

**Independent University Bangladesh**

Department of Electrical and Electronics Engineering

**Lab Report** **04**

Name: Injamamul Haque Sourov

Id: 1820170

Course code: EEE 321L

Couse name: Digital Signal Processing Lab

Lab no: 04

Lab title: Study of signal decomposition, downsampling, folding and shifting

Date: 08/12/2020

1. Function definitions
   1. Even and odd components

% function to determine even/odd component

function [xe,xo,m] = evenodd(x,n)

m = n;

xe = 0.5\*(x + fliplr(x)); % fliplr -> x(-n)

xo = 0.5\*(x - fliplr(x));

end

* 1. Down-sampling

% function to downsample a signal by factor M

function [y,m] = dnsample(n, x, M)

m = min(n./M):max(n./M);

y = x(1:length(m));

end

1. Signal operations
   1. Decomposition of x[n] = u(n) – u(n-10) where -10 ≤ n ≤ 10

Code:

% even/odd components

n = -10:10;

x = stepseq(0,-10,10) - stepseq(10,-10,10);

stem(n,x)

figure(2)

[xe,xo,m] = evenodd(x,n);

stem(m,xe)

figure(3)

stem(m,xo)

Outputs:

  
Figure: Original signal

  
Figure: Even component

  
Figure: Odd component

* 1. Down-sampling x[n] = sin(0.125\*pi\*n) where -50 ≤ n ≤ 50

Code:

% downsampling

n = -50:50;

x = sin(0.125\*pi\*n);

stem(n,x)

figure(2);

[y,m] = dnsample(n,x,4);

stem(m,y)

Output:

  
Figure: Original signal

  
Figure: Down-sampled signal

* 1. Time reversal of x[n] = u(n-2) for -10 ≤ n ≤ 10

Code:

% find x[-n] given that x[n] = u(n-2)

n = -10:10;

x = stepseq(2,-10,10);

stem(n,x)

figure(2)

y = fliplr(x);

stem(n,y)

Output:

  
Figure: Original signal x(n)

  
Figure: Time reversed signal x(-n)

1. Assignment
   1. Function definition

% function to generate x(n^2) from x(n)

function [y, m] = n\_square(x, n)

m = []; % domain

y = []; % range

c = abs(min(n)); % to set reference value (0)

% iterate over positive domain (>0) for square numbers

for i=1:max(n)

r = sqrt(i);

if floor(r) == r % check for perfect square

% append m (absolute) and y (relative to reference)

m(end+1) = r;

y(end+1) = x(c+i+1);

end

end

% correct for negative and 0 positions

m = [-fliplr(m) 0 m];

y = [fliplr(y) x(c) y];

end

* 1. Calls and outputs

Example 1

% x(n) = u(n+1) + u(n-2), -5:5

n = -5:5;

x = stepseq(-1,-5,5) + stepseq(2,-5,5);

stem(n,x); grid;

[y, m] = n\_square(x,n);

figure(2); stem(m,y); grid;

  
x(n)

  
 x(n2)

Example 2

% x(n) = rampseq delayed by -3, -10:10

n = -10:10;

x = rampseq(-3,-10,10);

stem(n,x); grid;

[y, m] = n\_square(x,n);

figure(2); stem(m,y); grid;

  
x(n)

  
 x(n2)

\*\* Some references show only **positive** and **square** values on of *n* (1, 4, 9 instead of 1, 2, 3), to do so change the negative sign before the *fliplr* should be removed and the m array should be appended with *i* instead of *r* in the function definition*.* Example 2 output in this format is shown below.

  
x(n)

  
 x(n2)