

Федеральное государственное автономное образовательное
учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа №3
Жесткая фильтрация

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

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

Санкт-Петербург
2025

Содержание

Задание 1. Жесткие фильтры	3
Задание 1.1. Убираем высокие частоты	3
Задание 1.2. Убираем специфические частоты	7
Задание 1.3. Убираем низкие частоты?	13
Приложение к пункту 1.3	17
Задание 2. Фильтрация звука.	18
Приложение с кодом для задания 1.	19
Приложение с кодом для задания 2.	21

Задание 1. Жесткие фильтры

Для этого задания зададимся следующей функцией прямоугольной волны:

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

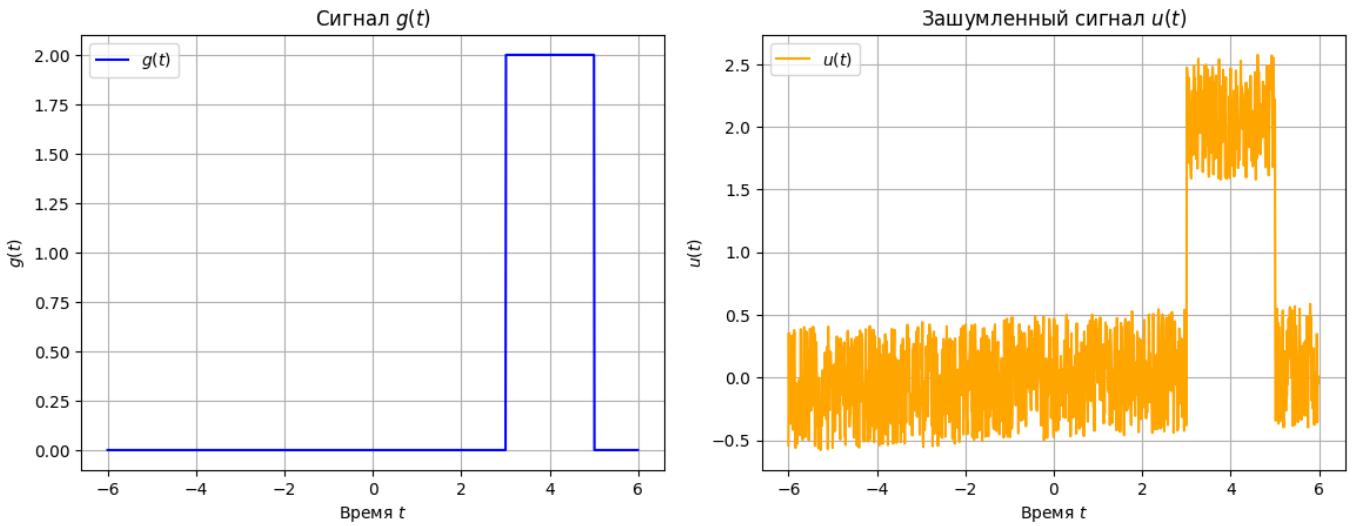


Рис. 1: График $g(t)$ и $u(t)$. $b = 0.5$ $\omega = 0.2$ $= 0.1$

А также и зашумленной версией этой функции:

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

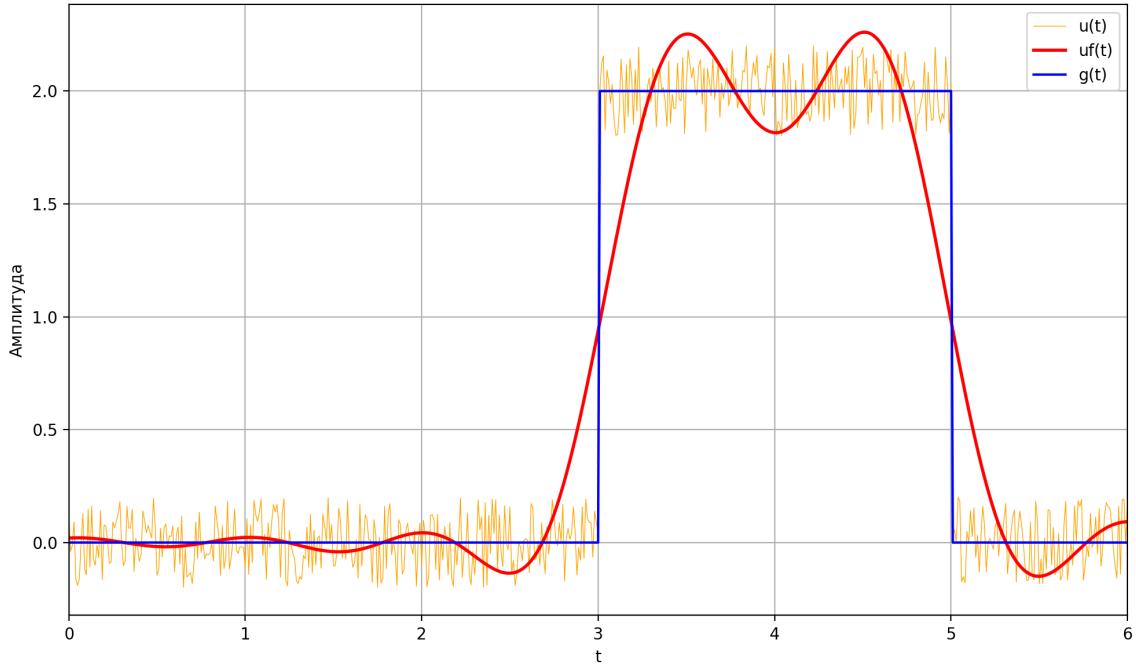
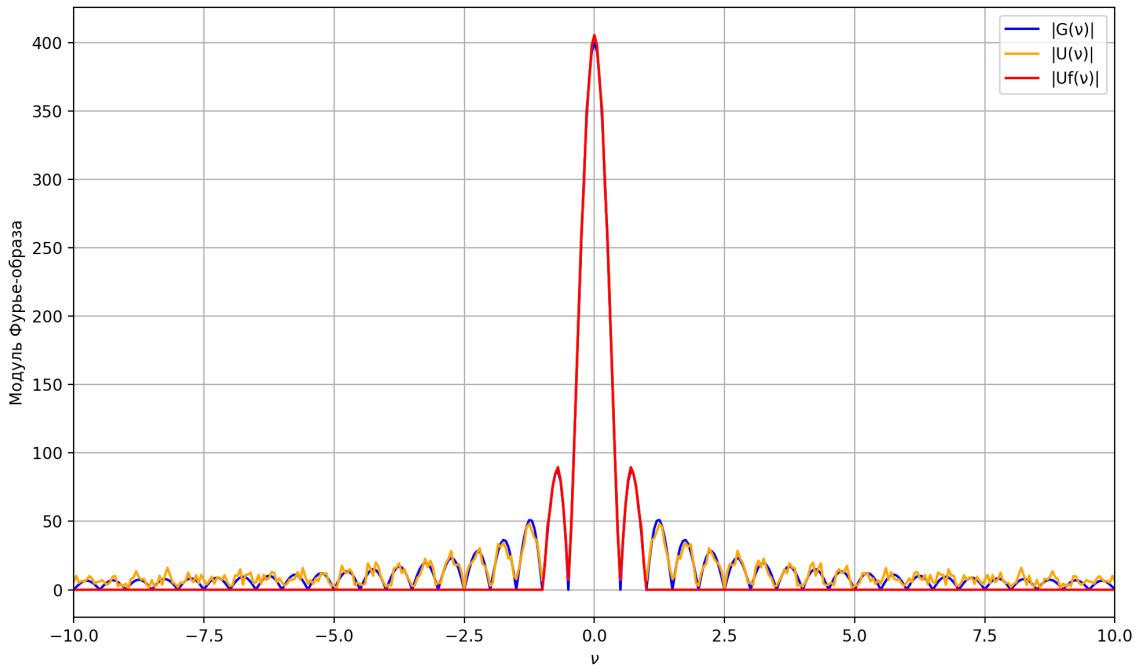
$\xi(t) \sim U[-1, 1]$ - равномерное распределение, а значения b, c, ω - настраиваемые параметры возмущений

Задание 1.1. Убираем высокие частоты

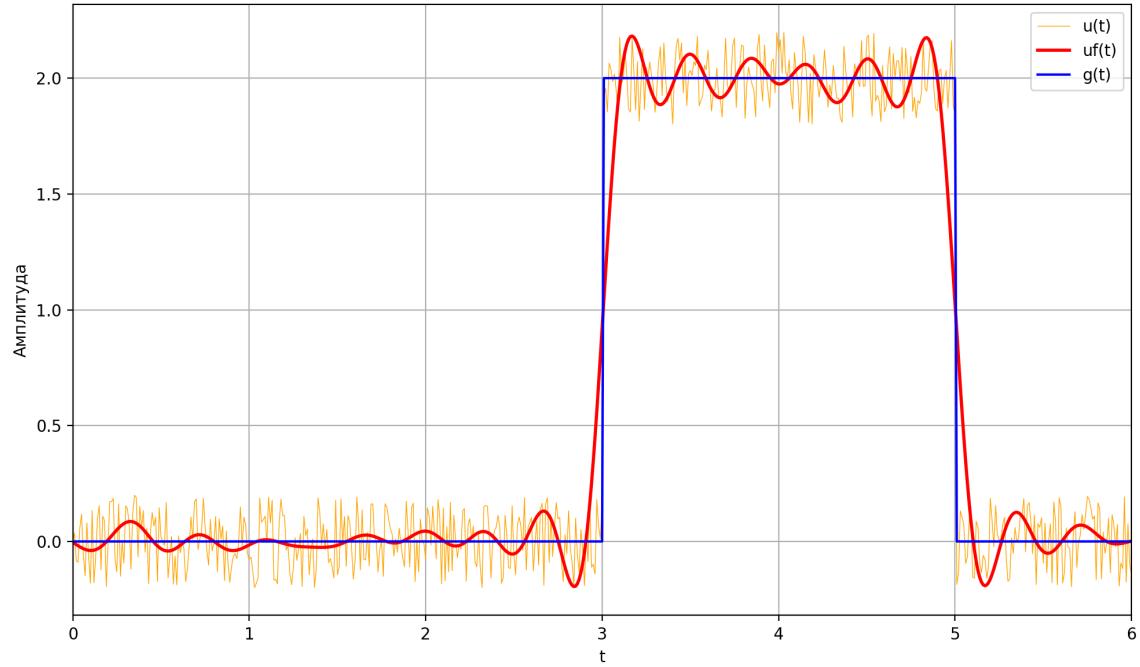
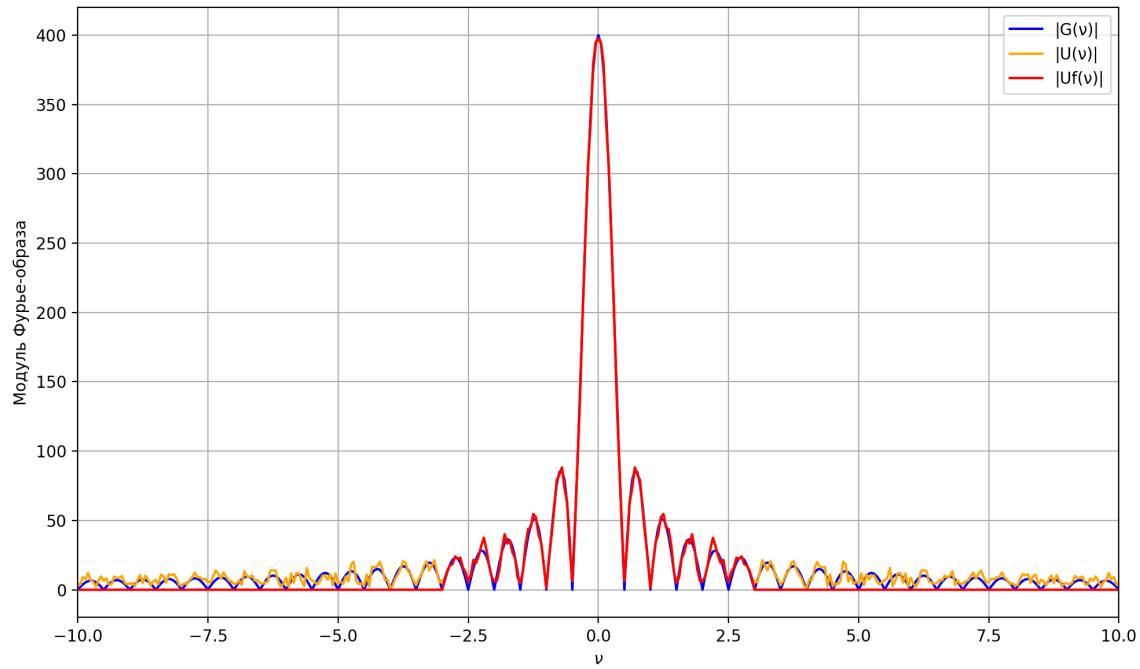
Для этого задания выберем $u(t)$ с коэффициентом $c = 0$. Найдем Фурье-образ получившейся функции для параметра $b = 0.2$.

$$u(t) = g(t) + b\xi(t)$$

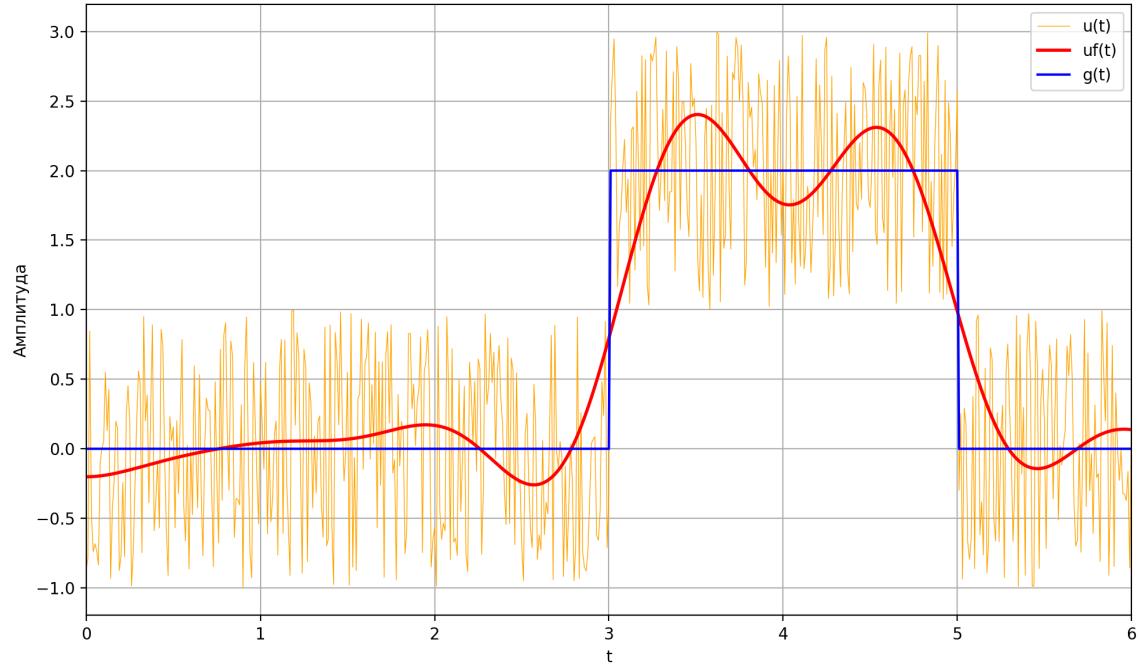
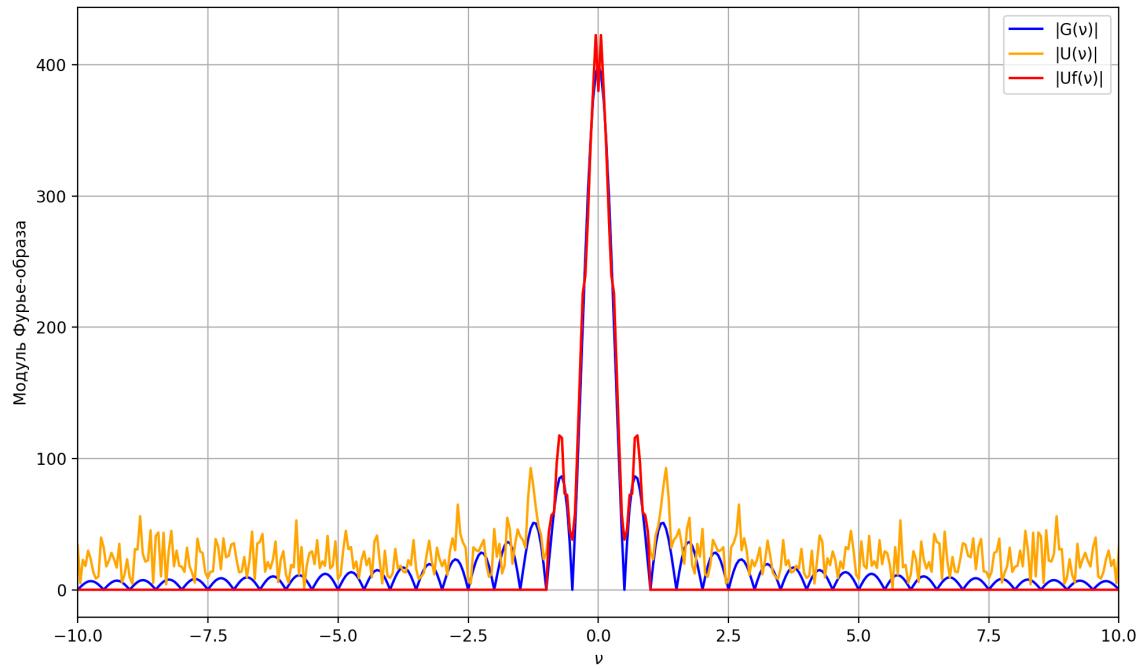
После этого применим наш фильтр с параметром верхней границы $v_0 = 1$ а следом за ним применим обратное преобразование. Теперь мы можем оценить результат при разных значениях параметров v_0 и b .

Рис. 2: Все версии графика при $v_0 = 1, b = 0.2$ Рис. 3: Все версии Фурье образа при $v_0 = 1, b = 0.2$

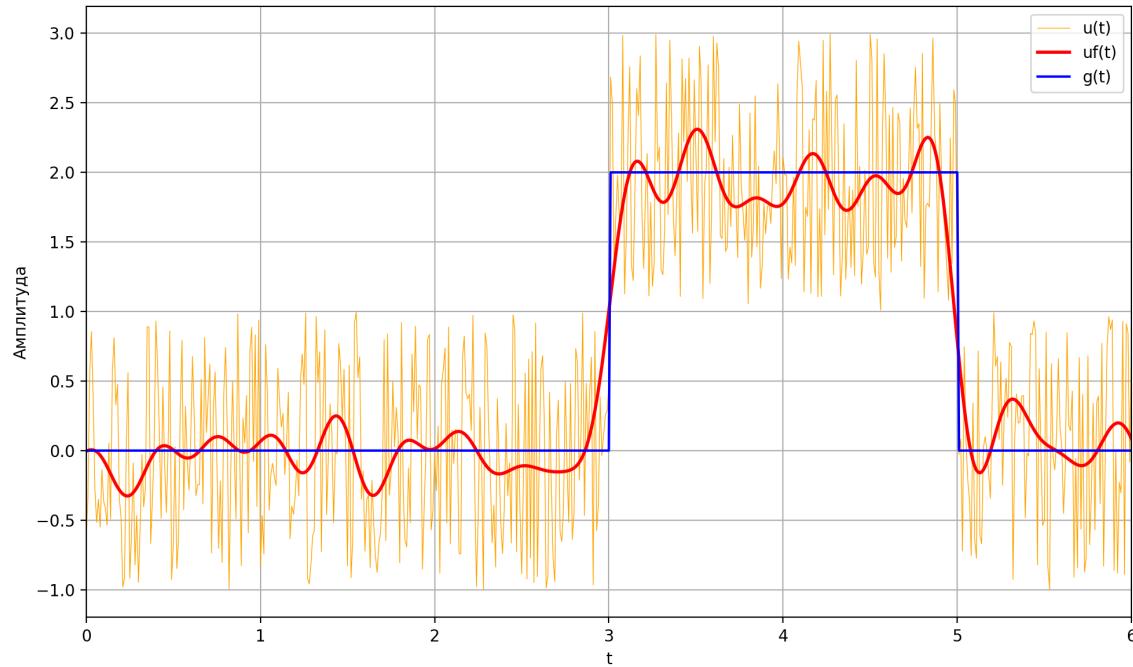
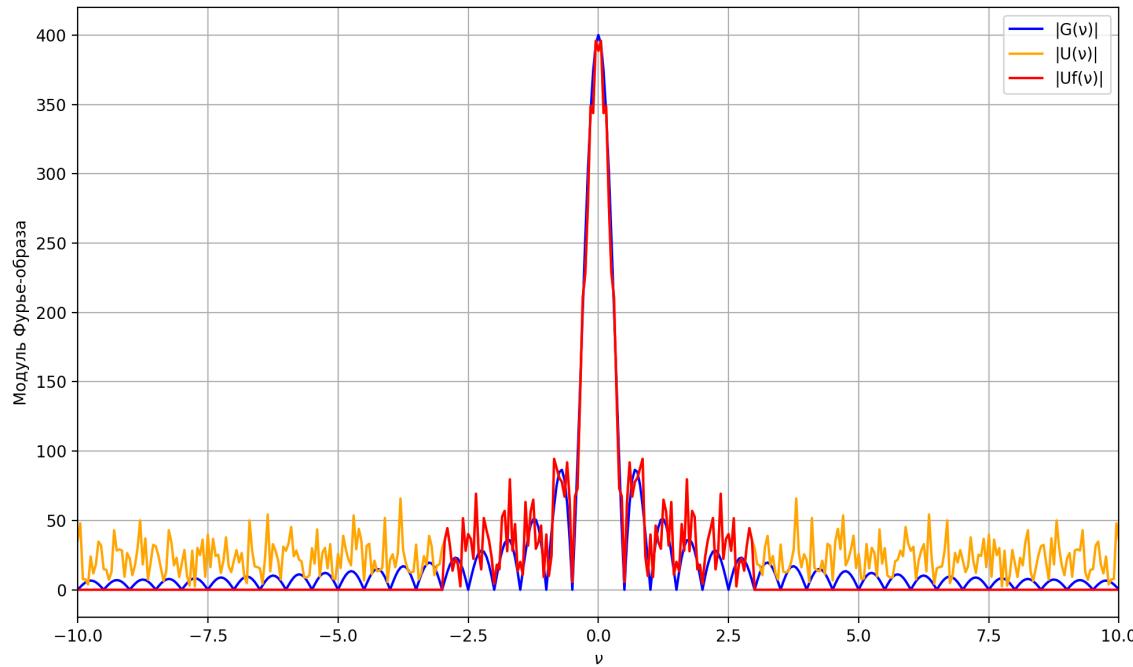
Результат вышел не очень. Все таки порог частоты $v_0 = 1$ слишком узок, чтобы сохранить достаточно частот для воспроизведения изначального графика. Но, стоит отметить, что "нулевые" участки достаточно ровные для сигнала с помехами - это заслуга низкого значения параметра b .

Рис. 4: Все версии графика при $v_0 = 3$, $b = 0.2$ Рис. 5: Все версии Фурье образа при $v_0 = 3$, $b = 0.2$

Пример лучшего сочетания параметров. $v_0 = 3$ позволяет собрать достаточно информации для уверенной схожести с $g(t)$, а низкий уровень b не дает функции сильно искажаться. Спойлер, но это будет лучший результат из всех в этой подборке.

Рис. 6: Все версии графика при $v_0 = 1$, $b = 1$ Рис. 7: Все версии Фурье образа при $v_0 = 1$, $b = 1$

Пример худшей комбинации параметров. Узкий порог v_0 не позволяет четко передать оригинальную функцию, а высокий уровень b создает мощные помехи на всех участках. В такой ситуации догадаться об оригинальной природе функции можно лишь визуально и без четких приближений параметров функции, таких как t_0 , t_1 и a .

Рис. 8: Все версии графика при $v_0 = 3$, $b = 1$ Рис. 9: Все версии Фурье образа при $v_0 = 3$, $b = 1$

Хороший результат, порог достаточно широк для воспроизведения прямоугольной волны, но шум от высокого b вносит свое сильное влияние на "плоских" участках графиках.

Подведу небольшой итог: при выборе порога фильтрации важно соблюдать баланс, ведь слишком низкое его значение губит картину функции вообще, а слишком высокий почти не даст полезной разницы в чистоте в сравнении с оригинальной функцией

Задание 1.2. Убираем специфические частоты

Теперь же обратимся к другому способу фильтрации частот. В этом задании мы будем использовать полноценную функцию со всеми ненулевыми параметрами.

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

Избавляясь от помех будем при помощи комбинации фильтра низких частот и полосового фильтра. Полосовой обнуляет все частоты на определенном промежутке, это крайне полезно при нейтрализации влияния гармонических колебаний, как тех, что имеются у нас. Приступим к исследованию. Изучим, как влияют параметры b , c , d на успешность фильтрования с помощью совмещенного фильтра.

Начнем со случая, с $b = 0$, это будет обычная функция $f(t)$ но с гармонической составляющей. Здесь не будем применять фильтр низких частот чтобы посмотреть на работу конкретно полосового фильтра. Отмету, что обнулять частоты полосовым фильтром будем стараться ровно по той частоте, которая является гармоническими колебаниями - эта частота численно равна коэффициенту d деленному на 2π .

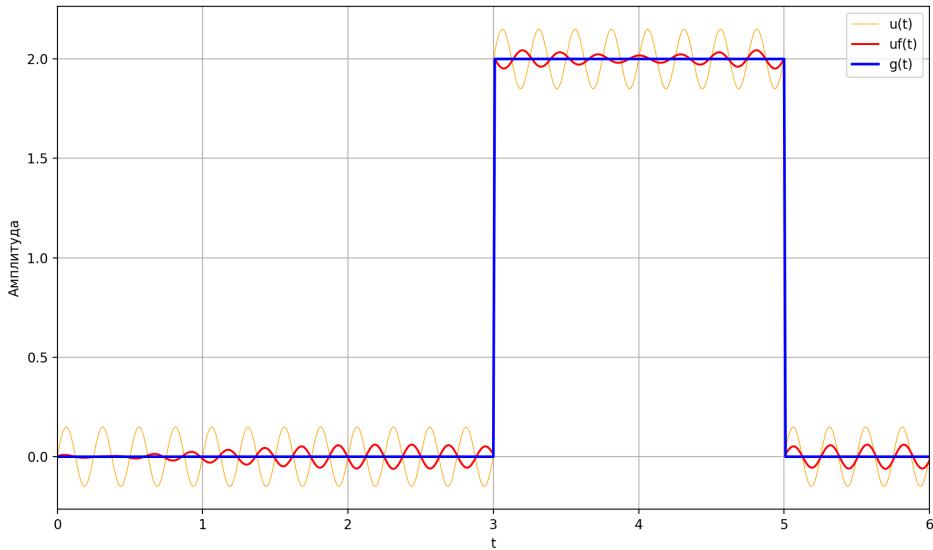


Рис. 10: Все версии графика при $b = 0$, $c = 0.15$, $d = 8\pi$

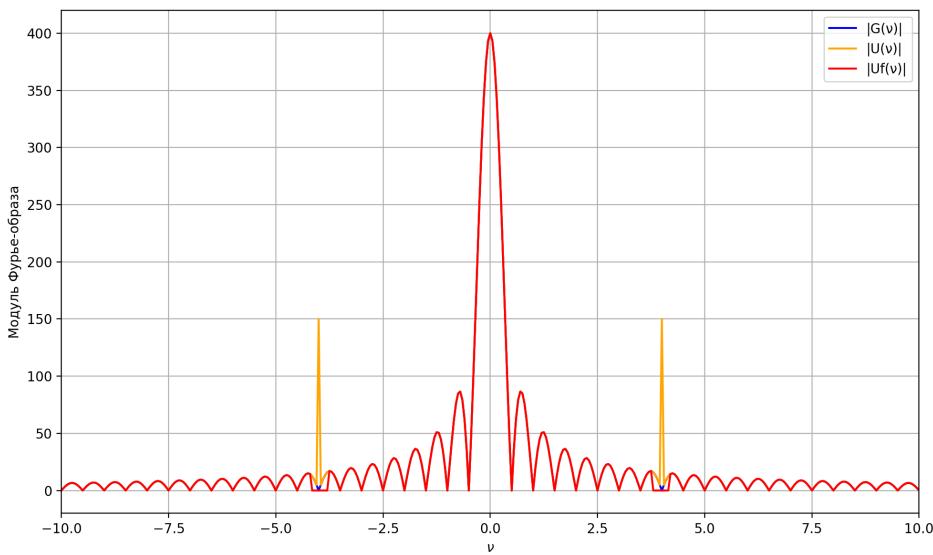


Рис. 11: Все версии Фурье образа при $b = 0$, $c = 0.15$, $d = 8\pi$

Результат хорош, единственная проблема в данном случае это неточное определение параметра δ , который отвечает за ширину области обнуления фильтра. Можно в идеале уменьшить эту погрешность, если точнее настроить δ , но я оставил эту неточность для наглядности сравнения параметров. Важно отметить, что окончательно эту погрешность не убрать, ведь так или иначе фильтр забирает информацию о кусочке Фурье-образа «чистой функции», и без этого не обойтись.

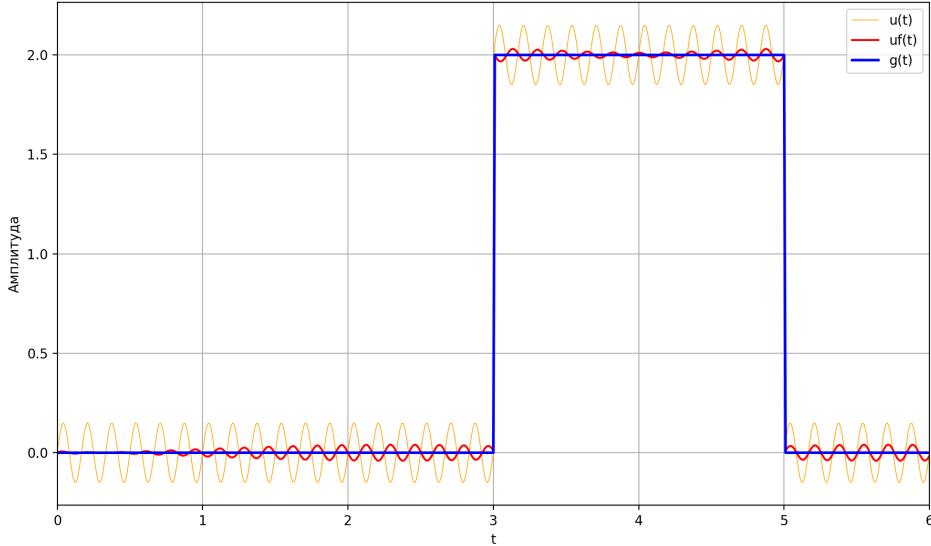


Рис. 12: Все версии графика при $b = 0, c = 0.5, d = 12\pi$

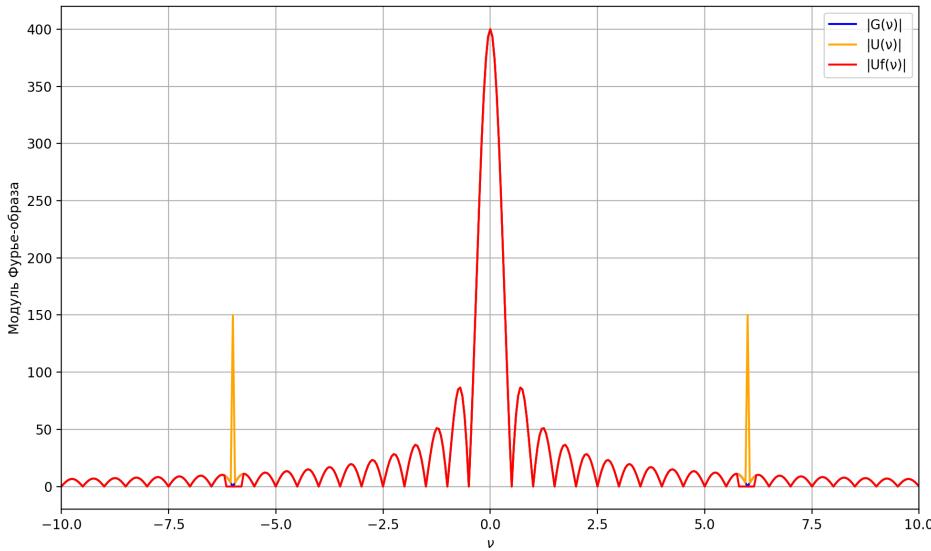
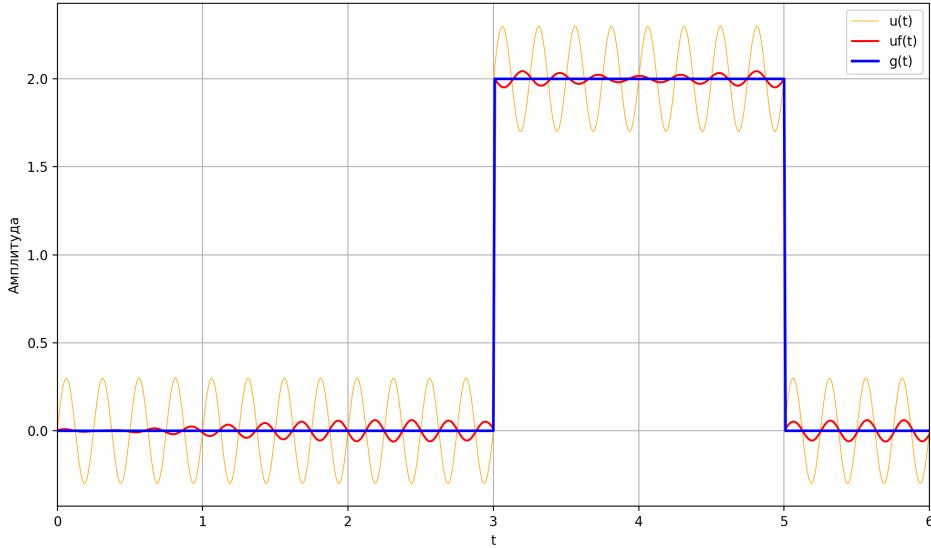
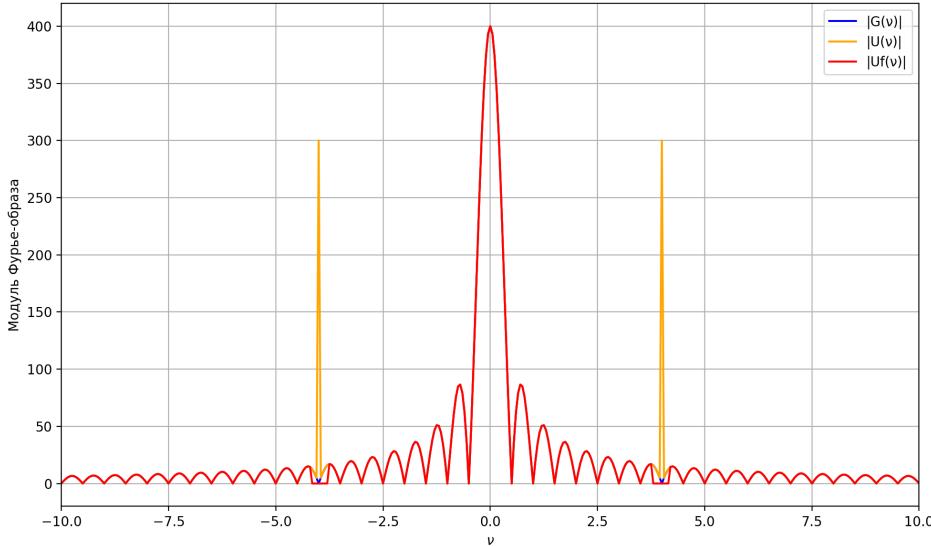


Рис. 13: Все версии Фурье образа при $b = 0, c = 0.5, d = 12\pi$

Разница между двумя экспериментами в измененном d . На практике в процессе изменились всего две вещи - амплитуда погрешностей в отфильтрованном графике повысилась, а в Фурье-образе сдвинулся участок обнуления функции (вслед за движением участка с частотой гармонического колебания). Опять же, причина погрешностей в итоговом варианте есть и увеличилась - во первых из за неточного δ , а во вторых, из за сдвинувшегося промежутка обнуления Фурье-образа, которое несло информацию об оригинальной функции.

Подведем небольшой итог по параметру d - глобально параметр будет лишь отвечать за то, информацию какого участка Фурье-образа мы потеряем в процессе полосовой фильтрации.

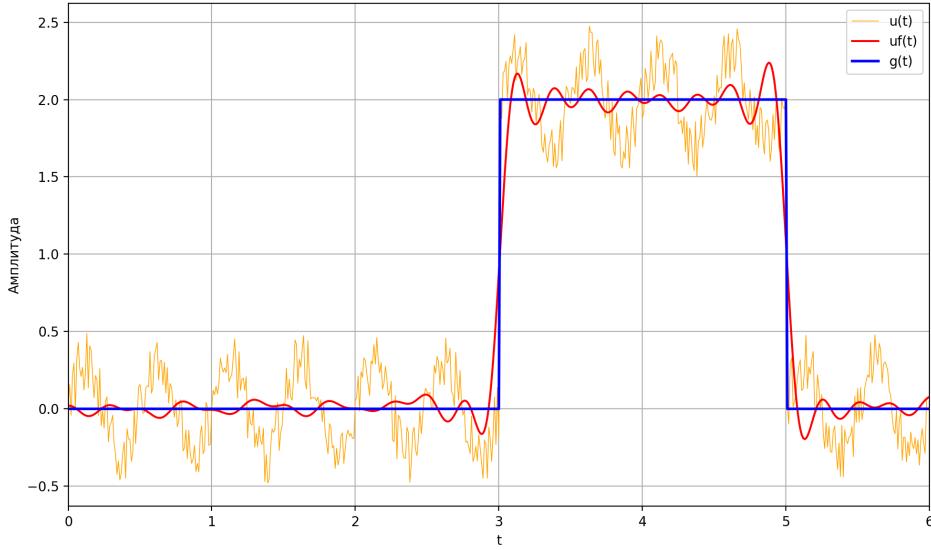
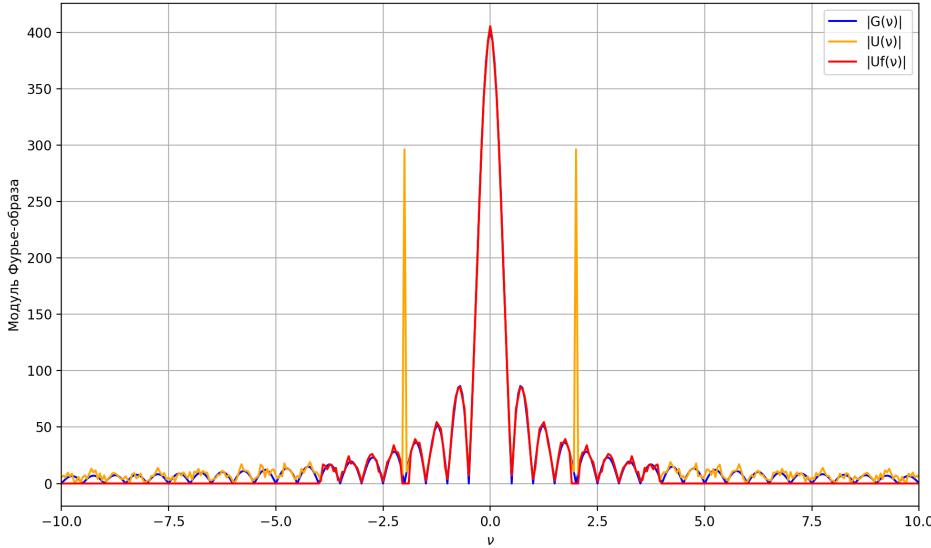
Рис. 14: Все версии графика при $b = 0$, $c = 0.3$, $d = 8\pi$ Рис. 15: Все версии Фурье образа при $b = 0$, $c = 0.3$, $d = 8\pi$

Приятные новости - параметр амплитуды гармонических колебаний c вообще не влияет на качество фильтрации помех. Полосовой фильтр гарантированно обнуляет все участки гармонического колебания на Фурье-образе, потому изменения амплитуды совершенно не имеют эффекта, кроме как увеличения амплитуды шума оригинальной функции $u(t)$.

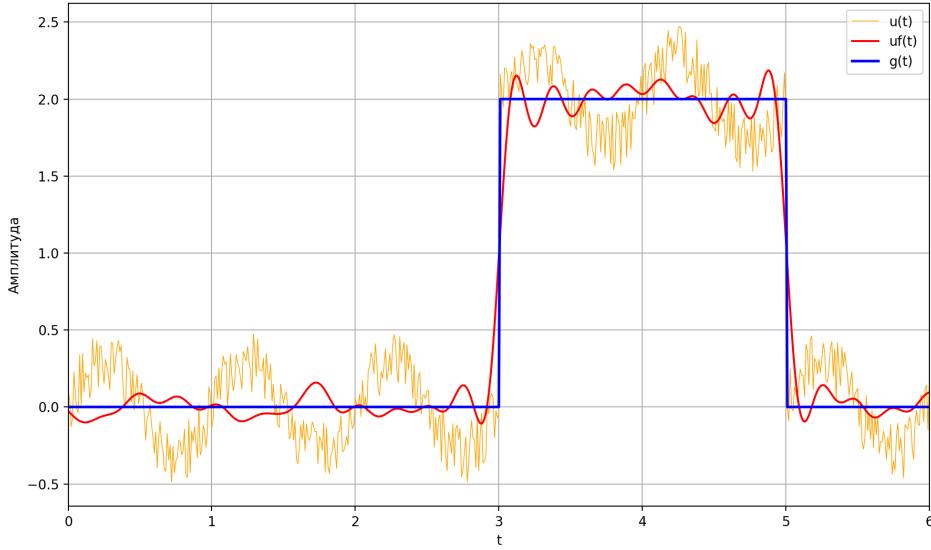
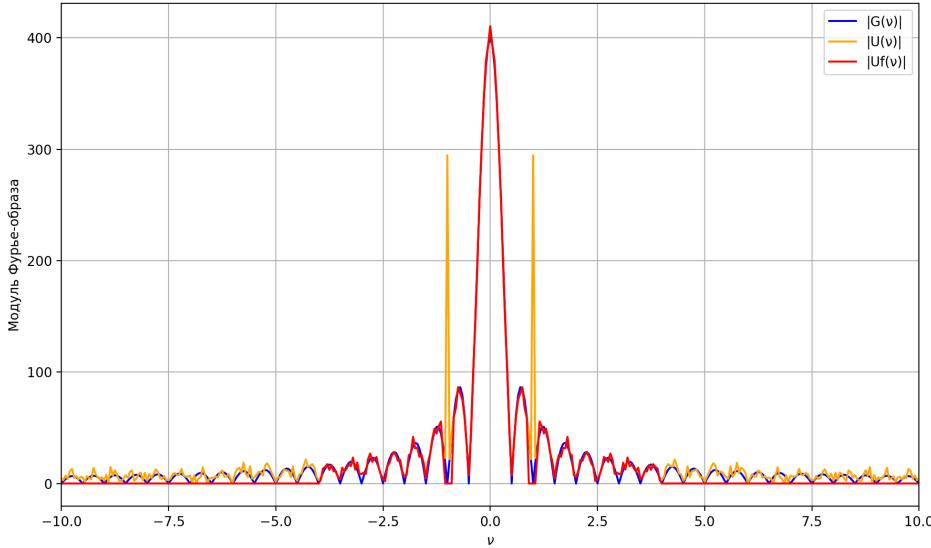
Параметр c может повлиять на качество фильтрации, если в качестве $delta$ взято малое число, и тогда необнуленные участки будут оставлять шум, но в нашем случае $delta$ подобрана достаточно хорошо для почти полного нивелирования этого эффекта. Поэтому в дальнейших экспериментах мы больше не будем рассматривать эффект влияния разных c , будем везде брать c равную 0.15.

Теперь же рассмотрим работу полосового фильтра при разных ненулевых значениях b .

Уже тут будем применять совмещенный фильтр. Чтобы частота обнуления полосового фильтра не накладывалась на фильтр низких частот, возьмем коэффициент d гораздо меньше.

Рис. 16: Все версии графика при $b = 0.2, c = 0.3, v_0 = 4, d = 4\pi$ Рис. 17: Все версии Фурье образа при $b = 0.2, c = 0.5, v_0 = 4, d = 4\pi$

Результат вышел очень хорошим - полосовой фильтр погасил шумы от гармонических колебаний, а случайные шумы сильно убавились за счет фильтра низких частот. Общая картина функции передана верно. Попробуем дальше экспериментировать с параметрами. Что будет если поменять параметр d ?

Рис. 18: Всё версии графика при $b = 0.2, c = 0.3, v_0 = 4, d = 2\pi$ Рис. 19: Всё версии Фурье образа при $b = 0.2, c = 0.3, v_0 = 4, d = 2\pi$

Разницы почти не видно, но все же результат стал хуже, оно и понятно, ведь низкие частоты более сильно влияют на общий контур функции и потеря промежутка шириной δ ближе к центру приведет к более заметному эффекту.

Изменять параметр нижней границы v_0 не вижу смысла - его основной эффект хорошо был продемонстрирован в задании 1.1.

Попробуем подвести итоги. Параметр b конечно же самый важный, ведь полосовой фильтр совершенно не предназначен для устранения случайных шумов на всей «частотной прямой» Фурье-образа. Параметр d влияет на конечный результат в меньшей степени, но чем он меньше, тем более важной «информацией» о функции придется жертвовать. Параметр c вообще не имеет эффект при верном подборе δ при фильтрации.

Задание 1.3. Убираем низкие частоты?

В этом задании попробуем перевернуть задачу из первого пункта - возьмем и обнулим все частоты ниже какого либо порога. Посмотрим, что мы сохраним при таком раскладе и имеет ли это какой либо смысл.

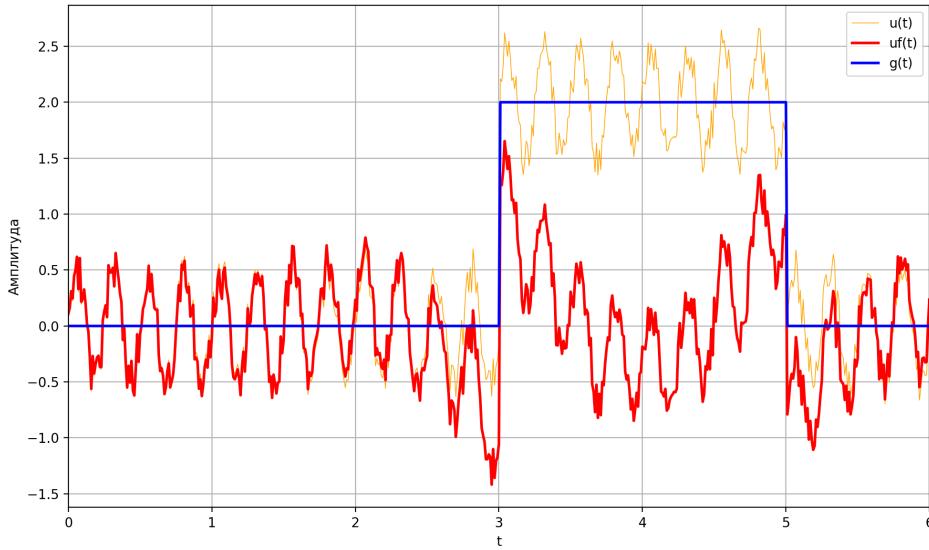


Рис. 20: Все версии графика при $b = 0.2$, $c = 0.5$, $d = 8\pi$

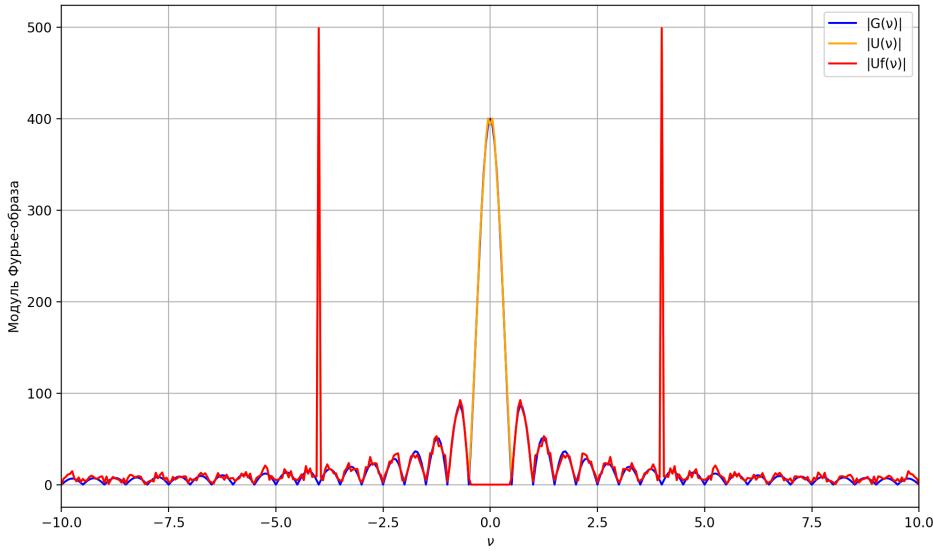
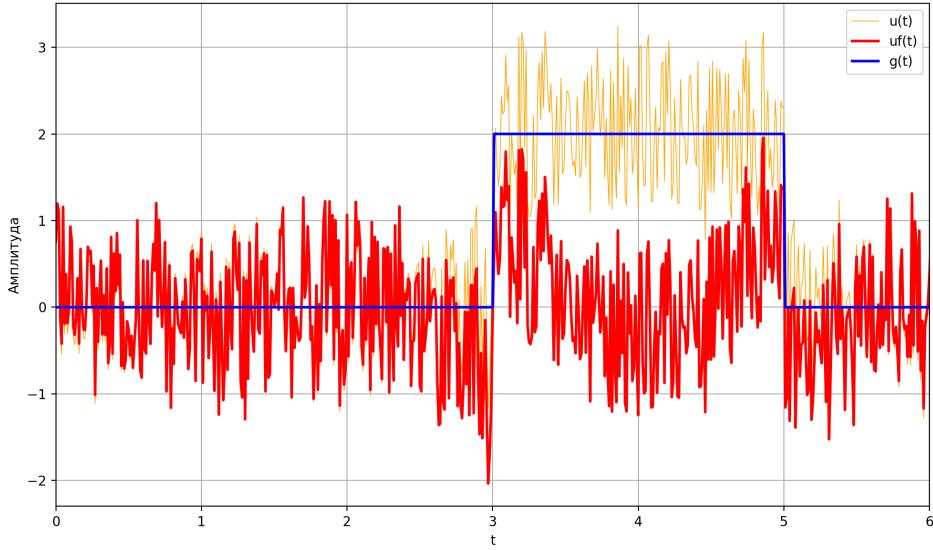
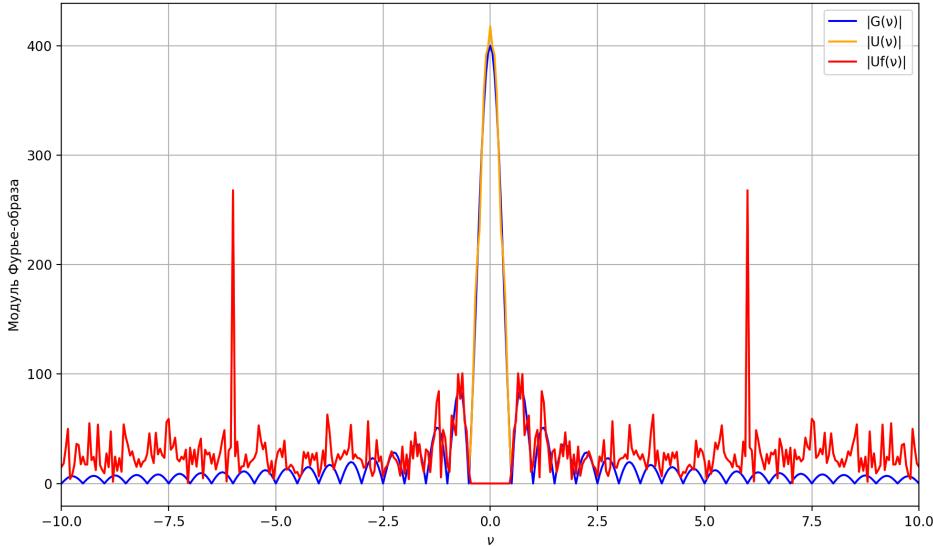
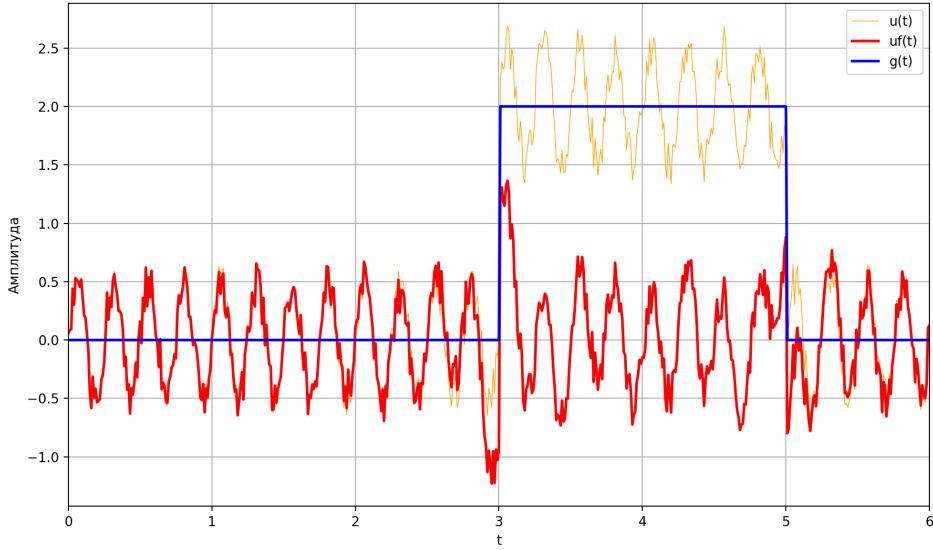
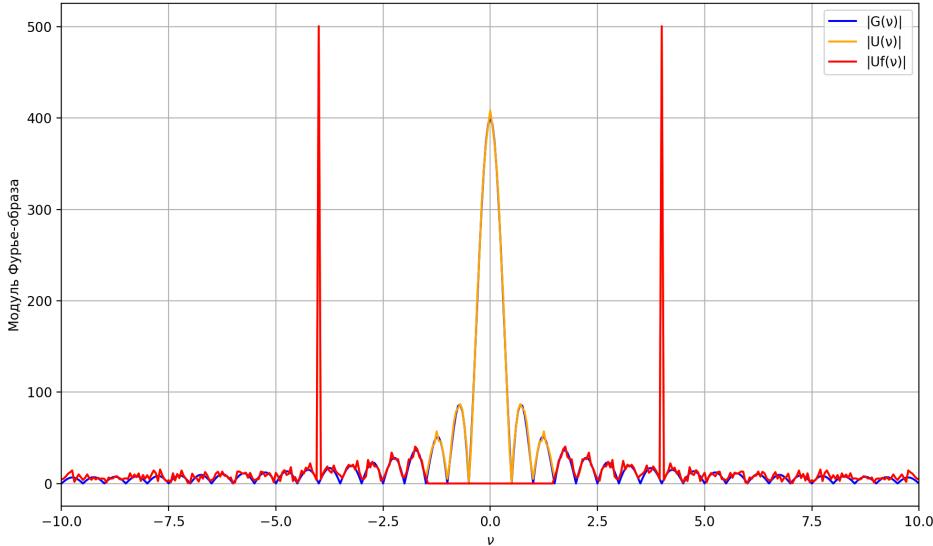


Рис. 21: Все версии Фурье образа при $b = 0.2$, $c = 0.5$, $d = 8\pi$

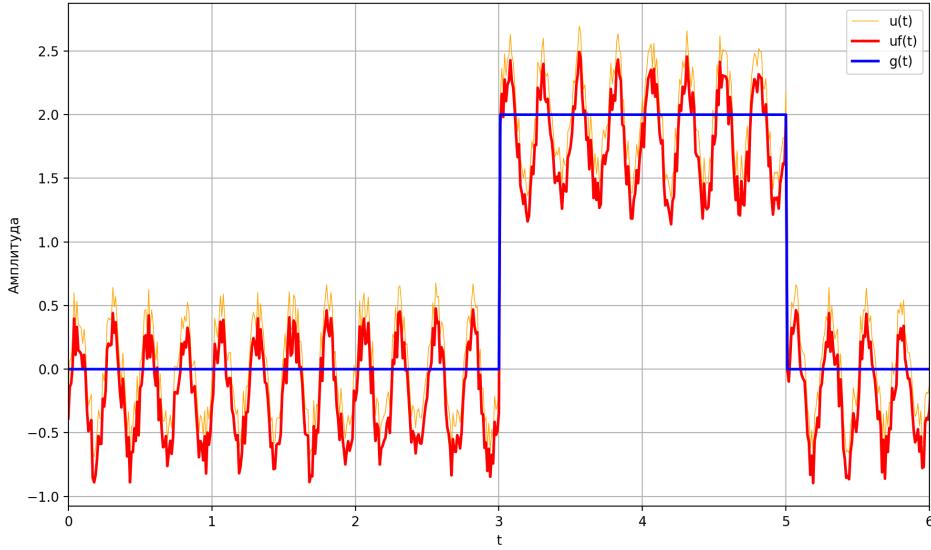
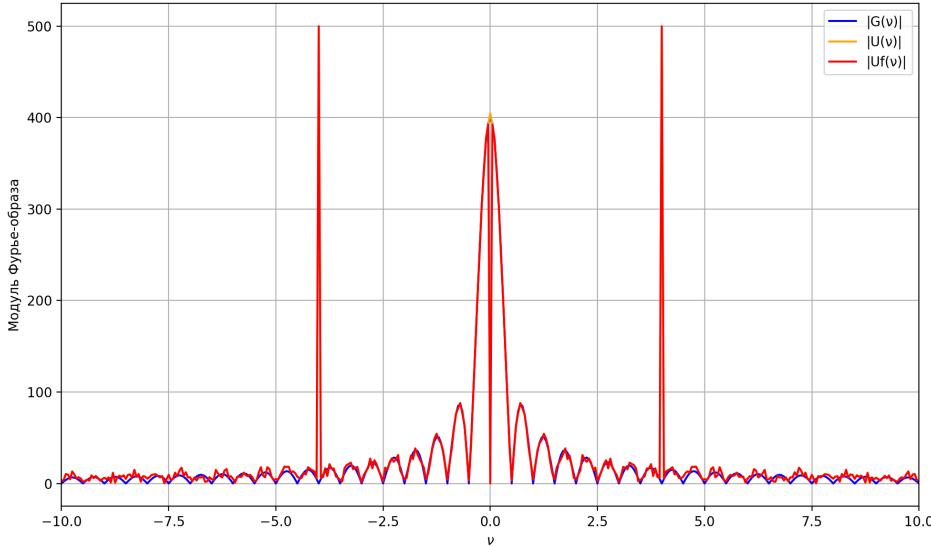
Протестируем фильтр с порогом $v_0 = 0.5$ на усредненной функции с невысокими параметрами силы обоих видов помех(случайных и гармонических). Результат явно не приблизил нас к чистой функции, впрочем, это и не удивительно. То что в первом пункте мы пытались «спасти», в этом мы наоборот обнуляем. Можно сказать, что мы получили функцию, состоящую в основном из помех изначальной функции - с этим тоже можно поэкспериментировать.

Рис. 22: Все версии графика при $b = 1$, $c = 0.3$, $d = 12\pi$ Рис. 23: Все версии Фурье образа при $b = 1$, $c = 0.3$, $d = 12\pi$

Менять параметры ошибок нет особого смысла, от этого результат принципиально не меняется, просто итоговая функция становится более или менее шумной. И так и так функция не пригодна для использования. Попробуем лучше изменять значения порога v_0 фильтра.

Рис. 24: Все версии графика при $b = 0.2$, $c = 0.5$, $d = 8\pi$ Рис. 25: Все версии Фурье образа при $b = 0.2$, $c = 0.5$, $d = 8\pi$

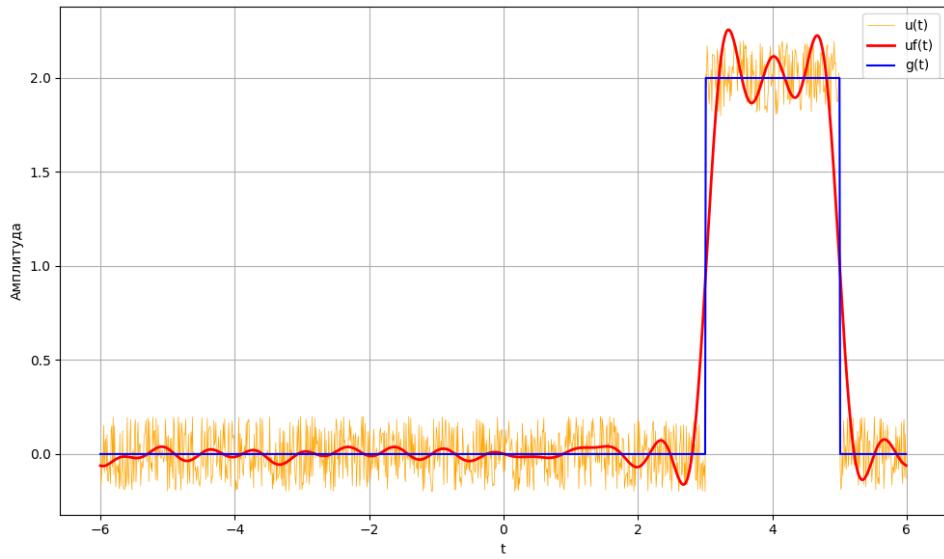
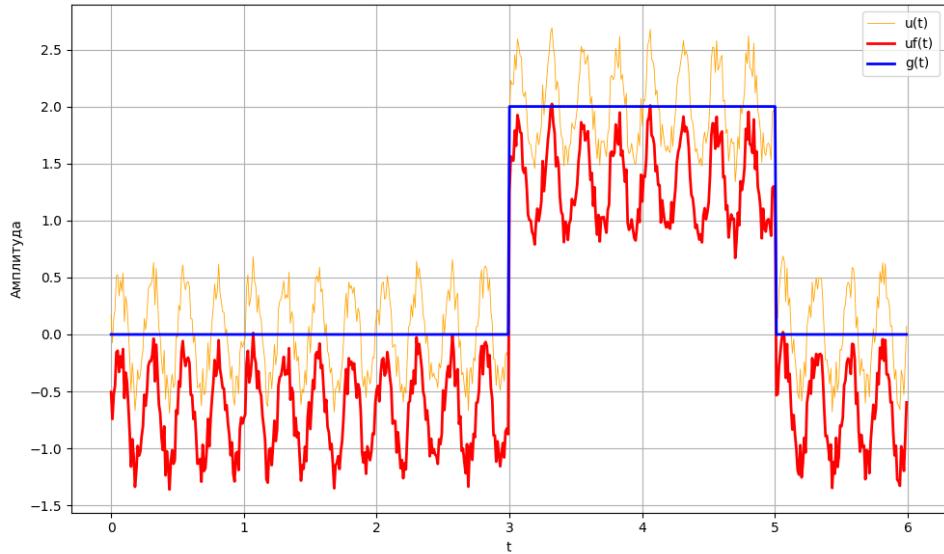
Увеличили порог до $v_0 = 1.5$ и теперь очень заметно, как прямоугольный характер функции почти полностью был скрыт. Это довольно примечательно, ведь при фильтре низких частот из первого пункта, значения $v_0 = 1.5$ было вполне достаточно, чтобы получить достаточно точные очертания оригинальной функции без гармонической компоненты помех (см. рис. 28 приложения). Увеличь мы порог еще больше, и функция бы полностью потеряла намеки на свою бывшую оригинальную природу. Все таки самые характерные особенности этой функции лежат ближе к центру Фурье-образа. Попробуем теперь взять значение порога намного ниже $v_0 = 0.001$.

Рис. 26: Все версии графика при $b = 0.2$, $c = 0.5$, $d = 8\pi$ Рис. 27: Все версии Фурье образа при $b = 0.2$, $c = 0.5$, $d = 8\pi$

Интересный результат - функция почти полностью повторяет $u(t)$ но за одним исключением. У $uf(t)$ наблюдается сдвиг вниз по сравнению с $u(t)$. Что примечательно - этот эффект будет таким же даже при крайне низких значениях порога(см. рис. 29 приложения). Видимо это потому, что за это «отвечает» участок Фурьеобраза на нулевых частотах.

Подводя итоги, можно сказать, что такой фильтр имеет место быть, если мы имеем дело с сигналом высокой частоты и при его передаче нам как раз мешают именно низкочастотные помехи. В случае нашей функции это полностью противопоказано, ведь Фурье-образ прямоугольной волны это кардинальный синус, и его пики приходятся именно на низкие частоты.

Приложение к пункту 1.3

Рис. 28: Пример с фильтром низких частот $v_0 = 1.5$ Рис. 29: Пример с фильтром высоких частот $v_0 = 0.001$

Задание 2. Фильтрация звука.

В этом задании мы применим изученный процесс фильтрации на практике. Загрязненную шумами аудиодорожку очистим и попробуем вычленить из нее только человеческий голос. Для этого нам надо подобрать подходящий фильтр и параметры для него. В процессе поиска информации я выяснил что лучше всего нам подойдет полосовой фильтр, который, в отличие от предшественника из прошлого задания, наоборот оставит только частоты из определенного промежутка. Этот промежуток от 300 до 3400 - именно в этом промежутке звучит человеческий голос в большинстве случаев, потому берем именно его.

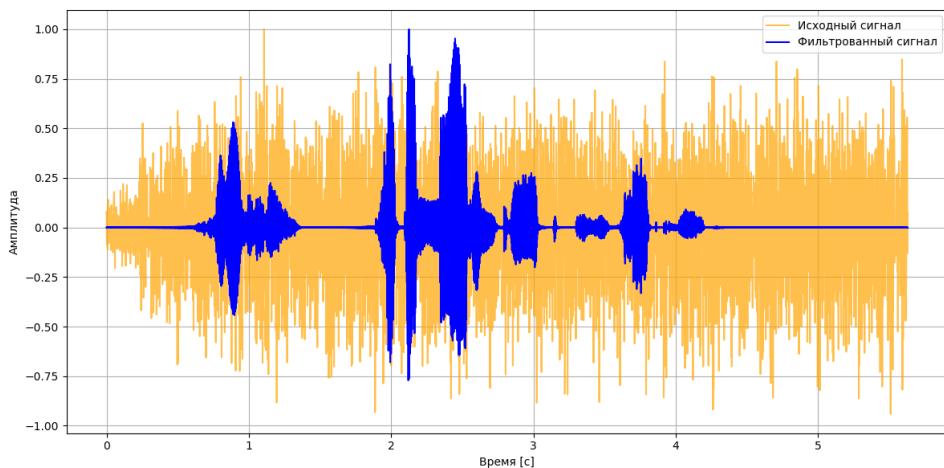


Рис. 30: Все версии звуковой дорожки

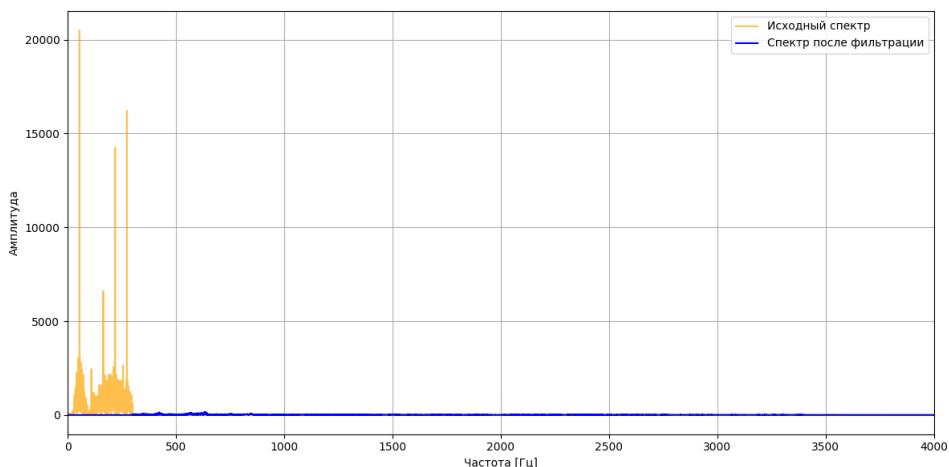


Рис. 31: Все версии Фурье-образа

Как и ожидалось, основной массив шумов приходится на низкие - их мы отсекаем, оставляем только неприметный, казалось бы по амплитуде участок, который и будет отвечать за человеческий голос. Прослушав результат можно точно сказать, что фильтр отработал точно как надо. Если предыдущая запись была забита шумами, то здесь их присутствие сведено к минимуму. Конечно есть определенные шумы на высоких и низких частотах и их можно было бы попробовать убрать конкретной для этой аудиозаписи, ведь спектр частот у каждого голоса разный, но в данном случае я решил взять универсальный промежуток, чтобы в случае чего можно было применить и для любого другого голоса.

(ссылка на [гугл диск](#) с записями)

Чтобы дополнительно опробовать фильтр попробую самостоятельно записать свой голос в шумном помещении коворкинга и попробовать его очистить.

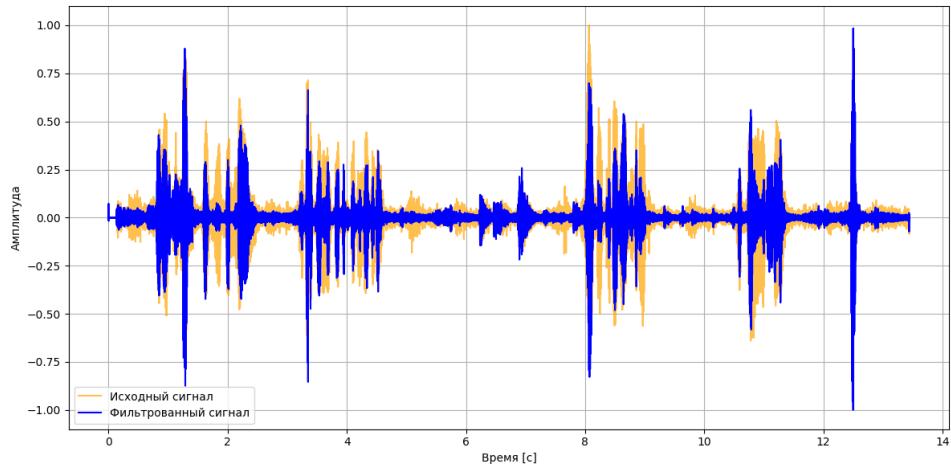


Рис. 32: Все версии звуковой дорожки

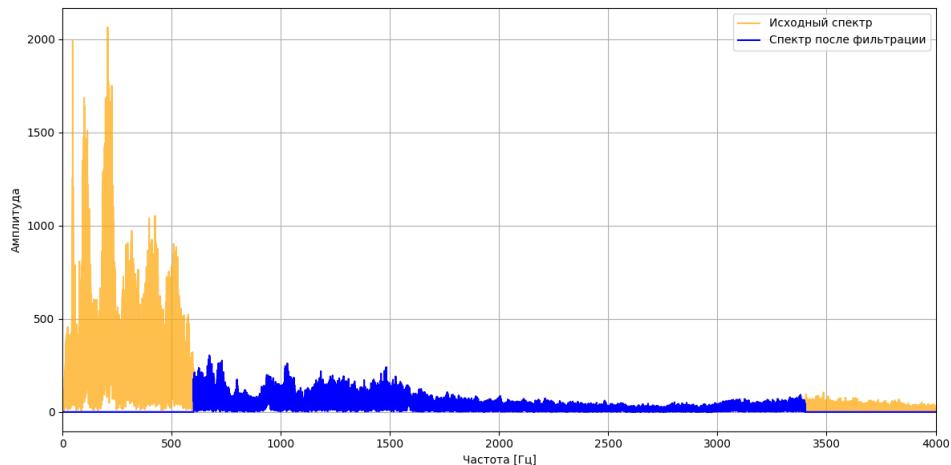


Рис. 33: Все версии Фурье-образа

Вот тут уже результат выглядит не так прелестно, как хотелось бы. Шумы в этой аудиозаписи более естественные и так же включают в себя человеческие голоса на фоне. Потому после фильтрации шумы по прежнему остаются частично, хоть и самые радикальные в «частотном» понимании того смысле слова были отсеяны. Что самое плохое, так это то что при этом конечная аудиодорожка звучит очень безжизненно, намного хуже чем в случае с примером для лабораторной работы. Но все же это все равно успех, хоть и не такой изящный.

Приложение с кодом для задания 1.

```

1 plt.figure(figsize=(10, 6))
2     plt.plot(t, u, label='u(t)', linewidth=.5, color="orange")
3     plt.plot(t, u_filtered, label='uf(t)', linewidth=2, color="red")
4     plt.plot(t, g, label='g(t)', linewidth=1.5, color="blue")
5     plt.xlim([0,6])
6     plt.legend()
7     plt.xlabel('t')
8     plt.ylabel('Амплитуда')
9     plt.grid()
10    plt.tight_layout()
11    plt.savefig(f'high_freq/high_freq_{b}_{nu}.png', dpi=200)
12    plt.show()
13
14
15 plt.figure(figsize=(10, 6))
16 plt.plot(v, np.abs(G), label='|G( )|', color="blue")
17 plt.plot(v, np.abs(U), label='|U( )|', color="orange")
18 plt.plot(v, np.abs(U_filtered), label='|Uf( )|', color="red")
19 plt.xlim([-10, 10])
20 plt.legend()
21 plt.xlabel('$\\nu$')
22 plt.ylabel('Модуль Фурье-образа')
23 plt.grid()
24 plt.tight_layout()
25 plt.savefig(f'high_freq/high_freq_spectre_{b}_{nu}.png', dpi=200)
26 plt.show()

```

Листинг 1: Код для отрисовки

```

1 t = np.arange(-10, 10, 0.01)
2 T = t.max() - t.min()
3 a = 2
4 N = len(t)
5 dt = t[1] - t[0]
6 t_1 = 3
7 t_2 = 5
8
9 g = np.zeros_like(t)
10 g[(t >= t_1) & (t <= t_2)] = a

```

Листинг 2: Код функции

```

1 bs = [1] # нужные значения параметров
2 nus = [3]
3
4 xi = np.random.uniform(-1, 1, size=t.shape)
5 for b in bs:
6     for nu in nus:
7         u = g + b * xi
8         U = np.fft.fftshift(np.fft.fft(u))
9         G = np.fft.fftshift(np.fft.fft(g))
10        v = np.fft.fftshift(np.fft.freq(N, d=T/N))
11        U_filtered = np.zeros_like(U)
12        ind = np.abs(v) <= nu
13        U_filtered[ind] = U[ind]
14        u_filtered = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered)))

```

Листинг 3: Код для задания 1.1

```

1 xi = np.random.uniform(-1, 1, size=t.shape)
2 bs = [0.2] # параметры
3 cs = [0.3]
4 ds = [1 * (2 * np.pi)]#, 4 * (2 * np.pi)]
5 vs = [4]
6
7 for b in bs:
8     for c in cs:
9         for d in ds:
10            for v0 in vs:
11                print(f"Параметры эксперимента: b = {b}, c = {c}, v0 = {v0}, d = {d / (2 * np.pi):.2f} Гц (частота гармоники)")
12
13                u = g + b * xi + c * np.sin(d * t)
14
15                U = np.fft.fftshift(np.fft.fft(u))
16                G = np.fft.fftshift(np.fft.fft(g))
17                v = np.fft.fftshift(np.fft.fftfreq(N, d=T / N))
18
19                freq_to_remove = d / (2 * np.pi)
20
21                delta = 0.1 # Ширина фильтрации
22                ind_remove = (np.abs(v - freq_to_remove) <= delta) | (np.abs(v + freq_to_remove) <=
delta)
23                U_filtered = U.copy()
24                U_filtered[ind_remove] = 0
25                ind_low = np.abs(v) >= v0
26                U_filtered[ind_low] = 0
27                u_filtered = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered)))

```

Листинг 4: Код для задания 1.2

```

1 xi = np.random.uniform(-1, 1, size=t.shape) # Случайный шум
2 bs = [0.2] # Амплитуда шума
3 cs = [0.5] # Амплитуда гармонической помехи
4 ds = [4 * (2 * np.pi)] # Частоты гармонической помехи
5
6 v0 = 0.5
7
8 for b in bs:
9     for c in cs:
10        for d in ds:
11            print(f"Параметры эксперимента: b = {b}, c = {c}, d = {d / (2 * np.pi):.2f} Гц (частота
гармоники)")
12            u = g + b * xi + c * np.sin(d * t)
13
14            U = np.fft.fftshift(np.fft.fft(u))
15            G = np.fft.fftshift(np.fft.fft(g))
16            v = np.fft.fftshift(np.fft.fftfreq(N, d=dt))
17
18            U_filtered = U.copy()
19            ind_remove = (np.abs(v) <= v0)
20            U_filtered[ind_remove] = 0
21
22            u_filtered = np.real(np.fft.ifft(np.fft.ifftshift(U_filtered)))

```

Листинг 5: Код для отрисовки

Приложение с кодом для задания 2.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.io import wavfile
4 from scipy.fftpack import fft, ifft, fftshift, ffftfreq
5 import sounddevice as sd
6
7 file_name = "MUHA.wav"
8 fs, audio = wavfile.read(file_name)
9
10 # преобразование вmono и нормализация для устакивания значений
11 if len(audio.shape) > 1:
12     audio = audio[:, 0]
13 audio = audio / np.max(np.abs(audio))
14
15 num_samples = len(audio)
16 time_step = 1 / fs
17 total_time = num_samples * time_step
18 time = np.linspace(0, total_time, num_samples)
19
20 freqs = fftshift(fftfreq(num_samples, time_step))
21 spectrum = fftshift(fft(audio))
22 amp_spectrum = np.abs(spectrum)
23
24 # Фильтр
25 low = 300
26 high = 3400
27 mask = (np.abs(freqs) >= low) & (np.abs(freqs) <= high)
28 filtered_spectrum = spectrum.copy()
29 filtered_spectrum[~mask] = 0
30
31 filtered_audio = np.real(ifft(fftshift(filtered_spectrum)))
32 filtered_audio = filtered_audio / np.max(np.abs(filtered_audio))
33
34
35 print("оригинальный звук")
36 sd.play(audio, fs)
37 sd.wait()
38
39 print("фильтрованный звук")
40 sd.play(filtered_audio, fs)
41 sd.wait()
42
43 output_name = "fMUHA.wav"
44 wavfile.write(output_name, fs, (filtered_audio * 32767).astype(np.int16))
45
46 #графики
47 plt.figure(figsize=(12, 6))
48 plt.plot(time, audio, label="Оригинал", alpha=0.7, color="orange")
49 plt.plot(time, filtered_audio, label="Фильтрованный", color="blue")
50 plt.xlabel("Время [с]")
51 plt.ylabel("Амплитуда")
52 plt.legend()
53 plt.grid()
54 plt.tight_layout()
55 plt.show()
56
57 plt.figure(figsize=(12, 6))
58 plt.plot(freqs, amp_spectrum, label="Оригинальный спектр", alpha=0.7, color="orange")
59 plt.plot(freqs, np.abs(filtered_spectrum), label="Фильтрованный спектр", color="blue")
60 plt.xlabel("Частота [Гц]")
61 plt.ylabel("Амплитуда")
62 plt.legend()
63 plt.grid()
64 plt.tight_layout()
65 plt.show()

```

Листинг 6: Код второго задания