Filtros digitais

DESIGN DE FILTROS DIGITAIS

Introdução

- Filtros digitais são uma classe particular de sistemas LTI

- Filtro seletor de frequência sugere algo que seleciona uma faixa de frequência e rejeita outras, mas filtros são mais que isso, qualquer **sistema que modifique frequências relativamente a outras** é chamado **filtro**

$$y[n] = \frac{1}{a_0}\left(\sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]\right)$$

H(z)

IIR

- H(z) aproximada por uma razão de polinômios

FIR

- H(z) aproximada por um polinômio

Introdução

- Filtros digitais são usados com duas finalidades principais
  - Separar sinais que foram combinados
  - Restaurar sinais que foram distorcidos de alguma maneira

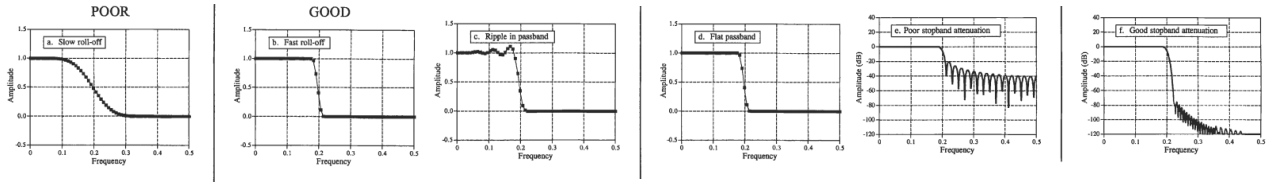
Introdução

- São muito importantes em PDS, sua extraordinária performance tornou o PDS tão popular

- Por que utilizar filtros digitais ao invés de analógicos?
  - Filtros analógicos são baratos e tem uma ampla faixa e atuação em amplitude e frequência
  - **Filtros digitais tem performances milhares de vezes melhores que um analógico**!

Como filtrar um sinal?

- Depende da aplicação
- Características dos sinais de entrada e saída



Especificação das propriedades do sistema

- Forma, frequência de corte com base no filtro ideal

Gerar o núcleo do filtro

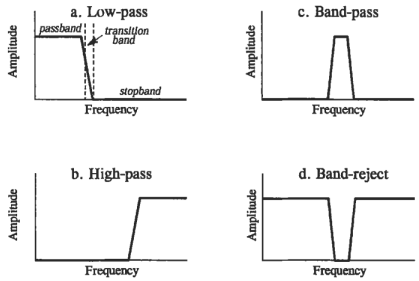
- Aproximação das propriedades por meio de uma sequência causal de tempo discreto que aproxime a RF desejada

Avaliar a RF do filtro gerado em 2

- H(z) aproxima corretamente o filtro ideal
- Parâmetros que determinam uma boa H(z):
  - Roll-off
  - Ripple
  - Atenuação

Filtrar o sinal

- Depende da tecnologia



$$y[n] = \frac{1}{a_0}\left(\sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]\right)$$

h[n]

H(z)

h[n]

Parâmetros para o domínio da frequência

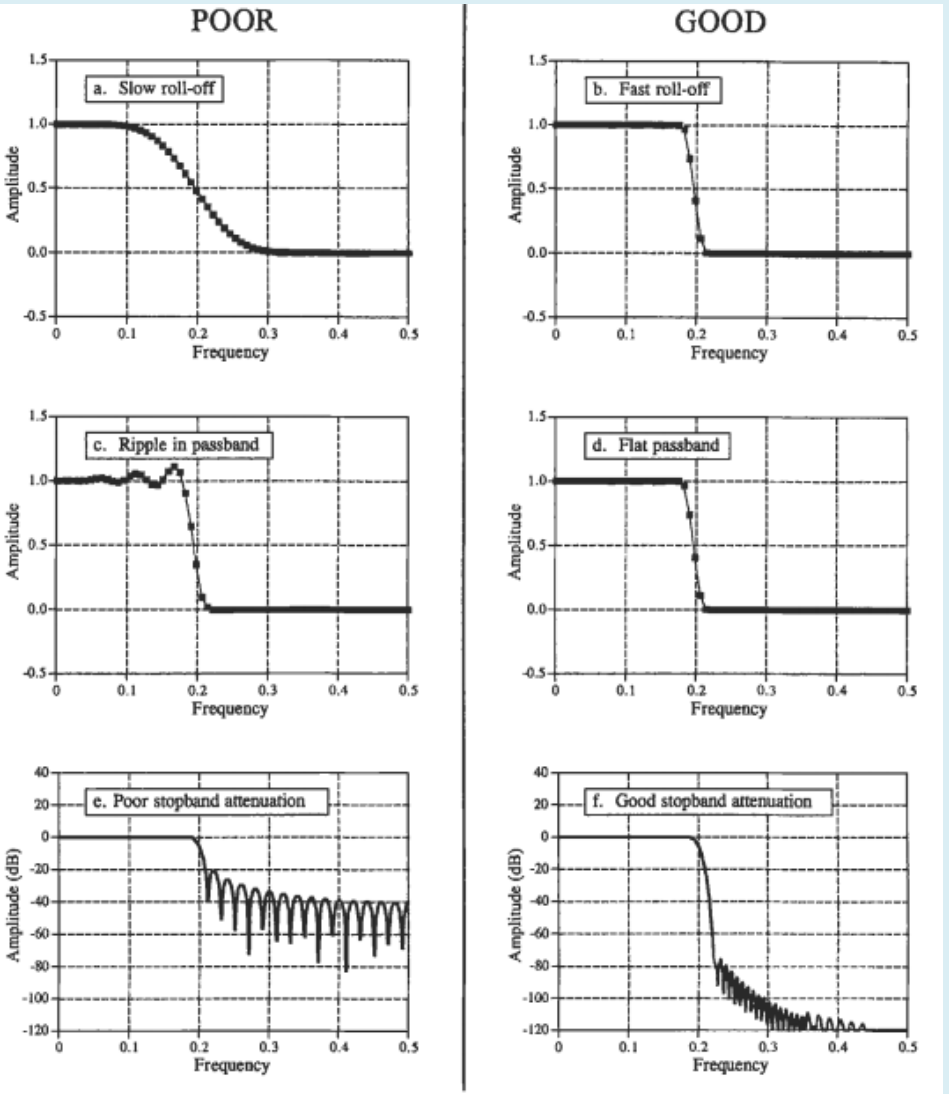
- ***Roll-off* rápido**: para separar duas frequências próximas
- Não pode haver ondulação (**ripple**) no passband
- **Boa atenuação** no stopband
- Por que não há nada sobre a fase?
  - A fase não é tão importante em aplicações no domínio da frequência

Parâmetros para o domínio da frequência

Roll-off

ripple

Atenuação



Parâmetros para o domínio da frequência

- A proposta destes filtros é permitir que **algumas frequências passem inalteradas e outras sejam eliminadas**
  - *Passband*: freqüências que passam
  - *Stopband*: freqüências que são bloqueadas
  - Banda de transição: entre as 2 anteriores
    - *Roll-off* rapido: banda de transição pequena
    - Freqüência de corte (*cutoff*): divisão entre a banda de passagem e a de transição

Design de filtros digitais

Resposta Impulsiva (núcleo do filtro)

- A maneira mais simples de **implementar filtros** digitais é **convoluir** o **sinal de entrada** com a **resposta impulsiva do filtro**

  – Quando a RI é usada com esta finalidade é chamada de **núcleo do filtro**

FIR

- Em um **filtro implementado por convolução**, cada amostra da saída é calculada "pesando-se" as amostras de entradas e adicionando-as

Design de filtros digitais

Recursão

- Filtros recursivos são uma extensão dos de convolução, só que eles **utilizam informação das saídas anteriores** além de pontos da **entrada, ao invés de utilizarem o núcleo, utilizam uma serie de coeficientes recursivos**

- É importante salientar que **TODOS os filtros tem uma RI, mesmo que ela não tenha sido usada para implementar o filtro**

  – Para conhecer a RI de um filtro recursivo, alimente-o com um sinal de entrada e observe a saída

Design de filtros digitais

Recursão

- A RI de filtros recursivos é composta por senóides que decaem exponencialmente em amplitude

IIR

- Isso faz com que tenham RI infinitamente longa

IIR x FIR

Recursão

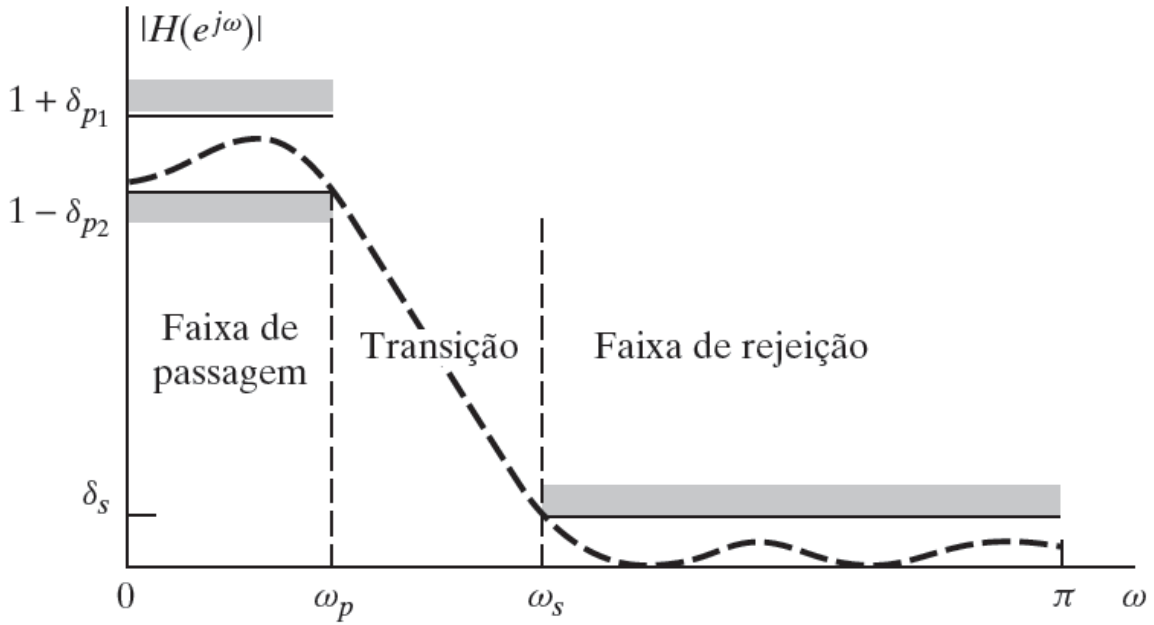
- Por causa dessa característica, filtros recursivos são chamados de Filtros de Resposta Impulsiva Infinita – **IIR** (*Infinite Impulse Response*)

Convolução

- Por outro lado, os filtros obtidos por convolução são chamados de Filtros de Resposta Impulsiva Finita – **FIR** (*Finite Impulse Response*)

Especificações do filtro

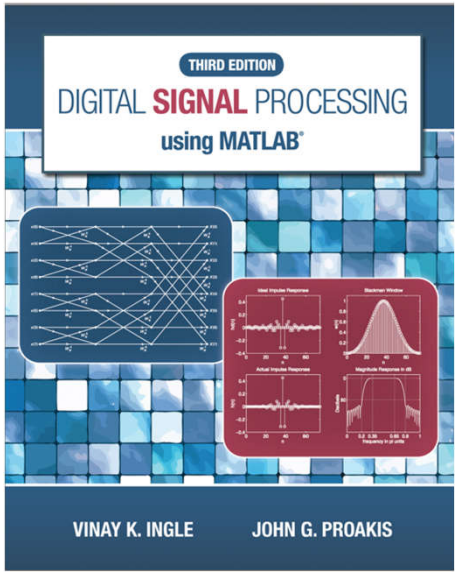
- Diagrama de tolerâncias de um filtro passa-baixas.



Design de filtros

MATLAB

- Capitulo 7 – FIR
- Capitulo 8 - IIR

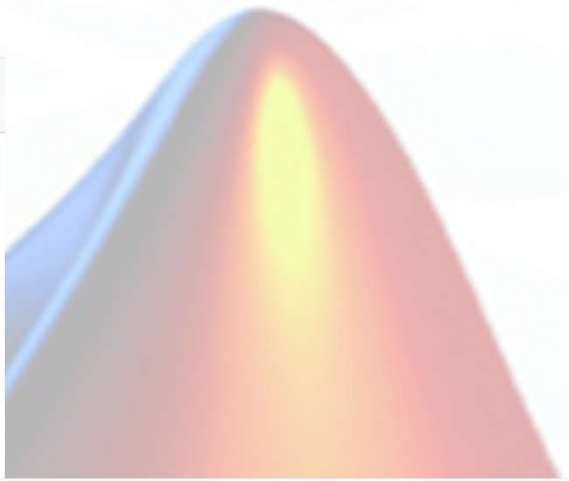


Teoria

- Capitulo 7

IIR Filters

butter	Butterworth filter design
buttord	Butterworth filter order and cutoff frequency
cheby1	Chebyshev Type I filter design
cheb1ord	Chebyshev Type I filter order
cheby2	Chebyshev Type II filter design
cheb2ord	Chebyshev Type II filter order
designfilt	Design digital filters
ellip	Elliptic filter design
ellipord	Minimum order for elliptic filters
polyscale	Scale roots of polynomial
polystab	Stabilize polynomial
yulewalk	Recursive digital filter design



FIR Filters

cfirpm	Complex and nonlinear-phase equiripple FIR filter design
designfilt	Design digital filters
fir1	Window-based FIR filter design
fir2	Frequency sampling-based FIR filter design
fircls	Constrained-least-squares FIR multiband filter design
fircls1	Constrained-least-squares linear-phase FIR lowpass and highpass filter design
firls	Least-squares linear-phase FIR filter design
firpm	Parks-McClellan optimal FIR filter design
firpmord	Parks-McClellan optimal FIR filter order estimation
gaussdesign	Gaussian FIR pulse-shaping filter design
intfilt	Interpolation FIR filter design
kaiserord	Kaiser window FIR filter design estimation parameters
maxflat	Generalized digital Butterworth filter design
rcosdesign	Raised cosine FIR pulse-shaping filter design
sgolay	Savitzky-Golay filter design

Filtros digitais

DESIGN DE FILTROS DIGITAIS

EXEMPLOS FIR

firls

Description

`b = firls(n,f,a)` returns row vector `b` containing the `n+1` coefficients of an order-`n` FIR filter. The frequency and amplitude characteristics of the resulting filter match those given by vectors `f` and `a`. example

`b = firls(n,f,a,w)` uses `w` to weigh the frequency bins. example

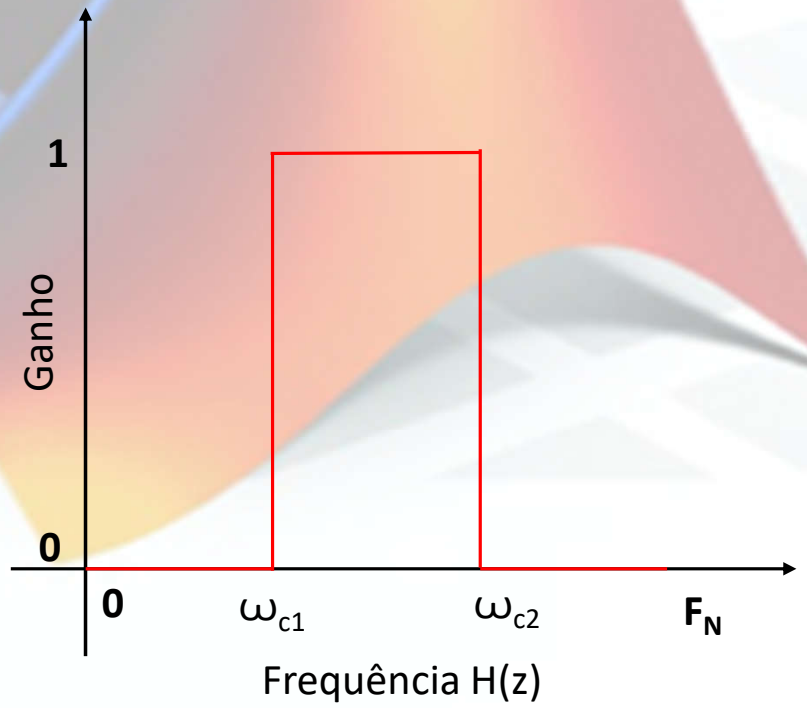
`b = firls(__,ftype)` designs antisymmetric (odd) filters, where `ftype` specifies the filter as a differentiator or Hilbert transformer. You can use `ftype` with any of the previous input syntaxes. example

Ordem do filtro

b=firls(n,f,a)

Vetor de amplitudes (ganho)

Vetor de frequências



MATLAB

firls

Description

`b = firls(n,f,a)` returns row vector `b` containing the `n+1` coefficients of an order-`n` FIR filter. The frequency and amplitude characteristics of the resulting filter match those given by vectors `f` and `a`. example

`b = firls(n,f,a,w)` uses `w` to weigh the frequency bins. example

`b = firls(__,ftype)` designs antisymmetric (odd) filters, where `ftype` specifies the filter as a differentiator or Hilbert transformer. You can use `ftype` with any of the previous input syntaxes. example

Ordem do filtro

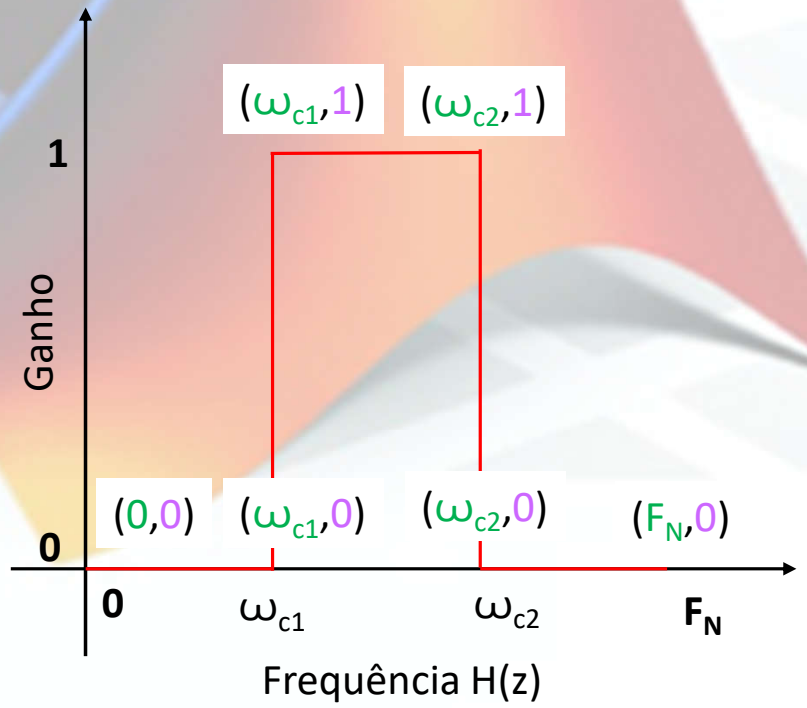
b=firls(n,f,a)

Vetor de amplitudes (ganho)

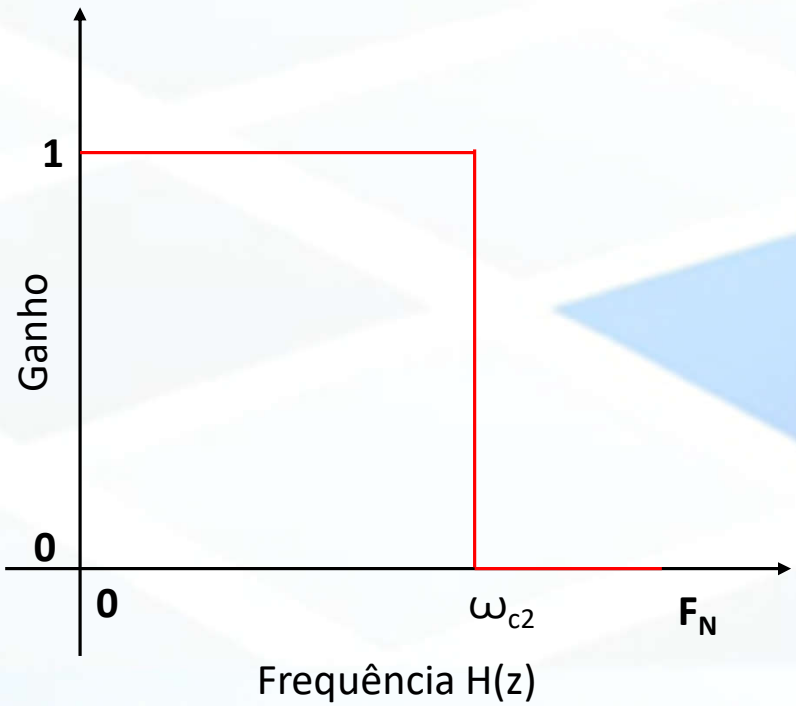
Vetor de frequências

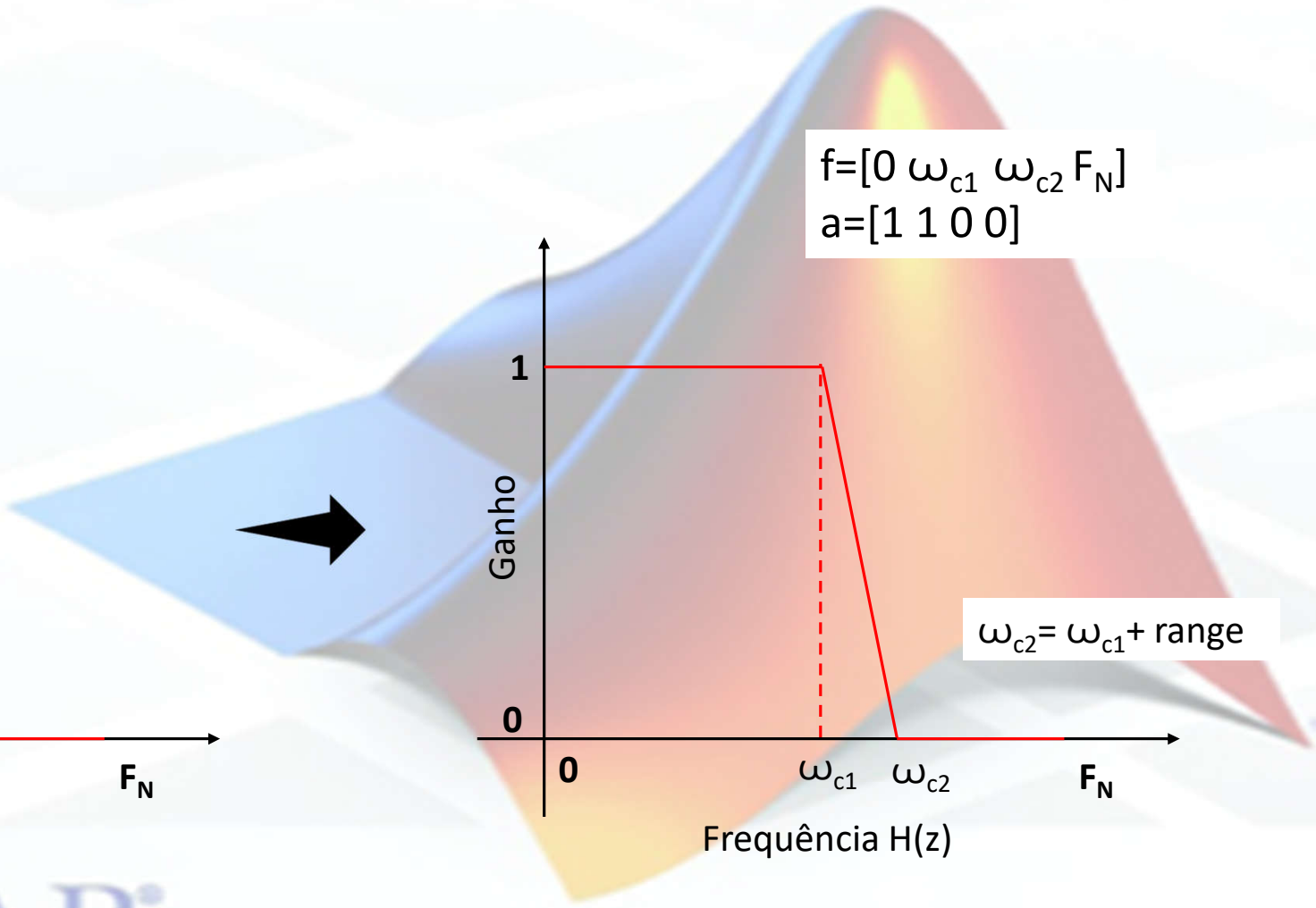
f=[0 ω_{c1} ω_{c1} ω_{c2} ω_{c2} F_N]

a=[0 0 1 1 0 0]



Filtro ideal





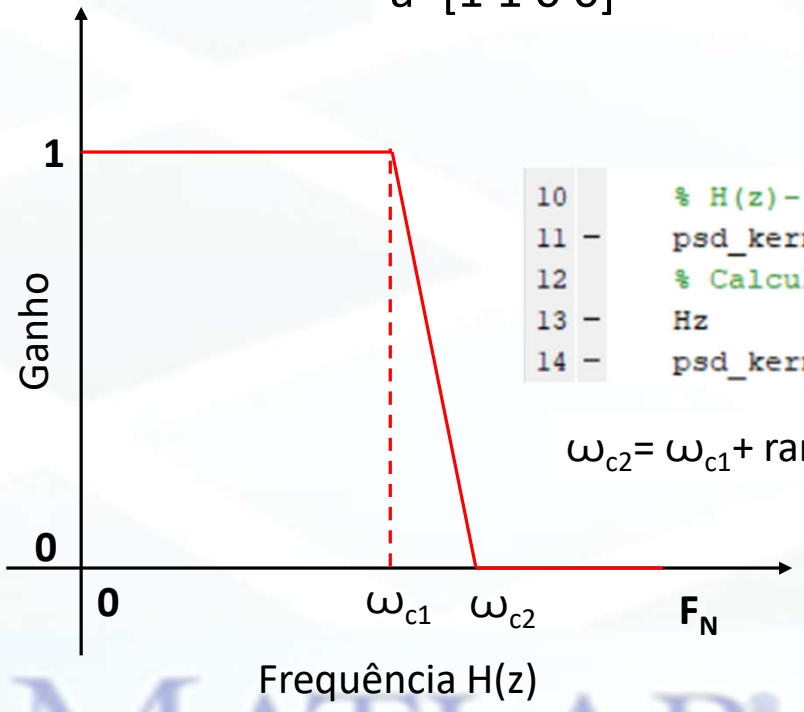
f=[0 ω_{c1} ω_{c2} F_N]
a=[1 1 0 0]

ω_{c2}= ω_{c1}+ range

MATLAB®

firls

f=[0 ω_{c1} ω_{c2} F_N]
a=[1 1 0 0]




```
1 -    clear
2 -    Fs    = 1024; % Hz
3 -    nyquist = Fs/2;
4 -    f   = [0 0.25 0.3 1];
5 -    a= [1 1 0 0];
6 -    ordem=50;
7      % filter kernel
8 -    kern = firls(ordem,f,a);
```

```
10     % H(z)- PSD do kernel
11 -   psd_kern = abs(fft(kern)).^2;
12     % Calcula o vetor de frequencias e exibe apenas a parte positiva (simetria)
13 -   Hz       = linspace(0,Fs/2,floor(length(kern)/2)+1);
14 -   psd_kern = psd_kern(1:length(Hz));
```

ω_{c2}= ω_{c1}+ range

firls

```
16 -     figure(1);
17 -     subplot(1,3,1);plot(kern)
18 -     xlabel('Pontos (tempo)')
19 -     title('Kernel (firls)')
20 -     axis square
```

```
33       % H(z) em escala logaritmica
34 -     subplot(1,3,3), hold on
35 -     plot(Hz,10*log10(psd_kern),'ks-')
36 -     xlabel('Frequencia (Hz)'), ylabel('Ganho (dB)')
37 -     title('H(z) (firls)')
38 -     axis square
```

```
22       % H(z) (PSD)
23 -     subplot(1,3,2), hold on
24 -     plot(Hz,psd_kern,'ks-','markerfacecolor','w')
25 -     plot(f*nyquist,a,'ro-','markerfacecolor','w')
26 -     axis square
27       % titulo, legenda
28 -     xlabel('Frequencia (Hz)'), ylabel('Ganho')
29 -     legend({'Projeto';'Ideal'})
30 -     title('H(z) (firls)')
31 -     axis square
```

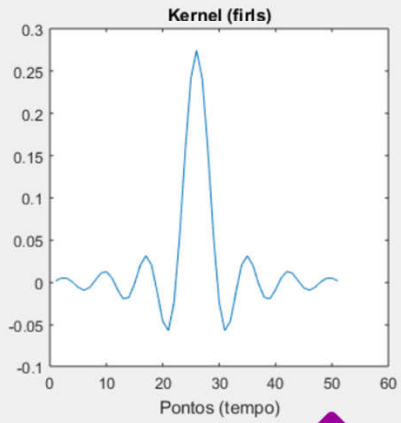
firls

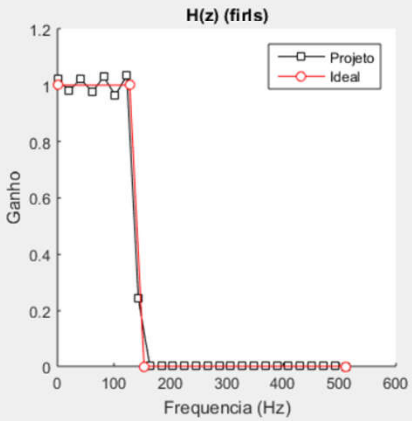
```
16 -      f
17 -      s
18 -      x
19 -      t
20 -      a

22        %
23 -      su
24 -      pl
25 -      pl
26 -      ax
27        %
28 -      xl
29 -      le
30 -      ti
31 -      axis square
```

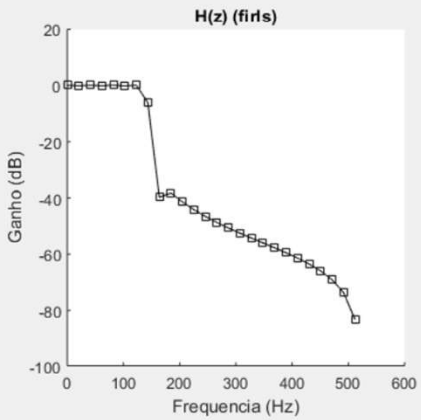
```
nho (dB)')
```

Figure 1

File Edit View Insert Tools Desktop Window Help

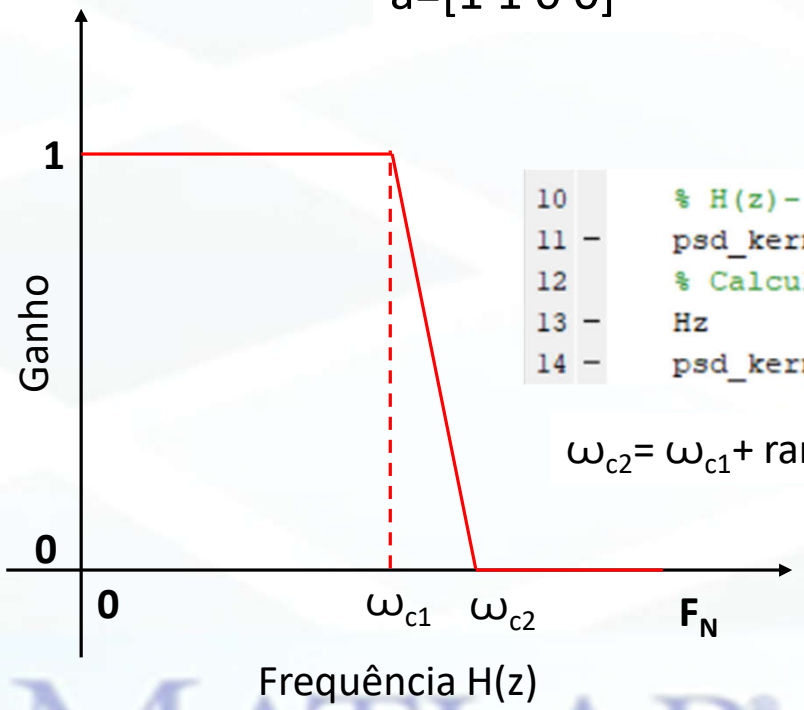






Filtro ideal

f=[0 ω_{c1} ω_{c2} F_N]
a=[1 1 0 0]




```
1 -    clear
2 -    Fs    = 1024; % Hz
3 -    nyquist = Fs/2;
4 -    f   = [0 0.25 0.3 1];
5 -    a= [1 1 0 0];
6 -    ordem=50;
7      % filter kernel
8 -    kern = firls(ordem,f,a);
```

```
10     % H(z)- PSD do kernel
11 -   psd_kern = abs(fft(kern)).^2;
12     % Calcula o vetor de frequencias e exibe apenas a parte positiva (simetria)
13 -   Hz       = linspace(0,Fs/2,floor(length(kern)/2)+1);
14 -   psd_kern = psd_kern(1:length(Hz));
```

ω_{c2}= ω_{c1}+ range

```
1 -    clear
```

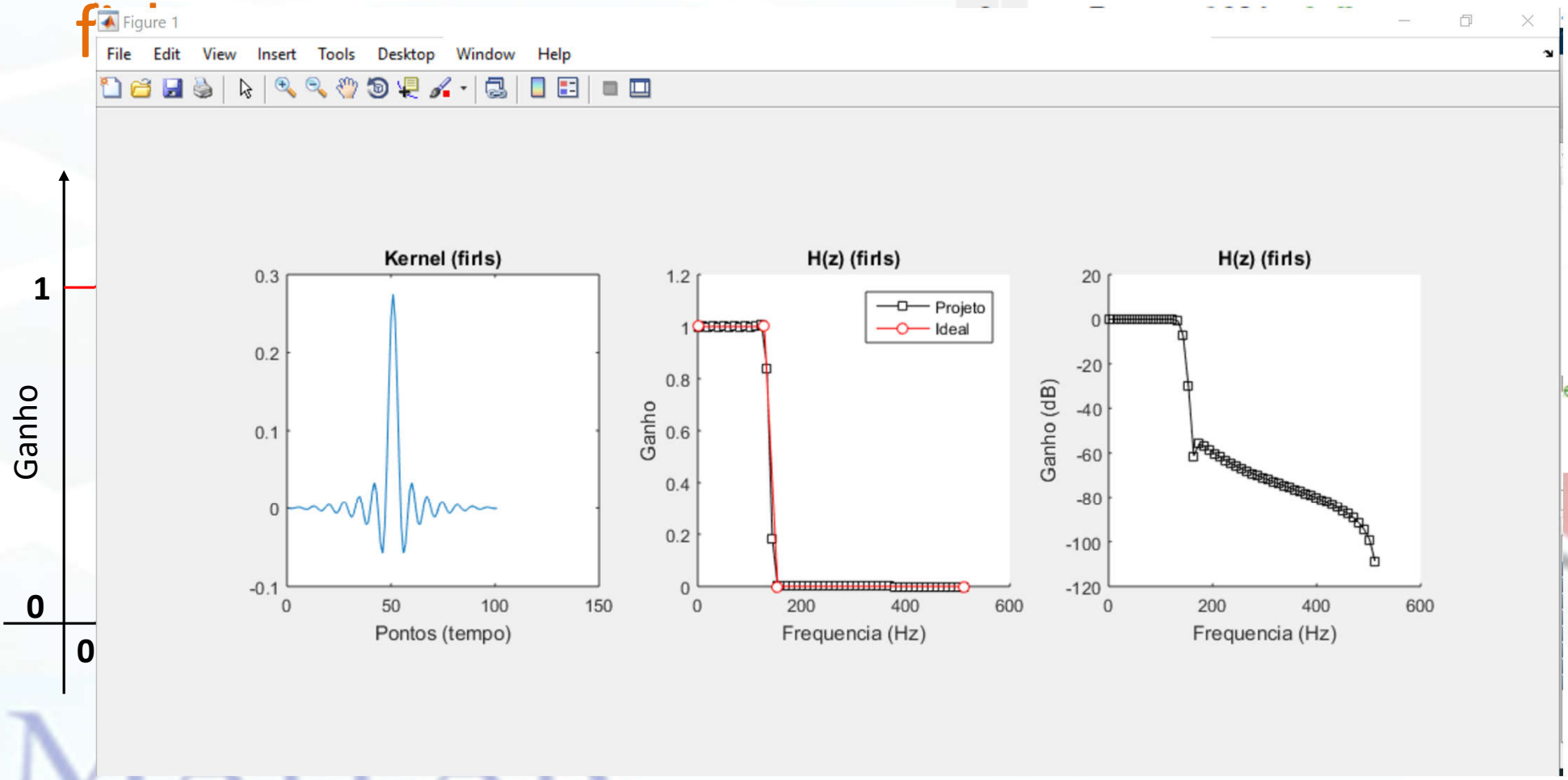


Figure 1
File Edit View Insert Tools Desktop Window Help
Kernel (firls)
Pontos (tempo)
H(z) (firls)
Ganho
Frequencia (Hz)
Projeto
Ideal
H(z) (firls)
Ganho (dB)
Frequencia (Hz)

1
Ganho
0
0

```
1 -    clear
```

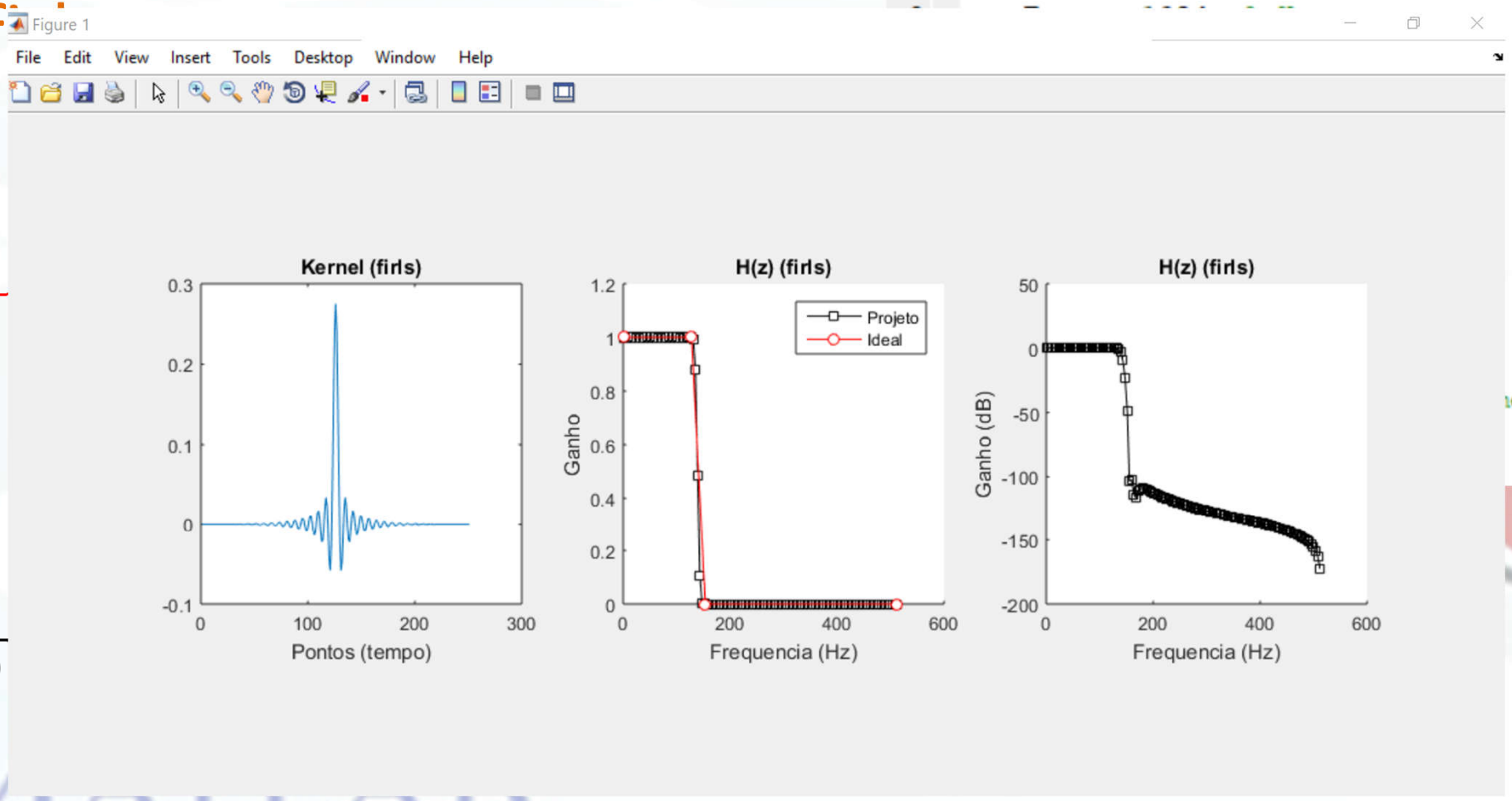


Figure 1
File Edit View Insert Tools Desktop Window Help
Kernel (firls)
Pontos (tempo)
H(z) (firls)
Ganho
Frequencia (Hz)
Projeto
Ideal
H(z) (firls)
Ganho (dB)
Frequencia (Hz)


1
Ganho
0
0

```
1 -    clear
```

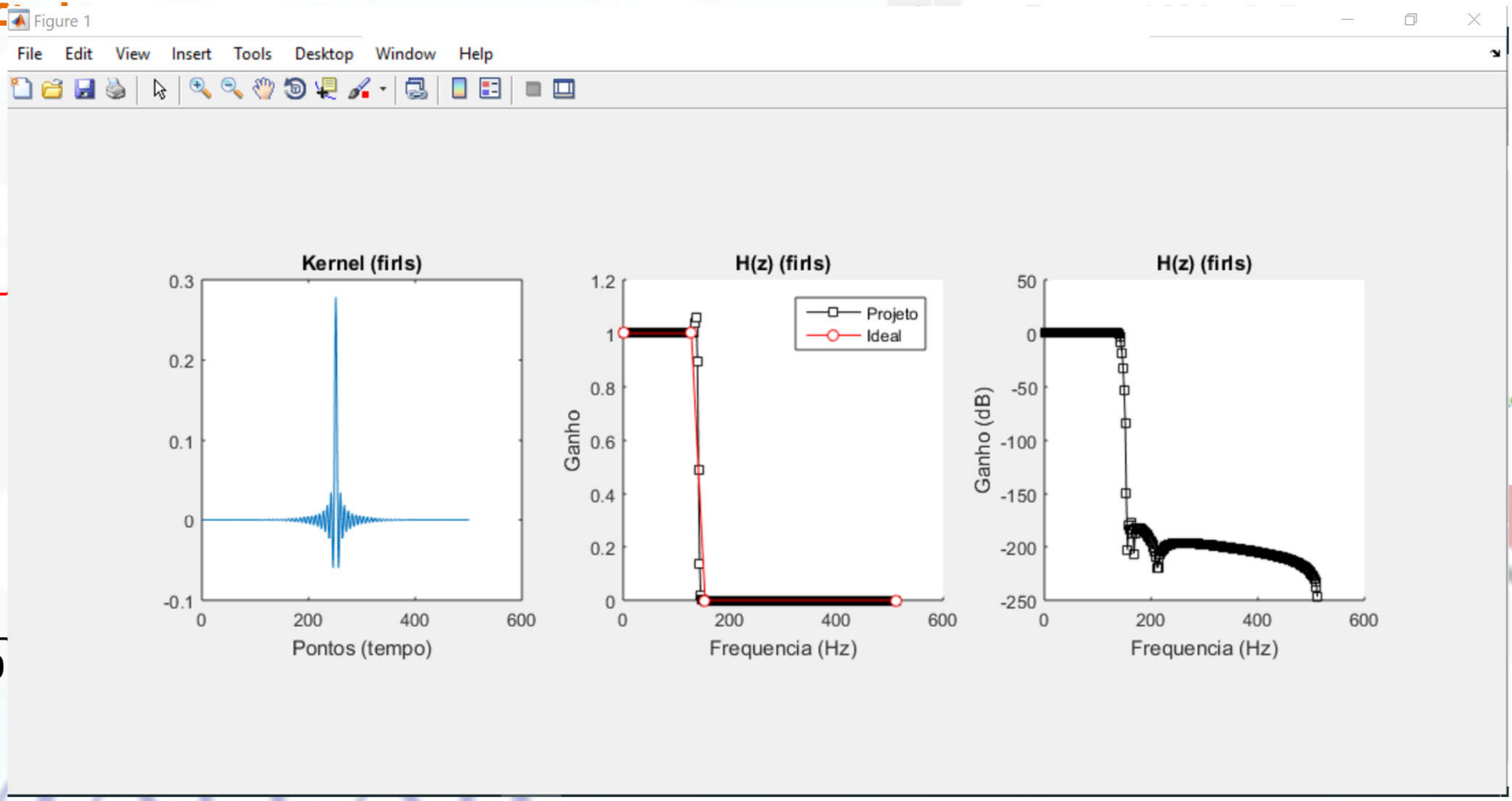


Figure 1
File Edit View Insert Tools Desktop Window Help
Kernel (firls)
Pontos (tempo)
H(z) (firls)
Ganho
Frequencia (Hz)
Projeto
Ideal
H(z) (firls)
Ganho (dB)
Frequencia (Hz)


1
Ganho
0
0

fir1

Description

`b = fir1(n,Wn)` uses a Hamming window to design an `n`th-order lowpass, bandpass, or multiband FIR filter with linear phase. The filter type depends on the number of elements of `Wn`. example

`b = fir1(n,Wn,ftype)` designs a lowpass, highpass, bandpass, bandstop, or multiband filter, depending on the value of `ftype` and the number of elements of `Wn`. example

FIR Highpass Filter

Load `chirp.mat`. The file contains a signal, `y`, that has most of its power above `Fs/4`, or half the Nyquist frequency. The sample rate is 8192 Hz.

View MATLAB Command

Design a 34th-order FIR highpass filter to attenuate the components of the signal below `Fs/4`. Use a cutoff frequency of 0.48 and a Chebyshev window with 30 dB of ripple.

```
load chirp

t = (0:length(y)-1)/Fs;

bhi = fir1(34,0.48,'high',chebwin(35,30));
freqz(bhi,1)
```

fir1

Description

`b = fir1(n,Wn)` uses a Hamming window to design an nth-order lowpass, bandpass, or multiband FIR filter with linear phase. The filter type depends on the number of elements of `Wn`. example

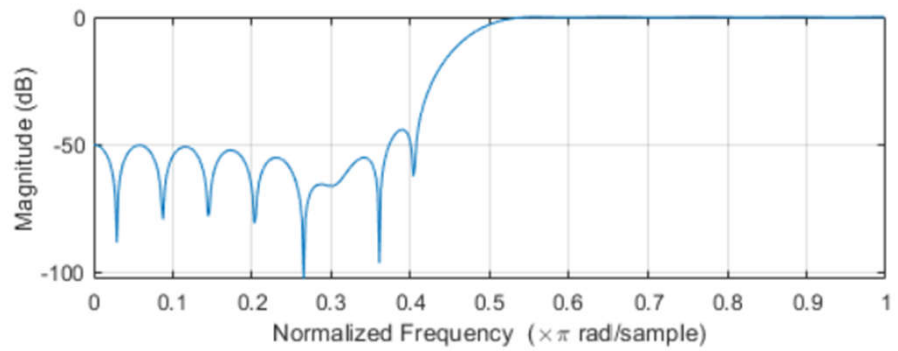
`b = fir1(n,Wn,ftype)` designs a lowpass, highpass, bandpass, bandstop, or multiba ample

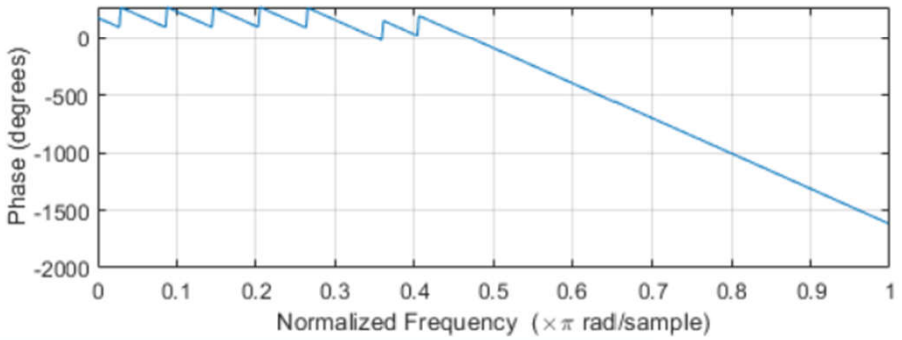
FIR Highpass Filter

Load `chirp.mat`. The file contains a signal, `y`, that has most of its power above `Fs/` rate is 8192 Hz.

Design a 34th-order FIR highpass filter to attenuate the components of the signal be and a Chebyshev window with 30 dB of ripple.

```
load chirp

t = (0:length(y)-1)/Fs;

bhi = fir1(34,0.48,'high',chebwin(35,30));
freqz(bhi,1)
```

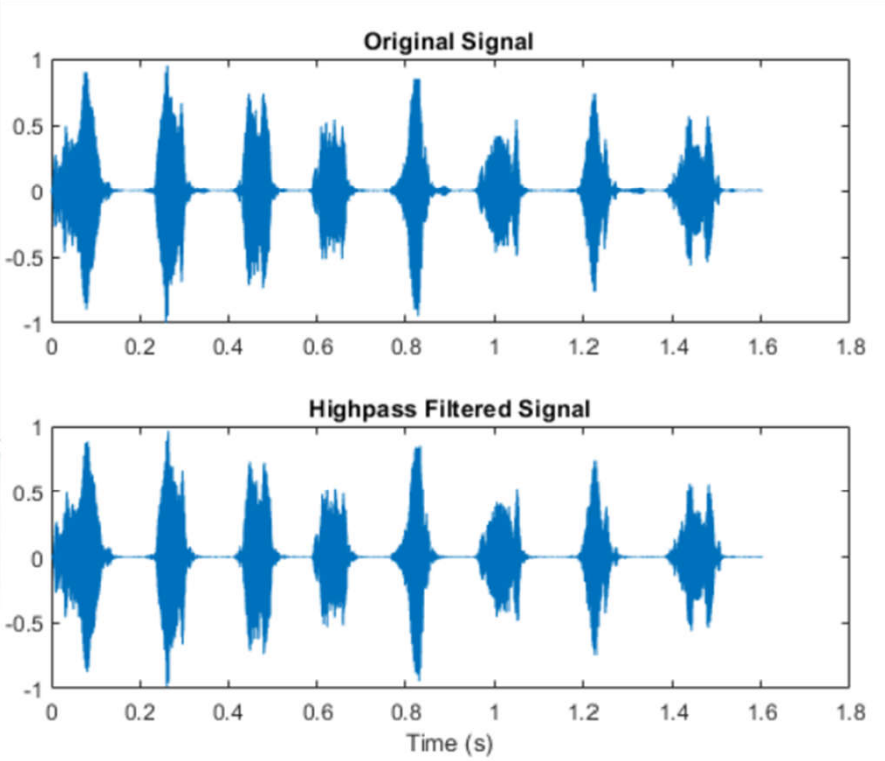




fir1

Filter the signal. Display the original and highpass-filtered signals. Use the same *y*-axis scale for both plots.

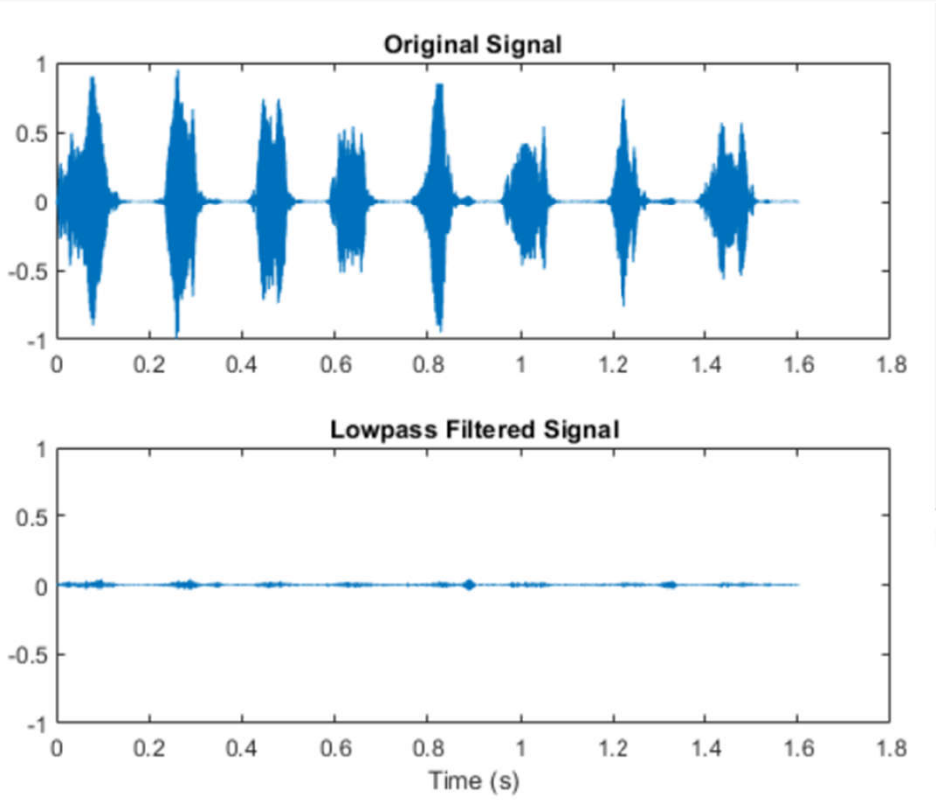
```
outhi = filter(bhi,1,y);

subplot(2,1,1)
plot(t,y)
title('Original Signal')
ys = ylim;

subplot(2,1,2)
plot(t,outhi)
title('Highpass Filtered Signal')
xlabel('Time (s)')
ylim(ys)
```



fir1

Design a lowpass filter with the same specifications. Filter the signal and compare the result to the original. Use the same *y*-axis scale for both plots.

```
blo = fir1(34,0.48,chebwin(35,30));

outlo = filter(blo,1,y);

subplot(2,1,1)
plot(t,y)
title('Original Signal')
ys = ylim;

subplot(2,1,2)
plot(t,outlo)
title('Lowpass Filtered Signal')
xlabel('Time (s)')
ylim(ys)
```



DESIGN DE FILTROS DIGITAIS

EXEMPLO IIR

butter

Description

`[b,a] = butter(n,Wn)` returns the transfer function coefficients of an `n`th-order lowpass digital Butterworth filter with normalized cutoff frequency `Wn`. example

`[b,a] = butter(n,Wn,ftype)` designs a lowpass, highpass, bandpass, or bandstop Butterworth filter, depending on the value of `ftype` and the number of elements of `Wn`. The resulting bandpass and bandstop designs are of order `2n`. example

Note: See Limitations for information about numerical issues that affect forming the transfer function.

`[z,p,k] = butter( __ )` designs a lowpass, highpass, bandpass, or bandstop digital Butterworth filter and returns its zeros, poles, and gain. This syntax can include any of the input arguments in previous syntaxes. example

`[A,B,C,D] = butter( __ )` designs a lowpass, highpass, bandpass, or bandstop digital Butterworth filter and returns the matrices that specify its state-space representation. example

`[ __ ] = butter( __ ,'s')` designs a lowpass, highpass, bandpass, or bandstop analog Butterworth filter with cutoff angular frequency `Wn`. example

Examples

collapse all

MATLAB®

butter

```
clear
Fs    = 1024; % Hz
nyquist = Fs/2;
f   = [20 45]/nyquist;
ordem=4;

% coeficientes do filtro (baseados na equação de diferenças)
[bn,an] = butter(ordem,f);

% H(z)- PSD do kernel
psd_bn = abs(fft(bn)).^2;
% Calcula o vetor de frequencias e exibe apenas a parte positiva (simetria)
Hz      = linspace(0,Fs/2,floor(length(bn)/2)+1);
psd_bn = psd_bn(1:length(Hz));
```

```
figure(1);
subplot(2,2,1);hold on; plot(bn*1e5, 'ks-','linew',1,'markersize',4,'markerfacecolor','w');
plot(an,'rs-','linew',1,'markersize',4,'markerfacecolor','w')
xlabel('Pontos (tempo)');ylabel('Coef. do filtro')
title('Coeficientes do filtro no dom do tempo')
legend({'bn';'an'})
```

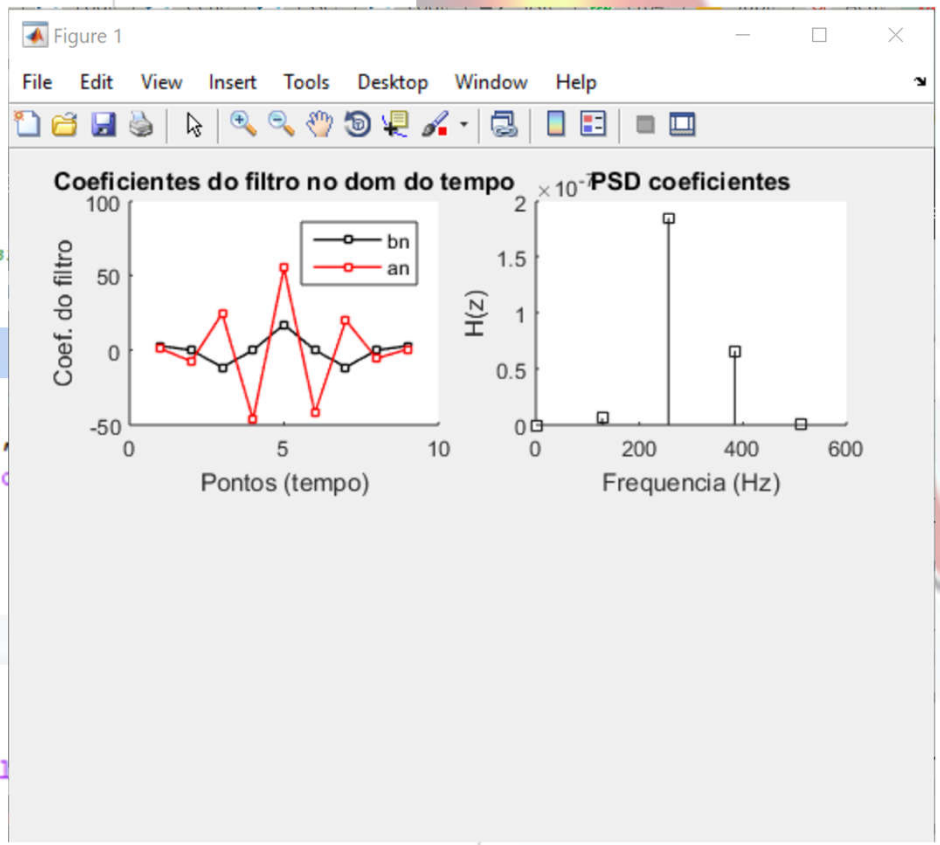
```
% H(z) (PSD)
subplot(2,2,2), hold on
stem(Hz,psd_bn(1:length(Hz)),'ks-','markerfacecolor','w')
xlabel('Frequencia (Hz)'), ylabel('H(z)')
title('PSD coeficientes')
```

butter

```
clear
Fs    = 1024; % Hz
nyquist = Fs/2;
f   = [20 45]/nyquist;
ordem=4;

% coeficientes do filtro (baseados na equação de diferenças)
[bn,an] = butter(ordem,f);

% H(z)- PSD do kernel
psd_bn = abs(fft(bn)).^2;
% Calcula o vetor de frequencias e exibe apenas a parte positiva (s
Hz       = linspace(0,Fs/2,floor(length(bn)/2)+1);
psd_bn = psd_bn(1:length(Hz));
```

```
figure(1);
subplot(2,2,1);hold on; plot(bn*1e5, 'ks-','linew',
plot(an,'rs-','linew',1,'markersize',4,'markerfaced
xlabel('Pontos (tempo)');ylabel('Coef. do filtro')
title('Coeficientes do filtro no dom do tempo')
legend({'bn';'an'})
```

```
% H(z) (PSD)
subplot(2,2,2), hold on
stem(Hz,psd_bn(1:length(Hz)),'ks-','markerfacecol
xlabel('Frequencia (Hz)'), ylabel('H(z)')
title('PSD coeficientes')
```

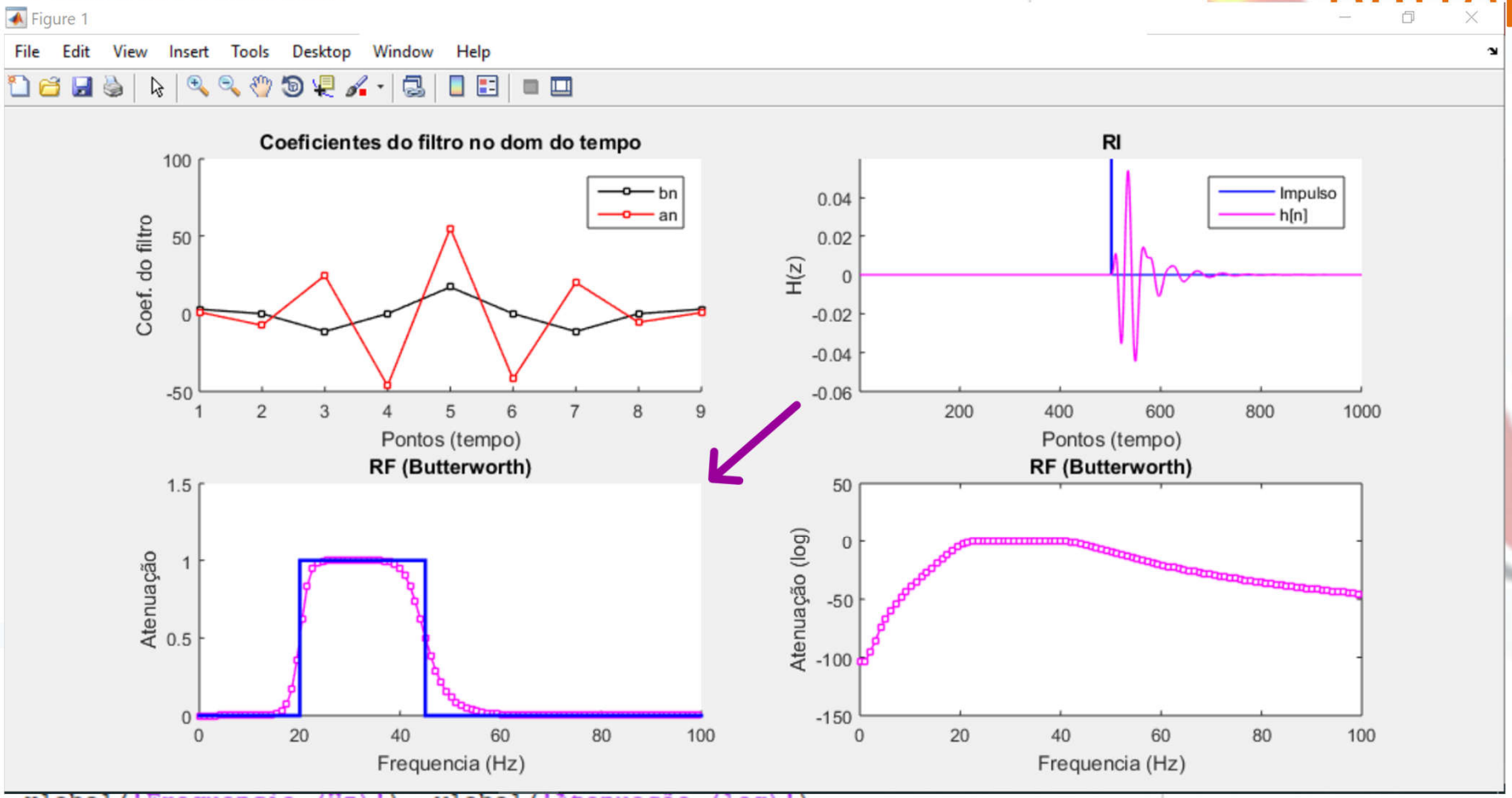


butter

```
29      % Obter a RI
30 -    impulso = impseq(501,0,1000);
31 -    hn=filter(bn,an,impulso);
32
33      %obter H(z) - PSD de hn
34 -    psd_imp = abs(fft(hn)).^2;
35 -    Hz = linspace(0,nyquist,floor(length(impulso)/2)+1);
36
37 -    subplot(2,2,2), cla, hold on
38 -    plot(impulso,'b','linew',1)
39 -    plot(hn,'m','linew',1)
40 -    set(gca,'xlim',[1 length(impulso)],'ylim',[-1 1]*.06)
41 -    legend({'Impulso';'h[n]'})
42 -    xlabel('Pontos (tempo)')
43 -    title('RI')
44
45      % H(z)
46 -    subplot(2,2,3), hold on
47 -    plot(Hz,psd_imp(1:length(Hz)),'ms-','linew',1,'markerfacecolor','w','markersize',4)
48 -    plot([0 f(1) f(1) f(2) f(2) 1]*nyquist,[0 0 1 1 0 0],'b','linew',2)
49 -    set(gca,'xlim',[0 100])
50 -    xlabel('Frequencia (Hz)'), ylabel('Atenuação')
51 -    title('RF (Butterworth)')
```

```
53      % H(z) em escala logaritmica
54 -    subplot(2,2,4)
55 -    plot(Hz,10*log10(psd_imp(1:length(Hz))),'ms-','linew',1,'markerfacecolor','w','markersize',4)
56 -    set(gca,'xlim',[0 100])
57 -    xlabel('Frequencia (Hz)'), ylabel('Atenuação (log)')
58 -    title('RF (Butterworth)')
```

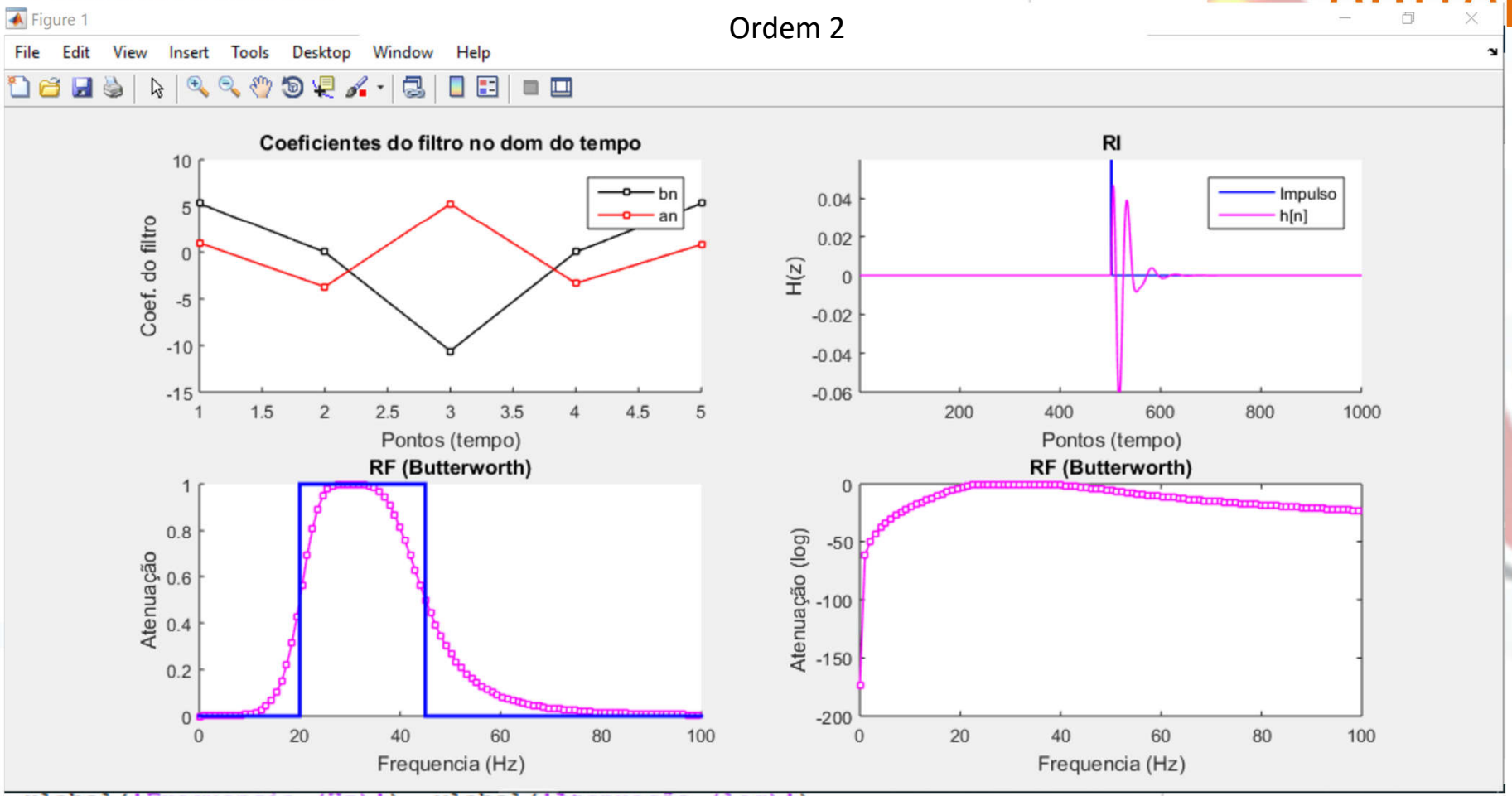
```
% Obter a RI
impulso = impseq(501,0,1000);
hn=filter(bn,an,impulso);
```




```
xlabel('Frequencia (Hz)'), ylabel('Atenuação (log)')
title('RF (Butterworth)')
```

```
% Obter a RI
impulso = impseq(501,0,1000);
hn=filter(bn,an,impulso);
```

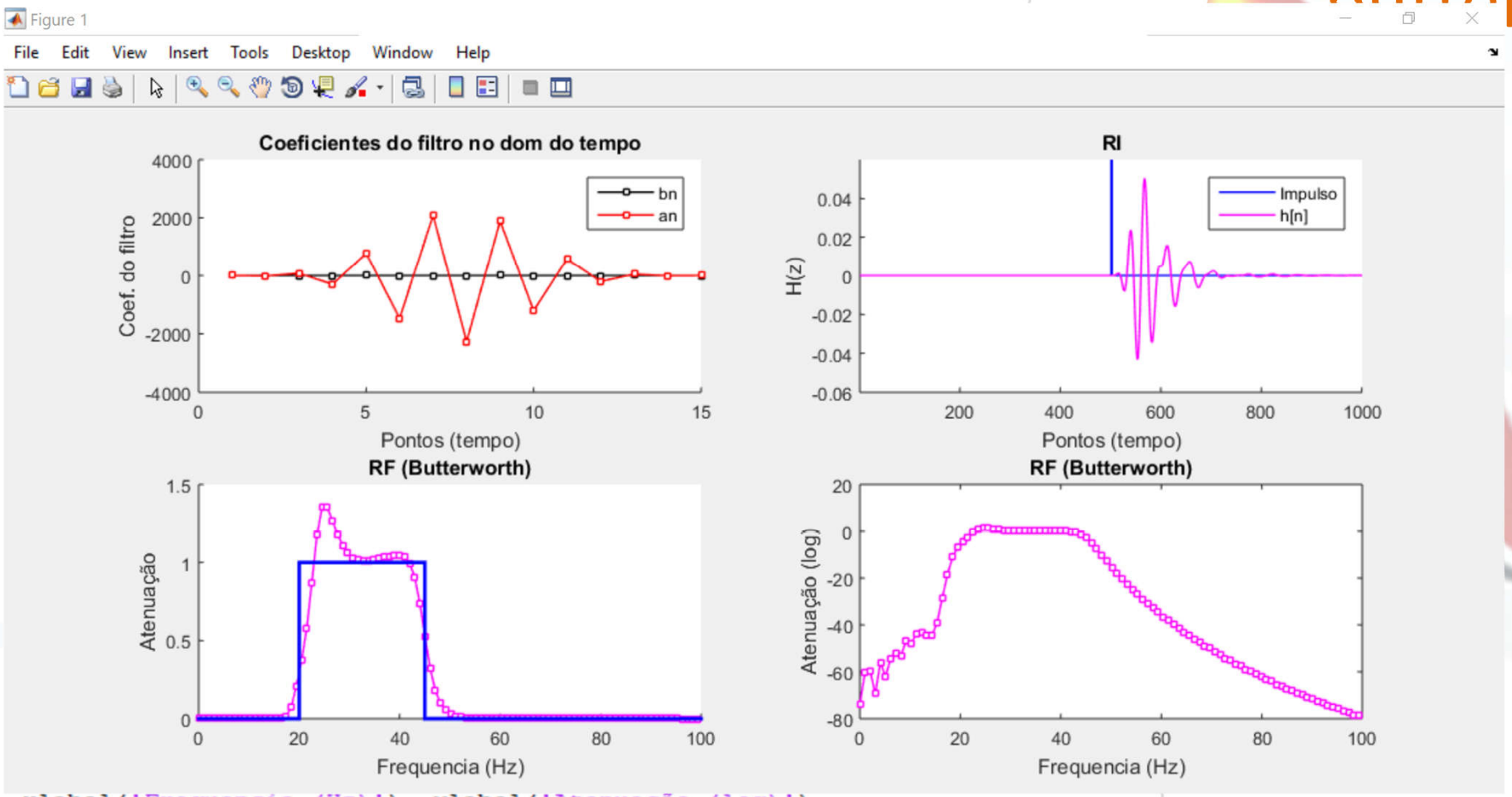
Ordem 2




```
xlabel('Frequencia (Hz)'), ylabel('Atenuação (log)')
title('RF (Butterworth)')
```

Ordem 7

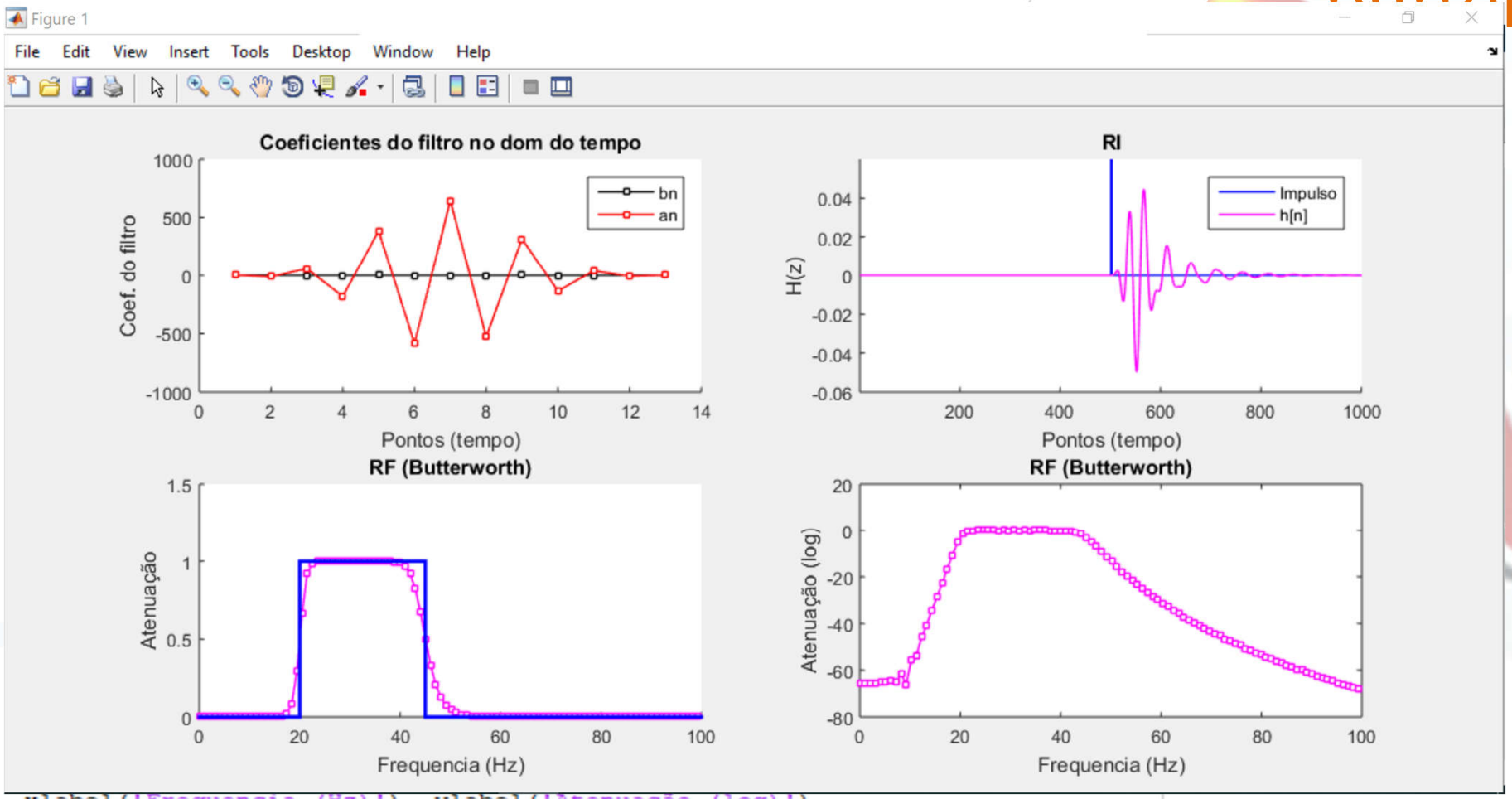
```
% Obter a RI
impulso = impseq(501,0,1000);
hn=filter(bn,an,impulso);
```




```
xlabel('Frequencia (Hz)'), ylabel('Atenuação (log)')
title('RF (Butterworth)')
```

Ordem 6

```
% Obter a RI
impulso = impseq(501,0,1000);
hn=filter(bn,an,impulso);
```




```
xlabel('Frequencia (Hz)'), ylabel('Atenuação (log)')
title('RF (Butterworth)')
```