lab_1

March 19, 2020

```
In [10]: %matplotlib inline
```

```
In [ ]: #
        import numpy as np
        import matplotlib.pyplot as plt

        x_1 = np.arange(20)

        #
        print('positive correlation',
              np.correlate(x_1, x_1))

        #
        print('negative correlation',
              np.correlate(x_1, -1*x_1))

        #
        np.random.seed(1)
        x_1 = np.random.randint(-10, 10, 20)
        x_2 = np.random.randint(-10, 10, 20)
        print('zero correlation',
              np.correlate(x_1, x_2))

        plt.figure(figsize=(14, 5))
        plt.xlabel("Sample")
        plt.ylabel("Amplitude")
        plt.plot(x_1)
        plt.plot(x_2)
        plt.grid(True)
```

```
In [ ]: #   -
        import numpy as np
        import matplotlib.pyplot as plt

        x_1 = np.random.randint(-5, 5, 20)
        delta_function = np.zeros(20)
        delta_function[7] = 1
        print('correlation', np.correlate(x_1, delta_function))
```

```
plt.figure(figsize=(14, 5))
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(x_1)
plt.plot(delta_function)
plt.grid(True)
```

In [ ]:
```
#
import numpy as np
import matplotlib.pyplot as plt

x = np.zeros(15)
x[5:10] = 1
acf_x = np.correlate(x, x, mode='same')
conv_same_x = np.convolve(x, x, mode='same')
conv_full_x = np.convolve(x, x, mode='full')

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(x)
plt.show()

plt.figure(figsize=(14, 5))
plt.title('AKF')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(acf_x)
plt.show()

plt.figure(figsize=(14, 5))
plt.title('Same convolve')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(conv_same_x)
plt.show()

plt.figure(figsize=(14, 5))
plt.title('Full convolve')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(conv_full_x)
plt.show()
```

In [ ]:
```
#
import numpy as np
```

```
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 32
frequency =4
t = np.arange(samplerate)
amplitude = 1
phase = 5
data_1 = amplitude * np.sin(2. *
                            np.pi * frequency *
                            t/samplerate + phase)
samplerate = 32
frequency = 8
t = np.arange(samplerate)
amplitude = 1 / 2
phase = 5
data_2 = amplitude * np.sin(2. *
                            np.pi * frequency *
                            t/samplerate + phase)

data = data_1 #+ data_2

plt.figure(figsize=(14, 5))
plt.title('Phase')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(data)
plt.show()

fft_data = fft.fft(data)
m_fft_data = np.abs(fft_data)
p_fft_data = np.angle(fft_data)
restores_data = fft.ifft(fft_data).real
plt.figure(figsize=(14, 5))
plt.title('Spectrum')
plt.xlabel("Amplitude freq samples")
plt.ylabel("Level")
plt.stem(m_fft_data)
plt.show()

plt.figure(figsize=(14, 5))
plt.title('Phase')
plt.xlabel("Phase freq samples")
plt.ylabel("Level")
plt.plot(p_fft_data)
plt.show()
```

```
plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(restores_data)
plt.show()
```

In [ ]:
```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 32
frequency =4
t = np.arange(samplerate)
amplitude = 1
phase = 0
data_1 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

samplerate = 32
frequency = 8
t = np.arange(samplerate)
amplitude = 1 / 2
phase = 10
data_2 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

data = data_1 + data_2

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(data)
plt.show()

fft_data = fft.fft(data)
m_fft_data = np.abs(fft_data)
p_fft_data = np.angle(fft_data)
restores_data = fft.ifft(fft_data).real
plt.figure(figsize=(14, 5))
plt.title('Spectrum')
plt.xlabel("Amplitude freq samples")
plt.ylabel("Level")
plt.stem(m_fft_data)
plt.show()

plt.figure(figsize=(14, 5))
```

```
plt.title('Phase')
plt.xlabel("Phase freq samples")
plt.ylabel("Level")
plt.plot(p_fft_data)
plt.show()
```

In [ ]:
```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 32
frequency =4
t = np.arange(samplerate)
amplitude = 1
phase = 0
data_1 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

samplerate = 32
frequency = 8
t = np.arange(samplerate)
amplitude = 1 / 2
phase = 10
data_2 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

w = 10
data = data_1 + data_2
shifted_data = data * np.exp(2j*np.pi*w*(np.arange(samplerate)/samplerate))

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(data)
plt.plot(shifted_data)
plt.show()

fft_data = fft.fft(data)
m_fft_data = np.abs(fft_data)
p_fft_data = np.angle(fft_data)

fft_shifted_data = fft.fft(shifted_data)
m_fft_shifted_data = np.abs(fft_shifted_data)
p_fft_shifted_data = np.angle(fft_shifted_data)

plt.figure(figsize=(14, 5))
plt.title('Spectrum')
```

```
        plt.xlabel("Amplitude freq samples")
        plt.ylabel("Level")
        plt.stem(m_fft_data)
        plt.stem(m_fft_shifted_data,'orange')
        plt.show()

        plt.figure(figsize=(14, 5))
        plt.title('Phase')
        plt.xlabel("Phase freq samples")
        plt.ylabel("Level")
        plt.plot(p_fft_data)
        plt.plot(p_fft_shifted_data)
        plt.show()
```

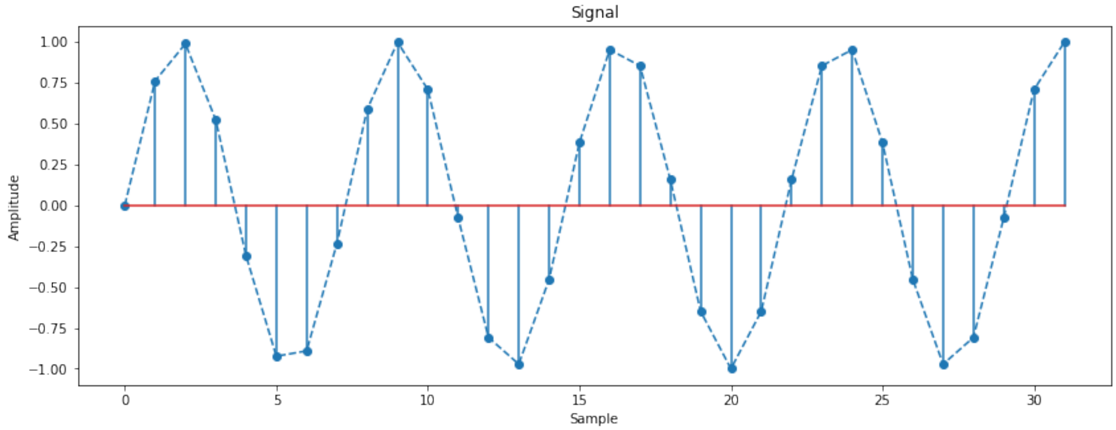
```
In [ ]: #
        import numpy as np
        from numpy import fft
        from scipy.io.wavfile import write
        from matplotlib import pyplot as plt

        samplerate = 32
        frequency =4.4
        t = np.arange(samplerate)
        amplitude = 1
        phase = 0
        data = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

        plt.figure(figsize=(14, 5))
        plt.title('Signal')
        plt.xlabel("Sample")
        plt.ylabel("Amplitude")
        plt.plot(data, linestyle='--')
        plt.stem(data)

        plt.show()

        fft_data = fft.fft(data)
        m_fft_data = np.abs(fft_data)

        plt.figure(figsize=(14, 5))
        plt.title('Spectrum')
        plt.xlabel("Freq samples")
        plt.ylabel("Level")
        plt.stem(m_fft_data)
        plt.show()
```

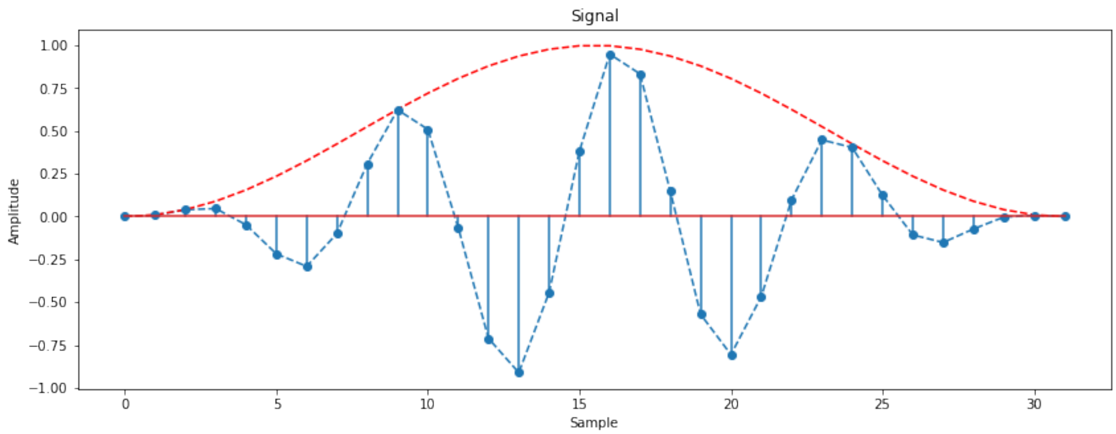
```
In [ ]: #
        import numpy as np
```

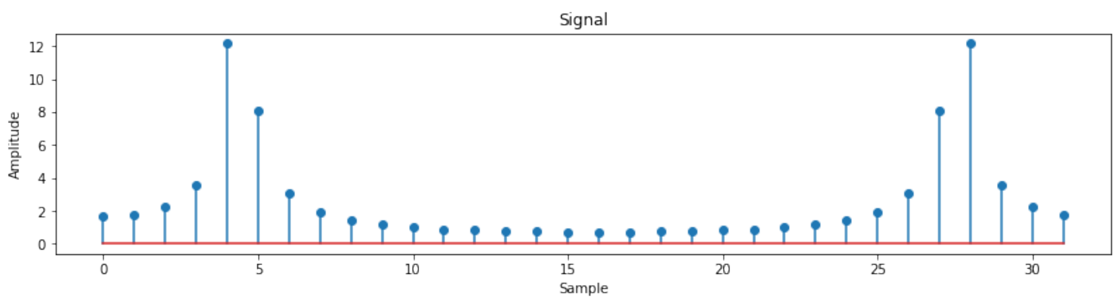
```
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 32
frequency =4
t = np.arange(samplerate)
amplitude = 1
phase = 0
data = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

rectangular_window = np.ones(samplerate)
plt.figure(figsize=(14, 5))
plt.title('rectangular_window')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(rectangular_window)
plt.show()

hanning_window = np.hanning(samplerate)
plt.figure(figsize=(14, 5))
plt.title('hanning_window')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(hanning_window)
plt.show()

hamming_window = np.hamming(samplerate)
plt.figure(figsize=(14, 5))
plt.title('hamming_window')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(hamming_window)
plt.show()
```

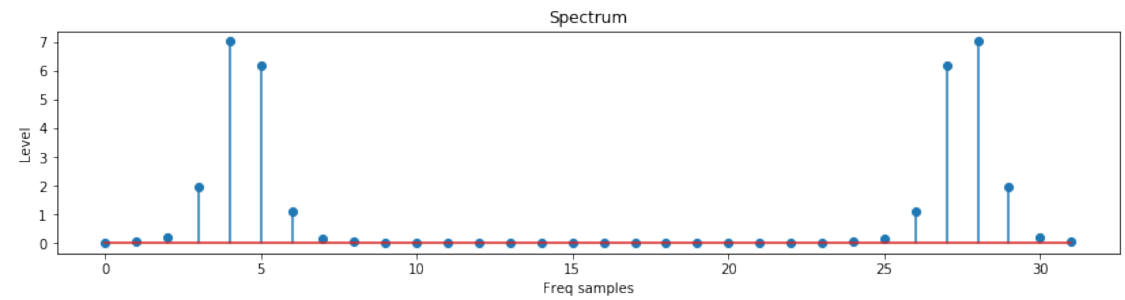
```
In [4]: #
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 32
frequency =4.4
t = np.arange(samplerate)
amplitude = 1
phase = 0

data = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)
```

```
window = np.hanning(samplerate)

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(data, linestyle='--')
plt.stem(data)
plt.show()

windowed_data = data * window

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(windowed_data, linestyle='--')
plt.plot(window,linestyle='--', color='red')
plt.stem(windowed_data)
plt.show()


fft_data = fft.fft(data)
m_fft_data = np.abs(fft_data)
plt.figure(figsize=(14, 3))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(m_fft_data)
plt.show()

fft_windowed_data = fft.fft(windowed_data)
m_fft_windowed_data= np.abs(fft_windowed_data)
plt.figure(figsize=(14, 3))
plt.title('Spectrum')
plt.xlabel("Freq samples")
plt.ylabel("Level")
plt.stem(m_fft_windowed_data)
plt.show()
```


Signal
Amplitude
Sample
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30


Signal
Amplitude
Sample
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30


Signal
Amplitude
Sample
12
10
8
6
4
2
0
0
5
10
15
20
25
30



0.1

: $f_d >= f_1$

:

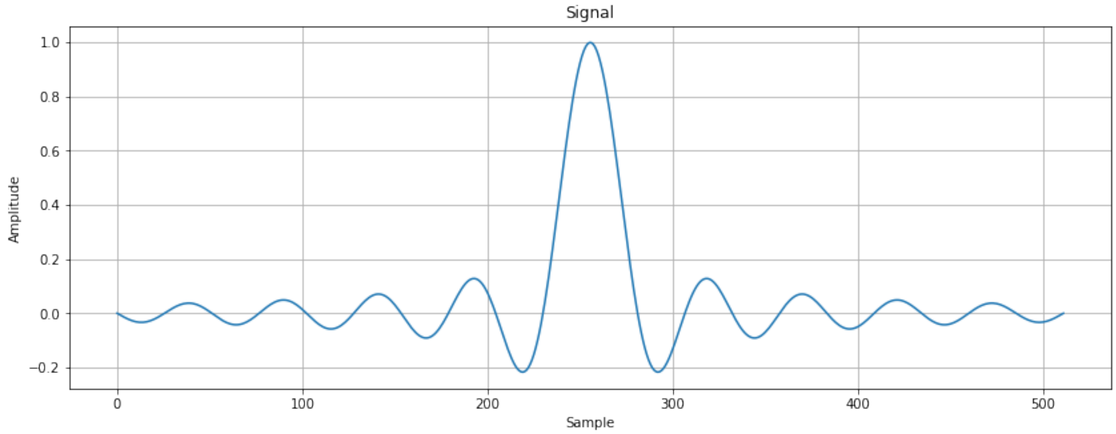
$$s(t) = \sum_{n=-\infty}^{\infty} x(n) \cdot sinc(t/T - n)$$

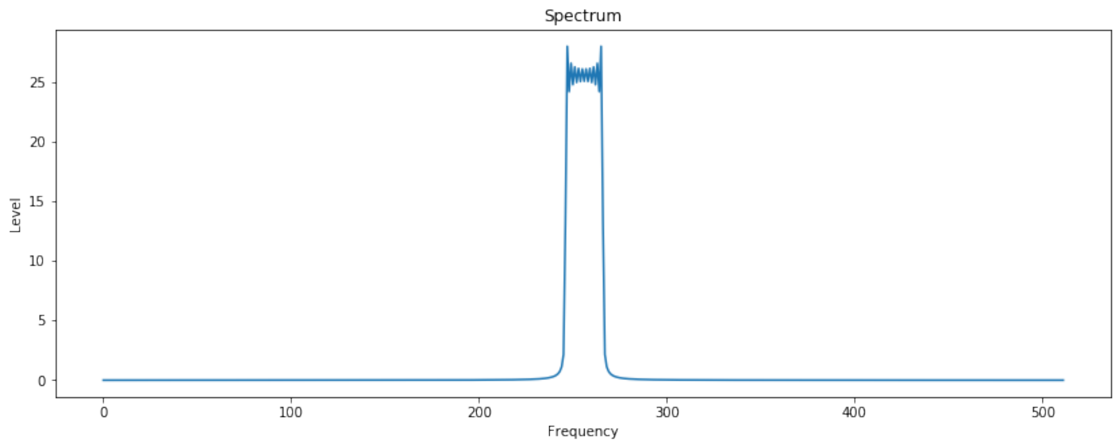
, : T-- ;

$sinc = \frac{\sin(\pi x)}{x}$

In [18]:
```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

sinc = np.sinc(np.linspace(-10,10, 512))
sinc_shifted = np.sinc(np.linspace(-10,10, 128)-1)
fft_sinc = fft.fft(sinc)

plt.figure(figsize=(14, 5))
plt.title('Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.grid()
plt.plot(sinc)
#plt.plot(sinc_shifted)

plt.figure(figsize=(14, 5))
plt.plot(fft.fftshift(np.abs(fft_sinc)))
plt.title('Spectrum')
plt.ylabel("Level")
plt.xlabel("Frequency")
```

Out[18]:
```
Text(0.5,0,'Frequency')
```

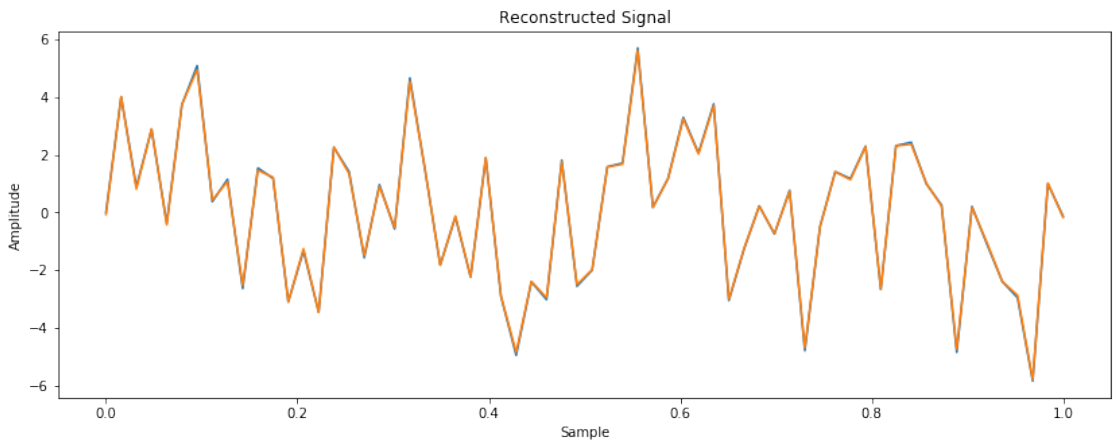





```
In [9]: #
        import numpy as np
        from numpy import fft
        from scipy.io.wavfile import write
        from matplotlib import pyplot as plt

        samplerate = 64
        t = np.arange(samplerate)
        data_1 = 1 * np.sin(1. * np.pi * 4 * t/samplerate)
        data_2 = 2 * np.sin(30. * np.pi * 6 * t/samplerate)
        data_3 = 2 * np.sin(3. * np.pi * 9 * t/samplerate)
        data_4 = 2 * np.sin(4. * np.pi * 2 * t/samplerate)

        signal = data_1 + data_2 + data_3 + data_4

        up_samplerate = 64
```

```
reconstructed_signal = np.linspace(0,1,up_samplerate)
for i, t in enumerate(reconstructed_signal):
    sinc_f = np.sinc((np.linspace(0,1,samplerate)-t)*samplerate)
    # plt.plot(sinc_f)
    reconstructed_signal[i] = np.sum(signal*sinc_f)

plt.figure(figsize=(14, 5))
plt.title('Reconstructed Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.plot(np.linspace(0,1, samplerate), signal)
plt.plot(np.linspace(0,1, up_samplerate),reconstructed_signal)
```

Out[9]: [<matplotlib.lines.Line2D at 0xb19cffa58>]



In [ ]:
```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 8
signal = np.zeros(samplerate)
signal[2:6] = 1

up_samplerate = 8
reconstructed_signal = np.linspace(0,1,up_samplerate)
for i, t in enumerate(reconstructed_signal):
    sinc_f = np.sinc((np.linspace(0,1,samplerate)-t)*samplerate)
    plt.plot(np.linspace(0,1, samplerate), sinc_f, alpha=0.5)
    reconstructed_signal[i] = np.sum(signal*sinc_f)
```

```
#plt.figure(figsize=(14, 5))
plt.title('Reconstructed Signal')
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.stem(np.linspace(0,1, samplerate), signal)
plt.plot(np.linspace(0,1, up_samplerate),reconstructed_signal, linewidth=2)
```

In [ ]:

```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 512
signal =  np.zeros(samplerate)
signal[192:320] = 1

fft_signal = fft.fft(signal)
shifted_fft_signal = fft.fftshift(np.abs(fft_signal))
f = np.linspace(-256, 256, samplerate, endpoint=True)

# Plot input signal in time domain
plt.figure(figsize=(14, 5))
plt.plot(signal)
plt.title('Input signal')
plt.xlabel('Samples')
plt.ylabel('Amplitude')

# Plot signal in freq domain
plt.figure(figsize=(14, 5))
plt.stem(f,shifted_fft_signal)
plt.title('Spectrum')
plt.xlabel('Frequency')
plt.ylabel('Level')
plt.xlim([-100, 100])
```

In [ ]:

```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 512
signal =  np.zeros(samplerate)
signal[192:320] = 1

fft_signal = fft.fft(signal)
```

```
number_of_harmonics = 128
fft_copy = fft_signal.copy()
fft_copy[number_of_harmonics:]=0
restored_signal = np.real(fft.ifft(fft_copy))
plt.figure(figsize=(14, 5))
plt.plot(restored_signal)
plt.title('Restored signal')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.grid()
```

In [ ]:
```
#
import numpy as np
from numpy import fft
from scipy.io.wavfile import write
from matplotlib import pyplot as plt

samplerate = 2048
frequency = 50
t = np.arange(samplerate)
amplitude = 1
phase = 0
data_1 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

samplerate = 2048
frequency = 200
t = np.arange(samplerate)
amplitude = 1 / 2
phase = 0
data_2 = amplitude * np.sin(2. * np.pi * frequency * t/samplerate + phase)

data = data_1 + data_2
data[500:550] = 0
plt.figure(figsize=(14, 5))
plt.plot(data)
plt.figure(figsize=(14, 5))
plt.specgram(data, NFFT=64, Fs=samplerate, noverlap=0)
plt.show()
```

In [ ]:
```
#
```

In [ ]:
```
#
import numpy as np
from numpy import fft
import matplotlib.pyplot as plt

def calculate_power(data):
    data = np.cumsum(data**2)
```

```
        data /= data[-1]
        return(data)

    x = np.random.uniform(-1, 1, 1000)
    plt.figure(figsize=(14, 5))
    plt.plot(x)
    plt.show()

    fft_x = fft.rfft(x)
    plt.figure(figsize=(14, 5))
    plt.plot(np.abs(fft_x))
    plt.show()

    plt.figure(figsize=(14, 5))
    plt.plot(calculate_power(np.abs(fft_x)))
    plt.show()
```

```
In [ ]: #
        import numpy as np
        from numpy import fft
        import matplotlib.pyplot as plt

        def calculate_power(data):
            data = np.cumsum(data**2)
            data /= data[-1]
            return(data)

        x = np.random.uniform(-1, 1, 1000)
        x = np.cumsum(x)
        x = x - x.mean()
        high, low = abs(max(x)), abs(min(x))
        x = 1 * x / max(high, low)

        plt.figure(figsize=(14, 5))
        plt.plot(x)
        plt.show()

        fft_x = fft.rfft(x)
        plt.figure(figsize=(14, 5))
        plt.plot(np.abs(fft_x))
        plt.show()

        plt.figure(figsize=(14, 5))
        plt.plot(calculate_power(np.abs(fft_x)))
        plt.show()
```

```
In [ ]: #
        import numpy as np
```

```
from numpy import fft
import matplotlib.pyplot as plt

def calculate_power(data):
    data = np.cumsum(data**2)
    data /= data[-1]
    return(data)

beta = 1 #    0  1,  0 --   ,  2 --
x = np.random.uniform(-1, 1, 1000)

fft_x = fft.rfft(x)
#
freqfft_x = fft.rfftfreq(len(x), 1/1000)
denom = freqfft_x ** (beta/2)
denom[0] = 1
fft_x = fft_x / denom

plt.figure(figsize=(14, 5))
res_x = fft.irfft(fft_x)
plt.plot(res_x)
plt.show()

plt.figure(figsize=(14, 5))
plt.plot(np.abs(fft_x))
plt.show()

plt.figure(figsize=(14, 5))
plt.plot(calculate_power(np.abs(fft_x)))
plt.show()
```