\overline{z}$ can be written including the undefined points as follows

$$f(z_1,z_2,z_3) = y_0\overline{z}_1\overline{z}_2\overline{z}_3 + u_1z_1\overline{z}_2\overline{z}_3 + y_2\overline{z}_1z_2\overline{z}_3 + y_3z_1z_2\overline{z}_3$$
$$+ y_4\overline{z}_1\overline{z}_2z_3 + u_5z_1\overline{z}_2z_3 + u_6\overline{z}_1z_2z_3 + y_7z_1z_2z_3$$

Substituting $\overline{z}_j = 1-z_j$ and grouping terms gives

$$
\begin{aligned}
f(z_1 z_2 z_3) = {} & y_0 + z_1(u_1-y_0) + z_2(y_2-y_0) + z_3(y_4-y_0) \\
& + z_1 z_2(y_3-y_2-u_1\ y_0) + z_1 z_3(u_5-y_4-u_1\ y_0) \qquad (18) \\
& + z_2 z_3(u_6-y_4-y_2\ y_0) + z_1 z_2 z_3(y_7-u_6-u_5\ y_4\ y_2\ u_1-y_0)
\end{aligned}
$$

If the undefined points are treated as "prohibited" or not allowed then the following is true:

The z_1 term exists if and only if the z_1z_2, z_1z_3, or $z_1z_2z_3$ and z_1z_2 and z_1z_3 terms exist. Thus, the coefficient of the z_1 term can be dropped if it is added to the z_1z_2 and z_1z_3 terms and subtracted from the $z_1z_2z_3$ term.

The z_1z_3 term exists if and only if the $z_1z_2z_3$ term exists, thus, the coefficient of z_1z_3 can be added to the coefficient of $z_1z_2z_3$ and the z_1z_3 term dropped. The z_2z_3 term can be handled similarly.

Performing these groupings on 18 gives

$$
\begin{aligned}
f(z_1,z_2,z_3) = {} & y_0 + z_2(y_2-y_0) + z_3(y_4-y_0) + z_1 z_2[y_3-y_2-u_1 \\
& + y_0 + (u_1-y_0)] + z_1 z_2 z_3[y_7-u_6-u_5-y_3+y_4+y_2+u_1 \\
& - y_0 + (u_6-y_4-y_2\ y_0) + (u_5-y_4-u_1\ y_0) + (u_1-y_0) \\
& - (u_1-y_0)]
\end{aligned}
$$

which reduces to

$$
\begin{aligned}
f(z_1,z_2,z_3) = {} & y_0 + z_2(y_2-y_0) + z_3(y_4-y_0) + z_1 z_2(y_3-y_2) \\
& + z_1 z_2 z_3(y_7-y_3-y_4 + y_0)
\end{aligned}
\qquad (19)
$$

The function 19 passes through the defined points of Table 5 and the coefficients are independent of the undefined function values. Of course if any point of undefined value is substituted in 19 some definite number will result but this number if a function of the function values at defined points.

The next theorem is a significant property of this simple function.

Theorem 9: Given any incomplete function of n two-valued variables defined at n points; one and only one polynomial in $\{z_j\}$ of the form

$$P_s\{z_j\} = c_1p_1 + c_2p_2 + \ldots + c_ip_i \tag{20}$$

passing through the defined points, can be found. The $\{c_i\}$ in equation 20 are constant coefficients and $\{p_i\}$ are found from

$$p_i = \prod_{j=1}^{n} (z_j)^{z_{jk}}$$

where the i index represents the i th defined point and z_{jk} is the value of z_j for the i th defined point. Thus, P_s is a polynomial that has, in general, n terms.

Proof: The existance of 20 is shown in general in Appendix C and the example above of Table 5. The uniqueness of 20 is shown in general in Appendix D and is illustrated by the following example using Table 5.

Any polynomial in z_1 represents uniquely (from Theorem 3) some complete function say that of Table 5. This function can be written as

$$\begin{aligned} P_s(z_1,z_2,z_3) = {} & y_0\bar{z}_1\bar{z}_2\bar{z}_3 + u_1z_1\bar{z}_2\bar{z}_3 + y_2\bar{z}_1z_2\bar{z}_3 + y_3z_1z_2\bar{z}_3 \\ & + y_4\bar{z}_1\bar{z}_2z_3 + u_5z_1\bar{z}_2z_3 + u_6\bar{z}_1z_2z_3 + y_7z_1z_2z_3 \end{aligned} \tag{21}$$

and also from 20

$$P_s(z_1,z_2,z_3) = c_1 + c_2z_2 + c_3z_3 + c_4z_1z_2 + c_5z_1z_2z_3 \tag{22}$$

Substituting $\bar{z}_j = 1-z_j$ in 21 gives

$$P_s(z_1,z_2,z_3) = y_0 + z_1(u_1-y_0) + z_2(y_2-y_0) + z_3(y_4-y_0)$$
$$+ z_1z_2(y_3-y_2-u_1+y_0) + z_1z_3(u_5-y_4-u_1+y_0)$$
$$+ z_2z_3(u_6-y_4-y_2+y_0) + z_1z_2z_3(y_7-u_6-u_5-y_3$$
$$+ y_4+y_2+u_1-y_0) \qquad (23)$$

Since 22 and 23 are identities then they may be equated term by term.

$$c_1 = y_0$$
$$0 = u_1-y_0 \quad ; \quad u_1=y_0$$
$$c_2 = y_2-y_0$$
$$c_3 = y_4-y_0$$
$$c_4 = y_3-y_2-y_1+y_0$$
$$0 = u_5-y_4-u_1+y_0 = u_5-y_4-y_0+y_0 = u_5-y_4 \quad ; \quad u_5=y_4$$
$$0 = u_6-y_4-y_2+y_0 \quad ; \quad u_6=y_4+y_2-y_0$$
$$c_5 = y_7-u_6-u_5-y_3+y_4+y_2+u_1-y_0$$
$$= y_7-y_4-y_2+y_0-y_4-y_3+y_4+y_2+y_0-y_0$$
$$=y_7-y_4-y_3+y_0$$

Thus $P_s(z_1,z_2,z_3)$ describes the complete function of Table 6. Every complete function value of $P_s(z_1,z_2,z_3)$ is uniquely determined. Therefore, $P_s(z_1,z_2,z_3)$ is unique from the uniqueness of polynomials of complete functions.

Table 6. Example of a simple polynomial P_s

z_3	z_2	z_1	$P_s(z_1,z_2,z_3)$
0	0	0	y_0
0	0	1	y_0
0	1	0	y_2
0	1	1	y_3
1	0	0	y_4
1	0	1	y_4
1	1	0	$y_4+y_2-y_0$
1	1	1	y_7

This means that every other polynomial passing through the defined points must have in one or more of its non-zero terms a product

$$p_L = \prod_{j=1}^{n} (z_j)^{z_{jL}}$$

where L is an index corresponding to an undefined point of the incomplete function.

Another important polynomial relating to an incomplete function is the least squares best fitting approximation. An incomplete function can be thought of as a function defined on all of a set of reduced points. In general there is no orthogonal representation of the reduced points, therefore, there is generally no polynomial that can be written that can be reduced to a least squares best fit simply by dropping terms. Thus, finding a least squares best fit becomes a far more difficult task and translating to $\{v_j\}$ has no great advantage. The general approach to finding the least squares best fit is outlined below for generalized variables $\{x_j\}$.

Given a finite incomplete function of two-valued variables $\{x_j\}$ defined for q points the least squares best fit of the form

$$\hat{y}_k = c_1 p_1 + c_2 p_2 + \ldots + c_m p_m$$

where $p_1, p_2, \ldots, p_m$ are the allowable product terms (including possibly a constant) of $\{x_j\}$ may be found as follows

$$E = \sum_{k=1}^{q} (y_k - \hat{y}_k)^2$$

$$= \sum_{k=1}^{q} (y_k - c_1 p_{1k} - c_2 p_{2k} - \ldots - c_m p_{mk})^2$$

Differentiating with respect to $c_1, c_2, \ldots c_m$ and setting the results equal to zero yields the system of $L = 1,2,\ldots,m$ equations in m unknowns of the form

$$c_1 \sum_{k=1}^{q} x_{1k} x_{Lk} + c_2 \sum_{k=1}^{q} x_{2k} x_{Lk} + \ldots + c_m \sum_{k=1}^{q} x_{mk} x_{Lk} = \sum_{k=1}^{q} x_{Lk} y_k$$

This system can usually be solved for $c_1, c_2, \ldots, c_m$ to give the least squares polynomial coefficients desired. Note that to find a best 10 term polynomial requires solving a system of 10 linear equations. This processing is fairly amenable to a digital computer solution. In practice it turns out that using a system of variables $\{z_j\}$ tends to give a fairly simple set of numbers for the system of equations.

The preceding material provides a firm foundation for the applications that follow. The most significant conclusions are:

1) Any complete finite function of two-valued variables can be uniquely written as a polynomial in any set of two-valued variables.

2) The polynomial coefficients may be readily determined by the methods of Theorem 3 or equations 5 and 6 then transformed to any set of two-valued variables.

3) The orthogonality of the variable $\{v_j\}$ leads to finding least squares best fitting polynomials of complete functions.

4) Any incomplete function can be written uniquely in the form of Theorem 9.

III. LOGIC WITH REAL POLYNOMIALS

A. Representation of Some Common Logic Operations

It has been shown that the truth table or logic function of Boolean algebra can be represented as a binary function of binary variables. A compilation of some common logic operations represented as polynomials in the variables $\{z_j\}$ and $\{v_j\}$ follow. $\{w_j\}$ is used as the logical symbol. $f\{v_j\}$ has value + 1 for $f\{w_j\} = 1$ and -1 for $f\{w_j\} = 0$ and $f\{z_j\}$ has value + 1 for $f\{w_j\} = 1$ and 0 for $f\{w_j\} = 0$.

1. Logical product $w_1 . w_2$ (Table 7)

$$f_z(z_1,z_2) = z_1 z_2$$
$$f_v(v_1,v_2) = \frac{1}{2}(v_1 + v_2 + v_1 v_2 - 1)$$

Table 7. Logical product truth table

w_2	w_1	$f(w_1,w_2)$
0	0	0
0	1	0
1	0	0
1	1	1

2. Logical inclusive OR $w_1 \vee w_2$ (Table 8)

$$f_z(z_1,z_2) = z_1 + z_2 - z_1 z_2$$
$$f_v(v_1,v_2) = \frac{1}{2}(1 + v_1 + v_2 - v_1 v_2)$$

Table 8. Logical inclusive OR truth table

w_2	w_1	$f(w_1,w_2)$
0	0	0

Table 8. Continued

w_2	w_1	$f(w_1,w_2)$
0	1	1
1	0	1
1	1	1

3. Logical exclusive OR $w_1 \oplus w_2$ (Table 9)

$$f_z(z_1,z_2) = z_1 + z_2 - 2z_1z_2$$

$$f_v(v_1,v_2) = - v_1v_2$$

Table 9. Logical exclusive OR truth table

w_2	w_1	$f(w_1,w_2)$
0	0	0
0	1	1
1	0	1
1	1	0

4. Negation $\overline{w}$ (Table 10)

$$f(z) = \overline{z} = 1-z$$

$$f(v) = \overline{v} = -v$$

Table 10. Negation truth table

w	$f(w) = \overline{w}$
0	1
1	0

5. 3-term majority logic $w_1\#w_2\#w_3$ (Table 11)

$$f_z(z_1,z_2,z_3) = z_1z_2 + z_1z_3 + z_2z_3 - 2z_1z_2z_3$$

$$f_v(v_1,v_2,v_3) = \frac{1}{2}(v_1 + v_2 + v_3 - v_1v_2v_3)$$

Table 11. Majority logic truth table

w_3	w_2	w_1	$f(w_1,w_2,w_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

6. General odd parity function $w_1 \oplus w_2 \oplus w_3 \oplus \ldots \oplus w_n$

$f\{w_j\} = 1$ at point k if $\{w_{jk}\}$ contains an odd number of 1's and $f\{w_j\} = 0$ if $\{w_{jk}\}$ contains an even number of 1's.

$$f\{v_j\} = -v_1v_2\ldots v_n$$

$$f\{z_j\} = \text{mod } 2 \sum_{j=1}^{n} z_j$$

7. General logical product $w_1 \cdot w_2 \cdot w_3 \cdot \ \ldots \ \cdot w_n$

$f\{w_j\} = 1$ if $w_1,w_2,\ldots,w_n$ are all 1 and $f\{w_j\} = 0$ otherwise.

$$f\{z_j\} = z_1z_2\ldots z_n$$

The last two logic operations are of particular interest because they give a logic interpretation to the real products in $\{v_j\}$ and $\{z_j\}$.

The functions above are written so that the two values of the functions are the same as the values of the dependent variables and have a direct correspondence to the logic variable and functions. Given a function f_1 of two other logic functions f_2 and f_3

$$f_1(f_2,f_3)$$

and given the corresponding polynomials as above in z and v; $f_{z_1}, f_{z_2}, f_{z_3}$ and $f_{v_1}, f_{v_2}, f_{v_3}$, then

$$f_{z_1}(f_{z_2}, f_{z_3})$$

and

$$f_{v_1}(f_{v_2}, f_{v_3})$$

will correspond to

$$f_1(f_2,f_3)$$

such that when $f_1(f_2,f_3) = 0$, $f_{z_1}(f_{z_2}, f_{z_3})$ will equal 0 and $f_{v_1}(f_{v_2}, f_{v_3})$ will equal -1, and when $f_1(f_2,f_3) = 1$, $f_{z_1}(f_{z_2}, f_{z_3})$ will equal 1 and $f_{v_1}(f_{v_2}, f_{v_3})$ will equal +1.

B. Proofs of Theorems of Logic

The capability of replacing logic operations with polynomial functions leads directly to proof of Boolean logic theorems in terms of real variables. By applying more mathematical rigor to this discussion it should be possible to satisfactorily develop the basis and properties of Boolean algebra from real variable theory. It is not the intent of this paper to go that far. However, some samples of proofs are of interest. A couple of theorems on powers of the variables $\{z_j\}$ and $\{v_j\}$ are needed first.

<u>Theorem 10</u>: $(z_j)^n = z_j$; n 1,2,...

<u>Proof</u>: From definition 6 $z_j^1=0$ and $z_j^2=1$. Thus

$$(z_j^1)^n = 0 = z_j^1$$

and

$$(z_j^2)^n = 1 = (z_j)^n \quad .$$

Since this exhausts all possible values of z_j the theorem follows.

<u>Theorem 11</u>: $(v_j)^{2n} = 1$ and $(v_j)^{2n+1} = v_j$; $n = 1,2,\ldots$

<u>Proof</u>: From definition 8 $v_j^1 = -1$ and $v_j^2 = +1$.

$$(v_j^1)^{2n} = (-1)^{2n} = 1$$

and

$$(v_j^2)^{2n} = (\ 1)^{2n} = 1 \quad .$$

Since this exhausts all possible values of v_j then $(v_j)^{2n} = 1$.

$$(v_j^1)^{2n+1} = (-1)^{2n+1} = -1 = v_j^1$$

and

$$(v_j^2)^{2n+1} = (+1)^{2n+1} = +1 = v_j^2 \quad .$$

Thus, $(v_j)^{2n+1} = v_j$.

The following are some fundamental (5) theorems of Boolean algebra.

1. Theorems on complementation

a. $w_1 \vee \overline{w}_1 = 1$

Writing the left hand side in terms of z_1 and $\overline{z}_1$ gives

$$z_1 + \overline{z}_1 - z_1\overline{z}_1 \quad .$$

Substituting $\overline{z}_1 = 1-z_1$ gives

$$z_1 + (1-z_1) - z_1(1-z_1)$$

which equals

$$z_1 + 1 - z_1 - z_1 + (z_1)^2$$

Since $(z_1)^2 = z_1$ we have

$$z_1 + 1 - z_1 - z_1 + z_1 = 1 \quad .$$

Q.E.D.

b. $w_1 . \overline{w}_1 = 0$

Writing $w_1 . \overline{w}_1$ as a polynomial in z_1 gives

$$z_1 \overline{z}_1 = z_1(1-z_1) = z_1 - z_1^2 = 0 \quad .$$

Q.E.D.

2. Theorem on double negation $\overline{\overline{w}}_1 = w_1$

Writing in terms of z_1 gives

$$(\overline{\overline{z}}_1) = (\overline{1-z_1}) = 1-(1-z_1) = z_1$$

3. DeMorgan's theorems

a. $\overline{w_1 \vee w_2} = \overline{w}_1 . \overline{w}_2$

Writing the left hand side in terms of z_1, z_2 gives

$$\begin{aligned} (\overline{z_1 + z_2 - z_1 z_2}) &= 1 - (z_1 + z_2 - z_1 z_2) \\ &= 1 - z_1 - z_2 + z_1 z_2 \\ &= (1-z_1)(1-z_2) \\ &= \overline{z}_1 \overline{z}_2 \end{aligned}$$

and $\overline{z}_1 \overline{z}_2$ corresponds to $\overline{w}_1 . \overline{w}_2$.

b. $\overline{w}_1 \vee \overline{w}_2 = \overline{w_1 . w_2}$

Writing the left hand side as a polynomial in z_1, z_2, z_3 gives

$$\begin{aligned} \overline{z}_1 + \overline{z}_2 - \overline{z}_1 \overline{z}_2 &= (1-z_1) + (1-z_2) - (1-z_1)(1-z_2) \\ &= 1 - z_1 + 1 - z_2 - 1 + z_1 + z_2 - z_1 z_2 \\ &= 1 - z_1 z_2 \\ &= \overline{z_1 z_2} \end{aligned}$$

4. Theorem on distribution $(w_1 \vee w_2).(w_1 \vee w_3) = w_1 \vee w_2.w_3$

Writing the left hand side as a polynomial in z_1, z_2, z_3 gives

$$(z_1 + z_2 - z_1z_2)(z_1 + z_3 - z_1z_3) = z_1^2 + z_1z_2 - z_1^2z_2 + z_1z_3$$

which equals

$$z_1^2 + z_1z_2 - z_1^2z_2 + z_1z_3 + z_2z_3 - z_1z_2z_3 - z_1^2z_3 - z_1z_2z_3 + z_1^2z_2z_3$$

which reduces to

$$z_1 + z_1z_2 - z_1z_2 + z_1z_3 + z_2z_3 - 2z_1z_2z_3 - z_1z_3 + z_1z_2z_3$$

which equals

$$z_1 + z_2z_3 - z_1z_2z_3 = z_1 + z_2z_3 - z_1(z_2z_3)$$

which can be written as the Boolean equation $w_1 \vee w_2.w_3$.

C. Non-negation Logic

The polynomial representation of any Boolean function can be readily used directly to develop a logic circuit. Note that the negations of the variables are not necessary in such a system whereas the usual Boolean logic expressions involve negation.

A circuit for providing a logical function of several logical variables can be most easily seen from the polynomial in $\{z_j\}$. Such a function

$$f\{z_j\} = c_1p_1 + c_2p_2 + \ldots + c_np_n$$

can be instrumented by a logic circuit that will accept both positive and negative weighted inputs for the positive and negative coefficients. It should be a type of threshold circuit that will give an output voltage corresponding to a logical 1 when the sum of the inputs (sign considered) are greater than any number corresponding to the voltage value (loading considered) between logical 0 and 1. The inputs can be made up by generating the p terms and using appropriate weighting resistors. The p terms can be generated readily by standard logical AND circuits since the logical product and real product are of the same form.

Such a logic system has some rather interesting properties. Since the sum of the terms nominally takes on only one of two values the peak summing values are well controlled. Since there is a distinct difference between the two sums the actual threshold of the summing-switching circuit is noncritical. Another interesting property is that some of the weighted terms can be quite far off from the nominal or missing altogether with a high probability of the circuit giving the correct output. It is really only necessary that the logic function be linearly separable (see part IV) in the remaining p terms of the polynomial. Thus, logic circuits can be made up using ANDed terms of the input variables and a weighted summation into a threshold-switching circuit with a high probability of giving the proper output even with failure of one or more parts of the circuit. This could be an important advantage in digital processors with extreme reliability requirements. In general, the polynomial instrumentation will be more complex than normal Boolean logic instrumentation although not requiring negated variables can be a simplifying factor in many cases.

A very interesting property of the $\{z_j\}$ polynomial form of a Boolean function that may have value in some cases is that the real summation of terms may be replaced by modulo-2 summation.

<u>Theorem 12</u>: Given any complete function of two-valued variables written as a polynomial in $\{z_j\}$ with integer function values, all coefficients of the polynomial will be integers.

Proof: Appendix C show that the coefficients of the polynomial in z are found by sums and differences of function values. If the function values are integers then the coefficients must be integers.

Theorem 13: Given any two-valued complete function of two-valued variables with function value 0 and 1 and written as a polynomial in $\{z_j\}$, the polynomial may be reduced to the modulo-2 sum only of the terms with odd integer coefficients and those coefficients set equal to one.

Proof: From Theorem 12 all coefficients must be integers. Since the p terms are all either zero or one, the sum and difference of coefficients will add to only one of zero or one. Since zero is even and one is odd it is necessary only to determine if the sum and difference of the coefficients as specified by the p terms is even or odd which can be found by a modulo-2 summation of the terms. Since an even number is zero in modulo-2 addition and an odd number is one in modulo-2 addition all terms with even coefficients may be eliminated and odd valued coefficients may be reduced to $+1$.

Theorem 13 leads to a fairly simple polynomial representation of the Boolean function except that modulo-2 addition (which is the logical odd-parity function) is not too easy to instrument as a parallel operation (16,18) although it is quite simple as a serial operation (16). It is significant that the terms still do not involve negated variables. This discussion leads to the following theorem in logic terms.

Theorem 14: Any Boolean function can be written as an odd-parity function of some set of products of the non-negated variables.

It seems likely that this function is unique but the proof is not obvious.

Although other logic systems are possible, the two described here seem the most likely to have practical value. The first because it is potentially a high reliability system in spite of circuit failure and the other because of its potential simplicity in certain applications.

IV. WEIGHTED SUM DECISION LOGIC

A. Fundamental Theory

Definition 10: (22,23)

f is a linearly separated truth function (also called linearly separable function and linear-input function) in the Boolean variables $\{w_j\}$ if there exists a real polynomial with $\{w_j\}$ replaced by $\{z_j\}$ of the form

$$\lambda = a_1 z_1 + a_2 z_2 + \ldots + a_n z_n + b$$

where $a_1, a_2, \ldots, a_n, b$ are real numbers such that when $f(w_1, w_2, \ldots, w_n)$ is true (or 1) λ is positive and when $f(w_1, w_2, \ldots, w_n)$ is false (or 0) λ is negative.

This definition can be extended to nonlinear functions.

Definition 11:

f is a separable function in the logical products $q_1, q_2, \ldots, q_m$ of a set of Boolean variables $\{w_j\}$ if there exists a real polynomial in terms of the real product terms $p_1, p_2, \ldots, p_n$ of the real binary variable $\{z_j\}$ of the form

$$\lambda = a_1 p_1 + a_2 p_2 + \ldots + a_n p_n + b \tag{24}$$

where $a_1, a_2, \ldots, a_n, b$ are real numbers such that when $f(w_1, w_2, \ldots, w_n)$ is true (or 1) λ is positive and when $f(w_1, w_2, \ldots, w_n)$ is false (or 0) λ is negative.

There have been many papers (4,6,9,11,13,18,22,23,24,25) recently studying linearly separable functions. This interest comes about because such logic circuits as core logic, parametron logic, and transistor-resistor logic lend themselves naturally to instrumenting linearly separable functions.

Most papers are involved with either finding mathematical characteristics of linearly separable functions, finding methods of determining appropriate coefficient values, or algorithms for showing that a given function is linearly separable. Linear programming is commonly used and work by K. Fan (10) on linear inequalities is applicable.

Little work has been done on non-linear separability primarily because of the difficulty of analysis. The theory of real polynomials of binary variables can be useful in the study of non-linear separability in the case of completely specified Boolean function. In this case the polynomial 24 can be transformed to the set of variables $\{v_j\}$ and a non-linear separable function of any n of the p-terms can be treated as a linearly separable function. This is true because the $\{v_j\}$ and all possible products make up a complete orthogonal set such that no other variable of the form of a variable v_j can be orthogonal to any term of the set. Thus any n of the variables and p-terms form a basis for the complete orthogonal set.

Orthogonal polynomial theory is useful in separable functions because it allows a more rigorous definition of separability as in definitions 10 and 11 and because if can be used to allow certain non-linearly separable functions to be treated as linearly separable.

B. Hypothesis on Linear Separability

A number of authors have given techniques for proving that a given function is linearly separable and in some cases incidentally yielding the coefficients. Linear programming is used by some (9,12) and C. Gaston (13) presents a matrix reduction technique based on the work of K. Fan (10) and C. Chow (4). All such methods are quite laborious and usually involve some cut-and-try work.

The hypothesis below is not proved but is a straightforward method of showing linear separability in the case of a completely specified Boolean function.

<u>Hypothesis</u>:

Given a complete function f of n binary variables $\{v_j\}$, f is linearly separable if and only if the polynomial in $\{v_j\}$ of the form

$$c_1v_1 + c_2v_2 + \ldots + c_nv_n = \lambda$$

is such that $\lambda > b$ if f is true (or 1) and $\lambda < b$ if f is false (or 0), or $\lambda > b$ if f is false (or 0) and $\lambda < b$ if f is true (or 1), where $c_1, c_2, \ldots, c_n$ are the coefficients that would appear on the respective first order terms of the real polynomial f_v in $\{v_j\}$ corresponding to f such that when f = 1 then $f_v = 1$ and when f = 0 then $f_v = -1$, and b is some real number.

The sufficiency of the hypothesis is readily seen since the values of $c_1, c_2, \ldots, c_n$ can be calculated and the value of λ for each function point calculated. These λ's can be checked to see if the conditions of the hypothesis hold true.

The necessity, however, has been neither proved nor disproved. Counter examples are difficult to work with because of the difficulty in showing whether a function is actually linearly separable or not. The hypothesis can be readily proven for two variables by exhaustion but exhaustive analysis of even three variables is prohibitive due to the fact there are 256 possible Boolean functions of three variables. This number can be reduced by recognizing certain symmetries.

All linearly separable functions are Unate (22) and all Unate functions are linearly separable for three or less variables. But, Unate functions of more than three variables are not all linearly separable (25). Thus, in this area exhausting three variable functions is not really very convincing anyway.

A hypothesis without proof may seem out of order in a paper such as this but the hypothesis is definitely of use in many cases for finding suitable weighting functions and the proof can be left as an exercise for some later Graduate student.

V. WEIGHTED AND NON-WEIGHTED CODES

A. Some Characteristics of Weighted Codes

Definition 12:

A weighted code with weights $b_0, b_1, \ldots, b_n$ is a function f of two-valued variables such that the function values are the set of integers $0,1,2,\ldots,m$ and a linear polynomial P_w in $\{z_j\}$

$$P_w = b_0 + b_1 z_1 + b_2 z_2 + \ldots + b_n z_n$$

can be written such that $P_w = f$ for all defined points of f.

The weighted codes that have been most thoroughly studied are those for which the function values are the integers $0,1,2,\ldots,9$. Such codes are called binary coded decimal or BCD codes (1,26,29). R. Richards (26) gave a listing of 4-variable (or 4-bit) BCD codes found by trial and error. C. Weeg (29) showed that Richards had found the complete set of 4-bit BCD codes with positive weights and went on to give a complete list with both positive and negative weights showing Richards' work to be incomplete in that area. Weeg restricted the weights to be integers $-9 \leq I \leq 9$.

Theorem 15: A function f of n independent two-valued variables defined at the points $(1,0,0,\ldots,0),(0,1,0,\ldots,0),(0,0,1,0,\ldots,0),\ldots,$ $(0,0,\ldots,0,1)$ and $(0,0,\ldots,0)$ having function value $0,0,\ldots,m$ is a weighted code with integer weights $b_1, b_2, \ldots, b_n$ if and only if the m-term polynomial P_s as in Theorem 9 is of the form

$$P_s = b_0 + b_1 z_1 + b_2 z_2 + \ldots + b_n z_n \tag{25}$$

Proof: The sufficiency is obvious. The integer weights follow from Theorem 12. The necessity follows from the uniqueness of P_s. Since f is defined at the points where the variables are either all zero or only one variable is non-zero then P_s from Theorem 9 will contain the constant term and all first order terms plus a term for every other defined point of f. Since P_s is unique, then all other polynomials passing through the defined points must have one or more non-zero terms of products other than those in P_s. Since P_s contains the constant term and all first order terms of $\{z_j\}$ then all other polynomials passing through the defined points of f will have one or more non-zero terms of order higher than 1. Thus no polynomial passing through the defined points of f other than P_s can be of the form of 21. Q.E.D.

Theorem 15 gives a method for finding the weights of weighted codes if such weights exist and the restriction on having certain points defined is observed.

B. Decoding Non-weighted Codes

If the function is not of a weighted code form, real polynomials can be used to find a decoding function. Any polynomial passing through the defined points can be used although using judicious choice such as the simple polynomial P_s of Theorem 9 can aid in keeping the decoding simple. A simple example of non-weighted decoding is shown using Table 12. This code is a complete function shown with both the binary variables w_1, w_2, w_3 and the real variables v_1, v_2, v_3. It is of the type known as reflected binary and is useful in mechanical analog to digital converters.

Table 12. Reflected binary code.

w_3	w_2	w_1	v_3	v_2	v_1	f
0	0	0	-1	-1	-1	0

Table 12. Continued

w_3	w_2	w_1	v_3	v_2	v_1	f
0	0	1	-1	-1	1	1
0	1	1	-1	1	1	2
0	1	0	-1	1	-1	3
1	1	0	1	1	-1	4
1	1	1	1	1	1	5
1	0	1	1	-1	1	6
1	0	0	1	-1	-1	7

The system of equations 5 and 6 gives the polynomial

$$f(v_1,v_2,v_3) = \frac{1}{2}(7 + 4v_3 - 2v_2v_3 + v_1v_2v_3) \tag{26}$$

which can be transformed to a polynomial in z

$$f(z_1,z_2,z_3) = z_1 + 3z_2 + 7z_3 - 2z_1,z_2 - 2z_1z_3 - 6z_2z_3 + 4z_1z_2z_3 \tag{27}$$

$$= \frac{1}{2}\{7 + z_3[4 - z_2(2-z_1)]\} \quad . \tag{28}$$

The form of 26 is useful if parity functions of the variables are available (18) and the form of 28 can be useful in certain applications. 27 is more complicated than the others but can be instrumented by finding logical products and taking the weighted sum; both operations are quite straight forward.

VI. FUNCTIONAL DECODING

A. General Considerations

One of the most useful applications of real polynomials of binary functions is functional decoding. This can mean circuits with an analog output that is a function of a digital input or transformation to some discrete or digital form of a function of a digital input. The real polynomial theory above applys directly to such problems since the polynomial itself is a description of the decoding.

In this section it is common to talk about a sine function or square function. This can be interpreted to mean the function under consideration is the sine or square function of a variable that increases linearly down the function table. In examples both the function and linear interpretation denoted by g will be given.

B. Square Function

A simple example of functional decoding is the square function illustrated in Table 13.

Table 13. Complete three variable square function

g	w_3	w_2	w_1	$F(\ ) = g^2$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	4
3	0	1	1	9
4	1	0	0	16
5	1	0	1	25
6	1	1	0	36

Table 13. Continued

g	w_3	w_2	w_1	$F(\ ) = s^2$
7	1	1	1	49

Two real polynomials describing Table 13 are

$$f(v_1,v_2,v_3) = \frac{1}{2}(35+7v_1+14v_2+28v_3+2v_1v_2+4v_1v_3+8v_2v_3)$$

and

$$f(z_1,z_2,z_3) = z_1+4z_2+16z_3+4z_1z_2+8z_1z_3+16z_2z_3 \quad .$$

Thus the function can be instrumented to find an analog output by taking a weighted sum of the first order and second order logical products. The weighting and addition could also be done by digital operations and the result would be a digital number representing f. Note that no third order term appears in the functions of $\{z_j\}$ and $\{v_j\}$.

C. Theorem Relating Binary Power and Function Power

Theorem 16: Given any real ordinary polynomial f of order o in a set of variables $g_1,g_2,\ldots,g_m$ where $g_1,g_2,\ldots,g_m$ are all functions of a set of two-valued variables such that $\{g_i\}$ are each a weighted code with weights $b_o,b_1,b_2,\ldots,b_n$, then f can be written as a polynomial in $\{z_j\}$ of order no more than o.

Proof: The variable g_i can be written as

$$g_i = b_0 + b_1z_{1i} + b_2z_{2i} +\ldots+ b_nz_{ni}$$

The highest order terms of $f\{g_i\}$ contain at most o terms of $\{g_i\}$ and since $\{g_i\}$ are each linear in a set $\{z_{ji}\}$ the order of the highest possible term in $\{z_{ji}\}$ is o.

This theorem has application not only in fitting polynomials of weighted code function but in approximations. Note that transformation from $\{z_{ji}\}$ to any set of two-valued variables $\{x_{ji}\}$ does not increase the order o of the polynomial.

Corollary 1: Given a function f of a set of weighted code functions $\{g_i\}$ and an approximate polynomial in $\{g_i\}$ of order o there can be no better fit by any given criteria than the best fit polynomial in $\{z_{ji}\}$ of order o.

With a best fit criteria such as least squares the best fit in $\{z_{ji}\}$ of order o is usually far superior to the best fit in $\{g_i\}$ of order o.

From Corollary 1 the great usefulness of the orthogonal variables $\{v_j\}$ is obvious. For well behaved functions of ordinary real variables a better and better fit will be achieved as the order of an approximate polynomial is increased. The same must hold true for binary variables of the real variables if they are weighted code functions of binary variables.

An example of approximate fitting is given later. For incomplete functions the least squares analysis described previously can be used, however, a different method of treating incomplete functions is given next.

D. Segmented Approximation

Segmented approximation is the use of different polynomials to describe different parts of a given curve (15,28). This can be particularly useful in incomplete functions. Note that the function f of a weighted code variable g_i is complete in $\{z_{ji}\}$ only if g_i is complete thus taking on exactly 2^n values, n $1,2,\ldots$. This will certainly not always be the case. This problem can be handled in several ways:

1) Find the least squares best fit in the incomplete function by the method of Section II F.

2) Pick a convenient set of values to define the undefined points.

3) Partition the domain g into sets of points that are complete in $\{z_i\}$.

4) Combine 2) and 3).

Convenient ways of picking arbitrary points are to either extend the function f if it is known for other values (this is done in a later example) or use some form of extrapolation to find the additional points (15).

The example given in Table 14 is used to demonstrated partitioning.

Table 14. Incomplete three variable square function

g	z_3 w_3	z_2 w_2	z_1 w_1	$f(.) = g^2$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	4
3	0	1	1	9
4	1	0	0	16
5	1	0	1	25
6	1	1	0	36

g can be partitioned at the first four variables giving the complete function of Table 15.

Table 15. Table for f_1

g	z_3 w_3	z_2 w_2	z_1 w_1	$f_1(.) = g^2$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	4
3	0	1	1	9

$$f_1(z_1,z_2) = z_1 + 4z_2 + 4z_1z_2$$

From the remaining part of Table 14 $g = 4$ and $g = 5$ form the complete function given in Table 16.

Table 16. Table for f_2

g	z_3 w_3	z_2 w_2	z_1 w_1	$f_2(.) = g^2$
4	1	0	0	16
5	1	0	1	25

$$f_2(z_1) = 16 + 9z_1$$

The function for $g = 6$ is simply $f_3(z) = 36$. These functions can now be recombined by using the fact that $z_3 = 0$ for Table 15, $z_2 = 1$ and $z_3 = 0$ for Table 16 and $z_3 = 1$, $z_2 = 1$, and $z_1 = 0$ for the value at $g = 6$. The combination is as follows:

$$\begin{aligned} f(z_1,z_2,z_3) &= \bar{z}_3f_1(z_1,z_2) + z_3\bar{z}_2f_2(z_1) + z_3z_2\bar{z}_1f_3(z) \\ &= \bar{z}_3(z_1+4z_2+4z_1z_2) + z_3\bar{z}_2(16-9z_1) + 36z_3z_2\bar{z}_1 \\ &= z_1+4z_2+16z_3+4z_1z_2+8z_1z_3+16z_2z_3-49z_1z_2z_3 \quad . \end{aligned}$$

but $z_1z_2z_3$ is "not allowed" thus

$$f(z_1,z_2,z_3) = z_1+4z_2+16z_3+4z_1z_2+8z_1z_3+16z_2z_3 \quad (29)$$

The technique arriving at 29 is quite similar to the system of finding a polynomial in z and $\overline{z}$ described in Theorem 2. 29 is an exact fit but the functions f_1,f_2 and f_3 could be approximations in general.

The least squares best fit technique 1) is always better than these other approaches (at least in a least squares sense), however, in some cases the calculation of the least squares coefficients might be more expensive than the added complexity.

E. Interpolation

A function of binary variables can be interpreted to have useful meaning when the binary variables are allowed to take on values other than the two defined values for at least two kinds of weighted functions used as variables. An easily understandable variation occurs when binary variables are allowed to be continuous one at a time.

The first case to be examined is the ordinary binary function shown for three variables in Table 17. Extension to more variables is fairly obvious.

Table 17. Ordinary binary coded decimal code

z_3	z_2	z_1	ξ	$f(.)$
0	0	0	0	y_0
0	0	1	1	y_1
0	1	0	2	y_2
0	1	1	3	y_3
1	0	0	4	y_4

Table 17. Continued

z_3	z_2	z_1	g	$f(.)$
1	0	1	5	y_5
1	1	0	6	y_6
1	1	1	7	y_7

$$g = z_1 + 2z_2 + 4z_3$$

Let z_1 be a continuous variable. Then if z_2 and z_3 are held constant the function $f(z_1,z_2,z_3)$ will be a linear function in z_1 passing through the pair of points defined by the value of f corresponding to the value of z_2 and of z_3. If z_1 is allowed to take on the values $-\frac{1}{2} \leq z_1 \leq 1\frac{1}{2}$ the result is shown graphically in Figure 1. From this graph it can be seen that the variation of z_1 is a form of linear interpolation between defined points of the function. This interpolation can be used to give "finer grained" functions by replacing the variable z_1 by 2 or more binary variables that will give several points on the interpolating line. For example

$$z_1 = \frac{1}{4}[z_{11} + 2z_{12} + 4z_{13}]$$

will give 8 points on a line from $-\frac{1}{4}$ to $+ 1\frac{1}{2}$ if z_{11},z_{12},z_{13} are organized in sequence as ordinary binary variables of the form of Table 17. Note that this interpolation increases the number of terms of a $\{z_j\}$ polynomial representation but not the order of the polynomial since z_1 is linear in the added variables.

If many points of a well behaved function are taken the interpolation will be quite good. The slope of lines through adjacent points as described above will take on the value of the derivative between the points by the

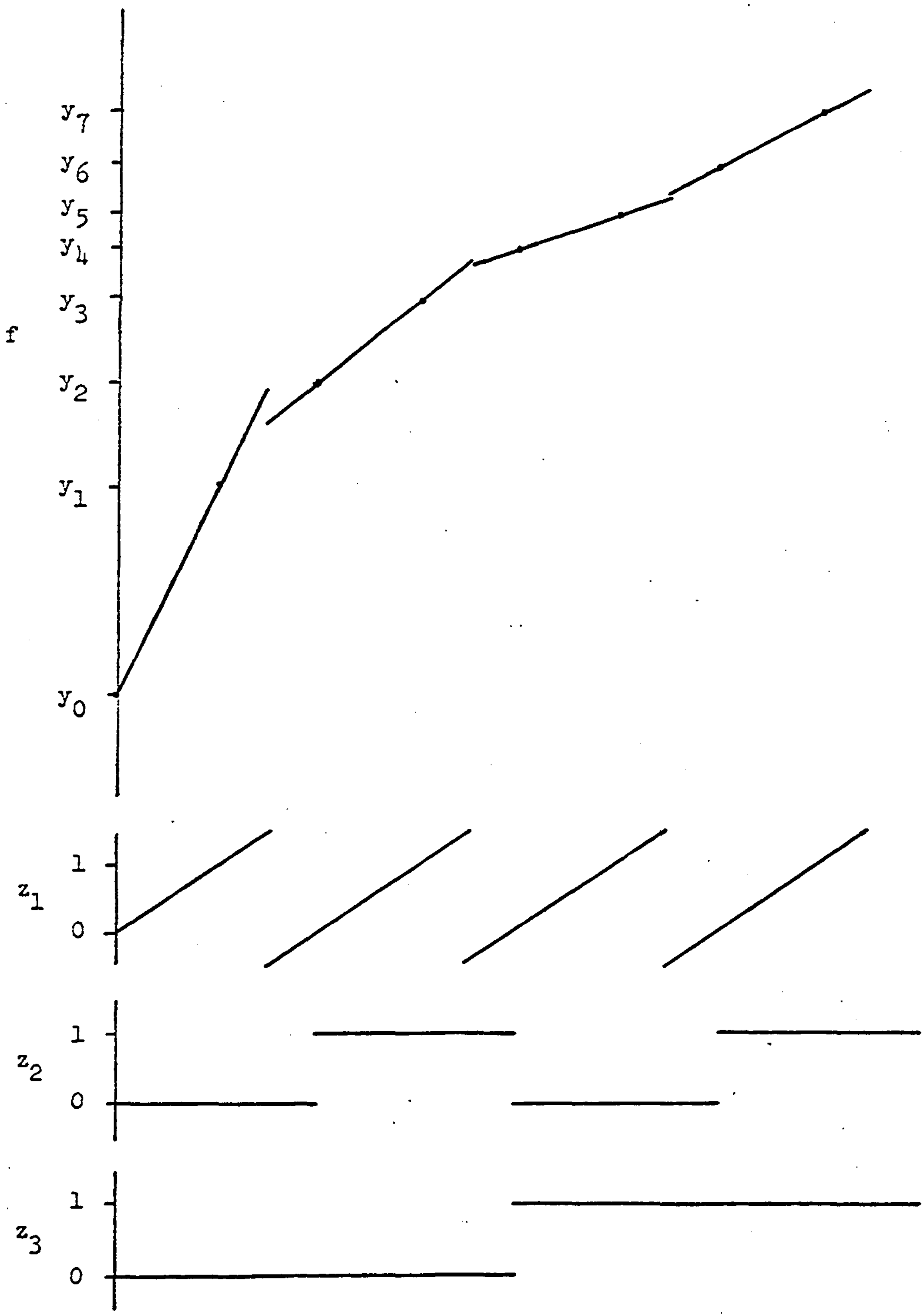

Figure 1. Interpolation example

law of the mean and as shorter and shorter line segments are taken, the derivatives of adjacent curve sections become more nearly equal.

Another weighted code system that lends itself to interpolation, in theory at least, is the reflected binary function of Table 18. Extension to more than three variables is obvious.

Table 18. General reflected binary code

g	z_3	z_2	z_1	f(.)
0	0	0	0	y_0
1	0	0	1	y_1
2	0	1	1	y_2
3	0	1	0	y_3
4	1	1	0	y_4
5	1	1	1	y_5
6	1	0	1	y_6
7	1	0	0	y_7

Reflected binary codes have the characteristic that in going from one integer of g to an adjacent one only one of the binary variables changes. If the domain z is visualized as a cube (or hypercube in general) this amounts to going from one vertex to an adjacent one. If at any point the variable that changes is caused to change continuously from 0 to 1 (or from 1 to 0) instead of discretely the result with respect to f will be a linear change in the function value from the value of the original vertex to the value of the next vertex. It must be linear because the function f is linear in any single variable of $\{z_j\}$ and it must pass through the adjacent function values by the nature of the binary polynomial.

This reflected binary example is interesting in theory but difficult to instrument. Some insight into the type of surface in n-space that is generated by treating all $\{z_j\}$ as continuous can now be seen. The intersection of the surface with any of the extended planes of the faces of the hypercube must be linear but the intersections with any other planes are in general non-linear.

F. Sine Function Example

A practical problem would be to generate the sine of an ordinary weighted binary function for angles from 0° to 90°. Such a function is given in Table 19 for 2° increments. The angle from 0° to 90° inclusive in 2° steps takes 46 points. By extending the definition to 92° and 94° 48 points are defined. This can then be broken down into a 32 point complete function in z_1, z_2, z_3, z_4, z_5 and a 16 point complete function in z_1, z_2, z_3, z_4. The least squares best polynomial P_1 of order 3 for f_1 is

$$\begin{aligned}
P_1(z_1,z_2,z_3,z_4,z_5) = {} & -0.0000918 + 0.0349875z_1 + 0.069890z_2 \\
& + 0.1393200z_3 + 0.2757713z_4 + 0.5300831z_5 \\
& - 0.0001975z_1z_2 - 0.0005775z_1z_3 \\
& - 0.001650z_1z_4 - 0.0057750z_1z_5 \\
& - 0.0012225z_2z_3 - 0.0035950z_2z_4 \\
& - 0.0121200z_2z_5 - 0.0083450z_3z_4 \\
& - 0.0265700z_3z_5 - 0.0627425z_4z_5 \\
& - 0.018380z_3z_4z_5 - 0.009170z_2z_4z_5 \\
& - 0.004585z_2z_3z_5 - 0.004600z_1z_4z_5 \\
& - 0.0022350z_2z_3z_4 - 0.0023150z_1z_3z_5 \\
& - 0.0010650z_1z_3z_4 - 0.0011350z_1z_2z_5
\end{aligned}$$

Table 19. Sine function example

angle °	$z_6z_5z_4z_3z_2z_1$	f_1 exact sine	P_1 approx. sine	error
0	000000	0.00000	0.000092	+0.000092
2	000001	0.03490	0.034896	+0.000004
4	000010	0.06976	0.069798	-0.000038
6	000011	0.10453	0.104588	-0.000058
8	000100	0.13917	0.139228	-0.000058
10	000101	0.1736	0.17364	-0.00004
12	000110	0.2079	0.20790	+0.00000
14	000111	0.2419	0.24181	+0.00009
16	001000	0.2756	0.27568	-0.00008
18	001001	0.3090	0.30902	-0.00002
20	001010	0.3420	0.34198	+0.00002
22	001011	0.3746	0.37453	+0.00007
24	001100	0.4067	0.40666	+0.00004
26	001101	0.4384	0.43835	+0.00005
f_1 28	001110	0.4695	0.46949	+0.00001
30	001111	0.5000	0.50010	-0.00010
32	010000	0.5299	0.52999	-0.00009
34	010001	0.5592	0.55920	-0.00000
36	010010	0.5878	0.58776	+0.00004
38	010011	0.6157	0.61564	+0.00007
40	010100	0.6428	0.64274	+0.00006
42	010101	0.6691	0.66906	+0.00004
44	010110	0.6947	0.69470	-0.00000
46	010111	0.7193	0.71939	-0.00009
48	011000	0.7431	0.74302	+0.00008
50	011001	0.7660	0.76598	+0.00002
52	011010	0.7880	0.78803	-0.00003
54	011011	0.8090	0.80907	-0.00007
56	011100	0.8290	0.82905	-0.00005
58	011101	0.8480	0.84805	-0.00005
60	011110	0.8660	0.86601	-0.00001
62	011111	0.8829	0.88280	+0.00001
64	100000	0.8988	0.89868	+0.00012
66	100001	0.9135	0.91352	-0.00002
68	100010	0.9272	0.92725	-0.00005
70	100011	0.9397	0.93975	-0.00005
72	100100	0.9511	0.95120	-0.00010
f_2 74	100101	0.9613	0.96130	+0.00000
76	100110	0.9703	0.97028	+0.00002
78	100111	0.9781	0.97802	+0.00008
80	101000	0.9848	0.98490	-0.00010
82	101001	0.9903	0.99030	-0.00000
84	101010	0.9945	0.99448	+0.00002
86	101011	0.9976	0.99752	+0.00008
88	101100	0.9994	0.99932	+0.00008

Table 19. Continued

angle °	$z_6z_5z_4z_3z_2z_1$	F_1 exact sine	F_1 approx. sine	error
90	101101	1.0000	0.99998	-0.00002
92	101110	0.9994	0.99940	+0.00000
94	101111	0.9976	0.99770	-0.00010

$$-0.0005050z_1z_2z_4 - 0.0003000z_1z_2z_3$$

and the least squares best fitting polynomial P_2 of order 2 for f_2 is

$$\begin{aligned} P_2(z_1,z_2,z_3,z_4) = {} & 0.898675 + 0.014850z_1 + 0.028575z_2 \\ & +0.052525z_3 + 0.086225z_4 + 0.002350z_1z_2 \\ & -0.004750z_1z_3 - 0.009450z_1z_4 \\ & -0.009500z_2z_3 - 0.019000z_2z_4 - 0.038100z_3z_4 \end{aligned}$$

and the approximating function P of f is

$$P(z_1,z_2,z_3,z_4,z_5,z_6) = \overline{z}_6P_1(z_1,z_2,z_3,z_4,z_5)+z_6\overline{z}_5P_2(z_1,z_2,z_3,z_4) .$$

P_1 contains 26 terms and P_2 contains 11 terms, therefore, a total of 37 weights are required. Since the sine is a smooth function and can be approximated quite accurately with linear interpolation of 4° segments, interpolation for z_1 is definitely possible. Using

$$z_1 = 1/64\ (z_{16}+2z_{15}+4z_{14}+8z_{13}+16z_{12}+32z_{11})$$

will give an angular resolution of 1/16 degree which is in the same order as the accuracy of the approximation P. Every term in which z_1 appeared would now have 6 terms giving a total of 127 weights required (one for each term of P_1 and P_2 containing a z_1 term), however, many of these new coefficients would be negligible reducing the total number of weights required to about 80. Thus, a weighted sum of logical products using about 80 weights (many of which could be fairly inaccurate) can provide a sine function output of an ordinary binary variable input with an accuracy of about 0.1%.

G. Applications

There are many applications of functional decoding, and a few of the more interesting ones are mentioned here.

A fairly common problem in radar systems or any system requiring processing of coordinate data from angle sensors is conversion from a digital shaft encoder (usually in reflected binary) to trionometric functions. Functional decoding is very convenient in defining a conversion circuit.

Although conversion to analog functions is the most obvious, conversion to a digital number can also be convenient. In the sine function example above, the 30 weights could be stored and the sine (in digital form) of a binary number could be found by programming the appropriate additions and subtractions. The important fact here is that only addition and subtraction are needed and not multiplication, division, or taking powers. This principle could be extended to any function.

An interesting possibility is that of using functional decoding to find an approximate product of two numbers. A direct approach or quarter-square multiplier approach might be used.

It is important to note that the real polynomial approach to functional decoding can be applied to functions of more than one variable. This might turn out to be one of its most useful properties. It is difficult to get analog functions of several variables by any technique and digital functions of several variables can require extensive storage or long calculation.

VII. PATTERN RECOGNITION

The Introduction points out that pattern recognition schemes have been extensively studied recently. Some problem areas are 1) lack of a good useable description of patterns, 2) graininess of the mathematical observation, and 3) inability (in most cases) to handle multitone patterns.

The polynomial of binary variables approach to functional decoding has interesting application to these problem areas. The degree of darkness can be treated as a function of the two (or possibly three) coordiante variables and can be found as a polynomial in binary coded coordinate variables. The most interesting polynomial to use is in $\{v_j\}$. Due to the orthogonality the contribution to the function of any term can be determined directly from the coefficient of the term. Thus the coefficients of the polynomials in $\{v_j\}$ for a set of patterns can be investigated and selected coefficients can be used as the "parameters" describing the pattern.

Note that multitone functions are readily treated. Graininess can be improved by considering the amount of dark area in a given area that provides a function point. Note also that three dimensional patterns can be handled in a similar manner.

These parameters can be analyzed and the decision as to pattern can be by either an adaptive system or direct statistical analysis. This type of problem can be handled very nicely by statistical analysis.

VIII. Conclusions and Summary

This dissertation has developed the idea of real polynomials of binary variables and developed and suggested several areas of application.

Several uses have been given in the areas where Boolean algebra is traditionally applied. The whole area of Boolean functional separability can possibly be treated from real variables and some approachs have been suggested.

The study of weighted codes seems to fall naturally into the category of real function analysis. The very definition of weighted codes is more meaningful in terms of real variables.

Functional decoding is another natural application. This decoding can be digital to analog or digital to digital and functions of several variables can be treated.

Pattern recognition is the least developed area of application given in this dissertation. This is not due to its lack of importance, on the contrary, it may be the most important area. However, an adequate analysis of this application would be a lengthy dissertation in itself and is beyond the scope of this dissertation.

Some specific applications developed are:

1) a different technique for proof of Boolean logic theorems and identities;

2) an inherently reliable logic system;

3) a simple non-negation logic system;

4) a hypothesis on linear separability of Boolean functions;

5) a representation of and a theorem on weighted codes;

6) design of decoding circuits for non-weighted codes;

7) techniques for circuits to give functions of binary variables;

8) a unique method of storing (digitally) function tables.

Some obvious extensions of this material are:

1) a complete analysis of application to pattern recognition;

2) develop Boolean algebra from real variable theory;

3) a more complete study of non-negation logic including circuits;

4) prove the hypothesis of part IV, section B and further extend the applications to separability theory;

5) develop further restriction on weighted codes and study characteristics of certain non-weighted codes;

6) extend functional decoding applications.

IX. LITERATURE CITED

1. Beyer, W. A. Uniqueness of weighted code representations. II. Institute of Electrical and Electronic Engineers Transactions on Electronic Computers EC-12: 137. 1963.

2. Bledsoe, W. W. and Browning, I. Pattern recognition and reading by machine. Eastern Joint Computer Conference Proceedings 16: 225-232. 1959.

3. Boole, George E. An investigation of the laws of thought. London, England, Macmillan and Co. 1854.

4. Chow, C. K. Boolean functions realizable with single threshold devices. Institute of Radio Engineers Proceedings 49: 370-371. 1961.

5. Chu, Yaohun. Digital computer design fundamentals. New York, N. Y., McGraw-Hill Book Co., Inc. 1962.

6. Coleman, Robert P. Orthogonal functions for the logical design of switching circuits. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 379-383. 1961.

7. Cox, Jerome R., Jr. A squaring analog-digital converter. Institute of Radio Engineers Transactions on Electroni Computers EC-9: 487-494. 1960.

8. David, H. T. 2^K factorials. Unpublished class notes for Statistics 402C, January 1963. Mimeographed. Ames, Iowa, Department of Statistics, Iowa State University of Science and Technology. 1963.

9. Einhorn, Sidney N. The use of the simplex algorithm in the mechanization of Boolean switching functions by means of magnetic cores. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 615-622. 1961.

10. Fan, M. On systems of linear inequalities. Annals of Mathematical Studies No. 39: 99-156. 1956.

11. Gableman, Irving J. The synthesis of Boolean functions using a single threshold element. Institute of Radio Engineers Transactions on Electronic Computers EC-11: 639-642. 1962.

12. Garvin, Walter W. Introduction to linear programming. New York, N. Y., McGraw-Hill Book Co., Inc. 1960.

13. Gaston, C. A. A simple test for linear separability. Institute of Electrical and Electronic Engineers Transactions on Electronic Computers EC-12: 134-135. 1963.

14. Highleyman, W. H. An analog method for character recognition. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 502-512. 1961.

15. Hildebrand, F. B. Introduction to numerical analysis. New York, N. Y. McGraw-Hill Book Co., Inc. 1956.

16. Hofheimer, R. W. and Perry, K. E. Digital analog function generators. (Lincoln Laboratory, Massachusetts Institute of Technology) Technical Report No. 162 on Contract AF19(122)458. 23 August 1957. Original not available; cited in U. S. Government Research Reports 31: 459. June 12, 1959.

17. Horwitz, L. D. and Shelton, G. L., Jr. Pattern recognition using autocorrelation. Institute of Radio Engineers Proceedings 49: 175-185. 1961.

18. Kautz, William H. The realization of symmetric switching functions with linear input logic elements. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 371-378. 1961.

19. Keller, H. B. Finite automata, pattern recognition and perceptions. Association of Computing Machines Journal 8: 1-20. 1961.

20. Marill, T. and Green, D. M. Statistical recognition functions and the design of pattern recognizers. Institute of Radio Engineers Transactions on Electronic Computers EC-9: 472-477. 1960.

21. Massachusetts Institute of Technology. Servomechanizms Laboratory. Notes on analog-digital conversion techniques. 2nd ed. Rev. New York, N. Y., Technology Press. 1958.

22. McNaughton, R. Unite truth functions. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 1-5. 1961.

23. Minnick, Robert C. Linear-input logic. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 6-16. 1961.

24. Muroga, S., Toda, I. and Takasu, S. Theory of majority decision elements. Franklin Institute Journal 271: 376-418. 1961.

25. Paull, M. C. and McCluskey, C. J., Jr. Boolean functions realizable with single threshold devices. Institute of Radio Engineers Proceedings 48: 1335-1337. 1960.

26. Richards, R. K. Arithmetic operations in digital computers. Princeton, N. J., D. Van Nostrand Co., Inc. 1955.

27. Shannon, C. E. Symbolic analysis of relay and switching networks. American Institute of Electrical Engineers Transactions 57: 713-723. 1938.

28. Sokolnikoff, I. S. and Redheffer, R. M. Mathematics of physics and modern engineering. New York, N. Y., McGraw-Hill Book Co., Inc. 1958.

29. Weeg, G. P. Uniqueness of weighted code representations. Institute of Radio Engineers Transactions on Electronic Computers EC-9: 487-489. 1960.

30. Whitehead, A. N. and Russell, B. Principia Mathematica. Vol. 1. Cambridge, England, Cambridge University Press. 1910.

31. Young, D. A. Automatic character recognition. Electronic Engineering 32: 2-10. 1960.

X. ACKNOWLEDGEMENTS

The writer wishes to express his appreciation to his major professor, Dr. A. V. Pohm, and to Dr. H. T. David for their review and comment.

The writer also wishes to thank Mrs. Skola for her typing and his wife for performing many calculations, for technical criticism and her editorial review of the text.

XI. APPENDIX A

A. Orthogonality of the Variables $\{v_j\}$

Given a set of n two-valued variables $\{v_j\}$ with value +1 and -1 consider the array of all the possible combinations $k=1,2,\ldots,2^n$ of these variables and all possible products of $\{v_j\}$. Lable in any order the variables $v_1,v_2,\ldots,v_n$ and the products by $p_1,p_2,\ldots,p_{2^n-1}$. The p-terms are mutually orthogonal such that

$$\sum_{k=1}^{2^n} p_{ik}p_{jk} = 0 \tag{30}$$

$$\sum_{k=1}^{2^n} p_{ik}p_{ik} = 2^n \tag{31}$$

and all orthogonal to any finite constant c giving

$$\sum_{k=1}^{2^n} cp_{ik} = 0 \tag{32}$$

<u>Proof</u>: The proof is by mathematical induction. The array for n=2 is given in Table 20.

Table 20. Variable table for v_1,v_2

		p_1	p_2	p_3
k	c	v_1	v_2	v_1v_2
1	c	-1	-1	1
2	c	-1	1	-1
3	c	1	-1	-1
4	c	1	1	1

It is readily seen that 30, 31, and 32 hold for Table 20. Note also that the sign of any or all columns may be changed and the resultant array still satisfies 30, 31, and 32, and further that

$$\sum_{k=1}^{2^n} p_{ik}(-p_{ik}) = -2^n$$

Now consider the $2^n \times 2^n$ array for n variables given in Table 21 where all the conditions given for Table 20 hold true.

Table 21. Variable table for $v_1 v_2 \ldots v_n$

k	c	p_1	p_2	...	p_{2^n-1}
1	c	v_{11}	v_{21}	...	$v_{11}v_{21}\ldots v_{n1}$
2	c	v_{12}	v_{22}	...	$v_{12}v_{22}\ldots v_{n2}$
3	c	v_{13}	v_{23}	...	$v_{13}v_{23}\ldots v_{n3}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
2^n	c	v_{12^n}	v_{22^n}	...	$v_{12^n}v_{22^n}\ldots v_{n2^n}$

If a variable v_{n+1} is added the new array will be of the form of Table 22. From 30 p_{2^n} is orthogonal to $p_1, p_2, \ldots, p_{2^n-1}$ since $\sum_{k=1}^{2^n} (+1)p_{ik} = 0$ and $\sum_{k=2^n+1}^{2^{n+1}} (-1)p_{ik} = 0$. Considering the rows $k=1,2,\ldots,2^n$ each column p_{2^n+i} is orthogonal to $c, p_1, p_2, \ldots, p_{2^{n+1}-1}$ except p_i and $p_{2^{n+1}+i}$ for which

$$\sum_{k=1}^{2^n} p_{ik} p_{2^n+i_k} = 2^n$$

and

$$\sum_{k=1}^{2^n} p_{2^n+i_k} p_{2^n+i_k} = 2^n$$

Similarly for $k=2^n+1, 2^n+2, \ldots, 2^{n+1}$ each column p_{2^n+i} is orthogonal to every $c, p_1, p_2, \ldots, p_{2^{n+1}-1}$ except p_i and p_{2^n+i} for which

$$\sum_{k=2^n+1}^{2^{n+1}} p_i p_{2^n+i} = -2^n$$

and

$$\sum_{k=2^n+1}^{2^{n+1}} p_{2^n+i} p_{2^n+i} = 2^n$$

Now

$$\sum_{k=1}^{2^{n+1}} p_i p_{2^n+i} = 2^n - 2^n = 0$$

and

$$\sum_{k=1}^{2^{n+1}} p_{2^n+i} p_{2^n+i} = 2^n + 2^n = 2^{n+1} \quad .$$

Also for $p_1, p_2, \ldots, p_{2^n-1}$

$$\sum_{k=1}^{2^{n+1}} p_i p_i = \sum_{k=1}^{2^n} p_i p_i + \sum_{k=2^n+1}^{2^{n+1}} p_i p_i = 2^n + 2^n = 2^{n+1} \quad .$$

Thus, the conditions of Table 20 are true for n+1 if true for n. Thus, 30, 31, and 32 hold for any number of variables $\{v_j\}$.

Table 22. Variable table for $v_1 v_2 \cdots v_n v_{n+1}$

k	p_{2^n}	c	p_1	p_2	$\cdots$	p_{2^n-1}	p_{2^n+1}	p_{2^n+2}	$\cdots$	$p_{2^{n+1}-1}$
1	+1	c	v_{11}	v_{21}	$\cdots$	$v_{11}v_{21}\cdots v_{n1}$	$+v_{11}$	$+v_{12}$	$\cdots$	$+v_{11}v_{21}\cdots v_{n1}$
2	+1	c	v_{12}	v_{22}	$\cdots$	$v_{12}v_{22}\cdots v_{n2}$	$+v_{12}$	$+v_{22}$	$\cdots$	$+v_{12}v_{22}\cdots v_{n2}$
3	+1	c	v_{13}	v_{23}	$\cdots$	$v_{13}v_{23}\cdots v_{n3}$	$+v_{13}$	$+v_{23}$	$\cdots$	$+v_{13}v_{23}\cdots v_{n3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^n	+1	c	v_{12^n}	v_{22^n}	$\cdots$	$v_{12^n}\, v_{22^n}\cdots v_{n2^n}$	$+v_{12^n}$	$+v_{22^n}$	$\cdots$	$+v_{12^n}\, v_{22^n}\cdots v_{n2^n}$
2^n+1	-1	c	v_{11}	v_{21}	$\cdots$	$v_{11}v_{21}\cdots v_{n1}$	$-v_{11}$	$-v_{21}$	$\cdots$	$-v_{11}v_{21}\cdots v_{n1}$
2^n+2	-1	c	v_{12}	v_{22}	$\cdots$	$v_{12}v_{22}\cdots v_{n2}$	$-v_{12}$	$-v_{22}$	$\cdots$	$-v_{12}v_{22}\cdots v_{n2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^{n+1}	-1	c	v_{12^n}	v_{22^n}	$\cdots$	$v_{12^n}\, v_{22^n}\cdots v_{n2^n}$	$-v_{12^n}$	v_{22^n}	$\cdots$	$-v_{12^n}\, v_{22^n}\cdots v_{n2^n}$

XII. APPENDIX B

A. Coefficients of a Polynomial in $\{v_j\}$

Given a finite function of n two-valued variables the function can be written as a polynomial in z and $\bar{z}$ by Theorems 2 and 4. If this is done the contribution to the function due to a coefficient y_k is

$$y_k \bar{z}_1 \bar{z}_2 \ldots \bar{z}_j z_{j+1} \ldots z_n \tag{33}$$

where z_1 thru z_j are 0 and z_{j+1} thru z_n are 1 for the row of the function table with function value y_k. This term of the function in $z, \bar{z}$ variables can be transformed to the function in variables $\{v_j\}$ by the transformations $z_j = \frac{1+v_j}{2}$ $z_j = \frac{1-v_j}{2}$. The y_k term in variables $\{v_j\}$ becomes

$$\frac{y_k}{2^n}(1-v_1)(1-v_2)\ldots(1-v_j)(1\ v_{j\ 1})\ldots(1\ v_n)\ . \tag{34}$$

Note that in a function table in $\{v_j\}$ the row of variables corresponding to the function value y_k is -1 for v_1 thru v_j and +1 for v_{j+1} thru v_n. Expanding 34 each term of the polynomial in $\{v_j\}$ of the function appears once and only once and has a coefficient $+\frac{y_k}{2^n}$ if the product of variables $\{v_j\}$appearing in a given term is positive for the row corresponding to function value y_k and the coefficient is $-\frac{y_k}{2^n}$ otherwise.

Since the polynomial in z, $\bar{z}$ is made up of 2^n terms of the form 33 then the polynomial in $\{v_j\}$ will be the sum of 2^n terms of the form 34. Adding the contribution to a v polynomial term from each term of the form 34 results in the following equations for the coefficients $c_{i,j,\ldots,m}$ of the $v_i v_j \ldots v_m$ term of the v matrix

$$c_0 = \frac{1}{2^n} \sum_{k=1}^{2^n} y_k$$

$$c_i = \frac{1}{2^n} \sum_{k=1}^{2^n} v_{ik}\, v_k$$

$$c_{ij,\ldots,m} = \frac{1}{2^n} \sum_{k=1}^{2^n} v_{ik} v_{jk} \cdots v_{mk} y_k \quad .$$

XIII. APPENDIX C

A. Existance of a Simple Polynomial of any Incomplete Function

Given any incomplete function of n two-valued variables defined at m points, a polynomial can be written that takes on the function value at all defined points and is of the form

$$P_s = \sum_{k=1}^{m} c_k \prod_{j=1}^{n} (z_j)^{z_{jk}} \tag{35}$$

where the function table is arrayed such that the first m points are the defined points.

<u>Proof</u>: The function can be written in the form of z and $\overline{z}$ as in Theorem 2 using the function value for the first m points and zero for the undefined points. The resultant will be

$$P_m = y_1 p_1(z,\overline{z}) + y_2 p_2(z,\overline{z}) + \ldots + y_m p_m(z,\overline{z}) \; .$$

Each term $p(z,\overline{z})$ contains each of $z_1, z_2, \ldots, z_n$ either as such or negated. Replacing $\overline{z}_j = 1 - z_j$ gives a polynomial in $z_1, z_2, \ldots, z_n$ that contains in general every term of the form

$$p_k = \prod_{j=1}^{n} (z_j)^{z_{jk}}$$

which is every possible combination of products of $\{z_j\}$ plus unity. Note that every coefficient of a term is formed by summing and differencing $y_1, y_2, \ldots, y_m$ for defined points.

Since the function is undefined for points $m+1, m+2, \ldots, 2^n$ the value at those points is of no interest. A combination of terms can now be made that does not influence the value of the polynomial at defined points. Assume there is an undefined point that would give a first order term from

$$p_k' = \prod_{j=1}^{n} (z_j)^{z_{jk}'} = z_k'$$

This term p_k' of the polynomial would be 1 when the z_k' is one alone or when any other term containing z_k' is one. Since there is no interest in the case when z_k' is one alone, the z_k' term of the polynomial can be eliminated by adding the coefficient c_k' to every other term containing z_k'. A similar argument can be advanced for all first order and higher terms corresponding to undefined points. Thus, all terms of P_m except those corresponding to the first m points are eliminated leaving only terms of the form of 35. Notice again that the final coefficients c_k are a combination of addition and subtraction of $y_0, y_1, \ldots, y_m$.

XIV. APPENDIX D

A. Uniqueness of a Simple Polynomial of any Incomplete Function

Given any incomplete function of n two-valued variables no more than one polynomial of the form

$$P_s = \sum_{k=1}^{m} c_k \prod_{j=1}^{n} (z_j)^{z_{jk}} \tag{36}$$

where the first m points of the function table are the defined points, can be found that takes on the function value at defined points.

Given a polynomial of the form 36 there must be a polynomial P_m in $(z,\bar{z})$ of the form of Theorem 2 passing through the same points as P_s.

$$P_m = y_1 p_1(z,\bar{z}) + y_2 p_2(z,\bar{z}) + \ldots + y_m p_m(z,\bar{z}) + u_{m\ 1} p_{m\ 1}(z,\bar{z})$$
$$+ \ldots + u_{2^n} p_{2^n}(z,\bar{z}) \tag{37}$$

Substituting $\bar{z}_j = 1-z_j$ into 37 gives

$$P_m = \sum_{k=1}^{2^n} c_k' \prod_{j=1}^{n} (z_j)^{z_{jk}} \tag{38}$$

Since 36 and 38 are identities their coefficients can be equated term by term. Note that each coefficient c_k' is a linear function of those of $y_1, y_2, \ldots, y_m, u_{m+1}, \ldots, u_{2^n}$ that are function values of points corresponding to products of the form

$$p_k' = \prod_{j=1}^{2^n} (z_j)^{z_{jk}'}$$

wholly included in

$$p_k = \prod_{j=1}^{2^n} (z_j)^{z_{jk}} \quad .$$

Thus, each first order term corresponding to an undefined point will be a linear function of at most a constant and the unknown value corresponding to that point. Thus, the unknown (u_k) value is uniquely determined by a defined value. Extending the argument to higher order terms corresponding to undefined points, the only unknown (u_k) value appearing in the coefficient will be the value whose point corresponds to the term; all other unknowns of lower order having already been written as a function of known values. Thus, the new unknown is a unique function of known values.

Thus a polynomial of the form 36 uniquely defines all possible function value points in terms of defined points. Thus no more than one polynomial of the form 36 takes on the function values at all defined points.