Fourier Transform

TA: Junhee Lee

Example #10-1. Generate the sum of two sine functions and implement its FFT .

Python Code [1]

```
## import modules
import numpy as np
from scipy.fftpack import fft
import matplotlib.pyplot as plt

## set the number of sample points
N = 600

## sample spacing
T = 1.0 / 800.0

## generate the sum of sine functions
x = np.linspace(0.0, N*T, N)
y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)

## FFT
yf = fft(y)
xf = np.linspace(0.0, 1.0/(2.0*T), N//2)

## plot the results
#plt.plot(x,y)
plt.plot(xf, 2.0/N * np.abs(yf[0:N//2]))
plt.grid()
plt.show()
```

Example #10-2. Plot the FFT of two complex exponentials.

Python Code [1]

```
## import modules
import numpy as np
from scipy.fftpack import fft, fftfreq, fftshift
import matplotlib.pyplot as plt

## number of signal points
N = 400

## sample spacing
T = 1.0 / 800.0

## generate the functions
x = np.linspace(0.0, N*T, N)
y = np.exp(50.0 * 1.j * 2.0*np.pi*x) + 0.5*np.exp(-80.0 * 1.j * 2.0*np.pi*x)

## FFT
yf = fft(y)
xf = fftfreq(N, T)
xf = fftshift(xf)

## plot the results
yplot = fftshift(yf)
plt.plot(xf, 1.0/N * np.abs(yplot))
plt.grid()
plt.show()
```

Example #10-3. Demonstrate a 2-dimensional IFFT and plot the resulting 2-dimensional time-domain signals.

Python Code [1]

```
## import modules
import numpy as np
from scipy.fftpack import ifftn
import matplotlib.pyplot as plt
import matplotlib.cm as cm

## Number of sample points
N = 30

## main

f, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(2, 3, sharex='col', sharey='row')

xf = np.zeros((N,N))
xf[0, 5] = 1
xf[0, N-5] = 1

Z = ifftn(xf)

ax1.imshow(xf, cmap=cm.Reds)
ax4.imshow(np.real(Z), cmap=cm.gray)

xf = np.zeros((N, N))
xf[5, 0] = 1
xf[N-5, 0] = 1

Z = ifftn(xf)

ax2.imshow(xf, cmap=cm.Reds)
ax5.imshow(np.real(Z), cmap=cm.gray)

xf = np.zeros((N, N))
xf[5, 10] = 1
xf[N-5, N-10] = 1

Z = ifftn(xf)

ax3.imshow(xf, cmap=cm.Reds)
ax6.imshow(np.real(Z), cmap=cm.gray)

plt.show()
```

Example #10-4. Show a signal x and two reconstructions, x_{20} and x_{15}, from the signal's DCT(Discrete Cosine Transforms) coefficients.

Python Code [1]

```
import numpy as np
from scipy.fftpack import dct, idct
import matplotlib.pyplot as plt

N = 100

t = np.linspace(0,20,N)
x = np.exp(-t/3)*np.cos(2*t)
y = dct(x, norm='ortho')

window = np.zeros(N)
window[:20] = 1

yr = idct(y*window, norm='ortho')
sum(abs(x-yr)**2) / sum(abs(x)**2)

plt.plot(t, x, '-bx')
plt.plot(t, yr, 'ro')

window = np.zeros(N)
window[:15] = 1

yr = idct(y*window, norm='ortho')
sum(abs(x-yr)**2) / sum(abs(x)**2)

plt.plot(t, yr, 'g+')
plt.legend(['x', '$x_{20}$', '$x_{15}$'])
plt.grid()
plt.show()
```

References

[1] Fourier Transforms (scipy.fftpack), https://docs.scipy.org/doc/scipy/reference/tutorial/fftpack.html (accessed May 16th, 2018)