Лабораторная работа №10
Линейные стационарные системы

Смирнов Никита

6 мая 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Упражнение 10.1

Усечём сигналы до 2^{16} значений, а затем обнудим их до 2^{17}.

Я взял запись выстрела и сделал над ним изменения.

```
response = thinkdsp.read_wave('180960__kleeb__gunshot.wav')

start = 0.12
response = response.segment(start=start)
response.shift(-start)

response.truncate(2**16)
response.zero_pad(2**17)

response.normalize()
response.plot()
thinkplot.config(xlabel='Time (s)', ylim=[-1.05, 1.05])
```

Листинг 1.1: Усечение сигнала

Получил следующее изображение:

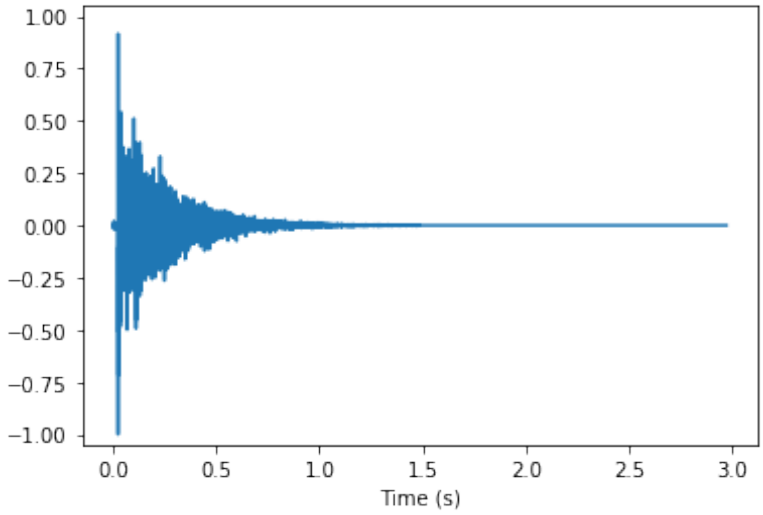

Рис. 1.1: Визуализация сигнала

Вычисляю спектр:

```
transfer = response.make_spectrum()
transfer.plot()
thinkplot.config(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 1.2: Спектр сигнала

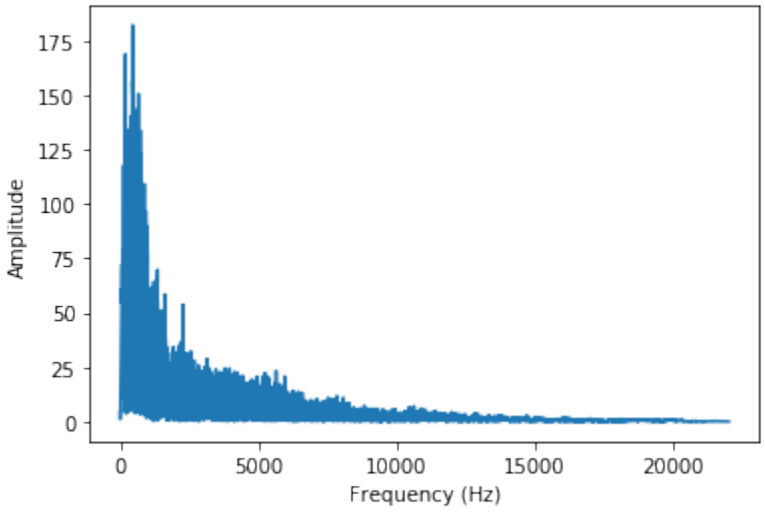

Рис. 1.2: Спектр сигнала

Создаю другой сигнал:

```
violin = thinkdsp.read_wave('92002__jcveliz__violin-origional.wav')

start = 0.11
violin = violin.segment(start=start)
violin.shift(-start)

violin.truncate(2**16)
violin.zero_pad(2**17)

violin.normalize()
violin.plot()
thinkplot.config(xlabel='Time (s)', ylim=[-1.05, 1.05])
```

Листинг 1.3: Усечение сигнала

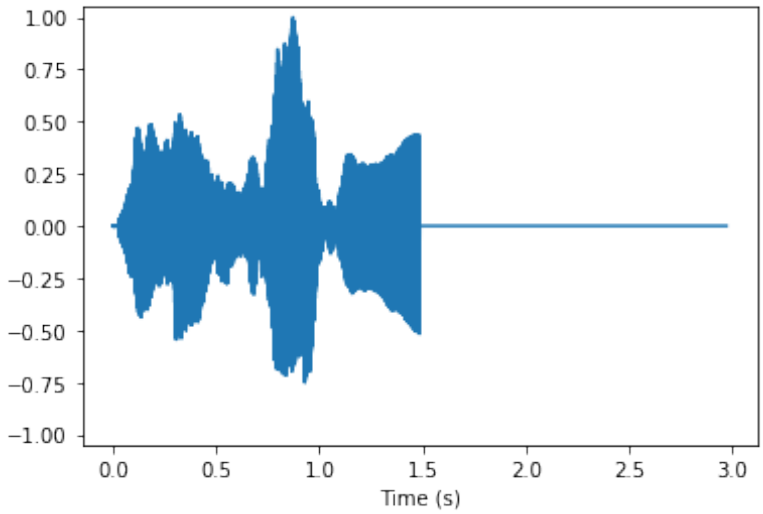

Рис. 1.3: Визуализация сигнала

Составим спектр:

```
spectrum = violin.make_spectrum()
```

Листинг 1.4: Спектр сигнала

Теперь умножим ДПФ сигнала на передаточную функцию и преобразуем обратно в волну:

```
output = (spectrum * transfer).make_wave()
output.normalize()
```

```
output.plot()
```

Листинг 1.5: Совмещение сигнала

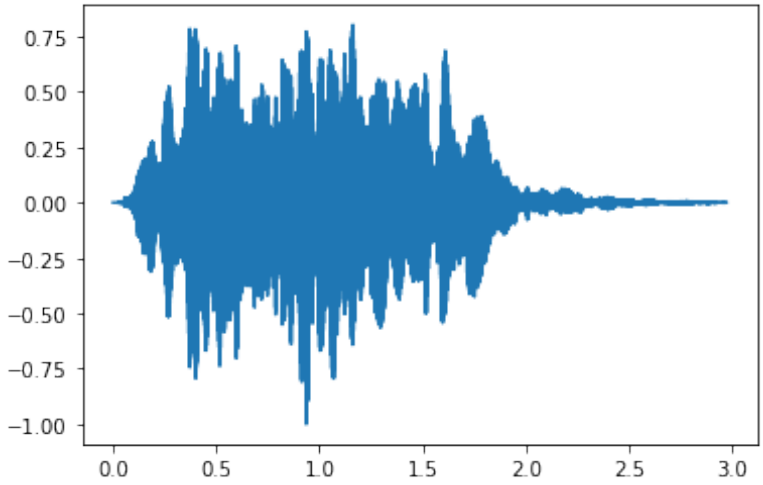

Рис. 1.4: Визуализация сигнала

Лишнюю ноту в начале не слышно. Избавимся от нулевого отступа:

```
response.truncate(2**16)
response.plot()

violin.truncate(2**16)
violin.plot()
```

Листинг 1.6: Избавление от нулевого отступа

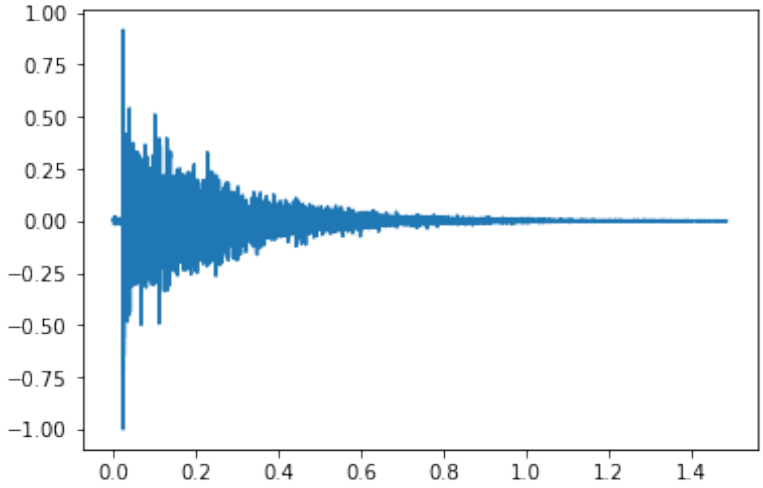

Рис. 1.5: Визуализация сигнала

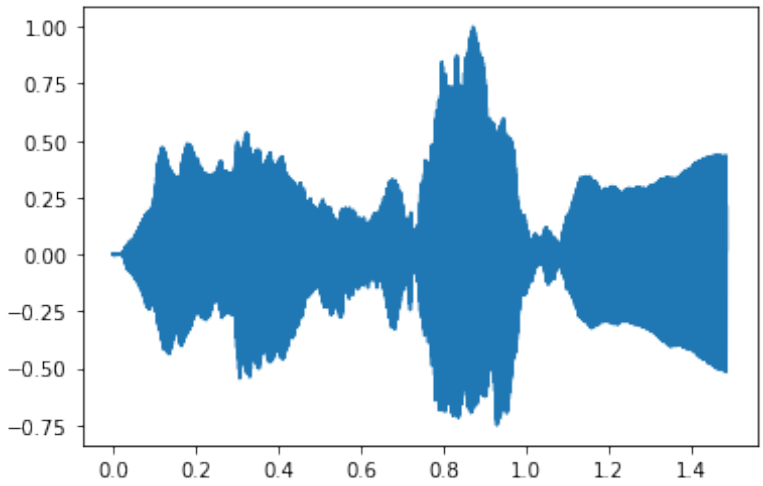

Рис. 1.6: Визуализация сигнала

Теперь мы можем сравнить с `np.convolve`:

```
output2 = violin.convolve(response)
output2.plot()
```

Листинг 1.7: Объединение сигналов

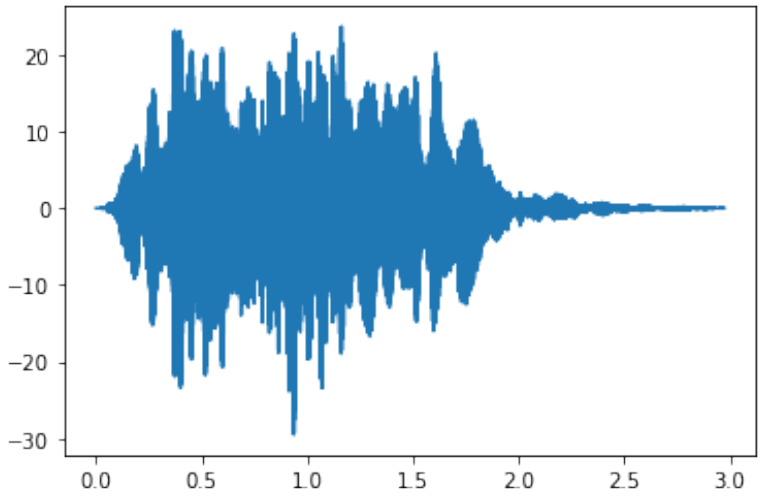

Рис. 1.7: Визуализация сигнала

Результаты похожи. Звук такой же, но длина не та.

```
len(output), len(output2)
```

Листинг 1.8: Сравнение длин сигналов

На выходе получились значения `(131072, 131071)`.

`scipy.signal.fftconvolve` делает то же самое, но, как следует из названия, он использует ДПФ, поэтому он значительно быстрее:

```
import scipy.signal
ys = scipy.signal.fftconvolve(violin.ys, response.ys)
output3 = thinkdsp.Wave(ys, framerate=violin.framerate)
```

Листинг 1.9: Изменение сигнала

```
output3.plot()
```

Листинг 1.10: Визуализация сигнала

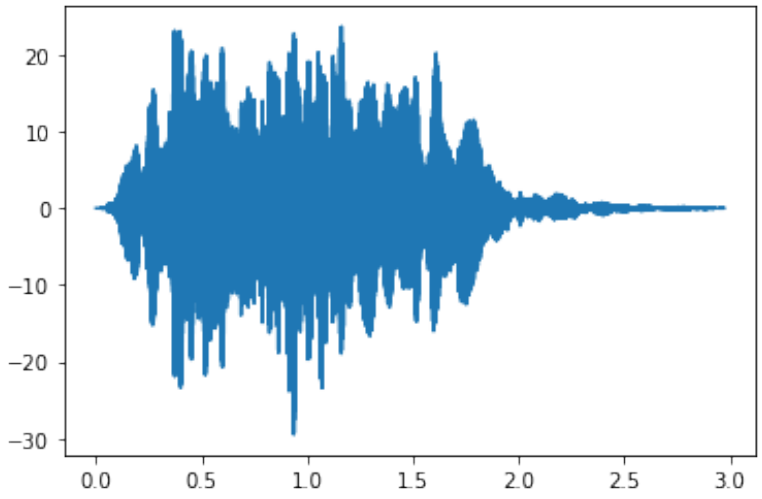

Рис. 1.8: Визуализация сигнала

Результат тот же. В пределах ошибки с плавающей запятой результаты такие же:

```
output2.max_diff(output3)
```

Листинг 1.11: Сравнение результатов

Ошибка равна `2.1316282072803006e-14`.

Глава 2

Упражнение 10.2

Для начала я взял запись, которая будет использована в качестве импульсной характеристики.

```
response = thinkdsp.read_wave('mono.wav')

start = 0
duration = 5
response = response.segment(duration=duration)
response.shift(-start)

response.normalize()
response.plot()
thinkplot.config(xlabel='Time (s)', ylim=[-1.05, 1.05])
```

Листинг 2.1: Загрузка звука

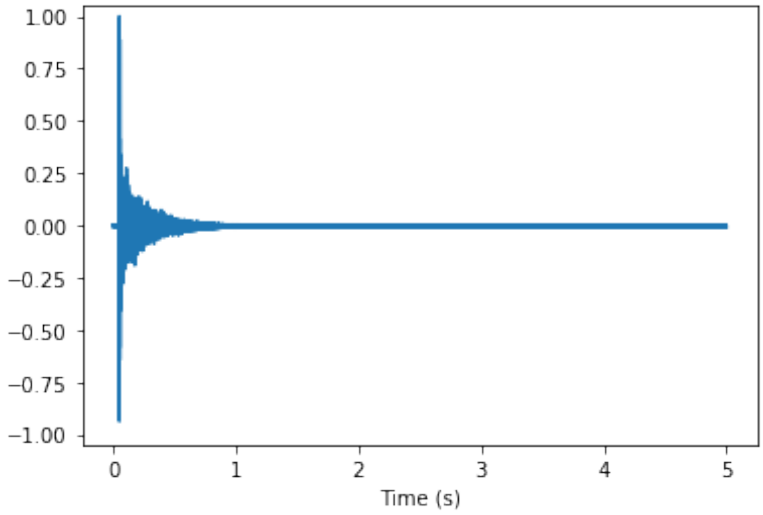

Рис. 2.1: Визуализация звука

```
transfer = response.make_spectrum()
transfer.plot()
thinkplot.config(xlabel='Frequency (Hz)', ylabel='Amplitude')
```

Листинг 2.2: Спектр звука

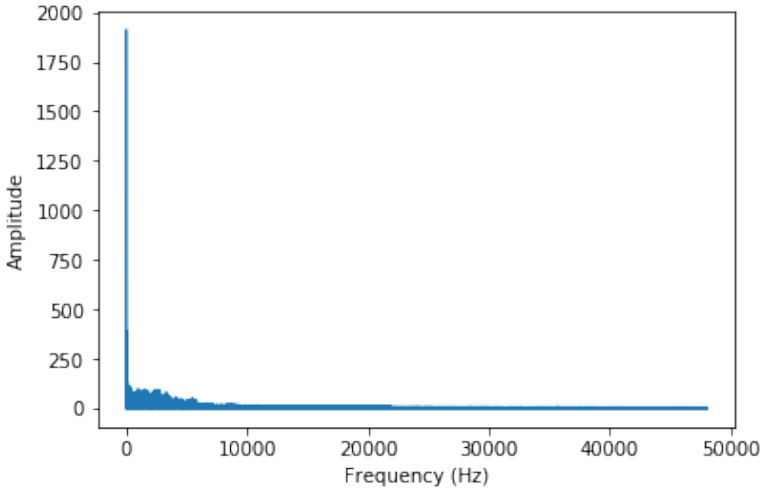

Рис. 2.2: Спектр звука

Рассмотрим передаточную функцию:

```
transfer.plot()
thinkplot.config(xlabel='Frequency (Hz)', ylabel='Amplitude',
                 xscale='log', yscale='log')
```

Листинг 2.3: Визуализация звука

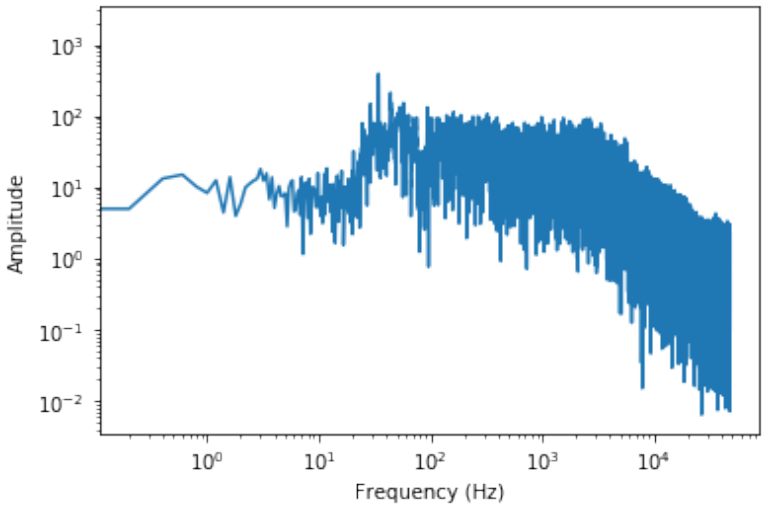

Рис. 2.3: Визуализация звука

Беру вторую запись с такой же частатой дескритизации:

```
wave = thinkdsp.read_wave('38849.wav')

start = 0.0
wave = wave.segment(start=start)
wave.shift(-start)

wave.truncate(len(response))
wave.normalize()
wave.plot()
thinkplot.config(xlabel='Time (s)', ylim=[-1.05, 1.05])
```

Листинг 2.4: Загрузка звука

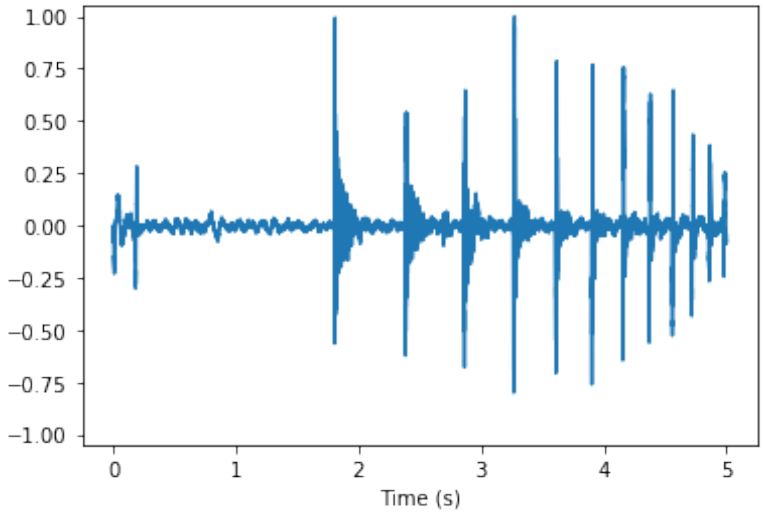

Рис. 2.4: Визуализация звука

Смотрим частоту дискретизации.

```
from pydub import AudioSegment
song = AudioSegment.from_mp3("mono.wav")
song.frame_rate
```

Листинг 2.5: Частота дискретизации 1

```
song2 = AudioSegment.from_mp3("38849.wav")
song2.frame_rate
```

Листинг 2.6: Частота дискретизации 2

Оба звука имеют частоту дискретизации 96000.
Теперь мы вычисляем ДПФ.

```
spectrum = wave.make_spectrum()
```

Листинг 2.7: Спектр звука

Обрежем запись до той же длины, что и импульсная характеристика:

```
len(spectrum.hs), len(transfer.hs)
```

Листинг 2.8: Длина записей

Длины совпадают и равны `(240001, 240001)`.

```
spectrum.fs
```

Листинг 2.9: Длина первой записи

```
transfer.fs
```

Листинг 2.10: Длина второй записи

Массивы значений совпадают и равны `array([ 0. , 0.2, 0.4, ..., 47999.6, 47999.8, 48000. ])`.

Мы можем умножить в частотной области и преобразовать обратно во временную область.

```
output = (spectrum * transfer).make_wave()
output.normalize()
```

Листинг 2.11: Перемножение

Рассмотрим сравнение оригинальной и преобразованной записи:

```
wave.plot()
```

Листинг 2.12: Визуализация оригинального звука

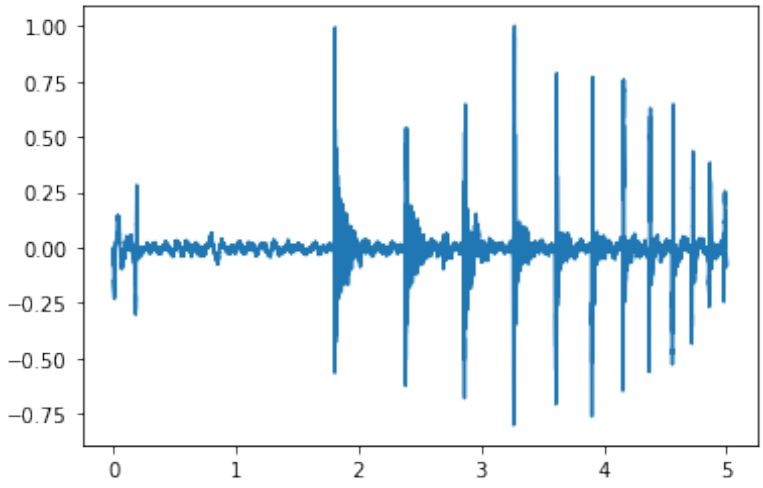

Рис. 2.5: Визуализация оригинального звука

```
output.plot()
```

Листинг 2.13: Визуализация преображённого звука

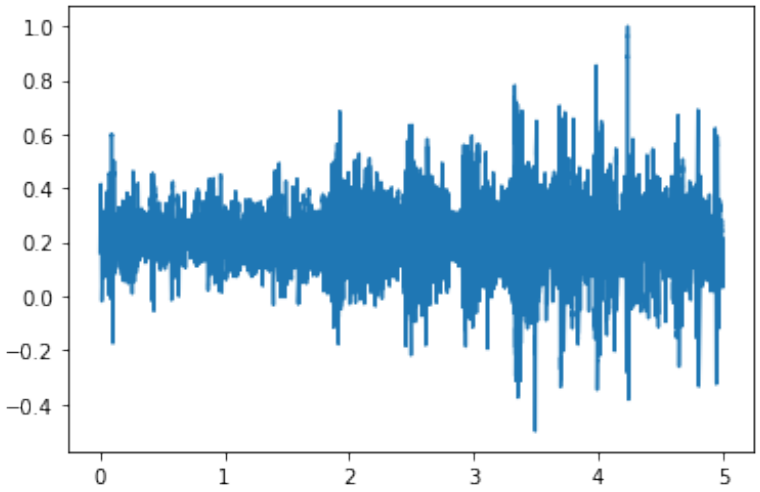

Рис. 2.6: Визуализация преображённого звука

Теперь, когда мы распознаем эту операцию как свёртку, мы можем вычислить ее с помощью метода `convolve`:

```
convolved2 = wave.convolve(response)
convolved2.normalize()
convolved2.make_audio()
```

Листинг 2.14: Моделирование при помощи `convolve`

Глава 3

Выводы

Во время выполнения лабораторной работы получены навыки работы с теорией сигналов и систем, а также применения теоремы свёртки, характеризуя линейные, инвариантные во времени системы.