

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

**Сигналы с ограниченным спектром**

Отчет по лабораторной работе 5

по дисциплине «**Обработка сигналов**»

студента 3 курса группы ИВТ-б-о-222

Гоголева Виктора Григорьевича

Направления подготовки 09.03.01 «Информатика и вычислительная техника»

Симферополь, 2025

## Лабораторная работа №5

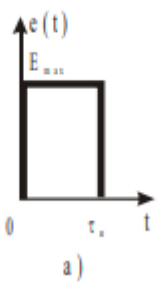
**Тема:** Спектральный анализ периодических сигналов

**Цель работы:** Задан сигнал. Определить эффективную ширину спектра данного сигнала. Рассчитать отсчетные значения этого сигнала, необходимые для его однозначного восстановления. Восстановить сигнал по его отсчетным значениям. Построить следующие графики:

1. исходный сигнал;
2. отсчетные значения исходного сигнала по оси времени;
3. восстановленный сигнал.

Параметры сигнала приведены в таблице

### Вариант № 17

| Сигнал  | Е, В | t, мкс |
|---|------|--------|
|  | 34   | 700    |

### Теоретические сведения

Любой сигнал имеет бесконечный спектр. Однако, как правило, существует эффективная ширина спектра ( $\omega_B$ ), в пределах которой передается основная мощность сигнала. Определяется эффективная ширина спектра с

использованием теоремы Парсеваля (22). Эффективная ширина выбирается исходя из потери не более 10% энергии сигнала.

Сигнал с ограниченным спектром может быть представлен в виде набора дискретных отсчетных значений сигнала, взятых через равные промежутки времени  $\Delta\tau = 1/(2 \cdot f_b) = \pi/\omega_b$ .

Произвольный сигнал  $s(t)$  с ограниченным спектром может быть разложен в обобщенный ряд Фурье по базису Котельникова:

$$s(t) = \sum_{k=-\infty}^{\infty} c_k \cdot Sc_k(t; \omega_b)$$

где  $c_k$  – коэффициенты обобщенного ряда Фурье;  $Sc_k(t; \omega_b)$  –  $k$ -ая отсчетная функция, совокупность которых образует базис Котельникова.  $Sc_k(t; \omega_b)$  вычисляется по следующей формуле:

$$Sc_k(t; \omega_b) = \sqrt{\frac{\omega_b}{\pi}} \cdot \frac{\sin(\omega_b \cdot (t - k \cdot \pi/\omega_b))}{\omega_b \cdot (t - k \cdot \pi/\omega_b)}$$

Коэффициенты обобщенного ряда Фурье вычисляются по следующей формуле:

$$c_k = (s(t), Sc_k(t; \omega_b)) = \sqrt{\frac{\omega_b}{\pi}} \cdot \int_{-\infty}^{\infty} s(t) \cdot \frac{\sin(\omega_b \cdot (t - k \cdot \pi/\omega_b))}{\omega_b \cdot (t - k \cdot \pi/\omega_b)} \cdot dt$$

Зная спектральную плотность  $S(\omega)$  заданного сигнала  $s(t)$  и используя обобщенную формулу Рэлея, можно найти коэффициенты разложения через интеграл по частотному спектру:

$$c_k = \sqrt{\frac{\omega_b}{\pi}} \cdot \left\{ \frac{1}{2 \cdot \pi} \cdot \int_{-\omega_b}^{\omega_b} S(\omega) \cdot \exp[j \cdot k \cdot \pi \cdot \omega/\omega_b] \cdot d\omega \right\}$$

Мгновенное значение сигнала  $s(t)$  в  $k$ -ой отсчетной точке  $t_k = k \cdot \pi/\omega_b = k/(2 \cdot f_b)$ :

$$s_k = \frac{1}{2 \cdot \pi} \cdot \int_{-\omega_B}^{\omega_B} S(\omega) \cdot \exp[j \cdot k \cdot \pi \cdot \omega / \omega_B] \cdot d\omega$$

Тогда ряд Котельникова имеет вид:

$$s(t) = \sum s_k \cdot \frac{\sin(\omega_B \cdot (t - k \cdot \pi / \omega_B))}{\omega_B \cdot (t - k \cdot \pi / \omega_B)},$$

$$c_k = \sqrt{\frac{\pi}{\omega_B}} \cdot s_k$$

### **Ход работы**

1. Пройти инструктаж по технике безопасности работы в компьютерном классе, изучить инструкции по технике безопасности и правилам оказания первой медицинской помощи.
2. Задан сигнал (таблица 5.1).
3. Разработать программное обеспечение для исследования сигналов с ограниченным спектром.
4. Определить эффективную ширину спектра данного сигнала.
5. Рассчитать отсчетные значения этого сигнала, необходимые для его однозначного восстановления.
6. Восстановить сигнал по его отсчетным значениям.
7. Построить график исходного сигнала, диаграмму полученных отсчетных значений, график восстановленного сигнала.
8. Сделать выводы по работе.

Согласно варианту задания № 17 сигнал имеет прямоугольный импульс, в программном коде была реализована функция для реализации сигнала такого вида

```
# Функция для прямоугольного импульса
def rect_pulse(t, amplitude, duration):
    return np.where((t >= 0) & (t <= duration), amplitude, 0)
```

Рисунок 1 – функция прямоугольного импульса

```
# Заданные параметры
E_max = 34 # В
t_duration = 700e-6 # 700 мкс
omega = 2 * np.pi / t_duration
f_v = omega / (2 * np.pi)
```

Рисунок 2 – установка константных параметров из условий задачи

Первым заданием в работе было рассчитать эффективную ширину спектра данного сигнала.

```

def calculate_energy(E_max, tau):
    """Вычисление энергии сигнала."""
    return E_max**2 * tau

def find_effective_bandwidth(target):
    """Поиск эффективной ширины спектра."""

    def integral_sinc_squared(x):
        result, _ = quad(
            lambda t: (np.sin(t) / t) ** 2 if t != 0 else 1.0, 0, x, limit=1000
        )
        return result

    def equation(x):
        return integral_sinc_squared(x) - target

    sol = root_scalar(equation, bracket=[0, 1000], method="brentq")
    return sol.root

```

Рисунок 3 – функции вычисления энергии сигнала и эффективной ширины спектра для сигнала( для энергии 99%)

```

def calculate_samples(tau, delta_tau):
    """Вычисление моментов отсчётов."""
    return np.arange(0, tau + delta_tau, delta_tau)

def restored_signal(t, valid_t_k, s_k_values, omega_eff):
    """Восстановление сигнала по отсчётам."""
    return sum(
        s_k * np.sinc(omega_eff / np.pi * (t - t_k))
        for t_k, s_k in zip(valid_t_k, s_k_values)
    )

```

Рисунок 4 – функции вычисления моментов отсчета

```

# Основные параметры
E_max = 34 # Амплитуда (В)
tau = 700e-6 # Длительность импульса (с)
target = 0.99 * np.pi / 2

# Расчёты
E = calculate_energy(E_max, tau)
x_eff = find_effective_bandwidth(target)
omega_eff = 2 * x_eff / tau
delta_tau = np.pi / omega_eff

# Генерация отсчётов
valid_t_k = calculate_samples(tau, delta_tau)
s_k_values = E_max * np.ones_like(valid_t_k)

```

Рисунок 5 – объявление параметров из варианта и вызов функций для вычислений

## Результаты

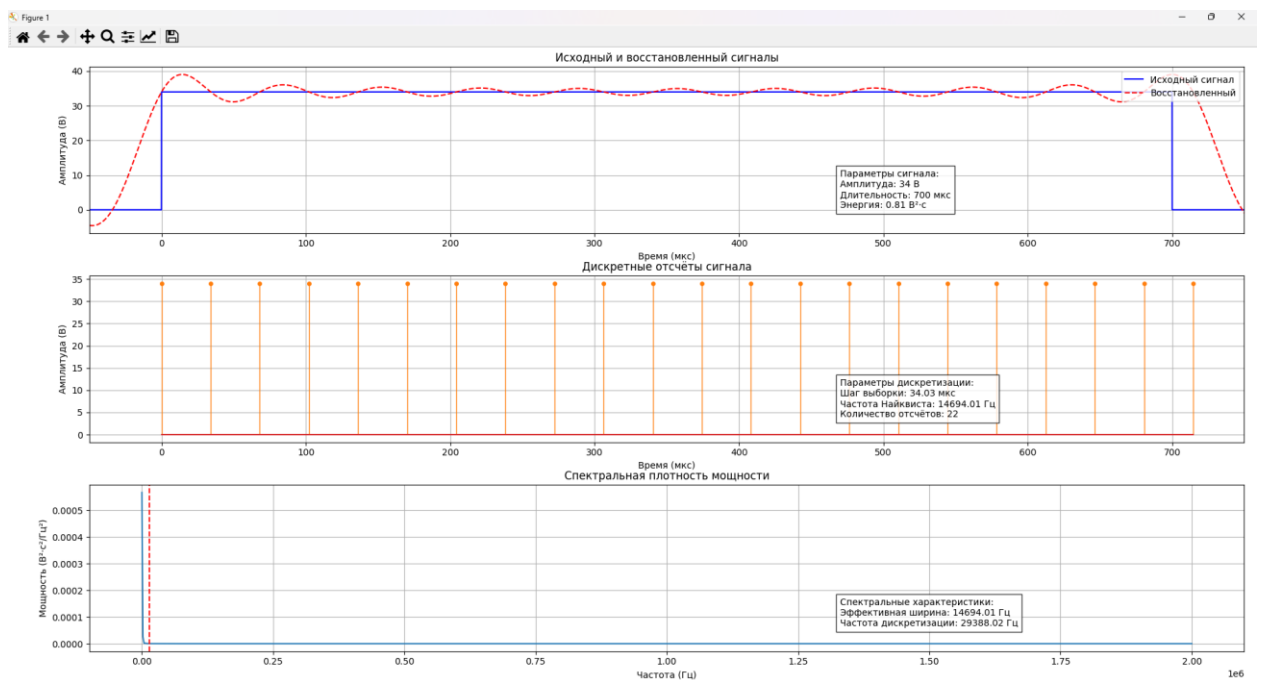


Рисунок 6 – общие результаты работы



Рисунок 7 – сравнение исходного и восстановленного сигнала

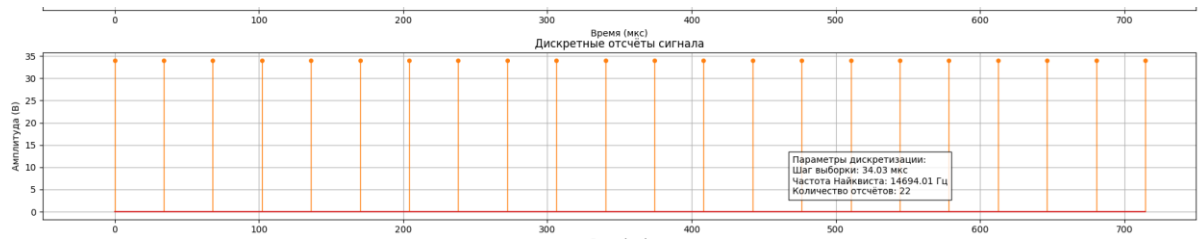


Рисунок 8 – распределение точек отсчета на временной плоскости сигнала



Рисунок 9 – график спектральной плоскости мощности сигнала

## Анализ результатов

Ширина спектра сигнала 14694 Гц. При такой частоте будет 22 отсчетов. В результате выполнения программы можно сказать, что восстановленная функция верна, т.к. она проходит через все дискретные точки.



**Вывод:** во время выполнения данной работы был изучен принцип нахождения ширины спектра сигнала, расчета дискретных значений сигнала, восстановления по отсчетным значениям с помощью ряда Котельникова. Также была найдена ширина спектра, отсчетные значения сигнала и построен восстановленный сигнал.

Была создана программа, которая вычисляет необходимые величины и строит графики отсчетных значений, изначального и восстановленного сигнала.

## ПРИЛОЖЕНИЕ

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad
from scipy.optimize import root_scalar

def calculate_energy(E_max, tau):
    """Вычисление энергии сигнала."""
    return E_max**2 * tau

def find_effective_bandwidth(target):
    """Поиск эффективной ширины спектра."""

    def integral_sinc_squared(x):
        result, _ = quad(
            lambda t: (np.sin(t) / t) ** 2 if t != 0 else 1.0, 0, x, limit=1000
        )
        return result

    def equation(x):
        return integral_sinc_squared(x) - target

    sol = root_scalar(equation, bracket=[0, 1000], method="brentq")
    return sol.root

def calculate_samples(tau, delta_tau):
    """Вычисление моментов отсчётов."""
    return np.arange(0, tau + delta_tau, delta_tau)

def restored_signal(t, valid_t_k, s_k_values, omega_eff):
    """Восстановление сигнала по отсчётам."""
    return sum(
        s_k * np.sinc(omega_eff / np.pi * (t - t_k))
        for t_k, s_k in zip(valid_t_k, s_k_values)
    )

def plot_results(
    t_continuous,
    original,
    restored,
    valid_t_k,
    s_k_values,
    tau,
    delta_tau,
    omega_eff,
```

```

E,
):
    """Построение графиков с дополнительной информацией."""
    plt.figure(figsize=(14, 12))

    # Конвертация времени в микросекунды
    t_micro = t_continuous * 1e6
    tau_micro = tau * 1e6
    valid_t_k_micro = valid_t_k * 1e6
    delta_tau_micro = delta_tau * 1e6
    omega_eff_hz = omega_eff / (2 * np.pi) # Перевод в Гц
    nyquist_freq = 1 / (2 * delta_tau) # Частота Найквиста в Гц

    # Создаем форматированные строки с параметрами
    params_base = (
        f"Параметры сигнала:\n"
        f"Амплитуда: {E_max} В\n"
        f"Длительность: {tau_micro:.0f} мкс\n"
        f"Энергия: {E:.2f} В²·с"
    )

    params_sampling = (
        f"Параметры дискретизации:\n"
        f"Шаг выборки: {delta_tau_micro:.2f} мкс\n"
        f"Частота Найквиста: {nyquist_freq:.2f} Гц\n"
        f"Количество отсчётов: {len(valid_t_k)}"
    )

    params_spectrum = (
        f"Спектральные характеристики:\n"
        f"Эффективная ширина: {omega_eff_hz:.2f} Гц\n"
        f"Частота дискретизации: {1/delta_tau:.2f} Гц"
    )

    # График 1: Исходный и восстановленный сигнал
    plt.subplot(3, 1, 1)
    plt.plot(t_micro, original, label="Исходный сигнал", color="blue")
    plt.plot(t_micro, restored, label="Восстановленный", color="red",
linestyle="--")
    plt.text(
        0.65,
        0.15,
        params_base,
        transform=plt.gca().transAxes,
        bbox=dict(facecolor="white", alpha=0.8),
    )

    plt.title("Исходный и восстановленный сигналы")
    plt.xlabel("Время (мкс)")
    plt.ylabel("Амплитуда (В)")
    plt.grid(True)
    plt.xlim(-50, tau_micro + 50)

```

```

plt.legend(loc="upper right")

# График 2: Отсчётные значения
plt.subplot(3, 1, 2)
markerline, stemlines, _ = plt.stem(
    valid_t_k_micro, s_k_values, linefmt="C1-", markerfmt="C1o"
)
plt.setp(stemlines, "linewidth", 1)
plt.setp(markerline, "markersize", 4)
plt.text(
    0.65,
    0.15,
    params_sampling,
    transform=plt.gca().transAxes,
    bbox=dict(facecolor="white", alpha=0.8),
)
plt.title("Дискретные отсчёты сигнала")
plt.xlabel("Время (мкс)")
plt.ylabel("Амплитуда (В)")
plt.grid(True)
plt.xlim(-50, tau_micro + 50)

# График 3: Спектральная плотность мощности
plt.subplot(3, 1, 3)
omega = np.linspace(0, 2 * np.pi * 2e6, 1000)
spectrum = (np.sinc(omega * tau / (2 * np.pi))) ** 2 * (E_max * tau) ** 2
plt.plot(omega / (2 * np.pi), spectrum) # Ось X теперь в Гц

plt.axvline(omega_eff_hz, color="red", linestyle="--")
plt.text(
    0.65,
    0.15,
    params_spectrum,
    transform=plt.gca().transAxes,
    bbox=dict(facecolor="white", alpha=0.8),
)
plt.title("Спектральная плотность мощности")
plt.xlabel("Частота (Гц)")
plt.ylabel("Мощность (В2·с2/Гц2)")
plt.grid(True)
plt.tight_layout()
plt.show()

# Основные параметры
E_max = 34 # Амплитуда (В)
tau = 700e-6 # Длительность импульса (с)
target = 0.99 * np.pi / 2

# Расчёты
E = calculate_energy(E_max, tau)

```

```

x_eff = find_effective_bandwidth(target)
omega_eff = 2 * x_eff / tau
delta_tau = np.pi / omega_eff

# Генерация отсчётов
valid_t_k = calculate_samples(tau, delta_tau)
s_k_values = E_max * np.ones_like(valid_t_k)

# Временная ось для графиков
t_continuous = np.linspace(-100e-6, tau + 100e-6, 10000)
original = np.where((t_continuous >= 0) & (t_continuous <= tau), E_max, 0)
restored = np.array(
    [restored_signal(t, valid_t_k, s_k_values, omega_eff) for t in t_continuous]
)

# Построение графиков
plot_results(
    t_continuous,
    original,
    restored,
    valid_t_k,
    s_k_values,
    tau,
    delta_tau,
    omega_eff,
    E,
)

```

Приложение 1 – программный код