HUITIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 1er au 5 JUIN 1981



# CORRECTION CONJOINTE D'ERREURS ADDITIVES ET D'ERREURS DE SYNCHRONISATION

# COMBINED CORRECTION OF SUBSTITUTION AND SYNCHRONIZATION ERRORS

Said A. MADKOUR
Philippe GODLEWSKI

ECOLE NATIONALE SUPERIEURE SUPERIEURE DES TELECOMMUNICATIONS DEPARTEMENT SYSTEMES ET COMMUNICATIONS
46 RUE BARRAULT 75634 PARIS CEDEX 13

## RESUME

On considère le problème de la synchronisation d'un train d'éléments binaires perturbé par divers types d'erreurs : erreurs additives (c'est-à-dire, substitutions), mais aussi insertions et destructions de symboles. Après un survol des techniques de synchronisation envisagées actuellement : utilisation de marques fixes (entêtes de trames), codes "comma free", codes préfixes synchronisants, codes en blocs correcteurs d'erreurs additives et d'erreurs de glissement ; nous présentons une nouvelle approche basée sur l'utilisation de codes convolutionnels. L'algorithme de décodage qui en découle permet une acquisition de synchronisation plus rapide que les procédures "d'essais et d'erreurs" envisagées précédemment pour ce type de codage. En cas d'absence d'erreur la correction d'insertions (destructions) périodiques est assurée jusqu'à des fréquences élevées (une insertion tous les vingt symboles binaires). Des résultats de simulation sont présentés dans le cas plus général ou des erreurs additives indépendantes se superposent à des erreurs de synchronisation.

## SUMMARY

We consider the synchronization problem of a binary digital train perturbed with different types of errors : additive (substitution) errors as well as insertion and deletion of symboles. A survey of previously proposed synchronization techniques such as the use of markers, "comma-free" codes, prefix synchronized codes and block codes correcting both additive and synchronization errors is given. Then we present a new approach based on the use of convolutional codes and allows for the correction of the two types of errors under consideration. The algorithm which we give here achieves quicker synchronization than that obtained by the trial and error procedures already proposed for this type of codes. In absence of additive errors, the correction of periodic insertions (deletions) with high frequency (one insertion every twenty binary digits) is possible. Simulation results are presented in the general case where independent additive errors and periodic timing errors are present in the same word.

# CORRECTION CONJOINTE D'ERREURS ADDITIVES ET D'ERREURS DE SYNCHRONISATION

# COMBINED CORRECTION OF SUBSTITUTION AND SYNCHRONIZATION ERRORS

## I. INTRODUCTION

The presence of noise in a data transmission system can disturb the transmitted data in one or more of the following ways :

1) The symbol detector can confuse some symbol with another, this results in a, so called, substitution or additive error.
2) The symbol clock may skip a beat and results in the loss of one data symbol and so, a delection error occurs.
3) The symbol clock may insert an extra beat and consequently an extra data symbol is created causing an insertion error.

Insertion and deletion errors are known as timing or synchronization errors. An insertion causes a shift to the right for all the data symboles following its position, while a deletion results in this shift to the left. Such shifts cause a loss of the mutual synchronization between the transmitter and the receiver. Whenever a data sequence is affected by timing errors and compared symbol by symbol to its original one, a large distance (in Hamming sense) between the two sequences is found.

Under the title "synchronisation of binary messages", two problems can be distinguished. The first one is to derive the clock signal from the received message so as to divide the received signal into binary symboles. Synchronizers fulfil this task with adequate performance whenever the transmitted data stream has high transition density. A study of this problem in the case of convolutionally encoded data can be found in /1/, /2/. The second problem is to recover synchronization and/or to correct synchronization errors when the treated binary sequence actually suffers from insertions and/or deletions in presence or absence of additive errors. Such situation can occur in systems employins multiplexing techniques, as a result of an error in the interpretation of control characters, or due to inaccuracy of the bit clock.

Our work here is related to this second problem, and utilizes some syntactic properties of the received data but not the modulation characteristics of a waveform. Related work can be found in literature where the concepts of frame, block, code word, or character synchronization are used, such concepts depend on the data model considered and they are not strictly defined in general.

Section II briefs some of the techniques already proposed to achieve synchronization of binary messages. In section III an algorithm is presented to decode the binary convolutionally encoded data in presence of both substitution and synchronization errors. Section IV shows some simulation results obtained when this algorithm is used to decode a rate 1/2 convolutional code suffering from both substitution and timing errors. Conclusions are given in section V.

## II. SYNCHRONIZATION OF BINARY MESSAGES :

Binary messages can be transmitted through the channel as blocks of binary digits. Means should be provided to locate the start of each block, and so to keep the synchronization between the transmitter and the receiver. In some cases, such as the case of the Morse code letter space, a third symbol (neither zero nor one) is used for this purpose and called a "comma".

Besides violating the hypothesis of dealing with a binary channel (such systems can be considered as being "comma-free" with messages over a 3-symbol alphabet !), the use of a comma largely decreases the efficiency E of the channel for data transmission. This reduction occurs because the use of frames consisting of a comma followed by n-1 symbols acheives transmission rate of 1-1/n bits/symbol while the capacity of a noiseless channel using a 3 - symbol alphabet is $\log_2 3$ bits/symbol, so :

$$E = \frac{1 - 1/n}{\log_2 3} < 63.09\ \% \qquad (1)$$

for very large n.

The efficiency problem shows the need for other approaches to the solution of the synchronization problem. One such approach is that of the comma-free codes first considered by Golomb, Welch, Delbruck, and Gordon /3/, /4/. A comma-free code over an alphabet of size B is a block code (subset of $B^n$) such that the overlap $(a_m, a_{m+1}, \ldots, a_n, b_1, b_2, \ldots, b_{m-1})$ of any two successive code words $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ is never a code word.

Example : For B = 2, and n = 5, a comma-free code is : 01011, 01010, 01111, 01000, 01110, 01100.
In 1965 Eastman /5/ proved the existence of comma-free codes of odd length n (with number of code words $|C|$) which attains the upper bound

$$|C| < \frac{1}{n} \sum_{d|n} \mu(d)\, B^{n/d} \qquad (2)$$

where the summation is extended over all the divisors d of n, and $\mu(d)$ is the Möbius function.
It is easy to prove that $|C| = \frac{1}{n} B^n (1 + \varepsilon)$ (3)
where $\varepsilon \longrightarrow 0$ as $n \longrightarrow \infty$ , and hence the code efficiency, from the redundancy point of view, is :

$$\text{efficiency} = \frac{\frac{1}{n} \log |C|}{\log B} \sim 1 - \frac{1}{n} \log_B n \qquad (4)$$

This yields redundancy of 93.33 % for codes of length n = 100.
When the comma-free codes are used, synchronization can be acheived after processing at most 2(n-1) symbols in the receiver.

The "comma" concept has been improved in such a way that synchronization can be achieved while the transmission efficiency is not highly sacrificed. This improvement owes its origin to Barker /6/. The special letter "comma" is replaced by a specified sequence of the source symbols referred to as a "marker" or "sync sequence". At the receiver, a search is initiated for this marker and acquisition occurs when it is found. In this way the data stream is divided into frames and the start of every data word is marked. A basic theory of frame synchronization based on the marker concept is presented by Scholtz /7/, where frame sync acquisition and verification algorithms are given along with a discussion for the marker design. He considers blocks of D data bits separated by a known marker sequence of length M bits and transmitted through a memoryless binary symmetric channel (BSC) or Gaussian channel. If, for instance, we want to avoid false locks on data with probability 99 % in a noiseless channel a marker with M = 13 followed by a data frame of 87 symbols may be used achieving efficiency of 87 %. Marker-oriented design can suffer from false acquisitions even in noiseless channels.

# CORRECTION CONJOINTE D'ERREURS ADDITIVES ET D'ERREURS DE SYNCHRONISATION

# COMBINED CORRECTION OF SUBSTITUTION AND SYNCHRONIZATION ERRORS

In 1960 Gilbert /8/ proposed a special class of comma-free codes capable of achieving sync after observing at most n symbols, where n is the code length. His idea is to start each code word by a marker of length M < n and to choose the remaining symbols of the code word such that the marker cannot appear in any other place of the word or due to an overlap with the marker of the next word. These codes are known as "comma codes" or "prefix synchronized codes". The number of different n-tuples which can be achieved under these constraints depends on the choice of the prefix and is maximum if the prefix is of the form 11..0. With the suitable prefix, the redundancy in these codes approaches that of the comma-free codes i.e redundancy $< (\log_2 n)/$ for sufficiently large n. Like comma-free codes, these codes cannot correct any additive or synchronization error. Encoding using the full capabilities of these codes is rather complicated, a similar but simpler approach is adopted by the HDLC procedures /9/. The HDLC procedures yield variable length blocks.

Neumann /10/ has dealt with some classes of variable length codes which can regain synchronisation after some average delay without the need of any sync sequence.

In 1965 Ullman has shown that the minimum redundancy necessary for a code of block length p to correct a single sync error approaches $\log_2 p + 1$, as p approaches infinity /11/. He has also presented fixed block length codes which correct single sync errors and have redundancy less than $\log_2 p + 4$ i.e they are near optimal /12/. The structure of Ullman codes is shown in Fig 1, and consists of a synchronizing sequence (the sequence 011) followed by a redundant internal word such that the last bit of the sync sequence is the first bit of the internal word.

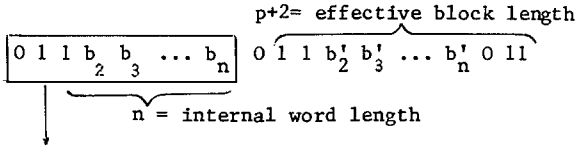


Fig 1. The structure of Ullman code.

If the length of the internal word is n, the repetitive pattern of the code has length p = n+2 , and the effective block length, defined as the distance over which only one error can be corrected, is p + 2. The sync sequence has the impotant property that it detects sync errors and determines their direction. The patterns 001 and 101 result only when there is an insertion, and 110 and 111 result only when there is deletion ; 010 may be a result of an insertion or a deletion in the sync sequence itself. The sync sequence may also appear to be correct in spite of the occurance of one sync error in it.

A parameter A on sequences of length n is defined. The sequence is divided into runs defined as subsequences of adjacent and identical bits. The runs are numbered in order starting with zero for the leftmost run. Let $N_j$ denotes the number of bits in the run whose number is j, and m denotes the total number of runs, then A is defined as :

$$A = \sum_{j=0}^{m} j\, N_j \qquad (5)$$

Ullman proved that : $A = \sum_{j=1}^{n-1} (n-j)\, u_j \qquad (6)$

where $u_j = b_j + b_{j+1}$

The expression (6) facilitates encoding of information bits into a code word. The n-bit internal word is selected only by the creterion that its value of A should be divisible by (2n-1).

At the decoder the calculated difference in A, $\Delta A$, and the observation of the sequence received in the position of the sync sequence are sufficient to correct any sync errors within the effective block length. Levenshtein has constructed codes with similar performance. He has defined a general metric which considers the simultaneous occurence of insertion, deletion and substitution errors, and known as Levenshtein metric /14/.

Several methods of constructing additive-error-correcting codes capable of recovering synchronization are known. One technique suitable for such purpose is based on "extending" the used cyclic code /13/. Consider the additive-error-correcting cyclic (n,k) code generated by g(x). Every code word in this code has the form :

$$C(x) = i(x).\, g(x) \qquad (7)$$

where deg i(x) < k

To extend the capabilities of this code to include sync recovery, we restrict i(x) to have the form :

$$i(x) = i'(x)\, g_o(x) + 1 \qquad (8)$$

where $g_o(x)$ has degree j , exponent e < n and divides $(X^n+1)/g(x)$ ; i'(x) is an arbitrary polynomial of degree less than k - j.
Let $C = (C_{n-1}, \ldots, C_{n-a}, \ldots, C_a, \ldots C_1, C_o)$

denote the coefficients of C(x). The corresponding coefficients of the "extended" word are :

$$C' = (C_{a-1}, \ldots, C_o)\ |C|\ (C_{n-1}, \ldots, C_{n-a}) \qquad (9)$$

where "|" stands for 'concatenation', and $a \leqslant \lfloor \frac{e-1}{2} \rfloor$ . (10)

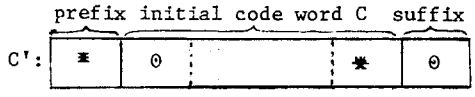


Fig 2. Format of "extended" cyclic code.

The resulting code is able to recover sync if slippage of s symbole occurs where - a ⩽ s ⩽ a but it does not correct any sync error.

Example : if we want to use the single - additive - error - correcting (105, 98) code to recover slippages of one symbol also, we may choose $g_o(x) = X^2+X+1$ with exponent e = 3, we get the "extended" (107, 96) code which has redundancy of about 10 %.

The cyclic property of cyclic - additive - error - correcting codes represent a difficulty for synchronization. One approach to overcome this difficulty is to break the cyclic structure by adding a specified pattern to every code word in the transmitter and then to subtract the same pattern in the receiver. The resulting code is referred to as a coset code /13/.

The general problem of correcting both substitution and sync errors seems to be rather complicated. A code proposed for this purpose is

presented in /16/ where correction of a single burst containing both sync and additive errors in the code word is possible. The key idea for the construction of this code is to use an interleaving of two codes, the first of them is a low rate code like the maximum length code used mainly for synchronization purpose, while the second one is a burst error correcting code whose function is to transmit the main part of the information. The additional redundancy needed to correct the sync errors depends on the capabilities of the two component codes.

When data is convolutionally encoded, two types of synchronization are of interest, namely branch sync and node (symbol) sync. Viterbi decoding algorithm implicitly ensures the branch sync. If node sync is lost, a sharp deviation of the likelihood metric to the wrong direction is noticed and a search is initiated for the good node using a trial and error technique within every branch /15/. The algorithm which we present here is a modified version of Viterbi algorithm and allows to correct not only additive errors but also sync errors besides achieving the two types of synchronization quickly.

## III. AN ALGORITHM FOR CORRECTING BOTH SUBSTITUTION AND TIMING ERRORS USING CONVOLUTIONAL CODES :

### a) Convolutional codes :

A (n,k) convolutional encoder with constraint length K is a K - stage shift register with n linear algebraic function generators, one for each output part. The input data, which is usually binary, is shifted along the register k bits at a time. The resulting code has rate R,

$$R = \frac{k}{n} \tag{11}$$

The well known (2,1) code with K = 3 which is usually treated in literature /13/, /15/ will be used here as a current example to explain the proposed decoding algorithm.

Results given in the next section correspond to this example. This code has the basic generator matrix G :

$$G = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{12}$$

which leads to the encoder of Fig. 3.
The output bits and transitions between states can be labeled by the state diagram of Fig. 4. In this diagram the "states"correspond to the last two input bits to the encoder, while transitions are denoted by solid directed line when the input is "0" and a dotted directed line for an input "1".

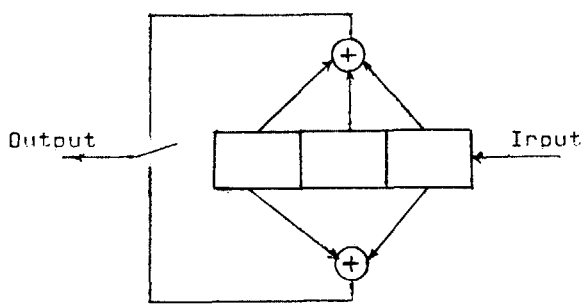


Fig. 3. (2,1) Convolutional code with K = 3

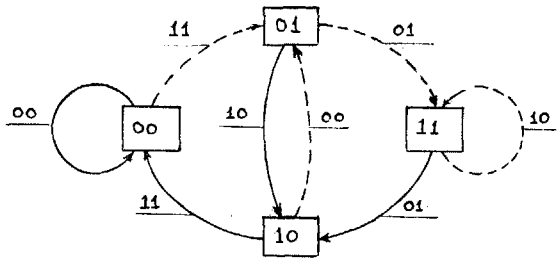


Fig. 4. State diagram of encoder in Fig. 3.

### b) Modification of the state diagram :

The state diagram of Fig. 4 is quite enough to decode whenever the treated sequence suffers from substitution errors only. When both substitution and timing errors are possible, it is advantageous to modify the state diagram in such a way that each branch is labeled using only one output (instead of two in our example). This modification is achieved using, so called, "intermediate states", in distinction of the "main states" of the initial state diagram. The intermediate states are specified by concatenating an input to the current main state. The number of states in the initial state diagram is $N_s$,

$$N_s = 2^{k-1} \tag{13}$$

while that of the modified diagram is $N_m$,

$$N_m = N_s(2^k(n-1) + 1) \tag{14}$$

The modified state diagram corresponding to our example is shown in Fig. 5. The possible transitions between states are considered in detail in Fig. 6. a, b whenever we are in a main or an intermediate state.

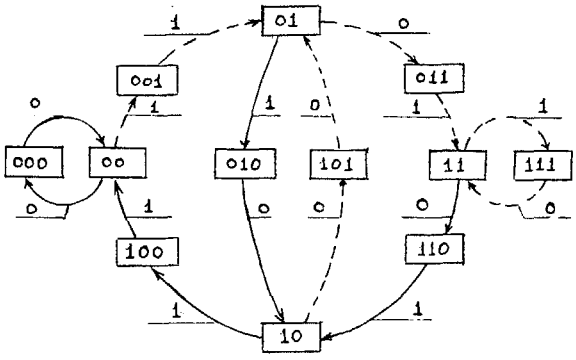


Fig. 5. Modified state diagram

CORRECTION CONJOINTE D'ERREURS ADDITIVES ET D'ERREURS DE SYNCHRONISATION

COMBINED CORRECTION OF SUBSTITUTION AND SYNCHRONIZATION ERRORS

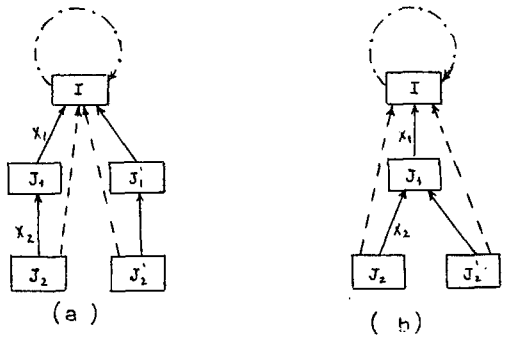


Fig. 6 a) Transitions leading to a main state.

b) Transitions leading to an intermediate state.

——→ additive error

----→ deletion

-·-·→ insertion

If the received symbol is considered as an insertion we should stay at the state where we were, while a deletion causes transition between states of the same type.

Transitions are weighted. For the case concerned, the weight DERR of a substitution error is the Hamming distance $d_H$ between the received symbol X and the label y of the approperiate branch in the state diagram.

i.e $DERR = d_H(x,y)$.

The weight allocated to an insertion is a constant DINS >DERR, and that associated with a deletion is another constant DDEL >DERR.
The values of DINS and DDEL are to be adjusted in accordance to the parameters of the channel under consideration.
States $J_1$, $J_2$, $J'_1$ and $J'_2$ are said to be leading to the state I (see Fig. 6) and the set Q(I) is defined as the set containing all the states leading to state I.

i.e $Q(I) = \{J ; J \text{ is leading to } I\}$

note that $I \in Q(I)$.
T(X, J, I) is defined as the weight of the transition from state J to state I when a symbol X is received
for insertion : $T(X, I, J) = V(I) + DINS$
for deletion :

$$T(X, I, J) = V(J_2) + DDEL + \min (d(X, X_1), d(X, X_2))$$

for additive error : $T(X, I, J) = V(J_1) + d(X, X_1)$
where V(J) is the log likelihood of state J.

c) The algorithm :

(1) Initialisation : $V(I) \leftarrow 0$ for all I, $1 \leqslant I \leqslant N_m$ ; $L \leftarrow 1$ ; $L_x \leftarrow L_{x1}$

(2) Read symbol X.

(3) For all I, $1 \leqslant I \leqslant N_m$
Compute :

$$V'(I) = \min_{J \in Q(I)} (V(J) + T(X, I, J))$$

and find the state $J_s$ which achieves the minimum ; store $J_s$ : $S(I,L) \leftarrow J_s$

(4) $L \leftarrow L+1$
$V(I) \leftarrow V'(I)$ , $1 \leqslant I \leqslant N_m$
IF $L < L_x$ GO TO (2)

(5) Stop.

The stored values S(I,L) are used to select the good survivor. This selection can be made in every step if we decode symbol by symbol, or every q steps if we output a block of q symbols every time.

d) Parameters affecting the performance of the algorithm :

Concerning the channel :

(1) The probability of substitution errors $P_e$.

(2) The insertion rate $r_s$ (deletion rate $r_d$).

Concerning the decoder :

(1) The values of DINS and DDEL.

(2) The decoding delay in symbols.

(3) The decoding starting state, if the decoder starts decoding from wrong state some errors occur before the branch synchronisation is achieved.

## IV SIMULATION RESULTS :

Results presented here correspond to the following parameters :

. The channel model is a cascade of sync error source introducing periodic insertions of a rate $r_s = 0.02$, and a memoryless B S C with substitution error probability $0.01 \leqslant P_e \leqslant 0.06$.

. DINS = 1.1 and DDEL = 100

. Decoding delay = 64 symbols.

. Length of each decoded word = 2040 symbols (including 40 insertions).

Parameters evaluated are :

(1) Residual error rate RER ($P_e$ , $r_s$) : It is evaluated as the Hamming distance $d_H$ between the input and the output of the decoder in parts of the word decoded without false timing errors. This distance is averaged over the number of decoded symbols.

(2) Residual insertion rate RIR ($P_e$ , $r_s$) : It is evaluated as an inverse of the average number of symbols between insertions which are falsely considered or actually present but neglected by the decoder. Residual insertions are not periodic but look like Bernoulli process
Performance evaluation :

a) In presence of insertions only :

Periodic insertions with rate up to 0.05 are corrected in all tested cases.

b) In presence of additive errors only :

Simulation agrees with the logic consequence that when DINS and DDEL are properly adjusted no false timing errors are considered by the decoder

and RER ($P_e$, 0) attains the same values evaluated using the ordinary Viterbi algorithm (Fig. 7). Bad values of DINS or DDEL introduce some false timing errors at the decoder output but the residual error rate remains to be identical to that resulting from Viterbi algorithm in the parts of the word between these false errors.

c) In presence of both additive and timing errors :

Fig. 7 shows the relation between log RER ($P_e$, $r_s$) and $P_e$ for periodic insertions of rate $r_s = 0.02$ and DINS $\simeq$ 1.1. The results show that we can roughly apply the following relation :

$$RER\ (P_e, r_s) \sim O(RER\ (P_e + r_s, 0))$$

Example : For $P_e = 0.03$ the residual insertion rate is $8.9 \times 10^{-5}$ for input insertions of the rate $r_s = 0.02$ and the corresponding residual error rate is $1.799 \times 10^{-2}$ which is not very far from the value of residual error rate of $1.197 \times 10^{-2}$ corresponding to $r_s = 0$ and $P_e = 0.05$.

Table 1 shows the values of RIR ($P_e$, 0.02) for different values of $P_e$.

| $P_e$ | RIR ($P_e$, 0.02) |
|---|---|
| 0.03 | $8.996 \times 10^{-5}$ |
| 0.04 | $2.402 \times 10^{-4}$ |
| 0.05 | $4.599 \times 10^{-4}$ |
| 0.06 | $7.698 \times 10^{-4}$ |

Table 1. Relation between RIR and $P_e$.

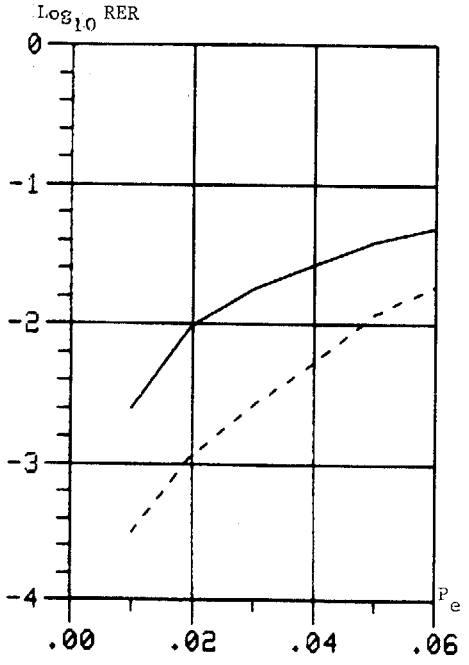


Fig 7. Simulation result for RER

... $r_s = 0$ (Viterbi) ___ $r_s = 0.02$

## V. CONCLUSION :

All previously proposed techniques for sync recovery or correction of sync errors require additional redundancy in the transmitted data besides more complexity included for sync purposes. To fix ideas, if we have a block of 100 bits, the additional redundancy varies from about 6 % for techniques intended only to recover sync to about 12 % for those trying to correct small number of sync errors. The required complexity in some techniques turns to be so high that such techniques become of theoretical interest only.
The algorithm presented here shows the capabilities of convolutional codes to correct sync errors as well as independent additive errors without any additional redundancy. The increase in computational complexity is not large for small values of n and k. Results are presented for the case of a channel introducing periodic insertions and independent additive errors. Simulation results indicate also that the performance is nearly the same whenever deletions are present instead of insertions.

## REFERENCES

/1/ L.D. Baumert, R.G. Mc Eliece, and H.C. Tilborg, "symbol synchronization in convolutionally coded Systems", IEEE Trans. Inform. Theory, Vol IT.25 pp. 362-366, May 1979.

/2/ M. Simon and J. Smith, "Alternate Symbol Inversion for Improved Symbol Synchronization in Convolutionally Coded System", IEEE Trans. Comm, Vol. COM. 28, N°2, Feb. 1980.

/3/ S.W. Golomb, L.R. Welch and M. Delbruck, "Construction and Properties of Comma-free Codes", Biol. Med. Dan. Vid. Selsk, Vol. 23, N° 9, pp 1-34, 1958.

/4/ S.W Golomb, B. Gordon and L.R. Welch, "Comma-free. Codes", Can. J. Math. Vol. 10 pp 202-209, 1958.

/5/ W.L. Eastman, "On the Construction of "Comma-free Codes", IEEE Trans. Inform. Theory. Vol. IT11 pp 351-356. Oct. 1964.

/6/ R.H. Barker, "Group synchronization of Binary Digital System", in Communication Theory, W. Jackson, Ed. New York : Academic Butterworth, 1953.

/7/ R.A. Scholtz "Frame Synchronization Techniques", IEEE Trans Comm. Vol. COM-28, N° 8, Aug. 1980.

/8/ E.N. Gilbert, "Synchronization of Binary Messages", IRE Trans. Inform. Theory pp. 470-477. Sept. 1960.

/9/ C. Macchi and J.F. Guilbert, "Téléinformatique", Bordas et C.N.E.T.-E.N.S.T. Paris, 1979.

/10/ P.G. Neumann, "Self-Synchronizing Sequential Coding with low Redundancy", The Bell System Tech. Journal. Vol. 50, N° 3, March 1971.

/11/ J.D. Ullman, "On the Capabilities of Codes to Correct Synchronization Errors", Princeton University, Priceton, N.J., Dept. of Elec. Engrg., Digital Sys. Lab., Tech. Rept. 44, March 1965.

/12/ _____ , "Near Optimal, Single-Synchronization - Error - Correcting Code", IEEE Trans. Inform. Theory, Vol. IT-12, N° 4, Oct. 1966.

/13/ W.W. Peterson and E.J. Weldon, "Error - Correcting Codes," The MIT Press, Second edition, 1971.

/14/ V.I. Levenshtein, "Binary codes for the correction of Insertions, Deletions, and changes of Symbols". Dokl. Akad. Nauk Ukr. SSR, Vol. 163, pp. 845-848, 1965

/15/ A.J. Viterbi, "Convolutional Codes and Their Performance in Communication System," IEEE Trans. Commun. Technol., Vol. COM. 19, pp. 751-772, Oct. 1971.

/16/ I. Iizuka, M. Kasahara and T. Namekawa, "Block Codes Capable of Correcting both Additive and Timing Errors," IEEE Trans. Inform Theory, Vol. IT. 26, N° 4, July 1980.