A Thesis by

Matthew Hannon

Bachelor of Science, University of Kansas, 2012

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

December 2015
CHANNEL STATISTICS-DEPENDENT FREQUENCY HOPPING

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering.

____________________________________
Hyuck M. Kwon, Committee Chair

___________________________________
Xiaomi Hu, Committee Member

____________________________________
Pu Wang, Committee Member
DEDICATION

To my dog Brian
ACKNOWLEDGEMENTS

I would like to thank Dr. Kwon for his tremendous patience, and for keeping a calm and smart head throughout this adventure. He was essential for this research, and he was always willing to help me whenever I ran into problems. He was very helpful in helping me produce this work, and I am grateful. Also, I would like to thank Dr. Pham, Dr. Lane, and all the people of the United States Air Force.

This work was supported in part by the U.S. Air Force Summer Faculty Fellowship Program, the Air Force Research Laboratory (AFRL) under Grant FA9453-15-1-0308, and the Asian Office of Aerospace R&D (AOARD) AFRL under Grant FA2386-14-1-0026. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFRL or the U.S. Government.
ABSTRACT

This paper studies a channel statistics-dependent, novel, frequency-hopping (FH) pattern generation scheme. Most existing FH patterns are determined by two encryption keys: one for FH in the frequency domain, and the other for time permutation in the time domain. These keys are independent of channel conditions. Hence, an FH signal generated by these two keys occupies the entire spectrum in both the frequency and time domains, and a jammer can have a low probability of detection. However, the probability of a hit (or jamming) on the desired user’s frequency channels by partial band tone jamming (PBTJ) can be high because it is inversely proportional to the total number of available frequency positions. Can an FH system with channel-dependent adaptive FH patterns safeguard the communications systems more effectively? If the answer to that question is yes, then is it possible to find an efficient channel-dependent adaptive FH pattern, and can it be implemented cost effectively for future communication systems against jamming? The aim of this paper is to study answers to the questions posed here.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>2.</td>
<td>SYSTEM MODEL</td>
</tr>
<tr>
<td>2.1</td>
<td>Optimum Signal FH Probability</td>
</tr>
<tr>
<td>2.2</td>
<td>Typical Sequence-Based Practical Signal FH Probability Vector</td>
</tr>
<tr>
<td>2.3</td>
<td>Symbol Error Probability of FH/MPSK under Rician Fading and PBTJ</td>
</tr>
<tr>
<td>3.</td>
<td>NUMERICAL RESULTS</td>
</tr>
<tr>
<td>4.</td>
<td>CONCLUSION</td>
</tr>
<tr>
<td>5.</td>
<td>FUTURE WORK</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>31</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>35</td>
</tr>
<tr>
<td>A.</td>
<td>Averaged Typical Sequence Code</td>
</tr>
<tr>
<td>B.</td>
<td>Core Typical Sequence Code</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Example of FH/MPSK patterns with $N_f = 2$ frequency channels, $N_h = 8$ time hops per frame, and $N_s = 1$ symbol per hop</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Block diagram of overall communication system</td>
<td>7</td>
</tr>
<tr>
<td>3.</td>
<td>DMCs for FH/MPSK: (a) in presence of PBTJ; (b) absence of PBTJ</td>
<td>10</td>
</tr>
<tr>
<td>4.</td>
<td>Example for $p^j = (0.7, 0.3)^T$. Maximum angle can be achieved when $p^5 = (0, 1)^T$</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Simulation BER versus $E_b/N_0$ in dB for five exemplary FH pattern cases when Rician factor $K = 12$, $E_b/N_f = 5$ dB, $p^j = \left(\frac{1}{8}, \frac{1}{4}, \frac{3}{8}\right)^T$, $N_f = 4$ frequency channels, and $N_h = 8$ hops per frame. Hopping pattern is repeated with $N_h = 8$ hop period</td>
<td>24</td>
</tr>
<tr>
<td>6.</td>
<td>Simulation BER versus $E_b/N_0$ in dB for five exemplary FH pattern cases when Rician factor $K = 12$, $E_b/N_f = 5$ dB, $p^j = \left(\frac{1}{160}, \frac{1}{160}, \frac{157}{160}\right)^T$, $N_f = 4$ frequency channels, and $N_h = 160$ hops per frame. Hopping pattern is repeated with $N_h = 160$ hop period</td>
<td>27</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Probability of hits and probability of detection by PBTJ for three methods</td>
<td>16</td>
</tr>
<tr>
<td>2. Proposed algorithm for typical or core typical sequence-based frequency-hopping pattern generation</td>
<td>16</td>
</tr>
<tr>
<td>3. Algorithm of finding hop-time location with uniform distribution for frequency $f_i$ and $f_{\pi_i}$, respectively, with $n_i = \lfloor p_i^s N_k \rfloor$ and $n_t = \lfloor N_k / N_{UJ} \rfloor$ number of uses in a frame of $N_h$ number of hops, $i = 0, 1, \cdots, N_f - 1$ and $N_{UJ} - 1$</td>
<td>19</td>
</tr>
<tr>
<td>4. Parameters used for example simulations</td>
<td>25</td>
</tr>
</tbody>
</table>
### LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>CSI</td>
<td>Channel State Information</td>
</tr>
<tr>
<td>DMC</td>
<td>Discrete Memoryless Channel</td>
</tr>
<tr>
<td>DVB-RCS</td>
<td>Digital Video Broadcasting-Return Channel via Satellite</td>
</tr>
<tr>
<td>DVB-S2</td>
<td>DVB Satellite Second Generation</td>
</tr>
<tr>
<td>ECU</td>
<td>End Cryptographic Unit</td>
</tr>
<tr>
<td>FER</td>
<td>Frame Error Rate</td>
</tr>
<tr>
<td>FH</td>
<td>Frequency Hopping</td>
</tr>
<tr>
<td>MA</td>
<td>Multiple Access</td>
</tr>
<tr>
<td>MAI</td>
<td>Multiple Access Interference</td>
</tr>
<tr>
<td>MAU</td>
<td>Multiple Access User</td>
</tr>
<tr>
<td>MPSK</td>
<td>$M$-ary Phase-Shift Keying</td>
</tr>
<tr>
<td>MQAM</td>
<td>$M$-ary Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>NSW</td>
<td>New Satellite Waveform</td>
</tr>
<tr>
<td>PBTJ</td>
<td>Partial-Band Tone Jamming</td>
</tr>
<tr>
<td>SER</td>
<td>Symbol Error Ratio (Probability)</td>
</tr>
<tr>
<td>SIJNR</td>
<td>Signal-to-Interference-Plus-Jamming-Plus-Noise Ratio</td>
</tr>
<tr>
<td>SJNR</td>
<td>Signal-to-Jamming-Plus-Noise Ratio</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>WRAN</td>
<td>Cognitive Radio-Based Wireless Area Network</td>
</tr>
</tbody>
</table>
### LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E[X]$</td>
<td>Expectation of random variable $X$</td>
</tr>
<tr>
<td>$x$</td>
<td>Vector</td>
</tr>
<tr>
<td>$x^T$</td>
<td>Transpose of vector $x$</td>
</tr>
<tr>
<td>$|x|$</td>
<td>Norm of vector $x$</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$[x]$</td>
<td>Largest integer smaller than or equal to $x$</td>
</tr>
<tr>
<td>$\langle x, y \rangle$</td>
<td>Inner product between two vectors $x$ and $y$</td>
</tr>
<tr>
<td>$\binom{n}{m} \triangleq \frac{n!}{m!(n-m)!}$</td>
<td>Combination formula</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

The new satellite waveform (NSW) employs a frequency-hopping (FH) $M$-ary phase-shift keying (MPSK) scheme in order to provide resistance to jamming and detection by enhancing the existing digital video broadcasting-return channel via satellite (DVB-RCS) and the DVB satellite second generation (DVB-S2) [1], [2], [3], [4]. A hopping keystream coming from an end cryptographic unit (ECU) in a terminal determines its FH pattern [1], [2]. Also, time permutation within a data frame is performed by another key stream different from the frequency-hopping key stream. This is done to ensure that the entire hopping spectrum is uniformly occupied over a large number of hops and to make it difficult for a jammer or an eavesdropper to detect the transmitted information.

$$\text{Pr}(A \text{ random signal tone hit by a single fixed tone jammer}) = \frac{1}{N_f} \quad (1)$$

where $N_f$ is the total number of frequency-hopping positions in the entire spectrum.

For illustration, Figure 1 shows an example of FH/MPSK patterns with $N_f = 2$ frequency channels, $N_h = 8$ time hops per frame, and $N_s = 1$ symbol per hop. Practical parameters are different from these. The blue and red colors show the FH patterns of the desired user and a single-tone jammer. The frequency channels 0 and 1 are hit by the jammer once and three times, respectively. Hence, Figure 1 shows that half of $N_h = 8$ time hops are jammed. In addition, when a jammer has sufficient jamming power, s/he can employ multiple tones instead of a single tone. This jamming is called partial-band tone jamming (PBTJ) [5]. Then, the partial-band tone jammer can increase the probability of a hit. Once a signal tone is hit by PBTJ, then the transmitted symbol is likely in error, and the bit error rate (BER) would be about $1/((\log_2 M) \cdot N_f)$, where $M$ represents MPSK modulation with gray encoding. For example, the BER can be
5.2 \times 10^{-3} \text{ when } N_f = 64 \text{ and } M = 8. \text{ This is an uncoded BER, but still high. Therefore, the current strategy of random hopping is not the best option. This is the main motivation of this paper.}

In addition, if the number of multiple-access (MA) friendly users is greater than \( N_f \) and the FH system wants to support all MA users using the entire spectrum, then multiple-access interference (MAI) cannot be avoided, even if the FH patterns of all MA users are distinct. This is because MA users’ FH patterns may hit each other at certain frequencies and certain hops within a data frame whenever the number of the MA users \( N_u \) is larger than the number of available frequency-hopping positions \( N_f \) in the entire hopping spectrum. Hence, the on-going NSW FH scheme is also exposed under the threat of MAI.

Recently, a dynamic FH pattern instead of a fixed FH pattern was adopted in the IEEE 802.22 standard for cognitive radio-based wireless area network (WRAN) applications [6], [7]. An MA user’s frequency hops around sequentially and cyclically in the entire spectrum, and different MA users use different sequence start times to avoid collisions. For example, if user 1’s

![Diagram of FH/MPSK patterns with frequency channels, time hops per frame, and symbol per hop.](image)

Figure 1. Example of FH/MPSK patterns with \( N_f = 2 \) frequency channels, \( N_h = 8 \) time hops per frame, and \( N_s = 1 \) symbol per hop.
FH sequence is \( \{f_0, \cdots, f_{N_f-1}, f_0, \cdots \} \), then user 2’s FH sequence is \( \{f_1, \cdots, f_{N_f-1}, f_0, f_1, \cdots \} \).

Here, no collisions can occur among MA users. However, the dynamic FH patterns are also independent of jamming conditions, and a user’s signal spectrum occupies the entire spectrum. Therefore, the spectrum can be jammed with the same probability as the random FH pattern by PBTJ. In addition, the FH pattern is cyclic, and thus the jammer can detect the FH pattern of a user with high probability.

An interesting question involving this problem is the following: Can an FH system with channel-dependent adaptive FH patterns safeguard NSW military satellite communications and future systems more effectively, i.e., yield a lower BER or a lower frame error rate (FER) under jamming and MAI than the FH system with channel-independent non-adaptive FH patterns? The answer is yes. Then how can an efficient channel-dependent adaptive FH pattern be found, and can it be implemented cost effectively for future satellite communication systems against jamming and MAI? The goal of this paper is to study answers to the questions posed here.

The majority of the existing literature assumes that perfect or imperfect instantaneous channel state information (CSI) is available at the transmitter and receiver. Then, typically the channel capacity, outage probability, and BER of the communication systems are presented [8]-[25]. These theoretical results can be useful because system designer engineers are able to observe the best possible performance using perfect or imperfect instantaneous CSI. Similarly, the best FH patterns can be studied by assuming instantaneous CSI. For example, if information about instantaneous channel coefficients, the instantaneous jamming state on frequencies, and the instantaneous signal-to-interference-plus-jamming-plus-noise power ratio (SIJNR) is available at the current hop, then a best-FH pattern with best performance can be designed by avoiding the jammed frequency channels. In reality, it would be difficult to obtain instantaneous
CSI during the ongoing FH time slot and frequencies at the transmitter. Even if instantaneous jamming CSI is available at the receiver, then it would be difficult for the transmitter to hop to another frequency immediately because it takes time and causes a time delay for CSI feedback from the receiver to the transmitter. Furthermore, it requires a transition time for the receiver to synchronize with the newly received frequency and phase. Therefore, overall throughput would not be the highest, as predicted by the theory when an instantaneous CSI scheme is employed in practice. This is the second main motivation of this paper.

Another assumption is quasi-static CSI. This assumption has been adopted by a significant number of existing communications systems in the literature [8], [19], [21]. For example, suppose that a tone frequency is found to be jammed during a current frame at the receiver. Then, the transmitter assumes that the same tone is likely jammed during the next frame, also based on feedback information from the receiver, and chooses a new FH pattern against the jamming frequencies. This quasi-static CSI assumption is more practical than the instantaneous CSI assumption. However, the jamming strategy is determined not by the transmitter but by the jammer, and the quasi-static CSI used by the transmitter during the next frame interval can be quite different from that of the jammer FH pattern.

This paper assumes neither instantaneous CSI nor quasi-static CSI but rather adopts a more practical and realistic assumption for implementation, which is based on the statistical information of CSI at each tone frequency. At the end of each data frame, it is not difficult to count the number of FH time slots jammed (or interfered with) at each tone frequency in the entire spectrum. In practice, the signal-to-noise ratio (SNR) is measured typically at each hop interval with no increasingly significant cost. If the measured SNR is lower than the threshold, then it can be assumed that the hop has been jammed or interfered with. In practice, an FH
The system employs multiple (say \( N_h \)) number of hops per frame, for example, \( N_h = 320 \) hops/\( \text{frame} \). This number would be sufficient to approximate the probabilities of each tone frequency jammed by taking a single frame or a certain number of data frames called an epoch. Furthermore, these probabilities are not likely to change quickly. Hence, in this paper, it is assumed that channel statistics are available, and in turn, the channel statistics-dependent FH pattern design is studied. This paper focuses on the jamming state information statistics instead of the fading channel state information, especially whether a signal tone frequency is jammed or not. The Rician fading channel is used only for performance evaluation and not for the FH pattern design. The main contributions of the paper are as follows:

- The proposed novel channel statistics-dependent FH pattern search scheme has a low complexity.
- The proposed novel channel statistics-dependent FH pattern shows significant gain in SNR at the same BER performance over the existing random FH pattern against PBTJ under additive white Gaussian noise (AWGN) and Rician fading environments.
- The proposed novel channel statistics-dependent FH pattern also shows significant gain in SNR at the same channel capacity over the existing random FH pattern against PBTJ under an AWGN or Rician fading environment.

The rest of this paper is organized as follows: Chapter 2 describes the system model, including an optimum signal FH probability vector for a given jamming FH probability vector. The signal FH probability vector means that the probabilities of each frequency channel are being used by the transmitter in the entire spectrum. Chapter 2 also presents a typical sequence-based signal FH probability vector, as well as the symbol error rate (SER), also referred to as
symbol error probability, of FH/MPSK under Rician fading and PBTJ. Chapter 3 provides the numerical results, Chapter 4 concludes the thesis, and Chapter 5 explores future work.
CHAPTER 2
SYSTEM MODEL

The aim of this paper is to study an FH pattern that requires low computational complexity to generate and simultaneously maximize the channel capacity (i.e., maximum transmission data rate with an arbitrary small bit error rate) and minimize the symbol error rate (or BER) of an FH MPSK (or other modulation) under PBTJ and Rician (or other) fading environments. This section describes the system model.

Figure 2 shows a block diagram of an overall communication system [26, Chapter 7]. A message $W$ is drawn from the index set $\{1, 2, \ldots, 2^R\}$, where $R$ denotes the transmission data rate, and then a channel encoder maps the drawn message into a codeword $X^n (W)$ and transmits it through a discrete memoryless channel (DMC) $p(y|x)$. Next, the receiver receives a random sequence $Y^n$ with channel transition probability $p(y^n|x^n)$. In other words,

$$p(y^n|x^n) = \prod_{i=1}^{n} p(y_i|x_i) \tag{2}$$

The decoder decodes the received codeword $Y^n$ into an estimate of message $\hat{W}$.

In this paper, the DMC model is adopted, and one symbol (of multiple bits) transmission per hop is assumed. The results here are still applicable for multiple symbols per hop because each transmitted and received symbol is independent of each other due to a memoryless channel. Also, the DMC model assumes a hard decision value at the demodulator, where each received
symbol signal is demodulated individually for the decoder instead of using the received soft value.

For numerical results, it is assumed that an MPSK symbol is transmitted through a Rician fading channel with Rician factor $K$ under AWGN and PBTJ environments. Other modulations such as $M$-ary quadrature amplitude modulation (MQAM) and other fading such as Rayleigh and Nakagami can also be included. Furthermore, it is assumed that the receiver measures the SNR at every hop and compares it with the required SNR threshold $Th$. If the received SNR over a hop is smaller than $Th$, then it is assumed that the signal during the hop is either jammed by PBTJ or interfered with by friendly MA users. The receiver counts the number of jammed hops at each FH tone frequency $f_i, i = 0, \cdots, N_f - 1$, and calculates the ratio of the number of jammed hops over the total number of hops at the end of a certain period, say an epoch or a data frame of $N_h = 320$ hops (sufficient to represent the jammed probability). Let $p'_i$ denote the ratio at the FH tone frequency $f_i, i = 0, \cdots, N_f - 1$, and the probability be represented as

$$\Pr(\text{tone } f_i \text{ is jammed by a PBTJ}) = p'_i = \frac{\text{Number of hops jammed at tone } f_i}{\text{Total number of hops in a data frame}}$$

Let

$$p' = (p'_0, \cdots, p'_{N_f-1})^T$$

denote the jamming probability, and assume that it is available at the transmitter for the next data frame transmission through feedback from the receiver.
2.1 Optimum Signal FH Probability Vector

This section presents the optimum signal FH probability vector \( p^S = (p_0^S, \cdots, p_{N_f-1}^S)^T \) that maximizes the channel capacity and minimizes SER of the FH/MPSK for a given \( p^J = (p_0^J, \cdots, p_{N_f-1}^J)^T \).

**Theorem 1:** The optimum signal FH probability vector \( p^S = (p_0^S, \cdots, p_{N_f-1}^S)^T \) that maximizes the channel capacity for a given \( p^J = (p_0^J, \cdots, p_{N_f-1}^J)^T \) is

\[
p_{opt}^S = \arg \min_{p^S} \langle p^J, p^S \rangle = (1)^T
\]

under probability constraints \( p_i^S \geq 0, i = 0, \cdots, N_f - 1 \), and \( \langle 1, p^S \rangle = \Sigma_{i=0}^{N_f-1} p_i^S = 1 \), where \( 1 = (1, \cdots, 1)^T \).

**Proof:** Figures 3(a) and (b) show DMCs for FH/MPSK in the presence and absence of PBTJ, respectively. Here, \( p_{cr}^J \) and \( p_{cr}^{UL} \) represent the average SER, or total crossover probability, that a symbol is transmitted but demodulated into a different MPSK symbol when the symbol is jammed and unjammed, respectively. The transition probabilities from the transmitted symbol to the \((M - 3)\) number of non-neighbor symbols are neglected because the Euclidean distances between the transmitted symbol constellation point and the non-neighbor constellations are larger than the distances between the transmitted symbol constellation point and two neighbor constellations in the MPSK.
Figure 3. DMCs for FH/MPSK: (a) in presence of PBTJ; (b) in absence of PBTJ

Even if the non-neighbor transition probabilities are included in the analysis, the same conclusion will be reached. The detailed expressions for $p_{cr}^{J}$ and $p_{cr}^{UJ}$ will be presented in section 2.3 during discussion of the SER of the FH/MPSK system under Rician fading and PBTJ. In fact, the conditional SERs, $p_{cr}^{J}$ and $p_{cr}^{UJ}$, are obtained, including all signal constellation points in section 2.3. The channel transition matrices can be written under the presence and absence of PBTJ, respectively, as

$$ p^{J}(y|x) = \begin{bmatrix} 1 - p_{cr}^{J} & p_{cr}^{J}/2 & 0 & p_{cr}^{J}/2 \\ p_{cr}^{J}/2 & 1 - p_{cr}^{J} & p_{cr}^{J}/2 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ p_{cr}^{J}/2 & 0 & p_{cr}^{J}/2 & 1 - p_{cr}^{J} \end{bmatrix} \quad (6) $$

$$ p^{UJ}(y|x) = \begin{bmatrix} 1 - p_{cr}^{UJ} & p_{cr}^{UJ}/2 & \cdots & p_{cr}^{UJ}/2 \\ p_{cr}^{UJ}/2 & 1 - p_{cr}^{UJ} & \cdots & p_{cr}^{UJ}/2 \\ \vdots & \vdots & \vdots & \vdots \\ p_{cr}^{UJ}/2 & \cdots & p_{cr}^{UJ}/2 & 1 - p_{cr}^{UJ} \end{bmatrix} \quad (7) $$

Note that the channel matrices are symmetric. Hence, the conditional channel capacities given the presence and absence of PBTJ conditions can be written, respectively, using Theorem 7.2.1 from the work of Thomas and Thomas [26] as

$$ C^{J}(p_{cr}^{J}) = \log_{2}(M) - H(1 - p_{cr}^{J}, p_{cr}^{J}/2, 0, \cdots, 0, p_{cr}^{J}/2) \text{ bits per transmission} \quad (8) $$
\[ C^{UJ}(p^{UJ}_{cr}) = \log_2(M) - H(1 - p^{UJ}_{cr}, p^{UJ}_{cr}/2, 0, \cdots, 0, p^{UJ}_{cr}/2) \text{ bits per transmission} \quad (9) \]

where \( H(a_0, a_1, \cdots, a_{M-1}) \) is the entropy function that can be written as

\[ H(a_0, a_1, \cdots, a_{M-1}) = - \sum_{i=0}^{M-1} a_i \log_2 a_i \quad (10) \]

Therefore, the channel capacity (bits per transmission) averaged over the jamming conditions can be written as

\[ C = \Pr(\text{hop is jammed by PBTJ}) C^J(p^J_{cr}) + \Pr(\text{hop is unjammed by PBTJ}) C^{UJ}(p^{UJ}_{cr}) \quad (11) \]

The probability that a hop is jammed or unjammed by the PBTJ can be written, respectively, as

\[ \Pr(\text{hop is jammed by PBTJ}) = \sum_{i=0}^{N_f-1} \Pr(\text{hop at tone } f_i \text{ is jammed}|\text{hop is at tone } f_i) \]

\[ \times \Pr(\text{hop is at tone } f_i) = \sum_{i=0}^{N_f-1} p^J_i p^S_i = \langle p^J, p^S \rangle \quad (12) \]

and

\[ \Pr(\text{hop is unjammed by PBTJ}) = \sum_{i=0}^{N_f-1} (1 - p^J_i) p^S_i = (1 - p^J, p^S) \quad (13) \]

Thus, the averaged channel capacity in (11) can be rewritten as

\[ C = \langle p^J, p^S \rangle C^J(p^J_{cr}) + (1 - p^J, p^S) C^{UJ}(p^{UJ}_{cr}) = C^{UJ}(p^{UJ}_{cr}) + \left( C^J(p^J_{cr}) - C^{UJ}(p^{UJ}_{cr}) \right) \langle p^J, p^S \rangle \quad (14) \]

Note that \((C^J(p^J_{cr}) - C^{UJ}(p^{UJ}_{cr})) \leq 0\) because \(p^J_{cr} \geq p^{UJ}_{cr}\). Also note that \(\langle p^J, p^S \rangle \geq 0\). Therefore, the optimum \(p^S\) that maximizes \(C\) in (14) should minimize \(\langle p^J, p^S \rangle\) under the probability constraints in equation (5).

**Theorem 2:** The optimum signal FH probability vector \(p^S = (p^S_0, \cdots, p^S_{N_f-1})^T\) that minimizes the symbol error rate for a given \(p^J = (p^J_0, \cdots, p^J_{N_f-1})^T\) is also

\[ p^S_{opt} = \arg \min_{p^S} \langle p^J, p^S \rangle \quad (15) \]
under probability constraints $p_i^S \geq 0, i = 0, \ldots, N_f - 1$ and $\langle 1, p^S \rangle = \sum_{i=0}^{N_f-1} p_i^S = 1$, where $1 = (1, \ldots, 1)^T$.

**Proof**: The average SER over the jamming conditions can be written as

$$P_S(E) =$$

$$\sum_{i=0}^{N_f-1} \Pr(\text{hop is jammed by PBTJ, hop is at tone } f_i) \cdot P_s \Pr(E | \text{hop is jammed by PBTJ, hop is at tone } f_i) +$$

$$\sum_{i=0}^{N_f-1} \Pr(\text{hop is unjammed by PBTJ, hop is at tone } f_i) \cdot P_s(E | \text{hop is unjammed by PBTJ, hop is at tone } f_i)$$

(16)

The joint probabilities between signal and jamming FH events in equation (16) can be written using the independence as

$$\Pr(\text{hop is jammed by PBTJ, hop is at tone } f_i) = p_i^J p_i^S$$

(17)

and

$$\Pr(\text{hop is unjammed by PBTJ, hop is at tone } f_i) = (1 - p_i^J) p_i^S$$

(18)

And the conditional symbol error probabilities can be written as

$$\Pr(E | \text{hop is jammed by PBTJ, hop is at tone } f_i) = p_{cr}^l$$

(19)

and

$$\Pr(E | \text{hop is unjammed by PBTJ, hop is at tone } f_i) = p_{cr}^u$$

(20)

Hence, the overall average symbol error probability in (16) can be rewritten as

$$P_S(E) = \sum_{i=0}^{N_f-1} p_i^J p_i^S p_{cr}^l + \sum_{i=0}^{N_f-1} (1 - p_i^J) p_i^S p_{cr}^u$$

(21)

Equation (21) can be rewritten as

$$P_S(E) = p_{cr}^l \langle p^l, p^S \rangle + p_{cr}^u \langle 1 - p^l, p^S \rangle = p_{cr}^u + (p_{cr}^l)$$

(22)
Note that \((p_{cr}^{-1}p_{cr}^{11}) \geq 0\) because \(p_{cr}^{-1} \geq p_{cr}^{11}\). Also note that \(\langle p^j, p^s \rangle \geq 0\). Therefore, the optimum \(p^s\) that minimizes \(P_s(E)\) in equation (22) should minimize \(\langle p^j, p^s \rangle\) under the probability constraints.

**Theorem 3:** An optimum signal FH probability vector \(p^s_{opt} = \arg \min_{p^s} \langle p^j, p^s \rangle\), which simultaneously maximizes the channel capacity and minimizes the symbol error rate for a given \(p^j = (p^j_0, \ldots, p^j_{N_f-1})^T\) and satisfies the probability constraints in equations (5) or (15), can be found by using a single signal tone frequency with probability 1. The location of the single signal tone with probability 1 is the same as the tone location of PBTJ with minimum probability. In other words, it can be written as

\[
p^s_{opt} = (0, \ldots, 0, 1, 0, \ldots 0)^T
\]

\(j = \arg \min_{i \in \{0,1,\ldots,N_f-1\}} p^j_i\) \hspace{1cm} (24)

If there are multiple tones in \(p^j\) with the minimum probability, then any one of them can be used for the signal tone location with probability 1.

**Proof:** \(\langle p^j, p^s \rangle = \|p^j\|\|p^s\| \cos \theta = \cos \theta\) because \(\|p^j\| = \|p^s\| = 1\). Therefore, the minimum of \(\langle p^j, p^s \rangle\) is achieved when the angle \(\theta\) between the two vectors \(p^j\) and \(p^s\) is maximum, \(\theta = \theta_{max}\). The \(\theta_{max}\) is achieved by using only the minimum component of the \(p^j\) for \(p^s\). Figure 4 shows an example for \(p^j = (0.7, 0.3)^T\). The maximum angle can be achieved when \(p^s = (0, 1)^T\). Rigorously, this can be proven as follows: Assume that \(p^j_i\) is minimum out of \(\{p^j_i, i = 0, \ldots, N_f - 1\}\). Then,

\[
\langle p^j, p^s \rangle = \sum_{i=0}^{N_f-1} p^j_i p^s_i \geq \sum_{i=0}^{N_f-1} p^j_i p^s_i = p^j_j \sum_{i=0}^{N_f-1} p^s_i = p^j_j = \langle p^j, p^s_{opt} \rangle
\]

(25)
In other words, the value of $p_j^I$ is the minimum possible value of $\langle p^I, p^S \rangle$ and is achieved by $p^S = p_{opt}^S$ in (23). In other words, $\langle p^I, p^S \rangle = \sum_{i=0}^{N_f-1} p_i^I p_i^S$ is an affine combination, and the minimum of $\langle p^I, p^S \rangle$ is achieved at the minimum of $p_i^I, i = 0, \cdots, N_f - 1$ [27].

![Figure 4. Example for $p^J = (0.7, 0.3)^T$. Maximum angle can be achieved when $p^S = (0, 1)^T$.](image)

2.2 Typical Sequence-Based Practical Signal FH Probability Vector

Theorem 3 states that the optimum signal FH pattern is just a single tone (i.e., no FH) located at the tone frequency used by PBTJ with a minimum probability. If a transmitter employs this optimum strategy, then both the probability of hits (or jamming) and the SER can be minimized. However, the jammer can detect with high probability which single tone the transmitter is using. This is not desirable in practice because the signal is exposed to the jammer or an eavesdropper with a high probability of detection. Hence, an engineer needs to seek
another FH pattern method that can achieve both a low probability of hit and a low probability of
detection. A typical sequence-based FH pattern method is proposed in this section.

Prior to discussion of the proposed method, Table 1 lists the probability of hits and the
probability of detection by the jammer for the three FH pattern design methods: (a) existing
random FH pattern method used by the NSW [1], [2]; (b) proposed typical sequence-based FH
pattern method; and (c) arbitrary FH pattern method. The total number of tones is $N_f = 100$, and
the PBTJ jams two tones, say $f_3$ and $f_5$, out of 100. Assume that PBTJ can sense which tones
have not been used by the transmitter and thus avoid those tones for detection. The random FH
method transmits a tone signal that is hopping around all 100 tones uniformly and independently
of the jamming statistics. The probability that PBTJ can detect the signal presence or not at a
given tone during a hop duration is $1/100$ for the random FH method. But the probability of hits
by PBTJ is $2/100$ for the random FH method. On the other hand, the proposed typical sequence-
based FH pattern uses one tone out of the 98 unjammed tones randomly. Hence, the probability
of detection is $1/98$, which is slightly smaller than that of the random hopping method, but the
probability of hits is 0 for the proposed method. An arbitrary FH pattern method uses one tone
out of five unjammed tones randomly. Hence, the probability of detection is $1/5$, and the
probability of hits is zero for the arbitrary FH pattern method. Therefore, the proposed method
can achieve both zero jamming probability and a low probability of detection simultaneously.
Zero-jamming probability is desirable in the SER sense because if a signal is jammed, then the
transmitted symbol will be likely demodulated into another symbol.
Table 1. Probability of hits and probability of detection by PBTJ for three methods.

<table>
<thead>
<tr>
<th>FH Pattern Method</th>
<th>PR (Hit by PBTJ)</th>
<th>PR Detection by PBTJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random FH Using 100 Tones Independently of Jamming Conditions in NSW</td>
<td>2/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Proposed Typical Sequence-Based FH Pattern Using 98 Unjammed Tones</td>
<td>0</td>
<td>1/98</td>
</tr>
<tr>
<td>Arbitrary FH Pattern Using 5 Unjammed Tones</td>
<td>0</td>
<td>1/5</td>
</tr>
</tbody>
</table>

Table 2 lists the proposed typical sequence-based frequency-hopping pattern generation steps. The signal frequency-hopping probability vector $p^s$ is denoted by $p_{\text{uniform}}^s$ and $p_{\text{inverse}}^s$ for core typical or typical sequence-based FH pattern generations, respectively. The $i$th component $p_{i,\text{inverse}}^s$ of $p_{\text{inverse}}^s$ represents the ratio (or probability) of the number of hops that the channel frequency $f_i$ is used by the transmitter over the total number of hops in a data frame. The $p_{\text{inverse}}^s$ is obtained by taking the inverse of jamming frequency vector $p'$. 

Table 2. Proposed algorithm for typical or core typical sequence-based frequency-hopping pattern generation.

<table>
<thead>
<tr>
<th>Step 0</th>
<th>Obtain the jamming frequency-hopping statistics $p' = (p'<em>0, \ldots, p'</em>{N_f-1})^T$ for every data frame (or epoch) using the SNR threshold test performed at every hop, where $p'_i$ denotes the ratio of the number of jammed hops at frequency $f_i$ over the total number of hops per frame (or epoch), $i = 0, \ldots, N_f - 1$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Find the frequencies $f_{\pi_j}$ with negligible probabilities $0 \leq p'<em>{\pi_j} \leq \epsilon$ for a small positive $\epsilon$. In other words, find the unjammed frequencies that a jammer has never used or used with a very low probability. Let $N</em>{UJ}$ denote the number of unjammed frequencies in the entire spectrum. If $N_{UJ} = 0$, then go to Step 3 or Step 3’. Else, go to Step 2 or Step 2’. Here, the notation $\pi_j$ denotes the unjammed tone location in the frequency domain, $j = 0, \ldots, N_{UJ} - 1$.</td>
</tr>
<tr>
<td>Step 2 (with Typical Sequence)</td>
<td>Compute entropy $H(X)$ and generate a typical sequence using $p_{\text{uniform}}^s = (p_{\pi_0}^s, \ldots, p_{\pi_{N_{UJ}-1}}^s)^T = (1/N_{UJ}, \ldots, 1/N_{UJ})^T$, the values of which are in the range between $f_{\pi_0}$ and $f_{\pi_{N_{UJ}-1}}$ with equal probabilities: $p_{\pi_j}^s = 1/N_{UJ}$. Here, $X$ denotes the random variable of FH tone frequency being used, and $j = 0, \ldots, N_{UJ} - 1$ (which is different from the transmitted symbol in Figure 2). Stop.</td>
</tr>
</tbody>
</table>
Step 2’ (Alternate with Core Typical Sequence) Generate a **core typical sequence** using $p_{\text{uniform}}^S = (p_{\pi_0}^S, \ldots, p_{\pi_N}^S)^T = (1/N_{UJ}, \ldots, 1/N_{UJ})^T$. In other words, use tone frequency $f_{\pi_j}$ with $p_{\pi_j}^S$. $N_{\text{frame}} = N_{\text{frame}}/N_{UJ}$ number of times during a frame, where $N_{\text{frame}}$ denotes the number hops per frame. Use a uniform number generator or a hopping keystream coming from an end cryptographic unit in a NSW terminal to find the time-hop location of $f_{\pi_j}$ in a frame for $j = 0, \ldots, N_{UJ} - 1$. (Sample MATLAB source code is given in **Recommendation** section below.) Stop.

Step 3 (with Typical Sequence) Compute the inverse probabilities, and use them as the signal FH probabilities $p_i^{S, \text{inverse}} = \frac{1/p_i^S}{\sum_{j=0}^{N_f-1} 1/p_j^S}$ for $i = 0, \ldots, N_f - 1$. Then, compute entropy $H(X)$ of $X$ using $p_{\text{inverse}}^S$, and generate a **typical sequence** of which the values are in the range between $f_0$ and $f_{N_f-1}$ with probabilities $p_i^{S, \text{inverse}}, i = 0, \ldots, N_f - 1$. Stop.

Step 3’ (Alternate with Core Typical Sequence) Compute the inverse probabilities, and use them as the signal FH probabilities $p_i^{S, \text{inverse}} = \frac{1/p_i^S}{\sum_{j=0}^{N_f-1} 1/p_j^S}$ for $i = 0, \ldots, N_f - 1$. Then, generate a **core typical sequence** using $p_{\text{inverse}}^S$. In other words, use tone frequency $f_i$ with $p_i^{S, \text{inverse}} \cdot N_{\text{frame}}$ number of times during a frame, $i = 0, \ldots, N_f - 1$. Use a uniform number generator or a hopping keystream coming from an end cryptographic unit in a NSW terminal to find the time hop location of $f_i$ for $i = 0, \ldots, N_f - 1$ (sample MATLAB source code is given in the **Recommendation** section below.) Stop.

Note 1 For multiple FH pattern generations for $N_u$ number of MA users, the Hamming distances between the generated FH patterns for user $k$ and user $l$ should be as maximum as possible in order to minimize or avoid the MAI, $k, l = 1, \ldots, N_u$.

For typical sequence tests in Steps 2 and 3, the definition of a typical sequence is necessary.

**Definition 1 (Typical Sequence)** [26, p. 59]: The typical set $A^{(n)}_{\varepsilon}$ with respect to the probability mass function $p(x)$ is the set of sequences $(x_1, \ldots, x_n) \in \mathcal{X}^n$ with the property

$$2^{-n(H(X)+\varepsilon)} \leq p(x_1, \ldots, x_n) \leq 2^{-n(H(X)-\varepsilon)} \quad (26)$$

where $\varepsilon$ is a positive small number, $\mathcal{X}$ is the alphabet of random variable $X$, and $H(X)$ is the entropy of $X$:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x)\log_2 p(x) \text{ bits} \quad (27)$$
Note 1: Here, the random variable $X$ does not represent the transmitted symbol in Figure 2 but rather the frequency channel used by the FH system at the current hop. Also, the alphabet of $X$ is $\mathcal{X} = \{f_0, \ldots, f_{N_f-1}\}$. The sequence length is $n = N_h$ number of hops in a data frame, which is a sufficiently large number. Why does this paper consider typical sequences for the FH pattern sequence generations? The answer is because the typical sequence with the empirical entropy $\left( -\frac{1}{n} \log_2 p(x^n) \right)$ is $\epsilon$-close to the true entropy $H(X)$. Entropy means uncertainty, and it is desirable to enhance the uncertainty of the FH pattern so that a jammer may have a low detection of probability.

Note 2: The number of typical sequences is approximately equal to $2^{nH(X)}$, and is bounded between $2^{n(H(X)-\epsilon)}$ and $2^{n(H(X)+\epsilon)}$. Hence, the number of typical sequences is exponentially growing with the FH sequence length $n$. It is not necessary to search all of these typical sequences for a finite number of MA users. The set of interesting typical sequences can be restricted to a set of "core typical sequences."

Definition 2 (Core Typical Sequence): Let the number $n_i$ denote the number of frequency $f_i$ being used in a data frame of $N_h$ hops. If $n_i = \lfloor p_i^SN_h \rfloor$, i.e., $p_i^S \approx \frac{n_i}{N_h}$ for all $i = 0, 1, \ldots, N_f - 1$ and $\sum_{i=0}^{N_f-1} n_i = N_h$, then the typical sequence is called a core typical sequence.

Theorem 4: All core typical sequences have the same properties. For example, they have the same probability of being hit by PBTJ, and they are the $\epsilon$-closest to the true entropy $H(X)$, i.e., $\frac{1}{N_h} \log_2 p(x^{N_h}) - H(X)$ is the smallest among all typical sequences of length $N_h$ for a given probability mass function of $p(x)$ of $X$. When $n_i = \lfloor p_i^SN_h \rfloor$ is ordered with $n_0$ the largest, the number of core typical sequences $N_{\text{core typical seq}}$ is equal to
Proof: All $N_{\text{core typical seq}}$ number of core typical sequences have the same empirical probability mass function and are closer to the true entropy $H(X)$ than the $\{2^{nH(X)} - N_{\text{core typical seq}} \}$ number of noncore typical sequences from the definition. \hfill \Box

Recommendation: Obtain an empirical PBTJ FH probability mass vector $p^f$ that can be available in practice with low complexity. Determine the probability mass vector $p^S$ of the signal frequency-hopping random variable $X$ using the algorithm in Table 2. Use core typical sequences. Determine the hop-time locations of frequency $f_i$ in a core typical sequence using a uniform random variable or a hopping keystream coming from an end cryptographic unit in a terminal [1], [2].

Table 3 lists an example MATLAB source code with a uniform random variable to find the hop-time locations of frequency $f_i$ with $n_i = \lfloor p^S_i N_h \rfloor$ number of uses in a frame of $N_h = 8$ number of hops, $i = 0, 1, \cdots, N_f - 1$ for the case where all frequencies are jammed, and the hop-time locations of frequency $f_{\pi_i}$ with $n_i = \lfloor N_h/N_{UJ} \rfloor$, $i = 0, 1, \cdots, N_{UJ} - 1$ for $N_{UJ} < N_f$ number of frequencies are jammed with nonnegligible probabilities.

| clear all; |
| $N_h = 8$; |
| $n_i=3$; |
| out = round ($N_h.*$rand($n_i,1$)); |

| Table 3. Algorithm for finding hop-time location with uniform distribution for frequency $f_i$ and $f_{\pi_i}$, respectively, with $n_i = \lfloor p^S_i N_h \rfloor$ and $n_i = \lfloor N_h/N_{UJ} \rfloor$ number of uses in a frame of $N_h$ number of hops, $i = 0, 1, \cdots, N_f - 1$ and $N_{UJ} - 1$. |
Many core typical sequences can be employed by a friendly user as indicated by equation (28) in Theorem 4. A further interesting problem is how many multiple access users (MAUs) can be supported simultaneously with no multiple access interference during a hop interval when all MAUs want to generate their FH patterns using the same signal FH vector \( \mathbf{p}^S = \left( p_0^S, \cdots, p_{N_f-1}^S \right)^T \). Theorem 5 answers this question.

**Theorem 5**: When all MAUs generate their FH patterns using the same signal FH vector \( \mathbf{p}^S = \left( p_0^S, \cdots, p_{N_f-1}^S \right)^T \), the maximum number of multiple access users that can be supported by the FH system with no MAI (i.e., no hit) is

\[
N_{U,\text{max with no MAI}} = \left\lfloor \frac{1}{\max_i \{p_i^S\}} \right\rfloor
\]

where \( i = 0, \cdots, N_f - 1 \).

**Proof**: Let \( f_i \) denote the frequency channel with the highest probability of being used by all MAUs. Then, \( \hat{i} = \arg \max_i \{p_i^S\} \). Since all MAUs want to use this best frequency \( f_i \) with \( n_i = N_h p_i^S \) number of times per hop, the maximum number of MAUs with no MAI should satisfy

\[
n_i N_{U,\text{max with no MAI}} \leq N_h
\]

Therefore,

\[
N_{U,\text{max with no MAI}} = \left\lfloor \frac{N_h}{n_i} = \frac{N_h}{N_h p_i^S} = \frac{1}{p_i^S} \right\rfloor.
\]

For example, \( \mathbf{p}^I = \left( p_0^I, \cdots, p_{N_f-1}^I \right)^T = \left( \frac{1}{8}, 1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8} \right)^T \) with \( N_f = 4 \). The inverse FH signal probability vector proposed uses \( \mathbf{p}_{\text{inverse}}^S = \left( p_0^S, \cdots, p_{N_f-1}^S \right)^T = \left( \frac{3}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8} \right)^T \), and the NSW employs a random signal FH probability vector \( \mathbf{p}^S = \left( \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \right)^T \). Therefore, the proposed FH
system with the inverse FH signal vector and the NSW system with the random signal FH vector can simultaneously support, respectively, \( N_{u, \text{max with no MAI}} = \left\lfloor \frac{1}{p_i^5} \right\rfloor \left\lfloor \frac{1}{3/8} \right\rfloor = 2 \) and \( \left\lfloor \frac{1}{p_i^r} \right\rfloor \left\lfloor \frac{1}{1/4} \right\rfloor = 4 \) number of MAUs with no MAI. Therefore, the random signal FH vector system can support twice as many MAUs with no MAI than the proposed system. However, the BER of the proposed system would be lower than that of the random FH system because of the fewer number of hits by jamming.

### 2.3 Symbol Error Probability of FH/MPSK under Rician Fading and PBTJ

Using the moment-generating function in [28]-[30] and Craig’s formula in [31], the conditional SER of FH MPSK under Rician fading given the PBTJ state \( j = 1 \) (jammed) or \( j = 0 \) (unjammed), i.e., the

\[ p_{cr}^j = \Pr(E|\text{hop at tone } f_i \text{ is jammed by PBTJ, hop is at tone } f_i) \] in equation (19) and

\[ p_{cr}^{uj} = \Pr(E|\text{hop at tone } f_i \text{ is unjammed by PBTJ, hop is at tone } f_i) \] in equation (20), can be written, respectively, as

\[
p_{cr}^j = \frac{1}{\pi} \int_0^{(M-1)\pi/M} \frac{(1+K)}{(1+K)+\frac{g}{\sin^2 \varphi_J}} \exp \left[ -K \frac{g}{\sin^2 \varphi_J} \right] d\varphi \quad (32)
\]

and

\[
p_{cr}^{uj} = \frac{1}{\pi} \int_0^{(M-1)\pi/M} \frac{(1+K)}{(1+K)+\frac{g}{\sin^2 \varphi_N}} \exp \left[ -K \frac{g}{\sin^2 \varphi_N} \right] d\varphi \quad (33)
\]

where \( g = \sin^2 \frac{\pi}{M} \cdot \tilde{\gamma}_J \) and \( \tilde{\gamma}_N \) denote, respectively, the average signal-to-jamming-plus-noise ratio (SJNR) when jammed and the average SNR when unjammed, which are, respectively,

\[
\tilde{\gamma}_J = \left( \frac{1}{\tilde{E}_S/N_J} \right) \left( \frac{1}{\tilde{E}_S/N_0} \right) \sqrt{\pi/2} e^{-K/2} \left( 1 + K \right) I_0 \left( \frac{K}{2} \right) + K I_1 \left( \frac{K}{2} \right) \quad (34)
\]

and
\[
\bar{\gamma}_N = \frac{E_S}{N_0} \sqrt{\frac{\pi}{2}} e^{-K/2} \left[ (1 + K) I_0 \left( \frac{K}{2} \right) + K I_1 \left( \frac{K}{2} \right) \right]
\]  
(35)

where \(I_0(x)\) and \(I_1(x)\) are the zeroth order and the first-order modified Bessel function of the first kind, respectively, \(K\) is the Rician factor, \(E_S/N_0 = (\log_2 M) E_b/N_0\) is the symbol-energy-to-noise power spectral density ratio, \(E_S/N_j = (\log_2 M) E_b/N_j\) is the symbol-energy-to-jamming power spectral density ratio, \(E_b\) is the bit energy, and \(\beta\) denotes the PBTJ fraction ratio, which is related to the probability of being jammed or hit through \(\beta = (1 - N_{UJ}/N_f)\).
CHAPTER 3

NUMERICAL RESULTS

Table 4 lists the parameters used for the simulation results for the five exemplary cases shown in Figure 5.

Table 4. Parameters used for example simulations.

<table>
<thead>
<tr>
<th>Modulation</th>
<th>BPSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fading</td>
<td>Rician Fading with Rician Factor $K = 12$</td>
</tr>
<tr>
<td>Jamming</td>
<td>PBTJ</td>
</tr>
<tr>
<td>$E_b/N_j$</td>
<td>$5,\mathrm{dB}$</td>
</tr>
<tr>
<td>Number of hops per frame</td>
<td>$N_h = 8$</td>
</tr>
<tr>
<td>Number of frequencies in entire spectrum</td>
<td>$N_f = 4$</td>
</tr>
<tr>
<td>Number of users</td>
<td>$N_u = 1$</td>
</tr>
</tbody>
</table>
| Case A (random jamming and random signal FH): Jamming probability vector in spectrum $p^j$ and signal FH probability vector in spectrum $p^s$, and core typical sequence based signal FH pattern in a frame using $p^s$. | $p^j = \left(p_0^j, \ldots, p_{N_f-1}^j\right)^T = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)^T$
$p^s = \left(p_0^s, \ldots, p_{N_f-1}^s\right)^T = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)^T$
Core Typical Sequence Based Signal FH pattern = $(f_0, f_0, f_1, f_1, f_2, f_2, f_3, f_3)$ |
| Case B (random signal FH): Jamming probability vector in spectrum $p^j$ and signal FH probability vector in spectrum $p^s$, and core typical sequence based signal FH pattern in a frame using $p^s$. | $p^j = \left(p_0^j, \ldots, p_{N_f-1}^j\right)^T = \left(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)^T$
$p^s = \left(p_0^s, \ldots, p_{N_f-1}^s\right)^T = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)^T$
Core Typical Sequence Based Signal FH pattern = $(f_0, f_0, f_1, f_1, f_2, f_2, f_3, f_3)$ |
| Case C (inverse metric signal FH): Jamming probability vector in spectrum $p^j$ and inverse metric signal FH probability vector in spectrum $p_{\text{inverse}}^s$, and core typical sequence based signal FH pattern in a frame using $p_{\text{inverse}}^s$. | $p^j = \left(p_0^j, \ldots, p_{N_f-1}^j\right)^T = \left(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)^T$
$p_{\text{inverse}}^s = \left(p_0^{s_{\text{inverse}}}, \ldots, p_{N_f-1}^{s_{\text{inverse}}}\right)^T = \left(\frac{3}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)^T$
Core Typical Sequence Based Signal FH pattern = $(f_0, f_0, f_0, f_1, f_1, f_2, f_2, f_3)$ |
| Case D (optimum signal FH): Jamming probability vector in spectrum $p^j$ and optimum signal FH probability vector in spectrum $p_{\text{opt}}^s$, and core typical sequence based signal FH pattern in a frame using $p_{\text{opt}}^s$. | $p^j = \left(p_0^j, \ldots, p_{N_f-1}^j\right)^T = \left(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)^T$
$p_{\text{opt}}^s = \left(p_0^{s_{\text{opt}}}, \ldots, p_{N_f-1}^{s_{\text{opt}}}\right)^T = (1,0,0,0)^T$
Core Typical Sequence Based Signal FH pattern = $(f_0, f_0, f_0, f_0, f_0, f_0, f_0, f_0)$ |
Table 4 (continued)

| Case E (PBTJ with $\beta = 1/2$): Jamming probability vector in spectrum $\mathbf{p}^j$ and signal FH probability vector in spectrum $\mathbf{p}^s$, and core typical sequence based signal FH pattern in a frame using $\mathbf{p}^s$. | $\mathbf{p}^j = \left( p^j_0, \ldots, p^j_{N_f-1} \right)^T = \left( \frac{1}{2}, \frac{1}{2}, 0, 0 \right)^T$  
$\mathbf{p}^s = \left( p^s_0, \ldots, p^s_{N_f-1} \right)^T = \left( 0, 0, \frac{1}{2}, \frac{1}{2} \right)^T$  
Core Typical Sequence Based Signal FH pattern $= (f_2, f_2, f_2, f_2, f_3, f_3, f_3)$ |

Figure 5. Simulation BER versus $E_b/N_0$ in dB for five exemplary FH pattern cases when Rician factor $K = 12$, $E_b/N_J = 5$ dB, $\mathbf{p}^j = \left( \frac{1}{5}, \frac{1}{4}, \frac{1}{4}, \frac{3}{8} \right)^T$, $N_f = 4$ frequency channels, and $N_h = 8$ hops per frame. Hopping pattern is repeated with $N_h = 8$ hop period.
Case A shows the results for random jamming and the random signal FH pattern that was used in the NSW [1], [2]. If there is no fading, then the BER can be computed under PBTJ and AWGN of $E_b/N_0 = 30$ dB using equation (22) as

$$P_b(E)_{\text{N}0 \text{ Fading}} = p_{cr}^{UJ}(p^J, p^S) + p_{cr}^{UJ}(1 - p^J, p^S) = Q\left(\sqrt{\frac{2}{E_b/N_0}}\right)_{\beta=1} \left\langle p^J, p^S \right\rangle + Q\left(\sqrt{\frac{E_b}{N_0}}\right)(1 - p^J, p^S) = Q(2.397)\left(\frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4}\right) + Q(44.7)\left(\frac{3}{4} \cdot \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4} + \frac{3}{4} \cdot \frac{1}{4}\right)$$

$$\frac{3}{4} \cdot \frac{1}{4} \approx 0.002$$

(36)

where $Q(\alpha) = \int_\alpha^\infty \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)$ is the tail probability of the normal Gaussian random variable.

It was assumed that the Rician fading channel of a strong line of sight component such as $K = 12$ can be modeled as an AWGN channel for a given jamming state. Simulation results show that $P_b(E) = 0.0016$, which is very close to the calculation for high $E_b/N_0$.

Case B shows the results for nonuniform random jamming and uniform random signal FH, which again has been used in the NSW [1], [2]. Simulation results show that $P_b(E) = 0.0016$, whereas the calculation is $P_b(E) \approx 0.002$, which is very close to the calculation for a high $E_b/N_0$. The slight discrepancy is due to the Rician fading effects on different FH frequencies used.

Case C shows results for the nonuniform random jamming used in Case B, and the proposed inverse metric-based signal FH pattern with a core typical sequence method. Simulation results show that $P_b(E) = 0.0014$, whereas the calculation is $P_b(E) \approx 0.00175$, which is very close to the calculation for a high $E_b/N_0$. The discrepancy is due to the Rician fading effects on different FH frequency probabilities, and the quantized value $n_i = \lfloor p^S_i N_h \rfloor$ is used in the simulation, whereas the true number $n_i = p^S_i N_h$ is used in the calculation. Both
Simulation and calculation results show that the proposed signal FH pattern shows better performance than the uniform random signal FH shown in Case B.

Case D shows results for nonuniform random jamming used in Cases B and C, and the optimum signal FH pattern for which a jammer can have a high probability of detection. Simulation results show that $P_b(E) = 0.0008$, whereas the calculation is $P_b(E) \approx 0.001$ for a high $E_b/N_0$. Again, the simulation is very close to the calculation for a high $E_b/N_0$. The discrepancy is due to the quantization effects and the Rician fading effects on different FH frequency probabilities, and the calculation result shows that the optimum signal FH pattern is the best among all cases considered except Case E (which has different PBTJ jamming FH probabilities from those considered in Cases A–D).

Case E shows the most significant gain of the proposed method over the random signal FH pattern in the NSW [1], [2]. Here, PBTJ with $\beta = 1/2$ is assumed. Simulation results show that the BER of the proposed FH pattern is superior to any other signal FH patterns, including the optimum FH pattern. For example, BER $P_b(E) = 0.001$ at $E_b/N_0 = 9 \text{ dB}$, whereas all other cases cannot reach $P_b(E) = 0.001$, even if $E_b/N_0 = \infty$. This is because the signal FH pattern does not use the jammed frequencies. The BER is independent of the jamming fraction $\beta = 1/2$ and dependent on the AWGN and Rician fading environment. The probability of detection by the jammer can be increased. However, as discussed in Table 1, the probability of detection can be slightly decreased when the number of frequency channels $N_f$ is high and the PBTJ can jam only a small fraction of the entire spectrum due to insufficient jamming power, which is a typical case in practice.

Figure 6 shows simulation results for the same five exemplary cases considered in Figure 5 except with two different environments: (a) the jamming probability vector...
\[ p^I = \left( p_0^I, \ldots, p_{N_f-1}^I \right)^T = \left( \frac{1}{160}, \frac{1}{160}, \frac{1}{160}, \frac{157}{160} \right)^T \] for Cases B, C, and D; and (b) the number of hops per frame \( N_h = 160 \) and the hopping pattern is repeated with \( N_h = 160 \) hop period, whereas the hopping pattern in Figure 5 is repeated with \( N_h = 4 \) hop period.

Figure 6. Simulation BER versus \( E_b/N_0 \) in dB for five exemplary FH pattern cases when Rician factor \( K = 12 \), \( E_b/N_f = 5 \) dB, \( p^I = \left( \frac{1}{160}, \frac{1}{160}, \frac{1}{160}, \frac{157}{160} \right)^T \), \( N_f = 4 \) frequency channels, and \( N_h = 160 \) hops per frame. Hopping pattern is repeated with \( N_h = 160 \) hop period.

As expected and observed in Figure 5, the random signal FH pattern in Case B of Figure 6 shows the same results as the random jamming and random signal FH patterns in Case A. Observe the more than one decade gain in BER for the proposed methods in Cases C, D, and E over Cases A and B. For example, the simulation results for Case C show that \( P_b(E) = 0.00006 \), which is very close to the calculation \( P_b(E) = 0.0000665 \) for a high \( E_b/N_0 \). And the simulation
results for Case D show \( P_b(E) = 0.00004 \), which is very close to the calculation \( P_b(E) = 0.00005 \) for a high \( E_b/N_0 \). All proposed methods in Cases C, D, and E show significant gains in SNR in dB at the same BER over the existing NSW random FH pattern methods in Cases A and B.
CHAPTER 4
CONCLUSION

This paper has made the following recommendations: (a) obtain an empirical PBTJ FH probability mass vector \( p' \), which can be available in practice with low complexity; (b) determine the signal frequency-hopping probability mass vector \( p^s \) of using the algorithm in Table 2; (c) use core typical sequences; and (d) determine the hop-time locations of frequency \( f_i \) in a core typical sequence using a uniform random variable or a hopping keystream coming from an end cryptographic unit in a NSW terminal. Both numerical and simulation results agree well with each other and show superior performance of the proposed FH patterns to the random signal FH pattern method used in the NSW. The probability of detection by a jammer can be increased slightly with the proposed FH pattern method, but it is negligible when the number of frequency channels \( N_f \) is high and PBTJ can only affect a small fraction of the entire spectrum due to insufficient jamming power. This is a typical case in most practical scenarios.
CHAPTER 5

FUTURE WORK

The number of possible typical sequences grows to a very large number when the frequency bands and/or the sequence pattern length increases. Therefore, the use of heuristics to efficiently choose the optimal typical sequences out of this large set while not processing all of them is paramount when under computational constraints. A number of heuristic techniques could be explored as well as experimenting with the Poisson distribution to help solve the complex computation issue. Furthermore, adding multiple users, with each given the most optimal sequences found while not allowing the sequences to have close proximity to each other’s frequency bands to avoid interference, would be advantageous. Another future work possibility would be to have a feedback loop that would alert the system when a previously picked sequence is being jammed and then automatically recalibrate that user’s sequence to a better pattern. Also, the measured SINR could be converted directly into the probabilities of which bands are being jammed. Techniques in artificial intelligence and machine learning could be applied to find more efficient sequences as well as learn about jammer behavior based on previous measurements and jammer changing patterns (i.e., number of times the jammer switched and patterns of where it switches to), in order to have an overall smarter system that is more capable of real-time adaptation in a dynamic and automatic fashion.
REFERENCES
REFERENCES


REFERENCES (continued)


REFERENCES (continued)


APPENDIX A

AVERAGED TYPICAL SEQUENCE CODE

%Typical Sequences Averaged Code Base
%BPSK - AWGN - Rician - Frequency Hopping w/ Typical Sequence Patterns Averaged
%By Matthew, Dr. Kwon, and Shaung
%8/5/2015

%close all;
clear all;
clc;

%Note: for debugging
format long

%Seed random generator (i.e., used to get some variation between runs)
rng('default')

%Define n which is the number of available time slots in simulation
n = 8; %160

%Define alphabet base (i.e., 0; 1; 2; 3)
base = 4; %8

%Variables that can/should be altered in this program
K = 12; %Rician factor / line of sight strength
Eb_div_Nj_dB = 5; %Signal-to-interference ratio in dB

%Beta term for PBTJ (depends on Eb/Nj and jamming BW)
beta = 1;

%Define number of bits
numBits = 10^6; %10^7 <--- for 64bit Matlab

%Define a tolerance epsilon
epsilon = 0.01; %Default: 0.01

%Note: the following code is responsible for converting the estimated real probabilities of what the jammer is going to an inverse relationship which will be used for calculating the probabilities for the desired hopping ratios which will be utilized to generate the typical hopping sequences.
APPENDIX A (continued)

Further note: future work will calculate these probabilities of what the real jammer is going to do (estimate from previous jamming data) from the measured signal to jamming ratio (SIR/SJR) at each frequency hopping slot per unit time (time in the past that is...We can't know the future exactly because we are only human but we can make some good guesses).

FreqSlot_Probabilities = zeros(1, base);

hasZero = 0;
counter = 0;

%Define variable for calculating summation of inverses to perhaps be used later
summation_of_inverses = 0;

%Define values for Pr[The i-th tone frequency is jammed]
%TODO: this is in the future is going to be calculated based on the measured signal to interference/jamming ratio at each frequency band slot
Pr_ith_tone_FrequencyIsJammed_0 = [1/2, 1/2, realmin('double'), realmin('double')];
Pr_ith_tone_FrequencyIsJammed_1 = [1/4, 1/8, 3/8, 1/4]; %curveColor = 'go-';
Pr_ith_tone_FrequencyIsJammed_2 = [1/8, 1/4, 1/4, 3/8]; %curveColor = 'bo-';
Pr_ith_tone_FrequencyIsJammed_3 = [3/8, 5/8, realmin('double'), realmin('double')]; %curveColor = 'mo-';
Pr_ith_tone_FrequencyIsJammed_4 = [1/4, 1/4, 1/4, 1/4]; %curveColor = 'yo-'

%Note: tone jamming
Pr_ith_tone_FrequencyIsJammed_5 = [realmin('double'), 8/8, realmin('double'), realmin('double')]; %curveColor = 'co-';

%Note: used for long sequence patterns
Pr_ith_tone_FrequencyIsJammed_6 = [1/160, 1/160, 1/160, 157/160];

%This sequence is for base 8 (eight frequency bands)
Pr_ith_tone_FrequencyIsJammed_diff = [1/8, 3/8, 2/8, 1/8, 1/32, 1/32, 1/32, 1/32];

%Note: Pick a jamming pattern here
Pr_ith_tone_FrequencyIsJammed = Pr_ith_tone_FrequencyIsJammed_2;

%Note: Pick the curve color
curveColor = 'r-'; %Possible colors to use: r g b c m y k

%The following code converts the jamming probabilities into optimal probabilities of where the signal should transmit
for i = 1:length(Pr_i_th_tone_FrequencyIsJammed)
    if Pr_i_th_tone_FrequencyIsJammed(i) == realmin('double') %approx. = 0
        hasZero = 1;
        counter = counter + 1;
    else
        summation_of_inverses = summation_of_inverses + (1 / Pr_i_th_tone_FrequencyIsJammed(i));
    end
end

if hasZero == 1
    for i = 1:length(Pr_i_th_tone_FrequencyIsJammed)
        if Pr_i_th_tone_FrequencyIsJammed(i) == realmin('double'); %approx. = 0
            FreqSlot_Probabilities(i) = 1/counter;
        else
            FreqSlot_Probabilities(i) = realmin('double'); %approx. = 0
        end
    end
else
    for i = 1:length(Pr_i_th_tone_FrequencyIsJammed)
        FreqSlot_Probabilities(i) = (( 1 / Pr_i_th_tone_FrequencyIsJammed(i) ) /
            summation_of_inverses);
    end
end

%Notes:
%   We know n = 4 (default) which is the length of each sequence (i.e., 0123 or 1111)
%   And since base 4 (default) alphabet we have 0; 1; 2; 3 as characters

%First compute the entropy H(x) = sum [ p(x) * log(1 / p(x)) ]
H_x = 0;
for i = 1:length(FreqSlot_Probabilities)
    H_x = H_x + FreqSlot_Probabilities(i)*log2(1 / FreqSlot_Probabilities(i)); %/log(4);
end

%Display entropy value
H_x

%Note: 4^4 = 256 possible hopping patterns in a 4x4 freq. hopping grid:
totalFrequencyPatterns = base^n

%Now we need to calculate the lower and upper bounds for the typical set'
APPENDIX A (continued)

%Calculate lower bound: \(2^{(-1 \times n \times (H(x) + \epsilon))}\)
lower_bound = \(2^{(-1 \times n \times (H_x + \epsilon))}\)

%Calculate upper bound
higher_bound = \(2^{(-1 \times n \times (H_x - \epsilon))}\)

%For typical set criteria to be met it must meet the following condition:
% lower_bound <= pmf(x^n) <= higher_bound

%Now lets define the pmf(x^n)
%where pmf is the probability mass function

%Generate all possible frequency hopping patterns
frequencyHoppingPatterns = dec2base(0:totalFrequencyPatterns-1, base, n);

%Now obtain values which determine whether hopping pattern is typical set
%So first calculate each pmf value for all combinations
pmf = ones(1, totalFrequencyPatterns);

for i = 1:length(frequencyHoppingPatterns)
    %Reset slot counters
    slot_counters = zeros(1, n);
    
    for j = 1:length(FreqSlot_Probabilities)
        %Convert number to string for string comparison
        %For example 3332 ----> '3332'
        tmp = num2str(frequencyHoppingPatterns(i, 1:n));

        %Calculate pmf for frequency hopping pattern
        %i.e., for '3332' --> pmf(255) = P_0^0 * P_1^0 * P_2^1 * P_3^3
        pmf(i) = pmf(i) * FreqSlot_Probabilities(j)^length(strfind(tmp, num2str(j-1)));

        %Debug
        %slot_counters(1, j) = length(strfind(tmp, num2str(j-1)))
    end
end

% Uncomment for debugging
% slot_counters
end
% Counter for number of typical sequences found
numTypicalSequences = 0;

% Figure out lowest entropy pattern
lowestPatternFound = 9999999;
lowestPatternFoundIndex = 9999999;

% Now find out how many are in the typical set
dataBitsFreqSlots1 = char([]);
for i = 1:length(pmf)
    if ((pmf(i) >= lower_bound) && (pmf(i) <= higher_bound))
        disp('In typical set: ')
        frequencyHoppingPatterns(i, 1:n) % uncomment me to see sequences
    end
    % Increment counter
    numTypicalSequences = numTypicalSequences + 1;

    % Find lowest entropy
    if pmf(i) <= lowestPatternFound
        lowestPatternFound = pmf(i);
        lowestPatternFoundIndex = i;
    end

    % Append found frequency hopping pattern to single string containing
    % previously found patterns as well.
    for j = 1:n
        dataBitsFreqSlots1 = [dataBitsFreqSlots1 ' ' frequencyHoppingPatterns(i, j:j)];
    end
    pmf(i);
end

% pmf

% Display number of typical sequences found
numTypicalSequences

% Note: used for debugging
% lowestPatternFound
% lowestPatternFoundIndex

% Now figure out how many repeats of the typical sets are needed to fill the bit
%buffer

%Fill up the bit buffer
numRepeats = ceil((numBits)/(length(dataBitsFreqSlots1)/2))
dataBitsFreqSlots = repmat(dataBitsFreqSlots1 , [1,numRepeats]);

%Convert string back to a number array now
dataBitsFreqSlots = str2num(dataBitsFreqSlots);

%trim it to proper size
dataBitsFreqSlots((numBits)+1:length(dataBitsFreqSlots)) = '';

%Now define tone jamming component power
%Note:  Pj = tone jamming component power = Nj / ( beta * Ts )
% Ts = Symbol time = 1 and beta = jamming fraction (0 <= beta <= 1)
Ts = 1;

%Note: can find Nj value from defined ratio of Eb/Nj = # = some number

%Now define value for Eb/Nj
Eb_div_Nj = 10^(Eb_div_Nj_dB/10);

%Now define value for Eb
Eb = 1;

%Now find Nj
Nj = Eb/Eb_div_Nj;

%Tone jamming component power
toneJammPower = Nj / ( beta * Ts );
sqrtToneJammPower = sqrt(toneJammPower);

%Generate simulation bits for data
dataBits = rand(1,numBits) > 0.5;
signalData = 2*dataBits-1; %Turn 1s and 0s into BPSK -1s and +1s
SNRindB_signal = 0:2:40;
sym_err_prob = zeros(1, length(SNRindB_signal));

%K = specularPower / nlosPower
rician_LOS = sqrt(K / (K+1));
rician_NLOS = sqrt(1 / (2*(K+1)));
APPENDIX A (continued)

```matlab
%Note: these variables are for the commented counting code below which
%keeps track of the number of hits and corresponding number of bit errors on
%hits
numHits = zeros(length(SNRindB_signal));
umBitErrors = zeros(length(SNRindB_signal));
umTotalBitErrors = zeros(length(SNRindB_signal));

for i = 1:length(SNRindB_signal)
    %Create AWGN noise channel
    noiseComponent = (1/sqrt(2))*(randn(1,numBits) + 1i*randn(1,numBits));

    %Create rician fading channel
    ricianChannel = rician_NLOS*randn(1,numBits) + rician_LOS +
    1i*rician_NLOS*randn(1,numBits);

    %No rician channel
    ricianChannel = 1;

    %Note: poisson_RV = poissrnd(probJammed, 1, numBits) > 0;
    % this is no longer used, and remains here from the times of
    % simulating jammed bits using poisson distribution rather than the
    % newer method used here

    %Note: What we are now doing is limiting the range of the random number
    %generator to only include the frequency bands that were previously
    %measured and determined that a jammer is active in the corresponding
    %bands represented by the row vector Pr_ith_tone_FrequencyIsJammed
    %which can be found above in the code. Further the following code will weight the random
    %number generator corresponding to the expected likelihood for jamming.

    %Future work will have the values
    %in this vector calculated from signal to jamming/interference ratio
    %measurements. Also note that in the case of random jamming the random
```
%number generator will be uniform on the entire range of frequency
%bands (i.e., 0, 1, 2, 3 in the current POC code case).

%Note: the following lines just makes a matrix of pairs with the first
%element being the jamming probability and the second field is the
%frequency band corresponding to its jamming probability
%Why? - used for faster look up because avoids extra loop

%Define ruler variables to keep track of the weighted random numbers
%generated and there ordinal
ruler = zeros(length(Pr_ith_tone_FrequencyIsJammed), 2);
rulerSize = 0;

for y = 1:length(Pr_ith_tone_FrequencyIsJammed)
    %Skip the zero jamming probability slots
    if Pr_ith_tone_FrequencyIsJammed(y) == 0 || Pr_ith_tone_FrequencyIsJammed(y) ==
        realmin('double')
        ruler(y, 1) = realmin('double');
        ruler(y, 2) = y-1;
        continue;
    end

    %We found one!
    ruler(y, 1) = Pr_ith_tone_FrequencyIsJammed(y);
    ruler(y, 2) = y-1;
    rulerSize = rulerSize + Pr_ith_tone_FrequencyIsJammed(y);
end

%Sanity check
if rulerSize ~= 1
    errorMsg = 'rulerSize did not add up to a value of 1! Probabilities must total 1.';
    error(errorMsg)
end

%First lets get some random floating point numbers between 0 and 1
randomNumbers = rand(1, numBits);

%Define variable to hold jammed frequency slots
jammedFreqSlots = zeros(1, numBits);

%Sort the ruler data structure just created by the first column
sortedRuler = sortrows(ruler)
tic; %Start timer
for x = 1:length(randomNumbers)
    rulerSize = 0;
    for y = 1:length(sortedRuler)
        if(sortedRuler(y) == 0 || sortedRuler(y) == realmin('double'))
            continue;
        end

        if( randomNumbers(x) <= rulerSize + sortedRuler(y) )
            rulerSize = rulerSize + sortedRuler(y);
            jammedFreqSlots(x) = sortedRuler(y, 2);
            break; %Get back out to main loop since we found correct jamm slot
        else
            rulerSize = rulerSize + sortedRuler(y);
        end
    end
end
toc; %End timer

%Send it through the channel; apply AWGN; apply jamming if needed
y = ricianChannel.*signalData + noiseComponent*10^(-SNRindB_signal(i)/20) +
    (jammedFreqSlots == dataBitsFreqSlots).*ones(1, numBits).*sqrtToneJammPower;

demodBits = real(y ./ ricianChannel) > 0; %Demod by dividing by channel
sym_err_prob(i) = size(find([dataBits - demodBits]), 2); %Add up errors

% Used for Debugging

% if jammedFreqSlots(x) == dataBitsFreqSlots(x)
%   if dataBits(x) ~= demodBits(x)
%       numBitErrors(i) = numBitErrors(i) + 1;
%   end
% end
% numHits(i) = numHits(i) + 1;
APPENDIX A (continued)

```matlab
%%
%% if dataBits(x) ~= demodBits(x)
%% numTotalBitErrors(i) = numTotalBitErrors(i) + 1;
%% end
%% end

% Note: these variables are used for debugging purposes when uncommented
% numHits
% numHits./length(y)
% numBitErrors
% numBitErrors./length(y)
% numTotalBitErrors
% numTotalBitErrors./length(y)

% Now turn it into ratio
sym_err_prob = sym_err_prob / numBits;

% Plot it!
semilogy(SNRindB_signal, sym_err_prob, curveColor, 'LineWidth', 2);
grid on;
hold on;

% Label stuff
% axis([0 40 10^-5 10^0]);
% xlabel('Eb/N0 [dB]');
% ylabel('BER');
% legend('randomJamming', 'randomJammAndHopping');
% legend('random Jamming', 'tone 1 is jammed', 'tone 2 is jammed',
% 'randomJammAndHopping');
% legend('randHopRandJam', '00112233', '00011223', '00000000', '22223333NoJam');
% legend('00112233', 'randHopRandJam', '22223333NoJam', '00000000','00011223');
```
APPENDIX B

CORE TYPICAL SEQUENCE BASE CODE

%Core Typical Sequences Code Base
%BPSK - AWGN - Rician - Frequency Hopping w/ Core Typical Sequence Patterns
%By Matthew, Dr. Kwon, and Shaung
%8/5/2015

%close all;
clear all;
clc;

%Note: for debugging
format long

%Seed random generator (i.e., used to get some variation between runs)
rng('default')

%Define n which is the number of available time slots in simulation
n = 160; %8

%Define alphabet base (i.e., 0; 1; 2; 3)
base = 4; %8

%Variables that can/should be altered in this program
K = 12; %Rician factor / line of sight strength
Eb_div_Nj_dB = 5; %Signal-to-interference ratio in dB

%Beta term for PBTJ (depends on Eb/Nj and jamming BW)
beta = 1;

%Define number of bits
numBits = 10^7; %10^7 <--- for 64bit Matlab

%Define a tolerance epsilon
epsilon = 0.01; %Default: 0.01

%Note: the following code is responsible for converting the estimated real
%probabilities of what the jammer is going to an inverse relationship
%which will be used for calculating the probabilities for the desired
%hopping ratios which will be utilized to generate the typical hopping
%sequences.
Further note: future work will calculate these probabilities of what the real jammer is going to do (estimate from previous jamming data) from the measured signal to jamming ratio (SIR/SJR) at each frequency hopping slot per unit time (time in the past that is... We can't know the future exactly because we are only human but we can make some good guesses).

\[
\text{FreqSlot\_Probabilities} = \text{zeros(1, base)};
\]

\[
\text{hasZero} = 0;
\]
\[
\text{counter} = 0;
\]

% Define variable for calculating summation of inverses to perhaps be used later
\[
\text{summation\_of\_inverses} = 0;
\]

% Define values for \(\Pr[\text{The i-th tone frequency is jammed}]\)
% TODO: this is in the future is going to be calculated based on the measured signal to interference/jamming ratio at each frequency band slot
\[
\begin{align*}
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_0 &= [1/2, 1/2, \text{realmin('double')}, \text{realmin('double')}]; \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_1 &= [1/4, 1/8, 3/8, 1/4]; & \text{\%curveColor = 'go-';} \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_2 &= [1/8, 1/4, 1/4, 3/8]; & \text{\%curveColor = 'bo-';} \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_3 &= [3/8, 5/8, \text{realmin('double')}, \text{realmin('double')}]; & \text{\%curveColor = 'mo-';} \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_4 &= [1/4, 1/4, 1/4, 1/4]; & \text{\%curveColor = 'yo-'} \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_5 &= [\text{realmin('double')}, 8/8, \text{realmin('double')}, \text{realmin('double')}]; & \text{\%curveColor = 'co-';} \\
\Pr\_\text{ith}\_\text{tone\_FrequencyIsJammed}_6 &= [1/160, 1/160, 1/160, 157/160]; & \text{\%Note: used for long sequence patterns} \\
\text{Pr\_ith\_tone\_FrequencyIsJammed\_diff} &= [1/8, 3/8, 2/8, 1/8, 1/32, 1/32, 1/32, 1/32]; & \text{\%Note: This sequence is for base 8 (eight frequency bands)} \\
\text{Pr\_ith\_tone\_FrequencyIsJammed} &= \text{Pr\_ith\_tone\_FrequencyIsJammed\_6}; & \text{\%Note: Pick a jamming pattern here} \\
\text{curveColor} &= 'r-'; & \text{\%Note: Pick the curve color} \\
\% Possible colors to use: r g b c m y k \\
% The following code converts the jamming probabilities into optimal probabilities of where the signal should transmit 
\text{for } i = 1:length(\text{Pr\_ith\_tone\_FrequencyIsJammed})
if Pr_i_th_tone_FrequencyIsJammed(i) == realmin('double') %approx. = 0
    hasZero = 1;
    counter = counter + 1;
else
    summation_of_inverses = summation_of_inverses + (1 / Pr_i_th_tone_FrequencyIsJammed(i));
end
end

if hasZero == 1
    for i = 1:length(Pr_i_th_tone_FrequencyIsJammed)
        if Pr_i_th_tone_FrequencyIsJammed(i) == realmin('double'); %approx. = 0
            FreqSlot_Probabilities(i) = 1/counter;
        else
            FreqSlot_Probabilities(i) = realmin('double'); %approx. = 0
        end
    end
else
    for i = 1:length(Pr_i_th_tone_FrequencyIsJammed)
        FreqSlot_Probabilities(i) = (( 1 / Pr_i_th_tone_FrequencyIsJammed(i) ) / summation_of_inverses);
    end
end

%Notes:
% We know n = 4 (default) which is the length of each sequence (i.e., 0123 or 1111)
% And since base 4 (default) alphabet we have 0; 1; 2; 3 as characters

%First compute the entropy H(x)= sum [ p(x) * log(1 / p(x)) ]
H_x = 0;
for i = 1:length(FreqSlot_Probabilities)
    H_x = H_x + FreqSlot_Probabilities(i)*log2(1 / FreqSlot_Probabilities(i)); %/log(4);
end

%Display entropy value
H_x

%Note: 4^4 = 256 possible hopping patterns in a 4x4 freq. hopping grid:
totalFrequencyPatterns = base^n

%Now we need to calculate the lower and upper bounds for the typical set'
%Calculate lower bound: 2^(-1 * n * (H(x) + epsilon))
lower_bound = 2^{(-1 \times n \times (H_x + \epsilon))}

% Calculate upper bound
higher_bound = 2^{(-1 \times n \times (H_x - \epsilon))}

%%%%%% Overwrite pattern %
% This is for core typical sequences %
%%%%%%

% Pick 0013
%dataBitsFreqSlots1 = ' 0 0 1 3'
dataBitsFreqSlots1 = ' 0 0 0 1 0 2 1'
dataBitsFreqSlots1 = ' 0 0 1 3 0 0 1 3'
dataBitsFreqSlots1 = ' 0 0 0 0 0 0 0 0'
dataBitsFreqSlots1 = ' 3 3 3 3 3 3 3 3'
dataBitsFreqSlots1 = ' 0 0 0 1 1 2 2 3'
dataBitsFreqSlots1 = ' 0 0 1 1 2 2 3 3'
dataBitsFreqSlots1 = ' 2 2 2 3 3 3 3 3'

% Note: Generate long frame sequences for the 160 sequence length case with code below

% Case A and B
band0 = '0'; band1 = '1'; band2 = '2'; band3 = '3';
longFrameSequenceAB = ''; % define this variable so it can be used below
for i = 1:40
    longFrameSequenceAB = [longFrameSequenceAB ' ' band0];
end
for i = 1:40
    longFrameSequenceAB = [longFrameSequenceAB ' ' band1];
end
for i = 1:40
    longFrameSequenceAB = [longFrameSequenceAB ' ' band2];
end
for i = 1:40
    longFrameSequenceAB = [longFrameSequenceAB ' ' band3];
end

longFrameSequenceAB

% Case C
band0 = '0'; band1 = '1'; band2 = '2'; band3 = '3';
longFrameSequenceC = ''; % define this variable so it can be used below
for i = 1:53
    longFrameSequenceC = [longFrameSequenceC ' ' band0];
end
for i = 1:53
    longFrameSequenceC = [longFrameSequenceC ' ' band1];
end
for i = 1:53
    longFrameSequenceC = [longFrameSequenceC ' ' band2];
end
for i = 1:1
    longFrameSequenceC = [longFrameSequenceC ' ' band3];
end
longFrameSequenceC

%      Case D      %
band0 = '0'; band1 = '1'; band2 = '2'; band3 = '3';
longFrameSequenceD = ''; %define this variable so it can be used below
for i = 1:160
    longFrameSequenceD = [longFrameSequenceD ' ' band0];
end
longFrameSequenceD

%      Case E      %
band0 = '0'; band1 = '1'; band2 = '2'; band3 = '3';
longFrameSequenceE = ''; %define this variable so it can be used below
for i = 1:80
    longFrameSequenceE = [longFrameSequenceE ' ' band2];
end
for i = 1:80
    longFrameSequenceE = [longFrameSequenceE ' ' band3];
end
longFrameSequenceE

%Set the manually configured pattern
dataBitsFreqSlots1 = longFrameSequenceC;

%Now figure out how many repeats of the typical sets are needed to fill the bit %buffer
%Fill up the bit buffer
numRepeats = ceil((numBits)/(length(dataBitsFreqSlots1)/2))
dataBitsFreqSlots = repmat(dataBitsFreqSlots1 , [1,numRepeats]);

%Convert string back to a number array now
dataBitsFreqSlots = str2num(dataBitsFreqSlots);

%trim it to proper size
dataBitsFreqSlots((numBits)+1:length(dataBitsFreqSlots)) = '';

%Now define tone jamming component power
%Note:  P_j = tone jamming component power = N_j / ( beta * T_s )
%      T_s = Symbol time = 1 and beta = jamming fraction (0 <= beta <= 1)
Ts = 1;

%Note: can find N_j value from defined ratio of Eb/N_j = # = some number

%Now define value for Eb/N_j
Eb_div_Nj = 10^(Eb_div_Nj_dB/10);

%Now define value for Eb
Eb = 1;

%Now find N_j
Nj = Eb/Eb_div_Nj;

%Tone jamming component power
toneJammPower = Nj / ( beta * T_s );
sqrtToneJammPower = sqrt(toneJammPower)

%Generate simulation bits for data
dataBits = rand(1,numBits) > 0.5;
signalData = 2*dataBits-1; %Turn 1s and 0s into BPSK -1s and +1s

SNRindB_signal = 0:2:40;
sym_err_prob = zeros(1, length(SNRindB_signal));

%K = specularPower / nlosPower
rician_LOS = sqrt(K / (K+1));
rician_NLOS = sqrt(1 / (2*(K+1)));
APPENDIX B (continued)

%Note: these variables are for the commented counting code below which keeps track of the number of hits and corresponding number of bit errors on hits
numHits = zeros(length(SNRindB_signal));
numBitErrors = zeros(length(SNRindB_signal));
numTotalBitErrors = zeros(length(SNRindB_signal));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:length(SNRindB_signal)
    %Create AWGN noise channel
    noiseComponent = (1/sqrt(2)*(randn(1,numBits) + 1i*randn(1,numBits)));

    %Create rician fading channel
    ricianChannel = rician_NLOS*randn(1,numBits) + rician_LOS +
    1i*rician_NLOS*randn(1,numBits);

    %No rician channel
    %ricianChannel = 1;

    %Note: poisson_RV = poissrnd(probJammed, 1, numBits) > 0;
    % this is no longer used, and remains here from the times of
    % simulating jammed bits using poisson distribution rather than the
    % newer method used here

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%Note: What we are now doing is limiting the range of the random number generator to only include the frequency bands that were previously measured and determined that a jammer is active in the corresponding bands represented by the row vector Pr_ith_tone_FrequencyIsJammed which can be found above in the code. Further the following code will weight the random number generator corresponding to the expected likelihood for jamming.

%Future work will have the values
%in this vector calculated from signal to jamming/interference ratio
%measurements. Also note that in the case of random jamming the random number generator will be uniform on the entire range of frequency bands (i.e., 0, 1, 2, 3 in the current POC code case).
% Note: the following lines just makes a matrix of pairs with the first
% element being the jamming probability and the second field is the
% frequency band corresponding to its jamming probability
% Why? - used for faster look up because avoids extra loop

% Save old value because it is going to be overwritten in the case of core sequences
Pr_ith_tone_FrequencyIsJammed_old = Pr_ith_tone_FrequencyIsJammed;
% Pr_ith_tone_FrequencyIsJammed = [1/8, 1/4, 1/4, 3/8];
% Pr_ith_tone_FrequencyIsJammed = [1/2, 1/2, realmin('double'), realmin('double')];  % 0, 0 ];
% Pr_ith_tone_FrequencyIsJammed = [1/4, 1/4, 1/4, 1/4];
Pr_ith_tone_FrequencyIsJammed = [1/160, 1/160, 1/160, 157/160];

% Define ruler variables to keep track of the weighted random numbers
% generated and there ordinal
ruler = zeros(length(Pr_ith_tone_FrequencyIsJammed), 2);
rulerSize = 0;
for y = 1:length(Pr_ith_tone_FrequencyIsJammed)
    % Skip the zero jamming probability slots
    if Pr_ith_tone_FrequencyIsJammed(y) == 0 || Pr_ith_tone_FrequencyIsJammed(y) ==
        realmin('double')
        ruler(y, 1) = realmin('double');
        ruler(y, 2) = y-1;
        continue;
    end

    % We found one!
    ruler(y, 1) = Pr_ith_tone_FrequencyIsJammed(y);
    ruler(y, 2) = y-1;
    rulerSize = rulerSize + Pr_ith_tone_FrequencyIsJammed(y);
end

% Sanity check
if rulerSize ~= 1
    errorMsg = 'rulerSize did not add up to a value of 1! Probabilities must total 1.';
    error(errorMsg)
end

% First lets get some random floating point numbers between 0 and 1
randomNumbers = rand(1, numBits);

% Define variable to hold jammed frequency slots
jammedFreqSlots = zeros(1, numBits);

%Sort the ruler data structure just created by the first column
sortedRuler = sortrows(ruler)
tic; %Start timer
for x = 1:length(randomNumbers)
    rulerSize = 0;
    for y = 1:length(sortedRuler)
        if(sortedRuler(y) == 0 || sortedRuler(y) == realmin('double'))
            continue;
        end
        if( randomNumbers(x) <= rulerSize + sortedRuler(y) )
            rulerSize = rulerSize + sortedRuler(y);
            jammedFreqSlots(x) = sortedRuler(y, 2);
            break; %Get back out to main loop since we found correct jamm slot
        else
            rulerSize = rulerSize + sortedRuler(y);
        end
    end
end
toc; %End timer

%Send it through the channel; apply AWGN; apply jamming if needed
y = ricianChannel.*signalData + noiseComponent*10^(-SNRindB_signal(i)/20) +
(jammedFreqSlots == dataBitsFreqSlots).*ones(1, numBits).*sqrtToneJammPower;

demodBits = real(y ./ ricianChannel) > 0; %Demod by dividing by channel
sym_err_prob(i) = size(find([dataBits - demodBits]), 2); %Add up errors
end

% Note: these variables are used for debugging purposes when uncommented
% numHits
% numHits./length(y)
% numBitErrors
% numBitErrors./length(y)
% numTotalBitErrors
% numTotalBitErrors./length(y)

% Now turn it into ratio
sym_err_prob = sym_err_prob / numBits;

% Plot it!
semilogy(SNRindB_signal, sym_err_prob, curveColor, 'LineWidth', 2);
grid on;
hold on;

% Label stuff
% axis([0 40 10^-5 10^0]);
% xlabel('Eb/N0 [dB]');
% ylabel('BER');
% legend('randomJamming', 'randomJammAndHopping');
APPENDIX B (continued)

```matlab
% legend('random Jamming', 'tone 1 is jammed', 'tone 2 is jammed', 'randomJammAndHopping');
% legend('randHopRandJam', '00112233', '00011223', '00000000', '22223333NoJam');
% legend('00112233', 'randHopRandJam', '22223333NoJam', '00000000', '00011223');
```