

# ELE 301, Fall 2011

## Laboratory No. 4

### 1 Background

The objective of this lab is to explore the Fourier transform, learn how to use the `fft` and `ifft` commands and to construct a spectrogram.

#### 1.1 Fourier Transform for Finite Duration Signals

In order to analyze the frequency content of a finite duration discrete time signal  $x$  with  $N$  samples, we use the Discrete Fourier Transform (DFT):

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n) e^{-i \frac{2\pi k}{N} n}, \quad k = 0, \dots, N-1.$$

This can be interpreted as the Fourier Transform of the finite duration signal evaluated at the frequencies  $f = k/N$ . Another interpretation is that the DFT is the Fourier Series of the periodic extension of  $x$  but is missing the  $1/N$  scaling factor. This second interpretation gives rise to the Inverse DFT formula.

In order to go from  $\hat{x}$  back to  $x$ , we use the Inverse DFT:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}(k) e^{i \frac{2\pi k}{N} n}, \quad n = 0, \dots, N-1$$

These equations can be written in matrix form as

$$\begin{aligned} x &= \frac{1}{N} F \hat{x}, \\ \hat{x} &= \bar{F} x, \end{aligned}$$

where  $\bar{F}$  is the complex conjugate of  $F$ , and  $F$  is the  $n \times n$  Fourier matrix:

$$F = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{i \frac{2\pi}{N}} & e^{i \frac{4\pi}{N}} & \dots & e^{i 2\pi \frac{N-1}{N}} \\ 1 & e^{i \frac{4\pi}{N}} & e^{i \frac{8\pi}{N}} & \dots & e^{i 2\pi \frac{2(N-1)}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{i 2\pi \frac{N-1}{N}} & e^{i 2\pi \frac{2(N-1)}{N}} & \dots & e^{i 2\pi \frac{(N-1)^2}{N}} \end{bmatrix}.$$

We will often want to graphically depict the frequency content found in the DFT  $\hat{x}$ . Remember that  $\hat{x}$  is in general a complex valued vector even if  $x$  is real valued, so it is usual to plot both the magnitude  $|\hat{x}(k)|$  and phase  $\arg(\hat{x}(k))$  on separate graphs.

The elements in the  $\hat{x}$  vector are coefficients for the frequencies  $f = 0, 1/N, 2/N, \dots, (N-1)/N$ . However, notice that the high frequencies are equivalent to low negative frequencies due to aliasing. For example,  $f = (N-1)/N$  is equivalent to  $f = -1/N$ . It is common to rearrange the vector  $\hat{x}$  to represent the frequency components in the range  $[-1/2, 1/2]$  rather than  $[0, 1]$ . Please read about the Matlab function `fftshift` which is used for this purpose.

## 1.2 MATLAB: fft, ifft and fftshift

To calculate the DFT of a function in MATLAB, use the function `fft`. FFT stands for Fast Fourier Transform, which is a family of algorithms for computing the DFT. A straight computation of the DFT from the formulas above would take  $n^2$  complex multiplications and  $n(n-1)$  complex additions. Algorithms have been developed (for example, the Cooley-Tukey FFT) which exploit symmetries of the roots of unity to reduce the number of calculations to  $O(n \log_2 n)$ . It is fastest when  $n$  is a power of 2, followed by composite numbers of small primes. Read the MATLAB help page for more information.

To take the DFT of a vector  $x$  of length  $n$ , use the command:

```
hx=fft(x);
```

If you want to explicitly specify the length  $m$ , then use:

```
hx=fft(x,m);
```

This will append the vector with zeros if  $m$  is greater than the length of  $x$  and it will truncate  $x$  if  $m$  is less than the length of  $x$ . To compute the IDFT of a vector  $hx$  use:

```
x=ifft(hx);
```

The command `fftshift` swaps the first and the second half of a vector. This is used before plotting `hx` to ensure the frequency range is in  $[-1/2, 1/2]$ . To plot the magnitude versus frequency with zero frequency centered on the axis, you could use the commands (assuming  $N$  is even):

```
hx=fft(x);
shx=fftshift(hx);
f=[-N/2:N/2-1]/N;
figure(1)
stem(f,abs(shx),'r')
xlabel('Frequency in [-1/2,1/2]')
ylabel('Magnitude of DFT(x)')
axis([-1/2 1/2 0 inf]);
grid
```

## 1.3 Sampled Signals

Suppose you begin with a continuous time signal  $x(t)$  defined over a finite time period  $[0, T]$  and then you construct a discrete-time signal  $x[n]$  by taking uniformly spaced samples of  $x(t)$ . If  $N$  uniformly spaced samples of  $x$  are taken, then the sampling frequency is  $F_s = N/T$  (the sampling frequency would technically be  $(N-1)/T$  if the first sample is  $x(0)$  and the last sample is  $x(T)$ ).

We can take the DFT of the discrete-time signal  $x[n]$  and use this to represent the frequency content of the continuous-time signal  $x(t)$ . This may not be valid, but let's go with it anyway. You'll learn more about this in the lecture on sampling. If we represent the discrete-time frequency spectrum on the range  $[-1/2, 1/2]$ , then we translate this to continuous-time frequencies by multiplying by  $F_s$ . Thus, given a sampled signal with sampling frequency  $F_s$ , we plot its DFT magnitude and phase versus frequencies in Hz in the range  $[-F_s/2, F_s/2]$ .

## 1.4 The Spectrogram

Let  $x$  be signal of length  $N$ . Consider consecutive segments (or "clips") of  $x$  of length  $m$  where  $m \ll n$  and let  $X \in \mathbb{R}^{m \times (N-m+1)}$  be the matrix with the consecutive segments as consecutive columns. In other words,  $[x[0], x[1], \dots, x[m-1]]^T$  is the first column,  $[x[1], x[2], \dots, x[m]]^T$  is the second column, and so forth. Both the rows and columns of  $X$  are indexed by time. We see that  $X$  is a highly redundant representation of  $x$ .

The **spectrogram** of  $x$  with window size  $m$  is the matrix  $\hat{X}$  whose columns are the DFT of the columns of  $X$ . So

$$\begin{aligned}\hat{X} &= \overline{F}X \\ X &= \frac{1}{m}F\hat{X}\end{aligned}$$

Note that the rows of  $\hat{X}$  are indexed by frequency and the columns are indexed by time. Each location on  $\hat{X}$  corresponds to a point in frequency and time. So  $\hat{X}$  is a mixed time-frequency representation of  $x$ . Because the transformation between  $X$  and  $\hat{X}$  is invertible,  $\hat{X}$  is also highly redundant.

The spectrogram is a matrix. To visualize it we can view the matrix as an image with the  $i, j$ -th entry in the matrix corresponding to the intensity or color of the  $i, j$ -th pixel in the image. In MATLAB this is done by calling the function **imagesc**. This first scales the image to the full range of the color map and then displays it as an image. The  $x$ -axis is the time axis and the  $y$ -axis is the frequency axis. You can change the color map with the command **colormap**. Since the signal is real, the first half and the second half of each column are redundant by a symmetry property of the DFT. Therefore, when you display  $\hat{X}$ , you only need to show the first half of each column. Also, by default **imagesc** plots from the top left corner, and we will want to see it upside down, so you use the following commands (assuming **HX** is your spectrogram):

```
ax=imagesc(t,f,HX); % t is the time vector for the x-axis
set(ax,'YDir','normal'); % and f is the frequency vector for the y-axis
```

Given that all of the information in  $\hat{X}$  is already in  $x$  what is its value? We are interested in analyzing signals whose frequency content changes in time (e.g. music) and we want to analyze how this occurs. Taking the DFT of the entire signal will show us the frequency content over the entire time period. We would like to take the DFT over a short period of time because this will give us a local snapshot in time of the frequency content of the signal during that short time period. We can do this by first taking the DFT of  $x(1:m)$ , then we can take the DFT of  $x(2:m+1)$ , and so on until we hit the end of the signal. This is exactly the spectrogram.

## 1.5 DTMF

On a touch tone phone, there is a frequency associated with each column of the keypad and there is a frequency associated with each row of the keypad. When a button is pressed, the signal transmitted over the telephone line is the sum of two sinusoids, one with the column frequency and one with the row frequency. This system is called a Dual Tone Multi-Frequency (DTMF) system. The frequencies for the columns are 1209, 1336, and 1477 Hz from left to right, with 1633 Hz associated with a fourth column that is usually not there. The row frequencies are 697, 770, 852, and 941 Hz, from top to bottom. The standard requires that each ‘tone’ must be at least 40 msec long and that the space between tones must be 40 msec. The encoding is shown in Figure 1.

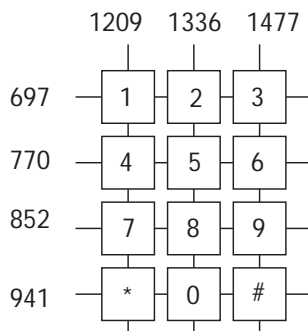


Figure 1: Telephone keypad and the row and column frequencies.

In this lab you will decode a DTMF signal. To find which buttons were pressed and the order in which they were pressed, the spectrogram might be a useful representation to explore. (In practice we would not need to compute the whole spectrogram, only at those specified frequencies).

## 2 Lab Procedure

### 2.1 FFT

First, we will get some practice using the MATLAB `fft` command.

### 2.2 Simple vector signals

Make an m-file `ffttest.m` to do the following. Set  $n = 64$ ,  $\Omega_1 = 10(2\pi/n)$  and  $\Omega_2 = 14(2\pi/n)$ . Then generate the vector signal

$$x(k) = \cos(\Omega_1 k) + 0.5 * \cos(\Omega_2 k)$$

for  $k = 0, \dots, n-1$ . Take the DFT of  $x$  and plot the DFT versus frequencies in the range  $[-\pi, \pi]$ . The zero frequency should be centered in the plot (use `fftshift`). Include vertical dotted lines on the plot at the frequencies  $\pm\Omega_1$  and  $\pm\Omega_2$ . Does the plot of the DFT agree with what you expected?

#### 2.2.1 Simple Sampled Signals

Make an m-file `ftsamptest.m` to do the following. First, generate a sine wave signal  $\mathbf{x}$  with frequency 400Hz sampled at 8kHz for  $n = 1000$  samples. Then take a 1000 point FFT of  $\mathbf{x}$  to produce  $\mathbf{hx}$  and apply `fftshift` to  $\mathbf{hx}$  to swap the first and second halves so that the zero frequency is in the middle. Generate a vector `fHz` of frequencies in Hz corresponding to each point in  $\mathbf{hx}$ . Plot a stem plot of the absolute value of the `fftshift`-ed  $\mathbf{hx}$  vector against `fHz` using the command `stem`. Is this what you expected?

Now do the same thing for the integral frequencies 401Hz and 402Hz and so on. What about 440Hz? What is happening? What is special about 400Hz and 440Hz? (it might help to plot the sine signal). Try changing the number of points in the FFT, how does this affect the plot?

Finally, compute and plot the DFT of this signal using 1024, 2024, 4096 samples at 8kHz:

$$x(t) = \sin(2\pi 400t) + 0.5 \sin(2\pi 420t) + 0.25 \sin(2\pi 440t)$$

What does this tell us about how to sample a signal and interpret the DFT?

### 2.3 Removing Unwanted Tones

You have been provided with a sound file, `buzzjc.wav`, with an audio clip of someone saying something, but it has been corrupted by adding a buzzer signal consisting of a few tones. First, load the signal into MATLAB using the command `wavread` which you can read more about in the MATLAB help file. Use the DFT to figure out the tone frequencies (at least approximately). List the tone frequencies here:

Next remove these tones (how?) and then listen to the clip (note: you can only play a real signal, so make sure you only take the real part or imaginary part) and play for your TA. Write down what the person in the clip is saying:

#### 2.3.1 Telephone dialing signal

You have been provided with a .wav file, `tel.wav`, with a telephone number encoded using the DTMF scheme described in the background section. Your objective is to decode the phone number.

Plot the magnitude and phase of the DFT of the signal versus frequency in Hz. Include vertical dotted lines at each of the DTMF column and row frequencies on the plot. How much information can you determine from the plots. It must be possible to decode the number from the signal and hence from its DFT (why?). Can you decode the number dialed from the DFT?

## 2.4 The Spectrogram

### 2.4.1 Spectrogram of telephone dialing signal

Using the telephone dialing signal, construct a matrix **X** with clips of length 205 in the columns, where the first clip is comprised of samples 1 through 205, the second is 2 through 206 and so on. Next construct the matrix **Y** in which the  $j$ -th column is the FFT of the  $j$ -th column of **X**. This matrix is the spectrogram of the signal. Visualize this using the command `image` or `imagesc`. Display this for your TA.

Can you use the spectrogram to decode the number dialed? If so write your answer here:

### 2.4.2 Spectrogram of Music and Speech

Now load the music file into MATLAB and display the spectrogram of the signal. Look at the beginning of the song when there are no instruments and then later when there are, can you tell this from the spectrogram? Note any other facts you pick up. Experiment with different clip lengths - do one short one and one longer one. What are the advantages and disadvantages of using short clips vs. using longer clips?

Next, construct the spectrogram with clips of 50% overlap and no overlap and write down how changing the overlap affects the spectrogram:

Remember, please print out all code and a few relevant plots.