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

Телекоммуникационные системы

Отчет по лабораторной работе №3
Линейная фильтрация

Работу выполнила:
Васильева В.В.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

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

Фильтр нижних частот (ФНЧ) — электронный или любой другой фильтр, эффективно пропускающий частотный спектр сигнала ниже некоторой частоты (частоты среза) и подавляющий частоты сигнала выше этой частоты. Степень подавления каждой частоты зависит от вида фильтра.
Идеальный фильтр нижних частот (sinc-фильтр) полностью подавляет все частоты входного сигнала выше частоты среза и пропускает без изменений все частоты ниже частоты среза. Переходной зоны между частотами полосы подавления и полосы пропускания не существует. Идеальный фильтр нижних частот может быть реализован лишь теоретически с помощью умножения спектра (преобразования Фурье) входного сигнала на прямоугольную функцию в частотной области, или, что даёт тот же эффект, свёртки сигнала во временной области с sinc-функцией.
Фильтр с конечной импульсной характеристикой (нерекурсивный фильтр, КИХ-фильтр) — один из видов электронных фильтров, характерной особенностью которого является ограниченность по времени его импульсной характеристики (с какого-то момента времени она становится точно равной нулю). Знаменатель передаточной функции такого фильтра — некая константа.
Фильтр с бесконечной импульсной характеристикой (рекурсивный фильтр, БИХ-фильтр) — электронный фильтр, использующий один или более своих выходов в качестве входа, то есть образует обратную связь.Основным свойством таких фильтров является то, что их импульсная переходная характеристика имеет бесконечную длину во временной области, а передаточная функция имеет дробно-рациональный вид. Такие фильтры могут быть как аналоговыми так и цифровыми.

4. Ход выполнения работы

4.1. Листинг

```
1 from __future__ import print_function
2 from scipy import signal
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6
7 def plot_graphic(x, y, title=None, x_label="x", y_label="y", gr_form='-', xlim=
    ↪ None, ylim=None, show=False, save=False):
8     plt.xlabel(x_label)
```

```
9       plt.ylabel(y_label)
10      if title != None:
11          plt.title(title)
12
13      plt.plot(x, y, gr_form)
14
15      if xlim != None:
16          plt.xlim(xlim[0], xlim[1])
17      if ylim != None:
18          plt.ylim(ylim[0], ylim[1])
19
20      if show:
21          plt.show()
22      if save:
23          plt.savefig(title + '.png')
24          plt.close()
25
26
27  if __name__ == '__main__':
28      sig_freq = 5
29      T = 1.0 / sig_freq
30      ampl = 1
31      fs = 1000
32      ts = 1.0 / fs
33      n = 1 << 13
34
35      t = np.arange(0, n * ts, step=ts)
36
37      sig = ampl * np.sin(2 * np.pi * sig_freq * t)
38
39      noise = np.random.normal(0, 0.5, n)
40      sig_noise = sig + noise
41
42      plot_graphic(
43          x=t[:int((n - 1) / 2)], y=sig_noise[:int((n - 1) / 2)],
44          x_label='time(S)', y_label='amplitude (V)',
45          xlim=[0, 1], ylim=[-2, 2],
46          show=False
47      )
48
49      plot_graphic(
50          x=t[:int((n - 1) / 2)], y=sig[:int((n - 1) / 2)],
51          x_label='time(S)', y_label='amplitude (V)',
52          xlim=[0, 1], ylim=[-2, 2],
53          show=False
54      )
55      plt.legend(('noise', 'original'), loc='upper right', shadow=True)
56      # plt.savefig('graphics/noise&sig.png')
57      plt.show()
58
59      nyq = 0.5 * fs
60      Wn = 2 * sig_freq / nyq
61      N = 6
62
63      filt_func = signal.butter
64      fnum, fdenom = filt_func(N=N, Wn=Wn)
65      filtered = signal.filtfilt(fnum, fdenom, sig_noise)
66
67      plot_graphic(
68          x=t[:int((n - 1) / 2)], y=sig[:int((n - 1) / 2)],
```

```
69          x_label='time(S)', y_label='signal',
70          xlim=[0, 1], ylim=[-2, 2],
71          show=False)
72
73      plot_graphic(
74          x=t[:int((n - 1) / 2)], y=filtered[:int((n - 1) / 2)],
75          x_label='time(S)', y_label='signal',
76          xlim=[0, 1], ylim=[-2, 2],
77          show=False)
78
79      plt.legend(('signal', 'filtered signal'), loc='upper right', shadow=False)
80      # plt.savefig('graphics/filtered.png')
81      plt.show()
82
83      fft_freq = np.fft.fftfreq(n, ts)
84      sig_fft = np.fft.fft(sig_noise) / n * 2
85
86      plot_graphic(
87          x=fft_freq[:int((n - 1) / 2)], y=abs(sig_fft)[:int((n - 1) / 2)],
88          x_label='frequency (Hz)', y_label='amplitude (V)',
89          xlim=[0, 50],
90          show=False
91      )
92
93      fft_freq = np.fft.fftfreq(n, ts)
94      sig_fft = np.fft.fft(filtered) / n * 2
95
96      plot_graphic(
97          x=fft_freq[:int((n - 1) / 2)], y=abs(sig_fft)[:int((n - 1) / 2)],
98          x_label='frequency (Hz)', y_label='amplitude (V)',
99          xlim=[0, 50],
100         show=False
101     )
102
103     plt.legend(('signal', 'filtered signal'), loc='upper right', shadow=False)
104     # plt.savefig('graphics/sector.png')
105     plt.show()
```

4.2. Граффики

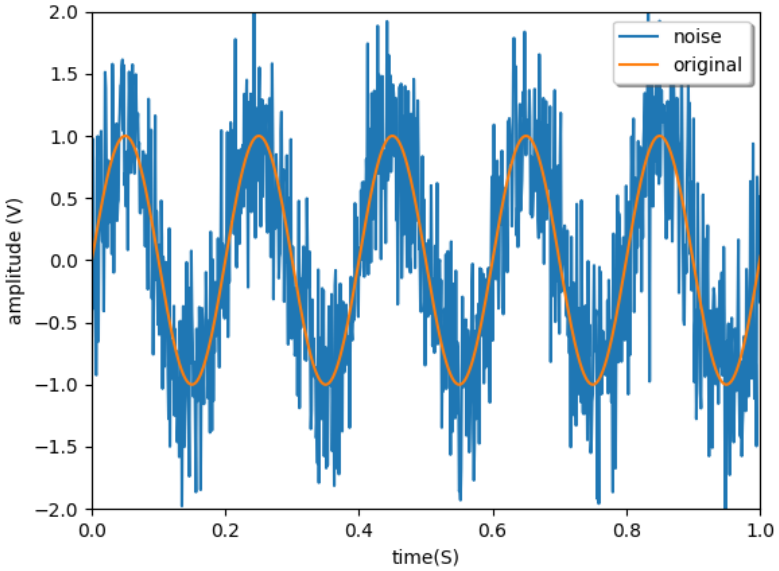

Рисунок 41. шум и сигнал

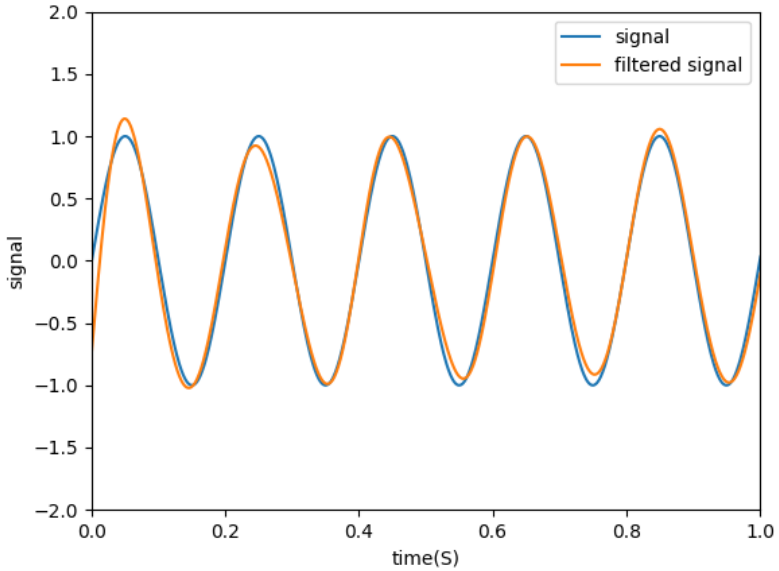

Рисунок 42. результат фильтрации

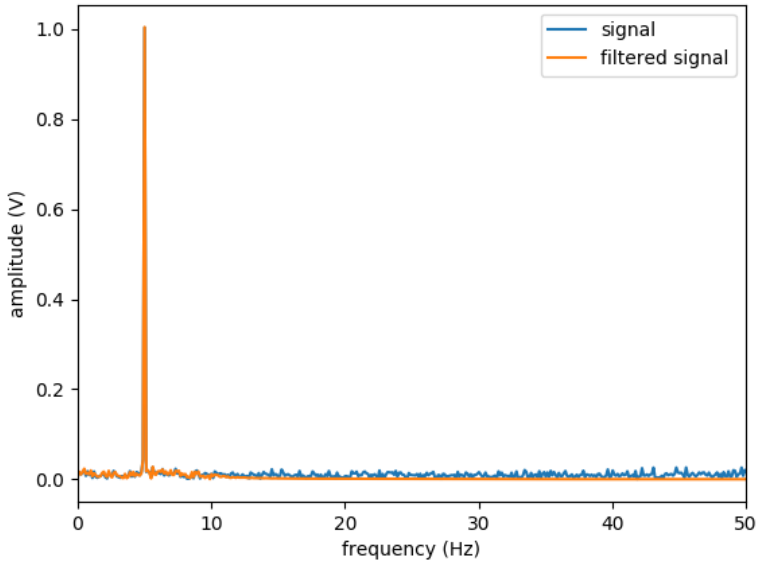

Рисунок 43. спектры шума и отфильтрованного сигнала

5. Выводы

В данной работе была произведенна фильтрация сигнала. Результат фильтрации очень близок к исходному сигналу. На граффике, иллюстрирующем спектры видно, что шумов стало меньше, особенно в области высоких частот.