Introduction to the Fourier Transform

Dennis Sun
Superposition of Signals

\[ 0.8 \cos(2\pi \cdot 220t) + 0.6 \cos(2\pi \cdot 330t) = x(t) \]

If we only observe the signal \( x(t) \), can we back out which frequencies are in the signal?
Superposition of Signals

It turns out to be mathematically more convenient to work with complex exponentials than to work with sines and cosines.

So a sinusoid of frequency $f$ is:

$$e^{i2\pi ft} = \cos(2\pi ft) + i \sin(2\pi ft).$$

We can express any function $x(t)$ as a superposition of sinusoids of different frequencies:

$$x(t) = \int_{-\infty}^{\infty} \hat{x}(f) e^{i2\pi ft} \, df.$$  

Note that $\hat{x}(f)$ is in general a complex number.

To calculate $\hat{x}(f)$ from $x(t)$, we take the **Fourier transform**:

$$\hat{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} \, dt.$$
Fourier Transform

\[ x(t) \quad \xrightarrow{\mathcal{F}} \quad |\hat{x}(f)|^2 \]

Note that we plot \( |\hat{x}(f)|^2 \) because \( \hat{x}(f) \) is a complex number.
Discrete-Time Signals

Most real-world signals with are discrete-time. A discrete-time signal is obtained by sampling a signal at regular time intervals.

The sampling rate $f_s$ (in Hz) specifies the number of samples per second.
Undersampling

It’s possible to choose a sampling rate $f_s$ that is too low.

We say that the higher frequency is aliased by the lower one.

Any frequency above $f_s/2$ will be aliased by a lower frequency.

We call this “maximum frequency” of $f_s/2$ the Nyquist limit.
The Fourier Transform in Practice

- In practice, signals are *discrete-time* and *finite in length*.
- For a signal \( x[n] \) of length \( N \), we use the **Discrete Fourier Transform** (DFT):

\[
\hat{x}[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad k = 0, \ldots, N - 1
\]

\( \hat{x} \) is a vector of length \( N \) that contains the value of the Fourier transform at \( N \) evenly spaced frequencies in \([0, f_s)\).

(But remember that frequencies above \( f_s/2 \) are aliased, so only the first half of \( \hat{x}[k] \) is meaningful.)

- The algorithm used to compute the DFT is called the **Fast Fourier Transform** (FFT). It is implemented in Numpy as \( \text{np.fft.fft} \).

Now go to JupyterHub and work on

/share/Signal Processing in Python.ipynb.