NORTHEASTERN UNIVERSITY

COMMUNICATIONS RESEARCH GROUP

QUARTERLY PROGRESS REPORT NO. 2

Period Covered: 1 December 1965 through 28 February 1966

Date Submitted: 23 March 1966

for

Grant NGR-22-011-013

Stephen J. O'Neil
Grant Monitor

Sponsored by Electronics Research Center, NASA

This report is intended for internal management uses of the grantee and the NASA.

Submitted for the staff by

Sze-Hou Chang

Sze-Hou Chang
Principal Investigator

1. Investigations Being Undertaken and Planned

During this period, a literature survey was continued, a series of information theory seminars was started, and basic research was conducted with emphasis placed equally on development of theory and its applications.

The seminars were presented in two parts. Part I is an introductory exposition of information theory. Part II consists of four research papers. There are altogether eight one hour lectures, each followed by a half hour discussion. The topics are as follows:

(1) Measure of Information,

(2) Source Encoding,

(3) Capacity of Information Channel,

(4) Channel Encoding,

(5) Dual Product Codes,

(6) Sampling,

(7) Intersymbol Interference,

(8) Non-Binary Orthogonal Codes.

Three lectures have already been presented as of the time of report.

Four items of work, three in the area of coding, one in the area of signal representation, are reported below.

(i)--On Decoding Binary Three-Error-Correcting (15,5) B-C-H Code

The existing decoding schemes for B-C-H codes involve solving simultaneous equations in an algebraic extension field GF(q), $q = p^m$, where p is a prime number and m is an integer. The calculation in the field GF(q) is, in general, much more complicated than that in the ground field, i.e., GF(p), especially when multiplications are involved which is the usual case in all the existing decoding schemes. Therefore, it is very desirable to seek some decoding schemes without resorting to calculations in the algebraic extension fields. If this is possible, a tremendous saving in both decoding circuitry and procedures can be achieved.

By making full use of the cyclic property of the binary 3-error-correcting (15,5) B-C-H code and one of its algebraic structures, we are able to correct all the 3-or-less errors in any received code word. Let the generator polynomial of the (15,5) code be

$$\begin{aligned} g(x) &= (x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1) \\ &= x^{10}+x^8+x^5+x^4+x^2+x+1 \ . \end{aligned} \tag{1}$$

The parity-check matrix H and the generator matrix G are then

$$H = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&1&0&0&0&0&0&0&0&0&1&1&1&1&1 \\ 0&0&1&0&0&0&0&0&0&0&1&1&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&1&0&1 \\ 0&0&0&0&1&0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&1&0&0&0&0&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&1 \end{bmatrix} \tag{2}$$

and

$$G = \begin{bmatrix} 1&1&1&0&1&1&0&0&1&0&1&0&0&0&0 \\ 0&1&1&1&0&1&1&0&0&1&0&1&0&0&0 \\ 1&1&0&1&0&1&1&1&1&0&0&0&1&0&0 \\ 0&1&1&0&1&0&1&1&1&1&0&0&0&1&0 \\ 1&1&0&1&1&0&0&1&0&1&0&0&0&0&1 \end{bmatrix} . \tag{3}$$

Since $x^4+x^3+x^2+x+1$ is a factor of $g(x)$, any code word satisfying the parity-check matrix H of Equation (2) should also satisfy the following matrix H' corresponding to $x^4+x^3+x^2+x+1$.

$$H' = \begin{bmatrix} 1&0&0&0&1 & 1&0&0&0&1 & 1&0&0&0&1 \\ 0&1&0&0&1 & 0&1&0&0&1 & 0&1&0&0&1 \\ 0&0&1&0&1 & 0&0&1&0&1 & 0&0&1&0&1 \\ 0&0&0&1&1 & 0&0&0&1&1 & 0&0&0&1&1 \end{bmatrix} . \tag{4}$$

The decoding procedure can now be listed as follows.

(1) Feed the received vector into the syndrome calculator as shown in Fig. 1.

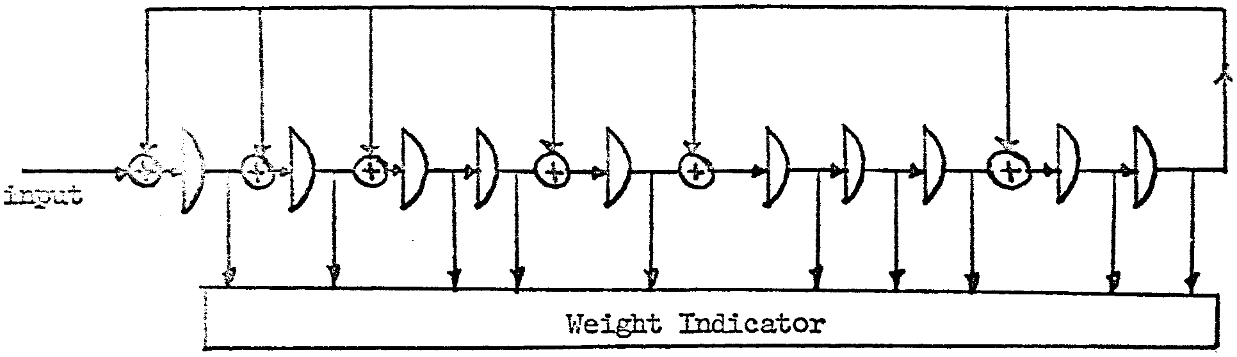


Fig. 1 Syndrome Calculator

If the weight indicator shows that the weight, i.e., the total number of 1's of the syndrome is zero, then there is no error in the received vector. If the weight shown is no more than 3, there is no error in the first k = 5 information digits. Thus the errors are in the check digits and no correction is required.

(2) If the weight of the syndrome is larger than 3, then there is at least one error in the first k = 5 information digits. Allow the syndrome to shift by itself with no further input until at the i^{th} shift the weight indicator shows a weight of 3 or less. Then take this shifted "syndrome", denoted by $\underbrace{S_s}$, and form $\underbrace{e'} = \underbrace{\underbrace{S_s}\,0\;0\;0\;0\;0}$. The true error pattern can be obtained by shifting $\underbrace{e'}$ backward i times.

(3) If the weight of each of the 15 shifted "syndromes" is larger than three, we know there are exactly three errors evenly spaced. Then we feed the received code vector into the auxiliary "subsyndrome" calculator shown in Fig. 2 whose parity check matrix is given by Equation (4).

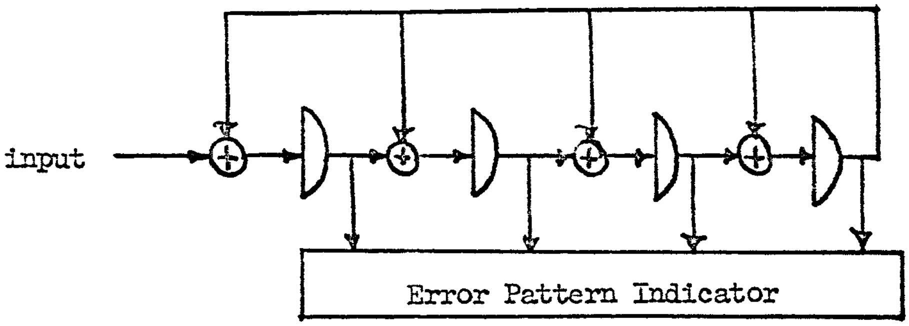


Fig. 2 The Auxiliary Subsyndrome Calculator

The error pattern indicator "transforms" the subsyndromes into the error patterns in the following manner:

(1000)	(10000 10000 10000)
(0100)	(01000 01000 01000)
(0010)	(00100 00100 00100)
(0001)	(00010 00010 00010)
(1111)	(00001 00001 00001)

(5) If the subsyndrome turns out to be something different from that listed in Steps 1 to 4, then there are more than 3 errors in the received vector which is out of the range of the error-correcting ability of the code.

The following examples will help to clarify the decoding procedure.

Example 1

Let $\underline{r}$ = (0 1 1 0 1 0 0 0 0 0 1 0 0 0 0) be the received vector. Then the syndrome $\underline{s}$ is

$$\underline{s} = \underline{r}\,H^T = (1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0).$$

The weight of the syndrome

$$w[\underline{s}] = 3.$$

Hence the information digits (10000) are correct. No correction is required.

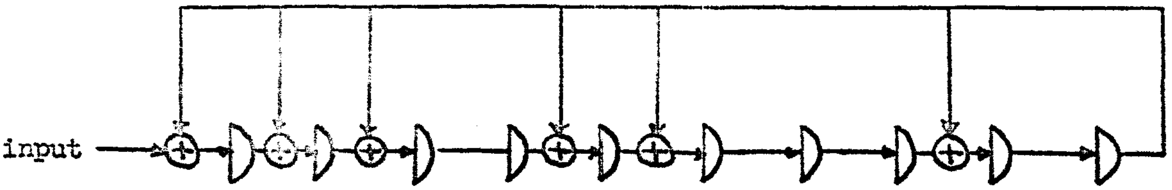
Example 2

$$\underline{r} = (0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0)$$

$$\underline{s} = \underline{r}\,H^T = (1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0)$$

$$w[\underline{s}] = 6 > 3.$$

Hence we should shift the syndrome in the syndrome calculator as shown in Fig. 3.



											weight
	1	1	0	0	0	1	1	1	1	0	6
1st shift	0	1	1	0	0	0	1	1	1	1	6
2nd shift	1	1	0	1	1	1	0	1	0	1	7
3rd shift	1	0	0	0	0	0	1	0	0	0	$\boxed{2}$

Fig. 3 Cyclic Shifts of a Syndrome

$\therefore$ $\underline{S_s} = (1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0)$, since $w[\underline{S_s}] = 2 < 3$

and $\underline{e'} = (\underbrace{1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0}_{\underline{S_s}}\ 0\ 0\ 0\ 0\ 0)$

$\underline{e}$ = 3 backward shifts of $\underline{e'}$

$= (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0)$.

Hence the correct code word $\underline{v}$ is

$$\underline{v} = \underline{r} + \underline{e}$$

$$= (0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0).$$

Example 3

$$\underline{r} = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0)$$

$$\underline{s} = \underline{r}\ H^T = (1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$$

$$w[\underline{s}] = 7 > 3.$$

The weight of each of the 15 shifts of the syndrome is larger than 3. Hence, we try $\underline{s}' = \underline{r}\,H' = (0\ 0\ 1\ 0)$, we know the error pattern should be

$$\underline{e} = (0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0)$$

and

$$\underline{v} = \underline{r} + \underline{e} = (1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0)$$

Example 4

$$\underline{r} = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)$$

$$\underline{s} = rH^T = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$$

$$w[\underline{s}] = 10 > 3$$

The weight of each of the 15 shifts of the syndrome is larger than 3. Hence, we try

$$\underline{s}' = \underline{r}\,H' = (0\ 0\ 0\ 0)$$

which does not belong to any of the error patterns listed before. Hence, we conclude that the received vector contains more than three erroneous digits.

Extension of this simple technique to the decoding of other cyclic codes is possible. However, a fundamental limitation exists; namely, $\epsilon < \frac{n}{k}$ where ϵ is the number of corrigible errors by this method, n is the total number of digits of a code word, and k is the number of information digits. In most cases ϵ is smaller than t, the number of theoretically corrigible errors.

In certain applications, it may be desirable to have an extremely simple decoding procedure which corrects a major fraction of theoretically corrigible errors and sounds an alarm whenever the error exceeds that fraction. The above procedure is particularly suitable for such applications.

(ii)--Dual Product Codes

The dual product codes are defined as follows:

Given:

Code	Generator Matrix	Parity-Check Matrix
C_1	G_1	H_1
C_1'(dual of C_1)	H_1	G_1
C_2	G_2	H_2
C_2'(dual of C_2)	H_2	G_2

Eight Dual Product Codes:

D_1	$G_1 \otimes G_2$	
D_1'(dual of D_1)		$G_1 \otimes G_2$
D_2	$G_1 \otimes H_2$	
D_2'(dual of D_2)		$G_1 \otimes H_2$
D_3	$H_1 \otimes G_2$	
D_3'(dual of D_3)		$H_1 \otimes G_2$
D_4	$H_1 \otimes H_2$	
D_4'(dual of D_4)		$H_1 \otimes H_2$

The following properties and applications are anticipated:

(a) A wide range of error control properties is covered by 8 dual product codes, applicable to variable channels, such as the communication and control channel of a space vehicle whose distance from the earth is increasing. Thus as S/N decreases,the information rate is decreased (by choice of the code) allowing protection of the information by redundancy.

(b) The 8 dual product codes may be implemented through suitable logic circuits interconnecting the two basic code circuits.

(c) Special applications, such as error-locating ability (to reduce the amount of feedback transmission), are possible.

(d) The codes can correct bursts and random errors in combinations.

To illustrate the capabilities of the component codes and the product codes, take C_1 to be a (15,5) B-C-H code whose decoding scheme is discussed above, and C_2 to be a (7,3) Hamming code. Then we have the following table:

		Error Control Capability		
Code	(n,k)	Min. Distance	No. of Detectable Errors	No. of Correctable Errors
C_1	(15,5)	7	6	3
C_1'	(15,10)	4	3	2 (adjacent)
C_2	(7,3)	4	3	2 (adjacent)
C_2'	(7,4)	3	2	1
D_1	(105,15)	28	27	13
D_2	(105,20)	21	20	10
D_3	(105,30)	16	15	7
D_4	(105,40)	12	11	5

For the dual product codes D_1' through D_4' , it is more convenient to consider that a code word is arranged in 15 subblocks each of size 7 digits.

		Error Control Capability					
Code	(n,k)	Detection		Location		Correction	
		No. of Subblocks	Errors in Subblocks	No. of Subblocks	Errors in Subblocks	No. of Subblocks	Errors in Subblocks
D_1'	(105,90)	3	2	2 (adj.)	2	2 (adj.)	1
D_2'	(105,85)	3	3	2 (adj.)	3	2 (adj.)	2 (adj.)
D_3'	(105,75)	6	2	3	2	3	1
D_4'	(105,65)	6	3	3	3	3	2 (adj.)

It should be noted that by reversing the order of multiplication in the product, the capabilities of the codes D_1 through D_4 remain unchanged, while those of the dual codes D_1' through D_4' can be interpreted by interchanging the columns under "number of subblocks" with the columns under "errors in subblocks". In the latter cases, a code word is arranged in 7 subblocks each containing 15 digits.

Various combinations can be obtained if different codes are used as C_1 and C_2, the components codes. For example, either C_1 or C_2 or both may be burst-error-correcting codes instead of independent-error-correcting codes.

(iii)--A Scheme for Implementing M-ary Logic

While generalized multi-level theory and abstractions have been used in cyclic codes, M-sequences, and orthogonal matrices, little practical implementation has ever been attempted in other than the binary case. This has motivated a search for a reliable means of realizing m-ary logic.

The philosophy of the approach used here is that, while ternary or higher level logic circuits are not presently available in a reliable form, binary logic elements are. Hence our experience with these allows us to deal with binary logic synthesis, minimization, and implementation in a straightforward way.

Normally, we desire to synthesize the following m-ary logical functions:

(1) multiplication modulo m,

(2) storage of an m-ary level for an arbitrary length of time,

(3) addition modulo m.

Using the scheme presented here, this synthesis may be accomplished using only and/or gates, threshold circuits, binary flip flops, and resistive weighting networks. All of these circuits are well known and have been integrated in numerous forms.

The method of implementation is primarily dependent upon the triggering of any or all m-1 threshold circuits (in a modulo m logic function). By a proper choice of these threshold circuits, the input level is determined, a Boolean transformation performed on the threshold circuit outputs to convert to binary form, and finally the binary form is converted to its respective analog equivalent (by a weighted resistive adder network).

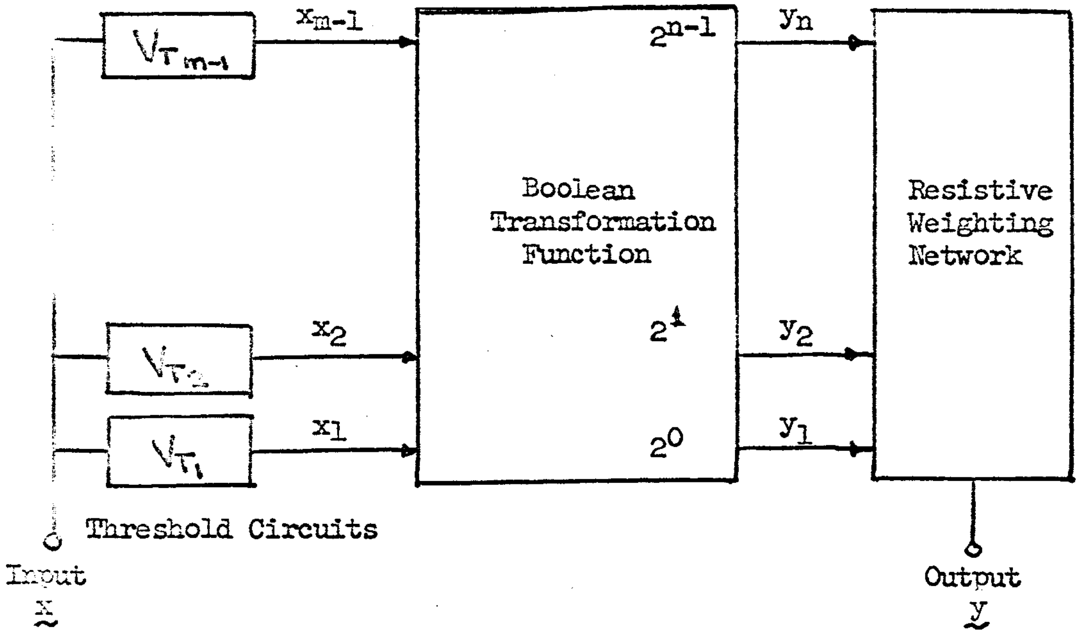


Fig. 4 Scalar Multiplier (xa)(mod m)

For example, multiplication modulo m is accomplished using the circuit of Fig. 4. The threshold circuits are adjusted so that their respective thresholds V_{T_i} are midway between each logic level. A function is selected, e.g., scalar multiplication (x3) (mod 7), and then the Boolean transformation is selected such that the binary outupts $(y_1, \ldots, y_n)$ correspond to this prescribed function. This means every binary output y_i is a Boolean function of the threshold outputs, i.e., $y_i = f_i(x_1, x_2, \ldots, x_{m-1})$. Minimization of these functions is straightforward to about modulo 7 (six threshold gates).

An m-ary flip flop may then be realized by storing each binary output y_i of a (x1) (modulo m) multiplier in a binary flip flop (at the incidence of a clock pulse or other selected signal).

The modulo m adder may be realized by converting the two inputs to be added into n-tuples using the threshold circuits, and then selecting a Boolean transformation to give addition modulo m. For example, two inputs $\underset{\sim}{a}$ and $\underset{\sim}{b}$ to be added would be converted to $(a_1, \ldots, a_{n-1})$ and $(b_1, \ldots, b_{n-1})$ respectively. The binary outputs y_i would then be a function of these, e.g., $y_i = f_i(a_1, \ldots, a_{n-1}, b_1, \ldots, b_{n-1})$, such that $\underset{\sim}{y} = (\underset{\sim}{a} + \underset{\sim}{b})$ (modulo m).

This scheme has these advantages:

(1) m-ary logic may be realized using standard, well known, and reliable binary circuits.

(2) While the m-ary circuits proposed are more complex than their binary counterparts, when integrated, their total physical size (with encapsulation, interconnection leads, etc.) may make them nearly equivalent.

(3) It allows implementation and heuristic investigation of multi-level codes, M-sequences, and orthogonal matrices, other than the traditional binary case.

(iv)--Signal Representation

It is well known that a time function $g(t)$ which lasts for all time and which contains no energy above f_c cps can be represented by sampling $g(t)$ at uniform intervals spaced $1/2f_c$ seconds apart. The samples would be $g_k = g(\frac{k}{2f_c})$, $k = \ldots -2,-1,0,1,2,\ldots$. The reconstructed $g(t)$ is

$$g(t) = \sum_{k=-\infty}^{\infty} g_k \, v_k(t)$$

where $\{v_k(t)\}$ is a set of orthogonal time functions which provide the correct interpolation between the samples. In fact $v_k(t)$ is a shifted version of $v_o(t)$, where

$$v_o(t) = \frac{\sin(2\pi f_c t)}{2\pi f_c t}$$

is the familiar $\frac{\sin x}{x}$ form with zeros at the sampling instants. See Fig. 5.

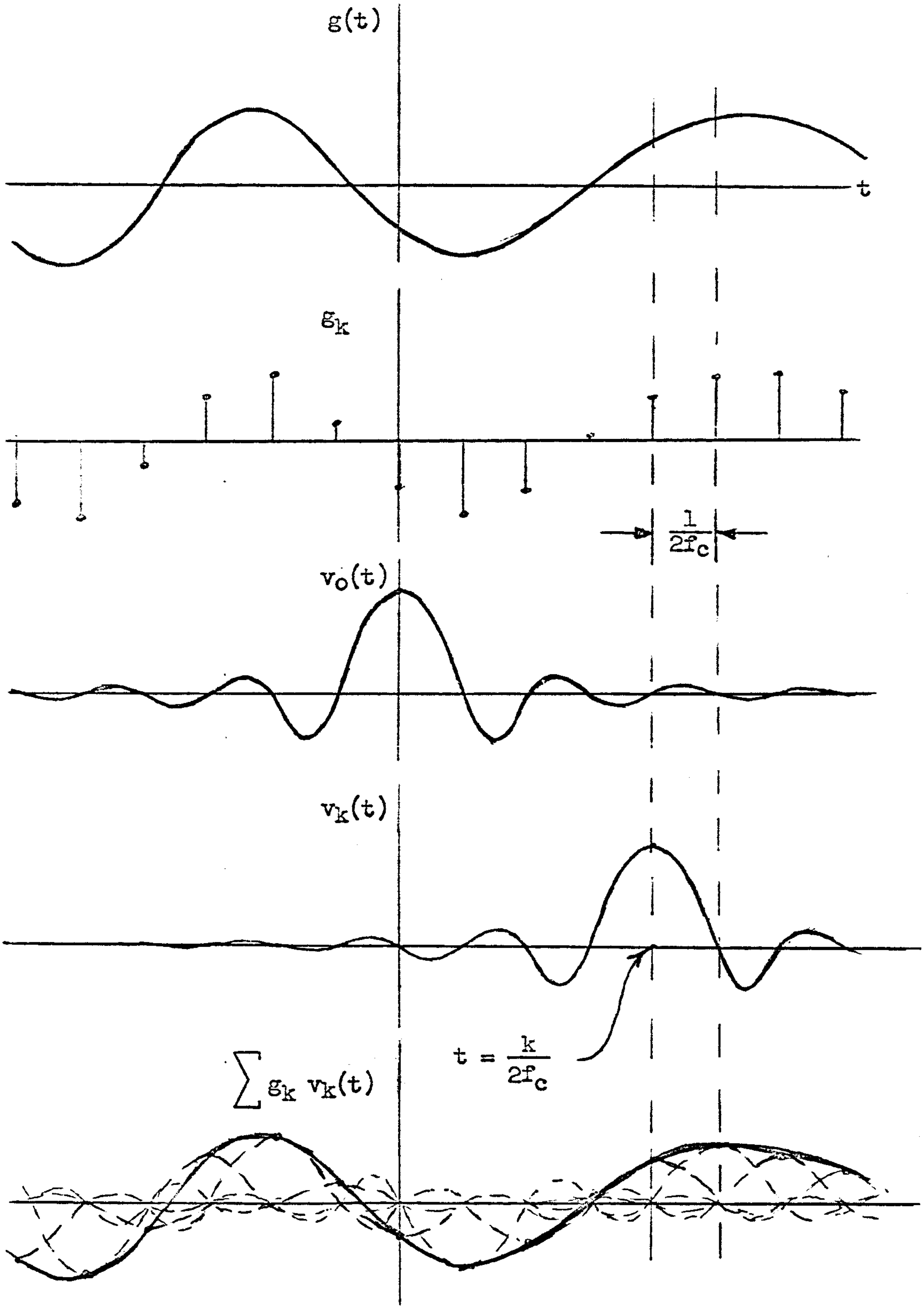


Fig. 5

This representation is known to have a wide range of usefulness because of (1) the simplicity of determining the coefficients g_k -- they are samples of $g(t)$, and (2) the set $\{v_k(t)\}$ is an orthogonal set.

We have been able to find a generalization for the "sampling theorem" presented above. The generalization provides a similar sampling theorem for time functions which are not necessarily bandlimited, but which do have similar restrictions. Thus the first generalization applies to polynomials of N^{th} degree or less.

The generalization hinges on the discovery of a "sampling function" for the particular class or space of time function under consideration. Such a sampling function $v_o(t)$ has a zero at all the sampling instants t_k, $k \neq 0$, and is unity at t_o. The orthogonal set $\{v_k(t)\}$ which provide the interpolation between samples is found by removing from $v_o(t)$ the zero at t_k and placing it at t_o. Thus the sampling function for the space of bandlimited time functions is exactly

$$\frac{\sin(2\pi f_c t)}{2\pi f_c t}$$

which is one at $t = 0$ and is zero at the other sampling instants $t_k = \frac{k}{2f_c}$. $v_k(t)$ is found by putting a zero at $t_k = \frac{k}{2f_c}$ and removing the one at $t = 0$. See Fig. 5.

Under certain restrictions* on an acceptable space $\mathfrak{F}_N$, the sampling function for the space is given by

$$v(t) = a_0 \sum_{k=0}^{N} \phi_k(t_o)\ \phi_k(t),$$

where $$a_0 = \frac{1}{\sum \phi_k{}^2(t)}\text{, a constant}$$

*These restrictions involve a continuity restraint and will not be discussed here. Proofs for these statements will appear in a forthcoming report or paper.

and $\{\phi_k(t)\}$ is any set of N+1 orthonormal time functions spanning $\mathfrak{F}_N$. Then any g(t) in $\mathfrak{F}_N$ has the representation

$$g(t) = \sum_{k=0}^{N} g_k \, v_k(t)$$

where g_k is g(t) sampled at $t = t_k$, and t_k is the kth zero of $v_o(t)$. $v_k(t)$ is $v_o(t)$ with the zero at $t = t_k$ replaced by one at $t = t_o = 0$.

This generalization extends to (but is not necessarily restricted to) the following spaces of time functions:

(a) the space of all Nth degree polynomials on the interval $A < t < B$;

(b) the space of all time functions with finite-dimensional Fourier series expansion on the interval $0 < t < T$;

(c) the space of all time functions spanned by $e^{-at}, te^{-at}, \ldots, t^N e^{-at}$ on the interval $0 < t < \infty$ (this is the space of the first N+1 Laguerre functions);

(d) the space of all time functions spanned by $e^{-at}, e^{-2at}, e^{-3at}, \ldots, e^{-Nat}$ on the interval $0 < t < \infty$ (this is the space of the first N+1 Legendre functions).

The above results may be useful in a wide variety of applications. For example: (1) Suppose the Laplace transform H(s) of a system function is an (N+1)-pole rational function where the poles must all be at a single point, or must be uniformly spaced along the negative real axis (corresponding to (c) and (d) above). Then the impulse response h(t) is completely characterized by N+1 samples of h(t) taken at time intervals dictated by the appropriate sampling function and the effect of H(s) (perhaps as a link in a control system) can be easily studied in the time domain. (2) In P.C.M. applications the sampling theorems dictate when one should sample the data in order to produce independent samples in the presence of white noise. Thus, if

$$s(t) = \sum s_k \, v_k(t)$$

is the signal to be sampled and $\{s_k\}$ is the set of samples, then the received samples

$$r_k = s_k + n_k ,$$

where n_k is a sample of white noise, are independent (with the resulting simplifications of analysis) if and only if the s_k are independent, or if the $v_k(t)$ are orthogonal. Such is the case for the samples of our sampling theorems.

2. Paper Presented

A short paper, on "Dual Product Codes" by Sze-Hou Chang and Lih-Jyh Weng, was presented to the IEEE International Symposium on Information Theory, January 31 - February 2, 1966 at UCLA.

3. Paper Submitted

A paper, entitled "A Note on Non-Binary Orthogonal Codes", by Sze-Hou Chang was submitted and accepted by the forthcoming IEEE International Communication Conference to be held on July 15-17, 1966 in Philadelphia.

4. Conferences and Meetings

On December 17, 1965, a project meeting was held at Northeastern University. It was attended by Charles F. Hobbs of AFCRL, Stephen J. O'Neil and Jean R. Roy of ERC, NASA, Communications Research group at Northeastern University and graduate students. Sze-Hou Chang and Robert Gonsalves, both of Northeastern University faculty, presented topics on coding and signal representation.

On January 31 - February 2, 1966, Sze-Hou Chang attended the symposium at UCLA and presented a paper as noted above.

George Sollman of Northeastern University serves as a liaison, and meets regularly with Stephen J. O'Neil and Jean R. Roy, both of ERC.