**Section -1: DFT and Circular Convolution**

* Compute and compare the eight point circular convolution for the following sequences x1[n] and x2[n] using

1. the circular convolution definition and
2. The DFT method.

**MATLAB CODE**



function output = CircularConv(signal\_1, signal\_2)

% CircularConv - Description

% Recives two Signal and Finds Circular Convolution of the Signal using Matrix Multiplication Method

N = length(signal\_1);

M = length(signal\_2);

Mat\_A = zeros(N,N);

Col\_B = zeros(M,1);

for ix = 0:N-1

Mat = zeros(size(1,N));

for iy = 0:N-1

Mat(iy+1) = signal\_1(mod((iy - ix + N),N)+1);

end

Mat\_A(:,ix+1) = Mat';

end

Col\_B = signal\_2';

output = (Mat\_A \* Col\_B)';

end

**MATLAB CODE**



function [DFT, K] = DFT(signal, n)

N = length(n);

K = 0:N-1;

DFT = zeros(size(K));

for i\_k = 1:length(K)

for i\_n = 1:N

DFT(i\_k) = DFT(i\_k) + signal(i\_n) \* exp(-1i\*(2\*pi/N)\*K(i\_k)\*n(i\_n));

end

end

end



function [IDFT, n] = IDFT(signal, K)

n = 0:length(K)-1;

N = length(n);

IDFT = zeros(size(n));

for i\_n = 1:N

for i\_k = 1:length(K)

IDFT(i\_n) = IDFT(i\_n) + signal(i\_k) \* exp(1i\*(2\*pi/N)\*K(i\_k)\*n(i\_n));

end

end

IDFT = IDFT./N;

end



function [output, X1, X2, K] = CircularConvDFT(signal\_1, signal\_2, n)

N = length(n);

[X1, ~] = DFT(signal\_1, n);

[X2, K] = DFT(signal\_2, n);

X3 = X1.\*X2;

[output, ~] = IDFT(X3,K);

end

**MATLAB CODE**



% CA-05 CIRCULAR CONVOLUTION

close all; clear;

n = 0:7;

signal\_1 = (1/4).^n;

signal\_2 = cos((3\*pi/8).\*n);

y1 = CircularConv(signal\_1, signal\_2);

figure; subplot(311);

stem(n,signal\_1,'x','LineWidth', 2);

title('Signal x1[n] = 1/(4^n)'); ylabel('x1[n]'); xlabel('n'); grid;

subplot(312);

stem(n, signal\_2,'x','LineWidth', 2);

title('Signal x2[n] = cos(3\*pi/8\*n'); ylabel('x2[n]'); xlabel('n'); grid;

subplot(313);

stem(n, y1,'x','LineWidth', 2);

title('Signal Y[n] = Circular Convolution X1[n] \* X2[n]'); ylabel('y[n]'); xlabel('n'); grid;

% CIRCULAR CONVOLUTION USING DFT

[y1, X1, X2, K] = CircularConvDFT(signal\_1, signal\_2, n);

figure;

subplot(311); stem(n,signal\_1,'x','LineWidth', 2);

title('Signal x1[n] = (1/4)^n'); ylabel('x1[n]'); xlabel('n'); grid;

subplot(312); stem(n, signal\_2,'x','LineWidth', 2);

title('Signal x2[n] = cos(3\*pi/8\*n)'); ylabel('x2[n]'); xlabel('n'); grid;

subplot(313); stem(n, y1,'x','LineWidth', 2);

title('Signal Y[n] = Inverse DFT of X3[k] = X1[k] x X2[k]'); ylabel('Y[n]'); xlabel('n'); grid;

figure; subplot(311);

stem(K, real(X1),'x','LineWidth', 2); hold on; stem(K, imag(X1),'o','LineWidth', 2);

title('Signal X1[k] => DFT of Signal x1[n]'); ylabel('X1[k]'); xlabel('K');

grid; legend('Real', 'Imag.');

subplot(312);

stem(K, real(X2),'x','LineWidth', 2); hold on; stem(K, imag(X2),'o','LineWidth', 2);

title('Signal X2[k] => DFT of Signal x2[n]'); ylabel('X2[k]'); xlabel('K');

grid; legend('Real', 'Imag.');

subplot(313);

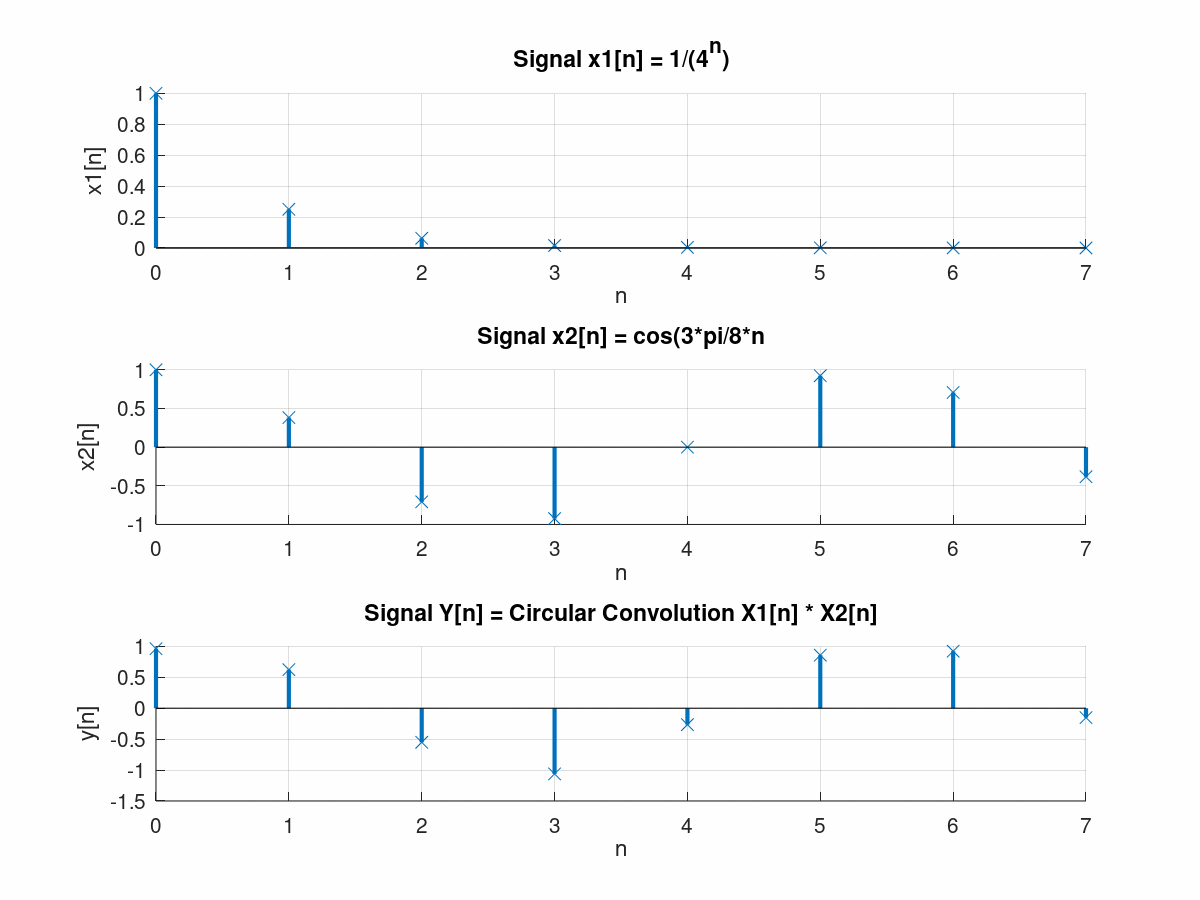
stem(K, real(X1.\*X2),'x','LineWidth', 2); hold on;

stem(K, imag(X1.\*X2),'o','LineWidth', 2);

title('Signal X3[k] = X1[k] x X2[k]'); ylabel('X2[k]'); xlabel('K');

grid; legend('Real', 'Imag.');

**OUTPUT – using Matrix Multiplication Method**



**OUTPUT – using DFT Method**

