Modelação Numérica 2017

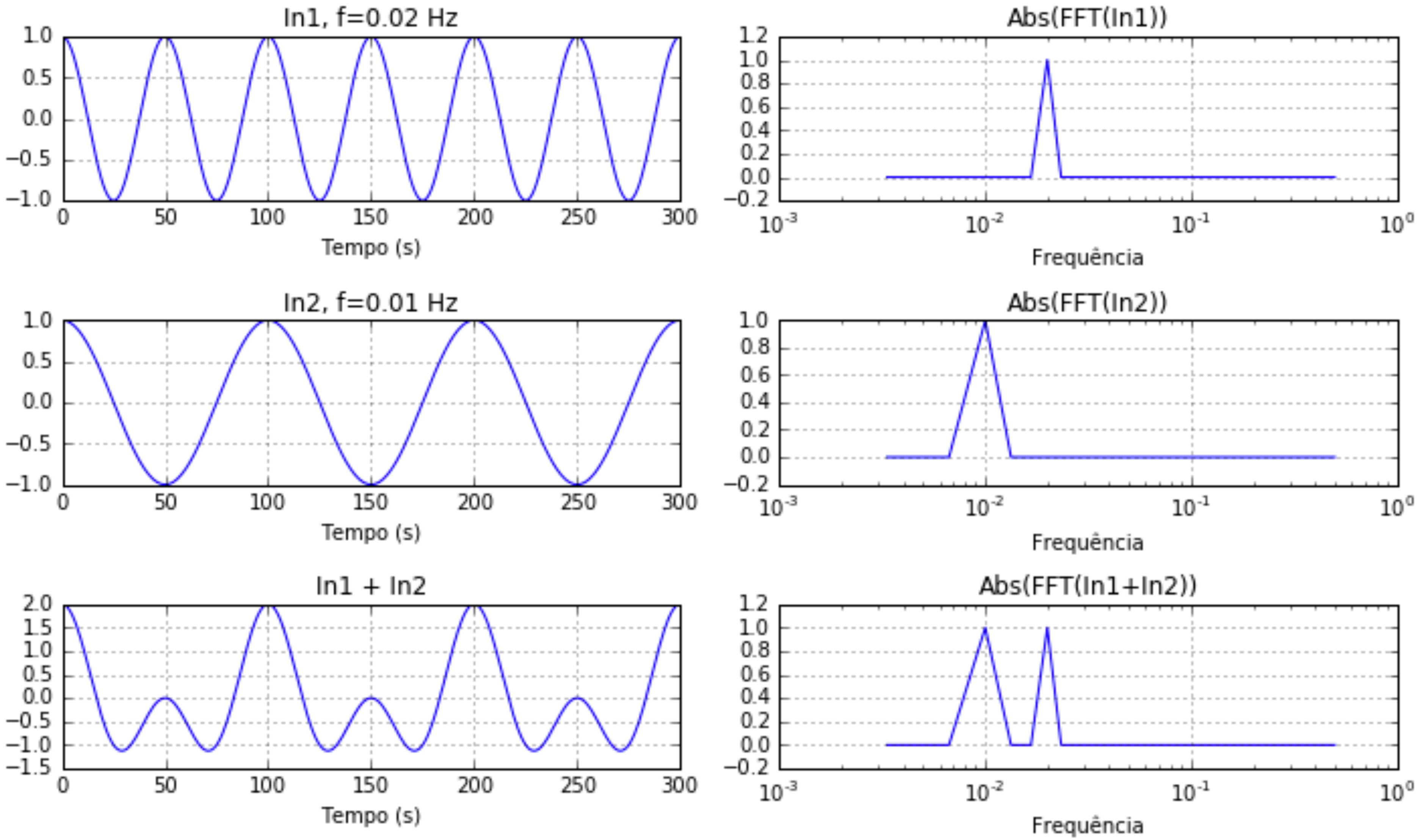
Aula 8, 14/Mar

- Filtros (revisão)
- Espectrogramas

http://modnum.ucs.ciencias.ulisboa.pt

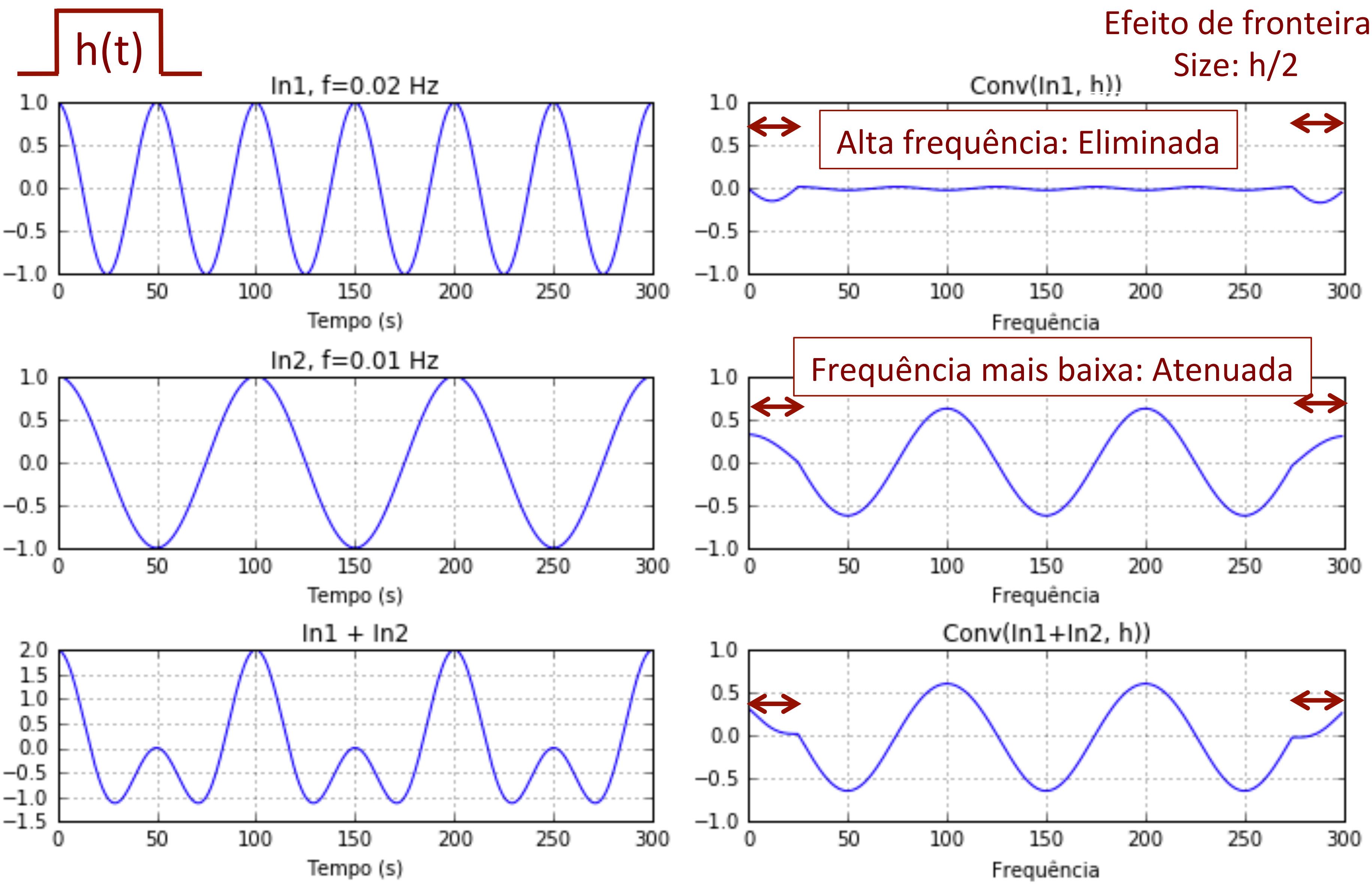
Aula passada

Filtros – Input



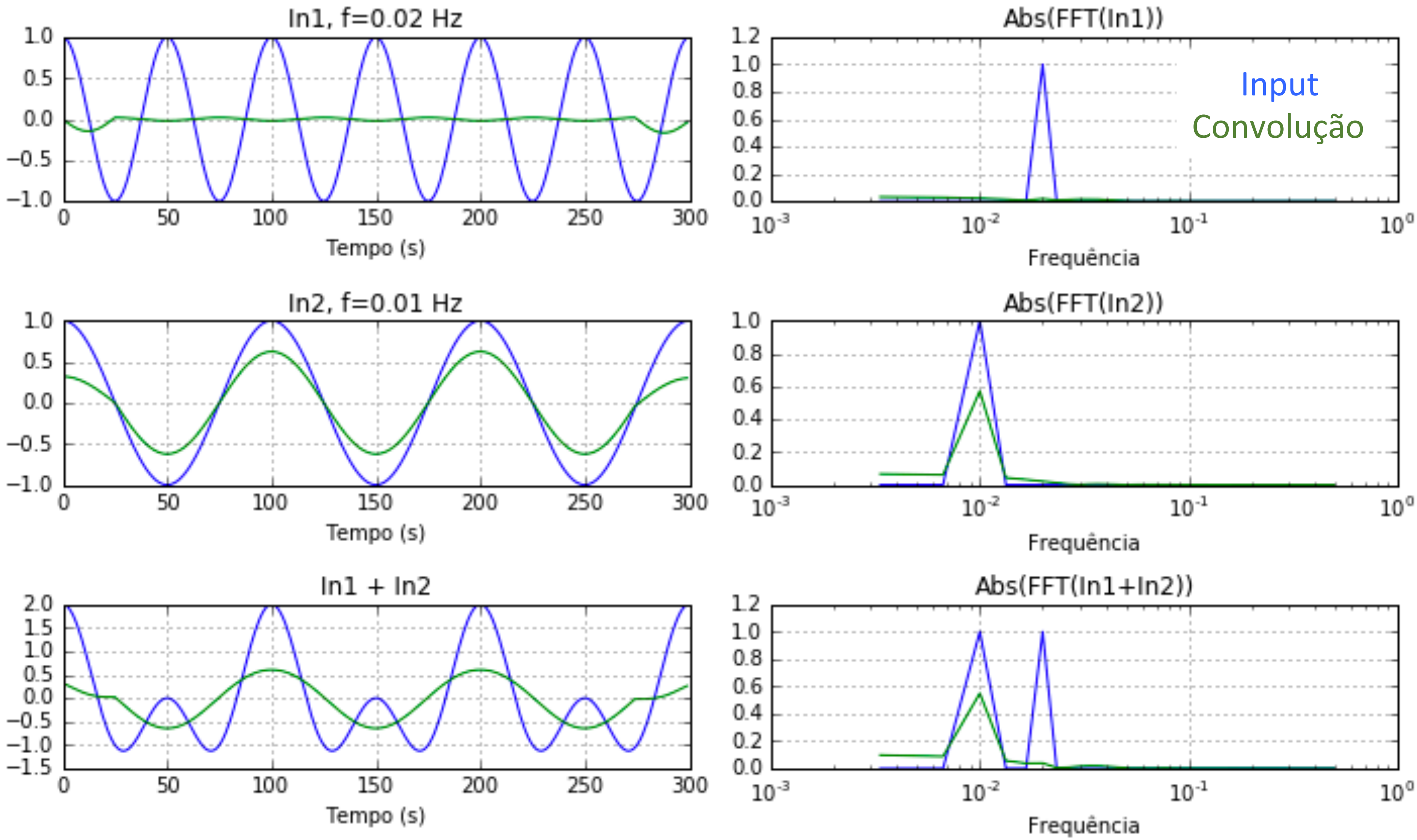
Aula passada

Filtro de média móvel (domínio do tempo)



Aula passada

Filtro de média móvel (domínio do tempo)

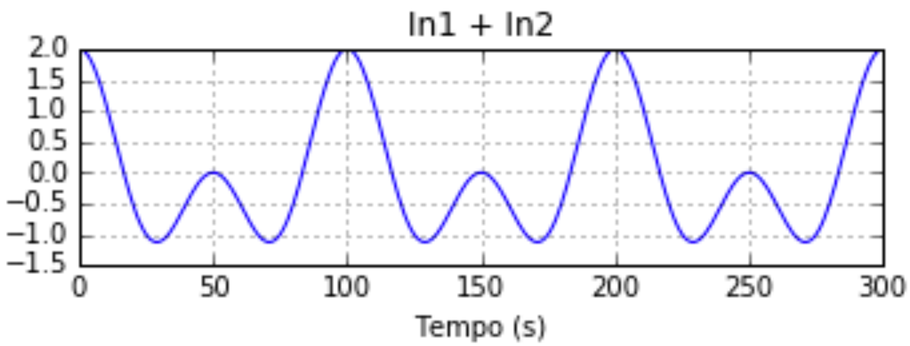


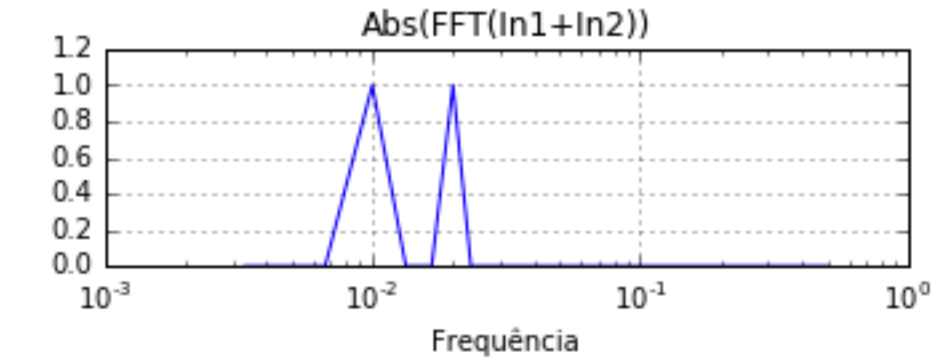
Aula passada

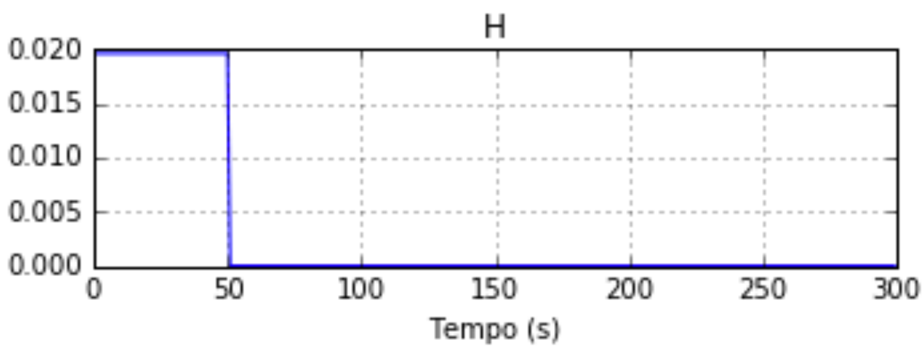
Filtro de média móvel (domínio espectral)

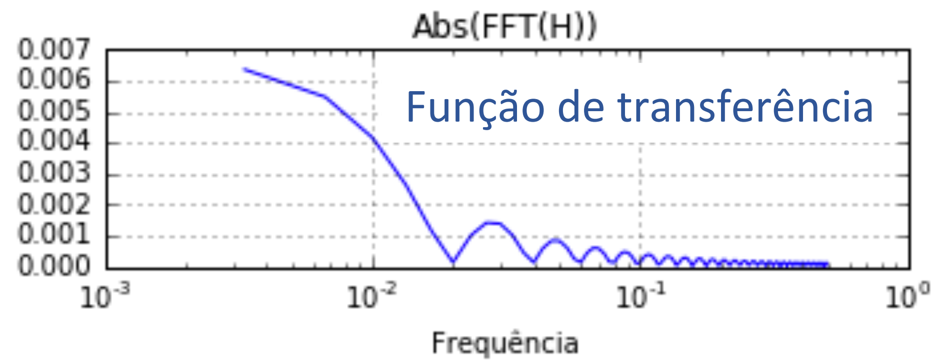
Domínio do tempo: convolução

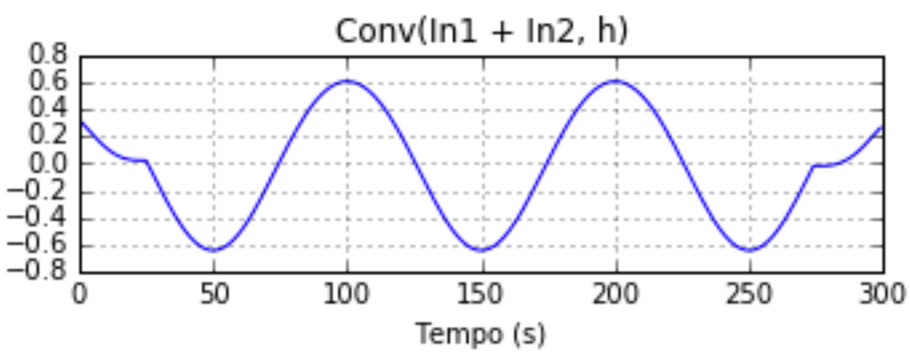
Domínio espectral: multiplicação

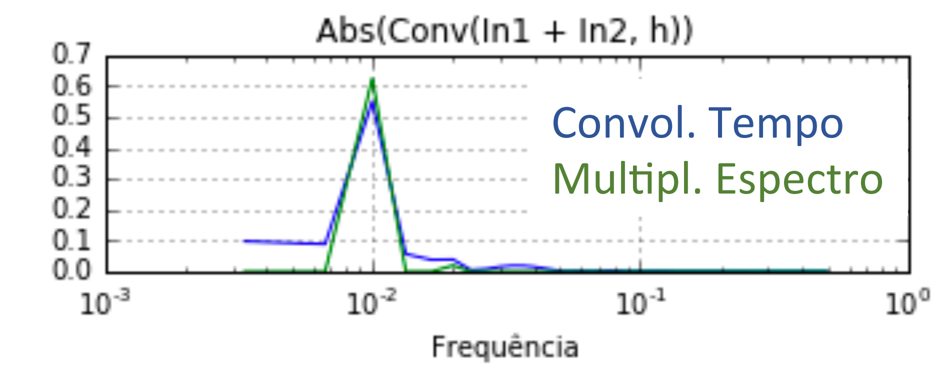












Filtro de média móvel (função de transferência)

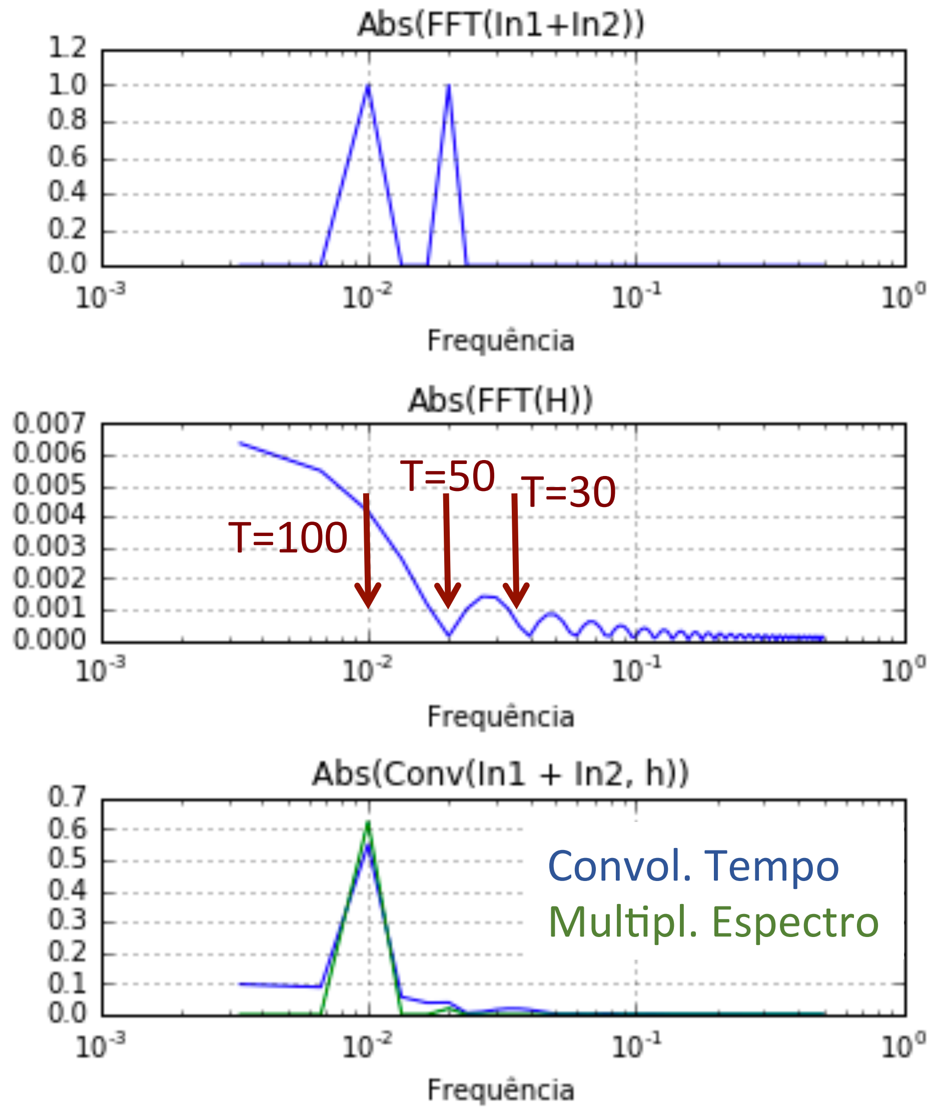
A função de transferência:

- Oscila
- Decai lentamente

É um mau filtro...

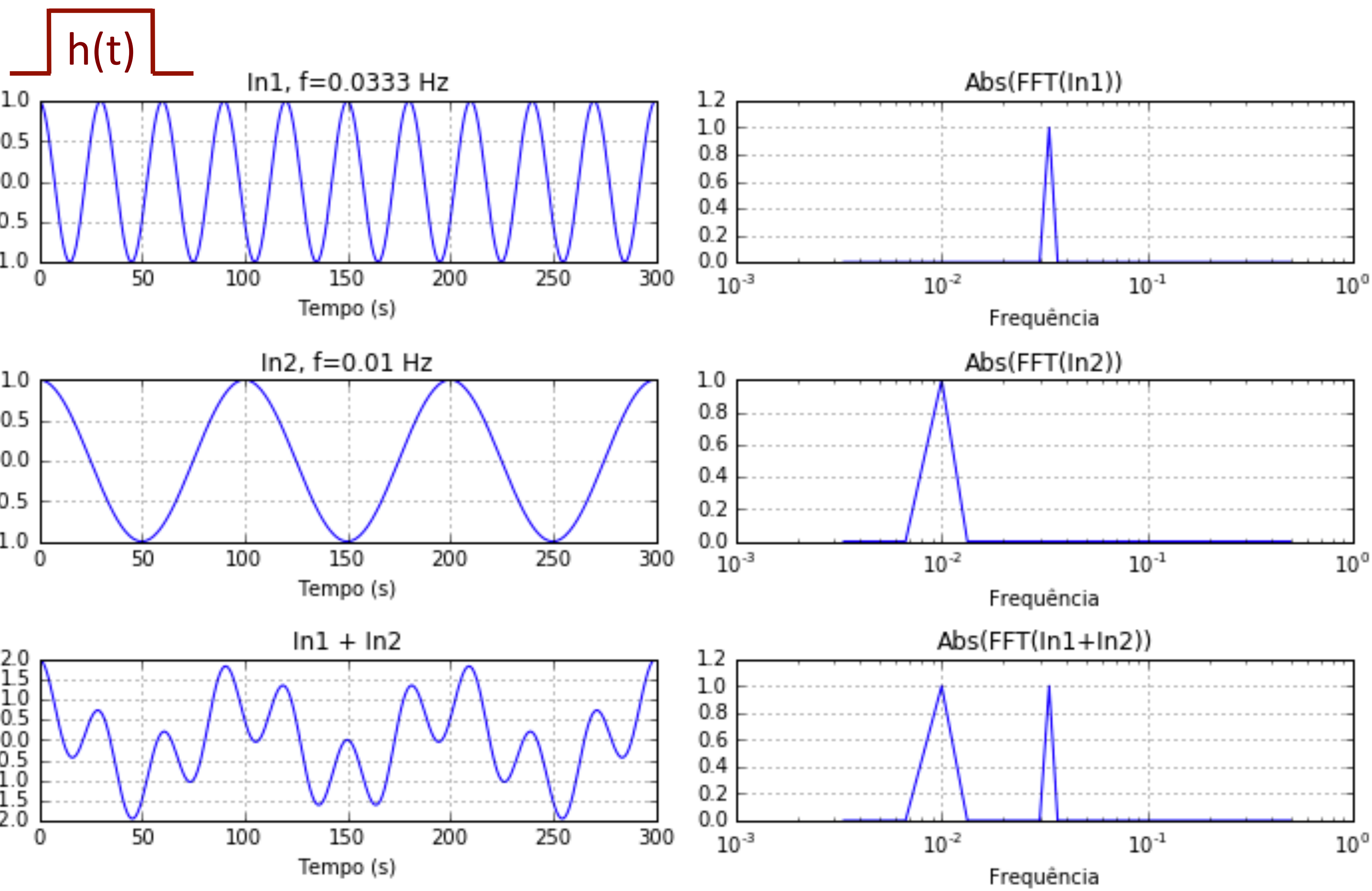
O comportamento do filtro de media móvel depende não só da frequência como também da sua localização exacta em relação às oscilações da função de transferência!

Domínio espectral: multiplicação

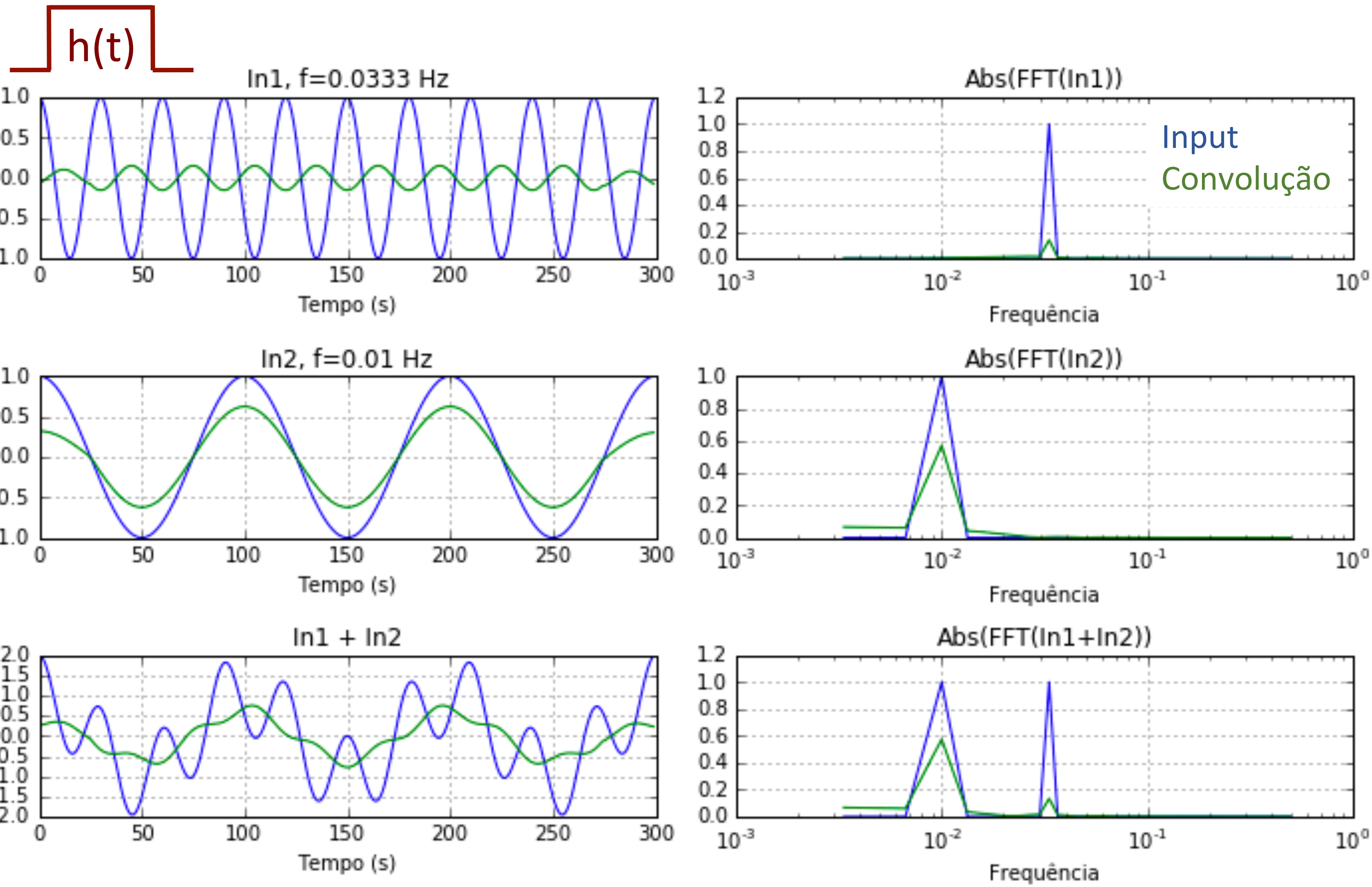


Aula passada

E com uma frequência mais alta? (T=30 s)

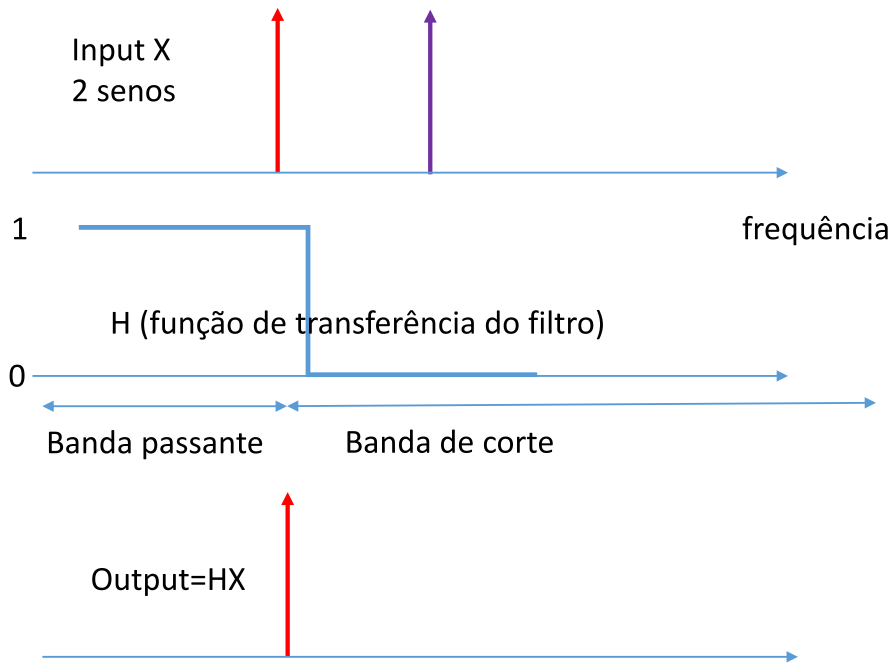


E com uma frequência mais alta? (T=30 s)
Convolução no domínio do tempo

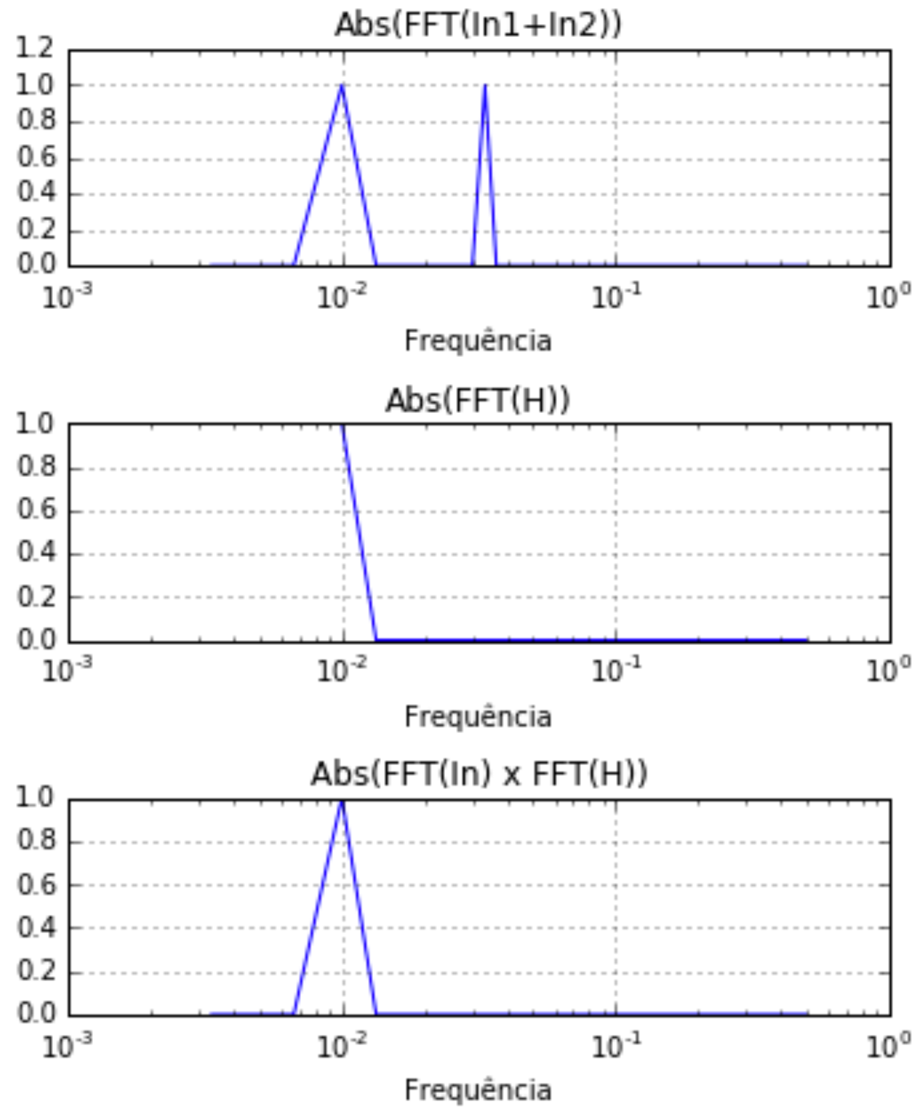


Aula passada

É melhor definir os filtros no domínio espectral



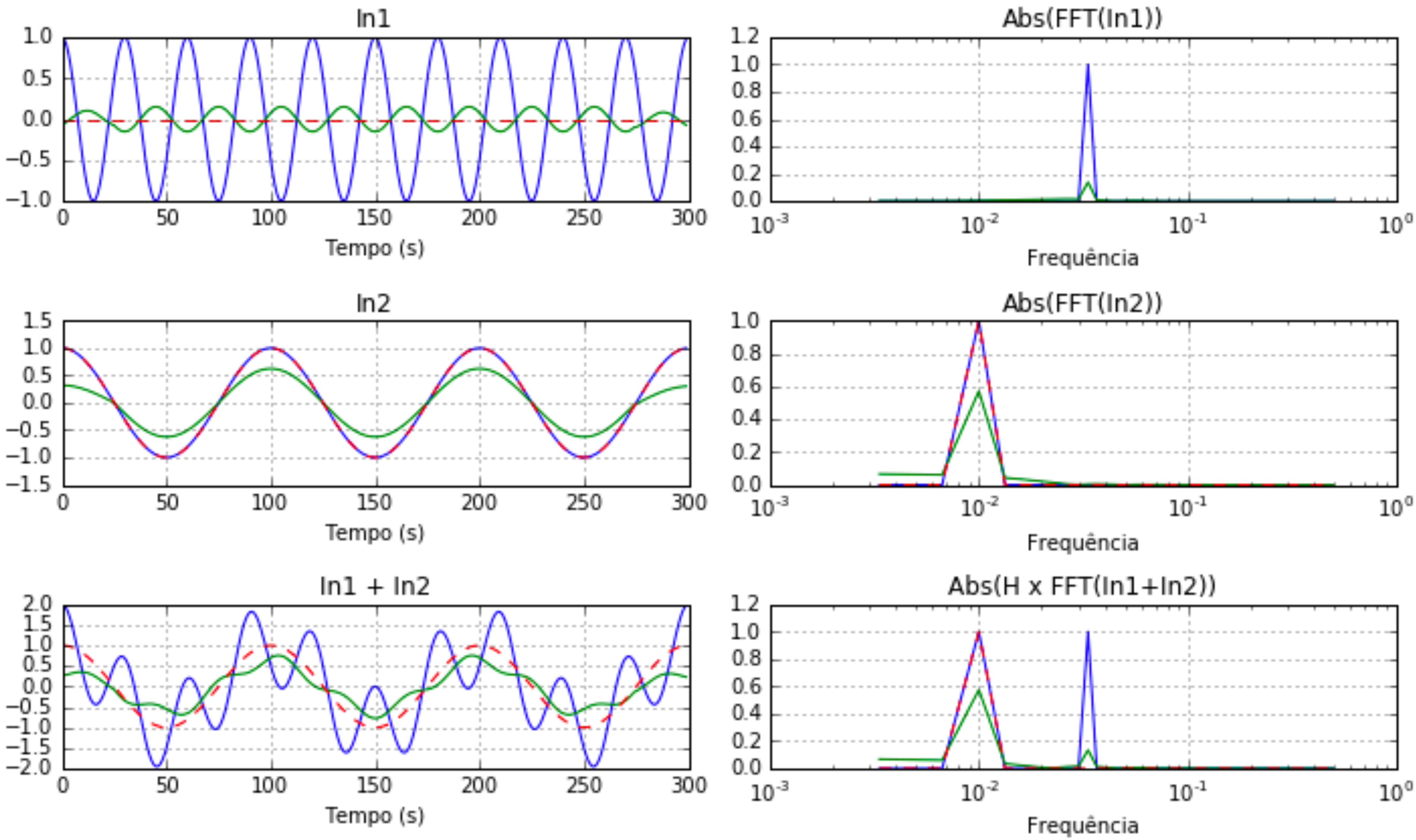
No caso discreto:



E com uma frequência mais alta? (T=30 s)
Multiplicação no domínio do espectro

Aula passada

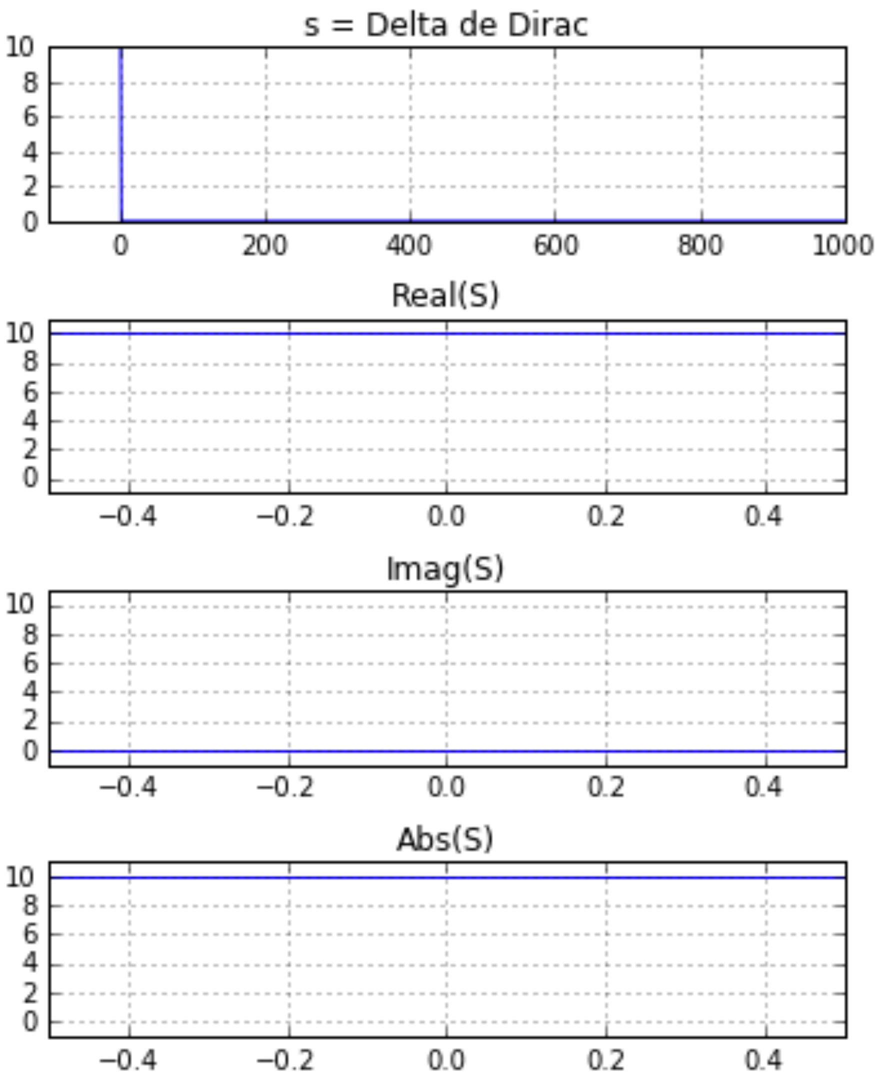
Input
Convol. Tempo
Multipl. Espectro



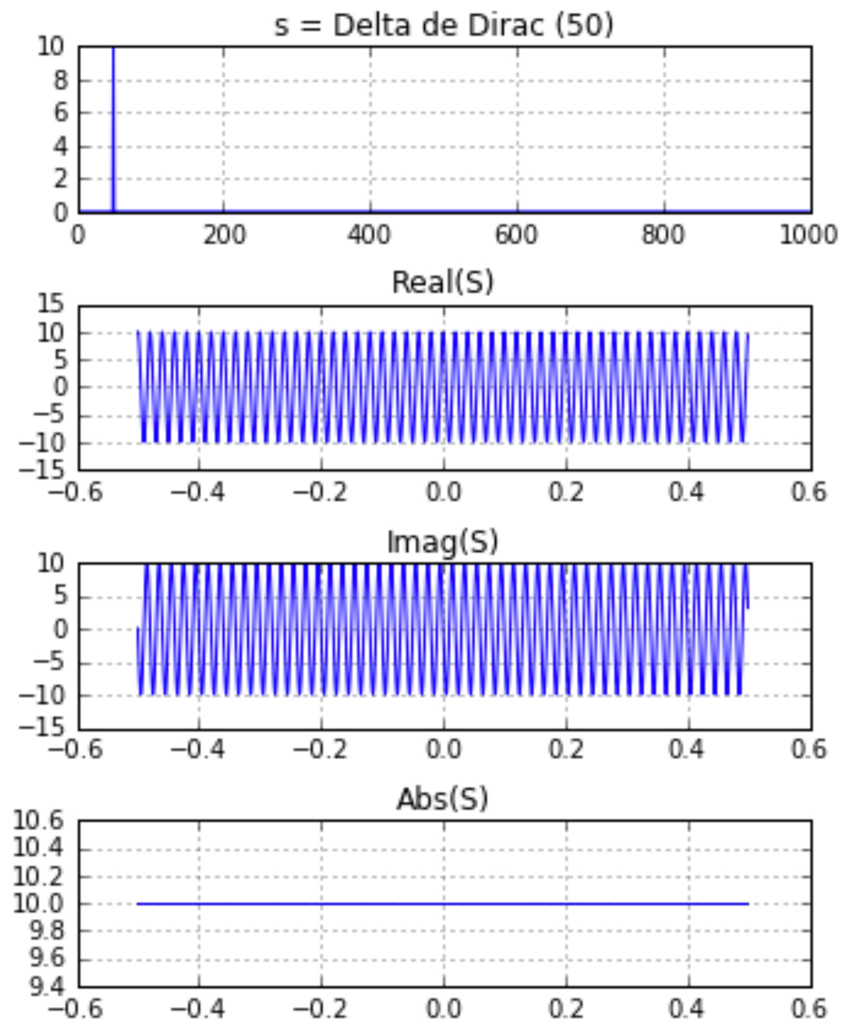
Aula passada

Transformadas de Fourier de séries importantes

$\mathcal{F}(\delta_0$ de Dirac) = constante



$\mathcal{F}(\delta_{50}$ de Dirac) = constante



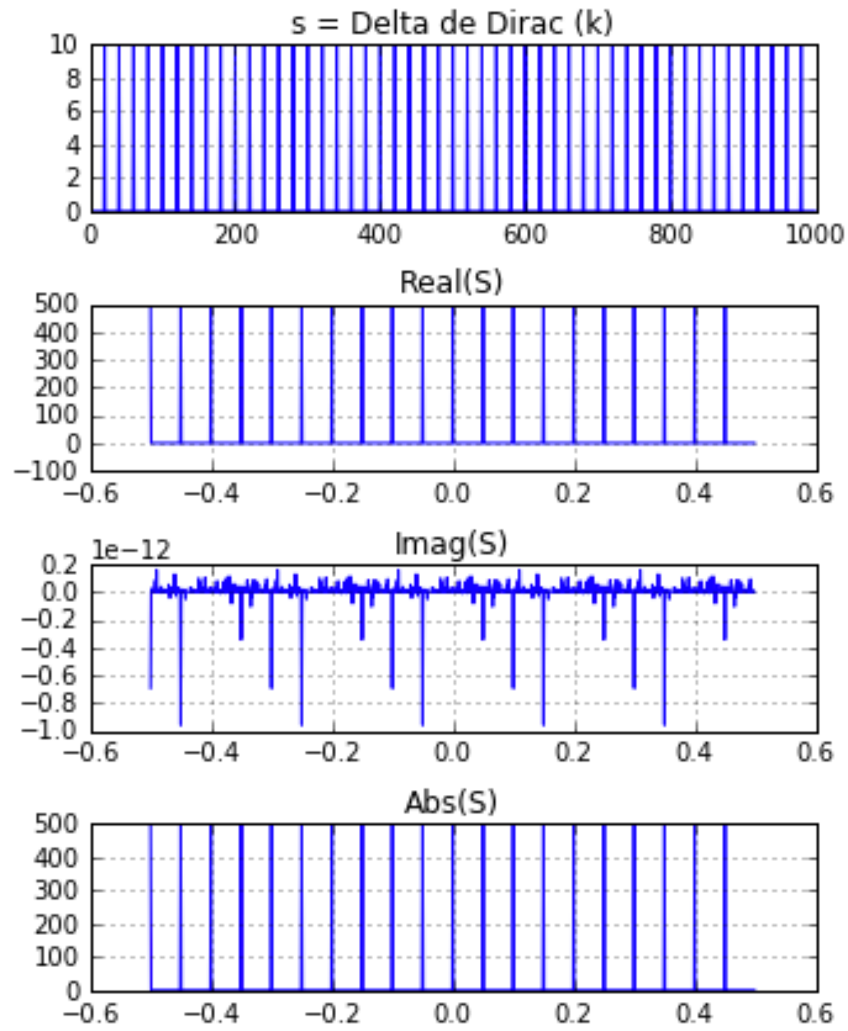
Aula passada

Transformadas de Fourier de séries importantes

```
#%% Pente
N=1000;
s=np.zeros(N)
t=np.arange(N)
dt=1.;
fNyq=1/(2*dt);
df=1/(dt*N);
freq=np.arange(-fNyq,fNyq,df);

#%%
s[range(0,N,20)]=10;
S=fft.fft(s);
SS=np.concatenate([S[N/2:], S[:N/2]])

plt.subplot(4,1,1);
plt.plot(t,s); plt.grid()
plt.title(u's = Delta de Dirac (k)');

plt.subplot(4,1,2);
plt.plot(freq,np.real(SS));
plt.title('Real(S)');  plt.grid()

plt.subplot(4,1,3);
plt.plot(freq,np.imag(SS));
plt.title('Imag(S)');  plt.grid()

plt.subplot(4,1,4);
plt.plot(freq,np.abs(SS));
plt.title('Abs(S)');  plt.grid()

plt.tight_layout()
```

Pente de Dirac: $\mathcal{F}(\delta_k \text{ de Dirac}) = \delta_n$

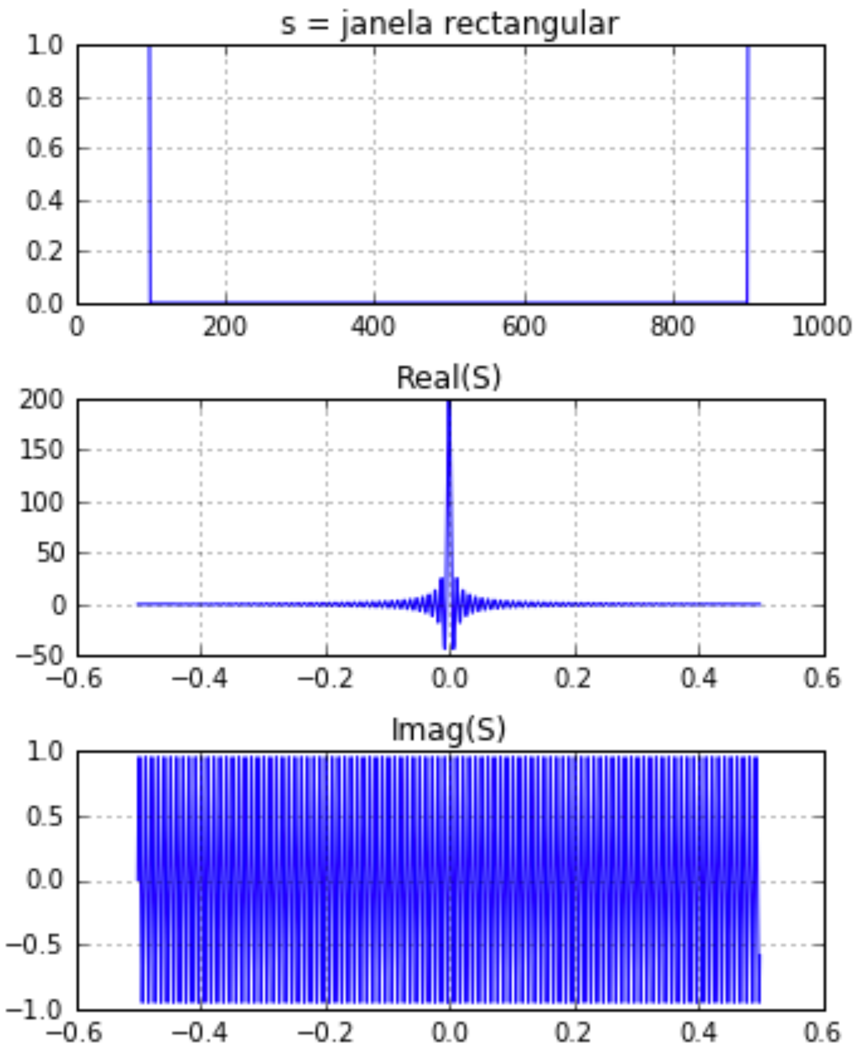
Amostrar regularmente = multiplicar por pente



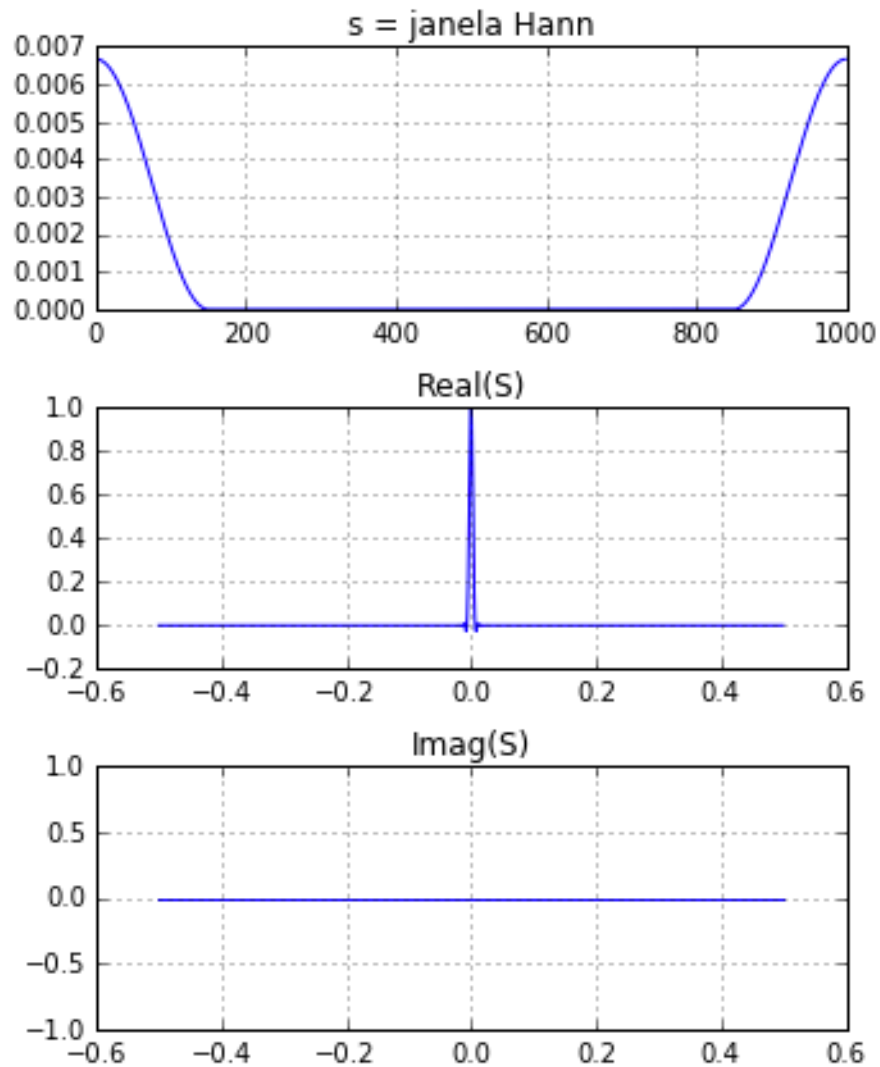
Aula passada

Transformadas de Fourier de séries importantes

Janela rectangular

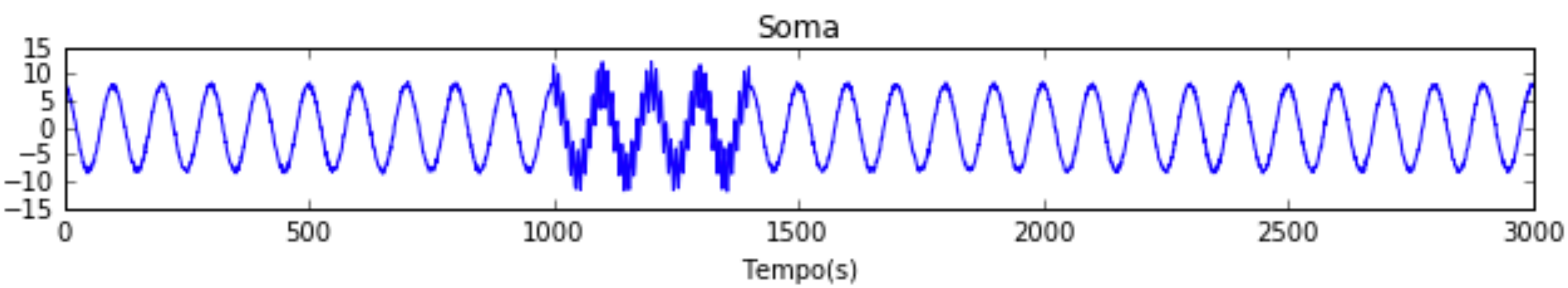


Janela de Hann

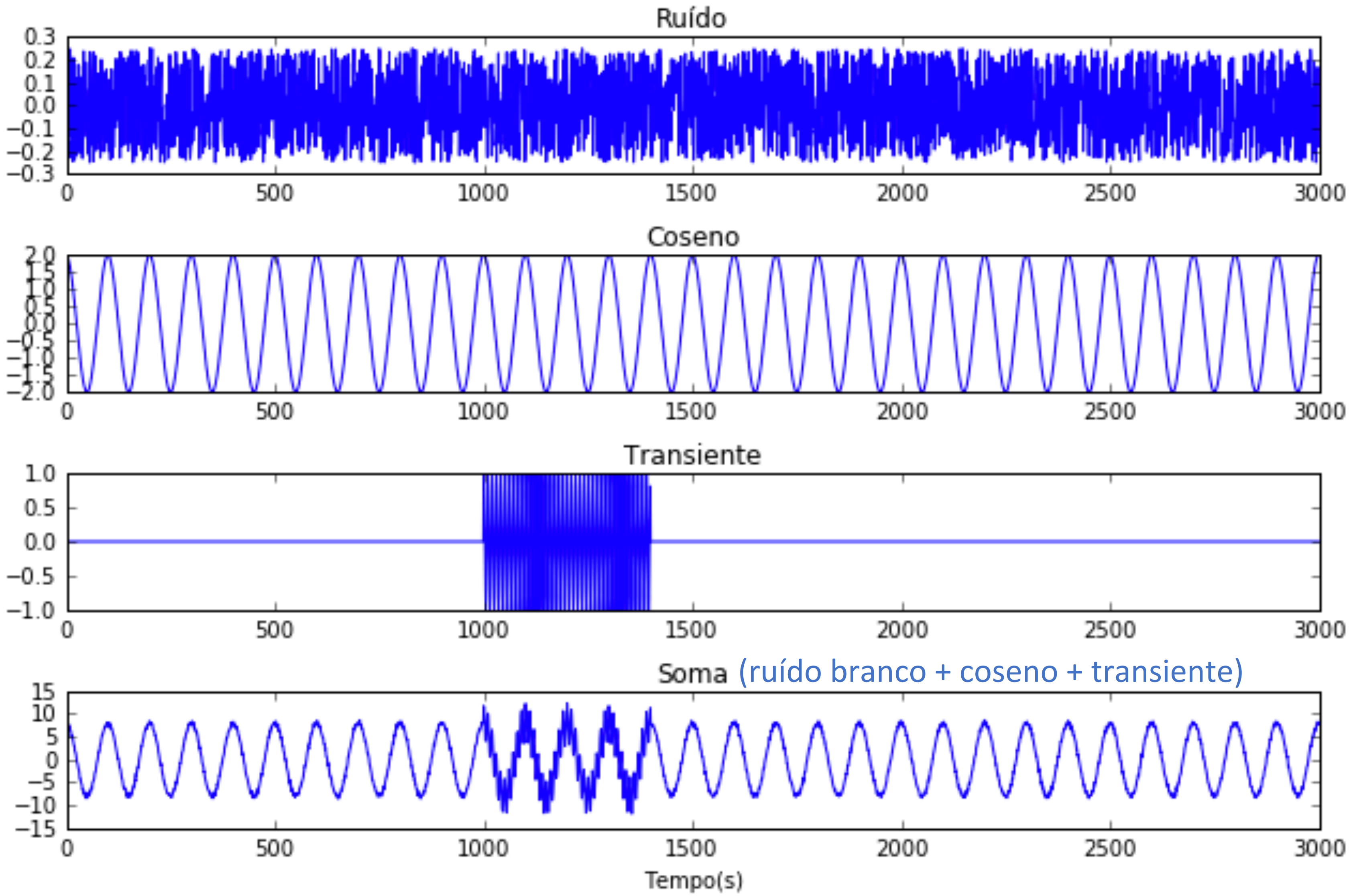


Espectrogramas

Muitas vezes pretendemos analisar a evolução temporal do espetro de um sinal. Por exemplo no caso de sinais que contém oscilações transientes de curta duração em certas regiões do espetro (e.g.: um tsunami de alta frequência sobreposto numa maré de baixa frequência). Pode também servir para analisar a variação annual da amplitude do ciclo diurno, e muitas outras coisas.



Exemplo



Ruído
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0
500
1000
1500
2000
2500
3000
Coseno
2.0
−2.0
Transiente
1.0
0.5
0.0
−0.5
−1.0
Soma (ruído branco + coseno + transiente)
15
10
5
0
−5
−10
−15
Tempo(s)

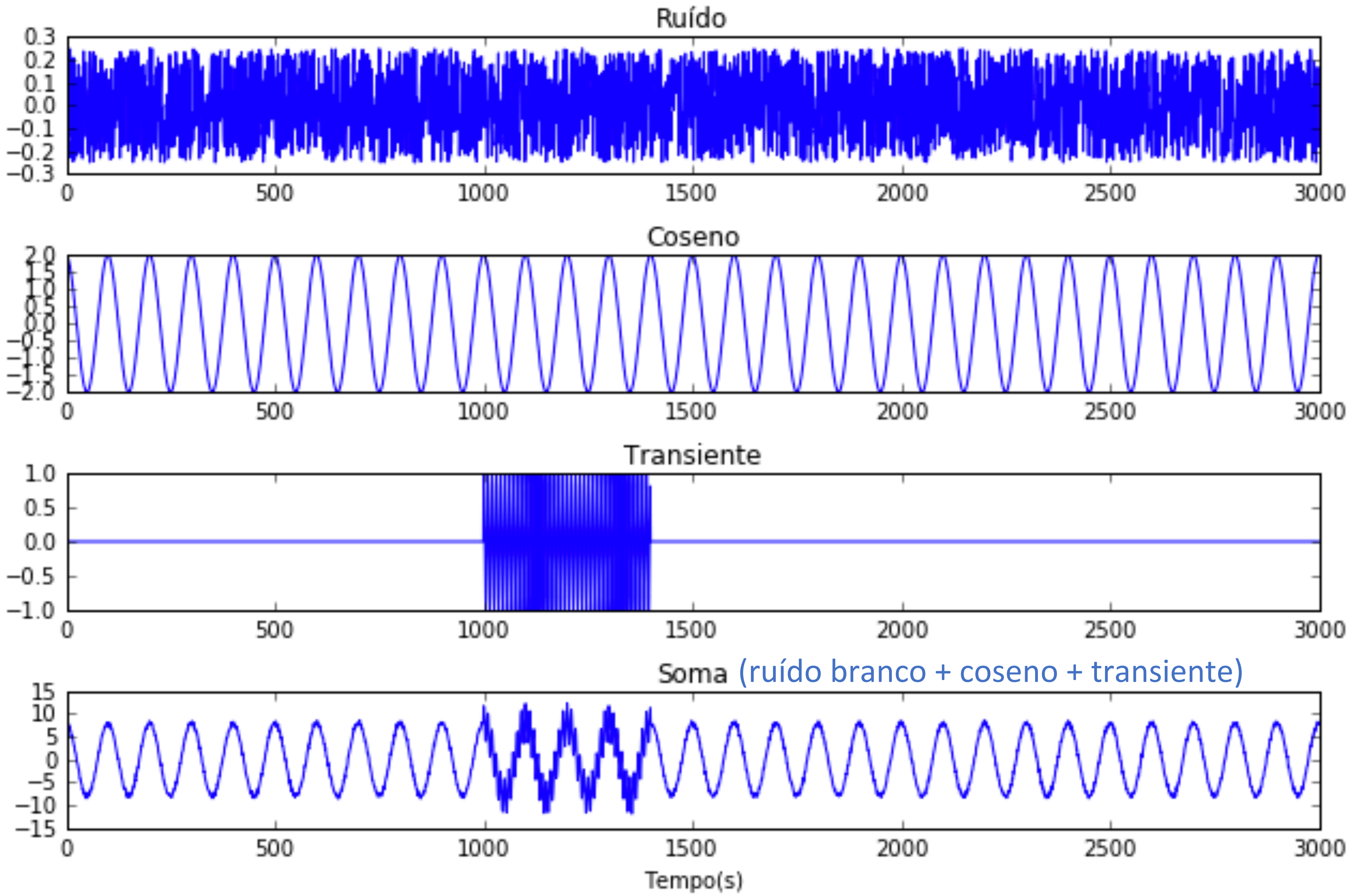
Exemplo - input

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import pi as pi
import numpy.fft as fft

plt.rcParams['figure.figsize'] = 10, 6

#%% Parâmetros
dt=1.                          # intervalo de amostragem
N=3000                        # número de pontos da amostra
t=np.arange(0,N*dt,dt)        # vector de tempos
T=np.array([50.,100.,10.])    # períodos dos vários sinais
NS=len(T);                    # número de sinais
NJ=200                        # número de pontos da janela do espectrograma

omega=2.*pi/T                 # frequências angulares dos vários sinais
ampsin=np.array([0.,2.,1.])    # amplitudes dos sinais
ampnoi=np.array([0.5,0.,0.])  # amplitude do ruído
IsT=np.array([0,0,1])         # O sinal é transiente? 0=não, 1=sim
transient=np.zeros(N)         # vector com sinal transiente
transient[5*NJ:7*NJ]=1        # pontos do sinal transiente diferentes de zero
s=np.zeros([N,NS])            # matriz com sinais, inicializada com zeros
sT=np.zeros(N)                # vector com a soma de todos os sinais
tit=[u'Ruído', 'Coseno', 'Transiente', 'Soma']   # Título dos plots
```

Exemplo - input

```
#%% Sinais
plt.close();
plt.rcParams['figure.figsize'] = 9, 6

for i in range(NS):
    s[:,i] = ampsin[i]*np.cos(omega[i]*t) + ampnoi[i]*(np.random.rand(N)-0.5)
    if IsT[i]==1:
        s[:,i]=s[:,i]*transient

    sT=sT+s[:,i]

    # Plot
    plt.subplot(NS+1, 1, i+1);
    plt.plot(t,s[:,i])
    plt.title(tit[i])

plt.subplot(NS+1, 1, 4);
plt.plot(t,sT)
plt.xlabel('Tempo(s)')
plt.title(tit[3])

plt.tight_layout()
```

Exemplo - input

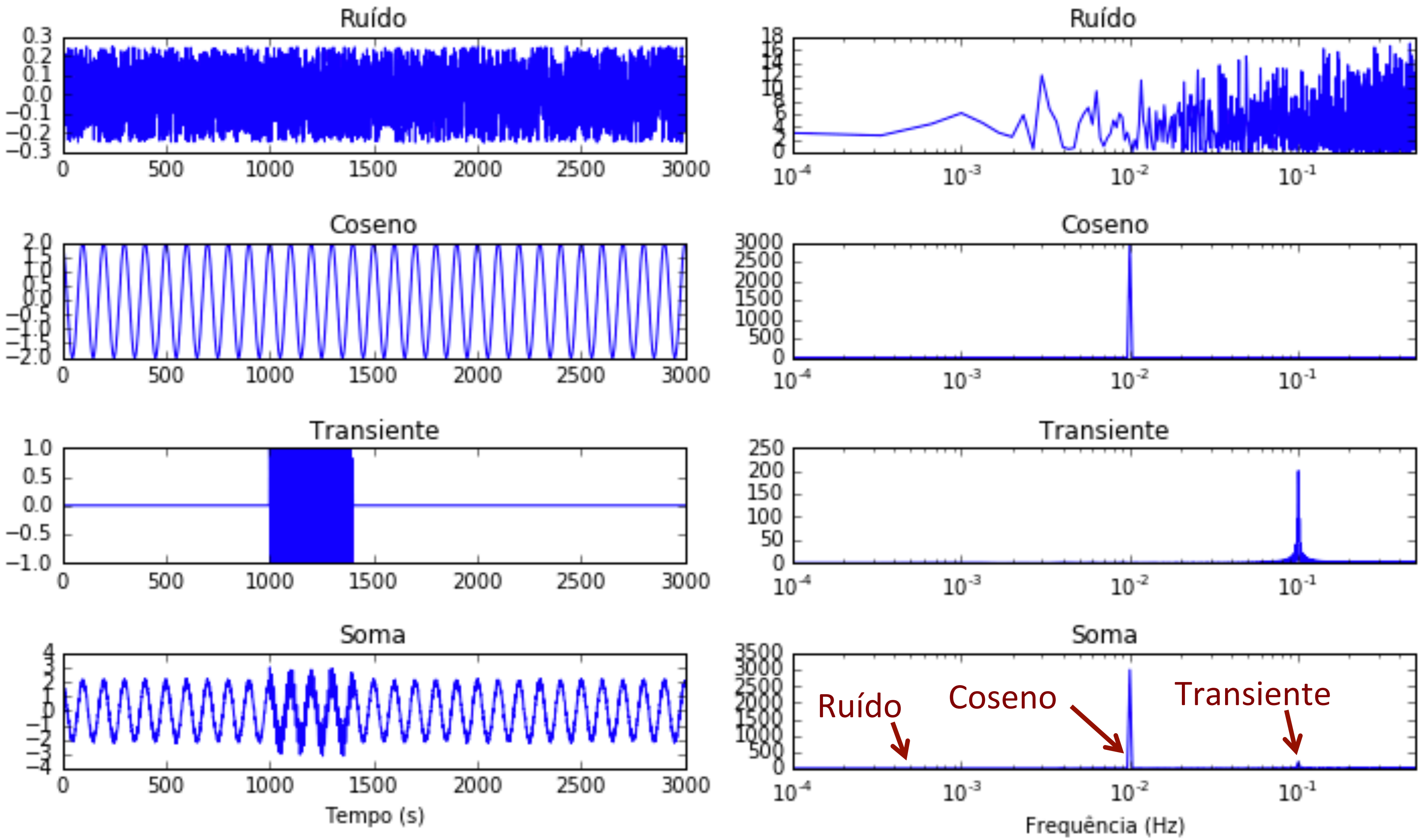


```python
#%% Espectros
tf = s*0.                    # Matriz dos espectros. Iniciar com zeros.
fNyq=1./(2.*dt);             # Frequência de Nyquist
df=1./(N*dt);                # Espaçamento da amostragem espectral
freq=np.arange(-fNyq, fNyq, df)     # Vector das frequências

#%% Plot
plt.close();
plt.rcParams['figure.figsize'] = 10, 6

for i in range(NS):
    tf[:,i]=fft.fft(s[:,i])     # Calcular os espectros

    plt.subplot(NS+1, 2, i*2+1);
    plt.plot(t,s[:,i])
    plt.title(tit[i])

    plt.subplot(NS+1, 2, i*2+2);
    plt.semilogx(freq[N/2:], np.abs(tf[:,i])[:N/2])
    plt.title(tit[i])
    plt.xlim([1e-4, fNyq])

tfT=fft.fft(sT)         # Calcular espectro

i=3
plt.subplot(NS+1, 2, i*2+1);
plt.plot(t,sT)
plt.title(tit[i])
plt.xlabel('Tempo (s)')

plt.subplot(NS+1, 2, i*2+2);
plt.semilogx(freq[N/2:], np.abs(tfT)[:N/2])
plt.title(tit[i])
plt.xlim([1e-4, fNyq])
plt.xlabel(u'Frequência (Hz)')

plt.tight_layout()
```

Exemplo

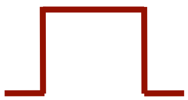
Domínio do tempo

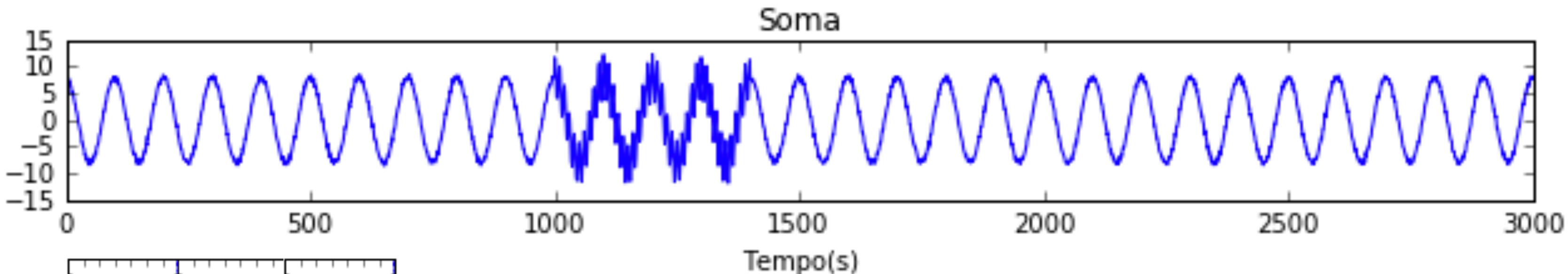
Domínio espectral

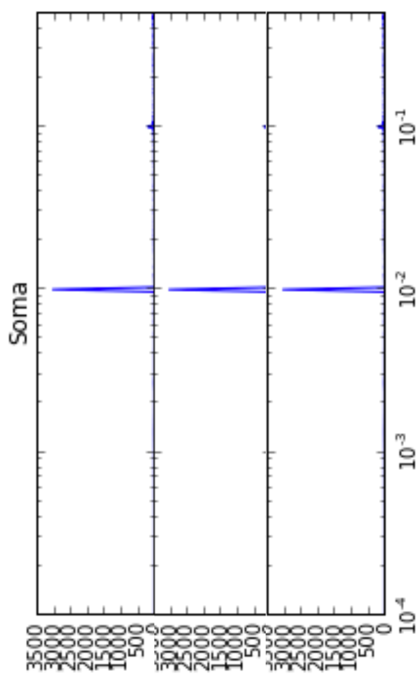


Espectrograma

Janela móvel (NJ = 200)





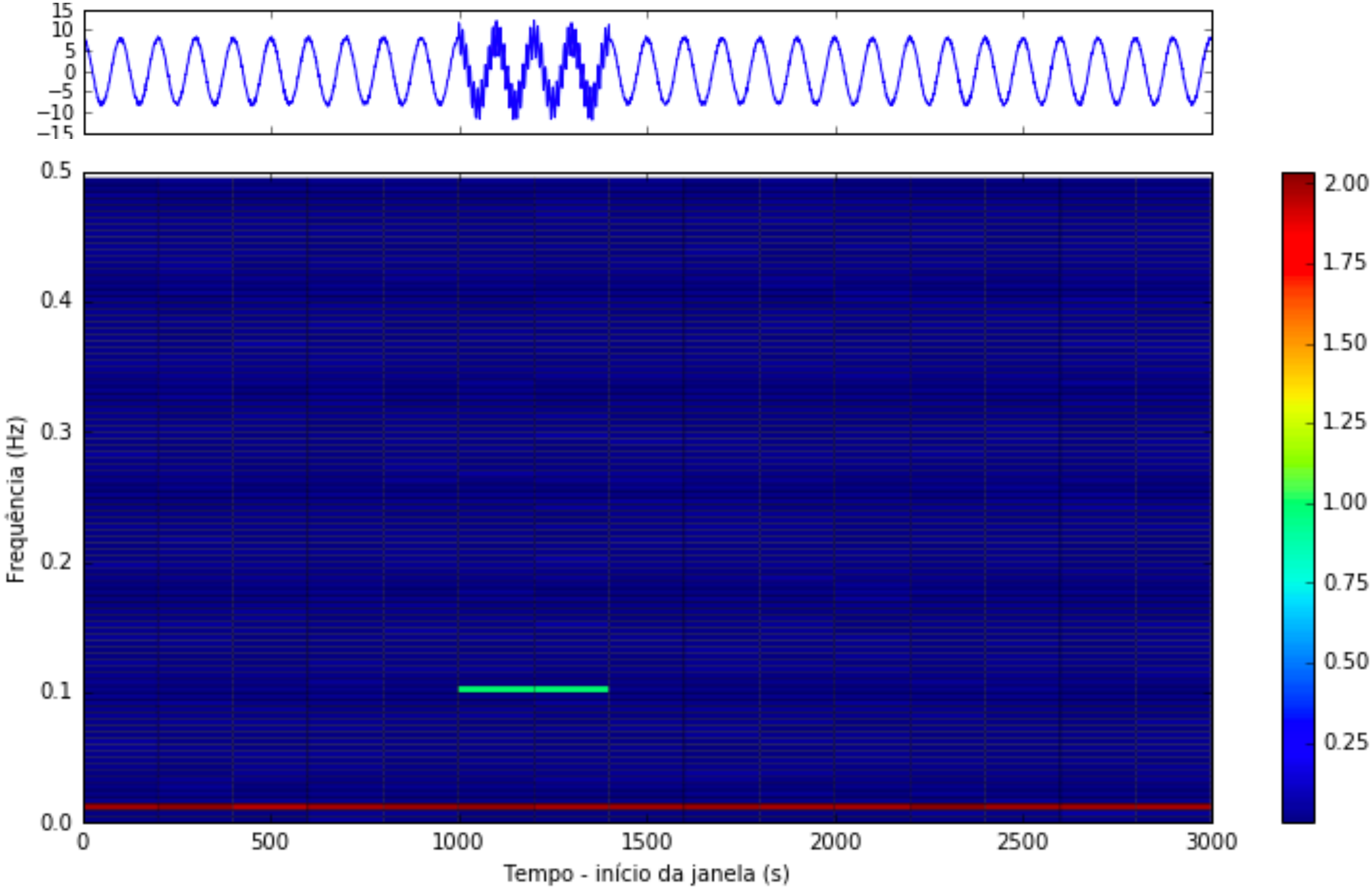


... => Para cada janela, calculamos um espectro.

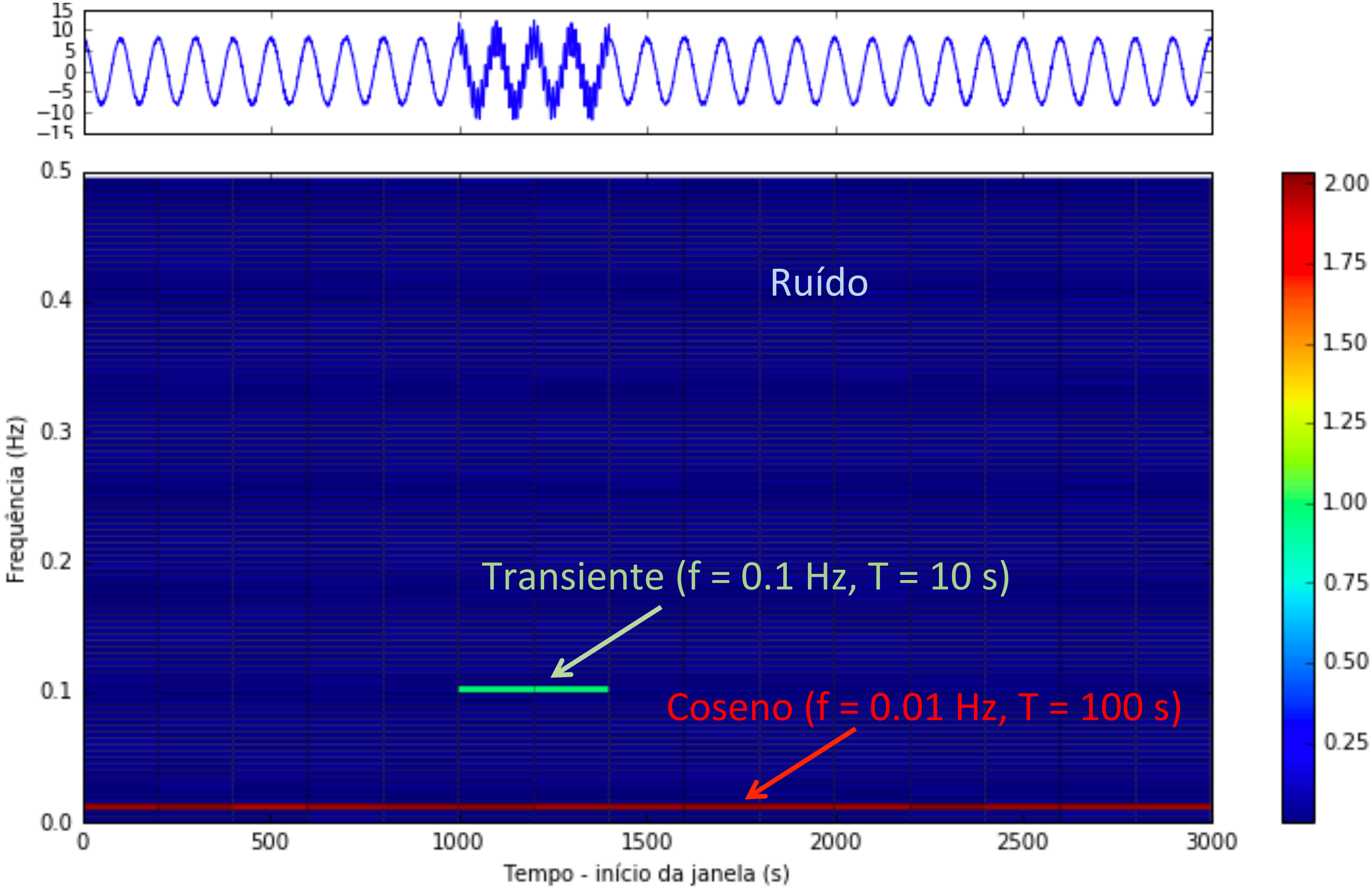
Espectrograma

```
#%% Espectrograma
NJ2=NJ/2                   # metade no número de pontos na janela do espectrograma
df=1./(NJ*dt)               # amostragem no espectro correspondente à janela escolhida
freq=np.arange(0,fNyq,df)   # vector de frequências do espectro de cada janela
tstart=np.arange(0,N*dt+NJ,NJ);  #inícios da janela móvel
Njanelas=len(range(0, N, NJ))    # número de janelas do espectrograma
specgram=np.zeros([NJ2,Njanelas])   # matriz com espectros para cada janela

kw=0;                       # Número da janela; inicializado a zero
for k in range(0, N, NJ):    # ciclo para cada janela
    sw=sT[k:k+NJ]              # Cortar o sinal (soma) em cada janela
    Fw=fft.fft(sw)      # Espectro da janela de sinal
    AFw=np.abs(Fw[:NJ2])/NJ2
    specgram[:,kw]=AFw;
    kw=kw+1                    # Número da janela: 0, 1, 2, ...

#%% Plot
plt.close();
plt.rcParams['figure.figsize'] = 10, 5

plt.pcolor(tstart, freq, specgram, edgecolors='k')
plt.colorbar()
plt.xlabel(u'Tempo - início da janela (s)')
plt.ylabel(u'Frequência (Hz)')


plt.tight_layout()
```

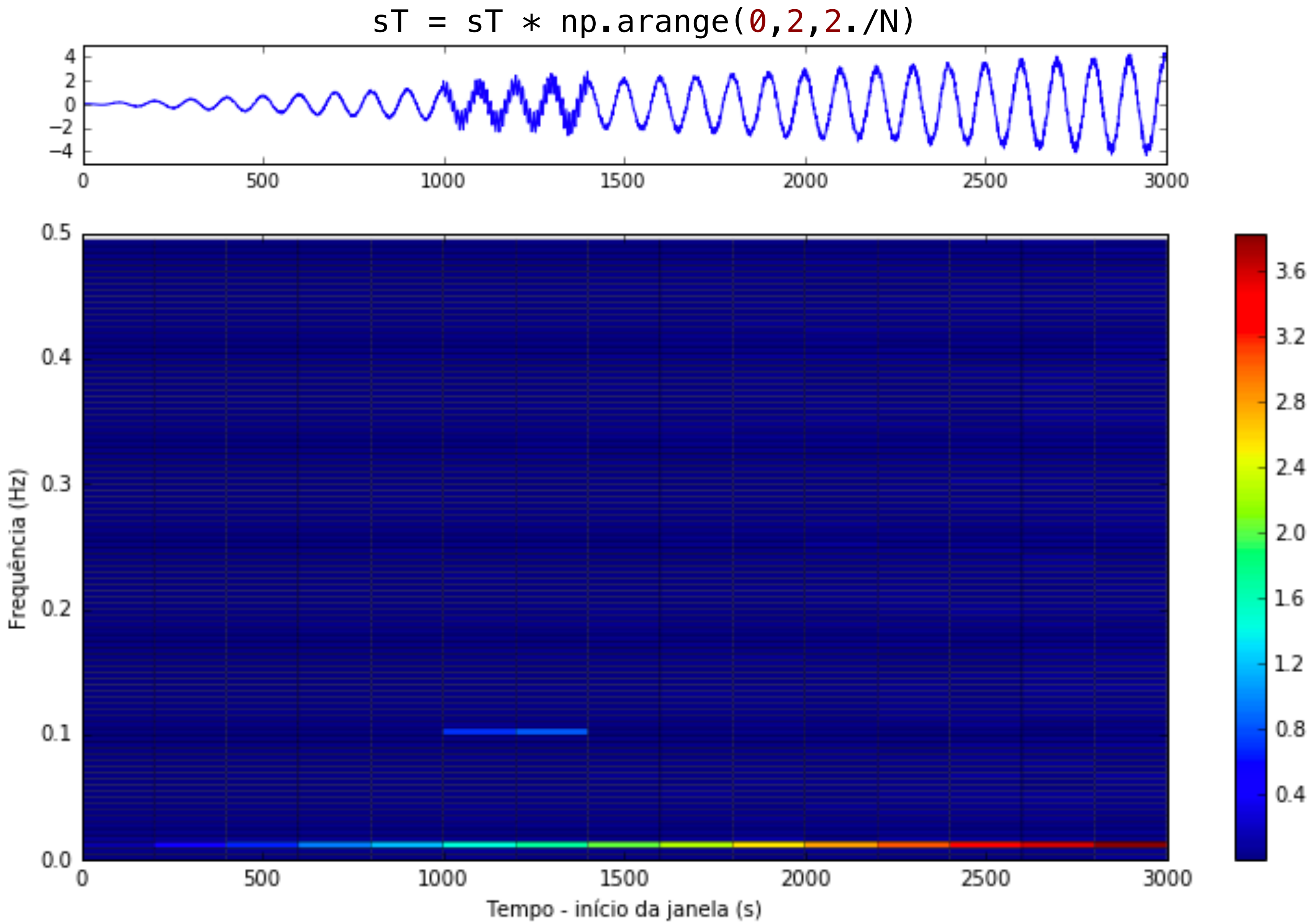
Espectrograma



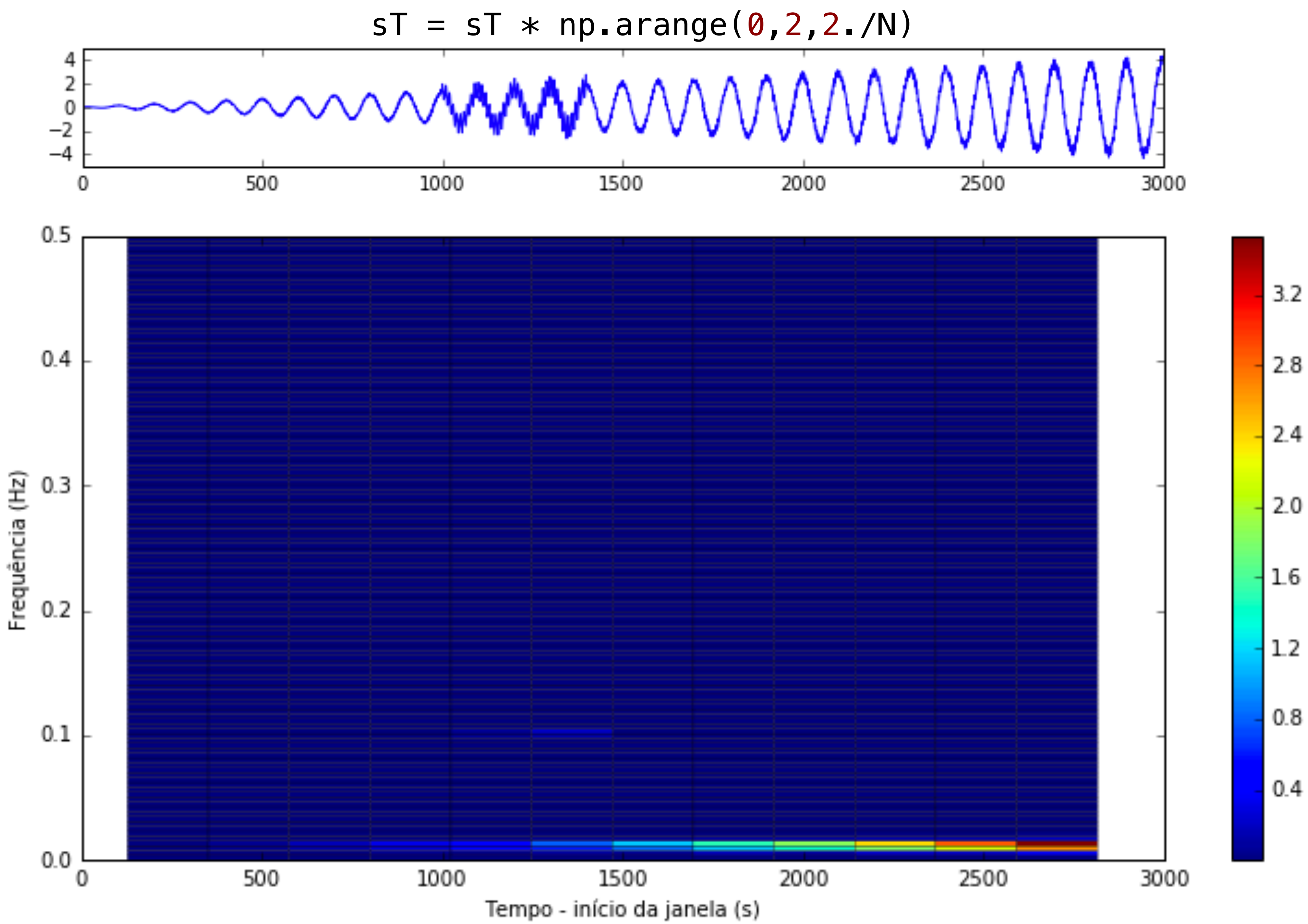
Espectrograma



Espectrograma de sinal variável



Espectrograma de sinal variável



```python
#%% Scipy spectrogram
from scipy import signal

fs, ts, Sxx = signal.spectrogram(sT, 1/dt, scaling='spectrum')

plt.close(); plt.rcParams['figure.figsize'] = 10, 5

plt.pcolor(ts, fs, Sxx, edgecolors='k')
plt.colorbar()
plt.xlabel(u'Tempo - início da janela (s)')
plt.ylabel(u'Frequência (Hz)')

plt.tight_layout()
```