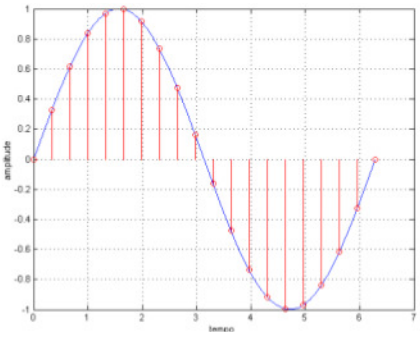
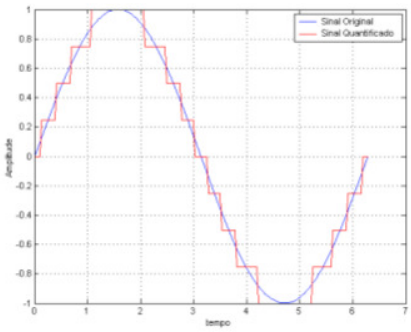
Trabalho introdutório - Breves experiências com sinais áudio no MATLAB

Objectivos: Este trabalho tem por objectivo introduzir os estudantes à manipulação de sinais de som no formato digital em MATLAB.

Introdução

Para manipular sons no computador torna-se necessário decidir o nº de amostras a recolher por unidade de tempo (frequência de amostragem) e o nº de valores com que vão ser representadas essas amostras (nº de níveis de quantização, ditado pelo nº de bits que se utiliza).

amostragem	quantização
	
$f_{amostragem}=20*f_{sinal}$	L=9 níveis de quantização -> $b=\log_2(L)=4$

Um sinal sonoro é representado como uma sequência de números/valores que representam a sua amplitude em determinados instantes, igualmente espaçados; isto é, esses números representam o valor das amostras que são recolhidas durante o processo de amostragem. Normalmente, antes da digitalização (amostragem), o sinal de som é filtrado para remover as componentes de frequência não desejadas, isto é, frequências cujo valor seja superior a metade do valor da frequência de amostragem que desejamos utilizar. As frequências que vamos manter, vão depender da aplicação e do próprio sinal:

a) para aplicações de voz, retêm-se tipicamente as componentes de frequência situadas entre 50Hz e 10kHz.

b) para sinais de música, eliminam-se as frequências apenas fora da faixa 20Hz-22kHz.

Após tomar a decisão sobre as frequências a reter, podemos decidir os níveis de quantização, ou número de bits, a utilizar. O nº de bits vai ter influência na qualidade e no débito gerado (assim como a frequência de amostragem).

Valores que são tipicamente utilizados em diferentes aplicações são apresentados na tabela seguinte:

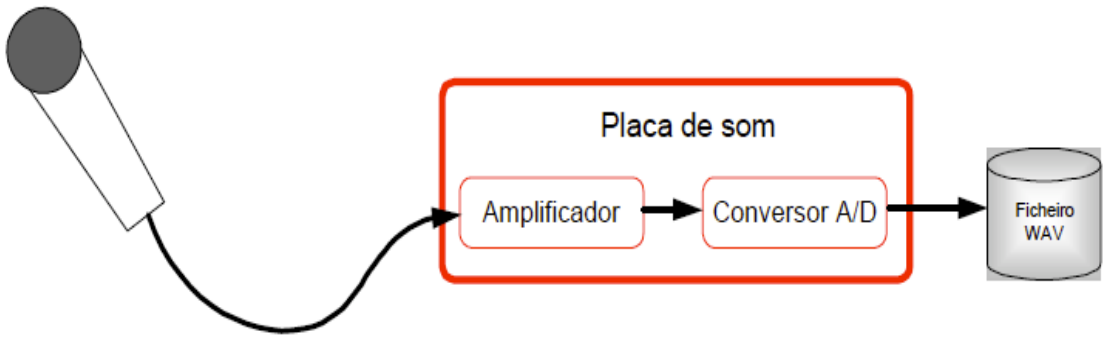
Quality	Sample Rate (Khz)	Bits per Sample	Mono / Stereo	Data Rate (uncompressed) (kB/sec)	Frequency Band (KHz)
Telephone	8	8	Mono	8	0.200-3.4
AM Radio	11.025	8	Mono	11.0	0.1-5.5
FM Radio	22.05	16	Stereo	88.2	0.02-11
CD	44.1	16	Stereo	176.4	0.005-20
DAT	48	16	Stereo	192.0	0.005-20
DVD Audio	192 (max)	24(max)	6 channels	1,200 (max)	0-96 (max)

Trabalho a desenvolver

NOTA: a versão Matlab a utilizar deverá ser a R2019a disponível no apps.fe.up.pt

Este trabalho pretende introduzir os estudantes à manipulação de sinais de som no formato digital. Os estudantes terão oportunidade de observar por experiência própria os efeitos que as operações fundamentais da digitalização de sinais media - a amostragem e a quantização - operam nesses mesmos sinais.

Para adquirir sons no computador este tem de estar equipado com uma placa de som. Com um microfone ligado, no Windows podemos adquirir som usando o programa "Gravador de sons" disponível no menu "Iniciar -> Acessórios -> Entretenimento -> Gravador de áudio". O som que é capturado é digitalizado automáticamente e armazenado em disco. Podemos indicar o formato que desejamos, incluíndo a frequência de amostragem e o número de bits por amostra a utilizar. Se usarmos o formato PCM, não estamos a introduzir qualquer tipo de compressão. Nota: a localização da aplicação pode diferir consoante a versão do SO.



1. Analisar sinais áudio em Matlab

Podemos também utilizar para este trabalho ficheiros de som digitais disponíveis na página da disciplina (ficheiros de som com o formato Wave que têm a extensão .wav).

Mude o directório de trabalho Matlab para a pasta onde gravou os ficheiros áudio. Para verificar o directório actual use a seguinte instrução:

```
>> pwd
ans =
    'C:\Users\mmfc\Documents\MATLAB'
```

Se pretender mudar para outro directório (p.ex.: Lab0):

```
>> cd 'C:\Users\mmfc\Documents\MATLAB\Lab0'
```

Ou apenas:

```
>> cd Lab0
```

Para listar os ficheiros de uma pasta utilize o comando "`ls`":

```
>> ls
.           ..          teste.wav
```

No Matlab podemos importar um ficheiro `.wav` usando a seguinte instrução:

```
>> som=audioread('teste.wav');
>> [som,fs]=audioread('teste.wav');
```

O som importado fica armazenado na variável 'som'. No segundo caso obtemos a frequência de amostragem na variável fs (a que foi utilizada quando o som foi digitalizado). De salientar que esta função do Matlab (audioread) é capaz de ler diversos formatos entre os quais MPEG-1 Layer 3 (mp3) e MPEG-4 AAC. Exemplo do manual da instrução "`audioread`":

Podemos obter informação sobre o som importado com a função "audioinfo":

```
>> info=audioinfo('teste.wav')
info =
  struct with fields:
             Filename: 'C:\Users\mmfc\Documents\MATLAB\Lab0\teste.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 2
           SampleRate: 44100
         TotalSamples: 447692
             Duration: 10.1517
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 16
```

A função "audioinfo" cria uma estrutura com vários campos onde coloca informação extraída do ficheiro de som. Podemos usar esses campos como variáveis. Por exemplo, o número de bits usado para representar cada amostra:

```
>> nbits=info.BitsPerSample

nbits =
    16
```

Para ouvir um som no Matlab que esteja armazenado na variável "som" utilizando uma frequência de amostragem "fs", podemos utilizar a função "sound" da forma seguinte:

```
>> sound(som,fs);
```

Este comando reproduz directamente para a saída da placa de som o som digital armazenado no vector "som" utilizando uma frequência de amostragem "fs", isto é fs amostras por segundo.

Note-se que o a amplitude do vector "som" deve estar compreendida no intervalo [-1...1]. Caso tal não aconteça, existem duas soluções possíveis:

1) dividir o vector "som" pelo valor da amostra de maior amplitude do sinal, isto é executar a instrução:

```
>>som=som/max(abs(som));
```

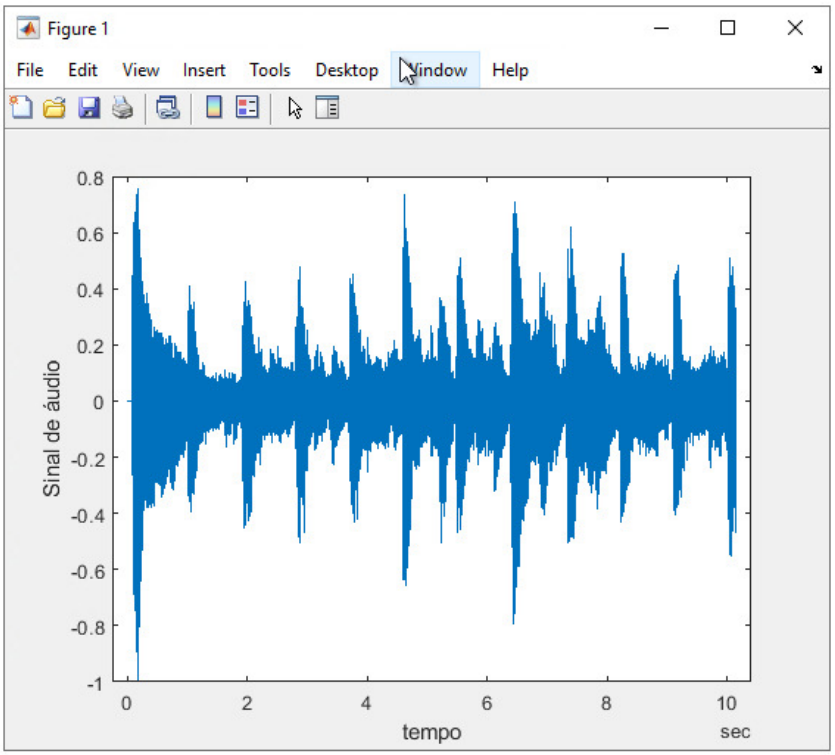
2) Utilizar a função "soundsc(x,fs)" que faz a operação da alínea anterior automaticamente:

```
>> soundsc(som,fs);
```

Podemos também observar a forma do sinal ao longo do tempo. Para isso devemos criar um vector `t` que tenha o mesmo tamanho de som importado "`som`":

```
>> t=0:seconds(1/fs):seconds(info.Duration);
>> t=t(1:end-1);
>> plot(t,som)
>> xlabel('tempo')
>> ylabel('Sinal de áudio')
```

A figura seguinte ilustra a forma do sinal ao longo do tempo. Sugestão: repetir os passos anteriores para outros sons exemplo.



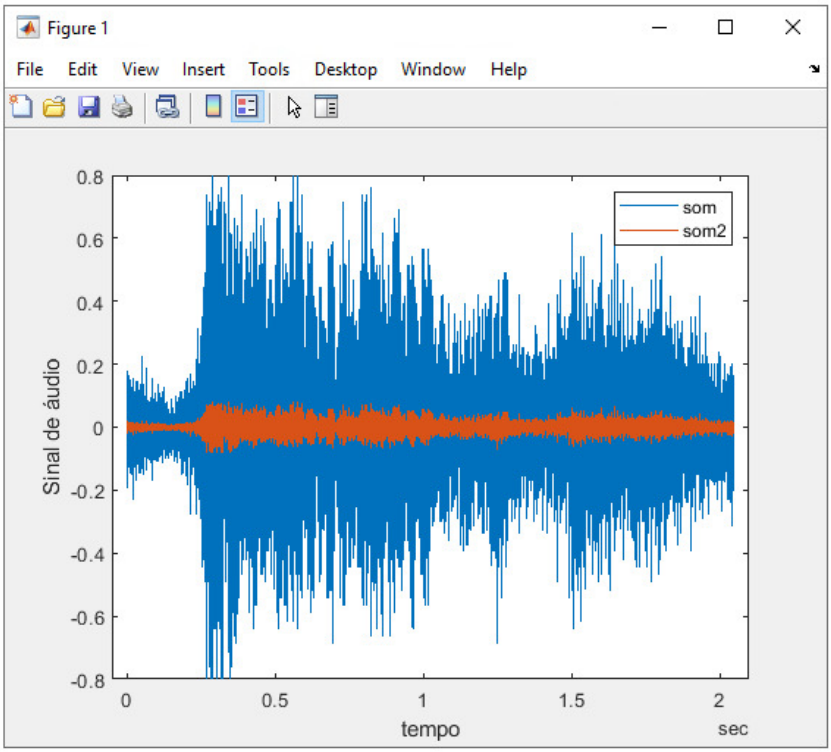
2. Amplitude sinais áudio

Vamos ler do disco para o Matlab uma música armazenada num ficheiro "wav". Vamos dividir a amplitude do sinal por 10, ouvir o resultado e gravá-lo no disco.

```
>>[som,fs] = audioread('handel100.wav'); %Ler o ficheiro do disco
>>info = audioinfo('handel100.wav')      %Ler informação áudio
>>nb = info.BitsPerSample; % obtém número de bits por amostra
>>fs                       % verifica o valor da frequência de amostragem
>>nb                       % verifica o número de bits (quantização)
>>sound(som,fs)            % ouve o som adquirido original
>>som2= som/10;            % diminui a amplitude do som
>>sound(som2,fs)           % ouve o resultado
>>audiowrite('handel100baixo.wav',som2,fs);     % grava o resultado
```

Observe a amplitude de ambos os sinais ao longo do tempo:

```
>> t=0:seconds(1/fs):seconds(info.Duration);
>> t=t(1:end-1);
>> plot(t,som)
>> xlabel('tempo')
>> ylabel('Sinal de áudio')
>> hold on
>> plot(t,som2)
>> legend('som','som2')
```



Agora faça o mesmo mas para ampliar o som.

3. Taxas de amostragem em sinais áudio

Vamos agora fazer várias experiências de utilização de diferentes taxas de amostragem, utilizando valores inferiores ao valor originalmente utilizado ao importar o ficheiro para o Matlab, nomeadamente usando fs/2.

```
>>[som,fs] = audioread('handel100.wav');
>>soundsc(som,fs/2);
```

e usando fs*2:

```
>>soundsc(som,fs*2);
```

4. Música com base em sons puros

Crie agora música com base em sons puros. Sons puros são sinais sinusoidais (de uma só frequência). Podemos criar sons puros usando a função "`sin`", tal como indicado de seguida.

Por exemplo para criar uma onda sinusoidal de amplitude 1 e frequência 523.25Hz (corresponde a um pitch C num piano, uma oitava acima do C médio):

```
>>c=sin(2*pi*523.25*(0:0.000125:0.5));
```

O vector "`c`" vai conter amostras de uma onda sinusoidal com a frequência de 523.25Hz; essas amostras são recolhidas desde o instante t=0s até ao instante t=0.5s, separadas entre si de 0.000125s (intervalo de amostragem Ts=1/fs), correspondente a fs=8kHz.

Grave esse som num ficheiro .wav:

```
>>fs = 8000;
>>audiowrite('c.wav',c, 8000);
```

Reproduza esse som:

```
>>[pitchC,fs]=audioread('c.wav');
>>sound(pitchC,fs);
```

Para obter a lista das frequências de diferentes notas: http://www.dolmetsch.com/musictheory27.htm

Crie mais notas, tais como:

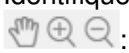
```
>>f = sin(2*pi*174.61*(0:0.000125:0.5));
>>g = sin(2*pi*195.99*(0:0.000125:0.5));
>>a = sin(2*pi*220*(0:0.000125:0.5));
>>b = sin(2*pi*246.94*(0:0.000125:0.5));
```

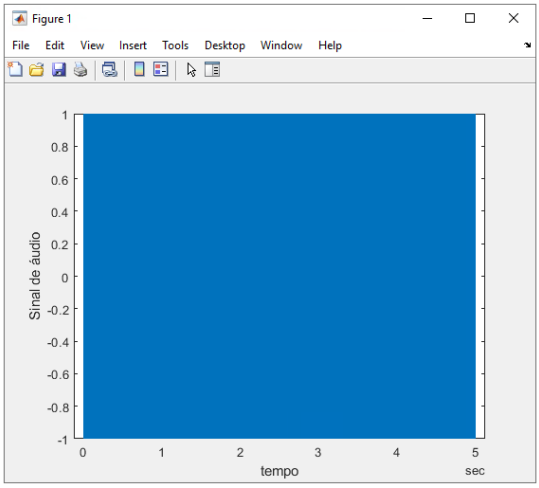
Crie sinais áudio através de composição com recurso às notas anteriores

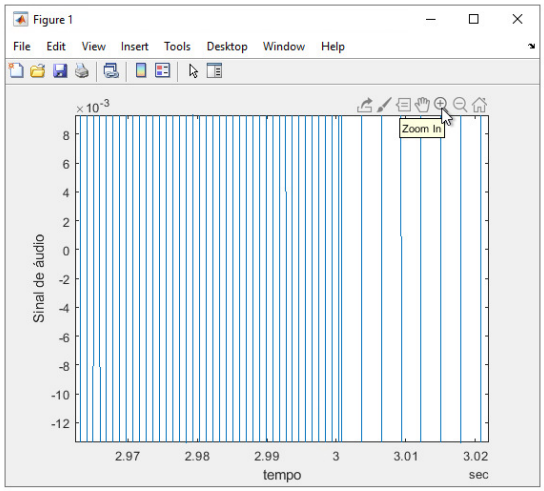
```
>>line1=[a,b,c,f,g];
>>line2=[c,f,g,b,a];
>>song=[line1,line2];
>>sound(song);
>>soundsc(song,fs);
```

Visualize e analise o sinal resultante através da função "`plot`".

```
>> t=0:seconds(1/fs):seconds(length(song)/fs);
>> t=t(1:end-1);
>> plot(t, song)
>> xlabel('tempo')
>> ylabel('Sinal de áudio')
```

Identifique visualmente uma transição de nota através das ferramentas de navegação e visualização  :





Alguns links relacionados com Matlab:

- MATLAB® Resources at MIT:
  - https://ocw.mit.edu/resources/res-18-002-introduction-to-matlab-spring-2008/other-matlab-resources-at-mit/
- MATLAB – Overview (tutorialspoint.com):
  - https://www.tutorialspoint.com/matlab/matlab_overview.htm
- Mathworks Matlab Examples:
  - https://www.mathworks.com/examples/
- Mathworks Audio Toolbox — Examples
  - https://www.mathworks.com/help/audio/examples.html?category=index&s_tid=CRUX_lftnav_example_audio-processing-algorithm-design

Podem consultar também alguns tutoriais disponíveis na página Moodle da UC.