Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Отчет по лабораторной работе №1
Сигналы телекоммуникационных систем

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.

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.

3. Теоретическая информация

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.

- *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:

- функции $cos(\omega t)$, $sin(\omega t)$ являются простыми и определены при всех значениях t, являются ортогональными и составляют полный набор при кратном уменьшении периода;

- 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.

4. Ход выполнения работы

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

4.1. Листинг 1. main.py

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


# x(t) = A * sin(w*t + fi_0)
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.cos(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
    n = 8192  # number of fft points, pick power of 2
    t = np.arange(0, n * ts, ts)  # time vector
    sig = np.sign(np.cos(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)
```

4.2. Результат работы

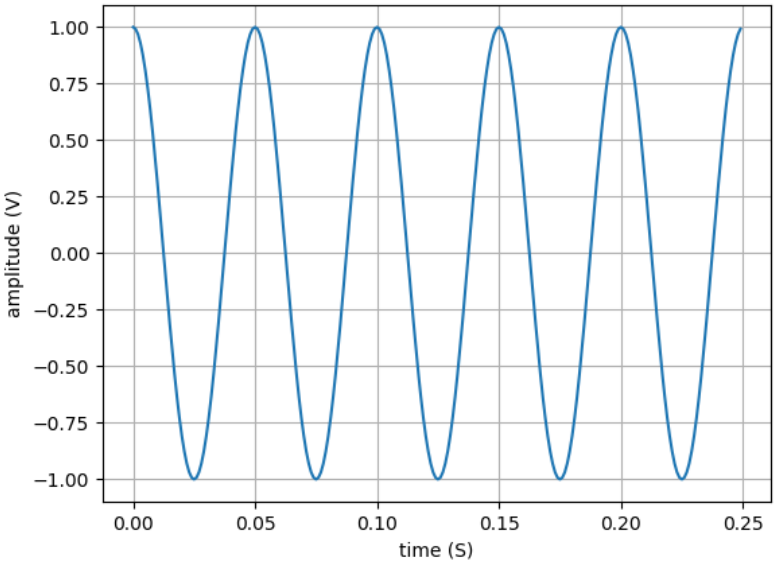

Рисунок 4.1. Синусоидальный сигнал

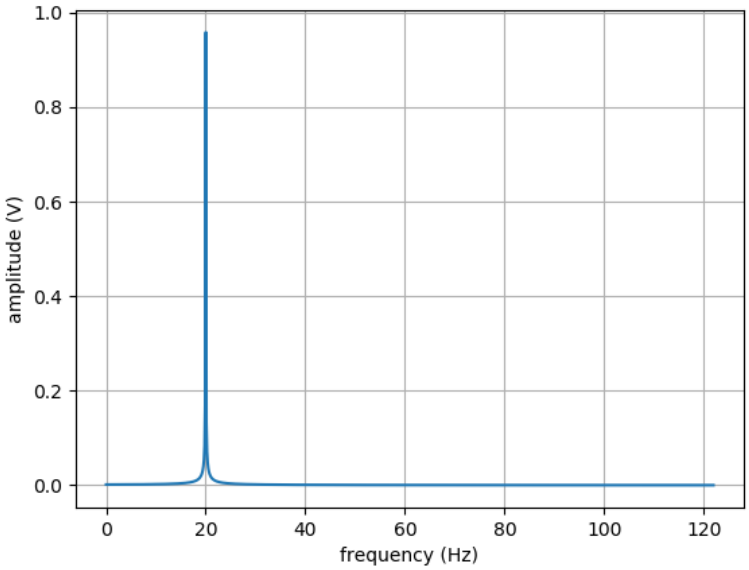

Рисунок 4.2. Спектр синусоидального сигнала

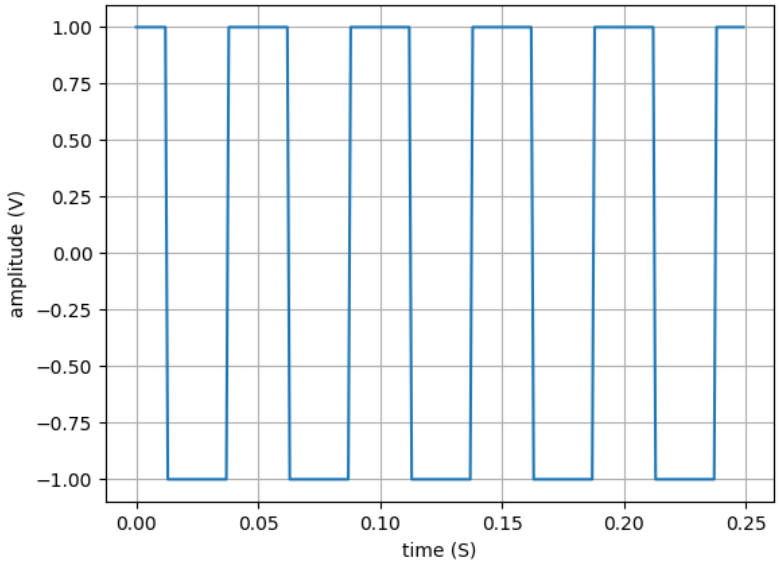

Рисунок 4.3. Прямоугольный сигнал

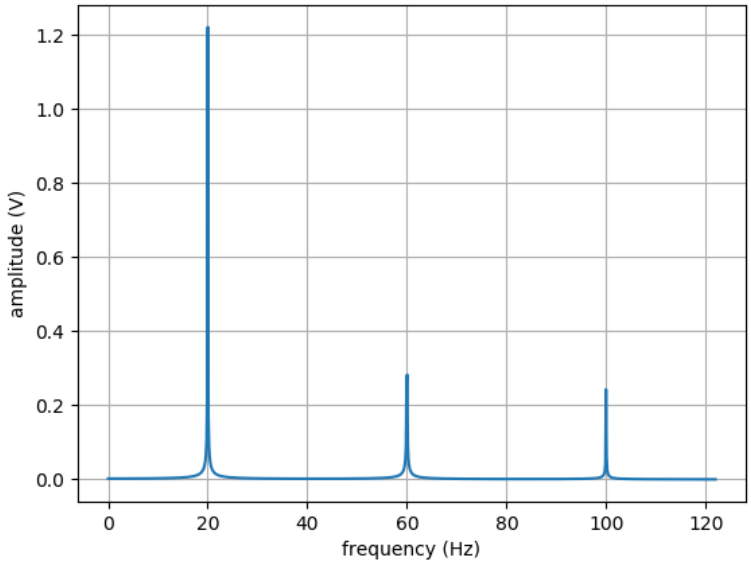

Рисунок 4.4. Спектр прямоугольного сигнала

5. Выводы

В данной лабораторной работе по курсу "Телекоммуникационные технологии"мной были промоделированы синусоидальный и прямоугольный сигналы, а также получены их спектры.

Сигналы бывают периодические и непериодические. Для периодических сигналов выполняется общее условие $s(t) = s(t + kT)$, где $k = 1, 2, 3, ...$ - любое целое число, - период, являющийся конечным отрезком независимой переменной.

В данной лабораторной работе мной выяснено, что периодический сигнал имеет дискретный спектр. Если же сигнал не периодический, то спектр в этом случае будет непрерывным.