รายงาน

เรื่อง Digital Modulation และ Digital Communication

วิชา ปฏิบัติการระบบโทรคมนาคม (Communication System Lab)

เสนอ

อาจารย์ ผศ. สิทธิพร เกิดสำอางค์

จัดทำโดย

นายโสภณ สุขสมบูรณ์ รหัสนักศึกษา 6201011631188

นักศึกษาชั้นปีที่3 สาขาวิชาวิศวกรรมไฟฟ้า (โทรคมนาคม)

ปฏิบัติการครั้งที่ 3

วิชา ปฏิบัติการระบบโทรคมนาคม ประจำภาคการศึกษา 2/2564

สาขาวิชาวิศวกรรมไฟฟ้า(โทรคมนาคม) ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Experiment 6

Digital Modulation

Procedure

In this experiment, the binary data rate R_b is 1 kbps and peak modulated signal amplitude is 1 V. The bit period $T_b = 1/R_b$ is represented by 100 samples.

A . Generation of Modulated Signals

Amplitude-Shift Keying (ASK)

A.1 Generate a binary sequence with the first 5 bits [1 0 0 1 0]:

```
>> b=[1 0 0 1 0 binary(45)];
```

A.2 To generate the ASK signal `sa`, with a carrier frequency of 8 kHz:

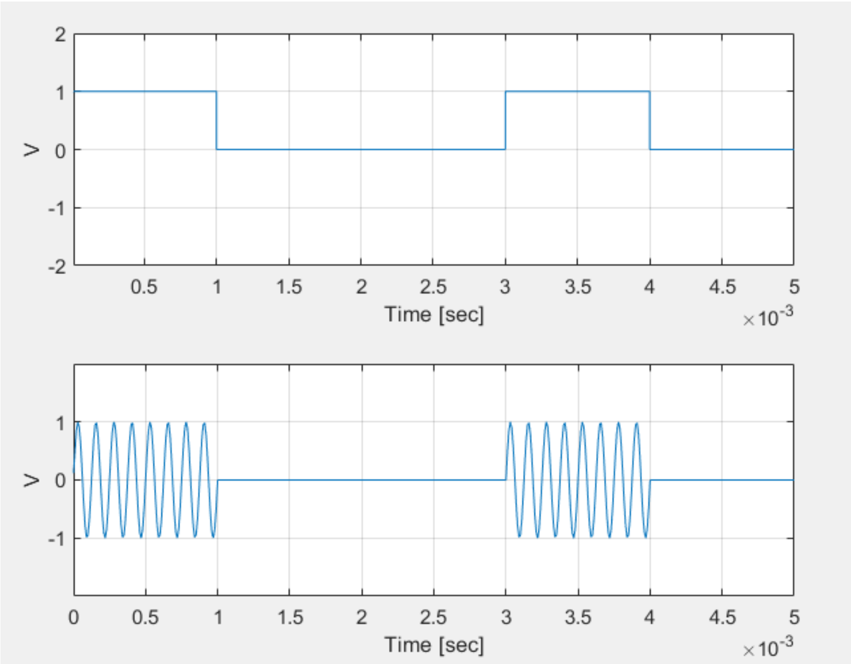
- generate a unipolar NRZ signal `xu`, from the sequence `b`;
- mix `xu` with the output of an oscillator operating at 8 kHz.

```
>> xu=wave_gen(b,'unipolar_nrz');
>> sa=mixer(xu,osc(8000));
```

A.3 Display the first 500 samples of the waveforms `xu` and `sa` representing the first 5 bits in the binary sequence `b`. Compare the two waveforms.

```
>> tt=[1:500];
>> subplot(211),waveplot(xu(tt))
>> subplot(212),waveplot(sa(tt))
```

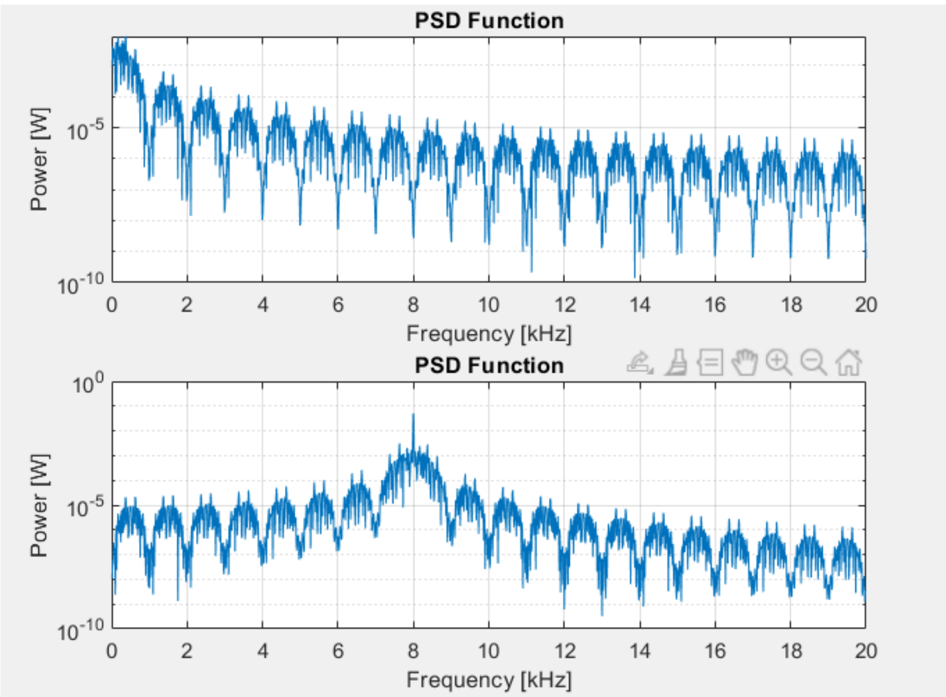
Display



Also display the respective PSD functions over the frequency interval [0, 20 kHz] and record in Graph 6.1 and 6.2. To display the PSD function over a specific frequency range you have issue the `psd` command with two arguments such that `psd(x,freq_range)` displays the PSD of `x` over the frequency interval defined by the vector `freq_range`.

```
>> fr=[0,20000];
>> subplot(211),psd(xu,fr)
>> subplot(212),psd(sa,fr)
>>
```

Display



Phase-Shift Keying (PSK)

A.4 To generate the PSK signal **sp**, with a carrier frequency of 8 kHz:

- generate a polar NRZ signal **xp**, from the sequence b;
- mix **xp** with the output of an oscillator operating at 8 kHz.

```
>> sp=mixer(xp,osc(8000));
>> subplot(211),waveplot(xp(tt))
```

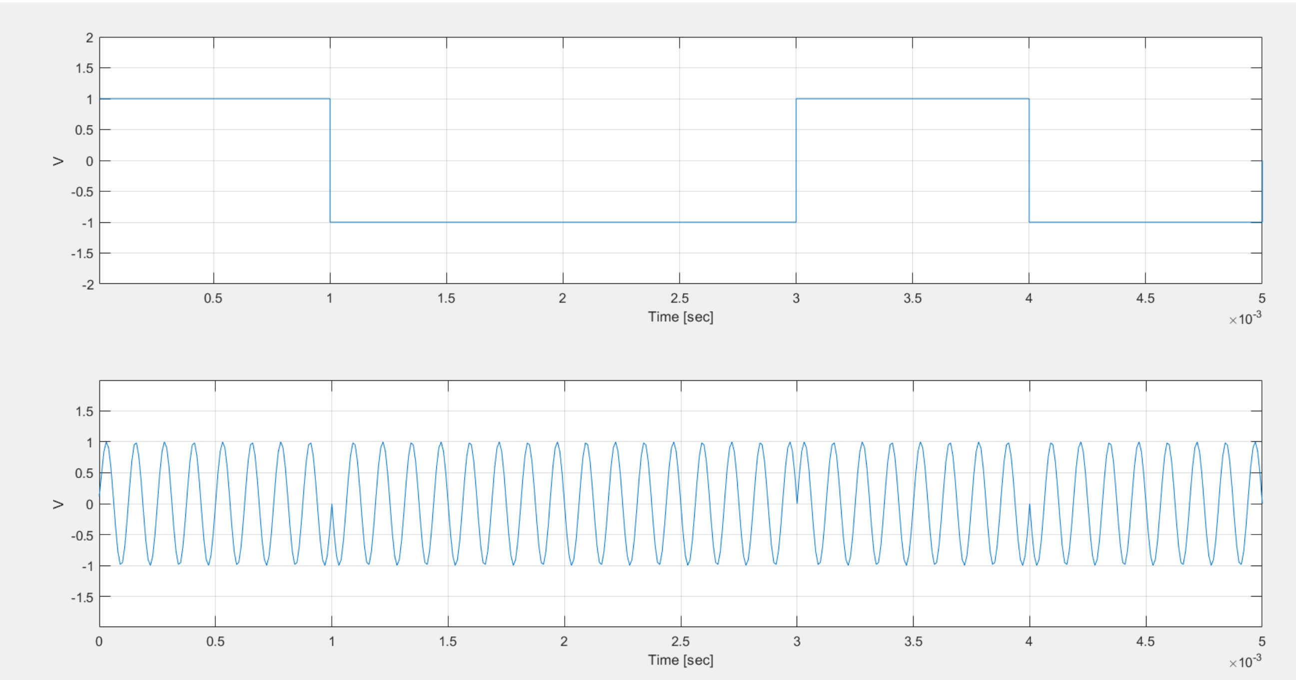
A.5 Display the first 500 samples of the waveforms **xp** and **sp**:

```
>> subplot(211),waveplot(xp(tt))

>> subplot(212),waveplot(sp(tt))
```

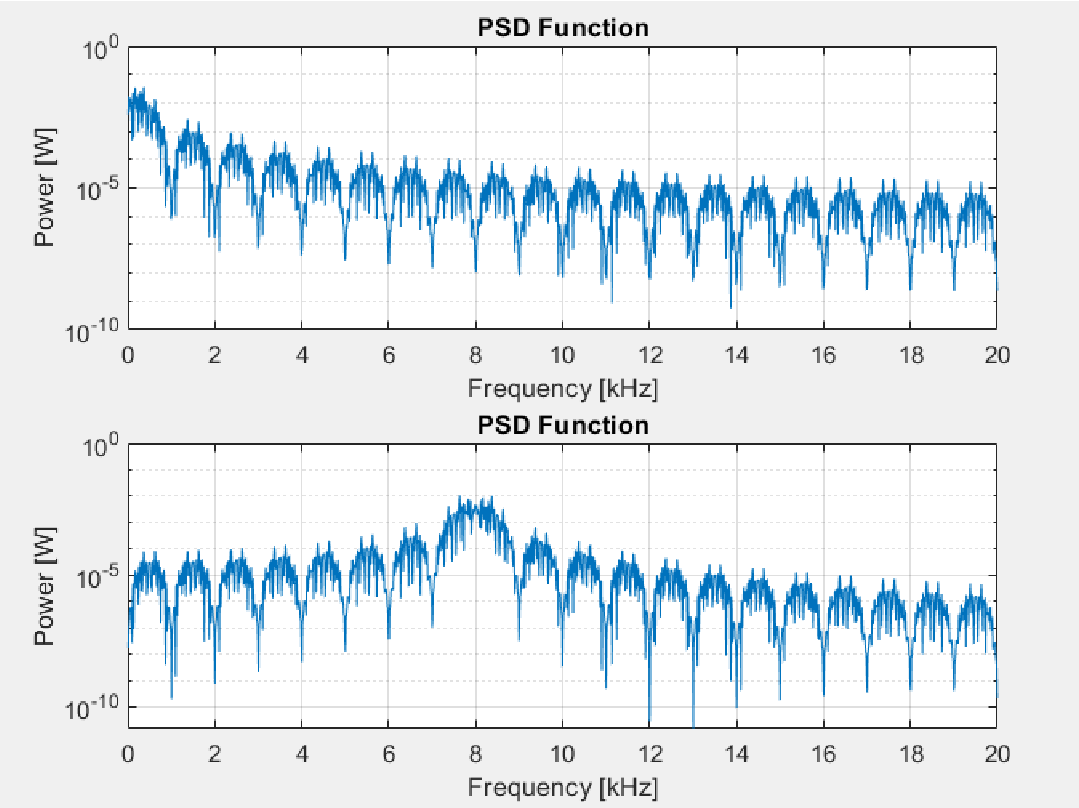
What is the phase difference between **sp** and the carrier $\sin(2\pi f_c t)$ during the first and second bit periods?

เมื่อมีการเปลี่ยนแปลงของบิต จะทำให้ phase ของสัญญาณเปลี่ยนด้วย ทำให้สัญญาณที่ทำ PSK มีความไม่ต่อเนื่องของสัญญาณอยู่

Display



A.6 Display the PSD functions of `xp` and `sp` over the frequency interval [0, 20 kHz]. Record main characteristics of each PSD function.



Frequency-Shift Keying (FSK)

A.7 To generate the *continuous phase* FSK signal `sf`, with *mark* and *space* frequencies of 4 and 8 kHz, respectively:

- generate a polar NRZ signal from the sequence `b`;
- apply the polar waveform to the input of a voltage controlled oscillator (VCO). In this experiment the VCO has the free-running frequency set to 6 kHz and has frequency sensitivity of -2 kHz/V.

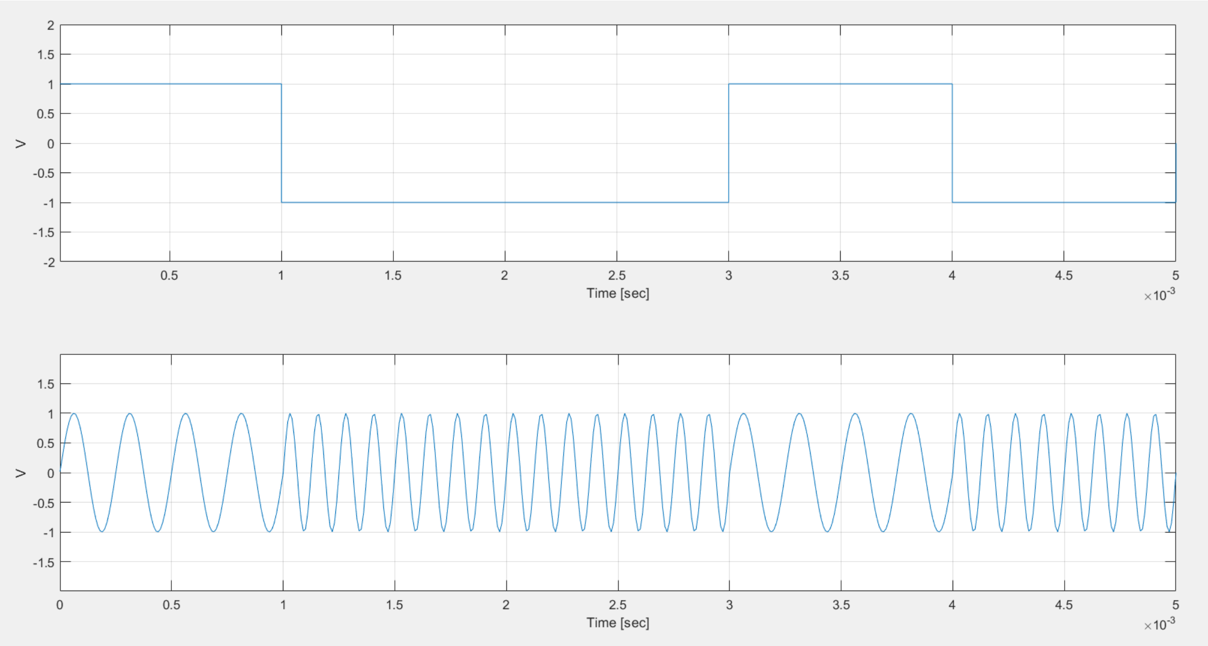
```
>> sf=vco(xp);
```

A.8 Display waveforms `xp` and `sf` for $0 < t < 5\,T_b$.

```
>> subplot(211),waveplot(xp(tt))
```

```
>> subplot(212),waveplot(sf(tt))
```

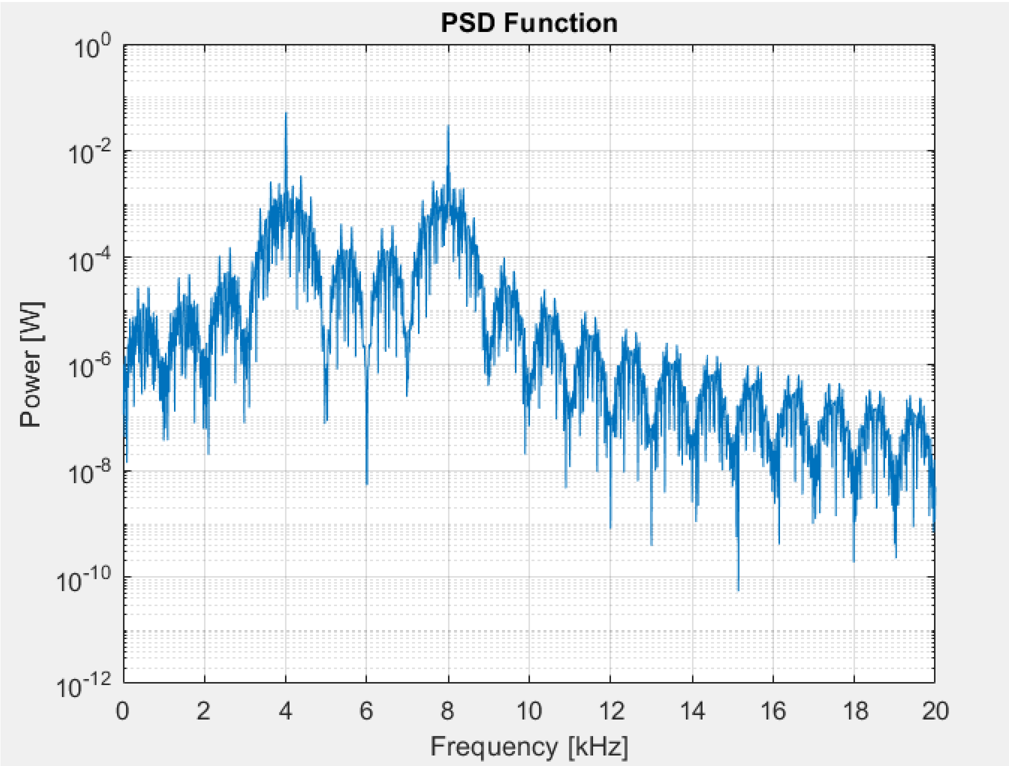
Display



Display the PSD function of the FSK signal and record in Graph 6.5.

```
>> clf
>> psd(sf,fr)
fx >>
```

Display



Q6.1 How can you generate an FSK signal from two ASK signals? For a system where efficient bandwidth utilization is required, which modulation scheme would you prefer?

ใช้ความถี่ซึ่งต้องหารกับคาบเวลาลงตัวเพื่อให้ได้สัญญาณContinuousซึ่งมีข้อดีคือใช้Bandwidth ต่ำ

B . Digital Modulated Signal Detection

Coherent Detection

B.1 A coherent detector for ASK and PSK signals is depicted in Fig. 6.1.

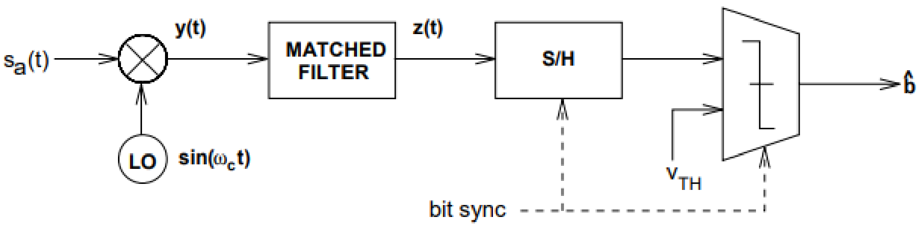
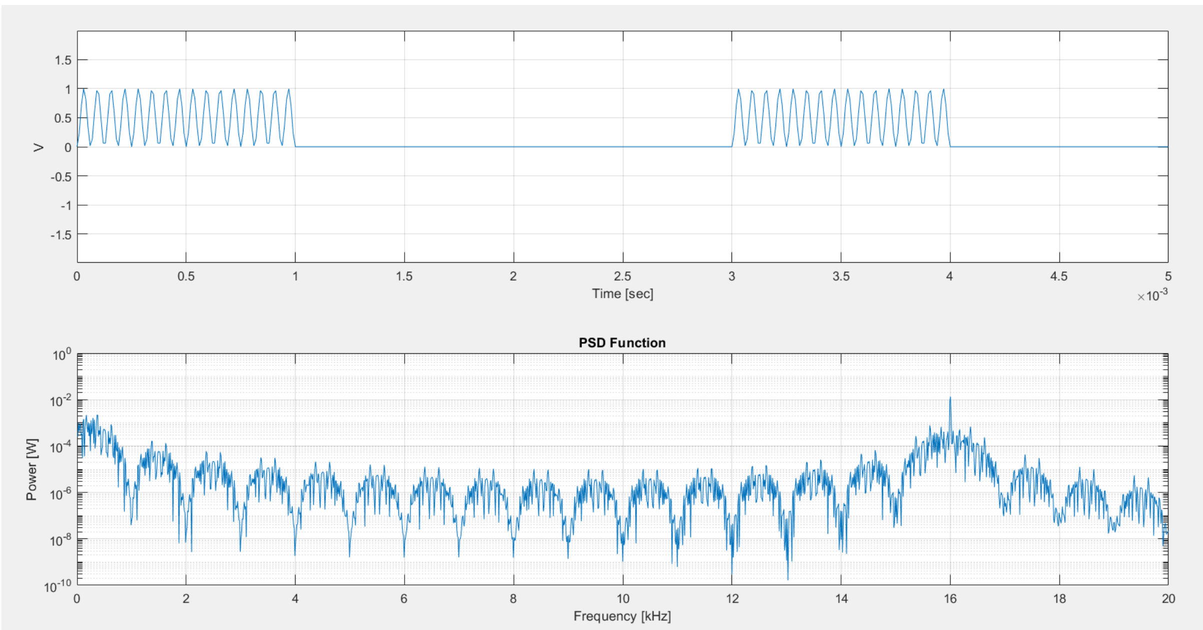


Fig. 6.1 Coherent Detector

To demodulate the ASK signal `sa`, first multiply `sa` by a locally generated carrier which has the same frequency and phase as the carrier used in generating `sa`. Display the waveform `ya` at the output of the multiplier for the first five bit periods. Also display the corresponding PSD function over the interval `fr` and record in Graph 6.6.

```
>> ya=mixer(sa,osc(8000));
>> clf,subplot(211),waveplot(ya(tt))
```

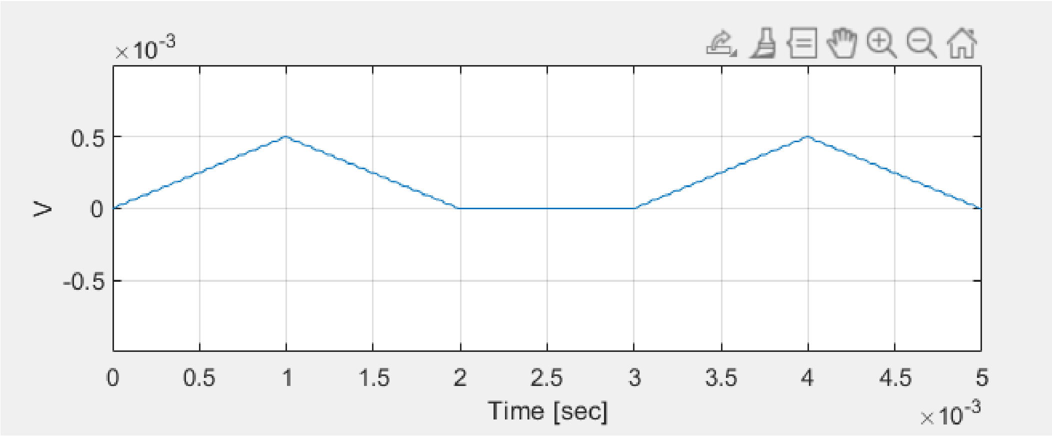
```
>> subplot(212),psd(ya,fr)
```



B.2 Apply ya to a matched filter and record its output for $0 < t < 5\,T_b$.

```
>> za=match('unipolar_nrz',ya);
>> subplot(212),waveplot(za(tt))
```

Display



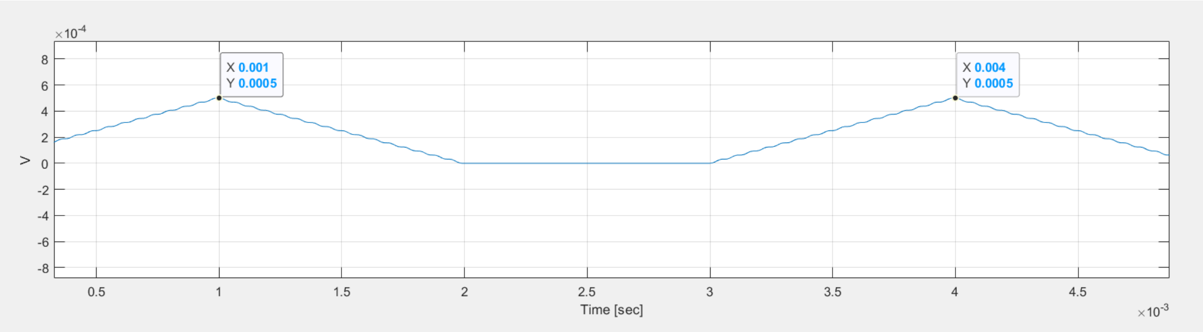
> **Q6.2** Determine the impulse response of the matched filter. Note that za is similar to the output of the matched filter for a unipolar NRZ signal. Why?

The major difficulty in implementing a coherent detector is carrier synchronization. In order to achieve optimum performance, the local oscillator should have the same phase and frequency as the incoming carrier. Phase or frequency deviation will result in degradation of detection performance.

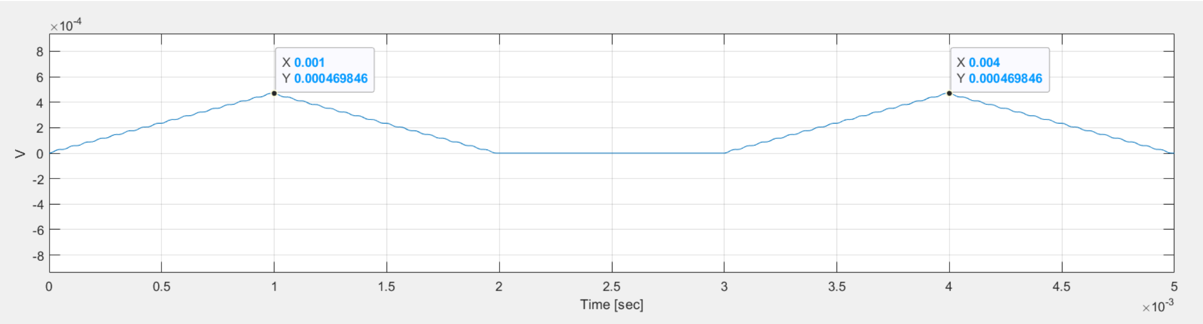
B.3 To observe the effect of phase error, demodulate sa using a local oscillator whose output is $\sin(2\pi f_c + \phi)$. Here, ϕ is the phase error measured with respect to the carrier. Record the peak signal amplitude at the matched filter output for each phase error shown in Table 6.1.

```
>> ya=mixer(sa,osc(8000,0));
>> za=match('unipolar_nrz',ya);
>> subplot(212),waveplot(za(1:500))
```

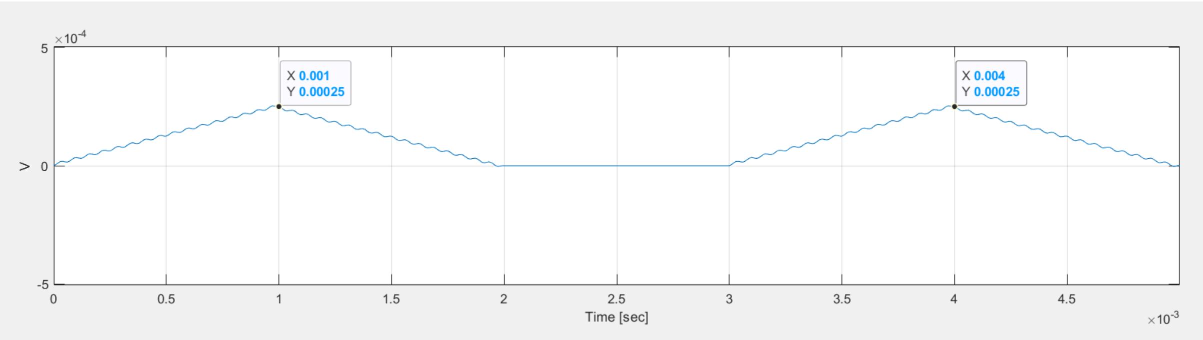
Phase Error : 0-degree   Peak Amplitude : 0.0005 V



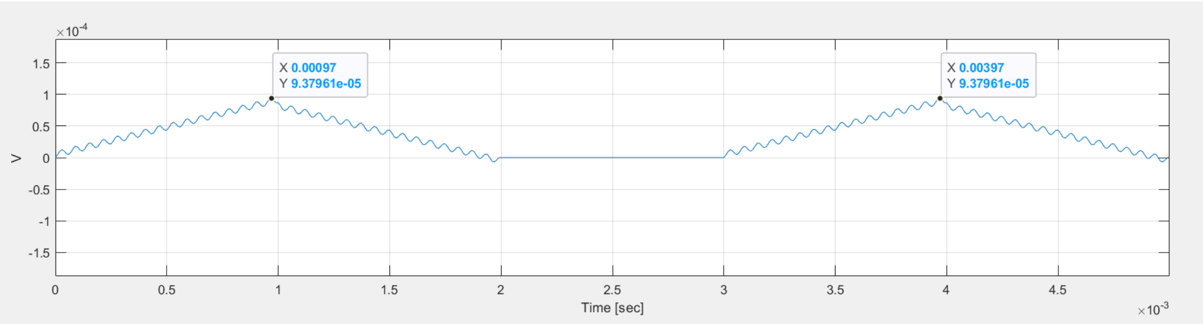
Phase Error : 20-degrees   Peak Amplitude : 0.000469846 V



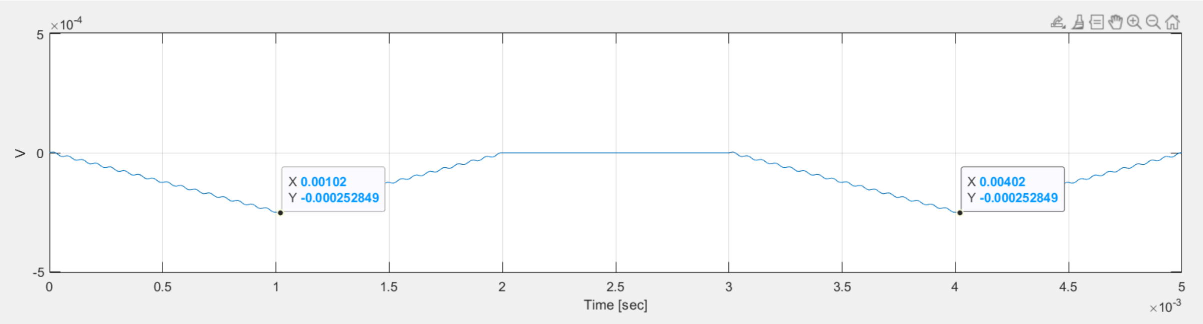
Phase Error : 60-degrees   Peak Amplitude : 0.00025 V



Phase Error : 80-degrees   Peak Amplitude : 0.000093961 V



Phase Error : 120-degrees   Peak Amplitude : -0.000252849 V



Phase Error (Degree)	Peak Amplitude (V)
0	0.0005
20	0.000469846
60	0.00025
80	0.000093961
120	-0.000252849

Table 6.1

Q6.3 Recall that the BER resulting from the detection of a signal in the presence of noise, is a function of peak signal amplitude at the receiver filter output. Determine from the results displayed in Table 6.1 which phase error will result in smallest BER.

ต้องให้ Phase ของผู้รับและผู้ส่ง มีค่าใกล้เคียงกันมากที่สุด จึงจะทำให้ค่า BER ต่ำ เมื่อใช้ Match filter จะทำสัญญาณมาทำ Convolution Integral ถ้า Phase ไม่ตรงกันจะทำให้ค่า Peak ที่ได้ลดลง

B.4 Demodulate sa with 60° and 120° phase errors. Decode the matched filter output to recover the first five bits of the sequence b. Record each decoded sequence and comment on the difference.

Phase error = $60°\ ;\ \hat{b}_{1-5} = 10010$

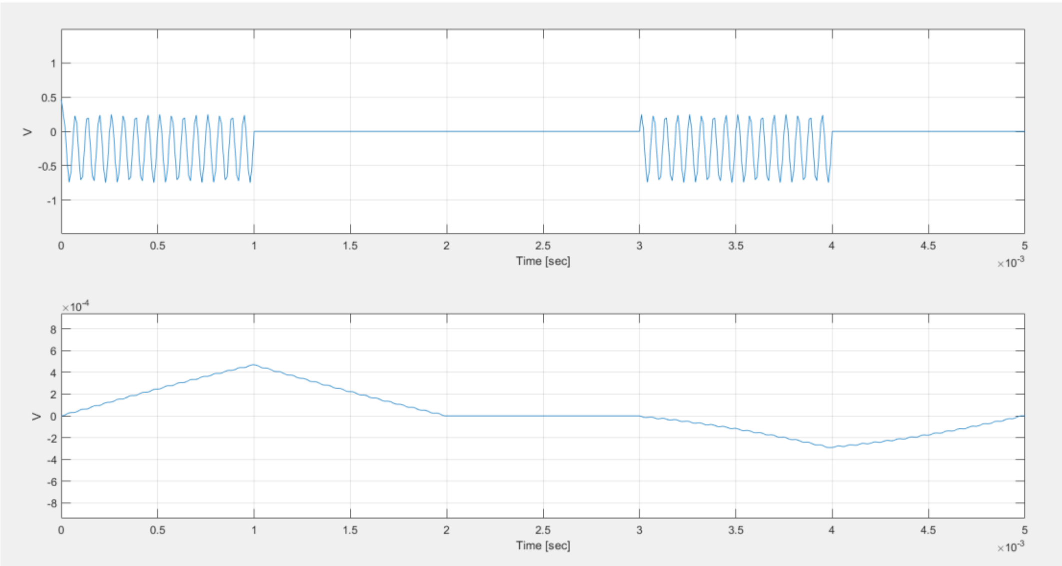
Phase error = $120°\ ;\ \hat{b}_{1-5} = 00000$

กำหนดให้การตัดสินใจคือ เมื่อค่ามากกว่า 0 V หรือ อยู่ฝั่ง V+ ให้ Decode เป็น 1 และน้อยกว่าเท่ากับ 0 หรืออยู่ฝั่ง V- ให้ Decode เป็น 0 และที่ Phase error ทำการ Decode ได้เป็น 00000 นั้น เป็น เพราะว่า ไม่มีส่วนใดเลยที่อยู่เหนือแกน y=0 หรือก็คือไม่มีส่วนใดเลยที่อยู่ในฝั่ง V+ นั่นเอง

B.5 To observe the effect of frequency deviation in demodulating an ASK signal, demodulate sa with a local oscillator set to 7,900 Hz. Display and compare the demodulated signals ya and ya1.

```
>> ya1=match('unipolar_nrz',mixer(sa,osc(7900)));
>> subplot(211),waveplot(ya(tt))
>> subplot(212),waveplot(ya1(tt))
```

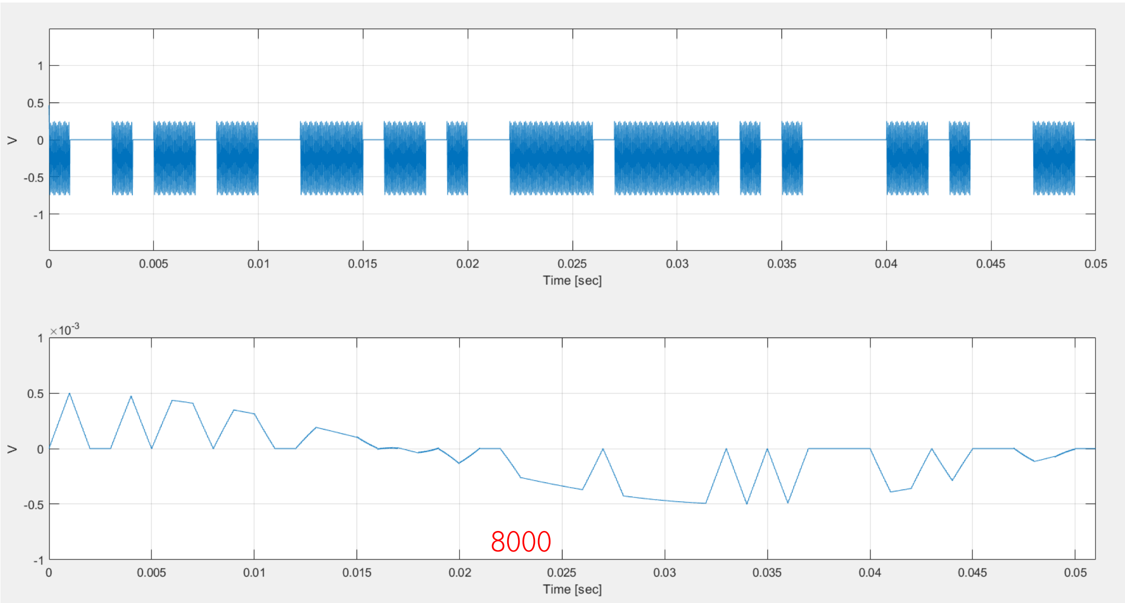
Display



Could the original binary sequence be recovered from ya1? Consider a second case where the local oscillator frequency is set to 7,985 Hz. Demodulate sa and generate the matched filter output:

```
>> ya2=match('unipolar_nrz',mixer(sa,osc(7985)));
>> subplot(211),waveplot(ya),subplot(212),waveplot(ya2)
```

Display



Q6.4 Consider an ASK signal $s_a(t)$ with carrier frequency of f_c. If $s_a(t)$ is demodulated by multiplying with the output of a local oscillator set to f_o, such that $f_o \neq f_c$, the envelope of detector matched filter output is modulated by a sinusoid. Determine the frequency of this modulating signal as a function of f_c and f_o.

$$f = |f_c - f_0|$$

Noncoherent Detection

Noncoherent detection of digital modulated signals does not require synchronization of the local oscillator with the carrier component. However, in the face of corruptive noise, a system using noncoherent detection experiences higher BER relative to coherent detection. Consider the noncoherent detector for an ASK signal shown in Fig. 6.2.

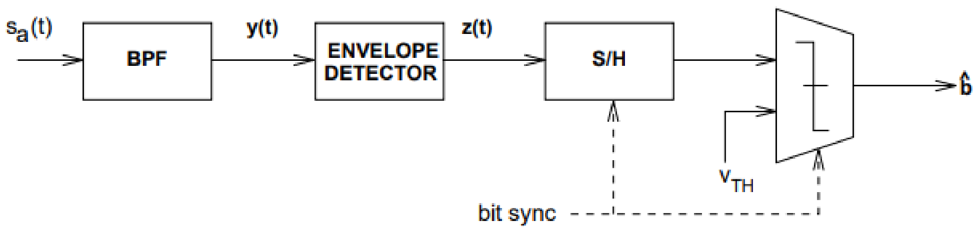


Fig. 6.2 Noncoherent Detector

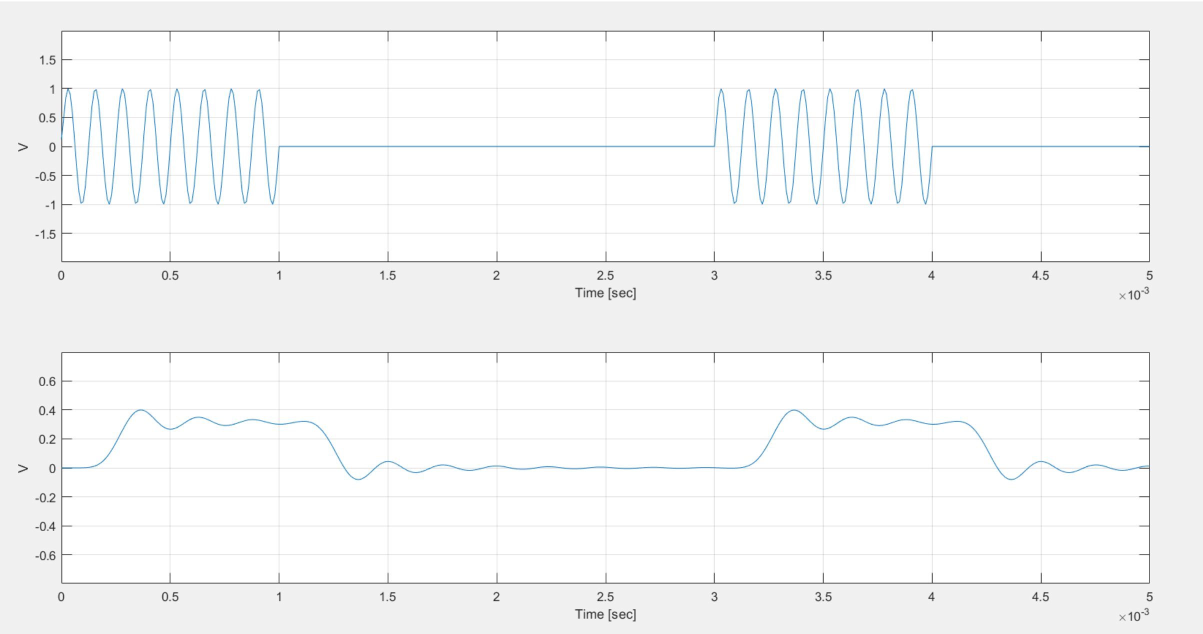
The function of the band-pass filter (BPF) is to reduce the out-of-band noise and interference. Assume the bandwidth of the BPF is appropriately chosen such that signal distortion is negligible; i.e., if the input to the BPF is $s_a(t)$, then the signal at the output of the BPF is also $s_a(t)$. The envelope detector consists of a rectifier followed by a low-pass filter (LPF) with bandwidth f_o, chosen according to the rule:

$$\text{signal bandwidth} \quad \ll \quad f_o \quad \ll \quad \text{carrier frequency.}$$

B.6 Let the bandwidth of the LPF used in the MATLAB function *envelope* be set to 4,000 Hz. Apply the ASK signal sa to the function *envelope* and display its output together with the ASK signal sa:

```
>> ya=envelope(sa,4000);
>> clf,subplot(211),waveplot(sa(tt))
>> subplot(212),waveplot(ya(tt))
```

Display



Decode the first 5 bits of the transmitted sequence.

1 0 0 1 0

Q6.5 Could noncoherent detection be used with PSK signals?

ไม่สามารถใช้ PSK signal ได้ เรื่องจากเป็นการทำ Phase Shift ซึ่งจะทำให้ขอบของสัญญาณเท่ากัน ทำให้ไม่สามารถ detect ได้

C . System Performance Under Noise

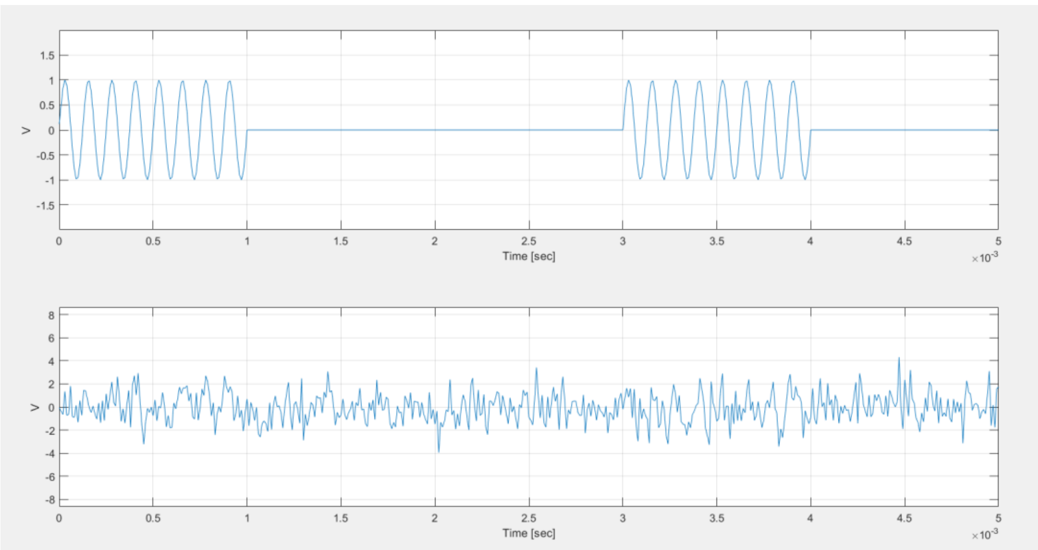
C.1 Generate an ASK signal representing a 500-sample binary sequence:

```
>> b=[1 0 0 1 0 binary(495)];
>> sa=mixer(wave_gen(b,'unipolar_nrz'),osc(8000));
>>
```

C.2 Apply sa to a channel with unity gain, channel noise $\sigma_n^2 = 1$ W, and of sufficient bandwidth such that no distortion is introduced to the signal. Display the ASK signal sa and the channel output y for $0 < t < 5T_b$.

```
>> y=channel(sa,1,1.5,49000);
>> subplot(211),waveplot(sa(tt))
>> subplot(212),waveplot(y(tt))
```

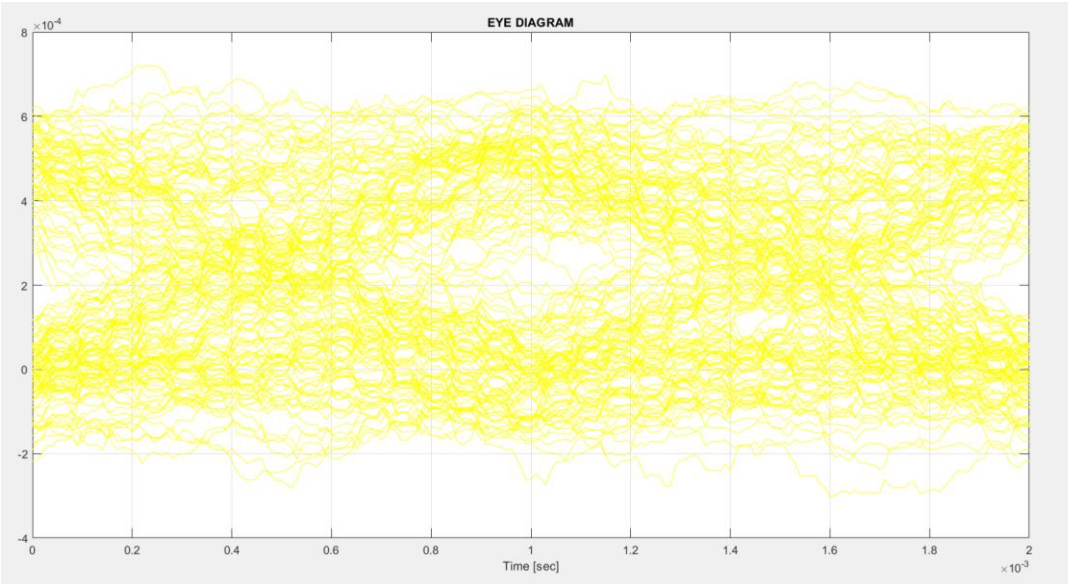
Display



C.3 Use a coherent detector to demodulate y. Display the eye diagram of the matched filter output.

```
>> zm=match('unipolar_nrz',mixer(y,osc(8000)));
>> clf,eye_diag(zm);
fx >>
```

Display




```
Probability of bit error (BER) =   0.008000.
fx >>
```

Q6.6 Compute the theoretical probability of bit error for the case considered above. Recall that the PSD function of the channel noise is

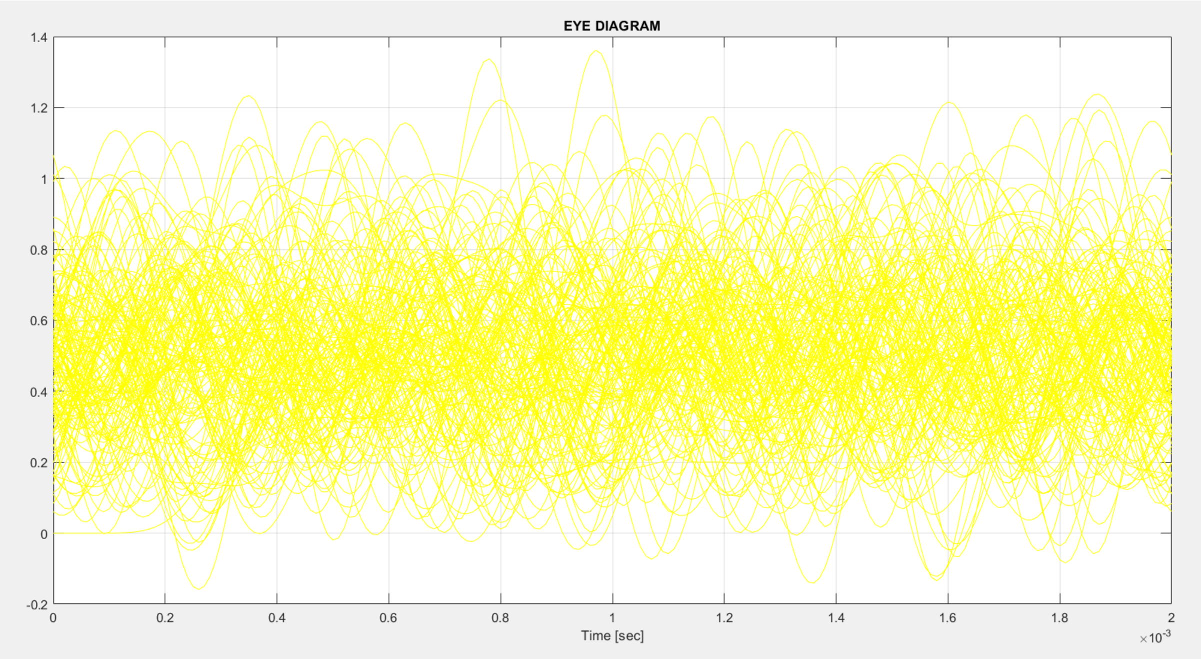
$$S_n(f) = \frac{N_o}{2} = \frac{\sigma_n^2}{2 \times \text{system bandwidth}}.$$

The system bandwidth in this experiment is 50 kHz.

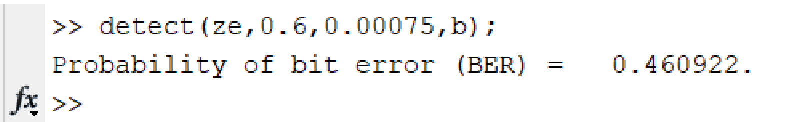
C.4 Use noncoherent detection to decode the bit sequence from the channel output y. Compare the resulting BER with the coherent case.

```
>> ze=envelope(y,4000);
>> clf,eye_diag(ze);
fx >>
```

Display



V_th = 0.6 sampling_instant = 0.00075



Experiment 7

Digital Communication

Procedure

A . Introduction

A.1 Analog Waveform to Channel Code Transformation;

The block diagram depicted in Fig 7.1 represents how an analog signal is transformed first into a digital format and then into a form compatible with channel characteristics. The main functions are the A/D converter and the transmitter represented by *a2d* and *tx*.

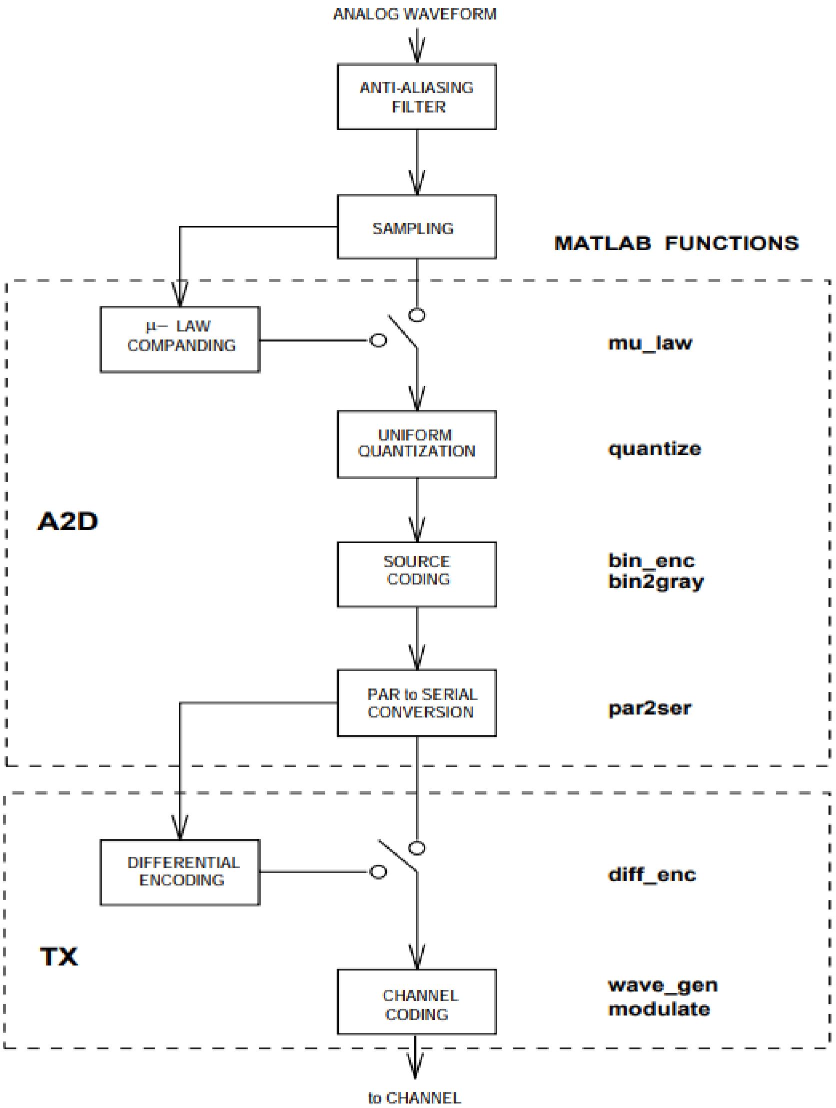


Fig. 7.1 A/D and transmitter block diagrams

A.2 Channel Output to Analog Waveform Transformation:

The block diagram depicted in Fig 7.2 represents how the channel output is processed by the receiver to recover the transmitted binary sequence. The estimated binary sequence is subsequently converted into an analog waveform. The two main blocks are the receiver and the D/A converter represented by the MATLAB functions *rx* and *d2a*.

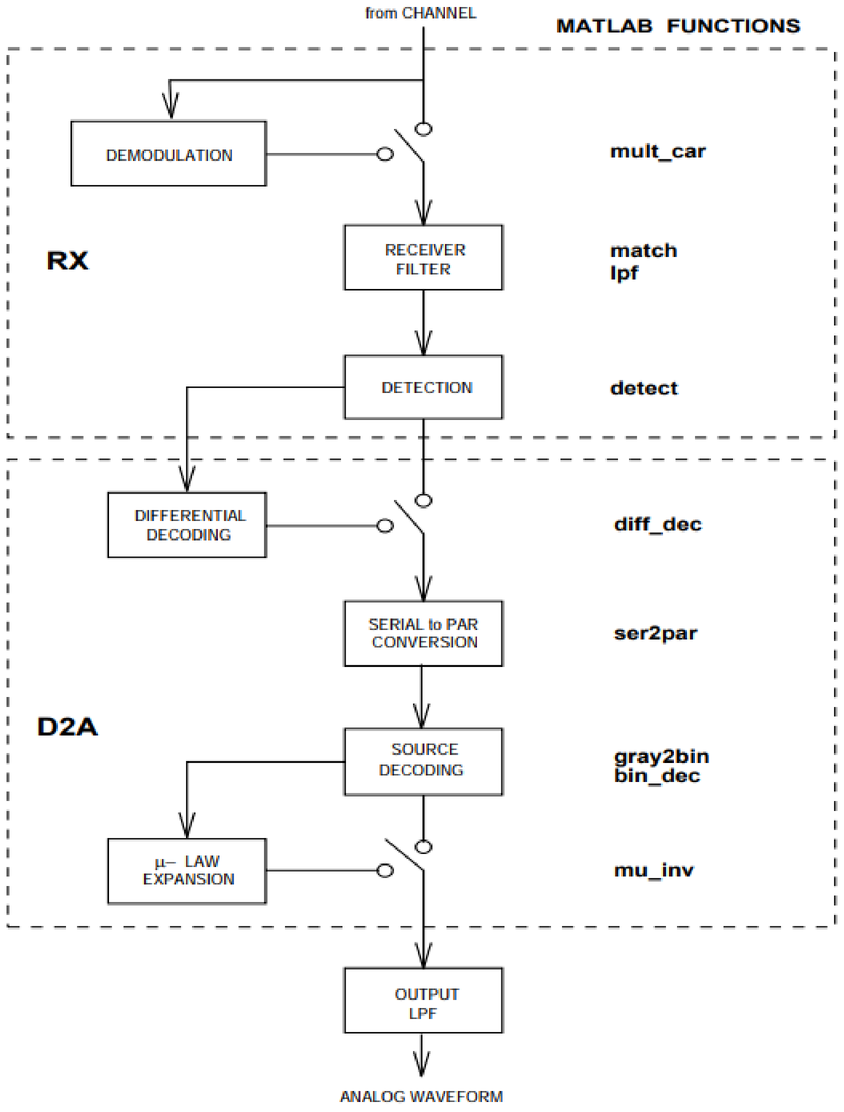


Fig. 7.2 Receiver and D/A block diagrams

REMARKS

- The MATLAB functions *a2d, d2a, tx* and *rx* have been designed to simplify and to automate tasks that constitute each block. Use the on-line help facility to obtain more information about each function.
- The output from the transmitter is send over the communication channel represented by *channel* and will serve as the input to the receiver.
- Depending on the channel characteristics you have to modify the sampling instances of the waveform at the output of the receiver filter. This information can be best extracted from the eye diagram at the filter output. One of options of the receiver function *rx* is to display the eye diagram and to prompt the user for the optimum sampling time. Type `help rx` to learn how to use this function.

B . A/D and D/A Conversion

Consider the problem of transmitting a message over a digital data channel. If the message signal is analog, it must be first converted into an equivalent digital representation. Within the present simulation environment, the analog waveform is in the form of a sampled data sequence. Thus, the filtering and sampling functions shown in Fig 7.1 are not implemented. You may recall from earlier experiments that the process of converting an analog signal into binary data is achieved by applying some or all of the following MATLAB functions:

- μ-law companding — *mu_law* (optional);
- uniform quantization — *quantize*;
- natural binary source coding — *bin_enc*;
- gray-code source coding — *bin2gray* (optional);
- parallel to serial conversion — *par2ser*.

The MATLAB function *a2d* contains all the above functions and directs the analog input data to appropriate functions according to user specified parameters.

Conversely, binary data at receiver output must be converted into analog form. The MATLAB function *d2a* represents the D/A conversion process, performed by applying the following functions on binary data:

- serial to parallel conversion — *ser2par*;
- gray-code source decoding — *gray2bin* (optional);
- natural binary source decoding — *bin_dec*;
- μ-law expansion — *mu_inv* (optional).

To test that functions *a2d* and *d2a* are inverse functions of each other, generate 100 samples from a typical speech signal:

```
>> s=speech(100);
>> s_binary=a2d(s,6);

                   A/D CONVERSION
            --------------------------


o PERFORMING QUANTIZATION :
     Quantization complete.
o PERFORMING SOURCE CODING :
     Natural binary coding;
         Natural Binary coding complete;
     Source coding complete.
o PERFORMING PARALLEL-TO-SERIAL CONVERSION :
     Parallel-to-serial conversion complete.
>> s_analog=d2a(s_binary,6);

                DECODING CHANNEL OUTPUT
            -------------------------------


o PERFORMING SERIAL-TO-PARALLEL CONVERSION :
     Serial-to-parallel conversion complete.
o PERFORMING BINARY-TO-QUANTIZED-ANALOG CONVERSION :
     Binary to quantization level conversion;
         BINARY to quantization level conversion complete;
     Source decoding complete.
o PERFORMING QUANTIZED-ANALOG-TO-ANALOG CONVERSION :
```

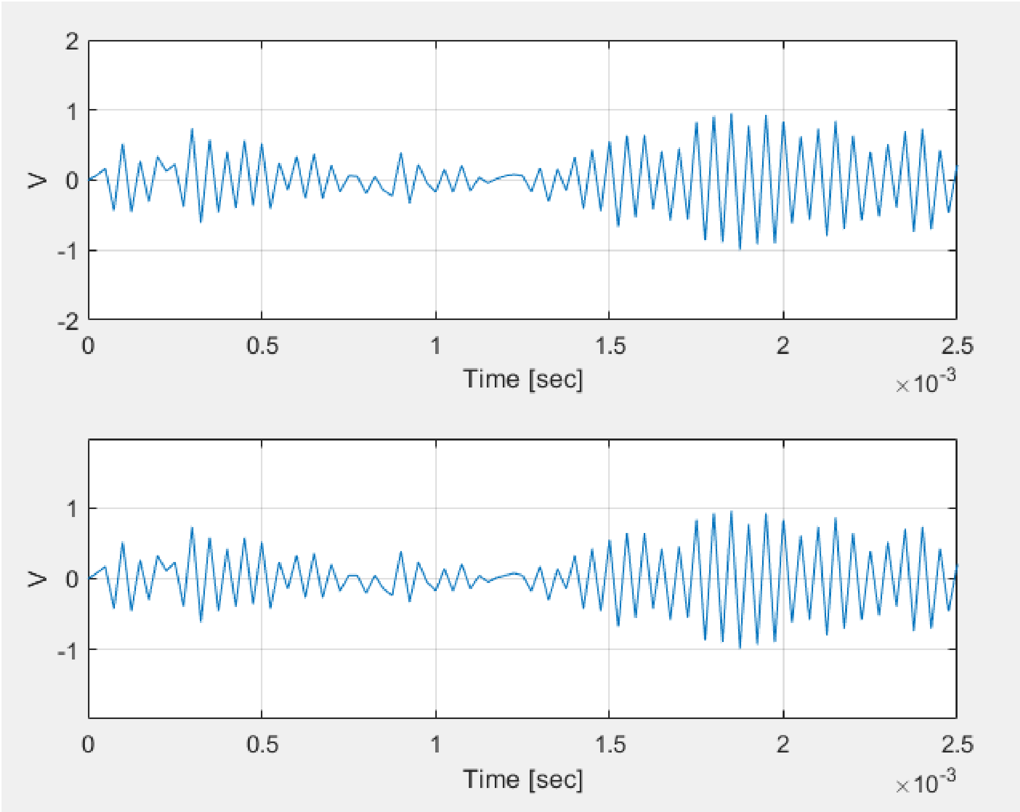
Verify that `s_binary` is indeed a binary sequence by displaying its first few elements:

```
>> s_binary(1:10)

ans =

  Columns 1 through 8

     0     1     0     0     0     1     1     0

  Columns 9 through 10

     1     0
```

Now compare the message signal represented by the data array `s` and the output from the A/D-D/A conversion, `s_analog`:

```
>> subplot(211),waveplot(s)
>> subplot(212),waveplot(s_analog)
```

Display



Q7.1 Does the process of converting analog signals first into digital and back to analog domain introduce any distortion? If your answer is yes, clearly state different types of distortion encountered in an *analog* – *digital* – *analog* conversion system and comment on which parameters have a direct effect on minimizing distortion.

การแปลงสัญญาณจาก Analog มาเป็น Digital ย่อมมีความเพี้ยนของสัญญาณเนื่องจากเราไม่ได้ทำทุกค่ามาทำเป็น Digital Signal แต่เราจะทำการ Sampling ค่ามาบางส่วนให้ใกล้เคียงกับค่าเดิมมากที่สุด ยิ่งเรา Sampling ค่ามาจาก Analog มากเท่าไหร่ Error ที่ได้ก็จะลดลง แต่ก็เพิ่มความ Complex ในการถอดรหัสเพิ่มขึ้น

C . Differential Encoding

C.1 Generate 100 samples of a sinusoid, convert into digital domain and prepare to resulting binary data for transmission over a baseband communication channel using *manchester* line code:

```
>> x=sin(2*pi*400*[1:100]/SAMPLING_FREQ);
>> x_pcm=a2d(x,6);

                    A/D CONVERSION
             --------------------------

o PERFORMING QUANTIZATION :
     Quantization complete.
o PERFORMING SOURCE CODING :
     Natural binary coding;
         Natural Binary coding complete;
     Source coding complete.
o PERFORMING PARALLEL-TO-SERIAL CONVERSION :
     Parallel-to-serial conversion complete.
>> xw=tx(x_pcm,'manchester','no_diff',1000);

o PERFORMING CHANNEL CODING :
     Generating waveform in the selected format;
     Channel coding complete.
>>
```

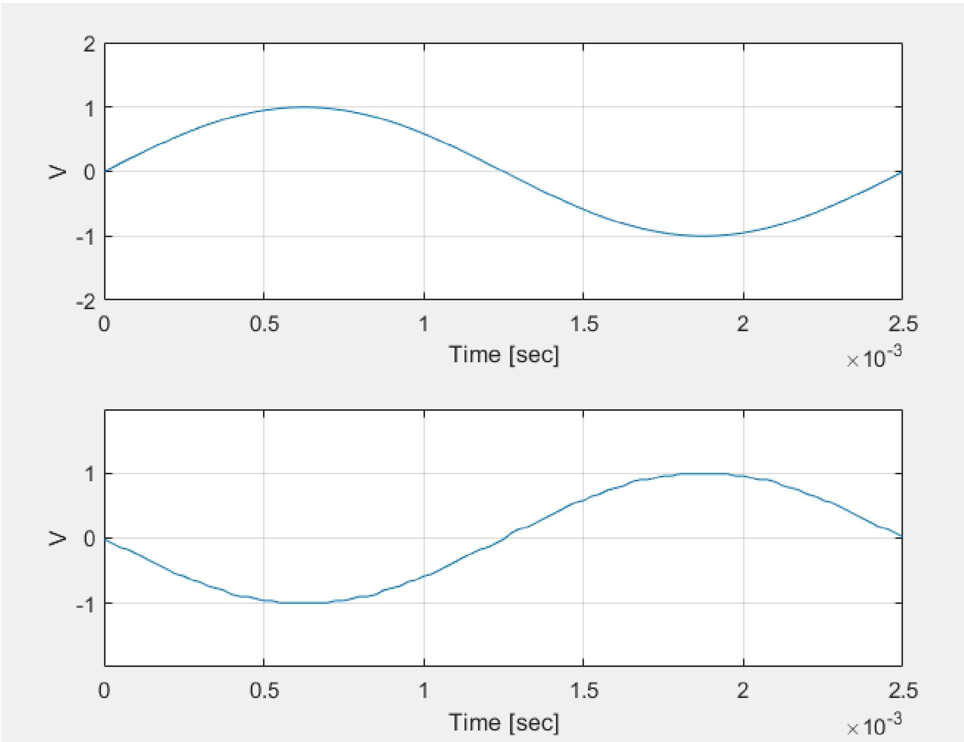
The MATLAB function *tx* represents the transmitter block as depicted in Fig 7.1. The last two parameters to the transmitter function indicate that no differential encoding is to be performed and a binary data rate of 1 kbps. Transmit `xw` over an **inverting** channel of 19,900 Hz bandwidth and noise power of 0.01 W:

```
>> y=-channel(xw,1,0.01,19000);
>>
```

Decode the channel output by matched filtering followed by detection and D/A conversion. Compare the waveforms `x` and `x_analog`:

```
>> x_digital=rx(y,'manchester');
>> x_analog=d2a(x_digital,6);

                DECODING CHANNEL OUTPUT
            -------------------------------

o PERFORMING SERIAL-TO-PARALLEL CONVERSION :
     Serial-to-parallel conversion complete.
o PERFORMING BINARY-TO-QUANTIZED-ANALOG CONVERSION :
     Binary to quantization level conversion;
         BINARY to quantization level conversion complete;
     Source decoding complete.
o PERFORMING QUANTIZED-ANALOG-TO-ANALOG CONVERSION :
>> subplot(211),waveplot(x)
>> subplot(212),waveplot(x_analog)
```

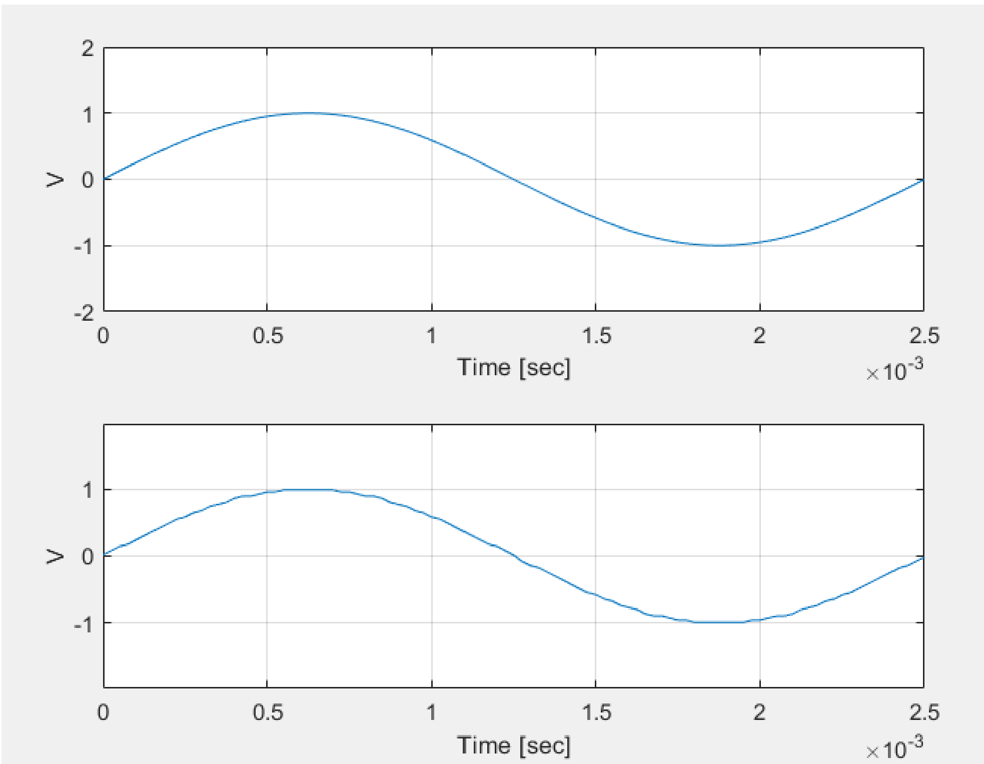
Display



C.2 Perform the above sequence of operations using differential encoding. Modify input parameters to *tx* and *rx* as shown:

```
>> u=tx(x_pcm,'manchester','diff',1000);

o PERFORMING CHANNEL CODING :
     Differential encoding;
         Differential encoding complete;
     Generating waveform in the selected format;
     Channel coding complete.
>> z=-channel(u,1,0.01,19000);
>> u_digital=rx(z,'manchester','diff');
o PERFORMING DIFFERENTIAL DECODING:
     Differential decoding complete;
>> u_analog=d2a(u_digital,6);

                DECODING CHANNEL OUTPUT
            -------------------------------

o PERFORMING SERIAL-TO-PARALLEL CONVERSION :
     Serial-to-parallel conversion complete.
o PERFORMING BINARY-TO-QUANTIZED-ANALOG CONVERSION :
     Binary to quantization level conversion;
         BINARY to quantization level conversion complete;
     Source decoding complete.
o PERFORMING QUANTIZED-ANALOG-TO-ANALOG CONVERSION :
>> subplot(211),waveplot(x)
>> subplot(212),waveplot(u_analog)
```

Display



Compare waveforms `x`, `x_analog` and `u_analog`.

> **Q7.2** Discuss whether it is more important for analog or digital signals to be protected against 180° phase reversal.

ในการทำ Match Filter จะมีการกลับสัญญาณ จึงต้องทำการกลับเฟสไว้ก่อนทำการส่ง เมื่อสัญญาณถึงผู้รับ จะได้สัญญาณเดิมจากที่ผู้ส่ง ส่งให้

D . Baseband Communication

D.1 Generate 1,000 binary samples to evaluate the bit error rate (BER) as waveforms in *unipolar NRZ* and *manchester* signalling formats with $R_b = 1$ kbps are transmitted over a baseband communication channel.

```
>> b=binary(1000);
>> Rb=1000;
>> u=tx(b,'unipolar_nrz',Rb);

o PERFORMING CHANNEL CODING :
     Generating waveform in the selected format;
     Channel coding complete.
>> m=tx(b,'manchester',Rb);

o PERFORMING CHANNEL CODING :
     Generating waveform in the selected format;
     Channel coding complete.
>>
```

Consider a low-pass communication channel with:

- channel gain = 0 dB;
- channel noise power[1], $\sigma_n^2 = 1$;
- channel bandwidth = 19 kHz;

Generate the output from this channel and estimate the transmitted binary sequence using the MATLAB function *rx*:

คำนวณหา Power (Eb) จาก command : meansq(...)

```
>> meansq(0.2*u)

ans =

    0.0195

>> meansq(0.3*u)

ans =

    0.0439

>> meansq(0.4*u)

ans =

    0.0781
```

```
>> meansq(0.5*u)

ans =

    0.1220

>> meansq(0.6*u)

ans =

    0.1757

>> meansq(0.7*u)

ans =

    0.2391
```

** u แทน ch_input ของ Unipolar NRZ และ m แทน ch_input ของ Manchester **

```
>> meansq(0.2*m)

ans =

    0.0400

>> meansq(0.3*m)

ans =

    0.0900

>> meansq(0.4*m)

ans =

    0.1600
```

```
>> meansq(0.5*m)

ans =

    0.2500

>> meansq(0.6*m)

ans =

    0.3600

>> meansq(0.7*m)

ans =

    0.4900

>>
```

A (Volt)	Power (ของ b ที่ 1000 bit)	
	Unipolar NRZ	Manchester
0.2	0.0195	0.0400
0.3	0.0439	0.0900
0.4	0.0781	0.1600
0.5	0.1220	0.2500
0.6	0.1757	0.3600
0.7	0.2391	0.4900

A (Volt)	Power per bit (/ 1000)	
	Unipolar NRZ	Manchester
0.2	0.0000195	0.0000400
0.3	0.0000439	0.0000900
0.4	0.0000781	0.0001600
0.5	0.0001220	0.0002500
0.6	0.0001757	0.0003600
0.7	0.0002391	0.0004900

A (Volt)	Unipolar NRZ		Manchester	
	E_b/N_0	P_e	E_b/N_0	P_e
0.2	0.3705	0.475000	0.7600	0.103000
0.3	0.8341	0.437000	1.7100	0.030000
0.4	1.4839	0.376000	3.0400	0.005000
0.5	2.3180	0.242000	4.7500	0.001000
0.6	3.3383	0.138000	6.8400	0.000000
0.7	4.5429	0.035000	9.3100	0.000000

Table 7.1 (Empirical Value)

$$N_0 = \frac{\sigma^2{}_n}{system\ bandwidth} = \frac{1^2}{19x10^3} = 5.263157x10^{-5}W/Hz$$

ค่า BER ที่เกิดจาก Ch_input Unipolar NRZ

```
>> ch_output=channel(0.2*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.475000.
>> ch_output=channel(0.3*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.437000.
>> ch_output=channel(0.4*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.376000.
>> ch_output=channel(0.5*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.242000.
>>
>> ch_output=channel(0.6*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.138000.
>> ch_output=channel(0.7*u,1,1,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.035000.
>>
```

ค่า BER ที่เกิดจาก Ch_input Unipolar NRZ

```
>> ch_output=channel(0.2*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.103000.
>> ch_output=channel(0.3*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.030000.
>> ch_output=channel(0.4*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.005000.
>> ch_output=channel(0.5*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.001000.
>>
>> ch_output=channel(0.6*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.000000.
>> ch_output=channel(0.7*m,1,1,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.000000.
>>
```

where `ch_input` is either the unipolar NRZ waveform `u` or the manchester waveform `m`. The value of `A` in the above command line will change the waveform amplitude and therefore the transmitter power measured in terms of E_b. Perform the BER computations for values of `A` shown in Table 7.1.

D.2 Consider the following baseband communications channel:

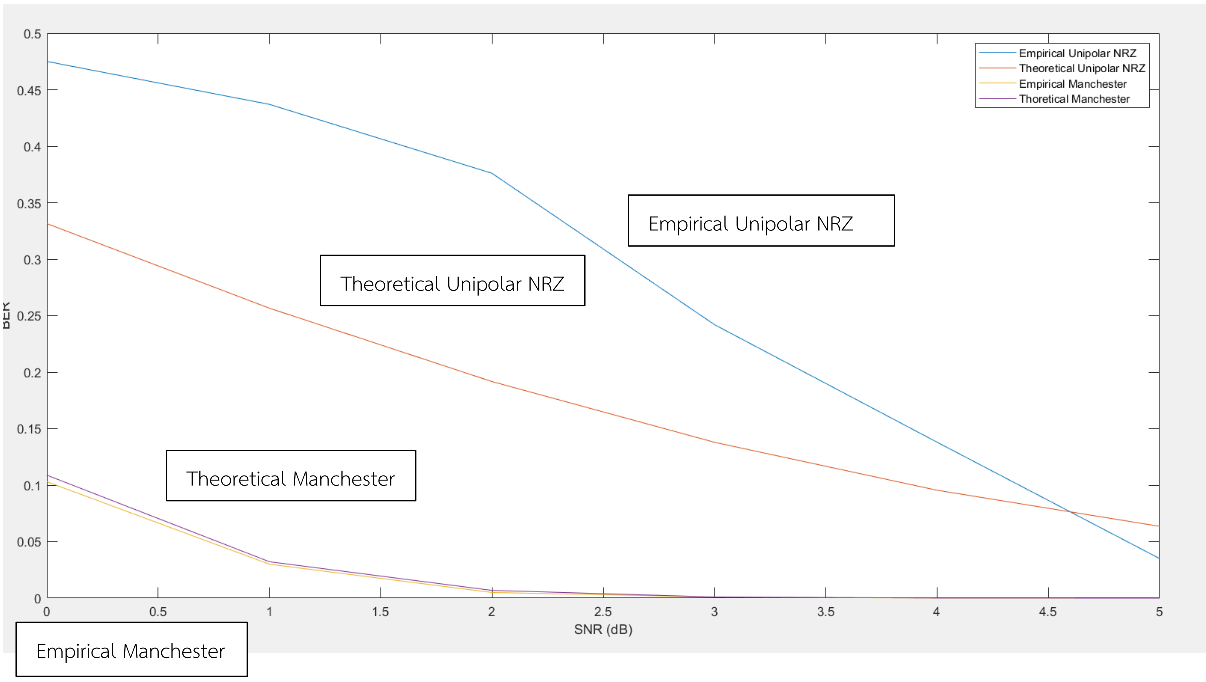
```
>> ch_output=channel(0.2*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.453000.
>> ch_output=channel(0.3*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.402000.
>> ch_output=channel(0.4*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.351000.
>> ch_output=channel(0.5*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.243000.
>> ch_output=channel(0.6*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.170000.
>> ch_output=channel(0.7*u,1,2,19000);
>> rx(ch_output,'unipolar_nrz',b);
Probability of bit error (BER) =   0.099000.
```

```
>> ch_output=channel(0.2*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.187000.
>> ch_output=channel(0.3*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.089000.
>> ch_output=channel(0.4*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.050000.
>> ch_output=channel(0.5*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.014000.
>> ch_output=channel(0.6*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.005000.
>> ch_output=channel(0.7*m,1,2,19000);
>> rx(ch_output,'manchester',b);
Probability of bit error (BER) =   0.002000.
>>
```

Theoretical Value

A (Volt)	Unipolar NRZ				Manchester			
	E_b/N_0	erfc	erfc (Table)	P_e	E_b/N_0	erfc	erfc (Table)	P_e
0.2	0.37999848	0.308220639	0.662916681	0.331458341	0.759999696	0.871779614	0.217619585	0.108809793
0.3	0.854999658	0.462330958	0.513218301	0.25660915	1.709999316	1.307669422	0.064411067	0.032205533
0.4	1.519999392	0.616441277	0.383328618	0.191664309	3.039998784	1.743559229	0.013672096	0.006836048
0.5	2.37499905	0.770551596	0.275834513	0.137917256	4.7499981	2.179449036	0.002054723	0.001027362
0.6	3.419998632	0.924661916	0.190985459	0.09549273	6.839997264	2.615338843	0.000216751	0.000108376
0.7	4.654998138	1.078772235	0.127105942	0.063552971	9.309996276	3.05122865	0.000015953	0.000000000

```
>> snr=[0 1 2 3 4 5];
>> unipolar_th=[0.331458341 0.25660915 0.191664309 0.137917256 ...]
0.09549273 0.063552971];
>> manchester_th=[0.108809793 0.032205533 0.006836048 0.001027362 ...]
0.000108376 0.000000000];
>> unipolar_em=[0.475 0.437 0.376 0.242 0.138 0.035];
>> manchester_em=[0.103 0.03 0.005 0.001 0.000 0.000];
>> plot(snr,unipolar_em),hold on;
>> plot(snr,unipolar_th),hold on;
>> plot(snr,manchester_em),hold on;
>> plot(snr,manchester_th),hold on;
>> ylabel('BER'),xlabel('SNR (dB)');
```



Determine N_0 corresponding to $\sigma_n^2 = 2$ and $R_b = 1$ kbps. If the channel input is `u`, determine from your pre-lab assignment the required transmitter power measured in terms of E_b to achieve $P_e \leq 10^{-2}$. For the calculated value of transmitter power, empirically determine BER using `u`. Repeat using the manchester encoded waveform `m`.

กำหนด

$\sigma_n^2 = 2$, $R_b = 1$ kbps , $P_e \leq 10^{-2}$, Find E_b

Unipolar NRZ

$Q\left(\sqrt{\frac{E_b}{N_0}}\right) \leq 0.01$; เปิด Appendix F

; $\sqrt{\frac{E_b}{N_0}} \geq 2.33$

$\frac{E_b}{N_0} \geq 5.4289$

$E_b \geq 5.4289\, N_0$

Manchester

$Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \leq 0.01$

$\sqrt{\frac{2E_b}{N_0}} \geq 2.33$

$E_b \geq 2.71445\, N_0$

เนื่องจาก

$$S_n(f) = \frac{N_0}{2} = \frac{\sigma_n^2}{2 \times B.W_{system}}$$

$$; \; N_0 = \frac{\sigma_n^2}{\text{system B.W.}}$$

$$= \frac{2}{19 \times 10^3}$$

$$\therefore N_0 = 105.263\ \mu W/Hz$$

จะได้ E_b (Unipolar NRZ) = 0.5714 mW

E_b (Manchester) = 0.2857 mW

E . Band-Pass Communication

E.1 Generate 100 samples from a speech signal:

```
>> s=speech(100);
>> b=a2d(s,8,'mu_law');

                    A/D CONVERSION
             --------------------------

o PERFORMING QUANTIZATION :
     Mu_law companding;
         Companding complete;
     Quantization complete.
o PERFORMING SOURCE CODING :
     Natural binary coding;
         Natural Binary coding complete;
     Source coding complete.
o PERFORMING PARALLEL-TO-SERIAL CONVERSION :
     Parallel-to-serial conversion complete.
>> xw=tx(b,'psk',100000);

o PERFORMING CHANNEL CODING :
     Generating waveform in the selected format;
     Channel coding complete.
>> y=channel(xw,0,1,[600000,1400000]);
>> s_digital=rx(y,'psk');
>> s_analog=d2a(s_digital,8,'mu_law');

                 DECODING CHANNEL OUTPUT
             -------------------------------

o PERFORMING SERIAL-TO-PARALLEL CONVERSION :
     Serial-to-parallel conversion complete.

o PERFORMING SERIAL-TO-PARALLEL CONVERSION :
     Serial-to-parallel conversion complete.
o PERFORMING BINARY-TO-QUANTIZED-ANALOG CONVERSION :
     Binary to quantization level conversion;
         BINARY to quantization level conversion complete;
     Source decoding complete.
o PERFORMING QUANTIZED-ANALOG-TO-ANALOG CONVERSION :
     Mu_law expansion;
         Expansion complete;
```

```
>> subplot(411),waveplot(s);
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 129)
>> subplot(412),waveplot(b);
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 96)
>> subplot(413),waveplot(s_digital);
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 96)
>> subplot(414),waveplot(s_analog);
Warning: AXIS('STATE') is obsolete and will be
eliminated in future versions. Use GET(GCA,...)
instead.
> In axis (line 227)
In waveplot (line 129)
>>
```