Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Отчет по лабораторной работе №1, 2

Сигналы телекоммуникационных систем. Ряд Фурье. Преобразование Фурье. Корреляция

Работу выполнил:
Графов Д.И.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

1. Цель работы

- Познакомиться со средствами генерации и визуализации простых сигналов.
- Получить представление о спектрах телекоммуникационных сигналов.

2. Программа работы

- С помощью языка программирования Python и его библиотек промоделировать синусоидальный и прямоугольный сигналы с различными параметрами. Получить их спектры. Вывести на график.
-
  - Для сигналов, построенных в лабораторной работе №1, выполните расчет преобразования Фурье. Перечислите свойства преобразования Фурье.
  - С помощью функции корреляции найдите позицию синхропосылки [101] в сигнале [0001010111000010]. Получите пакет данных, если известно, что его длина составляет 8 бит без учета синхропосылки. Вычислите корреляцию прямым методом, воспользуйтесь алгоритмом быстрой корреляции, сравните время работы обоих алгоритмов.

3. Теоретическая информация

3.1. Python и используемые библиотеки

Среди множества библиотек Python выделим основные, используемые для математических расчётов и визуализации.

- **NumPy** – это open-source модуль для Python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций. Они объединяются в высокоуровневые пакеты. Они обеспечивают функционал, который можно сравнить с функционалом MatLab. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами. SciPy (Scientific Python) расширяет функционал numpy огромной коллекцией полезных алгоритмов, таких как минимизация, преобразование Фурье, регрессия, и другие прикладные математические техники.
- **Matplotlib** — библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикация.

  Генерируемые в различных форматах изображения могут быть использованы в интерактивной графике, в научных публикациях, графическом интерфейсе пользователя, веб-приложениях, где требуется построение диаграмм (англ. plotting). В документации автор признаётся, что Matplotlib начинался с подражания графическим командам MATLAB, но является независимым от него проектом.

  Библиотека Matplotlib построена на принципах ООП, но имеет процедурный интерфейс pylab, который предоставляет аналоги команд MATLAB.

3.2. Сигнал и его спектр

- *Сигнал* - это физическое явление, служащее для передачи информации, которое может иметь различную природу. Должен также иметь различимые состояния (минимум 2), чтобы передавать информацию (например, наличие сигнала и его отсутствие).

- *Спектр сигнала* - это результат разложения сигнала на более простые в базисе ортогональных функций. В качестве разложения обычно используются преобразование Фурье и другие.

В радиотехнике в качестве базисных функций используют синусоидальные функции. Это объясняется рядом причин:

- гармоническое колебание является единственной функцией времени, сохраняющей свою форму при прохождении колебания через линейную систему с постоянными параметрами, могут только изменяться амплитуда и фаза;

- для гармонических функций имеется математический аппарат комплексного анализа;

- гармоническое колебание легко реализуемо на практике.

- Спектр сигнала $s(t)$ можно записать через преобразование Фурье (можно без коэффициента $1/\sqrt{2\pi}$) в виде:

  $S(\omega) = \int_{-\infty}^{+\infty} s(t)e^{-i\omega t}dt$, где ω - угловая частота равная $2\pi f$.

  Спектр сигнала является комплексной величиной и представляется в виде:
  $S(\omega) = A(\omega)e^{-i\phi(\omega)}$, где $A(\omega)$ - амплитудно-частотная характеристика сигнала, $\phi(\omega)$ - фазо-частотная характеристика сигнала.

3.3. Свойства преобразования Фурье

Перечислим некоторые свойства преобразования Фурье.

- Преобразование Фурье является линейным оператором.

- Свойство временного сдвига: задержка сигнала во времени приводит к изменению фазы его спектральной плотности без изменения амплитуды.

- Преобразование Фурье свертки сигналов: спектральная плотность свертки двух сигналов равна произведению их спектральных плотностей.

- Преобразование Фурье произведения сигналов: преобразование Фурье произведения сигналов пропорционально свертке спектральных плотностей этих сигналов.

4. Ход выполнения работы

4.1. Лабораторная работа №1

На языке python мной была написана программа, генерирующая синусоидальный и прямоугольный сигналы, а также отображающая их спектры.

Листинг 1. main.py

```python
import matplotlib.pyplot as plt
import numpy as np


# x(t) = A * sin(2 * pi * w * t)
def sine_wave(freq=20, show=False, save=False):
    fs = 1000  # sampling rate
    ts = 1 / fs  # sampling interval
    n = 8192  # number of fft points, pick power of 2
    t = np.arange(0, n * ts, ts)  # time vector
    sig = np.sin(2 * np.pi * freq * t)  # signal
    sig_fft = np.fft.fft(sig) / n * 2  # /N to scale due to python DFT equation,
    # *2 to make single sided
    fft_freq = np.fft.fftfreq(n, ts)  # python function to get Hz frequency axis

    plt.figure()
    plt.plot(t[0:250], sig[0:250])
    plt.xlabel('time (S)')
    plt.ylabel('amplitude (V)')
    plt.grid(True)
    if save:
        plt.savefig('../out/sine_time.png')
    if show:
        plt.show()

    plt.figure()
    plt.plot(fft_freq[:fs], abs(sig_fft)[:fs])
    plt.xlabel('frequency (Hz)')
    plt.ylabel('amplitude (V)')
    # plt.plot(fft_freq[:n // 2], 20 * np.log10(abs((sig_fft[:n // 2]))))
    # plt.ylabel('amplitude (dB20(V))')
    plt.grid(True)
    if save:
        plt.savefig('../out/sine_freq.png')
    if show:
        plt.show()


# x(t) = sign(sin(t))
def square_wave(freq=20, show=False, save=False):
    fs = 1000  # sampling rate
    ts = 1 / fs  # sampling interval
```

```python
    n = 8192  # number of fft points, pick power of 2
    t = np.arange(0, n * ts, ts)  # time vector
    sig = np.sign(np.sin(2 * np.pi * freq * t))  # signal
    sig_fft = np.fft.fft(sig) / n * 2  # /N to scale due to python DFT equation,
    # *2 to make single sided
    fft_freq = np.fft.fftfreq(n, ts)  # python function to get Hz frequency axis

    plt.figure()
    plt.plot(t[0:250], sig[0:250])
    plt.xlabel('time (S)')
    plt.ylabel('amplitude (V)')
    plt.grid(True)
    if save:
        plt.savefig('../out/square_time.png')
    if show:
        plt.show()

    plt.figure()
    plt.plot(fft_freq[:fs], abs(sig_fft)[:fs])
    plt.xlabel('frequency (Hz)')
    plt.ylabel('amplitude (V)')
    # plt.plot(fft_freq[:n // 2], 20 * np.log10(abs((sig_fft[:n // 2]))))
    # plt.ylabel('amplitude (dB20(V))')
    plt.grid(True)
    if save:
        plt.savefig('../out/square_freq.png')
    if show:
        plt.show()


if __name__ == '__main__':
    sine_wave(show=True, save=True)
    square_wave(show=True, save=True)
```

Результат работы

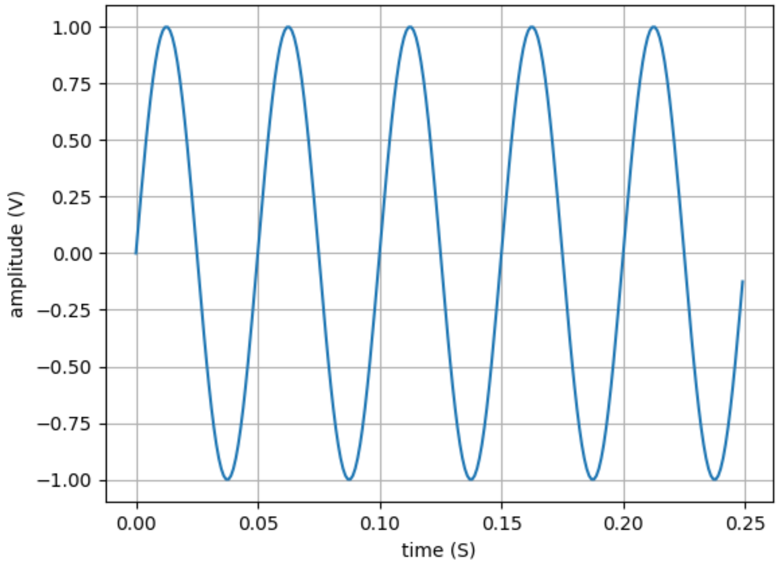

Рисунок 4.1. Синусоидальный сигнал

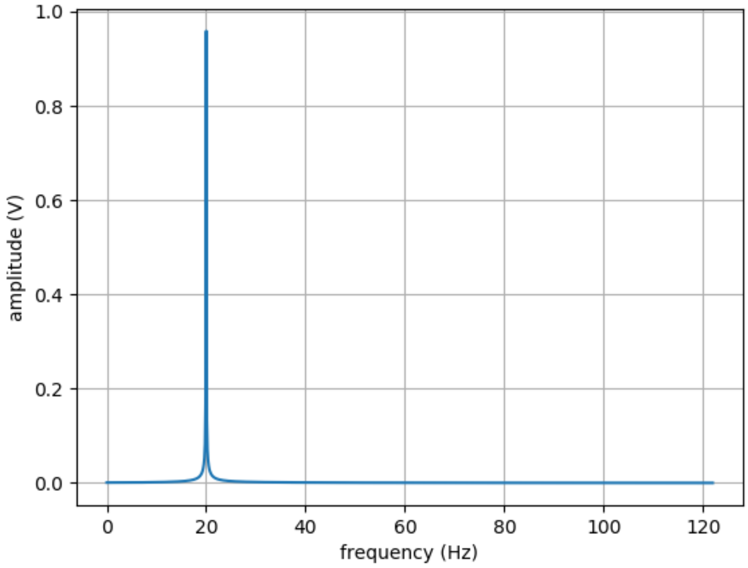

Рисунок 4.2. Спектр синусоидального сигнала

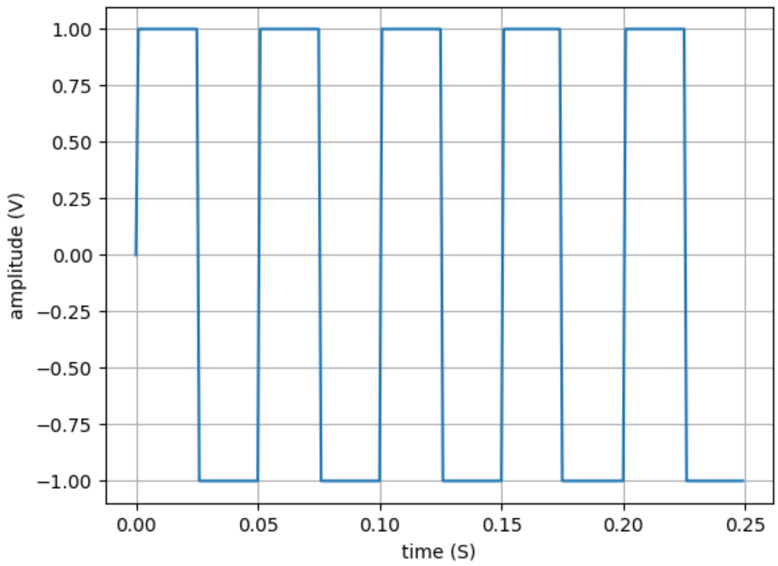

Рисунок 4.3. Прямоугольный сигнал

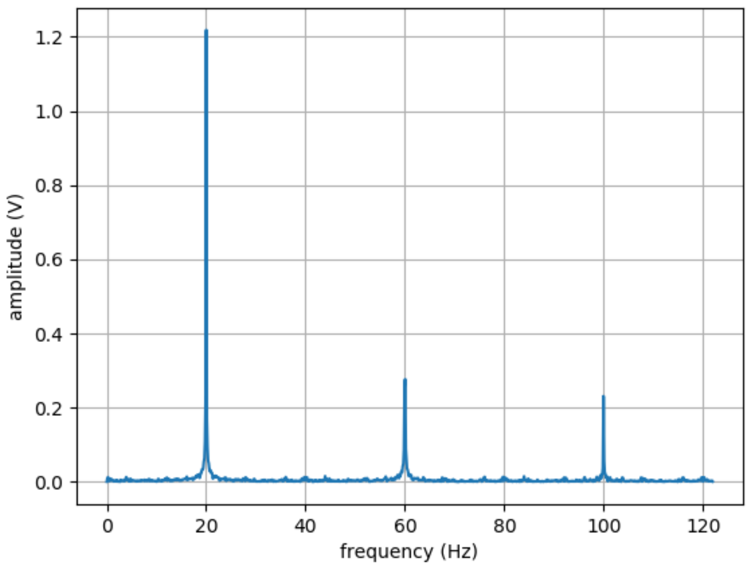

Рисунок 4.4. Спектр прямоугольного сигнала

4.2. Лабораторная работа №2

4.2.1. Расчёт преобразования Фурье

Вспомогательные формулы:

- $\sin(\omega_0 t) = \frac{e^{i\omega_0 t} - e^{-i\omega_0 t}}{2i}$
- Дельта-функция: $\delta(\omega) = \frac{1}{2\pi}\int_{\infty}^{\infty} e^{i\omega t}dt$
- $\delta(t) = \delta(-t)$

$F(\omega) = \frac{1}{\sqrt{2\pi}}\int_{\infty}^{\infty} f(t)e^{-i\omega t}dt = \frac{1}{\sqrt{2\pi}}\int_{\infty}^{\infty}\sin(\omega_0 t)e^{-i\omega t}dt = \frac{1}{\sqrt{2\pi}}\int_{\infty}^{\infty}\frac{e^{i\omega_0 t} - e^{-i\omega_0 t}}{2i}e^{-i\omega t}dt = \frac{\sqrt{2\pi}}{2\pi 2i}\int_{\infty}^{\infty}(e^{it(\omega-\omega_0)} - e^{-it(\omega-\omega_0)})dt = \frac{\sqrt{2\pi}}{2i}(\delta(\omega-\omega_0) - \delta(\omega+\omega_0))$

4.2.2. Программа

Листинг 2. main.py

```python
import numpy as np
from scipy import signal
import time
import matplotlib.pyplot as plt


def position(correlations, sinc_package):
    for i in range(0, len(correlations) - 3):
        if sum(correlations[i:i + 3]) == sum(sinc_package):
            return i + 1


# x(t) = convolve(sign(sin(t)), sign(sin(t)))
def triangle_wave(freq=20, show=False, save=False):
    fs = 1000  # sampling rate
    ts = 1 / fs  # sampling interval
    n = 8192  # number of fft points, pick power of 2
    t = np.arange(0, n * ts, ts)  # time vector
    sig = np.convolve(np.sign(np.sin(2 * np.pi * freq * t)), np.sign(np.sin(2 * np.pi * freq * t)), 'sa
    sig_fft = np.fft.fft(sig) / n * 2  # /N to scale due to python DFT equation,
    # *2 to make single sided
    fft_freq = np.fft.fftfreq(n, ts)  # python function to get Hz frequency axis

    plt.figure()
    plt.plot(t[0:250], sig[0:250])
    plt.xlabel('time (S)')
    plt.ylabel('amplitude (V)')
    plt.grid(True)
    if save:
        plt.savefig('../out/triangle_time.png')
    if show:
```

```python
        plt.show()

    plt.figure()
    plt.plot(fft_freq[:fs], abs(sig_fft)[:fs])
    plt.xlabel('frequency (Hz)')
    plt.ylabel('amplitude (V)')
    plt.grid(True)
    if save:
        plt.savefig('../out/triangle_freq.png')
    if show:
        plt.show()


def package():
    sinc_package = np.array([1, 0, 1], dtype=int)
    sig = np.array([0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0], dtype=int)

    start_time = time.time()
    correlations_direct = signal.correlate(sig, sinc_package, mode='valid', method='direct')  # прямой
    print("Direct method:\n--- %s seconds ---" % (time.time() - start_time))

    start_time = time.time()
    correlations_fft = signal.correlate(sig, sinc_package, mode='valid', method='fft')  # быстрая корре
    print("FFT method:\n--- %s seconds ---" % (time.time() - start_time))

    print("Correlations using direct method:", correlations_direct)
    print("Correlations using FFT method   :", correlations_fft)
    pos = position(correlations_direct, sinc_package)
    print("Position with direct correlation:", pos)
    print("Position with FFT correlation   :", position(correlations_fft, sinc_package))
    package = sig[pos + 3:][:8]
    print("Package: ", package)


if __name__ == '__main__':
    package()
    triangle_wave(show=True, save=True)
```

Результат работы

```
Direct method:
--- 2.5033950805664062e-05 seconds ---
FFT method:
--- 0.0203402042388916 seconds ---
Correlations using direct method: [0 1 0 2 0 2 1 2 1 1 0 0 0 1 0]
Correlations using FFT method   : [0 1 0 2 0 2 1 2 1 1 0 0 0 1 0]
Position with direct correlation: 3
Position with FFT correlation   : 3
Package:  [0 1 1 1 0 0 0 0]
```

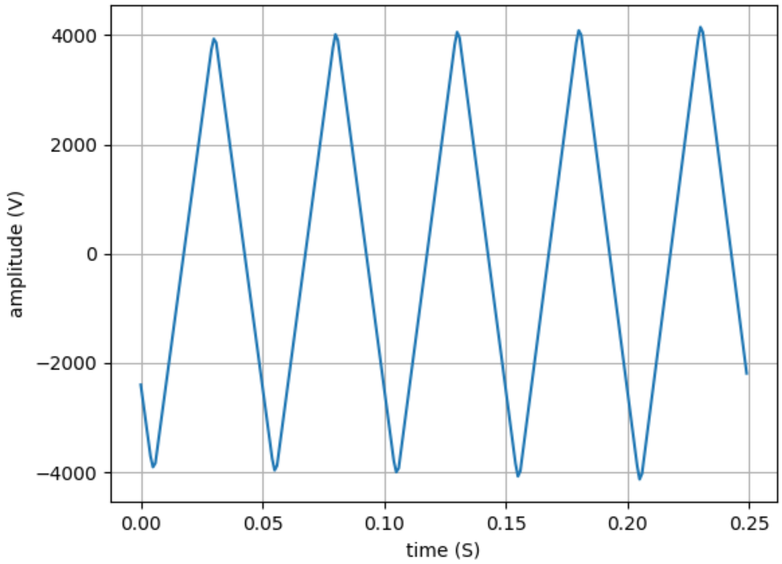

Рисунок 4.5. Треугольный сигнал

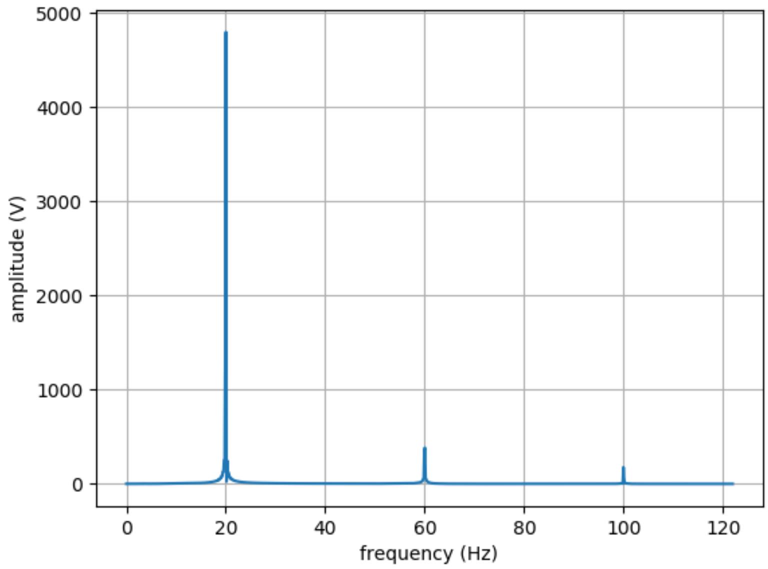

Рисунок 4.6. Спектр треугольного сигнала

5. Выводы

В данных лабораторных работах (1 и 2) мной были промоделированы синусоидальный и прямоугольные сигналы, получены их спектры.

Был проведён расчёт преобразования Фурье для синусоидального сигнала. Были перечислены свойства данного преобразования.

С помощью функции корреляции была найдена позиция синхропосылки в сигнале, был получен пакет данных. Корреляция была вычислена прямым методом, и методом быстрой корреляции.

В ходе выполнения работы я познакомился со средствами генерации сигналов, их визуализации. Также было получено представление о спектрах телекоммуникационных сигналов.