Signal Processing S2
Week 11: Fourier Properties

@btatmaja

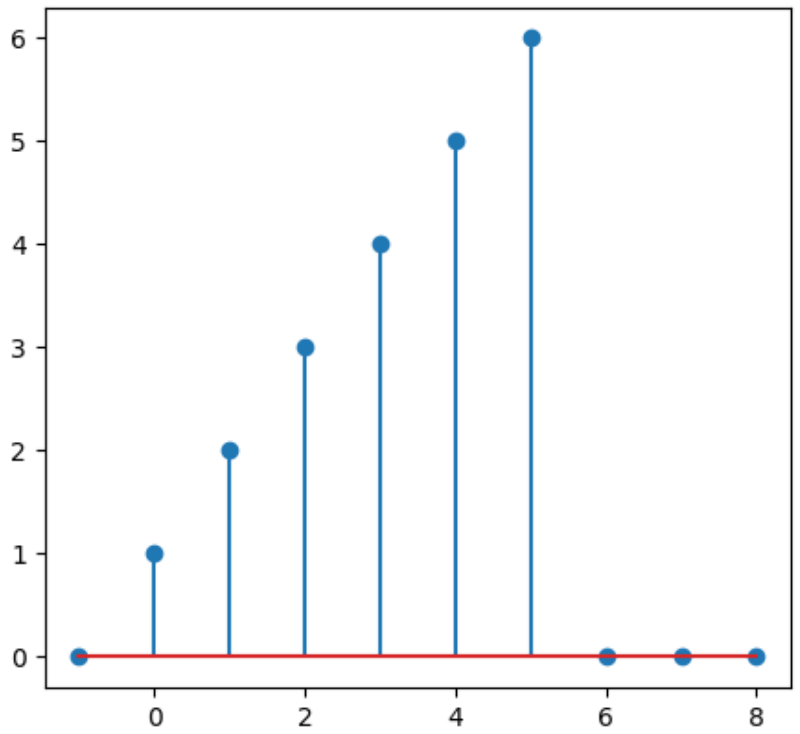
Index Fourier Properties I

- Linearity
- Shift
- Symmetry
- Convolution

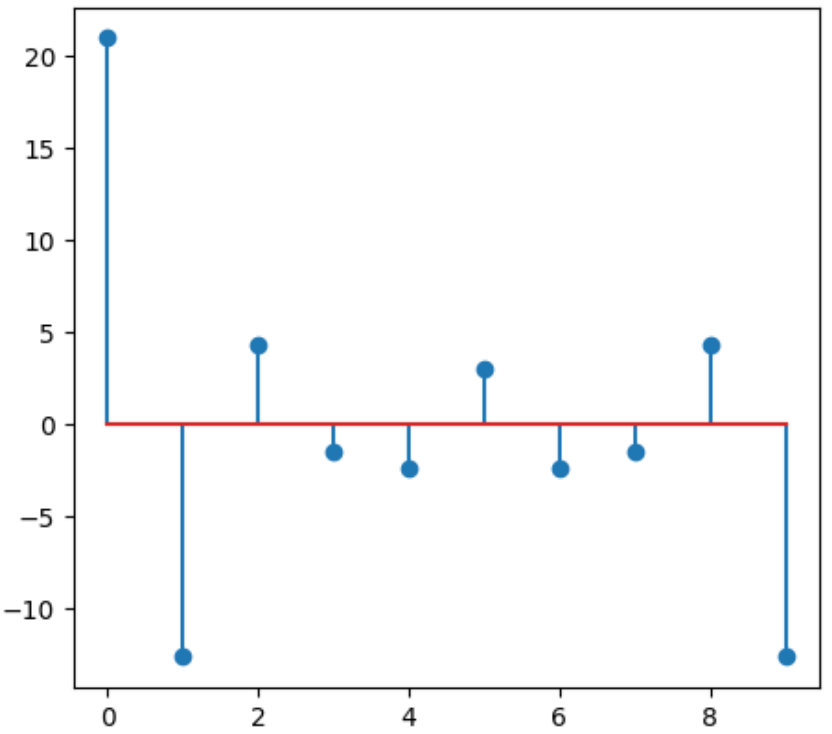
Recall DFT

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

$x[n]$

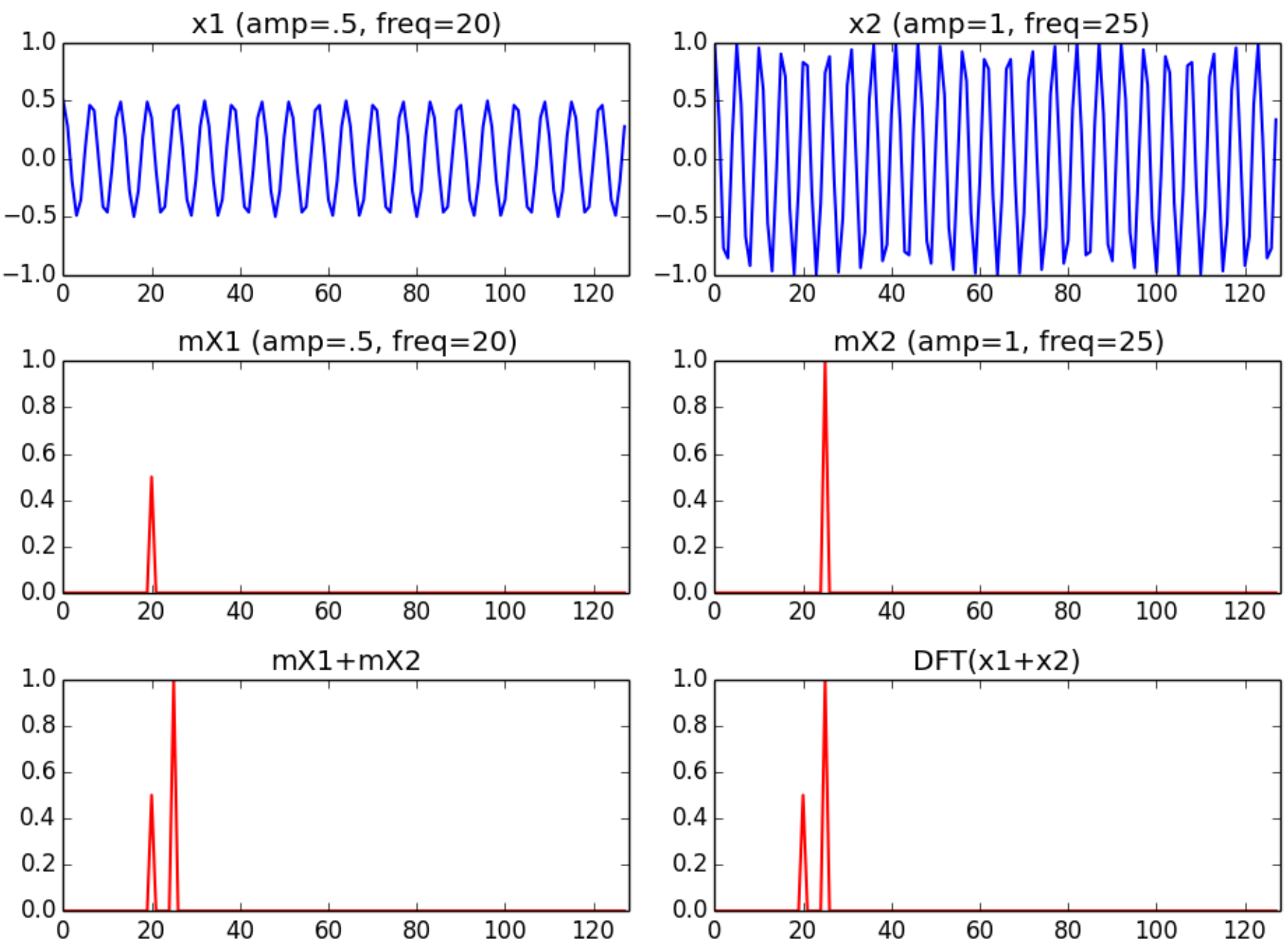


$X[k]$



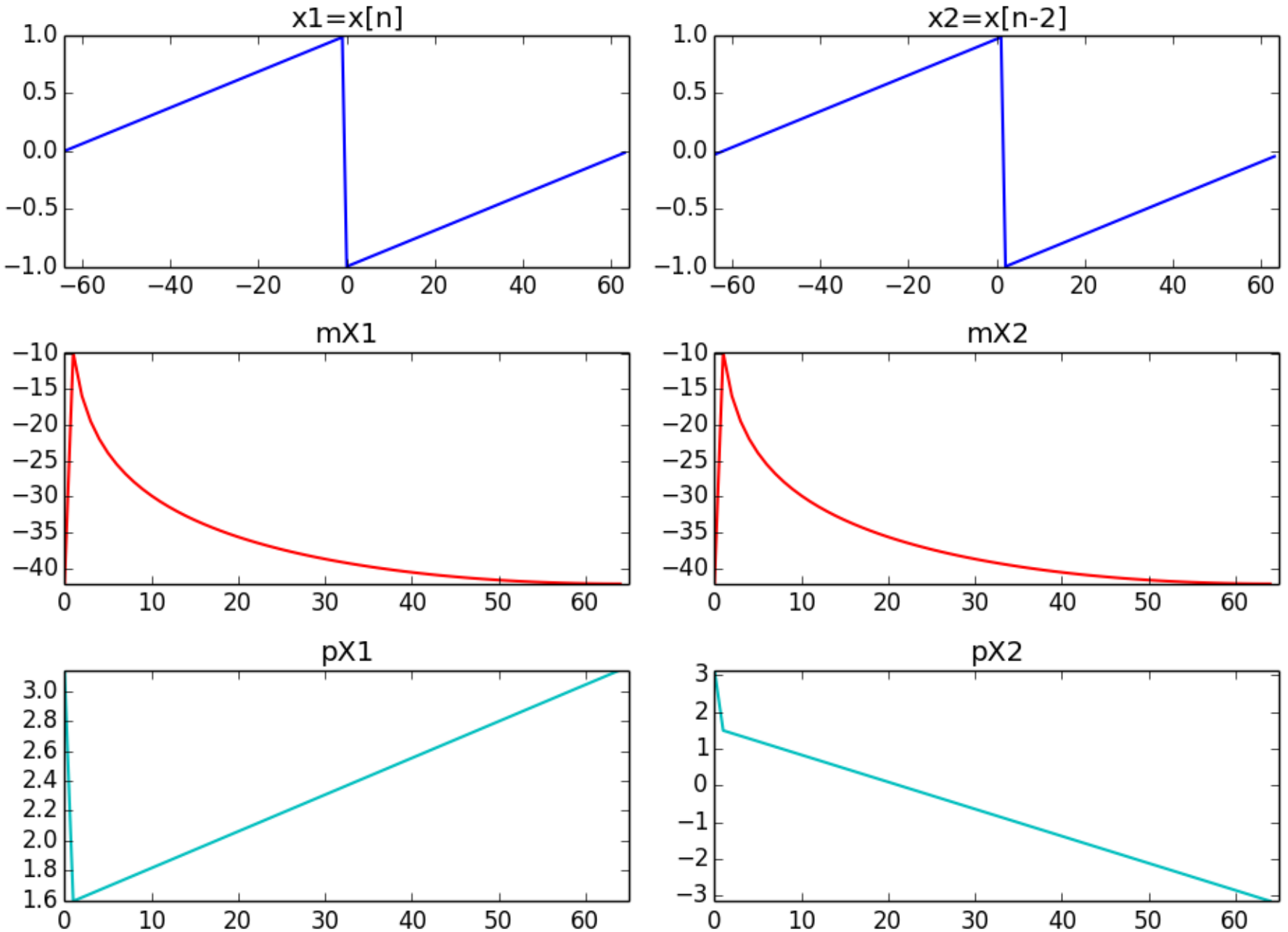
Linearity: $a\,x_1[n]+b\,x_2[n] \Leftrightarrow a\,X_1[k]+b\,X_2[k]$

$$
\begin{aligned}
&DFT\left(a\,x_1[n]+b\,x_2[n]\right) \\
&\quad=\sum_{n=0}^{N-1}\left(a\,x_1[n]+b\,x_2[n]\right)e^{-j2\pi kn/N} \\
&\quad=\sum_{n=0}^{N-1}a\,x_1[n]e^{-j2\pi kn/N}+\sum_{n=0}^{N-1}b\,x_2[n]e^{-j2\pi kn/N} \\
&\quad=a\sum_{n=0}^{N-1}x_1[n]e^{-j2\pi kn/N}+b\sum_{n=0}^{N-1}x_2[n]e^{-j2\pi kn/N} \\
&\quad=a\,X_1[k]+b\,X_2[k]
\end{aligned}
$$


x1 (amp=.5, freq=20)
x2 (amp=1, freq=25)
mX1 (amp=.5, freq=20)
mX2 (amp=1, freq=25)
mX1+mX2
DFT(x1+x2)

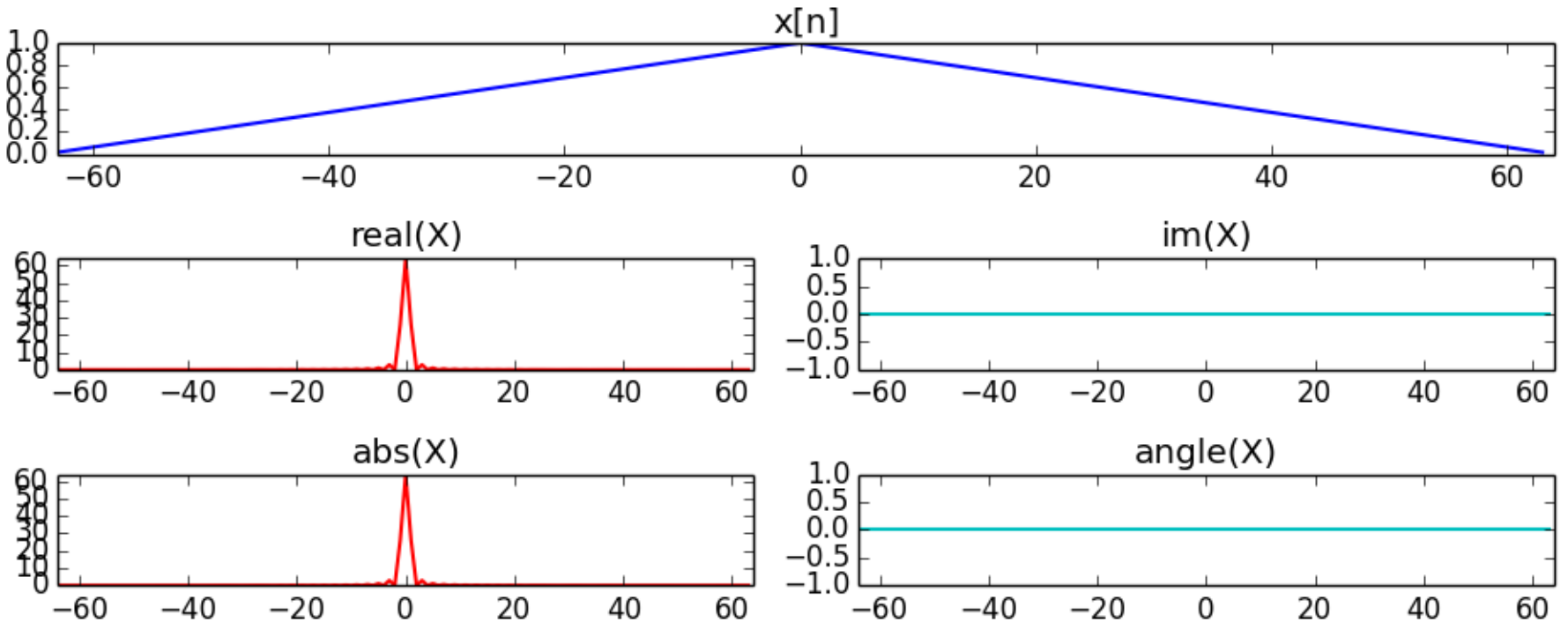
Shift: $x[n-n_0] \Leftrightarrow e^{-j2\pi k n_0/N} X[k]$

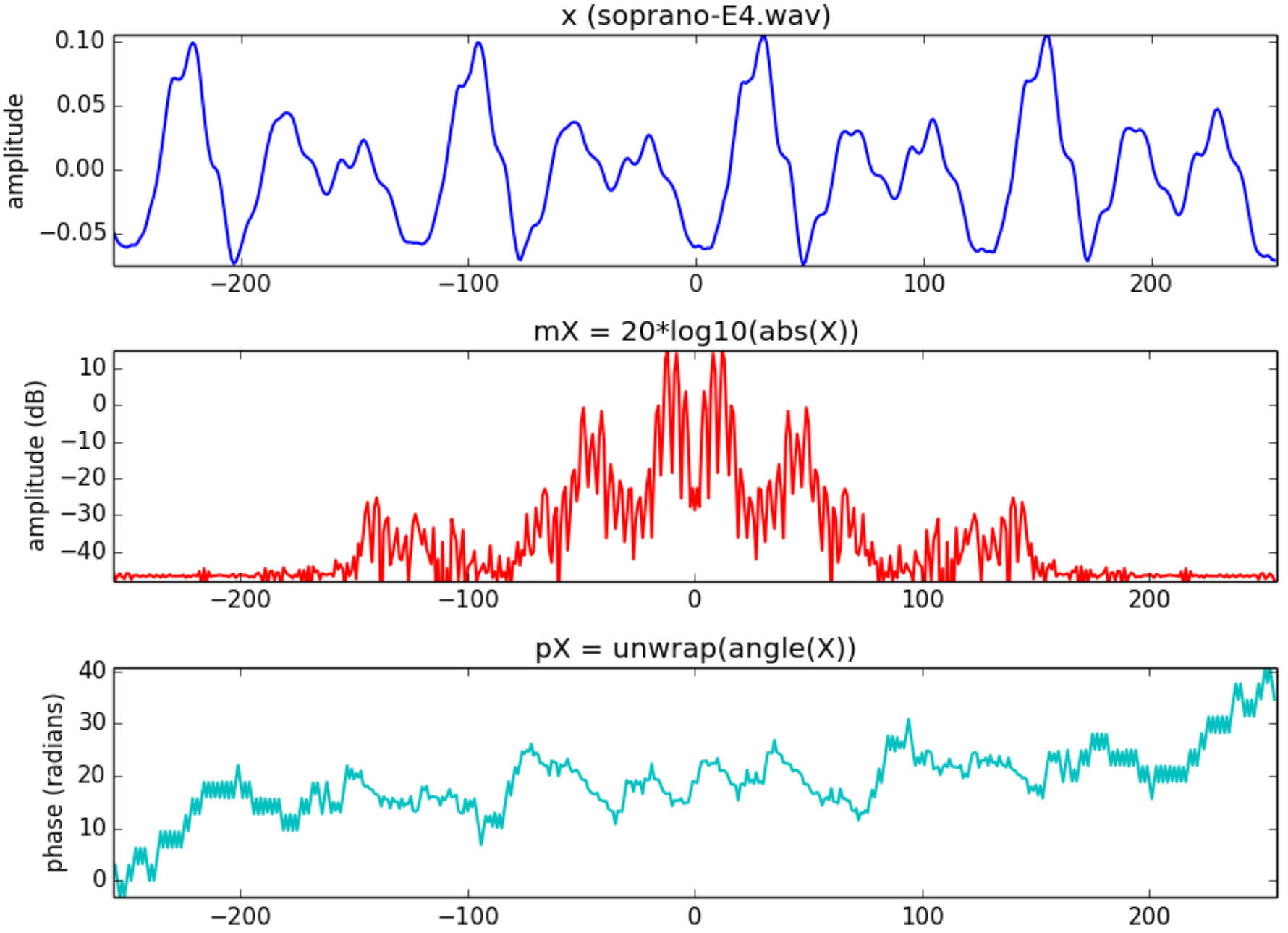
$$
\begin{aligned}
&DFT\left(x[n-n_0]\right) \\
&\quad = \sum_{n=0}^{N-1} x[n-n_0] e^{-j2\pi kn/N} \\
&\quad = \sum_{m=-n_0}^{N-1-n_0} x[m] e^{-j2\pi k(m+n_0)/N} \qquad (m = n-n_0) \\
&\quad = \sum_{m=0}^{N-1} x[m] e^{-j2\pi km/N} e^{-j2\pi kn_0/N} \\
&\quad = e^{-j2\pi kn_0/N} \sum_{m=0}^{N-1} x[m] e^{-j2\pi km/N} \\
&\quad = e^{-j2\pi kn_0/N} X[k]
\end{aligned}
$$


x1=x[n]
x2=x[n-2]
mX1
mX2
pX1
pX2

Symmetry:

$$x[n]\,real \;\Leftrightarrow\; \Re\{X[k]\}\,even \;\text{ and }\; \Im\{X[k]\}\,odd$$
$$\Leftrightarrow\; |X[k]|\,even \;\text{ and }\; \angle X[k]\,odd$$
$$x[n]\,real \text{ and } even \;\Leftrightarrow\; \Re\{X[k]\}\,even \;\text{ and }\; \Im\{X[k]\}=0$$
$$\Leftrightarrow\; |X[k]|\,even \;\text{ and }\; \angle X[k]=n\pi$$




x (soprano-E4.wav)
amplitude
0.10
0.05
0.00
−0.05
−200
−100
0
100
200
mX = 20*log10(abs(X))
amplitude (dB)
10
0
−10
−20
−30
−40
−200
−100
0
100
200
pX = unwrap(angle(X))
phase (radians)
40
30
20
10
0
−200
−100
0
100
200

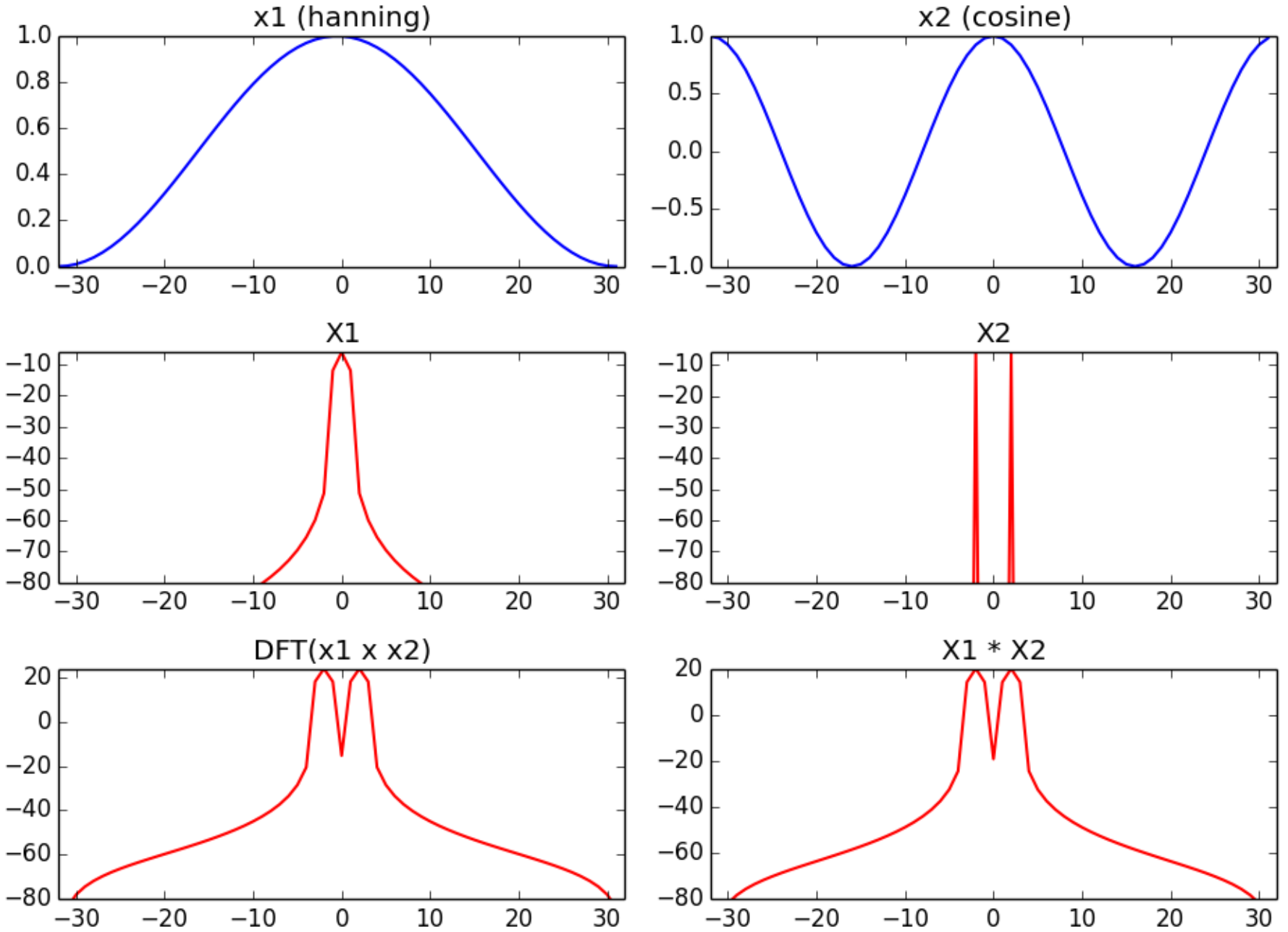
Phase unwrap

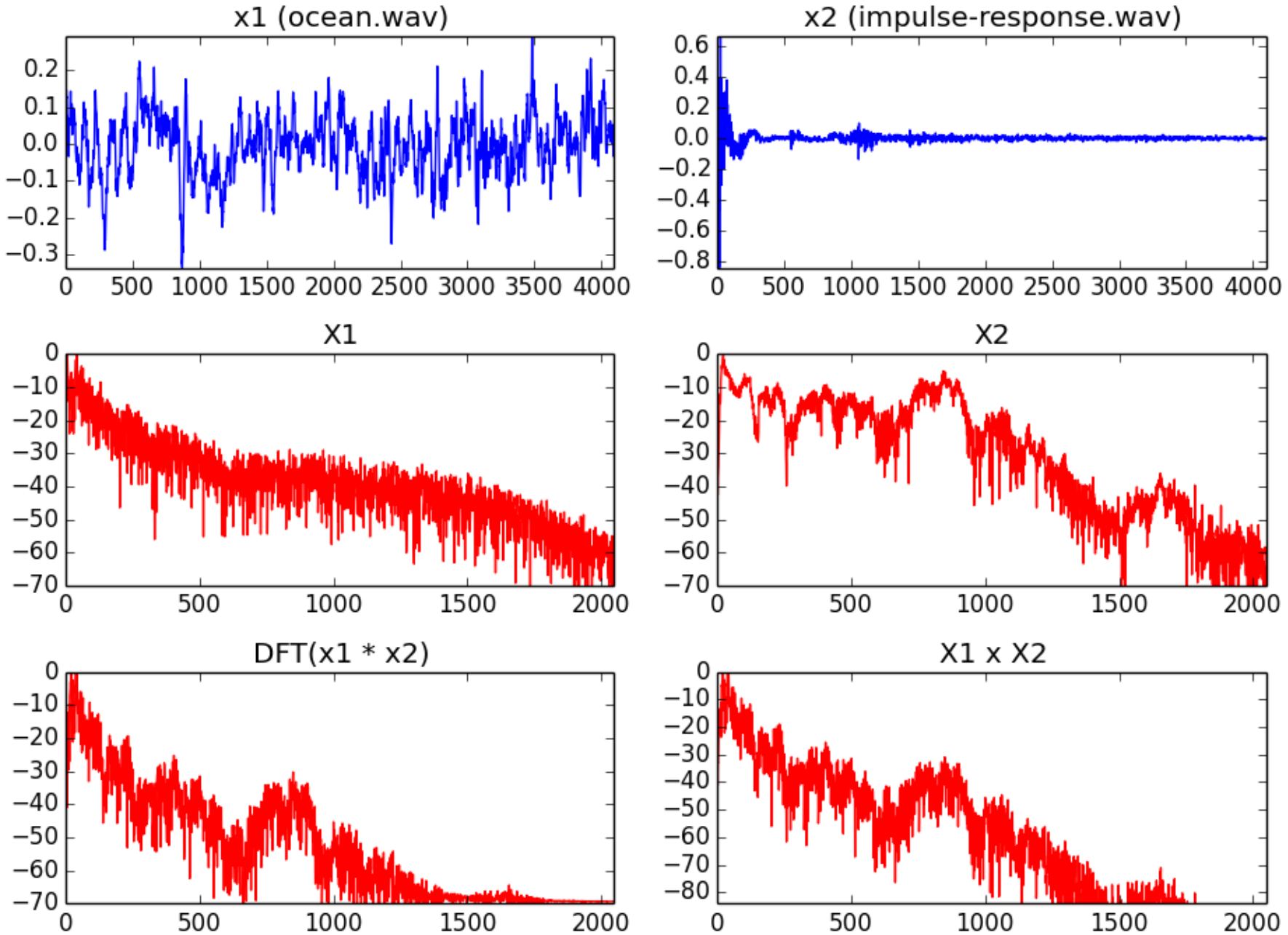
- Phase unwrapping ensures that all appropriate multiples of 2π have been included in phase response $\Theta(\omega)$
- Numpy: Unwrap by changing deltas between values to 2*pi complement.

```
phase = np.linspace(0, np.pi, num=5)
phase_wrap = np.copy(phase)
phase_wrap[3:] += np.pi

phase
array([0.        , 0.78539816, 1.57079633, 2.35619449, 3.14159265])
phase_wrap
array([ 0.        , 0.78539816, 1.57079633, 5.49778714, 6.28318531])
np.unwrap(phase)
array([ 0.        , 0.78539816, 1.57079633, -0.78539816, 0.        ])
```

Convolution: $x_1[n] * x_2[n] \Leftrightarrow X_1[k] \times X_2[k]$

$$
\begin{aligned}
& DFT\left(x_1[n] * x_2[n]\right) \\
&= \sum_{n=0}^{N-1} \left(x_1[n] * x_2[n]\right) e^{-j 2\pi kn/N} \\
&= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x_1[m] x_2[n-m] e^{-j 2\pi kn/N} \\
&= \sum_{m=0}^{N-1} x_1[m] \sum_{n=0}^{N-1} x_2[n-m] e^{-j 2\pi kn/N} \\
&= \left(\sum_{m=0}^{N-1} x_1[m] e^{-j 2\pi km/N} \right) X_2[k] \\
&= X_1[k] \times X_2[k]
\end{aligned}
$$


x1 (hanning)
x2 (cosine)
X1
X2
DFT(x1 x x2)
X1 * X2

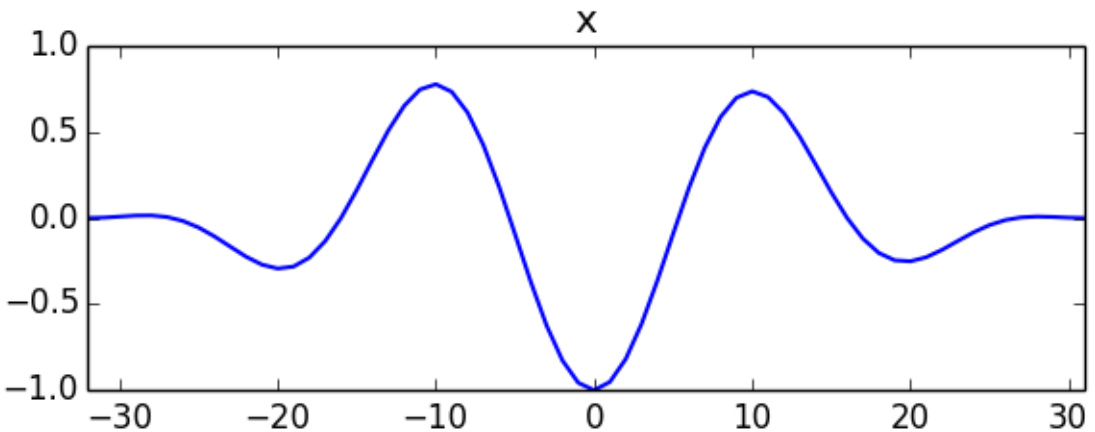

x1 (ocean.wav)
x2 (impulse-response.wav)
X1
X2
DFT(x1 * x2)
X1 x X2

Index Fourier Properties II

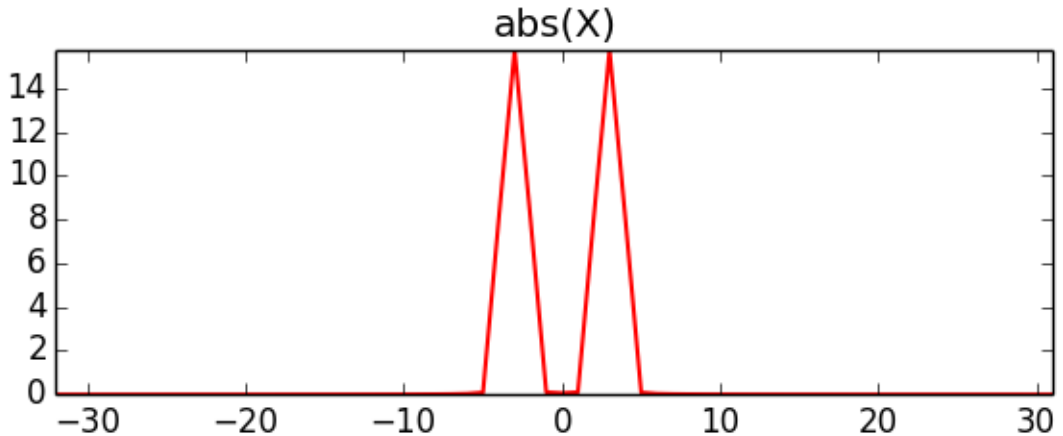
- Energy conservation & decibels
- Phase unwrapping
- Zero padding
- Fast Fourier Transform (FFT)
- FFT and zero-phase windowing
- Analysis/synthesis

Energy conservation

$$\sum_{n=-N/2}^{N/2-1} |x[n]|^2 = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} |X[k]|^2$$



$$\sum_{n=N/2}^{N/2-1} |x[n]|^2 = 11.81182$$

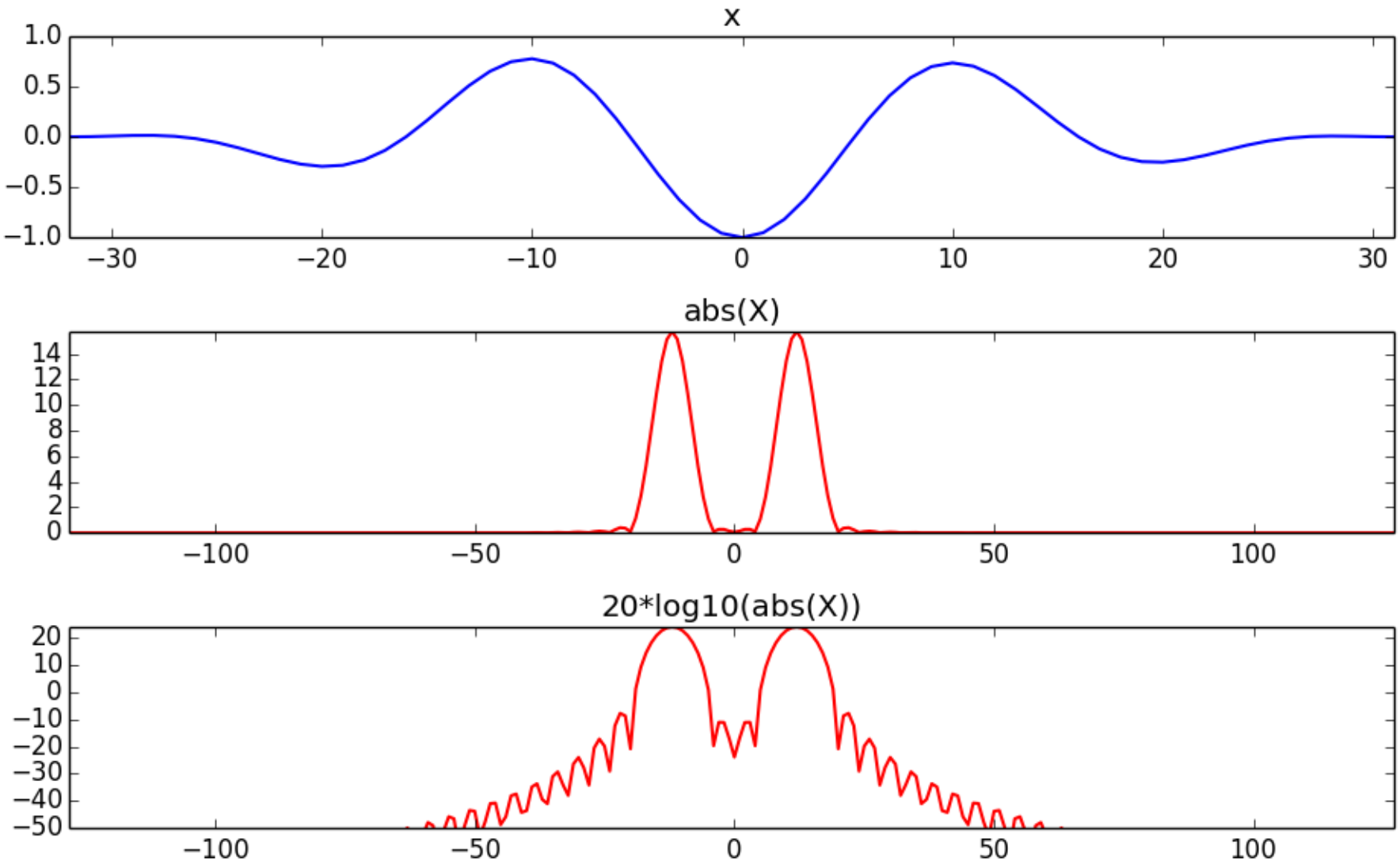


$$\frac{1}{N} \sum_{k=N/2}^{N/2-1} |X[k]|^2 = 11.81182$$

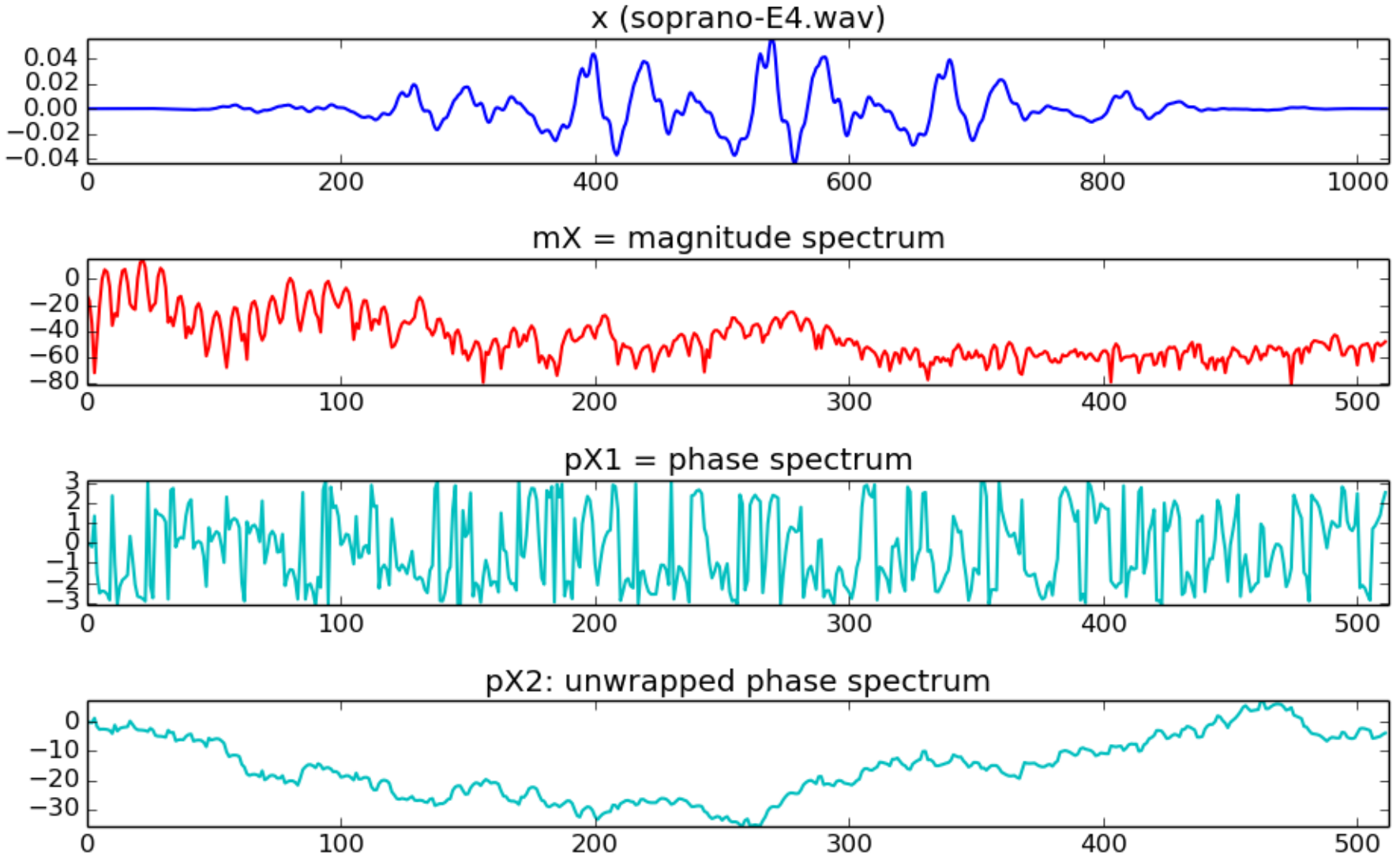
Implementation

```
# usually energy is calculated on short-time or frame-based
# it can be calculated using python list comprehension

hop_length = 256
frame_length = 512

energy = np.array([
    sum(abs(x[i:i+frame_length]**2))
    for i in range(0, len(x), hop_length)
])
```

Amplitude in decibels (dB)

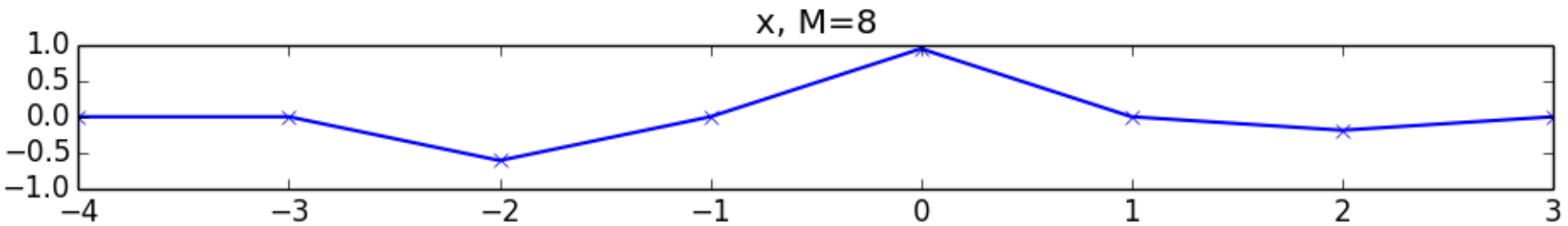


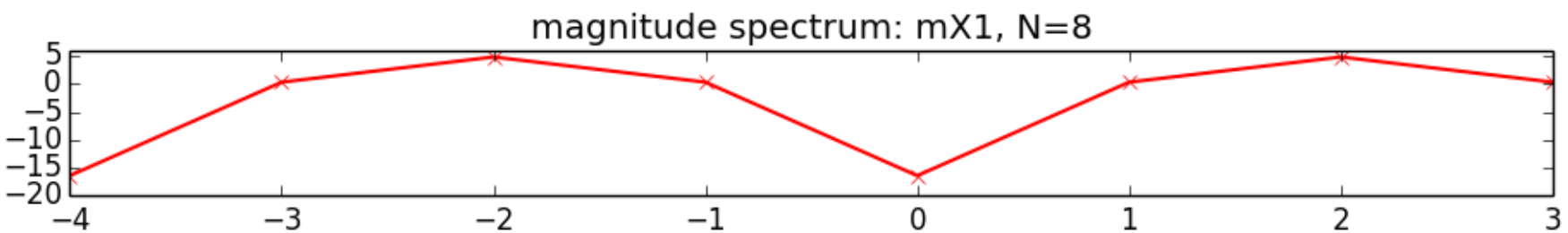
Phase unwrapping

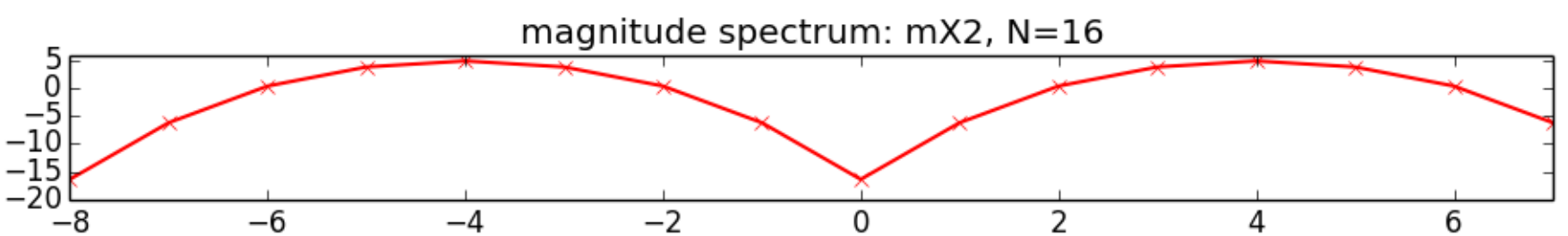


Zero-padding

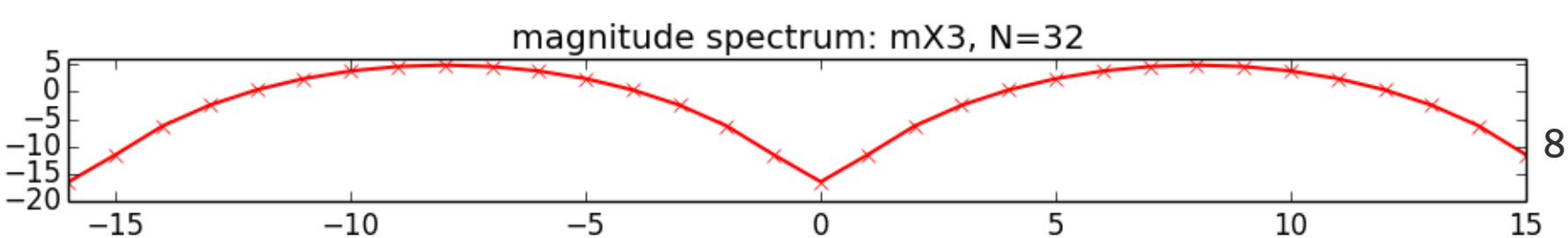
zero padding ↔ interpolation







8 + 8 zeros



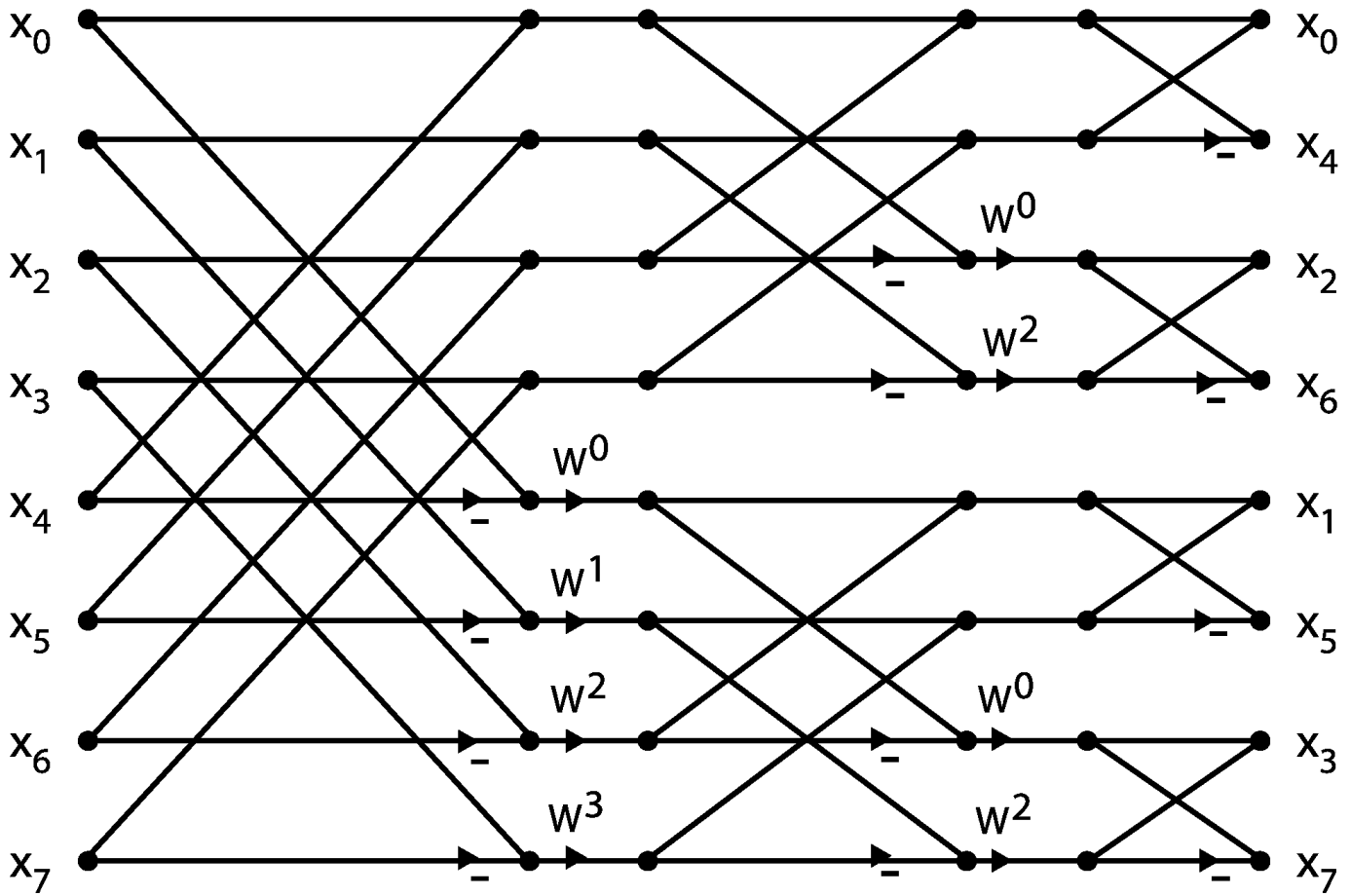
8 + 16 zeros

Implementation

```
a = np.array([0, 1, 2, 3])

# using a.resize()
a.resize(8, refcheck=False)

# output
array([0, 1, 2, 3, 0, 0, 0, 0])

# using np.pad
np.pad(a, (0, 4))

# using tensorflow
sequence = [[1], [2, 3], [4, 5, 6]]
tf.keras.preprocessing.sequence.pad_sequences(sequence)

# output
array([[0, 0, 1],
       [0, 2, 3],
       [4, 5, 6]], dtype=int32)
```

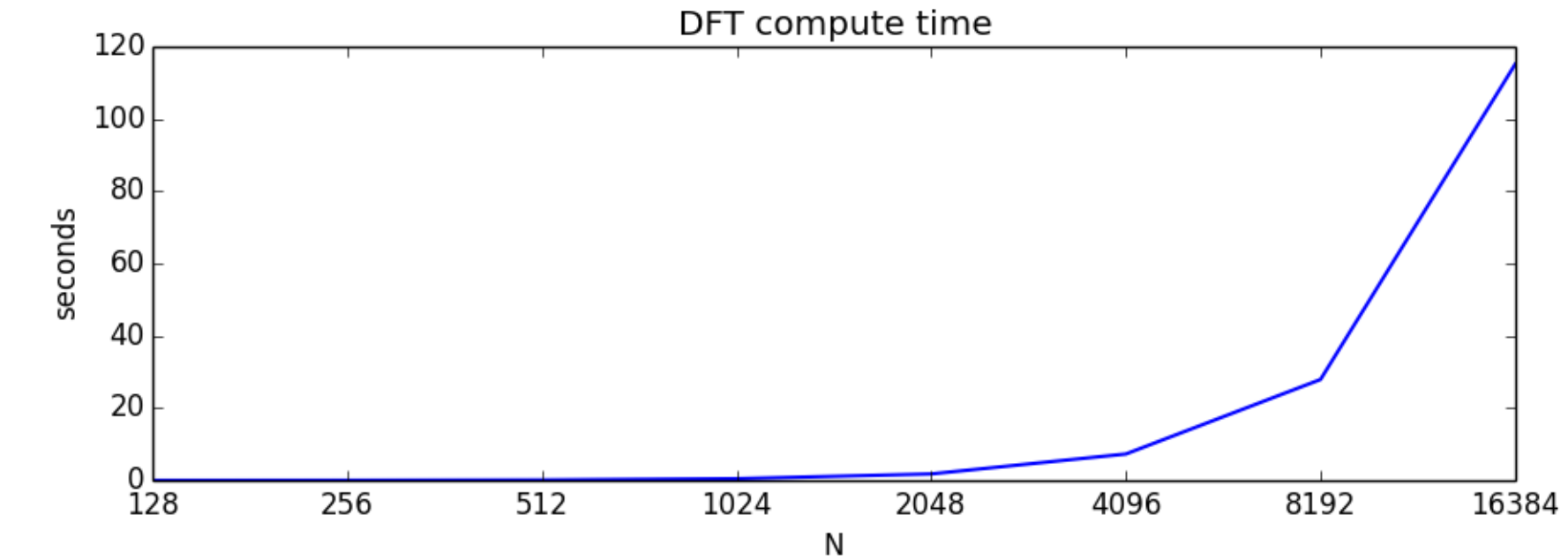
Fast Fourier Transform

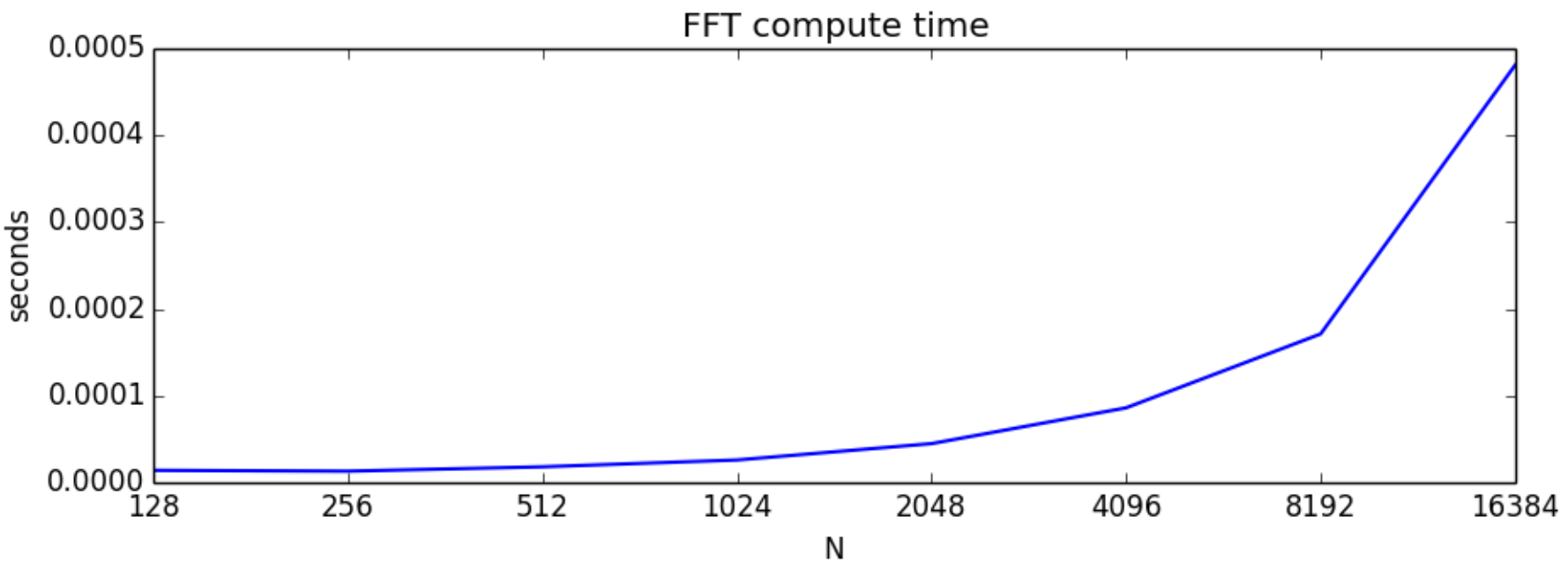
Cooley-Tukey algorithm: breaks down recursively the DFT of a power of 2 size into two pieces of size N/2.



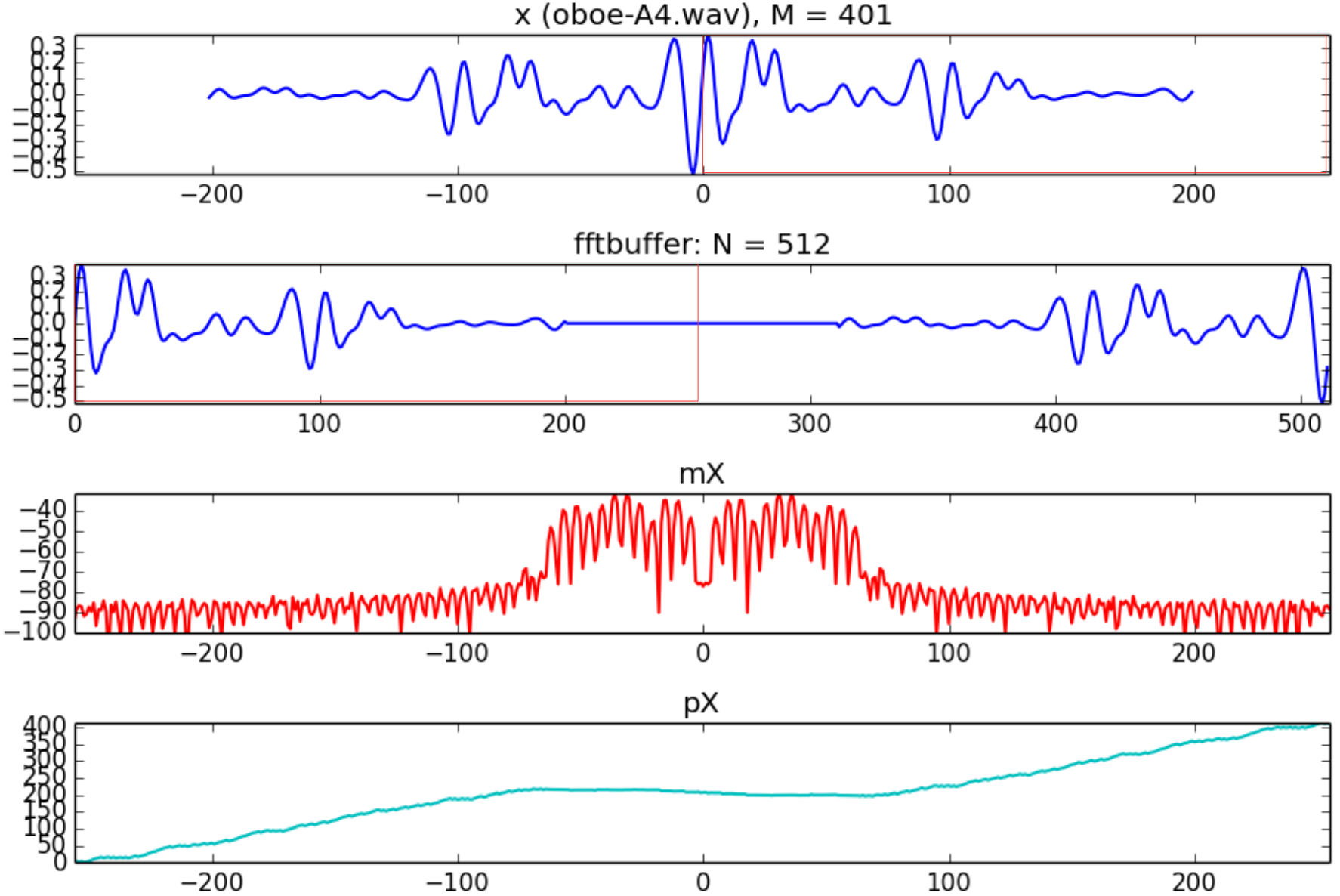
Implementation

```
a = np.array([1, 2, 3, 4])
np.fft.fft(a)
```

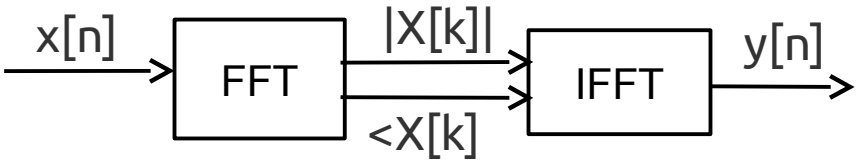

DFT compute time
seconds
120
100
80
60
40
20
0
128
256
512
1024
2048
4096
8192
16384
N

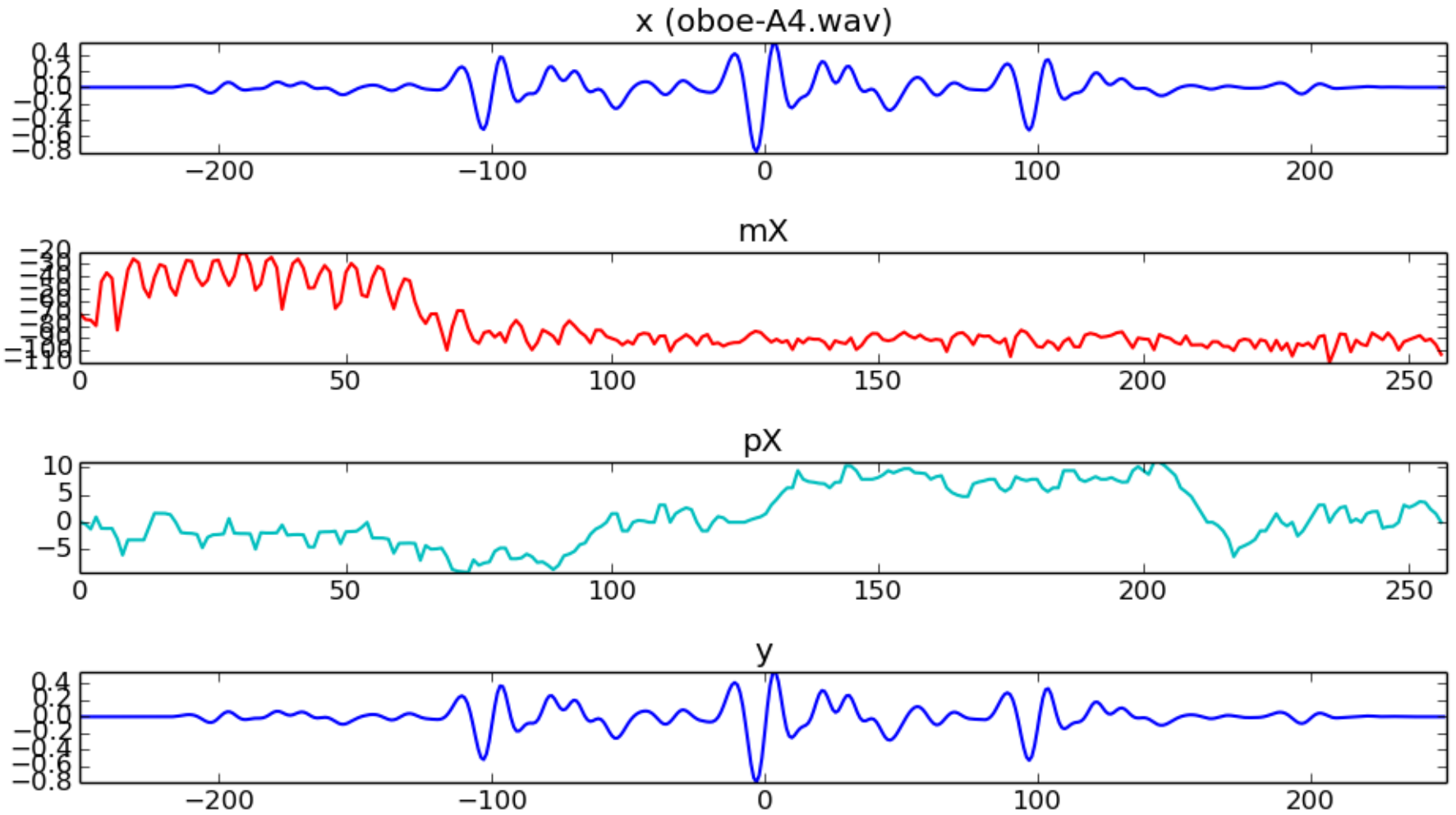

FFT compute time
seconds
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
128
256
512
1024
2048
4096
8192
16384
N

FFT and zero-phase windowing



Analysis/synthesis





Practice Session with Python

- sms-tools/lectures/03-Fourier-properties/plots-code
- sms-tools/workspace
- github.com/bagustris/python-for-signal-processing > notebook > frequency_resolution
- github.com/bagustris/python-for-signal-processing > notebook > more_fourier_transform

Final Project (Deadline Fri 7/1)

Signal Processing S2 TF ITS 2021

- Each students prepare one reference paper related to his/her research. IEEExplore is the preferred source.
- He/she demonstrates signal processing aspect of the paper/research: how to obtain data, conduct experiment, and visualize the results (plot/table).
- Submit via Teams: original paper (**.pdf**) + your (review) paper (**.pdf**) + presentation (**.pdf**) + codes (**.c/.m/.py/.ipynb**) in one **.zip/.rar./.gz** file.