ИТМО

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ
по лабораторной работе №5

Связь непрерывного и дискретного

Автор: Дженжеруха Кирилл
Учебная группа: R3242
Номер ИСУ: 368103
Преподаватель: Перегудин Алексей Алексеевич

Санкт-Петербург
2024

Содержание

Задание 1. Непрерывное и дискретное преобразование Фурье **3**

Задание 2. Сэмплирование **7**

Ум ценится дорого, когда дешевеет сила.
Джейсон Стетхэм

Эта лабораторная работа посвящена исследованию дискретных преобразований Фурье и сэмплированию сигналов.

Задание 1.
Непрерывное и дискретное преобразование Фурье

Для начала работы с непрерывным и дискретным преобразованиями Фурье, зададим прямоугольную функцию:

$$\Pi(t) = \begin{cases} 1, & |t| \leq 1/2, \\ 0, & |t| > 1/2; \end{cases} : \mathbb{R} \to \mathbb{R}.$$

Найдем аналитическое выражение для Фурье-образа заданной прямоугольной функции:

$$\hat{\Pi}(\nu) = \int_{-\infty}^{\infty} \Pi(t) e^{-2\pi i \nu t}\, dt = \frac{\sin \pi\nu}{\pi\nu}.$$

Построим графики исходной функции и ее аналитического Фурье-образа:

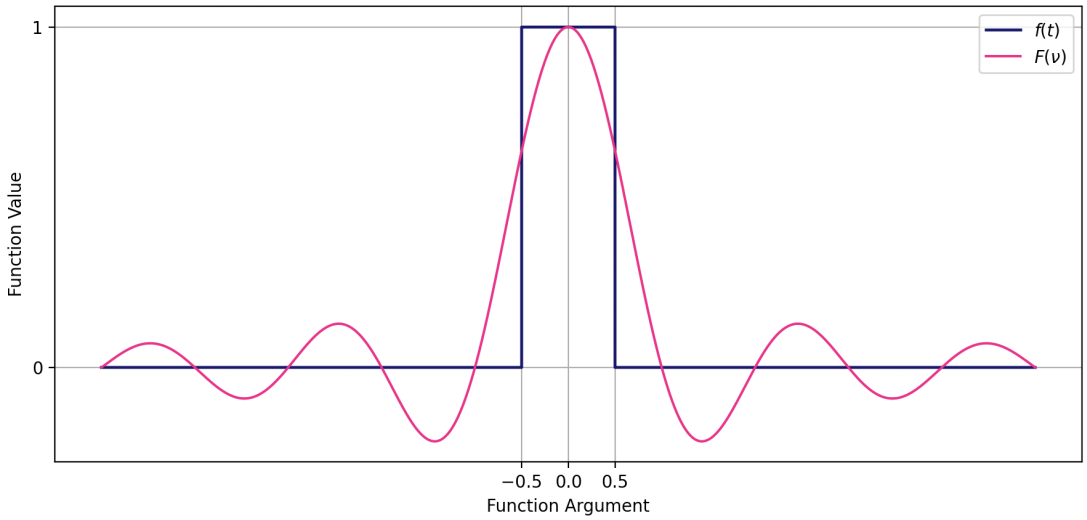


Рис. 1 – График исходной функции и аналитического Фурье-образа.

Следующим шагом выполним численное интегрирование заданной исходной функции и построим графики численного Фурье-образа, а также восстановленной и исходной функций:

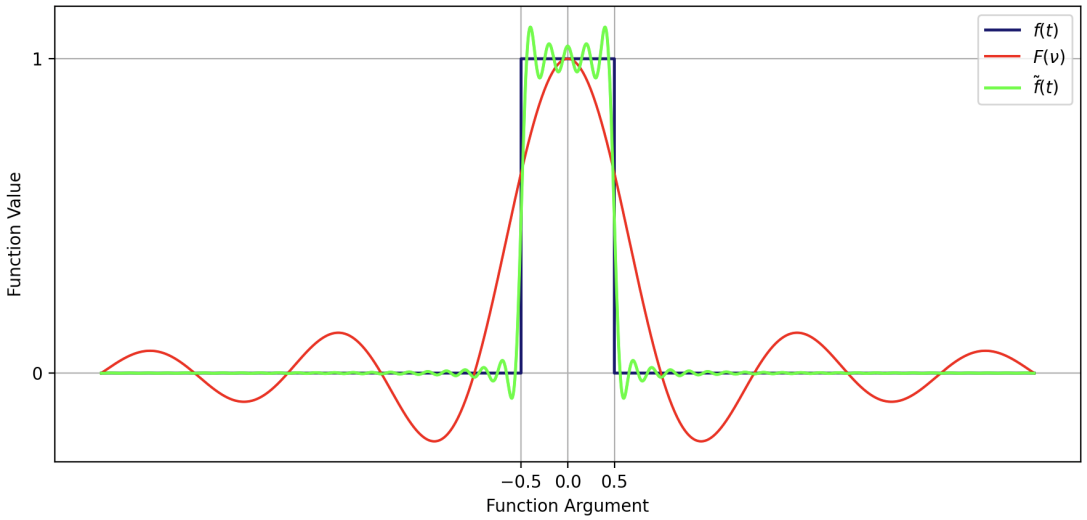


Рис. 2 – График численного Фурье-образа и восстановленной функции.

Исследуем, как величина шага интегрирования и размер промежутка интегрирования влияют на результаты преобразования и восстановления. Для этого вполне достаточно построить два дополнительных графика – на одном из них изменить шаг интегрирования, на другом – промежуток интегрирования:

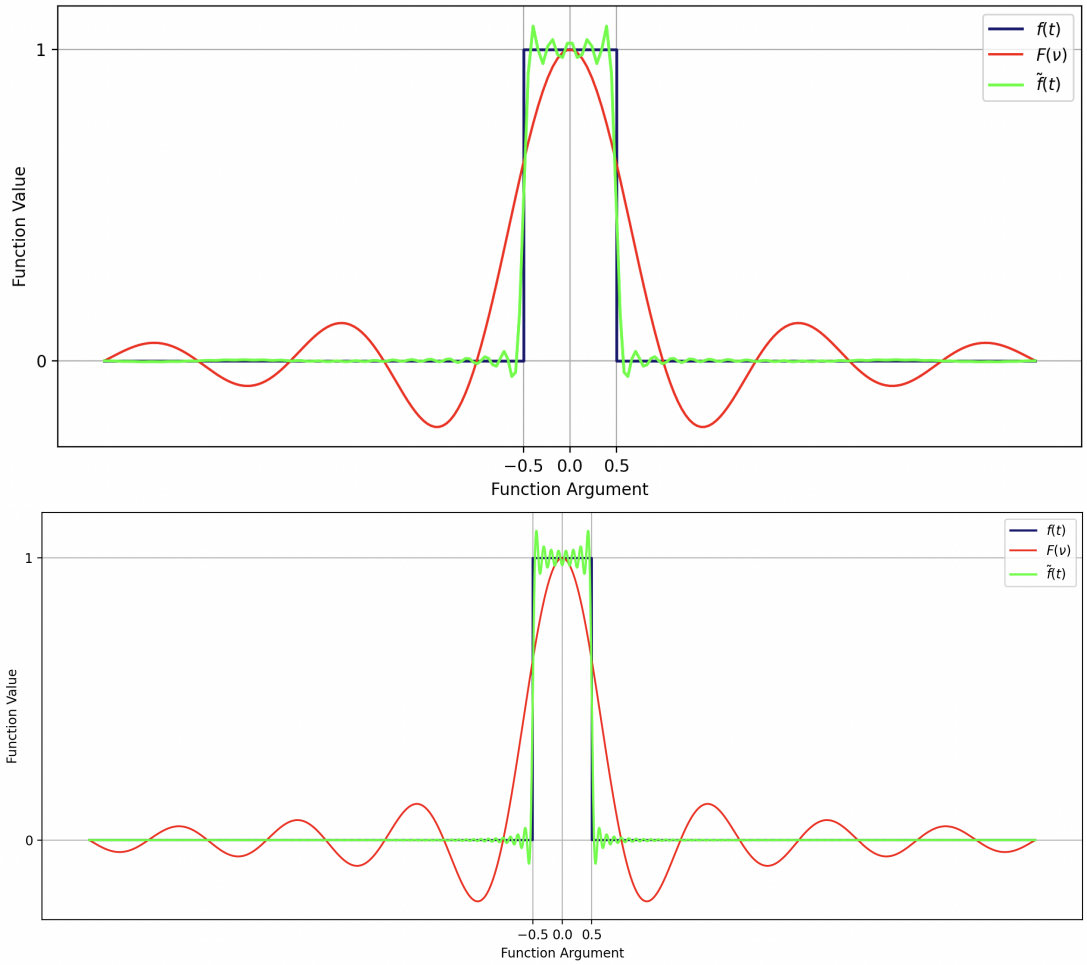


Рис. 3 – Графики видоизмененного численного Фурье-образа и восстановленной функции (сверху – увеличен шаг интегрирования, снизу – увеличен промежуток интегрирования).

Сделаем промежуточные выводы: метод численного интегрирования ввиду большого количества обрабатываемых данных работает сравнительно медленно – если шаг интегрирования достаточно мал (что следует из большого количества обрабатываемых данных), вычислительная сложность алгоритма, описывающего метод трапеций, больше, чем у алгоритма DFT, который будет рассмотрен на следующем шаге.

Приведем листинг кода, позволяющего выполнить численное преобразование Фурье:

Листинг 1. Численное преобразование Фурье

```
time = np.concatenate((np.linspace(-5, -0.5, 5500), np.linspace(-0.5, 0.5,
    1000), np.linspace(0.5, 5, 5500)))
value = np.concatenate(([0 for i in range(5500)], [1 for i in range(1000)
   ], [0 for i in range(5500)]))

nu = np.array([t for t in time])
num_image = np.array([])
for freq in nu:
    num_image_value = np.trapz(value * np.exp(2 * np.pi * complex(0, -1) *
        freq * time), time)
    num_image = np.append(num_image, num_image_value)

```

```
    num_recovered_func = np.array([])
    for t in time:
        recovered_func_value = np.trapz(num_image * np.exp(2 * np.pi * complex
            (0, 1) * nu * t), nu)
        num_recovered_func = np.append(num_recovered_func,
            recovered_func_value)
```

Теперь выполним преобразование Фурье с помощью алгоритма встроенного в библиотеку **SciPy** алгоритма DFT (*Discrete Furier Transformation*) и исследуем его на точность и время работы. Для сохранения унитарности преобразования будем выполнять по следующим формулам:

$$\hat{\Pi}[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{-2\pi i \frac{kn}{N}} \Pi[n]; \qquad \Pi[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i \frac{kn}{N}} \hat{\Pi}[k].$$

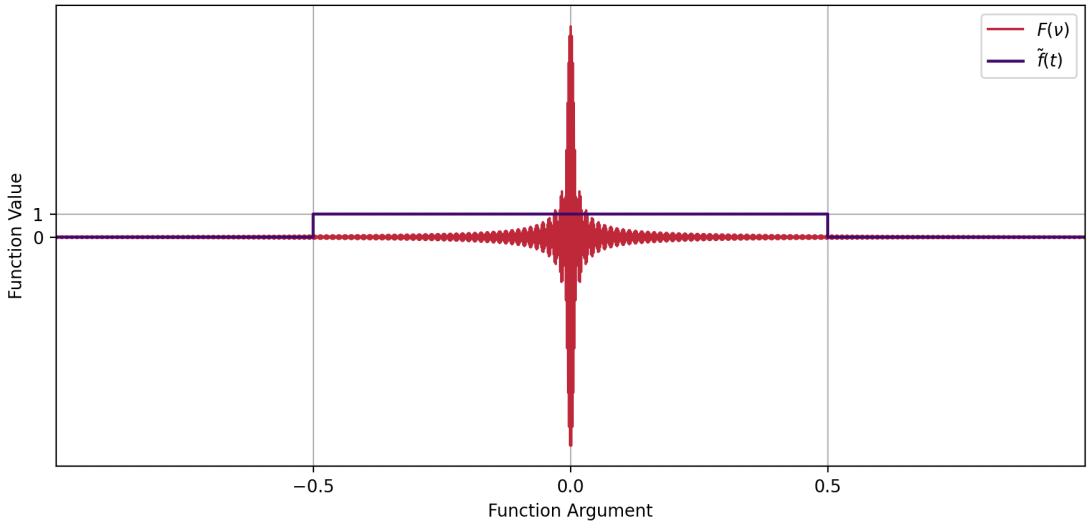


Рис. 4 – График Фурье-образа и восстановленной функции с помощью алгоритма DFT.

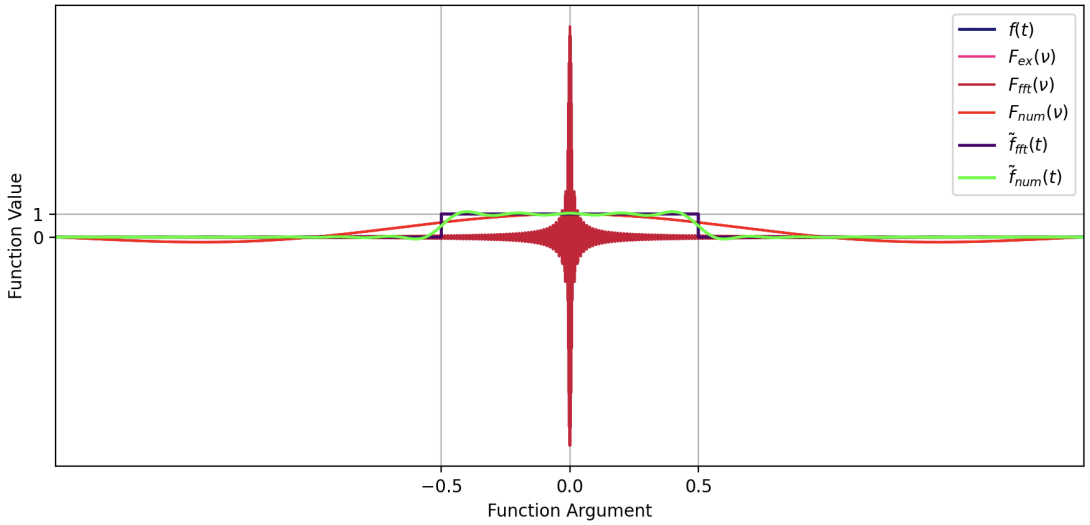


Рис. 5 – График Фурье-образа и восстановленной функции с помощью алгоритма DFT в сравнении с предыдущими методами.

Как мы видим, Фурье-образ, полученный с помощью алгоритма DFT, далек от “идеального”, но при этом обратное преобразование восстановило функцию с впечатляющей точностью. Кроме того, выполнение алгоритма заняло ощутимо меньше времени, чем использования алгоритма численного интегрирования, поскольку вычислительная сложность алгоритма DFT меньше – об этом пойдет речь далее.

Приведем листинг кода, реализующего алгоритм DFT применимо к нашей функции.

Листинг 2. Реализация алгоритма DFT.

```
time = np.concatenate((np.linspace(-5, -0.5, 5500), np.linspace(-0.5, 0.5,
    1000), np.linspace(0.5, 5, 5500)))
value = np.concatenate(([0 for i in range(5500)], [1 for i in range(1000)
    ], [0 for i in range(5500)]))

nu = np.array([t for t in time])
N = len(value)
fft_image = (1 / np.sqrt(N)) * scp.fft.fftshift(scp.fft.fft(value))
fft_recovered_func = (N / np.sqrt(N)) * scp.fft.ifft(scp.fft.ifftshift(
    fft_image))
```

Рассмотрим методы численного преобразования Фурье и быстрого дискретного преобразования Фурье в контексте их главных достоинств и недостатков. Очевидно (теперь уже – так как об этом было сказано ранее), что алгоритм быстрого преобразования FFT (или DFT) работает быстрее численного алгоритма за счет выигрыша в вычислительной сложности. За счет встроенного преобразования матрицы дискретного преобразования уменьшается количество итераций, которые алгоритм выполняет в процессе преобразования.

В то же время нетрудно заметить достаточно сильное расхождение графиков Фурье-образов при численном интегрировании и при встроенном дискретном преобразовании. Все дело – в немного разной природе численного интегрирования и алгоритма DFT. Хоть на первый взгляд они оба похожи на дискретные (иначе говоря, итеративные), численное интегрирование при достаточно большом приближении становится похожим на непрерывное преобразование (поскольку интеграл непрерывен). Для построения корректного Фурье-образа при численном интегрировании мы хотим увеличить количество отрезков разбиения, но сталкиваемся с сильным увеличением времени работы алгоритма. При этом в алгоритме DFT ситуация обратная – алгоритм полностью итеративен, и поэтому от количества итераций (а значит, и от величины шага) зависит точность восстановления функции, но не точность Фурье-образа – он далек от истины.

Можно провести следующую аналогию: мы понимаем, что в целом при численном интегрировании, начиная с какого-то момента точность перестает влиять на восприятие и анализ полученного результата (например, функция становится визуально “плавным”), а значит мы можем пренебречь той незначительной ошибкой интегрирования, так или иначе возникающей из-за недостижения супремума интегральных сумм. Логично было бы применить эти размышления и в случае с алгоритмом DFT, то есть предположить, что, начиная с какого-то числа итераций (то есть при достижении определенной величины шага), мы получим визуально идентичный результат восстановления функции и при этом близкий к правдивому Фурье-образ.

Утверждением, описывающим и подтверждающим логику этого процесса, является *теорема Найквиста-Шеннона-Котельникова*, задающая условия для восстановления непрерывного сигнала (непрерывной функции) по ее дискретным составляющим – сформулируем ее корректно.

Теорема (Найквиста-Шеннона-Котельникова)

Для безошибочного восстановления по дискретным данным функции, Фурье-образ которой лежит в промежутке $[-\alpha, \alpha]$ необходимо выполнение следующего условия для шага дискретизации Δt:

$$\Delta t < \frac{1}{2\alpha}.$$

Применив эту теорему и задав в коде из **Листинга 2** необходимый шаг дискретизации, получим следующую картину (условия: отрезок $[-30, 30]$, $\Delta t = 1/60$):

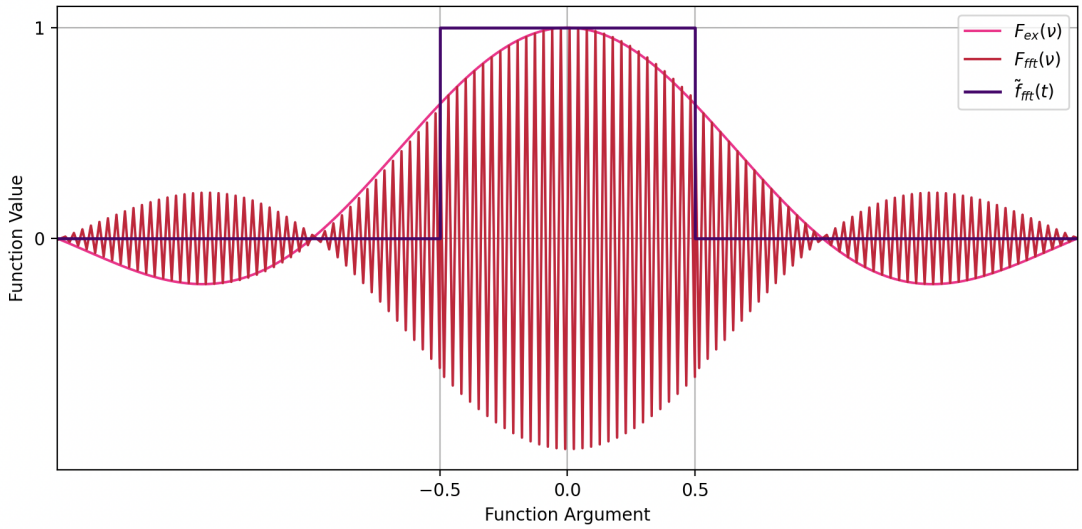


Рис. 6 – График Фурье-образа и восстановленной функции с помощью “умного” алгоритма DFT.

Следовательно, мы получили, что использованный модифицированный метод дискретного преобразования Фурье не теряет в скорости за счет внутреннего алгоритма, при этом качественно восстанавливает функцию и обеспечивает совпадение Фурье-образов. Получилось, что мы нашли оптимальное значение шага дискретизации, позволяющее быстро и точно выполнить преобразование Фурье – вычислительная сложность эквивалентна сложности алгоритма DFT (то есть $O(N \cdot \log N)$), а точность обусловлена выполнением теоремы Найквиста-Шеннона-Котельникова для шага дискретизации.

Задание 2. Сэмплирование

Зададим константы a_i, ω_i, φ_i и функцию следующего вида:

$$y(t) = a_1 \sin(\omega_1 t + \varphi_1) + a_2 \sin(\omega_2 t + \varphi_2).$$

Для значений всех констант, равных 1, задав два массива значений времени (один – для непрерывного графика, другой, разреженный – для дискретного), построим непрерывный и дискретный графики заданной функции:

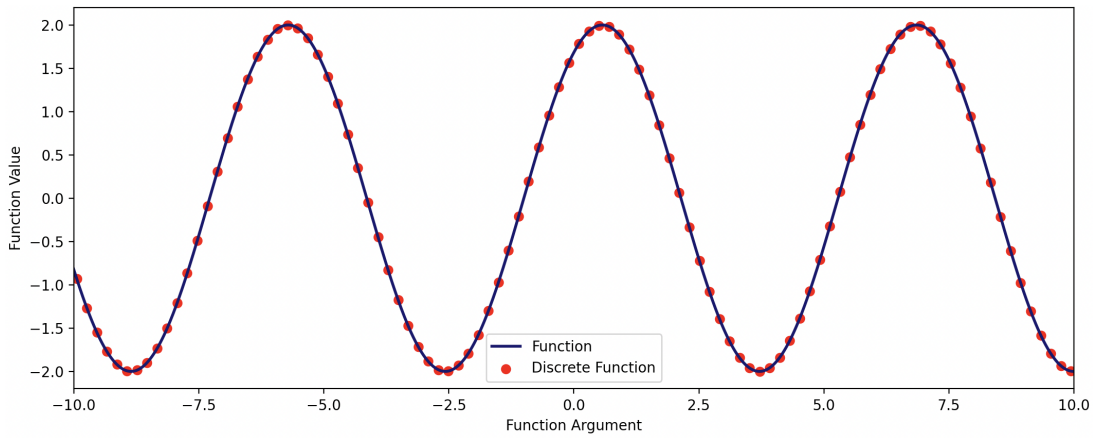


Рис. 7 – Графики (непрерывный и дискретный) заданной функции.

Попробуем применить интерполяционную формулу для того, чтобы по заданным (красным) точкам восстановить исходную функцию:

$$f(t) = \sum_{0}^{\infty} f(t_n) \cdot \operatorname{sinc}(2B(t - t_n)), \quad t_n = \frac{n}{2B}.$$

Примем значение шага дискретизации $\Delta t = 0.25$ и построим график восстановленной функции, наложив его на график исходно заданной функции:

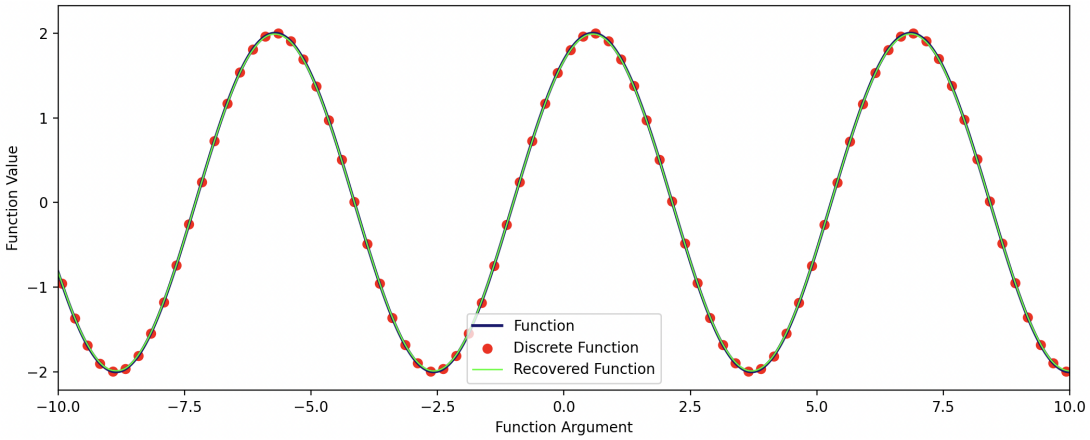


Рис. 8 – Графики восстановленной, исходной и сэмплированной функций.

Построим сравнительные графики при разных значениях Δt (шага дискретизации) и сделаем выводы о качестве восстановления функции:

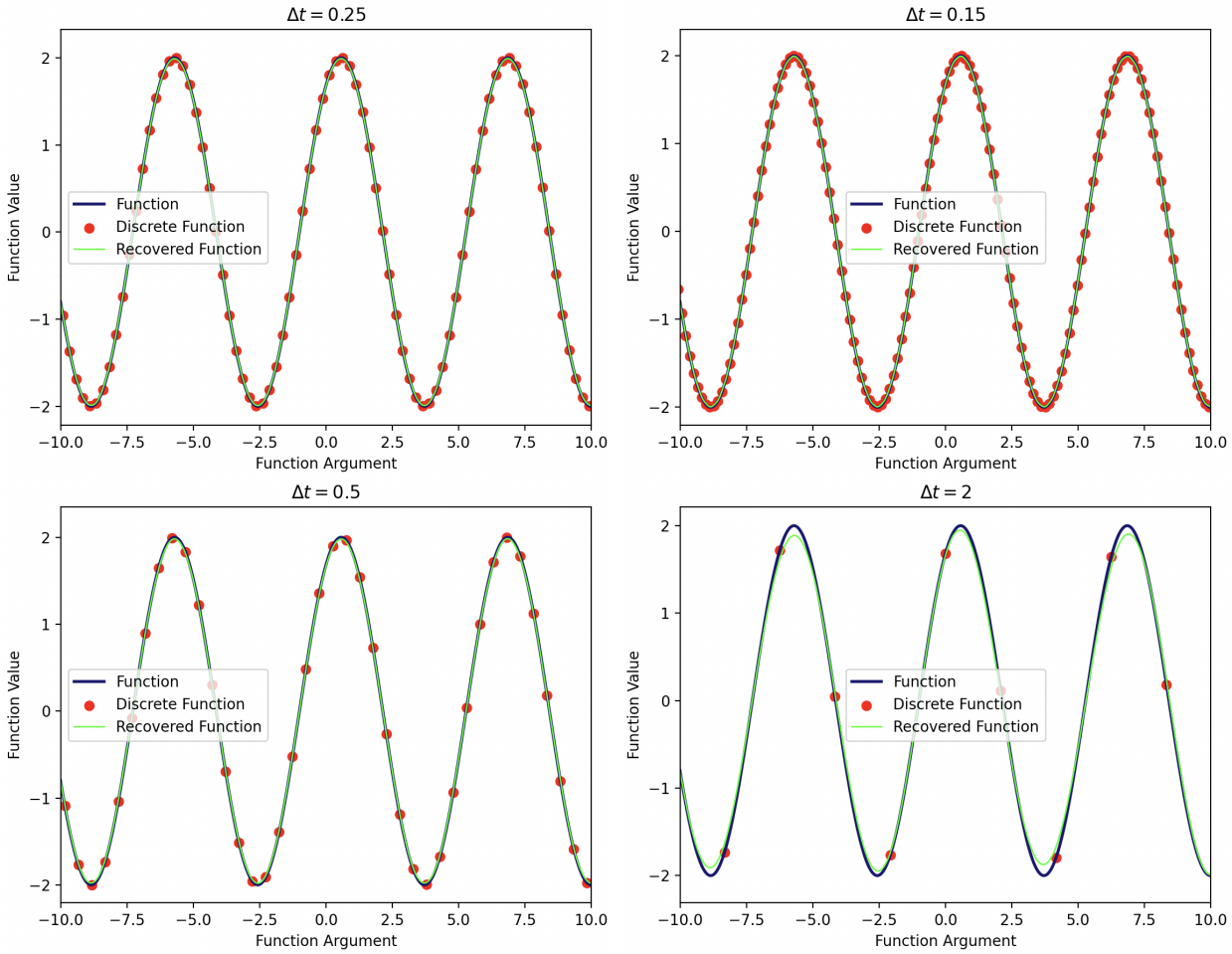


Рис. 9 – Сравнительные графики восстановления функции.

По приведенным графикам отчетливо заметно, что чем больше шаг дискретизации (то есть чем реже расположены точки), тем хуже интерполируется функция. Пользуясь записанной

ранее теоремой Найквиста-Шеннона-Котельникова, поймем, что именно при $\Delta t = 0.25$ достигается “граница применения” теоремы. Иными словами, при любом значении шага дискретизации меньшем, чем 0.25, функция будет восстановлена безошибочно точно, в то время как при больших значениях интерполяция будет недостаточно качественной.

Попробуем сэмплировать кардинальный синус вида

$$y(t) = \mathrm{sinc}(bt).$$

Для начала построим непрерывный и сэмплированный графики заданной функции для значения коэффициента $b = 2$:

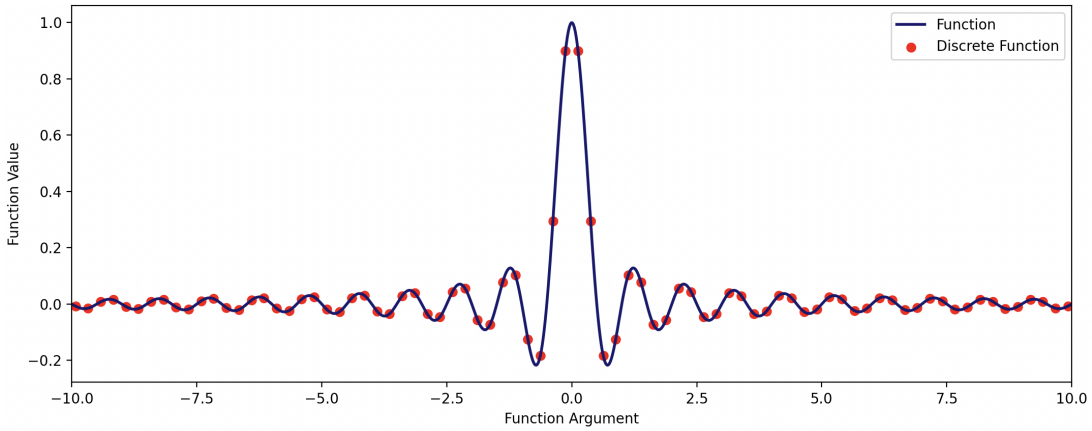


Рис. 10 – Графики (непрерывный и дискретный) заданной функции.

Аналогично предыдущему шагу, примем значение шага дискретизации $\Delta t = 0.25$ и восстановим функцию с использованием ранее приведенной интерполяционной формулы:

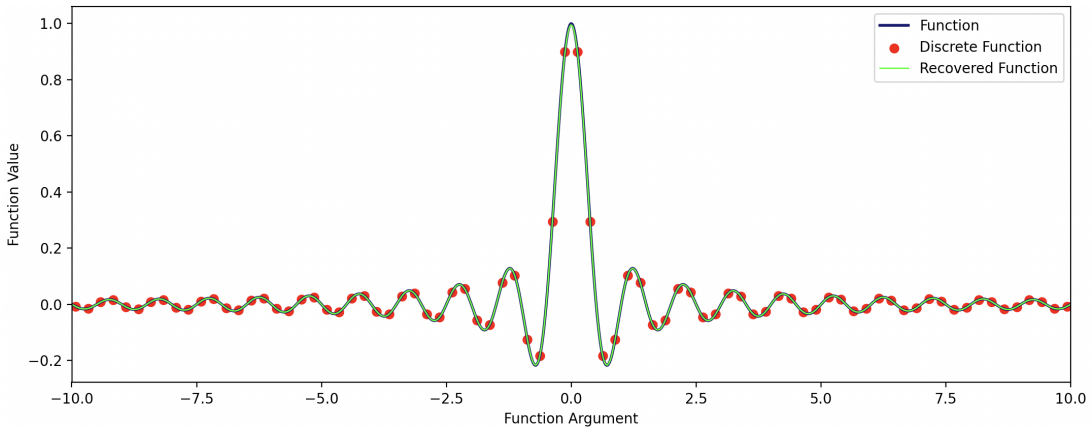
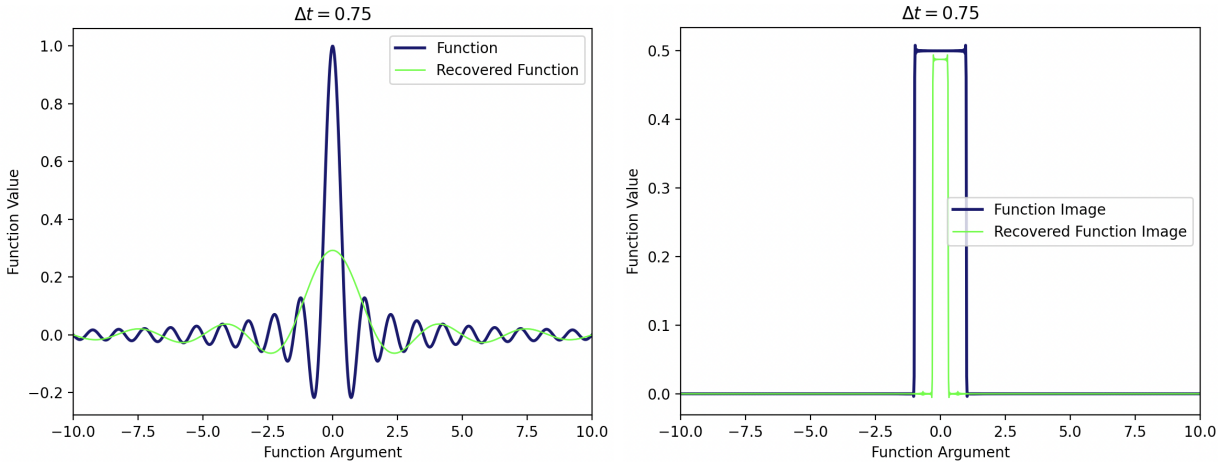


Рис. 11 – Графики восстановленной, исходной и сэмплированной функций.

Видно, что функция восстановилась достаточно хорошо. Исследуем поведения алгоритма восстановления функции для разных значений шага дискретизации:



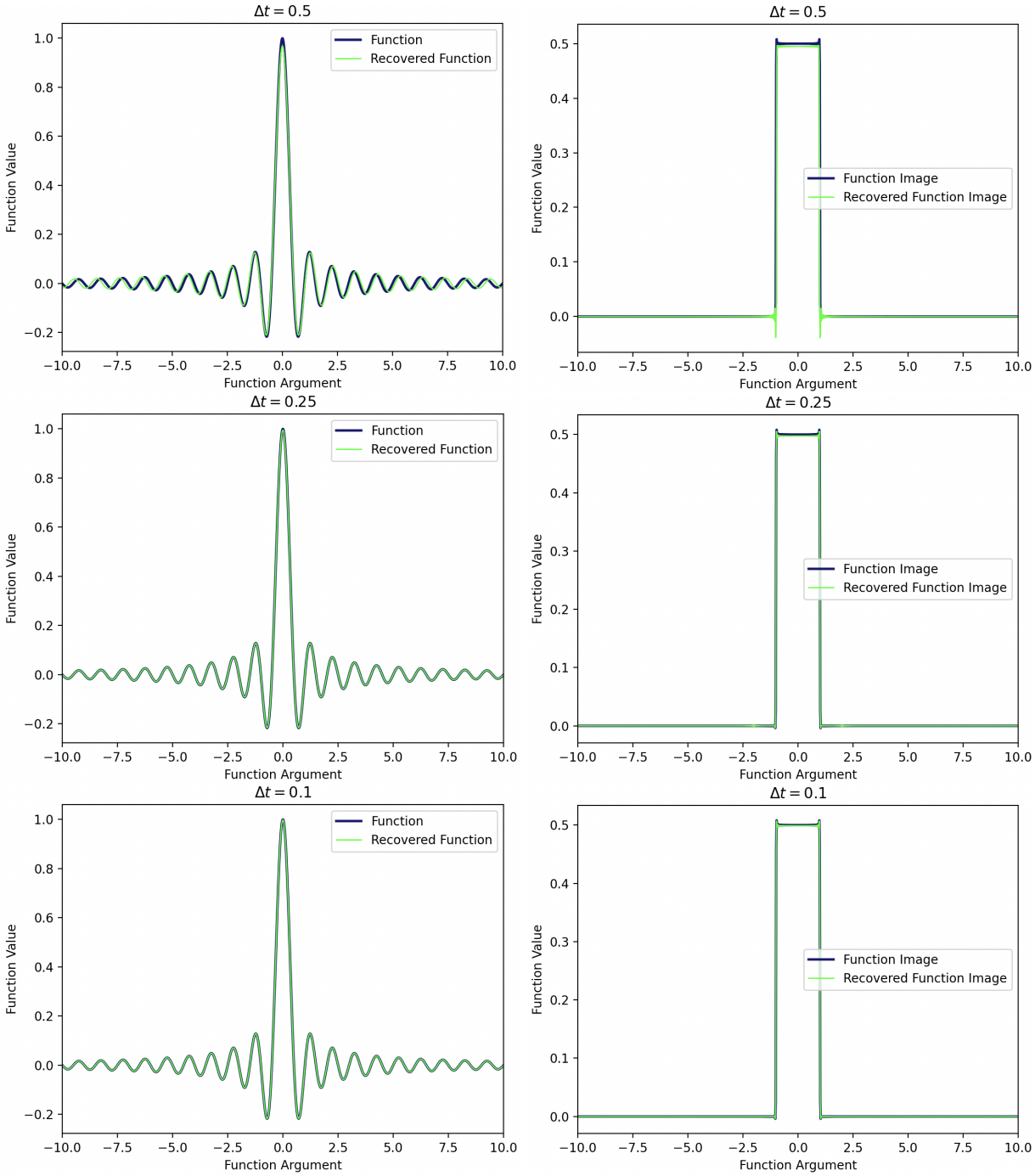


Рис. 12 – Сравнительные графики восстановления функции (слева) и Фурье-образы исходной и восстановленной функций (справа).

Приведем листинг кода, реализующего восстановление функций.

Листинг 3. Реализация восстановления заданной функции.

```
a1, a2 = 1, 1
omega1, omega2 = 1, 1
phi1, phi2 = 1, 1
b = 2

def f(t):
    return np.sinc(b * t)

def fourier_transform(func, freq, time):
    f_image = np.zeros_like(freq)
    for i in range(len(freq)):
```

```
            f_image[i] = np.trapz(func * np.exp(2 * np.pi * complex(0, -1) *
                freq[i] * time), time)
        return f_image

    t = np.linspace(-25, 25, 2500)
    y = f(t)
    freq = np.linspace(-25, 25, 2500)
    y_im = fourier_transform(y, freq, t)

    dt = 0.1
    t1 = np.linspace(-25, 25, int(50 / dt))
    y1 = f(t1)

    def sampling(t_sample, y_sample, time, B):
        sampled = np.zeros_like(time)
        for index, t in enumerate(time):
            summ = np.sum(y_sample * np.sinc(2 * B * (t - t_sample)))
            sampled[index] = summ
        return sampled

    B = 1 / (2 * dt)
    t_recovered = np.linspace(-25, 25, 2500)
    y_recovered = sampling(t1, y1, t_recovered, B)
    freq_recovered = np.linspace(-25, 25, 2500)
    y_im_recovered = fourier_transform(y_recovered, freq_recovered,
        t_recovered)
```

Сделаем некоторые выводы. С помощью графиков Фурье-образов можно понять, что чем меньше точек задано, то есть чем больше шаг дискретизации, тем хуже интерполируется функция, поскольку могут появиться искаженные частоты. Важно выбирать оптимальный (всегда – малый) шаг дискретизации, чтобы восстановление было качественным, а количество артефактов и искажений было минимизировано.

Рассмотренная в лабораторной работе теорема Найквиста-Шеннона-Котельникова работает – мы видим, что задание оптимального шага (равно как задание шага меньшего, чем оптимальный) позволяет совершенно точно интерполировать функцию по дискретному набору значений, в то время как при задании большего шага дискретизации мы можем наблюдать плохое качество интерполяции и плохое совпадение Фурье-образов (равно как появление большого количества искажений).