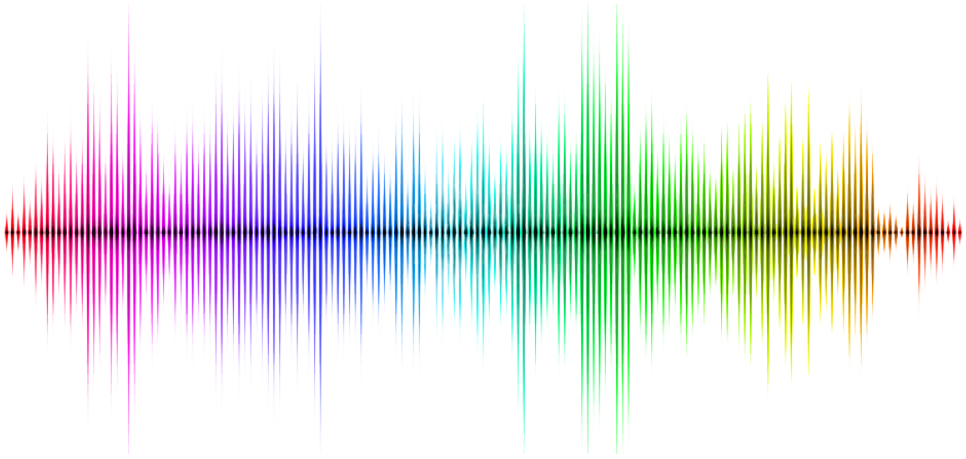
SIGNALS AND SYSTEMS

GYEDU, DENNIS OPPONG

12th April 2023



Home Work 4

HOME WORK 4

Task 1a

0.1 Why are audio files usually sampled at 44.1kHz?

Audio files are often sampled at 44.1kHz since that is the sampling rate used by the Compact Disc Digital Audio (CDDA) standard, which was created in the early 1980s. This sample rate was chosen because it enables for high fidelity sound reproduction while remaining compliant with the storage limits of CDs. The analog sound wave is sampled 44,100 times per second at 44.1kHz, with each sample represented by a 16-bit digital value. This enables for a frequency response range of up to 20kHz, which is the highest limit of human hearing.

While alternative sampling rates are employed in audio production, such as 48kHz or 96kHz, 44.1kHz remains the most often used sample rate for consumer music due to its compatibility with the CD format and ability to offer high-quality sound without requiring significant storage capacity.

0.2 For a signal of bandwidth fbw and sampling rate fs, what is the maximum carrier frequency to which one could upconvert?

The maximum frequency that can be accurately represented in a digital signal, according to the Nyquist-Shannon sampling theorem, is half of the sampling frequency. That is, the maximum carrier frequency to which a signal can be upconverted is half the sampling rate.

So, given a signal bandwidth of fbw and a sampling rate of fs, the maximum carrier frequency that can be upconverted is:

$$f_{c\text{-}max} = \frac{f_s}{2} - f_{bw} \tag{1}$$

Where fc max denotes the maximum carrier frequency, fs the sampling rate, and fbw the signal bandwidth.

0.3 What happens if you go beyond this carrier frequency?

Aliasing may occur in the upconverted signal if the carrier frequency exceeds the maximum frequency limit specified by the Nyquist-Shannon sampling theorem.

The aliasing effect can be viewed as frequency component overlap caused by undersampling, resulting in a distorted representation of the actual signal. As a result, frequency components that were not there in the original signal are frequently present.

Task 1b

The mathematical relationship, in the time domain, between the baseband signal y_i[n] and the real passband signal $\hat{y}_i$[n] is demostrated as follows.

Taking the inverse DFT on both sides, we get:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{jk\frac{2\pi}{N}n} \tag{2}$$

Provided the signal

$$y_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{jk\frac{2\pi}{N}n} \tag{3}$$

Where N is the finite Length of the signal $y_i[n]$ and $Y_i[k]$ is the DFT coefficient of the signal$y_i[n]$

Also, considering

$$Y_i^u(f) = Y_i(f - f_{ci}) \tag{4}$$

$$\tag{5}$$

The above equation can be rewritten to produce

$$Y_i(f - f_{ci}) \approx Y_i(k - k_c) \tag{6}$$

The shift in the Kth term of the DFT coefficient is as a result of the shift the frequency to the right, we now hve

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k - k_c] e^{jk\frac{2\pi}{N}n} \tag{7}$$

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{j(k+k_c)\frac{2\pi}{N}n} \tag{8}$$

Where $\hat{y}_i[n]$ is the upconverted signal of the original $y_i[n]$

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{j(k)\frac{2\pi}{N}n} e^{jk_c\frac{2\pi}{N}n} \tag{9}$$

From the above calculations and manipulations, we have;

$$\frac{1}{N}\sum_{k=0}^{N-1} Y_i[k]e^{jk\frac{2\pi}{N}n} = y_i[n] \tag{10}$$

Therefore, we have;

$$\hat{y}_i[n] = y_i[n]e^{jk_c\frac{2\pi}{N}n} \tag{11}$$

Knowing that,

$$\Omega_{kc} = \frac{2\pi k_c}{N} = 2\pi f_c T_s = \frac{2\pi f_c}{f_s} \tag{12}$$

f_c represent the carrier frequency. Hence, the relations is established in equation (13)

$$\hat{y}_i[n] = y_i[n]e^{jf_c\frac{2\pi}{f_s}n} \tag{13}$$

Task 1 c

Using the 'audio read' function, read in the original audio signal "champions.wav," which returns both the audio data and the sampling rate 'Fs'. The variable 'index' is then set to 0, indicating the initial index of the signal to be upconverted. The variable 'L' is set to the length of the audio stream, which will be used to calculate the ending index of the upconverted signal.

We utilize the 'upconverter' function with the input arguments 'index,' 'L,' 'Fs,' and 'F_bw,' which is set to 3.7 kHz, the bandwidth frequency, to upconvert the signal. This function produces a complicated sinusoidal wave of length 'L' that we multiply element by element with the original audio stream 'audio'. The upconverted signal 'y up' has the same length as the original audio signal 'audio'.

After running the code, we see that the upconverted signal 'y_up' has been produced. We have moved the original signal to a higher frequency range by raising the bandwidth frequency. This is seen in the frequency domain, where the original audio wave would contain most of its energy within the frequency range of human hearing, around 20 Hz to 20 kHz. The upconverted signal, on the other hand, would have the majority of its energy in the frequency range of 3.7 kHz to 23.7 kHz. In the temporal domain, the upconverted signal would be compressed as well, with a shorter period and higher frequency than the original audio signal.

Task 1d

This step involves reading three audio files and assigning them to the variables audio1, audio2, and audio3. The length of the third audio file, audio3, is then computed and assigned to the variable L. F_bw is given a bandwidth frequency of 3.7kHz.

The max() method determines the length of the longest audio file, and all audio signals are zero-padded to match the length of the longest signal. The zero-padded signals are then saved in the variables y1_padded, y2_padded, and y3_padded.

The zero-padded signals are then merged into a list called y list, and the mux() function is used to multiplex the signals in the list into a single signal, which is then saved in the mux signal variable.

Task 1e

Ultimately, this code sample provided in task 1e in the notebook demonstrates how to finish multiplexing by retrieving the final multiplexed signal and storing it as a WAV file.

Task 2a

Rewriting the relationship derived in Task 1(b) and implement the downconverter function that reverses the Upconversion.

We can take the inverse DFT of both sides to obtain

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{jk\frac{2\pi}{N}n} \tag{14}$$

Given the signal;

$$y_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{jk\frac{2\pi}{N}n} \tag{15}$$

Where N represents the finite Length of the signal $y_i[n]$ and $Y_i[k]$ represents the DFT coefficient of the signal$y_i[n]$

$$Given,\ Y_i^u(f) = Y_i(f - f_{ci}) \tag{16}$$

$$\tag{17}$$

Rewriting becomes;

$$Y_i(f + f_{ci}) \approx Y_i(k + k_c) \tag{18}$$

Shift in kth DFT coefficient produce right shift in frequency, going forward we have,

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k + k_c] e^{jk\frac{2\pi}{N}n} \tag{19}$$

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{j(k-k_c)\frac{2\pi}{N}n} \tag{20}$$

Where $\hat{y}_i[n]$ represents the up-converted signal of the original $y_i[n]$

$$\hat{y}_i[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{jk\frac{2\pi}{N}n} e^{-jk_c\frac{2\pi}{N}n} \tag{21}$$

considering the above calculations and manipulations, we have;

$$\frac{1}{N} \sum_{k=0}^{N-1} Y_i[k] e^{jk\frac{2\pi}{N}n} = y_i[n] \tag{22}$$

Therefore,

$$\hat{y}_i[n] = y_i[n] e^{-jk_c\frac{2\pi}{N}n} \tag{23}$$

Knowing that,

$$\Omega_{kc} = \frac{2\pi k_c}{N} = 2\pi f_c T_s = \frac{2\pi f_c}{f_s} \tag{24}$$

Where f_c represents the carrier frequency. Finally, we have;

$$\hat{y}_i[n] = y_i[n] e^{-jf_c\frac{2\pi}{f_s}n} \tag{25}$$

Task 2b

With a bandwidth frequency of 3.7 kHz, the function extracts two channels (indices 1 and 2) from the multiplexed signal. The demultiplexed signal that results is saved in the variable demux filtered.

Task 2c

The task 2c code is used to extract and store individual songs from the demultiplexed signal.

First, the demultiplexed signal is allocated to three variables that represent the three audio channels: Song1 1, Song2 2, and Song3 3. The first and second songs are then trimmed to the length of the original audio file using the [:661872] notation and stored in the "saved track" folder with the audio save() method. The third song is kept without being spliced.

The saved files are labeled "demux1 1.wav," "demux2 2.wav," and "demux3 3.wav," in that order.