Processamento Digital de Sinais
Instituto Superior Técnico
Author: João Sanches.
Revised: Margarida Silveira, Jorge Marques, Luís B. Almeida, Pedro Aguiar.

Sampling and Aliasing

This lab assignment addresses *sampling* and *aliasing* with synthetic and real-life signals. Digital computers are powerful tools which allow us to manipulate and store signals. However, digital computers can only manipulate discrete-time signals. This means that continuous-time signals (e.g., speech, audio, biomedical signals) have to be converted into discrete-time signals, by a sampling operation, before being processed in a computer. This conversion is lossless if the sampling rate is high enough: the sampling theorem states that no information is lost if the sampling frequency is higher than twice the highest frequency of the signal. If this condition is not met, there is loss of information, caused by *aliasing* (also called *spectral folding*). This creates an annoying effect, since high-frequency components of the signal are moved into the low frequencies, and distort the signal. This work studies these effects using synthetic and real-life audio data. The work also illustrates the usefulness of *anti-aliasing* filters.

Notes

This assignment is to be performed in the lab class. At the end of the class you must submit, through fenix, a file with your answers and MatLab code. The items that you should address are marked, below, in a format such as **R1.a)**.

Experimental work

1. Consider a continuous-time chirp[1] (with t measured in seconds),

$$x(t) = \cos\left[2\pi\left(\frac{1}{2}kt^2 + f_0 t + \phi_0\right)\right] \tag{1}$$

with instantaneous frequency $\omega(t) = 2\pi(kt + f_0)$. In this work, we will use $k = 1000$, $f_0 = 0$, and $\phi_0 = 0$.

Build a MATLAB vector `x` by sampling the chirp $x(t)$ at a rate of 8000 samples per second in the interval $[0, 4]$ seconds. Do your own implementation of the sampled signal, do not use any Matlab ready made function for chirp creation. Listen to the obtained signal by using the following command:

```
soundsc(x, 8000)
```

R1.a) Comment on the relationship between what you heard and the signal that you created in `x`.

2. Compute and display the spectrogram of `x` using the command

`N=32; spectrogram(x, hann(N), 3*N/4, 4*N, 8000, 'yaxis')`.

The first argument of the `spectrogram` function, `x`, is the signal of which you wish to compute the spectrogram. The second argument, `hann(N)`, indicates the window that should be used to compute the spectrogram, in this case the Hanning window. By varying the value of `N`, you can control the length of the window (`N` should be a multiple of 4). The fifth parameter, `8000`, indicates the sampling frequency of the signal. For more information about the `spectrogram` function, use MATLAB's help.

R2.a) Choose the window length that yields, in your opinion, the best representation of the signal's time-frequency structure. Justify your choice.

R2.b) What is the relationship between the spectrogram and the sound you heard?

3. Sample the signal $x(n)$ by obtaining the following signal: $y(n) = x(2n)$. Listen to $y(n)$ and observe its spectrogram. Note that you should use the correct value of the sampling frequency of $y(n)$ in the `soundsc`

[1] A chirp is a signal whose frequency varies monotonically (increasing or decreasing) with time.

command, since that is the sampling frequency of the signal that you are listening to. In the `spectrogram` command, you should adjust the window length to the sampling rate of the new signal, so that the window duration, measured in seconds, remains the same, and you should use the correct value of the sampling frequency.

R3.a) Indicate the sampling frequency of the signal $y(n)$. Justify your answer.

R3.b) Explain what you have heard and observed.

4. Load the sound file `romanza_pe.wav` using the command

```
[x, Fs] = audioread('romanza_pe.wav');
```

Write down the signal's sampling frequency, contained in the variable `Fs`. Listen to the signal's contents (don't forget to use the appropriate sampling frequency in the `soundsc` command).

Note: The signal is rather long. You can stop playing it with the command `clear sound`.

Visualize the spectrogram of the first ten seconds of the signal. In the computation of the spectrogram, choose a window length that allows you to clearly see the spectral lines of the violin's sound.

R4.a) Indicate which window duration you used for the computation of the spectrogram.

R4.b) Try to explain why the spectral lines are wavy.

5. Sample `x` so that the new sampling rate is `Fs/5`, and listen to the result (again, don't forget to use the appropriate sampling frequency in the `soundsc` command).

Visualize the spectrogram of the first ten seconds of the sampled signal (don't forget to properly set the window length and the sampling frequency).

R5.a) Describe, and try to explain, what you heard and observed.

6. Filter the original signal `x` by means of the following command:

```
xf = filter(fir1(100, 0.2), 1, x);
```

This command performs filtering by a order-100 low-pass FIR filter with a cut-off frequency of $0.2\,\pi$ (expressed in the angular-frequency scale of the discrete-time Fourier transform).

Sample the filtered signal `xf` so that the new sampling rate is `Fs/5`. Listen to the result and visualize its spectrogram, using the appropriate sampling frequency.

R6.a) Explain the differences in what you heard and observed, relative to what you heard and observed in item 5.