Project 1
Denoising an Audio Signal: How to Solve the Vuvuzela Problem during the 2010 World Cup

In this project, we will explore the frequency content of real audio signals. In particular, we will work on the practical problem of "denoising" an audio signal; that is, removing noise from an audio signal to enhance its quality.

1 Motivation

As some of you might remember, the 2010 FIFA World Cup was held in South Africa. It soon turned out that a very common habit among the South African football audience was to blow a local instrument called *vuvuzela* during the games. The vuvuzela is basically a horn that produces a monophonic sound typically around the note B flat.

Figure 1: 2010 World Cup audience blowing vuvuzelas. (Image courtesy of Wikipedia)

The problem with the vuvuzela during the 2010 World Cup was that its sound was so loud that it not only raised health and safety concerns, but also practically made it impossible for broadcasting organizations to present the matches. TV and radio audiences often heard only the sound of vuvuzelas during the games. While some of the football authorities proposed banning the use of vuvuzelas from stadiums, others argued that it was a central part of the African football experience: it was all about shouting, dancing, and excitement... In fact, this is what the matches looked like:
https://www.youtube.com/watch?v=bKCIFXqhLzo
https://www.youtube.com/watch?v=TMl1CKt-20o

In what followed, most of the prominent TV channels came up with a signal processing solution that filtered the vuvuzela sound from the ambient noise while maintaining game commentary. This solution turned out to be quite effective, making it possible for billions of people to enjoy this unique sports event comfortably from their TVs or radios.

For more information, please visit: https://en.wikipedia.org/wiki/Vuvuzela

2 Scope of the Project

In this project, we will first learn how to remove the vuvuzela sound from a sample audio signal. We will study a simple signal processing solution that takes the following basic steps to solve the vuvuzela problem:

1. Analyze the spectrum (frequency content) of a noisy audio signal contaminated with the vuvuzela sound

2. Analyze the vuvuzela sound itself, in order to determine the dominant frequencies in its spectrum

3. Denoise the noisy audio signal by suppressing (removing) the frequencies in its spectrum which come from the vuvuzela sound

We will then record our own voice signal. We will inspect its spectrum and determine whether it is similar to the spectrum of the vuvuzela or not.

3 Instructions

As you may recall from your MATLAB exercise session, a "signal" is defined as a function of time. In this project, we will deal with audio signals that can be represented as a function $x(t)$, where t is a time variable, and x is the name of the signal.

Remember also that the "spectrum" of a signal means its frequency content. In particular, an audio signal is generally made up of many spectral components, each of which has a different frequency, i.e., corresponds to different musical note. If high-pitched sounds are dominant in an audio signal, you will immediately notice the high frequencies when you look at its spectrum plot. Similarly, if the audio signal contains rather low-pitched sounds (bass tones), its spectrum will contain rather low frequencies.

Let's now get to work!

3.1 Analyzing the spectrum of the noisy signal

1. Our noisy audio signal is saved in the file x.mat, which you can download from ODTUClass. Load the signal on MATLAB with the command:

   ```
   % Load the noisy audio signal
   load x.mat
   ```

   Listen to the audio signal as follows:

   ```
   fs=44100;
   sound(x,fs);
   ```

   You may recall the sampling parameter `fs` from last week's MATLAB exercise session. `fs` represents the sampling rate: When the original analog sound signal was being recorded and converted to the digital format, `fs` samples were taken from it per second. To be able to listen to a signal, the value of `fs` needs to be provided in addition to the signal itself.

   Comment on what the noisy audio signal sounds like. It's not surprising that TV channels wanted to get rid of the vuvuzela sound. ☺

2. Like any time function, we can also visualize our audio signal. Generate a figure box and plot the noisy signal $x(t)$ as a function of t. Try to display the time axis in seconds, by thinking about how the sampling rate parameter `fs` should be used for this conversion. Label the axes and indicate the title of your plot. Save your plot and provide it in your report.

You can zoom into the signal to look more closely, where you will typically see repetitive patterns corresponding to the vibrations of the musical instruments and the singer's vocal chords in the song recording.

3. Let's now look into the spectrum of our noisy signal. Recall that the MATLAB command `fft` calculates the spectrum of a signal. Type the following commands to compute the spectrum:

```
% Compute the spectrum of the noisy audio signal
X=fft(x);
n = length(x);
f = (0:n-1)*(fs/n);
```

Here the vector `f` contains frequencies in units of Hertz. Compute also the magnitude of the spectrum using the `abs` command. Plot the magnitude of the spectrum as a function of the frequency `f` in Hz. Label the axes and indicate the title of your plot. Save your spectrum magnitude plot for the noisy audio signal and provide it in your report.

Remark: When you plot the spectrum of real signals in MATLAB, you notice that it always consists of two almost identical parts. The "original" spectrum lies at the leftmost part of the plot (near the frequencies around 0), while there is a "replica" of the original spectrum produced at the rightmost part of the plot. When inspecting the spectrum of signals, you can ignore the replica at the right and concentrate only on the "original" spectrum lying at the left.

4. It may not be intuitive to give meaning to the spectrum of the noisy signal at the first glance. If you play with the spectrum plot a little bit (by zooming in and looking closely), you may notice that in fact it roughly looks like in the following picture:

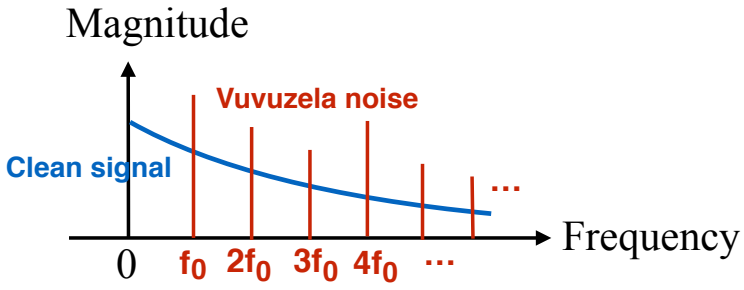

Figure 2: Spectrum of the noisy audio signal

Here, the blue part of the spectrum corresponds to the "clean" component of the audio signal - in this case it is a famous Queen song. On the other hand, the red part of the spectrum comes from the vuvuzela sound. The overall spectrum of the noisy signal is the sum of these two. You may notice that the spectrum of the vuvuzela sound comes as very strong spikes, peaking at regularly spaced frequency intervals. In fact, the vuvuzela spectrum concentrates very sharply around frequencies that are integer multiples of a base frequency f_0; going like f_0, $2f_0$, $3f_0$, and so on.

Look at the noisy spectrum you plotted in MATLAB, and comment whether you can already spot where the clean signal and the vuvuzela sound lie in the spectrum. Can you figure out around which value f_0 should be? It may not be obvious for everyone to see. That's why we will look into the spectrum of the vuvuzela sound in more detail in the next section.

3.2 Understanding the spectrum of the vuvuzela sound

1. It may have been hard for you to distinguish the clean signal spectrum from the vuvuzela spectrum in the previous part. This is not a problem as we know exactly what the vuvuzela sounds like, so we can analyze its spectrum conveniently. Download the signal `v.mat` from ODTUClass, which is an example vuvuzela sound recording. Load the vuvuzela sound signal and listen to it as follows:

```
% Load the vuvuzela signal
load v.mat

% Listen to the vuvuzela signal
sound(v,fs);
```

2. Now, plot the vuvuzela sound signal as a function of time in seconds. Label the axes properly and provide your plot in your report. You can zoom in to look at the vibration cycles of the vuvuzela sound more closely.

3. We can now analyze the spectrum of the vuvuzela. Compute the spectrum `V` of the vuvuzela signal `v` using the `fft` command. Then, compute the magnitude of the spectrum using the `abs` function. Plot the magnitude of the vuvuzela spectrum as a function of the frequency `f` in Hz. Label the axes and provide your plot in your report.

4. Now in the vuvuzela spectrum, zoom into the region where you see the peaks and examine them more closely.

   In fact, the spectrum of any monotone sound from a musical instrument (that is, a single note played by a musical instrument such as violin or flute) is structured in the same way: It contains sharp peaks at a base frequency f_0 and its integer multiples $2f_0, 3f_0, \ldots$ as illustrated in Figure 2. We will learn the reason for this in our third-year signal processing course EE301. The base frequency f_0 is called the *fundamental frequency* and its integer multiples $2f_0, 3f_0, \ldots$ are called the *harmonics*.

   Inspecting the spectrum of the vuvuzela sound, can you figure out its fundamental frequency f_0? At which frequencies do you observe the harmonics? Remember that you must focus only on the left part of the spectrum and ignore the replica at the right.

3.3 Removing the vuvuzela sound from the noisy audio signal

1. Now that we have understood the structure of the vuvuzela sound, we are finally ready to remove it from our noisy audio signal. For this purpose, we will design a "spectral mask" that aims to keep the frequencies belonging to the clean part of the audio signal, while removing the frequencies coming from the vuvuzela noise. As illustrated in Figure 3, in order to suppress the undesired frequencies, the spectral mask will have the value 0 near the frequencies $f_0, 2f_0, 3f_0, ...$, and it will have the value 1 otherwise. It is often safer to consider a narrow window around each undesired frequency; so let's say that we would like to remove the frequencies in the interval $[f_0 - \Delta, f_0 + \Delta]$ for the fundamental frequency, then $[2f_0 - \Delta, 2f_0 + \Delta]$ for the second harmonic, and repeat this until some K-th harmonic of our choice.

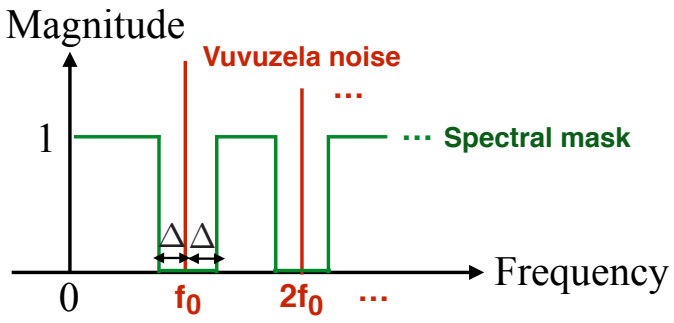

Figure 3: Spectral mask designed for removing the vuvuzela noise from the audio signal

As an engineer designing a denoising filter, you should now do the following:

- First, based on the spectrum of the vuvuzela sound you observed in Section 3.2, determine a suitable value for the fundamental frequency f_0 (in Hertz) and set the variable `f0` to this value.

- Then, observing the spectrum of the vuvuzela, determine a suitable value for the interval parameter Δ and set the variable `delta` to this value.
- Finally, determine a suitable value of the parameter K by deciding how many harmonics you would like to remove from the noisy spectrum. (Look only at the original spectrum at the left; ignore the replica of the spectrum at the right). Set the variable `K` to the integer value you chose.

Let's now form a spectral mask with the parameters you have chosen:

```
% Form the spectral mask to be applied to noisy spectrum
spectrum_mask=ones(length(X),1); % The mask initially consists of 1's
for k=1:K
    % Form a vector containing the frequencies around k*f0
    masked_freq_low=round((f0*k-delta)/fs*n):round((f0*k+delta)/fs*n);
    % Make sure that the range of frequencies stays in a valid range
    masked_freq_low(masked_freq_low<=0)=[];
    masked_freq_low(masked_freq_low>n)=[];

    % Take care of the replica at the right as well
    masked_freq_high=n+2-masked_freq_low;
    masked_freq_high(masked_freq_high<=0)=[];
    masked_freq_high(masked_freq_high>n)=[];

    % Set the selected frequencies of the spectral mask to 0
    spectrum_mask(masked_freq_low)=0;
    spectrum_mask(masked_freq_high)=0;
end
```

Plot your spectral mask as a function of the frequency in Hz. Label the axes and provide your plot in your report. Comment whether the spectral mask looks like what you want (you might need to zoom into the frequencies of interest for better visibility).

2. If you are happy with your spectral mask, you can now apply it to the noisy signal as follows: All you need to do is simply multiply the spectrum `X` of the noisy signal with the spectral mask `spectrum_mask`, so that the frequencies where the mask is 0 will be eliminated from the signal, while the other frequencies will remain. Name the result of the product as `Xdenoised`, which is nothing but the spectrum of the denoised signal. (Think about the two product operators you learnt in MATLAB. Which one should you use here?)

   Obtain the magnitude of the spectrum of your denoised signal and plot it as a function of frequency. Label your axes and provide your plot in your report. Comment on what you see. Have the undesired peaks coming from the vuvuzela noise disappeared?

3. We can now listen to the denoised signal. We should first convert the spectrum `Xdenoised` back to a time signal. We can use the `ifft` function for this, which applies the inverse of the operation in `fft`. Compute the denoised signal as follows:

   ```
   xdenoised=real(ifft(Xdenoised));
   ```

   You can listen to the denoised signal as

   ```
   sound(xdenoised,fs);
   ```

   Comment on what you hear. Have you been able to remove the vuvuzela noise from the audio signal? Also, try different values for the Δ and K parameters and comment on their effect on the quality of the denoised signal. What does the signal sound like when Δ is chosen too small or too large?

3.4 Analyze the spectrum of your own voice!

In this last part, we will analyze the spectrum of our own voice signal. Simple monophonic human voice signals, such as single-vowel sounds chanted as a single musical note, or the whistling of a single musical note, have a spectrum that is very similar to that of musical instruments such as the vuvuzela.

1. First practice a little bit how you can make a single-vowel sound with your own voice. It should sound like an "uuuu", or "aaaa" as if you were singing it in a song. Try not to change the pitch of your voice while doing this, i.e., you should sing your "uuuu" as a single musical note. Alternatively, if you are a good whistler, you can try to whistle a one-note tune in a similar way.

2. Now, record your single-vowel sound for a few seconds and save it as an audio file using any software of your choice on your computer, phone, etc.

3. Open your voice file in MATLAB. MATLAB can process many common audio file formats such as mp3 and wav. If your filename is for instance "vowel.m4a", you can load it as

   ```
   [y,fs] = audioread('vowel.m4a');
   ```

   Here `y` will give you the voice signal, and `fs` is the sampling rate parameter we saw earlier. Listen to your voice using the `sound` command. Plot your voice signal as a function of time in seconds and provide it in your report.

4. Now compute the spectrum of your voice signal. Plot the magnitude of the spectrum with respect to the frequency in Hz as you did in the previous exercises and provide it in your report.

5. Inspect the spectrum of your voice signal. Comment whether its spectrum looks like the spectrum of the vuvuzela. What is the fundamental frequency of your voice signal? Are there any harmonics in its spectrum?