Chapter 6 The Ambiguity Function - Discrete Coded Waveforms

The concepts of resolution and ambiguity were introduced in Chapter 4. The relationship between the waveform resolution (range and Doppler) and its corresponding ambiguity function was discussed and analyzed. It was determined that the *goodness* of a given waveform is based on its range and Doppler resolutions, which can be analyzed in the context of the ambiguity function. For this purpose, a few common analog radar waveforms were analyzed in Chapter 5. In this chapter, another type of radar waveform based on discrete codes is introduced. This topic has been and continues to be a major research thrust area for many radar scientist, designers, and engineers. Discrete coded waveforms are more effective in improving range characteristics than Doppler (velocity) characteristics. Furthermore, in some radar applications, discrete coded waveforms are heavily favored because of their inherent anti-jamming capabilities. In this chapter, a quick overview of discrete coded waveforms is presented. Three classes of discrete codes are analyzed. They are unmodulated pulse-train codes (uniform and staggered), phase-modulated (binary or polyphase) codes, and frequency modulated codes.

6.1. Discrete Code Signal Representation

The general form for a discrete coded signal can be written as

$$x(t) = e^{j\omega_0 t}\sum_{n=1}^{N} u_n(t) = e^{j\omega_0 t}\sum_{n=1}^{N} P_n(t)e^{j(\omega_n t+\theta_n)} \tag{6.1}$$

where ω_0 is the carrier frequency in radians, (ω_n, θ_n) are constants, N is the code length (number of bits in the code), and the signal $P_n(t)$ is given by

$$P_n(t) = a_n Rect\left(\frac{t}{\tau_0}\right) \tag{6.2}$$

the constant a_n is either (1) or (0), and

$$Rect\left(\frac{t}{\tau_0}\right) = \begin{cases} 1 & ; \quad 0 < t < \tau_0 \\ 0 & ; \ elsewhere \end{cases} \tag{6.3}$$

Using this notation the discrete code can be described through the sequence

$$U[n] = \{u_n \ , n = 1, 2, \ldots, N\} \tag{6.4}$$

which, in general, is a complex sequence depending on the values of ω_n and θ_n. The sequence $U[n]$ is called the code and for convenience it will be denoted by U.

In general, the output of the matched filter is

$$\chi(\tau, f_d) = \int_{-\infty}^{\infty} x^*(t)x(t+\tau)e^{-j2\pi f_d t}dt \tag{6.5}$$

Substituting Eq. (6.1) into Eq. (6.5) yields

$$\chi(\tau, f_d) = \sum_{n=1}^{N}\sum_{k=1}^{N}\int_{-\infty}^{\infty} u_n^*(t)u_k(t+\tau)e^{-j2\pi f_d t}dt \tag{6.6}$$

Depending on the choice of combination for a_n, ω_n, and θ_n, different class of codes can be generated. More precisely, pulse-train codes are generated when

$$\theta_n = \omega_n = 0 \quad ; \ and \ \ a_n = 1, or \ \ 0 \tag{6.7}$$

Binary phase codes and polyphase codes are generated when

$$\omega_n = 0 \quad ; \ and \ \ a_n = 1 \tag{6.8}$$

Finally, frequency codes are generated when

$$\theta_n = 0 \quad ; \ and \ \ a_n = 1, or \ \ 0 \tag{6.9}$$

6.2. Pulse-Train Codes

The idea behind this class of code is to divide a relatively long pulse of length T_P into N subpulses, each being a rectangular pulse with pulsewidth τ_0 and amplitude of 1 or 0. It follows that the code U is the sequence of 1's and 0's. More precisely, the signal representing this class of code can written as

$$x(t) = e^{j\omega_0 t}\sum_{n=1}^{N} P_n(t) = e^{j\omega_0 t}\sum_{n=1}^{N} a_n Rect\left(\frac{t}{\tau_0}\right) \tag{6.10}$$

One way to generate a train-pulse class code can be by setting

$$a_n = \begin{cases} 1 & n-1 = 0 \ \ modulu \ \ q \\ 0 & n-1 \neq 0 \ \ modulu \ \ q \end{cases} \tag{6.11}$$

where q is a positive integer that divides evenly into $N-1$. That is,

$$M-1 = (N-1)/q \tag{6.12}$$

where M is the number of 1's in the code. For example, when $N = 21$ and $q = 5$, then $M = 5$, and the resulting code is

$$\{U\} = \{10000 \ \ 10000 \ \ 10000 \ \ 10000 \ \ 1\} \tag{6.13}$$

This is illustrated in Fig. 6.1. In previous chapters this code would have been represented by the following continuous time domain signal

$$x_1(t) = e^{j\omega_0 t} \sum_{m=0}^{4} Rect\left(\frac{t-mT}{\tau_0}\right) \tag{6.14}$$

where the period is $T = 5\tau_0$. Using this analogy yields

$$\frac{T_p}{M-1} \equiv T \tag{6.15}$$

and Eq. (6.10) can now be written as

$$x(t) = e^{j\omega_0 t} \sum_{m=1}^{M-1} Rect\left(\frac{t - m\left(\frac{T_p}{M-1}\right)}{\tau_0}\right) \tag{6.16}$$

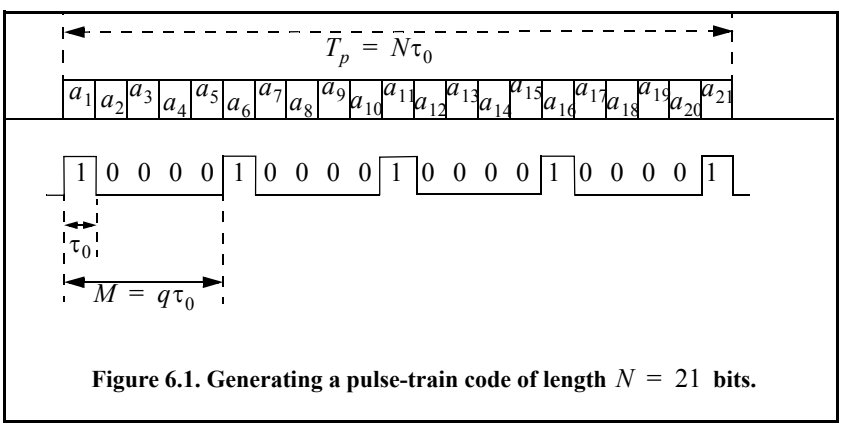

Figure 6.1. Generating a pulse-train code of length $N = 21$ bits.

In Chapter 5 (Section 5.2.3) an expression for the ambiguity function for a coherent train of pulses was derived. Comparison of Eq. (6.16) and Eq. (5.37) show that the two equations are equivalent when the condition in Eq. (6.15) is true except for the ratio $(1/\sqrt{N})$. It follows that the ambiguity function for the signal defined in Eq. (6.16) is

$$|\chi(\tau;f_d)| = \sum_{k=-M}^{M} \left| \frac{\sin\left[\pi f_d \left([M-|k|]\frac{T_p}{M-1}\right)\right]}{\sin\left(\pi f_d \frac{T_p}{M-1}\right)} \right| \left| \frac{\sin\left[\pi f_d \left(\tau_0 - \left|\tau - \frac{kT_p}{M-1}\right|\right)\right]}{\pi f_d} \right| \quad \textbf{(6.17)}$$

The zero Doppler and zero delay cuts of the ambiguity function are derived from Eq. (6.17). They are given by

$$|\chi(\tau;0)| = M\tau_0 \sum_{k=-M}^{M} \left[1 - \frac{|k|}{M}\right] \left(1 - \frac{\left|\tau - \frac{kT_p}{M-1}\right|}{\tau_0}\right) \quad \textbf{(6.18)}$$

$$|\chi(0;f_d)| = \sum_{k=-M}^{M} \left| \frac{\sin\left[\pi M f_d \left(\frac{T_p}{M-1}\right)\right]}{\sin\left(\pi f_d \frac{T_p}{M-1}\right)} \right| \left| \frac{\sin(\pi f_d \tau_0)}{\pi f_d \tau_0} \right| \quad \textbf{(6.19)}$$

Figure 6.2a shows the three-dimensional ambiguity plot for the code shown in Fig. 6.1, while Fig. 6.2b shows the corresponding contour plot. This figure can be reproduced using the following MATLAB code.

```
close all; clear all;
U = [1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1];
ambiguity = ambiguity_code(U);
```

A cartoon showing contour cuts of the ambiguity function for a pulse-train code is shown in Fig. 6.2c. Clearly, the width of the ambiguity function main lobe (i.e., resolution) is directly tied to the code length. As one would expect, longer codes will produce narrower main lobe and thus have better resolution than shorter ones. Further observation of Fig. 6.2 shows that this ambiguity function has strong grating lobe structure along with high sidelobe levels. The presence of such strong lobing structure limits the effectiveness of the code and will cause detection ambiguities. These lobes are a direct result from the uniform equal spacing between the 1's within a code (i.e., periodicity of the code). These lobes can be significantly reduced by getting rid of the periodic structure of the code, i.e., placing the pulses at nonuniform spacing. This is called code staggering (PRF staggering).

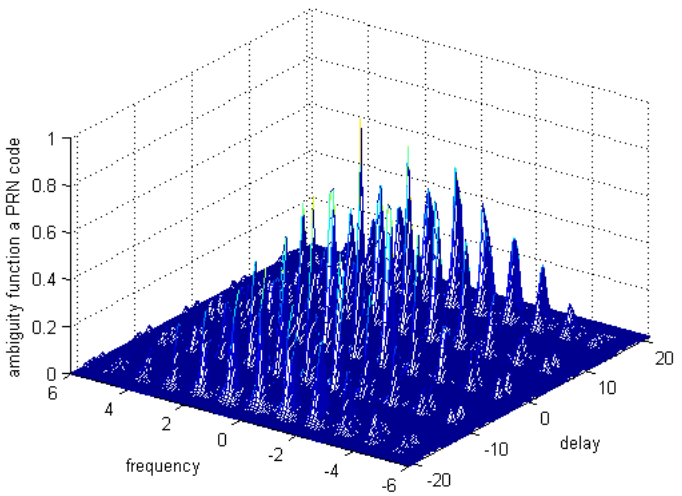

Figure 6.2a. Ambiguity function for the pulse-train code shown in Fig. 6.1.

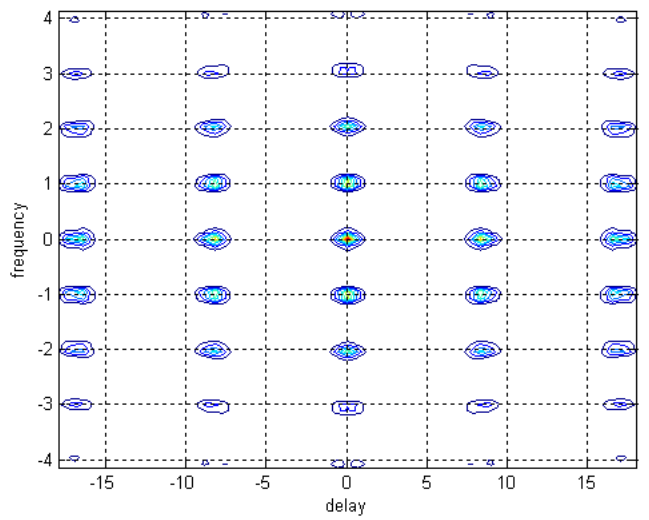

Figure 6.2b. Contour plot corresponding to Fig. 6.2a.

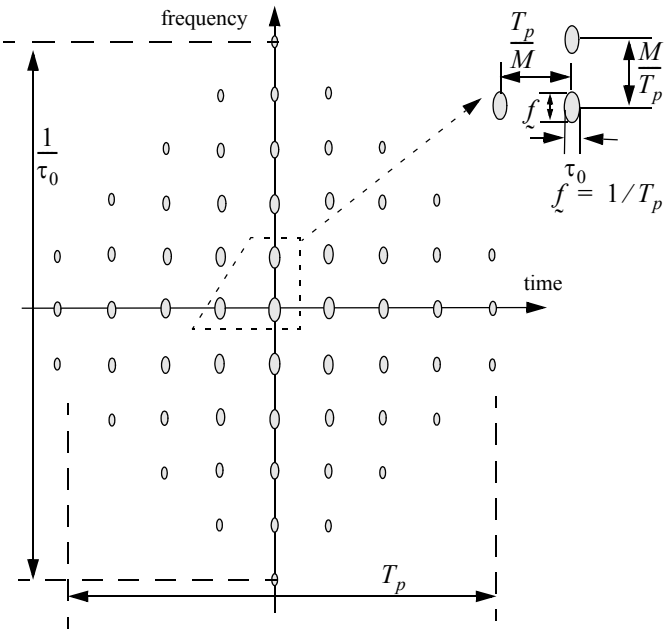

Figure 6.2c. Cartoon of the ambiguity contour plot for a pulse-train code.

For example, consider a pulse-train code of length $N = 21$. A staggered train-pulse code can then be obtained by using the following sequence a_n

$$\{a_n\} = 1 \qquad n = 1, 4, 6, 12, 15, 21 \tag{6.20}$$

Thus, the resulting code is

$$\{U\} = \{100101000001001000001\} \tag{6.21}$$

Figure 6.3 shows the ambiguity plot corresponding to this code. As indicated by Fig. 6.3 the ambiguity function corresponding to a staggered pulse-train code approaches a thumb-tack shape. The choice of the optimum staggered code has been researched extensively by numerous people. Resnick[1] defined the optimum staggered pulse-train code as that whose ambiguity function has absolutely uniform sidelobe levels that are equal to unity. Other researchers, have introduced different definitions for optimum staggering, none of which is necessarily better than the others, except when considered for the particular application being analyzed by the respective researcher.

1. Resnick, J. B., *High Resolution Waveforms Suitable for a Multiple Target Environment*, MS Thesis, MIT, Cambridge, MA, June 1962.

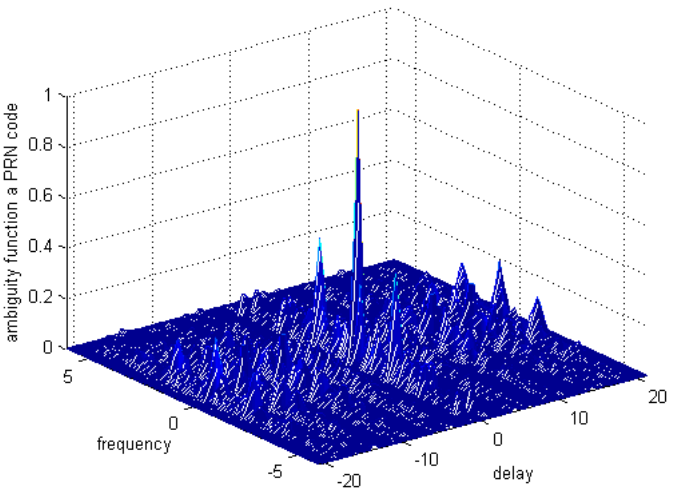

Figure 6.3a. Ambiguity function for the pulse-train code in Eq. (6.21).

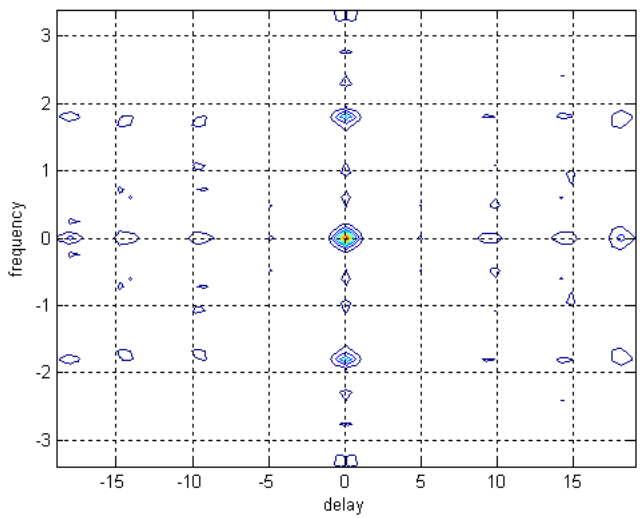

Figure 6.3b. Contour plot corresponding to Fig. 6.3a.

6.3. Phase Coding

The signal corresponding to this class of code is obtained from Eq. (6.1) by letting $\omega_n = 0$. It follows that

$$x(t) = e^{j\omega_0 t}\sum_{n=1}^{N} u_n(t) = e^{j\omega_0 t}\sum_{n=1}^{N} P_n(t)e^{j\theta_n} \tag{6.22}$$

Two subclasses of phase codes are analyzed. They are binary phase codes and polyphase codes.

6.3.1. Binary Phase Codes

In this case, the phase θ_n is set equal to either (0) or (π), and hence, the term *binary* is used. For this purpose, define the coefficient D_n as

$$D_n = e^{j\theta_n} = \pm 1 \tag{6.23}$$

The ambiguity function for this class of code is derived by substituting Eq. (6.22) into Eq. (6.5). The resulting ambiguity function is given by

$$\chi(\tau;f_d) = \begin{cases} \chi_0(\tau',f_d)\displaystyle\sum_{n=1}^{N-k} D_n D_{n+k} e^{-j2\pi f_d(n-1)\tau_0} + \\ \chi_0(\tau_0-\tau',f_d)\displaystyle\sum_{n=1}^{N-(k+1)} D_n D_{n+k+1} e^{-j2\pi f_d n\tau_0} \end{cases} \qquad 0 < \tau < N\tau_0 \tag{6.24}$$

where

$$\tau = k\tau_0 + \tau' \qquad \begin{cases} 0 < \tau' < \tau_0 \\ k = 0, 1, 2, \ldots, N \end{cases} \tag{6.25}$$

$$\chi_0(\tau',f_d) = \int_0^{\tau_0-\tau'} \exp(-j2\pi f_d t)dt \qquad 0 < \tau' < \tau_0 \tag{6.26}$$

The corresponding zero Doppler cut is then given by

$$\chi(\tau;0) = \tau_0\left(1-\frac{|\tau'|}{\tau_0}\right)\sum_{n=1}^{N-|k|} D_n D_{n+k} + |\tau'|\sum_{n=1}^{N-|k+1|} D_n D_{n+k+1} \tag{6.27}$$

and when $\tau' = 0$ then

$$\chi(k;0) = \tau_0 \sum_{n=1}^{N-|k|} D_n D_{n+k} \tag{6.28}$$

Barker Codes

In this case, a long pulse of width T_p is divided into N smaller pulses; each is of width $\tau_0 = T_p/N$. Then, the phase of each subpulse is chosen as either 0 or π radians relative to some code. It is customary to characterize a subpulse that has 0 phase (amplitude of +1 Volt) as either "1" or "+." Alternatively, a subpulse with phase equal to π (amplitude of -1 Volt) is characterized by either "0" or "-." Barker code is optimum in accordance with the definition set by Resnick. Figure 6.4 illustrates this concept for a Barker code of length seven. A Barker code of length N is denoted as B_N. There are only seven known Barker codes that share this unique property; they are listed in Table 6.1. Note that B_2 and B_4 have complementary forms that have the same characteristics.

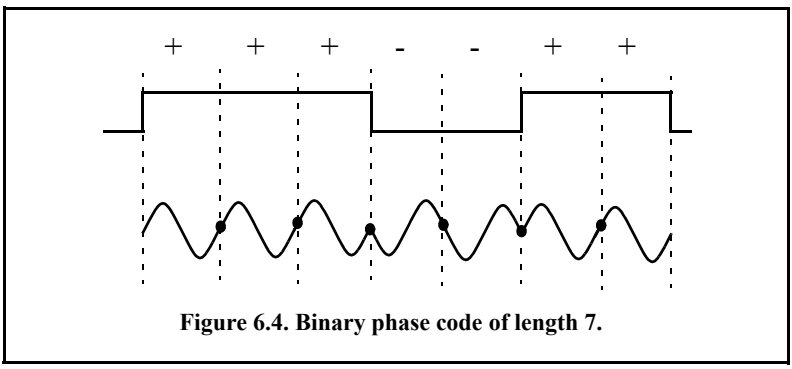

Figure 6.4. Binary phase code of length 7.

In general, the autocorrelation function (which is an approximation for the matched filter output) for a B_N Barker code will be $2N\tau_0$ wide. The main lobe is $2\tau_0$ wide; the peak value is equal to N. There are $(N-1)/2$ side-lobes on either side of the main lobe; this is illustrated in Fig. 6.5 for a B_{13}. Notice that the main lobe is equal to 13, while all side-lobes are unity.

The most side-lobe reduction offered by a Barker code is $-22.3dB$, which may not be sufficient for the desired radar application. However, Barker codes can be combined to generate much longer codes. In this case, a B_M code can be used within a B_N code (M within N) to generate a code of length MN. The compression ratio for the combined B_{MN} code is equal to MN. As an example, a combined B_{54} is given by

$$B_{54} = \{11101, 11101, 00010, 11101\} \tag{6.29}$$

and is illustrated in Fig. 6.6. Unfortunately, the side-lobes of a combined Barker code autocorrelation function are no longer equal to unity. Some side-lobes of a combined Barker code autocorrelation function can be reduced to zero if the matched filter is followed by a linear transversal filter with impulse response given by

$$h(t) = \sum_{k=-N}^{N} \beta_k \delta(t - 2k\tau_0) \tag{6.30}$$

where N is the filter's order, the coefficients β_k ($\beta_k = \beta_{-k}$) are to be determined, $\delta(\cdot)$ is the delta function, and τ_0 is the Barker code subpulse width. A filter of order N produces N zero side-lobes on either side of the main lobe. The main lobe amplitude and width do not change, as illustrated in Fig. 6.7.

TABLE 6.1. Barker codes

Code Symbol	Code Length	Code Elements	Side Lode Reduction (dB)
B_2	2	+- ++	6.0
B_3	3	++-	9.5
B_4	4	++-+ +++-	12.0
B_5	5	+++-+	14.0
B_7	7	+++--+-	16.9
B_{11}	11	+++---+--+-	20.8
B_{13}	13	+++++--++-+-+	22.3

In order to illustrate this approach, consider the case where the input to the matched filter is B_{11}, and assume $N = 4$. The autocorrelation for a B_{11} is

$$\phi_{11} = \{-1, 0, -1, 0, -1, 0, -1, 0, -1, 0, 11, \\ 0, -1, 0, -1, 0, -1, 0, -1, 0, -1\} \tag{6.31}$$

The output of the transversal filter is the discrete convolution between its impulse response and the sequence ϕ_{11}. At this point we need to compute the coefficients β_k that guarantee the desired filter output (i.e., unchanged main lobe and four zero side-lobe levels).

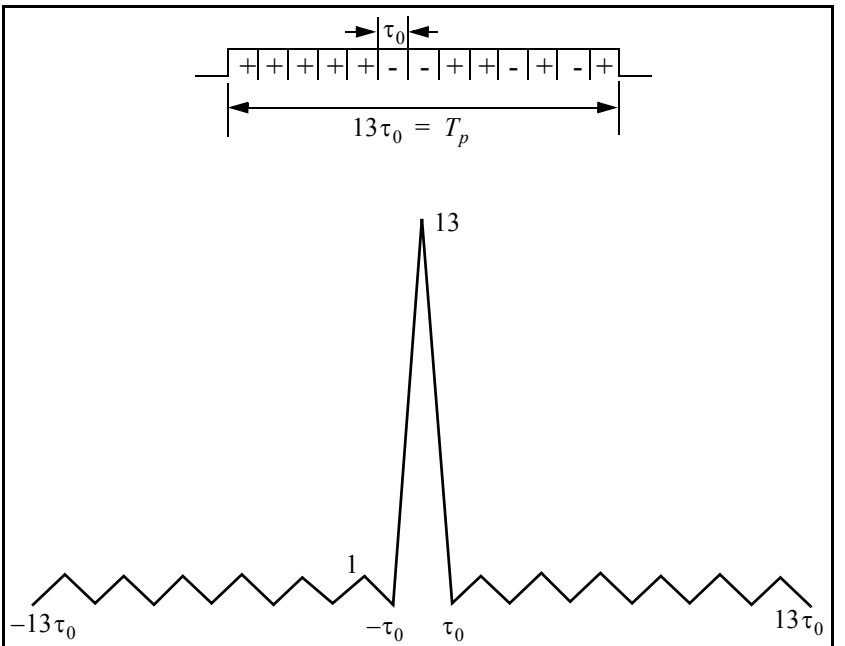

Figure 6.5. Barker code of length 13, and its corresponding auto-correlation function.

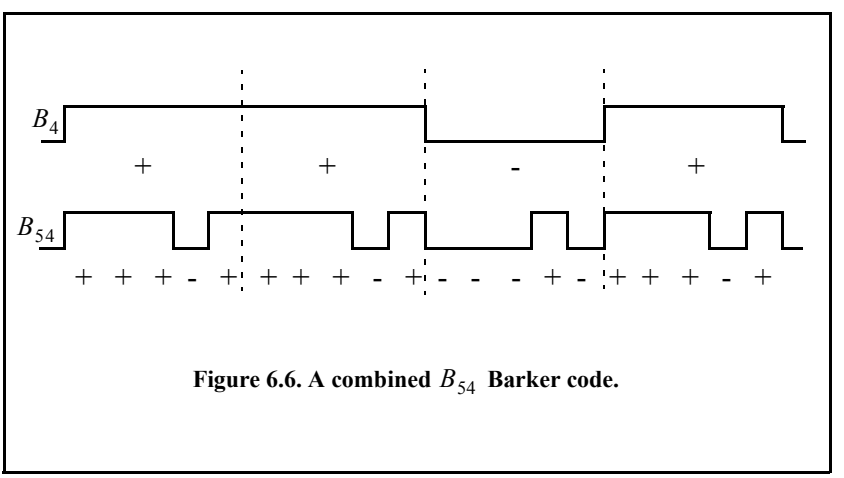

Figure 6.6. A combined B_{54} Barker code.

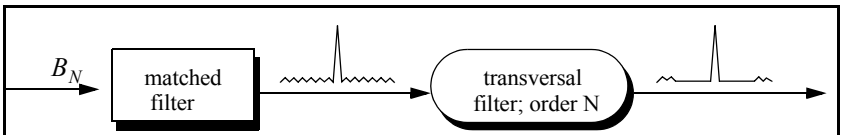

Figure 6.7. A linear transversal filter of order N can be used to produce N zero side-lobes in the autocorrelation function (N = 4).

Performing the discrete convolution as defined in Eq. (6.30) and collecting equal terms ($\beta_k = \beta_{-k}$) yield the following set of five linearly independent equations:

$$\begin{bmatrix} 11 & -2 & -2 & -2 & -2 \\ -1 & 10 & -2 & -2 & -1 \\ -1 & -2 & 10 & -2 & -1 \\ -1 & -2 & -1 & 11 & -1 \\ -1 & -1 & -1 & -1 & 11 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} 11 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.32}$$

Solving Eq. (6.32) yields

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} 1.1342 \\ 0.2046 \\ 0.2046 \\ 0.1731 \\ 0.1560 \end{bmatrix} \tag{6.33}$$

Note that setting the first equation equal to 11 and all other equations to 0 and then solving for β_k guarantees that the main peak remains unchanged, and that the next four side-lobes are zeros. So far we have assumed that coded pulses have rectangular shapes. Using other pulses of other shapes, such as Gaussian, may produce better side-lobe reduction and a larger compression ratio.

Figure 6.8 shows the output of this function when B_{13} is used as an input. Figure 6.9 is similar to Fig. 6.8, except in this case B_7 is used as an input. Figure 6.10 shows the ambiguity function, the zero Doppler cut, and the contour plot for the combined Barker code defined in Fig. 6.6.

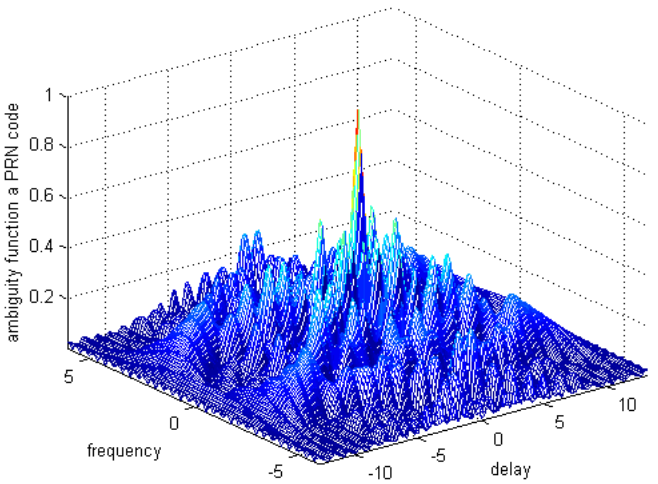

Figure 6.8a. Ambiguity function for B_{13} Barker code.

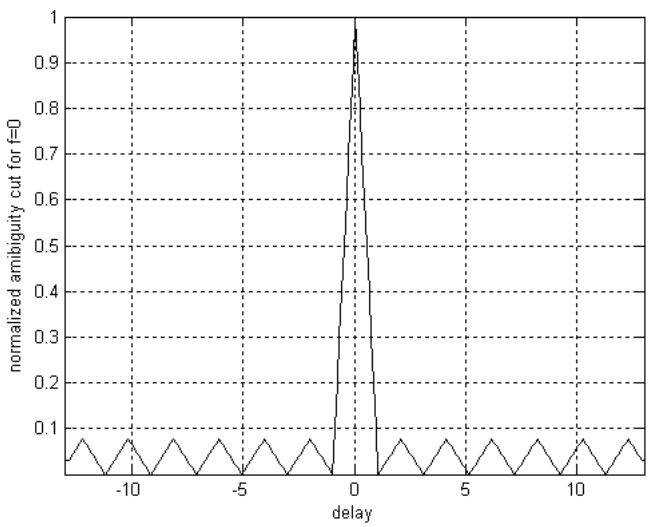

Figure 6.8b. Zero Doppler cut for the B_{13} ambiguity function.

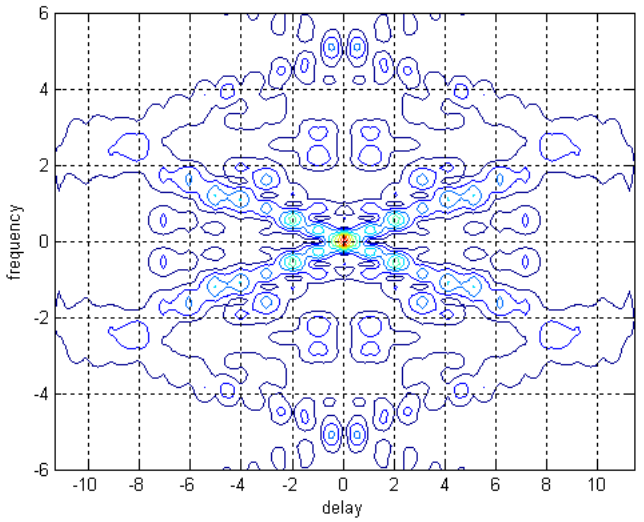

Figure 6.8c. Contour plot corresponding to Fig. 6.8a.

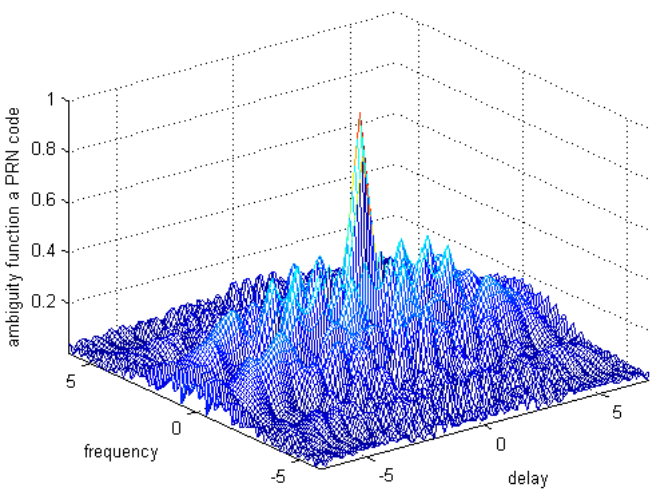

Figure 6.9a. Ambiguity function for B_7 Barker code.

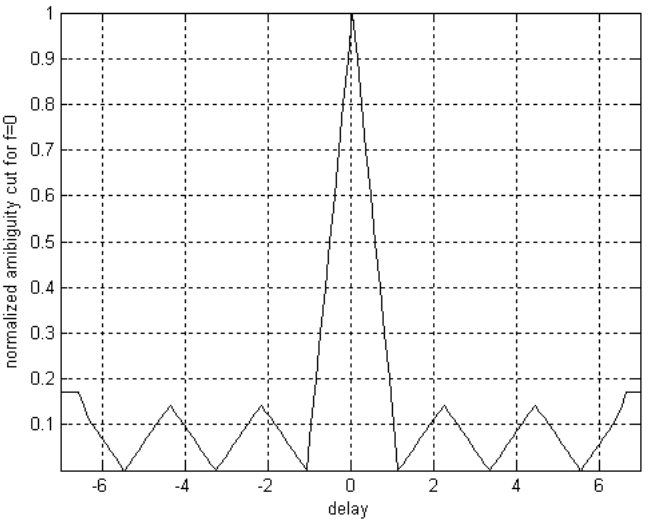

Figure 6.9b. Zero Doppler cut for the B_7 ambiguity function.

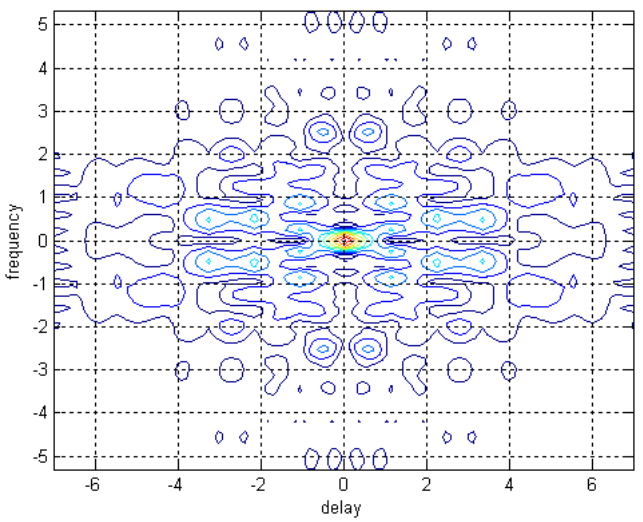

Figure 6.9c. Contour plot corresponding to Fig. 6.9a.

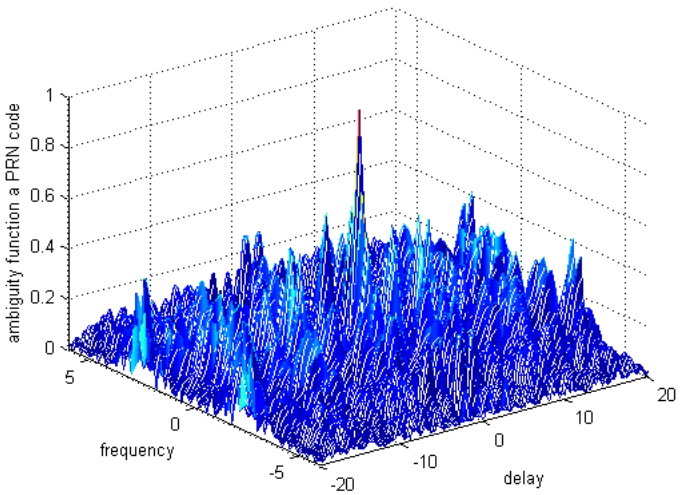

Figure 6.10a. Ambiguity function for B_{54} Barker code.

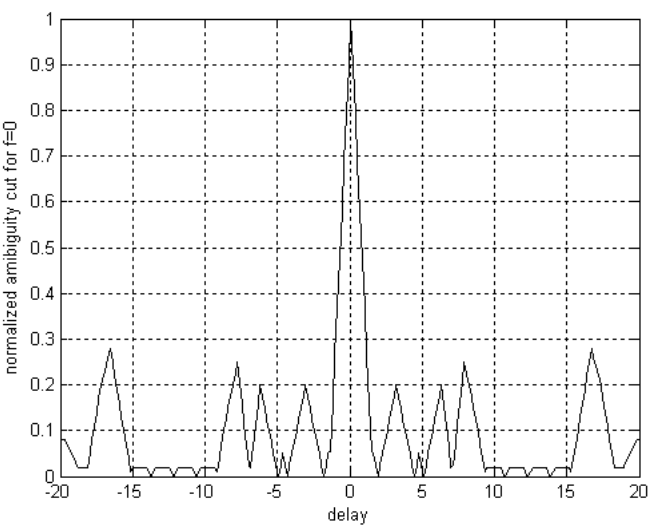

Figure 6.10b. Zero Doppler cut for the B_{54} ambiguity function.

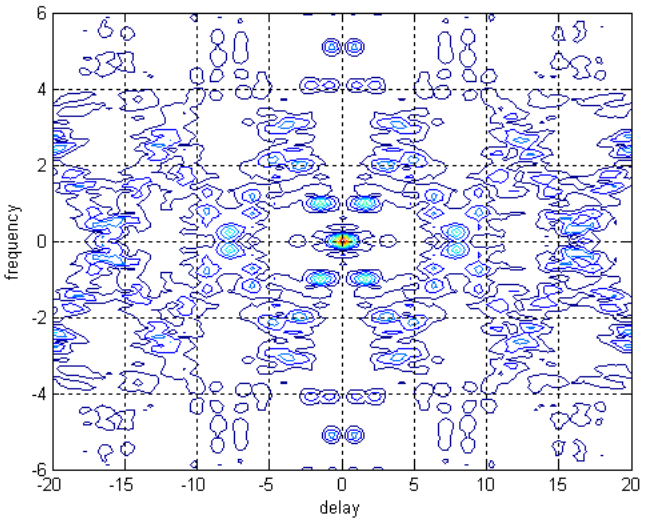

Figure 6.10c. Contour plot corresponding to Fig. 6.10a.

Pseudo-Random Number (PRN) Codes

Pseudo-Random Number (PRN) codes are also known as Maximal Length Sequences (MLS) codes. These codes are called pseudo-random because the statistics associated with their occurrence are similar to those associated with the coin-toss sequences. Maximum length sequences are periodic. The MLS codes have the following distinctive properties:

1. The number of ones per period is one more than the number of minus ones.
2. Half the runs (consecutive states of the same kind) are of length one and one fourth are of length two.
3. Every maximal length sequence has the "shift and add" property. This means that, if a maximal length sequence is added (modulo 2) to a shifted version of itself, then the resulting sequence is a shifted version of the original sequence.
4. Every n-tuple of the code appears once and only once in one period of the sequence.
5. The correlation function is periodic and is given by

$$\phi(n) = \begin{cases} L & n = 0, \pm L, \pm 2L, \ldots \\ -1 & elsewhere \end{cases} \quad \textbf{(6.34)}$$

Figure 6.11 shows a typical sketch for an MLS autocorrelation function. Clearly these codes have the advantage that the compression ratio becomes very large as the period is increased. Additionally, adjacent peaks (grating lobes) become farther apart.

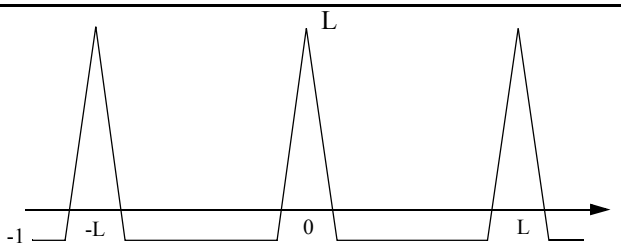

Figure 6.11. Typical autocorrelation of an MLS code of length L.

Linear Shift Register Generators

There are numerous ways to generate MLS codes. The most common is to use linear shift registers. When the binary sequence generated using a shift register implementation is periodic and has maximal length, it is referred to as an MLS binary sequence with period L, where

$$L = 2^n - 1 \quad \textbf{(6.35)}$$

n is the number of stages in the shift register generator. A linear shift register generator basically consists of a shift register with modulo-two adders added to it. The adders can be connected to various stages of the register, as illustrated in Fig. 6.12 for $n = 4$ (i.e., $L = 15$). Note that the shift register initial state cannot be 0.

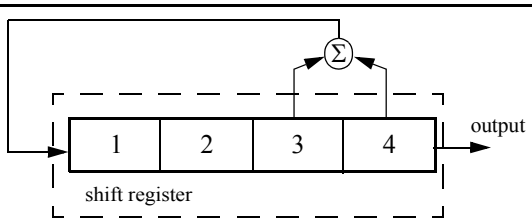

Figure 6.12. Circuit for generating an MLS sequence of length $L = 15$.

The feedback connections associated with a shift register generator determine whether the output sequence will be maximal. For a given size shift register, only a few feedback connections lead to maximal sequence outputs. In order to illustrate this concept, consider the two 5-stage shift register generators shown in Fig. 6.13. The shift register generator shown in Fig. 6.13 a generates a maximal length sequence, as clearly depicted by its state diagram. However, the shift register generator shown in Fig. 6.13 b produces three non-maximal length sequences (depending on the initial state).

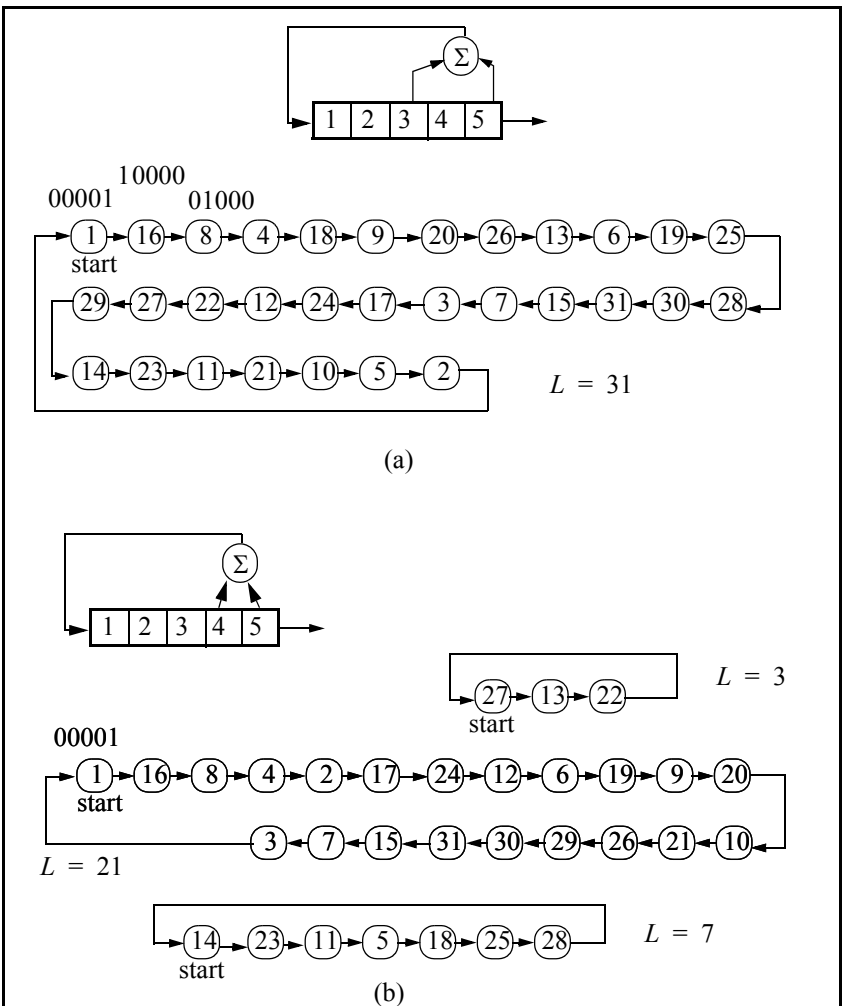

Figure 6.13. (a) A 5-stage shift register generator. (b) Non-maximal length 5-stage shift register generator.

Given an n-stage shift register generator, one would be interested in knowing how many feedback connections will yield maximal length sequences. Zierler[1] showed that the number of maximal length sequences possible for a given n-stage linear shift register generator is given by

$$N_L = \frac{\varphi(2^n - 1)}{n} \tag{6.36}$$

φ is the Euler's totient (Euler's phi) function and is defined by

$$\varphi(k) = k \prod_i \frac{(p_i - 1)}{p_i} \tag{6.37}$$

where p_i are the prime factors of k. Note that when p_i has multiples, only one of them is used. Also note that when k is a prime number, the Euler's phi function is

$$\varphi(k) = k - 1 \tag{6.38}$$

For example, a 3-stage shift register generator will produce

$$N_L = \frac{\varphi(2^3 - 1)}{3} = \frac{\varphi(7)}{3} = \frac{7 - 1}{3} = 2 \tag{6.39}$$

and a 6-stage shift register,

$$N_L = \frac{\varphi(2^6 - 1)}{6} = \frac{\varphi(63)}{6} = \frac{63}{6} \times \frac{(3 - 1)}{3} \times \frac{(7 - 1)}{7} = 6 \tag{6.40}$$

Maximal Length Sequence Characteristic Polynomial

Consider an n-stage maximal length linear shift register whose feedback connections correspond to n, k, m, etc. This maximal length shift register can be described using its characteristic polynomial defined by

$$x^n + x^k + x^m + \ldots + 1 \tag{6.41}$$

where the additions are modulo 2. Therefore, if the characteristic polynomial for an n-stage shift register is known, one can easily determine the register feedback connections and consequently deduce the corresponding maximal length sequence. For example, consider a 6-stage shift register whose characteristic polynomial is

$$x^6 + x^5 + 1 \tag{6.42}$$

1. Zierler, N., *Several Binary-Sequence Generators*, MIT Technical Report No. 95, Sept. 1955.

It follows that the shift register which generates a maximal length sequence is shown in Fig. 6.14.

One of the most important issues associated with generating a maximal length sequence using a linear shift register is determining the characteristic polynomial. This has been and continues to be a subject of research for many radar engineers and designers. It has been shown that polynomials which are both irreducible (not factorable) and primitive will produce maximal length shift register generators.

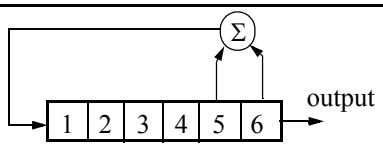

Figure 6.14. Linear shift register whose characteristic polynomial is $x^6 + x^5 + 1$.

A polynomial of degree n is irreducible if it is not divisible by any polynomial of degree less than n. It follows that all irreducible polynomials must have an odd number of terms. Consequently, only linear shift register generators with an even number of feedback connections can produce maximal length sequences. An irreducible polynomial is primitive if and only if it divides $x^n - 1$ for no value of n less than $2^n - 1$.

The MATLAB function *"prn_ambig.m"* calculates and plots the ambiguity function associated with a given PRN code. Figure 6.15 shows the output of this function for

u31 = [1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 -1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 1 1 -1 1 -1 -1]

Figure 6.16 is similar to Fig. 6.15, except in this case the input maximal length sequence is

u15=[1 -1 -1 -1 1 1 1 1 -1 1 -1 1 1 -1 -1]

6.3.2. Polyphase Codes

The signal corresponding to polyphase codes is as that given in Eq. (6.22) and the corresponding ambiguity function was given in Eq. (6.24). The only exception being that the phase θ_n is no longer restricted to $(0, \pi)$. Hence, the coefficient D_n are no longer equal to ± 1 but can be complex depending on the value of θ_n. Polyphase Barker codes have been investigated by many scientists and much is well documented in the literature. In this chapter the discussion will be limited to Frank codes.

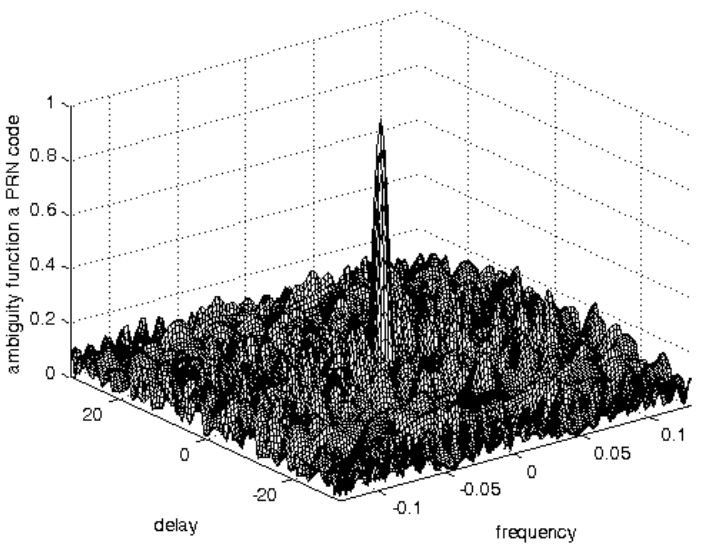

Figure 6.15a. Ambiguity function corresponding to a 31-bit PRN code.

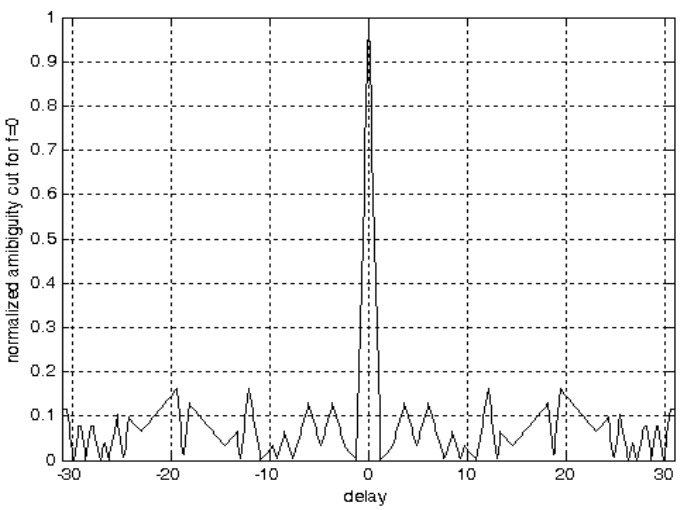

Figure 6.15b. Zero Doppler cut corresponding to Fig. 6.15a.

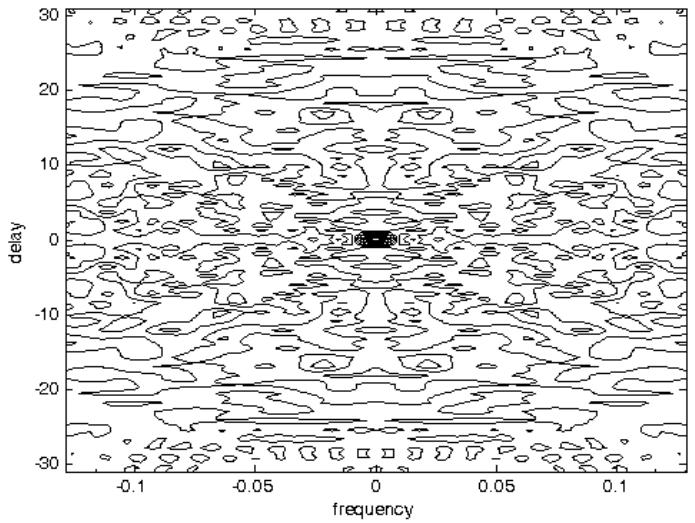

Figure 6.15c. Contour plot corresponding to Fig. 6.15a.

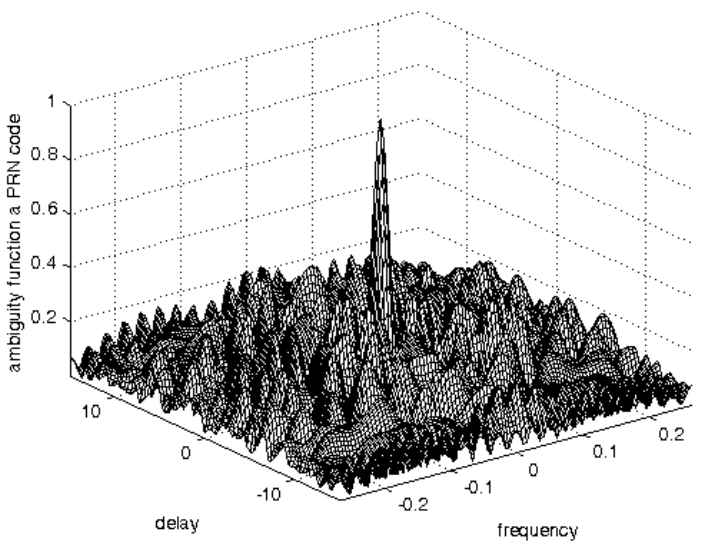

Figure 6.16a. Ambiguity function corresponding to a 15-bit PRN code.

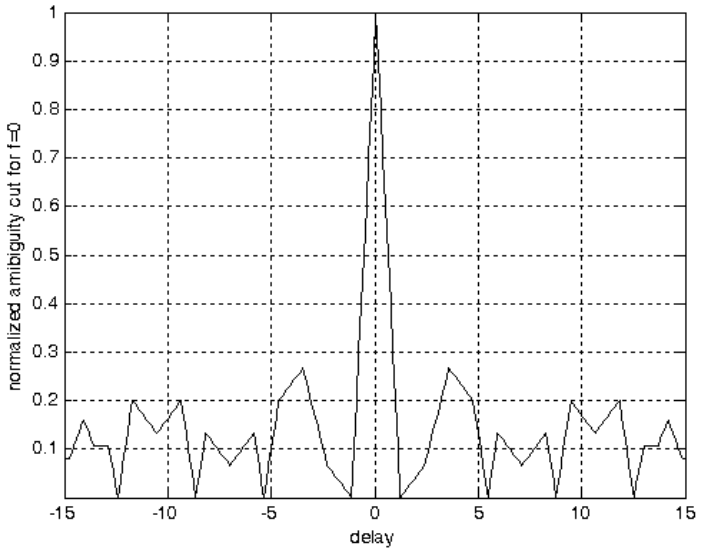

Figure 6.16b. Zero Doppler cut corresponding to Fig. 6.16a.

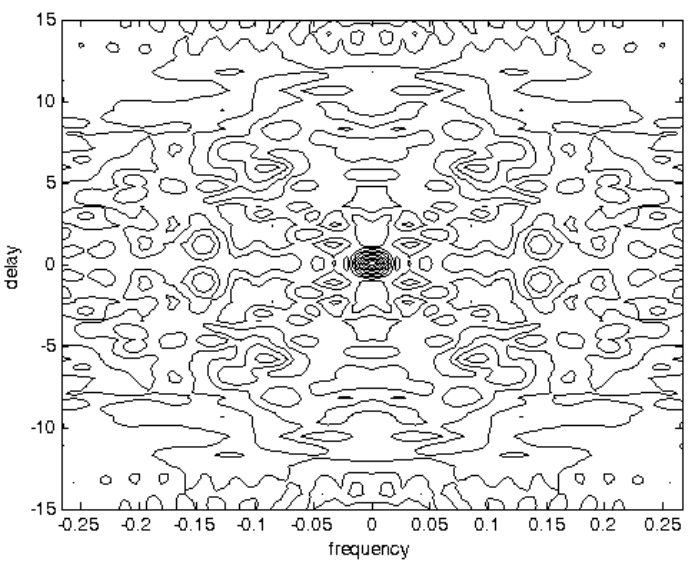

Figure 6.16c. Contour plot corresponding to Fig. 6.16a.

Frank codes

In this case, a single pulse of width T_p is divided into N equal groups; each group is subsequently divided into other N subpulses each of width τ_0. Therefore, the total number of subpulses within each pulse is N^2, and the compression ratio is $\xi = N^2$. As previously, the phase within each subpulse is held constant with respect to some CW reference signal.

A Frank code of N^2 subpulses is referred to as an N-phase Frank code. The first step in computing a Frank code is to divide $360°$ by N and define the result as the fundamental phase increment $\Delta\varphi$. More precisely,

$$\Delta\varphi = 360°/N \tag{6.43}$$

Note that the size of the fundamental phase increment decreases as the number of groups is increased, and because of phase stability, this may degrade the performance of very long Frank codes. For N-phase Frank code the phase of each subpulse is computed from

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 2 & 3 & \dots & N-1 \\ 0 & 2 & 4 & 6 & \dots & 2(N-1) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & (N-1) & 2(N-1) & 3(N-1) & \dots & (N-1)^2 \end{pmatrix} \Delta\varphi \tag{6.44}$$

where each row represents a group, and a column represents the subpulses for that group. For example, a 4-phase Frank code has $N = 4$, and the fundamental phase increment is $\Delta\varphi = (360°/4) = 90°$. It follows that

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 90° & 180° & 270° \\ 0 & 180° & 0 & 180° \\ 0 & 270° & 180° & 90° \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{pmatrix} \tag{6.45}$$

Therefore, a Frank code of 16 elements is given by

$$F_{16} = \{1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; j \;\; -1 \;\; -j \;\; 1 \;\; -1 \;\; 1 \;\; -1 \;\; 1 \;\; -j \;\; -1 \;\; j\} \tag{6.46}$$

A plot of the ambiguity function for F_{16} is shown in Fig. 6.17. Note the thumb-tack shape of the ambiguity function. This plot can be reproduced using the following MATLAB code. The phase increments within each row represent a step-wise approximation of an up-chirp LFM waveform. The phase increments for subsequent rows increase linearly versus time. Thus, the correspond-

ing LFM chirp slopes also increase linearly for subsequent rows. This is illustrated in Fig. 6.18, for F_{16}.

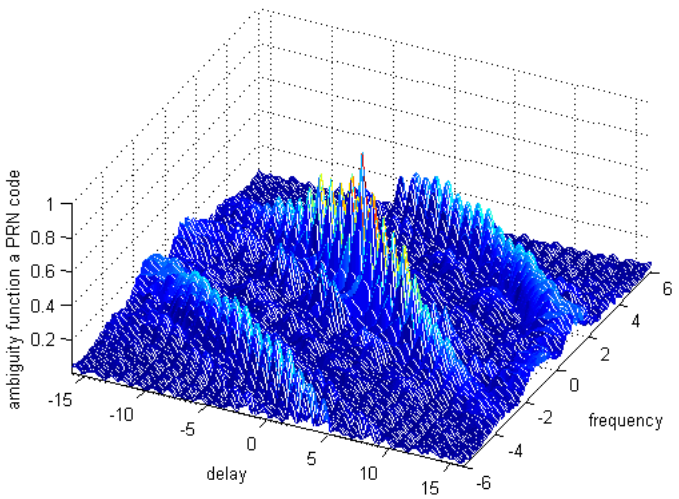

Figure 6.17a. Ambiguity plot for Frank code F_{16}.

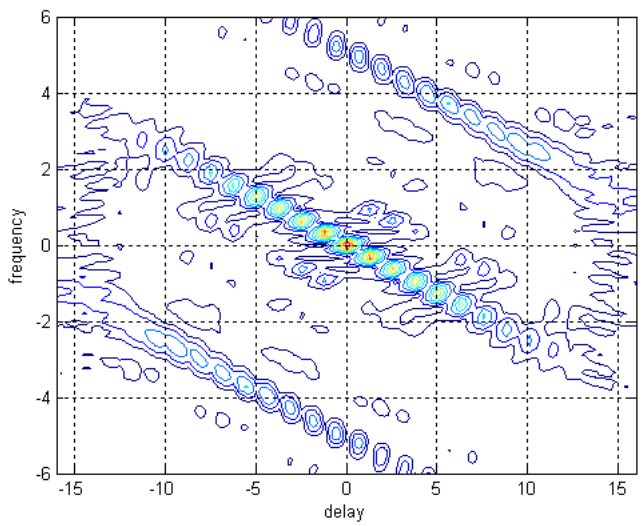

Figure 6.17b. Contour plot corresponding to Fig. 6.17a.

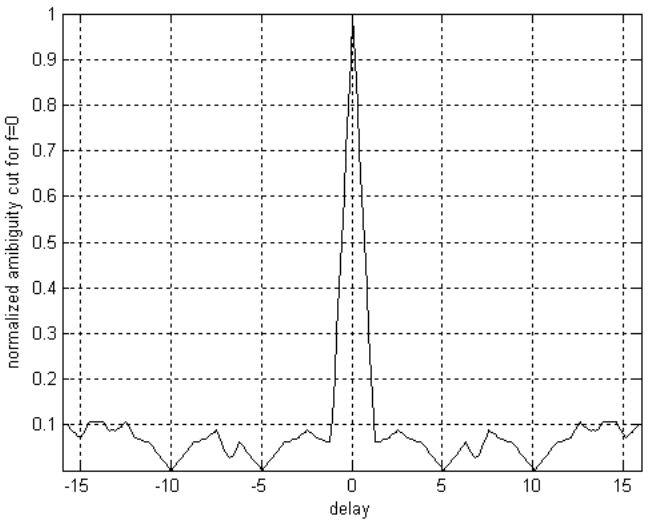

Figure 6.17c. Zero Doppler cut corresponding to Fig. 6.17a.

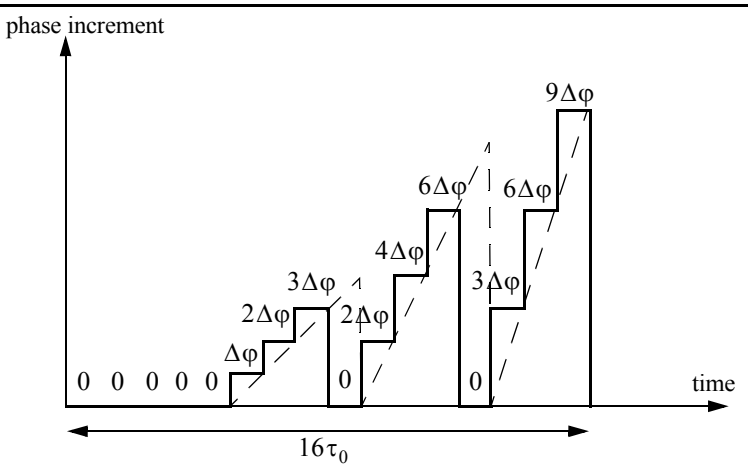

Figure 6.18. Step-wise approximation of an up-chirp waveform, using a Frank code of 16 elements.

6.4. Frequency Codes

Frequency codes are derived from Eq. (6.1) under the condition stated in Eq. (6.9) (i.e., $\theta_n = 0 \quad ;and \;\; a_n = 1, or \;\; 0$). The Stepped Frequency Waveform (SFW) discussed in the previous chapter is considered to be a code under this class of discrete coded waveforms. The ambiguity function was derived in Chapter 5 for SFW. In this chapter the focus is on another type of frequency codes that is called the Costas frequency code.

6.4.1. Costas Codes

Construction of Costas codes can be understood in the context of SFW. In SFW, a relatively long pulse of length T_p is divided into N subpulses, each of width τ_0 ($T_p = N\tau_0$). Each group of N subpulses is called a burst. Within each burst the frequency is increased by Δf from one subpulse to the next. The overall burst bandwidth is $N\Delta f$. More precisely,

$$\tau_0 = T_p / N \tag{6.47}$$

and the frequency for the ith subpulse is

$$f_i = f_0 + i\Delta f; \;\; i = 1, N \tag{6.48}$$

where f_0 is a constant frequency and $f_0 \gg \Delta f$. It follows that the time-bandwidth product of this waveform is

$$\Delta f T_p = N^2 \tag{6.49}$$

Costas[1] signals (or codes) are similar to SFW, except that the frequencies for the subpulses are selected in a random fashion, according to some predetermined rule or logic. For this purpose, consider the $N \times N$ matrix shown in Fig. 6.19 b. In this case, the rows are indexed from $i = 1, 2, \ldots, N$ and the columns are indexed from $j = 0, 1, 2, \ldots, (N-1)$. The rows are used to denote the subpulses and the columns are used to denote the frequency. A *dot* indicates the frequency value assigned to the associated subpulse. In this fashion, Fig. 6.19 a shows the frequency assignment associated with an SFW. Alternatively, the frequency assignments in Fig. 6.19b are chosen randomly. For a matrix of size $N \times N$, there are a total of $N!$ possible ways of assigning the dots (i.e., $N!$ possible codes).

The sequences of dot assignments for which the corresponding ambiguity function approaches an ideal or a *thumb-tack* response are called Costas codes.

1. Costas, J. P., A Study of a Class of Detection Waveforms Having Nearly Ideal Range-Doppler Ambiguity Properties, *Proc. IEEE* 72, 1984, pp. 996-1009.

A near thumb-tack response was obtained by Costas using the following logic: There is only one frequency per time slot (row) and per frequency slot (column). Therefore, for an $N \times N$ matrix the number of possible Costas codes is drastically less than $N!$. For example, there are $N_c = 4$ possible Costas codes for $N = 3$, and $N_c = 40$ possible codes for $N = 5$. It can be shown that the code density, defined as the ratio $N_c/N!$, gets significantly smaller as N becomes larger

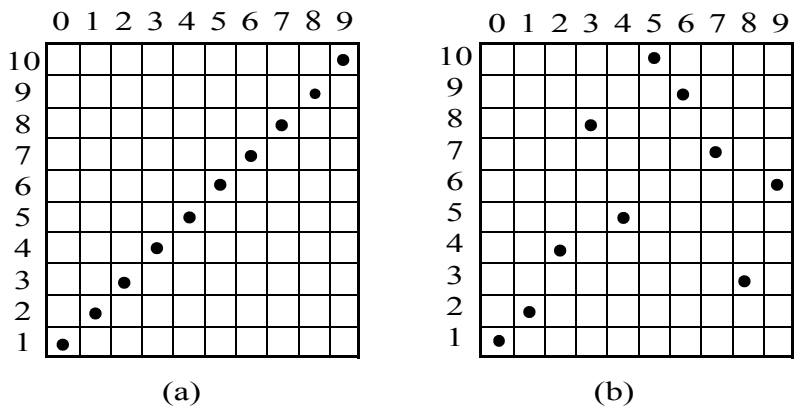

Figure 6.19. Frequency assignment for a burst of N subpulses. (a) SFW (stepped LFM); (b) Costas code of length Nc = 10.

There are numerous analytical ways to generate Costas codes. In this section we will describe two of these methods. First, let q be an odd prime number, and choose the number of subpulses as

$$N = q - 1 \tag{6.50}$$

Define γ as the primitive root of q. A primitive root of q (an odd prime number) is defined as γ such that the powers $\gamma, \gamma^2, \gamma^3, \ldots, \gamma^{q-1}$ modulo q generate every integer from 1 to $q - 1$.

In the first method, for an $N \times N$ matrix, label the rows and columns, respectively, as

$$\begin{aligned} i &= 0, 1, 2, \ldots, (q-2) \\ j &= 1, 2, 3, \ldots, (q-1) \end{aligned} \tag{6.51}$$

Place a dot in the location (i, j) corresponding to f_i if and only if

$$i = (\gamma)^j \ (modulo \ q) \tag{6.52}$$

In the next method, Costas code is first obtained from the logic described above; then by deleting the first row and first column from the matrix a new code is generated. This method produces a Costas code of length $N = q - 2$.

Define the normalized complex envelope of the Costas signal as

$$x(t) = \frac{1}{\sqrt{N\tau_0}} \sum_{l=0}^{N-1} x_l(t - l\tau_0) \tag{6.53}$$

$$x_l(t) = \begin{pmatrix} \exp(j2\pi f_l t) & 0 \le t \le \tau_0 \\ 0 & elsewhere \end{pmatrix} \tag{6.54}$$

Costas showed that the output of the matched filter is

$$\chi(\tau, f_d) = \frac{1}{N} \sum_{l=0}^{N-1} \exp(j2\pi l f_d \tau) \left\{ \Phi_{ll}(\tau, f_d) + \sum_{\substack{q=0 \\ q \ne l}}^{N-1} \Phi_{lq}(\tau - (l-q)\tau_0, f_d) \right\} \tag{6.55}$$

$$\Phi_{lq}(\tau, f_d) = \left(\tau_0 - \frac{|\tau|}{\tau_0}\right) \frac{\sin\alpha}{\alpha} \exp(-j\beta - j2\pi f_q \tau) \quad , \ |\tau| \le \tau_1 \tag{6.56}$$

$$\alpha = \pi(f_l - f_q - f_d)(\tau_0 - |\tau|) \tag{6.57}$$

$$\beta = \pi(f_l - f_q - f_d)(\tau_0 + |\tau|) \tag{6.58}$$

Three-dimensional plots of the ambiguity function of Costas signals show the near thumb-tack response of the ambiguity function. All side-lobes, except for a few around the origin, have amplitude $1/N$. Few sidelobes close to the origin have amplitude $2/N$, which is typical of Costas codes. The compression ratio of a Costas code is approximately N.

6.5. Ambiguity Plots for Discrete Coded Waveforms

Plots of the ambiguity function for a given code and the corresponding cuts along zero delay and zero Doppler provide strong indication about the code's characteristics in range and Doppler. Earlier, it was stated that the *goodness* of a given code is measured by its range and Doppler resolution characteristics. Therefore, plotting the ambiguity function of a given code is a key part of the design and analysis of radar waveforms. Unfortunately, some of the formulas for the ambiguity function are rather complicated and fairly difficult to code by the nonexpert programmer. In this section, a numerical technique for plotting

the ambiguity function of any code is presented. This technique takes advantage of the computation power of MATLAB by exploiting one of the properties of the ambiguity function. Three-dimensional plots are built successively from cuts of the ambiguity function as different Doppler mismatches.

For this purpose, consider the ambiguity function property given in Eq. (5.8) and repeated here as Eq. (6.59)

$$\left|\chi(\tau;f_d)\right|^2 = \left|\int X^*(f)X(f-f_d)e^{-j2\pi f\tau}df\right|^2 \quad \textbf{(6.59)}$$

where $X(f)$ is the Fourier transform of the signal $x(t)$. Using Eq. (6.59), one can compute the ambiguity function by first computing the FT of the signal under consideration, delaying it by some value f_d, and then taking the inverse FT. When the signal under consideration is a discrete coded waveform then the Fast Fourier transform is utilized. From this one can compute plots of the ambiguity function using the following technique:

1. Determine the code U under consideration. Note that U may have complex values in accordance with the class of code being considered.
2. Extend the length of the code to the next power of 2 by zero padding (see Chapter 2 for details on interpolation).
3. For better display utilize an FFT whose size is 8 times or higher than the power integer of 2 computed in step 2.
4. Compute the FFT of the extended sequence.
5. Generate vectors of frequency mismatches and delay cuts.
6. Calculate using vector notation the value of $X(f-f_d)$.
7. Compute and store the vector resulting from the point by point multiplication $X^*(f)X(f-f_d)$.
8. Compute the inverse FFT of the product in step 7 for each delay value and store in a two-dimensional (2-D) array.
9. Plot the amplitude square of the resulting 2-D array to generate the ambiguity plot for the specific code under consideration.

An implementation of this algorithm using MATLAB was completed; this program is called *"ambiguity_code.m."* The listing of this program is as follows:

```
function [ambig] = ambiguity_code(uinput)
% Compute and plot the ambiguity function for any give code u
% Compute the ambiguity function by utilizing the FFT
% through combining multiple range cuts
N = size(uinput,2);
tau = N;
code = uinput;
```

```
samp_num = size(code,2) * 10;
n = ceil(log(samp_num) / log(2));
nfft = 2^n;
u(1:nfft) = 0;
j = 0;
for index = 1:10:samp_num
    index;
    j = j+1;
    u(index:index+10-1) = code(j);
end
% set-up the array v
v = u;
delay = linspace(0,5*tau,nfft);
freq_del = 12 / tau /100;
j = 0;
vfft = fft(v,nfft);
for freq = -6/tau:freq_del:6/tau;
    j = j+1;
    exf = exp(sqrt(-1) * 2. * pi * freq .* delay);
    u_times_exf = u .* exf;
    ufft = fft(u_times_exf,nfft);
    prod = ufft .* conj(vfft);
    ambig(j,:) = fftshift(abs(ifft(prod))');
end
freq = linspace(-6,6, size(ambig,1));
delay = linspace(-N,N,nfft);
figure(1)
mesh(delay,freq,(ambig ./ max(max(ambig))))
% colormap([.5 .5 .5])
% colormap(gray)
axis tight
ylabel('frequency')
xlabel('delay')
zlabel('ambiguity function a PRN code')
figure(2)
plot(delay,ambig(51,:)/(max(max(ambig))),'k')
xlabel('delay')
ylabel('normalized amibiguity cut for f=0')
grid
axis tight
figure(3)
contour(delay,freq,(ambig ./ max(max(ambig))))
axis tight
% colormap([.5 .5 .5])
% colormap(gray)
ylabel('frequency')
xlabel('delay')
grid
```

Problems

6.1. Define $\{x_I(n) = 1, -1, 1\}$ and $\{x_Q(n) = 1, 1, -1\}$. (a) Compute the discrete correlations: R_{x_I}, R_{x_Q}, $R_{x_I x_Q}$, and $R_{x_Q x_I}$. (b) A certain radar transmits the signal $s(t) = x_I(t)\cos 2\pi f_0 t - x_Q(t)\sin 2\pi f_0 t$. Assume that the autocorrelation $s(t)$ is equal to $y(t) = y_I(t)\cos 2\pi f_0 t - y_Q(t)\sin 2\pi f_0 t$. Compute and sketch $y_I(t)$ and $y_Q(t)$.

6.2. Consider the 7-bit Barker code, designated by the sequence $x(n)$. (a) Compute and plot the autocorrelation of this code. (b) A radar uses binary phase coded pulses of the form $s(t) = r(t)\cos(2\pi f_0 t)$, where $r(t) = x(0),\ for\ \ 0 < t < \Delta t$, $\quad r(t) = x(n),\ for\ \ n\Delta t < t < (n+1)\Delta t$, and $r(t) = 0,\ for\ \ t > 7\Delta t$. Assume $\Delta t = 0.5\mu s$. (a) Give an expression for the autocorrelation of the signal $s(t)$, and for the output of the matched filter when the input is $s(t - 10\Delta t)$; (b) compute the time bandwidth product, the increase in the peak SNR, and the compression ratio.

6.3. (a) Perform the discrete convolution between the sequence ϕ_{11} defined in Eq. (6.31), and the transversal filter impulse response; and (b) sketch the corresponding transversal filter output.

6.4. Repeat the previous problem for $N = 13$ and $k = 6$. Use Barker code of length 13.

6.5. Develop a Barker code of length 35. Consider both B_{75} and B_{57}.

6.6. The smallest positive primitive root of $q = 11$ is $\gamma = 2$; for $N = 10$ generate the corresponding Costas matrix.

6.7. Compute the discrete autocorrelation for an F_{16} Frank code.

6.8. Generate a Frank code of length 8, i.e., F_8.

6.9. Using the MATLAB program developed in this chapter, plot the matched filter output for a 3-, 4-, and 5-bits Barker code.