Signal Processing using Matlab

Lesson 3

Dealing with Sound Signals

3.1 Introduction

In this lesson, we will explain how we deal with a common type of signal: sound signals. First of all, we will explain how we could load a sound file into the Matlab workspace. Afterwards, we will explain how sound signals are represented and stored. We will then explain how sound signals may be played or written to wave files. Finally, we will demonstrate the application of some signal processing operations to sound signals.

3.2 Importing Sound Signals

Sound signals may be imported using two methods. The first method is using the import wizard. One way of accessing the import wizard is through the file menu as depicted below.

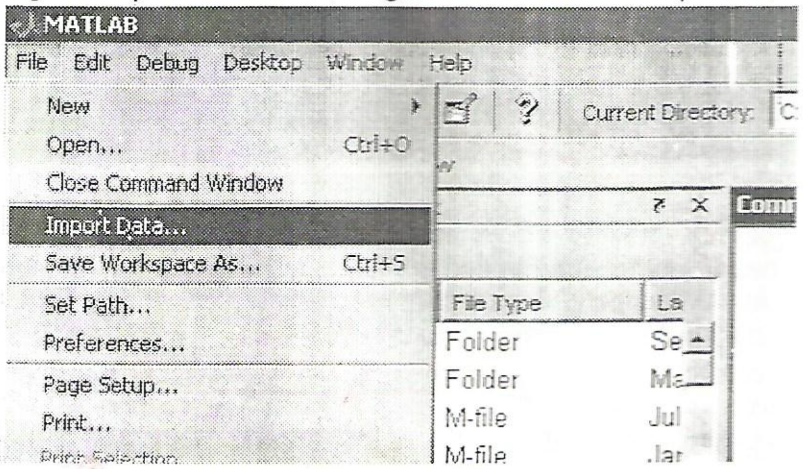

Figure 3.1

You will then browse until you reach the desired file, select it, then click "open". This method is common for importing sound, images, video, or data files. However, you can only load sound files of type .wav or .au.

The second method uses specialized sound importing functions. There are specialized functions for loading files of type .wav and .au that come with the Matlab package. There are also many toolboxes on the internet that give you files that could load other common sound file types, such as mp3. You can reach the MP3 toolbox by googling the following string "MATLAB Central File Exchange - MP3WRITE and MP3READ".

Now, we have two functions to use to load wave files and mp3 files: "wavread" and "mp3read". We will not deal with .au files because they are only used in Linux. In order to use wavread or mp3read, the file we want to load must be situated in a folder that exists on the Matlab search path. You can just put the file in the "Work" directory of Matlab, or you can go to Set Path in the file menu and the locations of your files.

The wavread function may be used with a variable number of input and output arguments. The following instructions show various calls to wavread.

```
>>Y=wavread('FileName');
>>[Y,Fs]=wavread('FileName');
>>[Y,Fs,Nbits]=wavread('FileName');
>>[Y,Fs]=wavread('FileName',N);
>>[Y,Fs]=wavread('FileName',[N1 N2]);
```

The first instruction shows the minimal form of using wavread. The input argument 'FileName' is a string containing the name of the wave file to be imported. The output argument is the variable in which the sound data will be stored. The second instruction features an additional output argument. When there are two or more output arguments, the second output argument will convey the sample rate of the wave file. The third instruction features a third output argument. This output argument conveys the number of bits per sample used in storing the file.

In the fourth and fifth instructions, we have added a second input argument. If this input is a scalar (N), as in the fourth instruction, Matlab will load the first N samples from each channel of the file. If this input is a two element vector ([N1 N2]), Matlab will load the samples number N1 through N2 from each channel of the file. This may be useful for loading only a small part of a large wave file.

3.3 Representation of Sound Signals in Matlab

In Matlab, a sound signal may either be a vector (row or column), or a two-column matrix. When the sound signal is a vector, the sound is monophonic. When the sound signal is a two-column matrix, the sound is stereophonic. The left column contains the samples comprising the left speaker signal. The right column contains the samples comprising the right speaker signal.

The number of elements per channel is equal to the duration of the file multiplied by the sample rate of the signal. The values of the samples lie in the range from -1 to 1.

3.4 Playing and Writing Sound Signals

When you process a sound signal, you will normally want to play the new signal to evaluate the performance of your code. You may play a sound signal using the "sound" function. This function has the syntax shown below.

```
>>sound(x,Fs);
```

The first input argument x is the sound signal to be played. It may be monophonic or stereophonic. The second input argument Fs is the sample rate of the sound signal. You may try playing a sound signal with a sample rate that is less than or greater than its original sample rate. However, you will discover that the sound signal sounds slower or faster than normal.

After you finish processing the signal, you might want to save the signal to a wave file. You may do so by using the "wavwrite" function. If you have downloaded the mp3 toolbox, you may also use the "mp3write" function to write the sound signal to an mp3 file. The following instruction shows the syntax of using "wavwrite".

```
>>wavwrite(SoundSignal,Fs,'OutputFileName);
```

3.5 Example of Processing Sound Signals

In the following example, we will load a wave file, process its sound signal, and then write the result in a new wave file. We want to extract the first 4 seconds of the wave file, and apply a fade-in and a fade-out effect. The sound should fade in in two seconds, and then fade out in two seconds. The file is assumed to be stereophonic and to have an initial duration longer than 4 seconds. We will list the instructions and then explain how they work.

```
>>[Y,Fs]=wavread('File1');
>>% Take the first 4 seconds only
>>Y=Y(1:4*Fs,:);
>>% Fade-In Ramp
>>Ramp_In=linspace(0,1,2*Fs);
>>% Fade-Out Ramp
>>Ramp_Out=linspace(1,0,2*Fs);
>>% Fading In
>>Y(1:2*Fs,1)=Y(1:2*Fs,1).*Ramp_In';
>>Y(1:2*Fs,2)=Y(1:2*Fs,2).*Ramp_In';
```

```
>>% Fading Out
>>Y(end-2*Fs+1:end,1)= Y(end-2*Fs+1:end,1).*Ramp_Out';
>>Y(end-2*Fs+1:end,2)= Y(end-2*Fs+1:end,2).*Ramp_Out';
>>sound(Y,Fs);
>>wavwrite(Y,Fs,'File2');
```