Федеральное государственное автономное образовательное
учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа №4

ЛИНЕЙНАЯ ФИЛЬТРАЦИЯ

Студенты: Загайнов А.А.
Поток: ЧАСТ.МЕТ. 1.2

Преподаватели: Перегудин. А.А., Пашенко А.В.

Санкт-Петербург
2025

Содержание

Задание 1. Линейные фильтры.

В этом задании мы опять обратимся к функции прямоугольной волны, которую использовали в предыдущей лабораторной работе. Вот ее характеристики:

$$g(t) = \begin{cases} 2, t \in [3,5], \\ 0, t \notin [3,5] \end{cases}$$

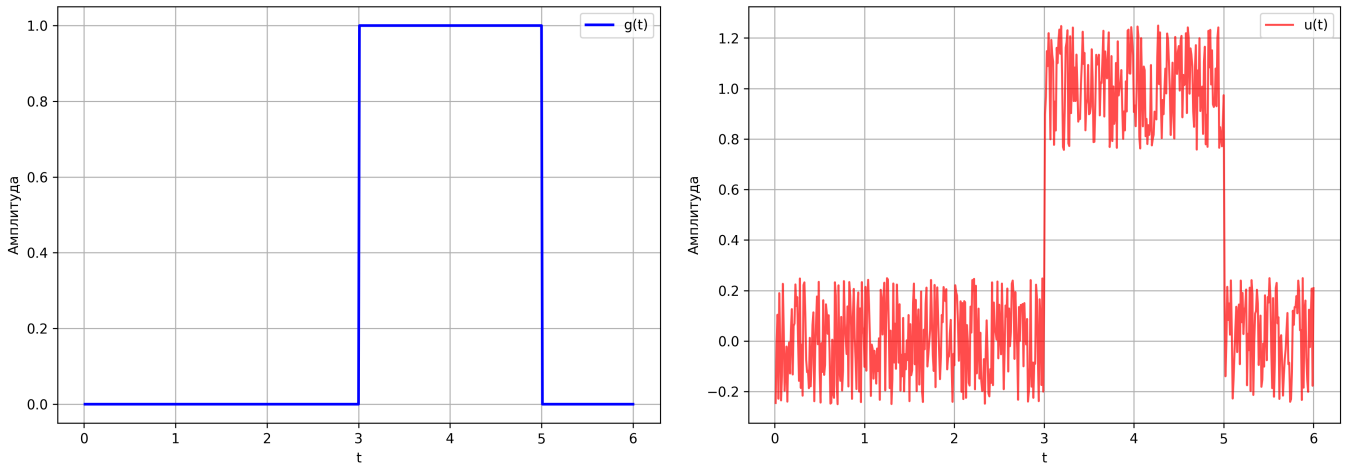

Рис. 1: График g(t) и u(t). $b = 0.5$ $\omega = 0.2$ $= 0.1$

А также вспомним и зашумленную версию этой функции:

$$u(t) = g(t) + b\xi(t) + c\sin(d2\pi t)$$

$\xi(t) \sim U[-1,1]$ - равномерное распределение, а значения b, c, ω - настраиваемые параметры возмущений

В этом задании мы опять будем применять фильтры, только теперь не жесткие, а линейные. Перейдем к первому пункту.

1.1 Фильтр первого порядка.

Как и в прошлый раз - пример для начала коэффициент $c = 0$. Будем пропускать нашу функцию $u(t)$ через линейный фильтр первого порядка:

$$W_1(p) = \frac{1}{Tp + 1}$$

Где $T > 0$ это некоторая постоянная времени.

Будем проверять, как влияет изменение параметра T и a на эффективность фильтрации. Для начала возьмем значения (T= 0.3, a= 1):

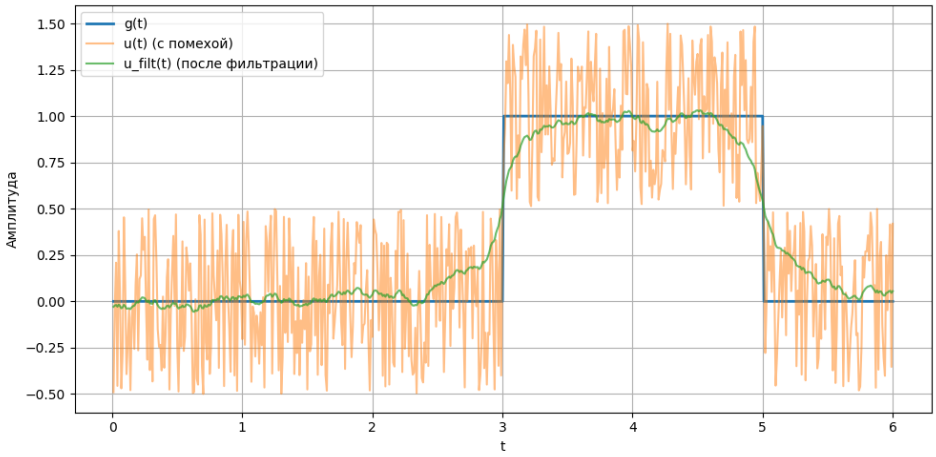

Рис. 2: Все версии графика при T=0.3, a=1

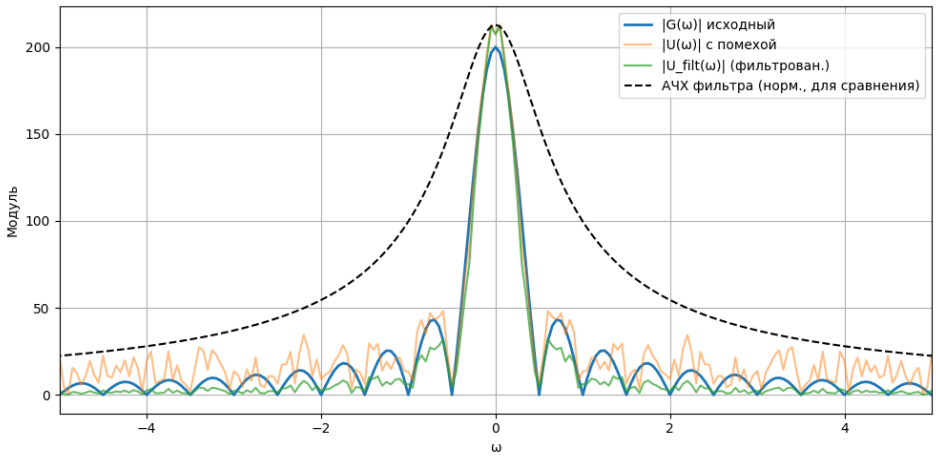

Рис. 3: Все версии Фурье образа при T=0.3, a=1

Можно довольно смело утверждать, что фильтр хорошо справился со своей работой. Гибкий линейный фильтр приглушил в основном именно низкие частоты, что в самый раз для стандартных задач случайным шумом.

Проверим еще один важный вопрос - будут ли различаться графики фильтрованного сигналов и сигнала полученного после обратного преобразования Фурье произведения частотной передаточной функции фильтра и образа зашумленного сигнала $(W_1(i\omega) \cdot \hat{u}(\omega))$?

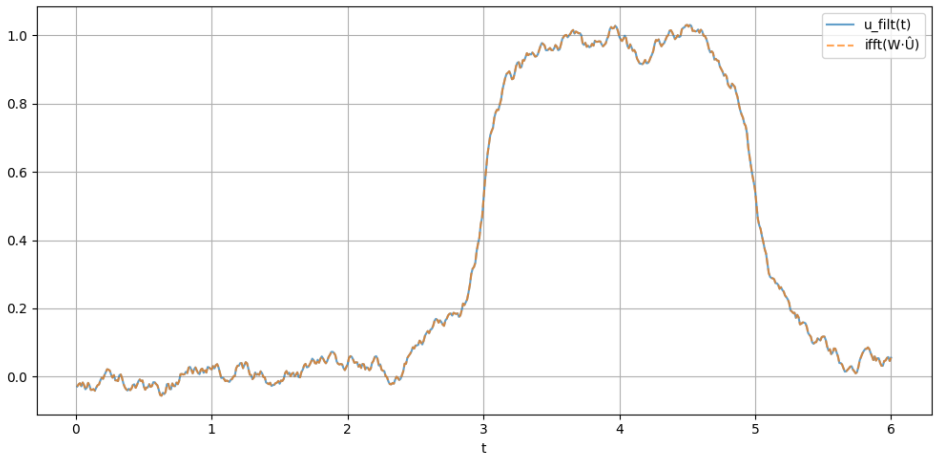

Рис. 4: Сравнение графиков при T=0.3, a=1

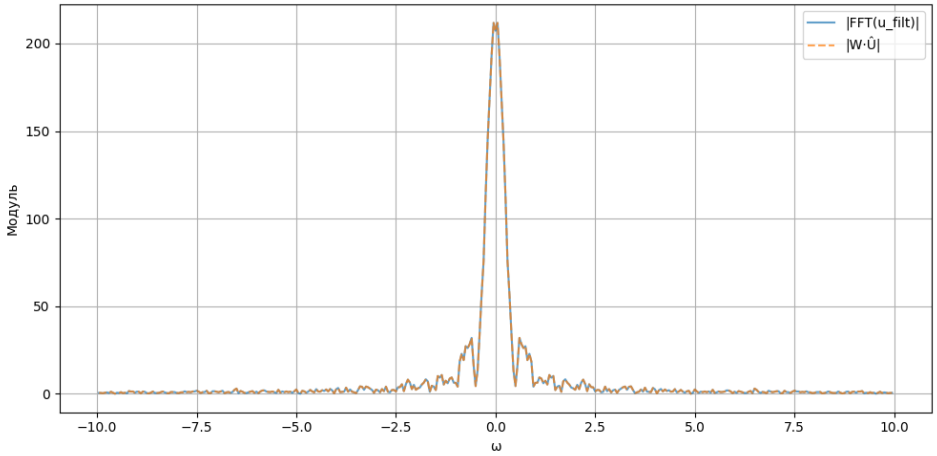

Рис. 5: Сравнение Фурье образов при T=0.3, a=1

Наглядно видно, что результат идентичен. Эти две операции являются одним и тем же.

Посмотрим на результат с параметрами (T=1, a=1):

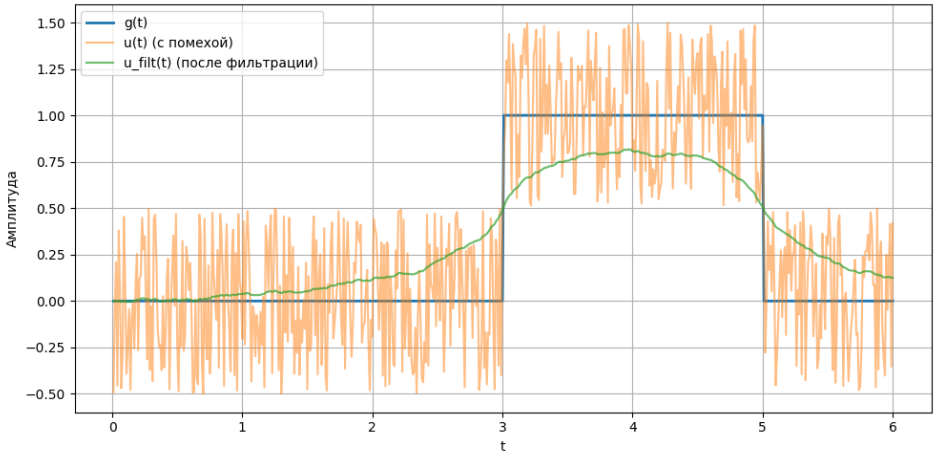

Рис. 6: Все версии графика при T=1, a=1

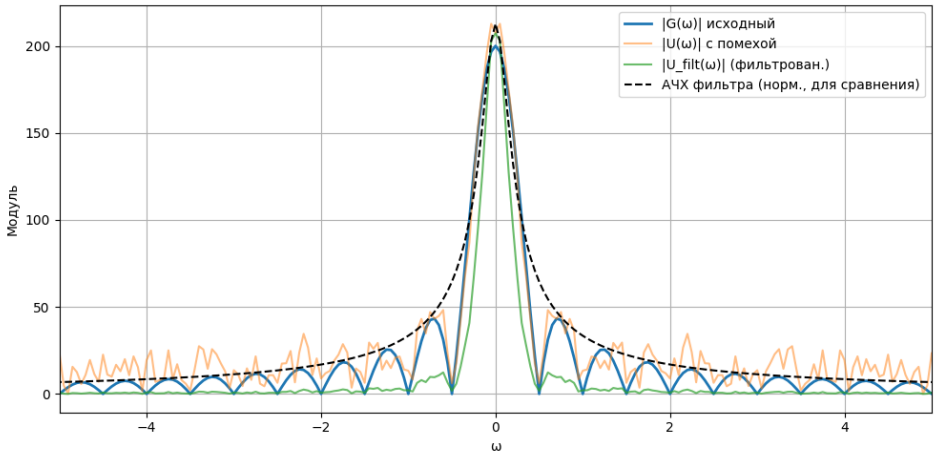

Рис. 7: Все версии Фурье образа при T=1, a=1

Результат стал гораздо хуже. Чем выше мы берем параметр T, тем сильнее график АЧХ фильтра сужается и остреет - соответственно и все более низкий порог для активно подавляемых частот. Значение T=1 явный перебор. Будем использовать значения ниже.

Сравним фильтрованный результат и результат полученный после обратного преобразования Фурье произведения частотной передаточной функции фильтра и образа зашумленного сигнала. Сделаем это в последний раз, чтобы точно удостовериться, что первый результат это не совпадение:

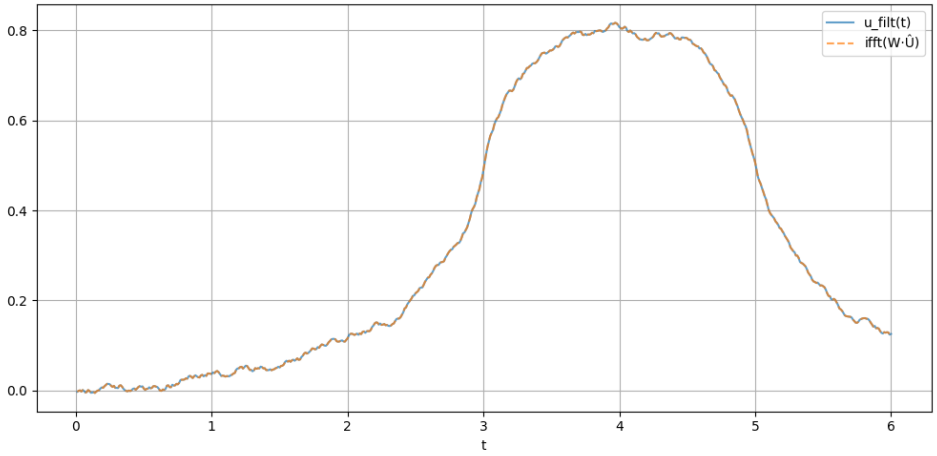

Рис. 8: Сравнение графиков при T=1, a=1

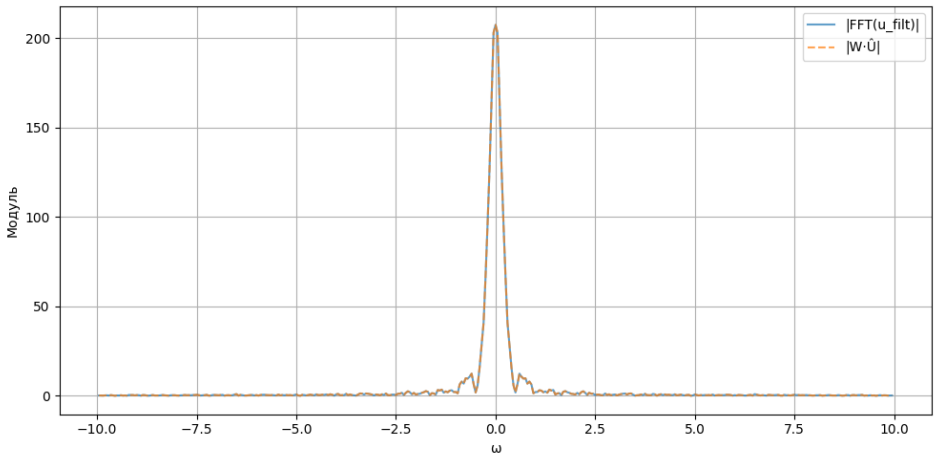

Рис. 9: Сравнение Фурье образов при T=1, a=1

Опять видим, что результаты идентичны. Впредь мы не будем сравнивать их еще - результат и так наглядный.

Обратимся к значениям ниже T=0.3. Посмотрим на результат с параметрами (T=0.1, a=1):

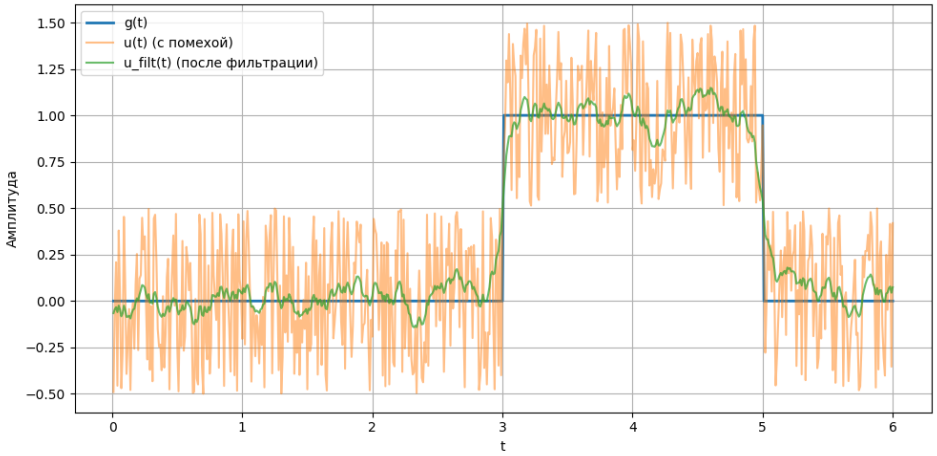

Рис. 10: Все версии графика при T=0.1, a=1

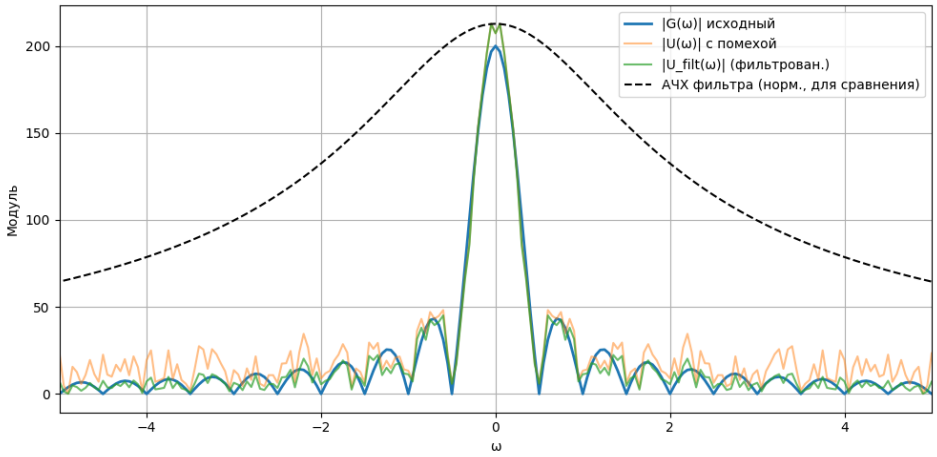

Рис. 11: Все версии Фурье образа при T=0.1, a=1

На удивление, результат стал лучше даже первого варианта - картина очень точная. Становится лучше понятно, почему в данном случае с данной функцией именно линейный фильтр справляется так хорошо. Фурье образ прямоугольной функции представляет собой кариднальный синус - самые важная информация о рисунке функции передана в низких частотах - их фильтр почти не трогает. Самое важное это то, что случайные помехи равномерно сильно влияют на частоты на всем промежутке. Если на низких частотах помехи не так сильно влияют на результат, то на низких они вносят очень большой вклад - именно низкие частоты спадающая АЧХ фильтра и подавляет эффективнее всего. Пропорционально усиливая подавление на высоких частотах отфильтрованный образ хорошо ложится на оригинальную функцию.

Если кратко говорить почему линейный фильтр лучше жесткого фильтра низких частот - жесткий фильтр полностью забирает информацию о высоких частотах, а линейный фильтр, прекрасной подходящий прямоугольной функции, пропорционально выправлет помехи к исходной функции.

Теперь возьмемся за эксперименты с параметром a. Посмотрим на результат с параметрами (T=0.1, a=2):

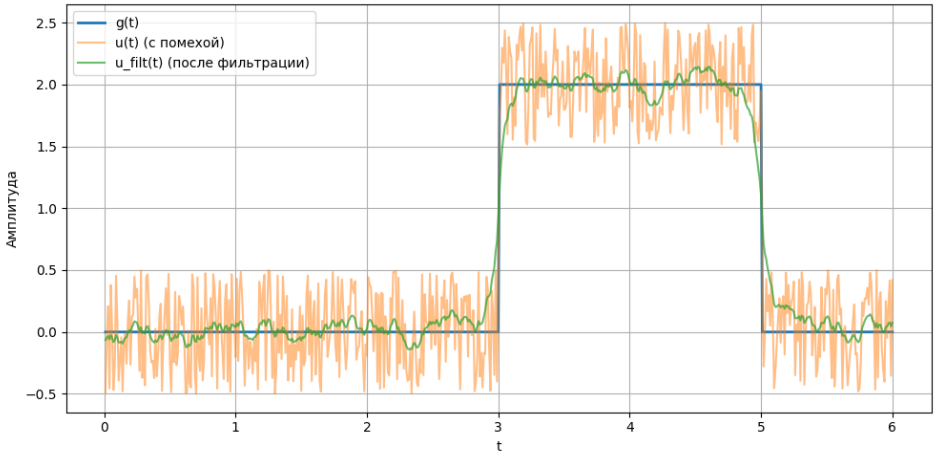

Рис. 12: Все версии графика при T=0.1, a=2

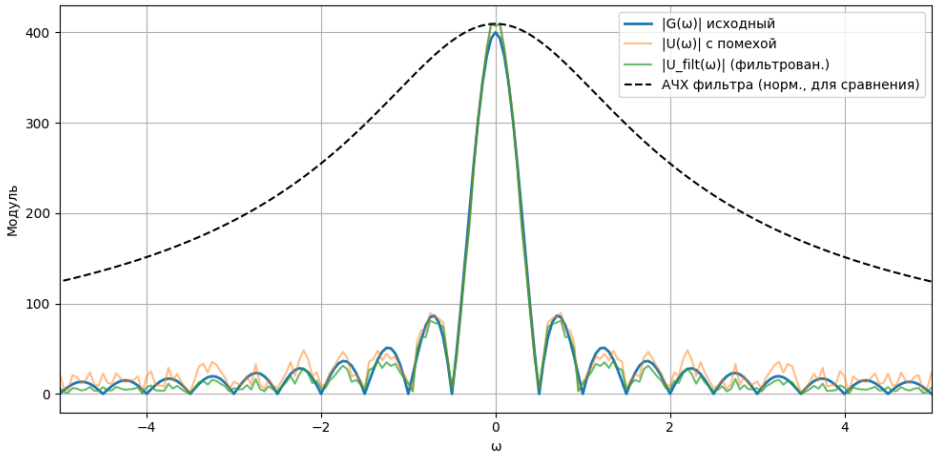

Рис. 13: Все версии Фурье образа при T=0.1, a=2

Сам параметр a отвечает за амплитуду исходной прямоугольной функции. В данном случае изменение параметра a так же влияет на амплитуду Фурье-образа(по свойству Масштабирования). И поэтому чем выше амплитуда Фурье-образа, тем меньший вклад вносит фиксированный по величине случайный шум. Для линейного фильтра нет особой разницы в параметре a. Он уменьшает частоты функции пропорционально поэтому масштабирование функции его не затрагивает.

В данном случае амплитуда была увеличена и, и без того менее значительные, шумы были подчищены и на фоне большого мастшаба стали видны еще меньше.

Но вот если мы возьмем малое значение a... Посмотрим на результат с параметрами (T=0.1, a=0.25):

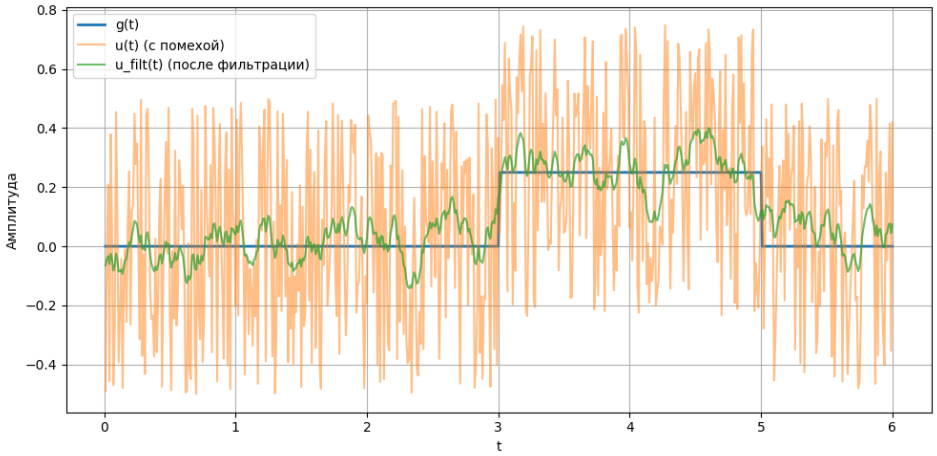

Рис. 14: Все версии графика при T=0.1, a=0.25

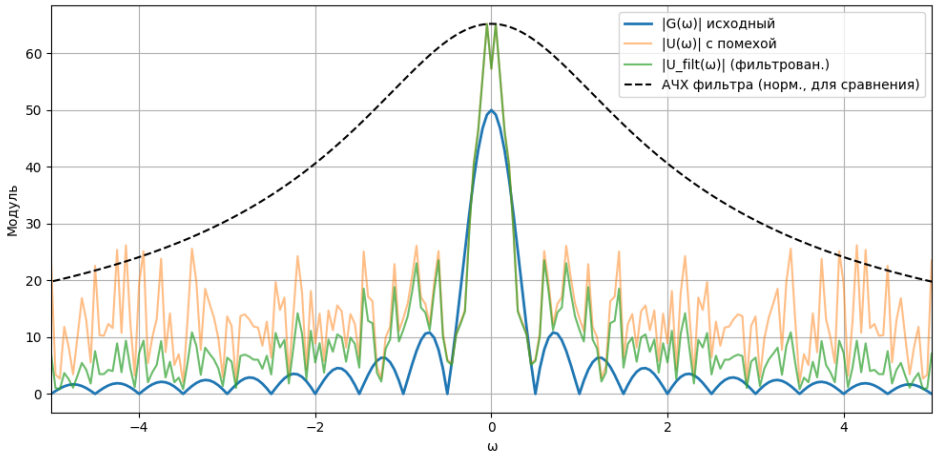

Рис. 15: Все версии Фурье образа при T=0.1, a=0.25

В данном случае изначальная зашумленная функция чрезмерно испорчена, потому даже хороший линейный фильтр сможет вынести отсюда слишком мало полезной информации. Мы можем предположить в каких местах оригинальной функции были подъемы, но это будет чресчур неподходящая информация. «Вины» линейного фильтра в данном случае нет.

Вывод.

Линейный фильтр хорошо справляется с фильтрацией случайных шумов. Параметр T отвечает за интенсивность работы фильтра, и на практике при несильных умеренных шумах нужны его небольшие значения - этого будет достаточно. Параметр a влияет на амплитуду функции и его Фурье-образа. Напрямую не влияет на работу фильтра, ведь он подавляет частоты пропорционально и коэффициент масштабирования попросту нивелируется.

1.2 Режекторный полосовой фильтр

Примем $b = 0, c = 0.5, d = 5$. Теперь мы рассматриваем линейный фильтр вида:

$$W_2(p) = \frac{p^2 + a_1 p + a_2}{p^2 + b_1 p + b_2}$$

Выберем для него такие параметры $a_1, a_2, b_1, b_2 \in R$, такие что:

- Фильтр устойчивый (b_1, b_2 будут больше нуля)
- Фильтр имеет единичное усиление на 0 и ∞
- Фильтр имеет нулевое усиление на частоте помехи $\omega_0 = d = 5$гц

Исходя из наших требований можем посчитать требуемые значения параметров:

- Из условия нулевого усиления на ω_0: $a_1 = 0$ и $a_2 = \omega_0^2 = (10\pi)^2 = 100\pi^2$.
- Из условия единичного усиления на $\omega = 0$: $b_2 = a_2 = 100\pi^2$.
- Из условия устойчивости: $b_1 > 0$ (значение b_1 будем менять) и $b_2 = 100\pi^2 > 0$

Изучим АЧХ фильтра, и оценим, как именно изменение b_1 влияет на работу фильтра.

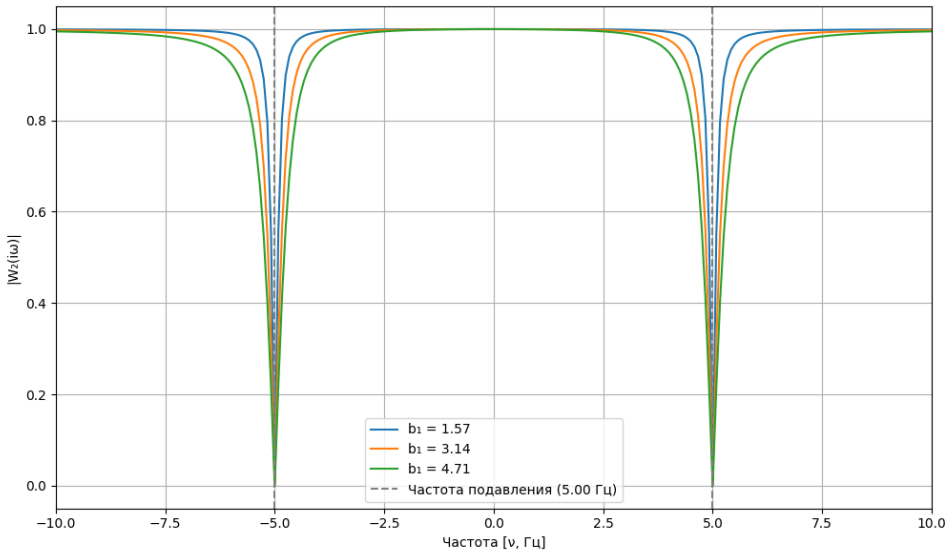

Рис. 16: Сравнение разных b_1

Результат на лицо - b_1 влияет именно на ширину области подавления фильтра. Примем это к сведению.

Попробуем теперь изучить, как будет изменяться результат работы фильтра, если смещать параметр b_1.

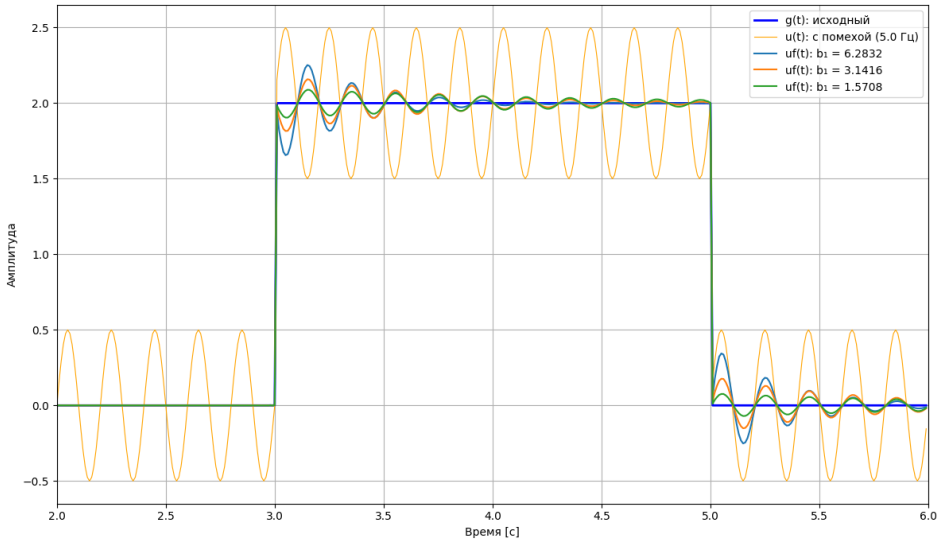

Рис. 17: Все версии графика

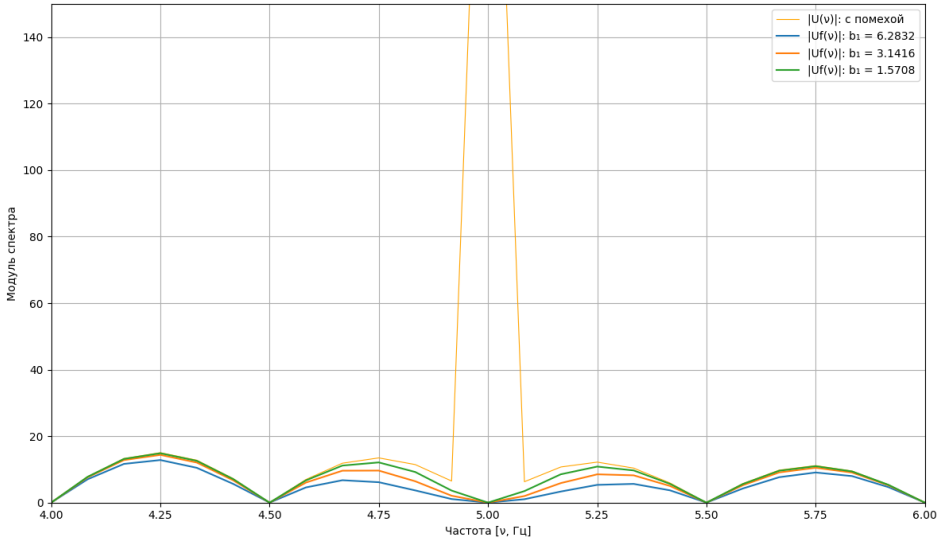

Рис. 18: Все версии Фурье образа

Видно - чем выше был выбран параметр b_1 и чем сильнее работал фильтр, тем более искаженным выходил итоговый вариант. Обьясняется это тем, что гармоническое зашумление довольно узкое в плане частотного спектра и ему не требуется широкого фильтра. Параметры b_1 равные π и 2π из за своей ширины задевали «здоровые» участки спектра и потому результат выходил довольно шумный, хоть и гораздо более лучший, чем изначально зашумленный.

Теперь будем смотреть, как будет работать фильтр, заточенный на подавление чатсоты $\omega_0 05$гц, с сигналами у которых гармонический шум на других частотах:

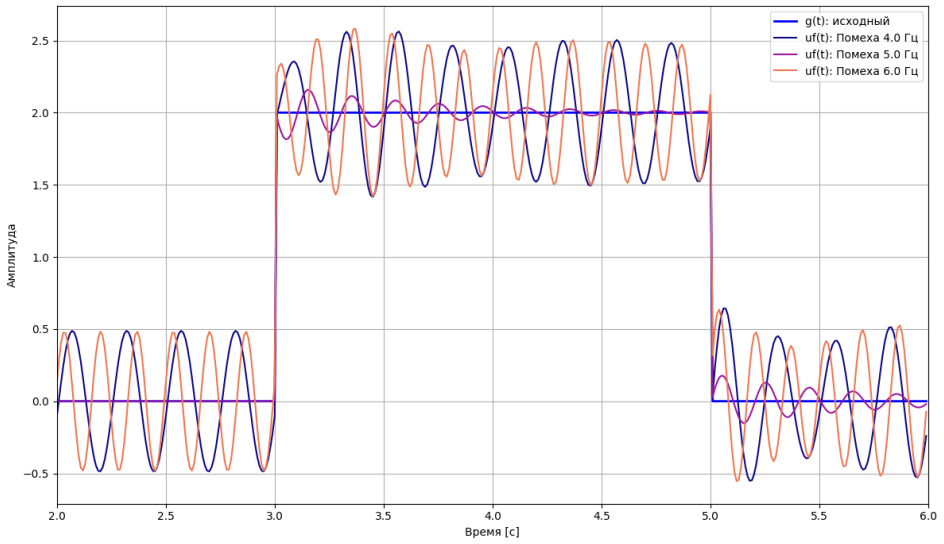

Рис. 19: Все версии графика

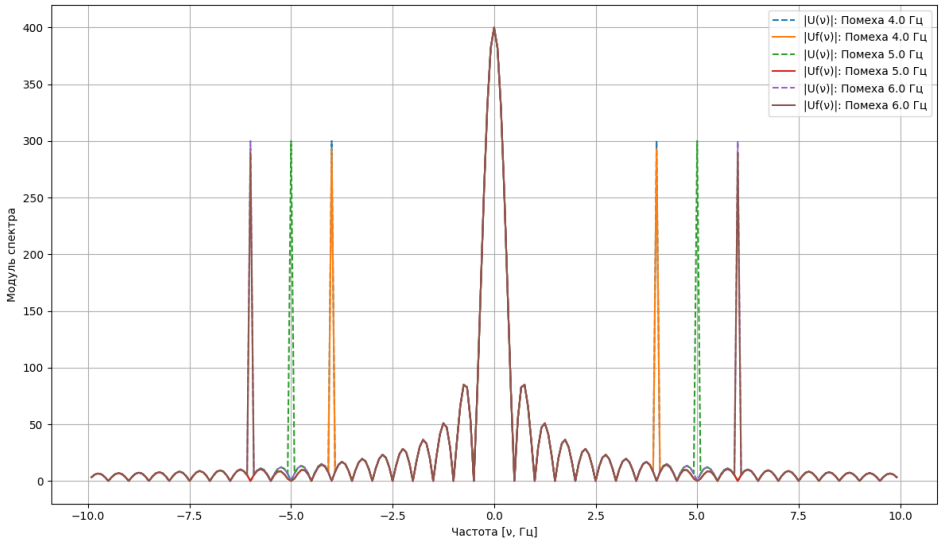

Рис. 20: Все версии Фурье образа

Результат ожидаемый - фильтр почти не повлиял на сигналы с помехой 4 и 6 герц. На графике это не видно, но на самом деле волны от колебаний после фильтрации сдвинулись немного в сторону от изначального положения, это обьясняется тем, что фильтр все же внес определенную лепту и погасил небольшой кусочек незатронутого шумом спектра. Еще стоит отметить насколько схожими являются графики от 4 и 6 герц - почти идентичные, только сдвинутые относительно друг друга. Наглядная иллюстрация природы гармонических помех.

Посмотрим так же и влияние влияние параметра c на эффективность фильтрации:

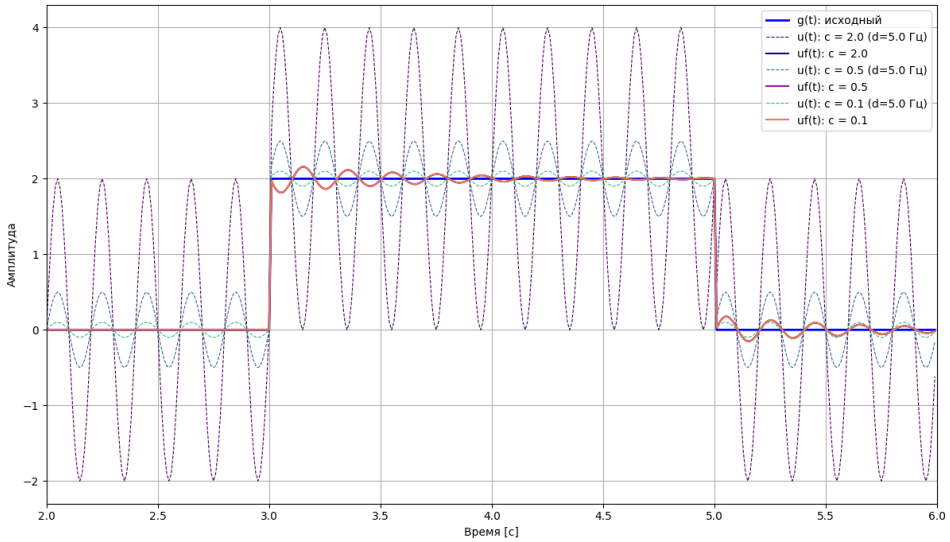

Рис. 21: Все версии графика

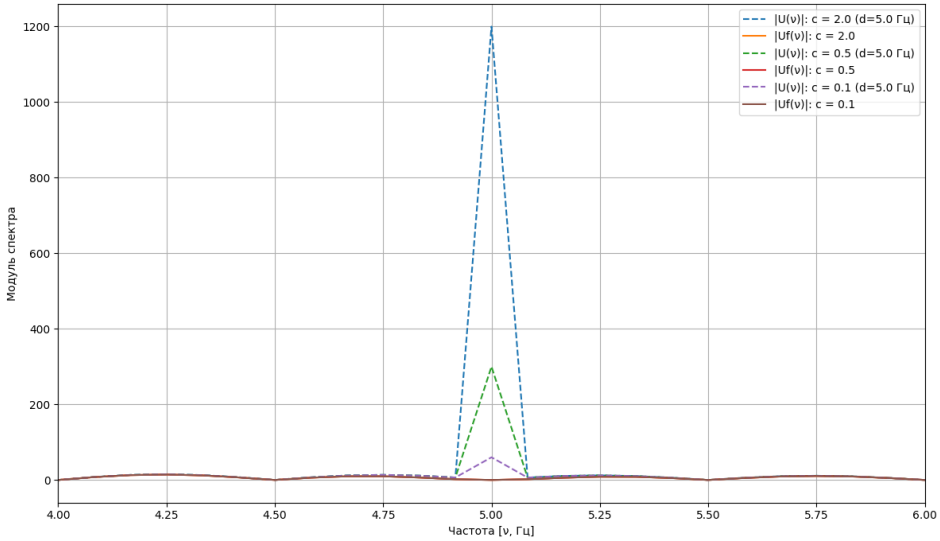

Рис. 22: Все версии Фурье образа

Результат очевиден еще по предыдущей лабораторной работе - амплитуда гармонических колебаний не влияет никаким образом на эффективность фильтрации. Все графики оказались идентичными. Разве что любопытно отметить, что остаточные помехи после фильтрации в пике оказались выше, чем гармонические колебания с наименьшим коэффициентом c.

Напоследок в этом задании еще раз убедимся, что между фильтрованным сигналом и произведением $W_2(i\omega) \cdot \hat{u}(\omega)$ нет разницы:

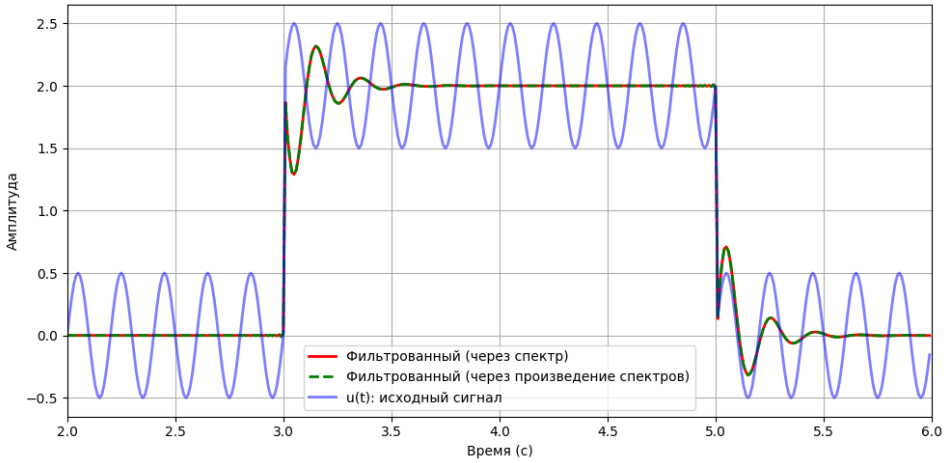

Рис. 23: Временные графики

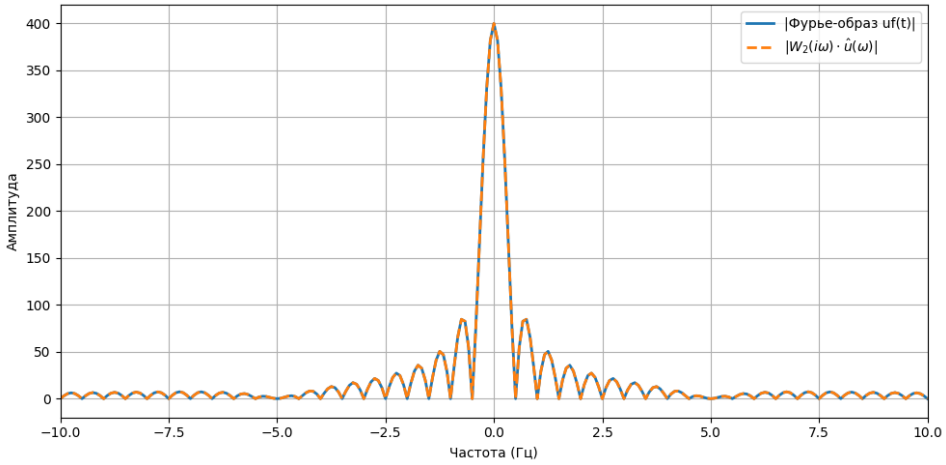

Рис. 24: Фурье образы

Вуаля, результат и правда один и тот же. Теперь можно точно быть в этом уверенным.

Вывод:

При работе с режекторным фильтром по очисте гармонических шумов прежде всего стоит обратить внимание на подбор подходящих параметров фильтра, чтобы нулевое усиление достигалось именно в том месте, где находится шум гармонического колебания в частотном спектре. Дальше необходим обратить внимание на коэффициент b_1, чтобы ширина подавления фильтра была подходящего размера. Амплитуда гармонических колебаний в данном случае не влияет ни на что.

2. Сглаживание биржевых данных

В данном задании мы будем сглаживать котировки акций, применяя разный временной период для линейного фильтра сглаживания. В качестве данных возьмем котировки Сбербанка за период с начала 2022 по апрель 2025. Временные периоды следующие: 1 день, 1 неделя, 1 месяц, 3 месяца, 1 год. В коде я отдельно поправил, чтобы результат фильтрации начинался со стартовой точки изначалных данных.

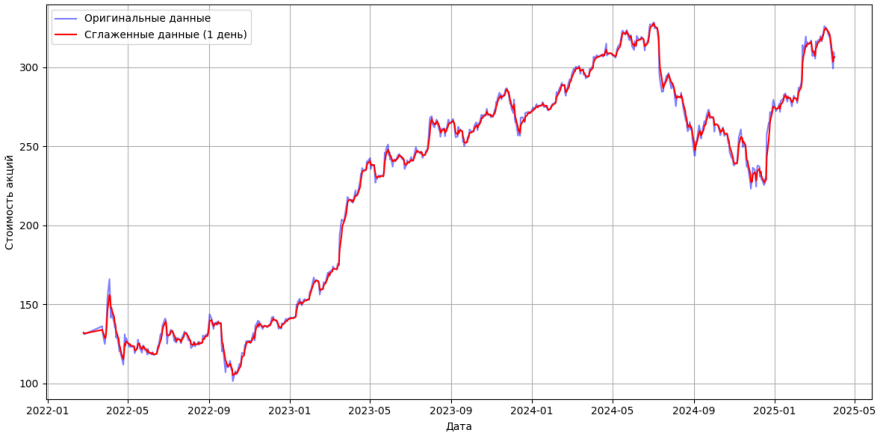

Рис. 25: Временной параметр 1 день

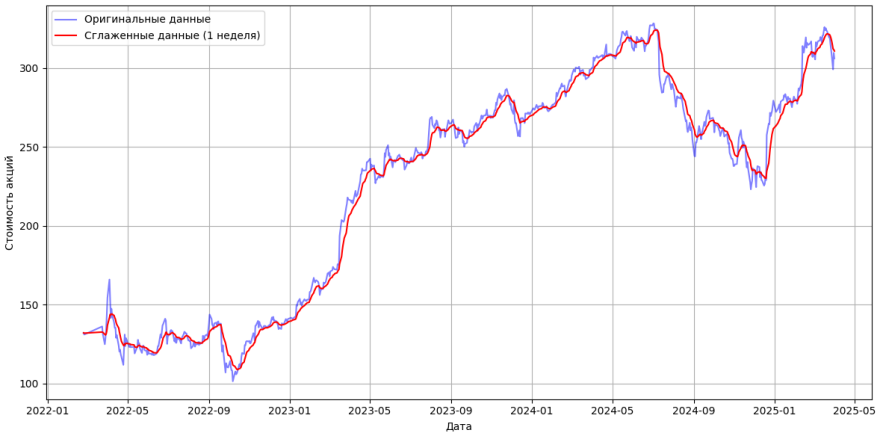

Рис. 26: Временной параметр 1 неделя

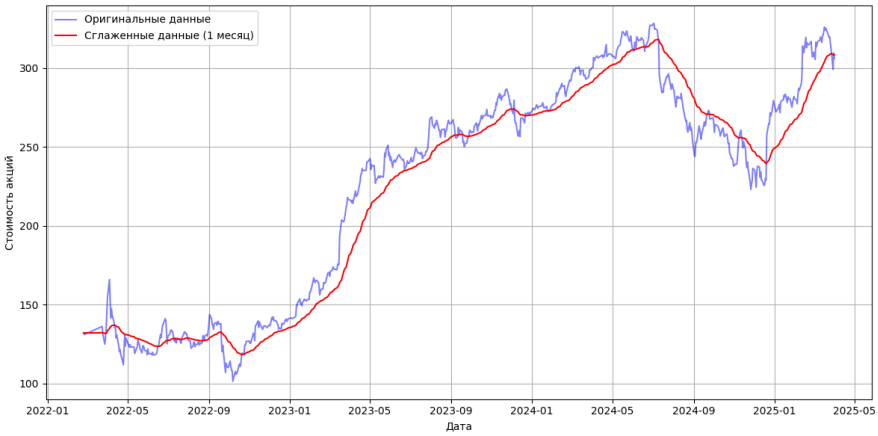

Рис. 27: Временной параметр 1 месяц

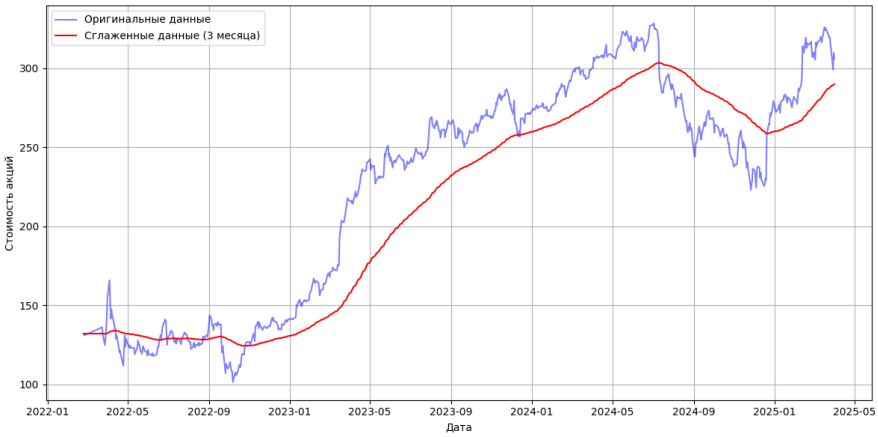

Рис. 28: Временной параметр 3 месяца

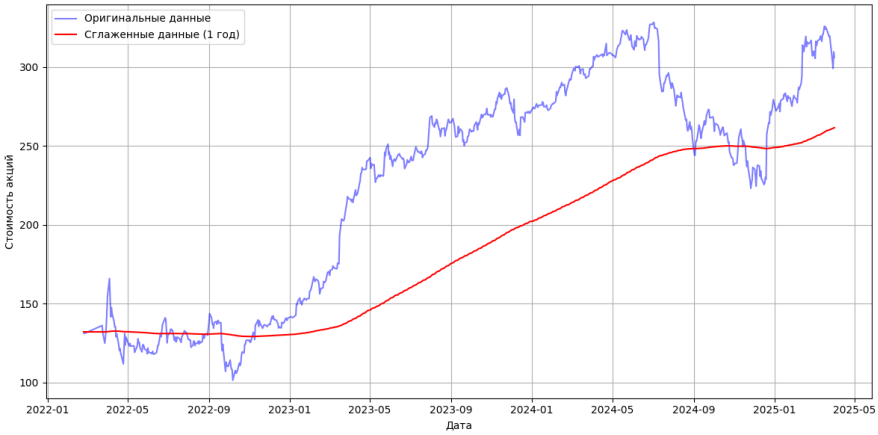

Рис. 29: Временной параметр 1 год

Первые две самые малые настройки времени выглядят очень схоже. И оба дают крайне хороший результат. На промежутке в неделю уже начинают проглядываться более заметные сглаживания. Чем больший временной промежуток берем, тем более простым становится график. T =1 год уже совсем символичную картину, из нее разве что можно выщепить общую тенденцию за весь промежуток времени.

Вывод:

Линейный фильтр прекрасно подходит для очень многих задач в нашей жизни, когда дело доходит до сглаживания каких либо сигналов. В зависимости от временного параметра T можно настраивать его точность.

Приложение

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


t = np.arange(-10, 10, 0.01)
N = len(t)
dt = t[1] - t[0]
t_1, t_2 = 3, 5
b = 0.5
c = 0
d = 5 * (2 * np.pi)
t_min = 0
t_max = 6

a_list = [0.25, 2]
T_list = [0.1, 1.0]
v = np.fft.fftshift(np.fft.fftfreq(N, d=dt))

freq_min = -10
freq_max = 10

for a in a_list:
    g = np.zeros_like(t)
    g[(t >= t_1) & (t <= t_2)] = a

    for T in T_list:
        xi = np.random.uniform(-1, 1, size=t.shape)
        u = g + b * xi + c * np.sin(d * t)
        W_abs = 1 / np.sqrt(1 + (T * 2 * np.pi * v) ** 2)
        U_f = np.fft.fftshift(np.fft.fft(u))
        U_filt_f = U_f * W_abs
        u_filt = np.fft.ifft(np.fft.ifftshift(U_filt_f)).real
        G_f = np.fft.fftshift(np.fft.fft(g))

        plt.figure(figsize=(10,5))
        trunc_mask = (t >= t_min) & (t <= t_max)
        plt.plot(t[trunc_mask], g[trunc_mask], label='g(t)', linewidth=2)
        plt.plot(t[trunc_mask], u[trunc_mask], label='u(t) (с помехой)', alpha=0.5)
        plt.plot(t[trunc_mask], u_filt[trunc_mask], label='u_filt(t) (после фильтрации)', alpha=0.7)
        plt.xlabel('t')
        plt.ylabel('Амплитуда')
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.savefig(f'1_1/signals_time_a{a}_T{T}.png')
        plt.show()
        plt.close()

        plt.figure(figsize=(10,5))
        freq_mask = (v >= freq_min) & (v <= freq_max)
        plt.plot(v[freq_mask], np.abs(G_f)[freq_mask], label='|G(  )| исходный', linewidth=2)
        plt.plot(v[freq_mask], np.abs(U_f)[freq_mask], label='|U(  )| с помехой', alpha=0.5)
        plt.plot(v[freq_mask], np.abs(U_filt_f)[freq_mask], label='|U_filt(  )| (фильтрован.)',
    alpha=0.7)
        plt.plot(v[freq_mask], W_abs[freq_mask] * np.max(np.abs(U_f)[freq_mask]), label='АЧХ фильтра
     (норм., для сравнения)', linestyle='--', color='k')
        plt.xlabel('  ')
        plt.ylabel('Модуль')
        plt.xlim([-5, 5])
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.savefig(f'1_1/spectra_a{a}_T{T}.png')
        plt.show()
        plt.close()
```

Листинг 1: Код задания 1.1 для отрисовки и настройки параметров

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

output_dir = 'filter_1_2'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

t = np.arange(-6, 6, 0.01)
N = len(t)
dt = t[1] - t[0]

t_1, t_2 = 3, 5
a = 2
g = np.zeros_like(t)
g[(t >= t_1) & (t <= t_2)] = a

b = 0

v = np.fft.fftshift(np.fft.fftfreq(N, d=dt))
omega_fft = 2 * np.pi * v

a1_fixed = 0

d_fixed_hz_for_b1_analysis = 5.0
omega0_fixed_for_b1_analysis = d_fixed_hz_for_b1_analysis * 2 * np.pi
c_fixed_for_b1_analysis = 0.5

a2_fixed_for_b1_analysis = omega0_fixed_for_b1_analysis**2
b2_fixed_for_b1_analysis = omega0_fixed_for_b1_analysis**2

u_fixed_d_c = g + b * np.zeros_like(t) + c_fixed_for_b1_analysis * np.sin(d_fixed_hz_for_b1_analysis
     * 2 * np.pi * t)

k_values_for_b1 = [0.2, 0.1, 0.05]
b1_values_to_test = [k * omega0_fixed_for_b1_analysis for k in k_values_for_b1]

filtered_signals_for_b1_analysis = []
noisy_spectrums_for_b1_analysis = []
filtered_spectrums_for_b1_analysis = []
labels_for_b1_analysis = []

for i, b1 in enumerate(b1_values_to_test):
    if b1 <= 0:
        continue


    numerator = -(omega_fft)**2 + a1_fixed * (1j * omega_fft) + a2_fixed_for_b1_analysis
    denominator = -(omega_fft)**2 + b1 * (1j * omega_fft) + b2_fixed_for_b1_analysis

    W2 = np.divide(numerator, denominator, out=np.zeros_like(numerator), where=denominator!=0)

    U_fixed_d_c_fft = np.fft.fftshift(np.fft.fft(u_fixed_d_c))
    U_filtered = U_fixed_d_c_fft * W2
    u_filtered = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered)))

    filtered_signals_for_b1_analysis.append(u_filtered)
    if i == 0:
         noisy_spectrums_for_b1_analysis.append(U_fixed_d_c_fft)
    filtered_spectrums_for_b1_analysis.append(U_filtered)
    labels_for_b1_analysis.append(f" b   = {b1:.4f}")
```

Листинг 2: Код задания 1.2 для отрисовки графиков и перебора параметров (часть 1)

```

plt.figure(figsize=(12, 7))
plt.plot(t, g, label='g(t): исходный', color='blue', linewidth=2)
plt.plot(t, u_fixed_d_c, label=f'u(t): с помехой ({d_fixed_hz_for_b1_analysis:.1f} Гц, c={
    c_fixed_for_b1_analysis:.1f})', color='orange', linewidth=0.8)
for filtered_signal, label in zip(filtered_signals_for_b1_analysis, labels_for_b1_analysis):
    plt.plot(t, filtered_signal, label=f'uf(t): {label}', linewidth=1.5)

plt.xlabel('Время [c]')
plt.ylabel('Амплитуда')
plt.legend()
plt.grid()
plt.tight_layout()
plt.xlim([2,6])
plt.savefig(os.path.join(output_dir, 'comparative_time_domain_b1_variation.png'))
plt.show()


plt.figure(figsize=(12, 7))
lim = 10
freq_mask = (v >= -lim) & (v <= lim)

if noisy_spectrums_for_b1_analysis:
    plt.plot(v[freq_mask], np.abs(noisy_spectrums_for_b1_analysis[0][freq_mask]), label='|U(  )|: c
    помехой', color="orange", linewidth=0.8)

for filtered_spectrum, label in zip(filtered_spectrums_for_b1_analysis, labels_for_b1_analysis):
    plt.plot(v[freq_mask], np.abs(filtered_spectrum[freq_mask]), label=f'|Uf(  )|: {label}',
    linewidth=1.5)

plt.xlabel('Частота [  , Гц]')
plt.ylabel('Модуль спектра')
plt.legend()
plt.grid()
plt.tight_layout()
plt.ylim([0, 150])
plt.xlim([4,6])
plt.savefig(os.path.join(output_dir, 'comparative_frequency_spectrum_b1_variation.png'))
plt.show()
d_filter_design_hz = 5.0
omega0_filter_design = d_filter_design_hz * 2 * np.pi
b1_fixed_for_d_analysis = b1_values_to_test[len(b1_values_to_test) // 2]
c_fixed_for_d_analysis = 0.5

numerator_fixed_filter = -(omega_fft)**2 + a1_fixed * (1j * omega_fft) + omega0_filter_design**2
denominator_fixed_filter = -(omega_fft)**2 + b1_fixed_for_d_analysis * (1j * omega_fft) +
    omega0_filter_design**2
W2_fixed_filter = np.divide(numerator_fixed_filter, denominator_fixed_filter,
                            out=np.zeros_like(numerator_fixed_filter), where=denominator_fixed_filter
    !=0)

d_values_for_fixed_filter_test_hz = [4.0, 5.0, 6.0]

noisy_signals_for_d_analysis = []
filtered_signals_for_d_analysis = []
noisy_spectrums_for_d_analysis = []
filtered_spectrums_for_d_analysis = []
labels_for_d_analysis = []

for d_hz_test in d_values_for_fixed_filter_test_hz:
    u_test_varying_d = g + b * np.zeros_like(t) + c_fixed_for_d_analysis * np.sin(d_hz_test * 2 * np
    .pi * t)

    U_test_varying_d_fft = np.fft.fftshift(np.fft.fft(u_test_varying_d))
    U_filtered_test = U_test_varying_d_fft * W2_fixed_filter
    u_filtered_test = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered_test)))

    noisy_signals_for_d_analysis.append(u_test_varying_d)
    filtered_signals_for_d_analysis.append(u_filtered_test)
    noisy_spectrums_for_d_analysis.append(U_test_varying_d_fft)
    filtered_spectrums_for_d_analysis.append(U_filtered_test)
    labels_for_d_analysis.append(f"Помеха {d_hz_test:.1f} Гц")
```

Листинг 3: Код задания 1.2 для отрисовки графиков и перебора параметров (часть 2)

```
1
2  plt.figure(figsize=(12, 7))
3  plt.plot(t, g, label='g(t): исходный', color='blue', linewidth=2)
4  for i, (noisy_signal, filtered_signal, label) in enumerate(zip(noisy_signals_for_d_analysis,
       filtered_signals_for_d_analysis, labels_for_d_analysis)):
5       color_u = plt.cm.viridis(i / len(d_values_for_fixed_filter_test_hz))
6       color_uf = plt.cm.plasma(i / len(d_values_for_fixed_filter_test_hz))
7
8       plt.plot(t, noisy_signal, label=f'u(t): {label} (c={c_fixed_for_d_analysis:.1f})', color=
       color_u, linewidth=0.8, linestyle='--')
9       plt.plot(t, filtered_signal, label=f'uf(t): {label}', color=color_uf, linewidth=1.5)
10
11 plt.xlabel('Время [c]')
12 plt.ylabel('Амплитуда')
13 plt.legend()
14 plt.grid()
15 plt.tight_layout()
16 plt.xlim([2,6])
17 plt.savefig(os.path.join(output_dir, 'comparative_time_domain_d_variation.png'))
18 plt.show()
19
20 plt.figure(figsize=(12, 7))
21 lim = 10
22 freq_mask = (v >= -lim) & (v <= lim)
23
24 for i, (noisy_spectrum, filtered_spectrum, label) in enumerate(zip(noisy_spectrums_for_d_analysis,
       filtered_spectrums_for_d_analysis, labels_for_d_analysis)):
25     plt.plot(v[freq_mask], np.abs(noisy_spectrum[freq_mask]), label=f'|U(  )|: {label} (c={
       c_fixed_for_d_analysis:.1f})', linestyle='--')
26     plt.plot(v[freq_mask], np.abs(filtered_spectrum[freq_mask]), label=f'|Uf(  )|: {label}')
27
28 plt.xlabel('Частота [  , Гц]')
29 plt.ylabel('Модуль спектра')
30 plt.legend()
31 plt.grid()
32 plt.tight_layout()
33 plt.xlim([-10, 10])
34 plt.savefig(os.path.join(output_dir, 'comparative_frequency_spectrum_d_variation.png'))
35 plt.show()
36
37 d_filter_design_hz_for_c_analysis = d_filter_design_hz
38 omega0_filter_design_for_c_analysis = omega0_filter_design
39 b1_fixed_for_c_analysis = b1_fixed_for_d_analysis
40 d_fixed_hz_for_c_analysis = d_filter_design_hz_for_c_analysis
41
42 numerator_fixed_filter_c = -(omega_fft)**2 + a1_fixed * (1j * omega_fft) +
       omega0_filter_design_for_c_analysis**2
43 denominator_fixed_filter_c = -(omega_fft)**2 + b1_fixed_for_c_analysis * (1j * omega_fft) +
       omega0_filter_design_for_c_analysis**2
44 W2_fixed_filter_for_c_analysis = np.divide(numerator_fixed_filter_c, denominator_fixed_filter_c,
45                             out=np.zeros_like(numerator_fixed_filter_c), where=
       denominator_fixed_filter_c!=0)
46
47 c_values_to_test = [2, 0.5, 0.1]
48
49 noisy_signals_for_c_analysis = []
50 filtered_signals_for_c_analysis = []
51 noisy_spectrums_for_c_analysis = []
52 filtered_spectrums_for_c_analysis = []
53 labels_for_c_analysis = []
54 for c_test in c_values_to_test:
55     u_test_varying_c = g + b * np.zeros_like(t) + c_test * np.sin(d_fixed_hz_for_c_analysis * 2 * np
       .pi * t)
56     U_test_varying_c_fft = np.fft.fftshift(np.fft.fft(u_test_varying_c))
57     U_filtered_test_c = U_test_varying_c_fft * W2_fixed_filter_for_c_analysis
58     u_filtered_test_c = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered_test_c)))
59
60
61     noisy_signals_for_c_analysis.append(u_test_varying_c)
62     filtered_signals_for_c_analysis.append(u_filtered_test_c)
63     noisy_spectrums_for_c_analysis.append(U_test_varying_c_fft)
64     filtered_spectrums_for_c_analysis.append(U_filtered_test_c)
65     labels_for_c_analysis.append(f"c = {c_test:.1f}")
```

Листинг 4: Код задания 1.2 для отрисовки графиков и перебора параметров (часть 3)

```
1
2  plt.figure(figsize=(12, 7))
3  plt.plot(t, g, label='g(t): исходный', color='blue', linewidth=2)
4  for i, (noisy_signal, filtered_signal, label) in enumerate(zip(noisy_signals_for_c_analysis,
       filtered_signals_for_c_analysis, labels_for_c_analysis)):
5       color_u = plt.cm.viridis(i / len(c_values_to_test))
6       color_uf = plt.cm.plasma(i / len(c_values_to_test))
7
8       plt.plot(t, noisy_signal, label=f'u(t): {label} (d={d_fixed_hz_for_c_analysis:.1f} Гц)', color=
       color_u, linewidth=0.8, linestyle='--')
9       plt.plot(t, filtered_signal, label=f'uf(t): {label}', color=color_uf, linewidth=1.5)
10
11 plt.xlabel('Время [c]')
12 plt.ylabel('Амплитуда')
13 plt.legend()
14 plt.grid()
15 plt.tight_layout()
16 plt.xlim([2,6])
17 plt.savefig(os.path.join(output_dir, 'comparative_time_domain_c_variation.png'))
18 plt.show()
19
20
21 plt.figure(figsize=(12, 7))
22 lim = 10
23 freq_mask = (v >= -lim) & (v <= lim)
24
25 for i, (noisy_spectrum, filtered_spectrum, label) in enumerate(zip(noisy_spectrums_for_c_analysis,
       filtered_spectrums_for_c_analysis, labels_for_c_analysis)):
26     plt.plot(v[freq_mask], np.abs(noisy_spectrum[freq_mask]), label=f'|U(  )|: {label} (d={
       d_fixed_hz_for_c_analysis:.1f} Гц)', linestyle='--')
27     plt.plot(v[freq_mask], np.abs(filtered_spectrum[freq_mask]), label=f'|Uf(  )|: {label}')
28
29 plt.xlabel('Частота [  , Гц]')
30 plt.ylabel('Модуль спектра')
31 plt.legend()
32 plt.xlim([4,6])
33 plt.grid()
34 plt.tight_layout()
35
36 plt.savefig(os.path.join(output_dir, 'comparative_frequency_spectrum_c_variation.png'))
37 plt.show()
```

Листинг 5: Код задания 1.2 для отрисовки графиков и перебора параметров (часть 4)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import lfilter, lfilter_zi


file_path = "SBER_220224_250401.csv"
data = pd.read_csv(
    file_path,
    sep=";",
    parse_dates=[['DATE', 'TIME']],
    dayfirst=True,
    names=["TICKER", "PER", "DATE", "TIME", "OPEN", "HIGH", "LOW", "CLOSE", "VOL"],  # Названия столбцов
    header=0
)

data.set_index('DATE_TIME', inplace=True)
prices = data['CLOSE'].astype(float)

T_values = {
    "1 день": 1,
    "1 неделя": 5,
    "1 месяц": 21,
    "3 месяца": 63,
    "1 год": 252
}

def first_order_filter_with_correct_initial(signal, T, dt=1):
    alpha = dt / (T + dt)
    b = [alpha]
    a = [1, alpha - 1]

    zi = (1 - alpha) * signal[0]

    smoothed, _ = lfilter(b, a, signal, zi=[zi])
    return smoothed

for label, T in T_values.items():
    smoothed = first_order_filter_with_correct_initial(prices.values, T)
    plt.figure(figsize=(12, 6))
    plt.plot(prices.index, prices, label='Оригинальные данные', color='blue', alpha=0.5)
    plt.plot(prices.index, smoothed, label=f'Сглаженные данные ({label})', color='red', linewidth=1.5)

    plt.xlabel("96ата")
    plt.ylabel("Стоимость акций")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.show()
```

Листинг 6: Код задания 2