BIRZEIT UNIVERSITY

Electrical and computer Engineering

ENCS431 Digital Signal Processing - Fall 2020

Homework #1 (chapter 2)

Deadline: Sunday 11/10/2020 23:55

▪ DISCRETE-TIME SIGNALS

Signals are broadly classified into analog and discrete signals. An analog signal will be denoted by x(t), in which the variable t can represent any physical quantity, but we will assume that it represents time in seconds. A discrete signal will be denoted by x(n), in which the variable n is integer-valued and represents discrete instances in time. Therefore it is also called a discrete-time signal, which is a number sequence and will be denoted by one of the following notations:

$$x(n) = \{x(n)\} = \{\cdots, x(-1), \underset{\uparrow}{x(0)}, x(1), \cdots\}$$

Where the up-arrow indicates the sample at n = 0.

In MATLAB we can represent a finite-duration sequence by a row vector of appropriate values. However, such a vector does not have any representation of x(n) would require two vectors, one each for x and n. For example, a sequence:

$$x(n = \{2, 1, -1, \underset{\uparrow}{0}, 1, 4, 3, 7\}$$

Can be represented in MATLAB by:

```
>> n=[-3,-2,-1.0,1,2,3,4];
>> x=[2,1,-1,0,1,4,3,7];
```

Generally, we will use the x-vector representation alone when the sample position information is not required or when such information is trivial (e.g. when the sequence begins at is = 0). An arbitrary infinite-duration sequence cannot be represented in MATLAB due to the finite memory limitations.

▪ TYPES OF SEQUENCES

We use several elementary sequences in digital signal processing for analysis purposes. Their definitions and MATLAB representations are given below.

1. Unit sample sequence

$$\delta(n) = \begin{cases} 1 & n=0 \\ 0 & n\neq 0 \end{cases}$$

However, the logical relation n = 0 is an elegant way of implementing δ(n). For example, to implement:

$$\delta(n-no) = \begin{cases} 1 & n=no \\ 0 & n\neq no \end{cases}$$

Over the interval $n1 \leq n \leq n2$, we will use the following MATLAB function:

```
=======================================================
% Function to generate x(n)=delta(n-no), n1<=n<=n2
function [x,n]=impseq(no,n1,n2)
n=[n1:n2];
x=[(n-no) == 0];
=======================================================
```

2. Unit step sequence

$$u(n) = \begin{cases} 1 & n\geq 0 \\ 0 & n<0 \end{cases}$$

In MATLAB the function zeros(1,N) generates a row vector of N zeros, which can be used to implement u(n) over a finite interval. However, the logical relation n = 0 is an elegant way of implementing u(n). For example, to implement

$$u(n-no) = \begin{cases} 1 & n=no \\ 0 & n\neq no \end{cases}$$

Over the interval $n1 \leq n \leq n2$, we will use the following MATLAB function:

```
=======================================================
% Function to generate x(n)=step(n-no), n1<=n<=n2
function [x,n]=stepseq(no,n1,n2)
n=[n1:n2];
x=[(n-no) >= 0];
=======================================================
```

3. Real-valued exponential sequence:

$$x(n) = a^n, \forall n;\ a \in \mathbb{R}$$

In MATLAB an array operator ". " is required to implement a real exponential sequence. For example, to generate x(n) = $(0.9)^n$, 0 ≤n≤ 10, we will need the following MATLAB script:

```
>> n=[0:10];
>> x=(0.9).^n;
```

4. Complex-valued exponential sequence:

$$x(n) = e^{(\sigma + j\omega_0)n}, \forall n$$

Where σ is called an attenuation and ωo is the frequency in radians.

MATLAB function script is used to generate exponential sequences. For example, to generate x(n) = exp [(2 + j3) n], 0≤n≤10, we will the following MATLAB script:

```
>> n=[0:10];
>> x=exp(2+3i).^n;
```

5. Sinusoidal sequence:

$$x(n) = \cos(\omega_0 n + \theta), \forall n$$

Where θ is the phase in radians. A MATLAB function cos(or sin) used to generate sinusoidal sequences. For example, to generate x(n) =3cos(0.lπn+π/3)+2sin(0.5πn), 0≤n≤10, we will need the following MATLAB script:

```
>> n = [0:10];
>> x = 3*cos(0.1*pi*n+pi/3) + 2*sin(0.5*pi*n);
```

6. Random sequences:

Many practical sequences can't be described by mathematical expressions. Those sequences are called random (stochastic) sequences and are characterized by parameters of the associated probability density function or their statistical moments. In MATLAB there are two types of random sequences. The ***rand(1,N)*** generates a length N random sequence whose elements are uniformly distributed between [0,1]. The ***randn(1,N)*** generates a length N Guassian random sequence with mean 0 and variance 1.

7. Periodic sequences:

A sequence x(n) is periodic if x(n)=x(n+N), the smallest integer n satisfies the previous relation is called the fundamental period.

▪ OPERATIONS ON SEQUENCES

Here we're briefly described basic sequence operations and their MATLAB equivalents.

1. Signal addition:

$$\{x_1(n)\} + \{x_2(n)\} = \{x_1(n) + x_2(n)\}$$

It's implemented by MATLAB using the operator "+". Notice that the two sequences x1(n) and x2(n) must have the same length and the same indexing. We can use the following MATLAB function to demonstrate the discrete sequences additions:

```
============================================================
% Function adds two discrete time sequences y(n)=x1(n)+x2(n)
function [y,n]=sigadd(x1,n1,x2,n2)
n=(min(n1),min(n2)):(max(n1),max(n2));
y1=zeros(1,length(n));
y2=y1;
y1=find((n>=min(n1))&(n<=max(n1==1))=x1;
y2=find((n>=min(n2))&(n<=max(n2==1))=x2;
y=y1+y2;
============================================================
```

2. Signal multiplication:

$$\{x_1(n)\} \cdot \{x_2(n)\} = \{x_1(n)x_2(n)\}$$

It's implemented in MATLAB by array operator ".*". Once again the similar restrictions apply for the ".*" operator as "+"

We can use the following MATLAB function to demonstrate the discrete sequences multiplication:

```
============================================================
% Function multiplies two discrete time sequences y(n)=x1(n)*x2(n)
function [y,n]=sigmult(x1,n1,x2,n2)
n=(min(n1),min(n2)):(max(n1),max(n2));
y1=zeros(1,length(n));
y2=y1;
y1=find((n>=min(n1))&(n<=max(n1==1))=x1;
y2=find((n>=min(n2))&(n<=max(n2==1))=x2;
y=y1.*y2;
============================================================
```

3. Signal scaling:

In this operation each sample is multiplied by a scalar α.

$$\alpha\{x(n)\} = \{\alpha x(n)\}$$

The operator "*" is used to implement scaling in MATLAB.

4. Signal shifting:

In this operation each sample is shifted by amount of k to obtain the shifted sequence y(n):

$$y(n) = \{x(n-k)\}$$

If we let m=n-k, then n=m+k and the above operation is given by:

$$y(m+k) = \{x(m)\}$$

Hence this operation has no effect on the vector x, but the vector n is changed by adding k to each element. This is can be implemented by the following MATLAB function:

```
=========================================================
% Function does signal shifting y(n)=x(n-no)
function [y,n]=sigshift(x,n,no)
n=n+no;
y=x;
=========================================================
```

5. Signal folding:

In this operation each sample of x(n) is flipped around n=0 to obtain a folded sequence y(n):

$$y(n) = \{x(-n)\}$$

In MATLAB this operation is implemented as follows:

```
=========================================================
% Function does signal folding y(n)=x(-n)
function [y,n]=sigfold(x,n)
n=n+no;
y=fliplr(x);
n=-fliplr(n);
=========================================================
```

6. Sample summation:

This operation adds all sample values of x(n) between n1 and n2.

$$\sum_{n=n_1}^{n_2} x(n) = x(n_1) + \cdots + x(n_2)$$

And it's implemented by MATLAB by:

>> sum(x(n1:n2));

7. Sample product:

This operation multiplies all sample values of x(n) between n1 and n2.

$$\prod_{n_1}^{n_2} x(n) = x(n_1) \times \cdots \times x(n_2)$$

And it's implemented by MATLAB by:

>> prod(x(n1:n2));

8. Signal energy:

The energy of a sequence x(n) is given by:

$$\mathcal{E}_x = \sum_{-\infty}^{\infty} x(n)x^*(n) = \sum_{-\infty}^{\infty} |x(n)|^2$$

The energy of a finite duration x(n) can be computed in MATLAB using:

>> E=sum(abs(x).^2);

9. Sample power:

The average power of a periodic sequence with fundamental period N is given by:

$$\mathcal{P}_x = \frac{1}{N} \sum_{0}^{N-1} |x(n)|^2$$

Exercise 1.1

Generate and plot each of the following sequences over the indicated intervals:

a. x1(n) = 2δ(n+2)-δ(n-4), -5≤ n ≤5

b. x2(n) = n[u(n)-u(n-10)]+10exp(-0.3(n-10))*[u(n-10)-u(n-20)], 0≤ n ≤20

c. x3(n) = cos(0.4πn)+0.2w(n), 0≤ n ≤50, where w(n) is a Guassian random sequence with 0 and variance 1.

d. x4(n) = exp(-0.1+j0.3)n, -10≤ n ≤10

▪ **DISCRETE SYSTEMS**

Mathematically, a discrete time system is described as an operator T[.] that takes a sequence x(n) (called *excitation*) and transforms it into another sequence y(n) (called *response*).

$$y(n) = T[x(n)]$$

Discrete systems are classified into linear and non-linear systems. We will deal mostly on linear systems.

Linear systems:

A discrete time system is called linear system if it satisfies the superposition principle.

$$L[a_1x_1(n) + a_2x_2(n)] = a_1L[x_1(n)] + a_2L[x_2(n)], \forall a_1, a_2, x_1(n), x_2(n)$$

Linear time-invariant (LTI) systems:

A linear system in which an input-output pair, x(n) and y(n), is invariant to a shift n in time is called a linear-time invariant system. For LTI system the L[.] and the shifting operators are available as shown below.

$$x(n) \longrightarrow \boxed{L[\cdot]} \longrightarrow y(n) \longrightarrow \boxed{\text{Shift by } k} \longrightarrow y(n-k)$$

$$x(n) \longrightarrow \boxed{\text{Shift by } k} \longrightarrow x(n-k) \longrightarrow \boxed{L[\cdot]} \longrightarrow y(n-k)$$

The impulse response of an LTI system is given by h(n).

$$x(n) \longrightarrow \boxed{h(n)} \longrightarrow y(n) = x(n) * h(n)$$

And it's represented mathematically by the linear convolution sum:

$$y(n) = LTI[x(n)] = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

$$y(n) \triangleq x(n) * h(n)$$

We will now explore some of the LTI system properties.

1. Stability:

This is a very important concept in linear system theory. The primary reason for considering stability is to void building harmful systems and to avoid burnout or saturation in systems operation. A system is said to be ***bounded-input bounded output*** stable if every input produced bounded output.

$$|x(n)| < \infty \Rightarrow |y(n)| < \infty, \forall x, y$$

An LTI system is BIBO if and only if its impulse response is ***absolutely summable***.

$$\text{BIBO Stability} \Longleftrightarrow \sum_{-\infty}^{\infty} |h(n)| < \infty$$

2. Casuality

This important concept is necessary to make sure that systems can be built. A system is said to be casual if the output at index no depends only on the input up to and including the index no; that is the output doesn't depend on the future values

An LTI system is casual if and only if its impulse response:

$$h(n) = 0, \quad n < 0$$

▪ CONVOLUTION

In DSP, the convolution is an important operation as it has many uses. It can be evaluated in many ways.

MATLAB provides a built-in function called ***conv*** that computes the convolution between two finite duration sequences. The ***conv*** function assumes that the two sequences begin at n=0 and is invoked by:

Example 3.1

Given the following two sequences

$$x(n) = \left\{3, 11, 7, \underset{\uparrow}{0}, -1, 4, 2\right\}, \quad -3 \le n \le 3; \qquad h(n) = \left\{\underset{\uparrow}{2}, 3, 0, -5, 2, 1\right\}, \quad -1 \le n \le 4$$

Determine the convolution y(n)=x(n)*h(n).

```
>> x=[3 11 7 0 -1 4 2];
>> h=[2 3 0 -5 2 1];
>> y=conv(x,h)
```

y =

6 31 47 6 -51 -5 41 18 -22 -3 8 2

Note that the ***conv*** function doesn't provide any timing information if the sequences have arbitrary support. A simple extension of the ***conv*** function called ***conv_m***, which perform the convolution of arbitrary support sequences.

```
=========================================================
% Modified convolution routine
function [y,ny]=conv_m(x,nx,h,hx)
nyb=nx(1)+nh(1);
nye=nx(length(x))+nh(length(h));
ny=[nyb:nye];
y=conv(x,h)
=========================================================
```

Example 3.2: Perform the convolution in example 3.1 using ***conv_m*** function.

▪ CORRELATION

The cross-correlation between two sequences can be evaluated by the formula:

$$r_{yx}(\ell) = y(\ell) * x(-\ell)$$

With the auto-correlation in the form:

$$r_{xx}(\ell) = x(\ell) * x(-\ell)$$

In MATLAB, cross-correlation between two sequences x(n) and y(n) is implemented using xcorr(x,y) function and the auto-correlation of a sequence x(n) is computed.

using xcorr(x).

Exercise 1.2

Let

$$x(n) = \left[3, 11, 7, \underset{\uparrow}{0}, -1, 4, 2\right]$$

A prototype sequence, and y(n) be its noise corrupted-and-shifted version:

$$y(n) = x(n-2) + w(n)$$

Where w(n) is a Guassian random sequence with 0 and variance 1. Compute the cross-correlation between x(n) and y(n).

▪ DIFFERENCE EQUATIONS

An LTI discrete system can also be described by a linear constant coefficients difference equation of the form:

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{m=0}^{M} b_m x(n-m), \quad \forall n$$

Another form of the previous equation is:

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{k=1}^{N} a_k y(n-k)$$

A routine called ***filter*** is available in MATLAB to solve the difference equation numerically, given the input and the difference equation coefficients. This routine is invoked as follows:

>> y=filter(b,a,x)

Where b=[b_0 b_1 b_2.....b_M], a=[a_0 a_1 a_2.....a_M] are the coefficient arrays of the difference equation and x is the input sequence. The output y has the same length as x and insures that $a_0 \neq 0$.

A useful use of ***filter*** function is that it can be used to find the numerical evaluation of the convolution of two sequences if one of them has infinite length**.** That's because we can't use ***conv*** function when one or both sequences of the convolution are of infinite length.

Exercise 1.3

Given the following difference equation:

$$y(n) - y(n-1) + 0.9y(n-2) = x(n); \quad \forall n$$

a. Calculate and plot the impulse response h(n) at n=-20,.....120.

b. Calculate and plot the unit step response s(n) at n=-20,.....120.

c. Is the system specified by h(n) stable?

Exercise 1.4

Consider that we have an input x(n) of finite duration sequence:

$$x(n) = u(n) - u(n-10)$$

While the impulse response of infinite duration:

$$h(n) = (0.9)^n\, u(n)$$

Compute the output signal y(n)?

Exercise 1.5

Consider the first order system y(n) – **a**y(n-1) = x(n). i.e. y(n) = x(n) + ay(n-1)
A – Use Matlab to generate a sinusoidal signal with unity amplitudes and the following frequencies: 1KHz, 2KHz, and 4KHz. Assume sampling frequency Fs = 10KHz.

B- Assume **a**=0.9 and use Matlab to implement the above system and compute the output signal y(n). Compare the frequencies of the input signal x(n) and y(n) output signal y(n)?

C- Change the value of the coefficient (**a**) such that the system passes frequencies 1KHz and 2KHz only. Record the value of (a) that you obtain and plot the input and the output signals.