CIM - Trabalho 2

Síntese de sinais musicais

CIM 2022/2023 **David Rainho** up201906994
Turma: 1MEEC_T02 **Jorge Pais** up201904841

1 Geração de uma onda dente de serra

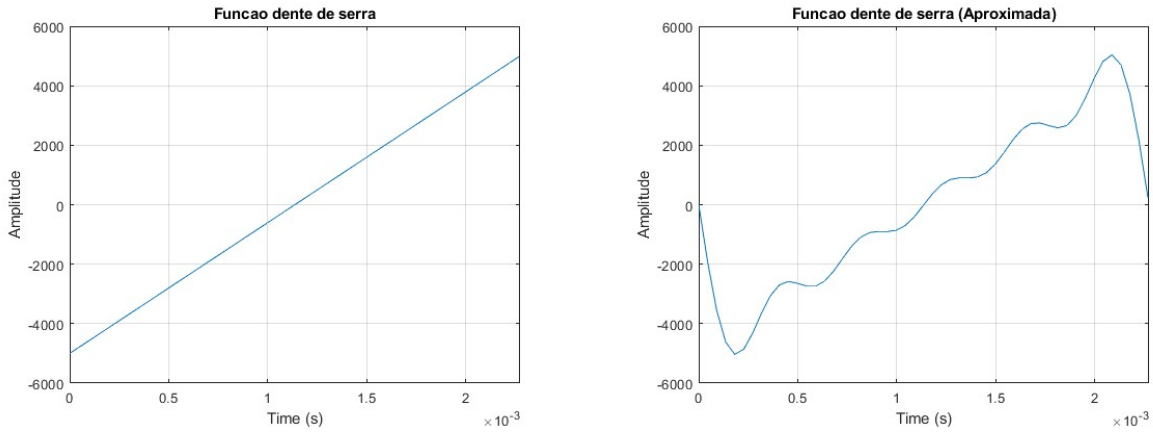
Funções contínuas periódicas que obedeçam às condições de Dirichlet, podem ser representadas na forma de série de Fourier 1, na forma exponencial[1],

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\omega_0 t}, \tag{1}$$

onde $\omega_0 = \frac{2\pi}{T_0}$ é a frequência fundamental e c_k um coeficiente complexo.
Neste exercício, pretende-se aproximar a função dente de serra, com período T = 1/440 Hz, através dos primeiros 5 termos da série de Fourier correspondente 2.

$$x(t) = \sum_{k=1}^{5} \frac{2A}{\pi k} \sin\left(\frac{2\pi}{T} kt\right) \tag{2}$$

A figura 1, representa um período das ondas ideal e aproximada, geradas em MATLAB, código `ex1.m`.



(a) (b)

Figura 1: Nesta figura são visíveis as funções dente de serra (a) ideal e (b) aproximada.

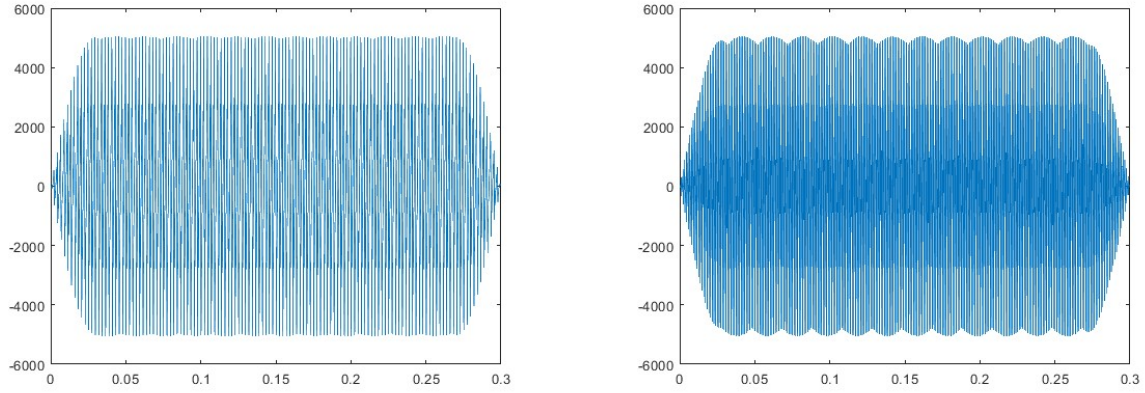
A aproximação obedece aos seguintes critérios:

- frequência de amostragem, Fs=22050 Hz
- amplitude, A=5000.

2 Geração ondas com envelopes/modulação de amplitude

Pretende-se neste exercício criar uma função que gere uma nota musical com forma de onda dente de serra e uma transição suave (baseada em 1/4 de período de onda sinusoidal) entre o valor máximo e mínimo, respetivamente A e 0. Além disso, deve ser possível alterar a nota (relativas a LA4, por potências de $\sqrt[12]{2}$), duração (em segundos) e frequência de amostragem. O código MATLAB gerado para realizar este exercício está no anexo com a designação `geranota.m`.

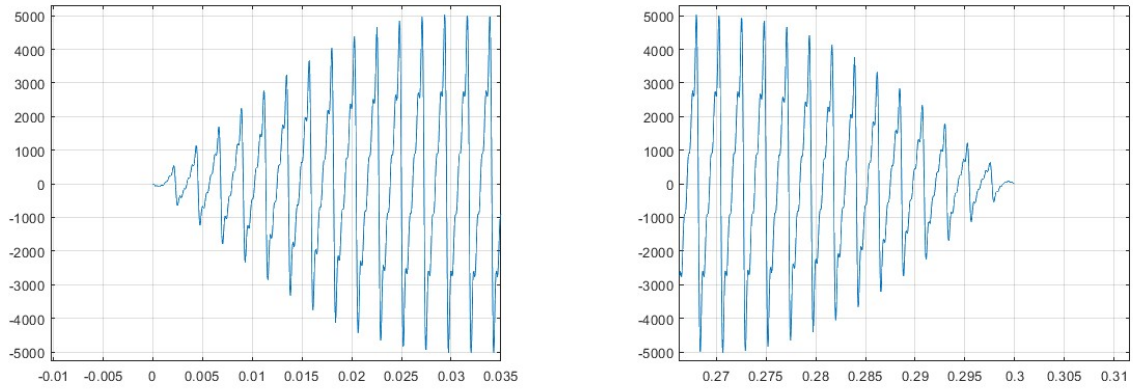
[1] No caso de apresentar descontinuidades, a série de Fourier converge, nesses pontos, para o valor médio



(a) (b)

Figura 2: Apresentam-se as funções de onda para as notas (a) LA4 (440 Hz) e (b) LA5 (880 Hz).

O *fadein* e *fadeout* ocorrem, cada um, durante um décimo da duração do sinal. Para verificar a correta implementação, gerou-se uma onda com duração de 0.3 segundos, traduzindo-se num *fadein* e *fadeout* nos intervalos 0 a 0.03 e 0.27 a 0.3, respetivamente. Na figura 3, verifica-se a curva esperada.



(a) (b)

Figura 3: Nas figuras estão as curvas de (a) *fadein* e (b) *fadeout*.

3 Implementação de uma melodia simples

No seguinte exercício é pretendido que uma pequena melodia seja sintetizada utilizando a função `geranota()` implementada previamente. A pauta a ser utilizada é a seguinte:

```
pauta = [do re mi fa fa fa do re do re re re do sol fa mi mi mi do re mi fa];
```

É pretendido que na melodia todas as notas tenham a mesma duração. Para tal, foi primeiro definida uma função `decodeNote()` de forma a converter uma string de nota (na notação "ABC+oitava", dó central = "C3") na frequência correspondente em temperamento igual.

Através desta função, percorreu-se o vetor da pauta e sintetizou-se cada uma das notas, concatenando a onda resultante no vetor `samples`. No final, adiciona-se meio segundo de silêncio, normalizam-se os valores das amostras para $[-1; 1]$ e o resultado é escrito no ficheiro `ex3.wav`. Na figura 4 é possível observar a forma de onda das primeiras três notas sintetizadas (dó, ré e mi).

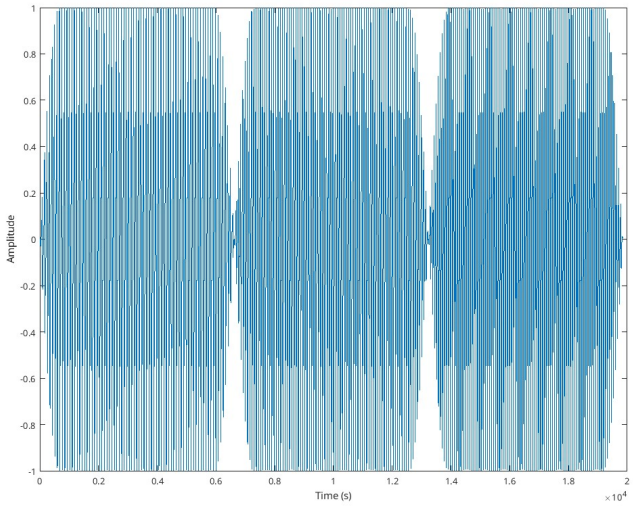


Figura 4: Forma de onda resultante do exercício 3

4 Implementação de uma pauta musical

No quarto problema é pretendido que uma qualquer pauta musical fosse implementada. Para o efeito, escolheu-se o popular tema Blue (Da Ba dee) dos Eiffel 65, que foi arranjado com duas vozes, uma para a melodia principal e outra para o baixo. Uma ideia do arranjo é apresentada na Figura 5:

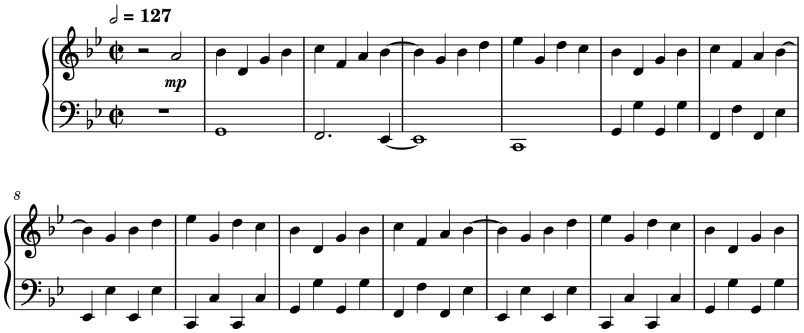


Figura 5: Pauta com o arranjo da Blue

Uma metodológica bastante similar ao exercício anterior foi tomada para cada uma das partes, com a exceção de que para a além de se utilizar um vetor para guardar que notas são tocadas, também se guardou a duração de cada nota. De forma a reduzir a verbosidade do código MATLAB, estes vetores encontram-se no ficheiro `sheet_blue.m`.
No ficheiro `ex4_blue.m`, estes vetores contendo as informações das notas de cada voz são percorridos e cada uma das vozes é sintetizada individualmente. No final as amostras de ambas são somadas, e o resultado final é normalizado de [-1, 1] e escrito no ficheiro `blue.wav`.

5 Análise da frequência fundamental num ficheiro áudio

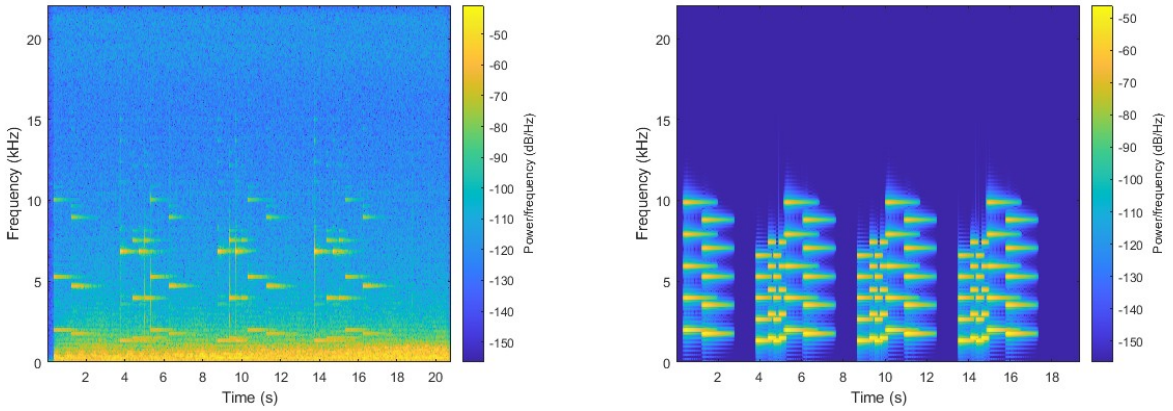
Neste problema, pretende-se determinar as notas do instrumento de percursão. Usaram-se três funções disponíveis no MATLAB:

1. função `spectogram()` (figura 6a), permitiu identificar o início e duração aproximada de cada nota, assim como adquirir alguma informação sobre os valores das frequências ("posição"relativa entre notas, recorrendo aos harmónicos),

2. função `pitch()`, permitiu determinar valores próximos, mas errados, das frequências fundamentais das notas,

3. função `pwelch()` (dentro de *for loop*), serviu para determinar efetivamente as frequências fundamentais e, por conseguinte, as notas obtidas.

Na figura 6 e reproduzindo o sinal obtido, verifica-se que foram obtidas as notas corretas. É possível observar o código criado para a resolução do problema nos Anexos com a designação `ex5.m`.



(a) (b)

Figura 6: Apresentam-se nas figuras os espetogramas (a) do aúdio original e (b) do sinal obtido.

A Código MATLAB

ex1.m

```
Fs = 22050; % Sampling frequency
A = 5000; % Wave amplitude coefficient
wave_freq = 440; % Wave fundamental frequency
K = 5; % Number of fourier coefficients

n = 0:(1/Fs):1; % Synthesize only one second
x = zeros(size(n));

for k = 1:K
    x = x - 2*A/(pi*k)*sin(2*pi*k*wave_freq.*n);
end

x = x./A; % Normalize volume

figure(1); plot(n(1:(1.1*Fs/wave_freq)), x((1:(1.1*Fs/wave_freq))))

sound(x, Fs)
```

geranota.m

```
function amostras = geranota(nota, duracao, Fs)
    T0 = 1.0/(440.0*nota); % nota = 1 -> LA4 ou A

    K = 5;
    A = 5000;

    t = 0:1/Fs:duracao;
    t = t(1:(length(t)-1)); % Retirar a ultima amostra

    x=0;
    for k=1:K
        x = x - ((2*A)/(pi*k))*sin((2*pi*k*t)/(T0));
    end

    T_sinusoide = 4*duracao/10; % Periodo da sinusoide de transicao

    t = 0:1/Fs:T_sinusoide/4;
    t = t(1:(length(t)-1));
    index = 1:length(t);

    sinusoidal = sin((2*pi/T_sinusoide)*t);
    x(index) = x(index).*sinusoidal;
    x(floor(Fs*duracao)-index+1) = x(floor(Fs*duracao)-index+1).*sinusoidal;

    amostras = x/max(abs(x)); % normalizar
end
```

ex3.m

```
Fs = 22050;

do = "C4"; re = "D4"; mi = "E4"; fa = "F4"; sol = "G4";
pauta = [do, re, mi, fa, fa, fa, do, re, do, re, ...
    re, re, do, sol, fa, mi, mi, mi, do, re, mi, fa];

samples = zeros(1);

for i = 1:length(pauta)
    note = decodeNote(pauta(i));
    samples = [samples geranota(note, 0.3, Fs)];
end
samples = [samples zeros(1, 0.5*Fs)]; % add silence to avoid clicks

%sound(samples, Fs)
audiowrite("ex3.wav", samples, Fs);


plot(1:0.9*Fs, samples(1:0.9*Fs));
ylabel("Amplitude");
xlabel("Time (s)");
```

decodeNote.m

```
function factor = decodeNote(name)
    match = regexp(name, '([A-Za-z#]+)(\d+)', 'tokens');
    letters = match{1}{1};
```

```
    numbers = str2double(match{1}{2});

    oct = 2^(numbers-4); % get the octave

    switch letters
        case "C"
            factor = oct * 2^(-9/12);
        case {"C#", "Db"}
            factor = oct * 2^(-8/12);
        case "D"
            factor = oct * 2^(-7/12);
        case {"D#", "Eb"}
            factor = oct * 2^(-6/12);
        case "E"
            factor = oct * 2^(-5/12);
        case "F"
            factor = oct * 2^(-4/12);
        case {"F#", "Gb"}
            factor = oct * 2^(-3/12);
        case "G"
            factor = oct * 2^(-2/12);
        case {"G#", "Ab"}
            factor = oct * 2^(-1/12);
        case "A"
            factor = oct;
        case {"A#", "Bb"}
            factor = oct * 2^(1/12);
        case "B"
            factor = oct * 2^(2/12);
    end
end
```

ex4_blue.m

```
BPM = 254; T_quarter = 60/BPM;
Fs = 22050;

% Load the notes+length for both voices
sheet_blue;

samples1 = zeros(1, floor(T_quarter*time1(1) * Fs)); % Initial Rest
for i = 2:length(voice1)
    nota = decodeNote(voice1(i));
    samples1 = [samples1 geranota(nota, T_quarter*time1(i), Fs)];
end

samples2 = zeros(1, floor(T_quarter*time2(1) * Fs));
for i = 2:length(voice2)
    nota = decodeNote(voice2(i));
    samples2 = [samples2 geranota(nota, T_quarter*time2(i), Fs)];
end

levelVoice1 = 0.5;
levelVoice2 = 0.5;
% Mix both tracks (and compensate for inconsistent note lenght)
samples = levelVoice1*samples1 + levelVoice2*samples2(1:length(samples1));
% Zero-padding + amplitude normalization to [-1, 1];
samples = [samples zeros(1, 100)]/max(abs(samples));

sound(samples, Fs);
audiowrite("blue.wav", samples, Fs);
```

sheet_blue.m

```
% All note duration relative to quarter note (seminima)
voice1 = ["-", "A4", ...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5", ...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5", ...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5", ...
    ...
```

```
    "Bb4", "D4", "G4", "Bb4",... % Variation
    "A4", "C4", "F4", "G4", ...
    "C4", "F4", "G4",...
    "F4", "G4", "A4",...
    ...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5",...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5",...
    "Bb4", "D4", "G4", "Bb4",...
    "C5", "F4", "A4", "Bb4", ...
    "G4", "Bb4", "D5",...
    "Eb5", "G4", "D5", "C5",...
    ...
    "Bb4", "D4", "G4", "Bb4",...
    "A4", "C4", "F4", "G4", ...
    "C4", "F4", "G4",...
    "F4", "G4", "A4", "Bb4"];
time1 = [2, 2, ...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 1,...
    1, 1, 1, 2,...
    1, 1, 2,...
    1, 1, 1,...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1,...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1,...
    1, 1, 1, 1, ...
    1, 1, 1, 2, ...
    1, 1, 1, ...
    1, 1, 1, 1, ...
    1, 1, 1, 1,...
    1, 1, 1, 2,...
    1, 1, 2,...
    1, 1, 1, 8];
voice2 = ["-", ...
    "G2", "F2", "Eb2", "C2",...
    "G2", "F2", "Eb2", "C2",...
    "G2", "F2", "Eb2", "C2",...
    "G2", "G3", "G2", "G3", "F2", "F3", "F2", "Eb3", "Eb2", "Eb3", "Eb2", "Eb3", "C2", "C3", "C2", "C3",...
    "G2", "G3", "G2", "G3", "F2", "F3", "F2", "Eb3", "Eb2", "Eb3", "Eb2", "Eb3", "C2", "C3", "C2", "C3",...
    "G2", "G3", "G2", "G3", "F2", "F3", "F2", "Eb3", "Eb2", "Eb3", "Eb2", "Eb3", "C2", "C3", "C2", "C3",...
    "G2", "G3", "G2", "G3", "F2", "F3", "F2", "Eb3", "Eb2", "Eb3", "Eb2", "Eb3", "C2", "C3", "C2", "C3", "G2"];
time2 = [4+16, 4, 3, 5, 4,...
    4, 3, 5, 4,...
    4, 3, 5, 4,...
    1,1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1,...
    1,1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1,...
    1,1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1,...
    1,1,1,1, 1,1,1, 1,1,1,1,1, 1,1,1,1, 8];
```

ex5.m

```
1
2  [x,FS]=audioread('u2.wav');
3
4  %sound(x,FS);
5
6
7  % Primeira opcao
8  figure(1);
9  spectrogram(x,hanning(512),256,1024,FS,'yaxis');
10
11 % Frequencias das notas
12 %x=highpass(x,800,FS);
13 %pitch(x,FS, WindowLength=1024, OverlapLength=128, Range=[800,2500])
14
15 % Obtencao do valor das notas (essencialmente o mesmo)
16 % for i = 1:512:length(x)
17 %     pwelch(x(i:(i+512)),rectwin(512),256,512,FS)
18 %     pause;
19 % end
20
21
22 do = nthroot(2,12)^(-9);
23 reb = nthroot(2,12)^(-8);
24 re = nthroot(2,12)^(-7);
25 mib = nthroot(2,12)^(-6);
26 mi = nthroot(2,12)^(-5);
27 fa = nthroot(2,12)^(-4);
28 solb = nthroot(2,12)^(-3);
29 sol = nthroot(2,12)^(-2);
30 lab = nthroot(2,12)^(-1);
31 la = 1;
32 sib = nthroot(2,12)^(1);
33 si = nthroot(2,12)^(2);
34
35
36 % Codigo sem sobreposicao das notas
37 %pauta1 = [NaN si la NaN mi solb mi solb si la NaN mi solb mi solb si la NaN mi solb mi
       aolb si la NaN];
38 %duracao1 = [0.3 0.9 0.8 1.8 0.6 0.25 0.25 0.25 0.9 0.8 1.8 0.6 0.25 0.25 0.25 0.9 0.8
       1.8 0.6 0.25 0.25 0.25 0.9 0.8 2];
39
40 % # FA
41 pauta1 = [NaN si NaN NaN mi NaN mi NaN  si NaN NaN mi NaN mi NaN  si NaN NaN mi NaN mi
       NaN  si NaN NaN];
42 pauta2 = [NaN NaN la NaN NaN solb NaN solb  NaN la NaN NaN solb NaN solb  NaN la NaN NaN
       solb NaN solb  NaN la NaN];
43 duracao1 = [0.3 1.7 0.8 1 0.8 0.05 0.40 0.10  1.7 0.8 1 0.8 0.05 0.40 0.10  1.7 0.8 1 0.8
        0.05 0.40 0.10  1.7 0.8 2];
44 duracao2 = [0.3 0.9 1.6 1 0.6 0.40 0.1 0.4  0.75 1.6 1 0.6 0.40 0.1 0.4  0.75 1.6 1 0.6
       0.40 0.1 0.4  0.75 1.6 2];
45
46 amostras1 = [];
47 amostras2 = [];
48 for i=1:length(pauta1)
49     amostras1 = [amostras1 geranota(4*pauta1(i), duracao1(i), FS)];
50     amostras2 = [amostras2 geranota(4*pauta2(i), duracao2(i), FS)];
51 end
52
53 amostras = amostras1 + amostras2;
54
55 x = amostras/5000;
56 x = x/8;
57
58 figure(2);
59 %sound(x,FS);
60 spectrogram(x,hanning(512),256,1024, FS,'yaxis');
61
62 audiowrite('batata.wav',x,FS);
63
64 function amostras = geranota(nota, duracao, Fs)
65     if isnan(nota)
66         t = 0:1/Fs:duracao;
67         %disp(length(t));
68         amostras = zeros(1,(length(t)-1));
69         return
70     end
```

```
    T0 = 1.0/(440.0*nota); % nota = 1 -> LA4 ou A

    K = 5;
    A = 5000;

    t = 0:1/Fs:duracao;
    t = t(1:(length(t)-1)); % Retirar a ultima amostra

    x=0;
    for k=1:K
        x = x - ((2*A)/(pi*k))*sin((2*pi*k*t)/(T0));
    end

    % Fade in e fade out
    T_sinusoide = 4*duracao/10; % Periodo da sinusoide de transicao

    t = 0:1/Fs:T_sinusoide/4;
    t = t(1:(length(t)-1));
    index = 1:length(t);

    sinusoidal = sin((2*pi/T_sinusoide)*t);
    x(index) = x(index).*sinusoidal;
    x(Fs*duracao-index+1) = x(Fs*duracao-index+1).*sinusoidal;

    %%%%%%%%%% Exponencial fade out %%%%%%%%%%
    t = 0:1/Fs:9*duracao/10;
    t = t(1:(length(t)-1));
    index = 1:length(t);

    exponential = exp(-3*t);
    exponential = exponential(length(t)-index+1);
    x(Fs*duracao-index+1) = x(Fs*duracao-index+1).*exponential;

    amostras = x;
end
```