Week -8 (Signals & Systems) –Part 2

➢ Defining/Creating the Discrete-time Signal: {Use "inline" and "stem" functions}

- **To achieve this increment chosen should be equal to "1" while defining array in time-axis.**

Consider a discrete time function: f [n] = $e^{-n/10}\cos(n\pi/10)u[n]$, can be expressed as,

>> f = inline('exp(-n/10).*cos(n*pi/5).* (n>=0) ','n') ------>defines the function for the signal variation for all value of n.

Sketching the discrete time function f[n] over presentation duration (-10 ≤ n ≥ 20).

>>n= [-10:20]; creates the range for running variables. (note: increment is 1).

>>stem(n,f(n),'k'); sketch of the above signal vs. discrete-time. Here, "n" and "f(n)" are arrays.

Constructing energy signal (finite duration signal) from everlasting signal

>>f= inline('exp(-n/10).*cos(n*pi/5).*((n>=0)&(n<11))','n')

creates a causal function with nonzero values up to n= 11.

>>stem(n,f(n/2),'k'); expand (interpolate) the above function by a factor 2

>>stem(n,f(n*2),'k'); compress (decimate) the above function by a factor 2

To calculate the Energy, Power, Even & Odd part of a signal:

- **Refer the method discussed in Lab manual Wee-3 with the increment chosen should be equal to "1" while defining array in time-axis.**

>>energy1=sum((f(n).*f(n))) % computes the energy of the signal using Method-1. OR,

>>x = exp(-n/10).*cos(n*pi/5).*((n>=0)&(n<11)) % signal definition with non-zero between -2 to 1.

>>energy2 = sum(x.^2) % computing energy. Method-2.

To find the ODD and EVEN components:

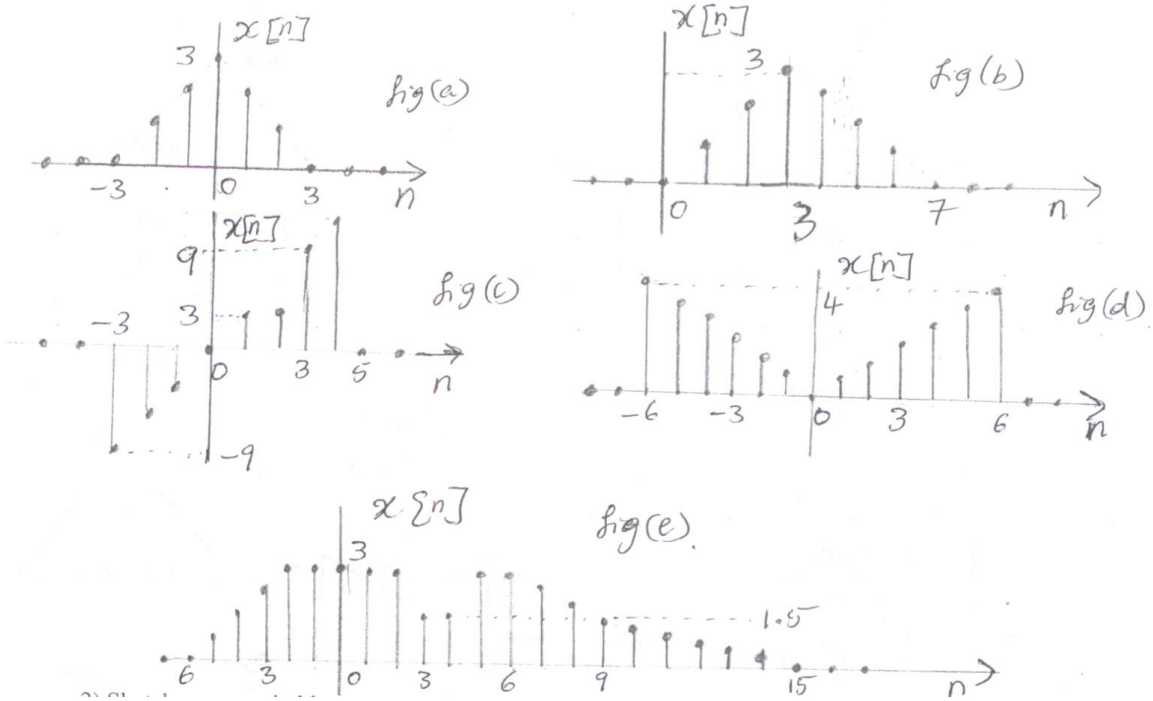
>> xe = 0.5*(f(n)+f(-n)); >> xo = 0.5*(f(n)-f(-n)); % computing even and odd part of signal respectively,

>>stem(t,xe); >> stem(t,xo); %sketch of the above computed even and odd function.

Note: **Care should be taken in selecting the presentation duration i.e. n=[-12:12], such a way that full duration of odd/even components are presented.**

Exercises

1) Plot the signals shown in **fig a) to fig e),** over a suitable range. Also calculate the size of the signals and find x(-n), x(n+6), x(n-6), x(3n), x(n/2), -3x(-0.2n+12)+5, x(3-n)



2) Sketch over a suitable range and comment on the signal class and size of the following signals:

$(1)^n$, $(-1)^n$, u[n}, $(-1)^n$u[n], $\cos\left[\frac{\pi}{3}n+\frac{\pi}{6}\right]$.

3) Sketch over a suitable range, the following signals, find out odd and even components.

a) $\frac{1}{2}2^n \sin\left(\frac{\pi n}{3}\right)$, **b)** n(n-1) υ^n u[n-2] for υ = 0.8 and -0.8. **c)** (u[n]+$(-1)^{n+1}$u[n]) $\cos\left(\frac{\pi n}{2}\right)$,

d) n{u[n] – u[n-7] }, **e)** (-n+8){u[n-6] – u[n-9] }, **f)** (n-2){u[n-2] – u[n-6] }+(-n+8){u[n-6] – u[n-9] }

4) Consider the signal y(t)= –(1/2)x(-3n+2) shown in Fig. a),

a) Determine x(n) as a function of y and n. Sketch the original signal x(n).

b) From the sketch, identify the key coordinate points. Then manually derive the equation to the line to reconstruct y(n).

c) Determine and sketch the even and odd portion of the original signal x(n).

5) Draw all possible cases of exponentials signals γ^nu(n) using subplots in a single figure window, where, λ = r/Ω, with r and Ω being real and positive. Also, provide commentary on the resulting signal patterns.

Week -9 (Discrete Time Domain Analysis) – Part 2

Method 1: Solving Difference equation using Iterative procedure to find Total Response:

Ex: Given the difference equation, y[n+2] – y[n+1] + 0.24y[n] = x[n+2] – 2x[n+1] and with initial conditions, y[-1] = 2 ; y[-2] = 1, and with forcing function x[n] = n u(n) (starting at n = 0).

Following is the MATLAB code:

```
n = [-2:10]';                          % defining the discrete time range for getting the solution.
y = [1 ; 2 ; zeros(length(n) -2 , 1)]; % predefining the output array with first two elements are initial
                                       conditions and subsequent elements are zeros.
x=[ 0; 0 ; n(3 : end)];                % defining the input array with first two entries corresponds to the
                                       time instants -2 and -1, which are zeros.
for k = 1 : length(n) -2               % Iterative computation starts here
  y(k+2) = y(k+1) - 0.24*y(k) + x(k+2) - 2*x(k+1);
end;

% NOTE : Since, y(1) and y(2) have already assigned with initial conditions, when k = 1, y(3) value is
         computed indicating the first member of the solution.

stem(n , y , 'k');                     % sketch of the solution including the initial conditions.
```

O/P -----> y = [1.0000 2.0000 1.7600 2.2800 1.8576 0.3104 -2.1354 - 5.2099 - 8.6974 -12.4470 - 16.3597 -20.3724 -24.4461]

- *For finding the zero-input response, do not consider the feed forward coefficients (the coefficients of terms corresponding to x[n] and its advance operations), in the difference equation*. OR *define the input array "x" as zeroes of respective matrix size.*
- *For finding the zero-state response, define the initial conditions as zeroes in array "y"*
- *For finding the impulse response, define the input array as impulse and* initial *conditions as zeroes. i.e.* x=[0; 0 ; 1 ; zeros(length(n)-3 , 1)];

Method 2: Solving Difference equation using "filter" command in MATLAB:

MATLAB's *filter* command provides an efficient way to evaluate the system response to a constant coefficient (time-invariant) linear difference equation represented in delay form as,

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{N} b_k x[n-k]$$

Filter requires three input arguments:

i. a length N+1 vector of feed forward coefficients $[b_o, b_1, b_2, \ldots b_N]$,
ii. a length N+1 vector of feedback coefficients $[a_o, a_1, a_2, \ldots a_N]$, and
iii. an input vector
iv. Optional 4th argument: initial state of the system should be appropriately inserted.

Finding Impulse Response: (using filter command)

For the same above example, following is the MATLAB code:

```
N = 2; b = [1 -2 0]; a = [1 -1 0.24];          %defining difference equation parameters.
n = [0 : 30]' ; delta = inline('n == 0','n');   %defining impulse function as input signal.
h = filter(b,a,delta(n));      stem(n,h);       % "filter" gets the solution to difference equation.
```

Result: h = {1 -1 -1.24 -1 -0.702 -0.462 -0.294 -0.183 -0.1123 .....}

NOTE : 'filter' function gives first N (i.e. order of system) values of the solution as auxiliary conditions, which is generally counted from running variable n = 0 to N-1. In the above example, first two values {1 and -1} of the solution are considered as auxiliary conditions of the solution.

Finding Zero –State Response: (using filter command)

For the same above example, following is the MATLAB code:

```
N = 2; b= [1 -2 0]; a= [1 -1 0.24];          %defining difference equation
n=[0:30]';     x = inline(' n ',' n ')       %defining input function
ys= filter(b,a,x(n));   stem(n,ys)
```

Result : ys = {0, 1, 1, -0.24, -2.48, -5.42, -8.82, -12.5258 . .....}

NOTE : "filter" function can't be used in MATLAB to find the ZIR of the difference equation, as the initial conditions can't be accommodated directly. One may use the iterative method discussed above and it may be added to ZSR to get the total response.

Method 3 : Convolution of two signals using 'conv' command

Case 1): Convolution of two finite-duration discrete time signals is accomplished by using the *conv* command.

"conv" command does not give the output with the region of support in the discrete-time axis.

Eg: 1) (u[n] - u[n-4) * (u[n] - u[n-4)

>> conv([1 1 1 1] , [1 1 1 1]) gives O/P as ----> 1 2 3 4 3 2 1

Regions of support in time axis is ($0 \le n \le 6$) is to be comprehended manually (7 elements = 4 + 4 - 1).

2) (u[n+4] - u[n) * (u[n] - u[n-4)

>> conv([1 1 1 1] , [1 1 1 1]) gives O/P as ----> 1 2 3 4 3 2 1

Regions of support in time axis is ($-4 \le n \le 2$) is to be comprehended manually (graphically supported).

The region of support for thus obtained output may be comprehended using the width property.

Case 2): In general, the *conv* command should be carefully used to convolve infinite – duration signals. *Conv* command can correctly compute a portion of such infinite duration signals.

By passing the first N samples of each signal, *conv* command returns a length (2N-1) member sequence. The first N samples are computed correctly; the remaining N-1 samples are wrong results.

For the difference equation mentioned in the above example, the impulse response (calculated analytically) ffor the system is $h[n] = \{-7\ (0.6)^n + 8\ (0.4)^n\}\ u[n]$, hence by defining it as the inline function and calling it for discrete-time range n = [0 : 30] ' as:

```
>> h = inline('(-7*0.6.^n+8*0.4.^n).*(n>=0)','n');
>> x = inline(' n ',' n ');
>> ytc= conv(h(n),x(n));
>> stem ( [0:60] , ytc ,'k')      Compare the result with earlier method.
```

In the result, $0 \le n \le 30$, output samples are correctly computed. The remaining $31 \le n \le 61$ output sample values are incorrectly computed. Theoretically, the output envelope should continue to grow as n tends to infinity, not decay. If "conv" command is used to convolve infinite – duration signals, normally incorrect values should be omitted and are not used for display as:

```
>> stem(n,ytc(0:30),'k')  will be identical to the earlier result.
```

Method 4 : Convolution of two signals: (using Graphical Method)

Try to modify the MATLAB code to convolve two discrete time signals graphically and use it. (Refer the MATLAB E code discussed in "Week-5 (Time Domain Analysis) –ZSR").

Hint: one may use "stem" instead of plot. Also, stem sketches only one signal in one axis. One may use 3 subplots one for $x(\tau)$, second for $h(n-\tau)$ vs. τ and third for $y(n)$ vs. n. And Increment should be 1.

In this case, irrespective of the signal duration, finite or infinite, one can get the output results with the regions of support in time axis. This method does give the output with the region of support in the time axis.

Exercises

{Note: Expectation is that having understood concepts in continuous time, same can be interpreted here with minor differences}

1) Solve the following difference equations manually, to find the Impulse Response, Zero–Input Response. Try to get the results **iteratively** (20-100 samples) and also using "**filter**" (as applicable) command. Choose appropriate initial conditions as y[-1] = -1 ; y[-2] = 2; y[-3] =1.

1) $y[n+2] - 1.9\, y[n+1] + 0.9y[n] = 2x[n] - 3x[n-1] + x[n-2]$

2) $y[n+2] - 1.732\, y[n+1] + y[n] = 3x[n] + 2x[n-1]$ AND $y[n+2] + 1.732\, y[n+1] + y[n] = 3x[n] + 2x[n-1]$

3) $y[n+2] - 1.6742\, y[n+1] + 0.7225\, y[n] = 3x[n]$ AND $y[n+2] + 1.6742\, y[n+1] + 0.7225\, y[n] = 3x[n]$

4) $y[n+2] - 1.9\, y[n+1] + 0.9025\, y[n] = 2x[n] + x[n-1]$ AND $y[n+2] + 1.9\, y[n+1] + 0.9025\, y[n] = 2x[n] + x[n-1]$

5) $(E^3 - 2.9\, E^2 + 2.8E - 0.9)y[n] = Ex[n]$

a) Sketch the output responses and comment on the nature of the responses (belongs to which class of signal model). Find the steady state values.

b) Comment (where ever applicable) on the Nature (characteristic polynomial, characteristic roots, characteristic modes), stability, and find the number of samples (in terms of sample count) to reach steady state (if applicable) of the system.

c) Find the period of oscillation (samples per cycle) of output response (if applicable).

2) Solve the following difference equations manually, to find the zero state response, total response. Try to get the results **iteratively** (20-100 samples) and also using "**filter**" (as applicable) command. Choose appropriate initial conditions as y[-1] = -1 ; y[-2] = 2; y[-3] =1.

1) $y[n+2] - 1.9\, y[n+1] + 0.9y[n] = 2x[n] - 3x[n-1] + x[n-2]$ with the input $x[n] = u[n]$, AND $x[n] = 0.9^n u[n]$

2) $y[n+2] - 1.732\, y[n+1] + y[n] = 3x[n] + 2x[n-1]$ with the input $x[n] = u[n]$, AND $x[n] = \cos(\pi n/6)u[n]$

3) $y[n+2] - 1.6742\, y[n+1] + 0.7225\, y[n] = 3x[n]$ with the input $x[n] = u[n]$, AND $x[n] = 0.85^n \cos(\pi n/18)\, u[n]$

4) $y[n+2] - 1.9\, y[n+1] + 0.9025\, y[n] = 2x[n] + x[n-1]$ with the input $x[n] = u[n]$, AND $x[n] = 0.95^n u[n]$

5) $(E^3 - 2.9\, E^2 + 2.8E - 0.9)y[n] = Ex[n]$ with the input $x[n] = u[n]$, AND $x[n] = 0.9^n u[n]$

Observe how the natural behavior changes with the application of forcing functions.

3) Find the zero state response of the LTID system, whose following unit impulse response h[n] and input x[n].

1) $h[n] = (-2)^n u[n-1]$; $x[n] = e^{-n} u[n+1]$

2) $h[n] = \frac{1}{2}\{\delta[n-2] - (-2)^{n+1}\}u[n-3]$; $x[n] = 3^{n-1} u[n+2]$

3) $h[n] = \{(2)^{n-2} + 3(-5)^{n+2}\}u[n-1]$; $x[n] = 3^{n+2} u[n+1]$

4) $h[n] = 3(n-2)(2)^{n-3}\, u[n-4]$; $x[n] = 3^{-n+2}\, u[n+3]$

5) $h[n] = (3)^n \cos(\pi/3 - 0.5)u[n]$; $x[n] = 2^n\, u[n-1]$

4) Use classical method, to solve the equations given in exercise 1), and 2), compare the results with the iterative methods.