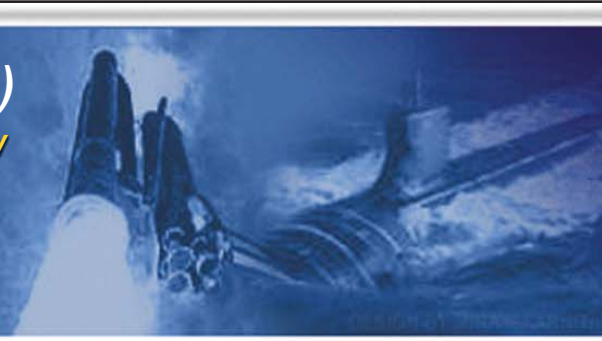
Applied Technology Institute (ATIcourses.com)

Stay Current In Your Field • Broaden Your Knowledge • Increase Productivity

349 Berkshire Drive • Riva, Maryland 21140
888-501-2100 • 410-956-8805
Website: www.ATIcourses.com • Email: ATI@ATIcourses.com



Fundamentals of Statistical Signal Processing: Chapter 2 by Dr. Steven Kay

ATI Provides Training In:

- Acoustic, Noise & Sonar Engineering
- Communications and Networking
- Engineering & Data Analysis
- Information Technology
- Radar, Missiles & Combat Systems
- Remote Sensing
- Signal Processing
- Space, Satellite & Aerospace Engineering
- Systems Engineering & Professional Development

Check Our Schedule & Register Today!

The Applied Technology Institute (ATIcourses.com) specializes in training programs for technical professionals. Our courses keep you current in state-of-the-art technology that is essential to keep your company on the cutting edge in today's highly competitive marketplace. Since 1984, ATI has earned the trust of training departments nationwide, and has presented On-site training at the major Navy, Air Force and NASA centers, and for a large number of contractors. Our training increases effectiveness and productivity.

Learn From The Proven Best!

Chapter 2

Computer Simulation

2.1 Introduction

Computer simulation of random phenomena has become an indispensable tool in modern scientific investigations. So-called *Monte Carlo* computer approaches are now commonly used to promote understanding of probabilistic problems. In this chapter we continue our discussion of computer simulation, first introduced in Chapter 1, and set the stage for its use in later chapters. Along the way we will examine some well known properties of random events in the process of simulating their behavior. A more formal mathematical description will be introduced later but careful attention to the details now, will lead to a better intuitive understanding of the mathematical definitions and theorems to follow.

2.2 Summary

This chapter is an introduction to computer simulation of random experiments. In Section 2.3 there are examples to show how we can use computer simulation to provide counterexamples, build intuition, and lend evidence to a conjecture. However, it cannot be used to prove theorems. In Section 2.4 a simple MATLAB program is given to simulate the outcomes of a discrete random variable. Section 2.5 gives many examples of typical computer simulations used in probability, including probability density function estimation, probability of an interval, average value of a random variable, probability density function for a transformed random variable, and scatter diagrams for multiple random variables. Section 2.6 contains an application of probability to the "real-world" example of a digital communication system. A brief description of the MATLAB programming language is given in Appendix 2A.

2.3 Why Use Computer Simulation?

A computer simulation is valuable in many respects. It can be used

a. to provide counterexamples to proposed theorems

b. to build intuition by experimenting with random numbers

c. to lend evidence to a conjecture.

We now explore these uses by posing the following question: What is the effect of adding together the numerical outcomes of two or more experiments, i.e., what are the probabilities of the summed outcomes? Specifically, if U_1 represents the outcome of an experiment in which a number from 0 to 1 is chosen at random and U_2 is the outcome of an experiment in which another number is also chosen at random from 0 to 1, what are the probabilities of $X = U_1 + U_2$? The mathematical answer to this question is given in Chapter 12 (see Example 12.8), although at this point it is unknown to us. Let's say that someone asserts that there is a theorem that X is *equally likely* to be anywhere in the interval $[0, 2]$. To see if this is reasonable, we carry out a computer simulation by generating values of U_1 and U_2 and adding them together. Then we repeat this procedure M times. Next we plot a *histogram*, which gives the number of outcomes that fall in each subinterval within $[0, 2]$. As an example of a histogram consider the $M = 8$ possible outcomes for X of $\{1.7, 0.7, 1.2, 1.3, 1.8, 1.4, 0.6, 0.4\}$. Choosing the four subintervals (also called *bins*) $[0, 0.5]$, $(0.5, 1]$, $(1, 1.5]$, $(1.5, 2]$, the histogram appears in Figure 2.1. In this

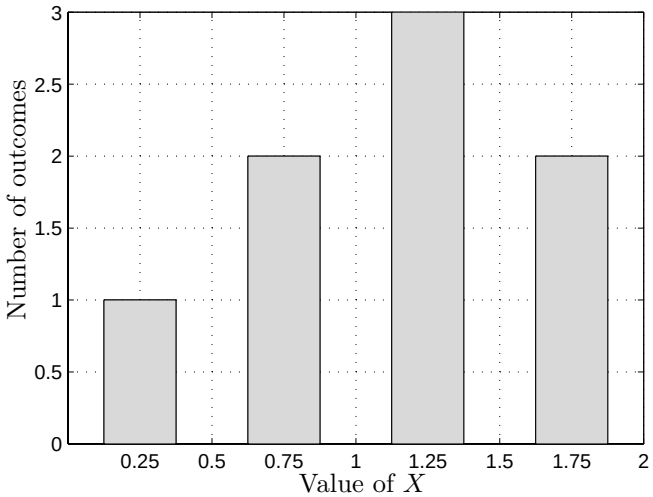


Figure 2.1: Example of a histogram for a set of 8 numbers in [0,2] interval.

example, 2 outcomes were between 0.5 and 1 and are therefore shown by the bar

centered at 0.75. The other bars are similarly obtained. If we now increase the number of experiments to $M = 1000$, we obtain the histogram shown in Figure 2.2. Now it is clear that the values of X are *not equally likely*. Values near one appear

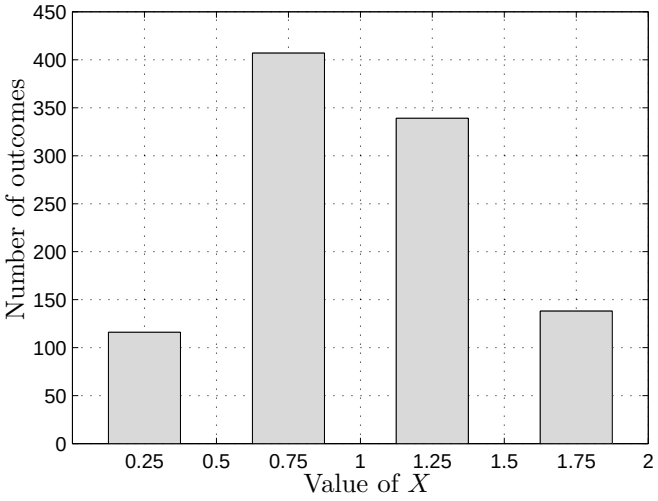


Figure 2.2: Histogram for sum of two equally likely numbers, both chosen in interval $[0, 1]$.

to be much more probable. Hence, we have generated a "counterexample" to the proposed theorem, or at least some evidence to the contrary.

We can build up our intuition by continuing with our experimentation. Attempting to justify the observed occurrences of X, we might suppose that the probabilities are higher near one because there are more ways to obtain these values. If we contrast the values of $X = 1$ versus $X = 2$, we note that $X = 2$ can only be obtained by choosing $U_1 = 1$ and $U_2 = 1$ but $X = 1$ can be obtained from $U_1 = U_2 = 1/2$ or $U_1 = 1/4, U_2 = 3/4$ or $U_1 = 3/4, U_2 = 1/4$, etc. We can lend credibility to this line of reasoning by supposing that U_1 and U_2 can only take on values in the set $\{0, 0.25, 0.5, 0.75, 1\}$ and finding all values of $U_1 + U_2$. In essence, we now look at a *simpler* problem in order to build up our intuition. An enumeration of the possible values is shown in Table 2.1 along with a "histogram" in Figure 2.3. It is clear now that the probability is highest at $X = 1$ because the number of combinations of U_1 and U_2 that will yield $X = 1$ is highest. Hence, we have learned about what happens when outcomes of experiments are added together by employing computer simulation.

We can now try to extend this result to the addition of three or more experimental outcomes via computer simulation. To do so define $X_3 = U_1 + U_2 + U_3$ and $X_4 = U_1 + U_2 + U_3 + U_4$ and repeat the simulation. A computer simulation with $M = 1000$ trials produces the histograms shown in Figure 2.4. It appears to

		U_2				
		0.00	0.25	0.50	0.75	1.00
	0.00	0.00	0.25	0.50	0.75	1.00
	0.25	0.25	0.50	0.75	1.00	1.25
U_1	0.50	0.50	0.75	1.00	1.25	1.50
	0.75	0.75	1.00	1.25	1.50	1.75
	1.00	1.00	1.25	1.50	1.75	2.00

Table 2.1: Possible values for $X = U_1 + U_2$ for intuition-building experiment.

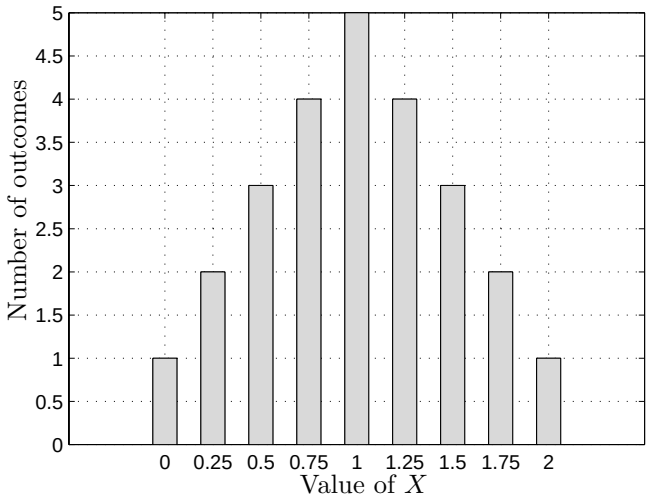
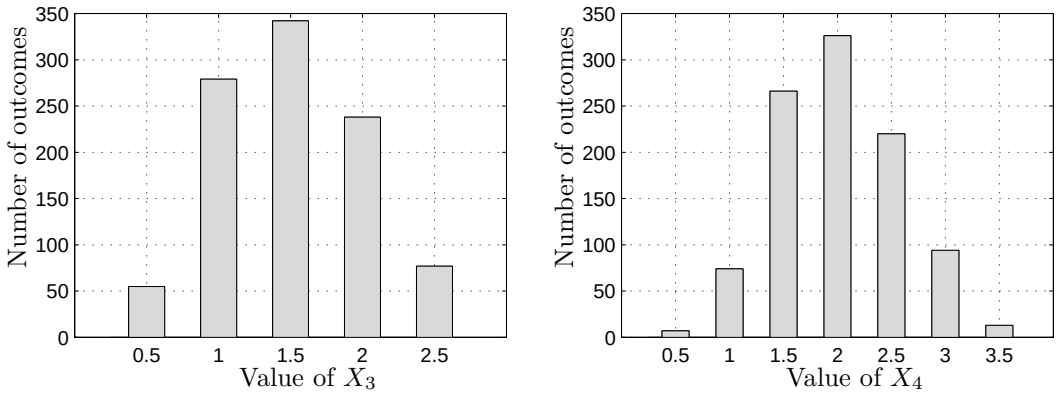


Figure 2.3: Histogram for X for intuition-building experiment.

bear out the conjecture that the most probable values are near the center of the $[0, 3]$ and $[0, 4]$ intervals, respectively. Additionally, the histograms appear more like a bell-shaped or Gaussian curve. Hence, we might now *conjecture*, based on these computer simulations, that as we add more and more experimental outcomes together, we will obtain a Gaussian-shaped histogram. This is in fact true, as will be proven later (see central limit theorem in Chapter 15). Note that we cannot *prove* this result using a computer simulation but only lend evidence to our theory. However, the use of computer simulations indicates *what* we need to prove, information that is invaluable in practice. In summary, computer simulation is a valuable tool for lending credibility to conjectures, building intuition, and uncovering new results.



(a) Sum of 3 U's (b) Sum of 4 U's

Figure 2.4: Histograms for addition of outcomes.

Computer simulations cannot be used to prove theorems.

In Figure 2.2, which displayed the outcomes for 1000 trials, is it possible that the computer simulation could have produced 500 outcomes in [0,0.5], 500 outcomes in [1.5,2] and no outcomes in (0.5,1.5)? The answer is yes, although it is improbable. It can be shown that the probability of this occuring is

$$\binom{1000}{500}\left(\frac{1}{8}\right)^{1000} \approx 2.2 \times 10^{-604}$$

(see Problem 12.27).

2.4 Computer Simulation of Random Phenomena

In the previous chapter we briefly explained how to use a digital computer to simulate a random phenomenon. We now continue that discussion in more detail. Then, the following section applies the techniques to specific problems ecountered in probability. As before, we will distinguish between experiments that produce discrete outcomes from those that produce continuous outcomes.

We first define a *random variable* X as the *numerical outcome* of the random experiment. Typical examples are the number of dots on a die (discrete) or the distance of a dart from the center of a dartboard of radius one (continuous). The random variable X can take on the values in the set $\{1, 2, 3, 4, 5, 6\}$ for the first example and in the set $\{r : 0 \leq r \leq 1\}$ for the second example. We denote

the random variable by a *capital letter*, say X, and its possible *values* by a small letter, say x_i for the discrete case and x for the continuous case. The distinction is analogous to that between a function *defined* as $g(x) = x^2$ and the *values* $y = g(x)$ that $g(x)$ can take on.

Now it is of interest to determine various properties of X. To do so we use a computer simulation, performing many experiments and observing the outcome for each experiment. The number of experiments, which is sometimes referred to as the number of *trials*, will be denoted by M. To simulate a discrete random variable we use `rand`, which generates a number at random within the $(0, 1)$ interval (see Appendix 2A for some MATLAB basics). Assume that in general the possible values of X are $\{x_1, x_2, \ldots, x_N\}$ with probabilities $\{p_1, p_2, \ldots, p_N\}$. As an example, if $N = 3$ we can generate M values of X by using the following code segment (which assumes `M,x1,x2,x3,p1,p2,p3` have been previously assigned):

```
for i=1:M
  u=rand(1,1);
  if u<=p1
    x(i,1)=x1;
  elseif u>p1 & u<=p1+p2
    x(i,1)=x2;
  elseif u>p1+p2
    x(i,1)=x3;
  end
end
```

After this code is executed, we will have generated M values of the random variable X. Note that the values of X so obtained are termed the *outcomes* or *realizations* of X. The extension to any number N of possible values is immediate. For a continuous random variable X that is Gaussian we can use the code segment:

```
for i=1:M
  x(i,1)=randn(1,1);
end
```

or equivalently `x=randn(M,1)`. Again at the conclusion of this code segment we will have generated M realizations of X. Later we will see how to generate realizations of random variables whose PDFs are not Gaussian (see Section 10.9).

2.5 Determining Characteristics of Random Variables

There are many ways to characterize a random variable. We have already alluded to the probability of the outcomes in the discrete case and the PDF in the continuous case. To be more precise consider a discrete random variable, such as that describing the outcome of a coin toss. If we toss a coin and let X be 1 if a head is observed

and let X be 0 if a tail is observed, then the probabilities are defined to be p for $X = x_1 = 1$ and $1 - p$ for $X = x_2 = 0$. The probability p of $X = 1$ can be thought of as the relative frequency of the outcome of heads in a long succession of tosses. Hence, to determine the probability of heads we could toss a coin a large number of times and estimate p by the number of observed heads divided by the number of tosses. Using a computer to simulate this experiment, we might inquire as to the number of tosses that would be necessary to obtain an accurate estimate of the probability of heads. Unfortunately, this is not easily answered. A practical means, though, is to increase the number of tosses until the estimate so computed converges to a fixed number. A computer simulation is shown in Figure 2.5 where the estimate

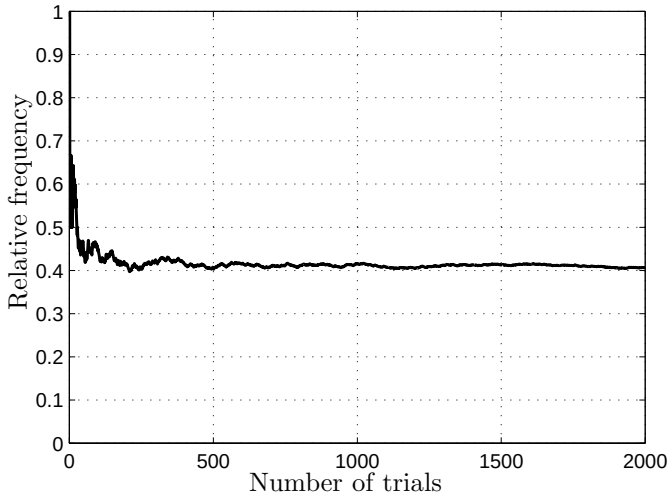


Figure 2.5: Estimate of probability of heads for various number of coin tosses.

appears to converge to about 0.4. Indeed, the true value (that value used in the simulation) was $p = 0.4$. It is also seen that the estimate of p is slightly higher than 0.4. This is due to the slight imperfections in the random number generator as well as computational errors. Increasing the number of trials will not improve the results. We next describe some typical simulations that will be useful to us. To illustrate the various simulations we will use a Gaussian random variable with realizations generated using `randn(1,1)`. Its PDF is shown in Figure 2.6.

Example 2.1 – Probability density function

A PDF may be estimated by first finding the histogram and then dividing the number of outcomes in each bin by M, the total number of realizations, to obtain the probability. Then to obtain the PDF $p_X(x)$ recall that the probability of X taking on a value in an interval is found as the area under the PDF of that interval

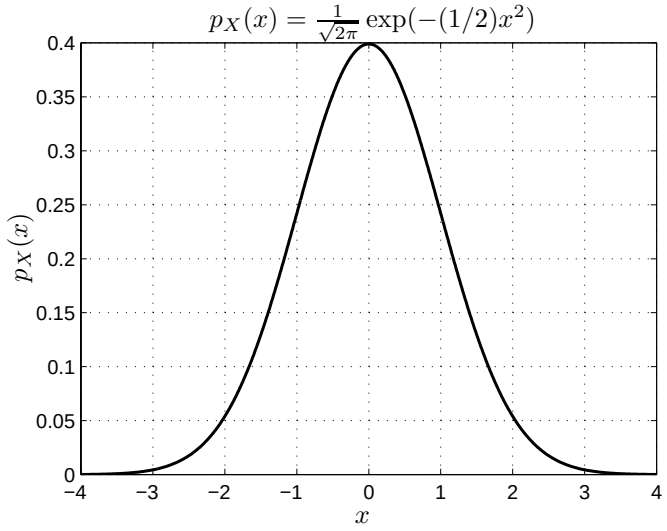


Figure 2.6: Gaussian probability density function.

(see Section 1.3). Thus,

$$P[a \leq X \leq b] = \int_a^b p_X(x)dx \tag{2.1}$$

and if $a = x_0 - \Delta x/2$ and $b = x_0 + \Delta x/2$, where Δx is small, then (2.1) becomes

$$P[x_0 - \Delta x/2 \leq X \leq x_0 + \Delta x/2] \approx p_X(x_0)\Delta x$$

and therefore the PDF at $x = x_0$ is approximately

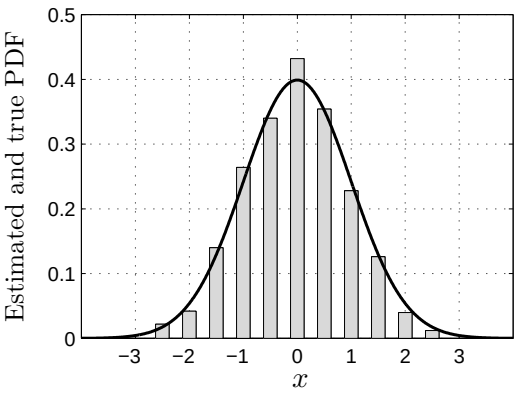
$$p_X(x_0) \approx \frac{P[x_0 - \Delta x/2 \leq X \leq x_0 + \Delta x/2]}{\Delta x}.$$

Hence, we need only divide the estimated probability by the bin width Δx. Also, note that as claimed in Chapter 1, $p_X(x)$ is seen to be the *probability per unit length*. In Figure 2.7 is shown the estimated PDF for a Gaussian random variable as well as the true PDF as given in Figure 2.6. The MATLAB code used to generate the figure is also shown.

♢

Example 2.2 – Probability of an interval

To determine $P[a \leq X \leq b]$ we need only generate M realizations of X, then count the number of outcomes that fall into the $[a, b]$ interval and divide by M. Of course




```
randn('state',0)
x=randn(1000,1);
bincenters=[-3.5:0.5:3.5]';
bins=length(bincenters);
h=zeros(bins,1);
for i=1:length(x)
  for k=1:bins
    if x(i)>bincenters(k)-0.5/2 ...
     & x(i)<=bincenters(k)+0.5/2
      h(k,1)=h(k,1)+1;
    end
  end
end
pxest=h/(1000*0.5);
xaxis=[-4:0.01:4]';
px=(1/sqrt(2*pi))*exp(-0.5*xaxis.^2);
```

Figure 2.7: Estimated and true probability density functions.

M should be large. In particular, if we let $a = 2$ and $b = \infty$, then we should obtain the value (which must be evaluated using numerical integration)

$$P[X > 2] = \int_2^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-(1/2)x^2\right) dx = 0.0228$$

and therefore very few realizations can be expected to fall in this interval. The results for an increasing number of realizations are shown in Figure 2.8. This illustrates the problem with the simulation of small probability events. It requires a large number of realizations to obtain accurate results. (See Problem 11.47 on how to reduce the number of realizations required.)

♢

Example 2.3 – Average value

It is frequently important to measure characteristics of X in addition to the PDF. For example, we might only be interested in the average or *mean* or *expected value* of X. If the random variable is Gaussian, then from Figure 2.6 we would expect X to be zero on the average. This conjecture is easily "verified" by using the *sample mean* estimate

$$\frac{1}{M} \sum_{i=1}^{M} x_i$$

M	Estimated $P[X > 2]$	True $P[X > 2]$
100	0.0100	0.0228
1000	0.0150	0.0228
10,000	0.0244	0.0288
100,000	0.0231	0.0288

```
randn('state',0)
M=100;count=0;
x=randn(M,1);
for i=1:M
  if x(i)>2
    count=count+1;
  end
end
probest=count/M
```

Figure 2.8: Estimated and true probabilities.

of the mean. The results are shown in Figure 2.9.

M	Estimated mean	True mean
100	0.0479	0
1000	−0.0431	0
10,000	0.0011	0
100,000	0.0032	0

```
randn('state',0)
M=100;
meanest=0;
x=randn(M,1);
for i=1:M
  meanest=meanest+(1/M)*x(i);
end
meanest
```

Figure 2.9: Estimated and true mean.

♢

Example 2.4 – A transformed random variable

One of the most important problems in probability is to determine the PDF for a transformed random variable, i.e., one that is a function of X, say X^2 as an example. This is easily accomplished by modifying the code in Figure 2.7 from `x=randn(1000,1)` to `x=randn(1000,1);x=x.^2;`. The results are shown in Figure 2.10. Note that the shape of the PDF is completely different than the original Gaussian shape (see Example 10.7 for the true PDF). Additionally, we can obtain the mean of X^2 by using

$$\frac{1}{M}\sum_{i=1}^{M} x_i^2$$

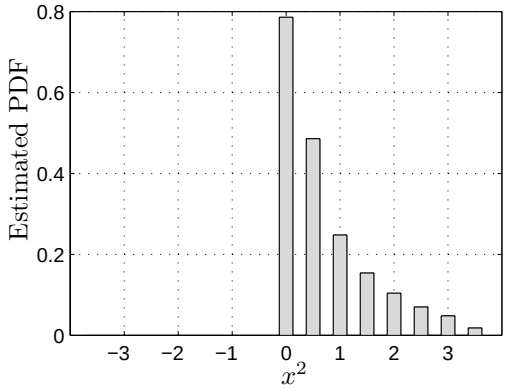


Figure 2.10: Estimated PDF of X^2 for X Gaussian.

as we did in Example 2.3. The results are shown in Figure 2.11.

M	Estimated mean	True mean
100	0.7491	1
1000	0.8911	1
10,000	1.0022	1
100,000	1.0073	1

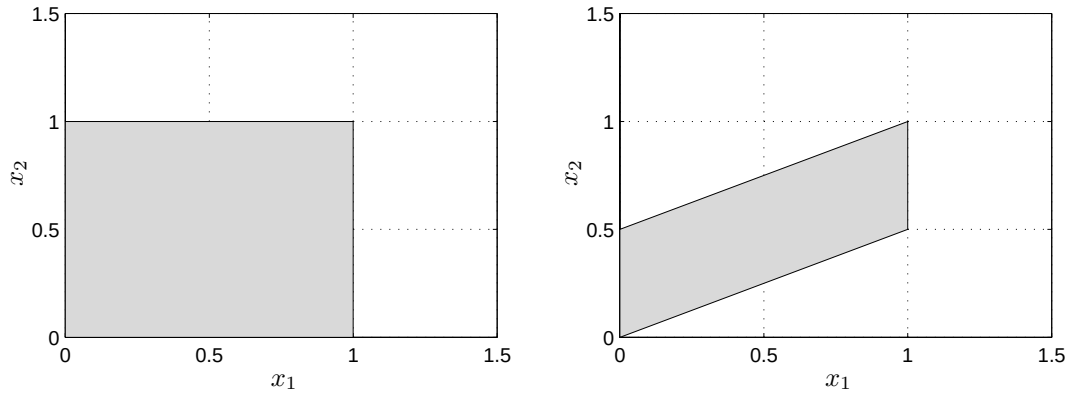
```
randn('state',0)
M=100;
meanest=0;
x=randn(M,1);
for i=1:M
  meanest=meanest+(1/M)*x(i)^2;
end
meanest
```

Figure 2.11: Estimated and true mean.

♢

Example 2.5 – Multiple random variables

Consider an experiment that yields two random variables or the *vector* random variable $[X_1\, X_2]^T$, where T denotes the transpose. An example might be the choice of a point in the square $\{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$ according to some procedure. This procedure may or may not cause the value of x_2 to depend on the value of x_1. For example, if the result of many repetitions of this experiment produced an even distribution of points indicated by the shaded region in Figure 2.12a, then we would say that there is no dependency between X_1 and X_2. On the other hand, if the points were evenly distributed within the shaded region shown in Figure 2.12b, then there is a strong dependency. This is because if, for example, $x_1 = 0.5$, then x_2 would have to lie in the interval $[0.25, 0.75]$. Consider next the random vector



(a) No dependency (b) Dependency

Figure 2.12: Relationships between random variables.

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$$

where each U_i is generated using `rand`. The result of $M = 1000$ realizations is shown in Figure 2.13a. We say that the random variables X_1 and X_2 are *independent*. Of course, this is what we expect from a good random number generator. If instead, we defined the new random variables,

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ \frac{1}{2}U_1 + \frac{1}{2}U_2 \end{bmatrix}$$

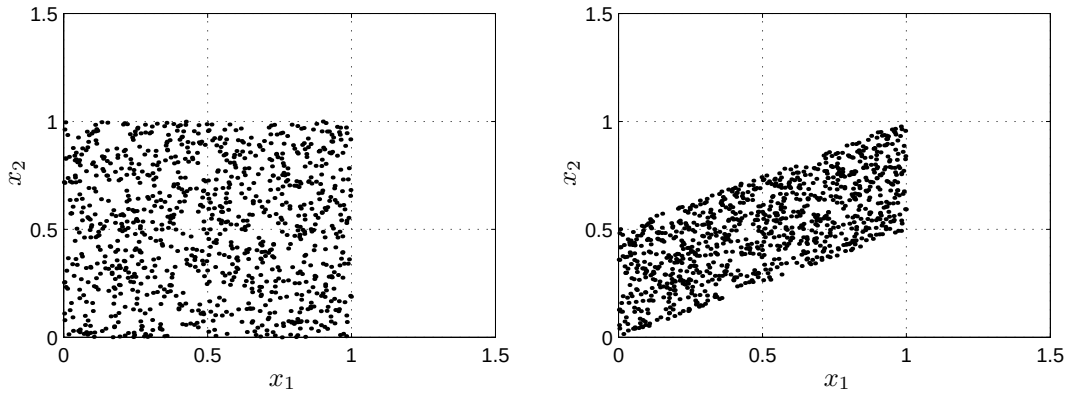
then from the plot shown in Figure 2.13b, we would say that the random variables are dependent. Note that this type of plot is called a *scatter diagram*.

♢

2.6 Real-World Example – Digital Communications

In a phase-shift keyed (PSK) digital communication system a binary digit (also termed a *bit*), which is either a "0" or a "1", is communicated to a receiver by sending either $s_0(t) = A\cos(2\pi F_0 t + \pi)$ to represent a "0" or $s_1(t) = A\cos(2\pi F_0 t)$ to represent a "1", where $A > 0$ [Proakis 1989]. The receiver that is used to decode the transmission is shown in Figure 2.14. The input to the receiver is the noise

corrupted signal or $x(t) = s_i(t) + w(t)$, where $w(t)$ represents the channel noise.



(a) No dependency (b) Dependency

Figure 2.13: Relationships between random variables.

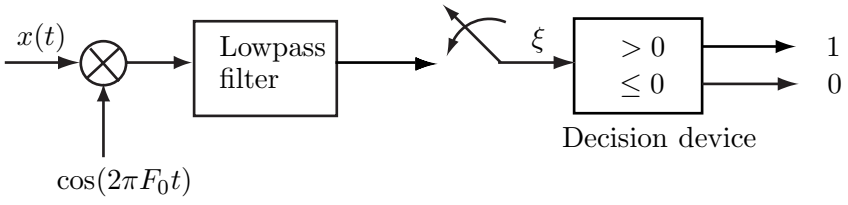


Figure 2.14: Receiver for a PSK digital communication system.

Ignoring the effect of noise for the moment, the output of the multiplier will be

$$\begin{aligned} s_0(t)\cos(2\pi F_0 t) &= A\cos(2\pi F_0 t + \pi)\cos(2\pi F_0 t) = -A\left(\frac{1}{2} + \frac{1}{2}\cos(4\pi F_0 t)\right) \\ s_1(t)\cos(2\pi F_0 t) &= A\cos(2\pi F_0 t)\cos(2\pi F_0 t) = A\left(\frac{1}{2} + \frac{1}{2}\cos(4\pi F_0 t)\right) \end{aligned}$$

for a 0 and 1 sent, respectively. After the lowpass filter, which filters out the $\cos(4\pi F_0 t)$ part of the signal, and sampler, we have

$$\xi = \begin{cases} -\frac{A}{2} & \text{for a 0} \\ \frac{A}{2} & \text{for a 1.} \end{cases}$$

The receiver decides a 1 was transmitted if $\xi > 0$ and a 0 if $\xi \leq 0$. To model the channel noise we assume that the actual value of ξ observed is

$$\xi = \begin{cases} -\frac{A}{2} + W & \text{for a 0} \\ \frac{A}{2} + W & \text{for a 1} \end{cases}$$

where W is a Gaussian random variable. It is now of interest to determine how the error depends on the signal amplitude A. Consider the case of a 1 having been

transmitted. Intuitively, if A is a large positive amplitude, then the chance that the noise will cause an error or equivalently, $\xi \leq 0$, should be small. This probability, termed the *probability of error* and denoted by P_e, is given by $P[A/2 + W \leq 0]$. Using a computer simulation we can plot P_e versus A with the result shown in Figure 2.15. Also, the true P_e is shown. (In Example 10.3 we will see how to analytically determine this probability.) As expected, the probability of error decreases as the

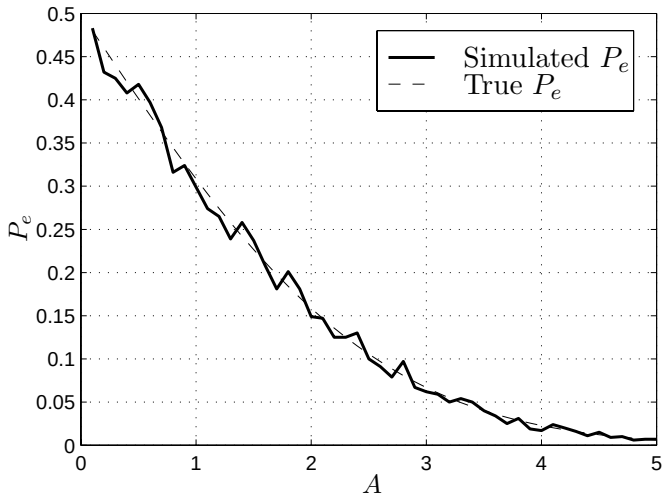


Figure 2.15: Probability of error for a PSK communication system.

signal amplitude increases. With this information we can design our system by choosing A to satisfy a given probability of error requirement. In actual systems this requirement is usually about $P_e = 10^{-7}$. Simulating this small probability would be exceedingly difficult due to the large number of trials required (but see also Problem 11.47). The MATLAB code used for the simulation is given in Figure 2.16.

References

Proakis, J., *Digitial Communications*, Second Ed., McGraw-Hill, New York, 1989.

Problems

Note: All the following problems require the use of a computer simulation. A realization of a *uniform* random variable is obtained by using `rand(1,1)` while a realization of a *Gaussian* random variable is obtained by using `randn(1,1)`.

```
A=[0.1:0.1:5]';
for k=1:length(A)
  error=0;
  for i=1:1000
    w=randn(1,1);
    if A(k)/2+w<=0
      error=error+1;
    end
  end
  Pe(k,1)=error/1000;
end
```

Figure 2.16: MATLAB code used to estimate the probability of error P_e in Figure 2.15.

2.1 (‿) **(c)** An experiment consists of tossing a fair coin twice. If a head occurs on the first toss, we let $x_1 = 1$ and if a tail occurs we let $x_1 = 0$. The same assignment is used for the outcome x_2 of the second toss. Defining the random variable as $Y = X_1X_2$, estimate the probabilities for the different possible values of Y. Explain your results.

2.2 (c) A pair of fair dice is tossed. Estimate the probability of "snake eyes" or a one for each die?

2.3 (‿) **(c)** Estimate $P[-1 \leq X \leq 1]$ if X is a Gaussian random variable. Verify the results of your computer simulation by numerically evaluating the integral

$$\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) dx.$$

Hint: See Problem 1.14.

2.4 (c) Estimate the PDF of the random variable

$$X = \sum_{i=1}^{12} \left(U_i - \frac{1}{2}\right)$$

where U_i is a uniform random variable. Then, compare this PDF to the Gaussian PDF or

$$p_X(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right).$$

2.5 (c) Estimate the PDF of $X = U_1 - U_2$, where U_1 and U_2 are uniform random variables. What is the most probable range of values?

2.6 (‿) (c) Estimate the PDF of $X = U_1 U_2$, where U_1 and U_2 are uniform random variables. What is the most probable range of values?

2.7 (c) Generate realizations of a discrete random variable X, which takes on values 1, 2, and 3 with probabilities $p_1 = 0.1$, $p_2 = 0.2$ and $p_3 = 0.7$, respectively. Next based on the generated realizations estimate the probabilities of obtaining the various values of X.

2.8 (‿) (c) Estimate the mean of U, where U is a uniform random variable. What is the true value?

2.9 (c) Estimate the mean of $X+1$, where X is a Gaussian random variable. What is the true value?

2.10 (c) Estimate the mean of X^2, where X is a Gaussian random variable.

2.11 (‿) (c) Estimate the mean of $2U$, where U is a uniform random variable. What is the true value?

2.12 (c) It is conjectured that if X_1 and X_2 are Gaussian random variables, then by subtracting them (let $Y = X_1 - X_2$), the probable range of values should be smaller. Is this true?

2.13 (‿) (c) A large circular dartboard is set up with a "bullseye" at the center of the circle, which is at the coordinate $(0, 0)$. A dart is thrown at the center but lands at (X, Y), where X and Y are two different Gaussian random variables. What is the average distance of the dart from the bullseye?

2.14 (‿) (c) It is conjectured that the mean of $\sqrt{U}$, where U is a uniform random variable, is $\sqrt{\text{mean of } U}$. Is this true?

2.15 (c) The Gaussian random variables X_1 and X_2 are linearly transformed to the new random variables

$$\begin{aligned} Y_1 &= X_1 + 0.1X_2 \\ Y_2 &= X_1 + 0.2X_2. \end{aligned}$$

Plot a scatter diagram for Y_1 and Y_2. Could you approximately determine the value of Y_2 if you knew that $Y_1 = 1$?

2.16 (c,w) Generate a scatter diagram for the linearly transformed random variables

$$\begin{aligned} X_1 &= U_1 \\ X_2 &= U_1 + U_2 \end{aligned}$$

where U_1 and U_2 are uniform random variables. Can you explain why the scatter diagram looks like a parallelogram? Hint: Define the vectors

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

and express $\mathbf{X}$ as a linear combination of $\mathbf{e}_1$ and $\mathbf{e}_2$.

Appendix 2A

Brief Introduction to MATLAB

A brief introduction to the scientific software package MATLAB is contained in this appendix. Further information is available at the web site www.mathworks.com. MATLAB is a scientific computation and data presentation language.

Overview of MATLAB

The chief advantage of MATLAB is its use of high-level instructions for matrix algebra and built-in routines for data processing. In this appendix as well as throughout the text a MATLAB command is indicated with the typewriter font such as `end`. MATLAB treats matrices of any size (which includes vectors and scalars as special cases) as *elements* and hence matrix multiplication is as simple as `C=A*B`, where `A` and `B` are conformable matrices. In addition to the usual matrix operations of addition `C=A+B`, multiplication `C=A*B`, and scaling by a constant `c` as `B=c*A`, certain matrix operators are defined that allow convenient manipulation. For example, assume we first define the column vector $\mathbf{x} = [1\,2\,3\,4]^T$, where T denotes transpose, by using `x=[1:4]'`. The vector starts with the element 1 and ends with the element 4 and the colon indicates that the intervening elements are found by incrementing the start value by one, which is the default. For other increments, say 0.5 we use `x=[1:0.5:4]'`. To define the vector $\mathbf{y} = [1^2\,2^2\,3^2\,4^2]^T$, we can use the matrix *element by element* exponentiation operator `.^` to form `y=x.^2` if `x=[1:4]'`. Similarly, the operators `.*` and `./` perform element by element multiplication and division of the matrices, respectively. For example, if

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Character	Meaning
+	addition (scalars, vectors, matrices)
-	subtraction (scalars, vectors, matrices)
*	multiplication (scalars, vectors, matrices)
/	division (scalars)
^	exponentiation (scalars, square matrices)
.*	element by element multiplication
./	element by element division
.^	element by element exponentiation
;	suppress printed output of operation
:	specify intervening values
'	conjugate transpose (transpose for real vectors, matrices)
...	line continuation (when command must be split)
%	remainder of line interpreted as comment
==	logical equals
|	logical or
&	logical and
~ =	logical not

Table 2A.1: Definition of common MATLAB characters.

then the statements `C=A.*B` and `D=A./B` produce the results

$$\mathbf{C} = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

respectively. A listing of some common characters is given in Table 2A.1. MATLAB has the usual built-in functions of `cos`, `sin`, etc. for the trigonometric functions, `sqrt` for a square root, `exp` for the exponential function, and `abs` for absolute value, as well as many others. When a function is applied to a matrix, the function is applied to each element of the matrix. Other built-in symbols and functions and their meanings are given in Table 2A.2.

Matrices and vectors are easily specified. For example, to define the column vector $\mathbf{c}_1 = [1\ 2]^T$, just use `c1=[1 2].'` or equivalently `c1=[1;2]`. To define the $\mathbf{C}$ matrix given previously, the construction `C=[1 4;9 16]` is used. Or we could first define $\mathbf{c}_2 = [4\ 16]^T$ by `c2=[4 16].'` and then use `C=[c1 c2]`. It is also possible to extract portions of matrices to yield smaller matrices or vectors. For example, to extract the first column from the matrix $\mathbf{C}$ use `c1=C(:,1)`. The colon indicates that all elements in the first column should be extracted. Many other convenient manipulations of matrices and vectors are possible.

Function	Meaning
`pi`	π
`i`	$\sqrt{-1}$
`j`	$\sqrt{-1}$
`round(x)`	rounds every element in **x** to the nearest integer
`floor(x)`	replaces every element in **x** by the nearest integer less than or equal to x
`inv(A)`	takes the inverse of the square matrix **A**
`x=zeros(N,1)`	assigns an $N \times 1$ vector of all zeros to `x`
`x=ones(N,1)`	assigns an $N \times 1$ vector of all ones to `x`
`x=rand(N,1)`	generates an $N \times 1$ vector of all uniform random variables
`x=randn(N,1)`	generates an $N \times 1$ vector of all Gaussian random variables
`rand('state',0)`	initializes uniform random number genetator
`randn('state',0)`	initializes Gaussian random number genetator
`M=length(x)`	sets M equal to N if `x` is $N \times 1$
`sum(x)`	sums all elements in vector **x**
`mean(x)`	computes the sample mean of the elements in **x**
`flipud(x)`	flips the vector **x** upside down
`abs`	takes the absolute value (or complex magnitude) of every element of **x**
`fft(x,N)`	computes the FFT of length N of **x** (zero pads if `N>length(x)`)
`ifft(x,N)`	computes the inverse FFT of length N of **x**
`fftshift(x)`	interchanges the two halves of an FFT output
`pause`	pauses the execution of a program
`break`	terminates a loop when encountered
`whos`	lists all variables and their attributes in current workspace
`help`	provides help on commands, e.g., `help sqrt`

Table 2A.2: Definition of useful MATLAB symbols and functions.

Any vector that is generated whose dimensions are not explicitly specified is assumed to be a *row* vector. For example, if we say `x=ones(10)`, then it will be designated as the 1×10 row vector consisting of all ones. To yield a column vector use `x=ones(10,1)`.

Loops are implemented with the construction

```
for k=1:10
  x(k,1)=1;
end
```

which is equivalent to `x=ones(10,1)`. Logical flow can be accomplished with the construction

```
if x>0
  y=sqrt(x);
else
  y=0;
end
```

Finally, a good practice is to begin each program or script, which is called an "m" file (due to its syntax, for example, `pdf.m`), with a `clear all` command. This will clear all variables in the workspace since otherwise, the current program may inadvertently (on the part of the programmer) use previously stored variable data.

Plotting in MATLAB

Plotting in MATLAB is illustrated in the next section by example. Some useful functions are summarized in Table 2A.3.

Function	Meaning
`figure`	opens up a new figure window
`plot(x,y)`	plots the elements of **x** versus the elements of **y**
`plot(x1,y1,x2,y2)`	same as above except multiple plots are made
`plot(x,y,'.')`	same as `plot` except the points are not connected
`title('my plot')`	puts a title on the plot
`xlabel('x')`	labels the x axis
`ylabel('y')`	labels the y axis
`grid`	draws grid on the plot
`axis([0 1 2 4])`	plots only the points in range $0 \leq x \leq 1$ and $2 \leq y \leq 4$
`text(1,1,'curve 1')`	places the text "curve 1" at the point (1,1)
`hold on`	holds current plot
`hold off`	releases current plot

Table 2A.3: Definition of useful MATLAB plotting functions.

An Example Program

A complete MATLAB program is given below to illustrate how one might compute the samples of several sinusoids of different amplitudes. It also allows the sinusoids to be clipped. The sinusoid is $s(t) = A\cos(2\pi F_0 t + \pi/3)$, with $A = 1$, $A = 2$, and $A = 4$, $F_0 = 1$, and $t = 0, 0.01, 0.02, \ldots, 10$. The clipping level is set at ± 3, i.e., any sample above $+3$ is clipped to $+3$ and any sample less than -3 is clipped to -3.

```
%  matlabexample.m
%
%  This program computes and plots samples of a sinusoid
%  with amplitudes 1, 2, and 4.  If desired, the sinusoid can be
%  clipped to simulate the effect of a limiting device.
%  The frequency is 1 Hz and the time duration is 10 seconds.
%  The sample interval is 0.1 seconds. The code is not efficient but
%  is meant to illustrate MATLAB statements.
%
clear all % clear all variables from workspace
delt=0.01; % set sampling time interval
F0=1; % set frequency
t=[0:delt:10]'; % compute time samples 0,0.01,0.02,...,10
A=[1 2 4]'; % set amplitudes
clip='yes'; % set option to clip
for i=1:length(A) % begin computation of sinusoid samples
   s(:,i)=A(i)*cos(2*pi*F0*t+pi/3); % note that samples for sinusoid
                                    % are computed all at once and
                                    % stored as columns in a matrix
   if clip=='yes' % determine if clipping desired
      for k=1:length(s(:,i)) % note that number of samples given as
                             % dimension of column using length command
         if s(k,i)>3 % check to see if sinusoid sample exceeds 3
            s(k,i)=3; % if yes, then clip
         elseif s(k,i)<-3 % check to see if sinusoid sample is less
            s(k,i)=-3;    % than -3 if yes, then clip
         end
      end
   end
end
figure % open up a new figure window
plot(t,s(:,1),t,s(:,2),t,s(:,3)) % plot sinusoid samples versus time
                                 % samples for all three sinusoids
grid % add grid to plot
xlabel('time, t') % label x-axis
```

```
ylabel('s(t)') % label y-axis
axis([0 10 -4 4]) % set up axes using axis([xmin xmax ymin ymax])
legend('A=1','A=2','A=4') % display a legend to distinguish
                          % different sinusoids
```

The output of the program is shown in Figure 2A.1. Note that the different graphs will appear as different colors.

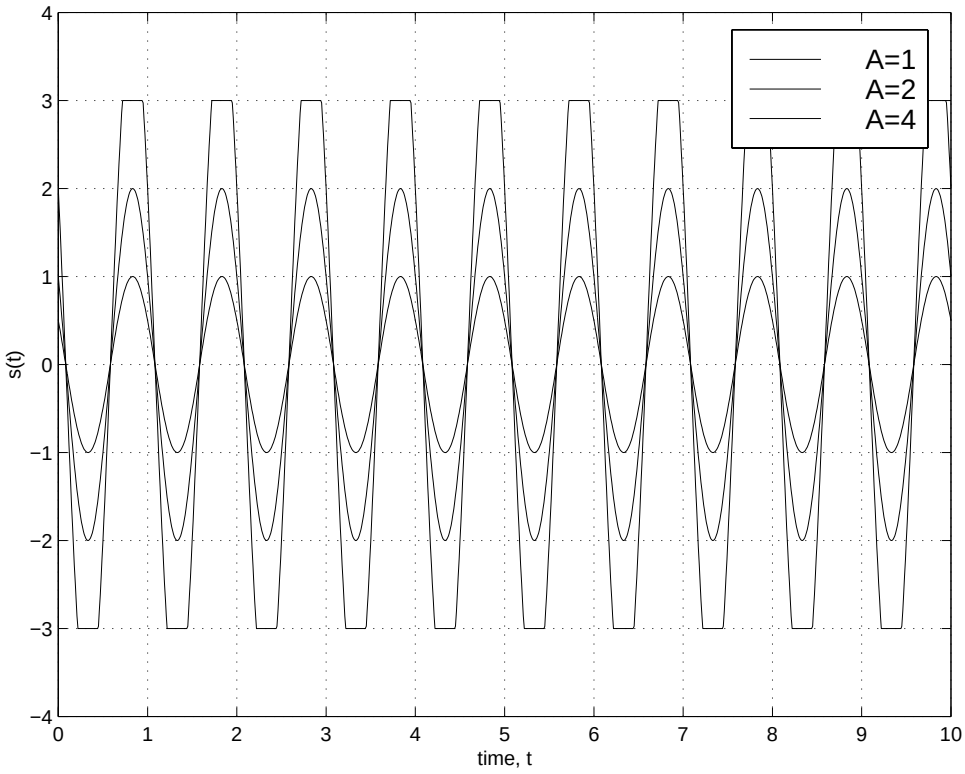


Figure 2A.1: Output of MATLAB program `matlabexample.m`.