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. Рассмотрим математические формулы некоторых сигналов, которые будут построены в данной работе.

Для формирования затухающей синусоиды задаим синус с соответствующей начальной фазой:

$$s1 = A * cos(2 * \pi * f0 * t + phi) \tag{1}$$

А затем домножим синусоиду(1) на экспоненту:

$$s2 = \exp(-alpha * t) * s1 \tag{2}$$

Для задания экспоненциального импульса воспользуемся следующим выражением:

$$s = A * \exp(-alpha * t), t >= 0 \tag{3}$$

Прямоугольный импульс зададим следующим образом:

$$s = A * |t|, |t| <= \frac{T}{2} \tag{4}$$

Несимметричный треугольный импульс можно рассчитать по следующей формуле:

$$s = A * \frac{t}{T}, 0 <= t <= T \tag{5}$$

Треугольный импульс, в общем случае, имеет следующий вид:

$$y = \begin{cases} \frac{2t+width}{width(skew+1)}, -\frac{width}{2} <= t < \frac{width*skew}{2} \\ \frac{2t-width}{width(skew-1)}, \frac{width*skew}{2} <= t < \frac{width}{2} \\ 0, |t| > \frac{width}{2} \end{cases}$$

Здесь, *width* - ширина импульса, а *skew* - коэффициент асимметрии импульса. Соответственно меняя данные параметры можем получать импульсы нужной нам длительности и с нужным положеним вершины.

В данной работе так же затрагивается гауссов радиоимпульс, вектор рассчитанных значений которого определяется по следующей формуле:

$$y = \exp(-at^2) * cos(2\pi f_c t) \tag{6}$$

Достаточно интересной является функция Дирихле, зависящая не только от параметра x, но и от n, которое является целым положительным числом. Данная функция описывается следующей фомрулой:

$$diric_n(x) = \frac{sin(nx/2)}{nsin(x/2)} \tag{7}$$

В данной работе будем рассматривать спектры простейших сигналов. Для построения спектра в осях амплитуда-частота, необходимо воспользоваться разложением в ряд Фурье. Ряд фурье может быть применен не только для периодических сигналов, но и для сигналов конечной длительности. При этом оговаривается промежуток, для которого этот ряд строится, во все остальное время сигнал считается равным нолю. Существует несколько форм записи ряда Фурье, однако целесообразнее записать его комплексную форму:

$$s(t) = \sum_{k=-\infty}^{\infty} C_k \exp(-jk\omega t) \tag{8}$$

Амплитуды A_k и фазы φ_k связаны с коэффициентами ряда C_k следующим образом:

$$C_k = \frac{1}{2} A_k exp(j\varphi_k) \tag{9}$$

$$A_k = 2|C_k| \tag{10}$$

3. Ход выполнения работы

Для начала научимся строить простейшие сигналы и изучим саму процедуру построения. Сформируем сигнал, описанный формулой (2), при помощи языка *Python*:

Листинг 1: Plot1.py

```
1  from scipy.fftpack import fft
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy import signal
5
6  Fdiscrete = 8e3
7  t = np.linspace(0, 1, int(Fdiscrete))
8  A = 2
9  f0 = 1e3
10 phi = np.pi / 4
11 s1 = A * np.cos(2 * np.pi * f0 * t + phi)
12 alpha = 1000
13 s2 = np.exp(-alpha * t) * s1
14
15 Nfft = int(2 ** np.ceil(np.log2(len(s2))))
16 sp = fft(s2, Nfft)
17 sp_dB = 20 * np.log10(np.abs(sp))
18 f = np.arange(0, Nfft - 1) / Nfft * Fdiscrete
19 plt.figure()
20 plt.grid()
21 plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
22
23
24 plt.figure(0)
25 plt.subplot(2, 2, 1)
26 plt.plot(s2[0:100])
27 plt.grid()
28 plt.subplot(2, 2, 2)
29 plt.stem(s2[0:100])
30 plt.grid()
31 plt.subplot(2, 2, 3)
32 plt.plot(s2[0:100], '.')
33 plt.grid()
```

```
34 plt.subplot(2, 2, 4)
35 plt.step(t[0:100], s2[0:100])
36 plt.grid()
37 plt.show()
```

Получаем следующие результаты(Рисунок. 3.1):

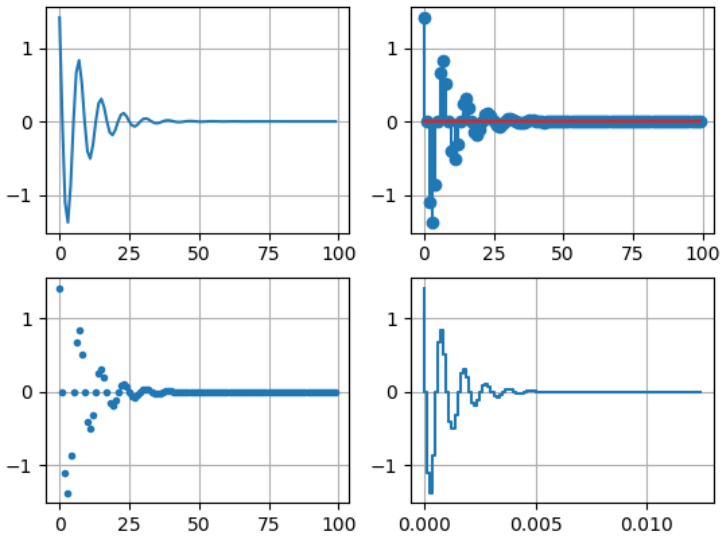

Рисунок 3.1. Гармонический сигнал представленный различными графическими функциями

Наблюдаем гаромонический сигнал, который затухает по экспоненте из-за домножения его на экспоненту. Построим спектр данного сигнала. На графике ожидаем увидеть ярко-выраженную амплитуду частоты заданной в коде программы. Результаты рассмотрим на Рисунке. 3.2

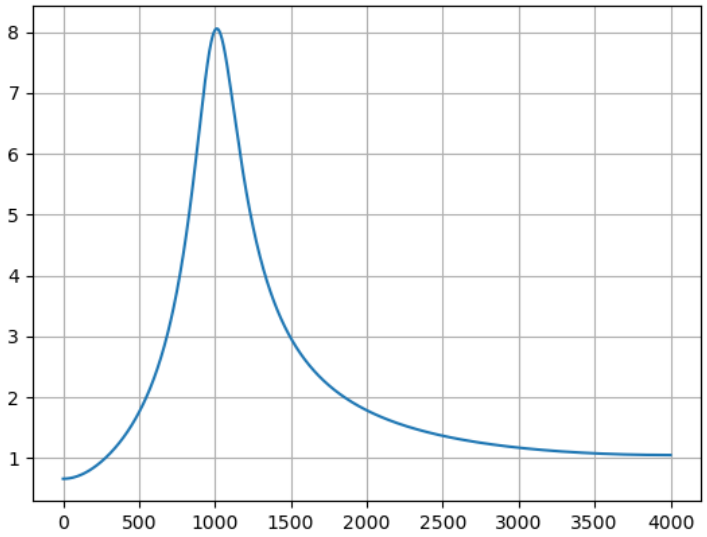

Рисунок 3.2. Спектр гармонического сигнала

Продолжая изучать графические возможности пакета, попробуем построить косинусы различной частоты(Рисунок. 3.3):

Листинг 2: Plot2.py

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


Fdiscrete = 32e3
t = np.linspace(0, 1, int(Fdiscrete))
f = np.asarray([600, 800, 1000])
s3 = [[np.cos(2 * np.pi * fi * ti) for ti in t] for fi in f]
plt.figure(1)
plt.grid()
for si in s3: plt.plot(si[0:100])
plt.figure()

for si in s3:
    Nfft = int(2 ** np.ceil(np.log2(len(si))))
    sp = fft(si, Nfft)
    sp_dB = 20 * np.log10(np.abs(sp))
    f = np.arange(0, Nfft - 1) / Nfft * Fdiscrete
    plt.grid()
    plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))

plt.axis([0, 4000, 0, 200])
plt.show()
```

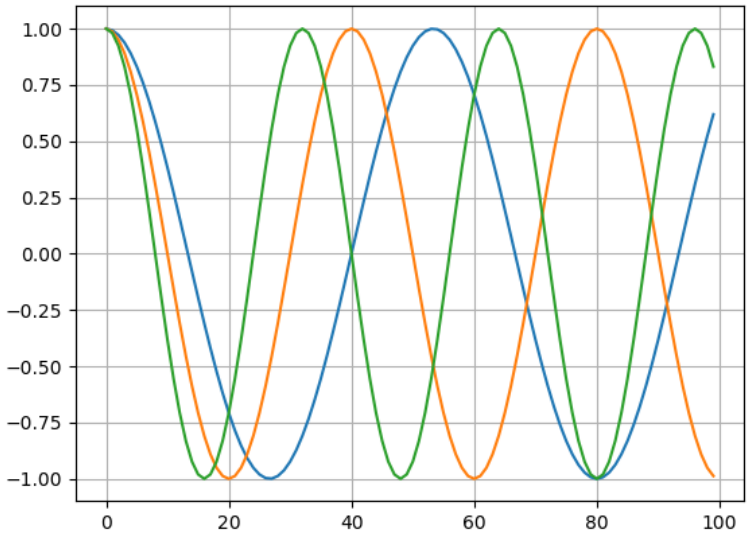

Рисунок 3.3. Косинусы различной частоты

Ограничимся тремя косинусами и рассмотрим их спектр.(Рисунок. 3.4)

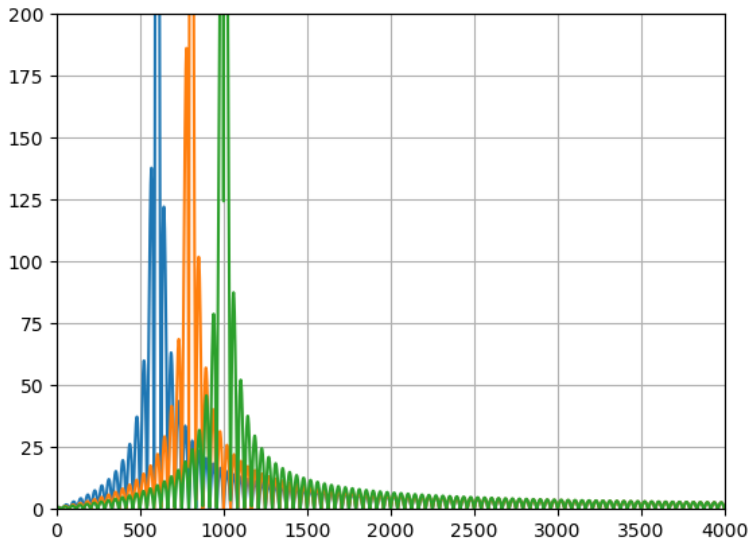

Рисунок 3.4. Спектры косинусы различной частоты

Как и ожидалось, наблюдаем наивысшые амплитуды сигналов в трех разных местах, соотвествущих частотам, заданным в коде программы.

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. 3.5), формулы которых представлены выше:

- Экспоненциальный

- Прямоугольный, 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, 10000)
T = 0.5
alpha = 10
A = 2

S1 = [A * np.exp(-alpha * ti) if ti >= 0 else 0 for ti in t]
plt.figure(2)
plt.plot(t, S1)

S2 = [A if np.abs(ti) <= T / 2 else 0 for ti in t]
plt.plot(t, S2)

S3 = [A * ti / T if 0 <= ti <= T else 0 for ti in t]
plt.plot(t, S3)


Nfft = int(2 ** np.ceil(np.log2(len(S1))))
sp = fft(S1)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * 10000
plt.figure()
plt.grid()
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
plt.axis([-10, 200, 0, 500])
#plt.plot(sp)

Nfft = int(2 ** np.ceil(np.log2(len(S2))))
sp = fft(S2)
print(sp)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * 10000
plt.figure()
plt.grid()
#plt.plot(sp)
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
plt.axis([-10, 200, 0, 2500])

Nfft = int(2 ** np.ceil(np.log2(len(S3))))
sp = fft(S3)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * 10000
plt.figure()
plt.grid()
#plt.plot(sp)
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
plt.axis([-10, 200, 0, 500])


plt.show()
```

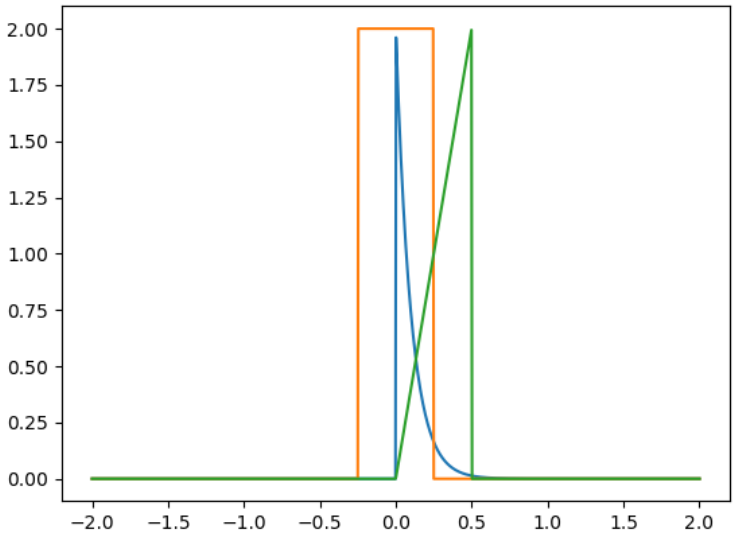

Рисунок 3.5. Различные виды импульсов

Их спектры, соответсвенно представлены ниже на Рисунках. 3.6,. 3.7,. 3.8.

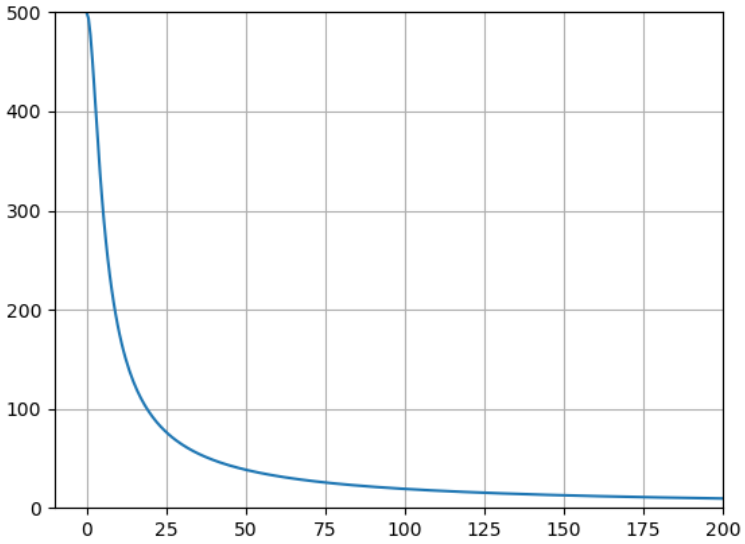

Рисунок 3.6. Спектр экспоненциального импульса

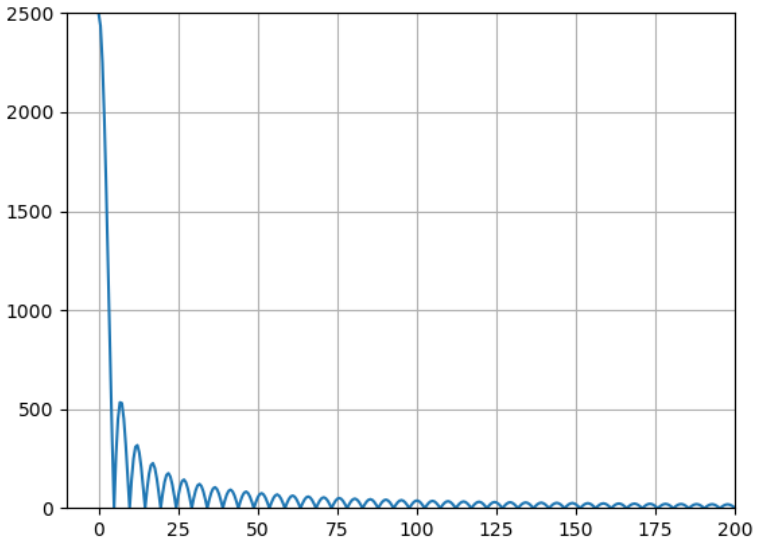

Рисунок 3.7. Спектр прямоугольного импульса

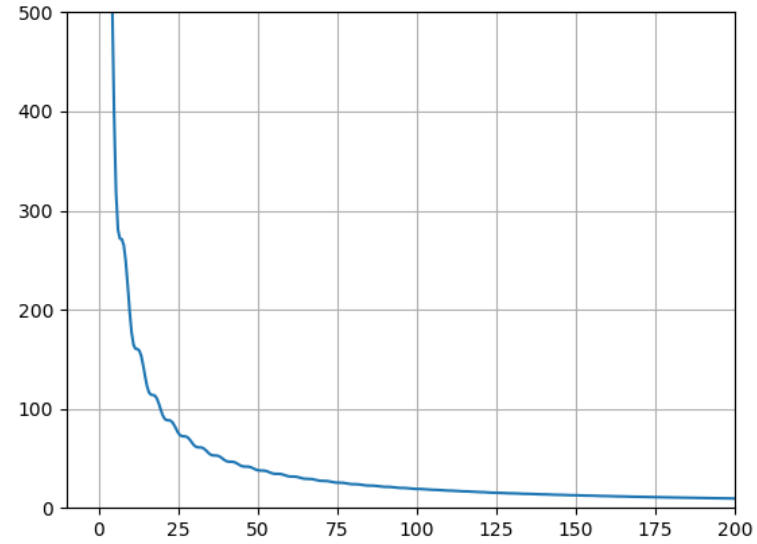

Рисунок 3.8. Спектр несимметричного треугольного импульса

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. 3.9

Листинг 4: Plot4.py

```
1 from scipy.fftpack import fft
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy import signal
```

```
5
6
7  Fs = 1e3
8  t = np.linspace(-40e-3, 40e-3, int(Fs))
9  T = 20e-3
10 A = 5
11
12 def Srect(t, width):
13     return [int(-width / 2 <= ti < width / 2) for ti in t]
14
15 S = -A * np.asarray(Srect(t + T / 2, T)) + A * np.asarray(Srect(t - T / 2, T))
16 plt.figure()
17 plt.plot(t[0:len(S)], S)
18 plt.axis(ylim=[-6, 6])
19
20 plt.figure()
21 Nfft = int(2 ** np.ceil(np.log2(len(S))))
22 sp = fft(S, Nfft)
23 sp_dB = 20 * np.log10(np.abs(sp))
24 f = np.arange(0, Nfft - 1) / Nfft * Fs
25 plt.grid()
26 plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
27 plt.axis([-10, 500, 0, 1500])
28
29 plt.show()
```

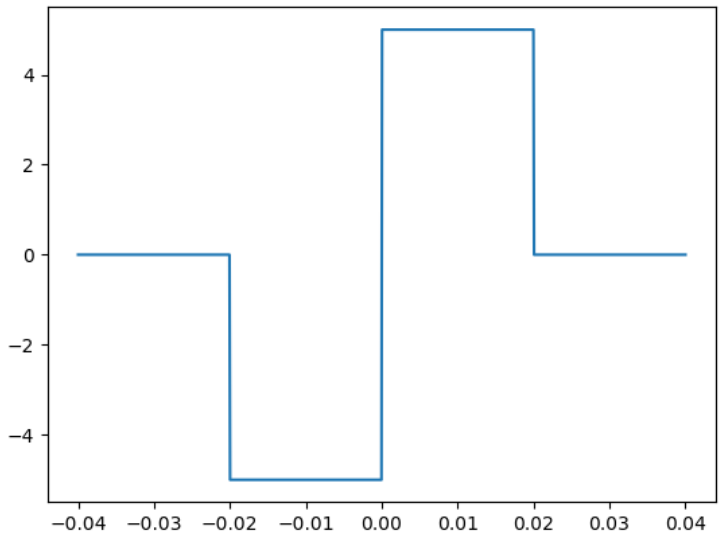

Рисунок 3.9. Пара разнополярных прямоугольных импульсов

Так же рассмотрим спектр данного импульса.(Рисунок. 3.10)

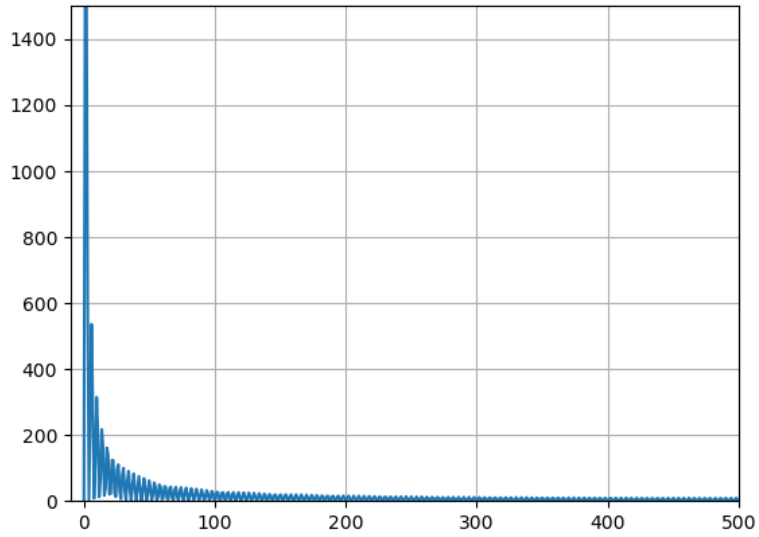

Рисунок 3.10. Спектр импульса, полученного путем сложения двух разнополярных прямоугольных импульсов

Спектр такого импульса получился очень похожим на спектр обыкновенного прямоугольного импульса.

Аналогичную процедуру проводим и для треугольного импульса, так же задавая функцию вручную по формуле представленной выше. Далее используем нашу функцию для задания трапецевидного импульса(Рисунок. 3.11)

Листинг 5: Plot5.py

```
1  from scipy.fftpack import fft
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy import signal
5
6  def Striang(t, width, skew=0):
7      S = []
8      for ti in t:
9          if -width / 2 <= ti < width * skew / 2:
10             S.append((2 * ti + width) / (width * (skew + 1)))
11         elif width * skew / 2 <= ti < width / 2:
12             S.append((2 * ti - width) / (width * (skew - 1)))
13         elif np.abs(ti) > width / 2:
14             S.append(0)
15     return np.asarray(S)
16
17
18 Fs = 1e3
19 T1 = 20e-3
20 t = np.linspace(-50e-3, 50e-3, int(Fs))
21 A = 10
22
23 T2 = 60e-3
24 s = A * (T2 * np.asarray(Striang(t, T2, 0)) - T1 * np.asarray(Striang(t, T1, 0))
       ↪ ) / (T2 - T1)
25 plt.figure()
26 plt.plot(t[0:len(s)], s)
```

```
27
28 plt.figure()
29 Nfft = int(2 ** np.ceil(np.log2(len(s))))
30 sp = fft(s, Nfft)
31 sp_dB = 20 * np.log10(np.abs(sp))
32 f = np.arange(0, Nfft - 1) / Nfft * Fs
33 plt.grid()
34 plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
35 plt.axis([-10, 80, 0, 1500])
36 plt.show()
```

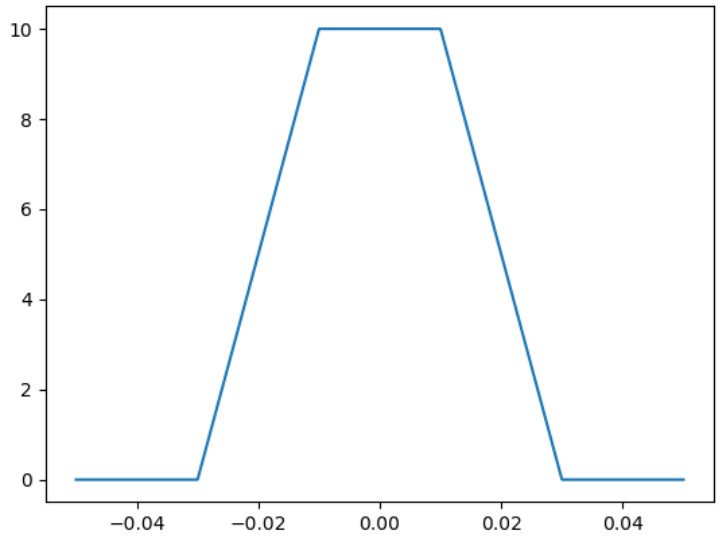

Рисунок 3.11. Трапецевидный импульс

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

$$y = sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{11}$$

Формируем радиоимпульс путем домножения прямоугольного импульса на косинус(Рисунок. 3.12) и строим с помощью этой функции(11) амплитудный спектр(Рисунок. 3.13).

Листинг 6: Plot6.py

```
1 from scipy.fftpack import fft
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy import signal
5
6 def Srect(t, width):
7     return [int(-width / 2 <= ti < width / 2) for ti in t]
8
9 Fs = 1e3
10 t = np.linspace(-0.1, 0.1, int(Fs))
11 f0 = 10
12 T = 1 / f0
13 s = np.asarray(Srect(t, T)) * np.cos(2 * np.pi * f0 * t)
```

```
14 f = np.linspace(-50, 50, 100)
15 sp = T / 2 * (np.sinc((f - f0) * T) + np.sinc((f + f0) * T))
16 plt.figure()
17 plt.plot(t[0:len(s)], s)
18 plt.figure()
19 plt.plot(f[0:len(sp)], abs(sp))
20 plt.show()
```

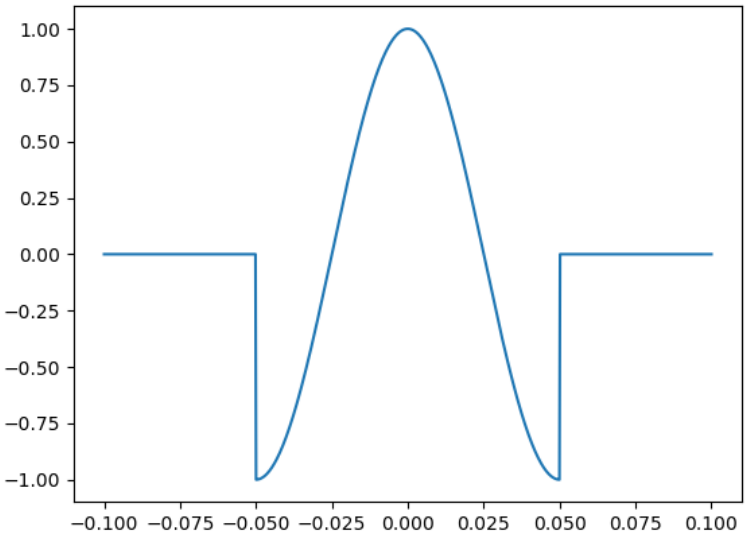

Рисунок 3.12. Одиночный радиоимпульс

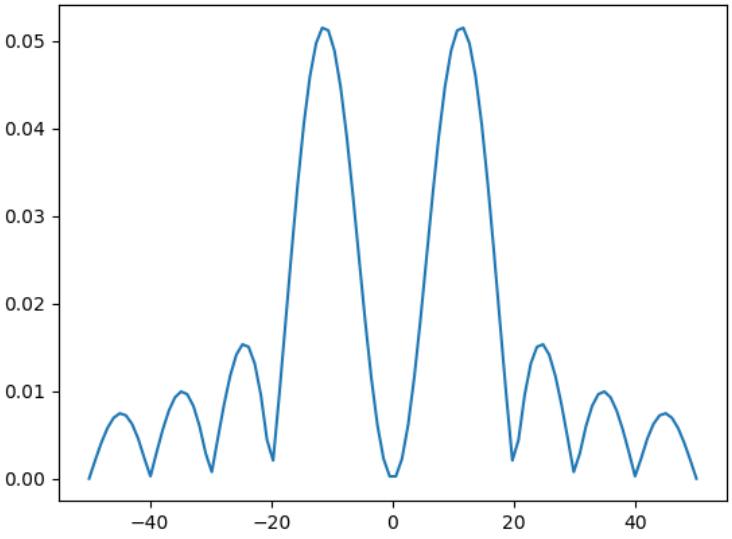

Рисунок 3.13. Амплитудный спектр

Видим, что спектр оказался несимметричным относительно частоты заполнения радиоимпульса.

Продолжим изучения различных импульсов построив одиночный радиоимпульс с гауссовой огибающей при помощи встроенной функции из пакета signal(Рисунок. 3.14). Так же построим его спектр((Рисунок. 3.15)).

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()
```

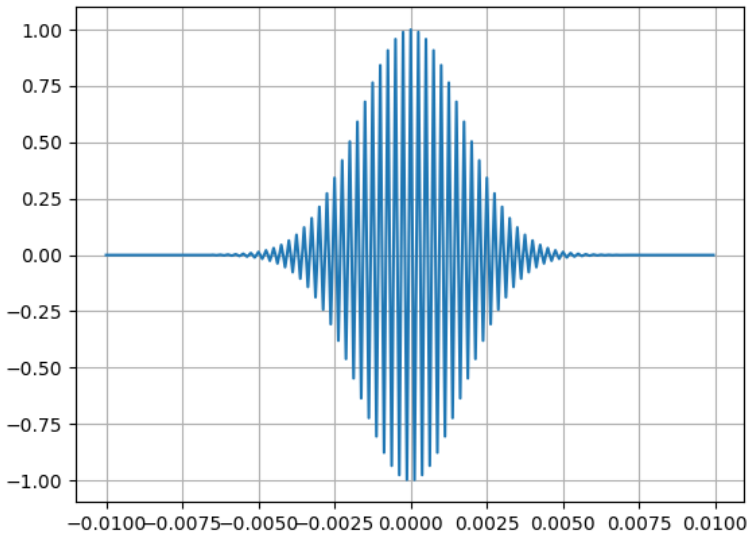

Рисунок 3.14. Гауссов радиоимпульс

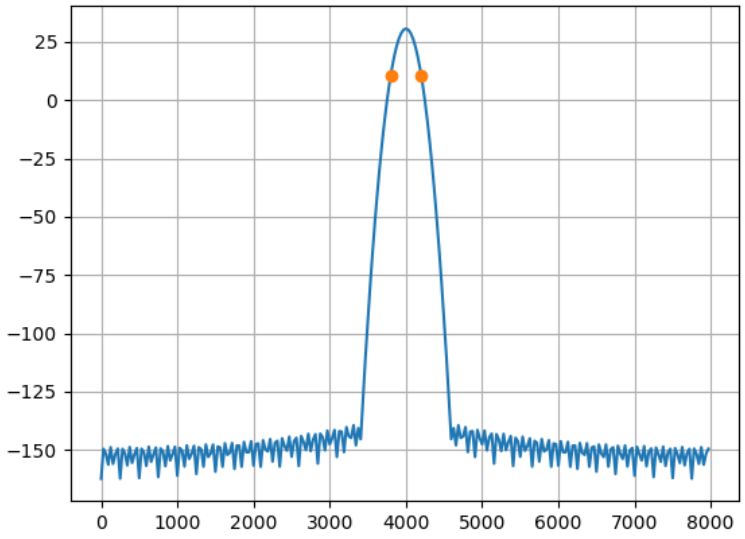

Рисунок 3.15. Амплитудный спектр

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)

Nfft = int(2 ** np.ceil(np.log2(len(y))))
sp = fft(y, Nfft)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * Fs
plt.figure()
plt.grid()
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))


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. 3.16

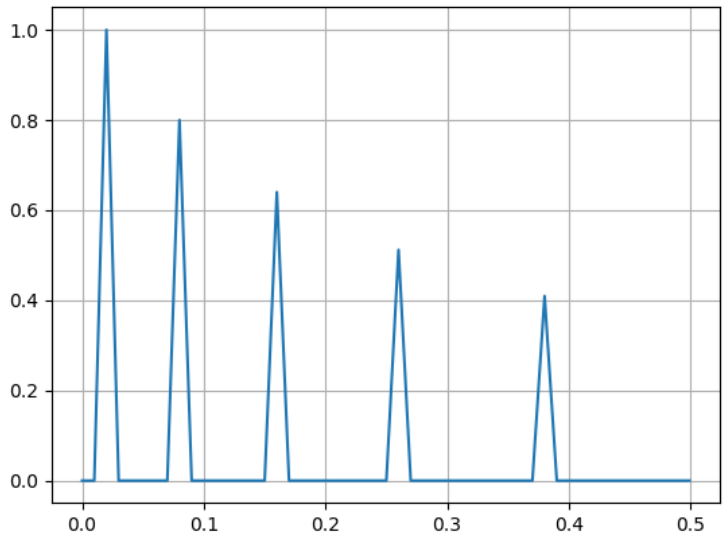

Рисунок 3.16. Последовательность треугольных импульсов

Интересно будет посмотреть спектр получившегося сигнала. Он представлен на Рисунке. 3.17

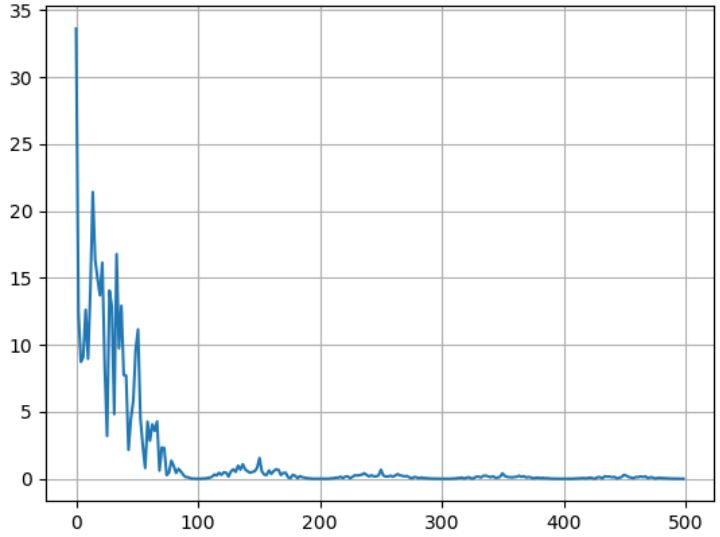

Рисунок 3.17. Спектр последовательности треугольных импульсов

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

- Прямоугольная(Рисунок. 3.18)
- Пилообразная(Рисунок. 3.19)

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)



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
plt.figure()
plt.grid()
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))

plt.show()
```

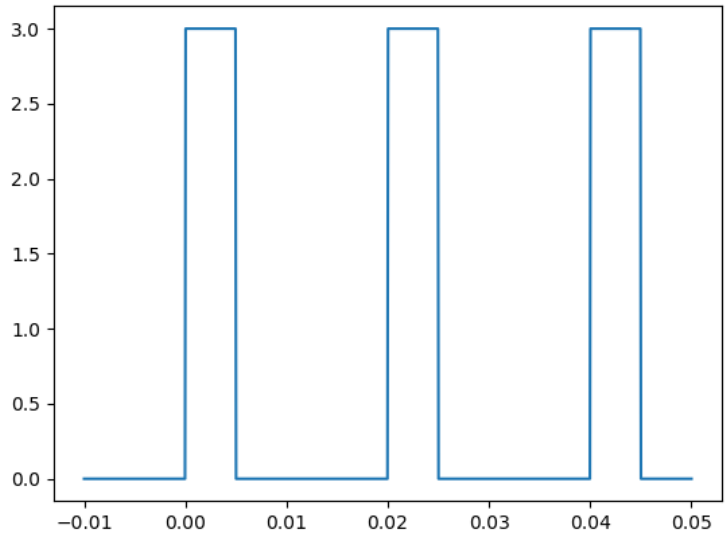

Рисунок 3.18. Последовательность прямоугольных импульсов

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)

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
plt.figure()
plt.grid()
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
plt.axis([-10, 40, 0, 3000])

plt.show()
```

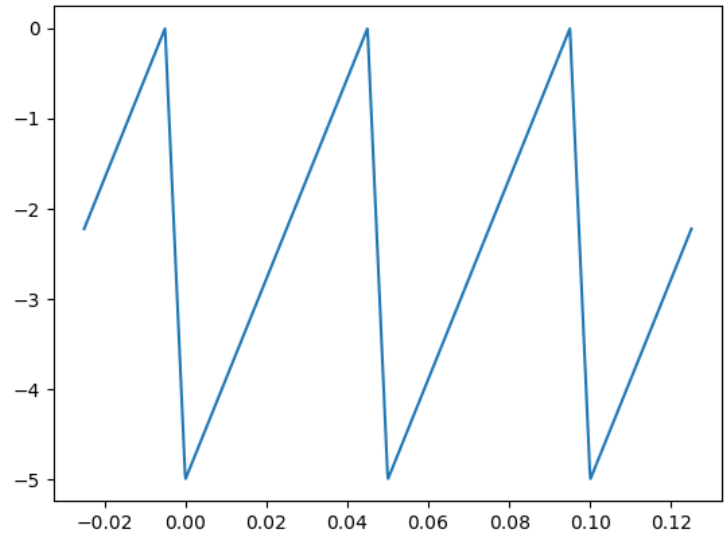

Рисунок 3.19. Последовательность пилообразных импульсов

Понаблюдаем за спектром этих сигналов на Рисунках. 3.20 и . 3.21

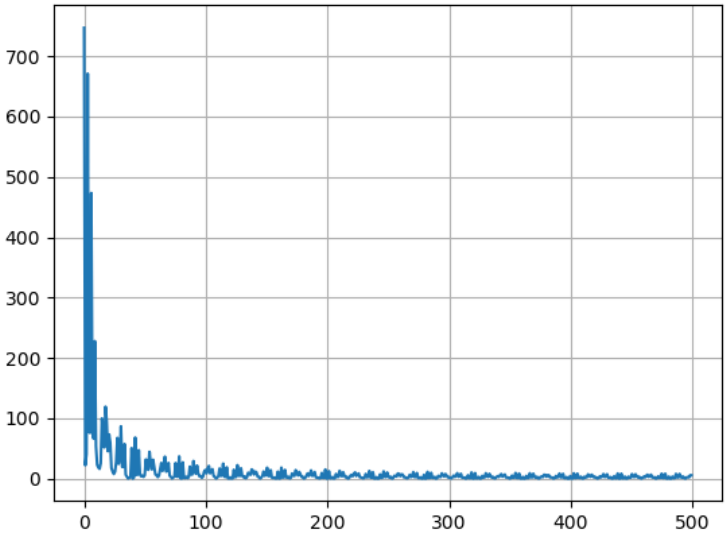

Рисунок 3.20. Спектр последовательности прямоугольных импульсов

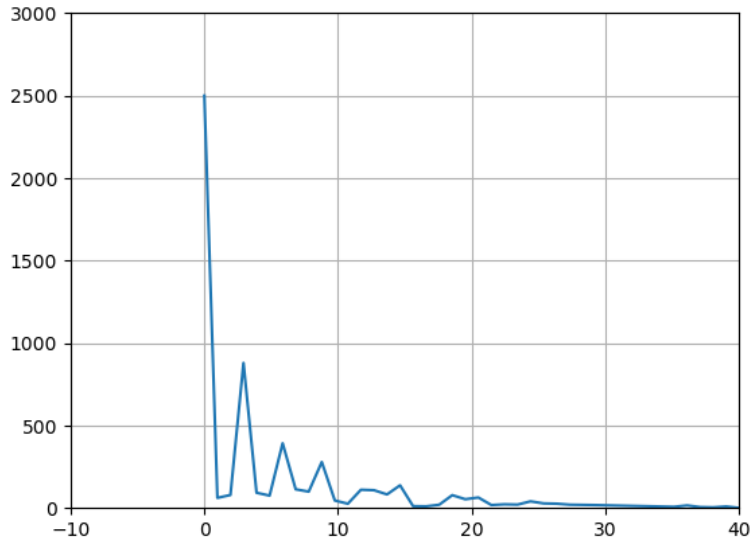

Рисунок 3.21. Спектр последовательности пилообразных импульсов

Функцию Дирихле так же пришлось писать вручную, так как ее нет в библиотеке. Для написания используем формулу(7).

Листинг 11: Plot10.py

```
1  from scipy.fftpack import fft
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy import signal
5
6  def diric(x, n):
7      S = []
8      for xi in x:
9          if xi == 0:
10             S.append(1)
11         else:
12             S.append(np.sin(n * xi / 2) / (n * np.sin(xi / 2)))
13     return np.asarray(S)
14
15
16 x = np.linspace(0, 15, 1 / 0.01)
17 Fs = 1 / 0.01;
18 plt.figure()
19 plt.plot(x, diric(x, 7))
20 plt.figure()
21 Nfft = int(2 ** np.ceil(np.log2(len(diric(x, 7)))))
22 print(Nfft)
23 sp = fft(diric(x, 7), Nfft)
24 sp_dB = 20 * np.log10(np.abs(sp))
25 f = np.arange(0, Nfft - 1) / Nfft * Fs
26 plt.grid()
27 plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
28 plt.figure()
29 plt.plot(x, diric(x, 8))
30 plt.figure()
31 Nfft = int(2 ** np.ceil(np.log2(len(diric(x, 8)))))
32 sp = fft(diric(x, 8), Nfft)
```

```
print(sp)
print(Nfft)
sp_dB = 20 * np.log10(np.abs(sp))
f = np.arange(0, Nfft - 1) / Nfft * Fs
plt.grid()
plt.plot(f[:int(Nfft / 2)], np.abs(sp[:int(Nfft / 2)]))
plt.show()
```

Так как функция Дирихле зависит от двух параметров - x и n, рассмотрим графики при $n = 7$ и $n = 8$(Рисунок. 3.22 и Рисунок. 3.23).

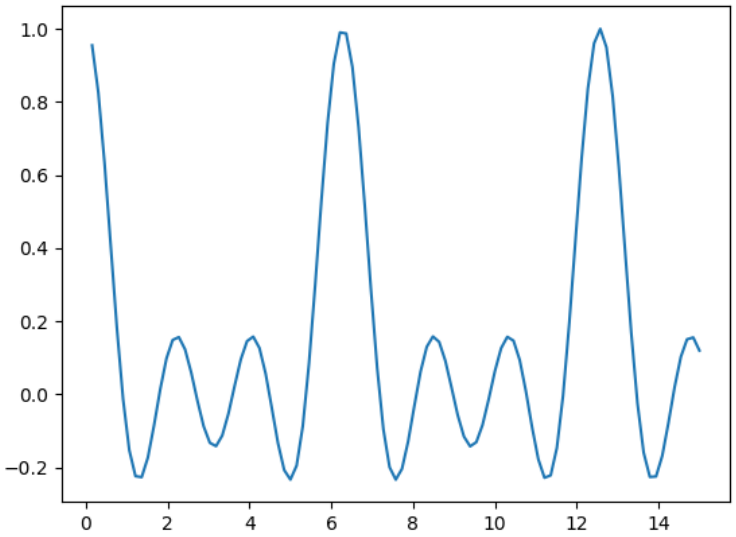

Рисунок 3.22. Функция Дирихле n = 7

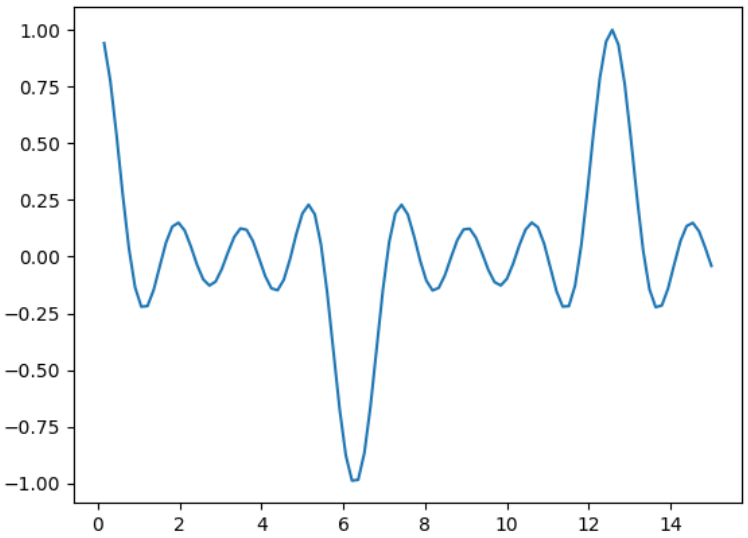

Рисунок 3.23. Функция Дирихле n = 8

Соответственно спектры для разных параметров n представлены на Рисунках. 3.24 и 3.25.

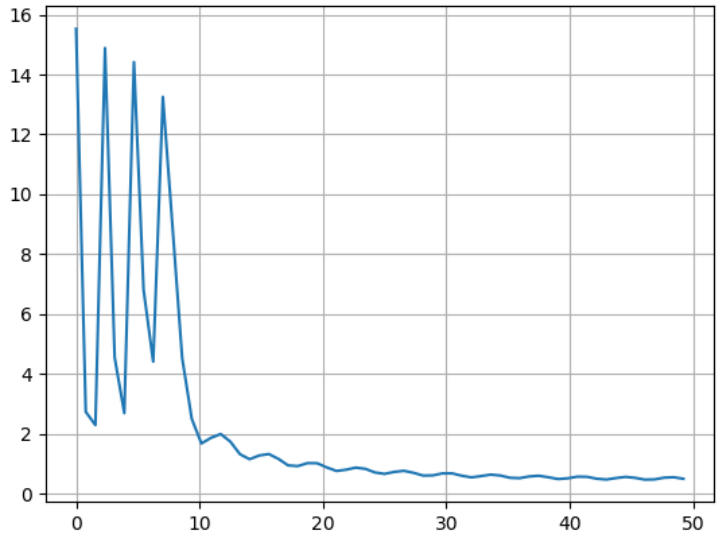

Рисунок 3.24. Спектр функции Дирихле n = 7

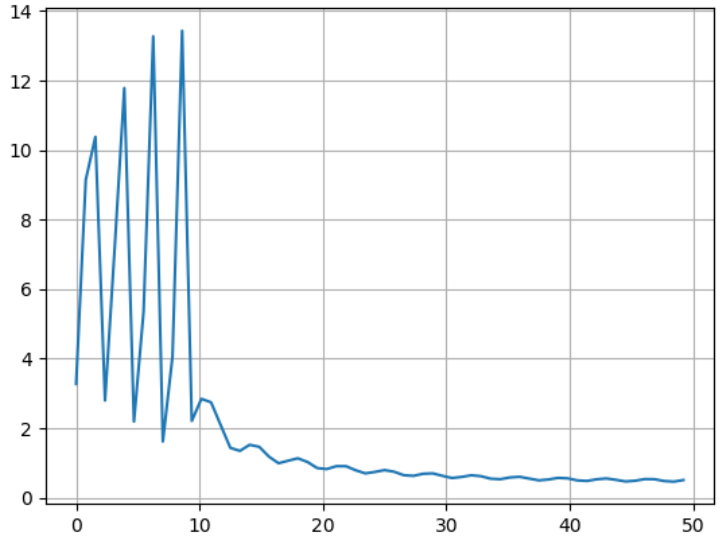

Рисунок 3.25. Спектр функции Дирихле n = 8

Стоит заметить, что оба спектра имеет общую природу.

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

- 'linear' (Рисунок. 3.26)

- 'quadratic' (Рисунок. 3.27)
- 'logarithmic' (Рисунок. 3.28)

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 кооридинатах "время - частота".

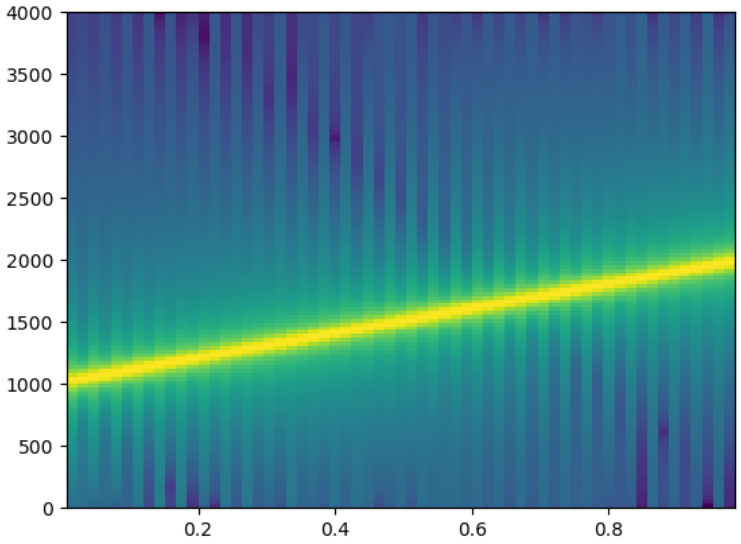

Рисунок 3.26. Спектрограмма сигнала при линейном законе изменения мгновенной частоты

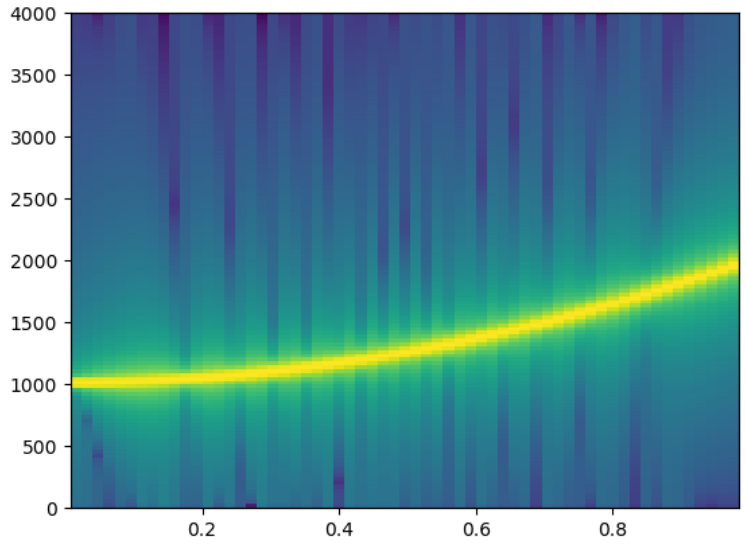

Рисунок 3.27. Спектрограмма сигнала при квадратичном законе изменения мгновенной частоты

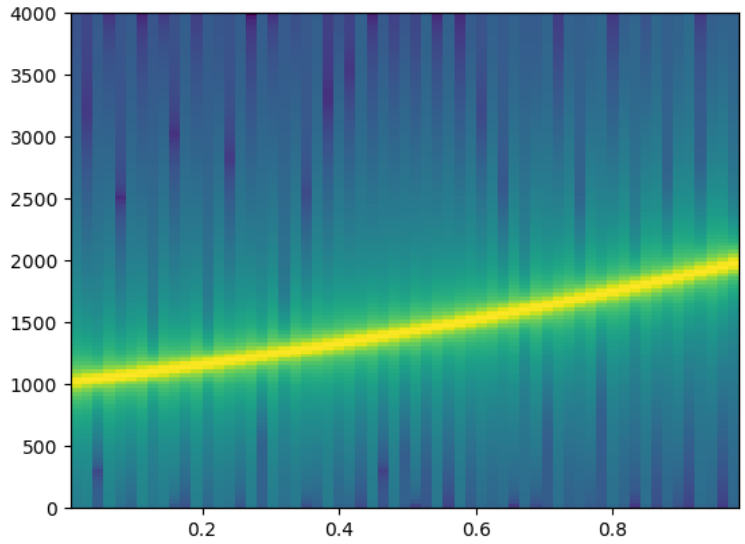

Рисунок 3.28. Спектрограмма сигнала при экспоненциальном законе изменения мгновенной частоты

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 постороении спектров сигналов. Были постороены спектры как одиночных импульсов, так и

последовательности импульсов. Построения спектров подтвердили наши ожидания о примерной форме спектра. В лабораторной работе были затронуты сигналы, которые определны только на определенных промежутках времени. Этим данные сигналы существенно отличаются от аналоговых. Стоит отметить, что в силу своей конечности, данные сигналы имеют бесконечный спектр. Говоря же о преобразовании Фурье, стоит сказать, что с его помощью мы можем переходить из одной системы координат в другую, в которой многие операции над сигналами могут быть выполнены наименее затратно и проблемно. Такой переход необходим для амплитудно-частотного анализа. Исходя из этого анализа, могут быть использованы различные фильтры для обработки сигнала или удаления шума, сигнал можеть быть сжат.