**Digital Signal Processing**
Instituto Superior Técnico
Lab assignment 3 – Analysys of nonstationary signals
Authors: João Sanches and Luís Caldas de Oliveira. Modified by Pedro Q. Aguiar, Luís B. Almeida and Margarida Silveira.

## Note

The report of this lab assignment should not exceed three A4 pages. In addition to the report, m-files with MatLab code for functions `dtmfdecode` and `dtmfdecode2` should be submitted.

## Introduction

Real-life signals often are nonstationary, i.e., they have characteristics that vary over time. A simple example is speech, whose characteristics change as different sounds are pronounced. To analyze nonstationary signals in frequency, the ordinary Fourier transform is often inadequate, because it does not clearly show the variation of the signals' spectral characteristics with time. A tool that is often used to analyze those signals, is the *Short-Time Fourier Transform* (STFT),[1] defined, for discrete-time signals, as

$$X(n,\omega) = \sum_{m=-\infty}^{+\infty} w(n-m)\, x(m)\, e^{-j\omega n}, \tag{1}$$

where $w(n)$ is a window function, a common example of which is the rectangular window

$$w(n) = \begin{cases} 1, & -N \le n \le N \\ 0, & \text{otherwise.} \end{cases}$$

For each value of $n$, Eq. (1) computes the Fourier transform of the signal $w(n-m)\,x(m)$. Assuming, for simplicity, that the window is rectangular, this signal is equal to $x(m)$ for $m \in \{n-N, \cdots, n+N\}$, and is zero outside that region. Therefore, for each value of $n$, $X(n,\omega)$ gives information on the spectral content of $x$ in a region around $n$, allowing us to analyze the variation of the spectral characteristics of $x$ along time. A tool that is closely related to the STFT, is the *spectrogram*, defined as

$$\bar{X}(n,\omega) = |X(n,\omega)|^2.$$

The type of analysis performed by the STFT and the spectrogram is called *time-frequency analysis*.

In this work, we'll analyze signals produced by the commonly used *Dual-Tone Multi-Frequency* (DTMF) telephone signaling system, also called *touch-tone* system. This system is used to transmit the user's key presses within the telephone system's audio band. Each telephone key is represented by a sum of two sinusoids (often called *tones*) with different frequencies. The correspondence between keys and sinusoid frequencies is given in the following table:

| Frequencies (Hz) | 1209 | 1336 | 1477 |
|---|---|---|---|
| 697 | 1 | 2 | 3 |
| 770 | 4 | 5 | 6 |
| 852 | 7 | 8 | 9 |
| 941 | $*$ | 0 | $\sharp$ |

The specifications of the DTMF signals that we will use, are the following:[2]

- The minimum duration of each tone segment is 40 ms.
- The minimum duration of the pauses between tone segments is 40 ms.
- In the tone segments, the two sinusoids have the same amplitude, which can have any value between 0.2 and 2.

[1] *Transformada Localizada de Fourier*, in Portuguese.
[2] The first three specifications correspond to a simplified version of the DTMF standard. 8 kHz is the sampling frequency used in most telephone networks, both land-line and mobile.

- The sampling frequency is 8 kHz.

**Experimental work**

You have available a function for generating DTMF signals, called `dtmfencode`. To obtain information on it, use the command `help dtmfencode`.

Write a Matlab function `dtmfdecode` that receives as argument a DTMF signal, and that outputs, in a string, the corresponding sequence of keys.

Suggestions:

- You may want to use the `spectrogram` function. Note that, despite its name, that function computes the STFT, and not the spectrogram. Be sure to use Matlab's help to understand what the function does and what is the meaning of its parameters.
- To visualize a function of two variables (e.g., a function of time and frequency), the `surf` command may be useful. After obtaining a plot of the function, the commands `shading` and `colormap` may be useful. It may also be useful to rotate the plot by means of the tool that is represented, in the Matlab interface, by a curved arrow around a tiny cube.

**R-a)** Give, in a separate m-file, the source-code listing of the function that you implemented.

**R-b)** Explain how your program works. Also explain how you chose the values of the parameters that are used in the program (including the parameters of the `spectrogram` function, if you used that function).

**R-c)** Generate several DTMF signals with different values of the parameters `toneDuration`, `pauseDuration`, and `amplitude`, in the `dtmfencode` function, and check whether your program can detect all noiseless signals that conform to the specifications given above. Also test signals with tone and/or pause durations lower than 40 ms, to check how far beyond the specifications your program can go. Comment on what you have observed.

**R-d)** Find the key sequence that is encoded in the file `touchtone.wav`.

**R-e)** Generate DTMF signals with various levels of noise, by varying the `noiseLevel` parameter of the `dtmfencode` function. Indicate, in your report, what is the approximate maximum value of the `noiseLevel` parameter that your program tolerates without making a significant number of errors. Be sure to try both short and long tone durations (in noisy signals with long tone durations, a single key press may be decoded as several consecutive key presses).

Note that, since noise is a random signal, you will need to perform several tests for each situation, in order to get a good idea of how your decoder behaves.

**R-f)** Try to improve your decoder in what concerns the resistance to noise. The Matlab function with your improved decoder should be named `dtmfdecode2`. Comment on what you have observed and on what you have changed in your decoder. Don't forget to indicate the approximate maximum value of `noiseLevel` that your improved detector tolerates without making a significant number of errors.