LAB003 - Python em SS:

Série de Fourier de Tempo Discreto:

A Série de Fourier de Tempo Discreto ou DTFS (*Discrete Time Fourier Series*) é a única representação de Fourier que tem valores discretos tanto no tempo com na frequência.

$$x[n]=\sum_{k=\langle N\rangle} X[k]e^{jk\omega_0 n} \overset{DTFS:\omega_0}{\longleftrightarrow} X[k]=\frac{1}{N}\sum_{n=\langle N\rangle} x[n]e^{-jk\omega_0 n}$$

Assim podem ser implementadas diretamente no computador.

O comando *fft* e *ifft* do modulo *numpy.fft* pode ser usado para avaliarmos os DTFS e sua inversa. Dado um vetor *x* de tamanho *N* que representa um período de um sinal com período *N*, *x*[*n*]:

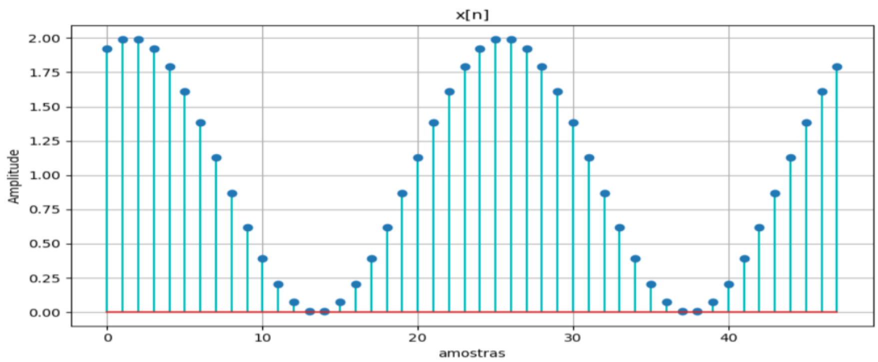
- O comando *X* = numpy.fft.fft(x)/N
  - Implementa a DTFS para o sinal *x* e *X* contém os coeficientes da DTFS.
- O comando *x* = numpy.fft.ifft(X)*N
  - Retorna o vetor *x* no tempo

Note que tanto a *fft* e a *ifft* os vetores devem ser divididos ou multiplicados por N.

Exemplo: Encontre a representação por DTFS para

$$x[n]=1+sen\left(\frac{\pi}{12}n+\frac{3\pi}{8}\right)$$

Este sinal tem período *N* = 24



```
import numpy as np
import matplotlib.pyplot as plt
from numpy import pi
from numpy.fft import fft, ifft, fftfreq, fftshift

# criando o vetor de amostra com tamanho

# nper  períodos do sinal
N = 24          # período do sinal
nper = 10       # quantidade de períodos em x[n]
n = np.arange(0,nper*N)

#criando o vetor do sinal x[n]
xp1 = np.ones(len(n))
x = xp1 + np.sin( ((pi/12)*n)+(3*pi/8) )

# calculando a DTFS

X = fft(x)/len(x)

#criando o vetor de frequência

 w = fftfreq(len(n), d=1/N)

# posicionando a freq. zero no meio do gráfico

Xd = fftshift(X)
w = fftshift(w)

# calculando o modulo - magnitude do espectro

ModX = np.abs(Xd)

# calculando a fase do espectro

phasX = np.angle(Xd)

# devido a erros de arredondamentos numéricos da fft devemos
# filtrar os sinais muito pequenos!

phasX[ModX < 0.00001] = 0

# retornando o sinal ao domínio do tempo

xr = ifft(X)*len(x)

# ignorando os erros de arredondamento do fft e ifft

xr = np.real(xr)

# A partir daqui é plotar os gráficos.
```

Os gráficos:

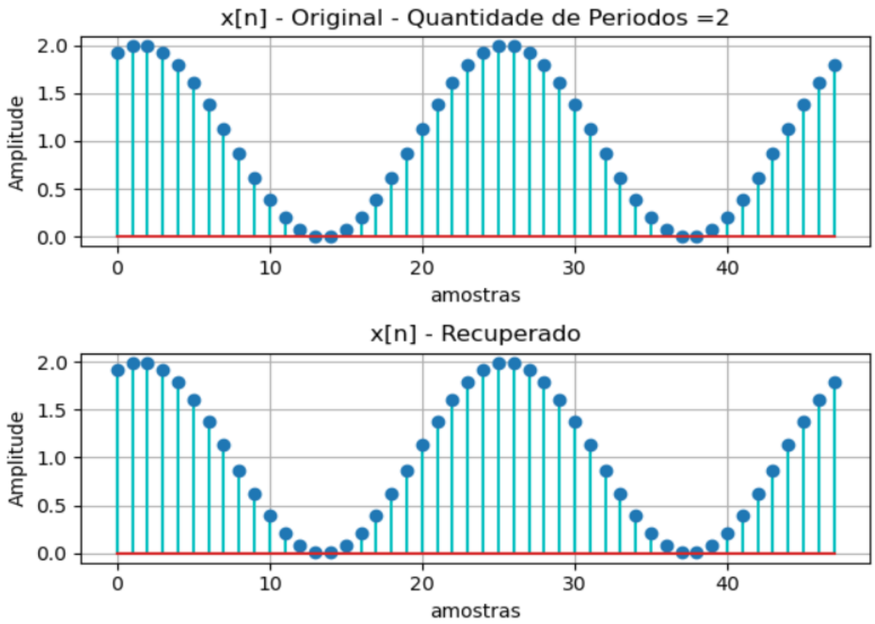


Figura 1 – Sinal original e sinal recuperado

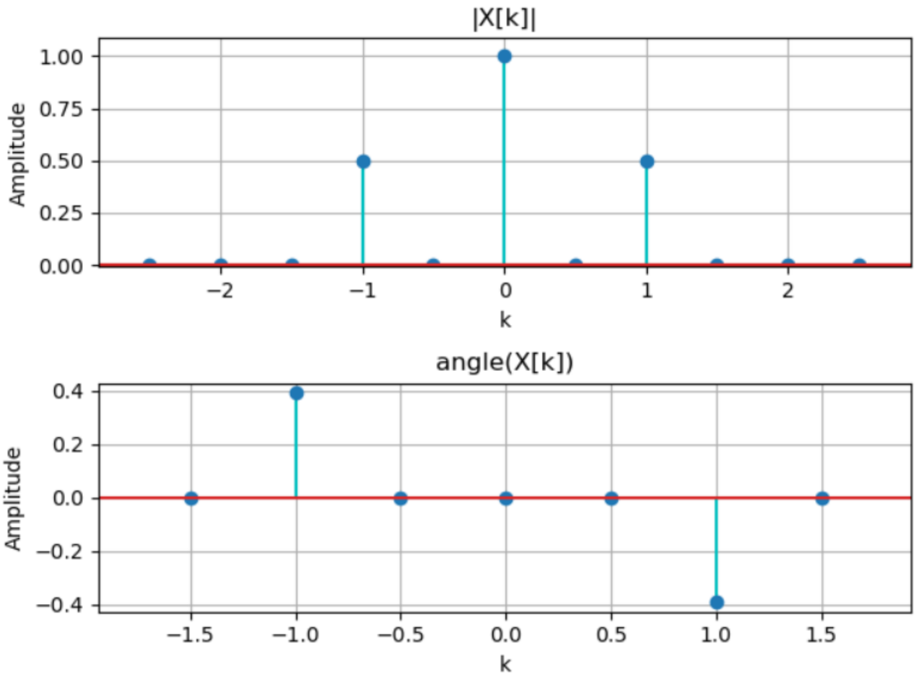
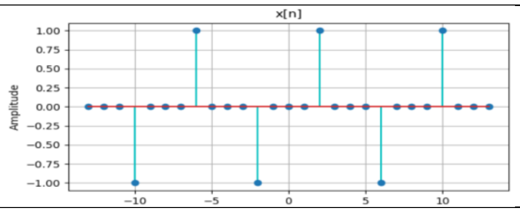
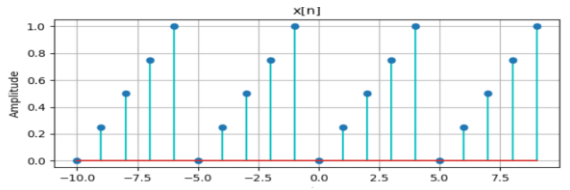
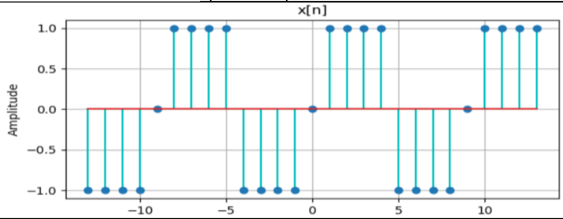


Figura 2 – Gráficos de magnitude e fase dos coeficientes de Fourier do sinal

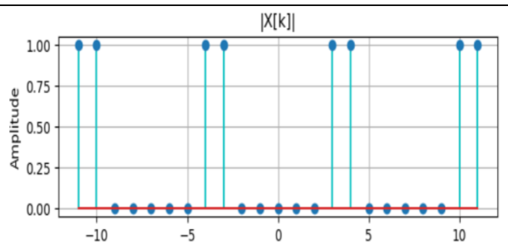
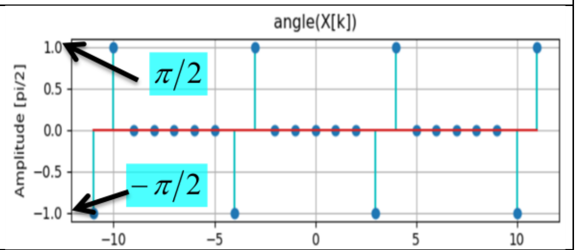
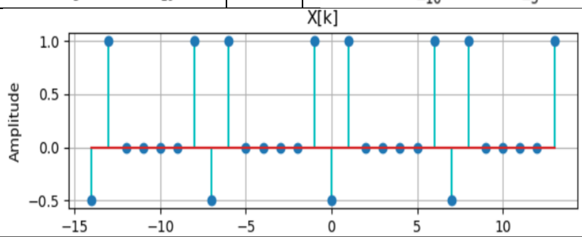
- O código completo em Python é o Exemplo_DTFS.py

Exercícios 1 e 2:

1. Esboce os espectros de magnitude e fase de:

<table>
<tr><td>a)</td><td>$x[n] = cos\left(\frac{6\pi}{13}n + \frac{\pi}{6}\right)$</td><td>b)</td><td>$x[n] = sen\left(\frac{4\pi}{21}n\right) + cos\left(\frac{10\pi}{21}n\right) + 1$</td></tr>
<tr><td>c)</td><td></td><td>d)</td><td></td></tr>
<tr><td>e)</td><td colspan="3"></td></tr>
</table>

2. Esboce os sinais no domínio do tempo:

<table>
<tr><td>a)</td><td>$X[k] = cos\left(\frac{6\pi}{17}k\right)$</td><td>b)</td><td>$X[k] = cos\left(\frac{10\pi}{21}k\right) + jcos\left(\frac{4\pi}{21}k\right)$</td></tr>
<tr><td>c)</td><td></td><td>d)</td><td></td></tr>
<tr><td>e)</td><td colspan="3"></td></tr>
</table>

A Série de Fourier

A série de Fourier - FS (Fourier Series) também pode ser calculada através das funções *fft* e *ifft.* Mas como o sinal no tempo está no domínio contínuo devemos ter cuidados ao amostrar o sinal no domínio do tempo.

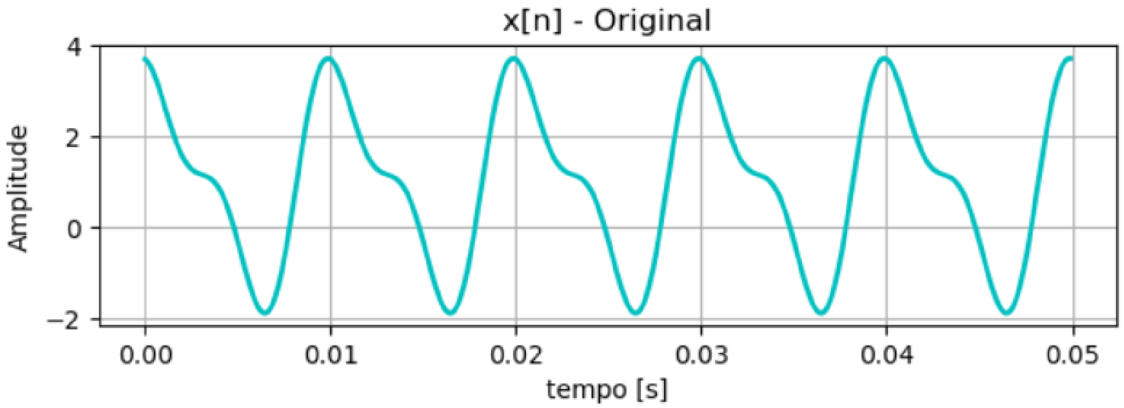
Analise das condições necessárias para uma amostragem correta do sinal será feita posteriormente em outro modulo.

Para as atividades deste modulo vamos garantir que o sinal contínuo esteja corretamente amostrado fazendo que o período de intervalo do sinal contínuo seja pelo menos 10 vezes menor que o período fundamental do sinal.

Exemplo: Encontre a representação por FS para

$$x(t) = 1 + sen(\omega_0 t) + 2cos(\omega_0 t) + cos\left(2\omega_0 t + \frac{\pi}{4}\right)$$

Com $\omega_0 = 200\pi$ rad/s



```
import numpy as np
import matplotlib.pyplot as plt
from numpy import pi
from numpy.fft import fft, ifft, fftfreq, fftshift

# definindo o sinal continuo periódico
fs = 100          # freq. do sinal periódico
w0 = 2*pi*fs      # frequencia angular
Ts = 1/fs         # período fundamental do sinal
Tam = Ts/100         # período de amostragem 100 vezes menor que o
período do sinal

# Criando o vetor de tempo,5 períodos, e intervalo de Tam
t = np.arange(0,Ts*5,Tam)
# criando x(t)
x = 1 + np.sin(w0*t)+2*np.cos(w0*t)+np.cos( (2*w0*t)+ pi/4 )

# calculando a FS
X = fft(x)/len(x)

#criando o vetor de frequencia
w = fftfreq(len(t), d=(1/Ts)*Tam)

# Os indices de frequencia são mudados de 0 a N-1 para (-N/2 + 1)
# a (N/2)
# posicionando a freq. zero no meio do gráfico
Xd = fftshift(X)
wd = fftshift(w)

# calculando o modulo - magnitude do espectro
ModX = np.abs(Xd)
# calculando a fase do espectro
phasX = np.angle(Xd)

# devido a erros de arredondamentos numericos da fft devemos
filtrar os sinais muito pequenos!
phasX[ModX < 0.00001] = 0 # cuidado com isso aqui, isso depende do
previo conhecimento do sinal

# retornando o sinal ao dominio do tempo
xr = ifft(X)*len(x)

xr = np.real(xr) # ignorando os erros de arrendondamento do fft e
ifft

# A partir daqui é plotar os gráficos.
```

Os gráficos:

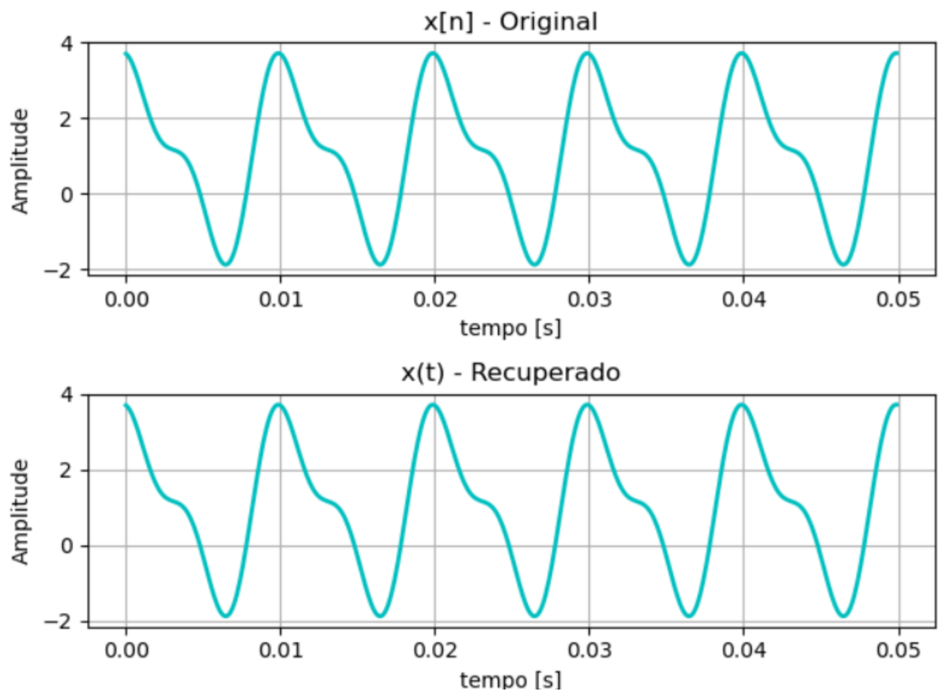


Figura 3 – Sinal original e sinal recuperado

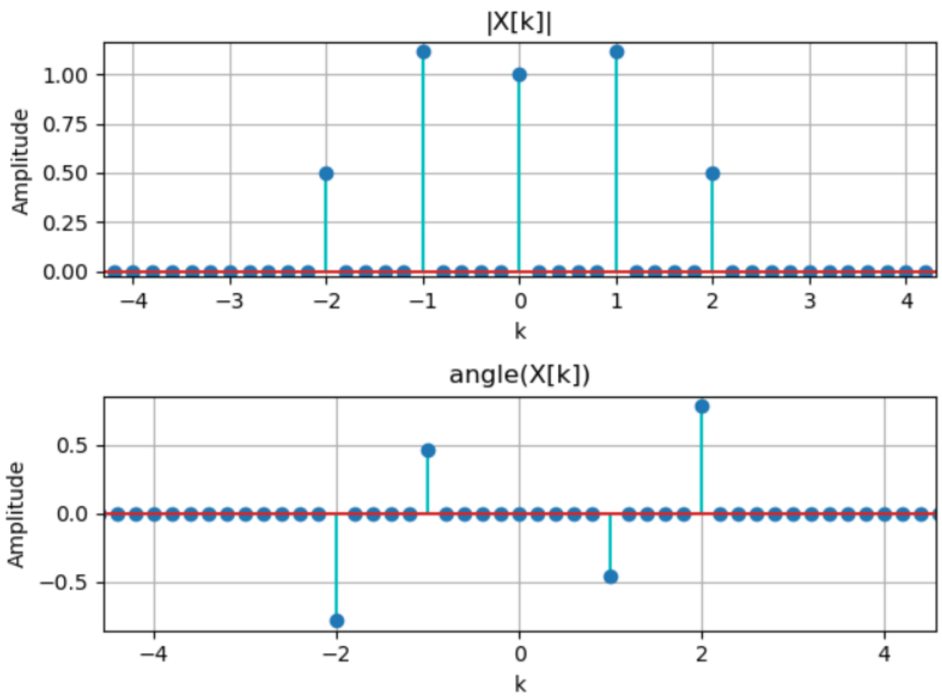
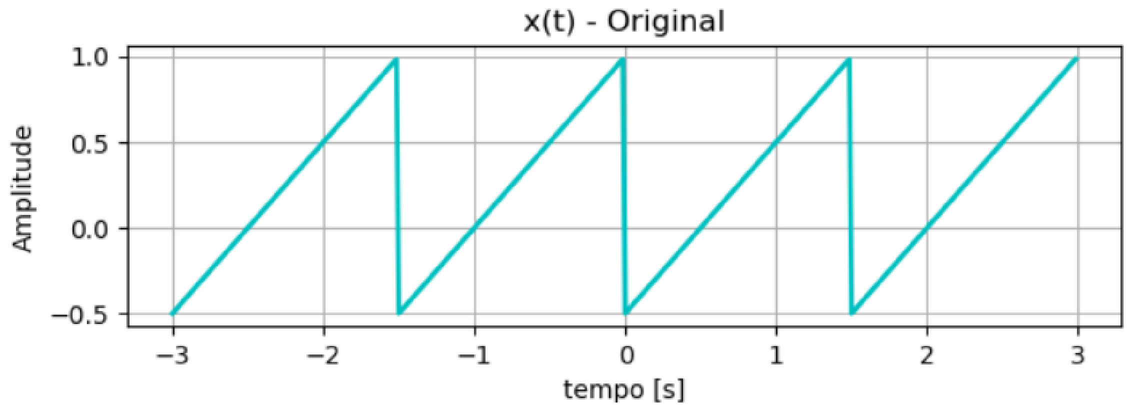


Figura 4 – Gráficos de magnitude e fase dos coeficientes de Fourier do sinal

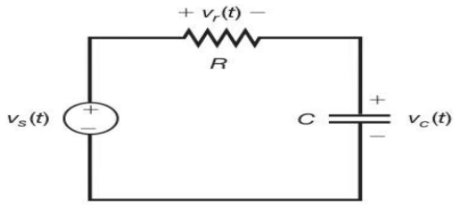
- O código completo em Python é o Exemplo_FS.py

Exercícios:

3. Encontre a representação por FS da onda periódica 'dente-de-serra' (onda triangular) da figura. Esboce os espectros de magnitude e fase:

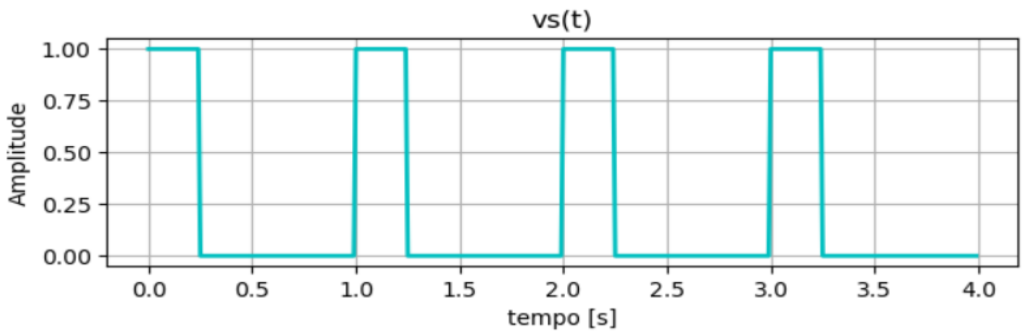


4. O filtro passa-baixas RC



$$H(j\omega) = \frac{1}{1 + j\omega RC}$$

tem como saída a tensão no capacitor $y(t) = v_c(t)$ e possui a resposta em frequência dada por $H(j\omega)$. Sendo a entrada $v_s(t)$ a onda quadrada apresentada na figura.



Apresente os gráficos:

a) Da representação FS da entrada $v_s(t)$:

b) De $y(t)$ no tempo e da sua representação FS com:

i. RC = 0.01 s

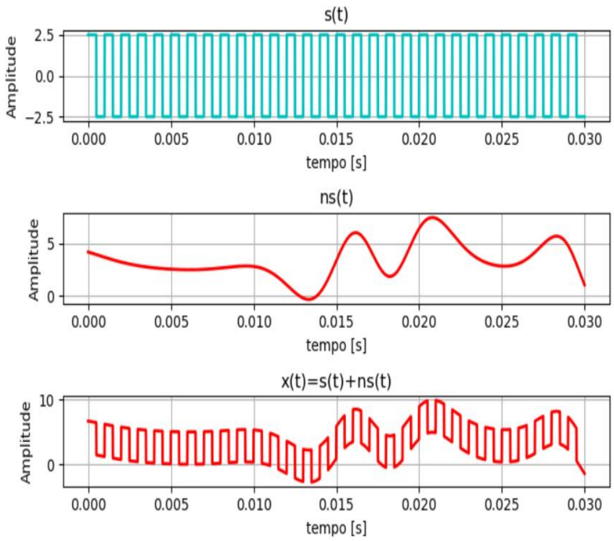
ii. RC = 0.1 s

iii. $RC = 1\ s$

5. Na tentativa de capturar o sinal $s(t)$, apareceu somado ao sinal obtido $x(t)$ um sinal interferente $ns(t)$. Onde o sinal $ns(t)$ é dado pela equação abaixo onde $Vfrq$ pode variar aleatoriamente entre 0 e 100 e Vdc pode variar aleatoriamente entre 0 e 5.

$$s(t) = 2.5square(2000\pi t)$$

$$ns(t) = Vdc + 2.5sen(2Vfrq\pi t)$$



a) Proponha um filtro IDEAL para retirar o sinal indesejado $n_s(t)$.

b) Use o programa em Python em anexo para gerar o sinal x(t).

   i. Implemente em Python o filtro proposto em (A) e aplique no sinal.

   ii. Plote a saída $y(t)$ no domínio do tempo e na frequência.

   iii. Plote a magnitude em frequência:

      1. Do filtro;

      2. De $x(t)$;

      3. Da saída do filtro $y(t)$;