MAE 598: Digital Signal Analysis

Anushka Subedi

September 2023

Step 1:

Code for Step 1:

```
clc;
clear;
Fs=8000;                % Sampling frequency = 8000 Hz
N=2*Fs;                 % Signal duration = 2s, N is number of samples for a 2-second tone
k=0:(N-1);              % Time vector
f=440;                  % Signal frequency = 440Hz
s=cos(2*pi*f*k/Fs);


sound(s,Fs);

%Changing the Volume
pause(5)
sound(s/5,Fs);
pause(5)
sound(5*s,Fs);
```

Step 2:

Code for Step 2:

```
clc;clear;
Fs=8000;                     % Sampling frequency = 8000 Hz
N=2*Fs;                      % Signal duration = 2s, N is number of samples for a 2-second tone
k=0:(N-1);                   % Time vector
f=[1000:1000:8000];          % Signal frequency = 440Hz


for i =1:8
    s=cos(2*pi*f(i)*k/Fs);
    sound(s,Fs);
    pause(3)
end
```

The sampling frequency here is Fs=8000Hz, the Nyquist frequency is, therefore, Fs/2=4000Hz. This is the maximum frequency below which aliasing won't occur. Here, the sound goes on increasing until 4000Hz and then goes on decreasing. What's happening here is that the sound after 4000Hz are aliased and they are incorporated someplace within 4000Hz, making them louder.

Step 3:

Code for Step 3:

```
clc;clear;
Fs=16000;                    % Sampling frequency = 8000 Hz
N=2*Fs;                      % Signal duration = 2s, N is number of samples for a 2-second tone
k=0:(N-1);                   % Time vector
f=[1000:1000:8000];


for i =1:8
    s=cos(2*pi*f(i)*k/Fs);
    sound(s,Fs);
    pause(5)
end
```

The sampling frequency here is Fs=16000Hz, the Nyquist frequency is, therefore, Fs/2=8000Hz. This is the maximum frequency below which aliasing won't occur. Here, the sound goes on increasing until 8000Hz and then goes on decreasing, but because the last frequency is 8000 we don't hear the decreasing sound patterns. What's happening here is that the sound after 8000Hz are aliased and they are incorporated someplace within 8000Hz, making them louder, but since there is no frequencies after 8000Hz, we can't hear those decreasing pattern.

Step 4:

We hear sound at both 4000Hz for Fs=8000Hz and 8000Hz for Fs=16000Hz. This is because the given wave is a cosine wave. Now, had the wave been sine, we wouldn't hear the sound at exactly the Nyquist frequency because all the sampling points would end up at zero forming a straight line. (Explained beautifully at 7:53 of this video: https://youtu.be/yWqrx08UeUs?feature=shared.) What we can conclude here is that we don't always hear the sound at Nyquist frequency, the requirement for these waves is that we set Sampling frequency to be a little higher than 2*Nyquist Frequency.

Code for Step 4:

```
%Problem4a
clc;clear;
Fs=8000;                               % Sampling frequency = 8000 Hz
N=2*Fs;                                % Signal duration = 2s, N is number of samples for a 2-
    second tone
k=0:(N-1);                             % Time vector
f=[900, 1800, 2700, 3600, 4500, 5400, 6300, 7200];


for i =1:8
    s=cos(2*pi*f(i)*k/Fs);
    sound(s,Fs);
    pause(4)
end

%Problem4b
clc;clear;
```

```
Fs=16000;                              % Sampling frequency = 8000 Hz
N=2*Fs;                                % Signal duration = 2s, N is number of samples for a 2-second tone
k=0:(N-1);                             % Time vector
f=[900, 1800, 2700, 3600, 4500, 5400, 6300, 7200];


for i =1:8
    s=cos(2*pi*f(i)*k/Fs);
    sound(s,Fs);
    pause(2)
end
```

We hear sound at all the frequencies for both the cases because:

1. For Fs=8000Hz, all the values lie below 8000Hz, the values until 3600Hz give increasing sound, there is no boundary of 4000Hz where we were supposed to hear no sound in sine waves, etc., the sound then is of lower sound until 7200Hz.

2. For Fs=16000Hz, all the values lie below 8000Hz, the values until 7200Hz give increasing sound, there is no boundary of 8000Hz, where we were supposed to hear no sound in sine waves, etc.,

The reason for both, again, as explained above is that the frequency values are lower than the Nyquist frequency.

Step 5:

Here, the sound is louder when two signals are added than when half of second signal and one times of the first signal are added. This is because the first case has a higher amplitude than the second case.

Code for Step 5:

```
clear;
clc;

Fs=8000;                % Sampling frequency = 8000 Hz
N=2*Fs;                 % Signal duration = 2s, N is number of samples for a 2-second tone
k=0:(N-1);              % Time vector
f2=440;
f1=200;

s1=cos(2*pi*f1*k/Fs);
s2=cos(2*pi*f2*k/Fs);

sound((s1+s2),Fs);
pause(5)
sound((s1+(0.5.*s2)),Fs);
```

Step 6:

Code for Step 6:

```
clc;
clear;
load handel.mat;
audio=y;

```

```
sound(audio)
Fs                  %Checking the value of Fs
```

I hear Handel's excerpt. It doesn't sound strange with default Fs=8192Hz. It sounds strange with other values of Fs with it being slower with lower Fs values and faster with higher Fs values.

The default Fs is 8192Hz, so it's not strange.

Step 7:

The audio vector is an array of the pressure signal. The plot is attached here. It looks different than the plot in the slide, because the time of Handle's excerpt here is 10 seconds and the one in the slide is plotted for 0.01s.

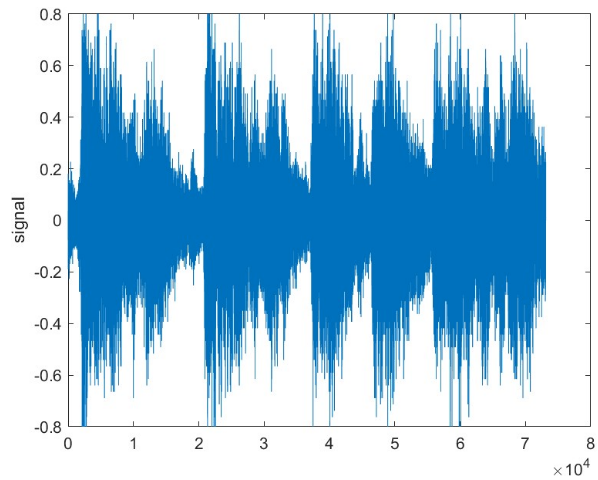


Figure 1: audio vector plotted

The plot after adjusting the length is attached here. It looks similar to the plot in the slide now.

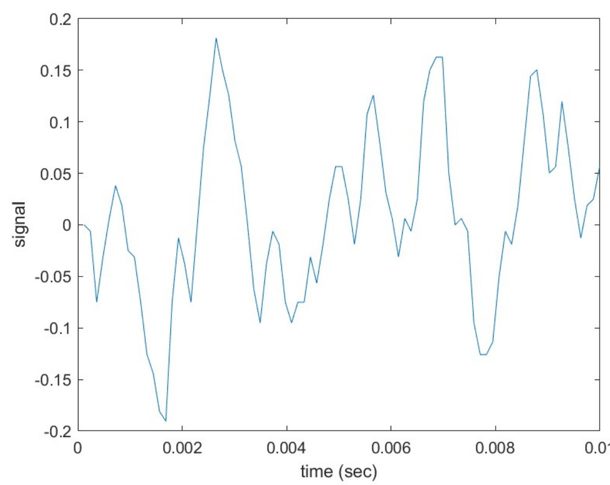


Figure 2: adjusted audio vector

Code for Step 7:

```
clc;
clear;
load handel.mat;
audio=y;

sound(audio)
figure(1)
plot(audio)
ylabel('signal')
saveas(figure(1),'figure7_1','jpg');

onesecond=8192;          %for one second number of samples is 8192
slidetime=0.01.*onesecond;
%rounding off up to 83 to match the slide
slidetime=83;

for i =1:slidetime
    audio_slide(i)=audio(i);
end

sound(audio_slide)
y=(0.01/83):0.0001204:0.01;  %to match the time=0.01s in slide
figure(2)
plot(y,audio_slide)
xlabel('time (sec)')
ylabel('signal')
saveas(figure(2),'figure7_2','jpg');
```

Step 8:

We just calculated the Fourier coefficients of the signal using FFT function. The plot is attached here.

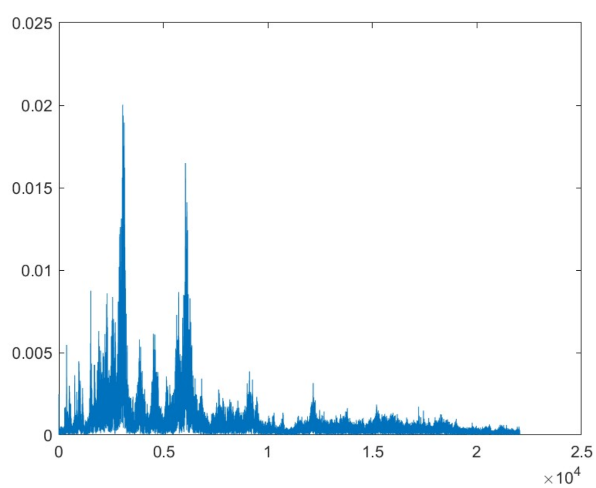


Figure 3: fourier coefficients

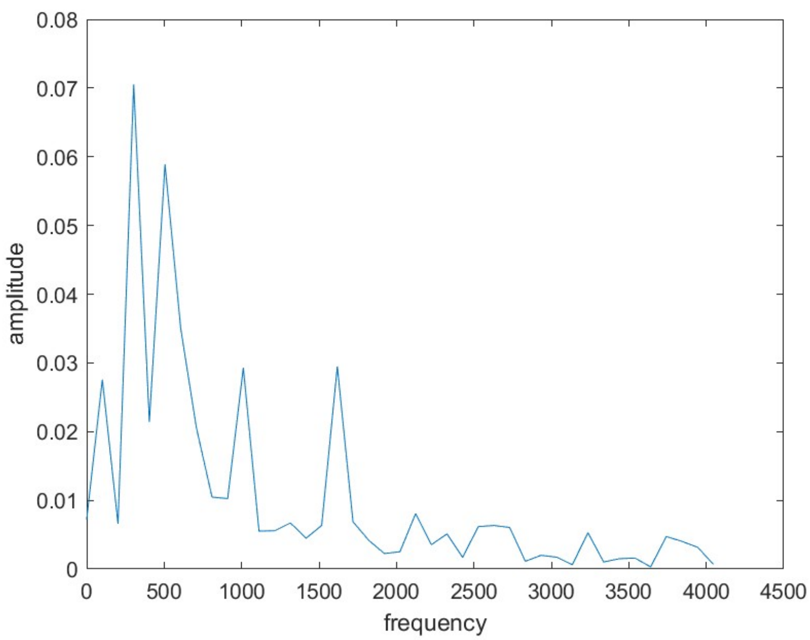
Code for Step 8a:

```
clc;
clear;

load handel.mat;
audio=y;
Fs=44100;
N=length(audio);
f = Fs*(0:(N/2))/N;


% Calculate spectrum
spectrum=fft(audio);
P2 = abs(spectrum/N);
P1 = P2(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);
plot(f,P1);
saveas(figure(1),'figure8a','jpg');
```

After adjusting the length to match the lecture slides, we get the following plots.



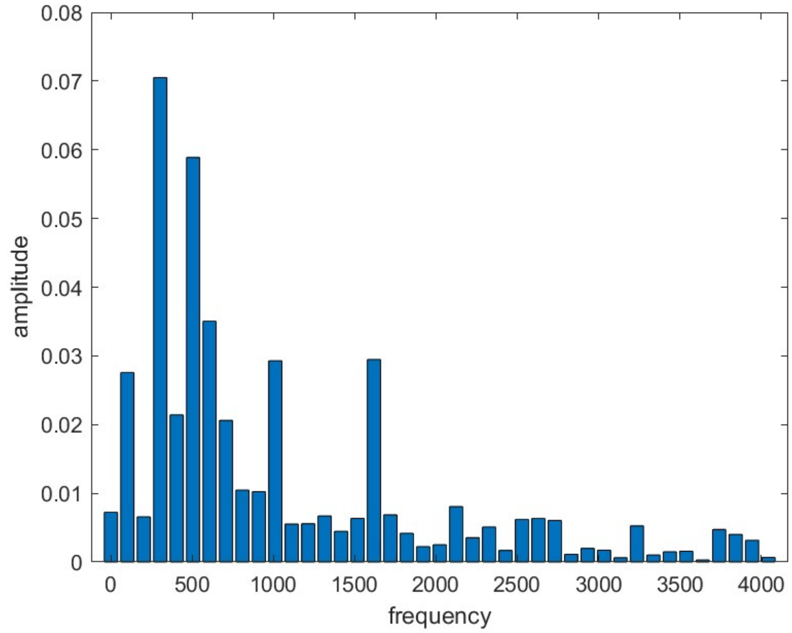


Figure 4: bar plot to match the slide

Code for Step 8b:

```
clc;
clear;

load handel.mat;
audio=y;
Fs=8192;
onesecond=8192;
slidetime=floor(0.01.*onesecond);

for i =1:slidetime
    audio_slide(i)=audio(i);
end

N=length(audio_slide);
f = Fs*(0:(N/2))/N;

spectrum=fft(audio_slide);
P2 = abs(spectrum/N);
P1 = P2(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);
figure(1)
plot(f,P1);
xlabel('frequency')
ylabel('amplitude')
saveas(figure(1),'figure8b_1','jpg');
figure(2)
bar(f,P1);
xlabel('frequency')
```

```
ylabel('amplitude')
saveas(figure(2),'figure8b_2','jpg');
```

Code for Step 9:

```

clc;
clear;
load handel.mat;
audio=y;
Fs=8192;    %it is default but is also the correct sampling frequency

half =36557;    %73K/2
last=73113;

audio_firsthalf = audio(1:half);
audio_secondhalf= audio(half:last);
sound(audio_firsthalf+audio_secondhalf);
```