

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

Телекоммуникационные технологии

Отчет по лабораторной работе №3

Линейная фильтрация

Работу

выполнил:

Графов Д.И.

Группа: 33531/2

Преподаватель:

Богач Н.В.

Санкт-Петербург
2019

Содержание

1. Цель работы	3
2. Программа работы	3
3. Теоретическая информация	3
4. Ход выполнения работы	4
5. Выводы	7

1. Цель работы

Изучить воздействие ФНЧ на тестовый сигнал с шумом.

2. Программа работы

- Сгенерировать гармонический сигнал с шумом и синтезировать ФНЧ.
- Получить сигнал во временной и частотной областях до и после фильтрации.
- Сделать выводы о воздействии ФНЧ 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 зависит от сигнала. В противоположность аддитивному, можно указать мультипликативный шум — шум, перемножающийся с сигналом.

4. Ход выполнения работы

Данная работа выполнялась на языке Python.

В качестве шума был выбран аддитивный белый гауссовский шум с параметрами:

- математическое ожидание: 0;
- среднеквадратичное отклонение: 0.5.

Затем данный шум был прибавлен к исходному гармоническому сигналу с амплитудой равной 1 и частотой равной 20 Гц.

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

Листинг 1. `main.py`

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from scipy import signal
4 from scipy.signal import lfilter
5
6
7 #  $x(t) = A * \sin(2 * \pi * w * t)$ 
8 def filtered_sine_wave(amplitude=1, freq=20, show=False, save=False):
9     fs = 1000 # sampling rate
10    ts = 1 / fs # sampling interval
11    n = 2 ** 13 # number of fft points, pick power of 2
12    t = np.arange(0, n * ts, ts) # time vector
13    sig = amplitude * np.sin(2 * np.pi * freq * t) # signal
14    noise = np.random.normal(0, 0.5, n)
15    sig_noised = sig + noise
16
17    # Nyquist frequency
18    nyq = fs / 2
19    order = 4
20    normal_cutoff = freq / nyq
21
22    fnum, fdenom = signal.butter(order, normal_cutoff)
23    sig_filtered = signal.filtfilt(fnum, fdenom, sig_noised)
24    # sig_filtered = lfilter(fnum, fdenom, sig_noised)
25
26    sig_noised_fft = np.fft.fft(sig_noised) / n * 2
27    sig_filtered_fft = np.fft.fft(sig_filtered) / n * 2 # /N to scale due to python DFT equation,
28    # *2 to make single sided
29    fft_freq = np.fft.fftfreq(n, ts) # python function to get Hz frequency axis
30
31    print_time(t, sig_noised, sig_filtered, show, save)
32    print_freq(fft_freq[:fs], abs(sig_noised_fft)[:fs], abs(sig_filtered_fft)[:fs], show, save)
33
34
35 def print_time(t, sig_noised, sig_filtered, show=False, save=False):
36     plt.figure()
```

```

37     plt.plot(t[:250], sig_noised[:250], label='Noised signal')
38     plt.plot(t[:250], sig_filtered[:250], label='Filtered signal')
39     plt.xlabel('time (S)')
40     plt.ylabel('amplitude (V)')
41     plt.legend()
42     plt.grid(True)
43     if save:
44         plt.savefig('../out/sine_time.png')
45     if show:
46         plt.show()
47
48
49 def print_freq(fft_freq, sig_noised_fft, sig_filtered_fft, show=False, save=False):
50     plt.figure()
51     plt.plot(fft_freq, sig_noised_fft, label='Noised signal')
52     plt.plot(fft_freq, sig_filtered_fft, label='Filtered signal')
53     plt.xlabel('frequency (Hz)')
54     plt.ylabel('amplitude (V)')
55     plt.legend()
56     plt.grid(True)
57     if save:
58         plt.savefig('../out/sine_freq.png')
59     if show:
60         plt.show()
61
62
63 if __name__ == '__main__':
64     filtered_sine_wave(show=True)

```

Результат работы

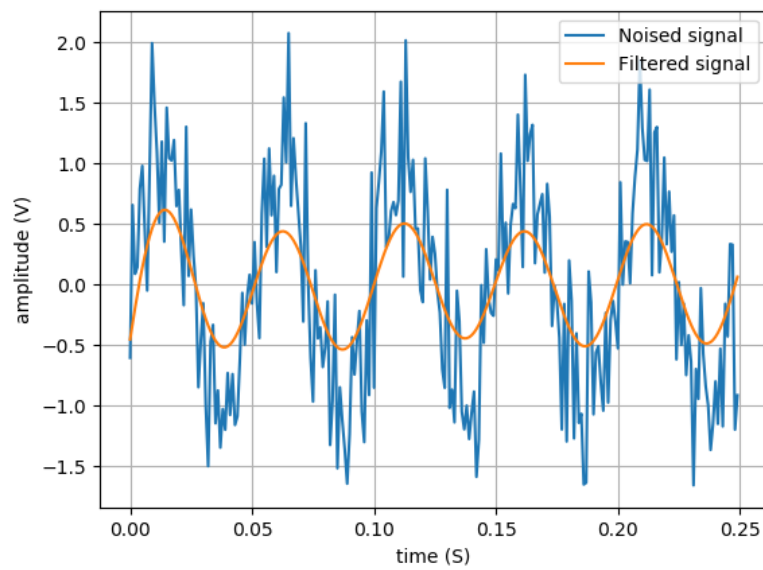


Рисунок 4.1. Синусоидальный сигнал

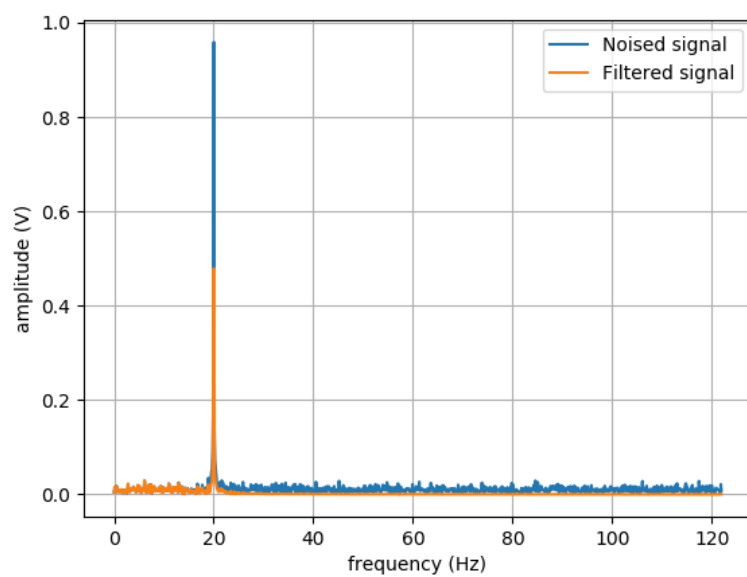


Рисунок 4.2. Спектр синусоидального сигнала

5. Выводы

В ходе выполнения работы мной была осуществлена фильтрация гармонического сигнала.