Fourier Transforms

This lesson describes the implementation of Fourier transform.

WE'LL COVER THE FOLLOWING

Introduction

Fourier transform is a method for expressing a function as a weighted sum of sinusoids. Fourier transforms are computed on a time domain signal to check its components in the frequency domain. Fourier transform has vast applications including signal, noise, image, and audio processing.

When both the function and its Fourier transform are replaced with their discretized counterparts, it is called the discrete Fourier transform (DFT). The `fftpack` module in SciPy helps the user compute the DFT using the algorithm Fast Fourier Transform (FFT).

The FFT $y[k]$ (length N) of sequence $x[n]$ (length N) is defined as:

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n]$$

We can go from the frequency domain to the time domain using an inverse Fourier transform. This is defined as:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi j \frac{kn}{N}} y[k]$$

The FFT is computed using `fft()` and inverse transform is computed using `ifft()`.

Implementation

In the example below, we will create a signal as a superposition of 40 Hz and 60 Hz sine waves. We then compute the Fourier transform of the signal and plot the magnitude of the absolute value of the complex discrete Fourier transform coefficients against the frequency. We will expect to see peaks at 40 Hz and 60 Hz.

> 💡 Fourier transform computes the peaks in the negative frequency domain as well. These peaks occur in theory only so we will ignore them and only consider the peaks in the positive frequencies.

```
from scipy import *
from scipy import fftpack
import matplotlib.pyplot as plt

sig_length = 0.5  # signal length seconds
sample_rate = 500  # sampling rate in Hz

dt = 1 / sample_rate   # time between samples in [s]
df = 1 / sig_length   # frequency between frequency points in [Hz]

# array for time
t = arange(0, sig_length, dt)

# only considering the positive frequencies
freq = arange(0, sample_rate/2, df)
n = len(freq)   # length of frequency array

# generating signal
y = 0.5 * sin(2 * pi * 40 * t) + sin(2 * pi * 60 * t + pi/2)

f = fft(y)      # computing fourier transform

# plotting signal
fig, ax = plt.subplots(2, 1)
ax[0].plot(t, y)
ax[0].set_title('time domain')
ax[0].set_xlabel('time(s)')

# plotting fourier transform
ax[1].plot(freq, abs(f[0:n]))
ax[1].set_title('FFT')
ax[1].set_xlabel('frequency(Hz)')
ax[1].set_ylabel('abs(DFT)')

fig.tight_layout()
```

The lower plot shows the discrete Fourier transform that was computed from the data shown in the upper plot. The peaks are at 40 Hz and 60 Hz.

Let’s test your knowledge of the concepts we’ve covered with a quiz.