COMMUNICATION SYTEMS II
ELE 045

LABORATORY MANUAL

O.M. ZEYTINOGLU & N.W. MA

DEPARTMENT OF ELECTRICAL and COMPUTER ENGINEERING

RYERSON POLYTECHNIC UNIVERSITY

COMMUNICATION SYSTEMS II ELE 045

Laboratory Manual

O.M. Zeytinoglu & N.W. Ma

Department of Electrical and Computer Engineering
Ryerson Polytechnic University

CONTENTS ...

PREFACE ...

This manual has been written as a laboratory reference for the fourth year electrical engineering course titled ELE 045 Communication Systems II. In the laboratory, the student will encounter seven experiments which demonstrate principles of communication — with special emphasis on digital communication — systems explored in the lectures.

The study of random signals and noise is essential for evaluating performance of communication systems. Therefore, the first two experiments present the groundwork for the description of random signals by illustrating basic concepts from *probability theory* and *stochastic processes.* Other experiments demonstrate *binary signalling formats, transmission channel characteristics, detection, quantization,* and *digital modulation.* In the final experiment the student will combine all elements studied in earlier experiments to design and simulate a complete digital communication system.

All experiments described in this manual are based on computer simulations. They are performed within the MATLAB environment, which is an interactive, matrix-based system for scientific and engineering calculations. Simulations can model the behaviour of real systems with remarkable degree of precision. However, if any conclusions are to be drawn on results from simulation studies, a few words of caution are on order. First, the simulation environment may constrain which physical phenomena can be possibly modelled. Second, the simulated environment — or any other design — is built on a set of fundamental assumptions. Thus, simulation results are considered representative for the behaviour for a real system, only, if these fundamental assumptions are not violated and if the simulation capability is not exceeded. It is of utmost importance to be familiar with the inherent limitations of a simulation environment. Appendix A introduces the simulation environment used in the experiments and provides an overview of the underlying assumptions.

The MATLAB program provides the user with a rich set of functions to manipulate and process data. In addition, there are several toolboxes consisting of high level functions for specific applications, such as the *Control System Toolbox.* The experiments described in this manual use an extensive set of routines performing various tasks frequently encountered in communication systems. We call this collection of routines the *Communication System Toolbox.*

In designing the *Communication System Toolbox*, we adhered to the principles of original MATLAB functions. Each function represents a simple and general purpose tool. Combining these apparently simple functions in a typical block diagram fashion, we can simulate the dynamic behaviour of complex communication systems with extreme ease. There are other tools based on existing MATLAB routines with their behaviour modified for the present simulation environment. Yet, another set of functions are designed as top-level blocks. They coordinate data flow between several subtasks to achieve the desired input-to-output mapping representing a particular system function. As the student gains an understanding of how the individual subtasks are performed, the use of top-level functions greatly simplify the command syntax. In most cases, only a single call has to be made to a top-level function which will replace several lines of code. Such a building block approach becomes a useful tool in analyzing complex systems. For example the function *a2d* represents the *analog-to-digital* conversion process. It consists of lower level routines that perform *companding, quantization* and *source coding.* Finally, there are special purpose functions which simplify specific tasks in experiments, as such their intended use is restricted to this course only. A complete list of functions in the *Communication System Toolbox* is given in Appendix B.

In designing these experiments our intention has been to provide the students with easy-to-use tools to study complex concepts. However, due to limited number of hours that can be formally devoted to laboratory sessions, not all functions in the *Communication System Toolbox* are used in the experiments. We encourage the student to use these functions in all aspects of the course. Feel free to experiment with these tools; extend the functionality, improve them or write your own functions to create your very own toolbox. Share your creative ideas with others. Remember that learning is an interactive process, and it is more gratifying if it is done collectively.

Toronto, Ontario
July, 1991

O. M. Z. and N. W. M.

INTRODUCTION

EVALUATION OF LABORATORY WORK ...

Laboratory Manual

All results are to be accumulated in this laboratory manual. At the end of the course you will submit your manual for grading. For each experiment, tables and graphs are provided to enter all results and observations. All data will be entered in a clear fashion since one of the criteria for judgement will be *clarity,* hence usefulness of the information contained in the manual. If the pre- and post-lab work is kept within this manual you will accumulate a comprehensive study guide which greatly enhances usefulness of the laboratory work.

You may also be asked to write a formal report on one of the experiments. A formal report will be evaluated on the basis of clarity, on the completeness of tabulated, graphed and/or calculated data, and most importantly on the conclusions drawn from the experiment.

Pre-Lab Assignment

For each experiment, there will be a pre-lab assignment consisting of problems listed under the *pre-lab* heading. The pre-lab assignment is to be done by all students individually and to be handed in as you come into the laboratory.

Post-Lab Assignment

A question that appears in a box indicates a problem that must be answered in the post-lab assignment for that particular experiment. These questions must be answered in writing and submitted to your instructor when you come into the laboratory for the next experiment.

Post-lab questions are placed at the end of relevant sections in the experiment procedure. Therefore, you will have a chance to read the question to ensure that you have performed all necessary steps and that you have not missed taking some important data. It is also recommended if you devote a few minutes at the end of each session to outline answers to these problems in the notes pages of your laboratory manual. This will allow you to clarify ambiguous points by consulting your instructor.

General

Unless justifiable cause for absence is shown, you are expected to be present and on time at each laboratory session.

No late pre- or post-lab assignments are to be accepted. If you have difficulty in answering any question it is your responsibility to consult your instructor before the due date of your assignment.

HOW TO USE THIS MANUAL ...

This manual is composed of the three parts. The first part is the Introduction detailing guidelines to be followed in the laboratory. The second part includes procedures of all experiments, as well as the pre- and post-lab questions. The final part contains appendices with supplementary material. Appendix A is a recommended reading for every student who wants to have a better understanding of the simulation environment as it provides an overview of the underlying assumptions. Appendix B gives a listing and a brief description of the MATLAB functions in the *Communication System Toolbox.*

The experiments described in this manual do not require any previous knowledge of the MATLAB program. All procedures are self contained in the sense that any student following the instructions should be able to complete the experiment with ease. Experience with MATLAB will come in handy if the student wants to experiment beyond the procedure described in this manual. The on-line help facility will usually suffice in a classroom setting.

The procedure of each experiment is preceded by a set of questions under the *pre-lab* heading. These problems are drawn from the course material relevant to the particular experiment. Their intended purpose is to prepare the student for the concepts to be studied in greater detail during the experiment. It is therefore of utmost importance that these problems are answered before coming into the laboratory.

TYPOGRAPHIC CONVENTIONS ...

This manual uses a number of typographic conventions:

- *Italic font* is used to indicate MATLAB functions when they are referred to in the text. It is also used to give special emphasis. For example:

  > Using the MATLAB functions *gaus_pdf* and *gaus_cdf* display the pdf and the cdf of the random variable.

- The $\gg$ symbol is indicates the MATLAB prompt that you will see on your terminal after starting the MATLAB program.

- `This font` is used within text to show characters or words that you type. Variables obtained from issuing MATLAB commands are represented in this font. It also indicates messages or prompts from the system that appear on your screen. For example:

  > Compute the mean and variance of the random sequence
  > $\gg$ `mean_abc = mean(abc);   var_abc = var(abc);`
  > Use `mean_abc` and `var_abc` to compute mean-square power.

- A capitalized word within text indicates a key that you press. For example:

  > If the second argument to the function *eye_diag* is negative, hit the Return key for the next trace to be displayed.

- A question that appears in a box indicates a post-lab assignment. It must be answered in writing and submitted when you come into the laboratory for the next experiment. See also the section on laboratory work evaluation. For example:

Q1.2 If the sample pdf is based on 2000 experiments, estimate the number of occurrence for each face value.

EXPERIMENTS

IMPORTANT REMARK

A number of global MATLAB variables are essential for the functions in the *Communication System Toolbox* to perform and to communicate properly.

The MATLAB function *start* initializes theses variables to their default values for each experiment. Whenever you enter the MATLAB workspace, your very first command line must be:

```
≫  start
```

You will be then greeted by a welcome screen and by other informative messages. **Please make sure that you initialize each MATLAB session with the *start* command.**

EXPERIMENT 1
PROBABILITY THEORY

OBJECTIVES

In this experiment you will study:

- properties of discrete and continuous random variables;
- uniform and Gaussian random variables in terms of their probability density and cumulative distribution functions;
- statistical measures of mean, variance and mean-square power.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 118–147.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 660–684.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 404–430.

PRE-LAB ASSIGNMENT

1. Consider a 4-faced die, with the face values numbered 1 through 4.

a. Determine the sample space Ω for the experiment where the die is rolled once.

b. In another experiment the die is rolled 2000 times and the following outcomes were observed:

Face Value	1	2	3	4
Number of Occurrences	112	784	865	239

Let $\mathbf{X}$ be a random variable representing the face value that is observed as the die is rolled. Use the relative frequency definition of probability to estimate $P(\mathbf{X} = k),\ \ k = 1, \ldots, 4$.

c. Based on the probabilities determined in part b, compute and plot the probability density function and the cumulative distribution function of $\mathbf{X}$.

d. Determine $P(1 < \mathbf{X} < 5)$ and $P(\mathbf{X} \geq 5)$.

e. Determine $P(\mathbf{X} = k)$, $k = 1, \ldots, 4$, if the die were a "fair" die.

2. Let $\mathbf{U} \sim \mathcal{U}(a, b)$ be a uniformly distributed random variable defined over the interval $[\,a,\ b\,]$ with $a < b$.

a. Determine and plot the probability density function of $\mathbf{U}$.

b. Determine and plot the cumulative distribution function of $\mathbf{U}$.

c. Determine $\mathrm{E}[\,\mathbf{U}\,]$, $\mathrm{E}[\,\mathbf{U}^2\,]$, and $\sigma^2_{\mathbf{U}}$ as a function of a and b.

d. For $a = 2$ and $b = 6$, evaluate $P(-2 < \mathbf{U} < 4)$ and $P(\mathbf{U} = 5)$.

3. Consider a normal random variable $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$ for some μ and σ. Let $\boldsymbol{\Phi}(\mathbf{z})$ be the cumulative distribution function of the standard normal random variable $\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, whose tabulated values are available.

a. Express $P(a < \mathbf{X} < b)$ and $P(\mathbf{X} > b)$ in terms of Φ, a, b, μ and σ.

b. For $\mu = 4$, $\sigma^2 = 4/3$, determine $P(3 \leq \mathbf{X} \leq 5)$ and $P(\mathbf{X} \geq 5)$.

PROCEDURE

A. Discrete Random Variables

A.1 Using the function *dice*, generate random samples representing the outcome of the experiment where a 4-faced, fair die is rolled 2,000 times:

```
≫  x = dice( 2000, 4, 'fair');
```

A.2 Compute and display the sample probability density function (pdf) and cumulative distribution function (cdf) of the sequence `x`:

```
≫  subplot(121), pdf(x)
≫  subplot(122), cdf(x)
```

Sample events representing the outcome of an experiment where this die is rolled once, are of the form $\{\textit{Face value} = k, \;\; k = 1, \ldots, 4\}$. Observe the relation between the pdf and the cdf that convey information about the sample event probabilities in different ways.

Q1.1 Why are probabilities of the sample events in this experiment not equal? Is this result expected? What should be done in step A.1, to improve this outcome?

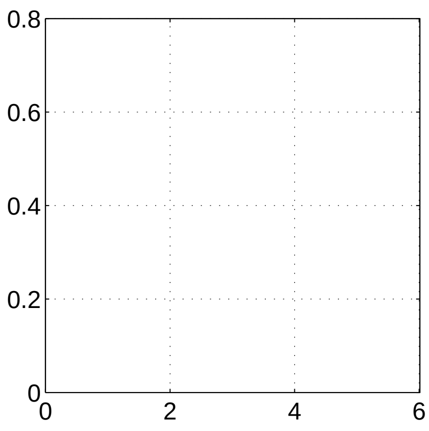
A.3 Using the function *dice* generate a sequence representing the outcome of an experiment where a 4-faced, biased die is rolled 2,000 times:

```
≫  y = dice( 2000, 4, 'biased');
```

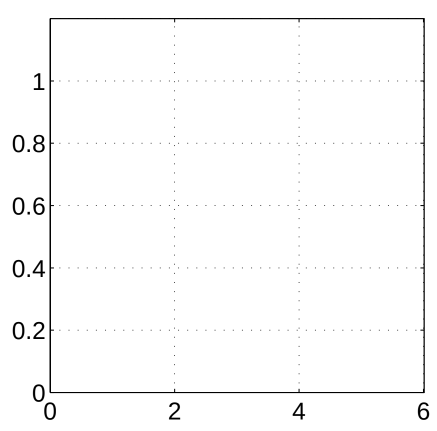
A.4 Clear the graphics screen and display the sample pdf of the random sequence `y`:

```
≫  clf
≫  pdf(y)
```

Plot the sample pdf in the space provided for Graph 1.1. Compute the sample cdf and show your result in Graph 1.2. Indicate which points belong to the cdf at the jump points. You can verify your answer using the MATLAB function *cdf*.



Graph 1.1



Graph 1.2

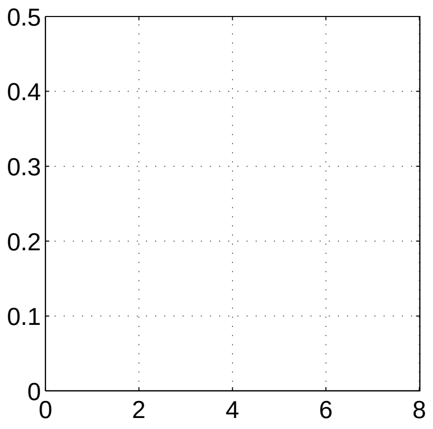
Q1.2 Given that the sample pdf is based on 2,000 experiments, estimate the number of occurrence for each face value.

B. Continuous Random Variables

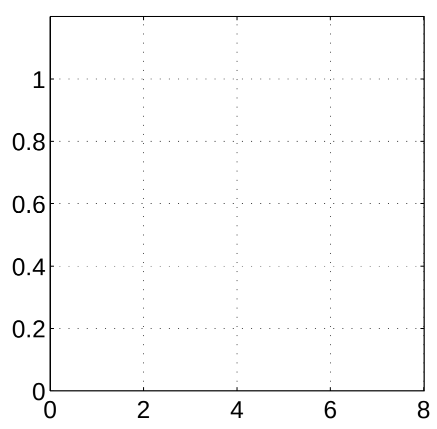
Uniform Random Variable

B.1 Using the functions *unif_pdf* and *unif_cdf* display the pdf and cdf of the uniform random variable $\mathcal{U}(2,6)$. Sketch the pdf and the cdf:

```
≫  subplot(121), unif_pdf(2,6), axis([0,8,-0.2,1.2]);
≫  subplot(122), unif_cdf(2,6), axis([0,8,-0.2,1.2]);
```



Graph 1.3



Graph 1.4

B.2 If $\mathbf{U} \sim \mathcal{U}(2,6)$ determine the following probabilities using either the pdf or the cdf.

$P(\ 0 < \mathbf{U} \leq 3\ )$	$P(\ 3 < \mathbf{U} \leq 5\ )$	$P(\ \mathbf{U} = 3\ )$

Table 1.1

Q1.3 Why is $P(\mathbf{U} = 3)$ different than $P(\mathbf{X} = 3)$ in step A.4?

B.3 Let's play statistics. Generate 500 samples from a $\mathcal{U}(2,6)$ distribution:

```
≫  u = uniform(2,6,500);
```

Compute the mean and the variance of the random sequence `u`:

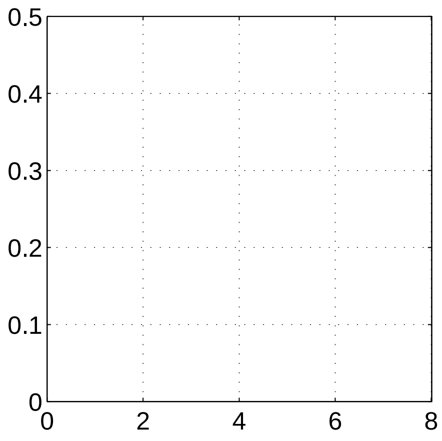
```
≫  mean_u = mean(u),    var_u = var(u)
```

Compare these results with answers from pre-lab Question 2. Comment on any difference. Can you predict the algorithms in MATLAB functions *mean* and *var*? Display their contents by using the MATLAB command *type*. Use `mean_u` and `var_u` to determine $\mathrm{E}[\,\mathbf{U}^2\,]$. Check your result with the MATLAB function *meansq*.

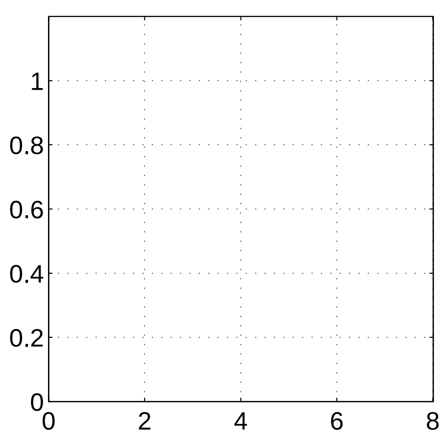
Gaussian (normal) Random Variable

B.4 Using the MATLAB functions *gaus_pdf* and *gaus_cdf* display the pdf and the cdf of the random variable $\mathbf{G} \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu =$ `mean_u`, and $\sigma^2 =$ `var_u` from step B.3. Sketch the pdf and the cdf.

```
≫  clg, subplot(121), gaus_pdf(mean_u,var_u)
≫  subplot(122), gaus_cdf(mean_u,var_u)
```



Graph 1.5



Graph 1.6

Indicate the horizontal axis values where the pdf is at its maximum and the cdf equals to 0.5. Compare these values with the mean value of the Gaussian distribution.

B.5 Determine the following probabilities:

$P(\ 0 < \mathbf{G} \leq 3\ )$	$P(\ 3 < \mathbf{G} \leq 5\ )$	$P(\ \mathbf{G} \geq 5\ )$

Table 1.2

Compare these results with those in Table 1.1. The Gaussian distribution used to generate these results and the uniform distribution in step B.2 have the same mean and variance; yet, the results are different. Can you explain?

B.6 Let $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$. Assume the mean value of $\mathbf{X}$ is fixed at $\mu = 1$ and $\sigma^2 \in \{0.5, 1, 2, 5, 10\}$. To observe effects of changing the value of σ^2 on the distribution of a Gaussian random variable, perform the following steps:

```
≫  clf
≫  m = 1; gaus_pdf(m,0.5)
≫  axis( [-10 10 0 0.6] ), hold on
≫  gaus_pdf(m,1)
     ⋮
≫  gaus_pdf(m,10)
```

Q1.4 Consider the event $\mathcal{A} = \{0 < \mathbf{X} < 2\}$ where $\mathbf{X} \sim \mathcal{N}(1, \sigma^2)$ with $\sigma^2 \in \{0.5, 1, 2, 5, 10\}$. Determine the value of σ^2 for which $P(\mathcal{A})$ is a maximum.

Now consider changes on the cdf:

```
≫  clf
≫  m = 1; gaus_cdf(m,10)
≫  axis( [-10 10 0 1] ), hold on
≫  gaus_cdf(m,5)
    ⋮
≫  gaus_cdf(m,0.5)
```

Can you predict the cdf as σ^2 becomes smaller? Try:

```
≫  gaus_cdf(m,0.00001)
```

What does it mean to have a probability distribution with very small variance? The corresponding pdf may help to illustrate this point:

```
≫  clf
≫  gaus_pdf(m,0.00001)
≫  axis( [0 2 0 200] )
```

Where in this experiment did you observe a similar pdf? If in the last step you had a uniformly distributed random variable with mean μ and variance 0.00001, instead of a Gaussian random variable, would the resulting pdf be different?

B.7 Now fix the variance of the Gaussian distribution at σ^2 and change the mean value μ such that $\mu \in \{-4, -1, 2, 5\}$.

```
≫  clf
≫  s = 1; gaus_pdf(-4,s)
≫  axis( [-8 8 0 0.5] ), hold on
≫  gaus_pdf(-1,s)
    ⋮
≫  gaus_pdf(5,s)
```

What is the effect of changing the mean value μ? Let $\mathbf{X}(\mu, \sigma^2)$ represent a random variable such that $\mathbf{X}(\mu, \sigma^2) \sim \mathcal{N}(\mu, \sigma^2)$. Compare $P(-5 < \mathbf{X}(-4, 1) < -3)$, and $P(4 < \mathbf{X}(5, 1) < 6)$. You may wish to make some observations on the effects of changing μ on the cdf. First clear the graphics screen and then for the above set of μ and σ^2 try the axis-setting $[-8, 8, 0, 1]$.

C. Mean, Variance and Power

C.1 Generate the following random sequences with different mean values:

```
≫  x = gauss(-5,1,100);
≫  y = gauss( 0,1,100);
≫  z = gauss( 5,1,100);
≫  clf
≫  plot(x)
≫  axis( [1 100 -10 10] ), grid on, hold on
≫  plot(y)
≫  plot(z)
```

Using engineering terminology, you can say that mean value changes the *dc* level of the waveform[1].

C.2 Generate random sequences from Gaussian distributions with different variance values:

```
≫  a = gauss(0, 4,100);
≫  b = gauss(0, 1,100);
≫  c = gauss(0, 0.5,100);
≫  d = gauss(0,0.01,100);
≫  clf
≫  subplot(221), plot(a), axis( [1 100 -10 10] )
≫  subplot(222), plot(b), axis( [1 100 -10 10] )
≫  subplot(223), plot(c), axis( [1 100 -10 10] )
≫  subplot(224), plot(d), axis( [1 100 -10 10] )
```

Using MATLAB functions *mean* and *var* determine the mean and variance of each sequence and enter your answers in Table 1.3.

[1] Displayed waveforms x, y, z are not identical as each call to the MATLAB function *gauss* generates a new random sequence. If you want to use the same random sequence each time you make a call to the function *gauss*, use it in the form `gauss(mean_value,variance,n,seed)` and provide the same seed value.

Sequence	**Mean**	**Variance**	**Mean Sq.**
a			
b			
c			
d			

Table 1.3

Determine the mean-square power of each waveform using the respective mean and variance values. Check your results using the MATLAB function *meansq*.

Q1.5 If the waveforms displayed in this part were to represent a noise signal, which one would you prefer to encounter in your communications system? If they were representing noise-free signal waveforms, which one would you prefer?

D. Dart Game

D.1 To illustrate the significance of mean and variance, use the MATLAB function *dart*, which simulates an experiment where a dart is thrown at a target at the origin of the polar plane. The user can specify the mean and variance of x and y coordinates which in turn determine the point of impact on the dart board. A good point to start experimenting is:

```
≫  mean_x = 0.2;   mean_y = 0.2;
≫  var_x = 0.1;   var_y = 0.1;
≫  no_dart = 20;
≫  clf
≫  dart( [mean_x mean_y], [var_x var_y], no_dart )
```

If you want to have more information about this simulation use the on-line help facility by typing `help dart`. Feel free to experiment with various mean and variance values. In particular, it is interesting to look at simulation results as one or both variances are set to zero. Try to correlate your observations with other parts of this experiment.

Q1.6 Consider two players with performance statistics given by:

$$\textbf{Player 1}: [\mu_x,\ \mu_y] = [0,\ 0], \quad [\sigma_x^2,\ \sigma_y^2] = [0.5,\ 0.5];$$
$$\textbf{Player 2}: [\mu_x,\ \mu_y] = [0.5,\ 0.5], \quad [\sigma_x^2,\ \sigma_y^2] = [0.01,\ 0.01];$$

What can you say about the skills of these players?

E. Integration by Monte-Carlo Simulation

In this part of the experiment you will demonstrate the use of relative frequency concept to integrate a function of one variable. Consider the unit square in the x-y plane and the function $f(x) = x$.

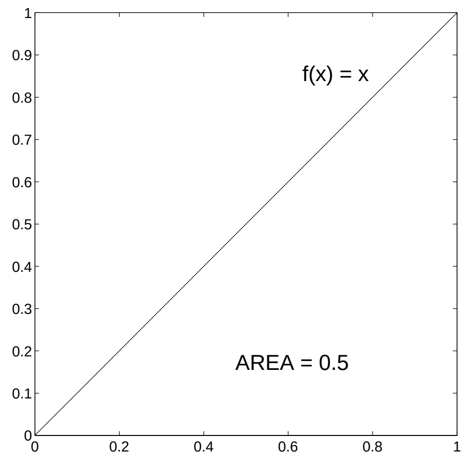


Fig. 1.7

To compute $\int_0^1 f(x)\,dx$, the following algorithm can be used:

1. Generate N samples from $\mathcal{U}(0,1)$ distribution: X-sequence.
2. Generate N samples from $\mathcal{U}(0,1)$ distribution: Y-sequence.
3. Combine X and Y sequences to form N random points on the unit square, each represented by a (X,Y) pair.
4. For each (X,Y) point determine its position with respect to the curve $y = x$, i.e. is it above or below $y = x$?
5. If the number of points below the line $y = x$ is K, estimate the value of the integral by K/N.

Use the MATLAB function *integral* to implement this algorithm with 20,000 sample points:

```
≫  integral('x', [0,1], 20000)
```

The same algorithm can be used to integrate an arbitrary function $f(x)$ over the interval $[\,a,\, b\,]$. The end points $(a,\ f(a))$ and $(b,\ f(b))$ determine the parameters of the uniform distributions from which the X and Y sequences are generated. Thus, the algorithm describes a simple method of evaluating the integral of a function. Use the MATLAB function *integral* in the form:

```
≫  integral('f(x)', [x_min,x_max], N)
```

to evaluate the integral of some sample functions. The above command displays a menu of functions from which you can make a selection. One of the selections in the menu is the pdf of the standard normal random variable. You should recall that there is no closed form solution for the integral of this pdf over an arbitrary interval — otherwise, you would not have to resort to a table of Φ values.

E.1 Integrate the pdf of the standard normal random variable numerically over the interval $[-4, 4]$:

```
≫  integral('f(x)', [-4,4], 20000)
```

When the menu is displayed, select the function representing $f_{\mathbf{X}}(x)$, the pdf of the standard random variable. What does this answer confirm about pdf's? If you were to integrate $f_{\mathbf{X}}(x)$ over $(-\infty, \infty)$ what should be the answer?

E.2 Integrate $f_{\mathbf{X}}(x)$ over the interval $[0, 1]$ and confirm the answer using a Φ-function table[1].

E.3 You can also integrate "any" function of your choice. for example to integrate $\cosh(x)$ over the interval $[-\pi, \pi]$ enter:

```
≫  integral('cosh(x)', [-pi,pi], 20000);
```

While this method may not be very accurate with only 20,000 points, it is robust. The algorithm can also be easily extended to compute the n-dimensional volume of a closed surface of the form $f(x_1, \ldots, x_n) = K$.

[1] Each time you use the function *integral*, you will receive a slightly different answer as the function uses a new set of N random points. For improved accuracy average these results.

EXPERIMENT 2
RANDOM PROCESSES

OBJECTIVES

In this experiment you will work with sample functions that will highlight basic concepts from the theory of random processes. In particular you will:

- study wide-sense stationarity and ergodicity;
- examine correlation as a measure of relation between random processes;
- investigate the relationship between a random process, its autocorrelation function and the power spectral density function.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 152-180.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 447-477.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 434-464.

PRE-LAB ASSIGNMENT

1. Let the random process $\mathbf{X}(\mathbf{\Theta}, t) = \cos(2\pi 1000t + \mathbf{\Theta})$ be defined in terms of the discrete-valued random variable $\mathbf{\Theta}$ such that $\Theta_1 = 0$, $\Theta_2 = \pi/2$, $\Theta_3 = \pi$ and $\Theta_4 = 3\pi/2$ with equal probability.

a. Determine the mean value $\mathrm{E}[\,\mathbf{X}(\mathbf{\Theta}, t)\,]$, the mean-square value $\mathrm{E}[\,\mathbf{X}^2(\mathbf{\Theta}, t)\,]$, and the autocorrelation function $R_{\mathbf{X}}(t+\tau, \tau)$.

b. Determine the time average values $\langle \mathbf{X}(\mathbf{\Theta}, t)\rangle$ and $\langle \mathbf{X}^2(\mathbf{\Theta}, t)\rangle$.

c. Is $\mathbf{X}(\mathbf{\Theta}, t)$ wide-sense stationary? Is it ergodic?

2. Determine and sketch the power spectral density of the random process $\mathbf{Y}(t)$, if the autocorrelation function $R_{\mathbf{Y}}(t)$ is given as:

a. $R_{\mathbf{Y}}(\tau) = \delta(\tau)$;

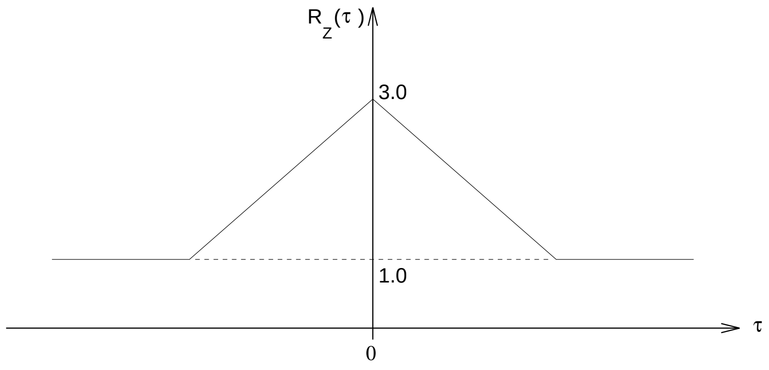
b. $R_{\mathbf{Y}}(\tau) = \Lambda(\tau)$;

c. $R_{\mathbf{Y}}(\tau) = \Lambda(\tau/T)$, where $T = 10$ msec;

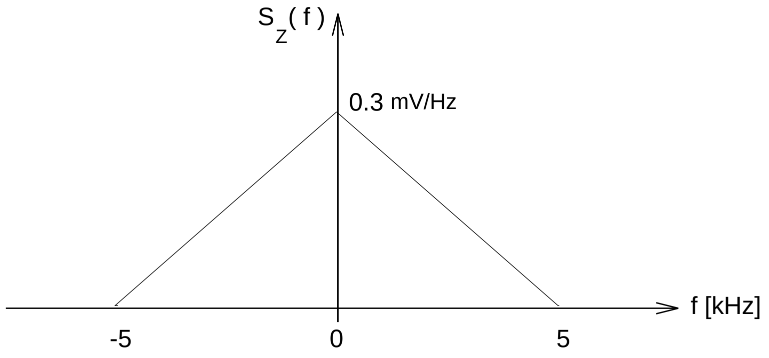
d. $R_{\mathbf{Y}}(\tau) = 1$.

3. Determine the mean and the mean-square value of the random process $\mathbf{Z}(t)$, if:

a. the autocorrelation function $R_{\mathbf{Z}}(\tau)$ is given as:



b. the power spectral density function $S_{\mathbf{Z}}(f)$ is given as:



4. Determine the output of a filter with frequency response

$$H(f) = \frac{f_0}{f_0 + j2\pi f},$$

if the input is a white noise process with power spectral density function $S_n(f) = N_0/2$. How is the power spectral density function of the output related to the filter frequency response $H(f)$?

PROCEDURE

A. Wide-Sense Stationarity and Ergodicity

A.1 Generate all 4 realizations — sample functions — of the random process $\mathbf{X}(\boldsymbol{\Theta}, t)$ described in Question 1 of the pre-lab assignment. Display the first 400 samples of each realization:

```
≫  x = realize( [0 pi/2 pi 3*pi/2] );
≫  subplot(221), waveplot( x(1,1:400) );
≫  subplot(222), waveplot( x(2,1:400) );
≫  subplot(223), waveplot( x(3,1:400) );
≫  subplot(224), waveplot( x(4,1:400) );
```

A.2 Record the value of each sample function of $\mathbf{X}(\boldsymbol{\Theta}, t)$ for $t = 0$, 0.5, 1.25, 2.2, 3.4 msec. For each t, evaluate the ensemble mean and the ensemble mean-square value.

	t [msec]				
	0.0	0.5	1.25	2.2	3.4
$\mathbf{X}(\Theta_1, t)$					
$\mathbf{X}(\Theta_2, t)$					
$\mathbf{X}(\Theta_3, t)$					
$\mathbf{X}(\Theta_4, t)$					
$\mathrm{E}[\mathbf{X}(\boldsymbol{\Theta}, t)]$					
$\mathrm{E}[\mathbf{X}^2(\boldsymbol{\Theta}, t)]$					

Table 2.1

A.3 Using the MATLAB function *ecorr* determine the autocorrelation function $R_{\mathbf{X}}(t_1, t_2)$, for the values of t_1 and t_2 shown in Table 2.2.

```
≫  ecorr( x, t1, t2 )
```

Observe that t_1 and t_2 are in milliseconds.

t_1 \ t_2	0.0	0.7	1.9	2.6
0.0				
0.7				
1.9				
2.6				

Table 2.2

Q2.1 Explain the symmetry in Table 2.2. Why are $R_{\mathbf{X}}(1.9, 2.6)$ and $R_{\mathbf{X}}(0.7, 0)$ equal? How can you evaluate $R_{\mathbf{X}}(t_1, t_2)$ from the plots observed in part A.1?

A.4 Determine the time average values $\langle \mathbf{X}(\mathbf{\Theta}, t)\rangle$ and $\langle \mathbf{X}^2(\mathbf{\Theta}, t)\rangle$ for each value of $\mathbf{\Theta}$. Record the results in Table 2.3.

```
≫  [ mean(x(1,:)) meansq(x(1,:)) ]
≫  [ mean(x(2,:)) meansq(x(2,:)) ]
≫  [ mean(x(3,:)) meansq(x(3,:)) ]
≫  [ mean(x(4,:)) meansq(x(4,:)) ]
```

$\mathbf{\Theta}$	$\langle \mathbf{X}(\mathbf{\Theta}, t)\rangle$	$\langle \mathbf{X}^2(\mathbf{\Theta}, t)\rangle$
Θ_1		
Θ_2		
Θ_3		
Θ_4		

Table 2.3

Compare the time average values from Table 2.3 with the ensemble average values shown in Table 2.1. What does this comparison say about the ergodicity of $\mathbf{X}(\mathbf{\Theta}, t)$?

A.5 Display the time-average autocorrelation function of $\mathbf{X}(\Theta_1, t)$.

```
≫  clf
≫  acf(x(1,:),100);
```

Is the time-average autocorrelation function of $\mathbf{X}(\Theta_1, t)$ different from that of $\mathbf{X}(\Theta_3, t)$?

Q2.2 If $\mathbf{X}(\mathbf{\Theta}, t)$ is an ergodic process, how would you evaluate $R_{\mathbf{X}}(2.6, 0.7)$ using the plot from part A.5?

A.6 Generate a second random process $\mathbf{Y}(\mathbf{\Phi}, t) = \cos(2\pi 1000t + \mathbf{\Phi})$ defined in terms of a discrete-valued phase variable $\mathbf{\Phi}$ such that $\Phi_1 = 0$ and $\Phi_1 = \pi$ with equal probability.

```
≫  y = realize( [0, pi ] );
```

Display both realizations of $\mathbf{Y}(\mathbf{\Phi}, t)$. Determine the ensemble mean and the ensemble mean-square value at $t = 1$ msec and $t = 1.25$ msec.

```
≫  subplot(211), waveplot( y(1,1:400))
≫  subplot(212), waveplot( y(2,1:400))
```

Q2.3 Is $\mathbf{Y}(\mathbf{\Phi}, t)$ wide-sense stationary? Is $\mathbf{Y}(\mathbf{\Phi}, t)$ an ergodic process? Prove your answers.

B . Correlation

Correlation measures the relationship between two sets of data in the sense that the closer the data sets are related, the larger will be the absolute value of the correlation measure. One of the applications of correlation is to estimate a random quantity from observations made on a related parameter.

B.1 Generate 1,000 records, each describing the age, the height and the weight of a person between the ages of 20 and 50.

```
≫  a = person_data(1000);
```

B.2 Using the data array `a`, generate scatter diagrams that display *weight vs. height*, *age vs. weight*, and *age vs. height*.

```
≫ subplot(131), stat_plot(a,'weight','height');
≫ subplot(132), stat_plot(a,'age','height');
≫ subplot(133), stat_plot(a,'age','weight');
```

B.3 Statistical relationship among age, height, and weight parameters can be studied by means of these scatter diagrams. To further investigate this relationship, play a game using the MATLAB function *guess*:

```
≫ guess('weight','height');
```

The function *guess* will display the weight of a person and will prompt you to guess the height. Try 10 rounds of guessing. Use the *weight vs. height* diagram to assist you.

B.4 Play a second guessing game in which you are given the age of a person and you are asked to guess the height.

```
≫ guess('age','height');
```

Use the *age vs. height* diagram to assist your guess.

Q2.4 Among the *age*, *height* and *weight* parameters which two have the largest correlation? Try to answer this question based on your scores in each guessing game. Which scatter diagram assisted you most?

C . Autocorrelation Function

C.1 Autocorrelation is the correlation measure of samples from the same random process. To study this concept, consider the random process $\mathbf{T}(t)$ that represents average day-time temperatures between June 1 and August 31.

```
≫ clf
≫ subplot(211), temperature
```

From the temperature of a given day, predict the temperature of the next day. Check the actual value and observe the degree of accuracy of your prediction. Try few more times!

C.2 Repeat the above step for a discrete valued, white Gaussian process $\mathbf{B}(t)$ that represents the number of new born babies in the City of Toronto between June 1 and August 31.

```
≫  subplot(212), new_born
```

Q2.5 Which random process has higher correlation between its adjacent samples?

C.3 Display the temperature and new-born waveforms, and the *normalized* autocorrelation functions of $\mathbf{T}(t) - \mathrm{E}[\,\mathbf{T}(t)\,]$ and $\mathbf{B}(t) - \mathrm{E}[\,\mathbf{B}(t)\,]$.

```
≫  clf, subplot(211), temperature(0)
≫  figure(2), subplot(211), new_born(0)
```

Q2.6 Which waveform fluctuates more rapidly? Relate your answer to the results observed in parts C.1 and C.2.

C.4 $R(3)$ quantifies correlation between samples of a random process that are separated by three "time-units". Measure $R(3)$ for temperature and new-born processes and relate these values to your answer in Question Q2.6.

$R_{\mathbf{T}}(3) =$

$R_{\mathbf{B}}(3) =$

Q2.7 If you were asked to quantify the correlation between the day-time temperatures for the days June 10 and June 28, which $R_{\mathbf{T}}$ value would you use? Explain why the autocorrelation function $R_{\mathbf{T}}(k)$ is a decreasing function of k, $k > 0$.

D . Autocorrelation Function and Power Spectral Density Function

D.1 Generate 4,096 samples from a correlated random process $\mathbf{Z}(t)$ and display the resulting waveform:

```
≫ close(2), clf
≫ z = corr_seq(0.85,4096,3,0);
≫ waveplot(z)
```

D.2 Compute and display the autocorrelation function $R_{\mathbf{Z}}(\tau)$ and the power spectral density function $S_{\mathbf{Z}}(f)$ of the random process $\mathbf{Z}(t)$.

```
≫ clf, subplot(211), acf(z)
≫ subplot(212), psd(z)
```

Compare $S_{\mathbf{Z}}(f)$ with your answer to Question 2 in the pre-lab assignment.

D.3 Evaluate the mean and mean-square value of $\mathbf{Z}(t)$ using $R_{\mathbf{Z}}(\tau)$:

$$\mathrm{E}[\,\mathbf{Z}(t)\,] =$$
$$\mathrm{E}[\,\mathbf{Z}^2(t)\,] =$$

D.4 Also determine the mean-square value of $Z(t)$ using $S_{\mathbf{Z}}(f)$. The mean-square value is equivalent to the area under the PSD function. Due to sampling of waveforms in the Matlab simulation environment, the mean-square value is equal to the area under the PSD function multiplied by a scaling factor. The scaling factor is the number of data samples divided by the sampling frequency (see Appendix A). In the present experiment, the value of the scaling factor is 0.04096.

$$\mathrm{E}[\,\mathbf{Z}^2(t)\,] = \qquad\qquad \times \quad 0.04096 \; =$$

E . White Noise

The PSD function of a white noise process is flat over the entire system bandwidth. A white noise process, used as an input, will excite all frequency modes of an *"unknown"* system. Therefore, calibrated white noise is a useful tool in system identification. For example, to determine the frequency response of an unknown system, a set-up such as shown in Fig. 2.1 can be used.

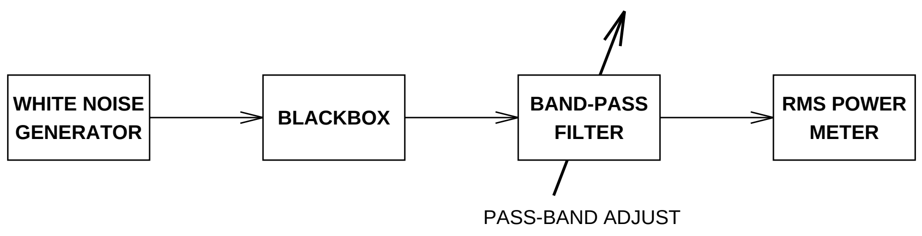


Fig 2.1 System measurements using white noise

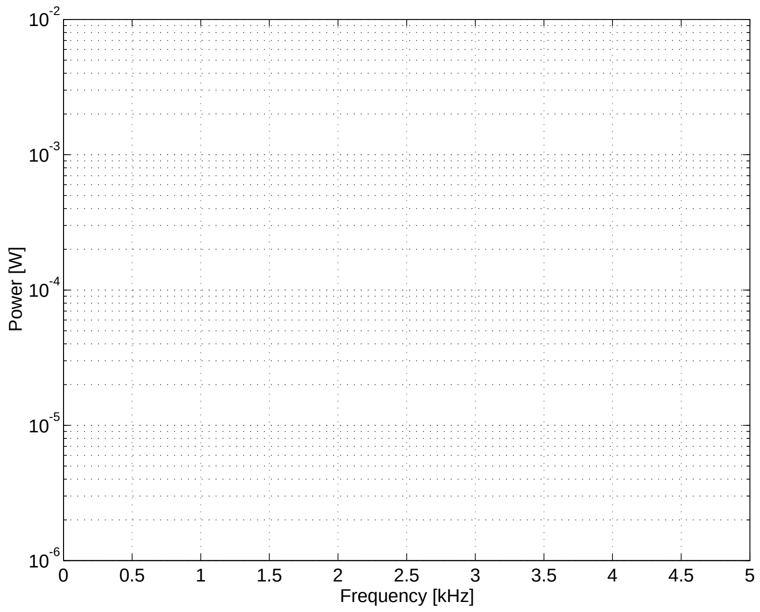
E.1 Generate 1,024 samples from a zero-mean white Gaussian noise process of unit power. Use this sequence as an input to a *blackbox*, that represents a filter with unknown bandwidth and order.

```
≫ clf
≫ wn = gauss(0,1,1024);
≫ cn = blackbox(wn);
```

E.2 Feed the output of the *blackbox* to a band-pass filter with variable pass-band.

```
≫ spect_est(cn)
```

When the function *spect_est* is invoked, you will be asked to specify the frequency range for the spectral estimation process and the bandwidth of the band-pass filter. Cover the spectral range [0, 5 kHz] and use a filter bandwidth of 250 Hz. After the required information is entered, the function will return the power of the output signal inside each pass-band range and display the estimated magnitude spectrum.



Graph 2.1

E.3 Plot the output from the function *spect_est* in Graph 4.1. Compare the accuracy of the spectral estimation which approximates the frequency response of the *blackbox* by superimposing the PSD function of the blackbox output:

```
≫ hold on, psd(cn)
```

Q2.8 Determine the pass-band and the order of the unknown filter represented by the *blackbox* function.

EXPERIMENT 3
BINARY SIGNALLING FORMATS

OBJECTIVES

In this experiment you will investigate how binary information is serially coded for transmission at baseband frequencies. In particular, you will study:

- line coding methods which are currently used in data communication applications;
- power spectral density functions associated with various line codes;
- causes of signal distortion in a data communications channel;
- effects of intersymbol interference (ISI) and channel noise by observing the eye pattern.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 381–391.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 144–165.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 197–200 and 227–230.

PRE-LAB ASSIGNMENT

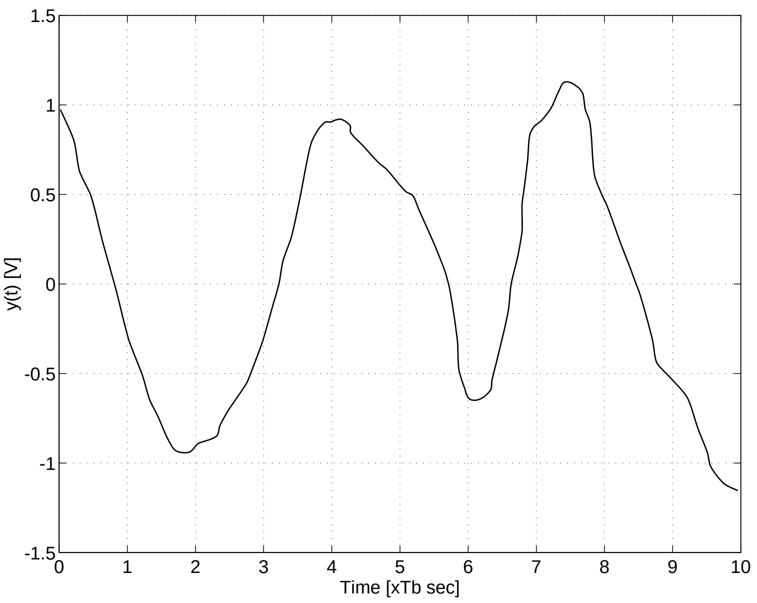
1. Given the binary sequence $b = \{1, 0, 1, 0, 1, 1\}$, sketch the waveforms representing the sequence b using the following line codes:

a. unipolar NRZ;
b. polar NRZ;
c. unipolar RZ;
d. bipolar RZ;
e. manchester.

Assume unit pulse amplitude and use binary data rate $R_b = 1\,\text{kbps}$.

2. Determine and sketch the power spectral density (PSD) functions corresponding to the above line codes. Use $R_b = 1\,\text{kbps}$. Let $f_1 > 0$ be the location of the first spectral null in the PSD function. If the transmission bandwidth B_T of a line code is determined by f_1, determine B_T for the line codes in question 1 as a function of R_b.

3. Let $y(t)$ be a waveform observed at the output of a communications channel. The waveform has been encoded using a polar NRZ line code.



To display an eye diagram, $y(t)$ is fed to a storage oscilloscope where the horizontal sweep time is set to $3\,T_b$. Sketch the corresponding eye diagram.

PROCEDURE

A. Binary Signalling Formats: Line Code Waveforms

Binary 1's and 0's such as in pulse-code modulation (PCM) systems, may be represented in various serial bit signalling formats called *line codes*. In this section you will study signalling formats and their properties.

A.1 You will use the MATLAB function *wave_gen* to generate waveforms representing a binary sequence:

```
wave_gen( binary_sequence, 'line_code_name', Rb )
```

where R_b is the binary data rate specified in bits per second (bps). If you use the function *wave_gen* with the first two arguments only, it will default to the binary data rate set by the variable *binary_data_rate*, which is 1,000 bps. Create the following binary sequence:

```
≫  b = [ 1 0 1 0 1 1 ];
```

Generate the waveform representing b, using unipolar NRZ line code with $R_b = 1$ kbps and display the waveform x.

```
≫  x = wave_gen( b, 'unipolar_nrz', 1000 );
≫  waveplot(x)
```

A.2 Repeat step A.1 for the following line codes:

- polar NRZ (`'polar_nrz'`);
- unipolar RZ (`'unipolar_rz'`);
- bipolar RZ (`'bipolar_rz'`);
- manchester (`'manchester'`).

You may want to simplify your command line by using:

```
≫  waveplot( wave_gen( b, 'line_code_name' ) )
```

Since you will compare waveforms at the same R_b, you can use the function *wave_gen* with only two arguments.

Q3.1 For the above set of line codes determine which will generate a waveform with no *dc* component regardless of binary sequence represented. Why is the absence of a *dc* component of any practical significance for the transmission of waveforms?

A.3 Power spectral density (PSD) functions of line codes: Generate a 1,000 sample binary sequence:

```
≫ b = binary(1000);
```

Display the PSD function of each line code used in part A.1:

```
≫ psd( wave_gen( b, 'line_code_name' ) )
```

Let:

- f_{p1}: first spectral peak;
- f_{p2}: second spectral peak;
- f_{n1}: first spectral null;
- f_{n2}: second spectral null;

such that all $f_{(.)} > 0$. Record your observations in Table 3.1.

$R_b =$	f_{p1}	f_{n1}	f_{p2}	f_{n2}	B_T
unipolar NRZ					
polar NRZ					
unipolar RZ					
bipolar RZ					
manchester					

Table 3.1

Location of the first spectral null determines transmission bandwidth B_T.

A.4 To illustrate the dependence of the PSD function on the underlying binary data rate, use the manchester line code and vary R_b:

```
≫ psd( wave_gen( b, 'manchester', Rb ) )
```

where `Rb` $\in$ {5 kbps, 10 kbps, 20 kbps}. You may replace manchester by any other line code used in part A.1. Observe the location of spectral peaks and nulls and relate them to R_b.

Q3.2 For a baseband data communications channel with usable bandwidth of 10 kHz, what is the maximum binary data rate for each of the line codes examined in part A.1.

B . Channel Characteristics

In this part you will simulate characteristics of a communications channel.

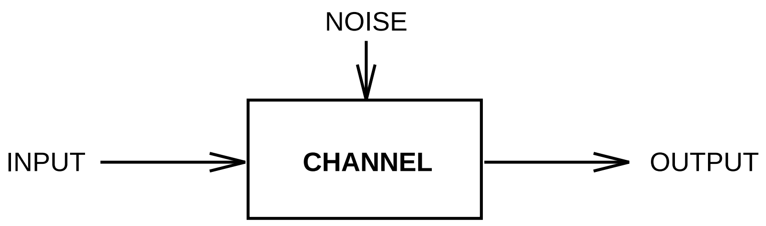


Fig 3.1 Channel model

The MATLAB function that represents the channel response is *channel* which is called with the following arguments:

```
channel( input, gain, noise_power, bandwidth )
```

B.1 Create a 10 sample binary sequence b and generate a waveform representing b in polar NRZ signalling format. Use $R_b = 1\,\text{kbps}$.

```
≫  b = binary(10);
≫  x = wave_gen( b, 'polar_nrz', 1000 );
```

From your observation in part A, determine the transmission bandwidth B_T of `x`:

$B_T =$ Hz.

B.2 Consider a baseband data transmission channel with unity gain and additive white Gaussian noise (AWGN) where the noise power is 10^{-2} W and the channel bandwidth is 4.9 kHz. Transmit waveform `x` over this channel. Display the channel input and output waveforms:

```
≫  y = channel( x, 1, 0.01, 4900 );
≫  subplot(211), waveplot(x)
≫  subplot(212), waveplot(y)
```

Estimate the transmitted sequence from the display of the channel output waveform.

$\hat{\texttt{b}} =$

Compare your estimate with the original sequence `b`.

B.3 **Effect of channel noise on the transmitted waveform:** Gradually increase the channel noise power while keeping the channel bandwidth at 4.9 kHz and observe changes in the channel output.

```
≫ subplot(212), waveplot( channel(x,1,sigma,4900) )
```

where `sigma` $\in \{0.1, 0.5, 1, 2, 5\}$. At what noise power level, does the channel output waveform becomes indistinguishable from noise?

B.4 You can also observe effects of increasing channel noise power by looking at the PSD of the channel output waveform.

```
≫ b = binary(1000);
≫ x = wave_gen(b, 'polar_nrz', 1000);
≫ clf, subplot(121), psd(x), a = axis;
≫ subplot(122), psd(channel(x,1,0.01,4900))
≫ axis(a), hold on
≫ psd(channel(x,1,1,4900))
≫ psd(channel(x,1,5,4900))
```

Q3.3 Since the channel noise is additive and uncorrelated with the channel input, determine an expression that will describe the PSD of the channel output in terms of the input and noise PSD functions.

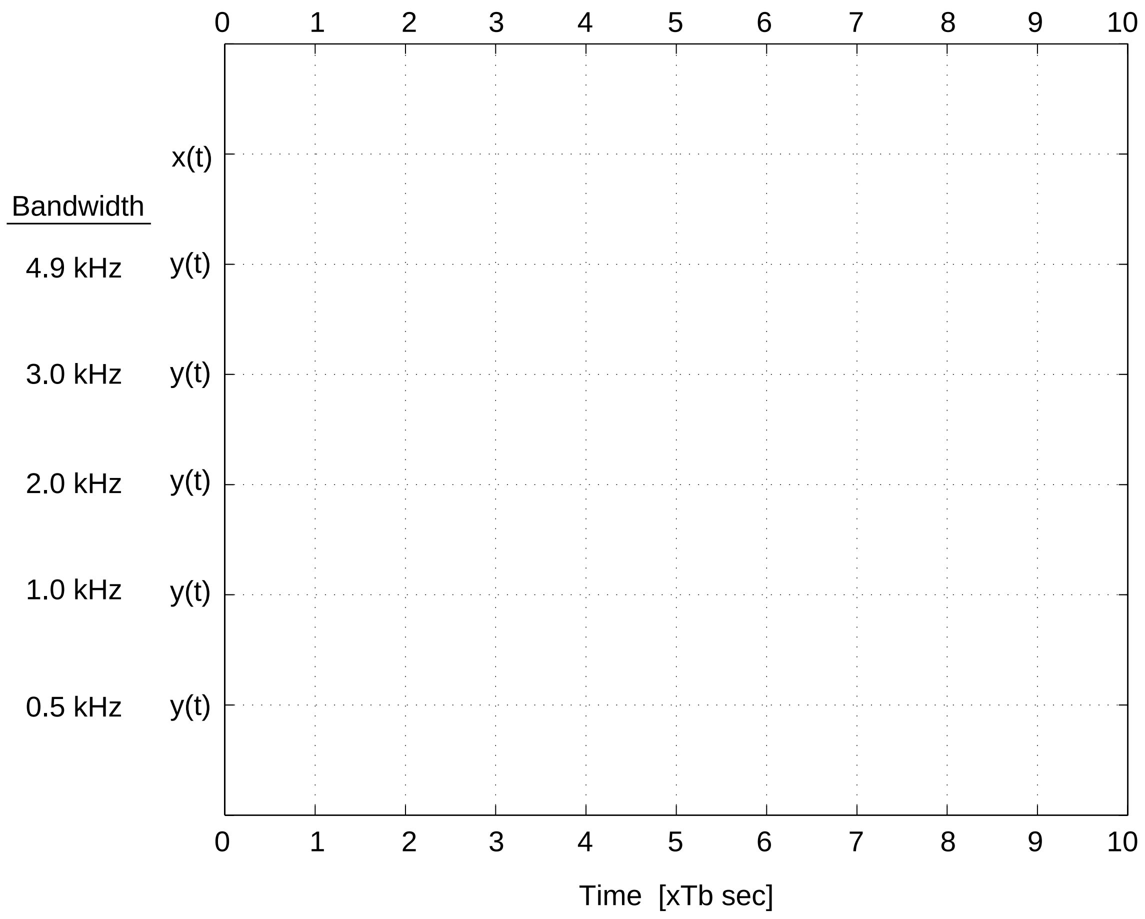
B.5 **Effects of channel bandwidth on transmitted waveform:** Distortion observed in the time display of the channel output is due to finite bandwidth of the channel and due to noise. To study distortion due to channel bandwidth only, set noise power to zero and regenerate the channel output waveform:

```
≫ clf
≫ b = binary(10);
≫ x = wave_gen( b, 'polar_nrz', 1000 );
≫ subplot(211), waveplot(x)
≫ subplot(212), waveplot(channel(x,1,0,4900))
```

B.6 Investigate effects of channel bandwidth on the output waveform.

```
≫ subplot(212), waveplot( channel(x,1,0,bw) )
```

where `bw` $\in \{3000, 2000, 1000, 500\}$. Observe the delay in the output waveform due to filtering characteristics of the channel. Plot the input and output waveforms.



Graph 3.1

C . Eye Diagram

Effects of channel filtering and noise can be best seen by observing the output waveform in the form of an *"eye diagram"*. The eye diagram is generated with multiple sweeps where each sweep is triggered by a clock signal[1] and the sweep width is slightly larger than the binary data period $T_b = 1/R_b$. In this simulation the eye diagram is based on a sweep width of $2T_b$.

C.1 Generation of Eye Diagram:

```
≫  b = [ 1 0 0 1 0 1 1 0 ];
```

[1] The clock signal is either externally provided or derived from the channel output. It is used as the external trigger input to a storage oscilloscope.

```
≫  x = wave_gen( b, 'polar_nrz', 1000);
≫  clf
≫  subplot(221), waveplot(x)
≫  subplot(223), eye_diag(x)
```

The eye diagram for the waveform x represents what you should expect to see for an undistorted signal. To observe how the eye diagram is generated and to observe effects of signal distortion as the signal x is transmitted over a finite bandwidth channel with no noise component:

```
≫  y = channel( x, 1, 0, 4000 );
≫  subplot(222), waveplot(y)
≫  subplot(224), eye_diag(y,-1)
```

If the second argument to the function *eye_diag* is negative, you have to hit the Return key for the next trace to be displayed. This will assist you to understand how the eye diagram is generated.

C.2 Key parameters to be measured with an eye diagram are shown below.

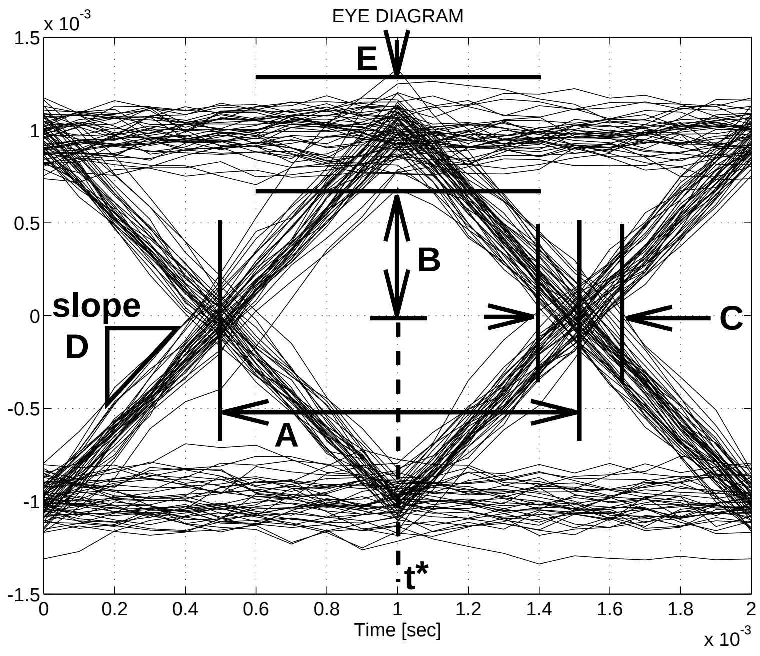


Fig 3.2 Interpretation of the eye pattern

A time interval over which the waveform can be sampled;
B margin over noise;
C distortion of zero crossings;
D slope: sensitivity to timing error;
E maximum distortion;

t* optimum sampling instant measured with respect to the time origin. If the binary data period is T_b, then the waveform will be sampled at t^*, $t^* + T_b$, $t^* + 2T_b$, $\ldots$ for signal detection.

Generate the eye diagram from a polar NRZ waveform at the channel output for values of noise variance `s2` and channel bandwidth `bw` shown in Table 3.2. Record t^*, A and B for each set of `s2` and `bw`.

```
≫  clf
≫  b = binary(100);
≫  x = wave_gen( b, 'polar_nrz', 1000 );
≫  eye_diag( channel( x, 1, s2, bw ) )
```

Polar NRZ Line Code				
`s2`	`bw`	t^*	**A**	**B**
0.01	3000			
	2000			
	1000			
0.02	4000			
0.08				
0.10				

Table 3.2

C.3 Repeat step C.2 for manchester line code and record your results in Table 3.3.

Manchester Line Code				
`s2`	`bw`	t^*	**A**	**B**
0.01	3000			
	2000			
	1000			
0.02	4000			
0.08				
0.10				

Table 3.3

Q3.4 When you compare the eye diagrams from C.2 and C.3 for `s2` = 0.01 and `bw` = 1000, for which line code do you observe a "reasonable" eye diagram? Explain the difference in terms of the respective line code properties.

C.4 Generate eye diagrams as in step C.2 for polar RZ and unipolar RZ and unipolar NRZ line codes and observe how the line code dictates the shape and the symmetry of the eye diagram.

EXPERIMENT 4
DETECTION

OBJECTIVES

In this experiment you will investigate the signal detection process by studying elements of a receiver and of the decoding process. In particular you will:

- investigate the characteristics of matched filters;
- study performance of various receiver structures based on different receiver filters by measuring probability of bit error;
- use the eye diagram as an investigative tool to set parameters of the detection process.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 391–400.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 497–504, 521–532.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 539–573.

PRE-LAB ASSIGNMENT

1. A matched filter is to be designed to detect the rectangular pulse

$$r(t) = rect\left(\frac{t - T_b/2}{T_b}\right), \quad \text{with } T_b = 1 \text{ msec.}$$

a. Determine the impulse response of the matched filter.

b. Determine the output of the matched filter if $r(t)$ is the input.

c. Repeat parts **a** and **b** for a triangular pulse of 10 msec duration.

2. Let $Y(t) = X(t) + n(t)$, represent the waveform at the output of a channel. $X(t)$ is a polar NRZ waveform with unit pulse amplitude and binary data rate R_b of 1 kbps. $n(t)$ is a white noise process with PSD function:

$$S_n(f) = N_o/2 = 0.5 \times 10^{-4} \text{ W/Hz}.$$

If $Y(t)$ is applied to a matched-filter receiver:

a. Determine the rms value of $n(t)$ and the peak signal amplitude at the output of the matched filter.

b. Determine E_b, the average energy of $X(t)$ in a bit period.

c. Determine the probability of bit error $P_e = Q(\sqrt{2E_b/N_o})$.

3. If $Y(t)$ in Question 2 is applied to a RC-filter with frequency response:

$$H_{rc}(f) = \frac{1}{1 + j2\pi fRC},$$

with $RC = 1/(2000\pi)$,

a. determine the peak signal amplitude and rms value of the noise at the filter output;

b. determine the probability of bit error P_e, if $X(t)$ were to be detected by a receiver based on the RC-filter.

PROCEDURE

A . Characteristics of Matched Filters

A.1 Generate a rectangular pulse with unit pulse amplitude and 1 msec pulse duration.

```
≫  r = wave_gen(1,'polar_nrz',1000);
```

A.2 Display `r` and the impulse response of a matched filter based on `r`.

```
≫  subplot(311), waveplot(r)
≫  subplot(312), match('polar_nrz')
```

A.3 Observe the matched filter output if `r` is applied to its input.

```
≫  rm = match('polar_nrz',r);
≫  subplot(313), waveplot(rm)
```

Determine the time when the filter output reaches its maximum value. How is this time related to the waveform `r`?

Q4.1 How would you determine the peak amplitude value of the matched filter output directly from the above plots?

A.4 Repeat parts A.1–A.3 for a triangular pulse with 10 msec pulse width and unit peak amplitude.

```
≫  r = wave_gen(1,'triangle',100);
≫  clf; subplot(311), waveplot(r)
≫  subplot(312), match('triangle')
≫  rm = match('triangle',r);
≫  subplot(313), waveplot(rm)
```

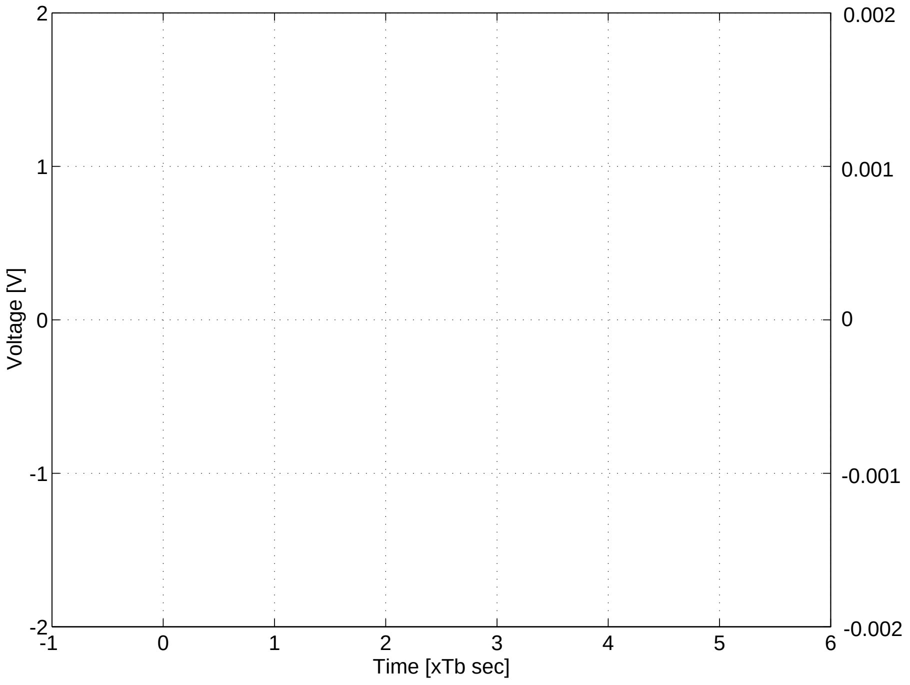
Q4.2 If the triangular pulse width is changed to 1 msec, determine the peak amplitude of the matched filter output?

A.5 Repeat parts A.1–A.3 for a manchester pulse with 10 msec pulse width and unit peak amplitude. Predict the matched filter impulse response and matched filter output. Verify your predictions using MATLAB functions.

A.6 Generate a polar NRZ waveform that represents the 5-sample binary sequence [1 0 0 1 0]. The binary data rate R_b is 1 kbps and the pulse amplitude A is 1 V.

```
≫  x5 = wave_gen([1 0 0 1 0],'polar_nrz',1000);
≫  clf, subplot(211), waveplot(x5)
```

Record the waveform x5 in Graph 4.1 using the amplitude scale shown on the left.



Graph 4.1

A.7 Apply x5 to a matched filter. Record the matched filter output in Graph 4.1 using the amplitude scale shown on the right.

```
≫  subplot(212), waveplot( match('polar_nrz',x5) )
```

Q4.3 Construct the waveform at a matched filter output if the input is a unipolar NRZ waveform that represents the binary sequence [1 0 0 1 0].

B . Signal Detection

B.1 Generate a 10-sample binary sequence and a waveform that represents this binary sequence in polar NRZ signalling format.

```
≫ b10 = binary(10);
≫ x10 = wave_gen(b10,'polar_nrz',1000);
≫ clf, subplot(211), waveplot(x10)
```

B.2 Apply `x10` to a channel with 4.9 kHz bandwidth and AWGN where the noise power is 2 W. Display the channel output waveform `y10`:

```
≫ y10 = channel(x10,1,2,4900);
≫ subplot(212), waveplot(y10)
```

Decode the binary sequence from the waveform `y10`:

$\widehat{\texttt{b10}} =$

B.3 Apply `y10` to a matched filter. Display the output waveform `z10`:

```
≫ z10 = match('polar_nrz',y10);
≫ subplot(212), waveplot(z10)
```

Let T_b be the binary data period. Sample the output of the matched filter at $k\,T_b$, $k = 1, \ldots, 10$ and apply the following decision rule:

$$\widehat{b_k} = \begin{cases} 0, & \text{if} \quad \texttt{z10}(k\,T_b) \leq 0; \\ 1, & \text{if} \quad \texttt{z10}(k\,T_b) > 0; \end{cases}$$

where $\widehat{b_k}$ is the estimated value of the k*th* element of the binary sequence `b10`. Apply this decision rule on the matched filter output `z10`:

$\widehat{\texttt{b10}} =$

Compare your estimate with the original sequence `b10`:

`b10` =

Q4.4 Comment on whether it is easier to decode the transmitted binary sequence directly from the channel output `y10` or from the matched filter output `z10`. If sampling instants other than those specified above are used, the probability of making a decoding error will be larger. Why?

C . Matched-Filter Receiver

C.1 Generate a 2,000-sample binary sequence `b` and a polar NRZ waveform based on `b`:

```
≫ b = binary(2000);
≫ x = wave_gen(b,'polar_nrz');
```

Apply `x` to a channel with 4.9 kHz bandwidth and AWGN where the noise power σ_n^2 is 0.5 W. Let `y` be the channel output waveform.

```
≫ y = channel(x,1,0.5,4900);
```

C.2 Apply `y` to a matched filter. Display the eye diagram of the matched filter output `z`.

```
≫ z = match('polar_nrz',y);
≫ clf, eye_diag(z);
```

From the eye diagram, determine the optimum sampling instants and threshold value `v_th` for the detector to decode the transmitted binary sequence `b`. Sampling instants for the matched filter output are measured with respect to the time origin. For example, if the binary data period is T_b and the `sampling_instant` parameter is set to t_i, then the detector will sample the signal at t_i, $t_i + T_b$, $t_i + 2T_b$, ... etc.

`v_th` = V.

`sampling_instant` = sec.

Use `v_th` and `sampling_instant` in the detector which will operate on the matched filter output. Record the resulting probability of bit error P_e (also referred to as bit error rate (BER)) in Table 4.1.

```
≫ detect(z,v_th,sampling_instant,b);
```

σ_n^2 $[W]$	P_e – *empirical*	P_e – *theoretical*
0.5		
1.0		
1.5		
2.0		

Table 4.1

C.3 Repeat C.1–C.2 for channel noise power of 1, 1.5, and 2 W without displaying the eye diagram of the matched filter output `z`. Record P_e results in Table 4.1. **Remark:** In Experiment 3 you have observed that the optimum sampling instants and the threshold value are independent of channel noise power. Therefore, you can use the optimum sampling instants determined in part C.2 to decode the matched filter output for different channel noise power levels.

C.4 If different sampling instants other than the optimum values are used, the resulting BER will be larger. You can observe this by decoding the binary sequence using values for the `sampling_instant` parameter that are 0.9 and 0.5 times the optimal value used in part C.3.

Q4.5 Evaluate theoretical probability of bit error values for all cases considered above and record in Table 4.1. Note that the PSD function of a white noise process can be determined as:

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

where the system bandwidth in this experiment is 5 kHz.

D . Low-Pass Filter Receiver

D.1 Apply a rectangular pulse to a first-order RC-filter of 1 kHz bandwidth. Display the filter output and measure the peak amplitude A_r:

```
≫ r = wave_gen(1,'unipolar_nrz'); r_lpf = rc(1000,r);
≫ subplot(211); waveplot(r)
≫ subplot(212), waveplot(r_lpf);
```

$A_r =$ V.

D.2 Generate 2,000 samples from a zero-mean white noise sequence of 0.5 W power. Apply the noise sequence to the RC-filter. Record the rms value of the output noise power.

```
≫ n = gauss(0,0.5,2000);
≫ meansq(rc(1000,n))
```

$\sigma_n^2 =$ W.

Q4.6 From the results in parts D.1 and D.2, determine the ratio A_r/σ_n, where A_r is the peak signal amplitude measured in D.1 and σ_n is the rms value of the output noise. If y in part C.1 is applied to a receiver which uses the above RC-filter, determine the resulting BER.

D.3 Regenerate y from part C.1. Apply y to the RC-filter. Display the eye diagram of the output waveform `z_lpf`.

```
≫ y = channel(x,1,0.5,4900);
≫ z_lpf = rc(1000,y);
≫ clf, eye_diag(z_lpf);
```

D.4 From the eye diagram, determine the optimum sampling instant and threshold value. Decode the binary sequence form `z_lpf`.

```
≫ detect(z_lpf,v_th,sampling_instant,b);
```

Compare the resulting BER with the BER evaluated in step C.2.

D.5 Repeat part D.4 for the channel noise power of 1, 1.5, and 2 W. Record results in Table 4.2.

σ_n^2 [W]	P_e	
	BW = 1.0 kHz	**BW = 0.5 kHz**
0.5		
1.0		
1.5		
2.0		

Table 4.2

D.6 Repeat parts D.3 – D.5 for a first-order RC-filter with 500 Hz bandwidth. Record the resulting BER values in Table 4.2.

```
≫ z_lpf = rc(500,y);
≫ eye_diag(z_lpf)
≫ detect(z_lpf,v_th,sampling_instant,b);
```

Q4.7 Explain why the BER resulting from a low-pass filter of 500 Hz bandwidth is smaller than the BER resulting from a low-pass filter of 1 kHz bandwidth. Will the BER be further decreased if a low-pass filter of 100 Hz bandwidth is used?

E . Implementation of Matched Filter Structure

Integrate-and-dump filter is a practical circuit implementation of a matched filter.

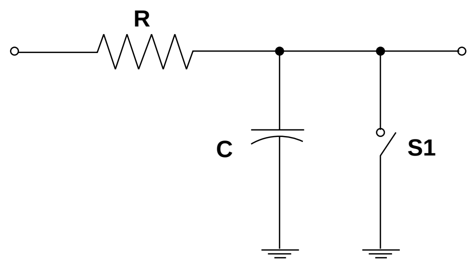


Fig 4.1 Matched filter implementation

In the circuit, the switch S1 is closed periodically for a short time to discharge the capacitor. The RC network with a very large time constant acts as an integrator.

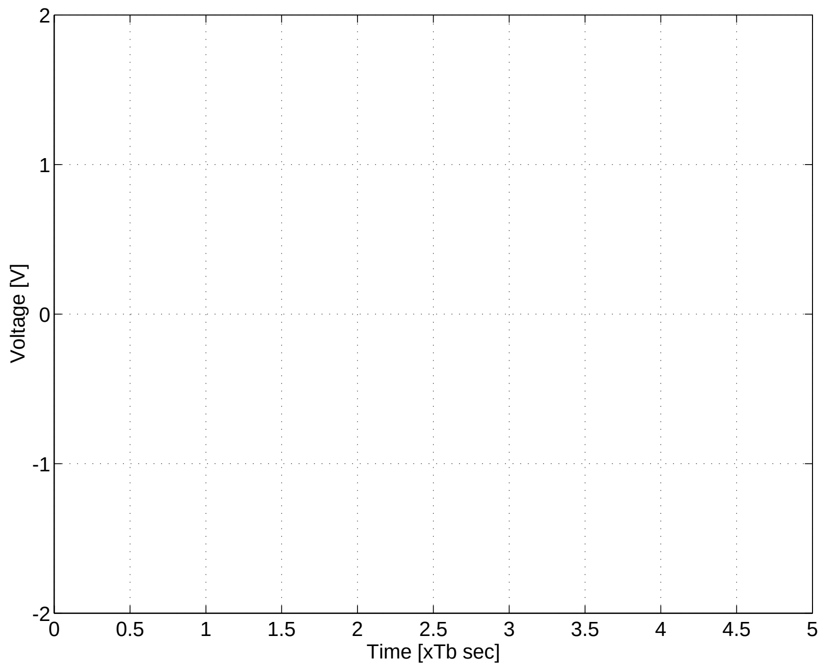
E.1 To understand the operation of the filter, apply `x5` generated in part A.6 to the filter and observe both input and output waveforms.

```
≫  y5 = int_dump(x5);
≫  clf; waveplot(x5); hold on; waveplot(5*y5)
```

Record `x5` and the magnified filter output `5 ∗ y5` in Graph 4.2.

Q4.8 In Graph 4.2 indicate when the capacitor is charging and discharging. When is the switch S1 closed? Can you specify the optimum sampling instants without studying the eye diagram?

E.2 Repeat parts D.3 – D.4, but use the integrate-and-dump filter instead of the low-pass filter. Compare the resulting BER with those resulting from a low-pass filter and a matched filter.



Graph 4.2

EXPERIMENT 5
QUANTIZATION

OBJECTIVES

In this experiment you will:

- study uniform and non-uniform quantization characteristics;
- observe effects of different types of quantizer distortion;
- measure signal to quantization error ratio for various quantizer types as a function of the input signal power.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 430–439.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 130–144.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 187–197.

PRE-LAB ASSIGNMENT

1. Consider a 2-bit uniform, mid-riser type quantizer. Assume that the quantizer has been designed for the input range $[-1,\ 1]$.

   a. Determine all quantization levels used by the quantizer.

   b. Determine the quantization interval for each quantization level.

   c. Sketch the input-to-output mapping characteristic.

2. Let the quantization noise be a uniformly distributed random variable defined over the interval $[-q/2,\ q/2]$, where q is the length of the quantization interval of a N-bit uniform quantizer:

   a. Determine q as a function of N.

   b. Determine the variance of quantization noise.

PROCEDURE

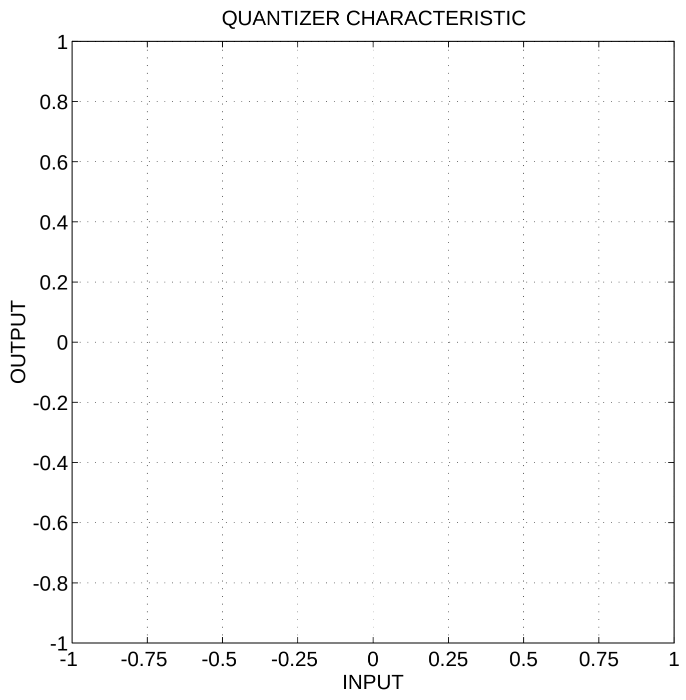
A. Uniform Quantization

After a message has been sampled, the next step is to quantize the samples. Because an analog signal has a continuous range of amplitudes, pulses in the sampled signal will have an infinite number of amplitude levels. It is not possible to assign unique, finite length code words to an infinite number of amplitudes. Quantization consists of approximating the sampled signal by a signal made up of discrete amplitudes.

A.1 Display the input-to-output mapping characteristic of a 2-bit uniform quantizer:

```
≫  quant_ch( 2, 'uniform' )
```

Sketch the quantizer characteristic in Graph 5.1.



Graph 5.1

A.2 Let x be the analog pulse amplitude, and x_q be the quantizer output, if x is the input. Assume that pulse amplitude x is normalized such that $|x| \leq 1$. From Graph 5.1 derive the quantizer mapping rule:

Quantization Interval	$\mathbf{x_q}$	**Binary Code**

Table 5.1

Observe the length of each interval that appears in the first column of Table 5.1. How is the quantization interval length related to the number of quantization levels of a uniform quantizer? The entries in the 2*nd* column are referred to as quantization levels; only these values appear at the quantizer output.

A.3 Generate the 8 element sampled sequence x where each element of x represents the analog pulse amplitude at that sampling instant:

```
≫  x = [ 0.8, 0.6, 0.2, -0.4, 0.1, -0.9, -0.3, 0.7 ];
```

Apply quantization rule described in Table 5.1 to sequence x:

Element	**x**	$\mathbf{x_q}$	**Binary Code**
1			
2			
3			
4			
5			
6			
7			
8			

Table 5.2

Compare results obtained from applying the quantizer mapping rule with the output from the MATLAB function *quantize*:

```
≫  xq = quantize( x, 2 )
```

A.4 Devise a rule which assigns a unique 2-bit binary symbol to each quantization level. Use a natural code[1] which assigns the binary symbol

[1] There are several other coding methods such as *magnitude-sign code*, *Gray code*, etc. Each code has unique properties that makes it suitable in certain applications.

"00" to the most negative quantization level (-0.75 in this example). As you move from the most negative to the most positive quantization level, numeric value of the binary symbol is incremented. This procedure is called *source coding*. In Table 5.1, fill in the *3rd* column with 2-bit natural code. Apply this coding procedure to the quantized sequence `xq` and enter the binary codes into the *3rd* column of Table 5.2. You may check your answer with the MATLAB function *bin_enc* (natural binary encoder):

```
≫  xbin = bin_enc( xq, 2 )
```

Q5.1 Is it possible to code pulse amplitudes in sequences `x` and `xq` with 2-bit binary symbols such that each pulse amplitude remains uniquely identifiable?

A.5 One of the most common methods of converting an analog signal into a stream of binary digits is *pulse code modulation* (PCM). In PCM, you perform the following sequence of operations:

- sampling;
- quantization;
- source coding;
- conversion into a serial data stream.

Thus, if the source coded variable `xbin` is converted into a serial data, you in fact, generate PCM data representing the sequence `x`:

```
≫  par2ser( xbin )
```

B. Distortion Induced by Quantization

B.1 Quantization generates an approximation to the original sequence. To observe this point generate samples from a sinusoid of unit amplitude:

```
≫  x = sin( 2*pi*20*[1:400]/SAMPLING_FREQ );
```

Default sampling frequency is set to 10 kHz by the variable `sampling_freq`. Use a 2-bit uniform quantizer and compare the quantized waveform with the original sequence `x`:

```
≫  clf, waveplot(x), hold on, waveplot(quantize(x,2))
```

Repeat using 3-, 4- and 5-bit uniform quantizers.

Since the quantized sequence is only an approximation to the original, quantization produces *distortion.* There are two types of distortion associated with a quantizer: overload (or clipping) distortion and quantization noise. Overload distortion occurs when the input signal exceeds the quantizer's input range — in this experiment $[-1, 1]$. Once the quantizer range has been exceeded, quantizer output will remain at its maximum or minimum value until the input falls within the quantizer's input range.

B.2 Set the amplitude of the sinusoid `x` to 0.9 and display `x` and its 3-bit quantized version:

```
≫  a = 0.9;
≫  clf
≫  waveplot(a*x), hold on, waveplot(quantize(a*x,3))
```

Observe that, since max $|\texttt{x}| = 0.9 < 1$, no clipping occurs. Now increase the signal amplitude and repeat the above step for `a` $= 1.25$, 2 and 5.

Q5.2 Determine the percentage of samples in `x` that are clipped by the quantizer when the amplitude is set to 0.9, 2 and 5. For the 3-bit uniform quantizer what are the minimum and maximum quantization levels?

To avoid clipping a quantizer is matched to the input signal. (This feature is also available to the MATLAB function *quantize*, if it is called with the *scaling* option. For more information type `help quantize`.) The second type of distortion is due to quantization noise which arises because of the difference between the input amplitude and the quantized sample amplitude.

B.3 Use samples from the sinusoid `x` to observe quantization error `x` $-$ `xq`:

```
≫  clf
≫  xq = quantize(x, N); xe = x - xq;
≫  waveplot(xe)
```

where `N` $= 2, 3, 4$ and 5. For each choice of `N`, record the maximum absolute value of the quantization error into Table 5.3.

N	**2**	**3**	**4**	**5**
max $|\texttt{xe}|$				

Table 5.3

Q5.3 Derive a formula for the maximum absolute quantization error due to a N-bit uniform quantization as a function of N. Under what conditions is the quantization error reaches this maximum?

B.4 Generate the following sequence and display its PSD function:

```
≫ x = sin( 2*pi*512*[1:2048]/SAMPLING_FREQ );
≫ psd(x)
```

For N $= 2, 3, 4, 5,$ and 6 display the PSD function of the quantized waveform xq and evaluate the corresponding quantization error power in dB. Enter results into Table 5.4.

```
≫ xq = quantize(x,N);
≫ psd(xq)
≫ sq2 = 10*log10(var(x - xq))
```

N	**2**	**3**	**4**	**5**	**6**
σ_q^2					

Table 5.4

Under the assumption that the quantization error amplitude is uniformly distributed over $[-q/2,\ q/2]$ where q is the quantization step size of an N-bit uniform quantizer, noise power is given by:

$$\sigma_q^2 = -\ (4.77 + 6.02N) \quad \text{dB}.$$

Compare results in Table 5.4 with those obtained from the formula.

C . Non-uniform Quantization

C.1 Generate 100 samples representing voiced segment of a typical speech signal and display the sequence:

```
≫ s = speech(100);
≫ subplot(211), waveplot(s)
```

Observe that the signal amplitude is restricted to $[-1,\ 1]$ and there are samples in s with amplitude ± 1. Thus, s spans the entire input range of the quantizer.

C.2 Quantize `s` using a 6-bit uniform quantizer and display the quantized waveform:

```
≫  sq = quantize(s,6);
≫  subplot(212), waveplot(sq)
```

The quantized waveform `sq` closely resembles the original sequence `s`. Now, scale `s` by a factor `K` such that max |`K s`| ≤ 0.01.

```
≫  x = normalize( s, 0.01 );
```

As you display `x`, you will observe that `x` is an attenuated version of `s`. Quantize `x` using the same 6-bit uniform quantizer and display the quantized signal `xq`:

```
≫  xq = quantize(x,6);
≫  subplot(211),waveplot(x),subplot(212),waveplot(xq)
```

Can you explain why `xq` differs so much from `s` or `sq`?

C.3 Non-uniform quantization: Display the quantizer characteristic of a μ-law companding 3-bit quantizer:

```
≫  clf, quant_ch( 3, 'mu_law' )
```

Q5.4 Consider a sampled signal with pulse amplitudes restricted to the interval $[-0.25, 0.25]$. If the signal is quantized, how many quantization levels are *actually* used by a μ-law companding 3-bit quantizer? By a uniform 3-bit quantizer?

C.4 Apply μ-law companding to sampled signals `s` and `x` and then quantize:

```
≫  msq = mu_inv( quantize(mu_law(s), 6) );
≫  mxq = mu_inv( quantize(mu_law(x), 6) );
```

Since quantized samples must be restored to their original values, non-uniform quantization is equivalent[1] to the following sequence of operations: *compand — uniform quantization — expand.* Display waveforms

[1] `mu_law` and `mu_inv` are inverse functions of each other. They represent companding and expansion characteristics, respectively. This you can verify by comparing any arbitrary sequence z with `mu_inv(mu_law(z))`.

`msq`, `mxq` and compare with original signals `s` and `x`. It is of particular interest to compare `msq` with `sq` and `mxq` with `xq`. First, the large amplitude signal `s`:

```
≫  clf, subplot(311), waveplot(s)
≫  subplot(312), waveplot(sq)
≫  subplot(313), waveplot(msq)
```

Now for the small amplitude signal `x`:

```
≫  figure(2), subplot(311), waveplot(x)
≫  subplot(312), waveplot(xq)
≫  subplot(313), waveplot(mxq)
```

C.5 For a given number of quantization levels, a uniform quantizer is optimum if input sample amplitudes are uniformly distributed over the input range $[-1,\ 1]$. However, speech signal amplitude distribution is best modelled by a Laplace distribution. Generate the sample pdf of 2,000 samples from a Laplace distribution with variance 0.01:

```
≫  close(2), figure(1), clf
≫  a = laplace( 2000, 0.01 );
≫  pdf( a, 30 )
```

Compand the sequence `a` by a μ-law compander and display the sample pdf of the resulting sequence:

```
≫  b = mu_law(a);
≫  hold on, pdf( b, 30 )
```

The pdf of the companded sequence `b` resembles a uniform pdf, which results in better utilization of available quantization levels.

C.6 In practise, any quantizer has to operate on signals at varying amplitude levels. Therefore, to assess the quantizer performance, it must be tested with signals at different power levels. One measure of quantizer performance is the signal to quantization noise ratio SQNR. Consider a 8-bit quantizer used in a typical voice grade A/D converter:

Algorithm:

1. Generate a sequence of 1,000 voice signal samples at a specified power level and measure the signal power: σ_s^2.
2. Quantize the sequence using an 8-bit uniform quantizer. Measure the resulting quantization error power: $\sigma_q^2(\text{uniform})$.

3. Quantize the sequence using a μ-law companding 8-bit quantizer. Measure the resulting quantization error power: $\sigma_q^2(\mu\text{-law})$.
4. Measure the $SQNR = \sigma_s^2/\sigma_q^2$ for both quantizer types.

This algorithm is available in MATLAB function *exp5_c6*

```
≫  exp5_c6( power )
```

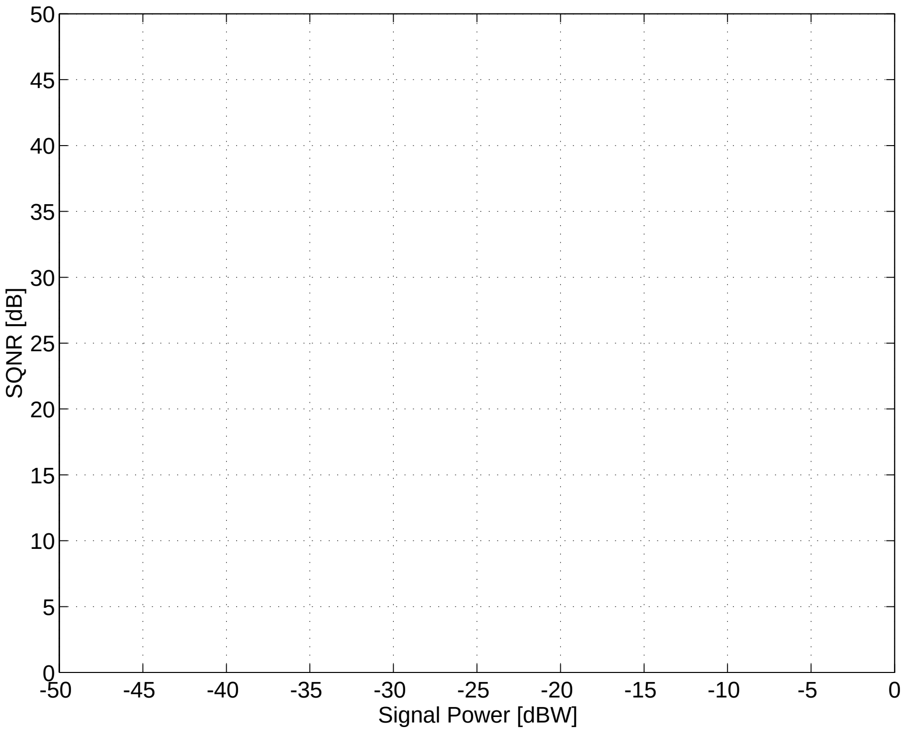
where `power` = [.0001, .001, .01, .1, 1]. The output is in the form:

$$[\ \sigma_s^2, \quad \text{SQNR(uniform)}, \quad \text{SQNR}(\mu\text{-law})\ ],$$

where all quantities are measured in dB. Enter results into Table 5.5. Also plot SQNR vs. σ_s^2 for both quantizer types in Graph 5.2.

σ_s^2	SQNR(uniform)	SQNR(μ-law)

Table 5.5



Graph 5.2

Q5.5 Based on the results depicted in Graph 5.2 comment on the preferred quantizer type for input signals if:

a. $\sigma_s^2 \in [-40, -10]$ dBW.

b. $\sigma_s^2 \in [-10, 0]$ dBW.

c. $\sigma_s^2 \in [-40, 0]$ dBW.

APPENDIX

WORKING WITH REAL SOUND SAMPLES

This appendix provides information on how you can work with real sound samples and how you can play-back processed sound samples through the built-in audio circuitry of your Sun-SPARC workstation. The Sun-SPARC workstation family has the capability to sample and to play-back sound at 8 kHz sampling frequency using 8-bit, μ-law companding quantization. You are provided with sample sound files that you can process, quantize and listen to. Play-back of the processed sound files allows qualitative assessment of the sound quality to be expected from different quantization schemes.

The sample sound file is ***sample_speech***. This file has been extracted from a larger sound file that has been created on a different hardware platform by sampling a CBS news broadcast at 8 kHz and using 14-bit uniform quantization. The original file contains 123,000 samples representing approximately 15 seconds of sound. The ***sample_speech*** file contains 20,000 samples to minimize the computational load and therefore to improve the response time of your workstation. The 14-bit quantized samples have been converted into floating point format and as such they will serve as ***"analog"*** samples. They are stored in the ***x_analog*** array.

You can process samples in the ***x_analog*** array and listen to the results. Since a 2.5 seconds long sound file may not convey full information about the quantization effects, you are also given versions of the 15 seconds long sound file that had been pre-processed with 1–8 bit uniform and μ-law companding quantization.

How Can I Load The Sample Sound File?

Use the command:

```
≫  load sample_speech
```

The sound samples are stored in the array ***x_analog***. Remember that the samples are played back through an 8-bit DAC. Therefore 8-bit quantization will be your sound quality reference.

How Can I Listen to Sound Samples?

To listen to ***x_analog*** enter the command:

```
≫  play( x_analog )
```

You can control the sound level through the "audio control panel". Start this utility by entering the command:

```
≫ !gaintool &
```

What Can I do With The Sound Samples?

On a minimal level you should try different quantization methods. For example, try to compare 4-bit uniform and μ-law companding quantization. First normalize the input array to prevent quantizer overload.

```
≫ xn = normalize( x_analog );
≫ q4 = quantize( xn, 4 );
≫ mq4 = mu_inv( quantize(mu_law(xn),4) );
≫ play( q4 )
≫ play( mq4 )
```

You should also try to deliberately turn-off scaling in order to introduce quantizer overload:

```
≫ q4_ovld = quantize( x_analog, 4 );
≫ play( q4_ovld )
```

It is also instructive to visually observe the quantized waveform(s). For example, try:

```
≫ subplot(211), time_index = [10000:10200];
≫ waveplot(xn(time_index)), waveplot(q4(time_index))
```

How Can I Listen to The Pre-Processed Sound Files?

Enter the command

```
≫ playback( N, type, volume_level )
```

where `N` $\in \{1, 2, \ldots, 8\}$, `type` is `'uniform'` or `'mu_law'`, and the optional `volume_level` is as before an integer between 0 and 100.

Remarks

It is likely that you will not hear much difference[1] in the sound quality of the 5–8 bit quantized samples; except the background noise which is the manifestation of the increasing quantization noise with decreasing

[1] This observation is partly due to the quality of the sound circuit and the built-in speaker of the workstation.

number of quantization levels. Also note that the quantization noise will appear as ***hiss*** which confirms the assumption that when sufficiently large number of quantization levels are used, the quantization noise will be uncorrelated with the input and can be accurately modeled as a white noise process. Working with a large number of quantization levels consumes plenty of processing power and you may have to wait much longer to create and listen to your results. Remember that you are doing a software emulation of a process that is usually performed in dedicated hardware. It is therefore recommended that you only listen to the pre-processed sound files at 5–8 bit quantization levels, and restrict your processing to 1–4 bit quantization.

Different volume levels may be misleading, as the human psychology — or shall we say the teen-age ear? — has the tendency to associate loud sound with “better quality”. As a result of normalization, and companding, the sound files you generate will most likely be at different sound levels. Therefore, some experimentation with the volume control is necessary.

EXPERIMENT 6
DIGITAL MODULATION

OBJECTIVES

In this experiment you will apply concepts of baseband digital transmission and analog continuous wave modulation to the study of band-pass digital transmission. You will examine:

- generation of digital modulated waveforms;
- *coherent* (synchronous) and *noncoherent* (envelope) detection of modulated signals;
- system performance in the presence of corrupting noise.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 512–542.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 331–344 and 532–551.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 368–374 and 574–583.

PRE-LAB ASSIGNMENT

1. Consider the binary sequence $b = [\ 1\ 0\ 0\ 1\ 0\ ]$. Let the binary data rate R_b be equal to 1 kbps and let the peak amplitude of all digital modulated waveforms be set to 1 V.

   a. Sketch the ASK waveform representing the binary sequence b using a carrier frequency of 5 kHz.

   b. Sketch the PSK waveform representing the binary sequence b using a carrier frequency of 5 kHz.

   c. Let the *mark* and *space* frequencies used by an FSK modulator be set to 3 and 6 kHz, respectively. Sketch the resulting FSK waveform representing the binary sequence b.

2. Sketch the power spectral density function for each of the modulated signals in Question 1.

3. If an ASK signal is applied to the input of a coherent detector shown in Fig. 6.1, sketch the waveforms at the output of each block.

4. Repeat Question 3 for a noncoherent detector shown in Fig. 6.2.

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.

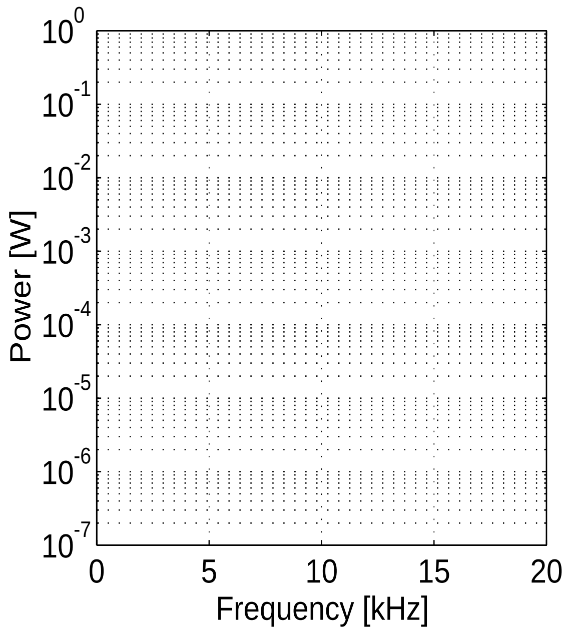
```
≫  xu = wave_gen(b,'unipolar_nrz');
≫  sa = mixer( xu, osc(8000) );
```

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.

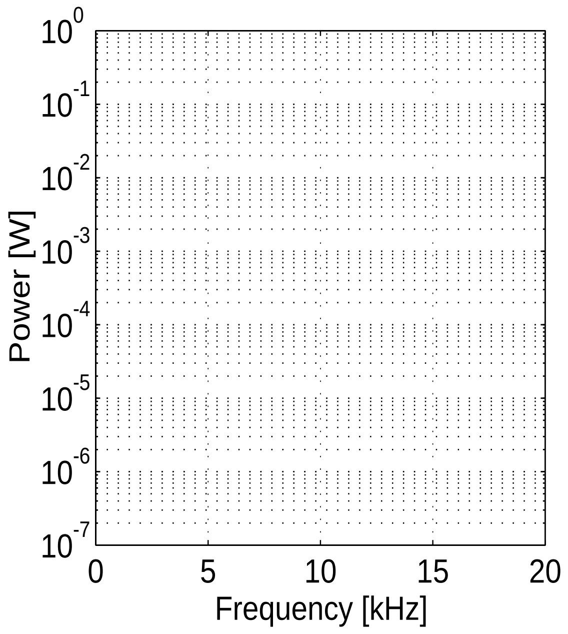
```
≫  tt = [1:500];
≫  subplot(211), waveplot(xu(tt))
≫  subplot(212), waveplot(sa(tt))
```

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)
```



Graph 6.1



Graph 6.2

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.

```
≫  xp = wave_gen(b,'polar_nrz');
≫  sp = mixer( xp, osc(8000) );
```

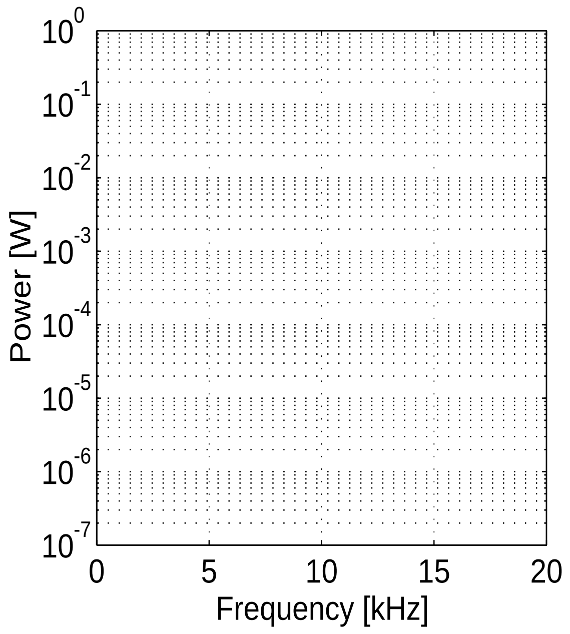
A.5 Display the first 500 samples of the waveforms `xp` and `sp`:

```
≫  subplot(211), waveplot(xp(tt))
≫  subplot(212), waveplot(sp(tt))
```

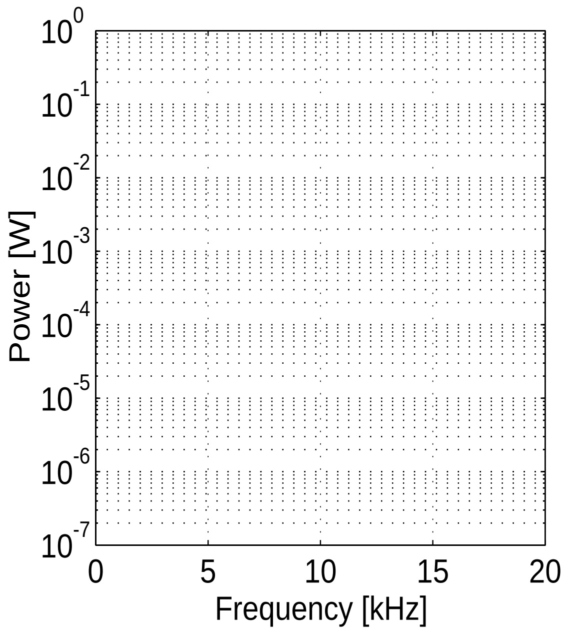
What is the phase difference between `sp` and the carrier $\sin(2\pi f_c t)$ during the first and second bit periods?

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

```
≫  subplot(211), psd(xp,fr)
≫  subplot(212), psd(sp,fr)
```



Graph 6.3



Graph 6.4

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.

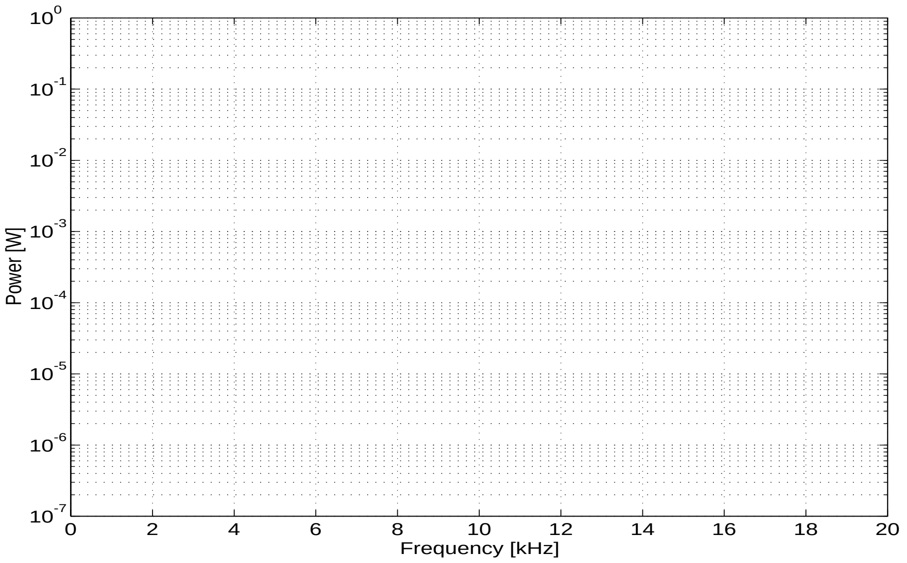
```
≫  sf = vco(xp);
```

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

```
≫  subplot(211), waveplot(xp(tt))
≫  subplot(212), waveplot(sf(tt))
```

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

```
≫  clf
≫  psd(sf,fr)
```



Graph 6.5

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?

B . Digital Modulated Signal Detection

Coherent Detection

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

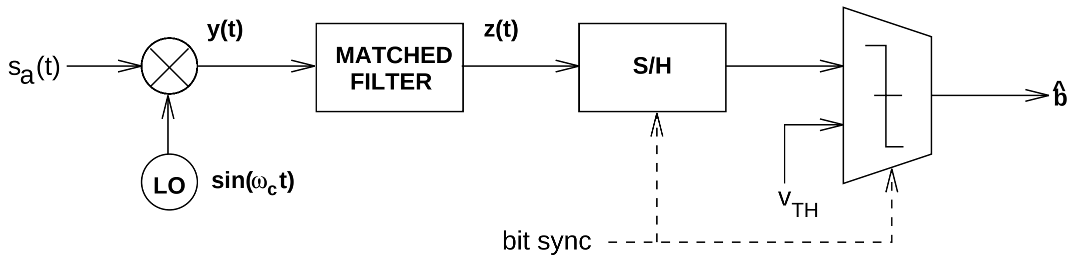
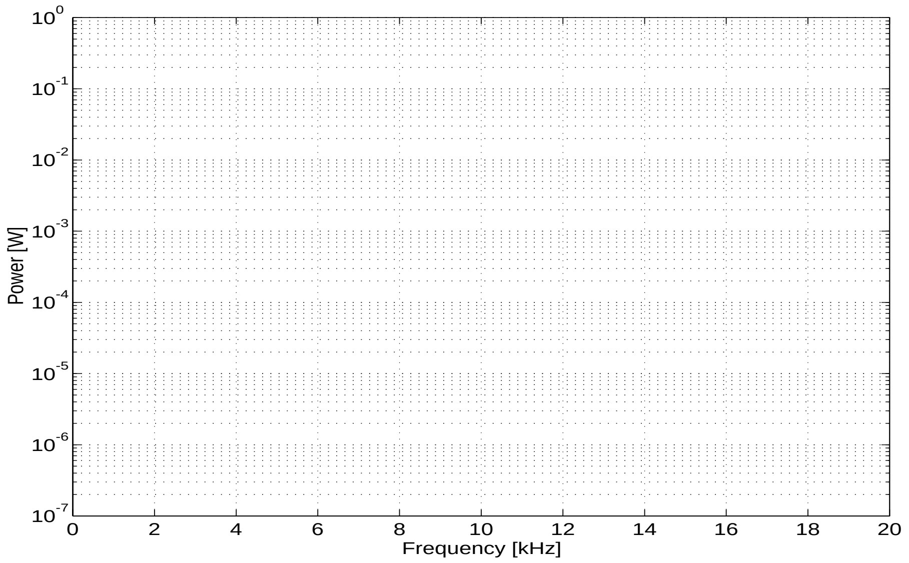


Fig. 6.1 Coherent Detector

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.

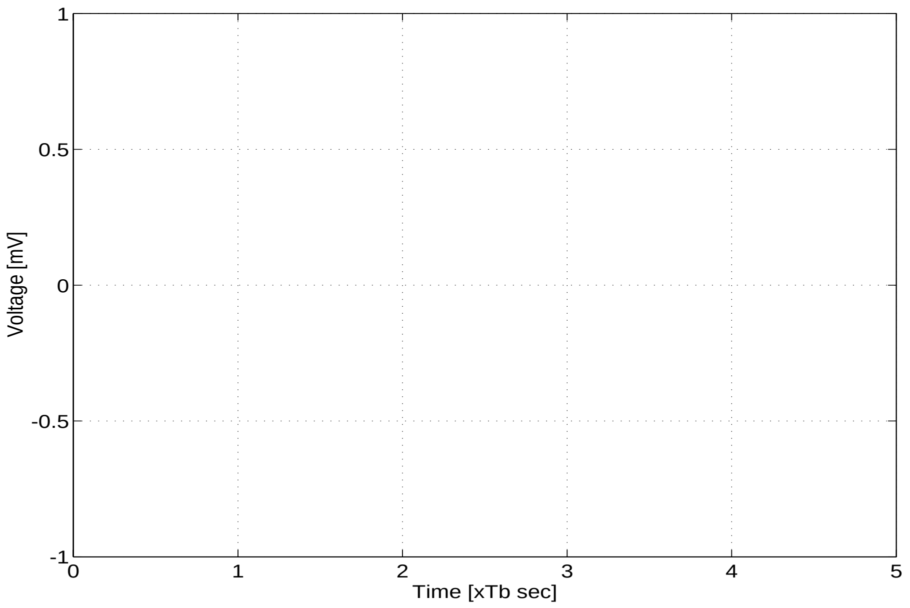
```
≫   ya = mixer( sa, osc(8000) );
≫   clf, subplot(211), waveplot(ya(tt))
≫   subplot(212), psd(ya,fr)
```



Graph 6.6

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))
```



Graph 6.7

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,phase_error) );
≫ za = match('unipolar_nrz',ya);
≫ subplot(212), waveplot(za(1:500))
```

PHASE ERROR	**PEAK AMPLITUDE** [V]
$0°$	
$20°$	
$60°$	
$80°$	
$120°$	

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.

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 $= 60°$; $\hat{\mathtt{b}}_{1-5} =$

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

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))
```

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)
```

Determine the frequency of the envelope of the matched filter output:
Envelope frequency = Hz.

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.

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.

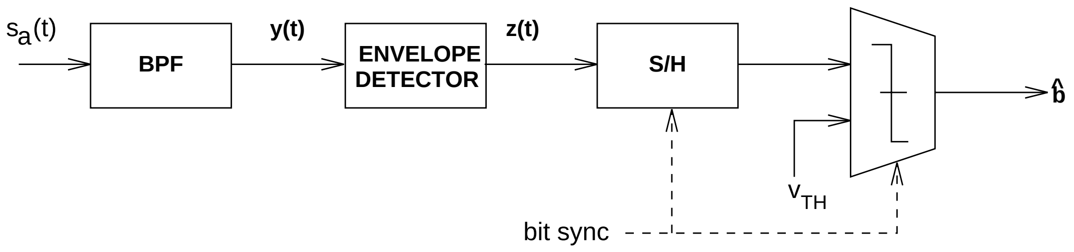


Fig. 6.2 Noncoherent Detector

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))
```

Decode the first 5 bits of the transmitted sequence.

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

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))
```

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);
```

From the eye diagram, determine optimum sampling instants and the threshold value. Apply `zm` to the decision circuit, and record the resulting probability of bit error.

```
≫  detect(zm,vth,sampling_instant,b);
```

Coherent detection: P_e =

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);
≫  detect(ze,vth,sampling_instant,b);
```

D . Optional

D.1 Determine the BER for a PSK signal using coherent detection under the same channel conditions as in part C.

D.2 Determine the BER for a FSK signals using noncoherent detection for the same channel conditions as in part C.

EXPERIMENT 7
DIGITAL COMMUNICATION

OBJECTIVES

In this experiment you will integrate blocks representing communication system elements into a larger framework that will serve as a model for digital communication systems. In particular, you will study:

- building blocks that constitute baseband and band-pass systems;
- inverse functional relations between A/D – D/A conversion; and transmitter – receiver elements;
- how to encounter phase reversal by differential encoding;
- how to select transmitter parameters based on empirically determined channel characteristics.

REFERENCES

1. **A. B. Carlson**, *"Communication Systems,"* third ed., McGraw-Hill Inc., New York, 1986, pp. 430–439 and 512–532.
2. **L. W. Couch II**, *"Digital and Analog Communication Systems,"* third ed., MacMillan Publishing Co., New York, 1990, pp. 130–144 and 547.
3. **S. Haykin**, *"An Introduction to Analog & Digital Communications,"* J. Wiley & Sons Inc., New York, 1989, pp. 177–202 and 539–580.

PRE-LAB ASSIGNMENT

1. Consider the *unipolar NRZ* and *manchester* binary signalling formats. Let the channel noise PSD function be $S_n(f) = N_0/2$.

a. Determine the energy per bit interval E_b for both line codes as a function of signal amplitude A, and binary data period T_b.

b. Determine the bit error rate P_e for matched filter based detection of both line codes as a function E_b and N_0. Sketch P_e for $E_b/N_0 \in [0.1, 10]$.

2. Determine the bit error rate P_e for coherent detection of ASK and PSK digital modulation schemes as a function E_b and N_0.

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*.

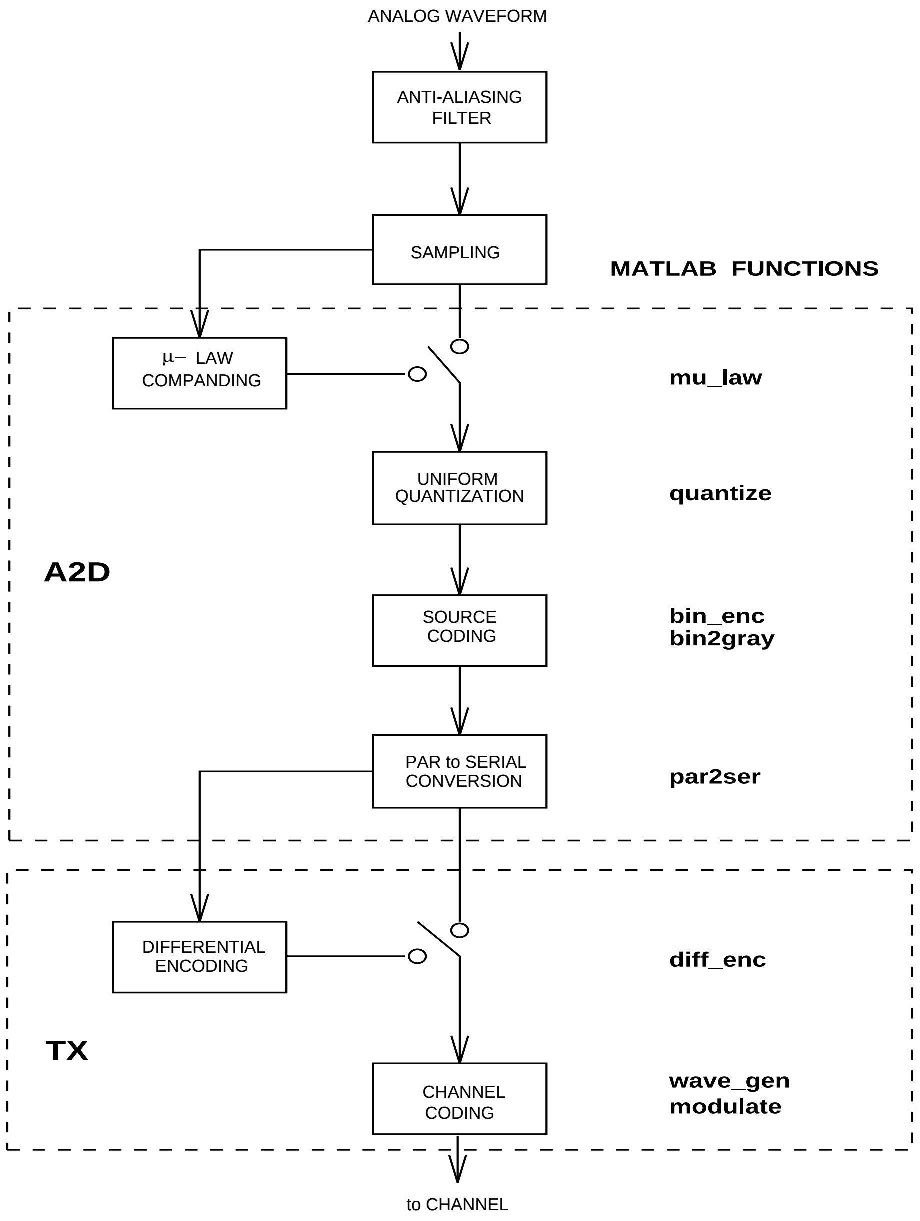


Fig. 7.1 A/D and transmitter block diagrams

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*.

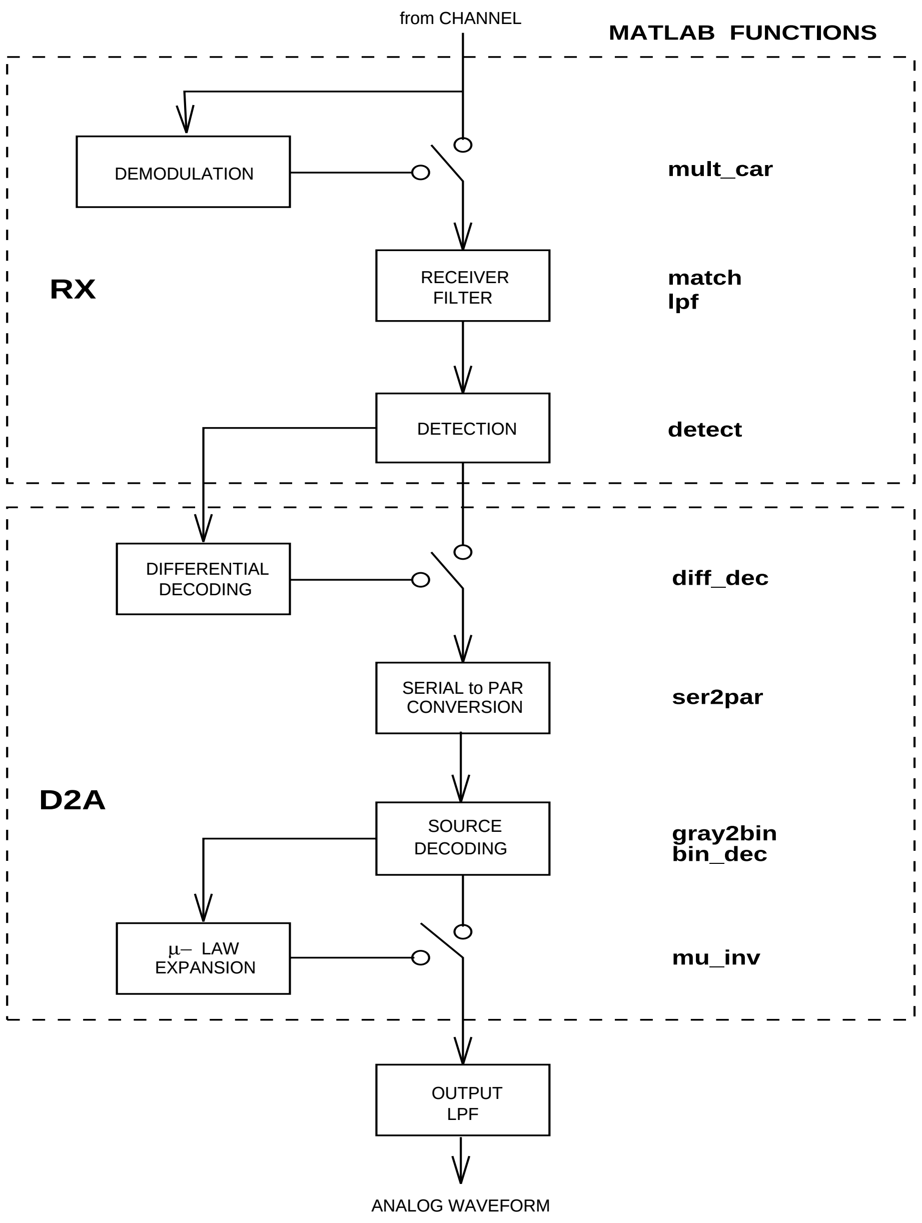


Fig. 7.2 Receiver and D/A block diagrams

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 );
≫  s_analog = d2a( s_binary, 6 );
```

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

```
≫  s_binary(1:10)
```

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)
```

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.

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);
≫  xw = tx( x_pcm, 'manchester', 'no_diff', 1000 );
```

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 );
≫  subplot(211), waveplot(x)
≫  subplot(212), waveplot(x_analog)
```

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 );
≫  z = -channel( u, 1, 0.01, 19000 );
≫  u_digital = rx( z, 'manchester', 'diff' );
≫  u_analog = d2a( u_digital, 6 );
≫  subplot(211), waveplot(x)
≫  subplot(212), waveplot(u_analog)
```

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.

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);
≫  m = tx(b,'manchester',Rb);
```

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*:

```
≫  ch_output = channel( A*ch_input, 1, 1, 19000 );
≫  rx( ch_output, 'linecode', b );
```

[1] N_0 and σ_n^2 are related as follows: $N_0 = \sigma_n^2/20,000$ (see Experiment 4).

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.

A	**UNIPOLAR NRZ**		**MANCHESTER**	
(volts)	E_b/N_0	P_e	E_b/N_0	P_e
0.2				
0.3				
0.4				
0.5				
0.6				
0.7				

Table 7.1

Compare the above empirical P_e values with theoretical values determined in the pre-lab assignment and plot them the same graph.

D.2 Consider the following baseband communications channel:

```
≫  ch_output = channel( ch_input, 1, 2, 19000 );
```

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`.

E . Band-Pass Communication

E.1 Generate 100 samples from a speech signal:

```
≫  s = speech(100);
```

Your task is to prepare a binary representation of `s` for transmission over a band-pass channel. Parameters that specify the transmitter, and channel characteristics are:

- **A/D conversion** — *a2d*:
  8-bit, μ-law quantization.

- **Transmitter** — *tx*:
  Digital modulation type: PSK.
  Binary data rate $R_b = 100$ kbps.
- **Channel** — *channel*:
  Channel gain: 0 dB.
  Channel noise power: 1 W.
  Usable Bandwidth: 600 kHz to 1,400 kHz.
- **Bit Error Rate:**
  $P_e = 10^{-2}$.

The channel input-output relation is then given by:

```
≫  out = channel( A*in, 1, 1, [600000, 1400000] );
```

where $N_0 = \sigma_n^2/(20R_b)$ with $\sigma_n^2 = 1$, and `A` is the waveform amplitude. Compute the channel SNR, E_b/N_0, to satisfy the BER requirement. Determine the required transmitter power, channel noise and carrier frequency. Empirically determine the bit error rate and compare it with its theoretical value. Convert the output from the receiver function *rx* into an analog form using the function *d2a* and compare the original speech waveform `s` with the recovered one.

$\mathbf{E_b}$	$\mathbf{N_0}$	$\mathbf{f_c}$	$\mathbf{P_e}$

Table 7.2

Q7.3 Is the empirical BER different than the theoretical value? If there is a large difference between these two values explain the discrepancy.

APPENDICES

APPENDIX A

SIMULATION ENVIRONMENT

If one attempts to model an analog signal on a digital computer, the simulation environment will impose certain limitations on the degree of precision that can be achieved in representing the analog signal. This appendix delineates characteristics of analog signals simulated on a digital computer and discusses their significance from a user's perspective.

- In MATLAB, a function $f(t)$ is defined by an array of data points that represents amplitude of the function at sampling instances. For example, the array $[\,f(t_1),\ f(t_2), \ldots, f(t_n)]$ represents $f(t)$ at sampling instances $t_1,\ t_2, \ldots, t_n$. Irrespective of the number of sample points included in the data array, the result still corresponds to a sampled version of the function. To make this observation tangible, assume that you issue the following MATLAB command:

  ```
  ≫   x = sin(t)
  ```

  MATLAB is not a symbolic computation environment such as MAPLE. As a result, the above command line will generate an error message, since the argument of the sin function `t`, has not been previously defined. If you attempt to define the argument, you can only do it by specifying `t` at sampling instances. For example:

  ```
  ≫   fs = 1
  ≫   t = [ 2 * pi * 1000 * [1:100] * (1/fs) ]
  ≫   x = sin(t)
  ```

  will work just fine, but what you have created through the above set of commands, are 100 samples of a 1 kHz sinusoid sampled at a frequency of only 1 Hz. If you leave out the term `1/fs` or modify its value, you only change the sampling frequency, but the fact that you are **sampling** remains unrevocably embedded in the very structure of the resulting data array.

  So how do you generate an analog waveform? The simple answer is that you can **not** generate a true analog signal. Consequently, one can only talk about *"pseudo-analog"* signals, that have been sampled at a sufficiently high rate so that the sampled signal will accurately represent the analog waveform. The MATLAB function *wave_gen* generates waveforms representing binary line codes exactly in this manner. For example, if you specify a binary data rate of 5 kHz, the sampling frequency is set to 50 kHz or higher depending on the experiment. The $f_s = K\, f_i$ rule, where f_i is the input signal frequency or binary data rate and $K \geq 10$ is the sampling constant, is a compromise[†] between the data array size and precision.

- A further implication of having to deal with *"pseudo-analog"* signals is the difficulty in displaying spectral representations over a frequency interval that extends beyond the $f_s/2$, where f_s is the sampling frequency. To illustrate this point consider an analog signal, a 1 kHz sinusoid. Also assume that the spectral display is constrained to the positive frequency axis. The spectrum of the sinusoid will display a single delta function at $f = 1$ kHz. However, as a result of the sampling theorem, the spectrum of the analog signal will be repeated at integer multiples of f_s and consequently, the spectrum of the *"pseudo-analog"* 1 kHz sinusoid sampled at 10 kHz will display delta functions at $f = (10\,n \pm 1)$ kHz with $n = 0, 1, \ldots$. In the present simulation environment, the MATLAB function *psd* restricts the frequency interval to $(\,0,\; f_s/2\,)$.

- Because an analog signal has a continuous range of amplitudes, pulses in the sampled signal will have an infinite number of amplitude levels. However, any number represented in a digital computer is only of finite precision due to finite register length. Since all numbers within MATLAB are represented in double precision floating point format, this limitation is of no practical consequence for the simulation environment. The MATLAB variable *eps* measures floating point relative accuracy. With an initial value of $2.204\,10^{-16}$ *eps* represents the distance from 1.0 to the next largest floating point number. This in turn, corresponds to 2^{53}-level quantization, which is more than adequate for the experiments.

[†] A similar problem is encountered by the digital oscilloscopes. Technical bulletins from of several manufacturers announce sampling rates of 20–250 Msamples/sec for a maximum captured frequency of 2–25 Mhz. The 10:1 ratio is based on the "*... generally accepted rule that you need a minimum of 10 samples per period to accurately reconstruct the waveform.*"

APPENDIX B

COMMUNICATION SYSTEM TOOLBOX

HOW TO GET STARTED ?

All global parameters are in the file *start.m*. When you first enter the MATLAB environment, type `start` to initialize global parameters.

RANDOM NUMBER GENERATION

binary	random binary digits
corr_seq	first order auto-regressive process
exponent	exponential random variate
gauss	Gaussian random variate
laplace	Laplace random variate
uniform	uniform random variate
realize	sinusoidal random process with random phase
speech	random voiced speech signal

PROBABILISTIC ANALYSIS

cdf	sample cdf of a random sequence.
exp_cdf	cdf of an exponential random variable
exp_pdf	pdf of an exponential random variable
gamma_pdf	pdf of a gamma random variable
gaus_cdf	cdf of a Gaussian random variable
gaus_pdf	pdf of a Gaussian random variable
lapl_cdf	cdf of a Laplacian random variable
lapl_pdf	pdf of a Laplacian random variable
meansq	mean-square power
pdf	sample pdf of a random sequence
Q	Q function
rayl_cdf	cdf of a Rayleigh random variable
rayl_pdf	pdf of a Rayleigh random variable
unif_cdf	cdf of a uniform random variable
unif_pdf	pdf of a uniform random variable
var	variance

PROBABILITY & RANDOM PROCESS GAMES

dice	random experiment with a die
dart	visual depiction of a dart game

guess	guess personal information data
new_born	sample function representing new born babies
person_data	generation of personal records
temperature	sample function representing day time temperature

GENERAL PURPOSE ANALYSIS TOOLS

acf	autocorrelation function
acf_plot	autocorrelation function display
ecorr	ensemble autocorrelation function
psd	power spectral density function
psd_plot	power spectral density function display

QUANTIZATION

a2d	analog-to-digital conversion
d2a	digital-to-analog conversion
mu_inv	μ-law expansion
mu_law	μ-law companding
quant_ch	quantizer characteristics
quant_ef	quantizer efficiency
quantize	uniform quantization

BINARY DATA PROCESSING

bcd	binary-coded-decimal coding
bin_enc	natural binary source coding
bin_dec	natural binary source decoding
bin2gray	natural binary to *gray*-code conversion
gray2bin	*gray*-code to natural binary conversion
bin2pol	binary to polar transformation
bin2bipo	binary to bipolar transformation
diff_dec	differential decoding
diff_enc	differential encoding
invert	1's complement of a binary sequence
par2ser	parallel-to-serial conversion
pol2bin	polar to binary transformation
ser2par	serial-to-parallel conversion
xor	exclusive OR

BINARY SIGNALLING FORMATS

manchest	Manchester pulse
rect_nrz	rectangular NRZ pulse
rect_rz	rectangular RZ pulse

triangle	triangular pulse
nyquist	Nyquist pulse
nyq_gen	generate Nyquist waveform
duob_gen	generate duobinary waveform
duobinar	modified duobinary pulse
modulate	digital modulated wave (ASK,BPSK,FSK)
osc	sinusoidal oscillator
vco	voltage controlled oscillator
wave_gen	binary signal waveform generation
waveplot	display binary signal waveform

DATA TRANSMISSION

bpf	band-pass filter
channel	data communication channel
eye_diag	eye diagram generation and display
lpf	low-pass filter
rc	1*st* order RC-filter
detect	binary data detection
envelope	envelope detector
int_dump	integrate-and-dump filter
mixer	two input mixer
match	matched filter
rx	receiver function
tx	transmitter function

UTILITY FUNCTIONS

blackbox	filter with unknown order and bandwidth
convert	convert a MATLAB array into a SUN audio file
exp5_c6	compute signal power and SQNR
fftsize	determine FFT size
fx	sample functions to be integrated
play	play-back a MATLAB array
playback	play-back a pre-processed sound file
limiter	limit input sequence to a user specified range
normalize	scale input sequence
rectify	rectify input sequence
sinc	$\sin(\pi x)/(\pi x)$
spec_est	spectral estimation
stair	modified version of the MATLAB function *stairs*
stat_plot	scatter diagram display