EE386 Digital Signal Processing Lab **Jul-Dec 2021**

Experiment 1

Author: Prasad Fidelis D'sa *Email:* prasadfidelisdsa.191mt032@nitk.edu.in

1 Introduction

The exercise is geared as an introduction to Digital Signal Processing using Python. The problems involve Generation and plotting of signals and their Fourier Transform analytically, generation and sampling of tones using the sampling theorem, convolution using the convolution function from SciPy in various different modes, writing to and reading `.wav` files, amplitude modulation and calculation of its Fourier Transform using the efficient `fft` algorithm.

2 Problems

1. Generating Signals

Create a user-defined function $x(t)$ to generate a decaying exponential with α as the time constant. Plot the following in different subplots:

- Plot the signal $x(t)$ for $t \in [0; 10]$.
- Plot the signal $x(t - 1.5\alpha)$ for $t \in [0; 10]$.
- Plot the signal $x(2t)$ for $t \in [0; 10]$.

Compute the Fourier transform $X(\Omega)$ of the signal $x(t)$ analytically. Plot the magnitude and phase spectra of the signal. For delayed and dilated signals, write its Fourier transform as a function of $X(\Omega)$.

Algorithms/Methods

- The Fourier transform is an integral transformation of $f(t)$ from the time domain to the frequency domain.
  For our signal $x(t) = e^{\frac{-t}{\alpha}}$ the Fourier transform is calculated analytically as

$$F(\Omega) = \frac{1}{\frac{1}{\alpha} + j\Omega} \tag{2.1}$$

- To calculate the Fourier transform of the delayed signal $x(t - 1.5\alpha)$ we utilise the time shift property of the Fourier transform

$$\mathcal{F}[f(t-a)] = e^{-j\Omega a}F(\Omega) \tag{2.2}$$

Therefore

$$\mathcal{F}[x(t-1.5\alpha)] = e^{-j\Omega 1.5\alpha}F(\Omega) \tag{2.3}$$

Where $F(\Omega)$ is defined in Eq. (2.1)

- To calculate the Fourier transform of the dilated signal $x(2t)$ we utilise the scaling property of the Fourier transform

$$\mathcal{F}[f(at)] = \frac{1}{|a|}F(\frac{\Omega}{a}) \tag{2.4}$$

Therefore

$$\mathcal{F}[x(2t)] = \frac{1}{2}\frac{1}{\frac{1}{\alpha} + j\frac{\Omega}{2}} \tag{2.5}$$

Results

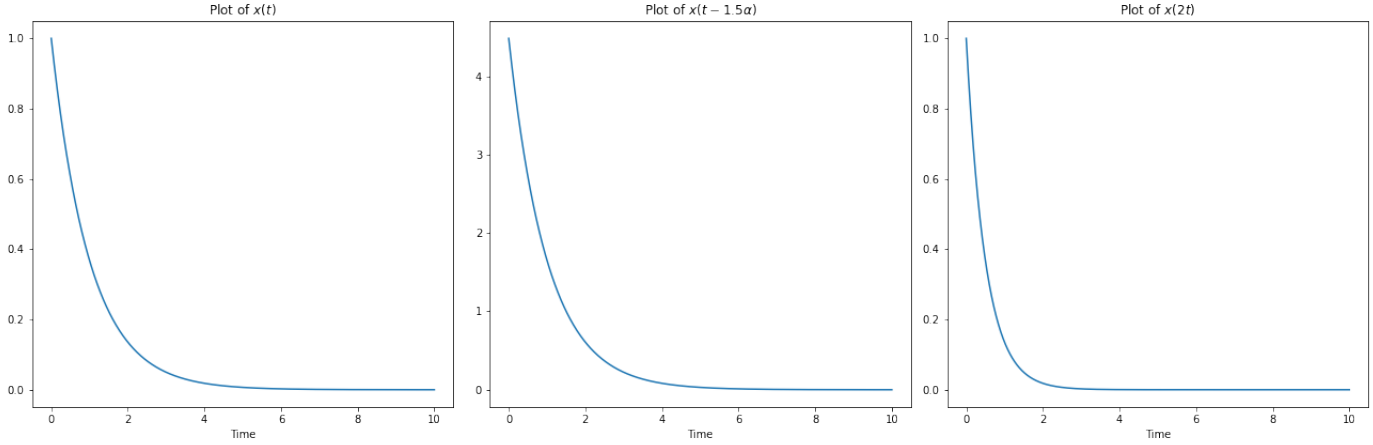

Figure 1: Subplots corresponding to the required signals

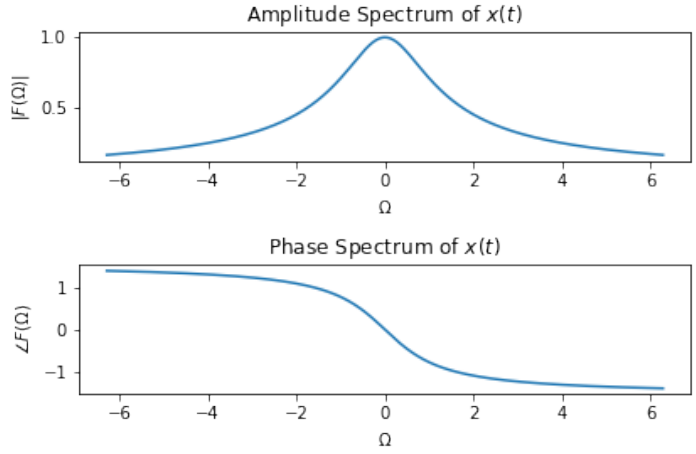

Figure 2: The Fourier Spectrum of $x(t)$

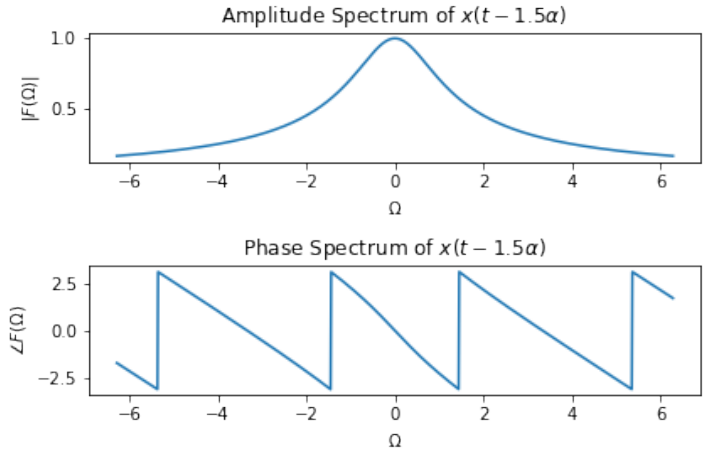

Figure 3: The Fourier Spectrum of $x(t-1.5\alpha)$

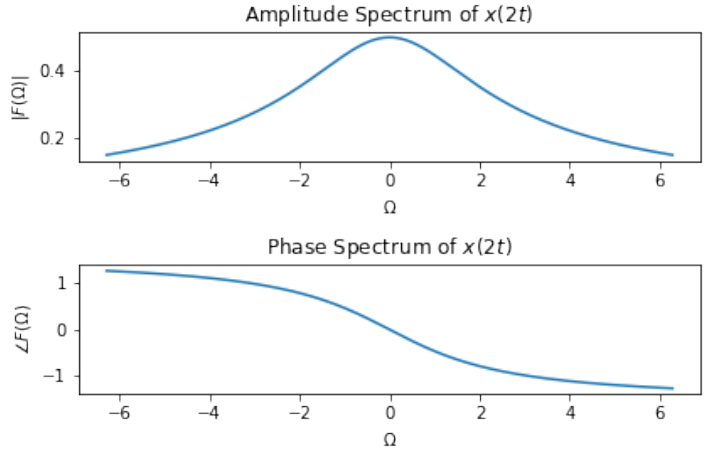

Figure 4: The Fourier Spectrum of $x(2t)$

2. Generating tones

Create 5000 samples of two sinusoids of 200αHz and 220αHz with a time resolution of 0.001s. Append them (to make a 2 x 5000 array), and write it into a .wav file. In two subplots, (i) use the plot function to plot the first 100 samples of the signal and (ii) use the stem function to plot the first 100 samples of the signal.

Algorithms/Methods

In analog systems, signals are processed in their entirety. However, in modern digital systems, only samples of signals are required for processing. This is possible as a result of the sampling theorem

Theorem 1. *A band-limited signal, with no frequency component higher than W hertz, may be completely recovered from its samples taken at a frequency at least twice as high as 2W samples per second.*

In other words, for a signal with bandwidth W hertz, there is no loss of information or overlapping if the sampling frequency f_s is at least twice the highest frequency in the modulating signal. Thus,

$$\frac{1}{T_s} = f_s \geqslant 2W \tag{2.6}$$

The sampling frequency $f_s = 2W$ is known as the *Nyquist frequency* or rate, and $\frac{1}{f_s}$ is the *Nyquist interval*

In our example $T_s = 0.001$s and $f_s = 1000$Hz
Since $\alpha = 1$, The frequencies of the two sinusoids will be $W_1 = 200$Hz and $W_2 = 220$Hz
The sampling theorem is satisfied in both the cases and signals can be sampled satisfactorily

Results

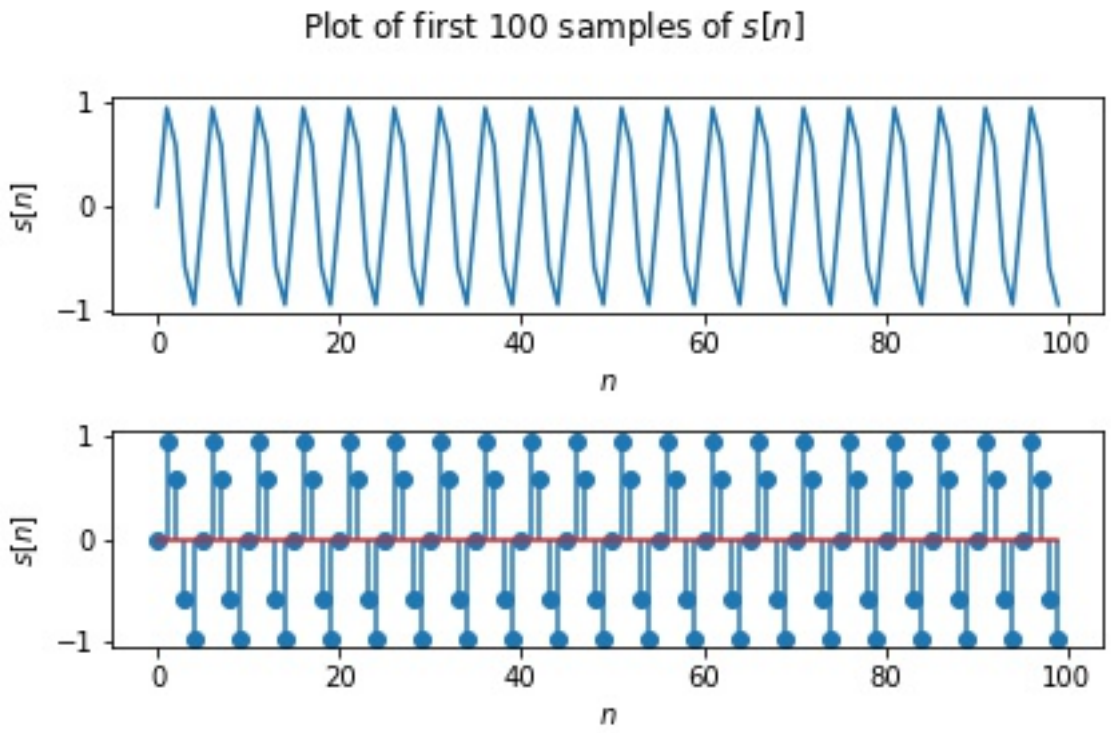

Figure 5: Using the plot and stem functions to plot the first 100 samples of the appended signal

3. Convolution

Load the data Track00α.wav. Load the data from the text file ConvFileα.txt and then convolve the two data streams. Use the convolution function from SciPy and try different modes. Store the result into a `.wav` file. What do you observe? Can you guess the type of filter?

Algorithms/Methods

- The convolution of two signals consists of time-reversing one of the signals, shifting it, and multiplying it point by point with the second signal, and integrating the product.
  In the time domain, the output $y(t)$ of the network is the convolution of the impulse response $h(t)$ with the input $x(t)$, i.e.

$$y(t) = x(t) * h(t) = \int_0^t x(\lambda)h(t-\lambda)d\lambda \tag{2.7}$$

- To convolve two N dimensional arrays in Python we have used
  `scipy.signal.convolve(in1,in2,mode)`
  By default the mode is *'full'* in which the output is the full discrete linear convolution of the inputs.
  The other modes that can be supplied as optional parameter include *'valid'* in which the output consists only of those elements that do not rely on the zero-padding and mode *'same'* in which the output is the same size as in1, centered with respect to the *'full'* output.
  Convolution is performed in all modes and are included in the form `Convolved_signal_mode_<modename>.wav` in the `.zip` submitted along with this report

Results

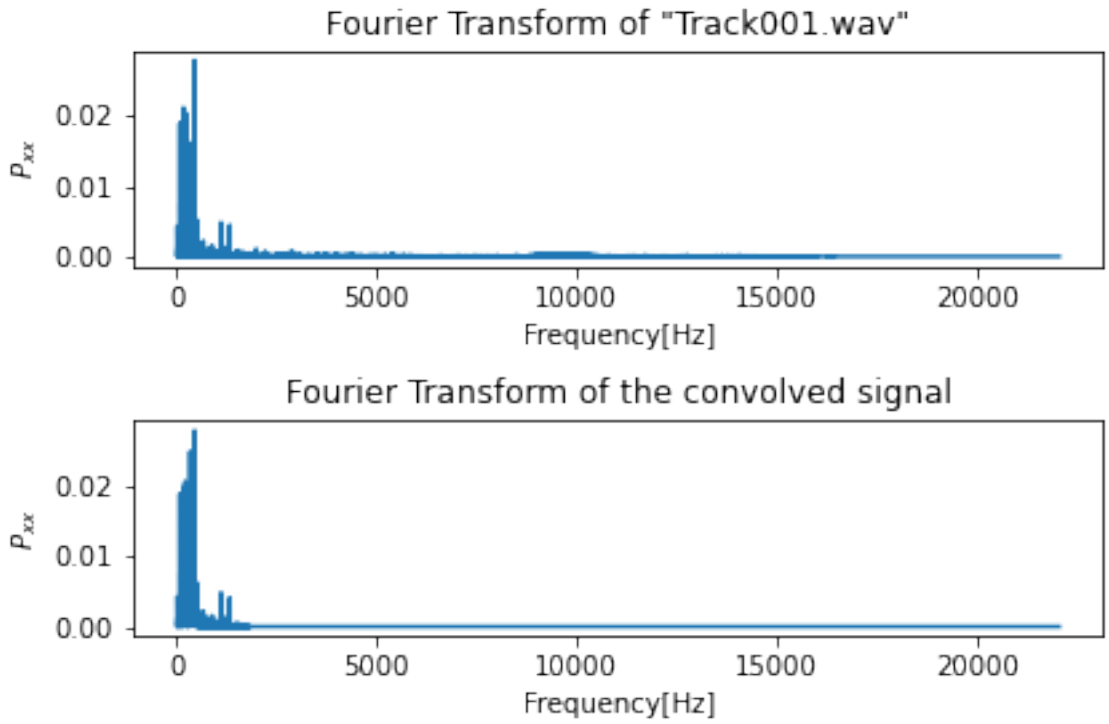

Figure 6: The Fourier transform of the signal before and after convolution

It is observed that the higher frequency components are attenuated after convolution Therefore we can guess that `ConvFile1.txt` is the impulse response of a low pass filter

4. Amplitude Modulation

Import the file speech.wav that contains the speech signal $s(n)$ with Fs as the sampling frequency. Write a user-defined function to obtain

$$y(n) = s(n)\cos(2\pi\frac{F}{Fs}n) \tag{2.8}$$

Generate $y(n)$ for a particular choice of $F = 500$Hz. Plot $s(n)$ and $y(n)$. Can you notice the differences between the signals? Compute, analytically, the Fourier transform of $y(t) = s(t)\cos(2\pi F_0 t)$ and explain the observation. Try: plotting the spectrum of the signals using the **fft** function and verify the claims.

Algorithms/Methods

- Amplitude modulation is a process whereby the amplitude of the carrier is controlled by the modulating signal.
  Amplitude modulation is one of the applications of Fourier Transform and its Frequency shifting property, i.e.
  $$\mathcal{F}[f(t)e^{j\Omega_0 t}] = F(\Omega - \Omega_0) \tag{2.9}$$
  Using Eq. (2.9)
  $$\mathcal{F}[f(t)\cos(\Omega_0 t)] = \frac{1}{2}F(\Omega - \Omega_0) + \frac{1}{2}F(\Omega + \Omega_0) \tag{2.10}$$
- Let $F(\Omega)$ represent the Fourier Transform of our given signal $s(t)$. Utilising the Eq. (2.9) we can compute analytically the Fourier Transform of $y(t) = s(t)\cos(2\pi F_0 t)$ as
  $$\mathcal{F}[y(t)] = \frac{1}{2}F(\Omega - 2\pi F_0) + \frac{1}{2}F(\Omega + 2\pi F_0) \tag{2.11}$$
  meaning a phase shift in the time domain adds a frequency shift in the frequency domain. Therefore if we take the single sided spectrum we will be able to see a right shift in the spectrum by F_0. This is verified in the results below
- **fft**
  A fast Fourier transform (FFT) is an algorithm that calculates the discrete Fourier transform (DFT) of some sequence efficiently. FFTs commonly change the time domain into the frequency domain.
  In Python FFT is calculated using `numpy.fft.fft(signal)`

Results

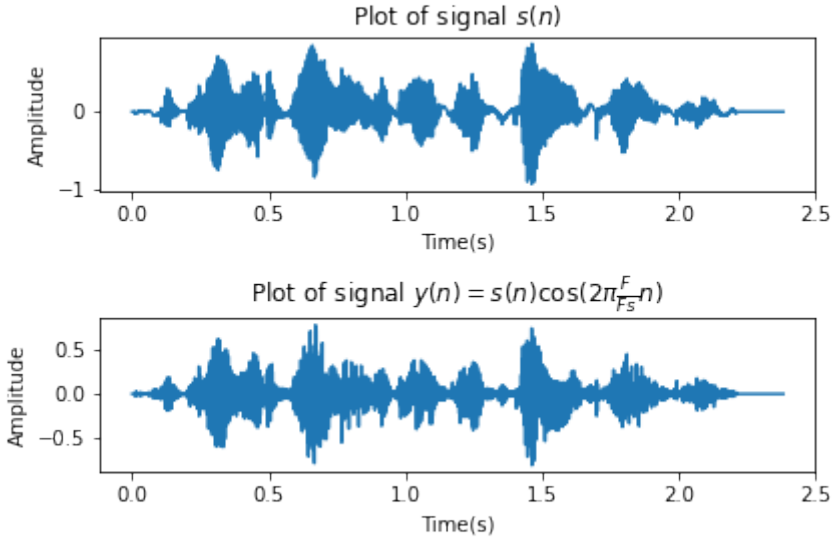

Figure 7: Plot of signal $s(n)$ and modulated signal $y(n)$ for $F = 500$Hz

A difference is heard in the audio of the two waveforms. The modulated signal audio is unclear and sounds like an Alien. The `.wav` files are included in the `.zip` file submitted along with this report.

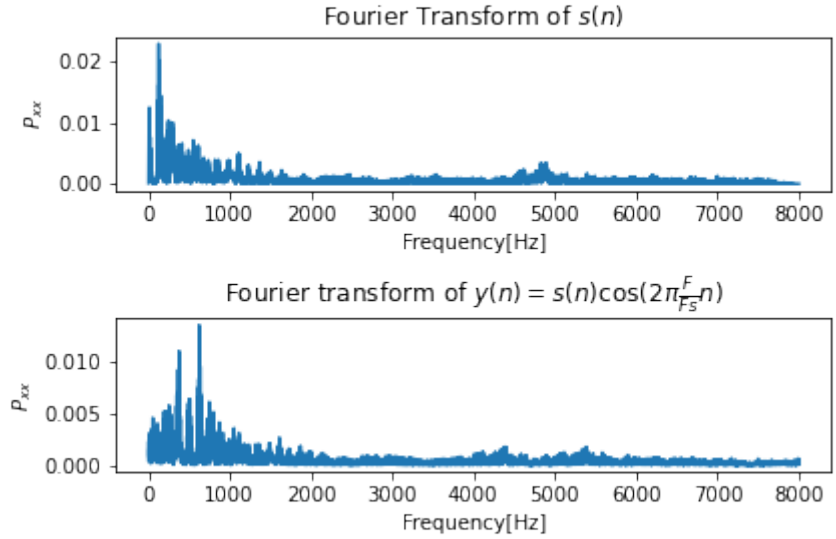

Figure 8: The Fourier transform of signal $s(n)$ and modulated signal $y(n)$

A clear right shift is observed in the Frequency spectrum of $s(n)$ and is evident by looking at the peak frequencies hence confirming our claims.

A Code Repositories

All the code for this exercise and the relevant files are included in the `.zip` file submitted along with this report.