The program must be saved in the same directory as `mvalue.m`. Save it with the name `disper.m`.

So far, we have made two subfunctions. Next, we will try to make the main function. Please open a new file, and then make a program and save it with the program name `main.m` in the same directory as the subfunctions. The `main.m` is given Program 2.3 where random data is generated automatically, and the values of mean and dispersion are calculated. After making the program, enter the directory that has `mvalue.m`, `disper.m`, and `main.m` and execute `main.m`, then you should obtain the data below.

```
>> main
meanvalue = 0.522608
dispersion = 0.111834
standard deviation=0.334415
```

As a result, we can utilize mean, dispersion, and standard deviation values in main function and shrink the volume of main function. In this book, we will produce many subfunctions. By the end you should have a good database for the functions.

2.3 Data Generation and Bit Error Rate

We analyze the bit error rate to evaluate system performance. This section defines bit error probability and shows how to calculate the BER with computer software with a few examples. Using the above example, we generate data, consisting of a 1-by-10 vector and the element in the data consists of 0 or 1. The vector, which is named `txdata`, is given as follows:

```
>> txdata = rand(1,10) > 0.5
txdata =
  1   0   1   0   1   1   0   0   1   0
```

If an error occurs on the communication channel and `txdata(1,7)` changes from 0 to 1, the received data `rxdata` is given as follows

```
>> rxdata = txdata ;
>> rxdata(1,7)=1

>> rxdata(1,9)=0
rxdata =
  1   0   1   0   1   1   1   0   0   0,
```

where the underlined data represents errors. By using transmitted data, `txdata`, and received data, `rxdata`, we count the number of transmitted pieces of data and the number of errors.

To count the number of transmitted pieces of data, we need only the length of txdata. MATLAB has a good command for the calculation of vector size—that is, length. We define the number of transmitted data as nod:

```
>> nod = length(txdata)
nod =
   10
```

To calculate the number of errors, we execute the following procedure. First, we subtract the transmitted data txdata from the received data rxdata. If no error exists, you obtain a zero vector with the length of nod. Otherwise, you obtain a nonzero vector in which "-1" or "1" data occurs at the error positions. The subtraction vector is defined as subdata, which is given as follows:

```
>> subdata=rxdata-txdata
subdata =
   0   0   0   0   0   0   1   0   -1   0
```

As you can see in the vector subdata, if an element in txdata changes from 0 to 1 because of an error, the element of vector subdata at the error position becomes "1". On the other hand, if an element in txdata changes from 1 to 0 because of an error, the element of vector subdata at the error position becomes "-1". By taking the absolute value of the subdata elements, we can make vector that has "1" at each error element.

```
>> abs(subdata)
ans =
   0   0   0   0   0   0   1   0   1   0
```

By adding all elements in the vector abs(subdata), we can calculate the number of errors. For the element summation, MATLAB also has a good command, "sum." If the number of errors is noe, we obtain, by utilizing the command "sum,"

```
>> noe=sum(abs(subdata))
noe =
   2
```

Therefore, we can calculate the bit error probability by using noe and nod. We define the bit error rate as *ber*, which is given by dividing noe by nod as follows:

```
>> ber=noe/nod
ber =
   0.2000
```

We use this procedure to obtain the BER throughout this book.

2.4 Definition of a Radio Communication Channel

This book analyzes system performance under AWGN and/or multipath fading environments. This section explains AWGN and multipath fading and shows how to simulate an AWGN and multipath fading environment by MATLAB.

2.4.1 AWGN Channel

If we construct a mathematical model for the signal at the input of the receiver, the channel is assumed to corrupt the signal by the addition of white Gaussian noise, as illustrated in Figure 2.2 [2]. When we define transmitted signal, white Gaussian noise, and received signal as $s(t)$, $n(t)$, and $r(t)$, the received signal is

$$r(t) = s(t) + n(t) \tag{2.4}$$

where $n(t)$ is a sample function of the AWGN process with probability density function (pdf) and power spectral density as follows:

$$\Phi_{nn}(f) = \frac{1}{2} N_0 [\text{W/Hz}] \tag{2.5}$$

where N_0 is a constant and often called the noise power density. To simulate in MATLAB, we simply use the built-in function `randn`, which generates random numbers and matrices whose elements are normally distributed with mean 0 and variance 1. Therefore, if we add AWGN noise with power 1 to the digital

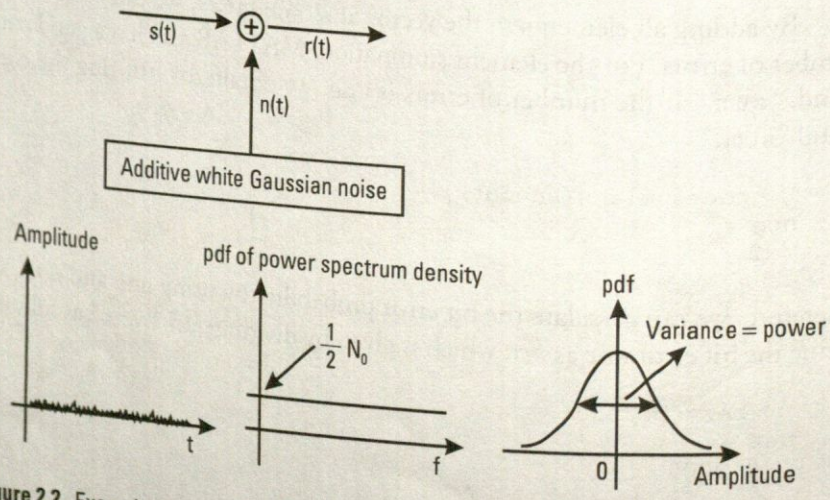

Figure 2.2 Example of an AWGN channel.

modulation signal with in-phase channel (I-channel) and quadrature-phase channel (Q-channel) data vectors, *idata* and *qdata*, respectively, the output data of I channel and Q channel, *iout* and *qout*, are given as follows:

$$\begin{aligned} iout(t) &= idata(t) + randn(t) \\ qout(t) &= qdata(t) + randn(t) \end{aligned} \tag{2.6}$$

However, in the simulation, we sometimes calculate the BER performance by varying the noise power, where we define the noise power as a variable, *npow*. *idata* and *qdata* are voltages, not powers. Therefore, we must change the notation of *npow* from power to voltage. We define a variable *attn* as the root of *npow* as

$$attn = \frac{1}{2}\sqrt{npow} \tag{2.7}$$

Therefore, the revised output data after contamination from noise with a power of *npow* becomes

$$\begin{aligned} iout(t) &= idata(t) + attn \times randn(t) \\ qout(t) &= qdata(t) + attn \times randn(t) \end{aligned} \tag{2.8}$$

Program 2.4 can add AWGN to the input data, which is expressed as digital quadrature phase modulation. For the input data, we use *idata* and *qdata*. The output data are *iout* and *qout*. We only input *attn*, *idata*, and *qdata*, to set the noise-contaminated signal.

2.4.2 Rayleigh Fading Channel

The path between the base station and mobile stations of terrestrial mobile communications is characterized by various obstacles and reflections. For example, an indoor environment has business machines and furniture, and buildings and trees constitute an outdoor environment. These have a large influence on the received signal, when the radio wave is propagated from the base station to the mobile station. The general characteristics of radio wave propagation in terrestrial mobile communications are shown in Figure 2.3. The radio wave transmitted from a base station radiates in all directions these radio waves, including reflected waves that are reflected off of various obstacles, diffracted waves, scattering waves, and the direct wave from the base station to the mobile station. In this case, since the path lengths of the direct, reflected,

3

PSK-Based Digital Transmission Schemes

3.1 Introduction

When we transmit digital data like +1 or −1 by using radio waves, the best way is to modulate carrier signals with frequency f_c in accordance with the information of digital information data. The meaning of "modulate" is to vary the peculiar component that is included in the carrier signal wave. The waveform of the carrier signal is written as follows:

$$S(t) = A\cos\{2\pi f_c t + \theta(t)\} \tag{3.1}$$

where A, f_c, and $\theta(t)$ and are the amplitude, center frequency, and time-variant phase of the carrier wave signal, respectively. In (3.1), we have three peculiar components by which users can change the value. These three are amplitude, frequency, and phase, and if we change the amplitude of (3.1) in accordance with the digital information data, we call the modulation scheme AM. Moreover, if we change the frequency of (3.1) with information digital data, then we call the modulation scheme FM. Finally, if we change the phase of (3.1) in accordance with the digital information data, we call the modulation scheme PM or PSK.

This section focuses on PSK-based digital communication, describes the basic configurations of the transmitter and receiver, and explains how to express the configurations by using computer simulations. Section 3.2 describes BPSK. Then, Section 3.3 explains QPSK to increase the transmission

rate with high-frequency utilization efficiency in comparison with BPSK. QPSK has several problems regarding its shared bandwidth from the viewpoint of the development of the prototype. To solve these problems, we introduce three modulation schemes: OQPSK, MSK, and GMSK in Sections 3.4–3.6, respectively. Moreover, to realize broadband data transmission in the limited bandwidth, we introduce QAM in Section 3.7. Each section also shows how to evaluate these systems by computer simulation. Finally, Section 3.8 concludes the chapter with a brief summary. All programs related to the chapter are presented in Appendix 3A and in the CD-ROM that accompanies the book.

3.2 BPSK

3.2.1 Basic Configuration of BPSK Transmission Scheme

In the modulation scheme, the input digital data 0 or 1 is directly converted to the phase of 0 or π, respectively. Therefore, the waveform is shown as follows:

$$S(t) = A\cos\{2\pi f_c t + \pi \cdot d_k\} \tag{3.2}$$

where d_k is the information data sequence.

Figure 3.1 illustrates a BPSK signal generation method. As shown in Figure 3.1, the waveform of a BPSK wave is generated by multiplying between the digital signal data and carrier wave. However, as for the limitation of the frequency bandwidth, we must control the shape with an adequate pulse-shaping filter. Therefore, in the procedure of BPSK signal generation, first of all, digital data are fed into a pulse-shaping filter circuit. Section 3.2.3 describes the method for the design of the adequate pulse-shaping. Then, the pulse-shaped signal is converted to an analog signal by a D/A converter, upconverted to the RF frequency by multiplying by carrier signal wave, and finally transmitted to the air.

At the receiver, the received wave passes a *bandpass filter* (BPF), where the spurious wave is eliminated. Then, the received signal is downconverted to the baseband by multiplying the received radio signal by the RF carrier frequency signal. Then, the signal is converted to digitally sampled data with an A/D converter, and the transmission digital data is recovered by DSPH. In the DSPH, the sampled data is filtered to eliminate the symbol inter- ference at a pulse-shaping filter circuit. Finally, a synchronization point is selected from the filtered digital sample signal. If the signal level is larger than 0 at the point, we can obtain the received digital data 1; otherwise, the received data becomes 0.

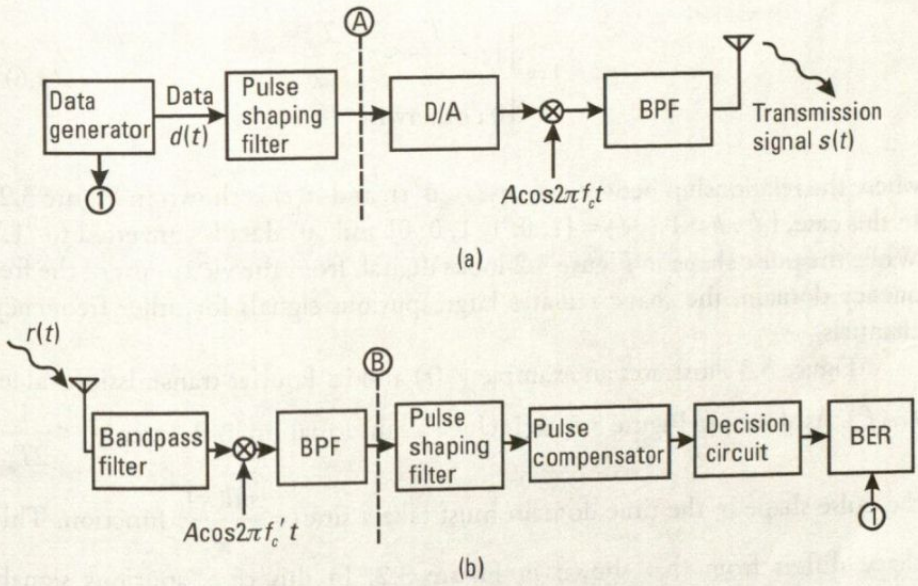

Figure 3.1 (a) Transmitter and (b) receiver of the BPSK transmission scheme.

3.2.2 Theoretical Notation of Signals

We first derived the theoretical education for the BPSK transmission scheme because we needed it for our computer-simulation program.

The transmission digital data sequence is given as:

$$d(t) = \sum_{k=-\infty}^{+\infty} d_k \cdot \left(g_T(t) \otimes \delta(t - kT_b)\right) \tag{3.3}$$

$$= \sum_{k=-\infty}^{+\infty} d_k \cdot g_T(t - kT_b) \tag{3.4}$$

where $d_k(d_k : k = 1, 2, \ldots)$, $g_T(t)$, and T_b are the transmission digital data, the pulse shape of each transmission digital data, and the bit duration, respectively. The reciprocal of T_b is the bit rate, and $\delta(t)$ is a delta function:

$$\delta(t) = \begin{cases} 1 : t = 0 \\ 0 : \text{otherwise} \end{cases} \tag{3.5}$$

When $g_T(t)$ is a trailing rectangular pulse,

$$g_T(t) = \begin{cases} 1; -\dfrac{T_b}{2} \le t \le \dfrac{T_b}{2} \\ 0; \text{otherwise} \end{cases} \tag{3.6}$$

where the relationship between d_k, $g_T(t)$, $\delta(t)$, and $d(t)$ is shown in Figure 3.2. In this case, $\{d_k : k = 1 \ldots 6\} = [1, 0, 1, 1, 0, 0]$ and "0" data is converted to "1." While the pulse shape in Figure 3.2 looks digital, from the viewpoint of the frequency domain, the shape radiates large spurious signals for other frequency channels.

Figure 3.3 illustrates an example $g_T(t)$ and its Fourier transmission value, $G_T(f)$. As shown in Figure 3.3, to include all information in the area $|f| < \dfrac{1}{2T_b}$, the pulse shape in the time domain must take a sinc $\left(= \dfrac{\sin(x)}{x}\right)$ function. This shape differs from that shown in Figure 3.2. In this case, spurious signals appear in the area

$$|t| > \frac{T_b}{2}$$

We thus need a filter with an adequate optimum shape that can reduce the number of spurious signals in the time and frequency domains.

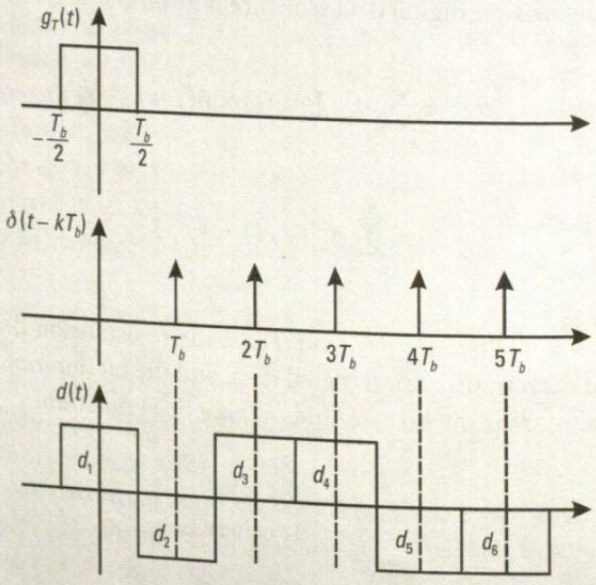

Figure 3.2 Relationship between $g_T(t)$, $\delta(t)$, and $d(t)$.

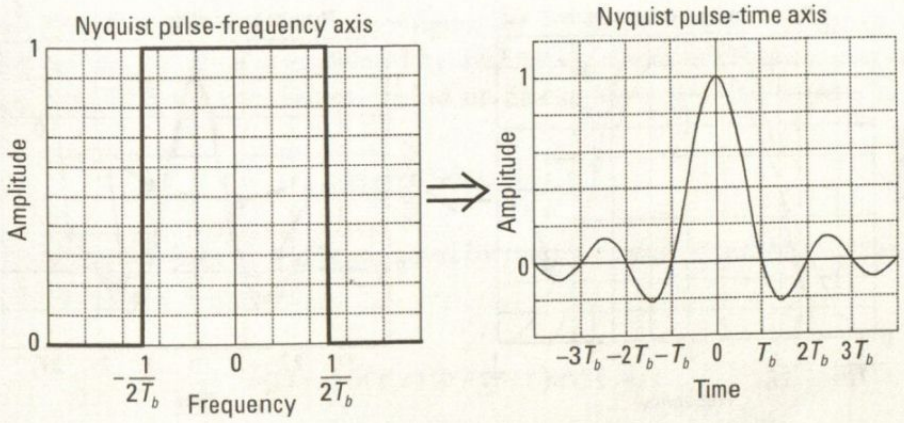

Figure 3.3 An example of $g_T(t)$ and its Fourier transmission value.

A widely used filter that reduces the number of spurious signals is the Nyquist filter. The frequency response of the Nyquist filter is given by [1]

$$G_N(f)=\begin{cases}1 & 0\le|f|\le\dfrac{(1-\alpha)}{2T_b}\\ \cos^2\left[\dfrac{T_b}{4\alpha}\left\{2\pi|f|-\dfrac{\pi(1-\alpha)}{T_b}\right\}\right] & \dfrac{(1-\alpha)}{2T_b}\le|f|\le\dfrac{(1+\alpha)}{2T_b}\\ 0 & |f|>\dfrac{(1+\alpha)}{2T_b}\end{cases} \tag{3.7}$$

where α is the roll-off factor, which determines the channel bandwidth.

Figure 3.4 illustrates $G_N(f)$ in the frequency domain and its impulse response in the time domain. The amplitude of the spurious signals in the frequency domain increases in the area $|f|>\dfrac{1}{2T_b}$ as roll-off factor α increases. In contrast, the amplitude of the spurious signals in the time domain decreases in the area $|t|>T_b$. Therefore, we can find a compromise value for the roll-off factor.

One feature of the Nyquist filter is that we always obtain 0 at nT_b (n is an integer) in the time domain. Therefore, when we set the synchronization point at nT_b, one symbol never interferes with other symbols at this point. Although the Nyquist filter has several useful features, there is a question of when and where to use it for an optimum result. Reference [1] discusses the optimum allocation method to maximize the synchronization point in an

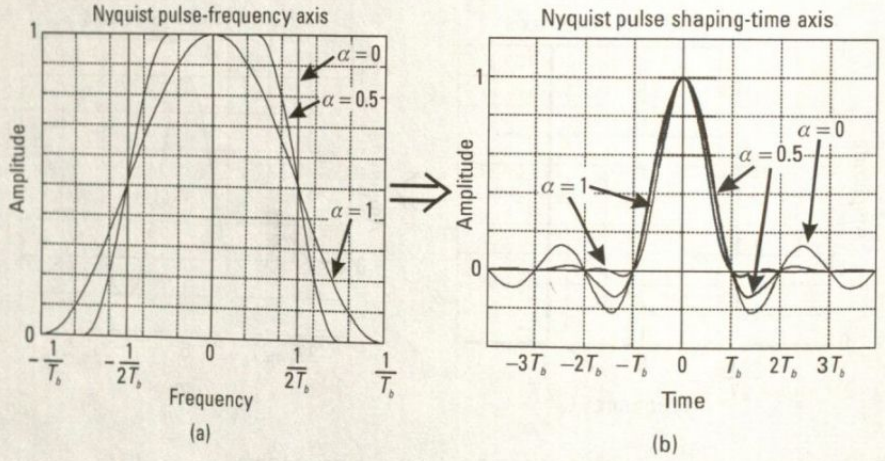

Figure 3.4 Configuration of the Nyquist filter: (a) $G_N(f)$ and (b) impulse response of $G_N(f)$.

AWGN environment. They found that two transmission and receiver filters with an equally distributed gain of $G_N(f)$, namely, the $G_T(f)=\sqrt{G_N(f)}$ and $G_R(f)=\sqrt{G_N(f)}$, where $G_T(f)$ and $G_R(f)$ are the frequency responses of the transmitter and receiver filters:

$$G_T(f)=G_R(f)=\sqrt{G_N(f)}$$

$$=\begin{cases}1 & 0\le|f|\le\dfrac{(1-\alpha)}{2T_b}\\ \cos\left[\dfrac{T_b}{4\alpha}\left\{2\pi|f|-\dfrac{\pi(1-\alpha)}{T_b}\right\}\right] & \dfrac{(1-\alpha)}{2T_b}\le|f|\le\dfrac{(1+\alpha)}{2T_b}\\ 0 & |f|>\dfrac{(1+\alpha)}{2T_b}\end{cases} \tag{3.8}$$

Such a filter is sometimes called a root Nyquist filter, and its impulse responses $g_T(t)$ and $g_R(t)$ are given as

$$g_T(t)=g_R(t)$$

$$=\frac{1}{\pi t}\frac{1}{1-\left(\dfrac{4\alpha t}{T_b}\right)^2}\sin\left\{2\pi(1-\alpha)\frac{1}{T_b}\right\}+\frac{1}{\pi}\frac{4\alpha/T_b}{1-\left(\dfrac{4\alpha t}{T_b}\right)^2}\cos\left\{2\pi(1+\alpha)\frac{1}{T_b}\right\} \tag{3.9}$$

Using (3.4) and (3.6), we can configure the BPSK transmission of the data. The configured signal is modulated by multiplying the carrier frequency signal (e.g., $\cos 2\pi fct$) and then transmitted to air. The transmitted signal is given by

$$s(t) = d(t)\cos 2\pi f_c t \tag{3.10}$$

The transmitted signal is contaminated by multipath fading and AWGN, and at the receiver it is received as

$$r(t) = \int_0^\infty h(\tau, t) \otimes s(t - \tau)d\tau + n(t) \tag{3.11}$$

where $h(\tau, t)$ is the impulse response of the radio channel at time t, and $n(t)$ is the receiver noise.

In the receiver, the received signal is first filtered by a BPF, which is assumed to have a sufficient passband so that the signal is negligibly distorted. The filtered signal is multiplied by a carrier wave that has the same frequency as the transmitter. However, the initial phase of the carrier signal source is different between the transmitter and receiver. Therefore, the carrier signal source at the receiver is mentioned as $\cos(2\pi f_c + \theta_1(t))$ where $\theta_1(t)$ is the initial phase value of the carrier signal. The multiplied signal is given by

$$r_2(t) = r(t) \times \cos(2\pi f_c t + \theta_1(t)) \tag{3.12}$$

where $r(t)$ is given by (3.11). For simplification we assume no filtering, so

$$\begin{aligned} r(t) &= R(t)\begin{pmatrix} \cos\theta_f(t) & -\sin\theta_f(t) \\ \sin\theta_f(t) & \cos\theta_f(t) \end{pmatrix}\begin{pmatrix} d(t)\cos 2\pi f_c t \\ 0 \end{pmatrix} + n(t) \\ &= \left[R(t)d(t)\left(\cos\theta_f(t) + n_1(t)\right)\right]\cos 2\pi f_c t + n_2(t) \end{aligned} \tag{3.13}$$

where $R(t)$ and $\theta_f(t)$ are the time-variant fluctuations of the amplitude and phase, respectively, by radio channel. The $n(t)$ is the receiver noise; it is given by $n(t) = R(t) \times n_1(t) \times \cos 2\pi f_c t + n_2(t)$, where $n_1(t)$ and $n_2(t)$ are the noise component around f_c and the other noise component, respectively. In addition,

$$\begin{pmatrix} \cos\theta_f(t) & -\sin\theta_f(t) \\ \sin\theta_f(t) & \cos\theta_f(t) \end{pmatrix}$$

is a rotation matrix with the phase of $\theta_f(t)$. Using (3.12), we can expand (3.11). When we use LPF for the expansion value, the high-frequency signal is eliminated, so

$$r_3(t) = \frac{1}{2}\Big(R(t)d(t)\big(\cos\theta_f(t) + n_1(t)\big)\Big)\cos\theta_1(t) \tag{3.14}$$

Then, $r_3(t)$ is filtered by a pulse-shaping, filter-based root Nyquist filter to reduce ISI.

$$\begin{aligned} r_4(t) &= r_3(t) \otimes g_R(t) \\ &= \frac{1}{2}R(t)\cos\theta_f(t)\cos\theta_1(t)\cdot\sum_{k=-\infty}^{+\infty} d_k \\ &\cdot\big(g_T(t)\otimes g_R(t)\otimes\delta(t-kT_b)\big) + \frac{1}{2}n_1(t)\big(\cos\theta_1(t)\otimes g_R(t)\big) \\ &= \frac{1}{2}R(t)\cos\theta_f(t)\cos\theta_1(t)\cdot\sum_{k=-\infty}^{+\infty} d_k \\ &\cdot\big(g_T(t)\otimes g_R(t)\otimes\delta(t-kT_b)\big) + n_3(t) \end{aligned} \tag{3.15}$$

Next, the filtered signal is oversampled at a sampling rate of $\frac{n}{T_b}$ (n is an integer) by using an A/D converter. The sampled signal is

$$r_4\left(k\cdot\frac{T_b}{n}\right) k = 0, 1, 2, \ldots$$

When $t = u \cdot T_b/n$ (u is an integer), we can obtain the maximum value of the pulse from the characteristics of the Nyquist filter. However, we must consider the synchronization method. In this section, we assume that we know the synchronization point. Then, we resample

$$r_4\left(u\cdot\frac{T_b}{n}\right)$$

every T_b or every n sample from the synchronization point. Finally, we obtain resampled data $r_5(l) = r_4$ (syncpoint + $l\cdot T_b$) $l = 0, 1, 2, \ldots$. We can then decide whether the received data is a 1 or a 0 by using the threshold condition equation:

$$\hat{d}_k = \begin{cases} 1 & (r_5(l) > 0) \\ 0 & (r_5(l) < 0) \end{cases} \tag{3.16}$$

By comparing $\hat{d}_k$ and d_k, we can calculate the BER, which depends on the volume of receiver noise. Theoretical BER in an AWGN and one-path Rayleigh fading channels have been reported [1–3].

$$\text{BER}_{\text{BPSK-AWGN}} = \frac{1}{2}\text{erfc}\left(\sqrt{E_b / N_0}\right) \tag{3.17}$$

$$\text{BER}_{\text{BPSK-FADING}} = \frac{1}{2}\left[1 - \frac{1}{\sqrt{1 + \dfrac{1}{E_b / N_0}}}\right] \tag{3.18}$$

The E_b/N_0 is the ratio between the energy per bit (E_b) and the noise power density N_0. Figure 3.5 illustrates the theoretical relationship of BPSK between E_b/N_0 (dB) and BER with AWGN and one-path Rayleigh fading channels.

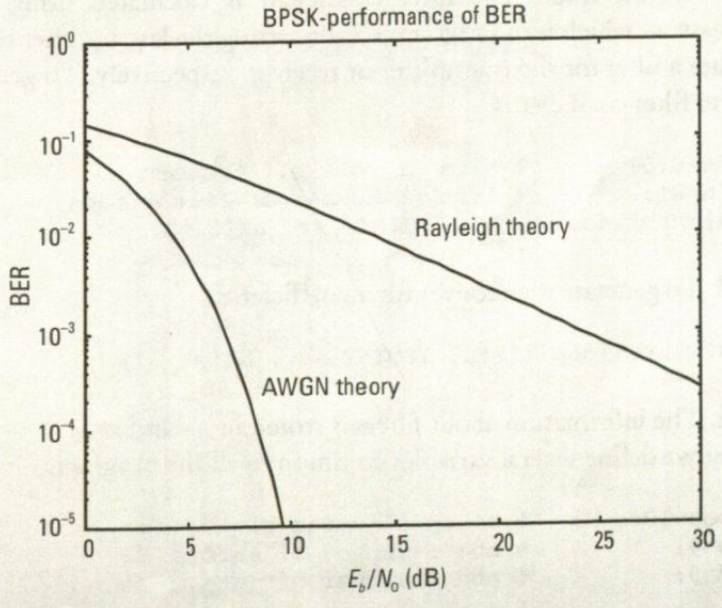

Figure 3.5 BER performance of BPSK transmission scheme under AWGN and Rayleigh fading environments (theoretical value).

3.2.3 Computer Simulation Program

To measure the BER performance of the BPSK scheme, we used computer simulation. The model we used is illustrated in Figure 3.6, and the three simulation programs we used are shown below. All functions used in this chapter are summarized in Table 3.1. Program 3.1 is the main program for BPSK. We will use it here to explain the procedures of the computer simulation.

Before using Program 3.1, we must set the values of the common variables used in the program.

```
sr=256000;          % sr : symbol rate
ml=1;               % ml : number of modulation levels
br=sr.*ml;          % br : bit rate
nd=1000;            % nd : number of symbols
ebn0=100;           % ebn0 : Eb/No
IPOINT=8;           % IPOINT : number of oversamples
```

For this program, we must assume an important point. As Section 3.2 explains, we oversample the received signal at the receiver. Therefore, we assume that the minimum time unit is equal to the oversampling duration. Namely, $\frac{1}{sr \cdot IPOINT}$ is the minimum time unit.

After defining several variables, we must set the filter coefficients of the root Nyquist filter. The filter coefficient is calculated using function `hrollfcoef.m`, which is in Program 3.2. By setting the last argument to 0 or 1, we generate a filter for the transmitter or receiver, respectively. To generate the transmitter filter coefficients

```
alfs=0.5;           % alfs  : roll-off factor
irfn=21;            % irfn  : number of filter taps
[xh]=hrollcoef(irfn, IPOINT, sr, alfs, 1);
```

is entered. To generate the receiver filter coefficients,

```
[xh2]=hrollcoef(irfn, IPOINT, sr, alfs, 0);
```

is entered. The information about filters is stored in `xh` and `xh2`.

Next we define several variables common to all the programs

```
nloop=100;          % nloop : number of loops
noe=0;              % noe : number of errors
nod=0;              % nod : number of data
```

After defining all of the variables, we can run the simulation to obtain the BER. First, we generate random data (0 or 1), in a 1-by-`nd*ml` vector called

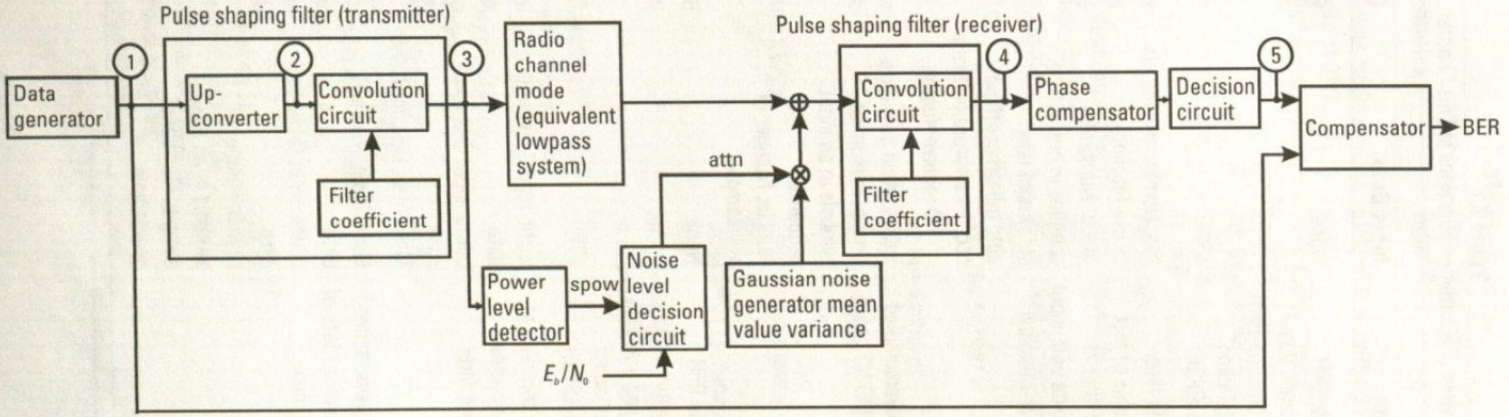

Figure 3.6 Model of the BPSK transmission system used to calculate BER performance.

Table 3.1
Overview of Simulation Functions in This Chapter

Name of Function	Function	Input Data	Output
bpsk.m (Program 3.1) bpsk_fading.m (Program 3.2)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (BPSK)	None	BER: bit error rate performance
hrollfcoef.m (Program 3.3)	Calculate filter coefficients of root Nyquist filter in accordance with input symbol or sampling rate	irfn: Number of symbols to use filtering ipoint: Number of samples in one symbol sr: symbol rate alfs: roll-off coefficient ncc: 1-transmission factor 0-receiver factor	xh: coefficient value in accordance with input symbol or sampling rate
oversamp.m (Program 3.4)	Output oversampled data of IPOINT times	indata: input sequence nsymb: number of symbols in an input sequence sample: Number of oversample	outdata: oversampled data
qpsk.m (Program 3.5) qpsk_fading.m (Program 3.6)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (QPSK)	None	BER: bit error rate performance
compconv.m (Program 3.7)	Perform convolution between complex signal and filter	idata: input I-channel data qdata: input Q-channel data filter: filter tap coefficient	iout: filtered I-channel data qout: filtered Q-channel data
compoversamp.m (Program 3.8)	Output oversampled I and Q channel data of IPOINT times	idata: input I-channel data qdata: input Q-channel data nsymb: number of symbols in an input sample: Number of oversample	iout: oversampled channel data qout: oversampled Q-channel data

Table 3.1 (continued)

Name of Function	Function	Input Data	Output
qpskmod.m (Program 3.9)	Output modulated data of QPSK modulation	paradata: parallel serial data para: the number of parallels nd: the number of symbols ml: the number of modulations	iout: modulated I-channel data qout: modulated Q-channel data
qpskdemod.m (Program 3.10)	Output demodulated data of QPSK modulation	idata: input I-channel data qdata: input Q-channel data para: the number of parallels nd: the number of symbols ml: the number of modulations	demodata: demodulated serial data
comb.m (Program 2.4)	Add AWGN	idata: input I-channel data qdata: input Q-channel data attn: amplitude of noise in each I and Q channel	iout: output of I-channel data qout: output of Q-channel data
oqpsk.m (Program 3.11) oqpsk_fading.m (Program 3.12)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (QPSK)	None	BER: bit error rate performance
msk.m (Program 3.13) msk_fading.m (Program 3.14)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (MSK)	None	BER: bit error rate performance
msk2.m (Program 3.15) msk2_fading.m (Program 3.16)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (MSK)	None	BER: bit error rate performance

Table 3.1 (continued)

Name of Function	Function	Input Data	Output
oversamp2.m (Program 3.17)	Output oversampled I and Q channel data of IPOINT times for MSK	in: input sequence ntot: number of symbols in an input sequence sample: Number of oversample	out: oversampled data
gmsk.m (Program 3.18) gmsk_fading.m (Program 3.19)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (GMSK)	None	BER: bit error rate performance
gaussf.m (Program 3.20)	Form Gaussian filter	B: `filter` coefficient irfn: Number of symbols to use filtering ipoint: Number of samples in one symbol sr: symbol rate ncc: 1-transmission factor 0-receiver factor	xh: coefficient value in accordance with input symbol or sampling rate
qam16.m (Program 3.21) qam16_fading.m (Program 3.22)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (16-QAM)	None	BER: bit error rate performance
qammod.m (Program 3.23)	Output modulated data of 16-QAM modulation	paradata: input serial data para: the number of parallels nd: the number of symbol ml: the number of modulation	iout: modulated I-channel data qout: modulated Q-channel data
qamdemod.m (Program 3.24)	Output demodulated data of 16-QAM modulation	idata: input I-channel data qdata: input Q-channel data para: the number of parallels nd: the number of symbol ml: the number of modulation	demodata: demodulated serial data

data1. Section 2.3 explains the method for generating the data. It is done by simply entering a command:

```
data=rand(1,nd*ml)>0.5;
```

Next, we convert the notation of the data from 0 and 1 digital data to −1 and 1 digital data. The converted data is stored in variable data1:

```
data1= 2.*data-1;
```

The data to be transmitted is shown in Figure 3.7(a). It is pulse-shaped as follows:

1. Using data1, we make an impulse train sequence $d(t) = \sum_{k=-\infty}^{+\infty} d_k \delta(t - kT_b)$ as shown in (3.2), where the minimum time duration is 1/(sr·IPOINT). The function oversample.m is used to generate the impulse train sequence as follows

   ```
   data2=oversamp(data1,nd, IPOINT);
   ```

   The waveform of data2 is shown in Figure 3.7(b).

2. To generate the impulse train sequence, we perform convolution using filter coefficient xh. The filtered data is given by

   ```
   data3=conv(data2, xh);
   ```

   The waveform of data3 is shown in Figure 3.7(c).

The transmission signal is then passed through a radio channel (an equivalent lowpass system) and transmitted. At the receiver, the received signal is first contaminated by AWGN. The noise function used, comb.m, was described in Section 2.4. If we use this function in the simulation, we need not input qch data. Moreover, we must decide variable attn.

We want to determine the relationship between E_b/N_0 and BER. That means we must vary attn while keeping E_b/N_0 constant. The variable attn is calculated as follows. First, energy per bit E_b and noise power density N_0 are defined:

$$E_b = \frac{spow}{br} \quad (\mathrm{W} \cdot T/\mathrm{bit}) \tag{3.19}$$

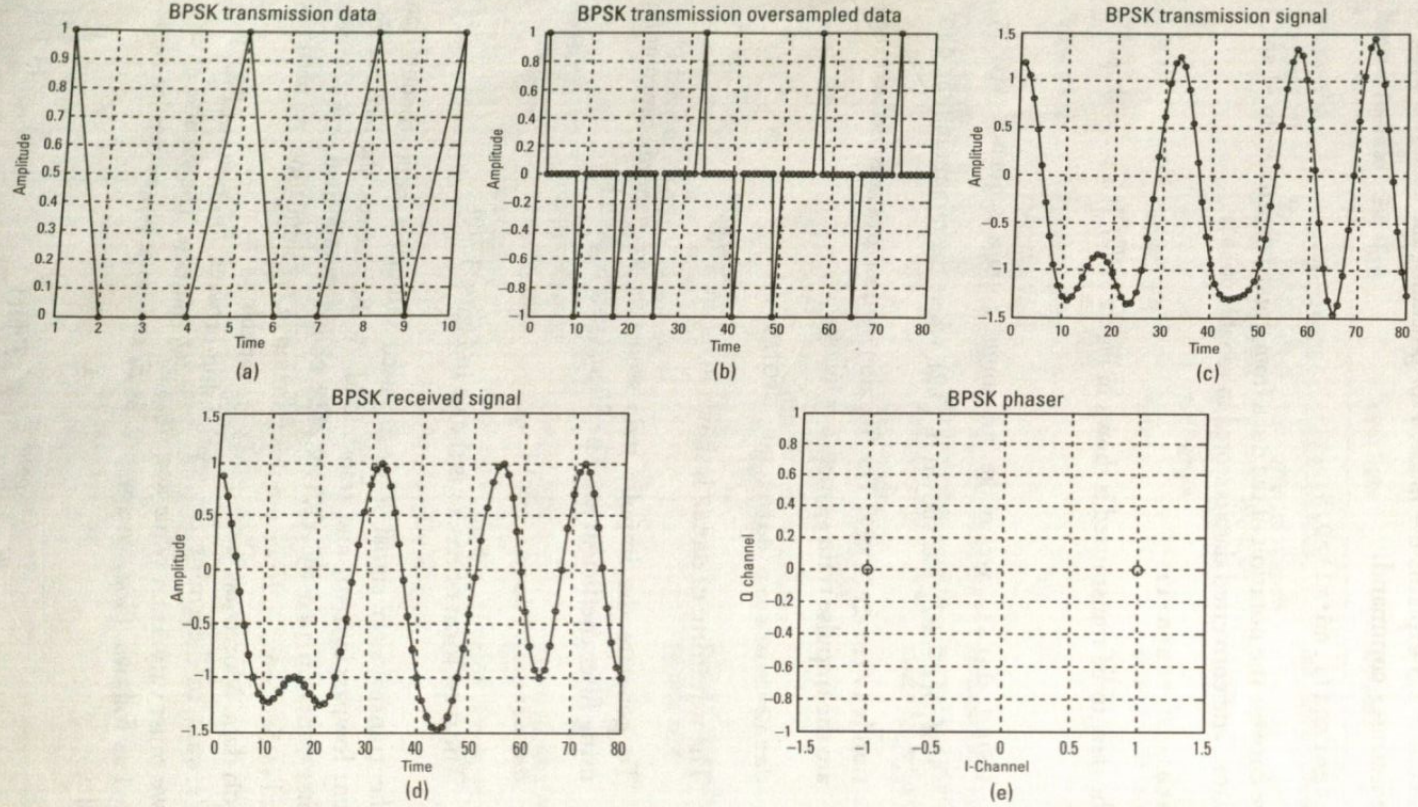

Figure 3.7 Simulated instantaneous BPSK waveform: (a) BPSK transmission data, (b) BPSK transmission oversampled data, (c) BPSK transmission signal, (d) BPSK received signal, and (e) BPSK phaser.

$$N_0 = \frac{npow}{sr} \quad (\text{W/Hz}) \tag{3.20}$$

where `spow`, `npow`, `br`, and `sr` are the signal power per symbol, the noise power per symbol, the bit rate, and the symbol rate, respectively.

From combining (3.19) and (3.20), we get

$$E_b/N_0 = \frac{spow}{br} \cdot \frac{sr}{npow} \tag{3.21}$$

After manipulation, we get

$$npow = \frac{spow}{br} \cdot \frac{sr}{E_b/N_0} \tag{3.22}$$

Since E_b/N_0 is in decibels, (3.22) can be written as

$$npow = \frac{spow}{br} \cdot \frac{sr}{10^{\frac{E_b/N_0}{10}}} \tag{3.23}$$

meaning that we can calculate `npow`. In this simulation `data`, `data2`, and `data3` are expressed in voltage, so the noise signal must be expressed in voltage. Gaussian noise is normally distributed equally in in-phase and quadrature-phase channels. Therefore,

$$attn = \sqrt{\frac{1}{2} npow} \tag{3.24}$$

In Program 3.1, we express these relationships as

```
spow=sum(data3.*data3)/nd;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);
```

Using `attn`, we contaminate the transmitted data with AWGN.

```
inoise=randn(1,length(data3)).*attn;
data4=data3+inoise;
```

where `randn(a,b)` is a built-in function that generates Gaussian noise with mean `a` and variance `b`. The received signal, `data4`, is filtered with a root Nyquist filter, with coefficient `xh2`, which is set at the beginning of the simulation.

```
data5=conv(data4, xh2);
```

To use the `conv.m` function, we must recover the time delay due to convolution, which has `length(xh)`. The time delay of each `conv.m` is $\frac{irfn + IPOINT}{2}$. Since we use two `conv.m` functions, the time delay is `irfn+IPOINT` samples. Therefore, we resample `data5` from sample point `irfn*IPOINT+1`. We define the start point as `sampl`:

```
sample=irfn*IPOINT+1;
```

The filtered signal `data5` is an oversampled signal and shown in Figure 3.7(d). To recover the transmitted data, we must select optimum points from the oversampled signal. As shown in Figure 3.7(d), we can obtain a 1 or −1 value by resampling `data5` every `IPOINT(=8)` samples from the start point. The resampled value is stored in `data6`. From the operation, we can understand that the transmission signal is recovered. The phase of the recovered signal is shown in Figure 3.7(e).

```
data6=data5(sampl: 8:8*nd+sampl-1); .
```

If the received signal is contaminated by AWGN, the value of `data6` is not 1 or 0, and we need to decide whether the value is 0 or 1. The threshold value for deciding is given in (3.16). We use it to execute the following condition function:

```
demodata=data6>0;
```

If each element is larger than 0, the data is considered to be a 1, otherwise 0.

Next, we investigate the number of errors. The procedure was described in Section 2.3. In this simulation, the transmission data is `data`, and, the received data is `data6`.

```
noe2=sum(abs(data-demodata));
nod2=length(data);
noe=noe+noe2;
nod=nod+nod2;
```

Finally, the BER is given by

```
ber=noe/nod;,
```

For the simulation in the Rayleigh fading environment, we can use Program 3.2. The difference between Programs 3.1 and 3.2 is shown as follows. In

Program 3.2 we first set the fading parameters discussed in Section 2.4.2. In this case we try to generate one-path Rayleigh fading.

```
%*************Fading initialization **********

% Time resolution
tstp=1/sr/IPOINT;
% Arrival time for each multipath normalized by tstp
itau = [0];
% Mean power for each multipath normalized by direct wave.
dlvl = [0];
% Number of waves to generate fading for each multipath.
n0=[6];
% Initial phase of delayed wave
th1=[0.0];
% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
itnd1=[1000];
% Number of direct wave + Number of delayed wave
now1=1;
% Maximum Doppler frequency [Hz]
fd=160;
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal
%  (phase and amplitude are fluctuated))
flat =1;
```

Moreover, by using the above parameters, the transmitted BPSK signal `data3` is fluctuated by fading simulator.

```
%******************** Fading channel ********************

% Generated data are fed into a fading simulator

[ifade,qfade]=sefade(data3,zeros(1,length(data3)),itau,
  dlvl,th1,n0,itnd1,now1,length(data3),tstp,fd,flat);

% Updata fading counter
itnd1 = itnd1+ itnd0;
```

The other function is basically the same as Program 3.1. To evaluate BER performance under AWGN and fading environments by using Program 3.1 (`bpsk.m`) and Program 3.2 (`bpsk_fading.m`), we just type the following command

```
clear
bpsk
```

or

```
clear
bpsk_fading
```

The simulated results are shown in Figure 3.8.

As shown in Figure 3.8, the simulated results closely matched the theoretical value. The BPSK scheme has a simple configuration. Several modulation schemes based on BPSK have been proposed to obtain higher frequency-utilization efficiency. Sections 3.3–3.7 introduce QPSK, OQPSK, MSK, GMSK, and QAM as representative examples.

3.3 QPSK

3.3.1 Basic Configuration of Quadrature Modulation Scheme

A QPSK signal is generated by two BPSK signals. To distinguish the two signals, we use two orthogonal carrier signals. One is given by $\cos 2\pi f_c t$, and the other is given by $\sin 2\pi f_c t$. The two carrier signals remain orthogonal in the area of a period

$$\int_0^{T_c} \cos 2\pi f_c t \times \sin 2\pi f_c t = 0 \tag{3.25}$$

where T_c is the period of the carrier signals and

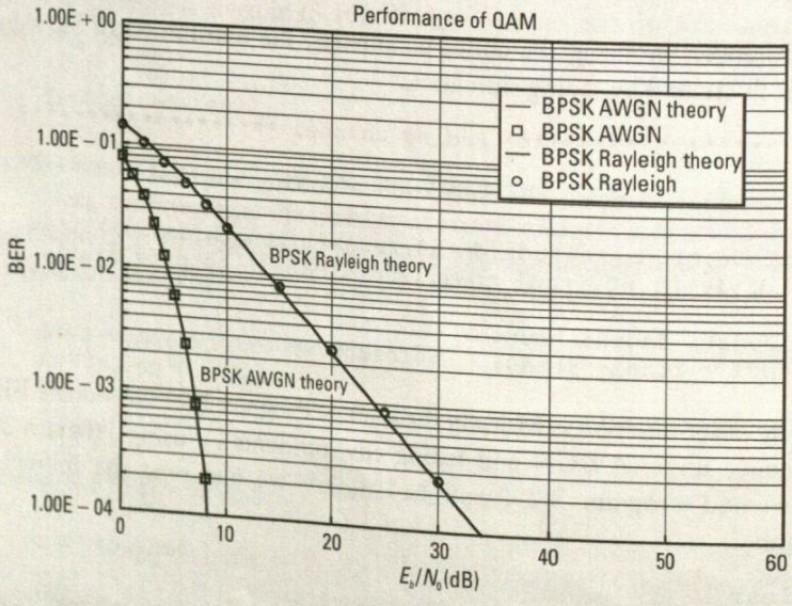

Figure 3.8 Theoretical and simulated BER performance of BPSK under AWGN and fading environments.

$$f_c = \frac{1}{T_c}$$

By using $\cos 2\pi f_c t$ and $\sin 2\pi f_c t$, we can represent QPSK signals by

$$s(t) = \frac{1}{\sqrt{2}} d_I(t)\cos(2\pi f_c t) + \frac{1}{\sqrt{2}} d_Q(t)\sin(2\pi f_c t) \qquad (3.26)$$

A channel in which $\cos 2f_c t$ is used as a carrier signal is generally called an in-phase channel, or Ich, and a channel in which $\sin 2f_c t$ is used as a carrier signal is generally called a quadrature-phase channel, or Qch. Therefore, $d_I(t)$ and $d_Q(t)$ are the data in Ich and Qch, respectively. Modulation schemes that use Ich and Qch are called *quadrature modulation schemes.* The basic configuration is shown in Figure 3.9.

In this system the input digital data, $d_k : k = 1, 2, \ldots$ is first converted into parallel data with two channels, Ich and Qch. The data are represented as $d_I(t)$ and $d_Q(t)$. The data is allocated using a mapping circuit block. Then, the data allocated to Ich is filtered using a pulse-shaping filter in Ich. Then, the pulse-shaped signal is converted into an analog signal by a D/A converter and multiplied by a $\cos 2\pi f_c t$ carrier signal. The data allocated to Qch is also filtered using a pulse-shaping filter. The pulse-shaped signal is converted into an analog signal by a D/A converter and multiplied by a $\sin 2\pi f_c t$ carrier signal. Then, the Ich and Qch signals are added and transmitted to the air.

At the receiver, the received wave first passes through a BPF, where spurious waves are eliminated. The received signal is then down-converted to the baseband by multiplying it by the RF carrier frequency. For Ich and Qch, we use $\cos[2\pi f_c t + \theta_1(t)]$ and $\sin[2\pi f_c t + \theta_1(t)]$, respectively, where $\theta_1(t)$ is the phase noise of the frequency source, the difference between the frequency sources of the transmitter and receiver. Then, in both Ich and Qch channels, the downconverted signal is digitally sampled by A/D converters, and the digital data is fed to DSPH. In the DSPH, the sampled data is filtered with a pulse-shaping filter to eliminate ISI. The signals are then synchronized, and the transmitted digital data is recovered.

Based on this quadrature modulation scheme, Section 3.3.2 discusses several modulation schemes.

3.3.2 Basic Configuration of QPSK Transmission Scheme

QPSK basically uses the configuration shown in Figure 3.9 with several blocks, specialized for QPSK. They are the ones for mapping, demapping, and pulse shaping.

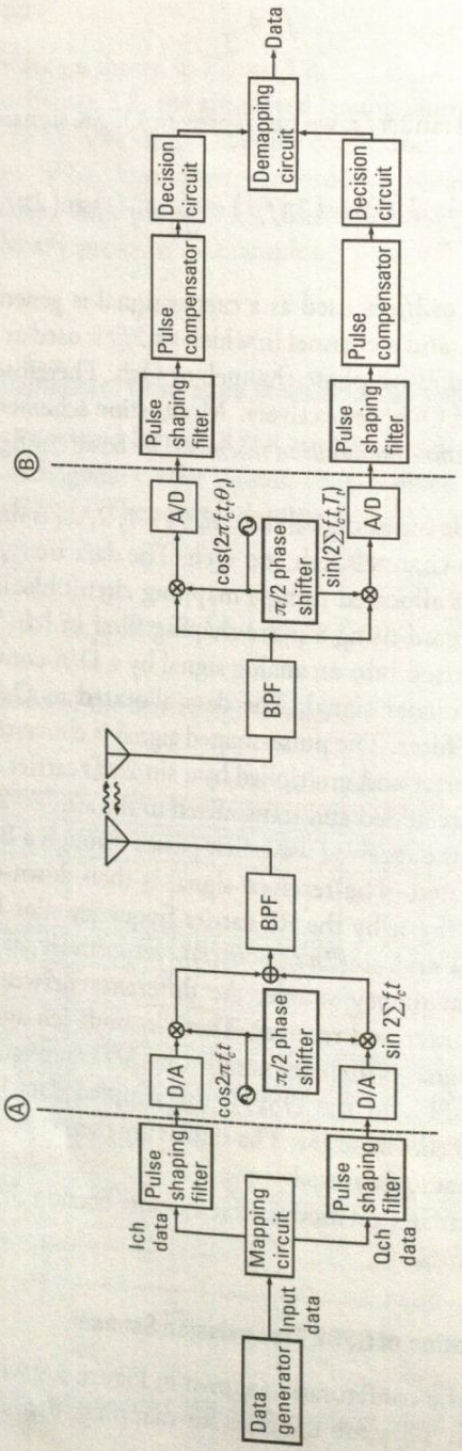

Figure 3.9 Basic configuration of the quadrature modulation scheme.

For mapping and demapping, we use the simple circuit shown in Figure 3.10. For pulse shaping, we use the root Nyquist filter described in Section 3.2.2. The theoretical BER values with AWGN and one-path Rayleigh fading have the QPSK transmission scheme given in [2].

$$\mathrm{BER}_{\mathrm{QPSK\_AWGN}} = \frac{1}{2}\mathrm{erfc}\left(\sqrt{E_b/N_0}\right) \tag{3.27}$$

$$\mathrm{BER}_{\mathrm{QPSK\_FADING}} = \frac{1}{2}\left[1 - \frac{1}{\sqrt{1+\dfrac{1}{E_b/N_0}}}\right] \tag{3.28}$$

In the simulation under the Rayleigh fading environment, we assume that the frequency rotation is compensated for.

3.3.3 Computer Simulation

The model we used to simulate the BER performance of QPSK is shown in Figure 3.11. A comparison of Figure 3.11 to Figure 3.6 shows how quadrature modulation differs from BPSK modulation. In quadrature modulation, there are two simulation flows, Ich and Qch. The simulation of these flows is independent and the same as that for BPSK. The simulation is thus quite easy to understand. The programs are Programs 3.5–3.10. In the main program, Program 3.5, we first decide which common variables are to be used throughout Program 3.5.

```
sr=256000;          % sr : Symbol rate
ml=2;               % ml : number of modulation levels
br=sr. *ml;         % br : bit rate
nd=1000;            % nd : number of symbols
ebn0=3;             % ebn0 : Eb/No
IPOINT=8;           % IPOINT : number of oversamples
```

We next decide the filter coefficients of the root Nyquist filter. The procedures are those used with BPSK (see Section 3.2.3).

Then, we define several variables common to all the programs, the same as with BPSK. (See Section 3.2.3.)

After defining all of the variables, we can run the simulation to obtain the BER. First, we generate random data (0 or 1) in a 1-by-`nd*ml` vector called as `data1`.

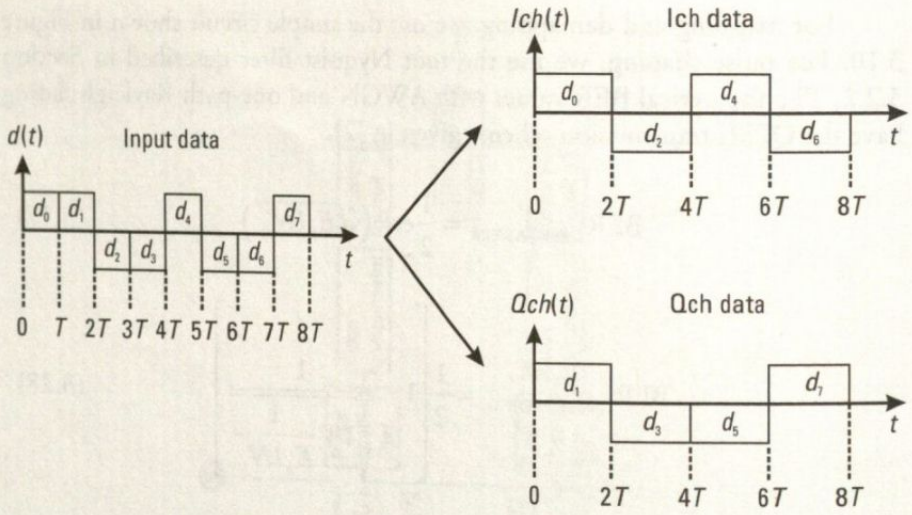

Figure 3.10 Mapping-circuit function for QPSK.

```
data1=rand(1,nd*m1) > 0.5;
```

The transmission data is shown in Figure 3.12(a).

Next, vector `data1` is fed into the mapping circuit, where the serial data is converted into parallel data of two channels, Ich and Qch, using predefined mapping. Then, using `compoversamp.m`, the data in both channels are oversampled and put into an impulse train sequence so that the pulse-shaping filter can be used. Convolution with filter coefficient `xh` is used to generate for the impulse train sequence. The simulation program works as follows:

```
[ich,qch]=qpskmod(data1,1,nd,m1);
[ich1,qch1]= compoversamp(ich,qch,length(ich),
  IPOINT);
[ich2,qch2]= compconv(ich1,qch1,xh);
```

In this procedure, `compoversamp.m` is an extension of `oversamp.m.`, and `compconv.m` becomes an extension of `conv.m`. The instantaneous waveforms, `ich1` and `qch1`, are shown in Figure 3.12(b) and (c).

The filtered signal is transmitted to air, passes through a radio channel (an equivalent lowpass system), and is received by the receiver.

At the receiver, the received signal is first contaminated by AWGN. The noise function, used `comb.m`, was described in Section 2.4.1.

We want to determine the relationship between E_b/N_0 and the BER. That means we must vary `attn` while keeping E_b/N_0 constant. The variable `attn` is calculated using the same procedure as for BPSK. (See Section 3.2.3.)

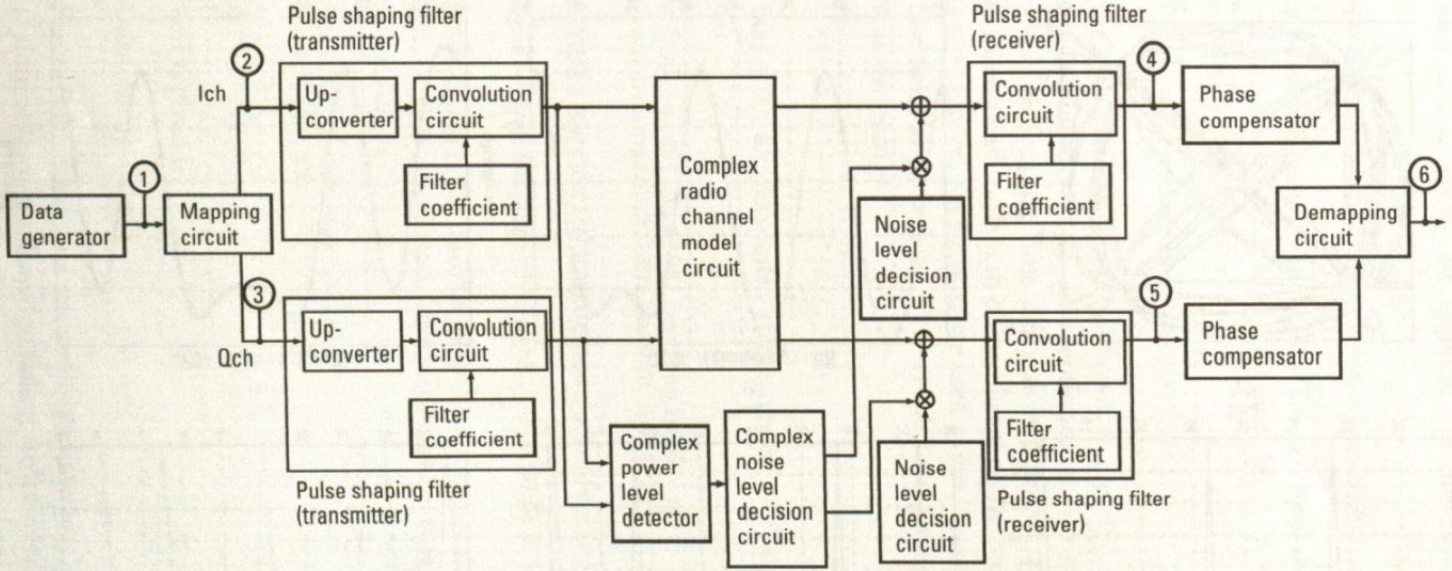

Figure 3.11 Model of quadrature modulation scheme used to simulate BER performance.

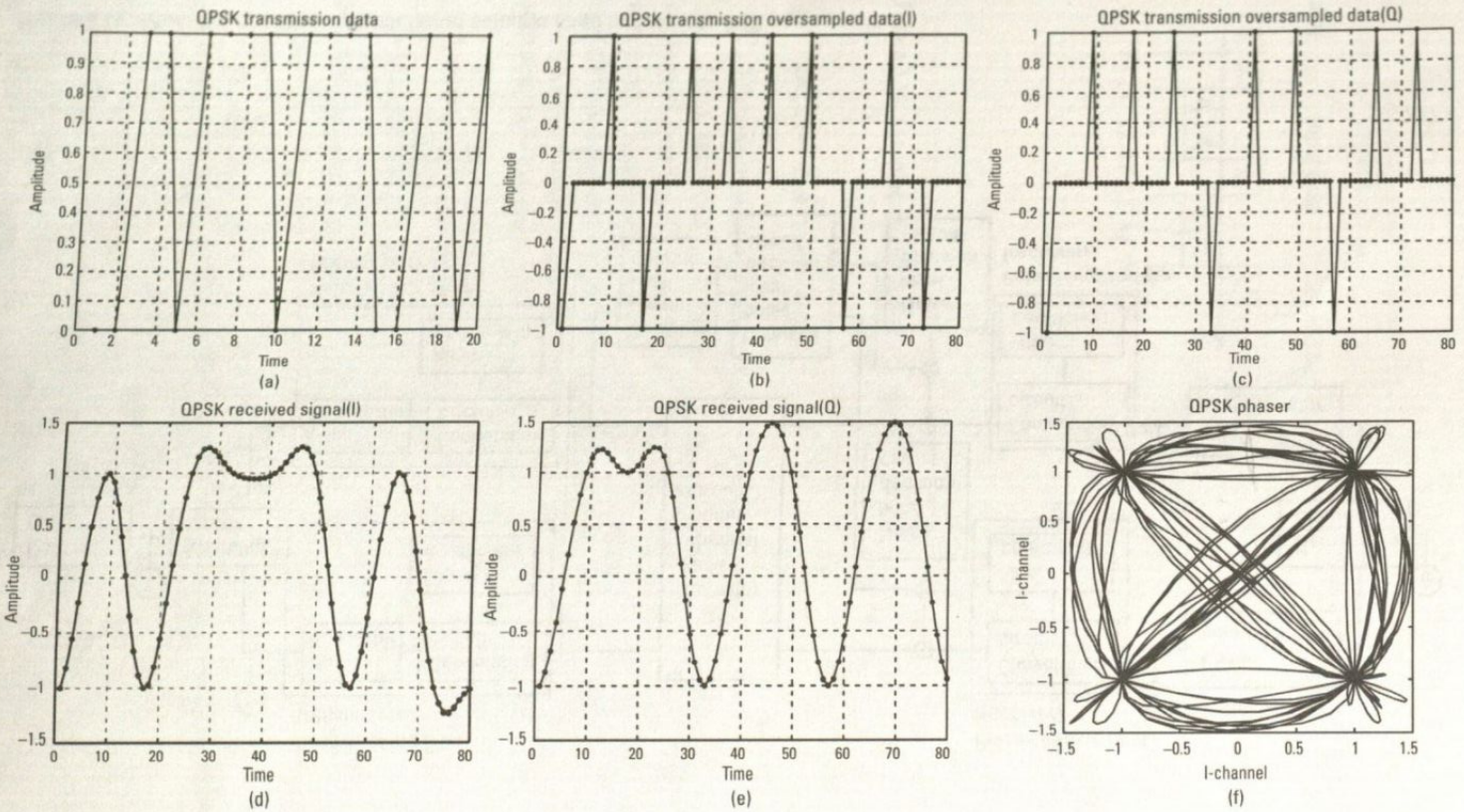

Figure 3.12 Simulated instance waveform: (a) QPSK transmission data, (b) QPSK transmission oversampled data (I), (c) QPSK transmission oversampled data (Q), (d) QPSK received signal (I), (e) QPSK received signal (Q), and (f) QPSK phaser.

Using attn and comb.m, we contaminate the transmitted data with AWGN.

```
[ich3,qch3]= comb(ich2,qch2,length(ich2),attn); ,
```

where `comb.m` is that described in Section 2.4.1. Then, the received signals, `ich3` and `qch3`, are filtered with a root Nyquist filter, in which the received data is correlated with coefficients `xh2`.

```
[ich4,qch4]= compconv(ich3,qch3,xh2);
```

The waveforms of `ich4` and `qch4` are shown in Figure 3.12(d) and (e); the phase is shown in Figure 3.12(f). As mentioned in Section 3.2.3, if we use `conv.m`, we must recover the time delay due to convolution. In one time, `conv.m` causes a delay half the length of the filter taps. In this case, the length of the taps is `(irfn*IPOINT)/2`. Because we used a root Nyquist filter, `conv.m` is used twice. The delay time is thus `irfn*IPOINT` samples, and we can define the start point as `syncpoint`.

```
syncpoint=irfn*IPOINT+1;
```

Then, `ich4` and `qch4` are resampled at the rate of `IPOINT` and converted to `ich5` and `qch5`.

```
ich5 = ich4(sampl:IPOINT:length(ich4));
qch5 = qch4(sampl:IPOINT:length(ich4));
```

Then, the resampled data are fed into demodulation function.

```
[demodata]=qpskdemod(ich5,qch5,1,nd,ml);
```

Next, we investigate the number of errors. The procedure was described in Section 2.3. In this simulation, the transmission data is `data` and the received data is `data7`.

```
noe2=sum(abs(data1-demodata);
nod2=length(data1);
noe=noe+noe2;
nod=nod+nod2;
```

Finally, the BER is given by

```
>> ber=noe/nod;
```

As shown in Figure 3.13, the simulation results are quite similar to the theoretical values. For the simulation in the Rayleigh fading environment, you can use Program 3.6, which inserted the fading parameters and simulator to Program 3.5.

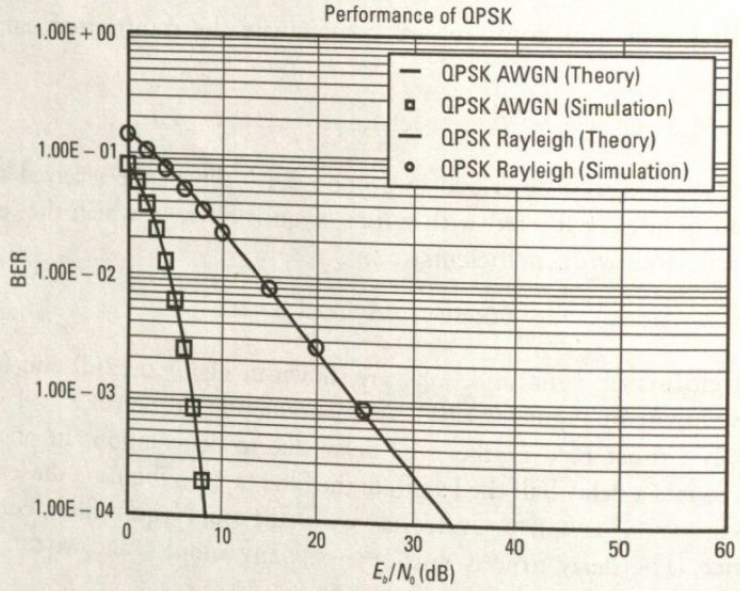

Figure 3.13 Theoretical and simulated BER performance of QPSK scheme under AWGN and one-path flat Rayleigh fading environments.

3.4 OQPSK

3.4.1 Basic Configuration of OQPSK

In the QPSK transmission scheme, the data of Ich and Qch (d_I, d_Q) has four states: A(1, 1), B(1, −1), C(−1, −1), and D(−1, +1). Because it can alternate between all candidates, the phase transmission is that shown in Figure 3.12(f). As shown in Figure 3.12(f), the phase transition sometimes passes the origin, meaning that the phase changes 180°, and the spectrum will be spread in comparison with that of other phase transitions. One way to reduce the number of origin crossings is to use OQPSK.

The configuration of OQPSK is almost the same as those of QPSK. The only different configuration is that for the mapping circuit. As shown in Figure 3.14, the data signal of the Qch signal is half a symbol delayed from that of the Ich signal. The configurations for the transmitter and receiver are the same as follows [1–3]. Therefore, the theoretical BER is the same as that for QPSK; it is shown in (3.27) and (3.28).

We can represent the OQPSK signal as those for [1–3]

$$s(t) = \frac{1}{\sqrt{2}} d_I(t)\cos(2\pi f_c t) + \frac{1}{\sqrt{2}} d_Q(t - T_b/2)\sin\left(2\pi f_c (t - T_b/2)\right) \tag{3.29}$$

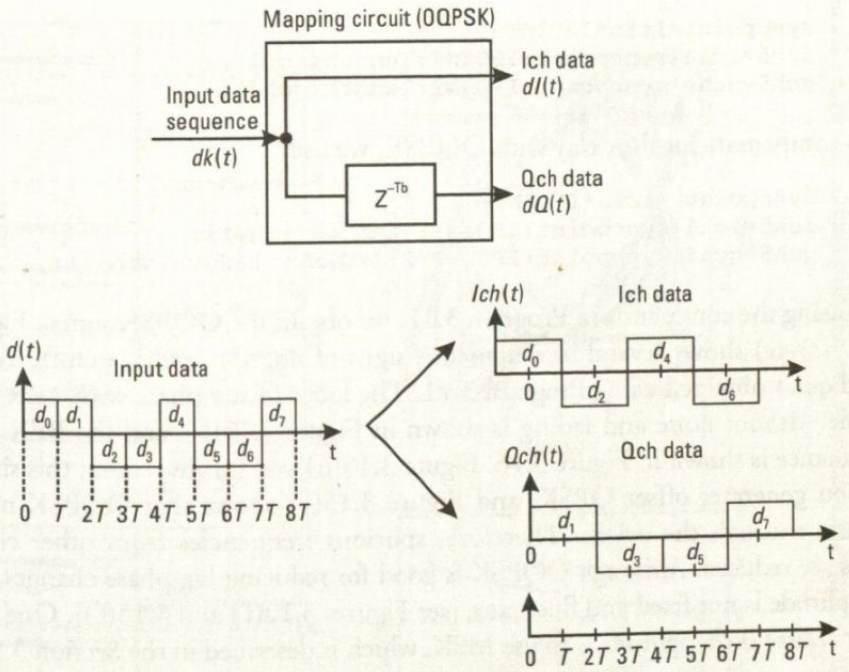

Figure 3.14 Mapping-circuit function for OQPSK.

3.4.2 Computer Simulation

The model we used to simulate the BER performance of OQPSK is based on that depicted in Figure 3.11 and shown as Program 3.11. As mentioned, the only difference is in the mapping circuit. The mapping-circuit commands for QPSK are listed as follows:

```
[ich, qch]=qpskmod(data1,1,nd,ml);
[ich1, qch1]=compoversamp(ich,qch,length(ich),IPOINT);
[ich2,qch2]=compconv(ich1,qch1,xh);
```

Those for OQPSK are listed as follows:

```
[ich,qch]=qpskmod(data1,1,nd,ml);
[ich1,qch1]=compoversamp(ich,qch,
  length(ich),IPOINT);
ich21=[ich1 zeros(1,IPOINT/2)];
qch21=[zeros(1,IPOINT/2) qch1];
[ich2, qch2]=compconv(ich21,qch21,xh);
```

As shown above, `qch21` delays `IPOINT/2` from `ich21`. The delay must be adjusted for after pulse-shaping at the receiver to demodulate the received signal. At the point for QPSK, we use