ADAPTIVE DIGITAL FUNCTION GENERATORS AND THEIR APPLICATION TO CONTROL SYSTEMS

A thesis submitted for the degree of
Doctor of Philosophy
of the University of London

by

Eric Laurence Sigurdson, BSc

Department of Electrical Engineering
Imperial College of Science and Technology
London S.W.7.

August 1966

ADAPTIVE DIGITAL FUNCTION GENERATORS AND THEIR APPLICATION TO CONTROL SYSTEMS

ABSTRACT

The need for adaptive digital function generators is illustrated. Continuous and discrete functional expansions are discussed and a new set of discrete polynomials used later in the thesis are developed along with their orthogonal properties, recurrence relations etc. Digital functions are defined leading into a general study of boolean functions. Networks for realizing boolean functions using nor gates with a 'fan-out' of one are developed as well as a method for minimizing these nets.

How continuity of the original function affects the quantization and nature of the digital function approximating it, is then explained, and ways of realizing these functions using boolean networks illustrated. The difficulty of adaptation is discussed in more detail and a specific control systems problem is solved by the application of an adaptive digital network.

The identification of on-line processes using continuous and discrete polynomial expansions of the input and output is discussed and a set of computer results, verifying the method for the discrete case included. How the various spectra may be up-dated continuously, along with the effect of noise on the identification efficiency is demonstrated. Finally, the identification method employing the discrete polynomials is extended to multi-dimensional systems.

ACKNOWLEDGEMENTS

The author would like to express his thanks to the Athlone Fellowship Committee and the National Research Council of Canada who, through their financial assistance, made this work possible. Thanks are due also to Professor J.H. Westcott and the remaining staff and students of the Control Systems Section at Imperial College for their help and criticism.

Finally, the author would mention his wife who has been a source of both inspiration and consolation throughout the preparation of this thesis.

To Jane

TABLE OF CONTENTS

CHAPTER I

Introduction

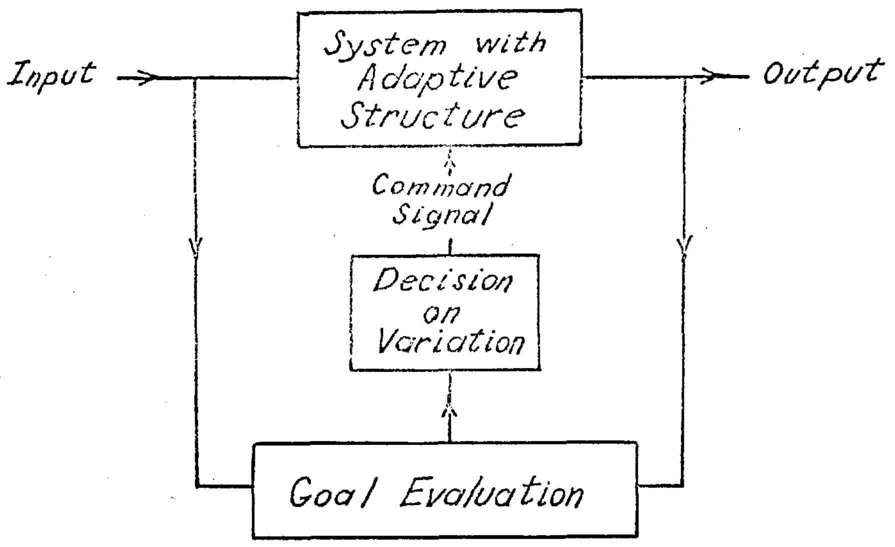
At first sight the title may appear to be rather imposing and, like all titles of its kind, rather misleading. It is the object of this Chapter to explain the title and to give a general outline of the thesis, along with a partial justification for exploring this particular aspect of control theory.

The thesis is divided into two main sections, roughly corresponding to 'Adaptive Digital Function Generators' and 'Their Application to Control Systems'. By far the greatest portion is concerned with the first part, the second being discussed here and in Chapter VI. There is good reason for this, as the control literature already abounds with reference to the second[12,37,49]. It is only necessary to read 'Digital Computers' for the first phrase and this is immediately evident.

In recent years the greatest effort has been applied in making control systems compatible with digital computers; that is, in trying to evolve mathematical models and expressions to represent the system, that can be easily handled by the computer. This has led to such sophisticated and powerful methods as dynamic programming[46] and the Pontryagin maximum principle[47] for solving control systems problems. Now, however, the methods have started to outpace even the largest and fasted machines on the market, due to the complexity of the mathematical analysis necessary. To circumvent this difficulty the trend has been to design learning systems[48], that have built into them variable structures and self analyzing equipment. Thus, instead of trying to do a complete mathematical analysis of the system under study, the necessary adaptive hardware is stored as an intrinsic part of the machine, along with information telling the system what it is trying to optimize, or, to use the learning terminology, setting a goal[50] for the system.

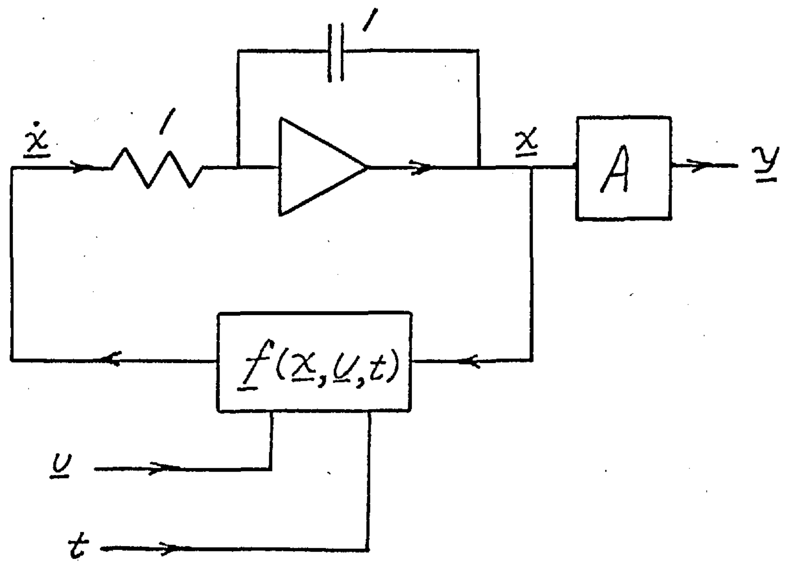
This leads to the block diagram shown in figure (1.1^a). The block labelled 'goal evaluation' could be redesignated as 'system performance evaluator', 'cost function calculator', or many other things. The decision block determines which way to change the adjustable parameters and may contain such things as gradient estimators, system identification procedures, model references etc. Its output is an order to change the variable configurations in the system to some other setting. In what follows it is assumed that, for the system under consideration, some method has already been evolved for designing these two blocks. Neither of these is a simple procedure and have absorbed most of the effort in the control field for the past few years. Our concern is only with the manner in which the system is changed having the two subsystems at our disposal.

The obvious variable structures to include in a system are variable gains or time constants[12], in order to compensate for changes in the system parameters. For these cases the advantages of going to digital circuitry is not immediately evident as there are severe limitations on the kind of adaptive operations that may be performed. The variable component for the systems below is assumed to be a zero-memory function. Here, if the function is obtained by a variable continuous network, the search problem becomes enormous due to the large number of possible



Generalized Learning System

(a)



Analogue Computer Realization of Dynamic Systems

Figure 1.1

functions available. Using digital systems it is an easy matter to limit the size of this class and hence to reduce the search to reasonable proportions. Why variable functions are chosen is illustrated below where the use of function generators in different configurations is discussed.

1.1 Use of Function Generators

The first application is in realizing compensating networks either in the forward, or the feedback path of a system. The behaviour of the compensating system may be defined by a set of equations[51] of the form

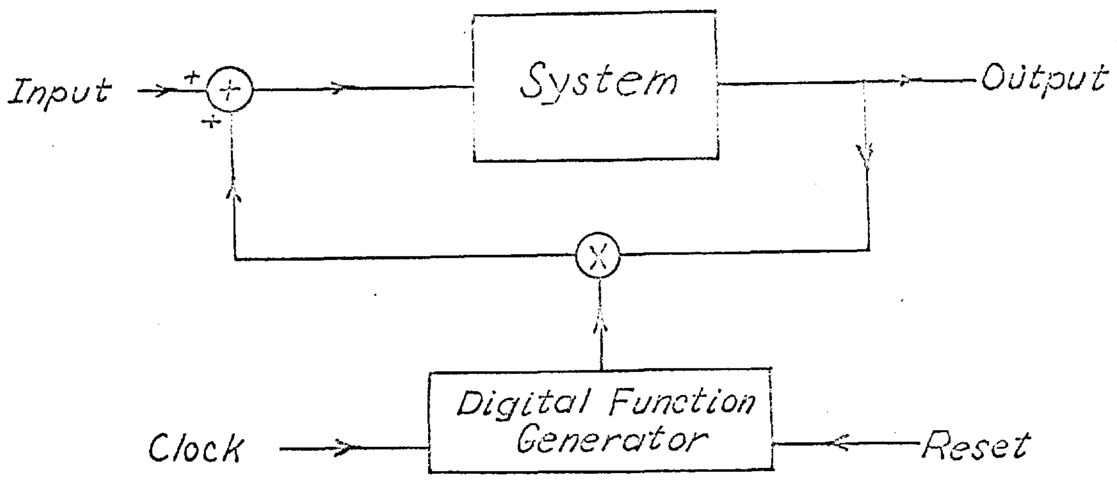
$$\dot{\underline{x}} = \underline{f}(\underline{x},\ \underline{u},\ t) \tag{1.1}$$

where $\underline{x}$ is an n vector, suitably chosen to represent the state of the system; $\underline{u}$, an r vector, is the input, and t is the time. The output $\underline{y}$ is some linear combination of the state variables x_i. This is equivalent to the analogue computer set up in figure (1.1^b) where the single integrator represents a set of n interconnected integrators, and the block A yields the required linear combination of the state. All compensating networks may be put in this form. If the required compensating system is time invariant then the variable t does not appear explicitely as shown in the diagram. The functions $\underline{f}$ are zero memory functions of n+r+1 variables. For adaptation purposes these could be realized by function generators giving a set of non-linear compensating elements, and allowing much more scope in design than the linear networks now in use.

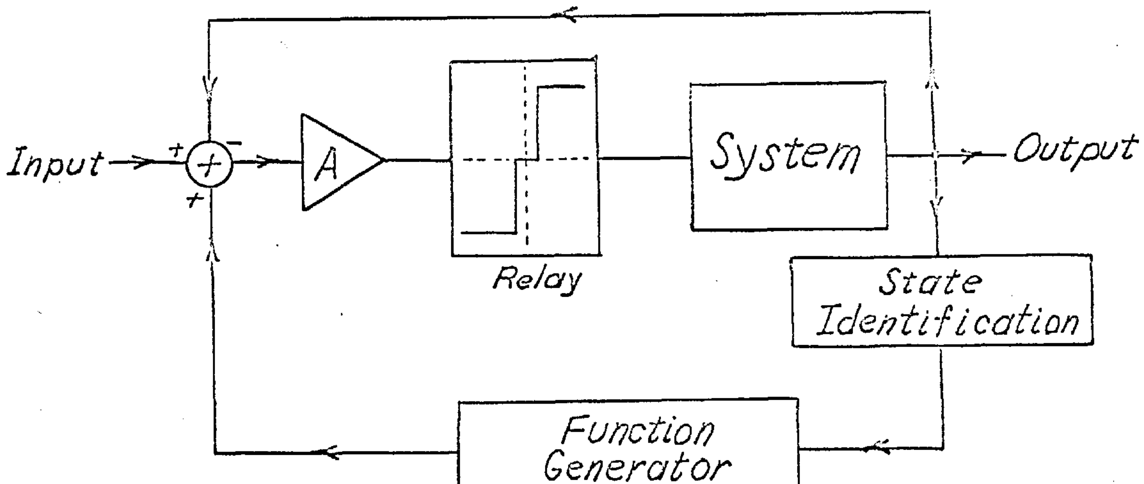
Merriam[47] has shown that for linear systems with quadratic cost functions, the optimal control scheme involves inserting a number of time varying gains in the forward and feedback paths.. The variable gains $k_i(t)$ can be realized by a function generator driven by a clock. An example of such a control system, with a variable gain in the feedback path, is shown in figure (1.2^a). This is ideally suited to the incremental function generators illustrated in Chapter V. The multiplier may either be an analogue multiplier, in which case a single D-A (digital to analogue) converter would be required on the output of the generator, or a digital multiplier, with the necessary A-D and D-A converters. A time varying gain using analogue components only is a difficult and complex device to design, whereas using digital circuitry, it becomes a straightforward procedure.

Perhaps the most obvious application for adaptive digital function generators is in obtaining switching boundaries for relay control systems. If the criterion function for the design of a system is to bring it to a specified final state in minimum time, and the allowed control effort is bounded, then a maximum effort or 'bang-bang' control system results. The control alternately moves from one bound to the other until the final state is reached when it drops to zero. The controlling element can be a relay, or set of relays, positioned as shown in figure (1.2^b). The relay (relays) position is uniquely defined by the input and the state of the system. The different possible positions of the relay correspond to different portions of the state space separated by a switching boundary.

This switching boundary can be represented by an equation of the form



System with Time Varying Gain
(a)



Relay Tracking System
(b)

Figure 1.2

$$g(\underline{x}, \underline{u}) = 0$$

defining an n-1 dimensional surface in the n space. If this function is realized by a generator, then as a particular trajectory passes through an $\underline{x}$, $\underline{u}$ state on the boundary, g must change sign. Thus this function could be used to drive the relay, provided the sign of g is chosen correctly on either side of the boundary.

For single input single output systems, where it is desired that the input follow the output, the system may be driven by the error signal plus some function of the remaining states. These remaining states, in effect, allow one to calculate the stored energy, and hence find how much sooner the relay must be switched to ensure that, when the system has zero error, it also has zero stored energy, implying that it must stay at rest. Figure (1.2^b) gives the block diagram of this system. The simplest and most natural state set to choose for most nth order systems, consists of the output and its first n-1 derivatives. An auxiliary device for estimating these derivatives is therefore necessary. A dead band relay is used to prevent the system from going into limit cycle operation for small errors. The amplifier A serves to give enough drive to the relay coil for small signal levels. This is discussed in greater detail following the general description of adaptive function generators.

1.2 Outline of Thesis

Having demonstrated the need for function generators it is not really necessary to argue for adaptation. The value of the epithet 'digital' is amply illustrated in the following pages, where it is shown that 'digital' does not necessarily imply the use of a full-fledged computer.

Chapter II contains a thorough summary of various functional representations along with a theorem, which, although it is probably already extant in some work, has not come to my knowledge before, at least not in this particular form. Both continuous and discrete functions are analyzed and a new set of discrete polynomials, developed with all their ramifications in Appendix I, is introduced. Following this, known methods of generating continuous functions, along with a new application of the Walsh functions, are discussed in order to demonstrate the need for introducing digital functions.

In analyzing digital functions it was found necessary to delve quite deeply into the boolean algebra and the various methods of representing boolean functions. Chapter IV develops the necessary tools and then goes on to discuss 'universal nets', and shows how these may be used to obtain arbitrary boolean functions. How the number of elements in these nets may be minimized, along with a number of examples, is also presented. In the next chapter the actual realization of digital functions using boolean functions is discussed in detail, and the concept of a 'continuous' digital function is mentioned. This last involved some quantization theory, and it was thought worthwhile to present a few results concerning the best methods of choosing quantization levels.

In Chapter VI the application of digital function generators to control systems is developed onward from the short introduction of the previous section. Here it is demonstrated how digital functions can be used to reduce the search problem for adaptive control systems, and can even result in some simplifications of the two auxiliary blocks of figure (1.1^a). A detailed example, using a relay controlled servo-motor, is performed to show exactly what difficulties are encountered and how these may be solved by using a digital generator.

Moving slightly away from the main flow of the thesis, Chapter VII shows how discrete functional expansions may be used to identify linear system parameters. It is demonstrated how a finite number of input and output spectral coefficients can determine the system parameters exactly. The determination procedure uses the discrete polynomials developed in the Appendix and an example, using a third order variable system, is worked out on the computer. How the overall system behaves in the presence of noise and with variations of the polynomial characteristics is illustrated. The method is then extended to multi-dimensional systems. Hopefully, the conclusions will speak for themselves.

CHAPTER II

Functional Representation

2.1 Definitions

Formally a function is defined as a mapping from a set of elements 'x' (the domain) to a set of elements 'y' (the range), the whole process to be viewed as a look-up table. When the domain is finite there is no difficulty; when the domain is infinite, to quote a well-known phrase, 'conceptually there is no difficulty', although the mind baulks at the thought of an infinite look-up table. Practically, one is forced to the notion of algorithm or rule to get from the input space to the output space. An algorithm amounts to a set of operations which, applied in a specified order to the domain element 'x', will yield the range element 'y'. This leads to the more familiar looking definition:

$$y = f(x)$$

where f is the algorithm in question.

Further restricting ourselves to considering the input space as the set of real n-tuples and the output space as the set of real numbers the notation becomes:

$$y = f(x_1, x_2 \; \ldots\ldots \; x_n)$$

where all the elements x_i and y are real numbers. Finally a dicotomy of the input elements into two classes, one called 'variables' and the other 'constants' or 'parameters' (variables with a difference) is performed, yielding:

$$y = f(a_1, a_2 \;.....\; a_r, \;\; x_1, x_2 \;.....\; x_s)$$

the a_1 ..... a_r are fixed numbers while the x_1 ..... x_s vary over the set of all real s-tuples. The domain has been reduced to the set of s-tuples and the function is contained in the operator f and the constants a_1 ..... a_r. The constants can be absorbed in the algorithm f to give

$$y = g(x_1, x_2 \;.....\; x_s).$$

The algorithm can take many different forms and the search for a 'simplest' form, due to its obvious application in computer programming, has received a great deal of attention in the literature lately[6,7]. The two most common forms are the polynomial and the differential equation form. It is hard to think of a representation that in the end does not reduce to one, or the other, or a combination of both of these.

As an example the function $y = Ae^{\alpha x}$ (variable x, parameters A and α) may be represented as

$$y = \left(\sum_{i=0}^{\infty} a_i x^i\right) \qquad \begin{array}{l} a_0 = A \\ a_i = \frac{\alpha}{i} a_{i-1} \end{array}$$

or equivalently as the solution to

$$\frac{dy}{dx} - \alpha y = 0 \qquad y(0) = A \;.$$

The polynomial representation is that used for all digital computation while the differential equation form is normally used for analogue computation. Note that in going from the first to the second equation one of the parameters (in this case A) is transferred from the equation to the initial condition. In general, if we have a function containing r parameters, it can be replaced by an r^{th} order differential equation independent of the parameters and a set of r initial conditions containing them.

For the above example, if the parameter α were to be eliminated also, the differential equation would become

$$\frac{d^2y}{dx^2} - \frac{1}{y}\left(\frac{dy}{dx}\right)^2 = 0 \qquad \begin{aligned} y(0) &= A \\ y'(0) &= A\alpha \end{aligned}$$

where $y'(x) = dy/dx$. We shall be concerning ourselves almost entirely with representations of the first type.

Following Sansone[5], a real 'Hilbert' space, defined over a measureable subset 'g' of the space of n-tuples, is the space of all functions f such that $\int_g f^2(\underline{x})\, \underline{dx}$ exists ($\underline{x}$ is an n-dimensional real vector and $d\underline{x}$ is the volume element in n-dimensional space).

Confining ourselves to a Hilbert Space is certainly no practical constriction and is necessary to ensure convergence for the approximating series which are discussed in the following.

2.2 Spectral Analysis

Unless otherwise indicated we will use the notation

$$f(\underline{x}) = f$$
$$\int_g f(\underline{x})\, d\underline{x} = \int f dx \; .$$

The next problem to consider is how 'best' to approximate a given function, f, of a Hilbert space by a finite set of known functions $h_0, h_1 \; \ldots\ldots \; h_n$ of the space. The form of the approximating function will be assumed as

$$h' = \sum_{i=0}^{n} a_i h_i \qquad (2.1)$$

where the a_i are constants to be determined.

As in most control situations the definition of 'best' has to be somewhat arbitrary and is taken to be that set of constants a_i that minimize the functional

$$E(a_0 \;..... \; a_n) \;=\; \int p(f-h')^2 \, dx \qquad (2.2)$$

where p is a non-negative weighting function $p(\underline{x})$ which enables us to weight errors in one portion of the space more heavily than in other portions. This error function yields the best 'approximation in the mean'[5].

For a minimum in E

$$\frac{\partial E}{\partial a_i} = 0 \qquad i = 0,1 \;..... \; n$$

Differentiating equation (2.2) gives

$$\frac{\partial E}{\partial a_i} = -\int p\left(f - \sum_{j=0}^{n} a_j h_j\right) h_i \, dx = 0 \qquad i = 0,1 \;..... \; n \quad (2.3)$$

implying that

$$\int p f h_i \, dx = \int \sum_{j=0}^{n} a_j h_j h_i \; p \, dx \qquad i = 0,1 \;..... \; n \;. \qquad (2.4)$$

Defining an (n+1) vector $\underline{c}$ such that

$$c_i = \int p f h_i \, dx \qquad (2.5)$$

and an (n+1) x (n+1) matrix H

$$h_{i,j} = \int h_j h_i \; p \, dx \qquad (2.6)$$

then the set of equations (2.4) for the coefficients in matrix form become

$$\underline{c} = H\,\underline{a}\,. \tag{2.7}$$

Provided the determinant of H ($|H|$) is not zero the solution is

$$\underline{a} = H^{-1}\underline{c} \tag{2.8}$$

$|H|$ is the Gram determinant for the set of functions $\sqrt{p}h_0$, $\sqrt{p}h_1$ ..... $\sqrt{p}h_n$ (see Sansone) and provided these functions are linearly independent then, by a well known theorem

$$|H| > 0$$

and H is positive definite implying that $\sum_{i=0}^{n}\sum_{j=0}^{n} d_i d_j h_{i,j}$ is greater than zero for all non-zero vectors $\underline{d}$.

Evidently the set $\{\sqrt{p}h_i\}$ are linearly independent if, and only if, $\{h_i\}$ are linearly independent. Therefore the condition that a unique solution for $\underline{a}$ exists is that the base functions h_i be linearly independent. Note that H is a function only of the base polynomials and p. c_i may be thought of as the projection of f on to h_i.

Deriving E a second time results in

$$\frac{\partial^2 E}{\partial a_j\,\partial a_i} = \int p h_i h_j\, dx = h_{i,j}.$$

Hence the positive definiteness of H ensures that we have a minimum at the solution point.

The original set of functions h_0 ..... h_n will be called an n-basis for the space, and will be represented by a single (n+1) dimensional vector $\underline{h}(\underline{x})$; therefore, in vector notation the approximating function h' is

$$h' = \sum_{i=0}^{n} a_i h_i = \underline{a}^t\underline{h}\,. \tag{2.9}$$

Substituting the solution for $\underline{a}$ found in equation (2.8) this can be expressed as

$$h' = (H^{-1}\underline{c})^t \ \underline{h}$$

$$= \underline{c}^t (H^{-1})^t \ \underline{h}$$

and using the fact that H is symmetric and that the transpose and inverse commute we arrive at

$$h' = \underline{c}^t H^{-1} \ \underline{h} \tag{2.10}$$

Equation (2.5) may be put in the form

$$\underline{c} = \int pf(\underline{h}) \ dx \tag{2.11}$$

and (2.6) in the matrix form

$$H = \int p(\underline{h} \ \underline{h}^t) \ dx \tag{2.12}$$

where in the first case the integration operator $\int pf \cdot dx$ operates on every element of the vector $\underline{h}$; in the second case the operator $\int p \cdot dx$ operates on every element of the matrix $\underline{h} \ \underline{h}^t$.

An interesting theorem, which seems to have escaped the literature, will now be proved. Although it is, perhaps, trivial when viewed from a certain angle, it does illustrate a significant invariant property of the approximating function h'.

Theorem 2.1 If $\underline{h}$ is an n-basis and $\underline{g}$ is another n-basis such that g is a non singular linear combination of $\underline{h}$ ($\underline{g} = A\underline{h}$, $|A| \neq 0$) then the two derived approximating functions g' and h' are identical.

Proof:- Let the projection of some arbitrary f on $\underline{h}$ be $\underline{c}$ and on $\underline{g}$ be $\underline{d}$, i.e.,

$$\underline{c} = \int pf(\underline{h}) \ dx$$

$$\underline{d} = \int pf(\underline{g}) \ dx \ .$$

Define matrices H and G as

$$H = \int p(\underline{h}\,\underline{h}^t)\,dx$$
$$G = \int p(\underline{g}\,\underline{g}^t)\,dx$$

the best approximating function for the two bases are from equation (2.10)

$$h' = c^t H^{-1} \underline{h} \qquad (2.13)$$

$$g' = d^t G^{-1} \underline{g} \qquad (2.14)$$

Now $\underline{g}\,\underline{g}^t = A\underline{h}\,(A\underline{h})^t$

and
$$\int \underline{g}\,\underline{g}^t p dx = \int A\underline{h}\,\underline{h}^t A^t p dx$$
$$= A(\int \underline{h}\,\underline{h}^t p dx)A^t$$

or
$$G = AHA^t. \qquad (2.15)$$

$$\underline{d} = \int pf\underline{g}dx = \int pfA\underline{h}dx$$
$$= A\int pf\underline{h}dx = A\underline{c}.$$

Substituting these in equation (2.14) yields

$$g' = (A\underline{c})^t\,(AHA^t)^{-1}A\underline{h}$$
$$= \underline{c}^t A^t (A^t)^{-1}\,H^{-1}A^{-1}A\underline{h} = \underline{c}^t H^{-1}\underline{h}$$

i.e., $h' = g'$ as required. Q.E.D.

The significance of this theorem is, that regardless of which particular linear combination of base functions is chosen at the beginning, the approximating function is the same. In the particular case of polynomials it makes no difference which polynomial set you start with, the kind of approximation you get being dependent only on the weighting function $p(x)$. All that need be considered is the ease with which the base functions can be generated.

Thus, for example, if we wanted a Tchebychev approximation of order n on the interval $(-1, 1)$, the weight function $1/\sqrt{1-x^2}$ would be used. The base functions could be chosen as $1, x \ldots\ldots x^n$ and the matrix

$$h_{i,j} = \int_{-1}^{1} \frac{x^i x^j}{\sqrt{1-x^2}} dx$$

calculated and inverted beforehand. Then using equations (2.5) and (2.8) the best approximation may be calculated easily, the most difficult computation arising through equation (2.5), although this will still be considerably easier than if the h_i had been the Tchebychev polynomials.

In fact the matrix H has the form

$$\left\|\begin{array}{cccccc} h_0 & h_1 & h_2 & \cdots\cdots & & h_n \\ h_1 & h_2 & & & h_n & h_{n+1} \\ h_2 & & & & & \\ \vdots & & & & & \\ h_n & h_{n+1} & & & h_{2n-1} & h_{2n} \end{array}\right\|$$

where $h_k = \int_{-1}^{1} \frac{x^k}{\sqrt{1-x^2}} dx$

which is a 'Hankel' matrix of order (n+1) (see Gantmacher 'Matrix Theory', Vol. 2, pp 207, and reference 9) and has a particularly easy inversion formula.

2.3 Orthonormal Functions

A set of functions $\{h_i\}$ is orthonormal with respect to the weight function p(x) if

$$\int_g p\, h_i h_j\, dx = \delta_{i,j}$$

where $\delta_{i,j}$ is the Kronecker delta. This, according to what has gone before, implies that H=I and equations (2.7) and (2.8) reduce to $\underline{c} = \underline{a}$, making the multiplication by H unnecessary. The great advantage of orthonormal functions, however, is to be found in equation (2.3), which reduces to

$$\frac{\partial E}{\partial a_i} = a_i - \int pfh_i\, dx = 0.$$

This implies that, if some method for evaluating E and an 'on-line' method of varying a_i were available, we could find the correct coefficient value by finding where $\frac{\partial E}{\partial a_i} = 0$ independently of all the other coefficients. This is a great advantage when the coefficients are to be found by some analogue method or are tracking a slowly varying function.

For an arbitrary n-base $\underline{h}$ we can find a corresponding orthonormal base from equation (2.15). Using the fact that H is a real, symmetric positive definite matrix, a matrix A can always be found such that

$$AHA^t = I$$

(see reference 10, pp 54-55). This implies that the n-basis

$$\underline{g} = A\underline{h}$$

must be an orthonormal basis.

It is interesting to note that the orthonormal basis is not necessarily unique for a given p(x) and range of integration. To do a simple example, consider the two independent polynomials x+1, x-1, and orthonormalize those over the range (-1, 1) with the weighting function of 1.

$$h_{1,1} = \int_{-1}^{1} (x+1)^2 \, dx = \frac{8}{3}$$

$$h_{1,2} = \int_{-1}^{1} (x+1)(x-1) \, dx = -\frac{4}{3} = h_{2,1}$$

$$h_{2,2} = \int_{-1}^{1} (x-1)^2 \, dx = \frac{8}{3} .$$

The matrix H will be

$$H = \frac{4}{3} \begin{Vmatrix} 2 & -1 \\ -1 & 2 \end{Vmatrix}$$

a matrix A which will satisfy

$$AHA^t = I$$

is

$$A = \frac{1}{2\sqrt{2}} \begin{Vmatrix} 0 & \sqrt{3} \\ 2 & 1 \end{Vmatrix}$$

as can be checked by substitution. Therefore an orthonormal basis found from $\underline{g} = A\underline{h}$ would be

$$g_1 = \frac{1}{2\sqrt{2}} \left(0\,(x+1) + \sqrt{3}\,(x-1)\right)$$

$$g_2 = \frac{1}{2\sqrt{2}} \left(2\,(x+1) + 1(x-1)\right)$$

$$g_1 = \frac{\sqrt{3}}{2\sqrt{2}} (x-1), \quad g_2 = \frac{3x+1}{2\sqrt{2}}.$$

That these are orthonormal is easily checked by doing the integrations.

Note that these two functions are very different from the first two normalized Legendre polynomials defined over the same range,

$$P_0 = \frac{1}{\sqrt{2}}$$

$$P_1 = \sqrt{\frac{3}{2}}\, x$$

obtained by starting with the polynomials 1,x and normalizing in the same way.

If the transformation matrix A had been chosen as

$$\frac{1}{2\sqrt{2}} \begin{Vmatrix} 1 & -1 \\ \sqrt{3} & \sqrt{3} \end{Vmatrix}$$

then the Legendre polynomials would have resulted.

As mentioned previously this method of orthogonalizing a set of functions is unsatisfactory due to the difficulty in finding the required transformation A. The normal method used is the classical 'Gram-Schmidt' orthogonalization technique. This generates the functions sequentially. Assume a set of orthogonal functions $\left\{\phi_0, \phi_1 \ldots\ldots \phi_k\right\}$ orthogonal with respect to the weight function p and a function g linearly independent of the set ϕ_i. Then the function defined by

$$\phi_{k+1} = g - \sum_{j=0}^{k} \left(\frac{\int p g \phi_j \, dx}{\int p \phi_j^2 \, dx} \right) \phi_j$$

is orthogonal to all members of the set ϕ_i. This is easily shown by forming

$$\int p \phi_{k+1} \phi_i \, dx = \int p g \phi_i \, dx - \sum_{j=0}^{k} \left\{ \left(\frac{\int p g \phi_j \, dx}{\int p \phi_j^2 \, dx} \right) \int p \phi_i \phi_j \, dx \right\}$$

$$= \int pg\phi_i \, dx - \left(\frac{\int pg\phi_i \, dx}{\int p\phi_i^2 \, dx} \right) \int p\phi_i^2 \, dx$$

$$= 0.$$

This completes the discussion on generalized functions. We will now proceed to briefly mention some known expansions along with some of their most important properties.

2.4 Polynomial Representations

The following polynomial sets may all be generated from their weighting function and the set of linearly independent functions $1, x \ldots\ldots x^n \ldots\ldots$ by the orthogonalization procedure above. Fortunately recurrence relationships for them all may be found. These, along with the first few members of the set, will constitute a complete definition of the polynomials for all orders. The normalizing factor ($\int p\phi_n^2 \, dx$) is also given.

2.4.1 Legendre Polynomials

Defined on the interval (-1, 1) with a weighting factor of 1. Therefore, they give the best mean square estimate of any arbitrary function.

$$P_0 = 1$$

$$P_1 = x$$

$$(n+1)\, P_{n+1} = (2n+1)\, x\, P_n - n\, P_{n-1}$$

The normalizing factor is $\sqrt{\frac{2n+1}{2}}$.

2.4.2 Tchebychef Polynomials

These are orthogonalized on the interval (-1, 1) with a weighting function of $1/\sqrt{1-x^2}$. Due to the form of the weighting function these polynomials yield an approximating function which is the polynomial whose maximum deviation from the actual function is smallest (see reference 11, page 58).

$$T_0 = 1$$

$$T_1 = x$$

$$T_n = 2\,x\,T_{n-1} - T_{n-2}\ .$$

Normalizing factor is $\sqrt{\frac{2}{\pi}}$.

2.4.3 Laguerre Polynomials

These are defined on the range $(0, \infty)$ with a weighting function of e^{-x}. They give a good approximation on the beginning of the range, a great many terms being necessary if a good approximation far from the origin is desired.

$$L_0 = 1$$

$$L_1 = -x+1$$

$$nL_n = (2n-1-x)L_{n-1} - (n-1)L_{n-2}\ .$$

The normalizing factor is 1.

2.4.4 Associated Laguerre Polynomials

These are defined over the same interval as above only with the weighting function $e^{-x}\,x^{\alpha}$ where $\alpha > -1$.

$$L_0^\alpha = 1$$

$$L_1^\alpha = -x + (\alpha+1)$$

$$nL_n^\alpha = (2n-1+\alpha-x)\, L_{n-1}^\alpha - (n-1+\alpha)\, L_{n-2}^\alpha$$

The normalizing factor is $\sqrt{n!\,/\,\Gamma(n-\alpha+r)}$ where $\Gamma(a) = \int_0^\infty e^{-x}x^{a-1}\,dx$. These reduce to the ordinary Laguerre functions for $\alpha = 0$.

2.4.5 Hermite Polynomials

Defined over the range $(-\infty, \infty)$ with a weighting function of e^{-x^2}.

$$H_0 = 1$$

$$H_1 = -2x$$

$$H_n = -2xH_{n-1} - 2(n-1)\,H_{n-2}$$

The normalizing factor is $1/\sqrt{2^n\, n!\,\sqrt{\pi}}$.

2.5 Non-Polynomial Bases

The most common of these is the fourier series using the functions sin iθ and cos iθ defined over the interval $(-\pi, \pi)$. The normalizing factor is $\frac{1}{\sqrt{\pi}}$ except for the first one, i.e., the constant, where it will be $\frac{1}{\sqrt{2\pi}}$. These have been adequately covered in the literature and require no further elaboration here.

The only other well-known non-polynomial basis is the so-called 'orthogonalized exponentials' (see 12, pp 311-315). These are generated from the set $e^{-\alpha_i x}$ where the α_i are distinct real or complex numbers. These find specific applications in system identification due to the ease with which they may be realized in the frequency domain.

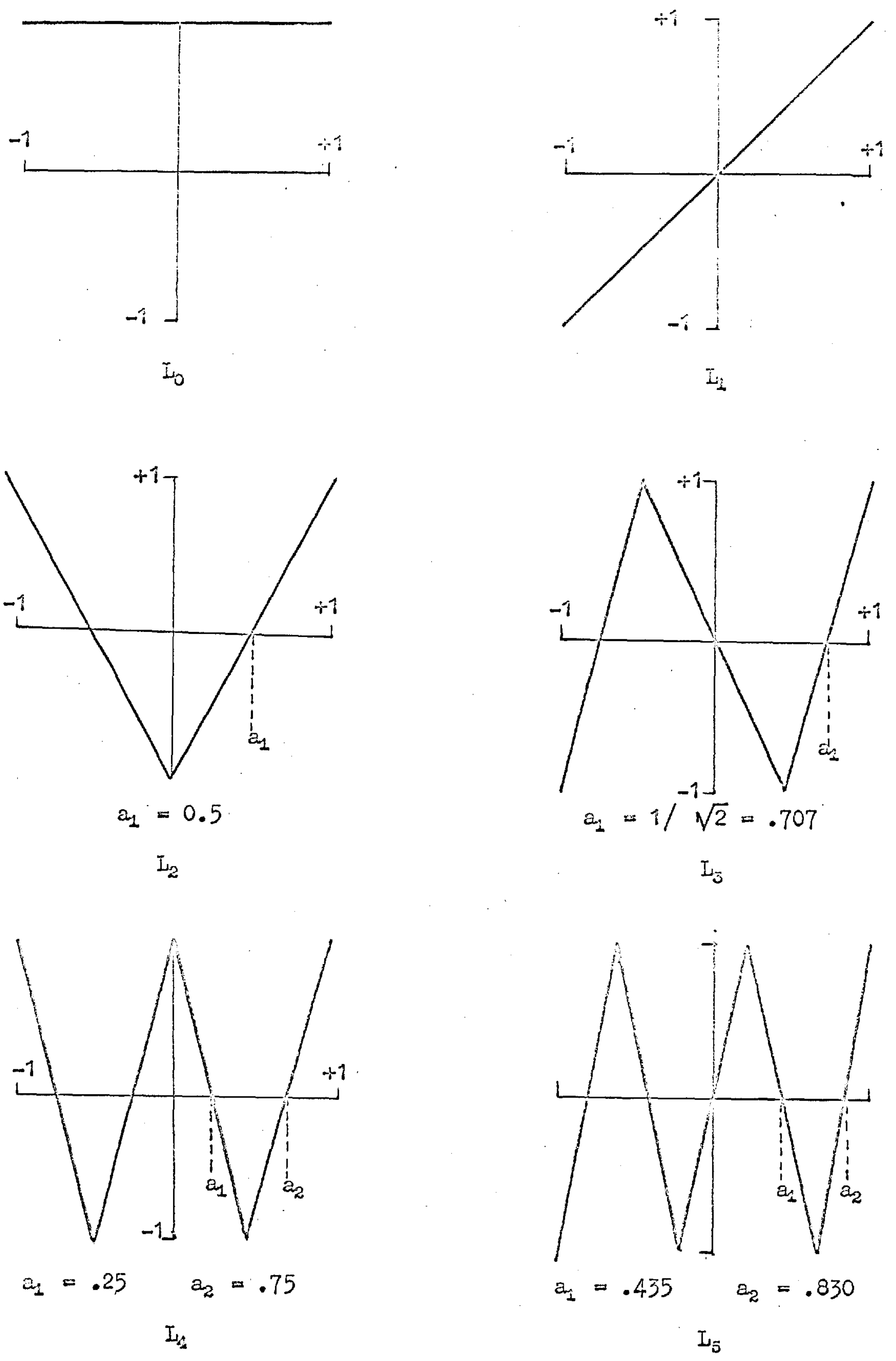
A last and rather interesting example of a non-polynomial basis may be found in the set of piecewise linear functions shown in figure 2.1. These functions are defined on the interval (-1, +1) by straight line segments between the upper and lower bounds of +1 and -1. In going from order k to order k+1 one more line segment (or equivalently one more zero) is introduced. Such a function of order k is specified entirely by giving its k zeros along with the restriction that it must pass through the point (+1, +1). The zeros may be chosen to ensure orthogonality; for suppose an orthogonal set up to order k has been generated and a function of order k+1 is to be found. Without loss of generality k+1 can be assumed to be odd, and the function L_{k+1} will be an odd function. It is already orthogonal to all the even ordered functions. There will be $\frac{k}{2}$ odd functions $L_1, L_3 \ldots\ldots L_{k-1}$, of order less than k, to which the function in question must be made orthogonal by choice of the zeros. For both even and odd functions the zeros are symmetrical with respect to the origin, the odd functions having a zero at the origin. L_{k+1} will have $\frac{k}{2}$ zeros, a_i, to the left of the origin. These may be ordered

$$0 < a_1 < a_2 \ldots\ldots < a_{\frac{k}{2}-1} < a_{\frac{k}{2}}$$

L_{k+1} may be expressed as a function of these $\frac{k}{2}$ parameters. The integrations

$$\int_{-1}^{1} L_{k+1} \left(a_1, a_2 \ldots\ldots a_{\frac{k}{2}}\right) L_i(x)\, dx = 0 \quad i = 1, 3 \ldots\ldots k-1$$

may be performed and set equal to zero, giving us $\frac{k}{2}$ equations in the $\frac{k}{2}$ unknowns a_i. These can be solved giving us the roots of L_{k+1} as



First Six Orthogonal
Linear Segment Functions

FIGURE 2.1

required. The first six of these functions are given in figure 2.1. The normalizing constant

$$\mu_i = \int_{-1}^{1} L_i^2 \, dx$$

is 2 for i=0 and $\frac{2}{3}$ for all other i.

Assuming an expansion formula of the form

$$f(x) = \sum_{i=0}^{\infty} a_i L_i (x)$$

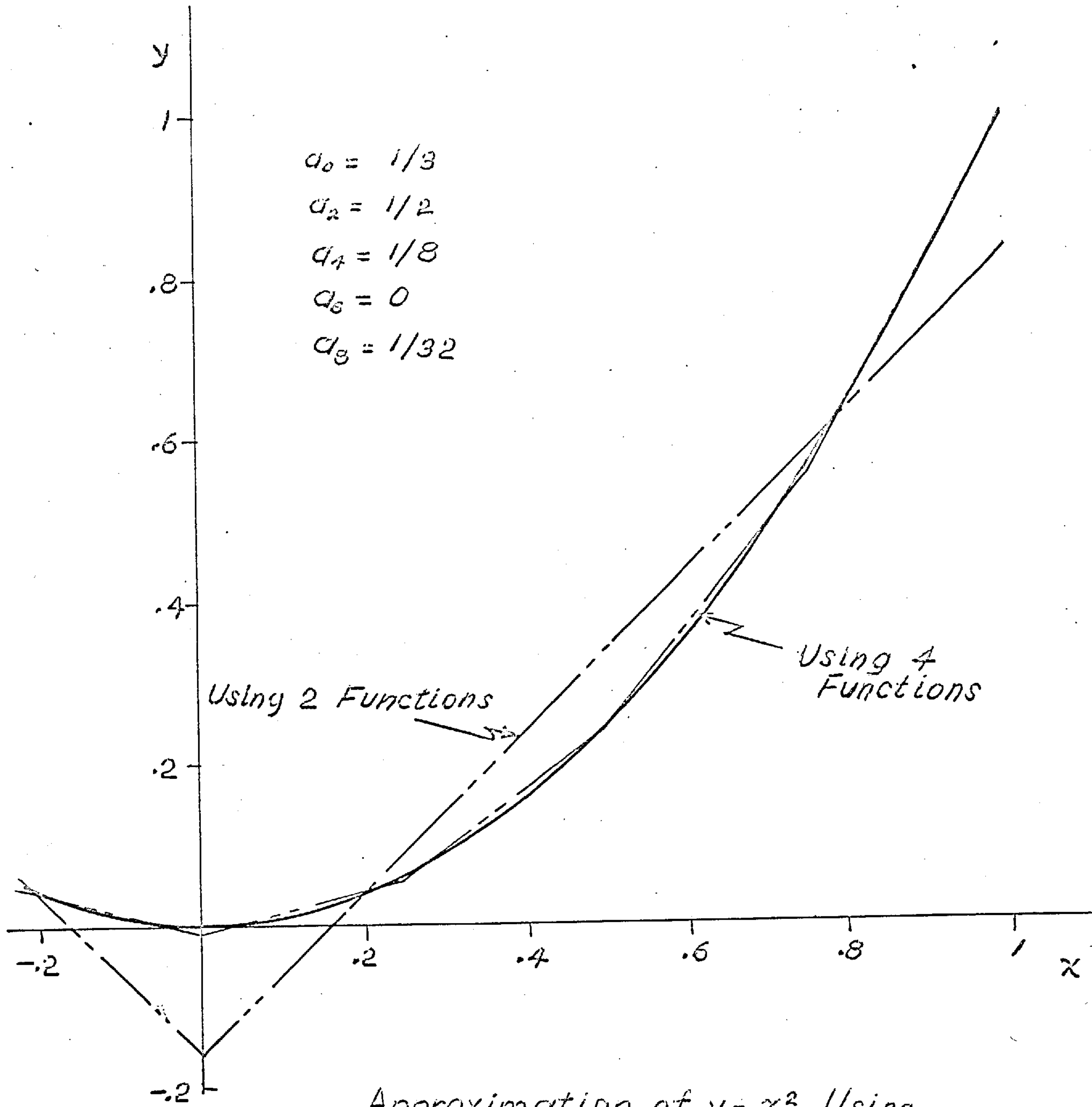
the coefficients a_i become

$$a_o = \frac{1}{2} \int_{-1}^{1} f(x) \, dx$$

$$a_i = \frac{3}{2} \int_{-1}^{1} f(x) \, L_i(x) \, dx \qquad i=1,2 \; .....$$

Using these formulae the function $f(x) = x^2$ is approximated in figure 2.2. The approximation using three terms and eight terms is given. Only four of the coefficients are non-zero in the final expansion as all the odd coefficients are zero and a_6 is also zero.

The study of these functions was carried no further as they are not in the main-line of the thesis. Questions of convergence and completeness have been left as well as obtaining a general expression for the roots of the functions. They have been mentioned only in passing and as a guide for possible future work. The application of such functions in the construction of adaptive analogue function generators is obvious.



Approximation of $y = x^2$ Using Orthogonal Linear Segments

Figure 2.2

2.6 Multi-dimensional Expansions

For examples of functions orthogonal over two or more dimensions, one need merely go to the theory of partial differential equations where nearly all the solutions to the equations are expansions in these functions. There they are called 'eigenfunctions'. The most commonly used of these is the multiple fourier series consisting of the set of functions

$\cos \mu x \cos \gamma y$	$\mu = 0,1 \ldots\ldots$	$\gamma = 0,1 \ldots\ldots$
$\sin \mu x \cos \gamma y$	$\mu = 1,2 \ldots\ldots$	$\gamma = 0,1 \ldots\ldots$
$\cos \mu x \sin \gamma y$	$\mu = 0,1 \ldots\ldots$	$\gamma = 1,2 \ldots\ldots$
$\sin \mu x \sin \gamma y$	$\mu = 1,2 \ldots\ldots$	$\gamma = 1,2 \ldots\ldots$

These form an orthogonal set over the rectangle

$$-\pi < x < \pi, \quad -\pi < y < \pi .$$

Any function of two variables defined over this region can be expanded in terms of them. The extension to n-dimensions is obvious. Expansion series of the form $\sum_{j=0}^{\infty} \sum_{i=0}^{\infty} \alpha_{ij} x^i y^i$ are also well-known as well as the corresponding orthogonalized functions (see Courant and Hilbert on Weierstrass's approximation theorem). Further examples abound (spherical harmonics, Sturm-Louville eigenfunctions etc.) in practically any physics book.

2.7 Discrete Functions

A discrete function is defined as a mapping from a finite, or denumerably infinite set of points X, into the set of real numbers. A simple example would be the function $f(x) = x^2 \quad x = 0,1,2 \ldots\ldots$ The qualifier 'discrete' is essentially a restriction of the domain of the function. Without loss of generality we can put the domain of the function in one-to-one correspondence with the integers allowing us to use the notation f(n) to denote a general discrete function.

All the results discussed for continuous functions are immediately applicable to the discrete case merely by substituting $\sum_{i=a}^{b} \cdot$ everywhere a form $\int_g \cdot \, d\underline{x}$ appears. The integers a and b define the range of definition of the function. In the following, for convenience of notation, these limits and the summation variable i will be left out, all sums assumed to be taken over the total range of definition. An expression like $\sum_{i=a}^{b} p(i)(f(i) - \sum_{j=1}^{n} a_j f_j(i))$ becomes $\sum p(f- \sum_{j=1}^{n} a_j f_j)$. The only restriction on all functions is that $\sum f^2(i)$ exist and be finite.

Analogously, given a discrete function f, a non-negative weight function p and a linearly independent set of discrete functions $h_1 \ldots\ldots h_n$, the best approximation in the mean may be found from minimizing the sum

$$E = \sum p(f- \sum_{j=1}^{n} a_j h_j)^2 .$$

The coefficients, determined by differentiation yield

$$\frac{\partial E}{\partial a_i} = - \sum p(f- \sum_{j=1}^{n} a_j h_j)h_i = 0 \quad i=1,2\ldots\ldots n$$

or defining the vector and matrix

$$c_i = \sum pfh_i$$

$$h_{i,j} = \sum ph_i h_j$$

$$\underline{c} = H\underline{a} .$$

The independence of the h_i ensure that $|H| \neq 0$, giving the solution for the coefficients as

$$\underline{a} = H^{-1}\underline{c}$$

the ultimate approximation to f being

$$h' = \underline{a}^t \underline{h} = \underline{c}^t H^{-1} \underline{h} .$$

Theorem 2.1 carries over into the discrete case also. However, in this context it has two rather interesting corollaries.

<u>Corollary 2.1.1</u> If a function f is defined over k distinct points and k linearly independent functions h_1 ..... h_k defined over the same points are given, then f may be realized exactly as a linear combination of h_1 ..... h_k.

<u>Proof</u>:- Evidently f may be realized exactly as a linear combination of the set

$$\begin{aligned} g_1 &= (1,0\ \ldots\ldots\ 0) \\ g_2 &= (0,1,0\ \ldots\ldots\ 0) \\ &\vdots \\ g_k &= (0,0\ \ldots\ldots\ 0,1) \end{aligned}$$

In fact $f(i) = \sum_{j=1}^{n} f(j)g_j(i) = g'$.

As the set $\underline{h}$ may also be realized by linear combinations of this set we have the existence of a non-singular A such that

$$\underline{h} = A\underline{g}$$

therefore by the theorem, the approximating function $g' = h'$ but $g' = f$ and the corollary results.

Corollary 2.1.2 There are at most k linearly independent functions defined over k points.

This is obvious from the previous result.

These corollaries are in a way analogous to the theorem on completeness for continuous functional approximations.

The Schmidt orthogonalization procedure carries over in an identical fashion to the continuous case and will not be repeated here. The remainder of the chapter will be devoted to briefly discussing a few known discrete expansion, followed by a fairly extensive study of a new set of discrete polynomials, which will be used again in the later chapters.

2.8 Examples of Discrete Expansions

2.8.1 Sampled Fourier Series

As might be expected, the most well-known discrete expansion is the discrete fourier series. The set of 2N functions

$$1,\ \cos x,\ \cos 2x,\ \ldots\ldots \cos (N-1)x,\ \cos Nx$$
$$\sin x,\ \sin 2x,\ \ldots\ldots \sin (N-1)x$$

are orthogonal over the discrete points $0, \frac{\pi}{N}, \frac{2\pi}{N} \ldots\ldots \frac{2N-1}{N}\pi$. The simple change of variables $y = \frac{N}{\pi}x$ gives us a set of functions orthogonal over the first 2N integers $0,1 \ldots\ldots 2N-1$. The orthogonality relations are

$$\sum_{y=0}^{2N-1} \sin\left(\frac{\pi}{N}ky\right)\sin\left(\frac{\pi}{N}my\right) \begin{array}{ll} = 0 & \quad m \neq k \\ = N & \quad m = k \end{array}$$

$$\sum_{y=0}^{2N-1} \sin\left(\frac{\pi}{N} ky\right) \cos\left(\frac{\pi}{N} my\right) = 0$$

$$\sum_{y=0}^{2N-1} \cos\left(\frac{\pi}{N} ky\right) \cos\left(\frac{\pi}{N} my\right) = \begin{cases} 0 & m \neq k \\ N & k = m \neq 0, N \\ 2N & k = m = 0 \text{ or } N \end{cases}$$

Any function defined over the first 2N integers can be expressed as a sum of the form

$$f(n) = a_0 + \sum_{i=1}^{N-1} a_i \cos\left(\frac{\pi}{N} in\right) + a_N + \sum_{i=1}^{N-1} b_i \sin\left(\frac{\pi}{N} in\right)$$

where

$$a_0 = \frac{1}{2N} \sum_{j=0}^{2N-1} f(j)$$

$$a_i = \frac{1}{N} \sum_{j=0}^{2N-1} f(j) \cos\left(\frac{\pi}{N} ij\right)$$

$$a_N = \frac{1}{2N} \sum_{j=0}^{2N-1} f(j) \cos(\pi j)$$

$$b_i = \frac{1}{N} \sum_{j=0}^{2N-1} f(j) \sin\left(\frac{\pi}{N} ij\right)$$

Evidently if all coefficients are calculated and used in the expansion then by theorem 2.1, the function f(n) will be described exactly. This formula has the added advantage of being defined for n not equal an integer and, therefore, may be used as an interpolating formula. For a more detailed discussion of these approximating functions see reference 13.

2.8.2 Rademacher - Walsh Functions

These functions were first discovered by Walsh in the twenties and developed further by Golomb[14] in the late fifties. They are defined over the first 2^n integers $0,1 \ldots\ldots 2^n-1$, and map into a range consisting only of the two numbers -1 and +1. Any integer on the above domain of definition may be represented as a binary number of the form $x_1, x_2 \ldots\ldots x_n$ where $x_i = 0$ or 1. For example the integer $6 = 1.2^2 + 1.2^1 + 0.2^0$ would be 110 or $x_1 = 1$, $x_2 = 1$, $x_3 = 0$.

A 'linear function' over the binary variables $x_1, x_2 \ldots\ldots x_n$ is defined as one of the form

$$f(x_1, x_2 \ldots\ldots x_n) = \text{mod } 2 \left(\sum_{i=1}^{n} a_i x_i \right) \qquad (2.16)$$

where all the a_i are 1 or 0, and the operator 'mod 2 gives f=1 if the sum is even, f=0 if the sum is odd. The 'ring sum' notation will be used in the following, implying that

$$\text{mod } 2 \left(\sum_{i=1}^{n} a_i x_i \right) = \sum_{i=1}^{n}{}^{\oplus} a_i x_i = a_1 x_1 \oplus a_2 x_2 \oplus \ldots\ldots \oplus a_n x_n .$$

As a simple example consider the function of two variables

$$f(x_1, x_2) = x_1 \oplus x_2$$

where $a_1 = 1$, $a_2 = 1$. It has a mapping given by the table below

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

If one more transformation is performed on the output of f (i.e., g(f)) such that g(1)=1, g(0)= -1 and, the variables $x_1 \ldots\ldots x_n$ are identified with those found in the binary number expansion of the domain, then the definition of the Walsh functions becomes

$$W(k) = g(f(x_1, x_2 \ldots\ldots x_n)) \qquad k = 0,1 \ldots\ldots 2^n-1$$

where f is in the form of equation (2.16). There are 2^n such functions corresponding to the 2^n choices of coefficients a_i. Using the binary number equivalent of the vector $a_1, a_2 \ldots\ldots a_n$ to index the particular Walsh function under consideration the set of 2^n functions may be denoted by

$$W_r(k) \qquad k = 0,1 \ldots\ldots 2^n-1.$$

These functions satisfy the orthogonality relation

$$\sum_{k=0}^{2^n-1} W_r(k)\, W_s(k) = \begin{cases} 0 & r \neq s \\ 2^n & r = s \end{cases}$$

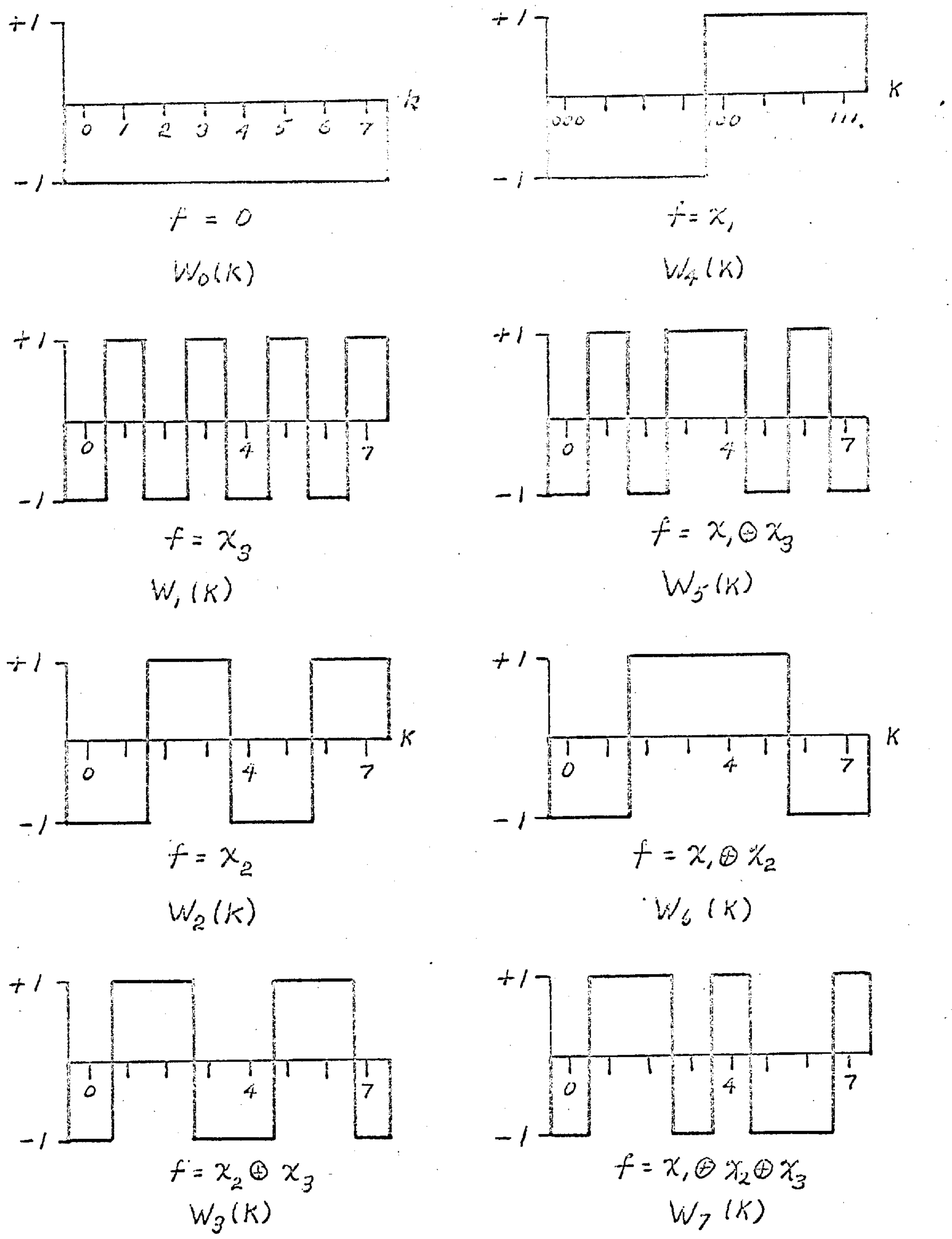
An arbitrary function h(k) defined over the same range may be expanded in terms of these functions

$$h(k) = \sum_{i=0}^{2^n-1} b_i W_i(k)$$

where

$$b_i = \frac{1}{2^n} \sum_{j=0}^{2^n-1} h(j)\, W_i(j).$$

The eight Walsh functions for n=3 are given in figure 2.3, along with their corresponding linear functions. Evidently these functions are linearly independent and thus will exactly realize a function over $(0,1 \ldots\ldots 2^n-1)$ if the entire expansion is used.



The Eight Walsh Functions
of Order 3

Figure 2.3

2.8.3 M-Sequences

The m-sequences or binary chain codes are generated by means of a feedback shift register with some linear function of the contents of the register in the feedback path. The maximal length codes of an n-stage shift register are 2^n-1 bits long and have the interesting property that any sequence is almost orthogonal to any delayed sequence, or equivalently the auto-correlation function of the code is almost a unit impulse. If, picking some arbitrary starting point, the string is designated by $f(n)$ then the orthogonality relationship looks like

$$\sum_{n=0}^{2^n-2} f(n)\, f(n+k) \begin{array}{ll} = -1 & \text{for } k \neq 0 \\ = 2^n-1 & \text{for } k = 0. \end{array}$$

For large n the codes may effectively be considered as orthogonal. Because of the form of the autocorrelation function, these codes have found considerable application in on-line identification of system parameters. For a thorough discussion of their use in realizing arbitrary functions, see reference 15. For a complete analysis on the generation of the codes refer to the classic paper of Elspas[16].

2.8.4 Discrete Laguerre Polynomials

These polynomials were developed to tackle a specific problem that occurs later in the thesis. As nothing like them seems to occur in the literature they were deemed worthy of the detailed analysis given in appendix I. The epithet 'Laguerre' is used, as an exponential type of weighting function was employed, and the final polynomials look suspiciously like the Laguerre polynomials. However, there does not seem to be any simple analytical relationship between the two.

The disꞌcrete laguerre polynomials are defined by the equation

$$L_i(n) = \sum_{k=0}^{i} (-1)^k \binom{i}{k} \binom{n+k}{i} \alpha^k$$

where α is a real number in the range $-1 < \alpha < 1$ and $\binom{a}{b}$ is the usual notation for the binomial coefficient. That these are i^{th} order polynomials in n arises from the fact that $\binom{n+k}{i}$ is an i^{th} order polynomial for all k. The first three are given below.

$$L_0(n) = 1$$

$$L_1(n) = n(1-\alpha)-\alpha$$

$$L_2(n) = \frac{n^2(1-2\alpha+\alpha^2)}{2} - \frac{n(1+3\alpha-2\alpha^2)}{2} + \alpha^2 .$$

In the appendix recurrence relationships and the difference equation these polynomials satisfy are found; also the fact that they form an orthogonal set over the integers (0,1 ..... n .....) is proved. The normalizing constant is

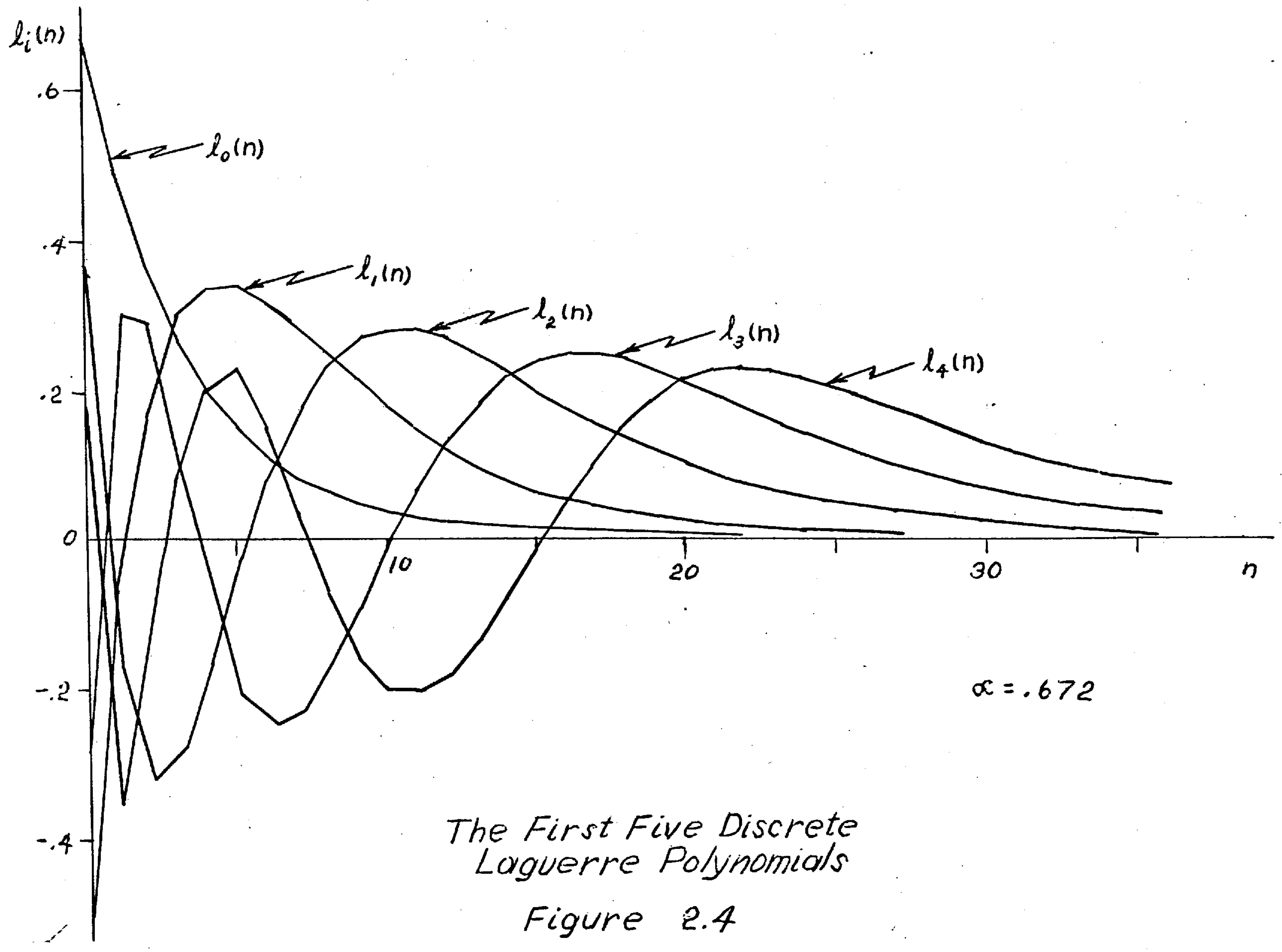
$$\mu_i^2 = \sum_{n=0}^{\infty} \alpha^n L_i^2(n) = \frac{\alpha^i}{1-\alpha}$$

Hence the functions

$$l_i(n) = \sqrt{1-\alpha} \;\; \alpha^{(n-i)/2} L_i(n)$$

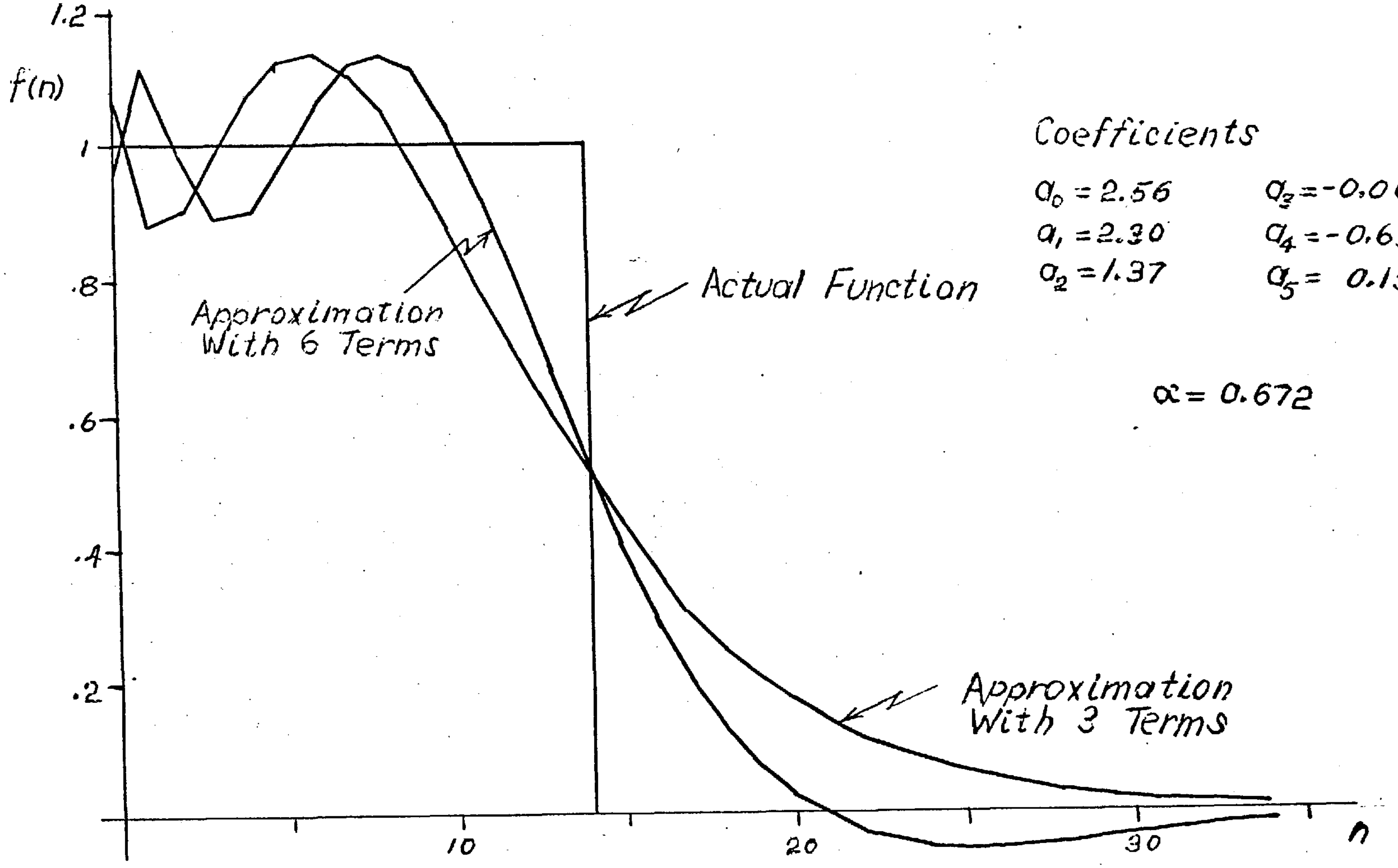
form an orthonormal set.

The first five of these polynomials for $\alpha = .672$ are drawn in figure 2.4. Their similarity to the Laguerre polynomials is immediately evident. Figure 2.5 shows their approximating capabilities when the square pulse shown is used as the sample function. The coefficients were calculated using the relationships (A.1.19) and (A.1.20)



The First Five Discrete Laguerre Polynomials

Figure 2.4



Approximation of Square Pulse on 0-14
Using Discrete Laguerre Polynomials

Figure 2.5

developed in appendix one. The remaining results in the appendix will be used in Chapter VII where identification of discrete processes is discussed.

This concludes the discussion on how functions may be represented in the continuous and discrete domain. The following few chapters will give some methods of practical realization of functions and mention some of the difficulties encountered.

CHAPTER III

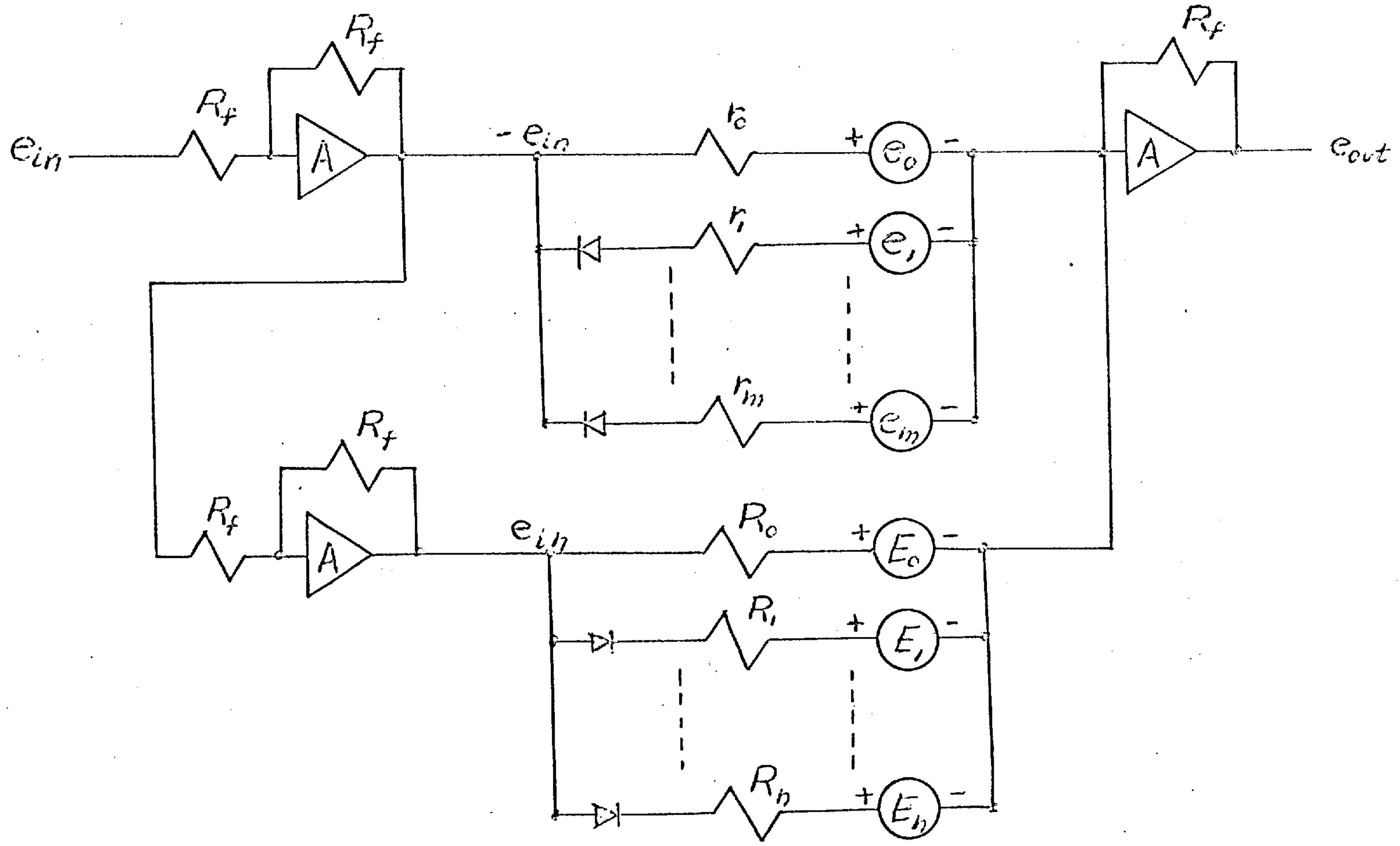
Deterministic Function Generation

This chapter will be divided into two main sections; the first discusses methods already known for realizing piecewise continuous functions, and the second gives a short preamble on the problem of generating discrete functions, the remainder of the topic being covered in Chapter IV. The restriction to piecewise continuous functions is no practical limitation, as the use of a function not falling into this class would be very specialized indeed. Just to dream up such an entity is a difficult task although one of my colleagues claims to have encountered a function discontinuous almost everywhere. At any rate, such functions are excluded from the following.

The most well-known generator, and the one most commonly used in the design of control systems, is the diode generator. Its main advantage is that it is cheap and reliable, although it is of little use when the problem of adaptation is encountered.

3.1 Diode Function Generator

Any continuous function may be approximated arbitrarily closely by a piecewise linear function. These functions may be realized by interconnections of diode and resistors as shown in figure 3.1. The first two analogue amplifiers serve to isolate the input from the diode net and give a low impedence source; the last acts as an adder. The constant voltage sources e_i and E_i are ordered so that



A Typical Diode Function Generator
Figure 3.1

$$e_m < e_{m-1} < \ldots\ldots < e_1$$

and $$E_1 < E_2 < \ldots\ldots < E_{n-1} < E_n \, .$$

If the input voltage is such that

$$e_{i+1} < -e_{in} < e_i$$

and $$E_j < e_{in} < E_{j+1}$$

then only the resistance paths r_0 to r_i and those from E_0 to E_j are conducting. Thus the input output relation, found by summing currents at the junction of the output amplifier, at this operating point is

$$-\frac{e_{out}}{R_f} = -\sum_{k=0}^{i} \frac{(e_{in}+e_k)}{r_k} + \sum_{k=0}^{j} \frac{(e_{in}-E_k)}{R_k} \tag{3.1}$$

which may be put in the form

$$\frac{e_{out}}{R_f} = \left[\sum_{k=0}^{i} \frac{1}{r_k} - \sum_{k=0}^{j} \frac{1}{R_k} \right] e_{in} + \sum_{k=0}^{i} \frac{e_k}{r_k} + \sum_{k=0}^{j} \frac{E_k}{R_k} \tag{3.2}$$

This is equivalent to the linear relation $\frac{e_{out}}{R_f} = a\, e_{in} + b$ where 'a' is determined only by the resistances of the net and 'b' by both the bias voltages and bias resistors. Evidently as e_{in} or $-e_{in}$ passes through one of the bias voltages e_i or E_j there is a jump change in slope, the change being positive if the bias in question was e_i, negative if the bias was E_j. The curve is continuous as the change in ordinate introduced by the first term is compensated for by a new constant in the second term. Thus the input output relation for this device will be a piecewise linear, continuous function.

Given a function of the sort described above, it is an easy matter to determine the values of resistors and bias supplies needed. Firstly the break points on the function determine the values of the constant voltage generators. If the slope increases beyond the break point the branch appears among the inputs to the upper amplifier, if it decreases, among those to the lower amplifier. Therefore, m, n and all the e_i, E_j other than e_0 and E_0, that appear in (3.1) are determined. Defining the conductances $G_k = 1/R_k$, $g_k = 1/r_k$ gives us these m+n+2 remaining unknowns to solve for. The two paths containing r_0 and R_0 determine the initial slope of the function. One of them will always represent an open circuit depending on whether the initial slope is positive or negative.

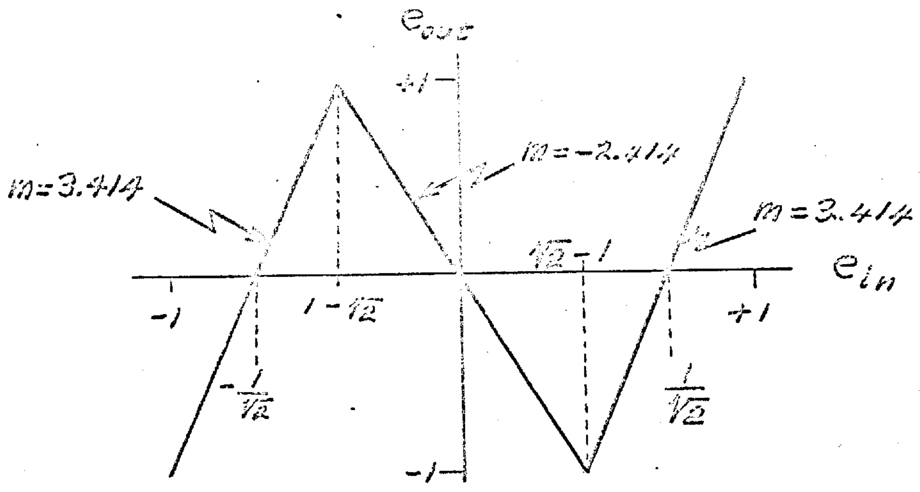
Starting with $e_{in} = -\infty$ and increasing, determine whether the initial slope, m, is positive or negative. If it is positive then $G_0=0$ and $g_0=m$, if negative $g_0=0$, $G_0=m$. The bias e_0 or E_0 is determined by the y intercept of this first segment. As e_{in} is increased a g_i or G_i is introduced depending on the change in slope. In this fashion the entire curve can be generated, and at the same time the values of the resistances determined directly.

As an example consider the third linear segment function mentioned in section 2.5, reproduced in figure (3.2^a) for convenience. The initial slope is $2+\sqrt{2}$ and the y intercept is $1+\sqrt{2}$. This gives the equation

$$e_{out} = R_f(g_0\ e_{in} + e_0 g_0) = 3.41\ e_{in} + 2.41$$

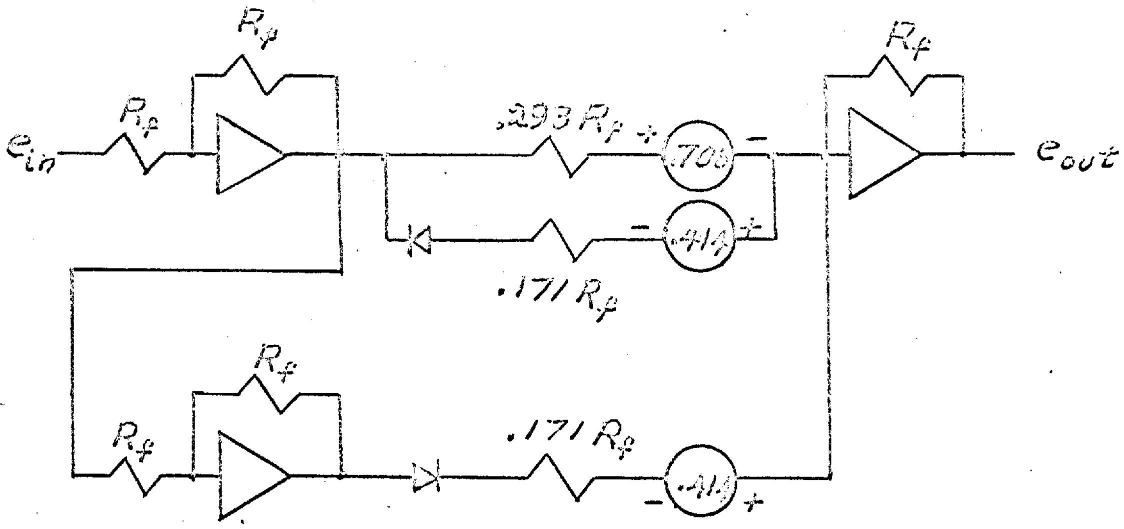
implying that $g_0 = \frac{3.41}{R_f}$, $e_0 = +0.706$. At $e_{in} = -.414$ there is a negative change in slope implying a circuit in the lower branch with $E_1 = -.414$. In the range $-.414 < e_{in} < .414$ the equation is $e_{out} = -e_{in}/.414$.

Here $$e_{out} = R_f\left[(g_0 - G_1)e_{in} + g_0 e_0 + G_1 E_1\right]$$



Third Linear Segment Function

Figure 3.2(a)



Circuit Realizing Above Function

Figure 3.2(b)

$$R_f\,(g_0 - G_1) = -1/.414 \quad \therefore \quad G_1 = 5.83/R_f.$$

The constant terms cancel to give the correct result. At $e_{in} = .414$ there is an increase in slope and so another path to the top amplifier with $e_1 = -e_{in} = -.414$. g_1 is obtained from

$$e_{out} = R_f\,\left(g_1 - \frac{1}{.414\,R_f}\right) e_{in} + R_f g_1 e_1 = 3.41\,e_{in} - 2.41$$

$$\therefore \qquad g_1 = 5.83/R_f.$$

The final realizing circuit is that shown in figure (3.2^b).

There are many realizations better than the one above (see 1, 2 and 3) but this one serves to illustrate. The bias voltages are normally set by potentiometers on line, to circumvent the rather difficult calculation introduced by the change in loading. The important thing to observe is that the function is determined by the m+n+1 resistances and m+n+1 voltage generators, the resistors determining the slopes and the voltages the break points with e_0 or E_0 establishing the initial d.c. value of the curve.

Given a function $y=f(x)$ the problem now reduces to choosing some best set of the 2(m+n+1) parameters or, equivalently, of fitting a 'best' piecewise linear approximation to the curve. The normal criterion function used for 'best' minimizes the maximum deviation of the error. Thus if we are concerned about the approximation on the interval $a < x < b$ the criterion function is

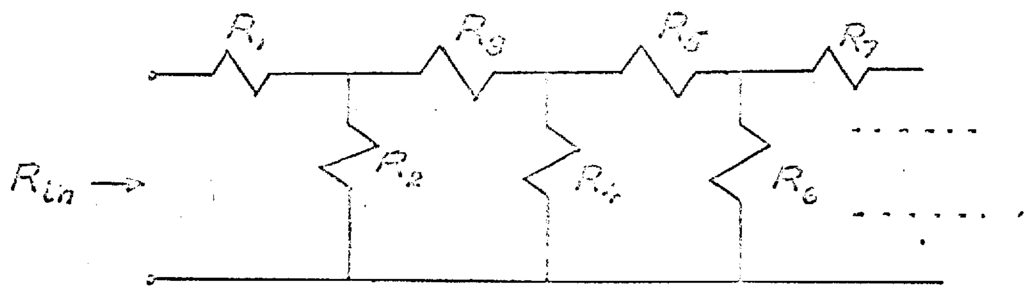
$$\min_{(r_i,R_j,e_i,E_i)} \left\{ \max_{a<x<b} |f(x) - L(x)| \right\}$$

where $L(x)$ is our piecewise linear function. The problem of getting an analytic solution for the parameters is very difficult and as yet has not been solved satisfactorily. Many heuristic methods (see 1 and 4) have been proposed, one being to place the allowable error bounds on a graph of the function and then draw lines such that on every linear segment this maximum error is achieved at least three times. At the moment there seems to be no more attractive solution and as this one suffices for most practical applications it is pointless to pursue a truly optimal solution further.

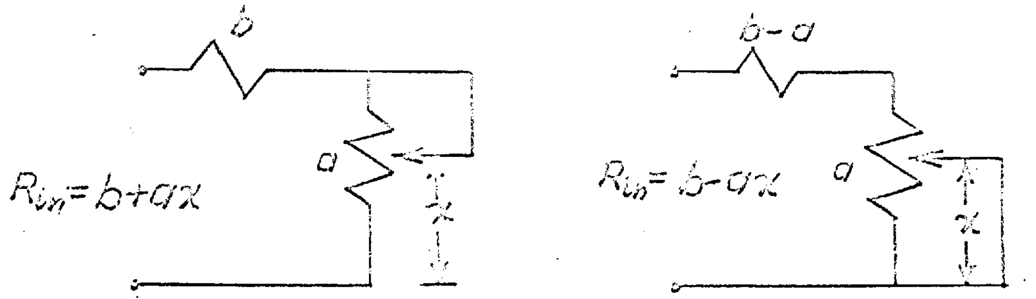
The method extends to two or more dimensions although the hardware involved increases exponentially as may be expected. For a somewhat more elegant method of realizing multi-dimentional functions using a combination of diode generators and logic nets, see the paper by Wilkinson[4].

3.2 Other Analogue Methods

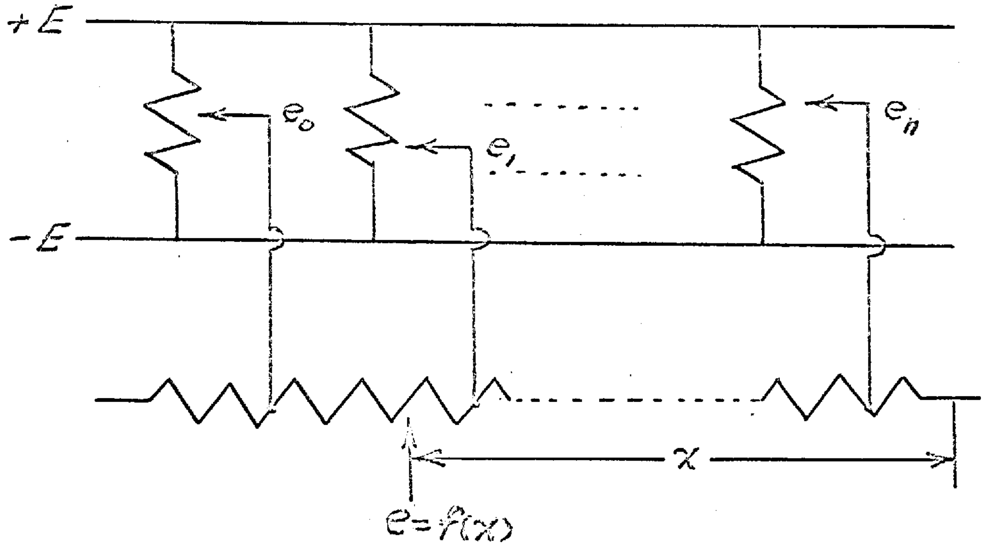
The main method for generating functions other than diode networks is the use of purely resistive networks. There are two basic design philosophies used here. The first uses an iterative resistive net with variable resistors (potentiometers) which are set by the value of the abscissa. Two terminals of the network are designated the input; the impedance looking in from these terminals is then a function of the potentiometer setting. The problem is to determine the resistances in order to give the desired function. The most common iterative net used is the ladder network shown in figure (3.3^a). The input impedance to this net will be given by the expression on the next page where the fraction is continued until the network terminates.



Ladder Network
(a)



Circuits for Obtaining
Linear Functions
(b)



Tapped Function Generator
(c)

Figure 3.3

$$R_{in} = R_1 + \cfrac{1}{\cfrac{1}{R_2} + \cfrac{1}{R_3 + \cfrac{1}{\cfrac{1}{R_4} + \cfrac{1}{R_5 + \ldots\ldots}}}} \qquad (3.3)$$

The domain of the function it is desired to realize is normalized to $0 \leqslant x \leqslant 1$. For f to be obtained from a ladder network it must be expandable in a form like (3.3) where all the R_i are linear functions of x and are positive over the range of x. This is a very severe restriction on f making this method useful only in special cases. Figure (3.3^b) shows how the linear functions b+ax and b-ax may be realized by a resistor and a servo-potentiometer. For some examples of functions realized by this method see reference 1.

A more practical method of obtaining functions by resistive networks is illustrated in figure (3.3^c). This again is a linear segment generator. x is set on a servo-pot which has a number of fixed voltages established on it through the secondary potentiometers, acting as voltage dividers. As x goes from one tap point to the next, the voltage will vary linearly between the set voltages. Loading on the terminal e will cause a non-linear variation which may be accounted for when setting the original voltages.

Both these methods suffer from the need for a servo driven potentiometer as their basic component. This requirement not only increases the cost, but also greatly reduces the speed of operation of such function generators. Because of this they have been going out of style lately, being replaced by the simpler and more rapid diode generators.

Function generators using other methods employing such apparatus as non-linear potentiometers, cathode-ray tubes, curve followers etc., may be found in the references already cited. These are useful for particular application but are not general enough to warrant a description here.

3.3 Digital Methods

If a digital method is to be used both the range and the domain must be discretized. A digital function differs from the discrete functions mentioned previously in that the value of the function is constrained to one of a finite set of specified values (the quantization levels). This implies the following definition.

Definition: A digital function is a mapping from a finite set of points X into a finite set of points Y.

If the number of elements in X is n and the number of elements in Y is k then there are exactly k^n possible functions from X to Y. The elements of X may be put into correspondence with the integers 0,1,..... n-1 and Y with 0,1,.....k-1. The finiteness of all the entities involved introduces many new and interesting problems. The first of these illustrates the danger in using the assumption that the same methods can be used to realize digital functions as are used to realize discrete functions.

In calculating a function by a computer, the function is normally resolved along some spectral axes and the final value taken as the accumulated sum of the component values. Assume a functional expansion of the form

$$f(x) = \sum_{i=0}^{k-1} a_i f_i(x) \tag{3.4}$$

where $f_i(x)$ are base functions, a_i being the expansion coefficients with respect to the basis. The a_i are real numbers and the $f_i(x)$ real functions. All numbers must be quantized before being put into the computer, implying that the $f_i(x)$ will be quantized functions. Denote the quantization operator by square brackets$[\ \ ]$. The function produced by the computer will be

$$f'(x) = \sum_{i=0}^{k-1} \left[a_i\right] \left[f_i(x)\right] . \tag{3.5}$$

Ideally we should like $f'(x)$ to be equal the quantized value of $f(x)$ in equation (3.4). This will not be true in general. For if the functions $f_i(x)$ and the coefficients a_i are each represented by n binary bits, the $(n+1)^{th}$ being neglected, then each multiplication and addition may make the last bit in the result erroneous. This assumes that subtractions have somehow managed to be avoided; otherwise the result may be reduced to nonsense. As there are k multiplications and k-1 additions in formulating the function, the final value obtained for $f'(x)$ may be as much as 2k-1 levels or $\log_2$ (2k-1) bits, in error. Falling back on the modern addage 'if you have enough bits it will not matter' is not really satisfying. Yet the alternative seems to be to discard all of spectral analysis.

The situation may be salvaged to some extent if a set of integer base functions, f_i, (for example the Walsh function) is employed. Returning to the equations for $\underline{c}$ and H on page 36, if f, the expansion functions h_i, and the weighting function p, are all integer, then the vector $\underline{c}$ and the matrix H are integer. The determinant of H will also be an integer as well as all its minors and co-factors. Therefore H^{-1} may be represented as an integer matrix $(H^{-1})'$ divided by the determinant $|H|$. The expansion coefficients are

$$\underline{a} = \frac{(H^{-1})'}{|H|} \quad \underline{c} = \frac{\underline{a}'}{|H|}$$

where the vector $\underline{a}' = (H^{-1})\ \underline{c}$ has integer components. The approximating function f' is

$$f'(x) = \sum_{i=0}^{k-1} a_i h_i(x) = \left\{ \sum_{i=0}^{k-1} a_i' h_i(x) \right\} / |H|.$$

Hence it is possible to perform integer operations throughout until the last step, where a division is necessary. This will reduce the error in realization considerably. A short example using the Walsh functions is discussed below.

All the Walsh functions of a given order may be realized by a net of modulo two adders. If all the $\binom{n}{r}$ functions containing r variables have been obtained, then those containing r+1 variables can be formed by adding $\binom{n}{r+1}$ modulo two gates with one input from the variable set and the other from one of the functions of r variables. As the $\binom{n}{1}=n$ Walsh function consisting only of the x_i are already available, the total number of gates required will be

$$\sum_{r=2}^{n} \binom{n}{r} = 2^n - (n+1).$$

Suppose a function, defined over the first 8 integers, is required. The example done here was chosen from a table of random digits but may, in a practical case, be a coding of the digits, or perhaps be some obscure compensation function. The function is

$$(0,1,2,3,4,5,6,7) \rightarrow (9,6,1,8,9,4,1,1).$$

Using the functions given in figure (2.3) and the coefficient equation

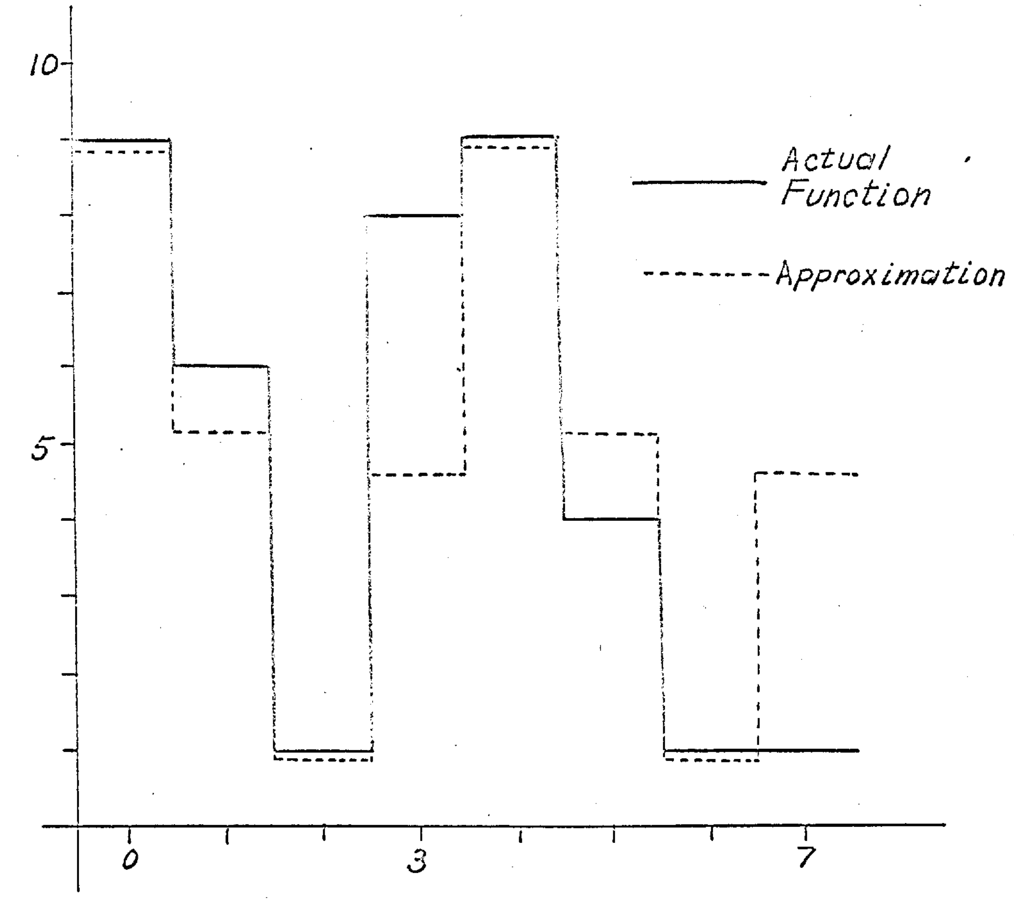
$$a_i = \frac{1}{2^3} \sum_{i=0}^{7} f(n)W_i(n)$$

the values for the coefficients are calculated as

$$a_0 = -\frac{39}{8} \qquad a_4 = -\frac{9}{8}$$
$$a_1 = -\frac{1}{8} \qquad a_5 = \frac{9}{8}$$
$$a_2 = -\frac{17}{8} \qquad a_6 = \frac{5}{8}$$
$$a_3 = -\frac{15}{8} \qquad a_7 = -\frac{5}{8}$$

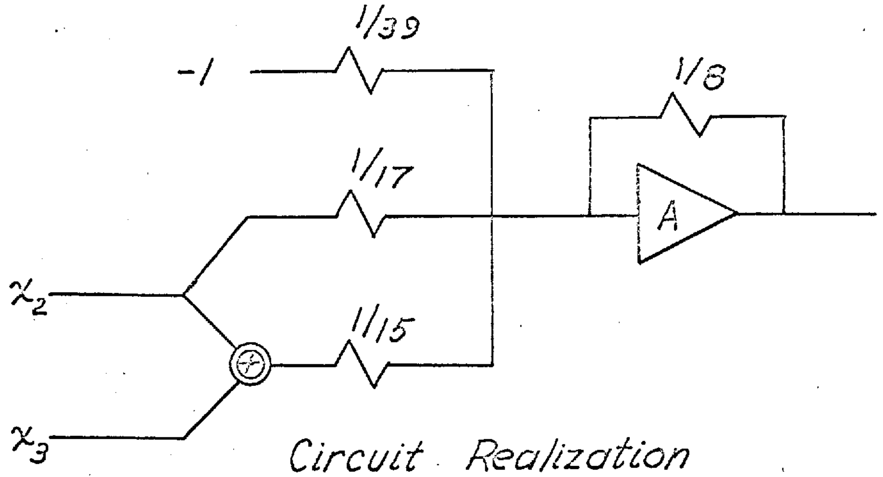
If all expansion terms are used then the function will be realized exactly. This would require eight coefficients and would result in no saving of hardware from merely storing the functional values directly. Such a result is expected as no approximations have been made. Hence, no simplification has resulted.

Figure (3.4^a) shows the function and the approximation to the function obtained by taking the three most significant coefficients a_0, a_2, a_3. This approximation is not disastrously bad considering the irregulatiry of the initial function, although it shows a wide variance for two of the values. A circuit for realizing the approximate funct-ion is given in (3.4^b). The input variables x_1, x_2, x_3 come from an analogue to digital converter, the digital to analogue conversion being carried out by the summing amplifier. Note how it is possible to use



Approximation of a Discrete Function Using Three Walsh Functions

(a)



Circuit Realization

(b)

Realization of Discrete Quantized Functions Using the Walsh Functions

Figure 3.4

integer arithmetic throughout until the last stage when the normalizing constant $\frac{1}{8}$ must be introduced. This normalization is easily performed through the summing amplifier. The logic levels for x_1, x_2 and x_3, as well as the logical function used, have been converted to +1, -1 in order to give the Walsh functions directly.

This procedure is still not completely satisfying, as an analogue summing amplifier is necessary to reconstitute the function. Its presence cannot be escaped if a spectral resolution of the function is used; for in that case the final function is always made up by summing certain proportions of the base functions. The sum can be performed digitally although this increases the cost and complexity of the realization by an order of magnitude.

Many other methods of obtaining function using 'hybrid' techniques have been proposed[17], but most of them require either an adder or an integrator in the last stage. To get rid of this analogue hanger-on it is necessary to go more deeply into digital analysis and to introduce boolean functions along with all their concomitant problems.

CHAPTER IV

Boolean Function Realization

Boolean functions have been receiving increased attention in the last few years due to the advent of the digital computer and the renewed interest in switching theory. Their application in function generation lies in the fact that any discrete function may be realized by a set of boolean functions. That this is so will be demonstrated in the next chapter. Here we will confine ourselves to a fairly thorough study of boolean functions and show how they may be realized using practical circuit elements. A few theorems on minimizing networks will be proved along with some examples.

4.1 Boolean Algebras

To date, our studies have been confined to functions, members of the real Hilbert space; or in the case of discrete functions, members of a real space defined over a denumerable set of points. All realizations have depended on the linearity of the space, i.e., if x and y are members of the space then ax+by is also a member, for all real a and b. This linearity makes it possible to approximate a function arbitrarily closely by means of a linear combination of base functions. The concept of a spectra and expansion coefficients to represent the functions is then possible. As will be shown below all this is lost when dealing with boolean functions, as these form a linear space only in a very limited sense.

A boolean algebra[18] is defined as a set of elements $\mathcal{U}$ along with two binary operations '+' and '.' and a unary operation $\bar{\ }$ such that the following five axioms are satisfied for any three elements A, B and C of $\mathcal{U}$

1. $A + B = B + A$, $A.B = B.A$
2. $(A + B) + C = A + (B + C)$, $A.(B.C) = (A.B).C$
3. $(A + B).B = B$, $A.B + A = A$
4. $A.(B + C) = A.B + A.C$, $A + B.C = (A + B).(A + C)$
5. $A.\bar{A} + B = B$, $(A + \bar{A}).B = B$

Added to these are deMorgan's two laws set down here for convenience

$$\bar{A}.\bar{B} = \overline{A + B} \quad \text{and} \quad \bar{A} + \bar{B} = \overline{AB} .$$

These are a consequence of the axioms and may easily be obtained from them.

The most convenient way to visualize these axioms is to think of the members of $\mathcal{U}$ as ordinary sets where '+' is the union operation, '.' is the intersection, and $\bar{\ }$ is the complementation of the sets under study. The algebra of sets is a special case of a boolean algebra, other examples occurring in probability theory, topology, logic etc.

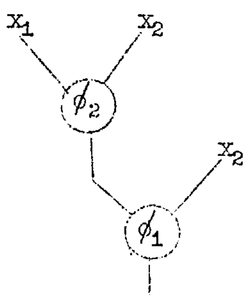
We will restrict ourselves to logic, identifying the elements of $\mathcal{U}$ with events that may be either true (1) or false (0) and the connections '+' and '.' with the logical 'or' and 'and' respectively; $\bar{\ }$ will be the logical negation. In the following the '.' will be omitted, the operator being denoted by the conjunction of the two elements in question. The logical algebra satisfies the axioms given above meaning that these axioms may be used in simplifying any expressions that occur in the following.

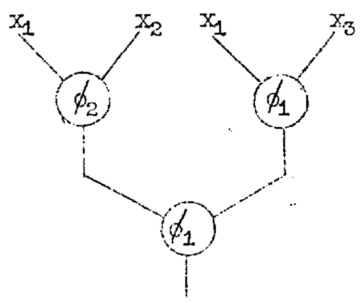
A boolean function of n variables $x_1, x_2 \ldots\ldots x_n$ is any logical expression containing the n variables. The set of all functions of n variables forms a boolean algebra. A boolean function will be denoted by

$$f(x_1, x_2 \ldots\ldots x_n).$$

There are only 2^{2^n} functions of n variables, for the arguments of f can have only 2^n distinct combinations and, as the function can have only the values 0 or 1 (corresponding to the logical expression being false or true) there are at most 2^{2^n} functional combinations that may be selected. There are infinitely many logical expressions of n variables implying that there must be infinitely many expressions for some particular functions. The central problem in switching theory is to find the simplest expression for a given function. As yet, the only solution to it has been to develop a systematic search procedure[28].

The non-uniqueness of the logical representation, along with the absence of what might be coefficients, thwart any attempt at resolving a boolean function into a spectrum in the sense of the previous discussion. It is still possible to talk of bases but first it is necessary to define what is meant by a superposition[19]. The best definition is to give an example. If we have two functions in a boolean algebra, say, $\phi_1(x_1, x_2)$ and $\phi_2(x_1, x_2)$, a superposition of these function might be $\phi_1(\phi_2(x_1, x_2), x_2)$ or $\phi_1(\phi_2(x_1, x_2), \phi_1(x_1, x_3))$ etc. Geometrically, if the ϕ_i are thought of as nodes of a network, then a superposition is equivalent to taking the output of some nodes and connecting them to the inputs of others. The two above examples are represented below.





A basis for a boolean algebra is defined as a set of functions $\phi_1(x_1, \ldots\ldots x_{k_1}), \phi_2(x_1, \ldots\ldots x_{k_2}) \ldots\ldots, \phi_r(x_1, \ldots\ldots x_{k_r})$ such that any boolean function of n variables may be realized by a superposition of them. The functions ϕ_i are called base functions. Post[20] has shown that a set of functions form a basis if there exist superpositions of these functions realizing the particular functions $\bar{x}_1$ and $x_1 x_2$. Evidently these two functions will form a basis.

Other examples of bases are

$$\bar{x}_1, \; x_1 + x_2$$

$$\overline{x_1 + x_2}$$

$$1, \; x_1 x_2, \; x_1 \bar{x}_2 + \bar{x}_1 x_2 \; .$$

The first is the dual basis to $\bar{x}_1, x_1 x_2$. The second is the nor base consisting of only one function. If the function $\overline{x_1 + x_2}$ is called $\phi(x_1, x_2)$ then $\phi(x_1, x_1) = \bar{x}_1$ and $\phi(\phi(x_1, x_1), \phi(x_2, x_2)) = x_1 x_2$. The third set of base functions is the ring sum or Huffman[21] basis. Note that the last function given is true if and only if one of the variables is true. This corresponds to the linear function mentioned previously in connection with the Walsh functions (page 40). This basis is more usually represented as

$$1, x_1 x_2, x_1 \oplus x_2$$

and is a basis as $x_1 x_2$ is already included and $x_1 \oplus 1 = \bar{x}_1$. Its great advantage is that it allows one to express any function without having to use any complementation operations. Many other bases exist and are finding increasing use in realizing boolean functions for switching networks.

4.2 Canonical Forms

The most common way of representing a boolean function is by means of the truth table. This is a tabulation of the output for all possible combinations of the input variables, and corresponds to the look-up table for ordinary functions. The truth table leads almost immediately to the 'conjunctive normal form'. Consider the function of n-variables

$$(x_1 \oplus \bar{a}_1)(x_2 \oplus \bar{a}_2) \;\ldots\ldots\; (x_n \oplus \bar{a}_n) \qquad (4.1)$$

where the a_i are constants, either 0 or 1. This represents a conjunction of the n variables (e.g., $(x_1 \oplus 1)(x_2 \oplus 0)(x_3 \oplus 1) = \bar{x}_1 x_2 \bar{x}_3$). It is true for only one value of the coefficients, namely $x_1 = a_1$, $x_2 = a_2 \ldots\ldots x_n = a_n$, so its truth table will have only one non-zero entry. In the following $\underline{\mu}_i$ will be used to represent a constant n vector of 0's and 1's, such that its binary number equivalent is i. For example, if functions of 4 variables are under discussion then $\underline{\mu}_7 = (0, 1, 1, 1)$, and so on.

Any function $f(x_1, x_2 \ldots\ldots x_n)$, in truth table form, is true over only a subset of the vector $\underline{\mu}_i$, $i = 0 \ldots\ldots 2^n-1$. The conjunction (4.1) is true on one particular $\underline{\mu}_i$ only. Hence the representation

$$f(\underline{x}) = \sum_{i=0}^{2^n-1} f(\underline{\mu}_i)(x_1 \oplus \bar{\mu}_{i1})(x_2 \oplus \bar{\mu}_{i2}) \ldots\ldots (x_n \oplus \bar{\mu}_{in}) \tag{4.2}$$

where μ_{ij} is the j th component of $\underline{\mu}_i$, and the $\sum$ sign implies an 'or' summation of all the functions following, is valid. That this expression is correct can be verified by substituting an arbitrary constant $\underline{\mu}_j$ into $f(\underline{x})$. Then, only one conjunction on the right is non-zero and the expansion reduces to the identity $f(\underline{\mu}_j) = f(\underline{\mu}_j)$. This expansion is known as the 'conjunctive normal form'.

The dual expansion to this, the 'disjunctive normal form' is based on the fact that a disjunction of the form

$$(x_1 \oplus a_1) + (x_2 \oplus a_2) + \ldots\ldots + (x_n \oplus a_n)$$

is false for only one $\underline{\mu}_i$, namely $\underline{\mu}_i = (a_1, a_2 \ldots\ldots a_n)$. The canonic expansion

$$f(\underline{x}) = \prod_{i=0}^{2^n-1} (f(\underline{\mu}_i) + \sum_{j=1}^{n} x_j \oplus \mu_{ij})$$

is immediate; for if $\underline{\mu}_k$ is substituted on the left then all the disjunctions on the right are true except for the one involving $\underline{\mu}_k$, and the expression reduces to $f(\underline{\mu}_k)$ yielding an identity.

Both these expansions use the redundant basis $\bar{x}_1$, $x_1 x_2$, $x_1 + x_2$. The use of deMorgan's laws can easily change this to an irredundant basis. Using the first law in the form

$$\sum_{i=0}^{2^n-1} f_i = (\prod_{i=0}^{2^n-1} \bar{f}_i) \oplus 1$$

equation (4.2) becomes

$$f(\underline{x}) = \left\{ \prod_{i=0}^{2^n-1} ((f(\underline{\mu}_i) \prod_{j=1}^{n} (x_1 \oplus \bar{\mu}_{ij})) \oplus 1 \right\} \oplus 1$$

(note that $f(\underline{x}) \oplus 1 = \bar{f}(\underline{x})$. This notation is used as it is somewhat clearer than drawing bars over the rather complicated expressions involved.) This form is inconvenient for most purposes and (4.2) is normally used.

The last canonical form is the 'ring-sum'[22] or Huffman[21] normal form. First, note that two functions f and g are equal if and only if $f \oplus g = 0$. This is obvious, for if $f = g$ then the two truth tables must be the same and the condition is satisfied. Consider the function of two variables defined by

$$a_3x_1x_2 \oplus a_2x_1 \oplus a_1x_2 \oplus a_0 \qquad (4.3)$$

where the a_i are constants. Constants belonging to conjunctions of two variables are said to belong to the second rank, ones belonging to one variable to the first rank etc. a_3 belongs to the second rank, a_2 and a_1 to the first rank and a_0 to the zeroth rank. For n-variables there are $\binom{n}{r}$ conjunctions of order r, and hence, in the general expansion similar to the one above $\binom{n}{r}$ constants of rank r. In the total expansion there will be

$$\sum_{r=0}^{n} \binom{n}{r} = (1+1)^n = 2^n \text{ constants.}$$

If all the $a_i \neq 0$ for a function g defined in the above fashion, then $g \neq 0$. For suppose the first non-zero constant is a_k in rank s. Without loss of generality the conjunction associated with a_k can be taken as $x_1 x_2 \ldots\ldots x_s$, as it only takes a permutation of the variables to bring this about. If a constant vector $\underline{\mu}_i = (1, 1, \ldots\ldots \underset{s}{1}, 0, \ldots\ldots 0)$ is chosen, then all other members of the s'th rank will be zero, as will all members of higher ranks. All the lower ranks are already zero, therefore, $g(\underline{\mu}_i) = 1$ and the function is not identically zero.

Furthermore, if two functions g and f have different coefficients a_i and b_i then $g \oplus f \neq 0$, i.e., $g \neq f$; for if a_k and b_k are different then the conjunction belonging to these will appear in $g \oplus f$. There are 2^n coefficients a_i and hence 2^{2^n} different functions can be defined in this manner. But there are only 2^{2^n} functions of n variables. Therefore every function of n variables is expressible in this form. This is the 'ring-sum' normal form.

The determination of the coefficients a_i for this form is somewhat more difficult than for the preceeding cases. A constant $\underline{\mu}_i$ will be said to be of rank k if it has k 1's and n-k 0's. Assume an expansion in n-variables of the form given in equation (4.3)

$$f(\underline{x}) = a_{2^n-1} x_1 x_2 \ldots\ldots x_n \oplus a_{2^n-2} x_1 x_2 \ldots\ldots x_{n-1} \oplus \ldots\ldots$$

$$\ldots\ldots \oplus a_2 x_{n-1} \oplus a_1 x_n \oplus a_0 .$$

Let the values in the truth table be expressed by $f(\underline{\mu}_i) = b_i$. The coefficients a_i will turn out to be linear combinations of the b_i. This can be shown inductively; for assume $\underline{\mu}_i$ is of rank k. Then on substituting $\underline{\mu}_i$ in the above equation all conjunctions of rank $> k$ are zero. There is one conjunction true in rank k, $\binom{k}{1}$ conjunctions true in rank k-1, $\binom{k}{r}$ conjunctions true in rank r, and $\binom{k}{k}$ conjunctions true in rank

0. In all there are 2^k conjunctions true. Therefore

$$f(\underline{\mu}_i) = b_i = \sum_{i=0}^{2^n-1}{}' \, a_{r_i} \tag{4.4}$$

where the set r_i is some specified subset of $(0, 1, 2, \ldots\ldots 2^n-1)$ of size 2^k. In particular $f(\underline{0}) = b_0 = a_0$. This is the beginning of an induction. If a formula for the a_i in terms of the b_i has been obtained for all a_i of rank $< k$, then the formula for elements of rank k+1 follows immediately from (4.4), since only one of the terms on the right is of rank (k+1). This may all seem rather obscure due to the notational difficulties involved. An example should serve to clarify the procedure.

Consider the truth table below for a function, f_1, of three variables

x_1	x_2	x_3	f_1	f_2
0	0	0	b_0	1
0	0	1	b_1	1
0	1	0	b_2	0
0	1	1	b_3	0
1	0	0	b_4	0
1	0	1	b_5	1
1	1	0	b_6	1
1	1	1	b_7	0

Assume an expansion for f_1 of the form

$$f_1(\underline{x}) = a_7x_1x_2x_3 \oplus a_6x_1x_2 \oplus a_5x_1x_3 \oplus a_4x_2x_3$$
$$\oplus\, a_3x_1 \oplus a_2x_2 \oplus a_1x_3 \oplus a_0.$$

Then

$$\begin{array}{ll}
f_1(0,0,0) = b_0 = a_0 & a_0 = b_0 \\
f_1(0,0,1) = b_1 = a_1 \oplus a_0 & a_1 = b_1 \oplus b_0 \\
f_1(0,1,0) = b_2 = a_2 \oplus a_0 & a_2 = b_2 \oplus b_0 \\
f_1(1,0,0) = b_4 = a_3 \oplus a_0 & a_3 = b_4 \oplus b_0 \\
f_1(0,1,1) = b_3 = a_4 \oplus a_2 \oplus a_1 \oplus a_0 & a_4 = b_3 \oplus b_2 \oplus b_1 \oplus b_0
\end{array}$$

etc.

If the vectors μ_i are arranged according to their rank in the truth table, then, with a little practice, it becomes an easy matter to write down this canonical expansion by inspection.

For a function like f_2 in the table above, the canonical expansion in the three different forms is

1) conjunctive normal form

$$f_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$$

2) disjunctive normal form

$$f_2 = (x_1 + \bar{x}_2 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$$

3) ring-sum normal form

$$f_2 = 1 \oplus x_2 \oplus x_1 \oplus x_1x_3.$$

One of the greatest advantages of the last form, aside from the fact that it contains no negations, is, that for any function f the expectation of the length of the expression describing f is smallest. Also it lends itself easily to analytic manipulations whereas the other two are rather cumbersome. Throughout the remainder of the chapter, these three forms will be used interchangeably.

4.3 Universal Nets

A universal net is a network of circuit elements with 2^n variable parameters $\{a_i\}$ and n input variables $x_1, x_2 \,.....\, x_n$ such that, for any two different sets of parameters $\{a_i\}$ and $\{b_i\}$ the function realized by the net $f(x_1, x_2 \,.....\, x_n)$ is different. As there are 2^{2^n} choices of the set $\{a_i\}$ the net must be capable of realizing all functions of n variables.

The basic theorem used in obtaining all the following nets is the Shannon[22] expansion theorem. Defining the function of (n-1) variables

$$f(0, x_2, x_3 \,.....\, x_n) = f_0$$

$$f(1, x_2, x_3 \,.....\, x_n) = f_1$$

this theorem states that

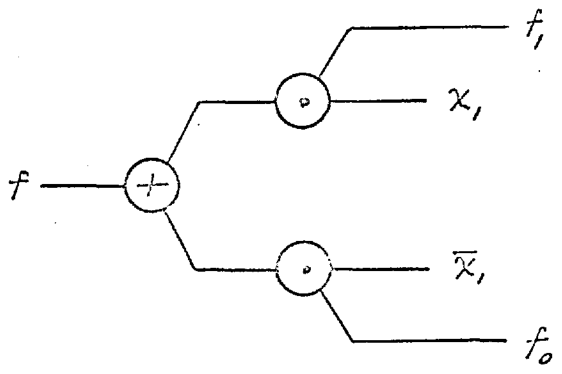
$$f(x_1, x_2 \,.....\, x_n) = \bar{x}_1 f_0 + x_1 f_1 . \tag{4.5}$$

Its validity is obvious, as substituting $x_1 = 0$ or 1 on the right will show. f_0 and f_1 may in turn be expanded giving

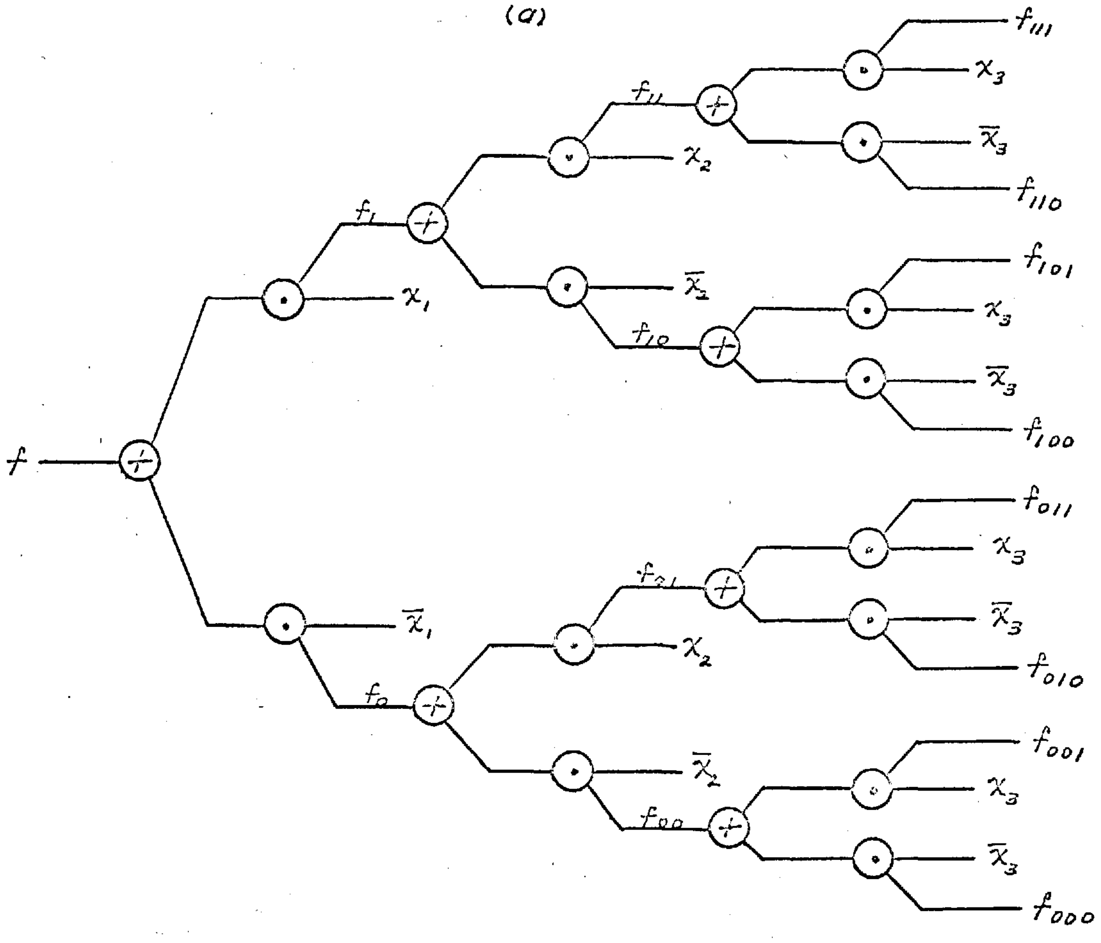
$$f(x_1 x_2 \,.....\, x_n) = \bar{x}_1 \bar{x}_2 f_{00} + \bar{x}_1 x_2 f_{01} + x_1 \bar{x}_2 f_{10} + x_1 x_2 f_{11} .$$

The process may be continued until all the variables are exhausted at which time the coefficients will be $f(\underline{\mu}_i)$ and the expansion will be identical to the conjunctive normal form.

The formula (4.5) allows one variable at a time to be separated from the function producing in each case two functions of fewer variables. Confining ourselves to two input 'and' and 'or' gates the basic recurrence block of a universal net would be as shown in figure (4.1^a). f_0 and f_1 can then be expanded separately and the process repeated until



Base Structure for 'and''or' Net
(a)



Universal Net Using two Input 'and''or' Gates
(b)

Figure 4.1

the universal net of (4.1^b) is obtained. Here the set $\{a_i\}$ are the constants set on the 2^n terminating branches labelled $f(\underline{\mu}_i)$.

The total number of elements in the net is of interest as this is a measure of the complexity of the realization. Let $g(n)$ be the total number of elements in a universal 'and' 'or' net for all functions of n variables. Then the recurrence relation

$$g(n+1) = 2g(n) + 3$$

follows immediately. Using $g(2) = 3$ the solution

$$g(n) = 3(2^{n-1} -1)$$

is obtained by a simple induction. This is the maximum number of elements of this kind that would be necessary to realize any function of n variables. The constants $f(\underline{\mu}_i)$ are introduced at the far right of the net where the resolving process has reduced the problem to realizing functionsof one variable. In the last stages the resolution looks like

$$f(\underline{\alpha},x_n) = \bar{x}_n \; f(\underline{\alpha},0) + x_n \; f(\underline{\alpha},1)$$

where $\underline{\alpha}$ is a constant n-1 vector. According to the values of $f(\underline{\alpha},0)$ and $f(\underline{\alpha},1)$ the input is $0,\bar{x}_n,x_n$, or 1. This normally leads to some simplification as theorems from the algebra, such as $\bar{x}_1 x_2 + x_1 x_2 = x_2$ and $\bar{x}_1 \; 1 + x_1 x_2 = \bar{x}_1 + x_2$, help to eliminate many of the elements.

Throughout the above discussion it has been assumed that the allowable fan-out (number of output leads from each element) is only one. If this restriction is relaxed, further simplification in the final realization results, as then many of the functions of fewer variables, found towards the end of the net, will be identical and can be realized by the same network. If this line of attack is developed a little further an interesting and quite significant result is obtained; but first it is necessary to discuss a universal net of a different kind.

A universal net of the second kind is defined as a network of circuit elements that explicitly realizes all functions of n variables. These networks may be obtained by first obtaining networks that realize all conjunctions of n-variables, and then using 'or' gates to produce the required functions as represented in the 'conjunctive normal form'. If a network that realizes all the 2^{n-1} conjunctions of n-1 variables has been obtained, it may be extended to n variables merely by adding 2.2^{n-1} 'and' gates, taking the 2^{n-1} outputs of the n-1 net and forming the conjunction of these with x_n and then with $\overline{x}_n$. If g(n-1) is the number of and gates in the n-1 net then

$$g(n) = g(n-1) + 2^n.$$

g(2) = 4 as there are 4 conjunctions of two variables. Therefore the value of g(n) is given by

$$g(n) = 4(2^{n-1} - 1).$$

Including the conjunctions of fewer than n variables available in the net, $4(2^{n-1} - 1)$ of the total of 2^{2^n} functions of n variables have already been found. As all the conjunctions in the conjunctive normal form are now available it is necessary only to form the disjunctions to obtain all the functions of n variables. If all the functions consisting of k terms have been obtained, then each function of k+1 terms may be obtained by adding one more 'or' gate with one input from a k term function and another from the set of conjunctions. Therefore at most $2^{2^n} - g(n)$ 'or' gates will be required or 2^{2^n} gates in all.

Before going further a result of Krichevskii[19] and Lupanov[23] is quoted. As the number of arguments increases the total number of 'bad', to quote Lupanov, functions in the class of all functions of n-variables becomes the dominating class. 'Bad' here means functions that are difficult to realize in any specified finite basis. Analytically the notion of dominating class implies that

$$\lim_{n\to\infty} \frac{\text{number of 'bad' functions}}{\text{total number of functions}} = 1.$$

This seems reasonable enough, only stating that most functions of many variables are hard to realize regardless of the basis chosen. The result proved by the two authors above is, that given a base set $\phi_1, \ldots\ldots \phi_n$ and costs $P_1, \ldots\ldots P_n$ associated with the base functions, the minimum cost of realizing the most difficult function with respect to the basis is asymptotically equal to the expressions below.

$$\max_{f}.(\min_{\{\phi\}}.(\text{cost } (f))) \sim \frac{\mu\, 2^n}{\log_2 n} \tag{4.6}$$

μ is a function of the costs P_i and the base functions ϕ_i

$$\mu = \min_{k_i \geqslant 2} \left(\frac{P_i}{k_i - 1}\right)$$

where k_i is the number of arguments of ϕ_i. The operations on the left of (4.6) are equivalent to first realizing all functions of n variables at a minimum cost by superposition of the base functions, and then taking the most costly of these.

If the cost $P_i = 1$ and the two input gates above are used then $\mu = 1$ and the cost given by (4.6) will be equal to the number of elements required. As most functions are like this most difficult function the result that the number of elements necessary to realize most functions of n variables $\sim 2^n/\log_2 n$ is immediate.

Returning to the 'and' 'or' nets above, suppose that a function is to be realized by a combination of nets of the first and second kinds. Extract k of the n variables using a net of the first kind. This requires $3(2^k-1)$ elements and leaves 2^k functions of n-k variables to be obtained. These functions can be obtained from a net of the second kind requiring at most $2^{2^{(n-k)}}$ elements. Therefore, the total number of elements N in the net realizing the function will be less than

$$3.2^k + 2^{2^{(n-k)}} > N$$

or

$$2^{k+2} + (2^{2^{(n-k)}} - 2^k) > N. \qquad (4.7)$$

There will be some best value of k that will minimize N, the number of elements required. Differentiating (4.7) with respect to k and equating to zero yields a transcendental equation in k and n that is very difficult to solve. Instead the heuristic 'choose k to be the smallest value such that

$$2^{2^{(n-k)}} - 2^k \leqslant 0$$

is used. This choice does coincide with the true integer minimum in most cases as working a few examples shows. The above heuristic implies that

$$n-k \leqslant \log_2 k$$

and

$$n-(k-1) \geqslant \log_2 (k-1)$$

or in terms of the number of variables n

$$\log_2 (k-1) + k-1 \leqslant n \leqslant k+\log_2 k. \qquad (4.8)$$

The condition expressed by equation (4.7) becomes

$$N < 2^{k+2}.$$

It will now be shown that using the above value of k the Lupanov bound (4.6) is satisfied for this realizing network. Substituting from (4.8) into the equation for the Lupanov bound yields

$$\frac{2^n}{\log_2 n} \geqslant \frac{2^{(k-1)+\log_2(k-1)}}{\log_2(k+\log_2(k))}$$

$$\text{i.e.} \geqslant \frac{(k-1)2^{k-1}}{\log_2(k+\log_2(k))} \qquad (4.9)$$

Furthermore using l'Hospital's rule it can be shown that

$$\lim_{k\to\infty} \frac{(k-1)}{\log_2(k+\log_2(k))} = \infty .$$

Therefore there exists a finite number M such that

$$\frac{(k-1)}{\log_2(k+\log_2(k))} \geqslant 8 \qquad \text{for all } k \geqslant M.$$

Returning to equation (4.9) the chain of inequalities

$$\frac{2^n}{\log_2 n} \geqslant 8.2^{k-1} = 2^{k+2} > N \qquad \text{for all } k \geqslant M$$

is obtained.

Equation (4.8) ensures that provided n is large enough $k \geqslant M$. Hence the result that the number of elements in the net for realizing any function of n variables is less than or equal to the Lupanov bound is obtained. This in turn implies that it is useless to look for some other method of realizing functions of many variables as, by the theorem, they must have at least the same order of elements as the above realization using 'and' and 'or' gates. This does not imply that all minimizing procedures should be discarded, but merely that the saving in the number of elements required will only be incremental and savings in orders of magnitude cannot be expected by any other design procedure that may be adopted. Also, as Lupanov's theorem is only an asymptotic result, it suffers from the failing of most results of its kind in that n must be very large before it is satisfied. Therefore, it is still worthwhile to look for minimizing procedures for universal nets. These are discussed in section 4.5.

A result, similar to the one above, may be demonstrated for all nets that are discussed in the following. As the development of a proof for each individual case is rather tedious and follows the approach already given, it will be omitted, although the result may be tacitly assumed.

4.4 Universal Nor Nets

Circuits using 'nor' gates as the basic element have received increasing attention[24,25,27] during the last few years. This stems from the fact that they have a particularly simple transistor realization, requiring only one transistor and some resistors. Also they do not require any isolating stages when connected together, as conventional diode 'and' and 'or' gates do. Universal nets for two and three input 'nor' gates are developed below and an example of realizing a specific function is given.

In the diagrams that follow, a 'nor' gate will be represented by a junction with the requisite number of input leads, and a single horizontal output lead. A function written on top of the output lead will represent the output function of that particular gate.

Looking at two input 'nor' gates first, from Shannon's formula, equation (4.5), the identity

$$f(x_1 \;.....\; x_n) = \overline{\overline{\bar{x}_1 f_0 + x_1 f_1}} = \overline{\overline{(\bar{x}_1 f_0)} \quad \overline{(x_1 f_1)}}$$

$$= \overline{(x_1+\bar{f}_0)(\bar{x}_1+\bar{f}_1)} = \overline{\bar{x}_1 \bar{f}_0 + x_1 \bar{f}_1} \qquad (4.9)$$

$$= \overline{\overline{x_1+f_0} + \overline{\bar{x}_1+f_1}}$$

is obtained. The base structure is as shown in figure (4.2^a). On comparing this structure with the 'and' 'or' structure already obtained it is evident that the total net will have exactly the same geometry and therefore contains $3(2^{n-1}-1)$ elements. All the subsequent results proved for the 'and' 'or' net are also immediately applicable to this structure.

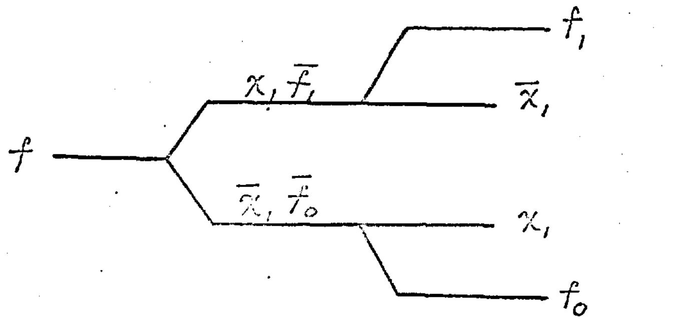
For three input gates first consider expressions of the form $x_1 \bar{f}_1$. This function may be developed as

$$x_1 \bar{f}_1 = \overline{\overline{x_1 \bar{f}_1}} = \overline{\bar{x}_1 + f_1}$$

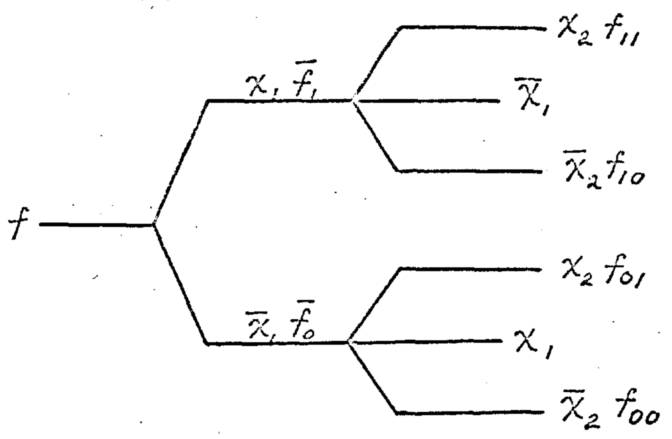
$$= \overline{\bar{x}_1 + \bar{x}_2 f_{10} + x_2 f_{11}} \;.$$

The expression for $\bar{x}_2 \; f_{10}$ and $x_2 \; f_{11}$ can be realized similarly by a three input gate, and the expansion continued until all variables are exhausted. The initial decomposition can be performed by a two input gate according to equation (4.9). The basic structure is shown in figure (4.2^b) and the total expansion out to the fourth level is given in figure (4.3).

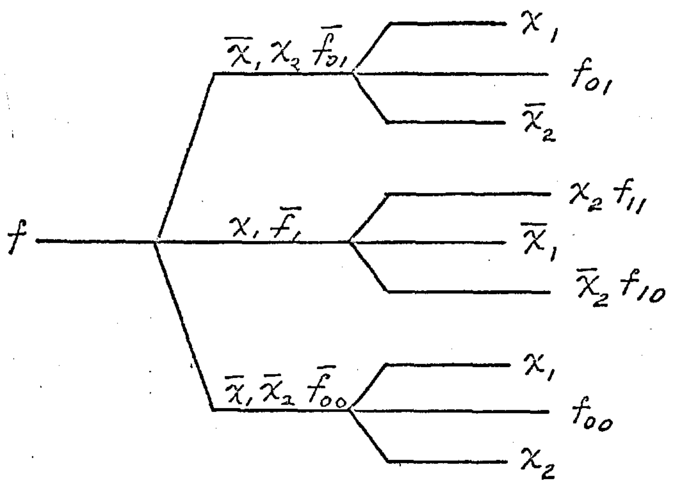
If g(n) is the number of elements in a complete net for n varia-



Two Input Nor Net
(a)



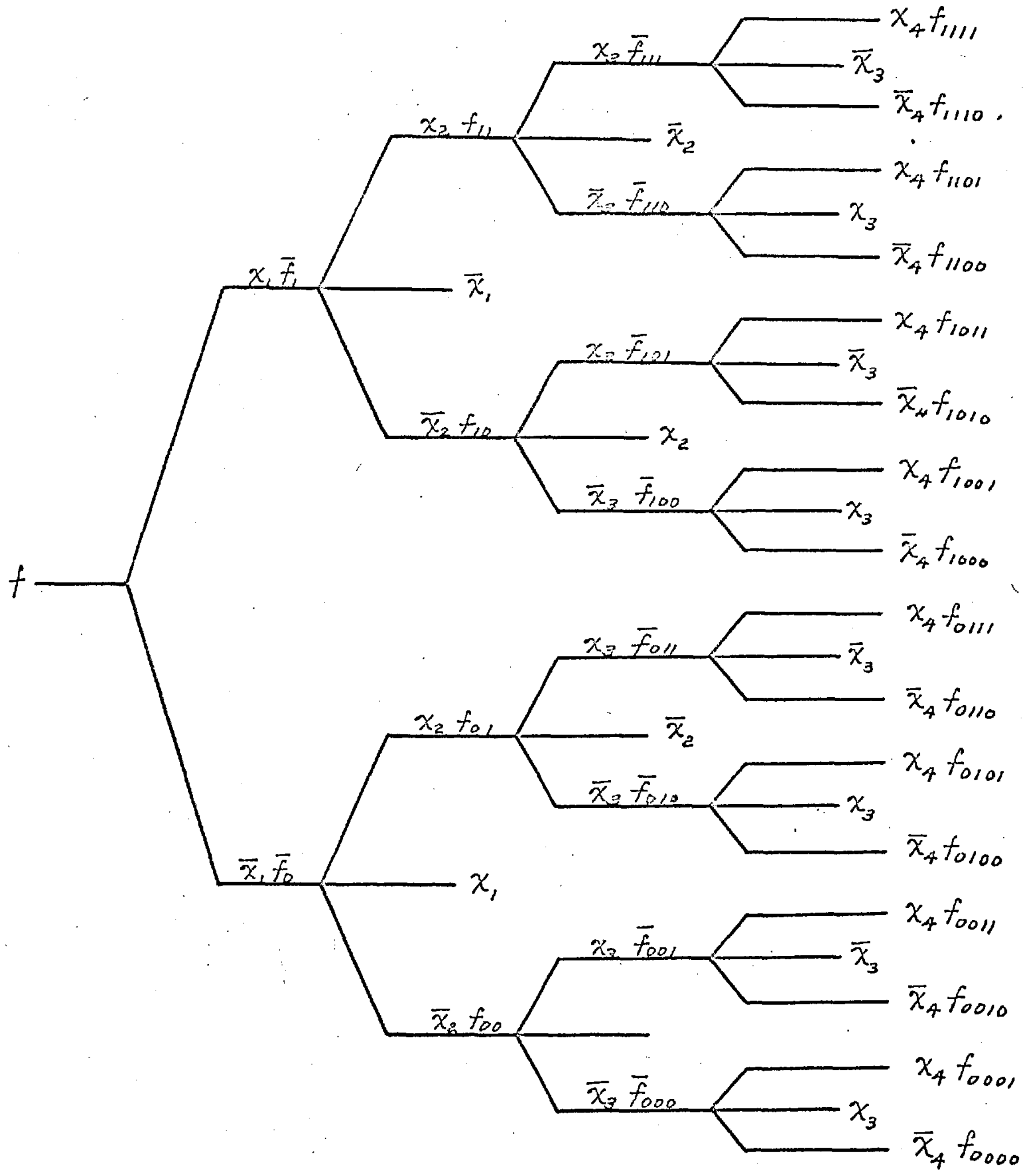
Three Input Nor Net
(b)



Modified Three Input Nor Net
(c)

Base Structures for Universal Nor Nets

Figure 4.2



Universal Net for Functions of Four Variables Using Three Input Nor Gates

Figure 4.3

bles then

$$g(n+1) = 2\ g(n) + 1\ . \qquad (4.10)$$

$$g(2) = 3\ . \qquad (4.11)$$

This equation has a solution

$$g(n) = 2^n - 1$$

as can easily be checked by using (4.11) as a base for an induction using (4.10). Note that if the expansion is carried out to an odd number of levels then the constants $\overline{f}(\mu_i)$ must be substituted, rather than $f(\mu_i)$. The $f(\mu_i)$ appear in the same order as they would appear in a truth table. It is interesting that this net has exactly the same geometry as the threshold function nets discussed by Miyata[26].

The first two input gate can be replaced by a three input gate using a slightly modified form of equation (4.9). Expanding f_0 by (4.9)

$$f_0 = \overline{x_2 \overline{f}_{01} + \overline{x}_2 \overline{f}_{00}}$$

or

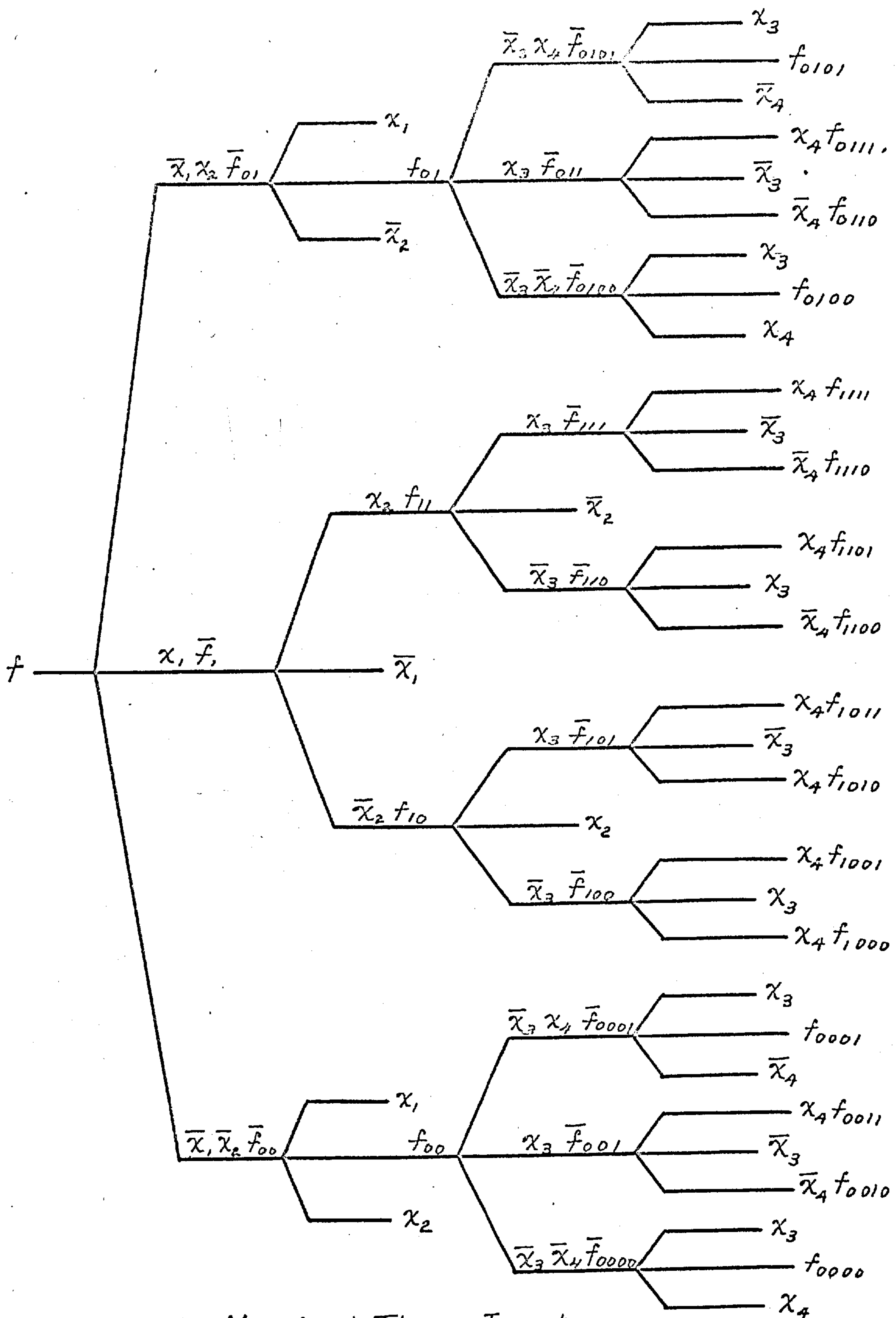
$$\overline{f}_0 = x_2 \overline{f}_{01} + \overline{x}_2 \overline{f}_{00}$$

and substituting this back into (4.9) for f, the expansion formula

$$f = \overline{x_1 \overline{f}_1 + \overline{x}_1 x_2 \overline{f}_{01} + \overline{x}_1 \overline{x}_2 \overline{f}_{00}}$$

is obtained. The first term may be expanded into a net like the previous case and each of the other two terms may be obtained from single three input nor gates. The base structure is shown in figure (4.2^c) and the expanded net out to four variables in figure (4.4).

Finding the number of elements in a complete net of this type is somewhat more difficult. From figure (4.2^c) and the previous result, stating that it requires $2^{n-1} - 1$ gates to obtain $x_1 \overline{f}_1$, the recurrence



Modified Three Input
Universal Nor-Net

Figure 4.4

relationship

$$g(n) = 2^{n-1}-1 + 2\ g(n-2) + 3$$

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

is obtained. From figure (4.4) $g(2) = 4$, $g(3) = 6$. For n even the solution for $g(n)$ is

$$g(n) = 2^n + 2^{\frac{n}{2}} - 2\ .$$

This is true for $g(2)$, and the induction yields

$$g(n) = 2^{n-1} + 2\ (2^{n-2} + 2^{\frac{n-2}{2}} - 2 + 1)$$

$$= 2^n + 2^{\frac{n}{2}} - 2$$

as required. For n odd

$$g(n) = 2^n - 2$$

as may easily be proved by a process similar to the above. Using this net in place of the first will result in a saving of one element for n odd; ~~and in~~ considerably more elements are needed for n even.

An example will now be done to show the general method of procedure in designing these nets. In practice, a boolean function is defined by a subset of the integer numbers between 0 and 2^n-1, to save writing out the function in the rather lengthy truth table form. The members selected for the subset are the integers, whose corresponding binary form in the truth table, have a value 1. The example function shown on page 71 would have a form

$$f_2 = (0,\ 1,\ 5,\ 6),$$

a much more compact representation indeed.

From a table of random digits an arbitrary function may be selected by taking a string of 2^n digits and filling in a truth table with 0's and

1's according to whether the corresponding decimal digit is even or odd. Using this method the function of five variables

$$f = \begin{pmatrix} 1, 3, 5, 6, 7, 8, 9, 12, 14, 15, \\ 16, 17, 19, 20, 25, 26, 29, 30, 31 \end{pmatrix}$$

was selected. As there are an odd number of variables the constants $\overline{f}(\underline{\mu}_i)$ are used to determine what the final terminations on the net are. These are shown by dotted lines in figure (4.5). The final net, after the most obvious simplifications are made, is shown by the heavy lines. The identities used to reduce the number of elements towards the end include

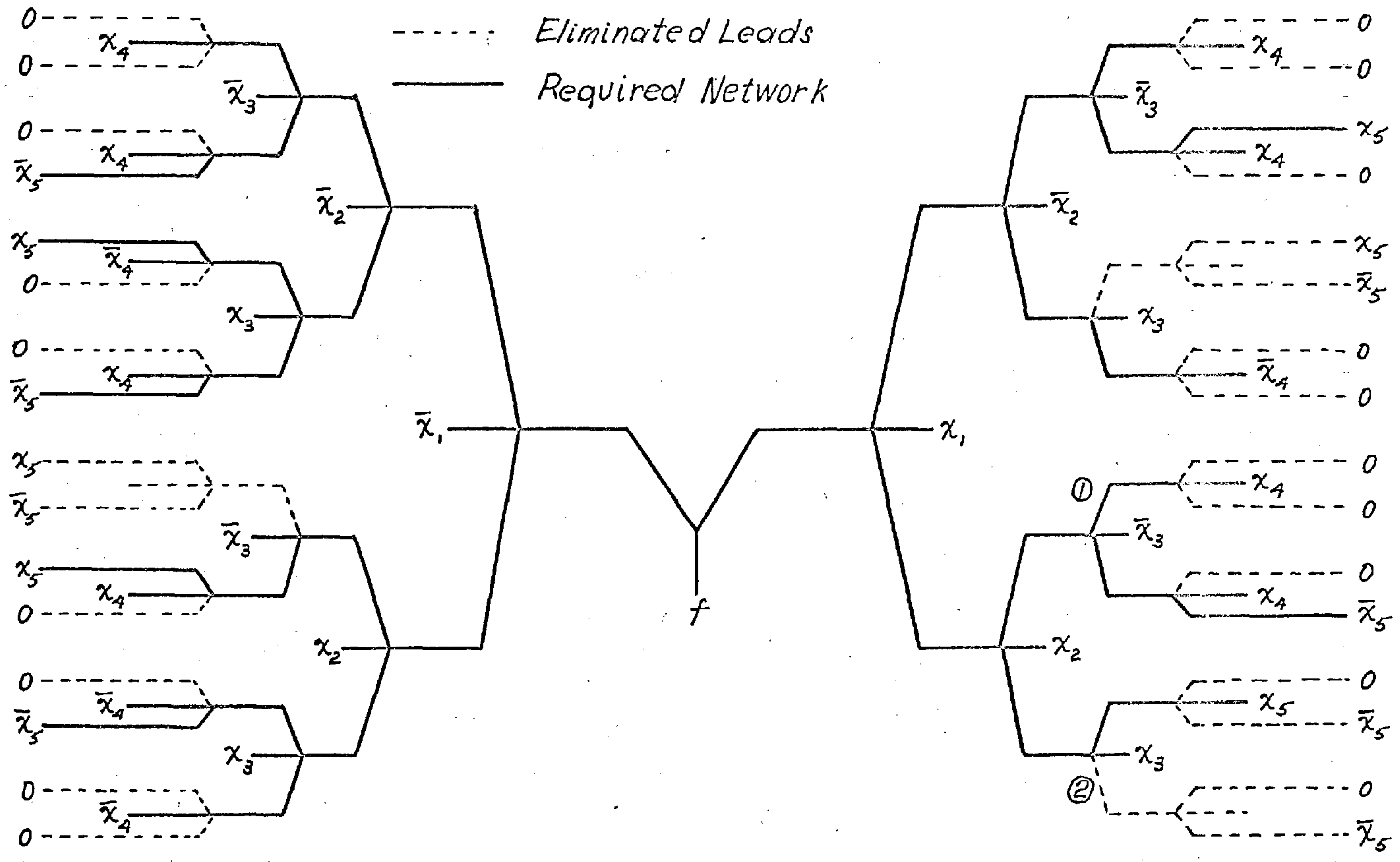
$$\overline{x_3 + x_5\overline{x_4} + x_5x_4} = \overline{x_3 + x_5}$$

and $$\overline{x_4 + x_5 + \overline{x_5}} = 0 .$$

The total number of gates required is 22, nine less than the number in the complete net. Using the second expansion method the number of gates is reduced to 19, illustrating that for odd numbers of variables this second method will normally give a more economical realization.

If the outputs of the gates are allowed to drive more than one input then the number of elements in figure (4.5) can be reduced to 18 by a cursory inspection. This, as will be shown, is not too inefficient, although some way from the optimal realization.

Expansions for four and five inputs, using various configurations, can also be developed in the same manner as above by performing different manipulations on the Shannon expansion formula. They do not usually give a more economical realization in terms of the number of elements



Realization of Example Function
Using a Universal Nor Net

Figure 4.5

used. Their main advantage is that the resulting net is not as 'deep', meaning that at each stage more than one of the variables may be separated. This will result in a faster response. Set against this is the necessity of increasing the design tolerance on elements to ensure their proper operation with the increased number of inputs.

4.5 Minimizing Nor Nets

The net to be considered in the following will be the first three input nor net cited above. This will be designated a 3n-net. It is evident that if the variables had been chosen in a different order, a different net, with perhaps fewer elements, would have resulted when realizing a specified function f. In fact there is even more freedom than this, for, having chosen the first variable, it is possible to rearrange the variables in f_0 and f_1 independently.

If $g(n)$ = total number of variable configurations for a function of n variables, then the recurrence relationship for $g(n)$ is

$$g(n+1) = (n+1) \times g^2(n). \qquad (4.12)$$

This is valid, as the first variable in an $(n+1)$ net can be chosen in n+1 different ways; and having chosen the first, the remaining n variables in f_0 and f_1 may be chosen in $g(n)$ ways independently.

$$g(1) = 1$$

as if functions of 1 variable are considered the choice can only be ordered in one way.

This recurrence relation has a solution

$$g(n) = \prod_{i=0}^{(n-1)} (n-i)^{2^i}.$$

The equation is true for n=1. From (4.12) and using the induction hypothesis

$$g(n+1) = (n+1)\left\{\prod_{i=0}^{n-1}(n-i)^{2^i}\right\}^2$$

$$= (n+1)\prod_{i=0}^{n-1}\left\{(n-i)^{2^i}\right\}^2$$

$$= (n+1)\prod_{i=0}^{n-1}(n-i)^{2^{i+1}}$$

$$= (n+1)^{2^0}\prod_{i=1}^{n}(n+1-i)^{2^i} = \prod_{i=0}^{n}(n+1-i)^{2^i}$$

and the result is proved. This is a very rapidly expanding function of n indeed, (for n=5 it is already greater than 1,500,000) making an exhaustive search through all possible selections out of the question. With the help of a theorem provided below this difficulty may be circumvented.

Divide the variables $\underline{x}$ into two disjoint subsets $\underline{y}$ and $\underline{z}$. Then

$$f(\underline{x}) = f(\underline{y}, \underline{z}).$$

In the network discussed previously, if, instead of assigning the truth values at the far right and then simplifying, we had started from the left and calculated the functions f_0, f_1 : f_{00}, f_{01}, f_{10}, f_{11} : ..... etc., a stage k would be reached when $f(\underline{\alpha}, \underline{z}) = a$, where $\underline{\alpha}$ is a k dimensional constant and $\underline{z}$ is an n-k dimensional vector. 'a' is a constant either 0 or 1. At that stage the expansion along that particular branch would stop.

<u>Definition</u>: a 'terminating branch' is a branch on the 3n-net such that $f(\underline{\alpha}, \underline{z}) = a$. 'a' is called the terminating constant and the

vector $y_1 \oplus \bar{\alpha}_1, y_2 \oplus \bar{\alpha}_2, \ldots\ldots y_k \oplus \bar{\alpha}_k$, the terminating resolution (the y_i's are some subset of the x_i's as above).

For example, going back to figure (4.6) the branches labelled (1) and (2) are terminating branches. For branch (1) $f(0, 0, 1, 1, x_5) = 1$; for (2) $f(0, 0, 0, x_4, 0) = 0$. The resolution vectors for the two are $(\bar{x}_1, \bar{x}_2, x_3, x_4)$ and $(\bar{x}_1, \bar{x}_2, \bar{x}_3, x_5)$ respectively. The formula $f(0, 0, 1, 1, x_5) = 1$ implies that the pair (6, 7) must be members of f; the second expression implies that (0, 2) must be members of $\bar{f}$.

Theorem 4.1: The number of elements in a 3n-net realizing a function f, is equal to one less than the number of terminating branches.

Proof: For functions of one variable the expansion only goes back one level. There are two terminating branches and one element and the theorem is valid. Suppose the result is true for functions of k variables.

Increasing the number of variables to k+1 one of three cases must arise. At the first level (i.e., on separating the first variable) there are either 0, 1 or 2 terminating branches. If there are 2 terminating branches then there is only one element. If there is 1 terminating branch then one of f_0 or f_1 is not a constant. Let it be f_1 and let it have r_1 terminating branches. As it is a function of k variables it must contain r_1-1 elements by the induction. In all there are $(r_1-1)+1$ elements and r_1+1 terminating branches and the theorem is valid. If there are no terminating branches at the first level then, let the number of terminating branches for f_0 and f_1 be r_0 and r_1, with r_0-1 and r_1-1 elements respectively. The total number of elements in the net is $(r_0-1) + (r_1-1)+1 = r_0+r_1-1$ and the number of terminating branches is r_0+r_1. Q.E.D.

The problem of minimizing the number of elements in the net is therefore equivalent to the problem of minimizing the number of terminating branches.

One of the most concise ways of representing a boolean function is the 'cubical' expansion as described by Miller[22]. This again represents a function by a subset of the numbers 0 to 2^n-1 but this time in binary form, in effect writing out the binary numbers from the truth table that have a truth value of one. Each number has a conjunction associated with it in the conjunctive normal form, i.e., the number $\underline{\alpha}$ has associated with it $(x_1 \oplus \bar{\alpha}_1)(x_2 \oplus \bar{\alpha}_2) \ldots\ldots (x_n \oplus \bar{\alpha}_n)$. Suppose two numbers $\underline{\alpha}$ and $\underline{\beta}$ differ in only one place say in the i^{th}. Then $\alpha_j = \beta_j$, $j \neq i$; $\alpha_i = \bar{\beta}_i$. Taking the 'or' sum of the two corresponding conjunctions

$$(x_1 \oplus \bar{\alpha}_1) \ldots\ldots (x_{i-1} \oplus \bar{\alpha}_{i-1})(x_{i+1} \oplus \bar{\alpha}_{i+1}) \ldots\ldots (x_n \oplus \bar{\alpha}_n)$$

gives a conjunction of n-1 variables. This is called a 'one-cube' and is represented in the cubical expansion by

$$\alpha_1\alpha_2 \ldots\ldots \alpha_{i-1}\mu\alpha_{i+1} \ldots\ldots \alpha_n.$$

Similarly two, and higher cubes, may be defined. For example the two conjunctions

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} = (1\ \mu\ 0\ 1\ 1\ 0)$$

and the function of three variables

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \mu & \mu & 1 \\ 1 & 1 & \mu \end{pmatrix}$$

The function of three variables may then be put in the form $x_3 + x_1x_2$ rather than the much longer expression that results from using the left-hand side directly. The problem of obtaining a conjunctive expansion having a minimal number of terms, then reduces to the problem of finding a cubical cover for the function that has a minimum number of cubes. This has already been solved by the method of Quine and McCluskey[22,28] and is described in Appendix II.

There is one other interesting property of these cubes that is of importance to us. For a k- cube a permutation of the variable will put it in the form

$$(a_1a_2 \;\ldots\ldots\; a_{n-k}\;\mu\mu \;\ldots\ldots\; \mu)$$

where k,μ's appear after the constants. If this cube is a member of a cubical cover for f, then

$$f(a_1,a_2 \;\ldots\ldots\; a_{n-k}\;y_{n-k+1} \;\ldots\ldots\; y_n) = 1.$$

Therefore, it is a terminating branch for f. The converse, that a terminating branch with value 1 defines a cube covering a portion of f is also true. Using the Quine (Appendix II) minimization method, it is possible to obtain minimum cubical covers for f and $\bar{f}$. The number of terminating branchescannot possibly be less than the total number of cubes contained in these two covers, for, if this was so, then, from the terminating branches, it would be possible to derive a cover for f (or $\bar{f}$) which had fewer cubes than the covers already stated to be minimum. This gives us a lower bound on the number of elements required. Unfortunately this lower bound is not always attainable.

The cubical-cover does give a clue as to which order to choose the variables. The terminating branches are divided into two classes, the zero terminations and the one terminations. The set of all one terminations describe a cover for f and the set of all zero termination a cover for $\bar{f}$. Looking at figure (4.5) again, and taking into account the inversion of $f(\underline{\alpha},\underline{y})$ at the odd levels, these two covers, as found from the terminating branches, are

f	$\bar{f}$
0 0 0 μ 1	0 0 0 μ 0
0 0 1 0 1	0 0 1 0 0
0 0 1 1 μ	0 1 0 1 μ
0 1 0 0 μ	0 1 1 0 1
0 1 1 0 0	1 0 0 1 0
0 1 1 1 μ	1 0 1 0 1
1 0 0 0 μ	1 0 1 1 μ
1 0 0 1 1	1 1 0 0 0
1 0 1 0 0	1 1 0 1 1
1 1 0 0 1	1 1 1 0 0
1 1 0 1 0	
1 1 1 0 1	
1 1 1 1 μ	

That these are covers can be checked by going back to the definition of the function. There are 23 terminating branches and hence 22 elements as required by the theorem. The first column of figure (4.6) is the minimum cover for f and $\bar{f}$ as found by the Quine method. The cover for $\bar{f}$ has an * after all its members. There are fifteen cubes in all, making this the minimum possible number of end points.

When a variable x_i is chosen and the function expanded about it, the cubes covering the function will be split into two classes corresponding to f_0 and f_1. For example the three cubes shown below when expanded about the third variable will split into four cubes covering f_0 and f_1.

$$\begin{array}{ccccc} \mu & \mu & 0 & 0 & 1 \\ 0 & 0 & \mu & \mu & 1 \\ 1 & 1 & 1 & \mu & 1 \end{array} \qquad \begin{array}{c} \bar{x}_3 \; \begin{array}{cccc} \mu & \mu & 0 & 1 \\ 0 & 0 & \mu & 1 \end{array} \\ \\ x_3 \; \begin{array}{cccc} 0 & 0 & \mu & 1 \\ 1 & 1 & \mu & 1 \end{array} \end{array}$$

the cube that has a μ where the variable occurs may be expressed as

$$0\ 0\ \mu\ \mu\ 1 \;=\; \begin{array}{ccccc} 0 & 0 & 0 & \mu & 1 \\ 0 & 0 & 1 & \mu & 1 \end{array}$$

and splits into the two cubes shown.

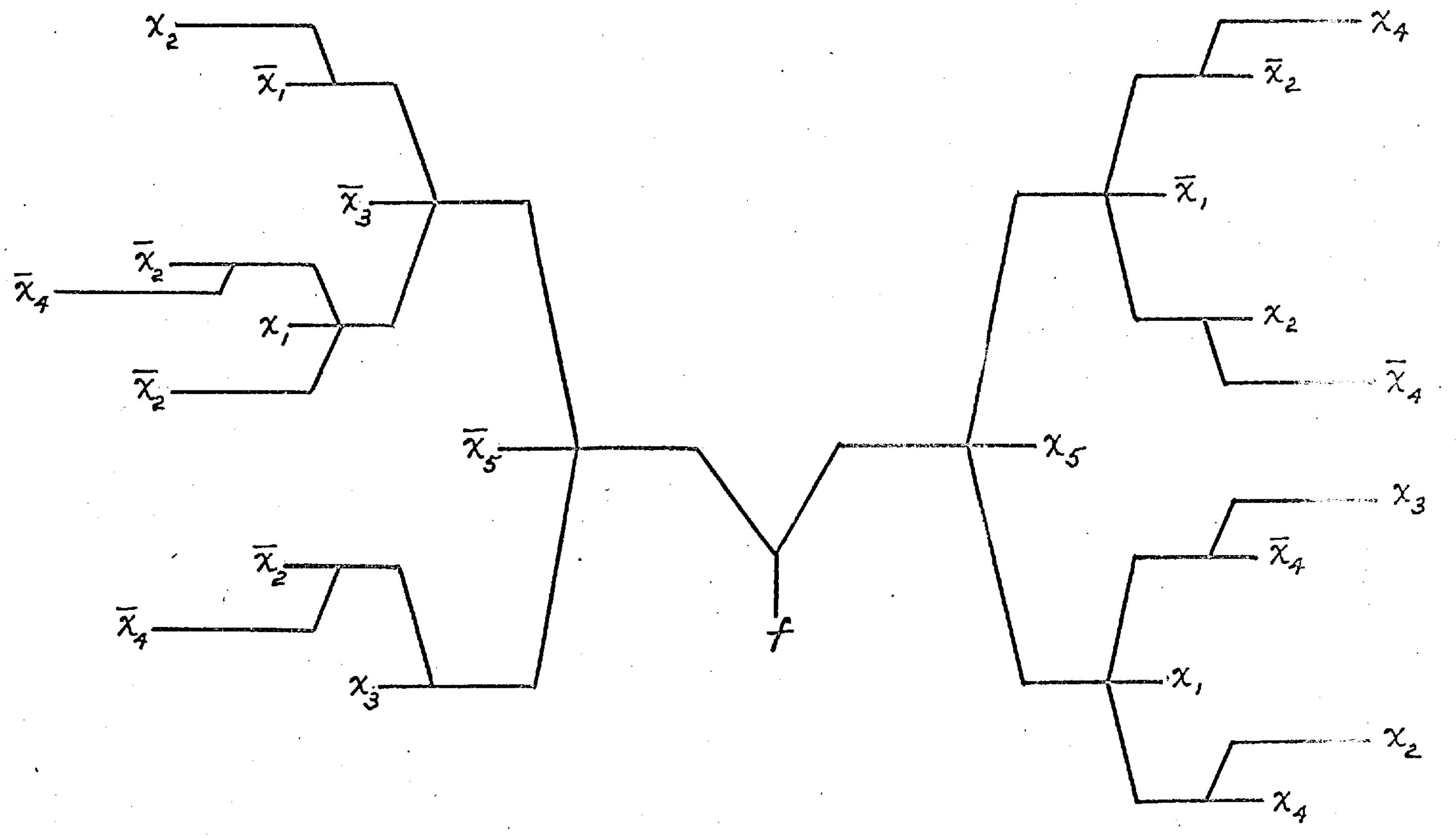
To minimize the number of terminating branches on f_0 and f_1 the variable, that has the fewest occurrences of μ for it, should be chosen first. This ensures that after the splitting procedure the total number of cubes in f_0 and f_1 is the least possible. The same criterion can be used to expand f_0 and f_1 and the process continued until terminating branches are reached. Figure (4.6) illustrates such a decomposition. From the chart, the 16 terminating branches are evident. Therefore, only 15 elements will be required to realize the function, a considerable saving from the first expansion method suggested. Note further that when a cube is split, one half of it may be absorbed by other members in the same class as the cube 00μ is absorbed by $\mu 0\mu$ in the example. Both halves cannot be absorbed as this would imply that the cover was not minimal to begin with. A continuous check can be kept on the process by observing that, after each decomposition, the total number of reduced cubes in $f(\underline{\alpha},\underline{x})$ and $\bar{f}(\underline{\alpha},\underline{x})$ must cover the entire space defined by $\underline{x}$.

From the decomposition chart it is a simple matter to obtain the net. The main point to be careful of is that a indicates a 0 branch for even levels and a 1 branch for odd levels. Figure (4.7) gives the

$x_1x_2x_3x_4x_5$		$x_1x_2x_3x_4$		0 0 μ	x_1	1 μ	x_2	μ		
				0 μ 1		0 μ *	$\bar{x}_2$	μ *		
		0 0 μ μ	x_3	1 1 μ						
		0 μ 1 1		1 0 μ *		0 μ	x_2	1	x_4	
0 0 μ μ 1		μ 0 0 μ		0 1 0 *	$\bar{x}_1$	μ 1		0 *	$\bar{x}_4$	*
0 μ 1 1 μ	x_5	μ μ 0 0				1 0 *				
μ 0 0 μ 1		1 1 1 μ					$\bar{x}_2$	μ		
μ μ 0 0 1		1 0 1 μ *								
1 1 μ 1 0		μ 1 0 1 *		0 0 μ	x_2	μ 0	x_4	μ *		
0 1 μ 0 0		0 1 1 0 *	$\bar{x}_3$	μ 0 μ		μ 1 *	$\bar{x}_4$	μ		
1 0 μ 0 0				μ μ 0						
1 1 1 μ 1				μ 1 1 *	$\bar{x}_2$	μ μ				
0 μ 0 1 0 *										
0 0 μ 0 0 *				1 μ 1	x_2	μ 1	x_4	μ		
1 0 1 μ 1 *		0 μ 1 1	x_1	0 μ 0		μ 0 *	$\bar{x}_4$	μ *		
1 1 μ 0 0 *		1 1 μ 1		1 μ 0 *						
μ 1 0 1 1 *		0 1 μ 0		0 μ 1 *	$\bar{x}_2$	μ 0	x_4	μ *		
1 0 μ 1 0 *	$\bar{x}_5$	1 0 μ 0				μ 1 *	$\bar{x}_4$	μ		
0 1 1 0 1 *		0 μ 0 1 *								
		0 0 μ 0 *		μ 1 1	x_4	μ 1	x_3	μ		
		1 1 μ 0 *	$\bar{x}_1$	1 μ 0		μ 0 *	$\bar{x}_3$	μ *		
		1 0 μ 1 *		μ 0 1 *						
				0 μ 0 *	$\bar{x}_4$	1 μ	x_2	μ		
						0 μ *	$\bar{x}_2$	μ *		

Decomposition Chart for
Example Function

FIGURE 4.6



Minimal Network as Found From Decomposition Chart

Figure 4.7

new network realizing the function. This net has the added advantage that the loading on the variables is more evenly distributed.

To make sure that this method does result in significant reductions in the number of elements a second example function, again chosen randomly, was selected

$$\begin{pmatrix} 1, 2, 3, 8, 9, 10, 11, 13, 15, 17, \\ 18, 19, 20, 21, 22, 23, 25, 29, 31 \end{pmatrix}$$

This has a minimum of 10 cubes to cover it and, on decomposition by the above algorithm, results in 11 terminating branches giving a total of 10 elements required to realize the function. It is doubtful whether any other method of synthesis would yield a more economical result.

4.6 Multiple Output Functions

In many cases our interest is not confined to a single boolean function but rather to a set of functions, or a multiple output function. If a parallel realization of the function is required then it is necessary to expand each function separately and use different nets for realizing them. There is usually some overlapping of the different nets and some saving results by taking advantage of this.

If a serial realization of the set of functions is allowable, then considerable saving can be effected. Introduce

$$s = \left\lceil \log_2 r \right\rceil$$

new variables, where r is the numbers of functions in the set (N.B. $\lceil x \rceil$ is the smallest integer greater than or equal to x). These s variables may be the output of a counter, shift register, or any other dynamic structure that gives r consequetive different combinations of the new variables $y_1 \ldots\ldots y_s$. Ordering the functions as they are wished to appear, and indexing them by the successive outputs of the y_i, a single function of s+n variables is obtained. This may be minimized by the method described above.

As an example consider the function defined on eight elements

$$(0,1,2,3,4,5,6,7) \longrightarrow (5,0,1,6,7,3,0,6)$$

which is equivalent to the truth table

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	1	1	0

Introducing two new variables y_1 and y_2 and assuming that they can be produced in the sequence (00, 01, 10) (by a binary counter for instance), 00 can be associated with f_1, 01 with f_2 etc. A function of five variables

$$\left\{\begin{matrix}(0,\ 3,\ 4,\ 7),\ (11,\ 12,\ 13,\ 15), \\ (16,\ 18,\ 20,\ 21),\ (\underline{\delta})\end{matrix}\right\}$$

is then defined, where $\underline{\delta}$ indicates that for the combination 11 any arbitrary subset of (24,25, ... 31) may be inserted.

This degree of freedom for the last eight elements can be taken advantage of by including all of them while calculating the prime-implicants (Appendix II, figure A.1.2) then use only the actual member elements when calculating the necessary cubes. This results in a minimum number of cubes to cover the defined elements; the remaining elements will automatically have been selected in the best fashion.

For the example above the best cover was found to be the ten cubes below

```
0 x x 1 1        0 x x 1 0  *
1 x 0 x 0        1 x 0 x 1  *
1 x 1 0 x        1 x 1 1 x  *
x 1 1 0 x        0 0 x 0 1  *
x 0 x 0 0        0 1 0 0 x  *
```

By selecting the order of the variables in the manner described in the previous section, only ten terminating branches were obtained, giving us nine elements in our realizing net. If the three functions are realized separately eleven elements is the minimum obtained, showing that even in this rather trivial case there is a substantial saving. As the number of functions and the number of variables increases you would expect this saving to be more noticeable.

CHAPTER V

Function Generation Using Digital Circuits

5.1 General Digital Function Generator

As mentioned previously (page 57) a digital function is a mapping from a finite set of points X to a finite set of points Y. If the number of points in X is P and the number of points in Y is Q, then X may be coded into $p = \lceil \log_2 P \rceil$ binary bits and Y into $q = \lceil \log_2 Q \rceil$ bits. In terms of these binary variables the function may now be defined by q boolean functions of p variables. $q \leqslant p$, otherwise the space Y would contain more elements than X, indicating that some X would have to map into two elements. Figure (5.4^a) is the circuit diagram for this realization. The single set of input leads into the function blocks f_i really represent all p leads coming from the input variables x_i. A similar notation is employed for the outputs and inputs of the coding blocks.

It is assumed that the domain of all functions under consideration is the integer set 0,1, ... P-1 and the range 0,1, ... Q-1. With this assumption, the natural coding of the input and output into their binary number representation may be used, eliminating the need for separate coding blocks. An example of realizing such a function by a sequential net has already been done (page 98). For different domains and ranges, input and output coding blocks must be inserted, compounding the problem considerably. For all practical cases such 'discontinuous' (in the sense that if integers 'a' and 'b' are members of the space then not all integers between 'a' and 'b' are) spaces do not arise.

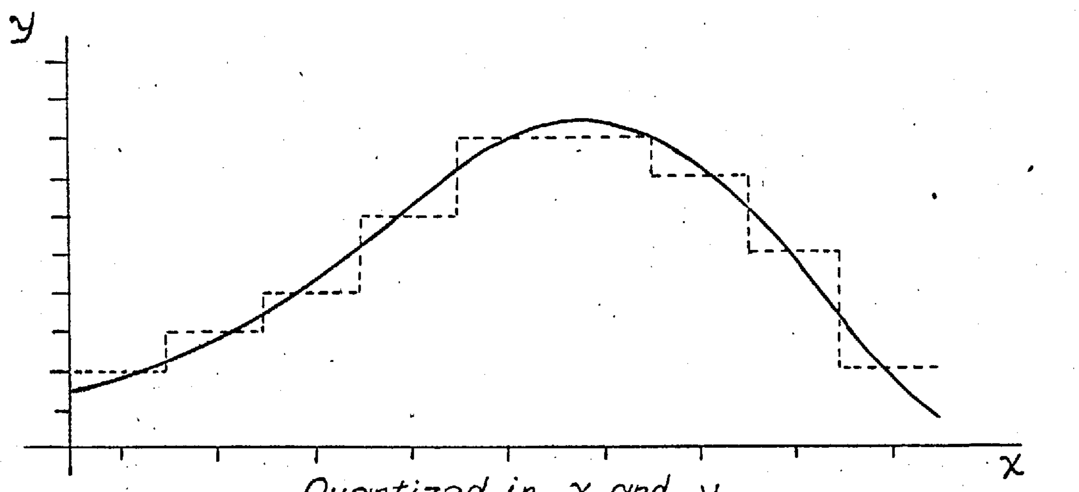
Such a design procedure may be used to obtain any digital function, representing an arbitrary mapping from X to Y. For most purposes this is far too general as normally not all functions from X to Y are required. This leads on to a discussion of exactly what kind of functions are of use and what simplifications this produces in the network above.

5.2 Extension of the 'Continuous' Concept

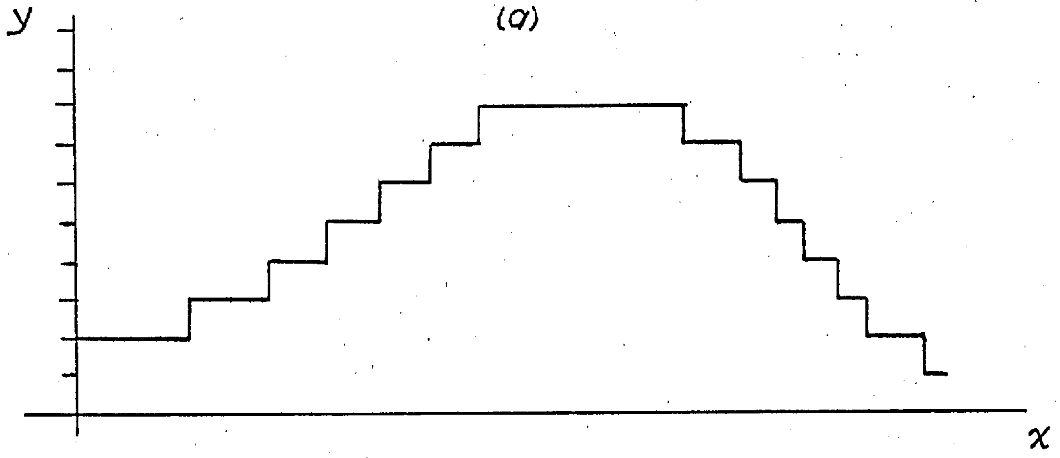
For control systems applications the kind of functions encountered are continuous, or at most, piecewise continuous; anything more abstract can only be of theoretical interest. The problem is to approximate this function by a digital function, implying quantization of both the ordinate and abscissa.

Figure (5.1) shows the effect of quantization on the two axes and (5.2) the errors incurred. y-quantization ensures that the error of the approximating function never exceeds one-half a level; x-quantization that the error at the sample points for x is zero. Very large errors may occur for this second case as shown in figure (5.2^b). When both co-ordinates are quantized all that can be said about the error is, that at the sample points for x, the error must be less than or equal to half a y level. The goodness of the approximation over the remainder of any single interval is dependent upon the value of the derivative. Note that the operations of x-quantization and y-quantization are commutative, meaning that the same final function is obtained regardless of which operation is applied first.

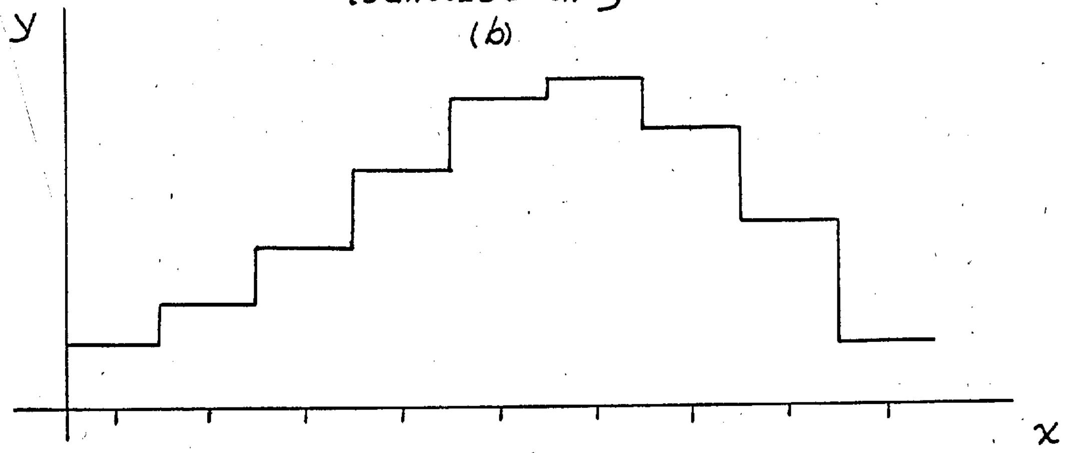
Let $a \leqslant x \leqslant b$ be the domain of definition for the functions.



Quantized in x and y
(a)



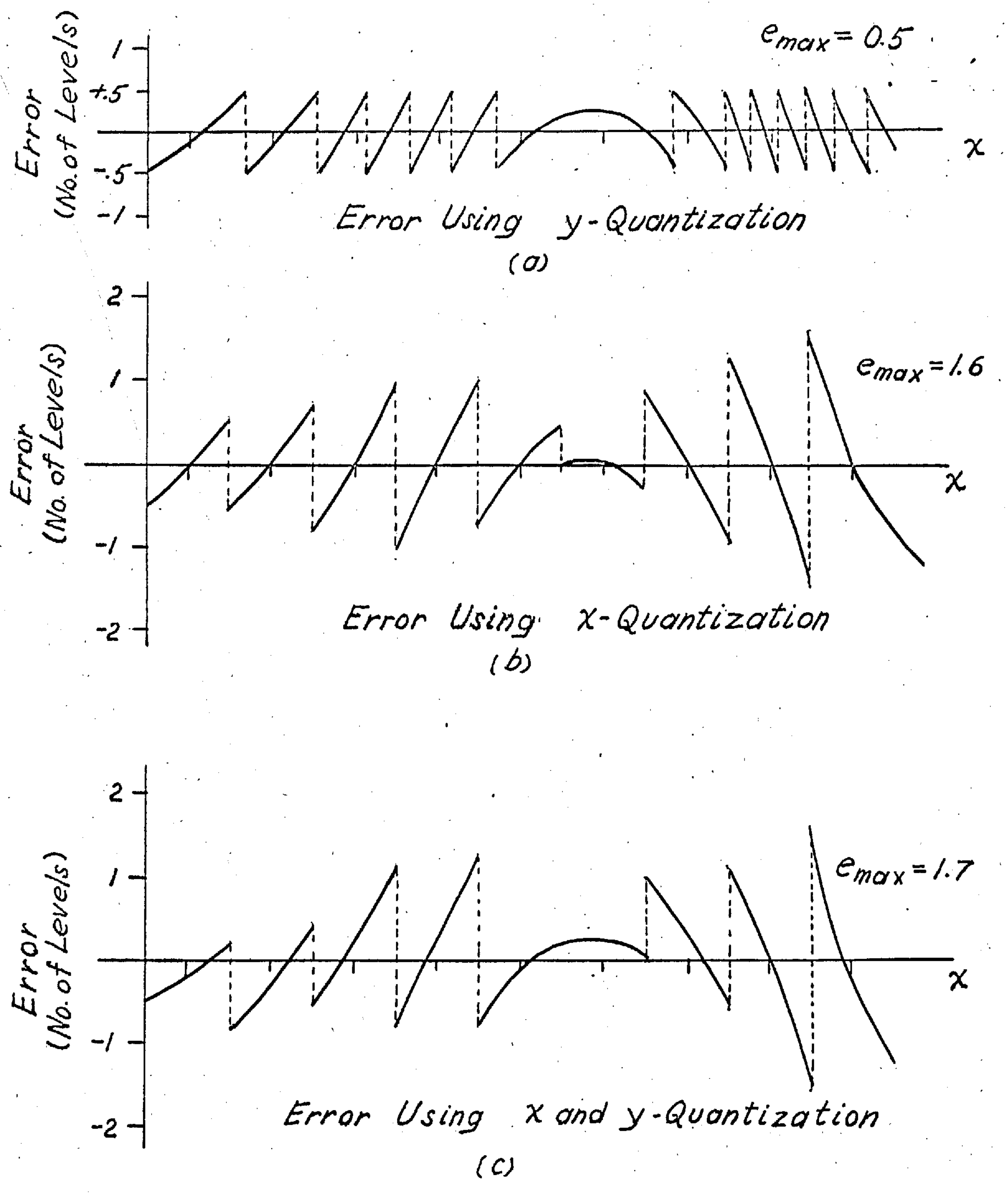
Quantized in y
(b)



Quantized in x
(c)

Methods of Quantizing a Continuous Function

Figure 5.1



Errors Incurred in Quantizing Continuous Functions

Figure 5.2

Define the three quantities below

$$A = \max y - \min y$$

$$B = b-a$$

$$C = \max_{a \leqslant x \leqslant b} \left|\frac{dy}{dx}\right| .$$

If y is quantized into r levels and x into s levels then one y level would be A/r and one x level B/s. The maximum error at the mid-point for x is then A/2r. If the maximum derivative occurs in this interval, then the change in going to the end points of the interval cannot exceed CB/2s, as drawing a simple diagram immediately shows. Hence, the following equation for the maximum possible deviation of the quantized function from the true function in any given interval results.

$$e_{max} \leqslant .5(A/r + BC/s). \tag{5.1}$$

Limiting the total number of quantization levels in x and y, i.e., r + s = k, to put an upper bound on the complexity, (5.1) becomes

$$e_{max} \leqslant .5 \left(\frac{A}{r} + \frac{BC}{k-r}\right). \tag{5.2}$$

Both r and s are greater than zero indicating that r in the above equation must satisfy the bounds

$$0 < r < k. \tag{5.3}$$

Differentiating the right-hand side of (5.2) and setting it equal to zero

$$\frac{A}{r^2} = \frac{BC}{(k-r)^2}. \tag{5.4}$$

Solving this equation for r and rejecting the solution that does not satisfy (5.3) yields the two values

$$r = k/(1 + \sqrt{\frac{BC}{A}}) \tag{5.5}$$

and $$s = k/(1 + \sqrt{\frac{A}{BC}}). \tag{5.6}$$

Substituting these values back into (5.1) gives the upper bound for the maximum error, using the best possible choice of r and s, as

$$e_{max} \leqslant \frac{(\sqrt{A} + \sqrt{BC})^2}{2k}. \tag{5.7}$$

Equations (5.5) and (5.6) constitute a method for distributing the quantizing capacity (k) between the x and y coordinates in a best fashion.

The distancesbetween adjacent y levels and adjacent x levels are

$$\frac{A}{r} = \sqrt{A}\ (\sqrt{A} + \sqrt{BC})/k \quad = \Delta y$$

$$\frac{B}{s} = \sqrt{B}\ (\sqrt{B} + \sqrt{A/C})/k \quad = \Delta x$$

respectively. In going from one value of x to an adjacent value of x the maximum change in y is $C\Delta x = \sqrt{BC}\ (\sqrt{BC} + \sqrt{A})/k$. Therefore, the maximum number of quantization levels the function may change by is

$$\mu = \frac{C\Delta x}{\Delta y} = \sqrt{\frac{BC}{A}}. \tag{5.8}$$

This constant is dependent only on the nature of the function under consideration and not on the total number of levels available. If the present value of y is known, then the next value of y can be one of ~~, up to 2μ+1 levels,~~ only $2\mu+1$ levels as it can change to any ~~is also known.~~ value from $y+\mu$ to $y-\mu$

<u>Definition</u>: A discrete function f(n) is said to be 'continuous' of order μ if in going from f(n) to f(n+1) the function can change by no more than μ levels.

Such functions as the above may therefore be specified by an initial value and a string of s-1 (2μ+1)-level digits. As an example consider realizing the function $f = \sin x$ on the interval $(0, \frac{\pi}{2})$. Suppose only $k = 20$ quantization levels are allowed. The constants A, B and C are 1, $\frac{\pi}{2}$ and 1 respectively. Therefore

$$\mu = \sqrt{\frac{\pi}{2}} = 1.253 < 2$$

$$r = 20/(1+\mu) = 9.05 \approx 9$$

$$s = k-r = 11.$$

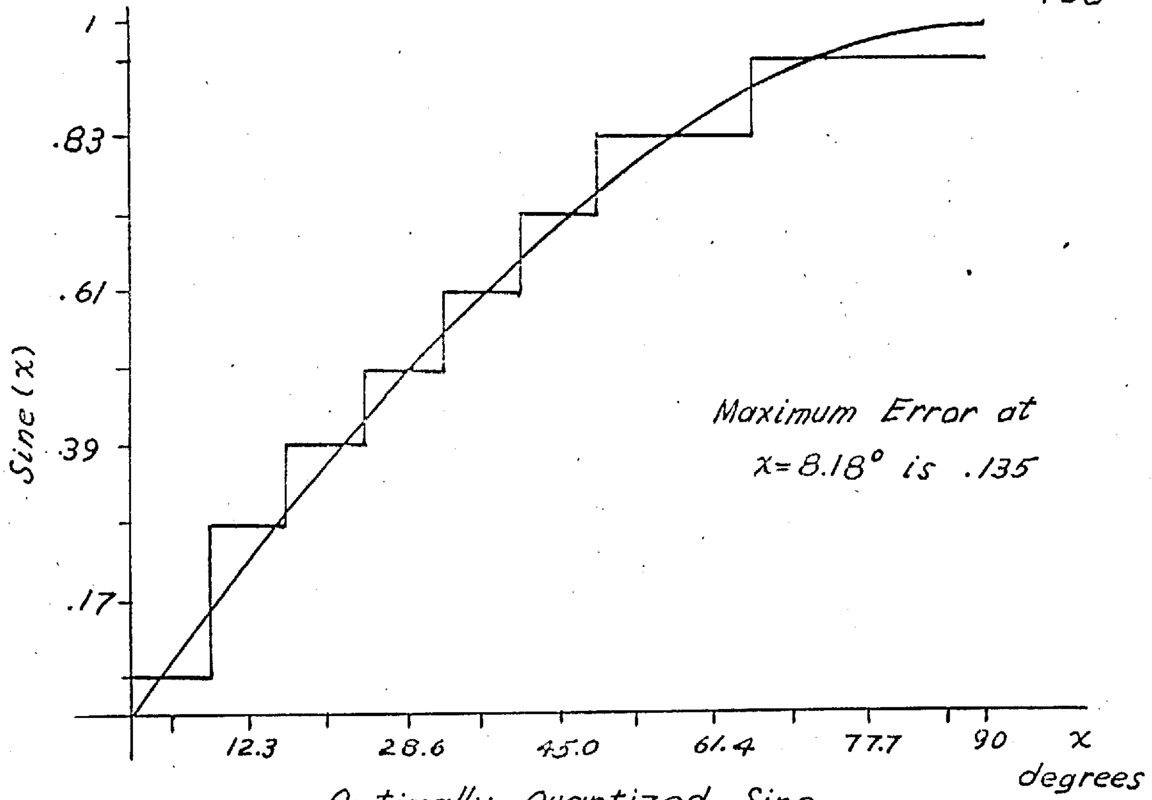
The quantization levels Δx and Δy are .1438 and .111 respectively. The first x level is $x = 0.0714 = 4.09^{\circ}$; the first y level .0556. From a table for sin x the quantized function, as shown in figure (5.3^{a}) is obtained. The maximum deviation of the quantized function is .135, whereas the maximum calculated from equation (5.7) is .127. The difference is accounted for by the fact that it is impossible to have a fractional number of quantization levels; the nearest whole number must be taken. The x levels may be identified with the numbers 0 through 10, and the y levels with 0 through 8. Reverting to the true value only involves a scale change on the two axes. As the function is 'continuous' of order 2, it may be represented by the initial value 0 and the string 2111110100.

Sometimes the number of y levels, r, is previously specified and it is required to find the number of x levels to ensure that over any single interval the function does not vary by more than one level from the quantized function. For this restriction (5.1) becomes

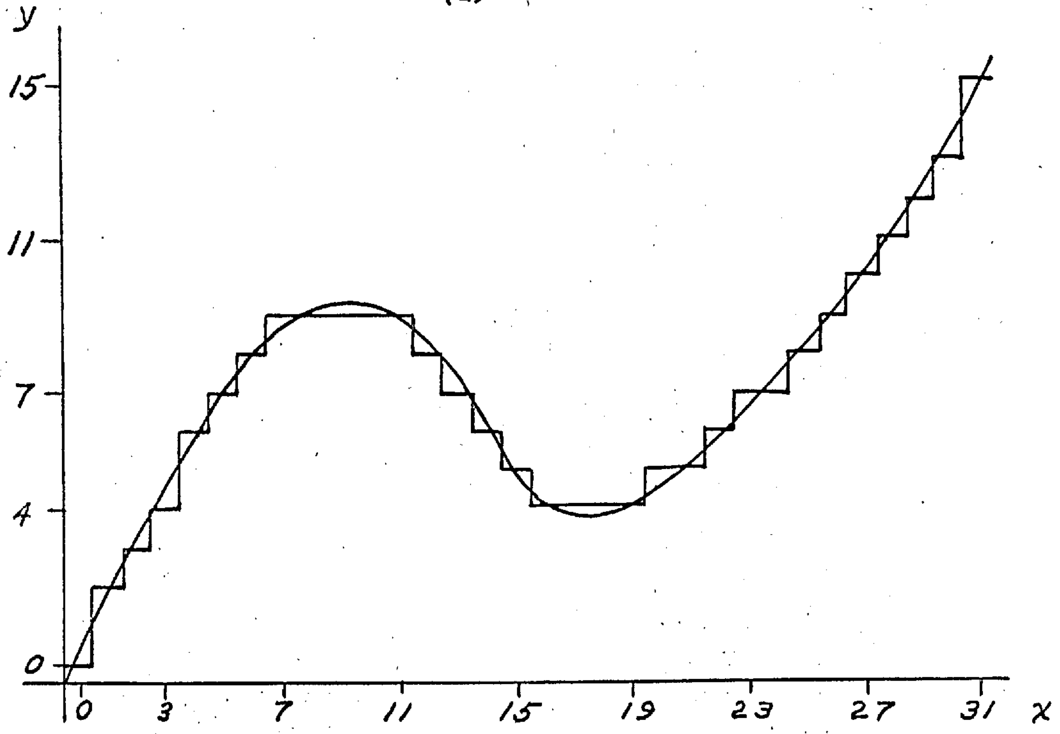
$$\frac{A}{r} = \frac{A}{2r} + \frac{BC}{2s}$$

or

$$\frac{A}{r} = \frac{BC}{s}$$



Optimally Quantized Sine
Wave
(a)



Example of a Digital
Realization
(b)

Figure 5.3

and the number of x points required is $s = (\frac{BC}{A})r$. BC is always greater than A so there are always more x sample points than y levels. The quantization intervals are $\Delta x = B/s$ and $\Delta y = A/r$. In going from one x interval to the next the change in the function cannot exceed $C\Delta x = \frac{BC}{s} = \Delta y$, implying that y can change by at most one interval. If after applying this method for choosing the number of x levels, it is decided that they are excessive and, therefore, they are arbitrarily cut down with the tacit allowance of an error of more than one quantization level, then, evidently it would have been better to choose larger quantization intervals to begin with, and simplify the entire problem. Hence, for all functions, realizations that are 'continuous' of order one may be used. Any function may then be represented by an initial state plus a three level string.

Our interest in digital realizations of the type discussed above will be justified later when adaptive networks are discussed. At the moment it is only necessary to emphasize the difference between the two types of quantization methods mentioned above. The first assumes that there are a fixed number of levels at our disposal to distribute between the two axes; a method for finding this best distribution was discussed. Note that the particular method of establishing the x and y levels is best only in a statistical sense. This means that using the above scheme the error can be guaranteed to be kept inside the bound (5.7) for any function having the parameters A,B and C. For a particular function there may exist a better quantization scheme, although this other scheme will result in larger errors than (5.7) for some other function with the same parameters.

The second method is somewhat more realistic as normally one is given an analogue to digital converter with a certain number of bits ($\log_2 r$) and it is required to find the sampling interval $\frac{B}{s}$ to make the best use of these bits. Again the criterion for establishing the x-quantization holds for all functions with the same parameters, whereas for a particular function it may be possible to use considerably fewer levels and get the same accuracy. The following section contains some practical methods for obtaining functions of the above type.

5.3 Realization of Digital 'Continuous' Functions

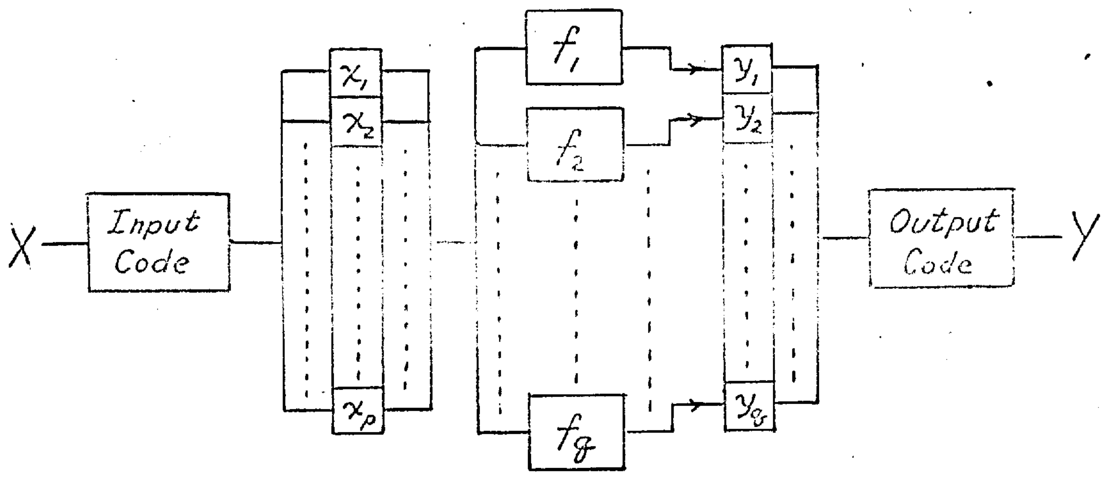
As shown above, if any discretion is shown in choosing the quantization levels then a 'continuous' function of some order μ less than the number of quantization levels results. On page 101 a method for realizing arbitrary digital functions has been discussed. For functions with many quantization levels, using this method results in a large number of boolean functions of many variables. The complexity may be reduced considerably by utilizing the 'continuous' property of the functions.

Having chosen the quantization level, the abscissa 'x' is normally presented to the function generator as a set of p binary bits (See fig 5.4?) from an analog to digital (A-D) converter. It is required to generate from these a set of q output bits, to represent the value of the function. A-D converters fall into two main classes[36,37,38]; the fixed sampling rate and the continuous output converters. Both of these have as their basic element a simple binary counter. The outputs of the counters are added according to the power of two they represent, an analogue network giving a signal proportional to the digital number stored in the register. If the counter is driven by a continuous string of pulses the output of the analogue components would be a sawtooth wave. The first class of A-D converters starts the counter in the zero state, compares the output with the signal to be quantized, and counts up if the counter output is less than the signal. The counter stops as soon as its output is greater than that of the signal; its contents are then a digital representation of the signal correct to one level. The second class has a facility for counting up and counting down. The output is continuously compared with the signal, if the signal is greater the count is increased, if it is less the count is decreased. Ignoring the first stage (which is usually a multivibrator driving the counter continuously) the remaining stages yield the required digital output to as many levels as desired. Corresponding to these two types of converters there are two main kinds of function generator that may be evolved.

Throughout the following the functions are mappings from 2^p elements into 2^q elements, or from p bits to q bits. Figure (5.4^a) shows a function generator that can produce any function of the above type. It requires q functions of p variables. These may be generated sequentially as shown in Chapter III. As mentioned previously, if any thought was applied in choosing the quantization levels then a 'continuous' function results. Figure (5.4^b) shows a method for obtaining 'continuous' functions of order μ. The bits of x are mapped into

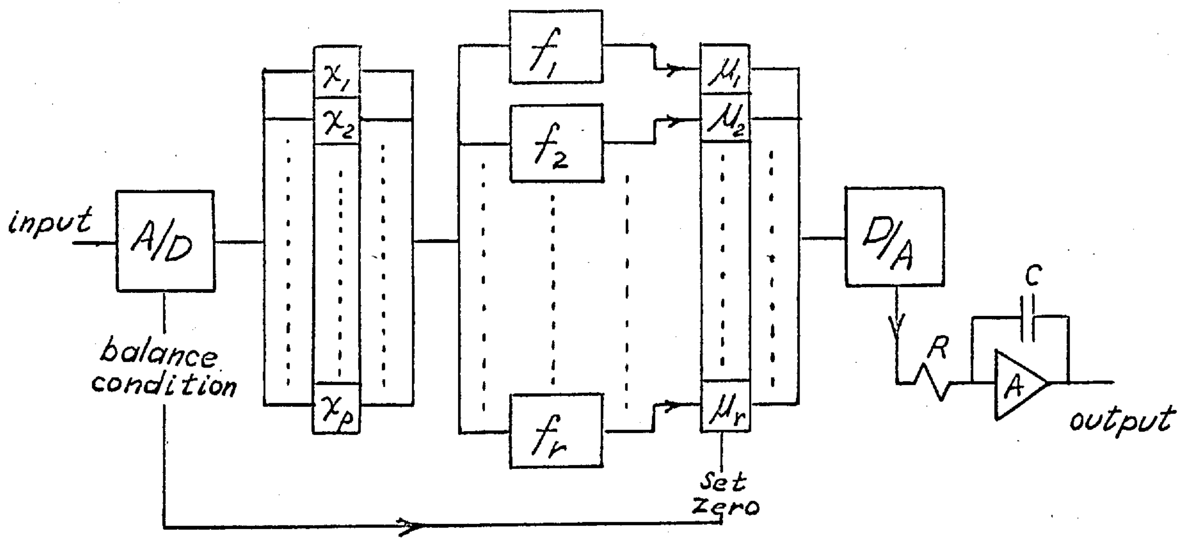
$$r = \left\lceil \log_2 (2\mu + 1) \right\rceil \qquad \text{functions}$$

representing the increment in the function in going from x to x + 1.



General Digital Function Generator

(a)



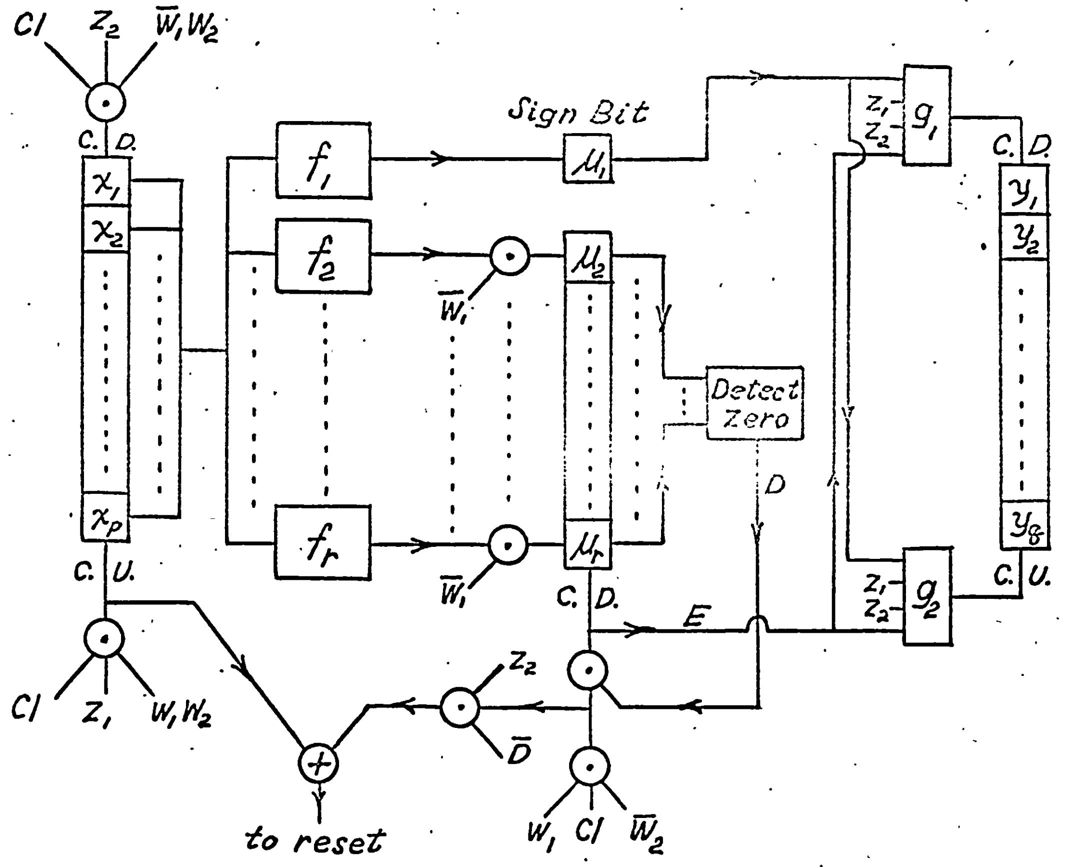
Digital Function Generator for Functions of Order μ

(b)

Figure 5.4

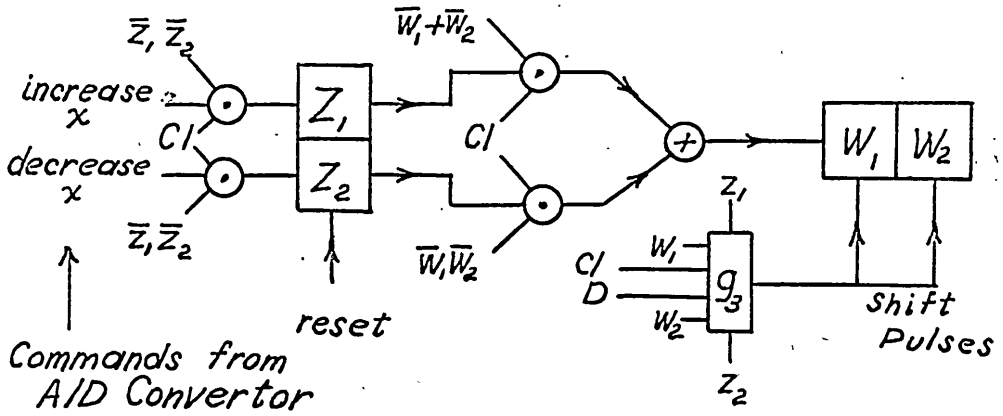
One of the r bits may be used as a sign bit, the remainder representing the value of the change required. r is considerably less than q, producing a saving in the total number of boolean functions. This scheme is suitable only for the first kind of A/D converter, as large errors will result from the integrator on the output if it is required to hold for any length of time. The reset circuitry has been omitted for clarity. With a few additions this circuit could be used in conjunction with an A/D converter of the second kind, provided that the integrator can be replaced with some more stable element.

Figure (5.5^a) shows an all digital method for realizing functions of order μ, (5.5^b) giving the required auxiliary circuitry for generating the timing variables w_1, w_2, z_1, and z_2. Although the apparatus may at first sight look rather complicated, its operation is simple. The input and output registers, x and y, are connected in a reversible binary counter configuration[39,40]; the input C.U. is the command to count up, the input C.D. the command to count down. The whole network is timed by a master clock represented by Cl. Register μ contains the increment in going from x to x+1, f_1 representing the sign bit (1 for positive, 0 for negative) and the remainder containing the absolute value. The addition or subtraction from register y is performed by counting down μ to zero, and simultaneously counting y up or down, depending on the sign, and whether x is increasing or decreasing. This counting configuration is used in preference to a straight adding and subtracting device as the number of bits r is usually 2 or 3 at most, not warrenting the use of a full digital adder.



Digital Function Generator for Functions of order μ

(a)



$\overline{z}_1 \overline{z}_2$
increase x
Cl
decrease x
$\overline{z}_1 \overline{z}_2$
Commands from A/D Convertor
Z_1
Z_2
reset
$\overline{W}_1 + \overline{W}_2$
Cl
$\overline{W}_1 \overline{W}_2$
W_1
W_2
z_1
W_1
Cl
D
W_2
g_3
z_2
Shift Pulses

Timing Circuitry

(b)

Figure 5.5

The first network in figure (5.5^b) is necessary to ensure that the command signals from the A/D converter are isolated from the timing circuits while the system is going through an accumulation phase. For x increasing the contents of μ must be added to y before x is incremented; for x decreasing the input register must first be reduced before μ is subtracted from y. This change in timing sequence is produced by the two memory elements w_1 and w_2 connected as a feedback shift register. The output D is 1 when μ is non-zero, 0 when μ is zero. The functions g_1 and g_2 ensure that the output register is counted in the right direction dependent upon which way x is changing ($z_1 = 1$ for x increasing, $z_2 = 1$ for x decreasing) and the sign of μ. These functions are

$$g_1 = (z_1 \ \text{SIGN} + z_2 \ \overline{\text{SIGN}}) \ E$$

$$g_2 = (z_1 \ \overline{\text{SIGN}} + z_2 \ \text{SIGN}) \ E$$

where the enabling gate E is driven by $D w_1 \overline{w}_2 \, Cl$. Thus, only when the shift register is in the state 10 and the contents of μ non-zero, does this transfer take place. The function g_3 driving the shift register is

$$g_3 = ((z_1+z_2)\ (\overline{w}_1+w_2) + \overline{D}\ w_1 \overline{w}_2)\ Cl.$$

For $z_1=z_2=0$ and no command from the comparator of the A/D converter to increase or decrease x, the network is in equilibrium. w_1 equals 0 and w_2 equals 1, ensuring that no shift pulses are getting through. Note that for $z_1=1$ the shift register passes through the state sequence $01 \rightarrow 10 \rightarrow 11 \rightarrow 01$; for $z_2=1$ the state sequence is $01 \rightarrow 00 \rightarrow 10 \rightarrow 01$ producing the different sequences required for the two modes of operation. The timing sequence for the command to increase x is shown below, each clock interval being indicated by a separate step. In the equilibrium state $w_1=0$ and the outputs of the f_i are connected to the register μ.

1. The command to increase x arrives; z_1 becomes 1 and the gates to z_1 and z_2 are closed.
2. A shift pulse changes w_1 to 1 and w_2 to 0, and the outputs of the networks f_i are disengaged from the register μ.
3. If $\mu=0$ then D=0 and another shift pulse is applied to the register taking it to the state 11. If $\mu\neq0$ then the clock pulse can pass through, counting down μ and accumulating in y. No shift pulses are being received by the register w.
4. Same as above this mode continuing until D=0, implying that the increment μ has been added to the contents of y.

.
.
.
.
.

$\mu+3$. D=0 and the shift register moves to state 11.

$\mu+4$. The x register is counted up one as required, and the shift register set to 01 as well as z_1 and z_2 being reset to zero in preparation for the next sequence of operations.

A similar sequence structure results if a command to decrease x is received although in this case the value for x is counted down before the increment μ is added into the output. The sequence register w first changes to 00 allowing the new value of μ to be read in before the counting procedure starts.

With this scheme there is a danger of getting completely lost if a clock pulse is missed, or some noise gets into the system. This can be avoided by taking the few most significant digits of x and, when the remainder of the digits are 0 (i.e., specifying a subset of x with a very long sampling period), calculate the exact value of y by functions of these fewer variables. The rest of the network can be short circuited hence fixing a set of y values in the output. The functions f_i are usually simplified also as there are now some x whose μ can be chosen arbitrarily; namely those x preceding the ones that have their complete functional value specified. This hybrid realization probably gives the most economical apparatus with the required reliability.

5.4 Example Realization

It is required to realize the function shown in figure (5.3^b) by a digital function generator. This may represent a tunnel diode characteristic necessary for some simulation work. 50 levels at most are allowed, resulting in 32 horizontal intervals and 16 vertical intervals, taking the values obtained from equations (5.5) and (5.6) to the nearest power of two. The resultant quantized function is shown superimposed on the original function. As it is of order 2, 31 five level digits are required to represent it. In fact only four of these digits occur (- 1,0,1,2) so that, with a few minor modifications, μ could be represented by two stages. However, it is assumed that three stages are required, to illustrate the simplification that results.

f_1 is the sign bit,equal 1 if the sign is positive, arbitrary where μ=0

$$f_1 = (0,1,2,3,4,5,6,19,21,22,24,25,26,27,28,29,30)$$

f_1 arbitrary on (7,8,9,10,16,17,18,20,23,31).

A minimal cover for f_1 and $\overline{f_1}$ employing the same notation as in Chapter IV, is found as

```
1 x x x x
x 0 x x x
0 1 x x x *
```

This requires only two nor elements in its realization. f_2 represents the most significant digit in the magnitude of μ and is covered by the three elements

$$f_2 = (0,3,30) \quad \text{arbitrary on } 31.$$

This requires eight three input nor gates. As $\mu=11$ can never occur, when f_2 is true, f_1 must be false. Thus by a little extra logical feedback from f_2 of the form

$$f_3 = \overline{f_2}\ \overline{f'} = \overline{f_2 + f'} \tag{5.9}$$

some simplification may be effected. $\overline{f'}$ is f_3 plus an arbitrary subset of $\overline{f_2}$, or, equivalently, f' is $\overline{f_3}$ minus an arbitrary subset of f_2. f' may be realized by a simpler net than the one that would produce f_3 directly. Using this fact, minimal covers for f' and $\overline{f'}$ are found as below

f'	$\overline{f'}$
0 1 0 0 x	x 1 1 x x *
0 1 0 x 0	1 1 x x x *
1 0 0 0 x	x x 0 1 1 *
1 0 0 x 0	x x 1 0 1 *
1 0 x 0 0	x x 1 1 0 *
x 0 1 1 1	0 0 x x 0 *
	0 0 0 x x *

This needs thirteen elements plus the one extra element required by (5.9). The entire structure is composed of only 24 nor gates, a considerable saving over the result obtained if the transformation is mapped directly. By varying the functions f_i it is possible to realize any function of order 3 using this same network.

As shown previously if accuracy to one quantization level is required then a function of order 1 results and only two boolean networks f_1 and f_2 are necessary. Again f_1 is a simple function as it has a large number of points where it may be chosen arbitrarily corresponding to where $\mu=0$. In this case the timing circuits of figure (5.5) may be simplified as it is known that at most, one count pulse is required at each stage. Furthermore, if the generator is to be used in conjunction with a type one A/D converter, no timing circuitry is required at all as the value may be fed directly into the y accumulator, conditional on the command for x to increase. It is also possible to use three level logic devices[41] although the mathematical difficulties[42,43] seem to increase more than proportionately.

CHAPTER VI

On-Line Adaptation

In all optimal control problems it is first necessary to decide exactly what the object of the design is, i.e., to specify a criterion (cost) function for the system. If the capability of adapting some of the system parameters is admitted then this cost function becomes a function of the adaptable parameters. It is then theoretically possible to find a minimum for the cost by finding where all the derivatives with respect to the parameters are zero, and ensuring that the cost function is concave at this point. For most adaptive systems some hill-climbing method, following the gradient of the function, of adjusting the parameters is proposed to ensure that optimal behaviour is maintained when the other system constants are changing.

If the system has a zero memory function, defining a switching surface for example, in its make-up,and this has to be chosen in an optimal fashion, then the problem is complicated by at least an order of magnitude. Now the criterion function is a function of a function and we find ourselves faced with hill-climbing in Hilbert space. This space can be approximated arbitrarily closely by sets of orthogonal functions, like those mentioned previously, producing an infinite set of co-ordinates to adjust. The hill-climbing problem can be tackled through the classical calculus of variations[44,45,47], or dynamic programming[46,47] both of which lead to enormous computational difficulties to achieve any analytic results. The answer seems to lie in having a learning system

[48,49,50] which adjusts itself empirically to achieve a certain goal[48,50], another mode of saying minimizes a criterion function.

Adaptation is only of use if the system is carried through a number of cycles of operation. Only then is it possible to evaluate the system performance with some known set of the adaptive parameters, following which any changes necessary may be made. Further, the change in the system parameters must be slow with respect to the operation of the performance evaluator, otherwise there is no clear indication of what the cost is with a given set of parameters, and no idea concerning which way to go for improvement can be infered. Therefore, in the following sections, the basic assumptions are that the parameters are slowly varying, and that the system under consideration is cyclic in the sense that it must perform the same kind of operations repetitively.

6.1 Adaptive Functions

When the performance evaluator comes to the conclusion that the particular function used was not the best, the form of command to the function generator is to increase or decrease the function over a specified range, much in the way that functions are varied to find an optimal solution using the calculus of variations. For sophisticated evaluation procedures some idea of the magnitude of this variation may also be given. The input command to the function generator is therefore a subrange specification $a \leqslant x \leqslant b$ and a magnitude of variation δ.

The diode function generator, as mentioned in Chapter III, is probably the most practical method for realizing fixed functions. For adaptation, one is faced with a host of potentiometers defining the break points and slopes, each requiring a servo motor to drive it. When the command comes to increase the function over a certain range, the calculation of what the new pot settings should be is no small problem. In the end the entire system becomes too complex to be of use and would probably require a tertiary control system to control the function generator itself.

If the function is expanded along some spectral axes f_i, with weighting function p, then the increment to the coefficients a_i can be calculated from

$$\Delta a_i = \delta \int_a^b pf_i \, dx. \tag{6.1}$$

Provided the domain (a,b) is small enough so that p and f_i do not change appreciably over it, then the approximate form

$$\Delta a_i = (b-a)\, p\, (\tfrac{b+a}{2})\, f_i\, (\tfrac{b+a}{2})\, \delta \tag{6.2}$$

may be used. Assuming n functions f_i are required to represent the desired function, then again n servo multipliers or servo-pots are necessary to set the coefficients. The way in which the change must be made is easily calculated from (6.2) but it seems impossible to get away from the need for servo-multipliers, an expensive and complex device.

The piecewise linear functions discussed on page 33 would probably be the most convenient set for realizing an adaptive generator of the above kind. The base functions are easily obtained from diode generators and as the weighting function is one, equation (6.1) becomes exactly

$$\Delta a_i = \delta\, f_i\, (\tfrac{a+b}{2})\, (b-a)$$

for a and b on the same lineal segment. With a break point between a and b there is a small increment or decrement, depending on whether it is a positive or a negative break point.

For discrete functional realizations the command is usually to increase or decrease the value of the function for a specific x_0 by the amount δ. The correction to the coefficients is

$$\Delta a_i = \delta \; p(x_0) \; f_i(x_0) \tag{6.3}$$

which is easily enough obtained. Again potentiometers, with all their attendant difficulties, are required to set the coefficients. The only way of circumventing this problem is to go to an entirely digital method for obtaining the necessary adaptation facility.

6.2 Adaptive Digital Function Generators

All the analogue functions mentioned above are realized by defining a set of real numbers to represent the function, i.e., a set of expansion coefficients or pot settings. With the given hardware it is possible to realize an infinite number of functions, producing almost perfect compensation but aggravating the search problem considerably. By using digital functions this 'function space' becomes finite, reducing the search problem but giving something less than perfect compensation. If this space has k members then at least

$$n = \left\lceil \log_2 k \right\rceil \tag{6.4}$$

bits are required to distinguish a particular function, regardless of the way it is obtained. Thus the size of the class of functions which it is wished to realize is of primary importance as far as complexity is concerned. n, given by equation (6.4) will be called the 'entropy' of the class, in imitation of this word's use in information theory[33,52].

Suppose the requirement is to have a method of realizing all functions from k elements to l elements. There are l^k such functions and the entropy of this class is $\lceil k \log_2 l \rceil$. It may have been decided to form a look-up table in a core memory, with k different addresses specified by the input, and each range element being represented by $\lceil \log_2 l \rceil$ bits. This requires a total of $k \lceil \log_2 l \rceil$ bits. If l is a power of two, which it normally is, then these two expressions are equal and one sees that it is futile to try and get network representation for this class of all functions, as it cannot possibly be an improvement over a look-up table.

However, as shown in the preceding chapter, if any consideration is given to the quantization of the functions, at worst all functions 'continuous' of order μ are necessary. Assuming the spaces of definition above, and that the functions are of order μ, the total number of functions in the class becomes $l\mu^{k-1}$ and the entropy is

$$\lceil \log_2 l\mu^{k-1} \rceil = \lceil \log_2 l + (k-1)\log_2 \mu \rceil ,$$

considerably less than $\lceil k \log_2 l \rceil$ for the class of all functions. The l appears in the equation for the size of the class as the initial condition may be chosen in l different ways. Again the best realization would be to have a store containing $(k-1) \lceil \log_2 \mu \rceil$ bits, if it is desired that all possible functions be contained in the adaptive networks. This class still seems too large for any practical realization. Even if μ is only 2, the introduction of at least $\lceil (\log_2 l)+k-1 \rceil$ new variables is involved. k, the number of elements in the domain, may be quite considerable and the resultant network may be very complex. Most members of this class, the members that oscilate rapidly, do not occur in practice and it is advisable to keep such functions out of any adaptive loops as, in all probability, they will produce spurious behaviour. Analytically, this represents a restriction on the size of the second and higher order derivatives that occur in the associated continuous function.

Hence the importance of determining beforehand the exact class of functions that may occur in the adaptation loop is evidenced, as the main concern is to reduce the entropy of this class. Having decided on the members of the class all that is necessary is to introduce enough new variables to distinguish each member and form a single function of this larger number of variables.

The purpose of adaptation is to keep a system operating optimally in spite of changes in the plant dynamics. If the plant dynamics and the range of variation of the parameters are known, or if a model has been obtained that simulates the plant satisfactorily and the variation of the model parameters is known, then the portion of parameter space covered by the variation can be found. Each point in parameter space has associated with it some optimal compensation function. Hence this space maps into some region of the function space.

Every digital function that has its x and y quantization levels defined, covers some volume of the function space, i.e., all members of that volume map into the same digital function when quantized. This is effectively the same thing as quantizing the function space. The class of digital functions needed is the class that covers the region in function space defined in the above paragraph.

Analytically the above discussion reduces to the chain of relationships shown below. The basic mechanics involve a system with known parameters that has an adaptive function g somewhere in its configuration. A criterion function I is also given, its value being dependent on the input $\underline{u}$, the state $\underline{x}$ the function g and the system parameters $\underline{a}$.

$$I = I(\underline{u}, \underline{x}, \underline{a}, g).$$

It is required to minimize this functional with respect to g. Given $\underline{u}$, $\underline{a}$ and g (i.e., the input and the system dynamics) $\underline{x}$ is determined. Hence

$$I = I'(\underline{u}, \underline{a}, g). \tag{6.5}$$

For a set of parameters $\underline{a}$ and an input $\underline{u}$ the minimization of (6.5) determines the optimal g. The space of variation for $\underline{a}$ is $\mathcal{A}$ and for $\underline{u}$, $\mathcal{U}$. Therefore a mapping s as shown below is properly defined where $\mathcal{G}$ is the resultant space of g.

$$\mathcal{A} \times \mathcal{U} \xrightarrow{s} \mathcal{G} \tag{6.6}$$

The mapping s is the solution for g from (6.5). A second mapping t, resulting from the quantization of $\mathcal{G}$, defines a subspace $\mathcal{F}$ of the digital function space. s and t may be combined into the single function v yielding

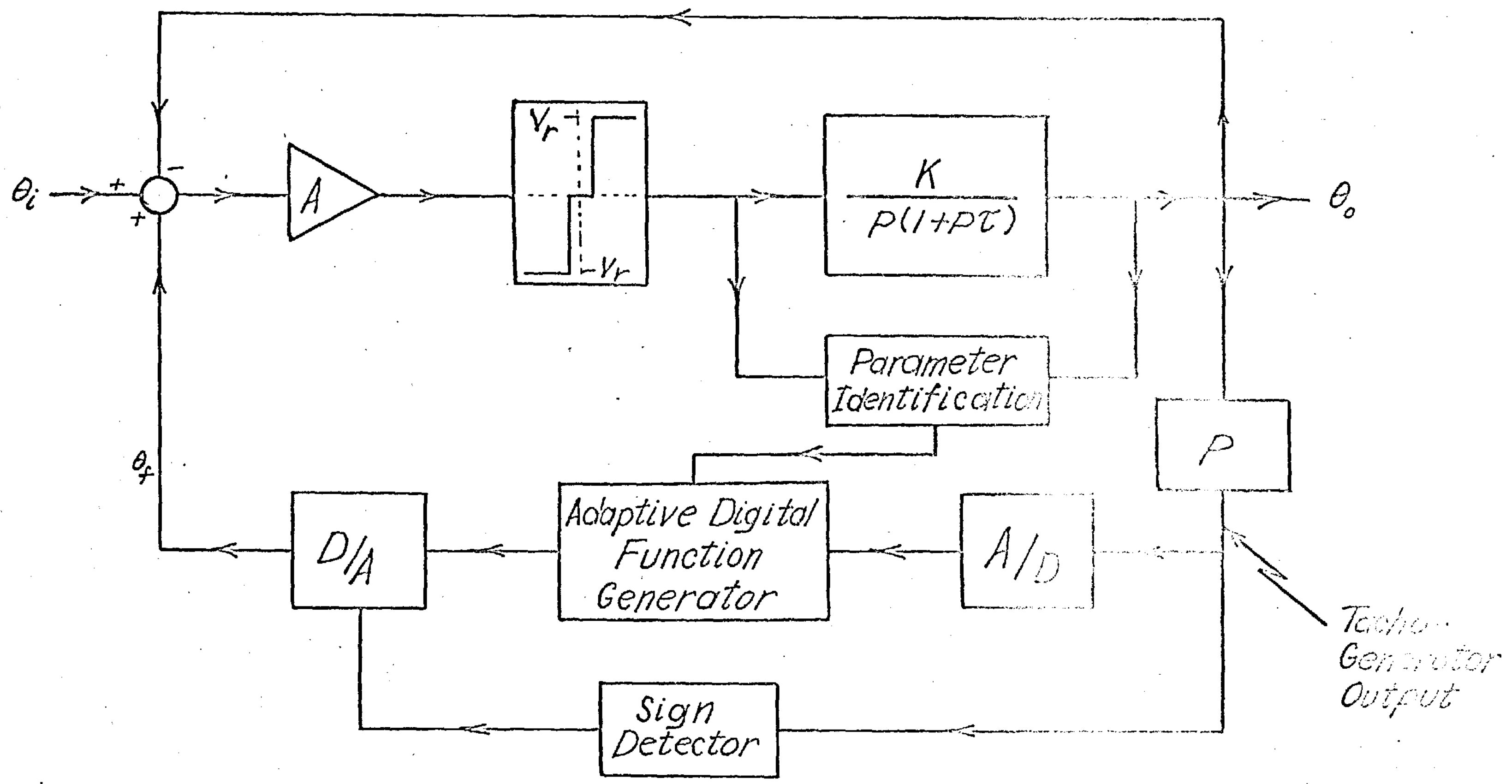
$$\mathcal{A} \times \mathcal{U} \xrightarrow{v} \mathcal{F} \tag{6.7}$$

The number of elements in $\mathcal{F}$ determines the number of functions that must be realized. Note that, although the mapping (6.6) is usually difficult, if not impossible to find, (6.7) may be a great deal simpler as $\mathcal{F}$ represents a finite set. v partitions the space $\mathcal{A} \times \mathcal{U}$ into disjoint regions making it necessary to decide only in which region the system is operating to determine which digital function to use. An example will help to clarify some of the above points as well as illustrating some of the difficulties encountered in a practical case.

6.3 Application of a Digital Function Generator to a Time Optimal Position Control System

The system considered is a time optimal, second order, position control system. This choice was made as the mathematical expression for the optimal switching function is easily obtained, allowing the example to be solved entirely by analytic methods. For more complicated systems a computer study would probably be needed to determine the necessary digital functions, although the final realization, in terms of hardware, need not be more complex. The theory of operation of this system is described thoroughly in reference 53.

A block diagram is given in figure (6.1), it being just a particular case of the relay controlled systems mentioned in the Introduction. The input θ_i consists of a series of step functions with the assumption that after any given step, the system comes to rest before the arrival of the next step. The relay coil is driven by the error signal, plus some function of the velocity $(\dot{\theta}_0)$ of the output. When the signal to the relay coil reverses sign a maximum decelerating torque is applied to the system. Provided the function of the velocity fed back gives the angle that would result if reverse power was applied at that speed, and the velocity allowed to drop to zero; switching then occurs at the correct instant to ensure that the system falls to zero error with zero velocity. At this point, due to the dead zone in the relay, no input is applied and the system remains at rest.



Time Optimal Position Control System

Figure 6.1

The required feedback function[53] is

$$y = x - \ln(1 + x) \tag{6.7}$$

where $y = \frac{\theta_0}{\dot{\theta}_m \tau}$ and $x = \frac{\dot{\theta}_0}{\dot{\theta}_m}$. $\dot{\theta}_m$, the maximum velocity of the motor, is KV_r. As this function must be symmetric only one half of it need be realized explicitely, the sign being taken care of by the outer loop as shown. Suppose that changes in the time constant τ and in the gain KV_r are to be compensated for. To assign numerical values, say that the space $\mathcal{A}$ of parameter variations is

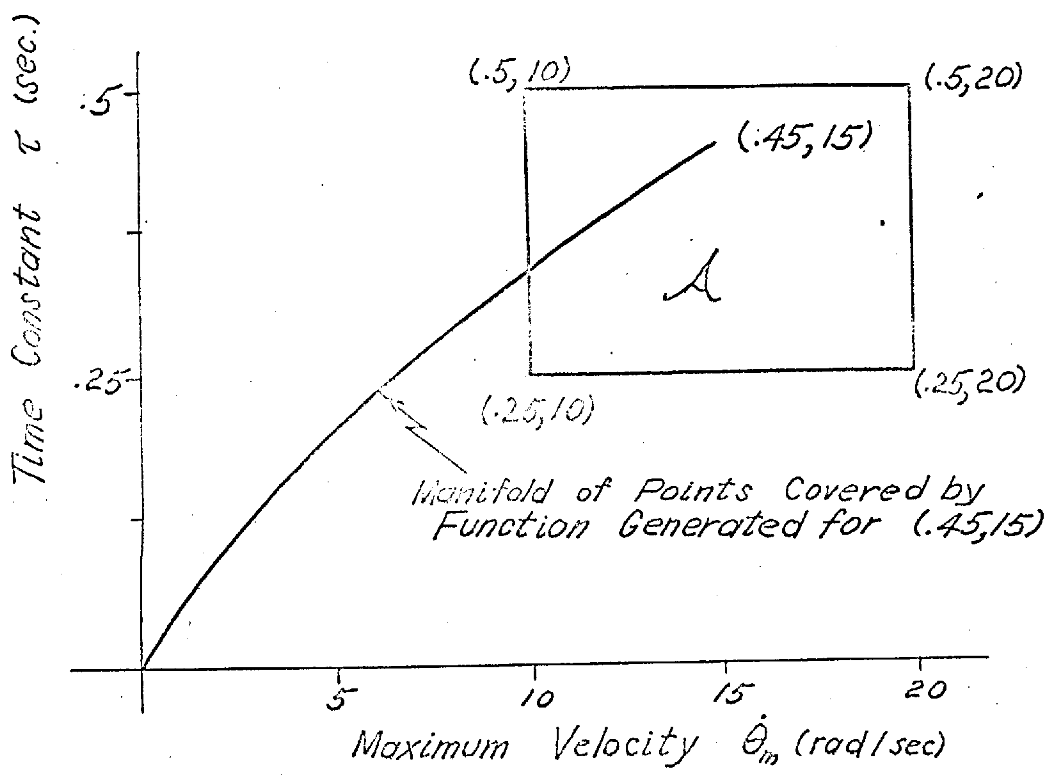
$$\mathcal{A} = \left\{ (\tau, KV_r) \ : \ \begin{matrix} .25 < \tau < .5 \\ 10 < KV_r < 20 \end{matrix} \right\}$$

where normal set notation has been used. This space is shown by the rectangle in figure (6.2^a). As $(\tau, \dot{\theta}_m)$ varies over this region equation (6.7) maps the feedback function into the shaded region of figure (6.2^b). The functions as determined by the four corners of $\mathcal{A}$, (.25, 10), (.25, 20), (.5, 10), (.5, 20), are shown by the heavy lines. The shaded region corresponds to members of the set $\mathcal{G}$, defined in the preceeding section by equation (6.6).

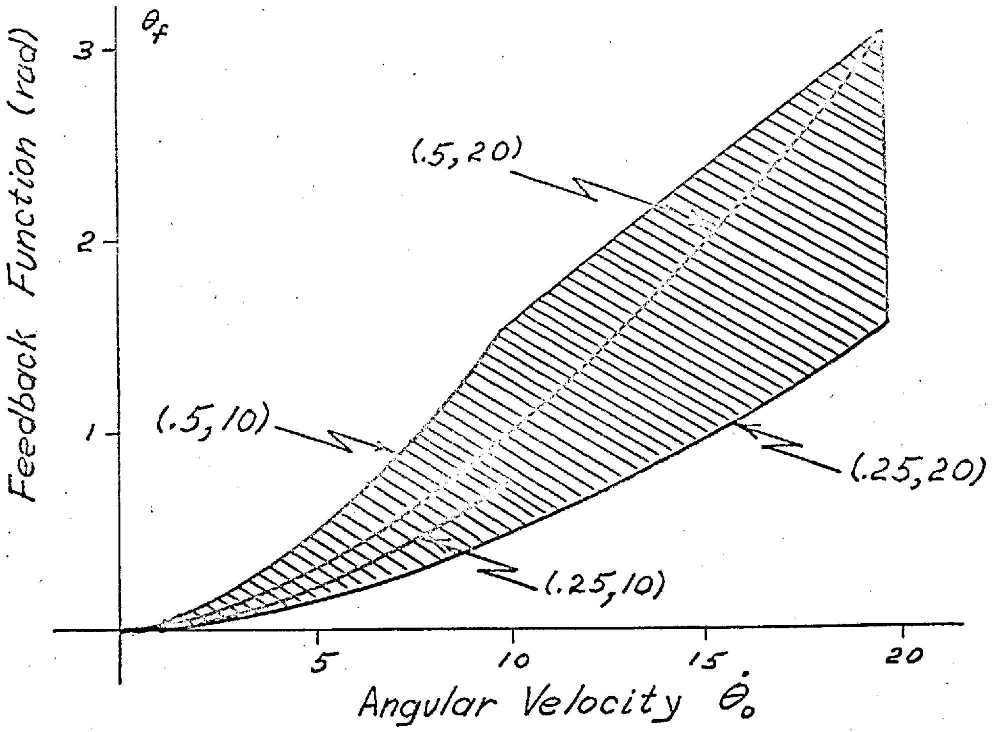
Assume the adaptive digital function generator, used to obtain representative members of this class, quantizes both the domain (x) and the range (y) into 16 levels. Then all member functions may be defined by digital functions of order two. Also, as the functions are monotonically increasing, three levels, and only two boolean functions, are required.

In terms of the primary variables, $\dot{\theta}_0$ and θ_0, equation (6.7) is

$$\theta_0 = \tau \left\{ \dot{\theta}_0 - \dot{\theta}_m \ln\left(1 + \frac{\dot{\theta}_0}{\dot{\theta}_m}\right) \right\} . \tag{6.8}$$



Space of Parameter Variations



Space of Feedback Function Variation $\mathcal{G}$

Figure 6.2

Looking at the curve for $(\tau, \dot{\theta}_m) = (.25, 10)$ in figure (6.2^a), it appears that this curve may be well approximated by some curve with a greater $\dot{\theta}_m$ (greater domain) and a different τ. Suppose the switching function for $\dot{\theta}_{m_1}$ and τ_1 has been obtained. The assumption that this curve is a good approximation for all curves with a smaller $\dot{\theta}_m$ that have their end points on the original curve, is employed. The equation for all curves with $\dot{\theta}_m < \dot{\theta}_{m_1}$ and any time constant may be expressed as

$$\theta_0 = \tau \left\{ \dot{\theta}_0 - \alpha\dot{\theta}_{m_1} \ln(1 + \frac{\dot{\theta}_0}{\alpha\dot{\theta}_{m_1}}) \right\}, \qquad 0 \leq \alpha < 1 \tag{6.9}$$

The end points of these equations are where $\dot{\theta}_0 = \dot{\theta}_m = \alpha\dot{\theta}_{m_1}$ and therefore may be found from

$$\theta_0 = \tau \left\{ \alpha\dot{\theta}_{m_1} - \alpha\dot{\theta}_{m_1} \ln 2 \right\} = .307\ \alpha\dot{\theta}_{m_1}\tau. \tag{6.10}$$

The value of the original curve at this point is

$$\theta_0 = \tau_1 \left\{ \alpha\dot{\theta}_{m_1} - \dot{\theta}_{m_1} \ln(1 + \alpha) \right\}.$$

These two are constrained to be equal implying that the new τ is

$$\tau = \tau_1 \left\{ \alpha - \ln(1 + \alpha) \right\} / (.307\ \alpha). \tag{6.11}$$

Therefore, having the feedback function for the point $(\tau_1, \dot{\theta}_{m_1})$ the function for the entire manifold

$$(\tau_1\ (\alpha - \ln(1 + \alpha))/(.307\alpha),\ \alpha\dot{\theta}_{m_1}) \qquad 0 \leq \alpha \leq 1$$

is also available. Such a manifold is illustrated by the heavy line in figure (6.2^a). Thus only the functions for the maximum values of $\dot{\theta}_m$ and τ need be realized, all functions in the interior of the space being covered by one of these boundary functions. This illustrates the saving that may be made over merely quantizing the parameter space, $\mathcal{A}$, and realizing a feedback function for each quantized area separately.

Suppose it is wished to span the space $\mathcal{G}$ with eight representative feedback functions corresponding to eight points on the boundary of $\mathcal{A}$. Figure (6.3^a) shows the eight points chosen as well as the eight parameter regions covered by these points. These points could have been chosen with a lot more care, for example by choosing them so that the proportion of $\mathcal{A}$ covered by each point was $\frac{1}{8}$ of the total. Also, if the statistical distribution of the parameters was known, they could have been chosen to ensure that the probability of being in any region was $\frac{1}{8}$. However, these are just frills on the main problem and could only serve to complicate matters.

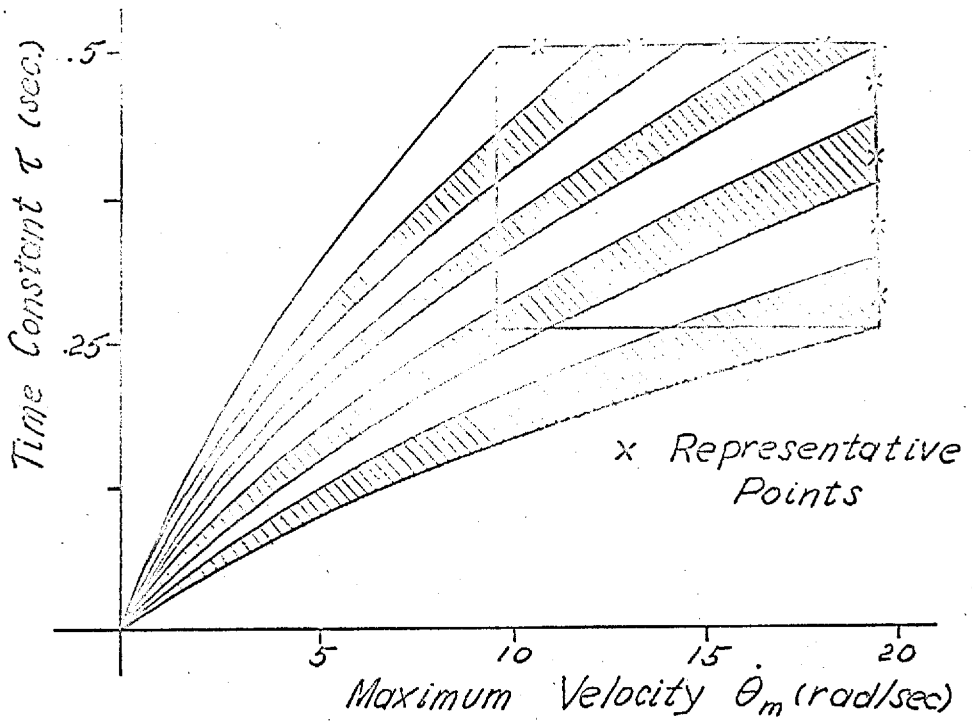
The eight points in parameter space map into the eight curves of figure (6.3^b). (6.3^b) also shows the regions that are represented by each curve. Quantizing these eight curves into sixteen levels, and using the incremental realization method of the previous chapter for obtaining function of order 2, the set of functions below are obtained. μ, the increment, is determined by its corresponding binary equivalent, f_α, being the least significant figure, f_β the most significant. The third function in each group f_a is the set over which the function may be chosen arbitrarily.

$$f_\alpha = \begin{pmatrix}3,5,7,9,10\\11,12,13,14\end{pmatrix} \qquad f_\alpha = \begin{pmatrix}3,5,6,7,9\\10,11,12,13\end{pmatrix}$$

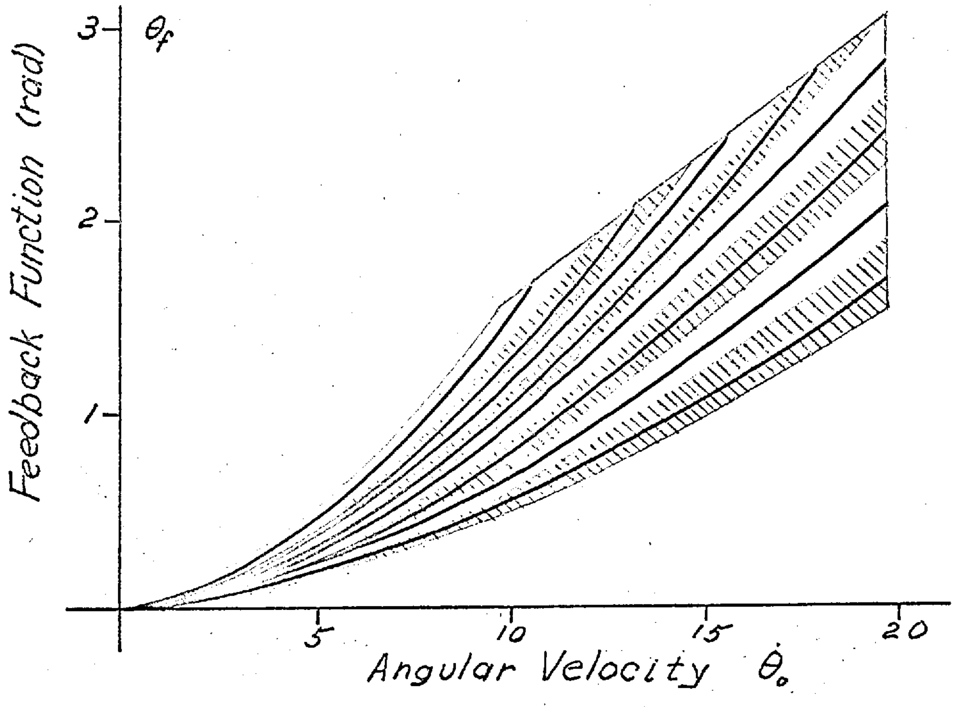
$$f_\beta = (\text{null}) \qquad f_\beta = (14)$$

$$f_a = (15) \qquad f_a = (15)$$

$$f_1 \qquad\qquad f_2$$



Representative Regions of the Parameter Space $\mathcal{A}$



Representative Curves of the Function Space $\mathcal{G}$

Figure 6.3

$f_\alpha = \begin{pmatrix}3,4,6,7,8\\9,10,12,13\end{pmatrix}$
$f_\beta = (11,14)$
$f_a = (15)$

f_3

$f_\alpha = \begin{pmatrix}2,4,5,6,7\\8,9,11,13\end{pmatrix}$
$f_\beta = (10,12,14)$
$f_a = (15)$

f_4

$f_\alpha = \begin{pmatrix}2,4,5,6\\8,9,11\end{pmatrix}$
$f_\beta = (7,10,12,13)$
$f_a = (14,15)$

f_5

$f_\alpha = \begin{pmatrix}2,3,4,5\\6,7,9\end{pmatrix}$
$f_\beta = (8,10,11)$
$f_a = (12,13,14,15)$

f_6

$f_\alpha = (2,3,4,5,7,9)$
$f_\beta = (6,8)$
$f_a = (10,11,12,13,14,15)$

f_7

$f_\alpha = (1,3,5,7)$
$f_\beta = (4,6)$
$f_a = (8,9.....15)$

f_8

Again, as $\mu=2$, it is possible to use the fact that for $f_\beta=1$, f_α must be zero and the simplification of the previous example (page 118) results. Three more variables y_1, y_2, y_3, can be introduced to distinguish each of the eight functions above. Hence, two functions, f and g, of the seven variables $y_1, y_2, y_3, x_1, x_2, x_3, x_4$, are obtained. The three new variables are introduced by defining $f_{000} = f_{1\alpha}$, $g_{000} = f_{1\beta}$, $f_{001} = f_{2\alpha}$, $g_{001} = f_{2\beta}$ etc., and then forming the corresponding conjunctive normal forms. These two new functions may then be realized by minimal nets, again noting that f may be chosen arbitrarily on members of g if one more gate on the output of the net is allowed.

Carrying the synthesis procedure through results in a net containing something less than 55 elements in all, to realize the two functions required. This may seem somewhat excessive, but for two functions of seven variables, and considering that a fan-out of only one is permitted, it is quite reasonable. If the fan-out was increased the number may fall to about half the above estimate due to considerations mentioned in Chapter II.

The values of y_1, y_2, y_3 are set according to the region of the parameter space in which the system is operating (see figure (6.3^a)). The determination of this region may be a difficult problem as the boundaries are defined by rather complicated expressions. This problem can be avoided by using a performance evaluator rather than trying to estimate the parameters of the system. The criterion is that the system should come to rest in minimum time, implying that the relay should switch only once and the system come to rest with zero velocity and zero error. If too much velocity feedback is applied the system switches too soon and the relay goes into a rapidly oscillatory mode[53] in its attempt to bring the error to zero. If too little feedback is used and the relay switches too late then zero error occurs before zero velocity. Both of these conditions are easily detectable from the relay input and output. Hence, whether to choose a higher or lower curve, may be decided merely by observing the relay behaviour without doing any system identification.

In the more general case, where nothing is known about the system, a function generator that realizes a much larger class will be required. Here the search problem must be considered and the folly of having too many functions in the class (too high an entropy) is evident as the convergence to the correct result will then be slow. However, quantizing a function space in an optimal manner is no easy matter, as illustrated by the simple example above. Exactly how this should be done requires a great deal more study and much deeper probing into the now rapidly expanding field of functional analysis.

The necessity of identifying the system in order to determine the compensating function has been mentioned above. The next chapter takes some measures to solve the identification problem and also contains a number of computed examples to substantiate its claims.

CHAPTER VII

Discrete Functions and On-Line Process Identification

7.1 System Representation

A system[51] is defined by a mapping between two sets of function spaces, one called the input space, the other the output space. This definition is far too general for any practical use and almost immediately some further structure must be imposed. This further structure is usually the assumption of linearity, implying that if $f_1(t)$ and $f_2(t)$ are two different members of the input space, the corresponding members of the output space being $g_1(t)$ and $g_2(t)$; then the member of the output space corresponding to $af_1(t) + bf_2(t)$ is $ag_1(t) + bg_2(t)$. This linear property allows the use of all the spectral analysis results developed before.

If the input function $f(t)$ is expanded along some base axes as $f(t) = \sum_{i=0}^{\infty} a_i f_i(t)$ and the way the system maps the set $\left\{f_i(t)\right\}$ into the output is known, then the way the system maps all functions is known by the linearity property. Thus, given the mapping m

$$f_i(t) \xrightarrow{m} g_i(t),$$

then the output function can be constructed by mapping the component vectors separately and performing the corresponding sum, yielding $g(t) = \sum_{i=0}^{\infty} a_i g_i(t)$. Note that now the mapping m represents the system and is dependent upon the particular set $\left\{f_i(t)\right\}$ chosen.

The further restriction to 'linear differential systems' or 'lumped parameter systems' is made. This implies that the input function $f(t)$

and the output function g(t) satisfy a differential equation of the form

$$\sum_{i=0}^{n} a_i \frac{d^i g}{dt^i} = \sum_{i=0}^{m} b_i \frac{d^i f}{dt^i} \qquad (7.1)$$

where the zeroth derivative is the function itself. For practical systems $m \leqslant n$, making this an nth order differential equation. Many systems do indeed obey a relation of this kind or may be approximated closely by such an equation. The differential equation along with the spectral functions $\left\{f_i(t)\right\}$ define the mapping m.

To clarify the above discussion consider the classical frequency response method of control engineering. Here the base set of function is chosen as

$$\left\{f(t,\omega)\right\} = \left\{\sin \omega t,\ \cos \omega t\right\} \qquad 0 \leqslant \omega < \infty \qquad (7.2)$$

and form an orthogonal set over the range $(-\infty,\infty)$. An arbitrary function f(t) may be expressed as a sum of these (in this case an integral as ω is a continuous variable)

$$f(t) = \int_{-\infty}^{\infty} F_s(\omega) \sin \omega t \, d\omega + \int_{-\infty}^{\infty} F_c(\omega) \cos \omega t \, d\omega \qquad (7.3)$$

where $F_s(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t) \sin \omega t \, dt,\ F_c(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t) \cos \omega t \, dt.$ (7.4)

The factor $\frac{1}{2\pi}$ is the normalizing constant for the functions (7.2). Presenting a base function of the form $\sin \omega t$ to a linear differential system the output is a sine wave of the same frequency with an attenuated amplitude and a phase shift.

$$\sin \omega t \xrightarrow{m} M(\omega) \sin (\omega t + P(\omega))$$

similarly

$$\cos \omega t \xrightarrow{m} M(\omega) \cos (\omega t + P(\omega)).$$

The system mapping m is determined by the two functions $M(\omega)$, the magnitude response, and $P(\omega)$, the phase response. Evidently if an input of the form (7.3), represented by 'coefficients' (7.4) is presented, then the output will be given by

$$g(t) = \int_{-\infty}^{\infty} F_s(\omega)\, M(\omega)\, \sin\,(\omega t + P(\omega))d\omega$$
$$+ \int_{-\infty}^{\infty} F_c(\omega)\, M(\omega)\, \cos\,(\omega t + P(\omega))d\omega.$$

The output frequency responses $G_s(\omega)$ and $G_c(\omega)$ as found from equation (7.3) are

$$G_s(\omega) = \frac{M(\omega)}{2\pi}\left\{F_s(\omega)\, \cos\,(P(\omega)) - F_c(\omega)\, \sin(P(\omega))\right\}$$
$$G_c(\omega) = \frac{M(\omega)}{2\pi}\left\{F_s(\omega)\, \sin\,(P(\omega)) + F_c(\omega)\, \cos(P(\omega))\right\}.$$

Solving this for $M(\omega)$ by squaring and adding the above two equations

$$M(\omega) = 2\pi \sqrt{\frac{G_s^2(\omega) + G_c^2(\omega)}{F_s^2(\omega) + F_c^2(\omega)}}\ . \qquad (7.5)$$

Taking the ratio and solving for $P(\omega)$ yields

$$P(\omega) = \tan^{-1}\left(\frac{F_c(\omega)\; G_s(\omega) - F_s(\omega)\; G_c(\omega)}{F_c(\omega)\; G_c(\omega) - F_s(\omega)\; G_s(\omega)}\right). \qquad (7.6)$$

Note that if formally the complex functions

$$F(\omega) = F_c(\omega) + j\ F_s(\omega)$$

and

$$G(\omega) = G_c(\omega) + j\ G_s(\omega)$$

are defined, then equations (7.5) and (7.6) are obtained by taking the magnitude and phase of the function $2\pi G(\omega)/F(\omega)$.

$$M(\omega) = 2\pi \left|\frac{G(\omega)}{F(\omega)}\right| \qquad P(\omega) = \text{Arg}\left|\frac{G(\omega)}{F(\omega)}\right|.$$

Usually the 2π is associated with (7.3) and not (7.4) as shown. $G(\omega)/F(\omega)$ is the classical transfer function of the system used throughout all frequency response analysis, and is an invariant for the system regardless of the input.

Another, perhaps more exotic, form of this spectral analysis method is the so-called 'impulse response' representation. Using this the input is assumed to be made up of a series of impulses, or Dirac functions, weighted according to the value of the function at that instant. The impulse function $\delta(t)$ is defined by the property that for $t \neq 0$ $\delta(t) = 0$ but $\int_{-\infty}^{\infty} \delta(t) = 1$. The complete set of spectral functions is $\delta(t-x)$, orthogonal for all different values of x.

$$\int_{-\infty}^{\infty} \delta(t-x)\ \delta(t-x')\ dt = 0 \qquad x \neq x'.$$

An arbitrary function $f(t)$ may be expanded in terms of these, the expansion 'coefficients' being found from

$$\int_{-\infty}^{\infty} f(t)\ \delta(t-x)\ dt = f(x). \tag{7.7}$$

Thus the spectrum and the function are identical.

The response of the system to an impulse at time zero is denoted by $h(t)$, yielding the mapping from the spectral function to the output as

$$\delta(t-x) \xrightarrow{m} h(t-x) \qquad t \geqslant x. \tag{7.8}$$

Assuming the spectra given by (7.7) is the input to a system and using the mapping defined by (7.8) the output function $g(t)$ is

$$g(t) = \int_{-\infty}^{\infty} f(x)\ h(t-x)\ dx = \int_{-\infty}^{t} f(x)\ h(t-x)\ dx \tag{7.9}$$

as $h(t) = 0$ for $t < 0$. This is the familiar convolution integral. The system is represented by the function $h(t)$ and must be obtained by deconvolution of equation (7.9). Actually the spectra of the impulse response is the frequency response function mentioned earlier[54] and so may be found merely by taking the inverse transform of $\frac{G(\omega)}{F(\omega)}$. Alternatively it may be obtained directly using correlation with noise[12], pseudo-random binary chain codes[12,15] or by direct analysis using the naturally occurring signals[54].

The most well-known system representation is the Laplace transform. Here the expansion functions are of the form $e^{(\alpha+j\omega)t}$ for all real α and ω. These have the same advantageous property as the frequency response functions in that they change the differential equation (7.1) into an algebraic equation that conforms much more readily to analysis. This is the standard transform used throughout all control and circuit theory engineering.

7.2 On-Line Identification

Our goal is to find a method for identifying linear differential systems using the naturally occurring input and output signals. The system may be completely specified by the coefficients a_i and b_i in equation (7.1), by the frequency or impulse response, or by the pole-zero pattern that arises out of the Laplace transform approach.

Although the representations mentioned above are excellent when considering the response of known systems, or synthesizing higher order systems from lower order systems, they do not fit easily into any on-line identification schemes. The frequency response method requires a continuous spectrum of the input and output waveforms and the evaluation of the functions in equations (7.5) and (7.6). Although this is convenient for off-line analysis, where it is possible to drive the system with the sinusoidal test signals and obtain the resultant Bode plots[55], for on-line work it requires spectrum analyzers on both the input and output of the system. This has been tried[12] with limited success. Many schemes have been developed for obtaining the impulse response directly by application of suitable test signals but these involve long correlation analyses, resulting in a long identification time. Also, having the impulse response it is still no small matter to convert it to the coefficients in (7.1) or to the frequency response as this involves finding a fourier transform which, in general, is not an easy operation[56].

Below a method of extracting the coefficients of (7.1) directly using expansions in Laguerre function is given. These polynomials have been defined in section (2.4.3) and have, for our purposes, one other interesting property. Instead of defining the functions on the range $(0,\infty)$ it is desired to define them on $(-\infty, 0)$, necessitating the change in variables $x \to -x$. These are called negative Laguerre polynomials and are denoted by $\Lambda_n(x)$. The Laguerre functions may be obtained from the differential equation

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x}) \qquad x \geqslant 0$$

Making the change in variables the negative Laguerre functions are generated by

$$\Lambda_n(x) = \frac{e^{-x}}{n!} \frac{d^n}{dx^n} (x^n e^x) \qquad x \leqslant 0 \qquad (7.9)$$

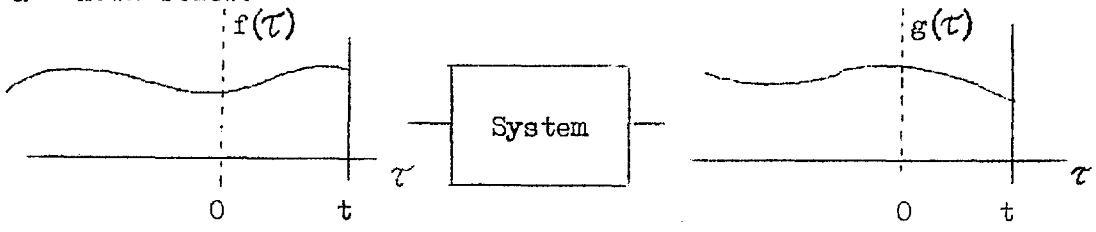
The first few may be found by replacing x by -x in the examples given in section (2.4.3). These functions satisfy the differential relation

$$\frac{d}{dx}\Lambda_n(x) = \sum_{i=0}^{n-1} \Lambda_i(x). \tag{7.10}$$

This may be proved by a simple induction using the definition of $\Lambda_n(x)$ given in equation (7.9). Its extension to the r^{th} differentiation is

$$\frac{d^r}{dx^r}\Lambda_n(x) = \sum_{i=0}^{n-r} \binom{n-i-1}{r-1}\Lambda_i(x). \tag{7.11}$$

The input and output of a system, at any time t, are functions defined over the half line $(-\infty, t)$, making a representation by expansions in the negative Laguerre polynomials possible. The situation is as shown below.



The input may be represented by

$$f(\tau,t) = \sum_{i=0}^{\infty} \alpha_i(t)\Lambda_i(\tau-t) \tag{7.12}$$

where

$$\begin{aligned}\alpha_i(t) &= \int_{-\infty}^{t} e^{\tau-t} f(\tau)\,\Lambda_i(\tau-t)\,d\tau \\ &= \int_{-\infty}^{0} e^{\tau} f(t+\tau)\,\Lambda_i(\tau)\,d\tau.\end{aligned} \tag{7.13}$$

Similarly

$$g(\tau,t) = \sum_{i=0}^{\infty} \beta_i(t)\,\Lambda_i(\tau-t) \tag{7.14}$$

where

$$\beta_i(t) = \int_{-\infty}^{0} e^{\tau} g(t+\tau)\,\Lambda_i(\tau)\,d\tau. \tag{7.15}$$

Assuming the system is represented by an equation like (7.1) then at time t the input and output function shown above must satisfy

$$\sum_{i=0}^{n} a_i \frac{d^i g(\tau,t)}{d\tau^i} = \sum_{i=0}^{m} b_i \frac{d^i f(\tau,t)}{d\tau^i} \qquad t \geqslant \tau \qquad (7.16)$$

Expanding $g(\tau,t)$ and $f(\tau,t)$ by equation (7.12) and (7.14) and substituting $\tau - t = x$ yields

$$\sum_{j=0}^{\infty} \beta_j(t) \sum_{i=0}^{n} a_i \frac{d^i \Lambda_j(x)}{dx^i} = \sum_{j=0}^{\infty} \alpha_j(t) \sum_{i=0}^{m} b_i \frac{d^i \Lambda_j(x)}{dx^i} \qquad (7.17)$$

The upper limit of (7.11) may be changed to n-1 as $\binom{a}{b} = 0$ for $a < b$. Also the upper index m on the right of equation (7.17) may be changed to n by introducing enough new coefficients b_i all equal to zero. Then the right and left sides of equation (7.17) are entirely symmetric so, for the moment, all operations need only be performed on the left side as the right side must assume an identical form.

Substituting the modified result of (7.11) into (7.17) changes the differential equation into a linear equation in $\beta_j(t)$, $\alpha_j(t)$, a_i, b_i and $\Lambda_j(x)$ of the form

$$\sum_{j=0}^{\infty} \beta_j(t) \left\{ \sum_{i=1}^{n} a_i \sum_{k=0}^{j-1} \binom{j-k-1}{i-1} \Lambda_k(x) + a_0 \Lambda_j(x) \right\} \qquad (7.18)$$

Interchanging the summations with respect to k and j yields

$$\sum_{k=0}^{\infty} \left\{ \sum_{j=k+1}^{\infty} \beta_j(t) \sum_{i=1}^{n} a_i \binom{j-k-1}{i-1} \Lambda_k(x) + a_0 \beta_k(t) \Lambda_k(x) \right\}.$$

As the functions $\Lambda_k(x)$ are linearly independent, for this relation to be satisfied the coefficients must be equal, resulting in the following relationship between the input and output coefficients, the right hand side being tacitly assumed as above

$$\sum_{j=k+1}^{\infty} \beta_j(t) \sum_{i=1}^{n} a_i \binom{j-k-1}{i-1} + a_0\beta_k(t) \qquad k=0,1,2.. \quad (7.19)$$

These represent an infinite set of linear equations, each one having infinitely many terms; a rather formidable object on first sight. Interchanging the summations over i and j and noting again that $\binom{a}{b} = 0$ for $a < b$, equation (7.19) assumes the form

$$\sum_{i=1}^{n} a_i \sum_{j=k+i}^{\infty} \beta_j(t) \binom{j-k-1}{i-1} + \beta_k(t)\, a_0 \qquad k=0,1,2.. \quad (7.20)$$

Taking this infinite set of equations and forming a new set by subtracting the $(k+1)^{th}$ from the k^{th} yields

$$\sum_{i=1}^{n} a_i \sum_{j=k+i+1}^{\infty} \beta_j(t) \left\{\binom{j-k-1}{i-1} - \binom{j-k-2}{i-1}\right\} + \sum_{i=1}^{n} a_i\beta_{k+i}(t) \quad (7.21)$$
$$+ (\beta_k(t) - \beta_{k+1}(t))a_0.$$

For i=1 the term in $\{\ \}$ is zero while for all other i it equals $\binom{j-k-2}{i-2}$. Therefore equation (7.21) becomes

$$\sum_{i=2}^{n} a_i \sum_{j=k+i+1}^{\infty} \beta_j(t) \binom{j-k-2}{i-2} + \sum_{i=1}^{n} a_i\, \beta_{k+i}(t) \quad (7.22)$$
$$+ (\beta_k(t) - \beta_{k+1}(t))\, a_0.$$

For this new set of equations the process can be repeated, namely subtract the $(k+1)^{th}$ equation from the k^{th} equation. After all possible simplifications are made the resulting set is

$$\sum_{i=3}^{n} a_i \sum_{j=k+i+2}^{\infty} \beta_j(t) \binom{j-k-3}{i-3} + \sum_{i=2}^{n} a_i \left[\beta_{k+i}(t) + (i-2)\, \beta_{k+i+1}(t)\right] \quad (7.23)$$
$$+ a_1(\beta_{k+1}(t) - \beta_{k+2}(t)) + a_0(\beta_k(t) - 2\beta_{k+1}(t) + \beta_{k+2}(t)).$$

The infinite number of terms is still contained in the first expression and the bottom index has now been increased to three.

Continuing the above subtraction process for n steps produces the final expression, including the right hand side

$$\sum_{i=0}^{n} a_i \sum_{j=0}^{n-i} (-1)^j \beta_{k+j+i}(t) \binom{n-i}{j} = \sum_{i=0}^{n} b_i \sum_{j=0}^{n-i} (-1)^j \alpha_{k+j+i}(t) \binom{n-i}{j} \qquad (7.24)$$
$$k=0,1,2.....$$

an infinite set of finite equations. These are more conveniently represented using matrix notation. Define $\underline{\alpha}(t)$ and $\underline{\beta}(t)$ as two infinite vectors representing the input and output spectra, the vectors $\underline{a}^T = (a_0, a_1, ... a_n)$, $\underline{b}^T = (b_0, b_1, ... b_n)$, and a matrix D_k^n with an infinite number of rows and n+1 columns. The first k rows and all rows beyond the (k+n+1)th of this matrix are zero. The (k+1)th to (k+n+1)th rows are defined by an (n+1)x(n+1) matrix D^n such that $d_{i,j} = 0$ for $i < j$ and $= (-1)^{i+j}\binom{n-j}{i-j}$ for $i \geqslant j$. Two of these D matrices for n = 3 and n = 5 are given below.

$$D^3 = \begin{Vmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 3 & -2 & 1 & 0 \\ -1 & 1 & -1 & 1 \end{Vmatrix} \qquad D^5 = \begin{Vmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -5 & 1 & 0 & 0 & 0 & 0 \\ 10 & -4 & 1 & 0 & 0 & 0 \\ -10 & 6 & -3 & 1 & 0 & 0 \\ 5 & -4 & 3 & -2 & 1 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 \end{Vmatrix}$$

Using this notation equation (7.24) becomes

$$\underline{\beta}(t)\, D_k^n\, \underline{a} = \underline{\alpha}(t)\, D_k^n\, \underline{b} \qquad k=0,1,2..... \qquad (7.25)$$

This, although not convenient for calculating the input-output behaviour, does give a useful method for determining the coefficients a_i and b_i, of the differential equation. Normalizing these equations (7.24) so that $a_n = 1$, each of them then represents a linear equation in the remaining 2n+1 coefficients $b_0, \ldots\ldots b_n$, $a_0, \ldots\ldots a_{n-1}$. The first 2n+1 equations are linearly independent, this fact following from the independence of the separate coefficients comprising $\underline{\alpha}$ and $\underline{\beta}$. Note that each equation has at least one of these spectral coefficients different. Using the representation (7.25) the first 2n+1 equations can be put in the matrix form

$$B(t)D^n\underline{a} = A(t)D^n\underline{b} \tag{7.26}$$

where $B(t)$ is the $(2n+1)\text{x}(n+1)$ matrix of the spectral coefficients $\beta_i(t)$ shown below

$$\left\|\begin{matrix} \beta_0(t), & \beta_1(t) & \ldots\ldots\ldots & \beta_n(t) \\ \beta_1(t) & \ldots\ldots\ldots & \ldots\ldots\ldots & \\ \vdots & & & \vdots \\ \beta_{2n}(t), & \beta_{2n+1}(t) & \ldots\ldots & \beta_{3n}(t) \end{matrix}\right\|$$

$A(t)$ is the same kind of matrix comprised of the coefficients $\alpha_i(t)$. By forming the compound $(2n+1)\text{x}(2n+2)$ matrix $C(t) = \left\|B(t) \,\vdots\, -A(t)\right\|$, the vector $\underline{c} = \begin{bmatrix} \underline{a} \\ \cdots \\ \underline{b} \end{bmatrix}$ and the $(2n+2)\text{x}(2n+2)$ square matrix $\Delta = \left\|\begin{array}{c|c} D^n & 0 \\ \hline 0 & D^n \end{array}\right\|$ equation (7.26) takes the more recognizable form

$$C(t)\, \Delta\, \underline{c} = \underline{0}\ . \tag{7.27}$$

Note that one of the components of $\underline{c}$ is 1, introducing a constant into each of these linear equations. Knowing the matrix $C(t)$, it is possible to solve this set for the required coefficients $\underline{c}$.

3n+1 input and output coefficients are required for a complete identification. These coefficients need not accurately represent the respective functions, i.e., it is not necessary to take enough coefficients to get a good approximation. Provided they can be determined exactly, then the system parameters can be determined exactly even with only this partial information concerning the input and output. If some of the parameters are already known, or zero, then fewer equations are required in (7.26) and it is necessary to determine fewer coefficients.

As an example consider a second order system of the form

$$\frac{d^2 f(t)}{dt^2} + a\frac{df(t)}{dt} + b = cg(t).$$

There are only three unknown parameters requiring three equations for their determination. The matrix equation (7.27) is

$$\left\|\begin{matrix} \beta_0 & \beta_1 & \beta_2 & -\alpha_0 & -\alpha_1 & -\alpha_2 \\ \beta_1 & \beta_2 & \beta_3 & -\alpha_1 & -\alpha_2 & -\alpha_3 \\ \beta_2 & \beta_3 & \beta_4 & -\alpha_2 & -\alpha_3 & -\alpha_4 \end{matrix}\right\| \left\|\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 \end{matrix}\right\| \begin{bmatrix} b \\ a \\ 1 \\ c \\ 0 \\ 0 \end{bmatrix} = 0$$

Expanding this product

$$(\beta_0 - 2\beta_1 + \beta_2)b + (\beta_1 - \beta_2)a + \beta_2 - (\alpha_0 - 2\alpha_1 + \alpha_2)c = 0$$
$$(\beta_1 - 2\beta_2 + \beta_3)b + (\beta_2 - \beta_3)a + \beta_3 - (\alpha_1 - 2\alpha_2 + \alpha_3)c = 0$$
$$(\beta_2 - 2\beta_3 + \beta_4)b + (\beta_3 - \beta_4)a + \beta_4 - (\alpha_2 - 2\alpha_3 + \alpha_4)c = 0$$

The constant terms in the equations are $\beta_2, \beta_3, \beta_4$ respectively. Knowing the first five input and output spectral coefficients these equations may be solved for the system parameters a,b and c.

Equation (7.27) may be solved in a slightly easier manner than performing the expansion as above and finding the solution directly. This method is discussed in detail following the introduction of the discrete Laguerre functions and their use in identifying discrete systems. In (7.27) $\underline{c}$ represents the system; $C(t)$ a matrix characteristic of the input and output, and Δ a matrix significant of a property of the particular expansion functions used. Actually, if some other set of polynomial expansion functions had been employed similar equations would have been obtained.

Another such expansion was performed using the Legendre polynomials. At first it may seem surprising that these polynomials should yield a valid result as they are defined only over a finite range. This would seem to imply that the initial state of the system, at the beginning of the interval, had nothing to do with the output spectral coefficients. However, remembering that in all calculations, an infinite expansion has been assumed, then at any point in the interval the function and all its derivatives must, in the limit, be approximated exactly. Thus, if a function is continuous, by Taylor's theorem the analytic continuation outside the interval must converge also. Therefore the initial state of the system is automatically absorbed in the representation of the input spectrum.

The main difficulty remaining is the evaluation of these spectra. For on-line purposes the spectra must be continuously updated as the process evolves, implying a great deal of calculation. How this can be done with a minimum of effort is demonstrated in the next section.

7.3 Continuous Updating of Spectra

Looking again at the figure on page 143, assume that the spectral coefficients at time t are known. At some later time t+Δt it will be necessary to re-evaluate these coefficients if equation (7.27) is to be used to detect variations in the parameters. The functions f(t) and g(t) may be stored and the integrals (7.13) and (7.15) evaluated again, although this seems a great waste as the coefficients at t have already been calculated and it seems evident that the information contained in these may be employed in the calculation of the required value at t+Δt.

This is indeed the case. Taking equation (7.13), at time t+Δt

$$\alpha_i(t+\Delta t) = \int_{-\infty}^{0} e^{\tau} f(t+\Delta t+\tau)\Lambda_i(\tau)\,d\tau$$

$$= \int_{-\infty}^{-\Delta t} e^{\tau} f(t+\Delta t+\tau)\Lambda_i(\tau)\,d\tau + \int_{-\Delta t}^{0} e^{\tau} f(t+\Delta t+\tau)\Lambda_i(\tau)\,d\tau \tag{7.28}$$

The first integral in this expression can be changed to

$$\int_{-\infty}^{0} e^{x-\Delta t}\, f(t+x)\,\Lambda_i(x-\Delta t)\,dx \tag{7.29}$$

by the change in variables $\tau+\Delta t = x$. The functions $\Lambda_i(x-\Delta t)$ can be expanded in terms of $\Lambda_j(x)$ as shown below

$$\Lambda_i(x-\Delta t) = \sum_{j=0}^{\infty} \gamma_{ij}(\Delta t)\,\Lambda_j(x) \tag{7.30}$$

where

$$\gamma_{ij}(\Delta t) = \int_{-\infty}^{0} e^{x}\Lambda_i(x-\Delta t)\,\Lambda_j(x)\,dx. \tag{7.31}$$

$\Lambda_j(x)$ must be orthogonal to all powers of x, x^k, for $k < j$, otherwise it could not be orthogonal to all the functions $\Lambda_k(x)$ for $k < j$. As

$\Lambda_i(x-\Delta t)$ is an ith order polynomial in x, $\Lambda_j(x)$ must be orthogonal to it for $j > i$ so that $\gamma_{ij}(\Delta t) = 0$ for $j > i$. Hence the summation in equation (7.30) need only be carried to i. Substituting this result back into (7.29) produces

$$e^{-\Delta t} \sum_{j=0}^{i} \gamma_{ij}(\Delta t) \int_{-\infty}^{0} e^{x} f(t+x) \Lambda_j(x)\, dx$$

$$= e^{-\Delta t} \sum_{j=0}^{i} \gamma_{ij}(\Delta t)\, \alpha_i(t). \tag{7.32}$$

As shown in the previous section, it is only necessary to carry some finite numbers 's' of coefficients in order to calculate the parameters of the system. These coefficients may be formed into two s-vectors $\underline{\alpha}(t)$ and $\underline{\beta}(t)$, and the constants $\gamma_{ij}(\Delta t)$ represented by an sxs matrix $\gamma(\Delta t)$. The second integral of (7.28) is the increment to the coefficients caused by including the portion of the function $f(\tau)$ between t and $t+\Delta t$. Forming the functions $\Lambda_i(\tau)$ into an s-vector $\underline{\Lambda}(\tau)$ equation (7.28) becomes

$$\underline{\alpha}(t+\Delta t) = e^{-\Delta t}\, \gamma(\Delta t)\, \underline{\alpha}(t) + \int_{-\Delta t}^{0} e^{\tau} f(t+\Delta t+\tau)\, \underline{\Lambda}(\tau)\, d\tau \tag{7.33}$$

This is the updating formula desired. Only the integration over the range Δt need be performed to get the new coefficients, the effect of the function previous to this interval being contained in the matrix operation $e^{-\Delta t}\gamma(\Delta t)$. For four coefficients the matrix $\gamma(\Delta t)$ is

$$\gamma(\Delta t) = \left\Vert \begin{array}{cccc} 1 & 0 & 0 & 0 \\ -\Delta t & 1 & 0 & 0 \\ \frac{\Delta t(\Delta t-2)}{2} & -\Delta t & 1 & 0 \\ \frac{-\Delta t(-\Delta t^2+6\Delta t-6)}{6} & \frac{\Delta t(\Delta t-2)}{2} & -\Delta t & 1 \end{array} \right\Vert$$

From the appearance of this matrix there would seem to be much more symmetry in the $\gamma_{ij}(\Delta t)$ then at first anticipated. Actually, with some difficulty it can be shown that $\gamma_{ij}(\Delta t) = \gamma_{i-1,j-1}(\Delta t)$. A similar result is proved in Appendix I for the discrete Laguerre functions and so the verification for the continuous case is omitted.

Similar updating formulas can be found for Legendre and Tchebychev polynomial expansions. However, as these are not convenient for purposes of system identification their development is left.

Equation (7.33) still involves very complicated operations due to the presence of the integral. As the final solution of equation (7.27) will probably have to be performed digitally, it would seem more logical to approach the problem directly using discrete methods, rather than trying to convert the integral into a discrete form.

7.4 System Identification using the Discrete Laguerre Polynomials

A discrete linear difference system[51,57] is described by a difference equation of the form

$$\sum_{i=0}^{p} a_i\ g(n-i) = \sum_{i=0}^{p} b_i\ f(n-i) \tag{7.34}$$

f(n) is the input to the system and g(n) the output, p being the order of the system. Usually the lower index on the right is something greater than 0 but this can be reduced to the above expression by introducing enough new constants b_i, all equal to zero. $b_0 \neq 0$ implies that the input can effect the output in zero time.

If the input and the output waveforms at time K are expanded in discrete Laguerre functions (see Appendix I, section 2) with spectral coefficients $\underline{\beta}(K)$ and $\underline{\gamma}(K)$ then the system coefficients satisfy the set of linear equations

$$B(K)\ Q_p\underline{a} = G(K)\ Q_p\underline{b} \tag{7.35}$$

where B(K) and G(K) are matrices composed of the spectral coefficients and have the same form as that shown for the Laguerre spectra on page 147. Q_p is a square matrix derived in the Appendix.

In choosing the spectra there is still one degree of freedom at our disposal; this is the choice of the weighting constant α. This constant may be assigned any value between 0 and 1. Returning to the definition of the discrete Laguerre polynomials on page 43, for $\alpha = 0$, $L_i(n)$ becomes

$$L_i(n) = \binom{n}{i}.$$

The normalized functions given in equation (2.18) have only one non-zero value corresponding to $n = i$

$$l_i(n) = 0 \qquad n \neq i$$
$$l_i(i) = L_i(i) = 1.$$

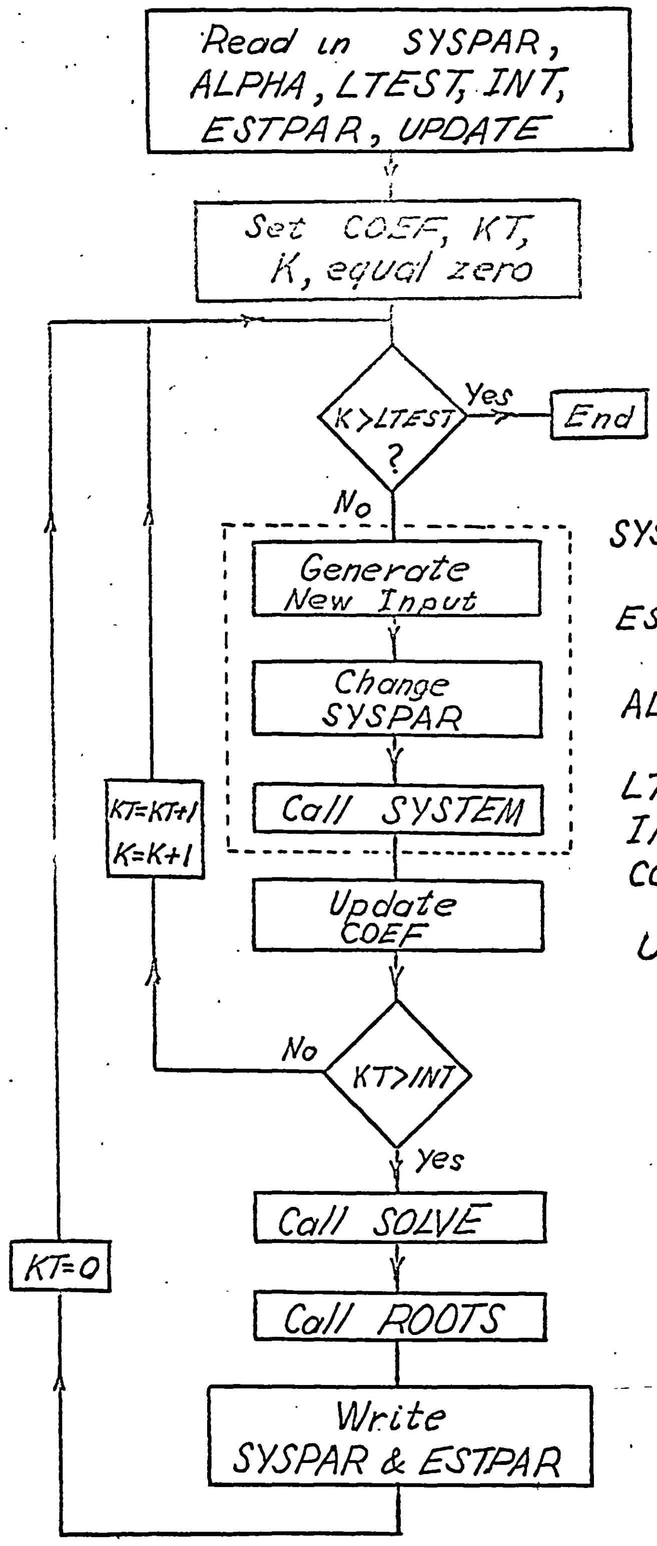
Therefore this limiting case yields the complete set of single unit pulses. The spectral coefficients $\beta_i(K)$ and $\gamma_i(K)$ become g(K-i) and f(K-i) respectively. The equation (7.35) then corresponds to just taking the last 2n+1 linear equations defined by (7.34) and using these to calculate the coefficients. This is obviously correct but is usually a very ill-advised approach. For one thing, if the sampling rate is high the successive f(n)'s and g(n)'s are almost the same, meaning that these linear equations are very nearly inconsistent and hence great difficulties will be encountered in trying to get a reliable solution.

Also the result is highly sensitive to noise on the input or output. The one advantage seems to be that as only the most up-to-date information is used, the most recent estimate of the parameters is obtained.

As α is increased, more and more information from the past is incorporated into the spectral coefficients $\underline{\beta}(K)$ and $\underline{\gamma}(K)$. One can expect the coefficients to become more uncorrelated and hence the conditioning of equations (7.35) to be improved. This implies that the solution for the coefficients is more reliable. Also, the polynomials act as filters to noise and so one might expect some noise immunity using a larger α. However, the response of the estimator is slower as it weights information further into the past in the calculations. Thus, for rapid changes in parameters, little confidence can be placed in the result obtained.

7.5 Computer Tests

Extensive computer tests were performed to verify the above hypotheses. The flow diagram describing the general outline of the program is given in figure (7.1). The spectral coefficients were evaluated by the continuous updating procedure described in the third section of Appendix I. The two main subroutines involved with the identification procedure are SOLVE, which finds the solution for the system coefficients $\underline{a}$ and $\underline{b}$ from equation (7.35), and ROOTS, which calculates the pole-zero diagram in the z-plane[57] corresponding to the system estimated by SOLVE. A detailed description of these two subroutines along with flow diagrams is given in Appendix III. The saving in computation that can be effected due to the special nature of the matrices B(K),



SYSPAR =	System Parameters
ESTPAR =	Estimated Parameters
ALPHA =	Polynomial Constant
LTEST =	Length of Test
INT =	Calculation Interval
COEF =	Spectral Coefficients
UPDATE =	Vectors for updating Coefficients

Portion within dotted lines not part of the identification routine

Flow Diagram for the Indentification of Systems Using Discrete Polynomials

Figure 7.1

G(K) and Q_p of equation (7.35) is also described. It was thought necessary to extract the roots as these are more truly representative of the system behaviour than the difference equation coefficients. Also they are less sensitive to computational errors than are the coefficients, that is the coefficients may appear to have changed substantially whereas the pole-zero pattern has remained almost stationary. All the runs performed using this identification scheme took under one minute of 7090 computing time, at least half of which was absorbed in compilation.

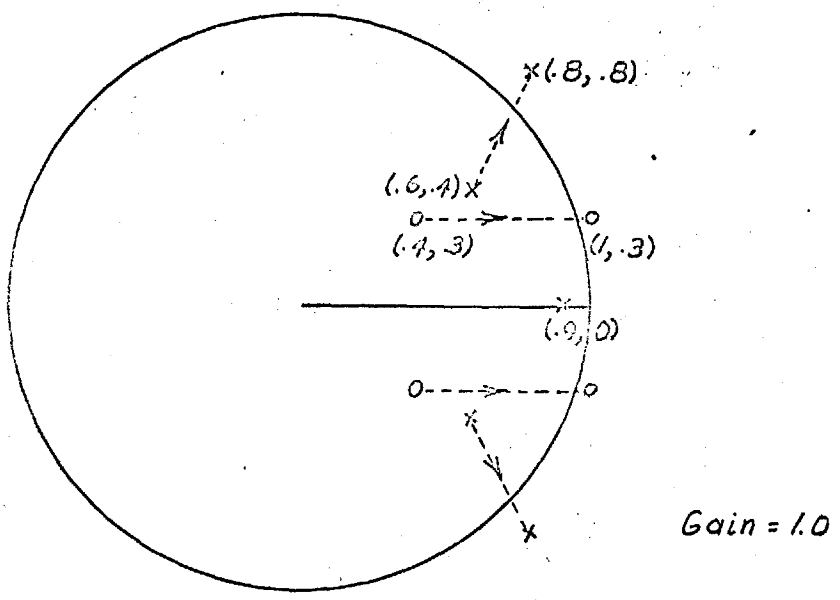
The first test performed was to check that for varying α the time for identification varied. A second order system was set up and square wave excitation applied. Half way through the test one of the system coefficients was given a step change. As expected for small α this change was detected in a matter of a few intervals, whereas for large α it took as much as 10 to 20 intervals to settle on the correct values.

Following this tests with slowly varying 2nd, 3rd and 4th order systems were performed for two different values of α(0.2 and 0.8) to determine what effect α had on the detection efficiency. The result for the third order system is shown below. For the other two systems, similar performance was observed, the 2nd order giving a more reliable estimate than the 4th order as would be expected.

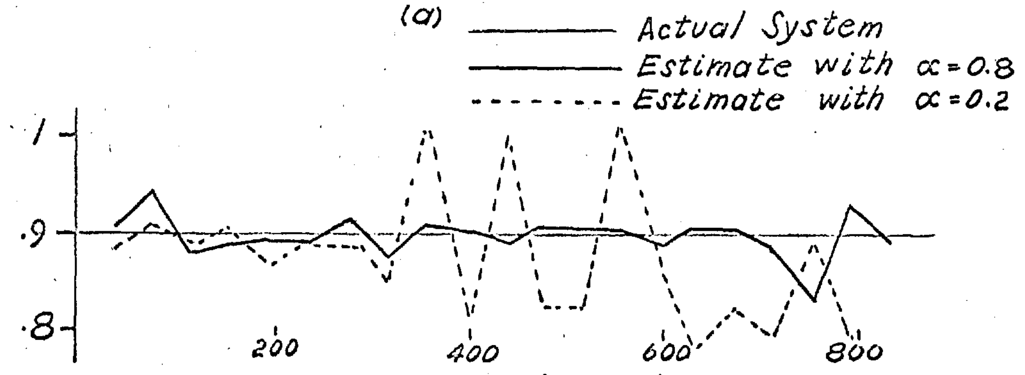
The third order system considered has the pole-zero pattern of figure (7.2^a), the gain being set to 1.0. The test was carried over a thousand intervals with an estimate of the complete pole-zero pattern being calculated every forty intervals. Two poles and two zeros were varied linearly in the manner shown. This particular configuration represents a severe test on the detection procedure, not only because the dominant pole at (0.9) tends to mask the other poles, at least for the first portion of the test, but also as the pole variations take them into the unstable region outside the unit circle. The poles cross the unit circle in the 685th interval; however, the estimator still gives results right up until the 840th interval when the instability causes the whole system to blow up and the equations (7.35) become inconsistent. The input was a square wave varying between ± 1 with 10% gaussian noise superimposed.

The remainder of figure (7.2), and figure (7.3) show the actual pole-zero variations along with the pole-zero pattern as estimated by solving equations (7.35) and then finding the roots of the characteristic equation produced. The scales, for comparison purposes, are the same on all graphs except for the gain (defined as the normalizing constant for the b_i's) which is displayed on the larger scale shown. The long estimator ($\alpha = 0.8$) is illustrated by the heavy line, the short estimator being represented by the dotted lines.

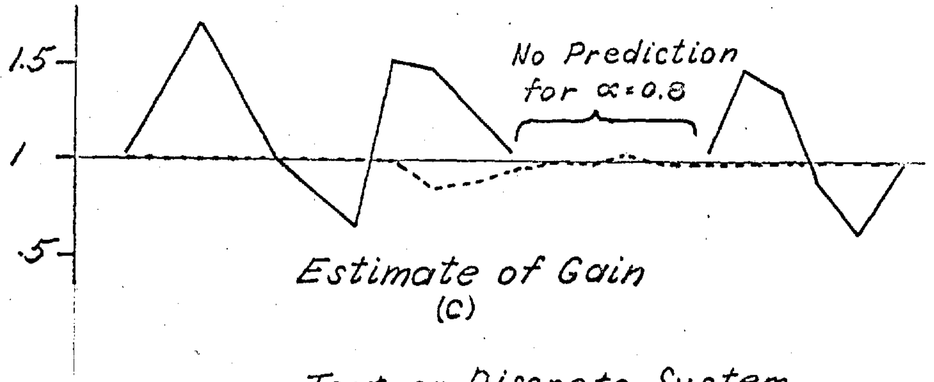
Considering the z-plane as a mapping of the p-plane of continuous analysis via the function $z=e^{Tp}$ where T is the sampling rate[57], then the unit circle corresponds to the portion of the frequency axis between $-\frac{\pi}{T} < \omega < \frac{\pi}{T}$. The axis (0,1) corresponds to the negative real axis of the p-plane. It may easily be demonstrated that radial lines in the z-plane represent lines of constant frequency or horizontal lines in the p-plane, and circles represent lines of constant attenuation or vertical lines. Therefore, as the angle with respect to the origin of a complex pole increases the frequency of the corresponding pole in the



Pole-Zero Diagram of
System under Test
(a)

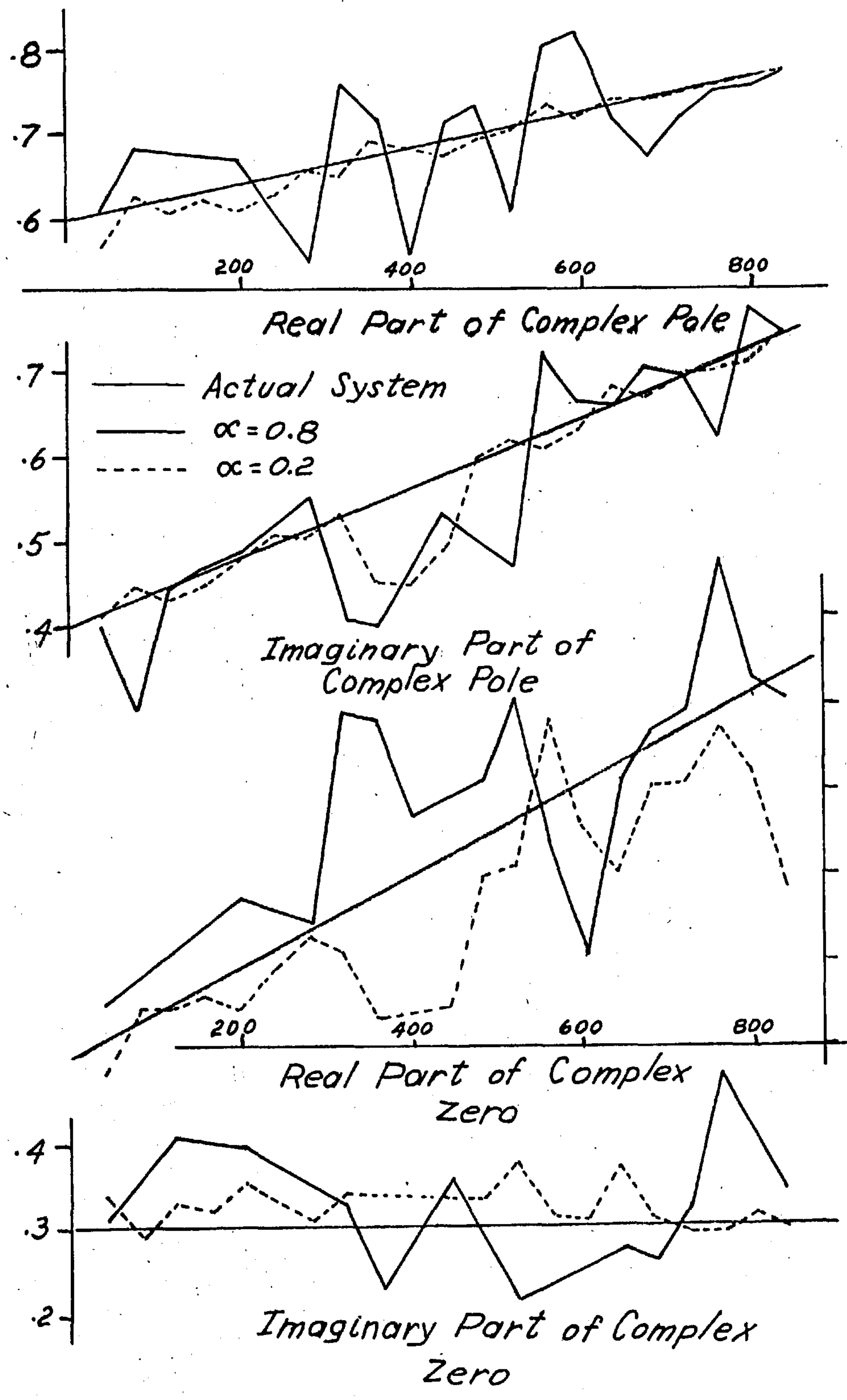


Estimate of Real Pole
(b)



Estimate of Gain
(c)

Test on Discrete System
Identification Procedure
Figure 7.2



Test on the Identification Procedure Using Discrete Polynomials

Figure 7.3

p-plane also increases.

The above paragraph yields a very nice interpretation of the results of figures (7.2) and (7.3). The short estimator corresponds to a high frequency estimator whereas the long one corresponds to a lower frequency. For the long time constant (.9) $\alpha = 0.8$ gives a very good estimate, whereas $\alpha = 0.2$ produces widely varying results. For the complex poles $\alpha = 0.2$ gives a very much better result as this pole represents a high frequency mode, whereas $\alpha = 0.8$, although appearing to have the correct trend, yields quite widely varying estimates. For the zeros neither of them give particularly impressive results, although again the short determination seems better, especially so for determining the gain.

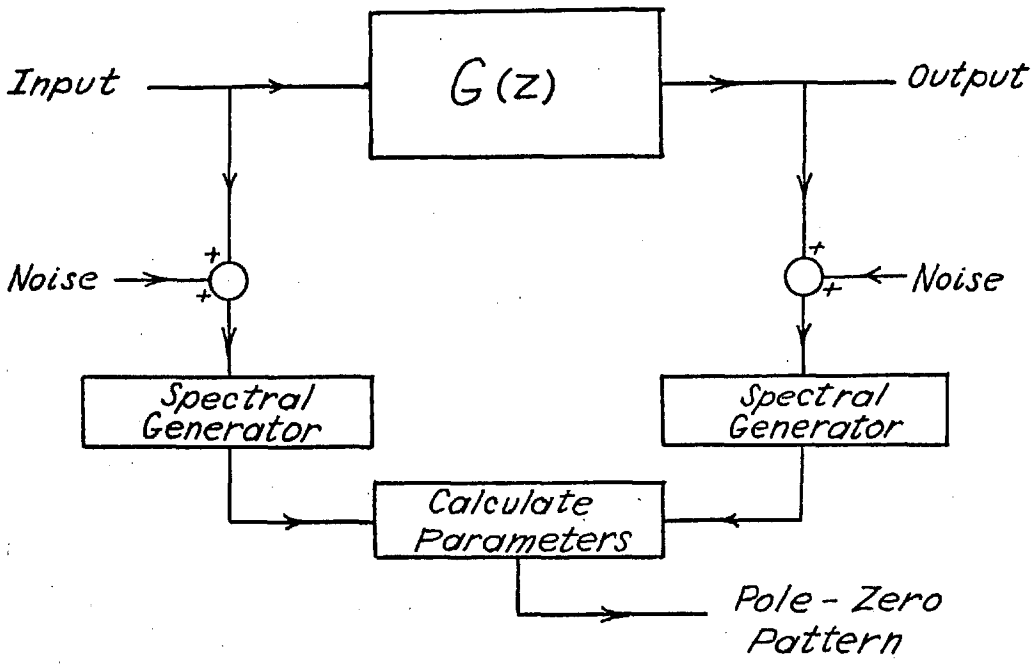
The same kind of behaviour was noticed in all the other tests performed, indicating that it may be worthwhile to have two estimators going at the same time, one for obtaining the low frequency poles, the other for the high frequency components. For rapidly sampled continuous systems the poles tend to congregate around the point (1,0) indicating that for these, the long time constant determination should be useful. Again it is emphasized that the above set of results was compiled and generated in less than one minute of computer time.

A constant sixth order system having six poles and five zeros was also tested, and, surprisingly enough, the estimates were remarkably close to the actual values. Different inputs such as ramps, step, white noise etc., were tried on other varying systems with almost the same degree of success as above. As long as the input has a fairly high harmonic content reasonable estimates are obtained.

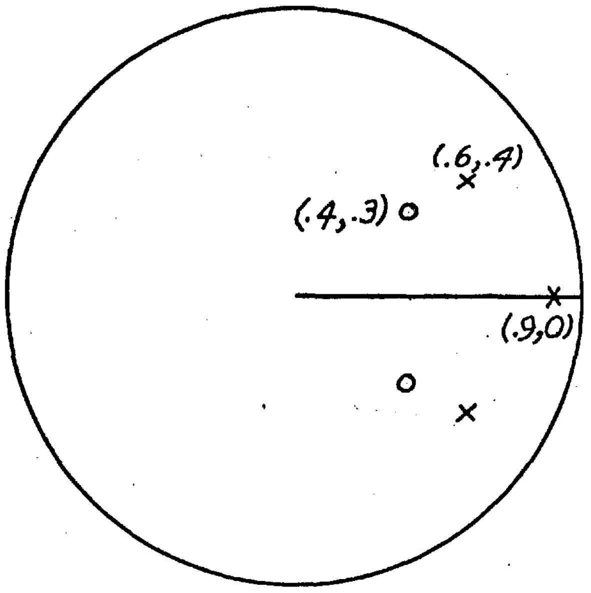
Tests using higher and lower order estimators than the order of the actual system were also done. On fitting a fourth order estimator to a third order system the estimator predicted the correct pattern, plus a nearly overlapping pole-zero pair. On using an estimator of order less than the actual system, the estimated pattern varied widely from step to step.

The last test performed was to see how the estimating procedure reacted to the presence of noise on the input and output. Independent gaussian noise sources were incorporated into the system as shown in figure (7.4^a). G(z) represents the system, the particular configuration being shown in (7.4^b). It is quite a problem to know exactly how best to present the results for this test considering the amount of data accumulated. A complete set of pole-zero estimates for a third order system was obtained for the two values of α above and for noise values of 0,1,2,3,5,10,50 percent of the signal level. The test in each case was run for a 160 intervals and the pole-zero pattern estimated every 20 intervals, giving only eight estimates for each singularity. This does not really represent enough data points for decent statistics but was thought sufficient to show up any trends. The applied signal was a square wave with a transient inserted at the 80th interval.

As expected for very low noise levels, the shorter estimating polynomials gave better results, but as the noise level increased above 1% these estimates degenerated into nonsense. Using $\alpha = 0.8$, although the initial estimates had a wider variance, this variance increased at a lot slower rate than for the first case. For both constants after the 3% noise level was reached the estimates of the complex poles and



System Configuration for Noise Tests
(a)



Pole-Zero Pattern for Discrete System Under Test
(b)

Figure 7.4

zeros were very bad. This arises due to their position quite far behind the dominating pole and partially cancelling each other. The estimates for the dominating pole in both cases were more consistent.

Figure (7.5) illustrates the results for this last estimate. (7.5^a) shows the mean value of the estimate of this pole using the two different constants α and (7.5^b) gives the standard deviation. For 50% noise the values obtained using $\alpha = 0.2$ were completely random and not really worth recording. As evidenced from the irregularities of these curves the statistics are hardly sufficient to draw any sound conclusions. However, the diagram does demonstrate with a fairly high probability that for noisy systems there is a distinct advantage to using the polynomials that have a longer reach into the past.

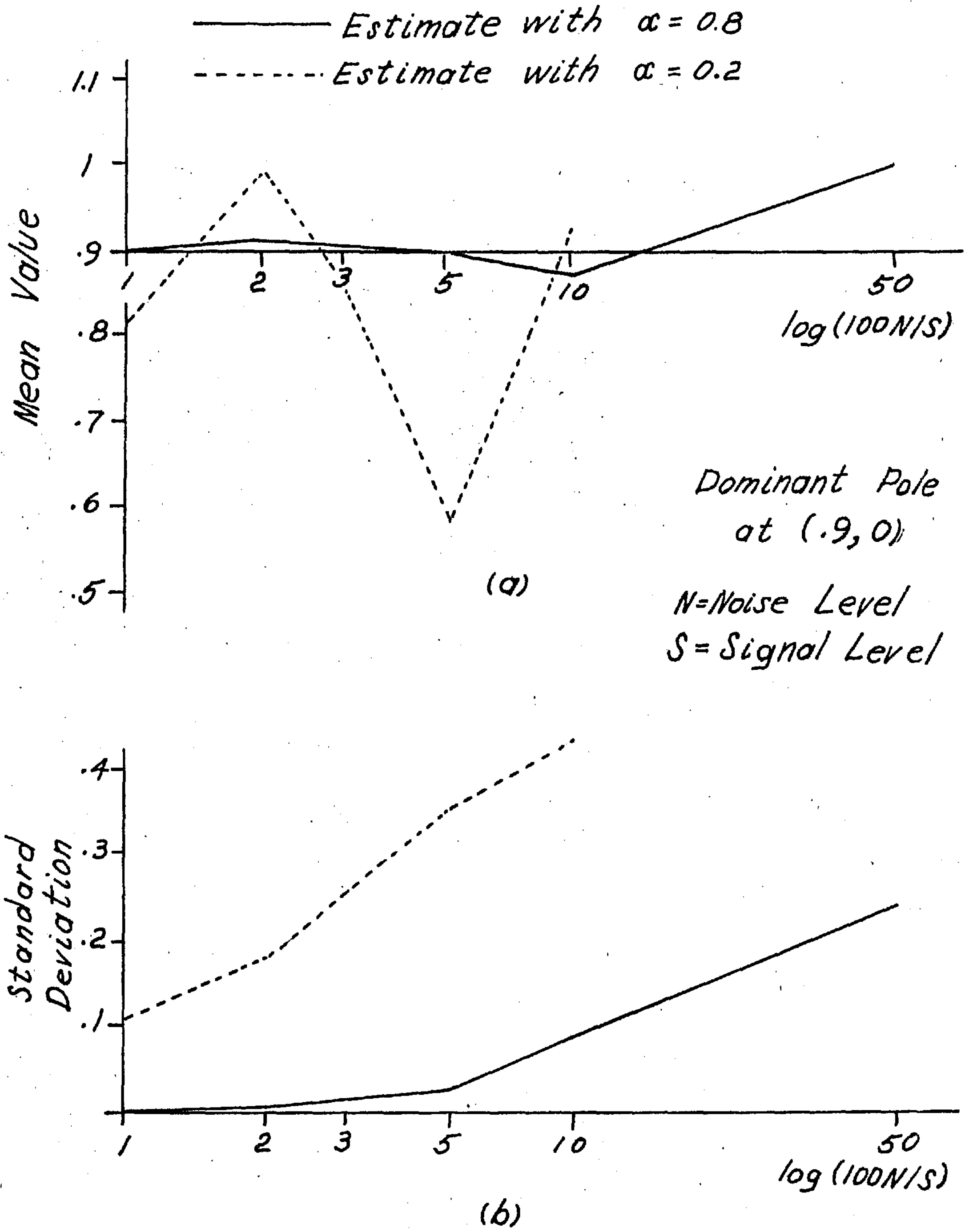
The use of the discrete Laguerre polynomials for the identification of single input single output discrete systems has now been discussed. It remains to develop a method of identification for multidimensional systems.

7.6 Discrete Laguerre Polynomials and Multi-dimensional Systems

In the multi-dimensional case linear discrete systems may be represented by the matrix equation

$$\underline{x}_k = A\underline{x}_{k-1} + B\underline{u}_k \tag{7.36}$$

where $\underline{x}_k$ is an r-dimensional state vector, $\underline{u}_k$ an s dimensional input vector, A the rxr state transition matrix, and B the rxs input transformation matrix. The problem is, given $\underline{x}_k$ and $\underline{u}_k$ and all their past history, to determine the matrices A and B. Equation (7.36) implies the



Variations in the Estimates for the Dominant Pole of a Third Order System

Figure 7.5

infinite set of equations

$$\underline{x}_{k-n} = A\underline{x}_{k-(n+1)} + B\underline{u}_{k-n} \qquad n=0,1,\ldots.. \tag{7.37}$$

The vectors $\underline{x}_{k-n}$ and $\underline{u}_{k-n}$ are expanded in discrete Laguerre polynomials

$$\underline{x}_{k-n} = \sum_{i=0}^{\infty} \underline{\gamma}_i(k)\, l_i(n)$$

$$\underline{u}_{k-n} = \sum_{i=0}^{\infty} \underline{\beta}_i(k)\, l_i(n)$$

in the same fashion as equations (A.1.23) in Appendix I. The coefficient vectors $\underline{\gamma}_i$ and $\underline{\beta}_i$ are defined by equations analogous to (A.1.22). Substituting these values in (7.37) yields the relation

$$\sum_{i=0}^{\infty} \underline{\gamma}_i(k)\, l_i(n) = A \sum_{i=0}^{\infty} \underline{\gamma}_i(k)\, l_i(n+1) + B \sum_{i=0}^{\infty} \underline{\beta}_i(k)\, l_i(n). \tag{7.38}$$

Using equation (A.1.34) of Appendix I, $l_i(n+1)$ can be expanded in terms of $l_j(n)$ for $j \leqslant i$. Making this substitution, interchanging the summation over i and j in the first term on the right, and then equating coefficients of $l_i(n)$ on both sides produces the set of relationships

$$\underline{\gamma}_i(k) = A \sum_{j=i}^{\infty} \underline{\gamma}_j(k)\, \mu_{j,1}(i) + B\underline{\beta}_i(k) \qquad i=0,1,2,\ldots.. \tag{7.39}$$

Taking the sum of the i th equation and $\alpha^{\frac{1}{2}}$ times the (i+1) th equation reduces this to the set of equations

$$\underline{\gamma}_i(k) + \alpha^{\frac{1}{2}}\underline{\gamma}_{i+1}(k) = A(\alpha^{\frac{1}{2}}\underline{\gamma}_i(k) + \underline{\gamma}_{i+1}(k)) + B(\underline{\beta}_i(k) + \alpha^{\frac{1}{2}}\underline{\beta}_{i+1}(k)) \tag{7.40}$$

an infinite set of vector equations with a finite number of terms in each.

Taking the first r+s of these equations and forming the corresponding matrix equation

$$S(k) = AT(k) + BQ(k)$$

where the matrix $S(k)$ has its columns made up of terms like $\underline{\gamma}_i(k) + \alpha^{\frac{1}{2}}\underline{\gamma}_{i+1}(k)$, $T(k)$ of columns like $\alpha^{\frac{1}{2}}\underline{\gamma}_i(k) + \underline{\gamma}_{i+1}(k)$ and similarly for $Q(k)$. $S(k)$ and $T(k)$ are $r\text{x}(r+s)$ matrices while $Q(k)$ has the dimensions $s\text{x}(r+s)$. Taking the transpose of both sides and forming the partitioned product yields

$$S^T(k) = T^T(k)\,A^T + Q^T(k)\,B^T = \left\| T^T(k) \;\vdots\; Q^T(k) \right\| \left\| \begin{array}{c} A^T \\ \hdashline B^T \end{array} \right\| .$$

The partitioned matrix $\left\| T^T(k) \;\vdots\; Q^T(k) \right\|$ has dimensions $(r+s)\text{x}r + (r+s)\text{x}s = (r+s)\text{x}(r+s)$. Provided this matrix is non-singular it has an inverse and the solution for A and B becomes

$$\left\| A \;\vdots\; B \right\|^T = \left\| T^T(k) \;\vdots\; Q^T(k) \right\|^{-1} S^T(k).$$

Taking the transpose of both sides and noting that the inverse and transpose commute the final solution for A and B is

$$\left\| A \;\vdots\; B \right\| = S(k) \left\| \begin{array}{c} T(k) \\ \hdashline Q(k) \end{array} \right\|^{-1} \tag{7.46}$$

The matrices S, T and Q depend only on the first r+s+1 spectral coefficients obtained for the various inputs and outputs. Thus, using a finite number of coefficients it is theoretically possible to obtain the system parameters, represented by A and B exactly, although the inversion of $\left\| \begin{array}{c} T(k) \\ \hdashline Q(k) \end{array} \right\|$ may prove to be a difficult problem. The coeff-icients can be continuously updated as shown for the one dimensional case, making it possible to detect changes in the system matrices as the process evolves.

CHAPTER VIII

Conclusions

The original contributions presented in this thesis fall into four main catagories corresponding to:

(1) Universal nets for realizing boolean functions.

(2) Actual construction of a digital function generator.

(3) Complete development of the discrete Laguerre polynomials.

(4) Identification of system parameters using truncated spectra.

Some secondary results, such as Theorem 2.1 and the introduction of the linear segment functions, were included but were not developed far enough to warrant further discussion here.

(1) To my knowledge this approach to realizing boolean functions by logical elements has not been attempted before. In the past the problem of finding a best circuit realization for a given function has been greatly complicated by the large number of possible configurations that the elements may assume. This has made the search problem very large and intricate and no powerful analytic methods for reducing this search have yet been developed. By constricting the geometry of the networks into the form of a universal net, although it may introduce more elements than the absolute minimum, at least gives a straightforward method for finding circuit realizations for any boolean function, with the assurance that none of the gates will be overloaded. Also, absolute minimal realization in this geometry may be obtained. It was

demonstrated that this method has at least the same order of elements (Lupanov's theorem) in the final network as some hypothetical best method of design would have.

(2) This approach to obtaining digital functions, other than by performing analytical operations on a computer, has not yet been followed through in the literature. The scheme presented may, at first sight, appear to be rather 'ad hoc' but does represent an attempt at eliminating the need for a digital computer in digital control systems and replacing it by a much simpler device. Certainly, in the example described in detail in Chapter VI, it would be difficult to find a network with the same capabilities that could be obtained more economically. This section would have profited greatly by an actual system being built and adaptation tests performed on it. A beginning on this was actually made but unfortunately time started to run out, not allowing this particular aspect of the project to be completed.

(3) The development of these polynomials was, perhaps, carried further than justified by their subsequent use in the thesis. However, as so little work seems to be extent on discrete polynomials and they appear to be almost a natural offspring of digital computer analysis, it was thought that a complete summary of all their properties would be useful. The difference equations, recurrence relationships, generating functions etc., have the same sort of appearance as the equivalent expressions obtained for orthogonal continuous polynomials. The fact that these new polynomials are basically different than the continuous Laguerre functions can be proven by comparing the forms of the D matrices (page 146) and the Q matrices (page 187) used in the identification of

systems by the continuous and discrete polynomials respectively. You would expect these to have the same form if there was some close relation between the two polynomials. The scope for future use of these functions in solving discrete difference equations and general computation problems seemsunlimited.

(4) System identification using polynomial expansions[54] has been tried before only with limited success. Then, however, the approach has always been to expand the actual system function (the impulse response $h(t)$ or the frequency function $G(\omega)/F(\omega)$ etc.) in terms of a set of polynomials and then state that, in the limit, this set must converge on the actual system. Here lies the difficulty, in that a great many expansion terms are required before a satisfactory representation is obtained. The scheme proposed above requires only a finite number of input and output coefficients for a complete identification. Thus, although at first this method may appear rather complex, the final closed form obtained justifies its use. No actual computer tests using the continuous functions of section (7.2) were performed, although the validity of the final expression was checked by some hand calculations. The discrete functional representation was thought more useful in that it adapts itself immediately to the digital computer without having to incorporate any integration subroutines. At any rate, as continuous systems can be approximated arbitrarily closely by discrete systems, provided the sampling rate is high enough, further investigation seemed unnecessary. It becomes only a question of where the approximations should be made, whether in the integration subroutine or by using a sampled model.

8.1 Possible Extensions

There is large scope for further work in the universal network studies begun in Chapter IV, especially when using elements with more than three inputs and with more than a fan-out of one. The great advantage of having more inputs on a gate is not so much the time saving that can be effected, but rather the far greater number of configurations in which these elements may be connected. Allowing a fan-out of more than one allows the same freedom. As stated previously, this larger number of configurations (two of which were illustrated for the three input nor gates) makes some kind of organized search necessary. It may be possible to design universal nets having a fan-out of two that subsume the nets described above. This may lead to a much greater saving in the number of elements used to obtain boolean functions. It is my own intention to pursue this particular line further, as well as try and find some analytic way for getting around the need to employ the Quine method to minimize the function before the best net may be obtained.

Nothing more remains to be done on adaptive digital function generators except to build one and apply it to a system; either a great or small task. How these may be used in the context of a general learning system deserves further study and would not be at variance with the present-day trend of control theory. Also how the modern developments in automata theory can be tied up with these discrete function generators should prove a fruitful field of investigation.

A great deal more work on the statistical properties of the discrete Laguerre polynomials remains. This section was rather hurried over and no real thought was applied to what kind of systems are actually encountered or how noise effects the system behaviour. It is evident from the results that were obtained that a great improvement in the estimates could be made if some sort of additional filtering was performed on the first estimates. A search into the exact requirements for a special purpose computer to perform the calculations needed for the identification procedure should result in fairly economical apparatus. The discussion in Chapter VII has shown that this method of system identification is possible. It remains to be shown that it is economically viable and stands up to comparisons with some of the known methods, such as model references and correlation techniques, now in use.

APPENDIX I

Discrete Laguerre Functions

A.1.1 Basic Properties

Before starting, a few elementary results from the calculus of finite differences will be necessary. Given a discrete function f(n), defined on the set of integers, the forward difference operator Δ is defined by

$$\Delta f(n) = f(n+1) - f(n).$$

Obviously, it is a linear operator implying

$$\Delta(af(n) + bg(n)) = a\Delta f(n) + b\Delta g(n).$$

If $\quad \Delta f(n) = h(n)$

then
$$\sum_a^b h(n) = \sum_a^b (f(n+1) - f(n)) = f(b+1) - f(a). \quad (A.1.1)$$

Finally a rather surprising identity of the form

$$f(n+1)\ \Delta(g(n)\ \Delta h(n)) - h(n+1)\ \Delta(g(n)\ \Delta f(n)) = \Delta(g(n) \left[f(n)\ \Delta h(n) - h(n)\ \Delta f(n) \right]). \quad (A.1.2)$$

This relationship may be checked merely by carrying out all the expansions implied by the differences.

Consider the generating function

$$\frac{(1+t)^n}{(1+\alpha t)^{n+1}} = \sum_{k=0}^{n} \binom{n}{k} t^k \sum_{i=0}^{\infty} \binom{-(n+1)}{i} (\alpha t)^i. \quad (A.1.3)$$

For $|\alpha| < 1$ and $|t| < 1$ the second series converges. The usual notation $\binom{a}{b}$ is used for the binomial coefficient $\frac{a!}{b!(a-b)!}$.

$$\binom{-(n+1)}{i} = \frac{(-(n+1))\ (-(n+1)-1)\ \ldots\ldots\ (-(n+1)\ -\ i+1)}{i!}$$

$$= \frac{(-1)^i\ ((n+1)\ (n+2)\ \ldots\ldots\ (n+i))}{i!}$$

$$= (-1)^i \binom{n+i}{i}.$$

Hence equation (A.1.3) is equivalent to

$$\sum_{i=0}^{\infty} \sum_{k=0}^{n} (-1)^i \binom{n}{k} \binom{n+i}{i} \alpha^i t^{k+i} .$$

Which, on gathering powers of t is

$$\sum_{i=0}^{\infty} \Big(\sum_{j=0}^{i} (-1)^{i-j} \binom{n}{j} \binom{n+i-j}{i-j} \alpha^{i-j}\Big) t^i.$$

Substituting i-j=k and using the identity

$$\binom{n}{i-k} \binom{n+k}{k} = \binom{i}{k} \binom{n+k}{i}$$

The expansion becomes

$$\sum_{i=0}^{\infty} \Big(\sum_{k=0}^{i} (-1)^k \binom{i}{k} \binom{n+k}{i} \alpha^k\Big)\ t^i = \sum_{i=0}^{\infty} L_i(n) t^i$$

where the discrete polynomials $L_i(n)$ have been defined as

$$L_i(n) = \sum_{k=0}^{i} (-1)^k \binom{i}{k} \binom{n+k}{i} \alpha^k. \qquad \text{(A.1.4)}$$

Differentiating the generating function with respect to t yields

$$\frac{(1+t)^n}{(1+\alpha t)^{n+1}} \left(\frac{(n-\alpha(n+1)-\alpha t)}{(1+t)(1+\alpha t)}\right) = \sum_{i=0}^{\infty} i\ L_i(n) t^{i-1}$$

or

$$\sum_{i=0}^{\infty} L_i(n) t^i\ (n-\alpha(n+1)-\alpha t) = \sum_{i=0}^{\infty} i\ L_i(n)\ (1+(1+\alpha)t+\alpha t^2) t^{i-1}.$$

Defining $L_{-1}(n)=0$ and equating coefficients of t^i the recurrence relation

$$(i+1)L_{i+1}(n) + \left[i(1+\alpha) - (n-\alpha(n+1))\right] L_i(n) + i\alpha L_{i-1}(n) = 0 \tag{A.1.5}$$

results.

Taking the first difference of (A.1.3) yields

$$\frac{(1+t)^n}{(1+\alpha t)^{n+1}} \left(\frac{t-\alpha t}{1+\alpha t}\right) = \sum_{i=0}^{\infty} \Delta L_i(n) t^i .$$

Again, expanding the generating function and equating coefficients, the difference equation

$$\begin{aligned} L_i(n)(1-\alpha) &= \alpha\Delta L_i(n) + \Delta L_{i+1}(n) \\ &= \Delta(\alpha L_i(n) + L_{i+1}(n)) \end{aligned} \tag{A.1.6}$$

is obtained.

The next problem is to find the difference equation which the polynomials satisfy. Taking the first difference of the recurrence relation (A.1.5) gives

$$\begin{aligned} (i+1)\, \Delta L_{i+1}(n) + (i(1+\alpha)+\alpha)\, \Delta L_i(n) + i\alpha\, \Delta L_{i-1}(n) \\ = \Delta n(1-\alpha)\, L_i(n) \\ = (1-\alpha)(n+1)\, \Delta L_i(n) + (1-\alpha)\, L_i(n). \end{aligned} \tag{A.1.7}$$

Using equation (A.1.6)

$$(i+1)\, \Delta L_{i+1}(n) = (L_i(n)(1-\alpha) - \alpha\Delta L_i(n))\, (i+1) \tag{A.1.8}$$

and

$$i\alpha\, \Delta L_{i-1}(n) = i(L_{i-1}(n)(1-\alpha) - \Delta L_i(n)) . \tag{A.1.9}$$

Substituting (A.1.8) and (A.1.9) into (A.1.7) and simplifying

$$i(L_{i-1}(n) + L_i(n)) = (n+1)\, \Delta L_i(n) . \tag{A.1.10}$$

Taking the difference of (A.1.10)

$$i(\Delta L_{i-1}(n) + \Delta L_i(n)) = \Delta((n+1)\ \Delta L_i(n)). \qquad (A.1.11)$$

Solving for $L_{i-1}(n)$ in (A.1.10) and substituting in (A.1.9) gives an equation for $\Delta L_{i-1}(n)$ in terms of $L_i(n)$

$$i\alpha\ \Delta L_{i-1}(n) = (n+1)\ \Delta L_i(n)(1-\alpha) - iL_i(n)(1-\alpha) - i\Delta L_i(n). \qquad (A.1.12)$$

Finally multiplying (A.1.11) by α and substituting in (A.1.12) the difference equation for the function $L_i(n)$ is obtained

$$\alpha\Delta((n+1)\ \Delta L_i(n)) - (1-\alpha)(n+1-i)\ \Delta L_i(n) + (1-\alpha)\ iL_i(n) = 0 \qquad (A.1.13)$$

or $$\alpha(n+2)\ \Delta^2 L_i(n) + (\alpha-(1-\alpha)(n+1-i))\ \Delta L_i(n) + (1-\alpha)\ iL_i(n) = 0.$$

It remains to show that the functions $L_i(n)$ satisfy the orthogonality relationship

$$\sum_{i=0}^{\infty} \alpha^n L_i(n)\ L_j(n) = 0 \qquad \text{for } i \neq j.$$

Before proving this, two other results are required. First from equation (A.1.4)

$$L_i(0) = (-\alpha)^i. \qquad (A.1.14)$$

This is obvious as $\binom{k}{i} = 0$ for $k < i$.

Also from (A.1.4)

$$\Delta L_i(n) = L_i(n+1) - L_i(n) = \sum_{k=0}^{i} (-1)^k \binom{i}{k}\left[\binom{n+k+1}{i} - \binom{n+k}{i}\right] \alpha^k$$

or $$\Delta L_i(n) = \sum_{k=0}^{i} (-1)^k \binom{i}{k}\binom{n+k}{i-1}\ \alpha^k$$

Hence $$\Delta L_i(0) = (-\alpha)^{i-1} i + (-\alpha)^i\ i$$
$$= (-\alpha)^{i-1} i\ (1-\alpha) \qquad (A.1.15)$$

Noting that

$$\Delta\alpha^n f(n) = \alpha^{n+1} f(n+1) - \alpha^n f(n)$$

$$= \alpha^n(\alpha\Delta f(n) - (1-\alpha)f(n))$$

equation (A.1.13) can be simplified by multiplying it by α^n and identifying $f(n)$ in the above formula as $(n+1)\ \Delta L_i(n)$. This gives

$$\alpha^n(\alpha\Delta((n+1)\ \Delta L_i(n)) - (1-\alpha)(n+1)\ \Delta L_i(n))$$

$$+ (1-\alpha)\ i\ (L_i(n) + \Delta L_i(n))\alpha^n = 0$$

or $$\Delta\alpha^n(n+1)\ \Delta L_i(n) + (1-\alpha)\ i\ \alpha^n L_i(n+1) = 0. \qquad \text{(A.1.16)}$$

Multiplying by $L_j(n+1)$ and interchanging the indices i and j gives the two equations

$$L_j(n+1)\ \Delta\alpha^n(n+1)\ \Delta L_i(n) + (1-\alpha)\ i\ \alpha^n L_i(n+1)\ L_j(n+1) = 0$$

$$L_i(n+1)\ \Delta\alpha^n(n+1)\ \Delta L_j(n) + (1-\alpha)\ j\ \alpha^n L_i(n+1)\ L_j(n+1) = 0.$$

Glancing again at equation (A.1.2) and identifying

$$f(n) = L_j(n), \qquad h(n) = L_i(n), \qquad g(n) = \alpha^n(n+1)$$

we get on subtracting the two equations above

$$\Delta\alpha^n(n+1)(L_j(n)\ \Delta L_i(n) - L_i(n)\ \Delta L_j(n))$$

$$+ (1-\alpha)(i-j)\alpha^n\ L_i(n+1)\ L_j(n+1) = 0.$$

Summing this equation from n=0 to infinity gives by equation (A.1.1)

$$L_j(0)\ \Delta L_i(0) - L_i(0)\ \Delta L_j(0)$$

$$+ (1-\alpha)(i-j) \sum_{n=0}^{\infty} \alpha^n L_i(n+1)\ L_j(n+1) = 0.$$

The value disappears at infinity as the series $\alpha^n n^k$ converges for $\alpha < 1$ and all finite k.

Substituting the values for $L_i(0)$ and $\Delta L_i(0)$ obtained in (A.1.14) and (A.1.15) yields

$$(1-\alpha)(-\alpha)^{j+i-1}(i-j) + (1-\alpha)(i-j) \sum_{n=1}^{\infty} \alpha^{n-1} L_i(n) L_j(n) = 0$$

or $$(-\alpha)^{j+i} + \sum_{n=1}^{\infty} \alpha^n L_i(n) L_j(n) = 0 \qquad \text{for } i \neq j.$$

But $(-\alpha)^{j+i} = L_i(0) L_j(0) \alpha^0$, and the final orthogonality relation

$$\sum_{n=0}^{\infty} \alpha^n L_i(n) L_j(n) = 0 \qquad i \neq j$$

is obtained.

A little more patience will see us through this maze. The last thing to determine is the normalizing constant μ_i. Multiplying the recurrence relation (A.1.5) by $\alpha^n L_{i-1}(n)$ and summing gives

$$i \sum_{n=0}^{\infty} \alpha^{n+1} L_{i-1}^2(n) - (1-\alpha) \sum_{n=0}^{\infty} \alpha^n n L_{i-1}(n) L_i(n) = 0. \qquad \text{(A.1.17)}$$

Substituting i=i-1 in the recurrence relation, multiplying by $\alpha^n L_i(n)$ and summing

$$i \sum_{n=0}^{\infty} \alpha^n L_i^2(n) - (1-\alpha) \sum_{n=0}^{\infty} \alpha^n n L_{i-1}(n) L_i(n) = 0. \qquad \text{(A.1.18)}$$

Subtracting (A.1.17) from (A.1.18) the recurrence relation for the normalizing factor is obtained.

$$\mu_i^2 = \sum_{n=0}^{\infty} \alpha^n L_i^2(n) = \alpha \sum_{n=0}^{\infty} \alpha^n L_{i-1}^2(n) = \alpha \mu_{i-1}^2$$

or $$\mu_i^2 = \alpha^i \mu_0^2$$

$$\mu_0^2 = \sum_{n=0}^{\infty} \alpha^n = \frac{1}{1-\alpha}.$$

Define the functions

$$l_i(n) = \frac{\alpha^{\frac{n}{2}}}{\mu_i} L_i(n)$$

These satisfy

$$\sum_{n=0}^{\infty} l_i(n)\, l_j(n) = \delta_{i,j}$$

and constitute an orthonormal set over the integers 0,1 ..... The completeness of the set $L_i(n)$ follows from the completeness of the set $1, x, x^2 \ldots\ldots x^n \ldots\ldots$ over the continuous range $(0,\infty)$. For if a continuous function

$$f(x) = \sum_{i=0}^{\infty} a_i x^i$$

converges, then on the integers,

$$f(n) = \sum_{i=0}^{\infty} a_i n^i$$

and the set $1, n, n^2 \ldots\ldots n^k \ldots\ldots$ must be complete over the range 0,1,2 ..... This in turn implies that the set $L_i(n)$ which are linear combinations of the above must also be complete (theorem 2.1).

Thus the expansion formula

$$f(n) = \sum_{i=0}^{\infty} a_i l_i(n) \tag{A.1.19}$$

where

$$a_i = \sum_{n=0}^{\infty} l_i(n)\, f(n) \tag{A.1.20}$$

is valid.

A.1.2 Discrete Laguerre Functions and Discrete Linear Systems

A linear difference system[57] is described by an equation of the form

$$\sum_{i=0}^{p} a_i g(n-i) = \sum_{i=0}^{p} b_i f(n-i) \qquad (A.1.21)$$

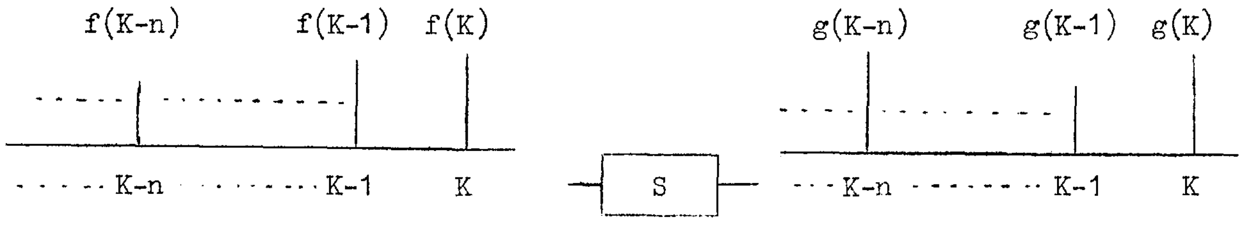
where f(n) is the input and g(n) the output waveforms. At any given time K the input and output functions may be expanded in discrete Laguerre functions $l_i(n)$ by reflecting the above waveforms and expanding according to equations (A.1.19) and (A.1.20). Hence the coefficients γ_i and β_i are defined by

$$\begin{aligned} \gamma_i(K) &= \sum_{n=0}^{\infty} f(K-n)\ l_i(n) \\ \beta_i(K) &= \sum_{n=0}^{\infty} g(K-n)\ l_i(n) \end{aligned} \qquad n \geqslant 0 \qquad (A.1.22)$$

the inverse relationships being

$$\begin{aligned} f(K-n) &= \sum_{i=0}^{\infty} \gamma_i(K)\ l_i(n) \\ g(K-n) &= \sum_{i=0}^{\infty} \beta_i(K)\ l_i(n). \end{aligned} \qquad n \geqslant 0 \qquad (A.1.23)$$

At instant K, the input and output waveforms for the system are shown below



These are related through the difference equation (A.1.21) or

$$\sum_{i=0}^{p} a_i g(K-(n+i)) = \sum_{i=0}^{p} b_i f(K-(n+i)) \qquad \text{for } n=0,1,2,\ldots..$$

Substituting equation (A.1.23) for g and f yields

$$\sum_{i=0}^{p} a_i \sum_{j=0}^{\infty} \beta_j(K)\, l_j(n+i) = \sum_{i=0}^{p} b_i \sum_{j=0}^{\infty} \gamma_j(K)\, l_j(n+i) \tag{A.1.24}$$

$l_j(n+i)$ is a j^{th} order polynomial in n multiplied by $\alpha^{(n+i)/2}$. Therefore, it must be expressible as a linear combination of the polynomials $l_s(n)$ for $s \leq j$. Hence

$$l_j(n+i) = \sum_{s=0}^{j} \mu_{ji}(s)\, l_s(n) \tag{A.1.25}$$

where the constants $\mu_{ji}(s)$ will be determined later. Substituting this into (A.1.24) and interchanging the order of summation over s and j yields

$$\sum_{s=0}^{\infty} \sum_{j=s}^{\infty} \sum_{i=0}^{p} \left[a_i \beta_j(K) - b_i \gamma_j(K)\right] \mu_{ji}(s)\, l_s(n) = 0. \tag{A.1.26}$$

Using the independence of the polynomials $l_s(n)$, the coefficients can be equated to zero yielding the set of equations

$$\sum_{j=s}^{\infty} \sum_{i=0}^{p} \left[a_i \beta_j(K) - b_i \gamma_j(K)\right] \mu_{ji}(s) = 0 \qquad s=0,1,2,\ldots.. \tag{A.1.27}$$

Before going further an expression for the $\mu_{jk}(s)$ used in equation (A.1.25) must be derived. In summation form they are defined by

$$\mu_{jk}(s) = \sum_{n=0}^{\infty} l_s(n)\, l_j(n+k). \tag{A.1.28}$$

The finite form below may be derived for $\mu_{jk}(s)$

$$\mu_{jk}(s) = \alpha^{\frac{j-s-k}{2}} \nu \sum_{m=0}^{k-1} \binom{j-s+m}{m}\binom{k}{m+1} (-1)^{j-s-1-m} \nu^{m} \qquad s=0,1,2,\ldots\ldots j-1$$

$$\mu_{jk}(s) = \alpha^{\frac{k}{2}} \qquad s=j \qquad (A.1.29)$$

$$\mu_{jk}(s) = 0 \qquad s=j+1,\ j+2,\ \ldots\ldots$$

where $\nu = (1-\alpha)$. This may seem a rather complicated expression but after much labour was the simplest finite form obtained. The proof of the above relationship is rather long and tedious, and is therefore presented in four steps. The truth of the last two expressions is demonstrated first. Then the first expression is proved for $j=1$ and all k, following which it is proved for $k=1$ and all j. Representing the points j,k by the lattice points in the first quadrant of a two-dimensional Cartesian system, the proof has then been obtained for all points on the axes of this quadrant. The induction used will be t, assume that the formula is correct for $\mu_{j-1,k}(s)$, $\mu_{j,k-1}(s)$ and $\mu_{j-1,k-1}(s)$, and then proceed to show that it must be true for $\mu_{j,k}(s)$. Using the two axes as bases, the result may then be extended to the entire plane by the above induction.

From (A.1.28) it is evident that for $s > j$, $\mu_{j,k}(s) = 0$ as $l_j(n+k)$ is a jth order polynomial in n and $l_s(n)$ is orthogonal to all such polynomials. For $j=s$ only the power of j in $l_j(n+k)$ need be considered in equation (A.1.28). Using the definition of $L_j(n)$ given in (A.1.4) and including the correct normalizing factor, $l_j(n+k)$ becomes

$$l_j(n+k) = \frac{\alpha^{\frac{n+k}{2}}(1-\alpha)^j}{\mu_j j!} n^j + \alpha^{\frac{n+k}{2}} Q(n) \qquad (A.1.30)$$

where $Q(n)$ is a polynomial of order less than j. Forming the product (A.1.28), $l_j(n)$ is orthogonal to $\alpha^{\frac{n+k}{2}} Q(n)$ and as $\sum_{n=0}^{\infty} l_j^2(n) = 1$, implying that

$$\sum_{n=0}^{\infty} l_j(n) \frac{(1-\alpha)^j n^j}{\mu_j j!} \alpha^{\frac{n}{2}} = 1 \qquad (A.1.31)$$

the fact that $\mu_{j,k}(j) = \alpha^{\frac{k}{2}}$ emerges immediately.

The first and most difficult expression will now be tackled. Explicitely

$$\mu_{1,k}(0) = \sum_{n=0}^{\infty} l_0(n)\, l_1(n+k)$$

$$= \sum_{n=0}^{\infty} \frac{\alpha^{\frac{n}{2}} \alpha^{\frac{n+k}{2}}}{\mu_0 \mu_1} ((n+k)(1-\alpha)-\alpha)$$

$$= \sum_{n=0}^{\infty} k \frac{\alpha^n \alpha^{\frac{k}{2}}}{\mu_0 \mu_1} (1-\alpha) = \frac{k\ \alpha^{\frac{k}{2}}\ (1-\alpha)}{\alpha^{\frac{1}{2}}}.$$

On substituting j=1, s=0 into (A.1.29) the value for $\mu_{1,k}(0)$ also reduces to this expression. The relations (A.1.29) are therefore proven for all s, all k, and j=1.

The second base axis requires the evaluation of $\mu_{j,1}(s)$. The difference equation (A.1.6) expressed in terms of the normalized Laguerre functions, is

$$l_{i+1}(n+1) = l_i(n) + \alpha^{\frac{1}{2}}\ l_{i+1}(n) - \alpha^{\frac{1}{2}}\ l_i(n+1). \qquad (A.1.32)$$

Equation (A.1.29) in this case reduces to

$$\mu_{j,1}(s) = \alpha^{\frac{j-s-1}{2}} \nu\ (-1)^{j-s-1}. \qquad (A.1.33)$$

The expected form for the expansion of $l_j(n+1)$ is therefore

$$l_j(n+1) = \alpha^{\frac{1}{2}} l_j(n) + \sum_{s=0}^{j-1} \alpha^{\frac{j-s-1}{2}} \nu (-1)^{j-s-1} l_s(n). \quad (A.1.34)$$

First testing $j=1$

$$l_1(n+1) = (1-\alpha)\, l_0(n) + \alpha^{\frac{1}{2}} l_1(n)$$

which may be verified by substituting the known values of l_0 and l_1. Therefore a base for an induction on equation (A.1.34) has been obtained. Assuming (A.1.33) is valid up to j, and using the relation (A.1.32)

$$l_{j+1}(n+1) = l_j(n) + \alpha^{\frac{1}{2}} l_{j+1}(n) - \alpha^{\frac{1}{2}} \left[\alpha^{\frac{1}{2}} l_j(n) + \sum_{s=0}^{j-1} \alpha^{\frac{j-s-1}{2}} (-1)^{j-s-1} \nu l_s(n)\right]$$

$$= \alpha^{\frac{1}{2}} l_{j+1}(n) + \nu l_j(n) + \sum_{s=0}^{j-1} \alpha^{\frac{j-s}{2}} (-1)^{j-s} \nu l_s(n) \quad (A.1.35)$$

completing this portion of the proof as it has now been shown that (A.1.33) must be valid for all j.

Having established the truth of expressions (A.1.29) on the j and k axes it remains to show that it is satisfied for all other points j,k. Equation (A.1.32) for the value (n+k) is

$$l_{i+1}(n+k) = l_i(n+(k-1)) + \alpha^{\frac{1}{2}} l_{i+1}(n+(k-1)) - \alpha^{\frac{1}{2}} l_i(n+k).$$

Expanding both sides of this equation using (A.1.25) and equating coefficients yields the following relation between the $\mu_{i,j}$

$$\mu_{i+1,k}(s) = \mu_{i,k-1}(s) + \alpha^{\frac{1}{2}} \mu_{i+1,k-1}(s) - \alpha^{\frac{1}{2}} \mu_{i.k}(s). \quad (A.1.36)$$

Assume equations (A.1.29) are true for all s and for the points (i, k-1), (i+1,k-1), (i,k). Required to show that it is valid for the point (i+1,k). It is a simple matter to show that the formula is valid for $s \geqslant i$. For $s < i$ the right-hand side of (A.1.36) becomes

$$\alpha^{\frac{i+1-k-s}{2}} \left\{ \sum_{m=0}^{k-2} \binom{i-s+m}{m}\binom{k-1}{m+1} (-1)^{(i-s-m-1)} \nu^{m} \right.$$

$$+ \alpha \sum_{m=0}^{k-2} \binom{i+1-s+m}{m}\binom{k-1}{m+1} (-1)^{(i-s-m)} \nu^{m} \qquad (A.1.37)$$

$$\left. - \sum_{m=0}^{k-1} \binom{i-s+m}{m}\binom{k}{m+1} (-1)^{(i-s-m-1)} \nu^{m} \right\}.$$

Substituting $\alpha=1-\nu$ in the second term contained in $\{\ \}$ changing it to two terms, shifting the summation limits from 0, k-2 to 1, k-1 in the new term created, and then combining the first three using the identity

$$\binom{a}{b} + \binom{a}{b+1} = \binom{a+1}{b+1} \qquad (A.1.38)$$

twice, reduces this expression to

$$\sum_{m-1}^{k-1} \binom{i-s+m}{m-1}\binom{k}{m+1} (-1)^{i-s-m} \nu^{m}$$

$$+ \sum_{m=0}^{k-1} \binom{i-s+m}{m}\binom{k}{m+1} (-1)^{i-s-m} \nu^{m}.$$

Adding these two together and substituting again in (A.1.37)

$$\alpha^{\frac{i+1-k-s}{2}} \sum_{m=0}^{k-1} \binom{i+1-s+m}{m}\binom{k}{m+1} (-1)^{i-s-m} \nu^{m}$$

which is indeed the expression given by (A.1.29) and the formula is proved.

Having managed to demonstrate the truth of the assertion by the above struggle, its implications can now be investigated. The first thing to notice is that $\mu_{j,k}(s)$ is a function only of j-s, and k, i.e., is essentially a two dimensional parameter.

$$\mu_{j,k}(s) = \mu_{j-s,k}(0).$$

Returning to equation (A.1.27) the following quantities are defined

$$\underline{\gamma}(K) = (\gamma_0(K),\ \gamma_1(K),\ \ldots\ldots,\ \gamma_n(K),\ \ldots\ldots)$$
$$\underline{\beta}(K) = (\beta_0(K),\ \beta_1(K),\ \ldots\ldots,\ \beta_n(K),\ \ldots\ldots)$$
$$\underline{a}^T = (a_0, a_1,\ \ldots\ldots\ a_p),\ \underline{b}^T = (b_0, b_1,\ \ldots\ldots\ b_p)$$

and the matrix

$$M(s) = \begin{array}{c} 0 \\ \vdots \\ s \\ \\ \\ \\ \end{array} \left\| \begin{array}{cccc} 0 & \ldots\ldots & & 0 \\ \vdots & & & \vdots \\ 0 & \ldots\ldots & & 0 \\ \mu_{0,0}(0), & \mu_{0,1}(0) & \ldots\ldots & \mu_{0,p}(0) \\ \mu_{1,0}(0) & \ldots\ldots & & \mu_{1,p}(0) \\ \vdots & & & \vdots \end{array} \right\| \qquad (A.1.39)$$

Using this notation equation (A.1.27) may be expressed as

$$\underline{\beta}(K)\,M(s)\,\underline{a} = \underline{\gamma}(K)\,M(s)\,\underline{b} \qquad s=0,1,2,\ldots\ldots$$

The elements $\mu_{i,k}(0)$ are independent of s, s only determining the number of zero rows. These represent an infinite set of infinite linear equations implying that , in theory, given $\underline{a}$, $\underline{b}$, M(s) and the input spectra $\underline{\gamma}(K)$, they may be inverted to find the output spectra $\underline{\beta}(K)$. As this involves the inversion of an infinite matrix it is hardly practicable. For obtaining an equation in the coefficients $\underline{a}$ and $\underline{b}$ it is of somewhat more use.

When making linear combinations of these equations it is only necessary to add or subtract the matrices M(s) as these are the only properties of the equation that are dependent on the equation number.

For example

$$\underline{\beta}(K)\, M(s)\, \underline{a} - \underline{\beta}(K)\, M(s+m)\, \underline{a} = \underline{\beta}(K) \left[M(s) - M(s+m) \right] \underline{a}.$$

Take the matrices (A.1.39) and form a new set of matrices $Q(s)$ by taking linear combinations of the $M(s)$ as shown

$$Q(s) = \sum_{i=0}^{p} M(s+i) \binom{p}{i} \alpha^{\frac{i}{2}} .$$

With some difficulty, using the properties of $\mu_{i,k}(0)$, $Q(s)$ can be proved to have the form

$$Q(s) = \left\| \begin{matrix} O_1 \\ Q_p \\ O_2 \end{matrix} \right\|$$

where O_1 is $(s+1)x(p+1)$ zero matrix, O_2 is an $\infty x(p+1)$ zero matrix and Q_p is a $(p+1)x(p+1)$ square matrix.

The elements of Q_p, symbolized by $q_{k,j,p}$ satisfy the recurrence relation

$$q_{k,j,p} = q_{k,j,p-1} + \alpha^{\frac{1}{2}}\, q_{k-1,j,p-1} \qquad \begin{matrix} 0 \leqslant k \leqslant p \\ 0 \leqslant j \leqslant p-1 \end{matrix} \qquad \text{(A.1.40)}$$

where it is assumed that both $q_{-1,j,p-1}$ and $q_{p,j,p-1}$ equal zero. Also

$$q_{k,p,p} = \binom{p}{k} \alpha^{(p-k)/2}$$

the two relations together yielding a method for obtaining Q_p from Q_{p-1}. It may easily be shown that

$$Q_1 = \left\| \begin{matrix} 1 & \alpha^{\frac{1}{2}} \\ \alpha^{\frac{1}{2}} & 1 \end{matrix} \right\|$$

Using the above equations and defining $\delta = \alpha^{\frac{1}{2}}$ the matrices Q_2 and Q_3 are found as shown below

$$Q_2 = \left\| \begin{matrix} 1 & \delta & \delta^2 \\ 2\delta & 1+\delta^2 & 2\delta \\ \delta^2 & \delta & 1 \end{matrix} \right\| \qquad Q_3 = \left\| \begin{matrix} 1 & \delta & \delta^2 & \delta^3 \\ 3\delta & 1+2\delta^2 & 2\delta+\delta^3 & 3\delta^2 \\ 3\delta^2 & 2\delta+\delta^3 & 1+2\delta^2 & 3\delta \\ \delta^3 & \delta^2 & \delta & 1 \end{matrix} \right\|$$

Matrices of as high an order as desired may be generated.

The relation between the input and output spectral coefficients at time K can now be expressed by

$$\underline{\beta}(K)\ Q(s)\ \underline{a} = \underline{\gamma}(K)\ Q(s)\ \underline{b} \qquad s=0,1,\ldots\ldots \qquad (A.1.41)$$

where Q(s) is an ∞ x(p+1) matrix with only a (p+1)x(p+1) submatrix non-zero. This now represents an infinite number of finite equations, linear in the system coefficients $\underline{a}$ and $\underline{b}$. These equations are normalized so that $a_0=1$. If the 2n+1 system parameters $\underline{a}$ and $\underline{b}$ remaining, are unknown, then from a knowledge of the spectral coefficients and equation (A.1.41), they may be determined. Taking the first 2n+1 equations of (A.1.41) the final expression

$$B(K)\ Q_p\underline{a} = G(K)\ Q_p\underline{b}. \qquad (A.1.42)$$

is derived, where B(K) and G(K) are matrices formed from the coefficients $\beta_i(K)$ and $\gamma_i(K)$ in the same fashion as the matrix B(t) on page 147.

B(K) and G(K) are (2n+1)x(p+1) matrices; $\underline{a}$ and $\underline{b}$ are p+1 vectors. This last equation may be put in the partitioned form

$$\left\| B(K) \;\vdots\; -G(K) \right\| \left\| \begin{array}{c|c} Q_p & 0 \\ \hline 0 & Q_p \end{array} \right\| \left[\frac{a}{b} \right] = 0 \qquad (A.1.43)$$

ready for solution. As $a_0=1$, each linear equation must have a constant term and the ordinary methods of solution may be applied.

A.1.3 Continuous Up-Dating of Spectra

Calculating the spectra at each stage by using equations (A.1.22) necessitates storing all the past history of the input and output and going through a long multiplication and addition process each time. Using the previous spectrum, however, the only additional information required to obtain this new spectrum is the present values of the corresponding functions.

At stage K+1 the spectral formula is

$$\begin{aligned}\beta_i(K+1) &= \sum_{n=0}^{\infty} f(K+1-n)l_i(n)\\ &= \sum_{n=0}^{\infty} f(K-n)l_i(n+1) + f(K+1)l_i(0)\\ &= \sum_{n=0}^{\infty} f(K-n) \sum_{j=0}^{i} \mu_{i,1}(j)l_j(n) + f(K+1)l_i(0)\\ &= \sum_{j=0}^{i} \mu_{i,1}(j)\, \beta_j(K) + f(K+1)l_i(0).\end{aligned}$$

In matrix form this equation is

$$\underline{\beta}(K+1) = \mu\, \underline{\beta}(K) + f(K+1)\, \underline{l}(0) \tag{A.1.44}$$

where the matrix μ is an infinite triangular matrix with elements

$$\begin{aligned}\mu_{i,1}(j) &= (1-\alpha)\alpha^{\frac{i-j-1}{2}} (-1)^{i-j-1} && \text{for } i-j > 0\\ &= \alpha^{\frac{1}{2}} && \text{for } i=j\\ &= 0 && \text{fpr } i-j < 0.\end{aligned}$$

The vector $\underline{l}(0)$ from formula (A.1.14) and the normalizing constant is

$$l_i(0) = (-1)^i\, \alpha^{\frac{i}{2}} \quad \sqrt{1-\alpha}.$$

As the matrix μ is defined by a linear array of elements the application of this up-dating formula is particularly easy. Also, since it is a triangular matrix, if only a finite number of the coefficients $\beta_i(K)$ are necessary then the matrix μ is finite and the updating process involves no approximations. By using higher order $\mu_{j,k}(s)$ terms it is possible to update every k^{th} interval instead of every interval.

APPENDIX II

Quine-McCluskey Minimization of Boolean Functions

Assume that a boolean function is to be expanded as a series of conjunctions. A conjunctive cover for a function f is defined as a set of cubes (conjunctions) such that all the members of f are included in at least one cube, and all members of $\bar{f}$ are excluded. Many examples of such covers occur in Chapter IV. The weight of a cover is defined as the total number of constants in the cover, or equivalently, by the number of occurrences of variables x_i in the series of conjunctions defining the function.

The function (1,3,5) may be covered by the two cubes

$$\begin{array}{ccc} x_1 & x_2 & x_3 \\ \\ 0 & x & 1 \\ x & 0 & 1 \end{array} = \bar{x}_1 \, x_3 + \bar{x}_2 \, x_3 .$$

The weight of this cover is four. The Quine-McCluskey method minimizes the weight for realizing an arbitrary function. This in turn implies minimization of the total number of cubes[22]. The method is best explained by an example.

The function chosen to illustrate is that defined on page 87. Firstly the 'zero' cubes, as taken directly from the truth table, are set down in a column, ordered according to the number of 1's appearing in them (figure A.1.1). One cubes can only be formed between adjacent vertical blocks, as for non-adjacent blocks the members must differ in at least two places. All possible one cubes are formed and a ✗ is placed

0 0 0 0 1 x̶	0 0 0 x 1 x̶	0 0 x x 1
0 1 0 0 0 x̶	0 0 x 0 1 x̶	x x 0 0 1
1 0 0 0 0 x̶	0 x 0 0 1 x̶	x 0 0 x 1
0 0 0 1 1 x̶	0 1 0 0 x	0 x 1 1 x
0 0 1 0 1 x̶	0 1 x 0 0	x 1 1 1 x
0 0 1 1 0 x̶	x 0 0 0 1 x̶	
0 1 0 0 1 x̶	1 0 0 0 x	
0 1 1 0 0 x̶	1 0 x 0 0	
1 0 0 0 1 x̶	0 0 x 1 1 x̶	
1 0 1 0 0 x̶	0 0 1 x 1 x̶	
0 0 1 1 1 x̶	0 0 1 1 x x̶	
0 1 1 1 0 x̶	0 x 1 1 0 x̶	
1 0 0 1 1 x̶	0 1 1 x 0	
1 1 0 0 1 x̶	x 1 0 0 1 x̶	
1 1 0 1 0 x̶	x 0 0 1 1 x̶	
0 1 1 1 1 x̶	1 0 0 x 1 x̶	
1 1 1 0 1 x̶	1 x 0 0 1 x̶	
1 1 1 1 0 x̶	0 x 1 1 1 x̶	
1 1 1 1 1 x̶	0 1 1 1 x x̶	
	x 1 1 1 0 x̶	
	1 1 x 0 1	
	1 1 x 1 0	
	x 1 1 1 1 x̶	
	1 1 1 x 1	
	1 1 1 1 x x̶	

Prime implicants are those without an x̶.

Quine-McCluskey Chart
Minimization Method

FIGURE A.1.1

by the zero cubes that go to make them. The process is then repeated with the one cubes to produce two cubes, the asterisks being placed in the same fashion. This is continued until no further higher order cubes can be found. The cubes that are unmarked (the prime implicants) form a cover for the function, the minimum cover being some subset of these cubes.

Which subset, is found by drawing up a 'prime-implicant' chart, (A.1.2), where each of the prime implicants found above are set down, along with the elements of f that they cover. If an element is only covered by one implicant then that implicant must be necessary for the minimal cover. After removing these cubes and deleting all elements that are covered by them, the reduced chart of figure (A.1.3^a) is obtained. In the example, 5 cubes are found necessary from this first inspection and six elements of f remain to be covered.

Eliminate all implicants in this chart that are completely covered by a single other implicant (e.g., x111x). In the example this reduces us to considering five implicants. Again include all essential implicants for this new chart and proceed until, either all the elements are exhausted, or until each of them is covered by at least two implicants. In the former case the minimum cubical cover will have been derived.

If, in a reduced chart, there appear to be no essential implicants, then recourse must be made to a process called 'branching'. One implicant is chosen at random and assumed to be a member of the minimum cover. All the elements covered by this implicant are eliminated and the table is reduced until the minimum cover, including this implicant, is obtained. Then the minimum cover without this implicant is also

Elements of f

Prime Implicants	1	3	5	6	7	8	9	12	14	15	16	17	19	20	25	26	29	30	31
0 0 x x 1	+	+	+		+														
x x 0 0 1	+						+					+			+				
x 0 0 x 1	+	+										+	+						
0 x 1 1 x				+	+				+	+									
x 1 1 1 x									+	+								+	+
0 1 0 0 x						+	+												
0 1 x 0 0						+		+											
1 0 0 0 x											+	+							
1 0 x 0 0											+			+					
0 1 1 x 0								+	+										
1 1 x 0 1															+		+		
1 1 x 1 0																+		+	
1 1 1 x 1																	+		+

Prime Implicant Chart for
Cubes of Figure A.1.1

FIGURE A.1.2

Elements of f
not covered

Prime Implicants	8	9	12	25	29	31
x x 0 0 1		+		+		
x 1 1 1 x						+
0 1 0 0 x	+	+				
0 1 x 0 0	+		+			
1 0 0 0 x						
0 1 1 x 0			+			
1 1 x 0 1				+	+	
1 1 1 x 1					+	+

First essential implicants

0 0 x x 1
0 x 1 1 x
x 0 0 x 1
1 0 x 0 0
1 1 x 1 0

Reduced Prime Implicant Table
(a)

Prime Implicants	8	9	12	25	29	31
x x 0 0 1		+		+		
0 1 0 0 x	+	+				
0 1 x 0 0	+		+			
1 1 x 0 1				+	+	
1 1 1 x 1					+	+

Second essential implicants

0 1 x 0 0
1 1 1 x 1

Third essential implicant

x x 0 0 1

Table after eliminating
Covered Implicants
(b)

FIGURE A.1.3

obtained. Finally, comparing the two, the true minimum can be found. This branching process may have to be repeated a number of times but it is normally not too difficult to pick out the best cover by a cursory inspection.

Miller[22] has evolved a method, similar to the above, for finding the prime implicants from an arbitrary cover of the function. This circumvents the rather large chart in step one that results when functions of many variables are considered.

APPENDIX III

Computer Subroutines

A.3.1 Subroutine SOLVE

This subroutine finds the solution for the system coefficients using the discrete Laguerre spectrum of the input and output, and equation (7.35). Three different methods were attempted to find the solution to these. The first and most obvious took no advantage of the special form of the matrices B(K) and G(K) (see page 147), merely multiplying these by Q_p as found in Appendix I and then proceeding to find the inverse directly. This method not only took longer than the other two methods below, but also gave less accurate results.

Equation (7.35) may be put in the equivalent form of (A.1.43) of Appendix I. Remembering that $a_0=1$ this equation may be represented as

$$\left\| B(K) \,\vdots\, - G(K) \right\| \underline{d} = 0 \qquad (A.3.1)$$

where $\underline{d}$ is a (2n+2) vector whose components are linear combinations of the 2n+1 unknowns a_i and b_i, produced by taking the product of the last two terms. This is further modified to

$$\left\| \underline{\alpha} \,\vdots\, C \right\| \begin{bmatrix} d_0 \\ \cdots \\ \underline{d}' \end{bmatrix} = 0 \qquad (A.3.2)$$

where C is now a square (2n+1)x(2n+1) matrix, d_0 is the first component of $\underline{d}$, $\underline{d}'$ being the remaining components, and $\underline{\alpha}$ the first column of the matrix in (A.3.1). This last equation can be changed to

$$C\underline{d}' = - d_0\underline{\alpha}. \qquad (A.3.3)$$

C being square has an inverse provided $|C|$ is non-zero. The inverse of C was found by orthogonalizing the rows of C by the Schmidt orthogonalization procedure (see chapter II, page 27). For an orthogonal matrix U, $UU^T = I = U^TU$[10]. Orthogonalization of C is equivalent to multiplying it by a transformation matrix A yet to be determined. Hence

$$(AC)^T(AC) = C^TA^TAC = I.$$

Multiplying both sides of (A.3.3) by C^TA^TA results in

$$\underline{d}' = - d_0 \, C^TA^TA \, \underline{\alpha}. \tag{A.3.4}$$

The great advantage of this form as far as computation is concerned is that after A has been determined there are no matrix multiplications as A, then A^T, then C^T may be applied successively to the vector $\underline{\alpha}$ to produce the required result.

Given a set of row vectors $\underline{u}_1, \underline{u}_2, \ldots\ldots \underline{u}_k$ to orthogonalize these by the Schmidt procedure[51] involves forming the successive terms

$$\begin{aligned} \underline{v}_1 &= \underline{u}_1 \\ \underline{v}_2 &= \underline{u}_2 - \langle \underline{u}_2, \underline{v}_1 \rangle \; \underline{v}_1 / \langle \underline{v}_1, \underline{v}_1 \rangle \\ &\vdots \\ \underline{v}_k &= \underline{u}_k - \sum_{i=1}^{k} \left(\langle \underline{u}_k, \underline{v}_i \rangle \; \underline{v}_i / \langle \underline{v}_i, \underline{v}_i \rangle \right) \end{aligned} \tag{A.3.5}$$

where $\langle \underline{a}, \underline{b} \rangle$ denotes the inner product. The normalizing constants for the vectors are defined by $\mu_i = \sqrt{\langle \underline{v}_i, \underline{v}_i \rangle}$, and give an opportunity for checking the consistency of the equations. In the programme if the value of μ_i^2 fell below a certain fixed value the equations were considered to be inconsistent and the process was stopped.

It may be shown that the only matrix product required for this orthogonalizing procedure is CC^T which, due to the special form of C, may be calculated in considerably fewer operations than calculating the matrix product directly. The number of operations to find a matrix product is of order n^3, while taking advantage of the symmetry in C, this may be reduced to an operation of order n^2. Matrix A is triangular and may easily be found from (A.3.5). There seems to be no time saving way of performing the orthogonalization of the equations. With a little thought the square root necessary to find the normalizing constant can be eliminating, saving another subroutine.

The best way to explain the final method for extracting the solution is to follow through an example. In the two dimensional case the equation for the coefficients $\underline{a}_i$ is

$$Q_2 \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = d_0 \begin{bmatrix} 1 \\ d_1/d_0 \\ d_2/d_0 \end{bmatrix} \qquad \text{(A.3.6)}$$

The constants d_i/d_0 have already been determined from (A.3.4) and are given by $-C^TA^TA\underline{\alpha}$. The Q_p have a very simple inverse; if $q_{i,j}$ is the i,j^{th} element of Q_p then it may be shown that $(-1)^{i+j}\, q_{i,j}\, (1-\alpha)^p$ is the i,j^{th} element of Q_p^{-1}. Therefore, multiplying both sides of (A.3.6) by Q_2^{-1} yields the equation

$$\begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = d_0\, Q_2^{-1} \begin{bmatrix} 1 \\ d_1/d_0 \\ d_2/d_0 \end{bmatrix}$$

where the last multiplication can be performed yielding an expression

$$\begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = d_0 \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

where all the α's are known. From this $d_0 = \frac{1}{\alpha_1}$ producing a final solution of the form

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \alpha_2/\alpha_1 \\ \alpha_3/\alpha_1 \end{bmatrix}$$

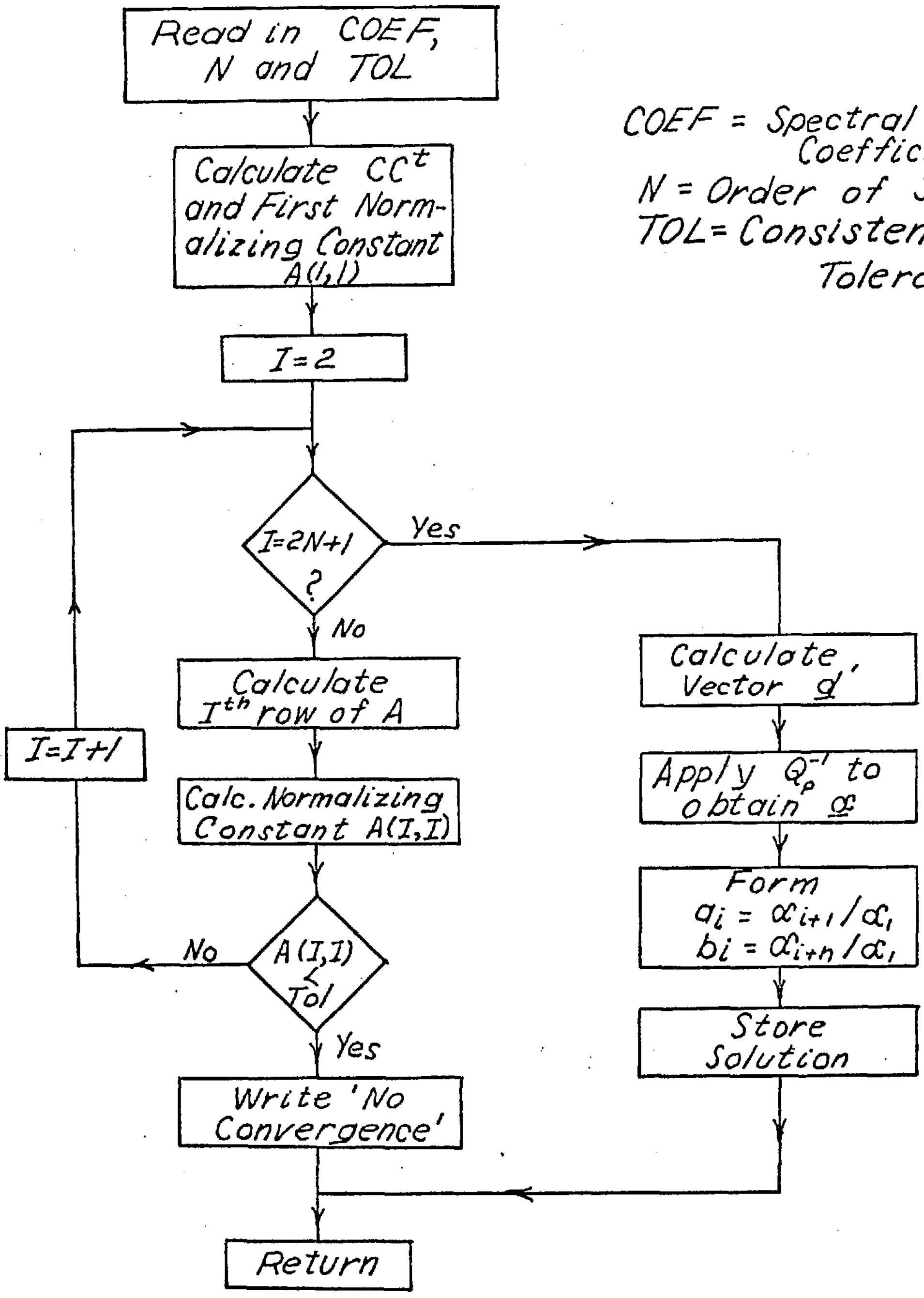
and the coefficients are found. Note that in taking the ratios the constant $(1-\alpha)^p$ disappears and so need not be carried through the calculations. The solution for $\underline{b}$ is found similarly.

For most practical systems $b_0=0$, as the input cannot normally affect the output instantaneously. This can also be incorporated into the calculations to reduce the number of parameters that require solution, although it involves some further complications in the matrix operations required. However, in all the results quoted in chapter VII the modified programme that assumes $b_0=0$ was employed.

Figure (A.3.1) gives a somewhat condensed flow diagram for the programme. Two modifications of this programme were tested, the first took the equationsin the order as they appeared, the second taking them in the reverse order to see if this would affect the orthogonalization procedure at all. There was nothing to choose between these two programmes when comparing the two final results.

A.3.2 Subroutine ROOTS

This subroutine takes a polynomial of any order and finds all its real and complex roots. It was developed in order to plot the pole-zero diagrams for the system identification procedure using the discrete Laguerre polynomials, and is used in conjunction with subroutine SOLVE described in the previous section.



Read in COEF, N and TOL
Calculate CC^t and First Normalizing Constant A(1,1)
I=2
I=2N+1 ?
Yes
No
Calculate I^{th} row of A
Calc. Normalizing Constant A(I,I)
A(I,I) < Tol
No
Yes
I=I+1
Write 'No Convergence'
Calculate, Vector $\underline{d}'$
Apply Q_p^{-1} to obtain $\underline{\alpha}$
Form $a_i = \alpha_{i+1}/\alpha_1$ $b_i = \alpha_{i+n}/\alpha_1$
Store Solution
Return

COEF = Spectral Coefficients
N = Order of System
TOL = Consistency Tolerance

Flow Diagram for Subroutine SOLVE

Figure A.3.1

The Newton-Raphson method is used to improve the estimate of the roots with a slightly modified scheme for obtaining the complex roots. $P_n(x)$ represents a polynomial in x of order n. For real roots the iteration scheme

$$x_n = x_{n-1} - P_n(x_{n-1})/P_n'(x_{n-1}) \tag{A.3.7}$$

is used where $P_n'(x)$ represents the first derivative evaluated at x. If in carrying out the above procedure the iterations start to diverge then the estimates must have passed through the real part of a complex root. The complex pair can be estimated by fitting a parabola through the point x_n using the values of the derivative and function already calculated at that point. Then the corresponding complex roots of this parabola are found as an initial estimate to the complex pair.

A two dimensional Newton method is used for refining the complex roots. A complex pair, defined in polar co-ordinates by $\rho e^{j\phi}$ and $\rho e^{-j\phi}$, must satisfy the two equations

$$\sum_{i=0}^{n} a_i \rho^i \, e^{ji\phi} = 0, \qquad \sum_{i=0}^{n} a_i \rho^i \, e^{-ji\phi} = 0,$$

where a_i are the coefficients of $P_n(x)$. Subtracting and adding these two results in the two real equations

$$f_1 = \sum_{i=0}^{n} a_i \rho^i \cos i\phi = 0, \qquad f_2 = \sum_{i=0}^{n} a_i \rho^i \sin i\phi = 0 \tag{A.3.8}$$

The trigonometric identities

$$\rho^i \sin i\phi = \alpha \rho^{i-1} \sin((i-1)\phi) + \beta \rho^{i-1} \cos((i-1)\phi)$$

and

$$\rho^i \cos i\phi = \alpha \rho^{i-1} \cos((i-1)\phi) - \beta \rho^{i-1} \sin((i-1)\phi) \tag{A.3.9}$$

where $\alpha = \rho\cos\phi$ is the real part of the root, and $\beta = \rho\sin\phi$ is the complex part, may be used to obtain the terms $\rho^i \sin i\phi$ and $\rho^i \cos i\phi$ from terms with lower index, and α and β. Given an arbitrary α and β and the above formula it is an easy matter to evaluate f_1 and f_2. The problem is to find a point (α, β) such that f_1 and f_2 are simultaneously zero.

The constants a and b, defined below, may also be obtained

$$\frac{\partial f_1}{\partial \phi} = -\sum_{i=1}^{n} i a_i \rho^i \sin i\phi = a = -\rho \frac{\partial f_2}{\partial \rho}$$

$$\frac{\partial f_2}{\partial \phi} = \sum_{i=1}^{n} i a_i \rho^i \cos i\phi = b = \rho \frac{\partial f_1}{\partial \rho}.$$

Hence
$$df_1 = a d\phi + \frac{b}{\rho} d\rho$$

$$df_2 = b d\phi - \frac{a}{\rho} d\rho$$

which on solving for the infinitesimals $d\phi$ and $d\rho$ yield

$$d\phi = (a df_1 + b df_2)/(a^2 + b^2)$$

$$d\rho = (-a df_2 + b df_1)/(a^2 + b^2).$$

From the definition of α and β

$$d\alpha = -\rho \sin\phi d\phi + \cos\phi d\rho$$

$$= -\beta d\phi + \alpha d\rho/\rho$$

$$d\beta = \alpha d\phi + \beta d\rho/\rho$$

producing the final relationships

$$d\alpha = \left\{(\alpha b - \beta a)\, df_1 - (\alpha a + \beta b)\, df_2\right\} /(a^2 + b^2)$$

$$d\beta = \left\{(\alpha a + \beta b)\, df_1 + (\alpha b - \beta a)\, df_2\right\} /(a^2 + b^2). \tag{A.3.10}$$

On fitting tangent planes to f_1 and f_2 at the point (α, β), the first order correction to the complex root may be found by finding where the line of intersection of these planes passes through the plane $f_1 = f_2 = 0$. Thus having the values f_1, f_2, a and b at (α, β) the correction to α and β may be found. This takes the form

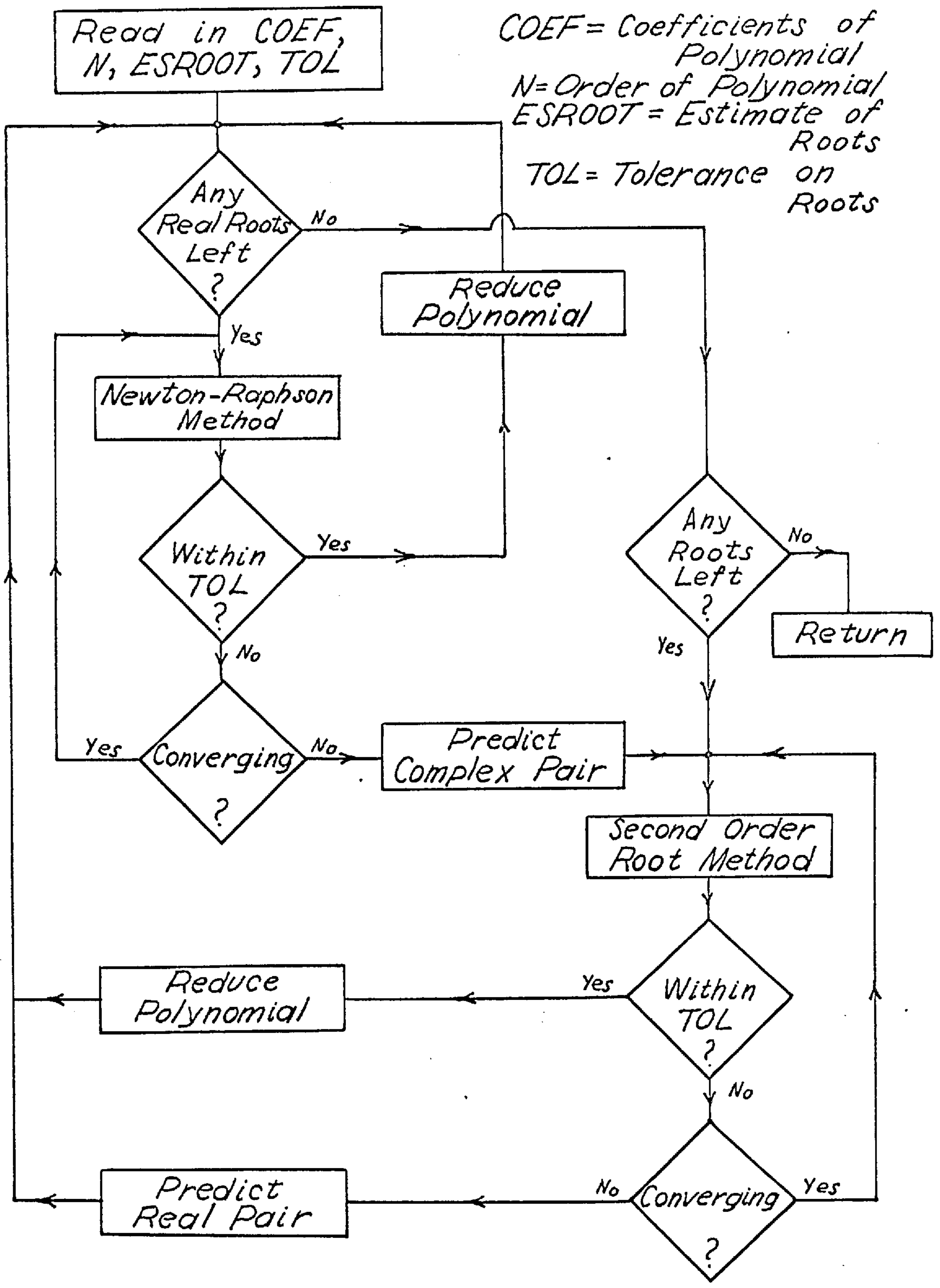
$$\alpha_n = \alpha_{n-1} - \Delta\alpha_{n-1}$$

$$\beta_n = \beta_{n-1} - \Delta\beta_{n-1}$$

where $\Delta\alpha_{n-1}$ and $\Delta\beta_{n-1}$ are calculated directly from (A.3.10).

There is, however, a danger in applying this procedure directly. Note that for $\beta = 0$, $\phi = 0$ and hence f_2 is always zero. Also for a complex root there is a saddle point on the $\beta = 0$ axis indicating that if this method happened to converge on this axis it could never get off. Because of this it was decided to follow f_2 contours until f_1 equals zero and then follow the f_1 contour to the final root. This implies setting $df_2 = 0$ in equation (A.3.10) until the value of f_1 is below a certain tolerance, and then keeping it within this tolerance until f_2 has converged to a satisfactory value. If this method fails to converge then there must be real roots and a pair of these is predicted, the mode being changed to the ordinary method described first. The root is said to be found if the value of the function is less than a certain tolerance that may be read into the programme separately.

Figure (A.3.2) represents a simplified flow diagram for the subroutine. An iteration test was included to make sure the procedure did not go on forever in some anomalous cases. The programme was tested for a number of root configurations including double roots and even



Flow Diagram for Subroutine ROOTS

Figure A.3.2

double complex roots and was found to give very satisfactory performance. For example it extracted all the roots of a sixth order equation, having two pairs of complex roots and a double real root, to a tolerance of 0.0001 in a matter of 70 iterations.

REFERENCES

1. Scott, N.R. 'Analogue and Digital Computer Technology'. McGraw Hill Series in Information Processing and Computers, pp. 60-88, published 1960.

2. Howe, R.M. 'Design Fundamentals of Analogue Computer Components'. D. Van Nostrand Co., pp. 187-204, published 1961.

3. Jackson, A.S. 'Analogue Computation'. McGraw Hill, pp. 485-494, published 1960.

4. Wilkinson, R.H. 'Generating Functions of Several Variables Using Analogue Diode Logic'. IEEE Trans. - E.C., pp. 112-129, April 1963.

5. Sansone, G. 'Orthogonal Functions'. Interscience Publishers, 1959.

6. Winograd, S. 'On the Time Required to Perform Addition'. IBM Research Paper RC-1183, May 1964.

7. Arbib, M. 'Steps Towards a Theory of Difficulty of Computation'. Lecture, University of London Institute of Computer Science, May 1964.

8. Gantmacher, F.R. 'Matrix Theory'. Chelsea Publishing Company, 1960.

9. Trench, F. 'An Algorithm for the Inversion of Finite Hankel Matrices'. J.Soc. Indust. and Applied Maths., (SIAM), pp. 1102-1107, December 1965.

10. Bellman, R. 'Introduction to Matrix Analysis'. McGraw Hill Series in Matrix Theory, published 1960.

11. Courant, R. and Hilbert, J.D. 'Methods of Mathematical Physics'. Interscience Publishers, 1953.

12. Mishkin, E. and Braun, L. 'Adaptive Control Systems'. McGraw Hill Series in Electrical and Electronic Engineering, published 1961.

13. Hamming, R.W. 'Numerical Methods for Scientists and Engineers'. McGraw Hill, 1962.

14. Golomb, S.W. 'On the Classification of Boolean Functions'. IRE Trans. on Circuit Theory, pp. 176-186, May 1959.

15. Lim, C.H. 'The Application of the Orthogonal Property of Binary Chain Codes'. Ph.D. Thesis, Imperial College, 1966.

16. Elspas, B. 'The Theory of Autonomous Linear Sequential Networks'. IRE Trans. on Circuit Theory, pp. 45-60, March 1959.

17. Schmid, H. 'Linear Segment Function Generator'. IRE Trans.-E.C., pp. 780-788, December 1962.

18. Sikorski, R. 'Boolean Algebras'. Springer-Verlag (publishers), 1960.

19. Krichevskii, R.E. 'Realization of Functions by Superpositions'. Problems of Cybernetics, 2, pp.458-477, 1959. (Translation from the Russian).

20. Post, E.L. 'Introduction to a General Theory of Elementary Propositions'. American Journal of Mathematics, 43, No. 1, pp. 163-185, 1921.

21. Huffman, D.A. 'Solvability Criterion for Simultaneous Logical Equations'. Quarterly Progress Report, Research Laboratory of Electronics, MIT., pp. 87-88, January 1958.

22. Miller, R.E. 'Switching Theory', Vol. I and II., John Wiley and Sons, 1965.

23. Lupanov, O.B. 'Complexity of Formula Realization of Functions of Logical Algebra'. Problems of Cybernetics, 1, 1961.

24. Ozernoy, V.M. 'Minimization of Combinational Logic Circuits Constructed from Unifunctional Elements'. Engineering Cybernetics, No. 1, 1964.

25. Smith R.A. 'Minimal Three-Variable Nor and Nand Logic Circuits'. IEEE Trans on Electronic Computers, pp. 79-81, February 1965.

26. Miyata, F. 'Realization of Arbitrary Logical Functions Using Majority Elements'. IEEE Trans on Electronic Computers, pp. 183-

27. Ellis, D.T. 'A Synthesis of Combinational Logic with Nand or Nor Elements'. IEEE Trans on Electronic Computers, pp. 701-705, October 1965.

28. McCluskey, E.J. and Bartie, T.C. 'A Survey of Switching Circuit Theory'. McGraw Hill, 1962.

29. Radchenko, A.N. and Filippov, V.I. 'Shift Registers with Logical Feedback' Automation and Remote Control . pp. 1467-1473, November 1959. (Translation from the Russian).

30. Preparata, F.P. 'State Logic Relationships for Autonomous Sequential Networks'. IEEE Trans. on Electronic Computers, pp. 542-548, October 1964.

31. Yoeli, M. 'Counting with Non-Linear Binary Feedback Shift Registers'. IEEE Trans. on Electronic Computers, August 1963.

32. Heath, F.G. and Gribble, M.W. 'Chain Codes'. Proc. IEE, <u>108C</u>, pp. 50-57, 1961.

33. Peterson, W.W. 'Error-Correcting Codes'. MIT Press and John Wiley and Sons, 1962.

34. Golomb, S.W. 'Digital Communications'. Prentice Hall.

35. Nichols, A.J. 'Minimal Shift Register Realization of Sequential Machines'. Trans. IEEE, E.C., October 1965.

36. Susskind, A.K. 'Notes on Analog-Digital Conversion Techniques'. MIT Press and John Wiley and Sons, 1958.

37. Tou, J.T. 'Digital and Sampled-Data Control Systems'. McGraw Hill, 1959.

38. Verster, T.C. 'A Method to Increase the Accuracy of Fast-Serial-Parallel Analog-to-Digital Counters'. Trans. IEEE, E.C., pp. 471-473, August 1964.

39. 'A Bidirectional Binary Counter'. Ferranti Applications Report, 1963.

40. 'A Reversible Decade Counter Using Combi.-Elements'. Mullard Applications Report, 1962.

41. Halworth, R.P. and Heath, F.G. 'Semiconductor Circuits for Ternary Logic'. Proc. IEE, <u>109C</u>, pp. 219-225.

42. Yoeli, M. and Rosenfeld, G. 'Ternary Switching Circuits'. IEEE Trans, E.C., February 1965.

43. Yarshavskii, V.I. 'Ternary Majority Logic'. Automatica e Telemechanica (translation), <u>25</u>, No. 5, pp. 612-622.

44. Forsyth, A.R. 'Calculus of Variations'. Published first in 1926, Dover edition 1960.

45. Leitmann, G. 'Optimization Techniques'. Academic Press, 1962.

46. Bellman, R.E. and Dreyfus, S.E. 'Applied Dynamic Programming'. Princeton University Press, 1962.

47. Merriam III, C.V. 'Optimization Theory and the Design of Feed-back Control Systems'. McGraw Hill, 1964.

48. Yovits, M.C., Jacobi, G.T. and Goldstein, G.D. 'Self-Organizing Systems'. Spartan Books, Washington, 1962.

49. Waltz, M.D. 'A Learning Control System'. JACC at Stanford, 1964, Paper No. 1.

50. Feigenbaum, E.A. and Feldman, J. 'Computers and Thought'. McGraw Hill, 1963.

51. Zadeh, L.A. and Desoer, C.A. 'Linear System Theory (the State Space Approach). McGraw Hill, 1963.

52. Fano, R.M. 'Transmission of Information'. MIT Press and John Wiley and Sons, 1961.

53. Blackman, P.F. 'Notes on a Time Optimal Position Control System'. Imperial College, 1962.

54. Lee, Y.W. 'Statistical Theory of Communication'. John Wiley and Sons, 1960.

55. D'Azzo, J.J. and Houpis, C.H. 'Control System Analysis and Synthesis'. McGraw Hill, 1960.

56. Zor[illegible], J. 'Methods of Evaluating Fourier Transforms with Application to Control Engineering'. Ph.D. Thesis, University of Delft, 1963.

57. Jury, E.I. 'Theory and Application of the z-Transform Method'. John Wiley and Sons, 1964.