Modelação Numérica 2022
Aula 4

Gestão de falhas em dados, espectrogramas

Atenção

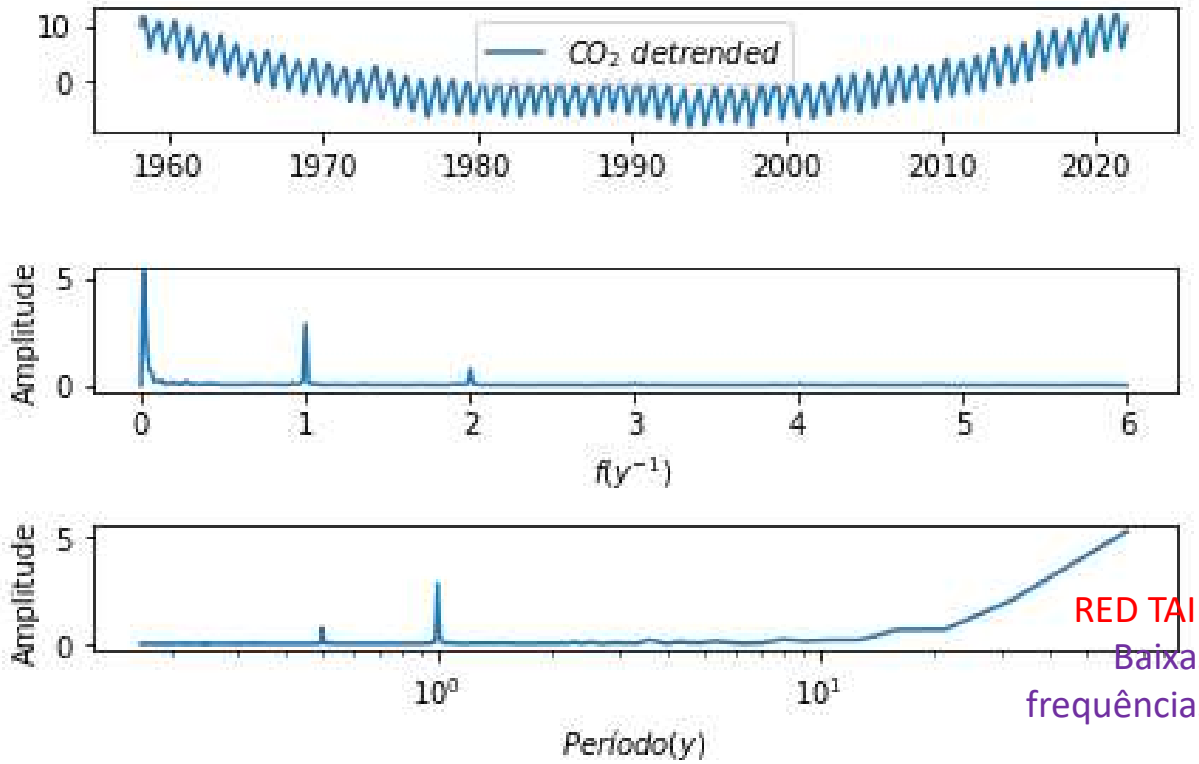
Tal como indicado no calendário: amanhã não há aula teórica.

Da aula anterior

A subtração da média impõe $F[0] = 0$.

A subtração da linha de tendência impõe $F[0] = 0$ e modifica as muito baixas frequências mas não as elimina.

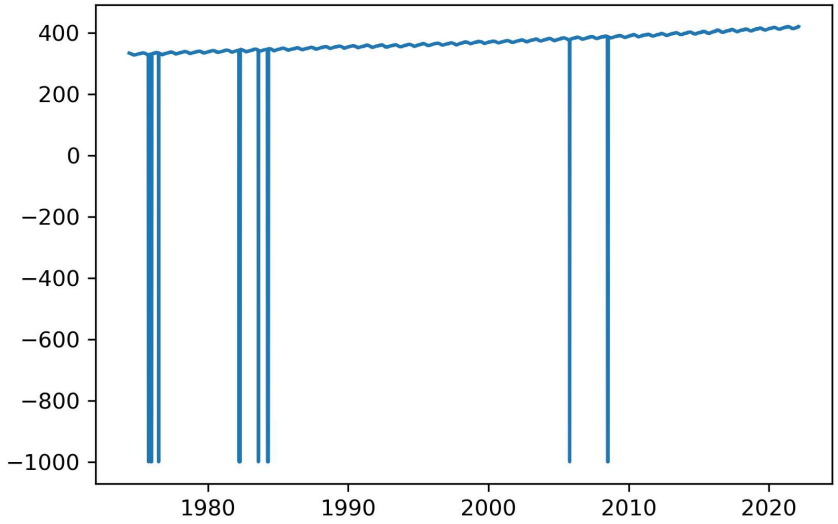
Precisamos de utilizar um filtro passa-alto.



RED TAIL
Baixas frequências

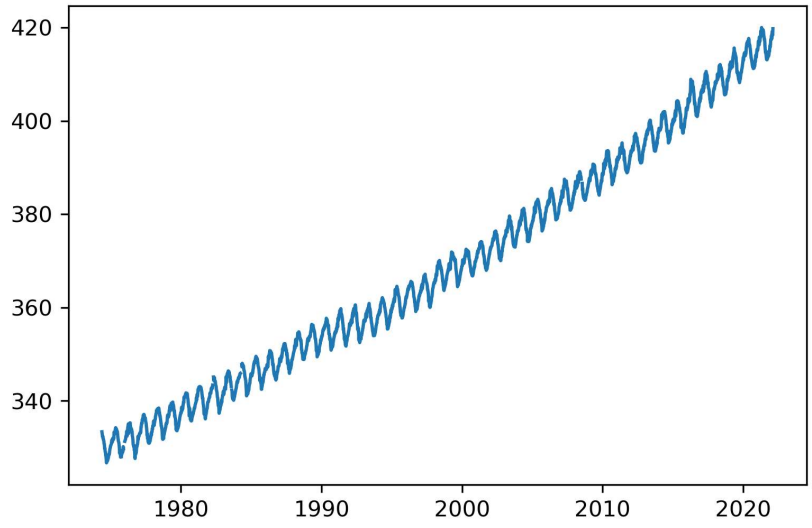
Antes disso, dados semanais (falha=-999)

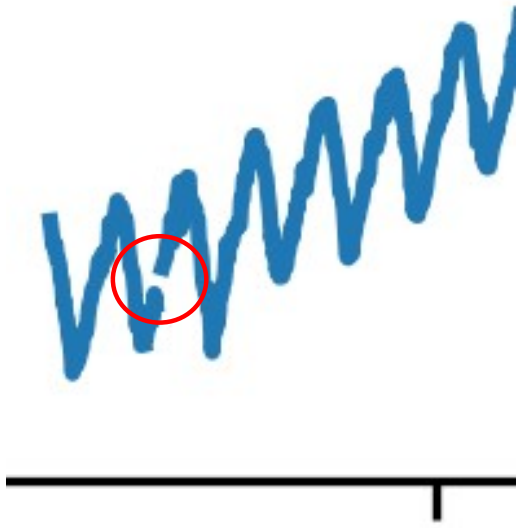
```
ML=np.loadtxt('MaunaLoa_Week_NOAA.txt')
year=ML[:,3]
co2=ML[:,4]
plt.plot(year,co2)
```



nan nas falhas (só para plt)

```
ML=np.loadtxt('MaunaLoa_Week_NOAA.txt')
year=ML[:,3]
co2=ML[:,4]
N=len(co2)
for k in range(N):
    if co2[k]<0:
        co2[k]=np.float('nan')
plt.plot(year,co2)
```





A utilização de nan “só” funciona nos gráficos… Preencher falhas

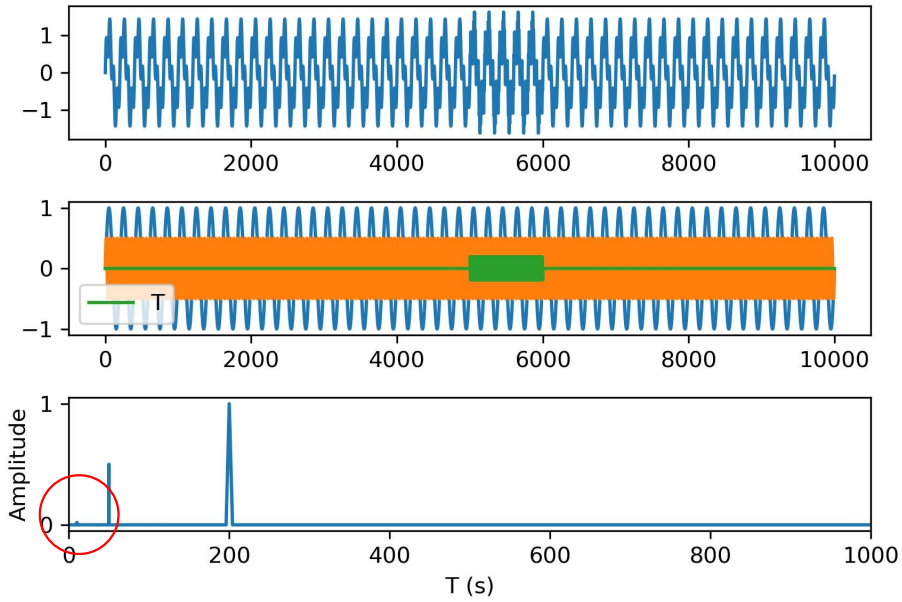
```
ML=np.loadtxt('MaunaLoa_Week_NOAA.txt')
dt=7/365.25
year=ML[:,3];co2=ML[:,4];N=len(co2)
co2V=[];yearV=[]
for k in range(N):
    if co2[k]>0:
        co2V.append(co2[k])
        yearV.append(year[k])
co2V=np.array(co2V) #dados válidos
yearV=np.array(yearV)
co2I=np.interp(year,yearV,co2V)
```

Sinais transientes e espectrogramas

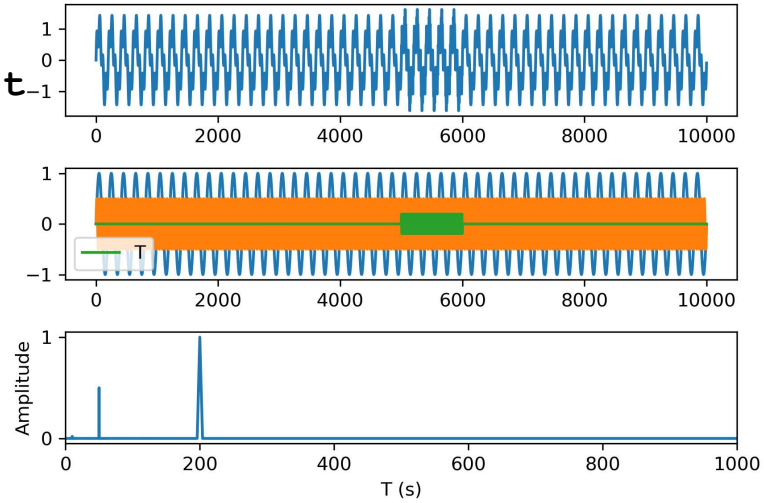
Sinais de curta duração podem ser impossíveis de detetar quando se analisa espectro.

No exemplo sintético estão sobrepostas 3 sinusoides, uma delas só durante uma janela temporal. No espectro de amplitude o sinal transiente (T=10s) é muito fraco e estaria ausente se existe um pequeno nível de ruído.

Se analisarmos por janelas podemos dar relevância a estes sinais transientes.

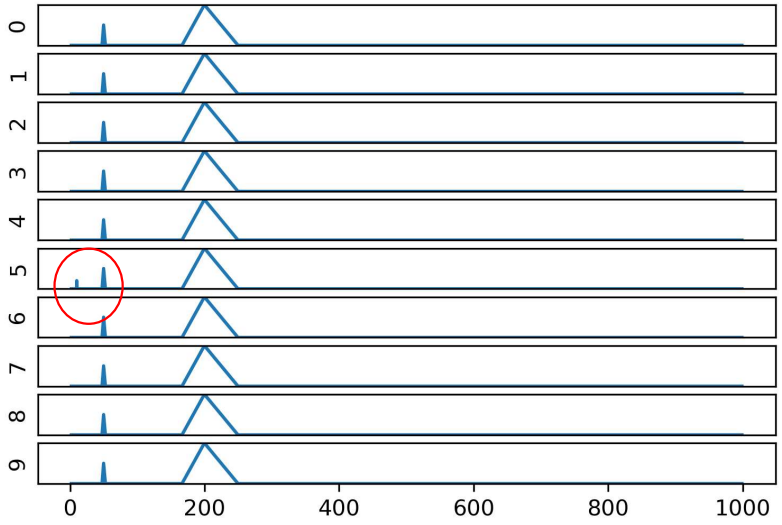


```
import numpy as np;import matplotlib.pyplot as plt
from scipy import fft
N=10000;dt=1; t=np.linspace(0,dt*(N-1),N)
T=[200,50];A=[1,0.5]
TT=10;AT=0.2;kT0=5000;kT1=6000;
fT=AT*np.sin(2*np.pi*t/TT);fT[0:kT0]=0;fT[kT1:]=0
f1=A[0]*np.sin(2*np.pi*t/T[0]);
f2=A[1]*np.sin(2*np.pi*t/T[1])
f=f1+f2+fT
fig,ax=plt.subplots(nrows=3)
ax[0].plot(t,f)
fNyq=1/(2*dt);df=2*fNyq/(N-1);freq=np.arange(0,fNyq+df,df)
ax[1].plot(t,f1);ax[1].plot(t,f2);ax[1].plot(t,fT,label=r'T')
ax[1].legend()
F=fft.fft(f)
ax[2].plot(1/freq,np.abs(F[0:N//2+1])/(N//2))
ax[2].set_xlim(0,1000)
ax[2].set_ylabel('Amplitude');plt.xlabel('T (s)')
```

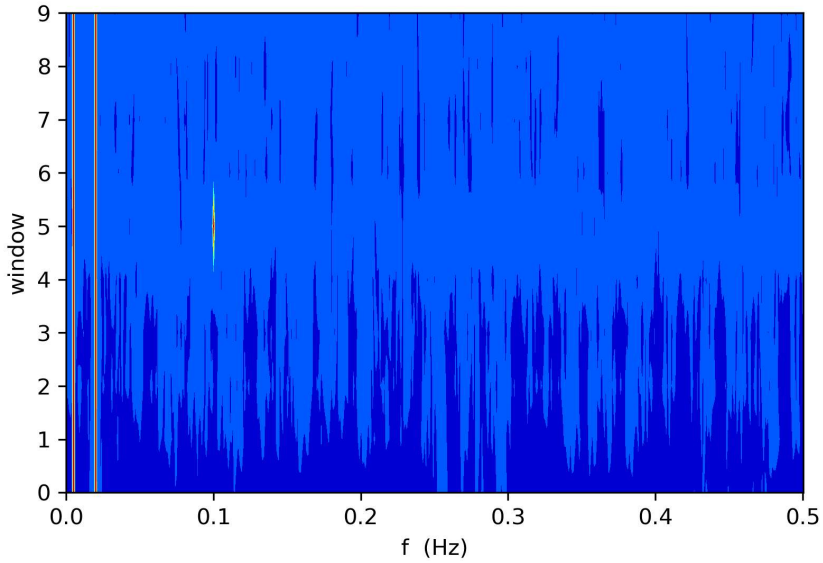


Espectrograma

```
fig,ax=plt.subplots(nrows=10)
window=1000
kw=-1
for k in range(0,N,window):
    kw+=1
    fw=f[k:k+window]
    Nw=len(fw)
    fNyq=1/(2*dt); df=2*fNyq/(Nw-1)
    freq=np.arange(0,fNyq+df,df)
    Fw=fft.fft(fw)
    ax[kw].plot(1/freq,np.abs(Fw[0:Nw//2+1])/(Nw//2))
    ax[kw].set_ylim(0,1)
    ax[kw].set_yticks([])
    if kw<9:
        ax[kw].set_xticks([])
    ax[kw].set_ylabel(str(k//1000))
```



```
#versão 2
window=1000
FF=np.zeros((window//2+1,N//window))
xis=np.copy(FF);yps=np.copy(FF)
kw=-1
for k in range(0,N,window):
    kw+=1
    fw=f[k:k+window];Nw=len(fw)
    fNyq=1/(2*dt); df=2*fNyq/(Nw-1)
    freq=np.arange(0,fNyq+df,df)
    xis[:,kw]=freq;yps[:,kw]=kw
    Fw=fft.fft(fw)
    FF[:,kw]=np.abs(Fw[0:Nw//2+1])/(Nw//2)
plt.figure()
plt.contourf(xis,yps,np.log(FF),cmap='jet')
plt.xlabel('f  (Hz)')
plt.ylabel('window')
```

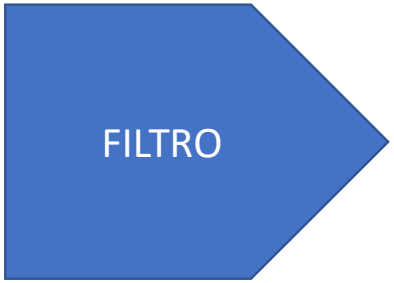


Filtro digital

INPUT

$x_k, k = 0, \ldots, N-1$

Sinal+ Ruído



OUTPUT

$y_k, k = 0, \ldots, N-1$

Sinal+…

Sinal e ruído

As séries da dados reais contém em geral informação proveniente de diferentes processos.

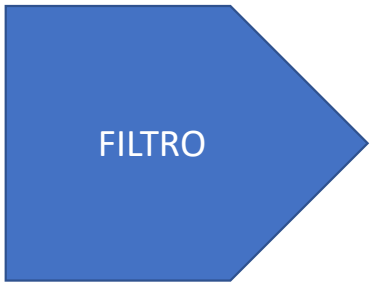
Dependendo do objetivo do analista só parte dessa informação é relevante, constituindo o sinal, sendo as outras componentes designadas por ruído.

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

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



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

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

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

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:

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

Depende de dois conjuntos de coeficientes (h, b)

Média móvel

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

Se:

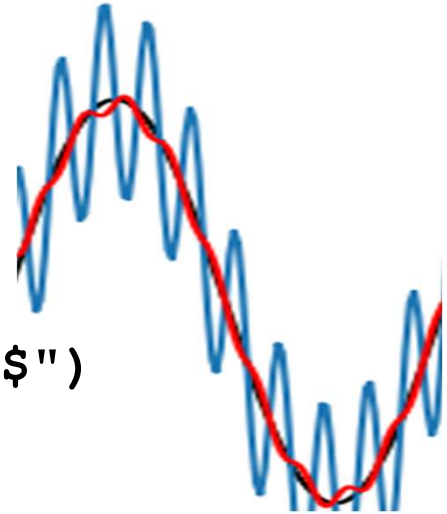
$$\sum_{n=0}^{J} h_j = 1$$

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

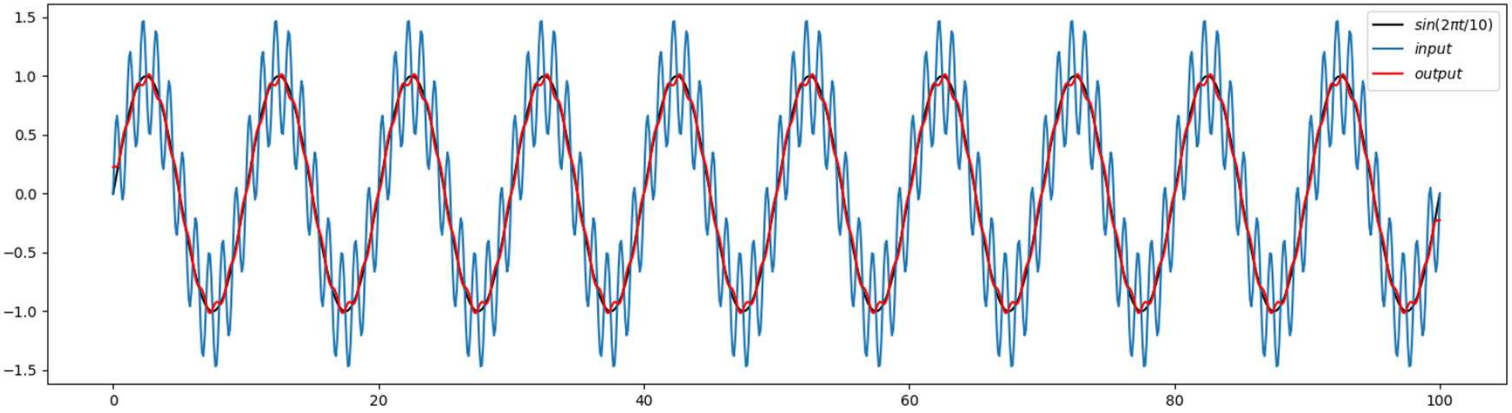
$$h_j = \frac{1}{J}$$

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.

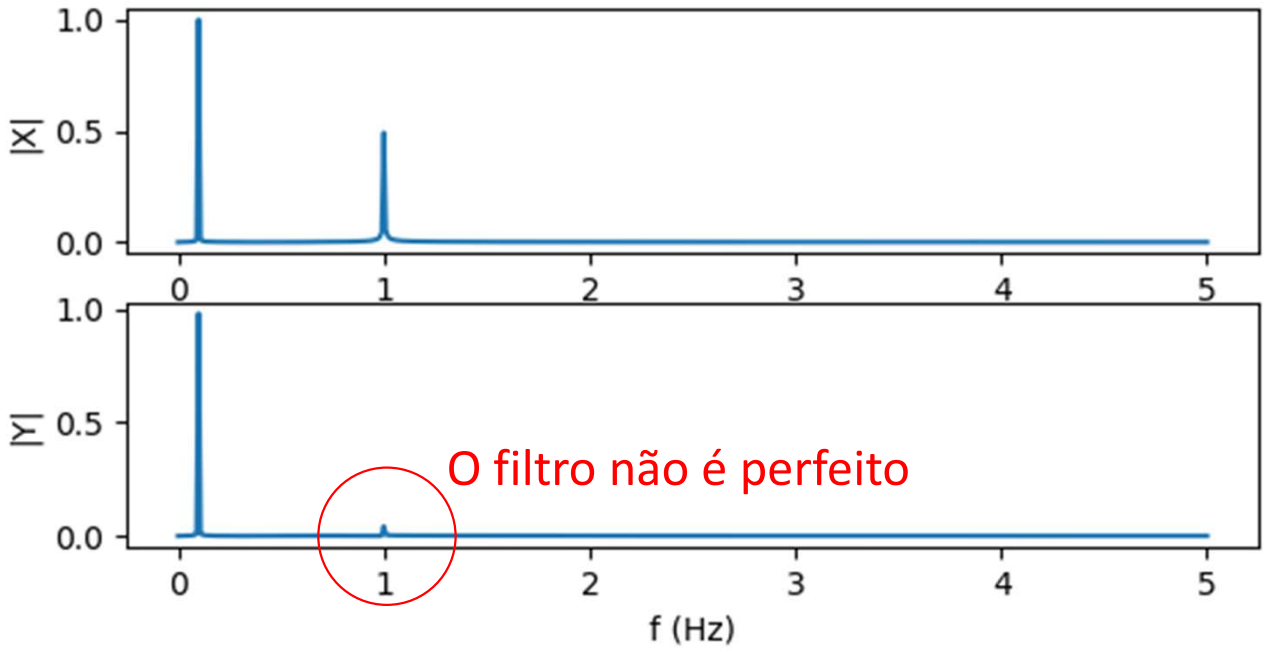
Constitui um filtro **passa-baixo**: deixa passar as baixas frequências.

```
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

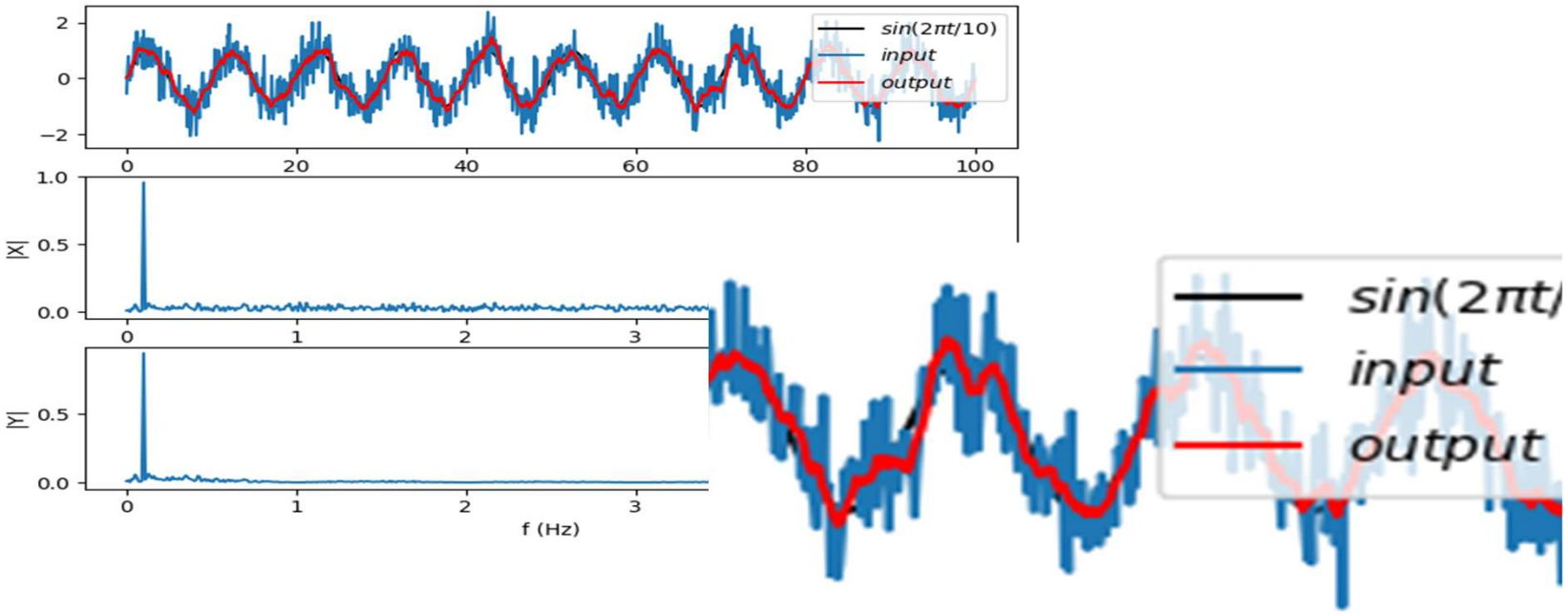
```
plt.figure()
X=np.fft.fft(x) #Espectro do input
Y=np.fft.fft(y) #Espectro do output
fNyq=1/(2*dt)
df=2*fNyq/(N-1)
freq=np.arange(0,fNyq+df,df)
plt.subplot(3,1,1)
plt.plot(freq,np.abs(X[0:N//2+1])/(N//2),label='X')
plt.ylabel('|X|') #espectro de amplitude de x
plt.subplot(3,1,2)
plt.plot(freq,np.abs(Y[0:N//2+1])/(N//2),label='Y')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



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):

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

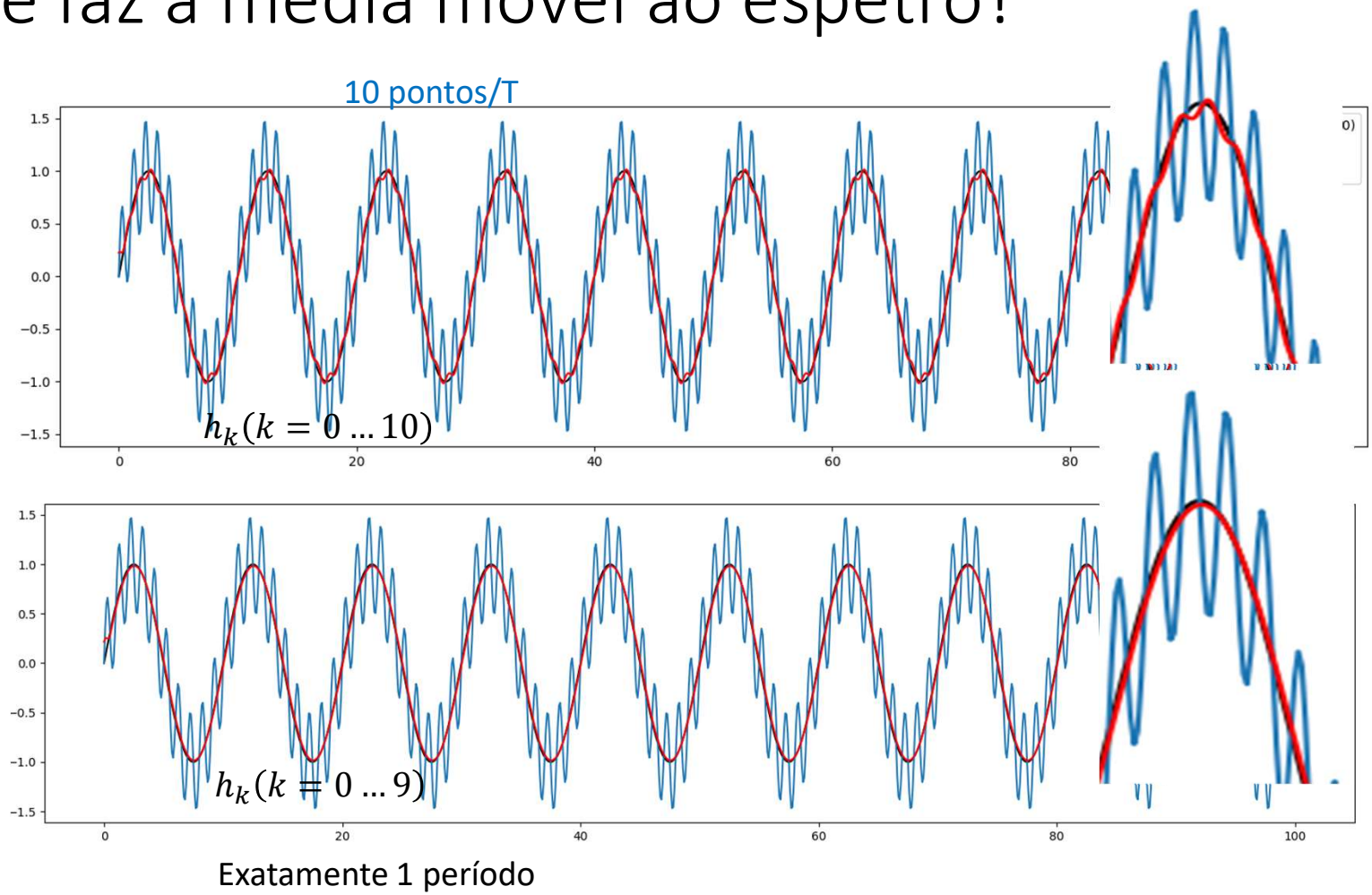
Exemplo $J = 10$:

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

Não existe

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

O que faz a media móvel ao espetro?



Exatamente 1 período

Teorema da convolução

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

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

Notar que Y, X, H são complexos.

H é designada por função de transferência do filtro h

Série sintética com 4 frequências

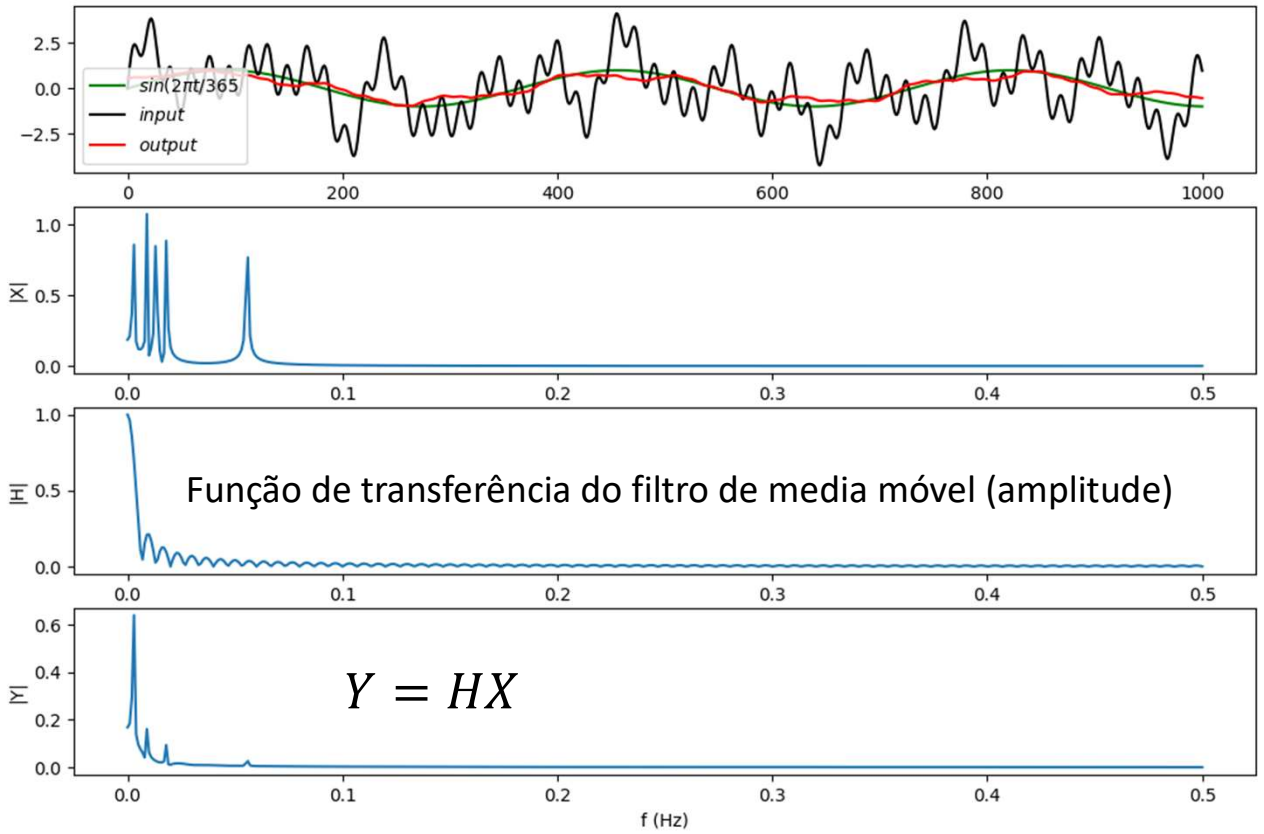
```
import numpy as np
import matplotlib.pyplot as plt
N=1001
dt=1.0
T=365.
t=np.arange(0.,(N-1)*dt+dt,dt)
x=np.sin(2*np.pi*t/T)
plt.subplot(4,1,1)
plt.plot(t,x,label=r"$sin(2\pi t/365$",color='green')
for Tr in [110.,75.,55.,18.]:
    x=x+np.sin(2*np.pi*t/Tr)
```

```
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

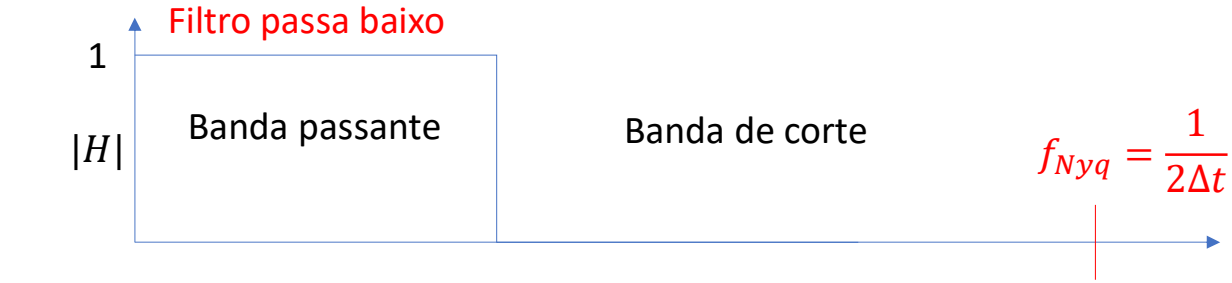
```
x=np.sin(2*np.pi*t/365)
for Tr in [110.,75.,55.,18.]:
    x=x+np.sin(2*np.pi*t/Tr)
```



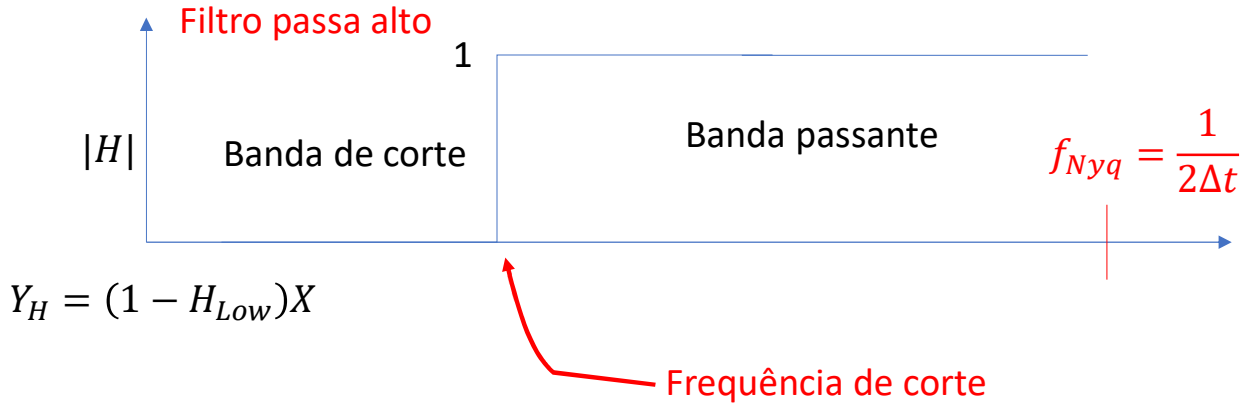
O que seria um filtro perfeito? (amplitude)

$Y_L = H_{Low}X$

$Y = HX$



$Y_H = H_{High}X$



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