Chapter 6

Lecture Note Developing on Simulation of DQPSK System in the AWGN Channel

Author: Tri Budi Santoso

Supervisors: Prof. Hiroshi SUZUKI

Prof. Kazuhiko FUKAWA

Book Reference:

1. Digital Communication, John G. Proakis.
2. Digital Communication by Satellite.

Topic: Differential QPSK Transmission System

6.1. Session 1: Review of Diff QPSK System

Differential QPSK or DQPSK is similar with the QPSK system, but in the DQPSK the initial phase of modulated signal is effected by the initial phase of the previous modulated signal. The QPSK phase shift is based on assumption that the previous modulated signal has the initial phase 0. In the DQPSK the initial phase of transmitted signal is the addition result from the phase shift with the initial phase of previous modulated signal.

6.1.1. DQPSK Signaling

As like in the QPSK that every sequential of two pair bits of information will modulate the phase of carrier signal, and determine the output initial phase of the modulated signal. In the QPSK the rule of transmission are:

Pair bits input:	phase shift:
00	$\pi/4$ radiant
01	$3\pi/4$ radiant
11	$5\pi/4$ radiant
10	$7\pi/4$ radiant

But for the DQPSK we can't determine as like this way. First step is determines the rule of phase shift as a function of pair bit input. The second step is addition the phase shift with the initial phase of previous signal. The result of its addition is initial phase of the modulated signal at present time, or phase output of the transmitter. It is easier described by the following example.

Example:

The sequential inputs for DQPSK system is: 00, 01, 11, 10, 10, and 10. Determine the phase output of this system.

- The first step is determines the rule of phase shift.

Pair bits input:	**phase shift:**
00	0 radiant
01	$\pi/2$ radiant
11	$3\pi/2$ radiant
10	$-\pi/2$ radiant

- The second step is determines the sequences phase shift. By using the assumption for initial condition, the phase of the previous modulated signal is $\pi/4$ radiant. Using the sequential bit inputs and the rule of phase shift above we will get the initial phase value of present modulated signal.

pair bits input:	**phase shift:**
00	0
01	$\pi/2$
11	π
10	$-\pi/2$
10	$-\pi/2$
10	$-\pi/2$

The relation between bit inputs and phase transition from above example can be described in Figure (6.1) as follows.

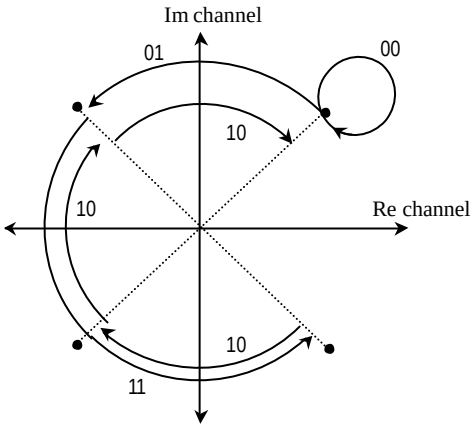


Figure (6.1) Phase Shift of DQPSK

The signal output from DQPSK transmitter is similar with the signal output from QPSK transmitter:

$$s_i(t) = \sqrt{Es/Ts}\cos[2pf_c t + f_I] + \sqrt{Es/Ts}\sin[2pf_c t + f_Q] \quad (6\text{-}1)$$

In this case parameter ϕ_i represent initial phase of the output signal from transmitter. By normalize the parameter Es/Ts = 1 of signal amplitude, the equation (6-1) will be:

$$s_i(t) = \cos[2pf_c t + f_I] + \sin[2pf_c t + f_Q] \quad (6\text{-}2)$$

6.1.2. Receiver of DQPSK

The receiver of DQPSK is similar with the QPSK system, but the output from LPF is differential decoded to get the information content. Instead, the phase of received signal in any given interval time is compared with the phase of previous signal. The demodulation and detection of DQPSK is illustrated by Figure (6.2) as follows.

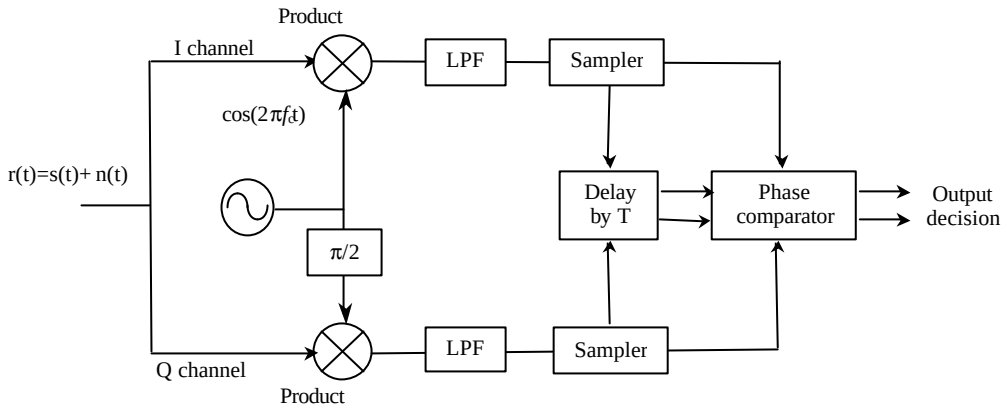


Figure (6.2) The DQPSK Receiver

The receiving process at the demodulator of DQPSK is similar with at the QPSK demodulator, but after LPF part the base band signal is sampled compared with the previous sampled signal. In Phase comparator part, the phase of present signal is compare with the phase of previous signal. The comparing process is done by detect the different value of phase between the present and the previous. Its different value is indicates the information content. Based on the rule of phase shift at the transmitter part, we will get the information content of the received signal.

Using the signal has transmitted from transmitter part; we will tray to recover the received signal by using DQPSK receiver. Our assumption that the channel is noise free, no ISI and demodulator and LPF worked perfectly.

The differential decoder will work as follows.

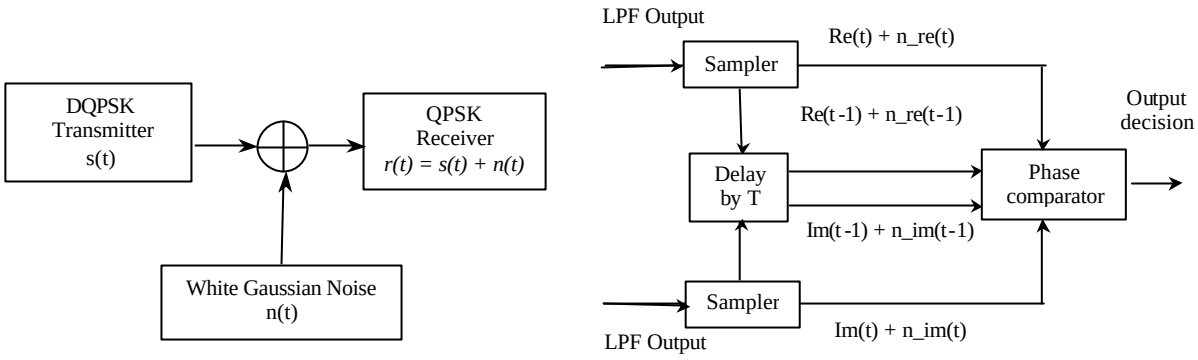
phase shift:	**pair bits output:**
0	00
$\pi/2$	01
π	11
$-\pi/2$	10
$-\pi/2$	10
$-\pi/2$	10

Same with the transmitter part, for initial condition we must set the initial phase is $\pi/4$, or the information content is 00. After we get the phase different between sequence signal, the nest process is information recovered based on the rule of phase shift at the transmitter part.
In this case we never worry if carrier recovery at he receiver part is not work perfectly and the initial phase of receiver part is not same with the phase of transmitter part. Because the information is represented by the differential value of phase between the present and the previous received signal.

Check this result and compare with the QPSK system. We looked that the result of DQPSK receiver part is same with the receiver part of QPSK, but the signal transmission is different.

6.1.3. DQPSK Transmission System

The transmission of DQPSK system through AWGN channel is similar with the DPSK system in the same channel. But in this case, the signal transmitted is content of real and imaginer component. Therefore in the transmission we must consider the real and imaginer components of the AWGN channel. The simplification form for this transmission can be described as Figure (6.3) follows.



(b) General description (b) The signal received

Figure (6.3) DQPSK transmission through AWGN channel

The transmitted signal is s(t) = Re(t) + Im(t), and from AWGN channel the component of additive noise is n(t) = n_re(t) + n_im(t). In this case Re is represent a real part of transmitted signal that resulted from product modulation between of signal with In-phase channel information. The Im is represents an imaginary part of transmitted signal that resulted from product modulation of carrier signal with quadrature channel information.

At the receiver of DQPSK the comparator is process the signal from present time:

$$r(t) = s(t) + n(t)$$
$$= [Re(t) + n\_re(t)] + [Im(t) + n\_im(t)] \quad (6\text{-}3)$$

and from previous time:

$$r(t-1) = s(t-1) + n(t-1)$$
$$= [Re(t-1) + n\_re(t-1)] + [Im(t-1) + n\_im(t-1)] \quad (6\text{-}4)$$

In some book, the received signal is expressed different style. Re(t) is represent by $\sqrt{Es}\cos(\theta_t - \phi)$ and Im(t) is represent by $j\sqrt{Es}\sin(\theta_t - \phi)$ respectively. The received signal will represent by $r(t) = \sqrt{Es}\exp[j(\theta_t - \phi)] + n(t)$, and the previous signal will be $r(t-1) = \sqrt{Es}\exp[j(\theta_{t-1} - \phi)] + n(t-1)$.

The decision variable for the phase detector is the phase different between these two received signals. By using complex conjugate operation, we will get the received signal as:

$$r(t)r^*(t-1) = Es\exp[j(\theta_t - \theta_{t-1})] + \sqrt{Es}\exp[j(\theta_t - \phi)]n^*(t-1) + \sqrt{Es}\exp[j(\theta_{t-1} - \phi)]n(t) + n(t)n^*(t-1) \quad \ldots.(6\text{-}5)$$

In the condition where is noise is abstain, we will get the received signal is

$$r(t)\, r^*(t-1) = Es\exp[j(\theta_t - \theta_{t-1})]$$

Thus we get the phase different as $(\theta_t - \theta_{t-1})$. This value is independent from the carrier phase value. The exponential factor in the equation (1-5) can be absorbed into the gaussian noise component. And the equation will be:

$$r(t)\, r^*(t-1) = Es + \sqrt{Es}\,[n(t) + n^*(t-1)] + n(t)\, n^*(t-1) \quad (6\text{-}6)$$

The value of n(t) n*(t-1) is small compare than the other component, and in calculation usually this component is negligible. Normalization by using parameter $\sqrt{Es}$ we will get:

$$r(t)\, r^*(t-1) = \sqrt{Es} + [n(t) + n^*(t-1)] \quad (6\text{-}7)$$

We will get the new set of decision metric as:

$$\text{Re} = \sqrt{Es} + \text{Re}\,[n(t) + n^*(t-1)]$$

$$\text{Im} = \sqrt{Es} + \text{Im}\,[n(t) + n^*(t-1)] \qquad (6\text{-}8)$$

The value of x and y are uncorrelated Gaussian random variables with identical variance σ_n^2 = No. The phase can be known as $\tan^{-1}(y/x)$.

The differential detection MPSK system in the AWGN channel will have a symbol error probability performance:

$$P_s \approx erfc[W] + \frac{2W\exp\left(-W^2\right)}{\sqrt{p}\left(8Es/No + 1\right)} \qquad where\ \ W = \sqrt{2Es/No}\sin\left(p/2M\right) \qquad (6\text{-}9)$$

For larger Es/No, it can be approximated as:

$$P_s \approx erfc\left[\sqrt{\frac{Es}{No}}\sin\left(\frac{p}{\sqrt{2}M}\right)\right] \text{ and the bit error probability is } P_b = \frac{P_s}{\log_2(M)} \qquad (6\text{-}10)$$

For DQPSK case, M = 4 and Es = $\log_2(4)$Eb or Es = 2Eb. The value of symbol error probability is:

$$\begin{aligned} Ps &= erfc\left[\sqrt{\frac{\log_2(2)Eb}{No}}\sin\left(\frac{p}{\sqrt{2}\log_2(4)}\right)\right] \\ &= erfc\left[\sqrt{\frac{2Eb}{No}}\sin\left(\frac{p}{2\sqrt{2}}\right)\right] \end{aligned} \qquad (6\text{-}11)$$

And the bit error probability is

$$Ps = \frac{1}{2}erfc\left[\sqrt{\frac{2Eb}{No}}\sin\left(\frac{p}{2\sqrt{2}}\right)\right] \qquad (6\text{-}12)$$

We can compare the performance of bit error probability between QPSK and DQPSK as follows:

$$\frac{P_b(QPSK)}{P_b(DQPSK)} = \frac{\frac{1}{2}erfc\left[\sqrt{2\,Eb/No}\sin\left(\frac{p}{2}\right)\right]}{\frac{1}{2}erfc\left[\sqrt{2\,Eb/No}\sin\left(\frac{p}{2\sqrt{2}}\right)\right]} \qquad (6\text{-}13)$$

The bit error probability performance as a function of Eb/No for DQPSK and QPSK is described in the Figure (6.4).

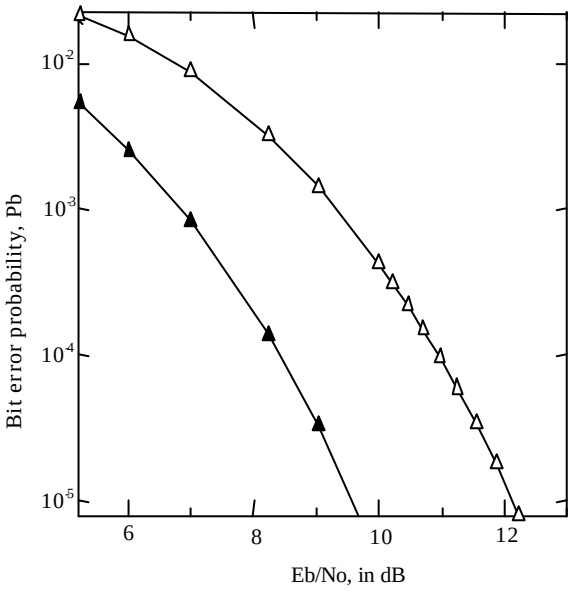


Figure (6.4) Performance comparison of DQPSK and QPSK

From this figure we know that to get the same value of bit error probability, the DQPSK system need Eb/No 2.3 dB larger that QPSK system.

6.1.4. Carrier Recovery of DQPSK System

Carrier-phase recovery by using Costas Loop for QPSK is a combination from two parallel carrier-phase recoveries, which we have applied it for BPSK system. General descriptions of Costas loop carrier recovery for QPSK in the base band term as the Figure (6.5). In this case we must redefined the carrier frequency of transmitted signal as f_{cT} and its initial phase as ϕ_i. And at receiver part we redefine the local carrier frequency generated as f_{cR} and its initial phase estimate as ϕ_{est}.

In DQPSK system the transmission signal is content of Real (Re) and Imaginary (Im) part:

$$s(t) = A_{Re}\cos(2\pi f_{cT}t + \phi_i) + A_{Im}\cos(2\pi f_{cT}t + \phi_i).$$

In this case the average value of A_{Re} and A_{Im} is 1. The synchronization process by using Costas Loop is as Figure (*.*). In synchronization process for DQPSK each channel is multiply by $\cos(2\pi f_{cR}t + \phi_{est})$ and $\sin(2\pi f_{cR}t + \phi_{est})$

- The output of product 1 is:

$$y_{p1}(t) = \cos(2\pi f_{cT}t + \phi_i)\cos(2\pi f_{cR}t + \phi_{est}) + n(t)_{Re}\cos(2\pi f_{cR}t + \phi_{est})$$
$$= (1/2)\cos\{2\pi(f_{cT} + f_{cR})t + (\phi_i + \phi_{est})\} + (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\}$$
$$+ n_{Re}(t)\cos(2\pi f_{cR}t + \phi_{est})$$

The low pass filter process will give the output as:

$$y_{LPF1}(t) = (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} = (1/2)\cos(2\pi \Delta f\, t + \Delta\phi_t)$$

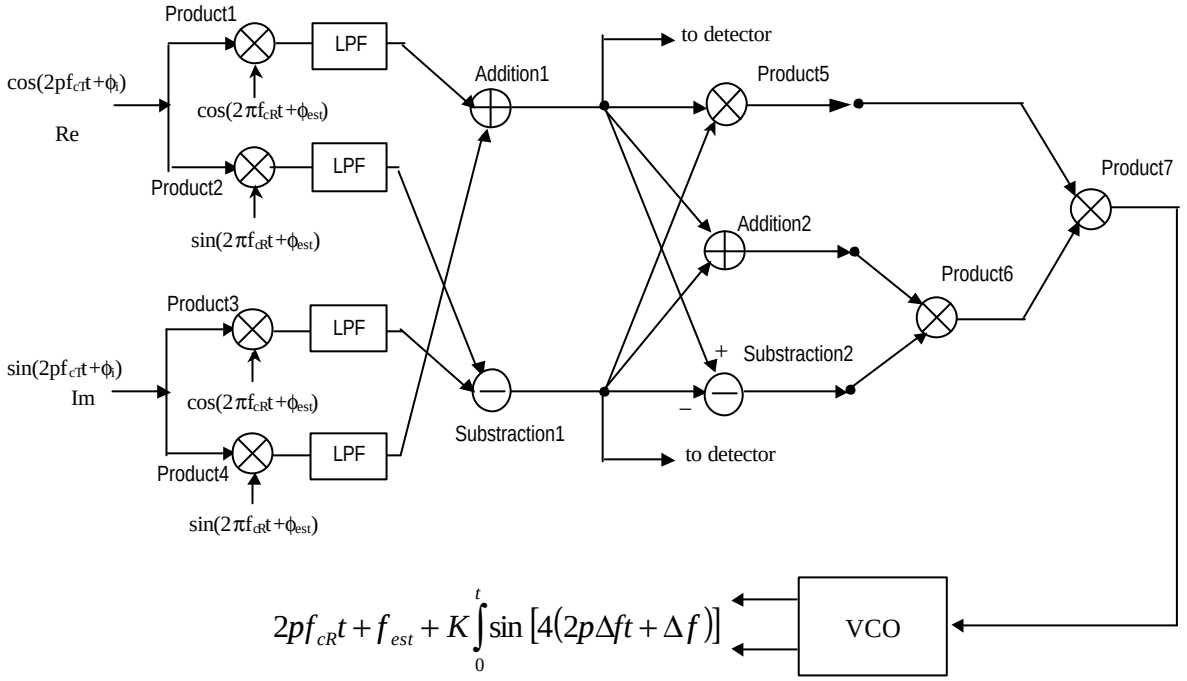


Figure (6.5) Simplification of carrier recovery for QPSK system

- The output of product 2 is:

$$y_{p1}(t) = \cos(2\pi f_{cT}t + \phi_i)(\sin(2\pi f_{cR}t + \phi_{est})) + n(t)_{Re}\sin(2\pi f_{cR}t + \phi_{est})$$
$$= (1/2)\sin\{2\pi(f_{cT} + f_{cR})t + (\phi_i + \phi_{est})\} - (1/2)\sin\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\}$$
$$+ n_{Re}(t)\sin(2\pi f_{cR}t + \phi_{est})$$

The low pass filter process will give the output as:

$$y_{LPF2}(t) = (-1/2)\sin\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} = (-1/2)\sin(2\pi \Delta f\, t + \Delta\phi_t)$$

- The output of product 3 is:

$$y_{p3}(t) = \sin(2\pi f_{cT}t + \phi_i)\cos(2\pi f_{cR}t + \phi_{est}) + n(t)_{Im}\cos(2\pi f_{cR}t + \phi_{est})$$
$$= (-1/2)\sin\{2\pi(f_{cT} + f_{cR})t + (\phi_i + \phi_{est})\} + (1/2)\sin\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\}$$
$$+ n_{Im}(t)\cos(2\pi f_{cR}t + \phi_{est})$$

The low pass filter process will give the output as:

$y_{LPF3}(t) = (1/2)\sin\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} = (1/2)\sin(2\pi \Delta f\, t + \Delta\phi_t)$

- The output of product 4 is:

$y_{p4}(t) = \sin(2\pi f_{cT}t + \phi_i)\sin(2\pi f_{cR}t + \phi_{est}) + n(t)_{Im}\sin(2\pi f_{cR}t + \phi_{est})$

$= -(1/2)\cos\{2\pi(f_{cT} + f_{cR})t + (\phi_i + \phi_{est})\} + (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\}$

$+ n_{Im}(t)\sin(2\pi f_{cR}t + \phi_{est})$

The low pass filter process will give the output as:

$y_{LPF4}(t) = (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} = (1/2)\cos(2\pi \Delta f\, t + \Delta\phi_t)$

The next steps are

- The output of addition1 is:

$add1(t) = (1/2)\cos(2\pi \Delta f\, t + \Delta\phi_t) + (1/2)\cos(2\pi \Delta f\, t + \Delta\phi_t) = \cos(2\pi \Delta f\, t + \Delta\phi_t)$

- The output of substraction1 is:

$subst1(t) = (1/2)\sin(2\pi \Delta f\, t + \Delta\phi_t) - (-1/2)\sin(2\pi \Delta f\, t + \Delta\phi_t) = \sin(2\pi \Delta f\, t + \Delta\phi_t)$

- The output of product5 is:

$y_{p5}(t) = \cos(2\pi \Delta f\, t + \Delta\phi_t)\sin(2\pi \Delta f\, t + \Delta\phi_t) = (1/2)\sin\{2(2\pi \Delta f\, t + \Delta\phi_t)\}$

- The output of addition2 is:

$add2(t) = \cos(2\pi \Delta f\, t + \Delta\phi_t) + \sin(2\pi \Delta f\, t + \Delta\phi_t)$

- The output of subtraction2 is:

$Sub2 = \cos(2\pi \Delta f\, t + \Delta\phi_t) - \sin(2\pi \Delta f\, t + \Delta\phi_t)$

- The output of product6 is:

$y_{p6}(t) = [\cos(2\pi \Delta f\, t + \Delta\phi_t) + \sin(2\pi \Delta f\, t + \Delta\phi_t)][\cos(2\pi \Delta f\, t + \Delta\phi_t) - \sin(2\pi \Delta f\, t + \Delta\phi_t)]$

$= \cos^2(2\pi \Delta f\, t + \Delta\phi_t) - \sin^2(2\pi \Delta f\, t + \Delta\phi_t)$

$= \cos\{2(2\pi \Delta f\, t + \Delta\phi_t)\}$

- The output of product7 or the error loop signal is:

$e(t) = (1/2)\sin\{2(2\pi \Delta f\, t + \Delta\phi_t)\}\cos\{2(2\pi \Delta f\, t + \Delta\phi_t)\}$

$=(1/4)[2\sin\{2(2\pi \Delta f\, t + \Delta\phi_t)\}\cos\{2(2\pi \Delta f\, t + \Delta\phi_t)\}]$

$=(1/4)\sin\{4(2\pi \Delta f\, t + \Delta\phi_t)\}$ (6-14)

It is known as error signal into the loop filter of QPSK carrier-phase recovery, and used to drive the VCO. The new output phase of VCO is given as

$$2pf_{cR}t + f_{est} + \int_0^t K_c e(t)dt$$

or (6-15)

$$2pf_{cR}t + f_{est} + K\int_0^t \sin\left(4(2p\Delta ft + \Delta f)\right)dt \text{ where is K = Kc/4.}$$

The loop phase error and its time derivative is

$$q_e(t) = \left(2pf_{cT}t + f_i\right) - \left(2pf_{cR}t + f_{est} + K\int_0^t \sin\left(4(2p\Delta ft + \Delta f)\right)dt\right)$$

$$\frac{d\left(q_e(t)\right)}{dt} = \left(2pf_{cT} - 2pf_{cR}\right) - K\sin\left(4(2p\Delta ft + \Delta f)\right)$$

If $f_{cT} = f_{cR}$ and letting K =1, the time derivative of the loop phase error is given by

$$\frac{d\left(q_e(t)\right)}{dt} = -\sin\left(4\Delta f\right) \quad (6\text{-}16)$$

The null value will be happen at $0, \pi/2, \pi, 3\pi/2$ radiant, and it's known as four phase ambiguity of phase-carrier recovery at DQPSK system. But in this case it doesn't problem, because the information is content in the phase different between the phase of the present and the phase of previous signal received.

6.2. Session 2: Question and Answer

Question 1

What is advantage if we use DQPSK system compare with QPSK system?

Answer:

In the QPSK by using coherent detection the receiver part must understand the frequency an phase of transmitted signal exactly. But in the real condition this process is very difficult, and carrier recovery process never get this ideal condition. Same with the DPSK system in the DQPSK the important think is different value of present and previous phase of received signal. By using this technique the receiver part doesn't need recover the phase of received signal exactly as like in the QPSK system. Or it can be said that the DQPSK system is easier to build than the QPSK system.

Question 2

How the DQPSK performance compare with QPSK performance in the AWGN channel condition for it SNR value?

Answer:

By using the equation (6-13) we can see that DQPSK system need SNR value 2.3 dB larger than QPSK system to get the same value of bit error probability. Or the other hand we can say that in the same SNR value the DQPSK performance is poor than QPSK system. More detail for it we can see the performance comparison of DQPSK and QPSK in the Figure (6.4).

6.3. Session 3: Simulation of Differential QPSK System

Here we tray to build the Differential QPSK (DQPSK) simulation system in the base band system. In this simulation, we use AWGN channel as transmission medium. The general description is in the Figure (6.6).

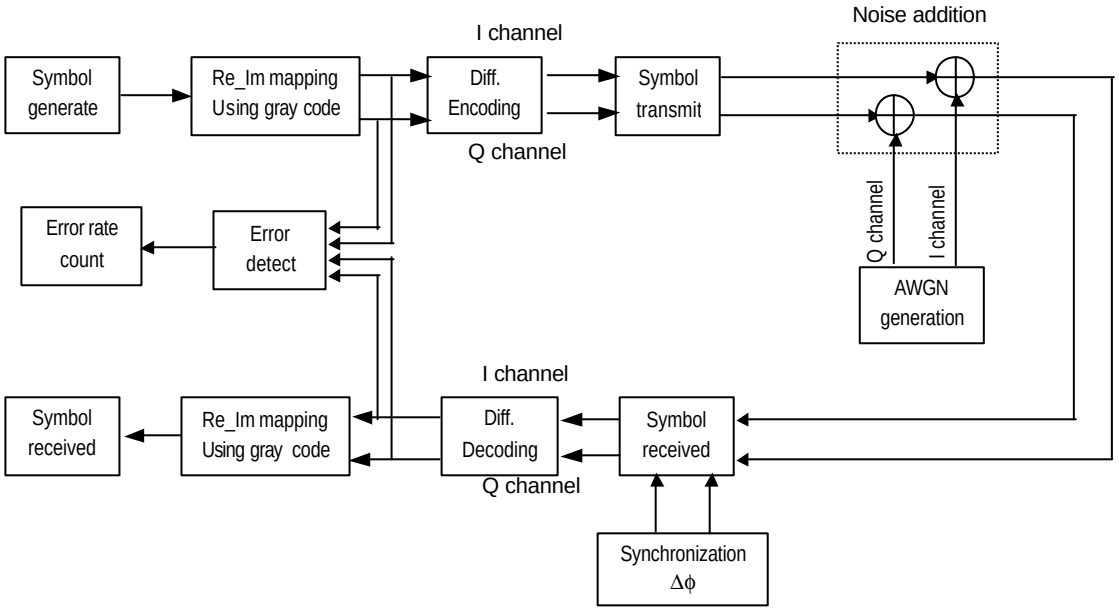


Figure (6.6). The block diagram of DQPSK Base band simulation system

6.3.1. Algorithm

Based on Figure (6.6) above, the simulation program for DQPSK transmission system is involving step:

- **Info Generate**

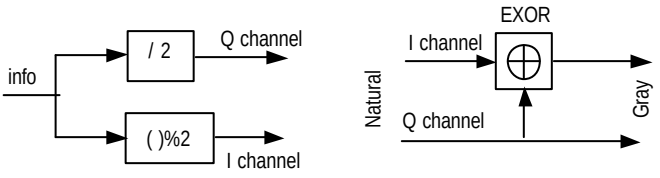
This function represent the symbol generate part in the Figure (6.6). By using standard uniform random generate for Microsoft C, we able to generate the sequence of integer value randomly uniform. Setting max value is 4, and it will generate the sequence value 0, 1, 2, or 3 with same probability.

We can use the statement *info = rand()%4;* has the meaning to generate an integer value from 0 to 3 uniform randomly.

- **Re channel and Im channel Mapping**

Different with QPSK system, in DQPSK system this step is pure I channel and Q channel map from integer value to binary value and codes the result by using gray code. Integer

information generated will map to two bit pair. Based on the Figure (6.7a) we will understand the following logic. If the integer information value is 0 the two bits pair generate are 00. If the integer information value is 1 the two bits pair are 01. If the integer information is 2 the two bits pair are 01. And if the integer information is 3 the two bits pair is 11. Then we set the LSB value as I channel info and the MSB value as Q channel info. The second step is Gray Code. I channel and Q channel info has natural bit value, by using the logic as like in the Figure (6. 7b) we able to make a gray code logic. If the value of I and Q is 10, the Q channel value will change to 1. If the value of I and Q is 11, it will change to 10. The other value is not change.



a) I and Q mapping b) Gray Code

Figure (6.7) I and Q mapping and gray code

♦ **Differential Encoding**

In this process data differential encoded before transmitted. To understand this process is easier if we see the Figure (6.8) and Table 6.1.

In this figure we saw that the gray coded data generate must be converted to natural before differential encoded. By using the logic EXOR the gray coded data will change to natural value. The next process is differential encoded by using the delay process.

Similar with the differential coding process in the DPSK, in this process the present output is affected by the present data input and the previous output. By using modulo–4 logic operation the process is done. The A_1 value is a_1^n*2 and the value of A_2 is a_2^n*1. The T_1 value is B_1*2 and the value of T_2 is B_2*1 from the previous time. And the value of B_1 and B_2 in the present time is Mod-4 of $[(A_1+A_2) + (T_1+T_2)]$. If the total value is $\leq$ 3 the output is same with the normal output from addition process. The example is the total value of addition is 3, then the output is $B_1 = 1$ and $B_2 = 1$. If the total value is $\geq$ 4, the output is original output – 4.The example for the addition result is 4, the output will

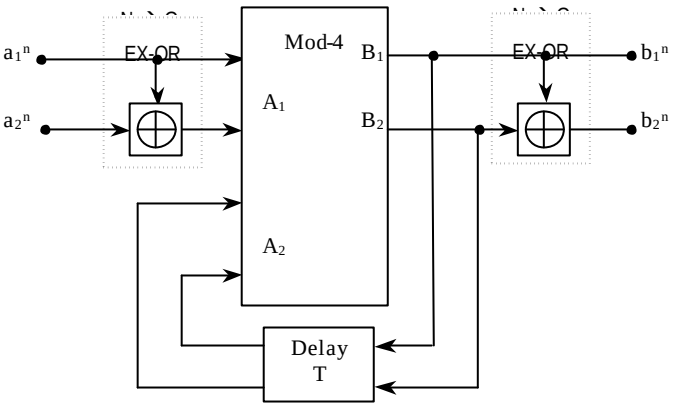


Figure (6.8) Differential Encoder of DQPSK

be 4 - 4 = 0. And the result is $B_1 = 0$ and $B_2 = 0$. If the addition value is 5, the output will be 1 and the result is $B_1 = 0$ and $B_2 = 1$.

Table 6.1. Truth table of differential Encoding for DQPSK

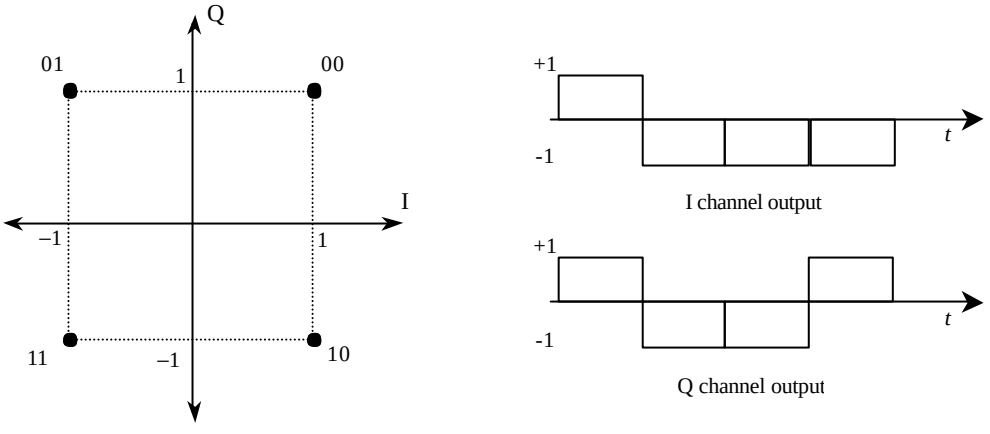
Step	info	Gray		G→ N		Mod-4		N→ G	
		a_1	a_2	A_1	A_2	B_1	B_2	b_1	B_2
1	0	0	0	0	0	0	0	0	0
2	2	1	1	1	0	1	0	1	1
3	0	0	0	0	0	1	0	1	1
4	3	1	0	1	1	0	1	0	1
5	1	0	1	0	1	1	0	1	1
6	2	1	1	1	0	0	0	0	0
7	1	0	1	0	1	0	1	0	1
8	2	1	1	1	0	1	0	1	0

After differential encoding process done, the binary data is graying encoded by using EXOR logic. Why we must convert from natural to gray and gray to natural in this process? It done to complete the requirement the gray coded data is like in the real system of DQPSK transmission system.

♦ **Transmitting Process**

I channel value and Q channel value has "1 or 0" info and have integer variable. By using this part it will change as double variable and have the value –1.00 and +1.00. The bit 1 will be represent by using symbol –1.00 and the bit 0 by +1.00. This is represents an antipodal signals.
From the above example we get the output from info generate are: 0,2,0,3, ...
And the bit pair from differential encoder are: 00, 11,11,01,….

Based on the figure (6.9a) as constellation diagram, we will get the symbol output from I channel of the transmitter: +1, -1, -1, -1. And the output from Q channel of transmitter is +1, -1, -1, and +1. And the antipodal signal from the transmitter is like Figure (6.9b). In program we use Re to represent I, and Im to represent Q channel.



a) Constellation diagram b) Transmitter output

Figure (6.9) Transmitter output of DQPSK

♦ AWGN Channel

The AWGN Channel has two parts, the first is AWGN channel generator and the second part is noise addition process. It is same with AWGN channel for QPSK.

- ***AWGN Channel Generator***

To generate AWGN channel we must do it by using two steps: uniform random generate and shifting from uniform to Gaussian distribution.

First step is generates two sequences of uniform random generator, for this purpose we can set variable x1 and x2. Both of them have double data type. These two variables have a value between 0 and 1. To do it we can generate the value x1 and x2 from 0 until default value of RAND_MAX or 32767.00. Then we divide x1 and x2 by RAND_MAX.

Second step do by using Box-Muller method. Two variable x1 and x2, which have the uniform distribution, shift by using formulation:

$$re = (s^2\ lnx1)^{1/2}\ cos(2*p*x2)\ \ ;real\ part$$

$$im = (s^2\ lnx1)^{1/2}\ sin(2*p*x2)\ \ ;imaginary\ part.$$

♦ Synchronization

In this function we must make sure that receiver part understand exactly the phase of the received signal from transmitter. Unfortunately in the real system, local carrier frequency and its initial phase does not same with the transmitter part. The Costas loop as carrier recovery is use to correct the phase different.

As we know that in the synchronization process by using Costas loop for BPSK system there two phase ambiguity among 0, $\pi/2$, π, and $3\pi/2$ radiant. But it doesn't matter because the information is content in the different value between the phase of present signal and phase of previous signal.

- ***Noise Addition***

The noise generated from the AWGN function are real and Imaginary part, and independent each one another. In this function we add Re channel with real noise and the Im channel with imagine noise. In some books Re channel from QPSK transmitter is called as real channel, and Im channel is called as imaginary channel. AWGN will give a degradation

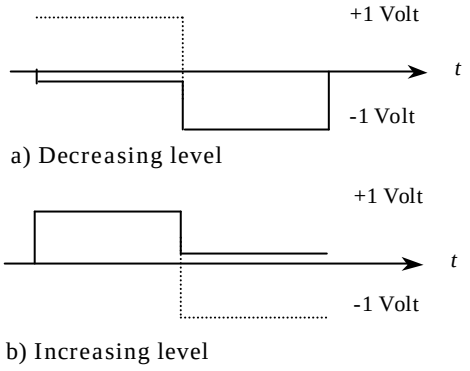


Figure (6.10) AWGN effect at the level of antipodal signal transmission

effect of amplitude level of signal transmission. If the value of AWGN is negative it will make the level of signal transmission is down, other wise if the AWGN value is positive is will make the level of signal transmission will up.

The effect of AWGN will change the level of transmitted signal as like in the Figure (6.10).

♦ **Detection**

This function represents the *Symbol Received* part. Based on the zero level as a threshold, the decision is made:

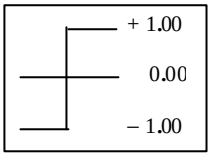


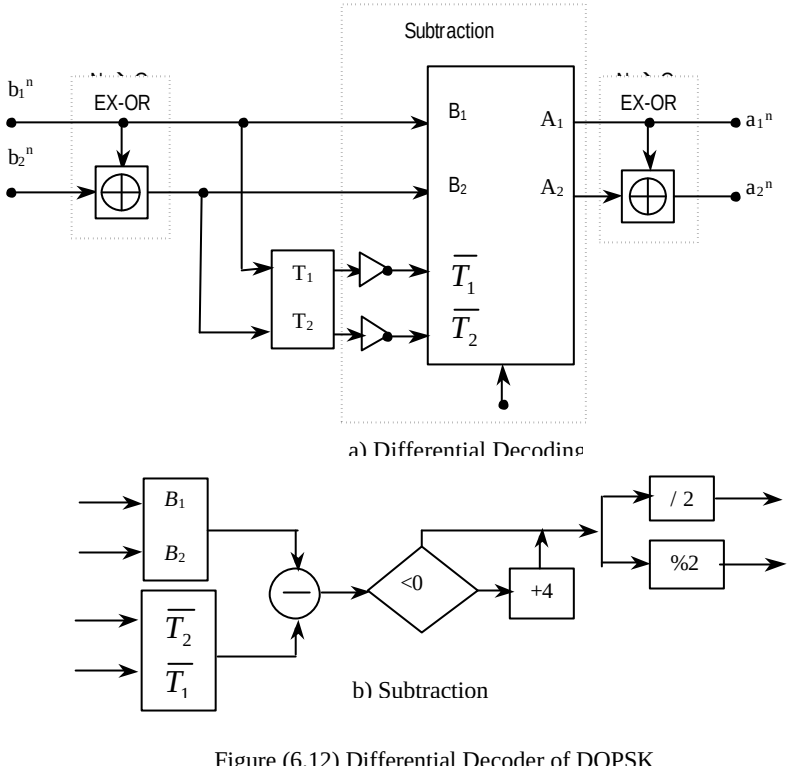
Figure (6.11) Decision symbol for received signal

if signal level < 0.0 the decision for symbol received is –1.0
Other wise
if signal level ≥ 0.0 the decision for symbol received is 1.0.

This rule for both of Re channel and Im channel signal received, and work independently.

♦ **Differential Decoding**

After the received symbol convert to binary values the next step is differential decoding process. The binary data input must be converted from gray coded to natural by using EXOR logic operation. The next process is subtraction the present input by the previous input. To get the previous input we use delay process, T. More detail this process can see in the Figure (6.12), in this figure the parameter B_1 and B_2 are the input from the present time. The parameter $\overline{T_1}$ and $\overline{T_2}$ are the input from the previous time.



a) Differential Decoding

b) Subtraction

Figure (6.12) Differential Decoder of DQPSK

After convert to decimal value, its parameter use as input of subtraction operation. If the result is < 0, the result is added by 4, other wise if the result is ≥ 0 it is not added. Then it value

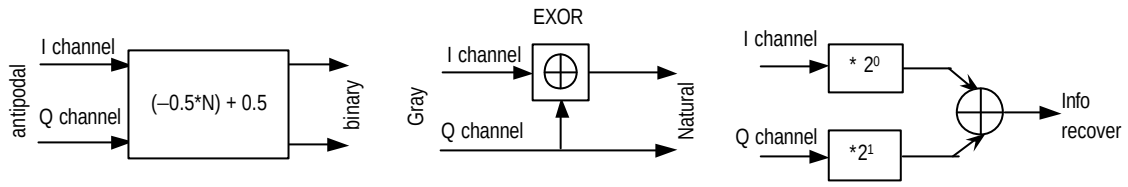
converted to binary code and known as parameter A_1 for 2^2 and A_2 for 2^0.

The next process is converting from natural to gray code. This operation is represent a subtraction of binary data, or in this case is operation of differential decoding process at the receiver part. We can use the function *conjugate(R_Tx_n,R_Rx0,I_Tx_n,I_Rx0,&R_Rx,&I_Rx).*

♦ **Rx Demap**

This function is same with RX Demap for QPSK system; it consists of two parts of block diagram in the Figure (6.13).

- *Gray de-codes*. First step in this block is converts from antipodal signal to binary signal. By using the mathematical expression opposite with we has used in the symbol transmission block. More easily we can see at the Figure (6.13a). Second step is converts from gray form to natural binary from. By using the XOR logic, for Re channel input we get the natural binary result as like Figure (6.13b).
- *I_Q demap*. The purpose of this part is to get the symbol value by convert the two pair bit (binary) decimal value. By multiply the MSB with 2 and LSB with 2^0 or 1, and continue with addition of both results, we will get the decimal information value.



a) Antipodal to binary

b) Gray de-Code

c) I and Q de-mapping

Figure (6.13) Description of Rx De-mapping

♦ **Error Detect**

By using the output from detection function and compare channel by channel to the output of I_Q map we able detect what the error in the receiving signal happen. If the value of output from detection function is same with the output from Re_Im map, the error didn't happen, other wise if these value is not same, the error was happening. By using channel by channel error detection we gets bit error information of Re and Im channels, and combine the result of these result we will get the information of symbol error detection. If Re channel error or Im channel error happened, the symbol error detect indicate the value 1, if no error both of channel there is no error happened and the value of symbol error detect indicate the value 0.

♦ **Error Rate Calculation**

In this part we sum all of error detected that happened during transmission process. And the result from this summation subtracted by the total bit transmitted. The result is *bit error rate*. If we compare the summation of symbol error detected to total symbol transmitted, the result is *symbol error rate*. This is the real performance resulted by our transmission system.

♦ **Error Probability Calculation**

In this part we calculate the bit error probability of QPSK system as a function of *energy per bit to noise ratio*. The output is theoretical value of *error probability* performances QPSK system. From this function we get the information of bit error probability and symbol error probability.

6. 3.2. The Listing Program

To develop a program in order the user understand it easily we must set the symbol which represent the parameters in theory:

N = total symbol transmit.

info = info generate.

Dr = real part data generate

Di = imaginary part data generate

R_Tx = real part symbol generate

I_Tx = imaginary part symbol generate

pha_Tx = phase of transmitted signal

phase = error phase resulted by carrier recovery

R_AWGN = real part AWGN

I_AWGN = imaginary part AWGN

var = noise varians;

R_Rx = real part of symbol received

I_Rx = imaginary part of symbol received

pha_Rx = phase received

pha_comp = different phase resulted in the phase comparator process

R_Rec = real part of symbol recovery

I_Rec = imaginary part of symbol recovery

rec_Dr = real part of bit recvery

rec_Di = imaginary part of bit recvery

info_rec = information recovery

BER = bit error rate

SER = symbol error rate

SNR = signal to noise ratio

Ps_DQPSK = probability of symbol error of DQPSK system

Pb_DQPSK = probability of bit error of DQPSK system

The listing program is as follows.

```
//Program of Differential QPSK
//editing by Tri Budi Santoso
//Suzuki Laboratory,Mobile Communication Group
//Tokyo Institute of Technology, Japan

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define N 1000

int i,nn=N-1;
int info[N];
double Dr[N],Di[N];
double R_Tx[N],I_Tx[N],pha_Tx[N];
double phase[N],Ave_Pha_err,Pha_err,deg_error,pha_i;
double rmax=RAND_MAX+1.0,R_AWGN[N],I_AWGN[N],pi=acos(-1.0),var=0.1;
double R_Rx[N],I_Rx[N];
double pha_Rx[N];
double pha_comp[N];
double R_Rec[N],I_Rec[N];
int info_rec[N];
double rec_Dr[N],rec_Di[N];
double BER,SER,SNR;
double Ps_DQPSK,Pb_DQPSK;

void info_generate(int info[]);
void QPSK_mapping(double Dr[],double Di[]);
void phase_shift(double R_Tx[],double I_Tx[],double pha_Tx[]);
void costas(double phase[],double *pha_er);
void synchronization(double pha_i,double *deg_er);
void AWGN_channel(double R_AWGN[],double I_AWGN[]);
void noise_addition(double R_Rx[],double I_Rx[]);
void phase_received(double pha_Rx[]);
void phase_comparator(double pha_comp[]);
void phase_symbol(double R_Rec[],double I_Rec[]);
void symb_decision(int info_rec[],double rec_Dr[],double rec_Di[]);
void error_detect(double *ber,double *ser);
void sig_noise_ratio(double *snr);
void error_prob(double *ps,double *pb);

void main()
{
FILE *fdata;
```

```
char filename[]="DifQPSK_costas2_data.txt";
fdata=fopen(filename,"a");

srand(1);
info_generate(info);
QPSK_mapping(Dr,Di);
phase_shift(R_Tx,I_Tx,pha_Tx);
costas(phase,&Ave_Pha_err);
AWGN_channel(R_AWGN,I_AWGN);
noise_addition(R_Rx,I_Rx);
phase_received(pha_Rx);
phase_comparator(pha_comp);
phase_symbol(R_Rec,I_Rec);
symb_decision(info_rec,rec_Dr,rec_Di);
error_detect(&BER,&SER);
sig_noise_ratio(&SNR);
error_prob(&Ps_DQPSK,&Pb_DQPSK);
fprintf(fdata,"\n%4.2f %f %f",SNR,BER,Pb_DQPSK);
fclose(fdata);
}
//info generate uniform randomly
void info_generate(int info[])
{
   for(i=1;i<=nn;++i)
   {
   info[i]=(int)rand()%4;
   }
}

//mapping by using Gray code
void QPSK_mapping(double Dr[],double Di[])
{
   for(i=1;i<=nn;++i)
   {
      if(info[i]==0)
      {Dr[i]=1.0;Di[i]=1.0;/* 0 */}
      else if(info[i]==1)
      {Dr[i]=-1.0;Di[i]=1.0;/* 0.5*pi */}
      else if(info[i]==2)
      {Dr[i]=-1.0;Di[i]=-1.0;/* pi */}
      else if(info[i]==3)
      {Dr[i]=1.0;Di[i]=-1.0;/* 1.5*pi */}
      else{exit(0);}
//             printf("\ninfo[%d]:%d Dr[%d]:%4.2f \tDi[%d]:%4.2f",i,info[i],i,Dr[i],i,Di[i]);
   }
}

//phase shift process
void phase_shift(double R_Tx[],double I_Tx[],double pha_Tx[])
{
   double shift,PI_2=1.570796;
   double R_Tx0=1.0,I_Tx0=1.0;
   for (i=1;i<=nn;++i)
   {
  shift=(double)info[i]*PI_2;
      R_Tx[i]=R_Tx0*cos(shift) - I_Tx0*sin(shift);
      I_Tx[i]=R_Tx0*sin(shift) + I_Tx0*cos(shift);
```

```
        pha_Tx[i]=atan(I_Tx[i]/R_Tx[i]);

        if (R_Tx[i]>0.0 && I_Tx[i]>0.0)
      {pha_Tx[i]=pha_Tx[i]/pi*180;}
      else if (R_Tx[i]>0.0 && I_Tx[i]<0.0)
      {pha_Tx[i]=360.0 + pha_Tx[i]/pi*180;}
      else if (R_Tx[i]<0.0 && I_Tx[i]>0.0)
      {pha_Tx[i]=180.0 - pha_Tx[i]/pi*180;}
      else if (R_Tx[i]<0.0 && I_Tx[i]<0.0)
      {pha_Tx[i]=180.0 + pha_Tx[i]/pi*180;}
      else
    printf("");

        R_Tx0=R_Tx[i];
        I_Tx0=I_Tx[i];
//    printf("\nR_Tx[%d]:%4.2f \tI_Tx[%d]:%4.2f",i,R_Tx[i],i,I_Tx[i]);
      }
}


void costas(double phase[],double *pha_er)
{
 double pha_i;
 Pha_err=0.0;
 for(i=1;i<=nn;++i)
 {
  pha_i=pha_Tx[i];
  synchronization(pha_i,&deg_error);
  phase[i]=fabs(deg_error);
 // printf("\n phase[i]:%f ",phase[i]);
    Pha_err += fabs(phase[i]);
 }
 Ave_Pha_err=Pha_err/nn;
  *pha_er=Ave_Pha_err; //printf("\nAve_Pha_err:%f",Ave_Pha_err);
}

void synchronization(double pha_i,double *deg_er)
{
 int d=1,i=1,symbol_rate=16000,phase_uni;
 double rad_error,fc=800e+6,T,delta_fc,phase_estim;
 double VCO,K=1.0;

 T=1.0/(double)symbol_rate;
 delta_fc=1e-6 * fc *T;
 do
 {
  phase_uni=(int)rand()%360;
  phase_estim=(double)phase_uni;
  deg_error=(delta_fc + pha_i - phase_estim);
  //printf("\nphase_estim:%f deg_error:%f ",phase_estim,deg_error);
   do
   {
    deg_error=(delta_fc + pha_i - phase_estim);
    rad_error=deg_error/360*2*pi;
    VCO=K*sin(4*rad_error);
    phase_estim = phase_estim + phase_estim;
    i++;
```

```
  }while(i<=10);
 d++;
 } while(fabs(deg_error)>=7.5);
 *deg_er=deg_error;
}

//function of gaussian noise generator
void AWGN_channel(double R_AWGN[],double I_AWGN[])
{
 double x1,x2;
 for(i=1;i<=nn;++i)
 {
    x1=(double)rand()/rmax;
    if (x1<=1e-38){x1=1e-38;}
    x2=(double)rand()/rmax;
    R_AWGN[i] = sqrt(-2.0*var*log(x1))*cos(2.0*pi*x2);
    I_AWGN[i] = sqrt(-2.0*var*log(x1))*sin(2.0*pi*x2);
//              printf("\nR_AWGN[%d]:%f \tI_AWGN[%d]:%f",i,R_AWGN[i],i,I_AWGN[i]);
 }
}
//function of noise addition
void noise_addition(double R_Rx[],double I_Rx[])
{
// double R_Tx_n,I_Tx_n;
 for(i=1;i<=nn;++i)
 {
    R_Rx[i] = R_Tx[i] + R_AWGN[i];
    I_Rx[i] = I_Tx[i] + I_AWGN[i];
 }
}

//function of phase received
void phase_received(double pha_Rx[])
{
  double y_x,deg_error=0;
  for(i=1;i<=nn;++i)
  {
   y_x=I_Rx[i]/R_Rx[i];
   y_x=atan(y_x)/2/pi*360  + deg_error;

    if(R_Rx[i]>=0.000000 && I_Rx[i]>=0.000000)
       pha_Rx[i]=y_x;
        else if(R_Rx[i]>=0.000000 && I_Rx[i]<0.000000)
          pha_Rx[i]=y_x+360.0;
   else if((R_Rx[i])<0.000000 && (I_Rx[i]>0.000000))
       pha_Rx[i]=y_x+180.0;
   else if((R_Rx[i])<0.000000 && (I_Rx[i]<=0.000000))
       pha_Rx[i]=y_x+180.0;
   else
  printf("");
  }
}

void phase_comparator(double pha_comp[])
{
    double pha_Rx0=0.0;
 for(i=1;i<=nn;++i)
```

```
 {
  pha_comp[i] = pha_Rx[i] - pha_Rx0;
  if (pha_comp[i]>=0.0)
  {pha_comp[i]=pha_comp[i];}
  else if (pha_comp[i]<0.0)
  {pha_comp[i]=pha_comp[i]+360.0;}
  else
  printf("");

  pha_Rx0=pha_Rx[i];
//  printf("\npha_Rx[%d]:%5.2f pha_comp[%d]:%5.2f ",i,pha_Rx[i],i,pha_comp[i]);
 }
}

void phase_symbol(double R_Rec[],double I_Rec[])
{
 double R_Rec0=1.0,I_Rec0=1.0,shift_Rx;
for(i=1;i<=nn;++i)
 {
  shift_Rx=(pha_comp[i]/360.0)*(2.0*pi);
   R_Rec[i]=R_Rec0*cos(shift_Rx) - I_Rec0*sin(shift_Rx);
  I_Rec[i]=R_Rec0*sin(shift_Rx) + I_Rec0*cos(shift_Rx);
//  printf("\nR_Rec[%d]:%f I_ Rec[%d]:%f",i,R_Rec[i],i,I_Rec[i]);
 }
}


//function of DQPSK bit and symbol decission
void symb_decision(int info_rec[],double rec_Dr[],double rec_Di[])
{
for(i=1;i<=nn;++i)
 {
    if (R_Rec[i]>=0.0 && I_Rec[i]>=0.0)
    {info_rec[i]=0;rec_Dr[i]=1.0;rec_Di[i]=1.0;}
    else if (R_Rec[i]<0.0 && I_Rec[i]>=0.0)
    {info_rec[i]=1;rec_Dr[i]=-1.0;rec_Di[i]=1.0;}
    else if (R_Rec[i]<0.0 && I_Rec[i]<0.0)
    {info_rec[i]=2;rec_Dr[i]=-1.0;rec_Di[i]=-1.0;}
    else if (R_Rec[i]>=0.0 && I_Rec[i]<0.0)
    {info_rec[i]=3;rec_Dr[i]=1.0;rec_Di[i]=-1.0;}
    else
    {printf("Decission error");exit (0);}
    }
}

//function of error detection
void error_detect(double *ber,double *ser)
{
    double Symb_error=0.0,bit_error=0.0,r_error=0.0,i_error=0.0;
    for(i=2;i<=nn;++i)
    {
    if (Dr[i]==rec_Dr[i])
   r_error=0.0;
    else
   r_error=1.0;

    if (Di[i]==rec_Di[i])
```

```
            i_error=0.0;
        else
            i_error=1.0;

        bit_error += (r_error+i_error);

        if (r_error+i_error>0)
        Symb_error +=1.0;
        else
        Symb_error = Symb_error;
        }
    BER=bit_error/(2*(nn-1));
    *ber=BER;
    SER=Symb_error/(nn-1);
    *ser=SER;
    printf("\nSER:%f BER%f",SER,BER);
    }



    void sig_noise_ratio(double *snr)
    {
        double gama;
        gama = 1.0/var;
        SNR=10*log10(gama)+2.3;
        *snr=SNR;
    printf("\nSNR:%f",SNR);
    }

    void error_prob(double *ps,double *pb)
    {
        double gama1;
        double x,term_1,term_3,term_5;
        gama1 = 1.0/var*cos(deg_error/360*2*pi);
        x=sqrt(2.0*gama1)*sin(3.1415/sqrt(2.0)/4.0);
        term_1=1.0/(2.0*pow(x,2));
        term_3=1.0*3.0/(2.0*2.0*pow(x,4));
        term_5=1.0*3.0*5.0/(2.0*2.0*2.0*pow(x,6));
        Ps_DQPSK=(exp(-x*x)/(x*sqrt(3.1415)))*(1-term_1+term_3-term_5);
        *ps=Ps_DQPSK;
        Pb_DQPSK=0.5*Ps_DQPSK;
        *pb=Pb_DQPSK;
        printf("\t Pb_DQPSK:%f",Pb_DQPSK);
    }
    Ps_DQPSK=2*Pb_DQPSK*(1-0.5*Pb_DQPSK);
}
```

6. 3.3. Simulation Result

The simulation result for the Differential QPSK (DQPSK) system is like in the Table 6.2.

Table 6.2. BER valure of DQPSK system

Eb/No	Pb (theory)	Simulation result
5.31	0.024131	0.024001
5.77	0.018773	0.018363
6.28	0.013668	0.013133
6.86	0.009092	0.008606
7.53	0.005303	0.004953
8.32	0.002516	0.002342
9.29	0.000837	0.000772
9.87	0.000385	0.000371
10.54	0.000138	0.000156
11.33	0.000033	0.000051
12.30	0.000004	0.000029

From the Table 6.2 we can see that the different value between probability of error and bit error rate (BER) as simulation result. As example for value of Eb/No = 5.31, the DQPSK probability of error value is 0.024131. But the simulation result is 0.0233963. In this case the bit error rate happen is smaller than the theoretical value (probability of error). It happens until the value of Eb/No = 9.87. But for higher Eb/No, the value of bit error rate as simulation result is higher than the value of probability of error. The example is at Eb/No = 10.54 the value of probability of error is 0.000138 and the value of bit error rate is 0.000156.

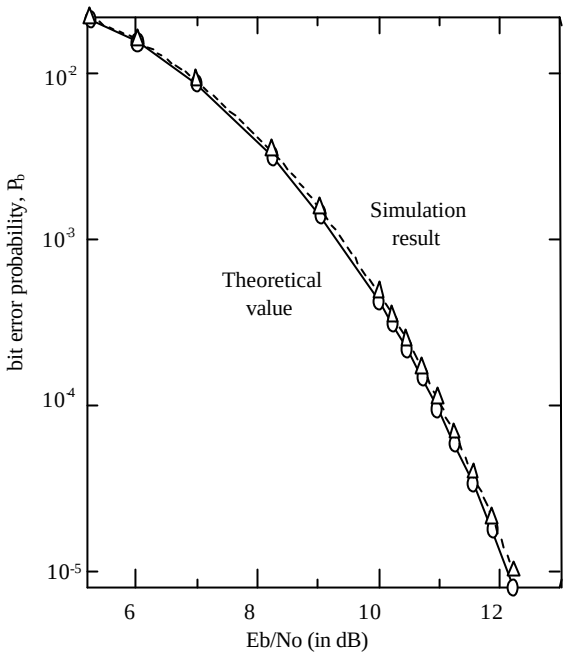


Figure (6.14) Performance evaluation of DQPSK in the AWGN Channel

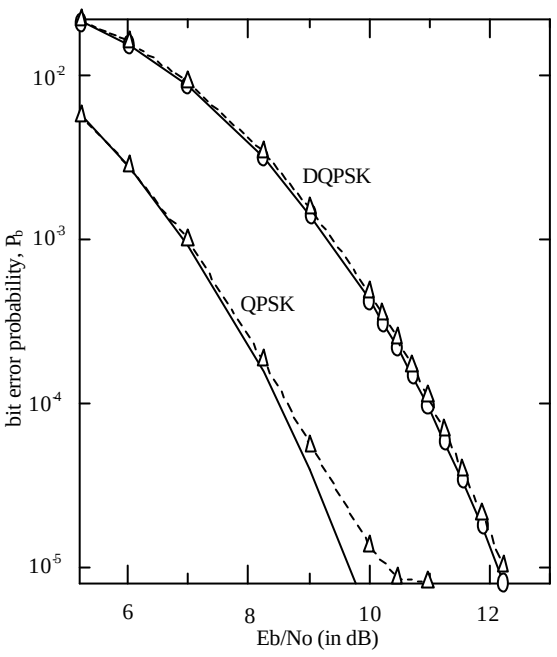


Figure (6.15) Performance comparison of QPSK and DQPSK in the AWGN Channel by simulation

The DQPSK performance comparison by simulation between QPSK and DQPSK in the

AWGN channel can see in the Figure (6.14). For a small value of Eb/No, the simulation result showed a good performance and very closed to pit comparison in the theoretically. But for high Eb/No value the simulation result showed the worse performance, and the comparison of QPSK and DQPSK performance in the AWGN channel is different with the theory.