Relatório 2 - Codificação de Informação Multimédia

CIM 2022/2023 **Class: 1MEEC_T02**
Pedro Silva | Tiago Ribeiro

1 Introdução

O presente relatório foi proposto no âmbito da unidade curricular de "Codificação de Informação Multimédia"e tem como objetivo a perceção de som com recurso à ferramenta MATLAB, bem como a codificação de música neste *software*. Neste são utilizados vários métodos empíricos para a codificação e ainda utilizados bibliotecas não pré-instaladas para a perceção de notas musicais no som fornecido.

2 Exercício 1

No primeiro exercício, codificou-se os 5 primeiros termos da seguinte equação:

$$x = -\sum_{1}^{\infty} \frac{2A}{\pi k} \times \sin \frac{2\pi}{T} kt \tag{1}$$

A equação foi programada com os seguintes valores para as varáveis:

$$\begin{cases} A = 5000 \\ F_s = 22\,050\,\text{Hz} \\ \dfrac{1}{T} = 440\,\text{Hz} \end{cases} \tag{2}$$

Esta representa uma sinal em dente de serra. Como é possível verificar, este sinal corresponde apenas a uma sobreposição ponderada de várias sinusoides.

A figura abaixo representa o primeiro período da onda gerada:

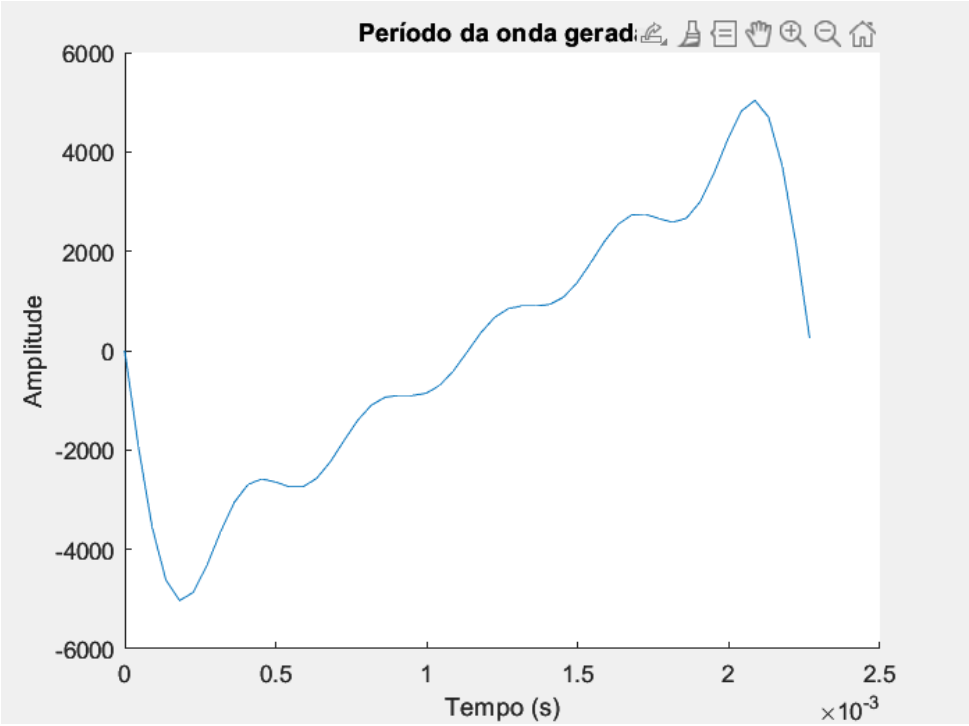

Figura 1: Período da Onda Gerada

É importante notar que a perceção do sinal em dente de serra é notável aumentando o número de períodos da onda visíveis no gráfico.

No secção dos Anexos é possível verificar o código Matlab produzido.

3 Exercícios 2 e 3

Nos exercícios 2 e 3, foi proposto a codificação de uma função, `geranota`, mediante uma série de parâmetros fornecidos e testada essa função com uma pauta pré-definida.

O efeito *fade-in* e *fade-out* pode ser confirmado através da visualização do gráfico seguinte:

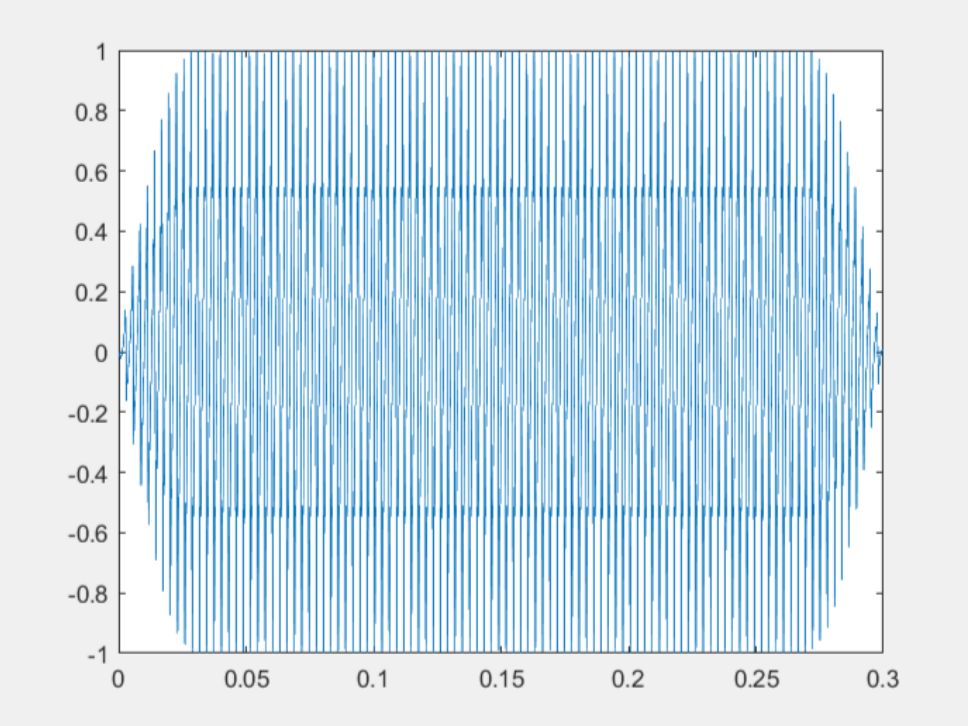

Figura 2: Amostra musical com efeito *fade-in* e *fade-out*

Deste modo, verifica-se, como esperado, o aumento gradual da amplitude das amostras desde o instante inicial, que se mantém enquanto as notas musicais vão sendo "tocadas". No final, verifica-se a diminuição gradual do valor da amplitude das amostras até 0.

Com este efeito, foi codificado adicionalmente uma função, `convernota`, que, recebendo uma nota como parâmetro, calcula a relação desta com a nota **Lá**, na **4ª Oitava**, recorrendo à razão $R = 2^{1/12}$, onde a nota cuja frequência se pretende equacionar, multiplica-se por um fator R^k, sendo k o número de semi-tons entre Lá e a nota cuja frequência se pretende descobrir.

Por exemplo, a nota "dó", estando a uma distância de 9 semi-tons da nota de referência Lá, a frequência da nota "dó"pode ser obtida multiplicando R^{-9} pela frequência de **Lá** $\longrightarrow 440\,\text{Hz}$.

Após saber-se a frequência da nota pretendida, utiliza-se a função `geranota` para calcular o valor das amostras, multiplicando o primeiro décimo desta pelo primeiro quarto de uma função seno para fazer o *fade-in* e o último décimo desta pelo segundo quarto da função seno para fazer o *fade-out*.

Assim, obtém-se a música pretendida, mais detalhes sobre a implementação podem ser verificados na secção Anexos, onde consta o código Matlab das funções referidas.

4 Exercício 4

Para a elaboração deste exercício, escolheu-se recriar a música Harold Faltermeyer - Axel F.

Analogamente ao exercício anterior, inicialmente recorreu-se a uma função auxiliar por nós definida, `convertenota_plus`. Esta função recebe como parâmetros a nota pretendida num formato específico, `input_nota`, e a velocidade da múscia em batidas por minuto, `bpm`.

Para o correto funcionamento desta função, a `nota_input` deve seguir o seguinte modelo "Zn-k", onde Z refere-se à nota (do, re, ...), n refere-se à oitava da nota e k à duração da nota, em que k diz respeito à k parte de uma semibreve (i.e. Colcheia $\longrightarrow$ k = 8 - representa a oitava parte duma semibreve).

Deste modo, é devolvido nos parâmetros `nota_R` e `duracao` o fator a multiplicar pela nota Lá4 e a duração em segundos, respetivamente.

Relativamente ao cálculo de `nota_R`, é importante notar que se multiplica `12 * escala` para obter a distância, em semi-tons, entre a nota Lá4 e a pretendida, podendo esta estar numa outra oitava distinta.

Após isto, recorreu-se novamente à função `geranota`.

Deste modo, ficam concluídos todos os passos necessários para a tocar a música, bastando providenciar as notas num vetor segundo o modelo definido.

O código programado pode ser visto na respetiva secção nos Anexos

5 Exercício 5

Inicialmente, para detetar as notas musicais da melodia da música fornecida "u2.wav", começou-se por fazer um espectograma do sinal, analisando as frequências que constituem a melodia. Através de uma breve análise, é possível depreender que o valor da primeira nota musical está próximo dos 2 kHz, que, pesquisando pelos valores tabelados das notas musicais, e comparando com o som de um piano real, conclui-se que esta nota musical poderá ser um Si5, cuja frequência é 1975.5 Hz.

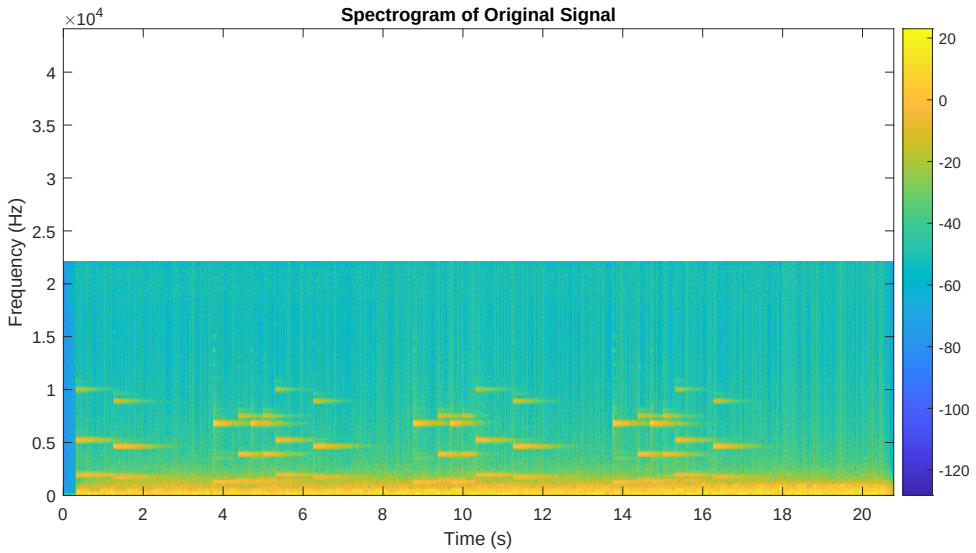

Figura 3: Espectograma da música original

Tendo descoberto esta primeira nota, usou-se de um piano real para recriar a melodia por ouvido, tocando iterativamente as notas no piano, até chegar à conclusão de que a melodia é constituída pelas notas Si6 - Lá6 - Mi6 - Fá♯6

Feito isto, obteve-se o seguinte sinal:

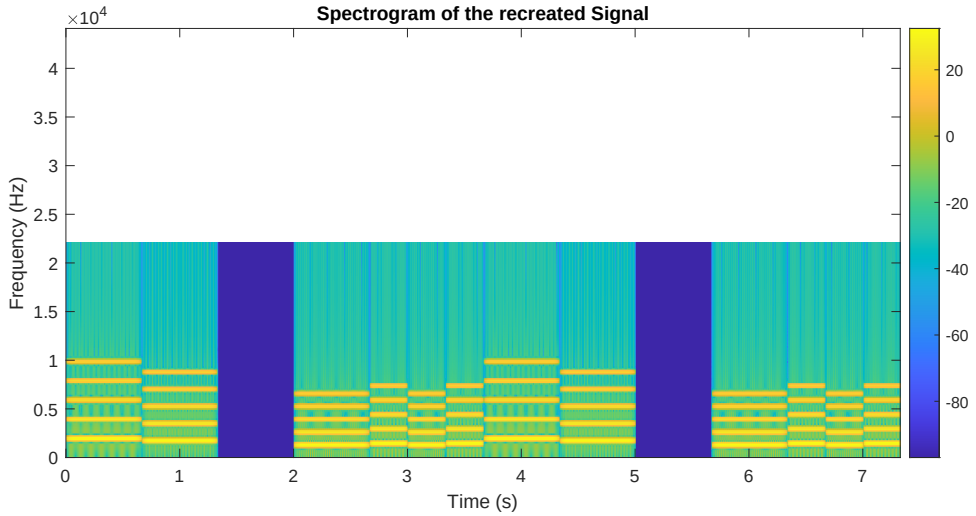

Figura 4: Espectograma do som recriado

Esteticamente, o som original e o som recriado em Matlab são extremamente diferentes apesar de corresponderem às mesmas notas. Esta diferença pode ser justificada fazendo uma comparação dos dois espectogramas, em que de facto a composição harmónica dos sons é também bastante diferente.

De um lado, tem-se o som original, um som orgânico, com pequenas variações no timbre, velocidade e frequência, que tornam o som mais musical e agradável ao ouvido humano, para além de que são tocadas outras notas musicais em simultâneo.

Do outro lado, tem-se o som recriado que, apesar de respeitar a melodia original, é demasiado robótico, demasiado preciso, o que desvirtua a musicalidade do som.

6 Conclusão

O objetivo principal do trabalho prendia-se por conseguir sintetizar e visualizar um sinal em dente de serra, bem como criar uma função capaz de gerar a frequência e amplitude das notas musicais, quer inicialmente de forma pré-definida, quer a música por nós criada. Adicionalmente, tentou-se ainda retirar as notas musicais de uma música fornecida.

É possível afirmar que todas as tarefas foram completadas com sucesso, podendo-se recriar e ouvir a música pré-definida e também a música por nós criada, verificando-se sempre os fenómenos de *fade-in* e *fade-out*.

Anexos

Exercício 1

```
% Constantes
A = 5000;
T0 = 1/440;
f0 = 1/T0;
Fs = 22050;
t = 0:1/Fs:T0;

% Sintetizacao dos primeiros 5 termos
x = zeros(size(t));
for k = 1:5
    x = x - (2*A/(pi*k))*sin((2*pi/T0)*k*t);
end

x = (x - min(x)) * 2 / (max(x) - min(x)) - 1;
sound(x,Fs);

% Plot
figure();
hold on
plot(t, x)
xlabel('Tempo (s)')
ylabel('Amplitude')
title('Periodo da onda gerada')
hold off
```

Função geranota

```
% nota
function amostras = geranota(nota, duracao, Fs)
    % Sintetizacao dos primeiros 5 termos
    T0 = 1.0/(440.0*nota);
    t = 0:1/Fs:duracao;
    A = 5000;

    amostras = zeros(size(t));
    if nota == 0
        return;
    end

    for k = 1:5
        amostras = amostras - (2*A/(pi*k))*sin((2*pi/T0)*k*t);
    end

    % Calcula o tamanho do fade in e fade out
    fade_len = floor(length(amostras) / 10);

    % Generate sine wave
    sine_period = 4*fade_len;
    sineWave = sin(linspace(0, 2*pi, sine_period))';

    % Apply sine wave to audio signal
    amostras(1:fade_len) = amostras(1:fade_len) .* sineWave(1:fade_len)';
    amostras(end-fade_len:end) = amostras(end-fade_len:end) .* sineWave(
        fade_len:2*fade_len)';

    amostras = amostras/max(abs(amostras));
    % plot(t, amostras);
end
```

Exercícios 2 e 3

```
Fs = 22050;

pauta = ["do" "re" 'mi' 'fa' 'fa' 'fa' 'do' 're' 'do' 're' 're' 're' 'do' '
    sol' 'fa' 'mi' 'mi' 'mi' 'do' 're' 'mi' "fa"];
melody = [];

for i = 1:length(pauta)
    nota = convertenota(pauta(i));
    waveform = geranota(nota, 0.3, Fs);
    melody = [melody waveform];
end

sound(melody, Fs);
audiowrite('audio\melody.wav', melody, Fs);

function nota_R = convertenota(nota)
    R = 2^(1/12);

    switch nota
        case 'do'
            nota_R = R^(-9);
        case 'do#'
            nota_R = R^(-8);
        case 're'
            nota_R = R^(-7);
        case 're#'
            nota_R = R^(-6);
        case 'mi'
            nota_R = R^(-5);
        case 'fa'
            nota_R = R^(-4);
        case 'fa#'
            nota_R = R^(-3);
        case 'sol'
            nota_R = R^(-2);
        case 'sol#'
            nota_R = R^(-1);
        case 'la'
            nota_R = R^(0);
        case 'la#'
            nota_R = R^(1);
        case 'si'
            nota_R = R^(2);
        otherwise
            nota_R = 0;
    end

end
```

Função convertenota_plus

```
function [nota_R, duracao] = convertenota_plus(input_nota, bpm)
    R = 2^(1/12);

    out = regexp(input_nota, '([a-z#]+)(\d+)-(\d+)', 'tokens');
    nota = out{1}{1};
    escala = str2double(out{1}{2})-4;
    intervalo = str2double(string(out{1}{3}));
    duracao = (60/bpm) * (4/intervalo);

    switch nota
        case 'do'
            nota_R = R^(-9 + 12*escala);
        case 'do#'
            nota_R = R^(-8 + 12*escala);
        case 're'
            nota_R = R^(-7 + 12*escala);
        case 're#'
            nota_R = R^(-6 + 12*escala);
        case 'mi'
            nota_R = R^(-5 + 12*escala);
        case 'fa'
            nota_R = R^(-4 + 12*escala);
        case 'fa#'
            nota_R = R^(-3 + 12*escala);
        case 'sol'
            nota_R = R^(-2 + 12*escala);
        case 'sol#'
            nota_R = R^(-1 + 12*escala);
        case 'la'
            nota_R = R^(0 + 12*escala);
        case 'la#'
            nota_R = R^(1 + 12*escala);
        case 'si'
            nota_R = R^(2 + 12*escala);
        case 'x'
            nota_R = 0;
        otherwise
            nota_R = 0;
    end
end
```

Exercício 4

```
% Harold Faltermeyer - Axel F
Fs = 22050;
bpm = 120;

loop1 = ["fa4-8" "x0-8" "sol#4-8" "x0-16" "fa4-8" "fa4-16" "la#4-8" "fa4-8" "re#4-8" ...
        "fa4-8" "x0-8" "do5-8" "x0-16" "fa4-8" "fa4-16" "do#5-8" "do5-8" "sol#4-8" ...
        "fa4-8" "do5-8" "fa5-8" "fa4-16" "re#4-8" "re#4-16" "do4-8" "sol4-8" "fa4-2"...
        "x0-8" "x0-2"];

loop2 = ["do5-8" "do5-8" "x0-16" "re#5-8" "re#5-8" "re#5-16" "do5-8" "do5-8" ...
        "x0-8" "do5-8" "do5-8" "x0-8" "re#5-8" "re#5-16" "re5-8" "do5-8" "x0-4" ...
        "sol#4-8" "sol#4-8" "sol#4-8" "sol#4-16" "la#4-8" "la#4-8" "la#4-8" ...
        "x0-16" "la#4-8" "do5-8" "do5-8" "do5-8" "la#4-16" "do5-8" "do5-8" "x0-4"];

musica = [loop1 loop2 loop1];
melody = [];
for i = 1:length(musica)
    [nota, duracao] = convertenota_plus(musica(i), bpm);
    waveform = geranota(nota, duracao, Fs);
    melody = [melody waveform];
end

sound(melody, Fs)
audiowrite('audio/AxelF.wav', melody, Fs);
```

Exericício 5

```
[x, Fs]=audioread('audio\u2.wav');
% sound(x, Fs);

% Spectogram parameters
N = 256;
Window = hamming(N);
Noverlap = N/2;
freq_range = [0, Fs];

[S, f, t] = spectrogram(x, Window, Noverlap, N, Fs);

% Plot spectrogram of original signal
figure;
imagesc(t, f, 20*log10(abs(S)));
axis xy;
xlabel('Time (s)');
ylabel('Frequency (Hz)');
xlim([0, max(t)]);
ylim(freq_range);
title('Spectrogram of Original Signal');
colorbar;

pauta = ["si5-4" "la5-4" "x0-4" "mi6-4" "fa#6-8" "mi6-8" "fa#6-8"];

musica = [pauta pauta];
bpm = 90;   % Estimativa apenas

melody = [];
for i = 1:length(musica)
    [nota, duracao] = convertenota_plus(musica(i), bpm);
    waveform = geranota(nota, duracao, Fs);
    melody = [melody waveform];
end

sound(melody, Fs)
audiowrite('audio/u2_remake.wav', melody, Fs);

[S, f, t] = spectrogram(melody, Window, Noverlap, N, Fs);

% Plot spectrogram of recreated signal
figure;
imagesc(t, f, 20*log10(abs(S)));
axis xy;
xlabel('Time (s)');
ylabel('Frequency (Hz)');
xlim([0, max(t)]);
ylim(freq_range);
title('Spectrogram of the recreated Signal');
colorbar;
```