The Discrete Fourier Transform

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra, Barcelona
http://mtg.upf.edu

Index

- DFT equation
- Complex exponentials
- Inner product
- DFT of complex sinusoids
- DFT of real sinusoids
- Inverse-DFT

Discrete Fourier Transform

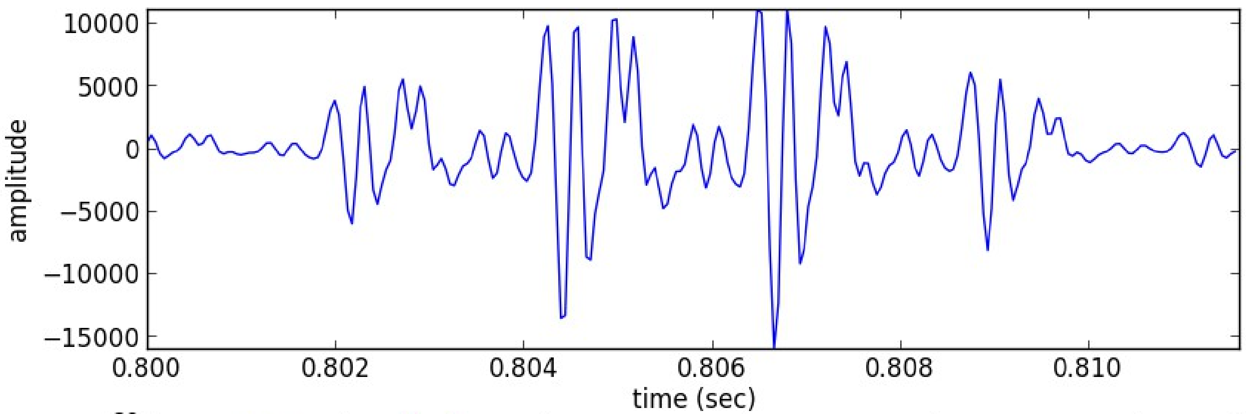
$$X[k]=\sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \quad k=0,\ldots,N-1$$

n: discrete time index in samples (normalized time, T=1)

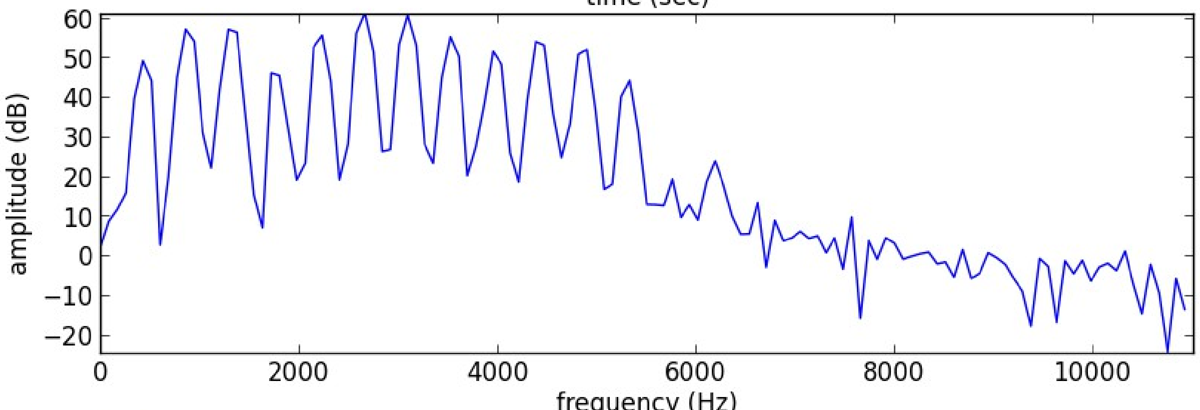
k: discrete frequency index in bins

$\omega_k=2\pi k/N$ frequency in radians

$f_k=f_s k/N$ frequency in Hz, where fs is the sampling rate

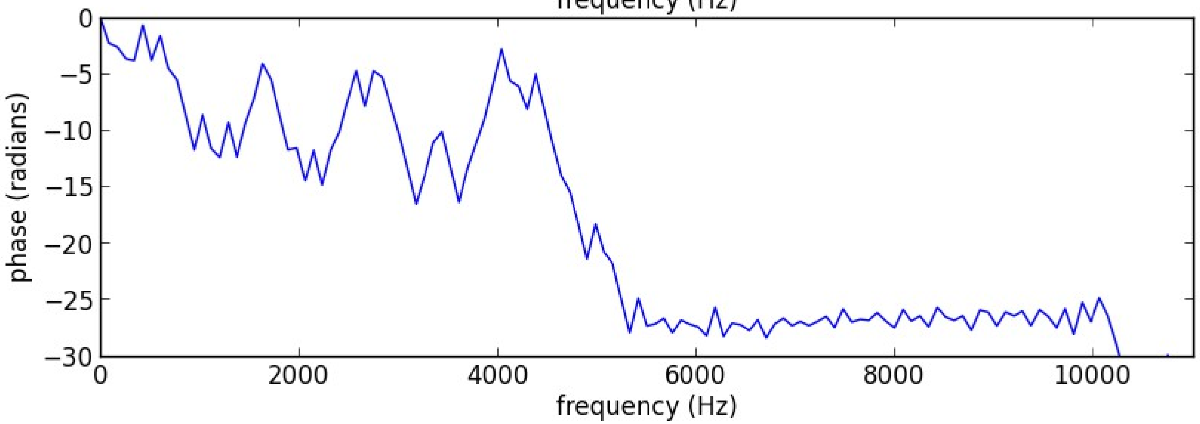


fragment of a sound, x[n]



magnitude spectrum

$20 * \log_{10}(X[k])$



phase spectrum

$\measuredangle X[k]$

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.io.wavfile import read
from scipy.fftpack import fft

(fs, x) = read('oboe.wav')
size = 256
start = .8*fs
xw = x[start:start+size] * np.hamming(size)

plt.subplot(311)
plt.plot(np.arange(start, (start+size), 1.0)/fs, xw)

X = fft(xw)
mX = 20 * np.log10(abs(X)/size)
pX = np.unwrap(np.angle(X))

plt.subplot(312)
plt.plot(np.arange(0, fs/2.0, float(fs)/size), mX[0:size/2])

plt.subplot(313)
plt.plot(np.arange(0, fs/2.0, float(fs)/size), pX[:size/2])
```

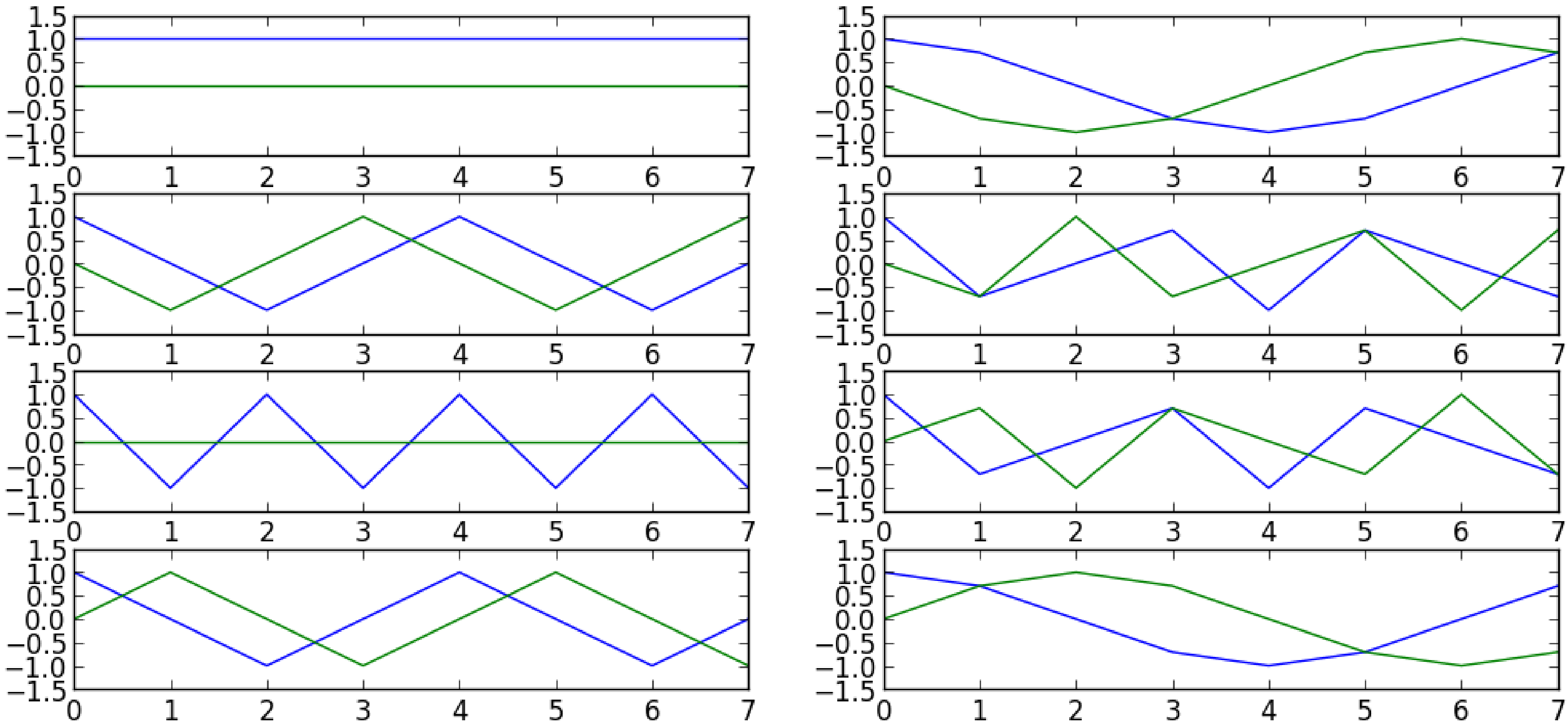
Complex exponentials

$$\bar{s}_k = e^{-j2\pi kn/N} = \cos(2\pi kn/N) - j\sin(2\pi kn/N)$$

$$\text{for } N=4, \quad \text{thus for} \quad n=0,1,2,3\,;\, k=0,1,2,3$$

$$\bar{s}_0 = \cos(2\pi*0*n/4) - j\sin(2\pi*0*n/4) = [1,1,1,1]$$
$$\bar{s}_1 = \cos(2\pi*1*n/4) - j\sin(2\pi*1*n/4) = [1,-j,-1,j]$$
$$\bar{s}_2 = \cos(2\pi*2*n/4) - j\sin(2\pi*2*n/4) = [1,-1,1,-1]$$
$$\bar{s}_3 = \cos(2\pi*3*n/4) - j\sin(2\pi*3*n/4) = [1,j,-1,-j]$$

Complex exponentials




```
import matplotlib.pyplot as plt
import numpy as np
N = 8
for k in range(N):
 s = np.exp(-1j*2*np.pi*k/N*np.arange(N))
 plt.subplot(N/2, 2, k+1)
 plt.plot(np.real(s))
 plt.subplot(N/2, 2, k+1)
 plt.plot(np.imag(s))
```

Inner product - DFT

$$\langle x , s_k \rangle = \sum_{n=0}^{N-1} x[n] * \bar{s}_k[n] = \sum_{n=0}^{N-1} x[n] * e^{-j2\pi kn/N}$$

Example:

$$x[n] = [1, -1, 1, -1]; N = 4$$

$$\langle x , s_0 \rangle = 1*1 + (-1)*1 + 1*1 + (-1)*1 = 0$$
$$\langle x , s_1 \rangle = 1*1 + (-1)*(-j) + 1*(-1) + (-1)*j = 0$$
$$\langle x , s_2 \rangle = 1*1 + (-1)*(-1) + 1*1 + (-1)*(-1) = 4$$
$$\langle x , s_3 \rangle = 1*1 + (-1)*(-j) + 1*(-1) + (-1)*j = 0$$

Inner product - DFT

```
import numpy as np

x = np.array([1,-1,1,-1])
print 'x = {}'.format(x)
N = 4
for k in range(N):
   s = np.exp(1j*2*np.pi*k/N*np.arange(N))
   print 's{0} = {1}'.format(k, s)
   X = sum(x*np.conjugate(s))
   print '<x,s{0}> = {1}'.format(k,X)
```

Output:

```
x = [ 1 -1  1 -1]
s0 = [ 1.+0.j  1.+0.j  1.+0.j  1.+0.j]
<x,s0> = 0.0
s1 = [ 1.+0.j  0.+1.j -1.+0.j -0.-1.j]
<x,s1> = 1.32693504719e-16
s2 = [ 1.+0.j -1.+0.j  1.-0.j -1.+0.j]
<x,s2> = 4.0
s3 = [ 1.+0.j -0.-1.j -1.+0.j  0.+1.j]
<x,s3> = 5.52708599219e-16
```

DFT of complex sinusoid

$$x_1[n] = e^{j2\pi k_0 n/N} \quad \text{for } n = 0, \ldots, N-1$$

$$X[k] = \sum_{n=0}^{N-1} x_1[n] e^{-j2\pi kn/N}$$

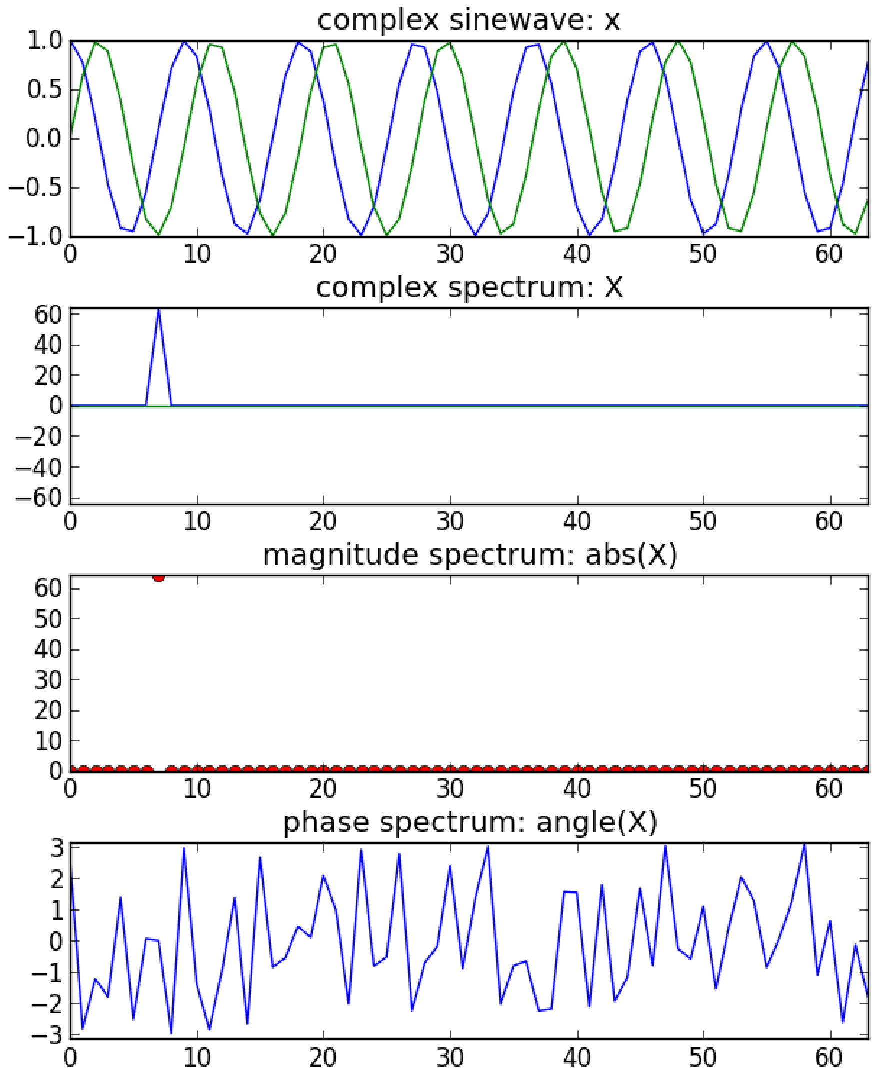
$$= \sum_{n=0}^{N-1} e^{-j2\pi k_0 n/N} e^{-j2\pi k n/N}$$

$$= \sum_{n=0}^{N-1} e^{-j2\pi (k-k_0) n/N}$$

$$= \frac{1 - e^{-j2\pi (k-k_0)}}{1 - e^{-j2\pi (k-k_0)/N}} \quad \text{(sum of a geometric series)}$$

if $k \neq k_0$: denominator $\neq 0$, numerator $= 0$

thus $X[k] = N$ for $k = k_0$ and $X[k] = 0$ for $k \neq k_0$




```
import numpy as np
import matplotlib.pyplot as plt

N = 64
k0 = 7
X = np.array([])
x = np.exp(1j*2*np.pi*k0/N*np.arange(N))

plt.subplot(411)
plt.plot(np.arange(N), np.real(x))
plt.plot(np.arange(N), np.imag(x))

for k in range(N):
 s = np.exp(1j*2*np.pi*k/N*np.arange(N))
 X = np.append(X, sum(x*np.conjugate(s)))

plt.subplot(412)
plt.plot(np.arange(N), np.real(X))
plt.plot(np.arange(N), np.imag(X))

plt.subplot(413)
plt.plot(np.arange(N), abs(X), 'ro')

plt.subplot(414)
plt.plot(np.arange(N), np.angle(X))
```

DFT of complex sinusoid

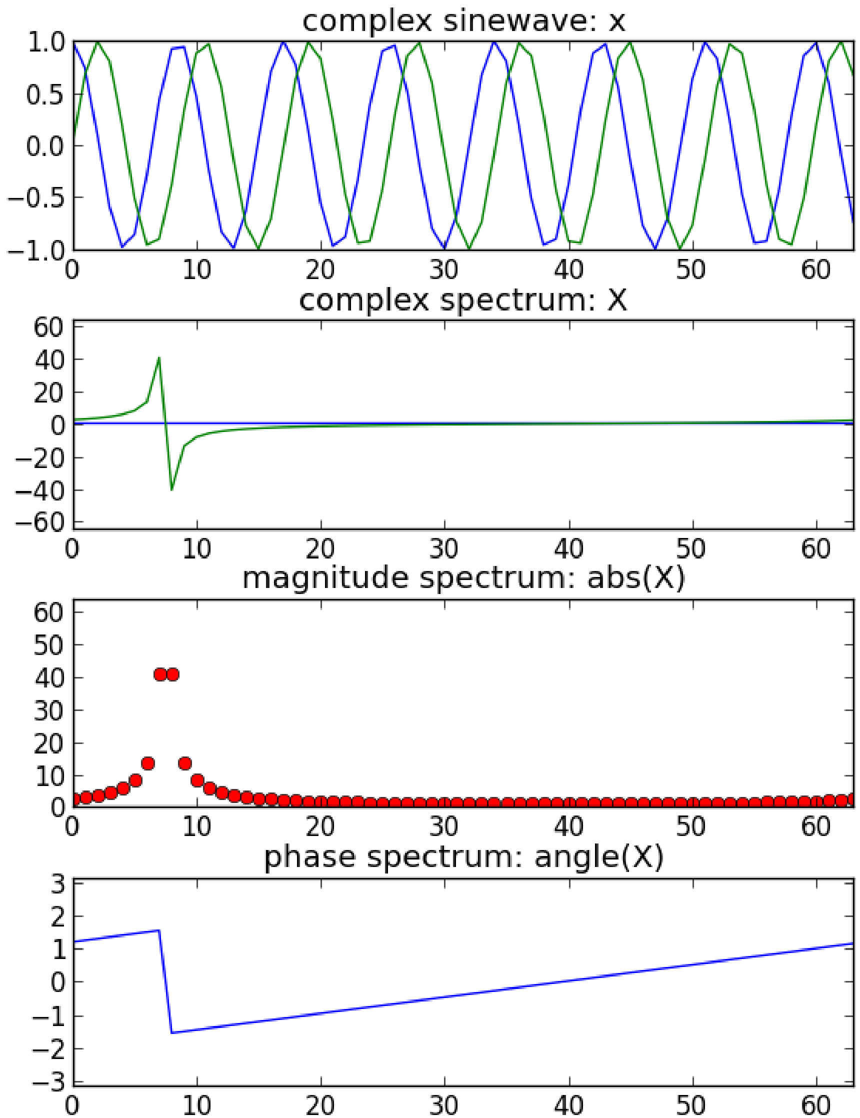
$$x_2[n]=e^{j2\pi f_0 n+\varphi} \quad \text{for } n=0,\dots,N-1$$

$$X_2[k]=\sum_{n=0}^{N-1} x_2[n]e^{-j2\pi kn/N}$$

$$=\sum_{n=0}^{N-1} e^{j2\pi f_0 n+\varphi} e^{-j2\pi kn/N}$$

$$=e^{j\varphi}\sum_{n=0}^{N-1} e^{-j2\pi(k/N-f_0)n}$$

$$=e^{j\varphi}\frac{1-e^{-j2\pi(k/N-f_0)N}}{1-e^{-j2\pi(k/N-f_0)}}$$




```
N = 64
f0 = 7.5/N
X = np.array([])
x = np.exp(1j*2*np.pi*f0/N*np.arange(N))

plt.subplot(411)
plt.plot(np.arange(N), np.real(x))
plt.plot(np.arange(N), np.imag(x))

for k in range(N):
 s = np.exp(1j*2*np.pi*k/N*np.arange(N))
 X = np.append(X, sum(x*np.conjugate(s)))

plt.subplot(412)
plt.plot(np.arange(N), np.real(X))
plt.plot(np.arange(N), np.imag(X))

plt.subplot(413)
plt.plot(np.arange(N), abs(X), 'ro')

plt.subplot(414)
plt.plot(np.arange(N), np.angle(X))
```

DFT of real sinusoids

$$x_2[n] = A_0 \cos(2\pi k_0 n/N) = \frac{A_0}{2} e^{j2\pi k_0 n/N} + \frac{A_0}{2} e^{-j2\pi k_0 n/N}$$

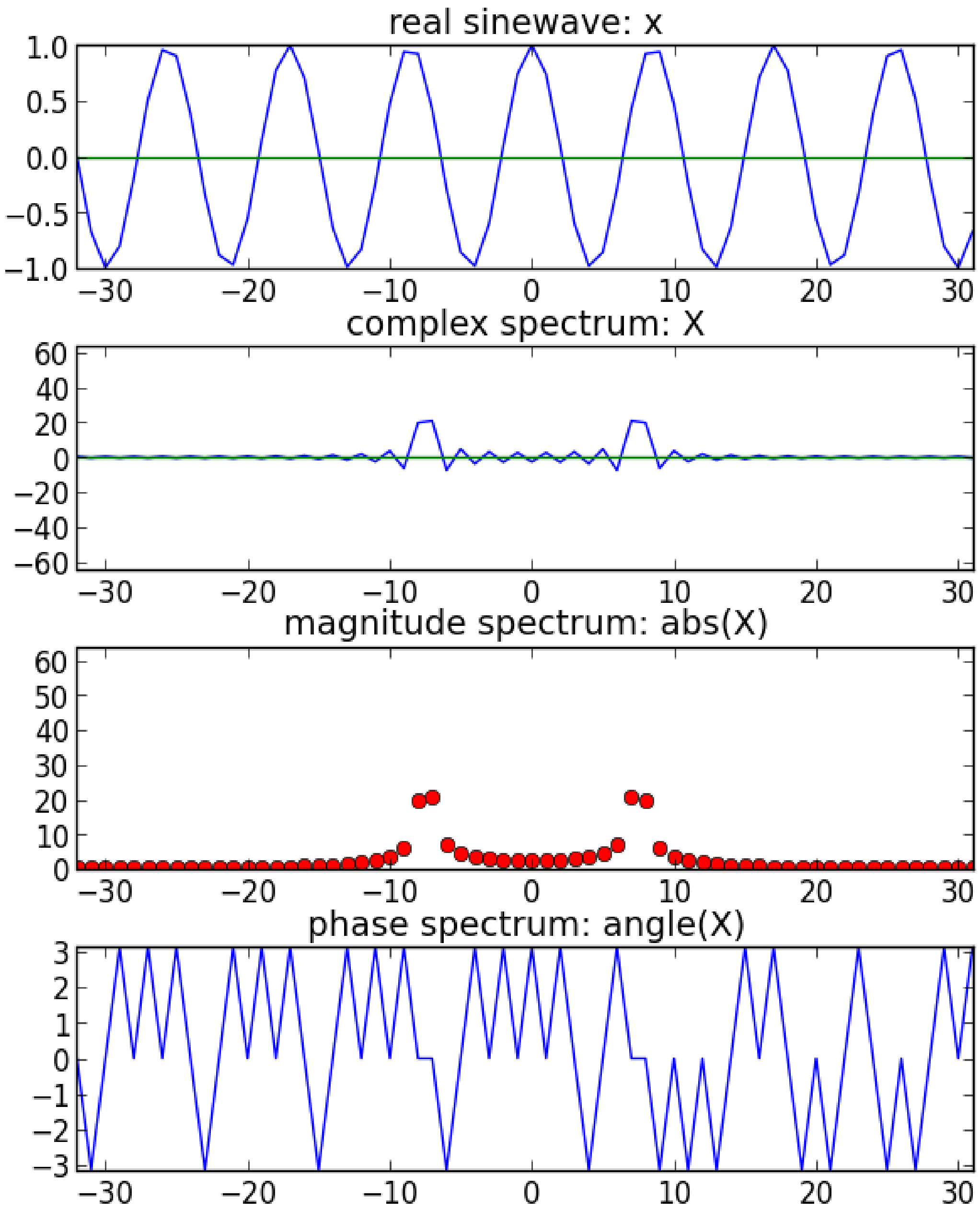
$$X[k] = \sum_{n=-N/2}^{N/2-1} x_2[n] e^{-j2\pi kn/N}$$

$$= \sum_{n=-N/2}^{N/2-1} \left(\frac{A_0}{2} e^{j2\pi k_0 n/N} + \frac{A_0}{2} e^{-j2\pi k_0 n/N}\right) e^{-j2\pi kn/N}$$

$$= \sum_{n=-N/2}^{N/2-1} \frac{A_0}{2} e^{j2\pi k_0 n/N} e^{-j2\pi kn/N} + \sum_{n=-N/2}^{N/2-1} \frac{A_0}{2} e^{-j2\pi k_0 n/N} e^{-j2\pi kn/N}$$

$$= \sum_{n=-N/2}^{N/2-1} \frac{A_0}{2} e^{-j2\pi(k-k_0)n/N} + \sum_{n=-N/2}^{N/2-1} \frac{A_0}{2} e^{-j2\pi(k+k_0)n/N}$$

$$= \frac{A_0}{2} \; for \; k = k_{0,} - k_0 ; 0 \; for \text{ rest of } k$$




```
import numpy as np
import matplotlib.pyplot as plt

N = 64
k0 = 7.5
X = np.array([])
nv = np.arange(-N/2, N/2)
kv = np.arange(-N/2, N/2)
x = np.cos(2*np.pi*k0/N*nv)

plt.subplot(411)
plt.plot(nv, np.real(x))
plt.plot(nv, np.imag(x))
for k in kv:
 s=np.exp(1j*2*np.pi*k/N*nv)
 X=np.append(X,sum(x*np.conj(s)))

plt.subplot(412)
plt.plot(kv, np.real(X))
plt.plot(kv, np.imag(X))

plt.subplot(413)
plt.plot(kv, abs(X), 'ro')

plt.subplot(414)
plt.plot(kv, np.angle(X))
```

Inverse DFT

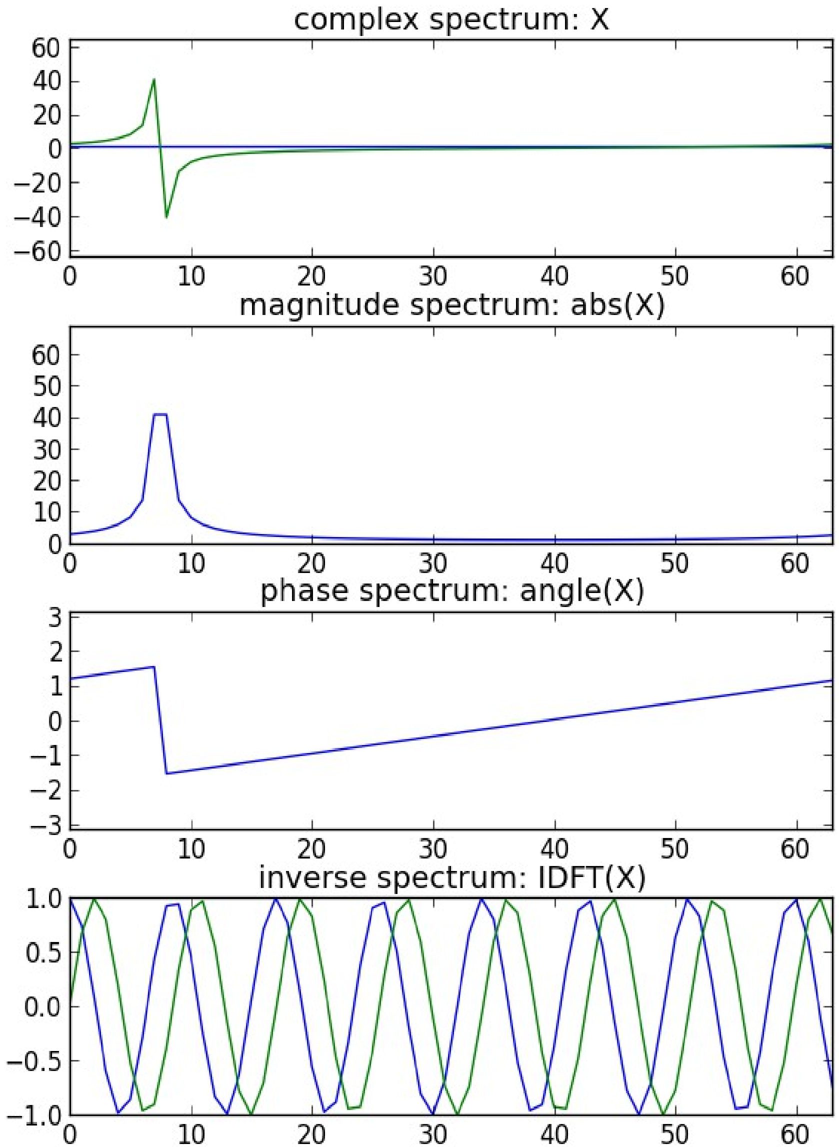
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] * s_k[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] * e^{j2\pi kn/N} \qquad n = 0,1,\ldots, N-1$$

Example:

$$X[k] = [0,4,0,0];\ N = 4$$

$$X * s_0 = 0*1 + 4*1 + 0*1 + 0*1 = 4$$
$$X * s_1 = 0*1 + 4*j + 0*(-1) + 0*(-j) = 4\text{j}$$
$$X * s_2 = 0*1 + 4*(-1) + 0*1 + 0*(-1) = -4$$
$$X * s_3 = 0*1 + 4*(-j) + 0*(-1) + 0*j = -4\text{j}$$

Inverse DFT example




```
import numpy as np
import matplotlib.pyplot as plt

N = 64
k0 = 7.5

x = np.exp(1j*2*np.pi*k0/N*np.arange(N))
for k in range(N):
 s = np.exp((1j*2*np.pi*k/N)*np.arange(N))
 X = np.append(X, sum(x*np.conjugate(s)))

plt.subplot(411)
plt.plot(np.arange(N), np.real(X))
plt.plot(np.arange(N), np.imag(X))

plt.subplot(412)
plt.plot(np.arange(N), abs(X))

plt.subplot(413)
plt.plot(np.arange(N), np.angle(X))

for n in range(N):
 s = np.exp((1j*2*np.pi*n/N)*np.arange(N))
 y = np.append(y, sum(X*s)/N)

plt.subplot(414)
plt.plot(np.arange(N), np.real(y))
plt.plot(np.arange(N), np.imag(y))
```

References

- https://ccrma.stanford.edu/~jos/mdft/
- Full code of plots and accompanying labs available at:https://github.com/MTG/sms-tools
-

Credits