

CHAPTER 7

Spread-spectrum Modulation

Problem 7.1

(a) The PN sequence length is

$$N = 2^m - 1 = 2^4 - 1 = 15$$

(b) The chip duration is

$$T_c = \frac{1}{10^7} \text{ s} = 0.1 \text{ } \mu\text{s}$$

(c) The period of the PN sequence is

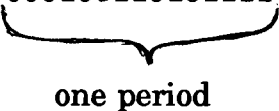
$$\begin{aligned} T &= NT_c \\ &= 15 \times 0.1 = 1.5 \text{ } \mu\text{s} \end{aligned}$$

Problem 7.2

<u>Shift number</u>	<u>Shift-register contents</u>	<u>Modulo-2 adder output</u>	<u>Shift-register output</u>
0	1000		
1	0100	$0 + 0 = 0$	0
2	0010	$0 + 0 = 0$	0
3	1001	$1 + 0 = 1$	0
4	1100	$0 + 1 = 1$	1
5	0110	$0 + 0 = 0$	0
6	1011	$1 + 0 = 1$	0
7	0101	$1 + 1 = 0$	1
8	1010	$0 + 1 = 1$	1

9	1101	$1 + 0 = 1$	0
10	1110	$0 + 1 = 1$	1
11	1111	$1 + 0 = 1$	0
12	0111	$1 + 1 = 0$	1
13	0011	$1 + 1 = 0$	1
14	0001	$1 + 1 = 0$	1
15	1000	$0 + 1 = 1$	1

The output sequence is therefore

11 000100110101111 0001

 one period

Problem 7.3

(a) From both Table 7.2a and Table 7.2b we note the following:

Balance property:

Number of 1s in one period = 16

Number of 0s in one period = 15

Hence, the number of 1s exceeds the number of 0s by one.

(b) Run property:

In both Tables 7.2a and 7.2b, we count a total of 8 runs of 1s and a total of 8 runs of 0s. Moreover, we note the following:

Runs of length 1 : 4

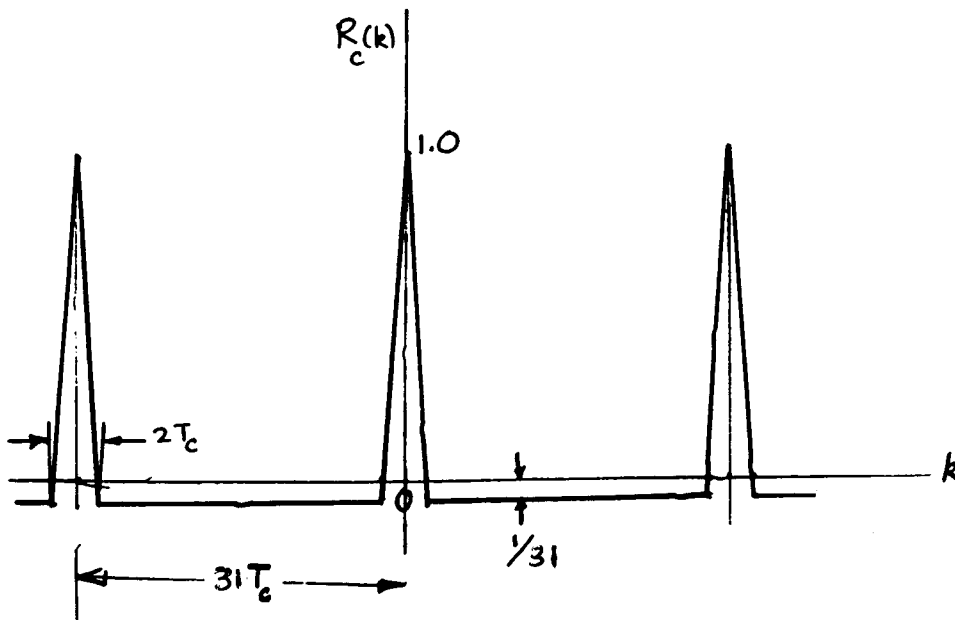
Runs of length 2 : 2

Runs of length 3 : 1

(c) Autocorrelation function:

$$R_c(k) = \begin{cases} 1, & k=1N \\ -\frac{1}{N}, & k \neq 1N \end{cases}$$

Hence, we have (not to scale)



Problem 7.4

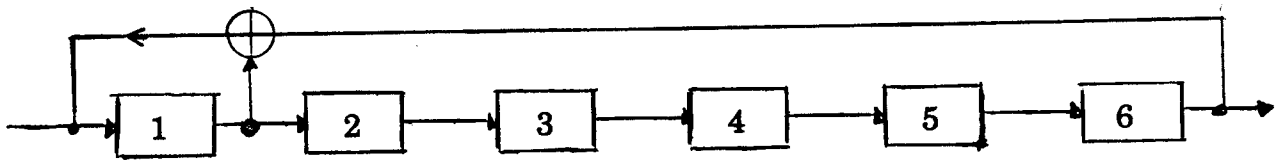


Table 1

Feedback symbol	State of Feedback-shift register	Output symbol
	1 0 0 0 0 0	
1	1 1 0 0 0 0	0
1	1 1 1 0 0 0	0
1	1 1 1 1 0 1	0
1	1 1 1 1 1 0	0
1	1 1 1 1 1 1	0
0	0 1 1 1 1 1	1
1	1 0 1 1 1 1	1
0	0 1 0 1 1 1	1
1	1 0 1 0 1 1	1
0	0 1 0 1 0 1	1
1	1 0 1 0 1 0	1
1	1 1 0 1 0 1	0
0	0 1 1 0 1 0	1
0	0 0 1 1 0 1	0
1	1 0 0 1 1 0	1
1	1 1 0 0 1 1	0
0	0 1 1 0 0 1	1
1	1 0 1 1 0 0	1
1	1 1 0 1 1 0	0
1	1 1 1 0 1 1	0
0	0 1 1 1 0 1	1
1	1 0 1 1 1 0	1

Table 1 continued

Feedback symbol	State of feedback- shift register	Output symbol
1	1 1 0 1 1 1	0
0	0 1 1 0 1 1	1
1	1 0 1 1 0 1	1
0	0 1 0 1 1 0	1
0	0 0 1 0 1 1	0
1	1 0 0 1 0 1	1
0	0 1 0 0 1 0	1
0	0 0 1 0 0 1	0
1	1 0 0 1 0 0	1
1	1 1 0 0 1 0	0
1	1 1 1 0 0 1	0
0	0 1 1 1 0 0	1
0	0 0 1 1 1 0	0
0	0 0 0 1 1 1	0
1	1 0 0 0 1 1	1
0	0 1 0 0 0 1	1
1	1 0 1 0 0 0	1
1	1 1 0 1 0 0	0
1	1 1 1 0 1 0	0
1	1 1 1 1 0 1	0
0	0 1 1 1 1 0	1
0	0 0 1 1 1 1	0
1	1 0 0 1 1 1	1
0	0 1 0 0 1 1	1
1	1 0 1 0 0 1	1
0	0 1 0 1 0 0	1
0	0 0 1 0 1 0	1

Table 1 continued

Feedback symbol	State of feedback- shift register	Output symbol
0	0 0 0 1 0 1	0
1	1 0 0 0 1 0	1
1	1 1 0 0 0 1	0
0	0 1 1 0 0 0	1
0	0 0 1 1 0 0	0
0	0 0 0 1 1 0	0
0	0 0 0 0 1 1	0
1	1 0 0 0 0 1	0
0	0 1 0 0 0 0	1
0	0 0 1 0 0 0	0
0	0 0 0 1 0 0	0
0	0 0 0 0 1 0	0
0	0 0 0 0 0 1	0
1	1 0 0 0 0 0	1

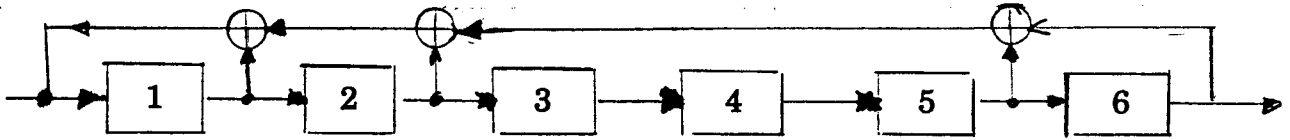


Table 2

Feedback symbol	State of feedback-shift register	Output symbol
	1 0 0 0 0 0	
1	1 1 0 0 0 0	0
0	0 1 1 0 0 0	0
1	1 0 1 1 0 0	0
1	1 1 0 1 1 0	0
1	1 1 1 0 1 1	0
0	0 1 1 1 0 1	1
0	0 0 1 1 1 0	1
1	1 0 0 1 1 1	0
1	1 1 0 0 1 1	1
0	0 1 1 0 0 1	1
0	0 0 1 1 0 0	1
0	0 0 0 1 1 0	0
1	1 0 0 0 1 1	0
1	1 1 0 0 0 1	1
1	1 1 1 0 0 0	1
0	0 1 1 1 0 0	0
1	1 0 1 1 1 0	0
0	0 1 0 1 1 1	0
1	1 0 1 0 1 1	1
1	1 1 0 1 0 1	1
1	1 1 1 0 1 0	1
1	1 1 1 1 0 1	0
1	1 1 1 1 1 0	1
1	1 1 1 1 1 1	0
0	0 1 1 1 1 1	1

Table 2 continued

Feedback symbol	State of feedback- shift register	Output symbol
1	1 0 1 1 1 1	1
1	1 1 0 1 1 1	1
0	0 1 1 0 1 1	1
1	1 0 1 1 0 1	1
0	0 1 0 1 1 0	1
0	0 0 1 0 1 1	0
0	0 0 0 1 0 1	1
1	1 0 0 0 1 0	1
0	0 1 0 0 0 1	0
0	0 0 1 0 0 0	1
0	0 0 0 1 0 0	0
0	0 0 0 0 1 0	0
1	1 0 0 0 0 1	0
0	0 1 0 0 0 0	1
1	1 0 1 0 0 0	0
1	1 1 0 1 0 0	0
0	0 1 1 0 1 0	0
0	0 0 1 1 0 1	0
1	1 0 0 1 1 0	1
0	0 1 0 0 1 1	0
1	1 0 1 0 0 1	1
0	0 1 0 1 0 0	1
1	1 0 1 0 1 0	0
0	0 1 0 1 0 1	0
0	0 0 1 0 1 0	1

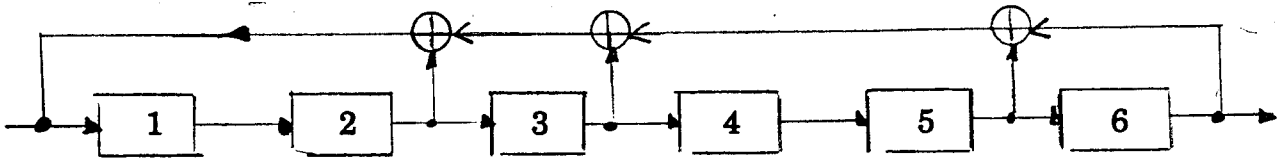


Table 3

Feedback symbol	State of feed-back shift register	Output symbol
	1 0 0 0 0 0	
0	0 1 0 0 0 0	0
1	1 0 1 0 0 0	0
1	1 1 0 1 0 0	0
1	1 1 1 0 1 0	0
1	1 1 1 1 0 1	0
1	1 1 1 1 1 0	1
1	1 1 1 1 1 1	0
0	0 1 1 1 1 1	1
0	0 0 1 1 1 1	1
1	1 0 0 1 1 1	1
0	0 1 0 0 1 1	1
1	1 0 1 0 0 1	1
0	0 1 0 1 0 0	1
1	1 0 1 0 1 0	0
0	0 1 0 1 0 1	0
0	0 0 1 0 1 0	1
0	0 0 0 1 0 1	0
1	1 0 0 0 1 0	1
1	1 1 0 0 0 1	0
0	0 1 1 0 0 0	1
0	0 0 1 1 0 0	0

Table 3 continued

Feedback symbol	State of feedback- shift register	Output symbol
1	1 0 0 1 1 0	0
1	1 1 0 0 1 1	0
1	1 1 1 0 0 1	1
1	1 1 1 1 0 0	1
0	0 1 1 1 1 0	0
1	1 0 1 1 1 1	0
1	1 1 0 1 1 1	1
1	1 1 1 0 1 1	1
0	0 1 1 1 0 1	1
1	1 0 1 1 1 0	1
0	0 1 0 1 1 1	0
1	1 0 1 0 1 1	1
1	1 1 0 1 0 1	1
0	0 1 1 0 1 0	1
1	1 0 1 1 0 1	0
0	0 1 0 1 1 0	1
0	0 0 1 0 1 1	0
1	1 0 0 1 0 1	1
1	1 1 0 0 1 0	1
0	0 1 1 0 0 1	0
1	1 0 1 1 0 0	1
1	1 1 0 1 1 0	0
0	0 1 1 0 1 1	0
0	0 0 1 1 0 1	1
0	0 0 0 1 1 0	1

Table 3 continued

Feedback symbol	State of feedback- shift register	Output symbol
1	1 0 0 0 1 1	0
0	0 1 0 0 0 1	1
0	0 0 1 0 0 0	1
1	1 0 0 1 0 0	0
0	0 1 0 0 1 0	0
0	0 0 1 0 0 1	0
0	0 0 0 1 0 0	1
0	0 0 0 0 1 0	0
1	1 0 0 0 0 1	0
1	1 1 0 0 0 0	1
1	1 1 1 0 0 0	0
0	0 1 1 1 0 0	0
0	0 0 1 1 1 0	0
0	0 0 0 1 1 1	0
0	0 0 0 0 1 1	1
0	0 0 0 0 0 1	1
1	1 0 0 0 0 0	1

Table 3 continued

Feedback symbol	State of feedback- shift register	Output symbol
1	1 0 0 1 0 1	0
0	0 1 0 0 1 0	1
0	0 0 1 0 0 1	0
1	1 0 0 1 0 0	1
1	1 1 0 0 1 0	0
1	1 1 1 0 0 1	0
1	1 1 1 1 0 0	1
0	0 1 1 1 1 0	0
0	0 0 1 1 1 1	0
0	0 0 0 1 1 1	1
0	0 0 0 0 1 1	1
0	0 0 0 0 0 1	1
1	1 0 0 0 0 0	1

Problem 7.5

	State of feedback- shift register	Output symbol
Initial state	1 0 0 0 0	
	0 1 0 0 0	0
	0 0 1 0 0	0
	0 0 0 1 0	0
	0 0 0 0 1	0
	1 1 1 0 1	1
	1 0 0 1 1	1
	1 0 1 0 0	1
	0 1 0 1 0	0
	0 0 1 0 1	0
	1 1 1 1 1	1
	1 0 0 1 0	1
	0 1 0 0 1	0
	1 1 0 0 1	1
	1 0 0 0 1	1
	1 0 1 0 1	1
	1 0 1 1 1	1
	1 0 1 1 0	1
	0 1 0 1 1	0
	1 1 0 0 0	1
	0 1 1 0 0	0
	0 0 1 1 0	0
	0 0 0 1 1	0
	1 1 1 0 0	1
	0 1 1 1 0	0
	0 0 1 1 1	0
	1 1 1 1 0	1
	0 1 1 1 1	0
	1 1 0 1 0	1
	0 1 1 0 1	0
	1 1 0 1 1	1
	1 0 0 0 0	1

The 31-element code generated by the scheme shown in Fig. P9.2 is exactly the same as that described in Table 9.2b. Note, however, the code described in Table 9.2b appears in reversed order to that described in the above table; this reversal is clearly of a trivial nature.

Problem 7.6

(a) The modulo-2 sum of $b(t)$ and $c(t)$, on a pulse-by-pulse basis, is as follows

		$b(t)$	
		0	1
$c(t)$	0	0	1
	1	1	0

(b) If symbol 0 is represented by a sinusoid of zero phase shift, and symbol 1 is represented by a sinusoid of 180° phase shift, the output of the modulo-2 adder takes on the same form as that described in Table 7.3 of the text.

Problem 7.7

$$j(t) = \sqrt{2J} \cos(2\pi f_c t + \theta)$$

The basis functions are

$$\phi_k(t) = \begin{cases} \sqrt{\frac{2}{T_c}} \cos(2\pi f_c t), & kT_c \leq t \leq (k+1)T_c \\ 0, & \text{otherwise} \end{cases}$$

$$\tilde{\phi}_k(t) = \begin{cases} \sqrt{\frac{2}{T_c}} \sin(2\pi f_c t), & kT_c \leq t \leq (k+1)T_c \\ 0, & \text{otherwise} \end{cases}$$

Hence, we may express the jamming signal $j(t)$ as

$$j(t) = \sqrt{JT_c} \cos\theta \sum_{k=0}^{N-1} \phi_k(t) - \sqrt{JT_c} \sin\theta \sum_{k=0}^{N-1} \tilde{\phi}_k(t)$$

Problem 7.8

The processing gain is

$$\frac{T_b}{T_c} = \frac{1/T_c}{1/T_b}$$

The spread bandwidth of the transmitted signal is proportional to $1/T_c$. The despreading bandwidth of the received signal is proportional to $1/T_b$. Hence,

$$\text{Processing gain} = \frac{\text{spread bandwidth of transmitted signal}}{\text{despreading bandwidth of received signal}}$$

Problem 7.9

$$m = 19$$

$$N = 2^m - 1 = 2^{19} - 1 \approx 2^{19}$$

The processing gain is

$$\begin{aligned} 10 \log_{10} N &\approx 10 \log_{10} 2^{19} \\ &= 190 \times 0.3 \\ &= 57 \text{ dB} \end{aligned}$$

Problem 7.10

(a) Processing gain = $10\log_{10}(2^m - 1) = 10\log_{10}(2^{19} - 1) = 57 \text{ dB}$

(b) Antijam margin = (Processing gain) - $10\log_{10}\left(\frac{E_b}{N_0}\right)$

The probability of error is

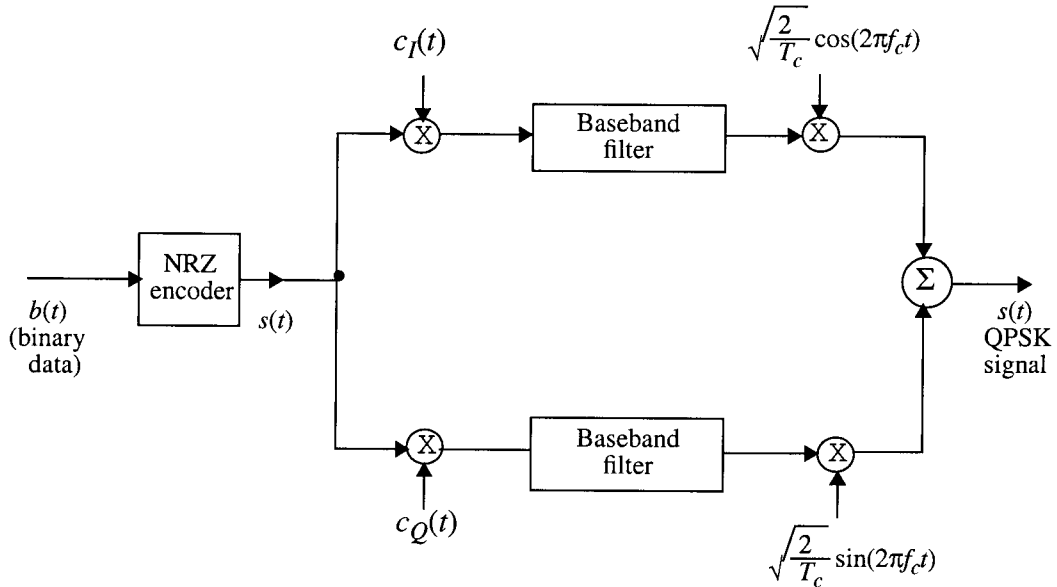
$$P_e = \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right)$$

With $P_e = 10^{-5}$, we have $E_b/N_0 = 9$. Hence,

$$\begin{aligned} \text{Antijam margin} &= 57 - 10\log_{10}9 = 57 - 9.5 \\ &= 47.5 \text{ dB} \end{aligned}$$

Problem 7.11

The DS/QPSK signal modulator is given below:



The DS/QPSK modulated signal is

$$x(t) = \pm \sqrt{\frac{E}{T}} c_I(t) \cos(2\pi f_c t) \pm \sqrt{\frac{E}{T}} c_Q(t) \sin(2\pi f_c t)$$

where $c_I(t) = \{c_{0,I}(t), c_{1,I}(t), \dots, c_{N-1,I}(t)\}$ and

$$c_Q(t) = \{c_{0,Q}(t), c_{1,Q}(t), \dots, c_{N-1,Q}(t)\}$$

denote the spreading sequences for $0 \leq t \leq T_s$, which are applied to the in-phase and quadrature channels of the modulator.

Consider the following set of orthonormal basis functions:

$$\phi_{c_{I,k}}(t) = \begin{cases} \sqrt{\frac{2}{T_c}} \cos(2\pi f_c t), & kT_c \leq t \leq (k+1)T_c \\ 0, & \text{otherwise} \end{cases}$$

$$\phi_{c_{Q,k}}(t) = \begin{cases} \sqrt{\frac{2}{T_c}} \sin((2\pi f_c t),) & kT_c \leq t \leq (k+1)T_c \\ 0, & \text{otherwise} \end{cases}$$

where T_c is the chip duration; $k = 0, 1, 2, \dots, N-1$, and $N = T/T_c$, that is, N is the number of chips per bit.

The DS/QPSK modulated signal can be written as follows (using the set of basis functions):

$$\begin{aligned} s(t) &= \pm \sqrt{\frac{T_c E}{2T}} \cdot \sqrt{\frac{2}{T_c}} \cos(2\pi f_c t) c_I(t) \pm \sqrt{\frac{T_c E}{2T}} \cdot \sqrt{\frac{2}{T_c}} \sin(2\pi f_c t) c_Q(t) \\ &= \pm \sqrt{\frac{E_b}{2N}} \sum_{k=0}^{N-1} c_{I,k} \phi_{c_{I,k}}(t) \pm \sqrt{\frac{E_b}{2N}} \sum_{k=0}^{N-1} c_{Q,k} \phi_{c_{Q,k}}(t) \end{aligned}$$

The channel output at the receiving end of the system has the following form

$$x(t) = s(t) + j(t)$$

where $j(t)$ denotes the interference signal. We may express the interference signal using the $2N$ -dimensional basis functions as follows:

$$j(t) = \sum_{k=0}^{N-1} c_{I,k}(t) \phi_{c_{I,k}}(t) + \sum_{k=0}^{N-1} j_{c_{Q,k}}(t) \phi_{c_{Q,k}}(t)$$

where

$$j_{c_{I,k}} = \int_{kT_b}^{(k+1)T_b} j(t) \phi_{c_{I,k}}(t) dt$$

$$j_{c_{Q,k}} = \int_{kT_b}^{(k+1)T_b} j(t) \phi_{c_{Q,k}}(t) dt$$

$$k = 0, 1, \dots, N-1$$

The average power of the interferer is given by

$$J = \frac{1}{T_b} \sum_{k=0}^{N-1} j_{c_{I,k}}^2 + \frac{1}{T_b} \sum_{k=0}^{N-1} j_{c_{Q,k}}^2$$

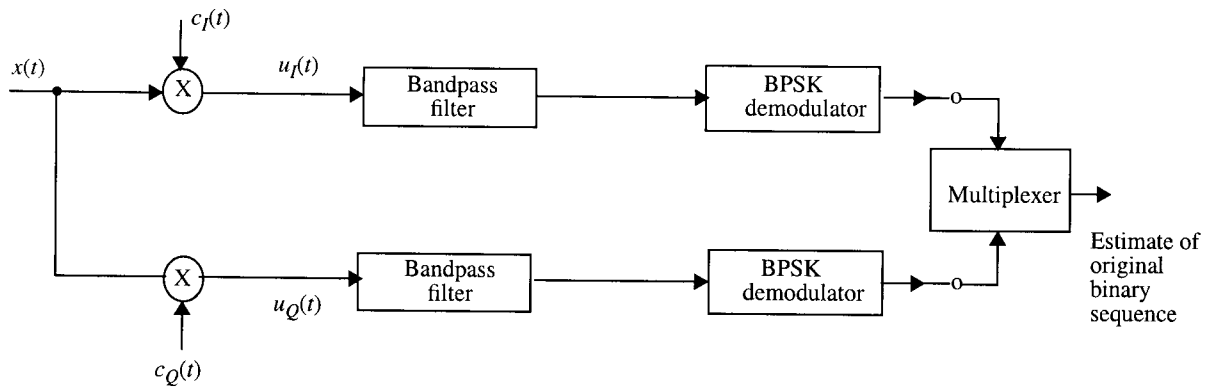
Assuming that the power is equally distributed between the in-phase and quadrature components:

$$J = \frac{1}{T_b} \sum_{k=0}^{2(N-1)} j_{c_{I,k}}^2$$

The mean of the interference signal is zero. The variance of the interference signal is therefore

$$\begin{aligned} \sigma_{jam}^2 &= \frac{1}{2N} \sum_{k=0}^{2(N-1)} j_{c_{I,k}}^2 \\ &= \frac{JT_c}{2} \end{aligned}$$

Demodulation



There are two stages of demodulation. First, the received signal $x(t)$ is multiplied by the despreading sequences $c_I(t)$ and $c_Q(t)$, yielding

$$u_I(t) = \pm \sqrt{\frac{E}{T}} \cos(2\pi f_c t) \pm c_I(t) c_Q(t) \sqrt{\frac{E}{T}} \sin(2\pi f_c t) + c_I(t) j(t)$$

$$u_Q(t) = \pm \sqrt{\frac{E}{T}} \sin(2\pi f_c t) \pm c_Q(t) c_I(t) \sqrt{\frac{E}{T}} \cos(2\pi f_c t) + c_Q(t) j(t)$$

The second terms in the right-hand side of $u_I(t)$ and $u_Q(t)$ are filtered by the bandpass filters, and the BPSK demodulators recover estimates of their respective binary sequences. Finally, the multiplexer reconstructs the original binary data stream.

Processing gain

The signal-to-noise ratio at the output of the receiver is

$$\begin{aligned} (\text{SNR})_0 &= \frac{\text{Instantaneous peak signal power}}{\sigma_{\text{jam}}^2} \\ &= \frac{E}{JT_c/2} = \frac{2E}{JT_c} \end{aligned}$$

The signal-to-noise ratio at the input of the coherent receiver is

$$\begin{aligned} (\text{SNR})_I &= \frac{\text{average input-signal power}}{\text{average interferer power}} \\ &= \frac{E/T}{J} = \frac{E}{JT} \end{aligned}$$

We may therefore write

$$10 \log_{10} \left[\frac{(\text{SNR})_0}{(\text{SNR})_I} \right] = 10 \log_{10} \left(\frac{2T}{T_c} \right) = 3 + 10 \log_{10} \left(\frac{T}{T_c} \right)$$

The QPSK processing gain = T/T_c

$$= \frac{2T_b}{T_c}$$

That is,

$$PG_{\text{QPSK}} = 2[PG_{\text{BPSK}}]$$

Solving for the antenna aperture:

Problem 7.12

The processing gain (PG) is

$$\begin{aligned} \text{PG} &= \frac{\text{FH bandwidth}}{\text{symbol rate}} \\ &= \frac{W_c}{R_s} \\ &= 5 \times 4 = 20 \end{aligned}$$

Hence, expressed in decibels,

$$\begin{aligned} \text{PG} &= 10\log_{10} 20 \\ &= 26 \text{ db} \end{aligned}$$

Problem 7.13

The processing gain is

$$\begin{aligned} \text{PG} &= 4 \times 4 \\ &= 16 \end{aligned}$$

Hence, in decibels,

$$\begin{aligned} \text{PG} &= 10\log_{10} 16 \\ &= 12 \text{ dB} \end{aligned}$$

Problem 7.13

Matlab codes

```
% Problem 7.13(a), CS: Haykin
% Generating 63-chip PN sequences
% polynomial1(x) = x^6 + x + 1
% polynomial2(x) = x^6 + x^5 + x^2 + x + 1
% Mathini Sellathurai, 10.05.1999

% polynomials
pol1=[1 0 0 0 0 1 1];
pol2=[1 1 0 0 1 1 1];

% chip size
N=63;

% generating the PN sequence
pnseq1 = PNseq(pol1);
pnseq2 = PNseq(pol2);

% mapping antipodal signals (0-->-1, 1-->1)
u=2*pnseq1-1;
v=2*pnseq2-1;
```

```

% autocorrelation of pnseq1
[corr1]=pn_corr(u, u, N)

% prints
plot(-61:62,corr1(2:125)); axis([-62, 62,-10, 80])
xlabel(' Delay \tau')
ylabel(' Autocorrelation function R_{c}(\tau)')

pause

%autocorrelation of pnseq2
[corr2]=pn_corr(v, v, N)

% prints
plot(-61:62,corr2(2:125)); axis([-62, 62,-10, 80])
xlabel(' Delay \tau')
ylabel(' Autocorrelation function R_{c}(\tau)')

pause

% cross correlation of pnseq1, pnseq2
[c_corr]=pn_corr(u, v, N)

% prints
plot(-61:62,c_corr(2:125)); axis([-62, 62,-20, 20])
xlabel(' Delay \tau')
ylabel(' Cross-correlation function R_{ji}(\tau)')

```



```

% Problem 7.13 (b), CS: Haykin
% Generating 63-chip PN sequences
% polynomial1(x) = x^6 + x + 1
% polynomial2(x) = x^6 + x^5 + x^2 + x + 1
% Mathini Sellathurai, 10.05.1999

% polynomials
pol1=[1 1 1 0 0 1 1];
pol2=[1 1 0 0 1 1 1];

% chip size
N=63;

% generating the PN sequence
pnseq1 = PNseq(pol1);
pnseq2 = PNseq(pol2);

% mapping antipodal signals (0-->-1, 1-->1)
u=2*pnseq1-1;
v=2*pnseq2-1;

% autocorrelation of pnseq1
[corrf]=pn_corr(u, u, N)

% prints
plot(-61:62,corrf(2:125)); axis([-62, 62,-10, 80])
xlabel(' Delay \tau')
ylabel(' Autocorrelation function R_{c}(\tau)')

pause

%autocorrelation of pnseq2
[corrf]=pn_corr(v, v, N)

% prints
plot(-61:62,corrf(2:125)); axis([-62, 62,-10, 80])
xlabel(' Delay \tau')
ylabel(' Autocorrelation function R_{c}(\tau)')

pause

% cross correlation of pnseq1, pnseq2
[c_corr]=pn_corr(u, v, N)

% prints

```

```
plot(-61:62,c_corr(2:125)); axis([-62, 62,-20, 20])  
xlabel(' Delay \tau')  
ylabel(' Cross-correlation function R_{ji}(\tau)')
```

```

function x = PNseq(p)
% Linear shift register for generating PN sequence of polynomial p
% used for problems 7.13, 7.14 of CS: Haykin
% Mathini Sellathurai, 10.05.1999

N = length(p) - 1; % order of the polynomial
p = fliplr(p);
X = [1 zeros(1, N-1)];
n = 1;

for i = 1 : n*(2^N - 1)
    x(i) = X(1);
    X = [X(2:N) p(N+1) * rem(sum(p(1:N) .* X(1:N)), 2)];
end

```

0

```
function [corrf]=pn_corr(u, v, N)

% funtion to compute the autocorreation/ cross-correlation
% function of two PN sequences
% used in problem 7.13, 7.14, CS: Haykin
% Mathini Sellathurai, 10 june 1999.

max_cross_corr=0;

for m=0:N
    shifted_u=[u(m+1:N) u(1:m)];
    corr(m+1)=(sum(v.*shifted_u));
    if (abs(corr)>max_cross_corr)
        max_cross_corr=abs(corr);
    end
end

corr1=flipud(corr);
corrf=[corr1(2:N) corr];
```

Answer to Problem 7.13

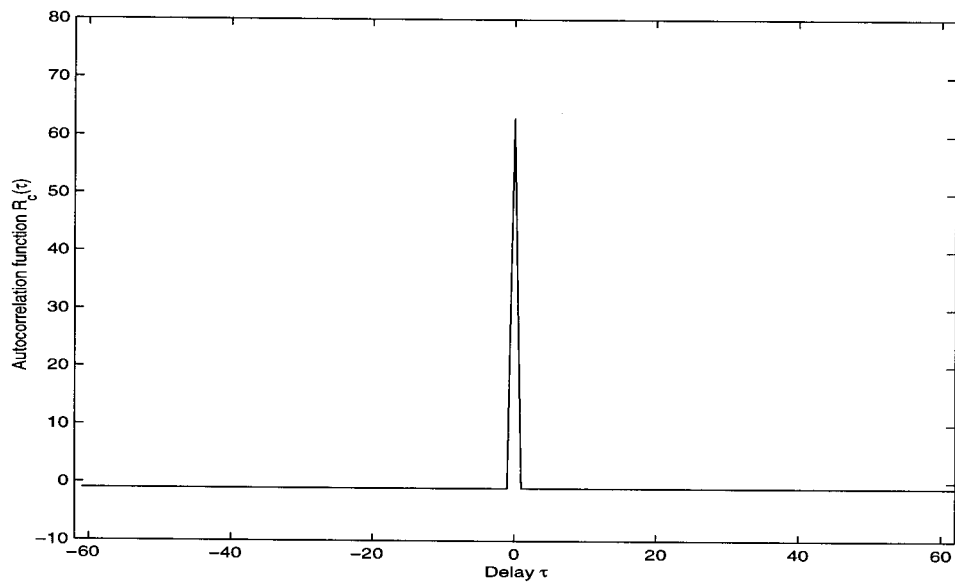


Figure 1: Autocorrelation function of $[6,5,2,1],[6,1]$

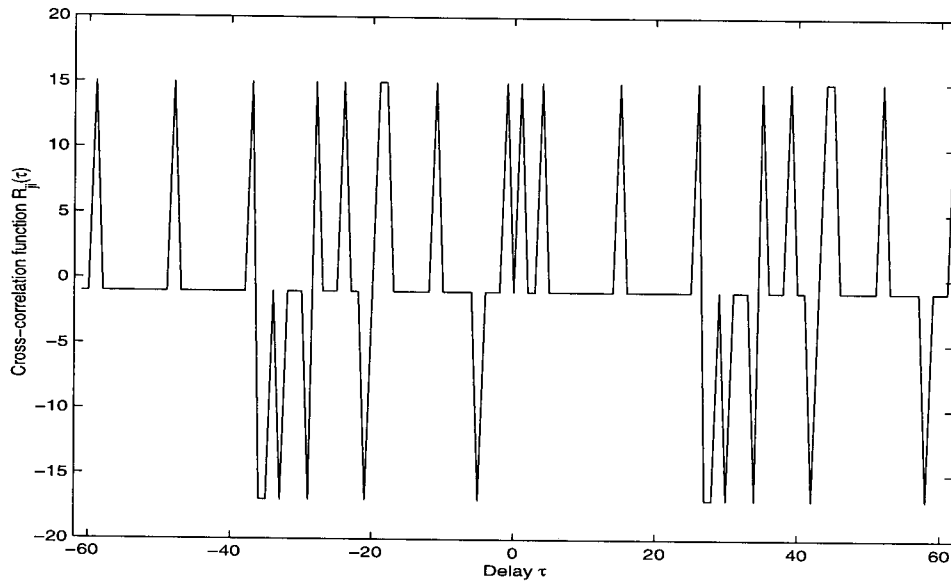


Figure 2: Cross-correlation function of $[6,5,2,1],[6,1]$

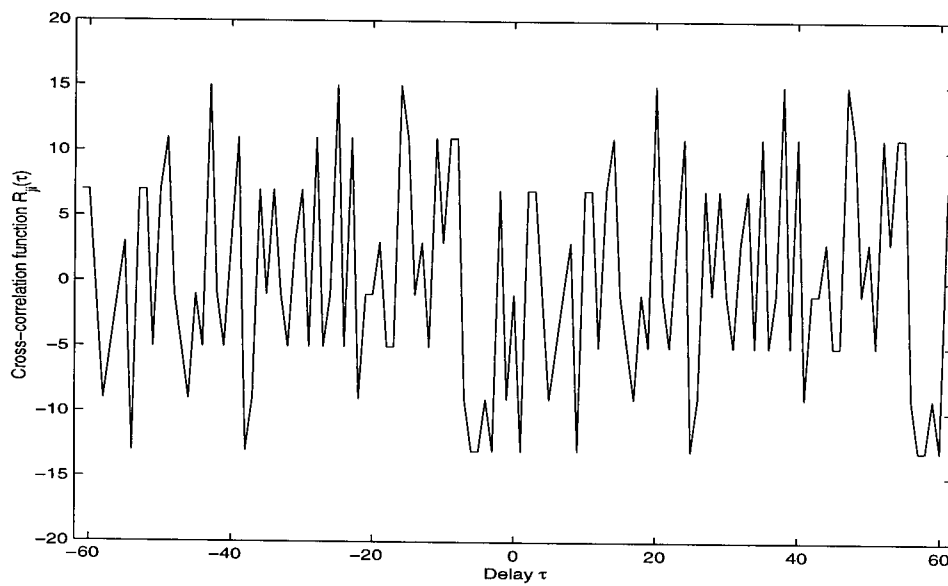


Figure 3: Cross-correlation function of $[6,5,2,1],[6,5,4,1]$

Problem 7.14

Matlab codes

```
% Problem 7.14 (a), CS: Haykin
% Generating 31-chip PN sequences
% polynomial1(x) = x^5 + x^2 + 1
% polynomial2(x) = x^5 + x^3 + 1
% Mathini Sellathurai, 10.05.1999

% polynomials
pol1=[1 0 0 1 0 1];
pol2=[1 0 1 0 0 1];

% chip size
N=31;

% generating the PN sequence
pnseq1 = PNseq(pol1);
pnseq2 = PNseq(pol2);

% mapping antipodal signals (0-->-1, 1-->1)
u=2*pnseq1-1;
v=2*pnseq2-1;

% cross correlation of pnseq1, pnseq2
[c_corr]=pn_corr(u, v, N)

% prints
plot(-30:31,c_corr); axis([-30, 31,-15, 15])
xlabel(' Delay \tau')
ylabel(' Cross-correlation function R_{ji}(\tau)')
```

```

% Problem 7.14 (b), CS: Haykin
% Generating 63-chip PN sequences
% polynomial1(x) = x^5 + x^3 + 1
% polynomial2(x) = x^5 + x^4 + x^2 + x + 1
% Mathini Sellathurai, 10.05.1999

% polynomials
pol1=[1 0 1 0 0 1];
pol2=[1 1 0 1 1 1];

% chip size
N=31;

% generating the PN sequence
pnseq1 = PNseq(pol1);
pnseq2 = PNseq(pol2);

% mapping antipodal signals (0-->-1, 1-->1)
u=2*pnseq1-1;
v=2*pnseq2-1;

% cross correlation of pnseq1, pnseq2
[c_corr]=pn_corr(u, v, N)

% prints
plot(-30:31,c_corr); axis([-30, 31,-10, 10])
xlabel(' Delay \tau')
ylabel(' Cross-correlation function R_{ji}(\tau)')

```



```

% Problem 7.14 (c), CS: Haykin
% Generating 63-chip PN sequences
% polynomial1(x) = x^5 + x^4 + x^3+1
% polynomial2(x) = x^5 + x^4 + x^2 + x + 1
% Mathini Sellathurai, 10.05.1999

% polynomials
pol1=[1 1 1 1 0 1];
pol2=[1 1 0 1 1 1];

% chip size
N=31;

% generating the PN sequence
pnseq1 = PNseq(pol1);
pnseq2 = PNseq(pol2);

% mapping antipodal signals (0-->-1, 1-->1)
u=2*pnseq1-1;
v=2*pnseq2-1;

% cross correlation of pnseq1, pnseq2
[c_corr]=pn_corr(u, v, N)

% prints
plot(-30:31,c_corr); axis([-30, 30,-10, 10])
xlabel(' Delay \tau')
ylabel(' Cross-correlation function R_{ji}(\tau)')

```

Answer to Problem 7.14

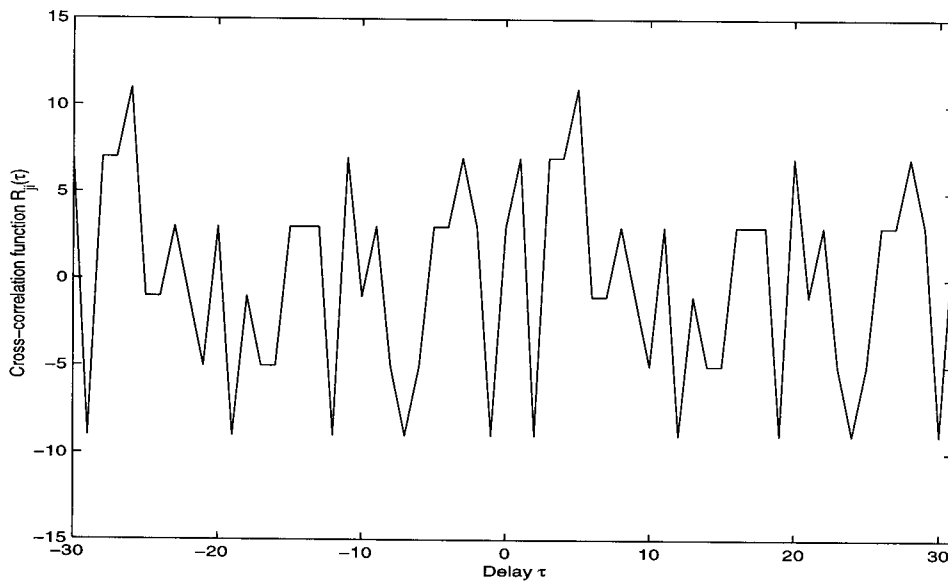


Figure 1: Cross-correlation function of [5,3],[5,2]

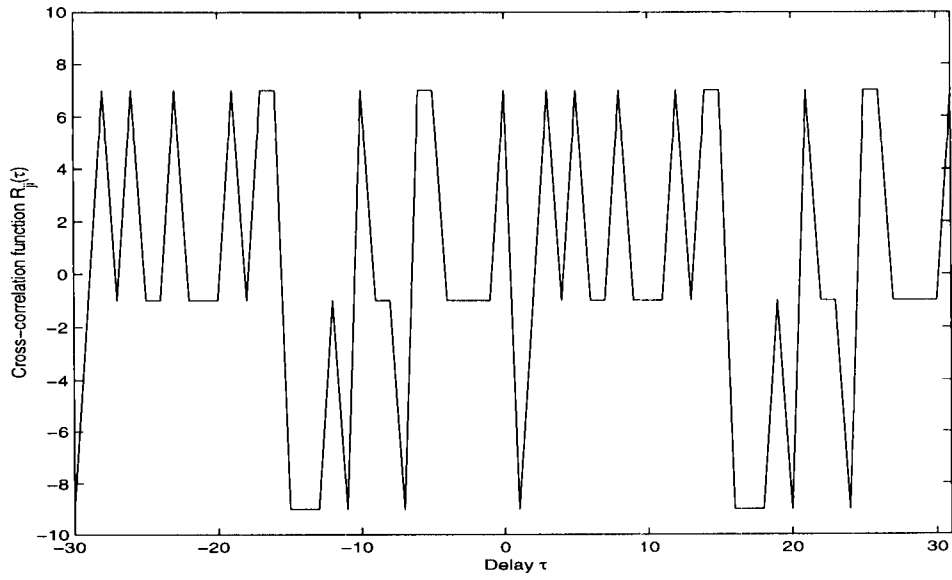


Figure 2: Cross-correlation function of $[6,5,2,1],[6,1]$

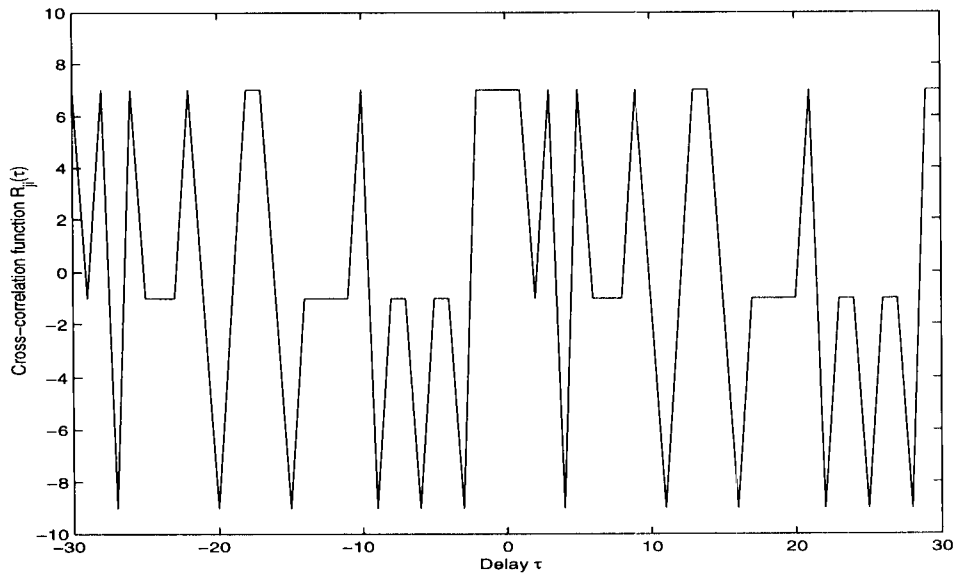


Figure 3: Cross-correlation function of $[6,5,2,1],[6,5,4,1]$