W. F. Rogers

A Practical Class of Polynomial Codes

Abstract: Error detecting polynomial codes are usually formed by defining a correspondence between data bits and coefficients of the representative polynomial. These codes are easily implemented in hardware using shift registers; however, implementation in character-oriented processors may be too time consuming. A new class of polynomial codes is described for which a correspondence between n-bit data characters and polynomial coefficients is defined. Two particular types of these "character polynomial codes" are discussed; these may be easily implemented with hardware or with processor character manipulations. The burst error detecting ability of these two types of codes is shown to be the same as for the common "bit polynomial codes."

Introduction

Polynomial codes are attractive for error detection because of their high efficiency and their ease of implementation. These codes are called polynomial codes because they are best described by their representation as polynomials, wherein successive channel symbols are normally represented by successive polynomial coefficients. Polynomial codes are often referred to as cyclic codes; however, by strict definition the codes generally described in this paper are not necessarily cyclic. Formation of the code word from the information word is described elsewhere.[1,2]

Codes for which the polynomial coefficients are taken from the field of two elements will be called "bit polynomial codes," to distinguish them from other codes to be discussed. The ease of implementation for bit polynomial codes occurs when shift registers with simple feedback are used for encoding and decoding.[1,2] If encoding or decoding with the processor of a general purpose digital computer is desired, implementation becomes more difficult. Implementation of bit polynomial codes is a bit-by-bit operation, whereas most computer processors operate on a character or word basis. Programming procedures such as shift register simulation or table look-up may be used; however, these methods might require too much time or storage.

A practical solution to the problem of the complexity of processor implementation of general bit polynomial codes is found by defining a new class of error detecting polynomial codes, which will be called "character polynomial codes." Hardware implementation of these codes is discussed, because the capability to implement them with inexpensive hardware is desirable in cases for which character addition and manipulation capability does not exist—such as at remote terminals. Character polynomial codes are described as a general class, followed by a discussion of two practical types of character polynomial codes, with a note about their implementations. The performance of the two codes is compared for burst error detection; finally, the detection of independent errors is considered by investigating the special case of polynomial generators of the form $X^r + 1$.

Character polynomial codes

In order to derive codes which can be easily encoded and decoded on a character basis, one may consider representing data and code words by polynomials whose coefficients represent n-bit characters rather than bits. Polynomial codes which have this correspondence between coefficients and characters are thus called "character polynomial codes." Code words are formed from data words, as with bit polynomial codes, by computing the remainder polynomial, except that in this case the polynomial coefficients are taken from an R-group of order 2^n rather than 2. An R-group is defined as an abelian additive group which has a multiplicative identity, 1. Multiplication in the R-group is defined only between 0 or 1 and some element of the R-group. This definition is motivated by the fact that multiplication by other than the elements 0 and 1 would greatly increase the complexities of hardware and processor implementation. To avoid this complexity, generator polynomials are restricted

to have only 0 or 1 for coefficients. With this restriction, multiplication by other than 0 and 1 is not required when computing the remainder $R(X)$.

Using these character polynomial generators, calculation of the remainder polynomial can be reduced to implementation of a polynomial division algorithm. The procedure consists of subtraction of the coefficient (from an R-group) of the highest degree term of the dividend from certain succeeding coefficients or characters determined by the generator (divisor). Next, the highest order coefficient of this difference is subtracted from certain succeeding coefficients; this process is continued until the remainder polynomial results. Implementation of this algorithm can be accomplished in a processor by executing a character manipulation routine as each character is received. The number of character subtractions required per code character should be less than the number of nonzero coefficients of the generator.

For bit polynomial codes the polynomial operations are unique, since there is only one R-group for two elements. However, for character polynomial codes the coefficients may be in any of several R-groups, each of order 2^n. Two R-groups, for which the polynomial division algorithm is easily implemented in the processor, involve the computer operations of componentwise addition modulo two and binary addition. The character polynomial codes using these R-groups will be referred to as Type 1 and Type 2 codes, respectively.

The other logical operations of inclusive OR and AND do not form R-groups, because some of the elements of the set do not have unique additive inverses. Error detection can still be accomplished with these addition operations if modifications are made to the procedure. However, these operations will not be considered, because the redundant bits obtained in separable codes do not provide as great a code word separation as with Type 1 or Type 2 codes.

Cyclic codes for which the polynomial coefficients are taken from a finite extension field over the Galois field of two elements $GF(2)$ have been discussed.[3] By comparison, coefficients for character polynomial codes are only restricted to be elements of an R-group rather than a field. For the Type 1 character polynomial codes the coefficients do belong to an extension field over $GF(2)$, and are discussed elsewhere[3] in connection with multiphase data transmission.

Type 1 codes can be shown to be equivalent to a class of bit polynomial codes whose generator polynomials have the form $G(X) = P(X^n)$. The characteristics and detecting ability of these codes are well known. Implementation of Type 1 codes in a processor requires only the successive use of the exclusive OR operation of characters; no shift registers are necessary. Hardware implementation is accomplished with a shift register.[1,2]

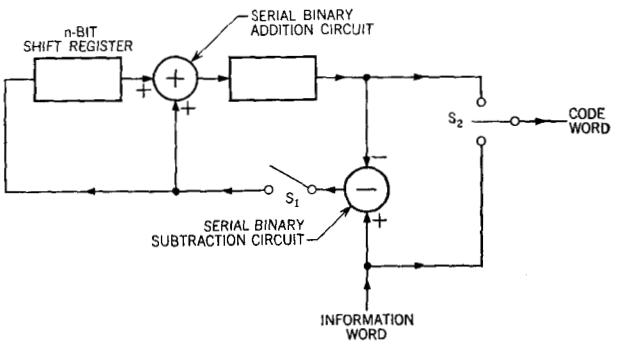

Figure 1 Circuit for Type 2 character polynomial encoding: $G(X) = X^2 + X + 1$.

Codes using binary addition

For the second code under consideration, the addition operation for the elements of the R-group from which the polynomial coefficients are taken is defined as binary addition of characters. The high order carry in the addition is neglected so that the set will be closed under this operation. That is, the 2^n elements are isomorphic to the integers modulo 2^n.

Implementation of Type 2 codes requires the distinction between addition and subtraction of elements, whereas, with the exclusive OR addition operation of Type 1 codes, addition and its inverse are the same. Subtraction is accomplished in this R-group by adding a character (minuend) to a complemented character (subtrahend) and adding a 1 in the low order bit position; any high order carry is neglected. When a processor is used for encoding and/or decoding, the polynomial division simply requires successive use of the binary subtraction instruction.

Implementation with hardware, although not as straightforward as with bit polynomial codes, is practical using shift registers and a small amount of additional circuitry. The bits are assumed to enter the encoder-decoder serially; it is therefore necessary to perform the Type 2 (binary) addition and subtraction operations serially.

One method of implementation is illustrated with the encoding circuit of Fig. 1. By propagating the carry, the serial binary adder performs addition (or subtraction) of characters whose bits enter the unit serially. A character synchronization signal is used to inhibit the high order bit carry of each character operation and add a 1 to the low order bit position for subtraction. The Type 2 addition or subtraction circuit need only consist of a flip-flop and a few gates. S_1 and S_2 are used to form the code word as in bit polynomial coding.

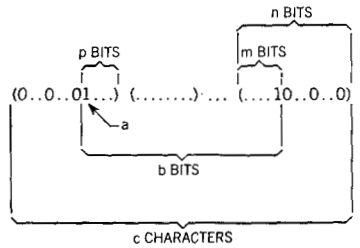

Figure 2 Representation of an error burst.

Error detecting capability

Due to the complex nature of error probabilities on most channels, the best way to determine the ability of a given code to detect errors is to run actual tests over the channel. However, for comparison purposes, the burst error detecting ability is derived for Type 2 codes; also, the fraction of double errors undetected by codes with the generator $G(X) = X^r + 1$ is determined.

An error burst of length b is here defined as b consecutive bits with the first and last bit in error, and any pattern of errors in between. A word in error will be assumed to contain one error burst, and all bursts of length b are assumed equally likely. Peterson[1] shows that all bit polynomial codes, whose generators are of the same degree, detect the same fraction of bursts of each length. If k is the degree of $G(X)$, the fraction of bursts of length b that are undetected is $2^{-(k-1)}$ if $b = k + 1$, 2^{-k} if $b > k + 1$, and none if $b \leq k$. Peterson's proof holds for Type 1 character polynomial codes, since they are a class of bit polynomial codes. That Type 2 codes have the same burst detecting ability as all bit polynomial codes of the same redundancy will now be proved:

Figure 2 shows the representation of an error burst, where the 1's represent bit positions in error and 0's represent bit positions not in error. The variables in this representation are related by Eq. (1):

$$c = \frac{b - p - m + 2n}{n}. \tag{1}$$

To analyze the detecting ability of this code, let $X^iE(X)$ denote the polynomial which, when added (Type 2 addition) to the code polynomial, gives the representative polynomial of the code word with undetectable errors. An error is undetectable if and only if $E(X)$ is a multiple of $G(X)$; $Q(X)$ is defined by $E(X) = G(X)Q(X)$, where X does not divide $G(X)$ or $E(X)$. $E(X)$ is of degree $c - 1$, $G(X)$ is of degree r, and, therefore, $Q(X)$ is of degree $c - r - 1$. Note that the polynomial coefficients are from an R-group and represent characters.

The fraction of errors of length b which are undetected will be determined by examining the number of choices of $Q(X)$ which when multiplied by $G(X)$ give $E(X)$ that represent b-bit error patterns. For a given error pattern, $E(X)$ is a function of the transmitted code word for Type 2 codes. However, the ratio of undetected to total errors is the same for each code word. The proof is divided into three cases for ease of comparison with bit polynomial codes; the cases are defined by b being less than, equal to, or greater than $nr + 1$.

For $b < nr + 1$ and $c < r + 1$, $E(X)$ is of lower degree than $G(X)$ and, therefore, is not divisible by $G(X)$; consequently, all error bursts for which $c < r + 1$ are detected. For $b < nr + 1$ and $c = r + 1$, $Q(X)$ is of zero degree and, therefore, there are 2^n possible error polynomials which are divisible by $G(X)$ corresponding to the 2^n possible values of $Q(X)$. Also, by the assumption that $G(X)$ has only 1 or 0 for coefficients, $E(X)$ has the same element, $Q(X) = q_0$, for all of its nonzero coefficients. Using Type 2 addition, the lowest order bit in error in each character corresponds to the lowest order nonzero bit of the element q_0. If $b < nr + 1$ and $c = r + 1$, then $m + p \leq n$, and there is no q_0 which when added to the first and last characters involved in the burst can produce a pattern of errors with $m + p \leq n$. Therefore, all errors are detected for which $b < nr + 1$, $c = r + 1$, and since there are no undetected errors for $c < r + 1$, all errors with $b < nr + 1$ are detected.

For the case of $b = nr + 1$, the number of characters involved in the error burst must be $r + 1$. As stated above, there are 2^n possible values of $Q(X)$, or 2^n possible undetected error patterns for each code word. From Fig. 2, $m + p = n + 1$, and the lowest order 1 of q_0 for an undetected error is in the $(n - m + 1)$'st or p'th position. Thus, for an undetected error, q_0 can be represented only as the number $+2^{p-1}$ or -2^{p-1}, depending on whether a 0 or 1 was transmitted in the position corresponding to "a" in Fig. 2. So, for each transmitted word there is one undetectable error pattern which represents an error burst of length $b = m + p + n(r - 1) = nr + 1$. Since the total number of possible error patterns of length b is 2^{b-2}, the fraction of bursts of length $nr + 1$ which are undetected is $1/2^{b-2} = 2^{-(nr-1)}$.

Finally, for the case of $b > nr + 1$, it is convenient to discuss the subcases of $c = r + 1$ and $c > r + 1$ separately. If $c = r + 1$ and $b > nr + 1$, then $m + p > n + 1$, and q_0 is restricted to have the $(n - m + 1)$'th bit position as the lowest order nonzero bit position. Of these values of q_0 there are $2^{m+p-n-2}$ values which, when added to a given transmitted character, result in a change to the p'th (but no higher) bit position of the character. This means that for each code word there are $2^{m+p-n-2} = 2^{b-nr-2}$ undetected error patterns of length b, for $c = r + 1$. Combining this information with the fact that there are 2^{b-2} different error patterns of length b, we observe that the average fraction of errors of length b

which are undetected is $2^{b-nr-2}/2^{b-2} = 2^{-nr}$. For $b > nr + 1$ and $c > r + 1$, there are $2^{p-1}2^{m-1}(2^n)^{c-r-2} = 2^{b-2-nr}$ choices for $Q(X)$; this gives the number of undetected error bursts of length b. Since the total possible error patterns is equal to 2^{b-2}, the fraction of errors that are undetected for $c > r + 1$ is $2^{b-2-nr}/2^{b-2} = 2^{-nr}$. Therefore, 2^{-nr} is the fraction of errors undetected for all $b > nr + 1$ and $c \geq r + 1$, regardless of the bit position in which the error burst started.

To summarize, it has been proved that the fraction of errors that are undetected by a Type 2 character polynomial code is 2^{-nr} if $b > nr + 1$, 2^{-nr+1} if $b = nr + 1$, and zero if $b \leq nr$. For the same number of redundant bits this burst error detecting ability is the same as that possessed by all bit polynomial codes.

To determine analytically the independent error detecting ability of a code, it is necessary to know the fractions of single, double, triple, etc. errors which are undetected by the code. All single errors are detected by Type 1 and Type 2 codes with generators having more than one term (one-term generators are trivial). Double errors will predominate over higher numbers of errors, and thus an indication of the independent error detecting ability is the fraction of double errors which are undetected.

Consider as an example the class of character polynomial codes whose generators are of the form $X^r + 1$. For Type 1 codes, the fraction of double errors that are undetected is $(1/nr)[(L - r)/(L - 1/n)]$, which approaches $1/nr$ as the word length increases; n is the character length and L is the number of characters per code word. This is because the Type 1 codes with $G(X) = X^r + 1$ are equivalent to the bit polynomial codes with $G(X) = X^{nr} + 1$. For Type 2 codes, let $G(X) = X^r + 1$ and $E(X) = e_sX^s + e_0$, where e_s and e_0 each cause single errors when added to the respective transmitted characters. If $G(X)$ divides $E(X)$, then r must divide s; furthermore, if s is an even multiple of r, $e_s = -e_0$, and if s is an odd multiple of r, $e_s = e_0$. Therefore, each error occurs in the same bit position of each character.

To cause a single bit error, e_s and e_0 can be represented by $+2^i$ or -2^i, depending upon whether the bit in error was transmitted as a 0 or a 1. Thus, if the errors occur in any except the high order bit position of a character, and s is a multiple of r, then an average of one-half of the errors are detected. If the errors are in the highest order bit position and s is a multiple of r, no errors are detected, since $+2^{n-1} = -2^{n-1}$. The average fraction of errors that are undetected is, therefore,

$$\left[(1/2)\left(\frac{n-1}{n}\right) + (1)\left(\frac{1}{n}\right)\right]\left(\frac{1}{nr}\right)\left(\frac{L-r}{L-1/n}\right) = \left(\frac{n+1}{n}\right)\left(\frac{1}{2nr}\right)\left(\frac{L-r}{L-1/n}\right),$$

which is slightly more than one-half of the average number of errors undetected by Type 1 codes with the same generator.

Summary

Codes have been described which may be easily encoded and decoded with simple shift registers as well as with common character manipulations in a processor. Type 1 character polynomial codes are a special class of bit polynomial codes with resulting simple hardware for implementation; Type 2 codes are implemented with very little (if any) increase in hardware over bit polynomial codes of the same redundancy. Type 1 and Type 2 character polynomial codes and all bit polynomial codes have the same burst error detecting ability. Thus Type 1 and Type 2 codes have the advantages of good burst detecting ability and the ease of hardware implementation desirable for remote terminals, while providing the additional feature of simple computer implementation. These character polynomial codes are thought to be particularly advantageous for data communications, storage, and other systems subject to burst type errors which have processors as part of that system.

Acknowledgments

The author wishes to express his appreciation to Mr. Carl O. Pingry of the Office Products Division, IBM, who suggested the problem which led to this investigation and who offered suggestions for possible solutions. Thanks are due also to Dr. Raymond J. Distler of the Electrical Engineering Department of the University of Kentucky for his instruction in coding theory and advice on specific questions related to cyclic codes.

References

1. W. W. Peterson, *Error Correcting Codes*, The MIT Press, Cambridge, Mass., 1961.
2. W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection," *Proc. IRE* 49, 228, 235 (1961).
3. M. Hanan and F. P. Palermo, "On Cyclic Codes for Multi-Phase Data Transmission Systems," *Journ. Soc. Industrial and Applied Math.* 12, 794 (1964).

Received July 26, 1965.
Revised manuscript received December 15, 1965.