Exercise 2 Audio Signals in the Time and Frequency Domains

1 Objectives
- Experiment with different resolutions of audio signals.
- Develop an understanding of Time and Frequency Domains.
- Experiment with basic transformations and filtering in both domains.

2 More on Sampling and Quantising

Sampling is the process of selecting regularly spaced points in time over a signal. In other words, it is the process of recording the level of a signal at certain regular intervals. It is the first step in the digitisation of a signal. Quantisation is the process of turning a continuous or analogue signal into one which has discrete numerical values at each point in time. Voice is sampled on the telephone at 8,000 samples/sec with each sample consisting of 8 bits. In audio applications, voltage is typically quantised into a 16-bit number, that can represent 65536 different levels of voltage. The numbers are usually proportional to the input voltages, but logarithmic relationships have also been used. It is important to note that quantisation is a separate process from sampling.

Nyquist’s theorem states that if we sample a signal in which the highest frequency we wish to reproduce correctly is $f$ then we must sample the signal at a minimum frequency of $2f$. For this sampling rate, the frequency $f$ is known as the Nyquist frequency, $f_N$.

A signal which is sampled below this minimum rate is undersampled, and the effect of doing this is to generate spurious contributions of lower frequencies in the reconstructed signal. This process is known as aliasing.

Analogue-to-digital conversion, is the overall process of taking an analogue signal, such as the fluctuating voltage from a microphone and turning it into a digital signal, consisting of a stream of numbers.

In the previous Laboratory Experiment you analysed sampling and quantising images. Quantising and sampling follow the same rules for audio and images, both are considered as signals. The difference is now we will be dealing with 1-D signals.

3 Hands-on Sampling and Quantising
First you will need to execute the addpath command, and then run the analogue-digital conversion script:
Analogue to Digital Conversion

A figure will appear with an Analogue to Digital conversion programme. An image of the program interface appears in figure 1. Basically, you can select from 3 types of signals, sine wave, square wave and triangular wave, you can modify the frequency of the signal, you can select the sampling frequency and you can select the number of bits for quantising. With these parameters you can convert the signal from analogue to digital and back.

Exercises

- Try different signals, frequencies and number of bits. When you can see problems of aliasing?
- How do the number of bits introduce noise?
4 Sampling and Quantising Audio Signals

Matlab is an extremely useful and powerful tool for handling audio files. You can visualise and hear an audio signal in the following way.

```matlab
>> load handel
```

This loads a (tiny) bit of the Messiah by Handel, the well-known Hallelujah. You will notice that two variables are loaded, $F_s = 8192$ samples/s, the sampling rate and $y$ a $73113 \times 1$ array. This looks something like the signal in figure 2.

**Exercises**

- To play this, (provided that your computer has the proper output, some of the computers in the MSc room do not have loud speakers. If you have your own computer with Matlab do it at home), type:
  ```matlab
  >> sound(y*0.5)
  ```

- The 0.5 value controls the volume, (try not to disturb other users!). You can manipulate $y$ as any other matrix, try for instance:
  ```matlab
  >> sound(y(end:-1:1)*0.5); % Will play the sound backwards
  >> sound(y(1:2:end)*0.5); % Plays half of the samples
  >> sound((y>0)*0.5); % Plays only positive part
  ```

- Now let’s quantise the signal. Use the `uencode` command for quantising. Besides you will need to keep the values between $-1$ and $1$:
  ```matlab
  >> y2=double(uencode(y,2))/1.5-1;
  ```

![Figure 3. Difference between the original signal and the quantised signal](image3.png)

![Figure 4. Interpolated and the original signal](image4.png)
That will quantise to 2 bits, i.e. 4 levels, what quality can you expect? Try now quantising with more bits of resolution, at which point does it sound like the original? Would this be the same if you would have a high fidelity equipment? Figure 3 shows the difference between the original signal and the sampled signal.

- There are several ways to reduce the number of samples, an easy one would be to decimate (sub-sampling) the signal and then interpolate it (to keep it in the same pitch):

```matlab
>> x2=interp(decimate(y,2),2); % Down sample by 2
>> x4=interp(decimate(y,4),4); % Down sample by 4
```

Play the signals and listen to them carefully, can you notice the degradation? Increase the down sampling and listen. Figure 4 shows a fragment of the interpolated signal.

- The previous instructions perform a linear interpolation, how could you interpolate using nearest neighbours? Try it and listen the difference, which one sounds better?

- Load other sounds and repeat the experiments. You can find some in

  /package/matlabr12/toolbox/matlab/audio/