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

Author: Tri Budi Santoso

Supervisors: Hiroshi SUZUKI

Kazuhiko FUKAWA

Book Reference:

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

Topic: QPSK Transmission System

Book Reference: Digital Communications, Proakis, John G.

Topic: Differential PSK (DPSK) (5.2.8) pp: 274 - 278

4.1 Session 1: Review of Differential PSK

4.1.1 The Concept of DPSK

BPSK and FSK uses double side band suppressed carrier (DCBSC). So, the discrete carrier frequency is abstains, it will be very difficult to receive the incoming signal by using coherent detection. The receiver must use a carrier recovery to make synchronization with the transmitter. One effect of carrier recovery for BPSK system is phase ambiguity between 0 and π radiant. Consequently we do not have an exactly phase estimation.

To solve this problem we can use differential detection technique. For binary phase shift keying this technique is called differential phase shift keying (DPSK).

This problem can be solves by using encoding technique. By using differential decoding in the BPSK transmitter, the information bit 1 will be transmitted by shift the phase of modulated signal 180^o relative to the previous phase of the modulated signal. And bit 0 will be transmitted without shift the phase of modulated signal relative to the previous of modulated signal. This signal then called as *differential encoded.*

The encoding operation is described mathematically by logic expression:

$$b_k = a_k \oplus b_{k-1} \tag{4-1}$$

Where a_k is binary information sequence into the encoder, b_k is the output sequence from the encoder, and b_{k-1} is the output encoder at previous time. The $\oplus$ denotes an addition modulo 2, or

in the logic circuit we know this as EXOR logic. The principle of differential encoding can be represented by the Figure (4.1).

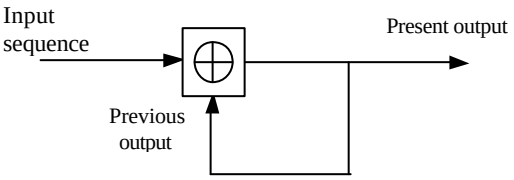


Figure (4.1) Differential Encoder

For example, the output form PN generator of transmitter part is: 1,1,0,0,0,1,0. By using that initialization modulated signal is $\sin(2\pi f_c t+0^o)$. The phase output of modulated signal an the waveform of will be:

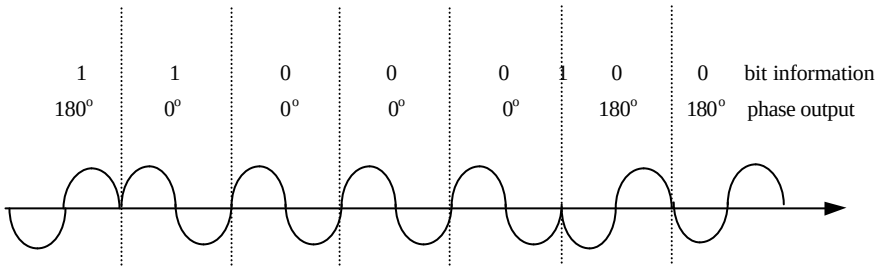


Figure (4.2) Phase shifting and waveform output of binary DPSK

To detect the differential encoded signal we do not require the estimation phase of carrier signal. This technique some times is called as a non-coherent demodulation. By using the phase different between the present symbol and the previous symbol to derive phase, this system then called as DPSK. The DPSK basically is same with BPSK, the receiver use differentially detection technique. The block diagram of DPSK receiver is like Figure (4.3)

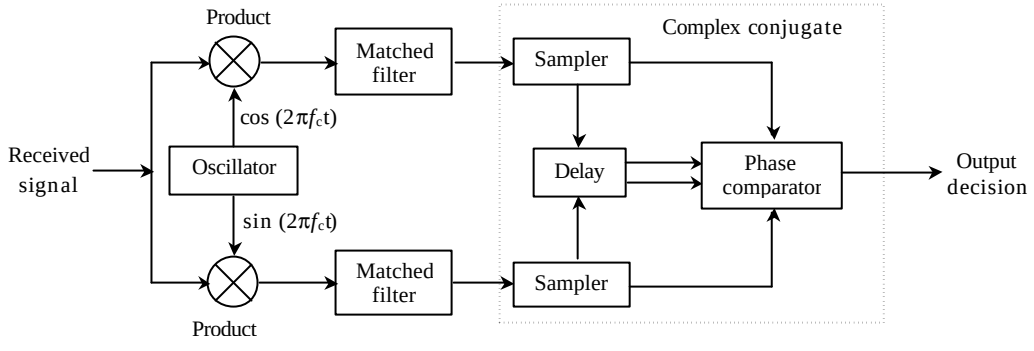


Figure (4.3) Receiver DPSK system

The detection process can be describes as follows. By using the 0^o for phase initialization at the receiver and the received signal from the above example, we can derive the information as:

- phase 1 = $|180^o - 0^o| = 180^o$, → the information is '1'
- phase 2 = $|0^o - 180^o| = 180^o$, → the information is '1'
- phase 3 = $|0^o - 0^o| = 0^o$, → the information is '0'
- phase 4 = $|0^o - 0^o| = 0^o$, → the information is '0'
- phase 5 = $|0^o - 0^o| = 0^o$, → the information is '0'
- phase 6 = $|180^o - 0^o| = 180^o$, → the information is '1'
- phase 7 = $|180^o - 180^o| = 0^o$, → the information is '0'

The general form for Differential Encoding PSK as Transmitter and DPSK Receiver can represent by a block diagram on the Figure (4.4).

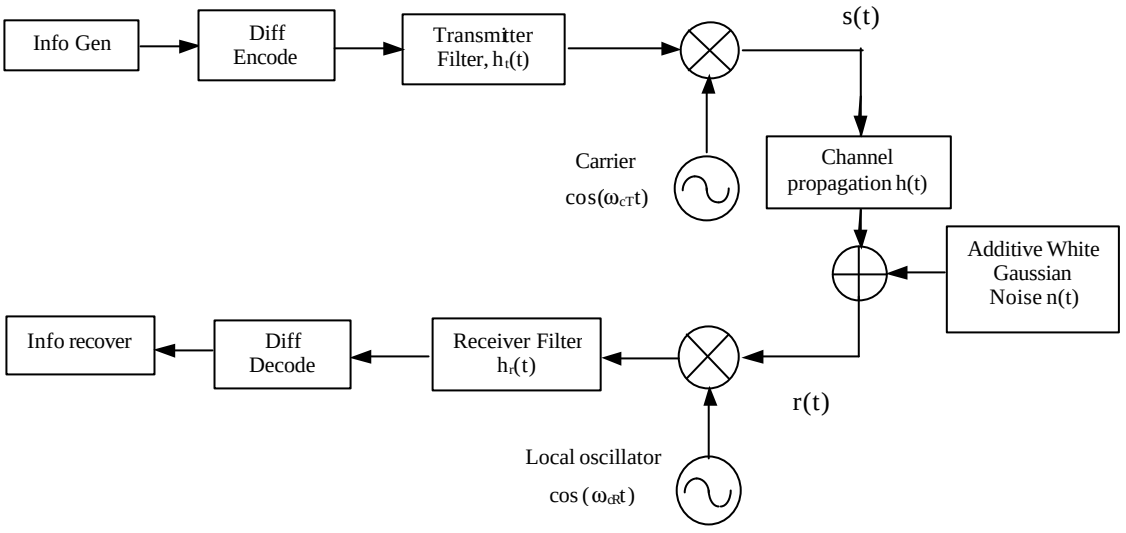


Figure (4.4) Block Diagram BPSK transmission system

4.1.2. AWGN Effect on the DPSK Performance

The additive white Gaussian noise (AWGN) works at all of frequency spectral. And by assumption that the DPSK system works at the ideal condition we can derive the effect of the AWGN similar with it effect on the BPSK coherent system.

We know that in the differential detection, we use two sequence of symbol to detect the phase different, and as we know that the both symbols were added a noise from the channel.

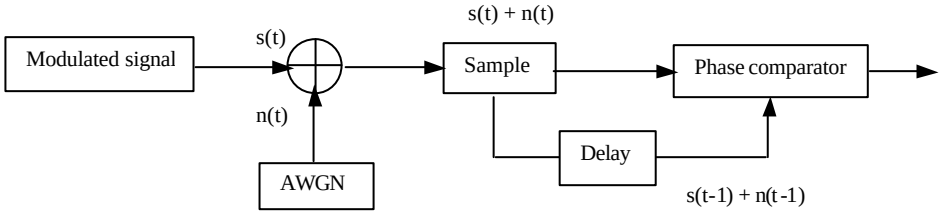


Figure (4.5) AWGN channel effect on the DQPSK

The DPSK signal has the form s(t), the noise form previous signal is n(t-1) and the present signal is n(t). After phase comparator process we will get the signal output that content the two successive noise. If the average energy of modulated signal is E, and the sequences of noise has n_k and n_{k-1} the real part of the received signal can be simplified as:

$$x = E^{1/2} + n_k + n_{k-1} \qquad (4\text{-}2)$$

As we know that the average energy density of AWG noise is No. So why in the DPSK system the performance has the performance 3 dB bellow the BPSK coherent detection.
In the BPSK coherent detection signal to noise ratio is E/No, so the logarithmic form has the value 10 $\log_{10}$ (Eb/No). In the DPSK, each the received signal has 2No, so the signal to noise ratio is Eb/2No, and the logarithmic form has the value 10 $\log_{10}$ (Eb/2No).
By compares these two formulas, we will get:
BPSK: SNR = 10 $\log_{10}$ (Eb/No)
DPSK: SNR = 10 $\log_{10}$ (Eb/2No) = 10 $\log_{10}$ (Eb/No) – 10 $\log_{10}$ (2)

$$= 10 \log_{10} (Eb/No) - 3.01 \text{ dB} \qquad (4\text{-}3)$$

One symbol or bit is received by error in the DPSK system can be approximate by using mathematical expression:

$$P_b = 1/2 \exp(-E_b/N_o) \qquad (4\text{-}4)$$

4.1.4. Carrier-Phase Recovery of DPSK System

The double-sided band suppressed carrier (DSBSC) technique, as like BPSK and FSK the signals spectral are symmetric with respect to the carrier frequency. In this system the transmission signal doesn't content the discrete carrier frequency. So why in the receiver part, phase-carrier synchronization is required. The carrier regeneration process, then called as *carrier recovery* can be achieved several way. One method for generating a properly phase-carrier for a double-sided band suppressed carrier signal is illustrated by the block diagram

shown in Figure (4.6). This scheme is developed by Costas (1956) and is called as Costas Loop. The receiver signal is multiplied by a $\cos(2\pi f_{cR}t + \phi_{est})$ and $-\sin(2\pi f_{cR}t + \phi_{est})$ which are the output from the VCO. Where f_{cR} is a local carrier frequency and ϕ_{est} is initial phase generated.

In the case of BPSK system s(t) content of $(E_b)^{1/2}\cos(2\pi f_{cT}t + \phi_i)$, where $(E_b)^{1/2}$ is the amplitude of signal transmitted, and in this case we make an assumption that the average value is 1. The f_{cT} is the carrier frequency of transmitter part, and ϕ_i is initial phase of transmitter signal. Further more it can be rewritten as $\cos(2\pi f_{cT}t + \phi_i)$.

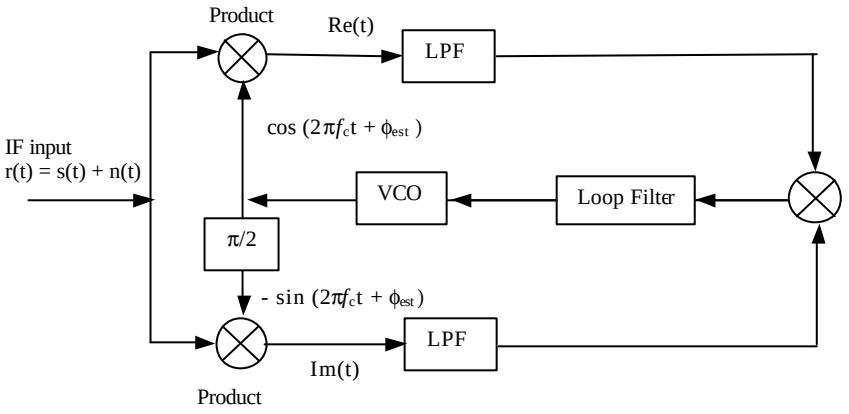


Figure (4.6) Costas loop carrier-phase recovery

There are two channels in the Costas Loop, real (Re) and imaginary (Im).

From the real part, the product output is:

$$y_{Re}(t) = [s(t) + n_{Re}(t)]\cos(2\pi f_{cR}t + \phi_{est})$$
$$= \cos(2\pi f_{cT}t + \phi_i)\cos(2\pi f_{cR}t + \phi_{est}) + n(t)_{Re}\cos(2\pi f_{cR}t + \phi_{est})$$
$$= (1/2)\cos\{2\pi(f_{cT} + f_{cR})t + (\phi_i + \phi_{est})\} + (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} \quad (4\text{-}5)$$
$$+ n_{Re}(t)\cos(2\pi f_{cR}t + \phi_{est})$$

After low pass filter process

$$y_{Re}(t) = (1/2)\cos\{2\pi(f_{cT} - f_{cR})t + (\phi_i - \phi_{est})\} + n(t)_{Re}\cos(2\pi f_{cR}t + \phi_{est})$$
$$= (1/2)\cos(2\pi\Delta f t + \Delta\phi) + n_{Re}(t)\cos(2\pi f_{cR}t + \phi_{est}) \quad (4\text{-}6)$$

Where the Δf is different value between frequency carrier at transmitter and receiver parts, and $\Delta\phi$ is different value between initial phase at transmitter and receiver parts.

From the imaginary part, the product output is:

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

$$- n_{Im}(t)\sin(2\pi f_{cR}t + \phi_{est}) \quad (4\text{-}7)$$

After low pass filter process

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

$$= (1/2)\sin(2\pi\Delta f t + \Delta\phi) - n_{Im}(t)\sin(2\pi f_{cR}t + \phi_{est}) \quad (4\text{-}8)$$

Where the Δf is different value between frequency carrier at transmitter and receiver parts, and $\Delta\phi$ is different value between initial phase at transmitter and receiver parts.

Multiplying the two outputs of low pass filter generates an error signal. Thus,

$$e(t) = y_{Re}(t)\, y_{Im}(t)$$

$$= [(1/2)\cos(2\pi\Delta f t+\Delta\phi) + n_{Re}(t)\cos(2\pi f_{cR}t+\phi_{est})][(1/2)\sin(2\pi\Delta f t+\Delta\phi) - n_{Im}(t)\sin(2\pi f_{cR}t+\phi_{est})]$$

$$= (1/4)\cos(2\pi\Delta f t +\Delta\phi)\sin(2\pi\Delta f t +\Delta\phi) + n(t)_{Re}\cos(2\pi f_{cR}t+\phi_{est})(1/2)\sin(2\pi\Delta f t+\Delta\phi) -$$

$$(1/2)\cos(2\pi\Delta f t+\Delta\phi)\, n_{Im}(t)\sin(2\pi f_{cR}t+\phi_{est}) - n_{Re}(t)\cos(2\pi f_{cR}t+\phi_{est})\, n_{Im}(t)\sin(2\pi f_{cR}t+\phi_{est}). \quad \text{.....(4-9)}$$

We note that the error signal into the loop filter consists of the desired term $(1/4)\cos(2\pi\Delta f t +\Delta\phi)\sin(2\pi\Delta f t +\Delta\phi)$ plus terms that involve *signal* x *noise* and *noise* x *noise.* It is interesting to note that the optimum low pass filter for rejecting the double-frequency terms in the Costas Loop filter is a filter matched to the signal pulse in the information-bearing signal. If matched filters are employed for the low-pass filters, their outputs could be sampled at the bit rate, at the end of each signal interval, and the discrete-time signal samples could be used to drive the loop. The use of the matched filter results in a smaller noise into the loop. And it will give the matched filter condition where is signal >> noise. The signal to drive VCO can be rewritten as:

$$e(t) = (1/4)\cos(2\pi\Delta f t +\Delta\phi)\sin(2\pi\Delta f t +\Delta\phi)$$

$$= (1/8)\sin\{2(2\pi\Delta f t +\Delta\phi)\} \quad (4\text{-}10)$$

The new output phase of VCO is given as

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

or

$$2pf_{cR}t + f_{est} + K\int_0^t \sin\left(2(2p\Delta f t + \Delta f)\right)dt \quad (4\text{-}11)$$

where is K = Kc/8.

The loop phase error and its time derivative is

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

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

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

$$\frac{d(q_e(t))}{dt} = \sin(2\Delta f) \qquad (4\text{-}12)$$

The null value will be happen at 0 and π radiant, and it's known as two phase ambiguity of phase-carrier recovery as in BPSK system, but it doesn't matter for DPSK system.

4.2. Session 2: Question and Answer

Question 1

Why in the DPSK topic we always include the *differential encoding*?

Answer:

In the DPSK the information content come from the phase different between the present and the previous signals. At the modulator we use *differential encoding* before transmit the signal. And in the receiver we can detect the phase different by compare the phase of any signal with the phase of the previous signal. This demodulator technique is called *differential detection*. The differentially encoded PSK signaling that is demodulated and detected as this system is called *differential PSK* (DPSK).

Question 2:

Where I must set the AWG noise in the DPSK for my simulation?

Answer:

We build the simulation in the base band system; it's similar with the simulation for the BPSK system with coherent detection or may be we call as coherent BPSK. The amplitude of transmitted signal is 1 for a bit "1" and –1 for a bit "0". We must remember, that BPSK has antipodal signals. The AWGN must be set on the transmitted signal and has a function as additive component. The receiver part will receive the transmitted signal plus the noise from AWGN channel. As we know that in the DPSK system we only have one channel, in this case is the real channel, so the noise that we use for AWGN is real part of AWGN.

4.3. Session 3: Simulation of DPSK

4.3.1. Algorithm

The simulation for DPSK transmission system using base band system, and here we use differential detection. Based on the Figure (4.7), to construct the program simulation of DPSK system we must build some function:

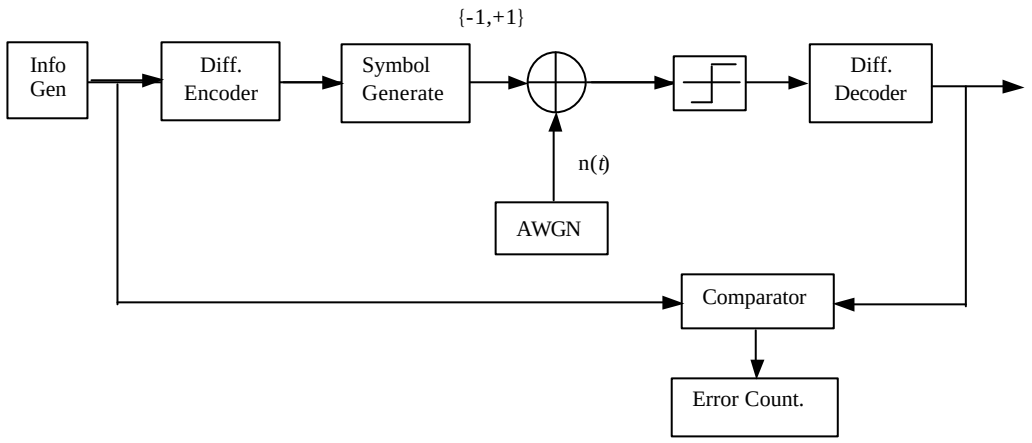


Figure (4.7) Block Diagram for DPSK Simulation

- **Info Generator**

Bit information generated randomly by Info generator function. The random number generated is set between value 0 and 1 as the binary number to represent digital signal. By using Microsoft C++ software, this function will generate a uniform random binary number

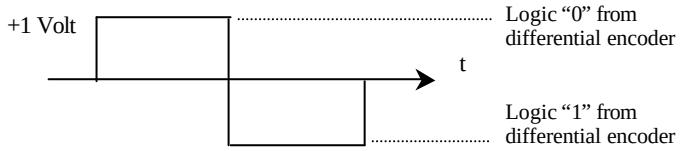
- **Differential Encoding**

Based on the Figure (1.1) and the equation (1-1), we can build the differential encoder function. The output from Info generator then is differentially encoded by this function. We set bk_1 = 0 as initial condition for the previous output, because in the initial condition the previous output is 0. If the bit sequence input is a_k = 1,1,0,0,0,1,0, the output will be b_k = 1,0,0,0,0,1,1.

- **Symbol Generator**

Basically we can use the symbol generate function same as we used in the BPSK simulation

system. But in this simulation we set if the output from differential encoding is 1, we set the phase generated is 180^o or π radiant or for antipodal signal form we set it as -1. Otherwise for the output from differential encoding is 0, we set the phase generated is 0^o or 0 radiant or for antipodal signal form we set it as 1. May be this rule of setting is difficult to understand, but actually the setting parameter for simulation is free for programming, the important thing is the simulation must be represent the characteristic of the system



Figure(4.8). Base band antipodal for DPSK simulation system

♦ **AWGN Channel**

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

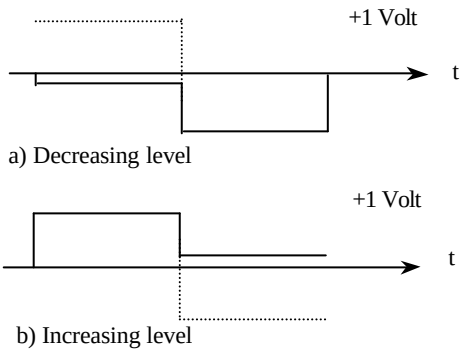


Figure (4.9) Effect AWGN at the level of antipodal signal transmission

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*

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

.

In the DPSK system actually we need the real and imaginer channels, because the basic formulation for the differential detector is by using the complex conjugate formulation. But we can set the real channel only for our simulation, because the value of imaginer part in DPSK case is very small compare with the real part.

- **Noise Addition**

By using this function we sum the output from Symbol Generate and AWGN channel. Its represent the addition of the signal transmission from transmitter and AWGN channel.

- **Signal to Noise Ratio Calculation**

Signal to noise ratio is a ratio of power average of signal transmitted and noise power average. In case of DPSK the power average of signal transmitted is 1 and the average power noise is equivalent with variant value. The result of this calculation is in dB value; this technique is same with the BPSK case.

- **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 between 0 and π radiant. In the case of DPSK it doesn't matter, because the information content can be detected from the phase different between sequence signal transmitted. We can detect the information content by compare the phase of present signal with the phase of previous signal.

- **Symbol Recover and Decision**

We must build 3 functions in this stage. Signal arrives at receiver is an output from the Noise Addition function. Its signal contents the signal transmission and AWGN components. We must change from the amplitude value to phase value. By mathematical trigonometric formulation *q = acos(amplitude)*, we will get the value of the phase of signal received. But some condition must make to avoid the absolute value of amplitude > 1.

The phase result from this mathematical formulation then saved as phase_1. By setting initial condition of phase_0 = 0, then we detect the different between present phase (phase_1) and the previous phase (phase_0). After one cycle finish, we change the value of the previous by the value of the present phase, phase_0 = phase_1..

After the different of phase detected, we must make a decision to get the bit information. By usin the logic is like in the transmitter part we set if value of phase different is 90< delta phase <270, we set the bit recover as 1, and the other is 0.

- **Error Detector**

The bit received then detect bi compare it to the original bit information at the transmitter part. If bit received has a same value with the bit information, the error is not happen, other wise if the value of bit received is different with the bit information the error is happen. This function is represents the part Comparator in the Figure (4.7).

- **BER Calculation**

The purpose of this part is to get ho many errors was happened during bit transmission process and compare this total error value to the total bit was transmitted. The result is known as *bit error rate* (BER), and the value is between 0 and 1. If we transmit 1000000 bits and 100 bits received by error, we get the information the *bit error rate* 100/1000000 = 10^{-4}.

- **Pb Calculation**

This part is to calculate the value of probability of error that will happen if we transmit a signal through an AWGN channel, which has some value of noise variants. By the equation (4-4) we can calculate this value.

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.

Re_info = information generate by transmitter part.

pi = π radiant.

Re_Tx = symbol generate by transmitter part.

pha_Tx = phase generate by transmitter part

Re_AWGN = noise AWGN

Re_Rx = symbol received at receiver part

Re_anti_Rx = symbol recovered at receiver part in the antipodal format

pha_receiv = phase receive after transmission

phase_error = phase error at a synchronization process

Ave_Pha_err = average phase error in the all the synchronization process

SNR = Signal to Noise Ratio

BER = bit error rate

Pb_DPSK = probability of error of DPSK system

The listing program is as follows.

```
//Program of DPSK Transmission in the AWGN Channel
//edited by Tri Budi Santoso
//Suzuki Laboratory, Mobile Communication Group
//Tokyo Institute of Technology, Japan

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

#define N 10

//declaration of global variable
int nn=N-1,i;
int info[N];
int bn_1[N];
double Tx_symbol[N],pha_Tx[N],pi=acos(-1.0);
double phase_error[N],Ave_Pha_err,Pha_err,pha_receiv,deg_error;
double Re_AWGN[N],var=0.00;
double Rx_symbol[N];
double phase_1[N];
double delta[N];
int bit_recover[N];
double BER;
double Pb_DPSK;
double SNR;

//prototipe function
void info_generate(int info[]);
void diff_encoder(int bn_1[]);
void symbol_generate(double Tx_symbol[],double pha_Tx[]);
void costas(double phase_error[],double *pha_er);
void synchronization(double pha_revceiv,double *deg_er);
void AWGN_Channel(double Re_AWGN[]);
void noise_addition(double Rx_symbol[]);
void symb_phase(double phase_1[]);
void diff_decoder(double delta[]);
void phase_bit(int bit_recover[]);
void error_detect(double *ber);
void error_prob(double *pb_DPSK);
void signal_noise_ratio(double *snr);
void main()
{
FILE *fdata;
char filename[]="DPSK2_data.txt";
fdata=fopen(filename,"a");

srand(2);
info_generate(info);
diff_encoder(bn_1);
symbol_generate(Tx_symbol,pha_Tx);
```

```
costas(phase_error,&Ave_Pha_err);
AWGN_Channel(Re_AWGN);
noise_addition(Rx_symbol);
symb_phase(phase_1);
diff_decoder(delta);
phase_bit(bit_recover);
error_detect(&BER);
error_prob(&Pb_DPSK);
signal_noise_ratio(&SNR);

printf("\nSNR:%4.2f Pb_DPSK:%f BER:%f",SNR,Pb_DPSK,BER);
fprintf(fdata,"\nSNR:%4.2f Pb_DPSK:%f BER:%f",SNR,Pb_DPSK,BER);

fclose(fdata);
}

void info_generate(int info[])
{
for(i=1;i<=nn;++i)
   {
   info[i]=(int)rand()%2;
   }
}

void diff_encoder(int bn_1[])
{
   int bn_0=0;
 for(i=1;i<=nn;++i)
 {
  bn_1[i] = (bn_0 + info[i])%2;
  bn_0=bn_1[i];
  printf("\ninfo[%d]:%d bn_1[%d]:%d",i,info[i],i,bn_1[i]);
 }
}

void symbol_generate(double Tx_symbol[],double pha_Tx[])
{
   double phase_tx;
 for(i=1;i<=nn;++i)
 {
  Tx_symbol[i]=((double)bn_1[i]-0.5)*(-2.0);
  phase_tx=acos(Tx_symbol[i]);
  pha_Tx[i]=phase_tx/pi*180;
//  printf("\nTx_symbol[%d]%4.2f pha_Tx[%d]:%5.2f",i,Tx_symbol[i],i,pha_Tx[i]);
 }
}

void costas(double phase_error[],double *pha_er)
{
   Pha_err=0.0;
 for(i=1;i<=nn;++i)
 {
  pha_receiv=pha_Tx[i];
  synchronization(pha_receiv,&deg_error);
  phase_error[i]=deg_error;
```

```
//  printf("\n phase_error[i]:%f",phase_error[i]);
  Pha_err += fabs(phase_error[i]);
 }
 *pha_er=Ave_Pha_err;
}

void synchronization(double pha_revceiv,double *deg_er)
{
 int d=1,i=1,symbol_rate=16000;
 double rad_error,fc=800e+6,T,delta_fc,phase_estim=0.0;
 double VCO,K=1.0;

 T=1.0/(double)symbol_rate;
 delta_fc=1e-6 * fc *T;

// printf("\n\tpha_receiv:%f",pha_receiv);
   do
   {
       deg_error=(delta_fc + pha_receiv - phase_estim);
       if (deg_error < -45.00)
       {deg_error=deg_error + 360.0;}
       else if (deg_error > 360.0)
       {deg_error=deg_error - 360.0;}
       else if ((deg_error>0.00) && (deg_error<360.00))
       {deg_error=deg_error;}
       else
       printf("");
       rad_error=deg_error/360*2*pi;
       VCO=K*sin(2*rad_error);
       phase_estim = phase_estim + VCO;
       //printf("\ndeg_error:%f",deg_error);
       i++;
   }while(i<=4);

if(deg_error>-90.00 && deg_error<=90)
{deg_error=deg_error;}

else if(deg_error>90.00 && deg_error<=270.00)
{deg_error=deg_error - 180.0;}

else {printf("");}

 *deg_er=deg_error;
}

void AWGN_Channel(double Re_AWGN[])
{
  double x1,x2,rmax=RAND_MAX +1.0,re_AWGN=0.0;
   for(i=1;i<=nn;++i)
   {
       rmax=(double)RAND_MAX +1.0;
       x1=(double)rand()/rmax;
       if (x1<=1.0e-38){x1=1.0e-38;}
       x2=(double)rand()/rmax;
       re_AWGN = sqrt(-2.0*var*log(x1))*cos(2.0*pi*x2);
```

```
      Re_AWGN[i] = re_AWGN;
   }
}


//Function of Noise addition Proccess
void noise_addition(double Rx_symbol[])
{
 double rx_symb;
   for(i=1;i<=nn;++i)
   {
      rx_symb= Tx_symbol[i]* cos(phase_error[i]/320*2*pi)+Re_AWGN[i];
      if ((rx_symb>=-1.0) && (rx_symb<=1.0))
      {Rx_symbol[i]=rx_symb;}
      else if(rx_symb=1.0)
      {Rx_symbol[i]=1;}
      else if(rx_symb<-1.0)
      {Rx_symbol[i]=-1;}
      else
      printf("");
//printf("\nRx_symbol[%d]:%3.2f",i,Rx_symbol[i]);
      }
}

//Function of symbol to phase
void symb_phase(double phase_1[])
{
 for(i=1;i<=nn;++i)
 {
 phase_1[i]=acos(Rx_symbol[i])/2/pi*360;
// printf("\nphase_1[%d]:%f",i,phase_1[i]);
 }
}


//Function of Differential Decoder or Phase Comparator
void diff_decoder(double delta[])
{ double diff,phase_0=0.0;
 for(i=1;i<=nn;++i)
 {
 diff = (phase_1[i] - phase_0);
 if (diff >360.00)
   delta[i]=diff-360.00;
 else if (diff<-360.00)
   delta[i]=diff +360.00;
 else if (fabs(diff)<=360)
   delta[i]=diff;
 else
   printf("");
// printf("\ndelta[%d]:%f",i,delta[i]);
 phase_0=phase_1[i];
 }
}
//function of phase to bit convertion
void phase_bit(int bit_recover[])
```

```
{
   for(i=1;i<=nn;++i)
   {
   if (fabs(delta[i])>=90 && fabs(delta[i])<270)
      bit_recover[i]=1;
   else
      bit_recover[i]=0;
   printf("\nbit_recover[%d]:%d",i,bit_recover[i]);
   }
}

//Function of Error Detect
void error_detect(double *ber)
{
double  bit_error=0.0;
for(i=1;i<=nn;++i)
{
   if (info[i]==bit_recover[i])
      bit_error =bit_error;
   else
      bit_error +=1.0;
}
BER=bit_error/nn;
*ber=BER;
printf("\nBER:%f",BER);
}

void error_prob(double *pb_DPSK)
{
 Pb_DPSK=cos(Ave_Pha_err/360*2*pi)*0.5*exp(-1.0/var);
*pb_DPSK=Pb_DPSK;
}

void signal_noise_ratio(double *snr)
{
   SNR=10*log10(1.0/var);
   *snr=SNR;
}
```

3.3. The Simulation Result

We can run the program after set the bit information = 1000000. It means that we send the 1000000 bits information every variants value. After run and change the value of variants from 0.5 until 0.1, we get the result as like in the Table 4.1.

Table 4.1. The result of BPSK Simulation

SNR (in dB)	Pb:	BER
3.01	0.067863	0.086521
3.47	0.054358	0.070304
3.98	0.041191	0.055414
4.56	0.028835	0.041234
5.23	0.017923	0.027937
6.02	0.009211	0.015367
6.99	0.003393	0.005624
8.24	0.000642	0.000705
9.03	0.000170	0.000117
10.00	0.000023	0.000017

Some different value between theoretical (Pb) values with simulation result (BER) was happen. For the variants value 0.5, which give the energy bit per noise (E_b/N_o) = 3,01 dB. The P_b shows the

value; the meaning is if we send 1000000 bits information, probability of total bit will be received by error are 67863 bit. But the simulation result showed the value 86521 bit were received by error or BER = 0.086521. From this data we look that the different value is 0.086521 - 0.067863 = 0,018654 or in the percent value is 27.48%.

For a value of E_b/N_o = 6,99 dB. The theoretical estimate Pb = 0.003393, the meaning is if we send 1000000 bits information data yang the probability of bit received by error are 3393 bits. But the simulation result showed the value of BER is 0.005624. From this data we get the different value is 0.005624 - 0.003393= 0.002231 or in the percent value is 65%. For a value E_b/N_o = 10,00 dB. The theoretical estimate Pb = 0.000023, the meaning is if we send 1000000 bits information data yang the probability of bit received by error are 23 bits. But the simulation result showed the value of BER is 0.000017. From this data we get the different value is |0.000017 - 0.000023| = 0.000005 or in the percent value is 29%.

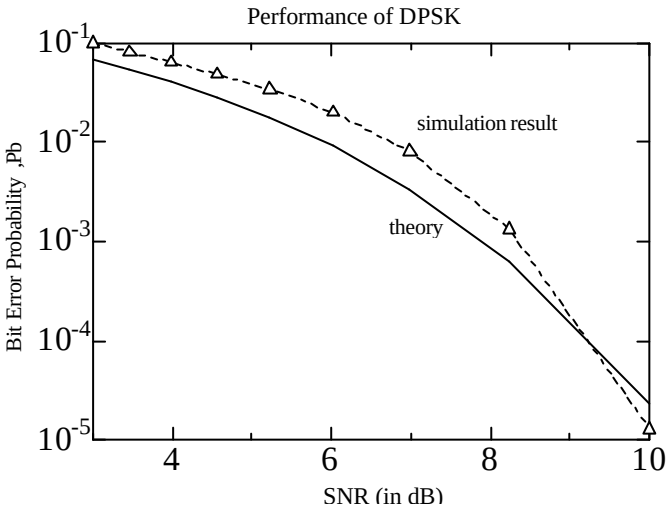


Figure 4.10. Simulation result of DPSK performance in the AWGN Channel

From the result of simulation we get that the different between simulation result and theoretical value is fluctuate. To get the better result we must repeat the simulation by using different random seed and change the add value of total bit transmission

By using the Table 1 we can build a graph for the DPSK performance from the simulation result as Figure 4.10. From this figure we look that the simulation result is closed to the theoretical value, although we didn't get the same value.