View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Johannesburg Institutional Repository

Good Synchronization Sequences for Permutation Codes

Thokozani Shongwe, *Student Member, IEEE*, Theo G. Swart, *Member, IEEE*, Hendrik C. Ferreira and Tran van Trung

***Abstract*—For communication schemes employing Frequency Hopping/Multiple Frequency Shift Keying modulation, we present an algorithm for finding good non-binary synchronization sequences, which are permutations, to be used with permutation codes to synchronize/resynchronize data in channels with background noise and interference(frequency jamming/fading). For the synchronization sequences, new analytical expressions for the probability of false acquisition are also given. Using simulation results, we show that our synchronization sequences perform better than some conventional non-binary synchronization sequences, in the presence of background noise and interference.**

***Index Terms*—Frame synchronization, markers, permutation codes, power line communications.**

I. INTRODUCTION

Frame synchronization for binary data is most commonly achieved by periodically inserting a synchronization sequence (*sync word*) in the data. Methods for locating the sync word in data, for both binary and M-ary communications systems over additive white Gaussian noise (AWGN) channels, have been extensively researched in [1] and [2]. Scholtz [3] presented an analytical study on the design of binary sync words in the presence of background noise (AWGN channel). It was stated in [3] that a good sync word has to be of the proper length in proportion to the frame length (that is, minimal insertion of redundancy in data), while allowing for a low probability of false acquisition in the data (P_{FAD}), and hence a high probability of acquisition (P_{A}). By acquisition we mean locating the synchronization sequence, as explained in [3].

Motivated by the research done in using permutation codes to combat noise found in the Power Line Communications (PLC) channel [4]–[7], we develop an algorithm for finding good synchronization sequences for permutation codes in this paper. The synchronization sequences are not specifically for the PLC channel, but also for channels where information is transmitted in several frequencies, such as Frequency Hopping/Multiple Frequency Shift Keying (FH/MFSK) modulation communication schemes. We only focus on channels with background noise (causing one data symbol to change into another) and frequency jamming/fading (causing symbol erasures).

In this work we present an algorithm for finding good synchronization sequences for permutation codes and present some examples of good synchronization sequences. We also suggest a way of extending the synchronization sequences to form longer sequences. The performance of our synchronization sequences is evaluated, by simulations, against available synchronization sequences in literature (sextic Barker sequences [8]). The performance measure parameters are P_{A} and P_{FAD} in the presence of background noise only, and background noise and jamming/fading.

The rest of the paper is organized as follows: next, we present the communication system model in Section II, giving a detailed description of how it affects data symbols in the channel. Section III gives new analytical expressions for the probability of false acquisition of permutation synchronization sequences on data. We present our algorithm for finding good synchronization sequences for permutation codes in Section IV, and present simulation results in Section V. Concluding remarks are in Section VI.

II. SYSTEM MODEL

We consider a digital communications system model where data is organised into frames. Each frame consists of L_D data symbols and a synchronization sequence of L_S symbols. We assume symbol synchronization has been achieved, hence we focus on frame synchronization. The frame synchronization algorithm sequentially searches for the synchronization sequence in the received frame, corrupted by channel noise. At each symbol time instant, an L_S-tuple of received symbols is compared with the synchronization sequence, and any L_S-tuple of received symbols that differs to the synchronization sequence in at most H positions is declared the synchronization sequence. H is the error tolerance of the synchronization algorithm [3].

Apart from background noise, the channel model also introduces frequency jamming or fading. We adopt a model where a symbol is marked an erasure if it is considered unreliable due to interference (frequency jamming or fading) as stated in [9].

A jammer sends energy in any one of the M frequencies used in the communication, causing communication to be impossible in that particular frequency. Alternatively, deep fading on any one of the M frequencies, is where in a particular frequency the energy is nullified causing communication to be impossible. In the rest of the paper we shall refer to either frequency jamming or fading as interference.

III. ANALYSIS OF SYNCHRONIZATION SEQUENCES FOR PERMUTATION CODES

Definition 1: A permutation code C consists of $|C|$ sequences of length M, where every sequence contains the M different integers $1, 2, \ldots, M$ as symbols. $|C|$ is the codebook cardinality.

In [3], Scholtz presented expressions for the probabilities of false acquisition on data, P_{FAD}, and true acquisition on the synchronization sequence, P_{TAM} for binary synchronization sequences.

We derive new expressions for P_{FAD} for permutation sequences used as permutation codes according to Definition 1. In Section III-A we shall discuss the case where all the $M!$ permutations of the symmetric group of permutations (S_M) are used in the permutation code C, and in Section III-B we shall look at a specific case where $C \subseteq S_M$.

A. Permutation Codes: $C = S_M$

Firstly, consider a worst case scenario where all the $M!$ permutation sequences are used in a codebook of a permutation code. The probability of getting an M-symbol permutation sequence, from the symmetric group of permutations is: $\frac{1}{M} \times \frac{1}{M-1} \times \frac{1}{M-2} \cdots \frac{1}{M-(M-1)} = \frac{1}{M!}$. If α consecutive M-symbol sequences are used to form the synchronization sequence then we have $\left(\frac{1}{M!}\right)^{\alpha}$. We come up with the following new expression for the probability of false acquisition on data for the symmetric group of permutation sequences for $\alpha = 1$:

$$P_{\text{FAD}} = \left(\frac{1}{M!}\right)\left[1 + \sum_{k=2}^{H} F_k \binom{M}{M-k}\right], \qquad (1)$$

where F_k[1] is a function which gives the number of permutation sequences that differ from the synchronization sequence in k symbols for the case when k is taken as the alphabet size for those k symbols.

The probability of synchronization sequence acquisition, P_{A} then is $P_{\text{A}} = (1 - P_{\text{FAD}})P_{\text{TAM}}$, where P_{TAM}, similar to the expression for binary synchronization sequences found in [3], is the probability of true acquisition on the marker in the presence of background noise.

Increasing α (forming longer synchronization sequences) lowers P_{FAD} at the cost of increasing redundancy. To get the full benefit of lowering P_{FAD}, the arrangement of the symbols when forming synchronization sequences of $\alpha > 1$ has to be considered, hence (1) needs a factor which will take the arrangements of symbols into account, A_F. The expressions for F_k are more complex to formulate for $\alpha > 1$. Some synchronization sequences of $\alpha > 1$ perform better than others, in terms of P_{FAD}, depending on the way the symbols are arranged in the synchronization sequence.

We present the following proposition without proof.

Proposition 1: Two adjacent permutation sequences $A = a_1 a_2 \dots a_M$, and $B = b_1 b_2 \dots b_M$, where $a_M = b_1$ will not reproduce a permutation sequence (non-repeating symbol) for up to $M - 1$ shifts to the left or right, for any M.

From Proposition 1 we form the basis of our $\alpha > 1$ permutation synchronization sequences. Sequences *A* and *B* can therefore be used together as a synchronization sequence to synchronize data codewords for a codebook which includes all the $M!$ permutation sequences (or uses some of the codewords of the code to form synchronization sequences).

We classify the $\alpha > 1$ synchronization sequences according to the idea in Proposition 1. For any M, we group synchronization sequences such that those with the same number of symbols separating closest similar symbols between the permutations, belong to the same group. With this grouping, for any M there will always be M classes, for example, for $M = 4$ and $\alpha = 2$, we can have: Class A (12344321, 12344213), Class B (12343421, 12342431), Class C (12342314, 12342341) and Class D (12341234, 21342134). This idea can be extended to $\alpha > 2$ by making sure that the next permutation is appended such that the overall synchronization sequence remains in the same class it was in before the appending of the extra permutation. We shall see later that the classifying of synchronization sequences in this way is of no significant effect when the algorithm in Section IV is employed to select synchronization sequences. From here onwards we focus on permutation codes where $C \subseteq S_M$.

B. Permutation Codes: $C \subseteq S_M$

Note that a permutation code C, is usually a subset of the symmetric group S_M of permutations, $C \subseteq S_M$. For permutation codes, where $C \subseteq S_M$, the $\alpha = 1$ synchronization sequences are simply chosen from the $M! - |C|$ permutation sequences that are not in the codebook. The key is to choose sequences that will lower P_{FAD} for that particular codebook. For this case, where $C \subseteq S_M$, we present a new upper bound on P_{FAD}, for $\alpha = 1$, as

$$P_{\text{FAD}} \leq \frac{N-1}{(NM - L_S + 1)|C|},$$

where N is the number of codewords being observed.

We were able to find some codebooks for permutation trellis codes, where $C \subseteq S_M$, in the literature [10] for $M \geq 5$ with the unused permutation sequences attaining $P_{\text{FAD}} = 0$. However, such permutation codes were not optimized to have the unused permutation sequences to attain $P_{\text{FAD}} = 0$, hence some of the permutation codes do not have any of the unused permutation sequences attaining the lower bound. To ensure that a permutation code will always have sequences that attain $P_{\text{FAD}} = 0$ among the unused permutation sequences, we developed an algorithm for finding good synchronization sequences, which are permutations, for permutation codes. We present our algorithm in the following section.

IV. GOOD SYNCHRONIZATION SEQUENCES ALGORITHM

The general idea of our algorithm is to first select permutations that will be used as synchronization sequences such that they attain $P_{\text{FAD}} = 0$ in the absence of errors, when used with permutation codes. The permutation codes are then formed with the remaining permutations after removing a particular set of permutations from the symmetric group. The set of permutations removed from the symmetric group includes permutations to be used as synchronization sequences and other permutations which may neither be used for synchronization sequences nor in the permutation code.

To facilitate the presentation of our algorithm, we first define the following sets: S_M (or $S_{\{1 \dots M\}}$), a symmetric group of permutations of M distinct elements, where $|S_M| = M!$. C, a codebook with permutation sequences of length M as codewords. S, a set of permutation sequences to be used as synchronization sequences, where each sequence attains $P_{\text{FAD}} = 0$ for codebook C. S_C, a set of permutations obtained from cyclically shifting a permutation $P \in S_M$, and S_E, a set of permutations to be excluded from C. Cardinalities of C and S_E are related by, $|C| = M! - |S_E|$.

---

[1] The definition of the function can be found in Appendix A at www.ujtrg.co.za/docs/good_sync_words_appendix.pdf

We employ the following procedure to find members of S, that is, synchronization sequences attaining $P_{\text{FAD}} = 0$ in the absence of any type of errors.

The Algorithm:

Set $S_E = \emptyset$.

1) Choose any permutation $P = P_1 P_2 \dots P_M$ from S_M, where $P_i \in \{1, \dots, M\}$.
2) Set $S_C^i = P_{i+1} \dots P_{M-1} P_M P_1 \dots P_i$, where $i = 0, \dots, M-1$.
3) For M odd, set $M = 2m+1$, and for M even, set $M = 2m$, where m is a positive integer.
   a) For $i = 0, \dots, m$ form a set of T_i permutations from S_C^i as follows: $T_i = \{P_{i+1} \dots P_M P'_1 \dots P'_i\}$, where $P'_1 \dots P'_i \in S_{\{P_1 \dots P_i\}}$.
   b) For $j = m+1, \dots, M-1$ form a set of T_j permutations from S_C^j as follows: $T_j = \left\{P_{j+1} \dots P_M P'_1 \dots P'_j\right\}$, where $P'_{j+1} \dots P'_M \in S_{\{P_{j+1} \dots P_M\}}$.
4) Set $S_E = T_i \cup \dots \cup T_{M-1}$.
5) Include P in S.

To add more permutations in S choose a new permutation in S_M, P' such that $P' \notin S_E$ and $P' \neq P$, and perform steps 1–5.

Example 1: 1) Let $P = 23154$, and hence $M = 5$.
2) $S_C^0 = 23154$, $S_C^1 = 31542$, $S_C^2 = 15423$, $S_C^3 = 54231$, $S_C^4 = 42315$.
3) $T_0 = \{23154\}$, $T_1 = \{31542\}$, $T_2 = \{15423; 15432\}$, $T_3 = \{54231; 45231\}$, $T_4 = \{42315\}$.
4) $S_E = \{23154; 31542; 15423; 15432; 54231; 45231; 42315\}$.
5) $S = \{23154\}$.

For any M, the cardinality of S_E in relation to S is given by the following expression: $|S_E| = |S|\sigma$, where the expression for σ is different for odd and even M, and will be called σ_{odd} and σ_{even}, respectively. $\sigma_{\text{odd}} = 1 + 2\sum_{i=1}^{m} i!$, for odd M, and $\sigma_{\text{even}} = 1 + m! + 2\sum_{i=1}^{m-1} i!$, for even M.

We can further minimize the number of permutations to be included into S_E, with each choice of P, by restricting the next permutations to be included in S, after the first P, such that they are related. An easy case of such a relationship between the members of S is permutations that are cyclic shifts of each other. It can be seen from the proof[2] of our algorithm that if the permutations are chosen from the same cyclic group, the next choice of P, after the first one, reduces the elements of S_E by M, compared to when the next permutation was not related to the previous one. In this case the cardinality of S_E, with each permutation included in S, then becomes: $|S_E| = M + |S|\left[1 + 2\sum_{i=1}^{m}(i! - 1)\right]$, for odd M. $|S_E| = M + |S|\left[1 + (m! - 1) + 2\sum_{i=1}^{m-1}(i! - 1)\right]$, for even M.

There are several advantages of having more than one synchronization sequence attaining the lower bound on P_{FAD}, one of which is immediately seen when forming synchronization sequences for $\alpha > 1$. Another advantage, which is beyond the scope of this paper, is to use different synchronization sequences between data symbols such that they improve synchronization performance.

V. Results and Interpretation

The performance of possible synchronization sequences is compared in terms of P_{A} and P_{FAD} in the presence of background noise and interference. Fig. 1 shows performance results in terms of P_{FAD}, for four $M = 4$ and $\alpha = 2$ synchronization sequences grouped into four classes, where $L_D = 400$ and background noise probability set at 0.01.

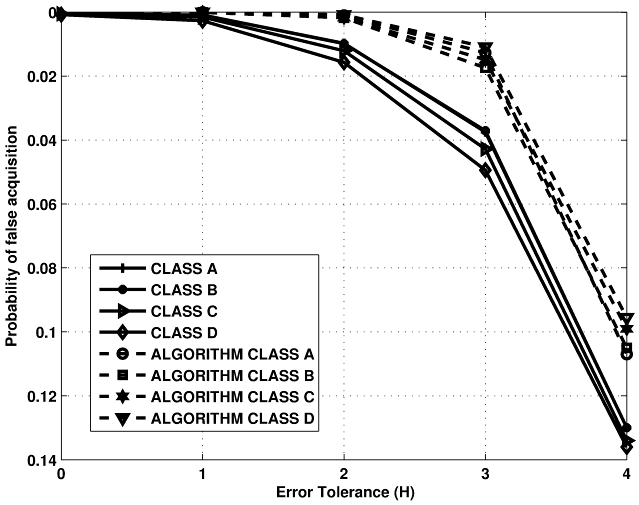

Fig. 1. $M = 4$, $\alpha = 2$ ($L_S = 8$) Synchronization sequences compared for probability of false acquisition, with symbol error rate set at 0.01 and various error tolerance values.

Synchronization sequences class A (32411432), class B (14323241), class C (14321324) and class D (41324132), formed from permutations selected by our algorithm (in Section IV), are used with the symmetric group, S_4 and codebook C_1[3].

For reference, we name the synchronization sequences when used with S_4 and C_1, CLASS A–D and ALGORITHM CLASS A–D, respectively. As seen in Fig. 1, ALGORITHM CLASS A–D sequences have comparable performance with each other, but outperform CLASS A–D. CLASS A and CLASS B have the best performance, followed by CLASS C and finally, CLASS D with the worst performance.

If one is constrained to use some of the permutations, from the permutation code, to form synchronization sequences, we recommend the use of Class A or B sequences.

---

[2] The proof can be found in Appendix B at www.ujtrg.co.za/docs/good_sync_words_appendix.pdf

[3] The codebook can be found in Appendix C at www.ujtrg.co.za/docs/good_sync_words_appendix.pdf

The following possible synchronization sequences are compared for P_{A}: SW1 (1122552415), SW2 (1113311415), SW3 (1122552515), SW4 (1523434152) and SW5 (2513434152), where SW1, SW2 and SW3 are sextic Barker sequences and SW4 and SW5 are our $\alpha = 2$, $M = 5$ sequences. In the simulations, we choose $H = 3$ because it produces optimal results for P_{A} for both our and sextic Barker sequences, and L_D is set at 500. Symbols were randomly selected from codebook C_2[4].

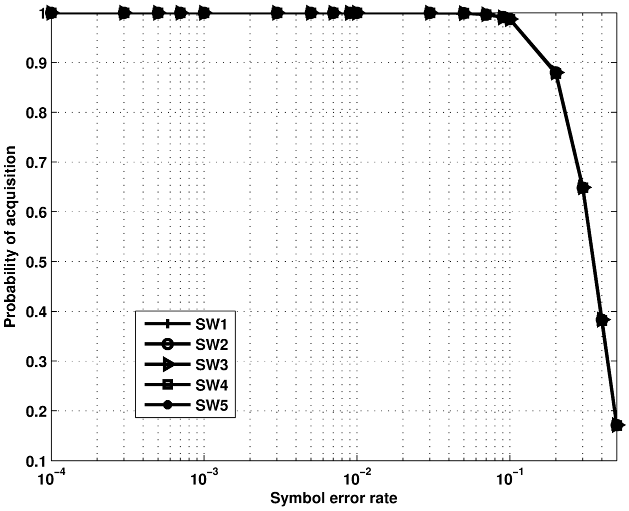

Fig. 2. Permutation synchronization sequences and sextic Barker sequences for $M = 5$ and $\alpha = 2$ ($L_S = 10$) compared in terms of probability of acquisition against symbol error rate.

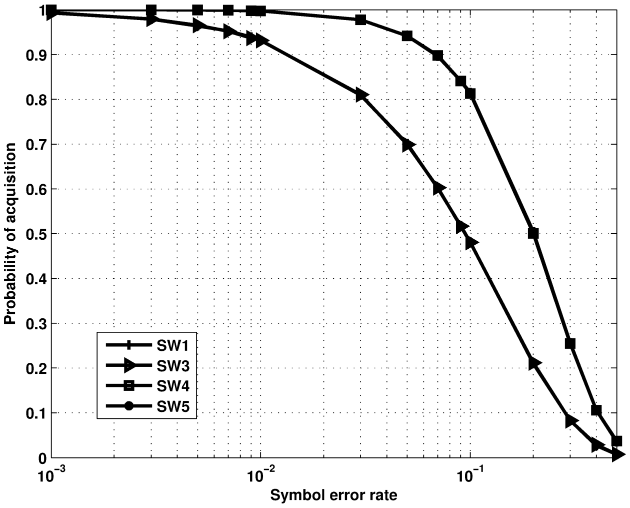

Fig. 3. Permutation synchronization sequences and sextic Barker sequences for $M = 5$ and $\alpha = 2$ ($L_S = 10$) compared in terms of probability of acquisition against symbol error rate in the presence of interference on frequency 1.

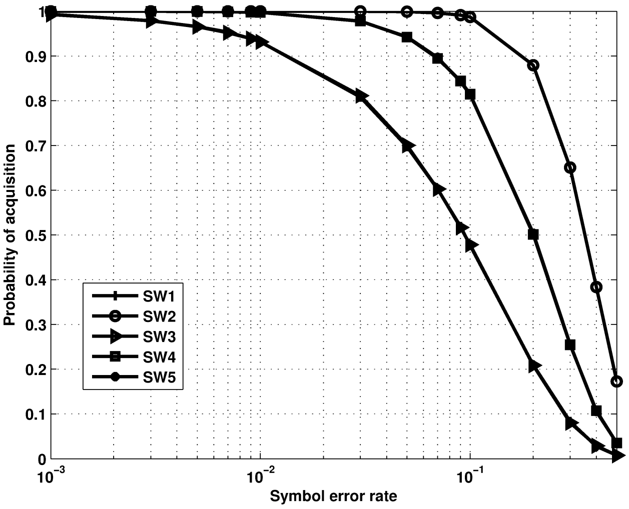

Fig. 4. Comparison of permutation synchronization sequences and sextic Barker sequences for $M = 5$ and $\alpha = 2$ ($L_S = 10$) compared in terms of probability of acquisition against symbol error rate in the presence of interference on frequency 2.

Fig. 2 shows the P_{A} performance comparison of our synchronization sequences against sextic Barker sequences in the presence of background noise only. In Figs. 3 and 4 our synchronization sequences are compared with the sextic Barker sequences in the presence of background noise and interference, where we show results only when there is interference at frequencies 1 and 2, corresponding to Fig. 3 and 4 respectively. In Fig. 3 SW2, which is a sextic Barker sequence, is not shown in the graph because it completely fails acquisition probability performance at $H = 3$. Our two synchronization sequences SW4 and SW5 have the best performance than all the sextic Barker sequences when frequency 1 is jammed.

Fig. 4 shows the results of the synchronization sequences when interference is on frequency 2: SW2 has the best performance because it does not have symbol 2 among its symbols, hence interference on frequency 2 has very little or no effect on SW2; our synchronization sequences SW4 and SW5 are the next best performing sequences after SW2, with the worst performing sequences being the sextic Barker sequences SW1 and SW3.

From Figs. 3 and 4 it can be seen that our synchronization sequences have a constantly good performance in the presence of interference. We say it is constantly good performance because the performance is the same for any one frequency with interference, whereas the sextic Barker sequences' performance in the presence of interference varies significantly depending on the jammed frequency. Our synchronization sequences are therefore more robust in the presence of interference compared to the sextic Barker sequences. Interference on other frequencies shows the same behaviour, leading to the same conclusion that our synchronization sequences show a constant performance over the sextic Barker sequences.

---

[4] The codebook can be found in Appendix C at www.ujtrg.co.za/docs/good_sync_words_appendix.pdf

VI. CONCLUSION

We presented a method for finding good synchronization sequences that can be used with permutation codes, and constructed synchronization sequences that are robust in the presence of background noise and interference.

A new expression for the probability of false acquisition on data was found for the symmetric group of permutations, and an upper bound for the probability of false acquisition on data for permutation codebooks, where $C \subseteq S_M$, in the absence of errors was established.

We showed that synchronization sequences for $\alpha > 1$ can be grouped into classes according to performance for $C = S_M$, and further presented results indicating that synchronization by our algorithm, where $C \subseteq S_M$, show good performance.

Synchronization sequences for $\alpha = 2$ and $M = 5$, that are permutations sequences for $C \subseteq S_5$, were formulated and performed better than sextic Barker sequences in the presence of background noise and interference, and with comparable performance in the presence of background noise only.

REFERENCES

[1] J. L. Massey, "Optimum frame synchronization," *IEEE Transactions on Communications*, vol. 20, no. 4, pp. 115–119, Apr. 1972.

[2] G. L. Lui and H. H. Tan, "Frame synchronization for gaussian channels," *IEEE Transactions on Communications*, vol. 35, no. 8, pp. 818–829, Aug. 1987.

[3] R. Scholtz, "Frame synchronization techniques," *IEEE Transactions on Communications*, vol. 28, no. 8, pp. 1782–1789, Aug. 1980.

[4] A. J. H. Vinck, "Coded modulation for powerline communications," *AEU International Journal of Electronics and Communications*, vol. 54, no. 1, pp. 45–49, Jan. 2000.

[5] H. C. Ferreira and A. J. H. Vinck, "Interference cancellation with permutation trellis codes," in *Proceedings of the IEEE Vehicular Technology Conference*, Boston, MA, USA, Sept. 24–28, 2000, pp. 2401–2407.

[6] H. C. Ferreira, A. J. H. Vinck, T. G. Swart, and I. de Beer, "Permutation trellis codes," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1782–1789, Nov. 2005.

[7] H. C. Ferreira, L. Lampe, J. Newbury and T. G. Swart, Eds., *Power Line Communications: Theory and Applications for Narrowband and Broadband Communications Over Power Lines*. Chichester, England: John Wiley and Sons, 2010, Ch. 5.

[8] F. Pingzhi and M. Darnell, *Sequence design for communications applications*. England: Taylor and Francis, Inc., 1996.

[9] P. F. Driessen, "Performance of frame synchronization in packet transmission using bit erasure information," *IEEE Transactions on Communications*, vol. 39, no. 4, pp. 567–573, Apr. 1991.

[10] T. G. Swart, I. de Beer, and H. C. Ferreira, "On the distance optimality of permutation mappings," in *Proceedings of the IEEE International Symposium on Information Theory*, Adelaide, Australia, Sept. 4–9, 2005, pp. 1068–1072.