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:
Косенков М.А.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

1. Цель работы

Изучить воздействие частотного фильтра на сигнал с шумом.

2. Программа работы

- Сгенерировать гармонические сигналы с шумом и синтезировать частотные фильтры для них:
  - синусоидальный сигнал
  - импульсный сигнал
  - пилообразный сигнал
- Получить сигнал во временной и частотной областях до и после фильтрации.
- Сделать выводы о воздействии частотного фильтра на спектр сигнала.

3. Теоретическая информация

Аддитивный белый гауссовский шум (АБГШ, англ. AWGN) — вид мешающего воздействия в канале передачи информации. Характеризуется равномерной, то есть одинаковой на всех частотах, спектральной плотностью мощности, нормально распределёнными временными значениями и аддитивным способом воздействия на сигнал. Наиболее распространённый вид шума, используемый для расчёта и моделирования систем радиосвязи. Термин «аддитивный» означает, что данный вид шума суммируется с полезным сигналом и статистически не зависим от сигнала. В противоположность аддитивному, можно указать мультипликативный шум — шум, перемножающийся с сигналом.

4. Ход работы

Данная работа выполнялась на языке Python.

Для добавления шума было использовано простое решение: Использовать стандартную функцию библиотеки NumPy random.randomsample() для генерации массива, аналогичного по длине исходному сигналу. Значения брались на отрезке [-1;1] Затем данный шум был прибавлен к исходным гармоническим сигналам с амплитудой, равной 1 и частотой, равной 20 Гц.

Для фильтрации сигнала было принято решение использовать функцию библиотеки SciPy signal.butter(), реализующую фильтр Баттерворта. Фильтрация происходила на отрезке [0,019; 0,02] - по нормированной шкале (техническое требование к функции signal.butter). Порядок фильтра был выбран равным 4 с помощью подбора.

Далее была осуществлена фильтрация с помощью filtfilt() пакета signal. Затем были получены графики во временных и частотных областях исходного и отфильтрованного сигнала.

Листинг 1. lab3.py

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


def get_plot(x, y, x_label, y_label, title, show, save, close):
    #plt.figure()
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title)
    plt.plot(x, y)
    plt.grid(True)
    if show:
        plt.show()
    if save:
        plt.savefig(title + '.png')
    if close:
        plt.close()


# синусоидальный сигнал
def sin_signal(time, frequency, amplit):
    sig = amplit * np.sin(2 * np.pi * frequency * time)
    return sig

def get_fft_signal(num, sampling, sig):
    # преобразование Фурье
    fft_sig = np.fft.fft(sig) / num * 2
    # частота
    freq_fft = np.fft.fftfreq(num, 1 / sampling)
    limit = sampling // 2
    return fft_sig, freq_fft, limit

```

```
34
35 if __name__ == '__main__':
36     fs = 1000
37     number = 2048
38     t = np.arange(0, number / fs, 1 / fs)
39     freq = 20
40     amplitude = 1
41     sig_array = [sin_signal(t, freq, amplitude)]
42     for input_signal in sig_array:
43         sig_fft, fft_freq, lim = get_fft_signal(number, fs, input_signal)
44         # добавляем шум
45         noise = 2 * np.random.random_sample(input_signal.size, ) - 1
46         sig_with_noise = input_signal + noise
47         get_plot(x=t[:lim], y=sig_with_noise[:lim], x_label='Time',
48                  y_label='Amplitude', title='Noise plot',
49                  show=False, save=False, close=False)
50         # график сигнала
51         get_plot(x=t[:lim], y=input_signal[:lim], x_label='Time',
52                  y_label='Amplitude', title='Signal plot',
53                  show=True, save=False, close=True)
54         # график спектра сигнала
55         get_plot(x=fft_freq[:lim], y=sig_fft[:lim], x_label='Frequency',
56                  y_label='Amplitude', title='Spectrum plot',
57                  show=False, save=False, close=True)
58         # создаем фильтр и применяем на зашумленный сигнал
59         b, a = signal.butter(4, Wn=[(freq - 1) / 500, (freq + 1) / 500], btype='bandpass')
60         filtered = signal.filtfilt(b, a, sig_with_noise)
61         get_plot(x=t[:lim], y=filtered[:lim], x_label='Time',
62                  y_label='Amplitude', title='Filtered signal',
63                  show=True, save=False, close=True)
64         # спектр отфильтрованного сигнала
65         filt_sig_fft, filt_fft_freq, filt_lim = get_fft_signal(number, fs, filtered)
66         get_plot(x=filt_fft_freq[:filt_lim], y=filt_sig_fft[:filt_lim], x_label='Frequency',
67                  y_label='Amplitude', title='FILTERED SPECTRUM',
68                  show=False, save=False, close=True)
```

Результат работы

Синусоидальный сигнал

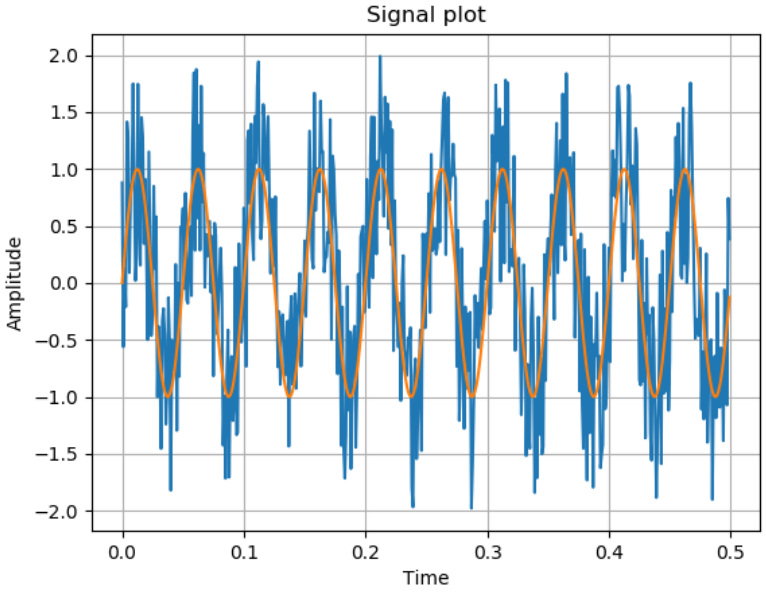

Рисунок 4.1. Синусоидальный сигнал (с шумом)

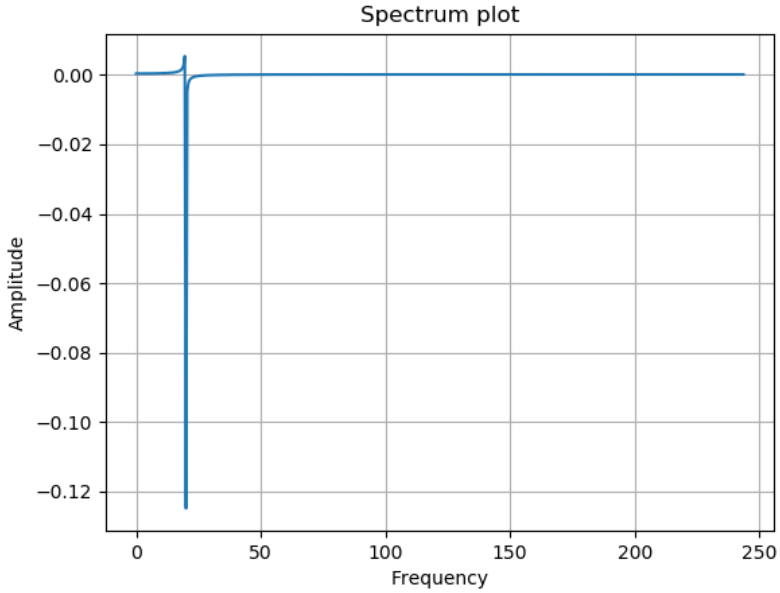

Рисунок 4.2. Спектр синусоидального сигнала

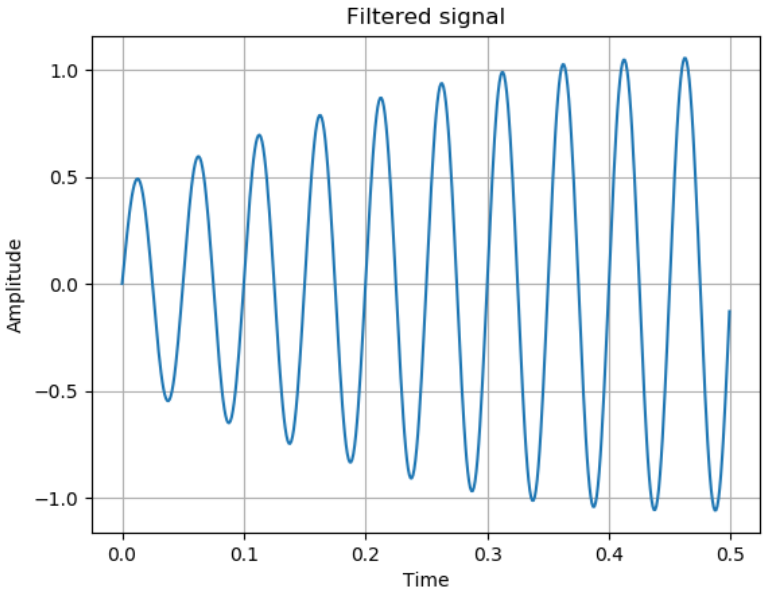

Рисунок 4.3. Отфильтрованный сигнал

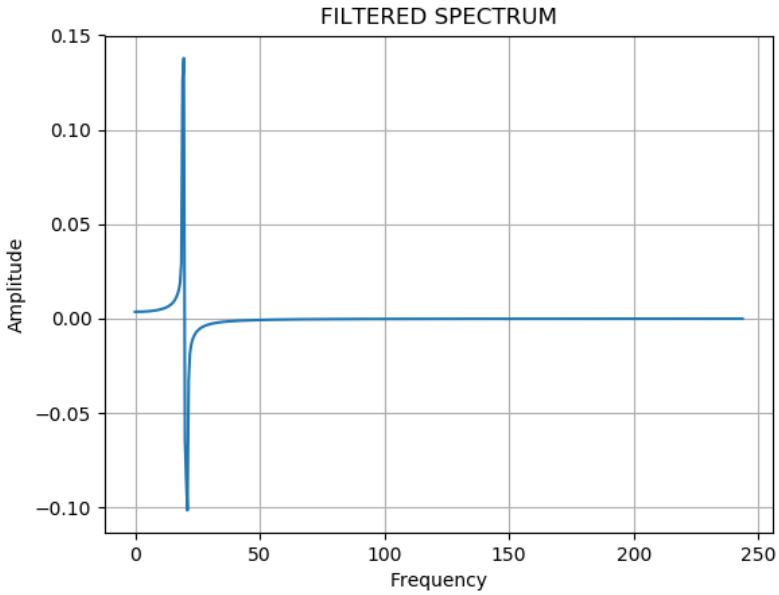

Рисунок 4.4. Спектр отфильтрованного сигнала

Как можно видеть на рисунках, частота сигнала сохранилась после фильтра, однако амплитуда не является постоянной, а увеличивается по закону, похожему на логарифмический. Это происходит вследствие специфики работы фильтра. С большой долей вероятности можно утверждать, что амплитуда станет постоянной через большее время.

5. Выводы

В ходе выполнения работы мной была осуществлена фильтрация гармонического сигнала.

Исходя из графика временной шкалы отфильтрованного сигнала, можно утверждать, что была сохранена амплитуда. В начале фильтрации потребовалось некоторое время, чтобы установить постоянную амплитуду сигнала, тем не менее она совпала с желаемой.

Спектр отфильтрованного сигнала совпадает со спектром исходного, что свидетельствует о том, что удалось сохранить и частотную характеристику сигнала.