Electrical Engineering 3TP3
Laboratory 2
Signal Convolution

In this lab we use Matlab to compute and study signal convolution. First, some Matlab convolutions are computed and compared with their hand-computed counterparts. Audio recording software is then used to record an audio track that is read into Matlab. A telephone network channel is simulated where there is an echo that creates a distorted version of the original audio signal. This is done both directly in Matlab and using convolution after forming the impulse response of the echo channel. The echo parameters are then varied to test acceptable echo levels using subjective listening tests. Finally, we form the impulse response of a reverberation channel and test its impact on the audio track using subjective listening tests.

1 Preparation

1. Make sure that you attend the lectures where is lab is introduced.
2. Signal Convolution Using Matlab: Discrete-time convolution can be easily computed in Matlab using the `conv` function. If `A` and `B` are vectors, then `y = conv(A, B)` returns a vector `y` that is the convolution of `A` and `B` and has a length of `length(A) + length(B) - 1`

2 Experiments

1. Do Question 2.7 from the textbook. In this question you are given three sets of discrete-time signals `x[n]` and `v[n]`. In Part (a) you manually compute the convolutions of `x[n]` and `v[n]`. Then in Part (b) you use the Matlab `conv` function to verify your results. In Part (b), write Matlab code that do stem plots of `x[n]`, `v[n]` and `x[n] * v[n]` (Use the subplot command to plot them above each other). Include the Matlab code and your plots in your writeup.
2. There are many available audio recording applications. An example is ocenaudio, which is a free cross-platform audio processing app. You can download and install it at[1]:

   `https://www.ocenaudio.com/download`

   Run ocenaudio and click on the record button to record either you or your lab partner speaking continually for about 15 seconds. Use the parameters 44100 Hz (sampling rate), Mono and 16 bits/sample. Export the audio clip to an audio file (e.g., `my_speech_clip.wav`) using the WAV format and default settings, e.g., 16 bit, linear PCM.

[1] You may have to override an installation security warning since the installer is not signed by the author.

3. Use Matlab to read in the audio file from Part 2 as follows.

```
[signal, Fs] = audioread('my_speech_clip.wav');

L = length(signal);  % Number of samples in the signal.
T = 1/Fs;            % Sampling period in seconds.
t = [0:L-1]*T;       % Time vector in seconds.
```

   After executing these statements, `signal` is a column vector containing the speech samples from the file, `Fs` is the sampling frequency and `L` is the number of samples.

4. Assume that this audio signal is transmitted through a telephone network where distortion in the form of an echo of the signal is created. The person listening hears the original signal plus an echoed version at a reduced amplitude factor, α, i.e., if the original signal is $f_s(t)$, then the received signal heard by the listener is $f_r(t) = f_s(t) + \alpha f_s(t - t_e)$, where t_e is the echo delay in seconds.

   In Matlab, define a variable, `Te`, which is the echo delay *in msec*, and use `signal` to create the received signal, `signalplusecho`. You can do this using a delayed/shifted version of `signal`. Then you can create a new wav file as discussed below.

   It is a good idea to first re-scale your output data to ensure that the magnitude of any of the samples does not exceed 1. Otherwise, clipping of the signal may occur when playing the sound file (Matlab may give you a warning if this happens). To re-scale, do the following to your output sound vector `signalplusecho`.

```
signalplusecho = signalplusecho/max(abs(signalplusecho));
```

   Then you can write a new wav file, `speechwithecho.wav`, using the following.

```
audiowrite('speechwithecho.wav', signalplusecho, Fs);
```

   You should now be able to listen to the distorted signal by playing the `speechwithecho.wav` audio file. Include your Matlab code in your report and Upload your wav file to the lab dropbox on Avenue To Learn when you upload your lab report.

5. Write Matlab code that creates the echo described above but instead uses convolution. To do that you need to find the impulse response, `IR`, of the echo channel. Explain your choice of impulse response and include that in your writeup.

   Now you can use the `conv` Matlab function to take the convolution of `IR` and `signal`. Save the result as a new wav file and verify that things are working properly. Upload your wav file to the lab dropbox on Avenue To Learn when you upload your lab report.

6. Experiment with different values of `Te` when `alpha` is equal to 1. How small does `Te` have to be before the quality of the speech is acceptable? Does your answer change when the value of `alpha` is decreased?

7. Design an impulse response vector that will create "reverberation". This is when the resulting signal includes the original plus multiple echos of decreasing intensity. One way of doing this is to create echoes that are exponentially decreasing in amplitude, i.e., you can create echos using

$$\sum_{i=1}^{N_e} \alpha^i f s(t - it_e),$$

where N_e is the number of echos.

Repeat Part 6 using your reverberated signal. Include your Matlab code in your writeup. Upload your wav file with reverberation to the lab dropbox on Avenue To Learn when you upload your lab report.

Report: Submit a single PDF file report for the lab plus the wav files mentioned above. Upload the files separately, i.e., not as a zip file. Each group (2 maximum) is responsible for their own experiments and writeup. Include in your writeup a description of everything that you did including all data and Matlab code.