



UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Código	Disciplina	Professor
ELE042	Processamento de Sinais	Hilton de Oliveira Mota

TRABALHO PRÁTICO

Condições

Grupos de 3 alunos.

Avaliação: 10 pontos. **Bônus:** 5 pontos extras (ver item 4).

Prazo: 05 de outubro de 2025, às 23:59.

Entregar:

- Documento em .pdf contendo identificação dos membros do grupo, descrição da resolução, apresentação das imagens, resultados e análises de desempenho.
- Scripts, códigos fontes e/ou executáveis implementados em Matlab, Python ou C/C++ *que possam ser inspecionados e executados localmente.*

Introdução

Filtros LIT digitais são algoritmos computacionais que agem como seletores de frequências e podem substituir filtros analógicos de forma eficiente quando associados a conversores AD e DA, conforme mostrado na figura abaixo.

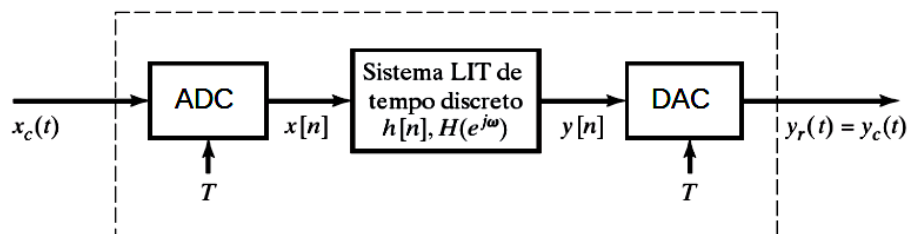


Figura 1: diagrama de blocos de um filtro digital.

Os algoritmos podem ser implementados de diversas maneiras. Por exemplo:

• Pela equação de diferenças:
$$y[n] = \sum_{k=0}^{M-1} b_k x[n-k] - \sum_{k=1}^{N-1} a_k y[n-k]. \quad (1)$$

• Pela convolução com a resposta ao impulso:
$$y[n] = \sum_{k=0}^{N-1} x[n] h[k-n]; \quad (2)$$

• Pela propriedade da multiplicação da Transformada de Fourier de tempo discreto (TFTD):

$$y[n] = x[n] * h[n] \xleftrightarrow{\mathcal{F}} Y(e^{j\omega}) = X(e^{j\omega}) \cdot H(e^{j\omega}). \quad (3)$$

A escolha de determinada forma depende de fatores como o tipo da resposta ao impulso (FIR ou IIR), a arquitetura de hardware do processador, a complexidade do algoritmo e o desempenho computacional almejado (ex.: paralelização, hardware dedicado, etc.).

Se o processamento for realizado em blocos, a filtragem pode ser implementada por meio da *convolução circular*¹ $y[n] = x[n] \odot h[n]$ composta por enchimento com zeros e deslocamentos circulares. O procedimento é mostrado na figura abaixo, em que N_x e N_h representam os comprimentos de $x[n]$ e $h[n]$, respectivamente.

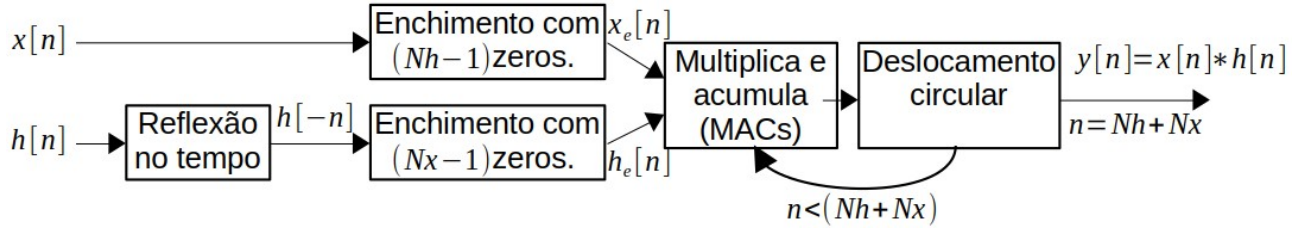


Figura 2: convolução circular.

No domínio da frequência, a convolução circular se torna uma multiplicação de espectros de frequência, como mostrado na figura abaixo.

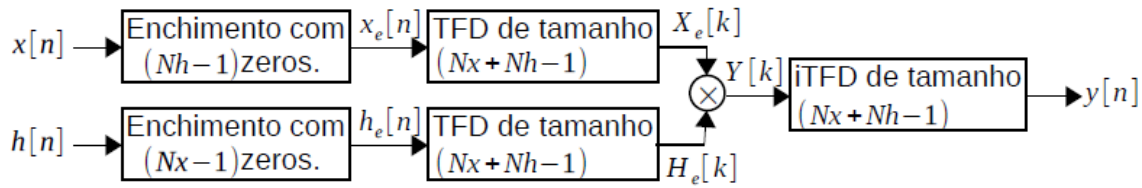


Figura 3: filtragem digital no domínio da frequência.

Do ponto de vista matemático, tem-se:

$$y[n] = \sum_{k=0}^{M-1} h[k] x[n-k] = h[0]x[n] + h[1]x[n-1] + \dots + h[M-1]x[n-(M-1)]$$

$$\downarrow \mathcal{F}$$

$$Y(e^{j\omega}) = h[0]X(e^{j\omega}) + h[1]e^{-j\omega}X(e^{j\omega}) + \dots + h[M-1]e^{-j(M-1)\omega}X(e^{j\omega})$$

$$= [h[0] + h[1]e^{-j\omega} + \dots + h[M-1]e^{-j(M-1)\omega}]X(e^{j\omega}).$$

O termo $[h[0] + h[1]e^{-j\omega} + \dots + h[M-1]e^{-j(M-1)\omega}] = H(e^{j\omega})$ é a Transformada de Fourier de tempo discreto de $h[n]$. Assim,

$$Y(e^{j\omega}) = H(e^{j\omega}) \cdot X(e^{j\omega}) \rightarrow \mathcal{F}(y[n]) = \mathcal{F}(h[n]) \cdot \mathcal{F}(x[n]) \rightarrow$$

$$y[n] = \mathcal{F}^{-1}[\mathcal{F}(h[n]) \cdot \mathcal{F}(x[n])].$$

Ou, usando a FFT²,

$$y[n] = \text{iFFT}[\text{FFT}(h[n]) \cdot \text{FFT}(x[n])].$$

Ambas abordagens podem ser usadas no caso de filtros FIR, entretanto em filtros IIR elas causarão distorções devido à truncagem da resposta ao impulso infinita.

1 OPPENHEIM, Alan V.; SCHAFER, Ronald W. Convolução circular. In: Processamento em tempo discreto de sinais. 3ª ed. São Paulo: Pearson Education do Brasil, 2012. Seção 8.6.5, pág. 386.

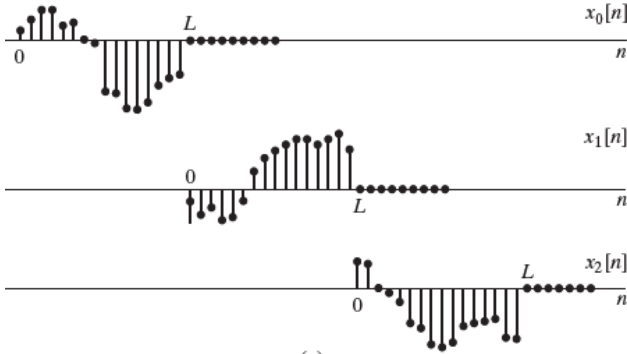
2 N. do a.: é preciso tomar cuidado ao calcular a Transformada de Fourier inversa (iFFT) porque geralmente sobram resquícios de números complexos devido à resolução limitada em ponto flutuante. É comum adotar-se a seguinte prática:

$$y[n] = \text{real}\{\text{iFFT}[\text{FFT}(h[n]) \cdot \text{FFT}(x[n])]\}.$$

Por fim, se o sinal $x[n]$ tiver um comprimento muito grande ou a filtragem for realizada em tempo real, o processamento pode ser realizado *em blocos* por meio dos métodos de sobreposição e soma (overlap-add) ou sobreposição e armazenamento (overlap-save)³, ilustrados abaixo.

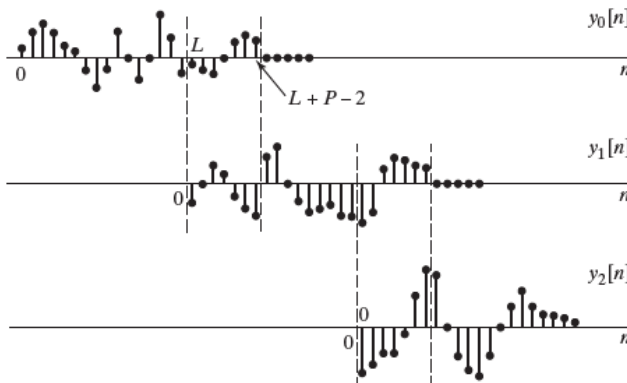
Overlap-add

a) Decomposição de $x[n]$ em blocos de L pontos não sobrepostos $x_k[n]$, preenchidos com zeros.



b) Convolução circular de cada bloco

$$y_k[n] = x_k[n] \odot h[n].$$

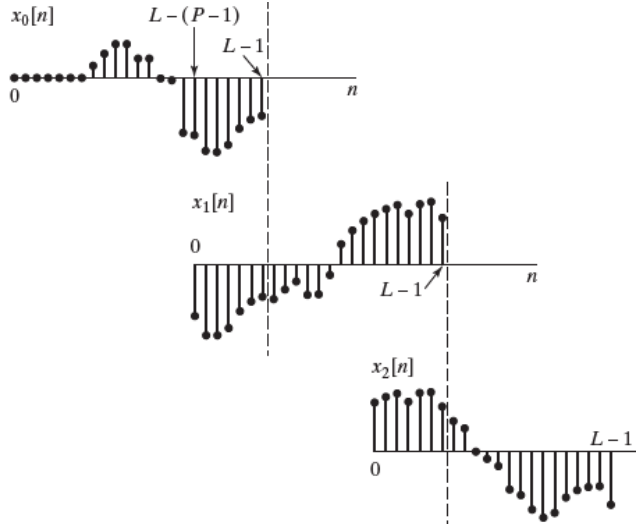


c) Soma dos resultados parciais devidamente deslocados.

$$y[n] = \sum_{k=0}^{\infty} y_k[n - kL]$$

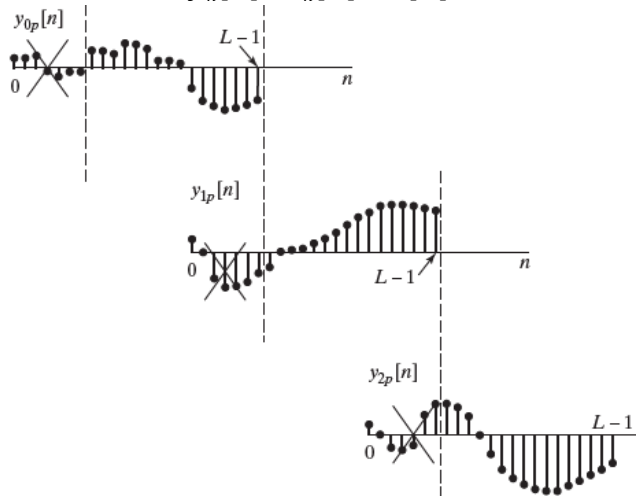
Overlap-save

a) Decomposição de $x[n]$ em blocos sobrepostos de $(L+P-1)$ amostras $x_k[n]$.



b) Convolução circular de cada bloco

$$y_k[n] = x_k[n] \odot h[n].$$



c) Descarte das amostras iniciais e concatenação dos resultados parciais devidamente deslocados.

$$y[n] = \sum_{k=0}^{\infty} y_k[n - k(L - P + 1) + (P - 1)]$$

Figura 4: filtragem digital em blocos.

3 OPPENHEIM, Alan V.; SCHAFER, Ronald W. Implementação de sistemas lineares invariantes no tempo usando a TFD. In: Processamento em tempo discreto de sinais. 3ª ed. São Paulo: Pearson Education do Brasil, 2012. Seção 8.7.3, pág. 394.
MITRA, Sanjit K. Linear convolution of a finite-length sequence with an infinite-length sequence. In: Digital signal processing: a computer-based approach. 4th ed., New York: McGraw-Hill, 2011. Section 5.10.3, pp. 239.

Especificação de requisitos

A figura abaixo apresenta a forma de onda e o espectro de frequências⁴ de um sinal de áudio contido no arquivo “*audio_corrompido.wav*”, fornecido em anexo⁵. O sinal foi corrompido por um ruído aleatório com espectro de faixa larga e duração finita entre aproximadamente 16 e 26 segundos, como também pode ser visto na figura.

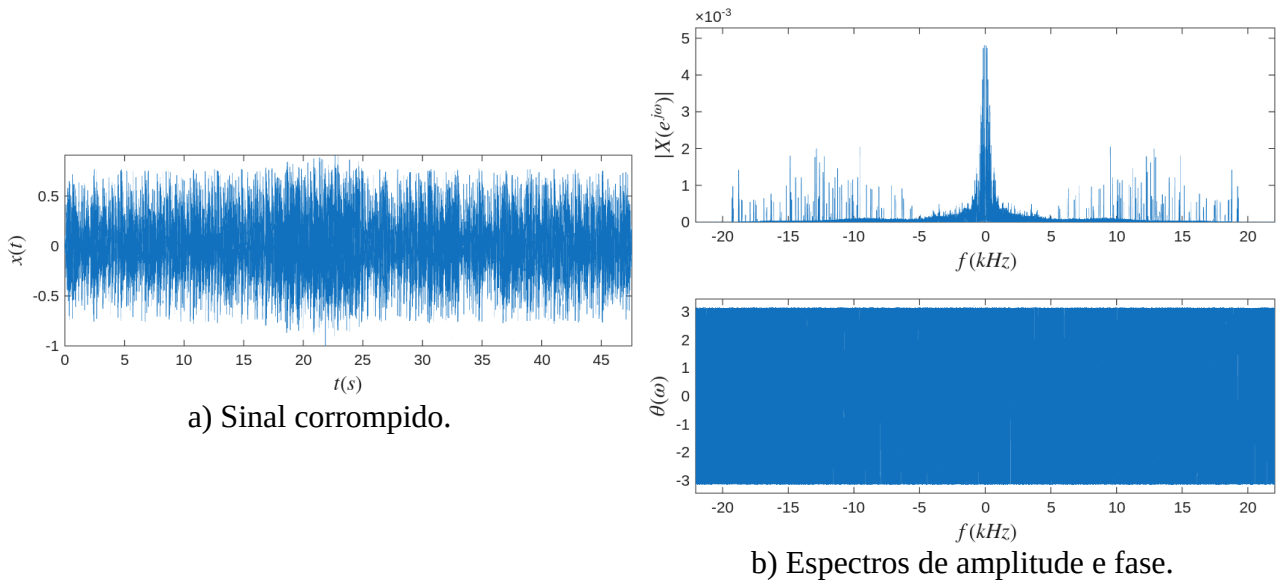


Figura 5: sinal nos domínios do tempo e da frequência.

A origem do ruído é desconhecida e a primeira impressão é de que ele pode ser eliminado por meio de um filtro passa-baixas com frequência de corte próxima a 6kHz. Entretanto, a análise do espectro de amplitude em *dB* permite perceber que o espectro do ruído inicia em aproximadamente 5kHz e se estende até cerca de 18 kHz, como visto na figura a seguir. Há considerável sobreposição do ruído com componentes de alta frequência do sinal, portanto compreende-se que a filtragem passa-baixas causará atenuações nas altas frequências, causando a sensação de um áudio “abafado”.

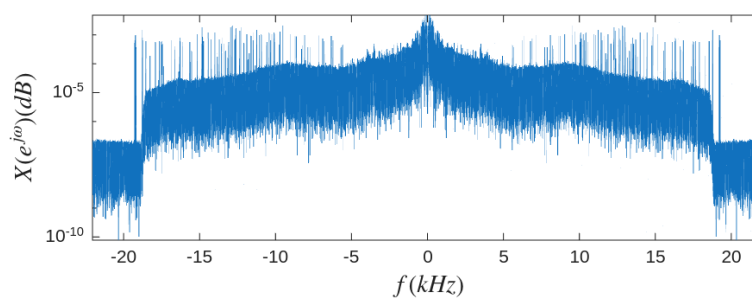


Figura 6: espectro de frequências em dB.

4 No Matlab, veja as funções:
`X = 1/N * fftshift(fft(x));`
`absX = abs(X);`
`angX = angle(X);`

5 No Matlab, o arquivo de áudio pode ser aberto e tocado por meio dos comandos
`[x, fs] = audioread('audio_corrompido.wav');`
`sound(x, fs);`
em que *x* representa o vetor de dados e *fs* a frequência de amostragem.

A figura a seguir apresenta as curvas de resposta em frequência⁶ de um filtro digital IIR Chebyshev de ordem $N = 13$, construído utilizando a Transformada Bilinear⁷. Os parâmetros do projeto são:

$$f_p = 5 \text{ kHz}, \quad f_s = 6 \text{ kHz}, \quad \alpha_p = 1 \text{ dB}, \quad \alpha_s = 60 \text{ dB}.$$

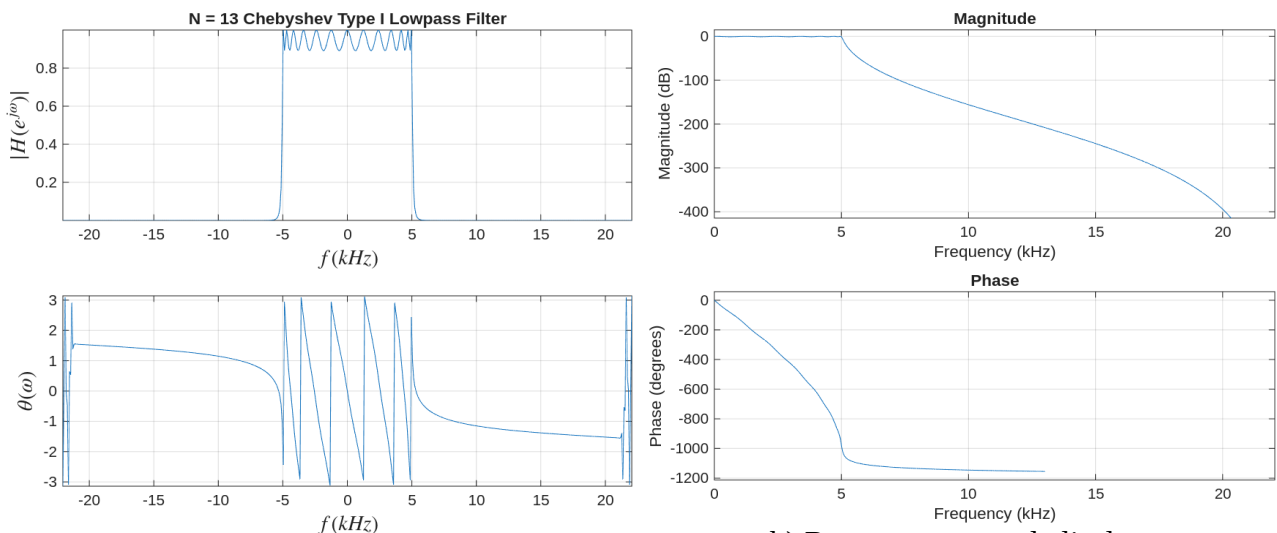
A equação de diferenças é dada pela expressão:

$$\begin{aligned} & y[n] - 10.77 y[n-1] + 54.93 y[n-2] - 175.4 y[n-3] + \dots - 0.5219 \\ & = \\ & 5.914e-10 x[n] + 7.688e-09 x[n-1] + 4.613e-08 x[n-2] + 1.691e-07 x[n-3] + \dots - 0.5219, \end{aligned}$$

que corresponde a uma resposta em frequência dada por:

$$H(e^{j\omega}) = \frac{5.914e-10 e^{j13\omega} + 7.688e-09 e^{j12\omega} + 4.613e-08 e^{j11\omega} + 1.691e-07 e^{j10\omega} + 4.228e-07 e^{j9\omega} + \dots - 0.5219}{e^{j13\omega} - 10.77 e^{j12\omega} + 54.93 e^{j11\omega} - 175.4 e^{j10\omega} + 390.7 e^{j9\omega} + \dots - 0.5219}.$$

Os coeficientes do filtro são fornecidos nos arquivos “coefs_num.txt” e “coefs_den.txt” em anexo.



a) Resposta em escala linear.

b) Resposta em escala lin-log.

Figura 7: resposta em frequência do filtro digital.

A resposta ao impulso⁸ $h[n]$ foi calculada para um comprimento de 1000 amostras, como visto na figura abaixo. Como esperado, a resposta ao impulso de um filtro IIR tem duração infinita. Entretanto, se este for causal e estável, ela deverá decair até se tornar desprezível dentro de um número finito de amostras, como observa-se.

⁶ No Matlab, veja a função:

`[resp, freq] = freqz(num, den, linspace(-fs/2, fs/2, 512), fs);`

⁷ OPPENHEIM, Alan V.; SCHAFER, Ronald W. Projeto de filtros IIR de tempo discreto a partir de filtros de tempo contínuo. In: Processamento em tempo discreto de sinais. 3ª ed. São Paulo: Pearson Education do Brasil, 2012. Seção 7.2, pág. 296.

⁸ No Matlab, veja a função

`[h, n] = impz(num, den, n_samples);`

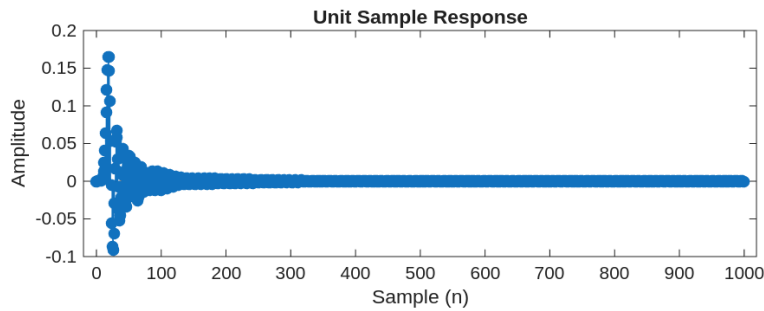


Figura 8: resposta ao impulso do filtro digital.

O sistema desenvolvido deverá realizar, no mínimo, as seguintes tarefas:

1. Carregamento de dados e apresentação de características básicas

- 1.1. Carregamento do arquivo “audio_corrompido.wav” e execução no sistema de áudio do computador visando a identificação do ruído.
- 1.2. Geração do gráfico da forma de onda em função do tempo, em s, similar ao mostrado na Figura 5a.
- 1.3. Geração dos espectros de amplitude $|X(e^{j\omega})| \times f(kHz)$ e fase $\theta(\omega) \times f(kHz)$, em kHz, similar ao mostrado na Figura 5b.
- 1.4. Carregamento dos coeficientes do filtro dos arquivos “coefs_num.txt” e “coefs_den.txt”.
- 1.5. Apresentação das respostas de magnitude $|H(e^{j\omega})| \times f(kHz)$ e fase $\theta(\omega) \times f(kHz)$ do filtro, na faixa de frequências correspondente ao espectro do sinal, similar à Figura 7.
- 1.6. Apresentação da resposta ao impulso $h[n] \times n$ com 1000 amostras, similar à vista na figura 8.

2. Implementação de funções para filtragem de 3 formas distintas:

- 2.1. Utilizando a equação de diferenças, tal como na eq. 1.
 - a) Esta função deverá ser chamada *amostra-por-amostra*, simulando o comportamento de um conversor AD, e, para cada amostra, deverá retornar a saída correspondente. Um possível protótipo em linguagem C teria o seguinte formato:

float y filtragemPorEqDif(float x);

- 2.2. Utilizando a convolução com a resposta ao impulso, tal como na eq. 2.
 - a) Faça uma truncagem da resposta ao impulso calculada no item 1.6. A truncagem deverá eliminar amostras *da cauda* menores do que 1% do valor de pico (obs.: é importante manter as amostras iniciais, mesmo que menores do que 1% do pico).
 - b) Apresentação da resposta ao impulso truncada $h_{trunc}[n]$ e do número de amostras correspondente Nh .
 - c) Filtragem por convolução circular. A função será chamada uma única vez, receberá um bloco de dados de tamanho Nx , calculará a saída e retornará o sinal filtrado $y[n]$ de tamanho $Ny = Nx + Nh - 1$. Um protótipo em C teria o seguinte formato:

float y filtragemPorConv(float* x, Nx);*

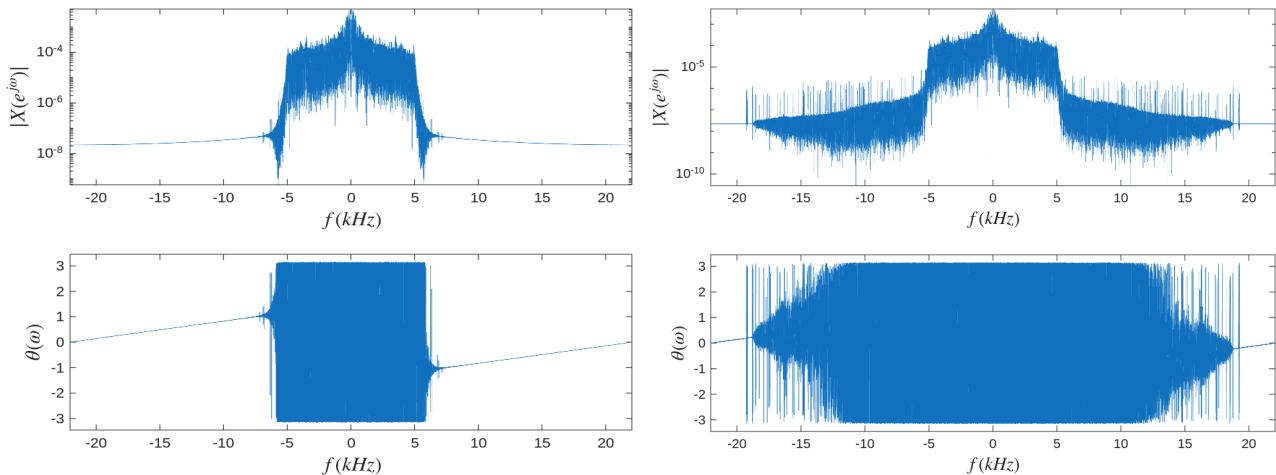
- 2.3. Utilizando a propriedade da multiplicação da Transformada de Fourier, tal como na eq. 3.
 - a) Ainda utilizando a resposta ao impulso truncada, calculada no item 2.2a, implemente a filtragem utilizando a propriedade da multiplicação da FFT.

- b) A função será chamada uma única vez, receberá um bloco de dados de tamanho N_x , calculará a saída e retornará o sinal filtrado $y[n]$ de tamanho $N_y = N_x + N_h - 1$. Um protótipo em C seria:

```
float* y = filtragemPorFFT(float* x, Nx);
```

3. Filtragem do sinal:

- 3.1. Filtragem do sinal de áudio utilizando as três formas implementadas.
- 3.2. Apresentação dos sinais filtrados nos domínios do tempo e da frequência. Um exemplo de resultado é mostrado na figura a seguir.



a) Método da equação de diferenças, em escala lin-log.

b) Método da convolução, em escala lin-log.

Figura 9: espectros de frequência do sinal filtrado.

- 3.3. Execução do sinal filtrado no sistema de áudio do computador.
- 3.4. Análise dos efeitos da filtragem linear sobre o sinal (ex.: qualidade do resultado, eficiência na eliminação do ruído, efeitos da truncagem da resposta ao impulso, distorções resultantes, desempenho computacional, etc.)

4. Bônus:

Como você deve ter observado, os algoritmos implementados são muito ineficientes quando um dos blocos é muito maior do que o outro (neste caso, o sinal de áudio é muito maior do que a resposta ao impulso truncada):

- a) a filtragem por equação de diferenças impõe um grande número de chamadas à função, o que é ineficiente do ponto de vista de tempo de execução;
- b) a filtragem por convolução circular, tanto no domínio do tempo quanto na frequência, é muito ineficiente do ponto de vista de alocação de memória por causa dos encheimentos com zeros.

O desempenho pode ser melhorado implementando-se uma das versões da convolução circular, mostradas na figura 4. Assim, propõe-se como bônus:

- 4.1. Implementação do algoritmo “overlap-add” utilizando as funções *filtragemPorConv* e *filtragemPorFFT* desenvolvidas anteriormente.
- 4.2. Filtragem do sinal de áudio utilizando blocos de tamanho $N_x = N_h$.
- 4.3. Apresentação dos resultados nos domínios do tempo e da frequência.
- 4.4. Execução do sinal filtrado no sistema de áudio do computador.

- 4.5. Análise comparativa com os métodos anteriores (ex.: qualidade do resultado, eficiência na eliminação do ruído, efeitos da truncagem da resposta ao impulso, distorções resultantes, desempenho computacional, etc.).