8. You can download another file using the **get** command or you can change back to the previous directory by typing **cd** followed by two periods.
9. When finished downloading all of the desired files, type **bye**.

To download the files from the FTP site to your Macintosh computer, a variety of FTP programs are available. The steps given below are for the **Fetch** program from Dartmouth College.

1. Open the FTP program.
2. Enter the following information in the **New Connection** dialog box. If you do not see this dialog box, open it by choosing **New Connection** from the **File** menu.

   `Host:` **iplserv.ece.ucsb.edu**
   `User ID:` **anonymous**
   `Password:` **Your E-mail address**
   `Directory` **/pub/mitra/Labs/mac**

3. Choose **OK**. You should see a window with the contents of the directory of the FTP site. You can change directories simply by double-clicking on the folder you want.
4. To transfer a file, select the file you want by double-clicking on it. A dialog box will pop up. Select the place on your computer where you want to store the file and choose **Save**. In recent versions of the **Fetch** program, entire directories can be downloaded at once by selecting the directory name and choosing **Get**.

For downloading via world wide web, follow the steps given below:

1. Open the available web browser.
2. Type **ftp://iplserv.ece.ucsb.edu** in the URL window.
3. Double-click on the desired directory (the directory for the PC and Macintosh versions are shown above).
4. Double-click on the desired file for downloading. You will get a dialog box asking where you would like to save the file.

1.3 Background Review

R1.1 A discrete-time signal is represented as a sequence of numbers, called *samples* . A sample value of a typical discrete-time signal or sequence $\{x[n]\}$ is denoted as $x[n]$ with the argument n being an integer in the range $-\infty$ and ∞. For convenience, the sequence $\{x[n]\}$ is often denoted without the curly brackets.

R1.2 The discrete-time signal may be a finite length or an infinite length sequence. A finite length (also called *finite duration* or *finite extent*) sequence is defined only for a finite

time interval:

$$N_1 \leq n \leq N_2, \tag{1.1}$$

where $-\infty < N_1$ and $N_2 < \infty$ with $N_2 \geq N_1$. The length or duration N of the finite length sequence is

$$N = N_2 - N_1 + 1. \tag{1.2}$$

R1.3 A sequence $\tilde{x}[n]$ satisfying

$$\tilde{x}[n] = \tilde{x}[n + kN] \qquad \text{for all } n, \tag{1.3}$$

is called a *periodic sequence* with a period N where N is a positive integer and k is any integer.

R1.4 The *energy* of a sequence $x[n]$ is defined by

$$\mathcal{E} = \sum_{n=-\infty}^{\infty} |x[n]|^2. \tag{1.4}$$

The energy of a sequence over a finite interval $-K \leq n \leq K$ is defined by

$$\mathcal{E}_K = \sum_{n=-K}^{K} |x[n]|^2. \tag{1.5}$$

R1.5 The *average power* of an aperiodic sequence $x[n]$ is defined by

$$\mathcal{P}_{av} = \lim_{K\to\infty} \frac{1}{2K+1}\mathcal{E}_K = \lim_{K\to\infty} \frac{1}{2K+1} \sum_{n=-K}^{K} |x[n]|^2. \tag{1.6}$$

The average power of a periodic sequence $\tilde{x}[n]$ with a period N is given by

$$\mathcal{P}_{av} = \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2. \tag{1.7}$$

R1.6 The *unit sample sequence* , often called the *discrete-time impulse* or the *unit impulse*, denoted by $\delta[n]$, is defined by

$$\delta[n] = \begin{cases} 1, & \text{for } n = 0, \\ 0, & \text{for } n \neq 0. \end{cases} \tag{1.8}$$

The *unit step sequence* , denoted by $\mu[n]$, is defined by

$$\mu[n] = \begin{cases} 1, & \text{for } n \geq 0, \\ 0, & \text{for } n < 0. \end{cases} \tag{1.9}$$

R1.7 The *exponential sequence* is given by

$$x[n] = A\alpha^n, \tag{1.10}$$

where A and α are real or complex numbers. By expressing

$$\alpha = e^{(\sigma_o + j\omega_o)}, \text{ and } A = |A|e^{j\phi},$$

we can rewrite Eq. (1.10) as

$$x[n] = |A|e^{\sigma_o n + j(\omega_o n + \phi)} = |A|e^{\sigma_o n}\cos(\omega_o n + \phi) + j|A|e^{\sigma_o n}\sin(\omega_o n + \phi). \tag{1.11}$$

R1.8 The *real sinusoidal sequence* with a constant amplitude is of the form

$$x[n] = A\cos(\omega_o n + \phi), \tag{1.12}$$

where A, ω_o, and ϕ are real numbers. The parameters A, ω_o, and ϕ in Eqs. (1.11) and (1.12) are called, respectively, the *amplitude* , the *angular frequency* , and the *initial phase* of the sinusoidal sequence $x[n]$. $f_o = \omega_o/2\pi$ is the *frequency*.

R1.9 The complex exponential sequence of Eq. (1.11) with $\sigma_o = 0$ and the sinusoidal sequence of Eq. (1.12) are periodic sequences if $\omega_o N$ is an integer multiple of 2π, that is,

$$\omega_o N = 2\pi r, \tag{1.13}$$

where N is a positive integer and r is any integer. The smallest possible N satisfying this condition is the *period* of the sequence.

R1.10 The *product* of two sequences $x[n]$ and $h[n]$ of length N yields a sequence $y[n]$, also of length N, as given by

$$y[n] = x[n] \cdot h[n]. \tag{1.14}$$

The *addition* of two sequences $x[n]$ and $h[n]$ of length N yields a sequence $y[n]$, also of length N, as given by

$$y[n] = x[n] + h[n]. \tag{1.15}$$

The *multiplication* of a sequence $x[n]$ of length N by a scalar A results in a sequence $y[n]$ of length N as given by

$$y[n] = A \cdot x[n]. \tag{1.16}$$

The *time-reversal* of a sequence $x[n]$ of infinite length results in a sequence $y[n]$ of infinite length as defined by

$$y[n] = x[-n]. \tag{1.17}$$

The *delay* of a sequence $x[n]$ of infinite length by a positive integer M results in a sequence $y[n]$ of infinite length given by

$$y[n] = x[n - M]. \tag{1.18}$$

If M is a negative integer, the operation indicated in Eq. (1.18) results in an *advance* of the sequence $x[n]$.

A sequence $x[n]$ of length N can be *appended* by another sequence $g[n]$ of length M resulting in a longer sequence $y[n]$ of length $N + M$ given by

$$\{y[n]\} = \{\{x[n]\}, \{g[n]\}\}. \tag{1.19}$$

1.4 MATLAB Commands Used

The MATLAB commands you will encounter in this exercise are as follows:

Operators and Special Characters

```
:    .    +    -    *    /    ;
%
```

Elementary Matrices and Matrix Manipulation

```
i    ones    pi    rand    randn    zeros
```

Elementary Functions

```
cos    exp    imag    real
```

Data Analysis

```
sum
```

Two-Dimensional Graphics

```
axis    grid    legend    plot    stairs
stem    title   xlabel    ylabel
```

General Purpose Graphics Functions

```
clf    subplot
```

Signal Processing Toolbox

```
sawtooth    square
```

For additional information on these commands, see the *MATLAB Reference Guide* [Mat94] and the *Signal Processing Toolbox User's Guide* [Mat96] or type `help commandname` in the Command window. A brief explanation of the MATLAB functions used here can be found in Appendix B.

1.5 Generation of Sequences

The purpose of this section is to familiarize you with the basic commands in MATLAB for signal generation and for plotting the generated signal. MATLAB has been designed to operate on data stored as vectors or matrices. For our purposes, sequences will be stored as vectors. Therefore, all signals are limited to being causal and of finite length . The steps to follow to execute the programs listed in this book depend on the platform being used to run the MATLAB.

MATLAB on the Windows PC

The program can be executed by typing the name of the program without `.m` in the Command window and hitting the carriage return. Alternately, choose **Open** from the **File** menu in the Command window and choose the desired M-file. This opens the M-file in the Editor/Debugger window in which an M-file can be executed using the **Run** command under the **Tools** menu.

MATLAB on the Macintosh

The program can be executed by typing the name of the program without `.m` in the Command window and hitting the carriage return. Alternately, it can be copied into the Editor Window by using the **Open M-File** command on your screen and then choosing the **Save and Execute** command on your screen.

Project 1.1 Unit Sample and Unit Step Sequences

Two basic discrete-time sequences are the unit sample sequence and the unit step sequence of Eqs. (1.8) and (1.9), respectively. A unit sample sequence `u[n]` of length `N` can be generated using the MATLAB command

```
u = [1    zeros(1,N-1)];
```

A unit sample sequence `ud[n]` of length `N` and delayed by `M` samples, where `M < N`, can be generated using the MATLAB command

```
ud = [zeros(1,M)    1    zeros(1,N-M-1)];
```

Likewise, a unit step sequence `s[n]` of length `N` can be generated using the MATLAB command

```
s = [ones(1,N)];
```

A delayed unit step sequence can be generated in a manner similar to that used in the generation of a delayed unit sample sequence.

Program P1_1 can be used to generate and plot a unit sample sequence.

```
% Program P1_1
% Generation of a Unit Sample Sequence
clf;
% Generate a vector from -10 to 20
n = -10:20;
% Generate the unit sample sequence
u = [zeros(1,10) 1 zeros(1,20)];
% Plot the unit sample sequence
stem(n,u);
xlabel('Time index n');ylabel('Amplitude');
title('Unit Sample Sequence');
axis([-10 20 0 1.2]);
```

Questions:

Q1.1 Run Program P1_1 to generate the unit sample sequence `u[n]` and display it.

Q1.2 What are the purposes of the commands `clf`, `axis`, `title`, `xlabel`, and `ylabel`?

Q1.3 Modify Program P1_1 to generate a delayed unit sample sequence `ud[n]` with a delay of 11 samples. Run the modified program and display the sequence generated.

Q1.4 Modify Program P1_1 to generate a unit step sequence `s[n]`. Run the modified program and display the sequence generated.

Q1.5 Modify Program P1_1 to generate a delayed unit step sequence `sd[n]` with an advance of 7 samples. Run the modified program and display the sequence generated.

Project 1.2 Exponential Signals

Another basic discrete-time sequence is the exponential sequence. Such a sequence can be generated using the MATLAB operators `.^` and `exp`.

Program P1_2 given below can be employed to generate a complex-valued exponential sequence.

```
% Program P1_2
% Generation of a complex exponential sequence
clf;
c = -(1/12)+(pi/6)*i;
K = 2;
n = 0:40;
x = K*exp(c*n);
subplot(2,1,1);
stem(n,real(x));
xlabel('Time index n');ylabel('Amplitude');
title('Real part');
subplot(2,1,2);
stem(n,imag(x));
xlabel('Time index n');ylabel('Amplitude');
title('Imaginary part');
```

Program P1_3 given below can be employed to generate a real-valued exponential sequence.

```
% Program P1_3
% Generation of a real exponential sequence
clf;
n = 0:35; a = 1.2; K = 0.2;
x = K*a.^n;
```

```
stem(n,x);
xlabel('Time index n');ylabel('Amplitude');
```

Questions:

Q1.6 Run Program P1_2 and generate the complex-valued exponential sequence.

Q1.7 Which parameter controls the rate of growth or decay of this sequence? Which parameter controls the amplitude of this sequence?

Q1.8 What will happen if the parameter `c` is changed to `(1/12)+(pi/6)*i`?

Q1.9 What are the purposes of the operators `real` and `imag`?

Q1.10 What is the purpose of the command `subplot`?

Q1.11 Run Program P1_3 and generate the real-valued exponential sequence.

Q1.12 Which parameter controls the rate of growth or decay of this sequence? Which parameter controls the amplitude of this sequence?

Q1.13 What is the difference between the arithmetic operators `^` and `.^`?

Q1.14 What will happen if the parameter `a` is less than 1? Run Program P1_3 again with the parameter `a` changed to `0.9` and the parameter `K` changed to 20.

Q1.15 What is the length of this sequence and how can it be changed?

Q1.16 You can use the MATLAB command `sum(s.*s)` to compute the energy of a real sequence `s[n]` stored as a vector `s`. Evaluate the energy of the real-valued exponential sequences `x[n]` generated in Questions Q1.11 and Q1.14.

Project 1.3 Sinusoidal Sequences

Another very useful class of discrete-time signals is the real sinusoidal sequence of the form of Eq. (1.12). Such sinusoidal sequences can be generated in MATLAB using the trigonometric operators `cos` and `sin`.

Program P1_4 is a simple example that generates a sinusoidal signal.

```
% Program P1_4
% Generation of a sinusoidal sequence
n = 0:40;
f = 0.1;
phase = 0;
A = 1.5;
arg = 2*pi*f*n - phase;
x = A*cos(arg);
clf; % Clear old graph
stem(n,x); % Plot the generated sequence
```

```
axis([0 40 -2 2]);
grid;
title('Sinusoidal Sequence');
xlabel('Time index n');
ylabel('Amplitude');
axis;
```

Questions:

Q1.17 Run Program P1_4 to generate the sinusoidal sequence and display it.

Q1.18 What is the frequency of this sequence and how can it be changed? Which parameter controls the phase of this sequence? Which parameter controls the amplitude of this sequence? What is the period of this sequence?

Q1.19 What is the length of this sequence and how can it be changed?

Q1.20 Compute the average power of the generated sinusoidal sequence.

Q1.21 What are the purposes of the `axis` and `grid` commands?

Q1.22 Modify Program P1_4 to generate a sinusoidal sequence of frequency `0.9` and display it. Compare this new sequence with the one generated in Question Q1.17. Now, modify Program P1_4 to generate a sinusoidal sequence of frequency `1.1` and display it. Compare this new sequence with the one generated in Question Q1.17. Comment on your results.

Q1.23 Modify the above program to generate a sinusoidal sequence of length 50, frequency `0.08`, amplitude `2.5`, and phase shift 90 degrees and display it. What is the period of this sequence?

Q1.24 Replace the `stem` command in Program P1_4 with the `plot` command and run the program again. What is the difference between the new plot and the one generated in Question Q1.17?

Q1.25 Replace the `stem` command in Program P1_4 with the `stairs` command and run the program again. What is the difference between the new plot and those generated in Questions Q1.17 and Q1.24?

Project 1.4 Random Signals

A random signal of length `N` with samples uniformly distributed in the interval `(0,1)` can be generated by using the MATLAB command

```
x = rand(1,N);
```

Likewise, a random signal `x[n]` of length `N` with samples normally distributed with zero mean and unity variance can be generated by using the following MATLAB command

```
x = randn(1,N);
```

Questions:

Q1.26 Write a MATLAB program to generate and display a random signal of length 100 whose elements are uniformly distributed in the interval $[-2, 2]$.

Q1.27 Write a MATLAB program to generate and display a Gaussian random signal of length 75 whose elements are normally distributed with zero mean and a variance of 3.

Q1.28 Write a MATLAB program to generate and display five sample sequences of a random sinusoidal signal of length 31

$$\{X[n]\} = \{A \cdot \cos(\omega_o n + \phi)\}, \tag{1.20}$$

where the amplitude A and the phase ϕ are statistically independent random variables with uniform probability distribution in the range $0 \leq A \leq 4$ for the amplitude and in the range $0 \leq \phi \leq 2\pi$ for the phase.

1.6 Simple Operations on Sequences

As indicated earlier, the purpose of digital signal processing is to generate a signal with more desirable properties from one or more given discrete-time signals. The processing algorithm consists of performing a combination of basic operations such as addition, scalar multiplication, time-reversal, delaying, and product operation (see R1.10). We consider here three very simple examples to illustrate the application of such operations .

Project 1.5 Signal Smoothing

A common example of a digital signal processing application is the removal of the noise component from a signal corrupted by additive noise. Let $s[n]$ be the signal corrupted by a random noise $d[n]$ resulting in the noisy signal $x[n] = s[n] + d[n]$. The objective is to operate on $x[n]$ to generate a signal $y[n]$ which is a reasonable approximation to $s[n]$. To this end, a simple approach is to generate an output sample by averaging a number of input samples around the sample at instant n. For example, a three-point moving average algorithm is given by

$$y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1]). \tag{1.21}$$

Program P1_5 implements the above algorithm.

```
% Program P1_5
% Signal Smoothing by Averaging
clf;
R = 51;
d = 0.8*(rand(R,1) - 0.5); % Generate random noise
m = 0:R-1;
s = 2*m.*(0.9.^m); % Generate uncorrupted signal
x = s + d'; % Generate noise corrupted signal
```

```
subplot(2,1,1);
plot(m,d','r-',m,s,'g--',m,x,'b-.');
xlabel('Time index n');ylabel('Amplitude');
legend('d[n] ','s[n] ','x[n] ');
x1 = [0 0 x];x2 = [0 x 0];x3 = [x 0 0];
y = (x1 + x2 + x3)/3;
subplot(2,1,2);
plot(m,y(2:R+1),'r-',m,s,'g--');
legend('y[n] ','s[n] ');
xlabel('Time index n');ylabel('Amplitude');
```

Questions:

Q1.29 Run Program P1_5 and generate all pertinent signals.

Q1.30 What is the form of the uncorrupted signal `s[n]`? What is the form of the additive noise `d[n]`?

Q1.31 Can you use the statement `x = s + d` to generate the noise-corrupted signal? If not, why not?

Q1.32 What are the relations between the signals `x1`, `x2`, and `x3`, and the signal `x`?

Q1.33 What is the purpose of the `legend` command?

Project 1.6 Generation of Complex Signals

More complex signals can be generated by performing the basic opertations on simple signals. For example, an *amplitude modulated signal* can be generated by modulating a high-frequency sinusoidal signal $x_H[n] = \cos(\omega_H n)$ with a low-frequency modulating signal $x_L[n] = \cos(\omega_L n)$. The resulting signal $y[n]$ is of the form

$$y[n] = A(1 + m \cdot x_L[n])x_H[n] = A(1 + m \cdot \cos(\omega_L n)) \cos(\omega_H n),$$

where m, called the *modulation index* , is a number chosen to ensure that $(1 + m \cdot x_L[n])$ is positive for all n. Program P1_6 can be used to generate an amplitude modulated signal.

```
% Program P1_6
% Generation of amplitude modulated sequence
clf;
n = 0:100;
m = 0.4;fH = 0.1; fL = 0.01;
xH = sin(2*pi*fH*n);
xL = sin(2*pi*fL*n);
y = (1+m*xL).*xH;
stem(n,y);grid;
xlabel('Time index n');ylabel('Amplitude');
```

Questions:

Q1.34 Run Program P1_6 and generate the amplitude modulated signal `y[n]` for various values of the frequencies of the carrier signal `xH[n]` and the modulating signal `xL[n]`, and various values of the modulation index `m`.

Q1.35 What is the difference between the arithmetic operators `*` and `.*` ?

As the frequency of a sinusoidal signal is the derivative of its phase with respect to time, to generate a swept-frequency sinusoidal signal whose frequency increases linearly with time, the argument of the sinusoidal signal must be a quadratic function of time. Assume that the argument is of the form $an^2 + bn$ (i.e. the angular frequency is $2an + b$). Solve for the values of a and b from the given conditions (minimum angular frequency and maximum angular frequency). Program P1_7 is an example program to generate this kind of signal.

```
% Program P1_7
% Generation of a swept frequency sinusoidal sequence
n = 0:100;
a = pi/2/100;
b = 0;
arg = a*n.*n + b*n;
x = cos(arg);
clf;
stem(n, x);
axis([0,100,-1.5,1.5]);
title('Swept-Frequency Sinusoidal Signal');
xlabel('Time index n');
ylabel('Amplitude');
grid; axis;
```

Questions:

Q1.36 Run Program P1_7 and generate the swept-frequency sinusoidal sequence `x[n]`.

Q1.37 What are the minimum and maximum frequencies of this signal?

Q1.38 How can you modify the above program to generate a swept sinusoidal signal with a minimum frequency of `0.1` and a maximum frequency of `0.3`?

1.7 Workspace Information

The commands `who` and `whos` can be used to get information about the variables stored in the workspace and their sizes created in running various MATLAB programs at any time.

Questions:

Q1.39 Type `who` in the Command window. What information is displayed in the Command window as a result of this command?

Q1.40 Type `whos` in the Command window. What information is displayed in the Command window as a result of this command?

1.8 Other Types of Signals (Optional)

Project 1.7 Squarewave and Sawtooth Signals

MATLAB functions `square` and `sawtooth` can be used to generate sequences of the types shown in Figures 1.1 and 1.2, respectively.

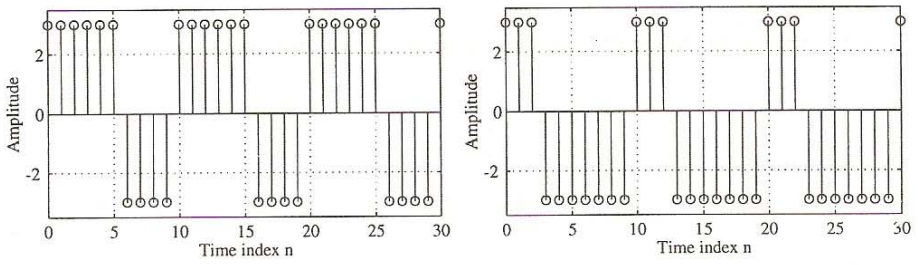

Figure 1.1 Square wave sequences.

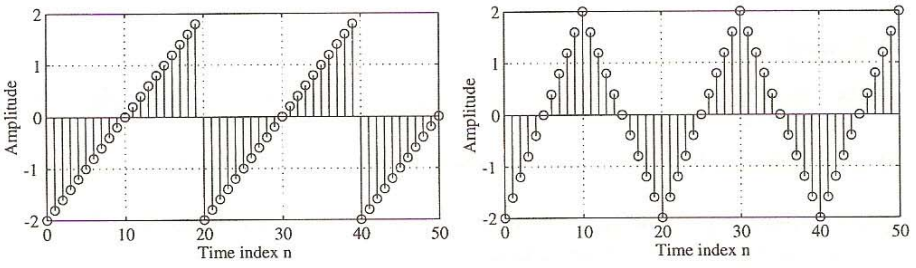

Figure 1.2 Sawtooth wave sequences.

Question:

Q1.41 Write MATLAB programs to generate the square-wave and the sawtooth wave sequences of the types shown in Figures 1.1 and 1.2. Using these programs, generate and plot the sequences.

1.9 Background Reading

[1] E. Cunningham. *Digital Filtering: An Introduction.* Houghton-Mifflin, Boston MA, 1992. Secs. 1.2–1.3.

[2] D. J. DeFatta, J. G. Lucas, and W. S. Hodgkiss. *Digital Signal Processing: A System Design Approach.* Wiley, New York NY, 1988. Secs. 2.1.2–2.1.4.

[3] L. B. Jackson. *Digital Filters and Signal Processing.* Kluwer, Boston MA, third edition, 1996. Secs. 2.2–2.3.

[4] R. Kuc. *Introduction to Digital Signal Processing.* McGraw-Hill, New York NY, 1988. Secs. 2-2, 2-4.

[5] L. C. Ludeman. *Fundamentals of Digital Signal Processing.* Harper & Row, New York NY, 1986. Sec. 1.2.

[6] S. K. Mitra. *Digital Signal Processing: A Computer-Based Approach.* McGraw-Hill, New York NY, 1998. Secs. 2.4–2.5.

[7] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs NJ, 1989. Sec. 2.1.

[8] B. Porat. *A Course in Digital Signal Procesing.* Wiley, New York NY, 1996. Sec. 2.7–2.8.

[9] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs NJ, 1996. Secs. 2.2–2.4.

[10] R. A. Roberts and C. T. Mullis. *Digital Signal Processing*, Addison-Wesley, Reading MA, 1987. Sec. 2.2.