Федеральное государственное автономное образовательное
учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа №5

СВЯЗЬ НЕПРЕРЫВНОГО И ДИСКРЕТНОГО

Студенты: Загайнов А.А.
Поток: ЧАСТ.МЕТ. 1.2

Преподаватели: Перегудин. А.А., Пашенко А.В.

Санкт-Петербург
2025

Содержание

Задание 1. Непрерывное и дискретное преобразование Фурье. **3**
Задание 1.1 Численное интегрирование 4
Задание 1.2 Использование DFT 12
Задание 1.3 Мои объяснения 17
Задание 1.4 Приближение непрерывного с помощью DFT. 17

Задание 2. Семплирование. **24**

Общий вывод **30**

Приложение **31**

Задание 1. Непрерывное и дискретное преобразование Фурье.

Для начала введем привычную нам прямоугольную функцию $\Pi : R \to R$:

$$\Pi(t) = \begin{cases} 1, |t| \leq 1/2, \\ 0, |t| > 1/2 \end{cases}$$

Найдем и её Фурье-образ:

$$\hat{\Pi}(v) = \int\limits_{-\infty}^{\infty} \Pi(t) e^{-2\pi i v t} dt = \int\limits_{-1/2}^{1/2} e^{-2\pi i v t} dt = \frac{e^{-2ivt}}{-2iv}\Big|_{-1/2}^{1/2} = \frac{e^{-\pi iv} - e^{\pi iv}}{-2\pi iv} = \frac{\sin(\pi v)}{\pi v}$$

Результат, это, как мы помним из предыдущих работ - кардинальный синус. Изобразим графики:

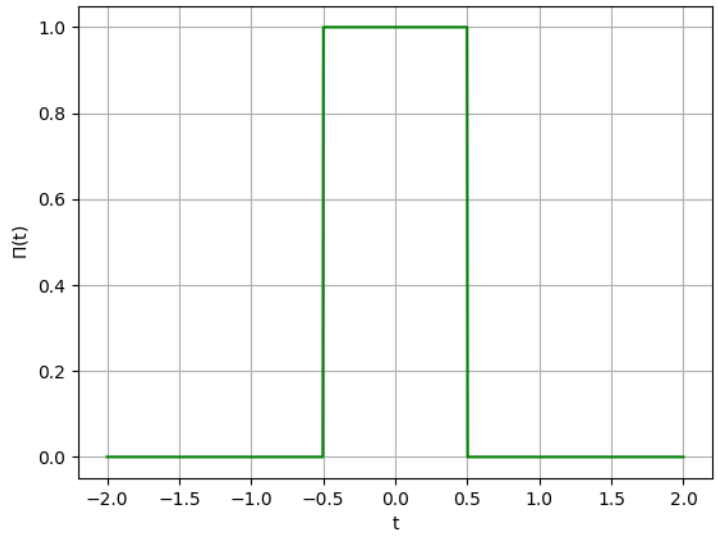

Рис. 1: График функции

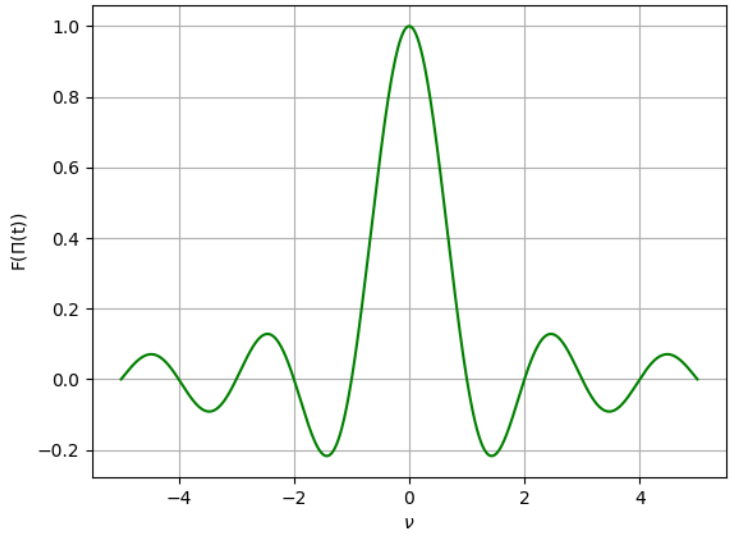

Рис. 2: Фурье образ функции

Задание 1.1 Численное интегрирование

В первом задании опробуем при помощи численного интегрирования функцией trapz из модуля numpy сделать обычное и обратное преобразования Фурье функции. Сравним результаты с аналитическими при разных параметрах T, V, dT, dV. Выводы подведем в конце

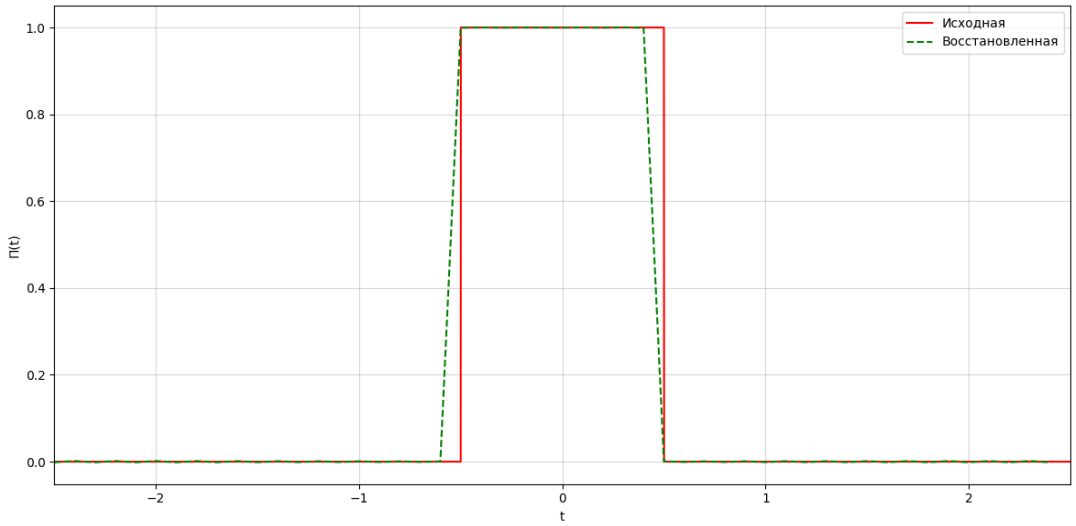

Рис. 3: Восстановленная и оригинальная функция. $T = 5, V = 10, dT = 0.1, dV = 0.1$

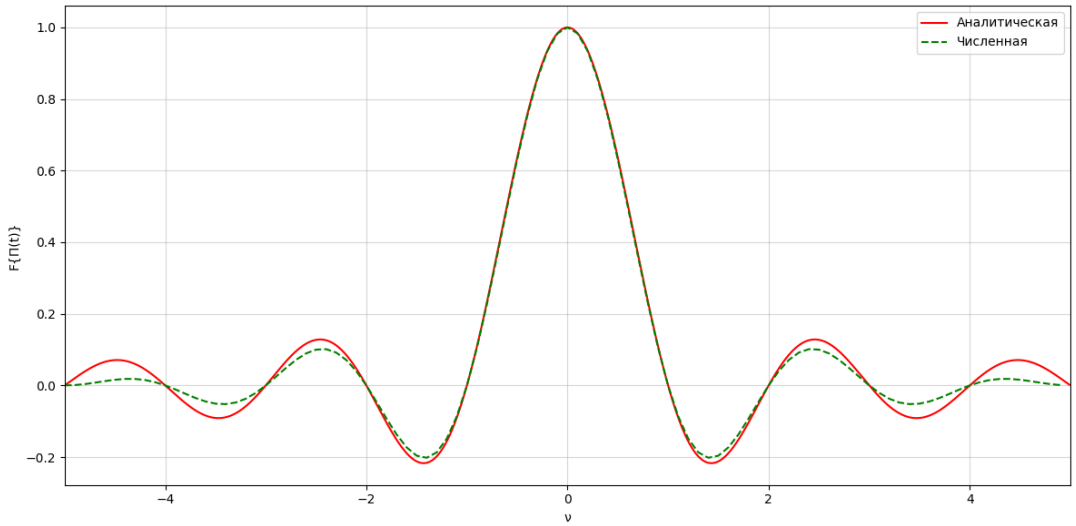

Рис. 4: Численный и аналитический Фурье образ. $T = 5, V = 10, dT = 0.1, dV = 0.1$

Результат умеренно близок к оригиналу. Возьмем эти параметры за отправную точку, при сравнении воздействия каждого из параметров на результат.

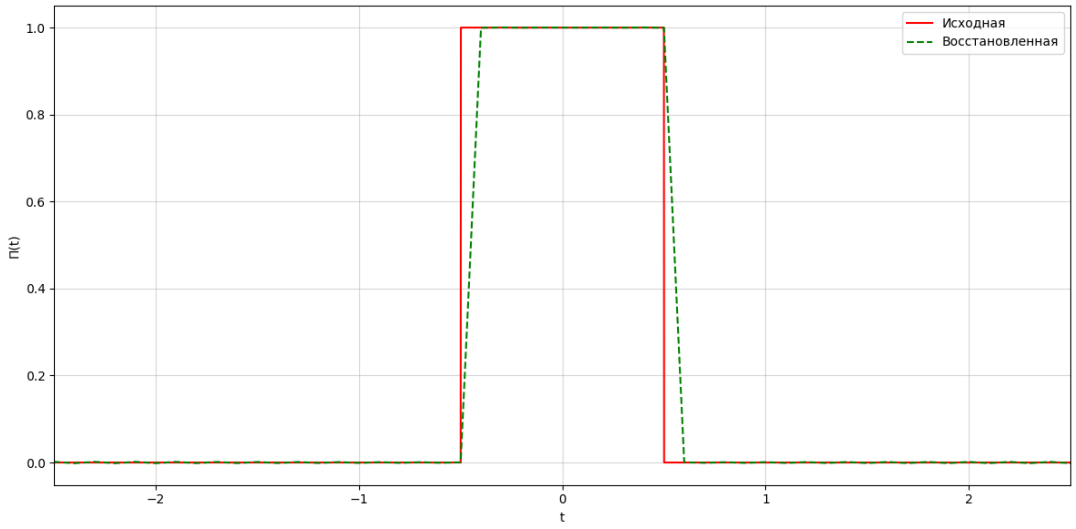

Рис. 5: Восстановленная и оригинальная функция. $T = 20, V = 10, dT = 0.1, dV = 0.1$

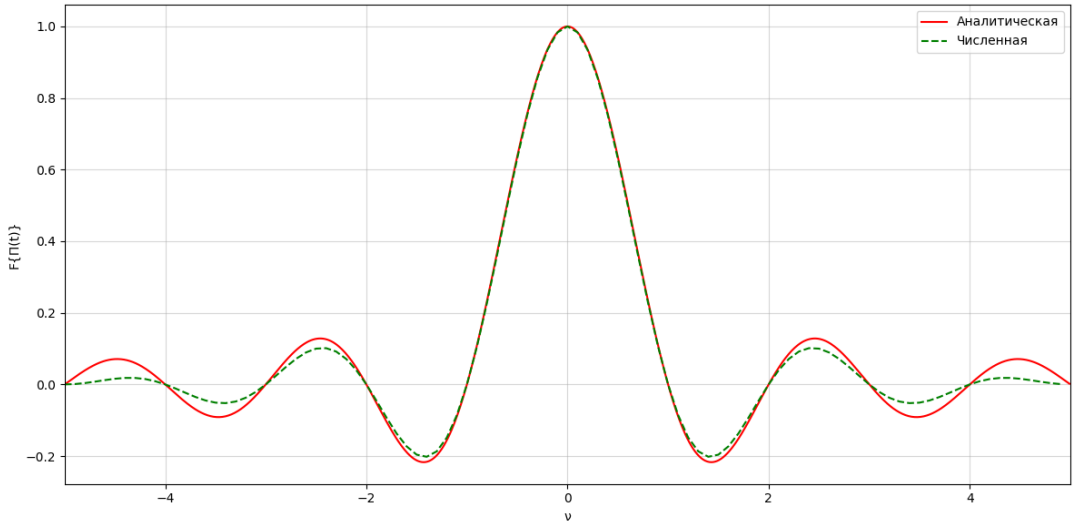

Рис. 6: Численный и аналитический Фурье образ. $T = 20, V = 10, dT = 0.1, dV = 0.1$

Увеличение T до 20 при данных параметрах почти не влияет на результат. Небольшой сдвиг восстановленной функции относительно T=5 скорее схоже на особенности влияния малых вычислительных погрешностей в зоне низких частот, даже малое различие может заметно сдвинуть функцию.

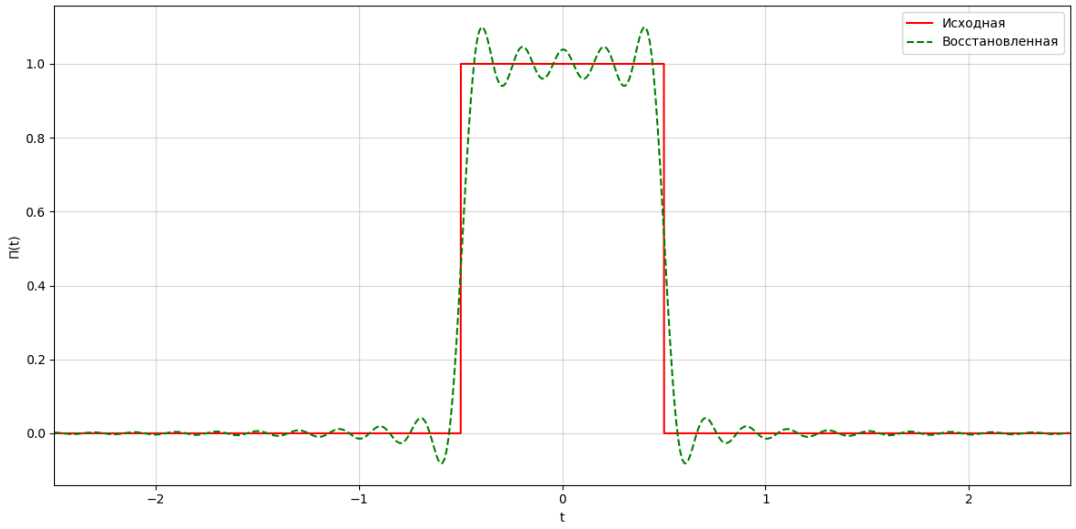

Рис. 7: Восстановленная и оригинальная функция. $T = 5, V = 10, dT = 0.01, dV = 0.1$

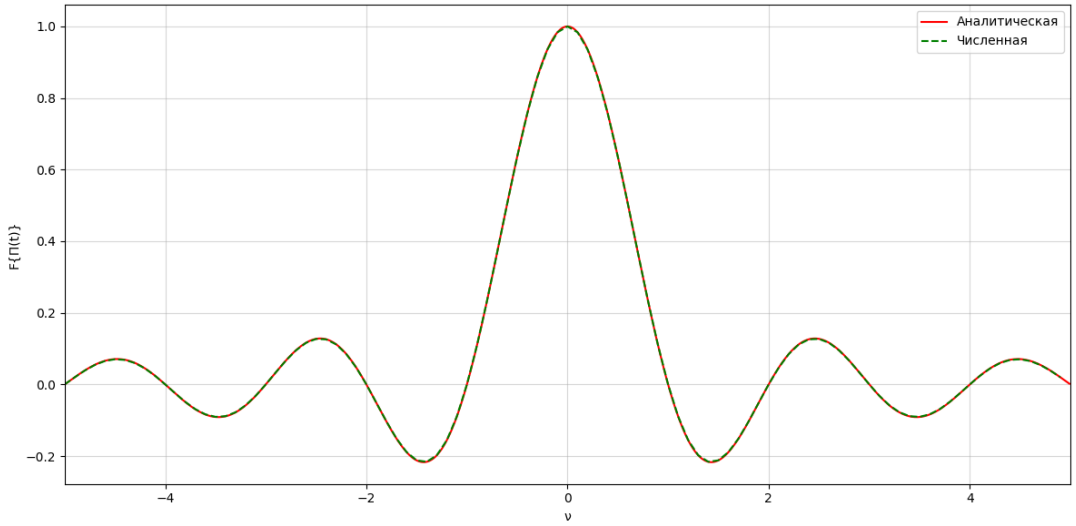

Рис. 8: Численный и аналитический Фурье образ. $T = 5, V = 10, dT = 0.01, dV = 0.1$

Уменьшение значения dT вызвало появление эффекта Гиббса на восстановленном графике. При первичном преобразовании Фурье больше деталей было сохранено, потому Фурье образ почти полностью совпадает с аналитическим. Отсюда же и берется эффект Гиббса, ведь Фурье-образ не идеален и небольшие отличия в высоких частотах и создают наблюдаемый эффект.

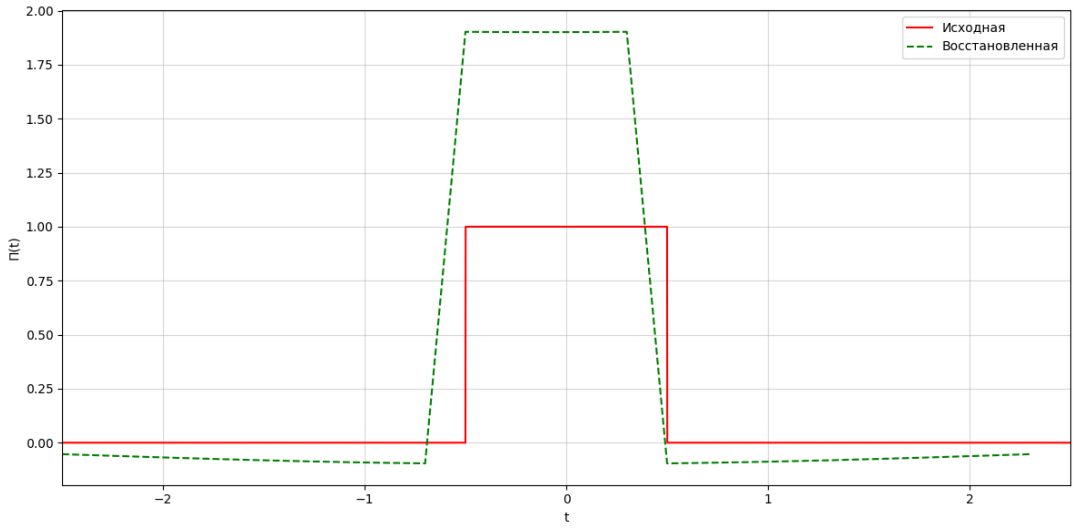

Рис. 9: Восстановленная и оригинальная функция. $T = 5, V = 10, dT = 0.2, dV = 0.1$

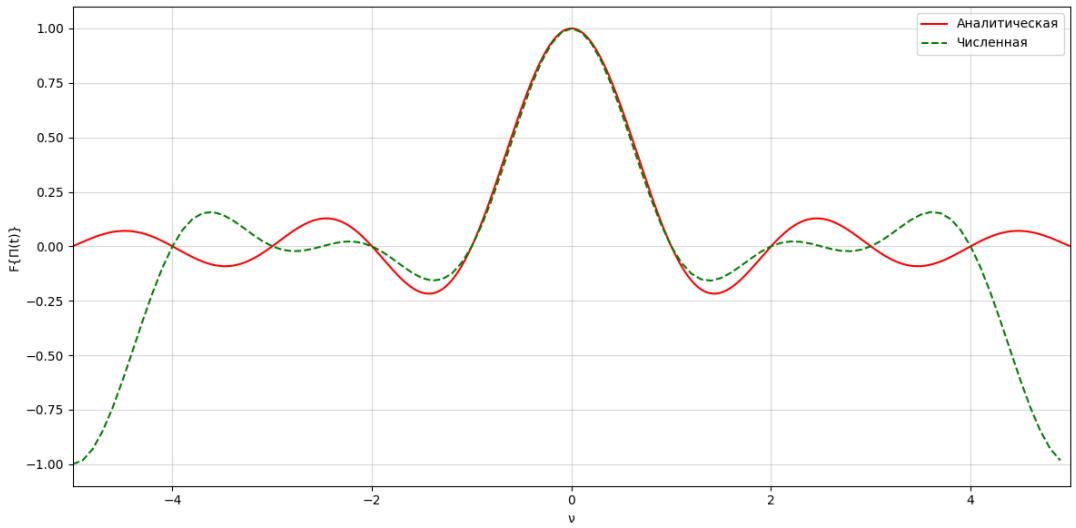

Рис. 10: Численный и аналитический Фурье образ. $T = 5, V = 10, dT = 0.2, dV = 0.1$

Увеличение dT привело к очень большим неточностям в сохранении рисунка функции. Результат очень сильно отличается, что видно как по восстановленному графику, так и по Фурье-образу.

Это прямое действие эффекта от теоремы Найквиста. Частота дискретизации должна быть выше или равна двойной максимальной частоте описываемого спектра. Самая высокая частота в нашем случае это $(V/2) * 2 =$ 10гц. Если частота дискретизации при $dT = 0.1$ была равна 10гц, что нас устраивало, то при $dT = 0.2$ она равна 5 гц, что не подходит. Потому и результат вышел страдающим от алиасинга.

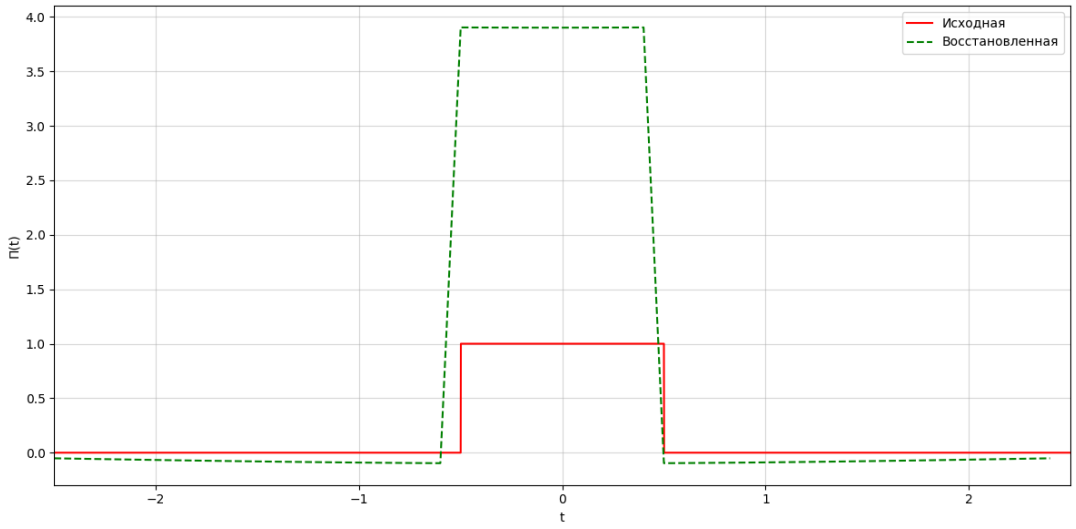

Рис. 11: Восстановленная и оригинальная функция. $T = 5, V = 40, dT = 0.1, dV = 0.1$

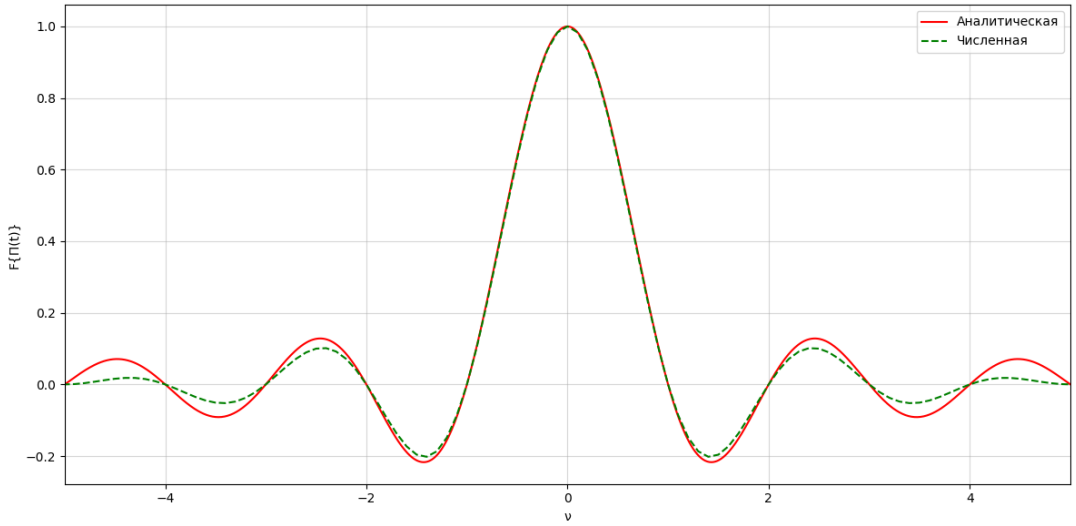

Рис. 12: Численный и аналитический Фурье образ. $T = 5, V = 40, dT = 0.1, dV = 0.1$

Расширение охватываемого диапазона V сыграло не на пользу результату. Конечно же это опять пример невыполнения условия Найквиста. Попробуем уменьшить значение dT, чтобы скопменсировать увеличение V.

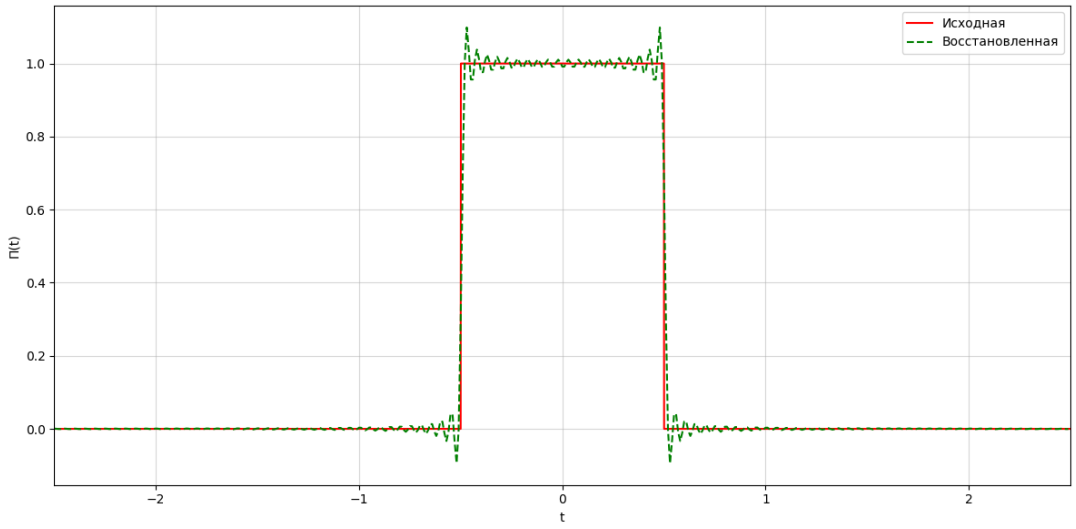

Рис. 13: Восстановленная и оригинальная функция. $T = 5, V = 40, dT = 0.01, dV = 0.1$

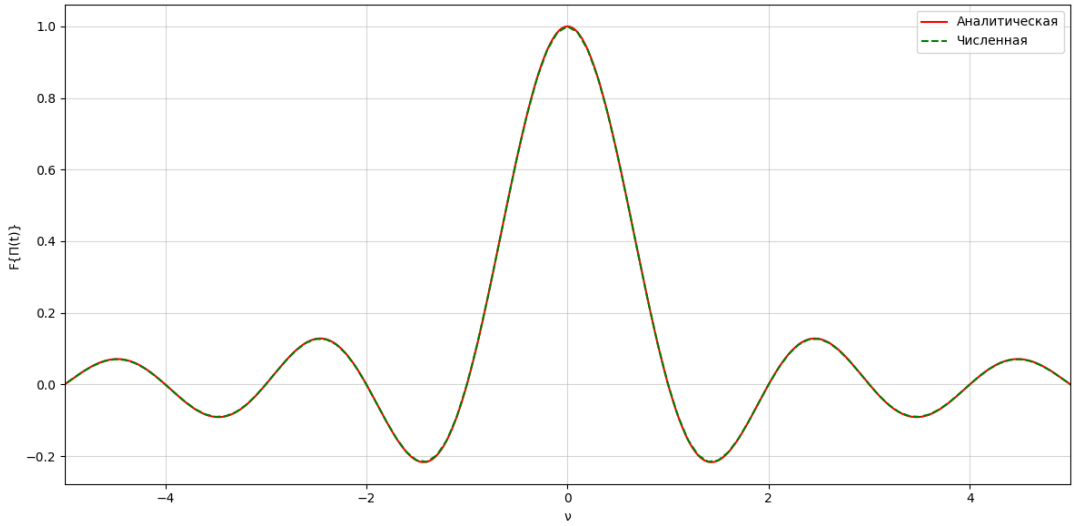

Рис. 14: Численный и аналитический Фурье образ. $T = 5, V = 40, dT = 0.01, dV = 0.1$

Теперь, когда условие Найквиста выполняется, можем порадоваться увеличенному V. Восстановленная функция стала очень близка к оригинальной функции(за исключением эффекта Гиббса).

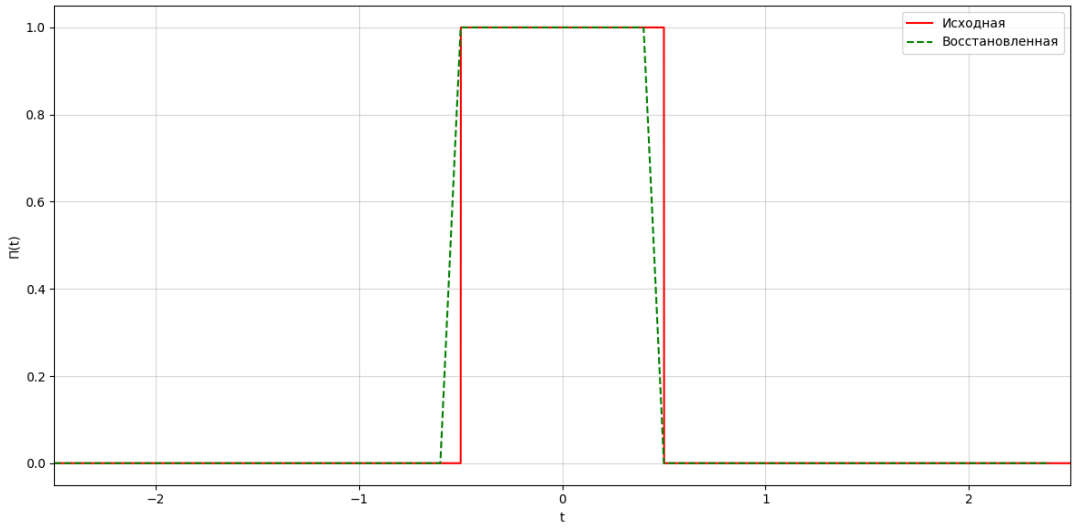

Рис. 15: Восстановленная и оригинальная функция. $T = 5, V = 10, dT = 0.1, dV = 0.01$

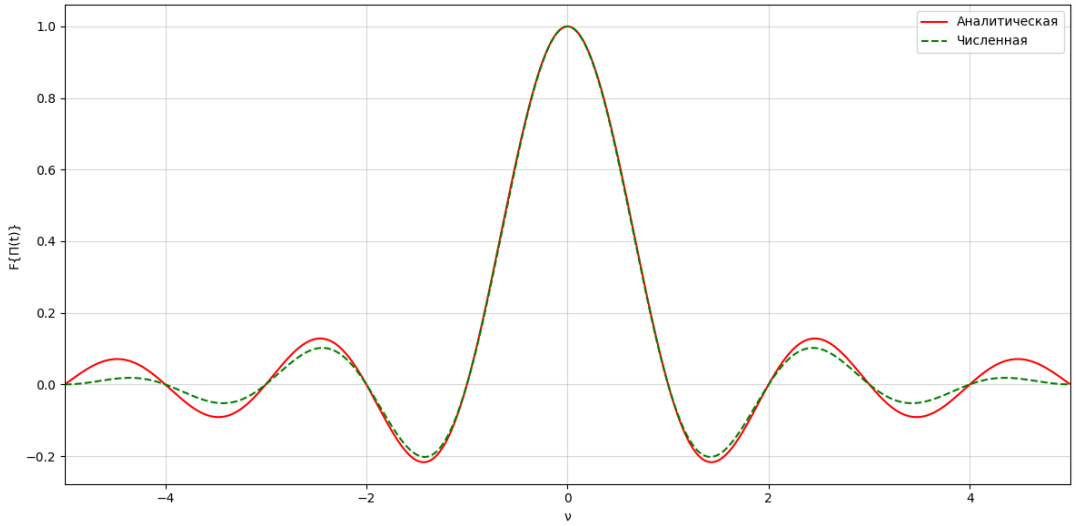

Рис. 16: Численный и аналитический Фурье образ. $T = 5, V = 10, dT = 0.1, dV = 0.01$

При уменьшенном значении dV не появилось каких-либо различий. Значения dV=0.1 было вполне достаточно для хорошей точности.

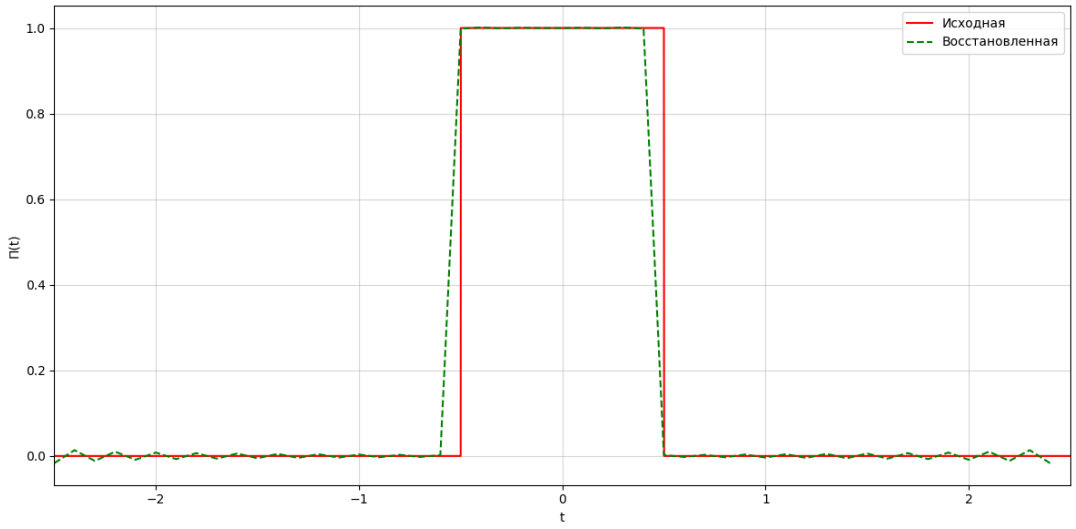

Рис. 17: Восстановленная и оригинальная функция. $T = 5, V = 10, dT = 0.1, dV = 0.3$

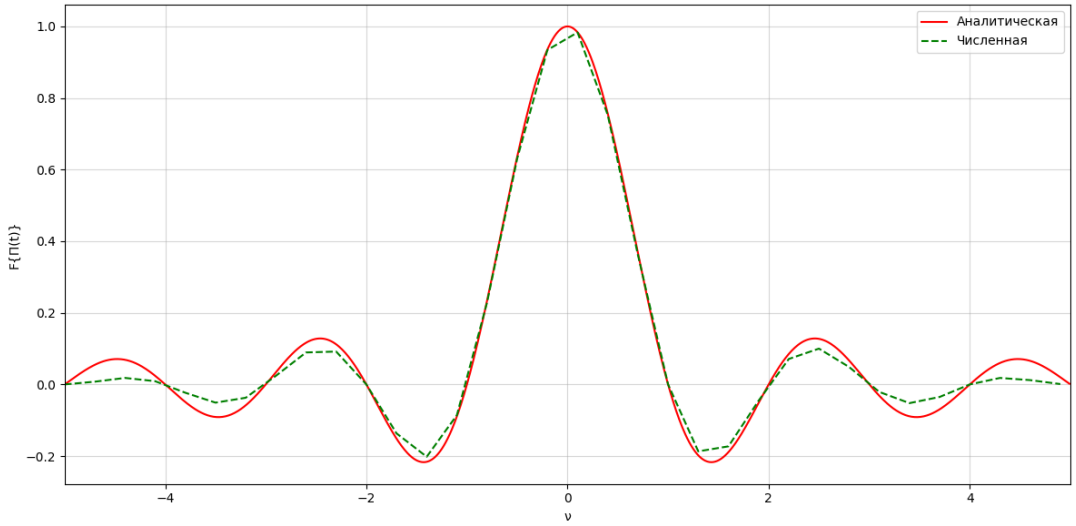

Рис. 18: Численный и аналитический Фурье образ. $T = 5, V = 10, dT = 0.1, dV = 0.3$

Повышение значения до 0.3 сделало изображение намного более «шероховатым». Прямые линии стали зигзагообразными. Все же значение 0.1 вполне подходит в качестве оптимального.

Вывод по заданию 1.1:

В ходе экспериментов я пришел к заключению, что значение $dV = 0.1$ и $dT = 0.01$ оказались оптимальными для численного преобразования Фурье(при $V = 10$). График похож на оригинал, но немного сдвинут. Можно добиться более ровно лежащей функции - при $dT = 0.025, V = 40$ будет ровно выполняться условие Найквиста, а эффект Гиббса еще не будет проявляться. Дальнейшее уменьшение дискретизации функции имеет не так много смысла, ведь требует очень большой вычислительной сложности.

Задание 1.2 Использование DFT

В этом задании будем исследовать работу DFT. Для работы на питоне воспользуемся функцией fft библиотеки numpy. Важно учесть тот факт, что преобразование должно быть унитарным (используем параметр **norm = 'ortho'**).

План действий такой:

1) Тестируем разные значения параметров T и Δt
2) Сравниваем аналитический фурье образ и наш дискретный
3) Сравниваем оригинальную и восстановленную функцию

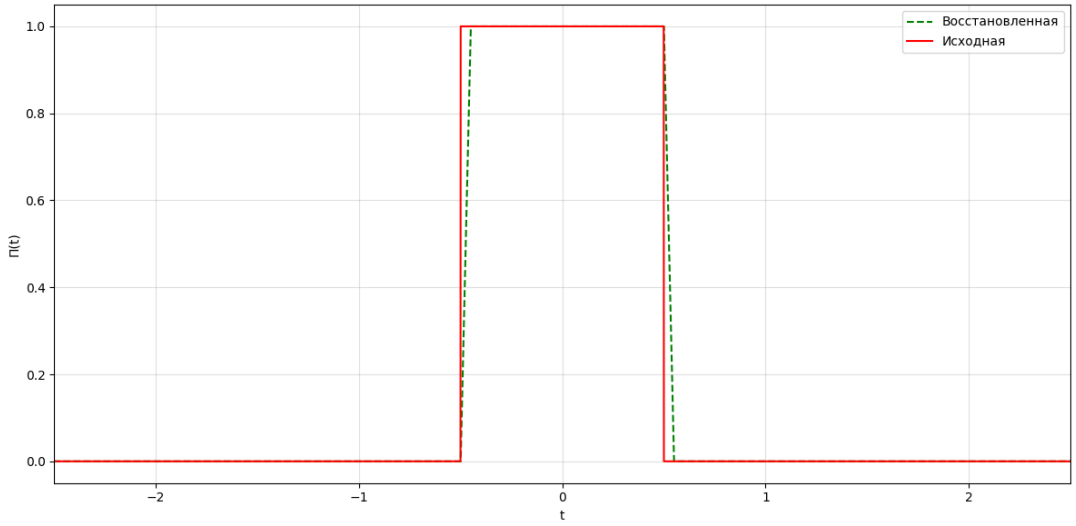

Рис. 19: Восстановленная и оригинальная функция. $T = 15, dT = 0.05$

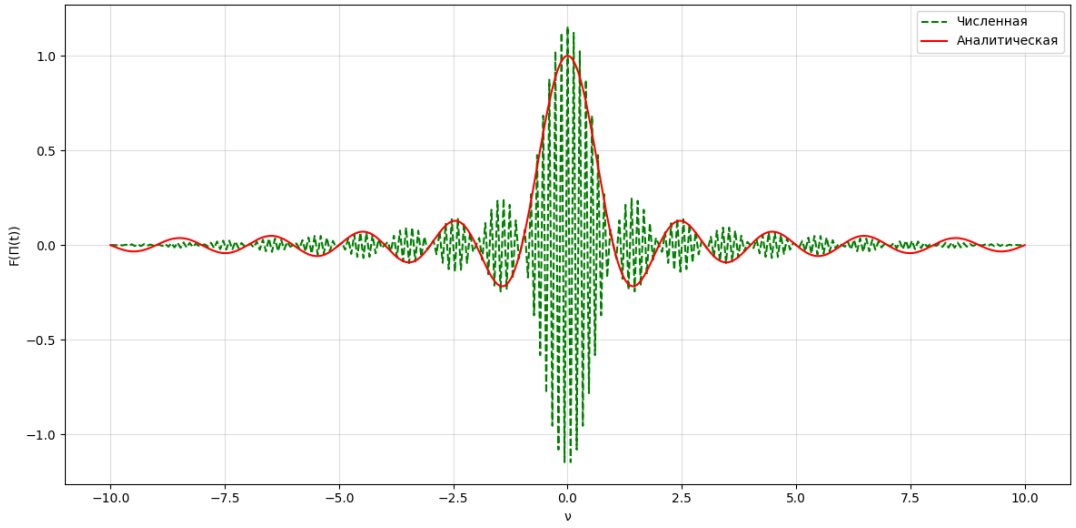

Рис. 20: Численный и аналитический Фурье образ. $T = 15, dT = 0.05$

Примем этот результат как отправную точку. Далее протестируем остальные варианты параметров

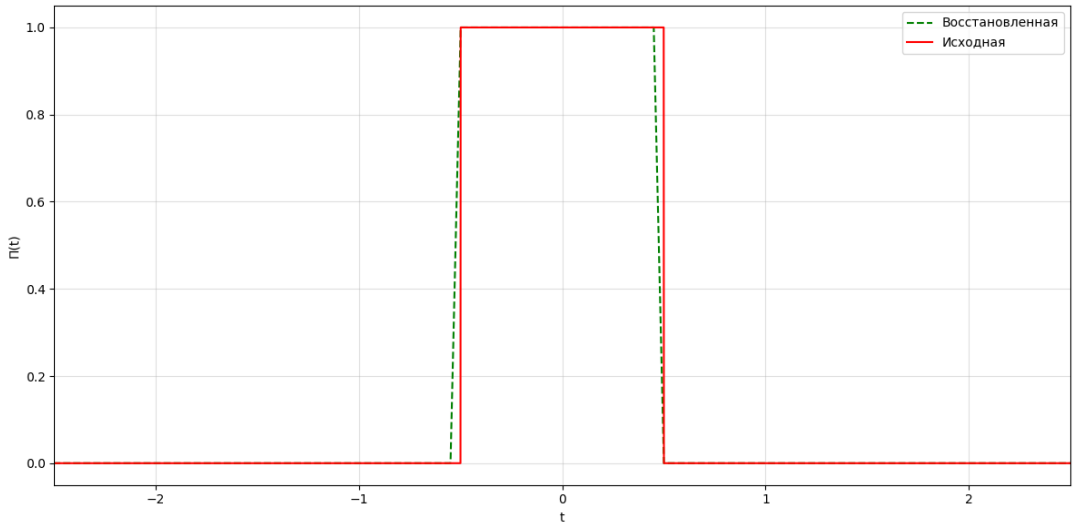

Рис. 21: Восстановленная и оригинальная функция. $T = 45, dT = 0.05$

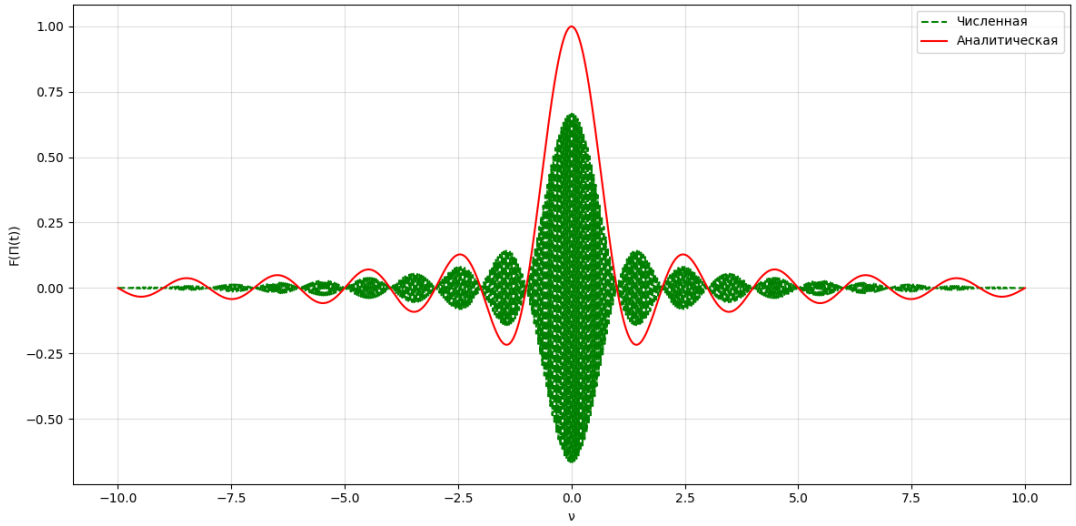

Рис. 22: Численный и аналитический Фурье образ. $T = 45, dT = 0.05$

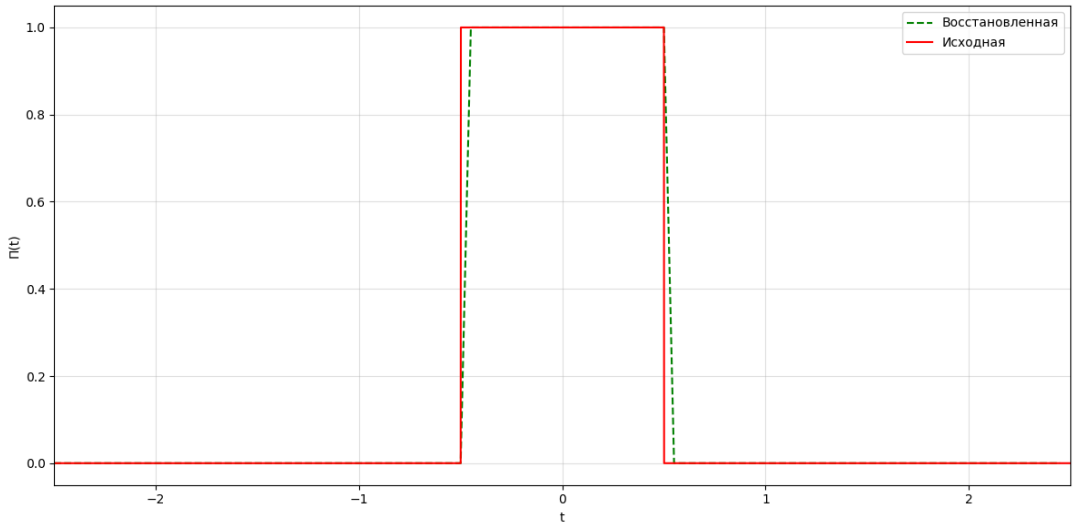

Рис. 23: Восстановленная и оригинальная функция. $T = 5, dT = 0.05$

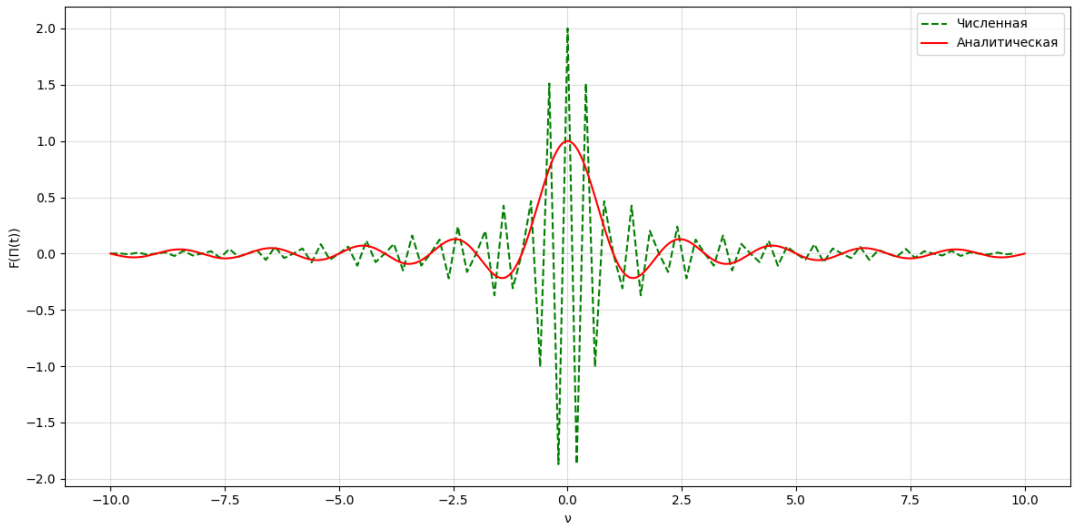

Рис. 24: Численный и аналитический Фурье образ. $T = 5, dT = 0.05$

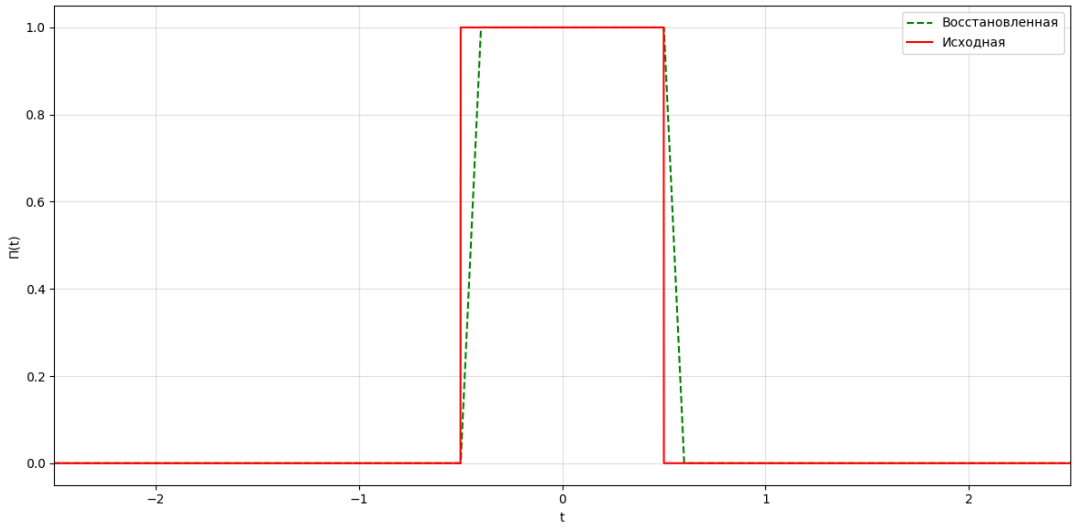

Рис. 25: Восстановленная и оригинальная функция. $T = 15, dT = 0.1$

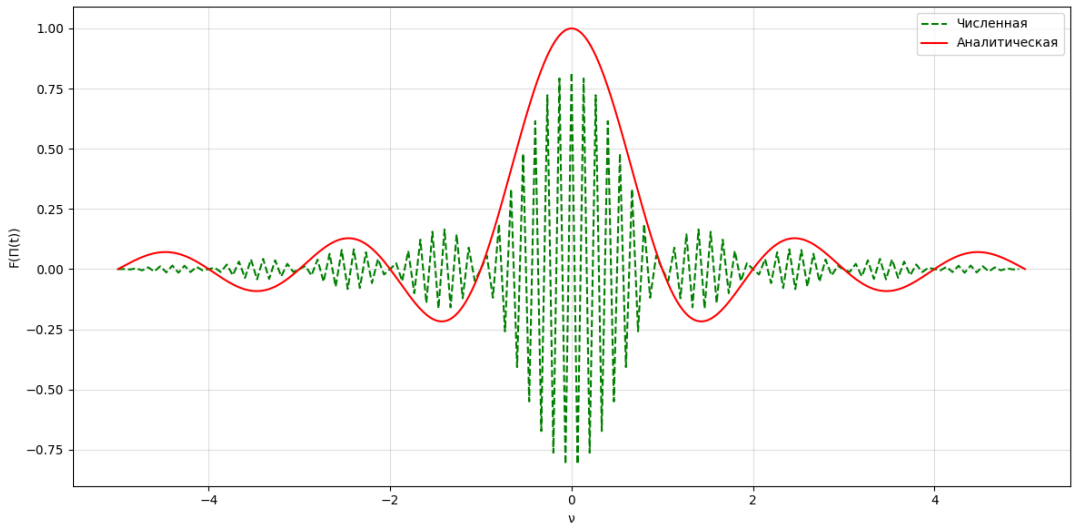

Рис. 26: Численный и аналитический Фурье образ. $T = 15, dT = 0.1$

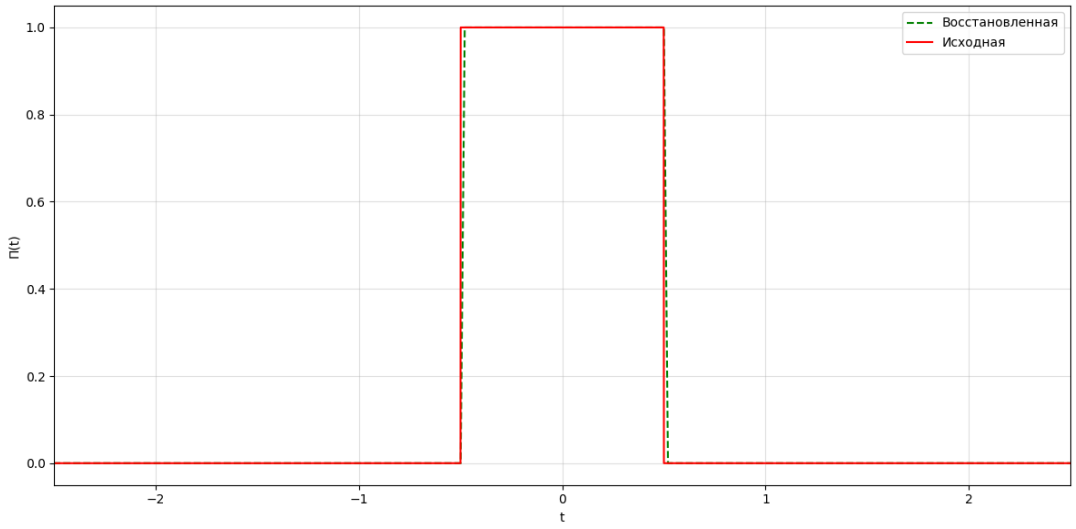

Рис. 27: Восстановленная и оригинальная функция. $T = 15, dT = 0.02$

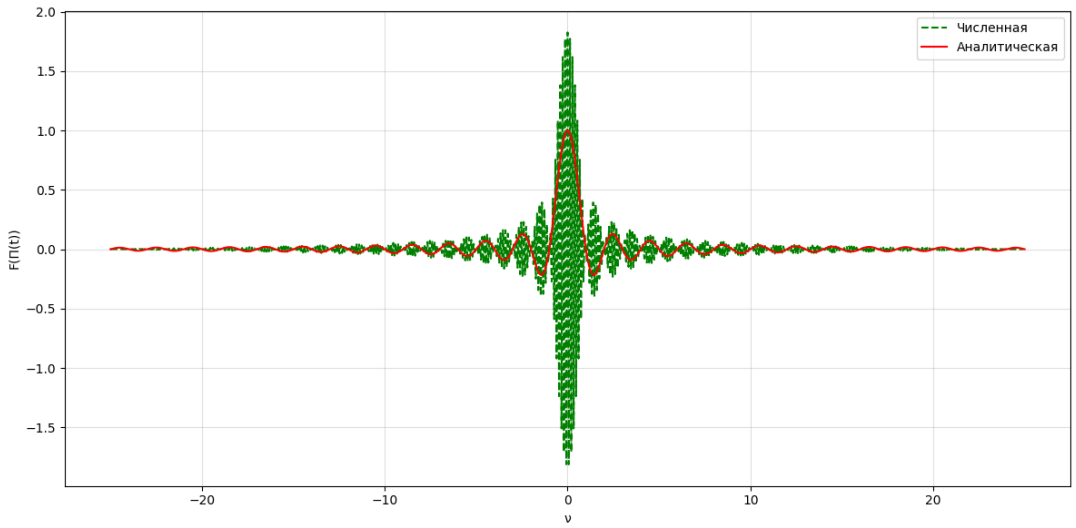

Рис. 28: Численный и аналитический Фурье образ. $T = 15, dT = 0.02$

Анализ и выводы по заданию 1.2:

По приведенным графикам видно наглядно, что увеличение T и, соответственно, уменьшение Δt увеличивают качество приближения к аналитическому результату. Но конечно это и заметно увеличивает временную сложность вычислений. Выше дискретизация и обьем захватываемой временной информации - выше сложность.

Задание 1.3 Мои объяснения

Анализируем результат и время. Заметим, DFT имеет плохую степень приближения, однако восстановленный график близок к истинному - метод работает быстро. В свою очередь, метод trapz даёт более точное приближение и требует заметно больше вычислительных мощностей. Не забываем про параметры Δv и V, их необходимо подбирать самостоятельно.

Почему результаты различаются? Во первых, DFT предполагает, что сигнал периодичен и представлен дискретными значениями. В отличие от него, метод trapz соответствует классу преобразования Фурье и лучше справляется с непрерывными временными сигналами. При достаточно малых интервалах дискретизации мы приближаемся к работе с непрерывными сигналами, что делает вычисление Фурье-образа более точным. В нашем случае сигнал является непериодическим, а интервалы дискретизации были довольно низки, так что в сумме факторов trapz справлялся лучше.

Во вторых, говоря о скорости - DFT использует матричные операции и внутренние симметрии для ускорения расчетов, что дает сложность $O(n \log n)$. В то время как метод trapz требует $O(n^2)$ операций, что делает его медленнее.

Задание 1.4 Приближение непрерывного с помощью DFT.

В этом задании попробуем улучшить наш обычный DFT метод парой новых идей для улучшения приближения непрерывного.

А именно:

Для приближения интегрирования воспользуемся суммой Римана:

$$\int_a^b f(t)\,dt = \sum_{n=0}^{N-1} f(t_n)\Delta t, \quad t_n = a + n\Delta t, \quad \Delta t = \frac{b-a}{N-1}$$

Преобразуем формулу преобразования Фурье с учетом суммы Римана и после подставляем $t_n = t_n + n\Delta t$:

$$\hat{f}(v) = \int_{-T/2}^{T/2} f(t)e^{-2\pi i v t}\,dt \longrightarrow c_m \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i n m/N} \quad \text{подставляем}$$

продолжаем

$$\hat{f}(\nu) \approx \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i \nu (t_0 + n\Delta t)}\Delta t$$

$$c_m = \frac{m}{N\Delta t}$$

$$\hat{f}(\nu) = \Delta t e^{-2\pi i \nu t_0} \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i n m/N} = \Delta t e^{-2\pi i \nu t_0}\mathrm{DFT}(f)$$

Наша новая формула делает поправки в фазе $e^{-2\pi i \nu t_0}$ и амплитуде Δt. Результат будет точнее, но при этом на такой же скорости как DFT.

Попробуем построить графики на разных параметрах T и Δt. Новый коэффициент в программе будем использовать следующим образом:

$$\mathcal{F} \to \mathrm{fftshift}(c_m \cdot \mathrm{fft}(f))$$

$$\mathcal{F}^{-1} \to \mathrm{ifft}(\mathrm{ifftshift}(\hat{f})/c_m)$$

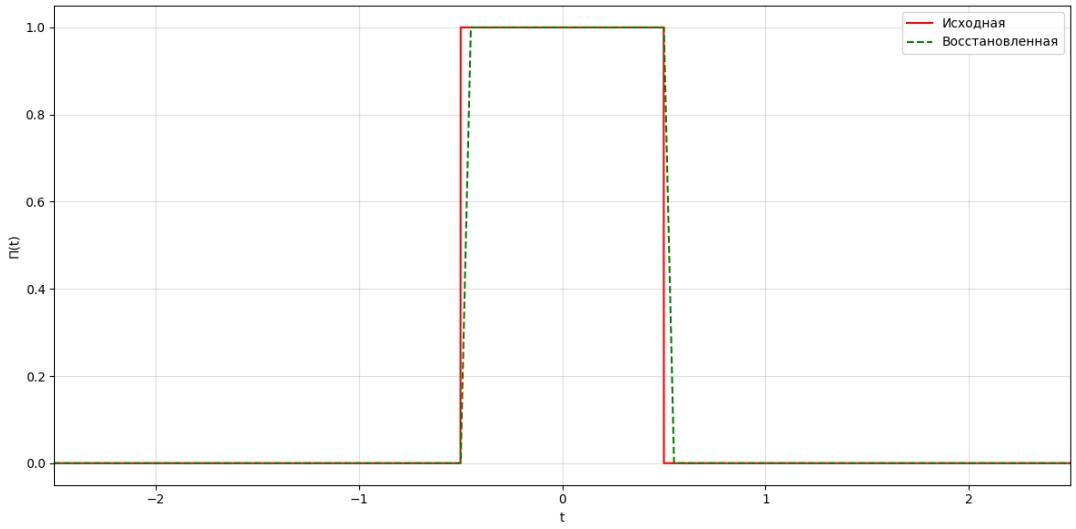

Рис. 29: Восстановленная и оригинальная функция. $T = 15, dT = 0.05$

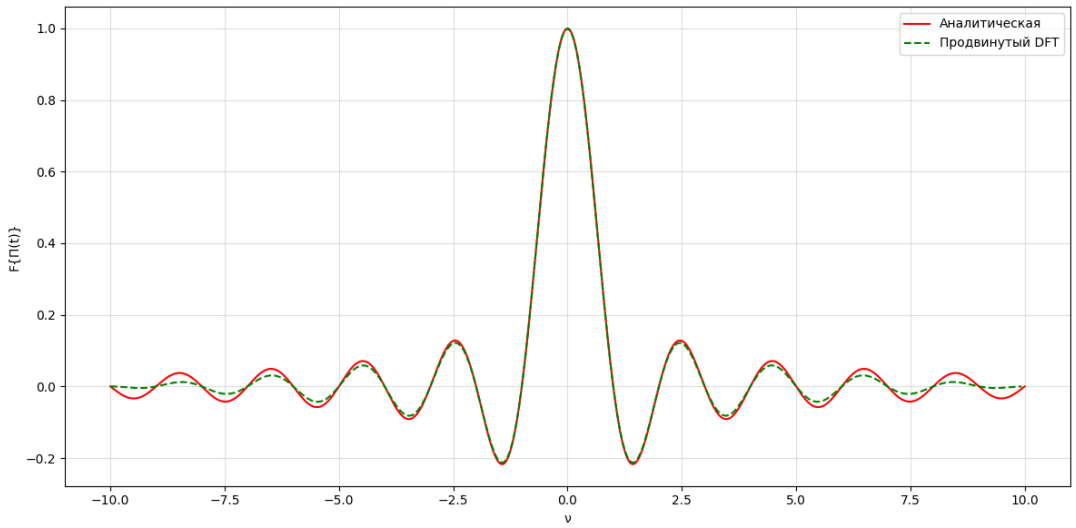

Рис. 30: Численный и аналитический Фурье образ. $T = 15, dT = 0.05$

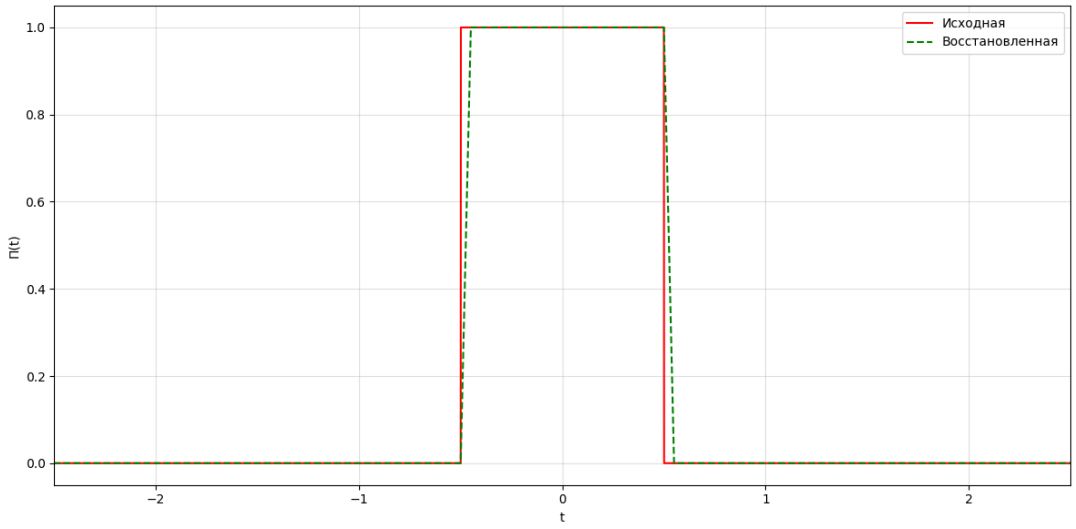

Рис. 31: Восстановленная и оригинальная функция. $T = 8, dT = 0.05$

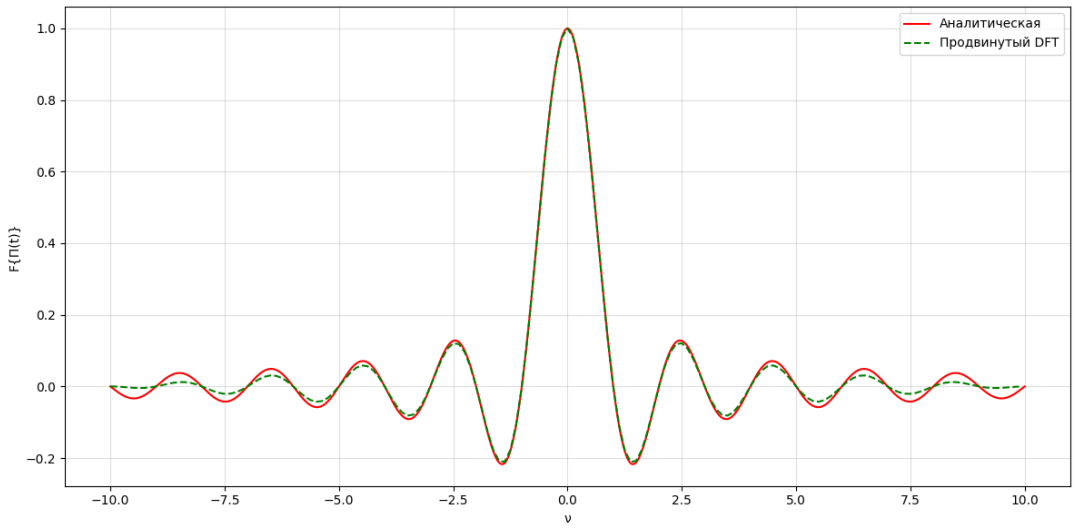

Рис. 32: Численный и аналитический Фурье образ. $T = 8, dT = 0.05$

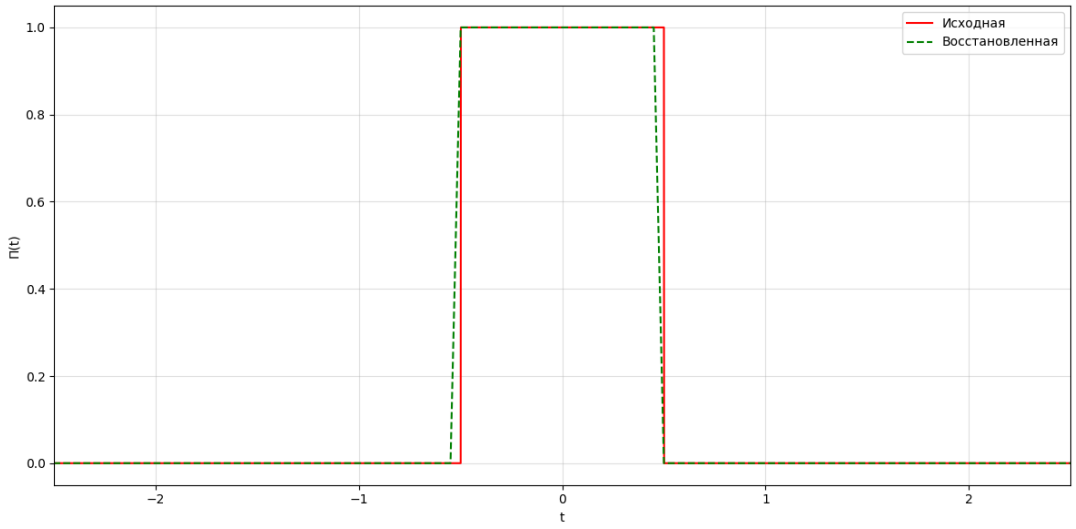

Рис. 33: Восстановленная и оригинальная функция. $T = 30, dT = 0.05$

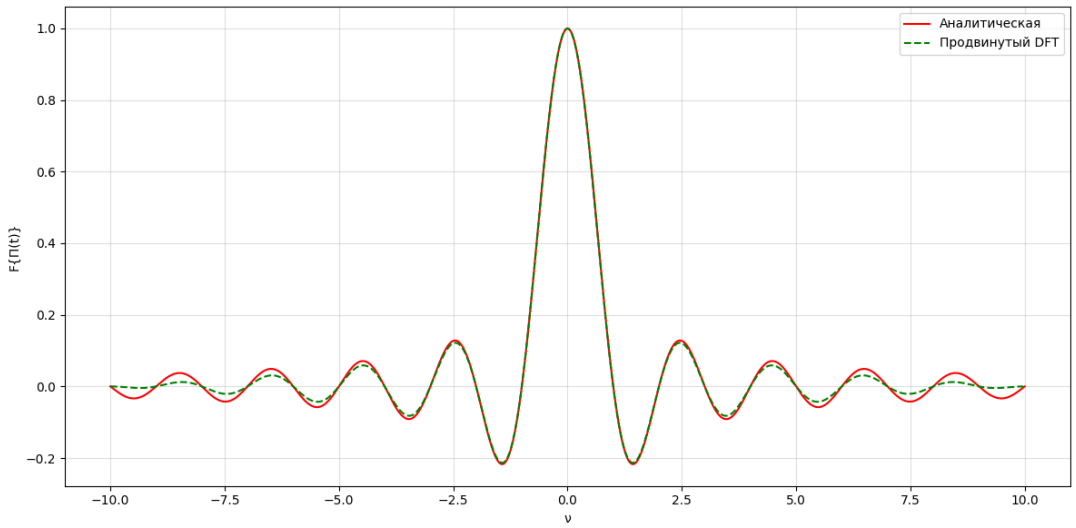

Рис. 34: Численный и аналитический Фурье образ. $T = 30, dT = 0.05$

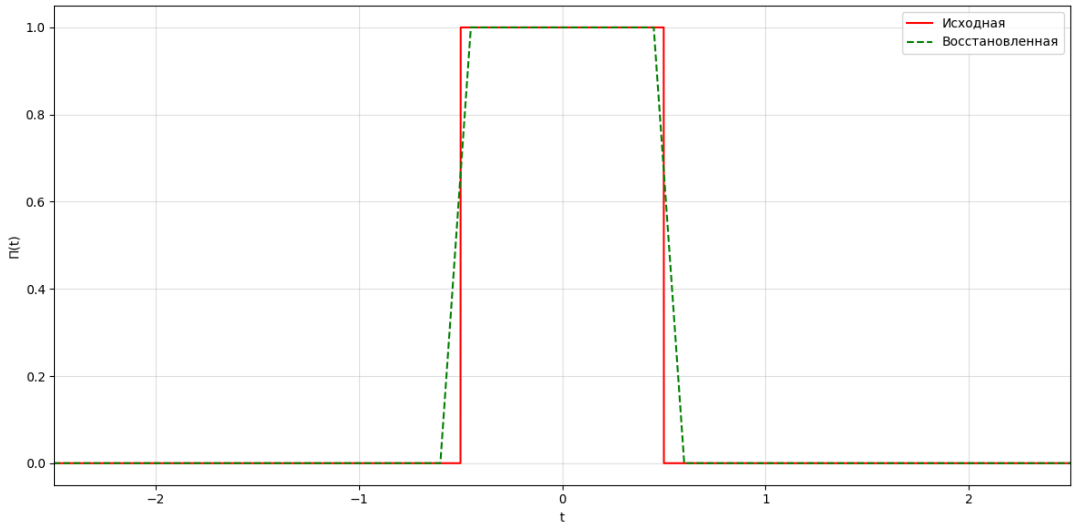

Рис. 35: Восстановленная и оригинальная функция. $T = 15, dT = 0.15$

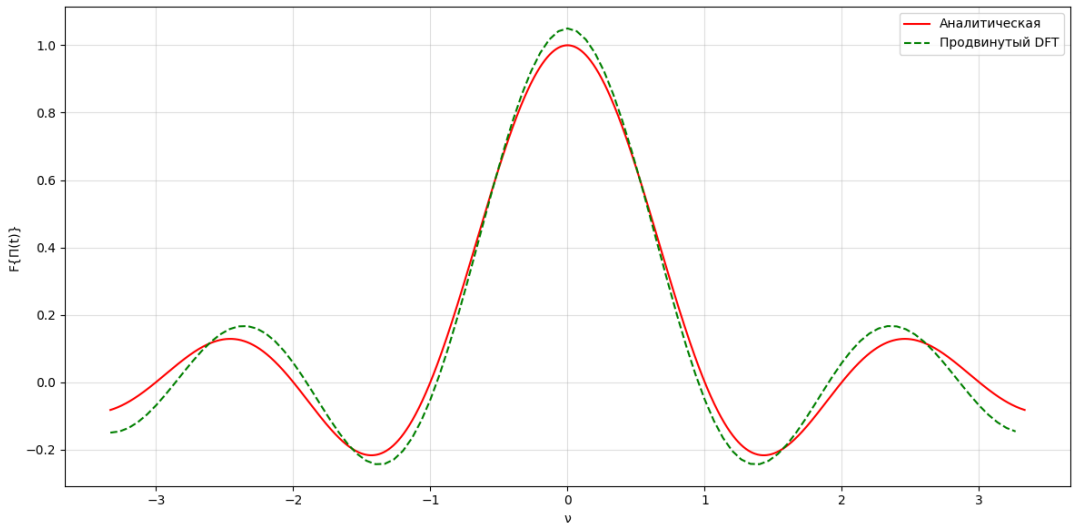

Рис. 36: Численный и аналитический Фурье образ. $T = 15, dT = 0.15$

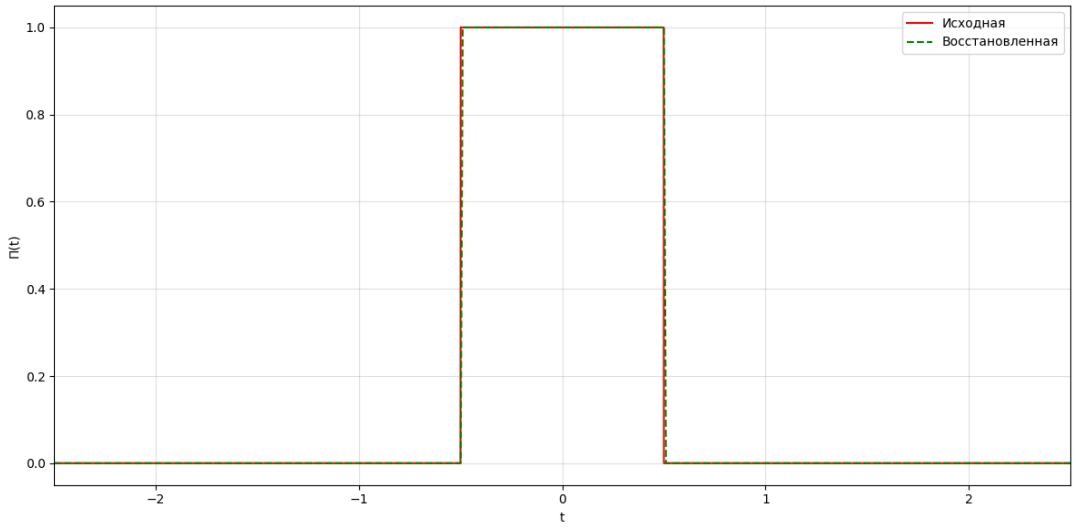

Рис. 37: Восстановленная и оригинальная функция. $T = 15, dT = 0.01$

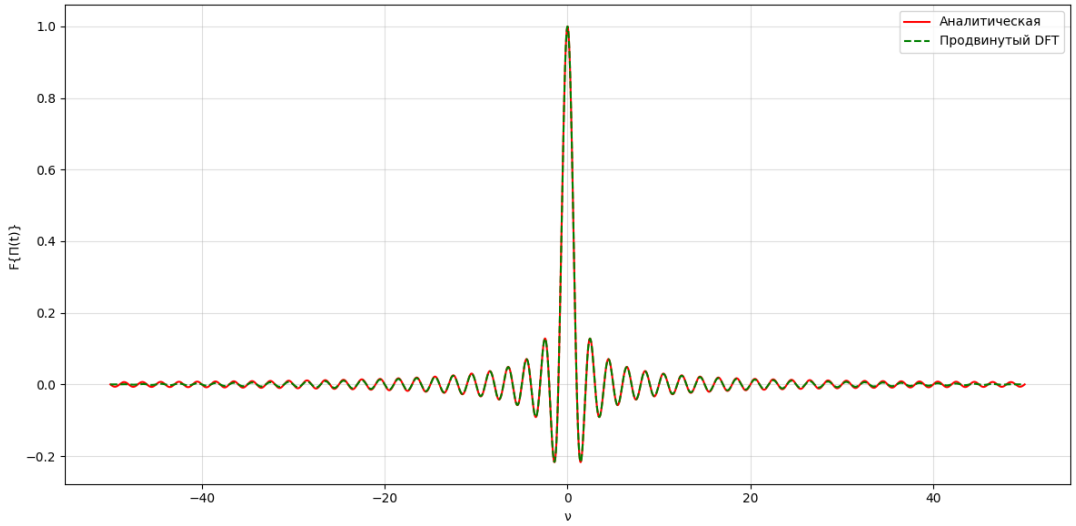

Рис. 38: Численный и аналитический Фурье образ. $T = 15, dT = 0.01$

Заметим, результат стал гораздо лучше - при достаточно малом Δt - как в последних двух графиках, результат стал почти идентичным оригинальному. Попробуем сравнить все оба метода на одном графике чтобы получить показательный результат нашей работы.

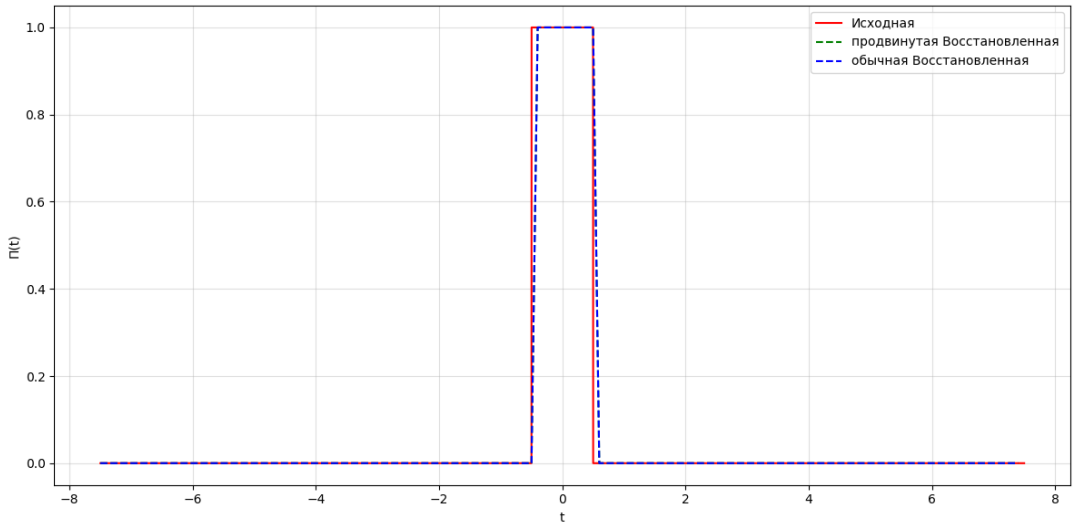

Рис. 39: Сравнение функций $T = 15, dT = 0.1$

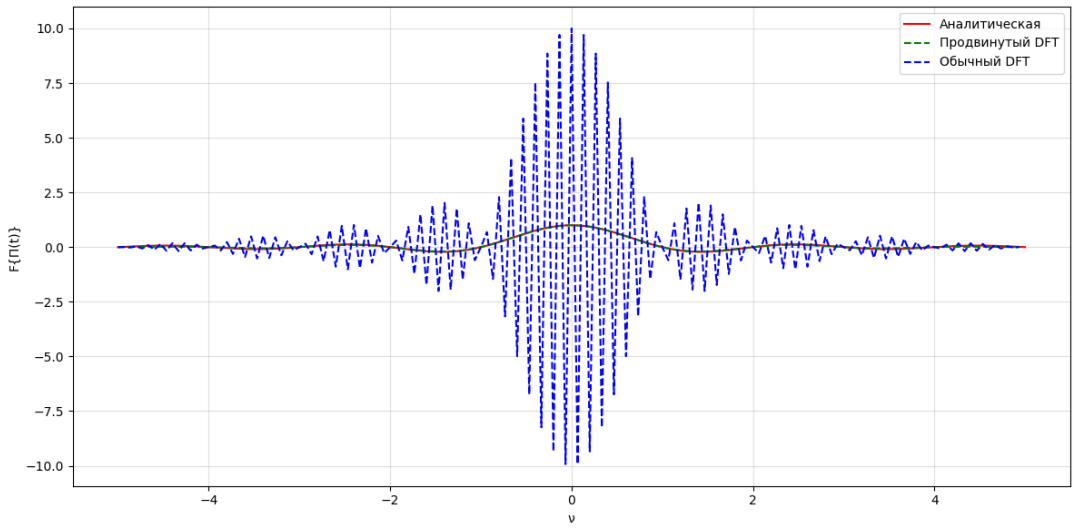

Рис. 40: Сравнение фурье-образов $T = 15, dT = 0.1$

Можем заметить, что результат во временной области почти идентичен, но зато в частотной области мы наконец то имеем здоровую картину, схожую с оригиналом.

Выводы по заданию 1

В этом задании я вдоволь поработал с численными способами преобразований Фурье. Главные выводы что получилось сделать в общем - при работе с численным преобразованием Фурье важно соблюдать баланс. С одной стороны обязательно сохранить планку минимум - условие Найквиста-Шеннона и удобоваримая точность,но так же важно и не уйти в слишком высокую точность - с уменьшением Δt и увеличением T время работы увеличивается пропорционально.

Задание 2. Семплирование.

Имеем следующие функции:

$$y_1(t) = a_1 \sin(w_1 t + \phi_1) + a_2 \sin(w_2 t + \phi_2$$
$$y_2(t) = \text{sinc}(bt)$$

Зададим им параметры:

$$a_1 = 1, \quad a_2 = 2, \quad \omega_1 = 6, \quad \omega_2 = 2, \quad \phi_1 = 1, \quad \phi_2 = 0, \quad b = 3$$

Заметим, что приведенные функции являются тригонометрическими в своей сути, потому можем допустить предположение что основная информация в образе функции содержится в некотором промежутке B от центра. Потому можем ввести таким образом критерий теоремы Найквиста-Шеннона-Котельникова, по которому при семплировании функции с частотой $\Delta t < \frac{1}{2B}$, мы сможем восстановить оригинальную функцию.

Для этого воспользуемся приближением по формулам:

$$y_s = \sum_{n=-\infty}^{\infty} f(t_n) \cdot \text{sinc}(2B(t - t_n)) \qquad y_s = \sum_{n=-\infty}^{\infty} f(t_n) \cdot \text{sinc}\left(\frac{t - t_n}{\Delta t}\right)$$

Учитывая заданные параметры для функции, параметр B численно равен 2. Поэтому критическое значение Δt соответсвенно равно $\Delta t = \frac{1}{2B} = 0.25$ Для большой точности надо взять крупные значения параметров T, N. Пускай они будут $T = 250, N = 10000$. Теперь протестируем разные значения Δt что бы проверить как будет выполняться условие теоремы.

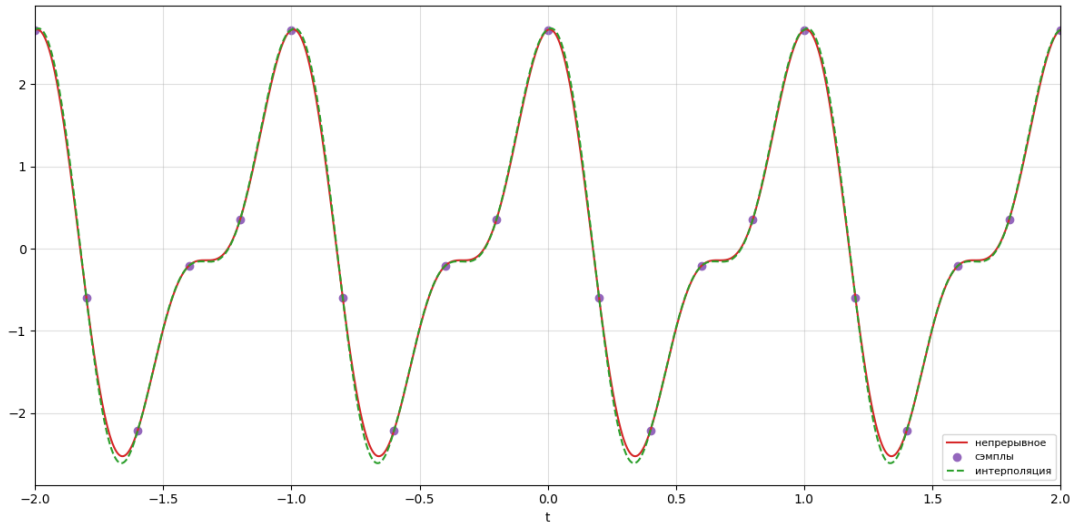

Рис. 41: Анализ семплирования (функция) $dT = 0.2$

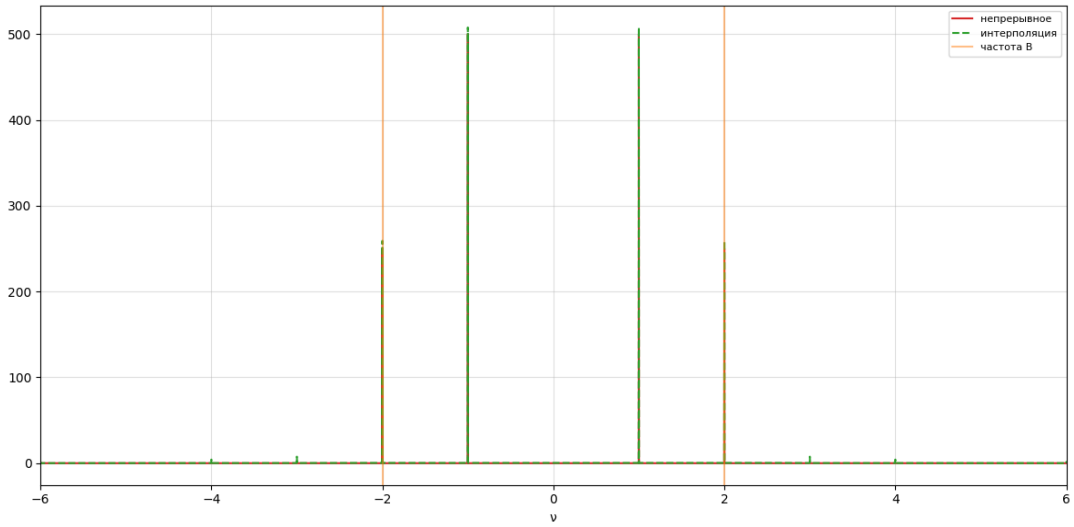

Рис. 42: Анализ семплирования (образ) $dT = 0.2$

Результат прекрасный - функция точно воспроизведена и накладывается на оригинал почит полностью. Попробуем теперь выбрать какое нибудь приграничное значение.

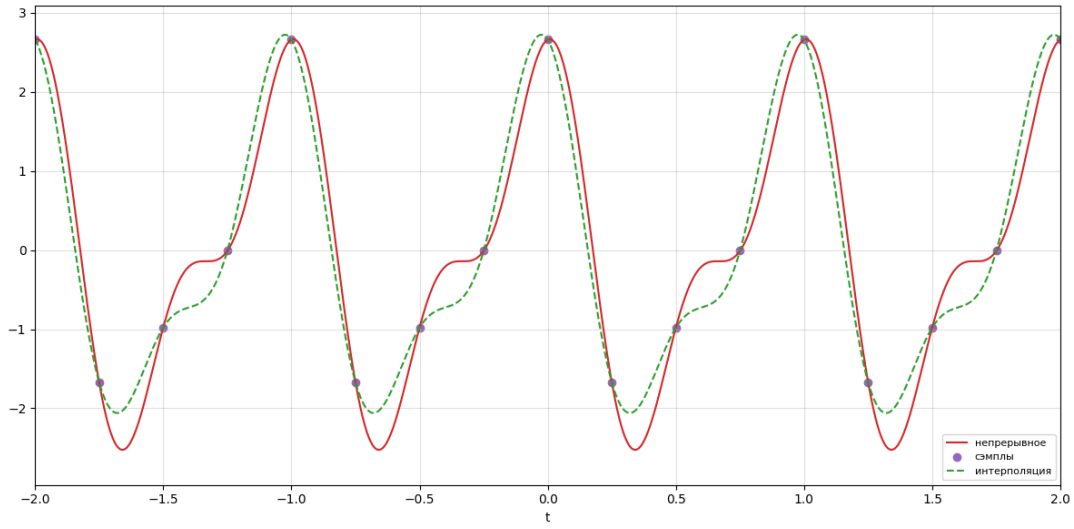

Рис. 43: Анализ семплирования (функция) $dT = 0.25$

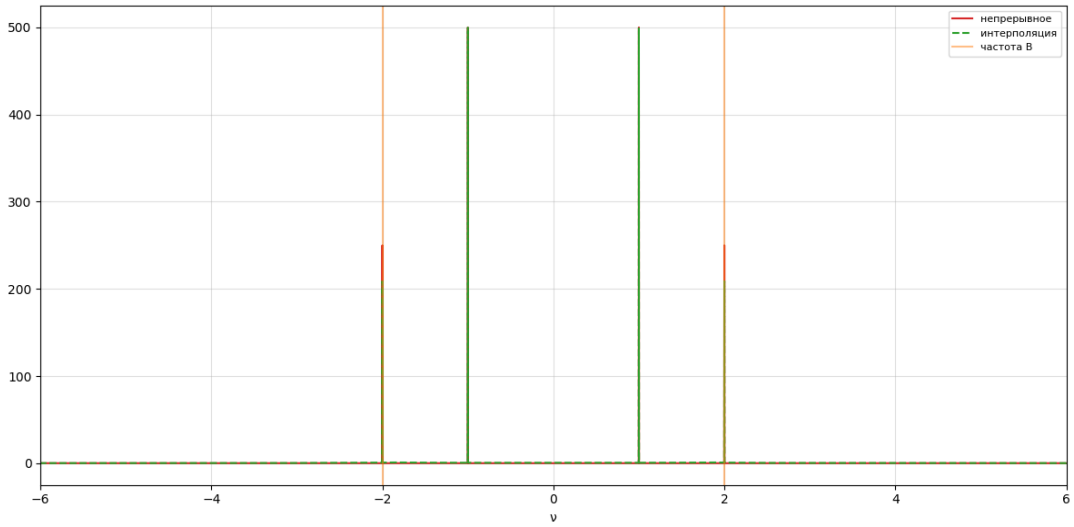

Рис. 44: Анализ семплирования (образ) $dT = 0.25$

Параметр стоит на границе, но результат не самый лучший. Все из за не самого точного численного вычисления интерполяции. Показательный пример что нужно делать поправки всегда. Интереса ради запустим код с очень большим значением Δt.

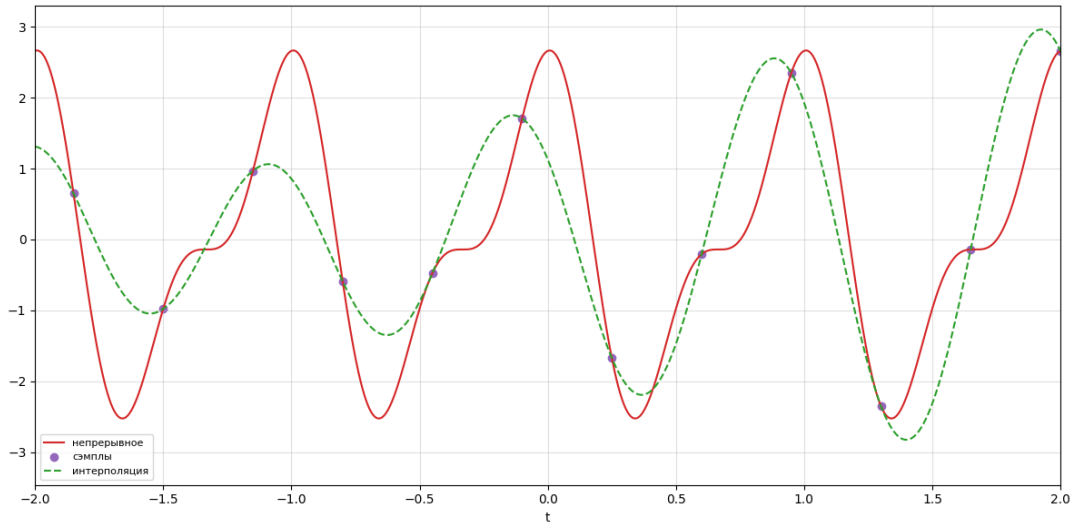

Рис. 45: Анализ семплирования (функция) $dT = 0.35$

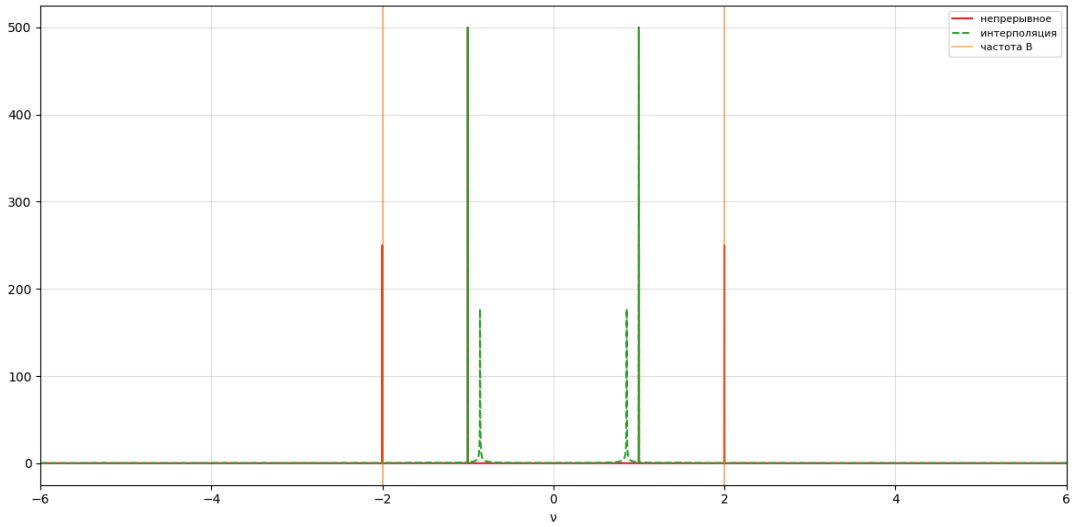

Рис. 46: Анализ семплирования (образ) $dT = 0.35$

Результат ужасен. На лицо сильные различия между оригиналом и интерполяцией.

Возьмемся за вторую функцию. Для нее параметр B будет равен 1.5, можем это говорить уверенно благодаря прямоугольной природе образа функции. Протестируем вновь.

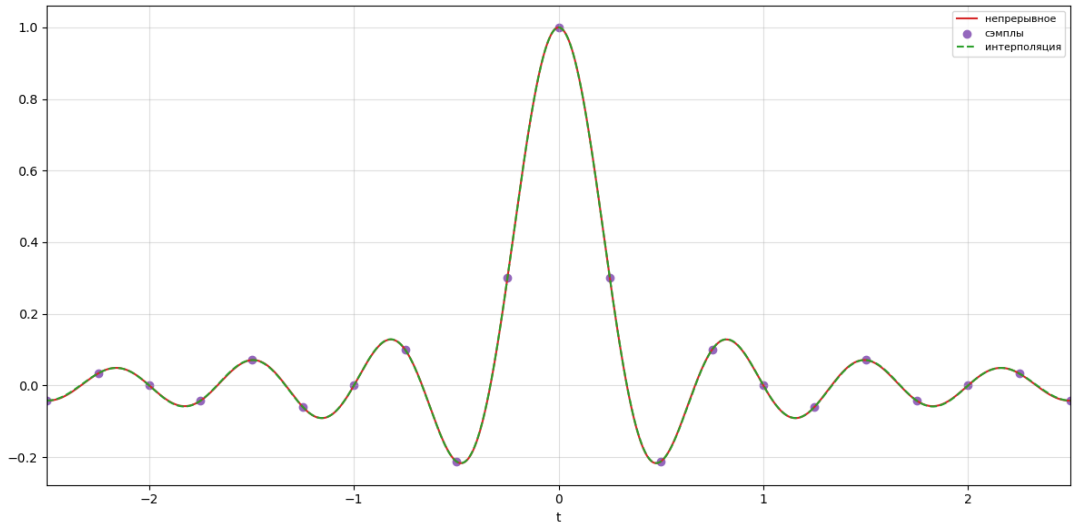

Рис. 47: Анализ семплирования (функция) $dT = 0.25$

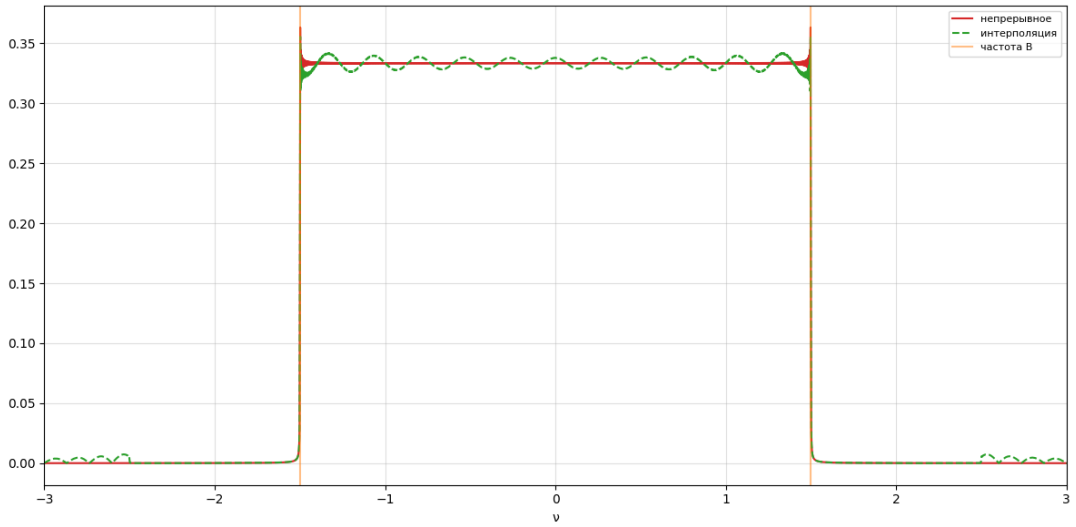

Рис. 48: Анализ семплирования (образ) $dT = 0.25$

Приближение хорошее, функция наложена почти идеально. Фурье образ немного отличается, но в совсем нестрашной форме. Пробуем теперь граничное значение

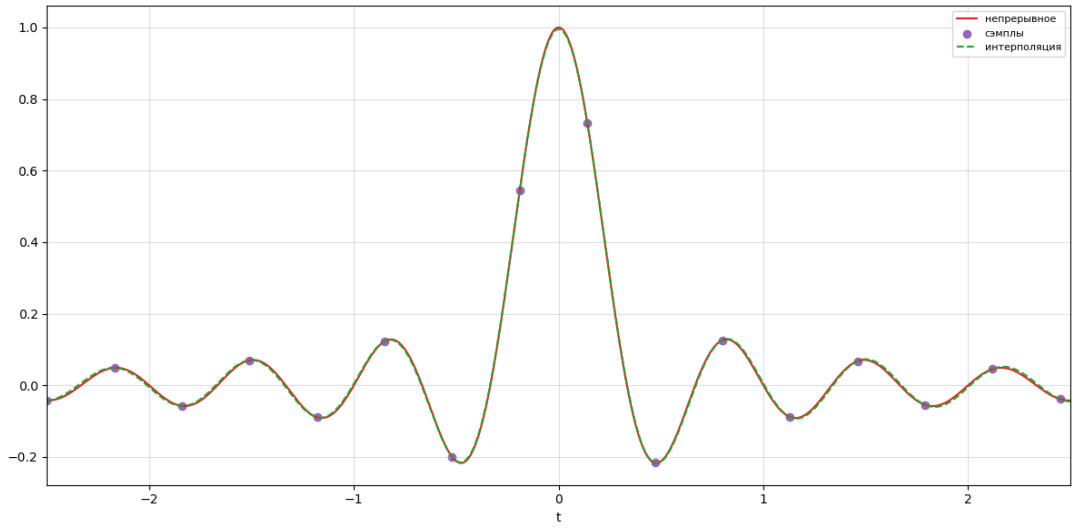

Рис. 49: Анализ семплирования (функция) $dT = 0.33$

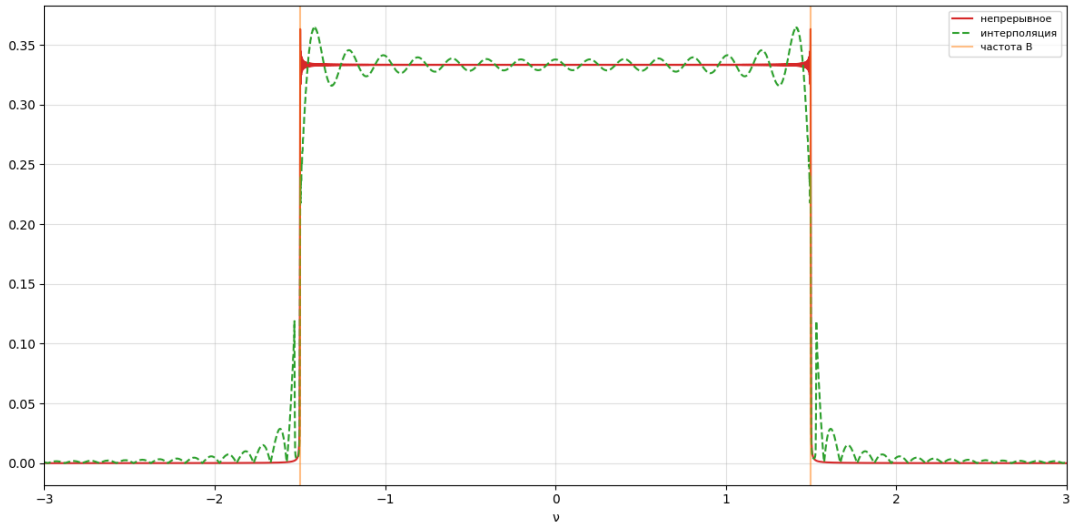

Рис. 50: Анализ семплирования (образ) $dT = 0.33$

Даже так - результат хороший. Уже теперь видны некоторые расхождения с оригиналом, но по прежнему очень точно. На выгоду приближению сыграл тот факт, что 0.33 это не самое точно приближения числа $\frac{1}{3}$. Теперь посмотрим на совсем большое значение.

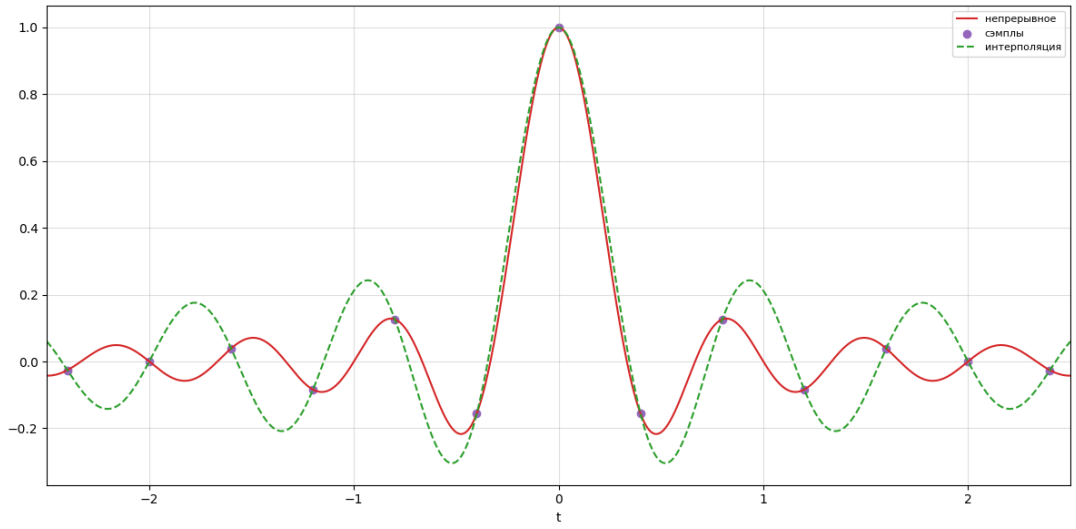

Рис. 51: Анализ семплирования (функция) $dT = 0.4$

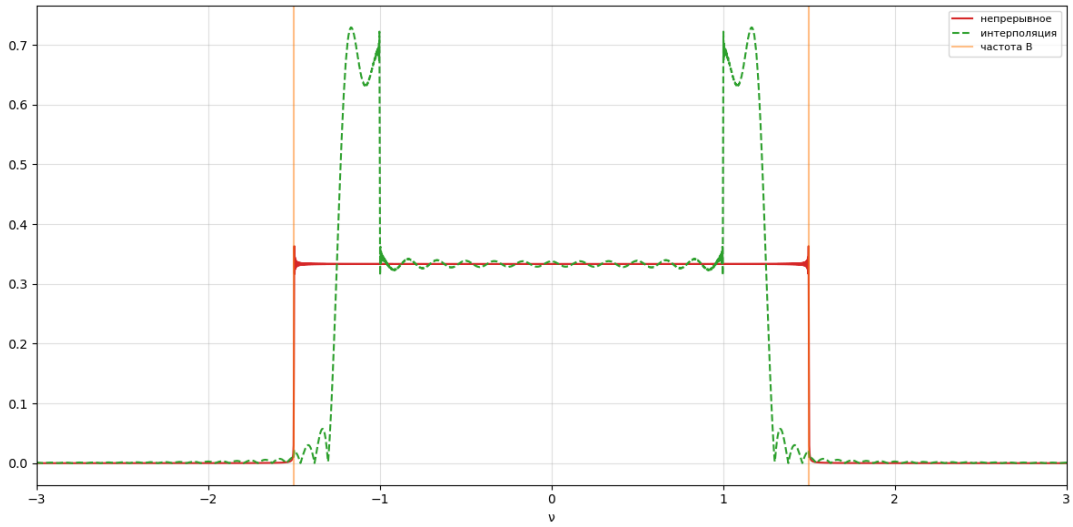

Рис. 52: Анализ семплирования (образ) $dT = 0.4$

Что и ожидалось - без соблюдения условия Найквиста-Шеннона-Котельникова результат не может быть адекватным. Функция слишком «уплыла» от начального положения, а ее фурье образ совсем не похож на оригинал, что очень любопытно как эффект.

Выводы по заданию 2 В рамках второго задания я вновь утвердился в важности теоремы Найквиста-Шеннона-Котельникова, посмотрел на практике тонкости работы с пограничным значениями дискретизации и наконец получше узнал о практике семплирования сигналов.

Общий вывод

В ходе второй лабораторной работы я углубился в работе с DFT. Ознакомился с его реализацией, улучшениями и потенциальными слабостями(строить графики было долгим занятием). Отдельно углубился в теорему Найквиста-Шеннона-Котельникова, теперь наглядно понял почему она так критически важна.

Приложение

```python
import numpy as np
import matplotlib.pyplot as plt

def rect_f(t):
    return np.where(np.abs(t) <= 0.5, 1.0, 0.0)
T = 5
V = 10
dt = 0.01
dnu = 0.01

t = np.arange(-T/2, T/2, dt)
nu = np.arange(-V/2, V/2, dnu)
f_t = rect_f(t)

dt_perf = 0.001
dnu_perf = 0.01
params = [
    # (5, 0.2, 10, 0.1),
    # (20, 0.1, 10, 0.1),
    # (5, 0.01, 10, 0.1),
    # (5, 0.01, 40, 0.1),
    (15, 0.1, 40, 0.1)
]
for param in params:
    T, dt, V, dnu = param
    t = np.arange(-T/2, T/2, dt)
    N = len(t)
    V = max(abs(nu)) * 2
    t_perf = np.arange(-T/2, T/2, dt_perf)
    f_t = rect_f(t)

    nu1 = np.arange(-V/2, V/2, dnu)
    nu_perf = np.arange(-V/2, V/2, dnu_perf)
    f_nu1 = np.array([
        np.trapz(f_t * np.exp(-2j * np.pi * n * t), t) for n in nu
    ])

    rec_t1 = np.array([
        np.trapz(f_nu1 * np.exp(2j * np.pi * t_i * nu), nu).real for t_i in t
    ])

    plt.figure(figsize=(12, 6))
    plt.plot(nu_perf, np.sinc(nu_perf),'r', label='Аналитическая')
    plt.plot(nu, np.real(f_nu1), 'g--', label='Численная')
    plt.xlabel("  ")
    plt.ylabel("F{П(t)}")
    plt.grid(alpha=0.5)
    plt.legend()
    plt.xlim(-5, 5)
    plt.savefig(f'1.1/FI_1.1[T={T},dt={dt},V={V},dnu={dnu}].png')
    plt.show()
    plt.figure(figsize=(12, 6))
    plt.plot(t_perf, rect_f(t_perf),'r', label='Исходная')
    plt.plot(t, rec_t1, 'g--', label='Восстановленная')
    plt.xlabel("t")
    plt.ylabel("  (t)")
    plt.grid(alpha=0.5)
    plt.legend()
    plt.xlim(-2.5, 2.5)
    plt.savefig(f'1.1/FR_1.1[T={T},dt={dt},V={V},dnu={dnu}].png')
    plt.show()

params = [
    (15, 0.1),
    # (45, 0.05),
    # (5, 0.05),
    # (15, 0.1),
    # (15, 0.02),
]
```

```
dt_perf = 0.001
for param in params:

    T, dt = param
    t = np.arange(-T/2, T/2, dt)
    t_perf = np.arange(-T/2, T/2, dt_perf)
    N = len(t)
    f_t = rect_f(t)
    t0 = -T/2
    nu = np.fft.fftshift(np.fft.fftfreq(N, dt))
    f_nu = np.fft.fftshift(np.fft.fft(f_t, norm='ortho'))
    dnu = 1 / (N * dt)
    V = max(abs(nu)) * 2
    nu_perf = np.arange(-V/2, V/2, dt_perf)

    rec_ft = np.fft.ifft(np.fft.ifftshift(f_nu), norm='ortho')

    plt.figure(figsize=(12, 6))
    plt.plot(nu, np.real(f_nu), 'g--',label='Численная')
    plt.plot(nu_perf, np.sinc(nu_perf),'r', label='Аналитическая')
    plt.ylabel("F(П(t))")
    plt.xlabel("  ")
    plt.grid(alpha=0.4)
    plt.legend()
    plt.savefig(f'1.2/FI_1.2[T={T},dt={dt}].png')
    plt.show()

    plt.figure(figsize=(12, 6))
    plt.plot(t, rec_ft.real, 'g--', label='Восстановленная')
    plt.plot(t_perf, rect_f(t_perf), 'r', label='Исходная')
    plt.ylabel("П(t)")
    plt.xlabel("t")
    plt.grid(alpha=0.4)
    plt.legend()
    plt.xlim(-2.5, 2.5)
    plt.savefig(f'1.2/FF_1.2[T={T},dt={dt}].png')
    plt.show()

    params = [
    (15, 0.1),
    # (8, 0.05),
    # (30, 0.05),
    # (15, 0.15),
    # (15, 0.01),
]
dt_perf = 0.001
for param in params:
    T, dt = param
    t = np.arange(-T/2, T/2, dt)
    t_perf = np.arange(-T/2, T/2, dt_perf)
    N = len(t)
    f_t = rect_f(t)
    t0 = -T/2
    nu = np.fft.fftshift(np.fft.fftfreq(N, dt))
    f_nu = np.fft.fftshift(np.fft.fft(f_t))
    f_nu_smart = f_nu * dt * np.exp(-2j * np.pi * nu * t0)
    dnu = 1 / (N * dt)
    V = max(abs(nu)) * 2
    nu_perf = np.arange(-V/2, V/2, dt_perf)

    rec_ft = np.fft.ifft(np.fft.ifftshift(f_nu))
    rec_good_ft = np.fft.ifft(np.fft.ifftshift(f_nu_smart * np.exp(2j * np.pi * nu * t0))) / dt

    plt.figure(figsize=(12, 6))
    plt.plot(nu_perf, np.sinc(nu_perf),'r', label='Аналитическая')
    plt.plot(nu, np.real(f_nu_smart), 'g--', label='Продвинутый DFT')
    plt.plot(nu, np.real(f_nu), 'b--', label='Обычный DFT')
    plt.xlabel("  ")
    plt.ylabel("F{П(t)}")
    plt.grid(alpha=0.4)

```

```
    plt.legend()
    plt.savefig(f'1.3/num_vs_analytic_transform_1.4[T={T},dt={dt}].png')
    plt.show()

    plt.figure(figsize=(12, 6))
    plt.plot(t_perf, rect_f(t_perf), 'r', label='Исходная')
    plt.plot(t, rec_good_ft.real, 'g--', label='продвинутая Восстановленная')
    plt.plot(t, rec_ft.real, 'b--', label='обычная Восстановленная')

    plt.xlabel("t")
    plt.ylabel("П(t)")
    plt.grid(alpha=0.4)
    plt.legend()

    plt.savefig(f'1.3/orig_vs_reconst_function_1.4[T={T},dt={dt}].png')
    plt.show()

def y1(t, a1=1.0, a2=2, f1=2, f2=1, phi1=1, phi2=2):
    w1, w2 = 2*np.pi*f1, 2*np.pi*f2
    return a1*np.sin(w1*t + phi1) + a2*np.sin(w2*t + phi2)

def y2(t, b=3):
    return np.sinc(b*t)

def interp_sinc(y_samp, t_samp, t_query, nsupport=50):
    T = t_samp[1] - t_samp[0]
    out = np.zeros_like(t_query)
    idx = np.searchsorted(t_samp, t_query)
    n_samp = len(t_samp)
    for i, tq in enumerate(t_query):
        start = max(0, idx[i] - nsupport//2)
        end = min(n_samp, idx[i] + nsupport//2)
        dt = tq - t_samp[start:end]
        out[i] = np.sum(y_samp[start:end] * np.sinc(dt / T))
    return out

def good_fft(sig, dt):
    return np.fft.fftshift(np.fft.fft(np.fft.ifftshift(sig))) * dt

T = [250]
N = 10000
dt_list1 = [0.25]
dt_perf = 0.001

B1 = 2
B2 = 1.5

def expand(t_samp, y_samp, t_perf, dt_perf):
    sig = np.zeros_like(t_perf)
    idx = ((t_samp - t_perf[0]) / dt_perf).astype(int)
    sig[idx] = y_samp
    return sig

for T_full in T:
    t_fine = np.arange(-T_full, T_full, dt_perf)
    y1_fine = y1(t_fine)
    nu_fine = np.fft.fftshift(np.fft.fftfreq(len(t_fine), d=dt_perf))
    Y1_fine = good_fft(y1_fine, dt_perf)

    for dt_samp in dt_list1:
        t_samp = np.arange(-T_full, T_full, dt_samp)
        y1_samp = y1(t_samp)
        y1_interp = interp_sinc(y1_samp, t_samp, t_fine, nsupport=30)
        plt.figure(figsize=(12,6))
        plt.plot(t_fine, y1_fine, color='tab:red', label="непрерывное")
        plt.scatter(t_samp, y1_samp, color='tab:purple', label="сэмплы")
        plt.plot(t_fine, y1_interp, '--', color='tab:green', label="интерполяция")
        plt.xlabel("t")
        plt.grid(alpha=0.4)
        plt.legend(fontsize=8)
        plt.xlim(-2, 2)
```

```
        plt.savefig(f'2/o_vs_f(dt={dt_samp}).png')
        plt.show()
        Y1_samp = good_fft(expand(t_samp, y1_samp, t_fine, dt_perf), dt_perf)
        Y1_interp = good_fft(y1_interp, dt_perf)

        plt.figure(figsize=(12,6))
        plt.plot(nu_fine, np.abs(Y1_fine), color='tab:red', label="непрерывное")
        plt.plot(nu_fine, np.abs(Y1_interp), '--', color='tab:green', label="интерполяция")
        plt.axvline(B1, color='tab:orange', label='частота B', alpha=0.5); plt.axvline(-B1, color='tab:orange', alpha=0.5)
        plt.xlim(-6, 6)
        plt.xlabel("  ")
        plt.grid(alpha=0.4)
        plt.legend(fontsize=8)
        plt.savefig(f'2/num_vs_an(dt={dt_samp}).png')
        plt.show()
dt_list2 = [0.25, 0.33, 0.4]
for T_full in T:
    t_fine = np.arange(-T_full, T_full, dt_perf)
    y2_fine = y2(t_fine)
    nu_fine = np.fft.fftshift(np.fft.fftfreq(len(t_fine), d=dt_perf))
    Y2_fine = good_fft(y2_fine, dt_perf)
    for dt_samp in dt_list2:
        t_samp = np.arange(-T_full, T_full, dt_samp)
        y2_samp = y2(t_samp)
        y2_interp = interp_sinc(y2_samp, t_samp, t_fine, nsupport=30)
        plt.figure(figsize=(12,6))
        plt.plot(t_fine, y2_fine, color='tab:red', label="непрерывное")
        plt.scatter(t_samp, y2_samp, color='tab:purple', label="сэмплы")
        plt.plot(t_fine, y2_interp, '--', color='tab:green', label="интерполяция")
        plt.xlabel("t")
        plt.grid(alpha=0.4)
        plt.legend(fontsize=8)
        plt.xlim(-2.5, 2.5)
        plt.savefig(f'2/o_vs_function(dt={dt_samp}).png')
        plt.show()
        Y2_samp = good_fft(expand(t_samp, y2_samp, t_fine, dt_perf), dt_perf)
        Y2_interp = good_fft(y2_interp, dt_perf)
        plt.figure(figsize=(12,6))
        plt.plot(nu_fine, np.abs(Y2_fine), color='tab:red', label="непрерывное")
        plt.plot(nu_fine, np.abs(Y2_interp), '--', color='tab:green', label="интерполяция")
        plt.axvline(B2, color='tab:orange', label='частота B', alpha=0.5); plt.axvline(-B2, color='tab:orange', alpha=0.5)
        plt.xlim(-3, 3)
        plt.xlabel("  ")
        plt.grid(alpha=0.4)
        plt.legend(fontsize=8)
        plt.savefig(f'2/num_vs_an(dt={dt_samp}).png')
        plt.show()
```