Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Отчет по лабораторной работе №1
Сигналы телекоммуникационных систем

Работу выполнил:
Чугунов А.А.
Группа: 33501/4
Преподаватель:
Богач Н.В.

Санкт-Петербург
2017

1. Цель работы

Познакомиться со средствами генерации и визуализации простых сигналов.

2. Теоретическая информация

Аналоговый сигнал с математической точки зрения представляет собой функцию (как правило - функцию времени), и при его дискретизации мы получаем отсчеты, являющиеся значениями этой функции, вычисленными в дискретные моменты времени. Поэтому для расчета дискретезированного сигнала необходимо прежде всего сформировать вектор дискретных значний времени. Сформировав его, можно вычислять значения сигнала, используя этот вектор в различных фомрулах.

3. Ход выполнения работы

Для начала научимся строить простейшие сигналы и изучим саму процедуру построения. Сформируем следующий сигнал при помощи языка *Pyton*:

Листинг 1: Plot1.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

Fdiscrete = 8e3
t = np.linspace(0, 1, int(Fdiscrete))
A = 2
f0 = 1e3
phi = np.pi / 4
s1 = A * np.cos(2 * np.pi * f0 * t + phi)
alpha = 1000
s2 = np.exp(-alpha * t) * s1
plt.figure(0)
plt.subplot(2, 2, 1)
plt.plot(s2[0:100])
plt.grid()
plt.subplot(2, 2, 2)
plt.stem(s2[0:100])
plt.grid()
plt.subplot(2, 2, 3)
plt.plot(s2[0:100], '.')
plt.grid()
plt.subplot(2, 2, 4)
plt.step(t[0:100], s2[0:100])
plt.grid()
plt.show()
```

Получаем следующие результаты(Рисунок. 3.1):

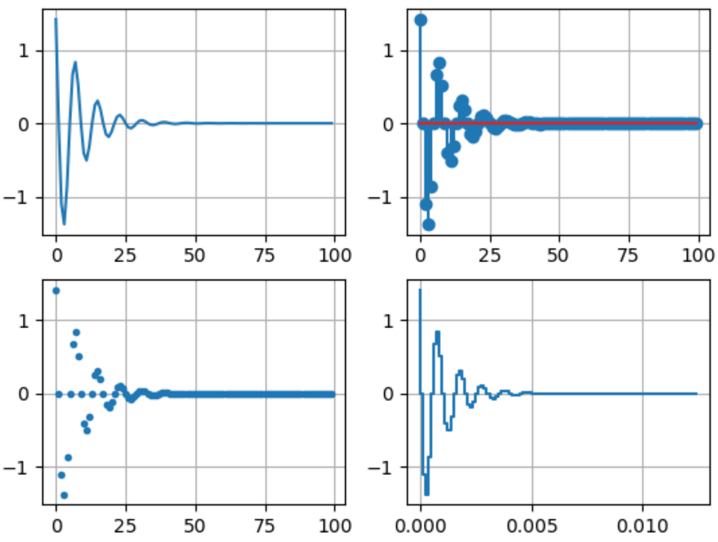

Рисунок 3.1. Гармонический сигнал представленный различными графическими функциями

У нас вышел гаромонический сигнал, который затухает по экспоненте из-за домножения его на экспоненту.

Продолжая изучать графические возможности пакета, попробуем построить косинусы различной частоты(Рисунок. 3.2):

Листинг 2: Plot2.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal


Fdiscrete = 8e3
t = np.linspace(0, 1, int(Fdiscrete))
f = np.asarray([600, 800, 1000, 1200, 1400])
s3 = [[np.cos(2 * np.pi * fi * ti) for ti in t] for fi in f]
plt.figure(1)
for si in s3: plt.plot(si[0:100])
plt.grid()
plt.show()
```

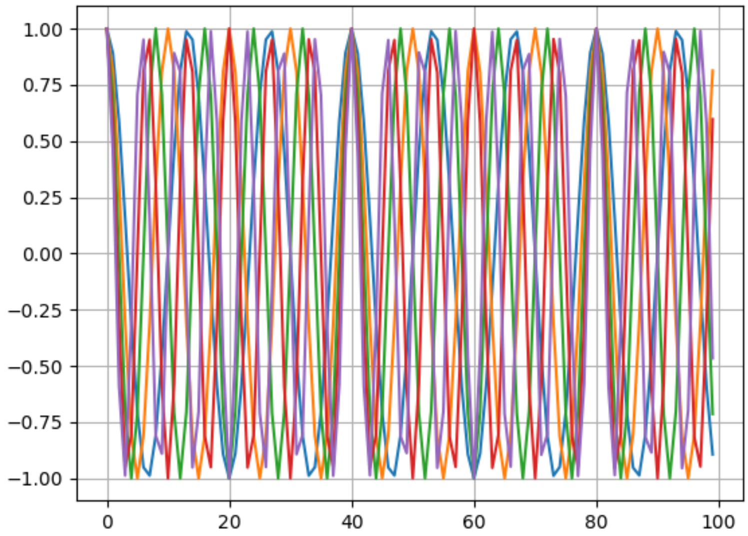

Рисунок 3.2. Косинусы различной частоты

Довольно часто необходимо изучать сигналы, которые на разных интервалах времени задаются разными формулами, таким образом, есть необходимость в рассмотрении построения кусочных зависимостей. Ниже представлены следущие импульсы(Рисунок. 3.3):

- Экспоненциальный
- Прямоугльный, центрированный относительно начала отсчета времени
- Несимметричный треугольный импульс

Зададим их следующим кодом:

Листинг 3: Plot3.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

t = np.linspace(-2, 2, 1000)
T = 0.5
alpha = 10
A = 2

S = [A * np.exp(-alpha * ti) if ti >= 0 else 0 for ti in t]
plt.figure(2)
plt.plot(t, S)

S = [A if np.abs(ti) <= T / 2 else 0 for ti in t]
plt.plot(t, S)

S = [A * ti / T if 0 <= ti <= T else 0 for ti in t]
plt.plot(t, S)

plt.show()
```

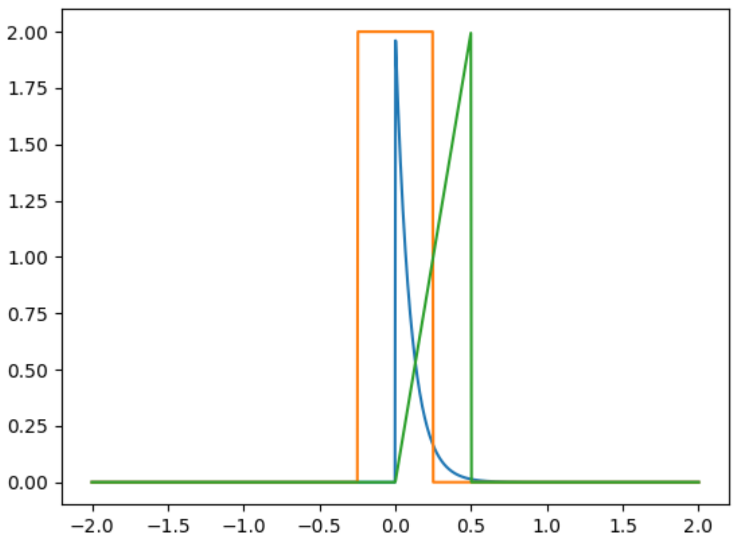

Рисунок 3.3. Различные виды импульсов

Рассмотрим различные одиночные импульсы. К сожалению, пакет signal языка Python не умеет строить одиночные импульсы, поэтому задавать такого вида импульсы будем вручную. Первый на очереди - прямоугольный импульс. С помощью сложения двух прямоугольных импульсов с разнополярными амплитудами получаем следующий Рисунок. 3.4

Листинг 4: Plot4.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal


Fs = 1e3
t = np.linspace(-40e-3, 40e-3, int(Fs))
T = 20e-3
A = 5

def Srect(t, width):
    return [int(-width / 2 <= ti < width / 2) for ti in t]

S = -A * np.asarray(Srect(t + T / 2, T)) + A * np.asarray(Srect(t - T / 2, T))
plt.figure()
plt.plot(t[0:len(S)], S)
plt.axis(ylim=[-6, 6])
plt.show()
```

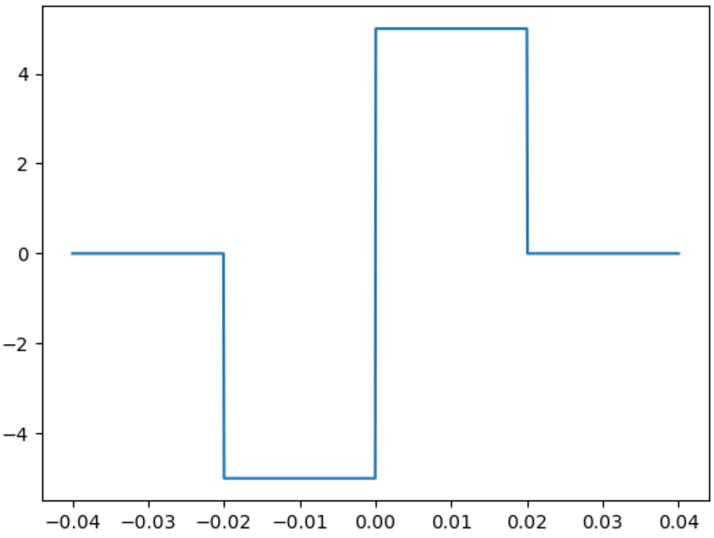

Рисунок 3.4. Пара разнополярных прямоугольных импульсов

Аналогичную процедуру проводим и для треугольного импульса, так же задавая функцию вручную. Далее используем нашу функцию для задания трапецевидного импульса(Рисунок. 3.5)

Листинг 5: Plot5.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def Striang(t, width, skew=0):
    S = []
    for ti in t:
        if -width / 2 <= ti < width * skew / 2:
            S.append((2 * ti + width) / (width * (skew + 1)))
        elif width * skew / 2 <= ti < width / 2:
            S.append((2 * ti - width) / (width * (skew - 1)))
        elif np.abs(ti) > width / 2:
            S.append(0)
    return np.asarray(S)


Fs = 1e3
T1 = 20e-3
t = np.linspace(-50e-3, 50e-3, int(Fs))
A = 10

T2 = 60e-3
s = A * (T2 * np.asarray(Striang(t, T2, 0)) - T1 * np.asarray(Striang(t, T1, 0))
    ) / (T2 - T1)
plt.figure()
plt.plot(t[0:len(s)], s)
plt.show()
```

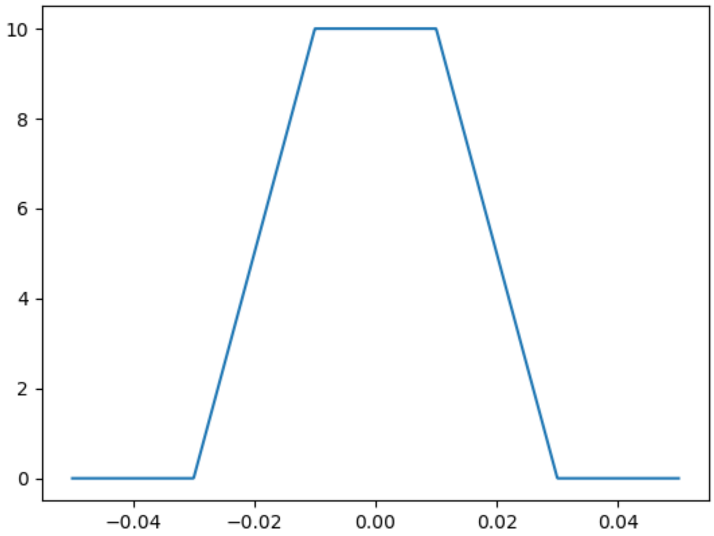

Рисунок 3.5. Трапецевидный импульс

Если же нам необходимо сформировать сигнал, кототрый имеет ограниченный по частоте спектр, можем использовать следующую функцию:

$$y = sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{1}$$

Формируем радиоимпульс путем домножения прямоугольного импульса на косинус(Рисунок. 3.6) и строим с помощью этой функции(1) амплитудный спектр(Рисунок. 3.7).

Листинг 6: Plot6.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def Srect(t, width):
    return [int(-width / 2 <= ti < width / 2) for ti in t]

Fs = 1e3
t = np.linspace(-0.1, 0.1, int(Fs))
f0 = 10
T = 1 / f0
s = np.asarray(Srect(t, T)) * np.cos(2 * np.pi * f0 * t)
f = np.linspace(-50, 50, 100)
sp = T / 2 * (np.sinc((f - f0) * T) + np.sinc((f + f0) * T))
plt.figure()
plt.plot(t[0:len(s)], s)
plt.figure()
plt.plot(f[0:len(sp)], abs(sp))
plt.show()
```

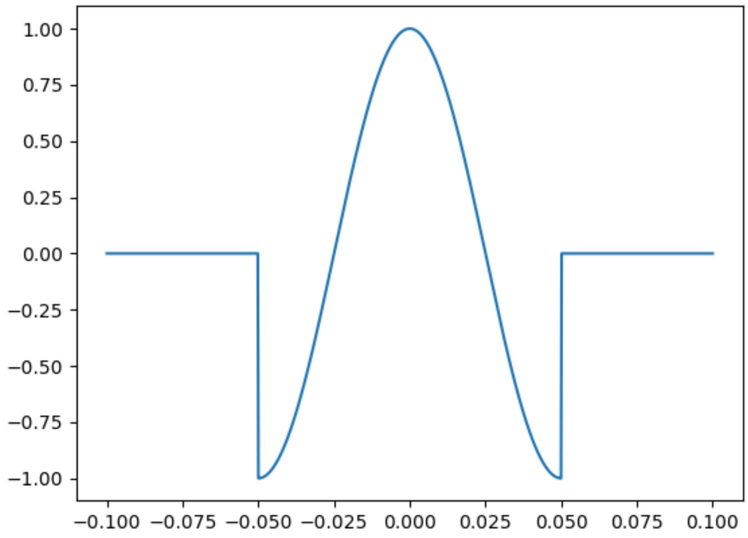

Рисунок 3.6. Одиночный радиоимпульс

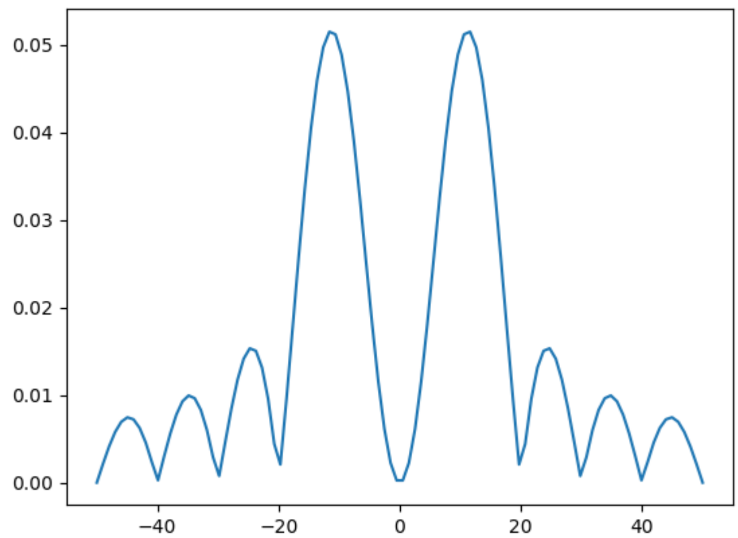

Рисунок 3.7. Амплитудный спектр

Видим, что спектр оказался несимметричным относильно частоты заполнения радиоимпульса.

Продолжим изучения различных импульсов построив одиночный радиоимпульс с гауссовой огибающей при помощи встроенной функции из пакета signal(Рисунок. 3.8). Так же построим его спектр((Рисунок. 3.9)).

Листинг 7: Plot7.py

```
from scipy.fftpack import fft
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy import signal

Fs = 16000
t = np.arange(-10e-3, 10e-3, 1 / Fs)
Fc = 4000
bw = 0.1
bwr = -20
s = signal.gausspulse(t, Fc, bw, bwr)
Nfft = int(2 ** np.ceil(np.log2(len(s))))
sp = fft(s, Nfft)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * Fs
sp_max_dp = 20 * np.log10(np.max(np.abs(sp)))
edges = Fc * np.asarray([1 - bw / 2, 1 + bw / 2])

plt.figure()
plt.grid()
plt.plot(t, s)

plt.figure()
plt.grid()
plt.plot(f[:int(Nfft / 2)], sp_dB[:int(Nfft / 2)])
plt.plot(edges, sp_max_dp * np.asarray([1, 1]) + bwr, "o")

plt.show()
```

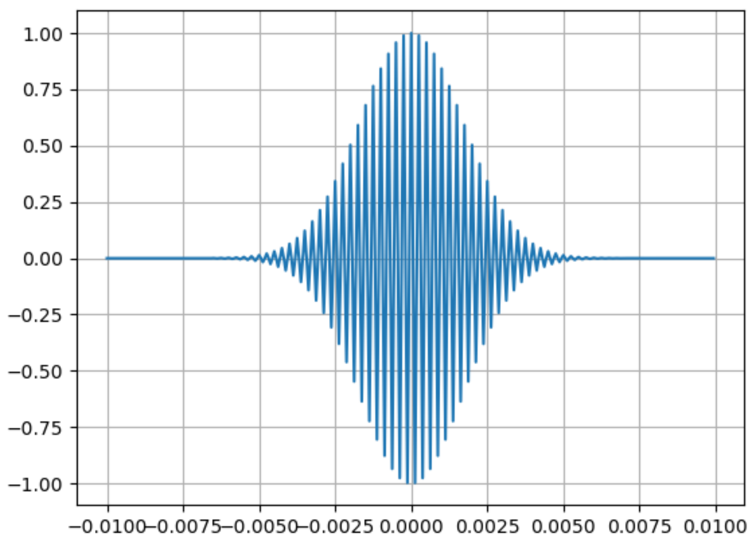

Рисунок 3.8. Гауссов радиоимпульс

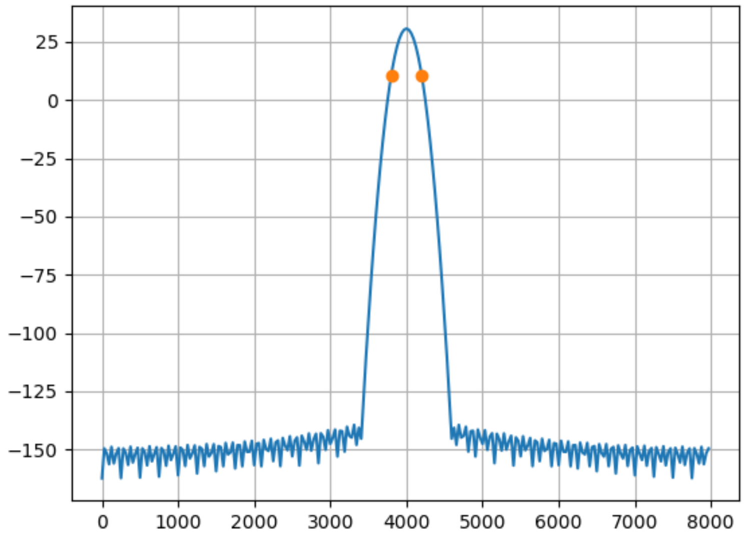

Рисунок 3.9. Амплитудный спектр

Спектр был построен при помощи быстрого преобразования Фурье из пакета signal. Так же был подсчитан максимальный уровень спектра в децибелах на граничных частотах. Границы спектра отображены на рисунке двумя точками.

После изучения одиночных импульсов целесообразно изучить последовательности импульсов. Для примера рассмотрим последовательность треугольных импульсов с различными амплитудами и задержками:

Листинг 8: Plot12.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def Striang(t, width, skew=0):
    S = []
    for ti in t:
        if -width / 2 <= ti < width * skew / 2:
            S.append((2 * ti + width) / (width * (skew + 1)))
        elif width * skew / 2 <= ti < width / 2:
            S.append((2 * ti - width) / (width * (skew - 1)))
        elif np.abs(ti) > width / 2:
            S.append(0)
    return np.asarray(S)

def pulstran(t, d, a, foo, *args, **kwargs):
    assert len(a) == len(d)
    acc = np.zeros(len(t))
    for di, ai in zip(d, a):
        acc += ai * foo(t - di, *args, **kwargs)
    return acc


Fs = 1e3
t = np.arange(0, 0.5, 1 / Fs)
tau = 20e-3
```

```
d = np.array([20, 80, 160, 260, 380]) * 1e-3
a = 0.8 ** np.arange(0, 5)
y = pulstran(t, d, a, Striang, tau)
plt.figure()
plt.grid()
plt.plot(t, y)
plt.show()
```

В отличие от MATLAB Python не имеет функции pulsetran. В связи с этим данную функцию опять же пришлось писать вручную. Результаты постороения отображены на Рисунок. 3.10

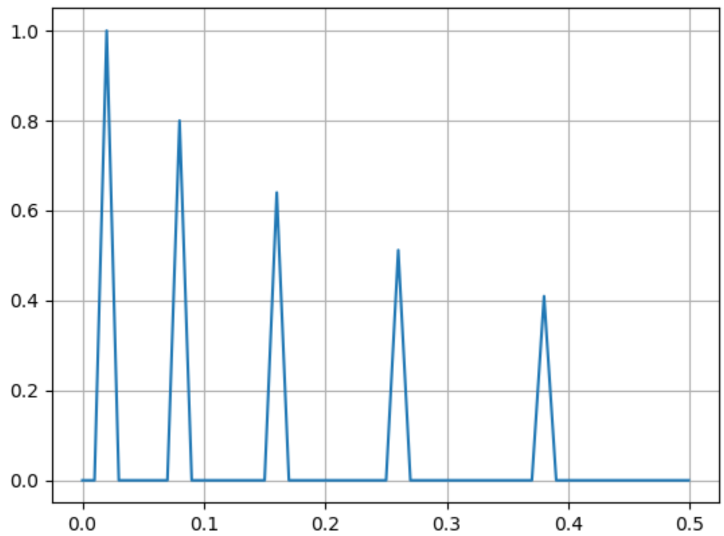

Рисунок 3.10. Последовательность треугольных импульсов

Рассмотрим последовательности, которые мы можем сгенерировать при помощи пакета signal:

- Прямоугольная(Рисунок. 3.11)
- Пилоoразная(Рисунок. 3.12)

Листинг 9: Plot8.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

Fs = 1e3
t = np.linspace(-10e-3, 50e-3, int(Fs))
A = 3
f0 = 50
tau = 5e-3
S = (signal.square(2 * np.pi * t * f0, f0 * tau) + 1) * A / 2
plt.figure()
plt.plot(t, S)
plt.show()
```

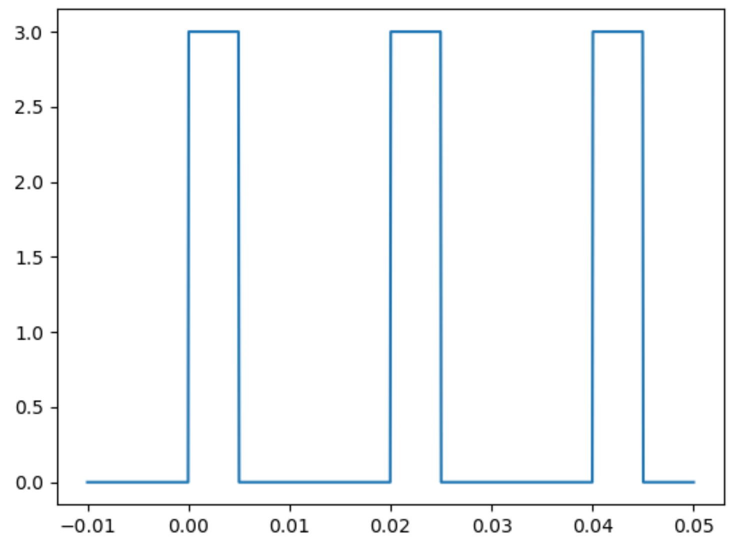

Рисунок 3.11. Последовательность прямоугольных импульсов

Листинг 10: Plot9.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal


Fs = 1e3
t = np.linspace(-25e-3, 125e-3, int(Fs))
A = 5
T = 50e-3
t1 = 5e-3
plt.figure()
S = (signal.sawtooth(2 * np.pi * t / T, 1 - t1 / T) - 1) * A / 2
plt.plot(t, S)
plt.show()
```

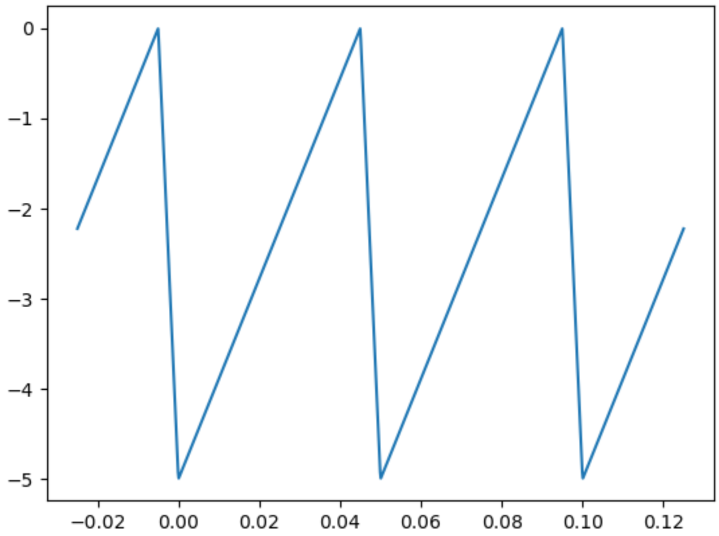

Рисунок 3.12. Последовательность пилообразных импульсов

Функцию Дирихле так же пришлось писать вручную, так как ее нет в библиотеке.

Листинг 11: Plot10.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

def diric(x, n): return np.sin(n * x / 2) / (n * np.sin(x / 2))


x = np.linspace(0, 15, 1 / 0.01)
plt.figure()
plt.plot(x, diric(x, 7))
plt.figure()
plt.plot(x, diric(x, 8))
plt.show()
```

Так как функция Дирихле зависит от двух параметров - x и n, рассмотрим графики при $n = 7$ *и* $n = 8$(Рисунок. 3.13 и Рисунок. 3.14).

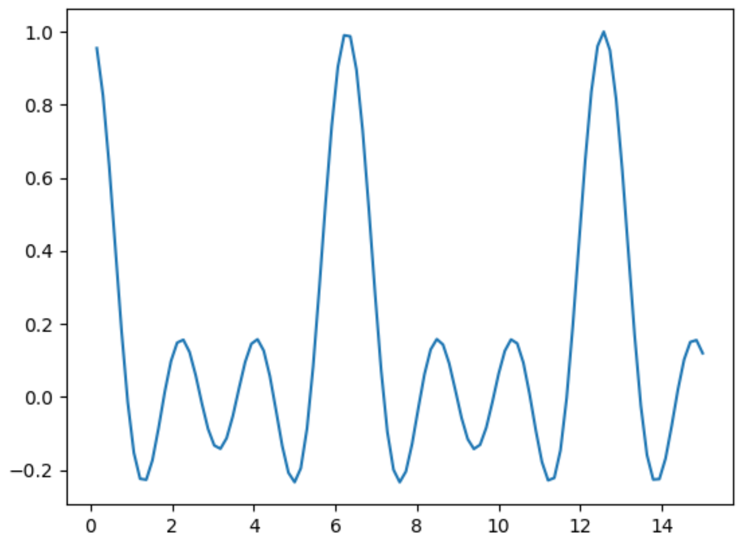

Рисунок 3.13. Функция Дирихле n = 7

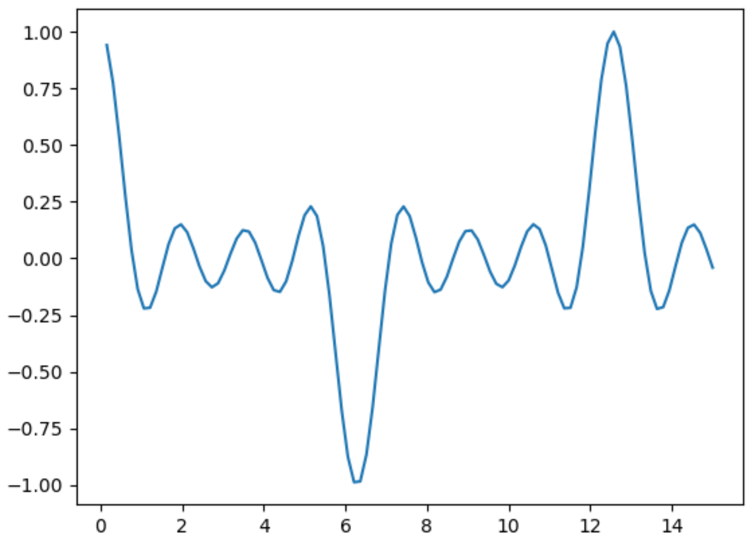

Рисунок 3.14. Функция Дирихле n = 8

Далее рассмотрим сигналы с менющейся мгновенной частотой. Мгновенную частоту можем менять по различным законам, подавая на вход функции chirp следующие параметры:

- 'linear' (Рисунок. 3.15)
- 'quadratic' (Рисунок. 3.16)
- 'logarithmic' (Рисунок. 3.17)

Листинг 12: Plot11.py

```
from scipy.fftpack import fft
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

Fs = 8e3
t = np.linspace(0, 1, int(Fs))
f0 = 1e3
t1 = 1
f1 = 2e3
s1 = signal.chirp(t, f0, t1, f1, 'linear')
s2 = signal.chirp(t, f0, t1, f1, 'quadratic')
s3 = signal.chirp(t, f0, t1, f1, 'logarithmic')
plt.figure()
plt.specgram(np.asarray(s1), None, int(Fs))
plt.figure()
plt.specgram(np.asarray(s2), None, int(Fs))
plt.figure()
plt.specgram(np.asarray(s3), None, int(Fs))
plt.show()
```

На графиках получаем спектрграммы - зависимость мгновенного амплитудного спектра сигнала от времени. Величина модуля спектральной функции отображается цветом в кориднатах "время - частота".

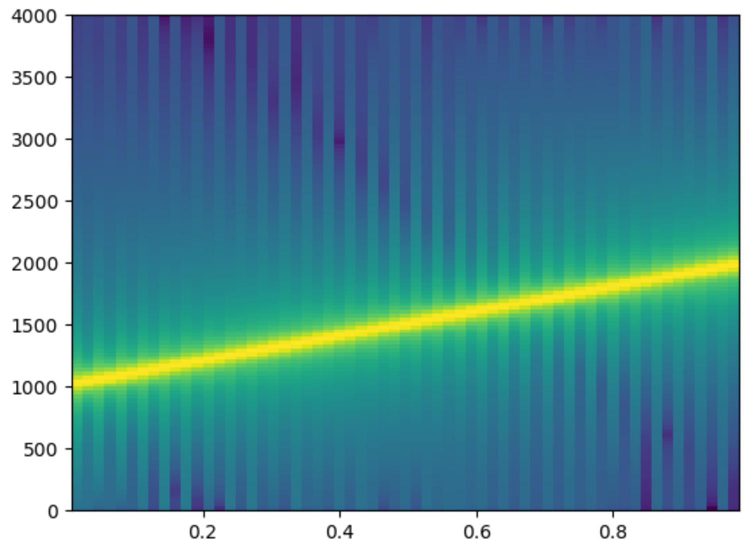

Рисунок 3.15. Спектрограмма сигнала при линейном законе изменения мгновенной частоты

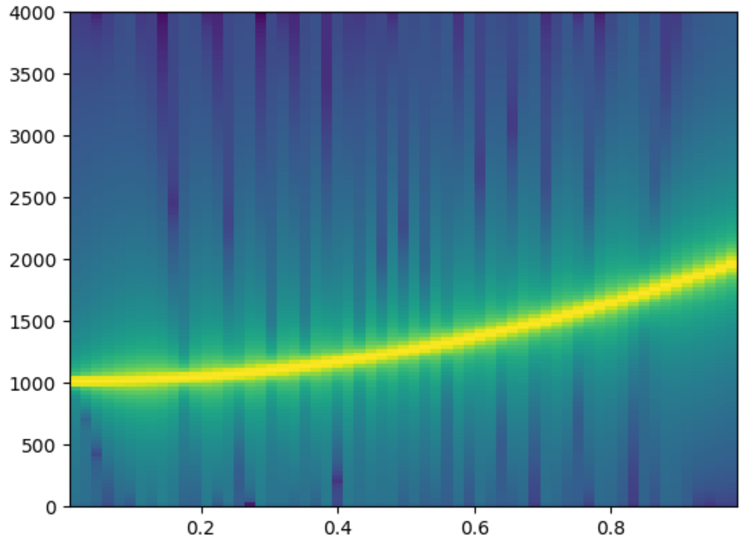

Рисунок 3.16. Спектрограмма сигнала при квадратичном законе изменения мгновенной частоты

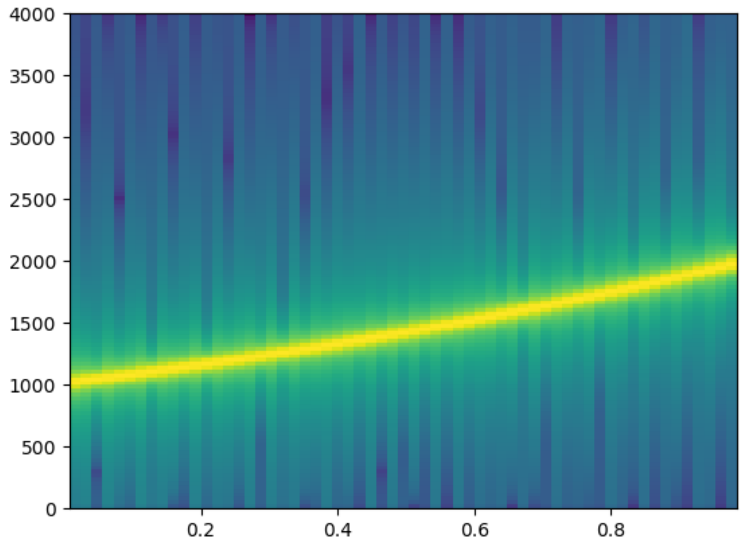

Рисунок 3.17. Спектрограмма сигнала при экспоненциальном законе изменения мгновенной частоты

4. Выводы

Проделав лабораторную работу, рассмотрели различные типовые сигналы часто использующиеся в ЦОС. Научились пользоваться различными библиотеками языка Python и самим языком Python. Сделали первые шаги в изучении преобразований Фурье и построении спектров сигналов. В лабораторной работе были затронуты дискретные сигналы, которые существенно отличаются от аналоговых. Были рассмотрены импульсы и последовательности импульсов.