R. E. Blahut

Transform Techniques for Error Control Codes

By using the theory of finite field Fourier transforms, the subject of error control codes is described in a language familiar to the field of signal processing. The many important uses of spectral techniques in error control are summarized. Many classes of linear codes are given a spectral interpretation and some new codes are described. Several alternative encoder/decoder schemes are described by frequency domain reasoning. In particular, an errors-and-erasures decoder for a BCH code is exhibited which has virtually no additional computations over an errors-only decoder. Techniques for decoding BCH, RS, and alternant codes (Goppa codes) a short distance beyond the designed distance are discussed. Also, a modification to the definition of a BCH code is described which reduces the decoder complexity without changing the code's rate or minimum distance.

Introduction

Fourier transforms have found wide application in signal processing and in the study of communication waveforms; study of these transforms has been well rewarded. A close analogue of the Fourier transform can be defined on the vector space of n-tuples over the Galois field GF(q) whenever $n = q^m - 1$ or a submultiple as was noticed by Pollard [1]. Transforms over Galois fields have recently been introduced into the study of error control codes as a vehicle to reduce decoder complexity first by Gore [2], and later by Michelson [3], Lempel and Winograd [4], and Chien and Choy [5]. However, these transforms can be made to play a much more central role in the subject. Known ideas of coding theory can be described in a frequency domain setting that is much different from the familiar time domain setting, but closely related to treatments based on the so-called Mattson-Solomon polynomial [6]. Cyclic codes can be defined as codes whose codewords have certain specified spectral components equal to zero. Alternant codes (and Goppa codes) can be given a similar interpretation. Also, the decoding of many codes (including BCH, Reed-Solomon, and Goppa codes) can be described spectrally.

This emerging viewpoint is welcome and important for a number of reasons. Firstly, any new vantage point on an established discipline will usually bring new insights. Thus, existing codes can be classified in yet another way, and hence interrelationships can be seen in a new light. Secondly, there are strong pedagogical advantages since most engineers are well versed in transform techniques, especially those dealing with waveform design or signal processing. Thirdly, computational or implementation advantages often are found in a frequency domain decoder. Certainly, it is important to know as many decoder techniques as possible so that the simplest can be chosen for a given application. Finally, new codes, algorithms, and techniques can be sought from another point of view.

This paper will begin with some tutorial sections that develop part of the subject of error control codes in a transform setting. First of all, we want to set up the viewpoint, terminology, and analogies with signal processing theory so that the new results that come later can be explained easily. But we also hope to stimulate interest in and to accelerate the development of a spectral point of view to coding and to popularize the ideas of [2, 4] and [5]. It is our belief that the spectral formulation and terminology bring the subject much closer to the subject of signal processing and make error control coding more accessible to the nonspecialist in coding theory.

We then turn from the tutorial tone to give some new techniques. The spectral interpretation is used to describe encoder/decoder implementations for BCH and Reed-

Solomon codes. A new technique for correcting erasures and errors is described. This technique requires virtually no additional complexity over the decoder for correcting errors only.

We also describe techniques for decoding codes, including Goppa codes, beyond the designed distance; techniques for modifying the definition of BCH codes so that the decoder is simpler; and techniques for designing codes of very long blocklength that, although weak in performance, can be decoded using arithmetic in small fields. These techniques are described for their own merits, but they are also presented as evidence supporting our claim that the spectral point of view is quite rewarding and worth further development.

Finally, by the examples discussed and the general tone of the paper, we hope to underscore our view that it is as important to massage codes to fit known decoding algorithms as it is to seek new codes with good properties but having no known practical decoders.

Finite field transforms

The Fourier transform plays a basic role in the study of real-valued or complex-valued signals when the time variable is continuous, and the discrete Fourier transform plays a parallel role when the time variable is discrete. Fourier transforms also exist for functions of a discrete index that take values in a finite field. Such transforms are very useful in the study of error control codes, but they are less well known than Fourier transforms over the complex field, and so we review them in this section. The basic ideas appear earlier in [1].

Recall the definition of the discrete Fourier transform of a vector of complex numbers

$$P_k = \sum_{i=0}^{N-1} e^{-j2\pi N^{-1}ik} p_i \qquad k = 0, \cdots, N-1,$$

where here $j = \sqrt{-1}$. The Fourier kernel is an Nth root of unity in the field of complex numbers. In the finite field, GF(q^m), an element α of order n is an nth root of unity. Drawing on the analogy between $e^{-j2\pi N^{-1}}$ and α, we have the following definition.

Definition 1 Let $\mathbf{e} = \{e_i \mid i = 0, \cdots, n-1\}$ be a vector over GF(q), where n divides $q^m - 1$ for some m, and let α be an element of GF(q^m) of order n. The finite field Fourier transform of the vector $\mathbf{e}$ is the vector over GF(q^m), $\mathbf{E} = \{E_j \mid j = 0, \cdots, n-1\}$, given by

$$E_j = \sum_{i=0}^{n-1} \alpha^{ij} e_i \qquad j = 0, \cdots, n-1.$$

For simplicity of exposition, we will usually restrict attention to values of n satisfying $n = q^m - 1$. These values of n will be called primitive blocklengths. Then α is a primitive element of GF(q^m). It is natural to call the discrete index i "time" and $\mathbf{e}$ the time domain function or the signal and also call the discrete index j "frequency" and $\mathbf{E}$ the frequency domain function or the spectrum. Just as real-valued functions can have complex-valued Fourier transforms, so too can GF(q)-valued signals have GF(q^m)-valued Fourier transforms.

Theorem 1 Over GF(q), a field of characteristic p, a vector and its spectrum are related by

$$E_j = \sum_{i=0}^{n-1} \alpha^{ij} e_i,$$

$$e_i = \frac{1}{n \text{ modulo } p} \sum_{j=0}^{n-1} \alpha^{-ij} E_j.$$

Proof Recall that in any field,

$$x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1),$$

and by definition of α, α^r is a root of the left side for all r. Hence α^r is a root of the last term for all $r \neq 0$ modulo n. But this is equivalent to

$$\sum_{j=0}^{n-1} \alpha^{rj} = 0 \qquad r \neq 0 \text{ modulo } n,$$

while if $r = 0$,

$$\sum_{j=0}^{n-1} \alpha^{rj} = n \text{ modulo } p,$$

which is not zero if n is not a multiple of the field characteristic, p. Using these facts, we have

$$\sum_{j=0}^{n-1} \alpha^{-ij} \sum_{k=0}^{n-1} \alpha^{kj} e_k = \sum_{k=0}^{n-1} e_k \sum_{j=0}^{n-1} \alpha^{(k-i)j} = (n \text{ modulo } p) e_i.$$

Finally, if the field has characteristic p, then for some integer s, $q - 1 = p^s - 1$ is a multiple of n, and consequently n is not a multiple of p. Hence n modulo $p \neq 0$. This proves the theorem.

The Fourier transform has many strong properties which carry over to the finite field case. Suppose that

$$e_i = f_i g_i \qquad i = 0, \cdots, n-1.$$

Then

$$E_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i \left(\frac{1}{n} \sum_{k=0}^{n-1} \alpha^{-ik} G_k \right) = \frac{1}{n} \sum_{k=0}^{n-1} G_k \left(\sum_{i=0}^{n-1} \alpha^{i(j-k)} f_i \right).$$

We then have the convolution property

$$E_j = \frac{1}{n} \sum_{k=0}^{n-1} F_{j-k} G_k,$$

with the understanding that all subscripts are interpreted

modulo n (or equivalently, that the spectra are defined for all k and are periodic with period n). There is also a Parseval formula. From the convolution,

$$E_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i g_i = \frac{1}{n} \sum_{k=0}^{n-1} F_{j-k} G_k.$$

Take $j = 0$ to get

$$\sum_{i=0}^{n-1} f_i g_i = \frac{1}{n} \sum_{k=0}^{n-1} F_{n-k} G_k.$$

When dealing with polynomials, the polynomial $e(x) = e_{n-1}x^{n-1} + \cdots + e_1 x + e_0$ is associated with a polynomial $E(x) = E_{n-1}x^{n-1} + \cdots + E_1 x + E_0$ by means of the finite field Fourier transform. This polynomial is called the spectrum polynomial or the associated polynomial of $e(x)$. It was introduced to the study of error control codes by Mattson and Solomon [6], although not in the terminology of the Fourier transform.

The following theorem relates the roots of these polynomials to the properties of the spectrum.

Theorem 2 a) The polynomial $e(x)$ has a root at α^j if and only if the jth spectral component E_j equals zero. b) The polynomial $E(x)$ has a root at α^{-i} if and only if the ith time component e_i equals zero.

Proof Part a is immediate since

$$e(\alpha^j) = \sum_{i=0}^{n-1} e_i \alpha^{ij} = E_j.$$

Part b follows in the same way.

Thus, in the finite fields, when one speaks of roots of polynomials or of spectral components equal to zero, one really speaks of the same thing, but the terminology and the insights are different, and the two notions appeal to different audiences.

Cyclic codes

A code over GF(q) is a set of time domain signals of length n called codewords. If a Fourier transform exists for length n, then each codeword has a spectrum in an extension field GF(q^m) called the frequency domain codeword. A cyclic code is a code such that the linear combination of two codewords is a codeword, and the cyclic shift of a codeword is a codeword.

A cyclic code over GF(q) is conventionally described in terms of a generator polynomial $g(x)$ over GF(q) of degree $n - k$. Every codeword is represented by a polynomial of degree $n - 1$, written as $c(x) = s(x)g(x)$, where $s(x)$ is a signal polynomial of degree $k - 1$. This is a convolution in the time domain

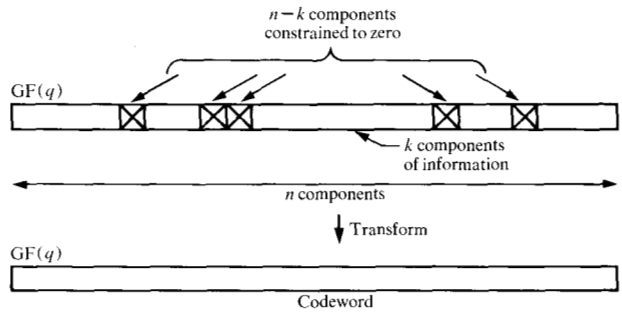

Figure 1 Encoding via the transform.

$$c_i = \sum_{k=0}^{n-1} g_{i-k} s_k.$$

Therefore, in the frequency domain, the encoding operation can be written as a product $C_j = G_j S_j$. For fixed generator G_j, any spectrum that satisfies this expression is a frequency domain codeword, provided that all components in the time domain are GF(q)-valued. Because the signal spectrum S_j is arbitrary, the only significant role of G_j is to specify frequencies where the codeword spectrum C_j is zero. Thus, we can define a cyclic code alternatively as follows. Given a set of spectral components $j_1, \cdots, j_{n-k}$ called parity frequencies, the set of words over GF(q) whose spectrum is zero in components $j_1, \cdots, j_{n-k}$ is a cyclic code.

Notice that although each codeword in a cyclic code is a vector over GF(q), the spectrum is a vector over GF(q^m). Hence, we may think of a cyclic code as the set of inverse Fourier transforms of all spectral vectors that are constrained to zero in several prescribed components provided that said Fourier transforms are GF(q)-valued. It is not possible to choose any spectrum that is zero in the prescribed components; some of these may have inverse transforms with components that are not in GF(q).

However, if $m = 1$, that is, if $n = q - 1$, then every spectrum consistent with the constraints yields a codeword. One may encode by filling the unconstrained components of the spectrum with information symbols and then taking the inverse transform as illustrated in Fig. 1.

The most popular class of cyclic code is the class of BCH codes. From the spectral point of view we have the following definition.

Definition 2 A primitive t-error-correcting BCH code of blocklength $n = q^m - 1$ is the set of all words over GF(q) whose spectrum is zero in a specified block of $2t$ consecutive components.

C_1	C_2	C_3	C_4	C_5	C_6	C_0
0	0	0	0	0	0	0
0	0	α^0	0	α^0	α^0	0
0	0	α^1	0	α^2	α^4	0
0	0	α^2	0	α^4	α^1	0
0	0	α^3	0	α^6	α^5	0
0	0	α^4	0	α^1	α^2	0
0	0	α^5	0	α^3	α^6	0
0	0	α^6	0	α^5	α^3	0

C_1	C_2	C_3	C_4	C_5	C_6	C_0
0	0	0	0	0	0	1
0	0	α^0	0	α^0	α^0	1
0	0	α^1	0	α^2	α^4	1
0	0	α^2	0	α^4	α^1	1
0	0	α^3	0	α^6	α^5	1
0	0	α^4	0	α^1	α^2	1
0	0	α^5	0	α^3	α^6	1
0	0	α^6	0	α^5	α^3	1

Figure 2 Hamming (7, 4) code in the frequency domain.

The adjective "consecutive" is the key one in specializing the definition of a cyclic code to that of a BCH code. It is well known that a BCH code corrects t errors. In the next section we give the proof couched in spectral terminology. The remainder of this section is concerned with encoding.

First suppose that $n = q - 1$ (or possibly a submultiple of $q - 1$). The BCH code is then known as a Reed-Solomon code. The encoding is as follows. Some set of $2t$ consecutive frequencies (*e.g.*, the first $2t$) are chosen as the symbols constrained to zero. The information symbols are loaded into the remaining $n - 2t$ symbols, and the inverse Fourier transform is taken to produce a (nonsystematic) codeword.

For the more general BCH code, the encoding is more complex. Again, $2t$ consecutive locations are chosen to be zero. The remaining symbols must be chosen to represent the information only in those q^k possible ways that have GF(q)-valued transforms; but of course we do not wish to do this by trial and error.

Recall that over the complex numbers, a function $V(f)$ has a real-valued transform if $V^*(-f) = V(f)$. The analogous condition is given by the following well-known theorem, which may be found in [7].

Theorem 3 Let C_j, $j = 0, \cdots, n-1$, take elements in GF(q^m) where n is a divisor of $q^m - 1$. Then c_i, $i = 0, \cdots, n-1$, are all elements of GF(q) if and only if the following equations, called conjugacy constraints, are satisfied.

$$C_j^q = C_{qj \bmod n} \qquad j = 0, \cdots, n-1.$$

Proof By definition

$$C_j = \sum_{i=0}^{n-1} \alpha^{ij} c_i \qquad j = 0, \cdots, n-1.$$

As is well known, for a field of characteristic p and any integer r, $(a+b)^{p^r} = a^{p^r} + b^{p^r}$. Therefore,

$$C_j^q = \left(\sum_{i=0}^{n-1} \alpha^{ij} c_i \right)^q = \sum_{i=0}^{n-1} \alpha^{qij} c_i^q.$$

If c_i is an element of GF(q) for all i, then $c_i^q = c_i$. Consequently,

$$C_j^q = \sum_{i=0}^{n-1} \alpha^{qji} c_i = C_{qj \bmod n}.$$

Conversely, suppose that for all j, $C_j^q = C_{qj}$. Then

$$\sum_{i=0}^{n-1} \alpha^{iqj} c_i^q = \sum_{i=0}^{n-1} \alpha^{iqj} c_i \qquad j = 0, \cdots, n-1.$$

Let $k = qj$. Since q is relatively prime to $n = q^m - 1$, as j ranges over all values between 0 and $n-1$, k also ranges over all values between 0 and $n-1$. Hence

$$\sum_{i=0}^{n-1} \alpha^{ik} c_i^q = \sum_{i=0}^{n-1} \alpha^{ik} c_i \qquad k = 0, \cdots, n-1,$$

and by uniqueness of the Fourier transforms $c_i^q = c_i$ for all i. Thus, c_i is a root of $x^q - x$ for all i and all such roots are elements of GF(q).

Using Theorem 3, we can easily construct the Hamming (7, 4) code in the frequency domain. This is shown in Fig. 2. Frequencies C_1 and C_2 are chosen as parity frequencies so that a single error can be corrected. The information is contained at frequencies C_0 and C_3. The remaining frequencies are constrained by Theorem 3. Theorem 3 also requires that $C_0^2 = C_0$ so that C_0 can only have the value 0 or 1. Thus, the equivalent "bit content" of C_0 is one bit. The equivalent bit content of C_3 is three bits. Thus, the four information bits of the Hamming code can be used to uniquely specify the spectrum. The information bits are inserted into the frequency domain rather than the time domain.

In the general case, the integers modulo n are divided into conjugacy classes of the form

$$A_j = \{qj, q^2j, q^3j, \cdots, j\}.$$

If the spectral component C_j is specified, then every other spectral component whose index is in the conjugacy class of j must be a power of C_j and hence cannot be separately specified. (It is suggestive to use the term "chord" for the set of frequencies whose indices are in the same conjugacy class.) Further, if the conjugacy class has r members, then we must have

$$C_j^{q^r} = C_j.$$

Consequently, we are not free to choose any element of GF(q^m) for C_j, but only those of order $q^r - 1$. Since every element of GF(q^m) has order dividing $q^m - 1$, it is clear that the size of every conjugacy class divides m.

Thus, to specify an encoder, we break the first $q^m - 1$ integers into conjugacy classes, and select one integer to represent each class. These representatives specify the uniquely assignable symbols. To form a BCH code, a block of $2t$ spectral components are chosen as parity frequencies and set to zero. The remaining assignable symbols are information symbols, arbitrary except for occasional constraints on the order. All other symbols in a chord are not free; they are obligatory frequencies.

Table 1 shows the situation for GF(64). If we choose the first column as free symbols and take C_1, C_2, C_3, C_4, C_5, and C_6 as parity frequencies, then we have a triple-error-correcting BCH code. Then C_0, C_7, C_9, C_{11}, C_{13}, C_{15}, C_{21}, C_{23}, C_{27}, and C_{31} are the information symbols. C_9 and C_{27} must be symbols of order 2. C_{21} must be an element of order 3. C_0 must be an element of order 1. All other symbols are arbitrary elements of GF(64). It requires a total of 45 information bits to specify these symbols. Hence, we have the (63, 45) $t = 3$ BCH code.

After these free symbols are loaded, the obligatory symbols are padded with appropriate powers. The complete spectrum is then transformed into the codeword.

Table 1 Structure of spectrum over GF(64).

Free frequencies	*Obligatory frequencies*					*Bit content*
$\{C_0\}$						1
$\{C_1$	C_2	C_4	C_8	C_{16}	$C_{32}\}$	6
$\{C_3$	C_6	C_{12}	C_{24}	C_{48}	$C_{33}\}$	6
$\{C_5$	C_{10}	C_{20}	C_{40}	C_{17}	$C_{34}\}$	6
$\{C_7$	C_{14}	C_{28}	C_{56}	C_{49}	$C_{35}\}$	6
$\{C_9$	C_{18}	$C_{36}\}$				3
$\{C_{11}$	C_{22}	C_{44}	C_{25}	C_{50}	$C_{37}\}$	6
$\{C_{13}$	C_{26}	C_{52}	C_{41}	C_{19}	$C_{38}\}$	6
$\{C_{15}$	C_{30}	C_{60}	C_{57}	C_{51}	$C_{39}\}$	6
—						—
—						—
$\{C_{21}$	$C_{42}\}$					2
$\{C_{23}$	C_{46}	C_{29}	C_{58}	C_{53}	$C_{43}\}$	6
—						—
$\{C_{27}$	C_{54}	$C_{45}\}$				3
$\{C_{31}$	C_{62}	C_{61}	C_{59}	C_{55}	$C_{47}\}$	6

Decoding in the frequency domain

The BCH bound is proved very simply and intuitively in the frequency domain. This proof is a variation of a time domain proof of Chien [8] that we have transferred into the frequency domain.

Theorem 4 (BCH bound) If a vector c_i, $i = 0, \cdots, n - 1$, has less than d nonzero components and if the spectrum is zero on any $d - 1$ successive components ($C_j = 0$, $j = j_0 + 1, \cdots, j_0 + d - 1$), then $c_i = 0$ for all i.

Proof Let $i_1, \cdots, i_\nu$ denote the ν nonzero components of $\mathbf{c}$, $\nu < d$. Define the locator polynomial $\Lambda(x)$:

$$\Lambda(x) = \prod_{k=1}^{\nu} (1 - x\alpha^{i_k})$$
$$= \Lambda_\nu x^\nu + \Lambda_{\nu-1}x^{\nu-1} + \cdots + \Lambda_1 x + \Lambda_0.$$

Interpreted as a vector, Λ is a frequency spectrum which is judiciously defined so that its inverse transform $\lambda = \{\lambda_i\}$ equals zero (by Theorem 2) at every time i, at which $c_i \neq 0$. The product $\lambda_i c_i$ in the time domain is zero; therefore, the convolution in the frequency domain is zero.

$$\sum_{k=0}^{n-1} \Lambda_k C_{j-k} = 0.$$

But the vector Λ is nonzero only in a block of length at most d and $\Lambda_0 = 1$ so that

$$C_j = -\sum_{k=1}^{d-1} \Lambda_k C_{j-k}.$$

This is the equation of a linear feedback shift register that generates all the C_j from any block of them of length $d - 1$. But $\mathbf{C}$ is zero in a block of length $d - 1$. Using this block as an initial condition, the linear feedback shift register generates all the rest; hence, all terms of $\mathbf{C}$ are zero, and $\mathbf{c}$ must be the all-zero vector.

Now consider the task of decoding a received word for a BCH or Reed-Solomon code. The original decoding technique described by Reed and Solomon [9] had a strong frequency domain flavor, but thereafter techniques evolved primarily in the time domain, although some frequency domain variables (the syndromes) do creep in. What amounts to a frequency domain decoder was first proposed by Mandelbaum [10], though in the terminology of the Chinese Remainder Theorem. Such a decoder was implemented in a computer program by Paschburg [11].

Consider decoding a received word $r_i = c_i + e_i$, $i = 0, \cdots, n - 1$, consisting of a codeword and an error word. Figure 3 shows a comparison of a time domain implementation of a BCH code, a frequency domain implementation, and several hybrid implementations. In the frequency domain implementation, one first computes $\mathbf{R}$, the transform of the received word $\mathbf{r}$. The transform consists of the transform of the codeword and the transform of the error:

$$R_j = C_j + E_j \qquad j = 0, \cdots, n - 1.$$

Since codeword C_j is zero on a block of $2t$ components (say from 1 to $2t$), we have $2t$ known values of E_j called the syndromes:

$$S_j = E_j = R_j \qquad j = 1, \cdots, 2t.$$

Suppose there are $\nu \leq t$ errors. As in the proof of Theorem 4, define the error-locator polynomial $\Lambda(x)$:

$$\Lambda(x) = \prod_{k=1}^{\nu} (1 - x\alpha^{i_k}).$$

Since in the time domain $\lambda_i = 0$ whenever $e_i \neq 0$, we have $\lambda_i e_i = 0$ and so

$$\sum_{j=0}^{n-1} \Lambda_j E_{k-j} = 0.$$

(It does no harm to sum out to $n - 1$ even though $\Lambda_j = 0$ for $j > t$.) The convolution is a set of n equations in $n - t$ unknowns: t coefficients of $\Lambda(x)$ and $n - 2t$ components of $\mathbf{E}$. Of the n equations, there are t equations that involve only components of $\mathbf{\Lambda}$ and known components of $\mathbf{E}$, and these are always solvable for $\mathbf{\Lambda}$. The remaining components of $\mathbf{E}$ can be obtained by recursive extension; that is, sequentially computed from $\mathbf{\Lambda}$ using the above convolution equation. This computation can be described as the operation of a linear feedback shift register with tap weights given by the coefficients of $\mathbf{\Lambda}$. In this way E_j is computed for all j, and $C_j = R_j - E_j$. An inverse Fourier transform completes the decoding.

The key step of computing $\mathbf{\Lambda}$ from the known $2t$ components of $\mathbf{E}$ can be done by the elegant algorithm of Berlekamp [12] which was described in terms of shift registers by Massey [13]. We will discuss some modifications of this algorithm in later sections, so we restate it here.

Theorem 5 (Berlekamp-Massey algorithm) Let S_{j_0+1}, $\cdots$, S_{j_0+2t} be given. Let the following set of recursive equations be used to compute $\Lambda^{(2t)}(x)$:

$$\Delta_r = \sum_{j=0}^{n-1} \Lambda_j^{(r-1)} S_{j_0+r-j},$$

(Δ_r is a scalar known as the discrepancy)

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) - \Delta_r x B^{(r-1)}(x),$$

$$B^{(r)}(x) = (1 - \delta_r) x B^{(r-1)}(x) + \delta_r x \Delta_r^{-1} \Lambda^{(r-1)}(x),$$

$r = 1, \cdots, 2t$; the initial conditions are $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, and $\delta_r = 1$ if both $\Delta_r \neq 0$ and $\deg B^{(r-1)}(x) > \max \deg \Lambda^{(i)}(x)$ and otherwise $\delta_r = 0$. Then $\Lambda^{(2t)}(x)$ is the smallest degree polynomial such that $\Lambda_0^{(2t)} = 1$ and

$$\sum_{j=0}^{n-1} \Lambda_j^{(2t)} S_{j_0+r-j} = 0 \qquad r = 1 + \deg \Lambda^{(2t)}, \cdots, 2t.$$

A proof can be found in [12] or [13]. If $S_{j_0+1}, \cdots, S_{j_0+2t}$ are the syndromes of a codeword for which at most t errors occurred, then $\Lambda^{(2t)}(x)$ has degree equal to the number of errors and

$$\sum_{j=0}^{n-1} \Lambda_j^{(2t)} S_{j_0+r-j} = 0 \qquad r = t + 1, \cdots, 2t$$

(but not conversely).

The decoder can be used with an encoder that is either in the time domain or the frequency domain. If the encoder is in the frequency domain, then the information symbols are used to specify certain components of the spectrum whose inverse transform then gives the time domain codeword. With this scheme, the corrected spectrum is the information. The decoder does not have an inverse transform. The frequency domain encoder may be simpler than the time domain encoder if n is composite because a fast transform may be simpler than a convolution.

The final circuit of Fig. 3 shows a hybrid implementation. Here the transform of the error pattern is computed by recursive extension in the frequency domain, but the correction is done in the time domain. This circuit is similar in appearance to the first circuit, but the development has been much different. The syndrome generator is the same as the direct transform. The Chien search is essentially the same as the inverse transform. It is a $\nu \times n$ transform with a GF(q^m)-valued output vector compared to an $n \times n$ transform with a GF(q)-valued output. The fourth circuit has the advantage of a simpler appearance than the first.

In view of the many variations summarized by Fig. 3, the designer has a number of options from which to choose. It should be obvious that his choice will depend not only on the code parameters such as blocklength and minimum distance, but also on how the implementation is divided between hardware and software, and even on the type of circuitry available to him.

Notice that each of the circuits of Fig. 3 has both a Fourier transform and an inverse Fourier transform, though in some cases these appear under the names "Chien search" or "syndrome generator," and in some cases not all of the output components need be computed. Thus, one needs efficient methods for computing the Fourier transforms. As is well known, the Fourier transform can be efficiently computed by a fast Fourier transform algorithm whenever $n = q^m - 1$ is composite, and this is sometimes used to justify choice of a composite blocklength. But even when $q^m - 1$ is prime the transform often is still practical. Circuitry to implement the full $n \times n$ matrix multiplication can be quite simple for moderate n. For example, when n is prime and α has a square root β, one can also use the chirp transform. This is a convenient variation of the Fourier transform based on the calculation

$$\beta^{-j^2} \sum_{i=0}^{n-1} \beta^{(i-j)^2} (\beta^{-i^2} c_i) = \sum_{i=0}^{n-1} \beta^{2ij} c_i = \sum_{i=0}^{n-1} \alpha^{ij} c_i = C_j.$$

The term on the left can be easy to implement in hardware. It consists of a pointwise product of c_i with β^{-i^2}

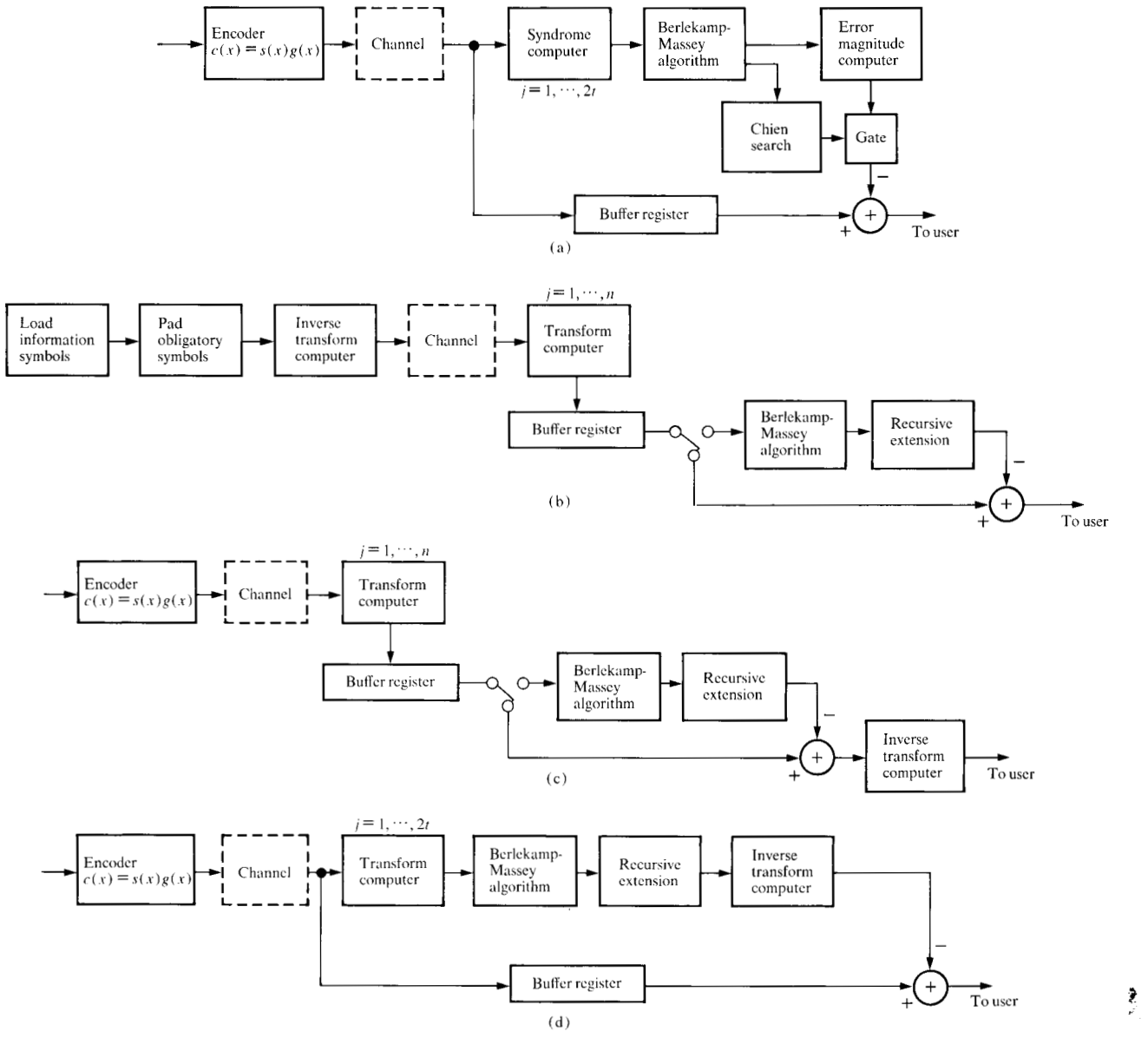

Figure 3 Implementation of BCH codes: (a) time domain implementation; (b) frequency domain implementation; (c) mixed domain implementation—time domain encoder, frequency domain decoder; (d) mixed domain implementation—time domain encoder, hybrid decoder.

(n multiplications) followed by a convolution with β^{j^2} (an n-tap digital filter) followed by a pointwise product with β^{-j^2} (n multiplications).

Erasure and error decoding

BCH codes also are used for protection with channels that make both erasures and errors. A decoding algorithm for this purpose was discovered by Forney [14]. The derivation is manipulative and difficult to understand intuitively since it introduces some new variables in an arbitrary way. By transforming the discussion into the frequency domain, algorithms for decoding messages with erasures and errors are very simply explained, and the reason why the decoder works is clearly visible. Further, the sharp insight allows us to propose a simple adaptation of the Berlekamp-Massey algorithm so that both erasures and errors can be decoded with virtually no hardware other than that required for the errors-only decoder.

Let $\mathbf{v}$ be the vector of erased symbols. Suppose that erasures are made in locations $i_1, i_2, \cdots, i_\rho$. (In other components $v_i = 0$.) The received word is a codeword corrupted by errors and erasures,

$$r_i = c_i + e_i + v_i \qquad i = 0, \cdots, n-1.$$

Let $\boldsymbol{\omega}$ be any vector that is zero at every erased location and otherwise nonzero. In particular, define $\boldsymbol{\omega}$ as follows. Let $U_l = \alpha^{i_l}$, $l = 1, \cdots, \rho$, denote the erasure locations. Define the erasure-locator polynomial

$$\Omega(x) = \sum_{k=0}^{n-1} \Omega_k x^k = \prod_{l=1}^{\rho} (1 - xU_l).$$

This is defined so that the inverse transform of the vector $\boldsymbol{\Omega}$ has components ω_i equal to zero whenever $v_i \neq 0$. Therefore, $\omega_i v_i = 0$. Then

$$\omega_i r_i = \omega_i(c_i + e_i + v_i) = \omega_i c_i + \omega_i e_i$$

or

$$r'_i = \omega_i c_i + e'_i,$$

where we have defined the modified received word $r'_i = \omega_i r_i$, and the modified error vector $e'_i = \omega_i e_i$. The modified error vector $\mathbf{e}'$ has errors in the same locations as $\mathbf{e}$. The problem now is to decode $\mathbf{r}'$ to find $\mathbf{e}'$. In the frequency domain

$$\mathbf{R}' = (\boldsymbol{\Omega} * \mathbf{C}) + \mathbf{E}'.$$

But $\boldsymbol{\Omega}$ is nonzero in a block of length $\rho + 1$ and by construction of a BCH code, $\mathbf{C}$ is zero in a block of length $2t$. Consequently, $\boldsymbol{\Omega} * \mathbf{C}$ is zero in a block of length $2t - \rho$. In this block, define the modified syndrome T_j by $T_j = R'_j$. Then $T_j = (\boldsymbol{\Omega} * \mathbf{R})_j = E'_j$.

Hence, just as in the errors-only case, from these $2t - \rho$ known values of $\mathbf{E}'$ we can find the error-locator polynomial $\Lambda(x)$ provided the number of errors is less than $(2t - \rho)/2$. Once the error-locator polynomial is known, we can combine it with the erasure-locator polynomial and proceed as in the errors-only case. To do this, first define the error-and-erasure-locator polynomial $\Gamma(x) = \Omega(x)\Lambda(x)$. The inverse Fourier transform of $\boldsymbol{\Gamma}$ is zero at every erasure or error. That is, $\gamma_i = 0$ if $e_i \neq 0$ or $v_i \neq 0$. Therefore, $\gamma_i(e_i + v_i) = 0$, $\boldsymbol{\Gamma} * (\mathbf{E} + \mathbf{V}) = 0$, and $\boldsymbol{\Gamma}$ is nonzero in a block of length at most $2t - \rho + 1$. Hence, the $2t$ known values of $\mathbf{E} + \mathbf{V}$ can be recursively extended to n values by using this convolution equation and the known value of $\boldsymbol{\Gamma}$. Then

$$C_i = R_i - (E_i + V_i).$$

An inverse Fourier transform completes the decoding.

The step of computing the error-locator polynomial from the modified syndromes can use the Berlekamp-Massey algorithm. However, it is possible to do much better by combining several steps. To describe how to do this it is necessary to refer back to the procedure of the Berlekamp-Massey algorithm as summarized by Theorem 5. The idea of the Berlekamp-Massey algorithm is to compute $\Lambda(x)$ by a recursive procedure, starting with an initial estimate $\Lambda^{(0)}(x) = 1$ and an initial choice of another polynomial called the update polynomial $B^{(0)}(x) = 1$, and proceeding through $2t$ iterations.

In the case of erasures, the syndrome is replaced with the modified syndrome in the equation for Δ_r,

$$\Delta_r = \sum_{j=0}^{n} \Lambda_j^{(r-1)} T_{n-j}.$$

After n iterations starting with the initial values $\Lambda^{(0)}(x) = B^{(0)}(x) = 1$, the error-locator polynomial $\Lambda(x)$ is obtained.

But what happens if we start instead with the values $\Lambda^{(0)}(x) = B^{(0)}(x) = \Omega(x)$? Then notice that

$$\Lambda^{(r)}(x)\Omega(x) = \Lambda^{(r-1)}(x)\Omega(x) - \Delta_r x B^{(r-1)}(x)\Omega(x),$$

$$B^{(r)}(x)\Omega(x) = (1 - \delta_r) x B^{(r-1)}(x)\Omega(x) + \delta_r x \Delta_r^{-1} \Lambda^{(r-1)}(x)\Omega(x),$$

and if we define $\Gamma^{(r)}(x) = \Lambda^{(r)}(x)\Omega(x)$ and compute Δ_r by

$$\Delta_r = \sum_{j=0}^{n} \Gamma_j^{(r-1)} S_{n-j} = \sum_{j=0}^{n} \Gamma_{n-j}^{(r-1)} S_j,$$

then

$$\Delta_r = \sum_{j=0}^{n} \left(\sum_{k=0}^{n} \Lambda_k^{(r-1)} \Omega_{n-j-k} \right) S_j = \sum_{k=0}^{n} \Lambda_k^{(r-1)} T_{n-k}.$$

Therefore, if we initialize the Berlekamp-Massey algorithm with $\Omega(x)$ instead of with 1, the modified syndromes are computed implicitly and need not explicitly appear, while the algorithm generates recursively the error-and-erasure-locator polynomial $\Gamma(x)$ according to the equations

$$\Gamma^{(r)}(x) = \Gamma^{(r-1)}(x) - \Delta_r x B^{(r-1)}(x),$$

$$B^{(r)}(x) = (1 - \delta_r) x B^{(r-1)}(x) + \delta_r x \Delta_r^{-1} \Gamma^{(r-1)}(x),$$

$$\Delta_r = \sum_{j=0}^{n} \Gamma_j^{(r-1)} S_{n-j}.$$

The resulting decoder is shown in Fig. 4. The only change from the decoder for errors only is the computation of the erasure-locator polynomial, which is trivial compared to other decoding computations.

Finally, notice that it does not matter what symbol actually appears in an erased symbol; it can be set to the most likely estimate of the received symbol, if the application uses this information to assess the probability of correct decoding.

Alternant codes

The decoding techniques we have described apply not only to BCH codes, but also to alternant codes. Alternant codes comprise a class of linear codes introduced by

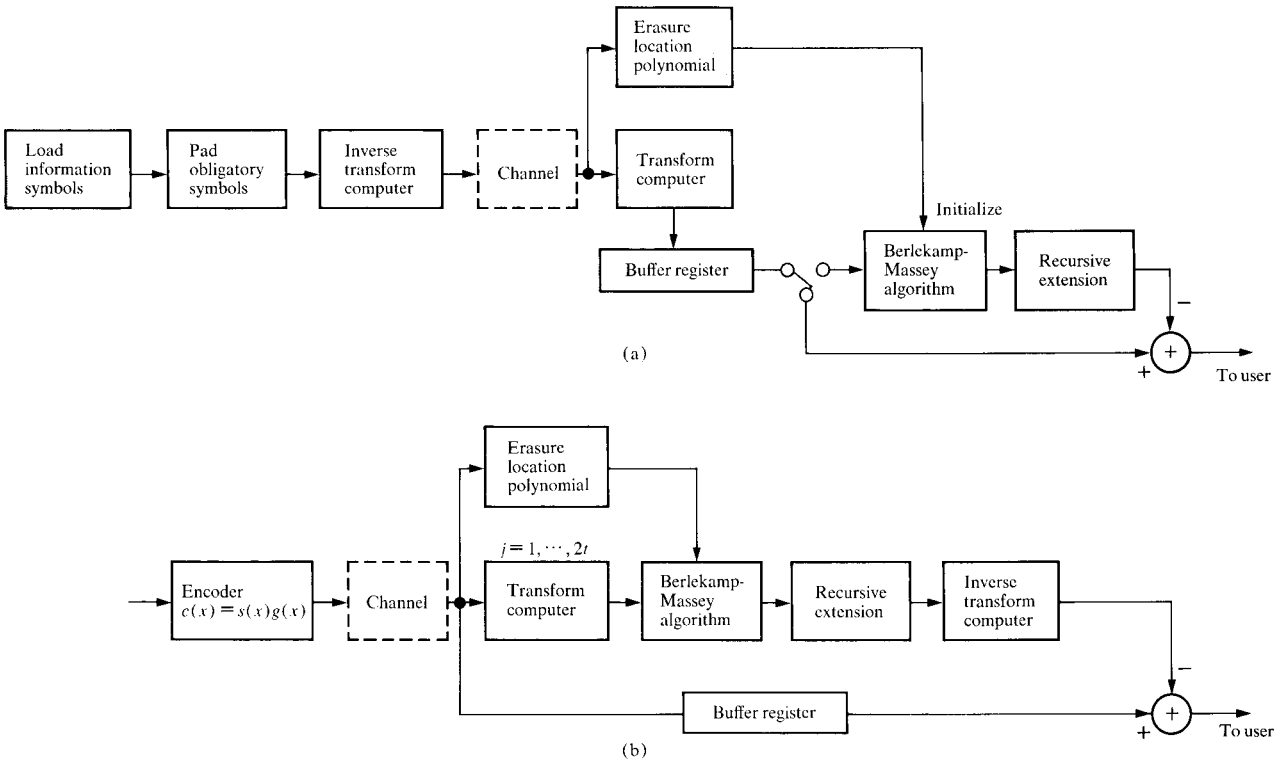

Figure 4 Error and erasure decoding for BCH codes: (a) frequency domain implementation; (b) mixed domain encoder—time domain encoder, hybrid decoder.

Helgert [15] and include the class of codes introduced by Goppa [16]. Delsarte [17] showed that an alternant code is a subset of slightly modified Reed-Solomon code. Choose **h** a fixed n-vector of nonzero components over GF(q^m), and choose a Reed-Solomon code over GF(q^m) with designed distance $2t + 1$. The alternant code consists of all GF(q)-valued vectors **c** such that $h_i c_i$, $i = 0, \cdots, n-1$, is a codeword in the Reed-Solomon code. Alternant codes are highly regarded because some of them have true minimum distance considerably larger than the designed distance (asymptotically close to the Gilbert bound).

The definition of the alternant codes is easily translated into the frequency domain where it takes on more of a signal processing flavor. Let $g_i = h_i^{-1}$, which is always defined since $h_i \neq 0$, and let **G** and **H** denote the transforms of **g** and **h**. Then since $g_i h_i = 1$ for all i, (**G** * **H**) equals one at $j = 0$ and otherwise equals zero. (**H** is an invertible filter.) The alternant code $\mathscr{C}$ is the set of vectors whose transforms C_j, $j = 0, \cdots, n-1$, satisfy two conditions.

$$\sum_{j=0}^{n-1} H_{j-k} C_j = 0 \qquad k = 1, \cdots, 2t$$

and $C_j^2 = C_{qj}$, with indices interpreted modulo n in both conditions. The first of these conditions is a convolution corresponding to the time domain product of the more usual definition; the second condition ensures that the time domain codewords are GF(q)-valued. Thus, the codeword spectrum is filtered by **H** prior to specifying the $2t$ contiguous parity frequencies.

A Goppa code is an alternant code of designed distance $2t + 1$ with **G** described by a polynomial of degree $2t$, called the Goppa polynomial.

The alternant codes can be decoded just as the BCH codes. All that needs to be added is a step to modify the syndromes by the inverse of the vector **h** either by multiplying in the time domain or convolving in the frequency domain. No other change is necessary. Hence, any frequency domain or time domain BCH decoder can decode alternant codes out to the designed distance $2t + 1$. However, since the appeal of alternant codes lies in their much larger minimum distance, it is not clear that an alternant code used with a BCH decoder has any advantage over a BCH code used with a BCH decoder. Alternant codes will not have practical importance until a constructive procedure is found for obtaining the good ones, and a decoding algorithm is found for decoding beyond the designed distance. Some small steps in this direction are discussed in the next section.

Decoding beyond the BCH bound

If every codeword in a code $\mathscr{C}$ has a certain set of $2t$ contiguous spectral components equal to zero, then by the BCH bound, the minimum distance of the code is at least $2t + 1$ and most of the common decoding algorithms will correct up to t errors. However, the minimum distance of the code might actually be larger than $2t + 1$, and in any case some patterns of more than t errors may be correctable.

The variations that can occur are illustrated by the following four examples:

1. A binary BCH code with all of the parity frequencies in a block of length $2t$ but with the actual minimum distance larger than $2t + 1$. (The Golay code is an example of such a code.)
2. A cyclic code with parity frequencies that are not contiguous.
3. A decoder for some of the $(t + 1)$-error patterns in a t-error-correcting Reed-Solomon code.
4. A decoder for an alternant code such as a Goppa code.

Berlekamp [12] discusses decoding of BCH codes beyond the BCH bound by forcing appropriate constraint equations in the frequency domain to be satisfied, but the techniques quickly become unmanageable as the number of errors increases beyond t. Hartmann [18] gives some closely related frequency domain techniques that again involve searching through sets of nonlinear equations for solutions. We will describe here some time domain decoding techniques that decode BCH or alternant codes a small distance beyond the designed distance. These techniques are motivated by [12, 18], but for some applications the complexity appears to grow more slowly as the number of errors increases beyond t. The basic idea is to add extra discrepancies as unknown variables, decode in terms of these variables, and then solve for the variables by some kind of search over low weight error patterns and when available, by using *a priori* facts such as that the codeword is GF(q)-valued. We will only discuss the decoding of errors; if desired, the ideas of the previous section may be added to decode erasures as well.

We start the discussion with a Reed-Solomon code of designed distance $2t + 1$. Then any polynomial $\Lambda(x)$ of degree $t + \nu$ with $t + \nu$ distinct roots is a legitimate error-locator polynomial if

$$\sum_{j=0}^{n-1} \Lambda_j S_{r-j} = 0 \qquad r = 1 + t + \nu, \cdots, 2t.$$

The smallest-degree such polynomial (if there is one) corresponds to the maximum-likelihood codeword. If it is of degree at most t, this polynomial is produced by the Berlekamp-Massey algorithm. Even if there are more than t errors, the smallest-degree polynomial may be unique, and the received word then can be uniquely decoded.

Suppose there are $t + 1$ errors, then the two unknown syndromes S_{2t+1}, S_{2t+2} will be enough, if known, to find $\Lambda(x)$. Hence, analytically continue the Berlekamp-Massey algorithm through two more iterations with these new syndromes as unknowns. Then we have

$$\Lambda^{(2t+2)}(x) = [1 - \delta_{2t+1}\Delta_{2t+2}\Delta_{2t+1}^{-1}x^2]\Lambda^{(2t)}(x) - [\Delta_{2t+1}x + (1 - \delta_{2t+1})\Delta_{2t+2}x^2]B^{(2t)}(x),$$

where $\delta_{2t+1} \in \{0, 1\}$, $\Delta_{2t+1}, \Delta_{2t+2} \in \mathrm{GF}(q^m)$, and $\delta_{2t+1} = 0$ whenever $\Delta_{2t+1} = 0$. Except for δ_{2t+1}, Δ_{2t+2}, and Δ_{2t+1}, everything on the right is known from the $2t$ syndromes. Transform the frequency domain vector $\mathbf{\Lambda}^{(2t+2)}$ into the time domain by transforming the two components on the right to get

$$\lambda_i^{(2t+2)} = [1 - \delta_{2t+1}\Delta_{2t+2}\Delta_{2t+1}^{-1}\alpha^{-2i}]\lambda_i^{(2t)} - [\Delta_{2t+1}\alpha^{-i} + (1 - \delta_{2t+1})\Delta_{2t+2}\alpha^{-2i}]b_i^{(2t)},$$

where we have used the general fact that if $E'_j = E_{j-1}$ then the inverse transform satisfies $e'_i = \alpha^{-i}e_i$. We must now choose the unknowns, if possible, so that the error pattern contains at most $t + 1$ nonzero components. If $\deg \Lambda^{(2t)} \leq t$ and the number of distinct zeros of $\lambda^{(2t)}$ equals the degree of $\Lambda^{(2t)}$, then the number of errors equals the degree of $\Lambda^{(2t)}$. This case is easily checked.

If there is only one solution for $\lambda^{(2t+2)}$ with $t + 1$ zero components and a corresponding $\Lambda^{(2t+2)}(x)$ of degree $t + 1$, a unique pattern of $t + 1$ errors can be found. Let $\Delta_{2t+1} = \alpha^{k_1}$, $\Delta_{2t+2} = \alpha^{k_2}$ whenever they are nonzero. The cases to be considered are

1. $\lambda_i^{(2t+2)} = \lambda_i^{(2t)} - \alpha^{k_1}\alpha^{-i}b_i^{(2t)}$,
2. $\lambda_i^{(2t+2)} = \lambda_i^{(2t)} - \alpha^{k_2}\alpha^{-2i}b_i^{(2t)}$,
3. $\lambda_i^{(2t+2)} = \lambda_i^{(2t)} - \alpha^{k_1}\alpha^{-i}b_i^{(2t)} - \alpha^{k_2}\alpha^{-2i}b_i^{(2t)}$,
4. $\lambda_i^{(2t+2)} = \lambda_i^{(2t)} - \alpha^{k_2}\alpha^{-k_1}\alpha^{-2i}\lambda_i^{(2t)} - \alpha^{k_1}\alpha^{-i}b_i^{(2t)}$.

Each of these cases is to be searched over $k_1 = 0, \cdots, q - 2$; $k_2 = 0, \cdots, q - 2$ for a solution with $\lambda_i^{(2t+2)} = 0$ for exactly $t + 1$ values of i. With these values for the unknowns, the polynomial $\Lambda^{(2t+2)}(x)$ must have degree $t + 1$. Then $\Lambda^{(2t+2)}(x)$ can be used to recursively extend the syndromes, starting from the known syndromes, and using

$$S_j = -\sum_{k=0}^{n-1} \Lambda_k^{(2t+2)} S_{k-j} \qquad j = 2t + 1, \cdots, n.$$

An inverse Fourier transform completes the decoding.

Searching through the four cases above appears quite tedious to the reader, but is very orderly and simple in structure, and shift register circuits can be easily designed to search for a solution. One can organize the search in a variety of ways. Among the possibilities, one of the less obvious is a histogram approach. For example, with case 4, for each value of k_1 prepare a histogram of $(\lambda_i^{(2t)} - \alpha^{k_1}\alpha^{-i}b^{(2t)})^{-1}(\alpha^{-2i}\lambda_i^{(2t)})$. Any component k_1 of the histogram that takes the value $t + 1$ corresponds to a possible $t + 1$ error pattern.

The decoder can be further extended to decode $t + \nu$ errors. Although the equations become lengthy, it seems that such an approach may be practical out to $t + 3$ or $t + 4$ depending on the blocklength of the code.

Now consider binary codes. These differ from Reed-Solomon codes in that the decoder can be simplified as described below, and also because many of the $t + \nu$ error patterns found may correspond to nonbinary error patterns and so must be discarded. When treating binary cyclic codes, we will make use of the fact that the Berlekamp-Massey algorithm of Theorem 5 can be simplified because $\Delta_r = 0$ for all even r. Published proofs of this important fact [7, 12] are quite lengthy. An easy proof is given in the following theorem.

Theorem 6 In GF(2^m), suppose for any linear feedback shift register $\Lambda(x)$, and any sequence $S_1, S_2, \cdots, S_{2\nu-1}$ satisfying $S_{2j} = S_j^2$, that

$$S_j = -\sum_{i=1}^{n-1} \Lambda_i S_{j-i} \qquad j = 2\nu - 1, \cdots, \nu.$$

Define the next member of the sequence by

$$S_{2\nu} = -\sum_{i=1}^{n-1} \Lambda_i S_{2\nu-i};$$

then $S_{2\nu} = S_\nu^2$.

Proof

$$S_\nu^2 = \left(\sum_{i=1}^{n-1} \Lambda_i S_{\nu-i}\right)^2 = \sum_{i=1}^{n-1} \Lambda_i^2 S_{\nu-i}^2 = \sum_{i=1}^{n-1} \Lambda_i^2 S_{2\nu-2i}.$$

On the other hand,

$$S_{2\nu} = -\sum_{k=1}^{n-1} \Lambda_k S_{2\nu-k} = \sum_{k=1}^{n-1}\sum_{i=1}^{n-1} \Lambda_k \Lambda_i S_{2\nu-k-i}.$$

But by symmetry every term in the sum with $i \neq k$ appears twice, so in GF(2^m) those two terms add to zero. Hence, only the diagonal terms contribute, and

$$S_{2\nu} = \sum_{i=1}^{n-1} \Lambda_i^2 S_{2\nu-2i},$$

which agrees with the expression for S_ν^2 and so proves the theorem.

Thus, for even r, Δ_r is zero and we can analytically combine two iterations to give, for even r,

$$\Lambda^{(r)}(x) = \Lambda^{(r-2)}(x) - \Delta_{r-1} x B^{(r-2)}(x),$$

$$B^{(r)}(x) = (1 - \delta_{r-1})x^2 B^{(r-2)}(x) + \delta_{r-1} x^2 \Delta_{r-1}^{-1} \Lambda^{(r-2)}(x).$$

Now suppose that we have a binary BCH code of designed distance $2t + 1$, and we wish to correct all patterns of $t + 1$ (or fewer) errors whenever they are uniquely decodable. The only measurement data available to the decoder are contained in the $2t$ syndromes $S_1, S_2, \cdots, S_{2t}$. All other frequencies either can take on arbitrary values or are completely determined from the syndromes by the constraints. The algorithm can be iterated again to give

$$\Lambda^{(2t+2)}(x) = \Lambda^{(2t)}(x) - \Delta_{2t+1} x B^{(2t)}(x).$$

Transform the frequency domain vector $\Lambda_j^{(2t+2)}$ by transforming the components on the right to get

$$\lambda_i^{(2t+2)} = \lambda_i^{(2t)} + \Delta_{2t+1}\alpha^{-i} b_i^{(2t)},$$

and suppose a pattern of t or fewer errors was not found.

Prepare a histogram of $\alpha^i \lambda_i^{(2t)}/b_i^{(2t)}$ over the nonzero components of GF(q). If one component (or more) of the histogram equals $t + 1$, this corresponds to a candidate error pattern for that value of Δ_{t+1}. For each of these candidates, the corresponding polynomial $\Lambda^{(2t+1)}(x)$ can then be used to extend the syndromes in the frequency domain. Those cases that do not satisfy the conjugacy constraints can be discarded at this point. An inverse Fourier transform for each candidate gives an error pattern. If it is unique, it is the correct error pattern.

Next consider a binary code for which the parity frequencies are not contiguous. An example is the (63, 28, 15) binary cyclic code with parity frequencies C_1, C_3, C_5, C_7, C_9, C_{11}, and C_{21}. This code should be preferred to the (63, 24, 15) BCH code because of a superior rate, but the BCH code might be chosen because of its well-known decoding algorithms. However, with a little extra complexity, we can modify a frequency domain BCH decoder to handle the (63, 28, 15) code. Using the procedure discussed above, all patterns of seven or fewer errors that agree with the twelve contiguous parity frequencies are found. Then S_{21} is computed for each of these candidates. Only one will agree with the measured value of S_{21}.

The same ideas apply to a BCH code with more than $t + 1$ errors. To extend the decoder more than one error beyond the BCH bound requires more complex equations, but to go a small distance they are still quite manageable. For $t + 2$ errors,

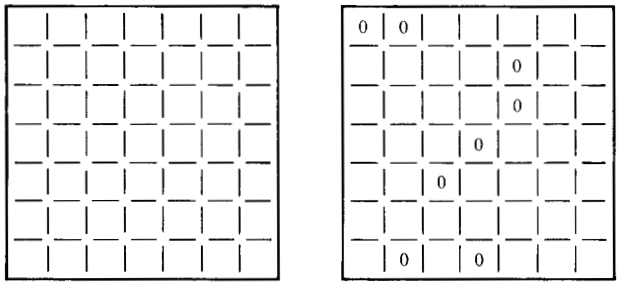

Figure 5 Two-dimensional spectrum over Galois field GF(8): (a) unconstrained spectrum, (b) constrained spectrum.

$$\Lambda^{(2t+2)}(x) = \Lambda^{(2t)}(x) - \Delta_{2t+1} x B^{(2t)}(x),$$

$$B^{(2t+2)}(x) = (1 - \delta_{2t+1}) x^2 B^{(2t)}(x) + \delta_{2t+1} x^2 \Delta_{2t+1}^{-1} \Lambda^{(2t)}(x),$$

and

$$\Lambda^{(2t+4)}(x) = (1 - \delta_{2t+1}\Delta_{2t+3}\Delta_{2t+1}^{-1} x^2)\Lambda^{(2t)}(x) - (\Delta_{2t+1} x + (1 - \delta_{2t+1})\Delta_{2t+3} x^3) B^{(2t)}(x).$$

Take the Fourier transform

$$\lambda_i^{(2t+4)} = (1 - \delta_{2t+1}\Delta_{2t+3}\Delta_{2t+1}^{-1}\alpha^{-2i})\lambda_i^{(2t)} - (\Delta_{2t+1}\alpha^{-i} + (1 - \delta_{2t+1})\Delta_{2t+3}\alpha^{-3i}) b_i^{(2t)}$$

and search for values of the unknowns that have $t + 1$ or $t + 2$ components of $\lambda_i^{(2t+4)}$ equal to zero. Many of these will correspond to nonbinary error patterns and so must be rejected later.

Finally, we come to Goppa codes. Let $G(x)$ be the Goppa polynomial, and from the spectrum of the received signal R_j, let the syndromes be computed by

$$S_j = \sum_{i=0}^{n-1} H_{i-j} R_i$$

(or instead, in the time domain, multiply to find $g_i^{-1} r_i = h_i r_i$). Decode beyond the designed distance just as for the Reed-Solomon code. The simplifications to the Berlekamp-Massey algorithm for BCH codes due to conjugacy constraints do not apply. In general, one can expect that many candidate error patterns with $t + 1$ errors will initially be found. For each of these, the filtered syndromes can be extended; then the inverse Fourier transform is taken and multiplied by g_i. No more than one pattern of $t + 1$ errors can be binary. Alternatively, working in the frequency domain, the filtered syndromes can be inverse-filtered using **G** as the tap weights of a finite impulse response filter. This convolves **G** with the filtered syndromes. The filter output must satisfy the conjugacy constraints or be rejected. Only one candidate error pattern will survive this test. An inverse Fourier transform gives the time domain error pattern.

Codes based on multidimensional transforms

Multidimensional Fourier transforms also can be used to define error control codes. We shall consider several examples, but the most familiar example is the two-dimensional product code. This is a two-dimensional array of elements from GF(q) such that every row is a codeword in a code $\mathscr{C}_1$ and every column is a codeword in a code $\mathscr{C}_2$. A cyclic product code is a product code in which the rows and columns are from cyclic codes $\mathscr{C}_1$ and $\mathscr{C}_2$. To ensure that the cyclic product code is actually cyclic, one imposes the condition that the number of rows and the number of columns are relatively prime. But, for a general multidimensional transform code, the dimensions need not be relatively prime.

Multidimensional transforms have been used for the study of error control codes in the guise of the Mattson-Solomon polynomial. A treatment of cyclic product codes with two-dimensional transforms can be found in Lin and Weldon [19]. Papers by Delsarte, Goethels, and MacWilliams [20] and by Kasami, Lin, and Peterson [21] are representative of the use of multidimensional transforms. For simplicity, we will limit discussion to the two-dimensional transform.

Let $e_{ii'}$ be an $n \times n'$, two-dimensional array, which will be called a two-dimensional time function, where n and n' both divide $q^m - 1$ for some m. Let β and γ be elements of GF(q^m) of order n and n' respectively. The array

$$E_{jj'} = \sum_{i=0}^{n-1} \sum_{i'=0}^{n'-1} \beta^{ij} \gamma^{i'j'} e_{ii'}$$

will be called the two-dimensional spectrum and the indices j and j' are the frequency variables. It is obvious that

$$e_{ii'} = \frac{1}{n} \frac{1}{n'} \sum_{j=0}^{n-1} \sum_{j'=0}^{n'-1} \beta^{-ij} \gamma^{-i'j'} E_{jj'}$$

by inspection of the one-dimensional inverse transform.

We can choose $n = n' = q^m - 1$. Then $\beta = \gamma = \alpha$, a primitive element, and

$$E_{jj'} = \sum_{i=0}^{n-1} \sum_{i'=0}^{n-1} \alpha^{ij} \alpha^{i'j'} e_{ii'},$$

$$e_{ii'} = \frac{1}{n} \frac{1}{n} \sum_{j=0}^{n-1} \sum_{j'=0}^{n-1} \alpha^{-ij} \alpha^{-i'j'} E_{jj'}.$$

Consider a two-dimensional spectrum over GF(q). For definiteness we will illustrate with GF(8) and $n = 7$ as shown in Fig. 5(a). Each square in the grid contains an octal symbol. We define a code by selecting a set of $N - K$ of these components to be (two-dimensional) parity frequencies, which are constrained to be zero as in Fig. 5(b).

The remaining set of K components are filled with K information symbols, and then the inverse transform (two-dimensional) is taken. The time function is the codeword corresponding to the information symbols. Clearly, this is a linear code, and any choice of the parity frequencies defines another linear code. In general, these codes are not cyclic codes.

If the code is in a subfield of GF(q) [GF(2) is the only subfield of GF(8)], one must restrict the set of codewords to those that have only components in the subfield. This is a subfield-subcode. One could also extend the idea of an alternant code to a multidimensional alternant code by multiplying by a two-dimensional template before extracting the subfield-subcode.

For an example, as shown in Fig. 6(a), choose all of the elements in a set of vertical stripes and a set of horizontal stripes to be parity frequencies. All the two-dimensional time domain functions with these frequencies equal to zero are the codewords. That is,

$$\sum_{i=0}^{n-1} \sum_{i'=0}^{n-1} \alpha^{ij} \alpha^{i'j'} e_{ii'} = 0$$

for each parity frequency jj'. This can be interpreted in another way by defining the two-dimensional polynomial

$$e(x, y) = \sum_{i=0}^{n-1} \sum_{i'=0}^{n-1} e_{ii'} x^i y^{i'}$$

so that the code satisfies $e(\alpha^j, \alpha^{j'}) = 0$ for every j and every j' that are parity frequencies. Since the parity frequencies were defined on vertical and horizontal stripes, we have

$$e(\alpha^j, y) = 0,$$

$$e(x, \alpha^{j'}) = 0$$

for every j and every j' that are parity frequencies. But this says that for fixed i, $e_{ii'}$ is a cyclic code and for fixed i', $e_{ii'}$ is a cyclic code. That is, $e_{ii'}$ is a product code. Product codes were studied by Elias [22], who showed that the minimum distance is the product of the minimum distances of the two codes. It was proved by Burton and Weldon [23] that if dimensions n and n' are relatively prime, then the product code of two cyclic codes is equivalent to a cyclic code.

If we take the stripes of parity frequencies to be contiguous, then we have a code that is the product of two Reed-Solomon codes. Figure 6(b) illustrates a (49, 25) $d = 9$ code over GF(8) defined spectrally. Each of the 25 information symbols can be loaded with an octal information character, and the result is transformed to the time domain to obtain the codeword.

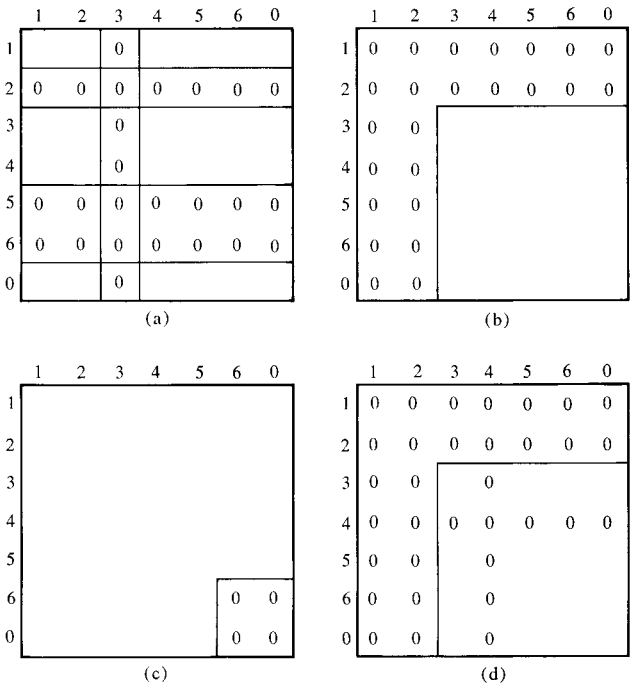

Figure 6 Spectra of some codes over Galois field GF(8): (a) product of cyclic codes; (b) product of Reed-Solomon codes; (c) dual of a product code; (d) product of (7, 4) BCH codes.

Table 2 Structure of two-dimensional spectrum over GF(8).

			Bit content				*Bit content*
$C_{1,1}$	$C_{2,2}$	$C_{4,4}$	3	$C_{3,3}$	$C_{6,6}$	$C_{5,5}$	3
$C_{1,2}$	$C_{2,4}$	$C_{4,1}$	3	$C_{3,5}$	$C_{6,3}$	$C_{5,6}$	3
$C_{2,1}$	$C_{4,2}$	$C_{1,4}$	3	$C_{5,3}$	$C_{3,6}$	$C_{6,5}$	3
$C_{1,3}$	$C_{2,6}$	$C_{4,5}$	3	$C_{0,1}$	$C_{0,2}$	$C_{0,4}$	3
$C_{3,1}$	$C_{6,2}$	$C_{5,4}$	3	$C_{1,0}$	$C_{2,0}$	$C_{4,0}$	3
$C_{1,5}$	$C_{2,3}$	$C_{4,6}$	3	$C_{0,3}$	$C_{0,6}$	$C_{0,5}$	3
$C_{5,1}$	$C_{3,2}$	$C_{6,4}$	3	$C_{3,0}$	$C_{6,0}$	$C_{5,0}$	3
$C_{1,6}$	$C_{2,5}$	$C_{4,3}$	3	$C_{0,0}$			1
$C_{6,1}$	$C_{5,2}$	$C_{3,4}$	3				

The same structure can be used to obtain a code over GF(2) by selecting only those codewords that are binary. To do this constructively in the frequency domain, only an independent set of frequencies may be specified. Theorem 3 is easily extended to a two-dimensional version which requires that

$$C_{jj'}^2 = C_{(2j \bmod n)(2j' \bmod n)},$$

from which we can construct Table 2. Each row of the table shows a constrained set of frequencies. Any member of the row can be chosen as parity or as an arbitrary

information symbol. The remaining symbols in a row are not arbitrary. The frequency $C_{0,0}$ can only be 0 or 1 because it is its own square. The remaining information symbols are octal. Altogether 49 bits specify the spectrum, but of course some of these are parity and contain no information.

Figure 6(d) shows the specialization of Fig. 6(b) to a binary code. There are only 16 open frequencies which, because of the constraints, can encode 16 bits. This is a consequence of the fact that row 1 and column 1 have their parity symbols scattered among different rows of Table 2. The code is an unimpressive (49, 16, 9) code.

The second case, illustrated in Fig. 6(c), is a dual to the idea of a product code. A rectangle a frequencies high and b frequencies wide is chosen for the set of parity frequencies. It is easily seen that the minimum distance satisfies

$$d \geq 1 + \min(a, b).$$

Hence the example gives (49, 45, 3) code over GF(8). The binary subfield subcode is a (49, 39) $d \geq 3$ code.

In the next two sections, we will make use of two-dimensional codes to introduce some new codes with special properties.

Fast BCH codes

We have seen that for any BCH code, the encoder/decoder involves two Fourier transforms, possibly realized as a Chien search or as a syndrome computer. If n is composite, then a fast algorithm can be used for the Fourier transforms so as to reduce considerably the computational load. However, the fast Fourier transform requires some adjustment terms when the factors are nonbinary. (Finite field transforms normally have nonbinary factors.) This is only a minor problem, but it does disrupt the otherwise orderly organization of the calculations. If it can be eliminated at no cost, it should be.

To see the adjustment terms, consider the Fourier transform

$$E_j = \sum_{i=0}^{n-1} \alpha^{ij} e_i,$$

and suppose $n = n'n''$. Replace each of the indices by a coarse and vernier index as follows:

$$\begin{aligned} i &= i' + n'i'' \qquad i' = 0, \cdots, n' - 1, \\ &\qquad\qquad i'' = 0, \cdots, n'' - 1; \\ j &= n''j' + j'' \qquad j' = 0, \cdots, n' - 1, \\ &\qquad\qquad j'' = 0, \cdots, n'' - 1. \end{aligned}$$

Then

$$E_{n''j'+j''} = \sum_{i''=0}^{n''-1} \sum_{i'=0}^{n'-1} \alpha^{(i'+n'i'')(n''j'+j'')} e_{i'+n'i''}.$$

Expand the product in the exponent and let $\alpha^{n'} = \gamma$, $\alpha^{n''} = \beta$. The term $\alpha^{n'n''i''j'} = 1$ and can be dropped. Then

$$E_{n''j'+j''} = \sum_{i'=0}^{n'-1} \beta^{i'j'} \left[\alpha^{i'j''} \sum_{i''=0}^{n''-1} \gamma^{i''j''} e_{i'+n'i''} \right].$$

Notice that the inner sum is an $n'' \times n''$ Fourier transform for each value of i' and the outer sum is an $n' \times n'$ Fourier transform for each value of i''. The factor multiplying the inner sum is a minor nuisance. We can make it vanish simply by changing the definition of the BCH code. We will define an equivalent two-dimensional code, whose performance properties are the same as a BCH code and which circumvents the need for the extra compensation factor.

Let $n = n'n''$ where n' and n'' are relatively prime. Let the code consist of all two-dimensional GF(q)-valued time functions c_{ii}, $i = 0, \cdots, n' - 1$, $i' = 0, \cdots, n'' - 1$, such that the two-dimensional transform $\{C_{jj'}\}$ satisfies

$$C_{11} = C_{22} = C_{33} = \cdots = C_{2t,2t} = 0$$

where the subscripts are modulo n' and modulo n'', respectively. This is a linear t-error-correcting code which is different from a BCH code in only a trivial way. The rate and minimum distance are unchanged. The rate is the same because of the following theorem.

Theorem 7 The two-dimensional conjugacy class of j modulo n' and j modulo n'' has the same number of elements as the conjugacy class of j modulo $n'n''$.

Proof Let r be the smallest integer such that both $2^r j = j$ modulo n' and $2^r j = j$ modulo n'' are satisfied. Let s be the smallest integer such that $2^s j = j$ modulo $n'n''$. Then $2^r j = an'n'' + j$, and $2^s j = bn'n'' + j$ for some a and for some b. Obviously, the smallest such r and the smallest such s are identical.

We show the distance of the code is at least $2t + 1$ by showing a decoding procedure for t errors. Given a received word with two-dimensional transform $R_{jj'}$, define the syndromes $S_j = R_{(j \bmod n',\, j \bmod n'')}$, $j = 1, \cdots, 2t$. Use the Berlekamp-Massey algorithm and a recursive extension to obtain S_j, $1, \cdots, n$; and set $E_{(j \bmod n',\, j \bmod n'')} = S_j$, $j = 1, \cdots, n$. Since n' and n'' are relatively prime, every syndrome finds its own place in $E_{jj'}$. We must prove that this procedure gives the correct frequency domain error pattern if fewer than t errors occurred.

But if a single error takes place in row i_k and column i'_k, then $S_j = (\beta^{i_k}\gamma^{i'_k})^j$. The parenthesized term is a power of the primitive element α, unique for each row and column

pair. Thus, for ν errors, the syndromes are of the form

$$S_j = \sum_{k=1}^{\nu} X_k^j,$$

where X_k is a unique power of α for each error location (i_k, i_k'). Recursively extending these syndromes and folding them back into the two-dimensional spectrum gives $E_{jj'}$ if t or fewer errors took place. Finally,

$$C_{jj'} = R_{jj'} - E_{jj'},$$

and a two-dimensional inverse transform completes the decoding.

Long codes in small fields

As the blocklength increases, BCH codes become unsatisfactory for several reasons. Not only does d/n vanish if the rate is fixed, but at the same time the decoding computations must take place in an ever larger field. The alternant codes represent one way to modify BCH codes to improve minimum distance and so offset the first disadvantage. The second disadvantage, however, has not received much attention. We will develop this problem here, and give some early steps toward a solution.

A practical decoder for a BCH code of blocklength $n = q^m - 1$ requires computations in GF(q^m). If n is large, this is a large Galois field. We will describe some codes of large blocklength that can be decoded in a small Galois field. Although the rate of these codes is inferior to BCH codes of the same n and d, their lesser complexity may make them the only affordable choice in some applications.

We will use a two-dimensional code with $n = 2^m - 1$ rows and the same number of columns. Hence, the blocklength of the code is n^2, but we can hope to do all of the decoding with computations in the field GF(2^m).

Before defining the codes, we first discuss decoding procedures and a two-dimensional version of the BCH bound. The codes will be defined to fit the desired decoding procedure. Let a single error occur at row i and column i', let the row locator be $X_1 = \alpha^i$, and let the column locator be $Y_1 = \alpha^{i'}$. Then the syndrome $S_{jj'}$ is

$$S_{jj'} = \alpha^{ij}\alpha^{i'j'} = X_1^j Y_1^{j'},$$

and if ν errors occur, then

$$S_{jj'} = \sum_{k=1}^{\nu} X_k^j Y_k^{j'}.$$

Suppose the syndromes $S_{11}, S_{12}, S_{13}, \cdots, S_{1,2t}$ are known and $\nu \leq t$. Then

$$S_{11} = X_1Y_1 + X_2Y_2 + \cdots + X_\nu Y_\nu,$$

$$S_{12} = X_1Y_1^2 + X_2Y_2^2 + \cdots + X_\nu Y_\nu^2,$$

$$\bullet$$
$$\bullet$$
$$\bullet$$

$$S_{1,2t} = X_1Y_1^{2t} + X_2Y_2^{2t} + \cdots + X_\nu Y_\nu^{2t}.$$

This set of equations is familiar from the decoding of a Reed-Solomon code. There is one difference; here the Y_k need not be distinct because several errors might occur in the same column. However, it is a simple matter to combine the terms with the same Y_k to obtain a similar set of equations with smaller ν and this smaller ν also satisfies $\nu \leq t$. Then, the Berlekamp-Massey algorithm followed by recursive extension will yield $S_{11}, S_{12}, \cdots, S_{1n}$. Because errors can occur in the same column the procedure does not uniquely give (X_k, Y_k) $k = 1, \cdots, \nu$, but does give partial information which can be used in decoding.

In general, we have the following.

Theorem 8 Suppose that $\nu \leq t$ errors occur, and for any integers m_0, m_0', a, a', the syndromes $S_{m_0+al,m_0'+a'l}$, $l = 1, \cdots, 2t$, are known. Then these uniquely define the syndromes $S_{m_0+al,m_0'+a'l}$, $l = 1, \cdots, n$.

Proof

$$S_{m_0+al,m_0'+a'l} = \sum_{k=1}^{\nu} X_k^{m_0+al} Y_k^{m_0'+a'l} = \sum_{k=1}^{\nu} (X_k^{m_0} Y_k^{m_0'})(X_k^a Y_k^{a'})^l.$$

Let ν' be the number of distinct $X_k^a Y_k^{a'}$ over k and let $\bar{Y}_{k'}$, $k' = 1, \cdots, \nu'$, denote these. Let $\bar{X}_{k'}$ denote the sum of the factors multiplying $\bar{Y}_k^l$ in each equation. (It is the same for each l.) Then

$$S_{m_0+al,m_0'+a'l} = \sum_{k'=1}^{\nu'} \bar{X}_{k'} \bar{Y}_{k'}^l \qquad l = 1, \cdots, 2t,$$

where the $\bar{Y}_{k'}$ are now distinct and $\nu' \leq t$. The Berlekamp-Massey algorithm followed by recursive extension will produce the remaining syndromes

$$S_{m_0+al,m_0'+a'l} = \sum_{k'=1}^{\nu'} \bar{X}_{k'} \bar{Y}_{k'}^l \qquad l = 1, \cdots, n.$$

Hence, by this theorem, any $2t$ syndromes in a straight line (horizontal, vertical, or at any angle) can be extended to all syndromes in that line. Further, because of conjugacy constraints, each of these new syndromes also determines all syndromes in its conjugacy class. We will return to this point in the examples below.

Now let us see how the BCH bound generalizes to two (or more) dimensions. Suppose that we had $2t$ contiguous syndromes anywhere in the first row. These can be extended to give all syndromes in the first row. Similarly $2t$

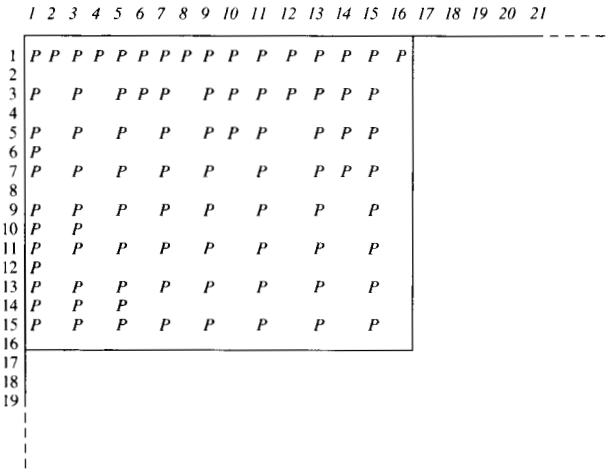

Figure 7 Two-dimensional parity frequencies.

contiguous syndromes anywhere in the second row can be extended to give all syndromes in the second row. Further, $2t$ contiguous syndromes anywhere in each of the first $2t$ rows can be extended to give all syndromes in each of the first $2t$ rows, and hence $2t$ contiguous syndromes in every column. These can be extended to give all syndromes and hence suffice to give the error pattern. The simplest example is the case of a square array of $2t \times 2t$ known syndromes. The general situation is as follows.

Theorem 9 Given any a, a' and m_l, $l = 1, \cdots, 2t$, the set of $(2t)^2$ syndromes

$$S_{a(m_l+l'),a'(m_l+l')+l} \qquad l' = 1, \cdots, 2t;\ l = 1, \cdots, 2t$$

uniquely determines the error pattern provided $\nu \leq t$ errors took place, and a is relatively prime to n.

Proof Apply Theorem 8 for each l to determine the syndromes $S_{a(m_l+l'),a'(m_l+l')+l}$, $l' = 1, \cdots, n$; $l = 1, \cdots, 2t$. Now, because l' ranges over all n values, we can redefine the index l' to absorb m_l so that the known syndromes are $S_{al',a'l'+l}$, $l' = 1, \cdots, n$; $l = 1, \cdots, 2t$. Apply Theorem 8 again for each l' to determine the syndromes $S_{al',a'l'+l}$, $l' = 1, \cdots, n$; $l = 1, \cdots, n$. We can now redefine the index l to absorb $a'l'$ so that we know $S_{al',l}$, $l' = l, \cdots, n$; $l = 1, \cdots, n$. Since a is relatively prime to n, the index al' ranges over all n values. Hence, all syndromes are determined, and so is the error pattern.

Based upon Theorem 9, for each a, a', m_l, $l = 1, \cdots, 2t$, we can define a two-dimensional code $\mathscr{C}$ as the set of arrays $c_{ii'}$ such that the two-dimensional transform satisfies

$$C_{a(m_l+l'),a'(m_l+l')+l} = 0 \qquad l = 1, \cdots, 2t;\ l' = 1, \cdots, 2t.$$

This code has minimum distance at least $2t + 1$ provided a is relatively prime to n.

We give an example of a binary code defined as a square two-dimensional code in the field GF(2^m); take $t = 8$, $n = 255$, so the blocklength is $255^2 = 65\,025$. We will work through the selection of parity frequencies so that all of the parity frequencies in the block $j = 1, \cdots, 2t$; $j' = 1, \cdots, 2t$ can be computed. Theorem 9 then guarantees that the remaining syndromes can be computed.

First take $S_{11}, S_{12}, \cdots, S_{1,2t}$ as parity frequencies. Each of these is in a different conjugacy class, and each class has eight elements, so each of these parity frequencies is equivalent to eight parity bits. These can be extended to $S_{1j'}, j' = 1, \cdots, n$, if at most t errors occurred and then by the conjugacy constraints $S_{1j'}$, $S_{2j'}$, $S_{4j'}$, $S_{8j'}$, and $S_{16j'}$, $j' = 1, \cdots, n$ are all known. Next take $S_{31}, S_{51}, S_{61}, S_{71}, S_{91}, \cdots, S_{15,1}$ as parity frequencies. This adds 11×8 more parity bits and determines $S_{j1}, S_{j2}, S_{j4}, S_{j8}, S_{j16}, j = 1, \cdots, n$. Continue in this way to choose all the parity frequencies shown in Fig. 7. These determine the remaining frequencies in the $2t \times 2t$ corner and hence all of the frequencies if at most t errors occurred. Each parity frequency is equivalent to eight parity bits. The code is a (65 025, 64 337, 17) code. Its virtue is that it is easily decoded despite its blocklength.

We first describe a conceptual frequency domain decoder; later we simplify this by bypassing many of the Fourier transforms.

Given a received word, compute its two-dimensional Fourier transform. This requires 510 two-hundred-and-fifty-five-point Fourier transforms. Perform a Berlekamp-Massey algorithm along the first row, recursively extend, and use conjugacy constraints to fill in rows 2, 4, 8, and 16. Do the same along the first column to find columns 2, 4, 8, and 16. Repeat for row 3, then column 3 and so on. When $2t$ rows are complete, then all columns can be found. An inverse two-dimensional Fourier transform gives the error pattern.

A simpler procedure is as follows. Compute the two-dimensional Fourier transform only at the 86 parity frequencies. Each of these is an eight-bit number. Insert these at the appropriate positions of a 16 by 16 array of numbers representing the 16 by 16 frequencies in the upper left corner. Now decode the first row, extending syndromes and using conjugacy constraints to fill in all possible entries in the 16×16 array. Take the inverse Fourier transform of the first row. The nonzero locations specify columns in the time domain codewords at which errors occur. Each nonzero magnitude gives the sum of

the row locators for errors in this column. Some columns with three or more errors may not show up here because the row locators add to zero. Discard the syndromes outside the 16 by 16 array.

Continue in this way with each odd numbered row of the sixteen rows, when necessary decoding a column just to obtain some needed syndromes through the conjugacy constraints. Each of the sixteen rows, when the inverse transform is taken, has nonzero values only in the eight (or fewer) columns containing errors. Identify these eight columns. In each column the transforms in the sixteen rows provide sixteen magnitudes. One such set of magnitudes can be written in terms of the row error locators for that column,

$$T_1 = X_1 + X_2 + \cdots + X_\nu,$$

$$T_2 = X_1^2 + X_2^2 + \cdots + X_\nu^2,$$

•

•

•

$$T_{10} = X_1^{16} + X_2^{16} + \cdots + X_\nu^{16},$$

where ν is the number of errors in that column. Since $\nu \leq 8$, this set can be decoded in the same way to find the rows in which this column has errors.

Altogether, this decoder requires the computation of 86 Fourier transforms, and 24 passes through the basic decoding algorithm, each such pass consisting of a Berklekamp-Massey algorithm, a recursive extension, possible computation of conjugacy relations, and a 255-point inverse Fourier transform. All data paths are eight bits wide, Galois field computations are eight bits by eight bits, and most of the computations simply re-exercise the same procedures, and so can use the same hardware.

References

1. J. M. Pollard, "The Fast Fourier Transform in a Finite Field," *Mathematics of Computation* **25,** No. 114, 365–374 (1971).
2. W. C. Gore, "Transmitting Binary Symbols with Reed-Solomon Codes," *Proceedings of Princeton Conference on Information Sciences and Systems*, Princeton, NJ, 1973, pp. 495–497.
3. A. Michelson, "A Fast Transform in some Galois Fields and an Application to Decoding Reed-Solomon Codes," *IEEE Abstracts of Papers—IEEE International Symposium on Information Theory*, Ronneby, Sweden, 1976, p. 49.
4. A. Lempel and S. Winograd, "A New Approach to Error Correcting Codes," *IEEE Trans. Inf. Theory* **IT-23,** 503–508 (1977).
5. R. T. Chien and D. M. Choy, "Algebraic Generalization of BCH-Goppa-Helgert Codes," *IEEE Trans. Inf. Theory* **IT-21,** 70–79 (1975).
6. H. F. Mattson and G. Solomon, "A New Treatment of Bose Chaudhuri Codes," *J. Soc. Indus. Appl. Math.* **9,** No. 4, 654–699 (1961).
7. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, Second Edition, The M.I.T. Press, Cambridge, MA, 1972.
8. R. T. Chien, "A New Proof of the BCH Bound," *IEEE Trans. Info. Theory* **IT-18,** 541 (1972).
9. I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *J. Soc. Indus. Appl. Math.* **8,** 300–304 (1960).
10. D. Mandelbaum, "On Decoding of Reed-Solomon Codes," *IEEE Trans. Info. Theory* **IT-17,** 707–712 (1971).
11. R. H. Paschburg, "Software Implementation of Error-Correcting Codes," MS Thesis, University of Illinois, Urbana, IL, 1974.
12. E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill Book Co., Inc., New York, 1968.
13. J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. Info. Theory* **IT-15,** 122–127 (1969).
14. G. D. Forney, Jr., "On Decoding BCH Codes," *IEEE Trans. Info. Theory* **IT-11,** 549–557 (1965).
15. H. J. Helgert, "Alternant Codes," *Info. Control* **26,** 369–381 (1974).
16. V. C. Goppa, "A New Class of Linear Error-Correcting Codes," *Prob. Peredach. Infor.* **6,** 24–30 (1970).
17. P. Delsarte, "On Subfield Subcodes of Modified Reed-Solomon Codes," *IEEE Trans. Info. Theory* **IT-21,** 575–576 (1975).
18. C. R. P. Hartmann, "Decoding Beyond the BCH Bound," *IEEE Trans. Info. Theory* **IT-18,** 441–444 (1972).
19. S. Lin and E. J. Weldon, Jr., "Further Results on Cyclic Product Codes," *IEEE Trans. Info. Theory* **IT-16,** 452–459 (1970).
20. P. Delsarte, J.-M. Goethals, and F. J. MacWilliams, "On Generalized Reed-Muller Codes and Their Relatives," *Info. Control* **15,** 403–442 (1970).
21. T. Kasami, S. Lin, and W. W. Peterson, "Polynomial Codes," *IEEE Trans. Info. Theory* **IT-14,** 807–814 (1968).
22. P. Elias, "Error Free Coding," *IRE Trans. Info. Theory* **IT-4,** 29–37 (1954).
23. H. O. Burton and E. J. Weldon, Jr., "Cyclic Product Codes," *IEEE Trans. Info. Theory* **IT-11,** 433–439 (1965).

Received July 7, 1978; revised October 24, 1978

The author is located at the IBM Federal Systems Division laboratory, Owego, New York 13827.