REPRESENTATION OF TWO-TONE IMAGES USING PSEUDORANDOMLY DERIVED SQUARE MATRICES

By

JOHN ERIK HERSHEY

Bachelor of Science in Physics
Massachusetts Institute of Technology
Cambridge, Massachusetts
1965

Bachelor of Science in Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts
1966

Master of Science
University of Arizona
Tucson, Arizona
1968

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
December, 1981

REPRESENTATION OF TWO-TONE IMAGES

USING PSEUDORANDOMLY DERIVED

SQUARE MATRICES

Thesis Approved:

Rao Yarlagadda
Thesis Adviser

Robert J. Mulholland

Richard L. Cummins

Marvin S. Keener

Norman N Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

First, I express my thanks to my wife and family for their encouragement and support in pursuit of this degree.

Second, I thank Dr. P. McManamon of the Institute for Telecommunication Sciences for his support.

Third, I thank Dr. Rao Yarlagadda for his guidance and friendship.

Fourth, I thank the members of my doctoral committee for their time and interest.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

F	The frame
K	The dimension of the frame
R	The number of codematrices
d_{α}	The hamming metric
C	The compression
ε	The acceptable error rate
σ	The probability that a specific codematrix falls in the tail region
$\Phi_{\Sigma}(\sigma,R)$	Probability that at least one codematrix falls in the tail region
C_{∞}	Compression afforded in limit as frame size becomes infinite
ϕ	Euler totient function
0_n	The all zero square matrix of nxn elements
I_n	The identity matrix of nxn elements
M	The finite field generator matrix
m_i	Element of the top row of the general form of the field generator matrix
$\underline{m}$	The top row of the general form of the field generator matrix
$L_{i,j}(\underline{m})$	A linear combination of elements taken from $\underline{m}$
P	A permutation matrix
δ	The kronecker symbol
λ	A finite field element

CHAPTER I

INTRODUCTION

1.1 Introductory Remarks

Of all the engineering tradeoffs which arise in large system design and management, one of the most prevalent, indeed nearly ubiquitous, of the inverse relations is that of computational speed versus storage. These are computational complexity and available communications bandwidth.

This is a technologically interesting and exciting age compared to even the near past. It used to be that "systems thinking" was secondary if it indeed ever occurred. The theoretician was not particularly concerned if his algorithm could not be implemented in any extant or even foreseeable computation device. The radio engineer did not really have to be overly concerned about bandwidth contention.

It was primarily due to military and space research and engineering programs that a unification was achieved between the theoreticians, communications engineers, computational specialists, and hardware experts. Modern military contingency planning requires the accurate and timely transfer of enormous quantities of information. The military and space efforts both require high speed communications to and from transceivers located at decentralized, remote and uninhabitable locations.

One discipline in particular, image compression and pattern recognition, is of great importance and involves tradeoffs among the four

variables. A typical scenario is shown in Figure 1. In Figure 1, Remotely Piloted Vehicles (RPVs) are depicted with on-board sensors reporting back to a centralized processing facility, via a satellite, elements from their optical field of view. If, for example, the RPVs are travelling at Mach 3 and it is required to send a 1024 x 1024 pixel, two-tone picture back every 1/2 mile, simple encoding of the picture with one bit per pixel results in a required bit rate of about a megabit per second per RPV. If there are 100 PRVs, then the satellite must handle 100 megabits per second.

It is clear, then, that the bit rates may easily exceed the transmission relay facilities available. For this reason it is desirable to seek methods by which the information may be compressed or otherwise winnowed before transmission.

Another application example is that of template storage. Consider a weapon delivery system that attempts to match electro-optical or other two-dimensional sensory data against stored templates. How can the desired templates be quickly coded and entered into a reasonably sized, moderately low cost (expendable) memory? What efficient and universal technique may be used to encode the image data for storage in the weapon system's on-board memory?

Let us assume that each template is a 1024 x 1024 pixel, two-tone image. If the system is to be able to recognize 1000 targets, complete and direct encoding would require a storage capability of over one billion bits.

Fortunately, as we will show later on in the paper, templates may be highly corrupted by noise and still perform well. This fact and the phenomenon of the "global" spreading of the information by the method to be

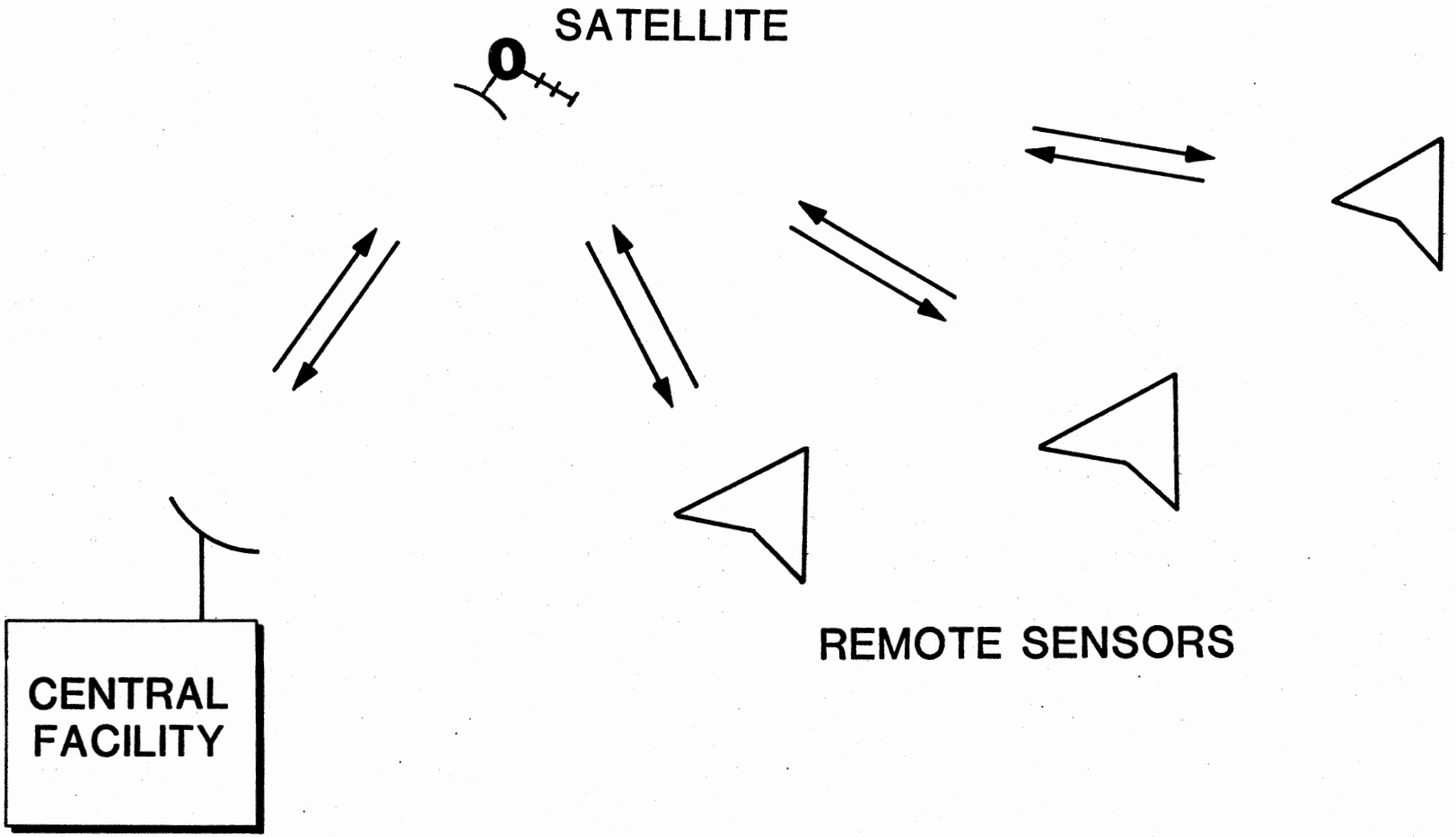


Figure 1. Typical Multiple-Access Random Demand Scenario

proposed in section 1.2 (and discussed in section 1.3) unite to commend the method to be proposed for consideration for template encoding and storage in limited capacity memory facilities.

1.2 Image Representation and Compression

The direction of contemporary image representation and compression research is towards orthogonalization schemes which winnow out the less important, small coefficients of various transforms and thereby achieve a bandwidth reduction [1-4], and other methods aimed at reducing redundancy such as run-length coding [5]. There is little doubt that this approach is efficacious provided, of course, that the image yields exploitable characteristics under the transform or other algorithm being applied. Also, there is the question of computational complexity, time, and hardware cost of implementation. Suboptimal methods such as Hadamard compression [6] can be made to operate relatively quickly as compared to a Karhunen-Loéve scheme, for example.

Huang [5], in his excellent paper on coding of two-tone images, opines that, "The three basic heuristic concepts are: skipping white; transmitting only boundary points between black and white; and pattern recognition." (p. 1408). In this paper we offer a fourth, truly heuristic method. The method will not usually be optimum but it will be universal and it is particularly motivated by practical concepts such as speed and ease of representation as well as flexibility and integrability into modern telecommunication architectures. What is proposed is image representation and subsequent compression by way of a relatively unsophisticated method, but a method that, nevertheless, has many

commending advantages for some specialized situations, particularly in the military arena.

The concept is directed primarily towards image representation rather than that of image compression; however, compression naturally results as the proposed representation scheme requires fewer elements of information than the image itself. The price paid for the compression is exacted in terms of noise.

1.2.1 The Image

We will be concerned with an image, P, that is mathematically expressible as a bounded set of contiguous lattice elements (such as small squares).

$$P = (p_{ij}) \tag{1.1}$$

where

$$p_{ij} \varepsilon \{0,1\}$$

Thus P is a two-tone matrix, e.g., each element is either black or white and an image of n elements per row with n rows is exactly described by a string of n^2 bits.

1.2.2 Decomposition Into Frames

A frame, F, is defined as a square submatrix of the image matrix. The frame is of dimension k x k and is derived as follows:

$$F = (f_{ij}) \qquad 1 \leq i, j \leq k \tag{1.2}$$

where

$$f_{ij} = p_{i+I, j+J}$$

where I and J move the frame across the image. To decompose the image into non-overlapping adjacent frames, k must divide n without remainder and

$$\begin{cases} I = k\ell \\ J = km \end{cases} \qquad 0 \le \ell,\ m \le \frac{n}{k} - 1 \tag{1.3}$$

How can we represent the frames with less than k^2 bits? This is the essential question of the paper.

1.2.3 Representation by Selected Matrices

We assume we have a set of k x k matrices:

$$\{N_1, N_2, \ldots, N_R\} \tag{1.4}$$

The set in Equation (1.4) is termed the "codematrix" set. Codematrix is a synthesized word, offered and employed as an analogy to the linear connoting term "codeword."

We calculate the number of disagreeing pixels, d_i, between a given frame and each of the codematrices. The disagreement numbers are placed into the set $D = \{d_1, d_2, \ldots, d_R\}$ in a natural ordering, i.e., d_i is the number of disagreeing pixels between the frame and codematrix N_i. Let $\{n^k_{ij}\}$ represent the elements of matrix N_k. The number of disagreements between the frame and this matrix, d_k, is the number of ones in the matrix produced by exclusive oring, term-by-term, N_k and F. This can be written directly as

$$\sum_{i=1}^{k} \sum_{j=1}^{k} (n^k_{ij} + f_{ij} - 2n^k_{ij} f_{ij}). \tag{1.5}$$

(In Equation (1.5) we used "+" and "-" as normal addition and subtraction, respectively. We will usually use "+" to represent modulo 2 addition. It

is believed that those cases wherein "+" denotes normal addition will be self-evident.) The frame is represented by a codematrix ν where

$$d_\nu = \min \{d_1, d_2, \ldots, d_R\}. \tag{1.6}$$

At this juncture, four remarks are in order:

1. The compression achievable is

$$\frac{k^2}{\lceil \log_2 R \rceil} \tag{1.7}$$

where

$$\lceil \alpha \rceil = \begin{cases} \alpha, & \alpha \text{ an integer} \\ [\alpha] + 1, & \alpha \text{ not an integer} \end{cases}$$

where $[\alpha]$ is the integer part of α.

2. Some sets of matrices will be better than others with respect to minimizing the average number of disagreements, the variance of the number of disagreements and other moments.

3. There will be an upper limit to the number of errors that can be made in each frame's representation. This limit will be determined by the maximum d_ν observed when all possible frame matrices are represented using the codematrix set given in Equation (1.4).

4. The system that implements the image representation will have to provide Rk^2 bits of memory in order to store the set of R matrices <u>unless</u> the matrices can be generated from a smaller amount of stored information.

The question then naturally presents itself: "Can one devise a set of matrices that provides adequate representation at a given compression ratio and does not require a burdensome amount of storage or computation time to create?" This question is an embellishment and refinement of the

paper's essential question and will be the main concern of Chapter III.

1.3 A Comparison

An old concept for achieving compression at the expense of resolution is to decimate the image field by transmitting only a portion of the elements such as every other line, every other point, etc. This type of compression is equivalent to laying a mesh or "grill" over the image and transmitting only that portion of the image that is visible. In addition to reduced resolution, there is a problem with periodic grill sampling (or indeed any periodic sampling) and this is the problem of aliasing [7]. The aliasing may be of two types. The first occurs if the image contains periodic structures. This situation may give rise to annoying and possibly obfuscating Moire patterns. The second type of aliasing results from the sampling lattice or grill being insensitive to certain regions of spatial frequencies. This can cause smearing, edge definition problems, and incorrect amplitude information.

Matching by randomly-derived matrices differs from the above "grill" concept in a number of ways and some rather unexpected advantages accrue. First of all, let us consider the aliasing problem. It is clear that we may still be plagued with aliasing if large portions of the image field are locally periodic, for then the selection of best matched matrices may also be periodic. For this reason we choose to prewhiten the image before representing it with a collage of randomly derived matrices. Prewhitening consists of merely adding (modulo two) randomly derived bits to the image before putting it into frames and matching the codematrices to the frames.

To explore another, more subtle, difference between this method and the grill method, let us reflect on the "philosophy" of the representation

methods by way of a simple example. Consider that we have a grill that skips every other line and every other column. A cell of the grill is shown in Figure 2. Note that the grill lets 1 bit of information pass through and suppresses 3 bits for each cell. Given the data allowed through the grill, and assuming that all images are equally likely, it is clear that the best estimate or reconstruction of the image is provided by filling in the obscured bits by randomly derived bits, such as from a balanced Bernoulli source. Following this line of reasoning we make the following two remarks.

1. The grill of Figure 2 provides a compression of 4:1.

2. One bit per cell is known exactly. Half of the filled-in bits will, on the average, be correct.

Thus, the reconstructed image will exhibit an average error rate of (3/2)/4 = 0.375. Note that we are sending one bit per grill cell and using this bit to specify precisely one, specific, pixel.

Consider now that we represent the image by specifying, for each 2 x 2 frame, the matrix from the arbitrarily chosen codematrix set:

$$\left\{\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix}\right. , \left.\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}\right\} \tag{1.8}$$

which exhibits the fewest disagreements. Figure 3 displays the results obtained by this alternate method.

The results displayed in Figure 3 show that the compression afforded by matching matrices according to fewest disagreements also yields a compression of 4:1 and yet exhibits an average error rate of only 0.3125 vice 0.375 for the grill method. Why is the error rate reduced? Why, from an information theoretic viewpoint, does the latter method result in a decreased error rate for the same bit rate or compression? The answer

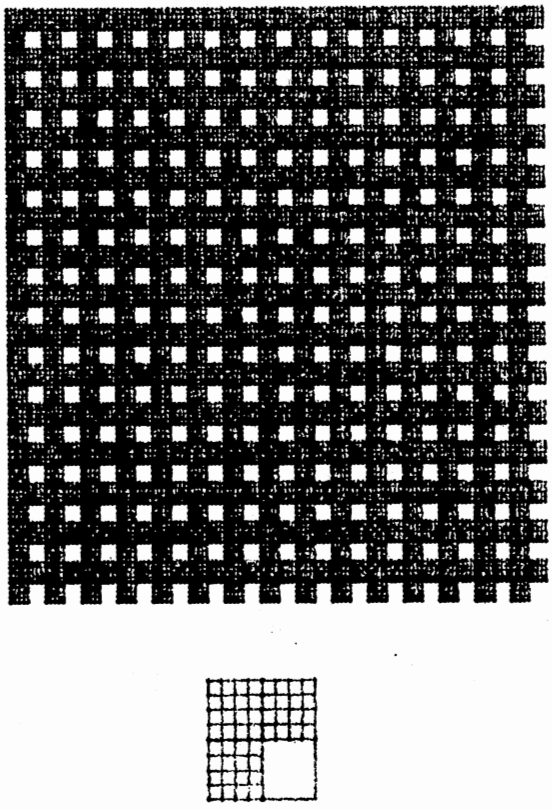


Figure 2. The Grill and One of Its Elements

All Possible 2x2 Frames	Disagreements Between Frame and $\begin{matrix}0&1\\1&0\end{matrix}$	Disagreements Between Frame and $\begin{matrix}1&0\\0&1\end{matrix}$
$\begin{matrix}0&0\\0&0\end{matrix}$	2	2
$\begin{matrix}0&0\\0&1\end{matrix}$	3	1
$\begin{matrix}0&0\\1&0\end{matrix}$	1	3
$\begin{matrix}0&0\\1&1\end{matrix}$	2	2
$\begin{matrix}0&1\\0&0\end{matrix}$	1	3
$\begin{matrix}0&1\\0&1\end{matrix}$	2	2
$\begin{matrix}0&1\\1&0\end{matrix}$	0	4
$\begin{matrix}0&1\\1&1\end{matrix}$	1	3
$\begin{matrix}1&0\\0&0\end{matrix}$	3	1
$\begin{matrix}1&0\\0&1\end{matrix}$	4	0
$\begin{matrix}1&0\\1&0\end{matrix}$	2	2
$\begin{matrix}1&0\\1&1\end{matrix}$	3	1
$\begin{matrix}1&1\\0&0\end{matrix}$	2	2
$\begin{matrix}1&1\\0&1\end{matrix}$	3	1
$\begin{matrix}1&1\\1&0\end{matrix}$	1	3
$\begin{matrix}1&1\\1&1\end{matrix}$	2	2

Figure 3. Codematrix Set (Eq. [1.8]) Disagreement Results

is subtle, intriguing, and important. It results from the way in which we use our single bit of information. Consider first the following gedanken experiments in probability theory:

EXPERIMENT A: An experimenter flips two unbiased coins. They land out of sight of the audience. The experimenter reports: "At least one of the coins shows a head."

EXPERIMENT B: An experimenter flips two unbiased coins. They land out of sight of the audience. The experimenter reports: "The coin on my left shows a head."

We now ask for the probability that both coins in Experiment A show heads and the same question regarding Experiment B.

The coins are unbiased and their flippings presumably uncorrelated. We know from the experimenter's reports that in both cases there is at least one head showing and we are perhaps tempted to believe that the probability that both coins show heads is one-half for both experiments. But let us consider what is implied by the experimenter's statements. Concerning Experiment A, all that the experimenter has told us is, that of the four possible cases:

Coin on Experimenter's Left	Coin on Experimenter's Right
H	H
H	T
T	H
T	T

only the last case is not possible. Thus, the probability that there are two heads showing in the outcome of Experiment A is one-third. Concerning Experiment B, the experimenter has told us that there are only two possible

cases: either the coin on his right shows a head or a tail. The outcome of the coin on his left is precisely known. Thus, the probability that there are two heads showing in Experiment B is one-half.

The above situation is somewhat analogous to our different coding methods. In the case of the grill, we are using our single bit of information to precisely specify one pixel. In the most closely matched matrix method, our single bit serves a more "global" function. It is spread equally over all the pixels and thus contributes little information regarding any single, specific pixel. This results in a more efficient, lower error rate coding. One further analogy is to the game of "twenty questions." If a player proceeds through a list of objects using the question: "Is it object i?" it is clear that he will be able to exhaust only twenty objects. If, however, he divides the objects into two equal size classes, discards the negative class, again divides the remaining class into two small, equal size classes, and so forth, he will be able with twenty questions, to identify a particular object out of a group of over one million entries.

1.4 An Example

Consider the 30 x 30 pixel two-tone "house" image shown in Figure 4. We will represent the house image by the method proposed in Equation (1.2). For our first experiment, we choose k = 3 and create the codematrix set shown in Figure 5. The first two members of the codematrix set, the all zero (all white) codematrix and the all ones (all black) codematrix, will, by convention, compose the first two entries of each codematrix set. The remaining six matrices were randomly derived by using a pseudorandom (deterministic) bit generator approximating a balanced binary

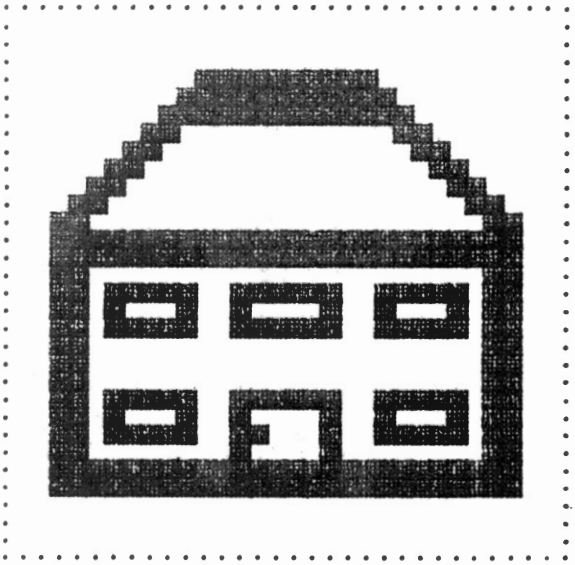


Figure 4. The 30x30 Pixel Two-Tone "House" Image

0 0 0	1 1 1
0 0 0	1 1 1
0 0 0	1 1 1
1 0 1	1 0 1
1 1 0	0 1 1
1 1 1	1 1 1
0 1 1	1 0 0
1 1 1	1 0 1
0 0 1	0 1 1
0 1 0	0 1 1
0 1 0	0 1 0
0 1 0	1 0 0

Figure 5. The Set of Eight Codematrices

Bernoulli source. As there are $k^2 = 9$ bits per frame and we rquire only $\log_2 8 = 3$ bits to represent a frame, we achieve a compression of 3:1.

Figure 6 shows the result of the above coding with the original image presented on the bottom for comparison. Note the periodic nature of the pixel structure especially on the edges. This is an example of the aliasing problem discussed in section 1.3.

To obviate the aliasing problem, we will prewhiten the image before coding as follows:

1. If the frame is all white or all black, no prewhitening is performed and the code for either the all-white or all-black frame is sent.

2. If the frame comprises a mixture of black and white pixels, then a spatially dependent, balanced, and pseudorandom (deterministic) binary bit stream is first added to the frame and then it is coded against the remaining codematrices. Upon reception, the pseudorandom stream is, of course, subtracted before presentation.

Figure 7 shows the result of performing encoding with the codematrices of Figure 5, but also using the prewhitening technique just described. Note that the aliasing problem is no longer apparent and, fortuitously, the error rate has been somewhat reduced.

Figures 8, 9, and 10 are the results of encoding (with prewhitening) using randomly derived codematrix sets successively augmented to 16, 32, and 64 members, respectively. Table I summarizes the results.

As an aside it should be noted that, for small frame dimensions, k, most images will contain many solid white/black frames. An improvement on the compression may sometimes be easily achieved by a simple encoding trick such as has been employed by Kunt and Johnsen [8]. Let:

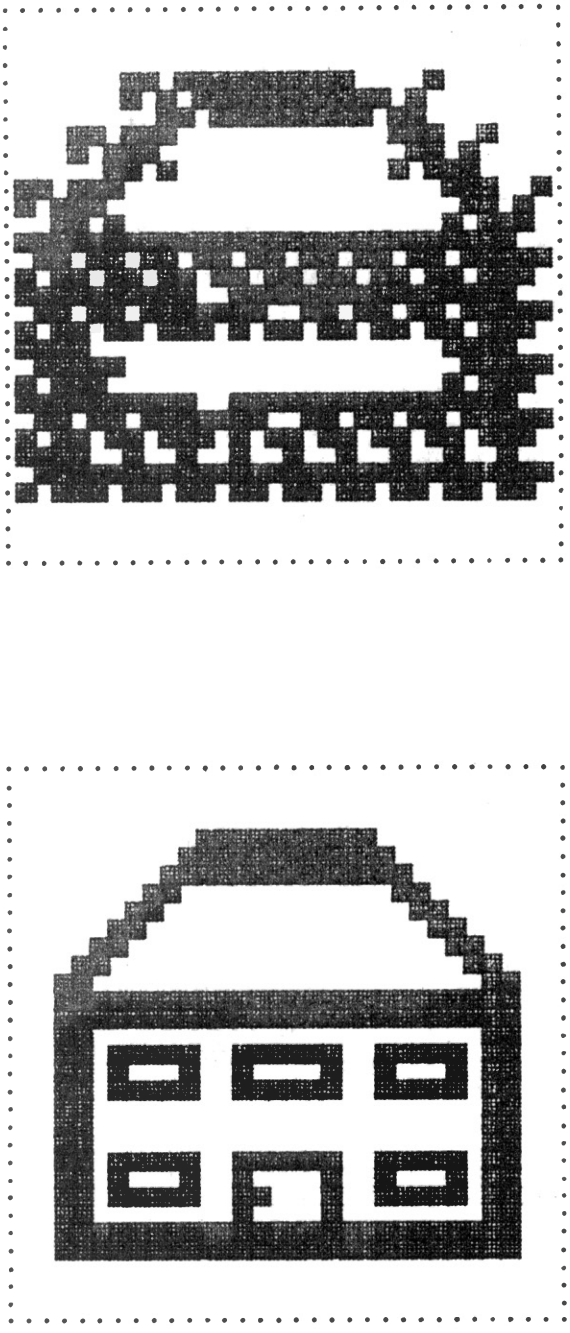


Figure 6. Aliasing, 3:1 Compression, No Prewhitening

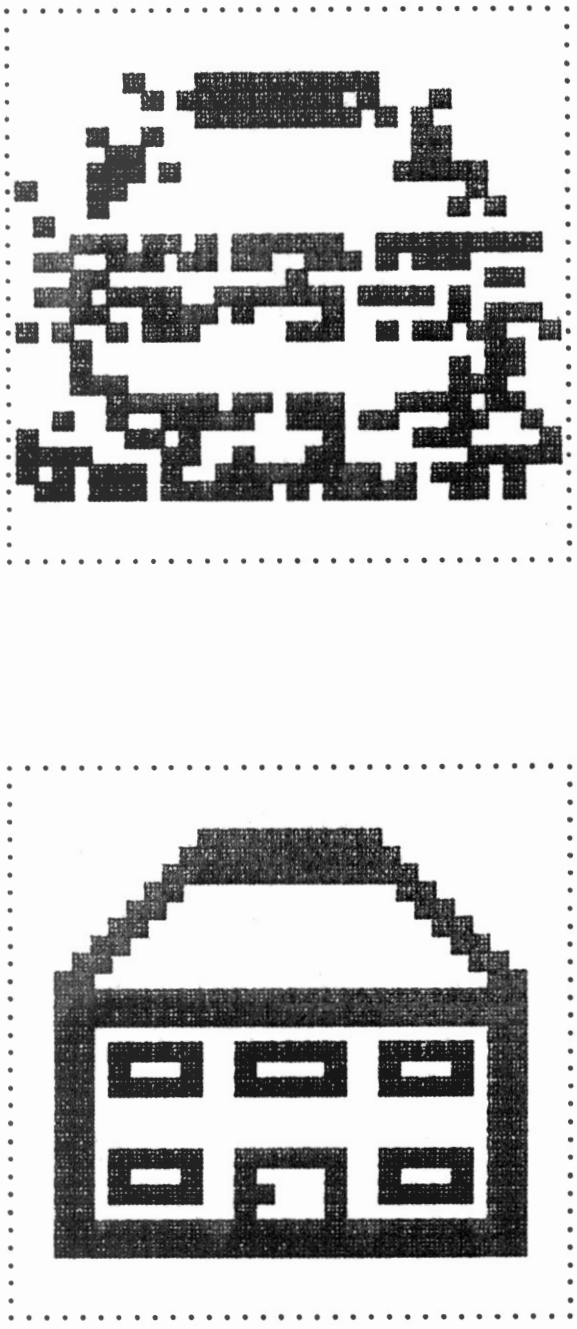


Figure 7. Prewhitening, 3:1 Compression

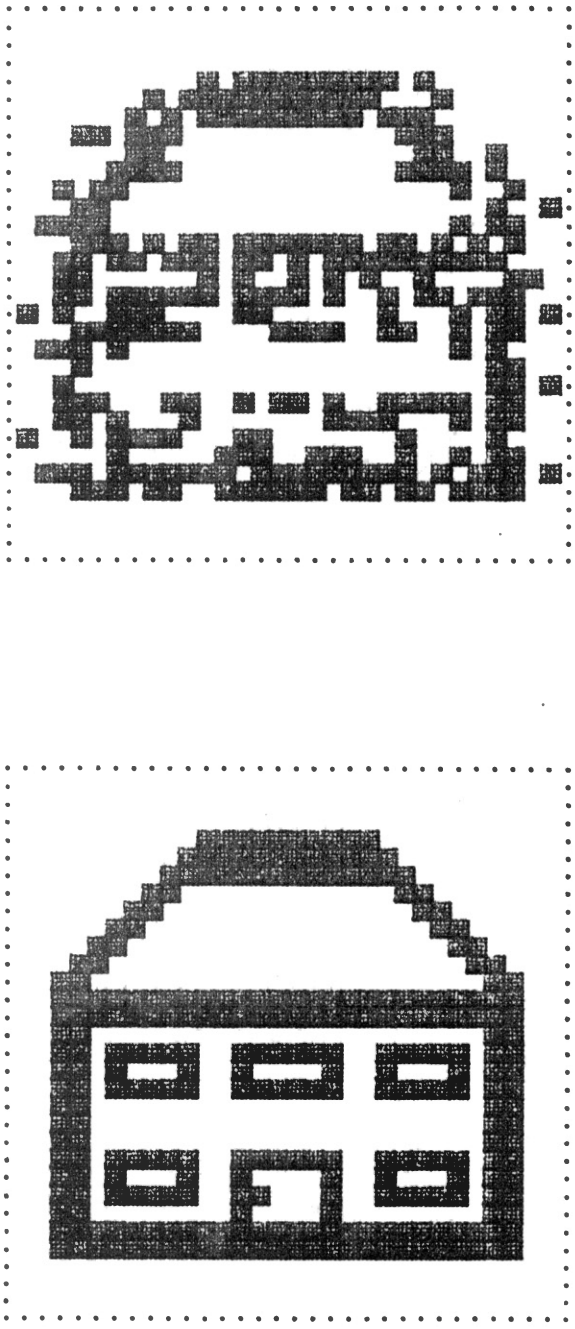
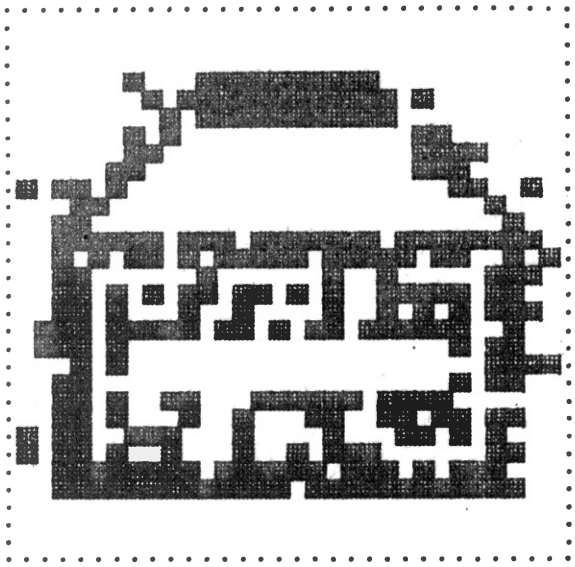


Figure 8. Prewhitening, 2.25:1 Compression



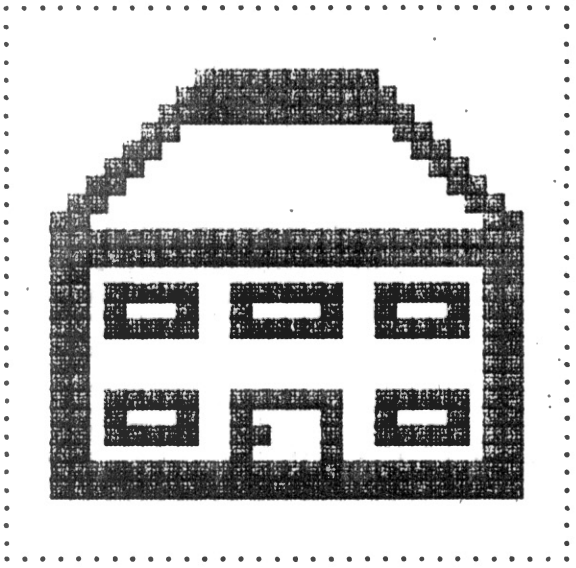
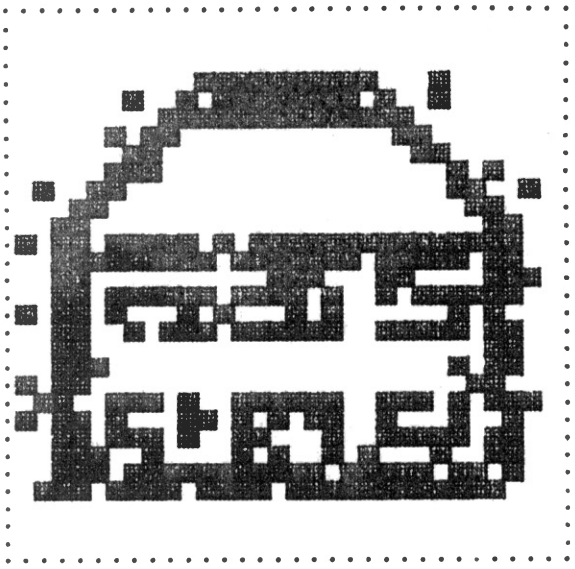


Figure 9. Prewhitening, 1.8:1 Compression



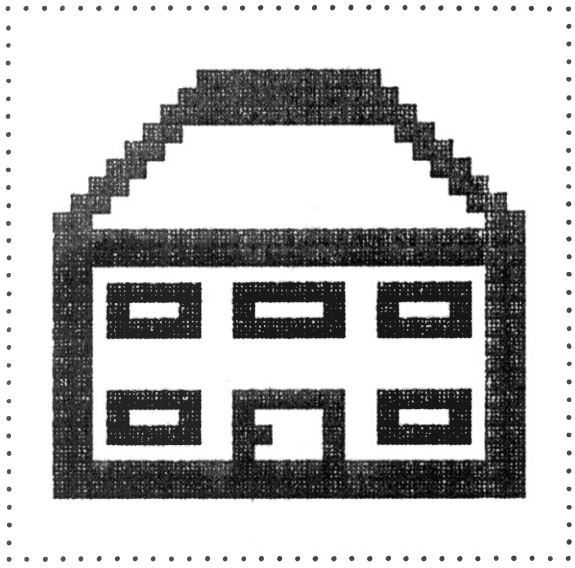


Figure 10. Prewhitening, 1.5:1 Compression

TABLE I

SUMMARY OF COMPRESSION RESULTS

Codematrix Set Size	Compression	Error Rate	Remarks
8	3:1	0.319	No Prewhitening
8	3:1	0.278	Prewhitening
16	2.25:1	0.230	Prewhitening
32	1.8:1	0.171	Prewhitening
64	1.5:1	0.131	Prewhitening

1. 00 represent a solid white frame (before prewhitening)

2. 01 represent a solid black frame (before prewhitening)

3. $1b_1b_2 \ldots b_m$ represent the codematrix (other than the all white and black codematrices) whose m-bit binary number is $b_1b_2 \ldots b_m$.

A binary stream coded with respect to the above convention is comma-free and the only special convention to be followed in decoding is that 00 and 01 indicate that no prewhitening was used for these frames and that the frames are solid white/black.

We have used an additional bit per frame description in the above encoding and therefore, if the method is to be an improvement, there must be a sufficient area of solid black and white frames to offset the compression loss incurred by the comma-free encoding of unequal word lengths. If s is the proportion of frames that are all black or all white, there will be an advantage to the above comma-free coding if

$$2s + (1 - s)[1 + \log_2 (R - 2)] < \log_2 R \, . \tag{1.9}$$

For large R ($R >> 1$), we can replace $R - 2$ with R and write the condition (Equation [1.9]) as

$$\left(\frac{R}{2}\right)^s > 2, \; R >> 1 \, . \tag{1.10}$$

1.5 Thesis Outline

This introductory chapter is followed by a chapter that derives closed form expressions both for the compression afforded by the codematrix method and for the size of the codematrix set. The best compression ratio possible by using this scheme is that ratio which obtains upon choosing the most efficient codematrix set. The best compression ratio

possible is examined with the aid of information theory and the results obtained.

The third chapter addresses implementation of a potential method for generating a large codematrix set through the use of finite field mathematics and matrix manipulations.

The fourth chapter is concerned with a Monte-Carlo experiment which validates the viability of the implementation philosophy set forth in Chapter III.

The fifth chapter is a set of statements concerning possible research tracts which would naturally follow this theoretical corpus. Especially promising is an idea put forward by the thesis adviser, Professor Rao Yarlagadda, that, in essence, proposes the melding of this "closest-codematrix-matching method" with standard redundancy removal techniques of image representation.

CHAPTER II

BENCHMARKS

2.1 Introduction

In this chapter we will obtain two sets of results. We will first derive a closed-form expression for the compression afforded by the random coding scheme introduced in the previous chapter. We will then proceed to calculate a theoretically best possible compression ratio via an information theoretic argument.

2.2 Purely Random Coding

Assume that the set of matrices with which we code the frames contains exactly R matrices. Let us call this set the codematrix set. Let us additionally assume that the codematrices are produced by a process that fills each of the k^2 positions of each codematrix with either a one or a zero. Further assume that the matrix-filling process comports to what Dynkin and Yushkevich [9] call the Markov principle: "that the future is independent of the past for a known present" (p. 1). Finally assume that ones and zeros are equally possible. The probability $p_a(i,j)$ that the bit in position i,j of the frame, f_{ij}, will agree with the bit in position i,j of the kth matrix of the codematrix set, n^k_{ij}, is

$$p(n^k_{ij}=0)\ p(f_{ij}=0) + p(n^k_{ij}=1)\ p(f_{ij}=1)$$

$$= \frac{1}{2}\, p(f_{ij}=0) + \frac{1}{2}\, p(f_{ij}=1)$$

$$= \frac{1}{2} [p(f_{ij} = 0) + p(f_{ij} = 1)] = \frac{1}{2} . \qquad (2.1)$$

Further, it is now clear that the conditional probability

$$p_a(i,j) | p_a(i',j') = p_a(i,j)$$

so long as $i \neq i'$ or else $j \neq j'$.

2.2.1 The Metric

We now wish to choose a metric, $d_\alpha = d_\alpha (N_\alpha, F)$, to define and measure a distance between F and the matrices of the codematrix set. A natural choice for d_α is the Hamming metric [10]. The Hamming metric considers F and the matrices of the codematrix set as points in k^2 space. Each component of the hyperpoint's location vector is either a one or a zero. The Hamming metric between F and any of the codematrices is computed as the number of coordinates in k^2 space in which they differ.

That this choice is an appropriate metric is easily shown by consideration of the requirements for a metric. Kowalsky, in his work [11], provides an excellently stated definition of a metric:

> A metric is an appropriate generalization of the intuitive notion of distance between two points. . . . a metric defined over the fundamental set R is a mapping : $R \times R \rightarrow R$ possessing the following properties:
>
> (M1) $\delta(p,q) \geq 0$ and $\delta(p,p) = 0$.
>
> (M2) $\delta(p,q) = \delta(q,p)$ (symmetry).
>
> (M3) $\delta(p,r) \leq \delta(p,q) + \delta(q,r)$ (triangle inequality).
>
> (M4 $\delta(p,q) = 0$ implies $p = q$ (pp. 50-51).

By inspection, the proposed metric satisfies M1, M2, and M4. To show that it also satisfies M3 requires only a simple argument. Consider that in computing the metric we compare the location vectors and sum disagreements of location components, component by component. As an example, consider

the two points in $k^2 = 4$ space: (1,0,0,0) and (1,0,1,1). We note that their third and fourth components differ and therefore the metric computes a distance of 2 between the two points. In general, if we have the three points p, q, and r and their location vectors: $p = (p_1, p_2, \ldots, p_{k^2})$, $q = (q_1, q_2, \ldots, q_{k^2})$, and $r = (r_1, r_2, \ldots, r_{k^2})$ we form the metrics specified in (M3) above by sequentially considering p_k, q_k, and r_k for each k. The following table lists the eight possibilities and computes the metrics specified in (M3).

TABLE II

EIGHT POSSIBILITIES AND COMPUTATION OF M3 METRICS

p_k	q_k	r_k	$\delta(p_k,q_k) + \delta(q_k,r_k)$	$\delta(p_k,r_k)$
0	0	0	0	0
0	0	1	1	1
0	1	0	2	0
0	1	1	1	1
1	0	0	1	1
1	0	1	2	0
1	1	0	1	1
1	1	1	0	0

We note that (M3) is satisfied for each triple p_k, q_k, r_k and is thus satisfied for the entire location vectors.

2.2.2 The Probability Density Function for the Metric

Having established that d_α is indeed a valid metric we now proceed to determine the probability distribution for $d_\alpha\ (N_\alpha, F)$. Consider again the probability density function $p_a\ (i,j)$. The properties it possesses show it to be equivalent to the classical Bernoulli density function. The metric $d_\alpha\ (N_\alpha, F)$ can then be viewed as the number of ones produced, in any order, from k^2 trials of $p_a\ (i,j)$. This description of the metric's probability density is the same description of the special case of the binomial density wherein the probability of a one is equal to the probability of a zero:

$$p_D\ (d_\alpha) = 2^{-k^2}\ \binom{k^2}{d_\alpha} \tag{2.2}$$

2.2.3 The Probability Density Function for the Minimum of the Metric

The next step is to investigate the behavior of the optimum match between F and our codematrix set of R matrices, $\{N_1 \ldots N_R\}$. To do this, consider that we construct the distance set $D = \{d_1 \ldots d_R\}$ as the set of Hamming distances from F to each of the N_i codematrices. The distances are placed into D with the natural ordering, i.e., distance d_k is the distance from the frame to N_k. Let $P_D(x)$ represent the probability that $\min \{d_1 \ldots d_R\} = d_\nu = x$. To calculate $P_D(x)$ we proceed as follows:

1. Let i_0 be the number of members in the set $\{d_1 \ldots d_R\}$ that are exactly zero. Similarly, let i_1 be the number that are exactly one. Generalizing, let i_k be the number that are exactly k.

REMARK I: If the maximum is to be x, then clearly $i_x > 0$ and $i_\ell = 0$ for all $\ell > x$.

REMARK II: $\sum_{k=0}^{x} i_k = R$.

2. The probability that i_0 <u>specific</u> members of $\{d_1 \ldots d_R\}$ will be equal to zero, i_1, <u>specific</u> members equal to one, . . . and i_x <u>specific</u> members equal to x is

$$p_D^{i_x}(x) \ldots p_D^{i_0}(0).$$

3. If we remove the above requirement that the set members be specifically assigned, then we see that there are

$$\frac{R!}{i_x! \ldots i_0!}$$

ways in which they can be chosen. Thus the probability that i_0 members of $\{d_1 \ldots d_R\}$ will be zero, i_1 members unity, and so on, is:

$$\frac{R!}{i_x! \ldots i_0!} p_D^{i_x}(x) \ldots p_D^{i_0}(0) . \tag{2.3}$$

REMARK III: The probability expression (2.3) is immediately recognized as the multinomial probability density function.

4. Thus the probability that the maximum of the set $\{d_1 \ldots d_R\}$ is exactly x, $p_D(x)$, is the discrete integral of the multinomial probability density function over the appropriate space, viz.:

$$p_D(x) = \sum_{i_x \ldots i_0} \frac{R!}{i_x! \ldots i_0!} p_D^{i_x}(x) \ldots p_D^{i_0}(0) \tag{2.4}$$

for all $i_x \ldots i_0$ such that

$$\text{(a)} \quad \sum_{k=0}^{x} i_k = R \tag{2.5a}$$

$$\text{(b)} \quad i_x > 0 \ . \tag{2.5b}$$

Unfortunately, the multinomial (2.3) is extremely difficult to integrate over the hyperspace specified by Equation (2.5a-2.5b). We are therefore either led to seek an approximation for $P_D(x)$ or to try an entirely different approach. It is appropriate at this juncture to review our near term goals before embarking on more mathematical pursuits. The first consideration is to recall that the compression, C, afforded is solely determined by R and shown, in Chapter I, to be

$$C = \frac{k^2}{\lceil \log_2 R \rceil}$$

The second consideration is that the moments of the distribution of the error rate

$$\frac{k^2 - d_\nu}{k^2} , \tag{2.6}$$

where

$$d_\nu = \min \{d_1 \ldots d_R\} \tag{2.7}$$

depends upon k and R. It thus seems reasonable (and, we hope, tractable) to express the compression as a function of k, R, and a requirement on the expectation, the first moment, of the error rate distribution.

Proceeding along this track we first set up a framework:

1. ERROR RATE: Let ε be an acceptable error rate. We define the first moment of the error rate distribution by stating that we expect $k^2\varepsilon$ errors. We desire, then, that $k^2 - d_\nu$ fall in the "tail" interval

$$[k^2(1-\varepsilon),\ k^2]. \tag{2.8}$$

2. The probability that a specific member of the codematrix set, α, exhibits a distance, d_α, such that $k^2 - d_\alpha$ falls in the tail interval Equation (2.8) is immediately found by (discretely) integrating Equation (2.2) over the tail interval. Let this probability be denoted by σ. We then have

$$\sigma = 2^{-k^2} \sum_{\ell=k^2(1-\varepsilon)}^{k^2} \binom{k^2}{\ell}. \tag{2.9}$$

3. Now let $\Phi_\Sigma(\sigma,R)$ be the probability that at least one of the d_i in the set $\{d_1 \ldots d_R\}$ causes $k^2 - d_i$ to fall within the tail interval. Because the codematrices have been generated independently of each other, we can write

$$\Phi_\Sigma(\sigma,R) = 1 - (1-\sigma)^R. \tag{2.10}$$

From Equation (2.10) we can determine R:

$$R = \frac{|\log [1 - \Phi_\Sigma(\sigma,R)]|}{|\log (1-\sigma)|} \tag{2.11}$$

The expression (2.11) may at first appear unorthodox because it yields R as a function of σ and Φ_Σ, the latter term appearing to be a function of R. But recall that $\Phi_\Sigma(\sigma,R)$ is a number that is fixed prior to the calculation.

4. Having determined R we can now compute the compression:

$$C \simeq \frac{k^2}{\log_2 R} = \frac{k^2}{\log_2 |\log [1 - \Phi_\Sigma(\sigma,R)]| - \log_2 |\log (1-\sigma)|}$$

$$= \frac{k^2 \log 2}{\log |\log [1 - \Phi_\Sigma(\sigma,R)]| - \log |\log (1-\sigma)|} \tag{2.12}$$

It is helpful to put the reasoning so far into the flowchart form of Figure 11.

It is clear that if we wish to study the behavior of R and C as a function of their parameters, we will have to acquire a suitable approximation to σ as a function of ε. The first thought that comes to mind is the normal or other standard approximations to the binomial. However, we are operating in the tail region and we must be careful. Consulting Fry [12] we find the caution:

> The normal distribution . . . is a fair approximation to the binomial distribution so long as y^3/σ [the cube of the standardized binomial variable divided by the standard deviation of the unstandardized binomial variable] is not too large. In the vicinity of the 'tails'--that is, when the deviations are large --it is never satisfactory (p. 232).

when using the binomial approximation when $p = 1/2$.

As it turns out, the approximation of the tail of a binomial distribution is a difficult problem to handle with a manageable (i.e., easily computable) form. Fortunately, Brockwell [13] has worked out such a form in the structure of an asymptotic expansion. Brockwell considers the sum

$$S_r(n) = \sum_{\ell=n}^{nr} \binom{nr}{\ell} p^{\ell} q^{nr-\ell} \tag{2.13}$$

where n and nr but not necessarily r are integers and $rp < 1$. Brockwell's approximation is

$$S_r(n) = \binom{nr}{n} p^n q^{nr-n} \left[\rho_0(p^{-1}) + \frac{\rho_1(p^{-1})}{n} + \ldots + \frac{\rho_k(p^{-1})}{n^k} + R_k\right] \tag{2.14}$$

where

$$\rho_0(y) = \frac{y-1}{y-r} \tag{2.15}$$

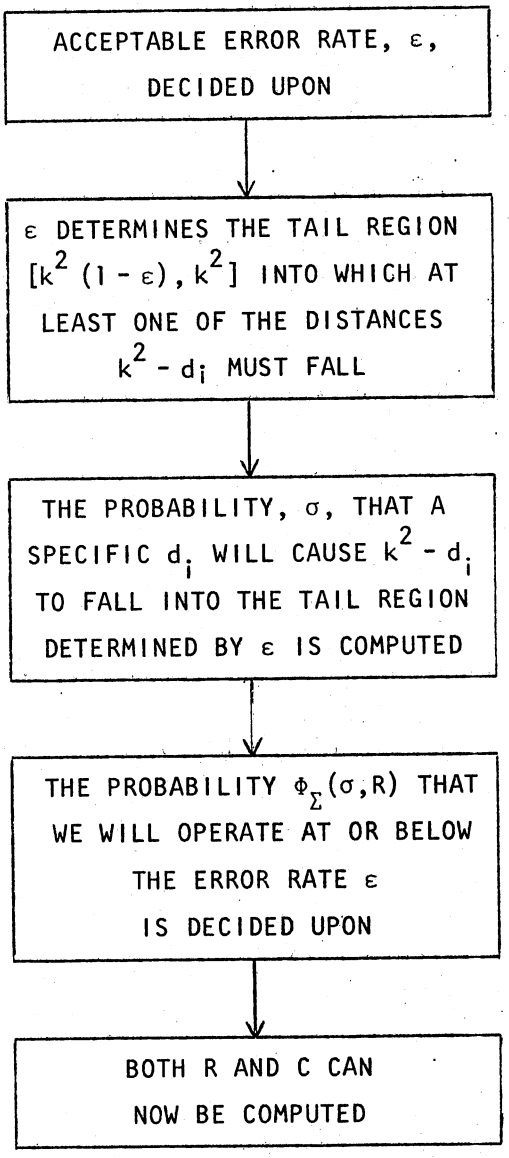


Figure 11. The Reasoning Process Used to Derive the Size of the Codematrix Set and the Afforded Compression

$$\rho_1(y) = \frac{y(y-1)}{y-r} \frac{d}{dy} \rho_0(y) \tag{2.16}$$

and

$$\rho_k(y) = \left(\frac{y(y-1)}{y-r} \frac{d}{dy} \right)^k \rho_0(y) \tag{2.17}$$

in general. The remainder term, R_k, in Equation (2.14) satisfies the following bound:

$$0 \leq (-1)^{k+1} R_k < \left| \frac{\rho_k(p^{-1})}{n^k} \right|. \tag{2.18}$$

Carrying out Brockwell's approximation to order $k = 1$, we find that for our case of interest, i.e., $p = q = 1/2$,

$$S_r(n) \simeq 2^{-nr} \binom{nr}{n} \left[\frac{1}{2-r} - \frac{\frac{2}{n}(r-1)}{(2-r)^3} \right]. \tag{2.19}$$

The maximum error, E_r, is, by Equation (2.18), easily shown to be

$$E_r = \frac{1}{\frac{n(2-r)^2}{2(r-1)} - 1}. \tag{2.20}$$

Examining Equation (2.20) we see that Brockwell's approximation can be an excellent one for tail regions beginning at large deviations from the mean. To convert Equations (2.19) and (2.20) into our notation we make the substitutions:

$$r \leftarrow \frac{1}{1-\varepsilon} \tag{2.21a}$$

$$n \leftarrow k^2 (1-\varepsilon) \tag{2.21b}$$

These substitutions give us the following expression for the first order approximation:

$$\sigma = 2^{-k^2} \binom{k^2}{k^2(1-\varepsilon)} \sum_{\ell=k^2(1-\varepsilon)}^{k^2} \binom{k^2}{\ell}$$

$$\simeq 2^{-k^2} \binom{k^2}{k^2(1-\varepsilon)} \left[\frac{1-\varepsilon}{1-2\varepsilon} - \frac{2\varepsilon(1-\varepsilon)}{k^2(1-2\varepsilon)^3} \right] \tag{2.22}$$

and

$$E_r = \frac{2\varepsilon}{k^2(1-2\varepsilon)^2 - 2\varepsilon} \tag{2.23}$$

Examining Equations (2.11) and (2.12) we find that both expressions contain the term $\log(1-\sigma)$. As $\sigma \ll 1$, we can approximate this factor by noting that $|\log(1-\sigma)| \simeq \sigma$ for small σ. A further approximation that is evidently needed is a tractable expression for the factor

$$\binom{k^2}{k^2(1-\varepsilon)} \tag{2.24}$$

that occurs in Equation (2.22).

Expanding Equation (2.24) we find that

$$\binom{k^2}{k^2(1-\varepsilon)} = \frac{k^2!}{[k^2(1-\varepsilon)]!\,[k^2\varepsilon]!} \tag{2.25}$$

To approximate Equation (2.25) we will need an approximation to the factorial. We choose the zeroth term approximation given by Stirling's series truncated to the first term, as described by Levy and Roth [14]:

$$n! \simeq \sqrt{2\pi}\, n^{n+\frac{1}{2}} e^{-n}. \tag{2.26}$$

Applying the above approximation to Equation (2.25) we obtain:

$$\binom{k^2}{k^2(1-\varepsilon)} \simeq \frac{1}{\sqrt{2\pi\varepsilon(1-\varepsilon)}\, k\, [(1-\varepsilon)^{1-\varepsilon}\, \varepsilon^{\varepsilon}]^{k^2}} \tag{2.27}$$

Substituting Equation (2.27) into Equation (2.22) we obtain:

$$\sigma \simeq \frac{1}{\sqrt{2\pi\varepsilon\,(1-\varepsilon)}\,k}\,[2(1-\varepsilon)^{1-\varepsilon}\,\varepsilon^{\varepsilon}]^{-k^2}\,\left[\frac{1-\varepsilon}{1-2\varepsilon} - \frac{2\,(1-\varepsilon)}{k^2\,(1-2\varepsilon)^3}\right] . \tag{2.28}$$

Checking Equation (2.28), we let $\varepsilon = 0.2$ and $k = 5$. From tables compiled by Harvard University [15], we find that $\sigma \simeq 0.00204$, the result of a direct computation to five decimal places. The approximation Equation (2.28) yields $\sigma \simeq 0.00205(294)$ which is quite respectable.

2.2.4 Closed Form Approximation for R and C

Using our approximation for σ, we can now determine an expression for the number of matrices required for the codematrix set, R:

$$R \simeq \frac{\sqrt{2\pi\varepsilon\,(1-\varepsilon)}\;\left|\log\left[1-\Phi_{\Sigma}(\sigma,R)\right]\right|}{\left[\frac{1-\varepsilon}{1-2\varepsilon} - \frac{2\varepsilon\,(1-\varepsilon)}{k^2\,(1-2\varepsilon)^3}\right]}\,k\,[2(1-\varepsilon)^{(1-\varepsilon)}(\varepsilon^{\varepsilon}]^{k^2} \tag{2.29}$$

In like manner, we can determine an expression for the compression, C:

$$C \simeq \frac{\log 2}{\frac{1}{k^2}\left\{\log\left|\log\left[1-\Phi_{\Sigma}(\sigma,R)\right]\right| + \log\sqrt{2\pi\varepsilon\,(1-\varepsilon)} - \log\left[\frac{1-\varepsilon}{1-2\varepsilon} - \frac{2\varepsilon\,(1-\varepsilon)}{k^2\,(1-2\varepsilon)^3}\right] + \log k\right\} + \log\left[2(1-\varepsilon)^{(1-\varepsilon)}(\varepsilon^{\varepsilon})\right]} \tag{2.30}$$

As

$$k \to \infty,\; R \to \sqrt{\frac{2\pi\varepsilon}{1-\varepsilon}}\,(1-2\varepsilon)\left|\log\left[1-\Phi_{\Sigma}(\sigma,R)\right]\right|\,k\,[2(1-\varepsilon)^{(1-\varepsilon)}(\varepsilon^{\varepsilon})]^{k^2} \tag{2.31}$$

and

$$C \to \frac{\log 2}{\log\left[2(1-\varepsilon)^{(1-\varepsilon)}(\varepsilon^{\varepsilon})\right]} \tag{2.32}$$

2.2.5 Analysis of C and R

It is an interesting property of C for large k that C is a function only of the acceptable error rate and not of the probability, $\Phi_{\Sigma}(\sigma,R)$, of achieving that error rate. We denote C for large k as C_{∞} which then serves as a convenient upper bound for the compression. It is useful to have a feel for C_{∞} as we will have to make engineering tradeoffs if we ever implement the system. For this reason we have plotted C_{∞} in Figure 12. Note that the initial gains in compression as we sacrifice fidelity are modest compared with the "avalanche" of increased compression that occurs in the higher error rate regions.

Considering R, we note that it depends on ε and the probability $\Phi_{\Sigma}(\sigma,R)$ for even large k. Let us look at its specific dependence on $\Phi_{\Sigma}(\sigma,R)$. We have, in Figure 13, plotted the factor $|\log[1-\Phi_{\Sigma}(\sigma,R)]|$ against $\Phi_{\Sigma}(\sigma,R)$ across a reasonable range of interest, $0.5 < \Phi_{\Sigma}(\sigma,R) < 0.95$. We note that R is fairly insensitive to $\Phi_{\Sigma}(\sigma,R)$ within this range. This is perhaps a bit surprising and deserves at least a passing analysis.

To best understand the behavior of R with respect to $\Phi_{\Sigma}(\sigma,R)$, we must return to the exact distribution in Equation (2.4), the integral of the multinomial distribution. Although it is computationally intractable to evaluate in general over the region specified, and thus derive $P_D(x)$, the probability density function of the minimum value contained in the set $\{d_1 \ldots d_R\}$, it is possible to compute nearly exact values for small R. Figure 14 shows the probability density function, $P_D(x)$, for the cases in which R = 1, 2, 3, 4, and 5 based on the binomial distribution with $k^2 = 9$.

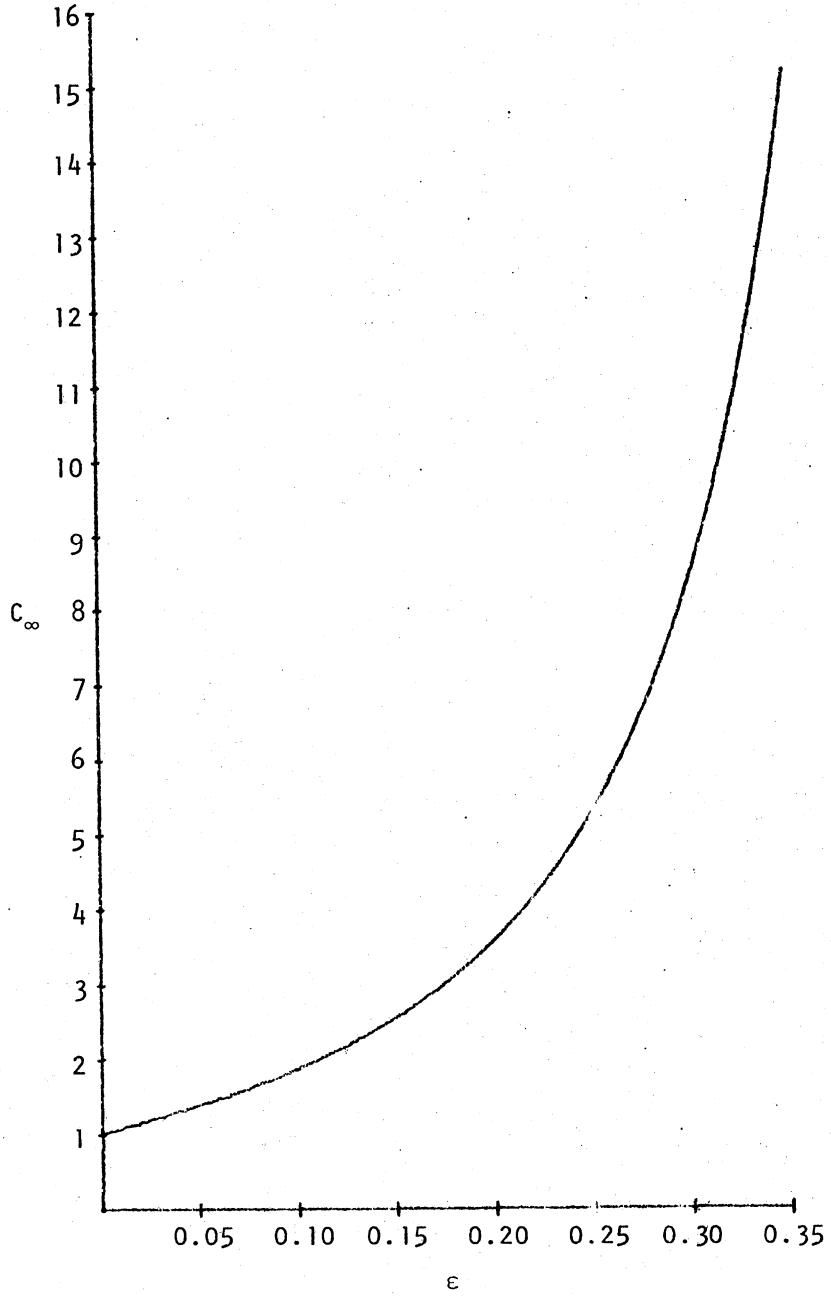


Figure 12. C_∞ Versus the Acceptable Error Rate ε

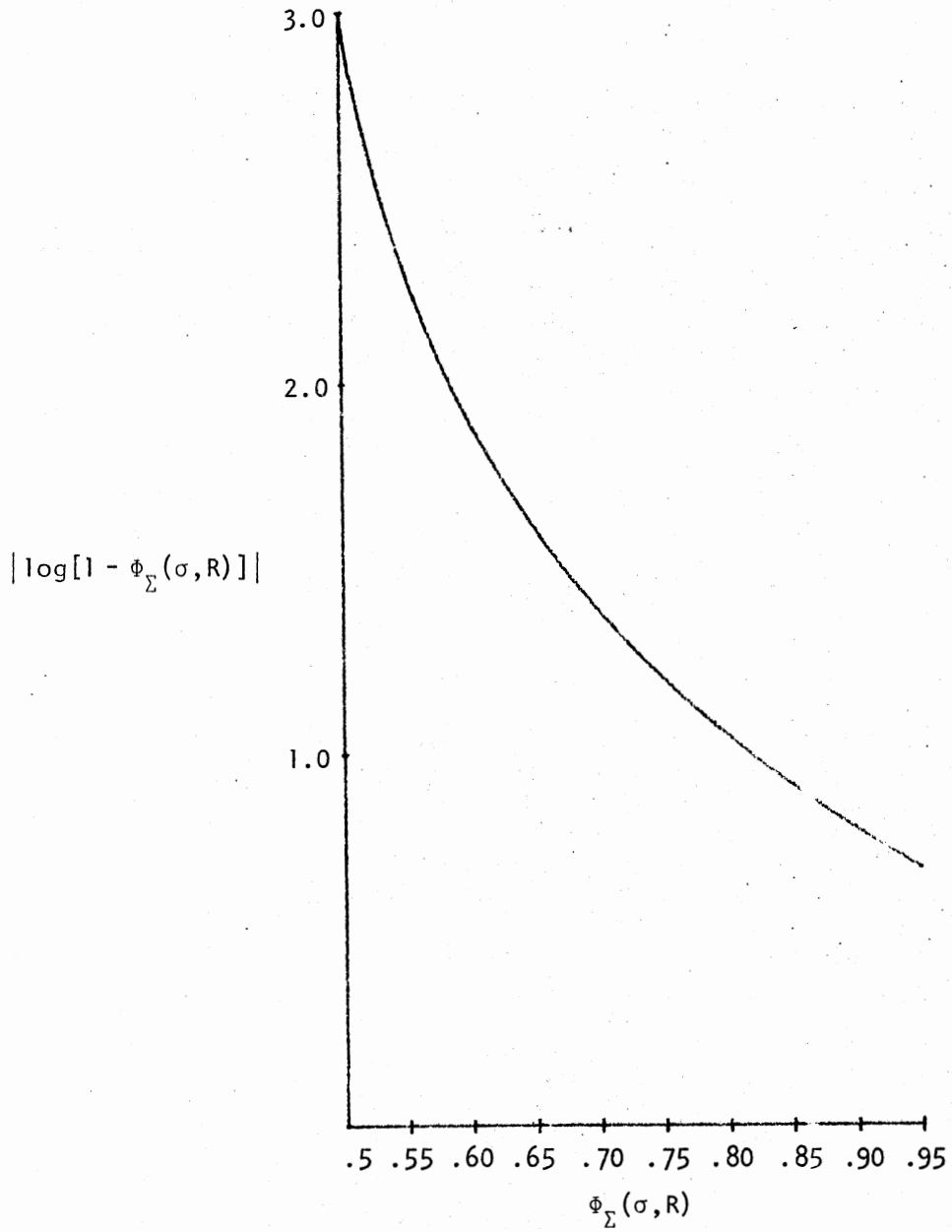


Figure 13. $|\log[1 - \Phi_{\Sigma}(\sigma,R)]|$ Versus $\Phi_{\Sigma}(\sigma,R)$

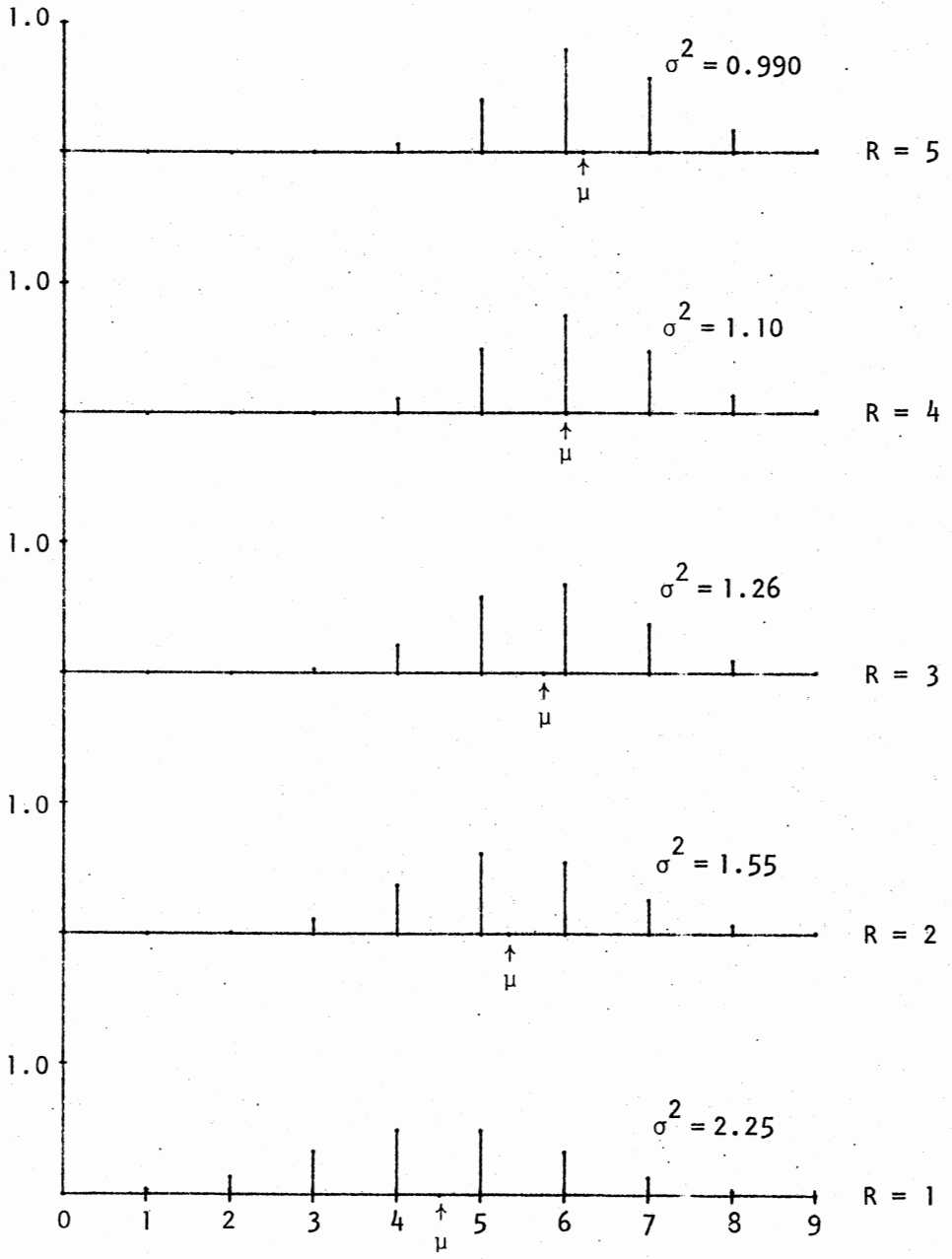


Figure 14. $P_D(x)$ for the First Five Values of R Showing the Migration of the Mean and Tightening of the Variance

For R = 1, $P_D(x)$ is, of course, the familiar binomial density function: $f_X(x) = 2^{-9}\binom{9}{x}$. Continuing on, we note two things from Figure 14. First, as R increases, the first moment, the mean, migrates to the right. This is intuitively expected. Second, as R increases the variance tightens up so that only a small range of values becomes probable. In light of this limited experiment it is not difficult to extend our intuition to larger k x k matrices and very large values of R where we expect a similar migration of the mean but, more importantly for the present discussion, also a tightening of the variance providing only a relatively small range of probable values for the maximum.

2.2.6 Simultaneous Consideration of R and C

Consider that we are operating with k x k matrices at an acceptable error rate and that we wish to increase our compression. We can do this in three ways. We can increase the error rate, we can increase k, or both. It is clear that increasing k will have an impact on computational time or computational complexity. This is so because of the great sensitivity of R to changes in k. If we try to more closely approximate C_∞ by holding the acceptable error rate constant, we may end up by driving R to a computationally infeasible magnitude. It is thus desirable to prepare a nomograph which will allow a quick evaluation of the choices. An example of such a nomograph is shown in Figure 15. The compression, C, is plotted vertically on the left end and the common logarithm of the number of matrices required for the codematrix set, R, is plotted vertically on the right. Both C and $\log_{10}R$ are plotted against the acceptable error rate, ε, for the presumed typical value of $\Phi_\Sigma(\sigma,R) = 0.9$.

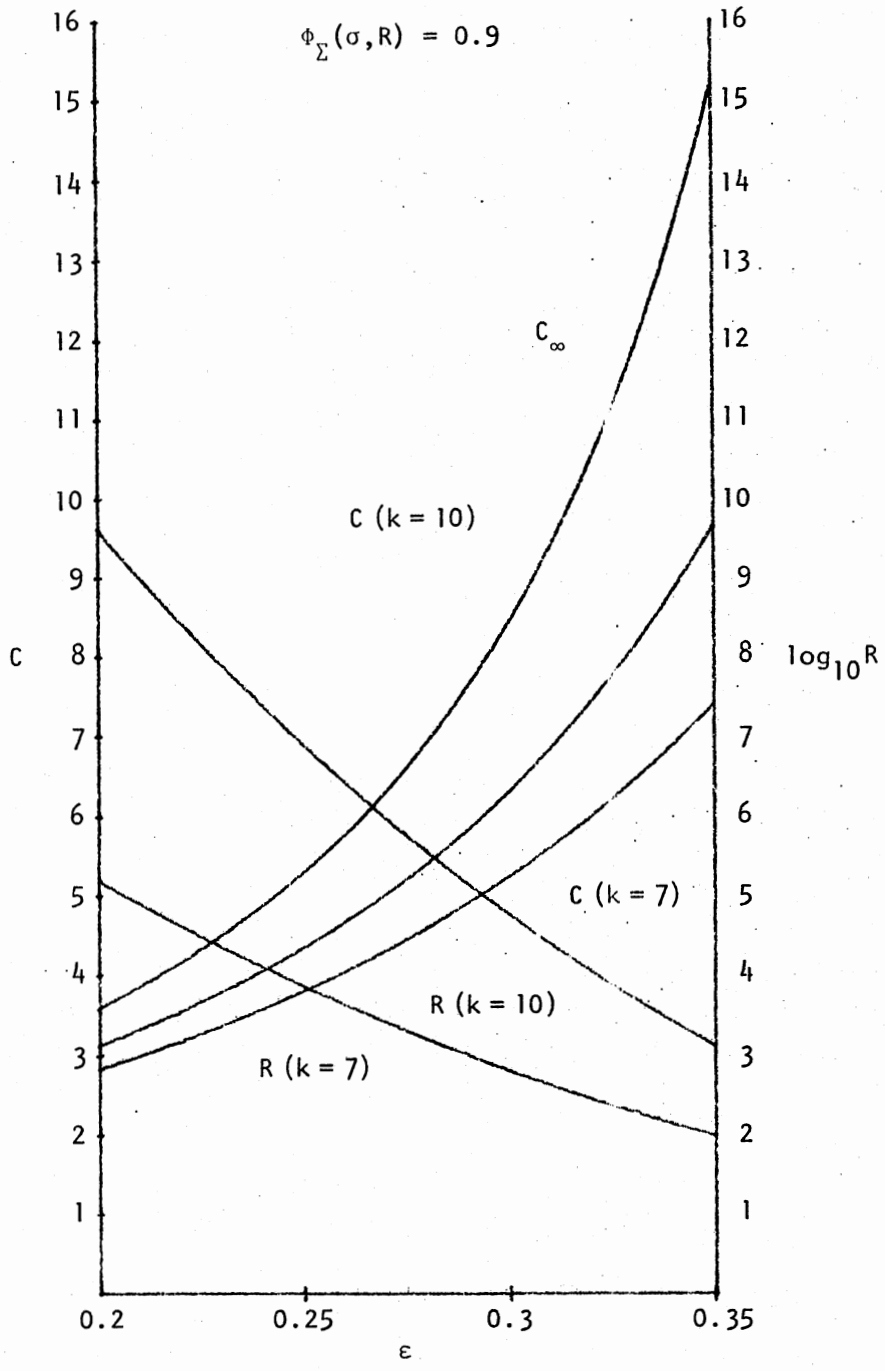


Figure 15. C and $\log_{10}R$ Plotted Against Acceptable Error Rate ε

2.3 An Upper Bound

So far we have considered only a codematrix set based on matrices derived from a balanced Bernoulli source. This seems perhaps unsophisticated and smacks of "brute force." Can we do better by a careful planning of our codematrix set?

This question is a reasonable one. Before giving it the analysis it deserves, let us consider a simple example along the lines of an experiment conducted in Chapter I. Suppose we are trying to encode 2x2 matrices using the following two 4-member codematrix sets, each of which provides a two-to-one compression:

SET A: $\begin{matrix}0&0\\0&1\end{matrix}$ $\begin{matrix}0&0\\1&0\end{matrix}$ $\begin{matrix}0&1\\0&0\end{matrix}$ $\begin{matrix}1&0\\0&0\end{matrix}$

SET B: $\begin{matrix}0&0\\0&0\end{matrix}$ $\begin{matrix}0&1\\1&0\end{matrix}$ $\begin{matrix}1&0\\0&1\end{matrix}$ $\begin{matrix}1&1\\1&1\end{matrix}$

Figure 16 shows the best results of the best match to all possible 2 x 2 frames for sets A and B. Assuming all frames are equally likely, the average error rate realized using codematrix set A is 0.28125 while that realized by codematrix set B is 0.25000. It is clear, then, that set B is superior. Set B was specially designed so that the four matrices were distributed in the four space so that no frame could exhibit a Hamming distance metric greater than 2. Notice that this is not so for codematrix set A. The all-unity frame $\begin{matrix}1&1\\1&1\end{matrix}$ is at a distance 3 from all of the codematrices.

It would seem, then, that for a given codematrix set size R there should be a best codematrix, or set of codematrices, that would generally surpass and only occasionally equal the performance of randomly generated sets.

All Possible 2x2 Frames	Set A Errors Resulting From Best Match	Set B Errors Resulting From Best Match
0 0 0 0	1	0
0 0 0 1	0	1
0 0 1 0	0	1
0 0 1 1	1	2
0 1 0 0	0	1
0 1 0 1	1	2
0 1 1 0	1	0
0 1 1 1	2	1
1 0 0 0	0	1
1 0 0 1	1	0
1 0 1 0	1	2
1 0 1 1	2	1
1 1 0 0	1	2
1 1 0 1	2	1
1 1 1 0	2	1
1 1 1 1	3	0

Figure 16. Results of Using Codematrix Sets A and B

The above supposition seems reasonable but the analysis is quite surprising. Consider that what we are doing by this coding scheme is achieving compression by representing a volume in hyperspace by a single point. The volume is, of course, all those frames which lie at a Hamming distance of $k^2\varepsilon$ from a point which is the closest codematrix. Diagrammatically, see Figure 17, we see that, in effect, we are chopping up the k^2 hypercube into a set of smaller volumes. There are efficient ways and inefficient ways in which this can be done. Codematrix set A, in the example just presented, is an example of an inefficient way. For codematrix set A to allow representation of all points in the hyperspace, we must allow a maximum distance of 3, else the all ones frame cannot be represented. Thus there is a great deal of overlap between the hypervolumes representable by the various codematrices. For example, frame $\begin{matrix}0 & 1\\ 1 & 1\end{matrix}$ is equally close, or distant, to three of the codematrices and maximally distant from the fourth.

Consider now that the coding were perfect. If so, the k^2 hypervolume, or frame space, would be equally partitioned into "hyperspheres" of radius $k^2\varepsilon$. The number of spheres, or messages, or codewords, or representations would then be

$$\frac{2^{k^2}}{\sum_{\ell=0}^{k^2\varepsilon} \binom{k^2}{\ell}} \tag{2.33}$$

where

$$\sum_{\ell=0}^{k^2\varepsilon} \binom{k^2}{\ell} \tag{2.34}$$

is the "volume" surrounding a point which would be represented exactly, i.e., a member of the codematrix set. (This is the same argument used by

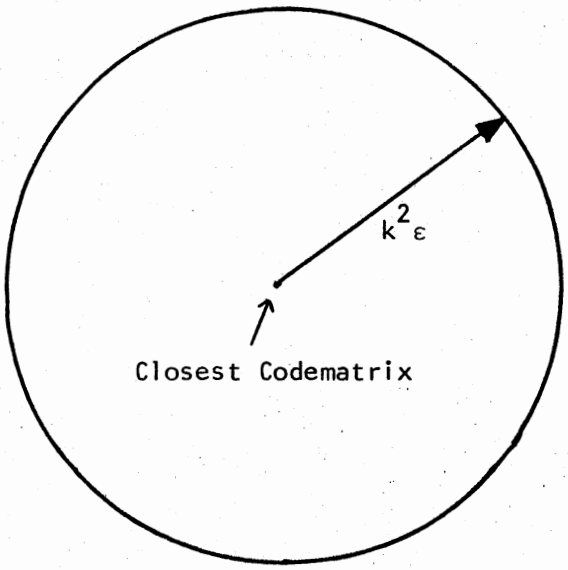


Figure 17. The Volume Encompassing All Frames Within a Hamming Distance of $k^2\varepsilon$ From the Closest Codematrix

Hamming in his paper on error-correcting codes (see Reference [10]). Given that the coding is perfectly efficient we use Equation (1.7) and determine the compression.

$$C = \frac{\log_2 (2^{k^2})}{\log_2 \left[\frac{2^{k^2}}{\sum_{\ell=0}^{k^2\varepsilon} \binom{k^2}{\ell}} \right]} = \frac{k^2}{k^2 - \log_2 \sum_{\ell=0}^{k^2\varepsilon} \binom{k^2\varepsilon}{\ell}}$$

$$= \frac{\log 2}{\log 2 - \frac{1}{k^2} \log \sum_{\ell=0}^{k^2\varepsilon} \binom{k^2\varepsilon}{\ell}} \tag{2.35}$$

Appealing to symmetry,

$$\sum_{\ell=0}^{k^2\varepsilon} \binom{k^2}{\ell} = \sum_{\ell=k^2(1-\ )}^{k^2} \binom{k^2}{\ell} . \tag{2.36}$$

Substituting Brockwell's approximation Equation (2.19) for Equation (2.36) we can write:

$$C = \frac{\log 2}{\log 2 - \frac{1}{k^2} \log \left(\frac{1}{\sqrt{2\pi\varepsilon(1-\varepsilon)}} k \left[\frac{1-\varepsilon}{1-2\varepsilon} - \frac{2\varepsilon(1-\varepsilon)}{k^2 (1-2\varepsilon)^3} \right] \right)}$$

$$+ \log \left[(1-\varepsilon)^{(1-\varepsilon)} (\varepsilon^{\varepsilon}) \right] \tag{2.37}$$

As $k \to \infty$ we see that

$$C \to \frac{\log 2}{\log 2 + \log \left[(1-\varepsilon)^{(1-\varepsilon)} (\varepsilon^{\varepsilon}) \right]} = \frac{\log 2}{\log \left[2(1-\varepsilon)^{(1-\varepsilon)} (\varepsilon^{\varepsilon}) \right]} . \tag{2.38}$$

But Equation (2.38) is the same as C as defined in Equation (2.32). We are thus led to a most interesting result that, in effect, says we need

not worry about trying to refine our codematrix structure for large k--a randomly derived codematrix set will exhibit behavior approaching the optimum codematrix set as k grows larger!

CHAPTER III

IMPLEMENTATION

3.1 Introduction

We now take up the question raised in section 1.2.3, the matter of devising a set of matrices that does not require an onerous amount of storage or computation time. Consider, again, that if we are to operate at a compression rate,

$$C = \frac{\log_2 (2^{k^2})}{\log_2 R} \tag{3.1}$$

then,

$$R = 2^{\frac{k^2}{C}} \tag{3.2}$$

and the number of bits that must be stored, unless generation is possible, is:

$$Rk^2 = k^2 \cdot 2^{\frac{k^2}{C}} \tag{3.3}$$

The following short table will impart to the reader the nature of the task via this approach for four frame sizes when the scheme is operating at a compression ratio C = 4 (see Table III). Clearly, for even moderate frame sizes, storage of all the matrices is impractical and we are naturally led to the important subquestion: "What families of matrices exist that (a)

can be generated from a relatively small amount of stored information, (b) can be generated quickly, and (c) possess at least some properties that would commend their use for this random, or better, pseudo-random coding scheme?"

TABLE III

BIT STORAGE REQUIREMENTS FOR DIFFERENT k FOR C = 4

Frame Dimension (k)	Storage Required/(Bits = $k^2 \cdot 2^{\frac{k^2}{C}}$
4	256
6	1.84×10^4
8	4.19×10^6
10	3.36×10^9
12	9.90×10^{12}
14	1.10×10^{17}
16	4.72×10^{21}

The above question suggests that we should look for our answer within formal mathematical structures, but where to start? Fortunately, a candidate answer is obvious, the theory of finite, or, Galois fields. Galois theory became important to the electrical engineering community in the late forties when memory elements and simple logical functions allowed the realization of the binary shift register with feedback provided by modulo two addition, the exclusive-or, of the contents of various stages. It was the synergism created by the melding of an emerging electronics technology

with a body of century-old mathematics, long regarded as a mathematical curiosity of little overt applicability outside the ken of abstract mathematicians, that brought forth one of the most important tools of modern day communications theory, the m-sequence.

3.2 Fields and Other Mathematical Paraphernalia

Before we consider finite fields, it is prudent to spend a few lines reviewing the concept of fields in general. We also introduce some ancillary concepts and terminology at this juncture.

3.2.1 Fields

Consider a set of elements $E = \{e_1, e_2, \ldots\}$ which may be either finite or infinite and an operation denoted by "+." If:

1. for any two elements, e_i, $e_j \in E$, $e_i + e_j \in E$,
2. for every e_i, e_j, and e_k in E, $e_i + (e_j + e_k) = (e_i + e_j) + e_k$,
3. there exists one and only one element in E, e_1 (the operation identity), such that for any element in E, $e_i + e_1 = e_i$,
4. for any e_i in E there exists one and only one element (the inverse) in E, e_j, such that $e_i + e_j = e_1$,

then we have a mathematical structure termed a GROUP. If, further, for every e_i and e_j in E, $e_i + e_j = e_j + e_i$, then the group is termed commutative or "abelian." All groups with which we shall work will be abelian.

An example of an infinite abelian group is the integers over addition. For this group the additive identity is 0 and the inverse of integer a is simply -a. Note, incidentally, that the integers do <u>not</u> form a group under subtraction because $e_i - (e_j - e_k) \neq (e_i - e_j) - e_k$.

If we introduce another commutative operation, "x," and require that the elements of the "+" group, excepting the "+" identity element, form a group under "x," and further require that for every e_i, e_j, and e_k in E, $e_i x (e_j + e_k) = (e_i x e_j) + (e_i x e_k)$, then we have a mathematical structure termed a FIELD.

An example of a field is the set of rational numbers over addition and multiplication. The additive identity is 0. The multiplicative identity is 1. The additive inverse of a is simply -a. The multiplicative inverse of a ($a \neq 0$) is 1/a.

3.2.2 Number Theoretic Operations and Functions

We define a set of elements $E = \{e_1, e_2, \ldots\}$. If elements $e_i = e_j + e_k \times e_\ell$, we note that e_k divides (is a factor of) $e_i - e_j$. We say that e_i is CONGRUENT to e_j modulo ("mod" for short) e_k. We denote this by the symbology $e_i \equiv e_j \bmod (e_k)$. As an example, all even (odd) integers are congruent to each other modulo 2.

The concept of relative primitivity is important. Two elements, a and b, are said to be RELATIVELY PRIME if they share no factors (excepting the multiplicative identity) in common. Thus, 6 and 35 are relatively prime even though neither number is itself a prime. A natural extension is the concept of GREATEST COMMON DIVISOR. The symbology c = (a,b) is defined over the positive integers as follows: "Integer c is the largest integer that can be divided without remainder into both integers a and b." If c = 1, then a and b are relatively prime.

A very important number theoretic function is the Euler totient or "phi" function denoted by ϕ. This function when applied to a positive integer m, $\phi(m)$, gives the count of the number of integers relatively

prime to m starting with 1 (which is relatively prime to all positive integers) and incrementing by 1 up to m. Thus, $\phi(6) = 2$ and $\phi(8) = 4$ for examples. For a prime, p, $\phi(p) = p - 1$. The function ϕ is said to be "weakly multiplicative." This means that $\phi(mn) = \phi(m)\phi(n)$ if $(m,n) = 1$, i.e., if m and n are relatively prime. One further result about ϕ is needed before it can be calculated for any positive integer and that is that $\phi(p^n) = p^{n-1}(p - 1)$ if p is prime. Thus, to calculate ϕ for any positive integer q, we proceed as follows:

1. Canonically decompose q into its (unique) product of primes, i.e.,

$$q = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r} .$$

2. Use the fact that ϕ is a weakly multiplicative function and write

$$\phi(p_1^{\alpha_1}) \phi(p_2^{\alpha_2}) \cdots \phi(p_r^{\alpha_r}) .$$

3. Sequentially evaluate all right-hand terms using the result that

$$\phi(p_k^{\alpha_k}) = p_k^{\alpha_k - 1} (p_k - 1) .$$

3.2.3 Finite Fields

Finite fields, fields with a finite number of elements, are possible if and only if the number of elements, N, is a power of a prime number, p, i.e., $N = p^n$ [16]. As we noted earlier, a field has two binary operators usually denoted by "+" and "x." As an example of a finite field in which the number of elements is a prime to the first power ($n = 1$), we consider $N = 3$. Let the set of elements be {A,B,C}. The following tables define the field operators:

+	A	B	C
A	A	B	C
B	B	C	A
C	C	A	B

x	A	B	C
A	A	A	A
B	A	B	C
C	A	C	B

The element A is clearly the additive identity as $A+e=e$, where $e \in \{A,B,C\}$. The element B is the multiplicative identity and $B \times e = e$.

As an example of a finite field in which the number of elements is a prime to other than the first power, we consider $N = 2^2 = 4$. Let the set of elements be {A,B,C,D}. The tables below define the field operators:

+	A	B	C	D
A	A	B	C	D
B	B	A	D	C
C	C	D	A	B
D	D	C	B	A

x	A	B	C	D
A	A	A	A	A
B	A	B	C	D
C	A	C	D	B
D	A	D	B	C

3.2.4 Finite Fields of Order 2^n Where $n > 1$

We now proceed to develop a practical framework for working with finite fields involving 2^n elements ($n > 1$).

Consider polynomials of degree $n - 1$ of the form

$$c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_1x + c_0 \qquad (3.4)$$

in which each of the coefficients is either a one or a zero, i.e., $c_i \in \{0,1\}$. We define addition of two polynomials $a_{n-1}x^{n-1} + \ldots + a_1x + a_0$ and $b_{n-1}x^{n-1} + \ldots + b_1x + b_0$ as a polynomial $r_{n-1}x^{n-1} + \ldots + r_1x + r_0$, where r_i is the modulo two sum of a_i and b_i. A little thought will show that there are 2^n polynomials possible of form (3.4) and that the set of

all these polynomials forms an abelian group under addition as just defined. We are thus "halfway" to a field structure and as the next logical step we ponder what will suffice for multiplication. Clearly normal multiplication of polynomials will not work, for if we consider the product $(a_{n-1}x^{n-1} + \ldots + a_1x + a_0)(b_{n-1}x^{n-1} + \ldots + b_1x + b_0) = a_{n-1}b_{n-1}x^{2n-2} + (a_{n-1}b_{n-2} + a_{n-2}b_{n-1})x^{2n-3} + \ldots + a_0b_0$ we note that the product is a polynomial of degree 2n-2. If we wish to retain multiplication in the above "normal" sense, we will have to manage it so that polynomials that exceed the n-1st degree will be mapped or reduced to polynomials of degree n - 1 or less. It is a remarkable result that such a mapping can be accomplished through the use of modular reduction based on a "primitive" polynomial.

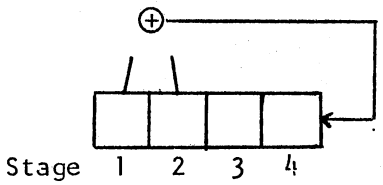
Like their counterparts, the integers, polynomials can be factored. For example, the polynomial $x^2 + 1$ cannot be factored over the field of polynomials with coefficients from the set of real numbers because two factors of the form $(x+a)(x+b) = x^2 + (a+b)x + ab$ cannot be found. However, over the field of polynomials with modulo two coefficients, $x^2 + 1$ can be factored; indeed, it is a perfect square, $x^2 + 1 = (x+1)^2$. If a polynomial is factorable into a product of polynomials of smaller degree, the polynomial is said to be REDUCIBLE. Polynomials that cannot be factored, such as $x^2 + x + 1$, are said to be IRREDUCIBLE. It is analogous to composite and prime numbers in the realm of integers. But this is where the analogy ends, for there is a further dichotomization to the set of of irreducible polynomials--those irreducible polynomials that are PRIMITIVE and those that are not.

Only a polynomial, P(x), that is primitive (and this is a practical way to define primitivity) can be used as a modulus of reduction so that

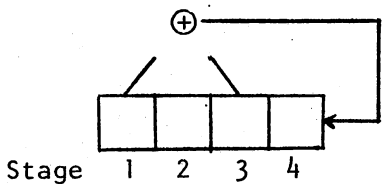
the set $\{0, \alpha, \alpha^2, \ldots, \alpha^{2^n-1}\}$ mod P (x) maps one-to-one onto the 2^n polynomials of form (3.4). The character α stands for a primitive element. One of the chief results of finite field theory is that there exists suitable P(x)'s and α's that allow construction of a finite field of 2^n elements.

3.3 A Finite Field Whose Members are Square Matrices

We have now come to the essential topic of the chapter. By way of introduction to this fascinating and important topic, let us try two experiments. We posit two left-shifting four-stage shift registers with feedback as shown in Figure 18. A momentary reflection regarding the properties of modulo two addition convinces us that if the shift registers are started with the contents of all their stages are set to zero, this set of contents or "state" will perpetuate itself as the registers are clocked. For this reason, we will avoid the "all-zero" state and start both registers from a non-all-zero state, say 0 0 0 1. A further consideration gives an immediate and important upper bound. Because there are a finite number of states that the register may assume, 2^n, and because the shift register with feedback is a deterministic sequential machine, the succession of states assumed over time is, or will become, a cycle. Further, as the all-zero state generates a "degenerate" cycle of length unity, it is clear that the longest cycle possible (although at this point not guaranteed achievable) is 2^n-1 steps in length. Let us now observe the results of the experiment in Figure 18. We see a marked difference in the behaviors of the two machines and only a very slight difference in their architectures, i.e., one feedback tap is taken from stage 2 for the machine



0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	1
0	0	1	1
0	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	0
1	0	0	0
0	0	0	1



0	0	0	1
0	0	1	0
0	1	0	1
1	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1

Figure 18. Two Four Stage Left-Shifting Shift Registers With Different Linear Feedbacks

on the left vice stage 3 for the machine on the right. We note that both machines exhibit non-transient states in the Markov sense (there are no lead-in tails to the cycle, i.e, all states exhibited are recurrent) and that the machine on the left exhibits a sequence of states that is of maximum length, an m-sequence for short.

How can we analyze the general shift register with linear feedback? The most direct approach would be through a state transition matrix. To construct this matrix we let the state vector $X^t = (x_1^t, x_2^t, \ldots)$, $x_i^t \varepsilon \{0,1\}$, represent the contents of stages one through n at time t. To obtain X^{t+1} we postmultiply X^t by M, where

$$M = \begin{pmatrix} 0 & 0 & \ldots & 0 & c_1 \\ \cdot & \cdot & \cdot & \cdot & c_2 \\ & & I_{n-1} & & \cdot \\ & & & & \cdot \\ & & & & \cdot \\ & & & & \cdot \\ & & & & c_n \end{pmatrix} \tag{3.5}$$

where I_{n-1} is the unit matrix of dimension n - 1 and $c_i \varepsilon \{c_1 \ldots c_n\}$ is one if state i is tapped for feedback and zero otherwise. We now have $X^{t+1} = X^t M$. For our previous example, M, for the lefthand, maximum length cycle producing, machine is

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.6}$$

Now that we have constructed the state transition matrix, how do we use it to analyze the cyclic behavior of the shift register with linear feedback?

> Each square matrix A has a determinant; though the determinant can be used in the elementary study of the rank of a matrix and in the solution of simultaneous linear equations, its most essential application in matrix theory is to the definition of the characteristic polynomial of a matrix (17, p. 299).

The statement above by Birkhoff takes on increased emphasis for us as we consider n x n matrices of elements taken from the finite field of two elements, for it is the characteristic polynomial that determines whether or not the n x n matrix in question will generate the non-zero elements of a finite field of 2^n elements.

Let us now consider the general state transition matrix M given in Equation (3.5). We note that it is in the form of a companion matrix and that $|M+\lambda I|$ will yield the characteristic polynomial which is

$$\lambda^n + c_n\lambda^{n-1} + \ldots + c_2\lambda + c_1 . \tag{3.7}$$

(Note that $M+\lambda I$ was written instead of $M-\lambda I$. Subtraction is not a field operator. Loosely, addition and subtraction are the same in GF(2).) The The characteristic equation is formed by setting the characteristic polynomial Equation (3.7) to zero:

$$\lambda^n + c_n\lambda^{n-1} + \ldots + c_2\lambda + c_1 = 0 \tag{3.8}$$

One magnificent result from matrix theory which will be needed later and is appropriate for introduction at this point is the Cayley-Hamilton theorem. Simply stated by Perlis [18, p. 136], "Every (square) matrix satisfies its characteristic equation." So, then, we know that

$$M^n + c_nM^{n-1} + \ldots + c_2M + c_1I = 0 \tag{3.9}$$

Note that Equation (3.9) guarantees (constructively) that powers of M greater than or equal to n can be expressed as a linear combination of

the set of matrices $\{M^{n-1}, M^{n-2}, \ldots, M, I\}$. Now, as we previously stated in section 3.2.4, if the set of matrices $\{I, M, M^2, \ldots, M^{2^n-1}\}$ are all distinct and $M^{2^n-1} = I$, then the characteristic polynomial is termed "primitive." It is one of the magnificent results of Galois theory that there exist primitive polynomials for $n \geq 3$, there being only one for the cases $n = 1$ and $n = 2$. Explicitly, there are $(1/n)\ \phi(2^n - 1)$ primitive polynomials of degree n, where ϕ is the Euler totient function [19].

There is only one field of 2^n elements. There may be more than one representation; however, all the fields are isomorphic [16]. For example, there are two primitive polynomials of degree 3, viz.,

$$\lambda^3 + \lambda + 1 \tag{3.10a}$$

$$\lambda^3 + \lambda^2 + 1 . \tag{3.10b}$$

Let us compute the characteristic equation of the general 3×3 matrix:

$$G = \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \tag{3.11}$$

If we take the determinant of $G + \lambda I_3$, we obtain

$$\begin{aligned} &\lambda^3 + \lambda^2 (e_{11} + e_{22} + e_{33}) \\ &\quad + \lambda(e_{11}e_{22} + e_{11}e_{33} + e_{22}e_{33} + e_{23}e_{32} + e_{12}e_{21} + e_{13}e_{31}) \\ &\quad + (e_{11}e_{22}e_{33} + e_{11}e_{23}e_{32} + e_{12}e_{21}e_{33} + e_{12}e_{23}e_{31} \\ &\quad + e_{13}e_{21}e_{32} + e_{13}e_{22}e_{31}) . \end{aligned} \tag{3.12}$$

Direct solution of the $\{e_{ij}\}$ for those cases yielding the polynomials in Equation (3.12) uncover no fewer than 48 distinct matrices which are arrayed in eight fields each with 2^3 members (each field contains I_3 and

0_3, the 3 x 3 multiplicative and additive identities, respectively, which do not, of course, possess a primitive characteristic polynomial). These eight fields are shown in Figure 19.

3.4 Implementations

The discovery that there are so many field generators for k x k matrix member fields is an indication that the family of matrices that can serve as generators of each other through exponentiation or through isomorphisms may be quite rich. Also, in a, for now, heuristic sense, these matrices, viewed as an ensemble, will exhibit several of the requisites for randomness. This claim, in effect, derives from the commonly held philosophy of using m-sequences as pseudo-random generators in simulation work as approximators of a balanced binary Bernoulli source. For example, each row (column) exhibits, pseudorandomly, all non-zero vectors under repeated exponentiation.

The isomorphisms of the basic Galois field can be of significant practical importance and should not be regarded as mere mathematical curiosities. For example, Warlick and the author have shown that a particular isomorphism class is of direct benefit to ultra-high-rate direct sequence spread spectrum systems [20]. The following from Berlekamp underscores this point:

> From an engineering standpoint, it is misleading to overstress the uniqueness of $GF(p^k)$, for this field may have many different representations. . . . The design and cost of circuitry to perform calculations in $GF(p^k)$ depend critically on the representation. For this reason, some engineers prefer to think of different representations of $GF(p^k)$ as different fields. This viewpoint is particularly justified in solutions where the cost of transforming from one representation to another is large [21, p. 104].

	Field A	Field B	Field C	Field D	Field E	Field F	Field G	Field H
	000	000	000	000	000	000	000	000
0_3	000	000	000	000	000	000	000	000
	000	000	000	000	000	000	000	000
	111	111	111	111	111	111	111	111
M	110	110	101	101	100	100	011	011
	100	011	100	011	110	101	110	101
	101	010	110	001	101	110	010	001
M^2	001	001	011	100	111	111	101	110
	111	101	111	110	011	010	100	010
	011	110	010	011	001	011	011	101
M^3	100	011	001	111	101	110	001	100
	101	100	110	010	010	100	111	011
	010	001	101	110	110	001	101	010
M^4	111	101	100	001	001	011	110	111
	011	111	010	101	100	111	010	110
	110	011	011	010	011	101	001	011
M^5	101	100	111	011	110	001	100	001
	010	010	101	100	111	110	011	100
	001	101	001	101	010	010	110	110
M^6	011	111	110	110	011	101	111	101
	110	110	011	111	101	011	101	111
	100	100	100	100	100	100	100	100
$M^7 = I_3$	010	010	010	010	010	010	010	010
	001	001	001	001	001	001	001	001

Figure 19. The Eight Fields of 3x3 Matrices With GF(2) Elements

One note of caution must be cited. If $2^n - 1$ is a (Mersenne) prime, all matrices, excepting the unit matrix, generated by a matrix whose characteristic polynomial is primitive can, themselves, serve as the field generator. If $2^n - 1$ is not prime, then this is not so. Let $2^n - 1$ be composite, i.e.,

$$2^n - 1 = pq \tag{3.13}$$

where p and q are integers satisfying $1 < p, q < 2^n - 1$. If M is a matrix whose characteristic polynomial is primitive, then the smallest non-zero integer, r, for which $M^r = I_n$ is $r = 2^n - 1$. Consider, however, M^p. It is clear that $(M^p)^q = M^{pq} = M^{2^n-1} = I$ and therefore M^p cannot generate the field, it is not a primitive element, or, equivalently, M^p does not possess a primitive characteristic polynomial. As an example, consider

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.14}$$

It is easily shown that M's characteristic polynomial is $\lambda^4 + \lambda + 1$ which is known to be primitive. Calculating $2^4 - 1 = 15$, we see that the period of the cycle generated by M is a composite number ($=3 \cdot 5$). From the discussion above, we know that M^3 and M^5 are not qualified generators of a maximum length cycle as they are not relatively prime to the period. To complete the example, we note, by direct calculation, that

$$M^5 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{3.15}$$

The characteristic polynomial of M^5 is easily shown to be

$$\lambda^4 + \lambda^2 + 1 . \tag{3.16}$$

Not only is Equation (3.16) not primitive, it is also reducible (a perfect square in fact):

$$\lambda^4 + \lambda^2 + 1 = (\lambda^2 + \lambda + 1)^2 . \tag{3.17}$$

3.5 A Key Theorem for Implementation

Consider the set of n x n matrices:

$$\{I, A, A^2, A^3, \ldots, A^{n-1}\} \tag{3.18}$$

If A possesses a characteristic polynomial that is primitive, then there will exist an independent set of α's such that

$$A^{n+i} = \alpha_1 A^{n+i-1} + \alpha_2 A^{n+i-2} + \ldots + \alpha_n A^i \tag{3.19}$$

where $\alpha_n = 1$. Notice that the form (3.19) requires that each matrix element of successive powers of A will exhibit the same sequence. This sequence will, in fact, be the same sequence exhibited by a shift register with linear feedback as shown, for example, in Figure 18. We call this sequence the inherent linear m-sequence. It is known that if an m-sequence is term-by-term added to a phase shift of itself, the resulting sequence will be yet another phase shift of the m-sequence. We also know that the n m-sequences that describe any column or row of successive powers of A will be independent of each other, i.e., no term-by-term sums of up to n - 1 of any of the row or column sequences will yield the nth remaining row or column sequence. These observations immediately yield the following theorem. Every power of A is expressible as the following matrix,

i.e., the following form is invariant over exponentiation:

$$\begin{pmatrix} m_1 & m_2 & \cdots & m_n \\ L_{2,1}(\underline{m}) & L_{2,2}(\underline{m}) & \cdots & L_{2,n}(\underline{m}) \\ \vdots & \vdots & & \vdots \\ L_{n-1,1}(\underline{m}) & L_{n-1,2}(\underline{m}) & \cdots & L_{n-1,n}(\underline{m}) \end{pmatrix} \tag{3.20}$$

The $m_1 \varepsilon \{0,1\}$. The $L_{i,j}(\underline{m})$ are linear combinations of $m_1, m_2, \ldots, m_n$.

Consider the field structure shown in Figure 20. By some elementary linear equations, the general form (3.20) of the matrices in the field is easily shown to be

$$\begin{pmatrix} m_1 & m_2 & m_3 \\ m_3 & m_1 & m_2 + m_3 \\ m_2 + m_3 & m_3 & m_1 + m_2 + m_3 \end{pmatrix} \tag{3.21}$$

Notice that all matrices of the field shown in Figure 20 can be derived by letting (m_1, m_2, m_3) assume all possible $2^3 = 8$ binary triples.

3.6 Other Matrices--Similarity Transformations

Recall that R, the number of matrices in the codematrix set, may be very large depending upon its parameters. It may be that R far exceeds $2^n - 1$, or the number of matrices that can be generated by a field generator. How then can we generate others? Fortunately there is a result from the mathematics of matrices that is extremely helpful. This result is the similarity transformation. The theorem, see Reference [22] for example, is as follows. Let P be any non-singular n x n square matrix and P^{-1} its inverse. If M is a non-singular square n x n matrix whose characteristic polynomial is $f(\lambda)$, then the n x n square matrix

$$0_3 = \begin{matrix}000\\000\\000\end{matrix}$$

$$M = \begin{matrix}111\\110\\011\end{matrix}$$

$$M^2 = \begin{matrix}010\\001\\101\end{matrix}$$

$$M^3 = \begin{matrix}110\\011\\100\end{matrix}$$

$$M^4 = \begin{matrix}001\\101\\111\end{matrix}$$

$$M^5 = \begin{matrix}011\\100\\010\end{matrix}$$

$$M^6 = \begin{matrix}101\\111\\110\end{matrix}$$

$$M^7 = I_3 = \begin{matrix}100\\010\\001\end{matrix}$$

Figure 20. The Eight Field Elements of Field B of Figure 19

$$PMP^{-1} \tag{3.22}$$

also possesses the characteristic polynomial $f(\lambda)$. The form (3.22) is called a similarity transformation of M.

Similarity transformations seem to offer a way of easily achieving some of the isomorphic field structures. But which similarity transformations, or classes of transformations, should be used and how hard are they to implement?

3.7 The Permutation Similarity Transformation

One particularly simple similarity transformation can be built from the simple concept of a permutation of rows and columns. It is well known that premultiplication by a permutation matrix swaps rows and postmultiplication swaps columns. A permutation matrix can be written either as FORM I:

$$\begin{matrix} \delta(p_1,1) & \delta(p_1,2) & \dots & \delta(p_1,n) \\ \delta(p_2,1) & \delta(p_2,2) & \dots & \delta(p_2,n) \\ \vdots & \vdots & & \vdots \\ \delta(p_n,1) & \delta(p_n,2) & \dots & \delta(p_n,n) \end{matrix} \tag{3.23}$$

or as FORM II:

$$\begin{matrix} \delta(1,c_1) & \delta(1,c_2) & \dots & \delta(1,c_n) \\ \delta(2,c_1) & \delta(2,c_2) & \dots & \delta(2,c_n) \\ \vdots & \vdots & & \vdots \\ \delta(n,c_1) & \delta(n,c_2) & \dots & \delta(n,c_n) \end{matrix} \tag{3.24}$$

where $\delta(i,j)$ is the Kronecker symbol and defined in the usual way:

$$\delta(i,j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3.25}$$

Either FORM I or FORM II may be used to specify a permutation matrix since permutation matrices are doubly stochastic. If we require that a matrix of FORM I have as its inverse a matrix of FORM II, then we must satisfy the n-equations:

$$\begin{aligned}
&\delta(p_1,1)\delta(1,c_1) + \delta(p_1,2)\delta(2,c_1) + \ldots + \delta(p_1,n)\delta(n,c_1) = 1 \\
&\delta(p_2,1)\delta(1,c_2) + \delta(p_2,2)\delta(2,c_2) + \ldots + \delta(p_2,n)\delta(n,c_2) = 1 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\delta(p_n,1)\delta(1,c_n) + \delta(p_n,2)\delta(2,c_n) + \ldots + \delta(p_n,n)\delta(n,c_n) = 1
\end{aligned} \tag{3.26}$$

Examination of Equation (3.26) convinces us that $\delta(p_i,j) = \delta(j,c_i)$ for all i and j, and thus P^T is the inverse of P.

Because the inverse of a permutation matrix is merely its transpose, a similarity transformation of the form

$$PMP^T \tag{3.27}$$

where P is a permutation matrix can be electronically implemented easily and very quickly. Let us study the behavior of the six transforms of the form (3.27) on the generator matrix of field family B of Figure 19. The results are shown in Figure 20. Note that six different field representations are created by application of the six transformations on the single generator matrix. It thus seems reasonable to ask the question: "When will all n! similarity transformations of the form (3.27) yield n! different field representations?"

Fortunately this question can be answered very simply. By way of introduction, consider again the field representation shown in Figure 20. The generator matrix is:

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{3.28}$$

We see in Figure 21 that the similarity transformation

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} M \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{3.29}$$

leads to a generator of the C field representation. Now consider the general form (3.20) of the field representation generated by M. It is

$$\begin{pmatrix} m_1 & m_2 & m_3 \\ m_3 & m_1 & m_2 + m_3 \\ m_2 + m_3 & m_3 & m_1 + m_2 + m_3 \end{pmatrix} \tag{3.30}$$

Applying the similarity transformation shown in Equation (3.29) to Equation (3.30), we obtain:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} m_1 & m_2 & m_3 \\ m_3 & m_1 & m_2 + m_3 \\ m_2 + m_3 & m_3 & m_1 + m_2 + m_3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} m_1 + m_2 + m_3 & m_3 & m_2 + m_3 \\ m_2 + m_3 & m_1 & m_3 \\ m_3 & m_2 & m_1 \end{pmatrix} \tag{3.31}$$

Note that $m_1 + m_2 + m_3$, m_3, and $m_2 + m_3$, the top row elements of Equation

$$\begin{pmatrix}100\\010\\001\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}100\\010\\001\end{pmatrix}=\begin{pmatrix}111\\110\\011\end{pmatrix}$$ Member of FIELD B

$$\begin{pmatrix}100\\001\\010\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}100\\001\\010\end{pmatrix}=\begin{pmatrix}001\\110\\010\end{pmatrix}$$ Member of FIELD H

$$\begin{pmatrix}001\\010\\100\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}001\\010\\100\end{pmatrix}=\begin{pmatrix}110\\011\\111\end{pmatrix}$$ Member of FIELD C

$$\begin{pmatrix}010\\100\\001\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}010\\100\\001\end{pmatrix}=\begin{pmatrix}110\\111\\101\end{pmatrix}$$ Member of FIELD G

$$\begin{pmatrix}001\\100\\010\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}010\\001\\100\end{pmatrix}=\begin{pmatrix}101\\111\\011\end{pmatrix}$$ Member of FIELD E

$$\begin{pmatrix}010\\001\\100\end{pmatrix}\begin{pmatrix}111\\110\\011\end{pmatrix}\begin{pmatrix}001\\100\\010\end{pmatrix}=\begin{pmatrix}101\\110\\111\end{pmatrix}$$ Member of FIELD D

Figure 21. The Results of Applying the Six Similarity Transforms to the FIELD B Generator Matrix

(3.31) are independent, as expected. Let us form a new basis, $\hat{m}_1, \hat{m}_2, \hat{m}_3$, such that

$$\hat{m}_1 = m_1 + m_2 + m_3 \tag{3.32a}$$

$$\hat{m}_2 = m_3 \tag{3.32b}$$

$$\hat{m}_3 = m_2 + m_3 \tag{3.32c}$$

Solving Equation (3.32) for m_1, m_2, and m_3, we obtain

$$\begin{pmatrix} \hat{m}_1 & \hat{m}_2 & \hat{m}_3 \\ \hat{m}_3 & \hat{m}_1 + \hat{m}_3 & \hat{m}_2 \\ \hat{m}_2 & \hat{m}_2 + \hat{m}_3 & \hat{m}_1 + \hat{m}_3 \end{pmatrix} \tag{3.33}$$

It is easily verified that Equation (3.33) is the general form of the field representation of FIELD C.

We thus can state that all n! similarity transformations of the form (3.27) will result in different field representations when the general form of M (Equation (3.20)) contains no symmetries.

If a generator matrix can be found whose general form contains no symmetries, then similarity transformations of the form (3.27) will be capable of generating

$$n!(2^n - 1) \tag{3.34}$$

different n x n matrices. The following will impart to the reader just how rapid the growth of Equation (3.34) is with respect to n (see Table IV).

3.8 Implementing the Codematrix Library

We have just seen that the permutation similarity transformation is

TABLE IV

GROWTH OF EQUATION (3.34)
WITH RESPECT TO n

n	$n!(2^n - 1)$
3	42
4	360
5	3720
6	45,360
7	640,080
8	10,281,600
9	1.854×10^8
10	3.712×10^9

an excellent candidate for helping to generate a codematrix family of $n!(2^n - 1)$ members. A permutation matrix is desirable from an electronic implementation point of view because its inverse, which is demanded by the similarity transformation, is simply its transpose. How then can we generate permutations?

There are many schemes available for generating permutations. One particular method of great interest is due to D. H. Lehmer and is often referred to as Lehmer's lexicographical method [23]. We shall describe this system by means of an example which is presented not in the hope that we will be mathematically rigorous, but rather in the hope that we show it as a procedure which can be quickly understood, generalized by the reader to other n and fruitfully applied.

Every non-negative integer, n, can be uniquely expressed in the "factorial number system," i.e.,

$$n = d_1 \cdot 1! + d_2 \cdot 2! + d_3 \cdot 3! + \ldots + d_k \cdot k! + \ldots \qquad (3.35)$$

where each d_j in Equation (3.35) satisfies

$$0 \leq d_j \leq j \qquad (3.36)$$

The n! permutations of the first integers (starting from 0) may be lexicographically ordered. Table V exhibits the lexicographical ordering of the 24 permutations of the integers 0, 1, 2, and 3. To specify a particular permutation, a number p is chosen such that $0 \leq p \leq 23$. The "factorial digits" of p are computed and arrayed from least significant to most significant along a 45° diagonal. The factorial digits of p are prefaced with an additional 0. This is shown in Figure 22 for p = 3. The isosceles right triangle is filled in by proceeding from left to right from the

TABLE V

THE 24 PERMUTATIONS OF 0123
ORDERED LEXICOGRAPHICALLY

Permutation Number	Permutation
0	0 1 2 3
1	0 1 3 2
2	0 2 1 3
3	0 2 3 1
4	0 3 1 2
5	0 3 2 1
6	1 0 2 3
7	1 0 3 2
8	1 2 0 3
9	1 2 3 0
10	1 3 0 2
11	1 3 2 0
12	2 0 1 3
13	2 0 3 1
14	2 1 0 3
15	2 1 3 0
16	2 3 0 1
17	2 3 1 0
18	3 0 1 2
19	3 0 2 1
20	3 1 0 2
21	3 1 2 0
22	3 2 0 1
23	3 2 1 0

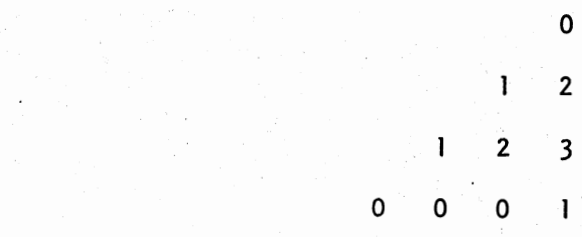


Figure 22. Computation of the Fourth Permutation of 0123

diagonal of factorial digits according to the rule: "If the column header, the factorial digit at the head of the column being filled in during the left-to-right progression, is greater than the integer which has just been filled in or occupies the position to the immediate left of the column, the integer is copied as is; otherwise, it is increased by one." Note that the triangle of Figure 22 has been filled in according to this rule and that the resulting permutation (the fourth permutation per our example) is derived by reading down the right-most column. For our example, we obtain 0231.

CHAPTER IV

AN EXAMPLE

4.1 Introduction

In an attempt to see whether or not the implementation method of Chapter III would produce a compression versus acceptable error rate as predicted for a kxk frame, a Monte-Carlo experiment was conducted. Because a general purpose digital computer is not amenable to storing a large codematrix set or modeling the great number of computations that could be quickly and efficiently performed by special hardware built to realize the architecture proposed in Chapter III, it was necessary to attempt a simulation of only a small frame size within a small range of acceptable errors.

4.2 The Experiment

A square frame with $k = 7$ was arbitrarily selected. The acceptable error range $0.24 \leq \varepsilon \leq 0.34$ was investigated. The implementation used the primitive trinomial

$$\lambda^7 + \lambda + 1 \tag{4.1}$$

The companion matrix for this trinomial is shown in Figure 23. By solving the appropriate simultaneous equations in $GF(2^7)$, the general form (3.20) is easily derived and shown in Figure 23. The codematrix set was created by producing each of the $2^7 - 1 = 127$ matrices corresponding to

0	0	0	0	0	0	1
1	0	0	0	0	0	1
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1

Figure 23. The Companion Matrix for the Primitive Trinomial $\lambda^7 + \lambda + 1$

powers of the matrix product (similarity transformed companion matrix)

$$PMP^{-1} \tag{4.2}$$

where the successive P's of (4.2) are the permutations of seven elements in lexicographic order. The P's can be generated by the Lehmer lexicographic method explained in section 3.8. We note that the general matrix form shown in Figure 24 possesses no symmetries and hence all of the matrices within the codematrix set, with the exception of the identity matrices $(PMP^{-1})^{127}$, will be different.

Figure 25 shows the results of the Monte-Carlo experiment as points overlaid on C(k = 7) computed by Equation (2.30) and displayed in Figure 15. The agreement between theory and experiment deomonstrated in Figure 25 lends credence to belief in the efficacy of the implementation method put forth in Chapter III.

m_1	m_2	m_3	m_4	m_5	m_6	m_7
m_7	$m_1 + m_2$	$m_2 + m_3$	$m_3 + m_4$	$m_4 + m_5$	$m_5 + m_6$	$m_6 + m_7$
m_6	m_7	$m_1 + m_2$	$m_2 + m_3$	$m_3 + m_4$	$m_4 + m_5$	$m_5 + m_6$
m_5	m_6	m_7	$m_1 + m_2$	$m_2 + m_3$	$m_3 + m_4$	$m_4 + m_5$
m_4	m_5	m_6	m_7	$m_1 + m_2$	$m_2 + m_3$	$m_3 + m_4$
m_3	m_4	m_5	m_6	m_7	$m_1 + m_2$	$m_2 + m_3$
m_2	m_3	m_4	m_5	m_6	m_7	$m_1 + m_2$

Figure 24. General Form of Powers of the Companion Matrix

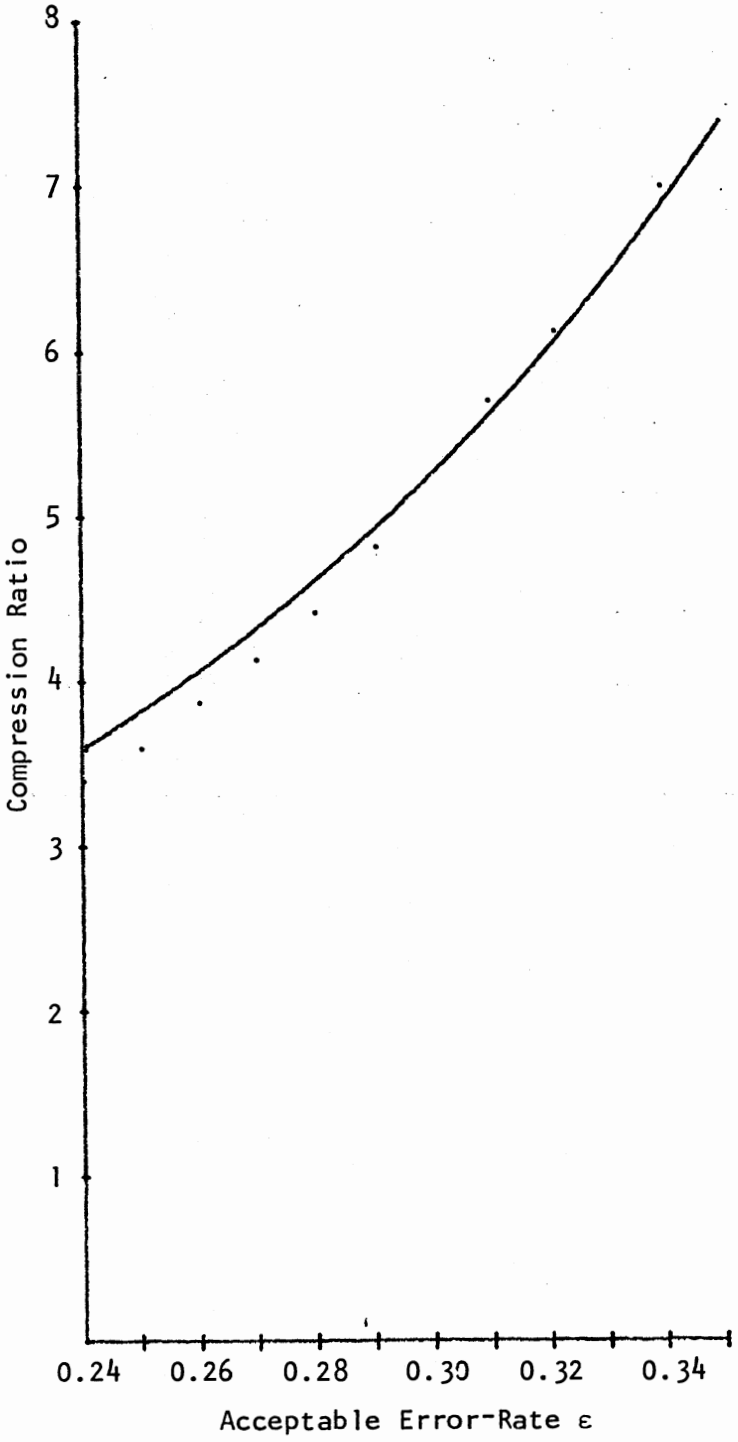


Figure 25. $C(k = 7)$ and $C_{Monte\text{-}Carlo}(k = 7)$ Plots Against Acceptable Error-Rate ε

CHAPTER V

SUMMARY AND POSSIBILITIES FOR EXTENDED STUDY

5.1 Summary

We have investigated image representation using a codematrix set of randomly derived square matrices. We have derived expressions relating compression to error rate. We have also explored some ideas regarding architectures that will allow for high-speed and efficient implementation of the method.

We have concluded that the codematrix matching method has value because it is universal. It will accept any two-tone image regardless of the image's statistical properties and it allows a near continuous tradeoff between compression and error rate. We also note that high compression rates are achievable only through a high error rate. While this is not a serious detriment to some defense oriented tasks such as template matching, it is a decided concern for image compression.

5.2 Extensions

An intriguing idea suggested by Dr. Rao Yarlagadda, and our first proposal, is to use the codematrix coding technique to <u>follow</u> a conventional redundancy removing technique such as the Discrete Cosine Transform (DCT) followed, upon receipt, by noise cleaning procedures such as median filtering. The appeal of this concept is that the conventional

or "front-end" transform would perform essentially the same function as prewhitening the data, i.e., it would remove the correlations between what the codematrix method can consider as pixels. Because the residuals will not, in general, be binary valued entities, they must be converted to binary form before applying the codematrix coding method. A number of conversion schemes appear possible and especially attractive are those PCM techniques that allow for Unequal Symbol Protection (USP).

Our second proposal is to examine other, nonbinary, number bases and various mathematical architectures for implementing the procedure. The use of number bases other than binary have particular problems as regards electronic implementation however, they may have special rewards. Extension of the mathematics to study these cases is not anticipated to be a difficult task but rather it is anticipated that the challenges will lie in creating the appropriate "closest-matching" metrics, influenced perhaps by psychovisual considerations, and implementation architectures should other number bases appear theoretically preferable to binary systems.

A third area for potential follow-on is selection of the best permutation scheme for achieving the similarity transformations put forth in Chapter III. The best choice will be influenced by many parameters, the chief one of which will be ease and speed of electronic implementation.

A potential fourth area for research concerns the search for a faster "best-frame-match" as opposed to the present architecture of exhausting all $2^k - 1$ matrices of $(PMP^T)^i$ for each P. This is probably an extremely difficult problem and it is not even known whether or not a method of subexponential complexity can indeed exist.

A SELECTED BIBLIOGRAPHY

[1] Netravali, A. N., and J. O. Limb. "Picture Coding: A Review." Proceedings of the IEEE, Vol. 68, No. 3 (March, 1980), pp. 366-406.

[2] Wintz, P. A. "Transform Picture Coding." Proceedings of the IEEE, Vol. 60, No. 7 (July, 1972), pp. 809-820.

[3] Jain, A. K. "Partial Differential Equations and Finite-Difference Methods in Image Processing, Part I: Image Representation." Journal of Optimization Theory and Applications, Vol. 23, No. 1 (September, 1977), pp. 65-91.

[4] Jain, A. K. "Image Data Compression: A Review." Proceedings of the IEEE, Vol. 69, No. 3 (March, 1981), pp. 349-389.

[5] Huang, T. S. "Coding of Two-Tone Images." IEEE Transactions on Communications, Vol. COM-25, No. 11 (November, 1977), pp. 1406-1424.

[6] Pratt, W. K., J. Kane, and H. C. Andrews. "Hadamard Transform Image Coding." Proceedings of the IEEE, Vol. 57, No. 1 (January, 1969), pp. 58-68.

[7] Legault, R. Perception of Displayed Information. (L. B. Biberman, Ed.) New York: Plenum Press, 1973.

[8] Kunt, M., and O. Johnsen. "Block Coding of Graphics: A Tutorial Review." Proceedings of the IEEE, Vol. 68, No. 7 (July, 1980), pp. 770-786.

[9] Dynkin, E. B., and A. A. Yushkevich. Markov Processes; Theorems and Problems. New York: Plenum Press, 1969.

[10] Hamming, R. W. "Error Detecting and Error Correcting Codes." The Bell System Technical Journal, Vol. 26, No. 2 (April, 1950), pp. 147-160.

[11] Kowalsky, H. J. Topological Spaces. New York and London: Academic Press, 1964.

[12] Fry, T. C. Probability and Its Engineering Uses. 2nd ed. Princeton, N.J.: Van Nostrand, 1965.

[13] Brockwell, P. J. "An Asymptotic Expansion for the Tail of a Binomial Distribution and Its Application in Queueing Theory." Journal of Applied Probability, Vol. 1 (1964), pp. 163-169.

[14] Levy, H., and L. Roth. Elements of Probability. Oxford: Clarendon Press, 1936, p. 67.

[15] "Tables of the Cumulative Binomial Probability Distribution." The Annals of the Computation Laboratory of Harvard University, Vol. 35. Cambridge, Mass.: Harvard University Press, 1955.

[16] MacDuffee, C. An Introduction to Abstract Algebra. New York: Wiley & Sons, 1940, p. 180.

[17] Birkhoff, G., and S. MacLane. A Survey of Modern Algebra. Revised Edition. New York: MacMillan, 1964.

[18] Perlis, S. Theory of Matrices. Cambridge, Mass.: Addison-Wesley, 1952.

[19] Golomb, S. W. Shift Register Sequences. San Francisco: Holden-Day, 1967.

[20] Warlick, W., and J. Hershey. "High-Speed m-Sequence Generators." IEEE Transactions on Computers, Vol. C-29, No. 5 (May, 1980), pp. 398-400.

[21] Berlekamp, E. Algebraic Coding Theory. New York: McGraw-Hill, 1968.

[22] MacDuffee, C. Vectors and Matrices. (Carus Mathematical Monographs, No. 7.) New York: Mathematical Association of America, 1943.

[23] Lehmer, D. H. "The Machine Tools of Combinatorics." Applied Combinatorial Mathematics. Ed. E. F. Beckenbach. New York: Wiley, 1964, pp. 5-31.

VITA

John Erik Hershey

Candidate for the Degree of

Doctor of Philosophy

Thesis: REPRESENTATION OF TWO-TONE IMAGES USING PSEUDORANDOMLY DERIVED SQUARE MATRICES

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in New York, New York on April 6, 1943, to Mr. and Mrs. P. J. Hershey.

Education: Bachelor of Science in Physics degree awarded by Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1965; Bachelor of Science in Electrical Engineering degree awarded by Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1966; Master of Science in Electrical Engineering awarded by the University of Arizona, Tucson, Arizona, in 1968; completed requirements for the Doctor of Philosophy degree in Electrical Engineering at Oklahoma State University in December, 1981.

Professional Background: Employed by the United States Government since 1968. Presently working for the Institute of Telecommunication Sciences, NTIA, Department of Commerce, Boulder, Colorado.