Draft Paper
To be submitted to: IEEE Symposium on Wireless and Microwave Circuits and Systems (WMCS), 2015

A Method for Intercepting and Demodulating Slow Frequency Hopping DPSK Signals

Scott Koziol*, Steve F. Russell†
* Electrical and Computer Engineering Department, Baylor University, Waco, Texas 76798
† Department of Electrical and Computer Engineering (Emeritus), Iowa State University, Ames, Iowa 50011
Email:scott_koziol@baylor.edu, sfr@iastate.edu

***Abstract*—This paper presents a non-coherent wireless communication receiver design for intercepting and demodulating frequency hopping spread spectrum (FH-SS) signals. This interception receiver is based on phase modulation to amplitude modulation conversion (PM to AM), and is specifically designed for demodulating a slow frequency-hopped signal that has differentially encoded phase shift keying (DPSK) modulation. The intercept receiver uses a bank of bandpass filters at the front end. Matlab simulations show that if a signal has a signal to noise ratio (SNR) greater than approximately 30 dB (where the bandwidth (BW) of each of the intercept filters in the simulation is five times the message bit frequency) , the intercept receiver can demodulate it by using an approach based on PM to AM conversion and adaptive pattern recognition. The performance of the receiver is characterized with simulations and analytical calculations.**

***Index Terms*—Frequency Hopping Spread Spectrum, noncoherent detection, digitally modulated signals, DBPSK**

I. Introduction and Background

Frequency hopping spread spectrum (FH-SS) is a type of Low Probability of Intercept communications [1], [2]. The messages are difficult to intercept because the intercept receiver must first synchronize to the pseudorandom hopping sequence before demodulating the data [3]. Circular Equivalent Vulnerable Distance (CEVR) methods can be used to quantify the operational level of LPI systems [4]. The CEVR describes a radius area within which the transmitter is vulnerable to interception. A numerical example using this method for FH-DPSK is found in [4]. In this paper, we describe a method for intercepting FH-SS signals when differential phase shift keying (DPSK) is the modulation scheme. Performance of slow frequency hopping multiple-access networks using DPSK is found in [5]. The system block diagram for the model of a FH-DPSK communication system is found in Fig 1.

Much of the unclassified literature related to FH-SS interception applies to the detection, not demodulation, of spread spectrum signals [6]. This focus on detection and not demodulation is perhaps because an interception method's performance is typically judged by its ability to determine if a transmission has occurred and not whether the message can be decoded or the source located. Coherent and noncoherent receiver structures for interception of FH-SS signals are presented in [7]. An optimal interceptor for FH-DPSK based on the maximum likelihood principle is found in [8]. The vulnerability of BPSK Direct Sequence (DS) spread spectrum waveforms to code clock extraction from bandlimitation induced envelope fluctuations has been described in [2].

This paper presents a FH-DPSK demodulator design that can intercept and demodulate the received signal with no synchronization to the hopping sequence or data sequence. The front end of our intercept receiver is similar to a filter-bank combiner (FBC) which is commonly used to intercept frequency-hopped signals [4]. Unlike [4], however, our method does not use an integrator after the filtering and square-law operation. The receiver uses a bank of filters to induce phase modulation to amplitude modulation (PM to AM) conversion [2] and then makes decisions about when bit changes have occurred based on a novel algorithm that combines the information from a bank of narrowband filters. A usual narrowband assumption is made (the information bandwidth is much smaller than the hopping bandwidth) [9]. The process of combining filter bank information and interpreting it permits decoding of the bits and the interpretation is done with pattern recognition (PR). In our intercept receiver, the main parameters that are of interest are: detection signal to noise ratio (SNR), total message time or hop time, and the spread spectrum bandwidth, (i.e. span or hop frequency spacing). The assumptions of this research and optimum signal characteristics for this receiver are as follows:

1) Individual hopping frequencies are unknown
2) Dwell times and bit epochs are unknown
3) There is no external synchronization available
4) The hopping span is estimated [10]
5) hopping is applied to the signal at bit transitions
6) Each bit is composed of an integer number of carrier waves
7) modulation is DPSK
8) The signal is slow frequency hopped such that there are many data bits per hop [11]
9) The first bit following a carrier frequency hop does not contain data (delay assumption)

Section II discusses the PM to AM conversion, Section III discusses how the PM top AM conversion is used to intercept the FH-DPSK signals, Section IV describes the performance and Section V is a closing summary

II. PM to AM Conversion

It is well known that a phase-modulated signal is converted into a signal with partial amplitude modulation when it is

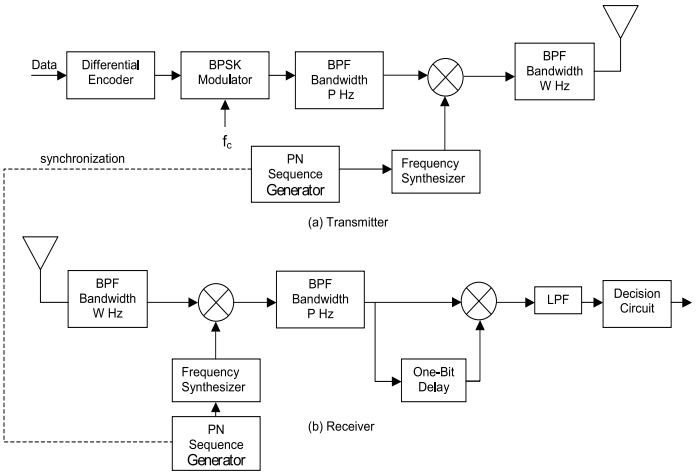

Fig. 1: Block diagram of a conventional synchronous FH-DPSK transmitter and receiver system

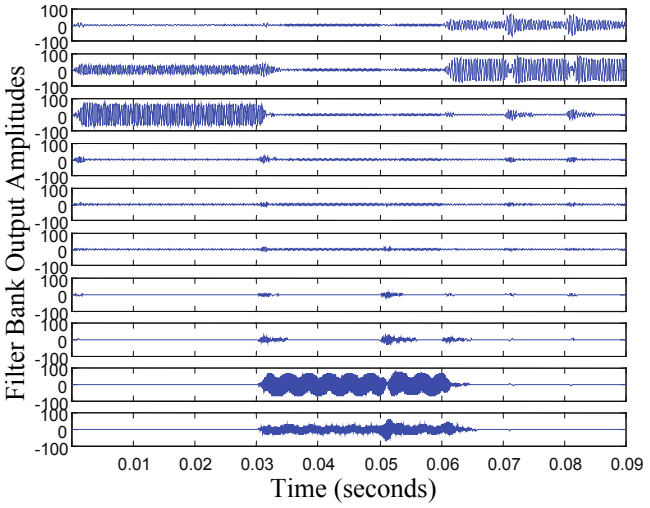

Fig. 2: FH-DPSK Interception Receiver filter bank outputs; BW = 500 Hz for each filter

passed through a narrow band filter or channel [12]. For BPSK, the filter output signal is the impulse response convolved with the sequence of BPSK waveforms as given by (1).

$$y(t) = \sum_{p=1}^{N} \left(\int_{-\infty}^{\infty} s_{i_p}(\tau) h_p(t-\tau) d\tau \right) \tag{1}$$

Convolution inside the summation causes a symbol to bleed into the next symbol's time epoch, creating inter-symbol interference (ISI) which is manifested as envelope fluctuations. In this case, ISI is a good thing because we use it to detect the bit transitions. The analysis shows that destructive ISI interference only occurs when bit changes occur. If a bit change does not occur the interference is constructive and the ISI actually compensates for the slow ramp-up of the output of the envelope out of a linear filter. This compensation has the effect of canceling out the dip in the envelope.

III. USING PM TO AM CONVERSION FOR INTERCEPTING SLOW FH DPSK SIGNALS

The basic idea is that the received RF signal is passed through a parallel bank of narrowband filters to create PM to AM conversion. The resulting waveforms are processed in a multi-step algorithm. The number of filters in the bank is governed by the span of the FH signal and the bandwidth of the filters. As more filters are required, a commensurate increase in hardware and software processing is needed. The PR decoding algorithm's three primary operations: Data Representation, Feature Vector selection, and Classification [13] and their respective steps will now be discussed along with the simulation results for a particular scenario.

The simulation example presented in this paper has the following parameters: The signal is composed of 9 bits with 3 bits per hop. The data vector: [011110000] was encoded and transmitted at a rate of 100 bps with hopping frequencies of 2700, 5900, and 2200 Hz. The received SNR is 45 dB and the detector has a bank of ten filters (each with a passband BW of 500Hz and center frequencies at 1800, 2300, 2800, 3300, 3800, 4300, 4800, 5300, 5800, and 6300). The output of the PM to AM inducing filter bank is shown in Fig 2.

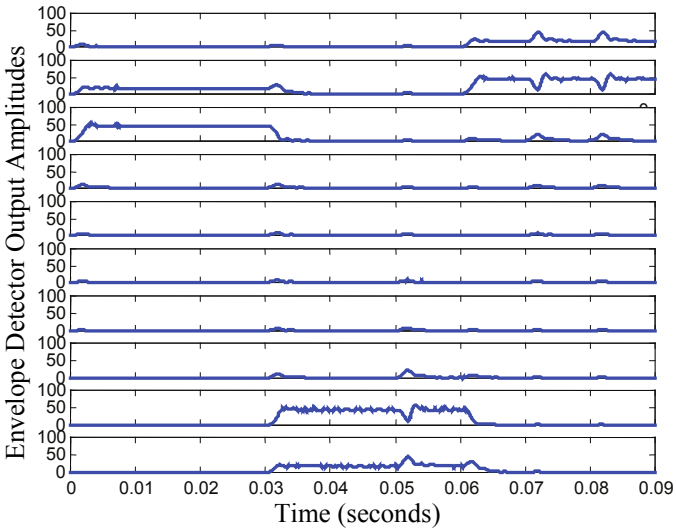

Fig. 3: Envelope data to be used as input to the PR system

A. Data Representation

1) *Step 1: Format filter bank output*

The front-end of the intercept receiver is a bank of narrowband filters. The output of each of these filters is processed using a Square-law-detector to form the envelope of each filter output in the filter bank. This method squares the signal, low pass filters the square, and then finds the square root. Fig 3 shows the envelope functions of the waveforms of Fig 2.

B. Feature Vector Selection

For the feature vector, two PR class memberships are identified: 1) a bit change and/or frequency hopping has occurred, or 2) a bit change and/or frequency hopping has not occurred. This combines the analog envelope information from each of

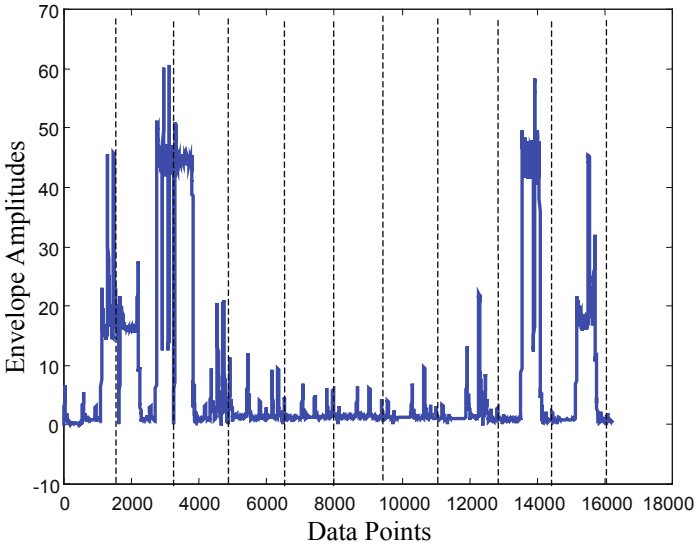

Fig. 4: Concatenated Envelopes of Fig 3

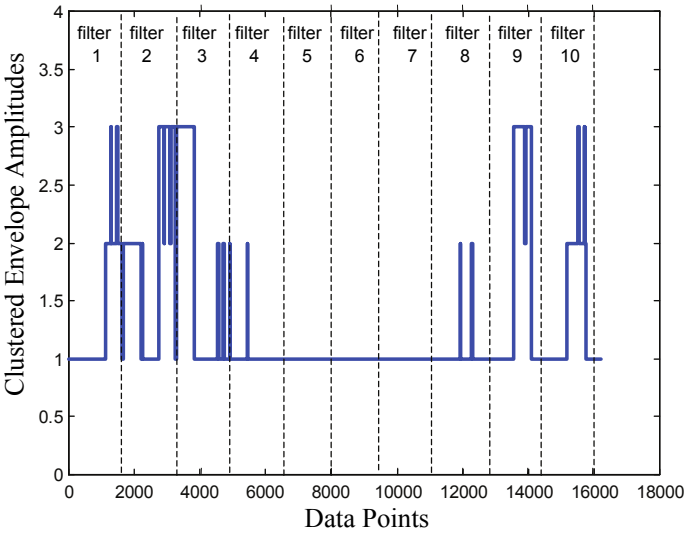

Fig. 5: Clustered envelope from Fig 4

the filters in the filter bank and produces a digital data vector that combines distinctive features that identify bit change and frequency hopping boundaries. In simple terms, it looks at each envelope and decides if there is a dip large enough to be considered an amplitude modulation, and hence, a bit change. Each filter in the filter bank will create a vector that contains an envelope value for each time sample. Hypothesis testing is used for making the detection decision. The detector must have a criterion for making a decision on each point so the K-means clustering algorithm was chosen [14].

2) *Step 2: Concatenate the envelopes from the N filters to produce a single vector*

The envelopes are first concatenated into a single vector so that all of the filters are clustered with the same resolution. Fig 4 is a graph of the envelopes in Fig 3 concatenated together. The dashed lines indicate the approximate beginning and ending of the individual filter outputs.

3) *Step 3: Cluster vector into 3 clusters*

The concatenated envelope function is clustered using K=3. This is plotted in Fig 5.

4) *Step 4: Separate the clustered vector into the original N vectors*

After clustering, the concatenated vector is separated back into N vectors. These vectors will represent the clustered versions of the N filter envelopes. Fig 6 is a plot of the separated clustered vector that was derived from the data in Fig 5.

5) *Step 5: Sum the N clustered vectors in parallel*

This method of combining the filter bank information combined with the next step is one of the novelties of this research. Fig 7 shows the results of summing the N filter banks in parallel. Typically, interception receivers using a bank of filters would, for a given time sample, select the largest signal from all of the filters and use that for further processing [11].The novelty of this method is that all of the filter data from the bank is used.

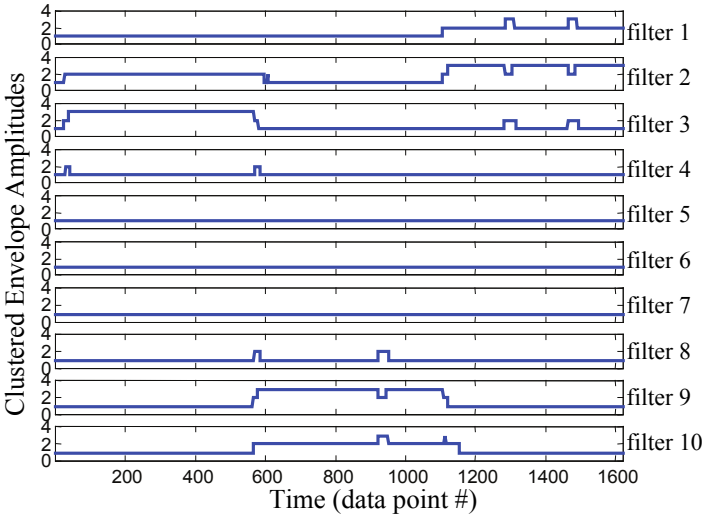

Fig. 6: Separating the clustered vector from Fig 5

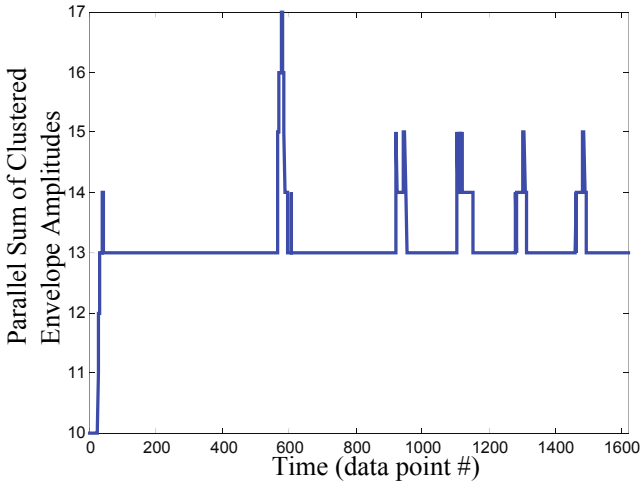

Fig. 7: Summing the N clustered filter banks from Fig 6 in parallel

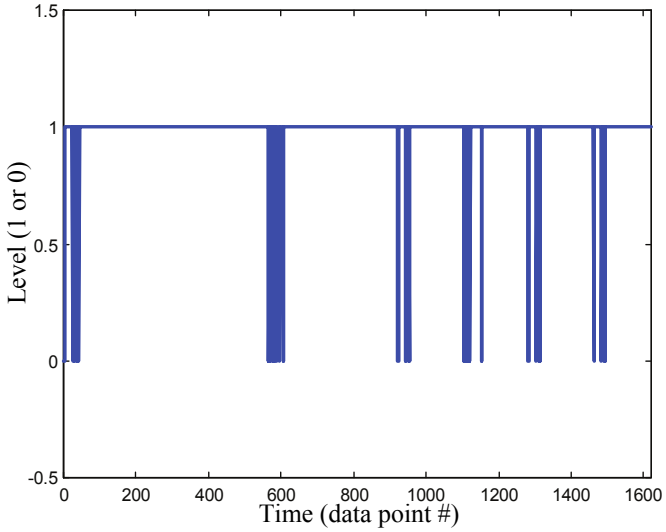

Fig. 8: Mapping the summed vector of Fig 7 into two values

6) *Step 6: Map the summed vector to either a zero or a one*

This is the step that identifies the features. The output is a vector that is the same length as the input vector. The first point is arbitrarily assigned to be a one. With the exception of the first point, the algorithm compares every data point in the vector to its most previous data point. If the two values differ, a zero is assigned to that point. If the two values are the same, a one is assigned to that point. Essentially, this is a logical AND operation. Fig 8 shows the result of performing Step 6 using the data in Fig 7.

C. Feature Classification

The features in this system now consist of zeros and ones and close inspection of Fig 8 reveals that the zeros are grouped into clusters. Each of these clusters represents either a bit transition or a frequency hop. The time between clusters represents some integer number of epochs. The classifier model in this algorithm has three main functions: 1) identify clusters of zeros, 2) estimate the epoch length, 3) decode the bits.

In order to decode the bits, the algorithm must have a good estimate of the epoch length, and in order to have a good estimate of the epoch length, the algorithm must be able to accurately identify clusters of zeros in the pattern. The K-means algorithm is very useful for clustering, but one must have a priori knowledge of the number of clusters. In this case, the interceptor does not know how many bits or hops are in the frame so the K-means algorithm is not especially helpful here. Another method of clustering the zeros was therefore developed for the classifier model.

The authors have developed an unsupervised learning algorithm that uses a recursive method to converge upon the epoch length. The present algorithm is not completely unsupervised in the strictest sense because currently the intercept receiver operator is responsible for changing variables if the epoch does not converge correctly. The optimization of the PR system could be a future area of research.

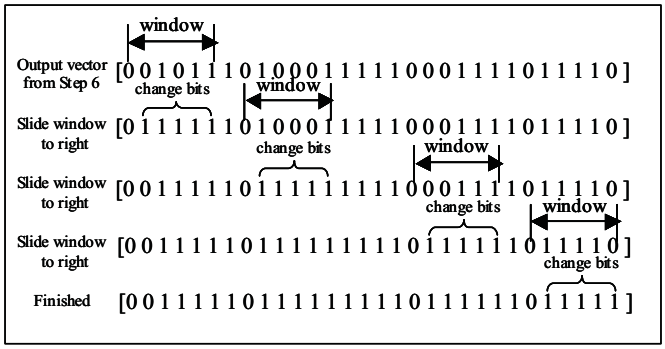

Fig. 9: Step 7 illustrated

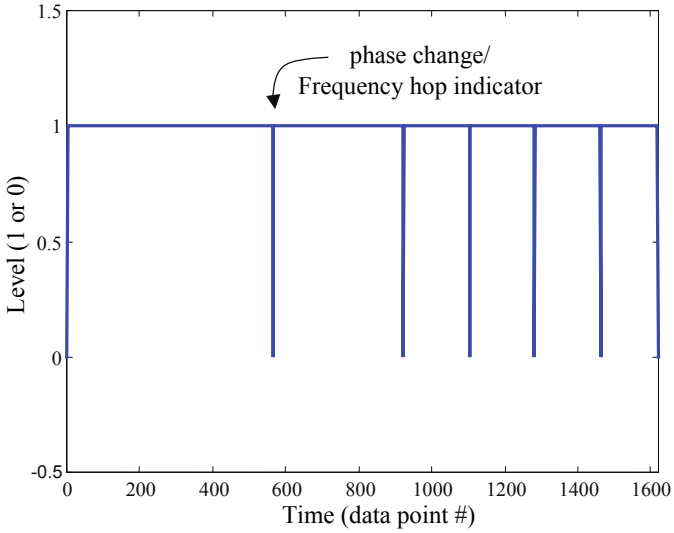

Fig. 10: Clustering the zeros of Fig 8

7) *Step 7: Identify clusters of zeros with an adaptive sliding window and represent each cluster with a single zero*

Clusters of zeros in Fig 8 are identified using a sliding window that is a scaled length of the current bit epoch estimate (2) (see Step 8 for bit epoch estimate).

$$window\_length = scale\_factor \times current\_epoch\_estimate \quad (2)$$

The window is slid across the data to the right and stops when the left edge of the window is on a zero. Then all bits within the window except for the left most zero bit are converted to ones. The window continues sliding until its left edge finds another zero and the process repeats. Step 7 is illustrated in Fig 9. Fig 10 illustrates applying Step 7 to Fig 8.

8) *Step 8: Using the distance between zeros, make an estimate of the epoch time and calculate a running average of the epoch*

For this decoding algorithm to work properly, it is imperative that an accurate estimate of the epoch be derived. Since the epoch is unknown, the PR system must learn it using input to the system. This part of the decoding algorithm was not optimized although it converged in the simulations. Convergence is a topic for future research.

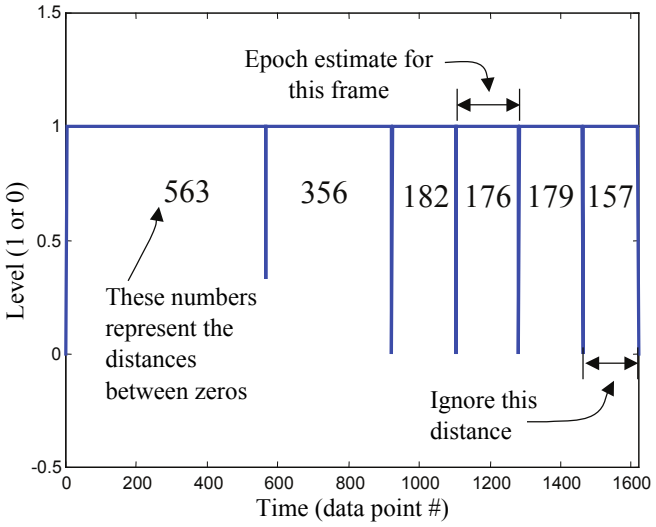

Fig. 11: Making a new estimate of the epoch

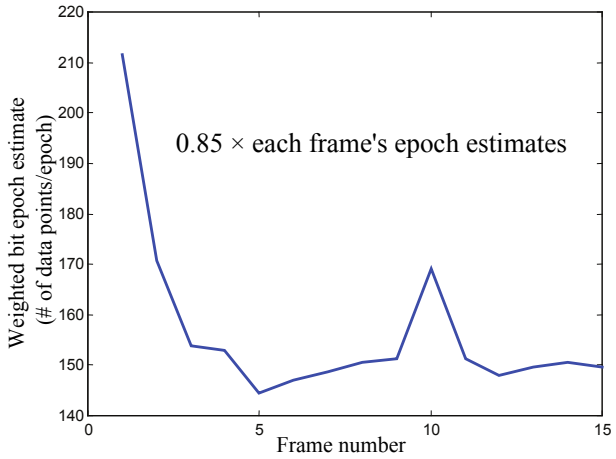

Fig. 13: Example epoch estimates from the frames (Point A in Fig 12)

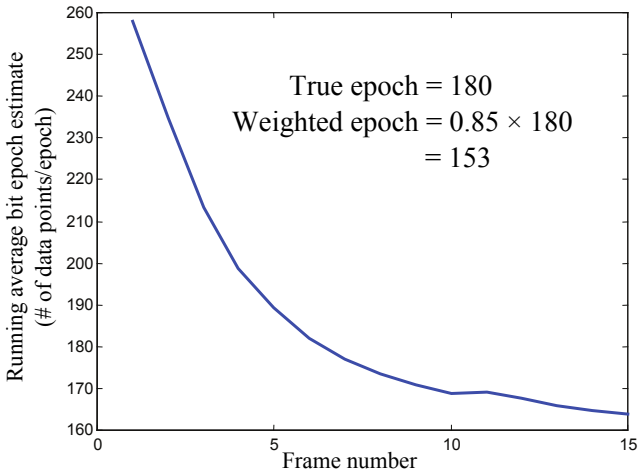

Fig. 14: Convergence upon the weighted true epoch (Point B in Fig 12)

The initial estimate of the epoch is chosen as follows: The algorithm first finds the longest distance between phase change/frequency hop indicators (which will be referred to as *zeros*). It then divides this by a number, C, and uses this as the initial epoch estimate. Choosing C=2 has produced good results. This epoch estimate is used to cluster the zeros for the first frame (see Step 7). After clustering the zeros, the algorithm uses the distance between zeros to make a second estimate of the epoch. Specifically, the algorithm finds the distance between these zeros and chooses the minimum distance as the new estimate of the epoch. It does however ignore the distance between the last zero and the end of the vector. Fig 11 illustrates the frames epoch estimation process. Ideally, one would want to look at the distances between zeros and choose an epoch estimate that is an integer number of all of the distances.

A weighted version of each frame's epoch estimate contributes to a running average, and this running average is used to cluster the zeros for the next frame's Step 7. A block diagram of the proof of concept convergence algorithm is found in Fig 12.

In the example problem, Fig 11, the weighted epoch estimate from this vector is 150. This was found by multiplying 0.8 (the weight value) by the frame epoch estimate, 176. The running average of the epoch estimate is 164. This is the result of averaging 15 frame's epoch estimates. Fig 13 shows the individual epoch estimates from each of the frames in the example, and Fig 14 shows how the running average estimate of the epoch converged.

9) *Step 9: Using the running average estimate of the epoch, estimate the number of bits occurring between each zero*

A vector is created that gives the distances between zeros. These distances are divided by the running average of the epoch estimate. This will indicate how many epochs occur between zeros. The example problem illustrates this operation in Fig 15. The running average of the epoch estimate will typically not divide evenly into the distances between zeros, so the numbers are rounded. Rounding these numbers to the closest integer results in an output vector of:

Step 9 output vector = [3 2 1 1 1 1]

10) *Step 10: Create a differential sequence from the estimated number of bits data and the knowledge of where the zeros occur*

Now it is shown why this decoder is made to work with differentially encoded bits. The algorithm uses the vector from Step 9 to recreate the transmitted differential sequence. If the vector from Step 9 indicates three bits before a phase change/frequency hop indicator then the first three bits of the recreated differential sequence are assigned to be either all ones or all zeros. Although this mapping is arbitrary, assume for further discussion that ones were assigned. If the vector from Step 9 has a two for its next number, the next two bits in the recreated differential sequence are assigned to be zeros (the complement of the previous mapped bits). If the vector from Step 9 has a one for its next number, the next bit in the recreated differential sequence is assigned to be a one (the complement of the previous mapped bits). The vector from Step 9 is thus used to recreate the differential sequence. Table I recreates the differential sequence of the example problem.

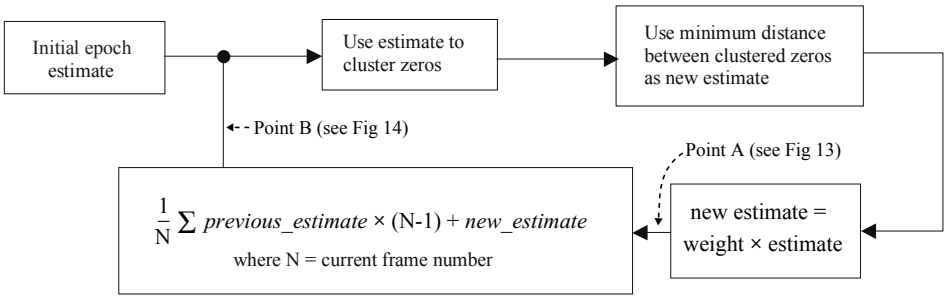

Fig. 12: Feedback system to converge on the true epoch

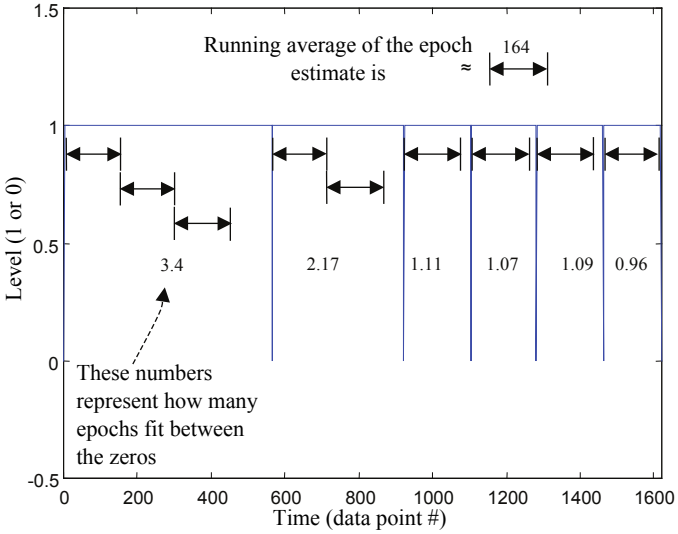

Fig. 15: Estimating the number of bits between each zero

TABLE I: Assigning differential bits based on Step 9

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
Bits/Bin	3	2	1	1	1	1
Differential Bits	1 1 1	0 0	1	0	1	0

11) *Step 11: Decode the differential sequence*

The same complemented XOR operation that was used to create the original differential sequence is used to decode the recreated differential sequence (3).

$$recreated\_differential\_sequence = [x_1, x_2, x_3, x_4..., x_Q] \tag{3}$$

The decoded output bits are (4):

$$decoded \quad sequence = \overline{x_1 \oplus x_2}, \overline{x_2 \oplus x_3}, \overline{x_3 \oplus x_4}, \overline{x_4 \oplus x_5}..., \overline{x_{Q-1} \oplus x_Q} \tag{4}$$

This decoding process by default will assign a zero to the first bit following a phase change/frequency hop indicator. Table II shows the results of decoding the differentially encoded sequence from Table I. The *X* in the decoded bits in Table II will be addressed in Step 12.

12) *Step 12: Add a zero to the front of each frame's decoded sequence*

TABLE II: Decoding the differential bits from Table I

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
Differential Bits	1 1 1	0 0	1	0	1	0
Decoded Bits	x 1 1	0 1	0	0	0	0

TABLE III: Decoded bits compared to the original message bits

	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
Decoded Bits	0 1 1	0 1	0	0	0	0
Original Message Bits	0 1 1	1 1	0	0	0	0

First, assume that the PR system has a training algorithm that is able to synchronize itself so that a cluster of zeros occurs in the beginning of each frame. This means that the beginning of each frame is a phase change/ frequency hop indicator. This would be reflected in the algorithm in that the output of Step 7 will have a zero for the first point of its output vector. The decoded bits are compared to the original sent bits in Table III.

The one error bit in this example is a bit that follows a frequency hop. Assuming that the transmission of a 1 or a 0 is equally likely, there is a 50 percent probability that the first bit following a frequency hop will be in error. This is because the PM to AM conversion process in this detector only works when adjacent bits have the same frequency. This information is why specifications 8 and 9 were given in the introduction. The example used to illustrate the bit detection algorithm depicted the interception of the final 9 bits of a 135 bit sequence. The algorithm broke the 135 bit sequence into 15 frames of 9 bits/ frame. The algorithm required two frames to train itself (18 bits). After two frames of training, the next 13 frames (117 bits) were decoded with the following results: For the post-training frames, there were 26 error bits out of 117 total valid bits. The probability of error is 22%.

IV. Demodulator Performance

Our SNR definition was the standard ratio of signal power to noise power in each filter. The noise is AWGN so each filter has the same noise power output. We used relative-frequency as the metric for correct and incorrect bits. Simulations were performed using Matlab. Estimates of bit error probability, P_e,

TABLE IV: Performance Summary

SNR (db)	P_e (%)	N_{total}	N_{wrong}	$N_{training}$	$N_{freqhops}$	$N_{\frac{bits}{hop}}$	$P_{calculated}$ (%)	Unpredicable error $\frac{100(P_{calculated}-P_e)}{P_{calculated}}$ (%)
20	60	984	593	18	328	3	17	260
30	17	906	151	96	302	3	17	1.8
35	18	948	174	54	316	3	17	10
40	17	918	155	84	306	3	17	1.3
45	18	966	171	36	322	3	17	6.2
45	2	900	22	120	45	20	2.5	2.2

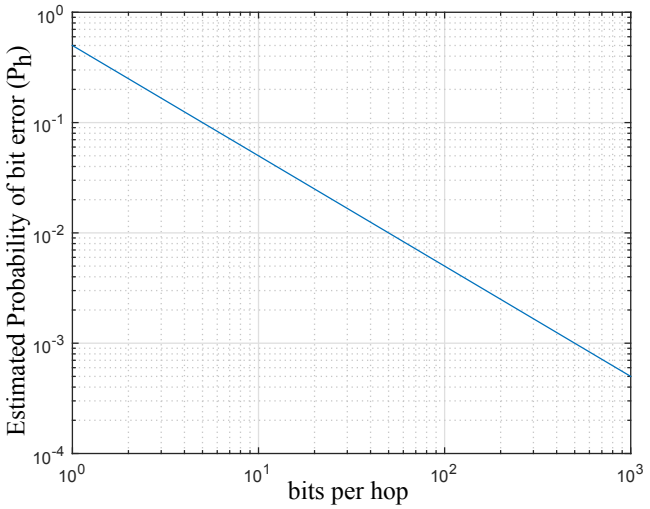

Fig. 16: Predicted probability of bit error when sending 1000 bits

were obtained for various SNR values. The performance results of the intercept receiver are found in Fig IV. When calculating P_e, the bits in the initial training frames were not used in any calculations. Once the system was trained, all subsequent bits were used in performance calculations. If the receivers bit detection algorithm decodes a frame and concludes that there are *x* bits, but there were actually *y* bits, the algorithm assumes that all *y* bits in that frame are in error.

As mentioned in Step 12, there is a 50 percent probability that the first bit following a frequency hop is incorrect. Knowing this and the number of hops that occurred in each simulation case, one can make an estimate of the probability of bit error due to frequency hops (3).

$$P_{calculated} = \frac{N_{freq\_hops}}{N_{Total}} \times \frac{1}{2} \tag{5}$$

The last row of Table IV shows that as the number of bits per hop increases, the probability of bit error decreases. According to the last column in the chart, with adequate SNR the unpredictable errors due to the algorithm are small (less than 11 percent). Fig 16 shows how the performance should improve if the number of bits per hop increases. This figure was calculated using the equation for $P_{calculated}$ from (5). As long as there is a high SNR, the algorithm is able to obtain a good estimate of the epoch, and the unpredictable error stays small, (5) should be an accurate prediction of the performance of the intercept receiver.

Table IV also shows that as the SNR falls below approximately 30 dB the probability of bit error dramatically increases. According to the limited number of test cases, there appears to be a SNR threshold below which this receiver is incapable of intercepting and decoding the message bits. Above this threshold, the receiver is extremely accurate (given the assumptions).

When the SNR is low, the noise effectively distorts the envelope so much that the K-means clustering algorithm clusters many adjacent points into different classes. This adds more zeros in the output of Step 6. It is then very difficult for Step 7 to cluster these zeros into the proper phase change/ frequency hop indicator locations.

V. Conclusion

This paper has presented a novel receiver design that intercepts and demodulates DPSK FH-SS with no synchronization to the hopping sequence or data sequence. MATLAB simulations have shown that excellent performance is obtained when signal-to-noise ratios are greater than 30dB (where the bandwidth (BW) of each of the intercept filters in the simulation is five times the message bit frequency).

Acknowledgment

The authors would like to thank Dr. Dong Fei for providing the initial Matlab code for the K-means algorithm.

References

[1] R. Mills and G. Prescott, "Detectability models for multiple access low-probability-of-intercept networks," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 36, no. 3, pp. 848–858, Jul 2000.

[2] R. Schoolcraft, "Low probability of detection communications-waveform design and detection techniques," in *IEEE Military Communications Conference*, vol. 2, 1991, pp. 832–840.

[3] J. Proakis and M. Salehi, *Digital communications*. McGraw-hill New York, 1995.

[4] P. Wu, "On sensitivity analysis of low probability of intercept (LPI) capability," *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pp. 2889 –2895 Vol. 5, oct. 2005.

[5] J. Yang and K. Cheun, "Uplink performance of slow frequency-hop multiple-access networks using binary DPSK," *Communications, IEEE Transactions on*, vol. 53, no. 9, pp. 1511 – 1521, sept. 2005.

[6] P. A. and N. C., "Advanced detection of unknown-frequency sinusoids in broadband noise," in *Proceedings of the International Conference on Communications (ICC)*, June 1986, pp. 266–270.

[7] N. Beaulieu, W. Hopkins, and P. J. McLane, "Interception of frequency-hopped spread-spectrum signals," *Selected Areas in Communications, IEEE Journal on*, vol. 8, no. 5, pp. 853–870, Jun 1990.

[8] P. Wu, "Optimal interceptor for frequency-hopped DPSK waveform," *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pp. 2896 –2902 Vol. 5, oct. 2005.

[9] J. Lee and M. Miller, "Error performance analyses of differential phase-shift-keyed/frequency-hopping spread-spectrum communication system in the partial-band jamming environments," *Communications, IEEE Transactions on*, vol. 30, no. 5, pp. 943–952, May 1982.

[10] J. Dunn, "Detection and classification of frequency-hopped spread spectrum signals," Ph.D. dissertation, Iowa State University, Ames, Nov. 1991. [Online]. Available: http://lib.dr.iastate.edu/rtd/10027

[11] D. J. Torrieri, *Principles of secure communication systems*. Artech House, Inc., 1985.

[12] J. Jones, "Filter distortion and intersymbol interference effects on PSK signals," *Communication Technology, IEEE Transactions on*, vol. 19, no. 2, pp. 120–132, April 1971.

[13] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification and scene analysis*. Wiley New York, 1973, vol. 3.

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1 (Univ. of Calif. Press)*, pp. 281–297, Jan 1967.