신호처리 (FFT and STFT)

Prof. Seungchul Lee

Industrial AI Lab.

Signal Representation by Harmonic Sinusoids

- Question
  - How to decompose it to harmonic sinusoids
  - How to find the coefficients of harmonic sinusoids
  - How to find the frequency components

→ Fourier Transform by FFT

Fast Fourier Transform (FFT)

FFT with Sampling Frequency

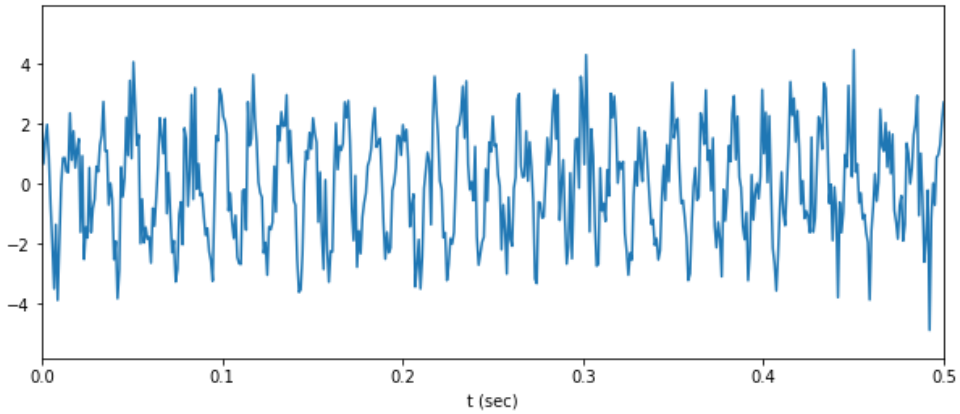
- You can think this signal as a vibration signal from a rotating machinery with 60 Hz

$$x[n] = 2\cos(2\pi 60t) \quad \Longrightarrow \quad f = 60$$

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

from scipy.fftpack import fft, fftshift
from scipy.signal import spectrogram
```

```
Fs = 2**10                # Sampling frequency
T = 1/Fs                  # Sampling period (or sampling interval)

N = 5000                  # Total data points (signal length)

t = np.arange(0, N)*T     # Time vector (time range)

k = np.arange(0, N)       # vector from 0 to N-1
f = (Fs/N)*k              # frequency range
```

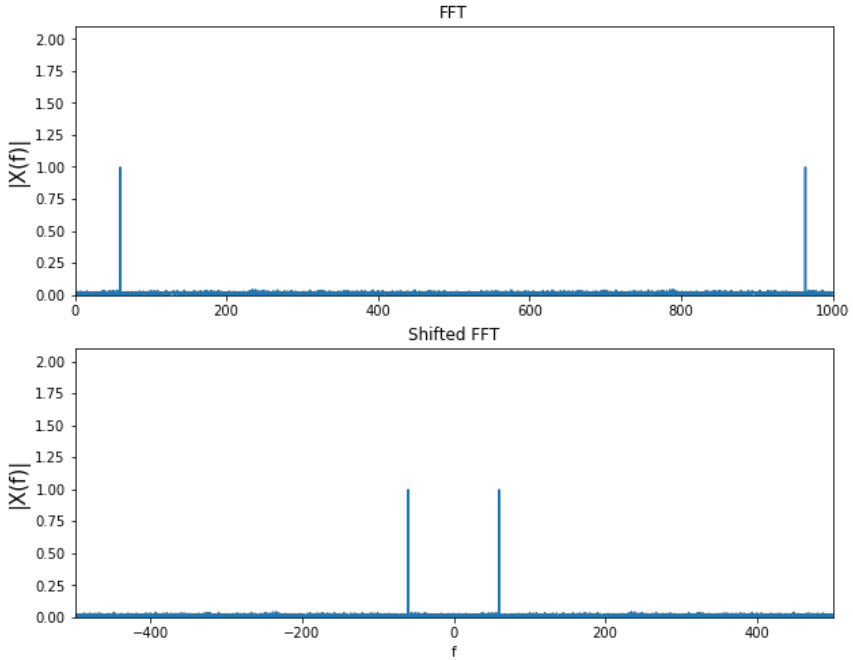
```
x = 2*np.cos(2*np.pi*60*t) + np.random.randn(N)
```



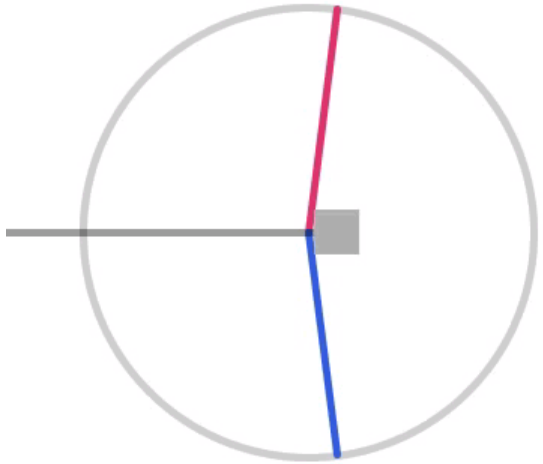
Implementing FFT Routine

$$x[n] = 2\cos(2\pi 60t) \quad \Longrightarrow \quad f = 60$$

```
# original fft

xt = fft(x)/N
xtshift = fftshift(xt)

kr = np.hstack([np.arange(0, N/2), np.arange(-N/2, 0)])
fr = (Fs/N)*kr
fs = fftshift(fr)
```



$$\cos \omega t = \frac{e^{i\omega t} + e^{-i\omega t}}{2}$$



Single-sided FFT (or Positive FFT)

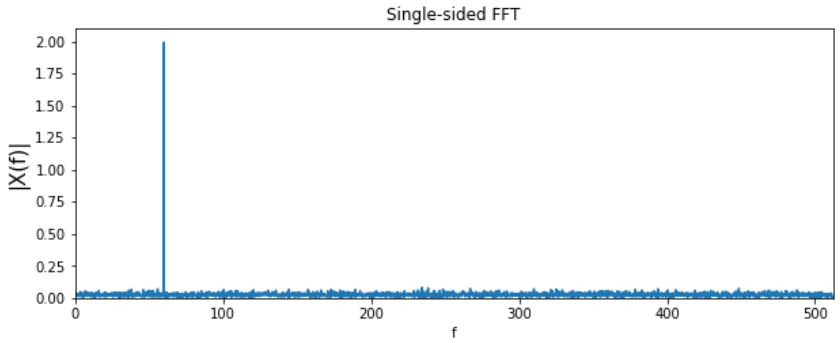
$$x[n] = 2\cos(2\pi 60t) \quad \Longrightarrow \quad f = 60$$

```
# original fft

xt = fft(x)/N
xtshift = fftshift(xt)

kr = np.hstack([np.arange(0, N/2), np.arange(-N/2, 0)])
fr = (Fs/N)*kr
fs = fftshift(fr)
```

- Only want the second half of the FFT, since the last is redundant
- 2×amplitude except the DC component

- Only want the second half of the FFT, since the last is redundant
- 2×amplitude except the DC component

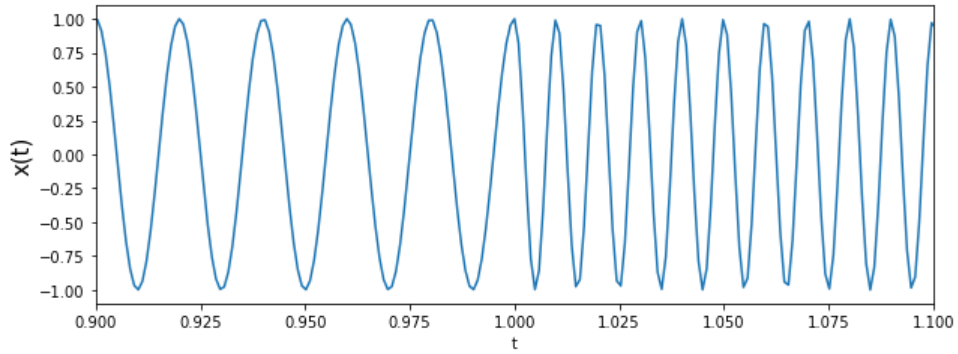

```
# single-sides fft

xt = fft(x)/N
xtss = xt[0:int(N/2)+1]       # 0:N/2
xtss[1:-1] = 2*xtss[1:-1]

fss = f[0:int(N/2)+1]
```



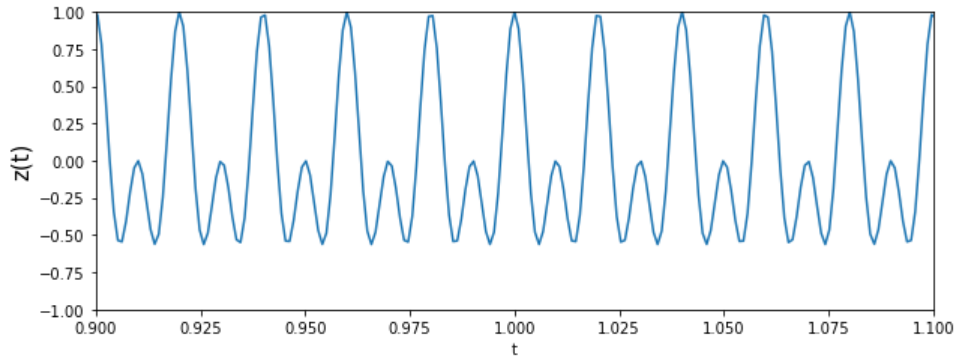
STFT (Short-Time Fourier Transformation)

Issue on FFT

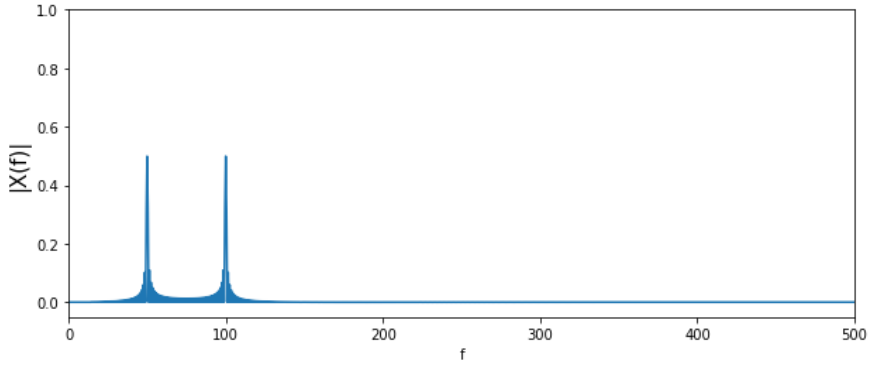
```
x1 = np.cos(2*np.pi*50*t)
x2 = np.cos(2*np.pi*100*t)

x = np.zeros(t.shape)
x[0:int(N/2)] = x1[0:int(N/2)]
x[int(N/2):-1] = x2[int(N/2):-1]
```

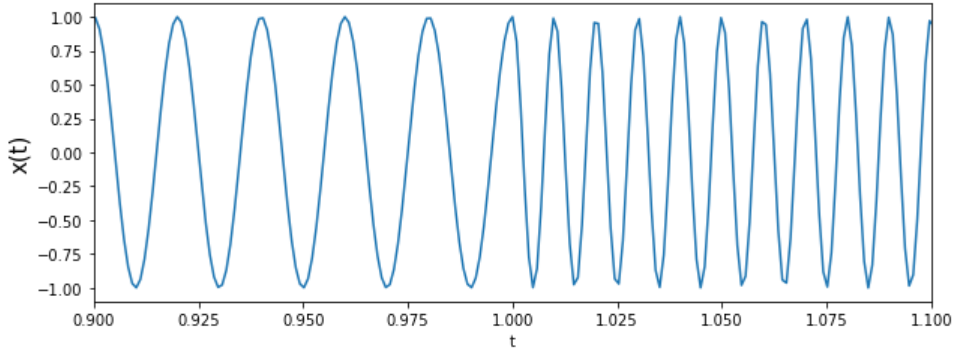


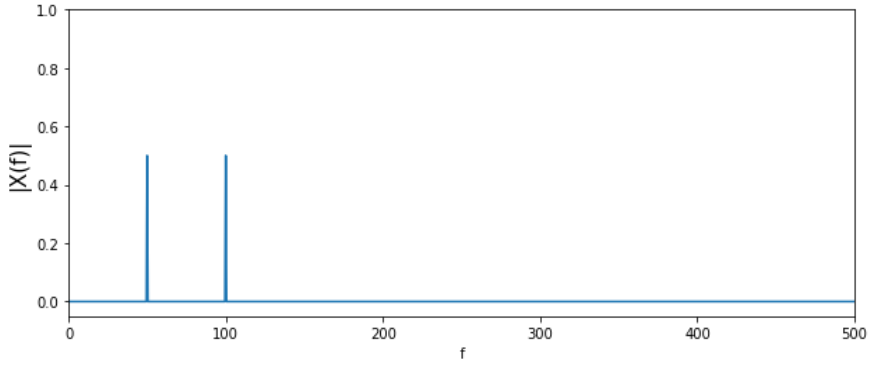

```
z = 1/2*(x1 + x2)
```

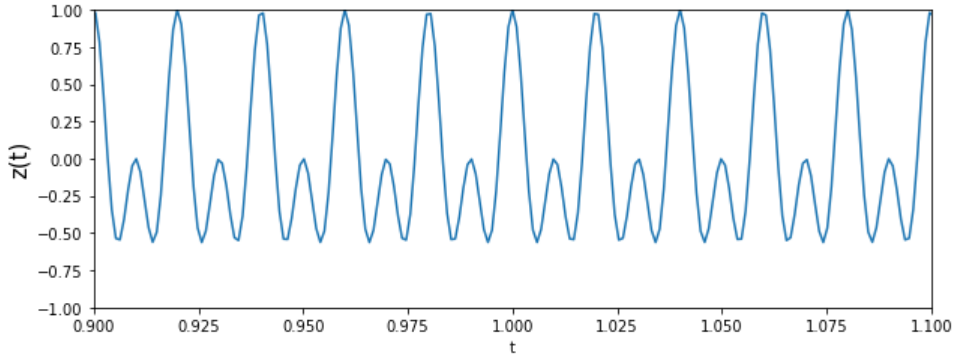


Issue on FFT









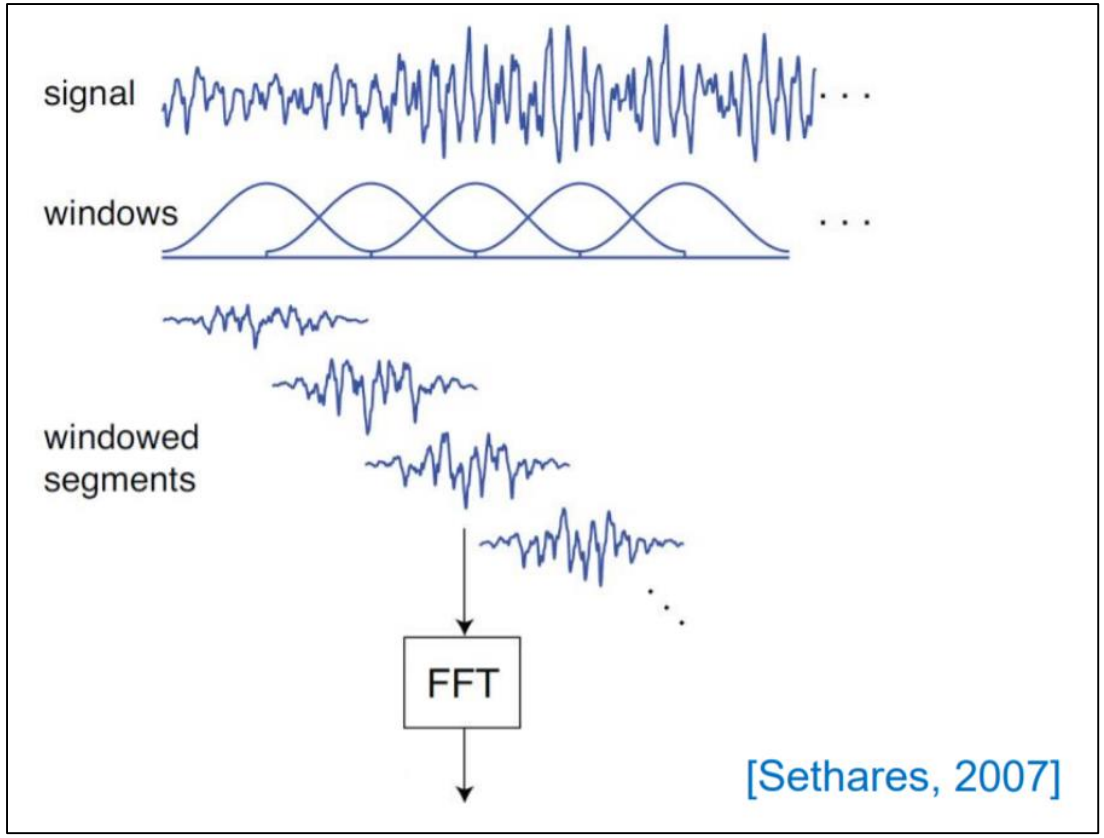
Short-Time Fourier Transformation

- The spectral content of speech changes over time (non-stationary)
  - As an example, formants change as a function of the spoken phonemes
  - Applying the DFT over a long window does not reveal transitions in spectral content

- To avoid this issue, we apply the DFT over short periods of time
  - For short enough windows, speech can be considered to be stationary
  - Remember, though, that there is a time-frequency tradeoff here

Short-Time Fourier Transformation

- Define analysis window
- Define the amount of overlap between windows
  - e.g., 30%
- Define a windowing function
  - e.g., Hann, Gaussian
- Generate windowed segments
  - Multiply signal by windowing function
- Apply the FFT to each windowed segment

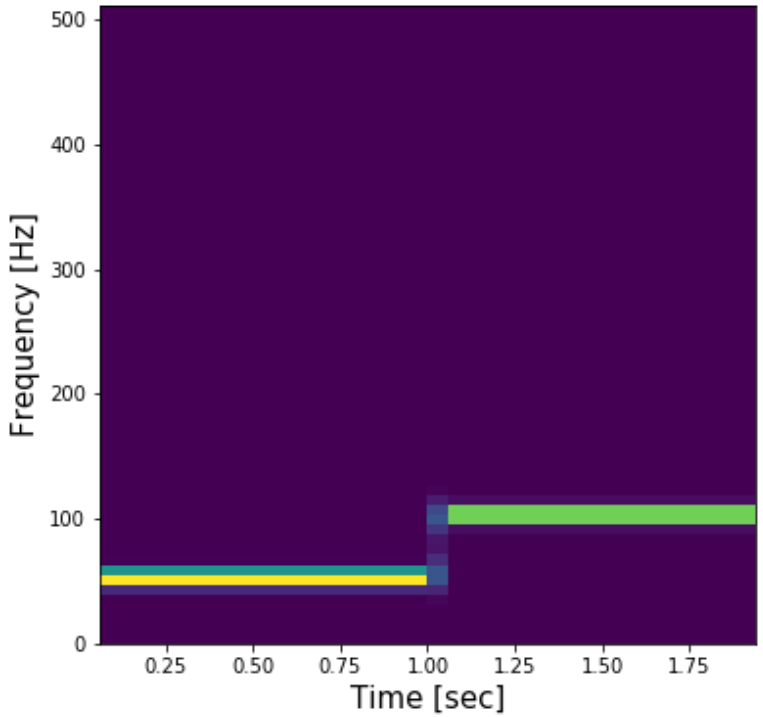


STFT in Python

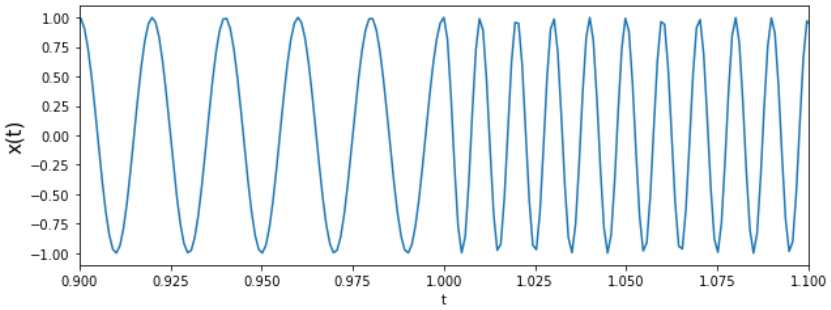
- Python function ‘spectrogram’ is useful for easily computing STFTs

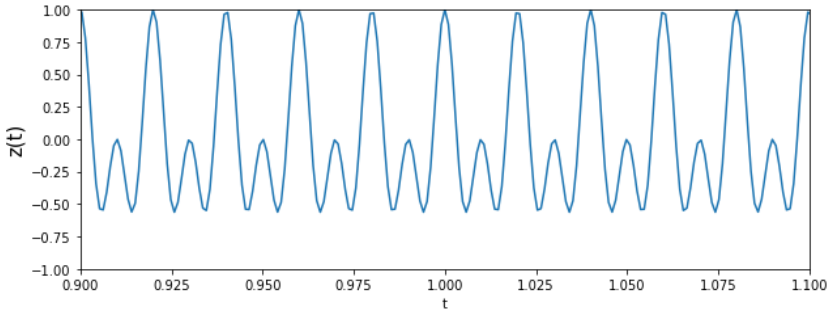
```
windowsize = 2**7
window = np.hanning(windowsize)
nfft = windowsize
noverlap = windowsize/2
```

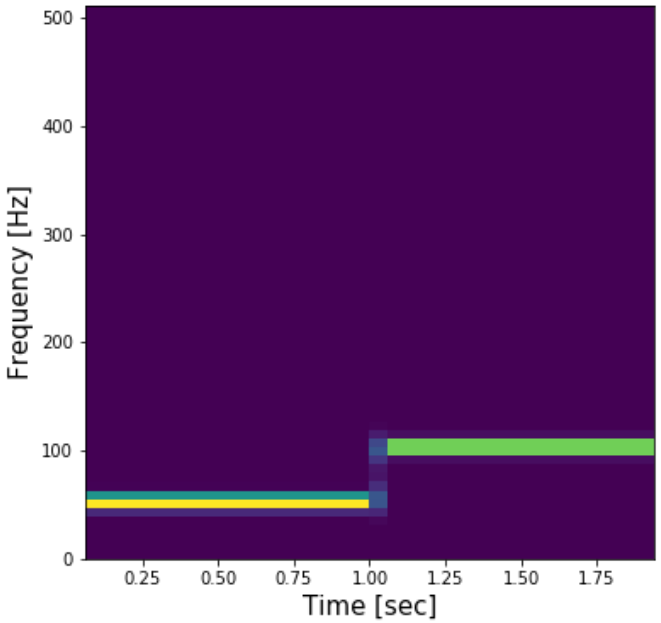
```
f, t, Sxx = spectrogram(x,
                        Fs,
                        window = window,
                        noverlap = noverlap,
                        nfft = nfft,
                        scaling = 'spectrum',
                        mode = 'psd')
```

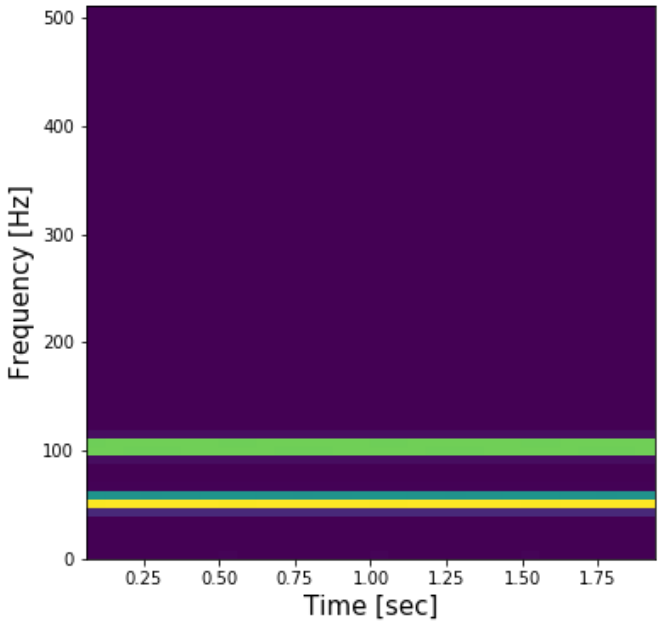


STFT: Time and Frequency

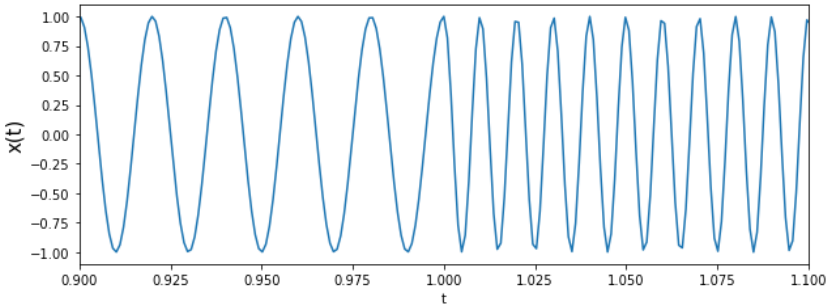


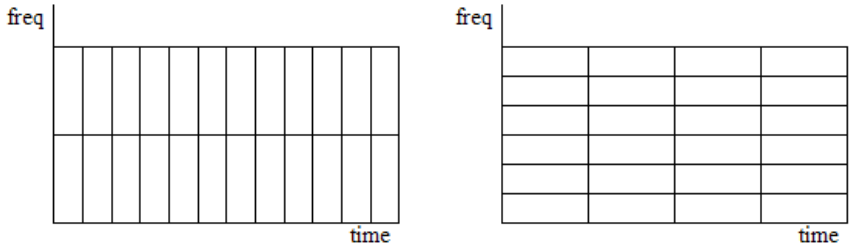




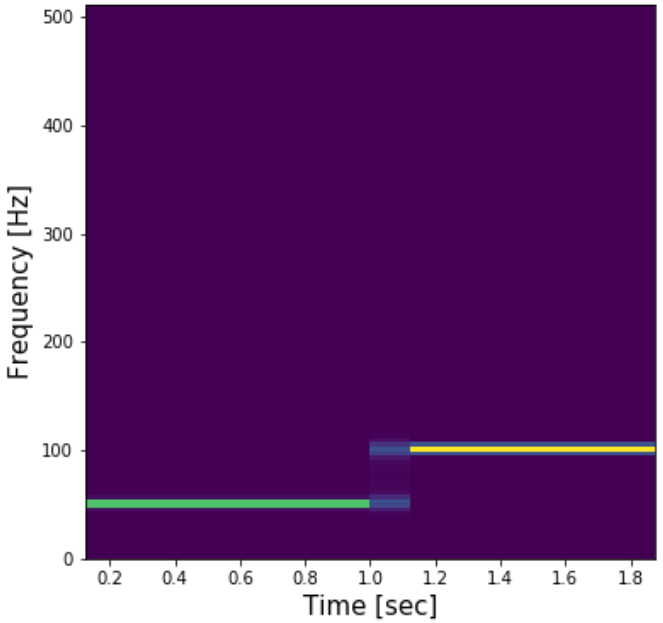


STFT Resolution Trade-off






```
windowsize = 2**8
```




```
windowsize = 2**6
```