

On correcting bursts (and random errors) in vector symbol (n, k) cyclic codes

John J. Metzner

Pennsylvania State University

Department of Computer Science and Engineering

University Park, Pennsylvania 16802

Abstract. Simple methods are shown for correcting bursts of large size and bursts combined with random errors using vector symbols and primarily vector XOR and feedback shift register operations. One result is that any (n, k) cyclic code with minimum distance > 2 can correct all full error bursts of length $n-k-1$ or less if the error vectors are linearly independent. If the bursts are not full but contain some error-free components the capability of correcting bursts up to $n-k-1$ or less is code-dependent. The techniques often work when there is linear dependence. For the case where most errors are in a burst but a small number of errors are outside, the solution, given error-correcting capability, can be broken down into a simple solution for the small number of outside errors, followed by a simple subtraction to reveal all the error values in the burst part.

Index terms: Burst error correction, Cyclic codes, vector symbol decoding

I. INTRODUCTION

It is well-known that any binary (n, k) cyclic code is capable of filling in any cyclic burst of $n-k$ erasures, given the other bits are error-free. However, if there are bit errors in unknown positions, the Rieger bound [1] on the ability to correct any cyclic error burst of length b requires $n-k \geq 2b$, only half the limit for erasure correction.

Gallager's method [2] of finding the shortest burst that could have occurred can often find bursts of greater length than the Rieger bound. However, if a burst of length significantly above the Rieger bound occurs, there is a rather high probability that a burst of shorter length has the same syndrome, giving a wrong result.

Burst correction can be done with a very simple feedback shift register circuit. Gallager's method requires two passes, one to find the shortest burst and one to trap it.

With nonbinary symbols, the prospect of correcting greater bursts is more promising. In [3], Reed-Solomon codes have been shown able to decoded bursts of length up to $n-k-2$, not always, but with high probability. The decoding in this case uses field operations and is much

more complex.

Another approach is to use binary vectors as symbols, with operations primarily vector XOR. Vector symbol coding/decoding [4-6] is a technique for the outer code of a concatenated code scheme. The technique is most effective when symbol errors are linearly independent with high probability. Individual invertible transformation of data within each symbol can be employed, different for different symbols, so that error events within a symbol are inverse transformed into random patterns that, as vectors, are likely to be linearly independent. A data symbol could be part of an inner code, which can attempt to correct the errors and then inverse transform the symbol data decision. An inner code with good correction capability, possibly using combined coding and modulation, could leave pseudo-random error patterns when it fails. This might remove the need for data transformation.

The use of vector XOR also has proved of value in packet-based decoding, where each symbol can be a packet or a large part of a packet. In reliable multicasting [7-10], where spare packets called “repair” packets are sent, the repair packets are derived either by vector XOR or Reed-Solomon codes. These techniques are mostly erasure decoding.

Cases where the error positions are unknown are more challenging. In Vector Symbol decoding, a “1” in some position p in an error location vector (ELV) indicates position p is almost surely correct, with certainty if the symbol errors are linearly independent vectors. The property that certain symbols can be “declared correct” or “verified” is inherent in Vector Symbol Decoding, and has been amplified in [11] and [12]. It is a property of great potential use. In [12], the use of verification with low density codes and vector XOR operations allows simple decoding. Majority-logic-like codes in [11] and short constraint length vector symbol convolutional codes in [13,14] are also somewhat low density and allow some simplicity.

For burst error correction, [15] shows certain classes of Vector Symbol cyclic codes that can correct all bursts where the errors are linearly independent, up to length $b = n-k-1$. This was shown possible only for a limited set of cyclic codes, all at rate less than $\frac{1}{2}$, often with not very good minimum distance. Some higher rate codes were also shown, at a small reduction in b . Decoding is with feedback shift registers, with almost the same operations as regular burst error correction, except that r -bit vector XOR replaces single-bit modulo two addition, and the register

elements must each hold r bits.

Use of the ELV is not employed in [15]. The ELV has some important properties: 1) A study of its properties shows the ability to use a much wider class of cyclic codes for burst error correction up to $n-k-1$ than described in [15]. 2) For cases where a burst error of length $< n-k-1$ cannot be found, Vector symbol decoding of a random error pattern can still be employed. The ELV can locate the error positions and allow solution of their values. Computation of the ELV adds some complexity, but feedback shift registers can help by computing syndromes and in simplifying solution for the error values. If the pattern is a burst, shifting the register to the proper position can discover all vector values directly. If the pattern is a burst plus a small number of outside errors, shifting to cover the burst part can allow quick computation of all error values.

A full error burst is a burst where all symbols in the burst are incorrect. For the case of a full burst, the current paper presents a much more general result than in [15]. It is shown that all cyclic codes whose minimum distance is greater than 2 are capable of correcting all full error bursts of length $n-k-1$ or less, given that all errors are linearly independent. Simple shift register operations are used as in [15]. The technique can work in many cases even when not all errors are linearly independent.

At first thought it would seem that if one could decode all full error bursts up to length $n-k-1$, one should also be able to correct all such bursts even if some of the symbols in the burst happen to be correct. Unfortunately, this is not the case in general, as will be shown.

II. VECTOR SYMBOL DECODING WITH CYCLIC CODES

A. Vector Symbol decoding

Vector Symbol decoding involves deriving an error location vector (ELV). Given an (n, k) binary code with binary check matrix H , the decoder can use H , operating on r -bit symbol inner code decisions, to derive a $(n-k)$ by r syndrome matrix S , as in equation (1).

$$[S] = \begin{matrix} & \begin{matrix} H \end{matrix} & \cdot & \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \dots \\ y_{n-1} \end{bmatrix} \\ \begin{matrix} (n-k) \\ \text{by } r \end{matrix} & & & \begin{matrix} n \text{ by } r \end{matrix} \end{matrix} \quad (1)$$

The Vector symbol decoder then performs column operations on S and discovers both the rank of S and null combinations, which span the subspace of the row space of H that is the orthogonal complement of the space of the error vectors. The symbol positions that are "1" in any null combination are very likely to be correct (certain given the errors are linearly independent). The logical OR of these null combinations forms the ELV. Once the ELV is found, if the rank of S matches the number of zeroes in the ELV, the error values can be solved for.

Note that a vector symbol code is not simply an interleaved code, in the same sense that turbo codes are not simply interleaved codes. In a turbo code, the data is scrambled over a long period involving many symbols. In a vector symbol decoding code, the data symbols are individually scrambled into an inner code or modulator, as described in the introduction.

Most of the power of vector symbol decoding comes when the error vectors are linearly independent, but some dependency does not necessarily preclude correctability [6,11]. For randomly-placed errors, the general result for linearly independent errors is that all errors can be corrected if the error set does not cover all but zero or one position of any code word.

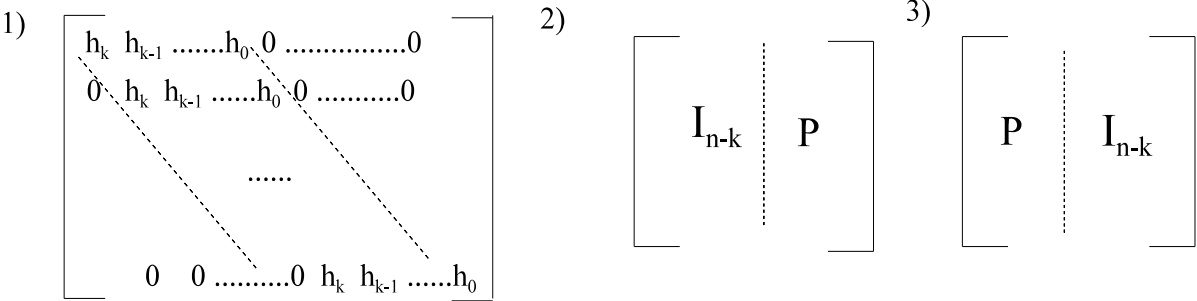
The probability that $n-k$ r -bit random vectors are dependent is roughly $2^{-[r \cdot (n-k)]}$, so for

effectiveness r should be substantially greater than $n-k$. The trend to greatly increased data transmission sizes encourages use of multi-bit symbols. The r -bit symbols may in fact represent the data in one of a series of packets, as in [12].

B. Cyclic code representation for Vector Symbol decoding

For cyclic codes a vector XOR feedback shift register can be used to compute syndromes. It will be shown that, for cyclic codes, the set of correctable error patterns can be expanded to include many burst patterns that actually do cover all but one position of a code word, without sacrificing the ability to correct all random errors in the correctable error set. Also, the use of cyclic codes greatly simplifies decoding.

Useful forms of the check matrix:



In 2), the bottom row is the coefficients of $h(X)$, high to low. In 3) the top row is the coefficients of $X^{n-k-1} h(X)$. Also, any cyclic shift of any of these forms is a possible H matrix. All have the same row space, since each can be derived from any other by elementary row operations.

The syndromes will be different for different choices of H , but since the row space is the same for all, they will all compute the same rank, the same null combinations and the same ELV. This is because the null combinations are derived from the same row space. For form 3) the syndromes are computed automatically by feeding of the incoming received sequence (after inner code processing and unscrambling) into a feedback shift register with feedback connections

determined by $g(X)$, but the register entries are r -bit vectors and addition is vector XOR.

By shifting that register with no further input, the error positions and the ELV are cyclically shifted by the amount of the shift.

We wish to find the set of correctable cyclic bursts if the error vectors are linearly independent. Additional goals are to correct random errors if there is no burst found, and to see effects of linear dependence. If the inner code adjusted received sequence experiences a burst of errors of length not exceeding $n-k$, the row space of syndrome matrix S will be the same as the space of error vectors. A burst of $n-k$ successive independent errors cannot be decoded, since, with syndrome rank $n-k$, there are no dependencies and no error-locating vector can be found. All that is known is that at least $n-k$ errors have occurred.

III. THEOREMS

Let \mathbb{C} be the set of all binary cyclic codes that have minimum distance > 2 .

Theorem 1. If a binary (n, k) cyclic code $\in \mathbb{C}$, $h(X)$ cannot have $n-k-1$ consecutive zeroes

Proof. The generator polynomial $g(X)$ has at least three nonzero terms. We can find the shortest length cyclic code and the associated $h(X)$ from $g(X) = 1+X^i + \dots+X^{n-k}$ by first adding $X^i g(X)$ to $g(X)$ to cancel X^i . If the result is $1+X^m$, then $m = n$ and $h(X) = 1+ X^i$. Since $i < n-k$, there are fewer than $n-k-1$ zeroes between 1 and i . If not, $1+ X^i$ is the first part of $h(X)$, and there are at least three nonzero terms in $(1+X^i)g(X)$, where the first is power 0 and the second is power $> i$, and the last is power $n-k+i$. Again, to cancel the second term requires a shift by less than $n-k$, so the number of zeroes in the next gap between nonzero powers in $h(X)$ is again $< n-k-1$. Continuing until power n is reached, the gap between successive nonzero powers in $h(X)$ is always $< n-k-1$.

If we found a longer length $n' > n$ cyclic code with the same $g(X)$, $g(X)h(X) = 1+X^n$ with the above computed $h(X)$ would be a code word of weight 2 in this longer code, so the code would not be in \mathbb{C} . □

To consider the capabilities of burst error correction up to some length, it is easier to start with the case that all symbols in the burst are erroneous. Call this a full burst.

Theorem 2. Let the errors be linearly independent and a full cyclic burst of length $n-k-1$, rank $n-k-1$. The error location vector will have a string of $n-k-1$ consecutive zeroes cyclically. The true burst is the only burst (full or not) of length $< n-k$ consistent with the ELV and rank of S .

Proof. Suppose the burst is in the first $n-k-1$ positions. By the cyclic code nature there is no loss in generality to assume this.

Any form of the H matrix could be used to find the error-locating vector, so assume form 1). The first $n-k-1$ rows of S will be linearly independent, and the last row will be zero, since it doesn't include any error positions. This gives the ELV from the last row:

$$\begin{array}{cccccccc} 0 & 0 & \dots\dots\dots & 0 & h_k & h_{k-1} & \dots\dots & h_0 \\ | & \text{n-k-1 zeroes} & & | & & & & \end{array}$$

By theorem 1, there is no other string of greater than $n-k-2$ zeroes. Furthermore, the rank is $n-k-1$, so there must be at least $n-k-1$ errors. To have a burst of $n-k-1$ other independent vectors including a length $n-k-2$ string, the string has a "1" on each side, so two more positions would be needed, giving a burst of length $n-k$ with $n-k-1$ errors. If the burst were j units shorter than $n-k-2$, it would have to encompass a 1 and j more zeroes at least, and would also have to be a burst of length $> n-k-1$. □

Given the $n-k-1$ known locations for the burst, there are $n-k-1$ independent equations to solve for the unknown error values. It is not actually necessary to solve these equations. Once the burst location is known, a feedback shift register can shift the syndromes to where the known burst location is trapped, and the solution then appears identical to the syndromes. In fact we will see it may not even be necessary to find the ELV or rank.

Theorem 3. For any full burst of length b , $b < n-k$, independent error symbols, there is no other burst, full or not, of length $< n-k$, that is consistent with the computed syndrome.

Proof. Suppose the burst length is instead $n-k-2$. Again, without loss of generality, we can place the burst in the first $n-k-2$ positions and use form (1) of H . Now the rank will be found to be $n-k-2$, and the last 2 rows of H will be null combinations. They are:

$$\begin{array}{cccccccc} 0 & 0 & \dots\dots\dots & h_k & h_{k-1} & \dots\dots & h_0 & 0 \\ 0 & 0 & \dots\dots\dots & .0 & h_k & h_{k-1} & \dots\dots & h_0 \\ | & \text{n-k-1 zeroes} & & & & & & | \end{array}$$

Logical OR:

$$\begin{array}{cccccccc} 0 & 0 & \dots\dots\dots & 0 & 1 & 1 & (..0 \text{ or } 1\dots) & 1 & 1 \\ | & \text{n-k-2 zeroes} & & & & & & & | \end{array}$$

The error-location vector will be a logical OR of these two rows. Recall that $h(X)$ has no internal string of more than $n-k-2$ zeroes. If there is such a string,

$$\begin{array}{cccccccc} 1 & 0 & 0 & \dots\dots\dots & .0 & 1 & \dots\dots\dots & \\ | & \text{n-k-2 zeroes} & & & & & & | \end{array}$$

the logical OR with a left shift knocks out the rightmost zero and reduces the string length to $n-k-3$ or less. Also, there now are at least two ones on both sides of this shorter burst.

Each reduction by one of the burst length will further reduce any apparent error position string in the ELV by at least one and increase the number of 1's on each side by at least 1. Thus, if the true burst length is b , the maximum other apparent error position string will remain at most $b-1$, and there will be at least $n-k-b$ 1's on each side of the false string. To give it enough error locations to match the rank, which is b , it would have to include another zero and all the 1 positions between, which would make its length at least $b-1+(n-k-b) + 1 = n-k$, which is outside the correction range. So, whenever errors are linearly independent, a full burst with length $b < n-k$ will match the rank and will always be the only possible burst of length $< n-k$ consistent with the rank of S . □

IV. EXAMPLE OF A (21,11) CODE

$$g(X) = 1+X^2+X^4+X^6+X^7+X^{10} \text{ or } 10101011001;$$

$$h(X) = 1+X^2+X^7+X^8+X^{11} \text{ or } 101000011001$$

H matrix:

Form 1	Form 2	Form 3
100110000 101000000000	1000000000 10011000010	10011000010 1000000000
010011000 010100000000	0100000000 01001100001	01001100001 0100000000
001001100 001010000000	0010000000 10111110010	10111110010 0010000000
000100110 000101000000	0001000000 01011111001	01011111001 0001000000
000010011 000010100000	0000100000 10110111110	10110111110 0000100000
000001001 100001010000	0000010000 01011011111	01011011111 0000010000
000000100 110000101000	0000001000 10110101101	10110101101 0000001000
000000010 011000010100	0000000100 11000010100	11000010100 0000000100
000000001 001100001010	0000000010 01100001010	01100001010 0000000010
000000000 100110000101	0000000001 00110000101	00110000101 0000000001

First, let's look at full bursts. Consider the first 9 positions, a full burst of $n-k-1$ errors. If the burst error vectors are linearly independent, column operations on S will yield a rank of 9. From form 1 or 2, the last syndrome will be zero, leaving an ELV equal to the last row:

$$000000000100110000101$$

To illustrate that any form leads to the same ELV, note that, in form 3, rows 2,3,8,10 add to zero in the first 9 positions. The vector XOR sum of these rows adds to the same error location vector. The rank 9 means there must be at least 9 errors. The longest string of possible error positions outside the burst is 4. For another burst to include 9 apparent error positions it has to cross over a "1" position, which makes its length $n-k = 10$.

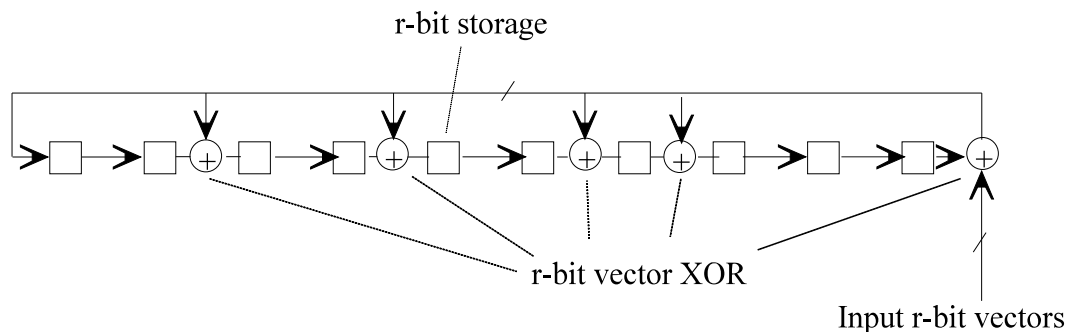
For a full burst of length b independent errors in the first b positions, the ELV is the logical OR of the last $n-k-b$ rows of form 1.

b	ELV
8	000000001101110001111
7	000000011111110011111
6	000000111111110111111
5	000001111111111111111

V. FEEDBACK SHIFT REGISTER IMPLEMENTATION

For the (21, 11) code, if the full burst positions are the last 9 positions, after all vectors were shifted in, the the following shift register reveals all the burst error values:

0 e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9



If the burst is in a different range, shifting can continue with no further input until this pattern appears. At this point there is just one zero syndrome, and the ELV is just the first row of H in form 3. If the error space has a rank of 9, there is only one null combination, namely the ELV just mentioned. If the ELV has been computed from a zero syndrome found in some different shift, it still reveals the correct burst location, and can be shifted the proper amount to reveal all the burst errors.

In general, for any case where the errors are all confined cyclically to a span of $n-k$ or less, whether independent or not, the correct error values and positions will appear in some shift of the register. Furthermore, if the burst is length b , $n-k-b$ successive shifts where the burst is trapped will show the same pattern of syndromes shifted. However, additional information, such as the ELV and possibly the rank of S , may be needed to verify the uniqueness of the error solution among the set of correctable errors.

Say there are β errors of rank ρ , all confined to a span of $n-k$ cyclically consecutive positions. Since, in a cyclic code, any $n-k$ cyclically successive columns of H must be linearly independent, the rank of the row space of S will be ρ . If $\rho = \beta$, no shift will show more than

$n-k-\beta$ zero syndrome rows, and the places where the errors are trapped will show exactly $n-k-\beta$ zero syndrome rows. In this case, the complete ELV is computable by taking any shift that has the maximum number of zero syndrome rows, and taking the vector logical OR of the $n-k-\beta$ rows of H in form 3. The ELV may verify the uniqueness of the burst solution. If needed, the rank of the e proposed error values can be computed for further verification. As shown in Theorem 3, if there is a full burst of length $b < n-k$ independent errors, there will be a unique, correct pattern found in this way. If the burst of $b < n-k$ independent errors is not full, there are cases where the solution is not unique, but the ambiguity can be recognized by the ELV, as will be described.

If the β errors are linearly dependent, $\rho < \beta$, the shift that traps all the errors still will show $n-k-\beta$ zero syndrome rows. Suppose we pick a position of the shift register that yields the maximum number of zero syndrome rows, as we would also do if the errors were linearly independent. In most cases this maximum will be where the errors are trapped, because errors outside the trap zone will contribute usually to numerous syndrome rows (half on average), so that the rare cases of error vectors adding to zero will not compensate for otherwise zero-valued syndrome rows eliminated by the outside error. From a maximum zero syndrome rows case a (partial) ELV can be computed. This may yield sufficient information about error location. If necessary, the nonzero rows of S could be investigated to find the total ELV and rank, to check consistency with the burst assumption..

If the burst is not full, the correct pattern will be seen in some shift, but it is not true that there will be a unique result for any code in \mathbb{C} , as will be discussed. For cases where the result is not unique, there usually are just 2 or, rarely, a small number greater than 2 of candidates. Since the code block is likely to cover thousands of bits, it is a small rate sacrifice to include a CRC check hidden in the data to eliminate the wrong alternative(s).

VI. REMOVING ERRORS INTERIOR TO A BURST

Assume the burst starts in the first position and form 2, $[I_{n-k} | P]$, is used to compute the error location vector (ELV).

The bottom row is the same row $h(X)$ in form 1. If the full burst is length $b < n-k$, the last $n-k-b$ rows are null combinations, and the ELV is their vector logical OR. Removing an error in position i introduces row i as an additional null combination. Outside the first $n-k$ positions, a number of 1's occur, some of which will cancel false zeroes. Thus these false zeroes tend to get reduced, but the effects will vary with the code chosen.

Let's look at the (21,11) code again.

A. Removal of one of the errors in a burst of 9 in the (21,11) code

Say the burst of length 9 had a correct entry in position 5. From form 2, the null combinations would be rows 5 and 10. The rank is 8:

Row 5	000010000010110111110
Row 10	000000000100110000101
ELV	000010000110110111111

For a rank of 8 there must be 8 zeroes. Positions 1-9 are a burst of length 9 with the proper number of zeroes. Positions 2 - 12 are a burst of length 11 with the proper number of zeroes, but this length is outside the range. So the correct burst is discovered, uniquely.

Say the correct entry was in position 2 instead of position 5.

Row 2	01000000001001100001
Row 10	000000000100110000101
ELV	010000000101111100101

Now the ELV shows two results - 1 to 9 and 3 to 11 - both bursts of length 9 with 8 ELV zeroes.

B. Why did the first case work while the second didn't?

In vector symbol decoding, decoding always is successful for independent errors if the error positions are linearly independent and don't come within one position of covering a code word of the binary code. In cyclic codes, a burst of length $n-k$ or less can't cover a code word, but it can come within one of covering a code word. If it comes within one of a code word, the position it doesn't cover always gives a false zero. This prevents unique random error correction. For burst correction, it is OK if the false zero is far away (cyclically), but if it is close to the burst it can give a false zero that could link to the true zeroes to give a false alternative cyclic burst.

The shortest code word burst is $g(X)$, which is a burst of length $n-k+1$ (11 in this example). $g(X) = 10101011001$ has ones in positions 1,3,5,7,8,11. These are covered except in position 11 by the burst 1,3,4,5,6,7,8,9, so the false zero in position 11 can't be eliminated. If we make any of the interior positions 3,5,7, or 8 error-free, we are 2 away from covering $g(X)$. In the example where the error in position 5 is eliminated, although the zero in position 11 is eliminated, there is another code word one away that creates a false zero in position 12. But no false alternative burst of length < 10 exists, as explained in VI.A.

If we make only interior position 2 error-free, we don't eliminate the false zero in position 11, since we still are one away from covering $g(X)$. But now, there are two bursts of length 9 that are both one away from covering $g(X)$.

Cyclically, $X^{n-2}g(X) \bmod X^n-1$ is another trouble candidate, with 1's in positions 20,1,3,5,6,9. Removing an error in position 8 will leave an alternative burst due to this code word. Removing an error in position 7 only will not leave an alternative burst of length < 10 in this code word. This is because rank 8 requires linking positions 1-6 and 8 to the false zero in position 20, which is a burst of length 10.

If we make position 4 the zero error position, it does not eliminate a false zero, but the false burst would then have to extend past the ELV 1 in position 4 as well as the 1 in position 10 or position 1, and cover 8 zeroes, giving a length of 10. So the ambiguous cases with one error-free interior symbol are making positions 2 or 8 error-free. Other ambiguous cases with different amounts of error-free symbols can be found.

C. Relating false zeroes to code words one away from a burst.

Look at the systematic form of the generator matrix. This form allows easy discovery of all the code words covered one away from a full burst starting in the first position. The j th row, $1 \leq j \leq k$, is

$$c_j(X) = r_j(X) + X^{n-k-1+j}$$

where

$$r_j(X) = X^{n-k-1+j} \bmod g(X)$$

For any j where $\deg\{r_j(X)\} < b$, where b is the full burst length, $c_j(X)$ is a code word one away from coverage by the burst. These are the only code words one away. No code word that is a combination of two or more rows in the systematic G can be only one away, since the combination would have to have at least two 1's in the last k columns, which are all outside the burst.

Each code word one away from the burst contributes exactly one false zero to the ELV.

For non-full bursts, removing an error in position c will eliminate any one-away $c_j(X)$ that has a "1" in column c , row j of G . The error removal cannot increase the number of false zeroes, and usually reduces the number. By the argument of theorem 3, removing an error from the end of a burst reduces all false zero strings by one.

If the same burst is in a different position, the systematic G can be shifted so the non-identity part starts at the burst start, and all the one-from-covered code words and the ELV will shift by the same amount.

As an example, consider the (21,11) code with a burst of length 9 and eight errors, with position 2 error-free, as illustrated in VI.3, Relative to form 2,

$$H = [I_{n-k} \mid P], \quad G = [P^T \mid I_k]$$

ELV position	power		← n-k=10 →		
			0100000001	01111100101	
			code word covered all but one		Row of G
11	10	$g(X)$	1010101100	1000000000	1
17	16	$X^{16+r_7}(X)$	0011110100	00000010000	7
18	17	$X^{17+r_8}(X)$	0001111010	00000001000	8
20	19	$X^{19+r_{10}}(X)$	1010110010	00000000010	10

From the ELV, we saw that two bursts of length 9 with position 2 error-free are possible, covering within one of $g(X)$. The false zero in the fourth code word would need a cyclic burst of 10 to encompass the needed 8 zero locations to get the rank of 8. The second and third false zeroes would need still longer bursts.

D. Conditions for two possible bursts of length $< n-k$ covering within one of a code word

Let's take a closer look at the case where the error burst was $a0cdefghi$, length 9, and alternative length 9 burst $00a'b'c'd'e'f'g'0i'$. Both of these are one from the code word $g(X)$, so we can concentrate on that case.

ELV			0100000001	01111100101	
position	power		code word covered	all but one	Row of G
11	10	$g(X)$	1010101100	1000000000	1

Let's compare the syndrome sets beginning from where these bursts start coming in the window of the feedback shift register.

	True syndromes	False syndromes
10 1000000000 100110000010	a+c 0 a	a' 0, i'
01 0100000000 01001100001	d a+c 0	b' a' 0
10 0010000000 10111110010	a+e d c	c' b' a'+i'
01 0001000000 01011111001	f a+e d	d' c' b'
10 0000100000 10110111110	a+g f e	e' d' c'+i'
11 0000010000 01011011111	a+h a+g f	f' e' d'
01 0000001000 10110101101	i a+h g	g' f' e'+i'
00 0000000100 110000010100	0 i h	0 g' f'+i'
10 0000000010 011000001010	a 0 i	i' 0 g'
01 0000000001 001100000101	0 a 0	0 i' 0
a0 c d e f g h i →		
a'b'c'd'e'f'g'0i' →		

Note that the column where the a lines up shows that a doesn't affect the second, fourth, seventh, eighth or tenth syndrome. These syndromes should be d, f, i, 0, 0, respectively, with no contribution from other errors. Under the second hypothesis, these places would be b', d', g', 0, 0, also with no contribution from other errors. Thus $d = b'$, $f = d'$, and $i = g'$, and we have found 3 of the errors.

Despite finding three of the errors, there still is not a unique solution. The remaining zeroes in the ELV exactly match the one-from-covered $g(X)$. Both have 5 errors but one is burst 8, the other burst 9.

remaining patterns:

```

a 0 c 0 e 0 g h
  a' 0 c' 0 e' f' 0 0 i'
0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1  ELV
1 0 1 0 1 0 1 1 0 0 1    g(X)

```

It is easy to see the reason for this failure to resolve when two different bursts exactly cover one from the same code word. Suppose the code word $c(X)$ has weight m . The m nonzero positions are not necessarily consecutive. Say the correct burst is precisely $[e_1, e_2, \dots, e_{m-1}, 0]$ in the first $m - 1$ positions of $c(X)$. Take any vector symbol v . The pattern $[v, v, \dots, v]$ in these m nonzero positions of $c(X)$, zero elsewhere has a zero syndrome. If $v = e_1$, the pattern $[0, e_1 + e_2, e_1 + e_3, \dots, e_1 + e_{m-1}, e_1]$ has the same syndrome as $[e_1, e_2, \dots, e_{m-1}, 0]$ and they cannot be distinguished. With the $v = e_i$, the zero could appear in any position, but cases where the zero appears elsewhere, or does not appear, correspond to a burst of length at least $n-k+1$. Thus there are only at most two candidates, one with a zero at the beginning, and one with a zero at the end. As a burst, they have the same number of errors, but may be different lengths, due to the irregular spacings of the m code word 1's.

When are two candidates within a one-from-covered code word not possible? With burst correction, if either $[e_1, e_2, \dots, e_{m-1}, 0]$ or $[0, e_1', e_2', \dots, e_{m-1}']$ is of length $> n-k-1$, it is not a possible candidate. In particular, suppose $c(X)$ starts with $1+X$. Then the span including e_1' and e_{m-1}' would stretch from X to at least X^{n-k} , a span of length at least $n-k$, which is not a candidate. A similar statement applies to e_1 and e_{m-1} if $c(X)$ ends with two 1's. If $c(X)$ both starts and ends with two ones, no two bursts of length $< n-k$ can cover one from that $c(X)$. In particular, this applies to $g(X)$, the least length code word burst.

E. Shorter bursts with the (21, 11) code.

ELV, full burst of 8			0000000011 01110001111	
position	power		code word covered all but one	row of G
11	10	$g(X)$	1010101100 1000000000	1
15	14	$X^{14}+r_5(X)$	1111010000 0000100000	5
16	15	$X^{15}+r_6(X)$	0111101000 0000010000	6
17	16	$X^{16}+r_7(X)$	0011110100 0000001000	7

Removal of error in position 2.

New ELV			8	0100000011 01111101111
position	power		9	code word covered all but one
11	10			1010101100 1
17	16			0011110100 0000001

A false burst of length 9 would look the same as a true burst of length 8, both of rank 7, and vice versa. Thus this case is not unique.

ELV, full burst of 7			000000011111110011111	OR of last 3 rows
position	power		code word covered all but one	
15	14	$X^{14}+r_5(X)$	111101000000001000000	
16	15	$X^{15}+r_6(X)$	011110100000000100000	

Removal of any interior error except solely position 5 or solely position 6 would remove both false zeroes, as would removing both 5 and 6. But removing just one of these would reduce the rank to 5, and it would not be possible to link a burst of length less than ten having 5 zero positions including the one remaining false zero. Thus all independent error bursts of length 7 or less are uniquely decodable with this code.

F. Relation to the methods in [15].

In [15], the ability to correct all bursts of length $< n-k$, assuming linearly independent errors is obtained only when $h(X)$ is a vector of all 1's. This limits the class of codes to a relatively small number, all with rate $< 1/2$. By allowing $h(X)$ to have no more than j consecutive interior ones, [15] shows that all bursts of independent errors up to length $n-k-1-j$ could be corrected uniquely. In these cases rates $> 1/2$ can be used. An example of this from [15] is a

(30,19) code capable of correcting all independent bursts of length up to 9, with

$$g(X) = 10111111101$$

$$h(X) = 10110110111101101101$$

burst length	ELV (assuming burst in first part)
10 (full)	0000000000 1011011011110110110
9 (full)	0000000001 11111111111111111111

It is obvious that there are no false zeroes with a burst length of 9, so removing interior errors would not lead to any false burst.

However, what was not shown in [15] is that the same code can also correct a full burst of length 10, as well as most other bursts of length 10. Put the burst in a center place to see the effects on both ends.

		x	
ELV centered		11101101101	0000000000 10110110111
one-from-covered words	1)		1011111111 01
	2)	10	111111101
	3)		1111011111 00001
	4)		1111111011 00000001

Also, 5) and 6) (not shown) are symmetric versions of 3) and 4). $g(X)$ and $h(X)$ are each their own reciprocals in this code.

Removing any two or more internal errors will keep the errors from being from one of any code word, since each code word in 1) to 6) has only a single zero in the burst range. Removing any error except the one in position 2 (see x) or the one in position 9 of the burst will keep the errors from being within one of code words 1) and 2), and remove the nearby false zeroes. Making position 5 error-free to keep within one of word 3) would not make it possible to span a burst of length < 11 including the outside zero position and 8 other zero positions. Similar statements can be mad about 4) to 6). Thus, of all the bursts of length 10 or less, the only ones that would lead to an ambiguity due to 1) and 2) are $x0xxxxxxx$ and $xxxxxxx0x$. In these cases there will be two bursts with the same syndrome.

Thus only 2 of the 256 inner patterns of a burst of independent errors with length 10 could cause an ambiguity. This is a significant extension over the ability to correct all independent error bursts of length up to 9 or less unambiguously.

H. The (23,12) Golay code

This well-known code [16] has $g(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}$ and a minimum distance of 7. It can be shown that the Golay code can correct unambiguously any burst of independent vector symbol errors of length $n-k-1 = 10$ or less, whether full or not. The details are omitted to save space. The arguments are similar to those in the (21,11) code example. In addition, as a vector symbol code, it can correct any random pattern of $7-2 = 5$ independent errors, as well as any other random independent error pattern of > 5 errors that doesn't come within one of covering a code word. The probabilities that a certain number of random errors > 5 can be decoded can be computed since the weight distribution of the Golay code is known. For example, since there are 253 words of minimum nonzero weight 7, there are $7 \cdot 253 = 1771$ patterns of 6 errors that cannot be decoded, which is $1/57$ of all 6-error patterns.

VII. DEPENDENT ERRORS AND ZERO SYNDROME COUNTS IN SHIFTS

The method of just shifting the feedback shift register to find the maximum number of zero syndromes often works even if errors are linearly dependent. When there are β errors, there will be exactly $n-k-\beta$ zero syndrome rows, whether or not the errors are independent. If they are independent, they reveal all $n-k-\beta$ null combinations. As explained in Section V, the maximum number of zero syndrome rows usually occurs where the burst is trapped, even with some dependencies.

Consider the (21,11) code. For $b = 6$, let a, b, c, d, e, f be the error values in a full burst. Below is shown the number of zero syndrome rows for various starting positions in the cases 1) all 6 errors are linearly independent, and 2) $\rho = 5$ with $b+d+e=0$, and 3) $\rho = 4$ with $a = c$ and $b+d+e = 0$. Part of the matrix is reproduced on both sides to show all cases.

```

abcd ef -
10011000010100000000010011000010
01001100001010000000001001100001
10111110010001000000010111110010
01011111001000100000001011111001
10110111110000010000010110111110
01011011111000001000001011011111
10110101101000000100010110101101
11000010100000000010011000010100
01100001010000000001001100001010
00110000101000000000100110000101

```

```

case 1,  $\rho = 6$       000000000124444420000
case 2,  $\rho = 5$       101111100004444430001
case 3,  $\rho = 4$       101111212234444430001

```

Even for these cases of $\rho = 5$ and $\rho = 4$, the decoder sees that the shortest possible burst is length 6, and traps the correct burst of length 6. The method works even without computing any ELV. Obviously it wouldn't work for rank 1, such as binary symbols, since in $g(X)$ two bursts of length 5 add to a code word so the binary burst correcting capability of this code can't be more than 4.

Let's look at the ELV, relative to when the burst is moved just into the last $n-k$ positions:

```

case 1,  $\rho = 6$       11110111111 0000001111
case 2,  $\rho = 5$       11110111111 0101101111
case 3,  $\rho = 4$       11111111111 1111101111

```

For cases 2 and 3 the case 1 ELV is a partial ELV, whose zeroes only can be reduced by other null combinations. Note it already indicates the burst location. Investigation of the found errors can discover their dependencies and the actual ELV, due to the logical OR combination. but this would probably not be necessary.

In [6], it has been shown how to correct most cases of t errors of rank $t-1$ when the error pattern doesn't cover one from a code word. However, no method has been given for when the rank is $t-2$. Here the error pattern actually is one from a code word, and in case 3 is rank $t-2$. So these results go beyond the capability shown in [6].

In the same code, consider a burst of length 6 with 3 errors: a00d0f

	a00d0f →	
	10011000010	1000000000
	01001100001	0100000000
	10111110010	0010000000
	01011111001	0001000000
	10110111110	0000100000
	01011011111	0000010000
	10110101101	0000001000
	11000010100	0000000100
	01100001010	0000000010
	00110000101	0000000001
independent	12222233333	7777733301
a=d, rank 2	23343366333	7777733412
ELV, independent	11111111111	0110101111
a=d additional	11000111011	1001000000
ELV, dependent	11111111111	1111101111

The observations of 7 zero syndromes finds the burst location and values. Again, the correct burst is found simply, even with a dependency. The partial ELV identifies the three errors. The total ELV identifies only the error not in the dependency. In this case the errors don't cover one from a code word, and the rank is $t-1$. Thus in this case the vector symbol decoding method in [6] also would have found the errors and values, but with greater effort.

VIII. CORRECTING BOTH BURSTS AND RANDOM ERRORS

There are many cases where most errors in a block code are clustered in a burst, but there are one or two errors outside the cluster. Trapping the cluster part can simplify decoding of all the errors.

A. A burst plus a single outside error

Suppose there are e errors clustered in a span of $n-k$ positions, and one error outside the span. Assume the $e+1$ errors are linearly independent and do not cover all but one position of any code word. Vector symbol decoding can find the error locations and solve for their values, but decoding can be simplified by using some principals related to the burst correction methods.

Do the usual feedback shift register computations of syndromes in different cyclic

positions. When the cluster of e is trapped, there would be n-k-e zero syndromes, except that the one outside error on average will change about half of the otherwise zero syndromes to the value of the outside error. If it doesn't change any it would mean that the column of the H matrix corresponding to the outside error is the sum of only positions of the other errors, and the errors cover a code word, contrary to the assumption. If it removes only one zero syndrome, that means that the error pattern is only one from covering a code word, which is also contrary to assumption. So the outside value will appear at least 2 times in the set of syndromes. The following illustrates how easily the errors can be found and evaluated.

a	b cd e	Errors syndrome rows	obvious null combinations
10011000010	1000000000	0	10011000010 1000000000
01001100001	0100000000	b	
10111110010	0010000000	a	
01011111001	0001000000	a+c	
10110111110	0000100000	a+d	
01011011111	0000010000	a	
10110101101	0000001000	0	10110101101 0000010000
11000010100	0000000100	a+e	
01100001010	0000000010	0	01100001010 0000000010
00110000101	0000000001	0	00110000101 0000000001
		Partial ELV	11111101111 1000001011
Additional zero syndrome, row 3 + row 6:			11100101101 0010010000
		Total ELV	11111101111 1010011011

The duplicate “a” is the outside error value, and the Partial ELV identifies the outside position. Thus the outside value “a” is revealed. (Or the total ELV can be computed by including row 3+row 6 as a null combination.) Then, add “a” to each syndrome where the outside column is a “1”, and all the burst error values appear directly as the syndrome values in the indicated positions..

In the (21,11) code, $d_{min} = 6$, so any case of 4 independent errors, 3 in a burst and one outside, can be corrected. In this case it worked with 5 errors because the errors happened not to all be in 5 of the one positions of some weight 6 code word.

Following is a case of two errors outside and three inside. In this case the Partial ELV from the two direct zero syndromes is not enough to identify the outside error locations or values. However, two duplicates can be observed readily, creating a further partial ELV, which shows

the two outside error positions and the first three syndromes are free of any errors inside the diagonal part. From the first two rows, we have the submatrix results

10 → a syndrome
 11 → a+b syndrome

We can then subtract the found values a and b from the contributions shown by columns 5 and 6. This reveals directly all the other errors: c, d, and e.

Errors		syndromes	
ab	cd e		
10011000010	1000000000	a	
01001100001	0100000000	a+b	
10111110010	0010000000	a+b	
01011111001	0001000000	a+b+c	
10110111110	0000100000	b+d	
01011011111	0000010000	a	
10110101101	0000001000	b	
11000010100	0000000100	e	
01100001010	0000000010	0	01100001010 0000000010
00110000101	0000000001	0	00110000101 0000000001
		Partial ELV	01110001111 0000000011
Additional zero syndrome, row 2 + row 3:			11110010011 0110000000
Additional zero syndrome, row 1 + row 6			11000011101 1000010000
		Further partial ELV	11110011111 1110010011
Additional zero syndrome, rows 1,2,7			01100001111 1100001000
total ELV			11110011111 1110011011

In general, if any calculation of the ELV splits into a part with 1 or 2 location zeroes and the rest of the zeroes in a span of n-k places, with the rank matching the number of zeroes, then by moving the register to trap the burst part, the solution for the 1 or 2 isolated errors involves just one or two equations, and all the other values are found by subtraction. Calculation of the ELV itself can be simplified by doing it in the shift position that shows the greatest number of obvious null combinations - zero syndromes and duplicate values.

IX. CONCLUSIONS

It has been shown how to expand prior capabilities of burst error correction with vector symbol cyclic codes to correct in a simple manner most events of most burst errors of length $< n-k$ and random error patterns that do not cover all but one position of any code word. By borrowing some simple methods used in burst error correction, a way of correcting random vector symbol errors more simply than in prior work is proposed. If most errors are clustered in a span of $n-k$, with a few outside errors, the outside errors can be discovered simply, and effects subtracted. Then the clustered errors are revealed immediately in a shift register.

A general result has been proven, for the special case of a full burst of independent errors, that all full bursts of length less than $n-k$ can be uniquely decoded by any cyclic code with minimum distance greater than 2. Unfortunately, this property does not extend in general to non-full bursts, where capabilities are code-dependent. The property does extend to some cyclic codes, including the (23,12) Golay code. Methods have been described for finding burst capabilities based on the ELV and code word coverage. It has been shown that burst correction capability often extends to cases where the errors cover all but one position of a code word.

Previous vector symbol decoding for random errors has been restricted to cases where errors are at least two away from coding any code word. Extension with bursts is possible because, if two different patterns with the same number of errors have the same syndrome, decoding is considered as failed for randomly-placed errors, but if one pattern is a burst of length $< n-k$ and the other is not, it is decoded as the burst.

It also has been demonstrated that much of the burst correction capability holds even when there is some dependence. A previous result for general parity check codes with Vector Symbol decoding is that most cases of t errors of rank $t-1$ can be decoded with some extra effort. It is shown here that such cases can be decoded very simply for cyclic codes, extending to many cases of rank even less than $t-1$.

There is considerable room for additional research on the ideas presented here. Although the paper describes some guiding principles, precise recipes for designing good burst-error-correcting codes are not given. Based on general results for randomly-chosen codes for vector

symbol decoding, it is suspected that the choice of code might not be too critical. It was shown in this paper that simply observing zero syndrome rows over a succession of feedback shift register shifts can allow simple decoding with dependent errors that is beyond the capabilities of previous methods of vector symbol decoding. However, general results defining precisely how much dependency can be tolerated have not been found. Simulations with different codes and error events could clarify capabilities.

REFERENCES.

- [1] S. H. Rieger, "Codes for the correction of 'clustered' errors," *IRE Trans. Inform. Theory*, Vol. IT-6, pp. 16-21, 1960.
- [2] R. G. Gallager, *Information Theory and Reliable Communication*, pp. 291-297. New York: Wiley, 1968.
- [3] G. Chen and P. Owsley, "A burst-error algorithm for Reed-Solomon codes," *IEEE Trans. Inform. Theory*, pp. 1807-1812, November 1992.
- [4] J.J. Metzner and E.J. Kapturowski, "a general decoding technique applicable to replicated file disagreement location and concatenated code decoding", *IEEE Trans. Inform. Theory*, Vol. 36, pp. 911-917, July 1990.
- [5] K.T. Oh and J.J. Metzner, "Performance of a general decoding technique over the class of randomly chosen parity check codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 160-166, January 1994
- [6] J. J. Metzner, Vector Symbol Decoding with List Inner Symbol Decisions and Dependent Errors. *IEEE Trans. Commun.*, vol. 51, pp. 371-380, March 2003.
- [7] J. J. Metzner, "An Improved Broadcast Retransmission Protocol," *IEEE Trans. Commun.*, vol. COM-32, pp. 679-683, June 1984
- [8] B. Whetten and G. Taskale, "An Overview of Reliable Multicast Transport Protocol II", *IEEE Network Magazine*, pp. 37-47, Jan/Feb 2000.
- [9] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient Erasure Correcting Codes," *IEEE Trans. Inform. Theory*, v. 47, pp.569-584, February 2001.
- [10] J. W. Byers, M. Luby and M. Mitzenmacher, "A Digital Fountain approach to Asynchronous Reliable Multicast," *IEEE J. Selec. Areas Comm.*, v. 20, pp. 1528-1540, October 2002.
- [11] J. J. Metzner, "Majority-Logic-Like Vector Symbol Decoding with Alternative Symbol Lists," *IEEE Trans. Commun.* v. 48, pp. 2005-2013, December 2000
- [12] M. G. Luby and M. Mitzenmacher, "Verification-based decoding for packet-based low-density parity check codes," *IEEE Trans. Inform. Theory*, vol. 51, pp. 120-127, January 2005.
- [13] J. J. Metzner, "Convolutionally Encoded Memory Protection," *IEEE Trans. Computers*, vol. C-31, pp. 547-551, June 1982.
- [14] U. Tuntoolavest and J. J. Metzner, "Vector Symbol Convolutional Decoding with List Symbol Decisions," *Integrated Computer-Aided Engineering*, vol. 9 no. 2, pp. 101-116. IOS

Press, Netherlands. 2002

[15] J. J. Metzner and Y. T. Cha, "On Simple Correction of Bursts up to Nearly Twice the Guaranteed Correction Bound," *Proceedings of the 1994 Princeton Conference on Information Systems and Sciences*. pp. 38-47, Princeton, NJ, March 1994. See also US Patent 5,657,331, August 1997.

[16] M. J. E. Golay, "Notes on digital coding," *Proc. IRE*, vol. 37, p. 657, June 1949.