Ficha Prática nº 9 de ATD 2022

Alberto Cardoso ©DEI2021/2022

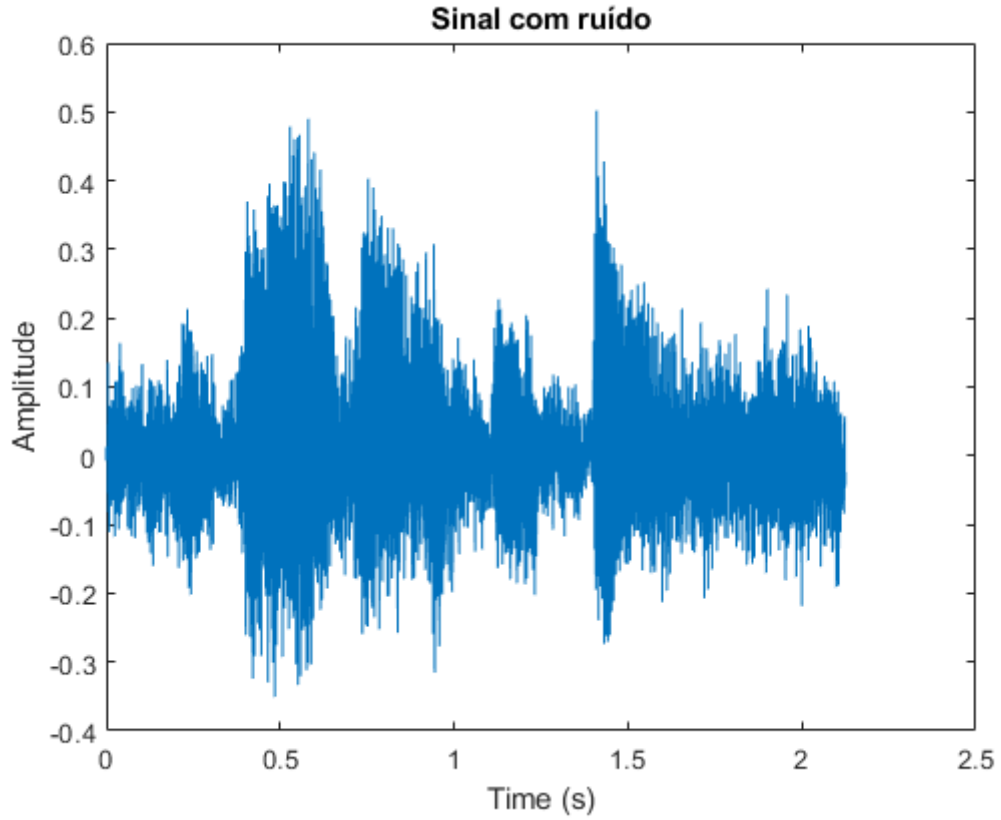
Objetivo: Pretende-se desenvolver filtros FIR e IIR para a filtragem de um sinal áudio corrompido por ruído. Os filtros deverão ser desenvolvidos usando a ferramenta MATLAB *filterDesigner*.

Exercício 1. Considerar o sinal áudio armazenado no ficheiro noisy_voice.wav. O sinal áudio foi corrompido por duas fontes independentes de ruído.

Exercício 1.1 Recorrendo ao cálculo da DFT do sinal, identifique a frequência de oscilação das duas fontes.

In [1]:
```
disp('FP9 - Exercício 1.1');
%% Ex1.1 Ler e escutar e representar o sinal áudio
clear yn fs
[yn,fs] = audioread('noisy_voice.wav');
disp('Frequência de amostragem (Hz):')
disp(fs)
sound(yn,fs)

N=length(yn);
disp('Número de amostras do sinal:')
disp(N)
t=[0:N-1]./fs;

figure(1);
plot(t,yn)
xlabel('Time (s)')
ylabel('Amplitude')
title('Sinal com ruído')
```

```
FP9 - Exercício 1.1
Frequência de amostragem (Hz):
        8000

Número de amostras do sinal:
       16998
```

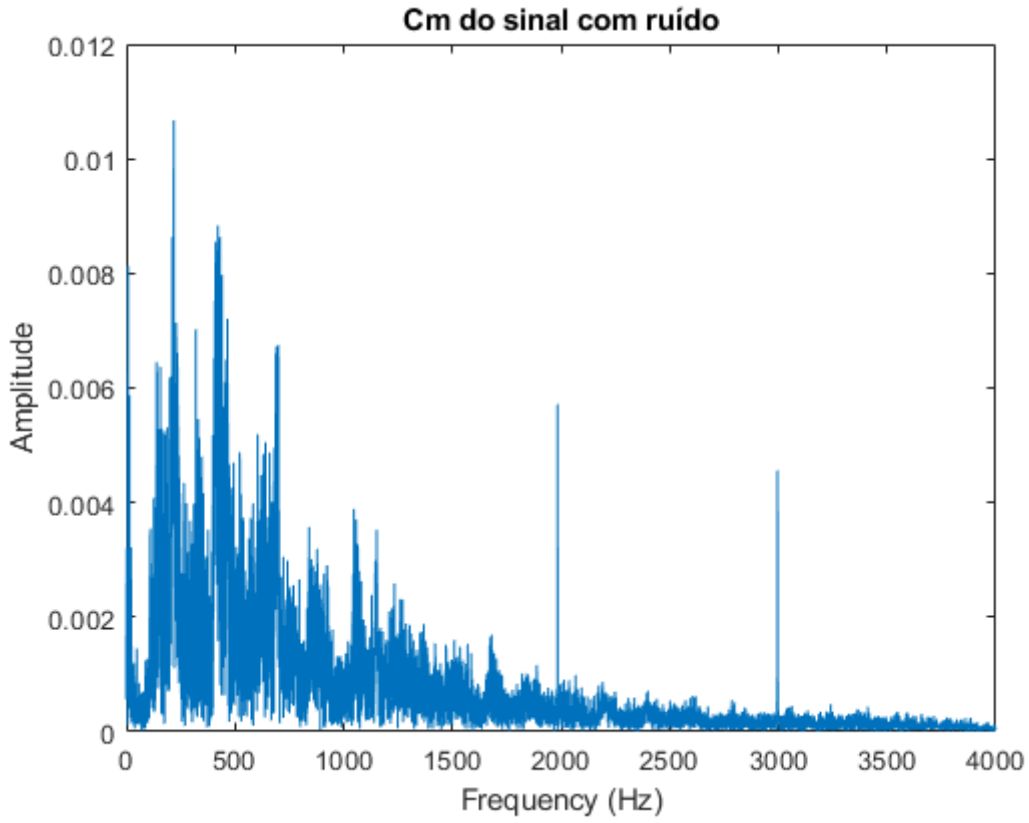


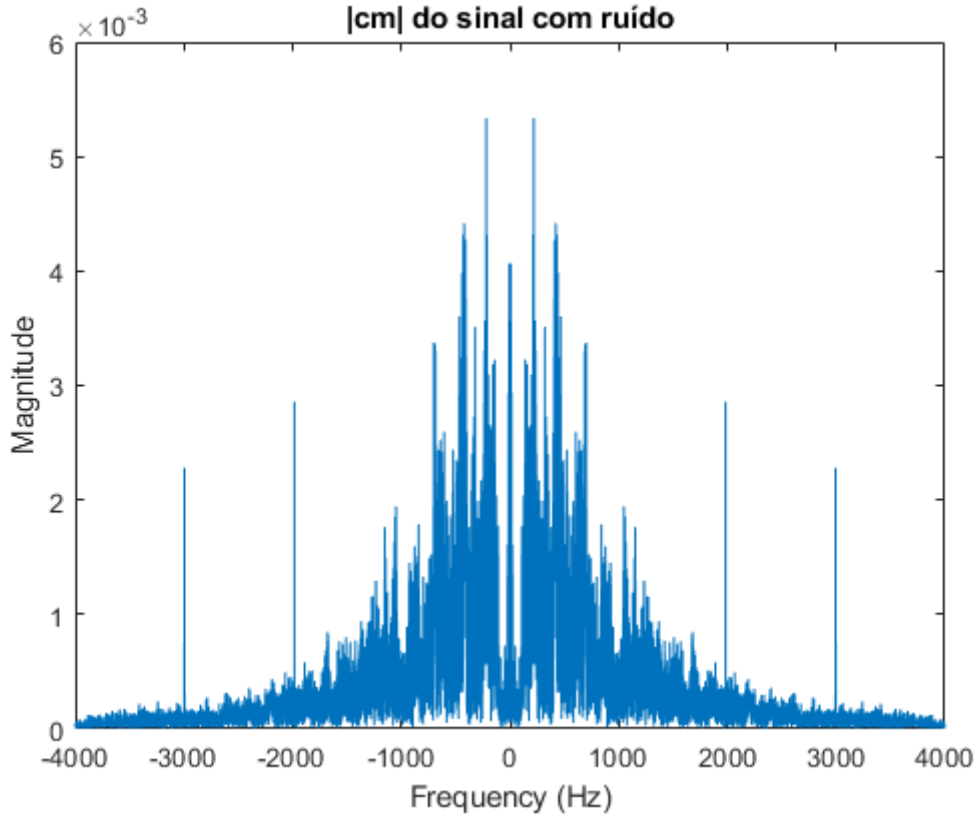
In [ ]:

```
%%file my_fft.m

function [f,cm,fnn,Cm,tetam]=my_fft(xn,fs)

N=length(xn); % dimensão do sinal de tempo discreto
X=fftshift(fft(xn)); %DFT
cm=X/N; %coeficientes da SF complexa
ind_f0=fix(N/2)+1; % índice da frequência 0

% coeficientes Cm e tetam da SF trigonométrica:
Cm=[abs(cm(ind_f0)); 2*abs(cm(ind_f0+1:end))];
tetam=angle(cm(ind_f0:end));

% vetor de frequências em Hz
if(mod(N,2)==0)
    f=-fs/2:fs/N:fs/2-fs/N;
else
    f=-fs/2+fs/N/2:fs/N:fs/2-fs/N/2;
end
fnn=f(ind_f0:end); % vetor de frequências não negativas
```

In [3]:

```
[f,cm_yn,fnn,Cm_yn]=my_fft(yn,fs);
Omega = linspace(0,pi,length(fnn));

figure(2);
plot(f,abs(cm_yn))
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title('|cm| do sinal com ruído')

figure(3);
plot(fnn,Cm_yn)
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Cm do sinal com ruído')
```





As fontes de ruído estão localizadas aproximadamente em: 2KHz e 3KHz.

Exercício 1.2 Que tipos de filtros as permitem remover?

Os filtros poderão ser do tipo passa-baixo, com uma frequência de corte inferior a 2KHz, ou rejeita-banda, aplicando dois filtros em série com bandas de rejeição em torno de 2KHz e de 3KHz.

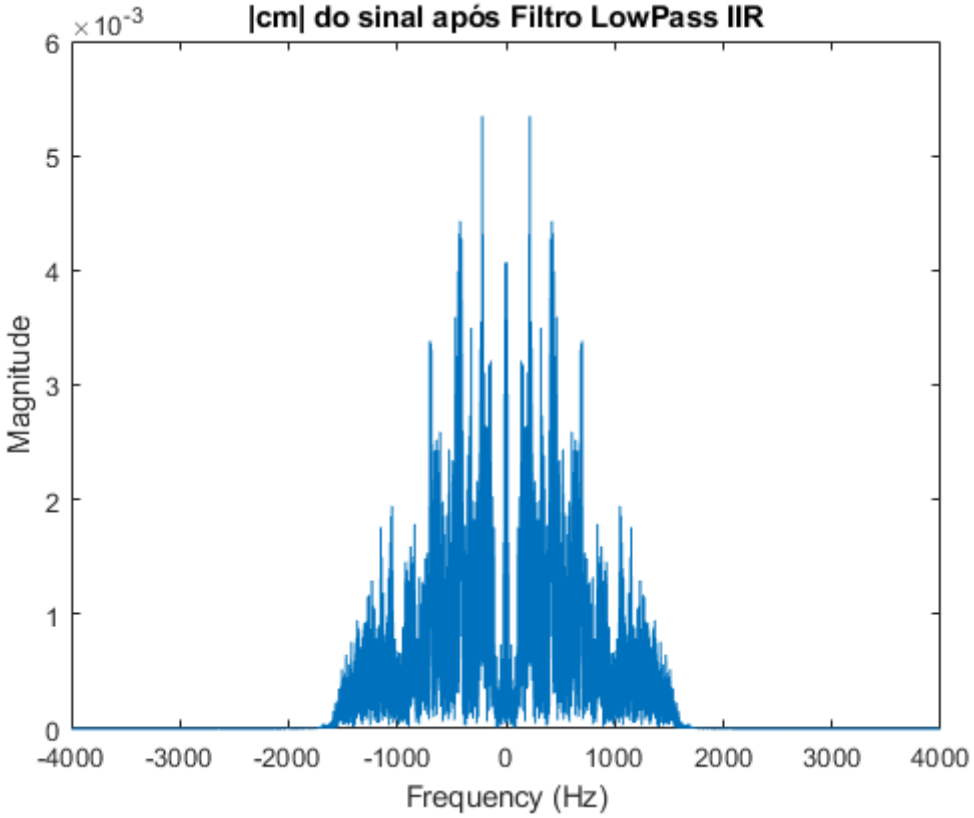
1.3 Desenvolva filtros FIR e IIR para os diferentes tipos de filtros. Use a ferramenta *filterDesigner*.

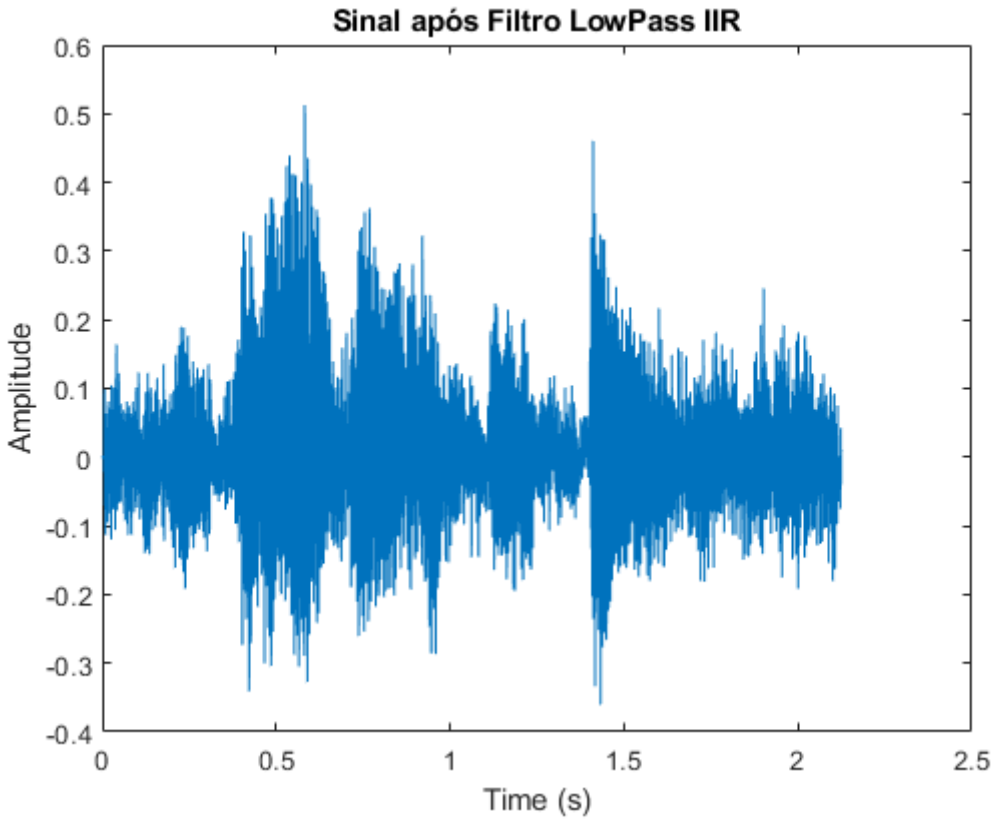
Nota: A ferramenta *filterDesigner* apenas calcula os parâmetros dos filtros. Para aplicar os filtros ao sinal deve exportá-los para Mat-files, e depois carregar cada Mat-file no script que desenvolver para implementar os filtros.

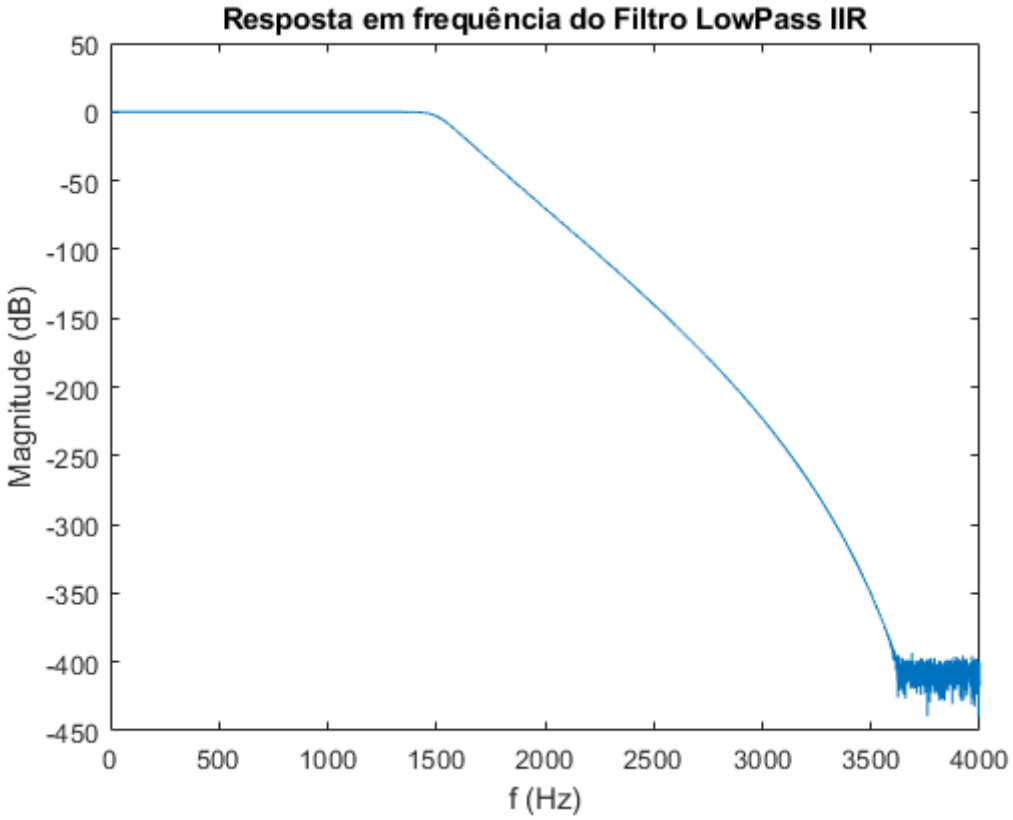
1.4 Aplique os filtros (use a função *filter*) e verifique a eficácia dos mesmos recorrendo à visualização da magnitude do espectro. Reproduza os sinais usando a função *sound*.

In [4]:
```
disp('FP9 - Exercício 1.4');
%=== Remover ruído usando um filtro Passa-Baixo IIR ===
% Filtro previamente projetado no filterDesigner
% e guardado em ficheiro .mat

fiir_LP=load('LowPassIIR.mat')
G_fiir_LP=fiir_LP.G;
SOS_fiir_LP=fiir_LP.SOS;

[b_fiir_LP,a_fiir_LP] = sos2tf(SOS_fiir_LP,G_fiir_LP);
yfiir_LP=filter(b_fiir_LP,a_fiir_LP,yn);

H_fiir_LP=freqz(b_fiir_LP,a_fiir_LP,Omega);

figure(4);
plot(fnn,20*log10(abs(H_fiir_LP)))
xlabel('f (Hz)')
ylabel('Magnitude (dB)')
title('Resposta em frequência do Filtro LowPass IIR')

figure(5);
plot(t,yfiir_LP)
xlabel('Time (s)')
ylabel('Amplitude')
title('Sinal após Filtro LowPass IIR')

[f,cm_yfiir_LP]=my_fft(yfiir_LP,fs);

figure(6);
plot(f,abs(cm_yfiir_LP))
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title('|cm| do sinal após Filtro LowPass IIR')
sound(yfiir_LP,fs)
```

```
FP9 - Exercício 1.4

fiir_LP =

  struct with fields:

      G: [11x1 double]
    SOS: [10x6 double]
```


|cm| do sinal após Filtro LowPass IIR
×10^-3
Magnitude
Frequency (Hz)
0
1
2
3
4
5
6
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000


Sinal após Filtro LowPass IIR
Amplitude
Time (s)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5
2
2.5

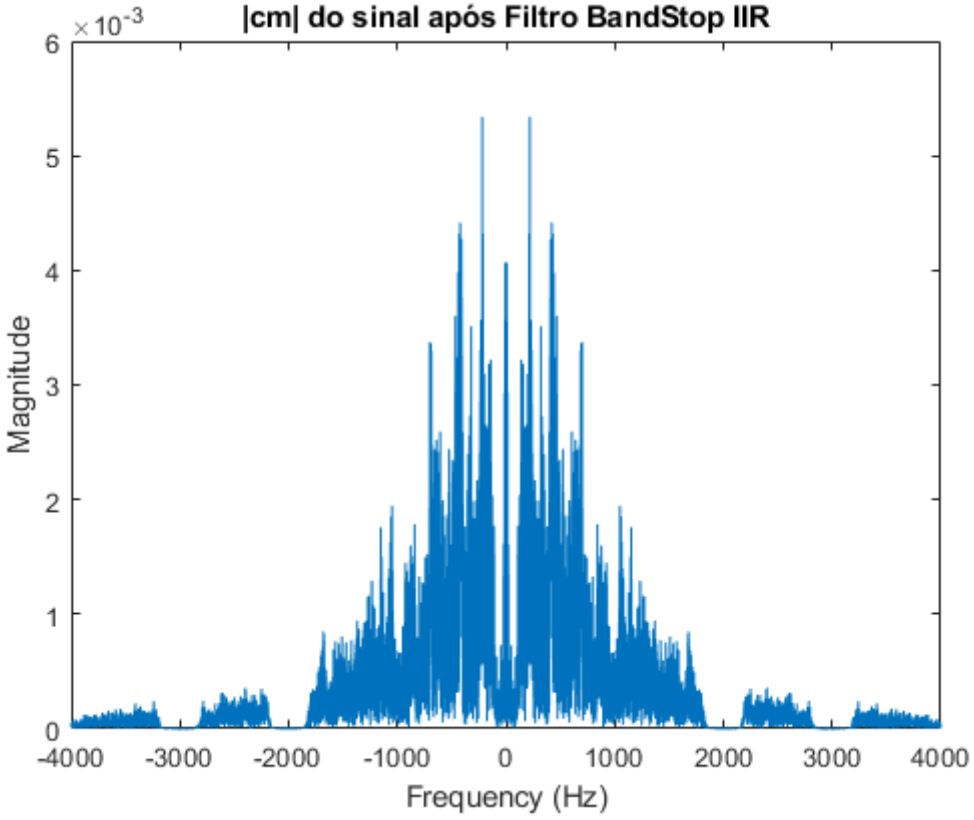


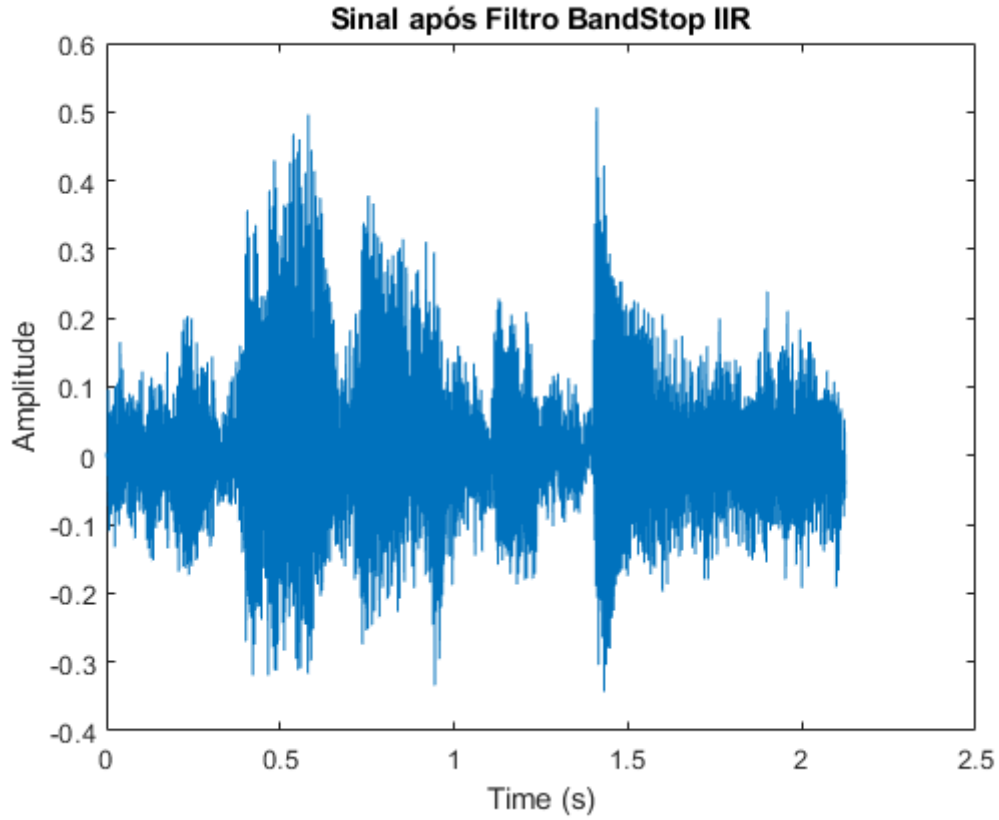
In [5]:

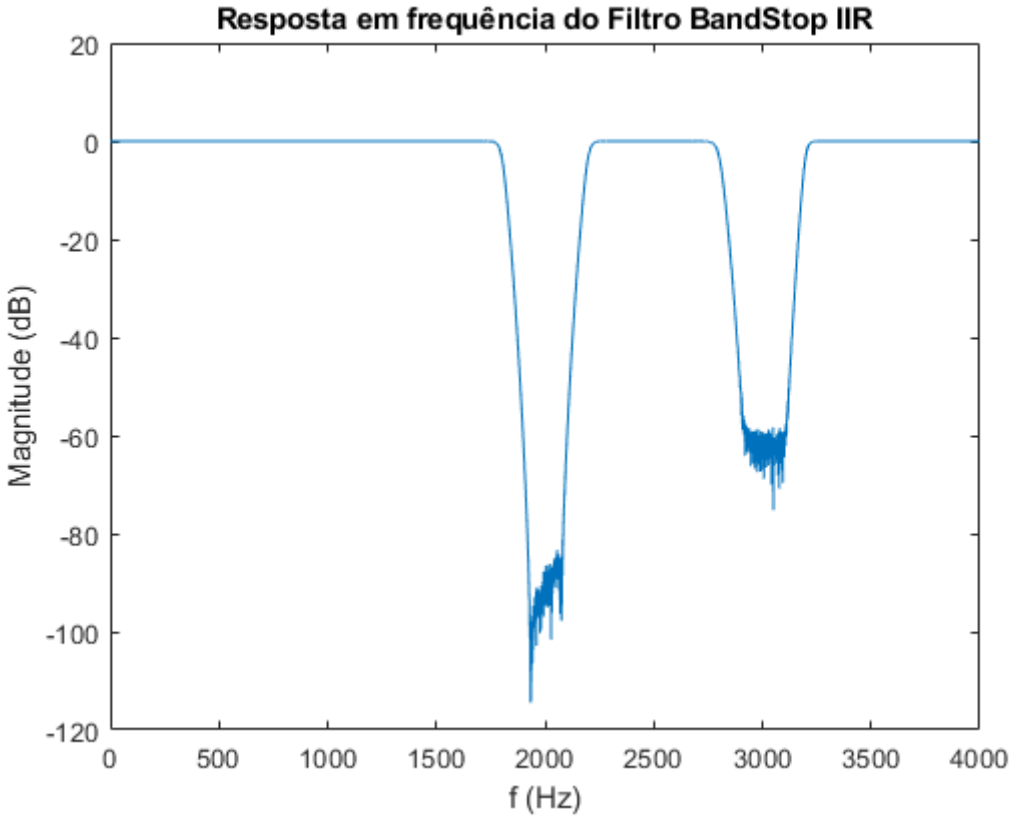
```
%=== Remover ruído usando filtros rejeita-banda IIR ===
fiir_BS2=load('BandStop2IIR.mat')
G_fiir_BS2=fiir_BS2.G;
SOS_fiir_BS2=fiir_BS2.SOS;

[b_fiir_BS2,a_fiir_BS2] = sos2tf(SOS_fiir_BS2,G_fiir_BS2);

yfiir_BS2=filter(b_fiir_BS2,a_fiir_BS2,yn);

fiir_BS3=load('BandStop3IIR.mat')
G_fiir_BS3=fiir_BS3.G;
SOS_fiir_BS3=fiir_BS3.SOS;

[b_fiir_BS3,a_fiir_BS3] = sos2tf(SOS_fiir_BS3,G_fiir_BS3);
yfiir_BS23=filter(b_fiir_BS3,a_fiir_BS3,yfiir_BS2);

H_fiir_BS23=freqz(conv(b_fiir_BS3,b_fiir_BS2),...
    conv(a_fiir_BS3,a_fiir_BS2),Omega);

figure(7);
plot(fnn,20*log10(abs(H_fiir_BS23)))
xlabel('f (Hz)')
ylabel('Magnitude (dB)')
title('Resposta em frequência do Filtro BandStop IIR')

figure(8);
plot(t,yfiir_BS23)
xlabel('Time (s)')
ylabel('Amplitude')
title('Sinal após Filtro BandStop IIR')

[f,cm_yfiir_BS23]=my_fft(yfiir_BS23,fs);

figure(9);
plot(f,abs(cm_yfiir_BS23))
xlabel('Frequency (Hz)')
```

```
ylabel('Magnitude')
title('|cm| do sinal após Filtro BandStop IIR')
sound(yfiir_BS23,fs)
```

```
fiir_BS2 =

  struct with fields:

      G: [11x1 double]
    SOS: [10x6 double]


fiir_BS3 =

  struct with fields:

      G: [11x1 double]
    SOS: [10x6 double]
```





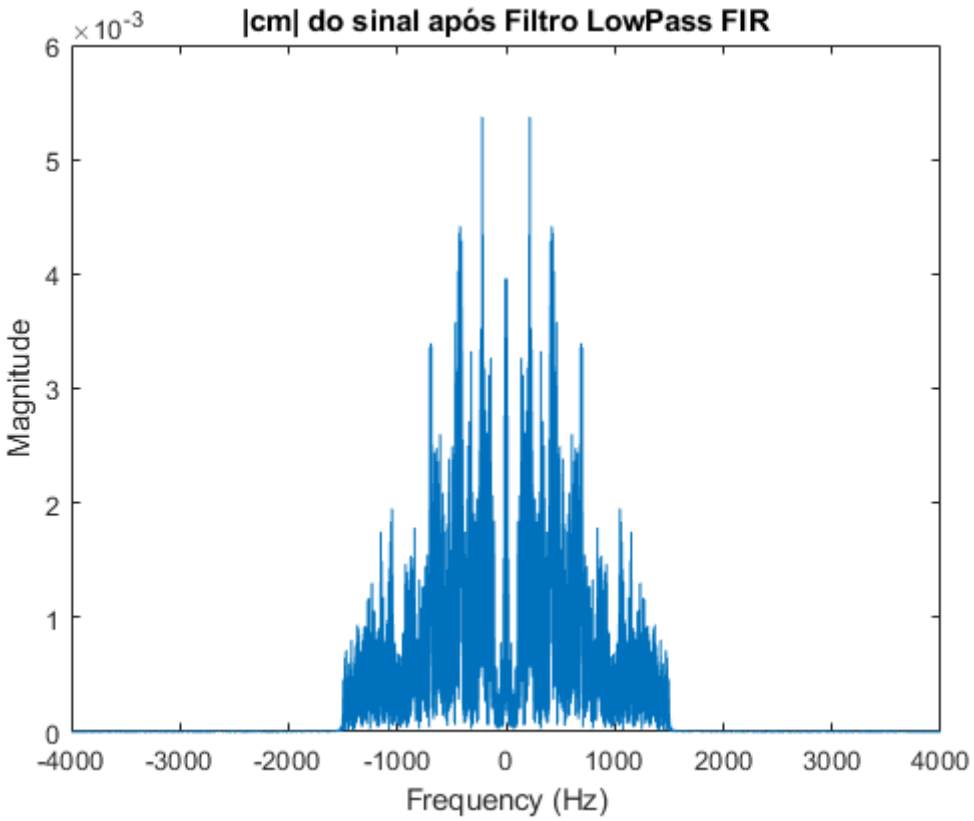


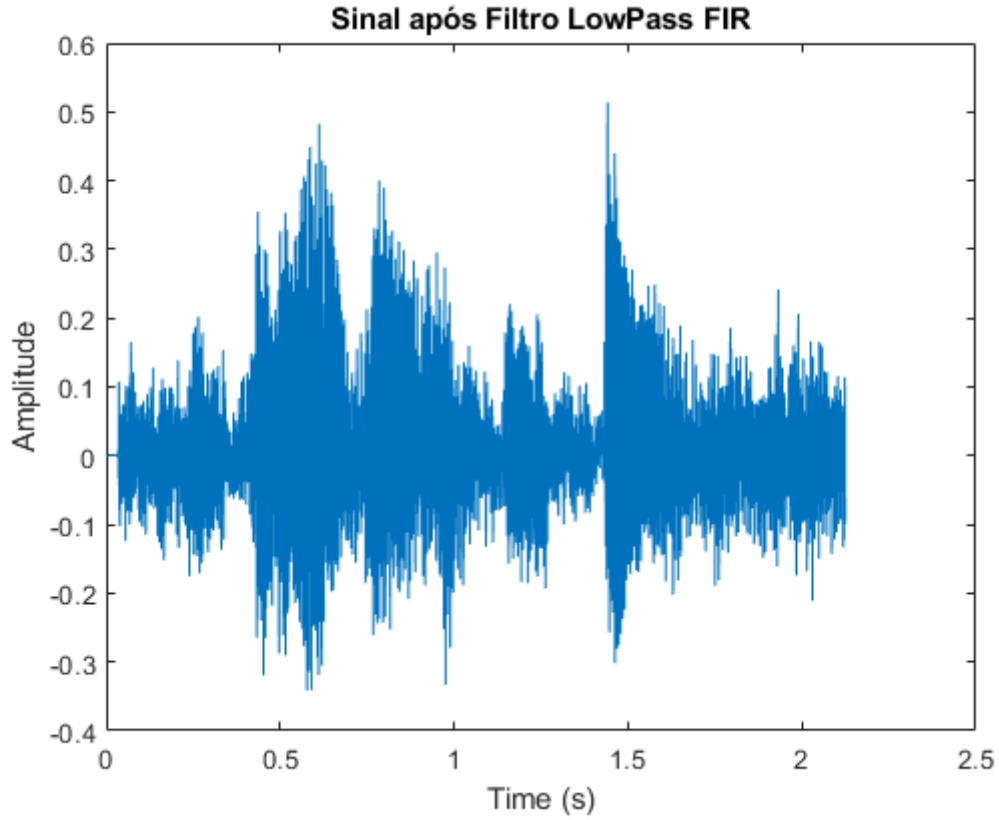
In [6]:

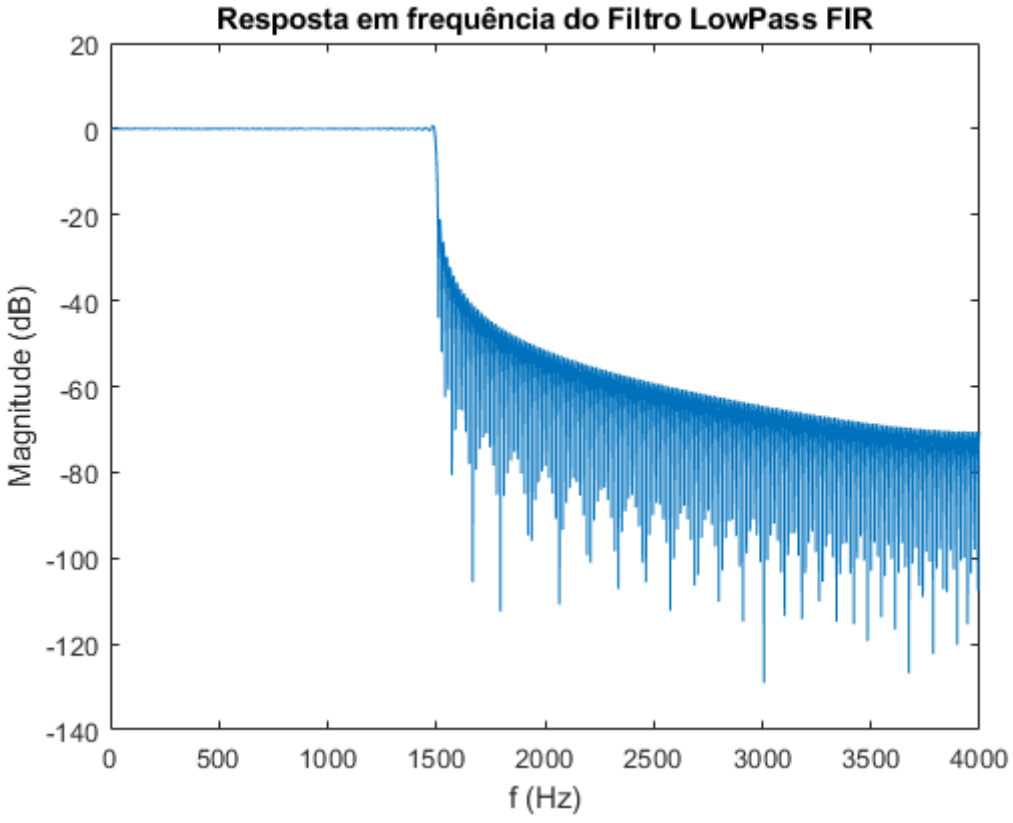
```
%=== Remover ruído usando um filtro passa-baixo FIR ===
ffir_LP=load('LowPassFIR.mat')
b_ffir_LP=ffir_LP.Num;

yfir_LP=filter(b_ffir_LP,1,yn);

H_ffir_LP=freqz(b_ffir_LP,1,Omega);

figure(10);
plot(fnn,20*log10(abs(H_ffir_LP)))
```

```
xlabel('f (Hz)')
ylabel('Magnitude (dB)')
title('Resposta em frequência do Filtro LowPass FIR')

figure(11);
plot(t,yfir_LP)
xlabel('Time (s)')
ylabel('Amplitude')
title('Sinal após Filtro LowPass FIR')

[f,cm_yfir_LP]=my_fft(yfir_LP,fs);

figure(12);
plot(f,abs(cm_yfir_LP))
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title('|cm| do sinal após Filtro LowPass FIR')
sound(yfir_LP,fs)
```

```
ffir_LP =

  struct with fields:

    Num: [1x501 double]
```





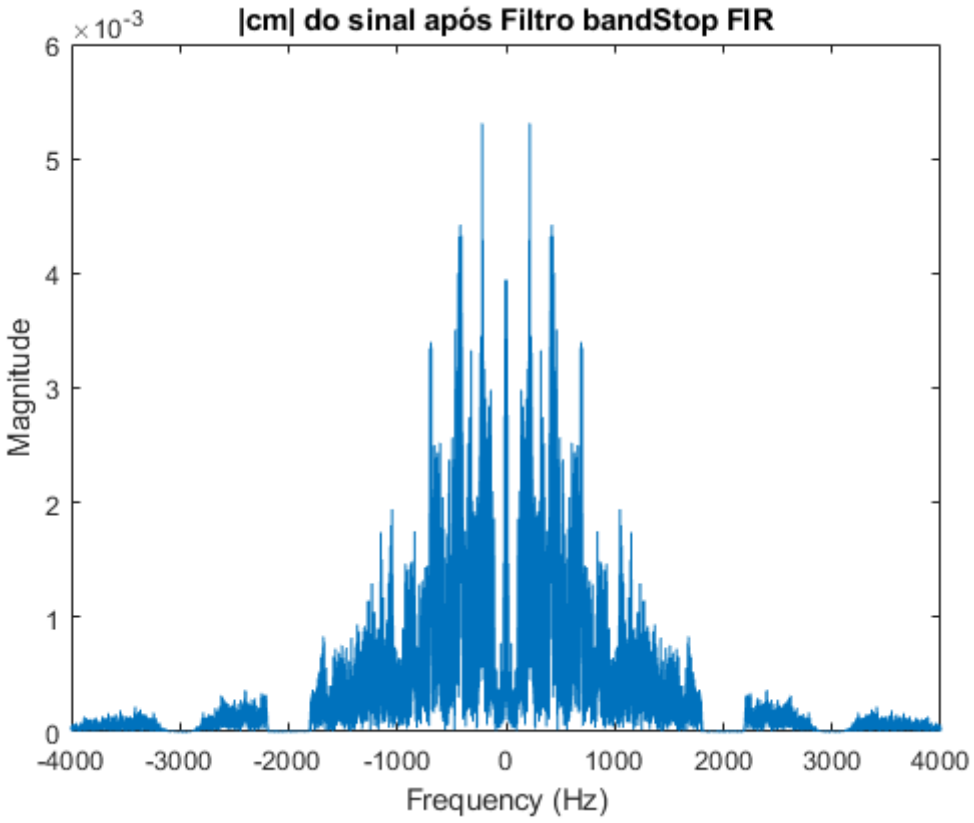


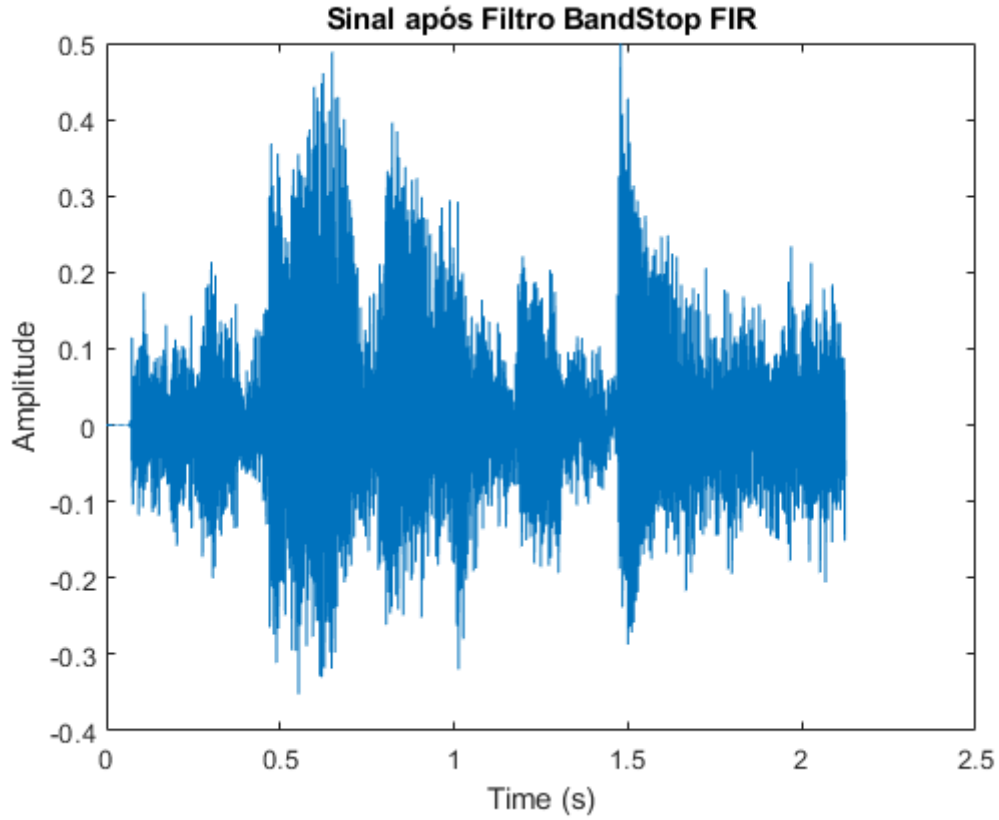
In [7]:

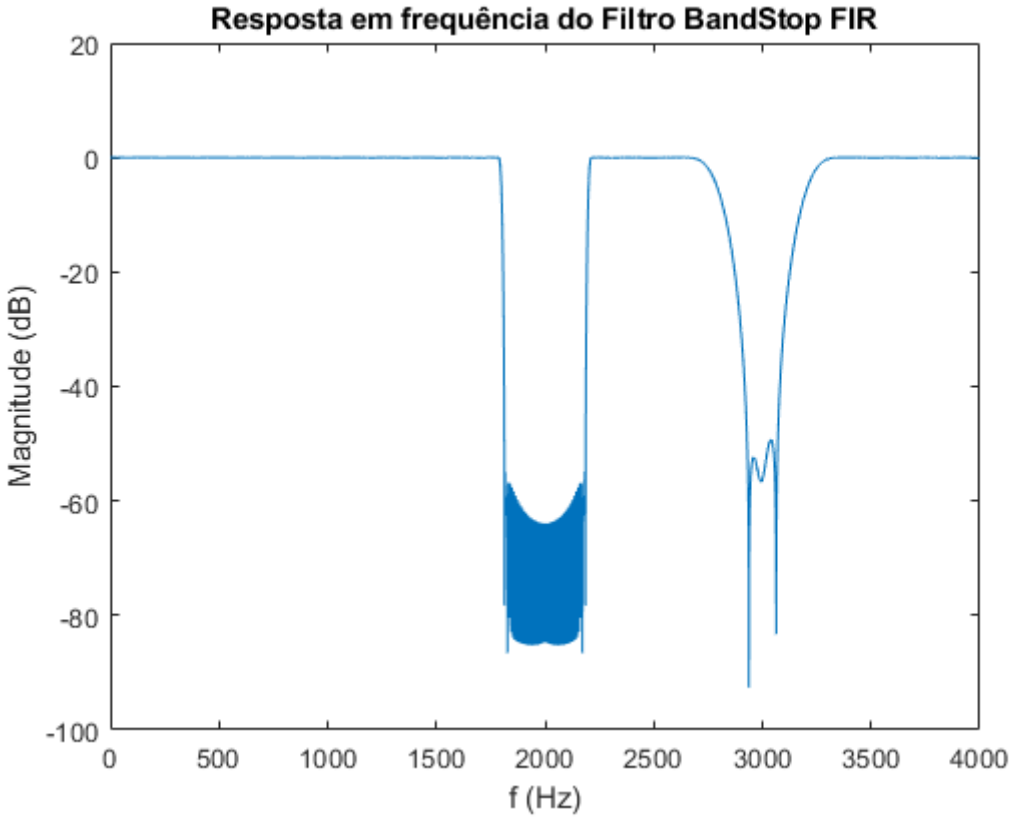
```
%=== Remover ruído usando filtros rejeita-banda FIR ===
ffir_BS2=load('BandStop2FIR.mat')
b_ffir_BS2=ffir_BS2.Num;

yfir_BS2=filter(b_ffir_BS2,1,yn);

ffir_BS3=load('BandStop3FIR.mat')
b_ffir_BS3=ffir_BS3.Num;

yfir_BS23=filter(b_ffir_BS3,1,yfir_BS2);
```

```
H_ffir_BS23=freqz(conv(b_ffir_BS3,b_ffir_BS2),1,Omega);

figure(13);
plot(fnn,20*log10(abs(H_ffir_BS23)))
xlabel('f (Hz)')
ylabel('Magnitude (dB)')
title('Resposta em frequência do Filtro BandStop FIR')

figure(14);
plot(t,yfir_BS23)
xlabel('Time (s)')
ylabel('Amplitude')
title('Sinal após Filtro BandStop FIR')

[f,cm_yfir_BS23]=my_fft(yfir_BS23,fs);

figure(15);
plot(f,abs(cm_yfir_BS23))
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title('|cm| do sinal após Filtro bandStop FIR')
sound(yfir_BS23,fs)
```

```
ffir_BS2 =

  struct with fields:

    Num: [1x101 double]


ffir_BS3 =

  struct with fields:

    Num: [1x1001 double]
```




Sinal após Filtro BandStop FIR
Amplitude
Time (s)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5
2
2.5


Resposta em frequência do Filtro BandStop FIR
Magnitude (dB)
f (Hz)
20
0
-20
-40
-60
-80
-100
0
500
1000
1500
2000
2500
3000
3500
4000