Lab 02 –

Convolution And Digital Signal Transmission

## Objectives:

1. To study convolution and correlation of discrete-time signals
2. To Study Discrete time Systems
3. To study Signal Transmission schemes

## Introduction:

*Convolution of sequences* As the mathematical operation, a linear convolution sum is defined as



In general , the convolution operation is used to describe the response of an LTI system. In DSP it is an important operation and has many other uses.

If arbitary sequences are of infinite duration, then MATLAB cannot be used directly to compute the convolution. MATLAB does provide a built-in function called **conv** the computes the convolution between two finite-duration sequences. The **conv** function assumes that the two sequences begin at n = 0 and is invoked by

**>> y = conv(x, h);**

However, the **conv** function neither provides nor accepts any timing information if the sequences have arbitrary support. To solve this inconvenience, a simple extension of the **conv** function, called **conv\_m**, which performs the convolution of arbitrary support sequences can now be designed as

**function [y, ny] = conv\_m(x, nx, h, nh)**

**% Modified convolution routine for signal processing**

**% --------------------------------------------------------------**

**% [y, ny] = concolution result**

**% [x, nx] = first signal**

**% [h, nh] = second signal**

**%**

**nyb = nx(1) + nh(1);**

**nye = nx(length(x)) + nh(length(h));**

**ny = [nyb:nye];**

**y = conv(x, h);**

**% End of the function**

**Task1**

( Before starting the lab, you’re supposed to have all functions from DSP-01 stored in your working folder.)

##### Convolution of Sequences

1. Perform the convolution of following sequence using the conv m function. Match it with your own manual solution

**x = [3, 11, 7, 0, -1, 4, 2]; nx = [-3:3];**

**h = [2, 3, 0, -5, 2, 1]; ny = [-1:4];**

**task1 Q1 : Can you verify the answer manually**

1. Convolution of Audio signals: Convolution Reverb

Develop a MATLAB code to convolve your speech signal with the impulse response of some acoustic space like ‘Long Echo Hall’. Listen to the output and explain the phenomenon of convolution reverb. Also, explain how special effects can be created in sound using convolution? Create three effects.

**Task2**

Tips

1. Use Wav File
2. Your Audio should have only one channel : It should be mono

# Introduction: Discrete time Systems

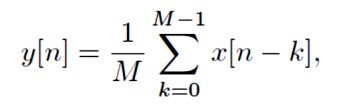
A discrete-time system processes an input signal in the time-domain to generate an output signal with more desirable properties by applying an algorithm composed of simple operations on the input signal and its delayed versions. The aim of this second exercise is to illustrate the simulation of some simple discrete-time systems on the computer using MATLAB and investigate their time domain properties.

# Background Review

##### Exercise The Moving Average System

Examination of Eq. (2.5) reveals that the three-point smoothing filter considered A causal version of the three-point smoothing filter is obtained by simply delaying the output by one sample period, resulting in the FIR filter described by y[n] = 1/3 (x[n] + x[n − 1] + x[n − 2]). (2.5)

Generalizing the above equation we obtain

(2.6)

which defines a causal M-point smoothing FIR filter. The system of Eq. (2.6) is also known as a moving average filter . We illustrate its use in filtering high-frequency components froma signal composed of a sum of several sinusoidal signals.

**Task 3**

% Simulation of an M-point Moving Average Filter

% Generate the input signal

n = 0:100;

s1 = cos(2\*pi\*0.05\*n); % A low frequency sinusoid

s2 = cos(2\*pi\*0.47\*n); % A high frequency sinusoid

x = s1+s2;

% Implementation of the moving average filter

M = input(’Desired length of the filter = ’);

num = ones(1,M);

y = filter(num,1,x)/M;

% Display the input and output signals

clf;

subplot(2,2,1);

plot(n,s1);

axis([0, 100, -2, 2]);

xlabel(’Time index n’);

ylabel(’Amplitude’);

title(’Signal # 1’);

subplot(2,2,2);

plot(n,s2);

axis([0, 100, -2, 2]);

xlabel(’Time index n’); ylabel(’Amplitude’);

title(’Signal # 2’);

subplot(2,2,3);

plot(n,x);

axis([0, 100, -2, 2]);

xlabel(’Time index n’); ylabel(’Amplitude’);

title(’Input Signal’);

subplot(2,2,4);

plot(n,y);

axis([0, 100, -2, 2]);

xlabel(’Time index n’); ylabel(’Amplitude’);

title(’Output Signal’);

axis;

**Questions:**

**Task 3 1** Is the above system a low pass filter or high pass filter? Explain your answer

**Task 3 .2** In above program, define a new value for num

n1=0:M-1;

num=(-1).^n1.\*ones(1,M);

How has this changed the response of the system? Is the system now Low pass or High Pass

**Task 3.3** Run the above program for M = for different values to to generate the output signal with x[n] = s1[n] + s2[n] as the input. Which component of the input x[n] is suppressed by the discrete time system simulated by this program? How does M effect the output

# Transmitting Digital Systems

Introduction:

Line coding is a crucial aspect of digital communication where binary data is translated into a digital signal. Different line coding schemes have distinct characteristics affecting the bandwidth, efficiency, and complexity of communication systems. Power Spectral Density (PSD) provides insights into the frequency content of a signal.

In this lab, you will use MATLAB to generate line codes for a given binary sequence (1010010101110) and analyze the PSD of each scheme. The generated graphs will visually represent the characteristics of each line coding scheme.

Why Line Coding is Essential

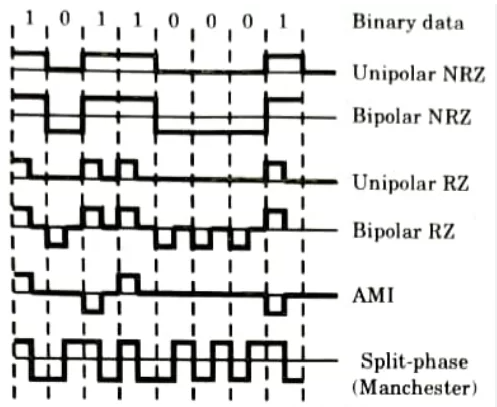
In the realm of digital communication, where binary data is the universal language of information, the process of transforming this abstract data into a physical, transmittable signal is crucial. Line coding serves as the cornerstone of this transformation. It ensures that the binary information is adequately represented in a manner suitable for transmission over diverse communication channels. Without effective line coding, digital information would remain trapped in the realm of zeros and ones, unable to traverse the complex landscapes of communication networks.

Comparing Line Coding Scheme

Line coding schemes are diverse, each with its own set of advantages and trade-offs. The choice of a particular scheme depends on the specific requirements of the communication system at hand. Parameters such as bandwidth utilization, spectral efficiency, and susceptibility to noise and interference come into play. In this lab, students will explore and compare various line coding schemes, understanding the implications of their design choices on the transmitted signal. By evaluating these schemes side by side, students will gain insights into which coding strategy aligns best with different communication scenarios.

Task 4

1. Generate functions to create following line codes based on the data provided by the user
2. Compare the line code in term of DC values, self synchronization , Power efficiency



Q