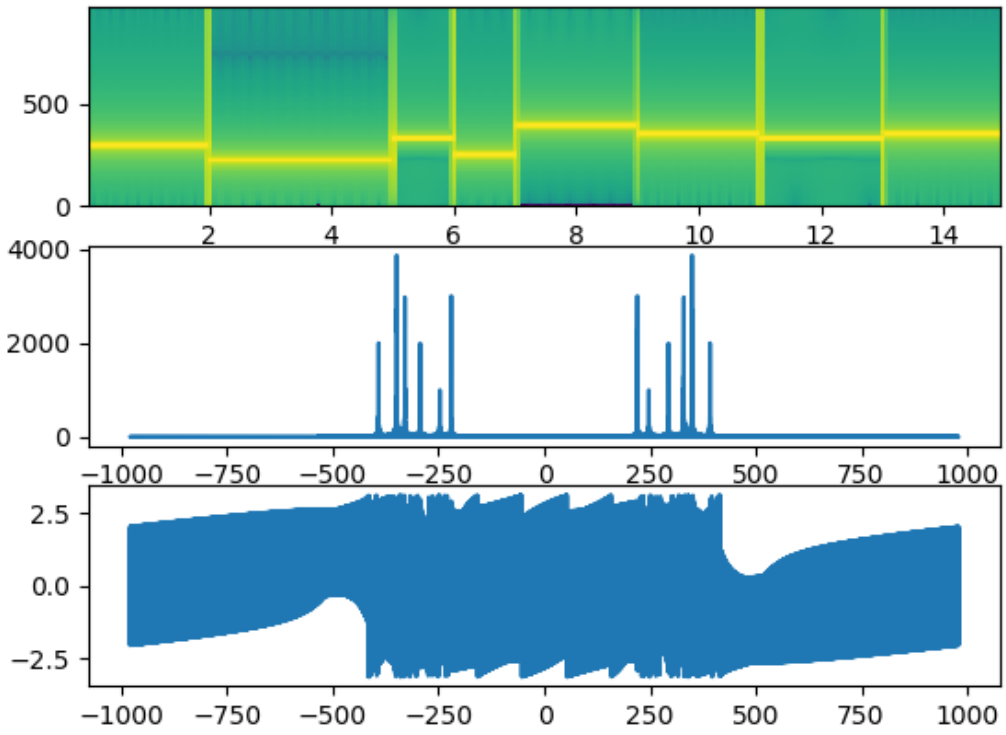
Licenciatura Engenharia Informática e Multimédia

Processamento Digital de Sinais – 1718SI

Ano letivo 2017/2018

Síntese e Análise de Sinais Áudio



Docente:

André Lourenço

Trabalho realizado por:

Miguel Távora Nº45102

Pedro Dias Nº45170

Sérgio Lopes Nº43740

Turma: 21D

Índice

Resolução de exercícios e gráficos

Grupo I:

I. Geração de Sinusóides

1. Considere o sinal periódico $x(t) = A_1 \cos(2\pi f_1 t)$

   a) Implemente uma função que gere este sinal, que produza a sua representação gráfica e um ficheiro *wav* do mesmo.

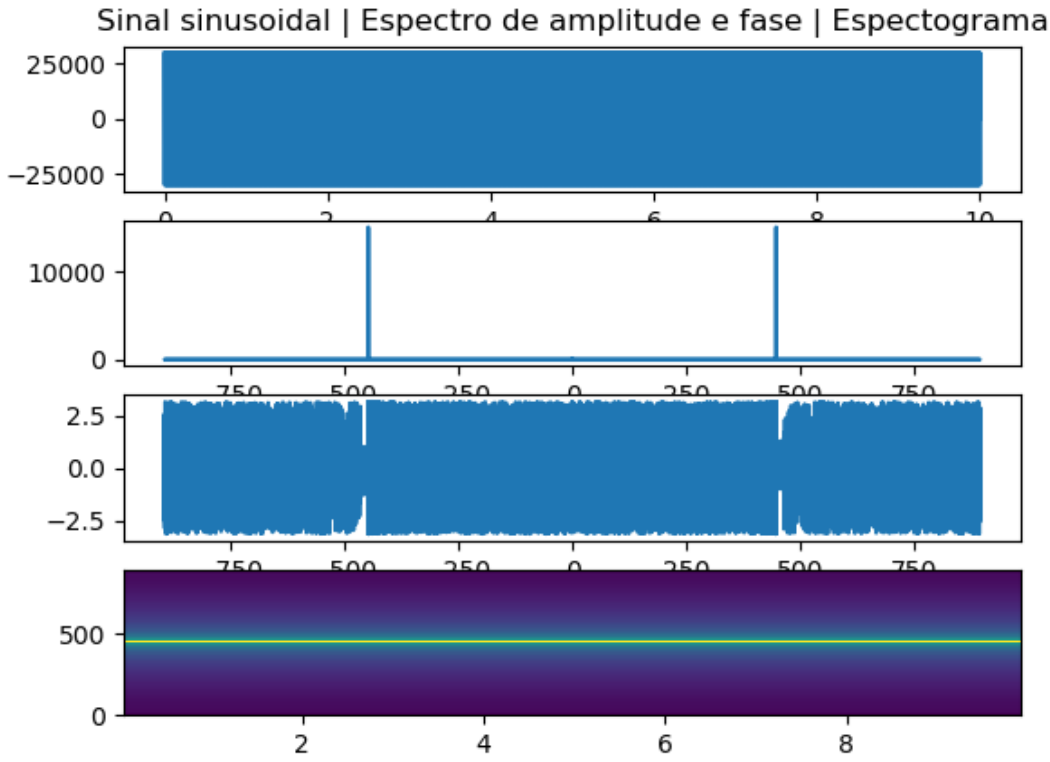
   b) Crie um script onde visualize uma sinusóide, cuja frequência é a média dos números dos alunos do grupo a dividir por 100.0 e que a amplitude é o número de alunos do grupo.

   b) Analise o sinal no domínio da frequência, calculando analiticamente o espectro de $x(t)$ e usando a *fft* no python para representar o seu espectro (módulo e fase).

   c) Use o python para visualizar o espectrograma deste sinal. Explique a diferenças entre o espectro e espectrograma.

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav


#Função que gera o sinal referido e cria um ficheiro wav
def f_sinal(A,f,d): #A é a amplitude | f é a frequência | d é a duração
    fs = 4 * f #fs é 4 vezes a fmax
    ts = 1/fs
    t = np.arange(0,d,ts)
    x_t = A * np.cos(2*np.pi*f*t)
    wav.write('Ex1.wav',fs,x_t.astype('int16'))
    plt.subplot(4,1,1)
    plt.plot(t,x_t)


#Sinusóide com frequência e amplitude pedidas
Frequencia = 447 #((43740+45102+45170)/3)/100 = 446.70
Amplitude = 30000 #Amplitude = 3 não se ouve
Duracao = 10
sinal = f_sinal(Amplitude,Frequencia,Duracao)


#Espectro do sinal
fs = 4 * Frequencia
ts = 1/fs
t = np.arange(0,Duracao,ts)
x_t = Amplitude * np.cos(2*np.pi*Frequencia*t)


F = np.fft.fft(x_t)
freqs = np.fft.fftfreq(len(F),ts)
plt.title("Sinal sinusoidal | Espectro de amplitude e fase | Espectograma")
plt.subplot(4,1,2)
plt.plot(freqs,np.abs(F)/len(F))
plt.subplot(4,1,3)
plt.plot(freqs,np.angle(F))
```

```
#Espectograma do sinal
plt.subplot(4,1,4)
plt.specgram(x_t, Fs = fs)
plt.show()

#Diferenças entre o Espectro e o Espectrograma

#O espectro é uma representação de um sinal no domínio da frequência
#que permite ver a relação entre esta e a amplitude/fase enquanto o
#espectrograma é uma forma de analisar o sinal em relação a tempo-frequência.
```



Grupo II:

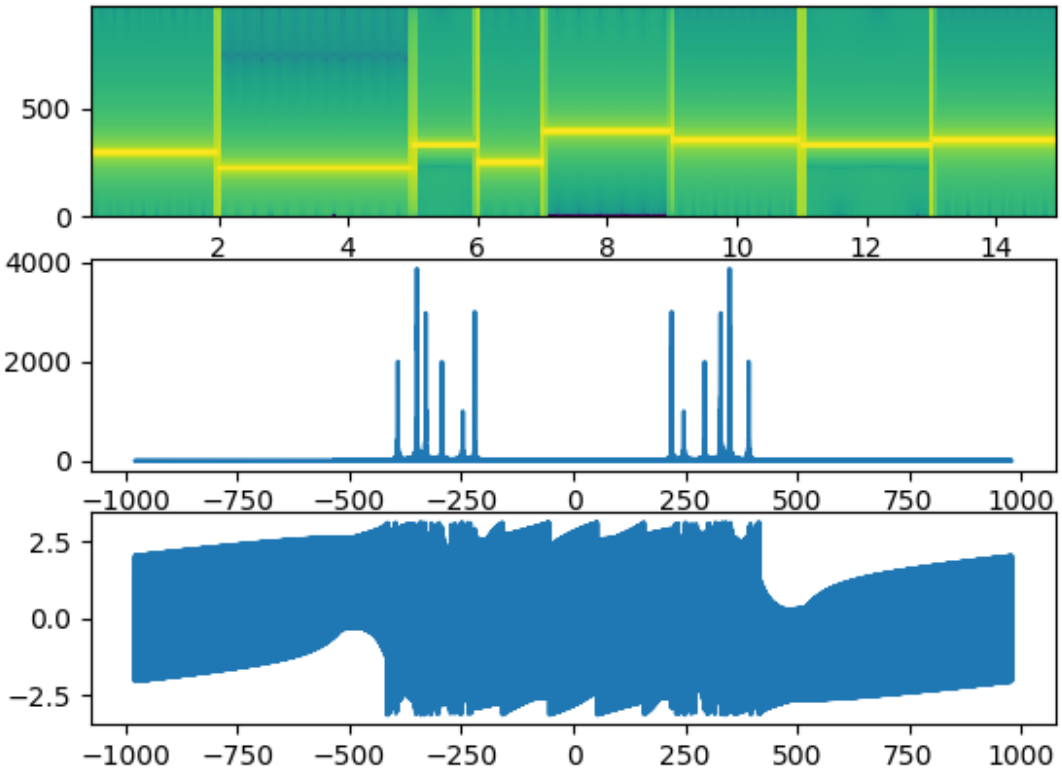
II. Sintese de notas musicais

2. Crie uma função que crie uma composição musical. A função tem como argumentos de entrada uma lista com uma sequência de tuplos (nota, número de unidades de tempo) e a unidade de tempo. A nota é expressa usando a notação ABC, exemplo: (('c', 4), ('e', 4), ('g', 4), ('c5', 1))). Deve retornar um array com as amplitudes com as amplitudes instantaneas correspondentes à composição musical. Crie uma composição onde varie a frequência e a duração das notas. Visualize o sinal no domínio do tempo e na frequência (espectro e espectrograma). Oiça o sinal criado.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav


#Frequências de cada nota
notas_musicais = {'a': 27.5,'b': 30.86,'c': 32.7,'d': 36.7,'e': 41.2 , 'f':43.65 , 'g' : 48.9,
                  'a2': 55,'b2': 61.7,'c2': 65.4,'d2': 73.4,'e2': 82.4 , 'f2':87.3 , 'g2' : 97.9,
                  'a3': 110,'b3': 123.47,'c3': 130.8,'d3': 146.83,'e3': 164.8 , 'f3':174.6 , 'g3' : 196,
                  'a4': 220,'b4': 246.94,'c4': 261.63,'d4': 293.66,'e4': 329.63 , 'f4':349.23 , 'g4' : 392,
                  'a5': 440,'b5': 493.88,'c5': 523.25,'d5': 587.33,'e5': 659.26 , 'f5':698.46 , 'g5' : 783.99,'a6': 880}

#Composição da música que se pode ouvir (pode ser alterado)
composicao =[('d4',2),('a4',3),('e4',1),('b4',1),('g4',2),('f4',2),('e4',2),('f4',2)]


#Função que cria o sinal do som da composição
def composicaoMusical(composicao,UnidadesTempo):

    print("unidade tempo: " + UnidadesTempo)
    frequencia_max = 0

    #for que encontra a maior frequência da composição para calcular a frequência de amostragem
    for n in composicao:

        frequencia = notas_musicais[n[0]]

        if (frequencia > frequencia_max):
            frequencia_max = frequencia

    fs = 5 * frequencia_max
    print(fs)
    frequencias = []

    musica = []
```

```
        #para cada uma das notas cria-se um sinal e faz-se hstack
        for n in composicao:
            t = np.arange(0,n[1],1/fs)
            nota = n[0]
            frequencia= notas_musicais[nota]
            frequencias.append(frequencia)
            x = 30000*np.cos(2*np.pi*frequencia*t)
            musica= np.hstack((musica,x))

        wav.write('Ex2.wav', int (fs),musica.astype('int16'))


        #Espectrograma
        plt.subplot(3,1,1)
        plt.specgram(musica,Fs = fs)


        #Espectros
        F = np.fft.fft(musica)
        freqs = np.fft.fftfreq(len(F),1/fs)
        plt.subplot(3,1,2)
        plt.plot(freqs,np.abs(F)/len(F))#espectro de amplitude
        plt.subplot(3,1,3)
        plt.plot(freqs,np.angle(F))
        plt.show()
        return frequencias


    #Frequências instantâneas
    unidades = "segundos"
    frequencias = composicaoMusical(composicao,unidades)
    print(frequencias)
```

```
unidade tempo: segundos
1960
[293.66, 220, 329.63, 246.94, 392, 349.23, 329.63, 349.23]
```

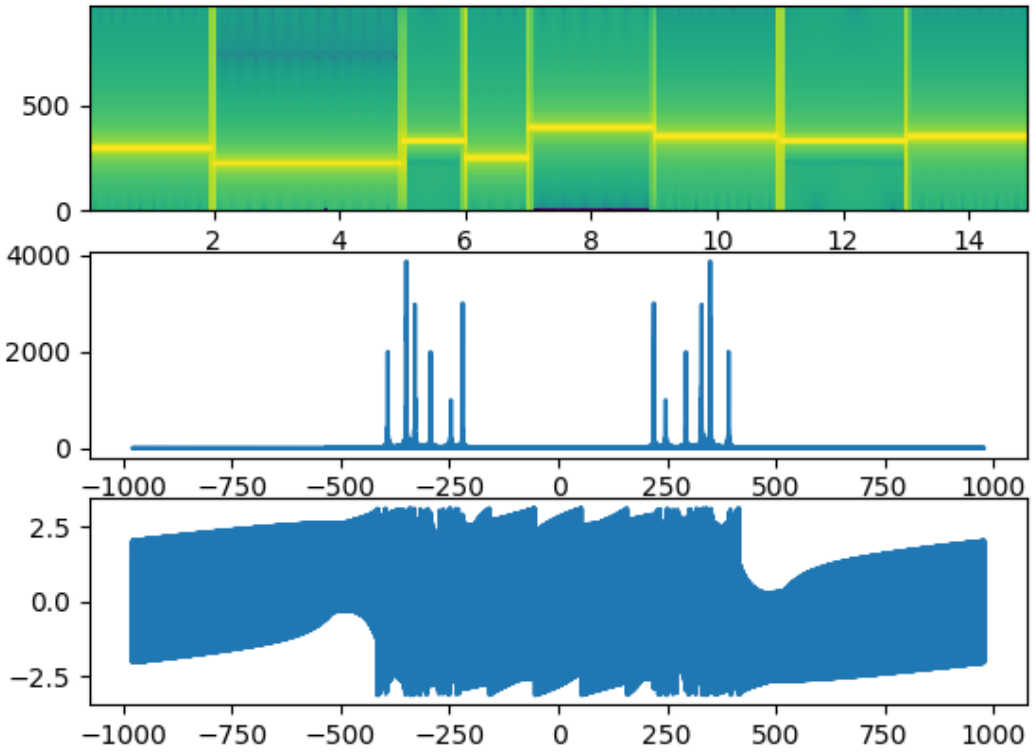


3. Implemente um envelope ADSR na função anterior. Deverá ser possível parametrizar o tempo de Attack/Decay/Sustain/Release em percentagem do tempo da nota.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav


#Frequências de cada nota
notas_musicais = {'a': 27.5,'b': 30.86,'c': 32.7,'d': 36.7,'e': 41.2 , 'f':43.65 , 'g' : 48.9,
                  'a2': 55,'b2': 61.7,'c2': 65.4,'d2': 73.4,'e2': 82.4 , 'f2':87.3 , 'g2' : 97.9,
                  'a3': 110,'b3': 123.47,'c3': 130.8,'d3': 146.83,'e3': 164.8 , 'f3':174.6 , 'g3' : 196,
                  'a4': 220,'b4': 246.94,'c4': 261.63,'d4': 293.66,'e4': 329.63 , 'f4':349.23 , 'g4' : 392,
                  'a5': 440,'b5': 493.88,'c5': 523.25,'d5': 587.33,'e5': 659.26 , 'f5':698.46 , 'g5' : 783.99,'a6': 880}

#Composição da música que se pode ouvir (pode ser alterado)
composicao =[('d4',2),('a4',3),('e4',1),('b4',1),('g4',2),('f4',2),('e4',2),('f4',2)]



#Função que cria o sinal do som da composição
def composicaoMusical(composicao,UnidadesTempo):

    print("unidade tempo: " + UnidadesTempo)
    frequencia_max = 0

    #for que encontra a maior frequência da composição para calcular a frequência de amostragem
    for n in composicao:

        frequencia = notas_musicais[n[0]]

        if (frequencia > frequencia_max):
            frequencia_max = frequencia

    fs = 5 * frequencia_max
    print(fs)
    frequencias = []

    musica = []
```

```
    #para cada uma das notas cria-se um sinal e faz-se hstack
    for n in composicao:
        t = np.arange(0,n[1],1/fs)
        nota = n[0]
        frequencia= notas_musicais[nota]
        frequencias.append(frequencia)
        x = 30000*np.cos(2*np.pi*frequencia*t)
        musica= np.hstack((musica,x))

    wav.write('Ex3_composicao.wav', int (fs),musica.astype('int16'))


    #Espectrograma
    plt.subplot(3,1,1)
    plt.specgram(musica,Fs = fs)


    #Espectros
    F = np.fft.fft(musica)
    freqs = np.fft.fftfreq(len(F),1/fs)
    plt.subplot(3,1,2)
    plt.plot(freqs,np.abs(F)/len(F))#espectro de amplitude
    plt.subplot(3,1,3)
    plt.plot(freqs,np.angle(F))
    plt.show()
    return frequencias

def envelope_ADSR(A=[]):

    duracao =(.1, .2, .6, .1)
    d_attack = np.floor(A*duracao[0])
    d_decay = np.floor(A*duracao[1])
    d_sustain = np.floor(E*duracao[2])
    d_release = A-d_arrack-d_decay-n_sustain

    attack = np.linspace (0, 1, n_attack)
    decay = np.linspace(1, .8, n_decay)
    sustain = np.ones(n_sustain)*.8
    release = np.linspace(.8, 0, n_release)

    Envelope = np.hstack((attack, decay, sustain, release))


     return Envelope



#Frequências instantâneas
unidades = "segundos"
frequencias = composicaoMusical(composicao,unidades)
print(frequencias)
```

```
unidade tempo: segundos
1960
[293.66, 220, 329.63, 246.94, 392, 349.23, 329.63, 349.23]
```



4. Use o PySynth (`https://mdoege.github.io/PySynth`) para experimentar outro tipo de sintetizadores: PySynth "A"/"B"/"E"/"S". Use a mesma composição e visualize o sinal no domínio do tempo e na frequência (espectro e espectrograma). Oiça o sinal criado.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav
import pysynth as pys
import pysynth_b as pysb
import pysynth_s as pyss
import pysynth_e as pyse


comp=[('d4',2),('a4',3),('e4',1),('b4',1),('g4',2),('f4',2),('e4',2),('f4',2)]

pys.make_wav(comp)
pysb.make_wav(comp)
pyss.make_wav(comp)
pyse.make_wav(comp)

plt.figure("Espectro A")
Fs, musica_a = wav.read('out.wav')
Z = np.fft.fft(musica_a)
freqs = np.fft.fftfreq(len(Z),1/Fs)
plt.subplot(2,1,1)
plt.plot(freqs,np.abs(Z)/len(Z))
plt.subplot(2,1,2)
plt.plot(freqs,np.angle(Z))
```

```
plt.figure("Espectro B")
Fs, musica_b = wav.read('outb.wav')
Z = np.fft.fft(musica_b)
freqs = np.fft.fftfreq(len(Z),1/Fs)
plt.subplot(2,1,1)
plt.plot(freqs,np.abs(Z)/len(Z))
plt.subplot(2,1,2)
plt.plot(freqs,np.angle(Z))

plt.figure("Espectro S")
Fs, musica_s = wav.read('outs.wav')
Z = np.fft.fft(musica_s)
freqs = np.fft.fftfreq(len(Z),1/Fs)
plt.subplot(2,1,1)
plt.plot(freqs,np.abs(Z)/len(Z))
plt.subplot(2,1,2)
plt.plot(freqs,np.angle(Z))

plt.figure("Espectro E")
Fs, musica_e = wav.read('oute.wav')
Z = np.fft.fft(musica_e)
freqs = np.fft.fftfreq(len(Z),1/Fs)
plt.subplot(2,1,1)
plt.plot(freqs,np.abs(Z)/len(Z))
plt.subplot(2,1,2)
plt.plot(freqs,np.angle(Z))

plt.figure("Espectrograma A")
plt.specgram(musica_a,Fs=Fs,NFFT=2048,noverlap=0)
axes = plt.gca()
axes.set_ylim([0,2500])


plt.show()
```

```
Piano key frequencies (for equal temperament):
Key number      Scientific name Frequency (Hz)
         1                   A0          27.50
         2                  A#0          29.14
         3                   B0          30.87
         4                   C1          32.70
         5                  C#1          34.65
         6                   D1          36.71
         7                  D#1          38.89
         8                   E1          41.20
         9                   F1          43.65
        10                  F#1          46.25
        11                   G1          49.00
        12                  G#1          51.91
        13                   A1          55.00
        14                  A#1          58.27
        15                   B1          61.74
        16                   C2          65.41
        17                  C#2          69.30
        18                   D2          73.42
        19                  D#2          77.78
        20                   E2          82.41
        21                   F2          87.31
        22                  F#2          92.50
        23                   G2          98.00
        24                  G#2         103.83
        25                   A2         110.00
        26                  A#2         116.54
        27                   B2         123.47
        28                   C3         130.81
        29                  C#3         138.59
        30                   D3         146.83
        31                  D#3         155.56
        32                   E3         164.81
```

36	G#3	207.65
37	A3	220.00
38	A#3	233.08
39	B3	246.94
40	C4	261.63
41	C#4	277.18
42	D4	293.66
43	D#4	311.13
44	E4	329.63
45	F4	349.23
46	F#4	369.99
47	G4	392.00
48	G#4	415.30
49	A4	440.00
50	A#4	466.16
51	B4	493.88
52	C5	523.25
53	C#5	554.37
54	D5	587.33
55	D#5	622.25
56	E5	659.26
57	F5	698.46
58	F#5	739.99
59	G5	783.99
60	G#5	830.61
61	A5	880.00
62	A#5	932.33
63	B5	987.77
64	C6	1046.50
65	C#6	1108.73
66	D6	1174.66
67	D#6	1244.51
68	E6	1318.51
69	F6	1396.91
70	F#6	1479.98
71	G6	1567.98
72	G#6	1661.22
73	A6	1760.00
74	A#6	1864.66
75	B6	1975.53
76	C7	2093.00
77	C#7	2217.46
78	D7	2349.32
79	D#7	2489.02
80	E7	2637.02
81	F7	2793.83
82	F#7	2959.96
83	G7	3135.96
84	G#7	3322.44
85	A7	3520.00
86	A#7	3729.31
87	B7	3951.07
88	C8	4186.01

Grupo III:

III. Análise de sinais musicais.

4. Pretende-se que seja realizada uma função que realize o processo inverso à sintese, isto é, receba um wav produzido por um dos sintetizadores anteriores e produza a notação ABC correspondente.

```
import scipy.io.wavfile as wav
import numpy as np
import matplotlib.pyplot as plt

notas = {'a': 27.5,'b': 30.86,'c': 32.7,'d': 36.7,'e': 41.2 , 'f':43.65 , 'g' : 48.9, 'a2': 55,'b2': 61.7,
        'c2': 65.4,'d2': 73.4,'e2': 82.4 , 'f2':87.3 , 'g2' : 97.9,'a3': 110,'b3': 123.47,'c3': 130.8,
        'd3': 146.83,'e3': 164.8 ,'f3':174.6 , 'g3' : 196,'a4': 220,'b4': 246.94,'c4': 261.63,'d4': 293.66,
        'e4': 329.63 , 'f4': 349.23 , 'g4' : 392,'a5': 440,'b5': 493.88,'c5': 523.25,'d5': 587.33,'e5': 659.26 ,
        'f5': 698.46 , 'g5' : 783.99,'a6': 880,'b6': 987.77,'c6': 1046.5,'d6': 1174.7,'e6': 1318.5 , 'f6':1396.9 ,
        'g6' : 1568}

Fs, musica = wav.read("Ex2.wav")
espectro = np.fft.fft(musica)
spect, f, t, ii = plt.specgram(musica, Fs = Fs, NFFT=512, noverlap = 0)

def Sem_duplicados(x):
        i = 1
        while i < len(x):
                if x[i] == x[i-1]:
                        x.pop(i)
                        i = i - 1
                i = i + 1
        return x

frequencia_notas = []
for a in range(0, spect.shape[1], 1):
        plt.plot(f,spect[:,a])
        frequencia_notas.append(format(3.8399*np.argmax(spect[:,a]),'.2f'))

frequencia_notas = Sem_duplicados(frequencia_notas)

print(frequencia_notas)
```

```
['295.67', '218.87', '330.23', '249.59', '391.67', '349.43', '330.23', '349.43']
```

Conclusão

Neste trabalho aprendemos a criar outro tipo de ficheiros que não só criam gráficos demonstrativos como também geram músicas a partir de frequências customizadas.

Foi criado no Python ficheiros com espectrogramas de amplitude e de fase e também ficheiros que criam frequências para notas através da leitura de outros ficheiros.