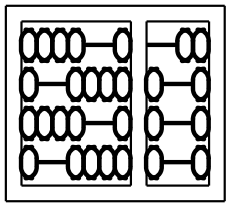


Universidade Estadual de Campinas - Unicamp
Instituto de Computação - IC

Convolução FFT

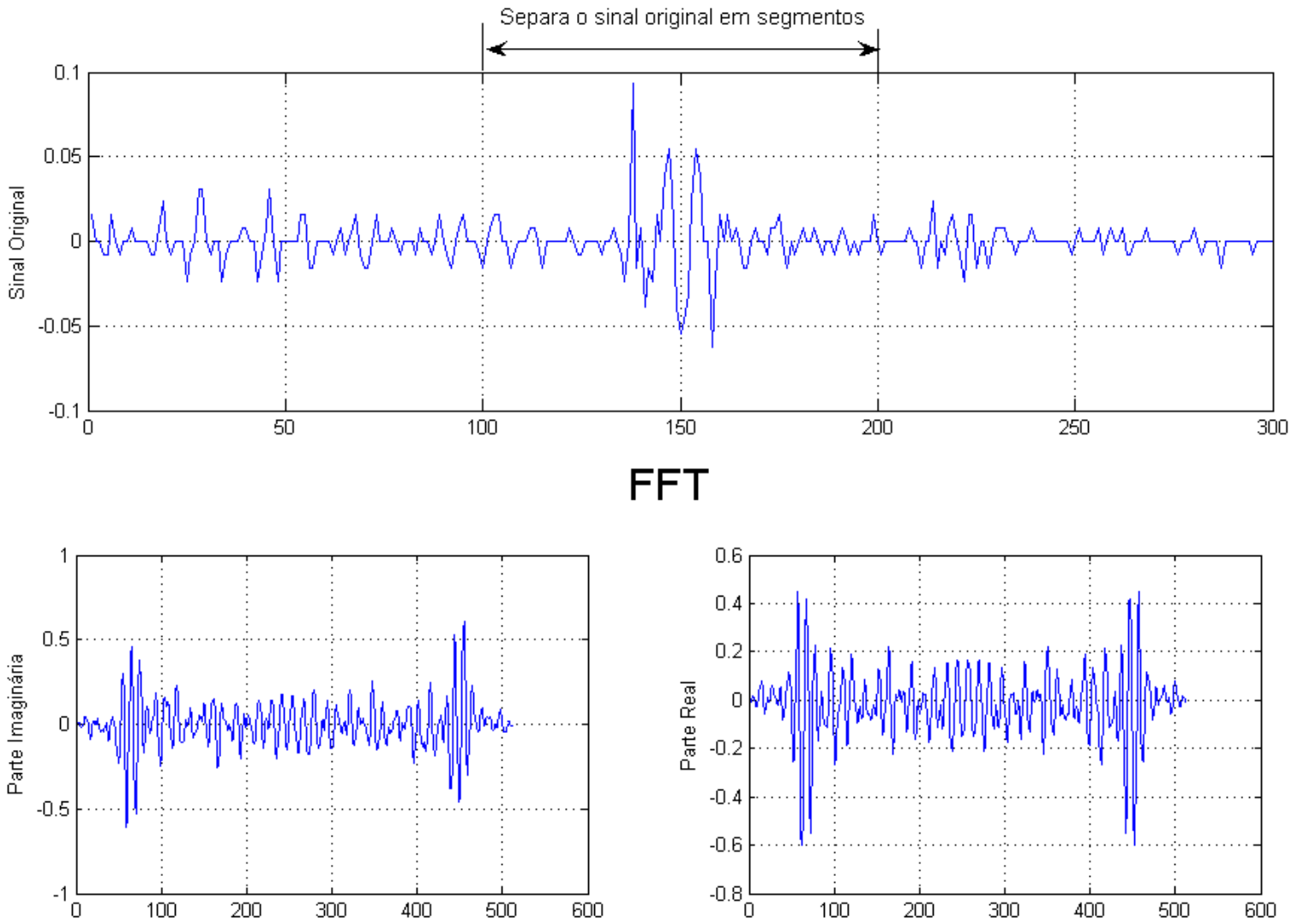
Leyza Baldo Dorini

Campinas, 13 de Setembro de 2006

Análise no domínio da frequência

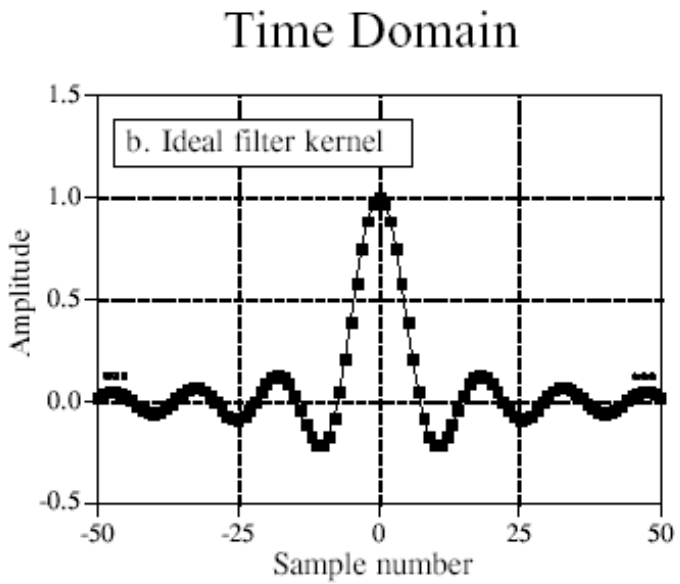
- A convolução utilizando FFT baseia-se no princípio que multiplicação no domínio da frequência corresponde à convolução no domínio do tempo.
- O sinal de entrada é transformado para o domínio da frequência utilizando a FFT, multiplicado pela resposta de frequência do filtro, e então transformada de volta para o domínio do tempo.

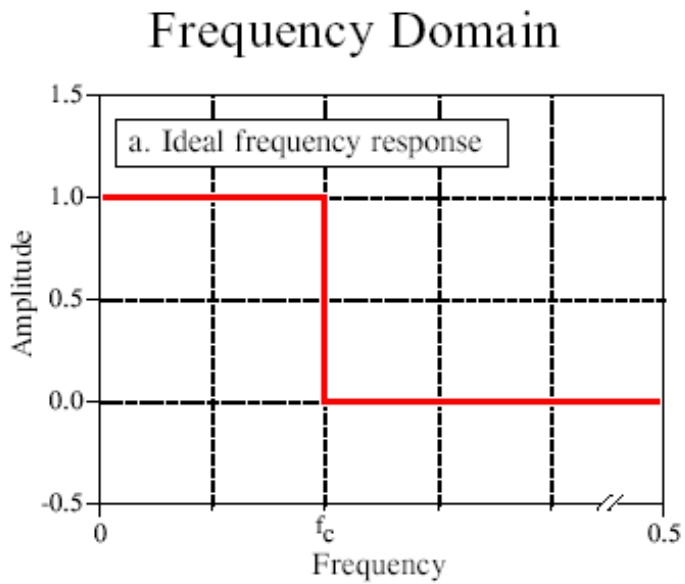
FFT Convolution



Filtro passa-baixa

Considere o filtro passa-baixa ideal ("brick-wall filter") com uma frequência de corte de ω_0 rad/s. Este filtro tem magnitude 1 em todas as frequências maiores que ω_0 e magnitude 0 em frequências entre ω_0 e π.





Filtro passa-baixa

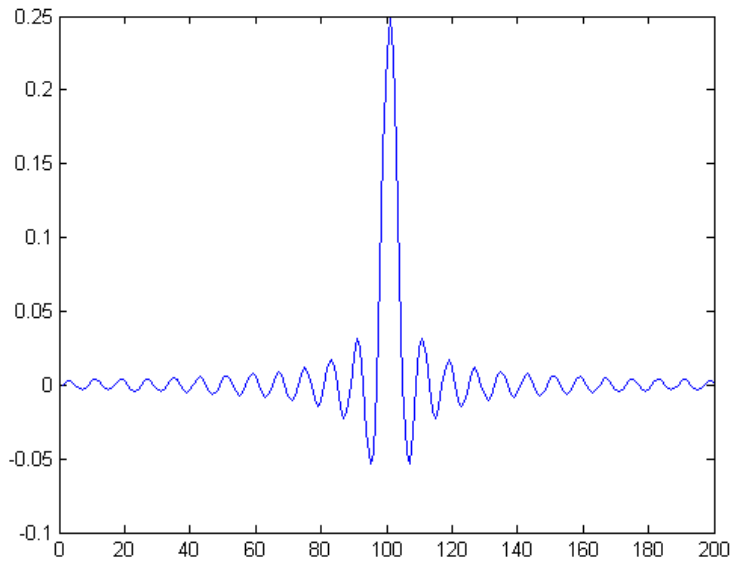
Sua resposta de impulso é dada por:

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0} e^{j\omega n}d\omega = \frac{\omega_0}{\pi}sinc\left(\frac{\omega_0}{\pi}n\right) \quad (1)$$

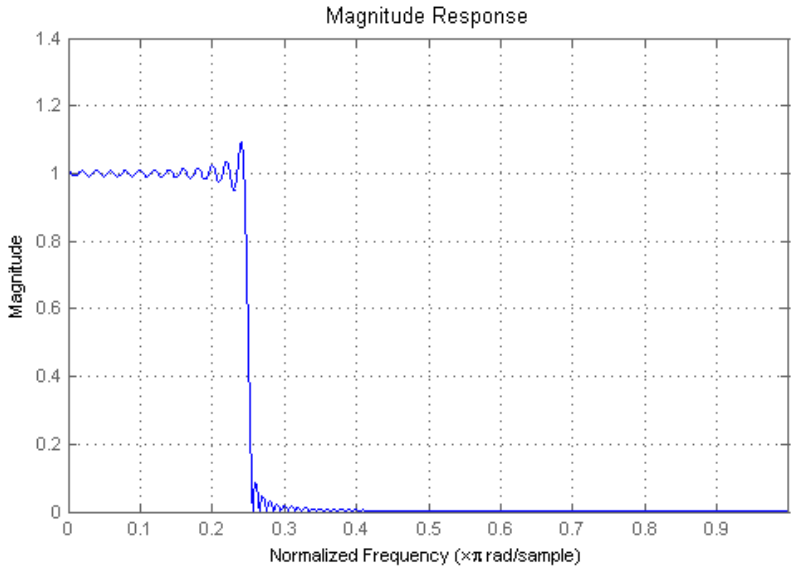
A resposta de impulso é simétrica em relação à $n = 0$, o que garante que o filtro possui uma fase linear.

Em Matlab...

```
sz = 200;
n = [0:sz];
omega = pi/4;
h = (omega/pi)*sinc((n-sz/2)*omega/pi);
```



(a) Tempo



(b) Frequência

Janelas

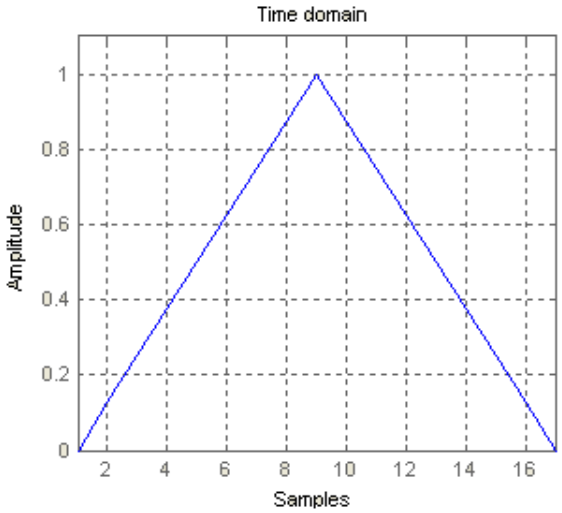
- Tal filtro não é implementável, sendo que para a resposta ideal, n deveria variar de $-\infty$ a ∞, levando o filtro a ser não-causal.
- Para criar uma resposta de impulso finita, é feito um truncamento utilizando uma janela. Além disso, o resultado é deslocando para produzir um filtro causal.

Janelas

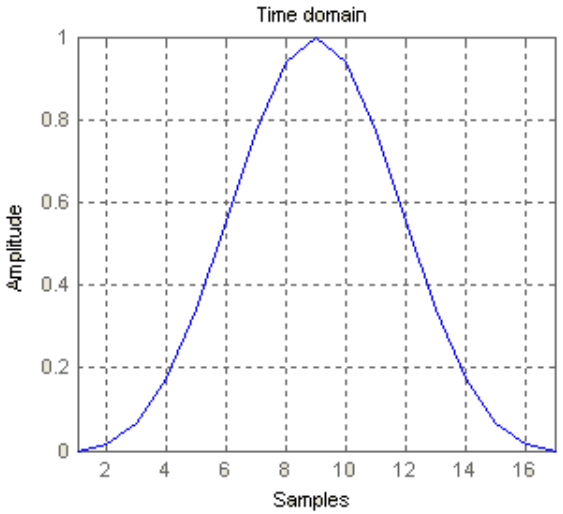
- São diversas as janelas existentes: Bartlett, Blackman, Hamming, Kaiser, etc.
- Em Matlab: `w = window(@gausswin, 64, alpha)` ou `w = gausswin(64, alpha)` irá retornar uma janela Gaussiana de tamanho 64 com desvio padrão igual a $1/alpha$

Janelas

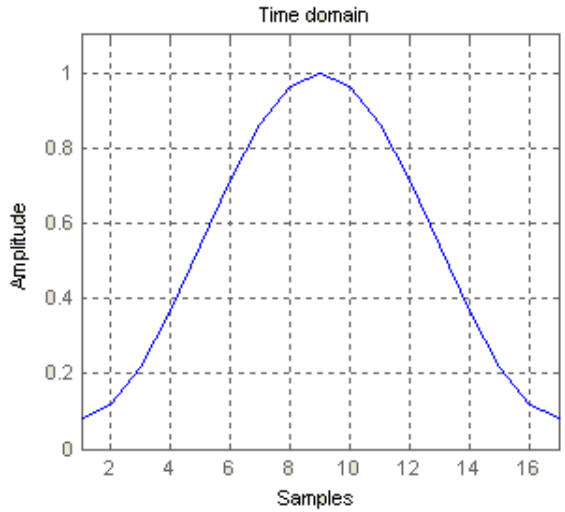
As janelas utilizadas neste trabalho foram:



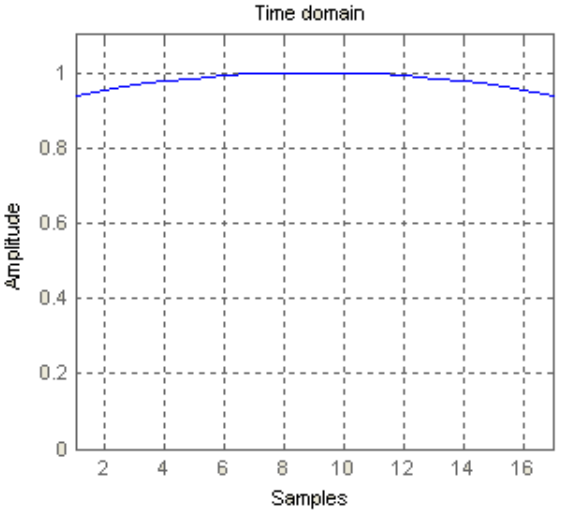
Bartlett



Blackman



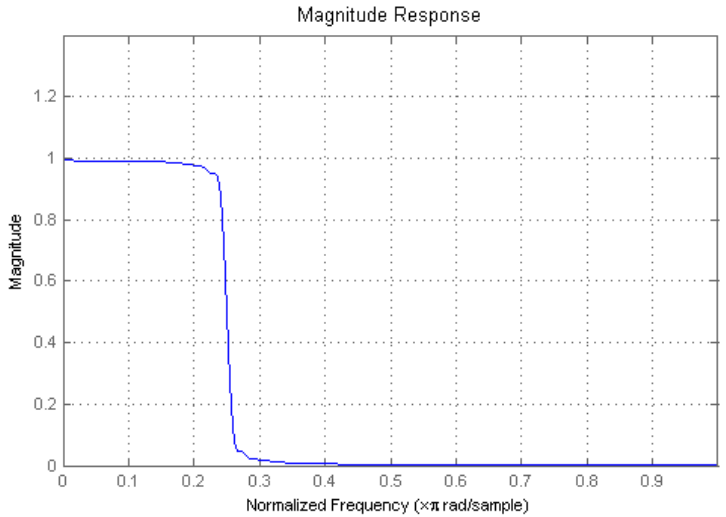
Hamming



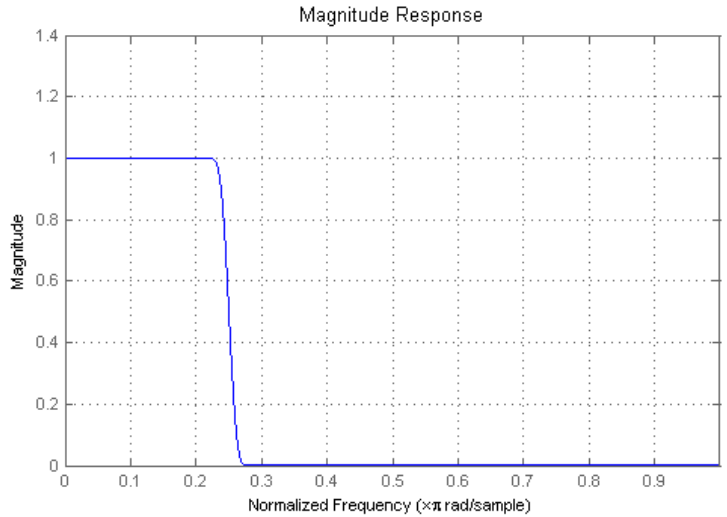
Kaiser

Em Matlab...

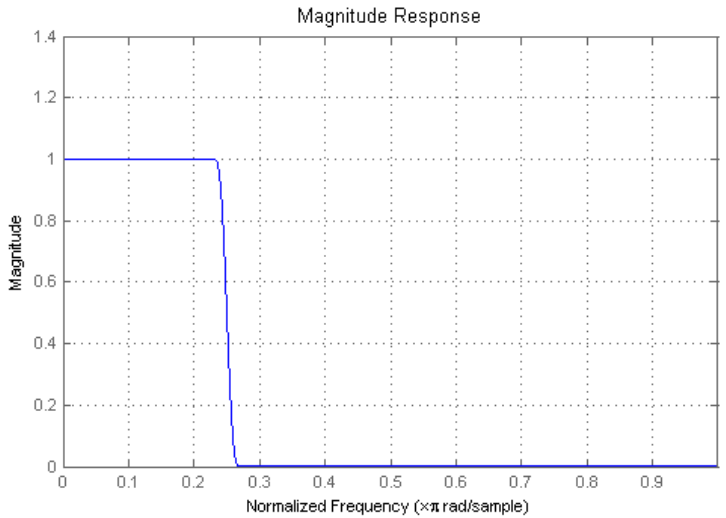
```
sz = 200;
n = [0:sz];
omega = pi/4;
h = (omega/pi)*sinc((n-sz/2)*omega/pi);
hw = h.*hamming(sz+1)';
```



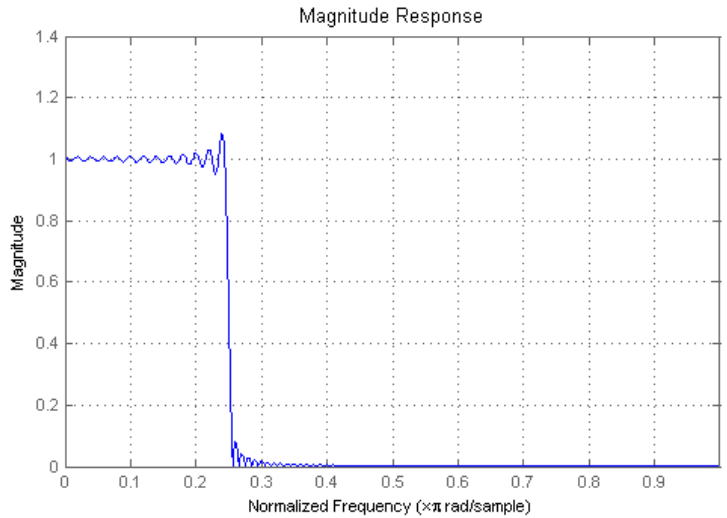
(a) Bartlett



(b) Blackman



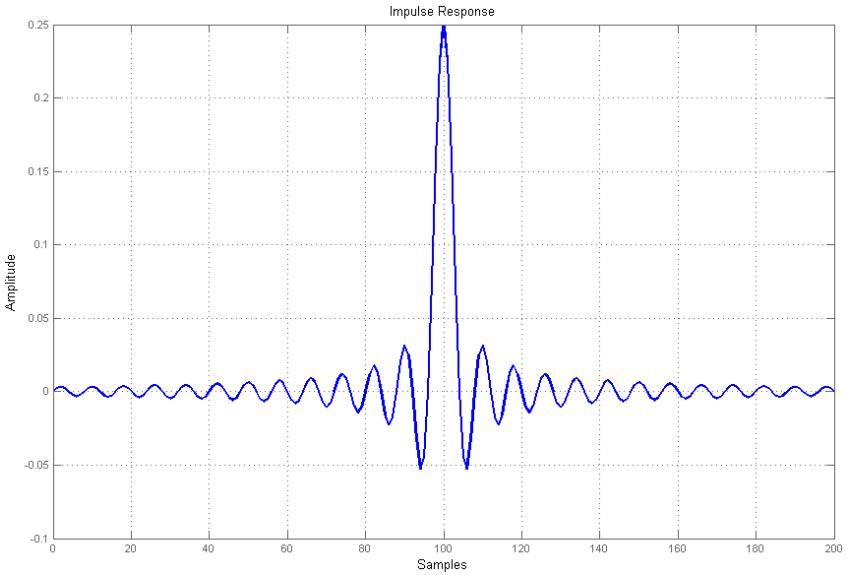
(a) Hamming



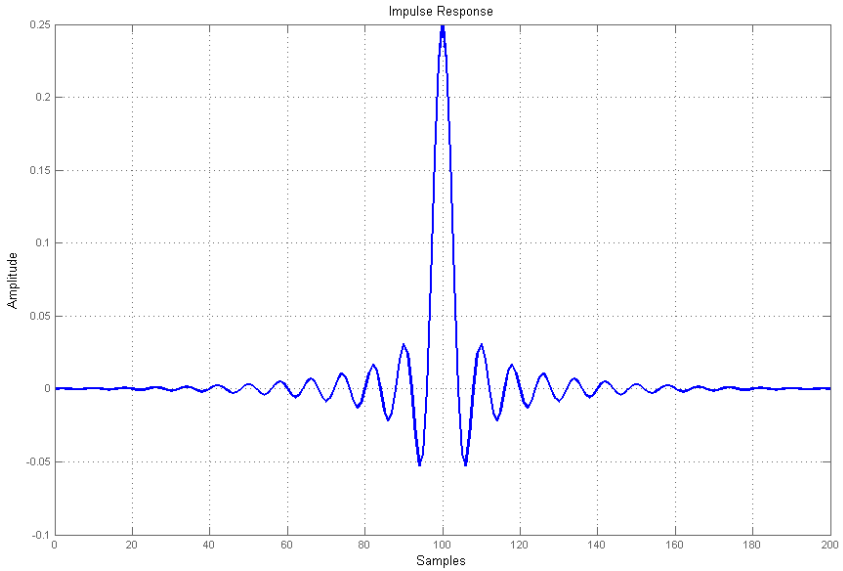
(b) Kaiser

Figura: Influência da ordem do filtro. `h = fir1(n,[0.4`

Exemplos (i)

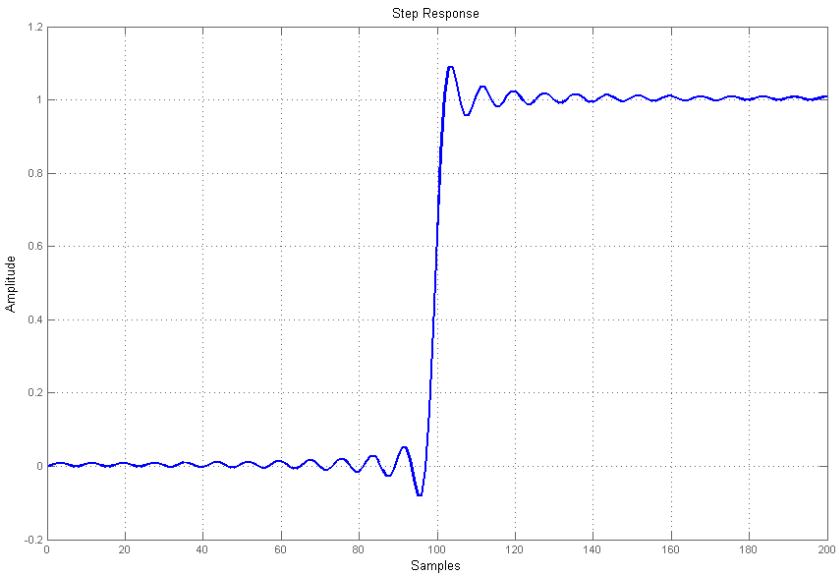


(a) Sem janela

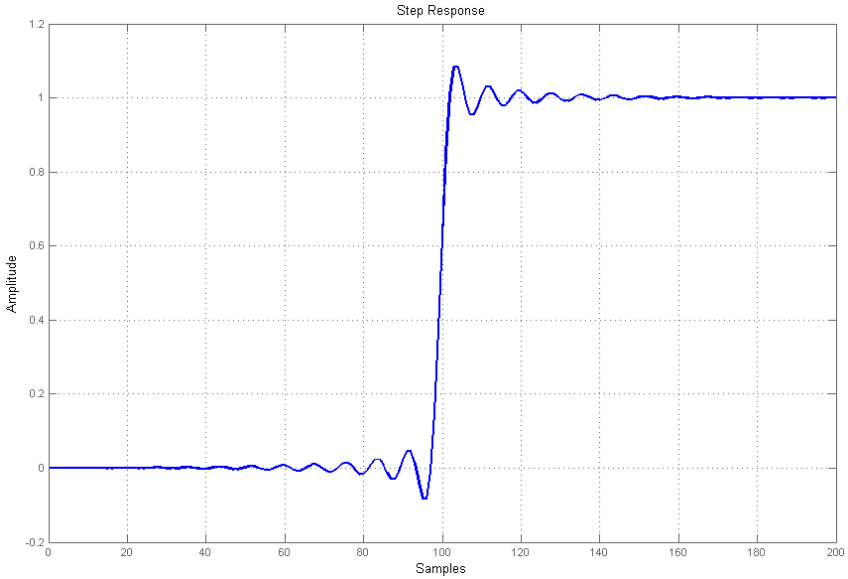


(b) Com Janela

Exemplos (ii)

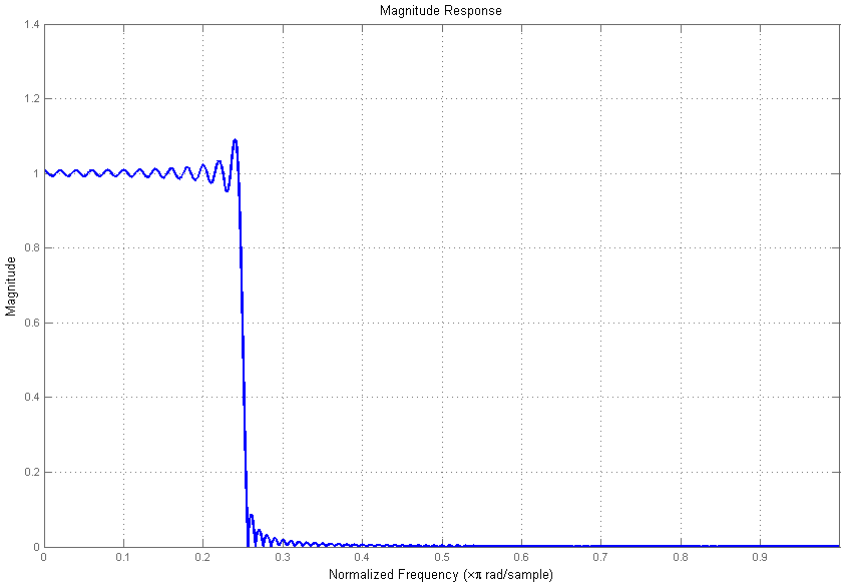


(a) Sem janela

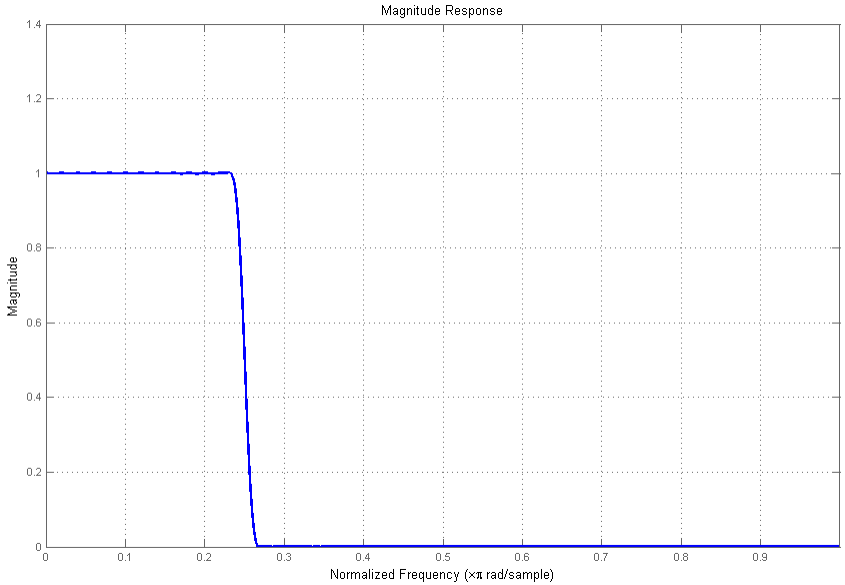


(b) Com Janela

Exemplos (iii)

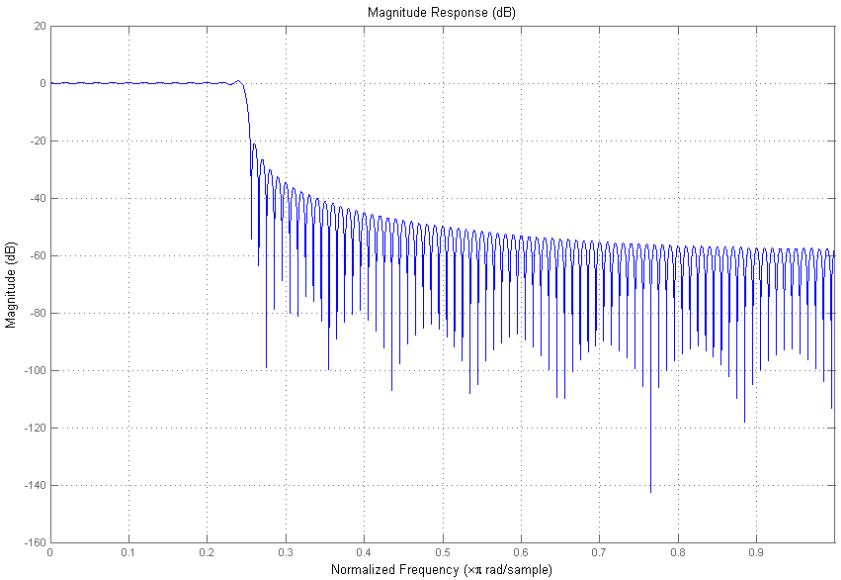


(a) Sem janela

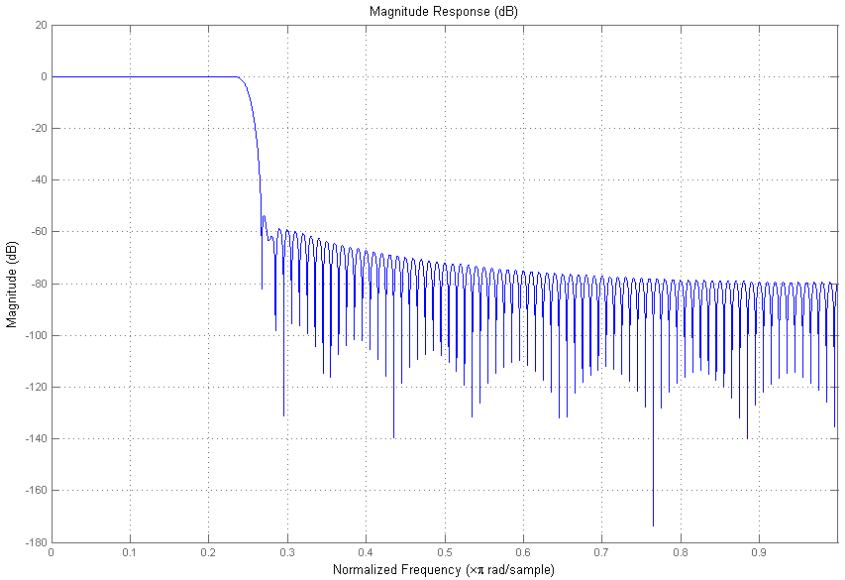


(b) Com Janela

Exemplos (iv)



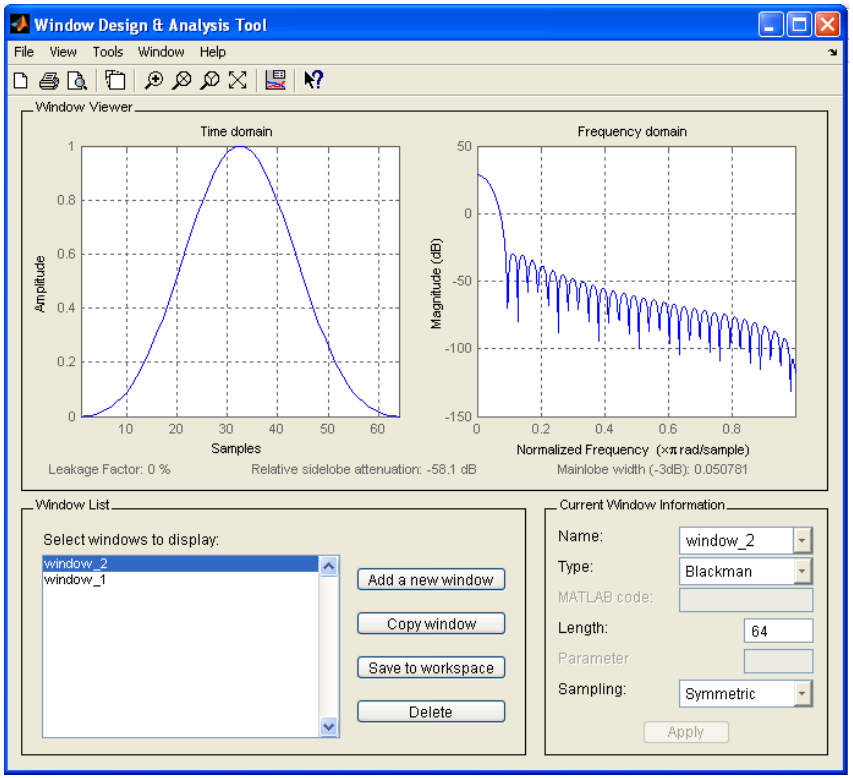
(a) Sem janela



(b) Com Janela

Detalhe...

O comando `wintool` possibilita analisar propriedades das diversas janelas, bem como compará-las entre si.



Além disso, é possível também analisar separadamente uma janela específica com o comando `wvtool(windowname(n))`

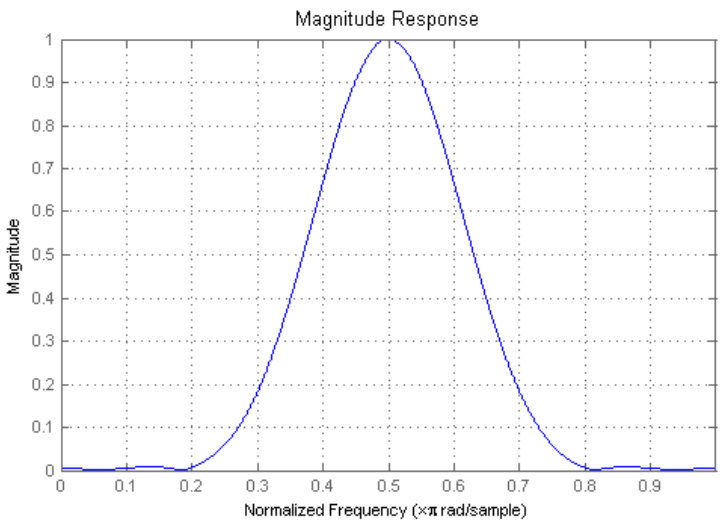
Projeto de filtros

Foram utilizados filtros FIR (Finite Impulse Response). Algumas vantagens:

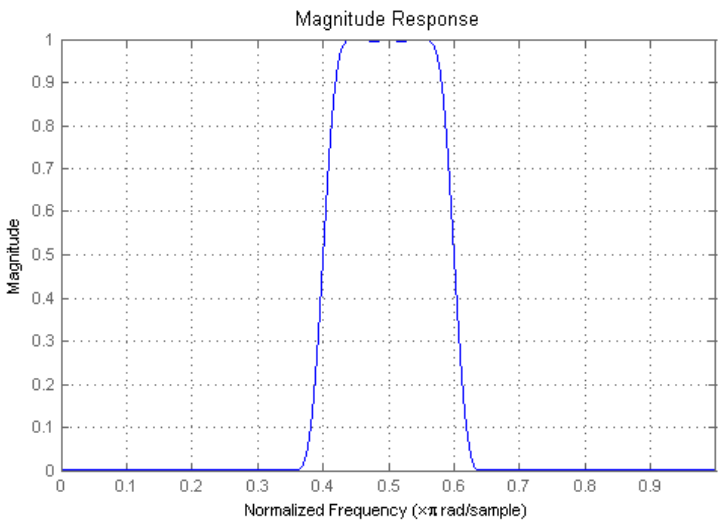
- podem possuir fase linear exata (simétrico em relação ao midpoint)
- são estáveis
- métodos de projeto geralmente são lineares
- podem ser implementados eficientemente em hardware

A principal desvantagem em relação aos filtros IIR (Infinite Impulse Response), é que os filtros FIR exigem uma ordem maior para alcançar um dado nível de desempenho.

Exemplos (i)



(a) $n = 16$



(b) $n = 96$

Figura: Influência da ordem do filtro. `h = fir1(n,[0.4 0.6],'bandpass')`

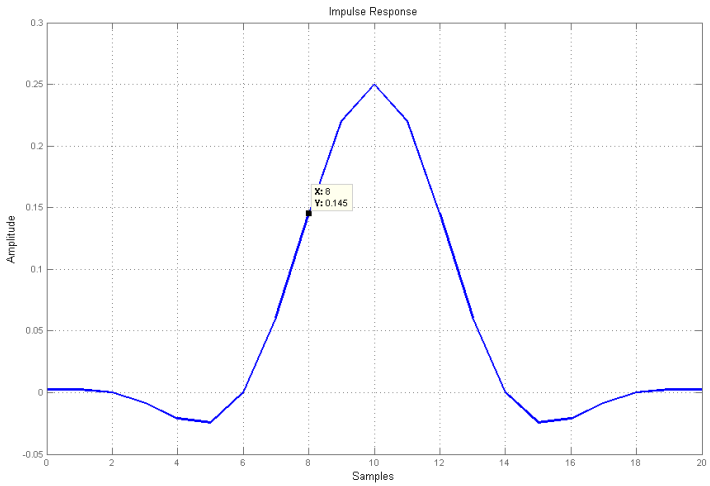
No Matlab

Foi utilizada a função `fir1`:

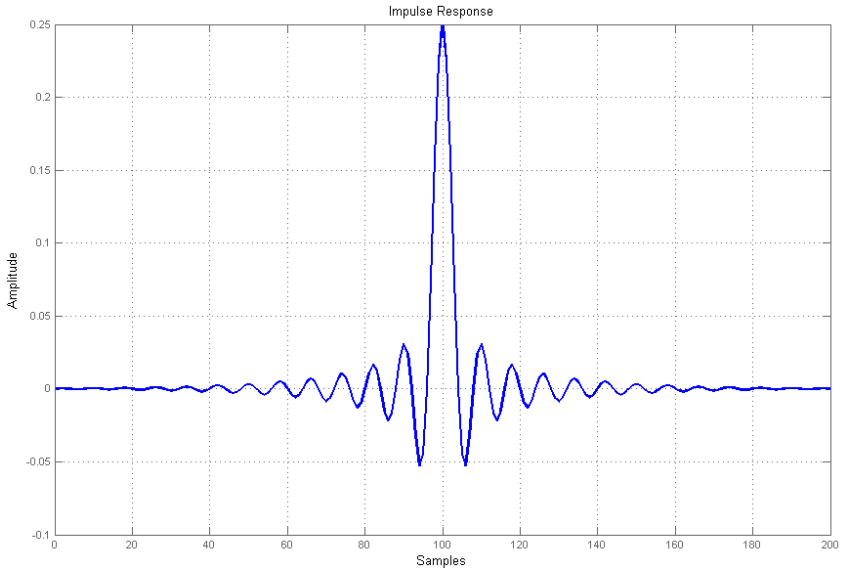
- h1 = fir1(n, fc, 'low');
- h2 = fir1(n, fc, 'high');
- h3 = fir1(n, [fci fcf], 'bandpass');
- h4 = fir1(n, [fci fcf], 'stop';

A resposta de impulso é simétrica em relação à $n = 0$, o que garante que o filtro possui uma fase linear.

Exemplos (i)

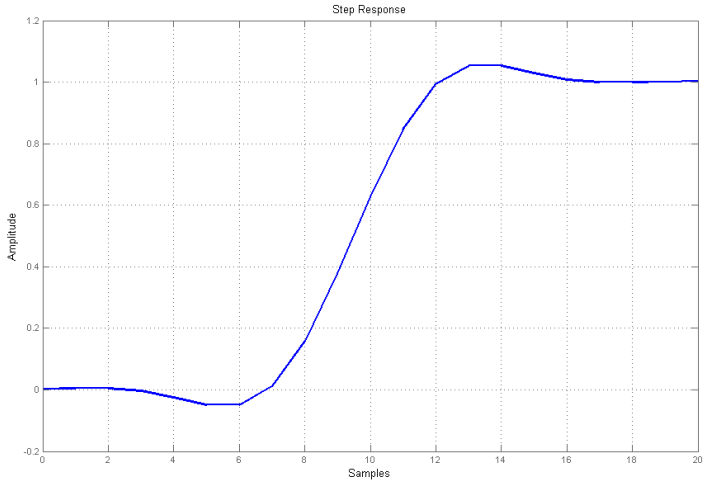


(a) $n = 20$

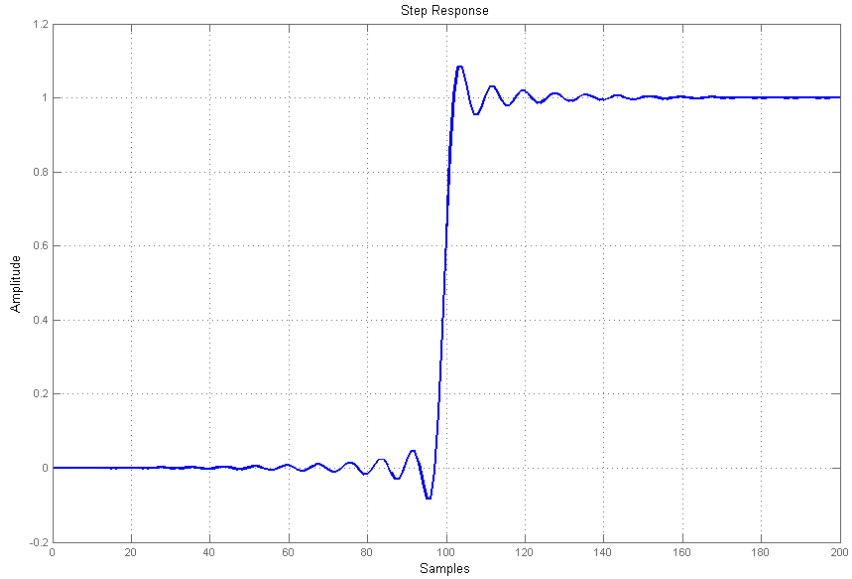


(b) $n = 200$

Exemplos (ii)

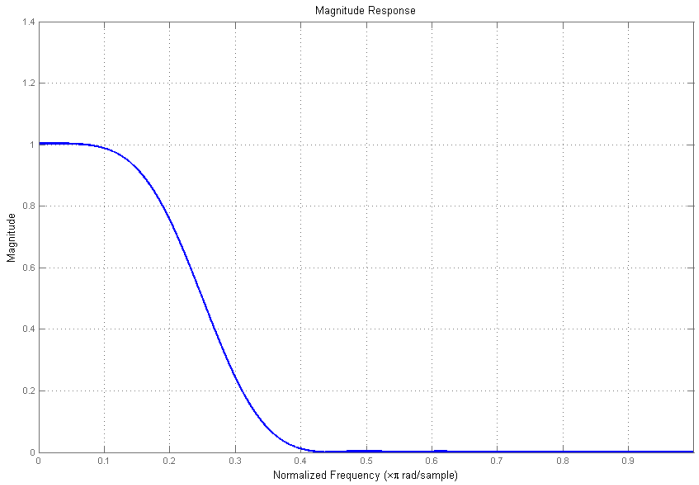


(a) $n = 20$

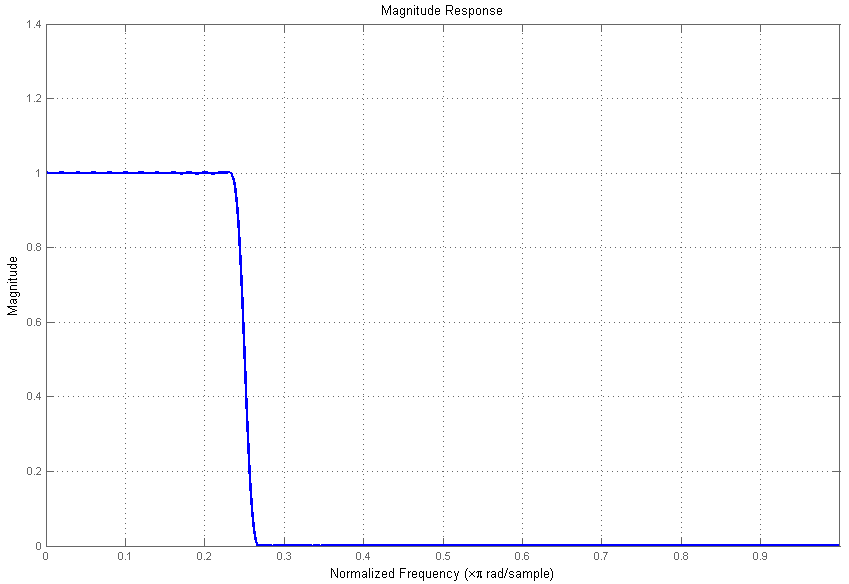


(b) $n = 200$

Exemplos (iii)

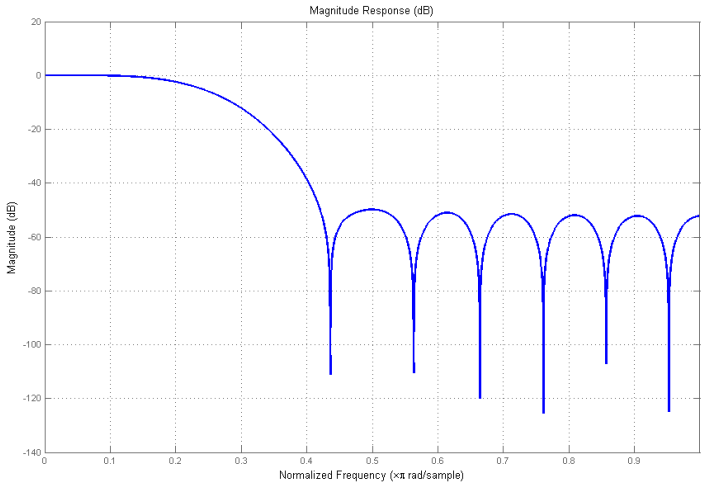


(a) $n = 20$

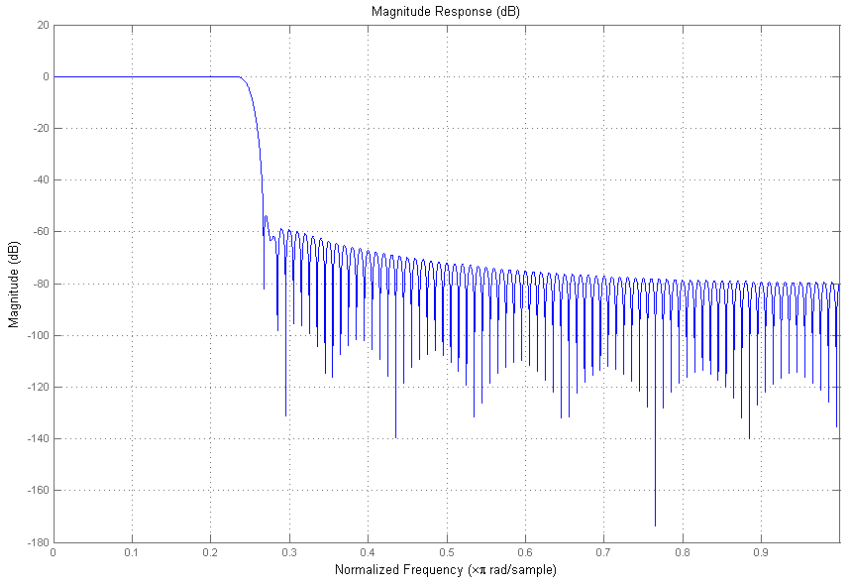


(b) $n = 200$

Exemplos (iv)



(a) $n = 20$



(b) $n = 200$

Métodos de avaliação

Os seguintes testes foram realizados:

- tempos de execução
  - convolução usando FFT (do Matlab)
  - convolução "padrão"
  - influência do tamanho do filtro $\times$ tamanho do sinal
- efeitos da escolha do tamanho de um segmento
- outras comparações

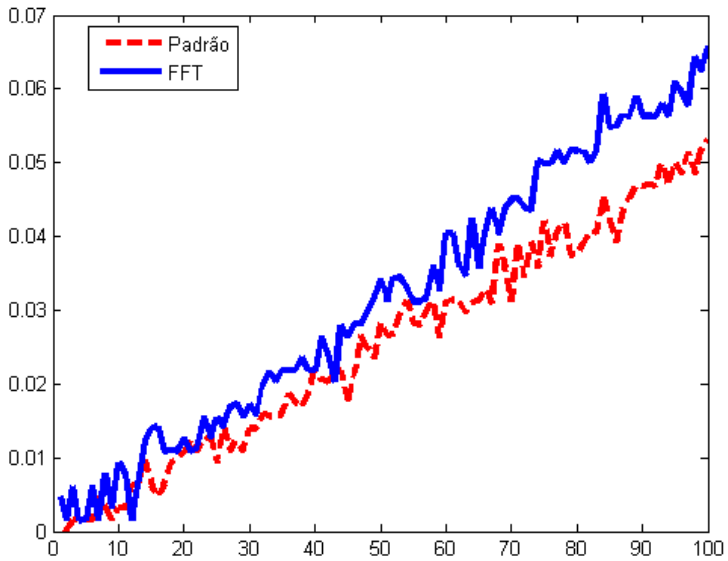
Convolução - Observação

Ao fazer a convolução uma consideração é importante: a ordem em que ela é feita. PS. O comando `conv` do Matlab leva isso em consideração. Dados para um filtro f de tamanho 256.

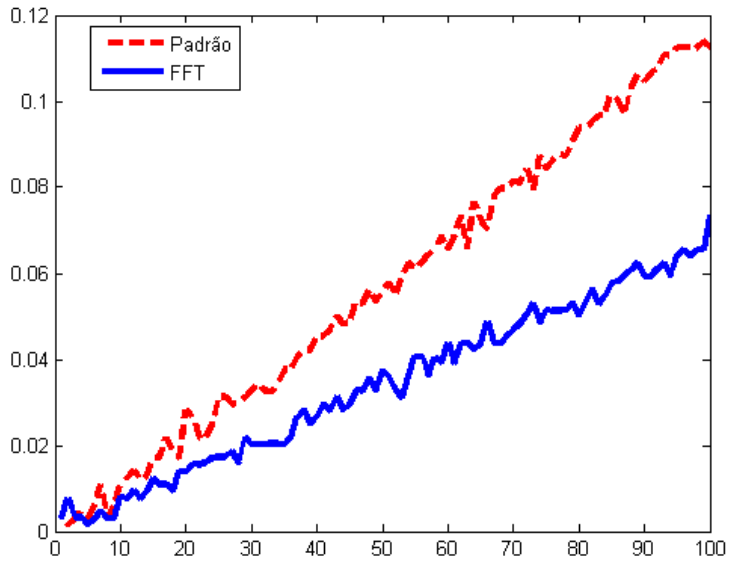
- n $= 10000$. conv$(f, s) = 0.015$s e conv$(s, f) = 0.87$s

Análise (i)

Comparação entre a utilização de dois tamanhos de filtro em diferentes tamanhos de sinais (1:1000:100000).



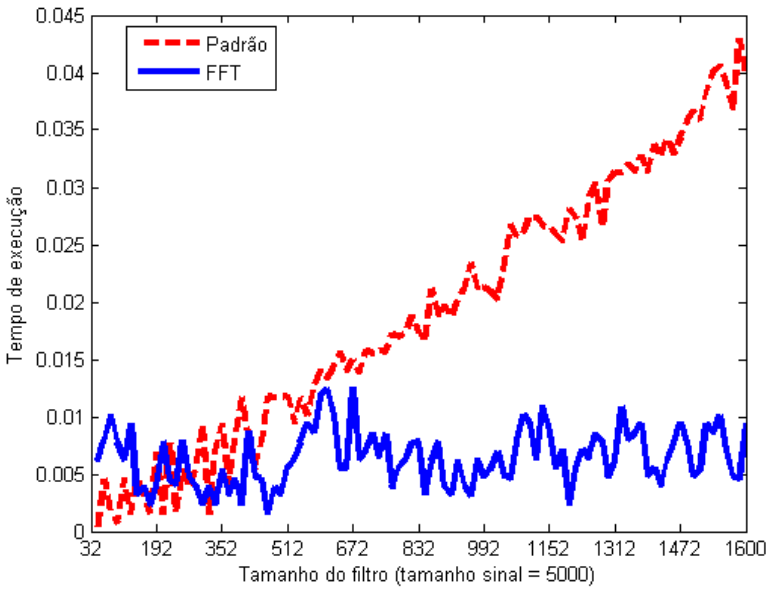
(a) $n = 64$



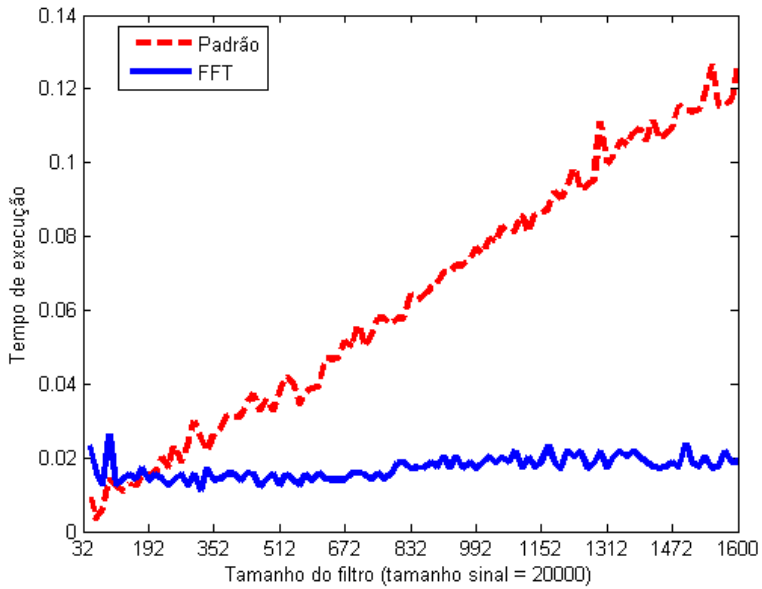
(b) $n = 256$

Análise (ii)

Comparação entre a utilização de dois tamanhos de sinais em diferentes tamanhos de filtro (k1=16:16:1600).



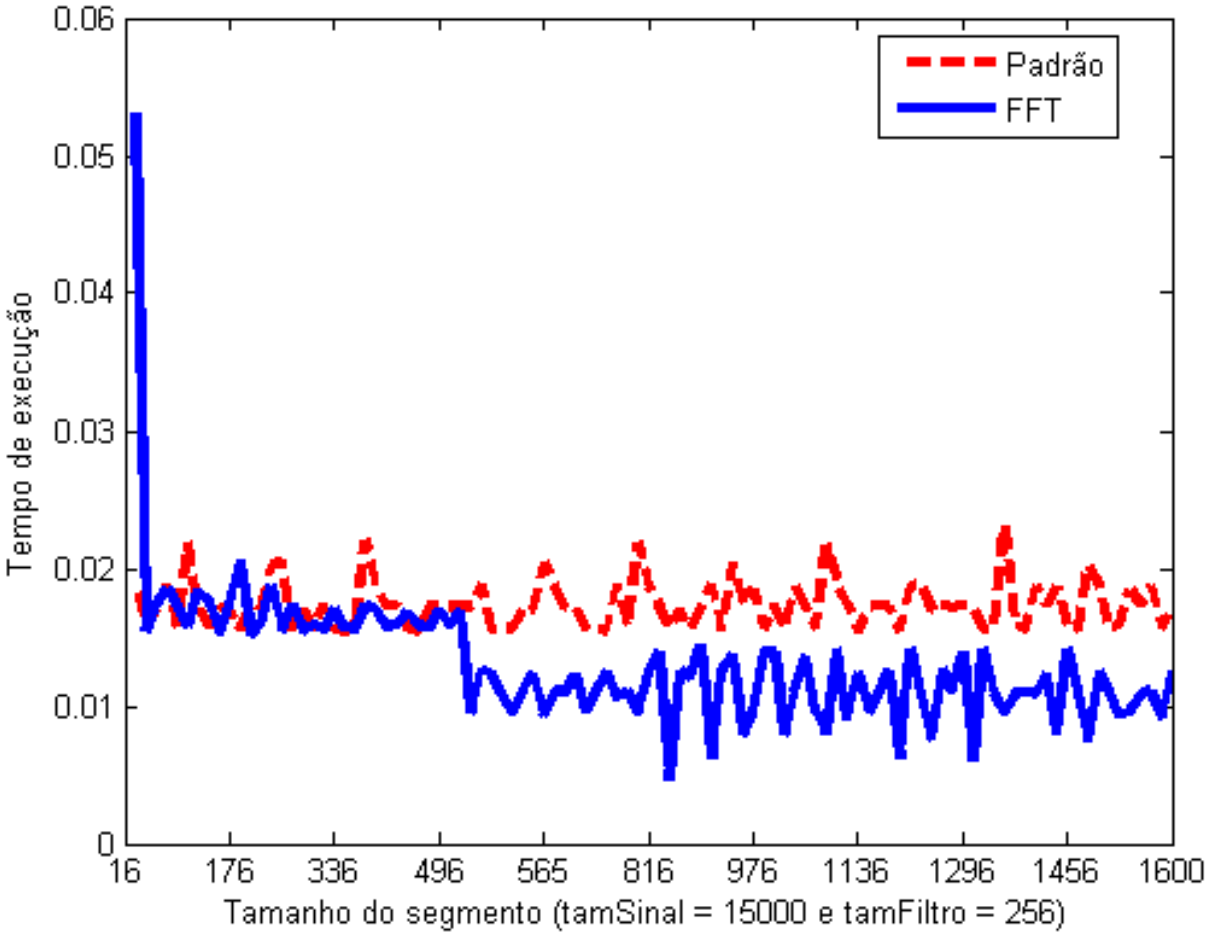
(a) $n = 5000$



(b) $n = 20000$

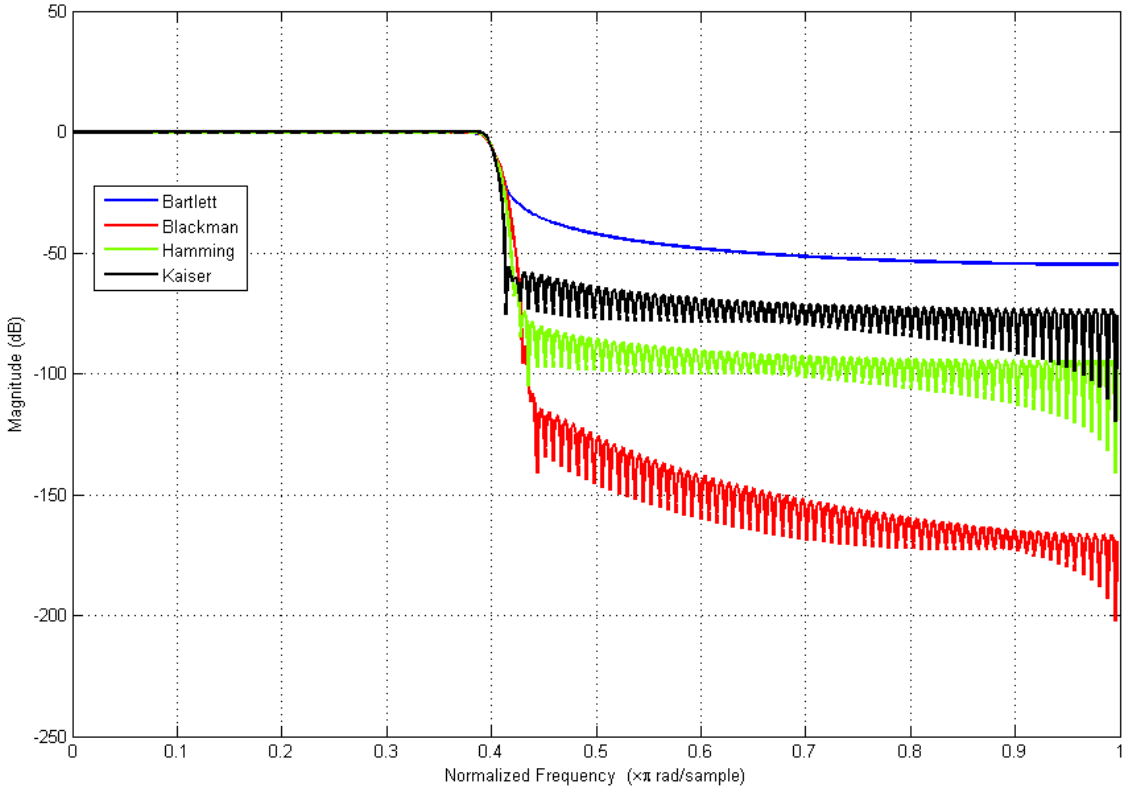
Análise (iii)

Importância da escolha de um bom tamanho de segmento. No Matlab: 3780



Análise (iv)

Diferentes janelas.



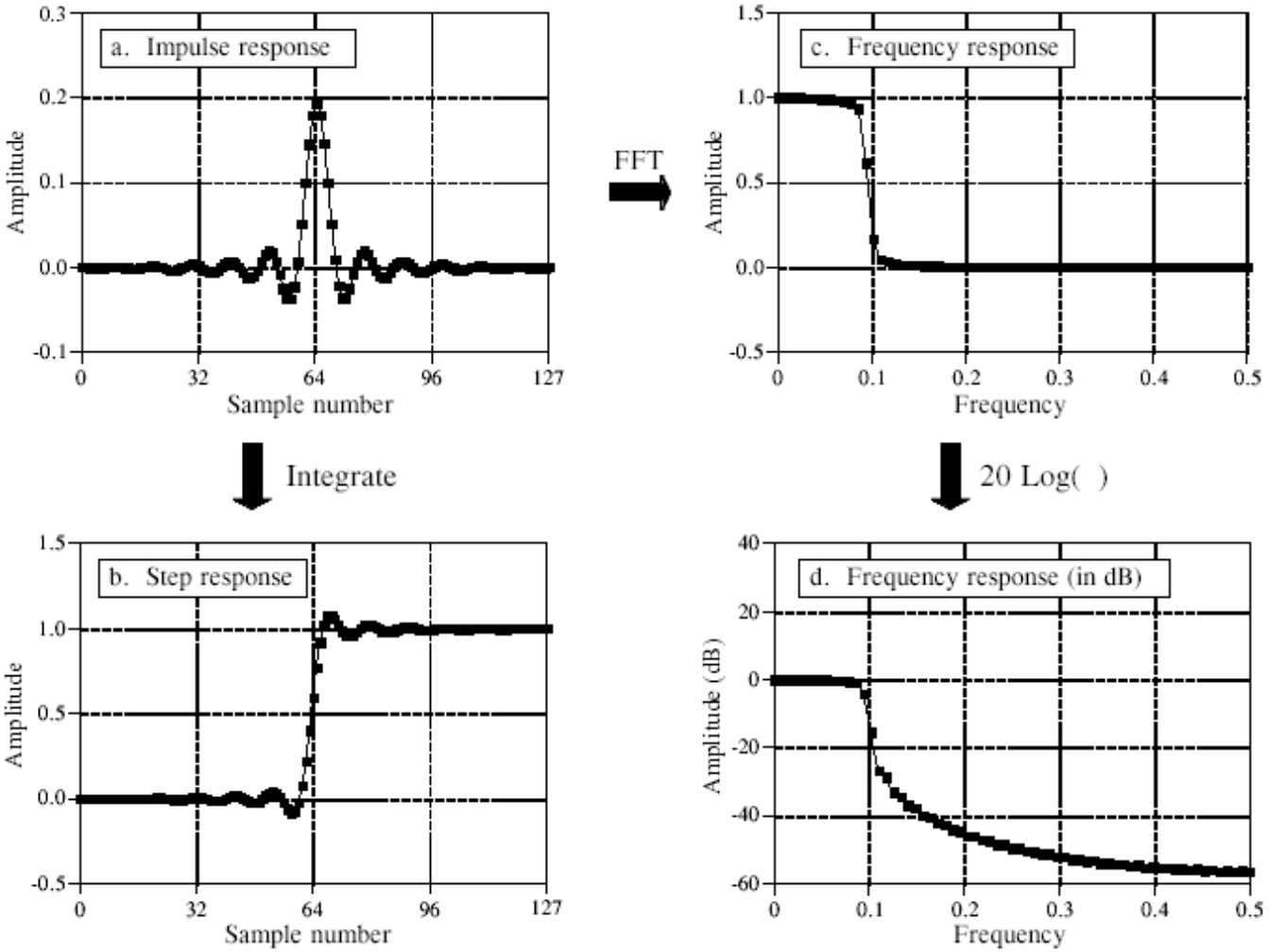
Máximo das diferenças entre as convoluções FFTs:
[0.00350 0.00389 0.00953 0.00305 0.00896 0.00591]

Análise (vi)

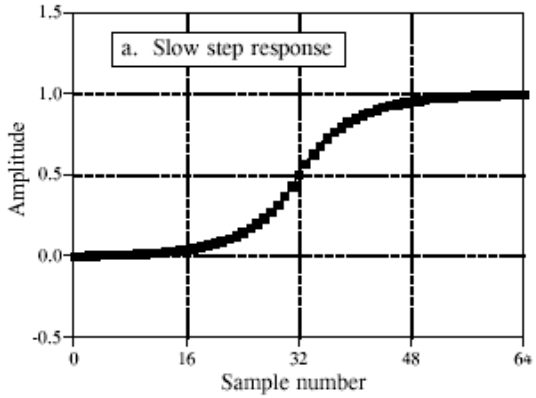
Algumas outras comparações.

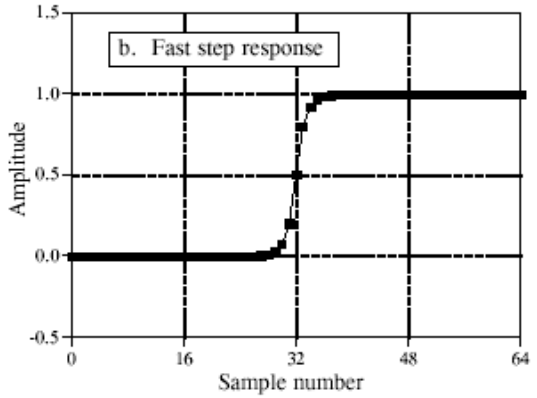
- Comparação entre a convolução padrão e a FFT inteira (mesmos tempos de execução); `ifft(fft([hw zeros(1,length(s)-1)]).*fft([s zeros(1,length(hw)-1)]));`
- Filtros passa-alta, passa-banda, para-banda (não apresentados) e análise destes filtros;
- Implementação do método (sem otimização) e comparação com o Matlab (diferença média de 1/3 no tempo de execução);

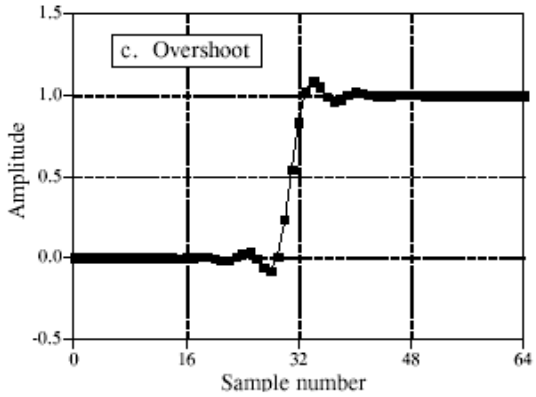
Parâmetros de filtros

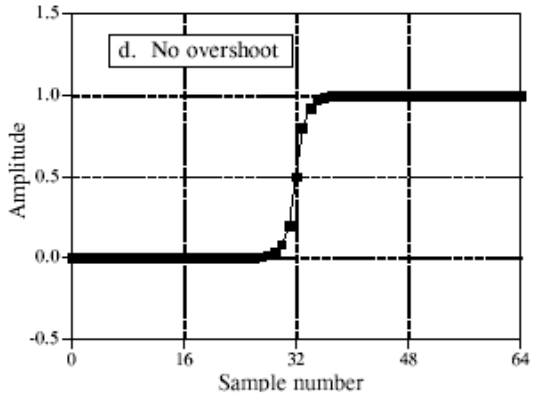


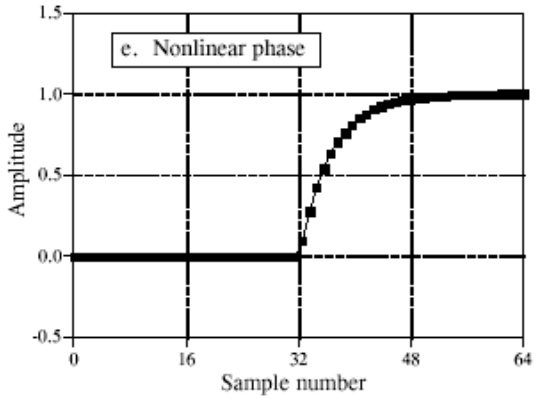
Domínio do tempo

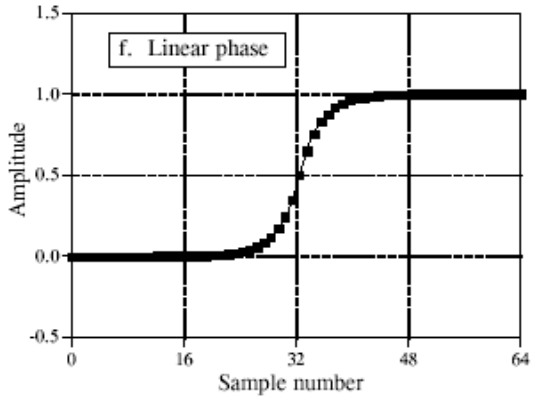












Domínio da frequência