Digital Signal Processing

Lab6: Exercises

Instructor: Eng\ Samar Shaaban
E-mail: ssa10@fayoum.edu.eg
Github Repo: https://github.com/SamarShabanCS/DSP
Slack workspace: https://fayoum-university-fci.slack.com

Exercise One

- Implement a generic function that can generate both Unit sample sequence:

$$\delta(n-n_0)=\begin{cases}1, & n=n_0\\ 0, & n\neq n_0\end{cases}$$

and Unit step sequence:

$$u(n-n_0)=\begin{cases}1, & n\geq n_0\\ 0, & n<n_0\end{cases}$$

over the n1 ≤ n0 ≤ n2 interval.

- `function [x,n] = imp_step_seq(n0,n1,n2,funcName)`

Exercise One cont…

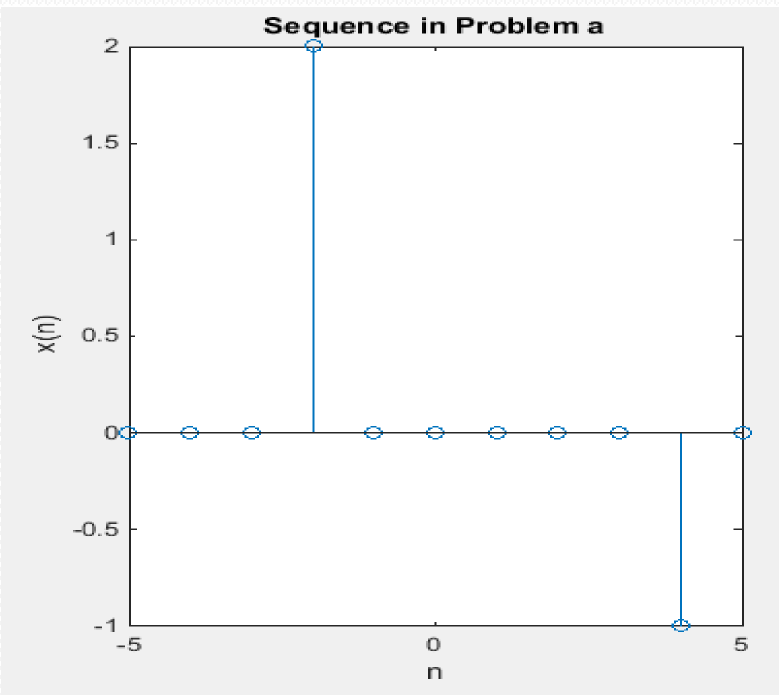
```
function [x,n] = imp_step_seq(n0,n1,n2,funcName)
 n = n1:n2;
if(strcmp(funcName,'impluse'))
 x = ( n - n0) == 0;
elseif(strcmp(funcName,'step'))
 x = (n - n0) >= 0;
end
end
```

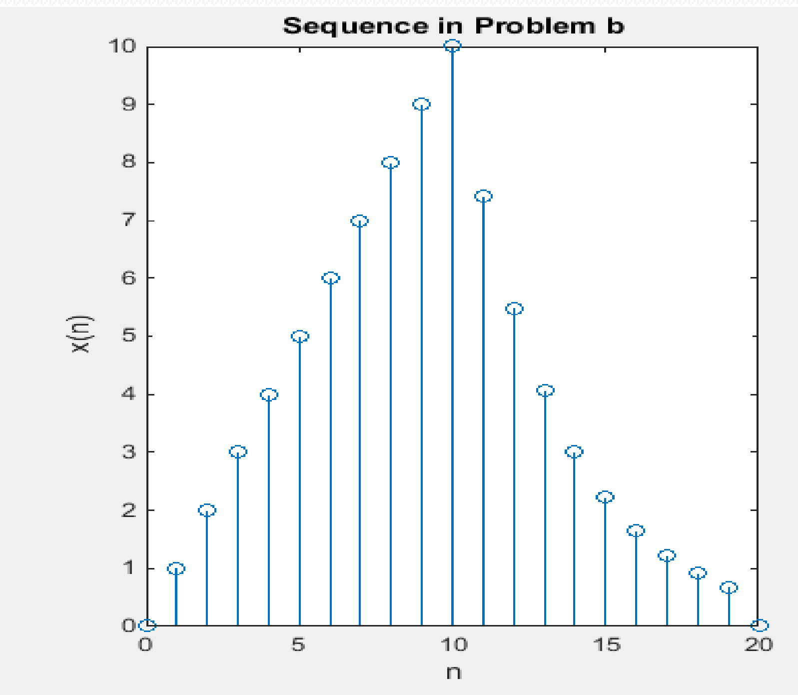
Exercise Two

- Test the previous function by generating and plotting each of the following sequences over the indicated interval:

a. $x(n) = 2\delta(n+2) - \delta(n-4), \quad -5 \le n \le 5.$

b. $x(n) = n[u(n) - u(n-10)] + 10e^{-0.3(n-10)}[u(n-10) - u(n-20)], 0 \le n \le 20.$





Exercise Two cont…

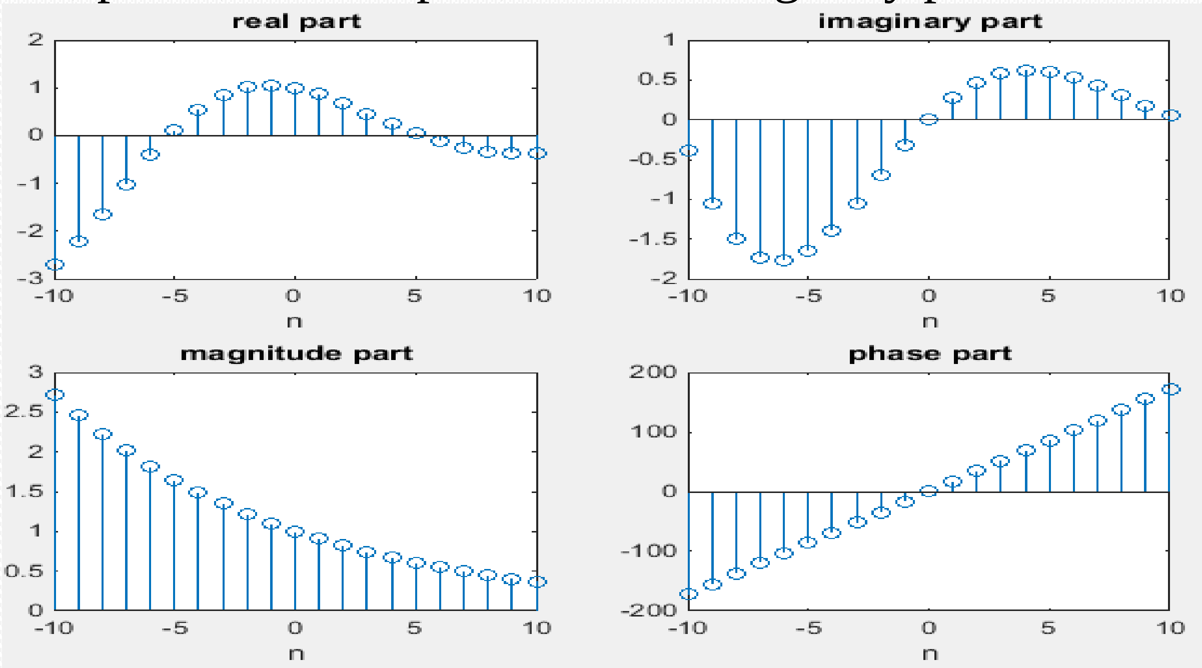
```
% x(n) = 2\delta(n + 2) - \delta(n - 4), -5 <= n <= 5
x = 2*imp_step_seq(-2,-5,5,'impluse') - imp_step_seq(4,-5,5,'impluse');
n = -5:5;
subplot(1,2,1);
stem(n,x);
title('Sequence in Problem a')
xlabel('n');
ylabel('x(n)');
% x(n) = n [u(n) - u(n - 10)] + 10e^(0.3(n-10)) [u(n - 10) - u(n - 20)], 0 <= n <= 20
n = 0:20;
x1 = n .*(imp_step_seq(0,0,20,'step') - imp_step_seq(10,0,20,'step'));
x2 = 10*exp(-0.3*(n-10)).*(imp_step_seq(10,0,20,'step')-imp_step_seq(20,0,20,'step'));
x = x1 + x2;
subplot(1,2,2);
stem(n,x);
title('Sequence in Problem b')
xlabel('n');
ylabel('x(n)');
```

Exercise Three

Generate the complex-valued signal

$$x(n) = e^{(-0.1+j0.3)n}, \quad -10 \le n \le 10$$

and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.



Exercise Three cont…

```
n = -10:1:10;
alpha = -0.1+0.3j;
x = exp(alpha*n);
subplot(2,2,1);
stem(n,real(x));
title('real part');
xlabel('n')
subplot(2,2,2);
stem(n,imag(x));
title('imaginary part');
xlabel('n')
subplot(2,2,3);
stem(n,abs(x));
title('magnitude part');
xlabel('n')
subplot(2,2,4);
stem(n,(180/pi)*angle(x));
title('phase part');
xlabel('n')
```

Exercise Four

- A real-valued sequence $x_e(n)$ is called even (symmetric) if $x_e(-n) = x_e(n)$

  Similarly, a real-valued sequence $x_o(n)$ is called odd (antisymmetric) if $x_o(-n) = -x_o(n)$

Then any arbitrary real-valued sequence x(n) can be decomposed into its even and odd components

$$x(n) = x_e(n) + x_o(n)$$

where the even and odd parts are given by

$$x_e(n) = \frac{1}{2}\left[x(n) + x(-n)\right] \quad \text{and} \quad x_o(n) = \frac{1}{2}\left[x(n) - x(-n)\right]$$

- Generate a function that decomposes a signal into its even and odd components

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

Exercise Four cont…

```
function [xe, xo, m] = evenodd(x,n)
% Real signal decomposition into even and odd parts%...%
if any(imag(x) ~= 0)
error('x is not a real sequence')
end
subplot(2,2,1);
stem(n , x);
m = -fliplr(n);
m1 = min([m,n]);
m2 = max([m,n]);
m = m1:m2;
nm = n(1)-m(1);
n1 = 1:length(n);
x1 = zeros(1,length(m));
x1(n1+nm) = x;
x = x1;
xe = 0.5*(x + fliplr(x));
xo = 0.5*(x - fliplr(x));
subplot(2,2,2);
stem(m,xe);
subplot(2,2,4);
stem(m,xo);
```

Exercise Four cont…

- Let $x(n) = u(n) - u(n - 10)$. Decompose $x(n)$ into even and odd components