

Лабораторная работа 3

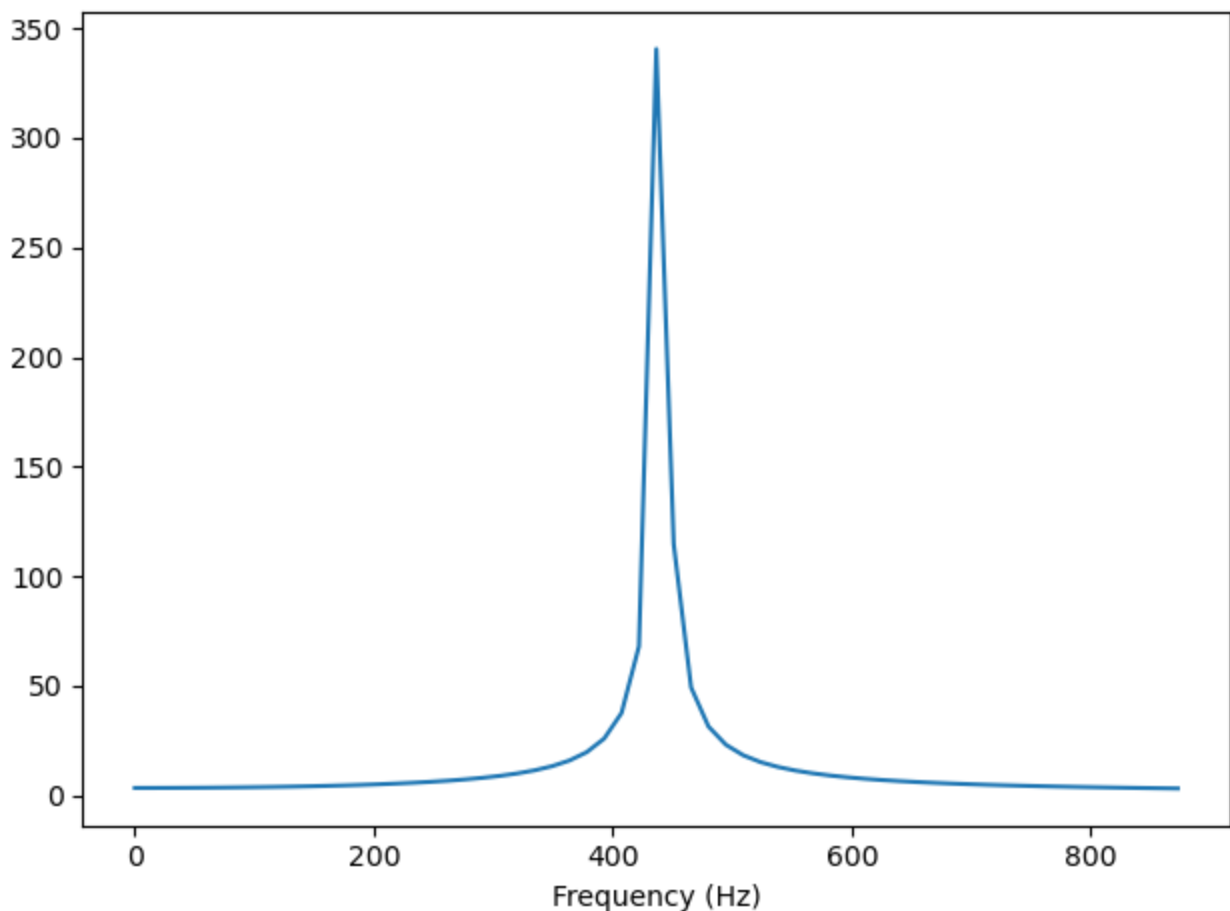
Упражнение 3.1

В примере с утечкой замените окно Хэмминга одним из окон, предоставляемых NumPy, и посмотрите, как они влияют на утечку

```
In [ ]: from thinkdsp import *
import numpy as np
from matplotlib import pyplot as plt

signal = SinSignal(freq=440)
duration = signal.period * 30.25
wave = signal.make_wave(duration)
spectrum = wave.make_spectrum()

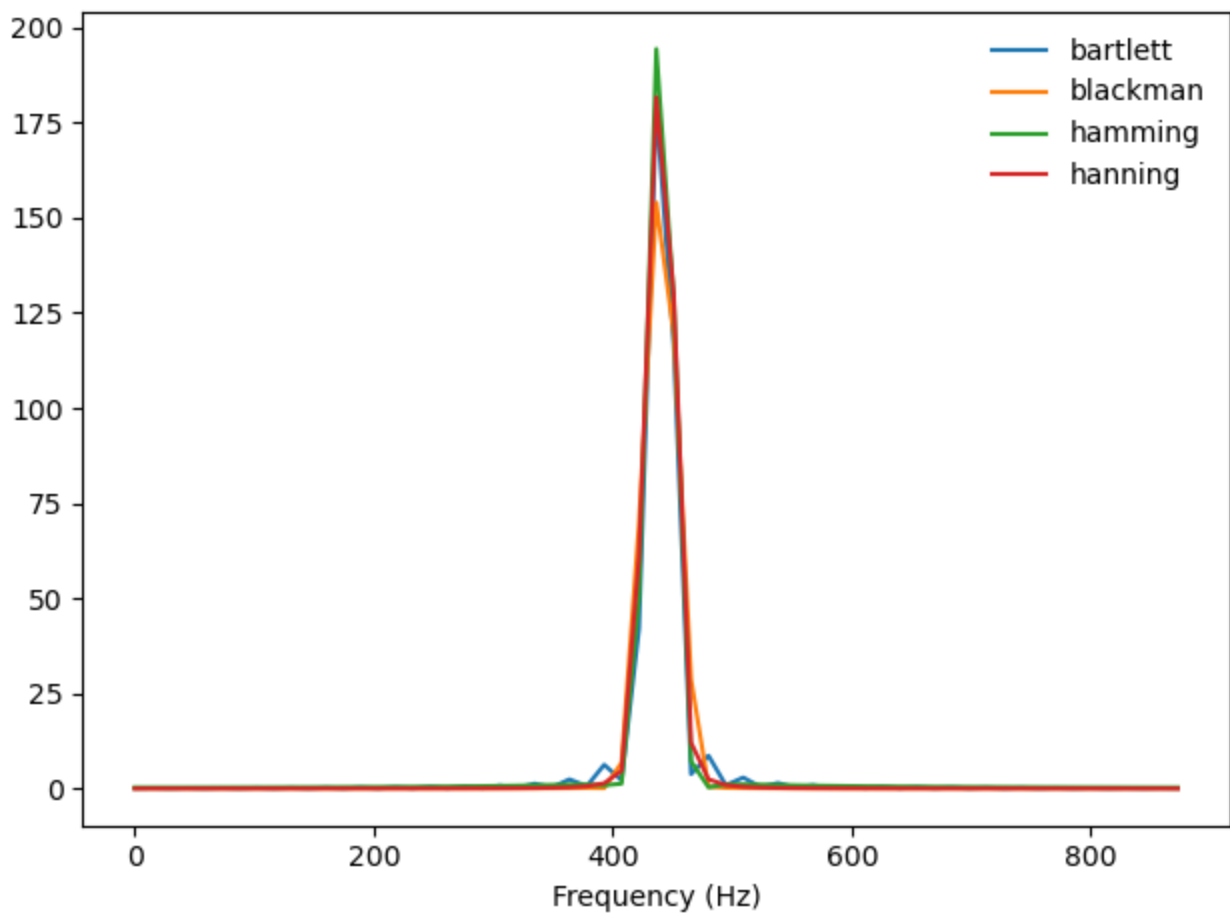
spectrum.plot(high=880)
decorate(xlabel='Frequency (Hz)')
```



```
In [ ]: for window_func in [np.bartlett, np.blackman, np.hamming, np.hanning]:
    wave = signal.make_wave(duration)
    wave.ys *= window_func(len(wave.ys))

    spectrum = wave.make_spectrum()
    spectrum.plot(high=880, label=window_func.__name__)

decorate(xlabel='Frequency (Hz)')
```



Упражнение 3.2

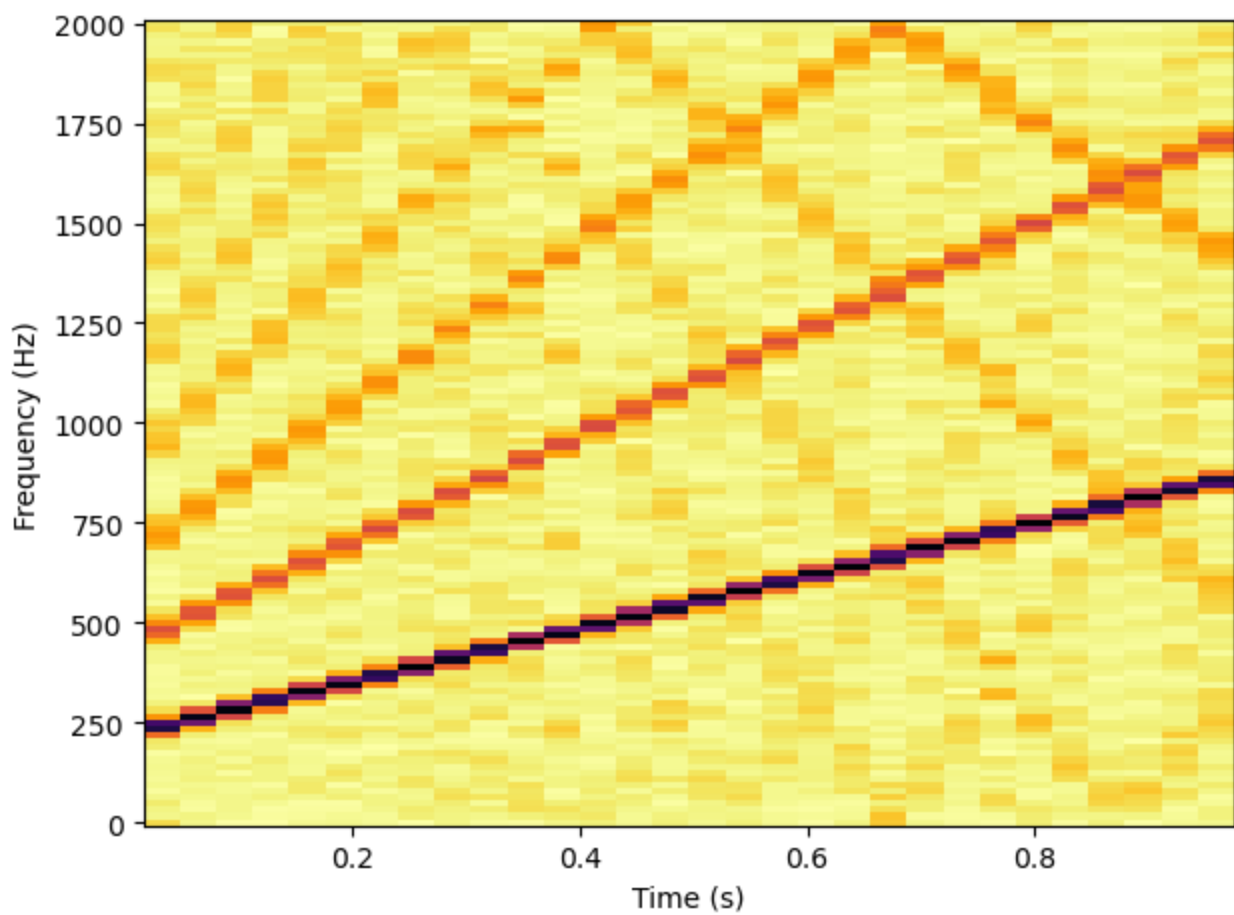
Напишите класс, называемый `SawtoothChirp`, расширяющий `Chirp` и переопределяющий `evaluate` для генерации пилообразного сигнала с линейно увеличивающейся частотой

```
In [ ]: class SawtoothChirp(Chirp):
    def evaluate(self, ts):
        freqs = np.linspace(self.start, self.end, len(ts))
        dts = np.diff(ts, prepend=0)
        dphis = PI2 * freqs * dts
        phases = np.cumsum(dphis)
        cycles = phases / (np.pi * 2)
        frac, _ = np.modf(cycles)
        ys = normalize(unbias(frac), self.amp)
        return ys

signal = SawtoothChirp(start=220, end=880)
wave = signal.make_wave(duration=1, framerate=4000)
wave.make_audio()
```

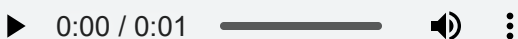
Out []:

```
In [ ]: sp = wave.make_spectrogram(256)
sp.plot()
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

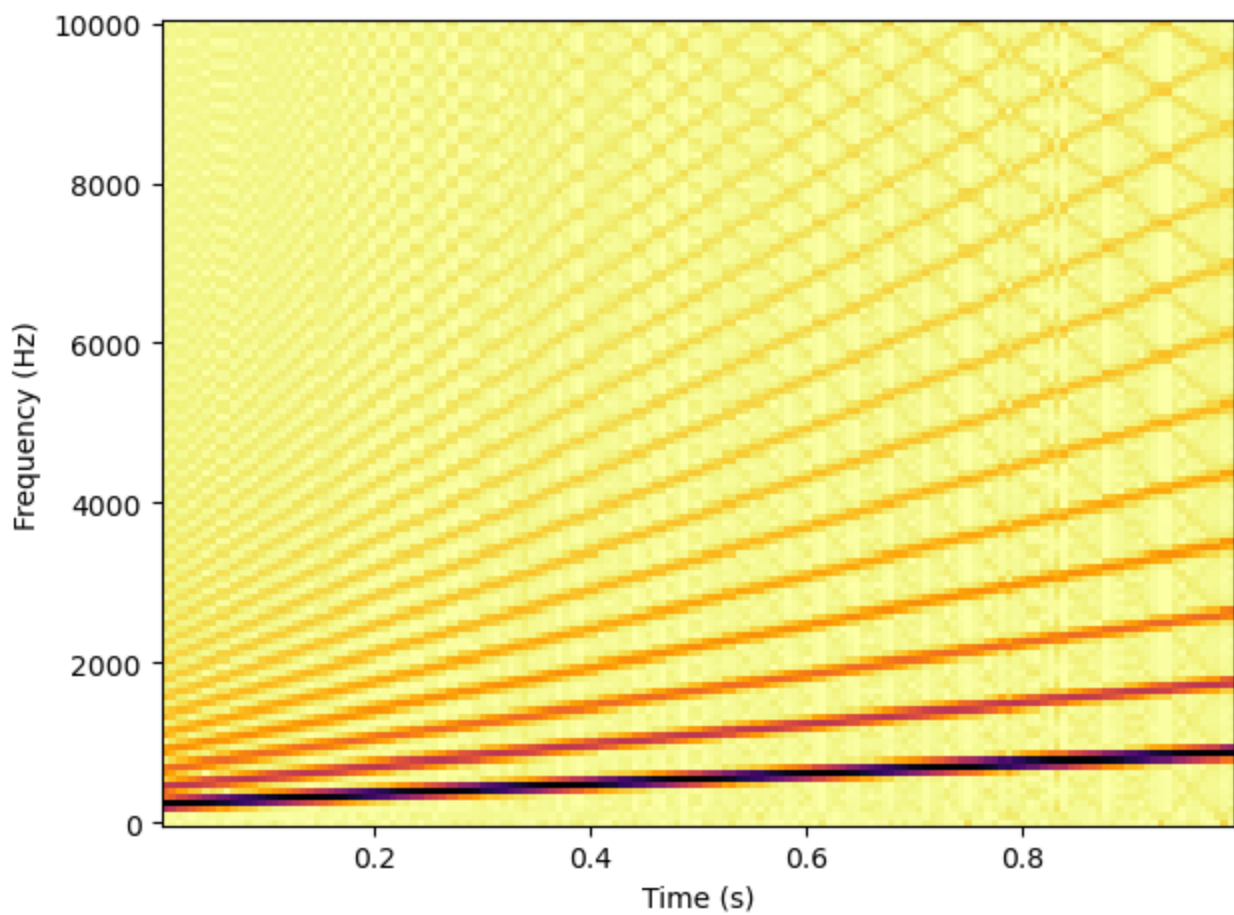


На спектрограмме можем заметить гармоники, которые отражаются от поворотной частоты в 2000 Гц. На звуке они слышны как небольшой шум. Если повысить количество кадров в секунду, этот шум пропадет

```
In [ ]: wave = signal.make_wave(duration=1, framerate=20000)
        wave.make_audio()
```

Out[]: 

```
In [ ]: sp = wave.make_spectrogram(256)
        sp.plot()
        decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```



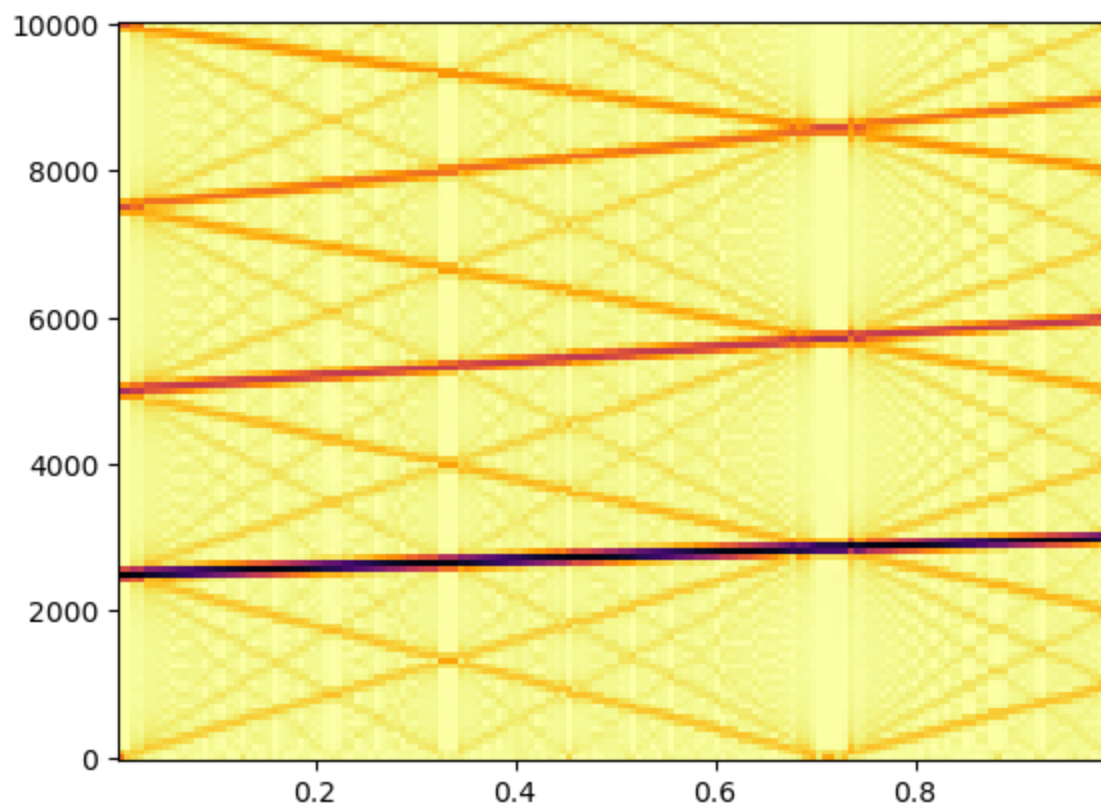
Упражнение 3.3

Создайте пилообразный чирп, меняющийся от 2500 до 300 Гц, и на его основе сгенерируйте сигнал длительностью 1 с и частотой кадров 20 кГц. Нарисуйте, каким примерно будет `Spectrum`. Затем распечатайте `Spectrum` и посмотрите, правы ли вы

```
In [ ]: signal = SawtoothChirp(start=2500, end=3000)
        wave = signal.make_wave(duration=1, framerate=20000)
        wave.make_audio()
```

Out[]:

```
In [ ]: sp = wave.make_spectrogram(256)
        sp.plot()
```

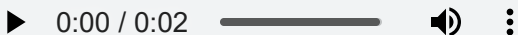


Упражнение 3.4

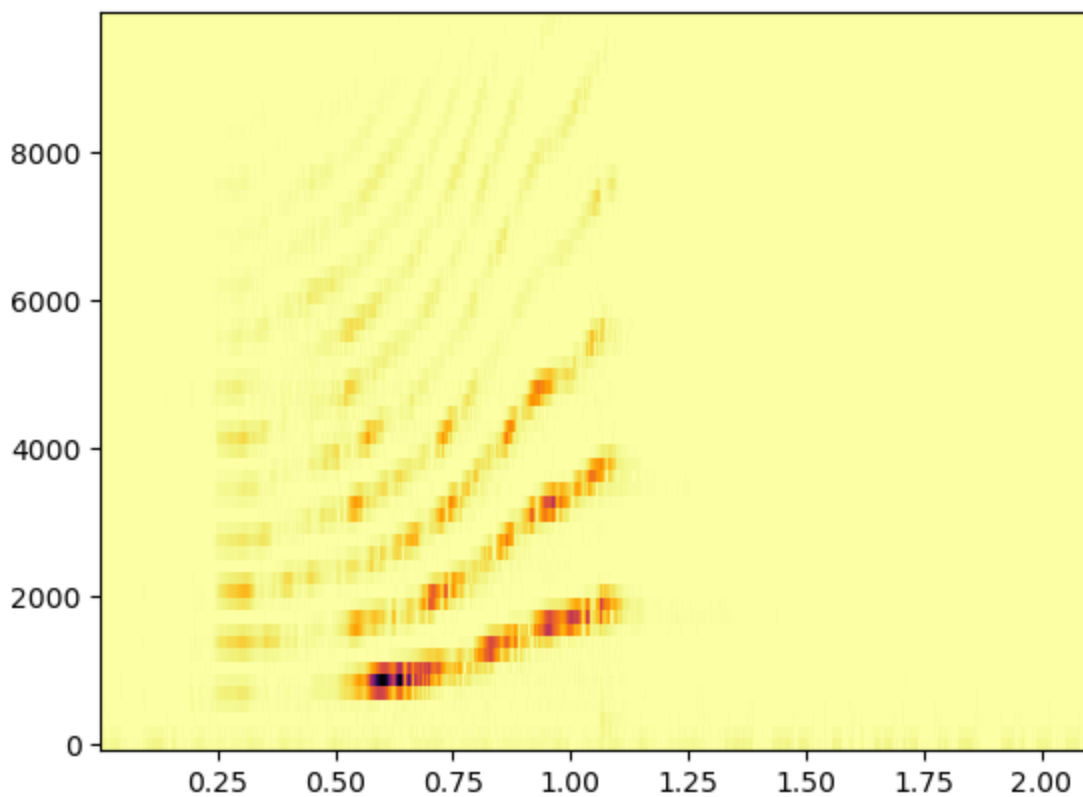
В музыкальной терминологии *глиссандо* - это нота, меняющаяся от одной высоты до другой, то есть своеобразный чирп

Найдите или запишите звук глиссандо и распечатайте спектрограмму первых нескольких секунд.

```
In [ ]: glissando = read_wave("glissando.wav")  
glissando.make_audio()
```

Out[]: 

```
In [ ]: glis_sp = glissando.make_spectrogram(256)  
glis_sp.plot(high=10000)
```



Полученная спектрограмма показывает, что чирп возрастает нелинейно

Упражнение 3.5

Тромбонист играет глиссаондо, непрерывно дует в мундштук и двигая кулису тромбона. При этом общая длина трубы меняется, а играемая нота обратно пропорциональна этой длине.

Если предположить, что музыкант двигает кулису с постоянной скоростью, как будет меняться во времени частота?

Напишите класс, называемый `TromboneGliss`, расширяющий `Chirp` и предоставляющий `evaluate`. Создайте сигнал, имитирующий глиссандо на тромбоне от C3 до F3, и обратно до C3. C3 - 262 Гц; F3 - 349 Гц.

Напечатайте спектрограмму полученного сигнала. На что похоже глиссандо на тромбоне - на линейный или же экспоненциальный чирп?

```
In [ ]: class TromboneGliss(Chirp):

    def evaluate(self, ts):
        l1, l2 = 1 / self.start, 1 / self.end
        lengths = np.linspace(l1, l2, len(ts))
        freqs = 1 / lengths

        dts = np.diff(ts, prepend=0)
        dphis = np.pi * 2 * freqs * dts
        phases = np.cumsum(dphis)
        ys = self.amp * np.cos(phases)

        return ys

gliss = TromboneGliss(262, 349)
```

```
wave1 = gliss.make_wave(duration=1)
wave2 = TromboneGliss(349, 262).make_wave(duration=1)

wave1.apodize()
wave2.apodize()

(wave1 | wave2).make_audio()
```

Out[]:

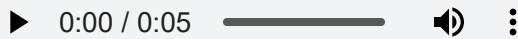


Упражнение 3.5

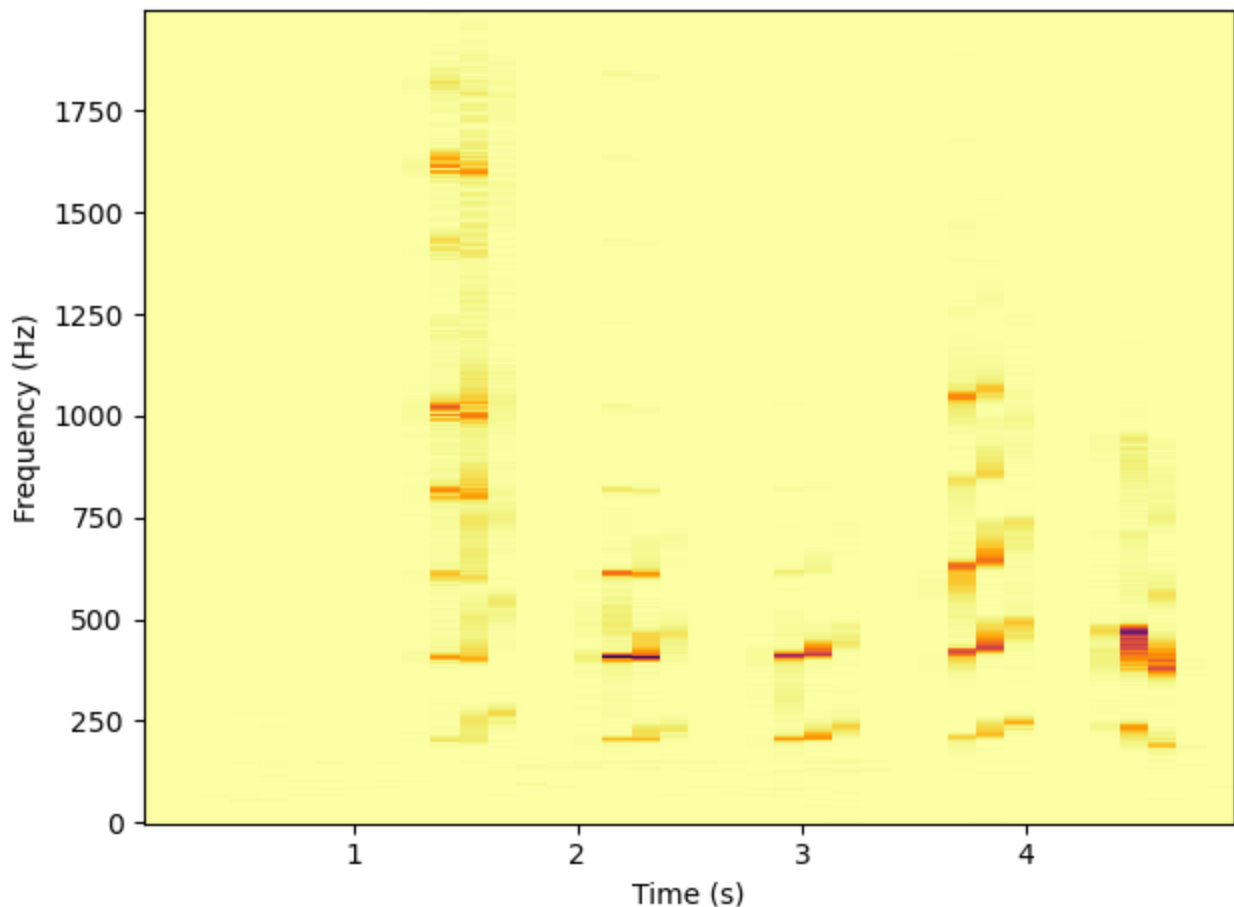
Сделайте или найдите запись серии гласных звуков и посмотрите на спектрограмму. Сможете ли вы различить разные гласные.

```
In [ ]: wave = read_wave('vowels.wav')
wave.make_audio()
```

Out[]:



```
In [ ]: wave.make_spectrogram(2048).plot(high=2000)
decorate(xlabel='Time (s)', ylabel='Frequency (Hz)')
```

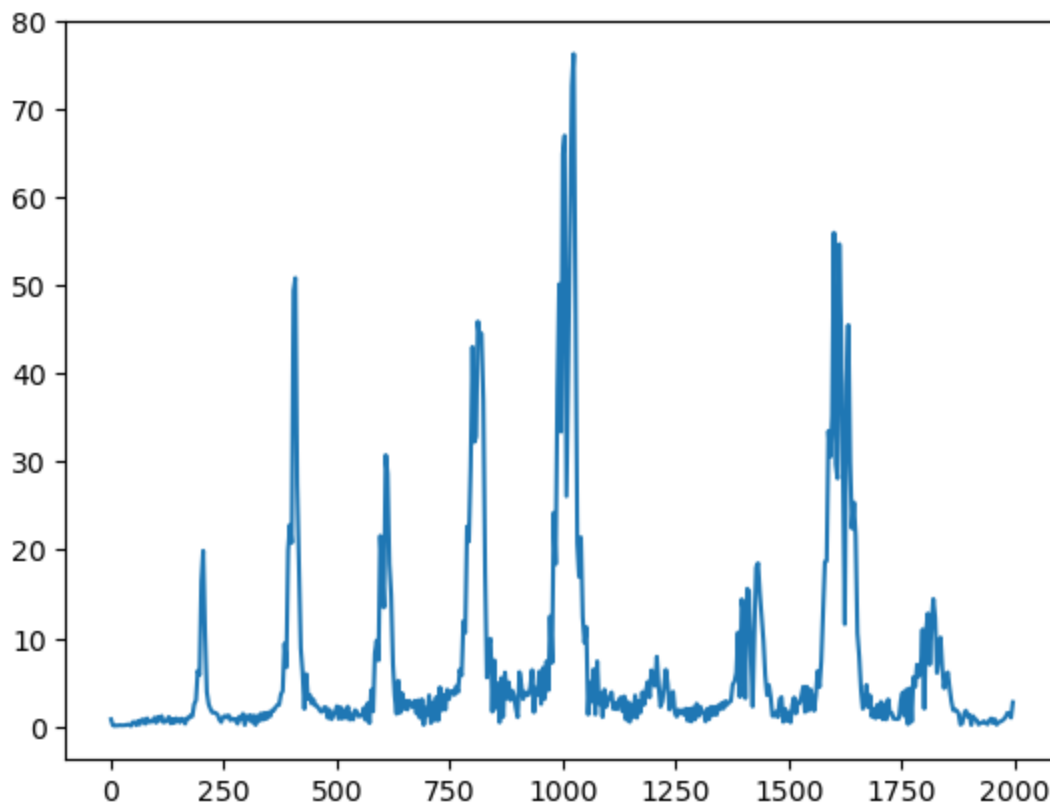


Столбцы в спектрограмме это моменты, в которых есть звуки. Каждый столбец состоит из полосочек, цвет которых зависит от амплитуды определенной частоты. Самую темную полосу (с самой большой

амплитудой) принято считать формантой. Форманта связана с уровнем частоты голоса и образует тембр

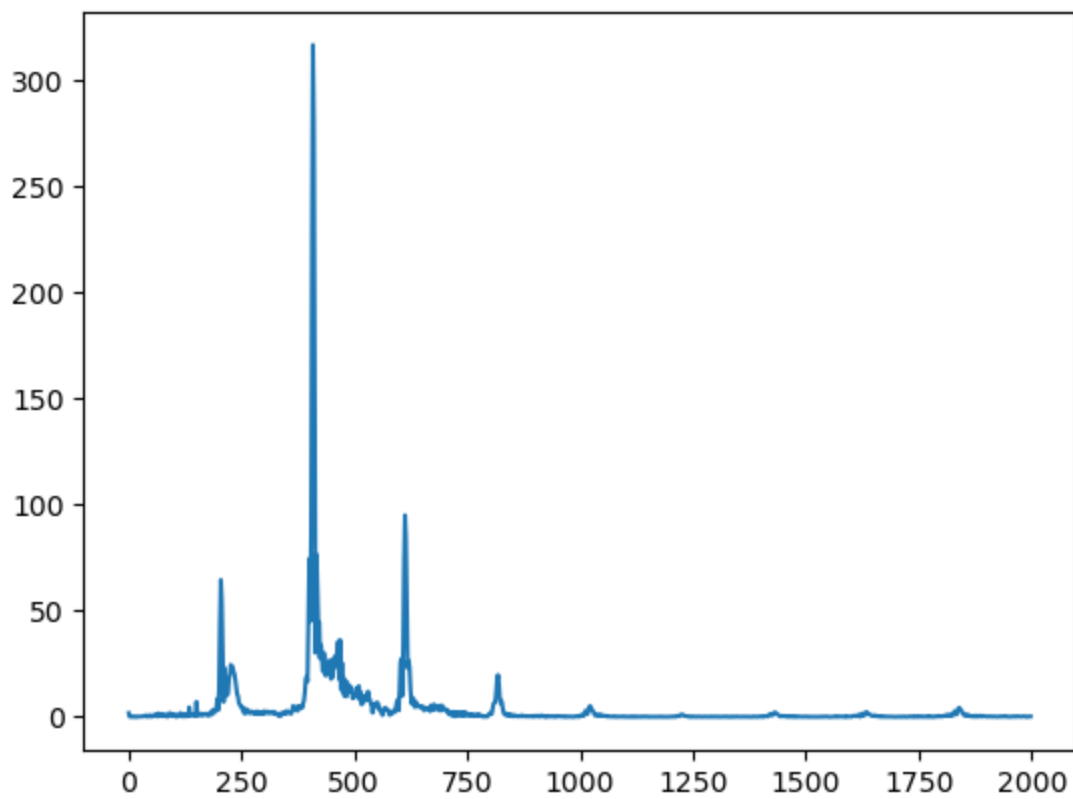
Посмотрим спектры каждой из букв отдельно

```
In [ ]: segment = wave.segment(start=1.25, duration=0.25)
segment.make_spectrum().plot(high=2000)
```



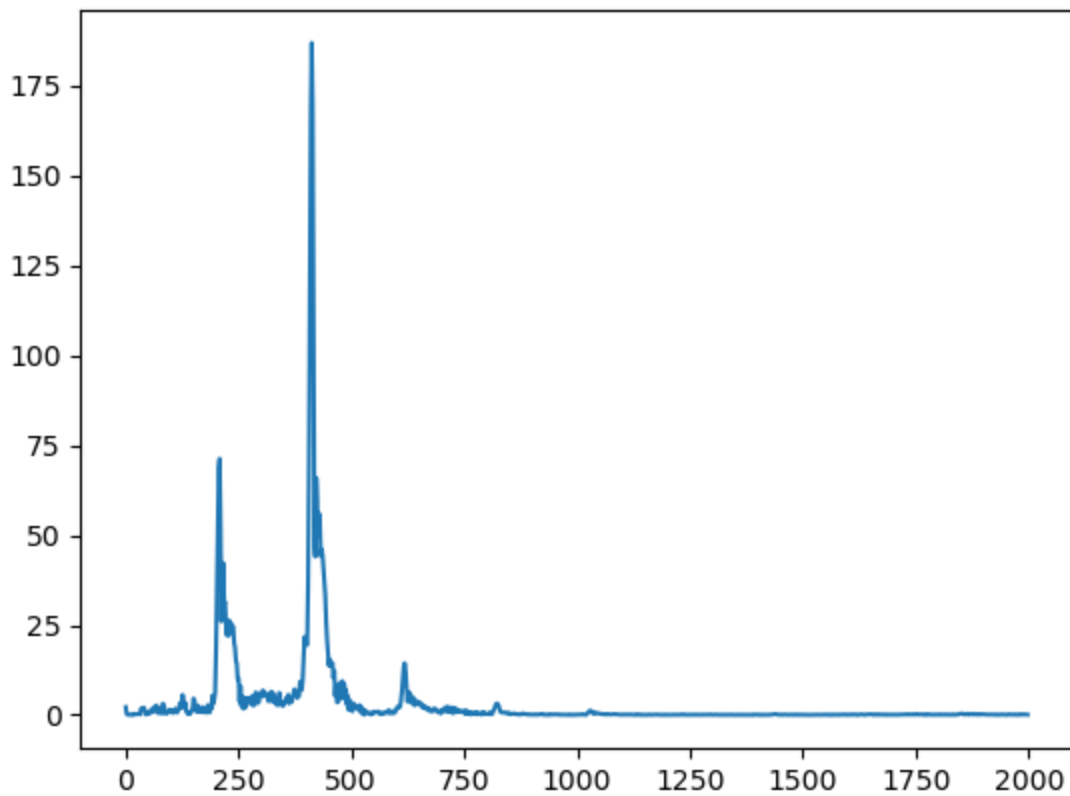
Форманта - 1000 Гц

```
In [ ]: segment = wave.segment(start=2, duration=0.5)
segment.make_spectrum().plot(high=2000)
```

Форманта около 400 Гц

```
In [ ]: segment = wave.segment(start=2.75, duration=0.5)
segment.make_spectrum().plot(high=2000)
```



Форманта около 400 Гц

Сравнивая полученные спектры и спектрограммы с соответствующими графиками в учебнике можем заметить, что звук "и" содержит меньше всего различных частот в то время как "а" больше всего.

