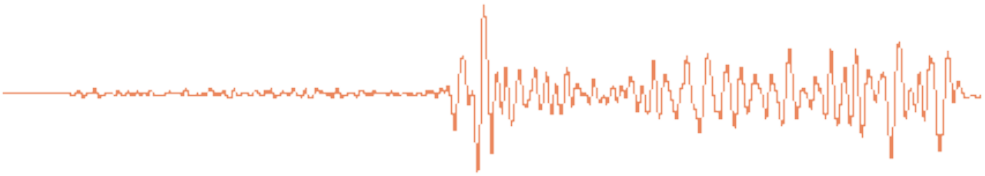


8. Filtros

Em processamento de sinais, um filtro digital é um sistema que executa uma operação matemática sobre um sinal discreto, com o objetivo de reduzir ou ampliar algum aspecto do sinal.

Em linguagem matemática, dado um sinal $f(t)$, definimos

$$g(t) = (f * h)(t) \tag{8.1}$$

o sinal f filtrado pelo filtro h.

Exercício 8.1 Seja $\hat{B}_a(\omega)$ um filtro passa-baixa no domínio da frequência, dado por

$$\hat{B}_a(\omega) = \begin{cases} 1 & ,|\omega| \leq a; \\ 0 & ,|\omega| > a. \end{cases} \tag{8.2}$$

Calcule o filtro $B_a(t)$ no domínio do tempo.

■

8.1 Filtro passa-baixa

Chamamos de filtro passa-baixa o sinal h que mantém do sinal f as baixas frequências, e elimina as altas frequências. Quando queremos eliminar ruídos de alta frequência, por exemplo, utilizamos este tipo de filtro. Lembrando que a eq. (8.1) pode ser representada no domínio da frequência por

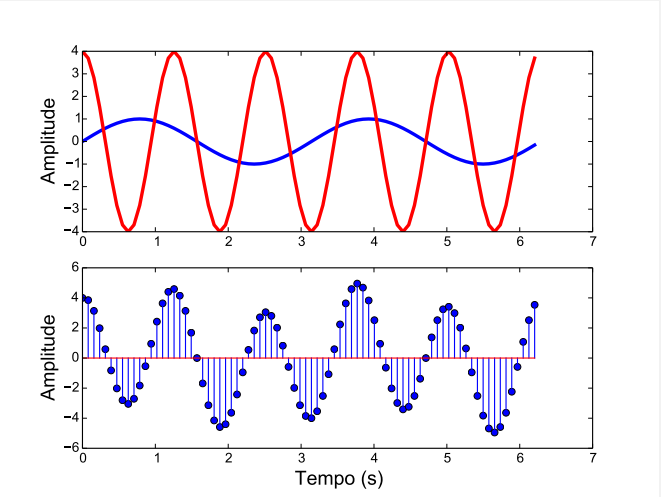
$$\hat{g}(\omega) = \hat{f}(\omega) \cdot \hat{h}(\omega), \tag{8.3}$$

um exemplo de filtro passa-baixa pode ser dado por

$$\hat{h} = \begin{cases} 1, & \omega \leq \omega_a, \\ 0, & \omega > \omega_a. \end{cases} \tag{8.4}$$

BOX 8.1.1 — FILTRO PASSA BAIXA. Para entendermos a ideia de um filtro passa baixa, vamos fazer um exemplo utilizando uma função simples.

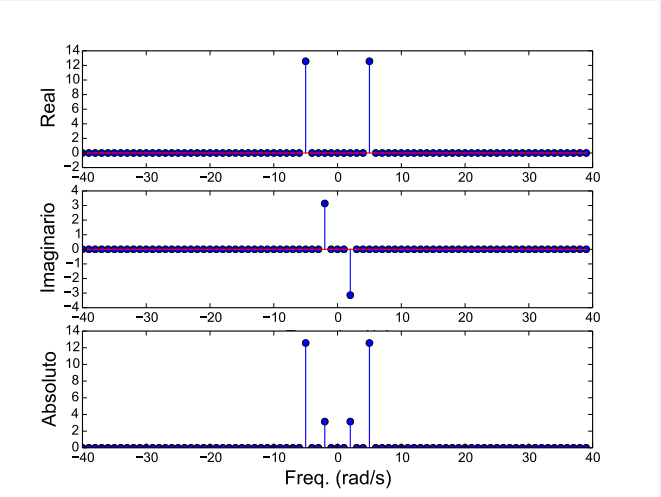
```
from numpy import fft
import numpy as np
import matplotlib.pyplot as plt

#Vamos criar um sinal no tempo
n = 80   # Numero de amostras
dt= np.pi/40    # Intervalo amostral (em segundos)
t =dt*np.arange(0,n)   # Eixo do tempo
f1 = 2.0  # frequencia (Se 2pi/f1 nao for divisor de dt*n, gera ...
    leakage)
f2 = 5.0   # frequencia

#Vamos gerar um sinal periodico
f_sin=1*np.sin(t*f1)
f_cos= 4*np.cos(t*f2)
fx= f_cos + f_sin # sinal


fig=plt.figure()
ax1=fig.add_subplot(211)
ax1.plot(t,f_sin,lw=3)  # Plot spectral power
ax1.plot(t, f_cos, color="red", lw=3)
ax1.set_ylabel('Amplitude', size = 'x-large')

ax2=fig.add_subplot(212)
ax2.stem(t, fx,  label='fx')
ax2.set_ylabel('Amplitude', size = 'x-large')
ax2.set_xlabel('Tempo (s)', size = 'x-large')
plt.show()
```



Observe o sinal `fx` (gráfico de baixo). Ele é formado pela soma de um sinal seno e um sinal cosseno (ilustrados no primeiro gráfico), e cada uma destas funções possui uma componente em

frequência distinta. O script a seguir, mostra como podemos visualizar o espectro de frequências deste sinal, utilizando a função `fft`:

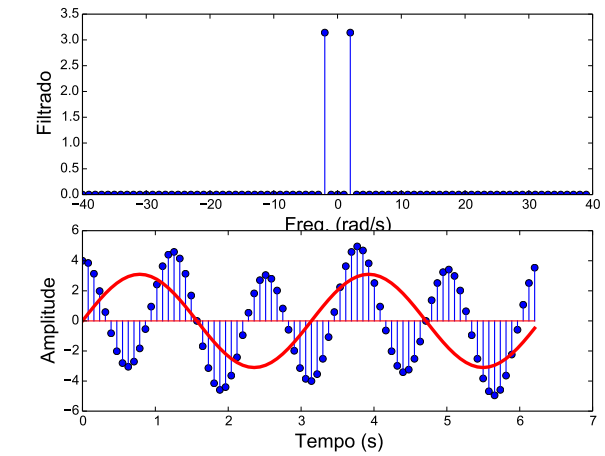
```
from numpy import fft
import numpy as np
import matplotlib.pyplot as plt

#Vamos criar um sinal no tempo
n = 80   # Numero de amostras
dt= np.pi/40    # Intervalo amostral (em segundos)
t =dt*np.arange(0,n)   # Eixo do tempo
f1 = 2.0  # frequencia (Se 2pi/f1 nao for divisor de dt*n, gera ...
    leakage)
f2 = 5.0   # frequencia

#Vamos gerar um sinal periodico
f_sin=1*np.sin(t*f1)
f_cos= 4*np.cos(t*f2)
fx= f_cos + f_sin # sinal

#Calcular a transformada de Fourier do sinal fx
F= dt*fft.fft(fx)   # Coeficientes de Fourier (vezes dt)
w = 2*np.pi*fft.fftfreq(n,dt)  # Eixo de frequencias (dw=1/N/dt)
F= fft.fftshift(F)   # Shift zero para o centro
w = fft.fftshift(w)   # Shift zero para o centro

fig=plt.figure()
ax1=fig.add_subplot(311)
ax1.stem(w,np.real(F),lw=3)   # Plot termos em cossenos
ax1.set_ylabel('Real', size = 'x-large')
ax2=fig.add_subplot(312)
ax2.stem(w, np.imag(F),  label='fx') # Plot termos em senos
ax2.set_ylabel('Imaginario', size = 'x-large')
ax2.set_xlabel('Freq. (rad/s)', size = 'x-large')
ax3=fig.add_subplot(313)
ax3.stem(w, np.abs(F),  label='fx') # Plot termos em senos
ax3.set_ylabel('Absoluto', size = 'x-large')
ax3.set_xlabel('Freq. (rad/s)', size = 'x-large')
plt.show()
```



No gráfico acima estão plotados os gráficos da Transformada de Fourier de `fx` real (componentes dos cossenos), imaginária (componente dos senos) e o valor absoluto do sinal filtrado.

Para filtrarmos o sinal, basta eliminarmos as frequências altas do sinal. No domínio da frequência, isso equivale a multipicar todas as frequências acima de um threshold por 0, como fzemos com a função `abs`:

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

#Vamos criar um sinal no tempo
n = 80   # Numero de amostras
dt= np.pi/40    # Intervalo amostral (em segundos)
t =dt*np.arange(0,n)   # Eixo do tempo
f1 = 2.0  # frequencia (Se 2pi/f1 nao for divisor de dt*n, gera ...
    leakage)
f2 = 5.0   # frequencia

#Vamos gerar um sinal periodico
f_sin=1*np.sin(t*f1)
f_cos= 4*np.cos(t*f2)
fx= f_cos + f_sin # sinal

#Calcular a transformada de Fourier do sinal fx
F= fft.fft(fx)*dt   # Coeficientes de Fourier (vezes dt)
w = 2*np.pi*fft.fftfreq(n,dt)  # Eixo de frequencias (dw=1/N/dt)
F= fft.fftshift(F)   # Shift zero para o centro
w = fft.fftshift(w)   # Shift zero para o centro

#Aqui vamos filtrar o sinal
F_filtered = F * (abs(w) < 3.0)
f_t_filtered = np.fft.ifft(np.fft.ifftshift(F_filtered))/dt
print(f_t_filtered[0])

fig=plt.figure()
ax1=fig.add_subplot(211)
ax1.stem(w,np.absolute(F_filtered))  # Plot espectro de frequencias
ax1.set_ylabel('Filtrado', size = 'x-large')
ax1.set_xlabel('Freq. (rad/s)', size = 'x-large')
#ax1.set_xlim([-1,1])
ax2=fig.add_subplot(212)
ax2.stem(t,fx)
ax2.plot(t, f_t_filtered, color="red", lw=3)
ax2.set_ylabel('Amplitude', size = 'x-large')
ax2.set_xlabel('Tempo (s)', size = 'x-large')
plt.show()
```



No gráfico acima, podemos observar que no domínio da frequência a componente mais alta foi apagada. Abaixo, mostramos o sinal no domínio do tempo, original (em azul) e filtrado (em vermelho). Observe que o sinal que restou é idêntico ao sinal seno, que é a componente com frequência mais baixa do sinal `fx`.

(Inserir exemplo fenômeno de Gibbs)

Se $f(x)$ é uma função suave por partes, o Teorema de Fourier garante que a Série de Fourier da $f(x)$ converge uniformemente para f em todo o intervalo fechado que não contém pontos de descontinuidade de f.

Se houver um ponto de descontinuidade neste intervalo I, a convergência não poderá ser uniforme em I. Gibbs estudou a convergência da série de Fourier próximo a um ponto p de descontinuidade e descobriu uma curiosidade, que é conhecida como fenômeno de Gibbs.

8.2 Filtro passa-banda

Em alguns problemas práticos, é comum estarmos interessados em apenas uma banda específica do sinal. Nestes casos, podemos aplicar um filtro passa-banda. No domínio da frequência, ele seria definido por

$$\hat{D}(\omega) = \begin{cases} 1, & \omega_a \leq |\omega| \leq \omega_b' \\ 0, & c.c. \end{cases} = \hat{B}_{\omega_b} - \hat{B}_{\omega_a}. \tag{8.5}$$

Temos que a transformada dinversa e Fourier de $\hat{D}$ é dada por

Seja $D(t) = \frac{\omega_b}{\pi} sinc\left(\frac{\omega_b}{\pi} t\right) - \frac{\omega_a}{\pi} sinc\left(\frac{\omega_a}{\pi} t\right) = B_{\omega_b} - B_{\omega_a}.$

Podemos usar um filtro passa-banda para amenizar o fenômeno de Gibbs. Tome $S(t)$ um filtro tal que

$$\hat{S}(\omega) = \begin{cases} 1, & 0 \leq |\omega| \leq \omega_a - d\omega, \\ \frac{\omega_a + d\omega - |\omega|}{2d\omega}, & \omega_a - d\omega \leq |\omega| \leq \omega_a + d\omega, \\ 0, & |\omega| \geq \omega_a + d\omega. \end{cases} \tag{8.6}$$

Exercício 8.2 Mostre que

$$\hat{S}(\omega) = \hat{B}_{\omega_a}(\omega) * \frac{1}{2d\omega}\hat{B}_{\omega_d\omega}(\omega). \tag{8.7}$$

■

Usando o Exercício 8.2, é fácil ver que

$$S(t) = 2\pi B_{\omega_a}(t) \cdot B_{d\omega}(t) \tag{8.8}$$

$$= 2\pi\frac{\omega_a}{\pi}sinc\left(\frac{\omega_a}{\pi}t\right) \cdot \frac{d\omega}{\pi}sinc\left(\frac{d\omega}{\pi}t\right) \cdot \frac{1}{2d\omega} \tag{8.9}$$

$$= \frac{\omega_a}{\pi}sinc\left(\frac{\omega_a}{\pi}t\right)sinc\left(\frac{d\omega}{\pi}t\right). \tag{8.10}$$

8.3 Vazamento Espectral

O vazamento espectral, ou *leakage*, é um fenômeno que ocorre quando tentamos calcular a transformada de Fourier de um sinal que não é periódico. Como vimos no Capítulo 4, a Transformada de Fourier é definida assumindo que o sinal é periódico. Assim, para o sinal discreto, o algoritmo FFT assume que, no intervalo de tempo fornecido, o sinal é periódico. Portanto, se o comprimento de onda do sinal não for um divisor do tamanho total do vetor fornecido, haverá vazamento. Por exemplo, considere o sinal fornecido no BOXX 8.1.1. Se ao invés de uma frequência inteira, colocássemos por exemplo, f1=2.3, veja o que aconteceria:

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

#Vamos criar um sinal no tempo
n = 80   # Numero de amostras
dt= np.pi/40    # Intervalo amostral (em segundos)
t =dt*np.arange(0,n)   # Eixo do tempo
f1 = 2.3  # frequencia (Se 2pi/f1 nao for divisor de dt*n, gera leakage)
f2 = 5.0   # frequencia

#Vamos gerar um sinal periodico
f_sin=1*np.sin(t*f1)
f_cos= 4*np.cos(t*f2)
fx= f_cos + f_sin # sinal

#Calcular a transformada de Fourier do sinal fx
F= dt*fft.fft(fx)   # Coeficientes de Fourier (vezes dt)
w = 2*np.pi*fft.fftfreq(n,dt)  # Eixo de frequencias (dw=1/N/dt)
F= fft.fftshift(F)   # Shift zero para o centro
w = fft.fftshift(w)   # Shift zero para o centro

#Aqui vamos filtrar o sinal
F_filtered = F * (abs(w) < 3.0)
f_t_filtered = dt*n*np.fft.ifft(dt*n*np.fft.ifftshift(F_filtered))
```

```
fig=plt.figure()
ax1=fig.add_subplot(111)
ax1.stem(w,np.absolute(F_filtered))  # Plot espectro de frequencias
ax1.set_ylabel('Absoluto', size = 'x-large')
ax1.set_xlabel('Freq. (rad/s)', size = 'x-large')
plt.show()
```