D. T. Tang
V. Y. Lum

Error Control for Terminals with Human Operators*

Abstract: The man-machine interface at any terminal in a computer system is a likely source of error and can be regarded as a noisy channel. Certain data, such as ID numbers, can be precoded to protect against most-likely errors, including transposition of adjacent symbols and substitutions, as well as deletions and insertions. This paper first considers certain basic requirements for error detection with minimum redundancy. An efficient special coding scheme designed for decimal terminals is described next. Finally, certain cyclic codes are shown to be adaptable to transposition error control when appropriate decoding schemes are implemented.

1. Introduction

Man-machine communication plays an important role in today's large-scale time-sharing computer systems. When data are entered at a remote keyboard or dial-up terminal, the man-machine interface becomes a likely source of error. Since any error control coding[1] introduced at the terminal clearly cannot protect against errors made by the operator in keying or dialing, certain data, such as ID numbers, must be precoded to guard against errors introduced in this "noisy channel"—the man-machine interface.

A coding scheme is most effective when it protects the data transmission against the most-likely errors resulting from the noise. When a man-machine interface exists in a transmission system, the most common types of errors are transpositions of adjacent symbols, substitutions and occasional insertions and deletions of symbols.

The first part of this paper deals with the detection capability of coding schemes with minimum redundancy, i.e., with one or two check symbols. A special coding scheme designed for decimal terminals is then described. The second part of this paper is concerned with cyclic coding schemes adapted for transposition error control.

2. Coding schemes with permutation mappings

Consider the problem of detecting single-transposition errors in addition to single-substitution errors in a variable length message. We shall derive conditions under which the desired detection capability can be achieved with only one check symbol.

* Some of the results included in this paper were presented at the Fifth Annual Allerton Conference on Circuit and System Theory, Monticello, Illinois, October, 1967, and at the Third Hawaii International Conference on System Sciences, January, 1970.

D. T. Tang is at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and V. Y. Lum is at the IBM Research Laboratory in San Jose, California.

Let us consider a check equation in the general form

$$f_1(a_1) + f_2(a_2) + \cdots + f_k(a_k) + f_{k+1}(c) \equiv 0 \bmod q, \quad (1)$$

where $f_i(x)$ are functions mapping the set of code symbols to itself; $a_i \in A = \{0, 1, \cdots, q-1\}$ is the information symbol in the ith digit; c the check symbol; and q the radix of the system (number of code symbols). Hereafter, in this section, a congruence sign, when used alone, will be understood to imply a modulo q operation.

If a single check equation is to detect all single errors, $f_i(a_j)$ must not be the same as $f_i(a_k)$ for any $a_j \neq a_k$, where $a_j, a_k \in A$. Hence, $f_i(a)$ must be a permutation function. Equation (1) can now be written as

$$p_1(a_1) + p_2(a_2) + \cdots + p_k(a_k) + p_{k+1}(c) \equiv 0, \quad (2)$$

where the p_i are permutation functions.

So far, no detection of transposition errors has been considered. To detect transposition errors, $p_i(a)$ must not be the same as $p_{i+1}(a)$. This is a necessary condition but not a sufficient one. Before we go on to derive the necessary and sufficient condition of single-transposition error detection, we need the following definition.

Definition: A permutation $p(a) = \bar{p}(a)$ is a "perfect-difference permutation" (PDP) if $\bar{p}(a_j) - a_j \not\equiv \bar{p}(a_k) - a_k$ for any $a_j \neq a_k$, where $a_j, a_k \in A$.

It is easy to see that if $\bar{p}(a)$ is a PDP, so are $[\bar{p}(a)]^{-1}$ and $\bar{p}(a) + m$, where $[\bar{p}(a)]^{-1}$ is the inverse mapping of $\bar{p}(a)$, and $m \in A$ is a constant.

Permutations and PDP's have corresponding graphs[2] with interesting properties. Let each of the symbols in A represent a node. Then a directed edge can be drawn from a_i to a_j if $p(a_i) = a_j$. Since p is a permutation, each node will be incident to two edges, one directed

a	0	1	2	3	4
$\bar{p}(a)$	0	2	4	1	3

a	0	1	2	3	4
$[\bar{p}(a)]^{-1}$	0	3	1	4	2

a	0	1	2	3	4
$\bar{p}(a)+2$	2	4	1	3	0

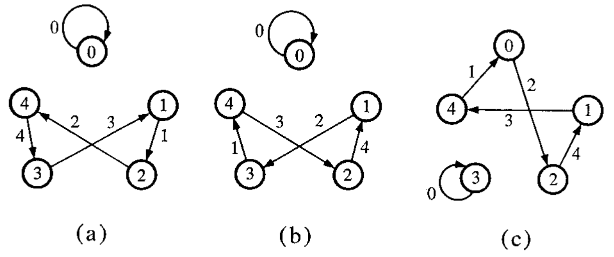


Figure 1 Some PDP's and their corresponding graphs for $q = 5$.

toward and one directed away from the node. An immediate result is that this graph, to be called G_p, the permutation graph of $p(a)$, is a disjoint union of cycles. Let us denote these cycles by $C_1, C_2, \cdots, C_s$, and define the distance of an edge b_k from a_i to a_j to be $d_{ij} \equiv a_j - a_i$. Alternatively, the distance of an edge b_k can be written as $d(b_k)$. Clearly, in a PDP graph, distinct edges must have distinct distances. When the distances of all the edges in a cycle are summed, each node number a_i on the cycle will appear twice in the summation with opposite signs, thus cancelling each other. That is,

$$\sum_{b_j \in C_i} d(b_j) \equiv 0 \tag{3}$$

for each C_i, and thus

$$\sum_{b_j \in G_p} d(b_j) = \sum_{C_i} \sum_{b_j \in C_i} d(b_j) \equiv 0 \tag{4}$$

for every permutation graph G_p.

Figure 1 shows some PDP's and their corresponding graphs.

The following theorem states the capability of a code using PDP mappings.

Theorem 1: A code with check equation as in Eq. (2) will detect single-transposition errors if and only if the mappings between permutation on adjacent positions, i.e., $p_{i+1}[p_i(a)]^{-1}$, $i = 1, 2, \cdots, k$, are all PDP's.

Proof: Let two adjacent symbols a_i, a_{i+1} be interchanged. To detect this error, the check equation must not be zero after the interchange; that is,

$$p_i(a_i) + p_{i+1}(a_{i+1}) \not\equiv p_i(a_{i+1}) + p_{i+1}(a_i). \tag{5}$$

Substituting $b_j = p_i(a_j)$, we obtain

$$b_i + p_{i+1}[p_i(b_{i+1})]^{-1} \not\equiv b_{i+1} + p_{i+1}[p_i(b_i)]^{-1}. \tag{6}$$

By definition, $p_{i+1}[p_i(b)]^{-1}$ is a PDP.

Conversely, if $p_{i+1}[p_i(a)]^{-1}$ for $i = 1, 2, \cdots, k$ are PDP's, then Eqs. (6) and (5) follow, and, therefore, all single-transposition errors will be detected.

One may think that it would be generally difficult to find enough permutations that satisfy the conditions of Theorem 1. However, simple observation reveals that the existence of a single PDP, $\bar{p}(a)$, is sufficient. To show this, let $p_i(a_i) = a_i$, for all odd i, and let $p_i(a_i) = \bar{p}(a_i)$ for all even i. The check equation becomes

$$a_1 + \bar{p}(a_2) + a_3 + \cdots + a_k + \bar{p}(c_1) \equiv 0 \text{ for } k \text{ odd}$$

$$a_1 + \bar{p}(a_2) + a_3 + \cdots + \bar{p}(a_k) + c_1 \equiv 0 \text{ for } k \text{ even}. \tag{7}$$

The existence of PDP's is considered in the following theorems.

Theorem 2: No PDP exists for q even.

Proof: We have noted earlier that in a PDP graph, distinct edges must have distinct distances. Since there are exactly q edges, we must have

$$\sum_{b_j \in G_p} d(b_j)$$

$$= 0 + 1 + \cdots + (q-1) = \frac{q(q-1)}{2} \equiv 0 \tag{8}$$

from Eq. (4), which requires $[q(q-1)]/2$ be divisible by q. It is easy to see that, if q is even, and thus $q - 1$ is odd, this condition cannot be satisfied.

Theorem 3: PDP exists for q odd.

Proof: Although Eq. (8) is satisfied for q odd, it is only a necessary but not a sufficient condition. To show that PDP exists for q odd, we shall rely on a constructive proof.

Consider the permutation graph of $p(i) = q - i$ for $i \in A$. The edge directed from i to $q - i$ has a distance $q - i - i \equiv -2i$.

For two edges to have the same distance,

$$-2i + 2j \equiv 2(j - i) \equiv 0. \tag{9}$$

Since 2 is relatively prime with the odd q, Eq. (9) is satisfied if and only if $i = j$. Thus, all distances are distinct and the permutation so defined is a PDP.

Let us now define a linear permutation p to be one which satisfies $p(a) \equiv ma$, where $a, m \in A$. Linear permutations are easy to implement since only one simple arithmetic operation is needed.

Theorem 4: If $p(a) \equiv ma$ is a PDP, then $p'(a) \equiv (m-1)a$ is a permutation function. Conversely, if $p(a) \equiv ma$ and $p'(a) \equiv (m-1)a$ are permutations, then p is a PDP.

Proof: If $p(a)$ is a PDP, then

$$p(a_i) - a_i \not\equiv p(a_j) - a_j, \text{ for } i \neq j. \tag{10}$$

For $p(a) \equiv ma$, Eq. (10) becomes

$$ma_i - a_i \not\equiv ma_j - a_j \tag{11}$$

or

$$(m-1)a_i \not\equiv (m-1)a_j. \tag{12}$$

It follows that $p'(a) \equiv (m-1)a$ is a permutation.

Conversely, if $p'(a) \equiv (m-1)a$ is a permutation, then Eq. (12) holds for $a_i \neq a_j$. Equations (11) and (10) then follow and, therefore, $p(a) \equiv ma$ is a PDP.

It follows from Theorem 4 that the function $f(a) \equiv ma$ is a PDP if and only if q, m and $(m-1)$ are pairwise relatively prime. Thus, if $q \geq 3$ and odd, and if $m = 2$, $(m-1) = 1$, then $p(a) \equiv 2a$ is a PDP. Similarly, $p(a) \equiv (q-1)a$ is another PDP which is described in the constructive proof of Theorem 3. Thus, for $q > 3$ and odd, there are at least two distinct PDP's.

For q an odd prime, all linear permutations except the identity are PDP's. The converse of this statement is not true. Figure 2 illustrates a nonlinear PDP for $q = 7$.

For q even, Theorem 2 states that no PDP exists. However, one may still be interested in finding a scheme that will detect as many single-transposition errors as possible. Indeed, for q even, there exists a check scheme in the form of Eq. (7) such that all single-transposition errors are detected except when the transposition is between a specific pair a_1 and a_2. This means that there exists a permutation $p(a)$ such that the condition $p(a_i) - a_i \not\equiv p(a_j) - a_j$ is satisfied for all $a_i, a_j \in A$ except for a specific pair a_1 and a_2. To show this, we see that first of all, for $q = 2$, the only confusion pair is (0, 1). For $q > 2$ and even, we consider the following two cases.

If $q = 4m$, let the permutation p be given by

$$p(a) = \begin{cases} a, & \text{for } a = 0, \dfrac{q}{4} \\ q - a, & \text{for } 0 < a < \dfrac{q}{4} \text{ and } \dfrac{3q}{4} < a < q \\ q + 1 - a, & \text{for } \dfrac{q}{4} < a \leq \dfrac{3q}{4}. \end{cases} \tag{13}$$

If $q = 4m + 2$, let p be given by

$$p(a) = \begin{cases} a, & \text{for } a = 0, \dfrac{q+2}{4} \\ q - a, & \text{for } 0 < a < \dfrac{q+2}{4} \\ & \text{and } \dfrac{3q-2}{4} < a < q \\ q + 1 - a, & \text{for } \dfrac{q+2}{4} < a \leq \dfrac{2q-2}{4}. \end{cases} \tag{14}$$

The graphs of these permutations are illustrated for the cases $q = 8$ and $q = 10$ in Fig. 3. It is clear from the graphs that the edges all have distinct distances except for the pair 0 and $q/4$ if $q = 4m$, or 0 and $(q+2)/4$ if $q = 4m + 2$. These pairs constitute the only undetectable single-transposition errors.

It should be pointed out that since often there exist more than one such permutation, one can choose a permutation in which the undetectable transposition error corresponds to the pair of symbols least likely to be confused. In the case of $q = 10$, the only undetectable transposition may be made to occur to any prescribed pair of symbols. Note that the permutations here are always nonlinear.

3. Detection of deletion and insertion errors

Suppose that we wish to detect single-deletion and single-insertion errors in addition to the single-substitution and single-transposition errors with two check digits c_1 and c_2, where c_1 is defined by Eq. (7) and c_2 is to be added at the end of the code word after c_1. If c_2 is deleted, then c_1 will be taken as c_2 and a_k as c_1. The first check will be satisfied if and only if $c_1 = 0$ for k even or $\bar{p}(c_1) = 0$ for k odd. On the other hand, if a digit is inserted after c_2, then c_2 will be taken as c_1. Here again, the first check will be satisfied if $c_2 = 0$ for k odd, or $\bar{p}(c_2) = 0$ for k even.

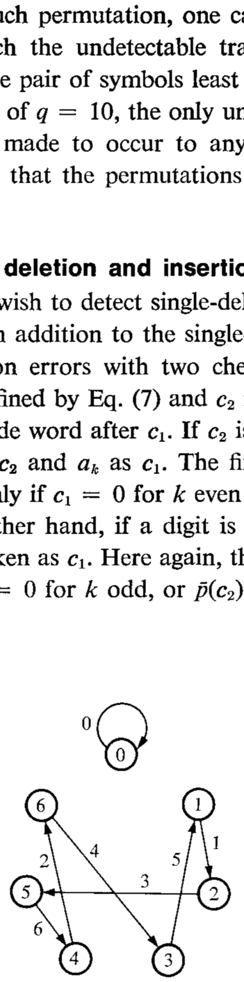
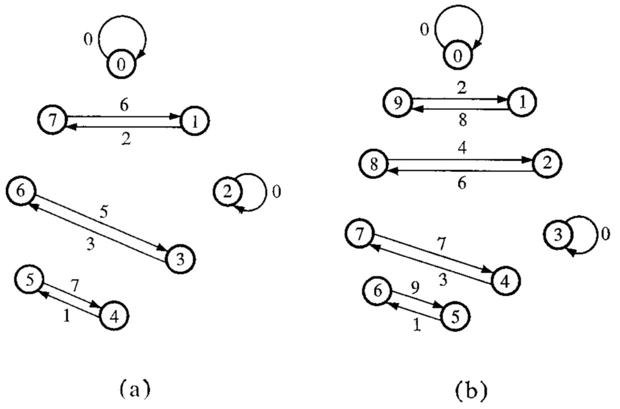
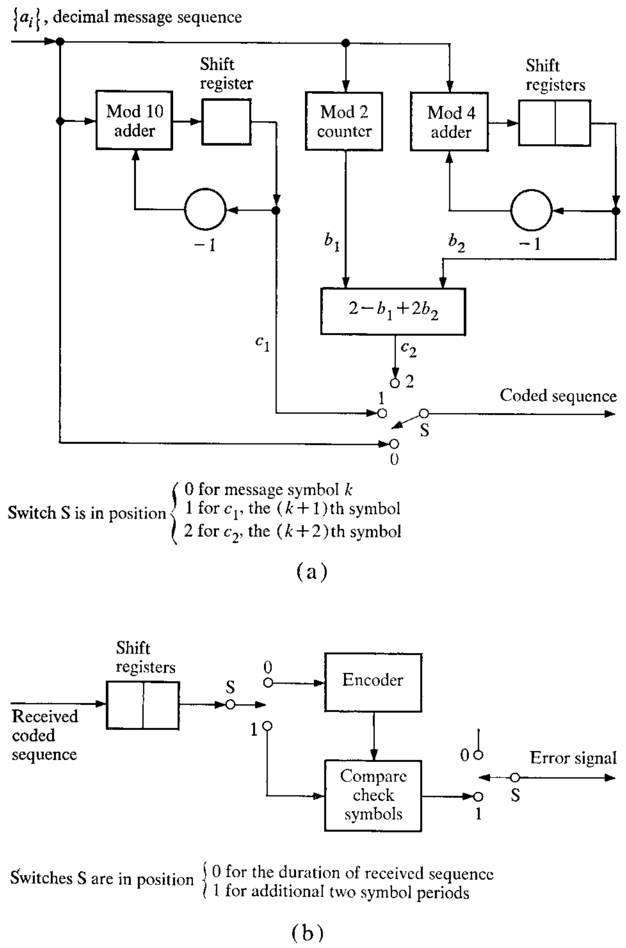


Figure 2 A nonlinear PDP for $q = 7$.

Figure 3 Scheme for detection of most single-transposition errors when q is even. (a) $q = 8$; (b) $q = 10$.





$\{a_i\}$, decimal message sequence
Shift register
Shift registers
Mod 10 adder
Mod 2 counter
Mod 4 adder
-1
b_1
b_2
-1
$2-b_1+2b_2$
c_1
c_2
Coded sequence
S
Switch S is in position { 0 for message symbol k ; 1 for c_1, the $(k+1)$th symbol ; 2 for c_2, the $(k+2)$th symbol
(a)
Shift registers
Received coded sequence
Encoder
Compare check symbols
Error signal
S
Switches S are in position { 0 for the duration of received sequence ; 1 for additional two symbol periods
(b)

Figure 4 Scheme for detecting single-substitution, single-transposition, single-insertion or single-deletion errors in a decimal message. (a) The encoder; (b) the decoder.

In light of the above observations, let us choose the PDP in Eq. (7) to be one such that $\bar{p}(0) = 0$. Let c_2 be nonzero and take different values for k even and odd. Since single substitutions, single transpositions, single deletions not involving c_2, or single insertions before c_2 will clearly be detectable, the only cases left consist of a deletion of c_2 or an insertion after c_2. The latter case is detectable since $c_2 \neq 0$ and $\bar{p}(c_2) \neq 0$, implying that the first check must fail. In the former case, where c_2 is deleted, the first check will be satisfied only if $c_1 = 0$. Therefore, either this check will fail, thus indicating an error, or we must have $c_1 = \bar{p}(c_1) = 0$. But c_1 is taken as c_2 and both c_2 and $\bar{p}(c_2)$ are not zero. The error is, therefore, also detectable.

We observe that, for single-deletion and single-insertion detection as described above, c_2 must take different nonzero values for even and odd k. This implies that $q \geq 3$. For $q > 3$, special schemes can be used to obtain additional detection capability. This is demonstrated for the decimal case in the next section.

The detection of deletion and insertion errors by an additional symbol as described above appears to be very practical. Furthermore, such a check can always be used in conjunction with any other nonbinary checking scheme in which the last check equation takes the general form of Eq. (1).

4. A coding scheme for decimal terminals

We now describe a special coding scheme for decimal message sequences of arbitrary length. With two check symbols the scheme is capable of detecting either a single-substitution, a single-transposition, a single-deletion or a single-insertion error.

Let the message sequence $a_1, a_2, \cdots, a_k$ be followed by two check symbols c_1 and c_2. The first check c_1 satisfies the following equation,

$$(-1)^{k-1}a_1 + (-1)^{k-2}a_2 + \cdots - a_{k-1} + a_k - c_1 \equiv 0 \mod 10. \qquad (15)$$

To obtain c_2, we first determine b_1 and b_2 as follows:

$$b_1 \equiv k \mod 2, \text{ and} \qquad (16)$$

$$\cdots + a_{k-4} - a_{k-2} + a_k - b_2 \equiv 0 \mod 4, \qquad (17)$$

where $b_1 = 0$ or 1 and $b_2 = 0, 1, 2$ or 3. The second check c_2 is determined by the equation

$$c_2 = 2 - b_1 + 2b_2. \qquad (18)$$

Note that c_2 never equals zero.

A sequential encoder and a sequential decoder of the above scheme are shown in Figs. 4(a) and 4(b), respectively. At the receiving end, the last two received symbols are taken as c_1' and c_2', and are compared with check symbols c_1'' and c_2'' regenerated with the received message sequence according to Eqs. (15)-(18). Detection of errors as claimed works as follows:

(*A*) *Single substitutions*

(1) A substitution in the first $k + 1$ symbols implies Eq. (15) fails.

(2) A substitution at the $(k + 2)$th symbol implies Eq. (18) fails.

(*B*) *Single transpositions*

(1) A transposition between a_i and a_{i+1} such that $a_i - a_{i+1} \not\equiv 5 \mod 10$ implies Eq. (15) fails.

(2) A transposition between a_i and a_{i+1} such that $a_i - a_{i+1} \equiv 5 \mod 10$ implies Eqs. (17) and (18) fail.

(3) A transposition between a_k and c_1 implies Eq. (17) fails.

(4) A transposition between c_1 and c_2 implies Eqs. (17) and (18) fail.

(*C*) *Single deletions*

(1) A deletion in the first $k + 1$ symbols implies Eqs. (16) and (18) fail.

(2) The deletion of the $(k+2)$th symbol implies either: a_k (taken as c_1') fails to satisfy Eq. (15); or a_k satisfies Eq. (15) implying $c_1 = 0$, which in turn implies c_1 (taken as c_2') fails to satisfy Eq. (18).

(*D*) *Single insertions*

(1) An insertion anywhere before the $(k+2)$th symbol implies Eqs. (16) and (18) fail.

(2) An insertion after the $(k+2)$th symbol implies $c_2 \neq 0$ (taken as c_1'), and therefore Eq. (15) fails.

It can be shown that, in addition to single errors of the above four types, more than 95 percent of all double-substitution and double-transposition errors are also detected.

5. Polynomial codes

In this section, several types of polynomial codes[1] are examined for their effectiveness in transposition-error control. We assume that the symbols of transmitted and received sequences are identified as elements of a finite field, GF(q). A code generated by the polynomial $g(x)$ of degree r consists of polynomials of degree no more than $n-1$ that are multiples of $g(x)$. The code length n is the smallest integer such that $g(x)$ divides $x^n - 1$. Note that we no longer deal with messages of arbitrary length as in the previous sections.

A transposition can always be regarded as a burst error of length two. However, the property that the values of the error in these adjacent positions must be equal but with the opposite signs enables one to obtain, in nonbinary cases, more efficient coding schemes than those obtained by considering that transpositions are bursts of length two.

• *Correction of all single-transposition errors*

Consider a full-length single-error correcting code generated by an irreducible polynomial $g(x)$ which has $\beta = \alpha^{q-1}$ as one of its roots, α being a primitive element in GF(q). If the degree of $g(x)$ is r, then the length is $n = (q^r - 1)/(q-1)$. Since the code is a full-length code, every r-tuple in the base field GF(q^r) can be written as a multiple of one of the column vectors of the code's check matrix. Hence, there exists a unique element c in GF(q) and a unique positive integer $b < n$ such that

$$x - 1 \equiv cx^b \mod g(x) \tag{19}$$

or

$$\frac{x^{n-b}}{c} \equiv \frac{1}{x-1} \mod g(x). \tag{20}$$

A transposition error takes the form of $e_i(x-1)x^i$, with the corresponding syndrome equal to

$$S(x) \equiv e_i(x-1)x^i \mod g(x). \tag{21}$$

Since $(x-1)$ is relatively prime with $g(x)$, it introduces a one-to-one mapping between the set of syndromes of single transpositions and the set of syndromes of single (substitution) errors. In order to use the same syndrome-recognition circuit as in single-error correction, the desired syndrome to be recognized should be

$$S'(x) \equiv e_i x^{i+r+1} \equiv \frac{x^{r+1}}{x-1} S(x)$$
$$\equiv \frac{x^{n-b+r+1}}{c} S(x) \mod g(x). \tag{22}$$

Therefore, $S'(x)$ can be obtained by premultiplying the received message, $S(x)$, by $f(x)$, where

$$f(x) = f_{r-1}x^{r-1} + \cdots + f_1 x + f_0$$
$$\equiv \frac{x^{n-b+r+1}}{c} \mod g(x). \tag{23}$$

The corresponding decoder is shown in Fig. 5. Note that the switch W is open (at position 0) until the leftmost $r-1$ register stages are all zero during the second n-digit decoding cycle.

• *Correction of all single-substitution and some single-transposition errors*

Consider a single-error correcting code with generator polynomial $g(x) = (x-1)p(x)$, where $p(x)$ is a primitive polynomial of degree $r-1$. The code length in this case is $n = q^{r-1} - 1$. This code will not correct all single-transposition errors since

$$e_i(x-1)x^i + e_j(x-1)x^j \equiv 0 \mod (x-1)p(x) \tag{24}$$

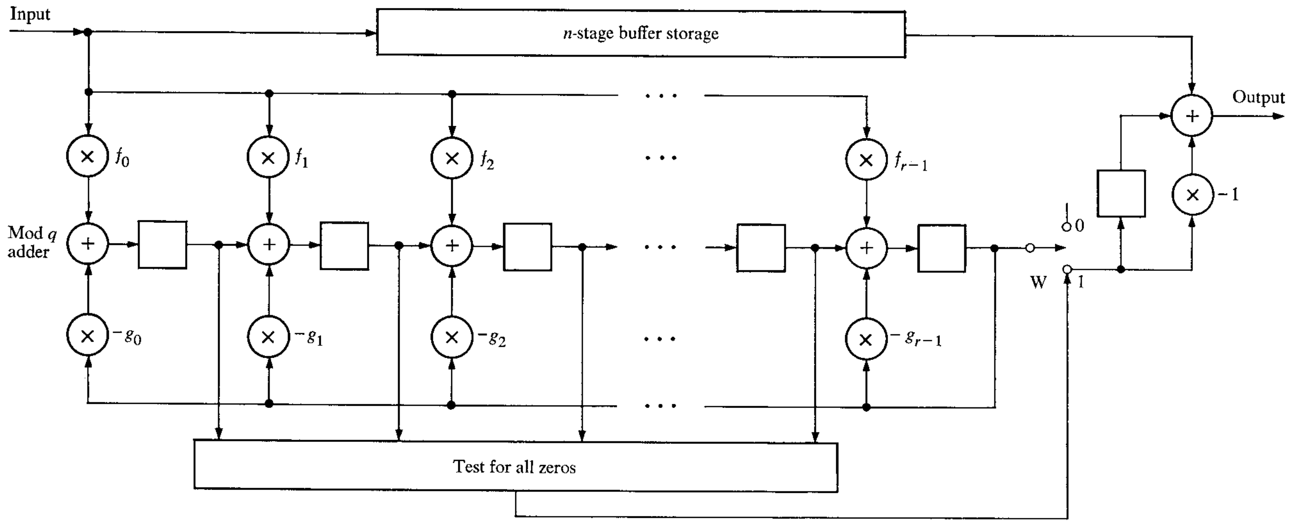
does have solutions. To show this, we write Eq. (24) as

$$x^{j'} \equiv -e_i/e_j \mod p(x), \tag{25}$$

where $j' = j - i$. Now if α is any root of $p(x)$, it can be shown that $\alpha^{n'}$ is a unique primitive element $\beta \in$ GF(q), where $n' = (q^{r-1} - 1)/(q-1)$. $(\alpha_1)^{n'} = (\alpha_2)^{n'} = \beta$ for any α_1, α_2 that are roots of $p(x)$. Therefore, Eq. (25) and, hence, Eq. (24) may be satisfied if j' is a multiple of n'.

A scheme that will correct a certain percentage of single-transposition errors and all single-substitution errors works as follows. With a given $p(x)$, the unique $\beta = \alpha^{n'}$ can be determined. At each i, $0 \leq i < n$, determine whether Eq. (25) is satisfied for $j' = j - i$ being $q-1$ nonzero multiples of n'. If Eq. (25) is satisfied, no correction is attempted. If not, we check to see if the message becomes a code polynomial when symbols at positions i and $i+1$ are transposed. If no code polynomial results, no correction takes place and the original received message is then investigated for next i. If a code polynomial results, the transposition between symbols at positions i and $i+1$ is assumed to be a successful correction.

The above partial correction of single-transposition errors is an extension to the code's original capability



⊗ x Multiply by x

☐ One-position shift register

⊕ Modulo q adder

Figure 5 Decoder for single-transposition error correction using a full-length single-error correction code.

of single-substitution correction. For if the received message is not divisible by $(x - 1)$, a substitution error may be assumed; while if the received message is divisible by $(x - 1)$ but not by $g(x)$, a transposition error may be assumed. One may also choose to correct single-substitution errors, but merely detect a possible transposition error in the meanwhile. This may indeed be an excellent strategy when errors are mostly single substitutions mixed with occasional transpositions.

• *Correction of multiple-transposition errors: $(x - 1)$ is not factor of $g(x)$*

Consider a multiple-error correcting code with a generator polynomial $g(x)$ that does not contain $(x - 1)$ as a factor. With minimum distance equal to $2t + 1$, the code can be used to correct t transposition errors.* This can be shown by writing the syndrome of any combination of $2t$ transposition errors as

$$S_{2t}(x) \equiv (x - 1)(e_1x^{i_1} + e_2x^{i_2} + \cdots + e_{2t}x^{i_{2t}}) \mod g(x). \tag{26}$$

It is easily seen that $S_{2t}(x)$ cannot be zero mod $g(x)$ since if it were, $S_{2t}(x)/(x - 1)$ must also be zero mod $g(x)$, as $g(x)$ does not contain $(x - 1)$ as a factor. This would, in turn, imply that the code has a minimum distance at most $2t$, which contradicts the original assumption. To correct t transposition errors, we see that

$$S(x) \equiv (x - 1)(e_1x^{i_1} + \cdots + e_tx^{i_t}) \mod g(x). \tag{27}$$

If we obtain the transformed syndrome

$$S'(x) \equiv \frac{xS(x)}{(x - 1)} \equiv x(e_1x^{i_1} + \cdots + e_tx^{i_t}) \mod g(x), \tag{28}$$

then the ordinary multiple-error syndrome recognition procedure can be carried out sequentially in time to make corrections. Let

$$g(x) = x^r + g_{r-1}x^{r-1} + \cdots + g_1x + g_0, \tag{29}$$

$$xS(x) \equiv s_{r-1}x^{r-1} + s_{r-2}x^{r-2} + \cdots + s_1x + s_0 \mod g(x), \tag{30}$$

and

$$\frac{xS(x)}{(x - 1)} \equiv s'_{r-1}x^{r-1} + s'_{r-2}x^{r-2} + \cdots + s'_1x + s'_0 \mod g(x). \tag{31}$$

It follows that

$$xS(x) \equiv (x - 1)(s'_{r-1}x^{r-1} + \cdots + s'_1x + s'_0) \equiv s'_{r-1}x^r + (s'_{r-2} - s'_{r-1})x^{r-1} + \cdots + (s'_0 - s_1)x - s_0 \mod g(x), \tag{32}$$

or,

* In this paper we are concerned only with disjoint transposition errors because they are the kind most likely to occur.

$$s_{r-1}x^{r-1} + \cdots + s_1 x + s_0 = [s'_{r-2} - s'_{r-1}(1 + g_{r-1})]x^{r-1} + [s'_{r-3} - (s'_{r-2} + s'_{r-1}g_{r-2})]x^{r-2} + \cdots + [-(s'_0 + s'_{r-1}g_0)]. \quad (33)$$

Equating coefficients and summing, we obtain

$$\begin{aligned} s_{r-1} &= s'_{r-2} - s'_{r-1}f_1, \\ s_{r-1} + s_{r-2} &= s'_{r-3} - s'_{r-1}f_2, \\ &\vdots \\ s_{r-1} + s_{r-2} + \cdots + s_1 &= s'_0 - s'_{r-1}f_{r-1}, \\ s_{r-1} + s_{r-2} + \cdots + s_0 &= -s'_{r-1}f_r, \end{aligned} \quad (34)$$

where $f_i = \sum_{i=0}^{i} g_{r-i}$.

Therefore,

$$\begin{aligned} s'_{r-1} &= (-1/f)(s_{r-1} + s_{r-2} + \cdots + s_0), \\ s'_{r-2} &= s_{r-1} + s'_{r-1}f_1, \\ s'_{r-3} &= (s_{r-1} + s_{r-2}) + s'_{r-1}f_2, \\ &\vdots \\ s'_0 &= (s_{r-1} + s_{r-2} + \cdots + s_1) + s'_{r-1}f_{r-1}. \end{aligned} \quad (35)$$

The above transformation is performed by the logic circuit augmented to the division circuit of $g(x)$, as shown in Fig. 6. Note that all the switches are thrown from position 0 to 1 after the *n*-digit received message is in. The syndrome transformation is completed after a single additional shift. Switches are then thrown back to position 0 and the ordinary syndrome recognition logic takes over. A premultiplication of x to the received message is applied to obtain the syndrome corresponding to $xS(x)$. An extra stage in buffer storage is used to compensate for the time needed for the syndrome transformation.

• *Correction of multiple-transposition errors: $(x - 1)$ is factor of* $g(x)$

Consider a multiple-error correcting code with a generator polynomial $g(x) = (x - 1)g'(x)$. If the minimum distance of the subcode generated by $g'(x)$ is $2t + 1$, then the original code can be used to correct t transposition errors and to detect, in addition, a single-substitution error. Instead of using a single division circuit corresponding to $g(x)$ to generate syndromes, we may cascade two division circuits corresponding to $(x - 1)$ and $g'(x)$ as shown in Fig. 7. The multiple-transposition errors, after passing through the division circuit of $(x - 1)$, will appear as multiple-substitution errors to the division circuit of $g'(x)$. In terms of syndromes, we have

$$S(x) \equiv (x - 1)(e_1 x^{i_1} + e_2 x^{i_2} + \cdots + e_t x^{i_t}) \mod g(x) \quad (36)$$

$$S'(x) \equiv \frac{xS(x)}{(x - 1)} \equiv x(e_1 x^{i_1} + e_2 x^{i_2} + \cdots + e_t x^{i_t}) \mod g'(x). \quad (37)$$

It follows that the same syndrome recognition circuit used for t-error correction with the code generated by $g'(x)$ can be used here. There are several minor modifications: 1) Multiple-transposition-error correction is carried out only if the division of the received message by $(x - 1)$ results in a zero syndrome. 2) If this syndrome is not zero, a substitution error is detected. 3) A premultiplication of x is used so that the correction of high-order digits can be done in time sequentially.

In describing the above correction schemes we have, in essence, demonstrated the validity of the following theorems:

Theorem 5: Let $g(x)$ be the generator polynomial of a code with minimum distance D. If $g(x)$ does not contain $(x - 1)$ as a factor, then the code can detect up to $(D - 1)$ transposition errors as well as up to $(D - 1)$ substitution errors. It can also be used to correct $t \leq [(D - 1)/2]$ transposition errors and meanwhile detect $(D - 1 - t)$ transposition errors.

Theorem 6: Let $g(x) = (x - 1)g'(x)$ be the generator polynomial of a code with minimum distance D. Also, let the subcode generated by $g'(x)$ be of minimum distance $(D - 1)$. Then this code can detect up to $(D - 2)$ transposition errors as well as up to $(D - 1)$ substitution errors. This code can be used to correct $t \leq [(D - 2)/2]$ transposition errors and meanwhile detect $(D - 2 - t)$ transposition errors. It also detects a single-substitution error in the presence of any number of transposition errors.

6. Concluding remarks

We have studied certain basic problems related to the detectability of most common types of errors encountered when digital data are entered at terminals by human operators. Several coding schemes with different degrees of control can be used in a detection-and-retransmission mode or, for some of them, in a correction mode.

Credit card systems with decimal key-punch terminals using one check symbol to detect single-substitution and most single-transposition errors have been in existence. Detection of all single transpositions in a decimal system with one digit has been suggested by Freeman.[3] Since many single-substitution errors cannot be detected by Freeman's scheme, it is effective only when transpositions are predominant among errors. The special decimal coding

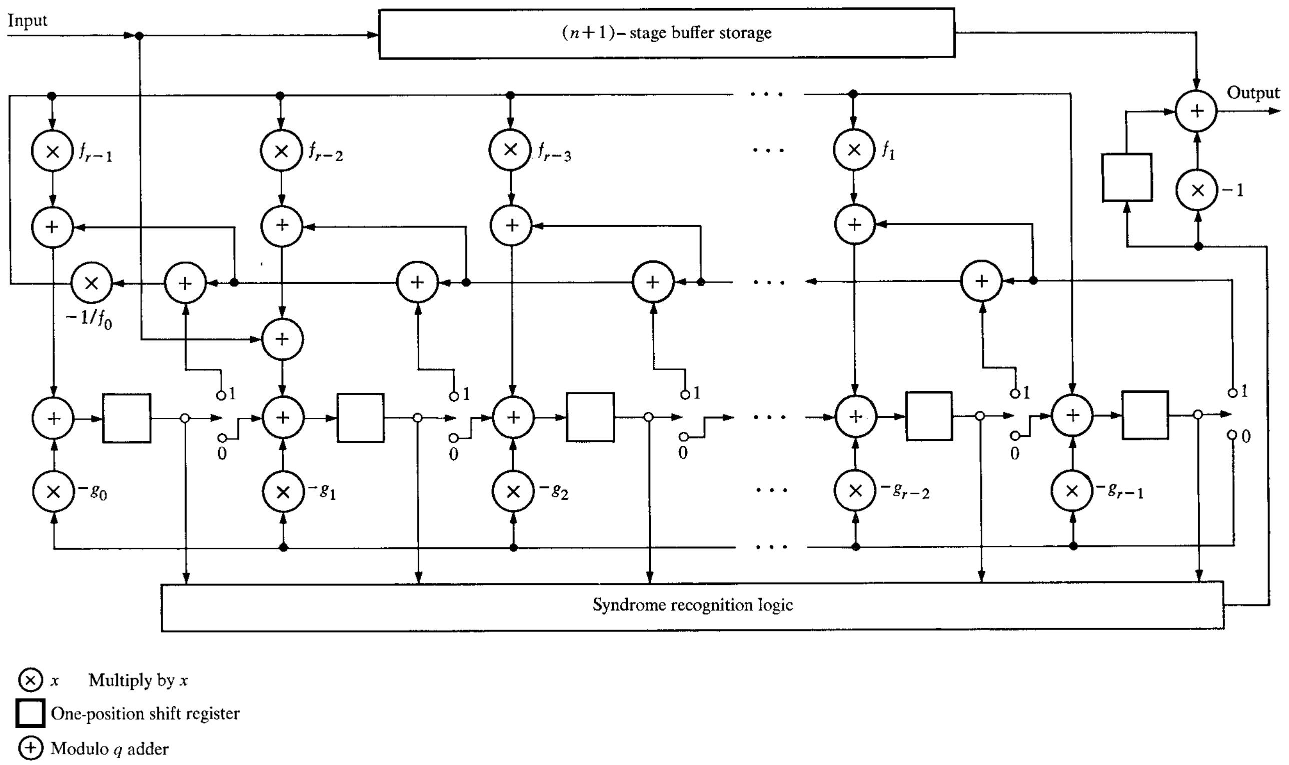
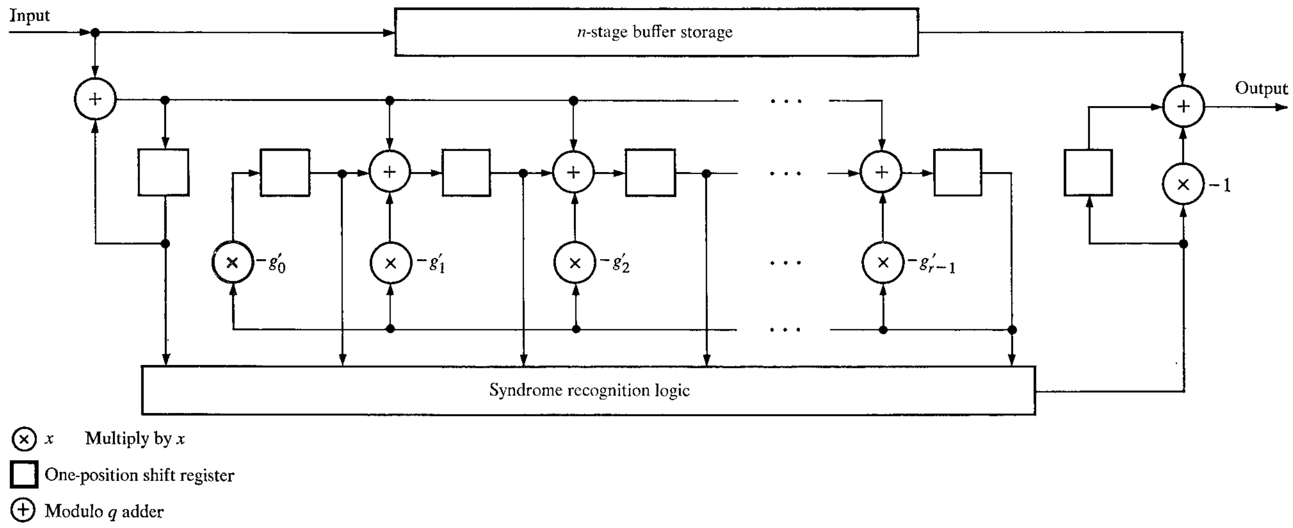


Figure 6 Decoder for multiple-transposition error correction, where the generator polynomial $g(x)$ does not contain the factor $(x - 1)$.

Figure 7 Decoder for multiple-transposition error correction, where the generator polynomial $g(x)$ contains the factor $(x - 1)$.



scheme described in Section 4 of this paper may be used if detection of both single substitutions and transpositions as well as single insertions and deletions is desired.

Polynomial codes described in the last section can be used in a variable length mode. The message length can be shorter than the natural code length n either where the information about the actual length is included in the message, or when the end of the message is identified by a special character or signal. Single deletions and insertions, originally detectable by the fixed code length, can now be detected if an extra check symbol is used as described in Section 3.

References

1. W. W. Peterson, *Error Correcting Codes,* MIT Press, Cambridge, Mass. 1961.
2. O. Ore, *Theory of Graphs,* American Mathematical Society, Providence, R. I. 1962.
3. H. Freeman, "Detection of transposition errors in decimal numbers," *Proc. IEEE* **55,** No. 8, 1500 (1967).

Received October 30, 1969