Федеральное государственное автономное образовательное учреждение высшего образования «Национальный Исследовательский Университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №3
ПРЕДМЕТ «ЧАСТОТНЫЕ МЕТОДЫ»
ТЕМА «ЖЕСТКАЯ ФИЛЬТРАЦИЯ»

Лектор: Перегудин А. А.
Практик: Пашенко А. В.
Студент: Румянцев А. А.
Поток: ЧАСТ.МЕТ. 1.3

Факультет: СУиР
Группа: R3241

Санкт-Петербург
2024

Содержание

1 Задание 1. Жесткие фильтры

В этом задании выполним жесткую фильтрацию сигнала. Для этого будем находить Фурье-образ сигнала, обнулять его значения на некоторых диапазонах частот, затем восстанавливать сигнал с помощью обратного преобразования.

Зададим такие числа a, t_1, t_2, что $t_1 < t_2$, и рассмотрим функцию g такую, что $g(t) = a$ при $t \in [t_1, t_2]$ и $g(t) = 0$ при других t.

$$\sqsupset a = 2, \quad t_1 = -1.5, \quad t_2 = 2.5, \quad g(t) = \begin{cases} 2, & t \in [t_1, t_2] \\ 0, & \text{otherwise} \end{cases}$$

Выберем интервал времени $T = 10$ и шаг дискретизации $dt = 0.01$. Зададим в python массив времени t от $-0.5 \cdot T$ до $0.5 \cdot T + dt$ с шагом dt и включим последнюю точку. Найдем список значений $g(t)$ и зададим зашумленную версию сигнала как

$$u = g(t) + b \cdot (\text{random}(\text{len}(t)) - 0.5) + c \cdot \sin(d \cdot t);$$

В заданном сигнале параметр a функции $g(t)$ отвечает за высоту, на которую поднимется часть сигнала от нуля на промежутке $[t_1, t_2]$. Оставшиеся параметры вносят различные виды шума в сигнал.

1.1 Убираем высокие частоты. Фильтр нижних частот

Положим параметр $c = 0$. Возьмем некоторый диапазон частот $[-\nu_0, \nu_0]$, на котором оставим Фурье-образ сигнала u неизменным, а на остальных частотах обнулим его значения. То есть применим фильтр *нижних* частот – он пропускает все частоты ниже частоты среза. Построим сравнительные графики исходного и фильтрованного сигналов, а также модулей их Фурье-образов. Исследуем влияние частоты среза ν_0 и значения параметра b на эффективность фильтрации.

Далее будут приведены рисунки полученных графиков. На каждом графике подписаны выбранные значения b, c, d, ν_0. Синим цветом обозначается оригинальный сигнал, красным фильтрованный.

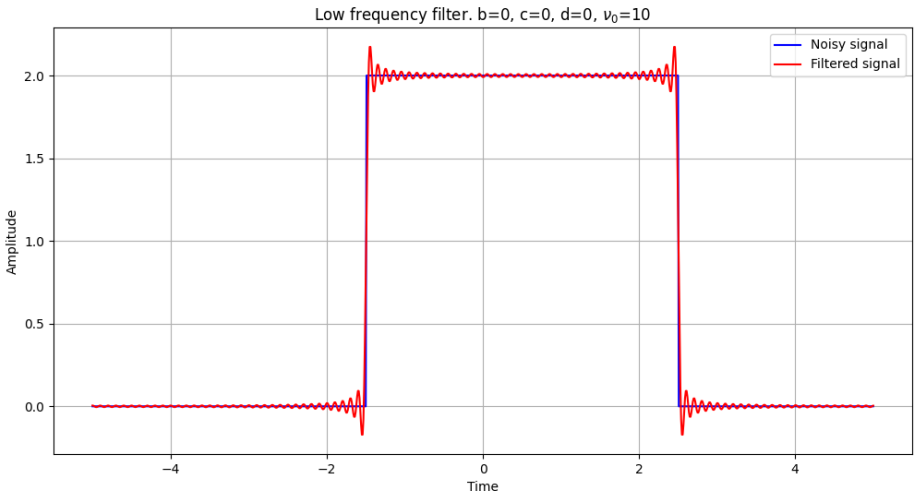


Рис. 1: График исходного и фильтрованного сигналов (1)

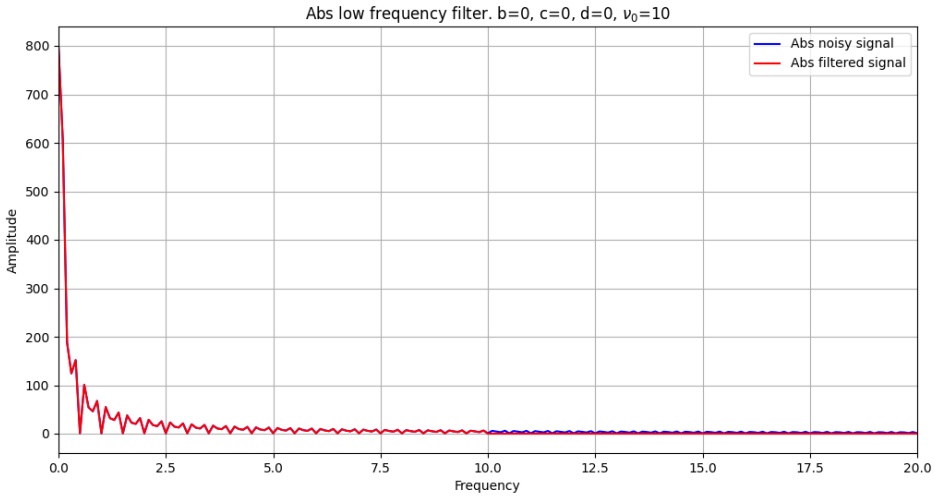


Рис. 2: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

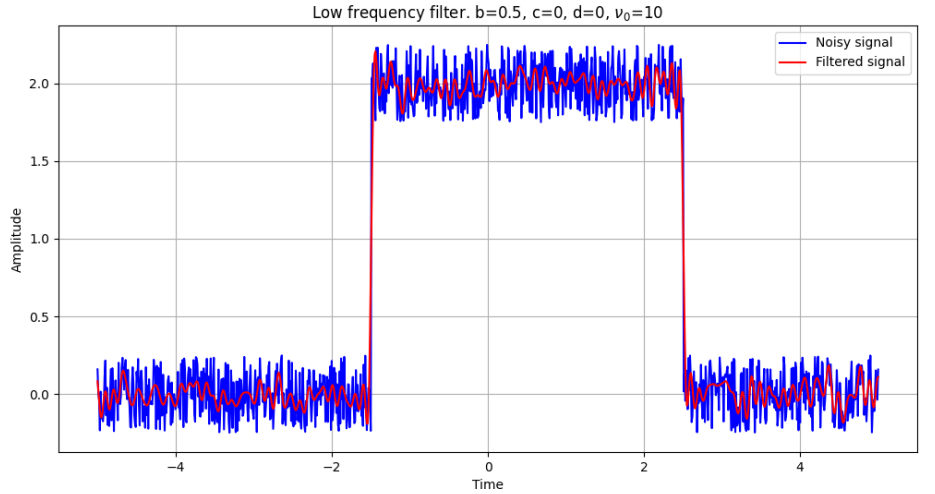


Рис. 3: График исходного и фильтрованного сигналов (2)

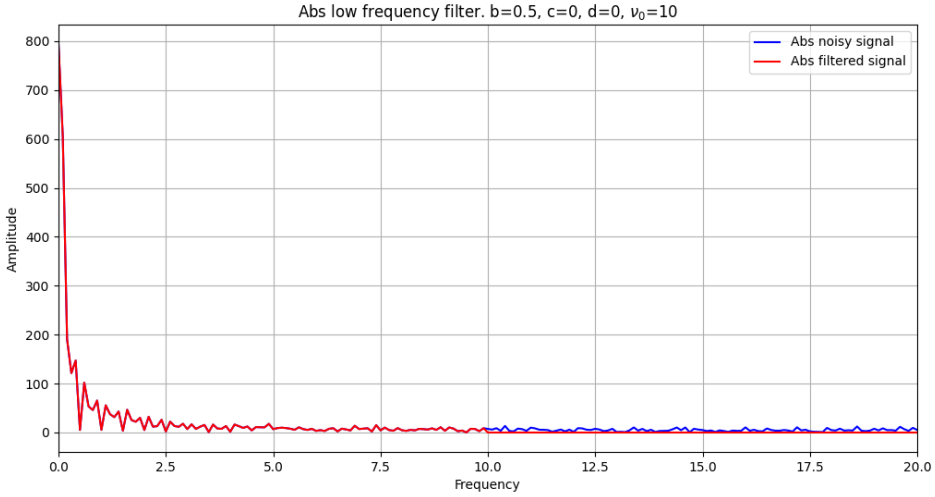


Рис. 4: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

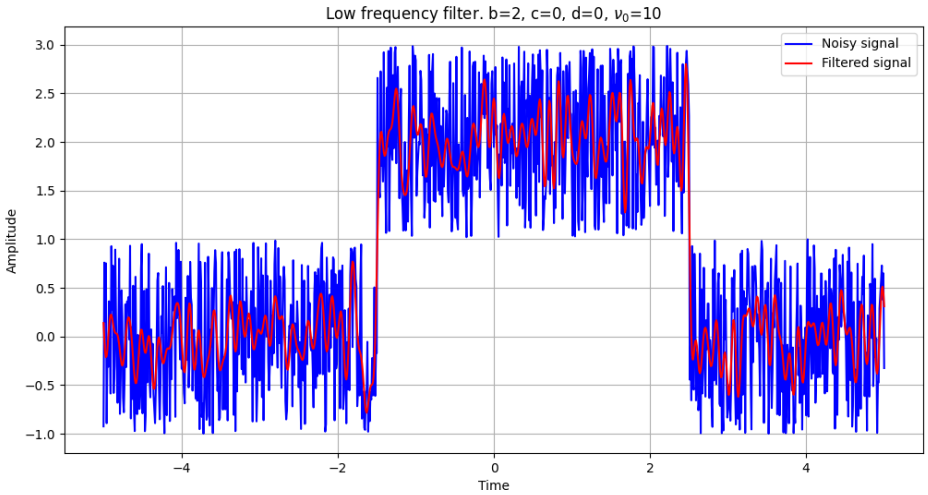


Рис. 5: График исходного и фильтрованного сигналов (3)

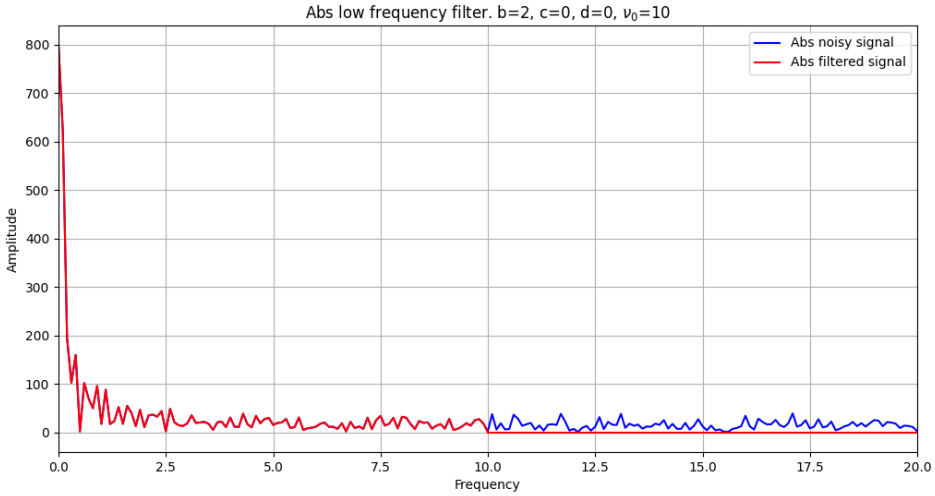


Рис. 6: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

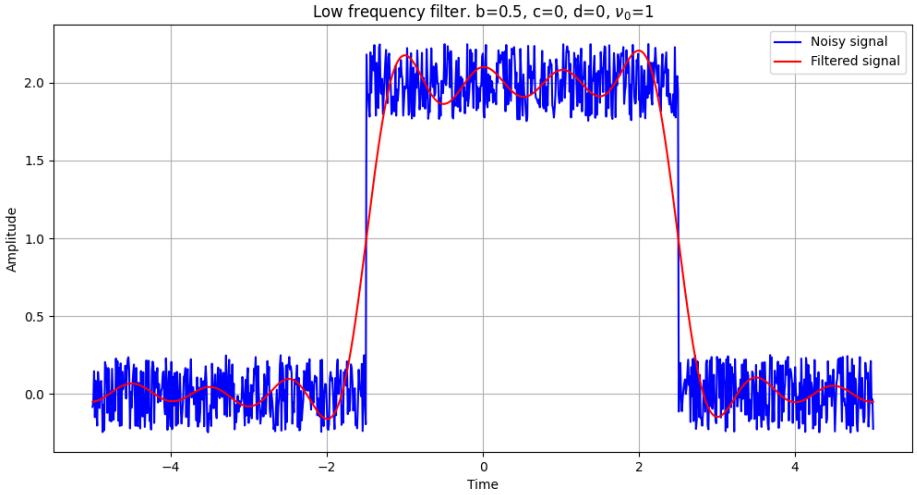


Рис. 7: График исходного и фильтрованного сигналов (4)

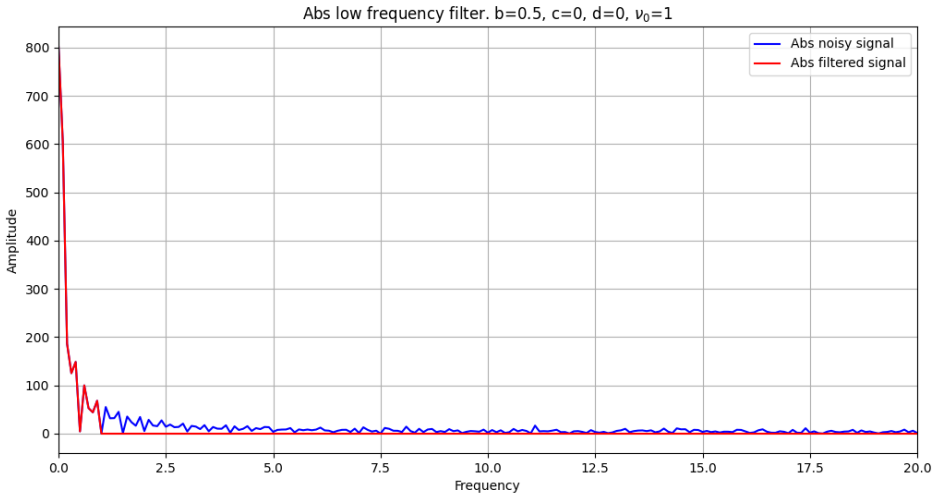


Рис. 8: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

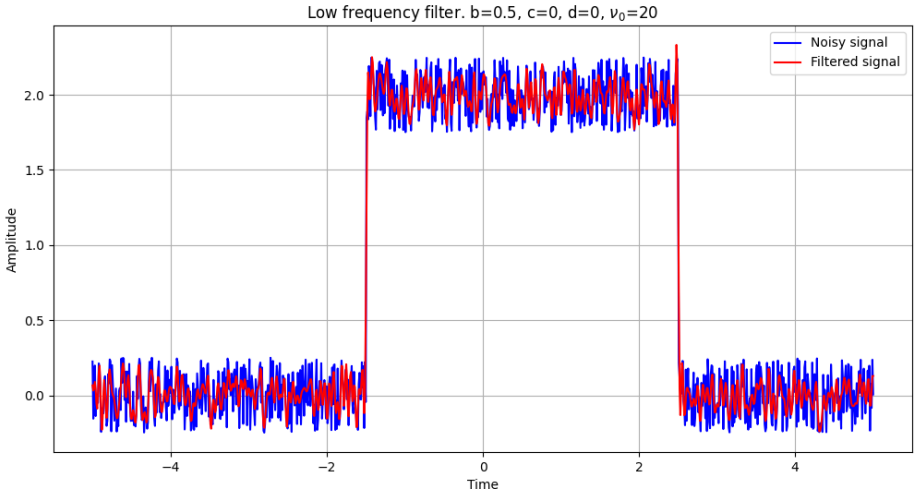


Рис. 9: График исходного и фильтрованного сигналов (5)

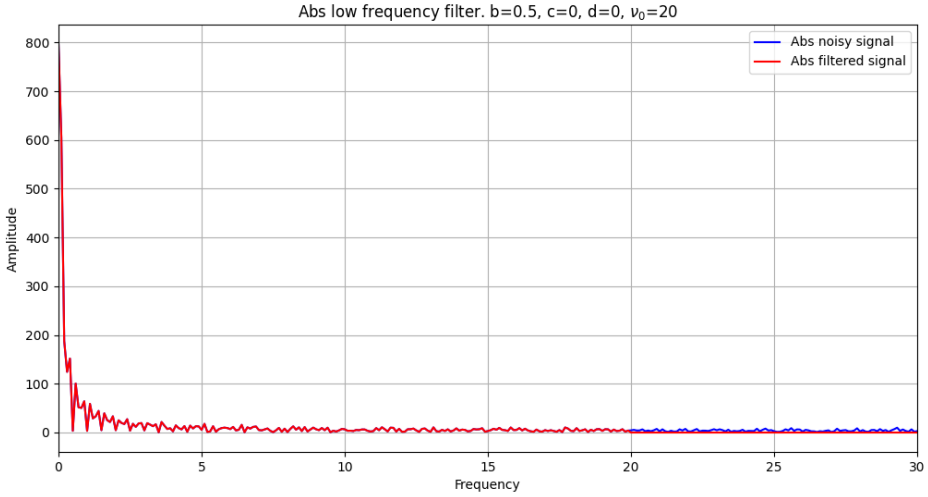


Рис. 10: График модуля Фурье-образа исходного и фильтрованного сигналов (5)

Исходя из графиков можно сделать вывод, что значение параметра b отвечает за белый шум. Чем больше значение b, тем больше белого шума (сравн. рис. 1, 3, 5) – сложнее фильтровать сигнал, так как увеличивается его концентрация в значащих частотах. В данном случае основная информация о сигнале находится на нижних частотах вблизи нуля – это мы видим на графиках Фурье-образов. При $b = 0$ сигнал становится прямоугольной функцией. Фильтрация такого сигнала выглядит как ее аппроксимация (см. рис. 1). При большом значении b исходный сигнал восстановить не получится.

Эффективность фильтрации зависит от выбора частоты среза ν_0. Зададим маленький диапазон – вырежем бóльшую часть шума и часть нужного сигнала (см. рис. 8). Большой диапазон оставит и сигнал и некоторую часть шума, которую можно было вырезать (см. рис. 10). Значение, сохраняющее наибольшее количество значащих частот, будет являться оптимальным и даст наилучшую фильтрацию.

1.2 Убираем специфические частоты. Режекторный фильтр

Возьмем ненулевые параметры b, c, d. Попробуем обнулять некоторые диапазоны частот, а также совместим различные варианты фильтрации, чтобы по возможности убрать влияние обеих компонент помехи. Исследуем влияние частот среза и значений параметров b, c, d на вид помехи и эффективность фильтрации. Отдельно рассмотрим случай для $b = 0$.

Синусоидальный шум отображается на графике Фурье-образа как высокий пик помимо исходного, который мы наблюдали ранее в диапазоне низких частот и который содержит наибольшее количество информации об исходном сигнале. Чтобы увидеть этот дополнительный высокий пик, необходимо взять немаленькое значение параметра d. Обнулим частоты в его диапазоне и посмотрим результат. Должен получиться сигнал, похожий на прямоугольную функцию. Также совместим фильтрацию специфических частот с низкими, то есть уберем высокие, которые несут наименьшее количество информации об исходном сигнале, но вносят значительный шум.

Далее будут приведены рисунки полученных графиков. На каждом графике подписаны выбранные значения b, c, d, ν_0. Синим цветом обозначается оригинальный сигнал, красным фильтрованный.

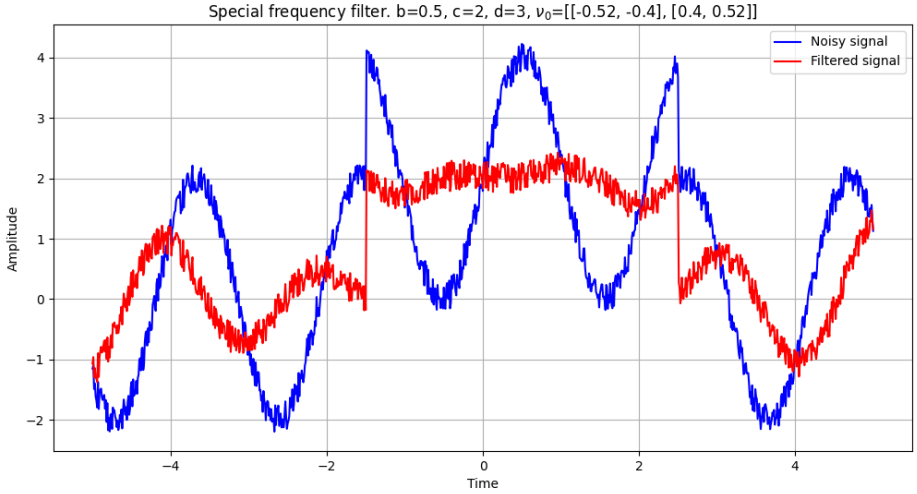


Рис. 11: График исходного и фильтрованного сигналов (1.1)

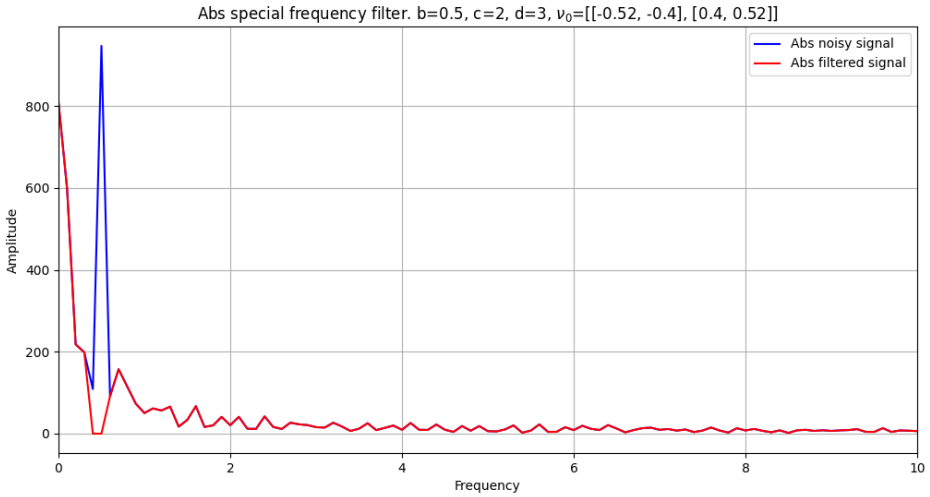


Рис. 12: График модуля Фурье-образа исходного и фильтрованного сигналов (1.1)

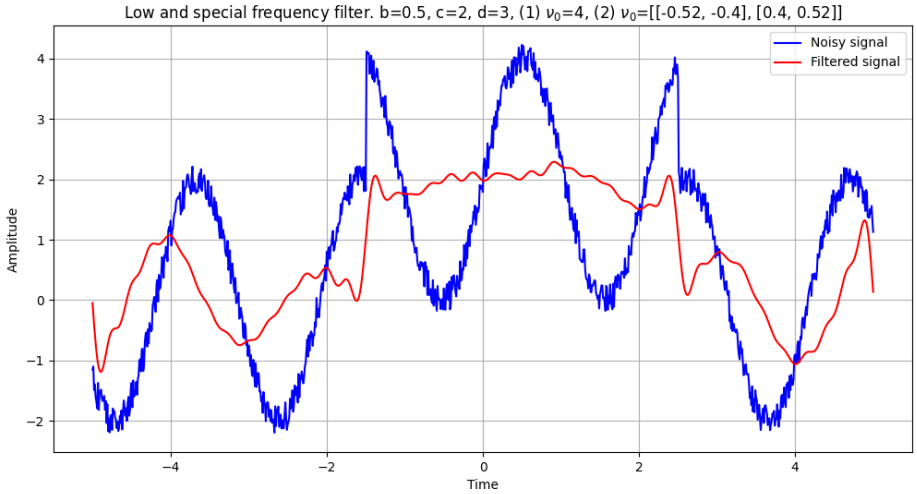


Рис. 13: График исходного и фильтрованного сигналов (1.2)

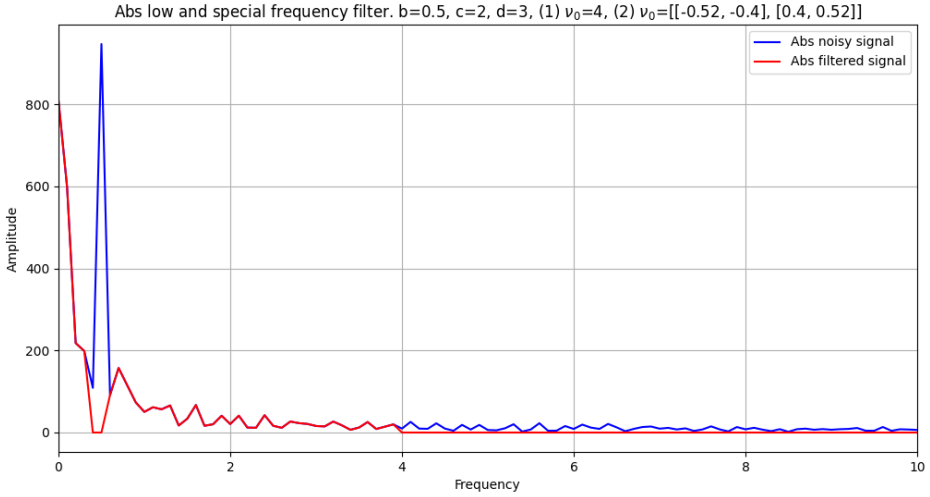


Рис. 14: График модуля Фурье-образа исходного и фильтрованного сигналов (1.2)

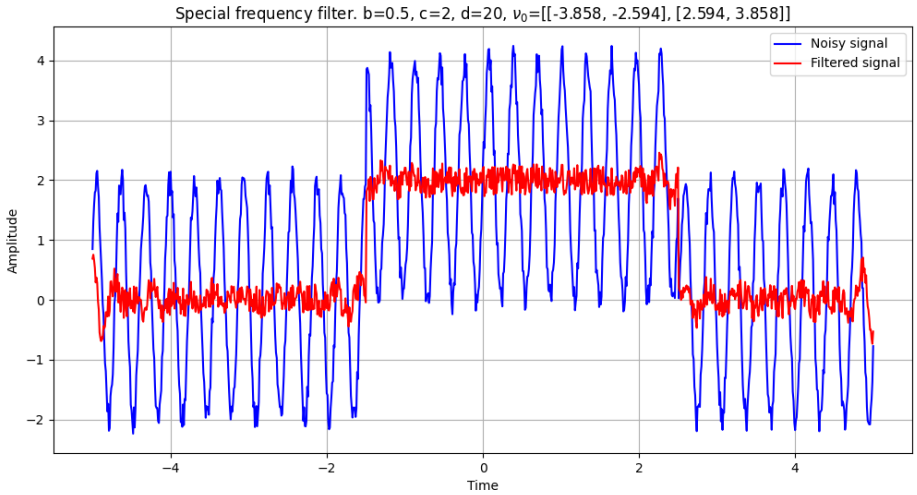


Рис. 15: График исходного и фильтрованного сигналов (2.1)

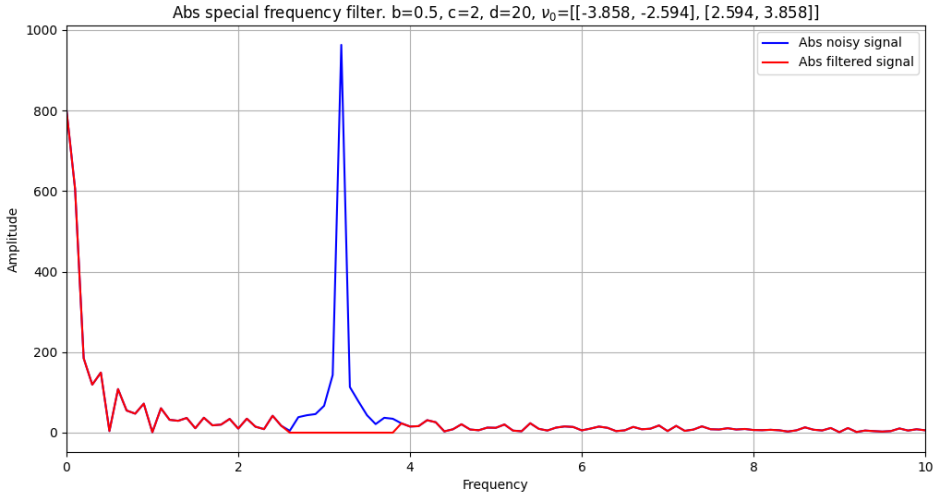


Рис. 16: График модуля Фурье-образа исходного и фильтрованного сигналов (2.1)

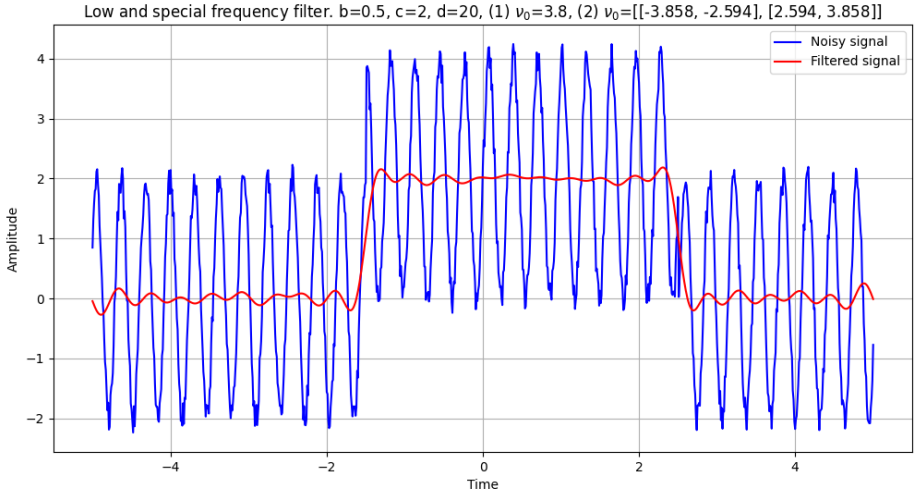


Рис. 17: График исходного и фильтрованного сигналов (2.2)

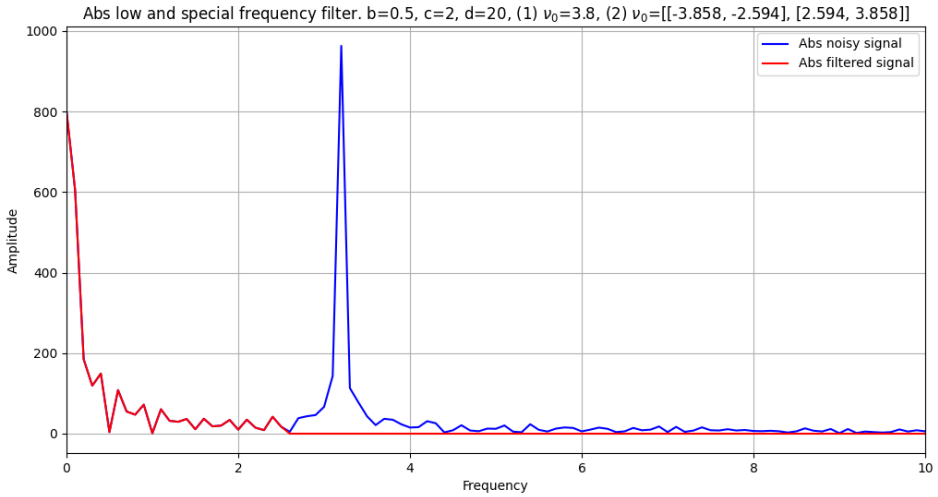


Рис. 18: График модуля Фурье-образа исходного и фильтрованного сигналов (2.2)

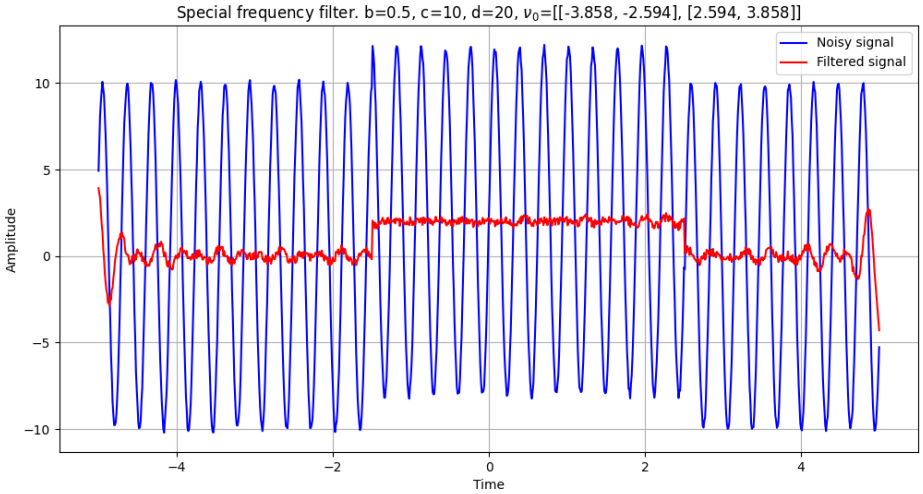


Рис. 19: График исходного и фильтрованного сигналов (3.1)

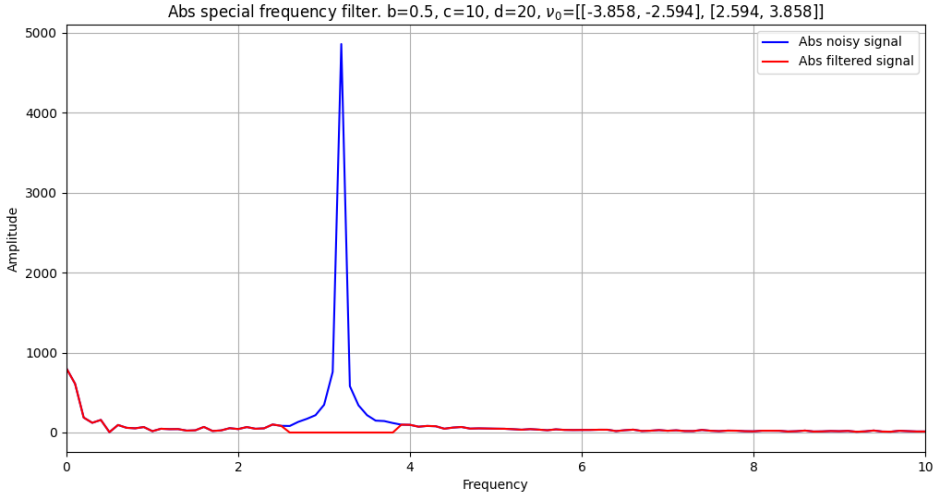


Рис. 20: График модуля Фурье-образа исходного и фильтрованного сигналов (3.1)

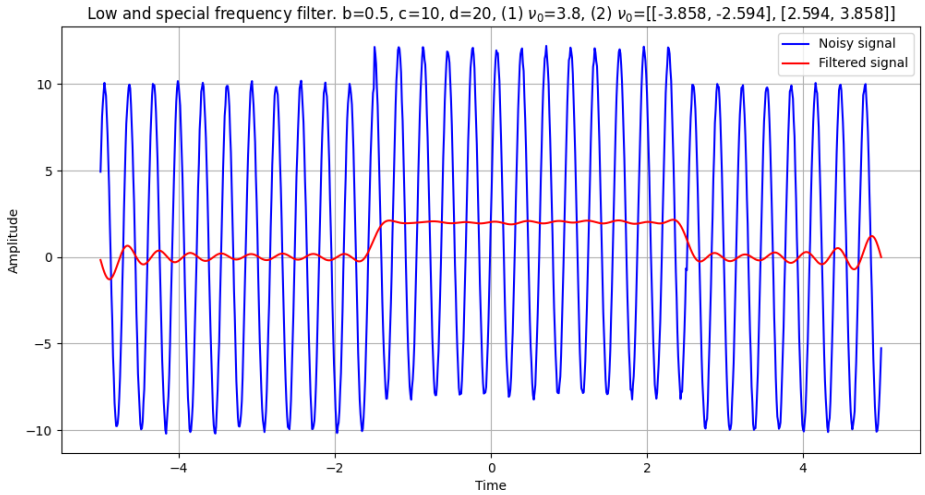


Рис. 21: График исходного и фильтрованного сигналов (3.2)

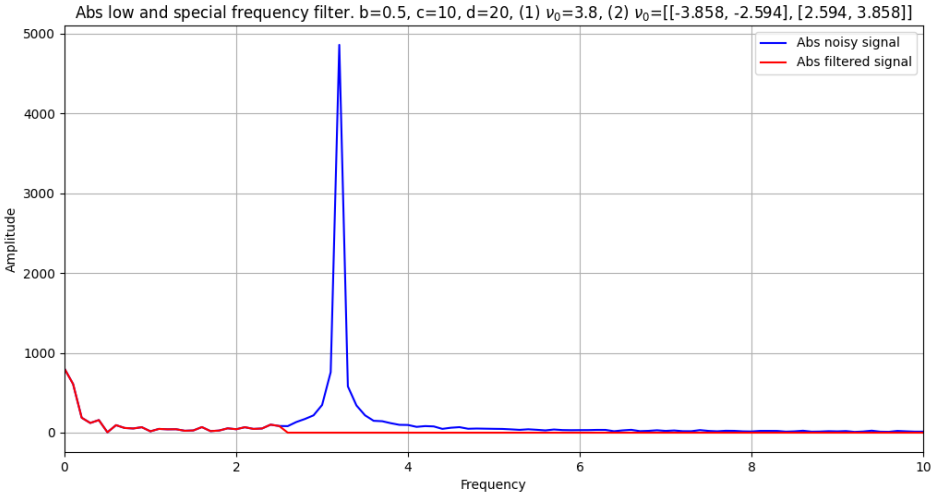


Рис. 22: График модуля Фурье-образа исходного и фильтрованного сигналов (3.2)

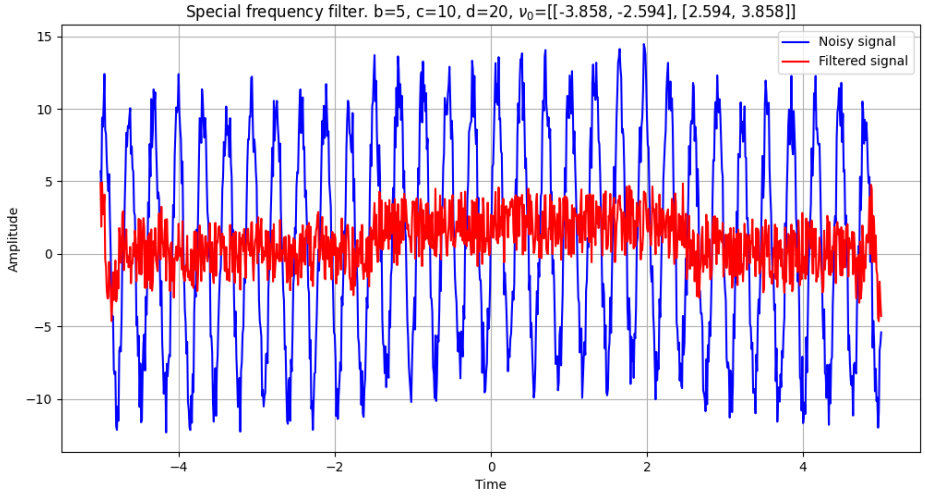


Рис. 23: График исходного и фильтрованного сигналов (4.1)

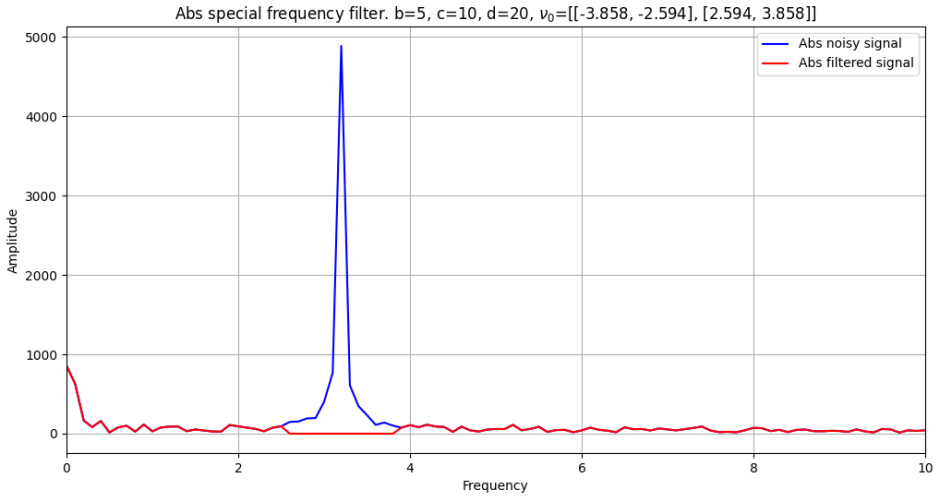


Рис. 24: График модуля Фурье-образа исходного и фильтрованного сигналов (4.1)

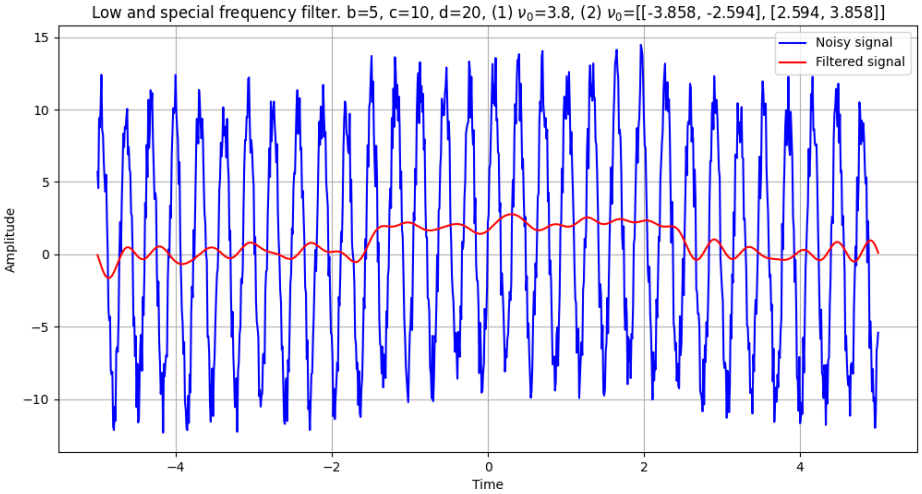


Рис. 25: График исходного и фильтрованного сигналов (4.2)

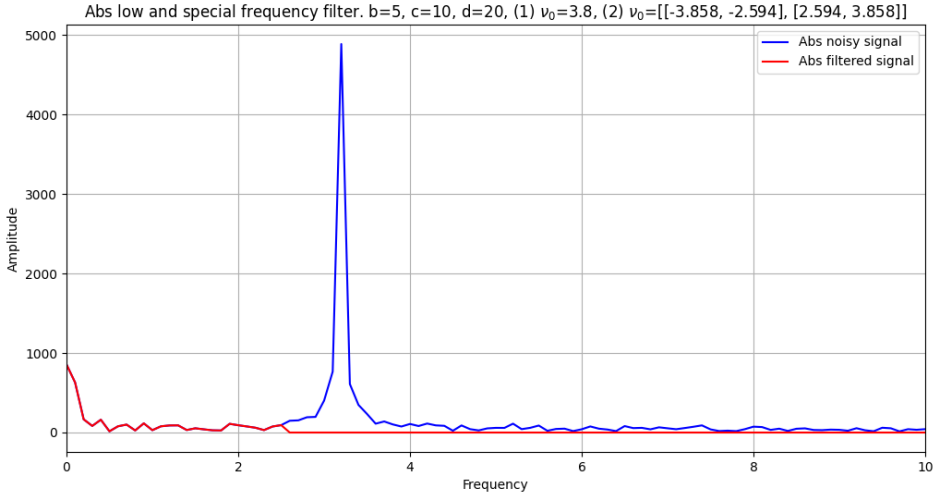


Рис. 26: График модуля Фурье-образа исходного и фильтрованного сигналов (4.2)

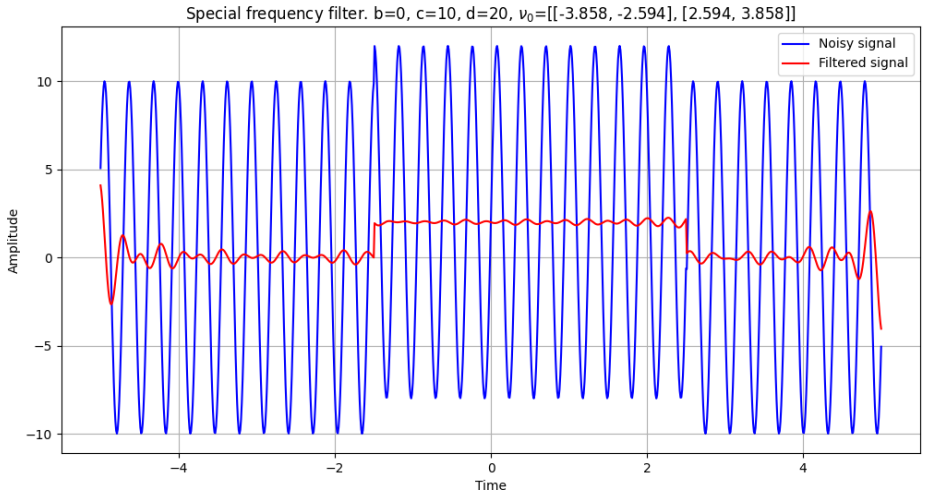


Рис. 27: График исходного и фильтрованного сигналов (5.1)

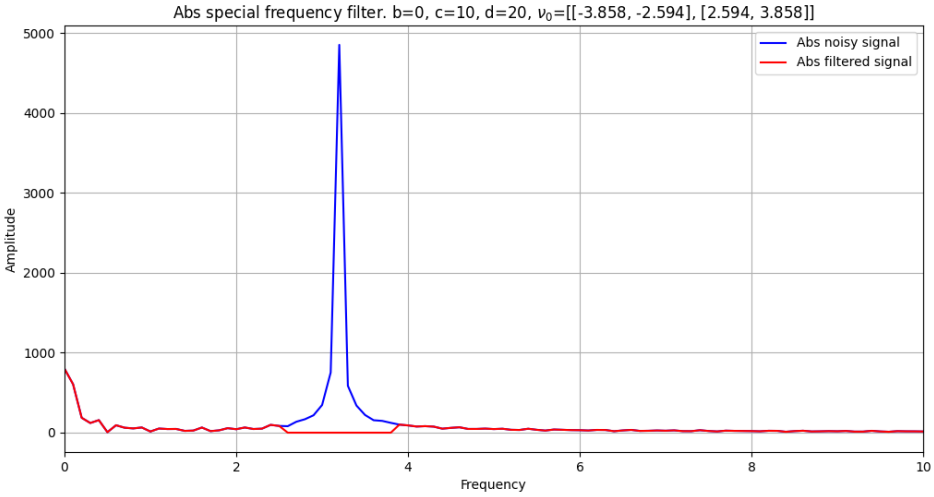


Рис. 28: График модуля Фурье-образа исходного и фильтрованного сигналов (5.1)

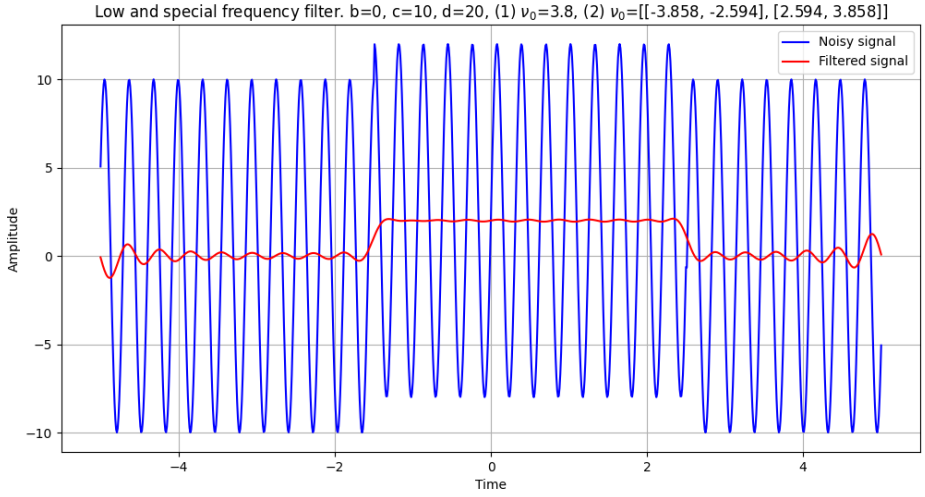


Рис. 29: График исходного и фильтрованного сигналов (5.2)

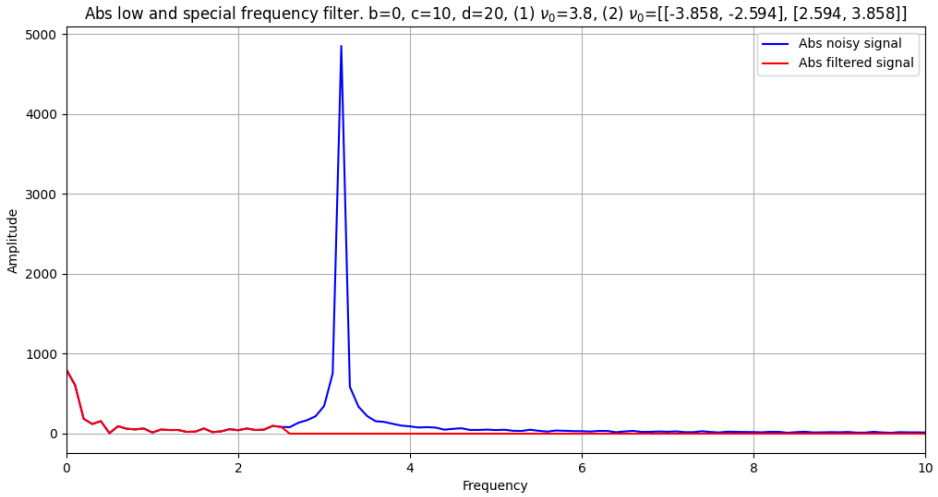


Рис. 30: График модуля Фурье-образа исходного и фильтрованного сигналов (5.2)

Исходя из графиков можно сделать вывод, что параметр c отвечает за синусоидальный шум. При увеличении параметра c сигнал во временной области растягивается по оси Oy, то есть возрастает амплитуда волн (сравн. рис. 15, 19). На графиках Фурье-образов амплитуды также становятся больше (сравн. рис. 16, 20). Затрудняет фильтрацию при больших значениях.

Параметр d характеризует растяжение сигнала во временной области по оси Ox. При увеличении его значения частота волн повышается. На графиках Фурье-образов увеличение значения параметра d удаляет от нижних частот высокий пик, отвечающий за гармонический шум (сравн. рис. 12, 16), что упрощает фильтрацию сигнала, так как, например, на рис. 12 синусоидальный шум находится в том же месте, где наибольшее количество информации о сигнале – его удаление приведет к потере некоторой информации об исходном сигнале, восстановленный будет выглядеть хуже, чем при бóльших значениях параметра d.

Чем меньше b, тем меньше белого шума и проще восстанавливать сигнал. При $b = 0$ сигнал u описывается формулой синусоиды

$$u = g + c \cdot \sin(d \cdot t + 0),$$

что можно увидеть на рис. 27.

Как виды помехи параметры можно назвать так: b – белый шум, c и d гармонический шум.

Наилучшей фильтрации мне удалось достичь при совмещении фильтра специфических и нижних частот.

1.3 Убираем низкие частоты. Фильтр верхних частот.

Рассмотрим графики, где в некоторой окресности точки $\nu = 0$ обнулим все значения частот Фурье-образа. Такое поведение соответствует фильтру *верхних* частот, так как он пропускает все частоты выше частоты среза. Окресность будет настраиваться выбором диапазона частот $[-\nu_0, \nu_0]$.

Далее будут приведены рисунки полученных графиков. На каждом графике подписаны выбранные значения b, c, d, ν_0. Синим цветом обозначается оригинальный сигнал, красным фильтрованный.

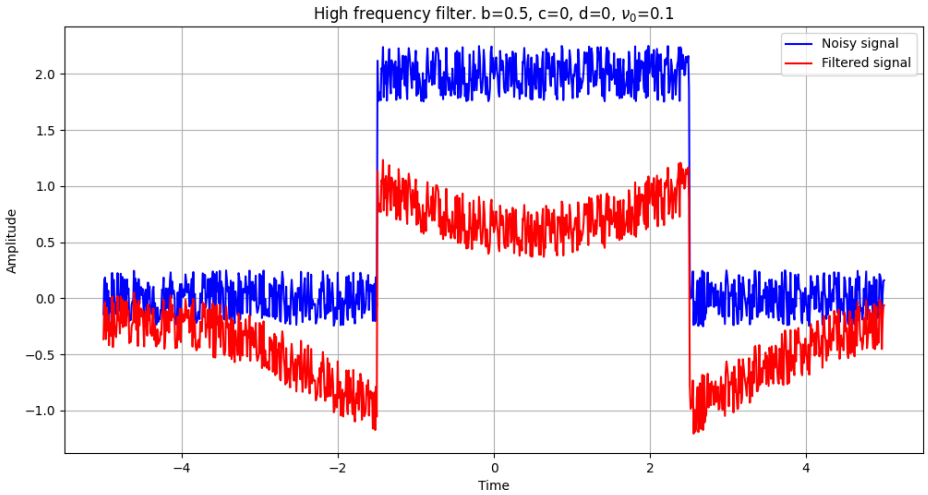


Рис. 31: График исходного и фильтрованного сигналов (1)

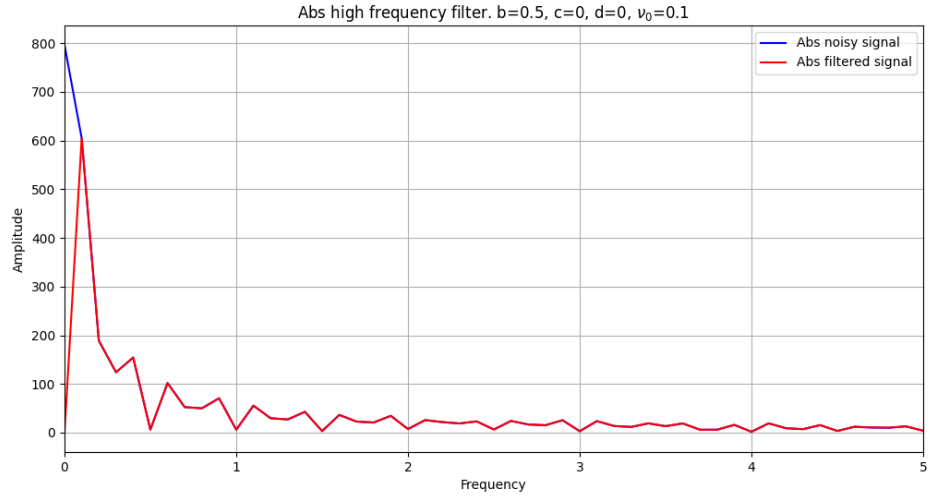


Рис. 32: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

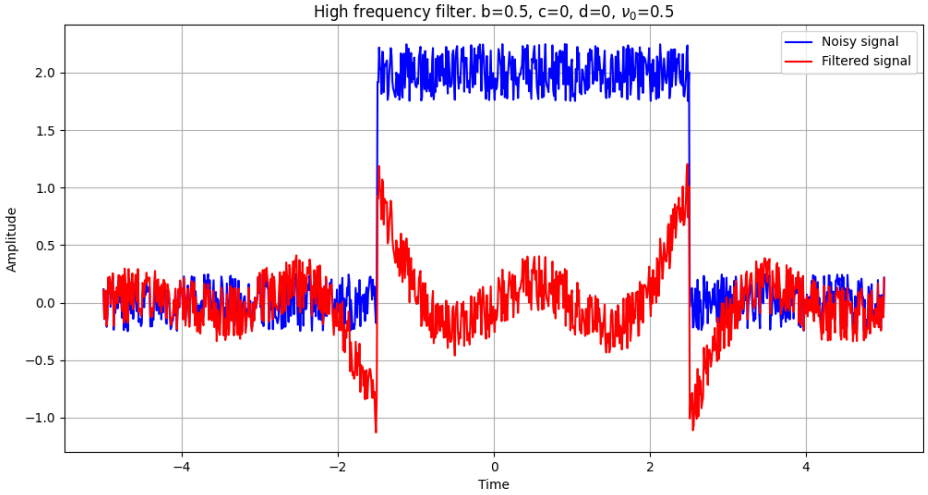


Рис. 33: График исходного и фильтрованного сигналов (2)

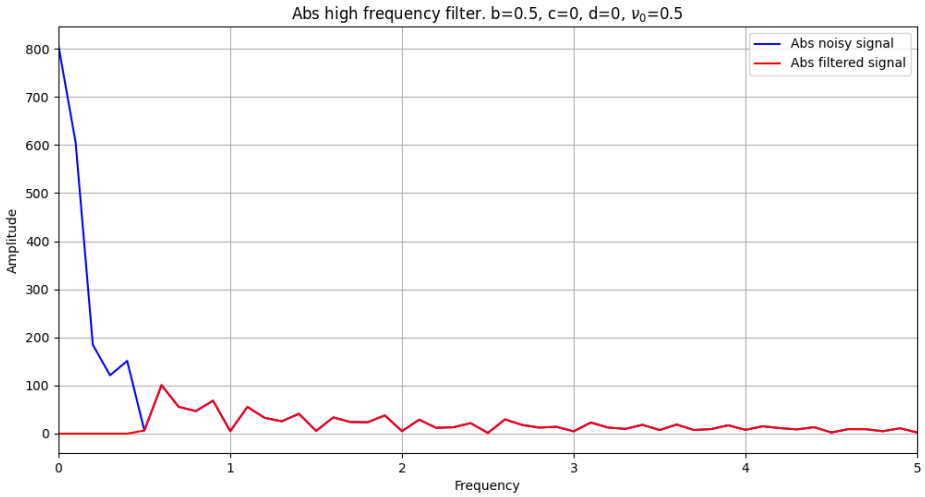


Рис. 34: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

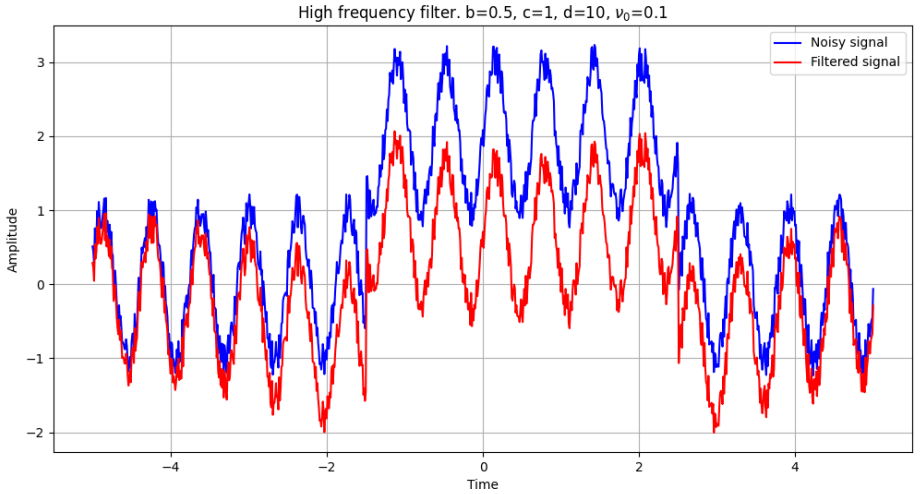


Рис. 35: График исходного и фильтрованного сигналов (3)

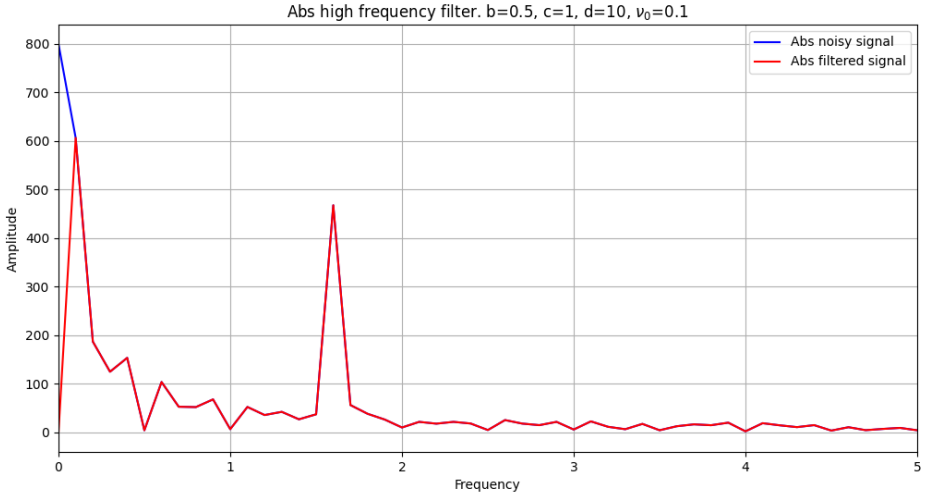


Рис. 36: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

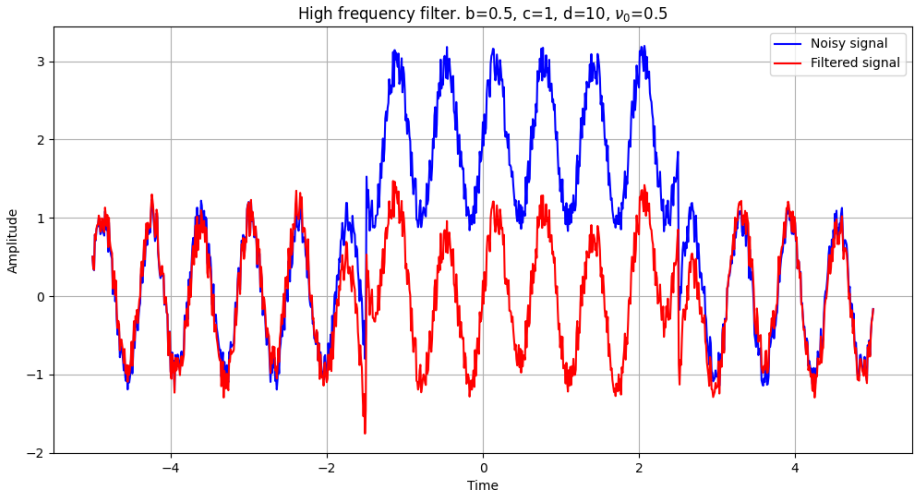


Рис. 37: График исходного и фильтрованного сигналов (4)

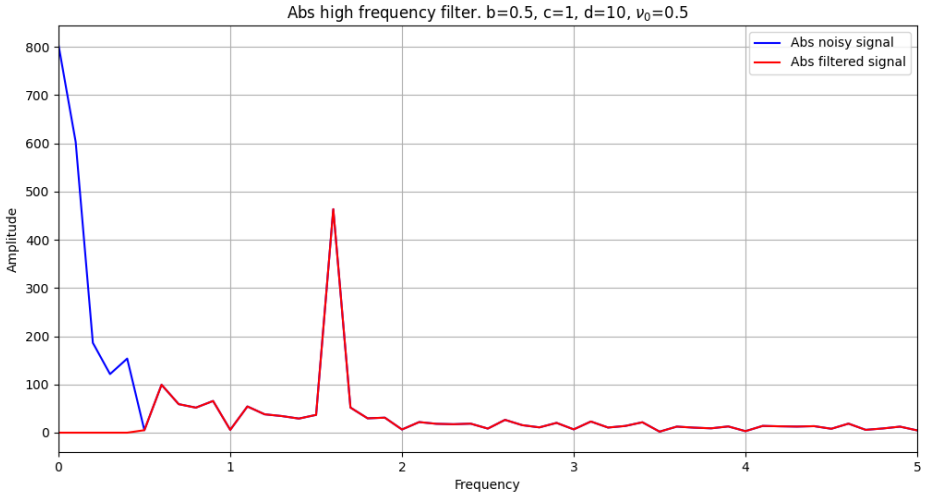
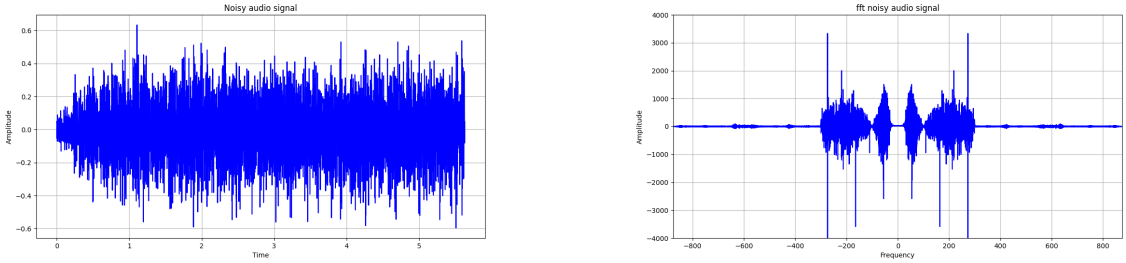


Рис. 38: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

По полученным графикам можно сделать вывод, что фильтр верхних частот в данном случае отрезает значимую часть сигнала, содержащуюся в нижних частотах, и оставляет только белый и/или гармонический шум, находящийся преимущественно на высоких частотах. Например, на рис. 37 видно, что фильтрованный сигнал почти не имеет подъема на том промежутке, где он должен быть, то есть почти вся информация о том, что сигнал имел подъем, была вырезана (см. рис. 38). Следовательно, для заданного сигнала u такой фильтр не подходит для восстановления исходного сигнала.

2 Задание 2. Фильтрация звука.

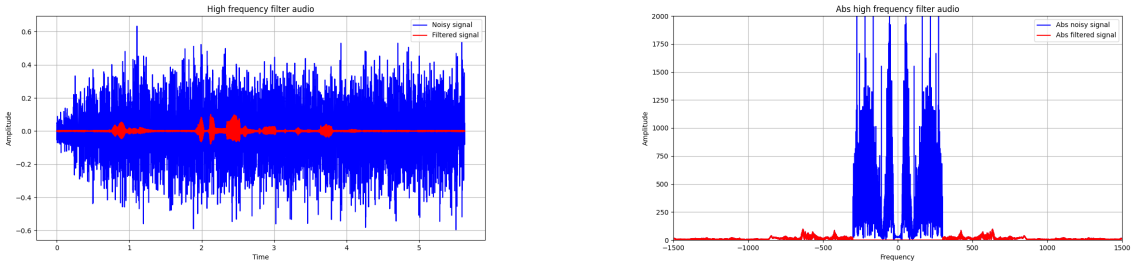
Построим графики исходного сигнала аудиозаписи и его Фурье-образа. На записи присутствует громкий гул, следовательно необходимо вырезать частоты из аудиозаписи, имеющие наибольшую амплитуду.



(a) Временная область (b) Частотная область

Рис. 39: Графики исходного сигнала и его Фурье-образа

Определим по второму графику, какие частоты могут создавать шумы. Самые большие амплитуды находятся в диапазоне частот $[-300, 300]$ Гц. Чтобы их вырезать, потребуется фильтр верхних частот, который мы рассматривали в задании 1. Применим фильтр и посмотрим результирующие графики.



(a) Временная область (b) Частотная область

Рис. 40: Графики исходного и фильтрованного сигналов и модулей их Фурье-образов

Видим как из сигнала во временной области пропал белый шум – оставшиеся амплитуды являются голосом, который мы хотели хорошо расслышать. На графике модулей Фурье-образов наблюдаем, что мы вырезали низкие частоты с наибольшей амплитудой, которые соответствовали громкому гулу. Запись filtered_MUHA.wav можно послушать, на этом гугл-диске. На записи голос слышно хорошо, однако остался некоторый звуковой эффект фейзер.

3 Листинги программных реализаций

Все программы написаны на языке python с подключенными библиотеками numpy и matplotlib.

Два файла, которые необходимы для задания массива времени и частот, вычисления массива функций g, задания сигнала u, а также подсчета Фурье-образа сигнала u. Достаточно передать в функции необходимые параметры и далее их результат можно использовать для выполнения фильтрации и построения графиков.

```
1     def get_t(T, dt):
2         return np.arange(-T/2, T/2 + dt, dt)
3
4     def get_v(V, dv):
5         return np.arange(-V/2, V/2 + dv, dv)
6
```

```
    def get_U(u):
        return np.fft.fftshift(np.fft.fft(u))

    def g_f(t, t_1, t_2, a):
        if (t_1 <= t <= t_2):
            return a
        return 0

    def u_f(g_fs: list, time: list, b, c, d):
        return np.array(g_fs) + \
               b*(np.random.rand(len(time))-0.5) + \
               c*np.sin(d*time)

    def get_g_fs(time: list, t_1, t_2, a):
        gs = []
        for t in range(len(time)):
            gs.append(g_f(time[t], t_1, t_2, a))

        return gs
```

Листинг 1: Файл help.py. Вспомогательные функции

```
    import help as hp

    a = 2
    t_1, t_2 = -1.5, 2.5

    T, dt = 10, 0.01
    V, dv = 1 / dt, 1 / T

    time = hp.get_t(T, dt)
    freq = hp.get_v(V, dv)
    g_fs = hp.get_g_fs(time, t_1, t_2, a)
```

Листинг 2: Файл static.py. Вспомогательные переменные

Программа для построения сравнительных графиков исходного и фильтрованного сигналов и модулей их Фурье-образов.

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

    from help import get_U

    def build_u_to_U(freq: list,
                     u: list,
                     clr='b',
                     xl1=None,
                     xl2=None,
                     yl1=None,
                     yl2=None,
                     xlab='Frequency',
                     ylab='Amplitude',
                     label=None,
                     title=None,
                     fz1=6.4,
                     fz2=4.8,
                     legend: bool = True,
                     grid: bool = True):
        U = get_U(u)
        build_u_or_U(freq, U, clr, xl1, xl2,
                     yl1, yl2, xlab, ylab, label,
```

```
                    title, fz1, fz2, legend, grid)

    def build_u_or_U(torv: list,
                     uorU: list,
                     clr='b',
                     xl1=None,
                     xl2=None,
                     yl1=None,
                     yl2=None,
                     xlab=None,
                     ylab='Amplitude',
                     label=None,
                     title=None,
                     fz1=6.4,
                     fz2=4.8,
                     legend: bool = True,
                     grid: bool = True):
        plt.plot(torv, uorU, color=clr, label=label)
        plt.xlabel(xlab)
        plt.ylabel(ylab)
        plt.xlim(xl1, xl2)
        plt.ylim(yl1, yl2)
        plt.title(title)
        if legend:
            plt.legend()
        plt.grid(grid)
        plt.gcf().set_size_inches(fz1, fz2)
        plt.show()

    def build_u__flt_u(time: list,
                     u: list,
                     flt_u: list,
                     clr1='b',
                     clr2='r',
                     lab1='Noisy signal',
                     lab2='Filtered signal',
                     xlab='Time',
                     ylab='Amplitude',
                     title=None,
                     fz1=6.4,
                     fz2=4.8,
                     legend: bool = True,
                     grid: bool = True,
                     xl1=None,
                     xl2=None,
                     yl1=None,
                     yl2=None):
        plt.plot(time, u, color=clr1, label=lab1)
        plt.plot(time, flt_u, color=clr2, label=lab2)
        plt.xlabel(xlab)
        plt.ylabel(ylab)
        plt.xlim(xl1, xl2)
        plt.ylim(yl1, yl2)
        plt.title(title)
        if legend:
            plt.legend()
        plt.grid(grid)
        plt.gcf().set_size_inches(fz1, fz2)
        plt.show()

```

```
84      def build_abs_u_to_U__flt_U(freq: list,
85                                  u: list,
86                                  flt_U: list,
87                                  clr1='b',
88                                  clr2='r',
89                                  lab1='Abs noisy signal',
90                                  lab2='Abs filtered signal',
91                                  xlab='Frequency',
92                                  ylab='Amplitude',
93                                  xl1=None,
94                                  xl2=None,
95                                  yl1=None,
96                                  yl2=None,
97                                  title=None,
98                                  fz1=6.4,
99                                  fz2=4.8,
100                                 legend: bool = True,
101                                 grid: bool = True):
102         U = get_U(u)
103         build_abs_U__flt_U(freq, U, flt_U, clr1,
104                            clr2, lab1, lab2, xlab,
105                            ylab, xl1, xl2, yl1, yl2,
106                            title, fz1, fz2, legend, grid)
107
108     def build_abs_U__flt_U(freq: list,
109                            U: list,
110                            flt_U: list,
111                            clr1='b',
112                            clr2='r',
113                            lab1='Abs noisy signal',
114                            lab2='Abs filtered signal',
115                            xlab='Frequency',
116                            ylab='Amplitude',
117                            xl1=None,
118                            xl2=None,
119                            yl1=None,
120                            yl2=None,
121                            title=None,
122                            fz1=6.4,
123                            fz2=4.8,
124                            legend: bool = True,
125                            grid: bool = True):
126         plt.plot(freq, np.abs(U), color=clr1, label=lab1)
127         plt.plot(freq, np.abs(flt_U), color=clr2, label=lab2)
128         plt.xlabel(xlab)
129         plt.ylabel(ylab)
130         plt.xlim(xl1, xl2)
131         plt.ylim(yl1, yl2)
132         plt.title(title)
133         if legend:
134             plt.legend()
135         plt.grid(grid)
136         plt.gcf().set_size_inches(fz1, fz2)
137         plt.show()
```

Листинг 3: Файл builder.py. Реализация построения графиков

Программа, реализующая методы фильтрации верхних, нижних и специфических частот. Здесь реализован алгоритм, описанный в первом задании – вычисляется Фурье-образ, фильтруется и преобразуется обратно из частотного пространства во временное.

```
    def filter_U(u: list, freq: list, v_0, filter):
        flt_U = get_U(u)
        for i in range(len(freq)):
            freq_i = freq[i]
            if filter(freq_i, v_0):
                flt_U[i] = 0

        return flt_U

    def low_filter(freq, v_0):
        if -v_0 <= freq <= v_0:
            return False
        return True

    def high_filter(freq, v_0):
        return not low_filter(freq, v_0)

    def special_filter_in(freq, v_0: list):
        for i in range(len(v_0)):
            if v_0[i][0] <= freq <= v_0[i][1]:
                return False
        return True

    def special_filter_out(freq, v_0: list):
        return not special_filter_in(freq, v_0)

    def filter_low(freq: list, u: list, v_0):
        if isinstance(v_0, list) or \
                len(freq) <= 0 or \
                len(u) <= 0:
            return None

        flt_U = filter_U(u, freq, v_0, low_filter)
        flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
        return flt_u, flt_U

    def filter_high(freq: list, u: list, v_0):
        if isinstance(v_0, list) or \
                len(freq) <= 0 or \
                len(u) <= 0:
            return None

        flt_U = filter_U(u, freq, v_0, high_filter)
        flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
        return flt_u, flt_U

    def filter_special_in(freq: list, u: list, v_0: list):
        if not isinstance(v_0, list) or \
                len(v_0) <= 0 or \
                len(freq) <= 0 or \
                len(u) <= 0:
            return None

        flt_U = filter_U(u, freq, v_0, special_filter_in)
        flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
        return flt_u, flt_U

    def filter_special_out(freq: list, u: list, v_0: list):
        if not isinstance(v_0, list) or \
                len(v_0) <= 0 or \
```

```
                len(freq) <= 0 or \
                len(u) <= 0:
            return None

        flt_U = filter_U(u, freq, v_0, special_filter_out)
        flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
        return flt_u, flt_U
```

Листинг 4: Файл filters.py. Реализация фильтров

Далее представлены программы, в которых используются все предыдущие наработки. Типовой алгоритм – задать параметры b, c, d и некоторый ν_0, далее воспользоваться функциями создания зашумленного сигнала, фильтрации нужных частот и построения необходимых графиков. Для работы с аудиозаписью подключены библиотеки librosa, scipy и playsound.

```
import help as hp
import static as st

import filters as ft
import builder as bd

time = st.time
freq = st.freq
g_fs = st.g_fs

b, c, d = 0.5, 0, 0.1
v_0 = 2.5

u = hp.u_f(g_fs, time, b, c, d)
flt_u, flt_U = ft.filter_low(freq, u, v_0)

bd.build_u__flt_u(
    time,
    u,
    flt_u,
    title=rf'Low frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0}',
    fz1=12,
    fz2=6)
bd.build_abs_u_to_U__flt_U(
    freq,
    u,
    flt_U,
    title=
    rf'Abs low frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0}',
    xl1=-5,
    xl2=5,
    fz1=12,
    fz2=6)
```

Листинг 5: Файл nohigh.py. Фильтрация нижних частот

```
import help as hp
import static as st

import filters as ft
import builder as bd

time = st.time
freq = st.freq
g_fs = st.g_fs
```

```

    b, c, d = 5, 10, 0.5
    v_0 = 0.3

    u = hp.u_f(g_fs, time, b, c, d)
    flt_u, flt_U = ft.filter_high(freq, u, v_0)

    bd.build_u__flt_u(
        time,
        u,
        flt_u,
        title=rf'High frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0}',
        fz1=12,
        fz2=6)
    bd.build_abs_u_to_U__flt_U(
        freq,
        u,
        flt_U,
        title=
        rf'Abs high frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0}',
        fz1=12,
        fz2=6,
        xl1=-5,
        xl2=5)
```

Листинг 6: Файл nolow.py. Фильтрация верхних частот

```
    import help as hp
    import static as st

    import filters as ft
    import builder as bd

    time = st.time
    freq = st.freq
    g_fs = st.g_fs

    b, c, d = 1.5, -2, -3
    v0, v01 = [[-0.52, -0.3], [0.3, 0.52]], 10
    u = hp.u_f(g_fs, time, b, c, d)

    flt_u_so, flt_U_so = ft.filter_special_out(freq, u, v0)
    bd.build_u__flt_u(
        time,
        u,
        flt_u_so.real,
        title=
        rf'Special frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v0}',
        fz1=12,
        fz2=6)
    bd.build_abs_u_to_U__flt_U(
        freq,
        u,
        flt_U_so,
        title=
        rf'Abs special frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v0}',
        fz1=12,
        fz2=6,
        xl1=-10,
        xl2=10)

```

```
    flt_u_lso, flt_U_lso = ft.filter_low(freq, flt_u_so, v01)
    bd.build_u__flt_u(
        time,
        u,
        flt_u_lso.real,
        title=
        rf'Low and special frequency filter. b={b}, c={c}, d={d}, (1) $\
           nu_0$={v01}, (2) $\nu_0$={v0}',
        fz1=12,
        fz2=6)
    bd.build_abs_u_to_U__flt_U(
        freq,
        u,
        flt_U_lso,
        title=
        rf'Abs low and special frequency filter. b={b}, c={c}, d={d}, (1) $\
           nu_0$={v01}, (2) $\nu_0$={v0}',
        fz1=12,
        fz2=6,
        xl1=-10 - v01,
        xl2=10 + v01)
```

Листинг 7: Файл nospec.py. Фильтрация специфических частот

```
    import numpy as np
    import librosa
    from scipy.io.wavfile import write
    from playsound import playsound

    import filters as ft
    import builder as bd

    title = 'High frequency filter audio'
    title2 = 'Abs high frequency filter audio'

    src = 'fm_lab3/sound/MUHA.wav'
    filename = 'fm_lab3/sound/filtered_MUHA.wav'
    audio, rate = librosa.load(src, sr=None)

    dt = 1 / rate
    T = len(audio) * dt

    time = np.linspace(0, T, len(audio), endpoint=False)
    freq = np.linspace(-rate / 2, rate / 2, len(audio), endpoint=False)

    flt_u, flt_U = ft.filter_high(freq, audio, 300)
    flt_u_float = flt_u.real.astype(np.float32)

    bd.build_u_or_U(time,
                    audio,
                    xlab='Time',
                    title='Noisy audio signal',
                    fz1=12,
                    fz2=6,
                    legend=False)
    bd.build_u_to_U(freq,
                    audio,
                    title='fft noisy audio signal',
                    legend=False,
                    fz1=12,
                    fz2=6,
```

```
                    xl1=0,
                    yl1=0,
                    yl2=20)

    bd.build_u__flt_u(time, audio, flt_u, title=title, fz1=12, fz2=6)
    bd.build_abs_u_to_U__flt_U(freq,
                               audio,
                               flt_U,
                               title=title2,
                               fz1=12,
                               fz2=6,
                               xl1=0,
                               yl1=0,
                               yl2=20)

    write(filename, rate, flt_u_float)
    playsound(filename)
```

Листинг 8: Файл audio.py. Фильтрация шумов в аудиозаписи