Chapter 1

Discrete-Time Signals and Systems

OBJECTIVE: The principle objective of this experiment of familiarize the student with matlab using some elementary DSP principles.

Student is referred to Chapter 2 of Oppenheim and Schafer for the theoretical background of any material in this experiment. This experiment is divided into two sections.

- **Properties of Systems**
- **Constant Coefficient Difference Equations**

Properties of Systems

A discrete-time system can be thought of as a transformation of an input sequence $x(n)$ to an output sequence $y(n)$ by some operator T as depicted in Figure 1.1, The operator $T[\cdot]$ can belong

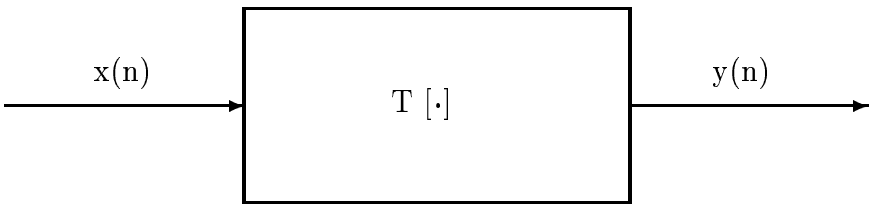

Figure 1.1: Input-Output of System defined by operator T

to a large class of systems depending on the conditions placed on it, i.e, linear time-invariant, non-linear,etc.

In this part of the experiment, it is required to experimentally determine four important properties of systems namely, 1) BIBO stability; 2) Causality; 3) linearity and 4) Shift-Invariance. You are given six unknown systems, one in each of the six matlab functions named, $func\_1$, $func\_2$,..., $func\_6$. By generating input test sequences for each of the systems, you must determine

whether each of the above four properties of the systems are valid or not. Some examples of possible input sequences (and their combinations) that may be used are:

- **Unit Sample**

$$x_1[n] = \delta(n - n_a), \qquad n_a = 2$$

- **Unit Step**

$$x_2[n] = U(n - n_b), \qquad n_b = 2$$

- **Real Exponential**

$$x_3[n] = exp^{-\alpha|n|}, \qquad \alpha = .2$$

- **Sinusoid**

$$x_4[n] = sin(2\pi\beta n), \qquad \beta = .12$$

The following example shows how one may test whether an arbitrary system, (i.e, $func\_?$) is linear.

1. Generate input test sequence of length N,
   $n = 1 : N;$
   $x_1 = sin(2\pi f_1 n);$ % f_1 and f_2 are normalized frequencies, $0 \leq f_1, f_2 \leq .5$
   $x_2 = sin(2\pi f_2 n);$
2. Generate the following outputs of the system,
   $y_1 = func\_?(ax_1)$ % a & b are arbitrary constants
   $y_2 = func\_?(bx_2)$
   $y_3 = func\_?(ax_1 + bx_2)$
3. From y_1, y_2 and y_3 determine whether the system is linear or not.

Problem 1

For the systems defined by each of the six given functions, determine the above four properties. Justify your answer and include any relevant graphs.
Note: To plot graphs see for example the functions **PLOT, STEM, SUBPLOT, POLAR, AXIS, TEXT, XLABEL, YLABEL ...**

Problem 2

Using a sinusoidal input $x[n]$ for the function $func\_2$, generate the output function $y[n]$. Now determine the Fourier Transform (FT) of $x[n]$ and $y[n]$ using the MATLAB function fft. If you give this routine a length N signal then it will evaluate

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}$$

at the frequencies $\omega = 2\pi k/N$ for $k = 0, 1, 2...N-1$ Note the differences between this and equation 2.113 in Oppenheim and Schafer. Here we only have a length N signal so we can't sum from $-\infty$ to ∞. Also with MATLAB on a digital computer we can't express a function of a continuous variable (ω in 2.113 is continuous). So we only save the values of the spectrum at certain frequencies, i.e. we sample the frequency spectrum ($X(e^{j\omega})$). The sample frequencies are evenly spaced. We know that for a digital signal that the spectrum will be periodic in 2π. So we only sample over one period. This yields sample frequencies where the lowest is 0 and the highest is (close to) 2π, as we have above.

As you know, in general the spectrum is complex. Here we will just plot the magnitude of the signal. Also we will need to generate a vector for the frequency values so that when you plot your spectrum you will have the proper scale on the horizontal axis.

The commands below will do all that.

Example

n = 1:N; $x = sin(\omega n)$;	% input sequence
$y = func\_2(x)$;	% output of system 2
$fy = fft(y)$;	% DFT of output sequence
$mag\_fy = abs(fy)$;	% get magnitude spectrum
$freq\_ind = k = 0 : (2 * pi/1000) : (2 * pi - 2 * pi/1000)$;	% proper frequency index
$plot(freq\_ind, mag\_fy)$;	% plot magnitude spectrum

This plots the magnitude spectrum of $y[n]$. Use similar commands to get the magnitude spectrum of $x[n]$. Compare the two spectrums (use help on the matlab commands: *plot* and *figure* to get the plots the way you want them.) What can you say about the system?

Constant Coefficient Difference Equations

In this section constant coefficient difference equations are explored by examining a simple application. We are interested in monitoring the average power of a signal which has been measured at a transducer. The measured signal has been lowpass filtered by the system anti-aliasing filter to a cutoff frequency of 100Hz. The filtered signal is then sampled and discretized. An approxi-

mate means of monitoring the average power is to be developed based on a numerical integration method. Recall that the average power of a continuous-time signal is defined by

$$P_{ave} = \frac{1}{T_{obs}} \int_{T_{obs}} |x(t)|^2 dt$$

where T_{obs} denotes observation time and s(t) the measured signal.

An example of a numerical integration method is the trapezoidal rule which is given as (see Faires and Burden)

$$\int_a^b f(t)dt = (b-a)\frac{f(a)+f(b)}{2} - \frac{f''(\epsilon)}{12}(b-a)^3$$

The last term is an error term.

To use this to evaluate our average power, let $f(t) = |x(t)|^2$, then split the interval $[0, T_{obs}]$ into N equal length subintervals of length T_s. Here T_s is the sampling period of the digital signal. Apply the trapezoidal rule over each subinterval and add them up. In your lab write-up give a formula for calculating N. Show a development which starts with the continuous integral for P_{ave} and arrives at the formula:

$$P_{ave} \approx \frac{1}{N}(|x[0]|^2/2 + |x[1]|^2 + \ldots + |x[N-1]|^2 + |x[N]|^2/2)$$

Discuss the following further approximation:

$$P_{ave} \approx \frac{1}{N}\left(|x[0]|^2 + |x[1]|^2 + \ldots + |x[N-1]|^2\right)$$

The system to be developed is the *average power monitor* (`APM`). The idea here is that we won't figure out how much energy there has been since the beginning of time ($n = 0$) but only in some "window" of recent times. For example for a window of size 10 we will find the total energy in the present sample plus the nine before that; we will add these up and then divide by ten. Due to real time constraints the computation allocation for the `APM` is 10`ms`, where an add operation takes 2`ms` and a multiplication takes 3`ms`. The first data sample enters the `APM` at time $n = 0$. Remember that the average power is to be updated every sample. (You will want to satisfy the 10`ms` requirement at the typical sample.)

`Hint:` The `APM` can be approximated by:

$$y[n] = \frac{1}{P}\sum_{k=0}^{P} |x[n-k]|^2$$

where P is the monitoring window size and N is the total number of samples in a sequence.

Problem 3

1. Design and test an `APM` based on the trapezoidal rule for numerical integration. The important point in this discussion is the computational restrictions placed on the `APM`. In this regard a direct or brute force application of the trapezoidal rule is not possible. The trapezoidal rule must be implemented recursively. By properly interpreting the numerical integration method in terms of a constant coefficient difference equation the constraints imposed on the design can be met. Testing the monitor will involve passing a few different sequences through the `APM`. Test sequences are to be 500 samples long and the monitoring window size is to be 3 samples. Make sure that you use sequences that represent constant average power, increasing average power, and decreasing average power. Plot all your test results. Repeat this for a window size of 20.

2. Is the `APM` a stable and causal system?

Problem 4

Give a plot of the signal you acquired in the pre-lab. Is your signal periodic? If so state the period.

A *stationary* signal is, broadly speaking, one whose general behavior doesn't change over time. Usually this is a matter of what scale we are looking at a signal. A heart rate signal with sampling rate 1000 Hz take for 10 seconds would likely be called stationary. The familiar pulses happen about 70 times a second throughout the duration of the signal.

An audio signal of a person saying a simple sentence would generally be called non-stationary. There would be pauses between words that would come at irregular intervals. So looking at the first second, and the last second of our 10 second signal we might see very different things.

Would you say that your signal is stationary? Comment.