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

Студент: Овчинников П.А.
Поток: ЧАСТ.МЕТ. 1.3

Преподаватель: Перегудин А.А.
Пашенко А.В.

Санкт-Петербург
2024

Содержание

Непрерывное и дискретное преобразование Фурье

Рассмотрим уже знакомую нам прямоугольную функцию единичного масштаба:

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

Сейчас мы опробуем на ней различные варианты Фурье-преобразования и выясним, чем они отличаются между собой, какой быстрее, какой точнее и как совместить достоинства разных подходов вместе.

Истинный Фурье-образ

Известно, что истинное Фурье-преобразование прямоугольной функции равно:

$$\hat{\Pi}(v) = \int_{-\infty}^{\infty} \Pi(t)\, \mathrm{e}^{-2\pi i v t}\, dt = \int_{-1/2}^{1/2} \mathrm{e}^{-2\pi i v t}\, dt = \frac{\mathrm{e}^{-\pi i v} - \mathrm{e}^{\pi i v}}{2\pi i v} = \frac{\sin \pi v}{\pi v} = \mathrm{sinc}(v)$$

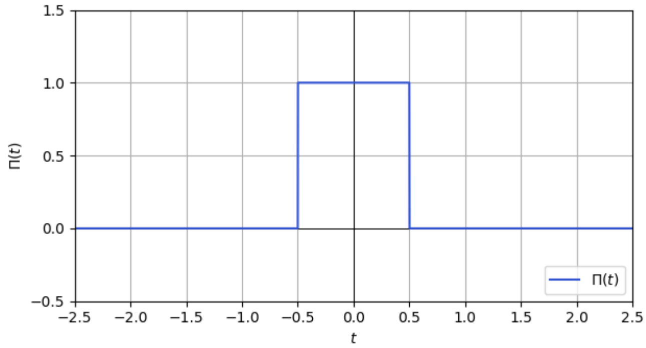

Рис. 1: График функции $\Pi(t)$

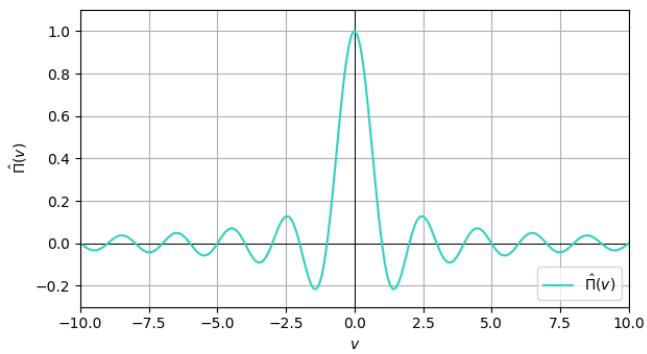

Рис. 2: График истинного Фурье-образа $\hat{\Pi}(v)$

Численное интегрирование

Для численного интегрирования воспользуемся функцией `trapz` из библиотеки `numpy`. Метод подразумевает, что мы интегрируем по ограниченному промежутку и с заданным шагом интегрирования. Начнём с промежутка $T = 20$ и разбиения на 1000 точек. Вычислим Фурье-образ $\hat{\Pi}_N(v)$ методом численного интегрирования:

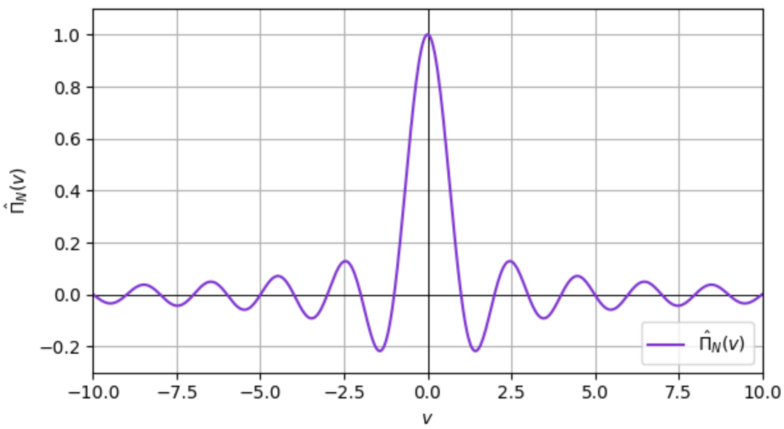

Рис. 3: График численного Фурье-образа $\hat{\Pi}(v)$ при $T = 20$ и $N = 1000$

Как видим, график численного Фурье-образа $\hat{\Pi}_N(v)$ внешне совпадает с истинным Фурье-образом $\hat{\Pi}(v)$, но только внешне.

Выполним обратное преобразование Фурье и посмотрим на результат:

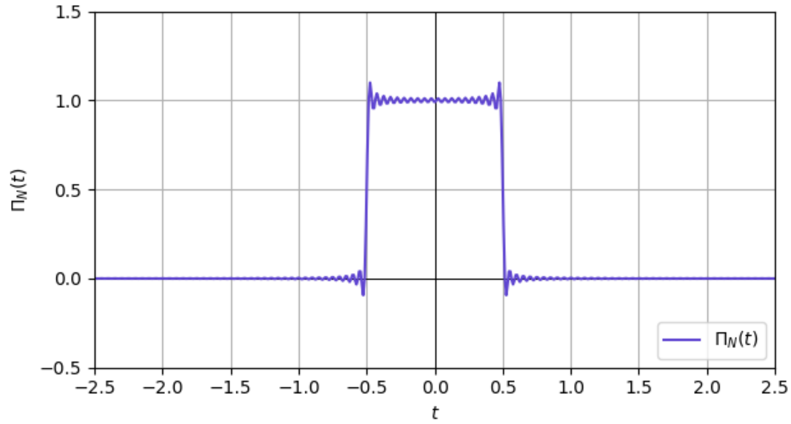

Рис. 4: График восстановленной из численного Фурье-образа функции $\Pi_N(t)$ при $T = 20$ и $N = 1000$

Возникает эффект Гиббса, который проявляется в виде колебаний вблизи точек разрыва. Это связано с тем, что мы интегрировали по конечному промежутку, а не по всей числовой прямой.Попробуем увеличить промежуток интегрирования до $T = 90$:

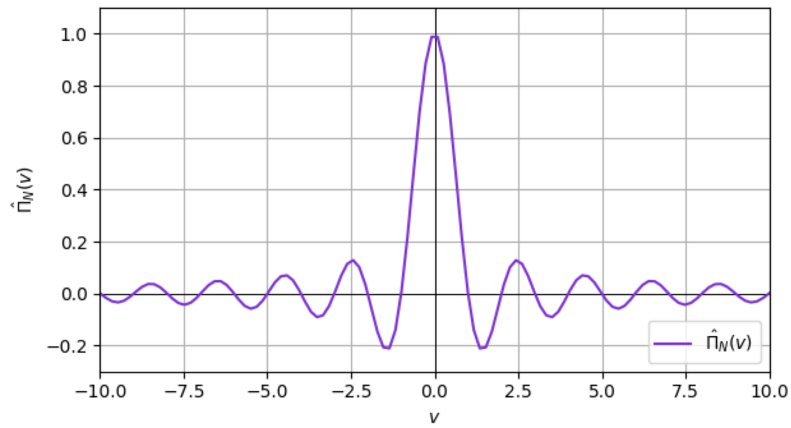

Рис. 5: График численного Фурье-образа $\hat{\Pi}(v)$ при $T = 90$ и $N = 1000$

Промежуток вырос, а то же количество точек равномерно распределилось по большему промежутку — это понижает точность в вычислении интеграла. На изгибах кардинального синуса наблюдаются неровности.

Теперь посмотрим на восстановленную функцию:

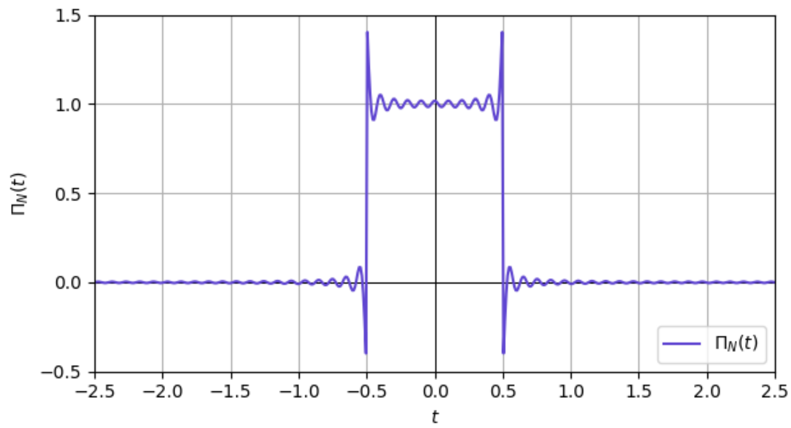

Рис. 6: График восстановленной из численного Фурье-образа функции $\Pi_N(t)$ при $T = 90$ и $N = 1000$

Ввиду того, что мы увеличили промежуток интегрирования, но не увеличили количество точек, на графике восстановленной функции колебания усилились, ведь гораздо больше высокочастотных гармоник было упущено в образе Фурье.

Теперь увеличим количество точек до $N = 10000$:

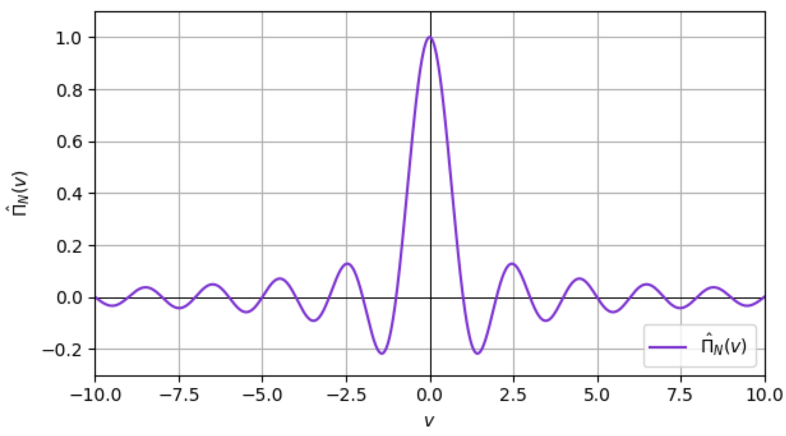

Рис. 7: График численного Фурье-образа $\hat{\Pi}(v)$ при $T = 90$ и $N = 10000$

Кривая образа снова стала гладкой — вновь восстановим из образа прямоугольную функцию путём обратного преобразования Фурье:

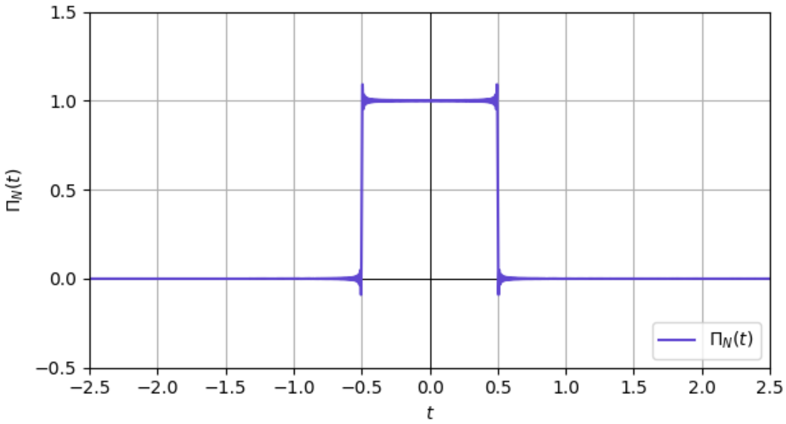

Рис. 8: График восстановленной из численного Фурье-образа функции $\Pi_N(t)$ при $T = 90$ и $N = 10000$

Амплитуда колебаний на границах разрыва уменьшилась, но они всё равно присутствуют, как ни крути. Что может сказать код на Python о быстродействии такого метода?

```
trapz #1: 3.07 s
trapz #2: 1.16 s
trapz #3: 19.98 s
```

Вывод:

> Метод `trapz` работает медленно. При 1000 точек он работает около 2 секунд, а при 10000 точек — уже 20 секунд. При этом метод достаточно точный, но плохо работает с функциями, имеющими разрывы первого рода.

Использование DFT

Попробуем воспользоваться дискретным преобразованием Фурье. Для этого нам понадобится функция `fft` из библиотеки `numpy`. Эта функция реализует быстрое преобразование Фурье, которое математически описывается так:

$$F(k) = \sum_{m=0}^{N-1} f(m) \exp\left\{-2\pi i \frac{mk}{N}\right\} \quad \leftrightarrow \quad f(m) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) \exp\left\{2\pi i \frac{mk}{N}\right\}$$

$$F(k) = \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f(m) \exp\left\{-2\pi i \frac{mk}{N}\right\} \quad \leftrightarrow \quad f(m) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) \exp\left\{2\pi i \frac{mk}{N}\right\}$$

Чтобы преобразование стало унитарным, нам потребовалось разбить множитель $1/N$ в обратном преобразовании на два $1/\sqrt{N}$ — это сохранит норму сигнала. В `numpy.fft` достаточно указать параметр `norm='ortho'`, чтобы преобразование стало унитарным.

Возьмём разбиение на 10000 точек и проведём преобразование Фурье нашей прямоугольной функции и посмотрим на график образа при таком преобразовании:

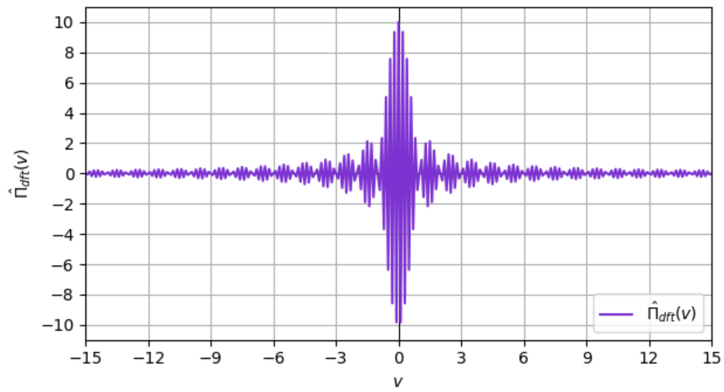

Рис. 9: График дискретного Фурье-образа $\hat{\Pi}(v)$

Это мало похоже на истинный Фурье-образ. Попробуем восстановить из него прямоугольную функцию:

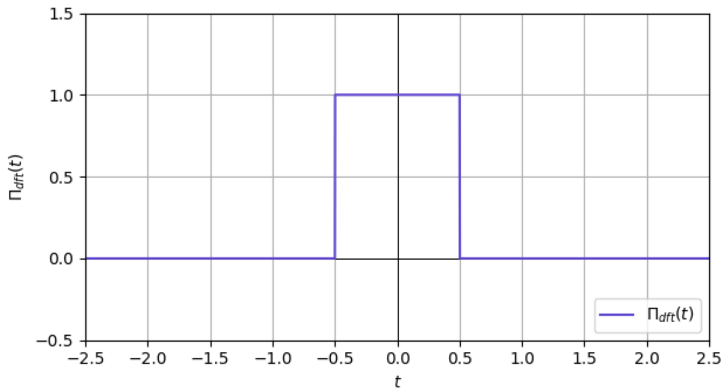

Рис. 10: График восстановленной из дискретного Фурье-образа функции $\Pi_N(t)$

Что удивительно, восстановленная функция выглядит ровно так же, как и исходная. При этом код на Python показывает следующие результаты:

```
dft: 0.64 s
```

Вывод:

> Метод `DFT` работает быстро и точно. При 10000 точек он работает около 1 секунды. Это более чем в 30 раз быстрее, учитывая равное количество точек с третьим испытанием метода `trapz`. При этом метод хорошо работает с функциями, имеющими разрывы первого рода. Но с Фурье-образом что-то пошло не так.

Выводы о работе trapz и DFT

Приблизиться к истинному Фурье-образу смог только метод `trapz`, но он работает медленно и неустойчив к разрывам. Метод `numpy.fft` работает быстро и точно, но его образ сильно отличается от истинного. С чем это связано?

Как следует из названия, метод `trapz` аппроксимирует интегрирование, разбивая площадь на маленькие трапеции шириной в dt, площадь которых вычислять проще. Для интегрирования с $N+1$ равномерно расположенными точками аппроксимация работает так:

$$\int_a^b f(x)\,dx \approx \frac{1}{2}\sum_{n=1}^{N}(x_{n+1}-x_n)\left(f(x_n)+f(x_{n+1})\right) = \frac{b-a}{2N}\sum_{n=1}^{N}\left(f(x_n)+f(x_{n+1})\right), \qquad \frac{b-a}{N} \sim (x_{n+1}-x_n) \sim dt$$

Формула с коэффициентом $^1/_2$ — для случаев, когда расстояние между точками не фиксированное. Если оно фиксированное, как в случае с преобразованием Фурье, то используется вторая формула.

Метод `numpy.fft` работает по-другому. Он преобразует сигнал в пространство частот, где каждая гармоника имеет свой вес. При этом важно, чтобы сигнал был периодичен, иначе возникнут артефакты. В нашем случае прямоугольная функция не является периодичной, поэтому и образ Фурье отличается от истинного. Формулы унитарного дискретного преобразования были приведены в предыдущем пункте.

Приближение непрерывного с помощью DFT

Попробуем переиграть метод `numpy.fft` и всё же приблизить истинный Фурье-образ $\hat{\Pi}(v)$ с помощью дискретного преобразования Фурье. Рассмотрим интеграл Фурье-преобразования:

$$F(v) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(t)e^{-2\pi ivt}\,dt$$

Аппроксимируем его с помощью суммы Римана — для этого введём замену $t = m\Delta t + t_0 \Rightarrow dt = \Delta t$:

$$F(v) \approx \frac{1}{\sqrt{N}}\sum_{m=0}^{N-1} f(m\Delta t+t_0)e^{-2\pi iv(m\Delta t+t_0)}\Delta t = \frac{1}{\sqrt{N}}\Delta t e^{-2\pi ivt_0}\sum_{m=0}^{N-1} f(m\Delta t+t_0)\,\mathrm{e}^{-2\pi ivm\Delta t}$$

Для частот введём замену $v_k = k\Delta v = {}^k/_{N\Delta t}$, а т.к. t_0 и Δt постоянные, то введём замену $f[n] = f(n\Delta t + t_0)$:

$$F(v_k) \approx \frac{1}{\sqrt{N}}\Delta t e^{-2\pi ivt_0}\sum_{m=0}^{N-1} f[n]\,\mathrm{e}^{-2\pi iv_k m\Delta t} = \frac{1}{\sqrt{N}}\Delta t e^{-2\pi ivt_0}\sum_{m=0}^{N-1} f[n]\,\mathrm{e}^{-2\pi i\frac{km}{N}}$$

Как ни странно, мы получаем формулу для дискретного преобразования Фурье, но с домножением на фазовый коэффициент $\Delta t\,\mathrm{e}^{-2\pi ivt_0}$. Назовём такое преобразование `cdft`. Проведём его на прямоугольной функции:

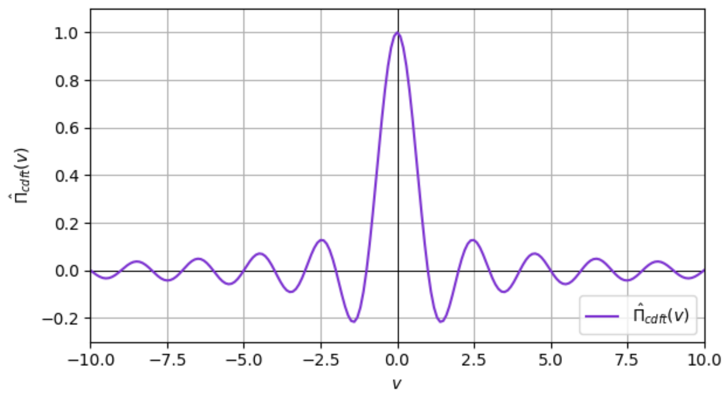

Рис. 11: График приближённого Фурье-образа $\hat{\Pi}_{cdft}(v)$

Действительно получаем гладкую кривую, похожую на истинный Фурье-образ.

Попробуем восстановить из него прямоугольную функцию:

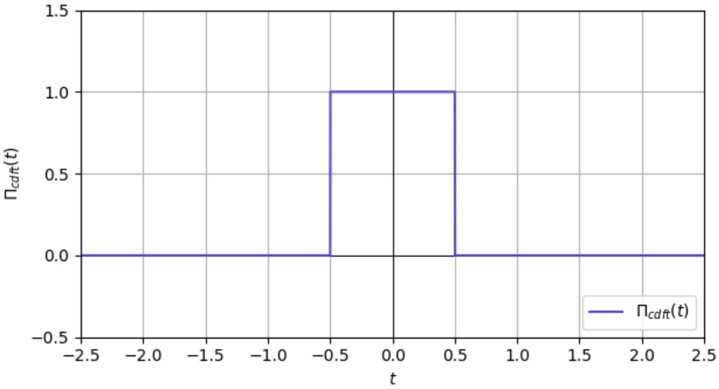

Рис. 12: График восстановленной из приближённого Фурье-образа функции $\Pi_{cdft}(t)$

Восстановленная функция выглядит так же, как и исходная. При этом код на Python показывает следующие результаты:

```
cdft: 0.63 s
```

Вывод:

> Проведя некоторые преобразования, мы смогли лишить метод `DFT` недостатков и приблизить его к истинному Фурье-образу. При этом метод остался быстрым и точным и всё так же отлично работает с функциями, имеющими разрывы первого рода.

Сэмплирование

Исследуем теорему Найквиста-Шеннона-Котельникова, которая гласит, что для безошибочного восстановления сигнала из образа, взятого по дискретным данным, на отрезке $[-B, B]$ необходимо, чтобы шаг дискретизации Δt подчинялся условию $\Delta t < 1/(2B)$. Мы рассмотрим два разных сигнала: сумму двух синусоид и кардинальный синус.

Для восстановления сигнала из его дискретного образа будем пользоваться интерполяционной формулой:

$$f(t) = \sum_{n=-\infty}^{\infty} f[n] \operatorname{sinc}\left(2B\left(t - t_n\right)\right), \quad t_n = \frac{n}{2B}$$

Сэмплирование синусов

Рассмотрим первую функцию — сумма синусоид, зададимся параметрами $a_1 = 1$, $\omega_1 = 2$, $\varphi_1 = 3$, $a_2 = 4$, $\omega_2 = 5$, $\varphi_2 = 6$. Так выглядит график функции:

$$y(t) = a_1 \sin(\omega_1 t + \varphi_1) + a_2 \sin(\omega_2 t + \varphi_2) = \sin(2t + 3) + 4\sin(5t + 6)$$

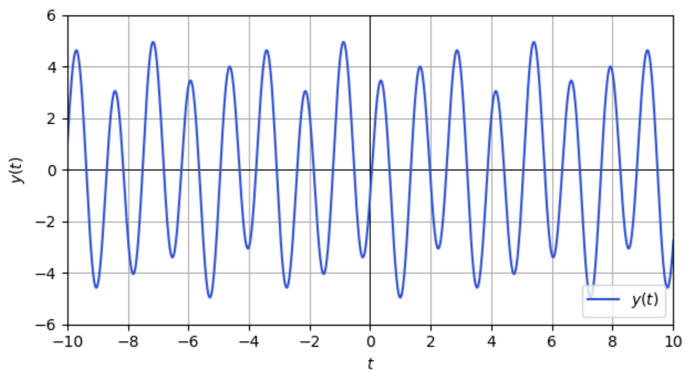

Рис. 13: График $y(t)$

Теперь проведём сэмплирование сигнала с шагом $\Delta t = 1/8$:

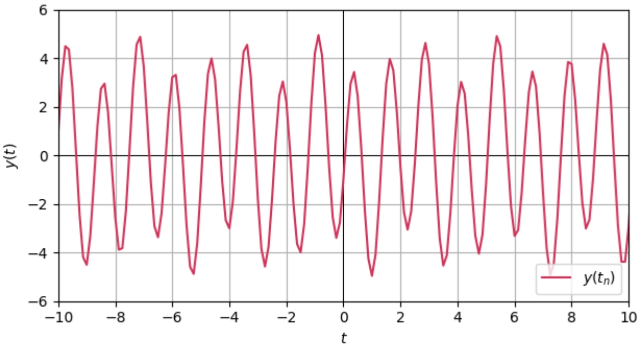

Рис. 14: График сэмплированного сигнала $y(t_n)$

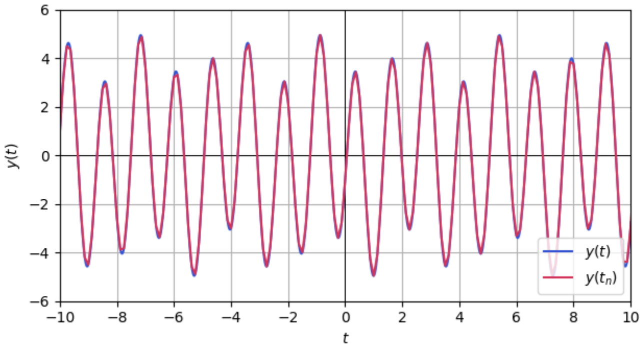

Рис. 15: Сравнительный график $y(t)$ и $y(t_n)$

На сравнительном графике не сильно заметна разница между сигналами, поэтому слева дополнительно приведён график сэмплированного сигнала отдельно. Теперь попробуем интерполяцией восстановить сигнал по сэмплированному образу с $B = 4$:

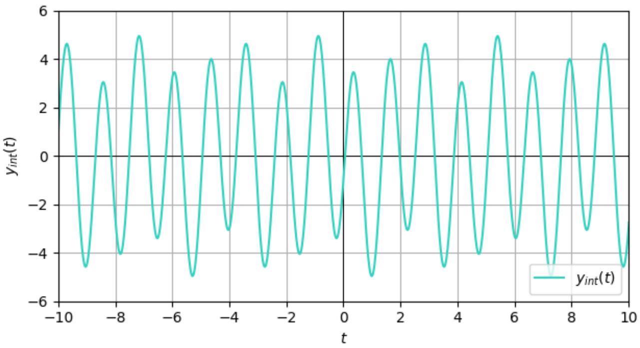

Рис. 16: График восстановленного сигнала $y_{int}(t)$

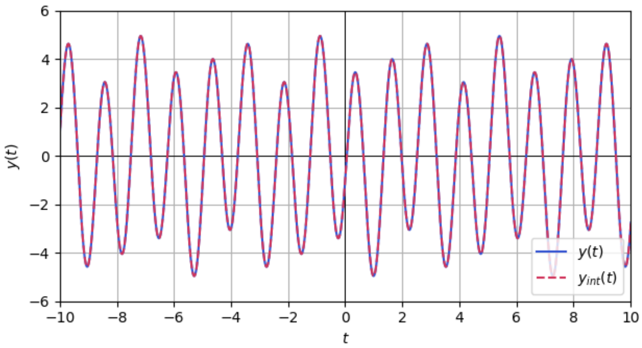

Рис. 17: Сравнительный график $y(t)$ и $y_{int}(t)$

Восстановленный сигнал идентичен исходному. Это связано с тем, что $\Delta t = 1/8 = 1/(2B)$.

Теперь увеличим шаг сэмплирования до $\Delta t = 1/4$:

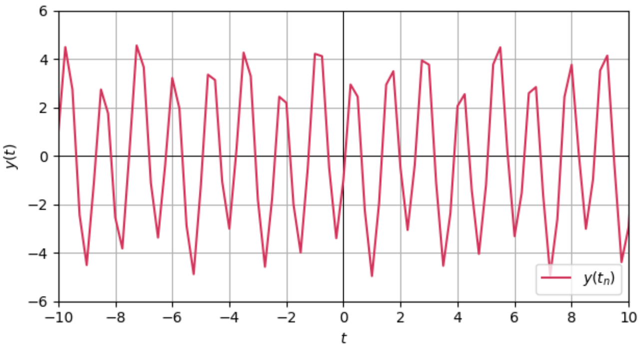

Рис. 18: График сэмплированного сигнала $y(t_n)$

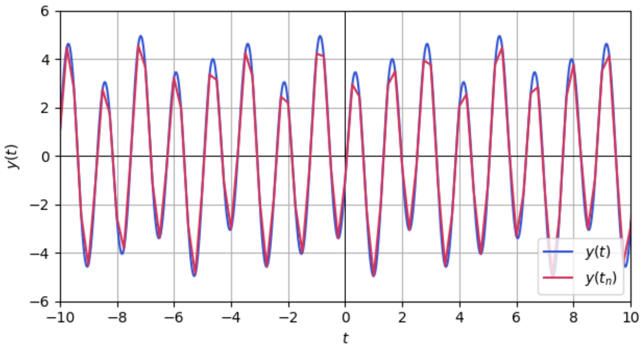

Рис. 19: Сравнительный график $y(t)$ и $y(t_n)$

Теперь разница между сигналами заметна на сравнительном графике.

Попробуем восстановить сигнал по сэмплированному образу с тем же значением B:

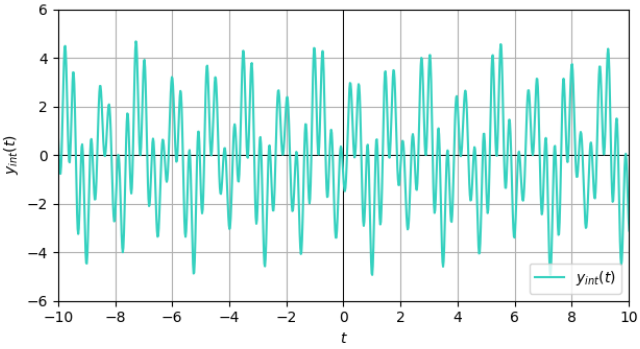

Рис. 20: График восстановленного сигнала $y_{int}(t)$

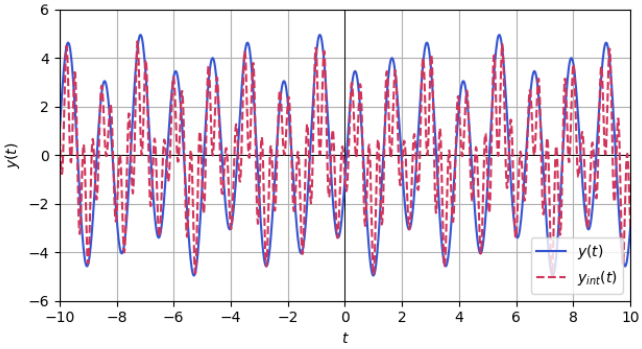

Рис. 21: Сравнительный график $y(t)$ и $y_{int}(t)$

Как мы видим, увеличение шага сэмплирования привело к тому, что формула интерполяции не сработала так, как должно — восстановленный сигнал отличается от исходного.

Чтобы это исправить, уменьшим B до 2 и попробуем ещё раз восстановить сигнал:

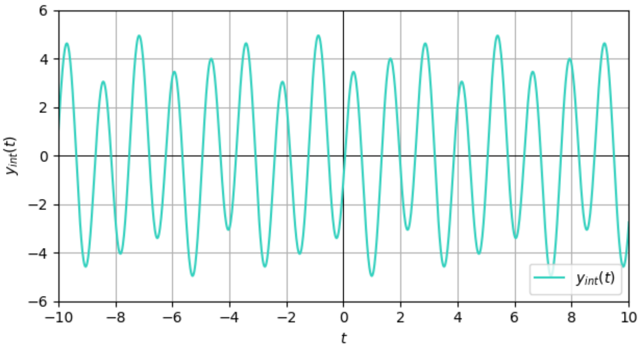

Рис. 22: График восстановленного сигнала $y_{int}(t)$

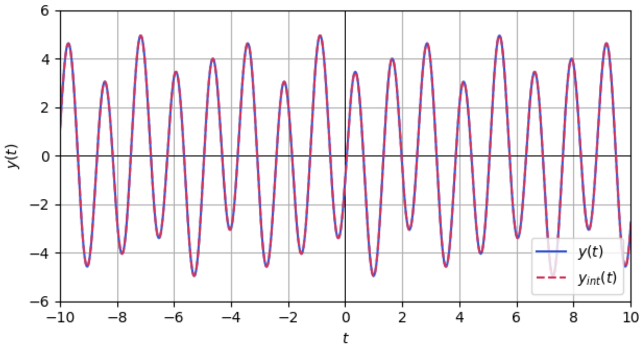

Рис. 23: Сравнительный график $y(t)$ и $y_{int}(t)$

Теперь восстановить сигнал из сэмплированного удалось.

Сэмплирование sinus cardinalis

Перейдём к сэмплированию и интерполяции кардинального синуса. Сначала зададимся параметром $b = 2$ и функцией:

$$y(t) = \text{sinc}(bt) = \text{sinc}(2t)$$

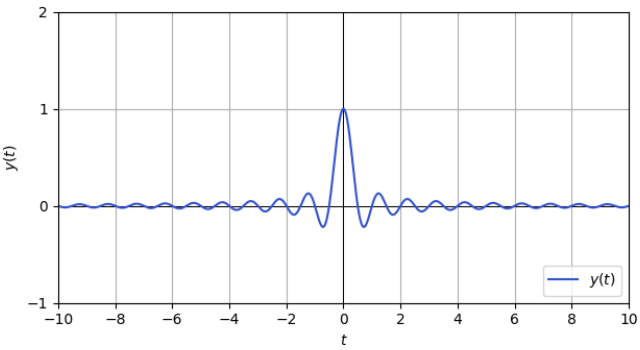

Рис. 24: График $y(t)$

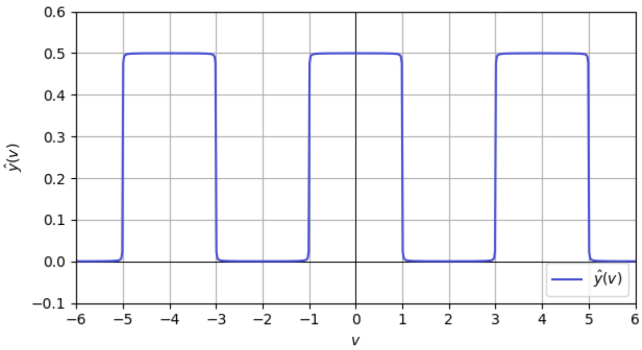

Рис. 25: Фурье-образ $y(t)$

Фурье-образ кардинального синуса, как и ожидалось, представляет собой прямоугольную функцию.

Оставим параметр $B = 2$, а шаг дискретизации установим $\Delta t = 1/2$. И вот так будет выглядеть сравнительный график исходного и сэмплированного сигналов:

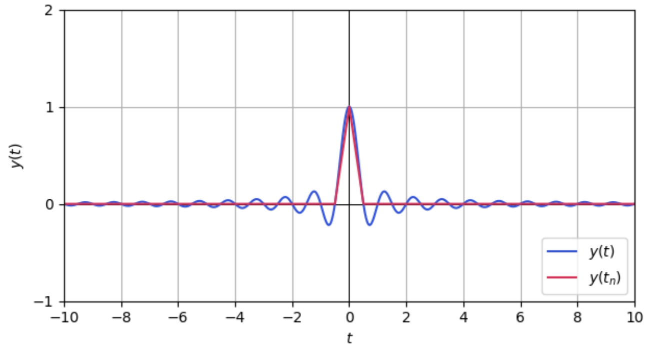

Рис. 26: Сравнительный график $y(t)$ и $y(t_n)$

Как мы видим, сэмплированный сигнал сведён к треугольной волне. Теперь попробуем восстановить сигнал:

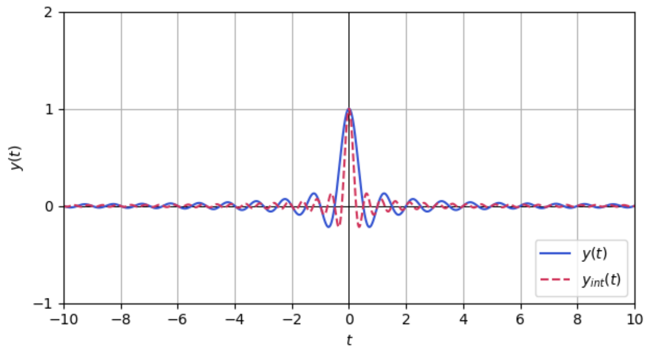

Рис. 27: Сравнительный график $y(t)$ и $y_{int}(t)$

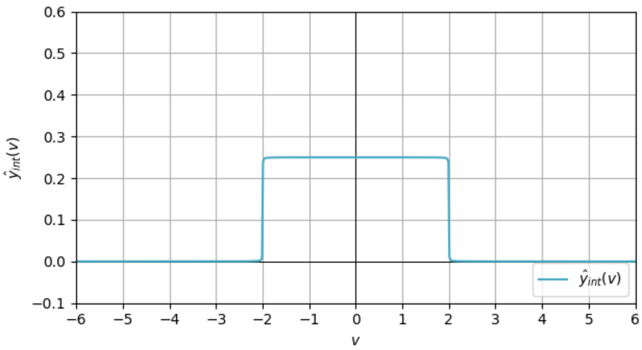

Рис. 28: Фурье-образ сигнала $y_{int}(t)$

Как видим, интерполяция не сработала, и восстановленный сигнал отличается от исходного. Также и Фурье-образ восстановленного сигнала представляет собой прямоугольную функцию другой ширины и амплитуды. Чтобы это исправить, изменим шаг сэмплирования до $\Delta t = 1/4$:

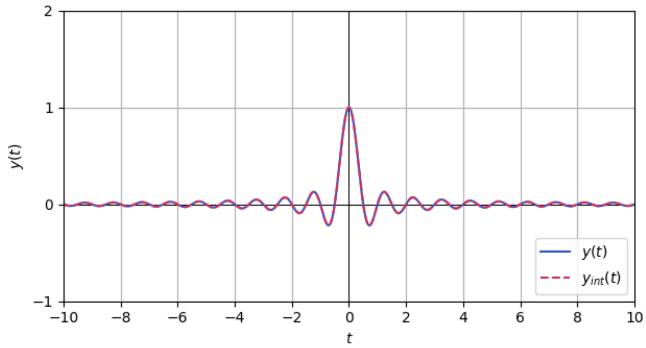

Рис. 29: Сравнительный график $y(t)$ и $y_{int}(t)$

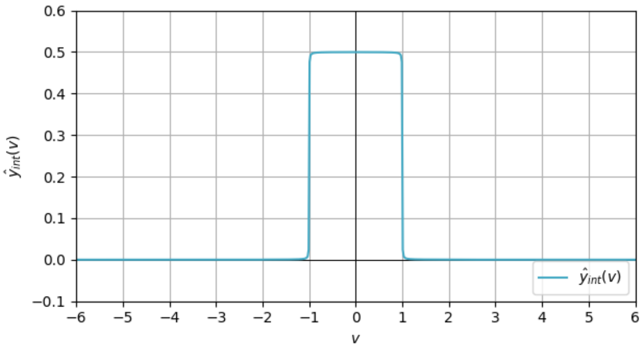

Рис. 30: Фурье-образ сигнала $y_{int}(t)$

Восстановленный сигнал и его Фурье-образ идентичны исходному. Заметим, что прямоугольная волна только одна в образе и образ больше не периодичный.

Вывод:

> Мы исследовали теорему Найквиста-Шеннона-Котельникова на примере двух разных сигналов. Дейсвительно, восстановить исходный сигнал получалось тогда, когда шаг дискретизации Δt был меньше $1/(2B)$. При этом образ Фурье восстановленного сигнала не периодичный, в отличие от образа Фурье исходного сигнала.