3 Use of DFT and Windowing II, FIR Filter Design

3.1 Spectrogram

3.1.1 A Signal's Spectrogram

Signals which carry information are usually non-stationary: they have a time-varying character. A spectrogram is a 2-dimensional representation of the evolution of the spectrum over time. The signal is windowed at regular intervals and its characteristics (here, the spectrum) are considered as the current characteristics of the original signal. Successive time-domain windows are usually chosen with 50 % overlap (or more). In the spectrogram, one axis (typically the horizontal X-axis) represents the time and the other (Y) axis represents frequency. Along a third (Z) axis, the value of the magnitude spectrum at frequency f for the signal window at time t is shown, creating a three-dimensional plot of the signal's spectrogram. However, because threedimensional plots are more difficult to interpret, a two-dimensional representation of the spectrogram is more often used. In this two-dimensional representation, the third (Z) axis is shown as a color- coded bitmap, in which each pixel's color represents the value of the magnitude spectum for the corresponding time t and frequency f. The correspondence between the colors and magnitude values is defined by a so-called **color map** (see the matlab help for the function '*colormap*'), which is visualised through the **color bar**, which is typically plotted next to the two-dimensional spectrogram (see function '*colorbar*').

3.1.2 Exercise: The Spectrogram of a Singing Voice

Load the signal “karaoke.wav” and listen to it with the function ’soundsc’. Use ’size’ to verify the number of channels of the signal and make a mono version if necessary (by making a sum of the different channels). Look at the spectrogram of the signal with the command ’spectrogram’. If we would like to measure the pitch (and thus the sung notes) of the signal, we need to measure the fundamental frequency and the first harmonics very accurately. Similar to the spectrum, a short analysis window will however cause low spectral resolution, while very long windows wil smear out the evolution of the signal parameters over time (low time resolution). A good choice of the analysis window’s size is thus important.
How do you interpret what you see on the spectrogram, knowing that the vocal cords will vibrate in an almost periodical way (according to the frequency of the corresponding note)? Can you see vibratos? Can you easily recognise where the different words and speech sounds start and end? Can you also recognise them looking at the wave form (time domain)?
What type of window does Matlab use by default to calculate the spectrogram? Why does it not use a rectangular window?
The most prominent harmonics are very close together at the bottom of the spectrogram. One possible way to zoom in on this area, is to reduce the sampling rate of the signal (downsampling effectively rescales the range of the frequency axis). This can be done by applying a low pass anti-aliasing filter and downsampling the signal to the desired lower sampling frequency. In Matlab the function ’resample’ can do this for us. Use this function to resample the signal to a well-chosen sampling frequency, and draw a new spectrogram. With the m-file ’notes.m’ which was provided along with these exercises [1], you can generate the frequencies that correspond to the most frequently used notes in western music. Can you determine which notes are played in the files ’noot1.wav’, ’noot2.wav’ and ’noot3.wav’?

3.2 Designing FIR Filters

3.2.1 FIR Filters

The general input/output equation of a causal FIR filter in the Z-domain is given by:

$$Y(z) = H(z)X(z) = (h(0) + h(1)z^{-1} + ... + h(N-1)z^{-(N-1)})X(z) \quad (3.1)$$

Here the time origin is chosen equal to zero. In the time domain the equation becomes a convolution:

$$y(n) = h(0)x(n) + h(1)x(n-1) + ... + h(N-1)x(n-(N-1)) \tag{3.2}$$

However, because in Matlab the index of a vector always starts from sample 1 (not 0), the first filter coefficient $h(0)$ will be at position 1 in its MatLab vector representation $hcoef$, causing the Matlab representation of the filtering equation to become:

$$y(n) = hcoef(1)x(n) + hcoef(2)x(n-1) + ... + hcoef(N)x(n-(N-1))$$

A convolution can be calculated in Matlab with the function 'conv'. Define your own FIR filter (i.e. choose a set of (randomised or well-chosen) filter coefficients) and calculate its corresponding impulse response using a unit impulse signal as input for filtering.

3.2.2 Exercise: Designing FIR Filters in MatLab

Matlab offers several standard functions for designing FIR filters and their impulse response (or the parameters hcoef). A couple of important functions are 'fir1', 'fir2', 'kaiser', 'kaiserord' (window design method) and 'firls', 'firpm', 'firpmord' (equiripple design). Try to find out how these functions work and design a low pass filter for a 5-times decimation of the speech signal from the previous task. Draw the impulse response, the amplitude and phase characteristics and the step response of the designed filter.
Use the filter and the 'conv' function in a program that allows 5-times decimation (downsampling) of the singing voice.

Bibliography

[1] prof. Verhelst's website: `http://www.etro.vub.ac.be/Personal/wverhelst/`