

Projeto A

Entrega por mail (ao professor da PL) esta semana

O nome do projeto deve incluir:

LN2010_A_PL2x_Gn.py

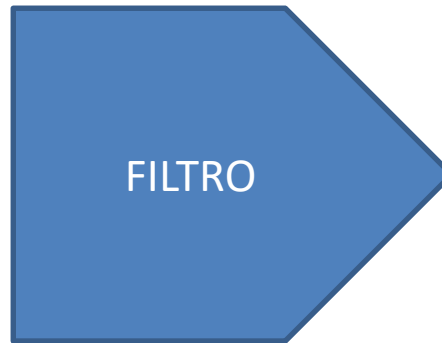
LN2010_A_PL2x_Gn.pptx

Se o ppt for grande demais mandem por wetransfer

Aula 16

Processamento de series temporais por Filtros digitais

INPUT
 $x_k, k = 0, \dots, N - 1$
Sinal+ Ruído



OUTPUT
 $y_k, k = 0, \dots, N - 1$
Sinal+...

Sinal e ruído

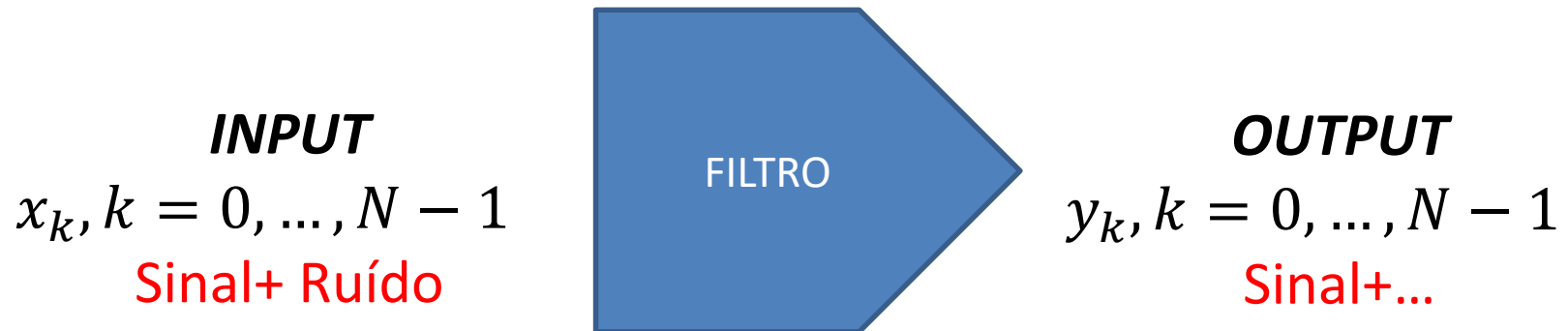
As séries de dados reais contêm em geral informação proveniente de diferentes **processos**.

Dependendo do objetivo do analista só parte dessa informação é relevante, constituindo o **sinal**, sendo as outras componentes designadas por **ruído**.

Separar o sinal do ruído é o objetivo dos **filtros**.

No caso dos **filtros digitais** a separação é feita de forma algébrica, tipicamente por operações lineares sobre os dados.

Filtro



Filtro linear **não recursivo** (o output depende linearmente do input)

$$y_k = \sum_{n=0}^N x_n h_{k-n} \Rightarrow y = x * h$$

convolução

Coeficientes do filtro $h_j, j = 0, \dots, J - 1$

Filtro recursivo

O output num dado instante depende do input, e também do próprio output em instantes anteriores:

$$y_k = \sum_{n=0}^N x_n h_{k-n} + \sum_{m=0}^{k-1} y_m b_{k-m}$$

Média móvel

$$y_k = \sum_{n=0}^N x_n h_{k-n} \implies y = x * h$$

Se:

$$\sum_{n=0}^J h_j = 1$$

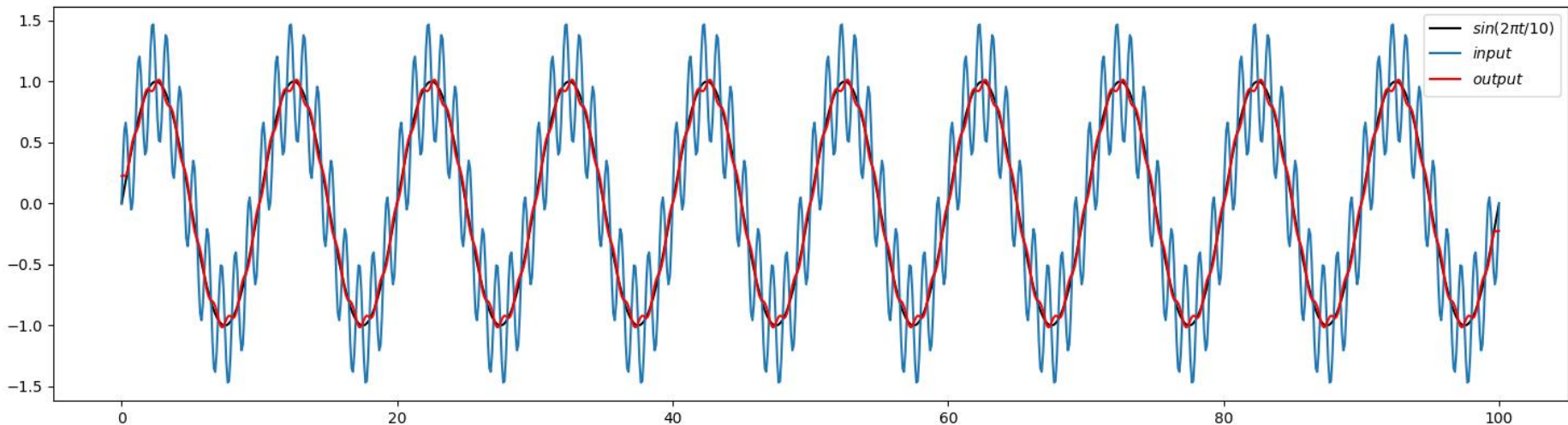
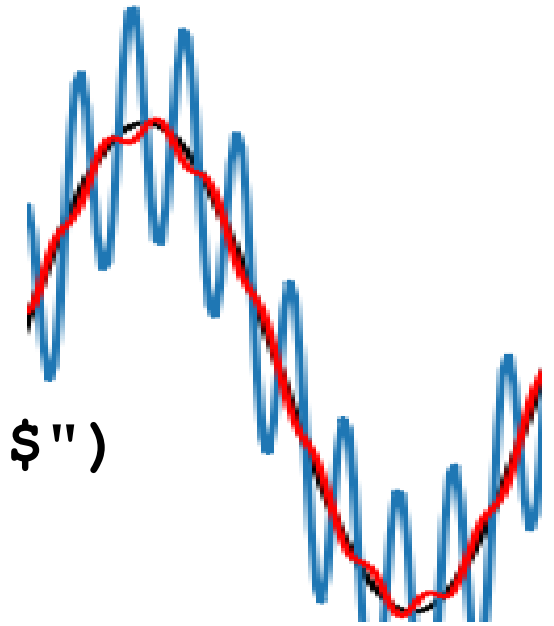
Trata-se de um filtro de média móvel. Exemplo:

$$h_j = \frac{1}{J}$$

Média móvel

```
import numpy as np; import matplotlib.pyplot as plt
N=1001;dt=0.1;T=10.
t=np.arange(0., (N-1)*dt+dt, dt)
x1=np.sin(2*np.pi*t/T) # "sinal"
x2=0.5*np.sin(2*np.pi*t/(T/10)) # "ruído"
x=x1+x2 # série a filtrar input
plt.plot(t,x1,color='black',\
         label=r"$\sin(2\pi t/10)$")
plt.plot(t,x,label=r"$input$")
nn=11 # comprimento do filtro
h=np.ones(nn)/nn
y=np.convolve(x,h,mode='same') # output
plt.plot(t,y,color='red',label=r"$output$")
plt.legend()
```

```
plt.plot(t,x1,color='black',\
         label=r"$\sin(2\pi t/10)$")
plt.plot(t,x,label=r"$input$")
plt.plot(t,y,color='red',\
         label=r"$output$")
```



A media móvel aplicada...

Diminuiu a amplitude das altas frequências.

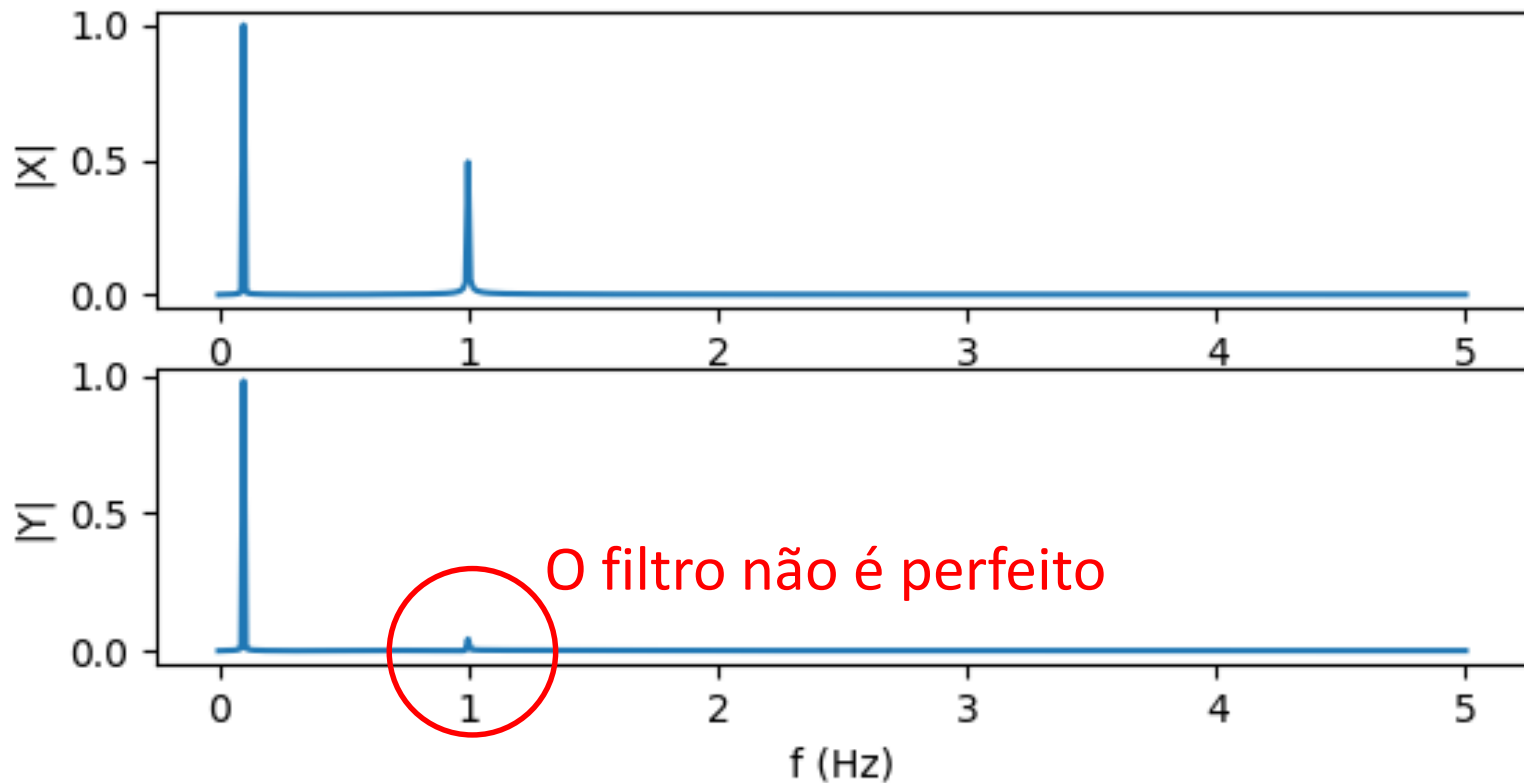
Constitui um filtro **passa-baixo**: deixa passar as baixas frequências.

```
x1=np.sin(2*np.pi*t/T) # "sinal" (100
pontos/T)
x2=0.5*np.sin(2*np.pi*t/(T/10)) # "ruído" (10
pontos/T)
x=x1+x2 # série a filtrar input
nn=11 # comprimento do filtro
h=np.ones((nn))/nn
y=np.convolve(x,h,mode='same') # output
```

Com a mesma série e filtro

```
plt.figure()
X=np.fft.fft(x) #Espectro do input
Y=np.fft.fft(y) #Espectro do output
fNyq=1/(2*dt)
df=2*fNyq/(N-1)
freq=np.arange(0,fNyq+df,df)
plt.subplot(3,1,1)
plt.plot(freq,np.abs(X[0:N//2+1])/(N//2),label='X')
plt.ylabel('|X|') #espectro de amplitude de x
plt.subplot(3,1,2)
plt.plot(freq,np.abs(Y[0:N//2+1])/(N//2),label='Y')
plt.ylabel('|Y|') #espectro de amplitude de y
plt.xlabel('f (Hz)')
```

```
x1=np.sin(2*np.pi*t/T) # "sinal"  
x2=0.5*np.sin(2*np.pi*t/(T/10)) # "ruído"  
x=x1+x2  
y=np.convolve(x,h,mode='same')
```

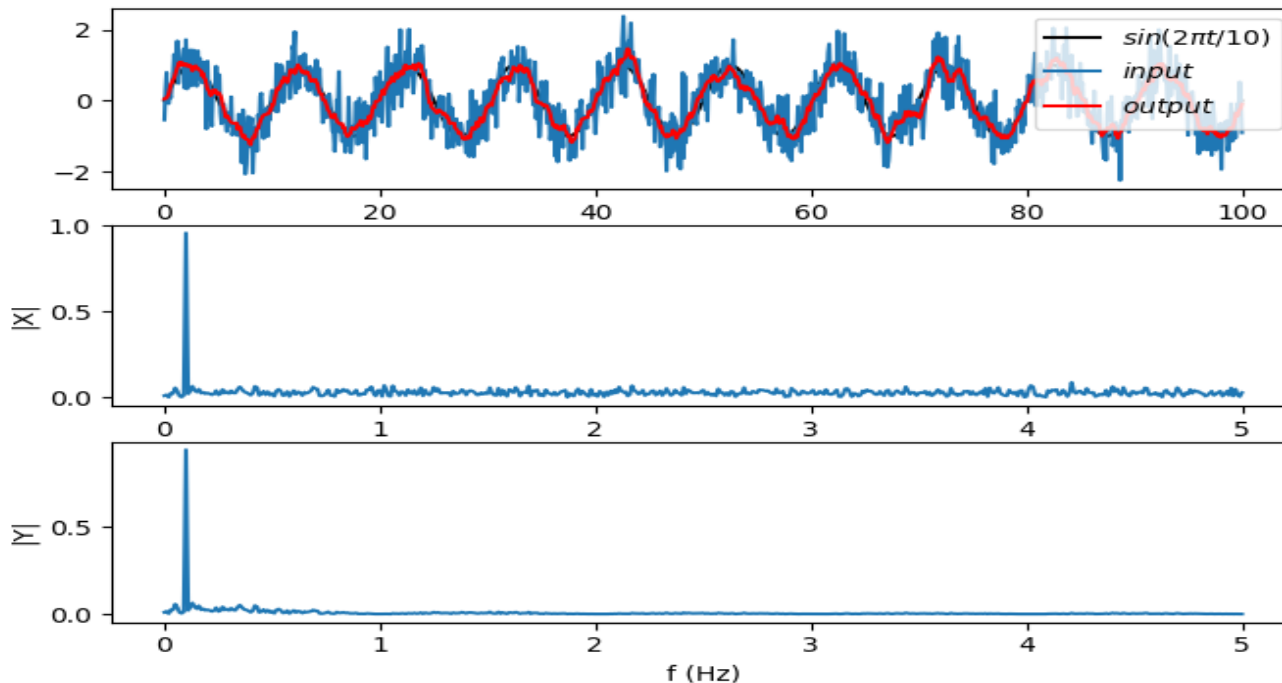
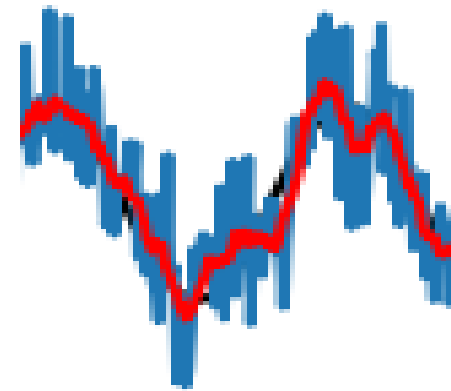


É claro que o ruído não costuma ser um seno de alta frequência...

```
x1=np.sin(2*np.pi*t/T)
```

```
r=0.5*np.random.randn(t.shape[0])
```

```
x=x1+r
```



Efeitos de fronteira

`y=np.convolve(x,h,mode='same')`
'same' prolonga a série x
Produzindo y com os mesmos pontos

Série com n pontos, filtro com J pontos

$$\{x_k\} = x_0, x_1, \dots, x_{N-1}$$
$$\{h_j\} = h_0, h_1, \dots, h_{J-1}$$

Filtragem (convolução):

$$y_k = \sum_{n=0}^N x_n h_{k-n}, (k-n) \in [0, J-1]$$

Exemplo $J = 10$:

$$y_9 = x_9 h_0 + x_8 h_1 + \dots + x_0 h_9$$

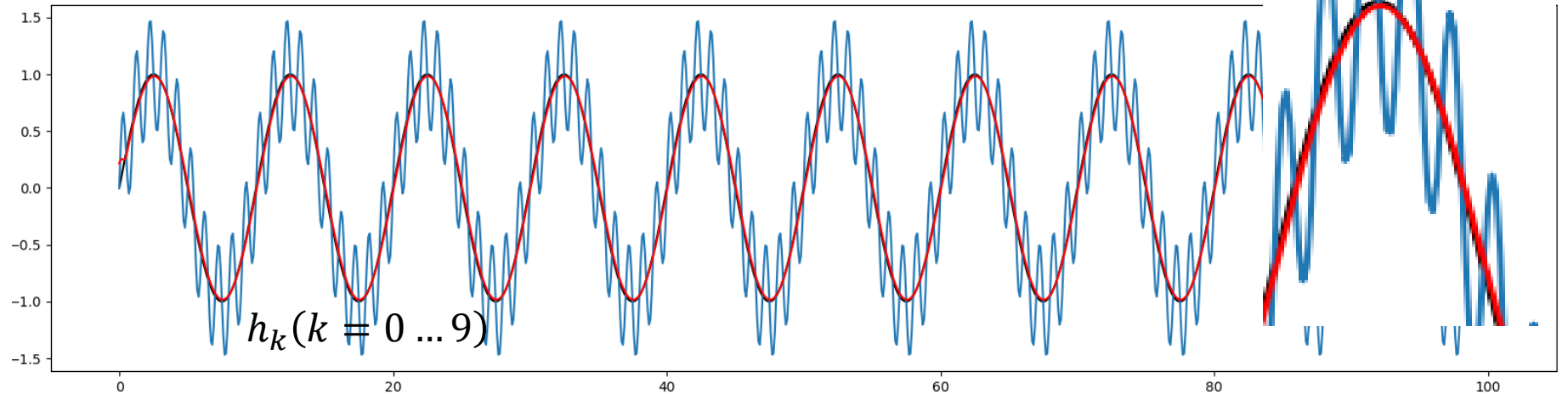
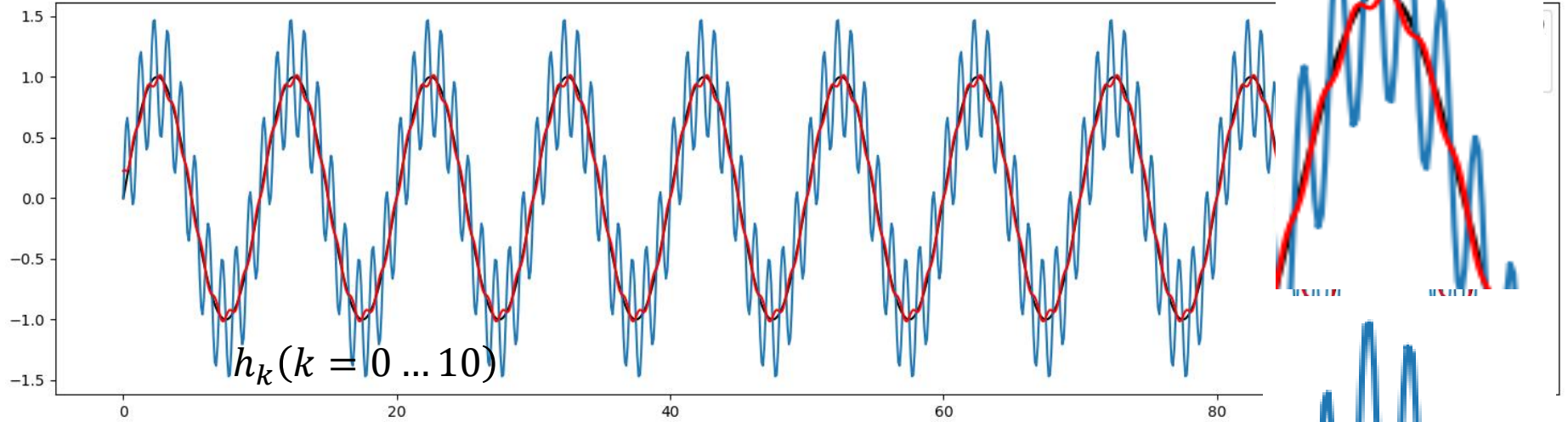
$$y_8 = x_8 h_0 + x_7 h_1 + \dots + x_1 h_7 + x_0 h_8 + x_{-1} h_9$$

Não existe



O que faz a media móvel ao espectro?

10 pontos/T



Teorema da convolução

A **transformada de Fourier da convolução** de duas séries é igual ao **produto das transformadas** das séries individuais:

$$y = x * h$$
$$Y = \mathcal{F}(y) = \mathcal{F}(x * h) = \mathcal{F}(x)\mathcal{F}(h) = XH$$

Notar que Y, X, H são **complexos**.

H é designada por **função de transferência** do filtro h

Série sintética com 4 frequências

```
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')
N=1001
dt=1.0
T=365.
t=np.arange(0., (N-1)*dt+dt, dt)
x=np.sin(2*np.pi*t/T)
plt.subplot(4,1,1)
plt.plot(t,x,label=r"$\sin(2\pi t/365)$",color='green')
for Tr in [110.,75.,55.,18.]:
    x=x+np.sin(2*np.pi*t/Tr)
```



```
plt.plot(t,x,label=r"$input$",color='black')
nn=150 #comprimento do filtro
h=np.ones(nn)/nn #média móvel
y=np.convolve(x,h,mode='same')
plt.plot(t,y,color='red',label=r"$output$")
plt.legend()
X=np.fft.fft(x)
Y=np.fft.fft(y)
```

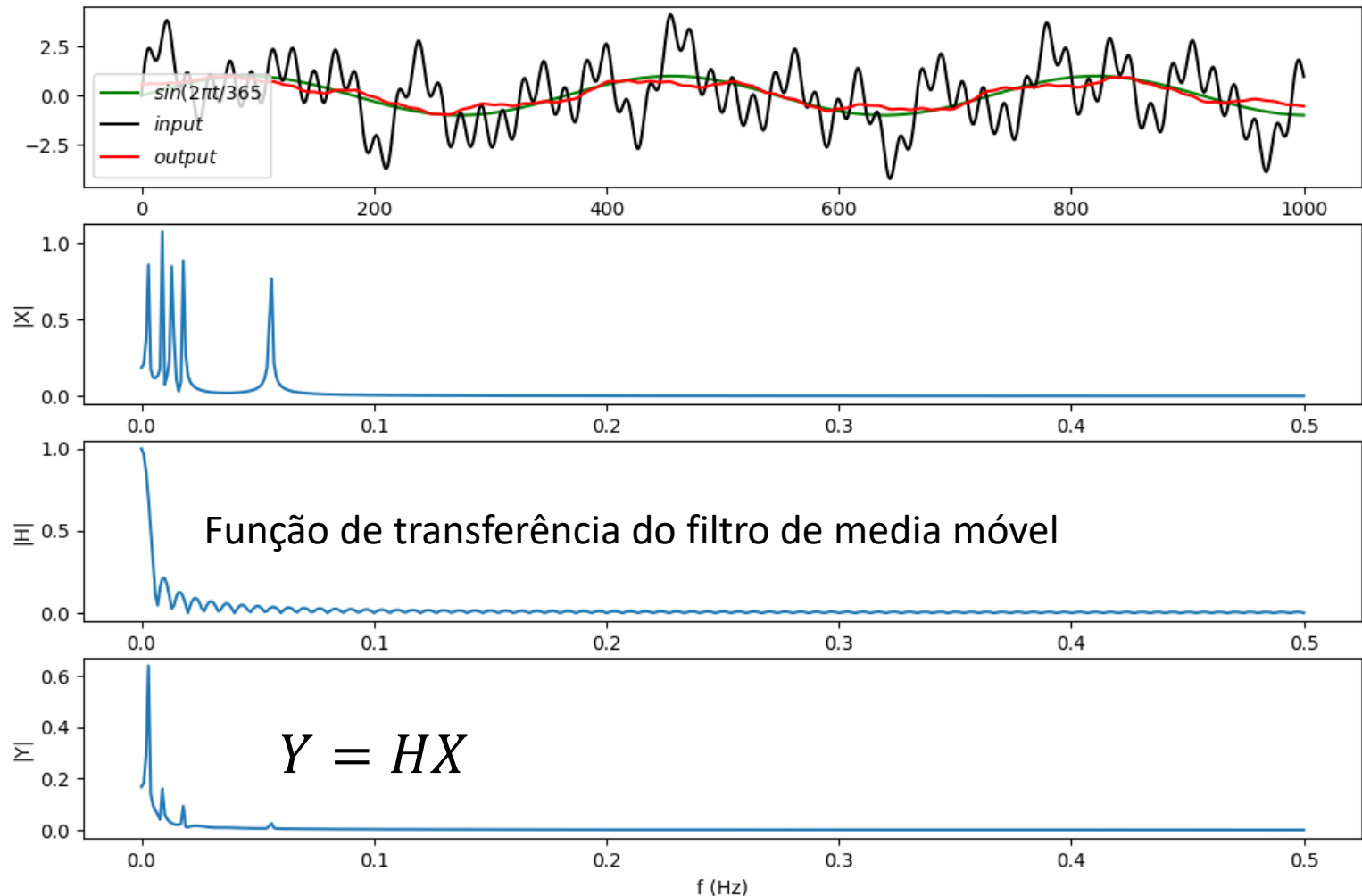
```

fNyq=1/(2*dt)
if N%2==0:
    df=2*fNyq/N
else:
    df=2*fNyq/(N-1)
freq=np.arange(0, fNyq+df, df)
plt.subplot(4,1,2)
plt.plot(freq,np.abs(X[0:N//2+1])/(N//2),label='X')
plt.ylabel('|X|') #espectro de amplitude de x
hExt=np.zeros(x.shape)
hExt[0:nn]=h #os outros termos são nulos
H=np.fft.fft(hExt)
plt.subplot(4,1,3)
plt.plot(freq,np.abs(H[0:N//2+1]),label='H')
plt.ylabel('|H|') #espectro de amplitude de h
plt.subplot(4,1,4)
plt.plot(freq,np.abs(Y[0:N//2+1])/(N//2),label='Y')
plt.ylabel('|Y|') #espectro de amplitude de y
plt.xlabel('f (Hz)')

```

Teorema da convolução

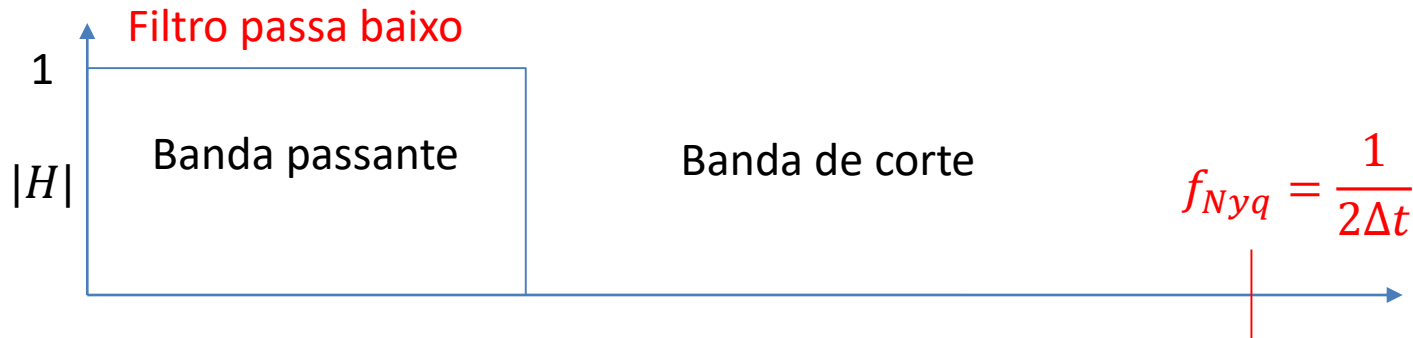
```
x=np.sin(2*np.pi*t/365)  
for Tr in [110.,75.,55.,18.]:  
    x=x+np.sin(2*np.pi*t/Tr)
```



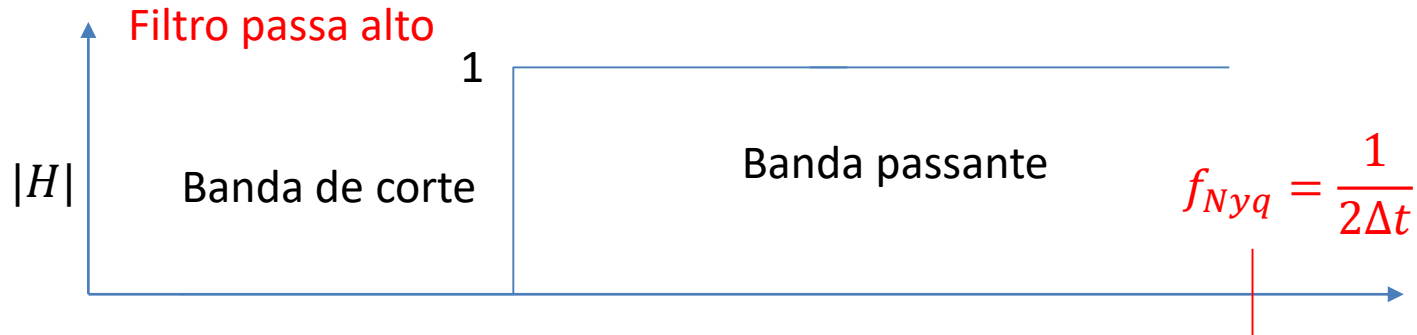
O que seria um filtro perfeito? (amplitude)

$$Y_L = H_{Low}X$$

$$Y = HX$$



$$Y_H = H_{High}X$$



$$Y_H = (1 - H_{Low})X$$

Filtros de Fourier

Se desenharmos diretamente $H(\omega)$, podemos fazer:

$$X = \mathcal{F}(x)$$

$$Y = HX$$

$$y = \mathcal{F}^{-1}(Y)$$

Como a **fft** é muito eficiente, o método permite implementar filtros quase-ideais.

Mas atenção: H (tal com X) é **complexo** e tem que ser definido em todo o domínio $[-f_{Nyq}, f_{Nyq}]$ com as simetrias adequadas.

Ver exemplo nos apontamentos!