**INSTITUTO FEDERAL DO ESPIRITO SANTO**

**Curso de Engenharia Elétrica**

**ANÁLISE DE SINAIS E SISTEMAS - 2023-1**

# PRÁTICA 5

1 –Usando as condições iniciais y[-1] = 2 e y[-2] = 1, determinar e traçar a resposta de entrada nula para o sistema descrito por: **y[n]-1,56y[n-1]+0,81y[n-2]=x[n-1]+3x[n-2]**

Usar como referência os exemplos: 8y[n] - 2y[n-1] - y[n-2] = x[n] + x[n-1] com

a) y[-1] = 1  y[-2] = 0

```
n = 0:10;
a = [8 -2 -1];% Q[E]
b = [1 1 0];  % P[E]
yi = [1 0];   % yi=[y[-1] y[-2]] para n = 0:10
xi = 0;       % Sem condições iniciais para x[n]
x=zeros(1,size(n,2)) % entrada nula
%x=2*n;       % exemplo entrada do sistema (não nula)
zi = filtic(b,a,yi,xi)
y = filter(b,a,x,zi)
stem(n,y)
```

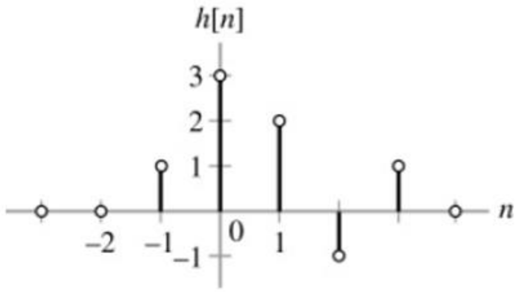
b) y[0] = 1  y[-1] = 0

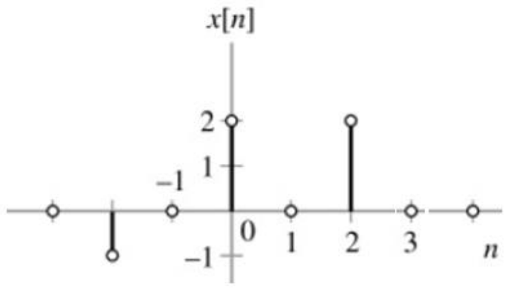
```
n = -1:10;
a = [8 -2 -1];% Q[E]
b = [1 1 0];  % P[E]
yi = [1 0];   % yi=[y[0] y[-1]] para n = -1:10
xi = 0;       % Sem  condições iniciais para x[n]
x=zeros(1,size(n,2)) % entrada nula
%x=2*n;       % exemplo entrada do sistema (não nula)
zi = filtic(b,a,yi,xi)
y = filter(b,a,x,zi)
stem(n,y)
```

2 – Utilize o comando <**ztrans**> e a função <**syms**> para transformar função da **questão 3** para o domínio de Z.

3 - Para os sinais fornecidos abaixo façam a convolução **y=x*h**, utilizando o comando CONV. Plote o resultado com a função STEM.

Utilizando os conceitos de Comvolução Gráfica, Crie um vetor de "n" compatível com o resultado de convolução.



4 – Os sinais discretos podem ser transformados para o "domínio da frequência", pela Transformada de Fourier no Tempo Discreto (FTDT)

Logo: a transformada de tempo contínuo

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-i2\pi ft}\, dt.$$

se transforma em

$$X_T(f) \stackrel{\text{def}}{=} \sum_{k=-\infty}^{\infty} X(f - kf_s) \equiv T \sum_{n=-\infty}^{\infty} x(nT)\, e^{-i2\pi fTn}.$$

Onde:

$$\omega = 2\pi fT = 2\pi \left(\frac{f}{f_s}\right).$$

Mas... NÃO É POSSÍVEL se obter, em um software, um **somatório infinito**, então a Transformada de Fourier no Tempo Discreto (DTFT) **foi modificada** para ser a Transformada Discreta de Fourier (DFT) que é finita.

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-i2\pi \frac{k}{N} n}$$

No Matlab essa função é implementada pela Transformada Rápida de Fourier (FFT) que é uma versão que substitui o tradicional e lento **Loop** por um algoritmo chamado de "Butterfly".

A função <fft>, então, é utilizada para fazer a Transformada de Fourier computacional de um vetor temporal de dados como pode ser visto no exemplo abaixo:

```
% Esse Exemplo faz a Transformada Discreta de Fourier de um sinal Periódico
% criado pela soma de 3 senóides
clc; clear all; close all
%% Senoides e aliasing
% Taxa de amostragem
fs=1000; Ts=1/fs;
% Frequencias, em Hz, do sinal
f1=20; DigFreq1=2*pi*f1/fs;
f2=30; DigFreq2=2*pi*f2/fs;
f3=40; DigFreq3=2*pi*f3/fs;
% O Sinal
N=1500; % Amostras
n = 0:1:N-1;
t_sample = [0 : Ts : (N-1)*Ts];
x=3.*cos(DigFreq1.*n)+cos(DigFreq2.*n)+2.*cos(DigFreq3.*n);
```

```
figure(1);
stem(t_sample,x)

%% FFT do sinal
Xf=fft(xn);
% os valores de xn são valores reais, enquanto os valores de Xf são
% complexos. Veja alguns exemplos:
xn(2:6)
Xf(30:34)
%Xf são valores complexos porque representam magnitude e fase!!!!
%Magnitude
Xf_mag=abs(Xf);
Xf_mag(30:34)
figure(2)
plot(Xf_mag)
```

Porém, o ESPECTRO DE FOURIER aparece na faixa de 0 a N [Hz], e essa é uma "aparência" complicada de se analisar, logo utiliza-se a função `<fftshift>` para colocar a frequência ZERO no centro do Gráfico, conforme mostra o algoritmo abaixo:

```
 clear all; close all; clc
%%
Tmax=0.5; % Intervalo de duração de cada onda
fs=200; % Frequência de amostragem
t=[0:(1/fs):Tmax+2]; % Amostragem no tempo
L=length(t);
% Pulso retangular
T0=0; % Instante de início do pulso retangular
T=Tmax; % Duração do pulso retangular
% Definição do início e final da janela
L_ini=length([0:(1/fs):T0]);
L_pulse=length([0:(1/fs):T]);
L_fin=L-L_ini-L_pulse;
win = rectwin(L_pulse);
wRect1 = [zeros(L_ini,1); win; zeros(L_fin,1)]';
figure(1)
subplot(311)
plot(t,wRect1,'LineWidth',1)
grid on
xlabel('Tempo(s)');
ylabel('Amplitude');
X=fft(wRect1);
subplot(312)
plot(t,abs(X))
grid on
xlabel('bins');
ylabel('Amplitude');
Y=fftshift(X);
```

```
subplot(313)
plot(t,abs(Y))
ylim([0 100])
grid on
xlabel('bins');
ylabel('Amplitude');
```

Contudo, o EIXO da frequência não condiz com os valores reais de frequência amostrados, logo o algoritmo abaixo demonstra como ajustar o eixo da frequencia para mostrar a frequencia real do espectro:

```
Fs = 1000;                    % Sampling frequency
T = 1/Fs;                     % Sampling period
L = 1000;                     % Length of signal
t = (0:L-1)*T;                % Time vector
%%Create a matrix where each row represents a cosine wave with scaled frequency. The result, X, is a 3-by-1000 matrix. The first row has a wave frequency of 50, the second row has a wave frequency of 150, and the third row has a wave frequency of 300.

x1 = cos(2*pi*50*t);          % First row wave
x2 = cos(2*pi*150*t);         % Second row wave
x3 = cos(2*pi*300*t);         % Third row wave

X = [x1; x2; x3];
%%Plot the first 100 entries from each row of X in a single figure in order and compare their frequencies.
figure(1)
for i = 1:3
    subplot(3,1,i)
    plot(t(1:100),X(i,1:100))
end


dim = 2;
%%Compute the Fourier transform of the signals.

Y = fft(X,L,dim);
%%Calculate the double-sided spectrum and single-sided spectrum of each signal.

P2 = abs(Y/L);
P1 = P2(:,1:L/2+1);
P1(:,2:end-1) = 2*P1(:,2:end-1);
```

```
%%In the frequency domain, plot the single-sided amplitude spectrum for
each row in a single figure.
figure(2)
for i=1:3
    subplot(3,1,i)
    plot(0:(Fs/L):(Fs/2-Fs/L),P1(i,1:L/2))
end

Fs = 100;
t = 0:1/Fs:1-1/Fs;
f1=15; % Hz
f2=40; % Hz
x1 = cos(2*pi*f1*t - pi/4);
x2 = cos(2*pi*f2*t + pi/2);
x = x1+x2;

%%Compute the Fourier transform of the signal. Plot the magnitude of the
transform as a function of frequency.
y = fft(x);
z = fftshift(y);
figure(3)
ly = length(y);
f = (-ly/2:ly/2-1)/ly*Fs;
subplot(2,1,1)
plot(t,x)
subplot(2,1,2)
stem(f,abs(z))
title('Double-Sided Amplitude Spectrum of x(t)')
xlabel('Frequency (Hz)')
ylabel('|y|')
Grid
```

**De posse de todos esses dados, some as frequências 60, 120 e 180 Hz e mostre, fazendo uso do algoritmo anterior, o espectro de frequência do sinal final.**