

**Национальный исследовательский университет ИТМО**

**Факультет систем управления и робототехники**

**Отчёт по дисциплине «Частотные методы»**

**Лабораторная №5**

**«Связь непрерывного и дискретного»**

**Студент:**

Мифтахов Тагир Рашидович

**Группа:**

R3342

**Преподаватель:**

Перегудин Алексей Алексеевич

Санкт-Петербург

2025 г.

# Содержание

<b>1 Непрерывное и дискретное преобразование Фурье</b>	<b>2</b>
1.1 Численное интегрирование . . . . .	2
1.2 Использование DFT . . . . .	18
1.3 Различия методов . . . . .	25
1.4 Приближение непрерывного с помощью DFT . . . . .	26
<b>2 Сэмплирование</b>	<b>33</b>
<b>3 Выводы</b>	<b>46</b>
<b>4 Приложение А</b>	<b>47</b>

# 1 Непрерывное и дискретное преобразование Фурье

Прежде, чем приступить к выполнению задания, рассмотрим знакомую прямоугольную функцию  $\Pi : \mathbb{R} \rightarrow \mathbb{R}$ :

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

Фурье-образом данной функции является аналитическое выражение

$$\hat{\Pi}(\nu) = \int_{-\infty}^{+\infty} \Pi(t) e^{-2\pi i \nu t} dt = \int_{-1/2}^{1/2} e^{-2\pi i \nu t} dt = \left[ \frac{e^{-2\pi i \nu t}}{-2\pi i \nu} \right]_{-1/2}^{1/2} = \frac{-2i \sin(\pi \nu)}{-2\pi i \nu} = \text{sinc}(\pi \nu).$$

Графики  $\Pi(t)$  и  $\hat{\Pi}(\nu)$  представлены на рисунках 1 и 2.

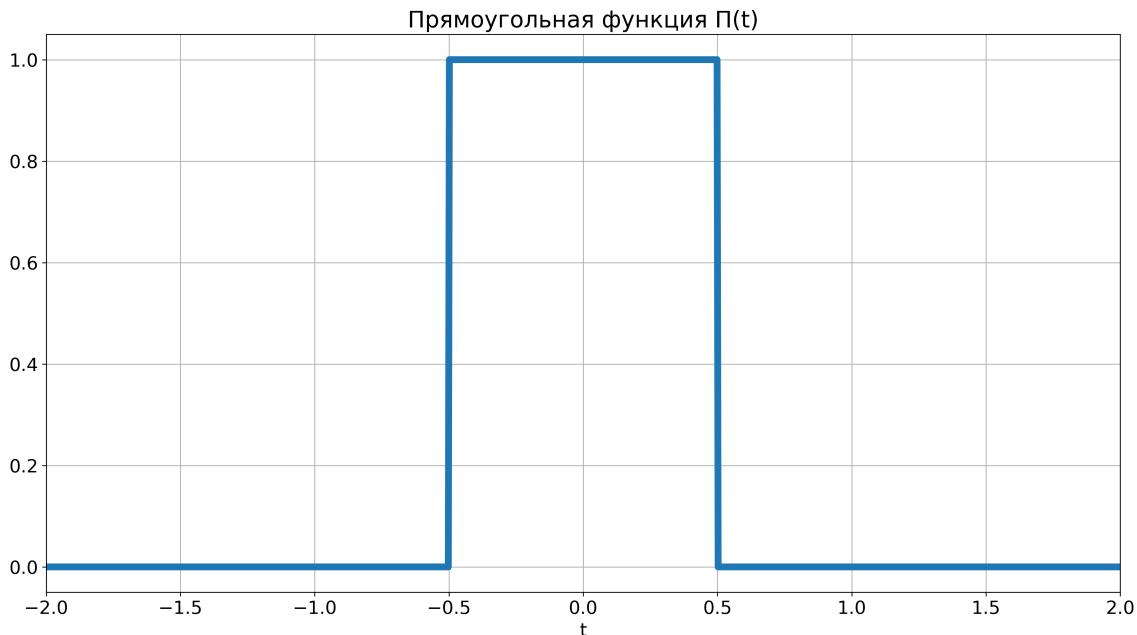


Рис. 1: График функции  $\Pi(t)$

В реальной жизни, когда перед нами стоит задача нахождения Фурье-образа произвольно заданной функции, мы практически всегда склонны прибегнуть к способам сэкономить время и не считать всё вручную, учитывая тот факт, что это очень трудоёмкий процесс, обходящийся нам очень дорого, к тому же не все интегралы возможно взять. Далее мы рассмотрим различные методы реализации вычисления образов и восстановленных по ним функций. Начнём с численного интегрирования.

## 1.1 Численное интегрирование

Идея метода находится на поверхности - давайте вместо прямого интегрирования при прямом и обратном преобразованиях Фурье руками воспользуемся численным

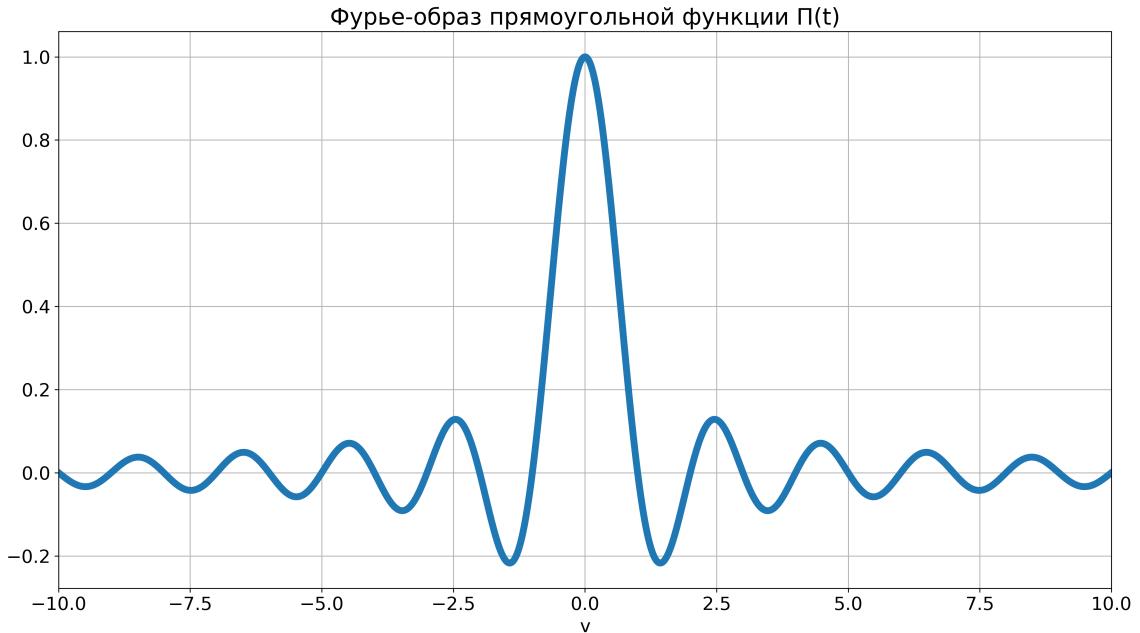


Рис. 2: График образа функции  $\Pi(t)$

(задействуем функцию ***trapz*** для расчета образа и восстановления).

Так как мы не способны задать бесконечное число точек как по времени, так и по частоте, а также потому что мы используем именно приближенные вычисления интегралов, введем параметры промежутка интегрирования по времени  $T$ , шага дискретизации  $\Delta t$ , промежутка интегрирования по частоте  $V$  и шага частот  $\Delta\nu$ , задающие сэмплирования  $\Pi(t)$  и  $\hat{\Pi}(\nu)$ . Посмотрим на получающиеся графики при независимой вариации каждого из параметров, тем самым проверим исследуемый метод в действии и выявим, что же является для нас оптимальным с точки зрения выбора значений этих параметров.

Примем  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$ . Вариация  $T$  при значениях 0.75, 10, 22 продемонстрирована на рисунках 3-8. Увеличение промежутка по времени приводит к увеличению и рассматриваемой области исходной функции и её восстановления. При этом параметр не оказывает никакого воздействия на качество вычисляемых Фурье-образов при  $T \geq 1$ , то есть когда покрыта вся основная волна. Чрезмерное увеличение промежутка по времени приводит к появлению искажений в восстанавливаемой по образу функции, которая при численном интегрировании приобретает периодичность, наблюдавшую при больших  $T$ .

Теперь примем  $T = 5$ ,  $V = 10$ ,  $\Delta\nu = 0.1$ . Вариация  $\Delta t$  при значениях 0.15, 0.05, 0.01 показана на рисунках 9-14. Уменьшение шага дискретизации приводит к густоте задаваемой временной сетке и сглаживанию восстанавливаемой функции. Мелкость параметра также увеличивает точность вычисления образа Фурье. Теперь, когда мы берем частые временные точки, мы способны вычислить интеграл более точным образом, так как при бесконечно малых  $\Delta t$  как раз и получается истинное значение пря-

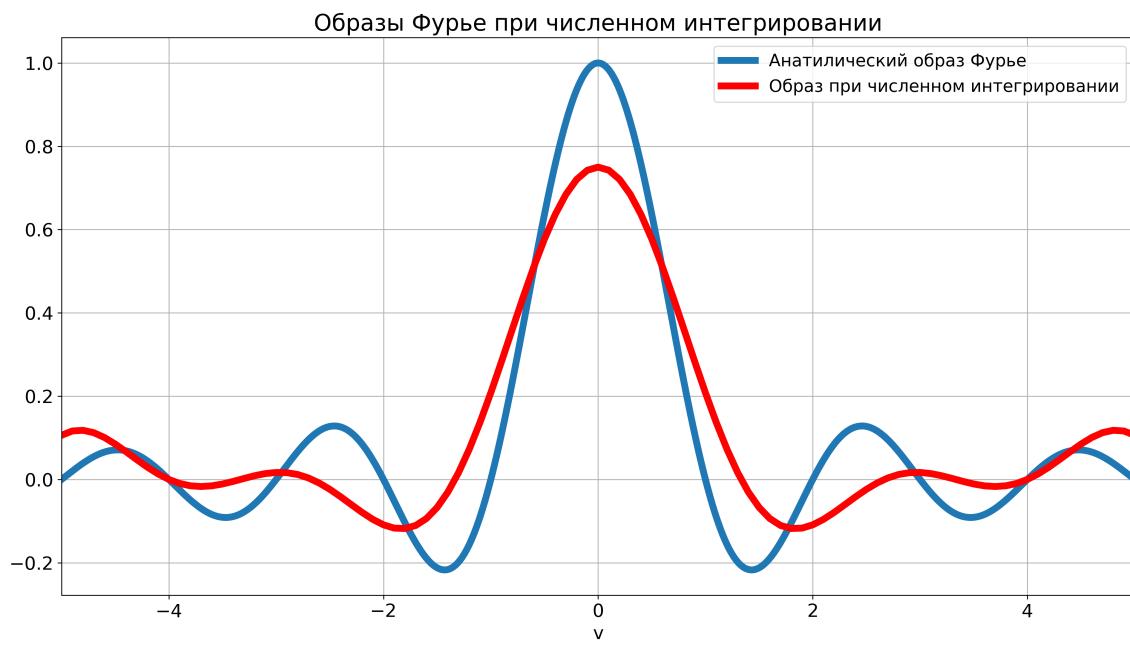


Рис. 3: Образы при численном интегрировании:  $T = 0.75$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

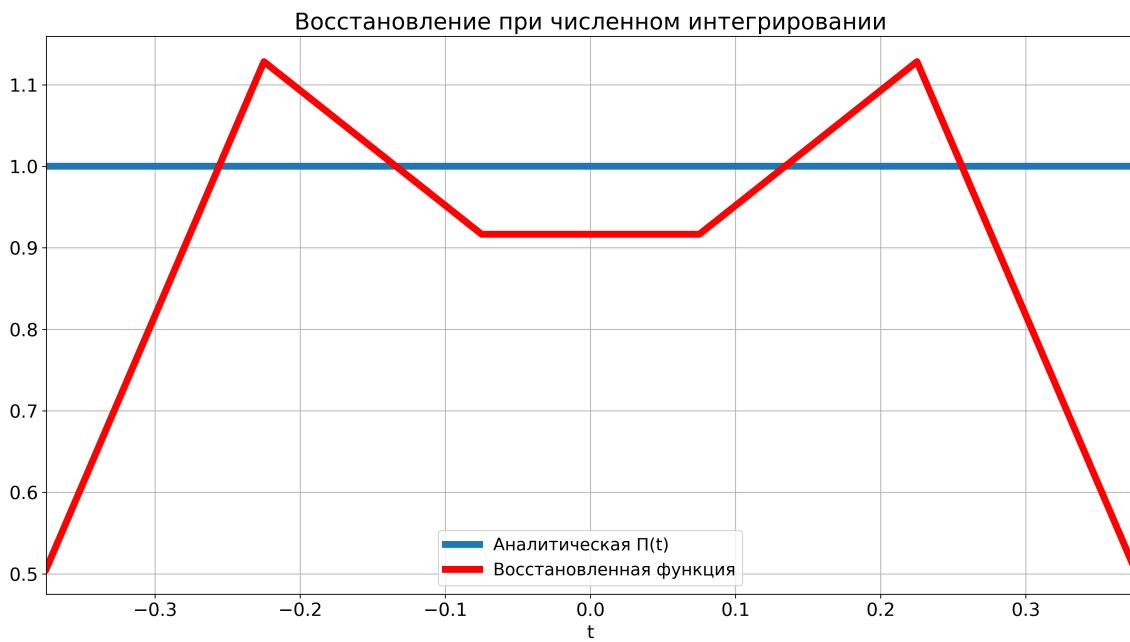


Рис. 4: Восстановление, численное интегрирование:  $T = 0.75$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

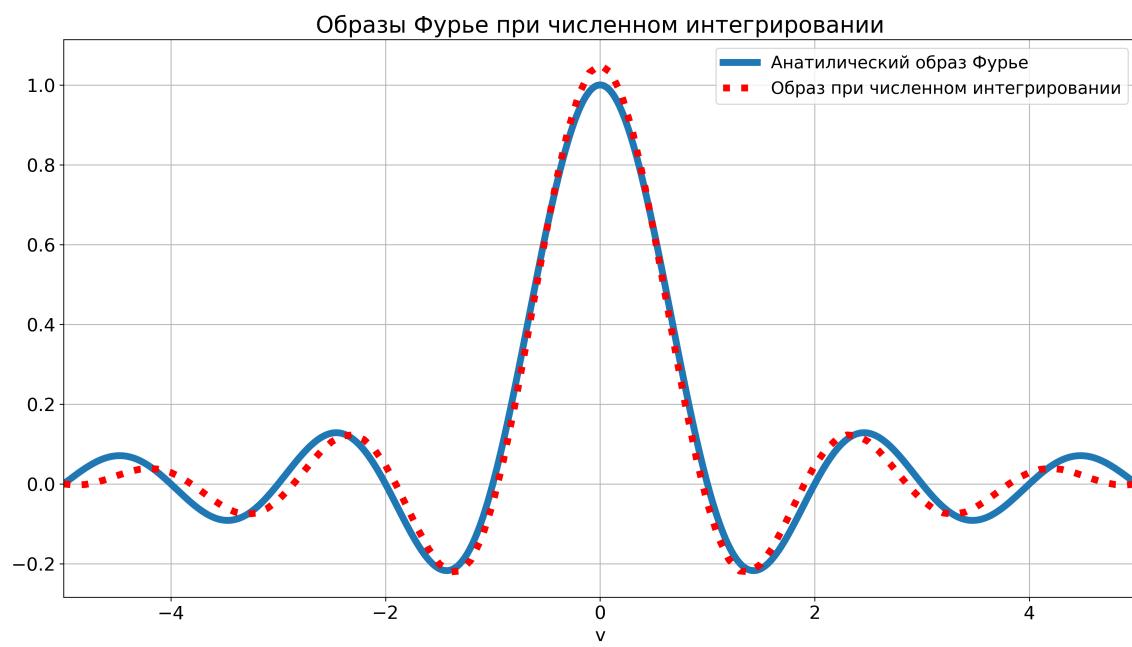


Рис. 5: Образы при численном интегрировании:  $T = 10$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

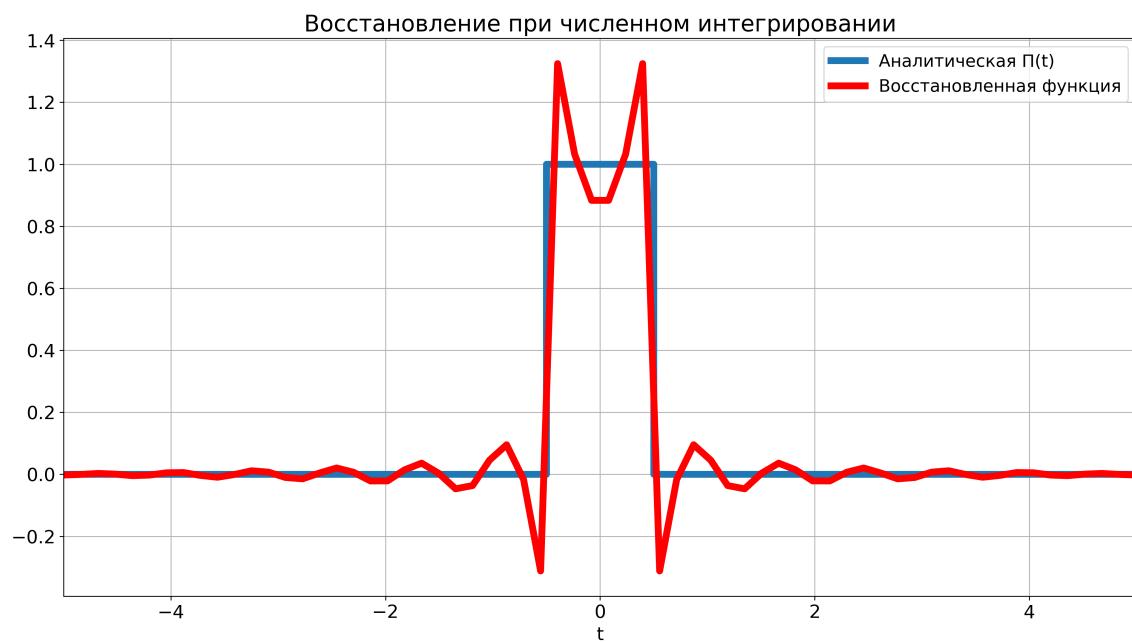


Рис. 6: Восстановление при численном интегрировании:  $T = 10$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

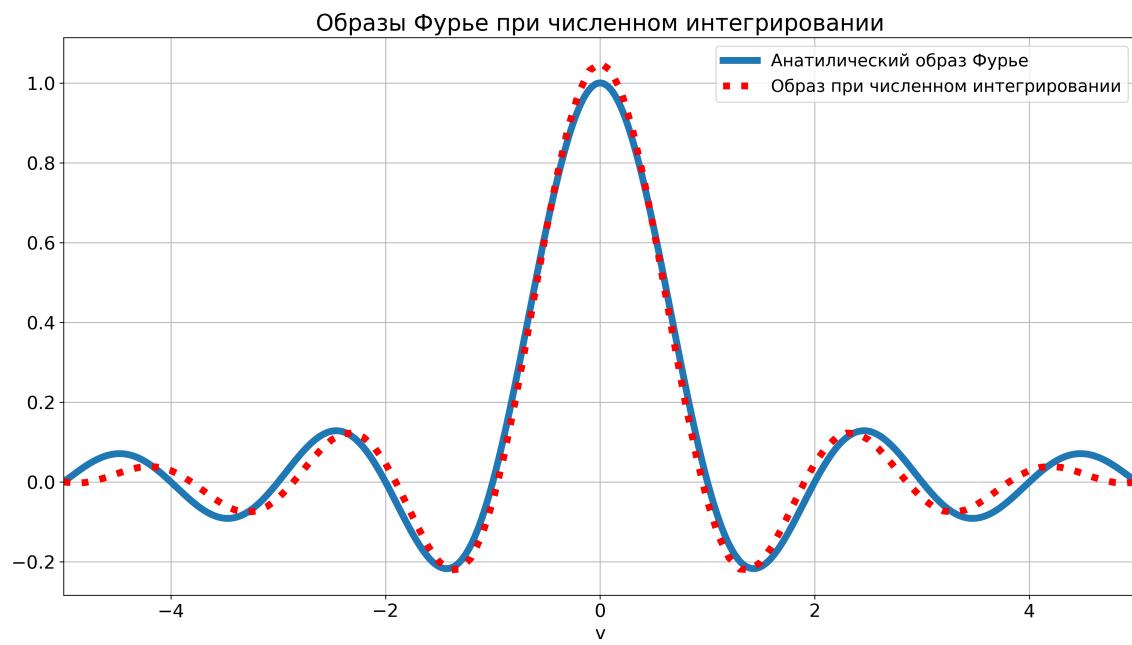


Рис. 7: Образы при численном интегрировании:  $T = 22$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

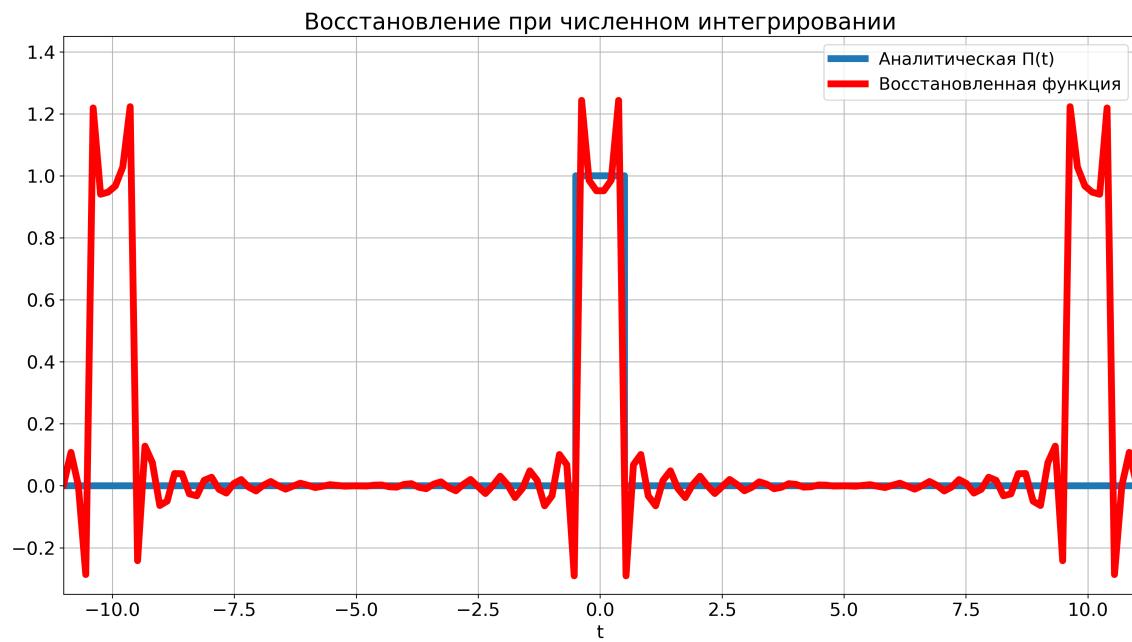


Рис. 8: Восстановление при численном интегрировании:  $T = 22$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

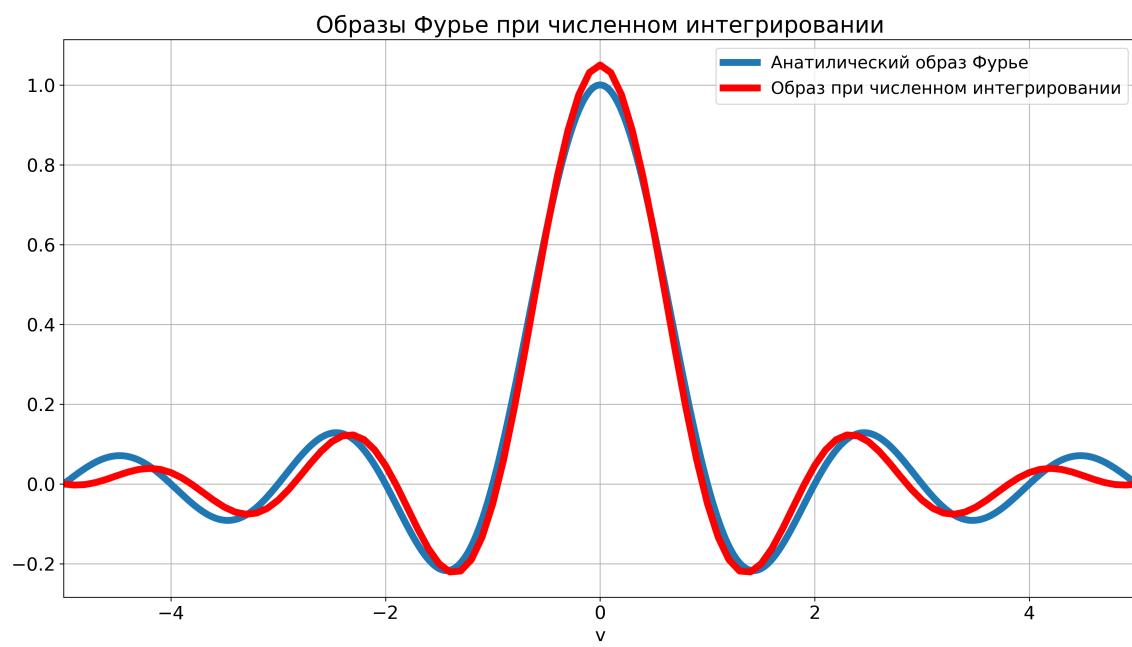


Рис. 9: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

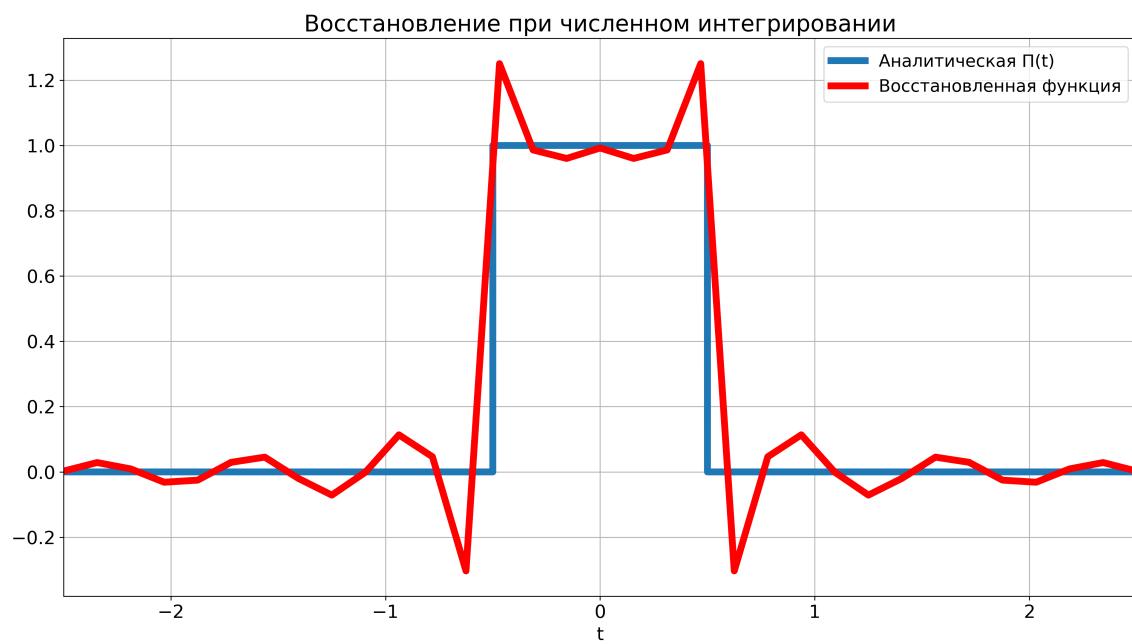


Рис. 10: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

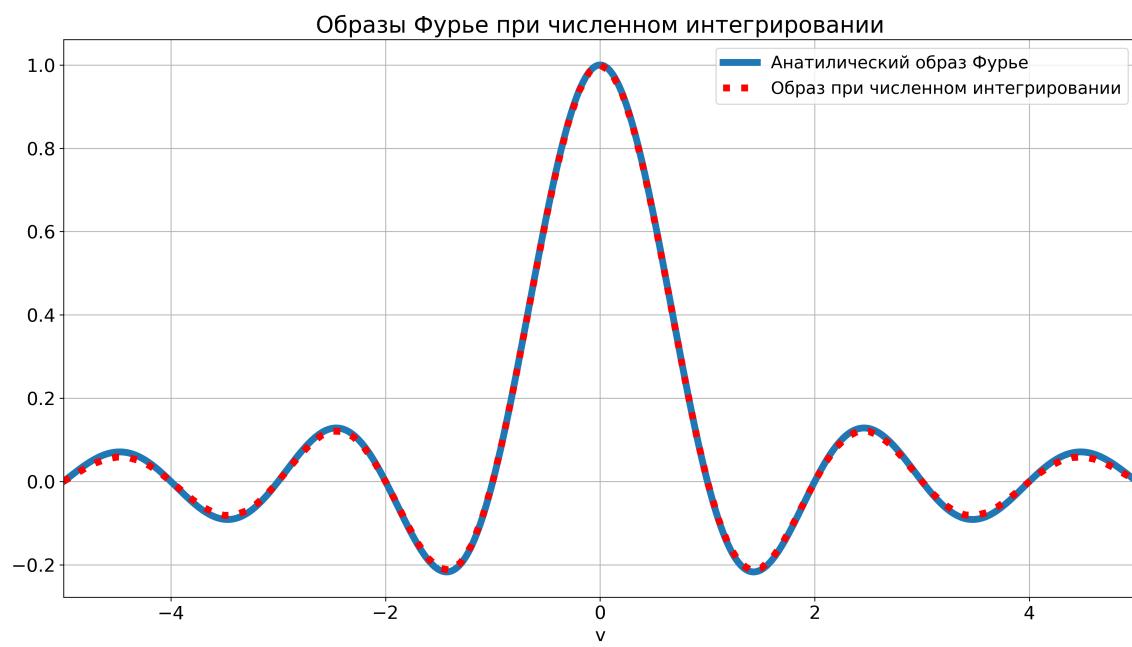


Рис. 11: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.05$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

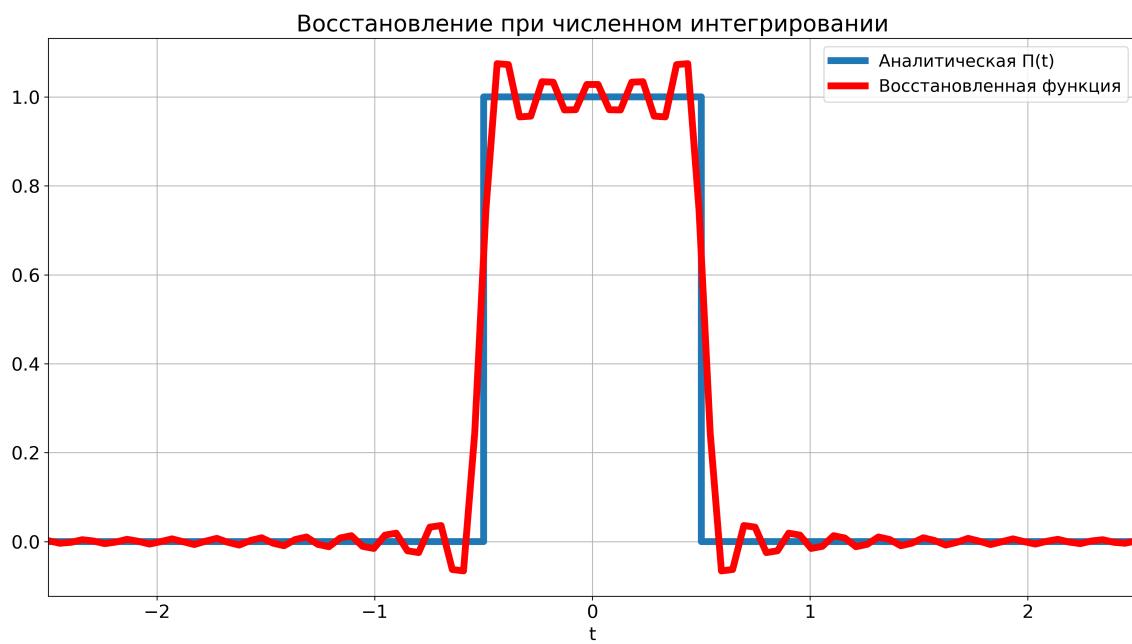


Рис. 12: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.05$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

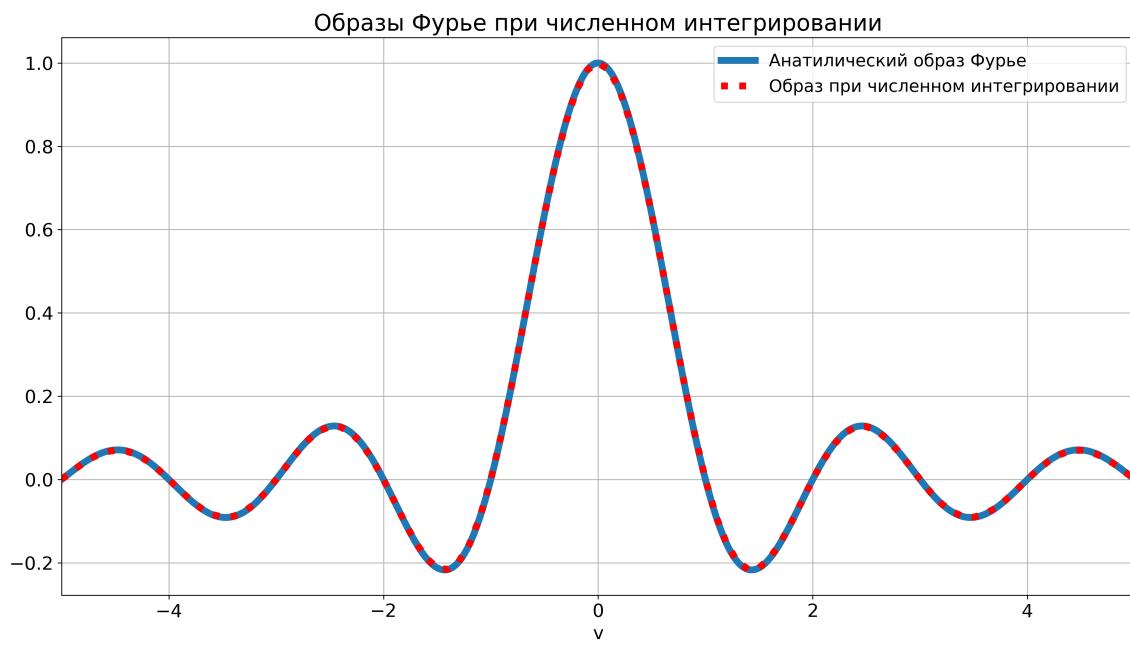


Рис. 13: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.01$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

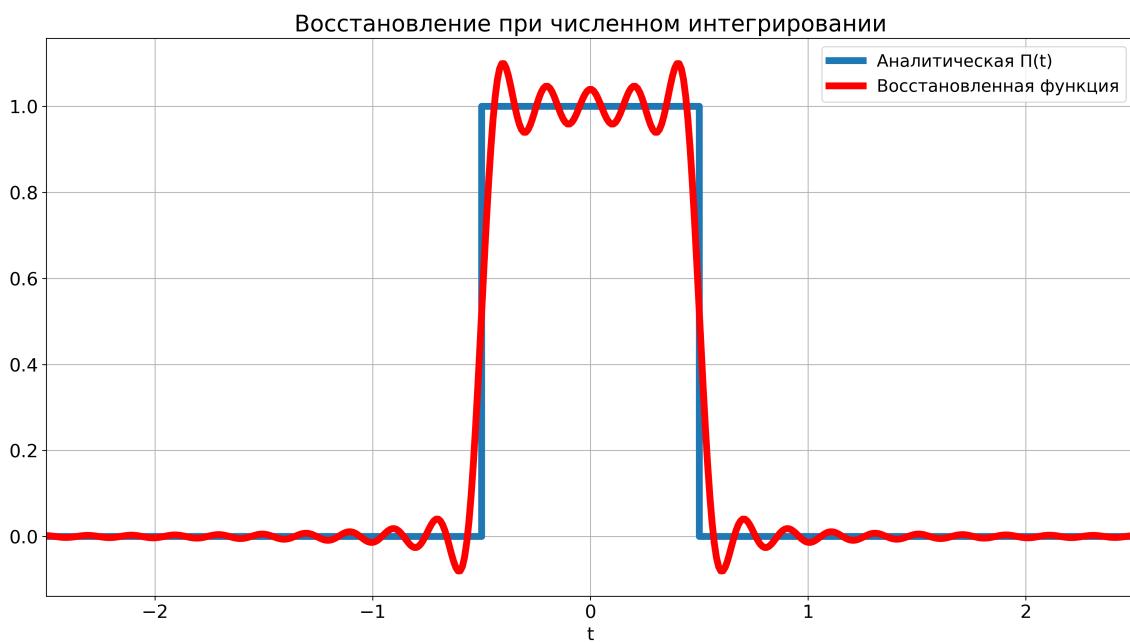


Рис. 14: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.01$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

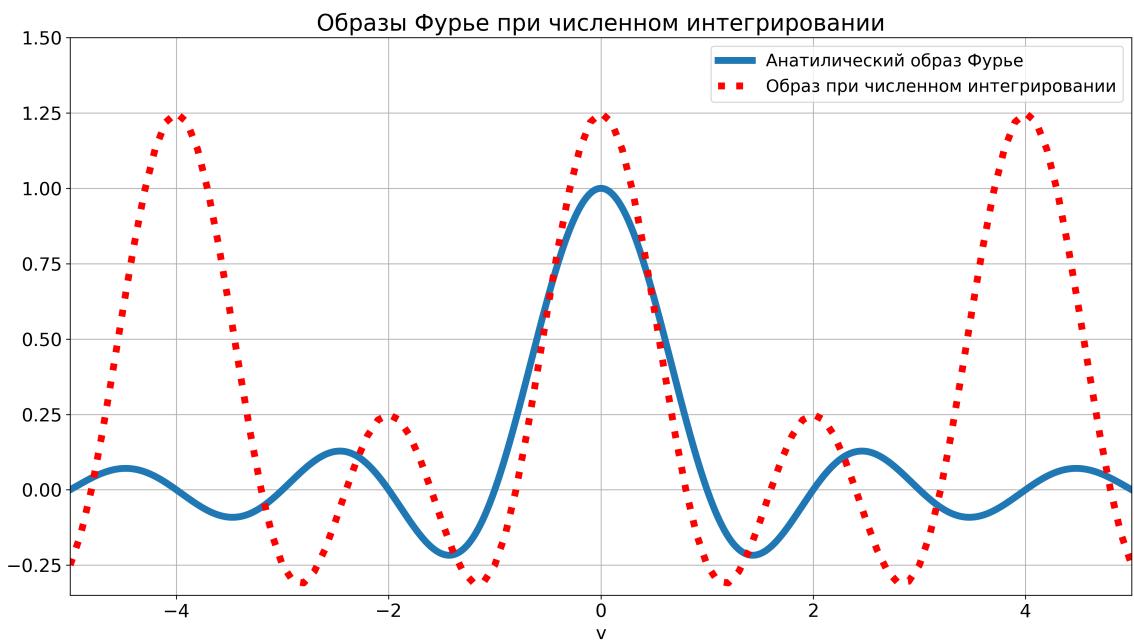


Рис. 15: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.25$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

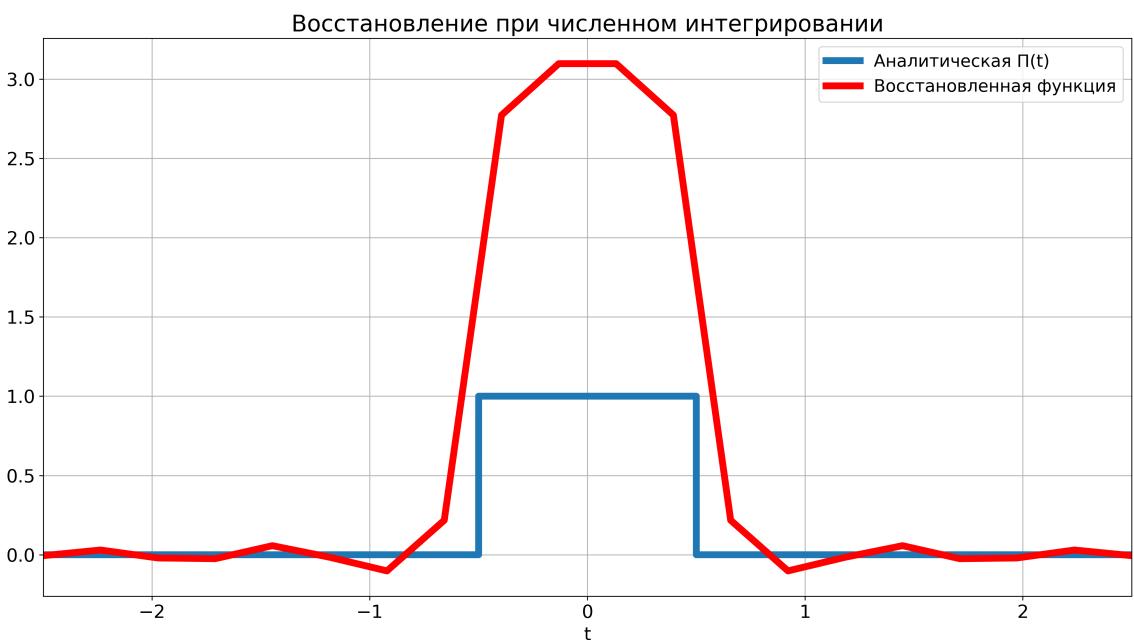


Рис. 16: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.25$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

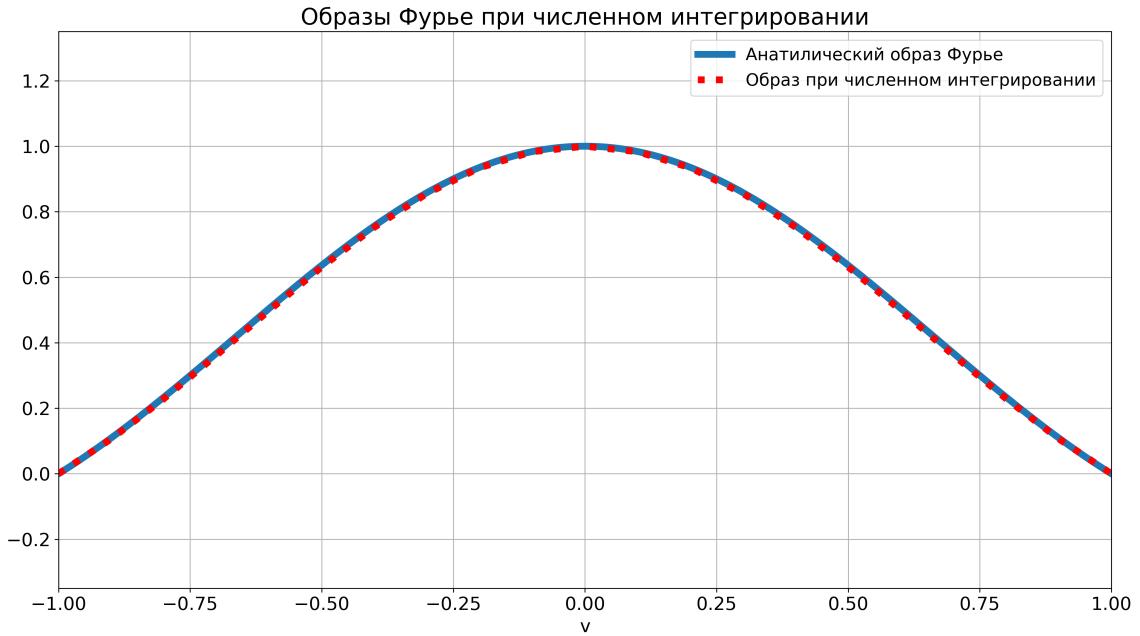


Рис. 17: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 2$ ,  $\Delta\nu = 0.1$

мого преобразования Фурье. Кроме того, параметр обратно пропорционально влияет на периодичность вычисляемого Фурье-образа, что заметно рисунках 15, 16, взятых при параметрах  $T = 5$ ,  $\Delta t = 0.25$ ,  $V = 10$ ,  $\Delta\nu = 0.1$ . Мелкая периодичность серьёзно сказывается на качестве высоких частот, поэтому  $\Delta t$  стоит столь малым, чтобы промежуток интегрирования по частотам не захватил лишнего и не искажал значения восстанавливаемой функции.

Исследуем влияние параметра  $V$  при фиксированных  $T = 5$ ,  $\Delta t = 0.1 = \Delta\nu = 0.1$ . Вариация промежутка интегрирования по частотам  $V$  при значениях 2, 8, 16 продемонстрирована на рисунках 17-22. Увеличение промежутка по частотам увеличивает и рассматриваемый частотный отрезок, что, как уже говорилось выше, может привести к искажениям в восстанавливаемой функции при больших значениях. В случае же чрезмерной обрезки образа получаем ситуацию нехватки частот для корректного приближения интеграла обратного преобразования. Таким образом, необходимо выбирать нечто среднее.

Наконец примем  $T = 5$ ,  $\Delta t = 0.15$ ,  $\Delta\nu = 0.1$ . Вариация параметра шага частот  $\Delta\nu$  при значениях 0.5, 0.1, 0.01 продемонстрирована на рисунках 23-28. Зеркально параметру  $\Delta t$  при увеличении  $\Delta\nu$  происходит сглаживание образов Фурье, а также увеличение периодичности восстанавливаемой функции.

В итоге при численном интегрировании функции и образы становятся периодическими на промежутках  $T$  и  $V$  соответственно, периоды обратно пропорциональны шагу дискретизации и шагу частот. Кроме того, мелкостью  $\Delta t$  и  $\Delta\nu$  достигаются лучшие значения интегралов, поэтому практически всегда необходимо выбирать очень малые шаги.

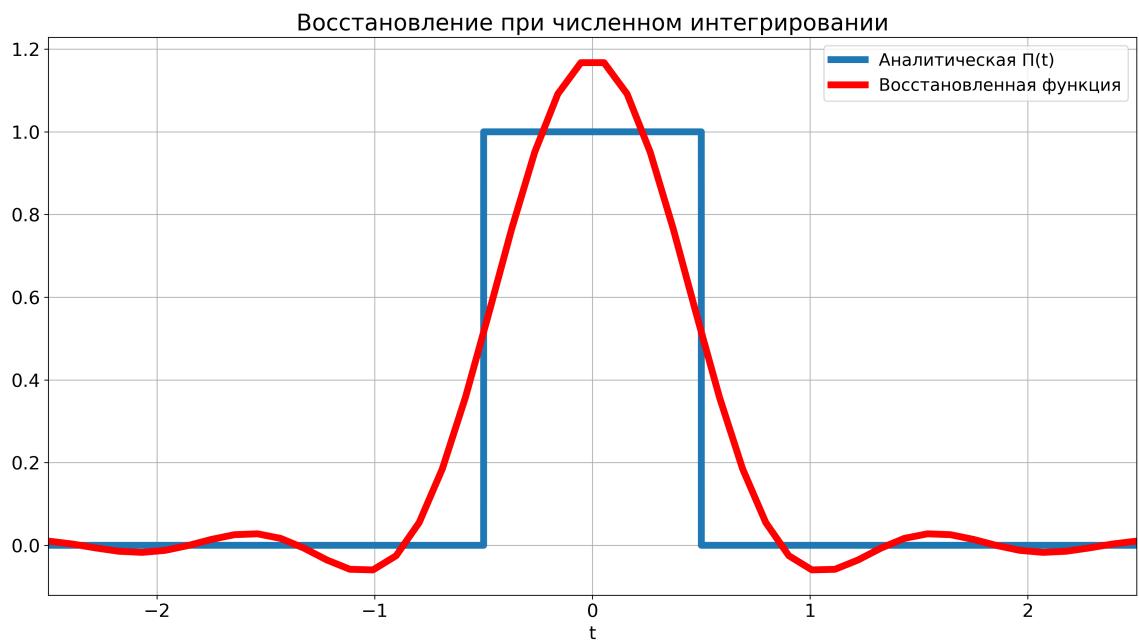


Рис. 18: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 2$ ,  $\Delta\nu = 0.1$

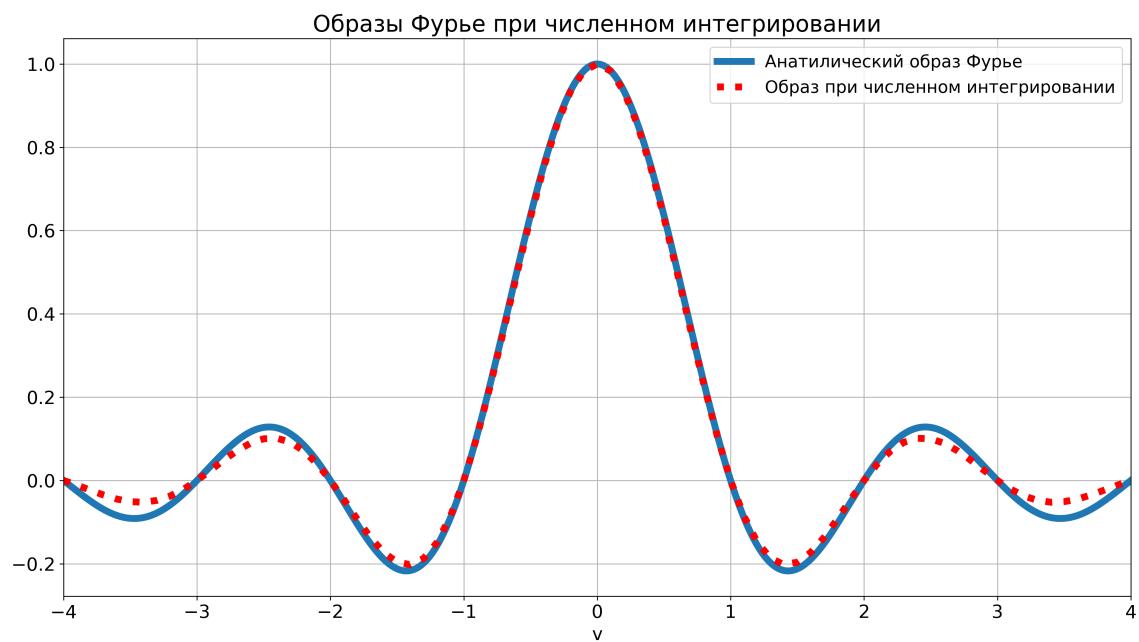


Рис. 19: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 8$ ,  $\Delta\nu = 0.1$

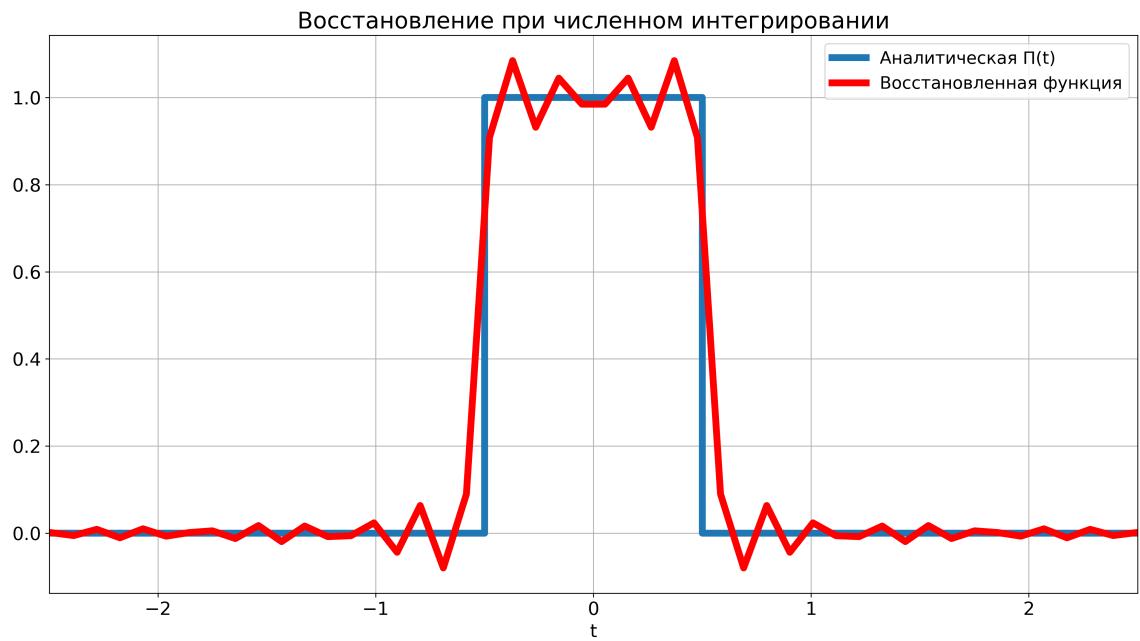


Рис. 20: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 8$ ,  $\Delta\nu = 0.1$

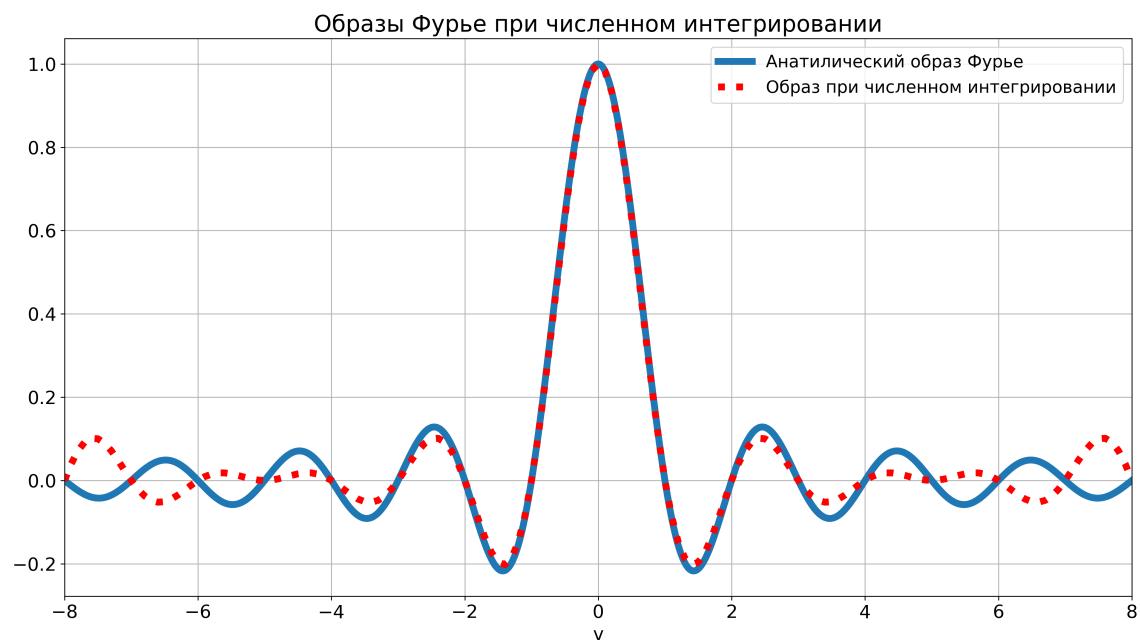


Рис. 21: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 16$ ,  $\Delta\nu = 0.1$

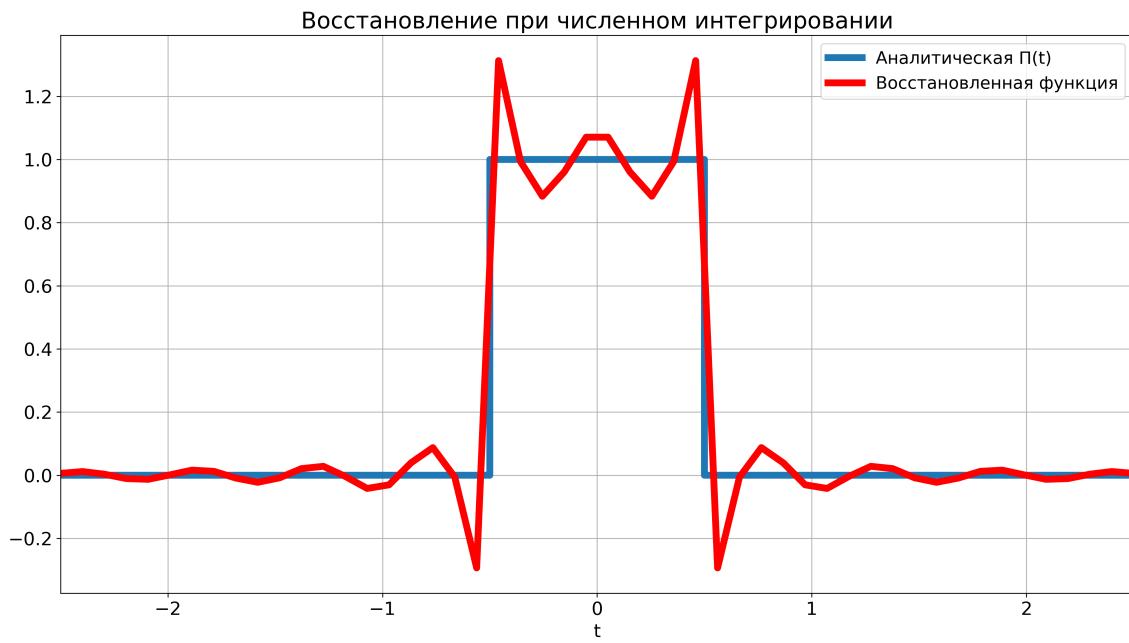


Рис. 22: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.1$ ,  $V = 16$ ,  $\Delta\nu = 0.1$

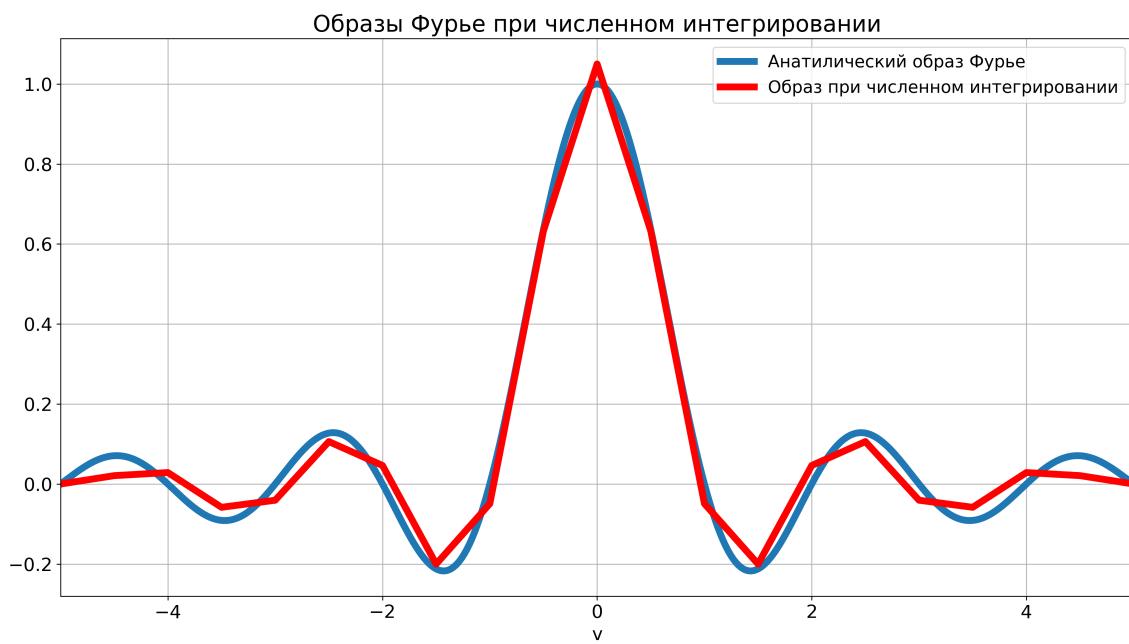


Рис. 23: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.5$

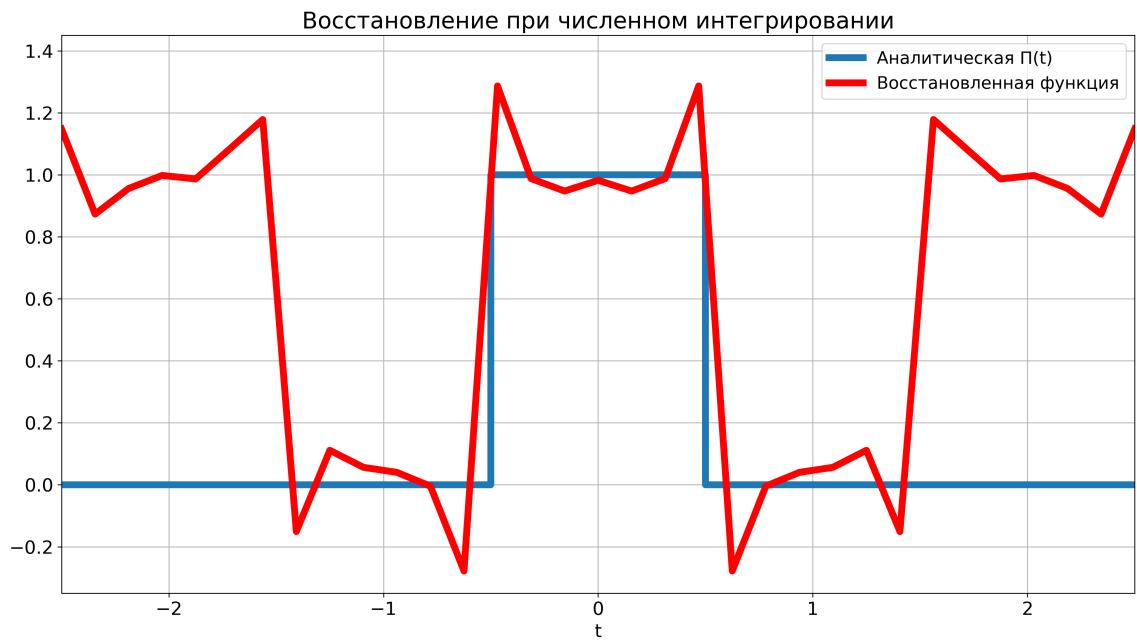


Рис. 24: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.5$

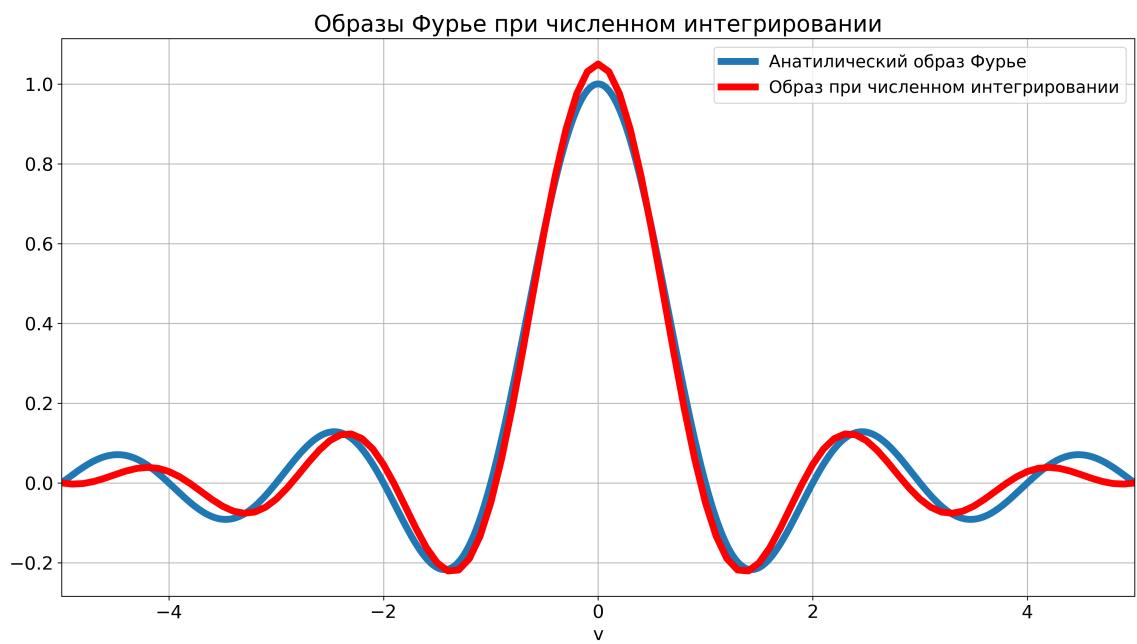


Рис. 25: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

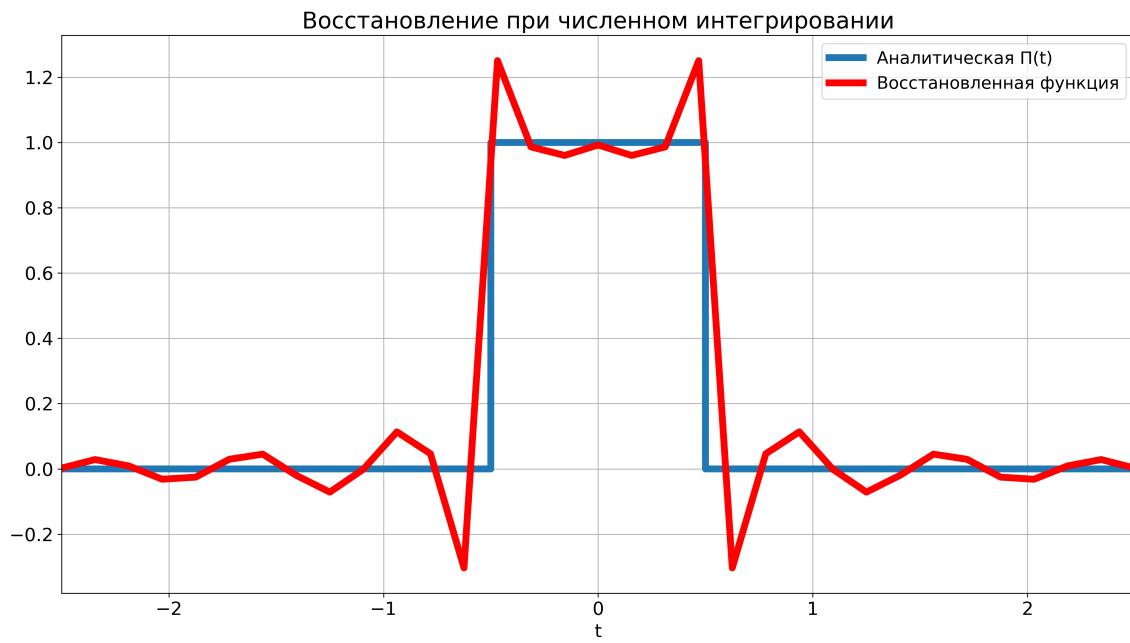


Рис. 26: Восстановление при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.1$

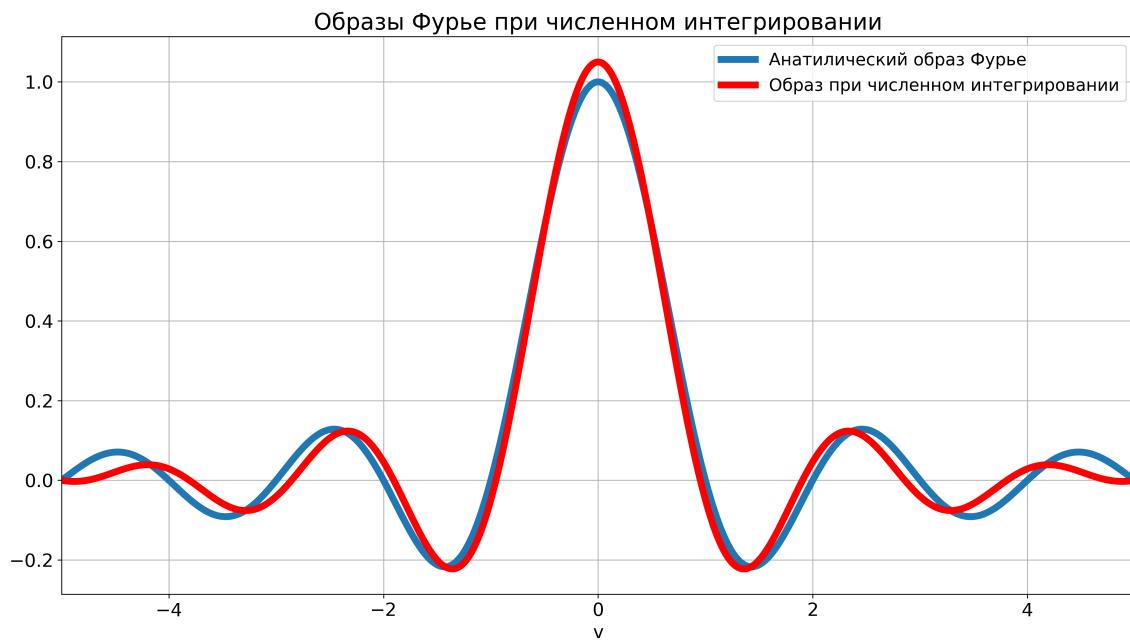


Рис. 27: Образы при численном интегрировании:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.01$

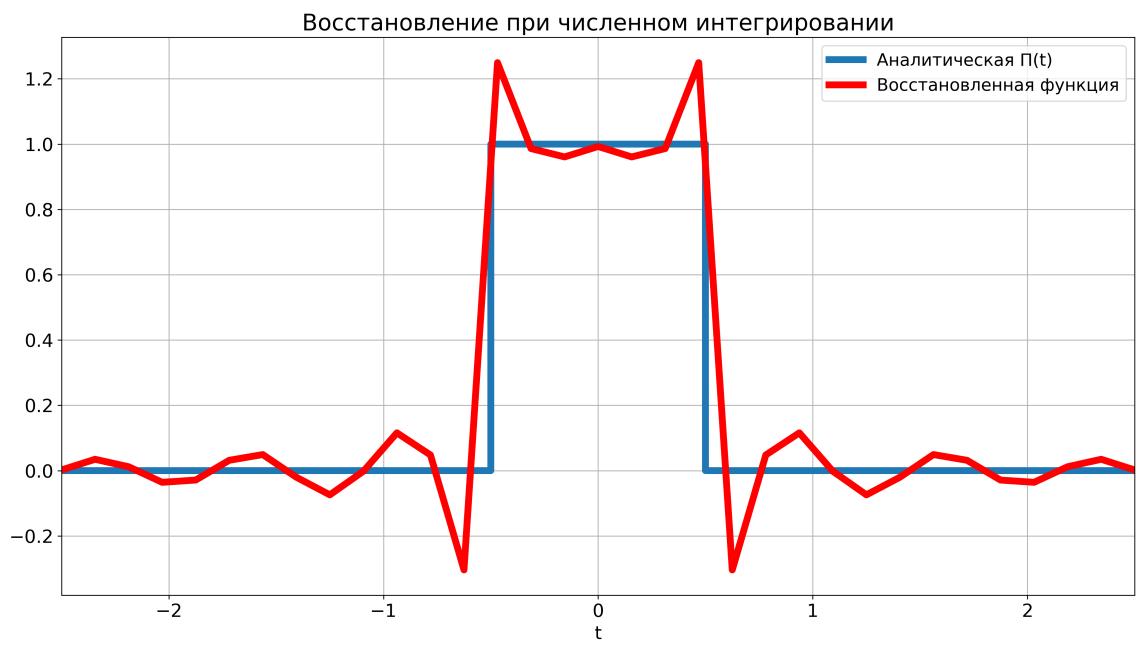


Рис. 28: Обратное численное интегрирование:  $T = 5$ ,  $\Delta t = 0.15$ ,  $V = 10$ ,  $\Delta\nu = 0.01$

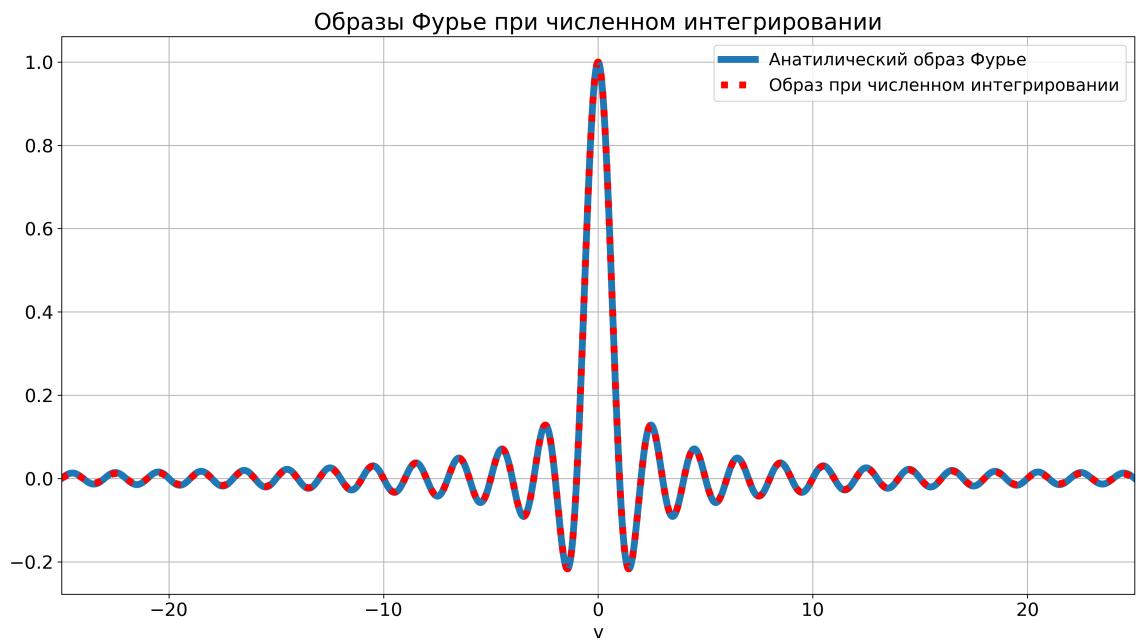


Рис. 29: Образы при численном интегрировании:  $T = 10$ ,  $\Delta t = 0.01$ ,  $V = 50$ ,  $\Delta\nu = 0.01$

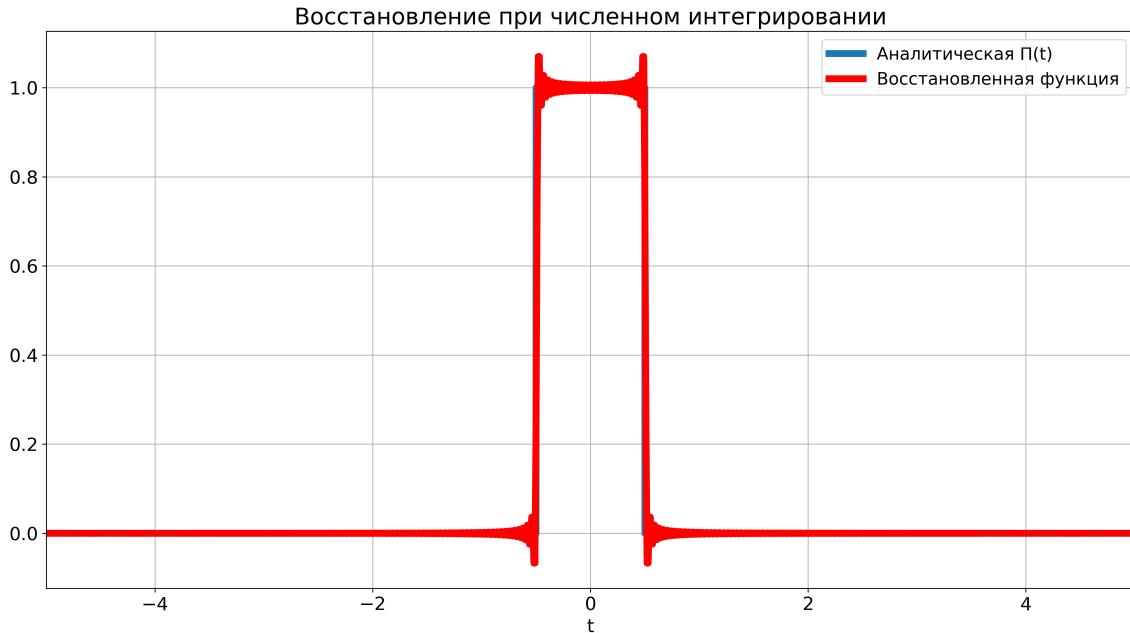


Рис. 30: Восстановление, численное интегрирование:  $T = 10$ ,  $\Delta t = 0.01$ ,  $V = 50$ ,  $\Delta\nu = 0.01$

Исходя из исследований выше, были выбраны параметры  $T = 10$ ,  $\Delta t = 0.01$ ,  $V = 50$ ,  $\Delta\nu = 0.01$  и построены графики, изображенные на рисунках 29 и 30. Как видим, с образами картина положительная - метод дал идеальное визуальное приближение, однако в восстановлении показал себя не лучшим образом - на краях прямоугольной функции наблюдаем серьёзные отклонения от аналистики. Возникшее объясняется поточечной сходимостью преобразования: при росте промежутка интегрирования и соответствующего шага мы способны достичь сколь угодно малого отклонения от аналистики. У непрерывных функций также существует и равномерная сходимость: мы приближаемся к аналитической, истинной форме во всех точках одновременно. Однако у функций с разрывами равномерной сходимости не существует, появляются бешеные отклонения в окрестности точек разрыва. Интуитивно это работает так: для передачи разрыва нам необходима экспонента с бесконечно большой частотой, а такого в реальной жизни, конечно же, не существует. В связи с этим численное интегрирование несколько ограничено в приближениях функций.

Также исследуемый метод довольно длителен в выполнении при большом наборе значений  $N$ , так как интегрирование - довольно объемная операция по времени, требующая больших вычислительных ресурсов.

## 1.2 Использование DFT

Идея следующего метода - так как мы всё равно сэмплируем наши функции, давайте проводить не *непрерывное* преобразование Фурье, а *дискретное* (задействуем функцию *fft* для расчета образа и *ifft* для восстановления).

В данном случае параметры промежутка времени  $T$  и шага дискретизации  $\Delta t$

явно связаны с соответствующими им промежутком по частотам  $V = \frac{1}{\Delta t}$  и шагом частот  $\Delta\nu = \frac{1}{T}$ , поэтому рассматриваться будут только  $T$  и  $\Delta\nu$ .

Примем  $\Delta t = 0.1$ ,  $V = 10$ . Вариация  $T$  при значениях 5, 10 и 20 представлена на рисунках 31-36, соответствующие шаги частот  $\Delta\nu = 0.2, 0.1, 0.05$ . Увеличение промежутка времени влечёт и увеличению рассматриваемой области, в которой задается восстановление. Также с ростом  $T$  уменьшается шаг частот  $\Delta\nu$ , что сказывается на густоте задаваемого частотного отрезка.

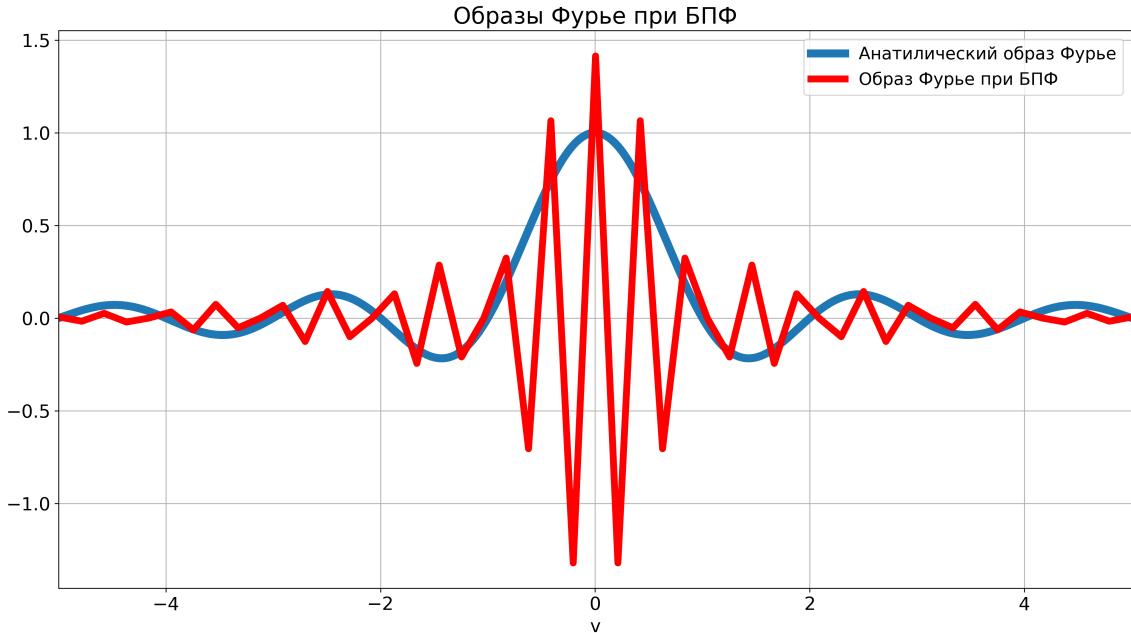


Рис. 31: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.1$

Теперь примем  $T = 5$ ,  $\Delta\nu = 0.2$ . Вариация  $\Delta t$  при значениях 0.25, 0.05, 0.01 продемонстрирована на рисунках 37-42, соответствующие области  $V = 4, 20, 100$ . Мелкость шага дискретизации приводит к лучшей передачи поведения функции по времени - уплотнение временной сетки даёт более частые значения, а значит происходит лучшее приближение непрерывности, резкие скачки и в целом поведение функции становятся естественными. С уменьшением  $\Delta t$  растёт также и рассматриваемый промежуток по частотам  $V$ , образы Фурье задаются на большем отрезке.

Можно было заметить, что образы Фурье будто бы не стремятся быть похожими на аналитические, и лишь изредка (например, рисунок 33) могут иметь приближенный вид, однако и в этом случае нам бы не помешало умножение на  $(-1)^n$  для избавления от периодических скачков. Полученное объясняется тем, что в исследуемом методе используется именно *дискретное* преобразование Фурье, образ функции в этом случае вовсе и не обязан быть похожим на аналитический, вычисленный при *непрерывном* преобразовании, подразумевающем ту же природу и у функций.

Точность метода по времени при этом стала гораздо выше, так как восстанавливается сэмплированная функция (дискретизация непрерывного), которая стремится

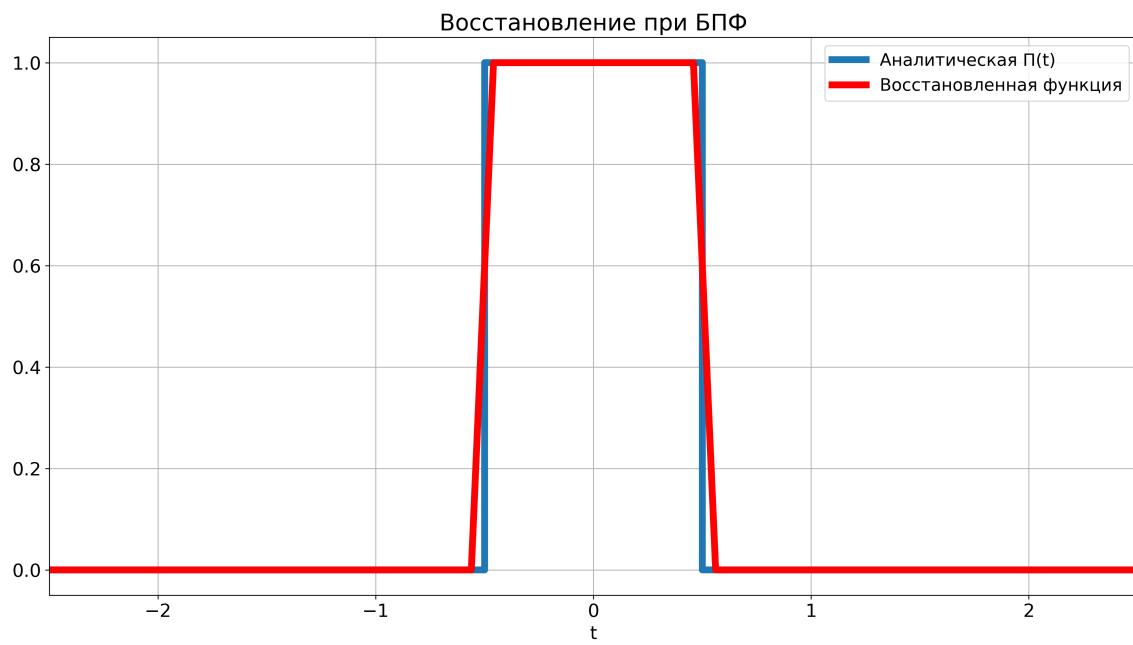


Рис. 32: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.1$

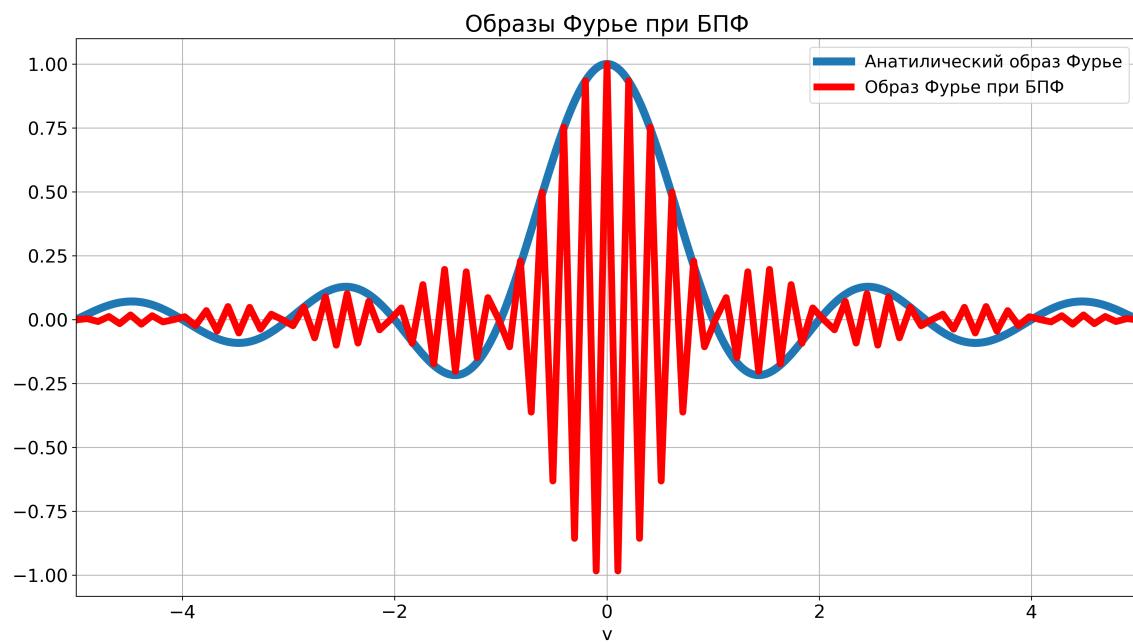


Рис. 33: Образы при быстром преобразовании Фурье:  $T = 10$ ,  $\Delta t = 0.1$

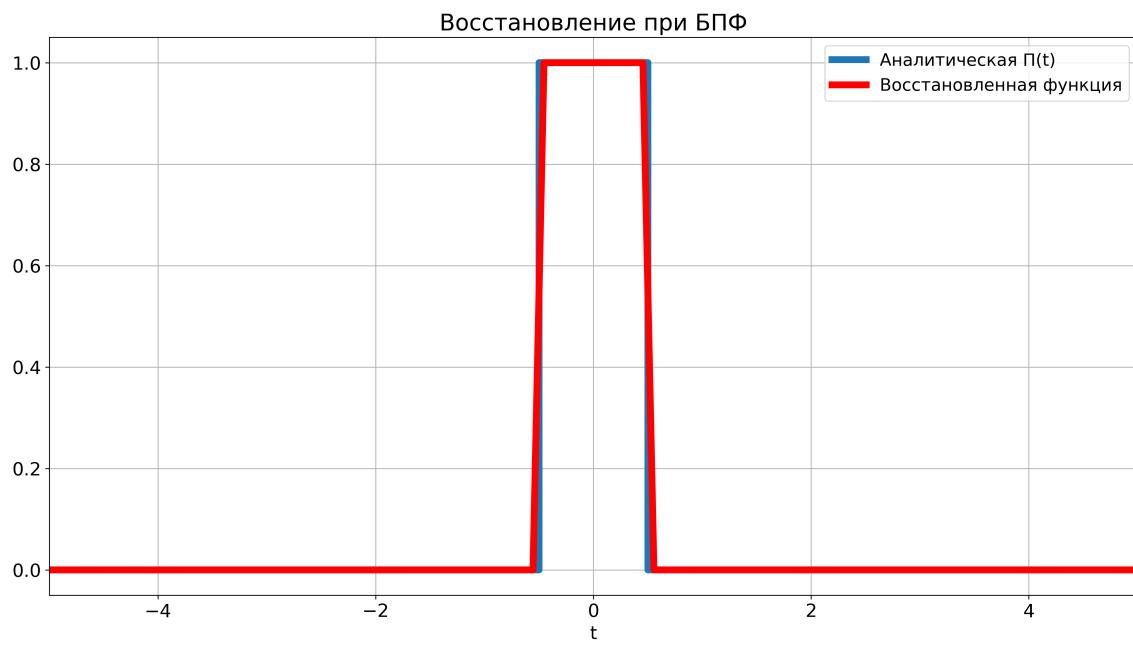


Рис. 34: Восстановление при быстром преобразовании Фурье:  $T = 10$ ,  $\Delta t = 0.1$

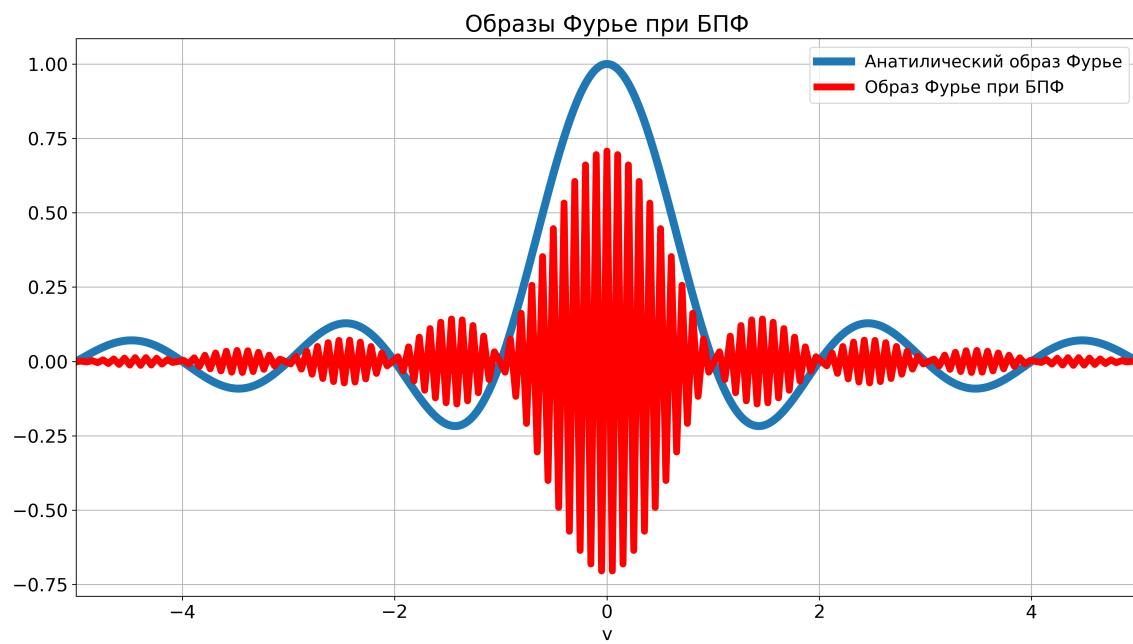


Рис. 35: Образы при быстром преобразовании Фурье:  $T = 20$ ,  $\Delta t = 0.1$

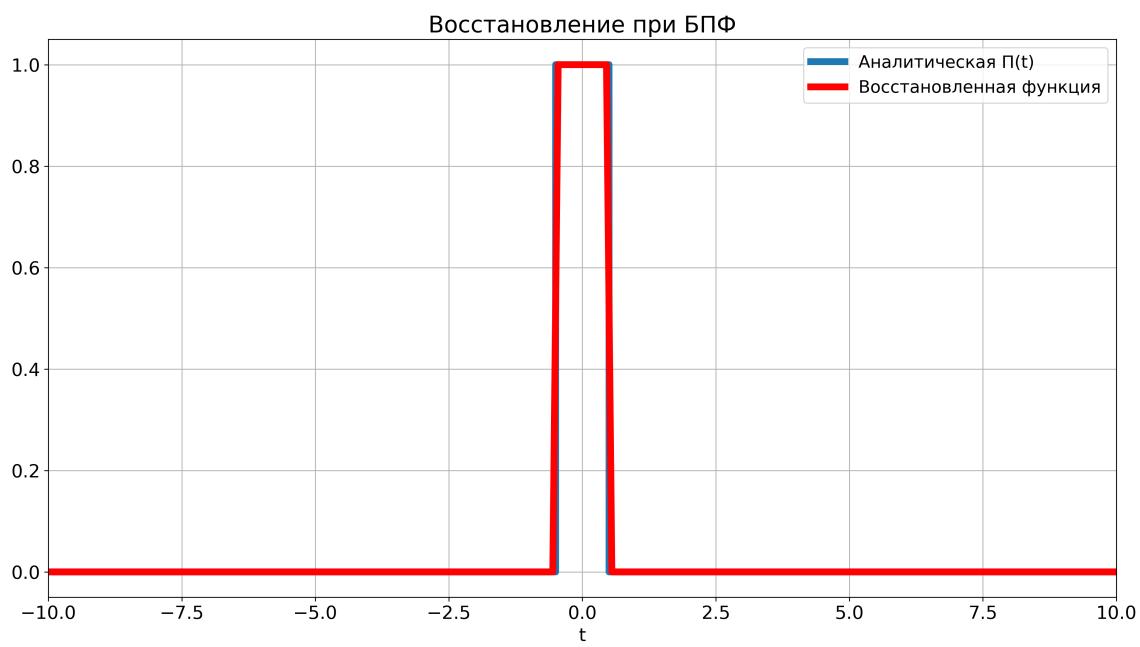


Рис. 36: Восстановление при быстром преобразовании Фурье:  $T = 20$ ,  $\Delta t = 0.1$

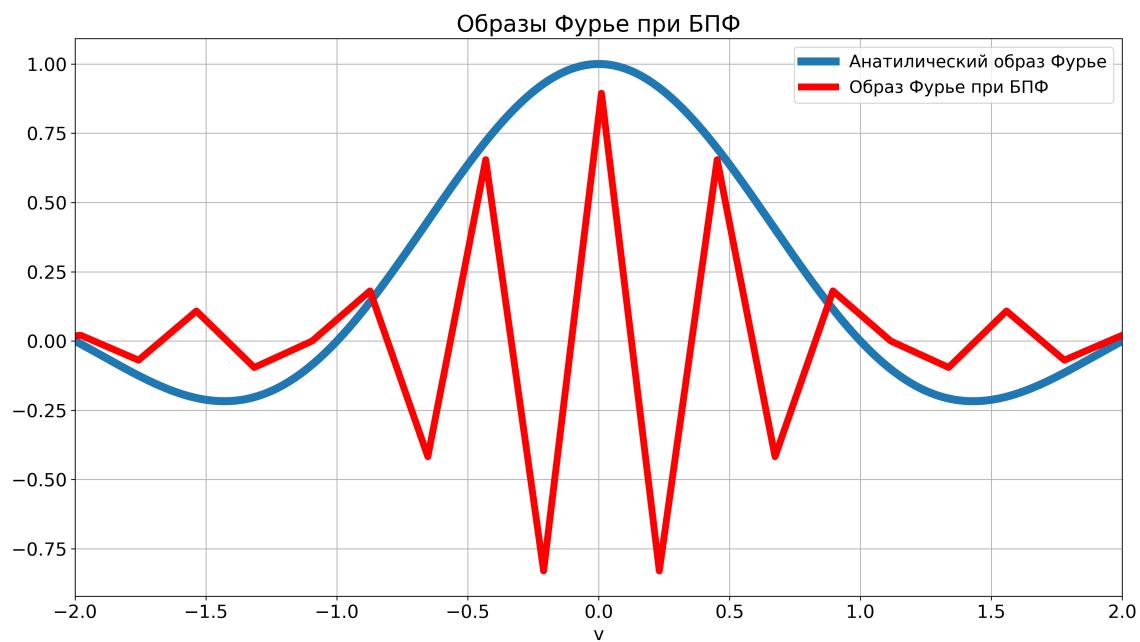


Рис. 37: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.25$

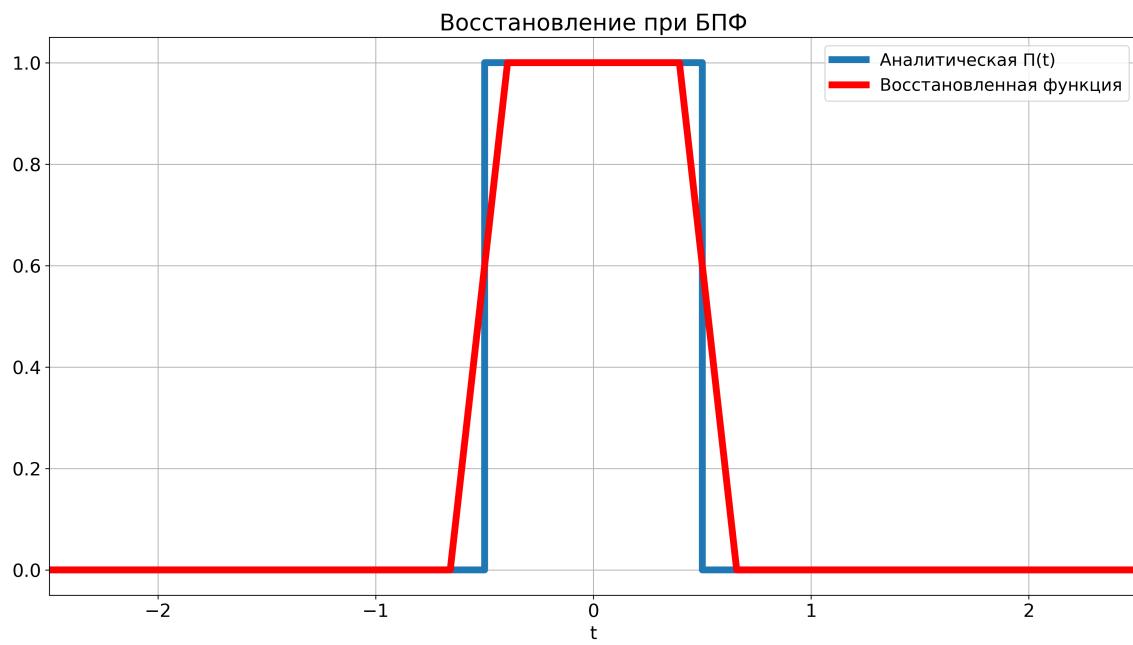


Рис. 38: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.25$

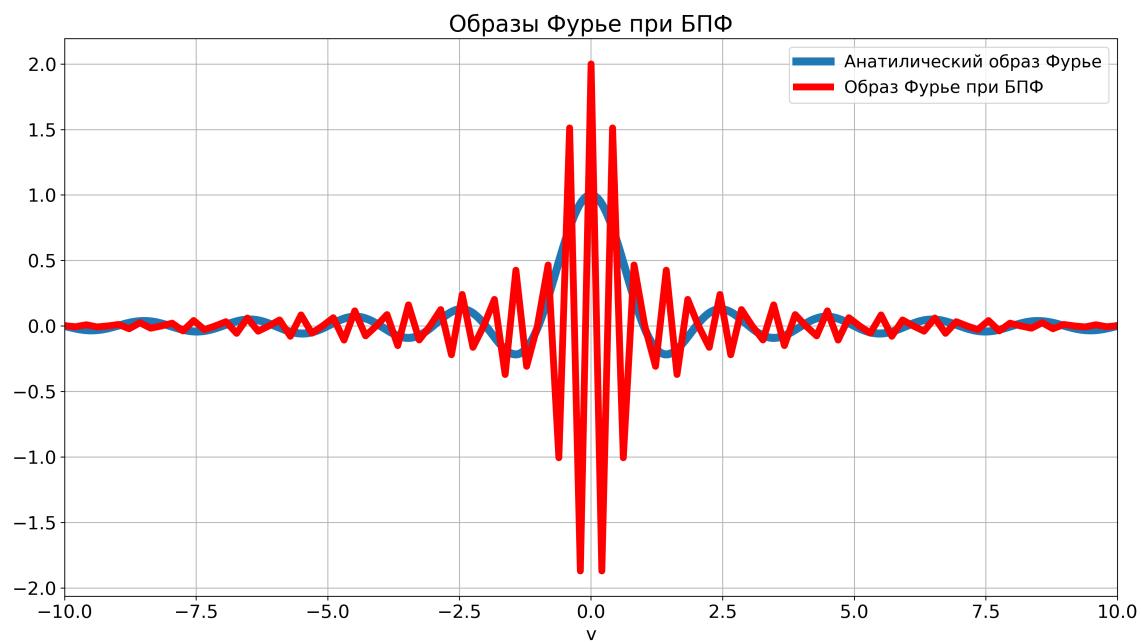


Рис. 39: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.05$

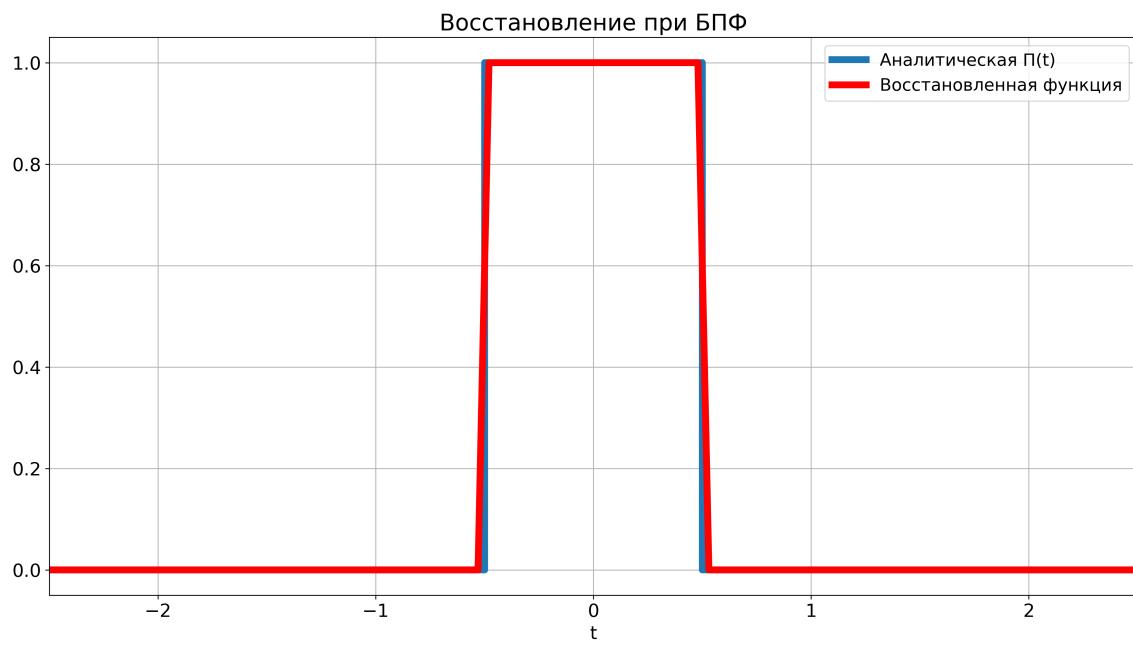


Рис. 40: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.05$

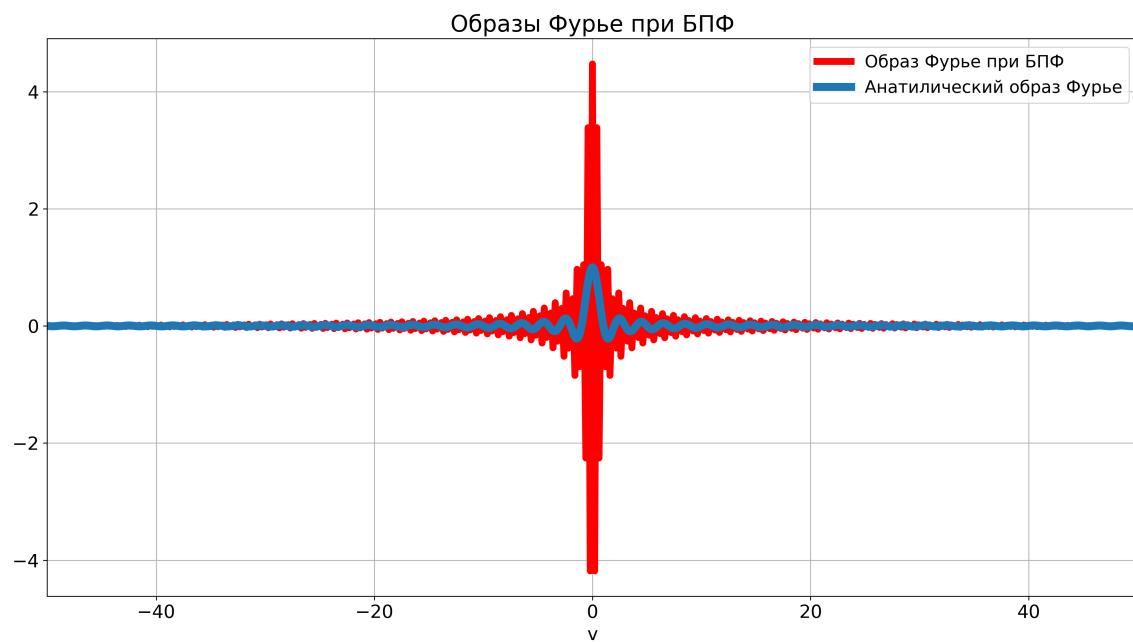


Рис. 41: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

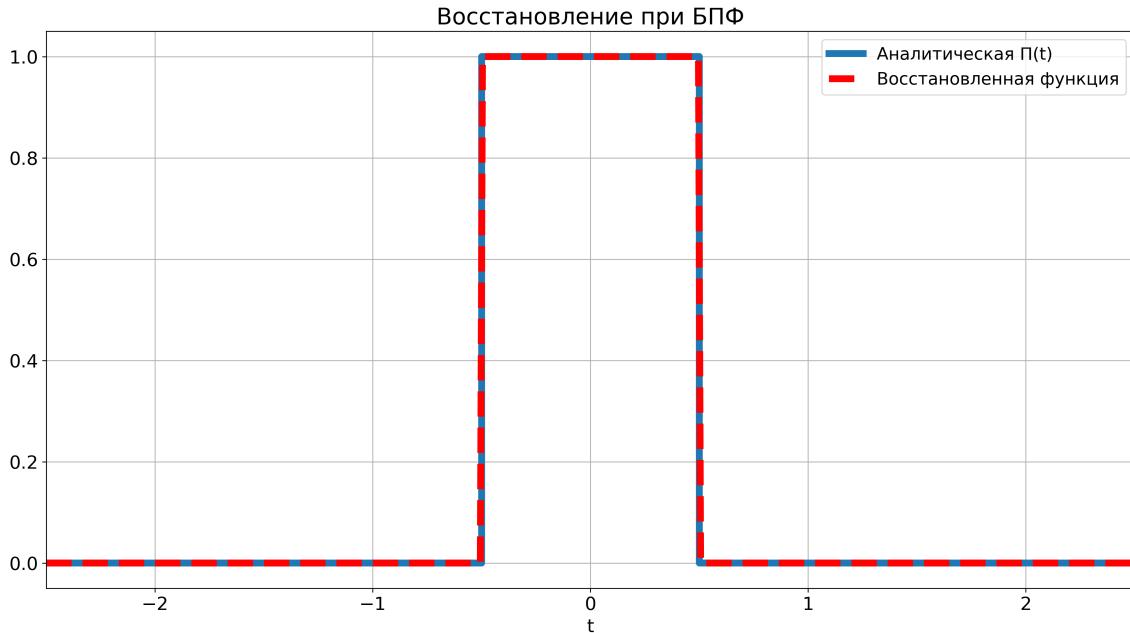


Рис. 42: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

при уменьшении  $\Delta t$  быть похожим на нечто непрерывное, задающееся при бесконечно малом шаге. Здесь нет ни поточечной, ни равномерной сходимостей, как в непрерывном случае - здесь явно берётся дискретизация, и чем чаще происходит её шаг, тем лучшее восстановление достигается, и в этом главное преимущество дискретного преобразования Фурье.

Стоит сказать и про быстродействие метода. Используемое *быстрое* преобразование Фурье оптимизирует вычисления, снижая сложность нахождения образа и восстановления с  $O(N^2)$  (при обычном дискретном преобразовании) до  $O(N \log(N))$ , где  $N$  - количество точек сэмплирования. Это даёт огромный выигрыш по времени в особенности при больших  $N$ . Отметим также и то, что у метода *trapz* вычислительная сложность *всего* образа Фурье равна  $O(N^2)$ , так что *fft* выигрывает и здесь.

### 1.3 Различия методов

Метод численного интегрирования работал сравнительно долго, так как использовал объемное количество операций, БПФ - быстро, ведь оптимизировал все вычисления. Как уже отмечалось, численному интегрированию удалось добиться успеха в приближении истинного образа Фурье, так как он предполагал его непрерывную природу, как и было получено аналитическое выражение; а дискретное преобразование потерпело здесь неудачу, как раз-таки потому что предполагало обратное, дискретное, и, соответственно, его и вычисляла.

Однако в восстановлении лучшие результаты показало БПФ. Метод численного интегрирования был ограничен в своих предположениях непрерывности, в отсутствии равномерной сходимости из-за разрыва функции он не смог точно передать

тот же скачок, с чем, напротив, успешно справилось дискретное преобразование, использовавшее сэмплирование.

## 1.4 Приближение непрерывного с помощью DFT

Идея метода - из сравнений непрерывного и дискретного преобразований выше стало ясно, что на временном отрезке нам выгоднее предполагать, что данные являются дискретными (работаем с дискретизацией), так как мы способны корректно обрабатывать большее количество классов функций; а на частотном отрезке - предполагать непрерывное, как и была получена аналитика, соответственно, получать лучшие её приближения. Выведем то, как должен действовать исследуемый метод.

Итак, преобразование Фурье на конечном промежутке  $T$  можно задать:

$$\hat{f}(\nu) = \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)e^{-2\pi i \nu t} dt.$$

Мы хотим его аппроксимировать, поэтому разобьём промежуток на  $N$  частей:

$$\Delta t = \frac{T}{N}, \quad t_n = -\frac{T}{2} + n\Delta t, \quad n = 0, 1, \dots, N-1.$$

Тогда интеграл можно приблизить суммой Римана:

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

Пусть  $\nu_m = \frac{m}{T}$  — дискретные частоты, согласованные с БПФ.

Подставим:

$$\hat{f}(\nu_m) \approx \Delta t \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i \nu_m t_n} = \Delta t \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i \frac{m}{T}(n\Delta t - \frac{T}{2})}.$$

Преобразуем выражение в показателе, исходя из факта  $T = N \cdot \Delta t$ :

$$= \Delta t \cdot e^{\pi i m} \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i \frac{nm}{N}}.$$

Таким образом:

$$\hat{f}(\nu_m) \approx c_m \cdot \text{DFT}_m(f),$$

где  $\text{DFT}_m(f) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(t_n)e^{-2\pi i \frac{nm}{N}}$ , а коэффициенты  $c_m = \sqrt{N} \Delta t \cdot (-1)^m$ .

Введенные коэффициенты  $c_m$  масштабируют дискретное преобразование Фурье, учитывая масштаб интеграла, и устраняют описанные в пункте про дискретное скачки. Метод работает основываясь на теории математического анализа: суммами Римана возможно аппроксимировать значение интеграла со сколь угодно заданной точ-

ностью, для этого необходимо уменьшать шаг дискретизации  $\Delta t$ . Коэффициенты  $c_m$  как раз сводят дискретный образ Фурье к суммам Римана, которые при заранее хорошо выбранном шаге  $\Delta t$  с данной точностью приближают значение вычисляемого интеграла образа в точке.

Метод берёт образ дискретного, быстрого преобразования Фурье и умножает на выведенные коэффициенты  $c_m$ , подстраивая его под непрерывную природу, а при восстановлении снимает всех их делением, выходя на обычное БПФ. То есть по своей сути это DTFT (discrete time fourier transform).

Так как выведенный метод DTFT является некоторой надстройкой над БПФ, то остаются и связи с параметрами по времени и частотам:  $V = \frac{1}{\Delta t}$ ,  $\Delta\nu = \frac{1}{T}$ . Теперь примем параметр  $\Delta t = 0.1$ ,  $V = 10$ . Вариация промежутка по времени  $T$  приведена при значениях 2, 6, 18 приведена на рисунках 43-48. Увеличение  $T$  влечет большие рассматриваемые отрезки, на которых задается восстанавливаемая функция. Также  $T$  обратно пропорционально влияет на  $\Delta\nu$ , делающей частотную сетку при своей мелкости более плотной. Соответственно, образы при росте  $T$  сглаживаются.

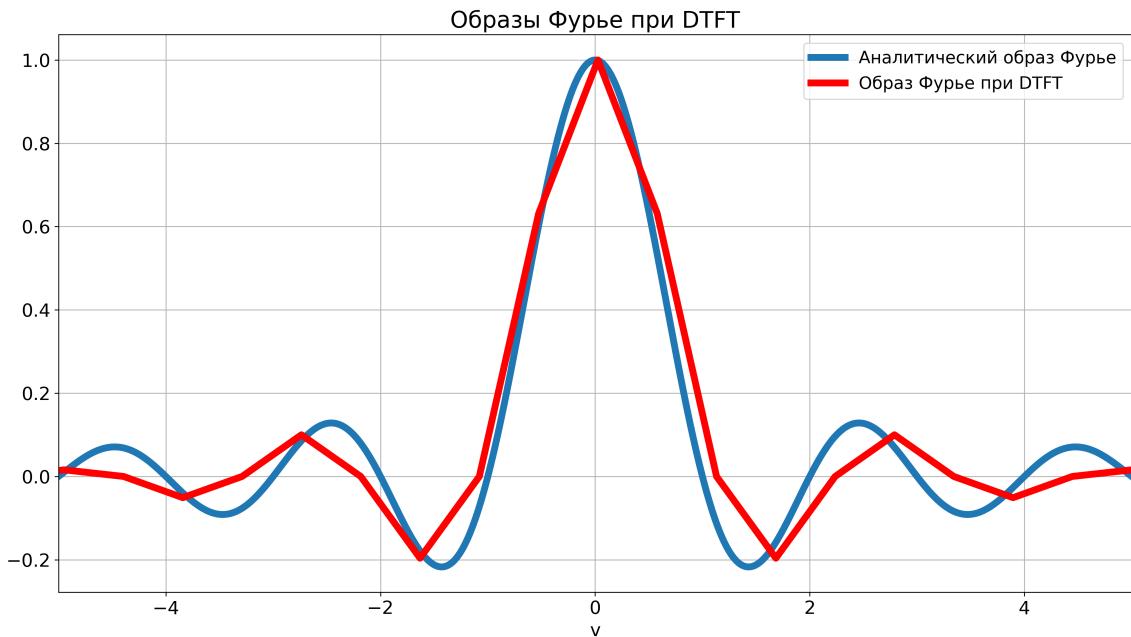


Рис. 43: Образы при DTFT:  $T = 2$ ,  $\Delta t = 0.1$

Примем  $T = 5$ ,  $\Delta\nu = 0.2$ . Вариация  $\Delta t$  при значениях 0.25, 0.05, 0.01 продемонстрирована на рисунках 49-54. Мелкость параметра шага дискретизации приводит к более точному сэмплированию, а значит, и восстановление лучше приближается к исходной функции  $\Pi(t)$  непрерывной природы на задаваемом промежутке по времени  $T$ . Из-за связи с суммами Римана уменьшение  $\Delta t$  также влечет лучшие аппроксимации интегралов, а значит, и более точные значения образов Фурье в задаваемых  $\nu_m = \frac{m}{T}$  точках. Малые шаги дискретизации приводят и к большей рассматриваемой области образов  $V$ .

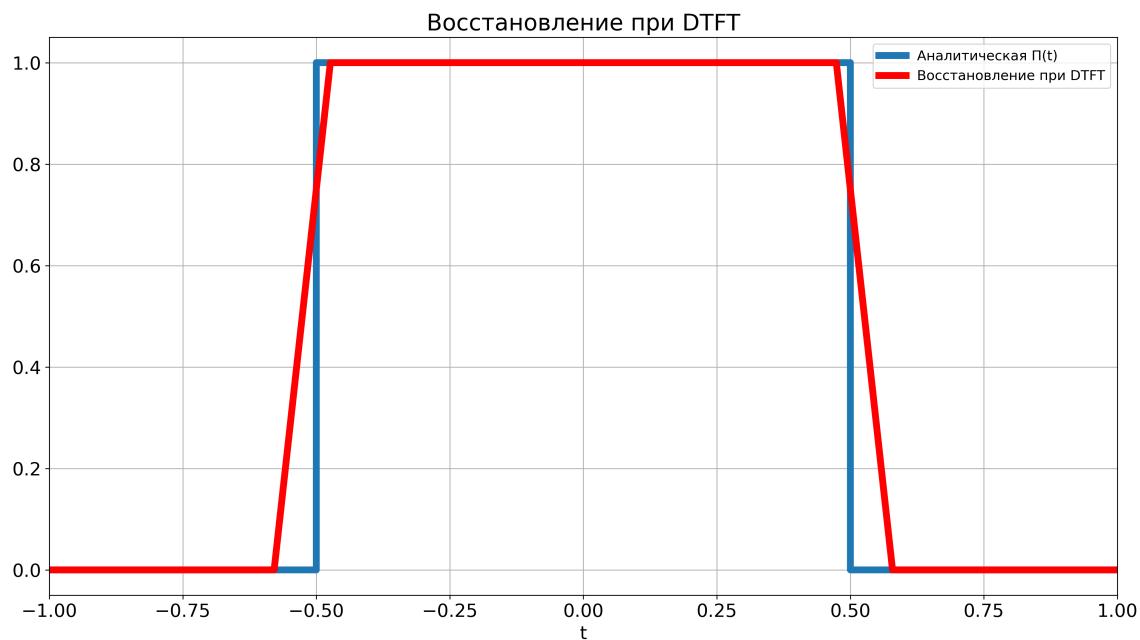


Рис. 44: Восстановление при DTFT:  $T = 2$ ,  $\Delta t = 0.1$

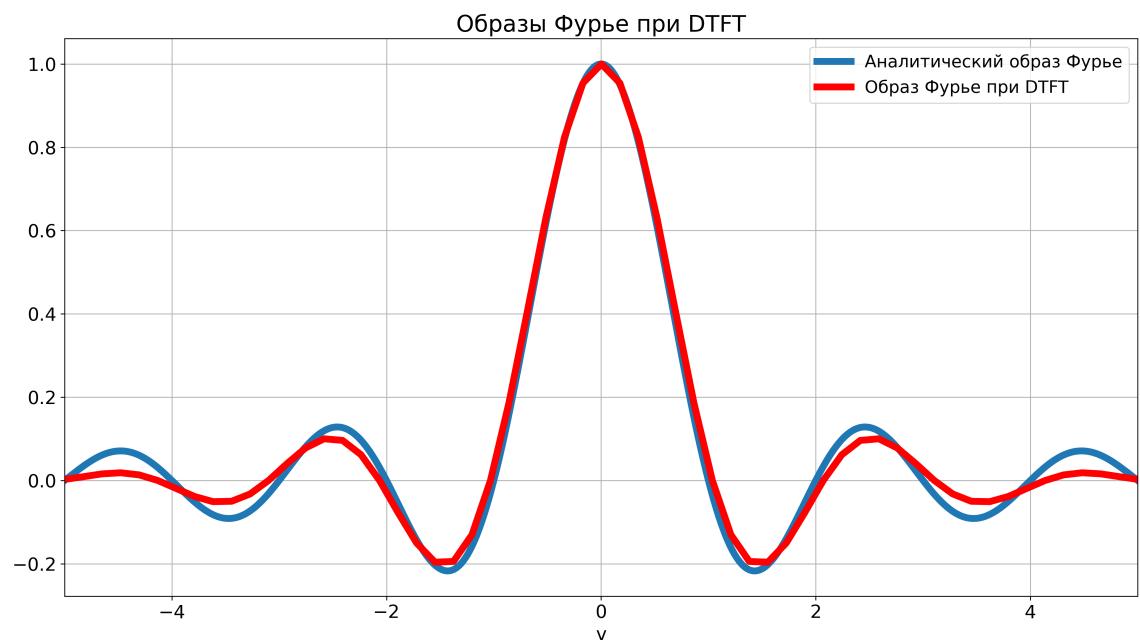


Рис. 45: Образы при DTFT:  $T = 6$ ,  $\Delta t = 0.1$

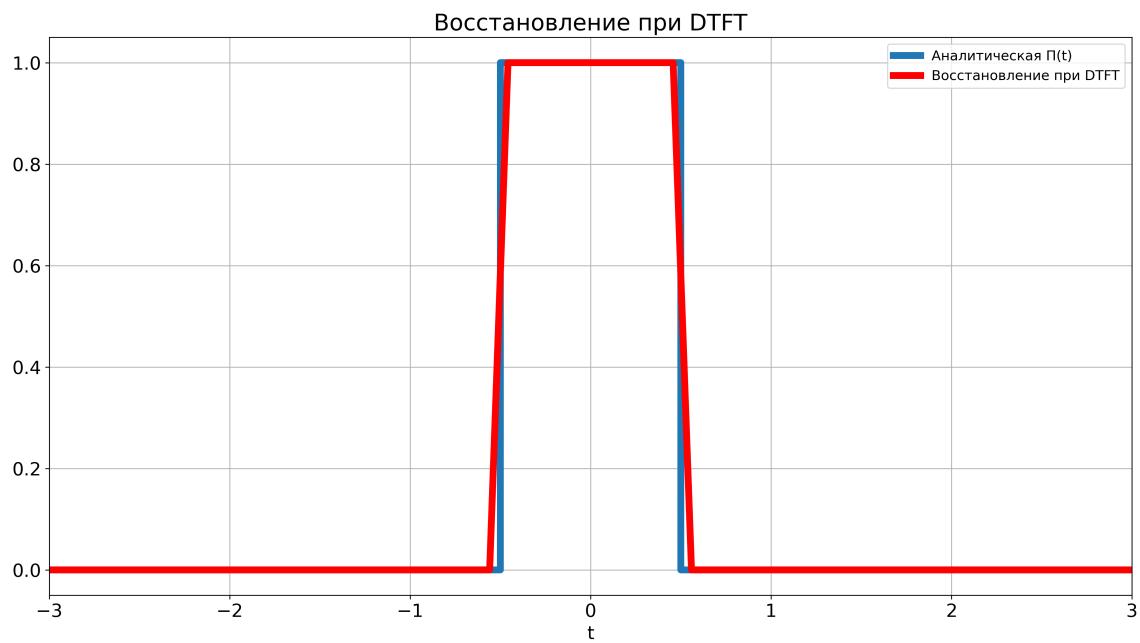


Рис. 46: Восстановление при DTFT:  $T = 6$ ,  $\Delta t = 0.1$

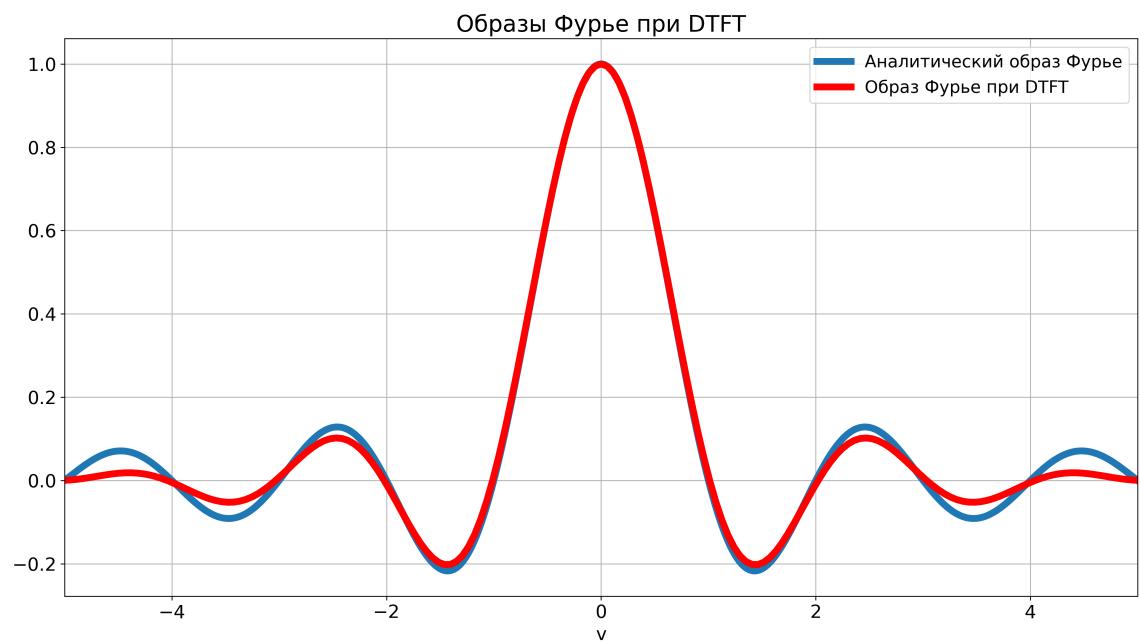


Рис. 47: Образы при DTFT:  $T = 18$ ,  $\Delta t = 0.1$

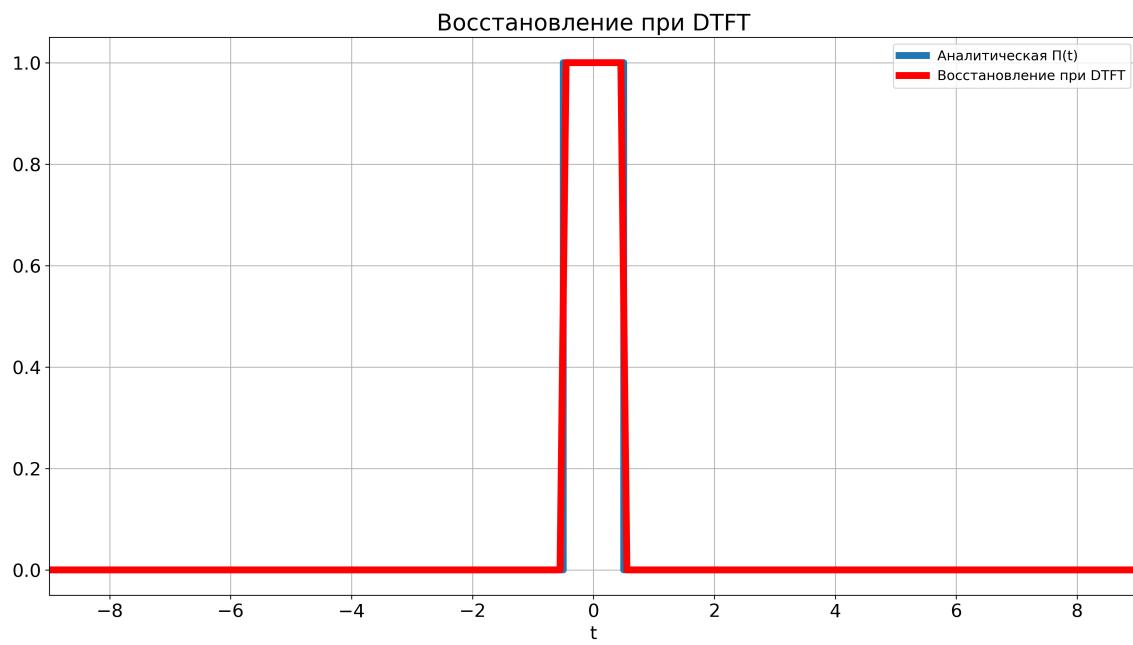


Рис. 48: Восстановление при DTFT:  $T = 18$ ,  $\Delta t = 0.1$

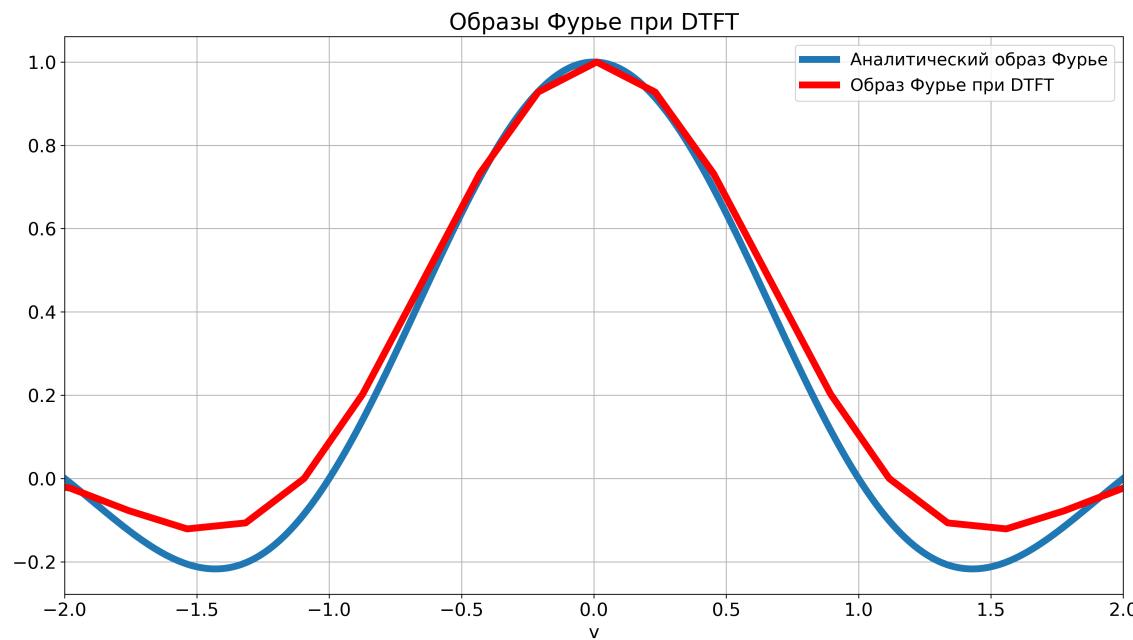


Рис. 49: Образы при DTFT:  $T = 5$ ,  $\Delta t = 0.25$

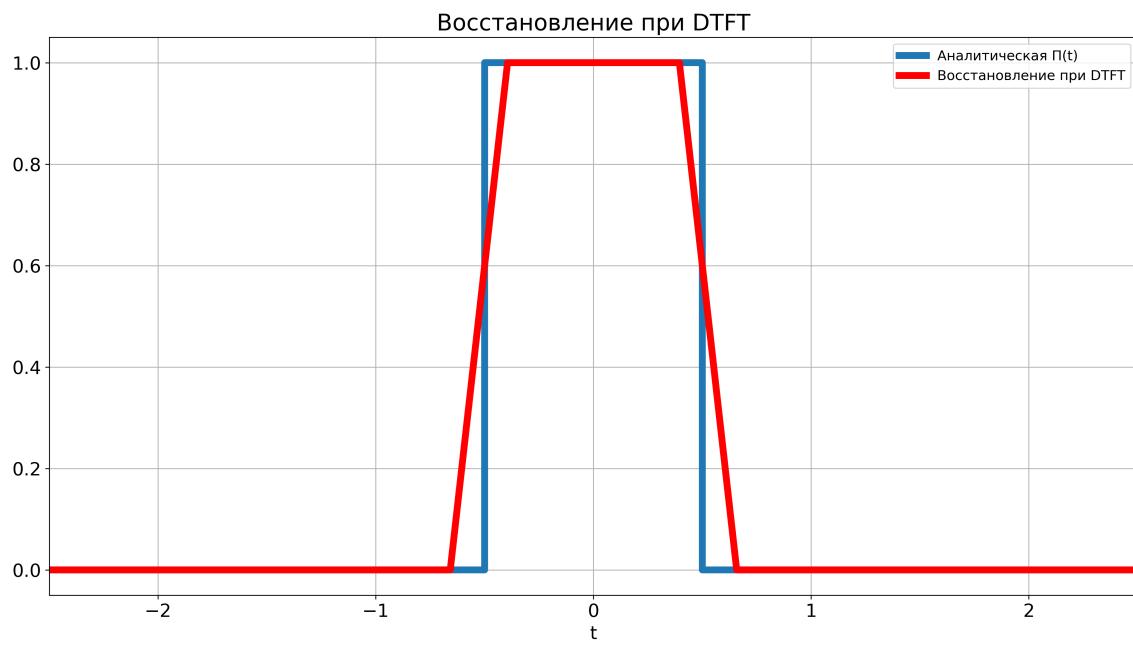


Рис. 50: Восстановление при DTFT:  $T = 5$ ,  $\Delta t = 0.25$

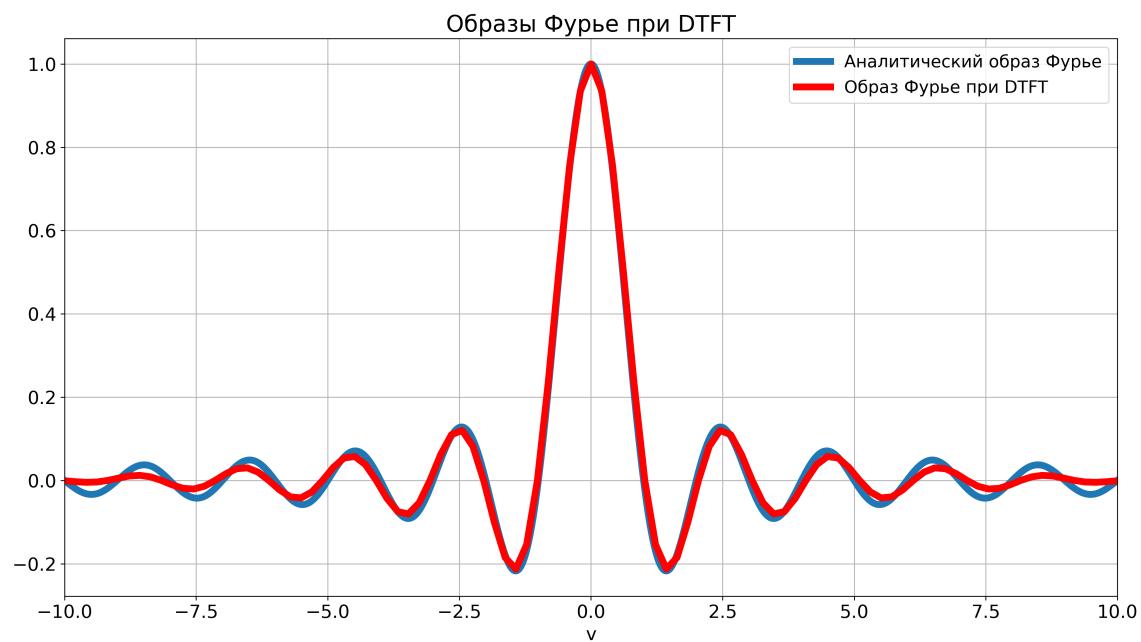


Рис. 51: Образы при DTFT:  $T = 5$ ,  $\Delta t = 0.05$

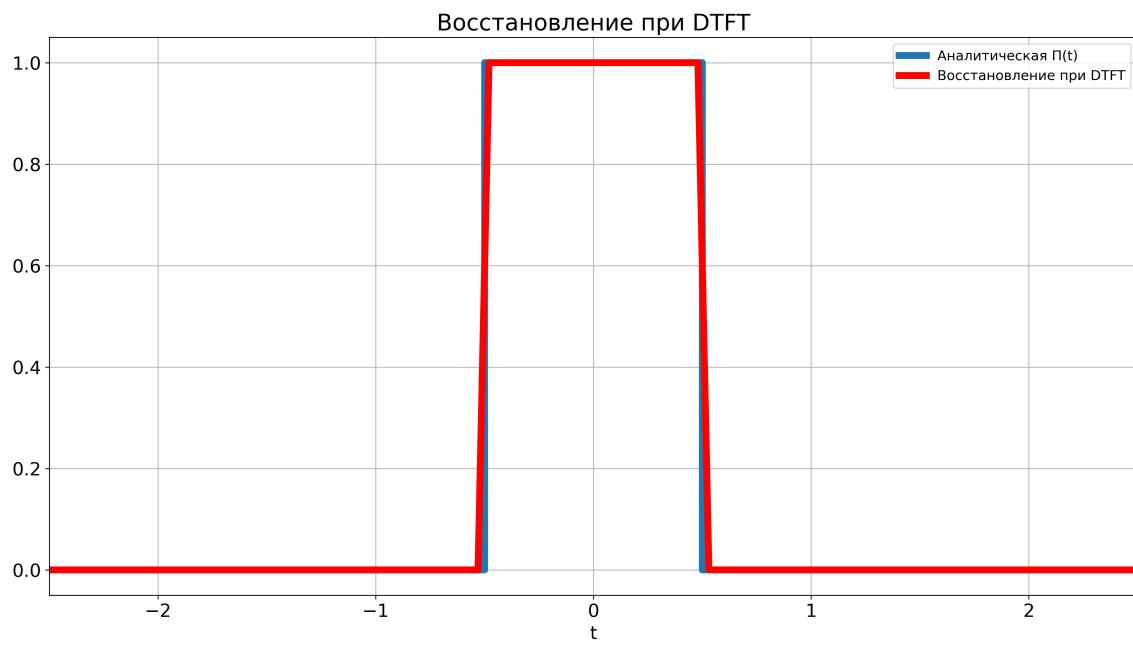


Рис. 52: Восстановление при DTFT:  $T = 5$ ,  $\Delta t = 0.05$

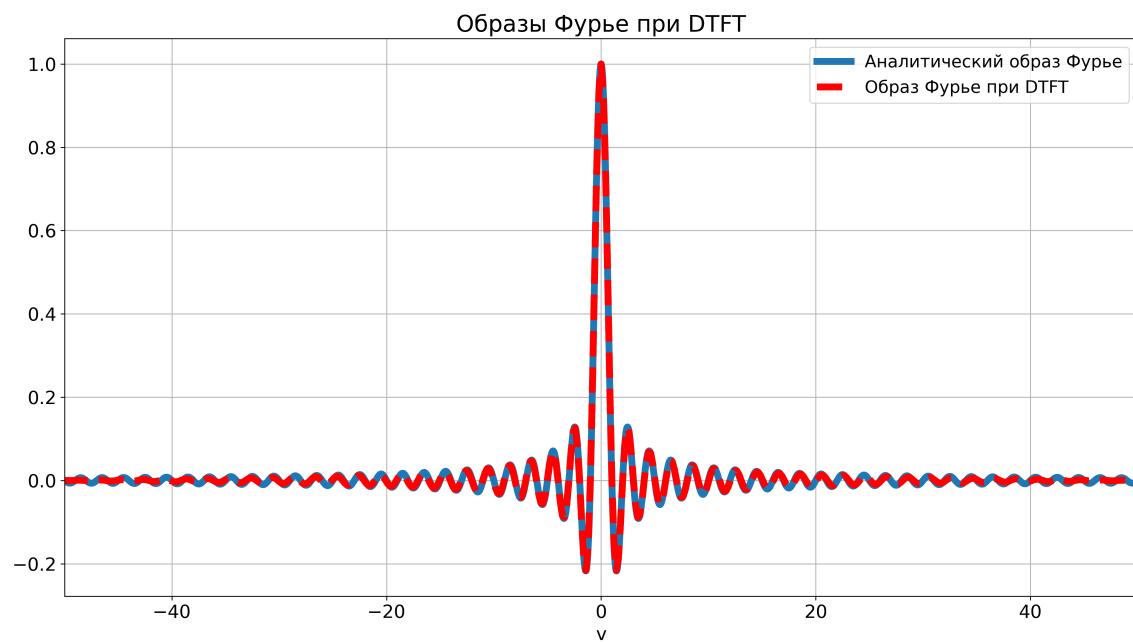


Рис. 53: Образы при DTFT:  $T = 5$ ,  $\Delta t = 0.01$

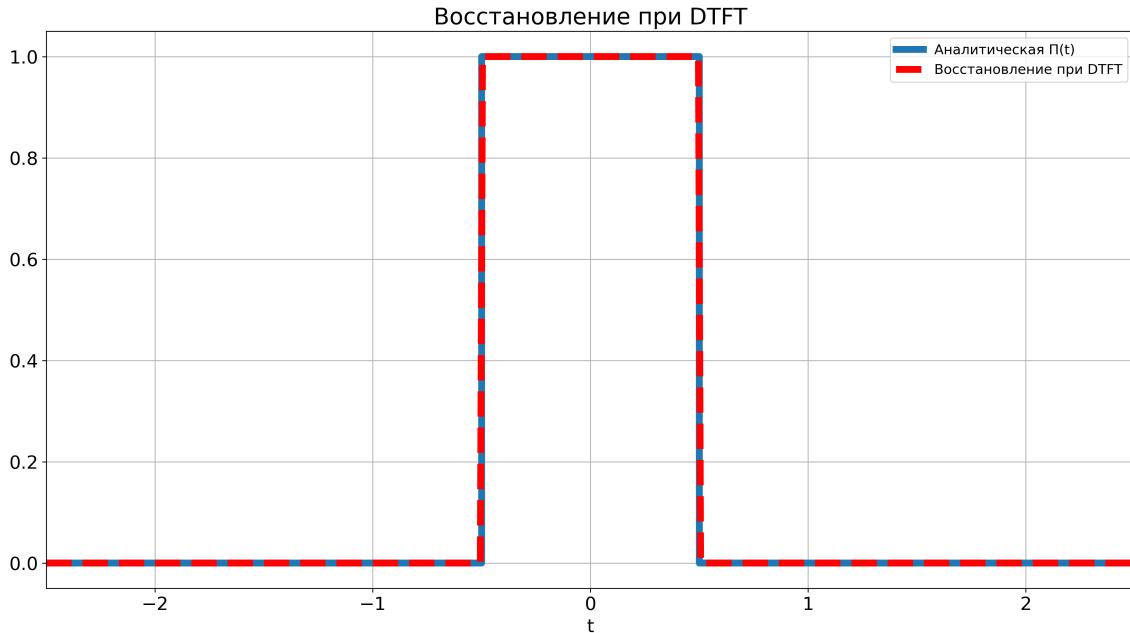


Рис. 54: Восстановление при DTFT:  $T = 5$ ,  $\Delta t = 0.01$

Сравнительные графики разработанного метода и алгоритмов из двух предыдущих пунктов при значениях параметров  $T = 2$ ,  $\Delta t = 0.1$  приведены на рисунках 55-57, при значениях  $T = 5$ ,  $\Delta t = 0.01$  - на рисунках 58-61. Можно видеть, что при малых  $T$  и большом шаге дискретизации  $\Delta t$  новый метод проигрывает в точности образов у численного интегрирования, однако с уменьшением  $\Delta t$  данная ситуация выравнивается, и DTFT при более быстром действии даёт практически идентичные графики в частотной области, при этом выигрывая в восстановлении. Также разработанный метод даёт идентичные результаты по времени в сравнении с БПФ, однако обладает лучшими Фурье-образами в смысле задачи приближения аналитически заданного непрерывного.

Новый метод крайне точен при малых шагах дискретизации  $\Delta t$  и больших промежутках по времени  $T$  как при вычислении образов, так и при восстановлении. Кроме того, DTFT существенно быстрее численного интегрирования, основанного на *trapz*, так как сохраняет оптимизацию быстрого преобразования Фурье, но может проигрывать в быстродействии у *fft*, ведь требует дополнительного вычисления коэффициентов  $c_m$  и их умножения на образ дискретного.

## 2 Сэмплирование

Зададимся параметрами  $a_1 = 1$ ,  $a_2 = 2$ ,  $\omega_1 = 8\pi$ ,  $\omega_2 = 16\pi$ ,  $\varphi_1 = \frac{\pi}{4}$ ,  $\varphi_2 = \frac{\pi}{3}$  и  $b = 10\pi$  рассмотрим функции

$$y_1(t) = a_1 \sin(\omega_1 t + \varphi_1) + a_2 \sin(\omega_2 t + \varphi_2), \quad y_2(t) = \text{sinc}(bt).$$

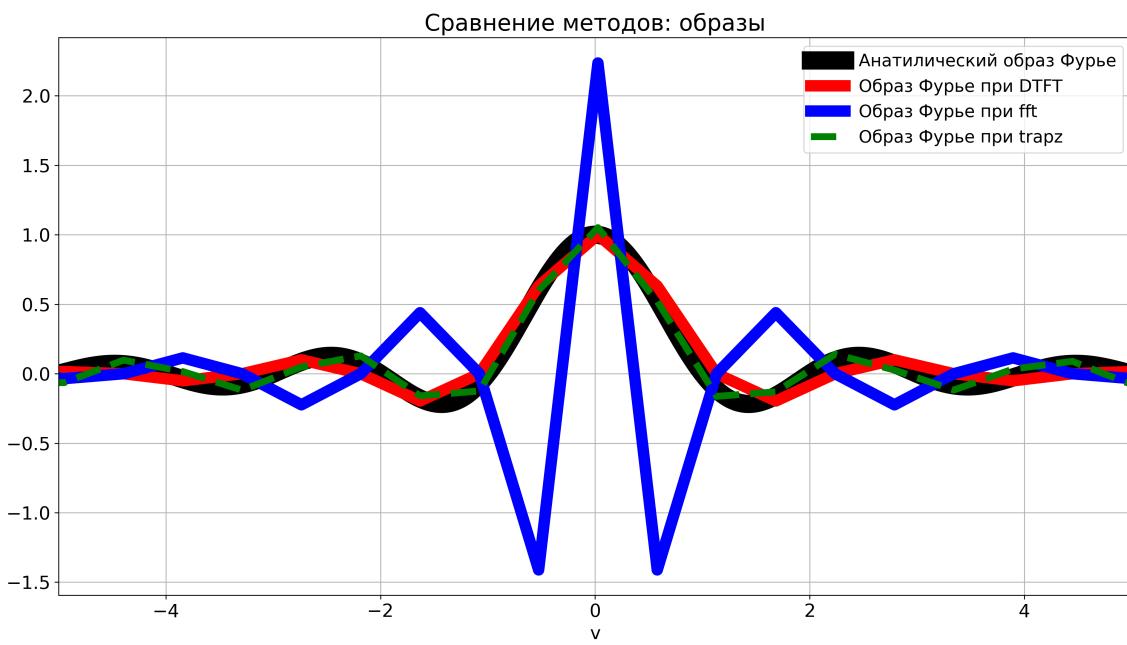


Рис. 55: Образы при быстром преобразовании Фурье:  $T = 2$ ,  $\Delta t = 0.1$

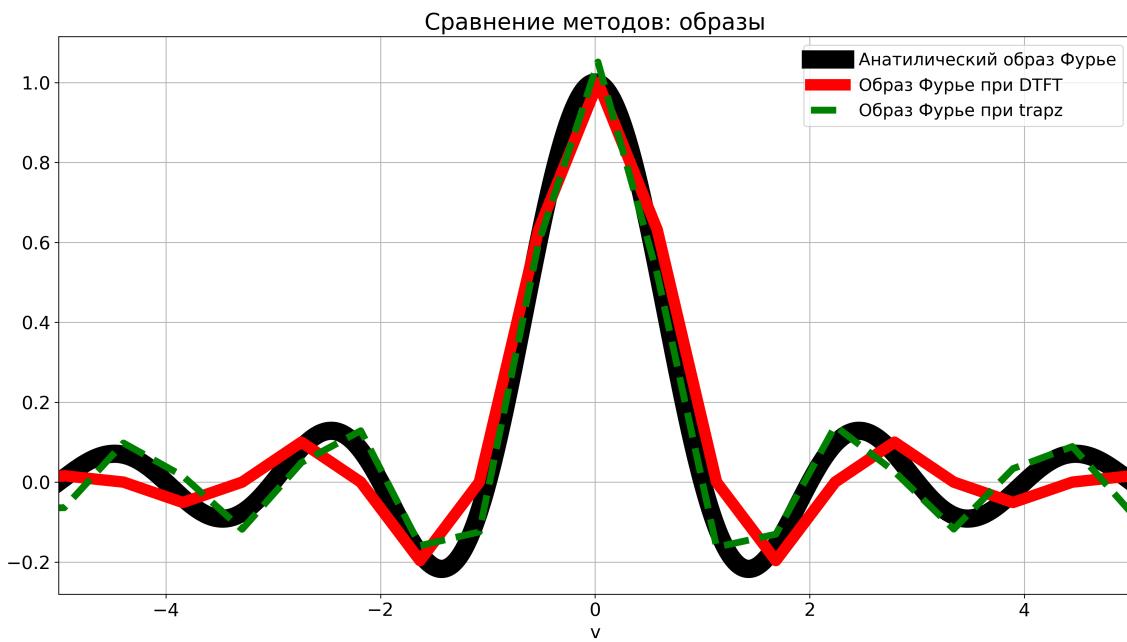


Рис. 56: Образы при быстром преобразовании Фурье:  $T = 2$ ,  $\Delta t = 0.1$

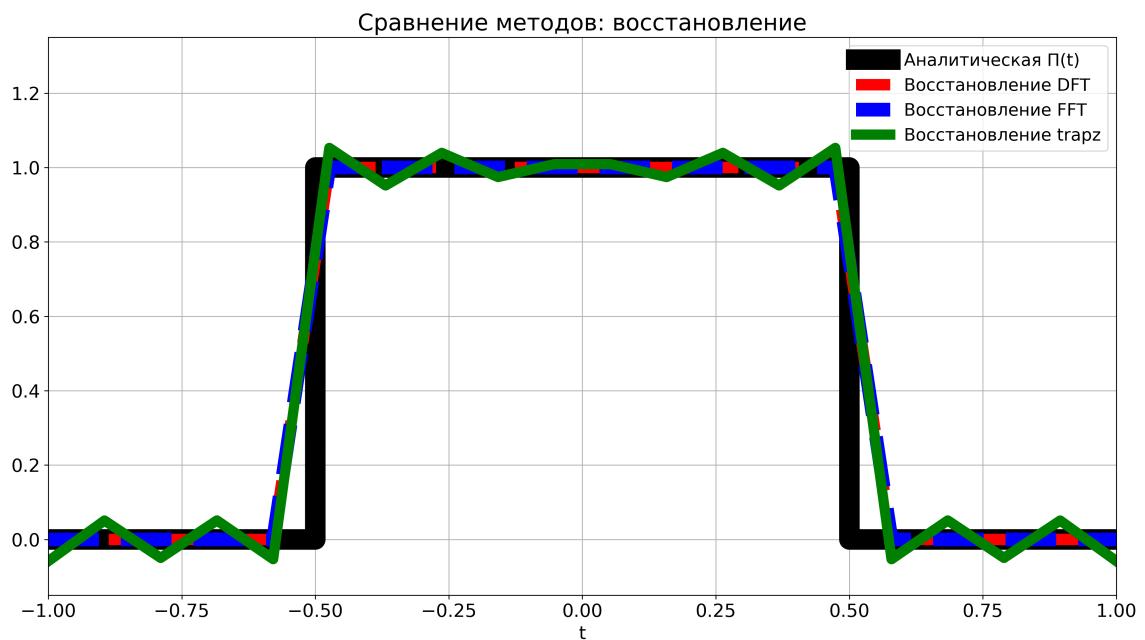


Рис. 57: Восстановление при быстром преобразовании Фурье:  $T = 2$ ,  $\Delta t = 0.1$

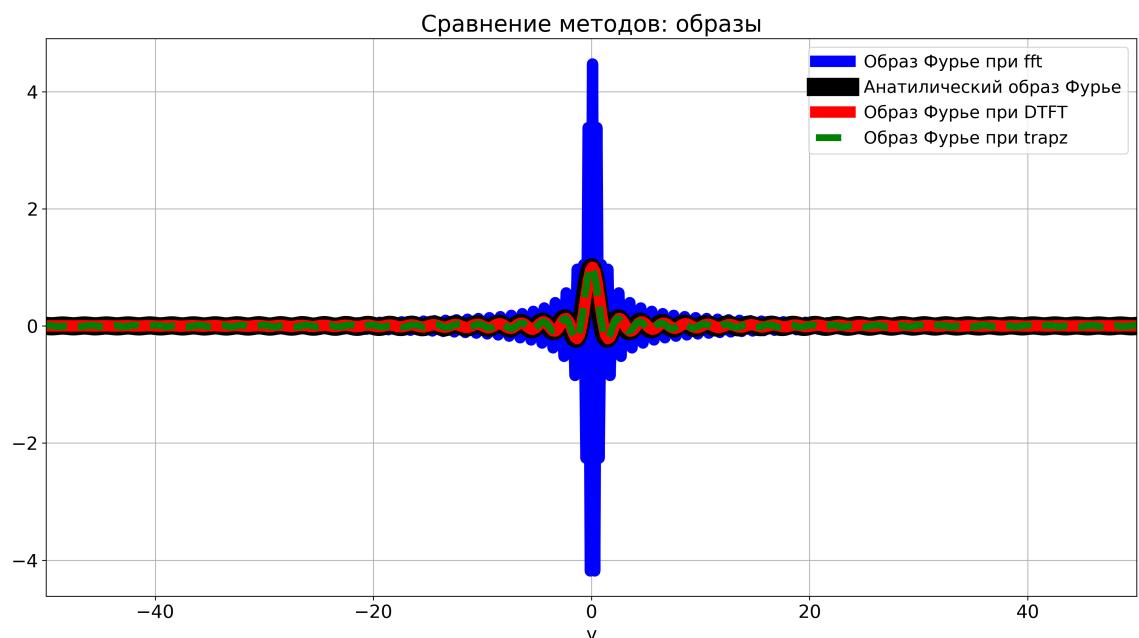


Рис. 58: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

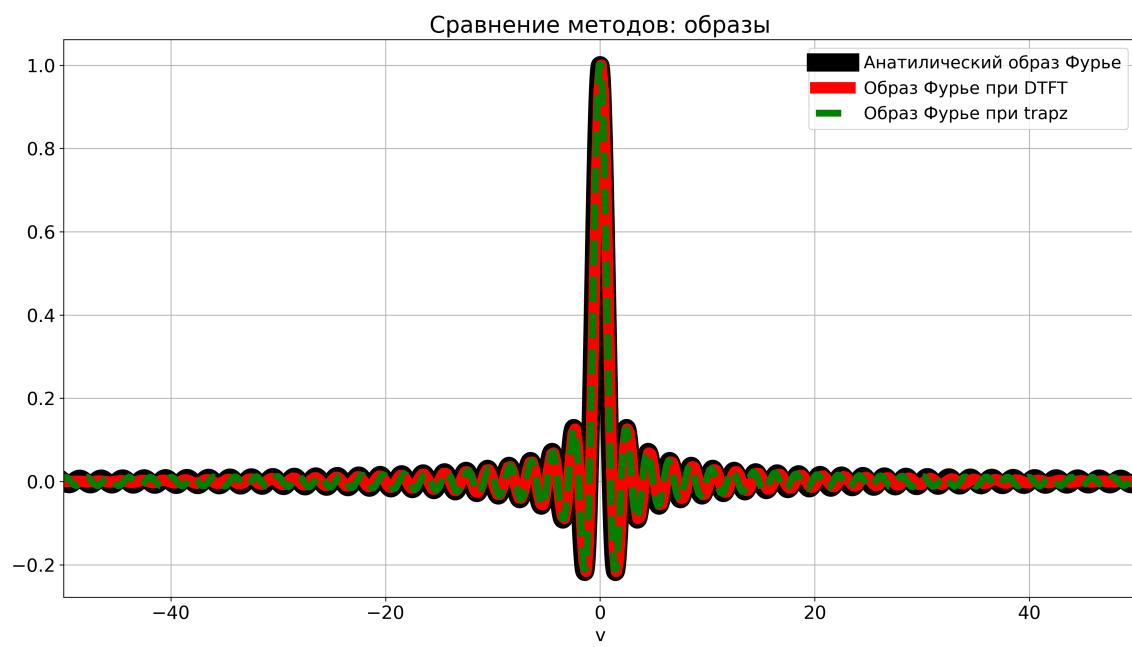


Рис. 59: Образы при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

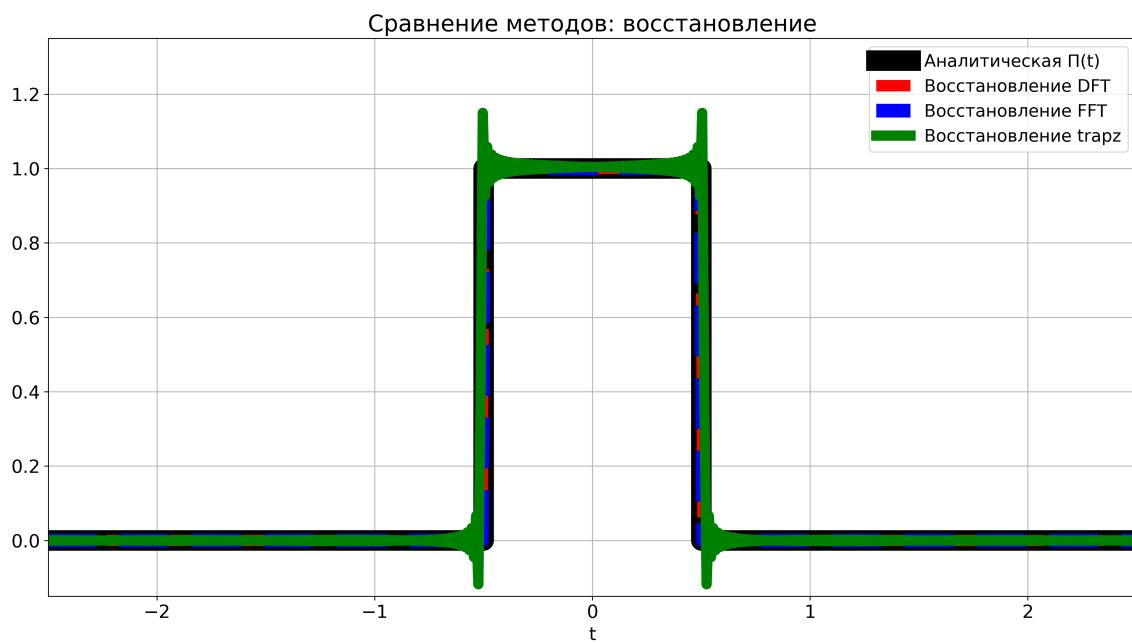


Рис. 60: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

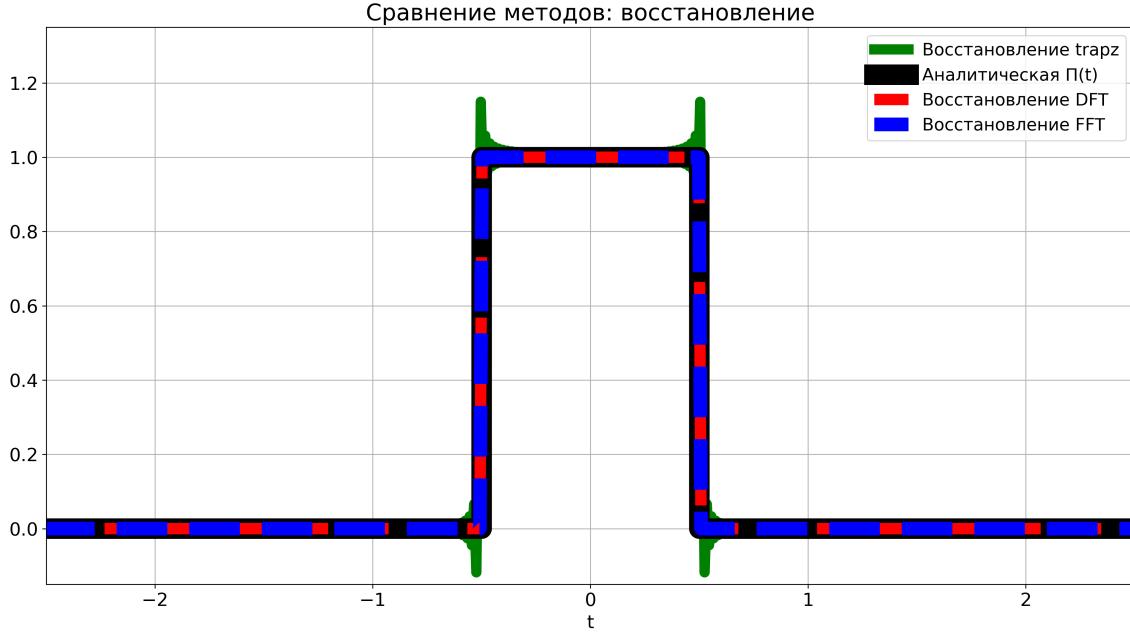


Рис. 61: Восстановление при быстром преобразовании Фурье:  $T = 5$ ,  $\Delta t = 0.01$

В этом пункте мы исследуем теорему Найквиста-Шеннона-Котельникова на примере двух этих функций. Суть теоремы в следующем:

Если непрерывный сигнал  $y(t)$  является ограниченным по спектру, т.е.

$$\hat{y}(\nu) = 0 \quad \text{при} \quad |\nu| > \nu_{max},$$

то он может быть полностью восстановлен по своему сэмплированию с шагом дискретизации

$$\Delta t < \frac{1}{2\nu_{max}}.$$

Формула для восстановления при этом имеет вид:

$$y(t) = \sum_{n=-\infty}^{\infty} y(n\Delta t) \operatorname{sinc}\left(\frac{\pi}{\Delta t}(t - n\Delta t)\right) = \sum_{n=-\infty}^{\infty} y(t_n) \operatorname{sinc}\left(\frac{\pi}{\Delta t}(t - t_n)\right),$$

где  $\operatorname{sinc}(x) = \frac{\sin(x)}{x}$ .

Следствием является тот факт, что при  $\Delta t > \frac{1}{2\nu_{max}}$  возникает наложение спектров (*aliasing*), и точное восстановление невозможно.

Проверим теорему в действии при функции  $y_1(t) = \sin(8\pi t + \frac{\pi}{4}) + 2 \sin(16\pi t + \frac{\pi}{3})$ , график которой изображен на рисунке 62 и которая имеет ограниченный спектр ( $\nu_{max} = 8$  - максимальная частота двух синусоид, превращающихся в дельты при преобразовании Фурье), то есть одно из условий теоремы уже выполняется. Далее возьмем знакомые параметры  $T = 0.75$  и  $\Delta t = 0.025 = \frac{1}{40} < \frac{1}{2\nu_{max}} = \frac{1}{16}$ , отвечающими за промежуток сэмплирования по времени и его шаг, и проведем дискретизацию, после чего применим интерполяционную формулу и попытаемся восстановить непре-

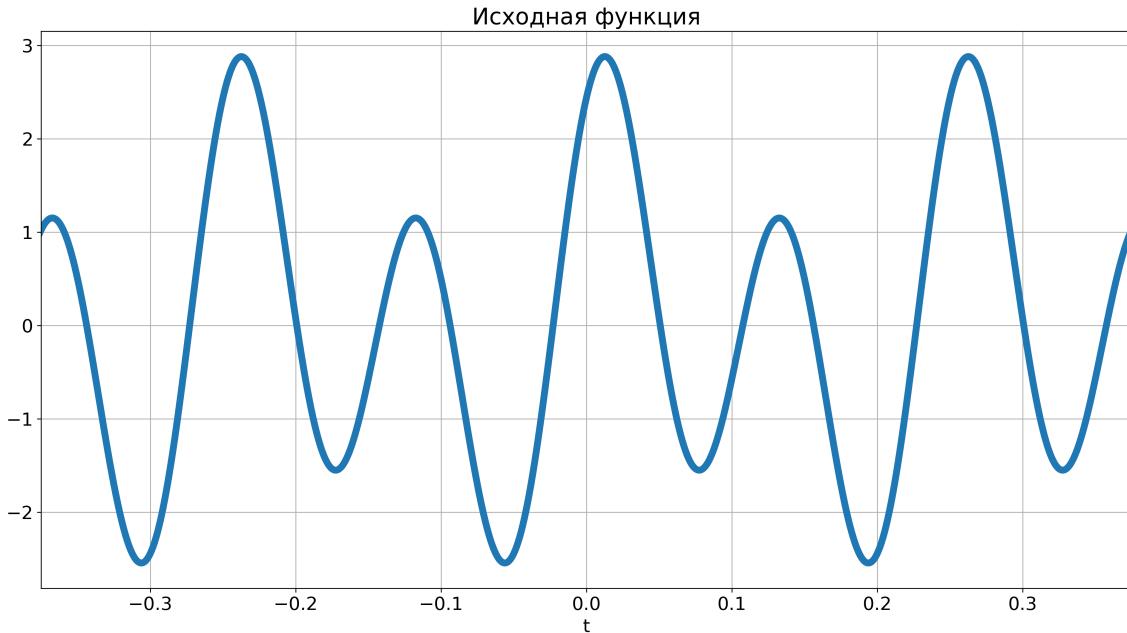


Рис. 62: График функции  $y_1(t)$

рывную функцию. Работа продемонстрирована на рисунках 63, 64, красным пунктиром на образах выделена та самая максимальная частота  $\nu_{max} = B$ ; отвечающие временным параметрам частотные равны  $V = 40$ ,  $\Delta\nu = 1.33$ .

Можем видеть практически идеальное сходство восстановления и непрерывной функций. Небольшие различия связаны с ограниченностью применения теоремы - в реальности задать кардинальный синус на бесконечно большом промежутке не представляется возможным, равно как и задание бесконечного числа точек неосуществимо на деле. По последней причине также точного представления на программном уровне исходной функции невозможно, и используется не сама формула интерполяции, а её модификация, подразумевающая сколь угодно точное приближение

$$y(t) \approx \sum_{n=0}^N y(t_n) \operatorname{sinc}\left(\frac{\pi}{\Delta t}(t - t_n)\right), \text{ где } t_n \text{ - точки дискретизации.}$$

При больших шагах сэмплирования  $\Delta t > \frac{1}{2\nu_{max}}$  приближения не достигается, так как возникает эффект наложения низких частот на высокие, мало периодический образ начинаетискажать используемый частотный отрезок  $[-\nu_{max}, \nu_{max}]$ , что демонстрируется на рисунках 65, 66, сделанных при  $T = 0.75$ ,  $\Delta t = \frac{1}{15} > \frac{1}{2\nu_{max}} = \frac{1}{16}$  и соответствующих им  $V = 15$ ,  $\Delta\nu = 1.33$ .

Увеличение параметра  $T$  приводит к увеличению рассматриваемой области сэмплирования и восстановления. Также из-за связи с  $\Delta\nu$  при возрастании промежутка сэмплирования сглаживаются вычисляемые образы Фурье. Работа при взятых параметрах  $T = 5$ ,  $\Delta t = \frac{1}{40}$ ,  $V = 40$ ,  $\Delta\nu = \frac{1}{5} = 0.2$  изображена на рисунках 67, 68. Можем видеть, что восстановление не теряет в точности, а вот промежуток расши-

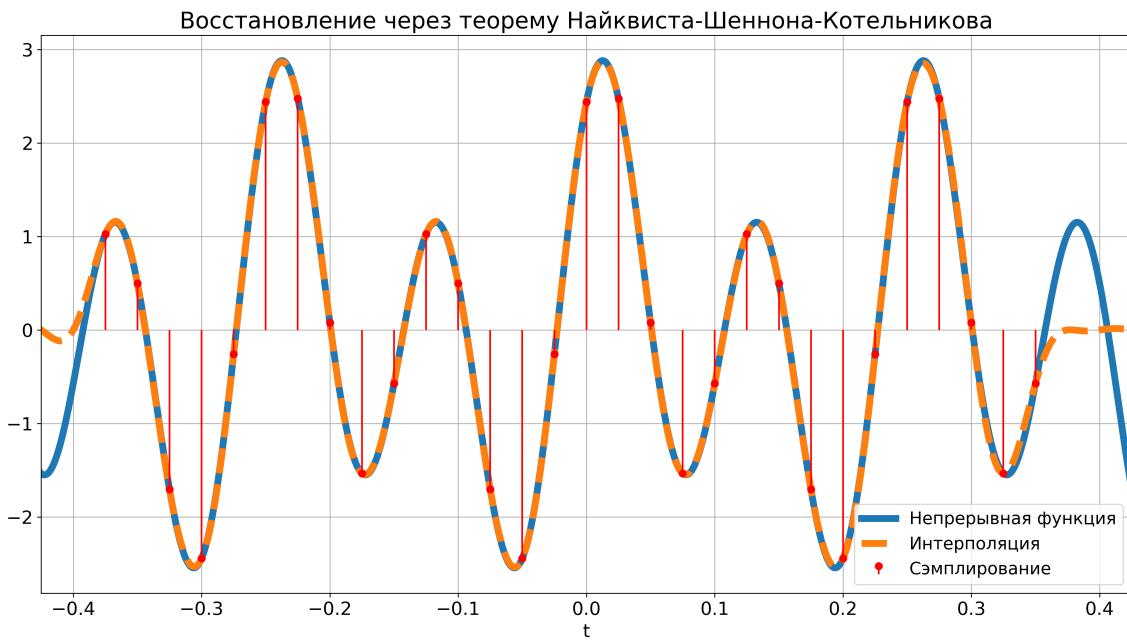


Рис. 63: Применение теоремы восстановления:  $T = 0.75$ ,  $\Delta t = \frac{1}{40}$

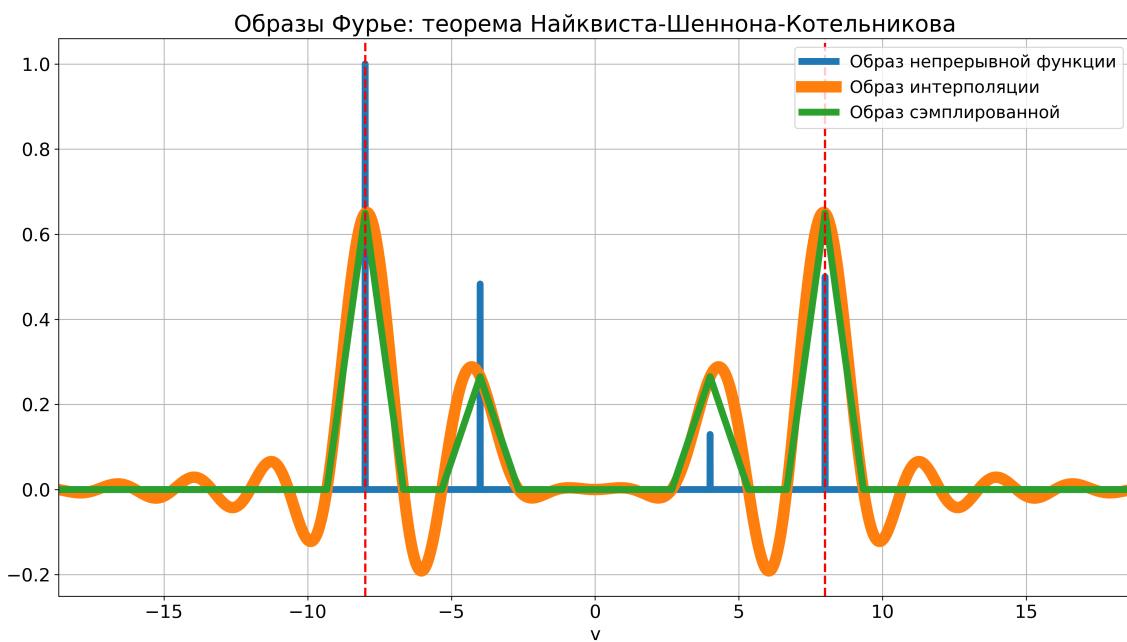


Рис. 64: Образы Фурье при применении теоремы:  $T = 0.75$ ,  $\Delta t = \frac{1}{40}$

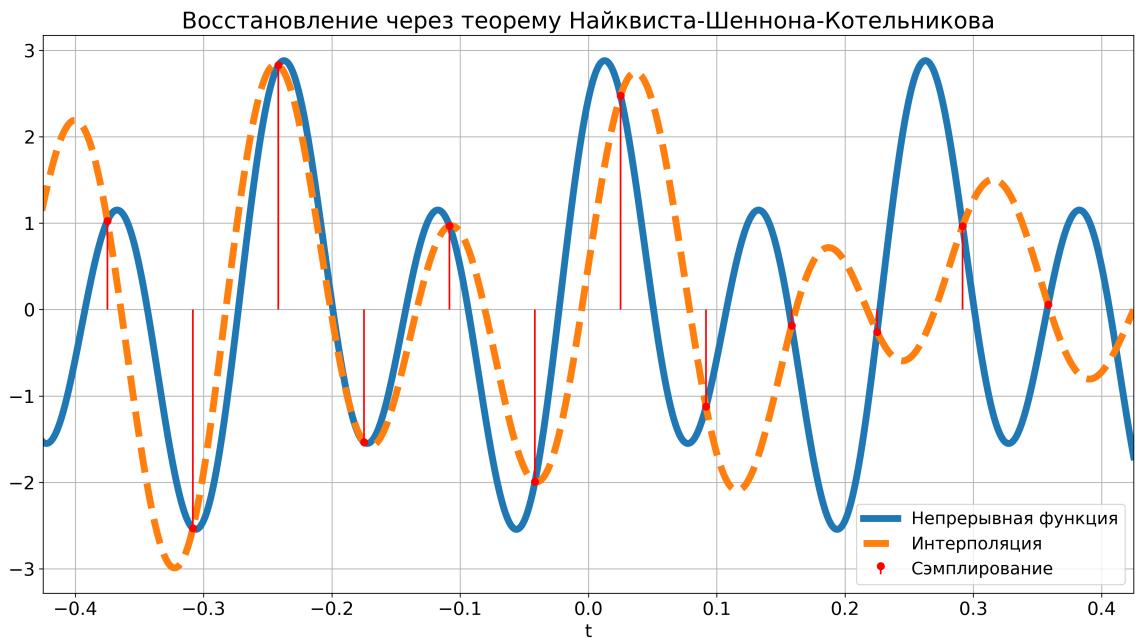


Рис. 65: Применение теоремы восстановления:  $T = 0.75$ ,  $\Delta t = \frac{1}{15}$

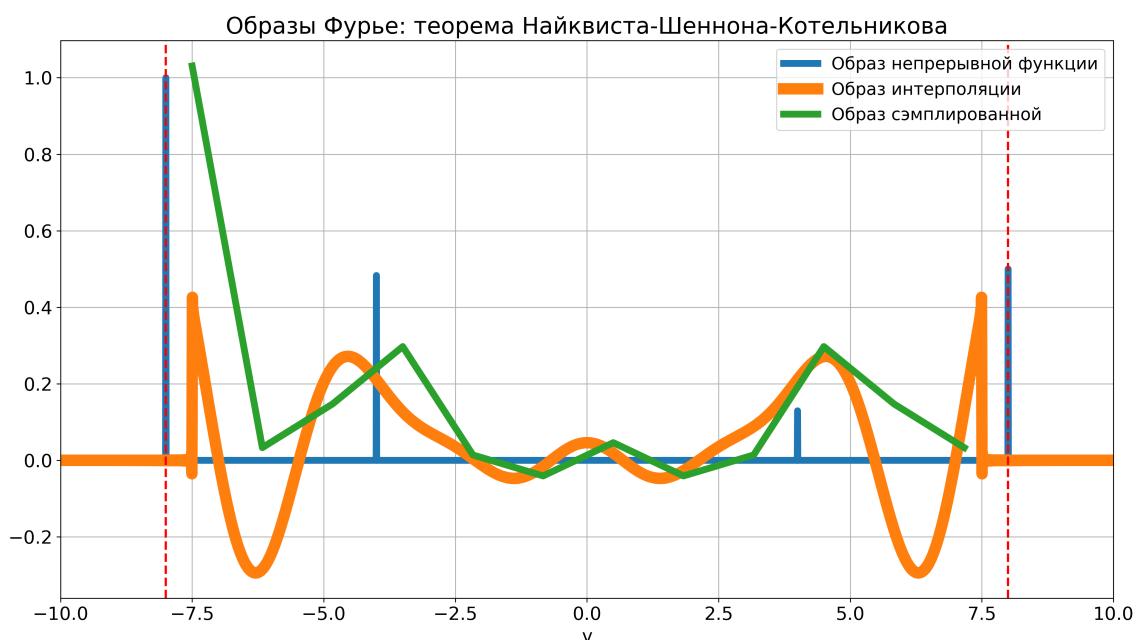


Рис. 66: Образы Фурье при применении теоремы:  $T = 0.75$ ,  $\Delta t = \frac{1}{15}$

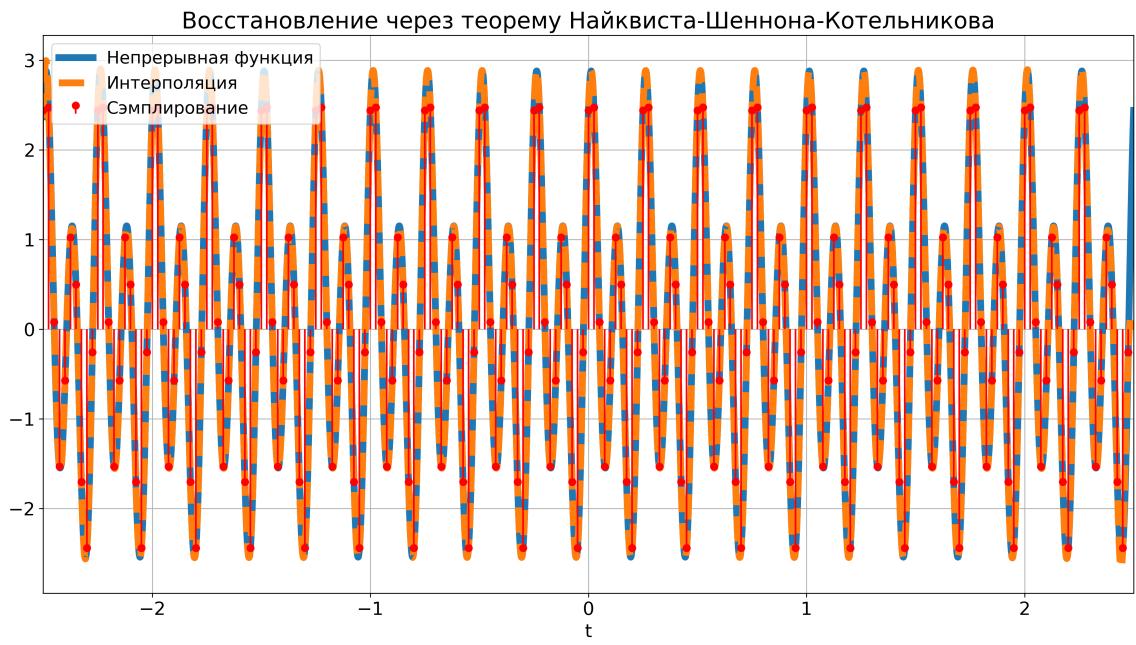


Рис. 67: Применение теоремы восстановления:  $T = 5$ ,  $\Delta t = \frac{1}{40}$



Рис. 68: Образы Фурье при применении теоремы:  $T = 5$ ,  $\Delta t = \frac{1}{40}$

ряется. Таким образом, эффективность метода действительно зависит только от шага сэмплирования. При малых  $\Delta t$  и больших  $T$  образ сэмплирования и интерполяции также стремится к образу непрерывной функции, что согласуется с полученными в пункте 1.4 результатами (все образы при проверке теоремы считаются с помощью полученного DTFT).

В конце рассмотрим применение теоремы при  $y_2(t) = \text{sinc}(10\pi t)$  (график изображен на рисунке 69), образом Фурье которой является знакомая прямоугольная функция расширенная  $\Pi(t)$ , которая зануляется при  $\nu > \nu_{max} = 5$ , а значит, важное условие применения теоремы выполняется.

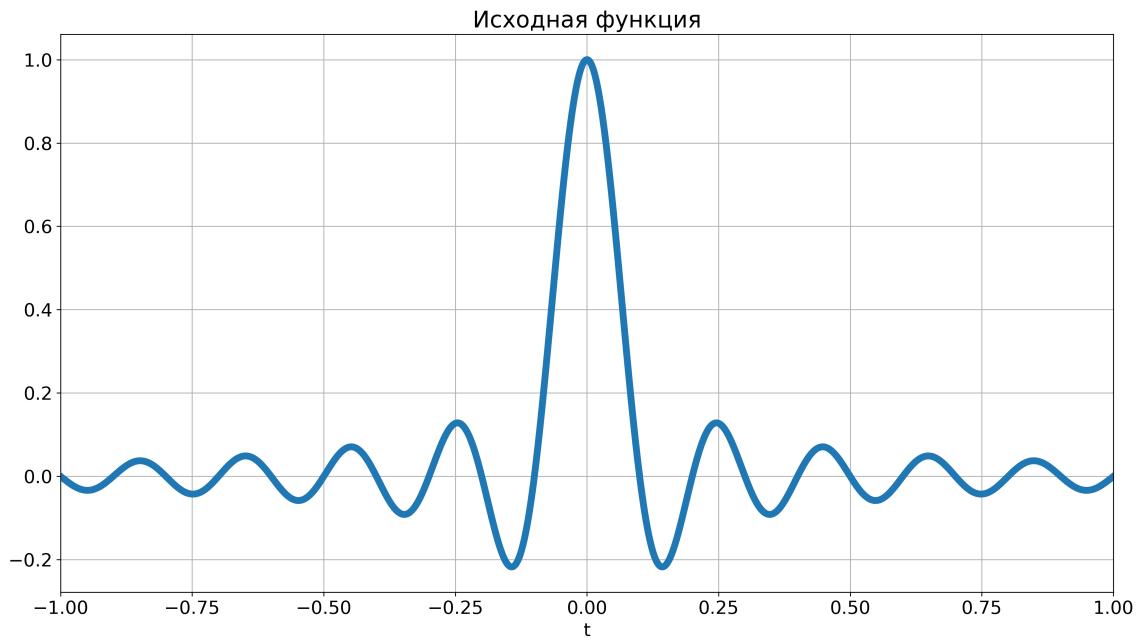


Рис. 69: График функции  $y_2(t)$

Зададимся теперь параметрами  $T = 2$ ,  $\Delta\nu = 0.5$ . Вариация шага сэмплирования  $\Delta t$  при значениях  $0.25 = \frac{1}{4} > \frac{1}{10}$  и  $0.05 = \frac{1}{20} < \frac{1}{10}$  продемонстрирована на рисунках 70-73. Как и было сказано в теореме, при больших шагах сэмплирования приближения не достигается, при малых - всё с небольшой погрешностью равняется друг другу.

Промежуток сэмплирования  $T$  всё так же прямо влияет на задаваемый отрезок сэмплирования и угущает частотную сетку за счёт увеличения  $\Delta\nu = \frac{1}{T}$ , придавая образам более непрерывный вид. Графики, показывающие данное, изображены на рисунках 74, 75 при взятых параметрах  $T = 5$ ,  $\Delta t = \frac{1}{20}$ ,  $V = 20$ ,  $\Delta\nu = 0.2$ .

В итоге теорема Найквиста-Шеннона-Котельникова применима в ситуациях ограничения используемой памяти - с её помощью мы можем восстановить непрерывное с любой точностью с помощью сэмплов, заданных с определенным шагом, при том единственном условии, что весь значимый, ненулевой образ функции содержится в определенном частотном отрезке. Теорема является крайне важной, ведь дает мощный инструмент работы с сигналами.

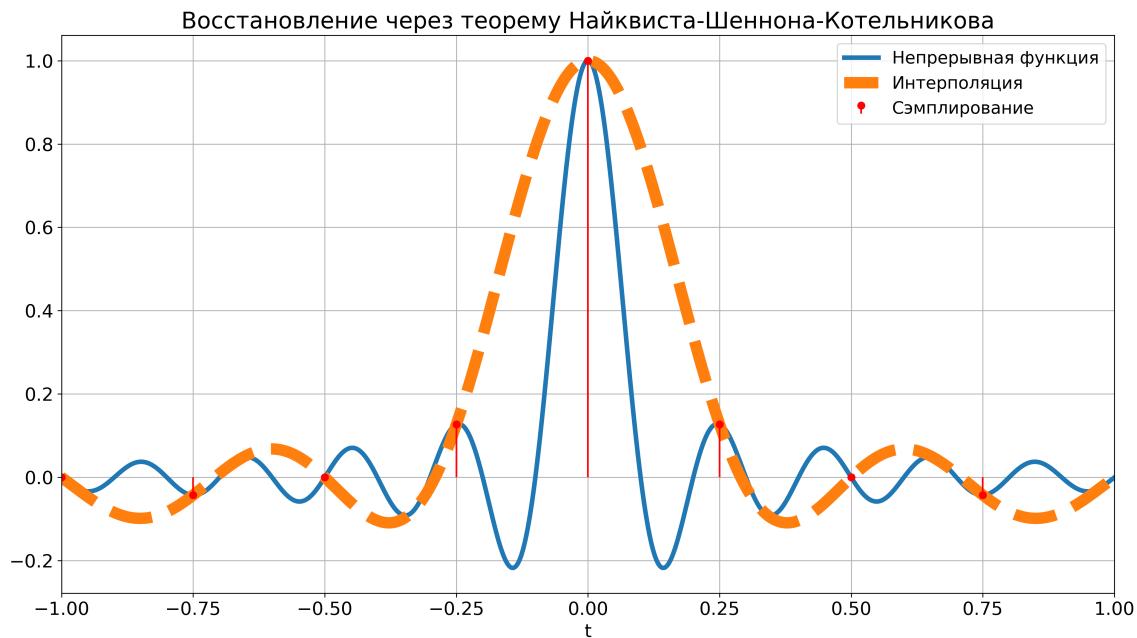


Рис. 70: Применение теоремы восстановления:  $T = 2$ ,  $\Delta t = \frac{1}{4}$

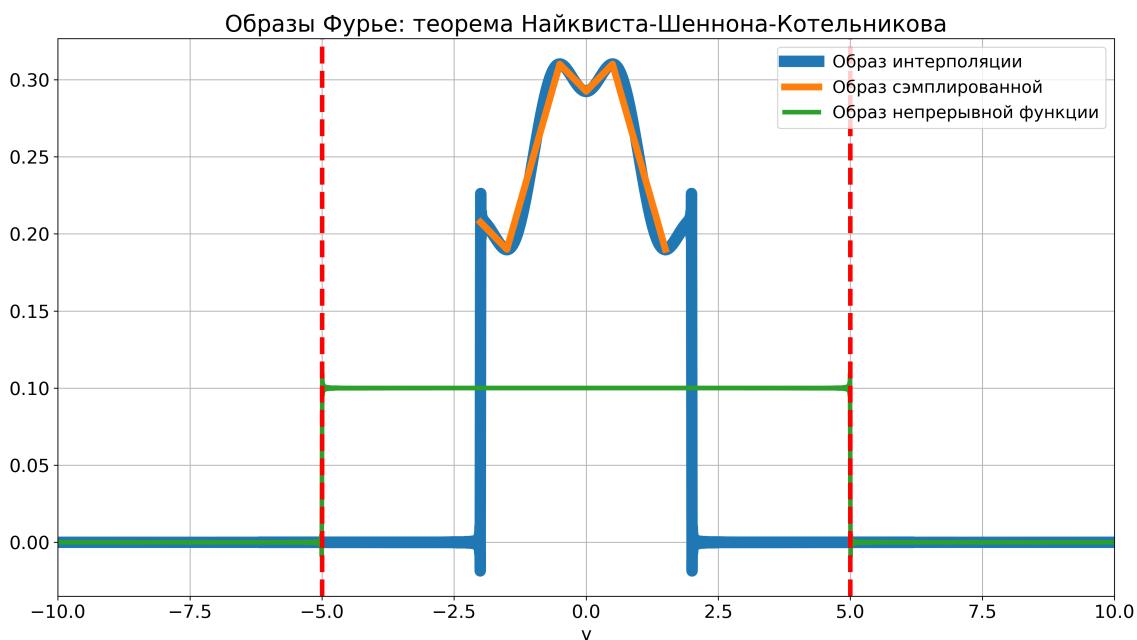


Рис. 71: Образы Фурье при применении теоремы:  $T = 2$ ,  $\Delta t = \frac{1}{4}$

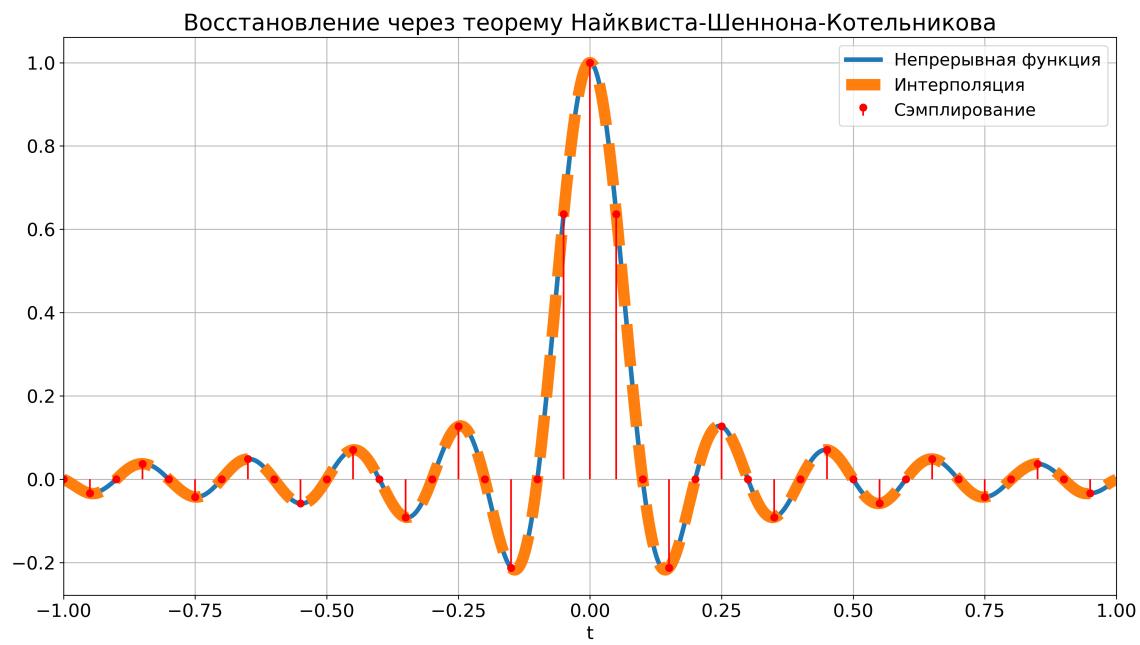


Рис. 72: Применение теоремы восстановления:  $T = 2$ ,  $\Delta t = \frac{1}{20}$

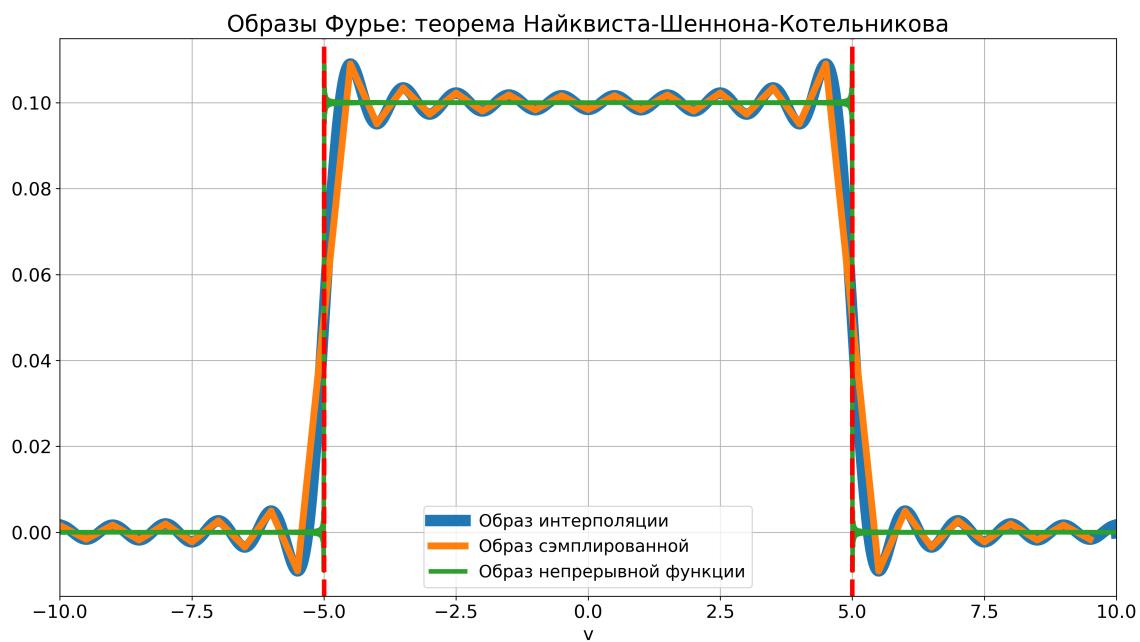


Рис. 73: Образы Фурье при применении теоремы:  $T = 2$ ,  $\Delta t = \frac{1}{20}$

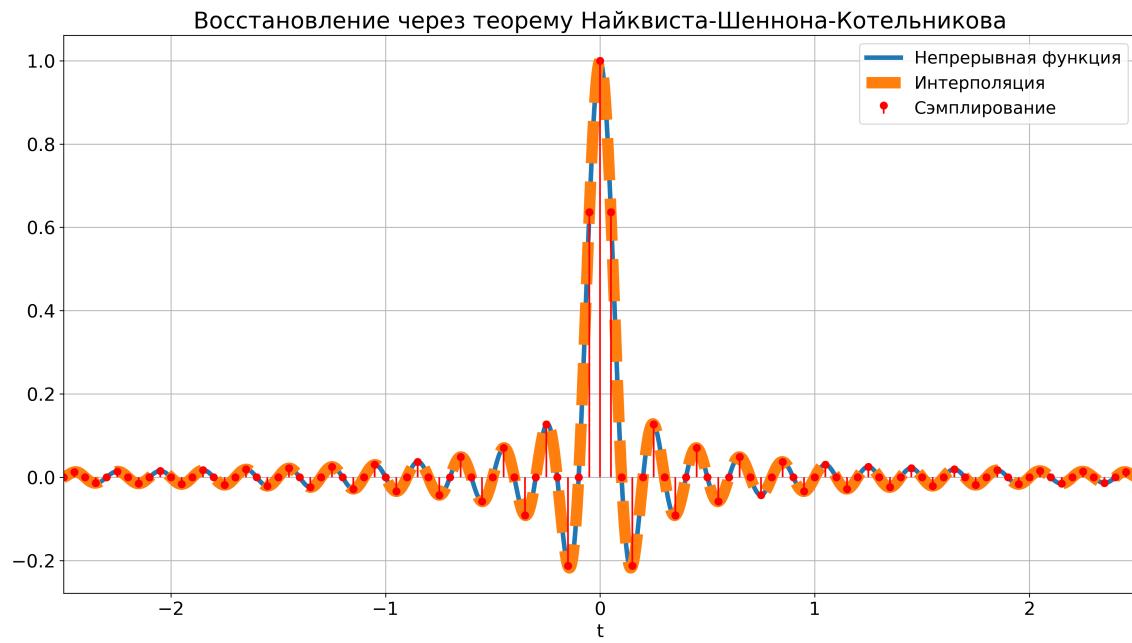


Рис. 74: Применение теоремы восстановления:  $T = 5$ ,  $\Delta t = \frac{1}{20}$

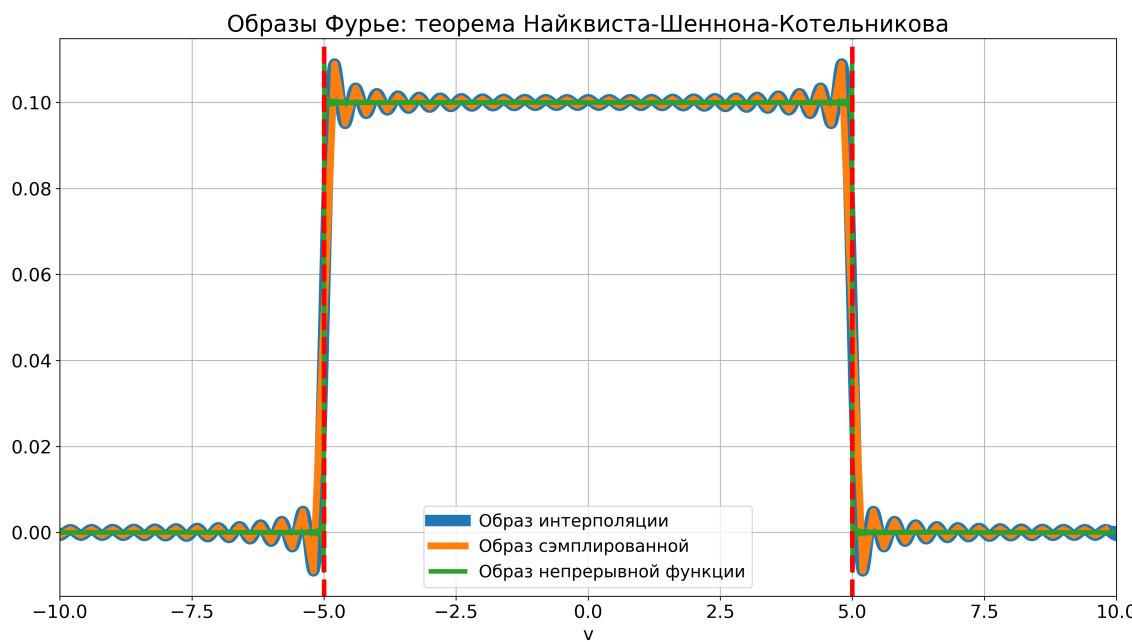


Рис. 75: Образы Фурье при применении теоремы:  $T = 5$ ,  $\Delta t = \frac{1}{20}$

### 3 Выводы

В рамках работы были изучены различные методы получения Фурье-образов функций, включая численное интегрирование, БПФ и DTFT, использовавшуюся в виде надстройки над fft. Выяснено, что точность преобразований сильно зависит от выбираемых параметров промежутка по времени  $T$ , шага дискретизации  $\Delta t$ , промежутка по частотам  $V$  и шага частот  $\Delta\nu$ , и тем больше, чем большие отрезки и мелкие шаги берутся.

Кроме того, исследовалась теорема Найквиста-Шеннона-Котельникова на конкретных функциях  $y_1(t)$  и  $y_2(t)$ . Было получено, что большие определенного значения шаги сэмплирования дают плохое восстановление исходно непрерывной функции, образ которой ограничен, а малые, напротив, уменьшают искажения и увеличивают точность интерполяционной формулы.

## 4 Приложение А

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def rect(t):
5     return np.where(np.abs(t) <= 0.5, 1, 0)
6
7 def fourier_transform(nu):
8     return np.sinc(nu) # np.sinc(x) = sin(pi x)/(pi x)
9
10 T = 5
11 dt = 0.15
12 N = int(T/dt)
13 t_int = np.arange(-T/2, T/2+dt, dt)
14
15 V = 10
16 dnu = 0.5
17 nu_int = np.arange(-V/2, V/2+dnu, dnu)
18
19 t = np.linspace(-T/2, T/2, 100000)
20 nu = np.linspace(-V/2, V/2, 100000)
21
22 f_numeric = []
23 for freq in nu_int:
24     integrand = rect(t_int) * np.exp(-2j * np.pi * freq * t_int)
25     val = np.trapz(integrand, t_int)
26     f_numeric.append(val)
27
28 rect_numeric = []
29 for time in t_int:
30     integrand = f_numeric * np.exp(2j * np.pi * time * nu_int)
31     val = np.trapz(integrand, nu_int)
32     rect_numeric.append(val)
33
34 plt.figure(figsize=(14, 8))
35 font_size = 16
36 line_width = 6
37
38 p = 1
39 if p == 1:
40     plt.plot(t, rect(t), linewidth=line_width)
41     plt.plot(t_int, rect_numeric, linewidth=line_width, color="r")
42     plt.xlabel('t', fontsize=font_size)
43     plt.grid(True)
44     plt.xticks(fontsize=font_size)
45     plt.yticks(fontsize=font_size)
46     plt.xlim([-T/2, T/2])
```

```

47     plt.ylim([-0.35, 1.45])
48     plt.legend(fontsize=font_size-1)
49     plt.tight_layout()
50     plt.savefig(f"images/time_{T}_{dt}_{V}_{dnu}.png", dpi=300)
51     plt.show()
52 else:
53     plt.plot(nu, fourier_transform(nu), linewidth=line_width)
54     plt.plot(nu_int, f_numeric, linewidth=line_width, color="r",
55               linestyle=":")
56     plt.xlabel('v', fontsize=font_size)
57     plt.grid(True)
58     plt.xticks(fontsize=font_size)
59     plt.yticks(fontsize=font_size)
60     plt.xlim([-V/2, V/2])
61     plt.ylim([-0.35, 1.35])
62     plt.legend(fontsize=font_size-1)
63     plt.tight_layout()
64     plt.savefig(f"images/freq_{T}_{dt}_{V}_{dnu}.png", dpi=300)
65     plt.show()

```

Листинг 1: Численное интегрирование

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def rect(t):
5     return np.where(np.abs(t) <= 0.5, 1, 0)
6
7 def fourier_transform(nu):
8     return np.sinc(nu) # np.sinc(x) = sin(pi x)/(pi x)
9
10 T = 5
11 dt = 0.05
12 N = int(T/dt)
13 t_int = np.linspace(-T/2, T/2, N)
14
15 V = 1/dt
16 dnu = 1/T
17 nu_int = np.linspace(-V/2-dnu, V/2, N)
18
19 t = np.linspace(-T/2, T/2, 100000)
20 nu = np.linspace(-V/2, V/2, 100000)
21
22
23 f_numeric = np.fft.fftshift(np.fft.fft(rect(t_int), norm="ortho"))
24
25 rect_numeric = np.fft.ifft(np.fft.ifftshift(f_numeric), norm="ortho")
26
27 plt.figure(figsize=(14, 8))
28 font_size = 16
29 line_width = 6
30
31 p = 1
32 if p == 1:
33     plt.plot(t, rect(t), linewidth=line_width)
34     plt.plot(t_int, rect_numeric, linewidth=line_width, color="r",
35               linestyle="--")
36     plt.xlabel('t', fontsize=font_size)
37     plt.grid(True)
38     plt.xticks(fontsize=font_size)
39     plt.yticks(fontsize=font_size)
40     plt.xlim([-T/2, T/2])
41     plt.ylim([-1.3, 1.3])
42     plt.legend(fontsize=font_size-1)
43     plt.tight_layout()
44     plt.savefig(f"images/time_fft_{T}_{dt}.png", dpi=300)
45     plt.show()
46 else:
47     plt.plot(nu_int, f_numeric, linewidth=line_width, color="r",
48               linestyle="--")

```

```
47 plt.plot(nu, fourier_transform(nu), linewidth=line_width+1)
48 plt.xlabel('v', fontsize=font_size)
49 plt.grid(True)
50 plt.xticks(fontsize=font_size)
51 plt.yticks(fontsize=font_size)
52 plt.xlim([-V/2, V/2])
53 plt.ylim([-0.35, 1.35])
54 plt.legend(fontsize=font_size-1)
55 plt.tight_layout()
56 plt.savefig(f"images/freq_fft_{T}_{dt}.png", dpi=300)
57 plt.show()
```

Листинг 2: Использование fft

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def rect(t):
5     return np.where(np.abs(t) <= 0.5, 1, 0)
6
7 def fourier_transform(nu):
8     return np.sinc(nu) # np.sinc(x) = sin(pi x)/(pi x)
9
10 T = 5
11 dt = 0.01
12 N = int(T/dt)
13 t_int = np.linspace(-T/2, T/2, N)
14
15 V = 1/dt
16 dnu = 1/T
17 nu_int = np.linspace(-V/2-dnu, V/2, N)
18
19 t = np.linspace(-T/2, T/2, 100000)
20 nu = np.linspace(-V/2, V/2, 100000)
21
22
23 m = np.arange(N)
24 c = np.sqrt(N) * dt * (-1)**m
25 f_numeric = np.fft.fftshift(c * np.fft.fft(rect(t_int), norm="ortho"))
26 rect_numeric = np.fft.ifft(np.fft.ifftshift(f_numeric) / c, norm="ortho")
27
28 f_numeric_fft = np.fft.fftshift(np.fft.fft(rect(t_int), norm="ortho"))
29 rect_numeric_fft = np.fft.ifft(np.fft.ifftshift(f_numeric_fft), norm="ortho")
30
31 f_numeric_trapz = []
32 for freq in nu_int:
33     integrand = rect(t_int) * np.exp(-2j * np.pi * freq * t_int)
34     val = np.trapz(integrand, t_int)
35     f_numeric_trapz.append(val)
36
37 rect_numeric_trapz = []
38 for time in t_int:
39     integrand = f_numeric * np.exp(2j * np.pi * time * nu_int)
40     val = np.trapz(integrand, nu_int)
41     rect_numeric_trapz.append(val)
42
43 plt.figure(figsize=(14, 8))
44 font_size = 16
45 line_width = 6
46
```

```

47 p = 2
48 if p == 1:
49     plt.plot(t, rect(t), linewidth=line_width)
50     plt.plot(t_int, rect_numeric, linewidth=line_width, color="r",
51               linestyle="--")
52     plt.xlabel('t', fontsize=font_size)
53     plt.grid(True)
54     plt.xticks(fontsize=font_size)
55     plt.yticks(fontsize=font_size)
56     plt.xlim([-T/2, T/2])
57     plt.ylim([-0.15, 1.25])
58     plt.legend(fontsize=font_size-4)
59     plt.tight_layout()
60     plt.savefig(f"images/time_newfft_{T}_{dt}.png", dpi=300)
61     plt.show()
62 else:
63     plt.plot(nu, fourier_transform(nu), linewidth=line_width)
64     plt.plot(nu_int, f_numeric, linewidth=line_width, color="r",
65               linestyle="--")
66     plt.xlabel('v', fontsize=font_size)
67     plt.grid(True)
68     plt.xticks(fontsize=font_size)
69     plt.yticks(fontsize=font_size)
70     plt.xlim([-V/2, V/2])
71     plt.ylim([-0.35, 1.35])
72     plt.legend(fontsize=font_size-1)
73     plt.tight_layout()
74     plt.savefig(f"images/freq_newfft_{T}_{dt}.png", dpi=300)
75     plt.show()

```

Листинг 3: Использование DTFT

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def rect(t):
5     return np.where(np.abs(t) <= 0.5, 1, 0)
6
7 def fourier_transform(nu):
8     return np.sinc(nu) # np.sinc(x) = sin(pi x)/(pi x)
9
10 T = 5
11 dt = 0.01
12 N = int(T/dt)
13 t_int = np.linspace(-T/2, T/2, N)
14
15 V = 1/dt
16 dnu = 1/T
17 nu_int = np.linspace(-V/2-dnu, V/2, N)
18
19 t = np.linspace(-T/2, T/2, 100000)
20 nu = np.linspace(-V/2, V/2, 100000)
21
22
23 m = np.arange(N)
24 c = np.sqrt(N) * dt * (-1)**m
25 f_numeric = np.fft.fftshift(c * np.fft.fft(rect(t_int), norm="ortho"))
26 rect_numeric = np.fft.ifft(np.fft.ifftshift(f_numeric) / c, norm="ortho")
27
28 f_numeric_fft = np.fft.fftshift(np.fft.fft(rect(t_int), norm="ortho"))
29 rect_numeric_fft = np.fft.ifft(np.fft.ifftshift(f_numeric_fft), norm="ortho")
30
31 f_numeric_trapz = []
32 for freq in nu_int:
33     integrand = rect(t_int) * np.exp(-2j * np.pi * freq * t_int)
34     val = np.trapz(integrand, t_int)
35     f_numeric_trapz.append(val)
36
37 rect_numeric_trapz = []
38 for time in t_int:
39     integrand = f_numeric * np.exp(2j * np.pi * time * nu_int)
40     val = np.trapz(integrand, nu_int)
41     rect_numeric_trapz.append(val)
42
43 plt.figure(figsize=(14, 8))
44 font_size = 16
45 line_width = 6
46
```

```

47 p = 2
48 if p == 1:
49     plt.plot(t, rect(t), linewidth=line_width)
50     plt.plot(t_int, rect_numeric, linewidth=line_width, color="r",
51               linestyle="--")
52     plt.xlabel('t', fontsize=font_size)
53     plt.grid(True)
54     plt.xticks(fontsize=font_size)
55     plt.yticks(fontsize=font_size)
56     plt.xlim([-T/2, T/2])
57     plt.ylim([-0.15, 1.25])
58     plt.legend(fontsize=font_size-4)
59     plt.tight_layout()
60     plt.savefig(f"images/time_newfft_{T}_{dt}.png", dpi=300)
61     plt.show()
62 else:
63     plt.plot(nu, fourier_transform(nu), linewidth=line_width)
64     plt.plot(nu_int, f_numeric, linewidth=line_width, color="r",
65               linestyle="--")
66     plt.xlabel('v', fontsize=font_size)
67     plt.grid(True)
68     plt.xticks(fontsize=font_size)
69     plt.yticks(fontsize=font_size)
70     plt.xlim([-V/2, V/2])
71     plt.ylim([-0.35, 1.35])
72     plt.legend(fontsize=font_size-1)
73     plt.tight_layout()
74     plt.savefig(f"images/freq_newfft_{T}_{dt}.png", dpi=300)
75     plt.show()

```

Листинг 4: Использование DTFT

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def fourier(t, y):
5     N = len(t)
6     dt = t[1] - t[0]
7     m = np.arange(N)
8     c = np.sqrt(N) * dt * (-1)**m
9     f_numeric = np.fft.fftshift(c * np.fft.fft(y, norm="ortho"))
10    return f_numeric
11
12 a1, a2 = 1, 2
13 w1, w2 = 8*np.pi, 16*np.pi
14 phi1, phi2 = np.pi/4, np.pi/3
15
16 dt = 1e-3
17 T_true = 200
18 t = np.arange(-T_true/2, T_true/2, dt)
19
20 y1 = a1 * np.sin(w1*t + phi1) + a2 * np.sin(w2*t + phi2)
21
22 dt_sample = 1 / 40
23 T = 5
24 t_sample = np.arange(-T/2, T/2, dt_sample)
25
26 y1_sample = a1 * np.sin(w1*t_sample + phi1) + a2 * np.sin(w2*t_sample +
27   phi2)
28
29 def sinc_interp(t, t_sample, y_sample, dt):
30     result = np.zeros_like(t)
31     for n in range(len(t_sample)):
32         result += y_sample[n] * np.sinc((t - t_sample[n])/dt)
33     return result
34
35 y1_interp = sinc_interp(t, t_sample, y1_sample, dt_sample)
36
37 b = 10*np.pi
38
39 y2 = np.sinc(b * t / np.pi)
40
41 y2_sample = np.sinc(b * t_sample / np.pi)
42
43 y2_interp = sinc_interp(t, t_sample, y2_sample, dt_sample)
44
45 plt.figure(figsize=(14, 8))
46 font_size = 16
47 line_width = 6

```

```

48
49 p = 3
50 if p == 1:
51     plt.plot(t, y1, linewidth=line_width)
52     plt.plot(t, y1_interp, linewidth=line_width, linestyle="--")
53     plt.stem(t_sample, y1_sample, markerfmt='ro', linefmt='r-', basefmt
54         ='')
55     plt.xlabel('t', fontsize=font_size)
56     plt.xlim([-T/2, T/2])
57     plt.grid(True)
58     plt.xticks(fontsize=font_size)
59     plt.yticks(fontsize=font_size)
60     plt.legend(fontsize=font_size-1)
61     plt.tight_layout()
62     plt.savefig(f"images/1_inter_{T}_{Fs}.png", dpi=300)
63     plt.show()
64 elif p == 2:
65     plt.plot(t, y1, linewidth=line_width)
66     plt.xlabel('t', fontsize=font_size)
67     plt.xlim([-T/2, T/2])
68     plt.grid(True)
69     plt.xticks(fontsize=font_size)
70     plt.yticks(fontsize=font_size)
71     plt.tight_layout()
72     plt.savefig(f"images/y1.png", dpi=300)
73     plt.show()
74 elif p == 3:
75     V_true = 1/dt
76     dnu_true = 1/T_true
77     nu_true = np.arange(-V_true/2, V_true/2, dnu_true)
78     f1 = fourier(t, y1)

79     V_sample = 1/dt_sample
80     dnu_sample = 1/T
81     nu_sample = np.arange(-V_sample/2, V_sample/2, dnu_sample)
82     f_sample = fourier(t_sample, y1_sample)

83     f_inter = fourier(t, y1_interp)
84     plt.plot(nu_true, f1/max(f1), linewidth=line_width)
85     plt.plot(nu_true, f_inter, linewidth=line_width+4)
86     plt.plot(nu_sample, f_sample, linewidth=line_width)
87     plt.axvline(x=-8, color='red', linestyle='--', linewidth=2)
88     plt.axvline(x=8, color='red', linestyle='--', linewidth=2)
89     plt.xlabel('v', fontsize=font_size)
90     #plt.xlim([-V_sample/2+dnu_sample, V_sample/2-dnu_sample])
91     plt.xlim([-10, 10])
92     plt.grid(True)
93     plt.xticks(fontsize=font_size)

```

```
95 plt.yticks(fontsize=font_size)
96 plt.legend(fontsize=font_size-1)
97 plt.tight_layout()
98 plt.savefig(f"images/fourier_{T}_{Fs}.png", dpi=300)
99 plt.show()
```

Листинг 5: Сэмплирование