МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Крымский федеральный университет им. В.И. Вернадского»

Физико-технический институт
Кафедра компьютерной инженерии и моделирования

Лабораторная работа № 5

«Сигналы с ограниченным спектром»
По дисциплине
«Обработка сигналов»

Выполнил:
Студент 3 курса
Направления подготовки 09.03.01 «Информатика и вычислительная техника» Группы ИВТ-б-о-222
Чудопалов Богдан Андреевич
Проверил:
Таран Е.П.
«_»_____20_г.
Оценка:_____ Подпись: _____

Симферополь, 2025

Цель работы: разработать программное обеспечение, которое будет осуществлять восстановление сигнала при помощи базиса Котельникова.

Техническое задание: задан произвольный сигнал. Необходимо разработать программное обеспечение, которое будет восстанавливать изначальный сигнал по его отсчётным значениям с использованием базиса Котельникова.

№ варианта	2
Сигналы (рисунок 3.1)	б
E_{max}, В	18
$t_и$, мкс	2

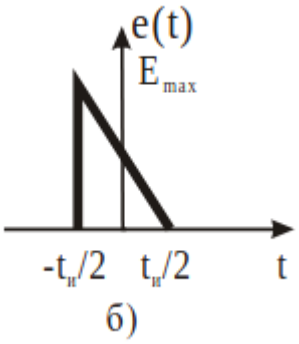

Рис. 1. Индивидуальный вариант.

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.

Ход работы

Вывод программы имеет вид:

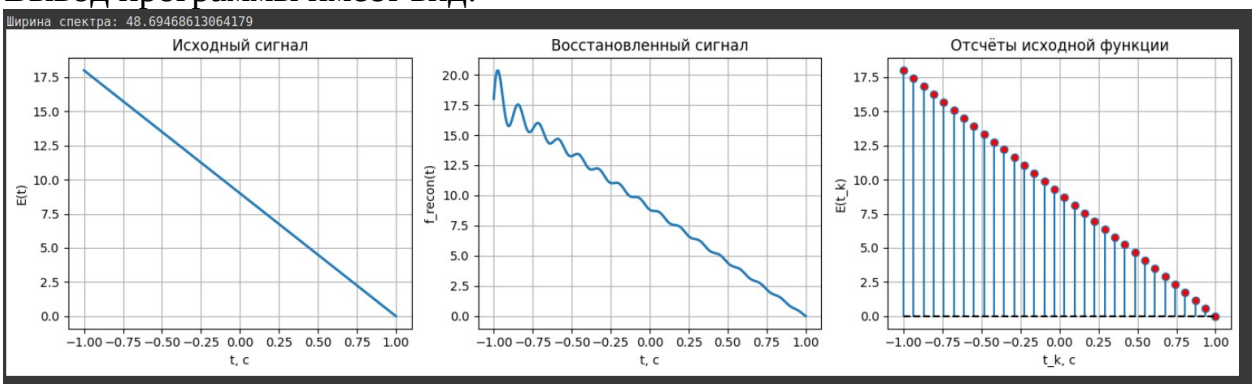

Рис. 1. Вывод программы.

Теоретические сведения и формулы для выполнения работы

Энергия Сигнала

Энергия сигнала вычисляется как интеграл от квадрата функции сигнала:

$$E = \int_{-\frac{T}{2}}^{\frac{T}{2}} |E(t)|^2 dt$$

В коде это вычисляется численно с использованием метода трапеций[2]:

$$E \approx \frac{h}{2} \sum_{i=0}^{N} w_i |E(t_i)|^2$$

где $h = \frac{T}{N}$, $t_i = -\frac{T}{2} + i \cdot h$, а w_i - весовые коэффициенты (1 для крайних точек и 2 для внутренних).

Преобразование Фурье

Преобразование Фурье функции $E(t)$ определяется как:

$$S(\omega) = \int_{-\infty}^{\infty} E(t) \cdot e^{-j\omega t} dt$$

Поскольку функция ограничена интервалом $[-T/2, T/2]$, интеграл фактически вычисляется как:

$$S(\omega) = \int_{-\frac{T}{2}}^{\frac{T}{2}} E(t) \cdot e^{-j\omega t} dt$$

Действительная и мнимая части преобразования Фурье вычисляются отдельно[1] [3]:

$$\mathrm{Re}[S(\omega)] = \int_{-\frac{T}{2}}^{\frac{T}{2}} E(t) \cdot \cos(\omega t) dt$$

$$\mathrm{Im}[S(\omega)] = \int_{-\frac{T}{2}}^{\frac{T}{2}} E(t) \cdot \sin(\omega t) dt$$

Спектральная Плотность

Спектральная плотность (модуль Фурье-образа) вычисляется как:

$$|S(\omega)| = \sqrt{(\mathrm{Re}[S(\omega)])^2 + (\mathrm{Im}[S(\omega)])^2}$$

Для треугольного импульса известно, что Фурье-образ связан с функцией sinc^2[1] [3]:

$$|S(\omega)| \propto \left| \frac{\sin(\omega T/2)}{\omega T/2} \right|^2$$

Вывод: В ходе выполнения лабораторной работы была определена эффективная ширина спектра с помощью базиса Котельникова и на ее основе был восстановлен исходный сигнал.

Приложение

```
def Func(t):
    if t < -T/2 or t > T/2:
        return 0.0
    elif math.isclose(t, -T/2, rel_tol=1e-12, abs_tol=1e-12):
        return E_max
    else:
        return E_max * (1 - ((t + T/2) / T))
```

1.1.Определение функции сигнала.

```
def signal_energy():
    a = -T/2
    b = T/2
    h = (b - a) / N
    E = 0.0
    for i in range(N + 1):
        x = a + i * h
        val = Func(x) ** 2
        if i == 0 or i == N:
            E += val
        else:
            E += 2 * val
    E *= (h / 2.0)
    return E
```

1.2. Вычисление энергии сигнала.

```python
def Real_calculate(w):
    a = -T/2
    b = T/2
    h = (b - a) / N
    result = 0.0
    for i in range(N + 1):
        x = a + i * h
        val = Func(x) * math.cos(w * x)
        if i == 0 or i == N:
            result += val
        else:
            result += 2 * val
    return result * (h / 2.0)

def Imag_calculate(w):
    a = -T/2
    b = T/2
    h = (b - a) / N
    result = 0.0
    for i in range(N + 1):
        x = a + i * h
        val = Func(x) * math.sin(w * x)
        if i == 0 or i == N:
            result += val
        else:
            result += 2 * val
    return result * (h / 2.0)
```

1.3. Вещественная и мнимая часть коэффициентов Фурье.

```python
def S_w_calculate(w):
    real = Real_calculate(w)
    imag = Imag_calculate(w)
    return math.sqrt(real * real + imag * imag)

def E_w_calculate(w, E):
    a = 0.0
    b = w
    steps = 1000
    h = (b - a) / steps
    integral = 0.0
    for i in range(steps + 1):
        w_i = a + i * h
        val = (S_w_calculate(w_i) ** 2)
        if i == 0 or i == steps:
            integral += val
        else:
            integral += 2 * val
    integral *= (h / 2.0) * (1.0 / math.pi)
    return integral
```

1.4 Вычисление спектральной амплитуды и спектральной энергии.

```python
if w_final == 0:
    w_final = w_start
delta_t = math.pi / w_final if w_final != 0 else T / 10
if delta_t <= 0:
    delta_t = T / 10

k_max = int(T / delta_t)
t_points = [ -T/2 + k * delta_t for k in range(k_max + 1) ]
f_points = [ Func(tp) for tp in t_points ]
```

1.5. Определение дискретных отсчетов.

```python
t_recon = np.linspace(-T/2, T/2, 1000)
recon_vals = []
for tr in t_recon:
    val = 0.0
    for t_k, f_k in zip(t_points, f_points):
        denom = w_final * (tr - t_k)
        if math.isclose(tr, t_k, rel_tol=1e-14):
            val += f_k
        else:
            val += f_k * (math.sin(denom) / denom)
    recon_vals.append(val)
```

1.6. Реконструкция сигнала.

```python
axes[1].plot(t_recon, recon_vals, lw=2)
axes[1].set_title("Восстановленный сигнал")
axes[1].set_xlabel("t, c")
axes[1].set_ylabel("f_recon(t)")
axes[1].grid(True)

markerline, stemlines, baseline = axes[2].stem(t_points, f_points, basefmt="k--")
plt.setp(markerline, 'markerfacecolor', 'red')
axes[2].set_title("Отсчёты исходной функции")
axes[2].set_xlabel("t_k, c")
axes[2].set_ylabel("E(t_k)")
axes[2].grid(True)

plt.tight_layout()
plt.show()
```

1.7. Построение графиков.