6.3000: Signal Processing

FFT and Window Functions

October 31, 2024

Inverse FFT

Here is the lecture code for the FFT algorithm.

```
from math import e,pi
def FFT(x):
    N = len(x)
    if N != 2*N//2:
        print('N must be a power of 2')
        exit(1)
    if N==1:
        return x
    xe = x[::2]
    xo = x[1::2]
    Xe = FFT(xe)
    Xo = FFT(xo)
    X = []
    for k in range(N//2):
        X.append((Xe[k]+e**(-2j*pi*k/N)*Xo[k])/2)
    for k in range(N//2):
        X.append((Xe[k]-e**(-2j*pi*k/N)*Xo[k])/2)
    return X
```

How would you change the code to compute the inverse FFT?

Note: If `FFT(f)` returns `F`, then `iFFT(F)` should return `f`.

Inverse FFT

The FFT computes what we have been calling the DFT **analysis equation**.
The iFFT should compute the DFT **synthesis equation**.

$$F[k] = \frac{1}{N} \sum_{n=0}^{N-1} f[n] e^{-j2\pi kn/N} \qquad \textbf{analysis equation}$$

$$f[n] = \sum_{k=0}^{N-1} F[k] e^{j2\pi kn/N} \qquad \textbf{synthesis equation}$$

The $\frac{1}{N}$ scale factor in the analysis equation is not in the synthesis equation.

The complex exponentials in the two equations are complex conjugates of each other.

Inverse FFT

What do you need to conjugate?

```
from math import e,pi
def FFT(x):
    N = len(x)
    if N != 2*N//2:
        print('N must be a power of 2')
        exit(1)
    if N==1:
        return x
    xe = x[::2]
    xo = x[1::2]
    Xe = FFT(xe)
    Xo = FFT(xo)
    X = []
    for k in range(N//2):
        X.append((Xe[k]+e**(-2j*pi*k/N)*Xo[k])/2)
    for k in range(N//2):
        X.append((Xe[k]-e**(-2j*pi*k/N)*Xo[k])/2)
    return X
```

Inverse FFT

What do you need to conjugate?

```
from math import e,pi
def FFT(x):
    N = len(x)
    if N != 2*N//2:
        print('N must be a power of 2')
        exit(1)
    if N==1:
        return x
    xe = x[::2]
    xo = x[1::2]
    Xe = FFT(xe)
    Xo = FFT(xo)
    X = []
    for k in range(N//2):
        X.append((Xe[k]+e**(+2j*pi*k/N)*Xo[k])/2
    for k in range(N//2):
        X.append((Xe[k]-e**(+2j*pi*k/N)*Xo[k])/2
    return X
```

The only complex numbers in the algorithm are in the complex exponential.

Inverse FFT

Where is the $\frac{1}{N}$ factor?

```
from math import e,pi
def FFT(x):
    N = len(x)
    if N != 2*N//2:
        print('N must be a power of 2')
        exit(1)
    if N==1:
        return x
    xe = x[::2]
    xo = x[1::2]
    Xe = FFT(xe)
    Xo = FFT(xo)
    X = []
    for k in range(N//2):
        X.append((Xe[k]+e**(-2j*pi*k/N)*Xo[k])/2)
    for k in range(N//2):
        X.append((Xe[k]-e**(-2j*pi*k/N)*Xo[k])/2)
    return X
```

Inverse FFT

Where is the $\frac{1}{N}$ factor?

```
from math import e,pi
def FFT(x):
    N = len(x)
    if N != 2*N//2:
        print('N must be a power of 2')
        exit(1)
    if N==1:
        return x
    xe = x[::2]
    xo = x[1::2]
    Xe = FFT(xe)
    Xo = FFT(xo)
    X = []
    for k in range(N//2):
        X.append((Xe[k]+e**(-2j*pi*k/N)*Xo[k])/2)
    for k in range(N//2):
        X.append((Xe[k]-e**(-2j*pi*k/N)*Xo[k])/2)
    return X
```

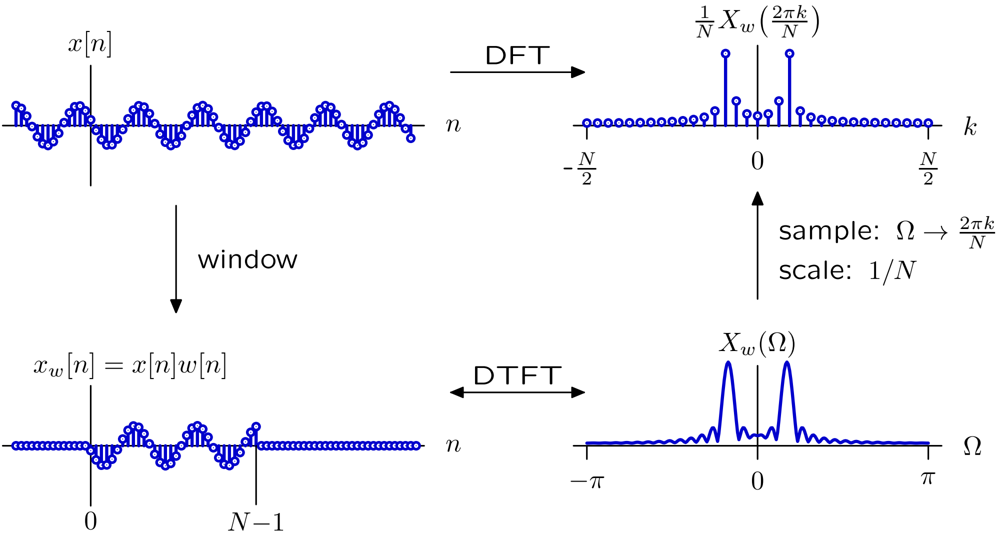
Q: How does a 2 result in N?

A: There is a 2 in each of the $\log_2(N)$ decimations.

Multiply 2 times itself $\log_2(N)$ times: $2^{\log_2(N)} = N$

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. We would like to understand which properties are important in which applications.

Rectangular Window

Definition:

$$w_r[n] = \begin{cases} \frac{1}{2M-1} & 0 \leq n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

- 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 Ω.

Rectangular Window

Definition:

$$w_r[n] = \begin{cases} \frac{1}{2M-1} & 0 \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

One approach would be to close the following sum analytically:

$$W_r(\Omega) = \sum_{n=-\infty}^{\infty} w_r[n] e^{-j\Omega n}$$

Alternatively, we could evaluate the above sum for $\Omega = \frac{2\pi k}{N}$ using a DFT:

$$W_r\left(\frac{2\pi k}{N}\right) = \sum_{n=-\infty}^{\infty} w_r[n] e^{-j\frac{2\pi k}{N}n}$$

Since $w_r[n] = 0$ outside the range $0 \le n \le 2M-2$ we can reduce the infinite sum to a finite sum, which can then be evaluated with a DFT.

$$W_r\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{2M-2} w_r[n] e^{-j\frac{2\pi k}{N}n} = N \times \text{DFT}\{w_r\}$$

We can choose the analysis length N of the DFT based on our desired frequency resolution.

Rectangular Window

Definition:

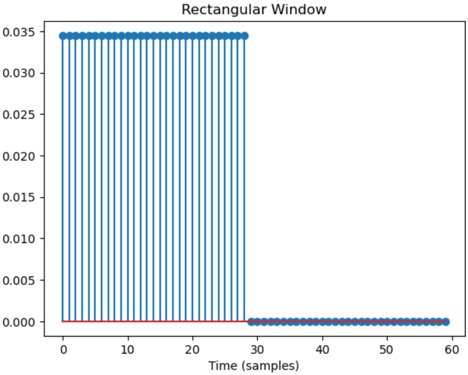
$$w_r[n] = \begin{cases} \frac{1}{2M-1} & 0 \leq n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

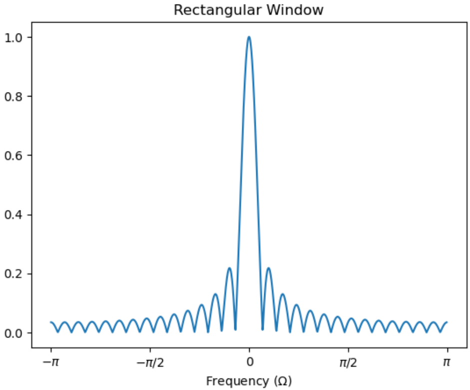
```
from matplotlib.pyplot import plot,stem,xticks,xlabel,title,legend,show
from lib6003.fft import fft
from math import pi,cos,sin

M = 15
N = 1024
w = [1/(2*M-1) for n in range(2*M-1)]
stem(w+(60-len(w))*[0])
xlabel('Time (samples)')
title('Rectangular Window')
show()
W = fft(w+(N-len(w))*[0])
plot([2*pi*k/N for k in range(-N//2,N//2)],[abs(W[k]/W[0]) for k in range(-N//2,N//2)])
xticks([-pi,-pi/2,0,pi/2,pi],['$-\pi$','$-\pi/2$','0','$\pi/2$','$\pi$'])
xlabel('Frequency ($\Omega$)')
title('Rectangular Window')
show()
```

Rectangular Window

Definition:

$$w_r[n] = \begin{cases} \frac{1}{2M-1} & 0 \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$





Triangular Window

Definition:

$$w_t[n] = \begin{cases} \frac{n+1}{M^2} & \text{if } 0 \leq n < M \\ \frac{2M-n-1}{M^2} & \text{if } M \leq n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

- 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 Ω.

Triangular Window

Definition:

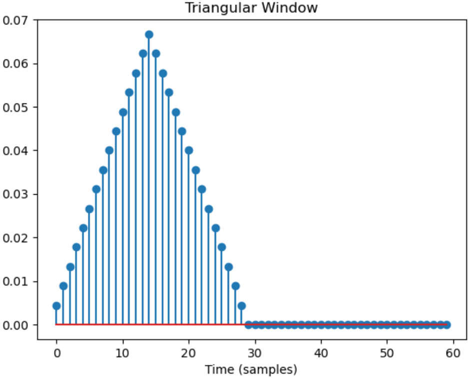
$$w_t[n] = \begin{cases} \frac{n+1}{M^2} & \text{if } 0 \le n < M \\ \frac{2M-n-1}{M^2} & \text{if } M \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

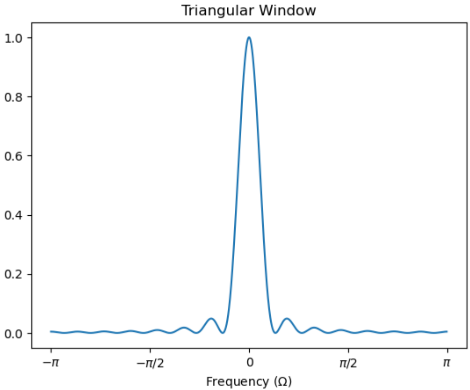
```
from matplotlib.pyplot import ion,plot,stem,xticks,xlabel,title,legend,show
from lib6003.fft import fft
from math import pi,cos,sin

w = [(n+1)/M/M for n in range(M)]+[(2*M-n-1)/M/M for n in range(M,2*M-1)]
stem(w+100*[0])
xlabel('Time (samples)')
title('Triangular Window')
show()
W = fft(w+(N-len(w))*[0])
plot([2*pi*k/N for k in range(-N//2,N//2)],[abs(W[k]/W[0]) for k in range(-N//2,N//2)])
xticks([-pi,-pi/2,0,pi/2,pi],['$-\pi$','$-\pi/2$','0','$\pi/2$','$\pi$'])
xlabel('Frequency ($\Omega$)')
title('Triangular Window')
show()
```

Triangular Window

Definition:

$$w_t[n] = \begin{cases} \frac{n+1}{M^2} & \text{if } 0 \le n < M \\ \frac{2M-n-1}{M^2} & \text{if } M \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$





Hann Window

Definition:

$$w_h[n] = \begin{cases} \frac{1}{5}\sin^2\left(\frac{\pi*(n+1)}{2M-1}\right) & 0 \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

- 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 Ω.

Hann Window

Definition:

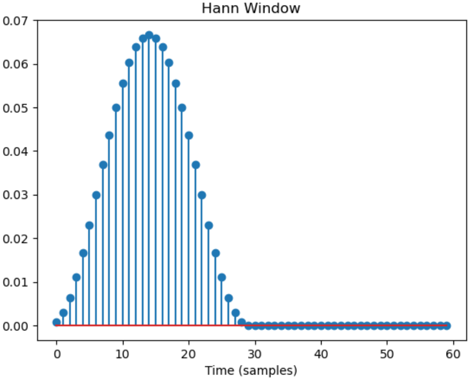
$$w_h[n] = \begin{cases} \frac{1}{5}\sin^2\left(\frac{\pi*(n+1)}{2M-1}\right) & 0 \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$

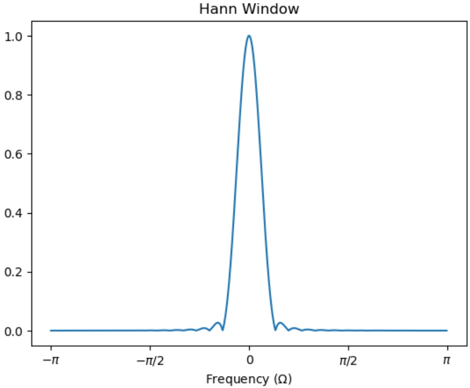
```
from matplotlib.pyplot import ion,plot,stem,xticks,xlabel,title,legend,show
from lib6003.fft import fft
from math import pi,cos,sin

M = 15
N = 1024
w = [sin(pi*(n+1)/(2*M))**2/M for n in range(2*M-1)]
stem(w+100*[0])
xlabel('Time (samples)')
title('Hann Window')
show()
W = fft(w+(N-len(w))*[0])
plot([2*pi*k/N for k in range(-N//2,N//2)],[abs(W[k]/W[0]) for k in range(-N//2,N//2)])
xticks([-pi,-pi/2,0,pi/2,pi],['$-\pi$','$-\pi/2$','0','$\pi/2$','$\pi$'])
xlabel('Frequency ($\Omega$)')
title('Hann Window')
show()
```

Hann Window

Definition:

$$w_h[n] = \begin{cases} \frac{1}{5}\sin^2\left(\frac{\pi*(n+1)}{2M-1}\right) & 0 \le n < 2M-1 \\ 0 & \text{otherwise} \end{cases}$$





Compare

Superpose the plots of $W_r(\Omega)$, $W_t(\Omega)$, and $W_h(\Omega)$.

What are the important differences?

Compare

Superpose the plots of $W_r(\Omega)$, $W_t(\Omega)$, and $W_h(\Omega)$.

What are the important differences?