2020 ASAS Homework 1: Basic Audio Signal Manipulations and Fourier Transform

Prof. Yi-Wen Liu

March 5, 2020

1.

1. In MATLAB, create an array of length N=16000, and fill the array with the following values: 𝑦[𝑛] = 0.5𝑔[𝑛 − 8000] sin(2𝜋𝑓0𝑛𝑇), where T = 1/16000 sec, n = 0:15999, 𝑓0 (Hz) is a frequency of your choice, and

cos2 (𝜋 𝑛 ) , −𝑀 ≤ 𝑛 ≤ 𝑀

𝑔[𝑛] = {

2 𝑀

0, elsewhere

So 𝑔[𝑛 − 8000] should be a bell-shaped function centered around time n=8000 and its width is 2𝑀 samples.

See program 1.1.c

1. Set the sampling rate 𝑓𝑠 = 16000, M=8000, and play y[n]. By appropriately choosing 𝑓0, you should hear a pip tone. Listen to it and record the range of 𝑓0 that you can hear something. #discussion

From ~20hz– 7999.9hz the pip tone is audible. Above is the Nyquist cut-off, and below is approximately the human hearing threshold cut-off.

1. Now, set 𝑓0 = 1000 Hz. Shorten M and record the range of M at which you can still hear a definite pitch.

The pip tone becomes more of an impulse around 1/1000 of the sample rate which can be heard all the way through 1/16000 of the sample rate.

1. Change sin(2𝜋𝑓0𝑛𝑇) to cos(2𝜋𝑓0𝑛𝑇) . Can you hear any difference? This is essentially asking the question of whether the *phase* of an audio signal is important. #discussion

, despite *f*0 not being changed.

(e) Change the definition of 𝑔[𝑛] to 𝑔[𝑛] = 1, −𝑀 ≤ 𝑛 ≤ 𝑀 , and 𝑔[𝑛] = 0

elsewhere. Can you hear a difference in 𝑦[𝑛]? #discussion

The pip tone has no envelope when we set g = 1, i.e. it stays at 1.

1. The discrete Fourier transform is defined as follows,

𝑁−1

𝑌(𝑘) = ∑ 𝑦[𝑛]𝑒−𝑗𝑘(2𝜋)𝑛/𝑁 ,

𝑛=0

where 𝑘 = 0,1,2, … , 𝑁 − 1.

1. Create an array of length N=16000 and calculate 𝑌(𝑘) for all 𝑘. Your result should be identical as MATLAB’s fast Fourier transform (FFT) function: Y = fft(y); but here the purpose is to verify that indeed this is the case. If so, you can feel free to use FFT from now on.

See program 1.2.c

1. Plot |𝑌(𝑘)| against frequency

𝑘

𝑓 = 𝑁 𝑓𝑠.

The result is called the magnitude

spectrum of the signal. Notes: Sometimes, we prefer to view |𝑌(𝑘)| in dB by taking 20 log10 |𝑌(𝑘)|.

See dft1000.svg. The magnitude, though, is affected by the window, unless I am confused. Therefore, the result does not show 0.5 magnitude as it would had we not applied a window.

1. Adjusting 𝑀 and 𝑓0 and observe how changes occur in the magnitude spectrum.

As expected, the frequency center of the magnitude shifts with changes in *f*o and the magnitude increases and shrinks with larger and smaller M, respectively.

1. The Short time *F*ourier transform 𝑌(𝑛, 𝜔) is defined as

∞

𝑌(𝑛, 𝜔) = ∑ 𝑦[𝑚]𝑤[𝑚 − 𝑛] 𝑒−𝑗𝜔𝑛,

𝑚=−∞

where 𝑤[𝑛] is a (usually positively-valued) *window function*.

1. Choose an appropriate window function w[n] and calculate 𝑌(𝑛, 𝜔) for 𝜔 ∈ [0, 𝜋]. Verify whether your implementation produces the same result as MATLAB’s spectrogram() function.

See program 1.3.c

1. In the audio field, it is a common practice to view the magnitude of STFT |𝑌(𝑛, 𝜔)|. Read some sound files into MATLAB by audioread(), and view the spectrogram to see if you can make sense of it. Are there any patterns that correspond to audible properties of the sound?
2. Is it possible to reconstruct the signal y[n] from the magnitude of STFT, |𝑌(𝑛, 𝜔)|? Why or why not? Note that by taking the absolute value of complex numbers we have ignored the phase information. #discussion

*Preview* of the next homework: We will *filter* audio signals in the sense that each frequency receives a different gain (and delay).