MAC0317/MAC5920

Introdução ao Processamento de Sinais Digitais

Seção 4.3: Teorema da convolução e filtragem

Revisão: Definição de convolução

Def. 4.2.1: Se $x, y \in \mathbb{C}^N$, definimos a convolução circular de x e y como o vetor $w \in \mathbb{C}^N$ dado por

$$w_n = \sum_{k=0}^{N-1} x_k \, y_{n-k}$$

e denotamos essa operação por $w = x * y$.

Propriedades da convolução

Teorema 4.2.1: Sejam $x, y, w \in \mathbb{C}^N$ e $a, b \in \mathbb{C}$. Então

1. Linearidade: $x * (ay + bw) = a(x * y) + b(x * w)$
2. Comutatividade: $x * y = y * x$
3. Formulação matricial: se $w = x * y$, então $w = M_y x$, onde

$$M_y = \begin{pmatrix} y_0 & y_{N-1} & \cdots & y_1 \\ y_1 & y_0 & \cdots & y_2 \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-1} & y_{N-2} & \cdots & y_0 \end{pmatrix}$$

e além disso $M_x M_y = M_{x*y}$

4. Associatividade: $x * (y * w) = (x * y) * w$
5. Periodicidade: se x e y são tratados por extensão periódica, então

$$w_n = \sum_{k=0}^{N-1} x_k y_{n-k}$$

também está definido $\forall n \in \mathbb{Z}$ e

$$w_n = w_{n \mod N}$$

Teorema da Convolução

Se $x, y, w \in \mathbb{C}^N$ onde $w = x * y$, então

$$W_k = X_k Y_k, \quad k = 0, 1, \ldots, N-1$$

onde $X, Y, W \in \mathbb{C}^N$ são as DFT's de x, y, w, respectivamente.

Prova: Basta usar a definição da DFT:

$$
\begin{aligned}
W_k &= \sum_{n=0}^{N-1} w_n e^{-i2\pi kn/N} \\
&= \sum_{n=0}^{N-1} \left(\sum_{m=0}^{N-1} x_m y_{n-m} \right) e^{-i2\pi kn/N} \\
&= \sum_{m=0}^{N-1} \sum_{r=-m}^{N-1-m} x_m y_r e^{-i2\pi k(r+m)/N} \\
&= \sum_{m=0}^{N-1} \sum_{r=-m}^{N-1-m} x_m y_r e^{-i2\pi kr/N} e^{-i2\pi km/N} \\
&= \left(\sum_{m=0}^{N-1} x_m e^{-i2\pi km/N} \right) \left(\sum_{r=-m}^{N-1-m} y_r e^{-i2\pi kr/N} \right) \\
&= X_k Y_k
\end{aligned}
$$

Ação de um filtro nas componentes do sinal

Considere um filtro arbitrário definido por um vetor de coeficientes $h \in \mathbb{C}^N$ com saída $y = x * h \in \mathbb{C}^N$:

$$x \longrightarrow \boxed{h} \longrightarrow y = x * h$$

$$X \longrightarrow \boxed{H} \longrightarrow Y = XH$$

Quando $x = E_{N,k}$, temos $X = (0, 0, \ldots, 0, N, 0, \ldots, 0)'$. Pelo teorema da convolução

$$Y = XH = (0, 0, \ldots, 0, NH_k, 0, \ldots, 0)'$$

Esta saída no domínio do tempo é

$$y = \frac{1}{N} \sum_{r=0}^{N-1} Y_r E_{N,r} = H_k E_{N,k}$$

Mas $y = x * h = M_h x$, logo

$$M_h E_{N,k} = H_k E_{N,k}$$

Esta expressão mostra que os vetores $E_{N,k}$ são **autovetores** da matriz M_h associada ao filtro, com correspondentes autovalores dados por H_k.

Cada componente $E_{N,k}$ de um sinal arbitrário é processada pelo filtro H produzindo a saída

$$
\begin{aligned}
y &= H_k E_{N,k} \\
&= |H_k| e^{i(\measuredangle H_k)} \begin{pmatrix} \vdots \\ e^{i2\pi kn/N} \\ \vdots \end{pmatrix} \\
&= |H_k| \begin{pmatrix} \vdots \\ e^{i(2\pi kn/N + \measuredangle H_k)} \\ \vdots \end{pmatrix} \\
&= |H_k| \begin{pmatrix} \vdots \\ e^{i2\pi k\left(n + \frac{N\measuredangle H_k}{2\pi k}\right)/N} \\ \vdots \end{pmatrix}
\end{aligned}
$$

ou seja, a magnitude da componente é multiplicada por $|H_k|$, e sua fase é adicionada de $\measuredangle H_k$ radianos em cada componente, o que equivale a um *shift* temporal de $\frac{N\measuredangle H_k}{2\pi k}$ amostras em cada componente.

No caso particular em que $\measuredangle H_k = \alpha k$ (resposta linear em fase), teremos

$$y = |H_k| \begin{pmatrix} \vdots \\ e^{i2\pi k\left(n+\frac{N\alpha}{2\pi}\right)/N} \\ \vdots \end{pmatrix}$$

Como $\frac{N\alpha}{2\pi}$ não depende da frequência, um filtro com resposta em fase linear preservará as relações de fase entre componentes senoidais diversas em um sinal de entrada geral. Assim, todas as componentes senoidais sofrem o mesmo *shift* de

$$\frac{N\alpha}{2\pi} \text{ amostras,}$$

que é chamado de **atraso de grupo**.

Exemplo 4.3:

O filtro da média

$$w_n = \frac{1}{2}x_n + \frac{1}{2}x_{n-1}$$

pode ser escrito como

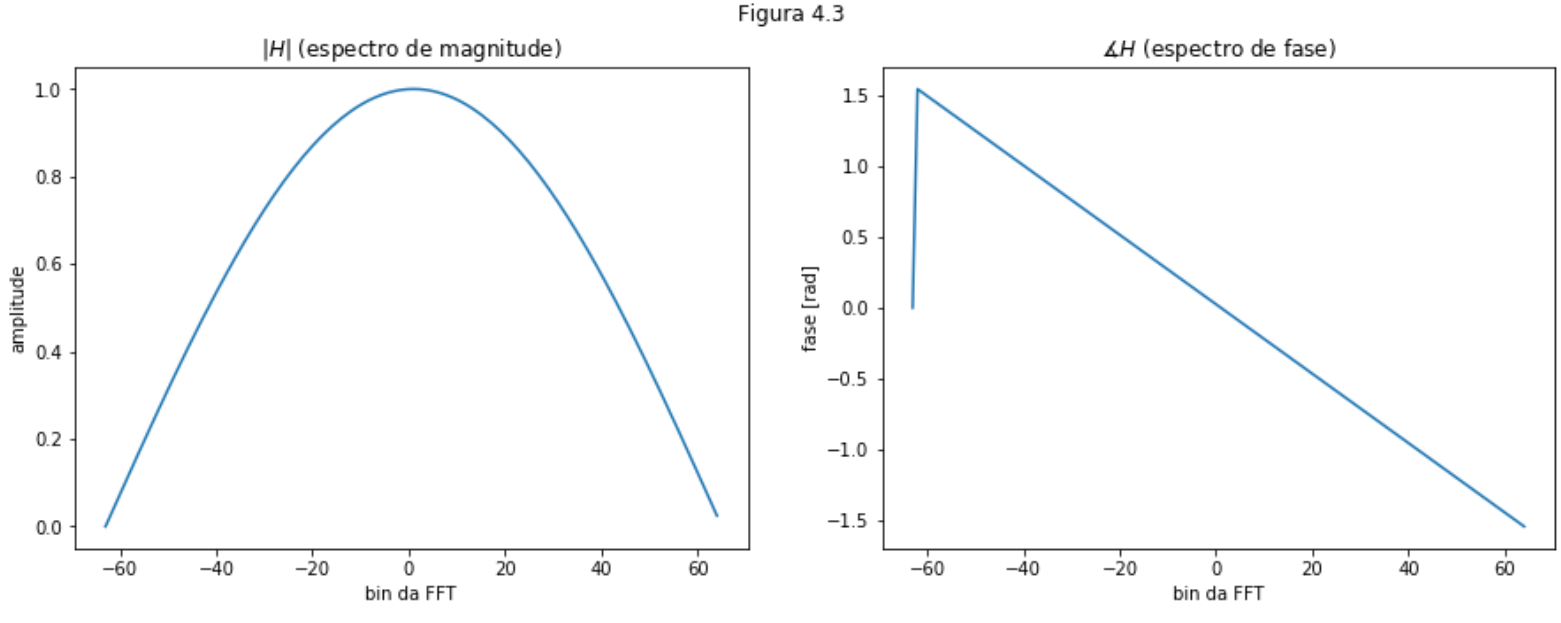
$$w = x * h,$$

onde os coeficientes do filtro são $h_0 = \frac{1}{2}$ e $h_1 = \frac{1}{2}$, de tal forma que

$$w_n = \sum_{k=0}^{N-1} x_k\, h_{n-k} = \sum_{k=0}^{N-1} h_k\, x_{n-k} = h_0\, x_n + h_1\, x_{n-1} = \frac{1}{2}x_n + \frac{1}{2}x_{n-1}.$$

In [3]:

```
N = 128;h = np.zeros(N);h[0] = h[1] = 0.5;H = np.fft.fft(h)
fig, ax = plt.subplots(1,2, figsize=(15,5))
ax[0].plot(np.arange(-N//2+1,N//2+1),np.fft.fftshift(np.abs(H)))
ax[0].set_title('$|H|$ (espectro de magnitude)');ax[0].set_xlabel('bin da FFT');
ax[0].set_ylabel('amplitude')
ax[1].plot(np.arange(-N//2+1,N//2+1),np.fft.fftshift(np.angle(H)))
ax[1].set_title('$\measuredangle H$ (espectro de fase)');ax[1].set_xlabel('bin d
a FFT');ax[1].set_ylabel('fase [rad]')
fig.suptitle("Figura 4.3")
plt.show()
```



Havíamos calculado a resposta em frequência do filtro da média como

$$H_k = \cos(2\pi k/N)e^{-i\pi k/N}$$

$$\left(\text{ou equivalentemente } M_h E_{N,k} = \cos(2\pi k/N)e^{-i\pi k/N} E_{N,k}\right)$$

logo $\measuredangle H_k = -\pi k/N$ (resposta linear em k), com atraso de grupo correspondente a

$$\frac{N\measuredangle H_k}{2\pi} = \frac{-N\pi/N}{2\pi} = -\frac{1}{2} \text{ amostra.}$$

Isso é intuitivo ao considerarmos que o sinal de saída é uma média entre o sinal original e o sinal todo atrasado em 1 amostra:

$$y_n = \frac{1}{2}x_n + \frac{1}{2}x_{n-1} \implies y = \frac{1}{2}\begin{pmatrix} (x_0, x_1, \dots, x_{N-1}) \\ + \\ (x_{-1}, x_0, \dots, x_{N-2}) \end{pmatrix}$$

$$= \left(x_{-\frac{1}{2}}, x_{\frac{1}{2}}, \dots, x_{N-1-\frac{1}{2}}\right)$$

onde os índices fracionários representam a interpolação linear do sinal.

Resposta ao impulso

Considere o vetor $h \in \mathbb{C}^N$ associado à equação $y = x * h$, e seja a entrada $x = \delta$ onde

$$\delta = \begin{cases} 1 & \text{se } n = 0 \\ 0 & \text{c.c.} \end{cases}$$

Esse sinal δ é chamado de **impulso unitário**. A resposta do filtro a esse impulso será $y = \delta * h$, onde

$$\begin{aligned} y_n &= \sum_{k=0}^{N-1} \delta_k h_{n-k} \\ &= h_n \end{aligned}$$

Assim, vemos que δ é o **elemento neutro** da operação de convolução, e h é denominado **resposta ao impulso** associada ao filtro.

Resposta em Frequência

Denominamos de **resposta em frequência** ao vetor $H = DFT(h) \in \mathbb{C}^N$, lembrando que cada coeficiente H_k modifica a componente de frequência k de um sinal da entrada x de acordo com o teorema da convolução

$$y = x * h \implies Y_k = H_k X_k.$$

Denominamos de **resposta em magnitude** ao vetor $|H| \in \mathbb{R}^N$, e de **resposta em fase** ao vetor $\measuredangle H \in \mathbb{R}^N$, levando em consideração que

$$Y_k = H_k X_k = |H_k| e^{i \measuredangle H_k} |X_k| e^{i \measuredangle X_k} = |H_k||X_k| e^{i(\measuredangle H_k + \measuredangle X_k)},$$

ou seja, que as magnitudes das componentes da entrada são multiplicadas por $|H_k|$ e as fases são acrescidas de $\measuredangle H_k$.

Desenho de filtros

Em princípio, podemos definir arbitrariamente qualquer resposta em frequência $H \in \mathbb{C}^N$ desejada, obtendo a resposta ao impulso pela expressão $h = IDFT(H)$.
Entretanto

1. Se H é um vetor qualquer, então h pode não ser real,
   e com isso $x * h$ também pode não ser real.

   Lembrando que $h \in \mathbb{R}^N \Leftrightarrow H_k = \overline{H_{-k}}$
   uma solução é definir H_k arbitrariamente para $k = 0, 1, \ldots, \frac{N}{2}$
   e depois copiar $H_{N-k} = H_{-k} = \overline{H_k},\ k = \frac{N}{2} + 1, \ldots, N - 1$.
2. Normalmente o vetor $h = IDFT(H)$ pode conter muitos coeficientes $\neq 0$,
   tornando o filtro computacionalmente caro.

Observação: custo da convolução

1. $y_n = \sum_{k=0}^{N-1} x_k h_{n-k}, n = 0, \dots, N-1$ (custo $\mathcal{O}(N^2)$)
2. $y = IDFT(Y)$ onde $Y_k = X_k H_k,\ k = 0, 1, \dots, N-1$ (custo $\mathcal{O}(N \log N)$)
3. se $\#h = \#\{h_n \neq 0\}$ é pequeno $(\mathcal{O}(1))$, então o cálculo direto

$$y_n = \sum_{k=0}^{N-1} x_k h_{n-k} = \sum_{k:h_k \neq 0} h_k x_{n-k}$$

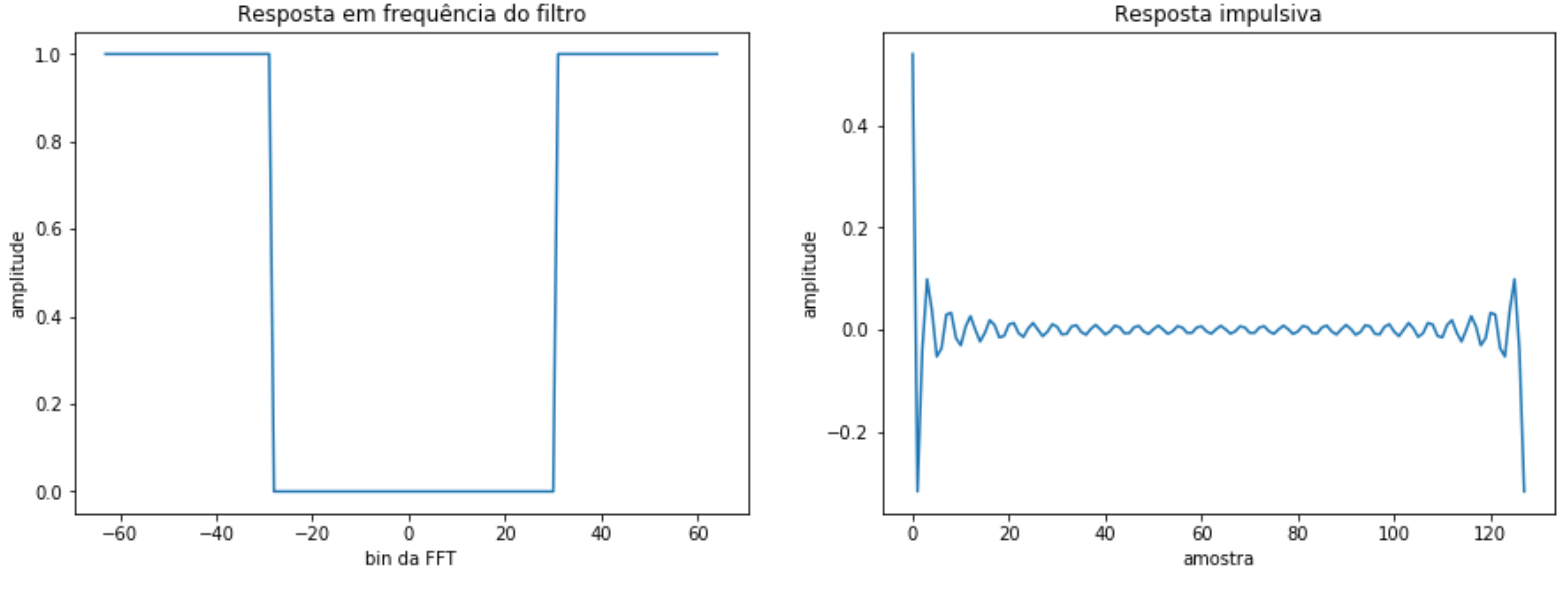
tem custo $\mathcal{O}(N)$.

Uma "solução" para deixar um filtro mais eficiente é zerar componentes de h que satisfaçam $|h_m| < \epsilon$ para algum $\epsilon > 0$ escolhido, como no algoritmo de compressão estudado no capítulo 3. O resultado será um filtro aproximado, cuja distorção pode ser medida a partir das respostas em frequência antes e depois da modificação.

Exemplo 4.4: filtro passa-alta projetado a partir de uma resposta em frequência ideal.

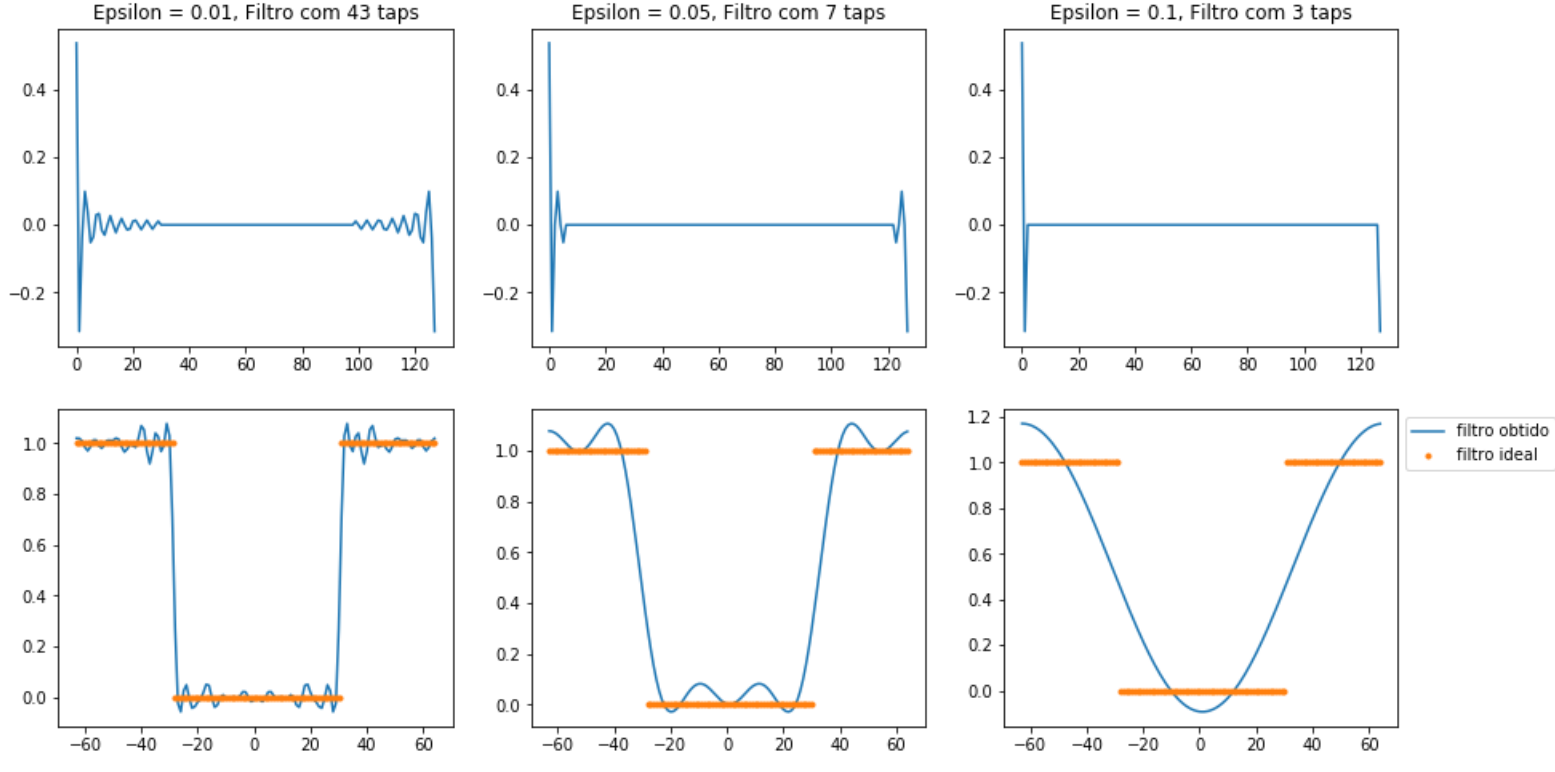
In [6]:
```
N = 128;H = np.concatenate((np.zeros(30),np.ones(69),np.zeros(29)));h = np.real
(np.fft.ifft(H))
fig, ax = plt.subplots(1,2, figsize=(15,5))
ax[0].plot(np.arange(-N//2+1,N//2+1),np.fft.fftshift(H));ax[0].set_title('Respo
sta em frequência do filtro');ax[0].set_xlabel('bin da FFT');ax[0].set_ylabel('a
mplitude')
ax[1].plot(h);ax[1].set_title('Resposta impulsiva');ax[1].set_xlabel('amostra'
);ax[1].set_ylabel('amplitude')
fig.suptitle("Figura 4.4");plt.show()
```

Figura 4.4



Versões computacionalmente mais ”baratas” (com menos taps) do mesmo filtro passa-alta.

In [16]:
```
fig, ax = plt.subplots(2,3, figsize=(15,8))
for k in range(len(EPSVALS)): plota_filtro_com_menos_taps(EPSVALS[k],h,ax,k)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1));plt.show()
```



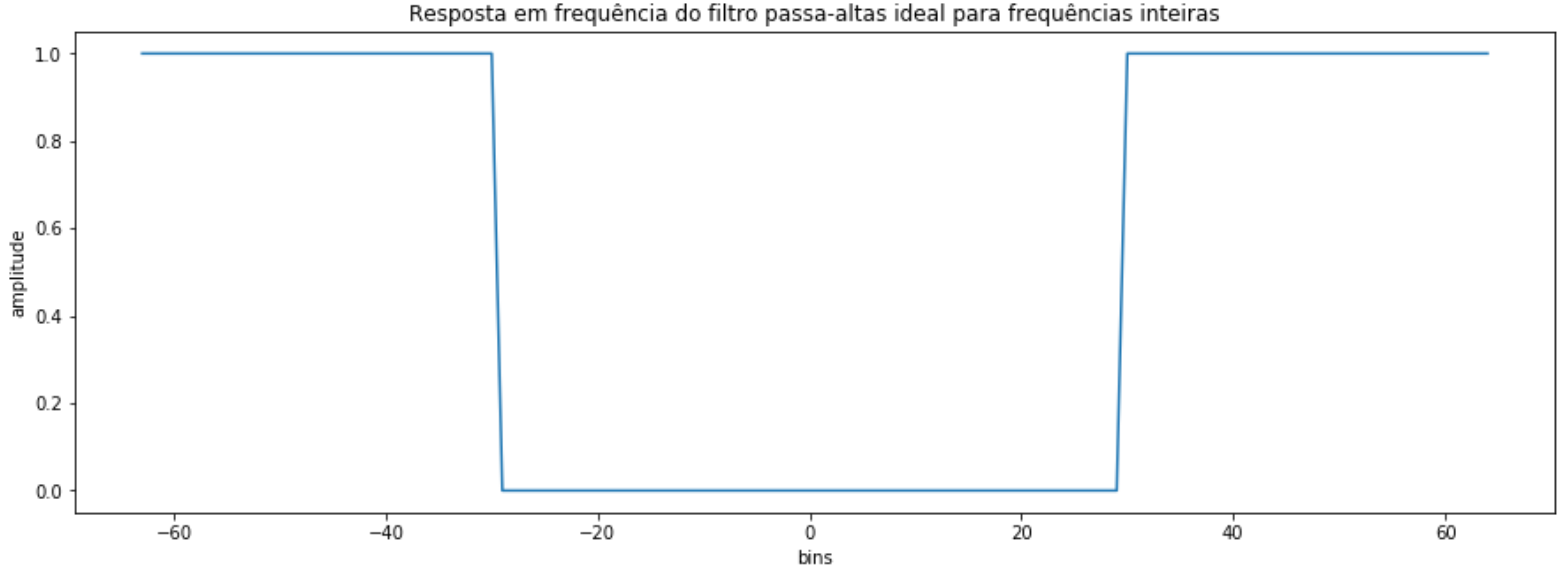
Colocando em perspectiva a "perfeição" da resposta em frequência do filtro original (com banda de corte nula).

Este exemplo (que não está no livro) se contrapõe à idéia de que o filtro desenhado a partir da resposta em frequência H tenha exatamente o comportamento de filtro ideal que o gráfico de H poderia sugerir.

Para este exemplo, construiremos "na mão" a resposta em magnitude do filtro H para funções "básicas" do tipo $\cos(2\pi f \frac{n}{N})$, inicialmente para frequências do tipo $f = 0, 1, \ldots, N-1$, e posteriormente para outras frequências intermediárias.

In [26]:
```
FREQS = np.arange(-N//2+1,N//2+1,1)
R = [ 0 ] * len(FREQS)
t = np.arange(N)
for k in range(N):
    # cria função básica de frequência k
    x = np.cos(2*np.pi*FREQS[k]*t/N)
    # filtra por h
    y = np.real(np.fft.ifft(np.fft.fft(x)*H))
    # constrói gráfico de magnitude da resposta
    # (fator de escala da saída do filtro)
    R[k] = np.linalg.norm(y)/np.linalg.norm(x)
```

In [27]:
```
# Mostra o gráfico da resposta de magnitude.
plt.figure(figsize=(15,5));plt.title("Resposta em frequência do filtro passa-alt
as ideal para frequências inteiras")
plt.plot(FREQS,R);plt.xlabel("bins");plt.ylabel('amplitude');plt.show()
```



Repete a mesma construção com frequências não-inteiras

Observe que a única diferença do código abaixo é que agora as frequências são f=0,0.1,0.2,...,N-1

In [28]:
```
FREQS = np.arange(-N//2+1,N//2+1,0.1)
R = [ 0 ] * len(FREQS)
t = np.arange(N)
for k in range(len(FREQS)):
    # cria função básica de frequência k
    x = np.cos(2*np.pi*FREQS[k]*t/N)
    # filtra por h
    y = np.real(np.fft.ifft(np.fft.fft(x)*H))
    # constrói gráfico de magnitude da resposta
    # (fator de escala da saída do filtro)
    R[k] = np.linalg.norm(y)/np.linalg.norm(x)
```

In [29]:

```
# Mostra o gráfico da resposta de magnitude.
plt.figure(figsize=(15,5));plt.title("Resposta em frequência do filtro passa-alt
as ideal para frequências racionais")
plt.plot(FREQS,R);plt.xlabel("bins");plt.ylabel('amplitude');plt.show()
```