1 Introduction

In this exciting lab we will introduce the use of the Fast Fourier Transform (FFT), windowing, zero padding, interpolation and decimation. These operations constitute an important class of tools in digital signal processing. We will investigate their application in the context of speech synthesis, in which samples of human speech are used to “train” a specialized algorithm which generates recognizable speech. You will perform windowing, upsampling and downsampling on a digital voice message, quantitatively analyze the behaviour of the speech synthesizer, and use your ears to qualitatively assess its behaviour.

2 Notes on the Lab

This lab makes little use of SIMULINK. Most of the exercises require you to use the MATLAB shell environment. As before, to set sampling rates and to initialize the workspace, run the start-up script lab2init before running simulations. Run the script lab2load to produce the required SIMULINK libraries.

3 Part I – Fundamental Operations

For this section of the lab, set the sampling period, $T_s$, equal to 1. You will be asked to study various DSP operations by simulating the system shown in Figure 1 and by running a number of small scripts which you write.

1. For the following, use the Signal Source A-block, found in Library: sources. Run l24lp1 to bring up your SIMULINK work window.
   (a) Compare the spectra of $x_a(t)$ and $x[n]$. Use the spec_ct and spec_dt functions as in the previous lab.

![Diagram](image)

Figure 1: System under investigation for Part I.
(b) Zero pad \( x[n] \). That is, write a script to generate

\[
x_1[n] = \begin{cases} 
    x[n] & 0 \leq n \leq 1023 \\
    0 & 1024 \leq n \leq 3071 
\end{cases}
\]

(c) Compare the magnitude of \( X_1[k] \) (\( X_1[k] := \text{fft}(x_1, 3072) \)) with the magnitude of \( X[k] := \text{fft}(x, 1024) \). What is the effect of zero padding? (In all frequency-domain plots, plot the magnitude in dB and normalize the frequency axis properly.)

(d) Using the \textit{interp} function in MATLAB, up-sample \( x[n] \) by a factor of 3 to generate the sequence \( x_u[n] \). Plot this sequence. What is its length?

(e) Compare the magnitude of \( X_u[k] := \text{fft}(x_u) \) with the magnitudes of \( X_1[k] \) and \( X[k] \). What is the effect of interpolation that you observe?

(f) Write a MATLAB script to insert two zeros after each sample of \( x[n] \) to produce the 3072-point signal \( x_{ZT}[n] \). Compare the spectra, \( X_{ZT}[k] \), \( X[k] \), and \( X_u[k] \). How does this type of interpolation effect the signal properties in the frequency domain? Discuss how \( X_{ZT}[k] \) would change if one zero, instead of two, was inserted after each sample of \( x[n] \).

2. Using point-wise multiplication, generate the sequences

\[
x_H[n] = x[n]w_H[n] \\
x_B[n] = x[n]w_B[n] \\
x_A[n] = x[n]w_A[n]
\]

using the 1024-point Hamming Window (\textit{hamming}), \( w_H \), the 1024-point Blackman Window (\textit{blackman}), \( w_B \), and the 1024-point Bartlett Window (\textit{bartlett}), \( w_A \). Plot these signals together in a single plot. How do they compare in the time-domain? Compare the spectra of the signals. Describe how the time-domain characteristics of the window functions affect the spectral characteristics (recall that multiplication in the time-domain is equivalent to convolution in the frequency domain), specifically, address the trade-off between spectral leakage and resolution.

4 Part II – Speech Synthesis

In this portion of the lab, you will be working with the MATLAB script, \textit{ss0.m}, provided with this experiment. This program implements a speech synthesizer using techniques of digital signal processing. Before describing what your specific tasks will be, we’ll explore the generation of speech.

Human speech is produced by the excitation of the vocal tract, which includes the larynx, pharynx, mouth, and nasal cavities. Under muscular control, the shape of the vocal tract can be altered to provide variations in sound. The excitation of the vocal tract is effected by the vocal cords, which create vibrations. These vibrations are processed by the filtering action of the vocal tract to make the sounds that we recognize as speech. The human speech system is illustrated in Figure 2. Because the shape of vocal tract changes with time, a time-varying model of the vocal tract is used. We represent the vocal tract using an \textit{auto-regressive} model, in which the discrete-time speech signal, \( s[n] \), is given by a linear combination of its past values, plus the current value of the excitation signal, \( e[n] \), multiplied by a gain parameter. In mathematical terms,

\[
s[n] = \sum_{i=1}^{N} \alpha_i[n] s[n-i] + G[n]e[n]
\]
where $\alpha_i$ and $G$ denote the time-varying vocal tract model parameters. We assume that these parameters are actually constant over small intervals of time— for good synthesis, the time-interval should be around 20-ms. To produce speech, we will need to determine a potentially large data set of parameters. Therefore, speech synthesis is greatly facilitated by the ease with which we can determine the model parameters.

Why use an auto-regressive model for the vocal tract? Of all the forms of linear, constant-coefficient difference equations, the auto-regressive model provides the easiest method to find the parameters$^1$. Thus, from sampled speech, we can quickly “train” our synthesizer.

4.1 Speech Synthesis Details

The auto-regressive model in Equation (5) is used to generate speech as follows. First, a digitized human speech signal is divided into blocks of approximately 20-ms in length (we use a fixed sampling rate of 8-kHz throughout the lab, therefore, a 20-ms block length corresponds to 160 samples). As mentioned above, we assume that the parameters $\alpha_i$ and $G$ are constant over each block. A special algorithm, encoded in the `findcoef.m` MATLAB function, determines the model parameters for any given block of data. This function is called by the `ssth.m` MATLAB script which produces the complete vocal tract model by applying `findcoef.m` to all blocks of data. The procedure for determining filter coefficients from sampled speech, known as “training,” is illustrated in Figure 3. Note that for any segment of speech, the speech signal is modeled as the impulse response of the filter corresponding to that block of data. Thus, your first task in this lab is to design a suitable excitation signal (in MATLAB code) for your synthesizer.

**HINT:** Bear in mind that the speech synthesizer output is generated as a succession of impulse responses. The impulse responses are of the filters corresponding to each segment of the recorded data.

You may record a segment of speech if your station has a microphone$^2$, or choose any of the .wav fragments which have been included in the lab, to get your synthesizer working.

---

$^1$The model parameters can be found by using a linear set of equations known as the Yule-Walker equations. You are not required to understand exactly how this is accomplished for the purposes of these exercises. Just be aware of it—the details are in the `findcoef.m` function included with this lab.

$^2$Please note that while profane messages are unacceptable, BONUS MARKS may be issued for recordings which exhibit the virtues of digital signal processing! ☺️
In this lab, you are asked to answer the following general questions. It is up to you to provide appropriate analyses to satisfy your audience, which you can assume consists of a group of inquisitive, well-adjusted, psychologically-fit engineers. (Hint: you may wish to include time- and frequency-domain analyses.)

1. How close do the original speech signal and the synthesized speech signal match? You may compare the two on the basis of a single block of data. Your answer should include a quantitative assessment.

2. How does windowing affect the accuracy of the synthesis? Note that a Hamming window is used by default in ss0.m.

3. How does the order \((N)\) of the auto-regressive model (Equation (5)) affect synthesis (try values between 8 and 15)?

4. How are the filters initialized at the start of each block in ss0.m? Why is this so?

5. Consider the cascade system consisting of a downsampler followed by an upsampler shown in Figure 4. To what extent can you downsample the synthesized speech signal before the reconstituted version becomes unintelligible? Why might you want to downsample it?

6. How could this synthesizer be used to create a system that would accept words typed in via the keyboard and “say” them?