6.003: Signal Processing

Short-Term Fourier Transform

- Window Functions
- FFT Algorithm
Window Functions

A defining feature of the DFT is its finite length $N$, which plays a critical role in determining both time and frequency resolution.

The finite length constraint is equivalent to multiplication by a rectangular window. What would happen if we used a different type of window?
Window Functions

Dozens of different window functions are in common use. We will look at three of them:

- rectangular window
- triangular window
- Hann window

These and other window functions have a variety of different properties. Which properties are important in which applications?
Rectangular Window

Definition:

\[ w_r[n] = \begin{cases} 
1 & |n| \leq M \\
0 & \text{otherwise} 
\end{cases} \]

- Make a plot of \( w_r[n] \) versus \( n \).
- Determine the DT Fourier Transform \( W_r(\Omega) \).
- Make a plot of \( W_r(\Omega) \) versus \( \Omega \).
Triangular Window

Definition:

\[ w_t[n] = \begin{cases} 
1 - \frac{|n|}{M} & |n| \leq M \\
0 & \text{otherwise}
\end{cases} \]

- Make a plot of \( w_t[n] \) versus \( n \).
- Determine the DT Fourier Transform \( W_t(\Omega) \).
- Make a plot of \( W_t(\Omega) \) versus \( \Omega \).
Hann Window

Definition:

\[ w_H[n] = \begin{cases} 
\cos^2 \left( \frac{\pi n}{2M} \right) & -M \leq n \leq M \\
0 & \text{otherwise}
\end{cases} \]

- Make a plot of \( w_H[n] \) versus \( n \).
- Determine the DT Fourier Transform \( W_H(\Omega) \).
- Make a plot of \( W_H(\Omega) \) versus \( \Omega \).
Compare

Superpose the plots of $W_r(\Omega)$, $W_t(\Omega)$, and $W_H(\Omega)$. What are the important differences?
The FFT algorithm is both elegant and critical to modern signal processing.

Start with the definition of the DFT of an $N$ point input $x[n]$:

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

Separate even- and odd-numbered terms (assuming $N$ is even):

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} + \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

$$= \frac{1}{N} \sum_{m=0}^{N/2-1} x[2m] e^{-j2\pi k2m/N} + \frac{1}{N} \sum_{m=0}^{N/2-1} x[2m+1] e^{-j2\pi k(2m+1)/N}$$

$$= \frac{1}{N/2} \sum_{m=0}^{N/2-1} x_e[m] e^{-j2\pi km/N/2} + \frac{1}{N/2} e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} x_o[m] e^{-j2\pi km/N/2}$$

$$= \frac{1}{2} X_e[k] + \frac{1}{2} e^{-j2\pi k/N} X_o[k]$$

where $X_e[k]$ and $X_o[k]$ represent the $N/2$ point DFTs of $x_e[n] = x[2n]$ and $x_o[n] = x[2n + 1]$ respectively.
def DFT(x):
    N = len(x)
    X = []
    for k in range(N):
        X.append(sum([x[n]*e**(-2j*pi*k*n/N)/N for n in range(N)]))
    return X

def FFT(x):
    N = len(x)
    if N==1:
        return x
    xe,xo = x[::2],x[1::2]
    Xe,Xo = FFT(xe),FFT(xo)
    w = e**(-2j*pi/N)
    X = [0]*N
    for k in range(N//2):
        X[k] = (Xe[k]+w**k*Xo[k])/2
        X[k+N//2] = (Xe[k]-w**k*Xo[k])/2
    return X
**FFT Performance**

Compare the running times of DFTs and FFTs.

```python
from random import random
from time import time

for N in (1024, 2048, 4096):
    x = [random() for n in range(N)]
    t0 = time()
    X = FFT(x)
    t1 = time()
    X = DFT(x)
    t2 = time()
    print('{0:.5f} {1:.5f}'.format(t1-t0, t2-t1))
```