



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	2
1.1 Убираем высокие частоты. Фильтр нижних частот	2
1.2 Убираем специфические частоты. Режекторный фильтр	12
1.3 Убираем низкие частоты. Фильтр верхних частот.	23
2 Задание 2. Фильтрация звука.	41
3 Листинги программных реализаций	43

1 Задание 1. Жесткие фильтры

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 .

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

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 и зададим зашумленную версию сигнала как

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

В данном задании мы выполняем жесткую фильтрацию сигнала u . Алгоритм следующий: находится Фурье-образ от сигнала, обнуляются его значения на некоторых диапазонах частот, затем сигнал восстанавливается обратным преобразованием Фурье. Далее строятся графики с помощью программы на языке python. Используемый код с пояснениями находится в отдельной секции.

В задаваемом сигнале параметр a отвечает за высоту, на которую поднимется часть сигнала от нуля, а t_1 и t_2 – начало и конец промежутка с подъемом соответственно. Таким образом, на интервале длины $t_2 - t_1 = 2.5 + 1.5 = 4$, начиная с $t_1 = -1.5$ и заканчивая $t_2 = 2.5$, на высоте $a = 2$ будет находиться часть от всего сигнала, который, в свою очередь, располагается на промежутке $[-0.5 \cdot T, 0.5 \cdot T] = [-5, 5]$ длины $2 \cdot T \cdot 0.5 = 10$. Параметры b, c, d отвечают за шум, присутствующий в сигнале. Далее будут рассмотрены графики и сделаны выводы о влиянии каждого параметра на сам сигнал и на его результат фильтрации.

1.1 Убираем высокие частоты. Фильтр нижних частот

Возьмем параметр $c = 0$. Далее действуем в соответствии с алгоритмом. Возьмем некоторый диапазон частот $[-\nu_0, \nu_0]$, на котором оставим Фурье-образ сигнала u неизменным, а на остальных частотах обнулим его значения. Такое поведение соответствует фильтру **нижних** частот, так как он пропускает все частоты ниже частоты среза. Построим сравнительные графики исходного и фильтрованного сигналов на некотором интервале $[t_1, t_2]$, а также модуля Фурье-образа исходного и фильтрованного сигналов. 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 (хотя, при условии, что $c = 0$, менять или рассматривать параметр d не требуется). Также отмечена легенда – синим цветом обозначается оригинальный сигнал, красным фильтрованный.

Исходя из графиков можно сделать вывод, что значение параметра b отвечает за амплитуду каждой волны. Чем больше значение b , тем зашумленнее, «грязнее» выглядит сигнал, так как амплитуды волн возрастают. Фильтрованный сигнал при больших значениях b также имеет более глубокие ямы и высокие подъемы, то есть испытывает увеличение амплитуды. Наглядно такое возрастание можно наблюдать на графиках модуля Фурье-образа исходного и фильтрованного сигналов – на рисунке 2 амплитуды частот больше прижимаются к линии $A = 0$, а на рисунке 8 они больше стремятся к линии $A = 50$, где A – амплитуда. Значение параметра b влияет на эффективность фильтрации – чем

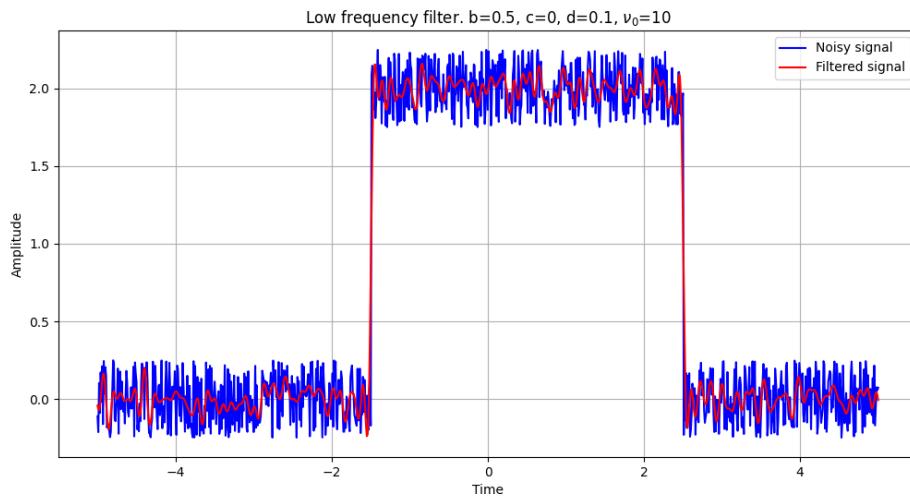


Рис. 1: График исходного и фильтрованного сигналов (1)

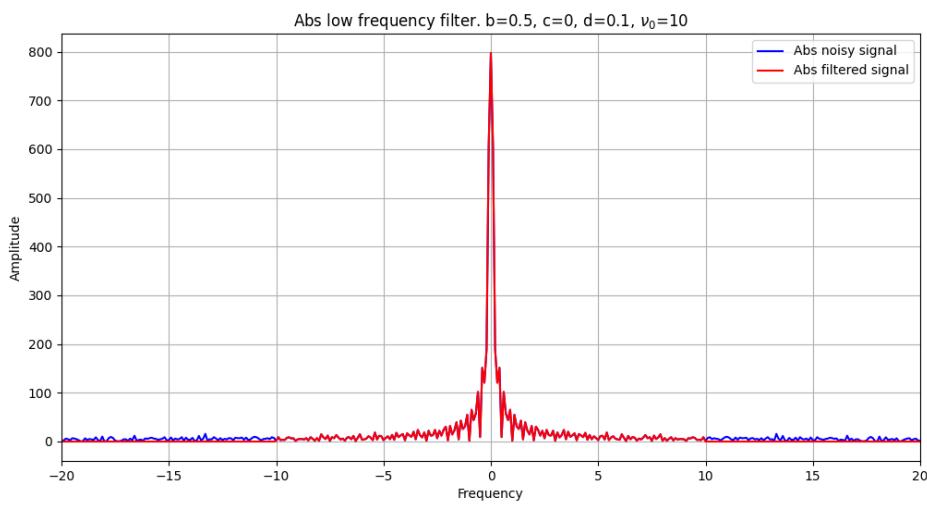


Рис. 2: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

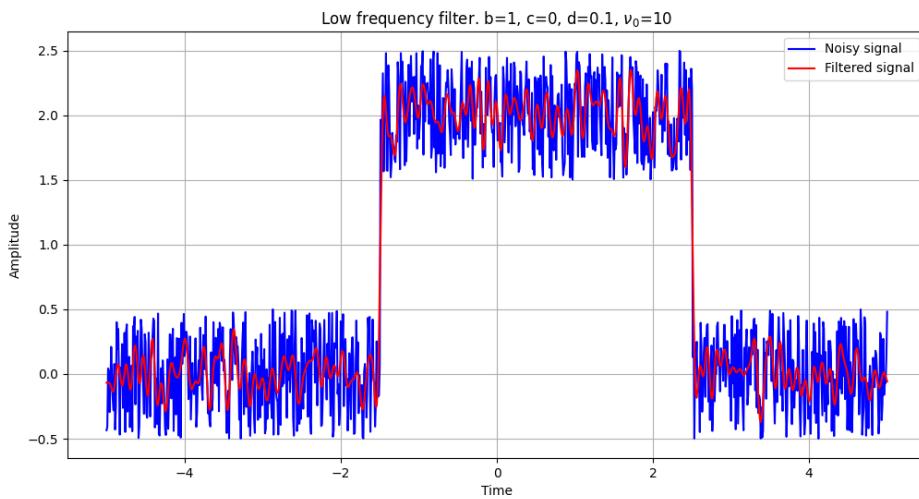


Рис. 3: График исходного и фильтрованного сигналов (2)

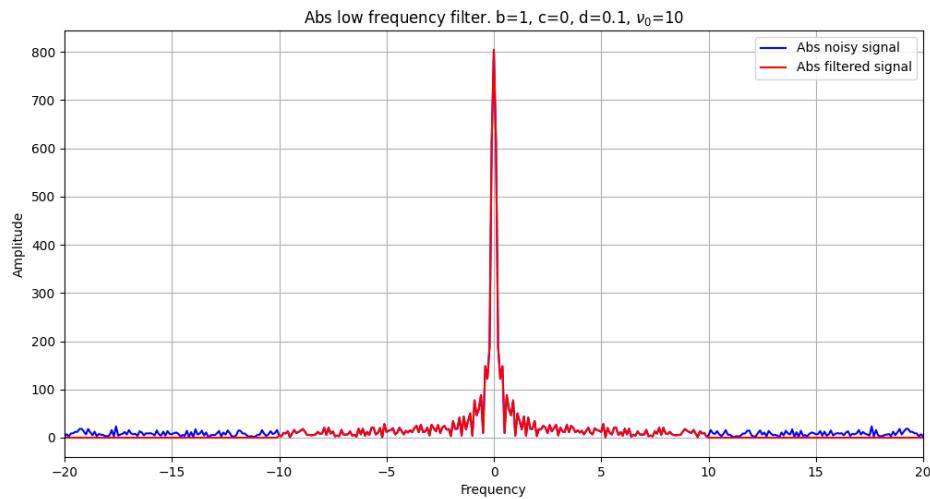


Рис. 4: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

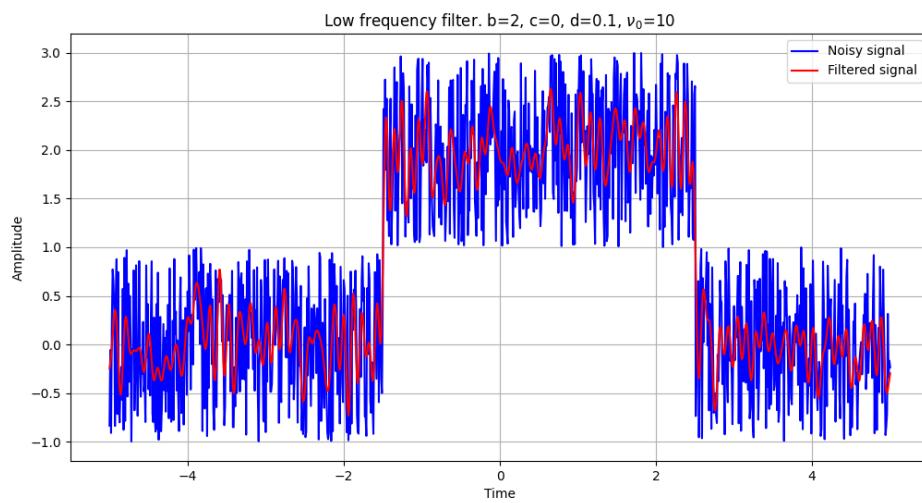


Рис. 5: График исходного и фильтрованного сигналов (3)

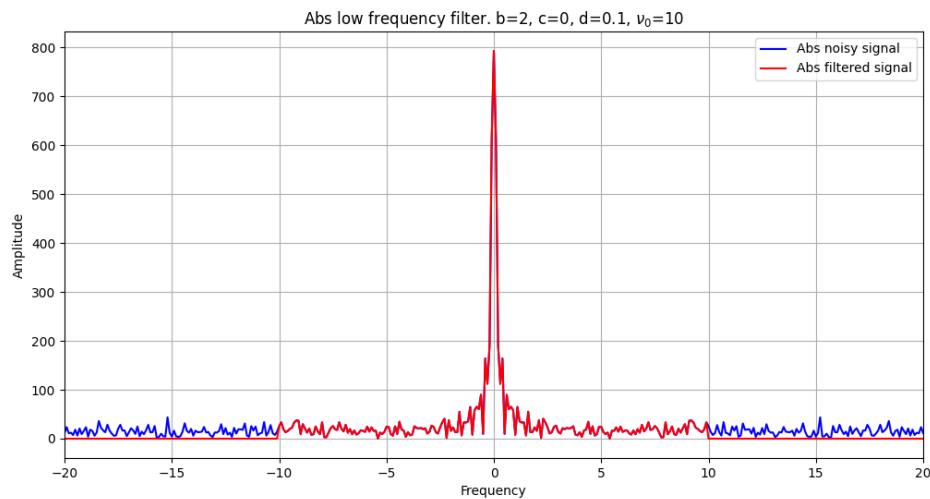


Рис. 6: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

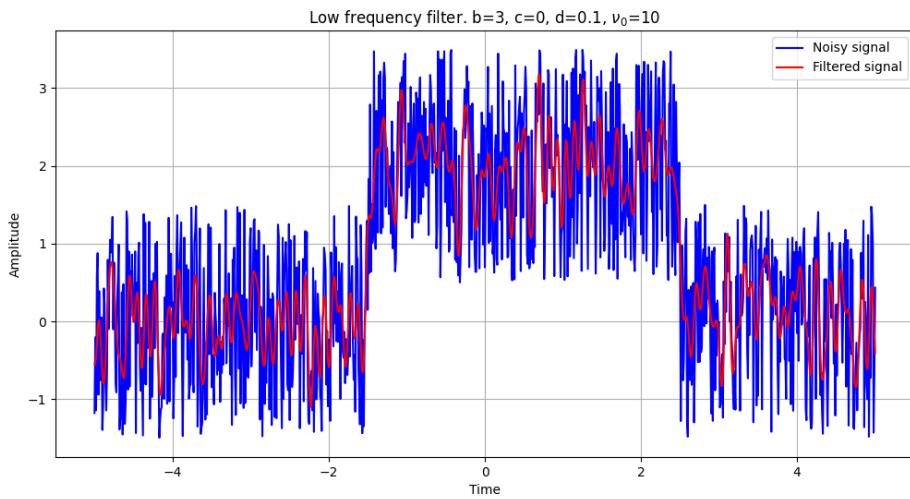


Рис. 7: График исходного и фильтрованного сигналов (4)

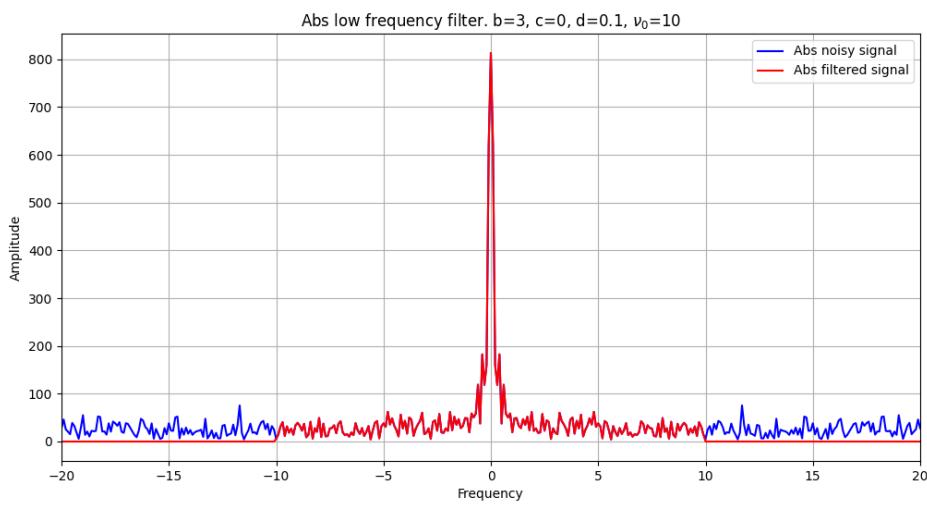


Рис. 8: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

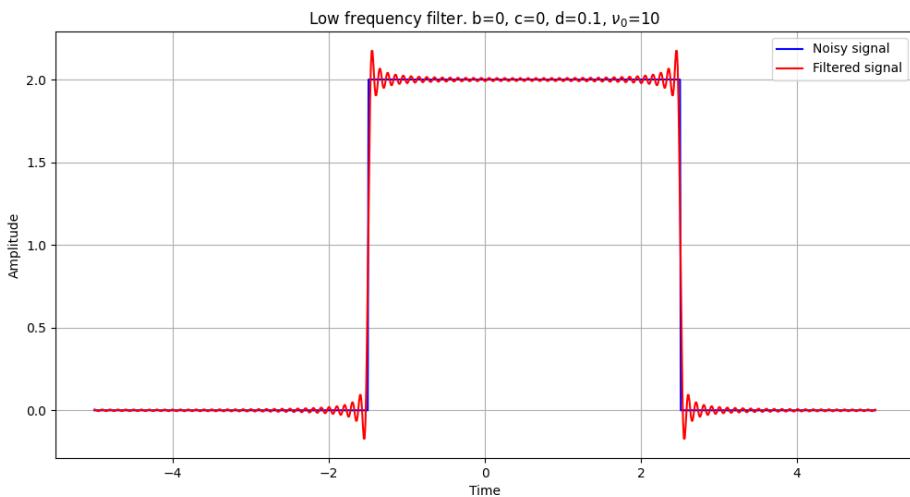


Рис. 9: График исходного и фильтрованного сигналов (5)

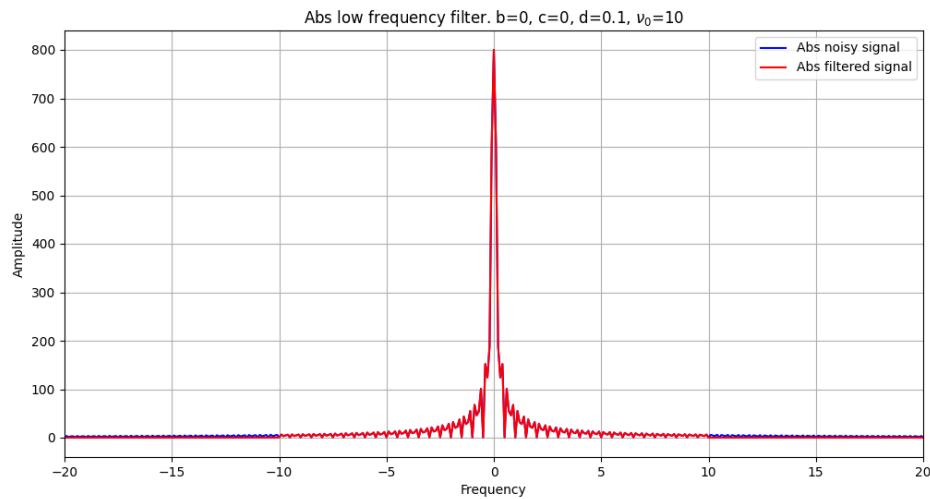


Рис. 10: График модуля Фурье-образа исходного и фильтрованного сигналов (5)

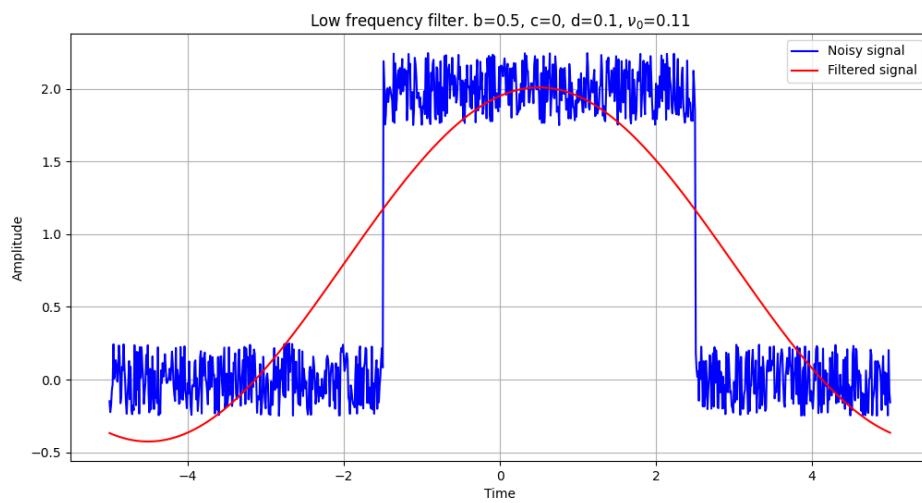


Рис. 11: График исходного и фильтрованного сигналов (6)

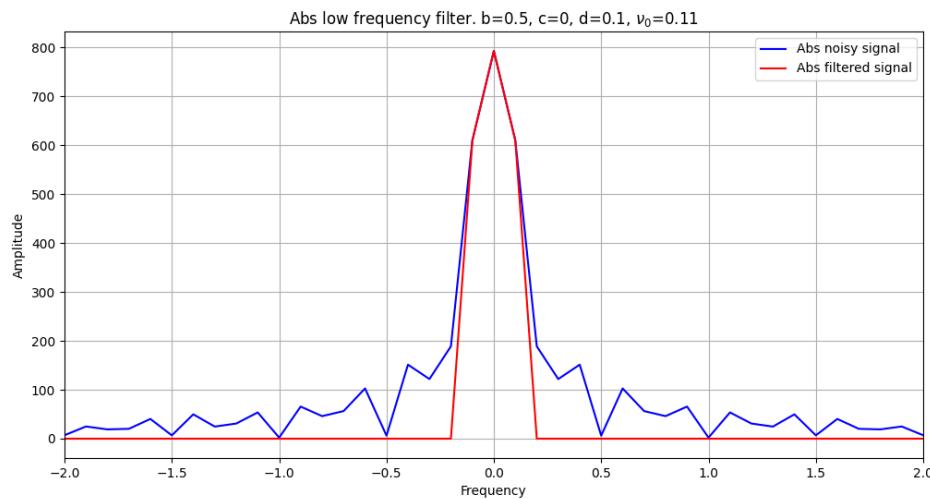


Рис. 12: График модуля Фурье-образа исходного и фильтрованного сигналов (6)

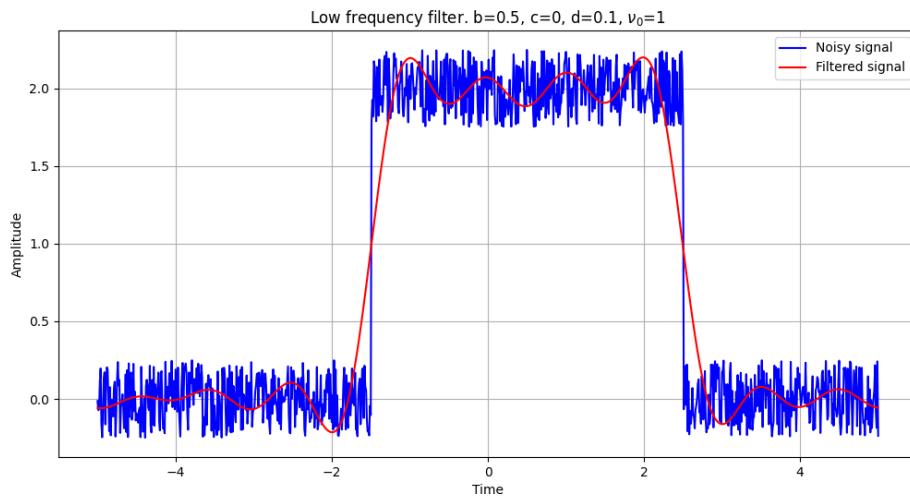


Рис. 13: График исходного и фильтрованного сигналов (7)

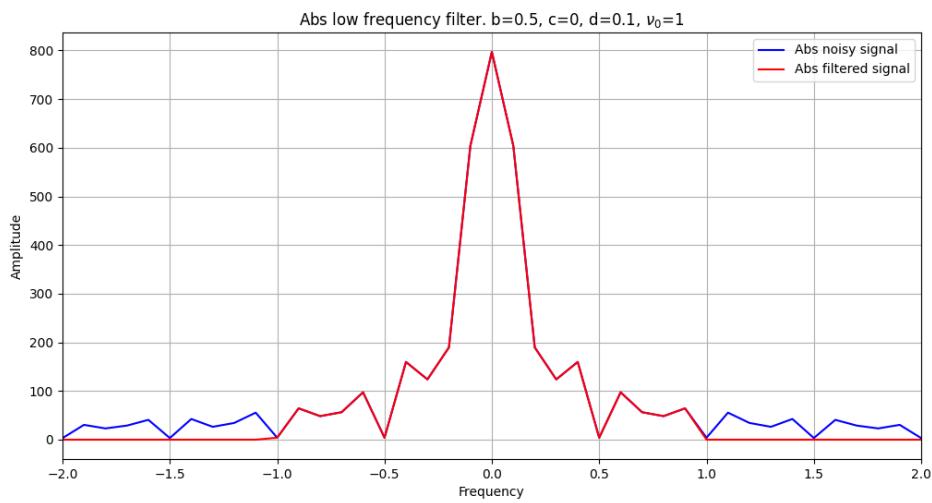


Рис. 14: График модуля Фурье-образа исходного и фильтрованного сигналов (7)

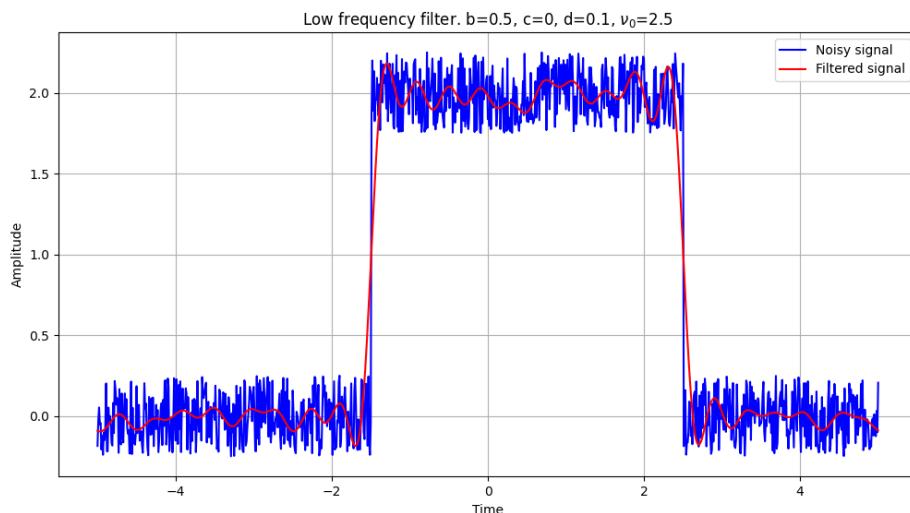


Рис. 15: График исходного и фильтрованного сигналов (8)

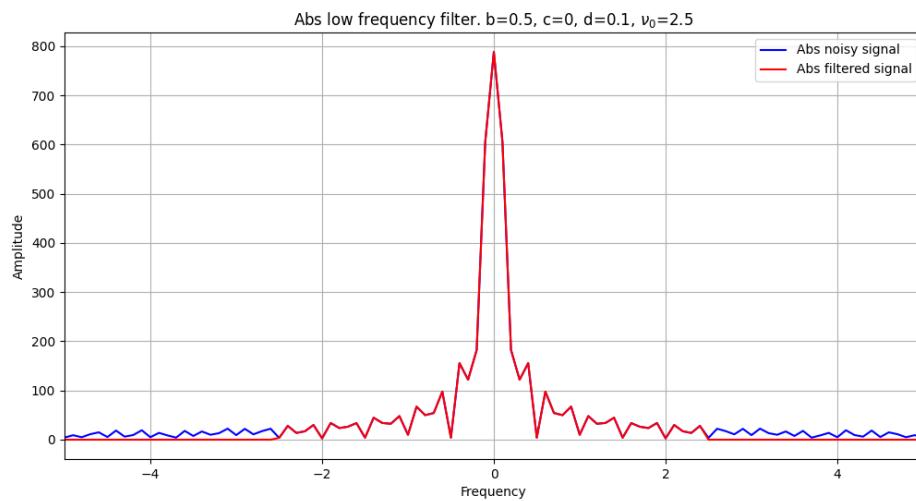


Рис. 16: График модуля Фурье-образа исходного и фильтрованного сигналов (8)

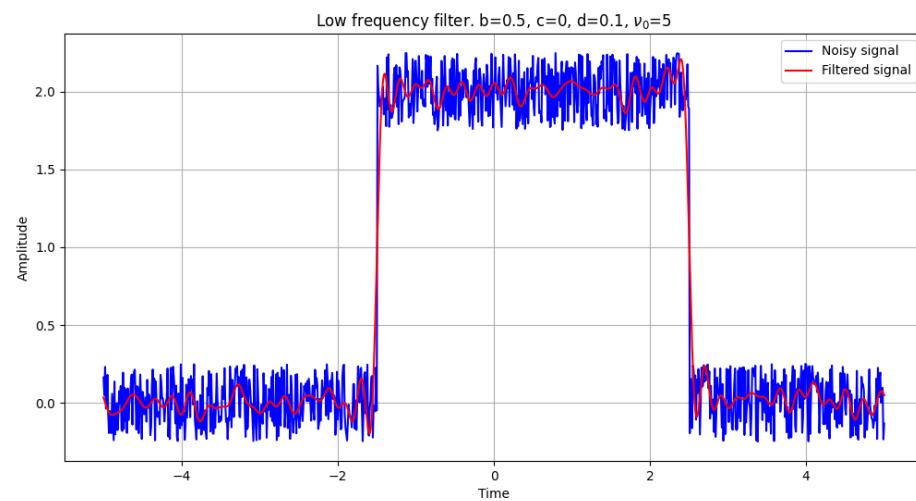


Рис. 17: График исходного и фильтрованного сигналов (9)

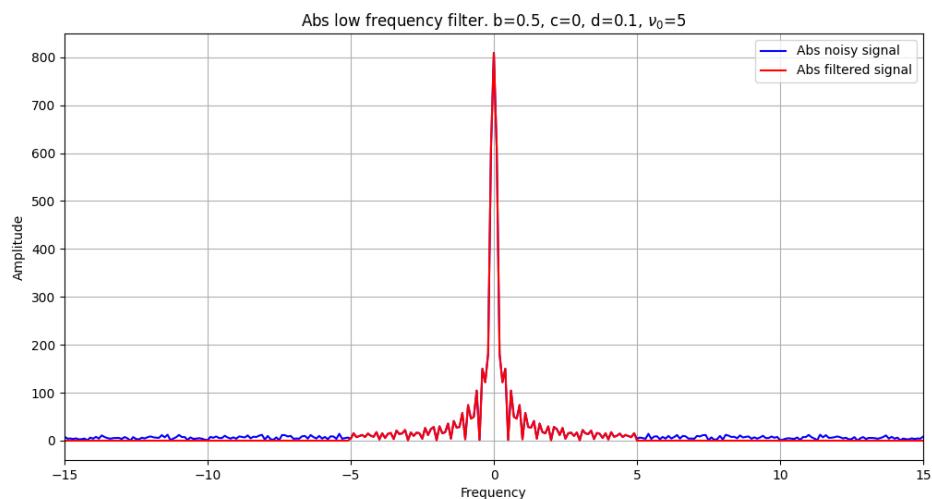


Рис. 18: График модуля Фурье-образа исходного и фильтрованного сигналов (9)

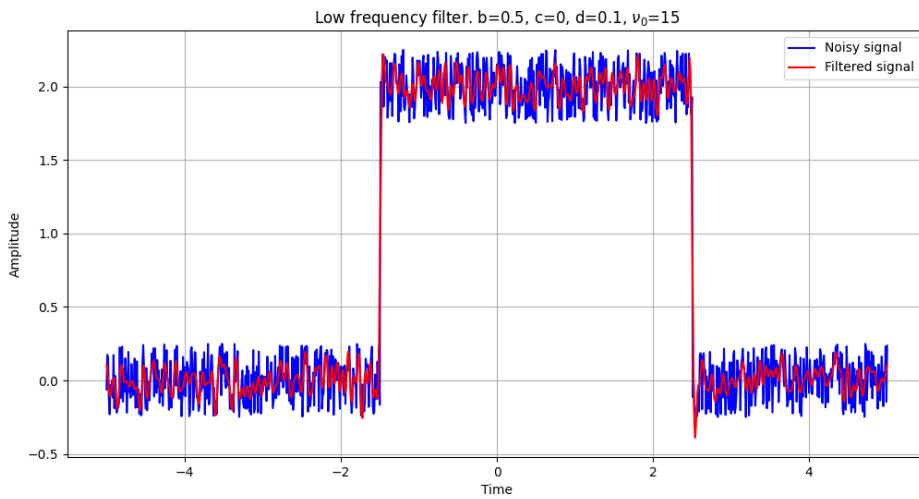


Рис. 19: График исходного и фильтрованного сигналов (10)

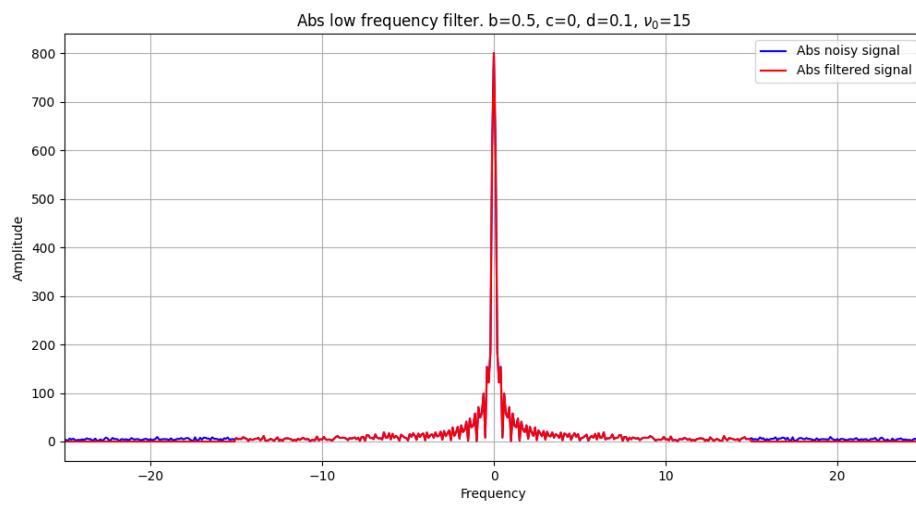


Рис. 20: График модуля Фурье-образа исходного и фильтрованного сигналов (10)

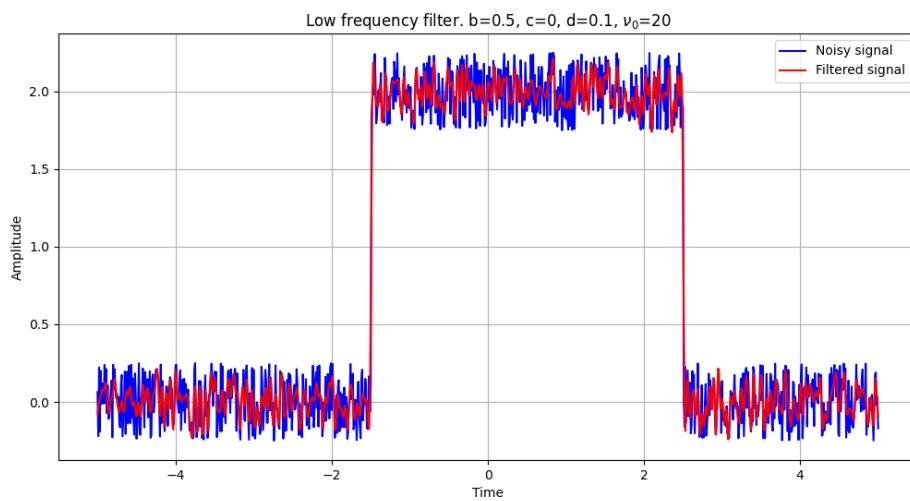


Рис. 21: График исходного и фильтрованного сигналов (11)

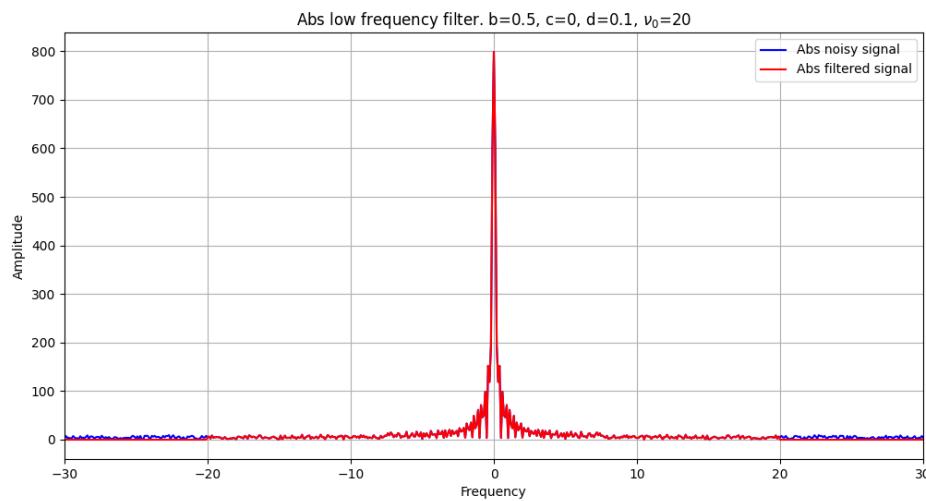


Рис. 22: График модуля Фурье-образа исходного и фильтрованного сигналов (11)

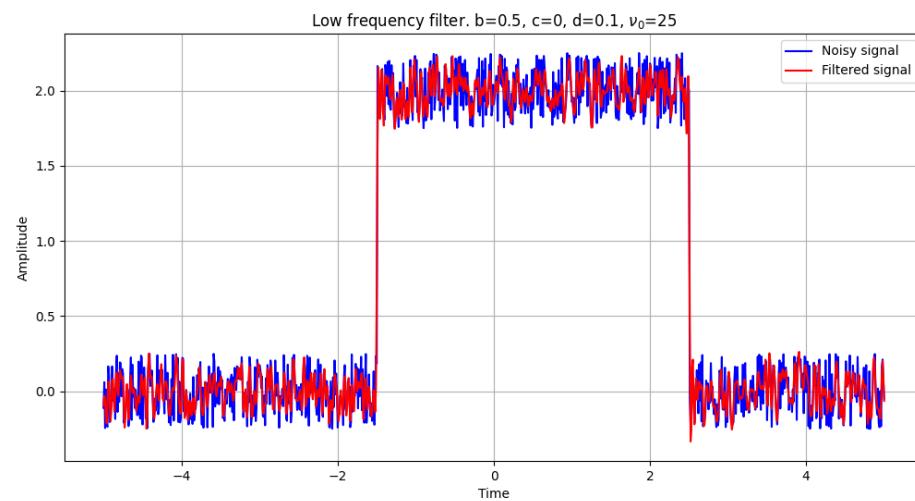


Рис. 23: График исходного и фильтрованного сигналов (12)

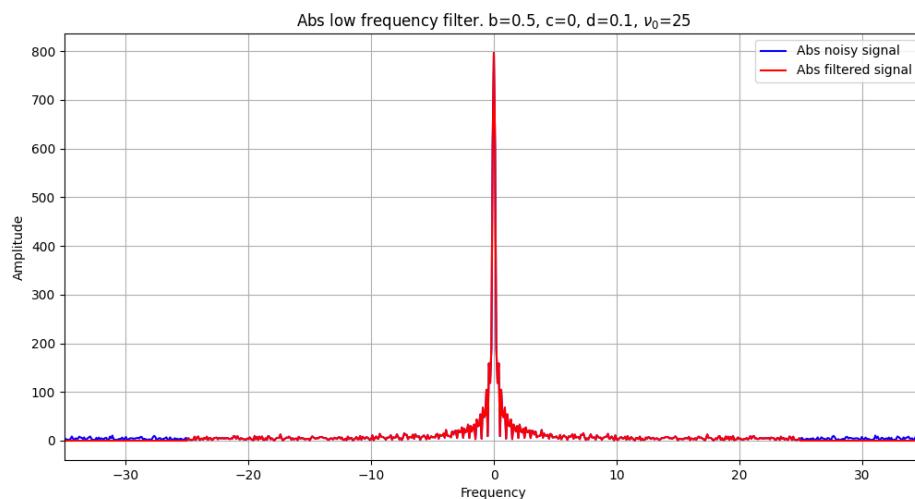


Рис. 24: График модуля Фурье-образа исходного и фильтрованного сигналов (12)

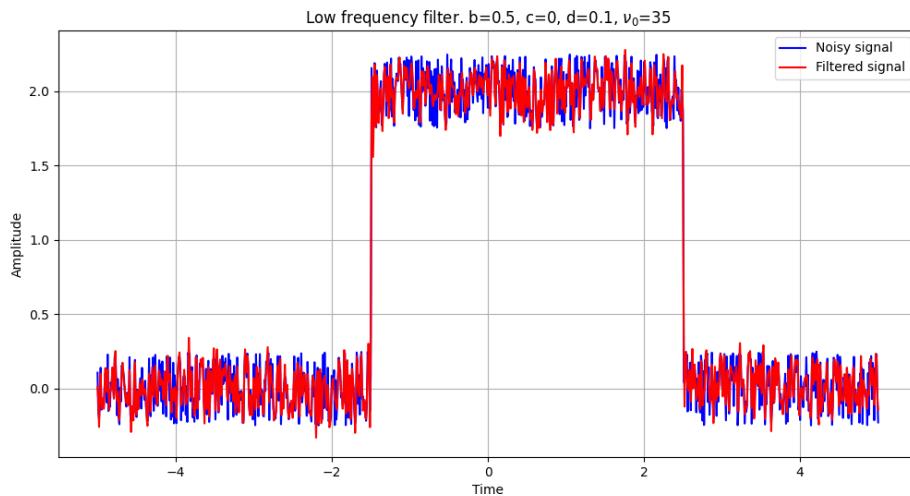


Рис. 25: График исходного и фильтрованного сигналов (13)

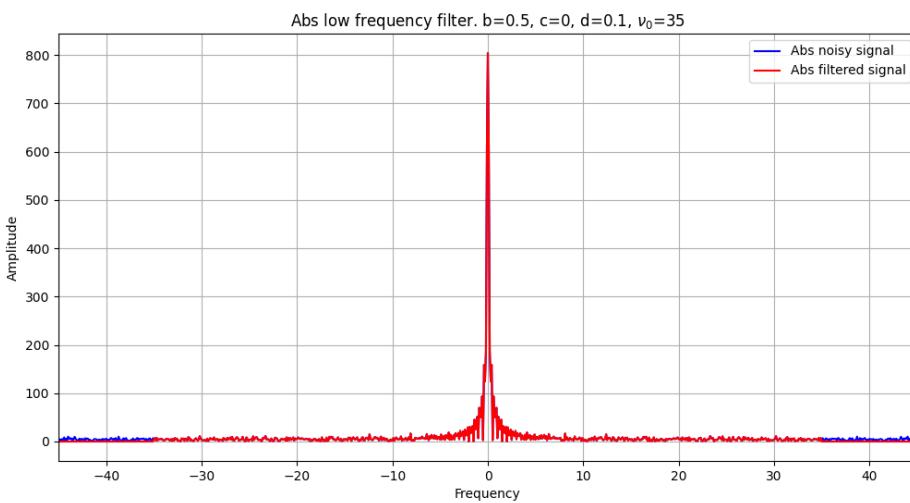


Рис. 26: График модуля Фурье-образа исходного и фильтрованного сигналов (13)

меньше значение b , тем чище сигнал и тем лучше фильтрация. При большом количестве белого шума фильтрация ухудшается (см. рис. 7). При $b = 0$ получается особый случай — сигнал превращается в прямоугольную функцию, а фильтрованный сигнал выглядит как ее аппроксимация.

Больше всего влияния на эффективность фильтрации оказывает частота среза ν_0 . Этот параметр необходимо подобрать так, чтобы оставались только те частоты, которые имеют заметно более высокую амплитуду по сравнению с остальными. Чем меньше частота среза ν_0 , тем заметно лучше фильтрация. Однако если взять слишком маленькое значение ν_0 , то фильтрация будет слишком сильной, и мы потеряем большую часть значащих частот в сигнале, что можно увидеть на рисунках 11, 12. Это происходит потому, что мы задели и нижние частоты тоже, оставив лишь малую их часть. Можно сказать, что ν_0 — это порог, которым мы определяем, является ли частота верхней или нижней. Такой параметр нужно подбирать с умом, чтобы получить от фильтрации ожидаемый результат. Если взять слишком большое значение ν_0 , то фильтрация будет не очень эффективной (см. рис. 25, 26), так как мы оставили некоторые верхние частоты, которые можно было обнулить.

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 – там содержится наибольшее количество информации об исходном сигнале. Обнулим этот высокий дополнительный пик 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.

После анализа графиков можно сделать вывод, что параметр c отвечает за синусоидальный шум. Чем больше значение параметра c , тем больше растягивается сигнал по оси Oy , то есть сильнее возрастает амплитуда волн. При $b = 0$ сигнал u описывается в формулой синусоиды

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

(см. рис. 31, 35). Чем больше g , тем выше поднимается график. Параметр d характеризует растяжение графика по оси Ox . При увеличении значения параметра d частота волн повышается. Значение параметра b как и в пунктах ранее влияет на загрязненность сигнала – чем меньше b , тем чище исходный и фильтрованный сигналы. Как виды помехи параметры можно назвать так: b – белый шум, c и d гармонический шум. Исследовав влияние частот среза можно сказать, что обнулять частоты пиков слева и справа от наивысшей(их) амплитуд(ы) недостаточно.

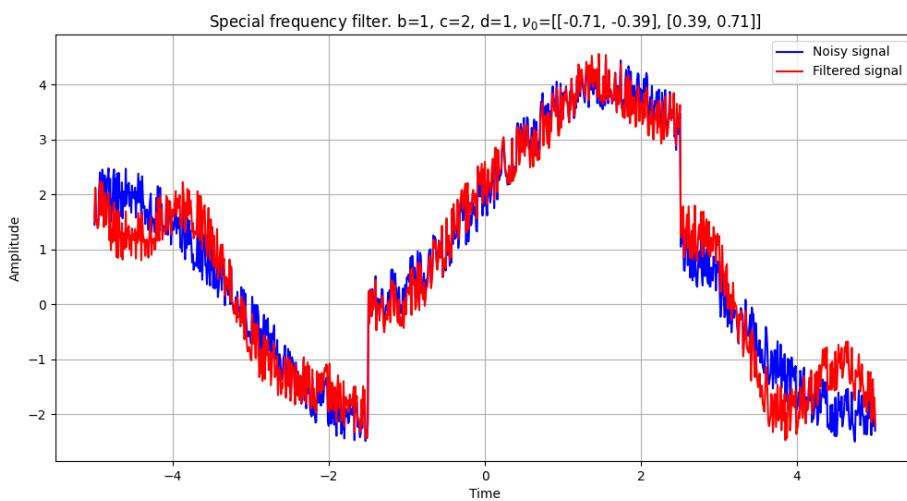


Рис. 27: График исходного и фильтрованного сигналов (1)

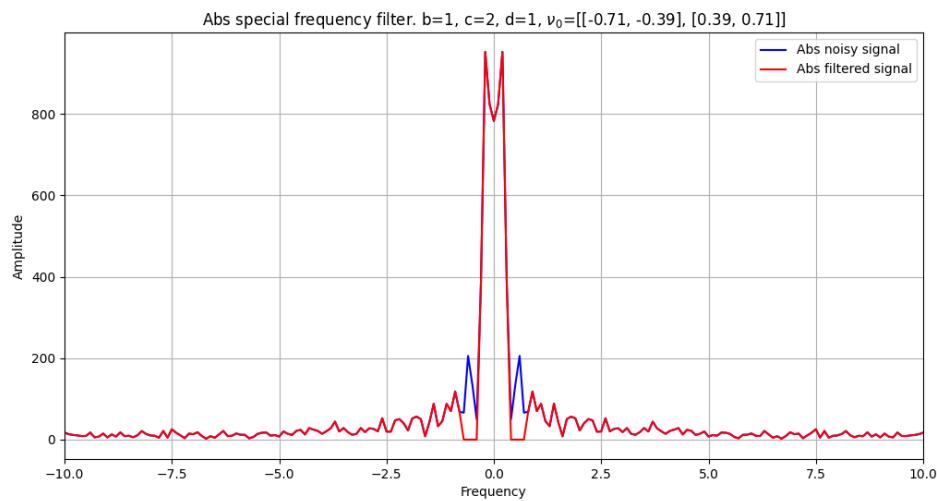


Рис. 28: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

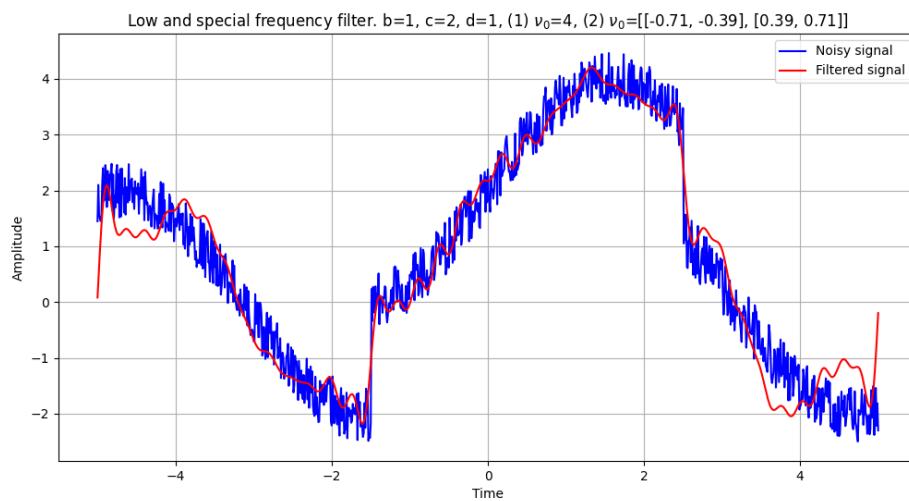


Рис. 29: График исходного и фильтрованного сигналов (1)

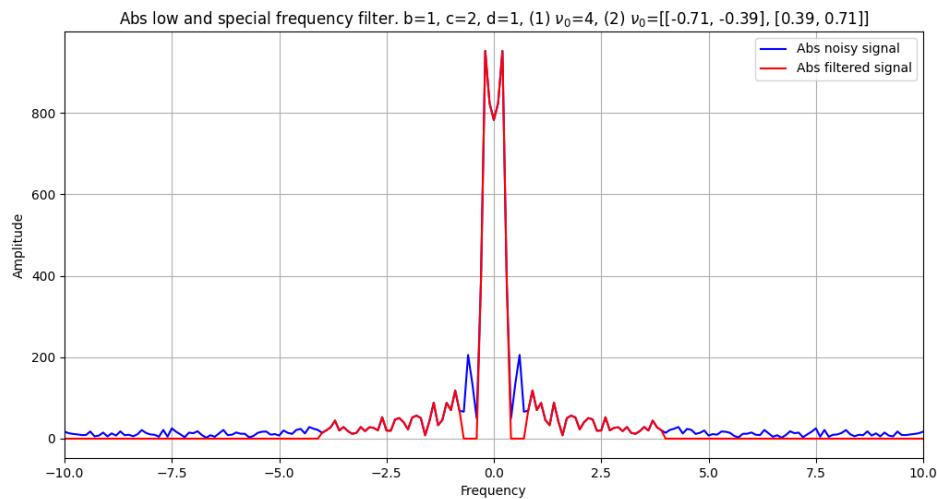


Рис. 30: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

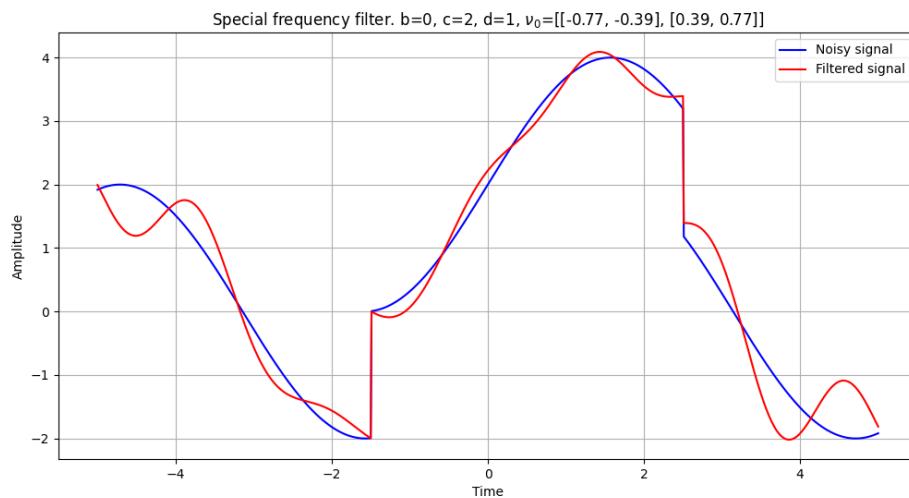


Рис. 31: График исходного и фильтрованного сигналов (2)

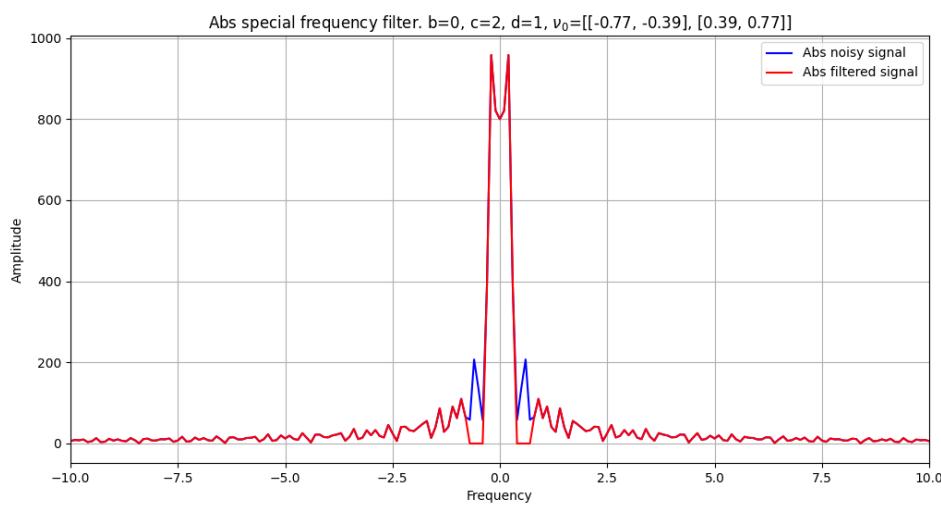


Рис. 32: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

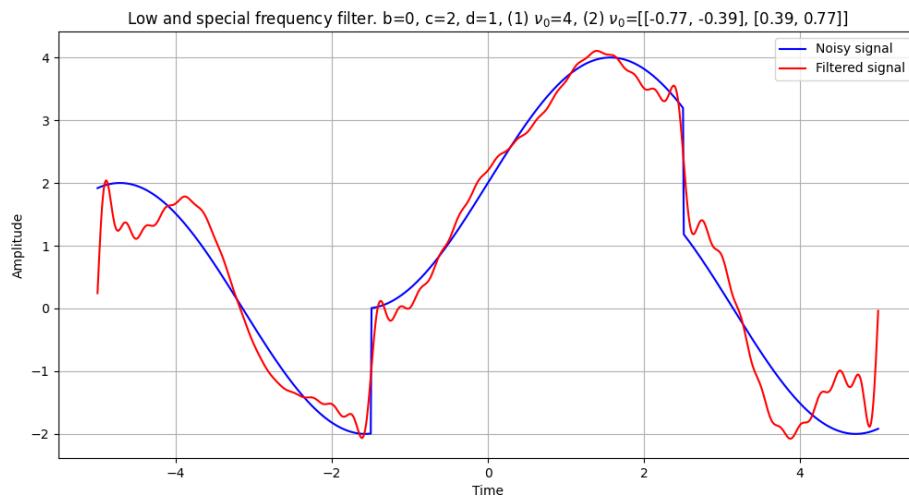


Рис. 33: График исходного и фильтрованного сигналов (2)

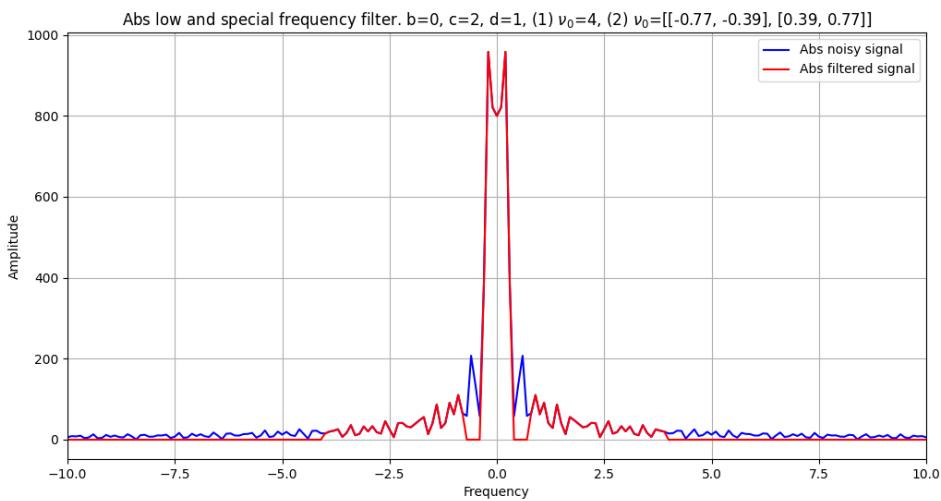


Рис. 34: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

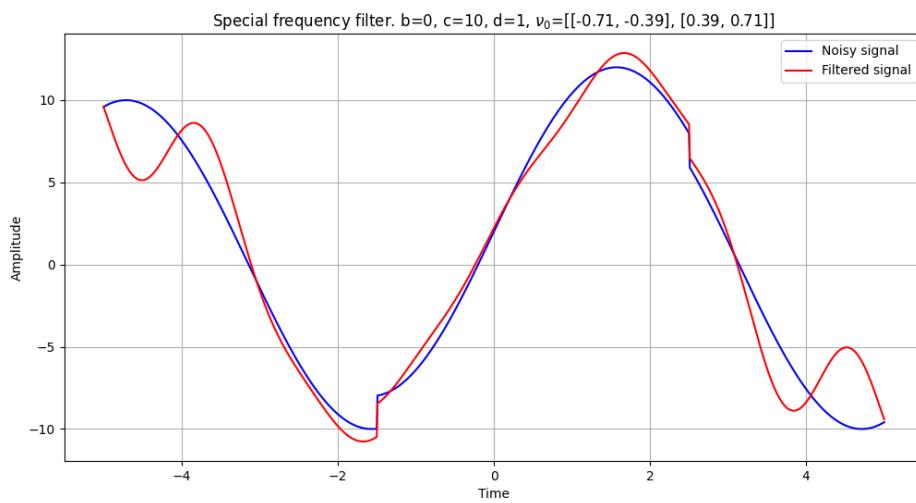


Рис. 35: График исходного и фильтрованного сигналов (3)

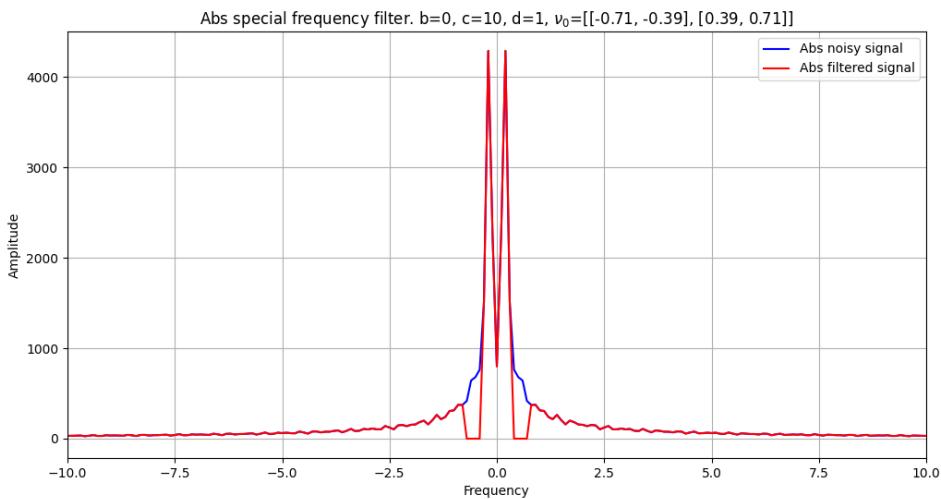


Рис. 36: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

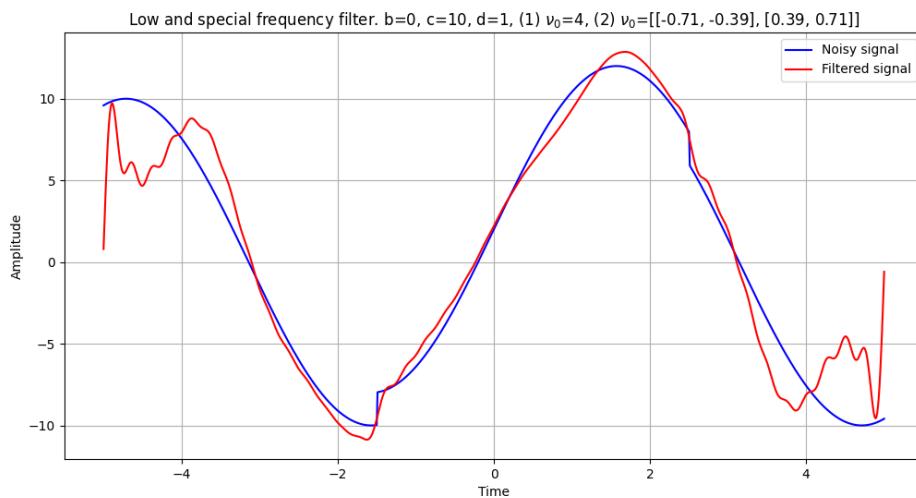


Рис. 37: График исходного и фильтрованного сигналов (3)

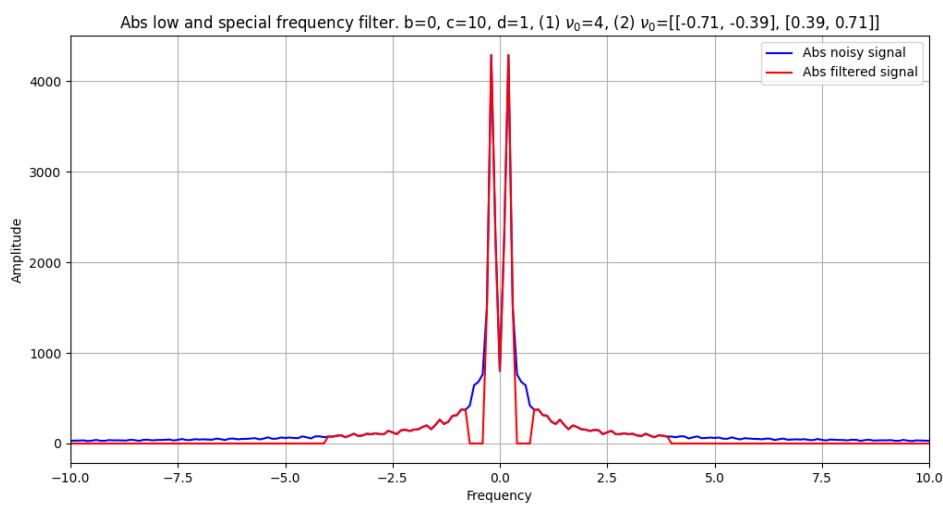


Рис. 38: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

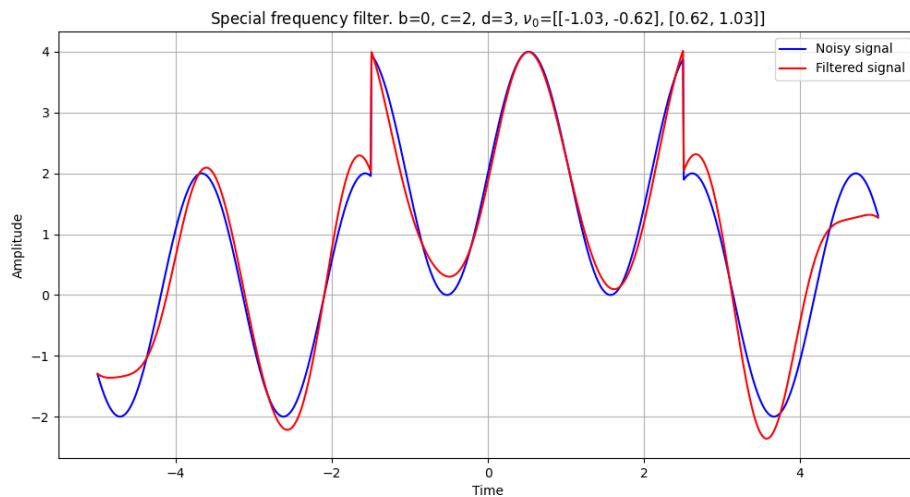


Рис. 39: График исходного и фильтрованного сигналов (4)

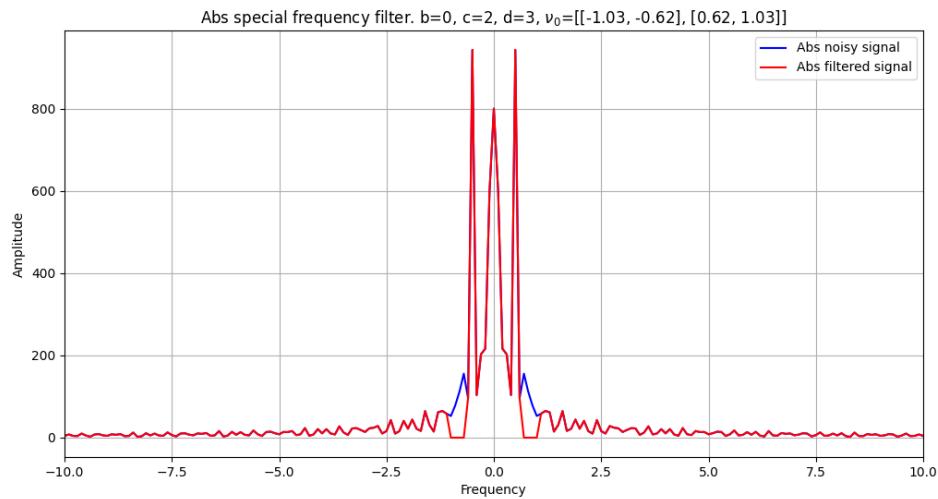


Рис. 40: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

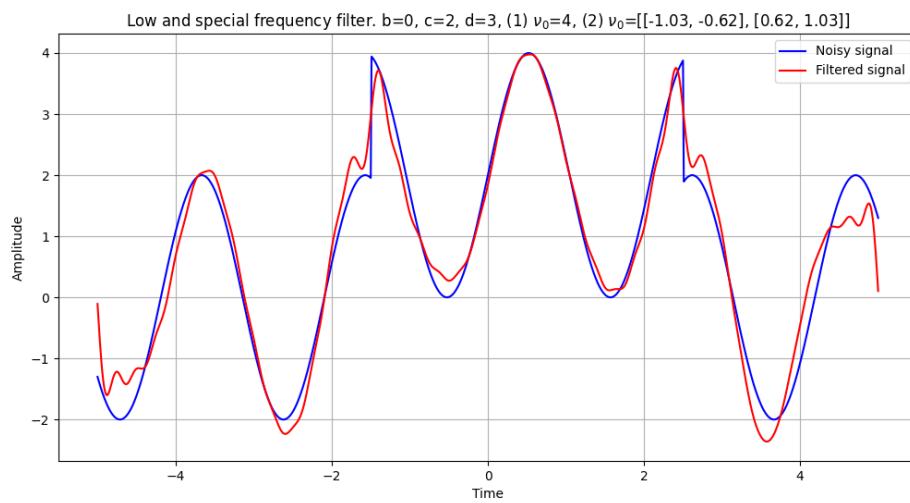


Рис. 41: График исходного и фильтрованного сигналов (4)

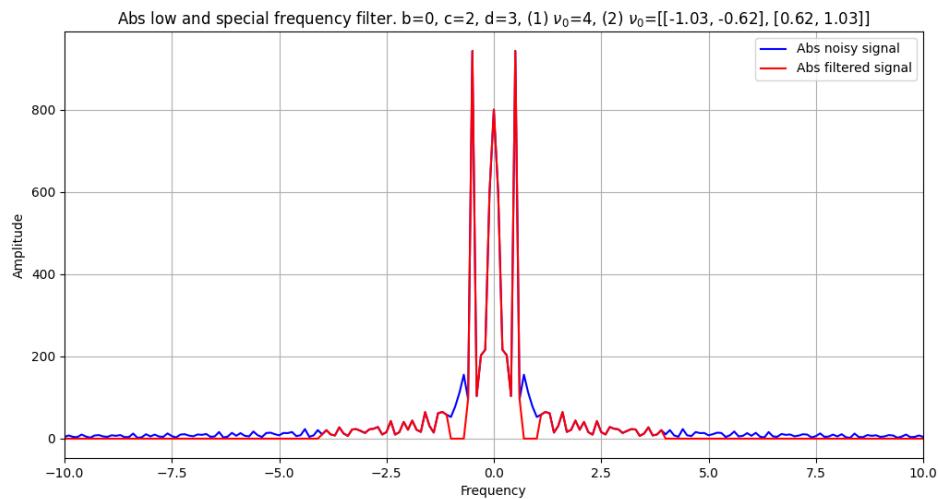


Рис. 42: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

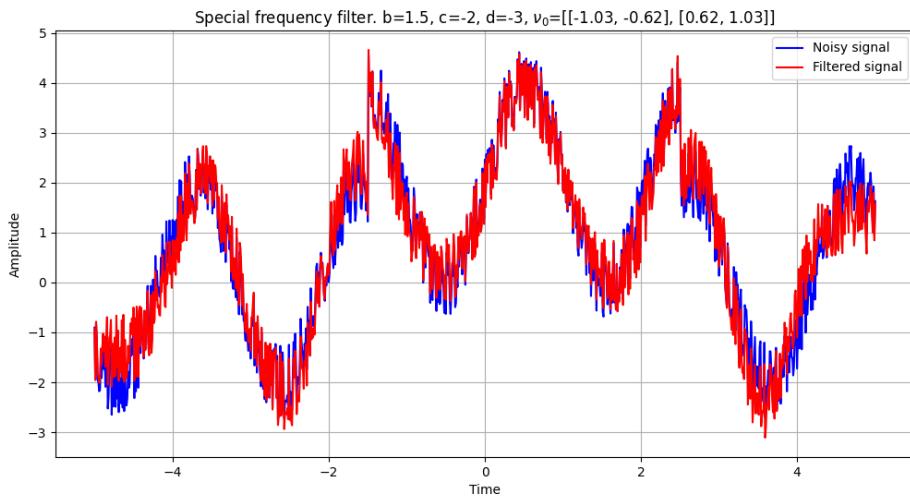


Рис. 43: График исходного и фильтрованного сигналов (5)

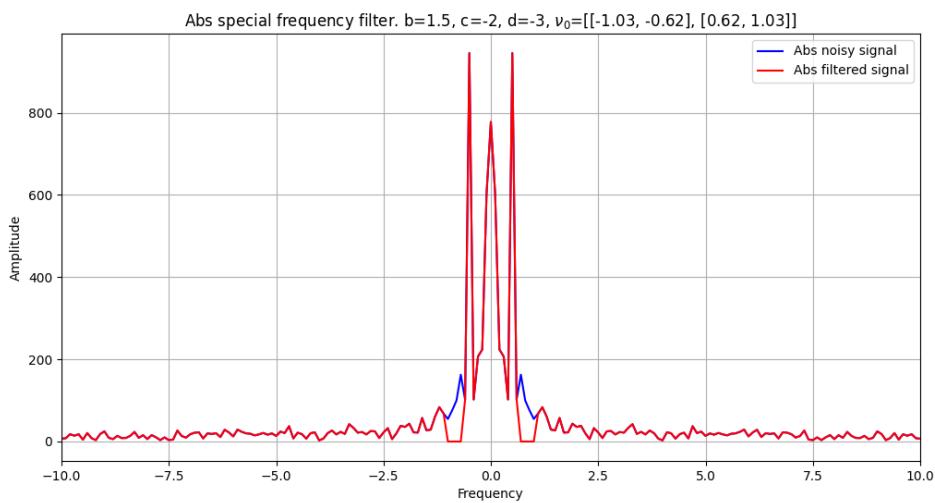


Рис. 44: График модуля Фурье-образа исходного и фильтрованного сигналов (5)

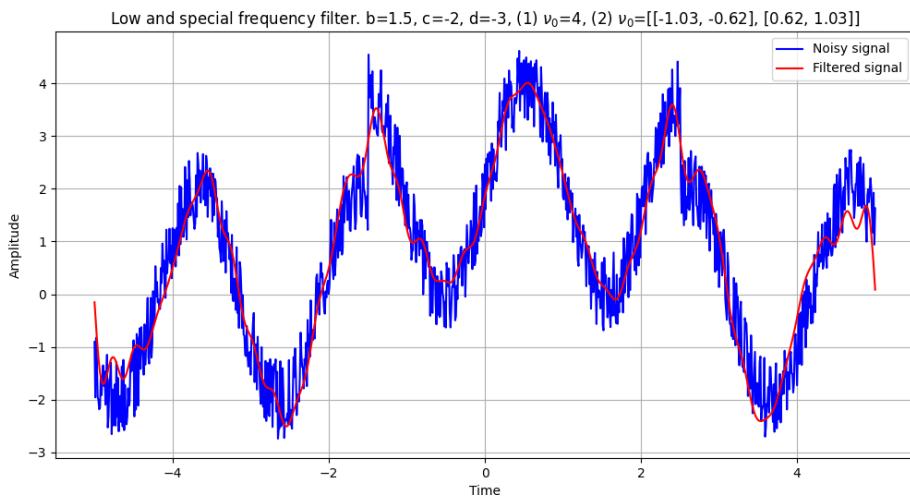


Рис. 45: График исходного и фильтрованного сигналов (5)

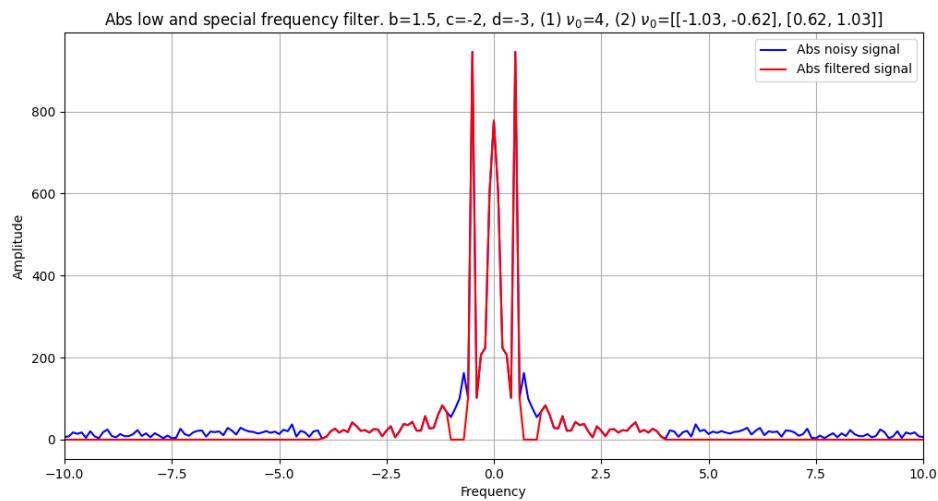


Рис. 46: График модуля Фурье-образа исходного и фильтрованного сигналов (5)

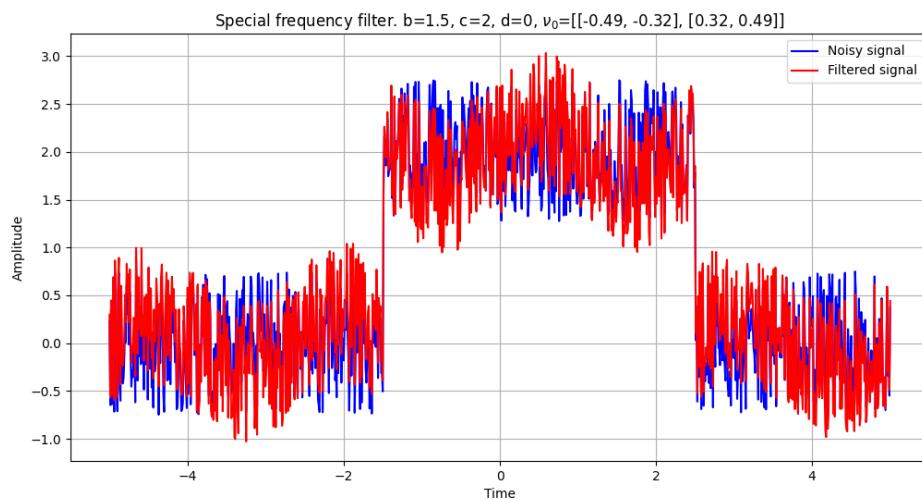


Рис. 47: График исходного и фильтрованного сигналов (6)

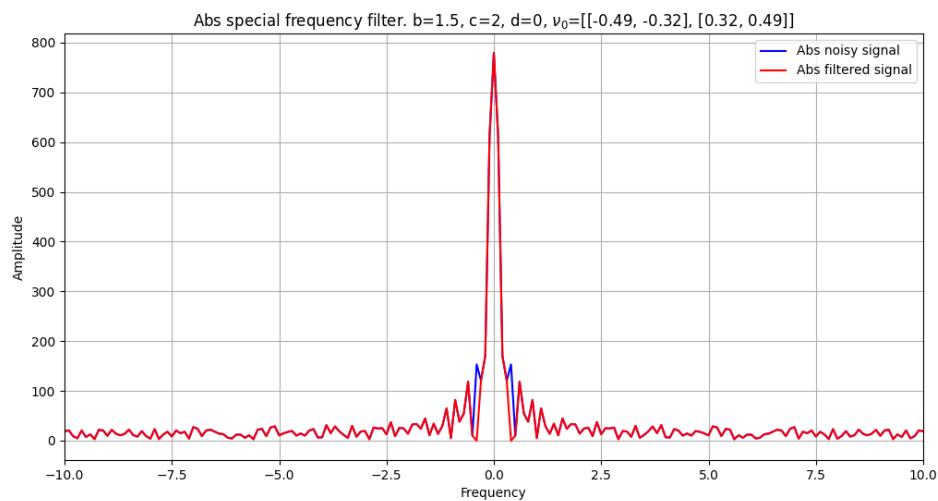


Рис. 48: График модуля Фурье-образа исходного и фильтрованного сигналов (6)

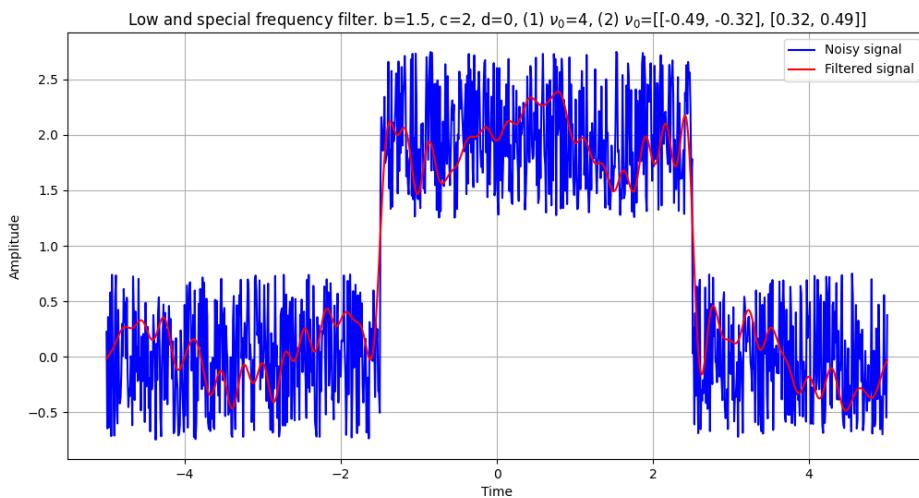


Рис. 49: График исходного и фильтрованного сигналов (6)

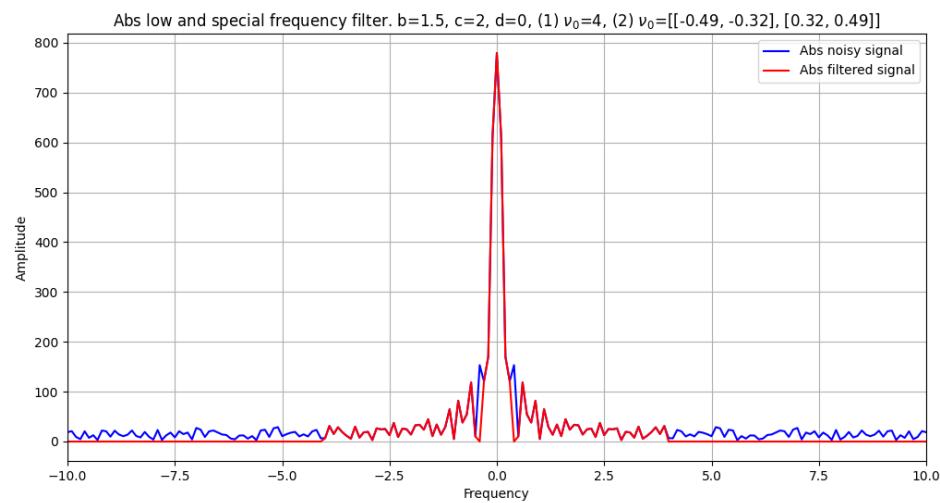


Рис. 50: График модуля Фурье-образа исходного и фильтрованного сигналов (6)

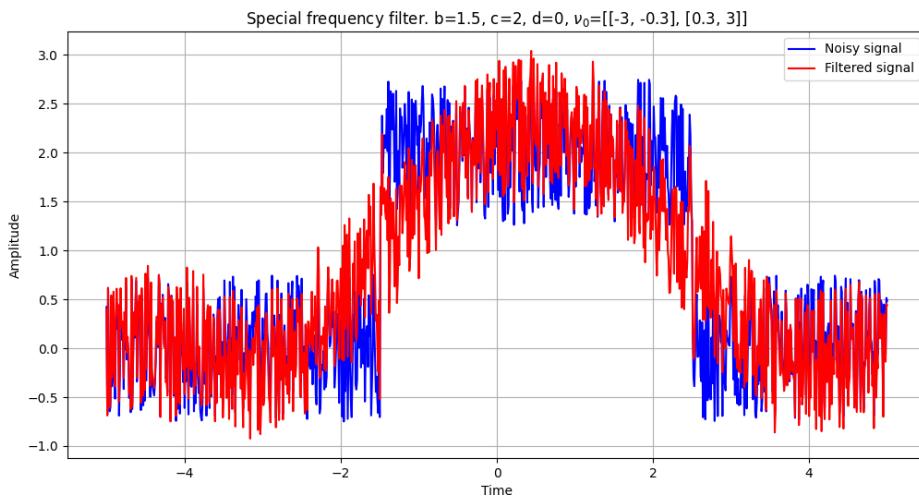


Рис. 51: График исходного и фильтрованного сигналов (7)

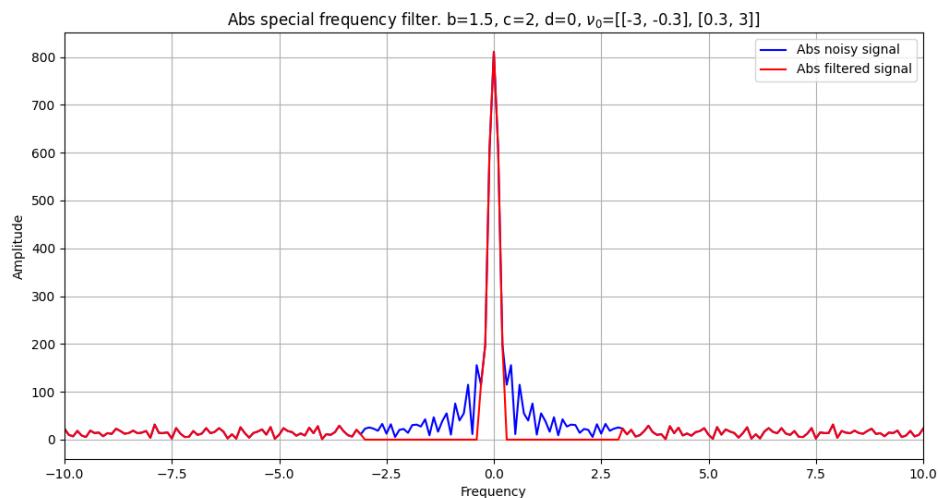


Рис. 52: График модуля Фурье-образа исходного и фильтрованного сигналов (7)

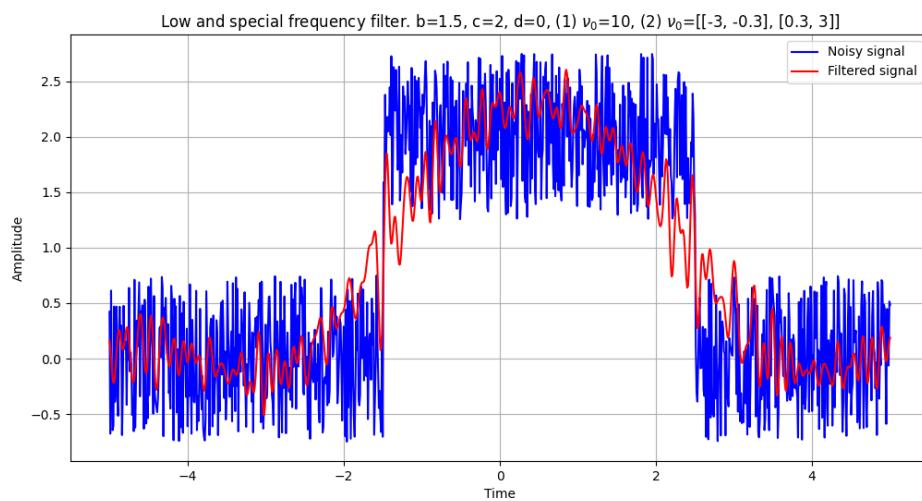


Рис. 53: График исходного и фильтрованного сигналов (7)

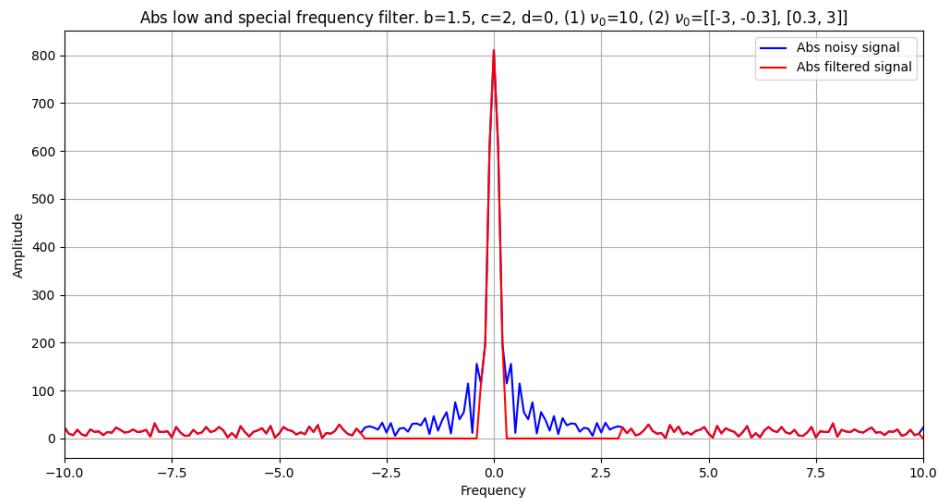


Рис. 54: График модуля Фурье-образа исходного и фильтрованного сигналов (7)

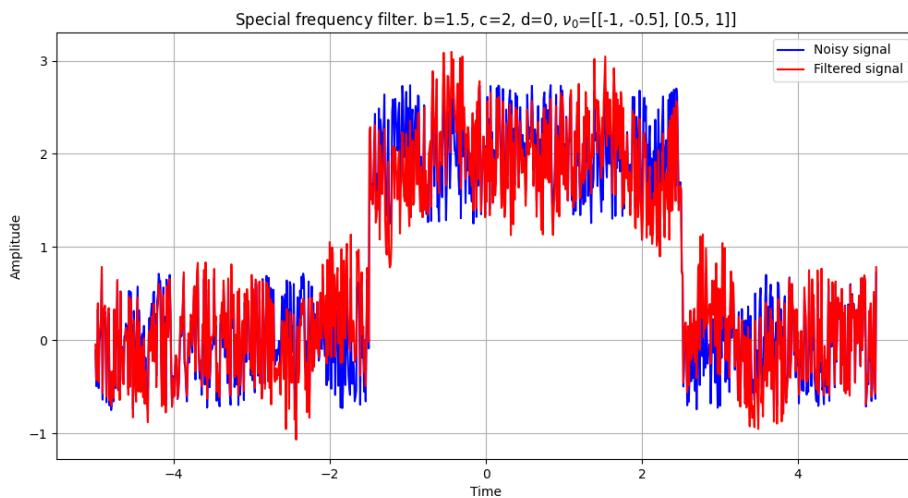


Рис. 55: График исходного и фильтрованного сигналов (8)

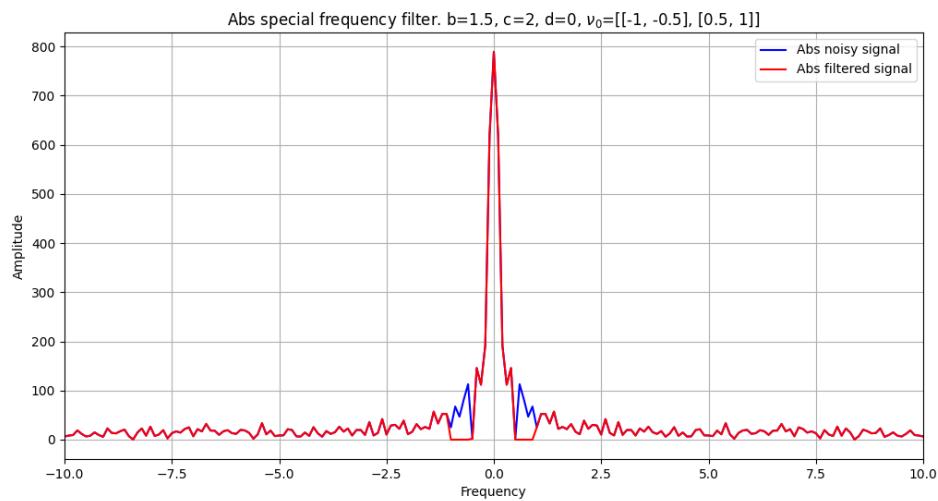


Рис. 56: График модуля Фурье-образа исходного и фильтрованного сигналов (8)

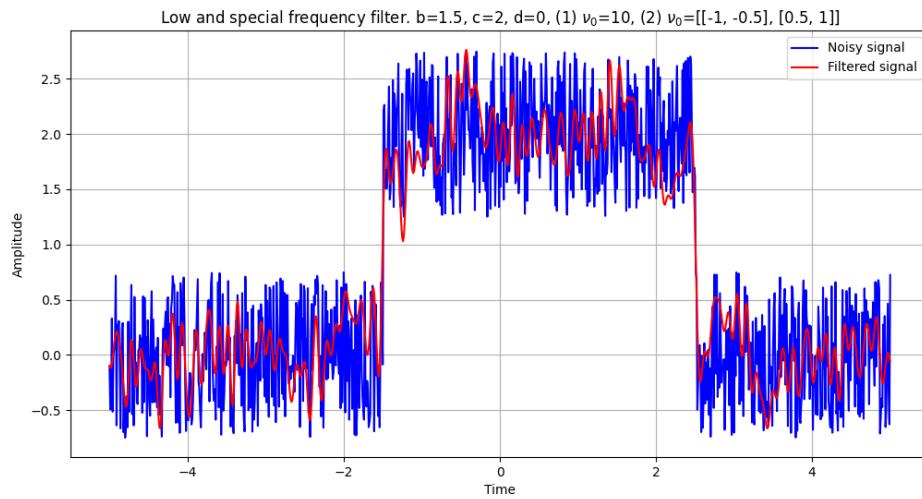


Рис. 57: График исходного и фильтрованного сигналов (8)

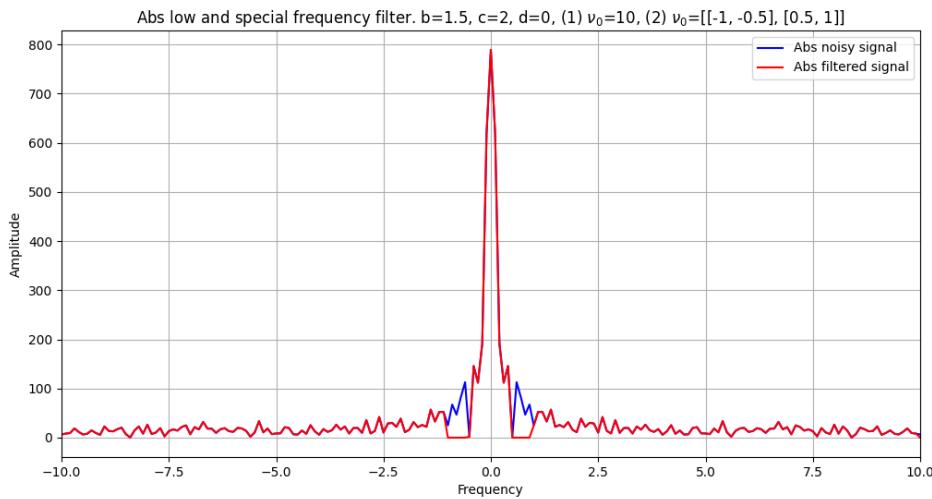


Рис. 58: График модуля Фурье-образа исходного и фильтрованного сигналов (8)

1.3 Убираем низкие частоты. Фильтр верхних частот.

Рассмотрим графики, где в некоторой окрестности точки $\nu = 0$ обнулим все значения частот Фурье-образа. Такое поведение соответствует фильтру *верхних* частот, так как он пропускает все частоты выше частоты среза. Окрестность будет настраиваться выбором диапазона частот $[-\nu_0, \nu_0]$. Для лучшей показательности выбраны как маленькие окрестности около нуля, так и большие. Также рассмотрим поведение фильтрации при различных параметрах b, c, d .

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

По полученным графикам можно сделать вывод — фильтр верхних частот в данном случае отрезает значимую часть сигнала, содержащуюся в низких частотах, и оставляет только белый и/или гармонический шум, находящийся преимущественно на высоких частотах.

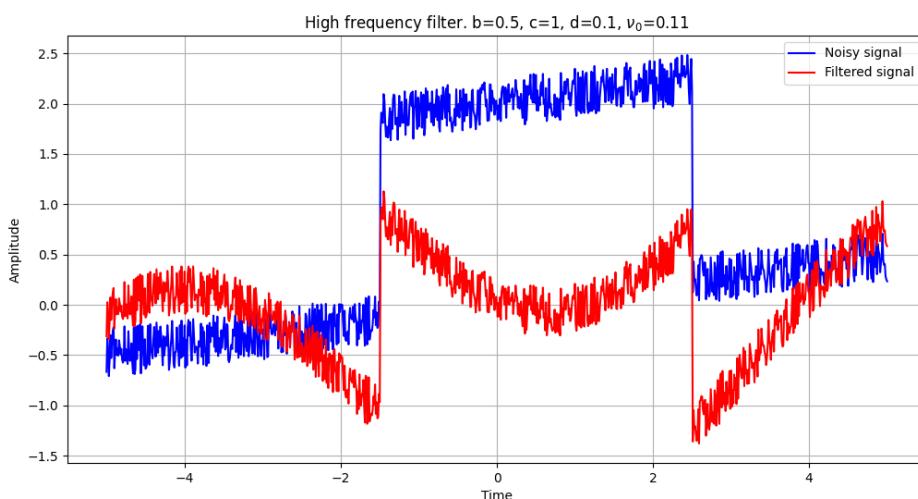


Рис. 59: График исходного и фильтрованного сигналов (1)

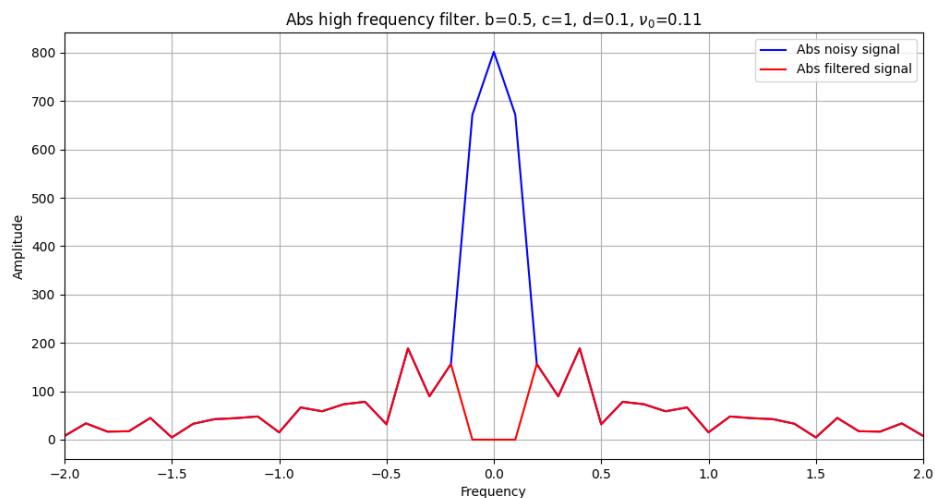


Рис. 60: График модуля Фурье-образа исходного и фильтрованного сигналов (1)

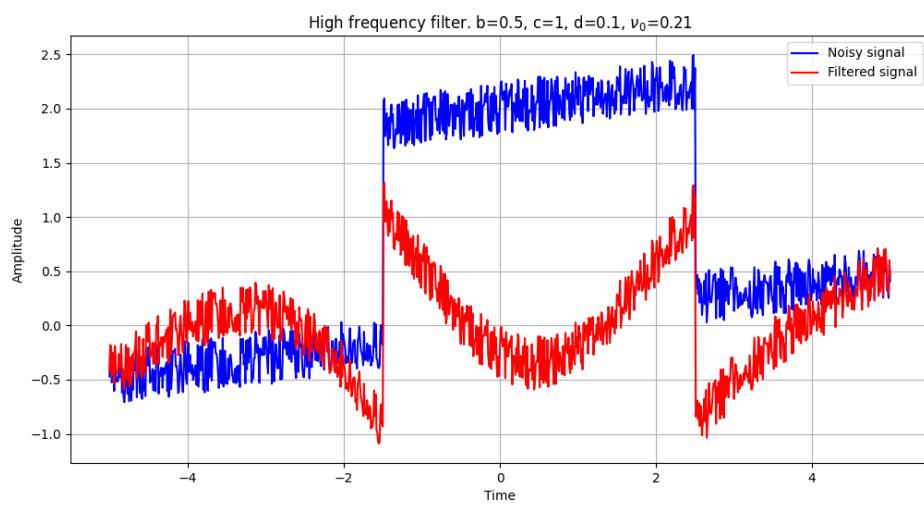


Рис. 61: График исходного и фильтрованного сигналов (2)

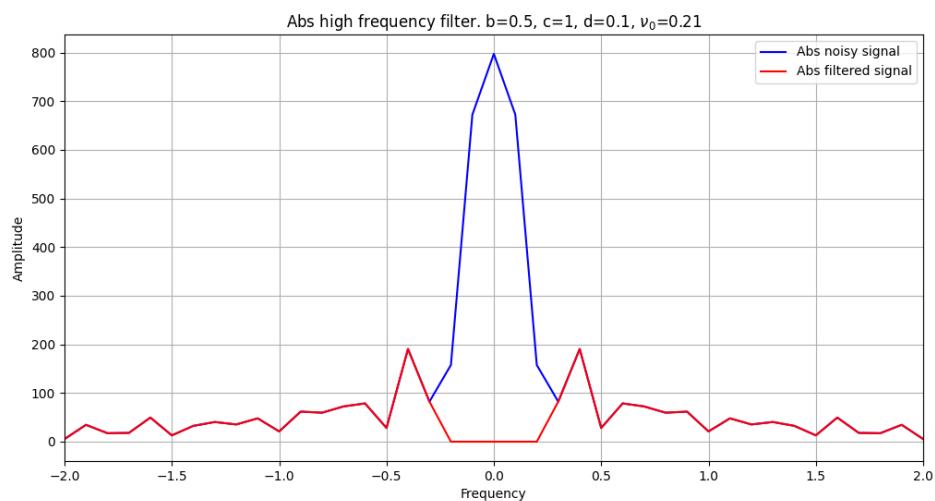


Рис. 62: График модуля Фурье-образа исходного и фильтрованного сигналов (2)

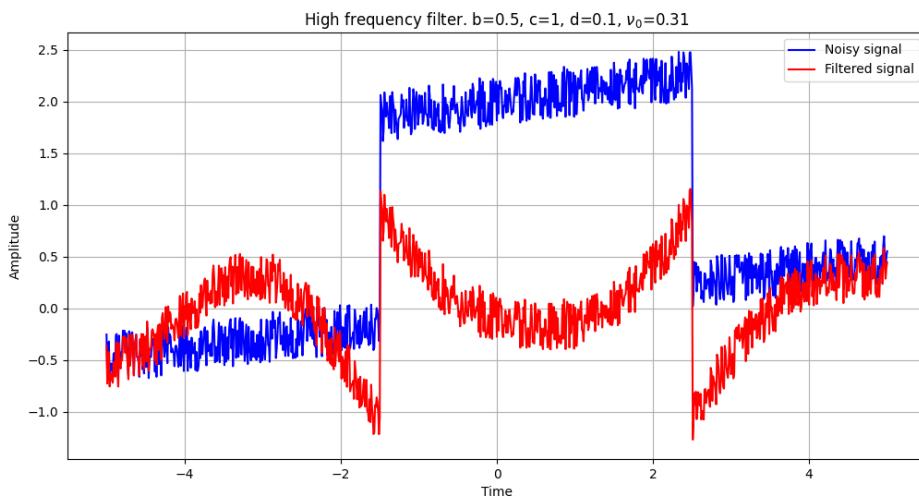


Рис. 63: График исходного и фильтрованного сигналов (3)

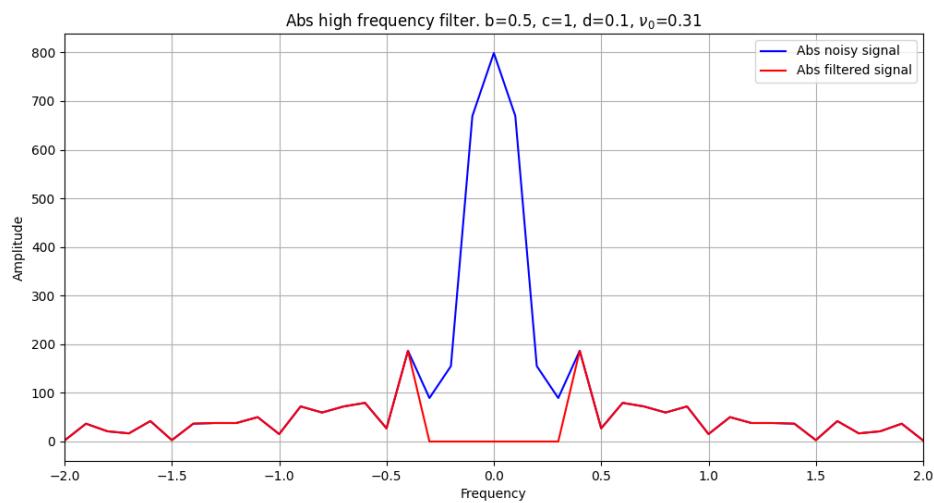


Рис. 64: График модуля Фурье-образа исходного и фильтрованного сигналов (3)

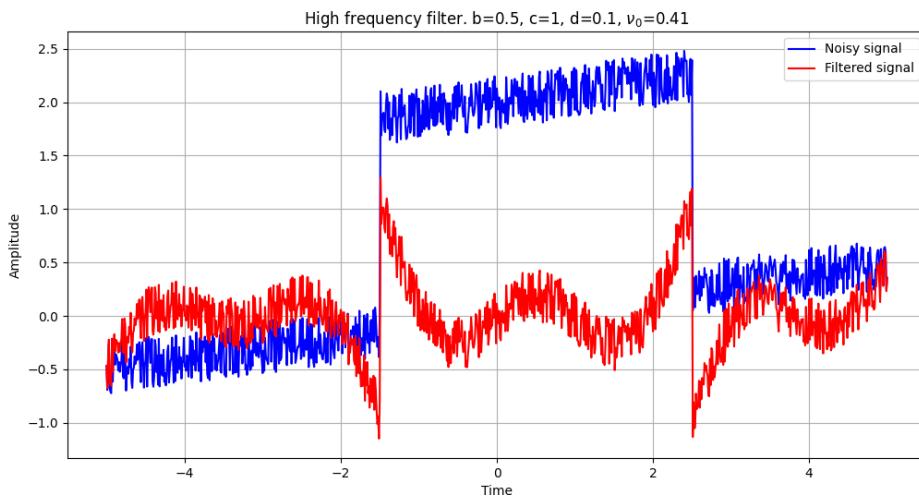


Рис. 65: График исходного и фильтрованного сигналов (4)

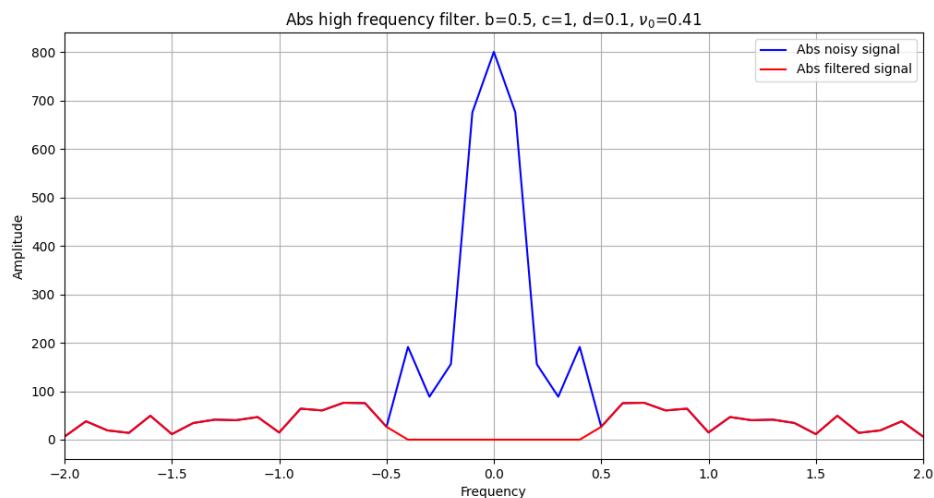


Рис. 66: График модуля Фурье-образа исходного и фильтрованного сигналов (4)

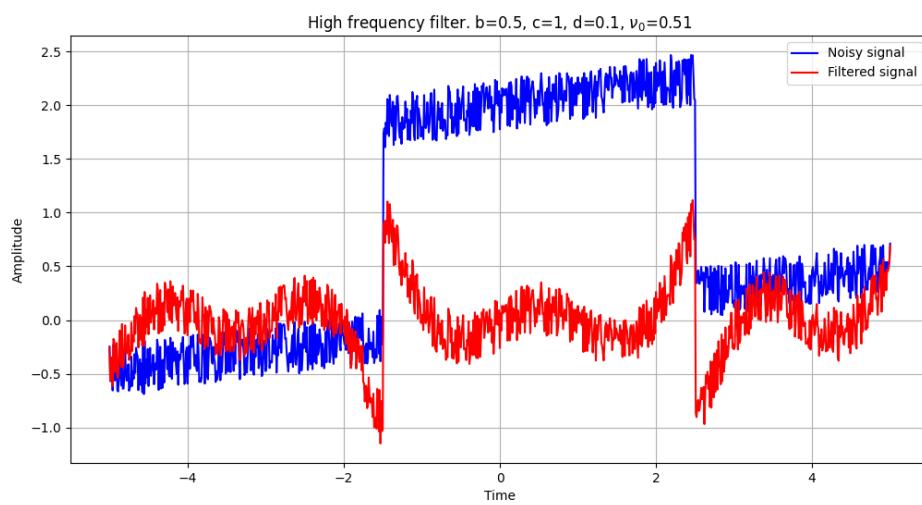


Рис. 67: График исходного и фильтрованного сигналов (5)

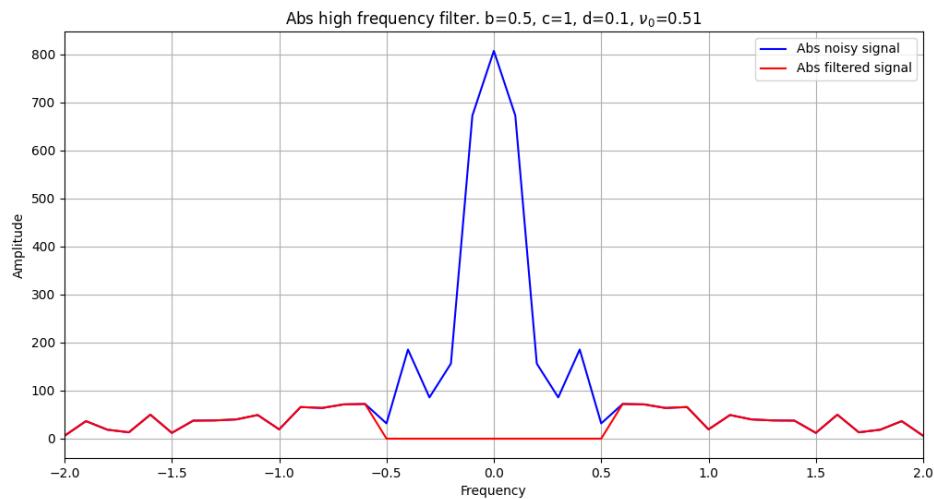


Рис. 68: График модуля Фурье-образа исходного и фильтрованного сигналов (5)

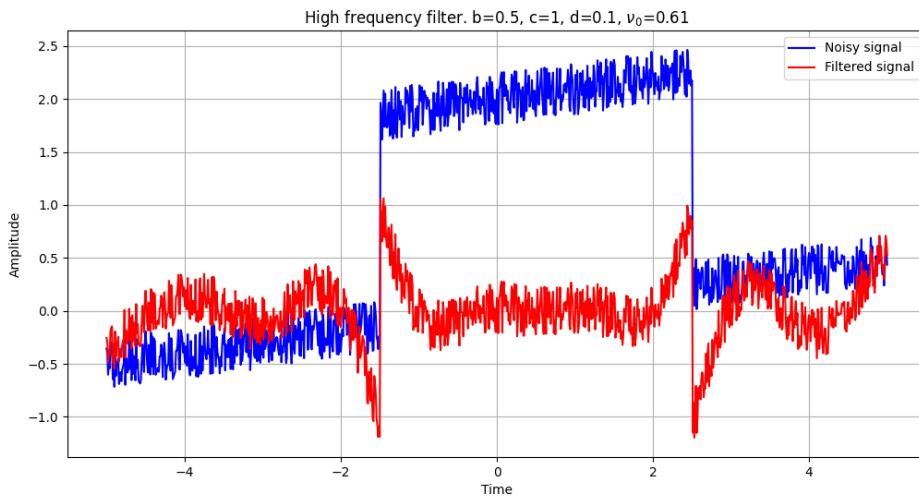


Рис. 69: График исходного и фильтрованного сигналов (6)

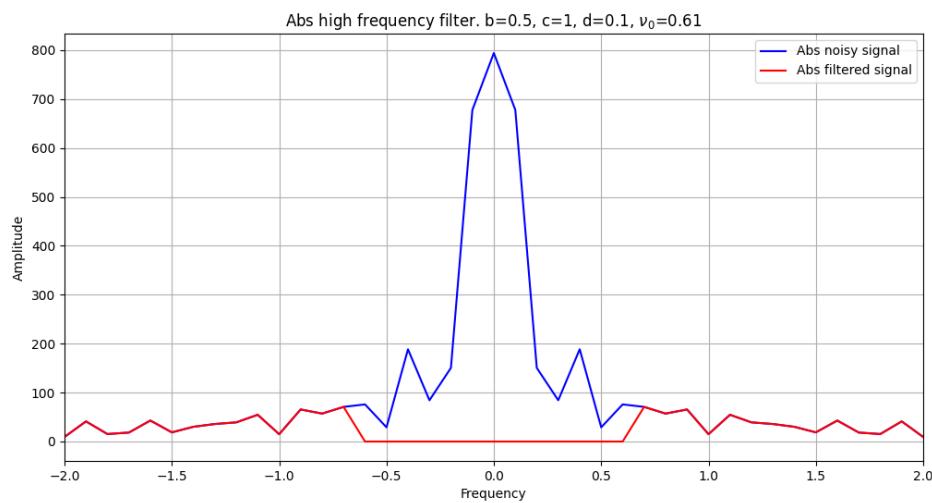


Рис. 70: График модуля Фурье-образа исходного и фильтрованного сигналов (6)

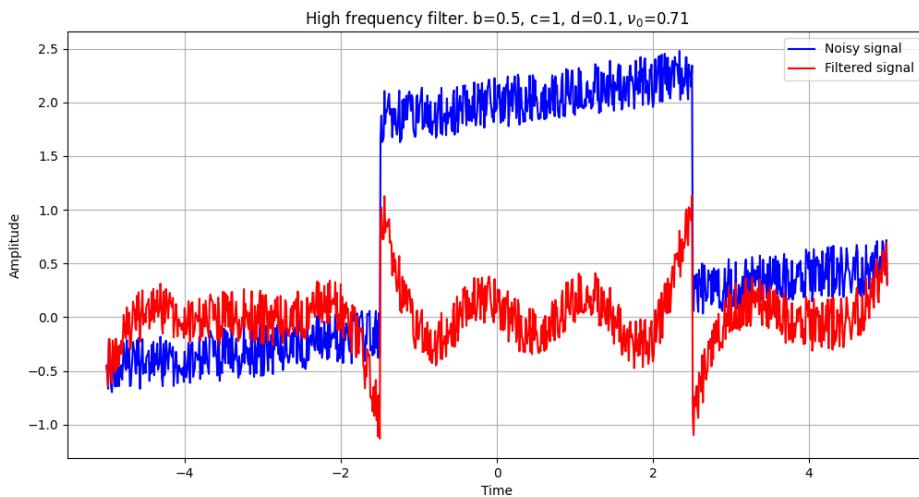


Рис. 71: График исходного и фильтрованного сигналов (7)

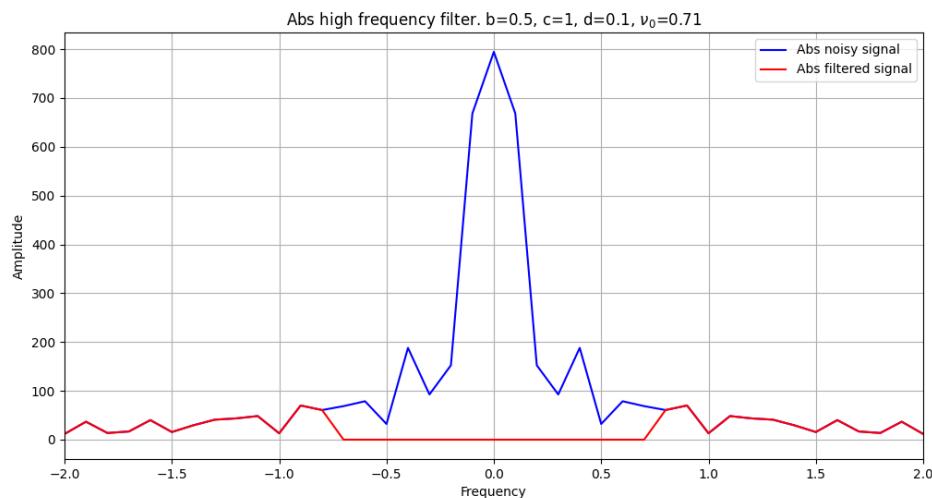


Рис. 72: График модуля Фурье-образа исходного и фильтрованного сигналов (7)

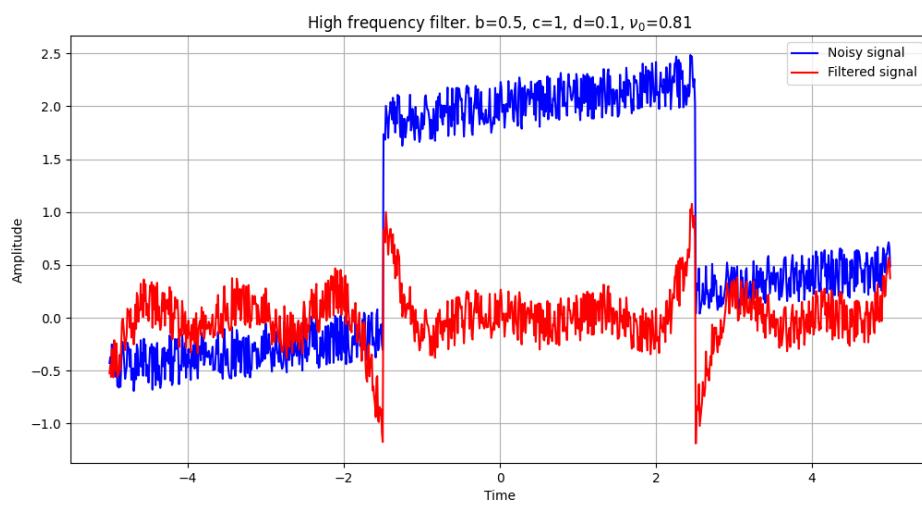


Рис. 73: График исходного и фильтрованного сигналов (8)

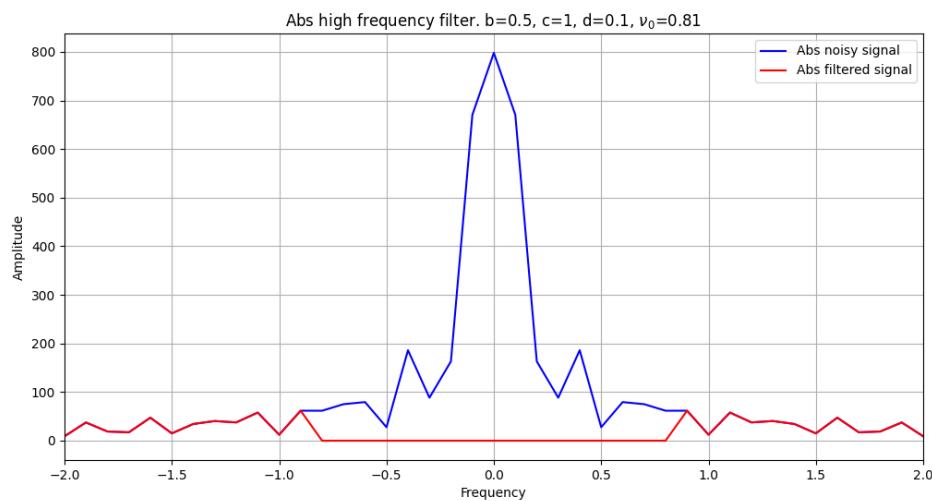


Рис. 74: График модуля Фурье-образа исходного и фильтрованного сигналов (8)

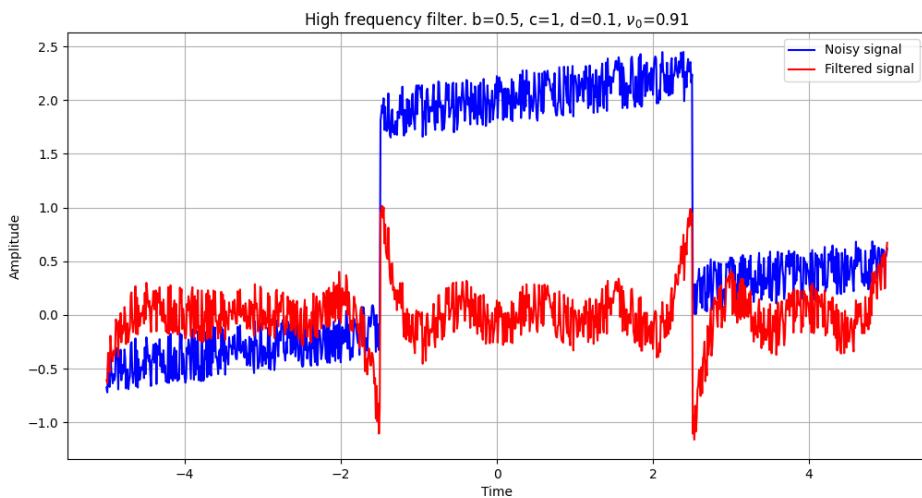


Рис. 75: График исходного и фильтрованного сигналов (9)

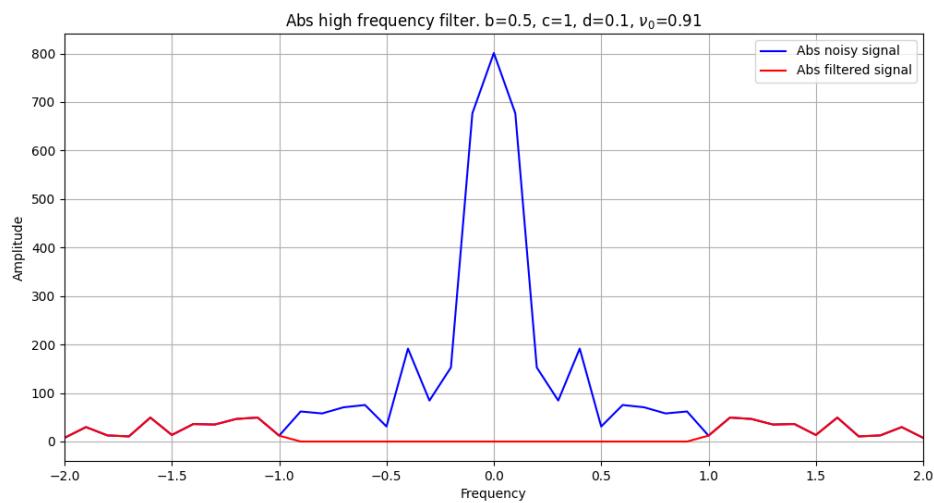


Рис. 76: График модуля Фурье-образа исходного и фильтрованного сигналов (9)

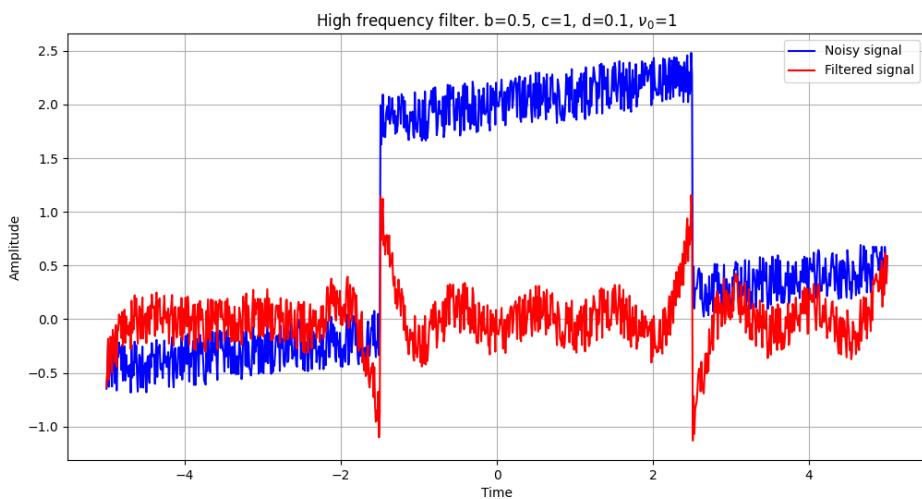


Рис. 77: График исходного и фильтрованного сигналов (10)

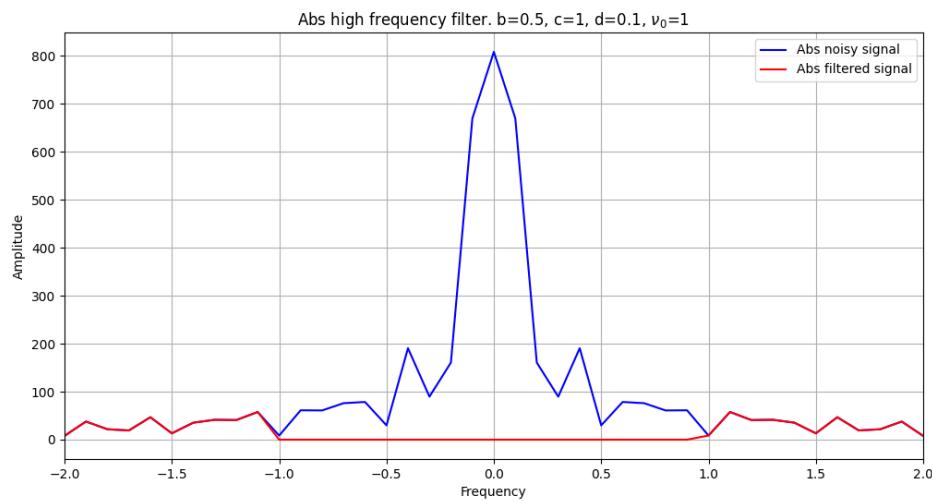


Рис. 78: График модуля Фурье-образа исходного и фильтрованного сигналов (10)

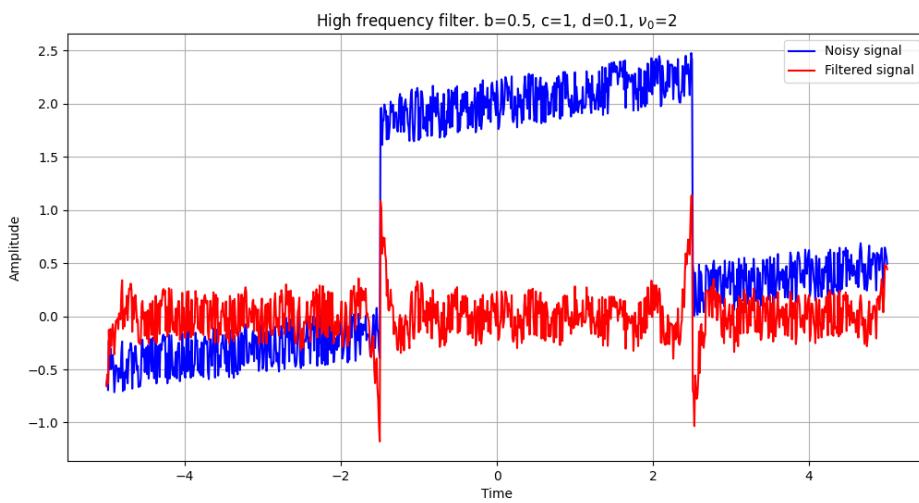


Рис. 79: График исходного и фильтрованного сигналов (11)

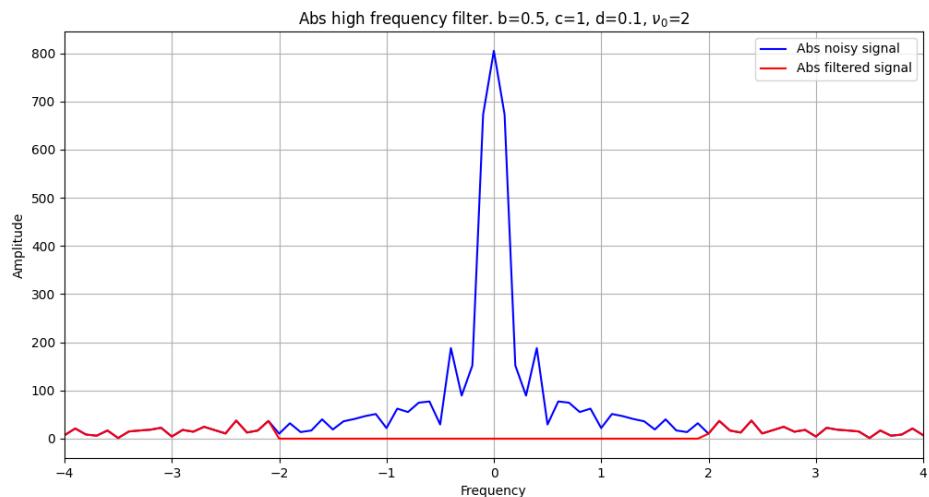


Рис. 80: График модуля Фурье-образа исходного и фильтрованного сигналов (11)

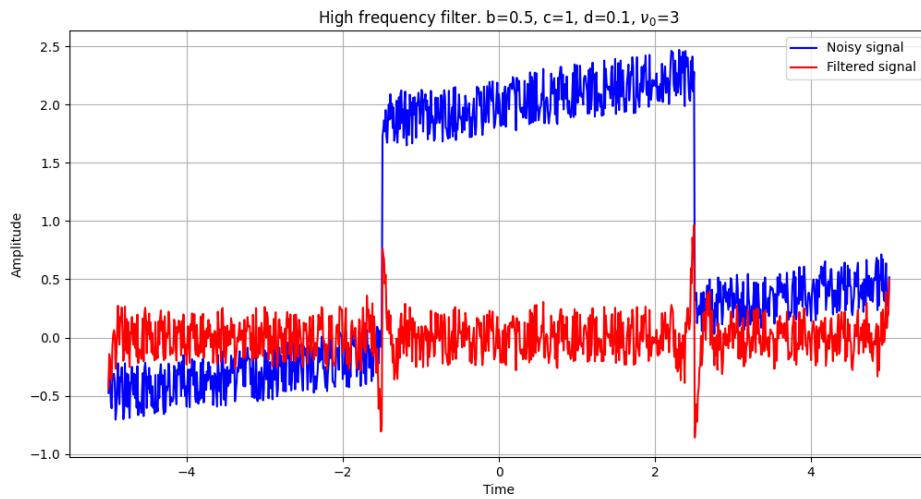


Рис. 81: График исходного и фильтрованного сигналов (12)

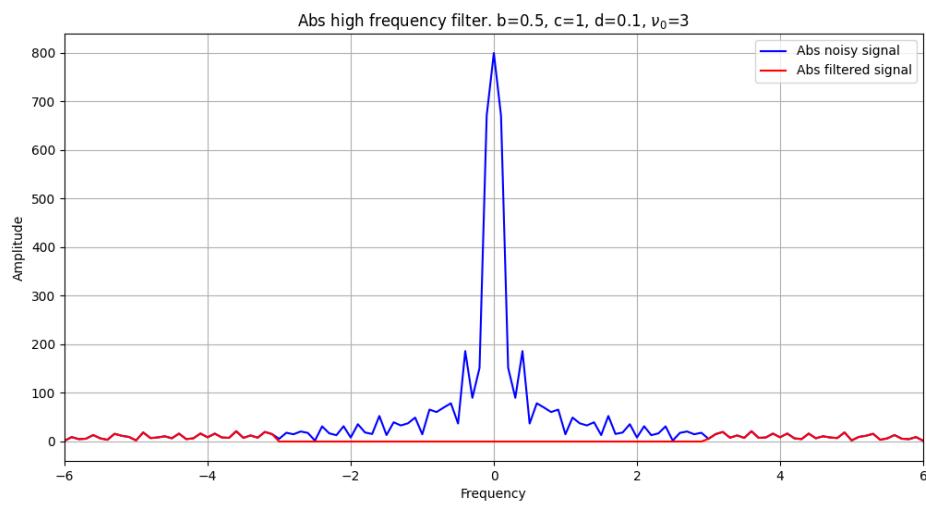


Рис. 82: График модуля Фурье-образа исходного и фильтрованного сигналов (12)

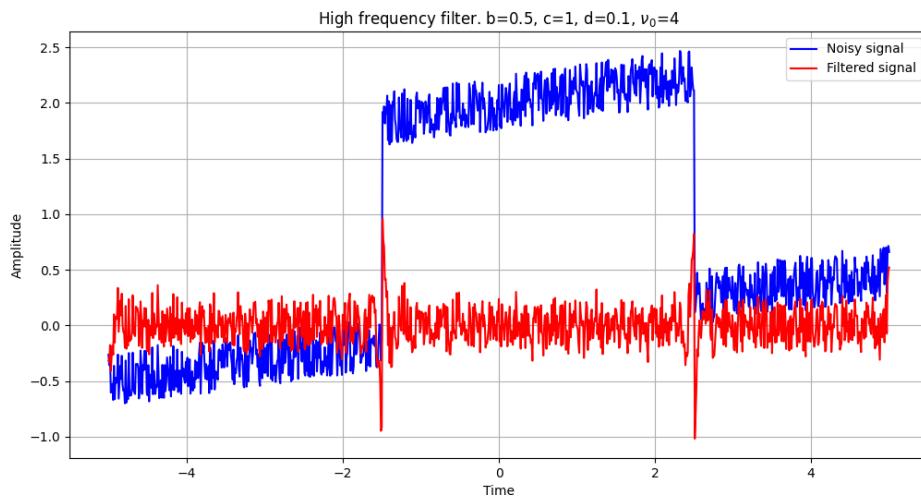


Рис. 83: График исходного и фильтрованного сигналов (13)

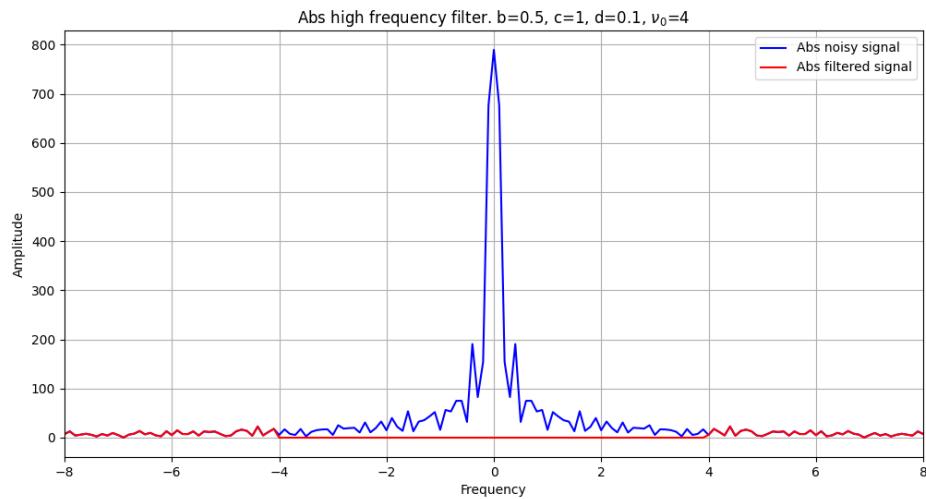


Рис. 84: График модуля Фурье-образа исходного и фильтрованного сигналов (13)

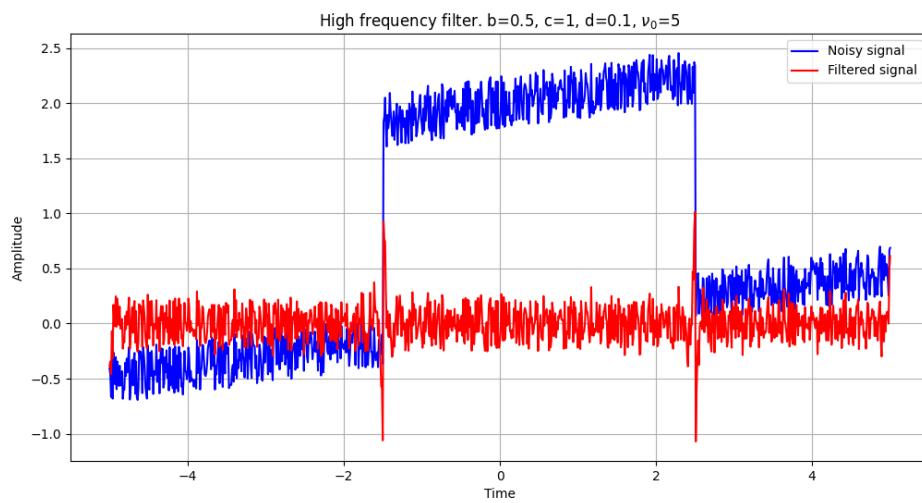


Рис. 85: График исходного и фильтрованного сигналов (14)

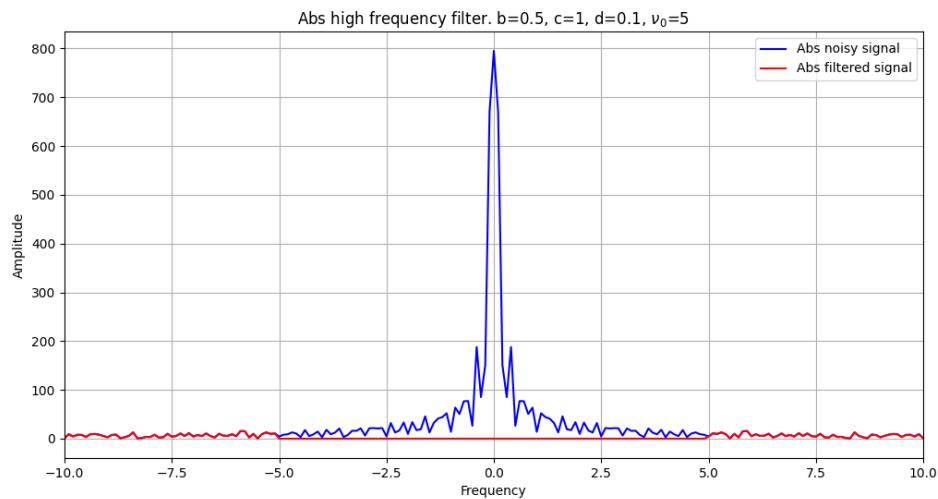


Рис. 86: График модуля Фурье-образа исходного и фильтрованного сигналов (14)

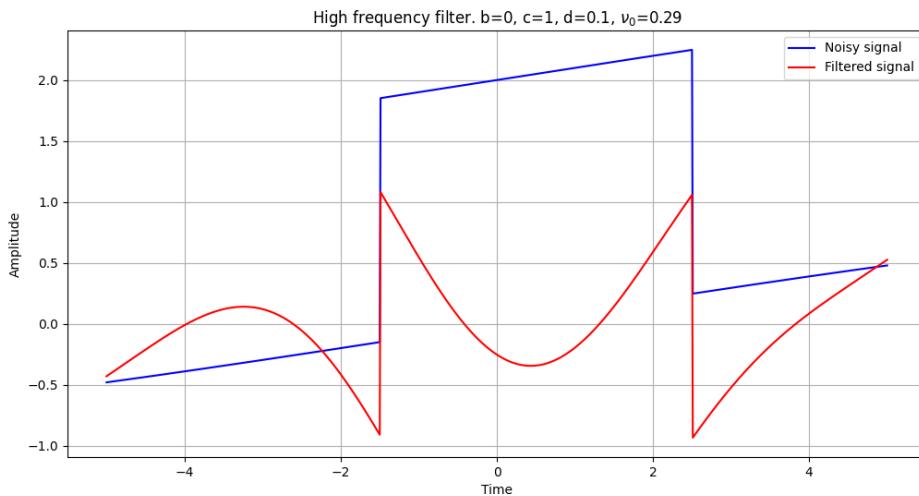


Рис. 87: График исходного и фильтрованного сигналов (15)

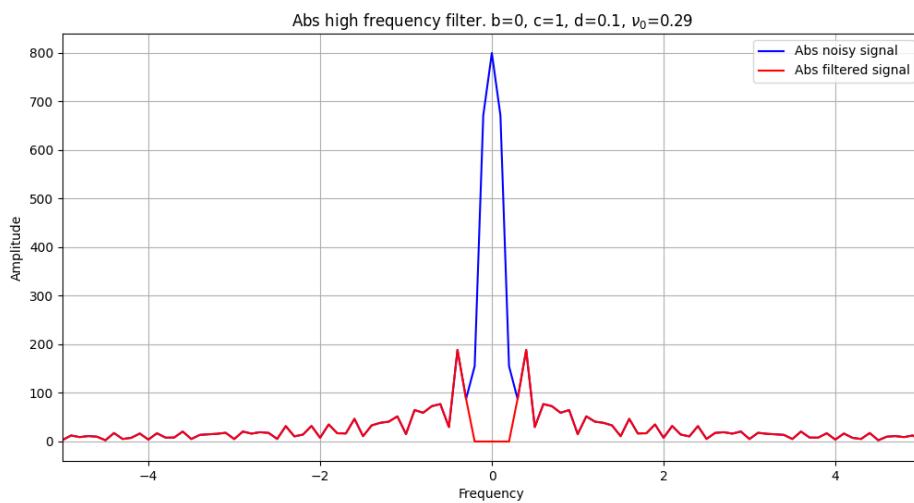


Рис. 88: График модуля Фурье-образа исходного и фильтрованного сигналов (15)

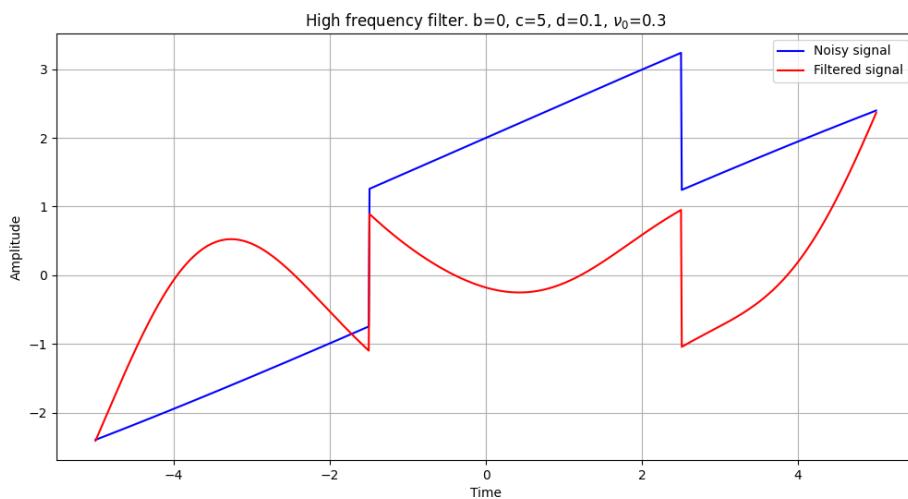


Рис. 89: График исходного и фильтрованного сигналов (16)

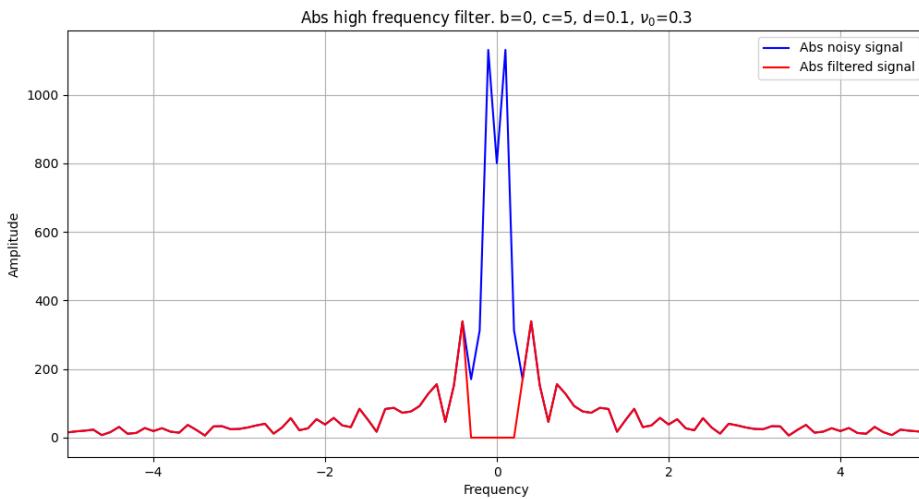


Рис. 90: График модуля Фурье-образа исходного и фильтрованного сигналов (16)

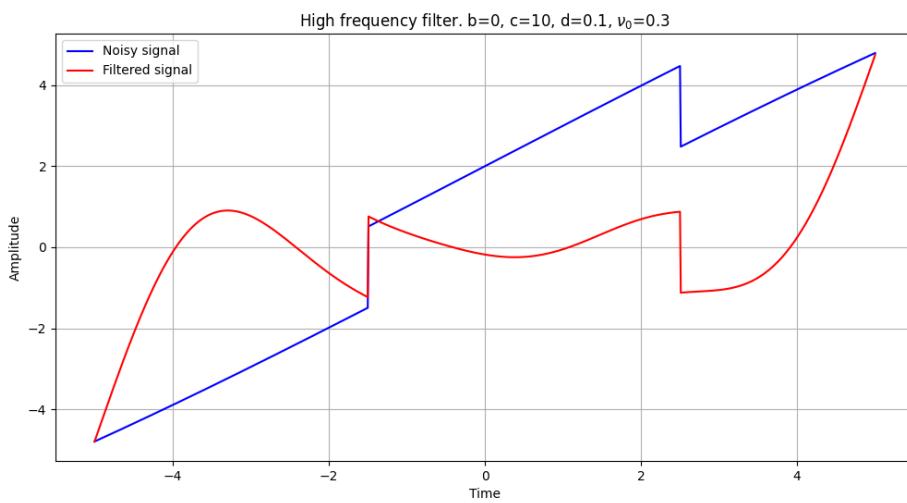


Рис. 91: График исходного и фильтрованного сигналов (17)

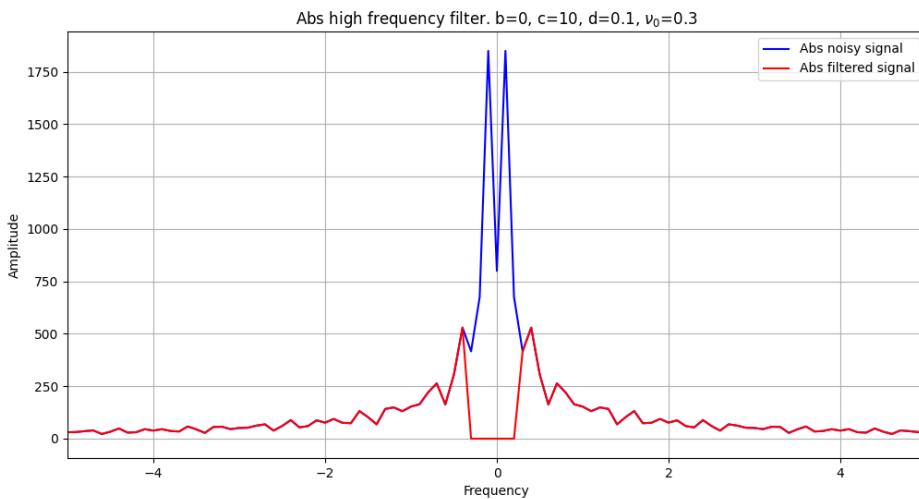


Рис. 92: График модуля Фурье-образа исходного и фильтрованного сигналов (17)

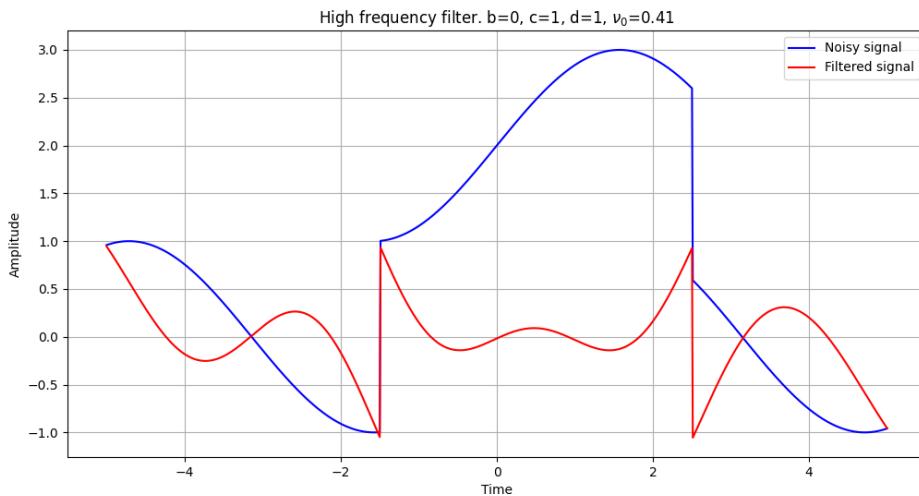


Рис. 93: График исходного и фильтрованного сигналов (18)

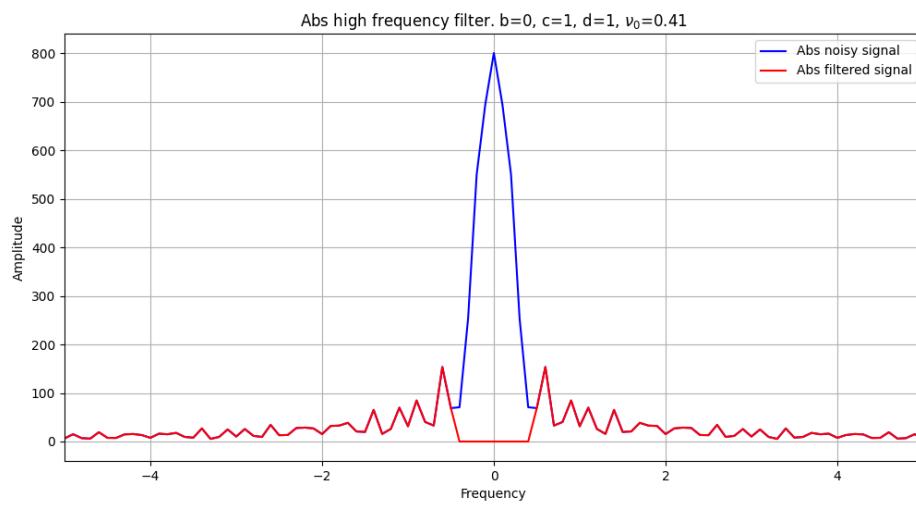


Рис. 94: График модуля Фурье-образа исходного и фильтрованного сигналов (18)

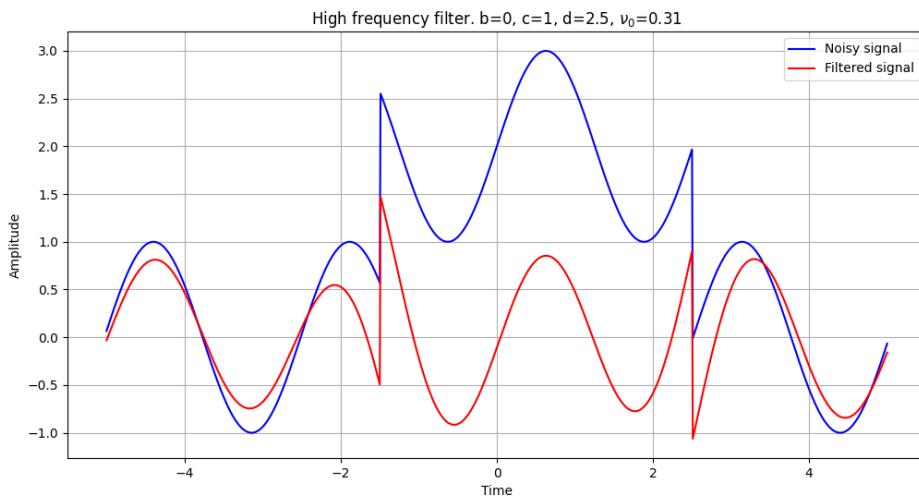


Рис. 95: График исходного и фильтрованного сигналов (19)

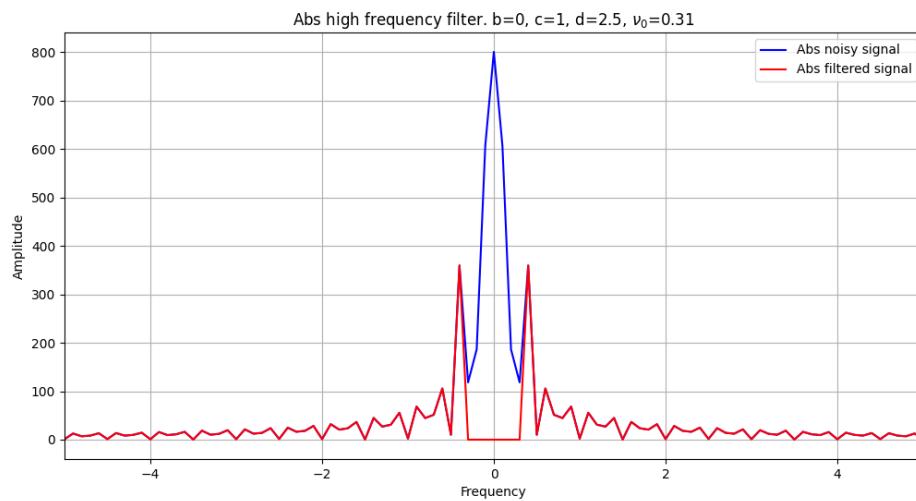


Рис. 96: График модуля Фурье-образа исходного и фильтрованного сигналов (19)

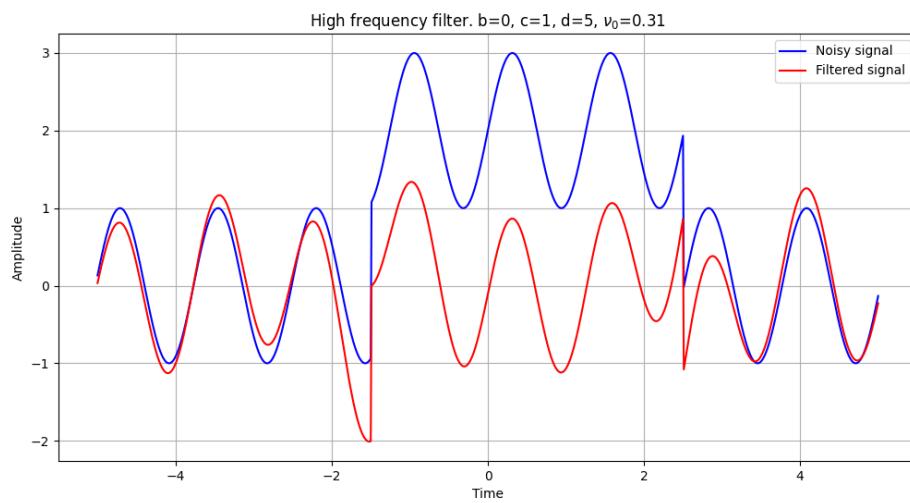


Рис. 97: График исходного и фильтрованного сигналов (20)

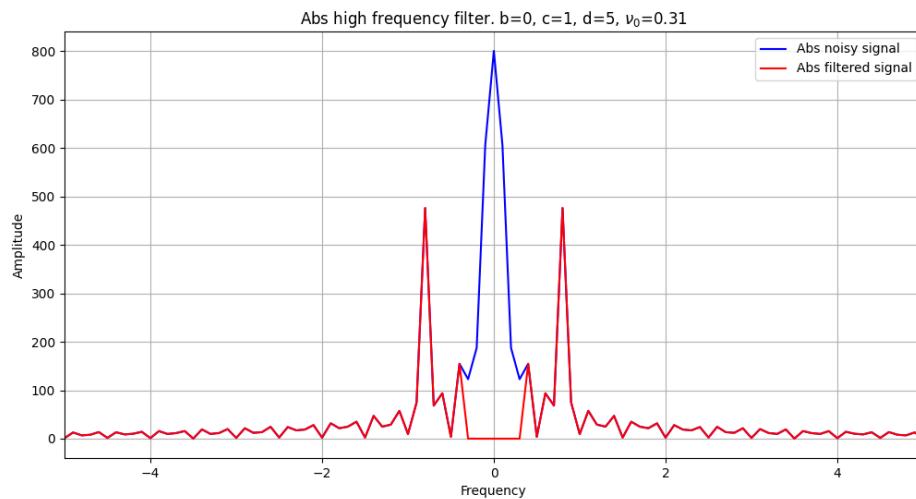


Рис. 98: График модуля Фурье-образа исходного и фильтрованного сигналов (20)

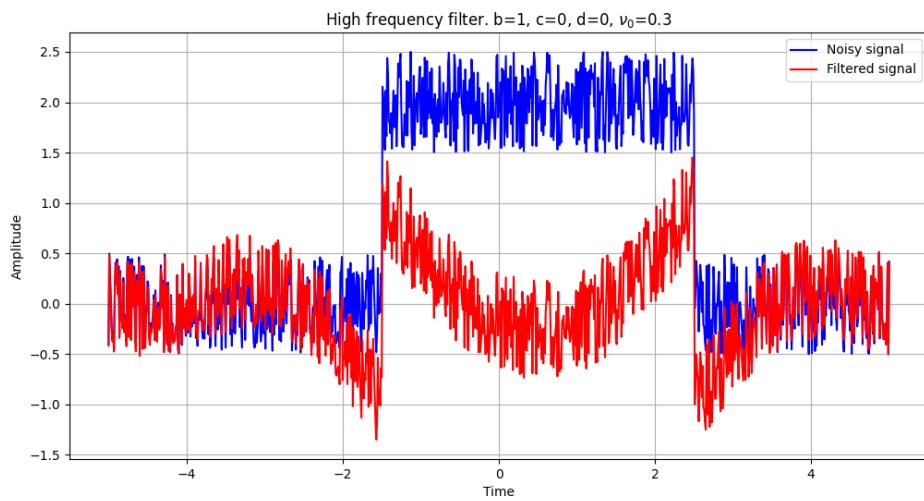


Рис. 99: График исходного и фильтрованного сигналов (21)

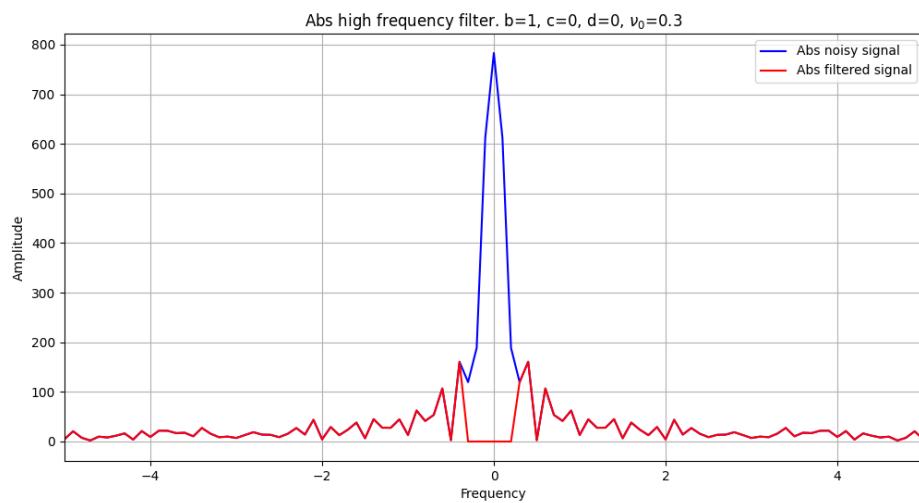


Рис. 100: График модуля Фурье-образа исходного и фильтрованного сигналов (21)

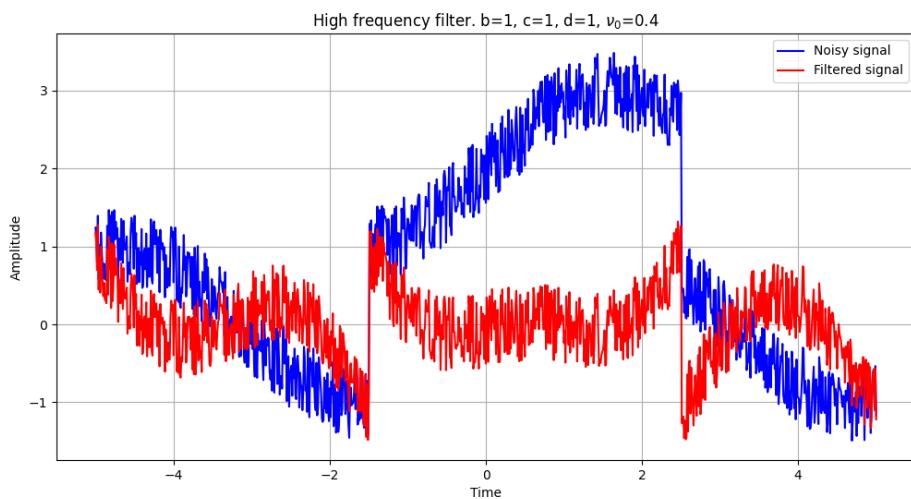


Рис. 101: График исходного и фильтрованного сигналов (22)

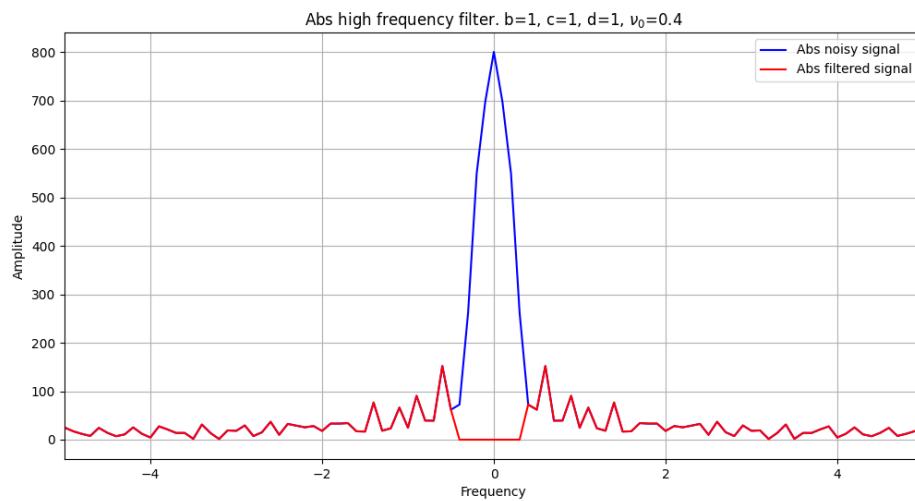


Рис. 102: График модуля Фурье-образа исходного и фильтрованного сигналов (22)

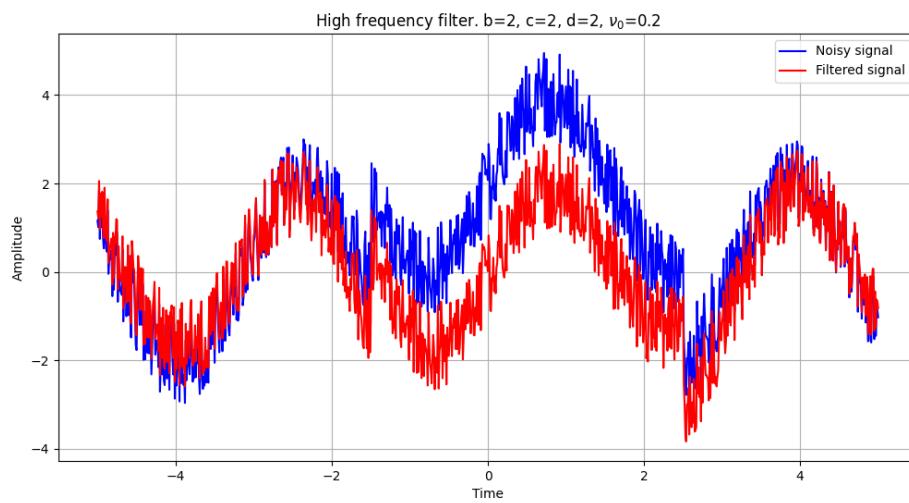


Рис. 103: График исходного и фильтрованного сигналов (23)

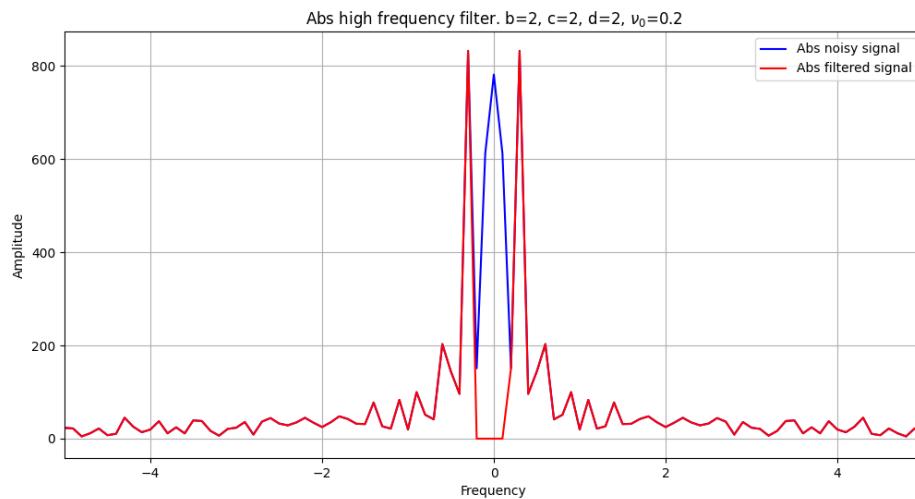


Рис. 104: График модуля Фурье-образа исходного и фильтрованного сигналов (23)

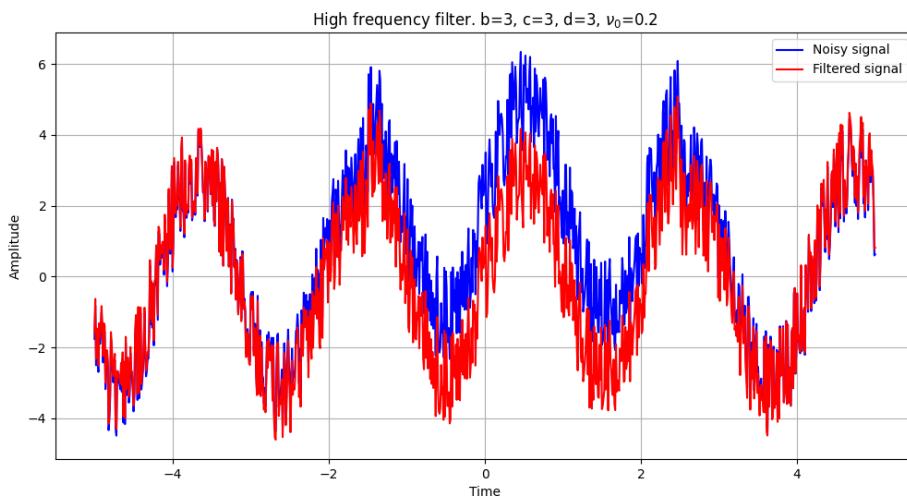


Рис. 105: График исходного и фильтрованного сигналов (24)

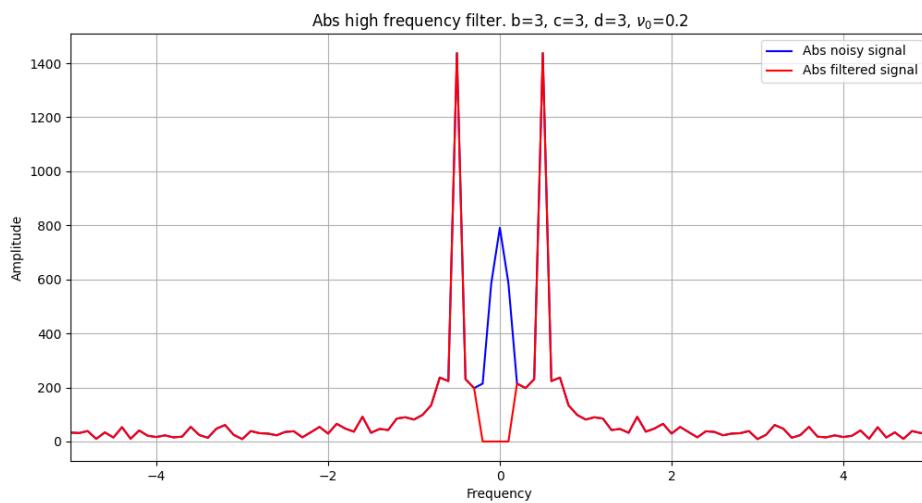


Рис. 106: График модуля Фурье-образа исходного и фильтрованного сигналов (24)

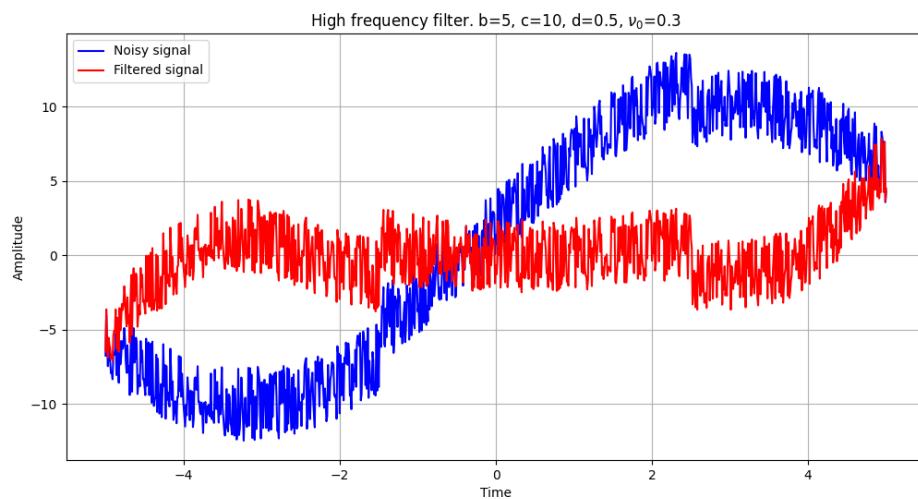


Рис. 107: График исходного и фильтрованного сигналов (25)

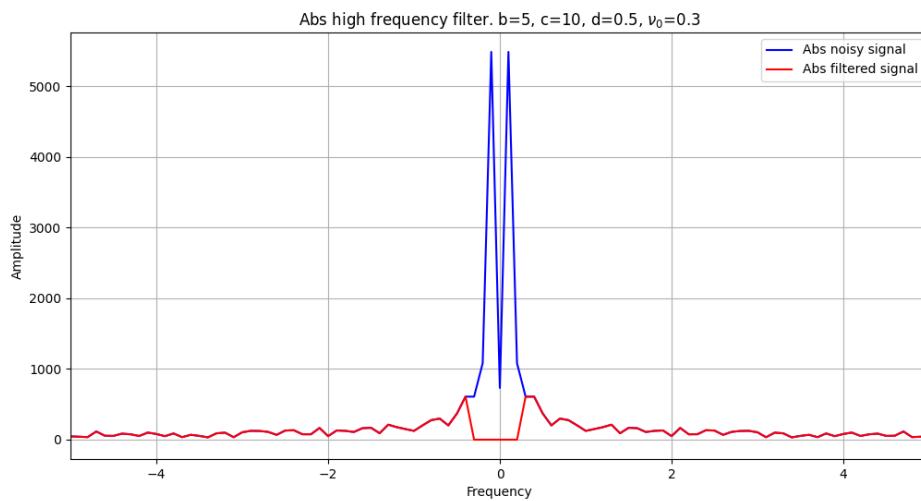


Рис. 108: График модуля Фурье-образа исходного и фильтрованного сигналов (25)

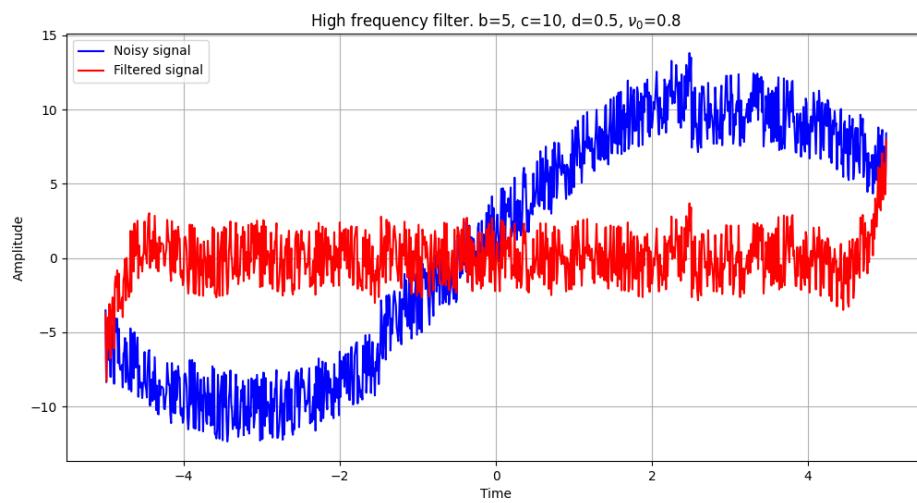


Рис. 109: График исходного и фильтрованного сигналов (26)

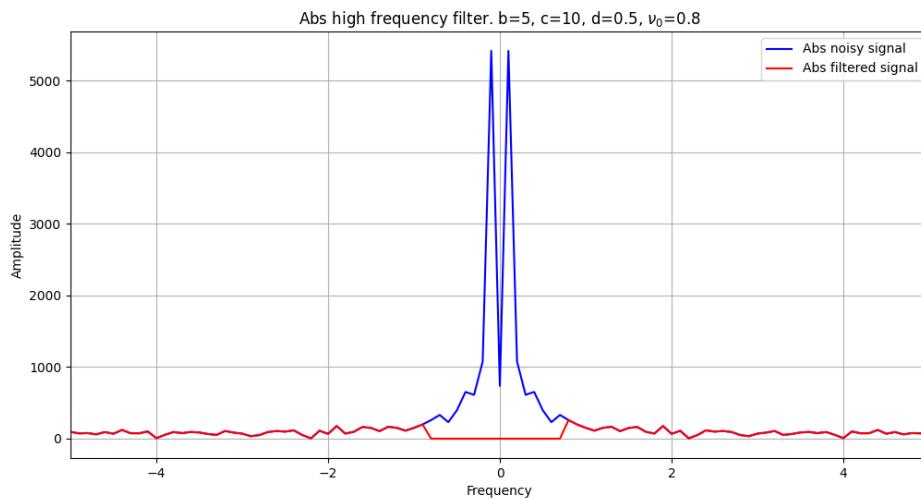


Рис. 110: График модуля Фурье-образа исходного и фильтрованного сигналов (26)

2 Задание 2. Фильтрация звука.

В данном задании необходимо убрать шумы из записи голоса в файле MUHA.wav так, чтобы остался только голос. При прослушивании записи голос слышно не очень хорошо, так как присутствует громкий гул. Построим графики исходного сигнала аудиозаписи и его Фурье-образа. Определим по второму графику, какие частоты могут создавать шумы. Так как гул громкий, нам необходимо вырезать частоты из аудиозаписи, имеющие наибольшую амплитуду. Такие частоты мы можем наблюдать примерно в диапазоне $[-300, 300]$ Гц. Чтобы вырезать эти частоты, потребуется фильтр верхних частот, который мы рассматривали в задании 1. После применения фильтра построим сравнительный график исходного и фильтрованного сигналов аудиозаписи. На нем мы видим, что мы успешно вырезали низкие частоты с наибольшей амплитудой, которые соответствовали громкому гулу. Теперь послушаем аудиозапись filtered_MUHA.wav, которая оставлена на этом [гугл-диске](#), и убедимся в том, что все посторонние шумы пропали. На записи голос слышно хорошо, однако остался некоторый звуковой эффект фейзер. Избавиться от него удалось обрезав частоты в диапазоне $[-22050, -1000]$ и $[1000, 22050]$ Гц, то есть применив фильтр нижних частот, однако голос сильно потерял в качестве. Прослушать этот вариант можно на том же [гугл-диске](#), файл выложен под названием filtered_MUHA_2.wav.

Далее представлены рисунки с графиками, о которых говорилось в предыдущем абзаце. Синим цветом обозначен исходный сигнал, красным – фильтрованный. Также представлены графики для фильтрованной аудиозаписи filtered_MUHA_2.wav. Стоит заметить, что на рисунке 116 видно, как окрестность нуля стала меньше по сравнению с рисунком 113, то есть в подобных диапазонах амплитуды частот уменьшились (убрали высокочастотный эффект фейзер, оставив диапазон с низкими частотами, но без тех, что создавали громкий гул).

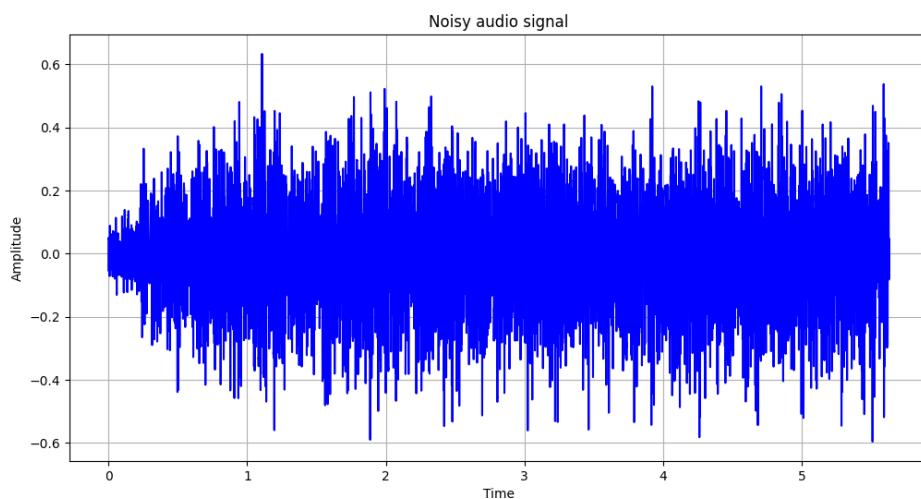


Рис. 111: График исходного сигнала аудиозаписи.

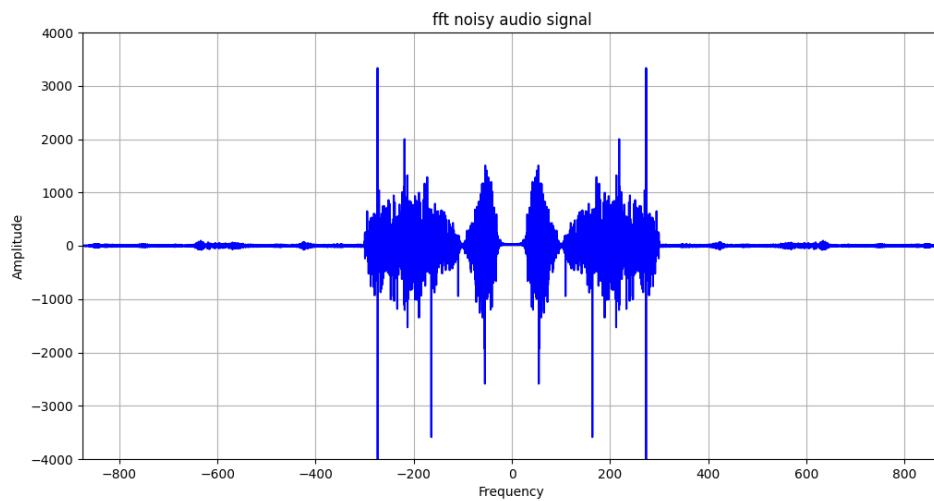


Рис. 112: График Фурье-образа исходного сигнала аудиозаписи.

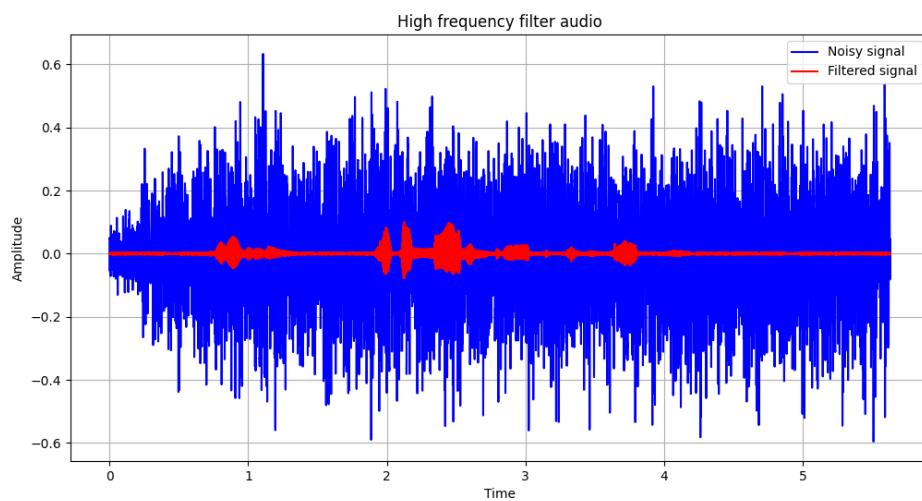


Рис. 113: График исходного и фильтрованного сигналов аудиозаписи (1)

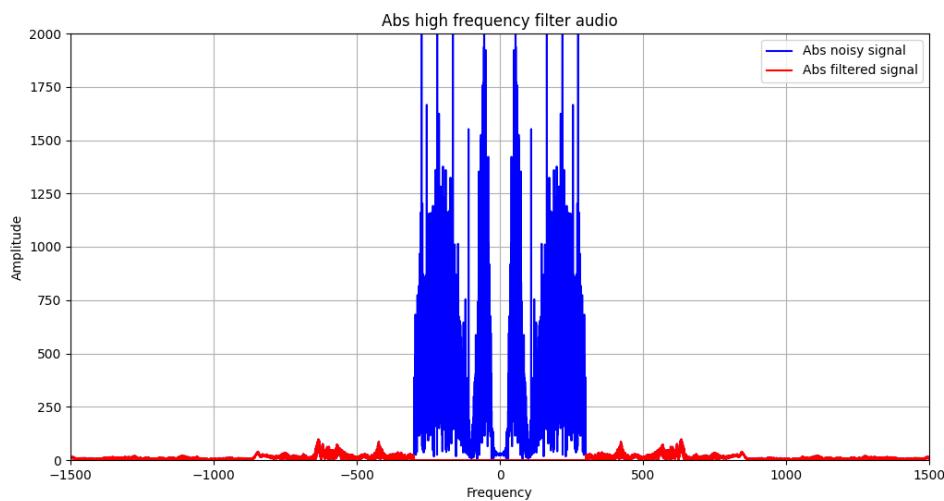


Рис. 114: График модуля Фурье-образа исходного и фильтрованного сигналов аудиозаписи (1)

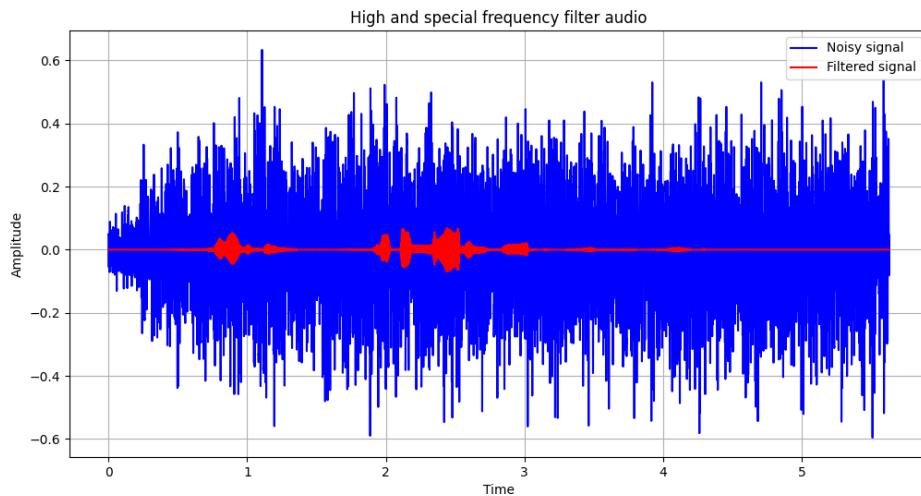


Рис. 115: График исходного и фильтрованного сигналов аудиозаписи (2)

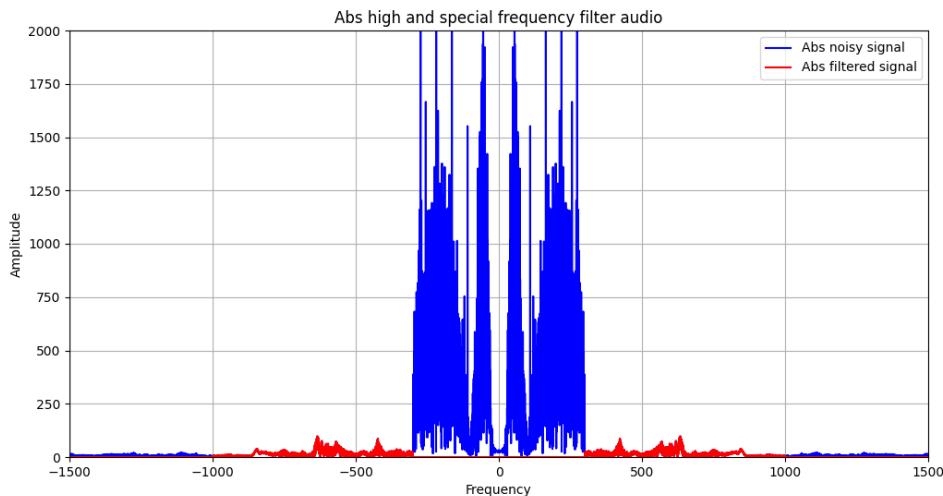


Рис. 116: График модуля Фурье-образа исходного и фильтрованного сигналов аудиозаписи (2)

3 Листинги программных реализаций

В этой секции будут рассмотрены программные реализации, которые использовались по ходу выполнения лабораторной работы. 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

```

```

7     def get_U(u):
8         return np.fft.fftshift(np.fft.fft(u))
9
10    def g_f(t, t_1, t_2, a):
11        if (t_1 <= t <= t_2):
12            return a
13        return 0
14
15    def u_f(g_fs: list, time: list, b, c, d):
16        return np.array(g_fs) + \
17            b*(np.random.rand(len(time))-0.5) + \
18            c*np.sin(d*time)
19
20    def get_g_fs(time: list, t_1, t_2, a):
21        gs = []
22        for t in range(len(time)):
23            gs.append(g_f(time[t], t_1, t_2, a))
24
25    return gs

```

Листинг 1: Файл help.py. Вспомогательные функции

```

1 import help as hp
2
3 a = 2
4 t_1, t_2 = -1.5, 2.5
5
6 T, dt = 10, 0.01
7 V, dv = 1 / dt, 1 / T
8
9 time = hp.get_t(T, dt)
10 freq = hp.get_v(V, dv)
11 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 (функция build_u__flt_u), модуля Фурье-образа исходного и фильтрованного сигналов (функции build_abs_U__flt_U и build_abs_u_to_U__flt_U, где вторая из приведенных в данном абзаце – вспомогательная. В нее можно передать исходный сигнал u , который преобразуется в Фурье-образ. Написана лишь для удобства пользователя) и графиков исходного сигнала или его Фурье-образа (функции build_u_or_U и build_u_to_U, где вторая из приведенных в данном абзаце нужна для удобства). Все функции принимают максимально много необязательных параметров для гибкой настройки графика.

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 from help import get_U
5
6 def build_u_to_U(freq: list,
7                   u: list,
8                   clr='b',
9                   x11=None,
10                  x12=None,
11                  y11=None,
12                  y12=None,
13                  xlab='Frequency',
14                  ylab='Amplitude',
15                  label=None,

```

```
16             title=None ,
17             fz1=6.4 ,
18             fz2=4.8 ,
19             legend: bool = True ,
20             grid: bool = True):
21     U = get_U(u)
22     build_u_or_U(freq, U, clr, x11, x12,
23                   y11, y12, xlab, ylab, label,
24                   title, fz1, fz2, legend, grid)
25
26 def build_u_or_U(torv: list ,
27                     uorU: list ,
28                     clr='b' ,
29                     x11=None ,
30                     x12=None ,
31                     y11=None ,
32                     y12=None ,
33                     xlab=None ,
34                     ylab='Amplitude' ,
35                     label=None ,
36                     title=None ,
37                     fz1=6.4 ,
38                     fz2=4.8 ,
39                     legend: bool = True ,
40                     grid: bool = True):
41     plt.plot(torv, uorU, color=clr, label=label)
42     plt.xlabel(xlab)
43     plt.ylabel(ylab)
44     plt.xlim(x11, x12)
45     plt.ylim(y11, y12)
46     plt.title(title)
47     if legend:
48         plt.legend()
49     plt.grid(grid)
50     plt.gcf().set_size_inches(fz1, fz2)
51     plt.show()
52
53 def build_u__flt_u(time: list ,
54                     u: list ,
55                     flt_u: list ,
56                     clr1='b' ,
57                     clr2='r' ,
58                     lab1='Noisy signal' ,
59                     lab2='Filtered signal' ,
60                     xlab='Time' ,
61                     ylab='Amplitude' ,
62                     title=None ,
63                     fz1=6.4 ,
64                     fz2=4.8 ,
65                     legend: bool = True ,
66                     grid: bool = True ,
67                     x11=None ,
68                     x12=None ,
69                     y11=None ,
70                     y12=None):
71     plt.plot(time, u, color=clr1, label=lab1)
72     plt.plot(time, flt_u, color=clr2, label=lab2)
73     plt.xlabel(xlab)
74     plt.ylabel(ylab)
75     plt.xlim(x11, x12)
```

```
76     plt.ylim(y11, y12)
77     plt.title(title)
78     if legend:
79         plt.legend()
80     plt.grid(grid)
81     plt.gcf().set_size_inches(fz1, fz2)
82     plt.show()
83
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 фильтрацию. Пользователь может вызвать такие функции, как filter_high, filter_low, filter_special_out/in для фильтрации верхних, нижних и специфических частот соответственно. Первые две функции принимают некоторый параметр ν_0 , а последняя принимает список диапазонов с некоторыми ν_0^i и ν_0^j (где i, j – индексы, не степени). Здесь реализован алгоритм, описанный в первом задании – берется Фурье-образ, фильтруется и преобразуется обратно из частотного пространства во временное. Функция filter_U обнуляет частоты, соответствующие конкретному шагу в зависимости от решения фильтров проверки high_filter, low_filter и special_filter_out/in.

```

1  def filter_U(u: list, freq: list, v_0, filter):
2      flt_U = get_U(u)
3      for i in range(len(freq)):
4          freq_i = freq[i]
5          if filter(freq_i, v_0):
6              flt_U[i] = 0
7
8      return flt_U
9
10
10 def low_filter(freq, v_0):
11     if -v_0 <= freq <= v_0:
12         return False
13     return True
14
15 def high_filter(freq, v_0):
16     return not low_filter(freq, v_0)
17
18 def special_filter_in(freq, v_0: list):
19     for i in range(len(v_0)):
20         if v_0[i][0] <= freq <= v_0[i][1]:
21             return False
22     return True
23
24 def special_filter_out(freq, v_0: list):
25     return not special_filter_in(freq, v_0)
26
27 def filter_low(freq: list, u: list, v_0):
28     if isinstance(v_0, list) or \
29         len(freq) <= 0 or \
30         len(u) <= 0:
31         return None
32
33     flt_U = filter_U(u, freq, v_0, low_filter)
34     flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
35     return flt_u, flt_U
36
37 def filter_high(freq: list, u: list, v_0):
38     if isinstance(v_0, list) or \
39         len(freq) <= 0 or \
40         len(u) <= 0:
41         return None
42
43     flt_U = filter_U(u, freq, v_0, high_filter)
44     flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
45     return flt_u, flt_U
46

```

```

47     def filter_special_in(freq: list, u: list, v_0: list):
48         if not isinstance(v_0, list) or \
49             len(v_0) <= 0 or \
50                 len(freq) <= 0 or \
51                     len(u) <= 0:
52                         return None
53
54         flt_U = filter_U(u, freq, v_0, special_filter_in)
55         flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
56         return flt_u, flt_U
57
58     def filter_special_out(freq: list, u: list, v_0: list):
59         if not isinstance(v_0, list) or \
60             len(v_0) <= 0 or \
61                 len(freq) <= 0 or \
62                     len(u) <= 0:
63                         return None
64
65         flt_U = filter_U(u, freq, v_0, special_filter_out)
66         flt_u = np.fft.ifft(np.fft.ifftshift(flt_U))
67         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. Любые остальные добавления в код нужны лишь для удобства, например, когда нужно построить много графиков и сравнивать их друг с другом или анализировать различные результаты по типу совмещения нескольких фильтров на одном сигнале. В случае очистки специфических частот задается список ν_0^i , ν_0^j . В каждый файл импортируются numpy как np, static как st, help как hp и filters как ft. Для работы с аудиозаписью подключены библиотеки librosa, scipy и playsound.

```

1  import help as hp
2  import static as st
3
4  import filters as ft
5  import builder as bd
6
7  time = st.time
8  freq = st.freq
9  g_fs = st.g_fs
10
11 b, c, d = 0.5, 0, 0.1
12 v_0 = 2.5
13
14 u = hp.u_f(g_fs, time, b, c, d)
15 flt_u, flt_U = ft.filter_low(freq, u, v_0)
16
17 bd.build_u__flt_u(
18     time,
19     u,
20     flt_u,
21     title=rf'Low frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0},
22     fz1=12,
23     fz2=6)
24 bd.build_abs_u_to_U__flt_U(
25     freq,
26     u,

```

```

27     flt_U ,
28     title=
29     rf 'Abs low frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0} ,
30     xl1=-5 ,
31     xl2=5 ,
32     fz1=12 ,
33     fz2=6)

```

Листинг 5: Файл nohigh.py. Фильтрация нижних частот.

```

1 import help as hp
2 import static as st
3
4 import filters as ft
5 import builder as bd
6
7 time = st.time
8 freq = st.freq
9 g_fs = st.g_fs
10
11 b, c, d = 5, 10, 0.5
12 v_0 = 0.3
13
14 u = hp.u_f(g_fs, time, b, c, d)
15 flt_u, flt_U = ft.filter_high(freq, u, v_0)
16
17 bd.build_u__flt_u(
18     time,
19     u,
20     flt_u,
21     title=rf 'High frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0} ,
22     fz1=12,
23     fz2=6)
24 bd.build_abs_u_to_U__flt_U(
25     freq,
26     u,
27     flt_U,
28     title=
29     rf 'Abs high frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v_0} ,
30     fz1=12,
31     fz2=6,
32     xl1=-5,
33     xl2=5)

```

Листинг 6: Файл nolow.py. Фильтрация верхних частот.

```

1 import help as hp
2 import static as st
3
4 import filters as ft
5 import builder as bd
6
7 time = st.time
8 freq = st.freq
9 g_fs = st.g_fs
10
11 b, c, d = 1.5, -2, -3
12 v0, v01 = [[-0.52, -0.3], [0.3, 0.52]], 10
13 u = hp.u_f(g_fs, time, b, c, d)
14
15 flt_u_so, flt_U_so = ft.filter_special_out(freq, u, v0)

```

```

16     bd.build_u__flt_u(
17         time,
18         u,
19         flt_u_so.real,
20         title=
21             rf'Special frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v0}',
22             fz1=12,
23             fz2=6)
24     bd.build_abs_u_to_U__flt_U(
25         freq,
26         u,
27         flt_U_so,
28         title=
29             rf'Abs special frequency filter. b={b}, c={c}, d={d}, $\nu_0$={v0}',
30             fz1=12,
31             fz2=6,
32             x11=-10,
33             x12=10)
34
35     flt_u_lso, flt_U_lso = ft.filter_low(freq, flt_u_so, v01)
36     bd.build_u__flt_u(
37         time,
38         u,
39         flt_u_lso.real,
40         title=
41             rf'Low and special frequency filter. b={b}, c={c}, d={d}, (1) $\nu_0$={v01}, (2) $\nu_0$={v0},
42             fz1=12,
43             fz2=6)
44     bd.build_abs_u_to_U__flt_U(
45         freq,
46         u,
47         flt_U_lso,
48         title=
49             rf'Abs low and special frequency filter. b={b}, c={c}, d={d}, (1) $\nu_0$={v01}, (2) $\nu_0$={v0},
50             fz1=12,
51             fz2=6,
52             x11=-10 - v01,
53             x12=10 + v01)

```

Листинг 7: Файл nospec.py. Фильтрация специфических частот.

```

1 import numpy as np
2 import librosa
3 from scipy.io.wavfile import write
4 from playsound import playsound
5
6 import filters as ft
7 import builder as bd
8
9 filter_all = True
10 build_U = True
11
12 title = 'High frequency filter audio'
13 title2 = 'Abs high frequency filter audio'
14 if filter_all:
15     title = 'High and special frequency filter audio'
16     title2 = 'Abs high and special frequency filter audio'
17
18 src = 'sound/MUHA.wav'

```

```

19     filename = 'sound/filtered_MUHA_2.wav'
20     try:
21         audio, rate = librosa.load(src, sr=None)
22     except:
23         lab = 'fm_lab3/'
24         src = lab + src
25         filename = lab + filename
26         audio, rate = librosa.load(src, sr=None)
27
28     dt = 1 / rate
29     T = len(audio) * dt
30
31     time = np.linspace(0, T, len(audio), endpoint=False)
32     freq = np.linspace(-rate / 2, rate / 2, len(audio), endpoint=False)
33
34     flt_u, flt_U = ft.filter_high(freq, audio, 300)
35     if filter_all:
36         flt_u, flt_U = ft.filter_special_out(freq, flt_u,
37                                             [[freq[0], -4500],
38                                              [4500, freq[-1]]])
39     flt_u_float = flt_u.real.astype(np.float32)
40
41     if build_U:
42         bd.build_u_or_U(time,
43                           audio,
44                           xlab='Time',
45                           title='Noisy audio signal',
46                           fz1=12,
47                           fz2=6,
48                           legend=False)
49         bd.build_u_to_U(freq,
50                           audio,
51                           title='fft noisy audio signal',
52                           legend=False,
53                           fz1=12,
54                           fz2=6,
55                           xl1=0,
56                           yl1=0,
57                           yl2=20)
58
59         bd.build_u__flt_u(time, audio, flt_u, title=title, fz1=12, fz2=6)
60         bd.build_abs_u_to_U__flt_U(freq,
61                                     audio,
62                                     flt_U,
63                                     title=title2,
64                                     fz1=12,
65                                     fz2=6,
66                                     xl1=0,
67                                     yl1=0,
68                                     yl2=20)
69
70     write(filename, rate, flt_u_float)
71     playsound(filename)

```

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