

# Лабораторная работа 1

## Упражнение 1.2

[Ссылка на Binder с примерами](#)

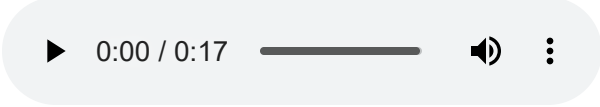
Скачайте с сайта <http://freesound.org> образец звука, включающий музыку, речь или иные звуки, имеющие четко выраженную высоту. Выделите примерно полусекундный сегмент, в котором высота постоянна. Вычислите и распечатайте спектр выбранного сегмента. Как связаны тембр звука и гармоническая структура, видимая в спектре?

Используйте **high\_pass**, **low\_pass** и **band\_stop** для фильтрации тех или иных гармоник. Затем преобразуйте спектры обратно в сигналы и прослушайте его. Как звук соотносится с изменениями, сделанными в спектре

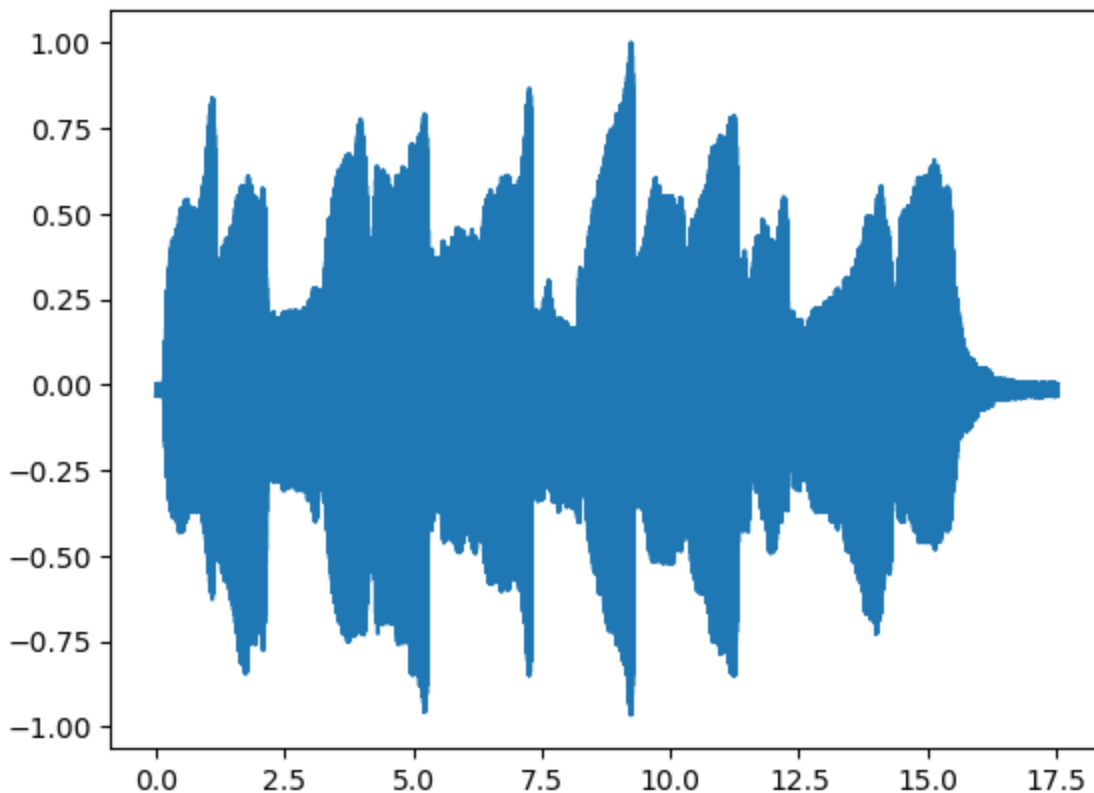
Загрузим звук с сайта

```
In [ ]: from thinkdsp import read_wave, decorate

wave = read_wave("violin.wav")
wave.make_audio()
```

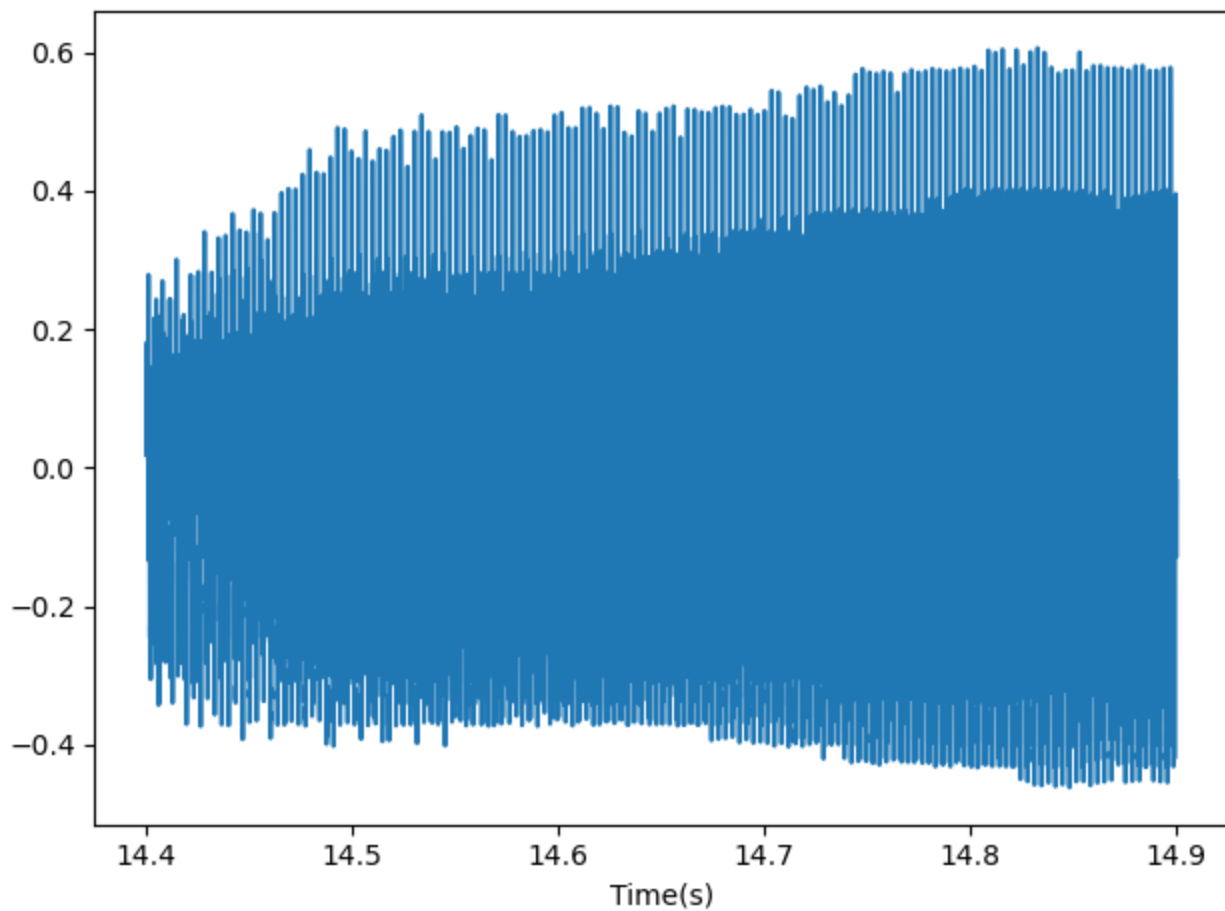
Out[ ]: 

```
In [ ]: wave.plot()
```



Выделим сегмент с примерно постоянной высотой

```
In [ ]: start = 14.4
duration = 0.5
segment = wave.segment(start, duration)
segment.plot()
decorate(xlabel='Time(s)')
```

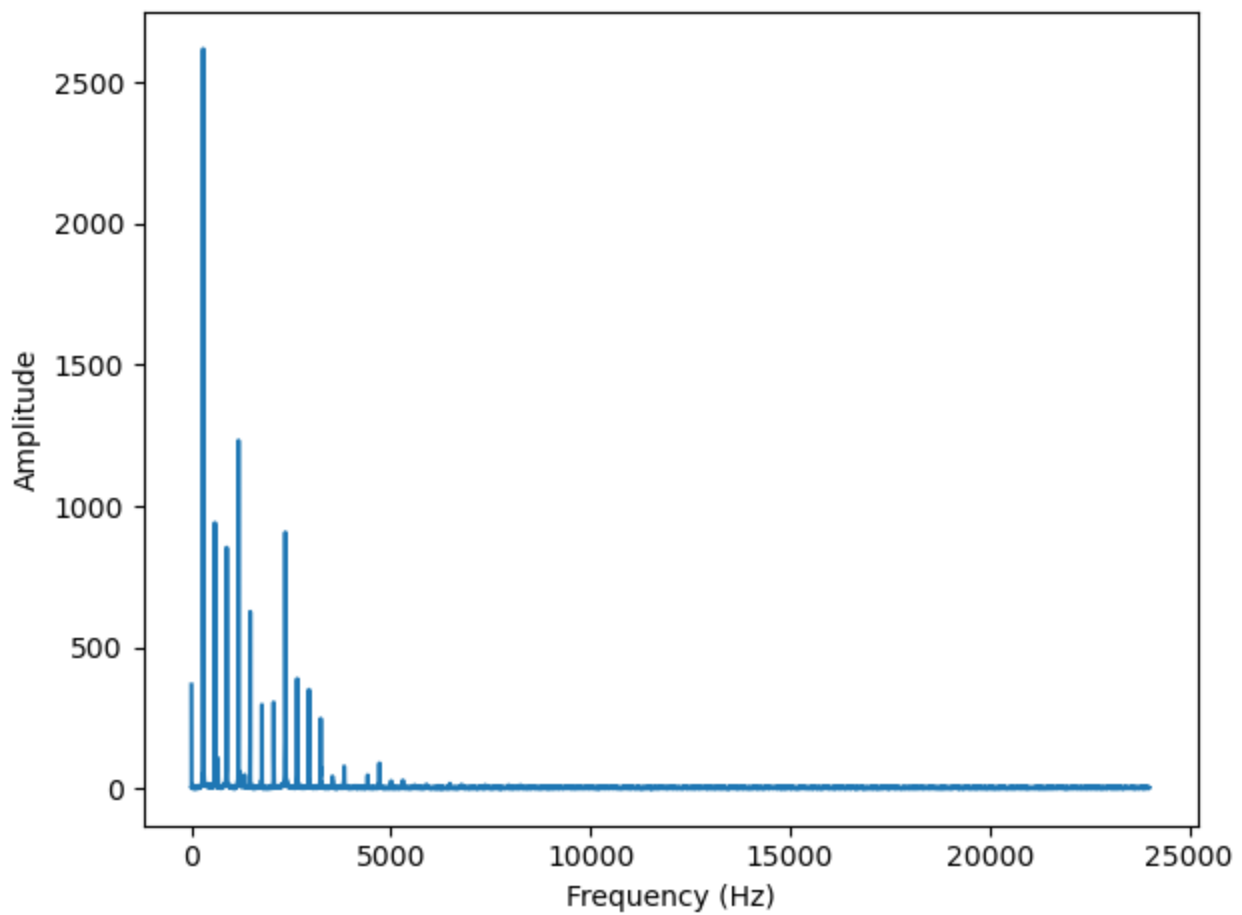


Рассчитаем спектр сегмента с помощью `make_spectrum` и распечатаем его с помощью метода `plot`

```
In [ ]: spectrum = segment.make_spectrum()
spectrum
```

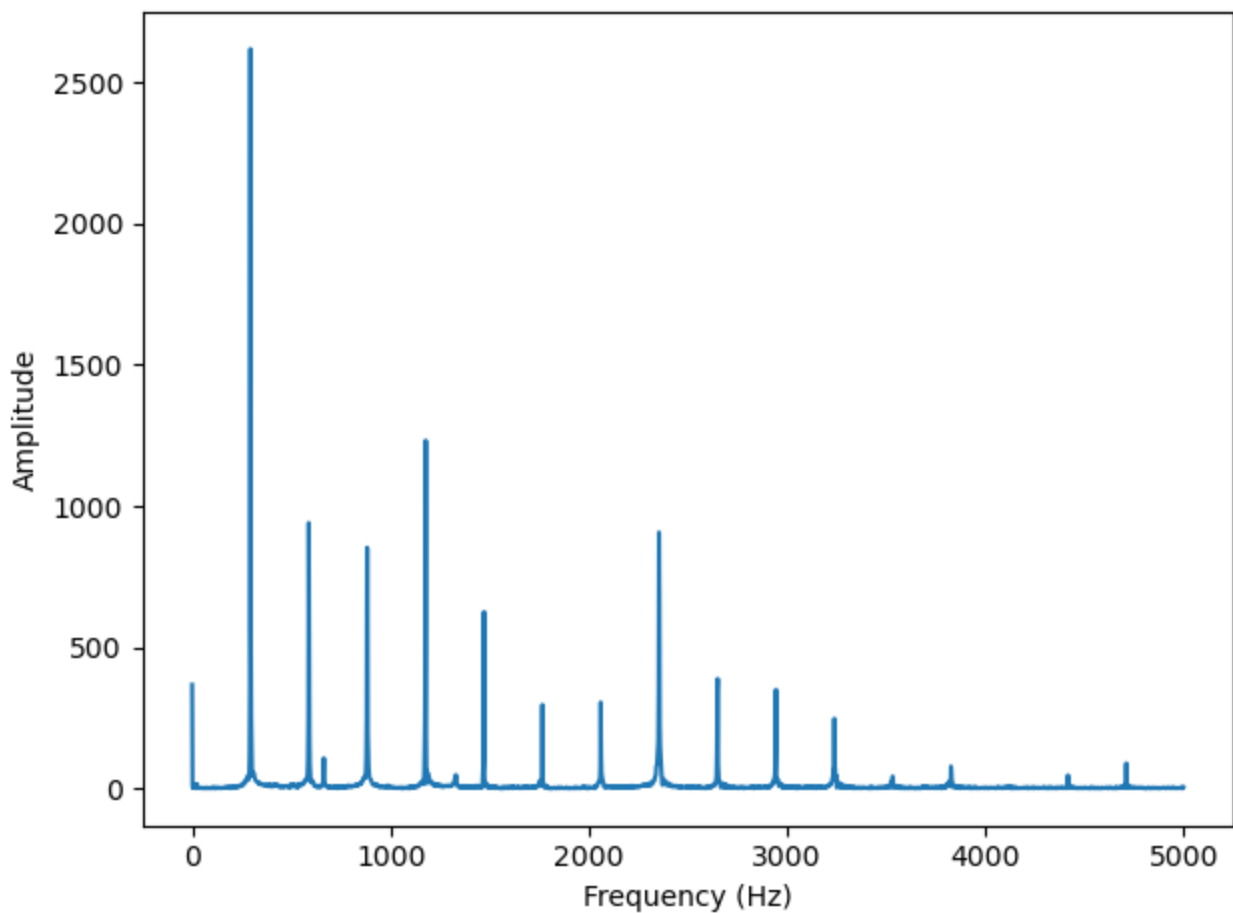
```
Out[ ]: <thinkdsp.Spectrum at 0x1e7b67ec7d0>
```

```
In [ ]: spectrum.plot()
decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```



Амплитуды частот выше 5 кГц очень малы. Можем поставить ограничение, чтобы лучше видеть низкие частоты

```
In [ ]: spectrum.plot(high=5000)
         decorate(xlabel='Frequency (Hz)', ylabel='Amplitude')
```



Выведем список 10 частот с наибольшей амплитудой

```
In [ ]: spectrum.peaks()[:10]
```

```
Out[ ]: [(2618.1084455164996, 294.0),
(1231.8685786319115, 1178.0),
(940.1886138323101, 588.0),
(906.7998610677967, 2354.0),
(852.7723016627676, 882.0),
(845.9360693564972, 884.0),
(674.9017561619022, 2356.0),
(625.4483467480884, 1472.0),
(577.915394701666, 296.0),
(546.605904693724, 1176.0)]
```

Базовая и доминирующая частота около 294 Гц - нота Ре первой октавы (D4). Другие пики:

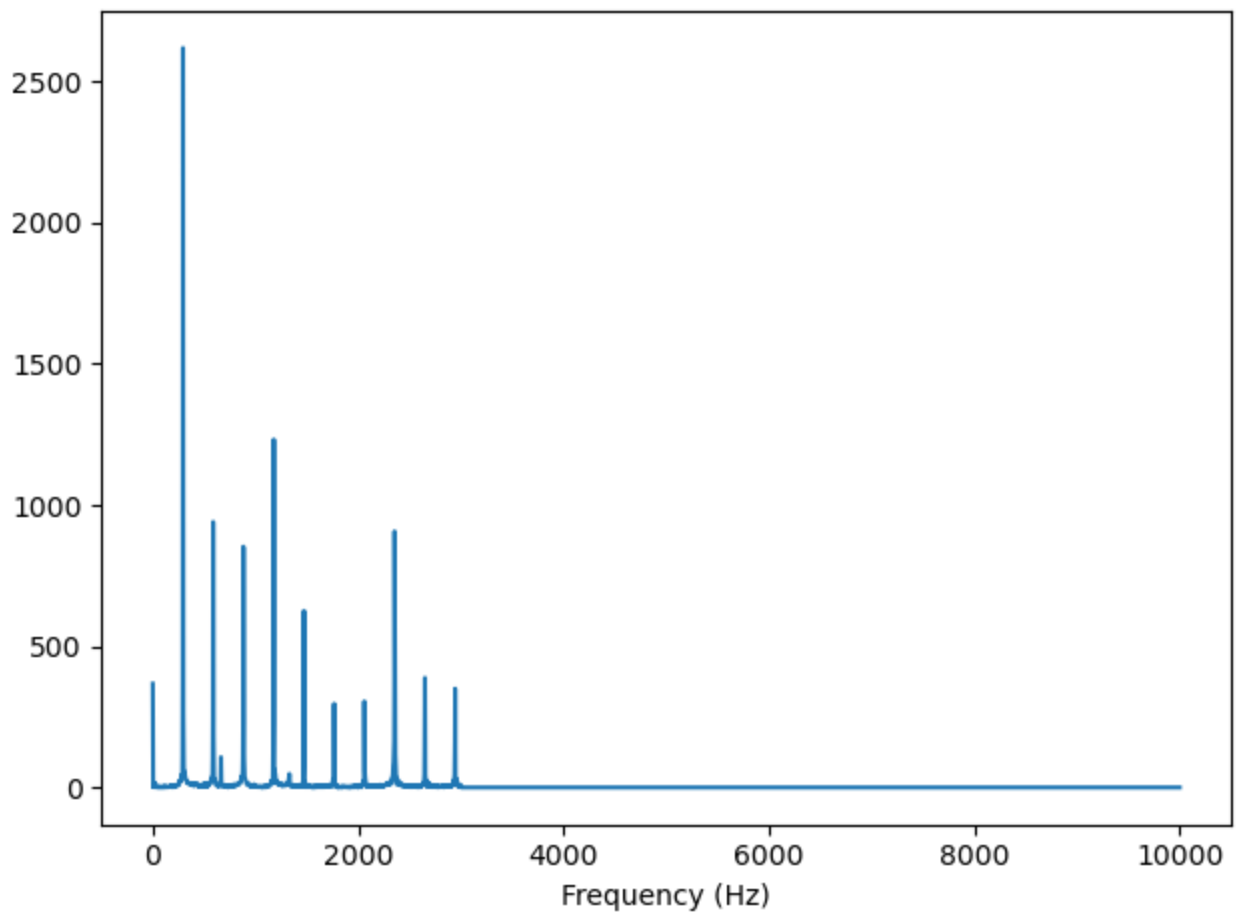
- 588 Гц - в 2 раза больше базы, следовательно октава. Ре второй октавы (D5)
- 882-884 Гц - Си бемоль(B5). Образует малую сексту с Ре (D5)
- 1178 Гц - две октавы от базы.
- ... Перечисленные частоты кратны базовой => гармоники

```
In [ ]: segment.make_audio()
```

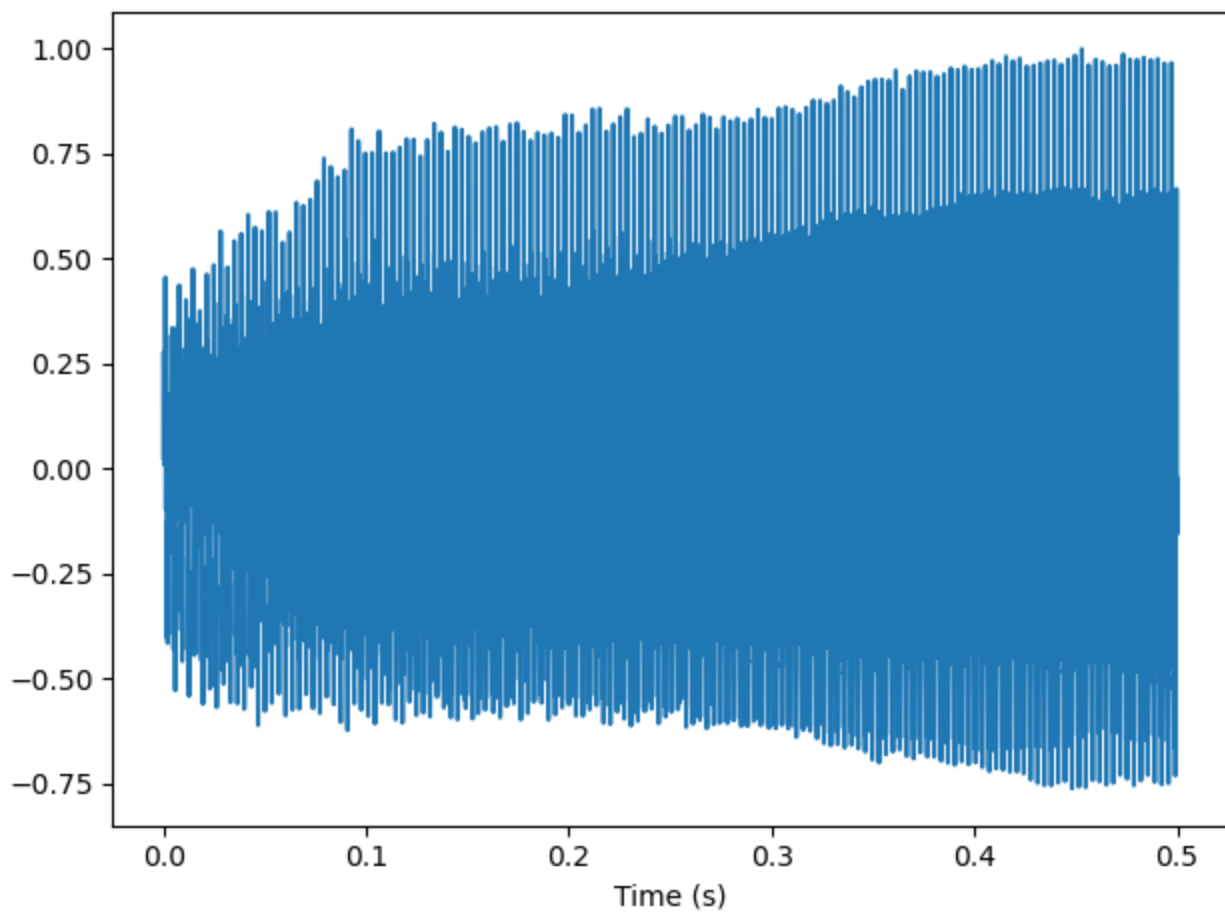
```
Out[ ]: ▶ 0:00 / 0:00 ————— 🔊 ⋮
```

Используя `low_pass` оставим только частоты ниже определенной.

```
In [ ]: spectrum.low_pass(3000)
spectrum.plot(high=10000)
decorate(xlabel='Frequency (Hz)')
```



```
In [ ]: filtered = spectrum.make_wave()
filtered.normalize()
filtered.plot()
decorate(xlabel='Time (s)')
```



Отфильтрованная волна выглядит более разрозненной

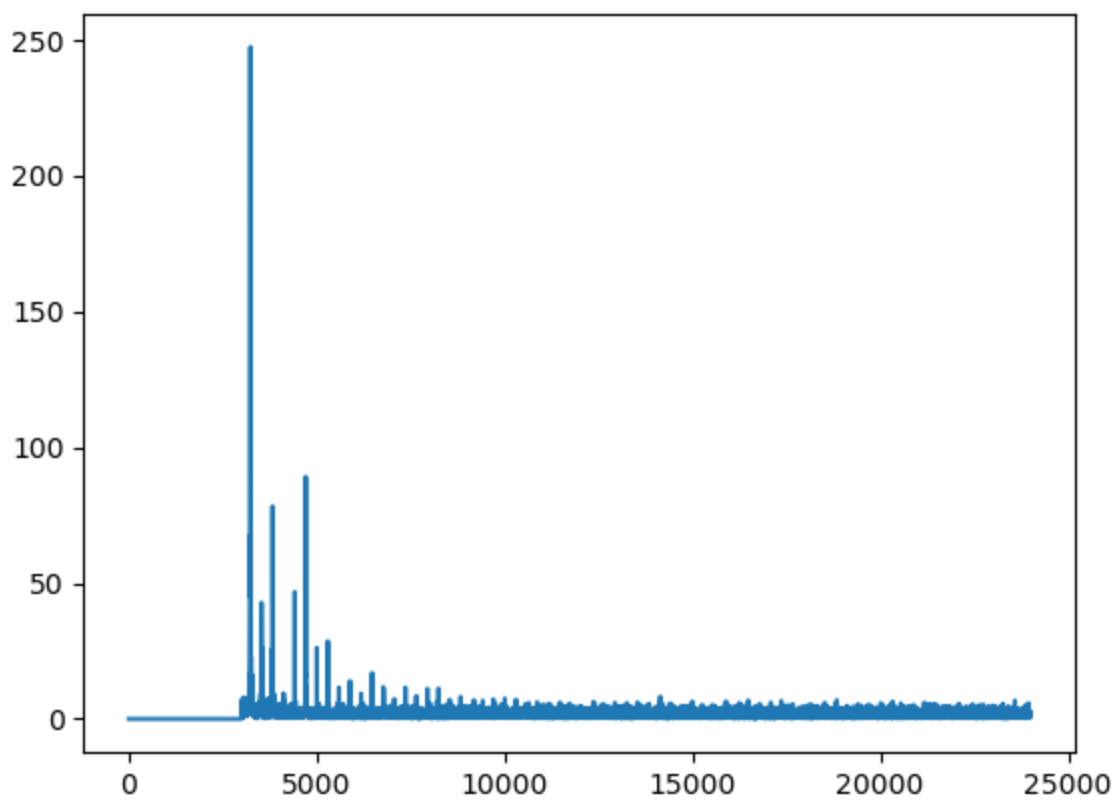
```
In [ ]: filtered.make_audio()
```

Out[ ]:

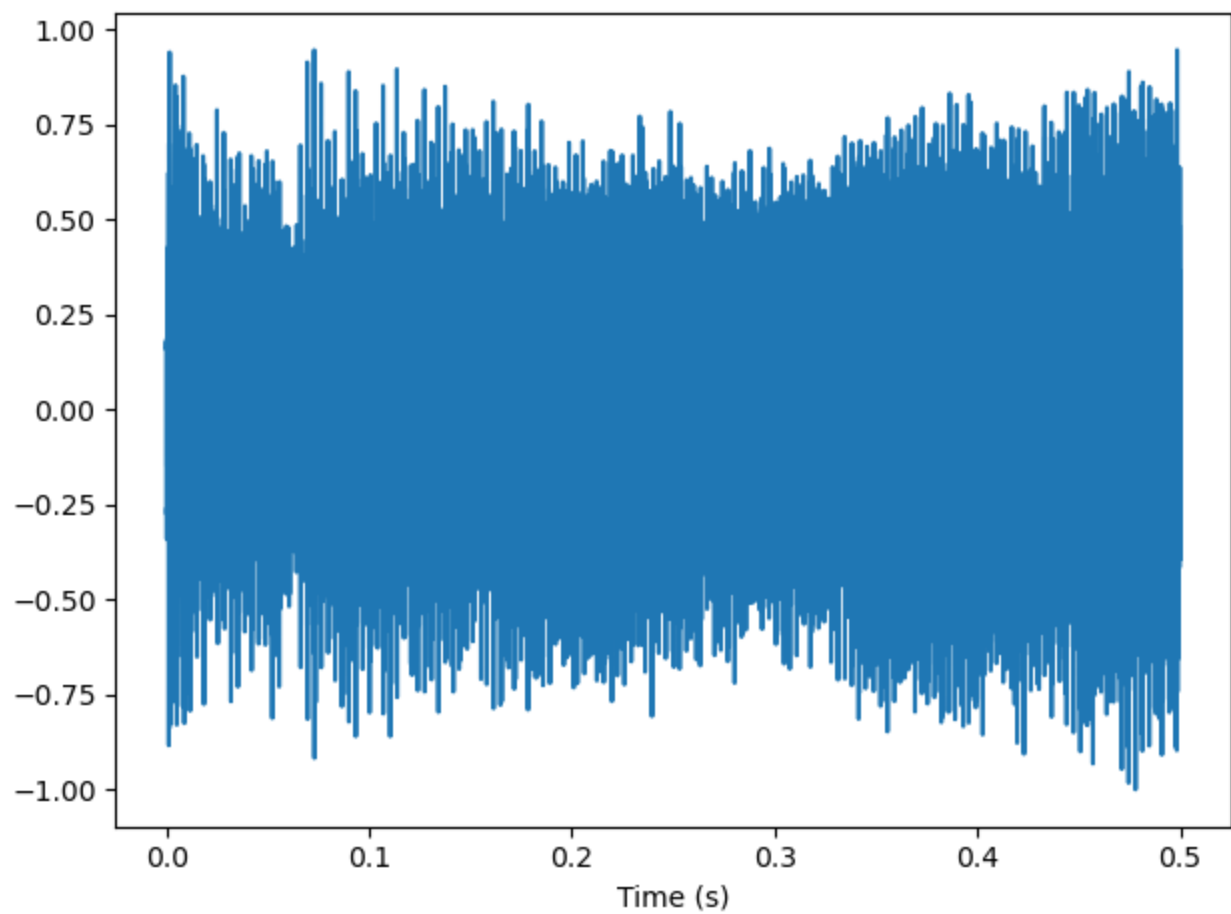


Полученный звук звучит более менее качественно, не так звонко. Как будто из телефона

```
In [ ]: spectrum = segment.make_spectrum()  
spectrum.high_pass(3000)  
spectrum.plot()
```

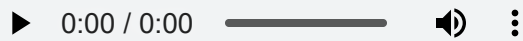


```
In [ ]: filtered = spectrum.make_wave()  
filtered.normalize()  
filtered.plot()  
decorate(xlabel='Time (s)')
```



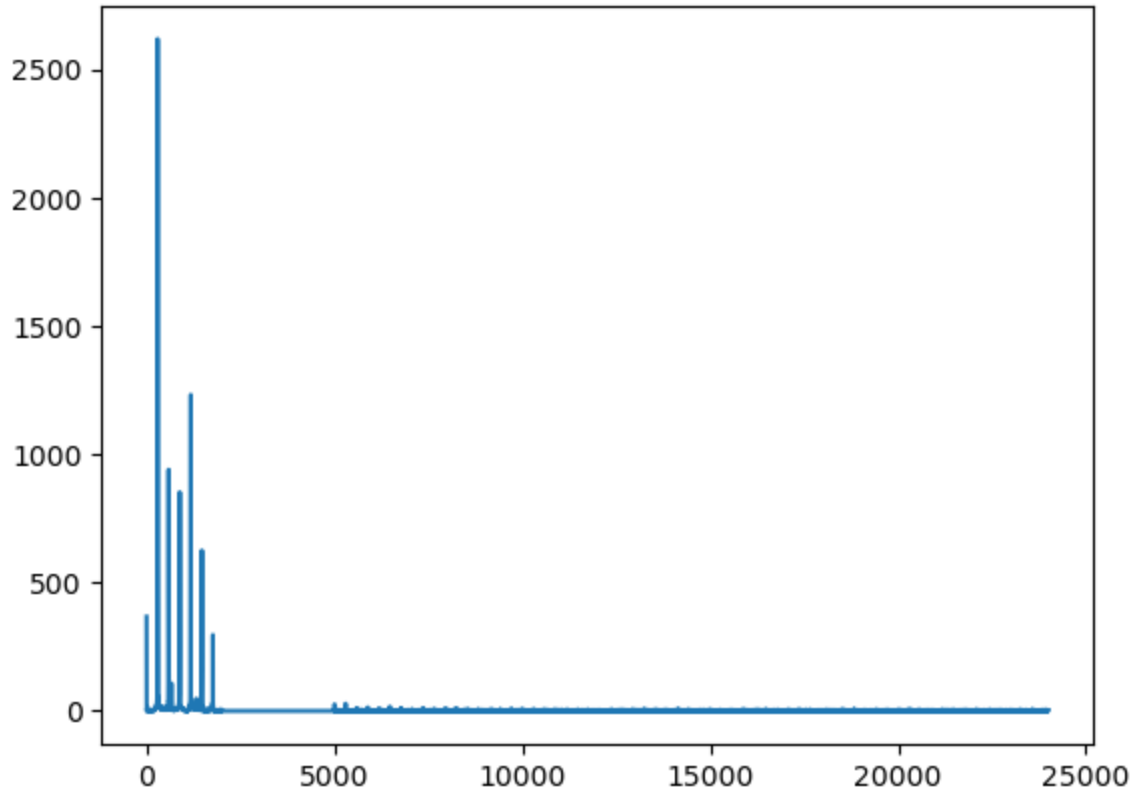
```
In [ ]: filtered.make_audio()
```

Out[ ]:



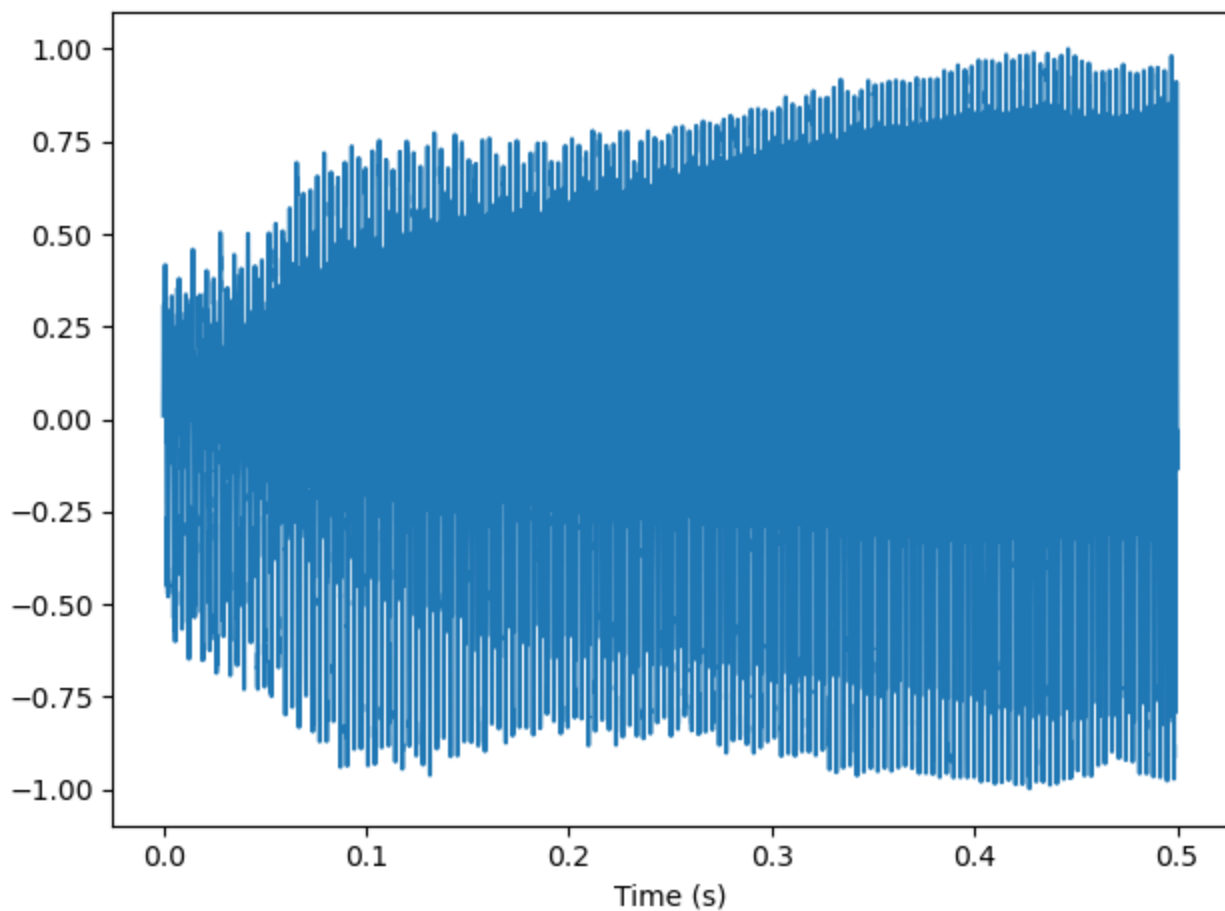
Полученный звук, который не имеет низких частот, звучит как кипящий чайник

```
In [ ]: spectrum = segment.make_spectrum()  
spectrum.band_stop(2000, 5000)  
spectrum.plot()
```



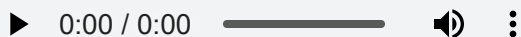
```
In [ ]: filtered = spectrum.make_wave()  
filtered.normalize()  
filtered.plot()  
decorate(xlabel='Time (s)')
```





```
In [ ]: filtered.make_audio()
```

Out[ ]:



Звук получился более гулким, как будто поместили под ведро

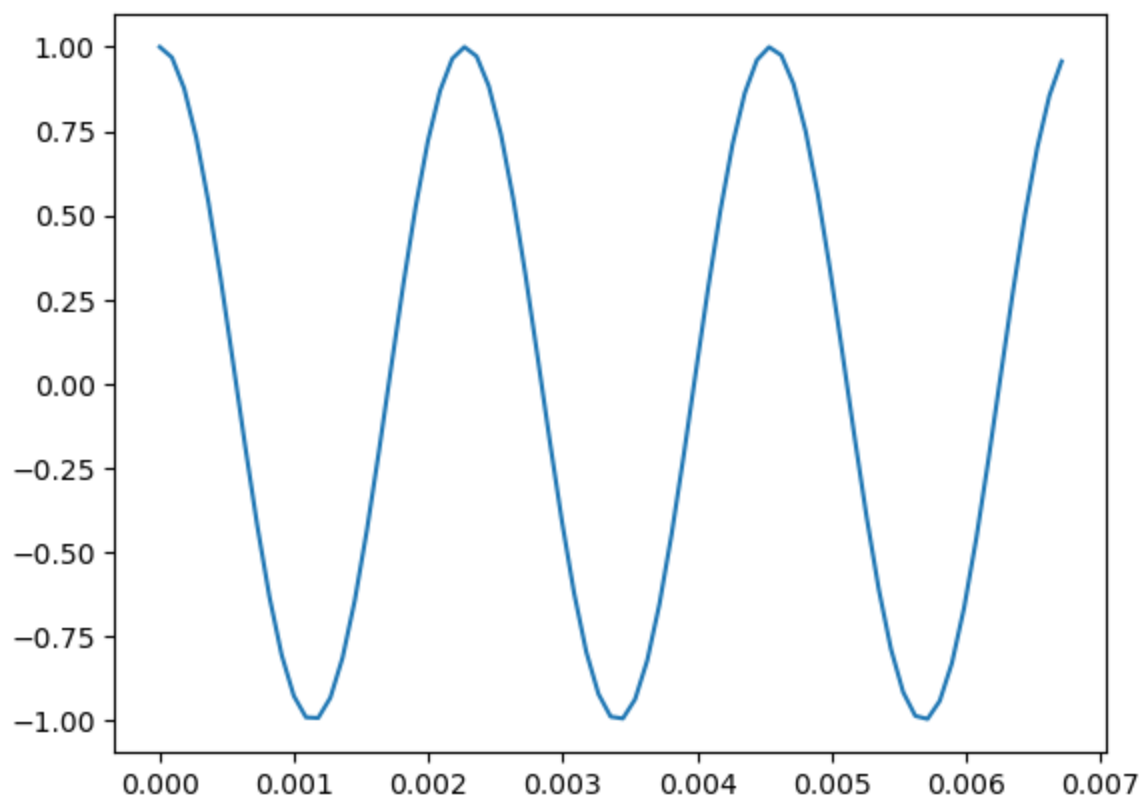
## Упражнение 1.3

Создайте сложный сигнал из объектов SinSignal и CosSignal, суммируя их. Обработайте сигнал для получения wave и прослушайте его. Вычислите spectrum и распечатайте. Что произойдет при добавлении частотных компонент, не кратных основным.

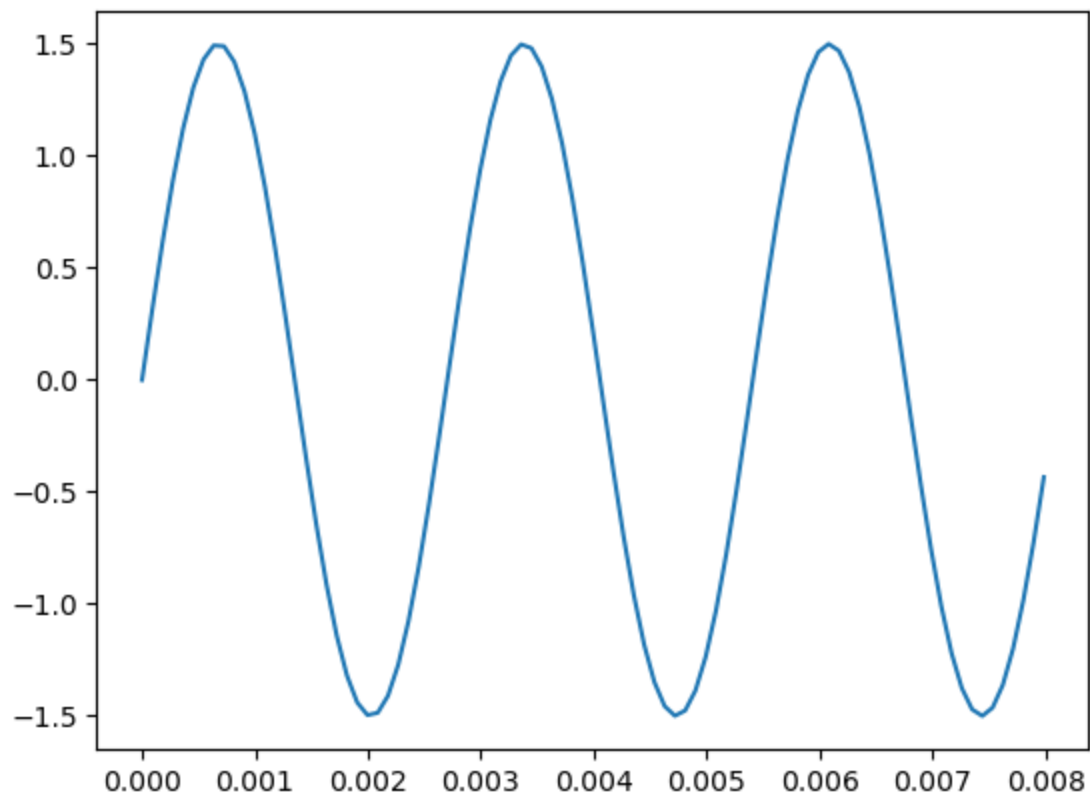
```
In [ ]: from thinkdsp import CosSignal, SinSignal

cos_sig = CosSignal(freq=440, amp=1.0, offset=0)
sin_sig = SinSignal(freq=370, amp=1.5, offset=0)
```

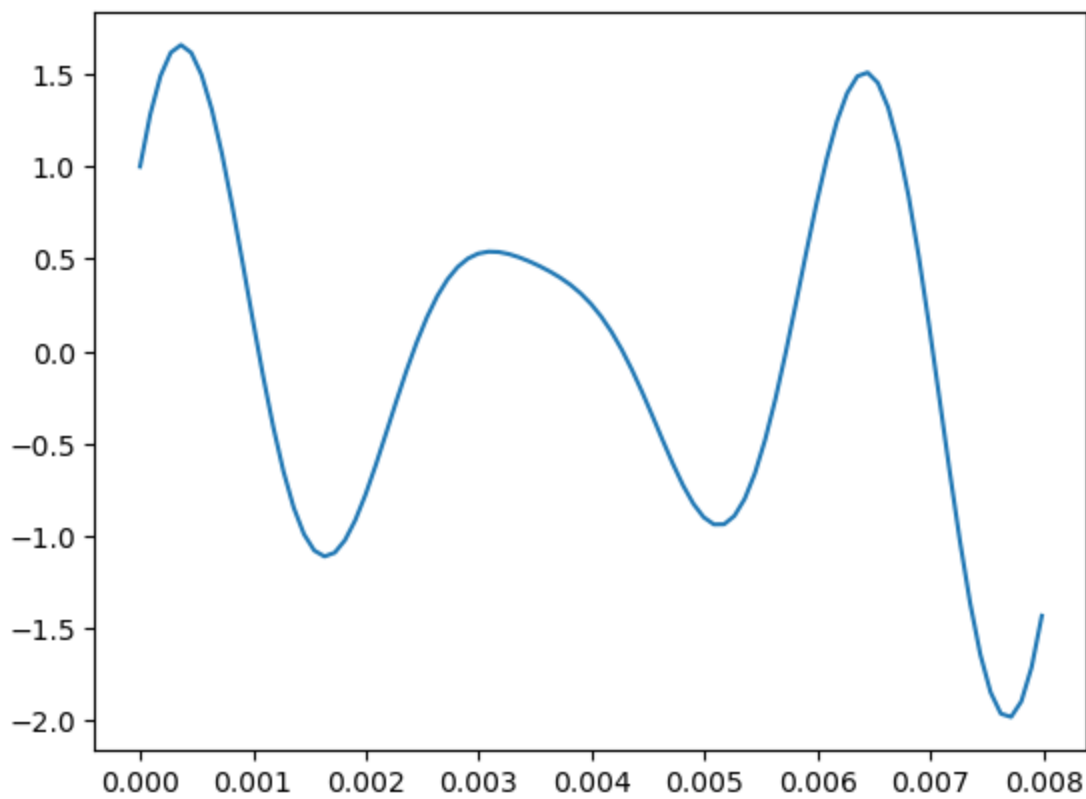
```
In [ ]: cos_sig.plot()
```



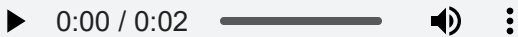
```
In [ ]: sin_sig.plot()
```



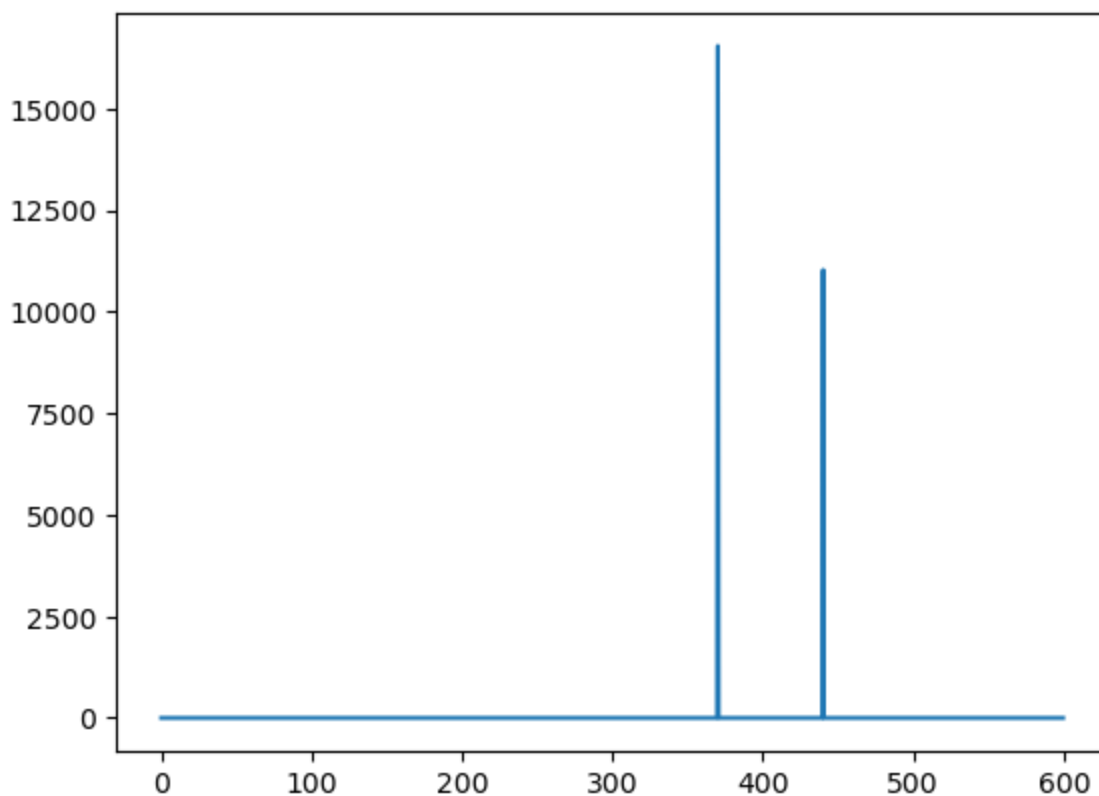
```
In [ ]: good_mix = sin_sig + cos_sig  
good_mix.plot()
```



```
In [ ]: good_wave = good_mix.make_wave(duration=2, start=0, framerate=11025)
good_wave.make_audio()
```

Out[ ]: 

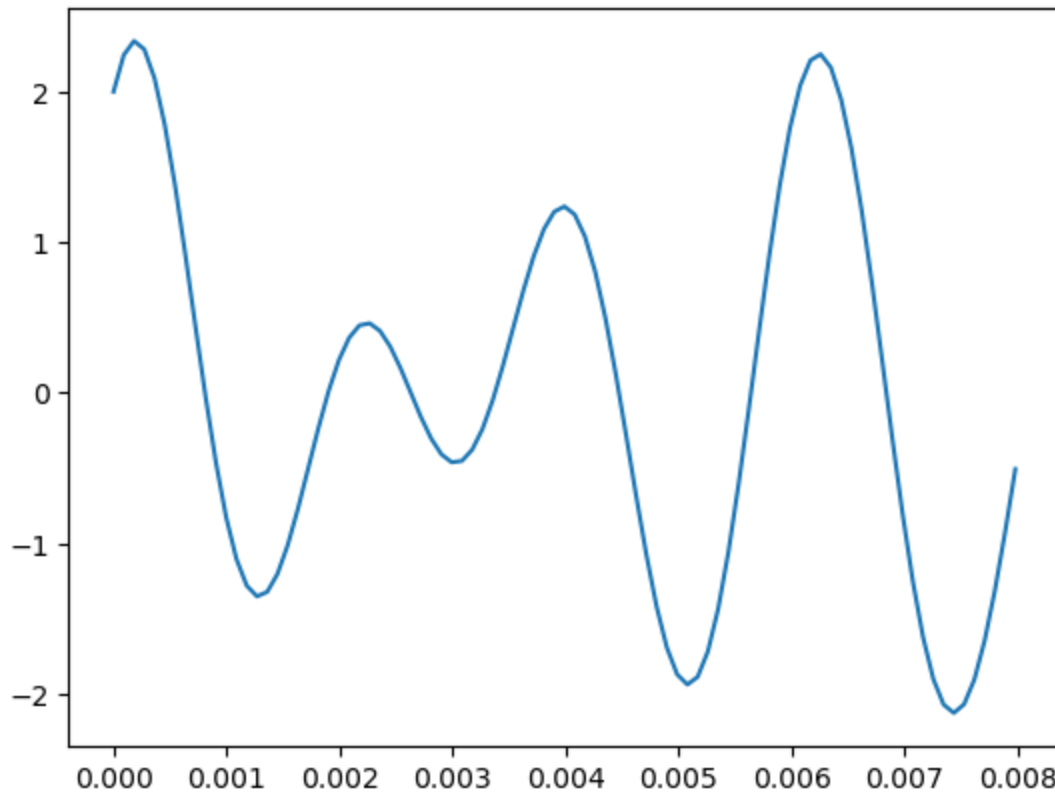
```
In [ ]: good_spectrum = good_wave.make_spectrum()
good_spectrum.plot(high=600)
```



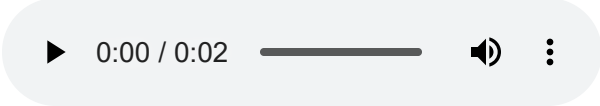
На спектре видим четко выраженные пики на частотах, которые мы задали волнам - 370 и 440 Гц. Они гармоничны, соответствуют нотам Фа диез и Ля соответственно

Теперь добавим к сигналу синусоиду, которая не будет кратна основной

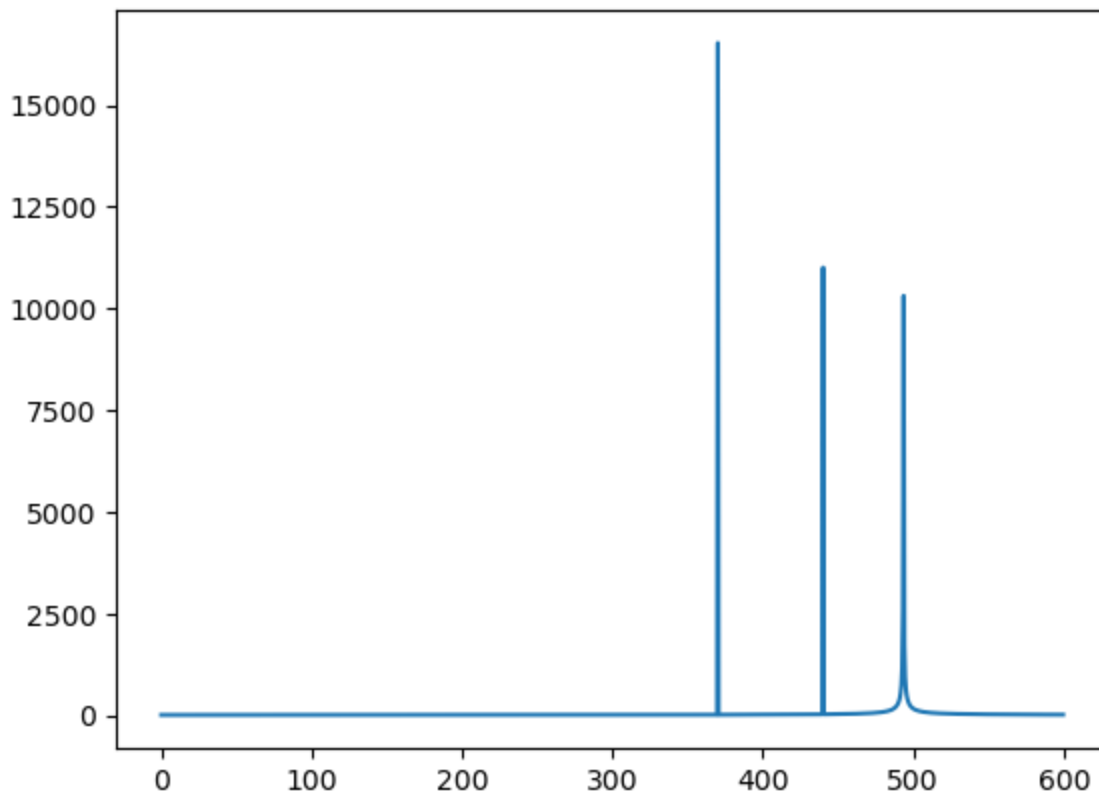
```
In [ ]: bad_mix = good_mix + CosSignal(freq=493.4, amp=1.0, offset=0)
bad_mix.plot()
```



```
In [ ]: bad_wave = bad_mix.make_wave(duration=2, start=0, framerate=11025)
bad_wave.make_audio()
```

Out[ ]: 

```
In [ ]: bad_spectrum = bad_wave.make_spectrum()
bad_spectrum.plot(high=600)
```



Полученный звук звучит не так приятно как предыдущий, так как мы добавили к волне негармоничную волну с частотой 493

## Упражнение 1.4

Напишите функцию `stretch`, берущую `wave` и коэффициент изменения. Она должна ускорять или замедлять сигнал изменением `ts` и `framerate`

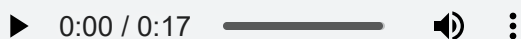
```
In [ ]: """
Speed up or slow down passed wave
params:
scale[in]: speed up wave in scale times
wave[out]:
"""

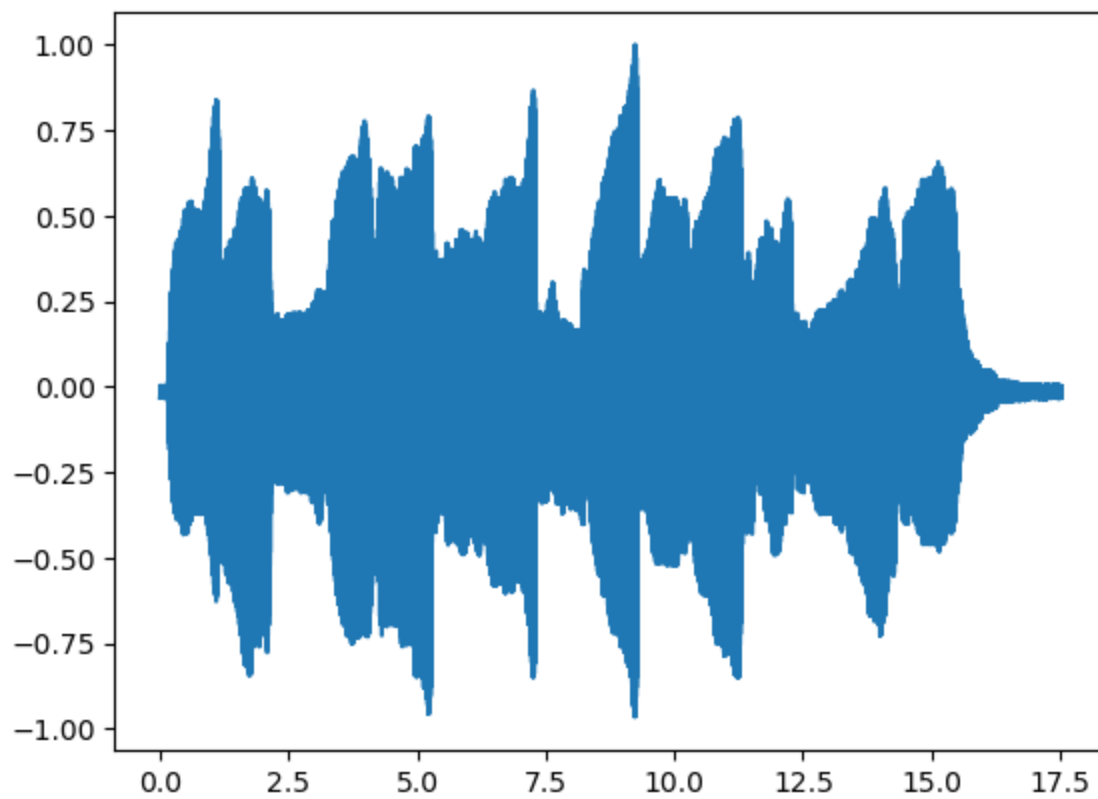
def stretch(wave, scale):
    wave.ts /= scale
    wave.framerate *= scale
```

Волна до изменений:

```
In [ ]: wave = read_wave("violin.wav")
wave.plot()
wave.make_audio()
```

Out[ ]:





Волна после ускорения в 2 раза:

```
In [ ]: stretch(wave, 2)  
wave.plot()  
wave.make_audio()
```

Out[ ]:

▶ 0:00 / 0:08 ————— 🔊 ⋮

