Санкт-Петербургский Государственный Политехнический Университет
Институт Компьютерных Наук и Технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №3
на тему
Апериодические сигналы

Работу выполнил
Студент группы 3530901/80203
Тарасенко Н.С.
Преподаватель
Богач Н.В.

Санкт-Петербург, 2021 год

1 Настройка проекта

Перед тем как выполнять задания необходимо настроить проект и сделать все необходимые импорты:

```
from __future__ import print_function, division

%matplotlib inline

import thinkdsp as tdsp
import thinkplot
import numpy as np
import math

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets
```

Рис. 1: 2

2 Упражнение номер №1

Необходимо посмотреть пример утечки и заменить окно Хэмминга одним из окон предоставляемых numPy, посмотреть, как они влияют на утечку.

Рассмотрим пример утечки, выведем его спектр:

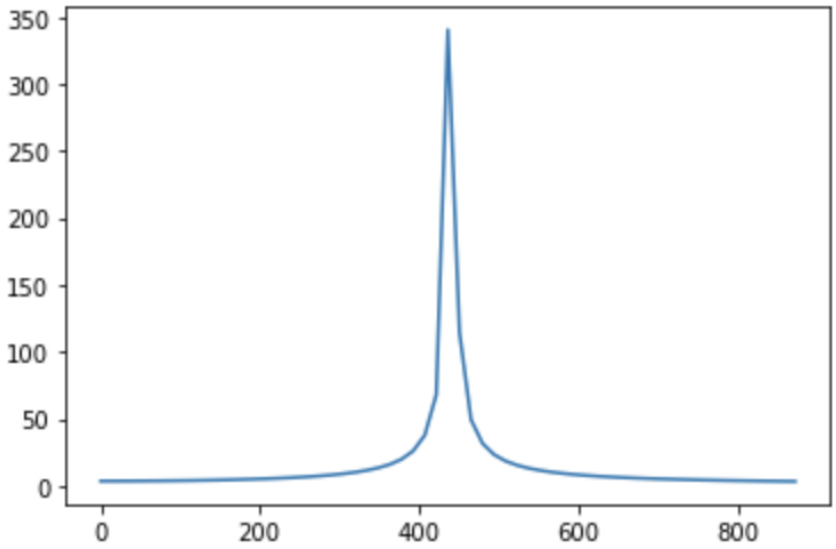

Рис. 2: 2

Сделаем представление ввиде других 4 окон и наложим полученные спектры:

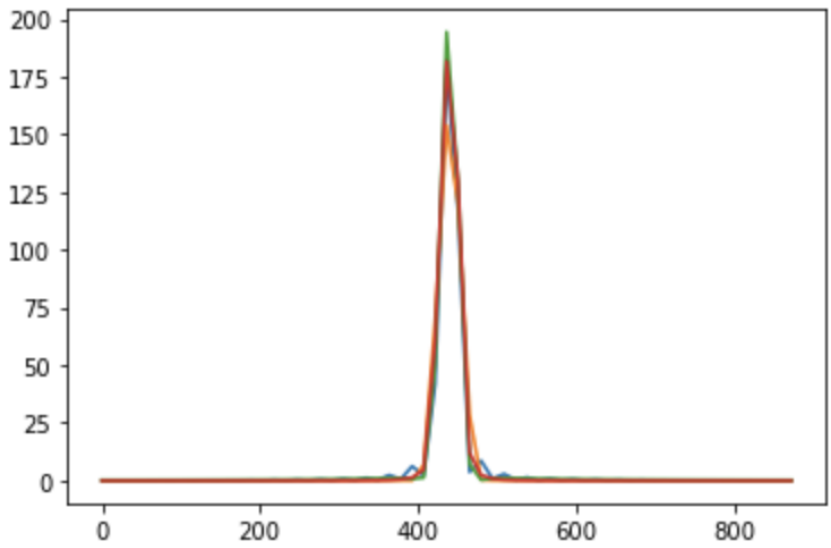

Рис. 3: 2

Все четыре хорошо справляются с уменьшением утечки. Фильтр Бартлетта оставляет некоторый остаточный «звон». Фильтр Хэмминга рассеивает наименьшее количество энергии.

3 Упражнение номер №2

1) Необходимо создать класс SawtoothChirp, расширяющий Chirp и переопределяющий evaluate для генерации пилообразного сигнала с линейно увеличивающийся/уменьшающейся частотой. 2) Необходимо нарисовать эскиз спектрограммы этого сигнала и распечатать его. Попытаться услышать эффект биения

Реализуем класс SawtoothChirp

```
PI2 = 2 * math.pi

class SawtoothChirp(thinkdsp.Chirp):
    def evaluate(self, ts):
        freqs = np.linspace(self.start, self.end, len(ts) - 1)
        dts = np.diff(ts)
        dphis = PI2 * freqs * dts
        phases = np.cumsum(dphis)
        phases = np.insert(phases, 0, 0)
        cycles = phases / PI2
        frac, _ = np.modf(cycles)
        ys = self.amp * frac
        return ys
```

Рис. 4: 2

Используя класс, который мы реализовали, создадим звук:

```
signal = SawtoothChirp(start=500, end=1300)
wave = signal.make_wave(duration=1, framerate=10000)
wave.apodize()
wave.make_audio()
```

Рис. 5: 2

Распечатаем спектрограмму получившегося звука:

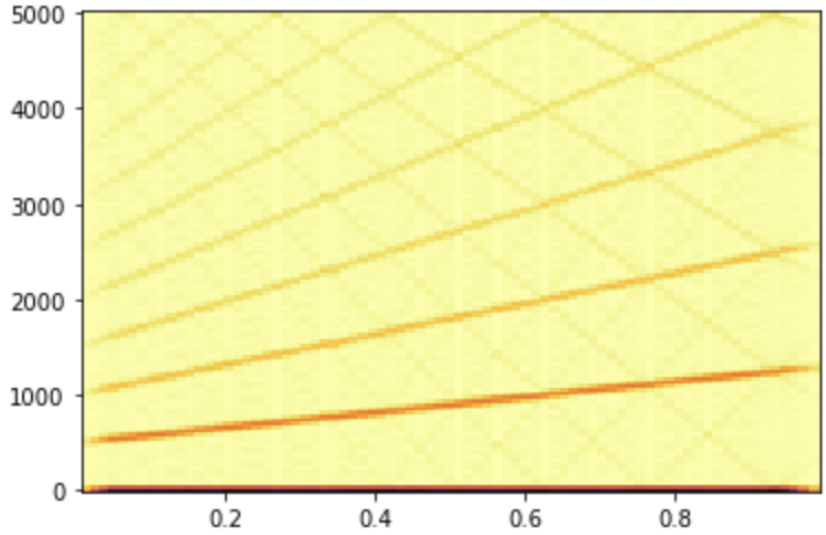

Рис. 6: 2

При относительно низкой частоте кадров вы можете увидеть, как гармоники с наложенными частотами отражаются от частоты сворачивания.

4 Упражнение номер №3

Необходимо создать пилообразный чирп, меняющийся от 2500 до 3000 Гц, и на его основе сгенерировать сигнал длительностью 1с и частотой кадров 20кГц. Распечатать спектр.

Поскольку основная частота колеблется от 2500 до 3000 Гц, мы ожидаем увижеть всплеск. Первая гармоника колеблется от 5000 до 6000 Гц, поэтому в даннос случае всплеск будет меньше чем в первом случае. Следующая гармоника колеблется от 7500 до 9000 значит всплеск будет еще меньше чем во 2 случае.

Другие гармоники повсюду накладываются друг на друга, и образуют энергию. Эта распределенная энергия создает интересные звуки.

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

Рис. 7: 2

Распечатаем получившийся спектр:

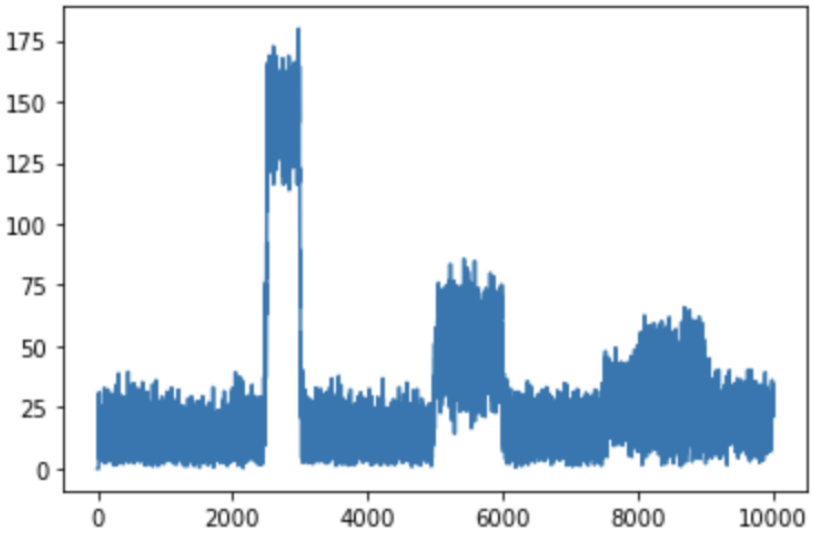

Рис. 8: 2

5 Упражнение номер №4

Необходимо найти или записать звук глиссандо и распечатать спектрограмму первых нескольких секунд:

Для этого упражнения я взял звук из данного нам репозитория и обрезал его до первых 11 секунд:

```
wave = thinkdsp.read_wave('gl.wav')
wave.make_audio()
```

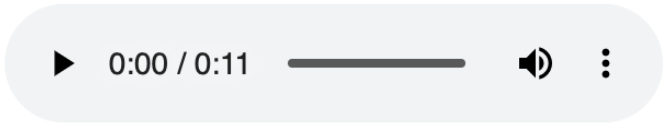

Рис. 9: 2

Рассмотрим спектрограмму:

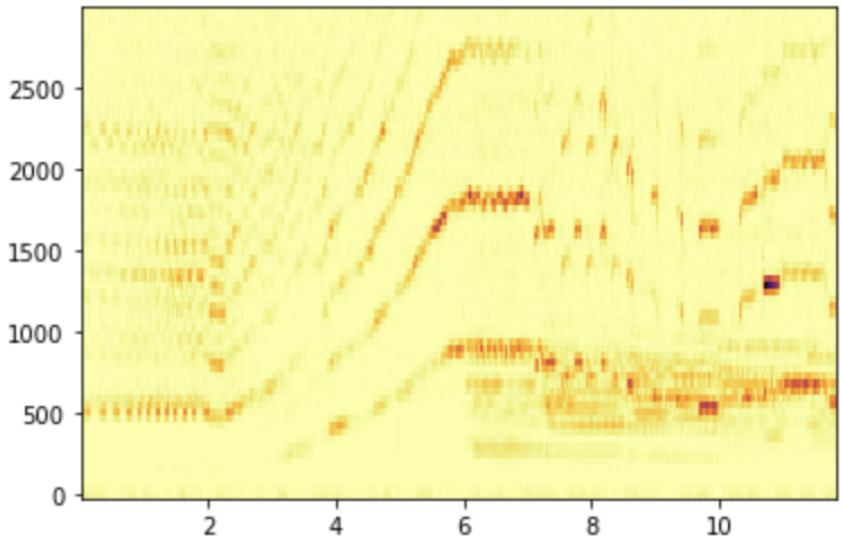

Рис. 10: 2

6 Упражнение номер №5

Необходимо проанализировать как будет меняться во времени частота, при условии, что музыкант двигает кулису с постоянной скоростью. Необходимо реализовать класс TromboneGliss, расширяющий chirp и предоставляющий evaluate. Создать сигнал имитирующий глиссандо на тромбоне от C3 до F3, и обратно до C3. Распечатать спектрограмму полученного сигналаи определить на что похоже глиссандо на линейный или экспоненциальный чирп.

Реализуем наш класс, которы будет представлять из себя сигнал схожий с тромбоподобными, имеющий переменную частоту.

```
: class TromboneGliss(thinkdsp.Chirp):
    def _evaluate(self, ts):
        l1, l2 = 1.0 / self.start, 1.0 / self.end
        lengths = np.linspace(l1, l2, len(ts)-1)
        freqs = 1 / lengths
        return self._evaluate(ts, freqs)
```

Рис. 11: 2

Рассмотрим созданный нами сигнал , возьмем лишь первую его часть:

```
low = 262
high = 349
signal = TromboneGliss(high, low)
wave1 = signal.make_wave(duration=1)
wave1.apodize()
wave1.make_audio()
```

Рис. 12: 2

Рассмотрим вторую часть сигнала:

```
signal = TromboneGliss(low, high)
wave2 = signal.make_wave(duration=1)
wave2.apodize()
wave2.make_audio()
```

Рис. 13: 2

И наконец соединим обе части и получим конечный результат:

Для получения конечного результата соединим первую и вторую части сигнала

```
B [67]: wave = wave2 | wave1
        wave.make_audio()
```

Out[67]: 0:00 / 0:02

Рис. 14: 2

Выведем спектрограмму:

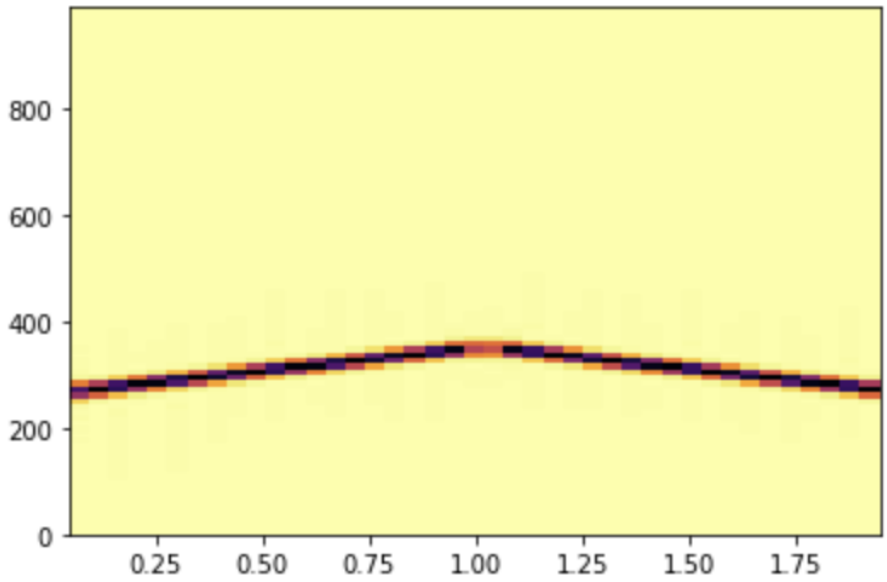

Рис. 15: 2

Как видно из полученного графика и звуков, частота сначала возрастает, а потом снова уменьшается до начальной, следовательно, написанный класс работает корректно. К сожалению, с такой маленькой разницей в начальной и конечной частотах невозможно определить на что похоже глиссандо на тромбоне - на линейный или экспоненциальный чирп.

7 Упражнение номер №6

Для данного упражнения снова возьмем звук из предложенного нам репозитория. На нем мужчина произносит гласные звуки с эффектом "эхо"

```
wave = thinkdsp.read_wave('task6.wav')
wave.make_audio()
```

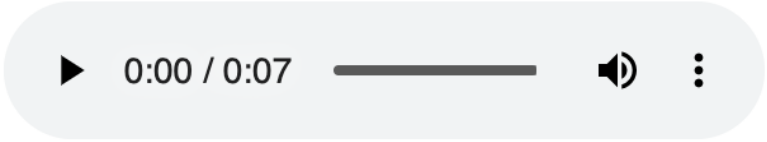

Рис. 16: 2

Рассмотрим спектрограмму данного звука:

```
wave.make_spectrogram(1024).plot(high=1000)
```

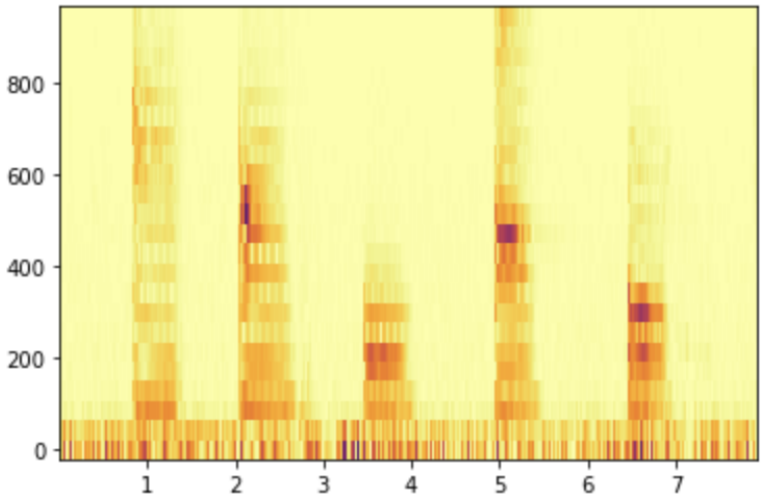

Рис. 17: 2

Полоса внизу, вероятно, является фоновым шумом. Пики на спектрограмме называются формантами.

Распечатаем спектр, выделив сегмент во время "а":

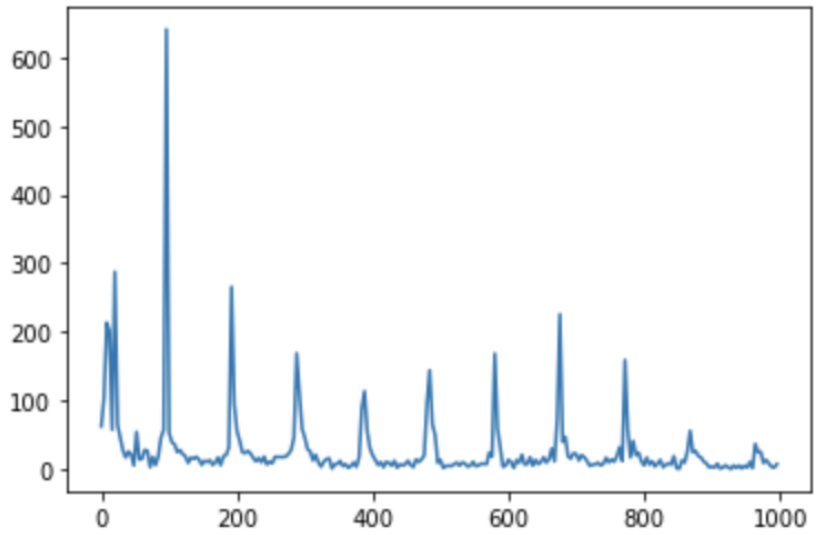

Рис. 18: 2

Мы можем увидеть форматы уже более четче.
Выведем спектры и для остальных звуков:

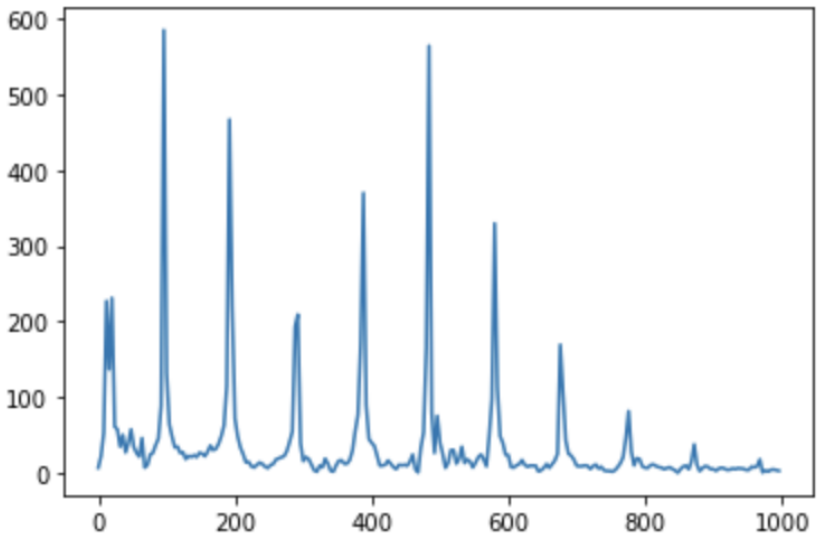

Рис. 19: 2

Сегмент «э» имеет высокоамплитудную форманту около 500 Гц.

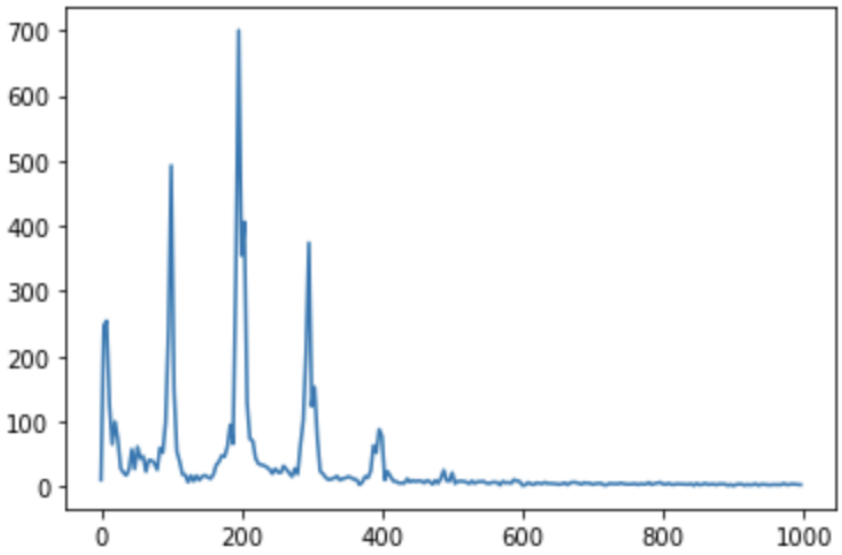

Рис. 20: 2

В сегменте "и"нет высокочастотных составляющих.

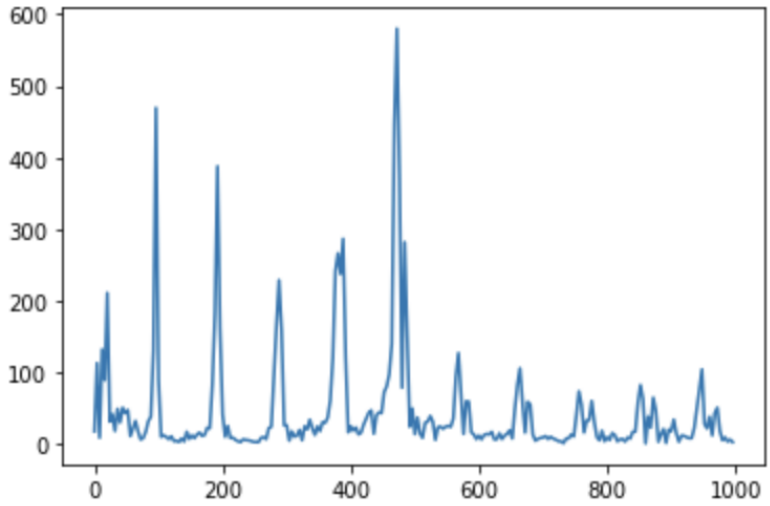

Рис. 21: 2

Сегмент «о» имеет высокоамплитудную форманту около 500 Гц, даже выше основной гармоники.

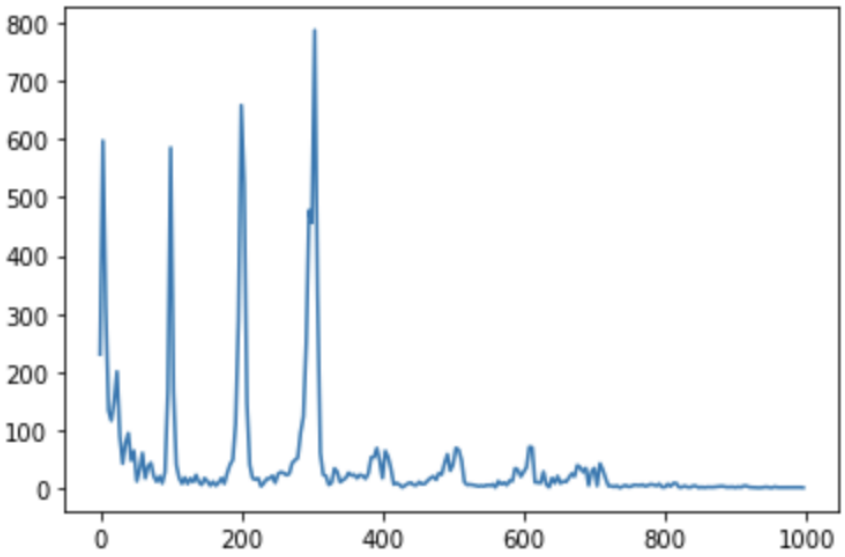

Рис. 22: 2

Сегмент "у"имеет высокоамплитудную форманту около 300 Гц и не имеет высокочастотных составляющих.