Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный Исследовательский Университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №2
ПО ПРЕДМЕТУ «ЧАСТОТНЫЕ МЕТОДЫ»
ПО ТЕМЕ «ПРЕОБРАЗОВАНИЕ ФУРЬЕ»

Лектор: Перегудин А. А.
Практик: Пашенко А. В.
Студент: Румянцев А. А.
Поток: ЧАСТ.МЕТ. 1.3

Факультет: СУиР
Группа: R3241

Санкт-Петербург
2024

Содержание

1 Введение

В заданиях 1 и 2 используется унитарное преобразование Фурье к угловой частоте ω. Подсчет Фурье-образа производится по формуле ниже

$$\hat{f}\left(\omega\right) = \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt$$

В задании 3 используется преобразование Фурье к обыкновенной частоте ν. В общем виде формула имеет вид

$$\hat{f}\left(\nu\right) = \int\limits_{-\infty}^{\infty} f\left(t\right) e^{-2\pi i \nu t} \, dt$$

Для проверки равенства Парсеваля используется формула ниже

$$||f||_2 = ||\hat{f}||_2,$$

где $||f||_2$ – вторая норма заданной фукнции, $||\hat{f}||_2$ – вторая норма Фурье-образа функции f. Для нахождения нормы используются формулы, представленные ниже

$$||f(t)||_2 = \sqrt{\int\limits_a^b f(t) \cdot f^*(t) \, dt}, \qquad ||\hat{f}(\omega)||_2 = \sqrt{\int\limits_a^b \hat{f}(\omega) \cdot \hat{f}^*(\omega) \, d\omega}$$

Все графики строятся программой, написанной на языке программирования python. В 1 и 2 заданиях используется библиотека sympy, в задании 3 numpy и matplotlib. По ходу отчета приводится код для каждого задания. Для всех интегралов и графиков есть место с общими переменными и значениями – файл static.py. Основные используемые данные приведены ниже

```
from sympy import Symbol, Piecewise, Abs, sinc, E, oo

t = Symbol('t')
omega = Symbol('omega')

interval = [-oo, oo]

a_b_pars = [(1, 2), (2, 3), (3, 4)]
consts = [-1, 0.5, 1]
colors_strs = ['red', 'purple', 'blue', 'cyan']
```

Листинг 1: Основные данные из файла static.py

В этом файле программно заданы функции, графики которых приводятся по ходу отчета. Также они необходимы для нахождения их Фурье-образа

```
def rectangular_function(a, b):
    return Piecewise((a, Abs(t) <= b), (0, Abs(t) > b))

def triangular_function(a, b):
    return Piecewise((a - Abs(a * t / b), Abs(t) <= b), (0, Abs(t) > b))

def cardinal_sinus(a, b):
    return a * sinc(b * t)

def gaussian_function(a, b):
    return a * E ** (-b * t ** 2)

def double_attenuation(a, b):
    return a * E ** (-b * Abs(t))
```

Листинг 2: Программно заданные функции для заданий 1 и 2

2 Задание 1. Вещественное

2.1 Прямоугольная функция

Рассмотрим прямоугольную функцию следующего вида

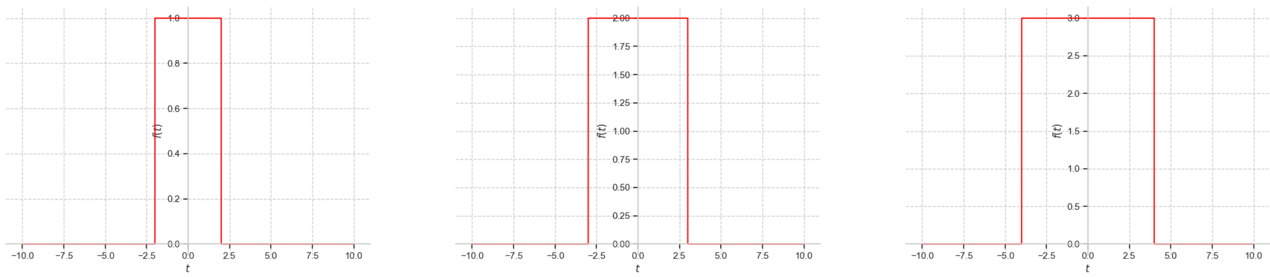
$$f(t) = \begin{cases} a, & |t| \le b, \\ 0, & |t| > b. \end{cases}$$

Рассмотрим программу. Сначала задается функция, принимающая параметры a и b, после чего методом build_f_t строится график $f(t)$. На строке 10 приведен пример использования кода

```
1     def build_f_t(f_t, clr, lbl):
2         if (lbl == None):
3             plot(f_t, line_color=clr, xlabel=r'$t$', ylabel=r'$f(t)$')
4         else:
5             plot(f_t, line_color=clr, xlabel=r'$t$', ylabel=r'$f(t)$', label=lbl, legend=True)
6
7     build_f_t(rectangular_function(1, 2), 'red', None)
```

Листинг 3: Программа для построения графика прямоугольной функции

Построенные графики $f(t)$ для нескольких значений параметров $a, b > 0$ расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 1: Прямоугольные функции при различных значениях a и b

Рассмотрим программу. Методом find_fimg находится Фурье-образ заданной функции. Далее методом build_fimg2 строится график Фурье-образа. На 16-17 строчках находится пример использования кода

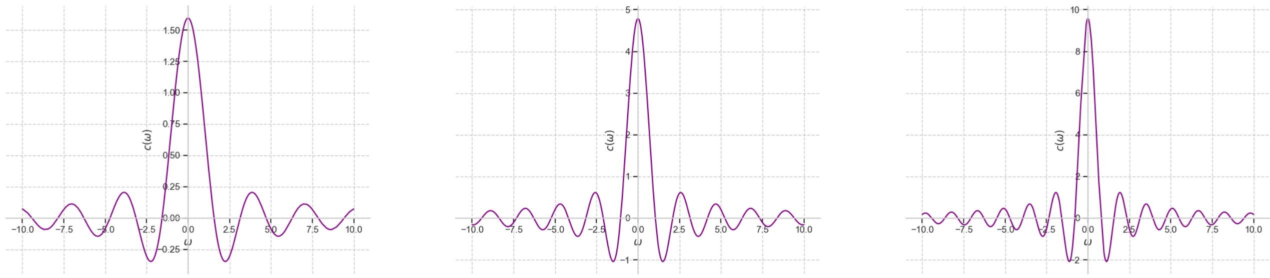
```
1     def find_fimg(f_t, lim1, lim2):
2         integrand = f_t * E ** (-I * omega * t)
3
4         result = integrate(integrand, (t, lim1, lim2))
5         return coeff * result
6
7     def build_fimg2(fimg, clr, lbl):
8         if (lbl == None):
9             plot(fimg, line_color=clr,
10                 xlabel=r'$\omega$', ylabel=r'$c(\omega)$')
11        else:
12            plot(fimg, line_color=clr,
13                 xlabel=r'$\omega$', ylabel=r'$c(\omega)$',
14                 label=lbl, legend=True)
15
16    rectfimg = find_fimg(rectangular_function, -oo, oo)
17    build_fimg2(rectfimg, 'purple', None)
```

Листинг 4: Программа для построения графика Фурье-образа некоторой функции $f(t)$

Построенные графики $\hat{f}(\omega)$ для тех же значений a и b расположены ниже

Рассмотрим программу. Методом find_norm2 находится вторая норма переданной функции. Метод find_parseval считает левую и правую части равенства Парсеваля. Пример использования кода расположен на 13-14 строчках листинга ниже

```
1     def find_norm2(f, lim1, lim2, var):
2         integrand = f * conjugate(f)
```



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 2: Фурье-образы прямоугольных функций при различных значениях a и b

```

    result = integrate(integrand, (var, lim1, lim2)).evalf()
    return sqrt(result).evalf()

def find_parseval(f, fimg, lim1, lim2):
    pleft = find_norm2(f, lim1, lim2, t)
    pright = find_norm2(fimg, lim1, lim2, omega)

    return pleft, pright

pl, pr = find_parseval(rectangular_function, rectfimg, -oo, oo)
print(f'p_{1}: {pl} ?= {pr}')
```

Листинг 5: Программа для нахождения левой и правой сторон равенства Парсеваля

Программа вывела в консоль результаты, представленные ниже

```
p_1: 2.00000000000000 ?= 2.0 + 0.e-114*I
p_2: 4.89897948556636 ?= 4.89897948556636 + 0.e-114*I
p_3: 8.48528137423857 ?= 8.48528137423857 + 0.e-114*I
```

Листинг 6: Результат выполнения программы для вычисления равенства Парсеваля

Мнимыми частями в правой части равенства Парсеваля пренебрежем вследствие их стремления к нулю. В таком случае равенство Парсеваля выполняется, что можно объяснить тем, что интеграл позволяет рассмотреть норму непрерывно на заданном промежутке, а ряд только дискретно, вследствие чего теряются какие-то члены ряда, которых не хватает для выполнения равенства Парсеваля

2.2 Треугольная функция

Рассмотрим треугольную функцию следующего вида

$$f(t) = \begin{cases} a - \left|\dfrac{at}{b}\right|, & |t| \leq b, \\ 0, & |t| > b. \end{cases}$$

Для построения графиков треугольной функции сначала задается необходимая функция, после используется код, приведенный в пункте 2.1 для прямоугольной функции

```
null
```

Листинг 7: Программно заданная треугольная функция

Построенные графики $f(t)$ для нескольких значений параметров $a, b > 0$ расположены ниже

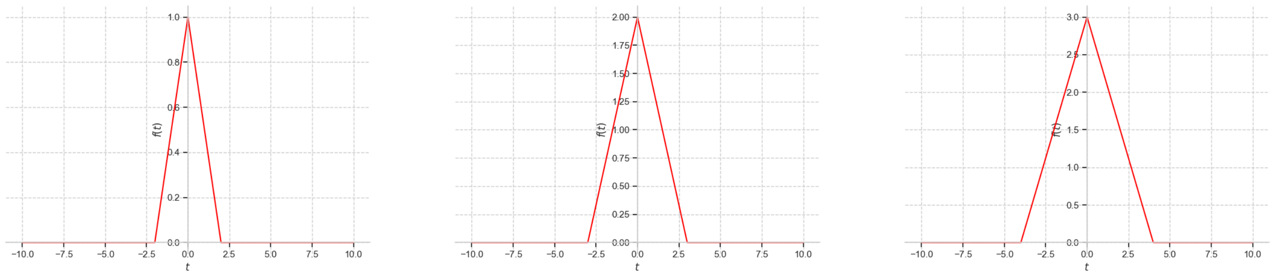
Построенные графики $\hat{f}(\omega)$ для тех же значений a и b расположены ниже

Проверим программой выполнение равенства Парсеваля

```
p_1: 1.15470053837925 ?= 1.15470053837925
p_2: 2.82842712474619 ?= 2.82842712474619
p_3: 4.89897948556636 ?= 4.89897948556636
```

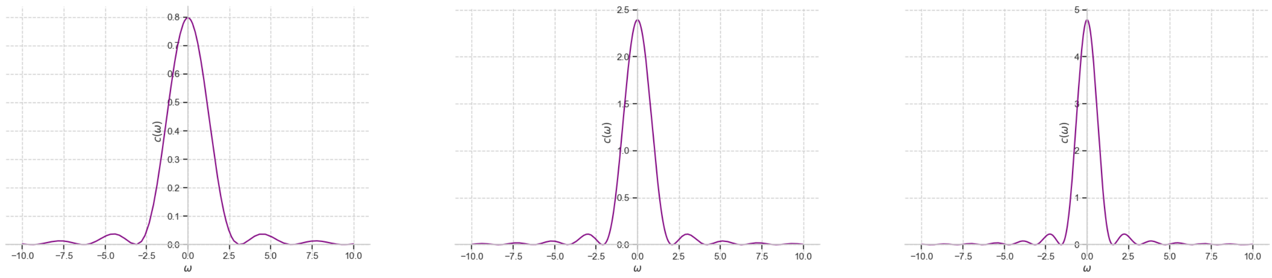
Листинг 8: Равенство Парсеваля для треугольных функций

Равенство Парсеваля выполняется для треугольной функции



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 3: Треугольные функции при различных значениях a и b



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

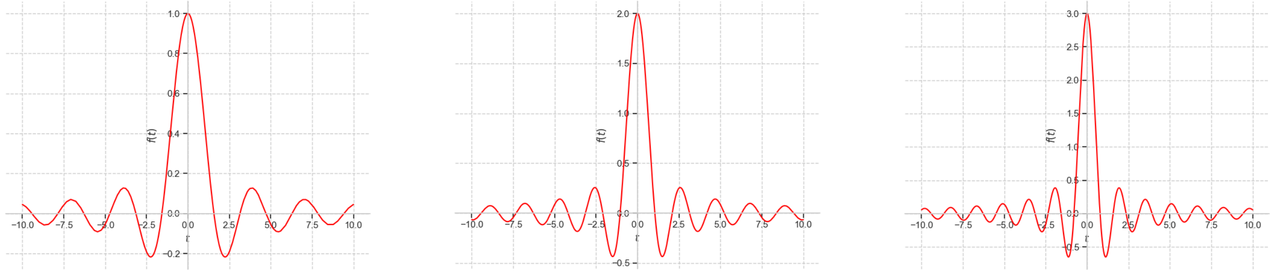
Рис. 4: Фурье-образы треугольных функций при различных значениях a и b

2.3 Кардинальный синус

Рассмотрим кардинальный синус следующего вида

$$f(t) = a \cdot \operatorname{sinc}(bt)$$

Построенные графики $f(t)$ для нескольких значений параметров $a, b > 0$ расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 5: Кардинальные синусы при различных значениях a и b

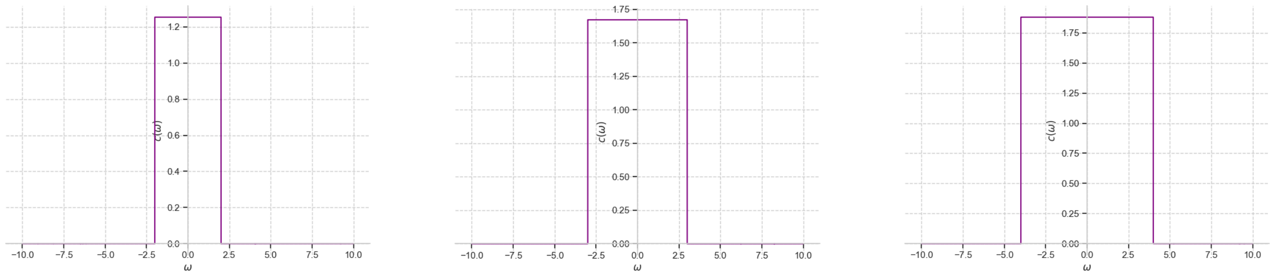
Построенные графики $\hat{f}(\omega)$ для тех же значений a и b расположены ниже

Проверим программой выполнение равенства Парсеваля

```
    p_1: 1.25331412734194 ?= 1.25331412734194
    p_2: 2.04665340503514 ?= 2.04665340503514
    p_3: 2.65868076577972 ?= 2.65868076577972
```

Листинг 9: Равенство Парсеваля для кардинальных синусов

Результат программы показывает, что равенство Парсеваля выполняется для кардинального синуса



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

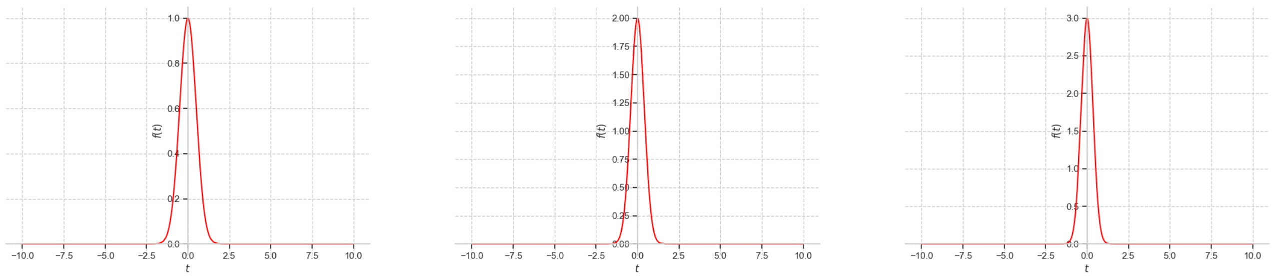
Рис. 6: Фурье-образы кардинальных синусов при различных значениях a и b

2.4 Функция Гаусса

Рассмотрим фукнцию Гаусса следующего вида

$$f(t) = ae^{-bt^2}$$

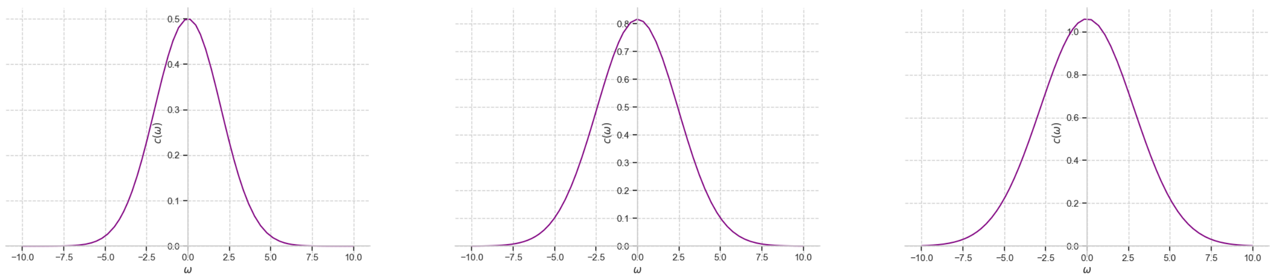
Построенные графики $f(t)$ для нескольких значений параметров $a, b > 0$ расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 7: Функции Гаусса при различных значениях a и b

Построенные графики $\hat{f}(\omega)$ для тех же значений a и b расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 8: Фурье-образы функций Гаусса при различных значениях a и b

Проверим программой выполнение равенства Парсеваля

```
    p_1: 0.941396263776715 ?= 0.941396263776715
    p_2: 1.70129510027892 ?= 1.70129510027892
    p_3: 2.37485023062924 ?= 2.37485023062924
```

Листинг 10: Равенство Парсеваля для функции Гаусса

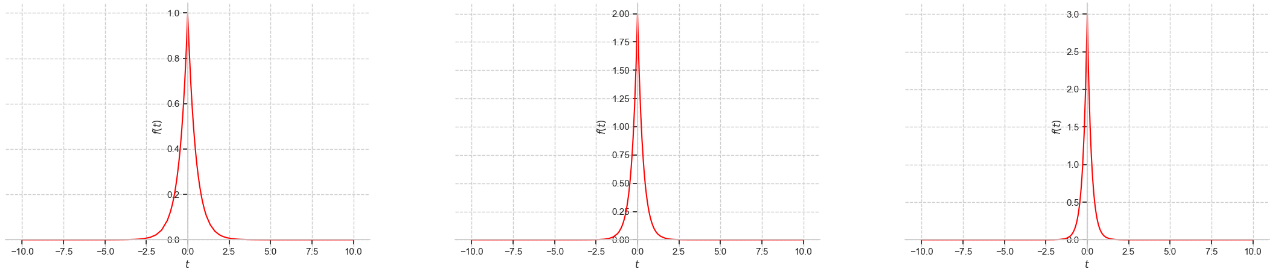
Результат говорит о выполнении равенства Парсеваля для фукнции Гаусса

2.5 Двустороннее затухание

Рассмотрим двустороннее затухание следующего вида

$$f(t) = ae^{-b|t|}$$

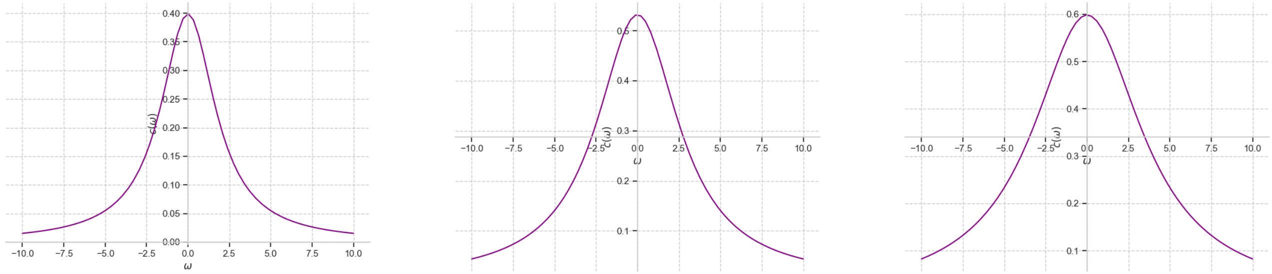
Построенные графики $f(t)$ для нескольких значений параметров $a, b > 0$ расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 9: Двусторонние затухания при различных значениях a и b

Построенные графики $\hat{f}(\omega)$ для тех же значений a и b расположены ниже



(a) $a = 1,\ b = 2$ (b) $a = 2,\ b = 3$ (c) $a = 3,\ b = 4$

Рис. 10: Фурье-образы двусторонних затуханий при различных значениях a и b

Проверим программой выполнение равенства Парсеваля

```
p_1: 0.700601299282005 ?= 0.700601299282005
p_2: 1.15326854220130 ?= 1.15326854220130
p_3: 1.49974838192519 ?= 1.49974838192519
```

Листинг 11: Равенство Парсеваля для функции Гаусса

Результат говорит о выполнении равенства Парсеваля для двустороннего затухания

3 Задание 2. Комплексное

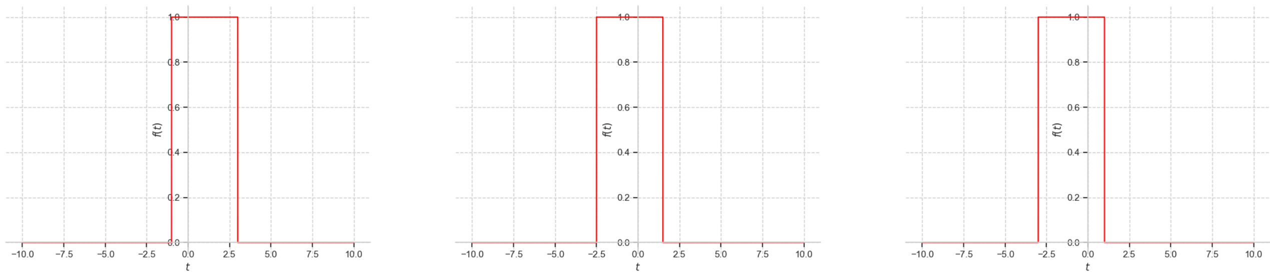
Для выполнения задания выбрана прямоугольная функция из задания 1 при зафиксированных параметрах $a = 1, b = 2$

$$g(t) = f(t+c) = \begin{cases} 1, & |t+c| \le 2, \\ 0, & |t+c| > 2. \end{cases}$$

Дополним файл static.py новой функцией

```
def shifted_rectangular_function(a, b, shift):
    if (shift == 0):
        return rectangular_function(a, b)
    return Piecewise((a, Abs(t + shift) <= b), (0, Abs(t + shift) > b))
```

Листинг 12: Добавление прямоугольной функции со смещением в файл static.py

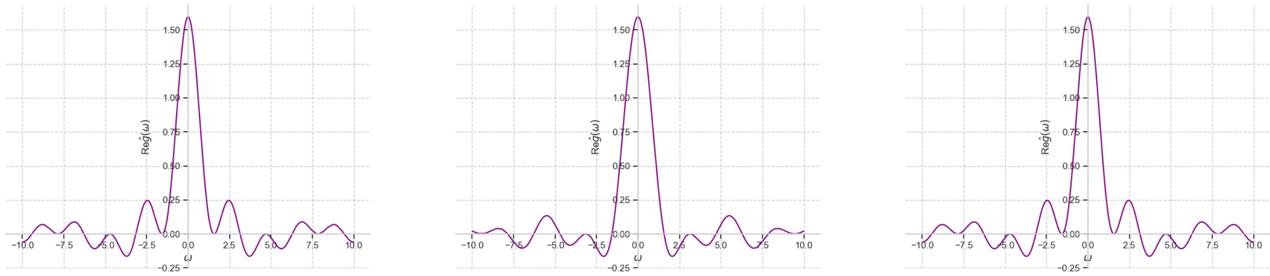


(a) $c = -1$ (b) $c = 0.5$ (c) $c = 1$

Рис. 11: Прямоугольные фукнции со смещением при различных значениях c

Построенные графики $g(t)$ для нескольких значений параметра c расположены ниже

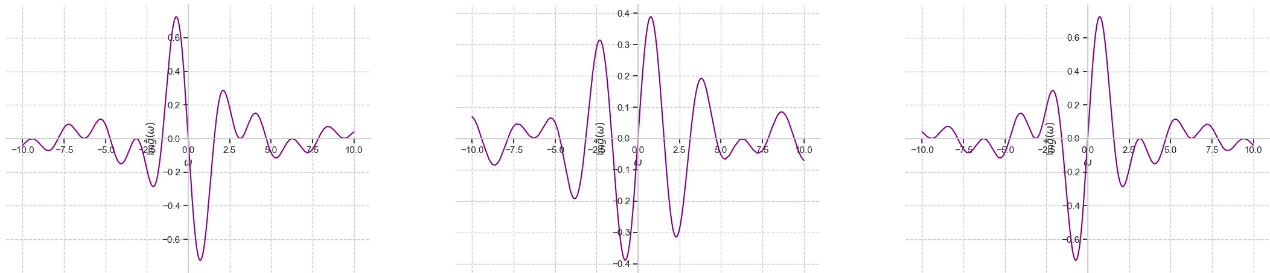
Построенные графики $\operatorname{Re}\hat{g}(\omega)$ для нескольких значений параметра c расположены ниже



(a) $c = -1$ (b) $c = 0.5$ (c) $c = 1$

Рис. 12: Действительные части Фурье-образов прямоугольных фукнций со смещением при различных значениях параметра c

Построенные графики $\operatorname{Im}\hat{g}(\omega)$ для нескольких значений параметра c расположены ниже



(a) $c = -1$ (b) $c = 0.5$ (c) $c = 1$

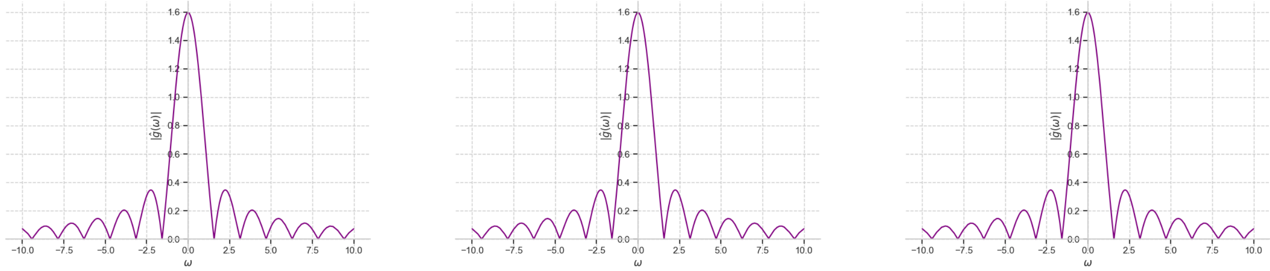
Рис. 13: Мнимые части Фурье-образов прямоугольных фукнций со смещением при различных значениях параметра c

Построенные графики $|\hat{g}(\omega)|$ для нескольких значений параметра c расположены ниже

4 Задание 3. Музыкальное

Для выполнения задания с представленного гугл-диска выбран аккорд номер 26

Для считывания аудиозаписи в список используется библиотека librosa. В переменную y запишутся значения амплитуд аудиосигнала, в переменной sr будет частота дискретизации аудиосигнала, а t – список временных отсчетов в секундах, соответствующий каждой амплитуде в списке y. На строках 15-17 показан пример использования метода. Указывается путь к mp3 файлу в переменной audio_file и в переменной select_channel хранится номер нужного звукового канала – выбран первый канал. Аудиозапись длится 4



(a) $c = -1$ (b) $c = 0.5$ (c) $c = 1$

Рис. 14: Модули Фурье-образов прямоугольных фукнций со смещением при различных значениях параметра c

секунды, последняя из которых вырезана на 19-20 строках, так как не несет в себе важной информации, аккорд заканчивается раньше

```
1  from librosa import load
2  from numpy import linspace
3
4  def get_y_sr_t(audio_file: str, select_channel: int):
5      y, sr = load(audio_file)
6
7      if select_channel >= y.ndim:
8          select_channel = 0
9
10     y = y[:, select_channel] if y.ndim > 1 else y
11     t = linspace(0, len(y) / sr, len(y))
12
13     return y, sr, t
14
15 audio_file = 'fm_lab2/chord/chord26.mp3'
16 select_channel = 0
17 y, sr, t = get_y_sr_t(audio_file, select_channel)
18
19 y = y[:3 * sr]
20 t = t[:3 * sr]
```

Листинг 13: Программа для считывания аудиозаписи в список

Программа ниже построит график аудиозаписи $f(t)$

```
import matplotlib.pyplot as plt

def build_audio_f_t(t, y, clr=None):
    if (clr != None):
        plt.plot(t, y, color=clr)
    else:
        plt.plot(t, y)

    plt.xlabel(r'$t$')
    plt.ylabel(r'$f(t)$')
    plt.grid(True)
    plt.show()

f_t_clr = colors_strs[0]
build_audio_f_t(t, y, f_t_clr)
```

Листинг 14: Программа для построения функции $f(t)$ аудиозаписи

Результат выполнения программы для считанной в список аудиозаписи

С помощью численного интегрирования trapz из библиотеки numpy найден Фурье-образ $\hat{f}(\nu)$ по частотам freqs и записан в список ampls. Пример использования расположен на 11 строке

```
1  def find_freqs_ampls(t, y, sr):
2      freqs = linspace(0, sr / 2, len(y) // 2)
3      ampls = []
4
5      for freq in freqs:
6          int = trapz(y * exp(-1j * 2 * pi * freq * t), t)
```

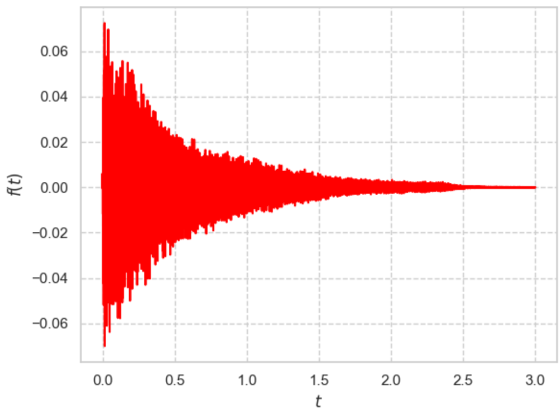


Рис. 15: График $f(t)$ аудиозаписи

```
            ampls.append(abs(int))

        return freqs, ampls

    freqs, ampls = find_freqs_ampls(t, y, sr)
```

Листинг 15: Программа для нахождения Фурье-образа аудиозаписи

Программа ниже построит график $|\hat{f}(\nu)|$. Пример использования расположен на 17-24 строках

```
1     def build_audio_f_v(freqs, ampls, start=None, stop=None, step=None, fz1=None, fz2=None,
          clr=None):
2         if ((fz1 != None) and (fz2 != None)):
3             plt.figure(figsize=(fz1, fz2))
4 
5         if (clr != None):
6             plt.plot(freqs, ampls, color=clr)
7         else:
8             plt.plot(freqs, ampls)
9 
10        plt.xlabel(r'$\nu$')
11        plt.ylabel(r'$\left|\hat{f}\left(\nu\right)\right|$')
12        plt.grid(True)
13        if ((start != None) and (stop != None and stop != 0) and (step != None and step != 0))
              :
14            plt.xticks(arange(start, stop, step))
15        plt.show()
16 
17    start = 0
18    stop = 10001
19    step = 1000
20    figsize1 = 10
21    figsize2 = 6
22    f_v_clr = colors_strs[1]
23 
24    build_audio_f_v(freqs, ampls, start=start, stop=stop, step=step,
25                    fz1=figsize1, fz2=figsize2, clr=f_v_clr)
```

Листинг 16: Программа для построения графика $|\hat{f}(\nu)|$ аудиозаписи

Построенный график $|\hat{f}(\nu)|$ расположен ниже

Так как на большом интервале частот трудно разобрать из каких нот состоит аккорд, нужен код для уменьшения интервала и шага без потери данных или искажения графиков

```
    r_start = 0
    r_end = 1000

    start_idx = next(idx for idx, freq in enumerate(freqs) if freq >= r_start)
    end_idx = next(idx for idx, freq in enumerate(freqs) if freq > r_end)

    r_ampls = ampls[start_idx:end_idx]
    r_freqs = freqs[start_idx:end_idx]
    r_step = 20
```

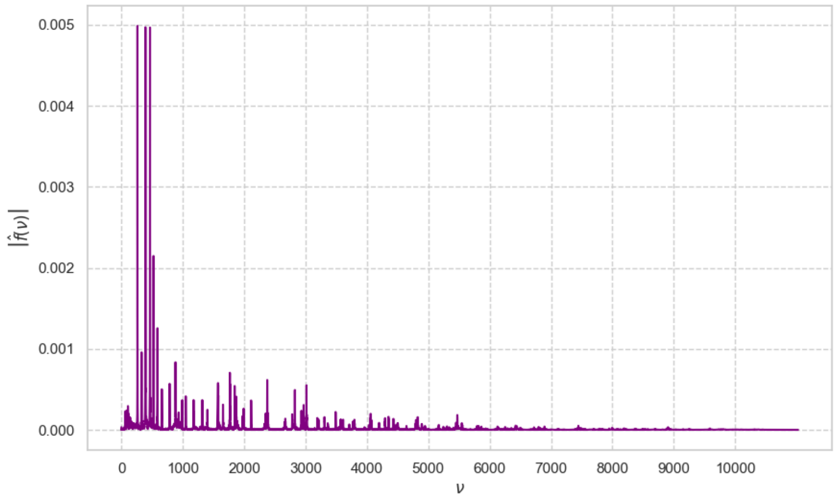


Рис. 16: График $|\hat{f}(\nu)|$ аудиозаписи

```
    build_audio_f_v(r_freqs, r_ampls, start=r_start, stop=r_end, step=r_step, fz1=figsize1,
        fz2=figsize2, clr=f_v_clr)
```

Листинг 17: Программа для изменения интервала частот и амплитуд

Построенный график $|\hat{f}(\nu)|$ на интервале частот от 0 до 1000

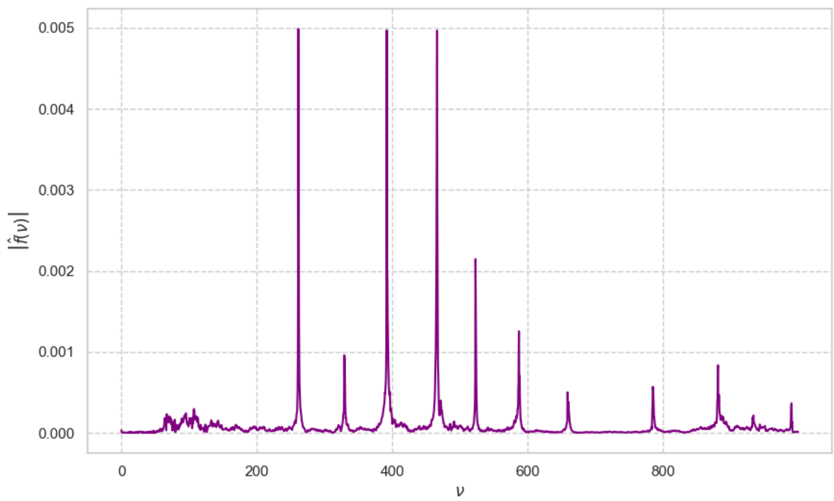


Рис. 17: График $|\hat{f}(\nu)|$ аудиозаписи на интервале частот от 0 до 1000

Для точности определения нот построен график $|\hat{f}(\nu)|$ на интервале частот от 200 до 500

Исходя из графика на уменьшенном интервале частот определены ноты: первая До, первая Соль, первая Ля диез

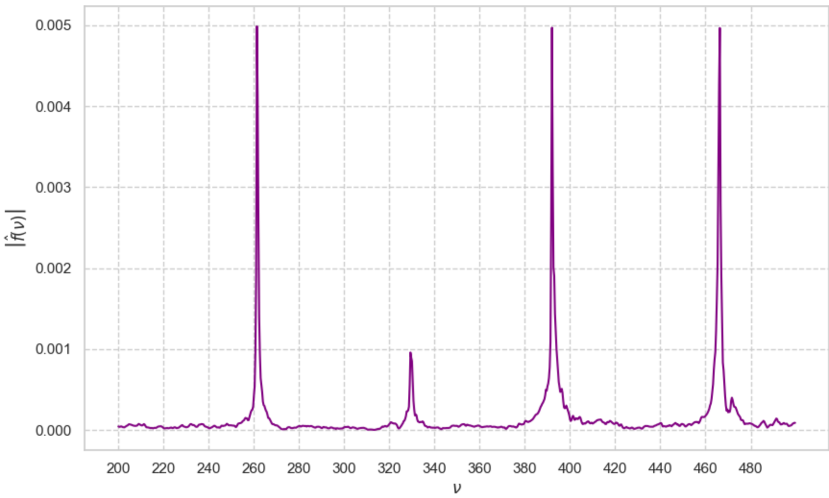


Рис. 18: График $|\hat{f}(\nu)|$ аудиозаписи на интервале частот от 200 до 500