NTIA Report 83-133

A Method for Counting Errors, Deletions, and Additions When Both Transmitted and Received Data Are Known

W. J. Hartman

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary
David J. Markey, Assistant Secretary
for Communications and Information

September 1983

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A METHOD FOR COUNTING ERRORS, DELETIONS, AND ADDITIONS WHEN BOTH TRANSMITTED AND RECEIVED DATA ARE KNOWN

W. J. Hartman*

The problem of counting the occurrences of bit errors, block errors, bit additions or deletions and block additions or deletions is considered. A general method is developed for counting such occurrences for cases when the occurrences are not excessive. The definition of excessive varies with the amount of correlation in the transmitted data. Modifications of the general method are given for various cases of blocked data, with or without accompanying block identification, and for the special case when the transmitted data can be selected.

1. INTRODUCTION

Federal Standard 1033 (GSA, 1979; Seitz, 1980) and a similar standard proposed by the industry group, American National Standards Institute (ANSI, 1982) provide a set of performance descriptors for application to digital communication services. A second proposed standard (Seitz et al., 1981) deals with methods for measuring the performance descriptors. Wortendyke et al. (1982) have measured some of these parameters on the ARPANET. The present report describes methods for measuring the values of some parameters associated with the user information transfer, specifically, the bit or block errors, bit or block loss, and bit or block additions.

If we call the errors, additions, and deletions, events, the problem we consider is to describe the events as accurately as possible for a given set of data which passed through a digital system. We do not consider the additional problem of using these event descriptions to characterize the expected probability of occurrences over the system.

The methods developed here are described using some terminology from computer programming, because it provides a concise way of describing the sequence of events. This is not intended as a structure for implementing an efficient program.

In order to be specific, much of the description is given in terms of 8-bit bytes. The extension to other word lengths should be clear.

The organization of the report is designed to describe first the case of data that are random and uncorrelated. This gives the simplest description of the method

---

*The author is with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce, Boulder, CO 80303.

even though a program implementing the method would take the longest time to run. As the data become more structured, the description becomes more complicated, and (generally) the running time becomes shorter.

2. DESCRIPTION OF THE PROBLEM

The problem we consider here is described as follows: given two strings of binary data, one of which is supposed to be a replica of the other, but which may have errors (E), deletions (D) of data, or additions (A) of data, find an algorithm for locating those parts of strings that match, and determine (when possible) whether errors, deletions, and/or insertions have been made. We assume that no error is introduced in the comparisons. The two data strings might be thought of as a "transmitted" string and a "received" string. We will identify quantities associated with the "transmitted" string by capital letters, and quantities identified with the "received" string by lower case letters. Note that this identification does not reduce the generality of the algorithms which follow.

It will also be assumed that the data are ordered, and that no transpositions occur in the ordering between the two sets of data. That is, if bit i precedes bit j in the transmitted data, and these two bits occur as bit i' and bit j' in the received data, then bit i' precedes bit j'.

As a convention, we assume that the bits are numbered from right to left, i.e., bit 1 is the right most bit and the last bit is the left most bit in the string. We denote by B(i, j) [b (i, j)] that string of eight consecutive bits (a byte) which is i bytes to the left of the byte which has the jth bit in its right most position. The capital letters denote the strings from the transmitted data, and small letters those from the received data. See Figure 1 for an example. If i is negative, the byte is to the right. We denote by B(i, j) [+] b(i', j') the total number of mismatches between the two bytes. We will also have occasion to make use of words designated by $W(i_1, i_2, \ldots, i_n)$ $[w(i_1, i_2, \ldots, i_n)]$, of length n, made up of the bits from positions i_1, i_2, etc. These will be shortened as follows: W(i, ..., n) means the consecutive bits from bit i to bit n, W(i ..., k, k+m ... n) means the consecutive bits from 1 to n, with the bits k+1 to k+m-1 omitted, and W(f(i); i = k, n) means the bits selected from the positions f(i), i = k to n. Similar to the definition for bytes, we denote by W(...) [+] w(...) the number of matches between the bits in the positions corresponding to the associated arguments. The matchfunction for bytes varies from 0 to 8, while the upper limit for the matchfunction for words depends on the number of arguments in the word.

3. STATISTICAL BACKGROUND

We assume that the transmitted data is a sample from a balanced binary string, i.e., a string with (nearly) equal numbers of zeros and ones. We also assume that correlation (if any exists) can be described by a first order Markov process either on the data directly, or on a decimated set from the original data. The state diagram for the process is shown in Figure 2. This same diagram will be used for the case where some correlation exists between consecutive bits, or for the case where every kth bit is generated by the process. This latter case will be used for the case of data, for example where the bytes are correlated, but the bits within a byte are not. The probability of a transition from a 0 to a 1 is the same as the probability of a transition from a 1 to a 0 in keeping with the hypothesis that the data are balanced.

If the received data is assumed to be generated by a similar process, we obtain the four state diagram of Figure 3, where the first element in each state represents transmitted data and the second element represents received data.

If we now designate the two states 00 and 11 (matches) as state 1, and 01 and 10 (mismatches) as state 0, we obtain the state diagram shown in Figure 4. It should be noted that the steady state probability for the diagram in Figure 4, $P(0) = P(1) = 1/2$, as long as $p_1, p_2 \neq 0, 1$, showing that the average number of matches between the two data sets is n/2 bits when n bits are compared. It should also be noted that since $(p_1p_2 + q_1q_2) - (p_1q_2 + p_2q_1) = (p_1 - q_1)(p_2 - q_2)$, if $p_1 \neq 1/2$ and $p_2 \neq 1/2$, the model shows a tendency to remain in the same state whenever $p_1 > q_1$ and $p_2 > q_2$, or $p_1 < q_1$ and $p_2 < q_2$, and a tendency to change state whenever $p_1 < q_1$ and $p_2 > q_2$ or $p_1 > q_1$ and $p_2 < q_2$. For many practical situations, it is reasonable to assume that $p_1 = p_2$, and although some generality is lost, this assumption will be made for the remainder of this paper.

Since we will be primarily concerned with a large sample (n) we will make use of the normal approximation to the distribution of the occurences of matches which requires the mean, $\overline{X}$, and standard deviation, σ. These are given by (Gabriel, 1959)

$$\overline{X} = n/2 \tag{1a}$$

$$\sigma = 1/2 \left[n \frac{(1 + d)}{(1 - d)} - \frac{(1 + d)}{(1 - d)^2} (1 - d^n) \right]^{1/2} , \tag{1b}$$

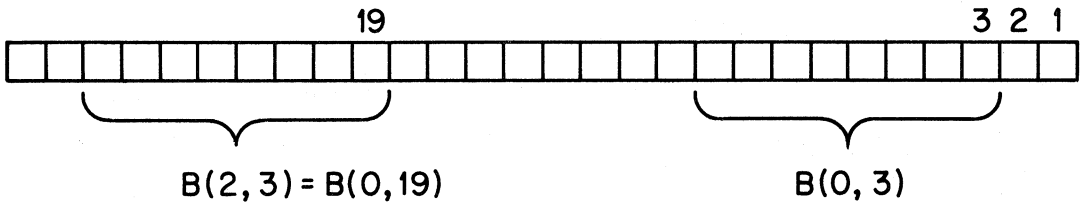

Figure 1. An example of the numbering of the bytes.

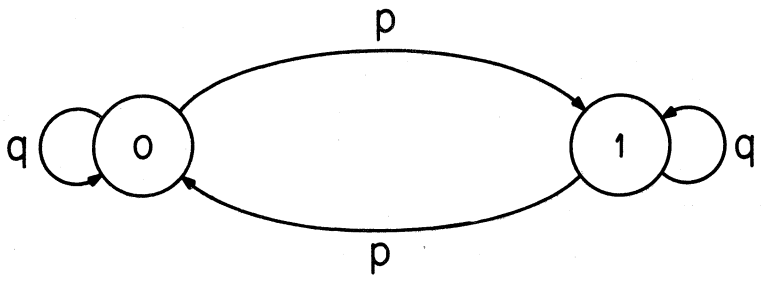

Figure 2. The state diagram for the Markov process describing the transmitted data.

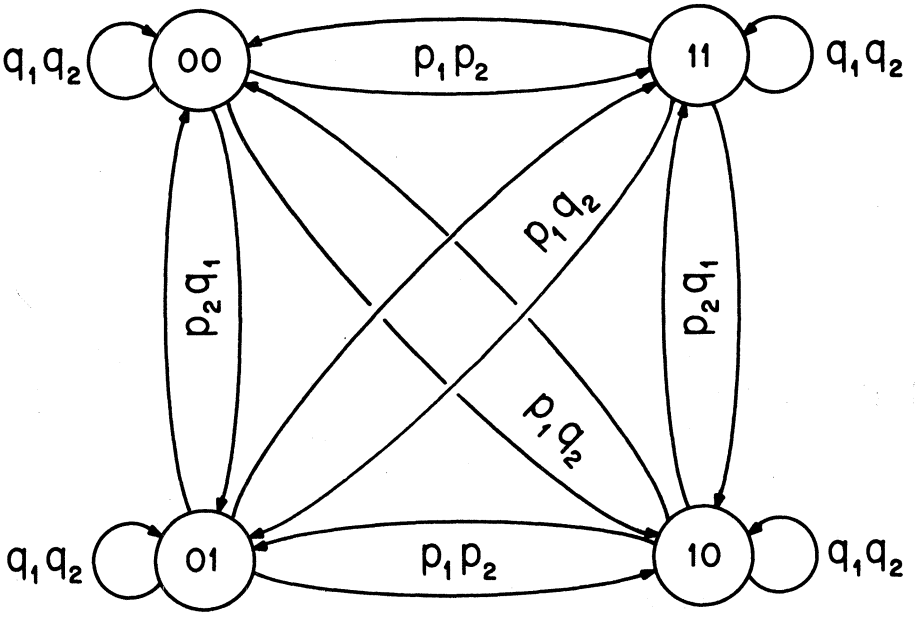

Figure 3. The four-state Markov process describing the combined transmitted and received data.

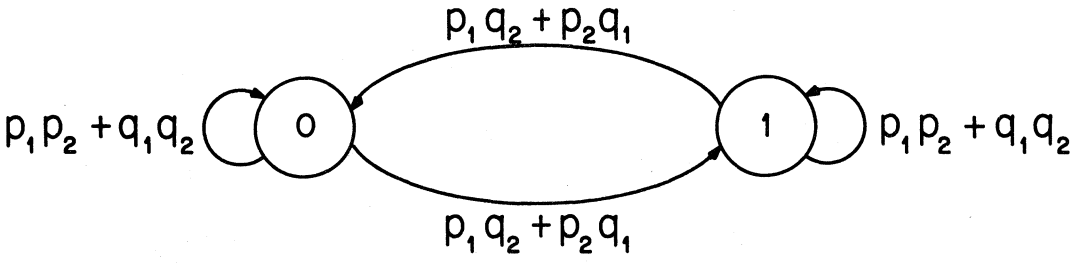

Figure 4. The state diagram for the Markov process describing matches (1) or mismatches (0) between the two data sets generated by the appropriate process.

where $d = (q - p)^2$. For $p = 1/2$, these reduce to the mean $(n/2)$ and standard deviation $(\sqrt{n/4})$ for the binomial distribution as expected. To estimate how much the term in (1b) deviates from the binomial distribution, we give values for d, p(or q), $\frac{1+d}{1-d}$ and $\sqrt{\frac{1+d}{1-d}}$ in Table 1. In this table, the length of a run (of 1's or 0's) which occurs with probability 1/2 is given, for the transmitted sequence as k, and for the comparison of the two sequences as k'. We see that large values of d, corresponding to large values of p, can produce long runs. As will be seen, this can cause problems in determining when two sequences are in synchronization.

The probability of k or more matches in n samples (between two sequences generated by the same Markov process as outlined above) is given approximately by

$$p_n \text{ (k or more matches)} \simeq \int_t^\infty \phi(x)\, dx,$$

$$\text{where } \phi(t) = \frac{e^{-t^2/2}}{\sqrt{2\pi}} \quad , \tag{2}$$

$$\text{and } t = \frac{k - 0.5 - \overline{X}}{\sigma} \quad ,$$

with $\overline{X}$ and σ given in (1). This approximation is best for small d and large n. Gabriel (1959) gives estimates of the fit of the actual distribution to the normal approximation for $d \leq 0.8$. An asymptotic estimate for p_n is given by

$$p_n \sim \frac{\phi(t)}{t} \tag{3}$$

From Table 1 it is seen that for d = .81, a run in the transmitted sequence of length 13 occurs with probability 0.5. For this d, a run of 89 occurs with probability 0.01 in the original sequence, and a run of 46 occurs in the comparison sequence with a probability of 0.01. It is clear that for highly correlated data, the chances of deciding that two strings match when in fact they are from two different processes is not negligible unless a large sample comparison is made. A method for overcoming this difficulty is presented later.

There are several computational methods available for estimating the correlation in the transmitted data, with the Fast Fourier Transform (FFT) methods being

Table 1. Comparisons of Parameters for Application of the Normal Approximation as Function of the Correlation of the Transmitted Bits, Expressed as p - q

d $(p-q)$	p	$\frac{1+d}{1-d}$	$\sqrt{\frac{1+d}{1-d}}$	(1) run length k for p	p^2+q^2	(2) run length k' for p^2+q^2
.01	.505	1.020	1.01			
.04	.6	1.083	1.04			
.09	.65	1.198	1.094			
.16	.7	1.381	1.175			
.25	.75	1.667	1.291			
.49	.85	2.922	1.709	4	.745	2
.81	.95	9.526	3.086	13	.9050	6
.9409	.97	32.841	5.731	22	.9418	11
.9801	.99	99.503	9.975	68	.9802	34

(1) that integer k such that $p^k \leq 1/2$ and $p^{k+1} \geq 1/2$

(2) that integer k' such that $(p^2+q^2)^{k'} \leq 1/2$ and $(p^2+q^2)^{k'+1} \geq 1/2$

Both of these may be interpreted as the number of consecutive 1's (0's), given an initial 1(0), that occur with probability 1/2.

among the most efficient. We will not consider the specific method used but only assume that we can compute the correlation function c(n) when necessary.

4. SOME ALGORITHMS

4.1 General

From our assumptions, we know the number of bits transmitted NT and the number of bits received NR. The first test is to determine NT - NR. If NT - NR > 0, then more bits have been deleted than added; if NT - NR = 0 then the same numbers of bits (perhaps zero) have been added as deleted; and if NT - NR < 0, then more bits have been added than deleted. Denote the number of deletions (additions) by ND (NA). If a priori information is available for ND (or NA), certain simplifications will occur as noted in the following.

The first problem is to obtain a synchronization of the received string with the transmitted string. This procedure varies for different transmitted strings. In all cases, however, we assume that not more than 25% of the received bits are in error, where deletions and additions are not counted as errors. It would be possible to allow a higher percent of errors if it were known that no deletions or additions occurred, but this will not be considered here.

4.2 Uncorrelated Data (Binomial Distribution)

We begin by zeroing the registers to be used for counting correct bits RC, Errors RE, additions RA, deletions RD, right data shifts RSR, and left data shifts RSL.

We next check to see if a tentative alignment exists at either end of the two strings. We first test for condition C1,

$$C1: \quad B(0, 1) \; [+] \; b(0, 1) \leq 1 \quad .$$

If this is true, we go to START. If not, test condition C2,

$$C2: \quad B(0, NT - 8) \; [+] \; b(0, NR - 8) \leq 1 \quad .$$

If this is true, reverse the numbering or both bit strings and go to START.

If neither C1 or C2 are true, we perform the following tests, where TL and K1 are constants, which are discussed later.

First check if

$$C = \sum_{i=0}^{TL} B(i, 1) \; [+] \; b(i, 1) \leq K1 \quad .$$

If it is

(1*) set RE = C, RC = 8 (TL + 1) - C

(2*) renumber the bits in both strings so that bit i (bi) becomes bit i - 8 (TL + 1),

(3*) discard the bits with negative numbers,

(4*) NT = NT - 8(TL + 1), NR = NR - 8(TL +1), and

(5*) go to START.

If $C > K_1$ check if

$$C = \sum_{i=0}^{TL} B(-i, NT - 8) \; [+] \; b(-i, NR - 8) \leq K1 \quad ,$$

where the (-i) indicates bytes to the right of the indicated start. If it is, reverse the numbering of both strings, perform (1*), (2*), (3*), (4*), and (5*). If NT = NR, go to ERROR, otherwise continue.

If C > K1, set m = min (NT, NR) and check if

$$C = \sum_{i=0}^{TL} B(-i, m-8) \; [+] \; b(-i, m-8) \leq K1 \quad .$$

If it is, reverse the numbering of the bits in both strings, starting with bm = b1, perform (1*), (2*), (3*),

(6*) if NR > NT, set RA = NR - NT and set NR = NT = m - 8(TL + 1) or if NT > NR, set RD = NT - NR and NR = NT = m - 8(TL + 1),

and (5*).

If NT < NR and C > K1, check if

$$C = \sum_{i=0}^{TL} B(i, 1) \; [+] \; b(i, NR - NT + 1) \leq K1 \quad .$$

If it is, perform (1*),

(7*) renumber the bits in the received string so that bit NR - NT + 1 becomes bit 1,

perform (2*), (6*), and (5*). If C > K1, go to ERROR.

If NT > NR and C > K1, check if

$$C = \sum_{i=0}^{TL} B(i, NT - NR + 1) \; [+] \; b(i, 1) \leq K1 \quad .$$

If it is, perform (1*),

(8*) renumber the bits in the transmitted stream so that bit NT - NR + 1 becomes bit 1,

perform (2*), (6*), and (5*). If C > K1, go to ERROR.

Except in the cases where we go to error routine, we have made the decision that we have obtained an alignment of the two sequences at either the beginning or the end, or at a shift of one sequence relative to the other. The cases that have been covered include those where all additions or deletions occur at the beginning or end of the sequence, and/or where error bursts do not occur at the beginning or end. Situations where bit additions and/or deletions occur both near the end and the beginning as well as throughout the data will result in no decision.

The probability of deciding that the data is aligned (by the preceding process) when in fact it is not cannot be calculated. However, under certain other hypotheses some estimates of the probability of this occurring at random is given in the discussion of the ERROR routine.

We have explicitly assumed that there are enough bits in both data strings to carry out the operations. However, in the following routines we must test to determine if we are approaching the end of a string, since at each stage we are updating NT and NR. If we are near the end, we go to a DONE routine, which handles the small number of remaining bits.

START: Let AT = [NT/8], AR = [NR/8] and AM = min (AT, AR), where [.] indicates the greatest integer. Calculate

$$C = B(i, 1) \; [+] \; b(i, 1) \quad i = 0, 1, \ldots AM.$$

At each step, if C = 0 and i = AM, go to DONE1. If C = 0 and i ≠ AM, calculate another C.

At the first occurrence of C ≠ 0 (say at i = i') set RC = RC + 8i', NR = NR - 8 i', NT = NT - 8i', and renumber the bits so that the bit in the right most position of B(i', 1) [or b(i', 1)] is now bit 1, then go to the error routine.

ERROR: calculate AT, AR, and AM as before. If NM < TL, go to DONE2. Define

$$C(i) = B(i, 1) \text{ [+] } b(i, 1), \; i = 0, 1, 2, \ldots TL \ldots$$

where T = TL is a limit chosen as described later. If

$$X = \sum_{i=0}^{TL} C(i) < K1$$

set RE = RE + X, RC = RC + 8(TL + 1) - X, NT = NT - 8TL - 8 and NR = NR - 8TL -8. Renumber the bits, with bit 876 + 8 + 1 = bit 1, and go to START.

If X > K1 and NM < TU, go to DONE3. If NM > TU, and if

$$X = X + \sum_{i=TL+1}^{TU} C(i) < K2 \; ,$$

where TU and K2 are constants described later,

(9*) set RE = RE + X, RC = RC + 8TU + 8 - x, NT = NT - 8TU - 8, NR = NR - 8TU - 8, and go to START.

If X ≥ K2, test if C(TU) ≤ 1. If it is, test if

$$Y = \sum_{i=TU}^{TU+TL+1} C(i) < K1 \; .$$

If it is, do (9*) and go to START.

If C(TU) ≥ 1, go to SHIFT. In effect, if we return to START, we have decided that the T(= TE or TU) bytes of received data are the same as the T bytes of transmitted data except for X bits in error. If we go to SHIFT we need to determine whether additions and/or deletions have been made, or possibly a large burst of errors occurred.

In order to determine T_L, T_U, K_1, and K_2, we examine the growth of the probability of k or fewer errors (n-k or more matches) occurring at random out of a

total of n bits where, to be specific, we consider n of the form 2^{i+2} and k of the form $2^i - 1$, for i = 1, 2, 3, ... This gives an error rate $n/k \to 1/4$, and the t for the normal approximation to the binomial distribution is

$$t = \frac{n - k - \bar{X} - .5}{\sigma} = \frac{2^{i+2} - 2^i - 2^{i+1} - .5}{\sqrt{2^i}} = 2^{i/2} - \frac{.5}{\sqrt{2^i}}$$

Thus, by choosing i sufficiently large, we can make the probability of the random occurrence as small as we wish. For i = 2, 3, (16 and 32 samples), k = 3, 7 and $p = 1.06 \times 10^{-2}$, 1.05×10^{-3} respectively, calculated from the binomial distribution. For i = 4 (64 samples), k = 15, and $p = 8.8 \times 10^{-5}$. For i = 8 (1024 samples), k = 255 and $p \sim 10^{-50}$. Thus, a reasonable choice of bounds is TL = 7 (i.e., 8 bytes), K1 = 15, TU = 127 (128 bytes), and K2 = 255. The algorithm allows large numbers of errors for short periods to account for error bursts. If additional information about the occurrence of errors is available, the limits can be adjusted for this.

SHIFT: Before describing the SHIFT algorithm, it is necessary to make some assumptions concerning the types of additions A (or deletions D) which are admissible. An addition is the insertion into the received string of a string (1 or more bits) of bits that did not originate from the transmitted string--similarly for deletions. We assume that between any pair, A-A, A-D, D-D, or D-A, a minimum number of bits, DM>>1, in the received data originated from the transmitted data. We choose $DM \geq 8TU + 8$. Thus, we do not limit the number of bits that can be inserted and/or deleted, but do prohibit certain occurrences as exhibited in the following examples. (1) We do not allow the replacement of one block of data by a (nontransmitter originated) different block of data, since this would constitute an A-D pair without the required intervening block. (2) Dropping (or adding) one bit from each byte is prohibited.

At this stage, we are reasonably certain that B(0, 1) and b(0, 1) are not in alignment. Thus, we begin by attempting to align the sequences by the same procedure used in the initialization stage. If this is unsuccessful, we use a more sophisticated alignment procedure below.

First, check if C = B(0, NT - 8) [+] b(0, NR - 8) $\leq$ 1. If it is, reverse the numbering of the string and go to START. If C > 1, check if

$$C = \sum_{i=0}^{TL} B(-i, NT - 8) [+] b(-i, NR - 8) \leq K1.$$

If it is, reverse the numbering of both strings, perform (1*), (2*), (3*), (4*), and (5*). If NT = NR, go to RSHF.

If $C \geq K1$, check if

$$C = \sum_{i=0}^{TL} B(-i, m - 8) [+] b(-i, m - 8) \leq K1.$$

If it is, reverse the bit numbers, perform (1*), (2*), (3*), (6*), and (5*).

If $C > K1$, reverse the bit numbers and check if

$$C = \sum_{i=0}^{TL} B(-i, m - 8) [+] (-i, m - 8) \leq K1.$$

If it is, again reverse the bit number, perform (1*), (2*), (3*), (6*), and (5*). If $C > K1$, set FLAG = 0, and if NT < NR, go to RSHF. If NT > NR, go to LSHF.

Only RSHF (right shift) will be described here, since LSHF is the same with the shift being done on the transmitted data. RSHF is intended to discover the condition shown in Figure 5a, and LSHF the condition shown in Figure 5b.

RSHF: Let

$$D(j) = \sum_{i=0}^{TL} B(i, 1) [+] b(i, j), \text{ for } j = 1, 2, \ldots NR - 8TL - 8.$$

If, for some j = j1,

$$D(j1) \leq K1, \text{ test if } C = \sum_{i=0}^{TU} B(i, 1) [+] b(j, j1) \leq K2.$$

If it is, set RSR = RSR + j1 - 1, RE = RE + C, RC = RC + 8TU + 8 - C, renumber the data in the transmitted string so that bit 8TU + 8 + 1 becomes bit 1, in the received string bit 8 TU + 8 + j1 + 1 becomes bit 1, let NT = NT - 8TU - 8, NR = NR - 8TU - 8 - j1 and go to START. If $C > K2$, continue testing.

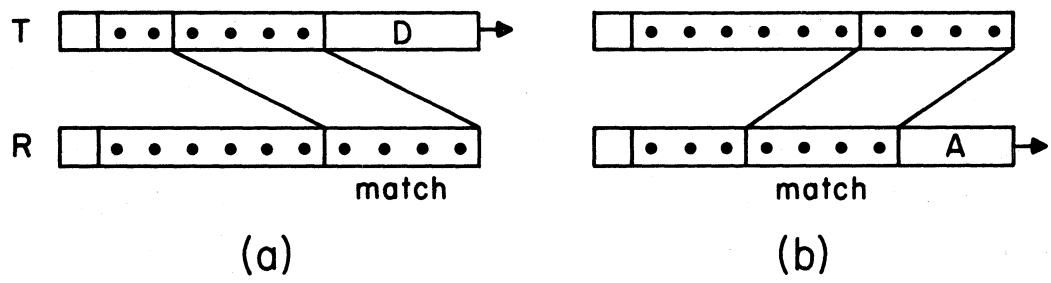

Figure 5. Diagrams showing (a) the left shift and the detection of deletions and (b) the right shift and the detection of additions, under the assumption that sufficient data exists between successive occurrences of additions or deletions to obtain a match.

If j > NR - 8TL -8 and FLAG ≠ 2, set FLAG = 1 and go to LSHF. If FLAG = 2, this implies that both the LSHF and RSHF routines have been used without finding a part of the sequences that are in sync. Thus, we declare that the data remaining at this point are ambiguous, and exit. It may be desirable to record the remaining data and reprocess it with different limits for the tests.

Rather than processing all of the data through one shift before going to the other, it may be desirable to process the data in blocks, alternating the shift direction.

DONE: These routines, which are not described in detail, are needed to process (approximately) left-over data. This block of data is not large enough to process in the shift routine, and hence will be regarded as having additions or deletions at the end of the string, if the string lengths are different, and having the error count produced by comparing the bits from 1 to m.

4.3 Correlated Data

One method of overcoming the problems encountered with correlated data is to increase the number of bytes examined at each stage of the program. Thus, the limits TL, TU, and DM would all be increased. This places some restrictions on the allowable errors, additions, and deletions, as well as requiring more computation time. A second method would be to use words in place of bytes, where the words consist of 8 bits of the form W(ki; i=1, 8), where k is a number such that the autocorrelation is essentially zero for shifts of k or more bits. For this method, DM should be increased to achieve the same results in the shift routine. The efficiency of the routine should be approximately the same as the original.

4.4 Formatted Data

Data are frequently grouped into records, or files, or both. These may or may not have identification (ID) such as record numbers, end of record information, etc. Two cases are most common, equal block length with no ID, or blocks with ID. If it is known that the possibility exists that an entire block (record, file) of data is added or deleted, the program can be modified to search for this possibility.

For data with block ID, this ID can be used to obtain an initial alignment of the two data strings or portions of these strings where each block that is aligned using the ID is treated in the usual way in the program, with any data for which a decision is reached being discarded, and any data which is declared

undecidable being replaced in its original location with respect to the remaining data.

For problem data, it is necessary to consider two types, that which contains only information such as an end of file mark, and that which contains file numbers or something similar.

In both cases, if an end of file mark is found, the number of bits in the file is then known. In addition, it is easy to count the number of files detected in the received data (NFR) and to compare this with the number of files in the transmitted data (NFT). The following situations can occur: (a) errors in an end of file mark, causing it to be read as data; (b) deletion of an end of file mark; (c) addition of an end of file mark; (d) addition of data with an end of file mark; (d') addition of data without an end of file mark; and, similarly (e) and (e') for deletion of data. As was the case with the additions and deletions of data in the unstructured data, it is necessary to restrict the addition and deletion of file marks. Thus, if the addition of a group of file marks (with or without accompanying added data) occurs, followed by some "good" data (possibly with errors), another file mark addition or deletion cannot occur until a number, MF, of "good" files with file marks have been received, and similarly for deletions. An example is shown in Figure 6. This prohibits insertions, for example, of end of file marks unless there is a minimum number of legitimate end of file marks between them. The same constraint as in the original program holds with respect to data additions after additions of file marks and/or data.

Thus the algorithm is designed as follows: Let $WF_k(\pm i;\ i = 1, TU)$ be the words following (+) or preceding (-) the kth file mark which we will denote by WF(k+), WF(k-), WF(k+), WF(k-) for short, where the choice of + or - depends on the application. We denote a right shift comparison by CR(i) = WF(1) [+] w F(i), i = 1, 2, ... NRF and a left shift.

$$CL(i) = WF(i)\ [+]\ wF(1),\ i = 1, 2, \ldots NLF.$$

By direct analogy with the shift routine, we see that we will obtain alignment around the file marks, unless one of the assumptions is violated. Once an alignment is found, we work both forward and backward (if necessary) from that point, using the program.

It can be seen that the number of shifts in the shift routine can be limited to approximately twice the length of the file under consideration, since we return to aligning files if we do not achieve synchronization in the shift routine.

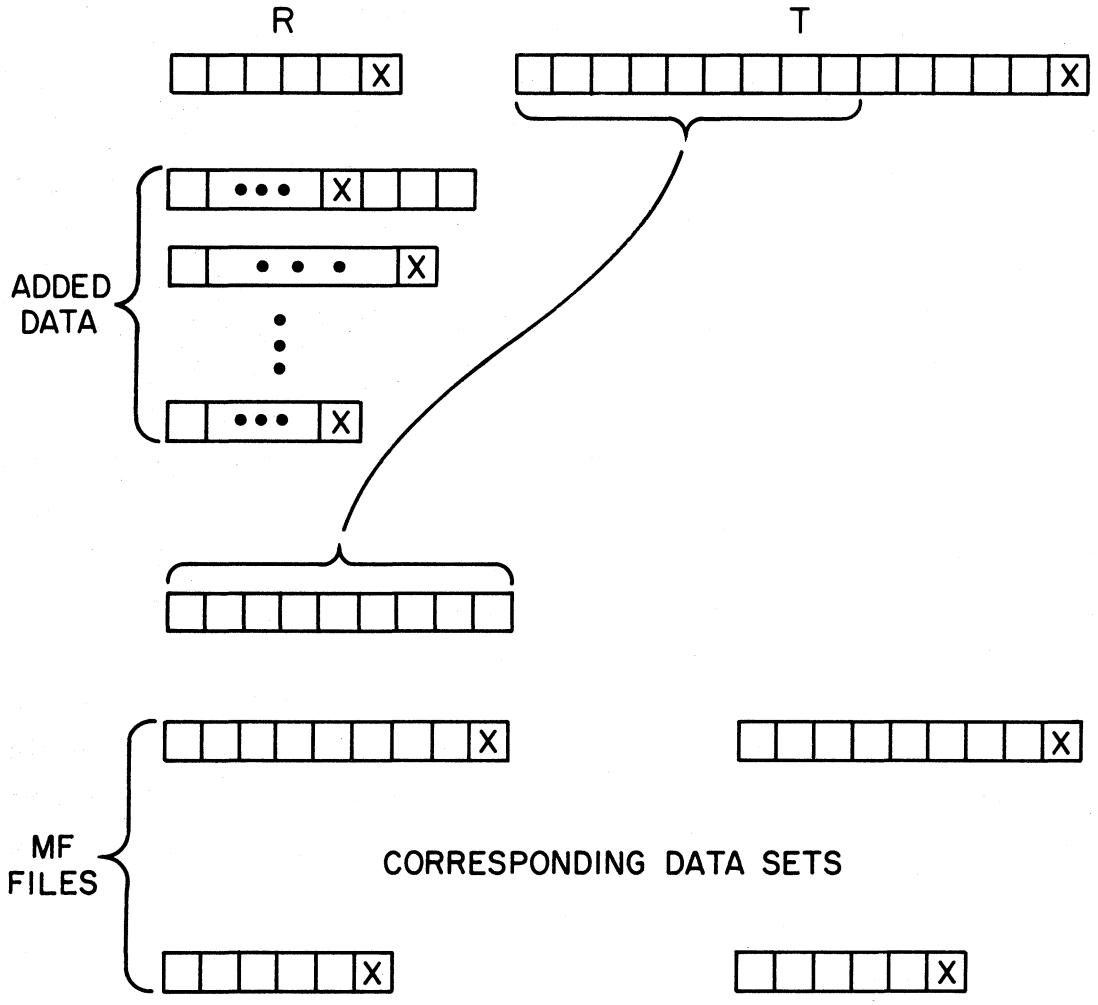

Figure 6. An example showing additions of data with file marks. The file marks are denoted by X.

It is also seen that either of the cases of a file mark having errors, or another symbol becoming a file mark because of errors, will be handled routinely in this process.

In the case of data having file numbers, additional information is available for alignment, and files for which the file numbers can be read in the received data can be easily processed through the program against the same number file from the transmitted string. The remaining data can then be processed in the same way as the data containing only file marks.

It is seen that the more information that is available from the data, the more complicated the algorithm, but usually, a given accuracy is obtained with less computations.

The last case that we consider is the case where the data is blocked into groups of equal size, without markers indicating the beginning or end of the blocks. In this case, we proceed similarly to the case of data with end of file markers. We limit the shift routine to approximately two block lengths, (or less if the blocks are long) and if synchronization is not achieved, we shift by block lengths to detect whether whole blocks have been added or deleted. If this does not occur, we must return and treat the data as if it were unformatted.

5. PRESELECTED DATA

If one has the opportunity to select the transmitted data, it is possible to select sequences which identify the bit and its position. One such sequence will be considered here.

The sequence we select is generated by a binary clock which updates for each bit of transmitted data. If the clock at time T has the representation

$$T = \sum_{i=o}^{n} a_i 2^i$$

where a_i = 0 or 1, then the bit transmitted at that time is

$$b = \sum_{i=o}^{n} a_i \pmod 2$$

[(mod 2) means that the bit is 1 if $\sum_{i=o}^{n} a_i$ is odd, or 0 if $\sum_{i=o}^{n} a_i$ is even].

This sequence is sometimes called the Thue-Morse sequence (Hershey, 1979). The sequence exhibits some desirable properties which are apparent upon examining Table 2 and Figure 7. In Table 2, it is seen that each string of the first 2^k bits is complemented in the string of 2^k bits starting at bit $2^k + 1$. This is shown in Figure 7 for 4, 8, and 16 bits where the bits are numbered left to right.

One can easily prove the following properties of the sequence.

If the k rightmost bits of the clock are 0's at time T, the sequence of generated bits bT ... b(T + 2^k - 1) are the same or the complement of the 2^k bits generated starting at T = 0.

We will designate a string of 2^k bits which starts with the clock having k rightmost 0 bits as either S(k) which is the string that starts at T = 0 or $\overline{S}(k)$, the complement of the string S(k). It can be shown by induction that at most two consecutive occurrences of S(k) or $\overline{S}(k)$ can occur. Hershey (1979) shows this for bits, i.e., at most two consecutive 0's or 1's can occur. In fact, if one starts at T = 0, and represents S(k) by {0}'s and $\overline{S}(k)$ by {1}'s, then the sequence of the {} symbols corresponds to the original sequence. Further, it can be shown that the sequence S(k) $\overline{S}(k)$ cannot occur more than twice in succession.

We note, however, that this matching of strings does not necessarily obtain for other starting points or shifts. For example, the 4 bit string b7 - b10 is the same as b1 - b4. Also, note the comparison between the strings b20 - b23 (1011) and b24 - b27 (0011) where there is one mismatch.

We use the symbol $\oplus$ to denote the bit-by-bit operation between two words, where for each corresponding bit,

$$1 \oplus 1 = 0 \oplus 0 = 0 \text{ and } 1 \oplus 0 = 0 \oplus 1 = 1 .$$

We shorten our definition of a word so that W(k, i) means the word of length 2^k bits starting with bit i. Thus, W(k, i) $\oplus$ W(k, i) = {0}, the string of 2^k zero bits, and W(k, i) $\oplus$ $\overline{W}$(k, i) = {1} the string of 2^k one bits.

b1							b16
0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
b17							b32
1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1

Figure 7. The alternation of bit patterns.

Table 2. Binary Clock Readings Starting at 0 and Corresponding Bit String

Clock	Bit	Bit #
0 0 0 0	0	1
0 0 0 1	1	2
0 0 1 0	1	3
0 0 1 1	0	4
0 1 0 0	1	5
0 1 0 1	0	6
0 1 1 0	0	7
0 1 1 1	1	8
.		9

(a) The first bit is complemented in the second bit.
(b) The first 2 bits are complemented in the third and fourth bits.
(c) The bits 1 through 4 are complemented in bits 5 through 8.

We now note that if W(k, i) is an S(k) (or $\overline{S}(k)$), then

$$\left.\begin{aligned} W(k, i) \oplus S(k) &= \begin{Bmatrix}\{0\}\\ \text{or}\\ \{1\}\end{Bmatrix} \\ \text{and} & \\ W(k, i + 2^k) \oplus S(k) &= \begin{Bmatrix}\{0\}\\ \text{or}\\ \{1\}\end{Bmatrix} \end{aligned}\right\} \quad \text{Condition 1.}$$

If W(k, i) is not an S(k) (or $\overline{S}(k)$), then condition 1 will not hold since the sequence S(k) (or $\overline{S}(k)$) cannot occur more than twice in succession, and the sequence $S(k)\overline{S}(k)$ (or $\overline{S}(k)S(k)$) cannot occur more than twice in succession. This then will form the basis for one of our tests.

As in the previous algorithms, we assume that NT and NR are known. We also assume a minimum number of error-free bits after any deletion or addition. Finally, since counting the errors, deletions, and additions is similar to the previous algorithm, this portion of the algorithm will not be included again. Instead, only the procedures for detecting these will be explained. Since the algorithm is designed to work with 8-bit bytes, we will further shorten W(3, i) to W(i) and S(3) to S.

We assume that the transmitted sequence begins with the clock at 0, and we define synchronization when the two data strings match, and the words W(i) and w(i) are both either S or $\overline{S}$. This is slightly different than the definition used in the previous algorithm where synchronization was defined for bytes starting at any bit.

5.1 Synchronization and Error Counting

Suppose ℓ bits of the transmitted string have been compared with ℓ bits of the received string (ℓ may be 0) and synchronization needs to be established.

(a) Choose $\ell' \geq \ell$ such that the binary representation of ℓ' has 0's in the three lowest order positions, and compute the string $W(\ell' + 1)$ and $W(\ell' + 9)$.

(b) Calculate, for i = 0, 1, 2 ... 7,

$$w(\ell' + 1 + i) \; [+] \; W(\ell' + 1) = C1(\ell' + i + 1)$$

$$w(\ell' + 9 + i) \; [+] \; W(\ell' + 1) = C2(\ell' + i + 1)$$

and $$w(\ell' + 9 + 1) \; [+] \; w(\ell' + 1 + i) = C3(\ell' + i + 1)$$

If $C1(\ell' + i_0 + 1) = 0$ or 8 and $C2(\ell' + i_0 + 1) = 0$ or 8, we declare byte alignment for the byte beginning at $\ell' + i_0 + 1$ and go to (c). If not, go to (d).

(c) At this point, we are reasonably sure that the data $w(\ell' + 1 + i)$ and $w(\ell' + 9 + i)$ came from the transmitted data, although it may not have come from the same section as $W(\ell' + 1)$. We can almost certainly rule out the possibility that the data $w(\ell' + 1 + i)$ and $w(\ell' + 9 + i)$ are added data. If $\ell' \neq 0$, and/or $i \neq 0$, then there is a high probability that some data have been deleted. If data have been deleted, then the situation is as pictured in Figure 8 where it is seen that the bit in the received data which we have tentatively labeled $\ell' + 1 + i$ should be labeled α. Thus, $\alpha - \ell - 1 - i_0$ bits have been deleted.

If the data do not have errors for a sufficiently long interval after the deletions, we can find the positioning of the clock using the method of Hershey and Lawrence (1981). This requires approximately $4(\alpha - \ell - i_0 + 1)$ error free bits, a larger number of bits if errors are permitted. Thus, if other information is not available, an error free period of approximately 4 times the length of the deletion is required for an exact determination of the clock position.

Instead of finding the exact position at this time, we can use the following error detection routine to count additional bytes in partial synchronization in order to try to limit the range of possible clock positions.

If $C1(\ell' + i + 1) = 0$, we renumber with bit $\ell' + i_0 + 1$ set to bit 1, and continue testing $C1(8\ell + 1) = 0$ $\ell = 0, 1, 2 \ldots$ until, for some ℓ_0, $C1(8\ell_0 + 1) \neq 0$.

If $C1(8\ell_0 + 1) = 0$ or 8 and $C2(8\ell_0 + 1) = 0$ or 8, we shift right by bytes, until we again reach the condition $C1(8\ell_0 + 1) = 0$ and return to the beginning of c. This step may not achieve total alignment all at once, but will eventually achieve it. This is guaranteed since we have the requirement that no errors or additions may follow directly an addition.

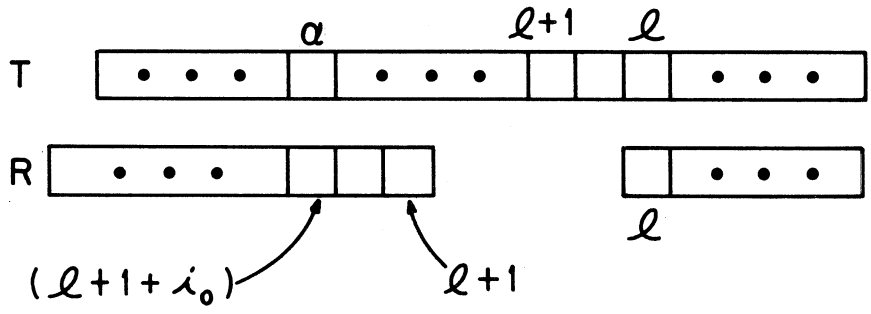

Figure 8. A demonstration of deleted data, showing the tentative alignment.

If $C1(8\ell_0 + 1) \neq 0$ or 8, but $C1(8\ell_0 - 7) = 0$, we assume that an error has occurred, or deletions have occurred, or additions have occurred.

If

$$C4: \quad |C3(8\ell_0 + 1) - C1(8\ell_0 + 1)| = 0 \text{ or } 8,$$

we assume that we are still in sync, but that errors have occurred. We count $C1(8\ell_0 + 1)$ as errors and go to d.

If C4 is false, we go to a.

C4 will be true whenever one has byte alignment, and 2 consecutive bytes do not have errors. If the former condition is not satisfied, we return to align the bytes, whereas if the latter condition is not satisfied, we go on to count the errors over a larger segment of data. Finally, we proceed to conduct additional tests for added data.

Since it is possible for the two strings to align over a given block of n bytes from the transmitted data and a different block from the received data, the possibility exists that a spurious alignment was achieved, and at the same point where the misalignment would be detected, errors, deletion or addition occurred. The probability of this is thought to be small, although very difficult to calculate.

(d) Upon entering this routine, we have one of several conditions. First, the data are aligned, but with an error burst. Renumbering, so that the first byte we examine starts with bit 1, we calculate

$$\sum_{i=0}^{7} C1(8i + 1) = C5.$$

if $C5 \leq 15$, we assume C5 errors. We then continue counting errors, using C5 until we reach the end of the data, or until $C5 > 15$, in which case we continue.

Second, the data are not in alignment (even partial alignment). This condition could be caused by either a deletion or an addition. If $NT > NR$, we use first the tests in (a), (b), and (c), and if this is unsuccessful, we assume an addition of data has occurred and go to the following.

Renumber the data so that the first bit to be examined is bit 1. As before, pick W(1) to be an S. Calculate $w(8k + i + 1)$ [+] $w(8k + i + 9) = C(k, i)$, $i = 0, 1 \ldots 7$. If $NT < NR$, we first try k such that $8k + 1 \geq NR - NT$. If this is successful in finding $C(k, i) = 0$ or 8, for $i = i_0$ we work backward byte-by-byte

to find the smallest k associated with $i = i_0$. If it is unsuccessful, or if NT > NR, we test using $k = k' \cdot 1024$, $k' = 1, 2 \ldots$ until we find $C(k, i) = 0$ or 8, for some k_0 and i. We then work backward to find the smallest k for this i_0. This determines the amount of data added. We then return to the calculation at the beginning of this section to determine errors. When again C5 > 15, we go to (a).

If we do not find k such that $C(k, i) = 0$ or 8, we declare the data undecidable.

(e) We now note that, even though we have not counted the number of bits deleted each time a deletion has occurred, by keeping track of the counts each time we set a bit number to 1, and by counting the number of additions from (d), we are able to count the total number of bit deletions. Thus, our imperfect alignment in (c) did not require the complete clock determination.

6. SUMMARY

It is seen that data with few errors, and a small number of deletions or additions, present no problems. It is also seen that certain conditions containing combinations of additions, deletions, and errors lead to situations in which it is impossible to make counts. The algorithms described are designed to make the "uncountable" situations rare, while making the calculations for "good" data as efficient as possible within the same general framework.

The algorithms effect a compromise between counting errors over a large segment of data and declaring data "out-of-sync." This is done to avoid the SHIFT routines when possible since these are the most time consuming portions of the procedure.

When the data are structured, the shift routines provide for a compromise between bit shifts and block shifts. The likely direction of the shift is determined by the bit counts in the two data strings.

For transmitted data with high autocorrelations, it is more difficult to make accurate counts. Even though this problem can be somewhat alleviated by increasing the appropriate limits used in the tests, an effort should be made to use uncorrelated data for best results.

7. REFERENCES

ANSI (1982), Proposed American National Standard No. X3.102, Data communication user oriented performance parameters, April. Available from the author of this report.

Gabriel, K. R. (1959), The distribution of the number of successes in a sequence of dependent trials, Biometrika 46, pp. 454-460.

GSA (1979), Interim Federal Standard 1033, Telecommunications: digital communication performance parameters, General Services Administration, August. Available from Office of the Manager, National Communications System, Technology and Standards Division, Washington, DC 20305.

Hershey, J. E. (1979), Comma-free synchronization of binary counters, IEEE Trans. Inform. Theory IT-25, No. 6, November.

Hershey, J. E., and W. F. Lawrence (1981), Counter synchronization using the Thue-Morse sequence and PSK, IEEE Trans. Commun. COM-29, No. 1, January.

Seitz, N. B. (1980), Interim Federal Standard 1033 reference manual, NTIA Report 80-55, December, (NTIS access. No. PB81-174898).

Seitz, N. B., K. P. Spies, and E. L. Crow (1981), Telecommunications: digital communication performance measurement methods, proposed Federal Standard 1043, Version 5, May. Available from the authors of this report.

Wortendyke, D. R., N. B. Seitz, K. P. Spies, E. L. Crow, and D. S. Grubb (1982), User-oriented performance measurements on the ARPANET: the testing of a proposed federal standard, NTIA Report 82-112, November, (NTIA access. No. PB83-159947).

FORM **NTIA-29**
(4-80)

U.S. DEPARTMENT OF COMMERCE
NAT'L. TELECOMMUNICATIONS AND INFORMATION ADMINISTRATION

BIBLIOGRAPHIC DATA SHEET

	1. PUBLICATION NO.	2. Gov't Accession No.	3. Recipient's Accession No.
	NTIA Report 83-133		

4. TITLE AND SUBTITLE

A Method for Counting Errors, Deletions, and Additions When Both Transmitted and Received Data are Known

5. Publication Date
September 1983

6. Performing Organization Code

7. AUTHOR(S)
W. J. Hartman

8. PERFORMING ORGANIZATION NAME AND ADDRESS
U.S. Department of Commerce
NTIA/ITS
325 Broadway
Boulder, CO 80303

9. Project/Task/Work Unit No.

10. Contract/Grant No.

11. Sponsoring Organization Name and Address

National Telecommunications and Information Administration
Washington, DC

12. Type of Report and Period Covered

13.

14. SUPPLEMENTARY NOTES

15. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The problem of counting the occurrences of bit errors, block errors, bit additions or deletions and block addition or deletions is considered. A general method is developed for counting such occurrences for cases when the occurrences are not excessive. The definition of excessive varies with the amount of correlation in the transmitted data. Modifications of the general method are given for various cases of blocked data, with or without accompanying block identification, and for the special case when the transmitted data can be selected.

16. Key Words *(Alphabetical order, separated by semicolons)*

17. AVAILABILITY STATEMENT	18. Security Class. *(This report)*	20. Number of pages
X☒ UNLIMITED.	Unclassified	30
☐ FOR OFFICIAL DISTRIBUTION.	19. Security Class. *(This page)*	21. Price:
	Unclassified	

☆U.S. GOVERNMENT PRINTING OFFICE: 1983–776-475/30