НИУ ИТМО

Отчет по лабораторной работе №1

“Фурри ряды”

Выполнил:

Александр Иванов, R3238, ЧМ 1.2

Преподаватель:

Перегудин А. А.

Санкт-Петербург, 2024

Содержание

1. Разложение в ряд Фурье

В общем случае частичная сумма ряда Фурье выглядит следующим образом:

$$F_N(t) = \frac{a_0}{2} + \sum_{n=1}^{N} (a_n \cos(\omega_n t) + b_n \sin(\omega_n t)) \tag{1}$$

$$a_n = \frac{2}{T} \int\limits_{h}^{h+T} f(t) \cos(\omega_n t) dt \qquad b_n = \frac{2}{T} \int\limits_{h}^{h+T} f(t) \sin(\omega_n t) dt \tag{2}$$

где $\omega_n = 2\pi n/T$, h – начало промежутка, на котором проводится разложение, T – период функции.

Или, для комплексного случая:

$$G_N(t) = \sum_{n=-N}^{N} c_n e^{i\omega_n t} \tag{3}$$

$$c_n = \frac{1}{T} \int\limits_{h}^{h+T} f(t) e^{-i\omega_i t} dt \tag{4}$$

где $\omega_n = 2\pi n/T$, h – начало промежутка, на котором проводится разложение, T – период функции.

2. Вещественные функции

2.1. Квадратная волна

Рассмотрим периодическую функцию с периодом $T = 3$:

$$f_1(t) = \begin{cases} 1, & t \in [1, 2) \\ 2, & t \in [2, 4) \end{cases} \tag{5}$$

График этой функции приведен на рис. 1.

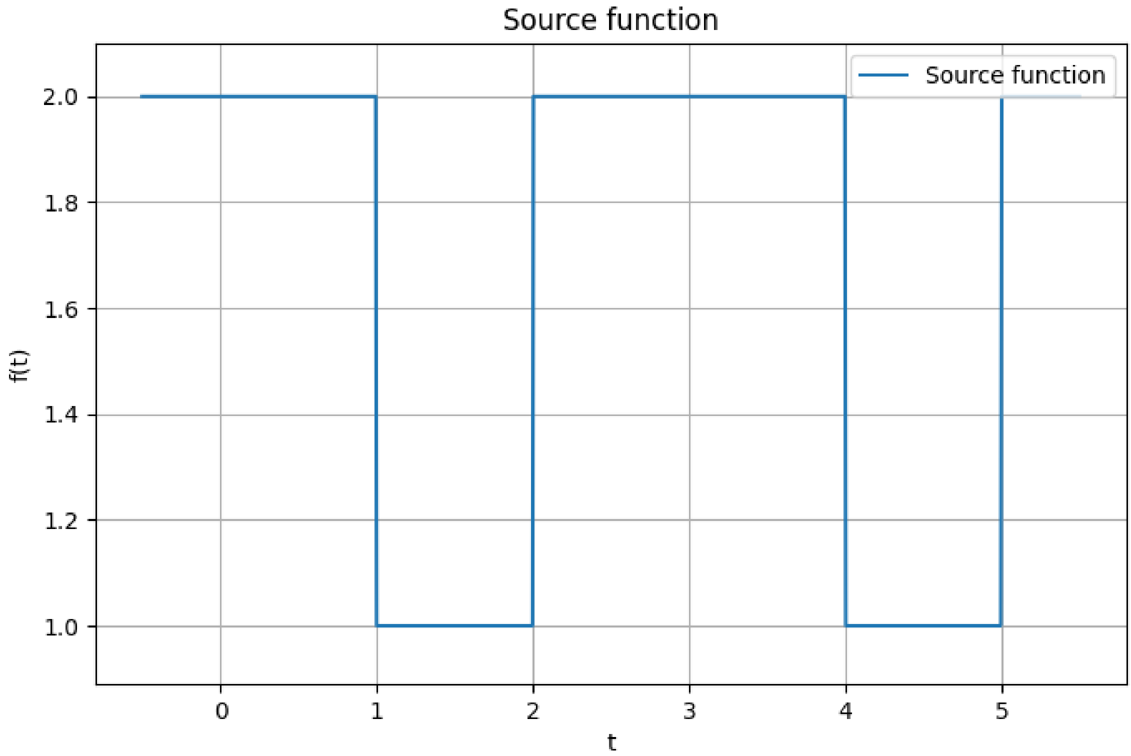


Рис. 1: График функции $f_1(t)$

2.1.1. Вычисление коэффициентов Фурье

Найдем частичные суммы ряда Фурье для этой функции:

Вычислим коэффициенты a_n, b_n и c_n для $n = 0, 1, 2$ в соответствии с формулами (2) и (4):

$$a_n = \frac{2}{3}\int\limits_1^4 f_1(t)\cos\left(\frac{2\pi n}{3}t\right)dt = \frac{2}{3}\left(\int\limits_1^2 \cos\left(\frac{2\pi n}{3}t\right)dt + \int\limits_2^4 2\cos\left(\frac{2\pi n}{3}t\right)dt\right) \tag{6}$$

$$\int \cos\left(\frac{2\pi n}{3}t\right)dt = \int \frac{3\cos(u)}{2\pi n}du = \frac{3}{2\pi n}\sin(u) = \frac{3}{2\pi n}\sin\left(\frac{2\pi n}{3}t\right)$$

$$\begin{aligned} a_n = \frac{2}{3}\left(\frac{3}{2\pi n}\sin\left(\frac{2\pi n}{3}t\right)\Big|_1^2 + \frac{3}{\pi n}\sin\left(\frac{2\pi n}{3}t\right)\Big|_2^4\right) = \\ \frac{2}{\pi n}\left(\frac{1}{2}\left(\sin\left(\frac{4\pi n}{3}\right) - \sin\left(\frac{2\pi n}{3}\right)\right) + \left(\sin\left(\frac{8\pi n}{3}\right) - \sin\left(\frac{4\pi n}{3}\right)\right)\right) = \\ \frac{1}{\pi n}\left(-\sin\left(\frac{4\pi n}{3}\right) - \sin\left(\frac{2\pi n}{3}\right) + 2\sin\left(\frac{8\pi n}{3}\right)\right) \end{aligned} \tag{7}$$

$$b_n = \frac{2}{3}\int\limits_1^4 f_1(t)\sin\left(\frac{2\pi n}{3}t\right)dt = \frac{2}{3}\left(\int\limits_1^2 \sin\left(\frac{2\pi n}{3}t\right)dt + \int\limits_2^4 2\sin\left(\frac{2\pi n}{3}t\right)dt\right) \tag{8}$$

$$\int \sin\left(\frac{2\pi n}{3}t\right)dt = \int \frac{3\sin(u)}{2\pi n}du = \frac{-3}{2\pi n}\cos(u) = \frac{-3}{2\pi n}\cos\left(\frac{2\pi n}{3}t\right)$$

$$\begin{aligned} b_n = \frac{2}{3}\left(\frac{-3}{2\pi n}\cos\left(\frac{2\pi n}{3}t\right)\Big|_1^2 + \frac{-3}{\pi n}\cos\left(\frac{2\pi n}{3}t\right)\Big|_2^4\right) = \\ \frac{-2}{\pi n}\left(\frac{1}{2}\left(\cos\left(\frac{4\pi n}{3}\right) - \cos\left(\frac{2\pi n}{3}\right)\right) + \left(\cos\left(\frac{8\pi n}{3}\right) - \cos\left(\frac{4\pi n}{3}\right)\right)\right) = \\ \frac{1}{\pi n}\left(\cos\left(\frac{4\pi n}{3}\right) + \cos\left(\frac{2\pi n}{3}\right) - 2\cos\left(\frac{8\pi n}{3}\right)\right) \end{aligned} \tag{9}$$

$$c_n = \frac{1}{3}\int\limits_1^4 f_1(t)e^{\frac{-i2\pi nt}{3}}dt = \frac{1}{3}\left(\int\limits_1^2 e^{\frac{-i2\pi nt}{3}}dt + \int\limits_2^4 2e^{\frac{-i2\pi nt}{3}}dt\right) \tag{10}$$

$$\int e^{\frac{-i2\pi nt}{3}}dt = \int e^u\frac{-3}{2\pi ni}du = \frac{-3}{2\pi ni}\int e^u du = \frac{-3}{2\pi ni}e^{\frac{-i2\pi nt}{3}}$$

$$\begin{aligned} c_n = \frac{-1}{\pi ni}\left(\frac{1}{2}e^{\frac{-i2\pi nt}{3}}\Big|_1^2 + e^{\frac{-i2\pi nt}{3}}\Big|_2^4\right) = \frac{-1}{\pi ni}\left(\frac{1}{2}\left(e^{\frac{-i4\pi n}{3}} - e^{\frac{-i2\pi n}{3}}\right) + e^{\frac{-i8\pi n}{3}} - e^{\frac{-i4\pi n}{3}}\right) \\ = -\frac{1 + e^{\frac{26\pi n}{3}} - 2e^{\frac{28\pi n}{3}}}{2\pi ne^{\frac{14\pi n}{3}}i} \end{aligned} \tag{11}$$

пупупу

После подстановки получаем следующие значения:

$$a_0 = \frac{2}{3}\int\limits_1^4 f_1(t)dt = \frac{2}{3}\left(\int\limits_1^2 dt + \int\limits_2^4 2dt\right) = \frac{2}{3}\left(t\Big|_1^2 + 2t\Big|_2^4\right) = \frac{2}{3}(1+4) = \frac{10}{3} \tag{12}$$

$$a_1 = \frac{1}{\pi}\left(-\sin\left(\frac{4\pi}{3}\right) - \sin\left(\frac{2\pi}{3}\right) + 2\sin\left(\frac{8\pi}{3}\right)\right) = \frac{\sqrt{3}}{3} \approx 0.55133 \tag{13}$$

$$a_2 = \frac{1}{2\pi}\left(-\sin\left(\frac{8\pi}{3}\right) - \sin\left(\frac{4\pi}{3}\right) + 2\sin\left(\frac{16\pi}{3}\right)\right) = \frac{-\sqrt{3}}{2\pi} \approx -0.27567 \tag{14}$$

$$b_1 = \frac{1}{\pi}\left(\cos\left(\frac{4\pi}{3}\right) + \cos\left(\frac{2\pi}{3}\right) - 2\cos\left(\frac{8\pi}{3}\right)\right) = 0 \tag{15}$$

$$b_2 = \frac{1}{2\pi}\left(\cos\left(\frac{8\pi}{3}\right) + \cos\left(\frac{4\pi}{3}\right) - 2\cos\left(\frac{16\pi}{3}\right)\right) = 0 \tag{16}$$

$$c_0 = \tag{17}$$

2.1.2. Вычисление коэффициентов Фурье с помощью программы

Воспользуемся программой для вычисления коэффициентов Фурье. Исходный код всех функций приведен в дополнении А.

```
func = np.vectorize(lambda x: 1 if 0 <= (x - 1) % 3 < 1 else 2)
a, b = fourier(func, 0, 2 * np.pi, 3)
print_fourier_coefficients(a, b)
c = fourier_exp(func, 1, 3, 3)
print_fourier_exp_coefficients(c)
```

Листинг 1: Вычисление коэффициентов Фурье

В результате выполнения программы (1) получим следующие значения (см. таблицу 1 и 2).

n	a_n	b_n
0	3.33335	0.0
1	0.55123	0.0
2	-0.2757	0.0
3	0.00020	0.0

Таблица 1: Коэффициенты Фурье для функции $f_1(t)$

n	c_n
-3	0.00010
-2	-0.13788
-1	0.27561
0	1.66677
1	0.27561
2	-0.13788
3	0.00010

Таблица 2: Коэффициенты Фурье для функции $f_1(t)$ (комплексный случай)

Полученные коэффициенты совпадают с вычисленными вручную значениями.

2.1.3. Построение графиков частичных сумм ряда Фурье

В качестве значений N выберем $N = 1, 2, 5, 15, 30$. Для каждого значения N вычислим частичную сумму ряда Фурье и построим график (см. рис. 2 и 3).

```
func = np.vectorize(lambda x: 1 if 0 <= (x - 1) % 3 < 1 else 2)
calc_and_plot(func, 1, 3, [1, 2, 5, 15, 30], './media/plots/func_1')
calc_and_plot_exp(func, 1, 3, [1, 2, 5, 15, 30], './media/plots/func_1_exp')
```

Листинг 2: Построение графиков частичных сумм ряда Фурье

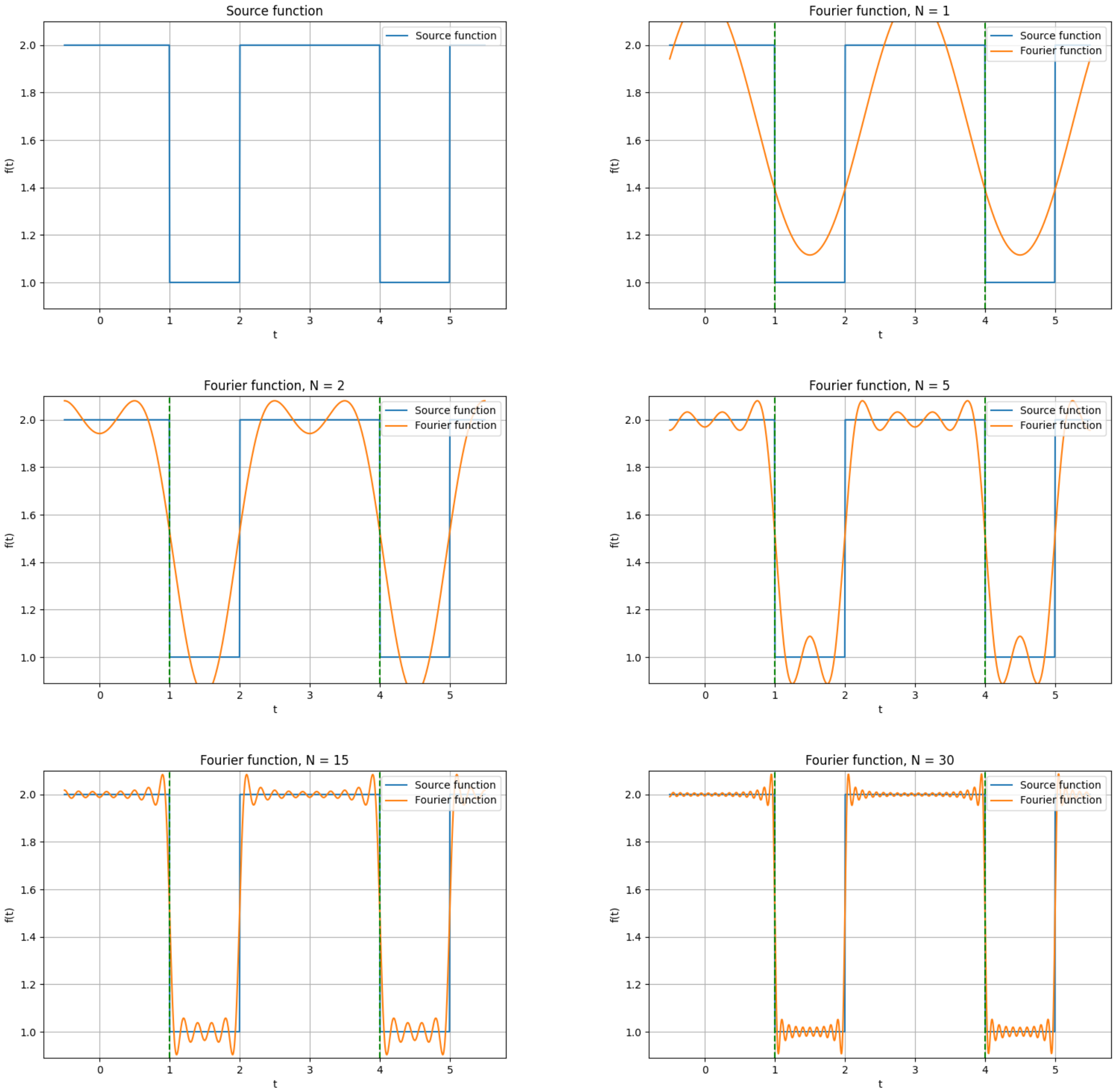


Рис. 2: График частичных сумм ряда Фурье для функции $f_1(t)$

Видим, что при увеличении N график частичной суммы ряда Фурье приближается к исходной функции. При $N = 5$ график частичной суммы ряда Фурье уже довольно хорошо приближает исходную функцию.

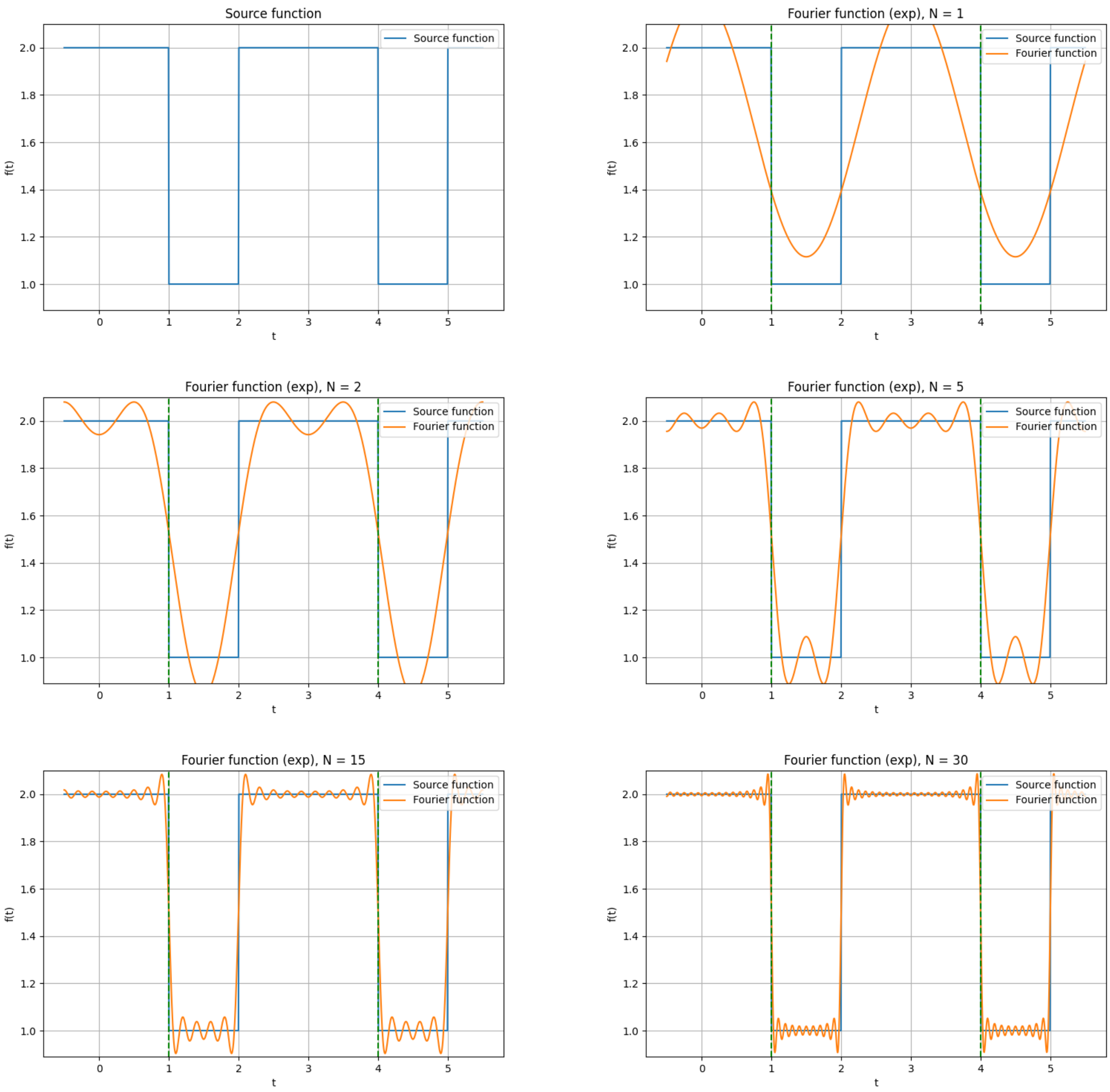


Рис. 3: График частичных сумм ряда Фурье для функции $f_1(t)$ (комплексный случай)

На графиках заметны *выбросы* в точках разрыва. Это связано с тем, что ряд Фурье сходится к функции по норме, но не обязательно в каждой точке.

2.2. Четная функция

Рассмотрим четную периодическую функцию с периодом $T = 2\pi$:

$$f_2(t) = \sin\left(\frac{5}{2}\cos(t)\right) \tag{18}$$

График этой функции приведен на рис. 4.

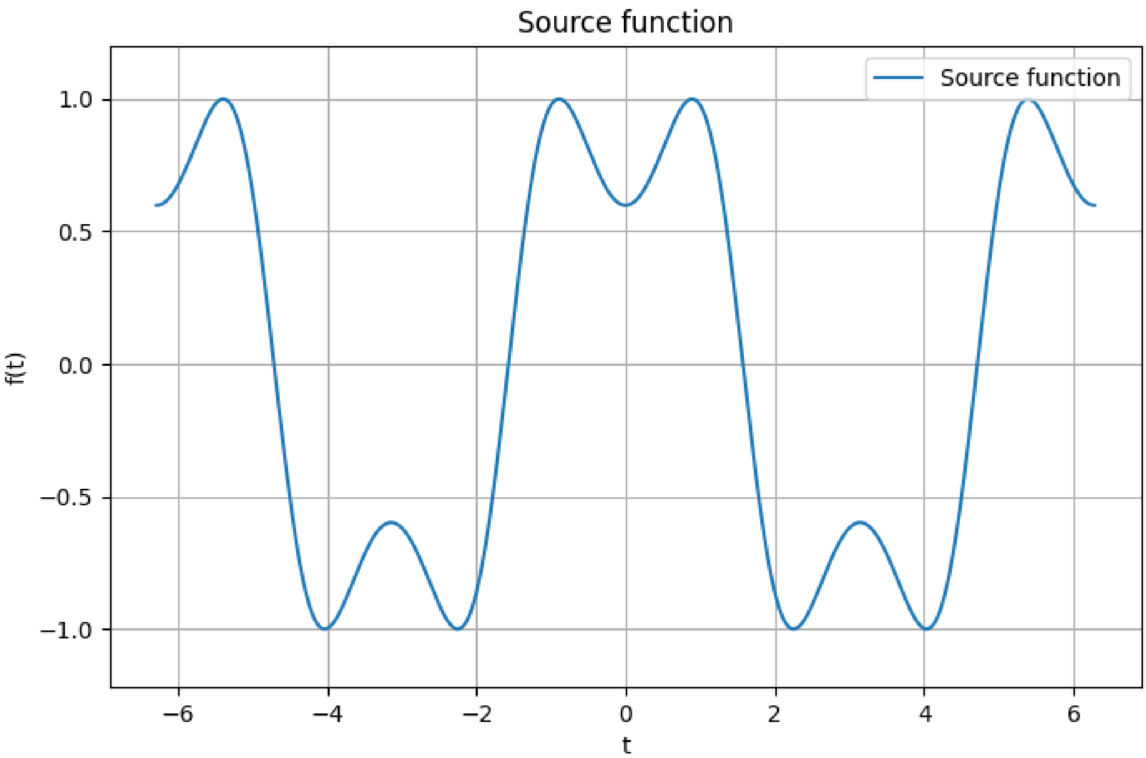


Рис. 4: График функции $f_2(t)$

2.2.1. Вычисление коэффициентов Фурье

Найдем коэффициенты ряда Фурье для этой функции:

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} \sin\left(\frac{5}{2}\cos(t)\right)\cos(nt)\,dt \tag{19}$$

$$b_n = \frac{1}{\pi} \int\limits_{-\pi}^{\pi} \sin\left(\frac{5}{2}\cos(t)\right) \sin(nt)\, dt \tag{20}$$

$$c_n = \frac{1}{2\pi} \int\limits_{-\pi}^{\pi} \sin\left(\frac{5}{2}\cos(t)\right) e^{-int} dt \tag{21}$$

Уууупс, неберущийся интеграл... Ну а кому его брать то надо? Мне не надо, я поплакался в чате :)

2.2.2. Вычисление коэффициентов Фурье с помощью программы

```
func = np.vectorize(lambda x: np.sin(5/2 * np.cos(x)))
a, b = fourier(func, -np.pi, 2*np.pi, 3)
print_fourier_coefficients(a, b)
c = fourier_exp(func, -np.pi, 2*np.pi, 3)
print_fourier_exp_coefficients(c)
```

Листинг 3: Вычисление коэффициентов Фурье

В результате выполнения программы (3) получим следующие значения (см. таблицу 3 и 4) .

n	a_n	b_n
0	0.00012	0.0
1	0.99431	0.0
2	0.00012	0.0
3	-0.43308	0.0

Таблица 3: Коэффициенты Фурье для функции $f_2(t)$

n	c_n
-3	-0.21654
-2	-0.00006
-1	0.49715
0	0.00006
1	0.49715
2	0.00006
3	-0.21654

Таблица 4: Коэффициенты Фурье для функции $f_2(t)$ (комплексный случай)

2.2.3. Построение графиков частичных сумм ряда Фурье

В качество значений N выберем $N = 1, 2, 3, 4, 5$. Для каждого значения N вычислим частичную сумму ряда Фурье и построим график (см. рис. 5 и 6).

```
func = np.vectorize(lambda x: np.sin(5/2 * np.cos(x)))
calc_and_plot(func, -np.pi, 2 * np.pi, [1, 2, 3, 4, 5],
   './media/plots/func_2')
calc_and_plot_exp(func, -np.pi, 2 * np.pi, [1, 2, 3, 4, 5],
   './media/plots/func_2_exp')
```

Листинг 4: Построение графиков частичных сумм ряда Фурье

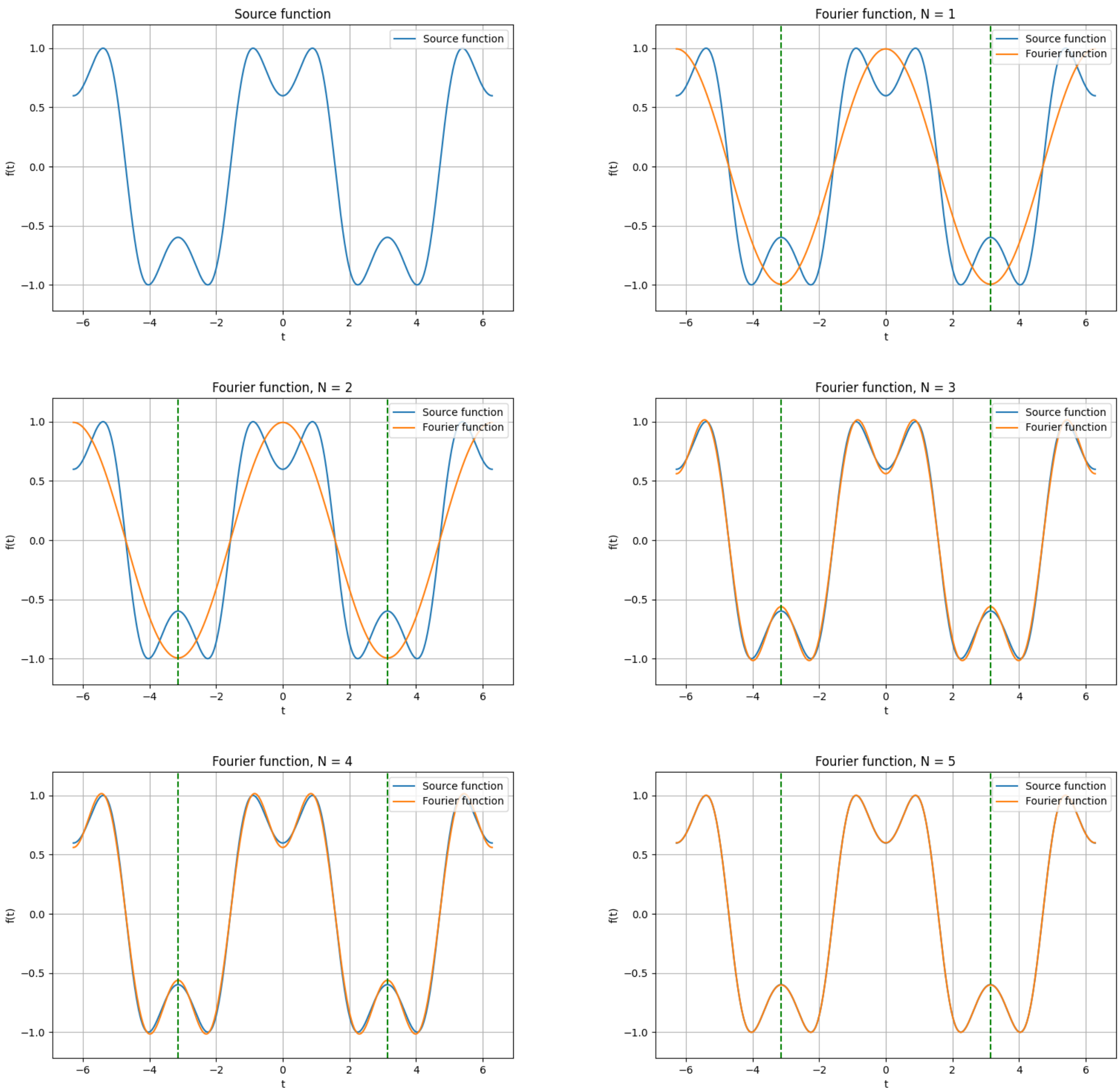


Рис. 5: График частичных сумм ряда Фурье для функции $f_2(t)$

Видим, что при увеличении N график частичной суммы ряда Фурье приближается к исходной функции. При $N = 5$ график частичной суммы ряда Фурье уже неотличим от

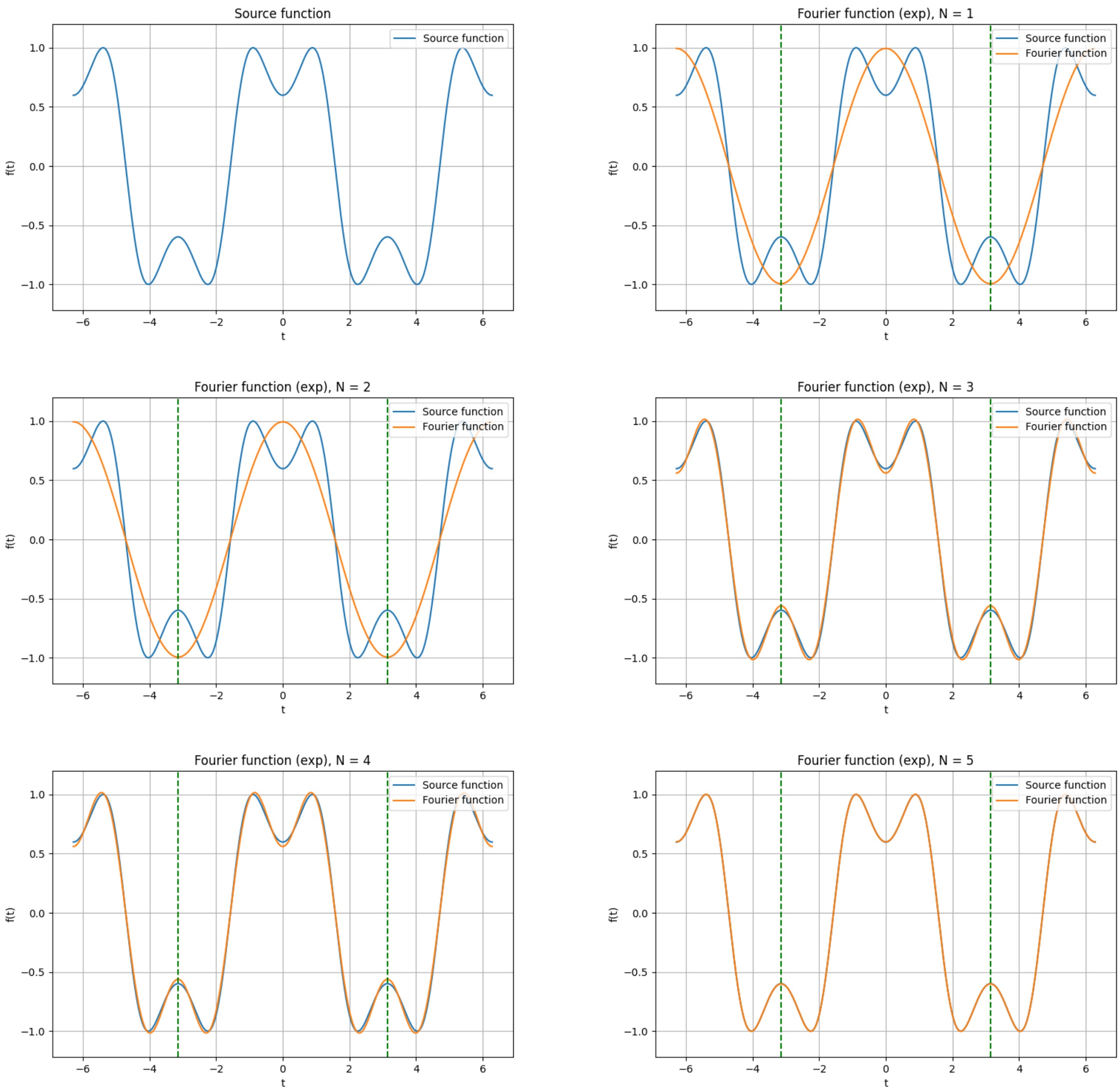


Рис. 6: График частичных сумм ряда Фурье для функции $f_2(t)$ (комплексный случай)

исходной функции.

2.3. Нечетная функция

Рассмотрим нечетную периодическую функцию с периодом $T = \pi$:

$$f_3(t) = |\cos(2t) \cdot \sin(t)| \tag{22}$$

График этой функции приведен на рис. 7.

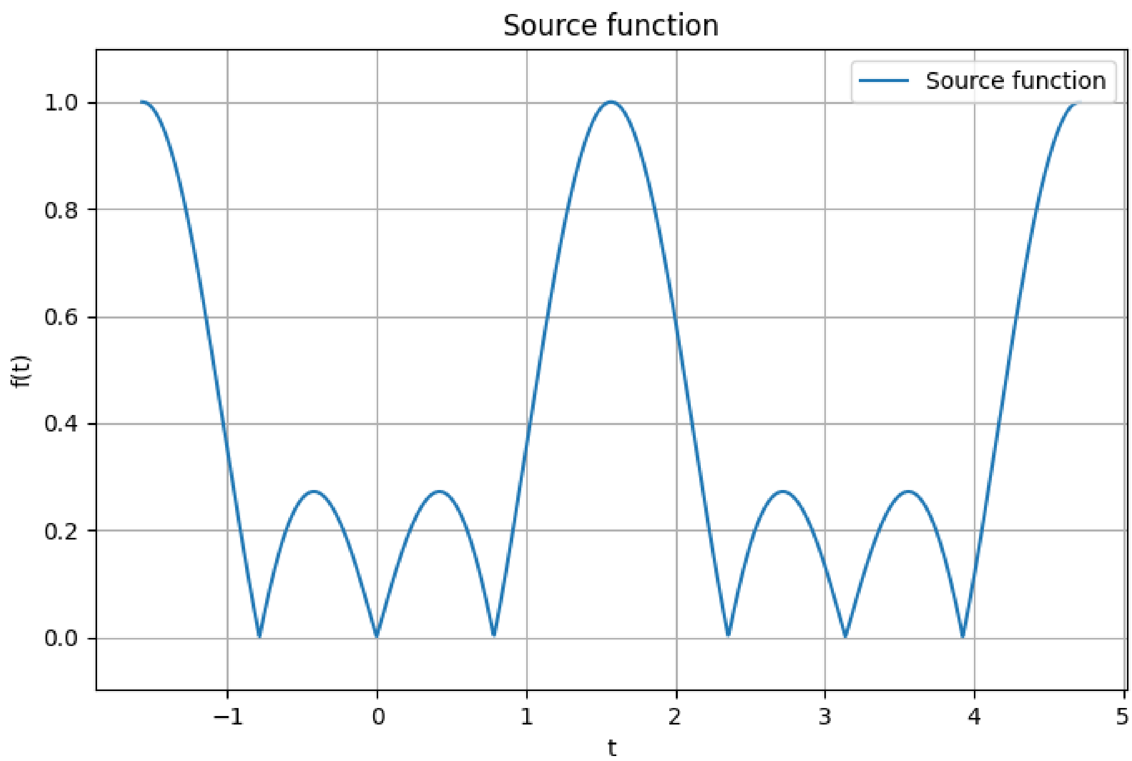


Рис. 7: График функции $f_3(t)$

2.3.1. Вычисление коэффициентов Фурье

Найдем коэффициенты ряда Фурье для этой функции:

$$a_n = \frac{2}{\pi} \int\limits_0^{\pi} |\cos(2t) \cdot \sin(t)| \cos(2nt) dt \tag{23}$$

$$b_n = \frac{2}{\pi} \int\limits_0^{\pi} |\cos(2t) \cdot \sin(t)| \sin(2nt) dt \tag{24}$$

$$c_n = \frac{1}{\pi} \int\limits_0^{\pi} |\cos(2t) \cdot \sin(t)| e^{-2int} dt \tag{25}$$

2.3.2. Вычисление коэффициентов Фурье с помощью программы

```
func = np.vectorize(lambda x: abs(np.cos(2 * x) * np.sin(x)))
a, b = fourier(func, 0, np.pi, 3)
print_fourier_coefficients(a, b)
c = fourier_exp(func, 0, np.pi, 3)
print_fourier_exp_coefficients(c)
```

Листинг 5: Вычисление коэффициентов Фурье

В результате выполнения программы (5) получим следующие значения (см. таблицу 5 и 6).

n	a_n	b_n
0	0.77601	0.0
1	-0.36616	0.0
2	0.21548	0.0
3	-0.09828	0.0

Таблица 5: Коэффициенты Фурье для функции $f_3(t)$

n	c_n
-3	-0.04914
-2	0.10774
-1	-0.18308
0	0.38800
1	-0.18308
2	0.10774
3	-0.04914

Таблица 6: Коэффициенты Фурье для функции $f_3(t)$ (комплексный случай)

2.3.3. Построение графиков частичных сумм ряда Фурье

В качестве значений N выберем $N = 1, 2, 3, 4, 5$. Для каждого значения N вычислим частичную сумму ряда Фурье и построим график (см. рис. 8 и 9).

```
func = np.vectorize(lambda x: abs(np.cos(2 * x) * np.sin(x)))
calc_and_plot(func, 0, np.pi, [1, 2, 3, 4, 5], './media/plots/func_3')
```

```
calc_and_plot_exp(func, 0, np.pi, [1, 2, 3, 4, 5],
    './media/plots/func_3_exp')
```

Листинг 6: Построение графиков частичных сумм ряда Фурье

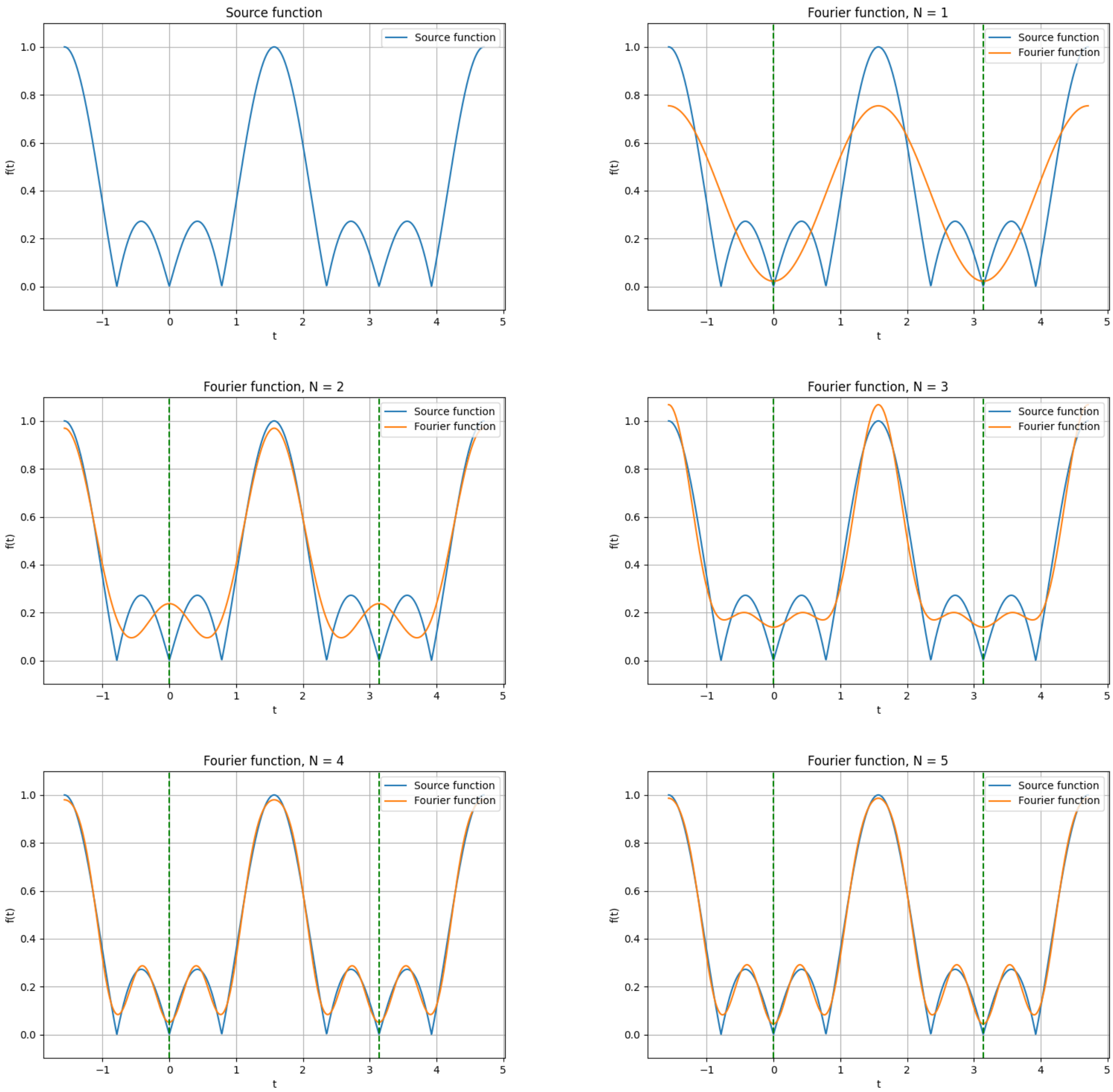


Рис. 8: График частичных сумм ряда Фурье для функции $f_3(t)$

Видим, что при увеличении N график частичной суммы ряда Фурье приближается к исходной функции, но, в отличие от четной функции, не становится неотличимым от исходной функции при $N = 5$.

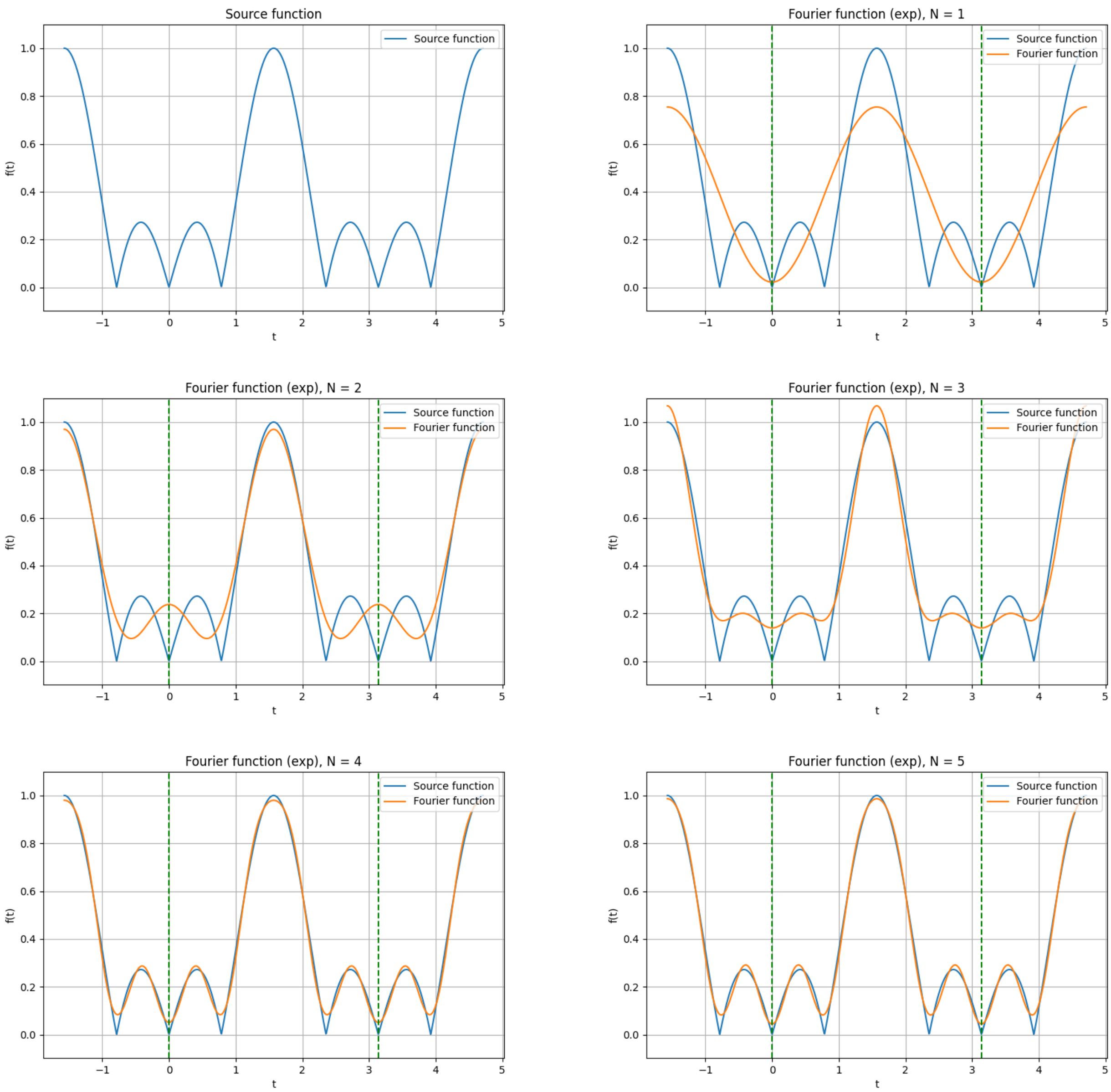


Рис. 9: График частичных сумм ряда Фурье для функции $f_3(t)$

2.4. Ни нечетная, ни четная функция

Рассмотрим периодическую функцию с периодом $T = 2\pi$:

$$f_4(t) = \sin(t)^3 - \cos(t) \tag{26}$$

График этой функции приведен на рис. 10.

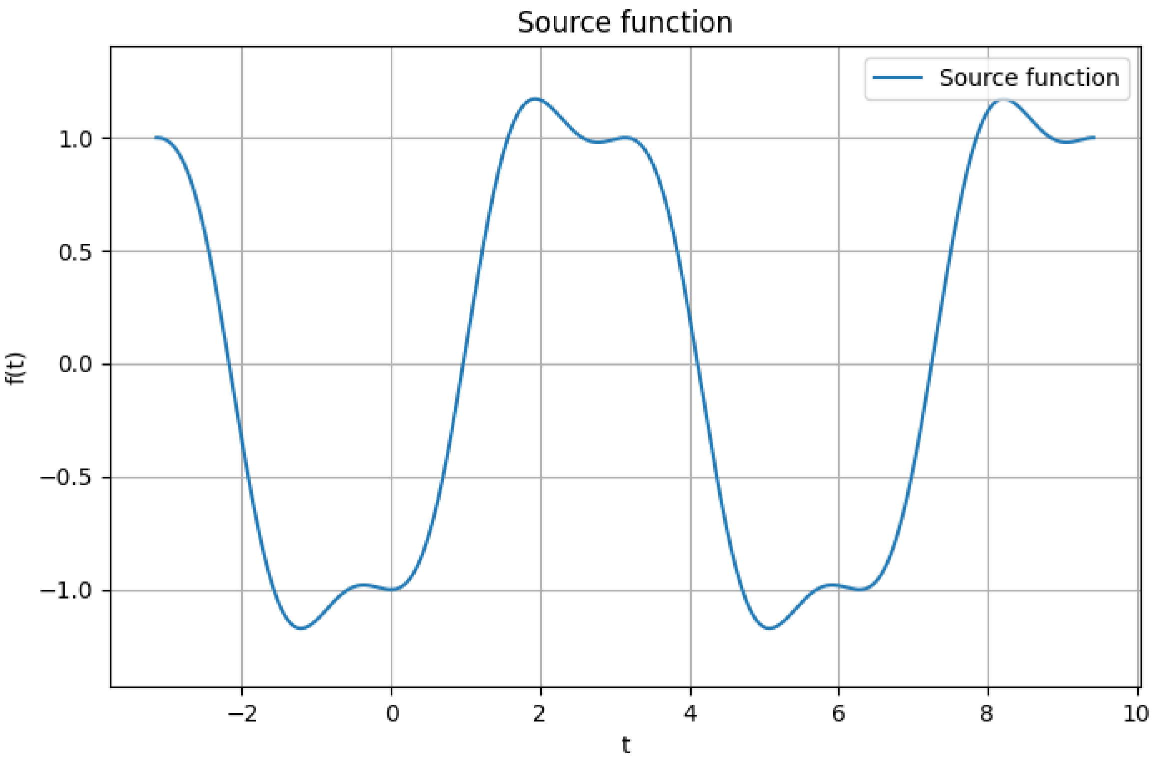


Рис. 10: График функции $f_4(t)$

2.4.1. Вычисление коэффициентов Фурье

Найдем коэффициенты ряда Фурье для этой функции:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} (\sin(t)^3 - \cos(t)) \cos(nt) dt \tag{27}$$

$$b_n = \frac{1}{\pi} \int\limits_{-\pi}^{\pi} (\sin(t)^3 - \cos(t)) \cos(nt) dt \tag{28}$$

$$c_n = \frac{1}{2\pi} \int\limits_{-\pi}^{\pi} (\sin(t)^3 - \cos(t)) e^{-int} dt \tag{29}$$

2.4.2. Вычисление коэффициентов Фурье с помощью программы

```
func = np.vectorize(lambda x: np.sin(x) ** 3 - np.cos(x))
a, b = fourier(func, 0, 2 * np.pi, 3)
print_fourier_coefficients(a, b)
c = fourier_exp(func, 0, 2 * np.pi, 3)
print_fourier_exp_coefficients(c)
```

Листинг 7: Вычисление коэффициентов Фурье

В результате выполнения программы (7) получим следующие значения (см. таблицу 7 и 8).

n	a_n	b_n
0	-0.00020	0.00000
1	-1.00020	0.75000
2	-0.00020	0.00000
3	-0.00020	-0.25000

Таблица 7: Коэффициенты Фурье для функции $f_4(t)$

n	c_n
-3	-0.00010-0.12500i
-2	-0.00010-0.00000i
-1	-0.50010+0.37500i
0	-0.00010+0.00000i
1	-0.50010-0.37500i
2	-0.00010+0.00000i
3	-0.00010+0.12500i

Таблица 8: Коэффициенты Фурье для функции $f_4(t)$ (комплексный случай)

2.4.3. Построение графиков частичных сумм ряда Фурье

В качестве значений N выберем $N = 1, 2, 3, 4, 5$. Для каждого значения N вычислим частичную сумму ряда Фурье и построим график (см. рис. 11 и 12).

```
func = np.vectorize(lambda x: np.sin(x) ** 3 - np.cos(x))
calc_and_plot(func, 0, 2 * np.pi, [1, 2, 3, 4, 5], './media/plots/func_4')
```

```
calc_and_plot_exp(func, 0, 2 * np.pi, [1, 2, 3, 4, 5],
    './media/plots/func_4_exp')
```

Листинг 8: Построение графиков частичных сумм ряда Фурье

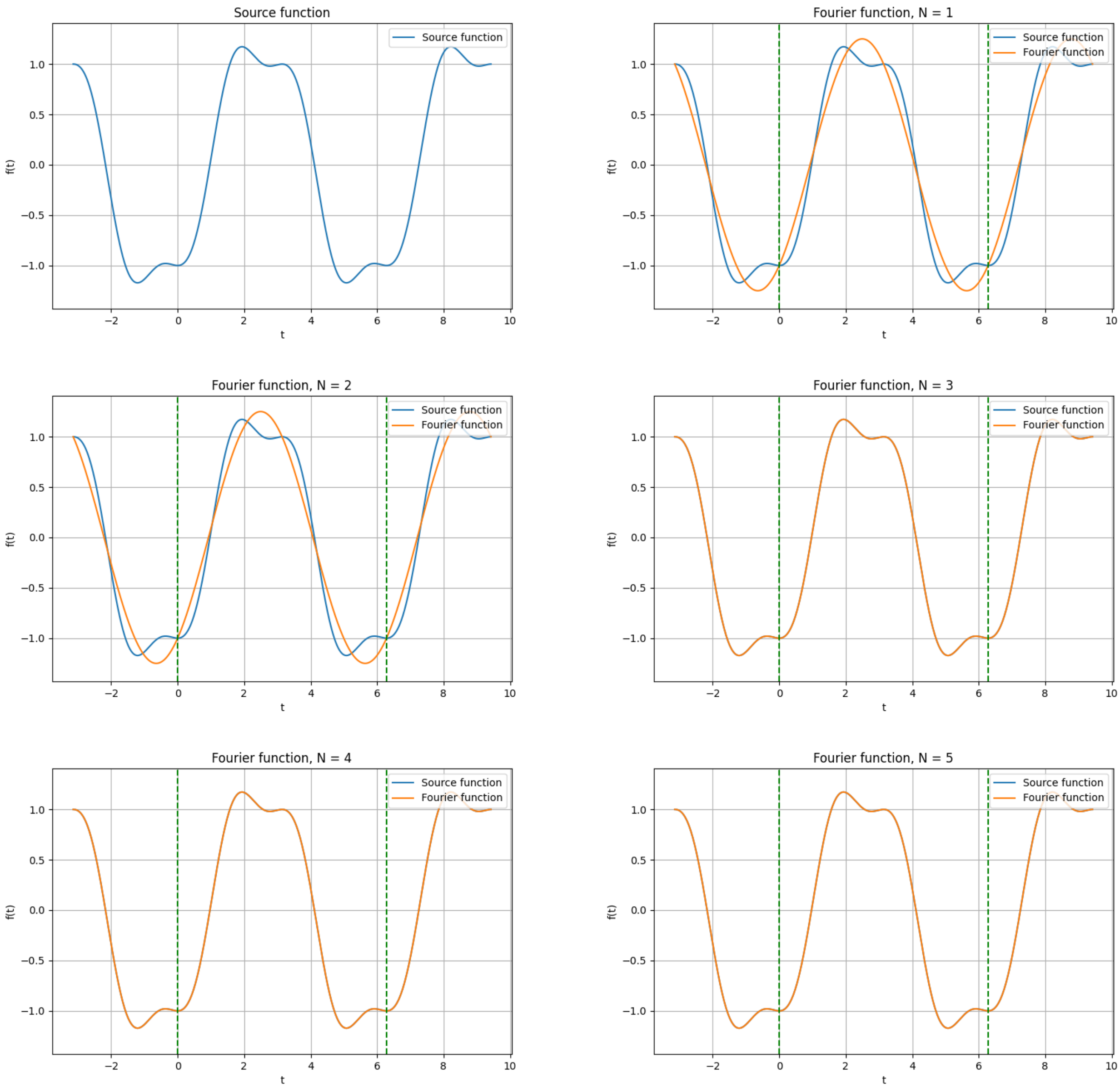


Рис. 11: График частичных сумм ряда Фурье для функции $f_4(t)$

Видим, что при увеличении N график частичной суммы ряда Фурье приближается к исходной функции, и уже при $N = 3$ график частичной суммы ряда Фурье неотличим от исходной функции.

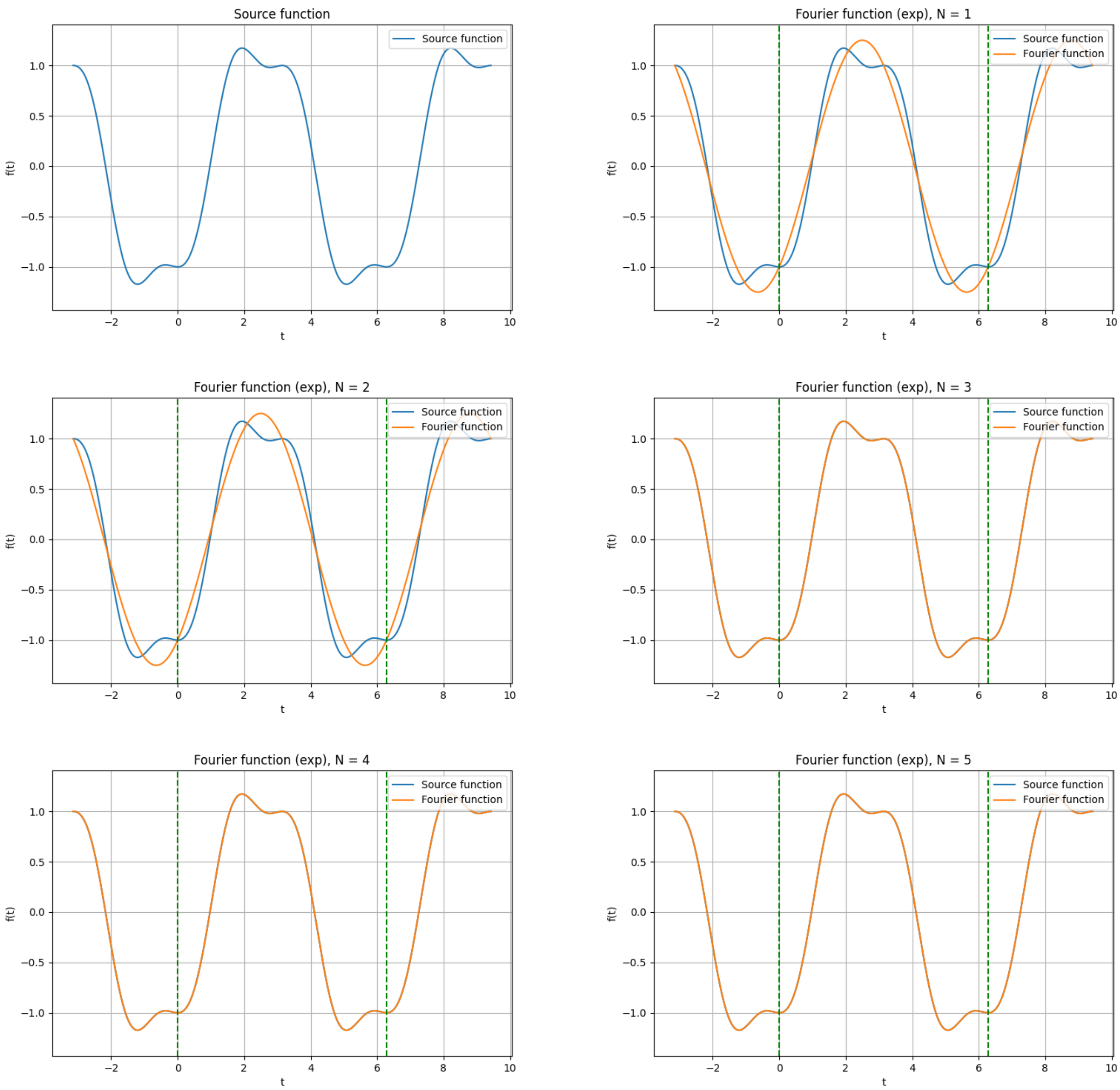


Рис. 12: График частичных сумм ряда Фурье для функции $f_4(t)$

2.5. Анализ результатов

Как и ожидалось, с увеличением количества коэффициентов график частичной суммы ряда Фурье (см. формулы 1 и 3) все точнее приближают исходную функцию. При этом, у функции, которая терпит разрывы первого рода в точках, наблюдаются *выбросы* в этих точках. Это связано с тем, что частичная сумма ряда Фурье не сходится к функции равномерно, а сходится по норме.

Графики частичных сумм ряда Фурье для вещественного и комплексного случая не отличаются.

3. Комплексная функция

3.1. Комплексный квадрат

Рассмотрим комплексную функцию $f_5(t)$:

$$Ref_5(t) = \begin{cases} 3, & t \in [-\frac{1}{2}, \frac{1}{2}) \\ 6 - 6t, & t \in [\frac{1}{2}, \frac{3}{2}) \\ -3, & t \in [\frac{3}{2}, \frac{5}{2}) \\ -18 + 6t, & t \in [\frac{5}{2}, \frac{7}{2}) \end{cases} \qquad Imf_5(t) = \begin{cases} 6t, & t \in [-\frac{1}{2}, \frac{1}{2}) \\ 3, & t \in [\frac{1}{2}, \frac{3}{2}) \\ 12 - 6t, & t \in [\frac{3}{2}, \frac{5}{2}) \\ -3, & t \in [\frac{5}{2}, \frac{7}{2}) \end{cases} \tag{30}$$

График этой функции представлен на рисунке 13.

3.1.1. Вычисление коэффициентов Фурье

Найдем коэффициенты ряда Фурье для этой функции в соответствии с формулой 4.

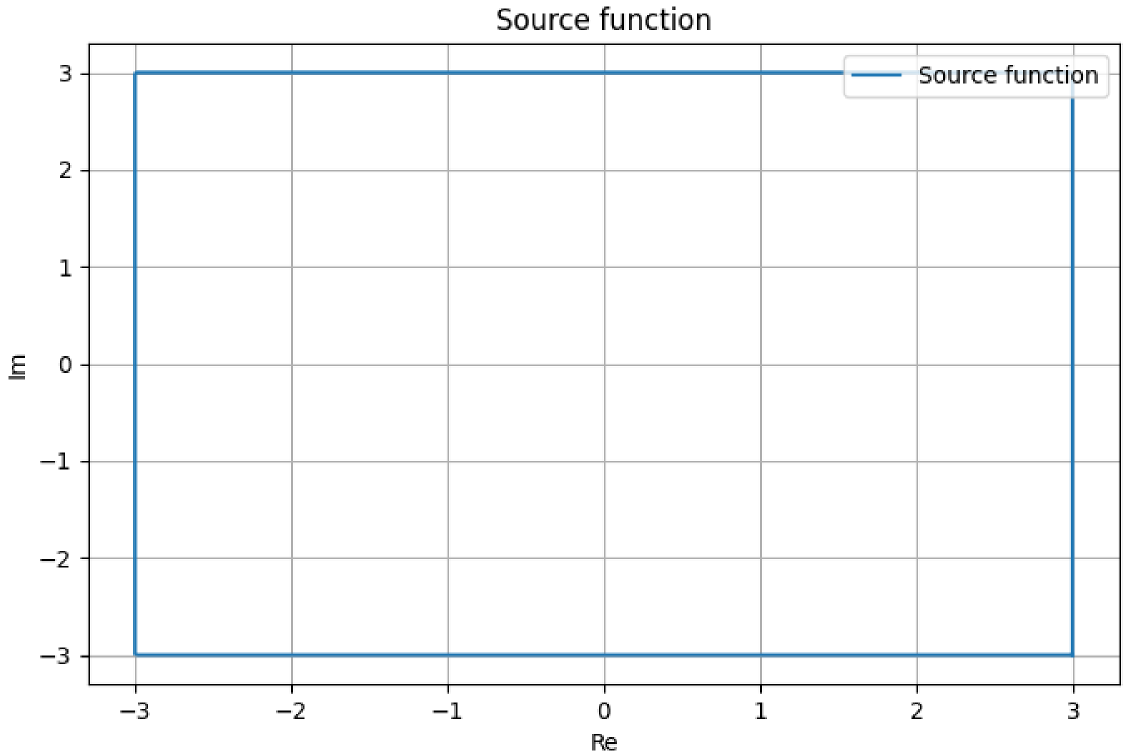


Рис. 13: График комплексной функции $f_5(t)$

$$c_n = \frac{1}{4}\int\limits_{-0.5}^{3.5} f_5(t)e^{-i\frac{\pi n t}{2}}dt = \frac{1}{4}(\int\limits_{-0.5}^{0.5}(3+6ti)e^{-i\frac{\pi n t}{2}}dt + \int\limits_{0.5}^{1.5}(6-6t+3i)e^{-i\frac{\pi n t}{2}}dt$$
$$+\int\limits_{1.5}^{2.5}(-3+(12-6t)i)e^{-i\frac{\pi n t}{2}}dt + \int\limits_{2.5}^{3.5}(-18+6t-3i)e^{-i\frac{\pi n t}{2}}dt) \quad (31)$$

3.1.2. Вычисление коэффициентов Фурье с помощью программы

```
R = 3
T = 4

def func(t):
    real = -1
    if -T/8 <= t < T/8:
        real = R
    if T/8 <= t < 3 * T / 8:
        real = 2 * R - 8 * R * t / T
    if 3 * T / 8 <= t < 5 * T / 8:
        real = -R
    if 5 * T / 8 <= t <= 7 * T / 8:
        real = -6 * R + 8 * R * t / T
```

```
    imag = -1
    if -T/8 <= t < T/8:
        imag = 8 * R * t / T
    if T/8 <= t < 3 * T / 8:
        imag = R
    if 3 * T / 8 <= t < 5 * T / 8:
        imag = 4 * R - 8 * R * t / T
    if 5 * T / 8 <= t <= 7 * T / 8:
        imag = -R

    return real + 1j * imag

func = np.vectorize(func)
c = fourier_exp(func, -T/8, T, 3)
print_fourier_exp_coefficients(c)
```

Листинг 9: Вычисление коэффициентов Фурье

В результате выполнения программы (9) получим следующие значения (см. таблицу 9).

n	c_n
-3	-0.38253+0.00000i
-2	-0.00030-0.00030i
-1	-0.00000-0.00042i
0	0.00030-0.00030i
1	3.43938+0.00000i
2	0.00030+0.00030i
3	-0.00000+0.00042i

Таблица 9: Коэффициенты Фурье для функции $f_5(t)$

3.1.3. Построение графиков частичных сумм ряда Фурье

В качестве значений N выберем $N = 1, 2, 3, 5, 10$. Для каждого значения N вычислим частичную сумму ряда Фурье и построим график (см. рис. 14).

```
    calc_and_plot_parametric(func, -T/8, T, [1, 2, 3, 5, 10],
   './media/plots/func_5')
```

Листинг 10: Построение графиков частичных сумм ряда Фурье

Видим, что как и в случае вещественных функций, при увеличении значения N график частичной суммы ряда Фурье все точнее приближает исходную функцию. При значениях $N = 1, 2$ график частичной суммы ряда Фурье представляет из себя эллипс, при $N = 3$ – уже похож на исходную функцию. При значениях $N = 10$ график частичной суммы ряда Фурье довольно точно повторяет исходную функцию.

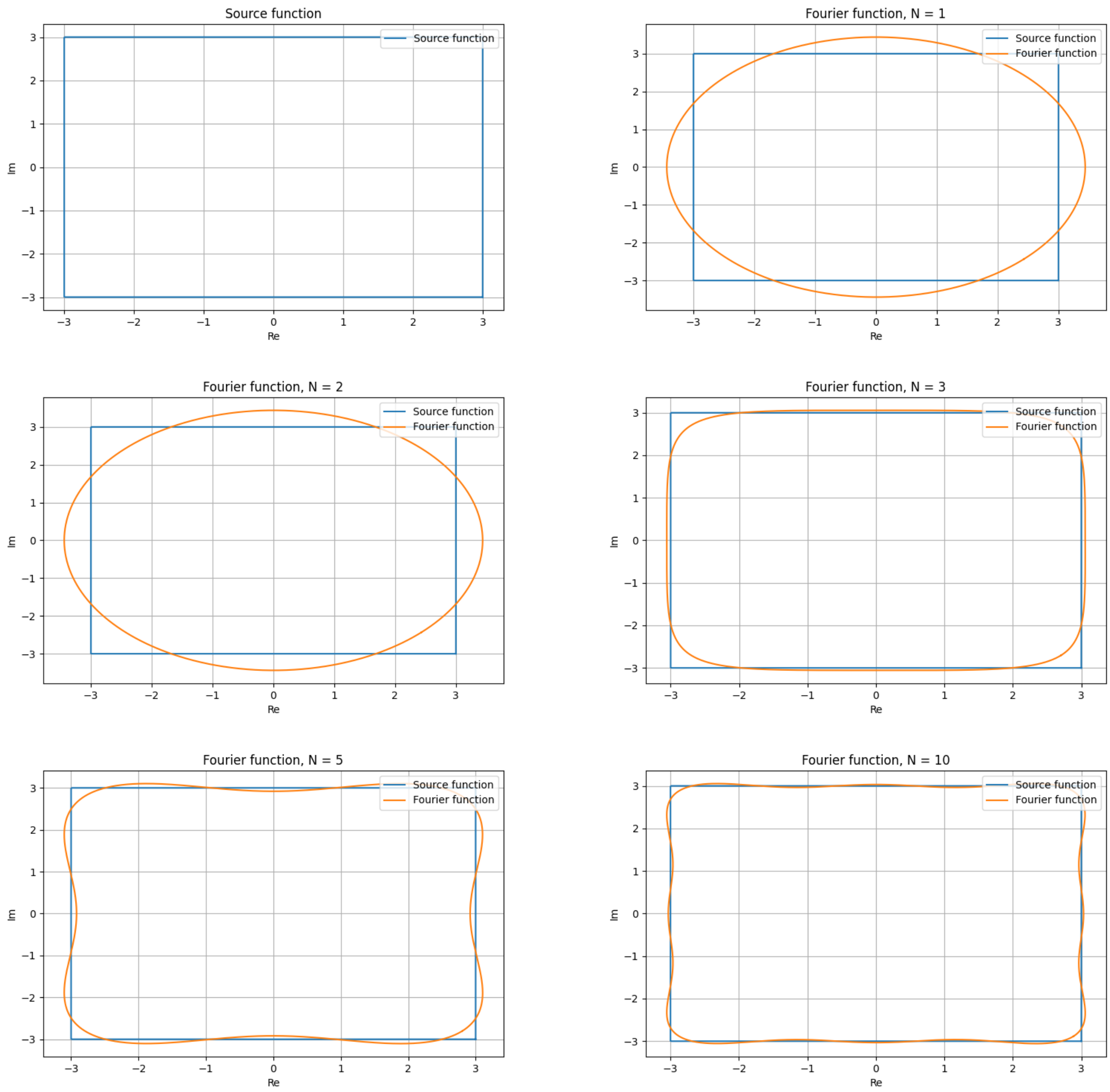


Рис. 14: Параметрический график частичных сумм ряда Фурье для функции $f_5(t)$

3.2. Графики вещественной и комплексной части отдельно

Рассмотрим графики $Re\ f_5(t)$ и $Im\ f_5(t)$ (см. рис. 15) и графики $Re\ G_n(t)$ и $Im\ G_n(t)$ для функции $f_5(t)$ (см. рис. 16 - 20).

Тут так же заметно, что с увеличением значения N график частичной суммы ряда Фурье все точнее приближает исходную функцию на выбранном промежутке.

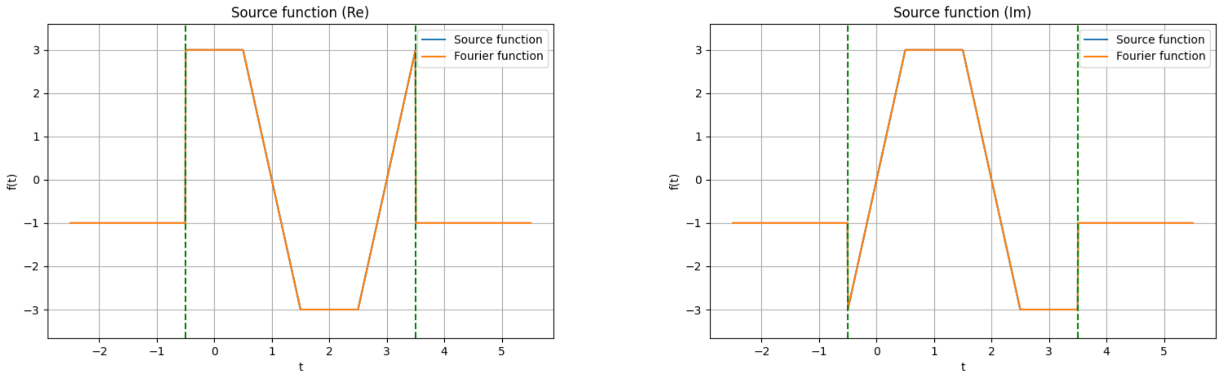


Рис. 15: График $f_5(t)$ (действительная и мнимая часть)

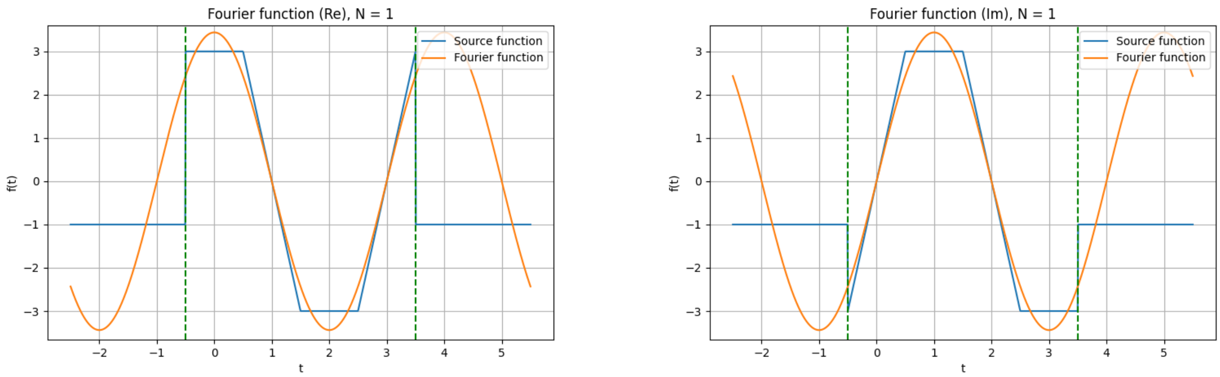


Рис. 16: График частичных сумм ряда Фурье для функции $f_5(t)$ $(N = 1)$ (действительная и мнимая часть)

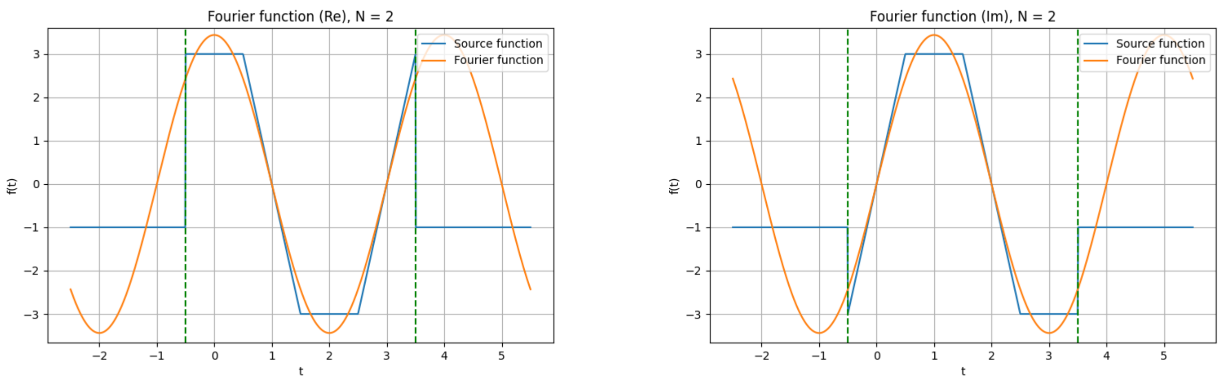


Рис. 17: График частичных сумм ряда Фурье для функции $f_5(t)$ $(N = 2)$ (действительная и мнимая часть)

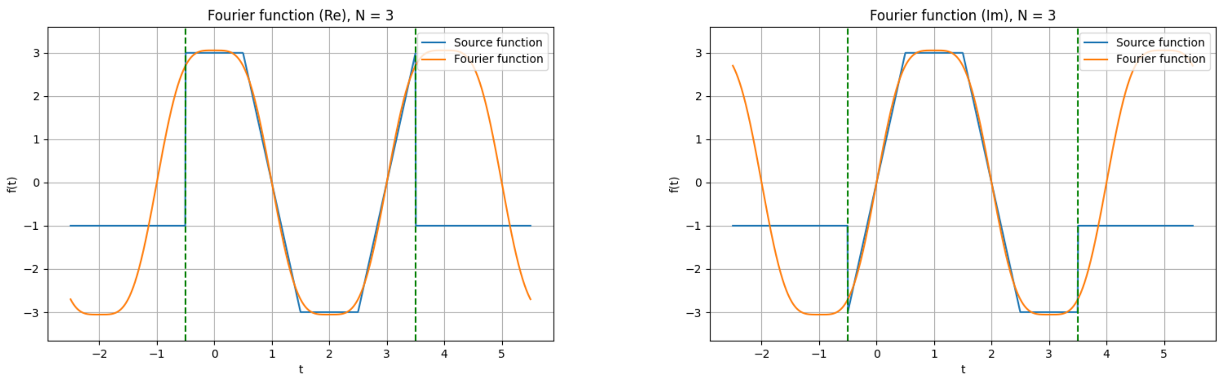


Рис. 18: График частичных сумм ряда Фурье для функции $f_5(t)$ $(N = 3)$ (действительная и мнимая часть)

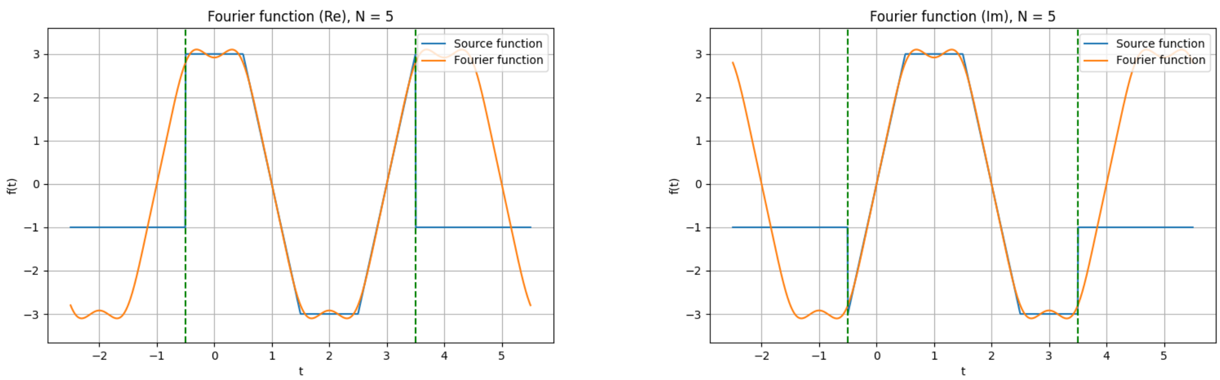


Рис. 19: График частичных сумм ряда Фурье для функции $f_5(t)$ $(N = 5)$ (действительная и мнимая часть)

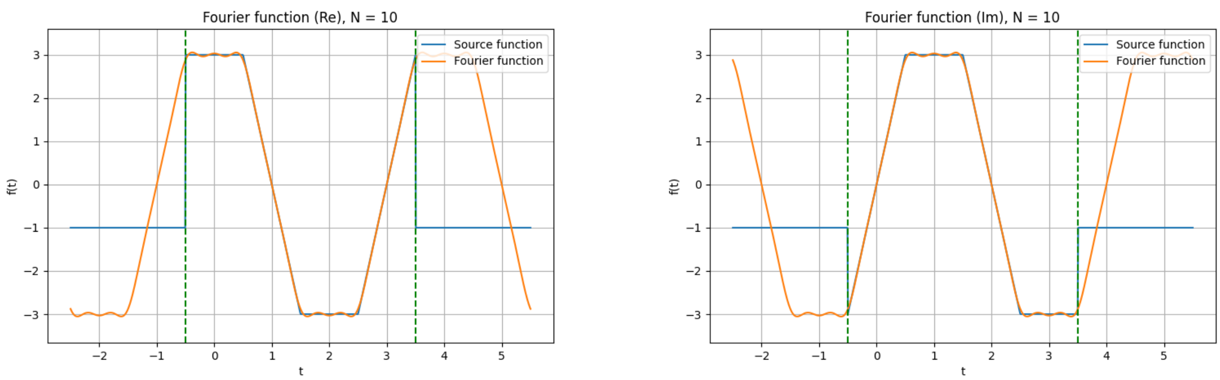


Рис. 20: График частичных сумм ряда Фурье для функции $f_5(t)$ $(N = 10)$ (действительная и мнимая часть)

4. Дополнительное задание

4.1. Разложение рисунка

По аналогии с разложением комплексной параметрической функции, можно найти разложение в ряд Фурье для функции, график которой представляет из себя интересующий нас рисунок.

Нам подойдет рисунок в векторном формате, который состоит из одной замкнутой линии. Она не должна иметь толщины. В редакторе CoralDraw для этого существует параметр *сверхтонкий абрис*.

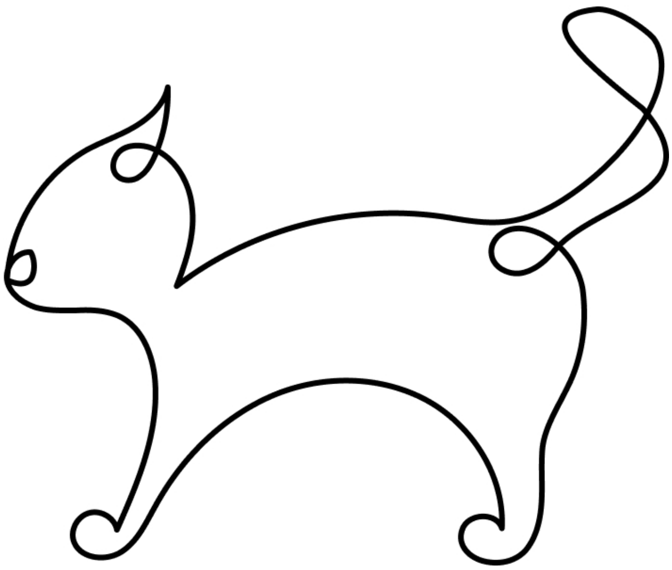


Рис. 21: Рисунок, который будет раскладывать

Возьмем рисунок с котиком. Для того, чтобы *достать* параметрический график из svg файла можно использовать библиотеку `svg.path` и ее функцию `parse_path`.

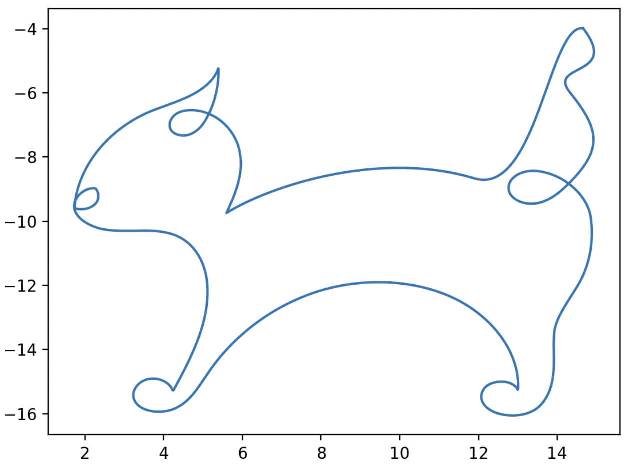
```
path = svg.path.parse_path(spline)
path_func = np.vectorize(lambda t: path.point(t) * scale)
```

Листинг 11: Получение параметрической функции рисунка

В результате работы кода (см. листинг 11) получаем параметрическую функцию $path\_func(t)$, где $t \in [0, 1]$, которая будет возвращать точку на комплексной плоскости.

Теперь данную функцию можно разложить в ряд Фурье на отрезке $[0, 1]$ с помощью функции `fourier_exp` (см. листинг 30). В результате работы этой функции получим набор коэффициентов c_n, которые являются коэффициентами искомого разложения рисунка.

Для проверки полученного результата можно построить *графики* полученных функций (количество коэффициентов $N = 20$) (см рис. 22). Видим, что оба графика крайне похожи друг на друга. Это значит, что разложение Фурье корректно применилось для нашего рисунка.



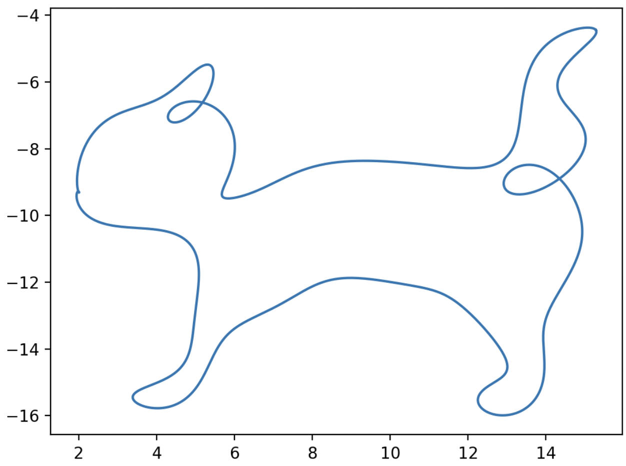


Рис. 22: Исходный рисунок и рисунок после разложения Фурье ($N = 20$)

4.2. Вращаем векторы

Теперь разберемся с тем, как можно представить полученные коэффициенты разложения с виде движущихся с целой частотой относительно друг друга векторов.

Из разложения (см. формулу 3) мы имеем слагаемые вида:

$$c_i \cdot e^{i\omega_i t}$$

Каждый из коэффициентов c_i имеет комплексное значение, при этом i определяет частоту вращения вектора (об/сек, если $i < 0$, то вращение в другую сторону).

Для определения радиуса круга с номером (и частотой вращения) i можно воспользоваться следующей формулой:

$$r = \sqrt{Re(c_i)^2 + Im(c_i)^2} \tag{32}$$

Это будет радиус окружности, внутри которой вращается вектор.

Начальное положение вектора будем задавать как часть дуги (от 0 до 1), на которую должен *указывать* вектор. Обозначим это значение ϕ Найти его можно следующим образом:

$$\phi = \begin{cases} \frac{atan2(-Im(c_i),\ Re(c_i))}{2\pi} & atan2(Im(c_i), Re(c_i)) > 0 \\ \frac{atan2(-Im(c_i),\ Re(c_i))+2\pi}{2\pi} & atan2(Im(c_i), Re(c_i)) <= 0 \end{cases} \tag{33}$$

После мнимой части стоит минус для того, чтобы рисунок не был перевернутым.

Теперь, когда у нас есть все нужные значения (частоты вращения векторов, длины векторов, начальные положения), мы можем перейти к отрисовке.

4.3. Отрисовка графики

Основной единицей в программе являются объект класса `RotatingCircle` (см. листинг 34), который наследован от класса `Circle` – вращающаяся окружность. В нем добавлены поля, хранящие окружность, по орбите которой она вращается, частота вращения и позиции *точки* на этой окружности.

Кроме того, добавлены классы `VectorInCircle` и `DotOnCircle`, наследованные от классов `Arrow` и `Dot` соответственно. Они нужны лишь для визуализации соответствующих элементов.

Все *действие* основано на обновлении точек на окружностях, и, соответственно, центов окружностей, которые привязных к этим точках.

Функция `rotating_circle_updater` изменяет положение точки на окружности, `dot_on_circle_updater` – перемещает точку по окружности, `vector_in_circle_updater` – перемещает начало вектора в центр окружности, в которой он расположен и конец вектора к точке на этой же окружности.

Таким образом, изменение положения центра окружности и положения точки на ней ведет к изменений положения всех элементов.

В основном методе рассчитываются коэффициенты c_i и создаются *вложенные* окружности (родителем каждой следующей окружности является предыдущая), центр которых привязывается к точке на родительской окружности. Таким образом создается зависимость положения дочерних окружностей от родительских.

В результате получаем анимацию с отрисовкой исходного рисунка (ссылка):

Так же можно посмотреть на анимации при различных значениях N (ссылка).

Исходный код находится в приложении B.

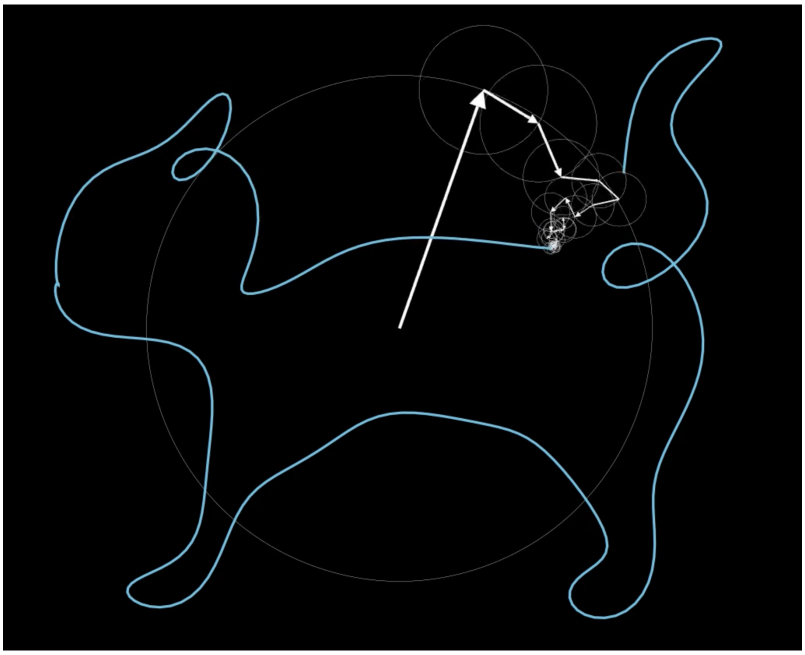


Рис. 23: Стоп-кадр из анимации

Полученный рисунок полностью совпадает с тем, что был получен из частичной сумму ряда Фурье (см. рис. 22).

A Исходный код

```
def scalar_product(f, g, a, b):
    x = np.linspace(a, b, 10000)
    dx = x[1] - x[0]
    return np.dot(f(x), g(x)) * dx
```

Листинг 12: Функция для вычисления скалярного произведения функций

Аргументы функции `scalar_product`: `f` и `g` – функции, для которых вычисляется скалярное произведение, `a` и `b` – границы интегрирования.

```
def get_sincosmt(T):
    sinmt = lambda x: np.sin(2 * np.pi * x[0] / T * x[1])
    cosmt = lambda x: np.cos(2 * np.pi * x[0] / T * x[1])

    return sinmt, cosmt
```

Листинг 13: Функция для получения вспомогательных функций

Аргументы функции `get_sincosmt`: `T` – период функции.

```
def get_expmt(T):
    return lambda x: np.e ** (1j * 2 * np.pi * x[0] / T * x[1])
```

Листинг 14: Функция для получения вспомогательных функций

Аргументы функции `get_expmt`: `T` – период функции.

```
def fourier(func, t0, T, N):
    sinmt, cosmt = get_sincosmt(T)
    a = []
    b = []
    for i in range(N + 1):
        sin = lambda x: sinmt([i, x])
        cos = lambda x: cosmt([i, x])

        a.append(scalar_product(func, cos, t0, t0 + T) * 2 / T)
        b.append(scalar_product(func, sin, t0, t0 + T) * 2 / T)
    return a, b
```

Листинг 15: Функция для вычисления коэффициентов Фурье

Аргументы функции `fourier`: `func` – функция, для которой вычисляются коэффициенты, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов.

```
def fourier_exp(func, t0, T, N):
    expmt = get_expmt(T)
    c = []
    for i in range(-N, N + 1):
        exp = lambda x: expmt([-i, x])
        c.append(scalar_product(func, exp, t0, t0 + T) / T)

    return c
```

Листинг 16: Функция для вычисления коэффициентов Фурье

Аргументы функции `fourier_exp`: `func` – функция, для которой вычисляются коэффициенты, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов.

```
def fourier_func(a, b, T, N):
    sinmt, cosmt = get_sincosmt(T)
    return lambda x: a[0] / 2 + sum([a[i] * cosmt([i, x]) + b[i] *
   sinmt([i, x]) for i in range(1, N + 1)])
```

Листинг 17: Получение функции частичной суммы ряда Фурье до N

Аргументы функции `fourier_func`: `a` и `b` – коэффициенты Фурье, `T` – период функции, `N` – количество коэффициентов.

```
def fourier_exp_func(c, T, N):
    expmt = get_expmt(T)
    return lambda x: sum([c[i + len(c) // 2] * expmt([i, x]) for i in
   range(-N, N + 1)])
```

Листинг 18: Получение функции частичной суммы ряда Фурье до N

Аргументы функции `fourier_exp_func`: `c` – коэффициенты Фурье, `T` – период функции, `N` – количество коэффициентов.

```
def fourierise(func, t0, T, N):
    a, b = fourier(func, t0, T, N)
    return fourier_func(a, b, T, N)
```

Листинг 19: Функция для получения разложения Фурье функции

Аргументы функции `fourierise`: `func` – функция, для которой проводится разложение, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов.

```
def fourierise_exp(func, t0, T, N):
    c = fourier_exp(func, t0, T, N)
    return fourier_exp_func(c, T, N)
```

Листинг 20: Функция для получения разложения Фурье функции

Аргументы функции `fourierise_exp`: `func` – функция, для которой проводится разложение, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов.

```
def plot_func(func, fourier_func, t0, T, caption = '', title = ''):
    # limits
    x_min = t0 - T * 0.5
    x_max = t0 + 1.5 * T
    ymin = min(func(np.linspace(x_min, x_max, 100)))
    ymax = max(func(np.linspace(x_min, x_max, 100)))

    ymax = ymax + 0.1 * (ymax - ymin)
    ymin = ymin - 0.1 * (ymax - ymin)

    x = np.linspace(x_min, x_max, 1000)
    plt.figure(figsize=(8, 5))

    plt.ylim(ymin, ymax)
    plt.plot(x, func(x))
    if fourier_func != func:
        plt.plot(x, fourier_func(x))
        plt.plot([t0, t0], [ymin, ymax], 'g--')
        plt.plot([t0 + T, t0 + T], [ymin, ymax], 'g--')
    # add labels
    plt.xlabel('t')
    plt.ylabel('f(t)')
    if fourier_func != func:
        plt.legend(['Source function', 'Fourier function'], loc='upper
   right')
    else:
        plt.legend(['Source function'], loc='upper right')
    # add caption
    plt.title(caption)
    plt.grid()
    plt.savefig(title + '.png')
```

Листинг 21: Функция для отрисовки графиков

Аргументы функции `plot_func`: `func` – исходная функция, `fourier_func` – функция частичной суммы ряда Фурье, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `caption` – подпись к графику, `title` – название файла для сохранения.

```
def plot_parametric_func(func, fourier_func, t0, T, caption = '', title =
   ''):
    t = np.linspace(t0, t0 + T, 1000)
    plt.figure(figsize=(8, 5))
    plt.plot(func(t).real, func(t).imag)
    if fourier_func != func:
        plt.plot(fourier_func(t).real, fourier_func(t).imag)
    # add labels
    plt.xlabel('Re')
    plt.ylabel('Im')
    if fourier_func != func:
        plt.legend(['Source function', 'Fourier function'], loc='upper
   right')
    else:
        plt.legend(['Source function'], loc='upper right')
    # add caption
    plt.title(caption)
    plt.grid()
    plt.savefig(title + '.png')
```

Листинг 22: Функция для отрисовки графиков (параметрическая функция)

Аргументы функции `plot_parametric_func`: `func` – исходная функция, `fourier_func` – функция частичной суммы ряда Фурье, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `caption` – подпись к графику, `title` – название файла для сохранения.

```
def calc_and_plot(func, t0, T, N, title):
    plot_func(func, func, t0, T, 'Source function', title)
    for n in N:
        right_caption = f'Fourier function, N = {n}'
        fourier_func = fourierise(func, t0, T, n)

        plot_func(func, fourier_func, t0, T, right_caption, title +
   f'_N_{n}')
```

Листинг 23: Функция для разложения в ряд Фурье с различными значениями N

Аргументы функции `calc_and_plot`: `func` – исходная функция, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов N, `title` – название файла для сохранения.

```
def calc_and_plot_exp(func, t0, T, N, title):
    plot_func(func, func, t0, T, 'Source function', title)
    for n in N:
        right_caption = f'Fourier function (exp), N = {n}'
        fourier_func = fourierise_exp(func, t0, T, n)
```

```
        plot_func(func, fourier_func, t0, T, right_caption, title +
   f'_N_{n}')
```

Листинг 24: Функция для разложения в ряд Фурье с различными значениями N

Аргументы функции `calc_and_plot_exp`: `func` – исходная функция, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов N, `title` – название файла для сохранения.

```
def calc_and_plot_parametric(func, t0, T, N, title):
    plot_parametric_func(func, func, t0, T, 'Source function', title)
    for n in N:
        right_caption = f'Fourier function, N = {n}'
        fourier_func = fourierise_exp(func, t0, T, n)

        plot_parametric_func(func, fourier_func, t0, T, right_caption,
   title + f'_N_{n}')
```

Листинг 25: Функция для разложения в ряд Фурье с различными значениями N (параметрическая функция)

Аргументы функции `calc_and_plot_parametric`: `func` – исходная функция, `t0` – начало промежутка, на котором проводится разложение, `T` – период функции, `N` – количество коэффициентов N, `title` – название файла для сохранения.

```
def print_fourier_coefficients(a, b):
    for i in range(len(a)):
        print(f'a_{i} = \t{a[i]:.5f}, \tb_{i} = \t{b[i]:.5f}')
```

Листинг 26: Функция для вывода коэффициентов

Аргументы функции `print_fourier_coefficients`: `a` и `b` – коэффициенты Фурье.

```
def print_fourier_exp_coefficients(c):
    for i in range(len(c)):
        print(f'c_{i - len(c) // 2} = \t{c[i]:.5f}')
```

Листинг 27: Функция для вывода коэффициентов

Аргументы функции `print_fourier_coefficients_exp`: `c`– коэффициенты Фурье.

```
func = np.vectorize(lambda x: 1 if 0 <= (x - 1) % 3 < 1 else 2)
print("Func 1")
a, b = fourier(func, 1, 3, 3)
print_fourier_coefficients(a, b)
c = fourier_exp(func, 1, 3, 3)
```

```
print_fourier_exp_coefficients(c)

calc_and_plot(func, 1, 3, [1, 2, 5, 15, 30], './media/plots/func_1')
calc_and_plot_exp(func, 1, 3, [1, 2, 5, 15, 30], './media/plots/func_1_exp')
```

Листинг 28: Пример работы с функциями

В данном примере функция `func` (согласно уравнению (5)) – функция с периодом $T = 3$. `a, b, c` – коэффициенты Фурье для $N = 3$

B Исходный код для дополнительного задания

Исходный код для дополнительного задания можно найти по ссылке: GitHub

```
def scalar_product(f, g, number_of_points):
    x = np.linspace(0, 1, number_of_points)
    dx = x[1] - x[0]
    return np.dot(f(x), g(x)) * dx
```

Листинг 29: |Скалярное произведение на интервале [1, 0]

Аргументы функции `scalar_product`: `f` и `g` – функции, для которых вычисляется скалярное произведение, `number_of_points` – количество точек, на которое разбивается кривая из картинки.

```
def fourier_exp(func, number_of_points, N):
    c = []
    for i in range(-N, N + 1):
        exp = lambda x: np.e ** (-1j * 2 * np.pi * i * x)
        c.append((scalar_product(func, exp, number_of_points), i))
    return c
```

Листинг 30: Вычисление коэффициентов Фурье

Аргументы функции `fourier_exp`: `func` – функция, для которой вычисляются коэффициенты, `number_of_points` – количество точек, на которое разбивается кривая из картинки, `N` – количество коэффициентов.

```
def spline_decomposition(spline, scale, number_of_points,
   number_of_components):
    path = svg.path.parse_path(spline)
    path_func = np.vectorize(lambda t: path.point(t) * scale)

    c = fourier_exp(path_func, number_of_points, number_of_components)
    return c
```

Листинг 31: Разложение линии по Фурье

Аргументы функции `spline_decomposition`: `spline` – кривая, для которой проводится разложение, `scale` – масштаб, `number_of_points` – количество точек, на которое разбивается кривая из картинки, `number_of_components` – количество коэффициентов.

```
def get_length_and_start_proportion(c):
    length = np.sqrt(c.real ** 2 + c.imag ** 2)
```

```
    prop = math.atan2(-c.imag, c.real)
    if prop < 0: prop += 2 * math.pi
    prop /= 2 * math.pi
    return length, prop
```

Листинг 32: Получение радиуса и начального положения

Аргументы функции `get_length_and_start_proportion`: `c` – коэффициент ряда Фурье.

```
def fourier_exp_func(c, N):
    return lambda x: sum([c[i + len(c) // 2][0] * np.e ** (1j * 2 * np.pi *
   i * x)  for i in range(-N, N + 1)])

def fourierise_exp(func, number_of_points, N):
    c = fourier_exp(func, number_of_points, N)
    return fourier_exp_func(c, N)
```

Листинг 33: Получение частичной суммы ряда Фурье

Аргументы функции `fourierise_exp`: `func` – функция, для которой проводится разложение, `number_of_points` – количество точек, на которое разбивается кривая из картинки, `N` – количество коэффициентов.

```
class RotatingCircle(Circle):
    def __init__(self, radius, frequency, initial_dot_position, parent,
   **kwargs):
        super().__init__(radius=radius, **kwargs)
        self.parent = parent
        self.frequency = frequency
        self.dot_position = initial_dot_position

        if parent is not None:
            self.move_to(parent.point_from_proportion(parent.dot_position %
   1))
```

Листинг 34: Исходный код класса окружности

```
class DotOnCircle(Dot):
    def __init__(self, inner_circle, **kwargs):
        super().__init__(**kwargs)
        self.inner_circle = inner_circle

        if inner_circle is not None:

   self.move_to(inner_circle.point_from_proportion(inner_circle.dot_position
   % 1))
```

Листинг 35: Исходный код класса точки на окружности

```
class VectorInCircle(Arrow):
```

```
    def __init__(self, inner_circle, **kwargs):
        super().__init__(start=inner_circle.get_center(),
   end=inner_circle.point_from_proportion(inner_circle.dot_position % 1))
        self.inner_circle = inner_circle
```

Листинг 36: Исходный код класса вектора внутри окружности

```
class Drawing(ZoomedScene):
    def construct(self):
        self.camera.frame.set(width=15 * frame_scale)

        def rotating_circle_updater(circle, dt):
            circle.dot_position += (dt * circle.frequency) # change on
   circle dot position
            if circle.parent is not None:

   circle.move_to(circle.parent.point_from_proportion(circle.parent.dot_
                position % 1)) # move center to parent circle dot position

        def dot_on_circle_updater(dot, dt):
            if dot.inner_circle is not None:
                dot.move_to(dot.inner_circle.point_from_
                proportion(dot.inner_circle.dot_position % 1))

        def vector_in_circle_updater(vector, dt):
            vector.put_start_and_end_on(vector.inner_circle.get_center(),

   vector.inner_circle.point_from_proportion(vector.inner_circle.dot_position
   % 1))


        coefficients = spline_decomposition(spline, scale,
   number_of_points, number_of_components)
        coefficients.sort(key=lambda x: np.sqrt(x[0].real ** 2 + x[0].imag
   ** 2), reverse=True) # sort by length
        # coefficients.sort(key=lambda x: x[1]) # sort by frequency

        components = []

        for coefficient in coefficients:
            r, a = get_length_and_start_proportion(coefficient[0])
            n = coefficient[1] / time_scale
            if n == 0: continue

            c = RotatingCircle(r, n, a, components[-1][0] if
   len(components) > 0 else None, color=WHITE, stroke_width=0.5)
            # d = DotOnCircle(c, color=BLUE)
            v = VectorInCircle(c, color=WHITE)

            c.add_updater(rotating_circle_updater)
            # d.add_updater(dot_on_circle_updater)
            v.add_updater(vector_in_circle_updater)

            components.append([c, v])
```

```
        for i in range(len(components)):
            self.add(components[i][0], components[i][1])


        trace = TracedPath(components[-1][1].get_end, stroke_color=BLUE,
 stroke_width=5, dissipating_time=4.8)
        self.add(trace)

        self.wait(10)
```
Листинг 37: Исходрый код основного класса анимации