МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра АПУ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Теория автоматического управления»
Тема: Типовые нелинейности

Студент гр. 7392 _____ И.В. Батура

Студентка гр. 7392 _____ А.Г. Цуркан

Студент гр. 7392 _____ Е.О. Амельченко

Преподаватель _____ А.М. Синица

Санкт-Петербург
2020

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Знакомство с типовыми нелинейными звеньями.

2 ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Основанная на Python библиотка SciPy является набором программных средств для математических, научных и инженерных расчетов(как аналитических, так и численных), включающих в себя такие пакеты как NumPy(работа с N-мерными матрицами), SciPy library (базовая библиотека для научных вычислений), Matplotlib (библиотека для построения графиков)

2.1 Пробные сигналы

В качестве пробных сигналов, как правило, используются гармонические. Основным пробным сигналом в вопросах идентификации является синусоида, однако бывают исключения, связанные с особенностью объекта управления. В лабораторной работе рассмотрены 3 типа пробных сигналов: синусоида, пилообразный сигнал, меандр.

2.2 Типовые нелинейные звенья

2.2.1 Идеальное реле(знак)

Нелинейность "реле"позволяет учесть в моделях одноименных электромеханические механизмы или их электронные аналоги (транзисторы и др.), кроме того такая нелинейность необходима для описания физических процессов не зависящих от амплитуды, но зависящих от знака воздействия (например, сухое трение).

2.2.2 Мертвая зона

Мертвой зоной принято называть не единственность положения равновесия (y = const, u = const => dy/dt = 0/ du/dt = 0). Примером объекта с такой нелинейностью может служить маятник. Нелинейность подобного рода приводит к конечности временных процессов.

Нелинейность "мертвая зона"позволяет учесть малые значения отклонений.

2.2.3 Усилитель с насыщением

Ограниченность уровня переменных связана в первую очередь с различными физическими ограничениями, препятствующими бесконечному

росту переменных состояния, например вязкое трение (принебрежимо малое при малых скоростях и имеющее квадратичные рост на больших) или различные процессы, связанные с насыщением.

Принцип работы такой нелинейности заключается в пропуске сигнала только до уровня предельной амплитуды.

Нелинейность "насыщение"позволяет учесть большие значения переменных.

3 ЗАДАЧА НА ЛАБОРАТОРНУЮ РАБОТУ

Изучение нелинейностей типа реле (знак), насыщения, мертвая зона; построение статической характеристики; спектров исходного, преобразованного и фильтрованного сигналов, знакомство с экосистемой математических вычислений и моделирования SciPy (Python).

Выполнить следующие действия над пробными сигналами (синусоида, меандр, пила) и всеми нелинейностями (реле, мертвая зона, насыщение).

1. Сгенерировать пробный сигнал длительностью 100 секунд, построить его спектр
2. Получить отклик типовых нелинейных звеньев на пробные сигналы, построить их спектры
3. Построить статическую характеристику нелинейного звена, объяснить разницу при разных пробных сигналах
4. Получить отклик линейного звена на преобразованный сигнал, построить его спектр
5. Изменить последовательность НЭ-ЛЗ на ЛЗ-НЭ. получить результирующий сигнал, построить его спектр

4 ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

4.1 Пробные сигналы

4.1.1 Синусоида

В данном пункте мы рассмотрим тестовый сигнал вида синусоида. По варианту задания мы должны были сформировать сигнал с амплитудой высчитываемой по формуле: 1 + вариант * 0.1; и частотой рассчитываемой следующим образом: 1 + вариант * 3.5. Так им образом учтя наш вариант получаем следующие значения: амплитудой равной 1.1 и частотой

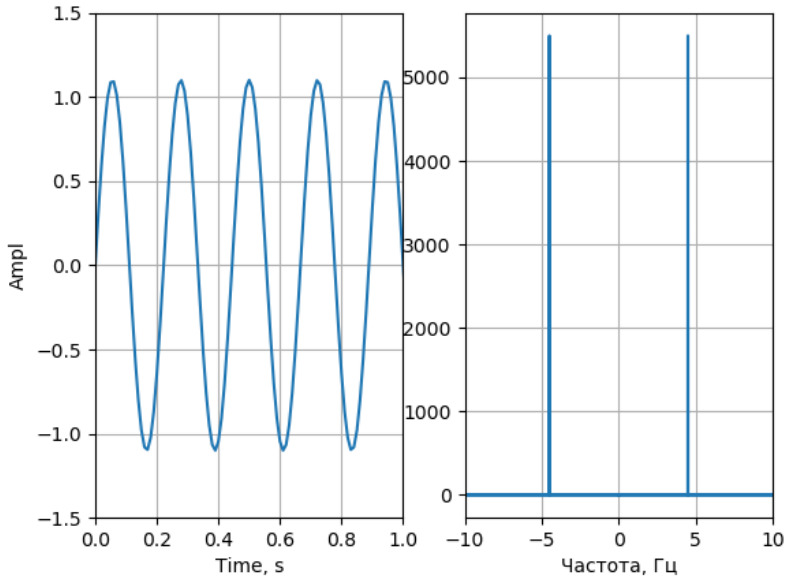

Рисунок 1 — Пробный сигнал синуса и его спектр

равной 4.5. Рассмотрим формирования сигнала на языке программирования Python.

```
t = np.arange(0, test_signal_duration, dt)

# Test signal sin
sig_sin = test_sig_ampl * np.sin(t * 2 * np.pi *
test_sig_freq)
```

Таким образом мы сформировали синусоидальный сигнал с нужными нам частотой и амплитудой. Далее построим спектр нашего сигнала на языке Python. Делается это следующим образом.

```
# Spectre test sin
sig_sin_spec = np.abs(np.fft.fft(sig_sin))
sin_freqs = np.fft.fftfreq(sig_sin.shape[0], dt)
```

Теперь убедимся что мы сделали все верно, построим графики на языке программирования Python(см. рисунок 1).

```
plt.subplot(1, 2, 1)
plt.grid()
plt.xlabel('Time, s')
plt.ylabel('Ampl')
plt.ylim(-1.5, 1.5)
plt.xlim(0, 1)
plt.plot(t, sig_sin)

plt.subplot(1, 2, 2)
```

```
plt.grid()
plt.xlabel('Freqs, Hz')
plt.xlim(-10, 10)
plt.plot(sin_freqs, sig_sin_spec)

plt.show()
```

Как мы можем заметить все отобразилось верно.

4.1.2 Меандр

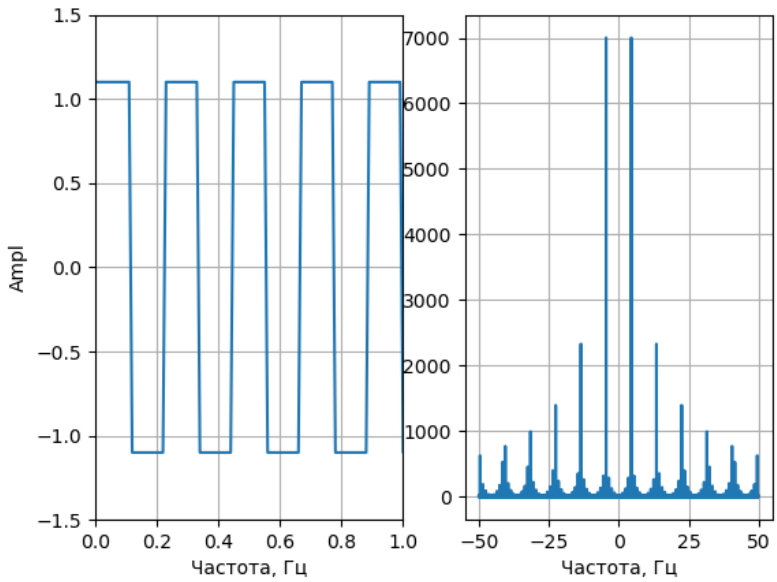

Рисунок 2 — Пробный сигнал меандра и его спектр

В этой секции мы построим меандр с помощью инструментом Python. Нам нужно построить меандр с амплитудой равной 1.1 и частотой равной 4,5 Гц. Для построения меандра мы воспользуемся библиотекой Python, которая подключается следующим образом:

```
from scipy import signal
```

Подключив данную библиотеку и вызвав из нее функцию signal.square(t, duty = 0.5), мы можем построить заданный нам условиями лабораторной работы сигнал, а также сразу построим спектр нашего сигнала. Что мы и сделаем далее.

```
# Test signal square
sig_square = test_sig_ampl * signal.square(t * 2 * np.pi * test

# Spectre test square
sig_square_spec = np.abs(np.fft.fft(sig_square))
square_freqs = np.fft.fftfreq(sig_square.shape[0], dt)
```

Чтобы убедиться что мы все сделали правильно построим графики спектра и самого сигнала.

```
plt.subplot(1, 2, 1)
plt.grid()
plt.ylabel('Ampl')
plt.ylim(-1.5, 1.5)
plt.xlabel('Freq, Hz')
plt.xlim(0, 1)
plt.plot(t, sig_square)

plt.subplot(1, 2, 2)
plt.grid()
plt.xlabel('Freq, Hz')
plt.plot(square_freqs, sig_square_spec)

plt.show()
```

Как мы можем убедиться из графика(см.рисунок 2), мы все сделали верно.

4.1.3 Пилообразный сигнал

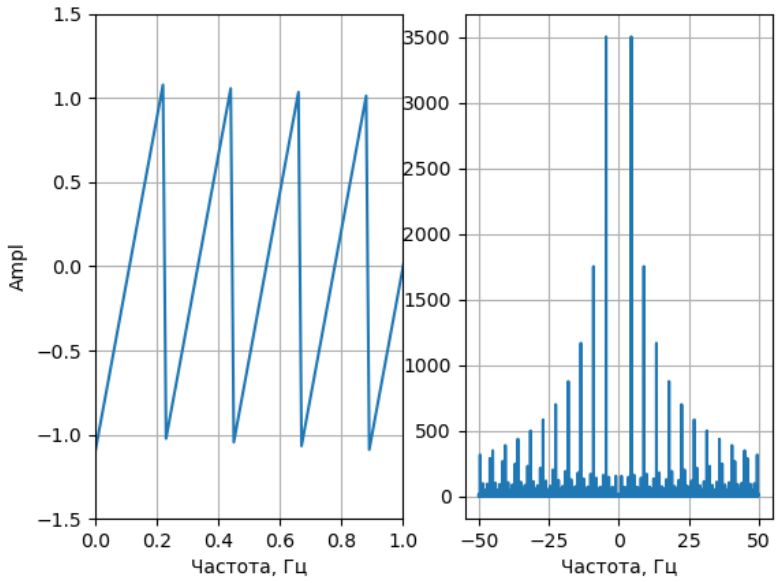

Рисунок 3 — Пробный пилообразный сигнал и его спектр

В этом разделе мы рассмотрим построение пилообразного сигнала средствами Python. Для упрощенного построения воспользуемся библиотекой signal подключенной в прошлом разделе. Вызвав функцию signal.sawtooth(), мы построим нужный нам сигнал. Так же построим спектр нашего сигнала.

```
# Test signal sawtooth
sig_saw = test_sig_ampl * signal.sawtooth(t * 2 * np.pi * test_

# Spectre test sawtooth
sig_saw_spec = np.abs(np.fft.fft(sig_saw))
saw_freqs = np.fft.fftfreq(sig_saw.shape[0], dt)
```

Для того что бы убедиться в правильности построения нашего сигнала, построим графики, аналогично вызовам которые мы делали в секциях построения синуса и меандра. Сделав это мы увидим следующий графики(см.Рисунок 3).

4.2 Реле

В данной секции мы рассмотри воздействие нелинейного элемента(реле) на три вида пробных сигнала.

4.2.1 Синусоида

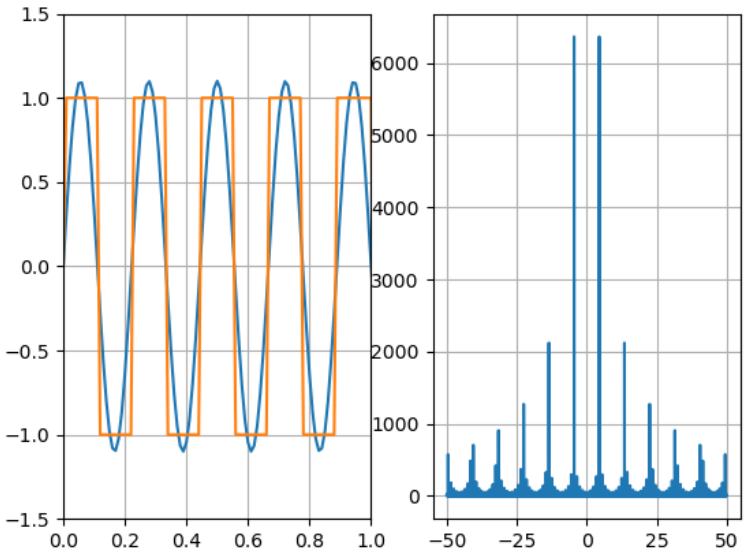

Рисунок 4 — Синус через реле и спектр выходного сигнала

В этой части отчета мы рассмотрим прохождения синуса через нелинейное звено - реле. Для реализации линейного звена в Python можно использовать функцию знака которая лежит в библиотеки numpy, вызвать ее можно следующим образом np.sign(). Применим данную функцию на наш сигнал тем самым имитируя прохождения синуса через реле.

```
# Sin after relay
sig_sin_relay = np.sign(sig_sin)
```

```
# Sin, relay and spectre
sig_sin_relay_spectre = np.abs(np.fft.fft(sig_sin_relay))
sig_sin_relay_freqs = np.fft.fftfreq(sig_sin_relay.shape[0], d
```

Построим график сигнала и сигнала на выходе, чтобы были более наглядны изменения сигнала.

```
plt.subplot(1, 2, 1)
plt.grid()
plt.ylim(-1.5, 1.5)
plt.xlim(0, 1)
plt.plot(t, sig_sin, t, sig_sin_relay)
plt.subplot(1, 2, 2)
plt.grid()
plt.plot(sig_sin_relay_freqs, sig_sin_relay_spectre)
```

Простроив графики мы увидим следующие изображение(см.рисунок4). Таким образом мы можем увидеть, что после прохождения данного звена, синус становится похожим на меандр, с амплитудой равно 1 и частотой равно частоте синуса.

4.2.2 Меандр

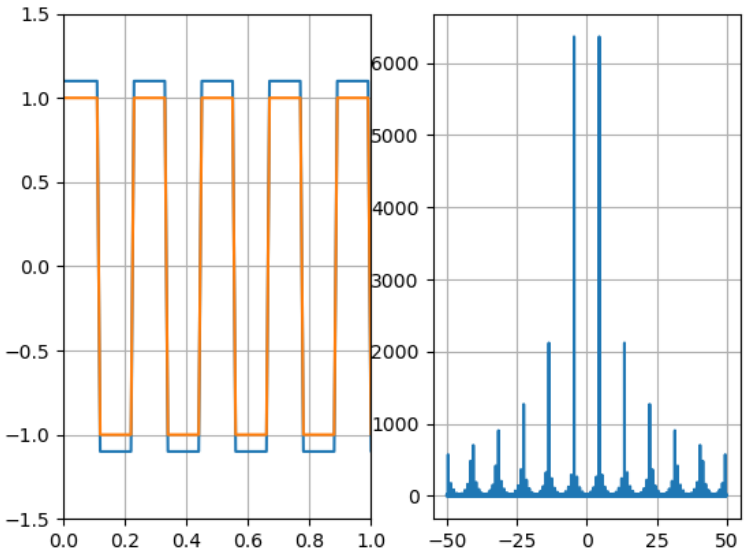

Рисунок 5 — Меандр через реле и спектр выходного сигнала

Здесь мы рассмотрим прохождения меандра через реле. Используем ту же функцию что и в части написанной выше.

```
# Square after relay
sig_square_relay = np.sign(sig_square)
```

```
# spectre square after relay
sig_square_relay_spec = np.abs(np.fft.fft(sig_square_relay))
sig_square_relay_freqs = np.fft.fftfreq(sig_square_relay.shape
```

И сразу же построим график использовав тоже самое о чем было написано выше. В результате должно получится изображение(см.рисунок 5)

Из данных графиков мы можем заметить что форма сигнала никак не меняется, но амплитуда выходного сигнала становится равной единице.

4.2.3 Пилообразный сигнал

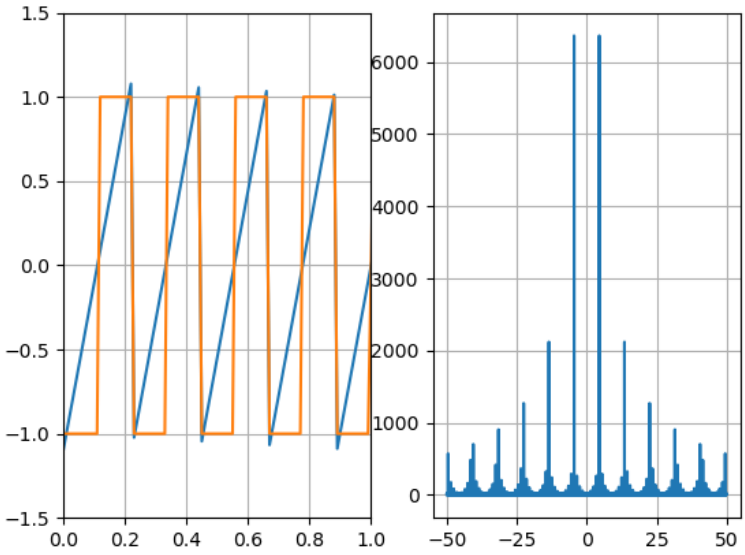

Рисунок 6 — Пилообразный сигнал через реле и спектр выходного сигнала

В этом разделе мы рассмотрим прохождения пилообразного сигнала через реле. Использовав те же функции что и в предыдущих двух главах мы должны получить изображение, которое отображено на рисунке 6.

```
# Sawtooth after relay
sig_saw_relay = np.sign(sig_saw)
# Sawtooth relay spectre
sig_saw_relay_spec = np.abs(np.fft.fft(sig_saw_relay))
sig_saw_relay_freqs = np.fft.fftfreq(sig_saw_relay.shape[0], d
```

Как можно заметить из построенных нами графиков пилообразный сигнал сменился меандром с амплитудой равной 1, а также сжался по частоте. Так же можно заметить что спектр данного сигнала не отличается от спектров выше построенных синуса или меандра.

4.3 Мертвая зона

В данном разделе мы рассмотрим мертвую зону, но для начала изучения ее нужно ее реализовать на языке Python так как ее нет в стандартных библиотеках использованных мной. Реализация этого звена выглядит следующим образом:

```
# function non linear element - dead zone
def dead_zone_scalar(x, wight = 0.5):
        if np.abs(x) < wight:
                return 0
        elif x > 0:
                return x - wight
        else:
                return x + wight

dead_zone = np.vectorize(dead_zone_scalar, otypes = [np.float]
```

4.3.1 Синусоида

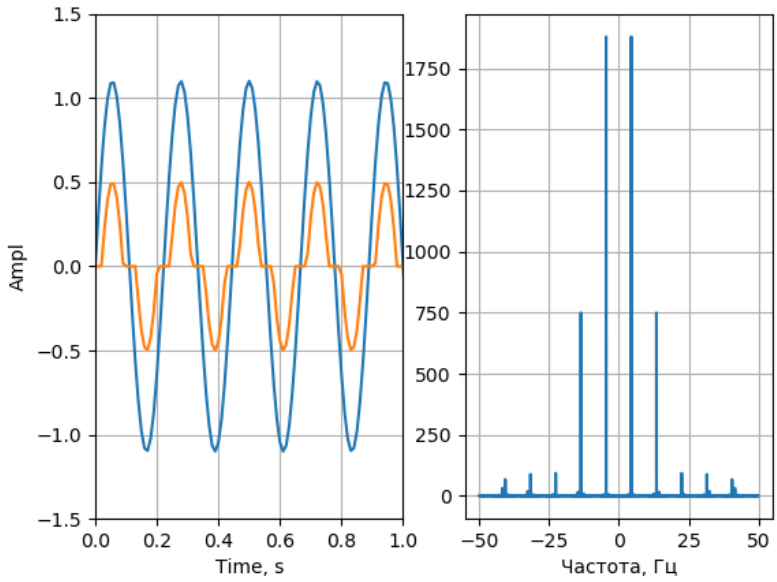

Рисунок 7 — Синус через мертвую зону и спектр выходного сигнала

В данном разделе мы рассмотрим прохождения синуса через типовое не линейное звено мертвая зона и посмотрим на изменения сигнала. Выше мы уже написали код для реализации мертвой зоны, поэтому используем его с не линейным параметром который задается нам следующим выражением: 0.5 + вариант * 0.1, таким образом мой не линейный параметр равен 0.6. Зная все это, мы можем пропустить сигнал через наш элемент.

```
# Sin after dead zone
sig_sin_dz = dead_zone(sig_sin, non_lin_param_1)
# Sin dead zone spectre
sig_sin_dz_spec = np.abs(np.fft.fft(sig_sin_dz))
sig_sin_dz_freqs = np.fft.fftfreq(sig_sin_dz.shape[0], dt)
```

Теперь построим графики входного и выходного сигнала(см.рисунок7), чтобы различия между входным и выходным сигналом были видны.

На графиках мы видим что амплитуда сигнала изменилась и появилась зона в которой сигнал становится равным нулю. Амплитуда уменьшилась на 0.6, т.е. на коэффициент нашего нелинейного звена. Случается это потому что до определенного момента сигнал просто не пропускается данным звеном, что мы видим на графики выходного сигнала(см.рисунок 7, оранжевый)

4.3.2 Меандр

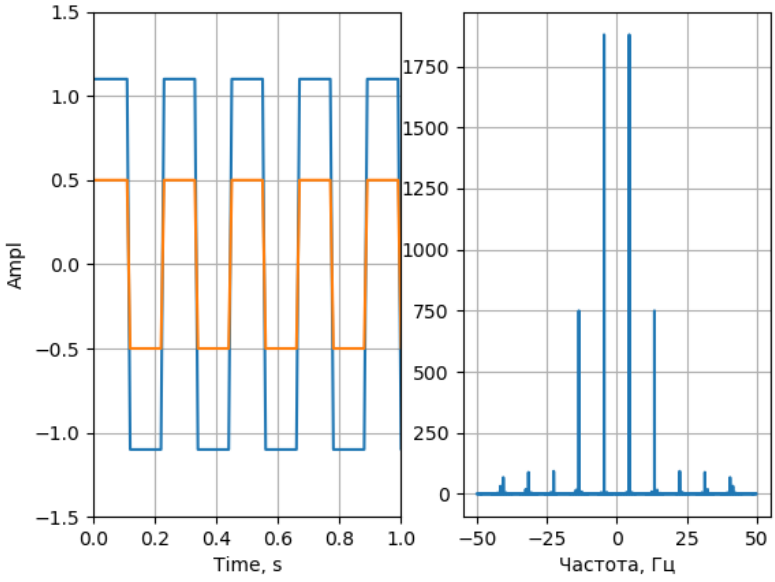

Рисунок 8 — Меандр через мертвую зону и спектр выходного сигнала

В этой секции мы рассмотрим прохождения меандра через мертвую зону. Используем выше описанный алгоритм для имитации прохождения сигнала через мертвую зону и используем выше посчитанный коэффициент.

```
# Square after dead zone
sig_square_dz = dead_zone(sig_square, non_lin_param_1)
# Square after dead zone and spectre
sig_square_dz_spec = np.abs(np.fft.fft(sig_sin_dz))
sig_square_dz_freqs = np.fft.fftfreq(sig_sin_dz.shape[0], dt)
```

Построим графики сигнала входа и выхода(см.рисунок 8), чтобы их проанализировать.

Как мы можем заметить из графиков на рисунке 8 амплитуда сигнала уменьшилась на 0.6 тоже мы увидели и на графике синуса. Описать это можно тем что сигнал все так же не пропускается до частоты 0.6.

4.3.3 Пилообразный сигнал

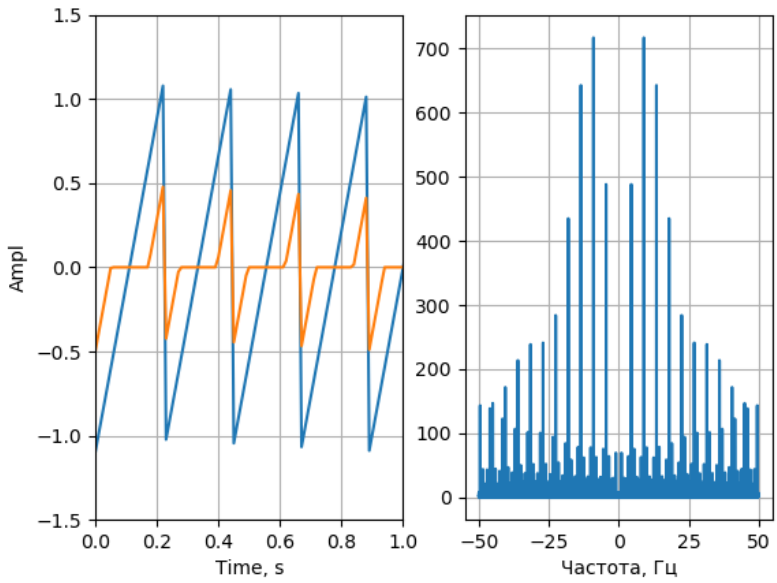

Рисунок 9 — Пилообразный сигнал через мертвую зону и спектр выходного сигнала

В этой части мы рассмотрим прохождение пилообразного сигнала через нелинейный элемент, мертвая зона. Используем все выше использованное на данный сигнал, т.е. алгоритмы мертвой зоны и ее нелинейный коэффициент.

```
# Sawtooth after dead zone
sig_saw_dz = dead_zone(sig_saw, non_lin_param_1)
# Sawtooth after dead zone and spectre
sig_saw_dz_spec = np.abs(np.fft.fft(sig_saw_dz))
sig_saw_dz_freqs = np.fft.fftfreq(sig_saw_dz.shape[0], dt)
```

Построим графики нашего сигнала чтобы убедиться что мы все сделали и верно и проанализировать входной и выходной сигнал, выполнив когда отображения сигнала мы увидим изображение на рисунке 9.

Мы видим что опять амплитуда исходного сигнала уменьшилась на 0.6, происходит это из-за того что наш нелинейный элемент с коэффициентов 0.6 не пропускает сигнал на это промежутке.

4.4 Насыщение

В этой части отчета мы будем смотреть на преобразование сигналов которые проходят через нелинейное звено - насыщение. Для того чтобы посмотреть нам нужно написать функцию которая будет имитировать работу насыщения на языке Python, так как ее нет в стандартных библиотеках. Насыщение на языке Python будет выглядит следующим образом:

```
# Function non linear element - saturation
def saturation_scalar(x, hight = 0.5):
        if np.abs(x) < hight:
                return x
        elif x < hight:
                return -hight
        else:
                return hight
saturation = np.vectorize(saturation_scalar, otypes = [np.float
```

4.4.1 Синусоида

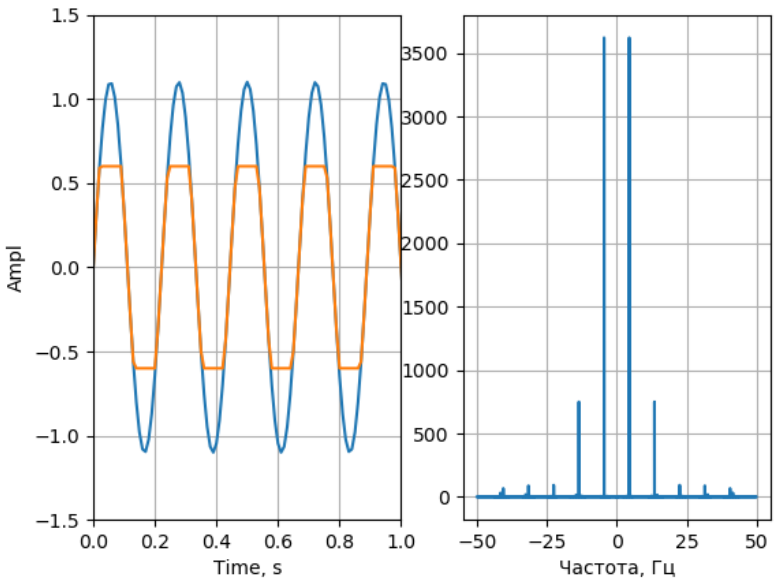

Рисунок 10 — Синус через насыщение и спектр выходного сигнала

В данном разделе мы рассмотрим прохождения сигнала через нелинейный элемент насыщение. Посмотрим на поведение сигнала. Для того что бы это посмотреть мы используем код функции который написали выше и коэффициент нелинейности который рассчитали ранее, таким образом у нас получается следующие:

```
# Sin after saturation
```

```
sig_sin_sat = saturation(sig_sin, non_lin_param_1)
# Sin after saturation spectre
sig_sin_sat_spec = np.abs(np.fft.fft(sig_sin_sat))
sig_sin_sat_freqs = np.fft.fftfreq(sig_sin_sat.shape[0], dt)
```

Построим графики чтобы убедиться что все верно построили и оценить изменения сигнала после прохождения через насыщение (см.рисунок 10).

Как мы можем заметить на рисунке 10 что до какого момента сигнал полностью проходит без искажение, но когда он доходи до нелинейного коэффициента, звено не пропускает сигнал таким образом мы видим срез вершины синуса.

4.4.2 Меандр

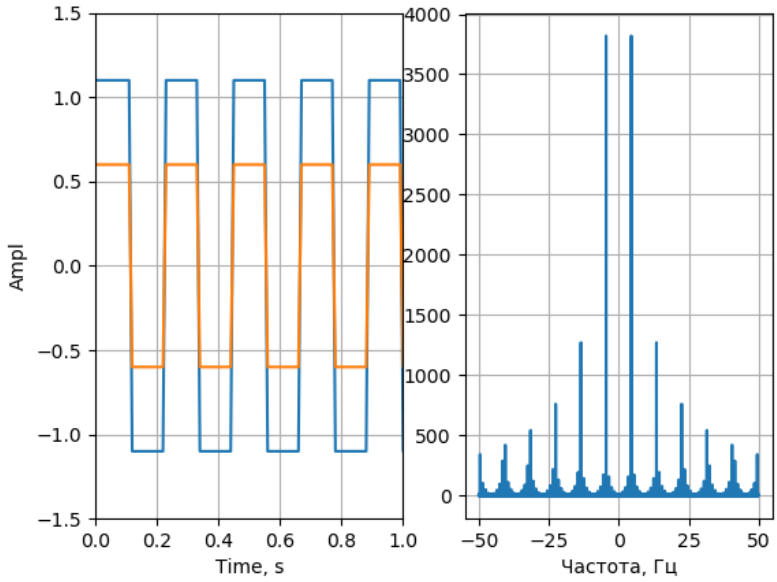

Рисунок 11 — Меандр через насыщение и спектр выходного сигнала

В данном разделе мы посмотрим как меандр проходит через насыщение. Для того чтобы с имитировать прохождение используем все что было написано выше, функцию насыщения и нелинейный элемент.

```
# Square after saturation
sig_square_sat = saturation(sig_square, non_lin_param_1)
# square after saturation spectre
sig_square_sat_spec = np.abs(np.fft.fft(sig_square_sat))
sig_square_sat_freqs = np.fft.fftfreq(sig_square_sat.shape[0],
```

Построим графики входного и выходного сигнала, таким образом после выполнения строк когда по выводу графиков у нас получится рисунок 11.

Из графика на рисунке 11 видно что амплитуда меандра опустила до 0.6, объясняется это тем что при пересечение границы насыщения сигнал выравнивается по уровню насыщения, что мы и видим на графике.

4.4.3 Пилообразный сигнал

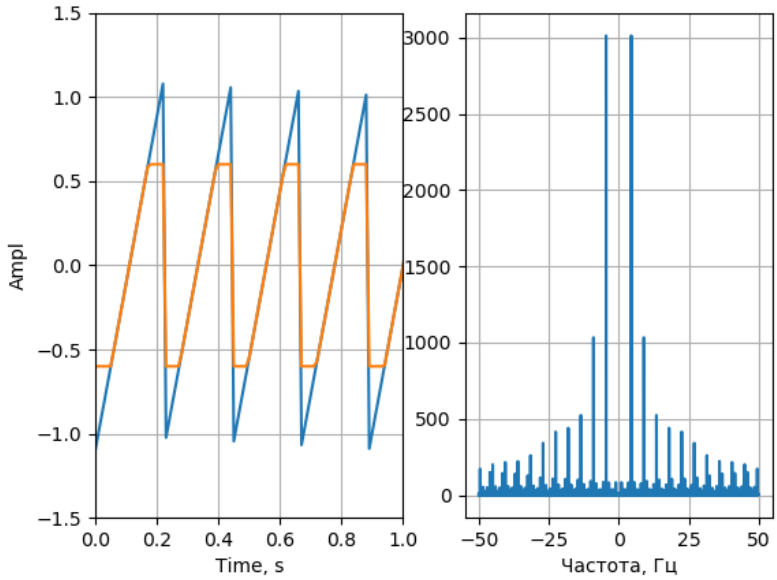

Рисунок 12 — Пилообразный сигнал через насыщение и спектр выходного сигнала

В этой части отчета мы рассмотрим прохождение пилообразного сигнала через насыщение. Для того чтобы имитировать прохождение сигнала мы используем выше описанную функцию и тот же нелинейный коэффициент. Проимитируем прохождение:

```
# sawtooth after saturation
sig_saw_sat = saturation(sig_saw, non_lin_param_1)
# Saw after saturation spectre
sig_saw_sat_spec = np.abs(np.fft.fft(sig_saw_sat))
sig_saw_sat_freqs = np.fft.fftfreq(sig_saw_sat.shape[0], dt)
```

Построим графики входа и выхода(см.рисунок 12) чтобы убедиться что все сделали верно и проанализировать графики.

Из графика (рисунок 12) видно что сигнал до амплитуды 0.6 проходит без изменений, но на амплитуде 0.6 происходит насыщения и элемент остается на своем максимальном значение.

4.5 Сигналы после нелинейного и линейного элемента

Для рассмотрения сигнала после нелинейного и линейного элемента, нам нужно задать линейный элемент. Линейный элемент, по условию, у нас будет в виде апериодического звена. Апериодическое звено задаются следующим образом $H(S) = \frac{k}{(TS+1)}$ для применения его в цифровом виде используем эквивалентный цифровой фильтр $x(i) = -Ax(i-1) + Bu(i)$, где

$A = \frac{1}{1+T/T}$, $B = \frac{k}{1+T/T}$

Для реализации таких фильтров в Python имеется функция scipy.signal.lfilter

4.5.1 Реле

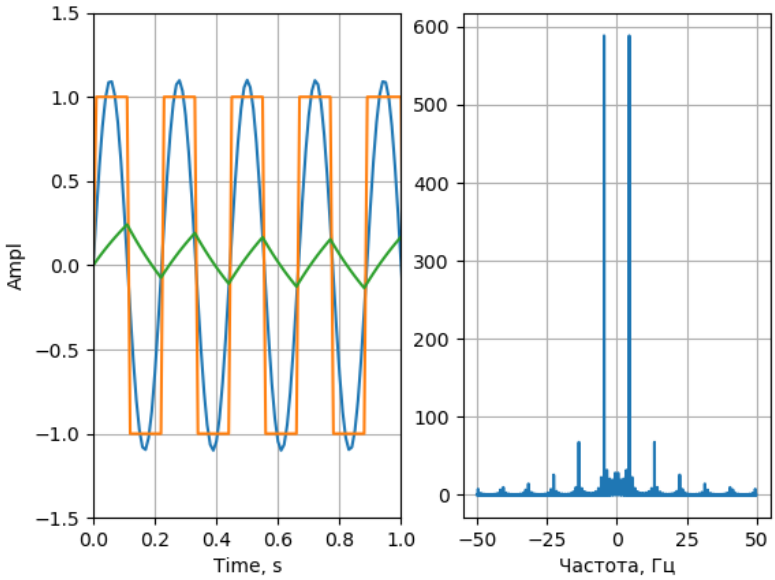

Рисунок 13 — Синус через реле и фильтр и его спектр

Для имитации прохождения сигнала через нелинейный элемент мы возьмем сигнал уже использовавшейся выше в соответствующих разделах и на него используем линейный фильтр. Проимитируем ситуацию следующим образом:

```
# Sin after relay and after lfilter
sig_sin_relay_lb = signal.lfilter(B, A, sig_sin_relay)
# Spectre
sig_sin_relay_lb_spec = np.abs(np.fft.fft(sig_sin_relay_lb))
sig_sin_relay_lb_freqs = np.fft.fftfreq(sig_sin_relay_lb.shape
```

Так же отобразим результат выполнения в графиках, отобразив у нас получится следующие изображение(см.рисунок 13).

Как проходит синус через мертвую зону, мы уже знаем из предыдущего раздела, тут комментарии будут излишними. А когда сигнал проходит через линейный элемент нас интересует, как мы можем заметить, что при прохождение синуса(но после того как синус прошел реле, это меандр) изменяет форму сигнала, на треугольную, плюс мы видим что заметно падает амплитуда.

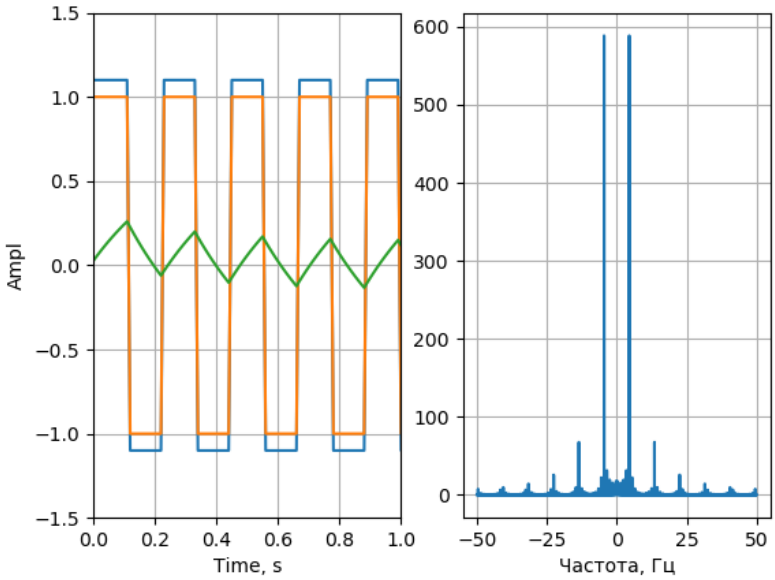

Рисунок 14 — Меандр через реле и фильтр и его спектр

Для прохождения меандра через реле и линейный элемент используем следующий код:

```
# Square after realy and after lfilter
sig_square_relay_lb = signal.lfilter(B, A, sig_square_relay)
# Spectre
sig_square_relay_lb_spec = np.abs(np.fft.fft(sig_square_relay_
sig_square_relay_lb_freqs = np.fft.fftfreq(sig_square_relay_lb
```

Построим графики функции входного сигнала, сигнала через реле и через реле и линейный элемент(см.рисунок **??**).

Как проходит меандр через реле, было описано выше, поэтому переписывать одно и тоже мы не станем. Нас больше интересует как проходит сигнал преодолев реле, но как мы уже знаем меандр не меняет свою форму, у него только падает амплитуда, но когда он проходит через фильтр, он изменяет форму на треугольную, так же резко падает амплитуда. Мы уже заметили что сигнал очень похож на сигнал синуса проходящего через ту же систему.

Для имитации прохождения пилообразного сигнала через линейный и нелинейный элемент, мы используем следующий код:

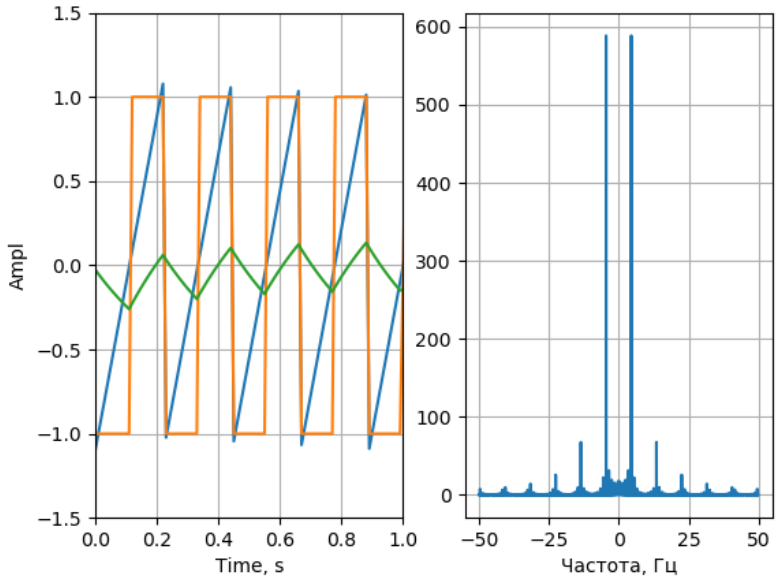

Рисунок 15 — Пилообразный сигнал через реле и фильтр и его спектр

```
# Sawtooth after realy and after lfilter
sig_saw_relay_lb = signal.lfilter(B, A, sig_saw_relay)
# Spectre
sig_saw_relay_lb_spec = np.abs(np.fft.fft(sig_saw_relay_lb))
sig_saw_relay_lb_freqs = np.fft.fftfreq(sig_saw_relay_lb.shape
```

Так же для проверки того что мы все сделали правильно и анализа выходного сигнала мы построили графики которые приведены на рисунке **??**.

Рассмотрим графики на рисунке **??**, мы видим два графика которые мы уже рассматривали выше, а третий график (ломаная зеленого цвета) показывает нам то, как проходит сигнал после реле и фильтра, и мы можем заметить что сигнал изменяет свою форму на треугольную и резко уменьшается амплитуда. Так же мы уже обратили внимание на то, что сигнал очень сильно похож на выше упомянутый сигналы.

Как мы можем заметить из трех графиков что у нас получился на выходе один и тот же сигнал, следовательно из этого можно сделать вывод что для реле и линейного элемента не важна форма сигнала, такая комбинация элемент все равно приведет к одному и тому же варианту сигнала. Так же можно сказать что смысла в такой системе нет, так все сигналы становятся схожими.

4.5.2 Мертвая зона

В данном разделе мы рассмотрим как проходят сигналы через мертвую зону и затем через линейный элемент, проанализируем как изменяется сигнал.

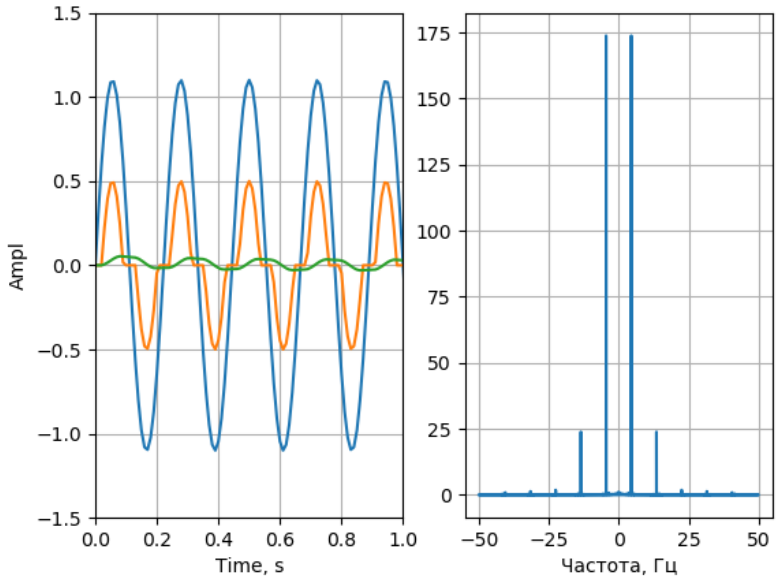

Рисунок 16 — Синус через мертвую зону и фильтр и его спектр

Для имитации прохождения синуса через мертвую зону и фильтр, используем синус после мертвой зоны который был использован в в соответствующем пункте выше. Таким образом, напишем следующий код в Python:

```
# Sin after dead zone and after lfilter
sig_sin_dz_lb = signal.lfilter(B, A, sig_sin_dz)
# Spectre
sig_sin_dz_lb_spec = np.abs(np.fft.fft(sig_sin_dz_lb))
sig_sin_dz_lb_freqs = np.fft.fftfreq(sig_sin_dz_lb.shape[0], d
```

После выведем с помощью средств Python графики соответствующих сигналов(см.рисунок 16)

На графике изображенном на рисунке 16, мы видим три кривых, про две из них уже говорилось выше, в соответствующем разделе, а вот про третью ломанную мы сейчас поговорим. Как мы видим, что пропуская сигнал через такую системы практически загасили сигнал, его амплитуда значительно уменьшилась по сравнению исходным сигналом.

Для того чтобы посмотреть как проходит меандр через мертвую зону и фильтр, мы используем уже полученный сигнал через мертвую зону, и допишем следующий код:

```
# Square after dead zone and after lfilter
sig_square_dz_lb = signal.lfilter(B, A, sig_square_dz)
# Spectre
sig_square_dz_lb_spec = np.abs(np.fft.fft(sig_square_dz_lb))
sig_square_dz_lb_freqs = np.fft.fftfreq(sig_square_dz_lb.shape
```

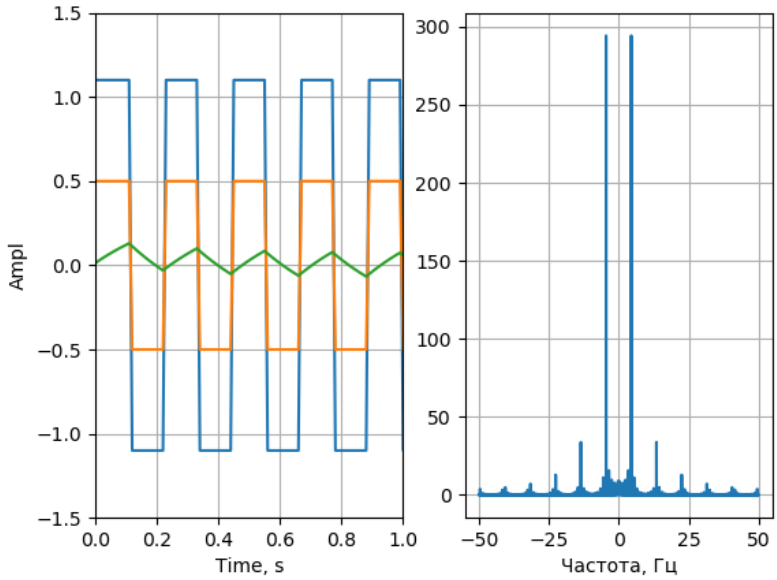

Рисунок 17 — Меандр через мертвую зону и фильтр его спектр

После того как мы с имитировали прохождения сигнала, выведем на экран (см.рисунок **??**) для того чтобы мы наглядно могли проанализировать графики.

Проанализируем выходной сигнал на графике представленном на рисунке **??**. Как мы видим(зеленая ломаная на рисунке **??**) сигнал на выходе изменил свою форму с прямоугольной(меандра) на треугольную, так же очевидно что амплитуда выходного сигнала низка, по сравнению с входным сигналом.

Для того чтобы пронаблюдать как поведет себя пилообразный сигнал после мертвой зоны и линейного элемента, мы воспользуемся соответствующем пунктом который описан выше и преобразованием которое последует ниже.

```
# Sawtooth after dead zone and after lfilter
sig_saw_dz_lb = signal.lfilter(B, A, sig_saw_dz)
# Spectre
sig_saw_dz_lb_spec = np.abs(np.fft.fft(sig_saw_dz_lb))
sig_saw_dz_lb_freqs = np.fft.fftfreq(sig_saw_dz_lb.shape[0], d
```

После имитации выведем графики сигналов(см.рисунок 18) для того чтобы убедиться что мы сделали все верно и проанализировать получившийся выход.

Проанализируем действие системы на пилообразный сигнал. Как мы можем заметить после прохождения сигнала через систему (мертвая зона и фильтр), сигнал практически был заглушен, есть не значительные колебания вокруг нуля. Что говорит о том, что это сигнал слаб для такого соединения элементов, его практически глушат.

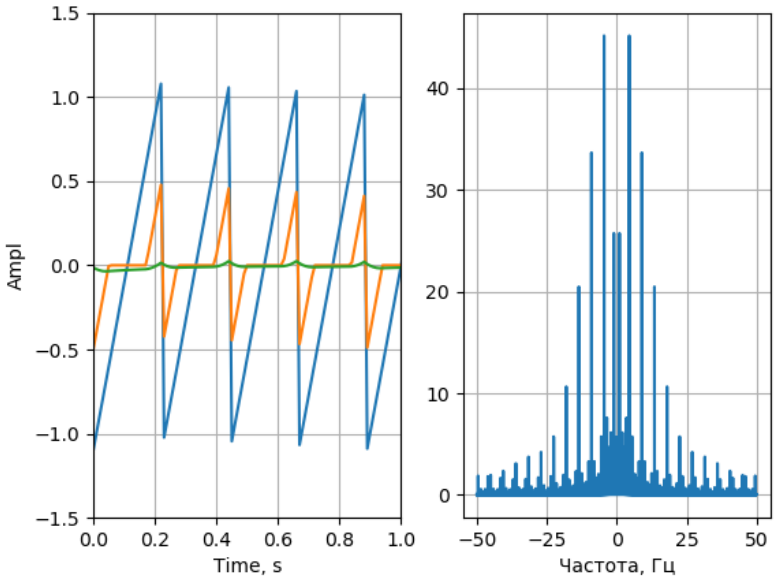

Рисунок 18 — Пилообразный сигнал через мертвую зону и фильтр и его спектр

Подведя итоги анализов данного соединения элементов можно сказать, что после такого соединения сигнал становится с очень маленькой амплитудой, его сигнал чуть отличим от нуля. Как можно заметить самым слабым сигналом для этого соединения оказался пилообразный сигнала, самым же сильным оказался меандр, это можно увидеть из графиков спектров данных сигналов.

4.5.3 Насыщение

В данном разделе мы рассмотрим, как проходят типовые сигналы через насыщение и линейный элемент.

Для того чтобы посмотреть как проходит синус через насыщение и фильтр одновременно, мы используем полученный раннее сигнал синуса прошедший через насыщение, и добавим к нему фильтр(как будет показано ниже) получим наш сигнал для исследования.

```
# Sin after saturation and after lfilter
sig_sin_sat_lb = signal.lfilter(B, A, sig_sin_sat)
# Spectre
sig_sin_sat_lb_spec = np.abs(np.fft.fft(sig_sin_sat_lb))
sig_sin_sat_lb_freqs = np.fft.fftfreq(sig_sin_sat_lb.shape[0],
```

Построим также график(см.рисунок 19) чтобы было легче проанализировать изменения сигнала и оценить правильность выполнения.

Как мы можем заметить из графиков на рисунке 19, из данных графиков мы видим что после линейного элемента синус практически восста-

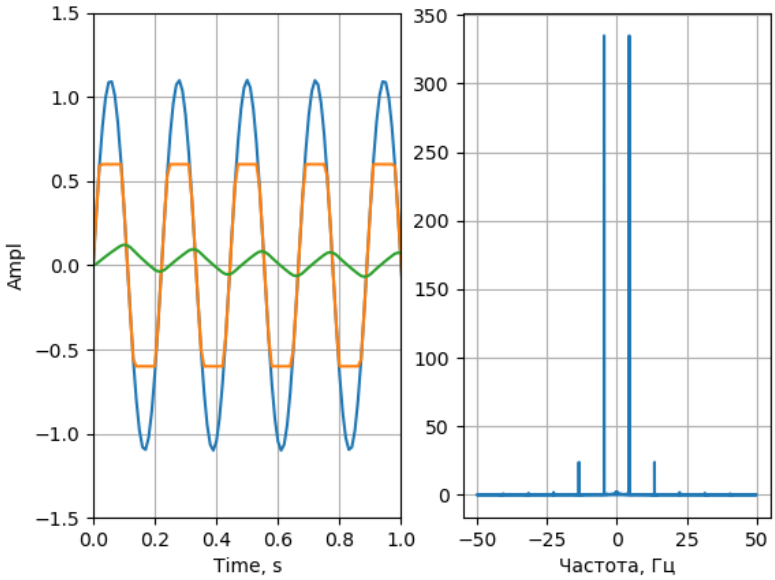

Рисунок 19 — Синус через насыщение и фильтр и его спектр

новил свою форму, но после такой системы у него заметно уменьшилась амплитуда.

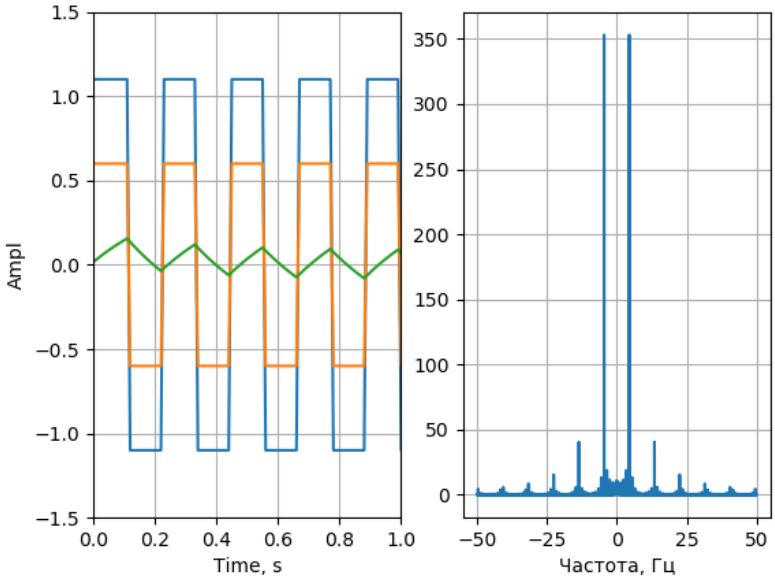

Рисунок 20 — Меандр через насыщение и фильтр и его спектр

Для наблюдения изменения меандра после прохождения насыщения и после фильтра, мы используем полученный, в соответствующем разделе, сигнал и применим на него фильтр следующим образом:

```
# Square after saturation and after lfilter
sig_square_sat_lb = signal.lfilter(B, A, sig_square_sat)
# Spectre
```

```
sig_square_sat_lb_spec = np.abs(np.fft.fft(sig_square_sat_lb))
sig_square_sat_lb_freqs = np.fft.fftfreq(sig_square_sat_lb.sha
```

Чтобы увидеть что у нас получилось на выходи и сравнить с тем что было на входе построим график(см.рисунок 20).

Как можно заметить график на рисунке 20 совпадает с графиков на рисунке 17, но если посмотреть на графики спектра то можно увидеть, что сигнал после насыщения и фильтра сильнее, но это очевидно, так как сигнал после насыщения сильнее, чем сигнал через мертвую зону.

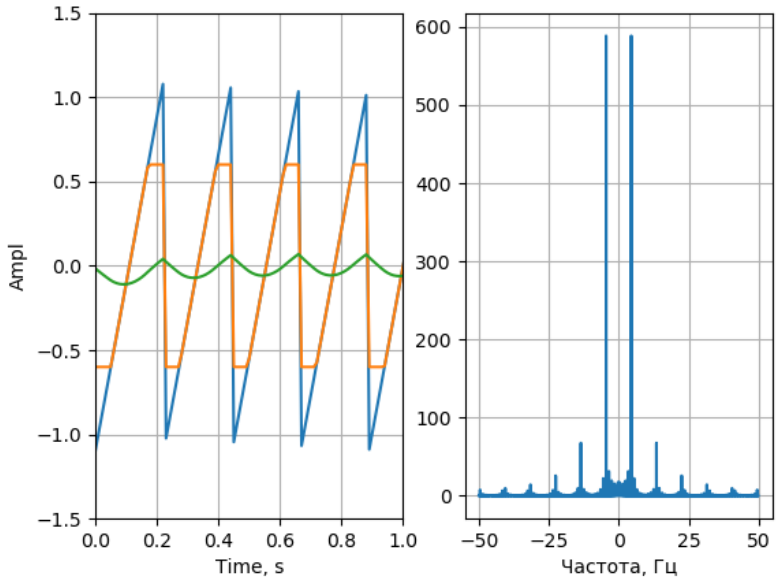

Рисунок 21 — Пилообразный сигнал через насыщение и фильтр и его спектр

Для имитации прохождения пилообразного сигнала через насыщение и фильтр, мы возьмем ранее полученный сигнал, и допишем пару строк кода.

```
# Sawtooth after saturation and after lfilter
sig_saw_sat_lb = signal.lfilter(B, A, sig_saw_sat)
# Spectre
sig_saw_sat_lb_spec = np.abs(np.fft.fft(sig_saw_relay_lb))
sig_saw_sat_lb_freqs = np.fft.fftfreq(sig_saw_sat_lb.shape[0],
```

Построим графики прохождения через каждый элемент, отследим и проанализируем изменения(см.график 21)

Как мы видим из графиков на рисунке 21, сигнал после прохождения заметно ослаб, но что не мало важно, он кардинально изменил форму.

Пропустив все типы тестовых сигналов через комбинации нелинейных элементов (реле, мертвая зона и насыщение) и линейного(фильтр), а также проанализировав результат мы заметили что после прохождения

данной системы сигнал меняет форму и у всех сигналов значительно уменьшается амплитуда, этим и сказывается действие фильтра после нелинейного элемента. Так же можно сказать что такая система чувствительна к сигналу, на каждый сигнал будет уникальный выход.

4.6 Сигнал после линейного и нелинейного элемента

В этом разделе мы рассмотрим сигналы проходящий сначала через линейный элемент(фильтр), а после уже нелинейных элементов трех типов (реле, мертвая зона, насыщение), про тестируем такую связь с помощью трех видов пробных сигналов.

4.6.1 Реле

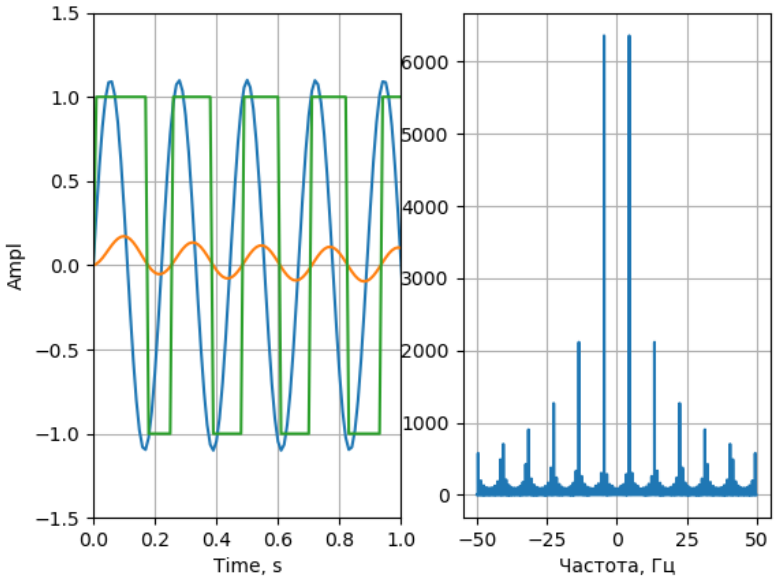

Рисунок 22 — Синус через фильтр и реле и его спектр

Для того чтобы рассмотреть сигнала синуса через линейный элемент и реле мы написали следующие строки в языке Python:

```
# Sin after lfilter and after relay
sig_sin_ln = signal.lfilter(B, A, sig_sin)
sig_sin_ln_relay = np.sign(sig_sin_ln)
# Spectre
sig_sin_ln_relay_spec = np.abs(np.fft.fft(sig_sin_ln_relay))
sig_sin_ln_relay_freqs = np.fft.fftfreq(sig_sin_ln_relay.shape
```

Для того чтобы проанализировать графики входного и выходного сигнала выведем их с помощью функции plt.plot()(см.рисунок 22)

Проанализируем полученный график на рисунке 22, первым в данном случае идет линейный элемент что мы и видим на изображение(оранжевая кривая), мы видим что у него резко изменилась амплитуда, но что не мало важно мы видим что после прохождения данного элемента синус получил сдвиг по фазе. После преобразования реле сдвиг становится очевидным, особенно если сравнивать с графиком на рисунке 13.

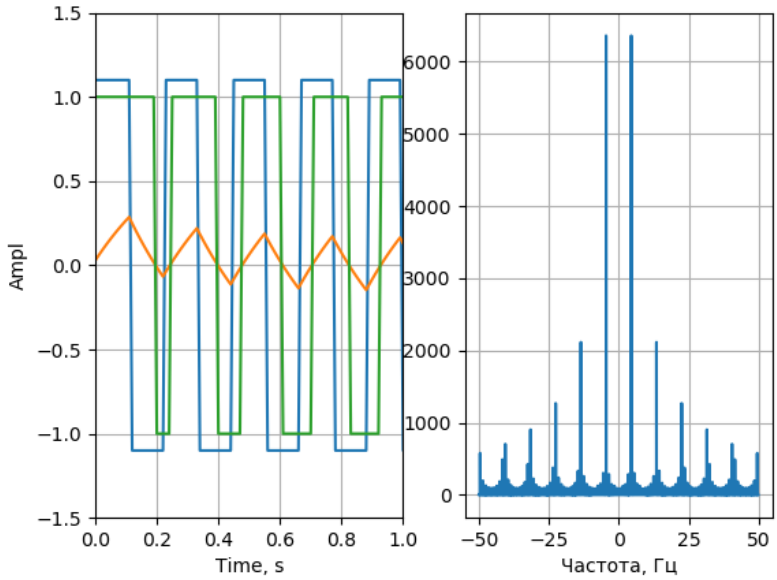

Рисунок 23 — Меандр через фильтр и реле и его спектр

Для того чтобы проимитировать прохождение меандра через линейный элемент и реле, мы используем следующие функции которые будут приведены ниже:

```
# Square after lfilter and after relay
sig_square_ln = signal.lfilter(B, A, sig_square)
sig_square_ln_relay = np.sign(sig_square_ln)
# Spectre
sig_square_ln_relay_spec = np.abs(np.fft.fft(sig_square_ln_rela
sig_square_ln_relay_freqs = np.fft.fftfreq(sig_square_ln_relay
```

Далее чтобы проанализировать график входа и выхода мы построим графики которые будут приведены на рисунке **??**.

Проанализируем полученный данный на рисунке **??**. Как мы можем заметить после прохождения фильтра наш сигнал изменил и форму и амплитуду, причем и то и другое довольно сильно. Но после прохождения нелинейного элемента(реле), он вернул свою прежнюю форму меандра, так же можно сказать что он практически вернул свою амплитудное значение, но это только потому что наш входной сигнал был таковым.

Для анализа пилообразного сигнала через линейный элемент и реле, мы построим его следующим образом:

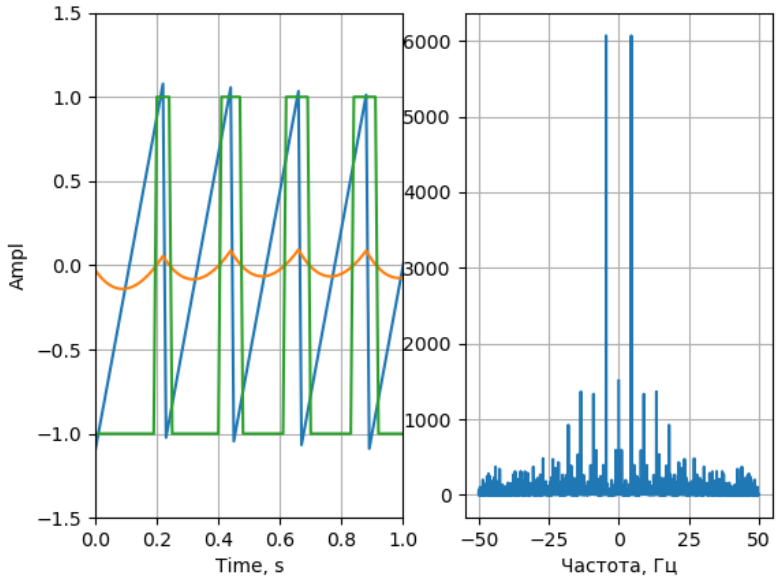

Рисунок 24 — Пилообразный сигнал через фильтр и реле и его спектр

```
# Swatooth after lfilter and after realy
sig_saw_ln = signal.lfilter(B, A, sig_saw)
sig_saw_ln_relay = np.sign(sig_saw_ln)
# Spectre
sig_saw_ln_relay_spec = np.abs(np.fft.fft(sig_saw_ln_relay))
sig_saw_ln_relay_freqs = np.fft.fftfreq(sig_saw_ln_relay.shape
```

Для анализа сигнала до прохождения система и после воспользуемся средствами Python и выведем графики сигнала до и после прохождения(см.рисунок **??**)

Рассмотрим графики на рисунке **??**. Мы видим что наш треугольный сигнал после прохождения фильтра изменил и форму, и амплитуду сигнала причем кардинально. Но потом наш сигнал проходит через реле и что мы видим, мы видим меандр, причем это меандр нам очень напоминает меандр полученный на графике 6, только перевернутый.

В заключение к этой главе можно сказать что это соединение элементов чувствительно к сигналу, если пропуская через это соединения синус и меандр, мы очевидным образом заметили что там все также получается меандр, но так же здесь становится заметно и то что линейное звено добавляется нам на сигнал сдвиг по фазе, что касается пилообразного сигнала тут не все так очевидно мы заметили что наш сигнал стал перевернутым относительно сигнала который получается на графике 6

4.6.2 Мертвая зона

Для рассмотрения сигнала после фильтра и мертвой зоны, мы проимитируем это прохождение на языке Python. Сразу же для оценки сигнала

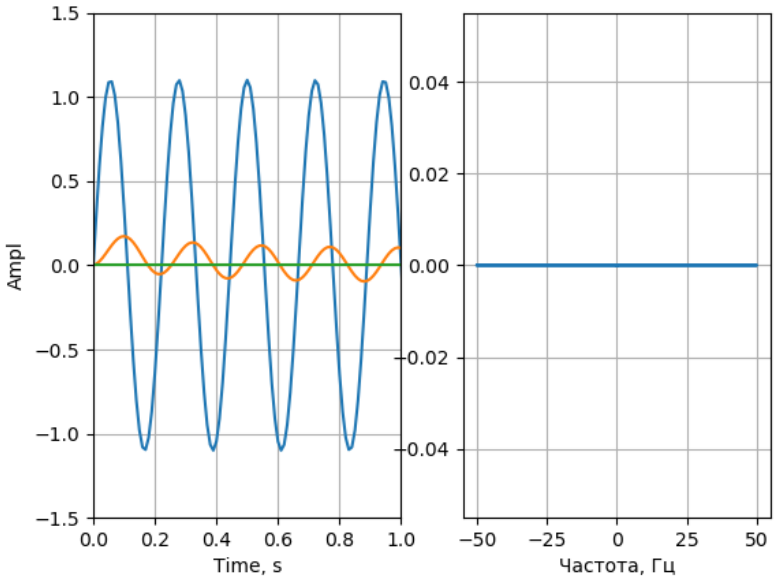

Рисунок 25 — Синус через фильтр и мертвую зону и его спектр

построим графики спектра выходного сигнала.

```
# Sin after lfilter and after dead zone
sig_sin_ln = signal.lfilter(B, A, sig_sin)
sig_sin_ln_dz = dead_zone(sig_sin_ln, non_lin_param_1)
# Spectre
sig_sin_ln_dz_spec = np.abs(np.fft.fft(sig_sin_ln_dz))
sig_sin_ln_dz_freqs = np.fft.fftfreq(sig_sin_ln_dz.shape[0], d
```

Для анализа выходного сигнала выведем его, а также его изменения на каждом этапе прохождения элементов, таким образом у нас должен получаются график продемонстрированные на рисунке 25.

Как можно заметить из графика(см.рисунок 25), отфильтровав сигнала мы заметно уменьшили его амплитуду тем самым, мы его сделали слабым, и ему не хватает силы пройти зону нечувствительности. Таким образом можно сказать что такая система, несколько бесполезна.

Для имитации прохождения меандра через фильтр и мертвую зону, мы написали следующий код на языке Python:

```
# Sin after lfilter and after dead zone
sig_sin_ln = signal.lfilter(B, A, sig_sin)
sig_sin_ln_dz = dead_zone(sig_sin_ln, non_lin_param_1)
# Spectre
sig_sin_ln_dz_spec = np.abs(np.fft.fft(sig_sin_ln_dz))
sig_sin_ln_dz_freqs = np.fft.fftfreq(sig_sin_ln_dz.shape[0], d
```

Для анализа входного сигнала построим графики сигналов до входа в систему, после прохождения первого элемента системы(фильтра) и

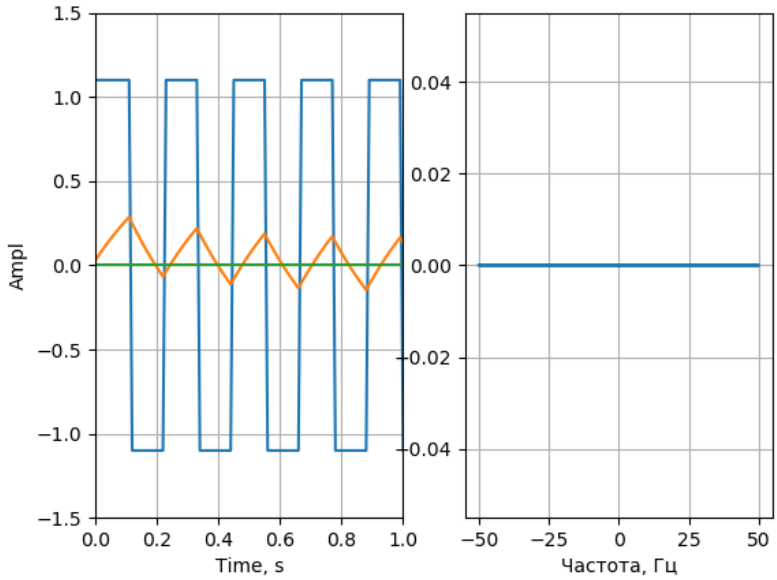

Рисунок 26 — Меандр через фильтр и мертвую зону и его спектр

сам выходной сигнал, таким образом у нас получился следующий график(см.рисунок **??**).

Теперь проанализируем графики на рисунке **??**, что мы видим из графиков, а мы видим то, что меандр проходя через линейный элемент изменил форму на треугольную, но в данном случае это не так важно, его амплитуда заметно сократилась. Из-за сокращения амплитуду сигнал не проходит мертвую зону, таким образом на выходе мы получаем 0, то есть нет сигнала.

Чтобы посмотреть как изменяется пилообразный сигнал после прохождения фильтра и мертвой зоны, мы создали следующий код:

```
# Square after lfilter and after dead zone
sig_square_ln = signal.lfilter(B, A, sig_square)
sig_square_ln_dz = dead_zone(sig_square_ln, non_lin_param_1)
# Spectre
sig_square_ln_dz_spec = np.abs(np.fft.fft(sig_square_ln_dz))
sig_square_ln_dz_freqs = np.fft.fftfreq(sig_square_ln_dz.shape
```

Для анализа меандр после прохождения системы из фильтра и мертвой зоны построим, графики выходного сигнала на каждом этапе прохождения, таким образом получаем графики которые изображены на рисунке **??**

Рассмотрим графики полученные на рисунке **??**. Как мы можем наблюдать что пилообразный сигнал после прохождения изменил форму, но в данном случае это не играет ни какой роли, так как нам нужно проанализировать амплитуду, и очевидно что она стала намного меньше, и не способна пробить мертвую зону. Мы видим выходной сигнал ноль, как и

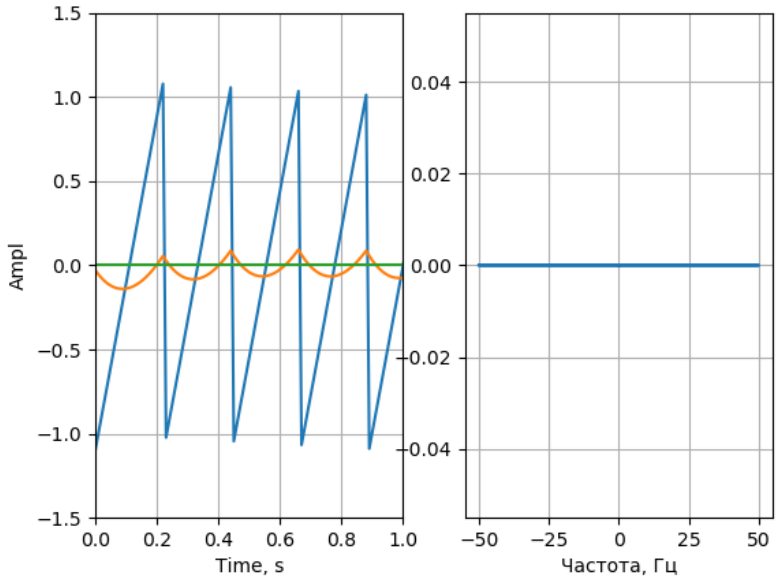

Рисунок 27 — Пилообразный сигнал через фильтр и мертвую зону и его спектр

было написано выше это все из-за того что мы не смогли "пробить"мертвую зону, поэтому мы видим отсутствие сигнала.

Проанализировав графики трех выходных сигналов, через систему из линейного элемента и мертвой зоны, мы можем сделать вывод что такая система особого смысла нет, так как он просто на просто загасил наш сигнал полностью.

4.6.3 Насыщение

В этой секции мы рассмотри прохождения всех типовых сигналов данной лабораторной работы(синус, меандр, пилообразный сигнал), через три вида нелинейных элемента, после того как сигнал пройдет через линейный элемент(фильтр). Таким образом мы сможем проанализировать поведения такой системы.

```
# Sin after lfilter and after saturation
sig_sin_ln = signal.lfilter(B, A, sig_sin)
sig_sin_ln_sat = saturation(sig_sin_ln, non_lin_param_1)
# Spectre
sig_sin_ln_sat_spec = np.abs(np.fft.fft(sig_sin_ln_sat))
sig_sin_ln_sat_freqs = np.fft.fftfreq(sig_sin_ln_sat.shape[0],
```

Для анализа сигнала выведем его, что продемонстрировано на графике(см.рисунок 28).

Как мы описывали в главах выше синус после прохождения фильтра не особо изменил свою форму, но амплитуда у него значительно уменьши-

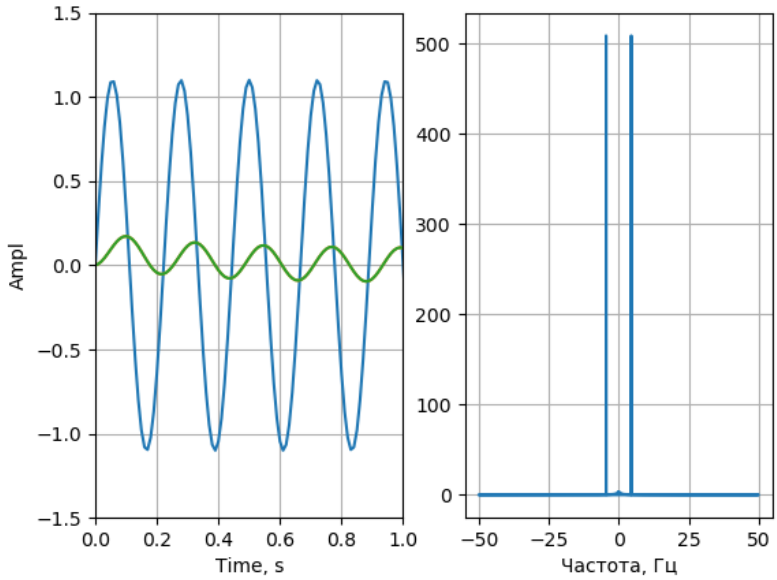

Рисунок 28 — Пробный сигнал синуса через фильтр и насыщение и его спектр

лась к тому же у него еще появился сдвиг по фазе, но все это не значительно по сравнению с преобразованием амплитуды, так как амплитуда после фильтра стала значительно ниже, амплитуда сигнала на выходе просто не достигает значения насыщения.

Для того что продемонстрировать прохождения сигнала через линейный элемент, воспользуемся инструмента языка программирования Python и напишем следующий код:

```
# Square after lfilter and after relay
sig_square_ln = signal.lfilter(B, A, sig_square)
sig_square_ln_sat = saturation(sig_square_ln, non_lin_param_1)
# Spectre
sig_square_ln_sat_spec = np.abs(np.fft.fft(sig_square_ln_sat))
sig_square_ln_sat_freqs = np.fft.fftfreq(sig_square_ln_sat.sha
```

Далее построим графики(см.рисунок 29) сигнала на каждом этапе, чтобы в дальнем проанализировать изменения на каждом этапе, и сделать выводы о системе и ее реакции на сигнал.

Как мы уже видели в секциях написанных выше, наш меандр изменил форму на треугольную, с значительным уменьшением амплитуды. Тогда, зная это и видя график на рисунке 29, мы можем с уверенностью сказать, что сигнал на выходе никогда не пересекает границу насыщение, тем самым мы видим сигнал после преобразования, чисты сигнал после преобразования.

Сейчас мы посмотрим на пилообразный сигнал после прохождения системы из двух элементов(фильтр и насыщение), использовав функции

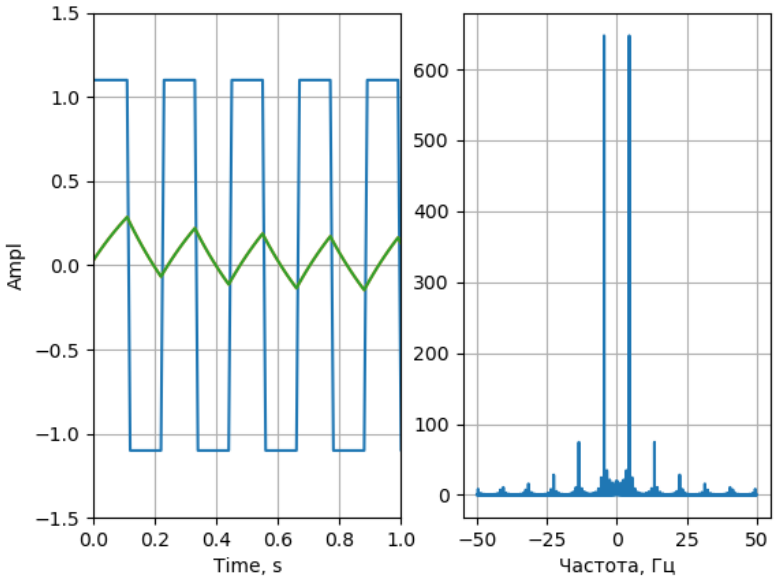

Рисунок 29 — Пробный сигнал меандра через фильтр и насыщение и его спектр

Python.

```
# Swatooth after lfilter and after realy
sig_saw_ln = signal.lfilter(B, A, sig_saw)
sig_saw_ln_sat = saturation(sig_saw_ln, non_lin_param_1)
# Spectre
sig_saw_ln_sat_spec = np.abs(np.fft.fft(sig_saw_ln_sat))
sig_saw_ln_sat_freqs = np.fft.fftfreq(sig_saw_ln_sat.shape[0],
```

Теперь выведем графики пилообразного сигнала(см. рисунок 30)) чтобы мы могли проанализировать сигнал на каждом этапе, а графически это будет нагляднее.

Как мы уже видели в главах выше, фильтр изменил наш пилообразный сигнал, на совершенно другой сигнал, но мы видим это сигнал так как сигнал на выходе после линейного элемент понизил амплитуду на столько что, сигнал не достигает границу насыщения, что и видно на графике (см.рисунке 30).

Как мы можем заметить из трех изображений графиков приведенных выше, что использование такой системы приводит к уменьшений амплитуды на столько что наш сигнал не пересекает границу насыщения. Поэтому в данном случае можно сказать что в такой системе можно оставить только линейное звено, или если мы хоти увидеть изменения сигнала после насыщение, то стоит сделать границу насыщения ниже.

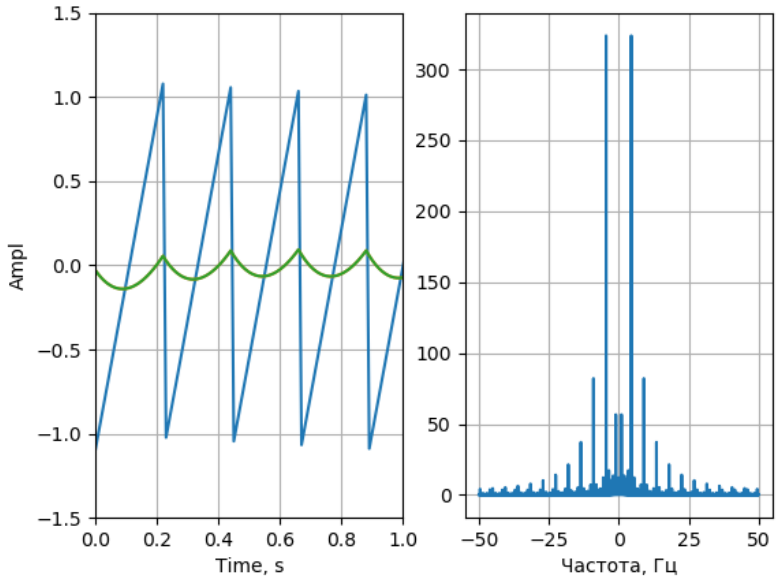

Рисунок 30 — Пробный пилообразный сигнал через фильтр и насыщение и его спектр

5 ВЫВОДЫ

В данном разделе мы подведем итоги лабораторной работы, обобщим полученный выводы и вынесем основные моменты.

Сделав эту лабораторную работу мы научились средствами языка программирования Python, мы изучили некоторые функции библиотеки SciPy и бибилиотки NumPy. Мы построили три пробных сигнала с помощью языка Python, так же мы создали три вида нелинейных элемента: реле, мертвая зона, насыщение. Мы посмотрели на реакцию этих элементов на каждый вид созданного нами тестового сигнала. Так же мы рассмотрели комбинацию нелинейный элемент - линейный элемент и комбинацию линейный элемент - нелинейный элемент.

Когда мы рассматривали комбинации элементов мы заметили то, что важна последовательность этой комбинации, так как на выходе получаются совершенно разные сигналы. Следовательно эти элементы не коммутативны.

Так же мы заметили что нелинейный элементы влияли только на амплитуду, в отличие от линейных элементов которые так же влияли и на фазу.