РОБОТА із ЗОБРАЖЕННЯМИ

Файл: Image_07_004

Частина 1. Фільтрація 1D сігналу (звук) у частотному просторі

Використуємо бібліотеку SCIPY

Приклад дивись

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read, write
from IPython.display import Audio
from numpy.fft import fft, ifft, fftfreq
from numpy.fft import rfft, irfft, rfftfreq # For only posetve (real) part
transform
%matplotlib inline
```

Read Audio

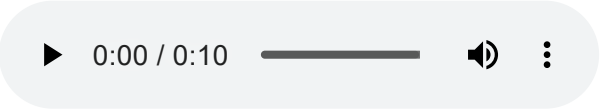
```
# Використовуйте свій аудіофайл (wav)
Fs, data_ = read ('test_audio_1.wav')
data = data_ [:,0]
print ('Частота дискретизации',Fs,' Герц')
print ('DATA SHAPE', data.shape, 'DATAE SIZE', data.size)
# print ('DATA', data [:10])
print ('MIN & MAX signl value', data.min(),'<-->',data.max() )
```

```
Частота дискретизации 44100  Герц
DATA SHAPE (470400,) DATAE SIZE 470400
MIN & MAX signl value -14604 <--> 15035
```

```
C:\Users\eab\AppData\Local\Temp\ipykernel_18760\113504413.py:1: WavFileWarning:
Chunk (non-data) not understood, skipping it.
  Fs, data_ = read ('test_audio_1.wav')
```

Play Audio

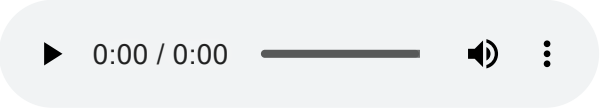
```
Audio(data, rate = Fs)
```



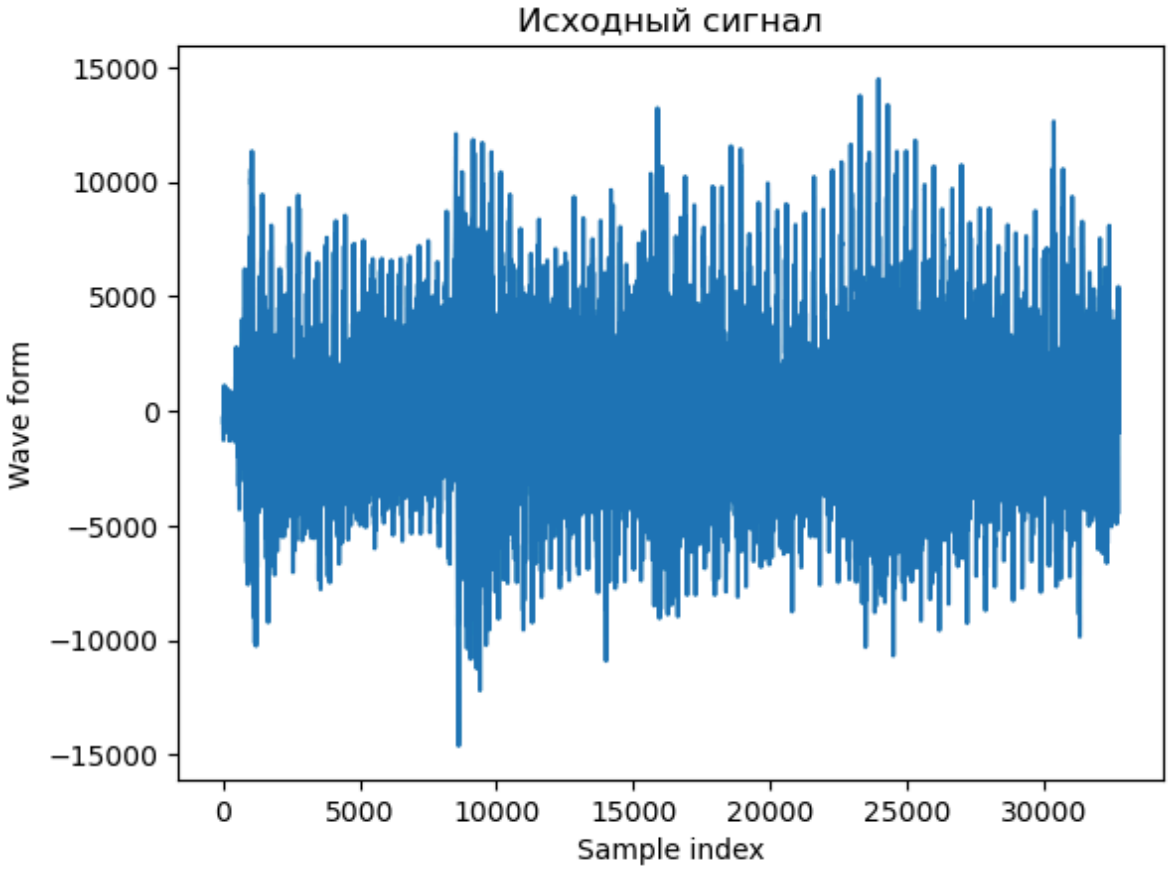
Crop Audio

```
Samples = 1024*32
data_short = data[:Samples]
print ('DATAE SIZE', data_short.size)
Audio(data_short, rate = Fs)
```

```
DATAE SIZE 32768
```



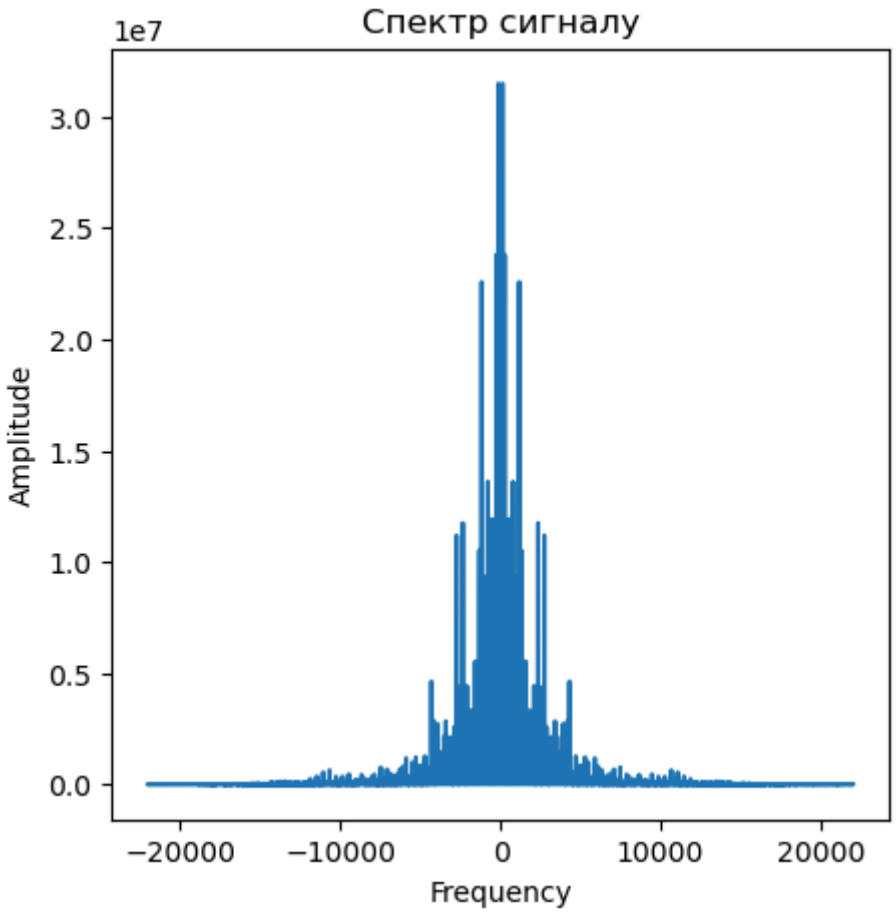

```
plt.figure()
plt.plot(data_short)
plt.title("Исходный сигнал")
plt.xlabel('Sample index')
plt.ylabel('Wave form')
plt.show()
```



Перетворення Фур'є одновимірного сигналу

```
freq_amplitude = fft(data_short)
freq = fftfreq(Samples, 1 / Fs)
print ('Мінімальна частота', np.int32(freq.min()), 'Максимальна
частота',np.int32(freq.max()))

plt.figure(figsize=(5, 5))
plt.plot(freq, np.abs(freq_amplitude))
plt.title("Спектр сигналу")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()
```

```
Мінімальна частота -22050 Максимальна частота 22048
```

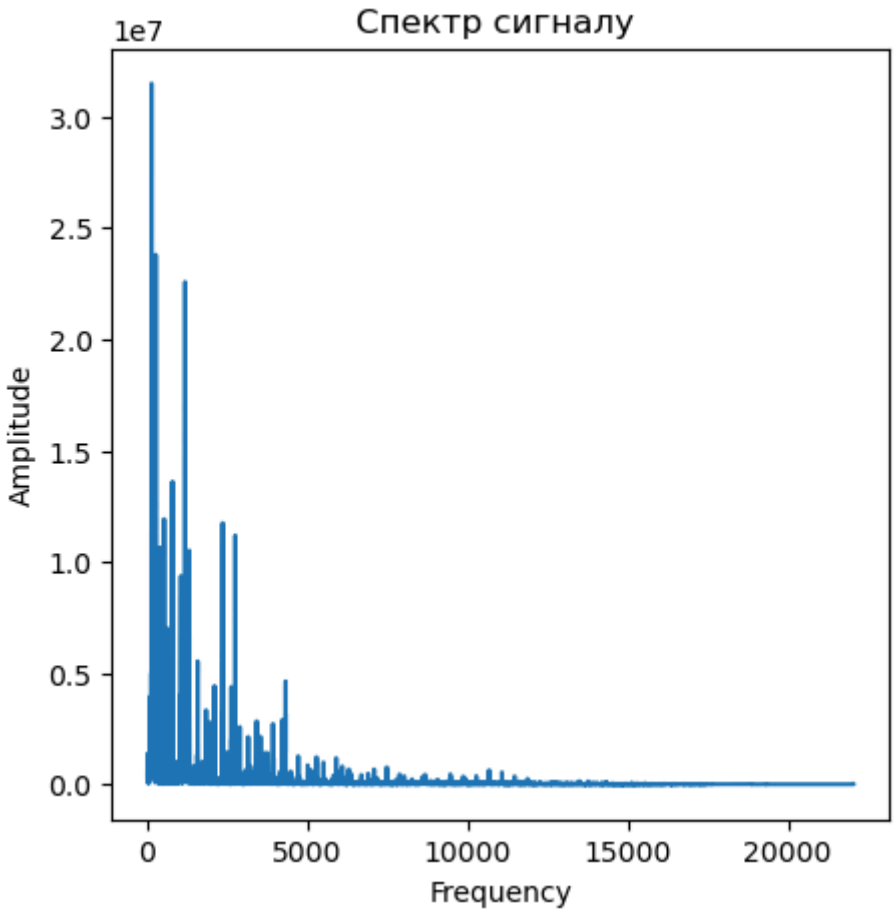


Перетворення Фур'є одновимірного сигналу

Позитивний частотни простір

```
freq_r_amplitude = rfft(data_short) # СПЕКТР
freq_r = rfftfreq(Samples, 1 / Fs)
print ('Минимальная частота', np.int32(freq_r.min()), 'Максимальная
частота',np.int32(freq_r.max()))

plt.figure(figsize=(5, 5))
plt.plot(freq_r, np.abs(freq_r_amplitude))
plt.title("Спектр сигналу ")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()
```

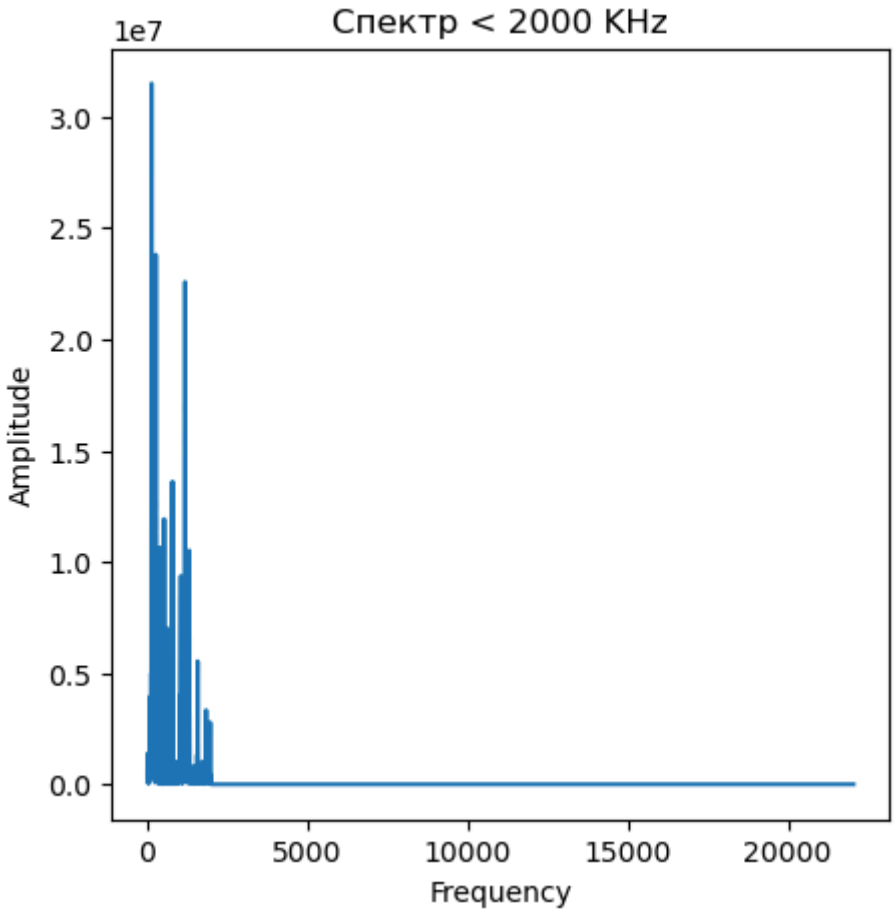
```
Минимальная частота 0 Максимальная частота 22050
```



Пригнічення частот вище 2000 Гц

```
Max_freq = 2000

print ('ЧАСТОТА ЗРІЗУ', Max_freq)
len_freq = len(freq_r_amplitude)
print(len_freq)

low_freq_amplitude = np.array (len_freq, dtype = np.complex128)
low_freq_amplitude = freq_r_amplitude.copy()

samples_per_freq = len(freq_r) / (Fs / 2)
target_index = np.int32(samples_per_freq*Max_freq)

low_freq_amplitude [target_index-1  : ] = 0 # ВСЕ, ЧТО ВЫШЕ 2000 Гц обнуляем

plt.figure(figsize=(5, 5))
plt.plot(freq_r, np.abs(low_freq_amplitude))
plt.title("Спектр < 2000 KHz")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()
```

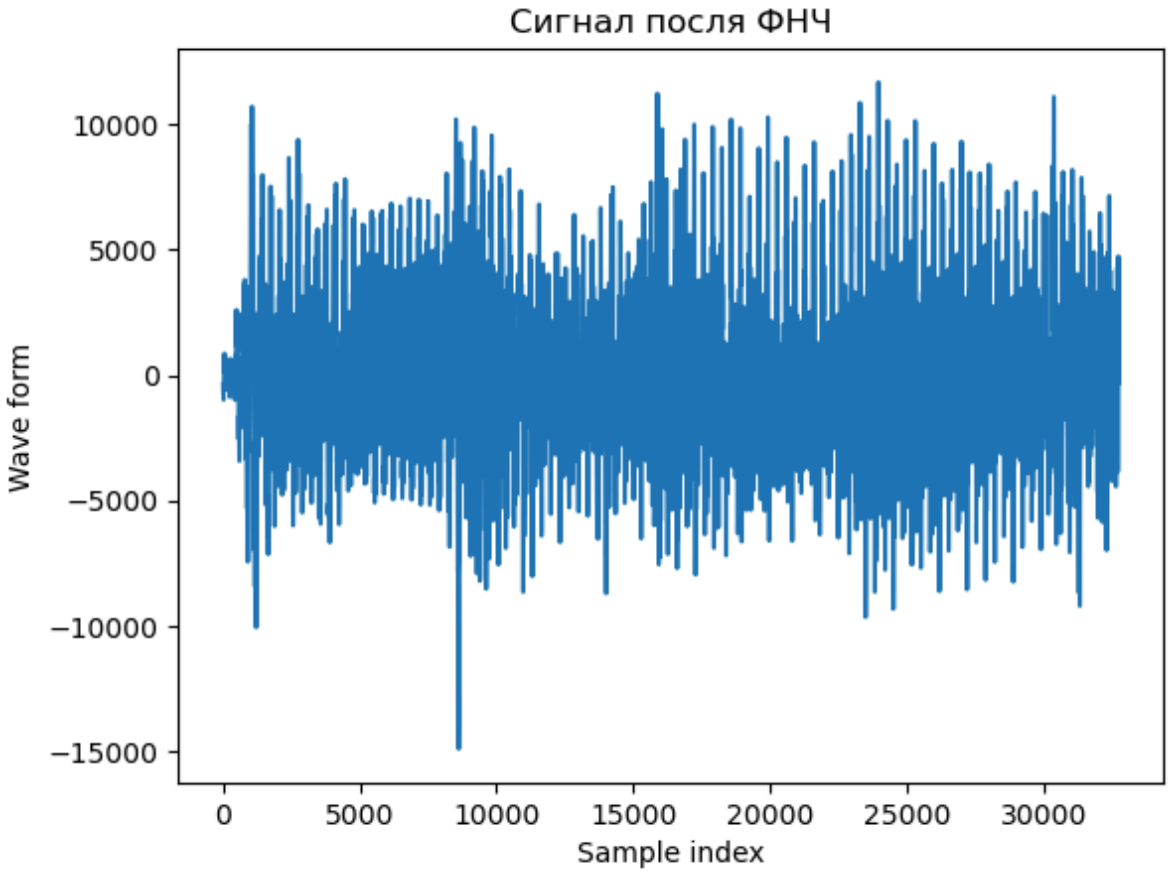
```
ЧАСТОТА ЗРІЗУ 2000
16385
```



Зворотнє перетворення Фур'є

```
low_filter_audio = irfft(low_freq_amplitude)

plt.figure(figsize=(5, 5))
plt.figure()
plt.plot(low_filter_audio)
plt.title("Сигнал после ФНЧ")
plt.xlabel('Sample index')
plt.ylabel('Wave form')
plt.show()

print (low_filter_audio.min(),low_filter_audio.max() )
abs_low_filter_audio = np.abs(low_filter_audio)
max_abs = abs_low_filter_audio.max()
print('Max value', max_abs)

norm_low_filter_audio = np.int16( low_filter_audio  * (16000 / max_abs ))
#norm_low_filter_audio = np.int16( low_filter_audio)
print ('Min & Max of normalise signal ',
norm_low_filter_audio.min(),norm_low_filter_audio.max() )

Audio(norm_low_filter_audio, rate = Fs)
```

```
<Figure size 500x500 with 0 Axes>
```




```
-14902.432711659718 11685.242428451074
Max value 14902.432711659718
Min & Max of normalise signal  -16000 12545
```

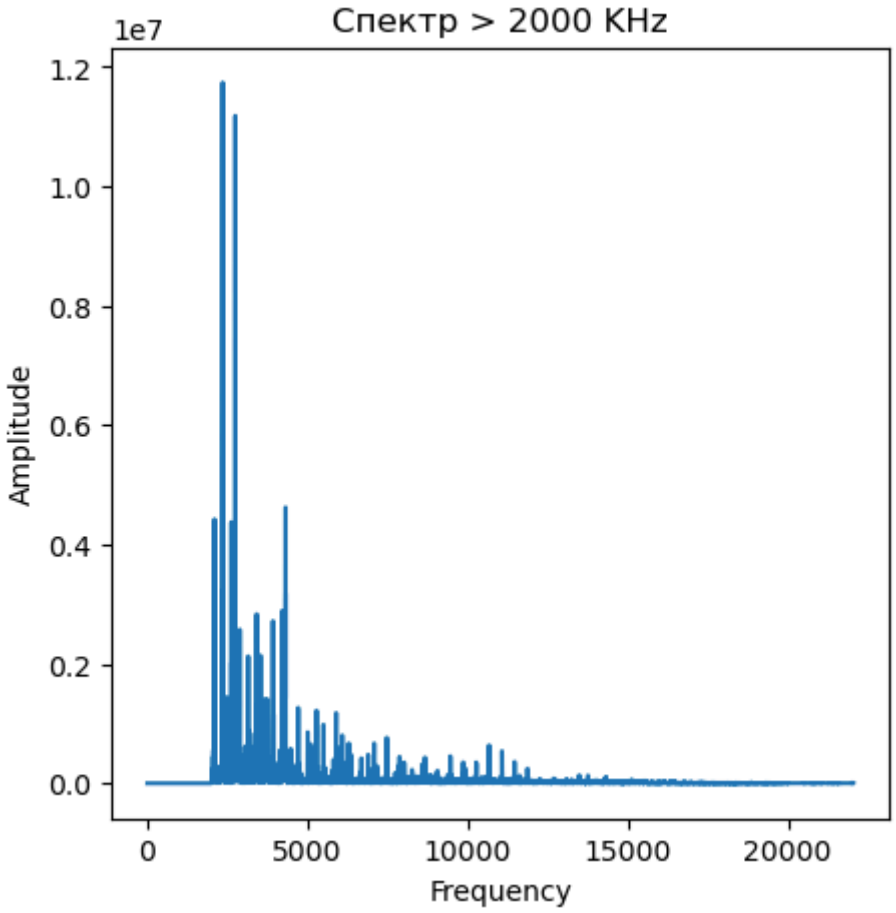
0:00 / 0:00

```
Audio(data_short, rate = Fs)
```

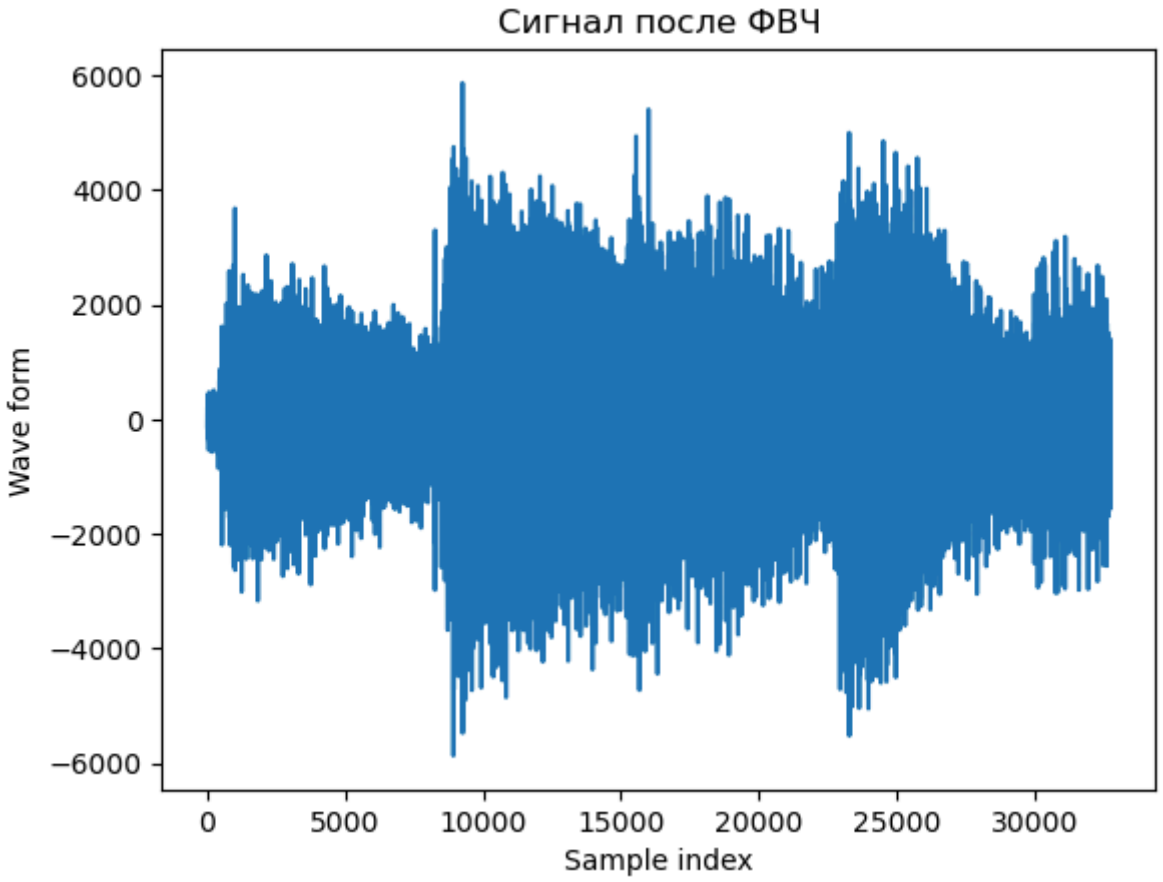
0:00 / 0:00

Пригнічення частот нижче 2000 Гц

```
high_freq_amplitude = np.array (len_freq, dtype = np.complex128)
high_freq_amplitude = freq_r_amplitude.copy()

high_freq_amplitude [ : target_index-1] = 0 # ВСЕ, ЧТО НИЖЕ 2000 Гц обнуляем


plt.figure(figsize=(5, 5))
plt.plot(freq_r, np.abs(high_freq_amplitude))
plt.title("Спектр > 2000 КНz")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()

high_filter_audio = irfft(high_freq_amplitude)

plt.figure(figsize=(5, 5))
plt.figure()
plt.plot(high_filter_audio)
plt.title("Сигнал после ФВЧ")
plt.xlabel('Sample index')
plt.ylabel('Wave form')
plt.show()

abs_high_filter_audio = np.abs(high_filter_audio)

max_abs = abs_high_filter_audio.max()
print(max_abs)
norm_high_filter_audio = np.int16( abs_high_filter_audio  * (16000 / max_abs ))

Audio(norm_high_filter_audio , rate = Fs)
```




```
<Figure size 500x500 with 0 Axes>
```




```
5872.60575991734
```

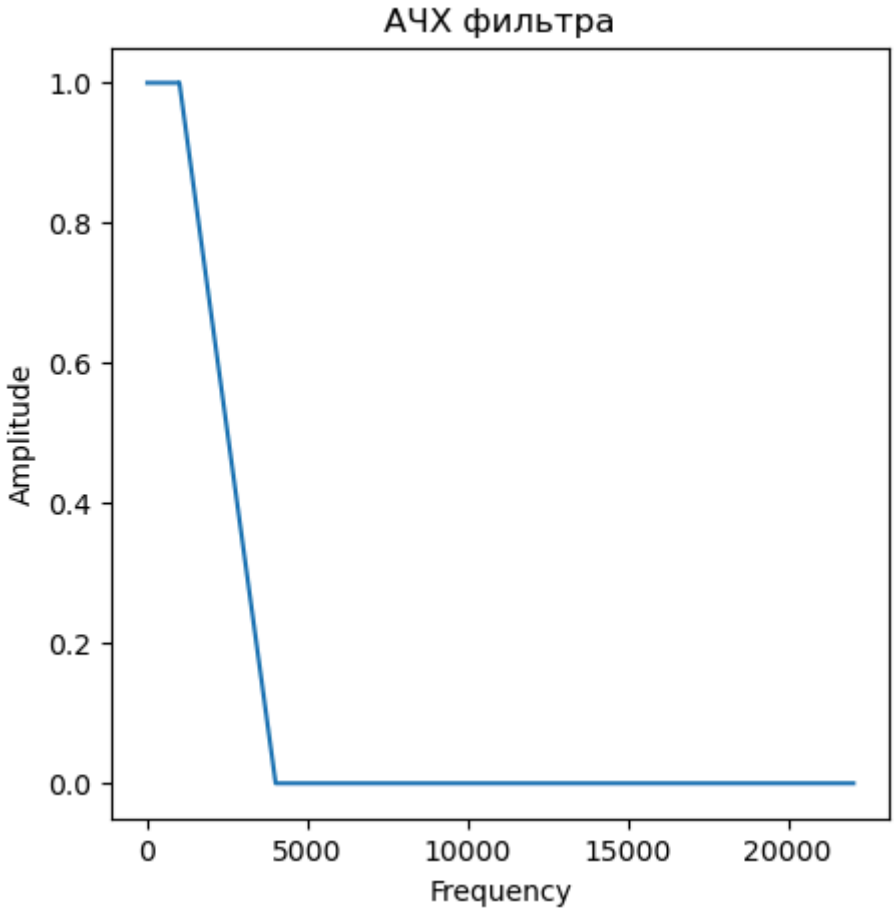
0:00 / 0:00

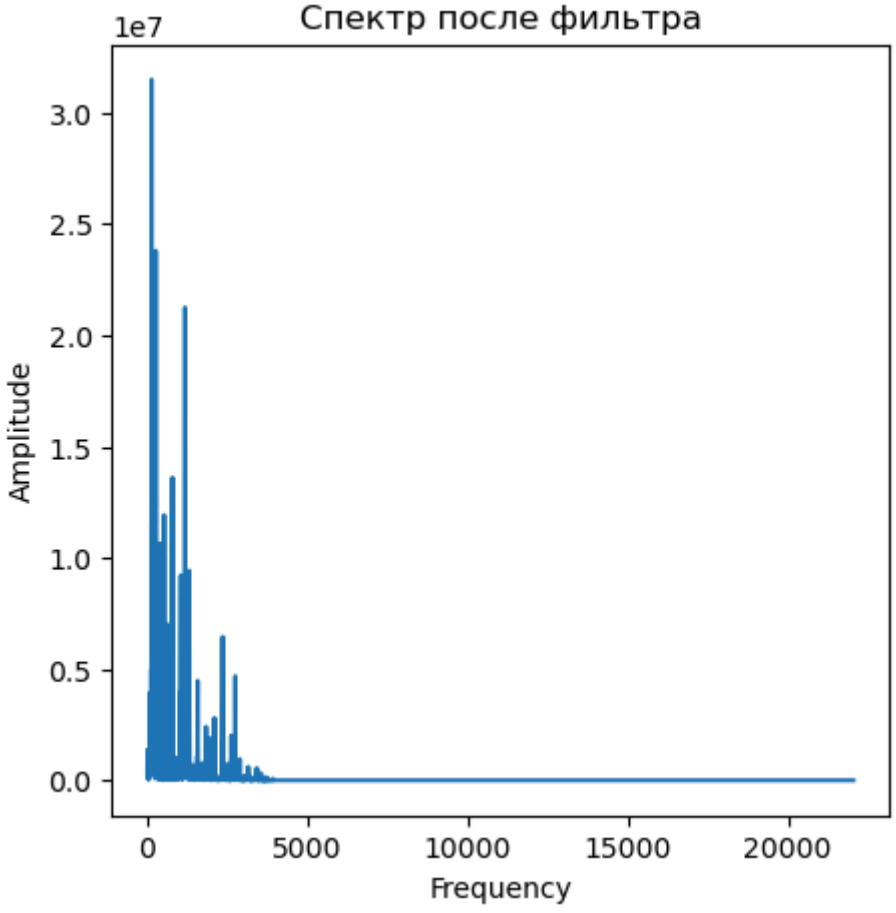
```
Audio(data_short, rate = Fs)
```

0:00 / 0:00

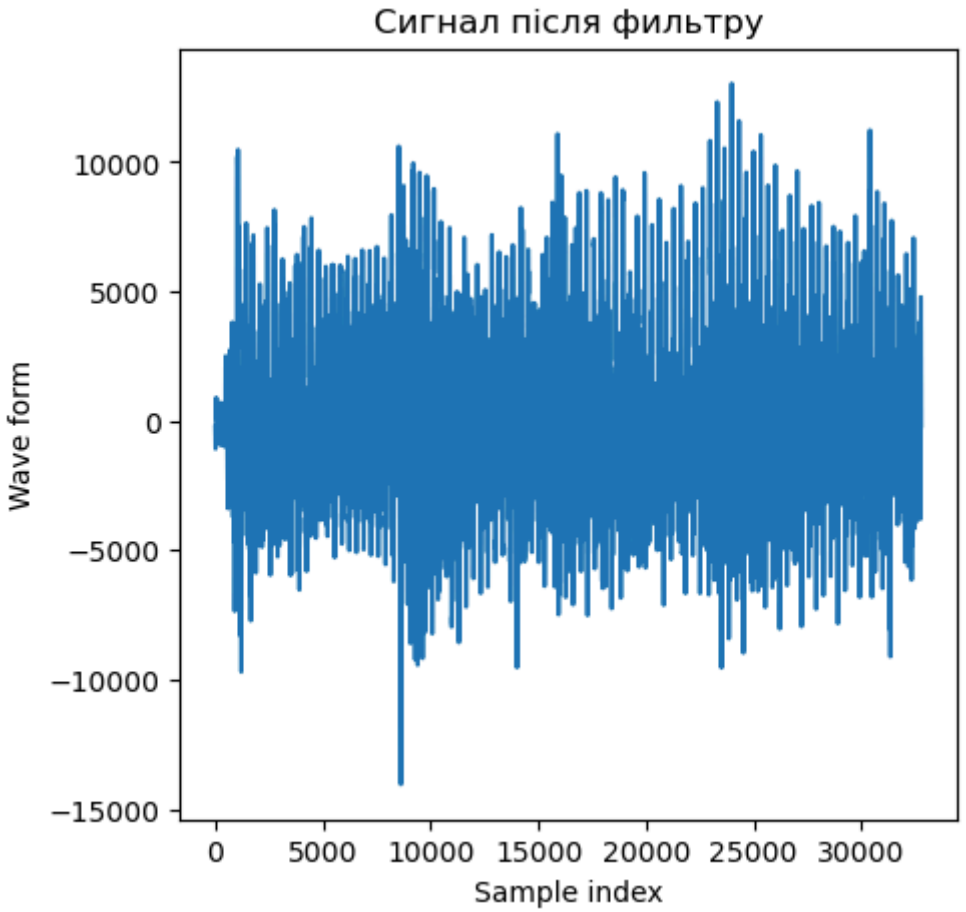
Завдання довільної АЧХ фильтру

```
# Визначаємо  АЧХ фильтра
filter_koeff  = np.zeros (len_freq, dtype = np.complex128)
# Трапецеидальный фильтр Freq_low = 1 KHz, Freq_high = 4 KHz
Freq_low  = 1000 # Частота среза
Freq_high = 4000 # Частота обнуления

target_index_low  = np.int32(samples_per_freq*Freq_low)
target_index_high = np.int32(samples_per_freq*Freq_high)

filter_koeff [ : target_index_low -1] = 1.0+0.0j

for i in range(target_index_low-1,target_index_high):
    filter_koeff [i] = 1.0-(1.0 / (target_index_high-target_index_low)*(i-
target_index_low)) + 0.0j

filter_freq_amplitude = np.array (len_freq, dtype = np.complex128)
filter_freq_amplitude = freq_r_amplitude * filter_koeff


plt.figure(figsize=(5, 5))
plt.plot(freq_r, np.abs(filter_koeff))
plt.title("АЧХ фильтра")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()

plt.figure(figsize=(5, 5))
plt.plot(freq_r, np.abs(filter_freq_amplitude))
plt.title("Спектр после фильтра")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.show()
```

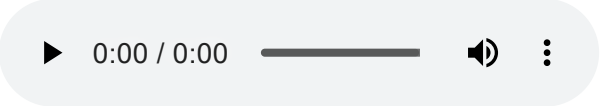

АЧХ фильтра
Amplitude
Frequency
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000


Спектр после фильтра
1e7
Amplitude
Frequency
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
5000
10000
15000
20000

```
filter_audio = irfft(filter_freq_amplitude)

plt.figure(figsize=(5, 5))
plt.plot(filter_audio)
plt.title("Сигнал після фильтру")
plt.xlabel('Sample index')
plt.ylabel('Wave form')
plt.show()


abs_filter_audio = np.abs(filter_audio)

max_abs = abs_filter_audio.max()
print(max_abs)
norm_filter_audio = np.int16( abs_filter_audio  * (16000 / max_abs ))

Audio(norm_filter_audio , rate = Fs)
```




```
14038.878558675884
```


0:00 / 0:00