INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE GRADUAÇÃO EM ELETRÔNICA INDUSTRIAL

DISCIPLINA: PROCESSADORES DE SINAIS DIGITAIS APLICADOS A ÁUDIO E VÍDEO

Atividade Prática 03

Aluno

Marcelo Brancalhão Gaspar

Professor

Fernando Santana Pacheco

Florianópolis, SC - 26 de abril de 2022

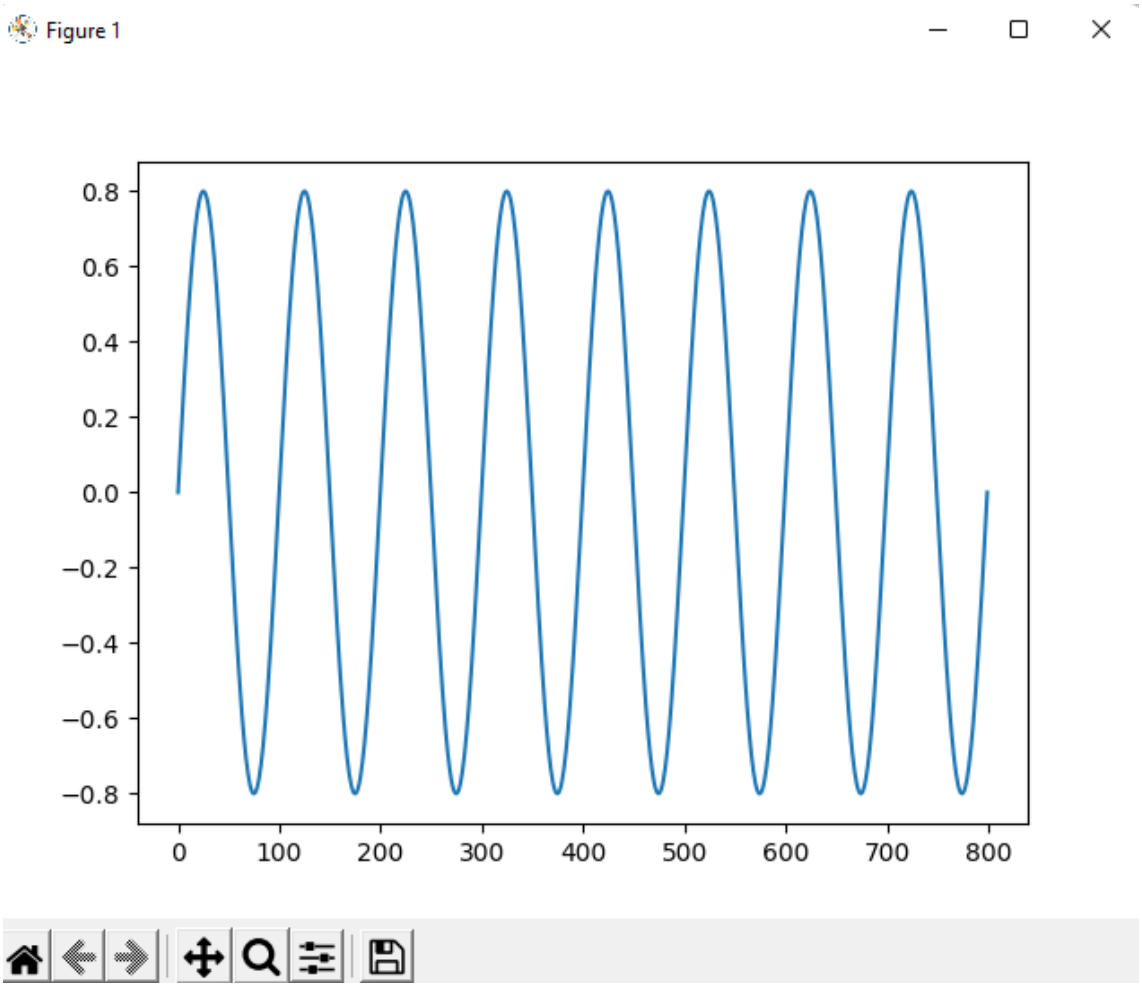
1) Criar uma função para gerar um tom senoidal.

Parâmetros de entrada: frequência (Hz), duração (s), amplitude (0 a 1) e frequência de amostragem (Hz).

Verifique se a frequência da saída está correta medindo um período do sinal.

Código e resultado;

```
############################################################

##  Created on: Abril 26, 2022

##      Author: Marcelo Brancalhão Gaspar

##      Instituto Federal de Santa Catarina

##      DSP 2 - Fernando Santana Pacheco

##

############################################################


import numpy as np

import matplotlib.pyplot as plt

import simpleaudio as sa

def tonegen(freq,duration=0.1,amp=1, fs=8000):

    t = np.linspace(0., duration, int(fs*duration))

    x = amp*(np.sin(2*np.pi*freq*t))

    audio = x*(2**15 - 1) / np.max(np.abs(x))

    audio = audio.astype(np.int16)

    play_obj = sa.play_buffer(audio, 1, 2, fs)

    play_obj.wait_done()

    return x

s440 = tonegen(80, 0.1, 0.8, 8000)

plt.plot(s440)

plt.show()
```



2) Criar uma função no Matlab que receba um vetor com uma sequência de dígitos e gere tons DTMF correspondentes. Os parâmetros de entrada da função são: sequência de dígitos, duração em segundos de cada dígito, intervalo em segundos entre cada dígito e amplitude (0 a 1). O sinal de saída deve ter frequência de amostragem de 8 kHz.

Para verificar se o sinal está correto, teste a saída usando um decodificador DTMF, como os listados:

Código e resultado; (Resultado de áudio é possível obter executando o código abaixo, que também está disponível no anexo.)

```
############################################################
##
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
```

```
##
############################################################
import numpy as np
import matplotlib.pyplot as plt
import simpleaudio as sa
from scipy.io import wavfile
def tonegen(freq,duration=0.1,amp=1, fs=8000):
    t = np.linspace(0., duration, int(fs*duration))
    x = amp*(np.sin(2*np.pi*freq*t))
    audio =x*(2**15 - 1) / np.max(np.abs(x))
    audio =audio.astype(np.int16)
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
    return x
  def dtmfgen (seq_fone, tempo, intervalo, maxima):
    y = seq_fone[0]
    fs = 8000
    dtmf = {0: {'dtmf1': 941, 'dtmf2': 1336},
            1: {'dtmf1': 697, 'dtmf2': 1209},
            2: {'dtmf1': 697, 'dtmf2': 1336},
            3: {'dtmf1': 693, 'dtmf2': 1477},
             4: {'dtmf1': 770, 'dtmf2': 1209},
            5: {'dtmf1': 770, 'dtmf2': 1336},
            6: {'dtmf1': 770, 'dtmf2': 1477},
            7: {'dtmf1': 852, 'dtmf2': 1209},
            8: {'dtmf1': 852, 'dtmf2': 1336},
            9: {'dtmf1': 852, 'dtmf2': 1477},
            0: {'dtmf1': 941, 'dtmf2': 1336},
            }
    x = [0]
    for idx in range (len(seq_fone)):
```

```
        fr_1 = dtmf[int(seq_fone[idx])]['dtmf1']
        fr_2 = dtmf[int(seq_fone[idx])]['dtmf2']
        x_dtmf = tonegen(fr_1, tempo, maxima/2, fs) + tonegen(fr_2, tempo, maxima/2, fs)
        x_dtmf = np.concatenate((x_dtmf, np.zeros(int(intervalo*fs))))
        x = np.concatenate((x, x_dtmf))
    audio = x * (2**15 - 1) / np.max(np.abs(x))
    audio = audio.astype(np.int16)
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
        return x
s981 = dtmfgen(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'], 0.1, 40e-3, 0.7)
plt.plot(s981, 'bo', s981, 'k')
```

3) Criar uma função para remover uma em cada duas amostras de um sinal, criando um novo sinal que tem somente as amostras de índice ímpar do sinal original.

Código e resultado;

```
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
##
############################################################
import numpy as np
import matplotlib.pyplot as plt
def halfsp(array):
    hs = array[0::2]
    return hs
x=np.array([7, 3, 9, 1, 0, 4])
result=halfsp(x)
print(result)
```

```
>>> %Run e3.py
  [7 9 0]
>>>
```

4) Criar uma função para criar um novo sinal que insere uma nova amostra entre cada duas amostras. Essa nova amostra é a média das duas amostras vizinhas.

Código e resultado;

```
############################################################
##
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
##
############################################################


import numpy as np
import matplotlib.pyplot as plt
def doublesp(x):
    ds = np.array([])
    for i in range(0,x.size-1):
        mean = (x[i]+x[i+1])/2
        ds = np.append(ds,[x[i],mean])
    ds = np.append(ds,x[i+1])
    return ds
x=np.array([7, 3, 9, 1, 0, 4])
result=doublesp(x)
print(result)
```

```
>>> %Run e4.py
  [7.  5.  3.  6.  9.  5.  1.  0.5 0.  2.  4. ]
>>>
```

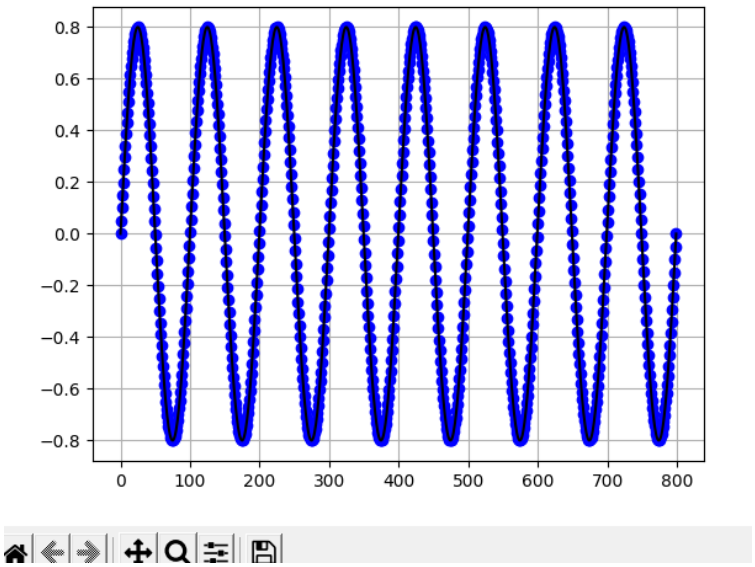
5) Gere um tom de 220 Hz. Aplique a função `halfsp`. Escute o resultado antes e depois da aplicação. O que ocorre na frequência e no tempo? Mostre graficamente o resultado na frequência através da transformada de Fourier (implementada pela função `numpy.fft`).

Código e resultado;

```
############################################################
##
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
##
############################################################
import simpleaudio as sa
import numpy as np
import matplotlib.pyplot as plt
def tonegen(freq,duration=0.1,amp=1, fs=8000):
    t = np.linspace(0., duration, int(fs*duration))
    x = amp*(np.sin(2*np.pi*freq*t))
    audio = x*(2**15 - 1) / np.max(np.abs(x))
    audio = audio.astype(np.int16)
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
    return x
s220 =tonegen(80, 0.1, 0.8, 8000)
   def halfsp(s220):
    hs = s220[0::2]
    return hs
result=halfsp(s220)
```

```
h220 = halfsp(s220)
plt.plot(s220,'bo', s220, 'k')
plt.grid()
plt.show()
plt.plot(h220,'bo', h220, 'k')
plt.grid()
plt.show()
plt.xlim(0,400)
plt.plot(s220,'k', h220, 'r')
plt.grid()
plt.show()
s220fft=np.fft.fft(s220)
ns=s220fft.size
fs=np.fft.fftfreq(ns,1/8000)
plt.xlim(0,500)
plt.plot(fs,abs(s220fft)/4000)
plt.show()
h220fft=np.fft.fft(h220)
nh=h220fft.size
fh=np.fft.fftfreq(nh,1/8000)
plt.plot(fh,abs(h220fft)/4000)
plt.show()
```

Antes


0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
100
200
300
400
500
600
700
800

Após:

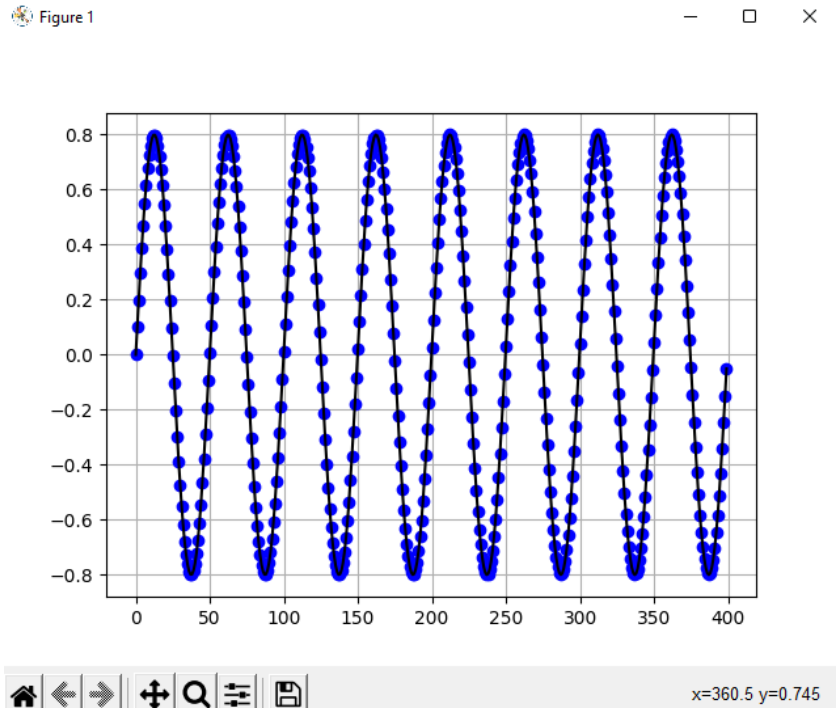
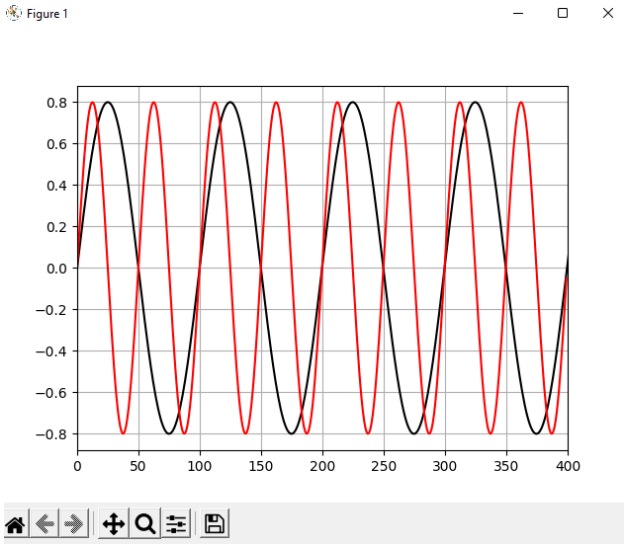
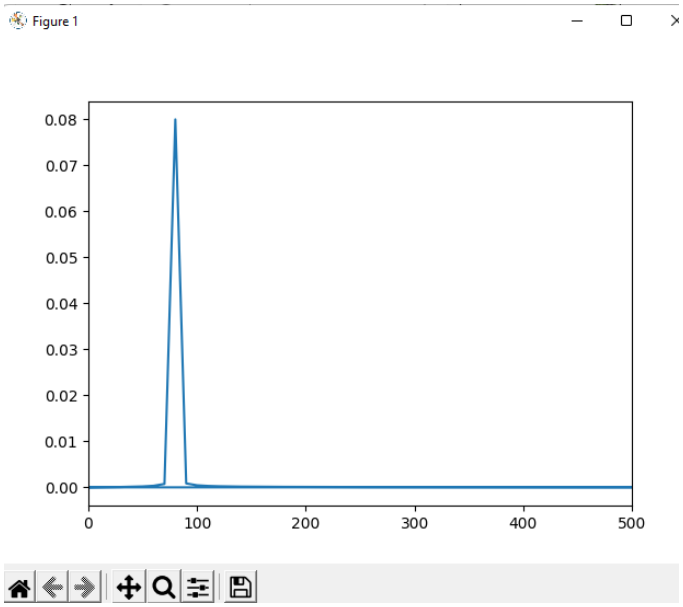


Figure 1
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
50
100
150
200
250
300
350
400
x=360.5 y=0.745

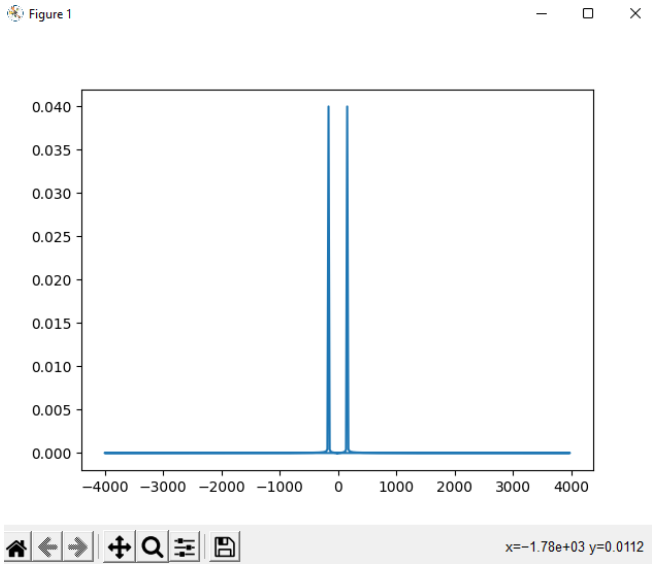
Comparação



FFT:



Após FFT



6) Faça a mesma análise no tempo e frequência em relação à função `doublesp`.

Código e resultado;

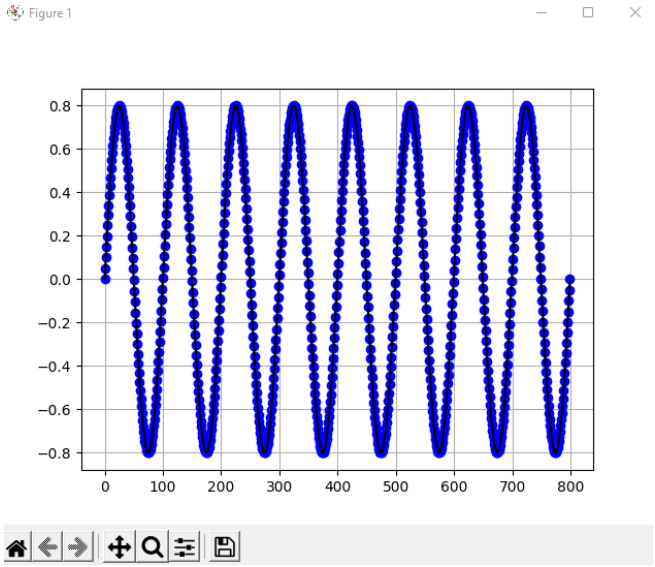
```
############################################################
##
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
##
############################################################
import simpleaudio as sa
import numpy as np
import matplotlib.pyplot as plt


def tonegen(freq,duration=0.1,amp=1, fs=8000):
    t = np.linspace(0., duration, int(fs*duration))
    x = amp*(np.sin(2*np.pi*freq*t))
     audio = x*(2**15 - 1) / np.max(np.abs(x))
    audio = audio.astype(np.int16)
```

```
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
    return x
 s220 =tonegen(80, 0.1, 0.8, 8000)
def doublesp(s220):
    ds = np.array([])
    for i in range(0,s220.size-1):
        mean = (s220[i]+s220[i+1])/2
        ds = np.append(ds,[s220[i],mean])
    ds = np.append(ds,s220[i+1])
    return ds
h220 = doublesp(s220)
plt.plot(s220,'bo', s220, 'k')
plt.grid()
plt.show()
plt.plot(h220,'bo', h220, 'k')
plt.grid()
plt.show()
plt.xlim(0,400)
plt.plot(s220,'k', h220, 'r')
plt.grid()
plt.show()
s220fft=np.fft.fft(s220)
ns=s220fft.size
fs=np.fft.fftfreq(ns,1/8000)
plt.xlim(0,500)
plt.plot(fs,abs(s220fft)/4000)
plt.show()
h220fft=np.fft.fft(h220)
nh=h220fft.size
fh=np.fft.fftfreq(nh,1/8000)
```

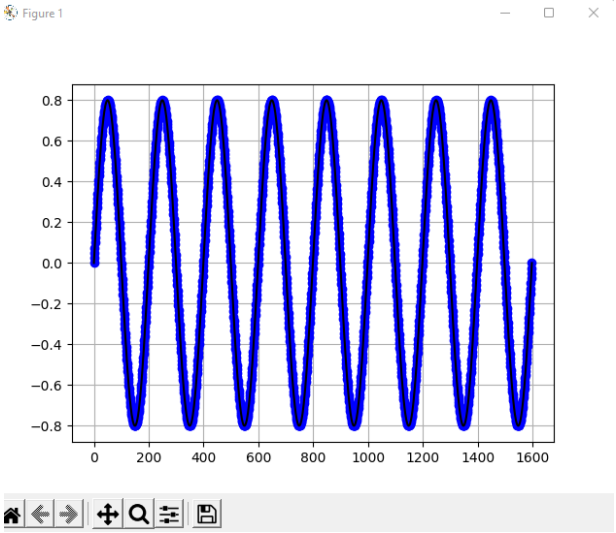
plt.plot(fh,abs(h220fft)/4000)

plt.show()

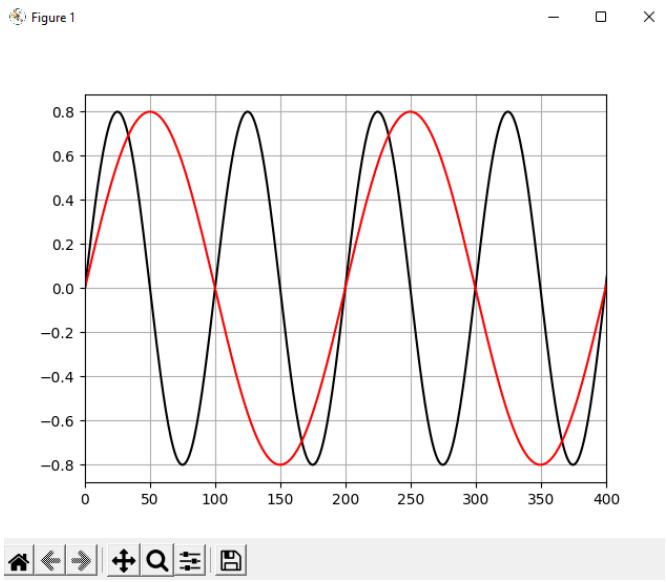
Antes;



Após

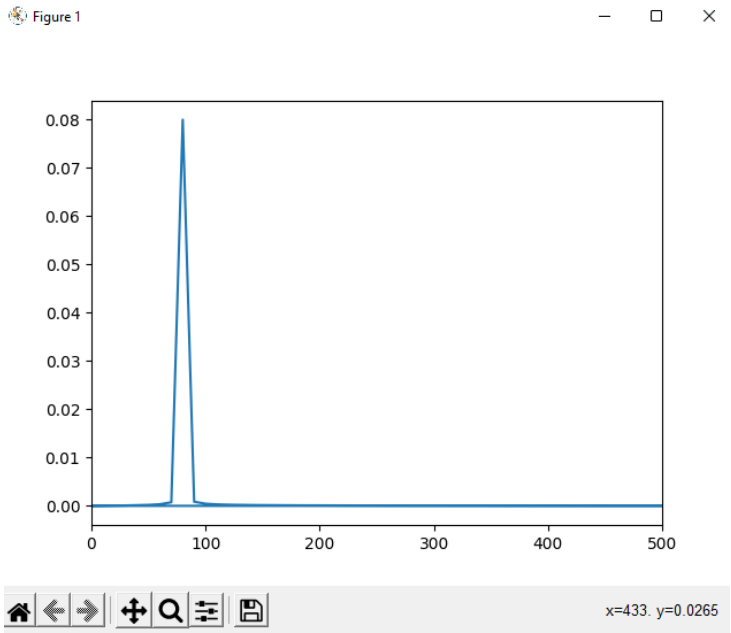


Comparativo

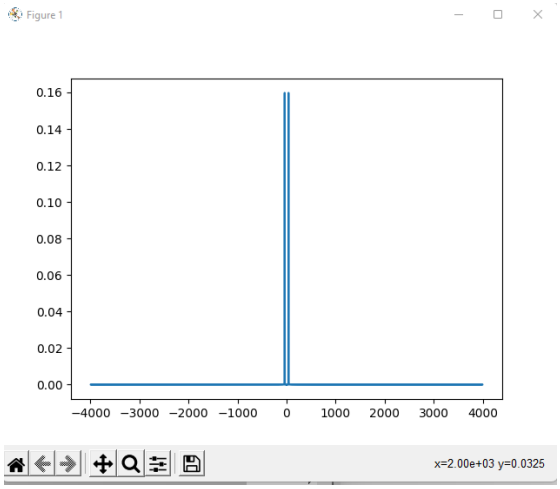


FFT

FFT;



Após FFT



7) Grave alguma frase com sua voz. Depois, observe e escute o resultado do processamento com:

- `halfsp`
- `doublesp`
- `numpy.fliplr`
- `numpy.flipud`

Código e resultado;

```
############################################################

##

##  Created on: Abril 26, 2022

##      Author: Marcelo Brancalhão Gaspar

##      Instituto Federal de Santa Catarina

##      DSP 2 - Fernando Santana Pacheco

##

############################################################

import numpy as np

import matplotlib.pyplot as plt

import simpleaudio as sa

import math as math

from scipy.io import wavfile

def tonegen(freq,duration=0.1,amp=1, fs=8000):

    t = np.linspace(0., duration, int(fs*duration))

    x = amp*(np.sin(2*np.pi*freq*t))
```

```
    audio = x*(2**15 - 1) / np.max(np.abs(x))
    audio = audio.astype(np.int16)
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
    return x
fs,x = wavfile.read('voz.wav')
plt.plot(x)
plt.show()
def halfsp(array):
    hs = array[0::2]
    return hs
result=halfsp(x)
plt.plot(x)
plt.show()
np.fliplr(x)
plt.plot(x)
plt.show()
np.flipud(x)
plt.plot(x)
plt.show()
def doublesp(x):
    ds = np.array([])
    for i in range(0,x.size-1):
        mean = (x[i]+x[i+1])/2
        ds = np.append(ds,[x[i],mean])
    ds = np.append(ds,x[i+1])
    return ds
result=doublesp(x)
plt.plot(x)
plt.show()
```

Half

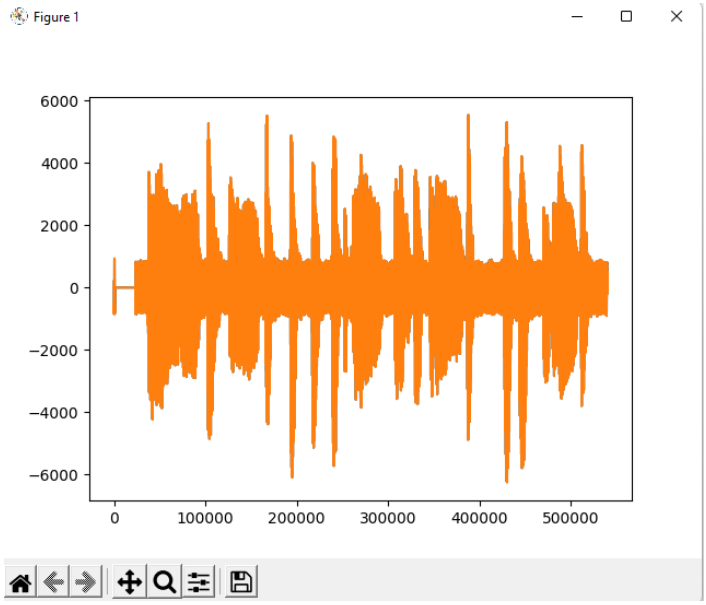


Figure 1
6000
4000
2000
0
−2000
−4000
−6000
0
100000
200000
300000
400000
500000

Double:

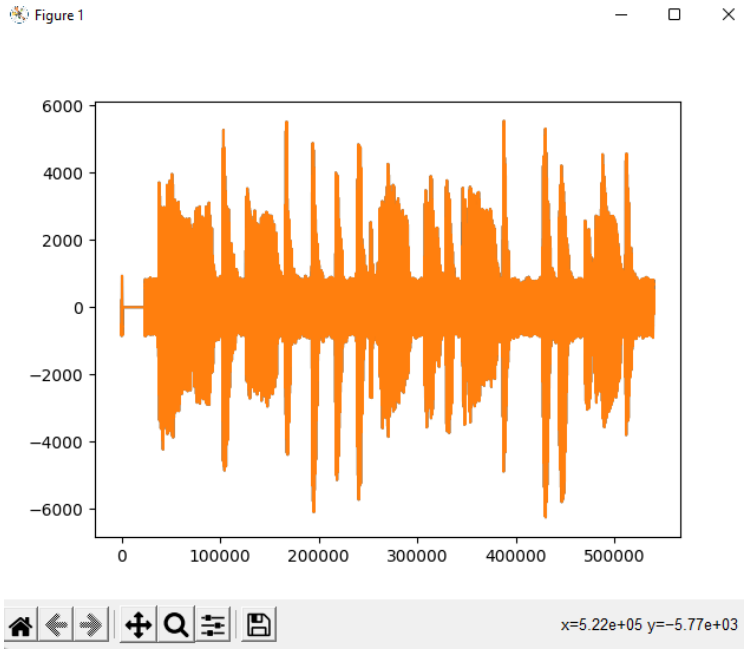
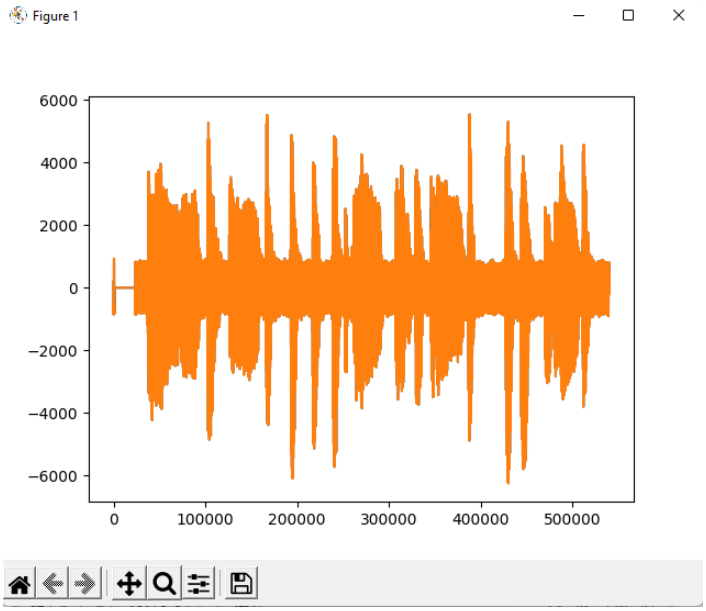
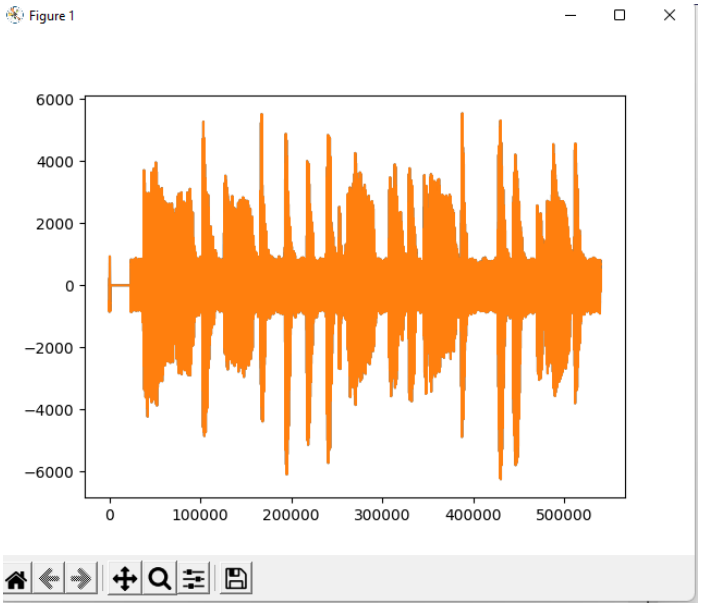


Figure 1
6000
4000
2000
0
−2000
−4000
−6000
0
100000
200000
300000
400000
500000
x=5.22e+05 y=−5.77e+03

Flipl:



Flipud:



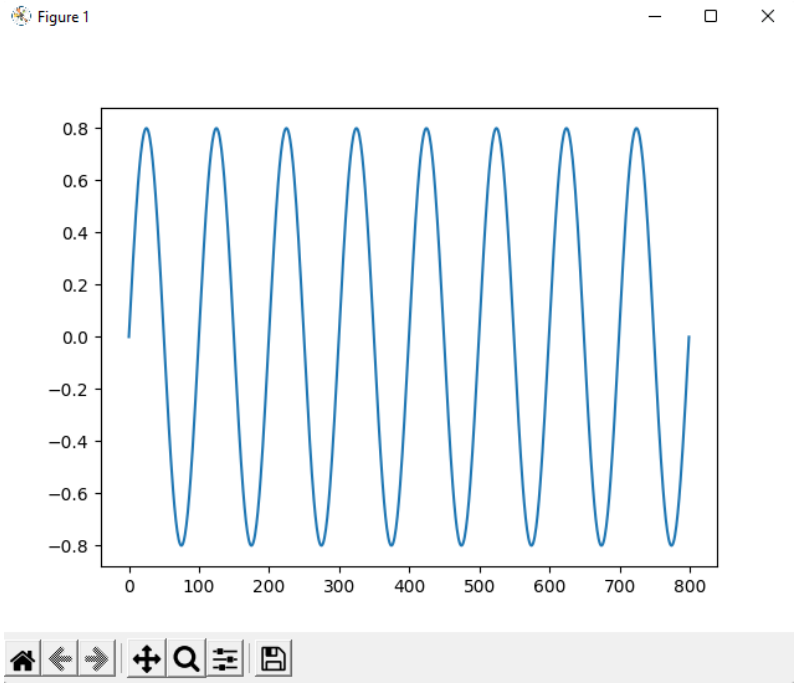
8) Crie uma função no Matlab que gere um modelo discreto de um sinal de eco $s_e(t)=\alpha s(t-T)$se(t)=αs(t−T), onde αα é o fator de atenuação, TT, o atraso (em segundos) e $s(t)$s(t), o sinal original. Assuma que $0\leq\alpha\leq1$0≤α≤1 e que $T\geq0$T≥0. O sinal resultante, com eco, é $r(t)=s(t)+s_e(t)$r(t)=s(t)+se(t).Gere sinais com eco a partir de uma gravação de voz (teste antes com um ou dois impulsos para verificar seu código). Experimente com diferentes valores de atenuação e atraso (teste, por exemplo, com fator de atenuação de 0,65 e atraso de 250 ms). Qual o impacto sobre o sinal sintetizado?

Código e resultado;

```
############################################################
##
##  Created on: Abril 26, 2022
##      Author: Marcelo Brancalhão Gaspar
##      Instituto Federal de Santa Catarina
##      DSP 2 - Fernando Santana Pacheco
##
############################################################
import numpy as np
import matplotlib.pyplot as plt
import simpleaudio as sa
import math as math
from scipy.io import wavfile
def tonegen(freq,duration=0.1,amp=1, fs=8000):
    t = np.linspace(0., duration, int(fs*duration))
    x = amp*(np.sin(2*np.pi*freq*t))
    audio =x*(2**15 - 1) / np.max(np.abs(x))
    audio =audio.astype(np.int16)
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
    return x
s440 = tonegen(80, 0.1, 0.8, 8000)
plt.plot(s440)
plt.show()
def echo(signal, attenuation, delay, fs):
     delayer = np.zeros(math.ceil (delay * fs))
     attSig = attenuation*signal
     delSin = np.append(delayer,attSig)
     sig = np.append(signal,delayer)
```

```
    return sig+delSin
fs,voz = wavfile.read('voz.wav')
echo1 = echo(voz, 0.65, 0.25, 8000)
plt.plot(echo1)
plt.show()
echo2 = echo(voz, 0.5, 0.4, 8000)
plt.plot(echo2)
plt.show()
echo3 = echo(voz, 0.9, 1, 4000)
plt.plot(echo1)
plt.show()
```

Original



Eco1

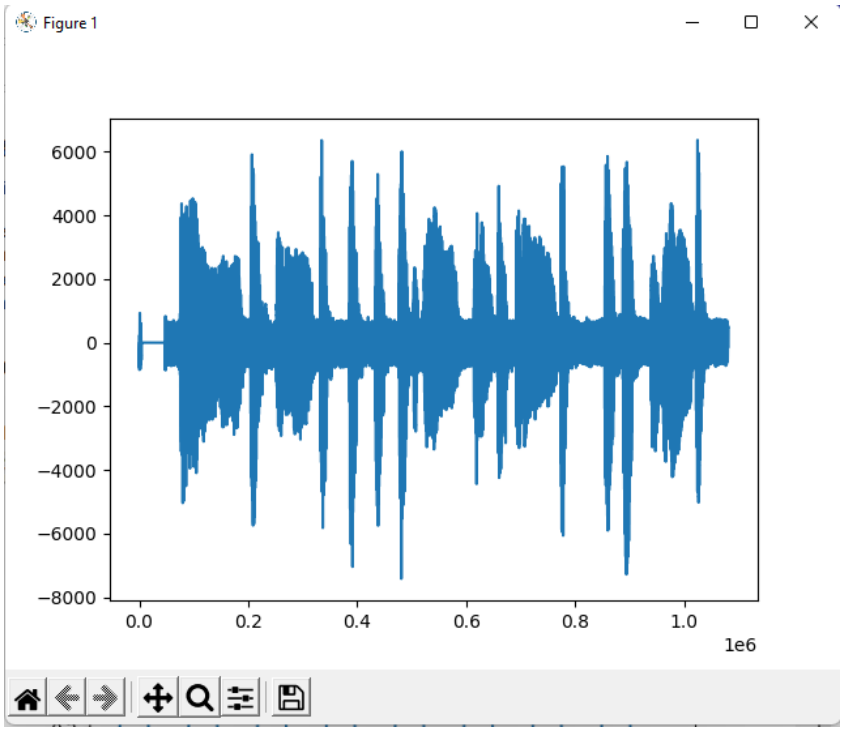


Figure 1
6000
4000
2000
0
−2000
−4000
−6000
−8000
0.0
0.2
0.4
0.6
0.8
1.0
1e6

Eco 2

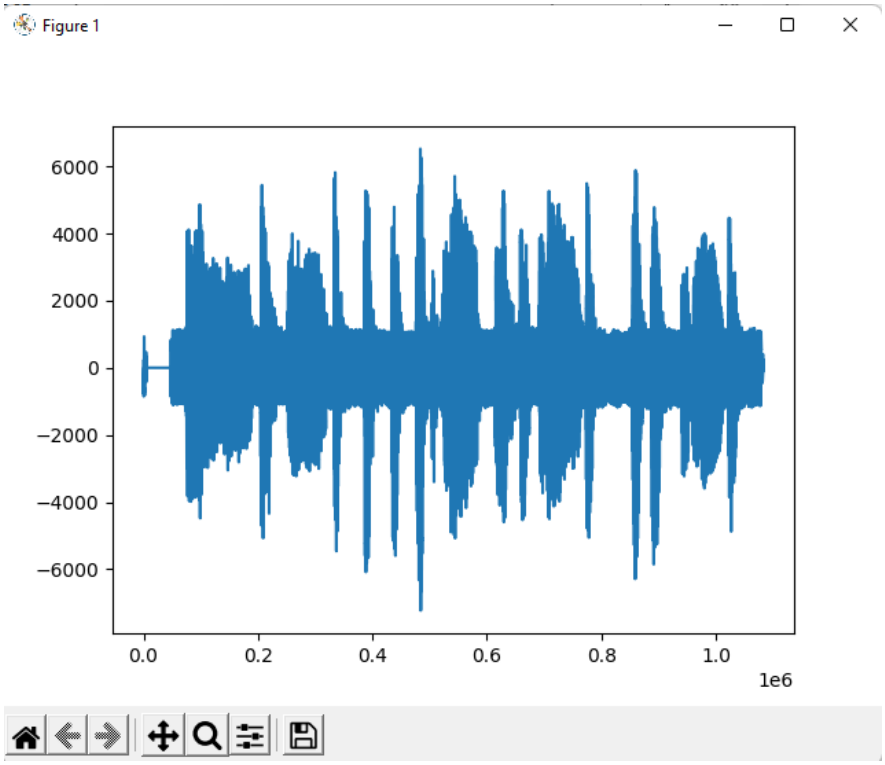
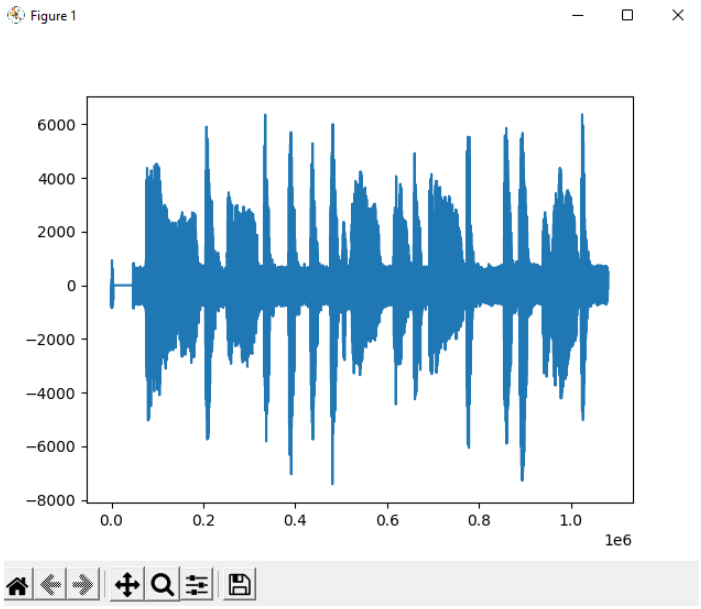


Figure 1
6000
4000
2000
0
−2000
−4000
−6000
0.0
0.2
0.4
0.6
0.8
1.0
1e6

Eco 3



9) Acesse o banco de respostas ao impulso de ambientes em http://www.openairlib.net, link IR Data. Observe que as respostas são longas, geralmente maiores que 2 segundos.Escolha uma resposta, baixe o arquivo e faça a operação de convolução do trecho de voz que você gravou com essa resposta de ambiente (certifique-se que as frequências de amostragem são as mesmas) usando `numpy.convolve`. Se as frequências de amostragem forem diferentes, você pode reamostrar um dos arquivos (áudio ou resposta) no Ocenaudio.Meça o tempo de processamento para vários tamanhos de resposta, usando uma das formas descritas em https://cmdlinetips.com/2018/01/two-ways-to-compute-executing-time-in-python/. Faça um gráfico do tempo de processamento em função do tamanho da resposta (entre 100 ms e 3 s, com pelo menos 5 pontos; para fazer isso, você irá cortar a resposta). Comente.