**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Signal Processing**

**Lab Number**

**2**

**SUBMITTED TO:**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Processing in MATLab

**Related Topic/Chapter in theory class:**

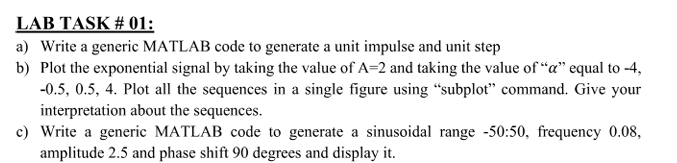
Basics Of Digital Signal Processing

**Hardware/Software required:**

Hardware: PC

Software Tool: MATLab

**Task 1:**

** (a)**

**Solution:**

%%

function y = u(x)

y = zeros(size(x));

y(x>=0) = 1;

end

function y = delta(x)

y = zeros(size(x));

y(x==0) = 1;

end

n = -10:10

impulse = u(n);

del = delta(n);

subplot(2, 1, 1);

stem(n, impulse)

xlabel('Samples')

ylabel('Amplitude')

title('Impulse Function')

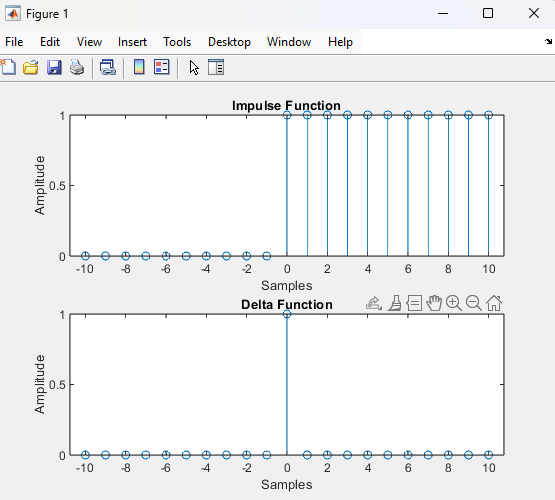
subplot(2, 1, 2);

stem(n, del)

xlabel('Samples')

ylabel('Amplitude')

title('Delta Function')

****

**(b)**

**Solution**

%%

n = -10:0.1:10;

A = 2;

% 2(-4^n)

exp\_1 = A .\* (-4 .^ n);

% 2(-0.5^n)

exp\_2 = A .\* (-0.5 .^ n);

% 2(0.5^n)

exp\_3 = A .\* (0.5 .^ n);

% 2(4^n)

exp\_4 = A .\* (4 .^ n);

subplot(4, 1, 1)

stem(n,exp\_1)

xlabel('Samples')

ylabel('Amplitude')

title('Alpha = -4')

subplot(4, 1, 2)

stem(n, exp\_2)

xlabel('Samples')

ylabel('Amplitude')

title('Alpha = -0.5')

subplot(4, 1, 3)

stem(n, exp\_3)

xlabel('Samples')

ylabel('Amplitude')

title('Alpha = 0.5')

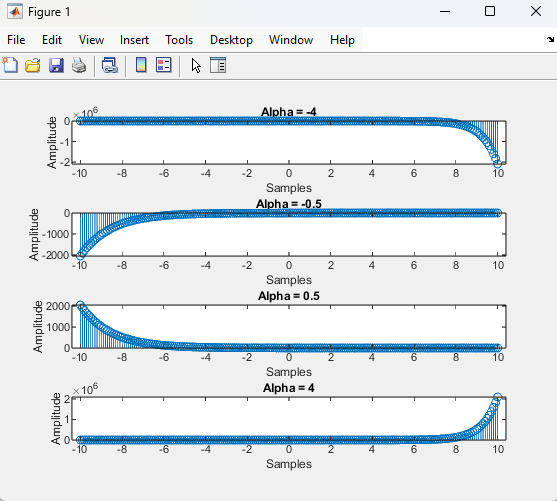
subplot(4, 1, 4)

stem(n, exp\_4)

xlabel('Samples')

ylabel('Amplitude')

title('Alpha = 4')



**(c)**

**Solution**

%%

n = -50:0.1:50

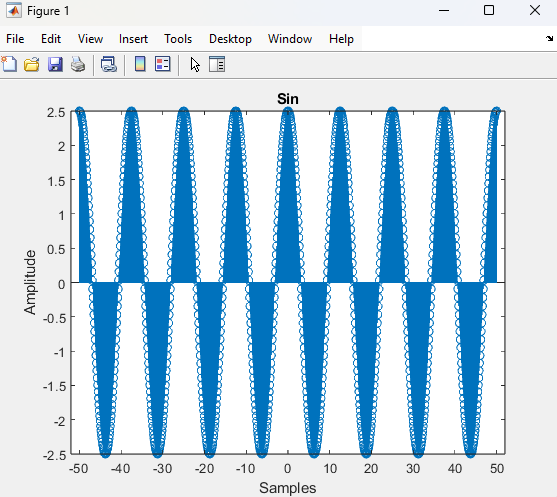
sinusoid =2.5.\* sin((2.\*pi .\* 0.08 .\* n) + (pi/2))

stem(n, sinusoid)

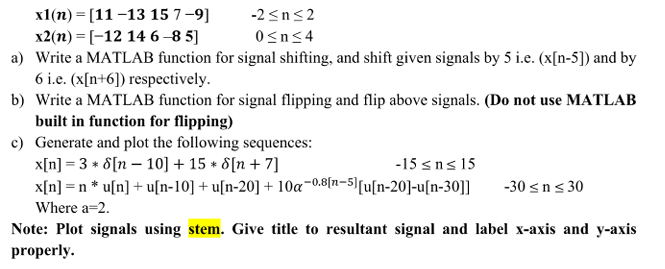
xlabel('Samples')

ylabel('Amplitude')

title('Sin')



**Task 2**

****

**(a)**

**Solution**

%%

function y = shift\_5(n)

y = n-5;

end

function y = shift\_6(n)

y = n+6;

end

x1 = [11 -13 15 7 -9];

n1 = -2:2;

x2 = [-12 14 6 -8 5];

n2 = 0:4;

n1\_shifted = shift\_5(n1);

n2\_shifted = shift\_6(n2);

subplot(4, 1, 1)

stem(n1, x1)

xlabel('Samples')

ylabel('Amplitude')

title('X1[n]')

subplot(4, 1, 2)

stem(n1\_shifted, x1)

xlabel('Samples')

ylabel('Amplitude')

title('X1[n-5]')

subplot(4, 1, 3)

stem(n2, x2)

xlabel('Samples')

ylabel('Amplitude')

title('X2[n]')

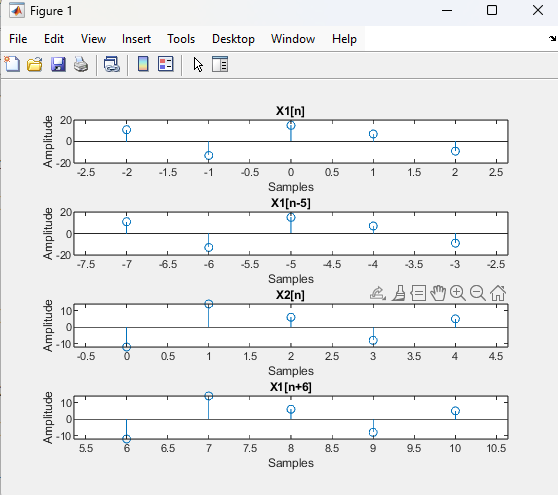
subplot(4, 1, 4)

stem(n2\_shifted, x2)

xlabel('Samples')

ylabel('Amplitude')

title('X1[n+6]')



**(b)**

**Solution**

%%

function y = flip(n)

y = -n;

end

n1\_flipped = flip(n1);

n2\_flipped = flip(n2);

subplot(4, 1, 1)

stem(n1, x1)

xlabel('Samples')

ylabel('Amplitude')

title('X1[n]')

subplot(4, 1, 2)

stem(n1\_flipped, x1)

xlabel('Samples')

ylabel('Amplitude')

title('X1[-n]')

subplot(4, 1, 3)

stem(n2, x2)

xlabel('Samples')

ylabel('Amplitude')

title('X2[n]')

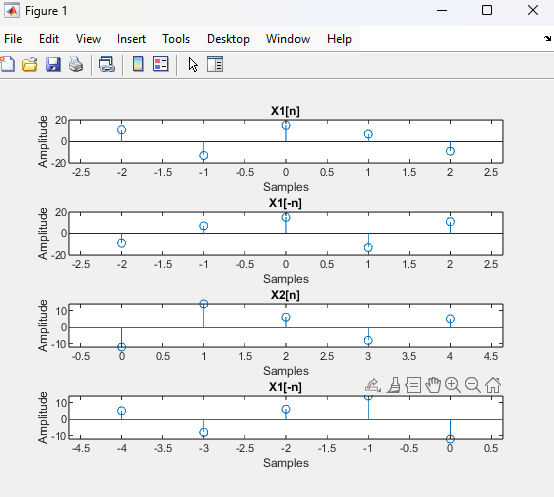
subplot(4, 1, 4)

stem(n2\_flipped, x2)

xlabel('Samples')

ylabel('Amplitude')

title('X1[-n]')



**(c)**

**Solution**

%%

function y = u(x)

y = zeros(size(x));

y(x>=0) = 1;

end

function y = delta(x)

y = zeros(size(x));

y(x==0) = 1;

end

n1 = -15:15;

n2 = -30:30;

x1 = 3 .\* delta(n1-10) + 15 .\* delta(n1+7);

x2 = n2 .\* u(n2) + u(n2-10) + u(n2-20) + (10 .\* 2 .\*(-0.8 .^ (n2-5))).\* (u(u(n2-20)-u(n2-30)));

subplot(2, 1, 1)

stem(n1, x1)

xlabel('Samples')

ylabel('Amplitude')

title('X1')

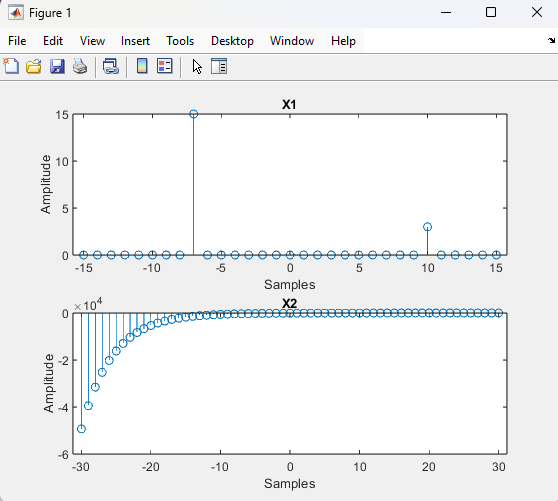
subplot(2, 1, 2)

stem(n2, x2)

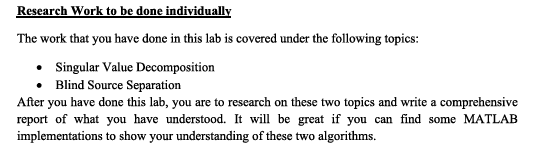
xlabel('Samples')

ylabel('Amplitude')

title('X2')

****

**Research Work:**

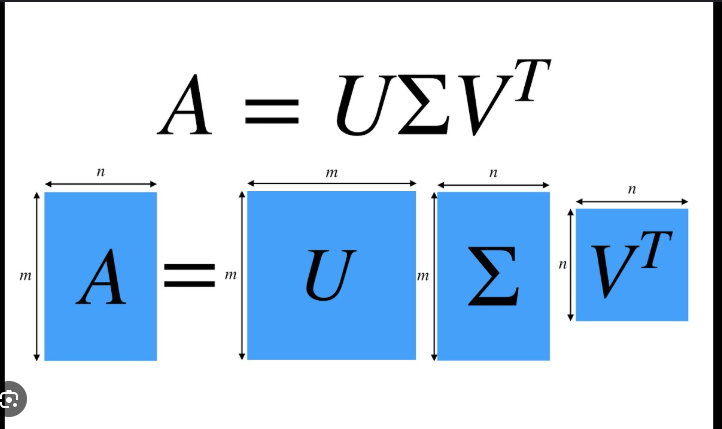
****

**Single Value Decomposition:**

Singular Value Decomposition (SVD) is a mathematical technique used to factorize a matrix into three matrices:

Where:

* U is an m × m orthogonal matrix (left singular vectors) (columns are eigenvectors of )
* Σ is an m × n diagonal matrix containing singular values (diagonal entries, square roots of eigenvalues of )
* is an n × n orthogonal matrix (right singular vectors) (columns are eigenvectors of )



**MATLAB Implementation:**

A = [4 0; 3 -5; 0 2];

% Compute SVD

[U, S, V] = svd(A);

% Display results

disp('U matrix:');

disp(U);

disp('Singular values (S matrix):');

disp(S);

disp('V matrix:');

disp(V);

% Verify the decomposition

A\_reconstructed = U \* S \* V';

disp('Reconstructed A:');

disp(A\_reconstructed);

**Output:**

U matrix:

-0.4059 0.8736 -0.2683

-0.8840 -0.3009 0.3578

0.2318 0.3824 0.8944

Singular values (S matrix):

6.4910 0

0 3.4449

0 0

V matrix:

-0.6587 0.7524

0.7524 0.6587

Reconstructed A:

4.0000 0.0000

3.0000 -5.0000

0.0000 2.0000

**Blind Source Separation:**

Blind Source Separation (BSS) is the process of separating mixed signals into their original independent sources without prior knowledge of the mixing process. The most common technique for BSS is **Independent Component Analysis (ICA)**.

Applications include:

* Audio signal separation (e.g., separating voices in a room)
* Biomedical signal processing (e.g., EEG signal separation)