Processamento Digital de Sinais

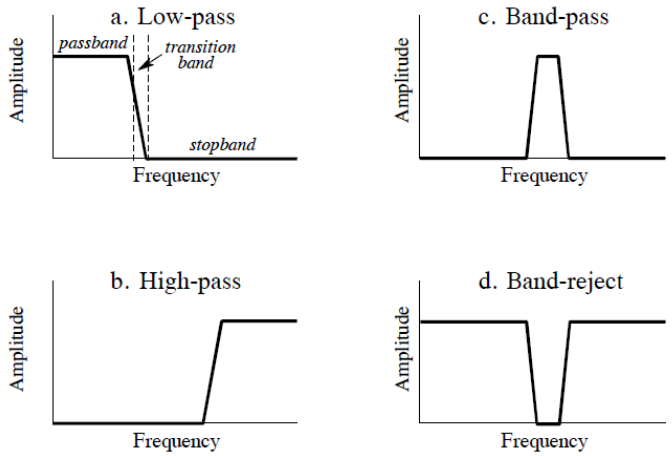
MÓDULO 8
Filtros Personalizados e Convolução por FFT

Gustavo Luís F. Vicente

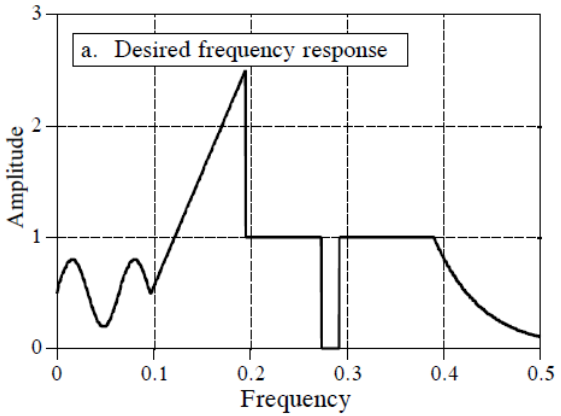
Filtros Digitais Personalizados

Introdução

- Vimos a implementação de filtros com comportamentos-padrão:



- E se desejarmos / precisarmos de um filtro com comportamento diferente destes?
  Por exemplo, algo assim:

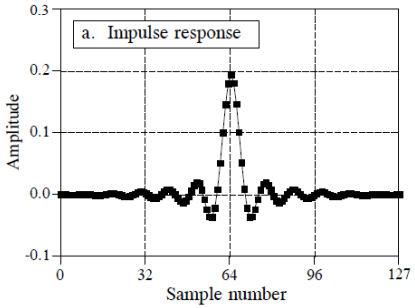


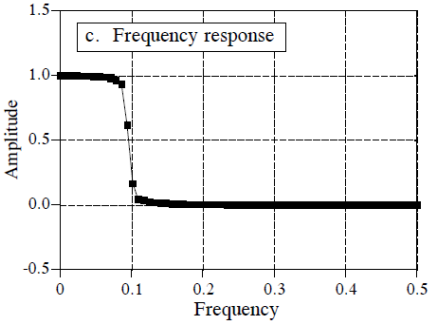
Filtros Digitais Personalizados

Introdução

- Filtros padrão são representados por expressões matemáticas:

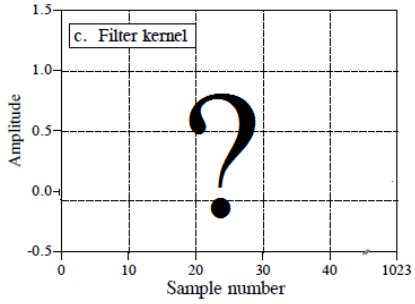
$$h[i] = \frac{\sin ix}{ix}$$

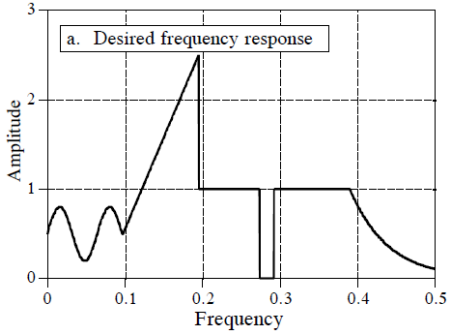




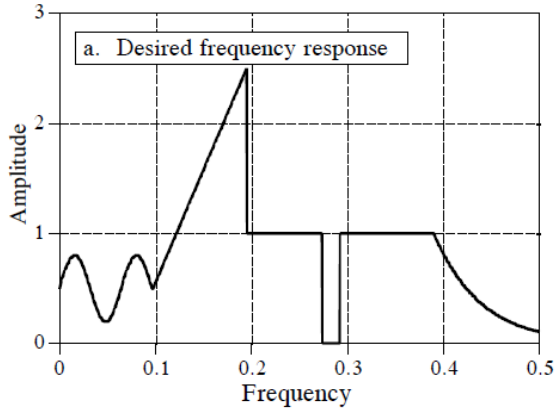
- Filtros personalizados são representados por um conjunto de valores em um vetor:

$$h[i] = \{a, b, c, \ldots\}$$

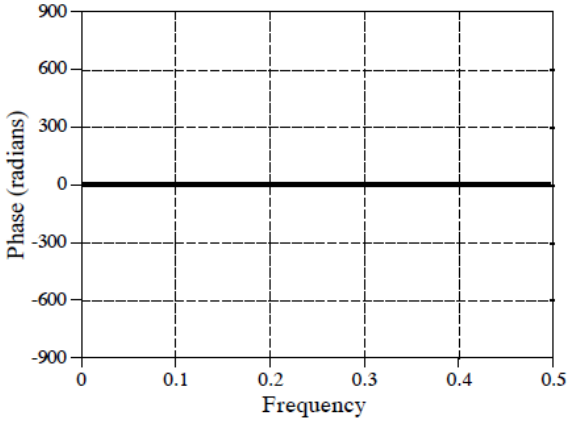




Filtros Digitais Personalizados

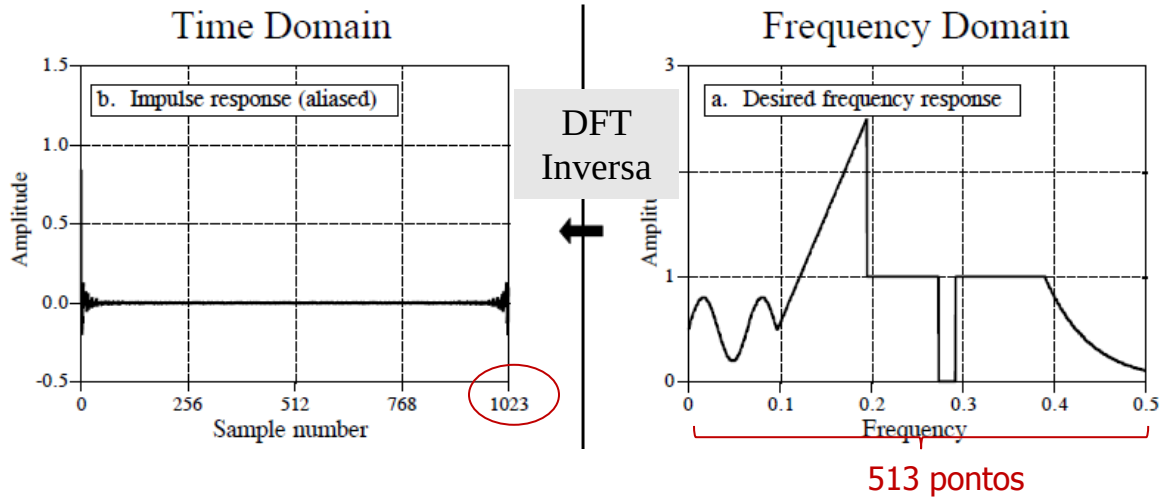


Representado por um vetor de valores que indicam a magnitude de cada componente espectral (cada frequência representada), distribuídas no intervalo entre 0 e 0,5f_s (taxa de amostragem).



Deve haver um outro vetor, de mesmo tamanho, representando a fase de cada componente espectral correspondente. No nosso caso, definimos as fases com zero em todo o vetor.

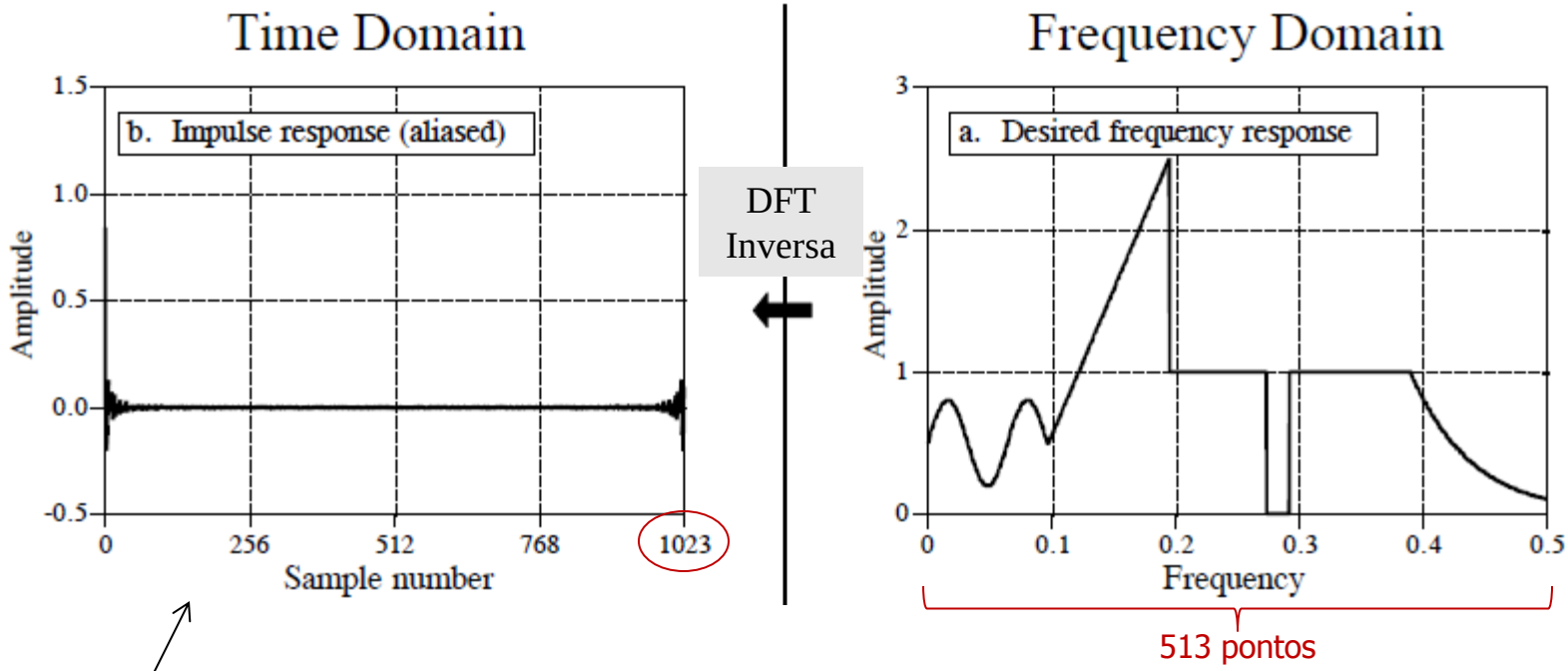
Filtros Personalizados



513 pontos

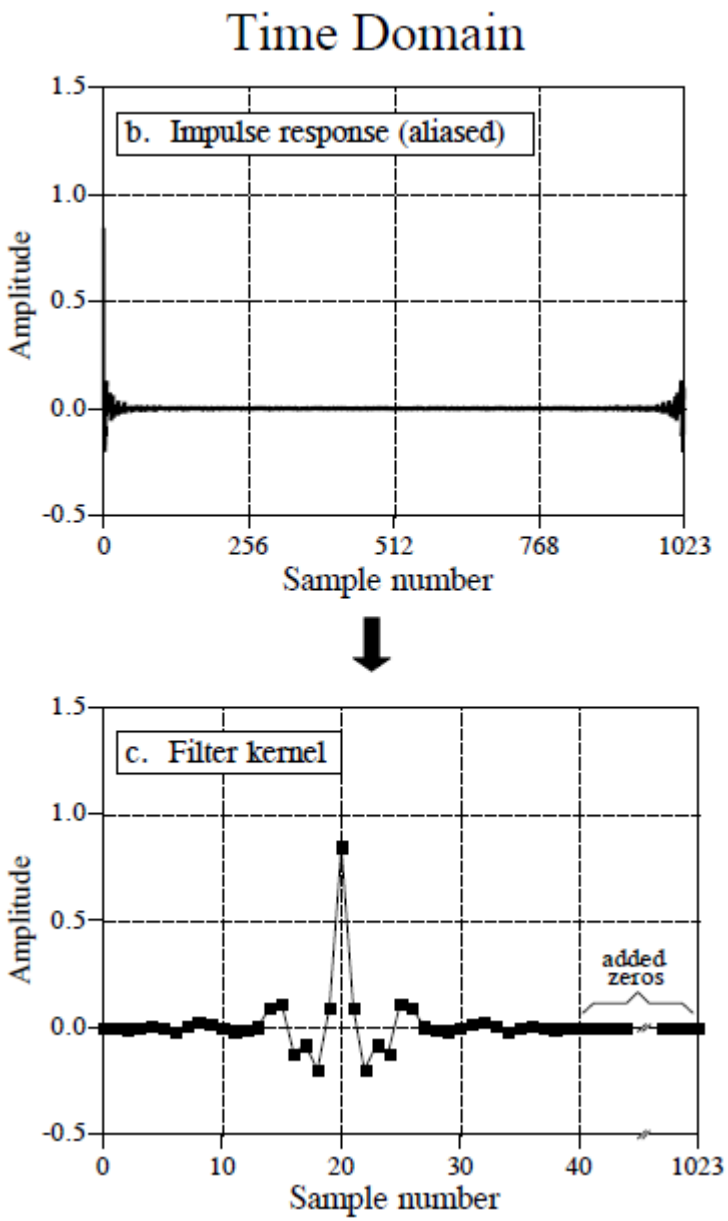
```
2000 'INVERSE FAST FOURIER TRANSFORM SUBROUTINE
2010 'Upon entry, N% contains the number of points in the IDFT, REX[ ] and
2020 'IMX[ ] contain the real and imaginary parts of the complex frequency domain.
2030 'Upon return, REX[ ] and IMX[ ] contain the complex time domain.
2040 'All signals run from 0 to N%-1.
2050 '
2060 FOR K% = 0 TO N%-1                    'Change the sign of IMX[ ]
2070   IMX[K%] = -IMX[K%]
2080 NEXT K%
2090 '
2100 GOSUB 1000                            'Calculate forward FFT
2110 '
2120 FOR I% = 0 TO N%-1                    'Divide the time domain by N% and
2130   REX[I%] =  REX[I%]/N%               'change the sign of IMX[ ]
2140   IMX[I%] = -IMX[I%]/N%
2150 NEXT I%
2160 '
2170 RETURN
```

Filtros Personalizados



- Resposta ao impulso correspondente à resposta em frequência ao lado
- Esta resposta ao impulso não está definida de forma conveniente à sua utilização em um filtro, por um computador
- Lembre-se que a DFT transforma:
  - N pontos no tempo ➔ N/2+1 pontos em frequência
  - e vice-versa (DFT inversa)

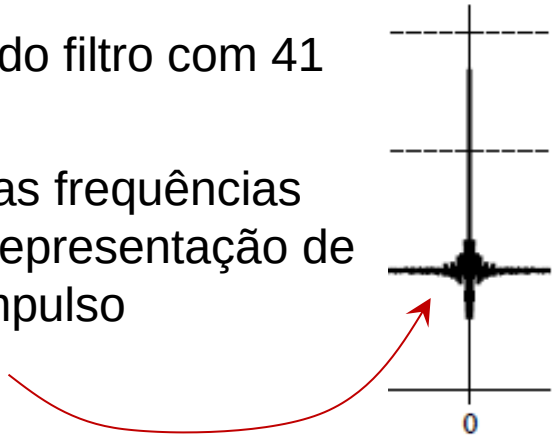
Filtros Personalizados



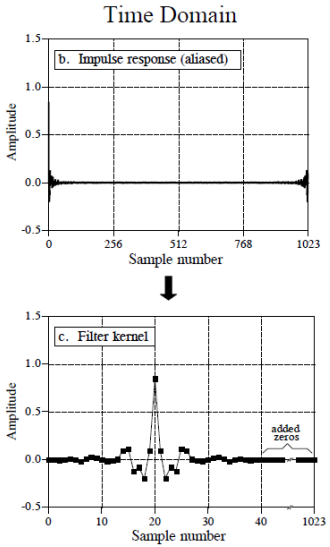
- Devemos:
  - Deslocá-la M/2 pontos para a direita
  - “Enjanelá-la”
  - Complementá-la com zeros à direita

Obs:

- Definimos M=40
  - kernel do filtro com 41 pontos
- Lembremos das frequências negativas na representação de resposta ao impulso

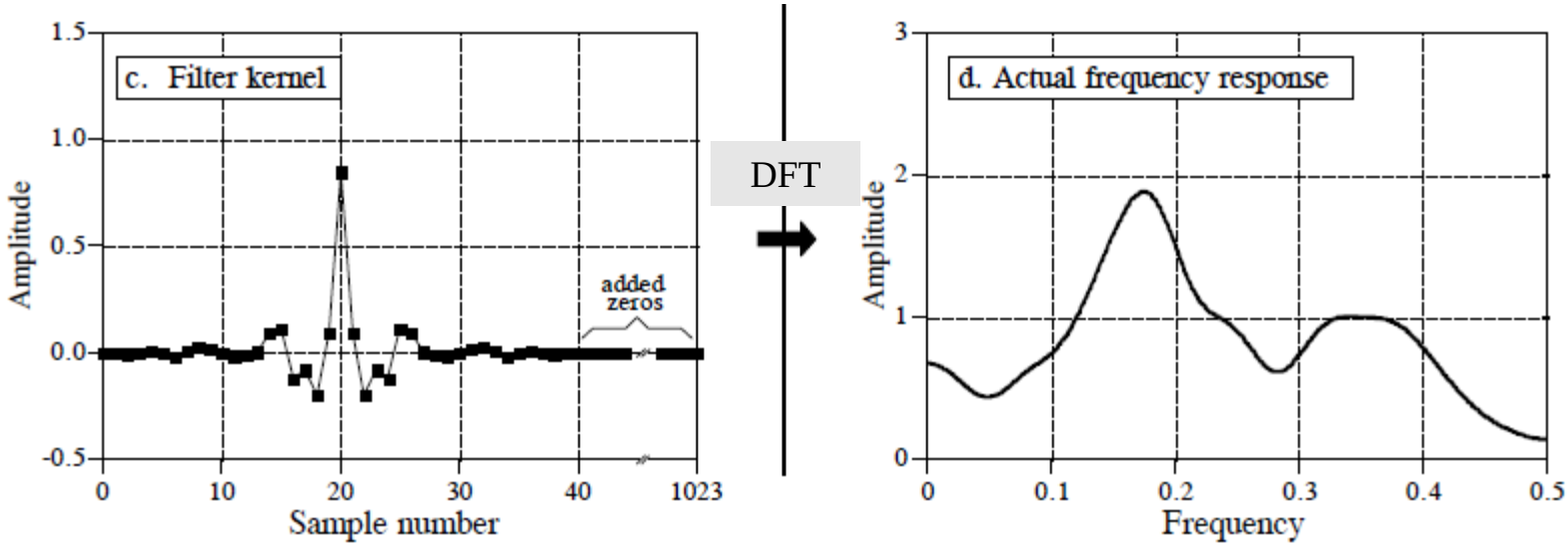


Filtros Personalizados



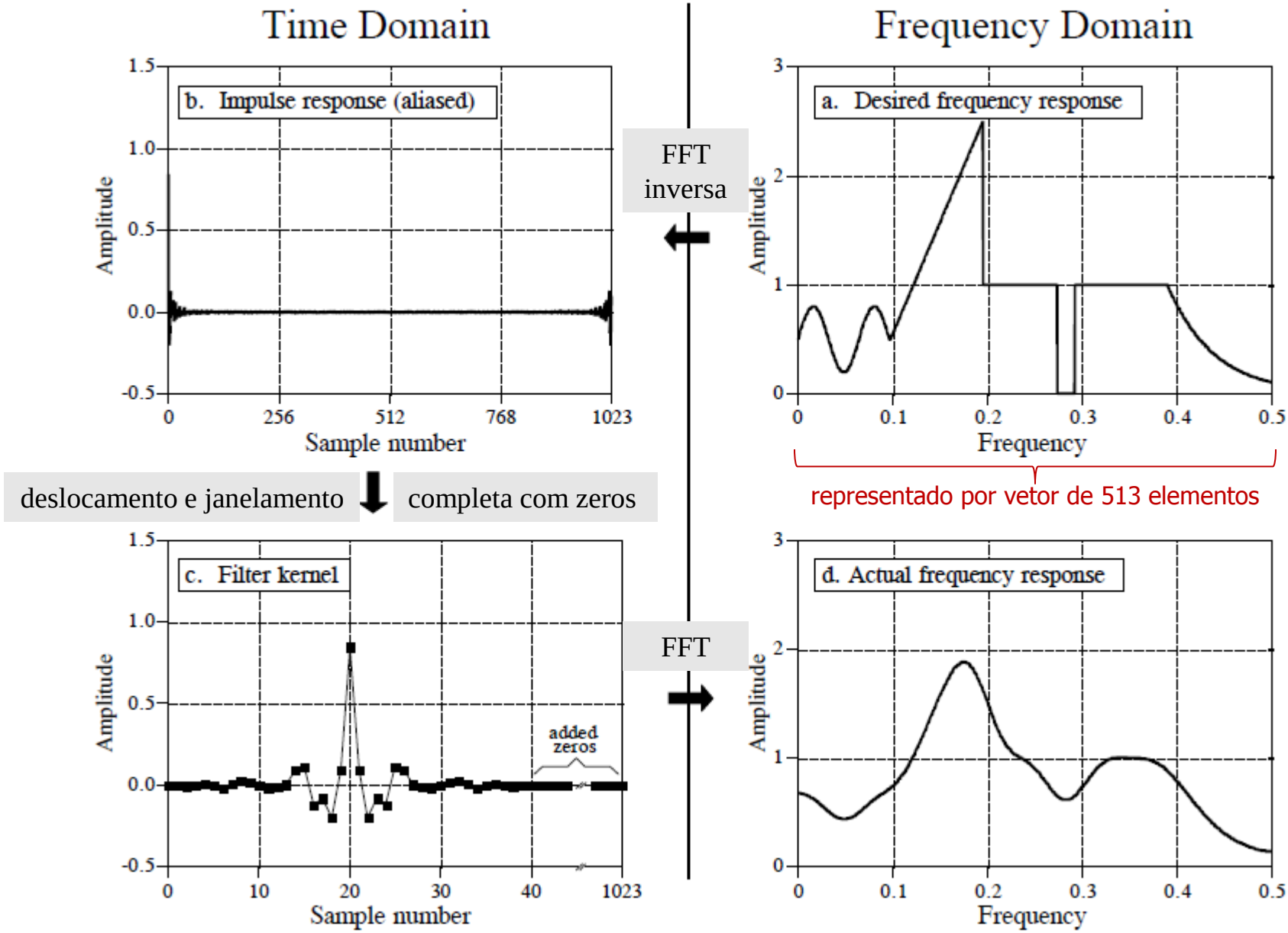

```
100 'CUSTOM FILTER DESIGN
110 'This program converts an aliased 1024 point impulse response into an M+1 point
120 'filter kernel (such as Fig. 17-1b being converted into Fig. 17-1c)
130 '
140 DIM REX[1023]                   'REX[ ] holds the signal being converted
150 DIM T[1023]                     'T[ ] is a temporary storage buffer
160 '
170 PI = 3.14159265
180 M% = 40                         'Set filter kernel length (41 total points)
190 '
200 GOSUB XXXX                      'Mythical subroutine to load REX[ ] with impulse response
210 '
220 FOR I% = 0 TO 1023    'Shift (rotate) the signal M/2 points to the right
230    INDEX% = I% + M%/2
240    IF INDEX% > 1023 THEN INDEX% = INDEX%-1024
250    T[INDEX%] = REX[I%]
260 NEXT I%
270 '
280 FOR I% = 0 TO 1023
290    REX[I%] = T[I%]
300 NEXT I%
310 '                               'Truncate and window the signal
320 FOR I% = 0 TO 1023
330   IF I% <= M% THEN REX[I%] = REX[I%] * (0.54 - 0.46 * COS(2*PI*I%/M%))
340   IF I% > M%  THEN REX[I%] = 0
350 NEXT I%
360 '                               'The filter kernel now resides in REX[0] to REX[40]
370 END
```

Filtros Personalizados



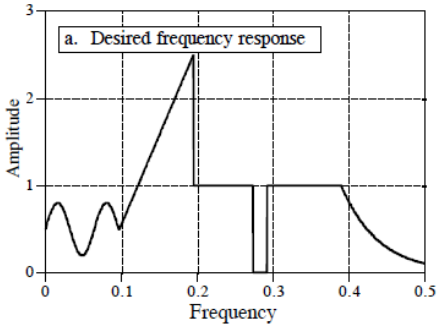
- Testamos o kernel criado, aplicando a DFT e verificando a resposta em frequência obtida

Filtros Personalizados



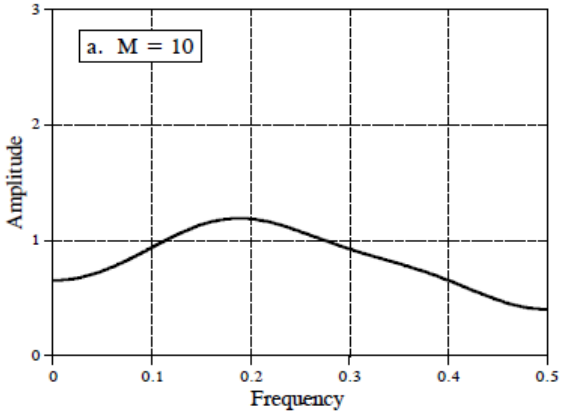
Filtros Personalizados

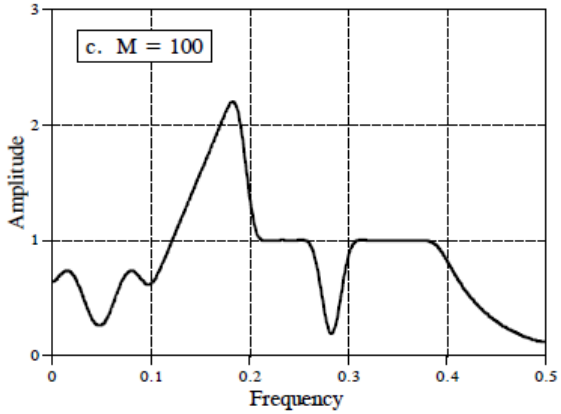
- Resposta em frequência do filtro em função do tamanho do kernel (M)
- Maior M ➔ maior precisão ➔ maior processamento

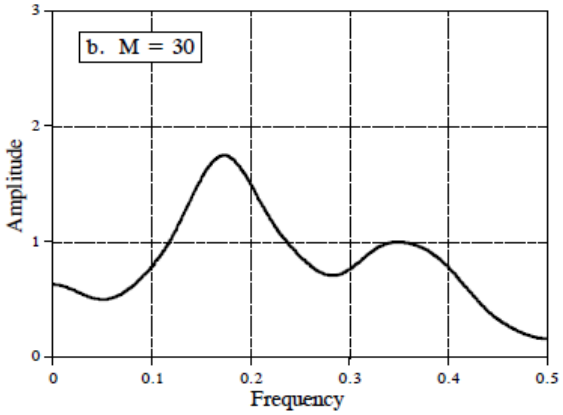


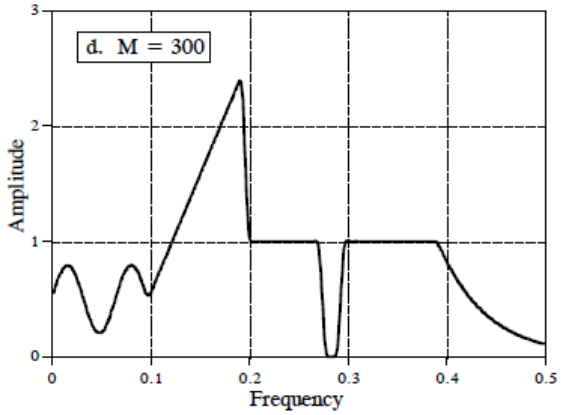
IDEAL

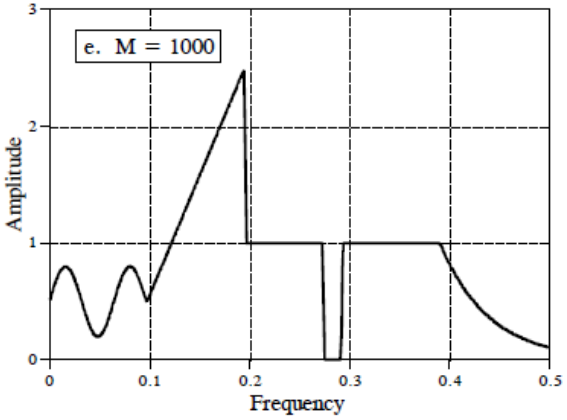
REAIS





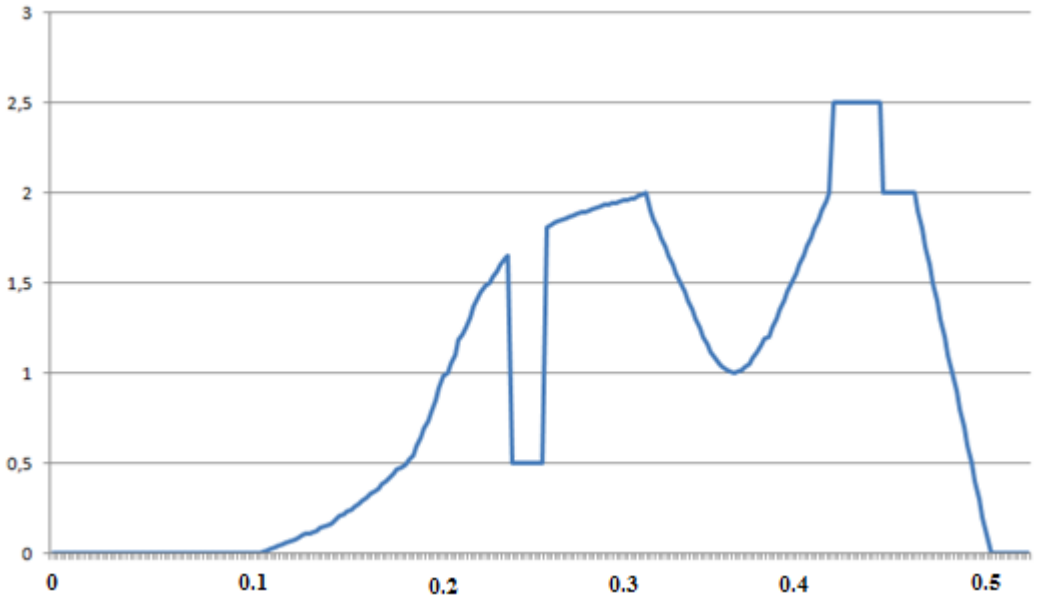






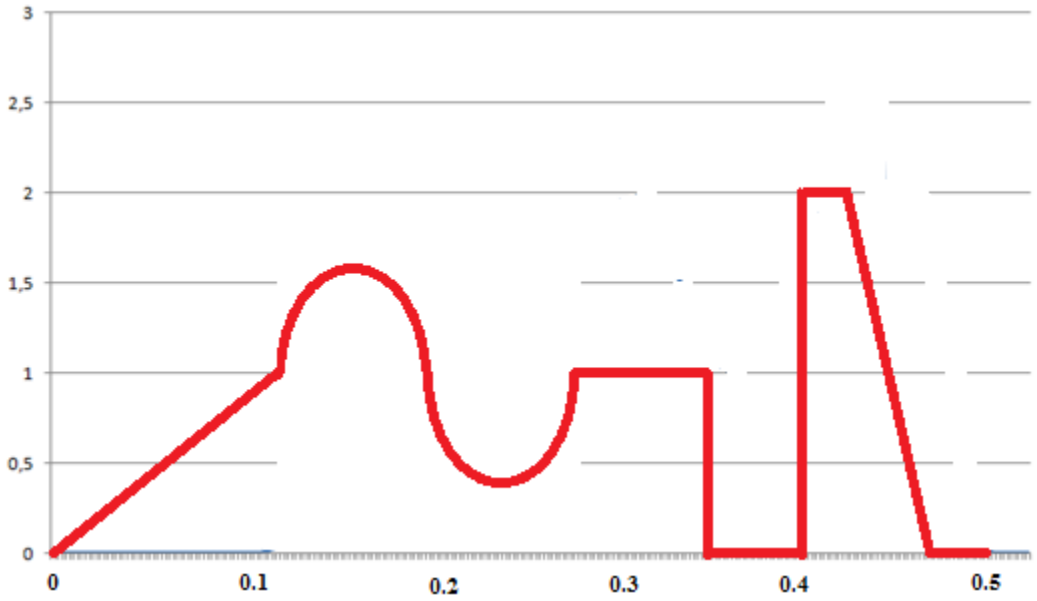
Filtros Personalizados

- TRABALHO - Grupo 1
- Gerar o kernel do filtro que produza a resposta em frequência mostrada na figura abaixo. Gerar o kernel com os tamanhos (M) iguais a 20, 100 e 500, testando-os e plotando as respostas em frequência correspondentes.
- Entregar no dia 23/06 (comporá a Nota2):
    - memória de cálculo resumida e gráficos



Filtros Personalizados

- TRABALHO - Grupo 2
- Gerar o kernel do filtro que produza a resposta em frequência mostrada na figura abaixo. Gerar o kernel com os tamanhos (M) iguais a 20, 100 e 500, testando-os e plotando as respostas em frequência correspondentes.
- Entregar no dia da prova (comporá a Nota2):
    - memória de cálculo resumida e gráficos



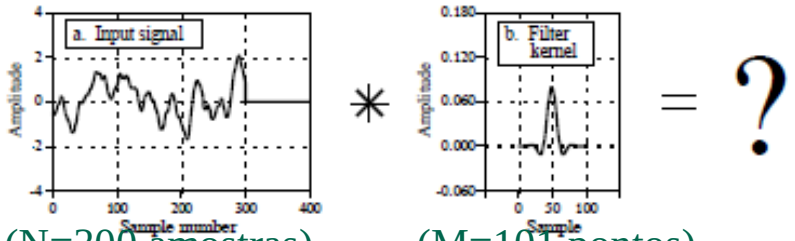
Método de Sobreposição

Situações

- Sinal com um conjunto de amostras muito grande
  - HiFi audio: 5Mbytes/min
  - Video: 500Mbytes/min
- Sinal processado em tempo real
  - Telefonia
- Melhora no tempo de processamento
  - Convolução por FFT
    - a(t) * b(t) ⇐⇒ A[f] x B[f]

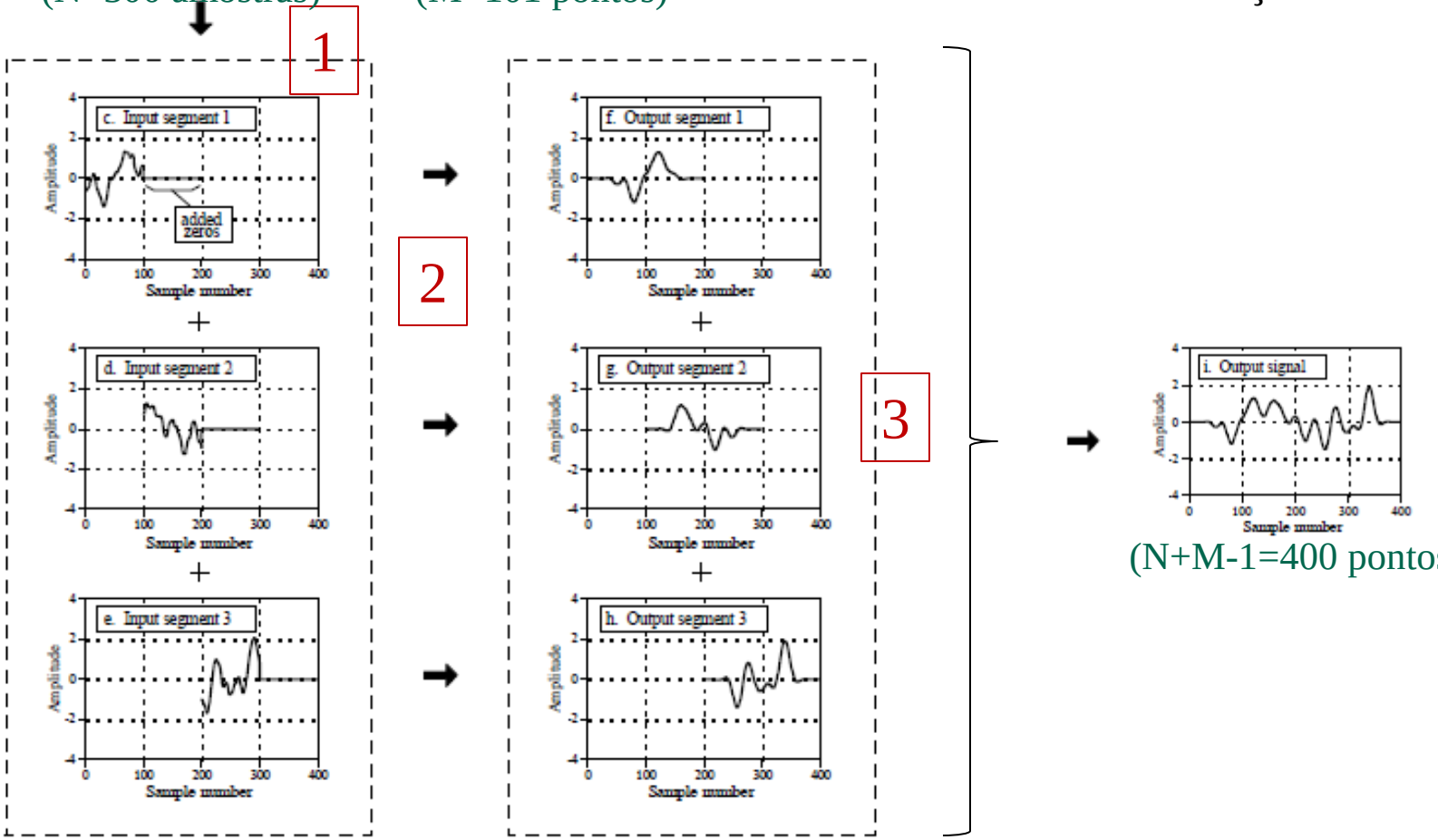
- Lembrando que:
  - $x_N * h_M \rightarrow y_{N+M-1}$

Método de Sobreposição



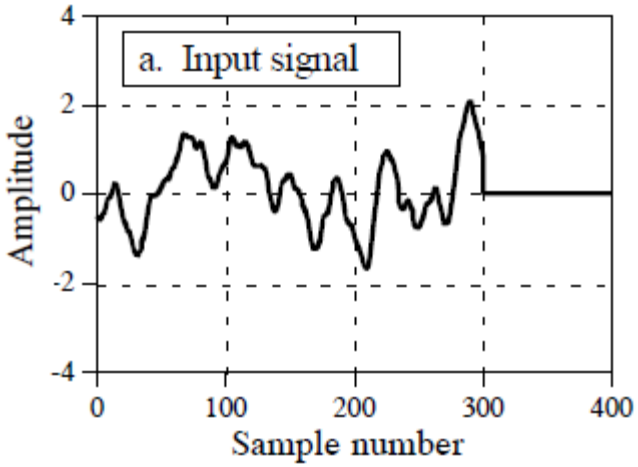
(N=300 amostras) (M=101 pontos)

1) Desmembra o sinal de entrada
2) Convolui cada pedaço com h[i]
3) Soma cada resultado das convoluções

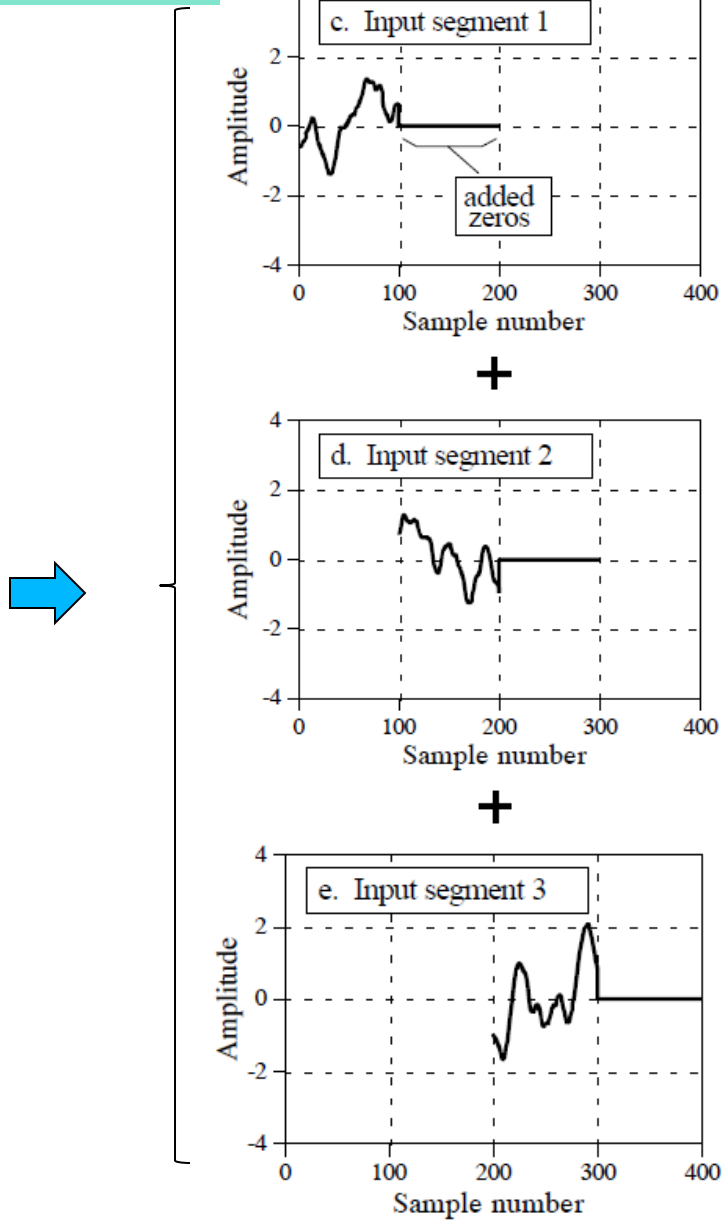


(N+M-1=400 pontos)

Método de Sobreposição



Sinal original: 300 amostras
Complementado com 100 zeros



Convolução por FFT

Considerando que:

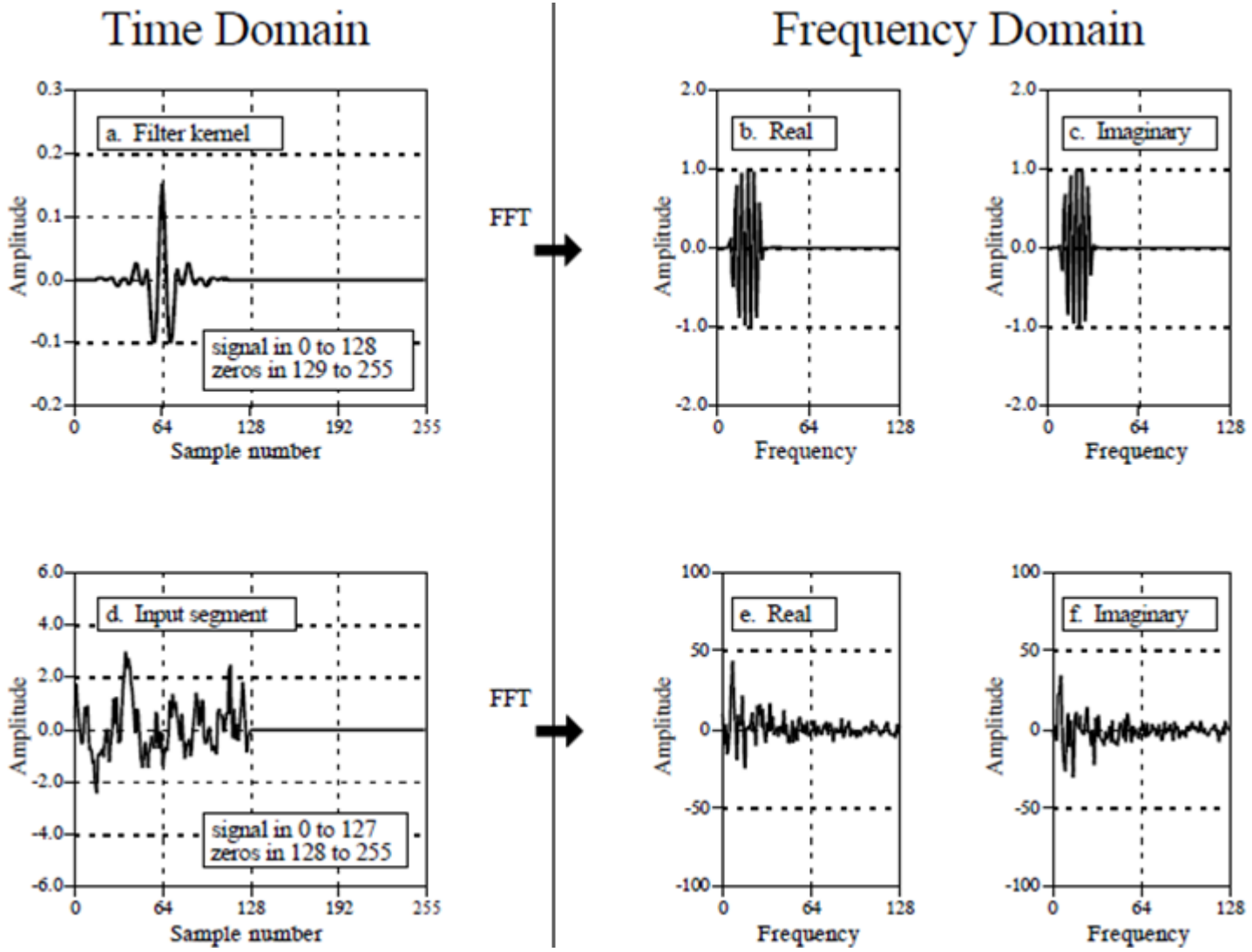
- A FFT é rápida
- a(t) * b(t) ⟷ A[f] x B[f]
- Método de sobreposição
- Abordagem mais rápida para kernel com M>64

- Convolução por FFT em 4 etapas
  1. Obtém a DFT (via FFT) do kernel do filtro
  2. Obtém a DFT (via FFT) do sinal de entrada
  3. Multiplica os dois sinais no domínio da frequência
  4. Obtém o sinal de saída através da DFT Inversa (via FFT Inversa - IFFT) do sinal produto

Convolução por FFT

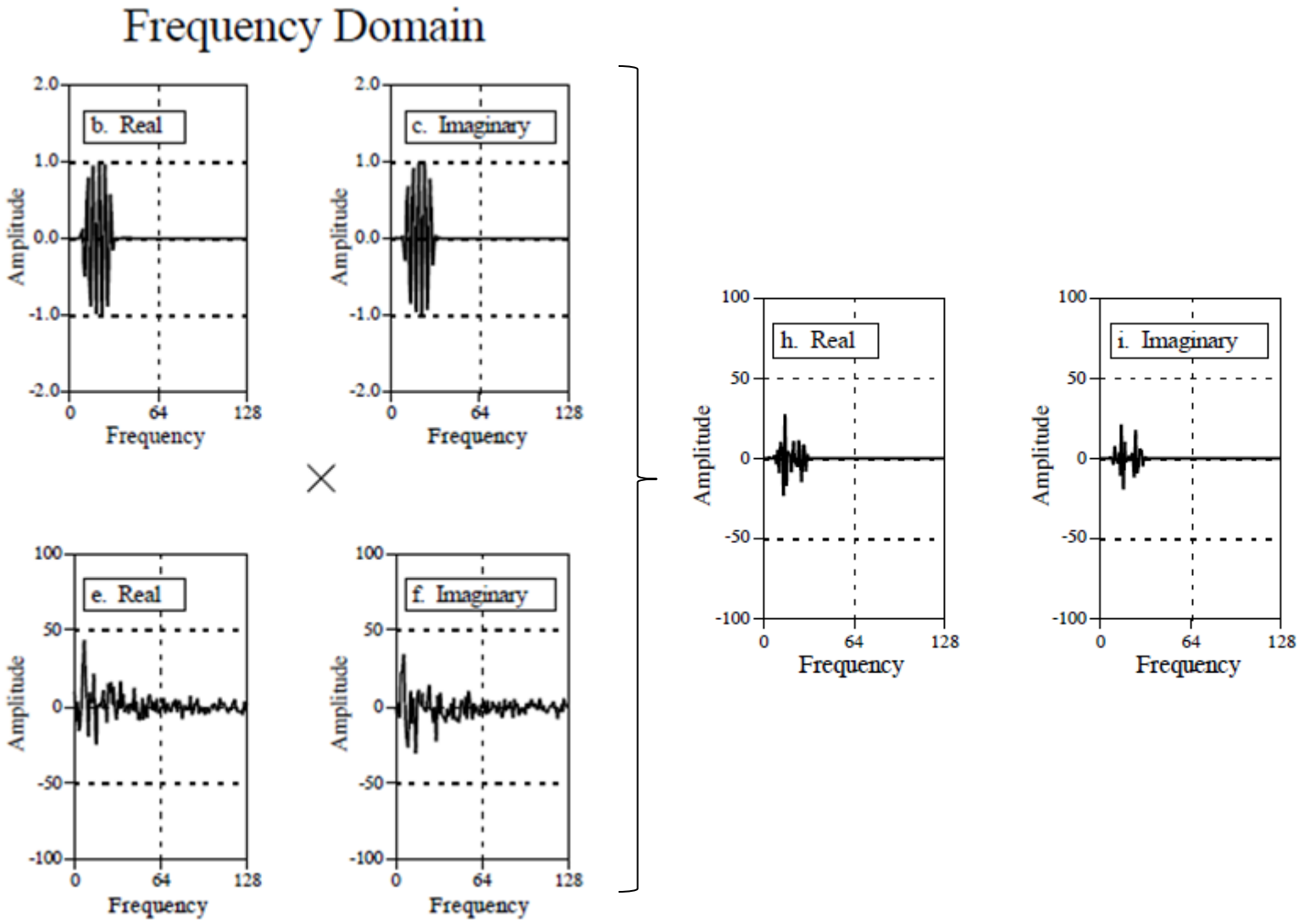
1ª Etapa

2ª Etapa



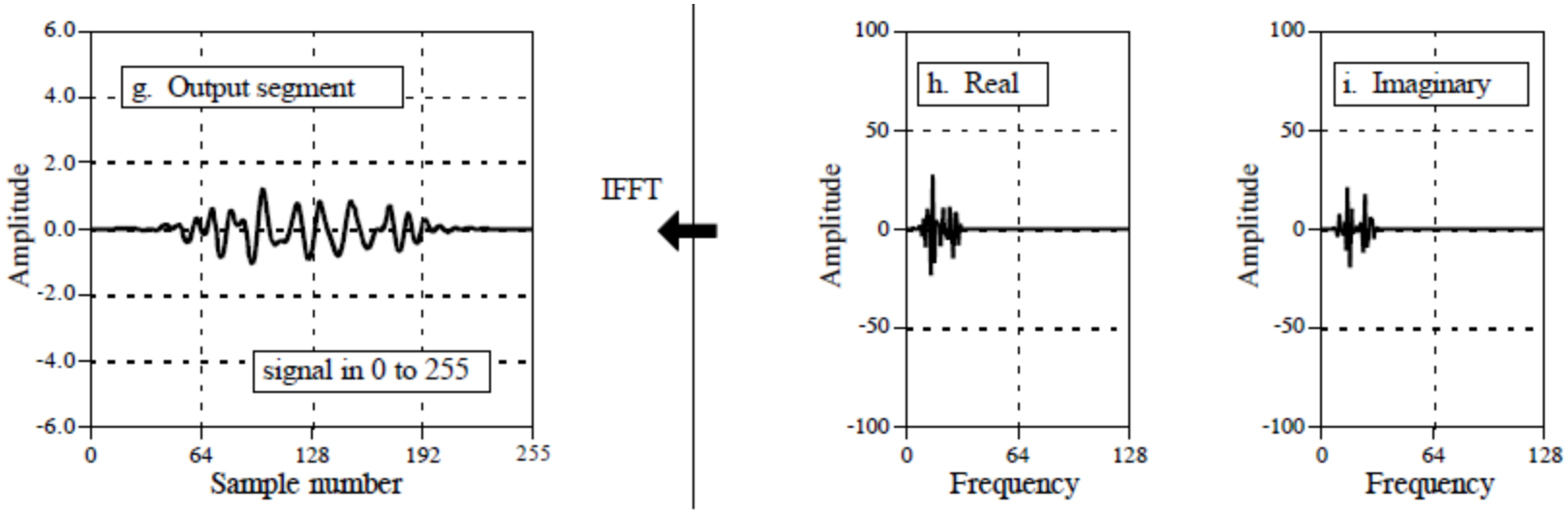
Convolução por FFT

3ª Etapa



Convolução por FFT

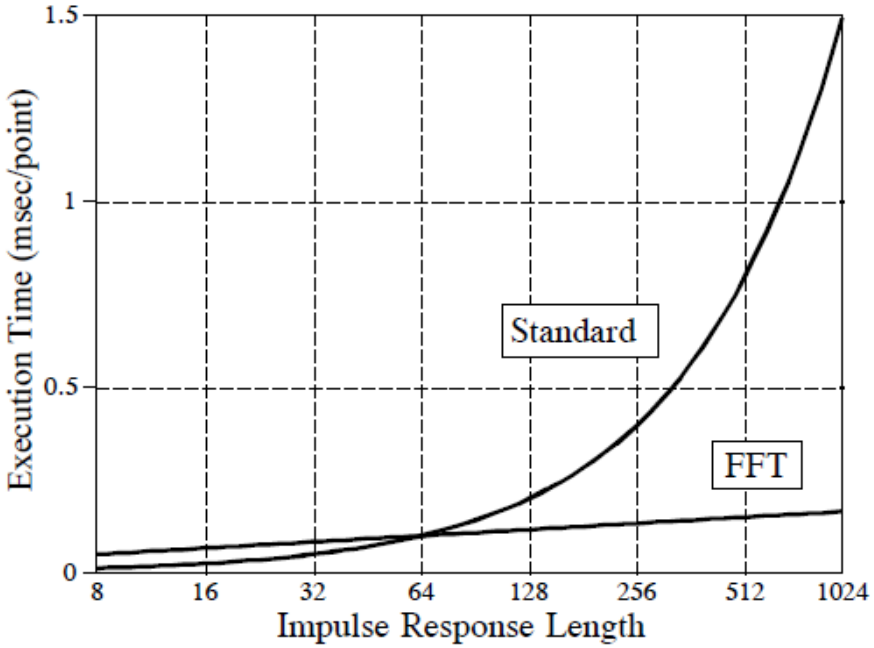
4ª Etapa



Convolução por FFT

Considerações sobre desempenho:

- Filtros com kernel longo pode ser construído sem perda de performance
  - Kernel com M=16000 apenas 2x mais lento que kernel com M=64
- Quanto mais rápido (menos iterações), mais preciso



Convolução por FFT

```
100 'FFT CONVOLUTION
110 'This program convolves a 10 million point signal with a 400 point filter kernel.  The input
120 'signal is broken into 16000 segments, each with 625 points.  1024 point FFTs are used.
130 '
130 '                                   'INITIALIZE THE ARRAYS
140 DIM XX[1023]                        'the time domain signal (for the FFT)
150 DIM REX[512]                        'real part of the frequency domain (for the FFT)
160 DIM IMX[512]                        'imaginary part of the frequency domain (for the FFT)
170 DIM REFR[512]                       'real part of the filter's frequency response
180 DIM IMFR[512]                       'imaginary part of the filter's frequency response
190 DIM OLAP[398]                       'holds the overlapping samples from segment to segment
200 '
210 FOR I% = 0 TO 398                   'zero the array holding the overlapping samples
220   OLAP[I%] = 0
230 NEXT I%
240 '
250 '                                   'FIND & STORE THE FILTER'S FREQUENCY RESPONSE
260 GOSUB XXXX                          'Mythical subroutine to load the filter kernel into XX[ ]
270 GOSUB XXXX                          'Mythical FFT subroutine:  XX[ ] -->  REX[ ] & IMX[ ]
280 FOR F% = 0 TO 512                   'Save the frequency response in REFR[ ] & IMFR[ ]
290   REFR[F%] = REX[F%]
300   IMFR[F%] = IMX[F%]
310 NEXT F%
320 '
330 '                                   'PROCESS EACH OF THE 16000 SEGMENTS
340 FOR SEGMENT% = 0 TO 15999
350   '
360   GOSUB XXXX                        'Mythical subroutine to load the next input segment into XX[ ]
370   GOSUB XXXX                        'Mythical FFT subroutine:  XX[ ]  -->  REX[ ] & IMX[ ]
380   '
390   FOR F% = 0 TO 512  'Multiply the frequency spectrum by the frequency response
400     TEMP     = REX[F%]*REFR[F%] - IMX[F%]*IMFR[F%]
410     IMX[F%]   = REX[F%]*IMFR[F%] + IMX[F%]*REFR[F%]
420     REX[F%]   = TEMP
430   NEXT F%
440   '
450   GOSUB XXXX                        'Mythical IFFT subroutine:  REX[ ] & IMX[ ]  -->  XX[ ]
460   '
470   FOR I% = 0 TO 398   'Add the last segment's overlap to this segment
480     XX[I%] = XX[I%] + OLAP[I%]
490   NEXT I%
500   '
510   FOR I% = 625 TO 1023              'Save the samples that will overlap the next segment
520     OLAP[I%-625] = XX[I%]
530   NEXT I%
540   '
550   GOSUB XXXX                        'Mythical subroutine to output the 625 samples stored
560   '                                 'in XX[0] to XX[624]
570   '
580 NEXT SEGMENT%
590 '
600 GOSUB XXXX                          'Mythical subroutine to output all 399 samples in OLAP[ ]
610 END
```