Adaptive Fuzzy Frequency Hopper

Peter J. Pacini, *Member, IEEE,* and Bart Kosko, *Member, IEEE*

Abstract—**An adaptive fuzzy system generates the frequency hopping sequence for a spread spectrum communications system. The system learns rules from data and acts as a pseudorandom number generator. The IMSL uniform random number generator gives training samples. An adaptive scheme learns associations between previous samples and the current sample and encodes these as fuzzy rules. The output fuzzy set for each rule acts as a conditional probability density function. The if-part of the rule states the conditions. At each step thirty prior outputs, scanned according to a fixed sampling pattern, give a new sample distribution x_k. The vector x_k partially matches the if-part distribution of a fuzzy rule and partially fires that rule's output fuzzy set. With the estimated output fuzzy sets the fuzzy system computes the conditional density $p_{Y|X}(y|x_k)$ for input field X and output field Y. Defuzzification yields the next number in the frequency hopping sequence. The rules, sampling pattern, and initial conditions fix the output sequence. An eavesdropper who did not know all three could not predict the sequence. This fuzzy system can generate a sequence uniform over any number of frequencies. We tested the fuzzy system with 100 and 1025 frequencies and compared it to a shift register with linear feedback. The fuzzy system had lower chi-squared values and thus gave a more uniform or more "random" spread than did the shift register. The fuzzy system was easier to change and harder to intercept.**

I. Introduction

A FREQUENCY hopping system spreads a transmitted signal's energy across a bandwidth larger than the minimum required for the signal. It spreads the signal when it changes or "hops" the transmission frequency many times per second. The transmitter and receiver must stay synchronized. They must both hop to the same frequency at the same time. If an eavesdropper does not know the frequency sequence, the hopped signal looks like low-intensity noise spread over the entire bandwidth.

Frequency hopping systems use a pseudorandom number (PN) generator to produce a "random" sequence of frequencies. Fig. 1 [2] shows the signal flow. Whatever the code used, both the transmitter and receiver must have a copy of the code or the procedure to generate it.

The most common modulation scheme for frequency hopping is M-ary frequency shift keying or MFSK

$$s_k \in \{0, m-1\}, \qquad s(t) = \sin(f(y_k) + s_k \Delta f). \quad (1)$$

Paper approved by D. P. Taylor, Editor for Signal Design, Modulation, and Detection of the IEEE Communications Society. Manuscript received February 12, 1993; revised May 14, 1993. This paper was presented in part at the Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, April 1, 1993.

The authors are with the Department of Electrical Engineering–Systems, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2564 USA.

IEEE Log Number 9410442.

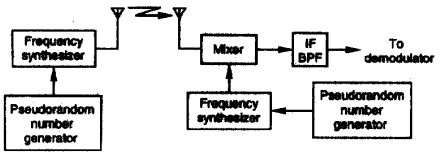


Fig. 1. Spread spectrum communications system that uses frequency hopping.

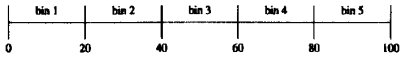


Fig. 2. One hundred frequencies partitioned into five "frequency bins."

s_k is the data sent at time k. y_k is the output of the PN generator in the transmitter and receiver. $s(t)$ is the waveform sent. When $m = 2$, (1) describes frequency shift keying (FSK).

The transmitter's frequency synthesizer emits a frequency $f(y_k)$ that modulates the input. The receiver's frequency synthesizer emits the term $f(y_k) + f_{IF}$. This term translates the received signal to *intermediate frequency* f_{IF}. A bandpass filter, or several BPF's in the case of MFSK, and demodulator choose which value s_k the transmitter sent.

This paper describes a *fuzzy rule-based* PN generator. Adaptive fuzzy rules map distributions of old output frequencies to new output frequencies. These rules, with some initial conditions and the sampling pattern that gives the previous outputs' distribution, fix the output sequence. Each sequence has an unknown length if it repeats at all. Such sequences are hard for an eavesdropper to predict if he does not know either the rules, the initial conditions, or the sampling pattern.

A complete communication system includes more subsystems than the PN generator. Source coding, encryption, channel coding, modulation schemes, and synchronization all affect how the system performs. These subsystems exceed the scope of this paper. We limit our discussion to the design and performance of the PN generator.

II. Fuzzy Pseudorandom Number Generator

Our simulations assumed a frequency synthesizer that gives N distinct frequencies. We partitioned the bandwidth into n *frequency bins* that each contain N/n frequencies. Fig. 2 shows the partitioned spectrum for $N = 100$ and $n = 5$. We numbered the frequencies $0, \cdots, N-1$. So the fuzzy system gave a sequence of integers in $\{0, \cdots, N-1\}$ such that each integer tended to occur equally often.

We chose $N = 100$ frequencies and $n = 5$ bins for initial simulations. After testing with 100 frequencies, we reset N to 1025 frequencies to compare with a shift register. Changing N does not affect the algorithm or the rules. So the system

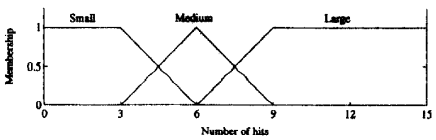


Fig. 3. Input fuzzy sets for each bin.

can use any number of frequencies. N should be a multiple of n, for any n, to give a balanced partition into bins. We chose the small number $n = 5$ to limit the number of rules. The adaptive scheme in Section II-C works for any n.

A. Inputs to the Fuzzy System

A fixed sampling pattern decides which 30 of the previous output frequencies will give the next output. For example, at time k we may sample the output frequencies from times $k-1, k-6, k-8, k-14$, etc. Varying the time delays between samples makes the samples appear independent from one step to the next. The number of samples that fall in the five frequency bins are the five inputs to the fuzzy system. We store them as a length-5 vector x_k.

The decision to use 30 samples followed directly from two other design choices. We first chose five frequency bins to give about 200 rules, as discussed in Section II-C. Then we designed the input fuzzy sets shown in Fig. 3. This process involved some trial and error. The Medium fuzzy set should have its maximum value at the expected number of hits per bin. Since Medium peaks at 6 and there are five bins, the number of samples should be $6 \times 5 = 30$.

A simpler sampling pattern, such as the last 30 output frequencies, would not work. Fuzzy systems tend to map close inputs to close outputs. If the inputs at times k and $k+1$ correlate highly, so will the outputs at times k and $k+1$. In designing the sampling pattern, we tried to minimize the output sequence's autocorrelation and minimize the maximum sample time delay, i.e., the time delay for the thirtieth sample.

For each bin we define three fuzzy sets: Small, Medium, and Large. Fig. 3 shows these fuzzy sets or set membership functions

$$m_S: \quad \{0, \cdots, 30\} \longrightarrow [0,1] \tag{2}$$

$$m_M: \quad \{0, \cdots, 30\} \longrightarrow [0,1] \tag{3}$$

$$m_L: \quad \{0, \cdots, 30\} \longrightarrow [0,1]. \tag{4}$$

$m_S(u)$ stands for the degree to which the number of hits u is Small. $m_M(u)$ and $m_L(u)$ stand for the degree to which u is Medium and Large. So the fuzzy sets in Fig. 3 map the number of hits $x_k[j]$ in each bin into a length-3 *fit vector* [5] $(m_S(x_k[j]), m_M(x_k[j]), m_L(x_k[j]))$ of fit or *fuzzy unit* values. An input value of 6, the expected value per bin, maps to (0, 1, 0), five maps to (1/3, 2/3, 0), and eight becomes (0, 1/3, 2/3). The fit vectors of the five integer inputs form a 5×3 matrix I_k

$$I_k[j,1] = m_S(x_k[j]) \tag{5}$$

$$I_k[j,2] = m_M(x_k[j]) \tag{6}$$

$$I_k[j,3] = m_L(x_k[j]). \tag{7}$$

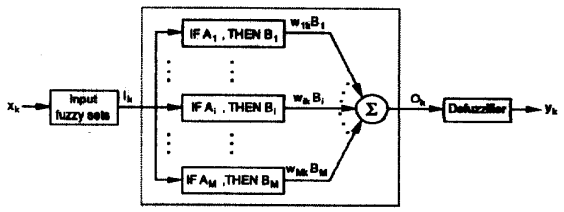


Fig. 4. Parallel algorithmic structure of the fuzzy system. Input x_k fires all rules to some degree since x_k belongs to each A_i to some degree. The output fuzzy set O_k is the weighted sum of the then-part fuzzy sets B_i. The defuzzifier converts O_k to a number y_k, the centroid or center of mass of O_k.

In the general case of n bins and p input fuzzy sets, I_k would have dimensions $n \times p$. I_k is the input to the fuzzy system in Fig. 4.

B. Fuzzy Rules

Each fuzzy rule is a conditional

IF s_k is A_i, THEN output is B_i.

This rule equals the fuzzy patch or Cartesian product $A_i \times B_i$ in the state input-output space $X \times Y$. The fuzzy rules or patches cover the graph of some unknown function $f: X \longrightarrow Y$. The fuzzy system averages patches that overlap and in this way approximates f. A fuzzy system can uniformly approximate any continuous or measurable function [4].

Each A_i is a length-5 vector of fuzzy-set values: Small, Medium, and Large. The antecedent (S, L, L, M, L) might read as

IF ($x_k[1]$ is Small AND $x_k[2]$ is Large AND $x_k[3]$ is Large AND $x_k[4]$ is Medium AND $x_k[5]$ is Large) $\cdots$

Or we can express A_i as a 5×3 matrix $\mathcal{A}_i$ similar to I_k. Each of the five fuzzy-set values in A_i maps to a unit bit vector

$$\begin{aligned} \text{Small} &\longrightarrow (1,0,0) \\ \text{Medium} &\longrightarrow (0,1,0) \\ \text{Large} &\longrightarrow (0,0,1). \end{aligned}$$

So for the above antecedent A_i = (S, L, L, M, L)

$$A_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

At each step the fuzzy sets in Fig. 3 map the five integer input values $x_k[j]$ to their membership or fit values, the degrees to which they belong to the Small, Medium, and Large fuzzy sets. Each conjunct in the antecedent or if-part of a rule takes on one of these fit values. Suppose $x_k[j] = 4$. Then "$x_k[j]$ is Small" has the fit value $m_S(4) = 2/3$. The system "fires" or activates each of the five if-part conjuncts in this way. A fuzzy AND operation takes the minimum of fit values [5]. So the complete antecedent "x_k is A_i" takes on the fit

value

$$m_{A_i}(x_k) = \min_j(m_{A_i[j]}(x_k[j])) \tag{8}$$

$$= \min_j(I_k A_i^T[j,j]). \tag{9}$$

The ith rule *fires* or *activates* to degree w_{ik}

$$w_{ik} = m_{A_i}(x_k). \tag{10}$$

Terms (8) and (9) give different views of a fuzzy rule. (8) includes the input fuzzy sets as part of each rule. So each rule looks at how much a numerical input x_k belongs to fuzzy set A_i in the if-part of the rule. In (9) each conjunct's fit value correlates two fuzzy sets, the jth rows of I_k and $\mathcal{A}_i$. The higher the correlation between I_k and $\mathcal{A}_i$, the higher the firing level w_{ik}. As Fig. 4 shows, firing level w_{ik} scales the then-part B_i of each rule. So (9) implies that each rule is a *fuzzy associative memory* [5] or FAM. The FAM rule (A_i, B_i) or patch $A_i \times B_i$ takes as input the fuzzy set I_k for if-part $\mathcal{A}_i$, the matrix equivalent of A_i, and gives as output the fuzzy set $B_i' = w_{ik}B_i$. If $I_k = \mathcal{A}_i$, then $B_i' = B_i$.

C. Adaptive Rule Generation

The then-part or consequent of each rule is a length-5 fit vector. To learn these consequents, the system trains with 10 000 samples from the International Mathematical and Statistical Library (IMSL) uniform random number generator, scaled and truncated to give integers between 1 and 5. These integers are the bin numbers. The fuzzy system's sampling pattern gives the distribution x_k of previous samples. Then we find the if-part term A_i closest to x_k, the one with maximum w_{ik}, and associate the next sample with A_i.

Example: Observe $x_k = (5, 6, 3, 9, 7)$ and the next output is 4. The closest if-part term is (M, M, S, L, M). So associate output bin 4 with (M, M, S, L, M).

Processing all 10 000 samples in this way gives a histogram of output bin values for each if-part term. Normalizing these histograms gives fit vectors B_i with Hamming norm 1. So the value of the jth component $B_i[j]$ gives the fraction of training samples that fall in bin j when the input distribution matches A_i. Each B_i should be nearly uniform, since the samples are "random" and uniformly distributed in $\{1, \cdots, 5\}$. The B_i closest to (.2, .2, .2, .2, .2) correspond to the A_i that occur most often.

We want a uniform output distribution. Symmetry helps reduce the number of training samples. Each step gives not one but five input-output pairs if we rotate the inputs and outputs. Consider the example above. From the observation we can derive the associations

(M, M, S, L, M) $\longleftrightarrow$ bin 4
(M, M, M, S, L) $\longleftrightarrow$ bin 5
(L, M, M, M, S) $\longleftrightarrow$ bin 1
(S, L, M, M, M) $\longleftrightarrow$ bin 2
(M, S, L, M, M) $\longleftrightarrow$ bin 3.

Thus 10 000 training samples with rotations give about the same results as 50 000 samples with no rotations.

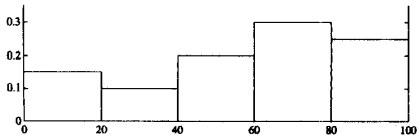


Fig. 5. Combined output fuzzy set $O_k = (.15, .10, .20, .30, .25)$.

In general the spectrum partition in Fig. 2 has n bins with p fuzzy-set values each. So there are n^p possible if-part terms A_i. Each one may give rise to a fuzzy rule. We used $n = 5$ and $p = 3$ for 243 rules maximum. Fifty-seven of the if-part terms cannot have firing levels above 1/3. This adaptive scheme does not associate these if-parts with any output bins. So each fuzzy system can have at most 186 fuzzy rules.

Some rules give better results than do others. A simple screening process checks the covariance of the rules' output fuzzy sets. Rules tend to perform well if their covariance looks like that of white noise. The rules that produced the simulation results in Sections IV and V-A have the following covariance matrix

$$0.0172 \times \begin{bmatrix} 1.000 & -.247 & -.253 & -.253 & -.247 \\ -.247 & 1.000 & -.247 & -.253 & -.253 \\ -.253 & -.247 & 1.000 & -.247 & -.253 \\ -.253 & -.253 & -.247 & 1.000 & -.247 \\ -.247 & -.253 & -.253 & -.247 & 1.000 \end{bmatrix}.$$

D. Combined Output Fuzzy Set

The *combined output fuzzy set* O_k equals the sum of the rule then-parts B_i weighted by their firing levels w_{ik}

$$O_k = \sum_i w_{ik}B_i. \tag{11}$$

This fuzzy inference method is *correlation-product inference* [5].

Only a few rules (32 maximum) can fire at any time. Fig. 4 shows that the system fires all rules in parallel. This gives a fast map from an integer input vector x_k to a combined output fuzzy set O_k.

E. Defuzzification

The combined output fuzzy set O_k *defuzzifies* or maps to a number or output frequency y_k. In additive fuzzy systems O_k tends to be unimodal and easy to defuzzify. The centroid of the output fuzzy set gives a value near the mode. But we designed the fuzzy frequency hopper so that O_k is nearly uniform for each iteration k. The centroid might give a value near the middle of the spectrum each time. So instead we use a new type of centroid defuzzification.

Figs. 5–7 show the extra steps that precede the centroid computation. First find the indices of the largest and smallest terms. Call them $j_{\max}$ and $j_{\min}$. Rotate O_k so that $j_{\max}$ is the center bin. Then the centroid of this fuzzy set will be near the center of bin $j_{\max}$. Next subtract $O_k[j_{\min}]$ from each term. This step subtracts off the uniform part and isolates the "noise" part of the output. Call this fuzzy set $O_{\text{noise},k}$.

We find the centroid $c_{\text{noise},k}$ and then use a previous output frequency, scaled by 1/5, to further modulate the position in

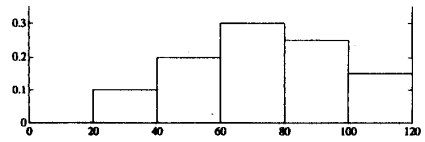


Fig. 6. Rotated O_k for Fig. 5.

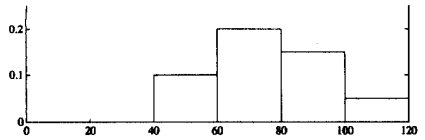


Fig. 7. "Noise" component of O_k in Fig. 5.

the center bin

$$y_k = \lfloor c_{\text{noise},k} - N/2n + y_{k-r}/n \rfloor \bmod N \tag{12}$$

where N is the number of frequencies and $n = 5$ is the number of bins. We used four different values for r and changed the value every 250 iterations. The previous output y_{k-r} looks like a random number but is just an entry in the *deterministic* frequency sequence.

Figs. 5–7 show the elements of the output fuzzy sets as rectangles with height equal to 1. The rectangles span the bins. Triangles, trapezoids, Gaussians, or any other shapes yield the same results if their centroids lie at the centers of the five bins. For correlation-product inference and centroid defuzzification, the shape of the output fuzzy-set elements does not affect the defuzzified result [7].

The centroid of O_k is

$$c_k = \frac{\sum_{j=1}^{5} O_k[j] m_j}{\sum_{j=1}^{5} O_k[j]} \tag{13}$$

where m_j is the center of the jth bin. Rotation changes only the m_j's

$$c_{\text{rotated},k} = \frac{\sum_{j=1}^{5} O_k[j] m'_j}{\sum_{j=1}^{5} O_k[j]} \tag{14}$$

where

$$m'_j = \begin{cases} m_j & \text{if } |j - j_{\max}| \le 2 \\ m_j - N & \text{if } j - j_{\max} > 2 \\ m_j + N & \text{if } j - j_{\max} < -2. \end{cases} \tag{15}$$

Separating O_k's uniform and noise components gives

$$O_k = O_{\text{uniform},k} + O_{\text{noise},k} \tag{16}$$

or

$$O_k[j] = O_k[j_{\min}] + O_{\text{noise},k}[j]. \tag{17}$$

The uniform part has centroid

$$c_{\text{uniform},k} = \frac{\sum_{j=1}^{5} O_k[j_{\min}] m'_j}{\sum_{j=1}^{5} O_k[j_{\min}]} \tag{18}$$

$$= \frac{O_k[j_{\min}] \sum_{j=1}^{5} m'_j}{5 O_k[j_{\min}]} \tag{19}$$

$$= \frac{1}{5} \sum_{j=1}^{5} m'_j. \tag{20}$$

$c_{\text{uniform},k}$ does not depend on O_k. So we drop the subscript k and call this centroid c_{uniform}. If the bins have equal width, c_{uniform} equals the center of the rotated spectrum.

The noise component has centroid

$$c_{\text{noise},k} = \frac{\sum_{j=1}^{5} O_{\text{noise},k}[j] m'_j}{\sum_{j=1}^{5} O_{\text{noise},k}[j]} \tag{21}$$

$$= \frac{c_{\text{rotated},k} - \left(\dfrac{5 O_k[j_{\min}]}{\sum_{j=1}^{5} O_k[j]} \right) c_{\text{uniform}}}{1 - \dfrac{5 O_k[j_{\min}]}{\sum_{j=1}^{5} O_k[j]}} \tag{22}$$

from (17), (14), and (20).

Rewriting (22) gives

$$c_{\text{rotated},k} = \left(\frac{5 O_k[j_{\min}]}{\sum_{j=1}^{5} O_k[j]} \right) c_{\text{uniform}} + \left(1 - \frac{5 O_k[j_{\min}]}{\sum_{j=1}^{5} O_k[j]} \right) c_{\text{noise},k}. \tag{23}$$

The first term in (23) draws $c_{\text{rotated},k}$ toward the center of the rotated spectrum. So $c_{\text{noise},k}$ varies more from the center than $c_{\text{rotated},k}$ does. The difference between them increases as O_k becomes more uniform and as the first term dominates. So using $c_{\text{noise},k}$ instead of $c_{\text{rotated},k}$ in (12) gives better results.

III. Functional Description of the Fuzzy System

The fuzzy system shown in Fig. 4 is a function $f: X \to Y$. For this application, X consists of length-5 vectors of integers, and $Y = [0, N)$, where N is the number of output frequencies. Truncation of the defuzzified output gives integers $y_k \in \{0, \cdots, N-1\}$. Fuzzy sets map inputs to matrices of fuzzy values in [0, 1]—g: $X \to Z$. The fuzzy rules, h_i: $Z \to \phi$, map these fuzzy matrices to output fuzzy sets. Defuzzification maps those output fuzzy sets to output values—d: $\phi \to Y$.

A. Output Fuzzy Sets as Conditional Densities

We can view output fuzzy sets as conditional probability density functions [4]

$$B_i(y) = p_{Y|Z}(y|A_i) \tag{24}$$

where A_i and B_i are the ith rule's if-part and then-part and are both *random sets*. Then the degree to which x_k belongs to A_i is the conditional probability that the input is A_i given the value x_k. So A_i defines the locus of these two-point conditional densities. The antecedent should be $\mathcal{A}_i$, the matrix equivalent of A_i. Since the mapping from A_i to $\mathcal{A}_i$ is a bijection, we just use the term A_i.

A probabilistic view of the firing levels w_{ik} uses the next definition.

Definition:

$$\Pr(A_i|X = x) = \frac{m_{A_i}(x)}{\sum_{A_j \in Z} m_{A_j}(x)} \tag{25}$$

where $m_{A_i}(x)$ stands for the degree to which x belongs to A_i, and A_i is a fuzzy set or vector of fuzzy sets.

(25) and (10) give

$$w_{ik} = \left(\sum_{j=1}^{M} w_{jk}\right) \Pr(A_i|X = x_k). \tag{26}$$

Note that

$$\Pr(A_i) = \int \Pr(A_i|X = x)\, p_X(x)\, dx \tag{27}$$

from Bayes' rule

$$= \int \left[\frac{m_{A_i}(x)}{\sum_{A_j \in Z} m_{A_j}(x)}\right] p_X(x)\, dx \tag{28}$$

$$= E_X\left[\frac{m_{A_i}(x)}{\sum_{A_j \in Z} m_{A_j}(x)}\right], \tag{29}$$

from the definition of the expected value operator, as Zadeh [9] suggested.

The output fuzzy sets B_i do not depend on x_k. So we can rewrite (24) as

$$B_i(y) = p_{Y|ZX}(y|A_i, x_k). \tag{30}$$

Putting (26) and (30) into (11) gives

$$O_k(x_k, y) = \sum_i \left(\sum_j w_{jk}\right) \Pr(A_i|X = x_k) \cdot p_{Y|ZX}(y|A_i, x_k) \tag{31}$$

$$= \left(\sum_j w_{jk}\right) \sum_i p_{YZ|X}(y, A_i|x_k) \tag{32}$$

$$= \left(\sum_j w_{jk}\right) p_{Y|X}(y|x_k). \tag{33}$$

So the fuzzy system uses the conditional densities $p_{Y|Z}(y|A_i)$ to compute $p_{Y|X}(y|x_k)$. Here, $O_k(x_k, y)$ estimates the uniform distribution. In the ideal case each $p_{Y|Z}(y|A_i)$ would be uniform and so would be $p_{Y|X}(y|x_k)$. This result shows that $p_{Y|X}(y|x_k)$ behaves as a uniform distribution. Simulations show that the covariance of $p_{Y|X}(y|x_k)$ is "white" in time. The variance of the additive white noise depends on the covariance of the output fuzzy sets.

B. Centroid Defuzzification

The centroid map acts as a defuzzifier of output set O_k. (33) shows why the centroid tends to perform well. The centroid of $O_k(x_k, y)$ is

$$c_k(x_k) = \frac{\int y O_k(x_k, y)\, dy}{\int O_k(x_k, y)\, dy} \tag{34}$$

$$= E[Y|X = x_k] \tag{35}$$

as derived in the Appendix. The mean-squared optimality properties of the conditional mean [6] suggest the centroid as the preferred way to defuzzify an output set.

IV. Simulation Results

Fig. 8 shows the results for 100 frequencies, averaged over four runs. The standard chi-squared test [1] compares the fuzzy system's output distribution to the desired uniform distribution. We compared this with the IMSL random number generator that gave the sample data to train the fuzzy system. A solid line marks the mean chi-squared values for the fuzzy system. A dotted line shows those for the IMSL random number generator. The lower the chi-squared value, the more closely the output distribution matches the uniform distribution.

Fig. 9 shows the results for the fuzzy system's four runs. Three of the four runs used random initial conditions. The fourth run, shown by the solid line, used all zero initial conditions. This extreme case shows the fuzzy system's robustness to initial conditions.

We also looked at the autocorrelation of sequences to see how predictable they were. In the best case the bin number of the output at time k would not depend on the bin number at time $k - \ell$ for any ℓ. But the time delays in the sampling pattern gave slightly higher correlations than did other values of ℓ. So a clever eavesdropper who could detect the frequency

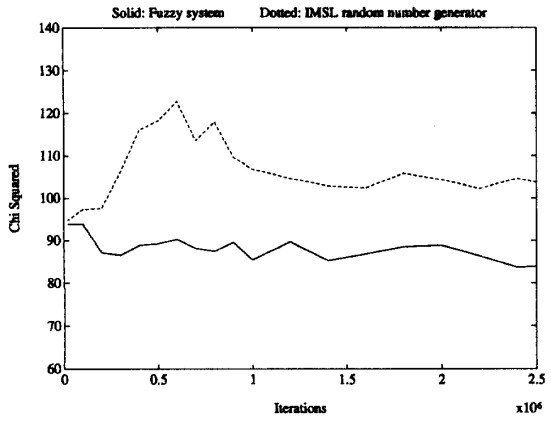


Fig. 8. Mean chi-squared values for the fuzzy system and the IMSL random number generator for 100 frequencies. The lower the chi-squared value, the more closely the output distribution matches the uniform distribution.

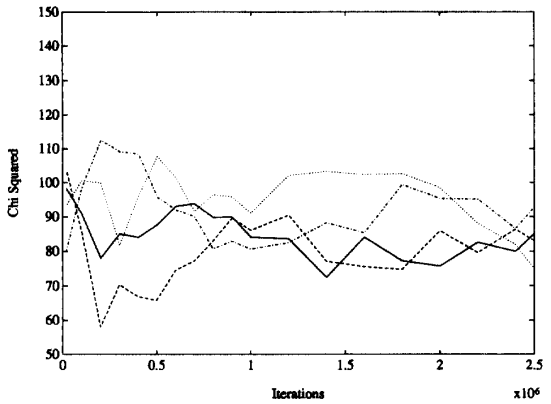


Fig. 9. Chi-squared values for the fuzzy system for 100 frequencies. The solid-line run used all zero initial conditions. The other three used random initial conditions.

sequence for 25 000–50 000 iterations could learn the sampling pattern. But he could not learn the fuzzy rules from just the frequency sequence. He might learn rules that worked as well. But he would not learn the same rules. He also could not find the time offsets r in (12) from the frequency sequence.

V. Comparison with a Shift Register with Linear Feedback

We compared the fuzzy spreader's output with a maximum-length shift-register sequence. The 33-stage shift register with the linear feedback connections in Fig. 10 generated a binary pseudonoise sequence of length $2^{33} - 1$ [8]. We sampled the output every ten clock cycles to get integers in the range $\{0, \cdots, 1023\}$. These integers mapped to the transmission frequencies for a frequency hopping system with 1024 frequencies. In this case the fuzzy spreader used 1025 frequencies. Recall from Section II that the fuzzy system can generate sequences for any multiple of five frequencies.

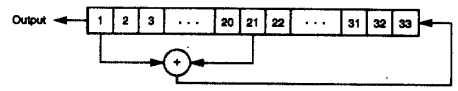


Fig. 10. Shift register with linear feedback.

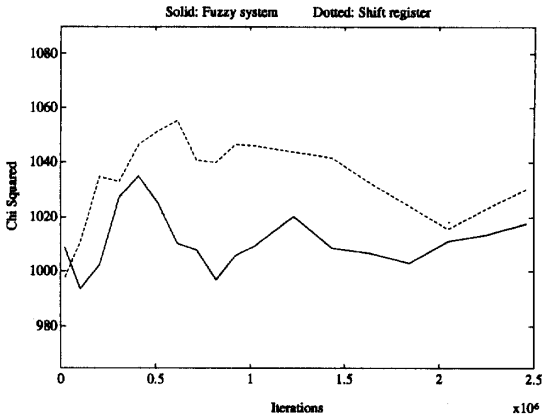


Fig. 11. Mean chi-squared values for the fuzzy system for 1025 frequencies and the shift register for 1024 frequencies.

A. Simulation Results

We performed a chi-squared test of both systems. Again lower chi-squared values indicated more uniform sequences. Fig. 11 shows the results averaged over five runs. The solid line marks the mean chi-squared values for the fuzzy system with 1025 frequencies. The dotted line shows those for the shift register with 1024 frequencies. Results for each run vary. On average the fuzzy system tended to give a more uniform sequence than did the shift register.

B. System Comparison

The fuzzy spreader and the shift register differ both in structure and in function. A shift register with m stages maps m *information bits*—the initial conditions—to a codeword with $2^m - 1i$ b. During the $2^m - 1$ clock cycles required to produce the output sequence, the shift register takes on every state except the all-zero state. So the initial conditions just choose which cyclic shift of the single codeword the system will generate.

The shift register's linear structure gives fast operation and aids algebraic analysis. But it leaves a frequency hopping system open to interception. An eavesdropper who can detect $2m + 1$ consecutive bits in the output sequence can build a code generator to produce the same sequence [2]. For our length-33 shift register an eavesdropper would need to detect $2 \times 33 + 1 = 67$ consecutive bits, or seven frequencies. To combat this weakness code designers often combine output sequences in nonlinear ways from several stages of the shift register or from multiple shift registers [3].

The above fuzzy system is nonlinear. At each step it estimates the uniform distribution conditioned on prior sampled outputs. Unless an eavesdropper knows the fuzzy rules driving the system, he cannot predict the output sequence. And the lack of algebraic structure and adaptive rule generation give

a sequence of unknown length. The shift-register sequence repeats every $2^m - 1$ b.

The shift register also limits the number of frequencies to a power of 2. The fuzzy system's rule structure works the same for any number of frequencies. We chose a multiple of 5 to allow an equal partition into five bins. We can relax this constraint as the number of frequencies grows.

VI. Conclusion

The fuzzy spreader is deterministic and suggests that we do not need "randomness" to securely spread and despread a signal. The adaptive scheme captures the autocorrelation structure of a uniform random number generator in a set of fuzzy rules. The fuzzy system computes the density conditioned on the current input. Centroid defuzzification computes the conditional mean. In our case it gives the next entry in the output sequence. The distribution of this sequence is at least as uniform as the distribution of the system that produced the training samples. The fuzzy system's output sequence can frequency hop a signal and offers advantages over the maximum-length shift-register sequences often used to spread a signal. The fuzzy system produces a sequence that is more uniform, harder to intercept, and easier to spread over any number of frequencies without changing the algorithm or the fuzzy rules.

Appendix
Derivation of the Centroid as the Conditional Mean

The centroid of $O_k(x_k, y)$ is

$$c_k(x_k) = \frac{\int y O_k(x_k, y)\, dy}{\int O_k(x_k, y)\, dy}. \tag{36}$$

$$\int y O_k(x_k, y)\, dy = \int y \left(\sum_j w_{jk} \right) p(y|x_k)\, dy \tag{37}$$

$$= \left(\sum_j w_{jk} \right) \int y p(y|x_k)\, dy \tag{38}$$

$$= \left(\sum_j w_{jk} \right) E[Y|X = x_k] \tag{39}$$

$$\int O_k(x_k, y)\, dy = \int \left(\sum_j w_{jk} \right) p(y|x_k)\, dy \tag{40}$$

$$= \left(\sum_j w_{jk} \right) \int p(y|x_k)\, dy \tag{41}$$

$$= \sum_j w_{jk}. \tag{42}$$

Putting (39) and (42) into (36) gives

$$c_k(x_k) = E[Y|X = x_k] \tag{43}$$

in agreement with Kosko [4].

References

[1] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, 3rd ed. Pacific Grove, CA: Brooks/Cole, 1991.

[2] R. C. Dixon, *Spread Spectrum Systems*, 2nd ed. New York: Wiley-Interscience, 1984.

[3] S. W. Golomb, *Shift Register Sequences*. San Francisco: Holden-Day, 1967.

[4] B. Kosko, "Fuzzy systems as universal aproximators," *IEEE Trans. Comput.*, vol. 42, no. 11, pp. 1329–1333, Nov. 1994.

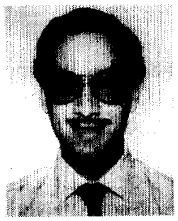
[5] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[6] J. M. Mendel, *Lessons in Digital Estimation Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[7] P. J. Pacini and B. Kosko, "Adaptive fuzzy system for target tracking," *Intelligent Systems Engineering*, vol. 1, no. 1, pp. 3–21, Autumn 1992.

[8] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1983.

[9] L. A. Zadeh, "Probability measures of fuzzy events," *J. Math. Anal., Applicat.*, pp. 421–427, 1968.



Peter J. Pacini (S'87–M'89) was born in Bakersfield, CA, on April 2, 1967. He received the B.S.E.E., M.S.E.E., and Ph.D. degrees from the University of Southern California, Los Angeles, in 1989, 1990, and 1993, respectively.

He was an Office of Naval Research Fellow from 1989–1993. His research interests include adaptive and nonadaptive fuzzy systems for communication and control.

Dr. Pacini is a member of Eta Kappa Nu, Tau Beta Pi, Phi Beta Kappa, and Phi Kappa Phi.

Bart Kosko (M'85) received the B.S. degrees in philosophy and economics from the University of Southern California, Los Angeles, the M.S. degree in applied math from the University of California, San Diego, and the Ph.D. degree in electrical engineering from the University of California, Irvine.

He is an Assistant Professor in the Department of Electrical Engineering-Systems at the University of Southern California, Los Angeles. He edits the book series *Lecture Notes on Neural Computing* (Springer-Verlag) and is author of *Neural Networks and Fuzzy Systems* and *Neural Networks for Signal Processing* (Prentice-Hall) and *Fuzzy Thinking* (Disney/Hyperion).

Dr. Kosko is a Governor of the International Neural Network Society and is on the Executive Advisory Board of the IEEE Transactions on Fuzzy Systems. He serves as an Associate Editor of *Neural Networks* and other journals. He was a program cochair of the 1993 World Congress on Neural Networks and the 1992 International Conference on Fuzzy Logic and Neural Networks (Iizuka-92) in Japan.