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.

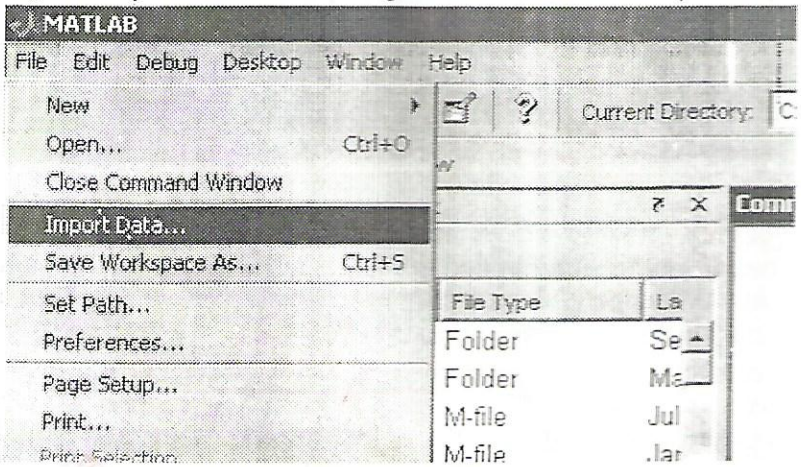


Figure 3.1

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';
```

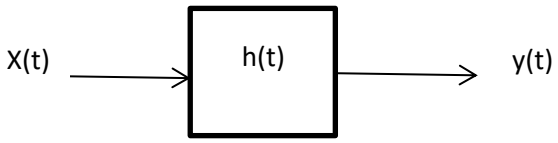
```
>>% Fading Out
>>Y(end-2*Fs+1:end,1)= Y(end-2*Fs+1:end,1).*Ramp_Out';
>>Y(end-2*Fs+1:end,2)= Y(end-2*Fs+1:end,2).*Ramp_Out';
>>sound(Y,Fs);
>>wavwrite(Y,Fs,'File2');
```

System representation

3.6 Time domain (Impulse response):

- The system is represented by its impulse response, and the relation between the input and the output of the system is as follows:

  y(t) = x(t) * h(t)



- Matlab performs only discrete convolution:

  Z = x*y

  Z[n] = x[n] * y[n]

  $= \sum x[k]y[n-k]$

  To implement continuous convolution, we must approximate it into discrete one:

  Z(t) = x(t) * y(t)

  $= \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau$

  $\cong \sum x(\tau)y(n-\tau)\Delta\tau$

  $\cong Ts.$ discrete convolution (x,y)

3.7 Matlab function of convolution:

For continuous signals,

Z = $\frac{1}{fs}$ conv(x,y)

- Start time for the output of the convolution (z) = start time of (x) + start time of (y)
- End time for the output of the convolution (z) = End time of (x) + End time of (y)
- Length of the output (z) = length of (x) + length of (y) - 1

3.8 Z-domain

- Systems can also be represented by difference equation. The output depends on the input and previous outputs:

  Ex: Y[n] - $\frac{1}{4}$ y [n-1] - $\frac{1}{8}$ y [n-2] = x[n] + x[n-1]

- Matlab deals with the difference equation, via z-transform transfer function:

  Ex: Y(Z) $-\frac{1}{4}\,Z^{-1} - \frac{1}{8}\,Z^{-2} = X(Z) + Z^{-1}\,X(Z)$

  H(Z) = $\frac{Y(Z)}{X(Z)} = \frac{1+Z^{-1}}{1-\frac{1}{4}Z^{-1}-\frac{1}{8}Z^{-2}} = \frac{b}{a}$

  b: coefficients of X, numerator of the transfer function = [1 1]

  a: coefficients of Y, denominator of the transfer function = [1 $\frac{-1}{4}$ $\frac{-1}{8}$]

  Ex: H(Z) = $\frac{Z^2}{Z^2+2\,Z+3}$, coefficients ordered in *descending* powers of *z*

  b = [1 0 0]

  a = [1 2 3]

  H(Z^{-1}) = $\frac{1}{1+2Z^{-1}+3\,Z^{-2}}$, coefficients ordered in *ascending* powers of z^{-1}

  b = [1]

  a = [1 2 3]

3.9 Matlab syntax of tf

- h = tf (num, den, Ts)

  This syntax generates the transfer function H(Z), which is the default, where Ts is the sampling time.
  To generates the transfer function H(Z^{-1}) :
  h = tf (num, den, Ts,'variable','Z^{-1}')

- Y = filter (b,a,X)

  Ex: Find the impulse response and step response of N points

  Impulse response is the output of the system when the input is $\delta[n]$
  h = filter (b, a, [1 zeros(1,N-1)])
  For step response:
  s = filter (b, a, ones (1,N))

Matlab instructions for lesson 3

%%Filters

```
>> b = [18 8 ];
>> %impulse
>> y = filter(b, 1, [1 zeros(1, 9)]);
>> stem(y);
>> %step
>> y = filter(b, 1, ones(1, 10));
>> stem(y);
>> b = [18 8 ];
>> a= [6 5 1];
>> % step:
>> y = filter(b, a, ones(1, 10));
>> stem(y);
```

%%Convolution

```
>> x = [ 1 2 0 1];
>> stem(x);
>> h = [ 2 2 1 1];
>> stem(h);
>> y = conv(x, h);
>> stem(y) ;
```

Signal Processing using Matlab

Lesson 4

Signal Processing in Frequency Domain

4.1 Introduction

In this lesson, we will explain how we transform a signal from time domain to frequency domain and vice versa, and how to carry out frequency domain operations. We will start by studying the FFT and IFFT functions, and the auxiliary FFT shift function. We will then consider two examples of operations in the frequency domain.

4.2 FFT and FFT-Related Functions

The Fast Fourier Transform (FFT) is a fast algorithm for computing the discrete Fourier transform of a discrete time limited signal. The following syntax shows how the FFT function is used.

```
>>X=fft(x);
```

In this instruction, x is the time domain signal, and X is its corresponding FFT result. The length of X is the same as the length of x.

There are several steps that are typically carried out after fft is invoked. The first typical step is invoking 'fftshift', a function that divides X into two halves, and swaps the position of the first and second half. This operation is usually required because the zero frequency location in the vector returned by FFT is not in the middle as we require, but at the start of the vector. After invoking the 'fftshift' function, the DC component is brought to the middle of the vector. This makes it easier in many cases to visualize the positive and negative frequency halves. The following two figures illustrate the effect of 'fftshift'. The spectrum in figure 4.1 (a) was generated without invoking fftshift. The spectrum in figure 4.1 (b) was generated using fftshift.

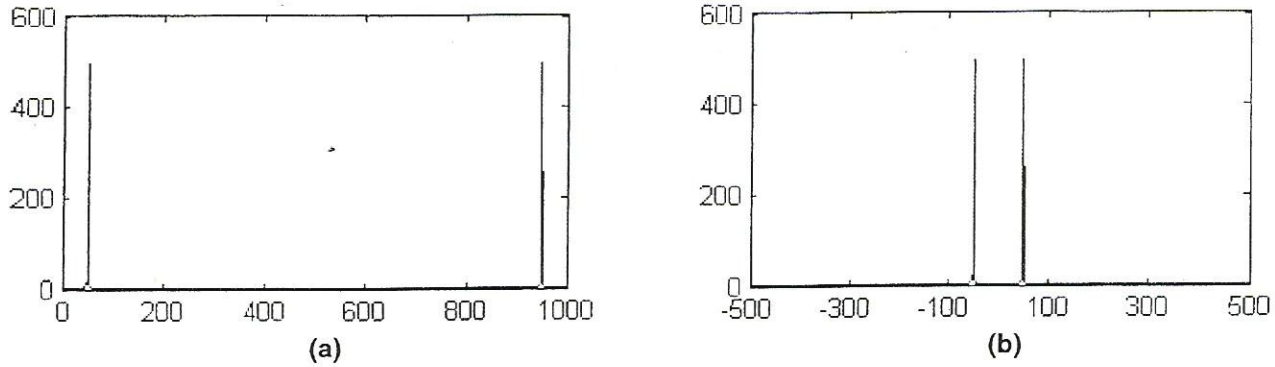


Figure 4.1

Note that the spectra plotted in figure 4.1 are for a cosine wave of frequency 50 Hz. The second plot is more illustrative of the positive and negative frequency halves.

We also invoke the 'abs' (absolute) function to return the magnitude or the 'angle' function to return the phase of each element in the complex spectrum X as shown below.

```
>>Xmagnitude=abs(X);
>>Xphase=angle(X);
```

The instructions we have written so far are responsible for generating the data for the y-coordinates of the plot points. Now, we wish to generate the data for the frequency-coordinates of the plot points. The vector of frequency coordinates is called the "frequency base vector". It may be generated using the following instruction:

```
>>Fvec=linspace(-fs/2,fs/2,Ns);
```

In the previous instruction, fs is the sample rate of the signal, and Ns is its number of samples. To sum up, let us consider the following example, in which we plot the magnitude and phase spectra of a signal comprising of the sum of two sinusoids.

```
>>t=linspace(0,5,1000);
>>y=0.5*cos(60*pi*t)+0.3*sin(80*pi*t);
>>Y=fftshift(fft(y));
>>Ymag=abs(Y); Yphase=angle(Y);
>>Fvec=linspace(-100,100,1000);
>>figure; plot(Fvec,Ymag); title('Magnitude Spectrum');
>>figure; plot(Fvec,Yphase); title('Phase Spectrum');
```

The lat FFT-related function we will consider is the inverse FFT (ifft) function.

```
>>x=ifft(X);
```

4.3 Examples on Frequency-Domain Operations

In this section, we will consider two examples of frequency-domain operations: convolution, and filtering. We will carry out convolution of two signals by multiplying their FFT results. We will carry out filtering by zeroing the frequency components of unwanted coefficients.

4.3.1 Convolution by Multiplication in Frequency Domain

In the following code, we want to convolute two time domain signals x1 and x2. We do so by first taking their FFT to get X1 and X2, multiplying them together, and then taking the IFFT of the product.

```
>>x1=[zeros(1,20) ones(1,40) zeros(1,20)];
>>x2=[zeros(1,20) linspace(0,1,20) linspace(1,0,20) zeros(1,20)];
>>X1=fft(x1); X2=fft(x2);
>>Y=X1.*X2;
>>y=real(ifft(Y));
```

The reason we have invoked 'real' in the last instruction is to get rid of residual imaginary values of infinitesimal magnitude. These values arise from the computational resolution of Matlab, and will typically be in the order of 10^{-16} or smaller.

4.3.2 Filtering

In the following code, we want to filter a sample of sound of duration 2 seconds using a bandstop filter that suppresses frequencies in the range from 1 kHz to 2 kHz. The sample rate of the sound signal is known to be 22,050 initially, but is converted to 10,000 Hz. Thus, the number of samples in the sound wave thus becomes 2×10,000 = 20,000 samples. The frequency base vector is thus 20,000 points long with a start value of -5,000 and an end value of 5,000.

The points corresponding to the frequencies from -2 kHz to -1 kHz and from 1 kHz to 2 kHz are the points number 6001 through 8000 and 12,001 through 14,000 (using ratio and proportion). In the following code, we will insert zeros in these locations in the FFT results.

```
>>y=wavread('File1');
>>y=resample(y,10000,22050);
>>y=y(1:20000);
>>Y=fftshift(fft(y));
>>Y([6001:8000 12001:14000])=0;
>>y2=real(ifft(Y));
>>sound(y2,10000);
```

Matlab Instructions for Lesson 4

1	t=linspace(0,5,1000);
2	y=0.5*cos(60*pi*t)+0.3*sin(80*pi*t);
3	Y=fftshift(fft(y));
4	Ymag=abs(Y);
5	Yphase=angle(Y);
6	Fvec=linspace(-100,100,1000);
7	figure; plot(Fvec,Ymag); title('Magnitude Spectrum');
8	figure; plot(Fvec,Yphase); title('Phase Spectrum');
9	y2=real(ifft(Y));
comments	
10	x1=[zeros(1,20) ones(1,40) zeros(1,20)];
11	x2=[zeros(1,20) linspace(0,1,20) linspace(1,0,20) zeros(1,20)];
12	X1=fft(x1); X2=fft(x2);
13	Y=X1.*X2;
14	y=real(ifft(Y));
comments	
15	y=wavread('File1');
16	y=resample(y,10000,22050);
17	y=y(1:20000);
18	Y=fftshift(fft(y));
19	Y([6001:8000 12001:14000])=0;
20	y2=real(ifft(Y));
21	sound(y2,10000);
comments	

%% convolution by FFT

```
>> x = [ 1 2 0 1];
>> h = [ 2 2 1 1];
>> y = ifft(fft([x zeros(1, 3)]).*fft([h zeros(1, 3)]));
>> stem(y, 'b--o')
>> hold on
>> y = conv(x, h);
>> stem(y, 'r-*')

%%
>> X = [zeros(1,20) ones(1,40) 5*ones(1,20)];
>> stem(X)
>> H1 = [ones(1,40) zeros(1,40)];
>> figure; stem(H1)
>> Y1 = X.*H1;
>> figure; stem(Y1)
>> H2 = [zeros(1,20) ones(1,60)];
>> Y2 = X.*H2;
>> figure; stem(Y2)
>> H3 = [zeros(1,20) ones(1,40) zeros(1,20)];
>> Y3 = X.*H3;
>>figure; stem(Y3)
```

For Practice

1- With $Y(Z) = 18X(Z) + 8Z^{-1}X(Z)$,find its transfer function ,impulse and step response with number of points =10 .

2- With sampling frequency=200 , and time $0 \leq t \leq 4$, find and sketch both the magnitude and phase of the spectrum of the signal.

$$0.5*\cos(40*pi*t) +0.3*\sin(60*pi*t)+ 0.3*\sin(80*pi*t)$$

And then transform the signal back to time domain

3- We have two signals x1 = [1 2 0 1] , x2 = [2 2 1 1]. find the convolution of the two signals using two different methods conv function and fft,ifft functions then compare the results.

4- A signal in frequency domain is defined as

x_f=(sinc(f/5000).*sinc(f/5000)) f is the frequency base vector

with sampling frequency fs=20k and number of samples Ns=100000 , we need to eliminate any frequency components less than -5khz or larger than 5khz.