

Лабораторная работа № 4.

Линейная фильтрация

Выполнил: Смирнов Алексей Владимирович — Р3242
409578

Содержание

Задание 1. Линейные фильтры.	1
Фильтр первого порядка	1
Фильтр второго порядка	9
Задание 2. Фильтрация биржевых данных	23
Листинги	26

Задание 1. Линейные фильтры.

Рассмотрим сигнал $g(t)$:

$$g(t) = \begin{cases} a, & t \in [1, 2] \\ 0, & t \notin [1, 2] \end{cases}$$

и его зашумленную версию:

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

где $\xi(t) \sim \mathcal{U}[-1, 1]$, а значения b, c, d - параметры возмущения.

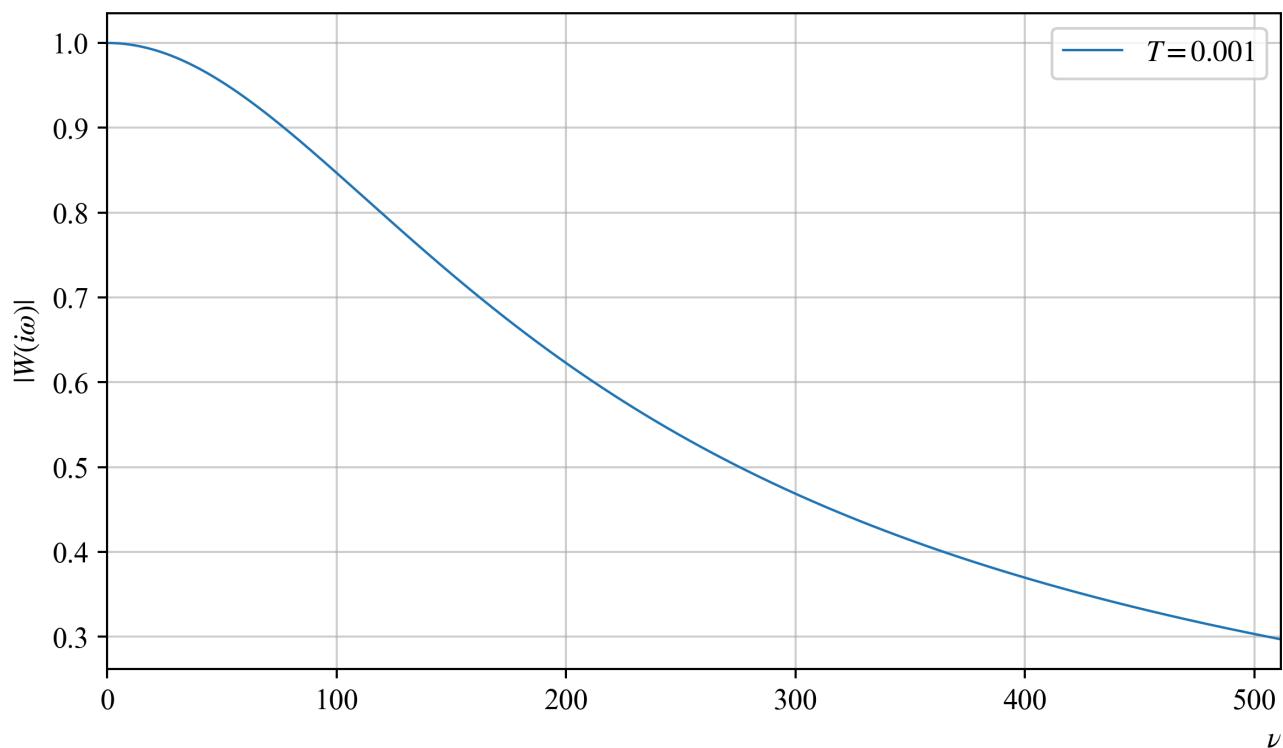
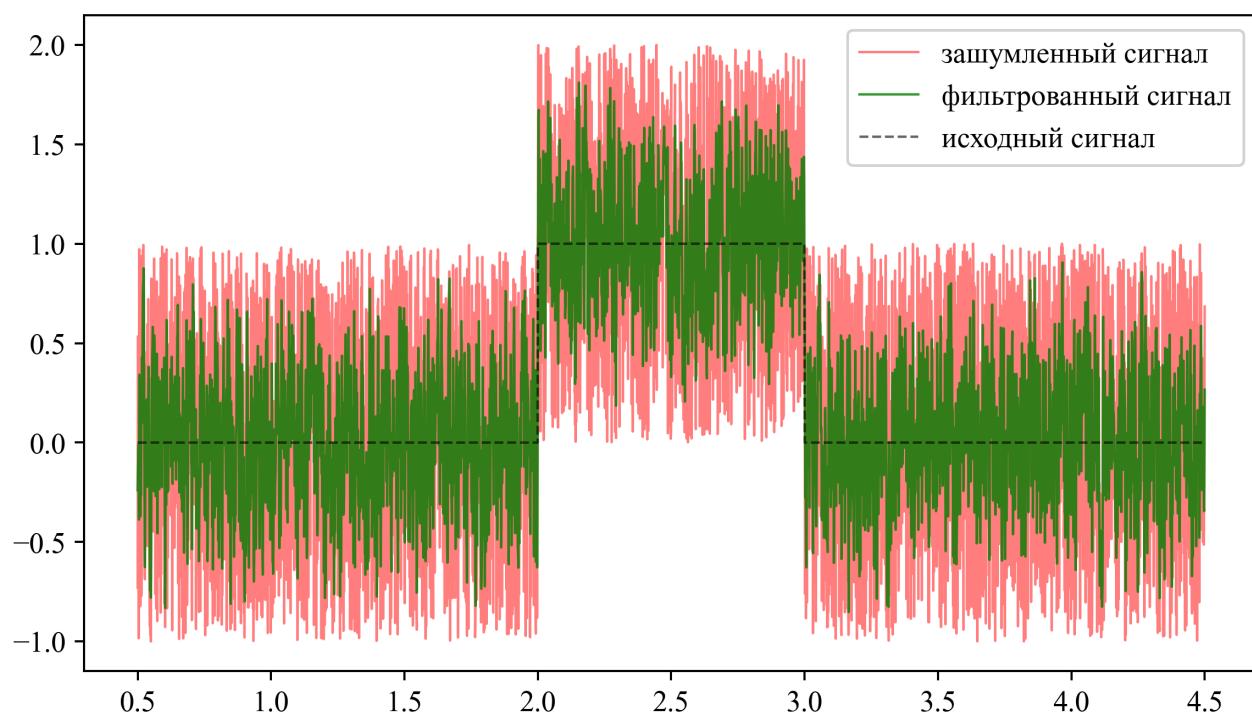
Фильтр первого порядка

Приняли $c = 0$. Зададим некоторую постоянную T и пропустим сигнал через фильтр первого порядка

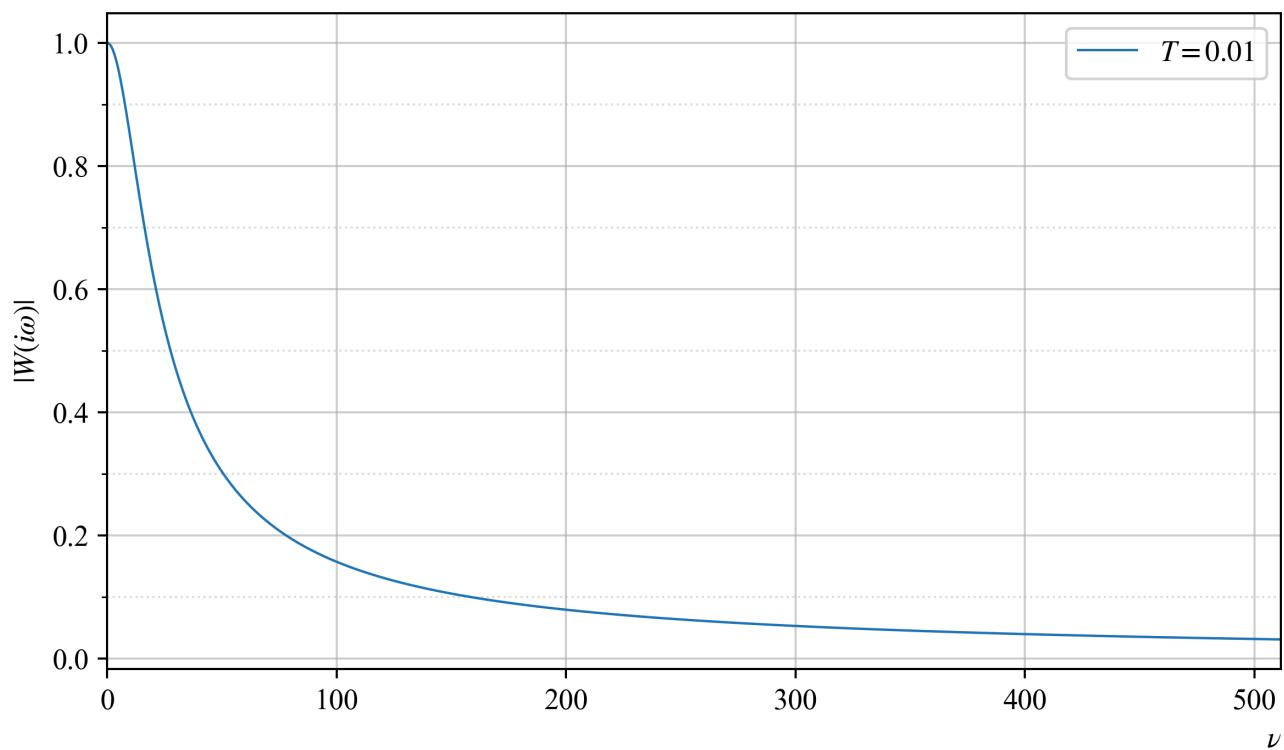
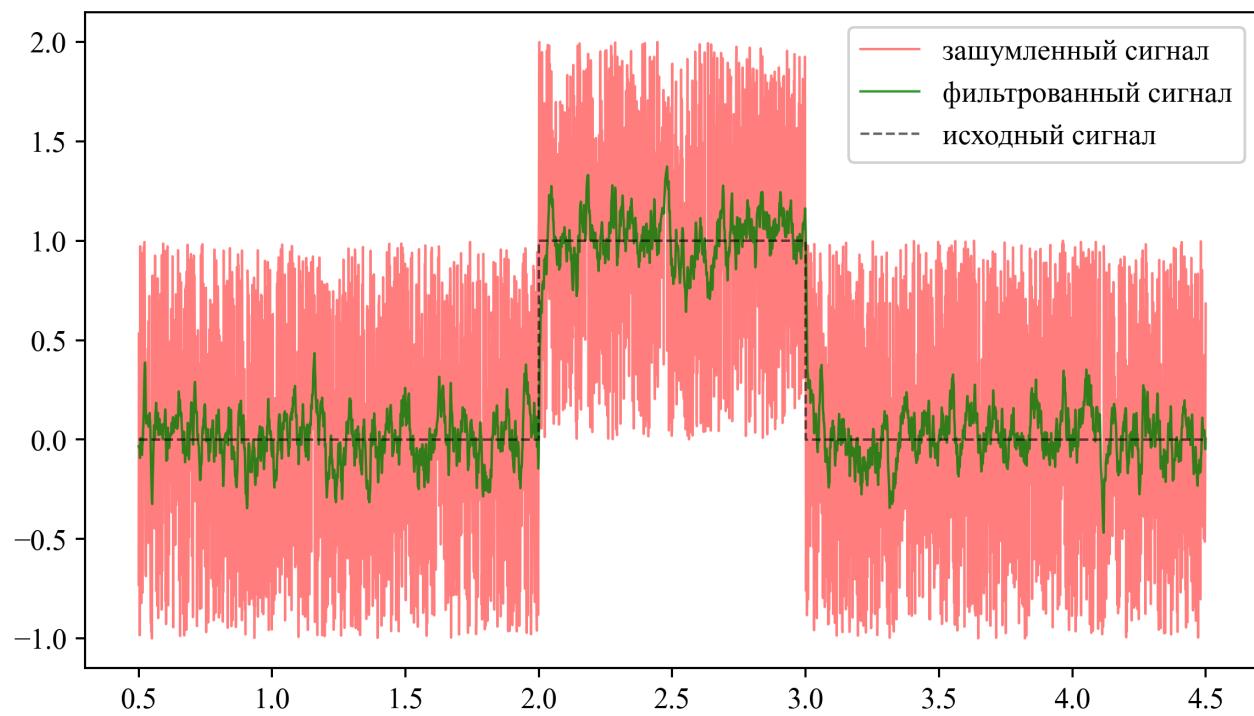
$$W_1(p) = \frac{1}{Tp + 1}$$

Исследуем на что влияет параметр T

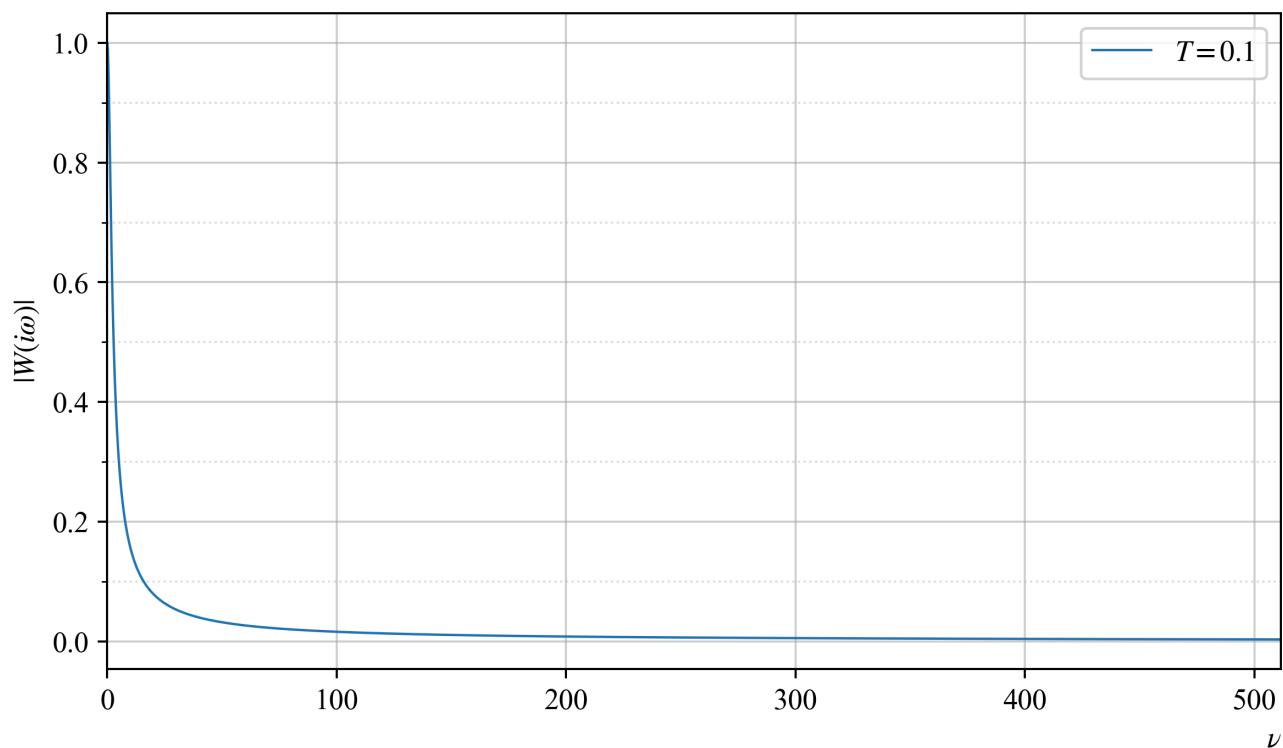
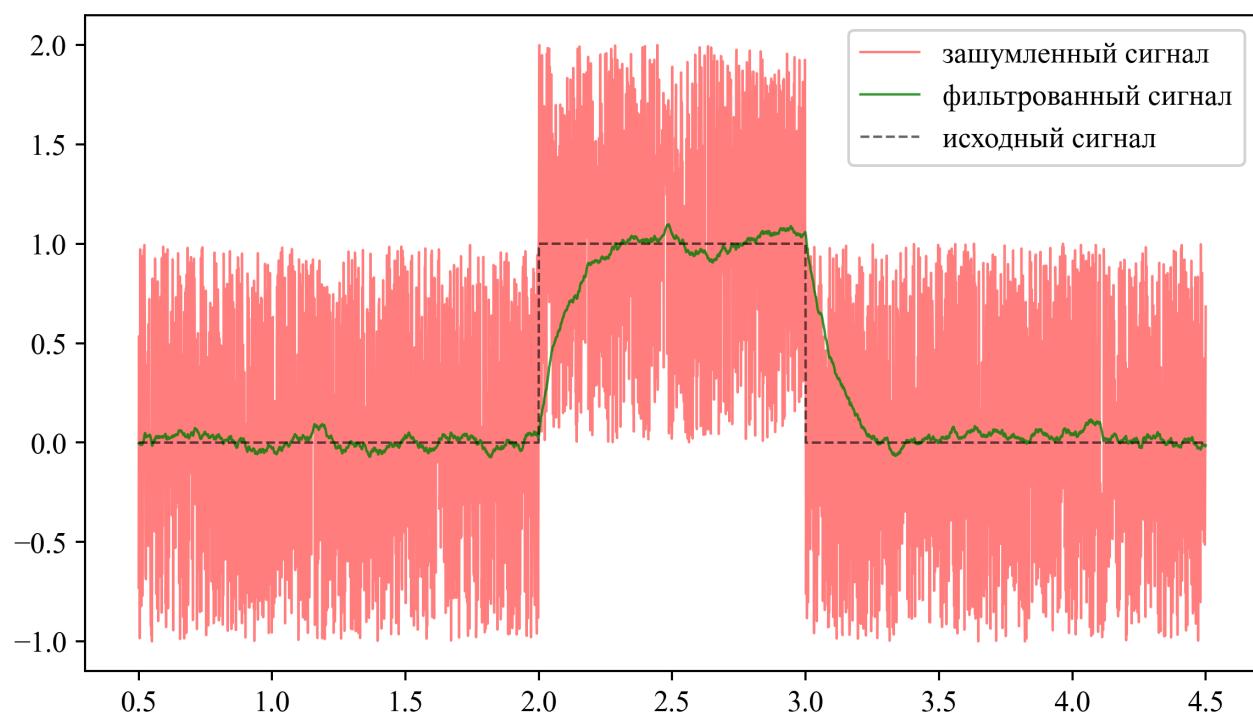
$$T = 0.001$$

Рис. 1. АЧХ фильтра первого порядка с $T = 0.001$ Рис. 2. Сравнение исходного и фильтрованного сигнала при $T = 0.001$

$T = 0.01$

Рис. 3. АЧХ фильтра первого порядка с $T = 0.01$ Рис. 4. Сравнение исходного и фильтрованного сигнала при $T = 0.01$

$T = 0.1$

Рис. 5. АЧХ фильтра первого порядка с $T = 0.1$ Рис. 6. Сравнение исходного и фильтрованного сигнала при $T = 0.1$

По графикам видно, что временной параметр T влияет на плавность фильтрованного сигнала. Слишком маленькое значение не фильтрует белый шум на достаточном уровне, а слишком большое значение T меняет форму сигнала, помимо удаления шумов.

Получили наилучший результат при $T = 0.01$.

Теперь исследуем на что влияет параметр a

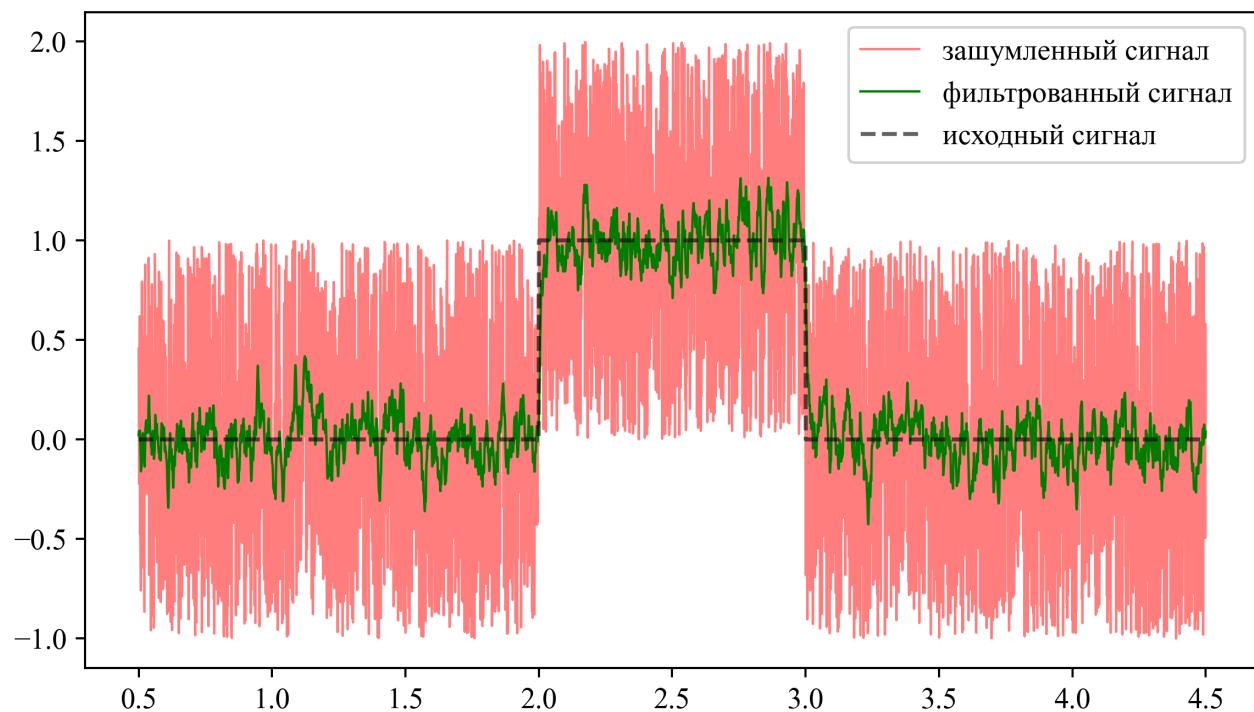
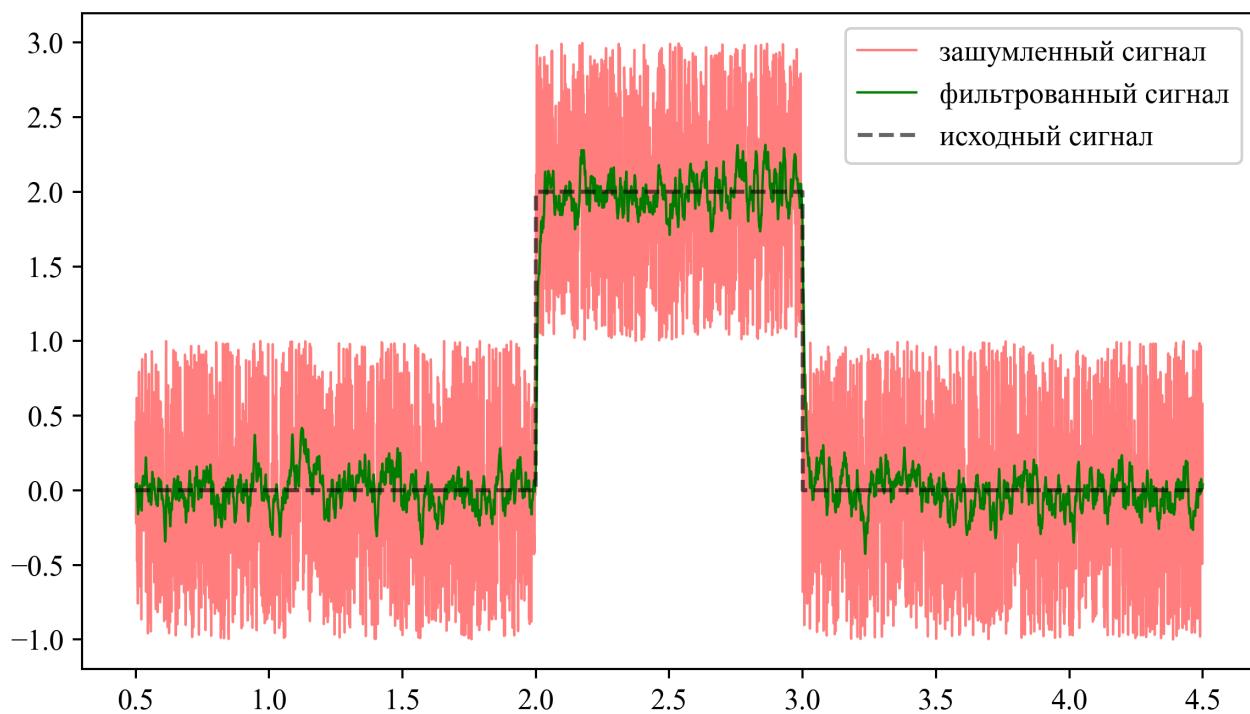
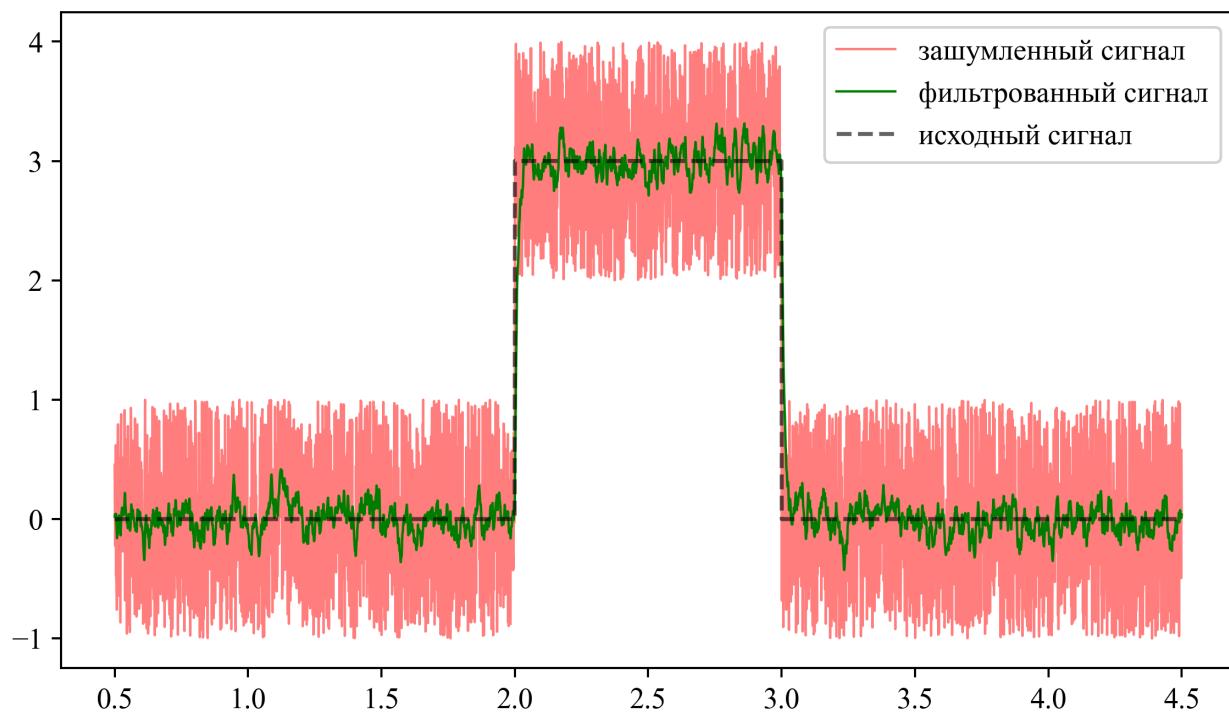


Рис. 7. Сравнение исходного и фильтрованного сигнала при $a = 1$

Рис. 8. Сравнение исходного и фильтрованного сигнала при $a = 2$ Рис. 9. Сравнение исходного и фильтрованного сигнала при $a = 3$

При постоянном параметре T белый шум проще удалить, если его амплитуда меньше амплитуды сигнала a и сложнее удалить, если величина шума $b \geq a$.

Попробуем получить такой же результат как и фильтрованный сигнал, умножив его передаточную функцию $W_1(i\omega)$ на Фурье-образ зашумленного сигнала. Частоты далекие от нуля на АЧХ после произведения должны исчезнуть или уменьшиться, после обратного преобразования, а близкие к нулю не изменятся.

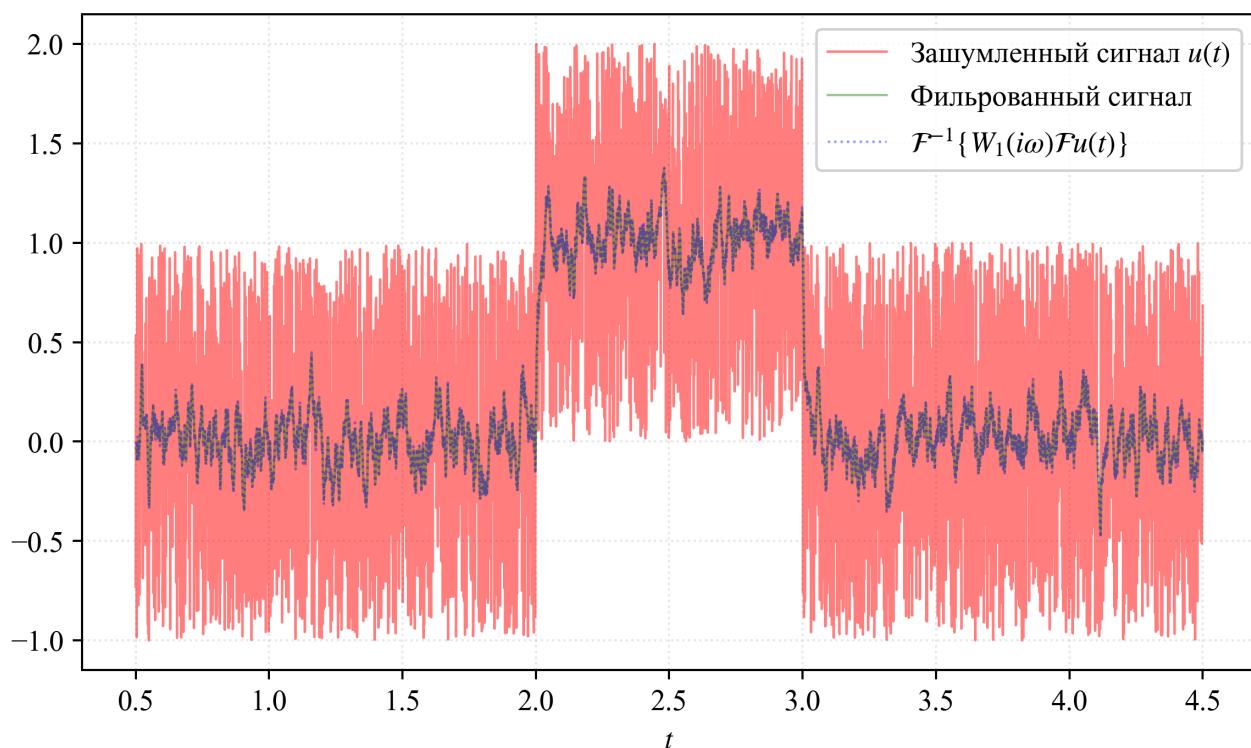


Рис. 10. Сравнение фильтрованного сигнала и сигнала полученного произведением АЧХ и образа зашумленного сигнала

Полученный таким образом сигнал полностью совпал с фильтрованным сигналом.

Теперь сравним Фурье-образы зашумленного, исходного и фильтрованного сигналов, чтобы увидеть как фильтр влияет на разные частоты.

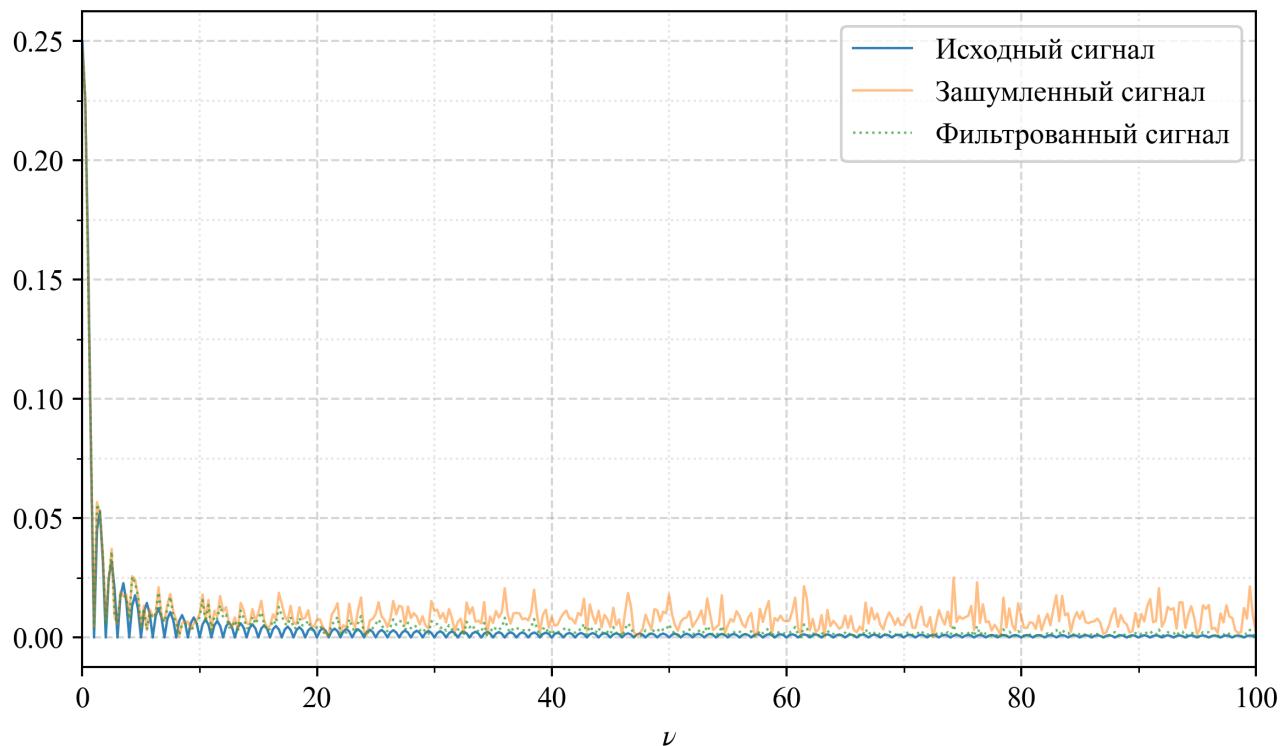
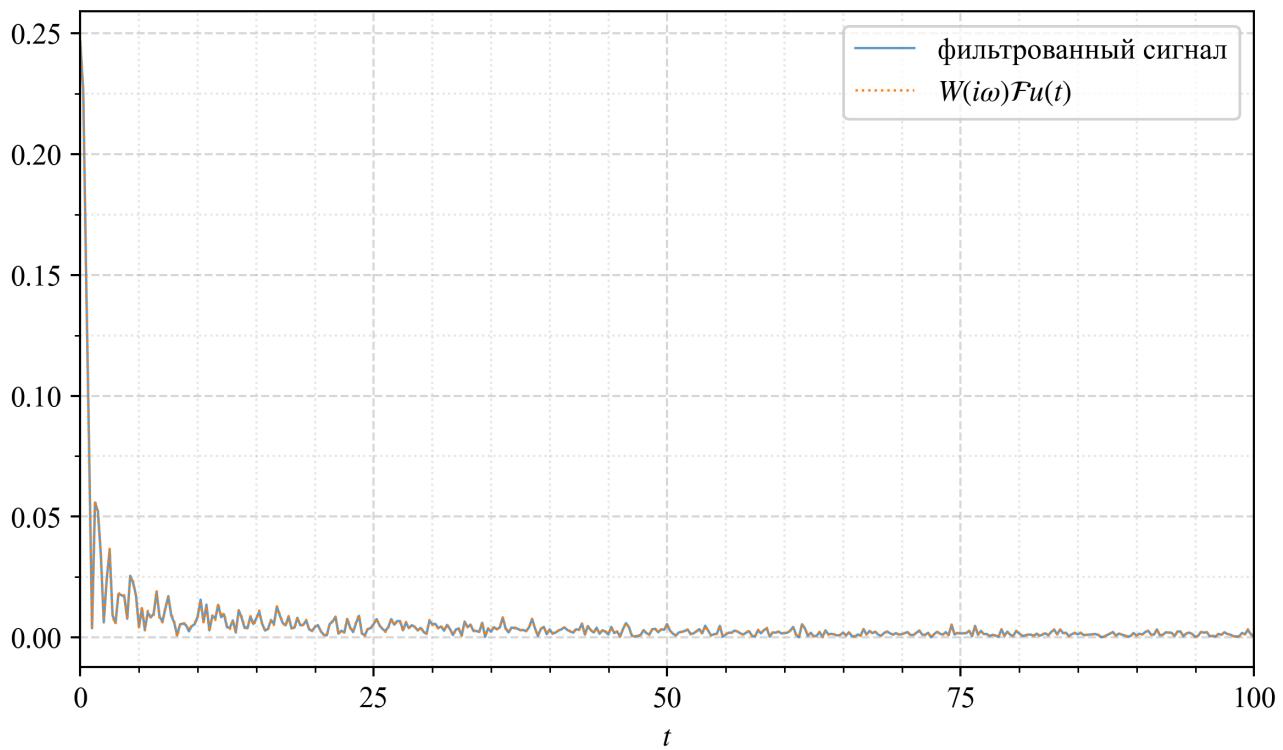


Рис. 11. Сравнение образов сигналов при линейной фильтрации первого порядка

Образ фильтрованного сигнала можно объяснить видом АЧХ фильтра: частоты, далекие от нуля подавляются, а близкие к нулю остаются почти не тронутыми. Фильтр эффективно работает для сигнала $g(t)$, потому что основные частоты квадратной волны находятся в диапазоне частот близком к нулю.

Убедимся в том, что произведение передаточной функции на образ зашумленного сигнала даёт такой же результат, как и образ фильтрованного сигнала фильтром с такой передаточной функцией, построив два графика образов на одном рисунке.



Графики совпали, как и ожидалось.

Фильтр второго порядка

Возьмем подобный зашумленный сигнал $u(t)$, но примем $b = 0$. Такой сигнал имеет только периодический шум, частота которого зависит от d , а точнее, шум будет на частоте $2\pi d$.

Рассмотрим линейный фильтр второго порядка вида

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

Наша задача подобрать такие коэффициенты a_1, a_2, b_1, b_2 , чтобы:

- Фильтр был *устойчивым*: у полинома знаменателя вещественная часть должна быть строго отрицательная.
- АЧХ фильтра должна быть равна 1 на нулевых частотах и при $\omega \rightarrow \infty$.
- Для некоторой частоты ω_0 АЧХ фильтра должна быть равна нулю.

Несложно догадаться, что частота ω_0 должна совпасть с частотой шума, то есть такой фильтр должен удалять определенную частоту или диапазон, не затрагивая остальной спектр.

Фильтр с такими характеристиками называется *Полосно-заграждающий* (*band-stop*), с передаточной функцией вида

$$\frac{p^2 + \omega_0^2}{p^2 + \omega_c p + \omega_0^2}$$

Построим АЧХ Такого фильтра

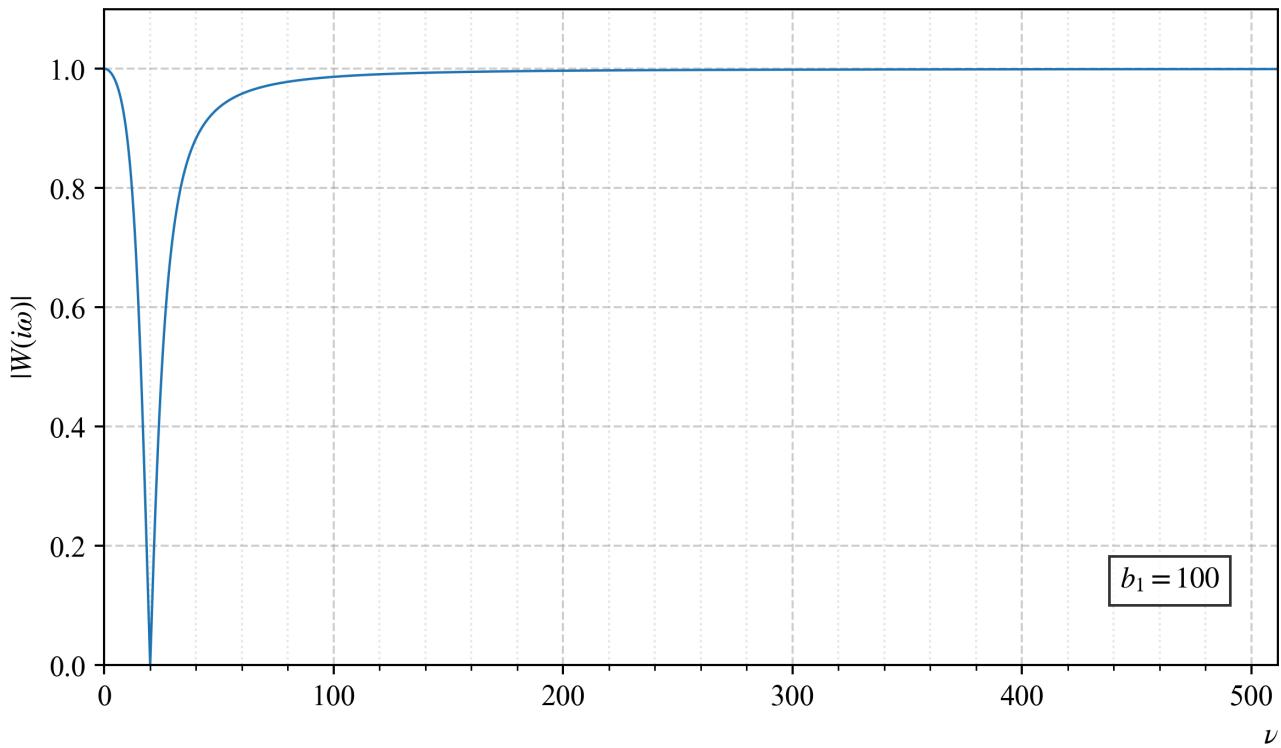
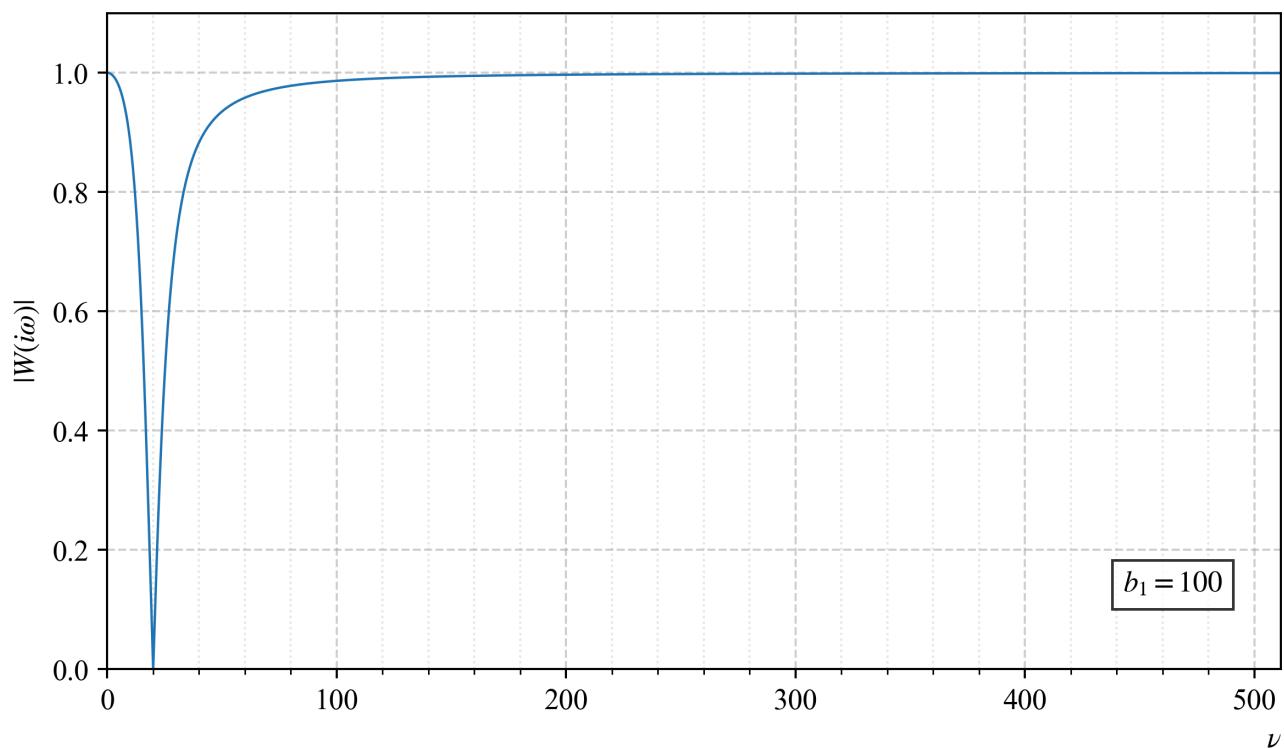
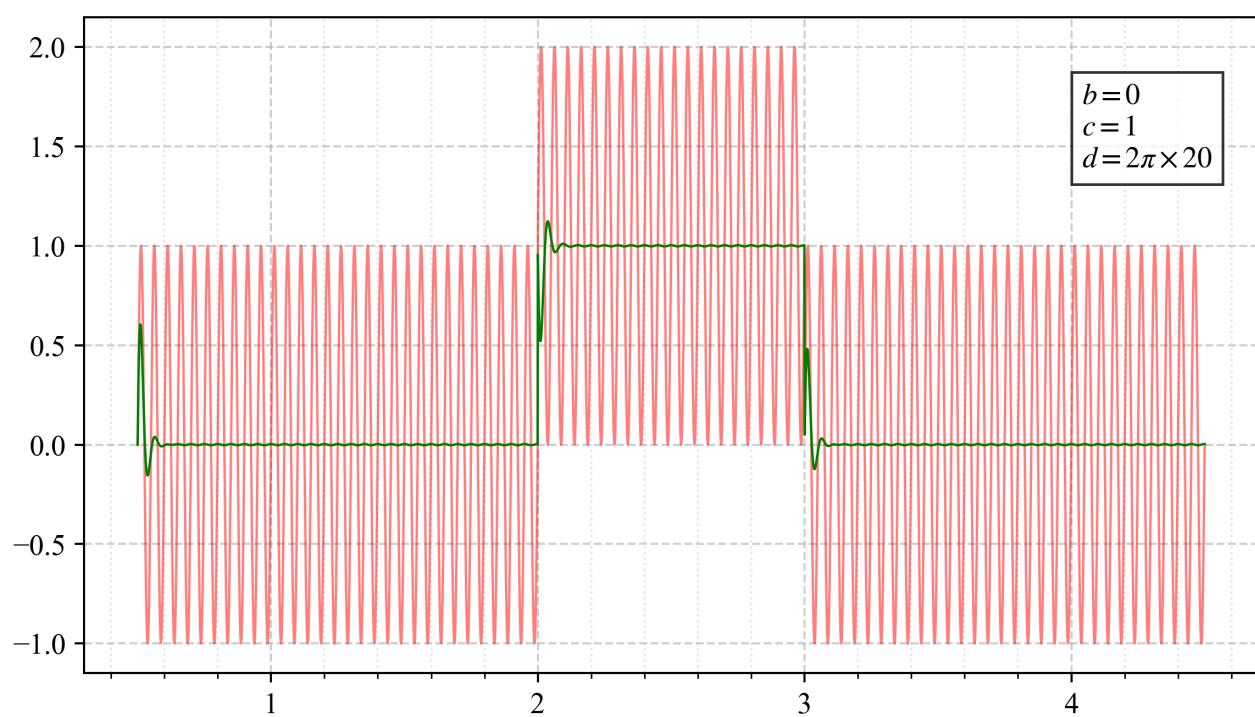


Рис. 13. АЧХ фильтра второго порядка

По рисунку видно, что у этого конкретного фильтра заграждаемая частота — 20 Гц.

Рассмотрим влияние параметра b_1 на АЧХ и результат фильтрации

$$b_1 = 100$$

Рис. 14. АЧХ фильтра второго порядка при $b_1 = 100$ Рис. 15. Результат фильтрации при $b_1 = 100$

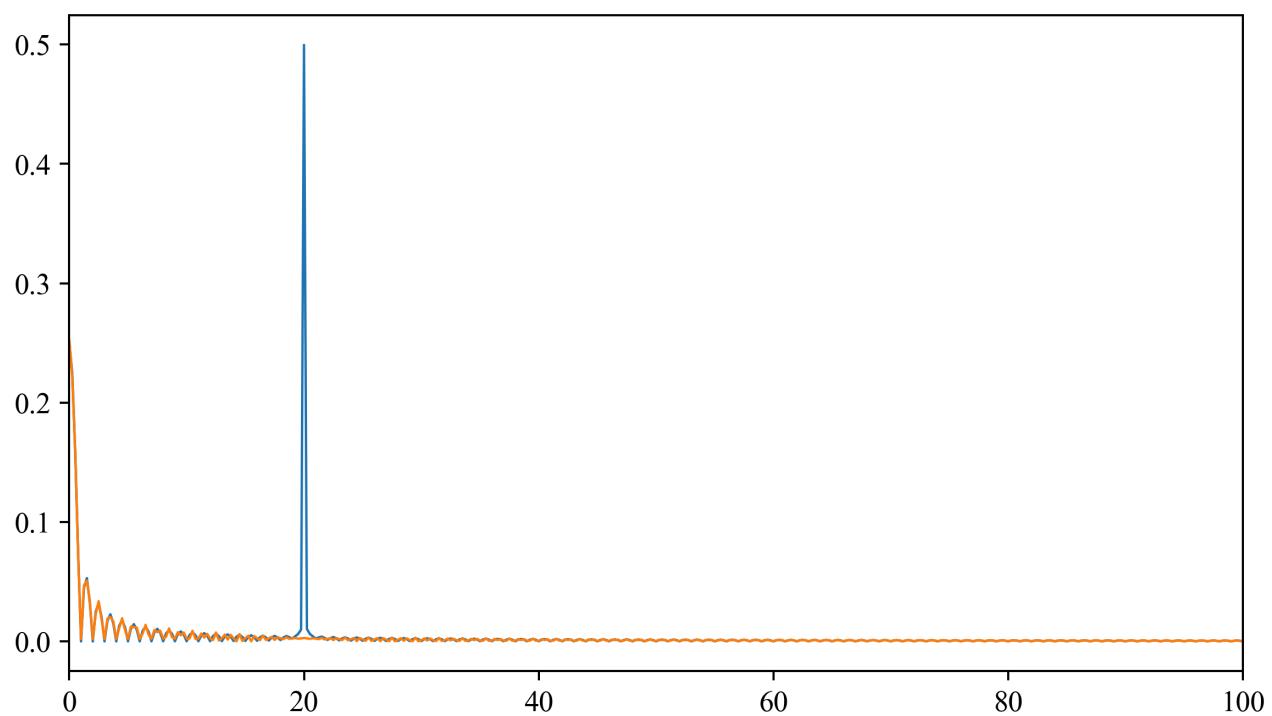
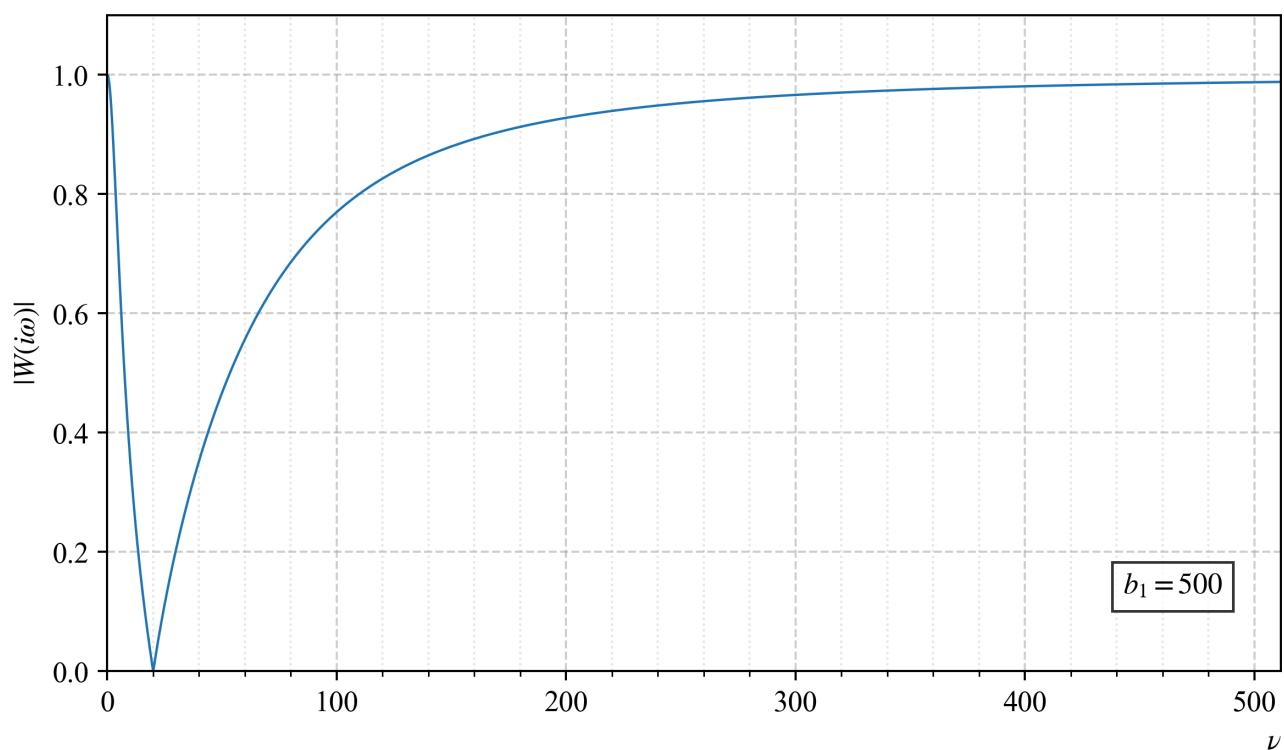
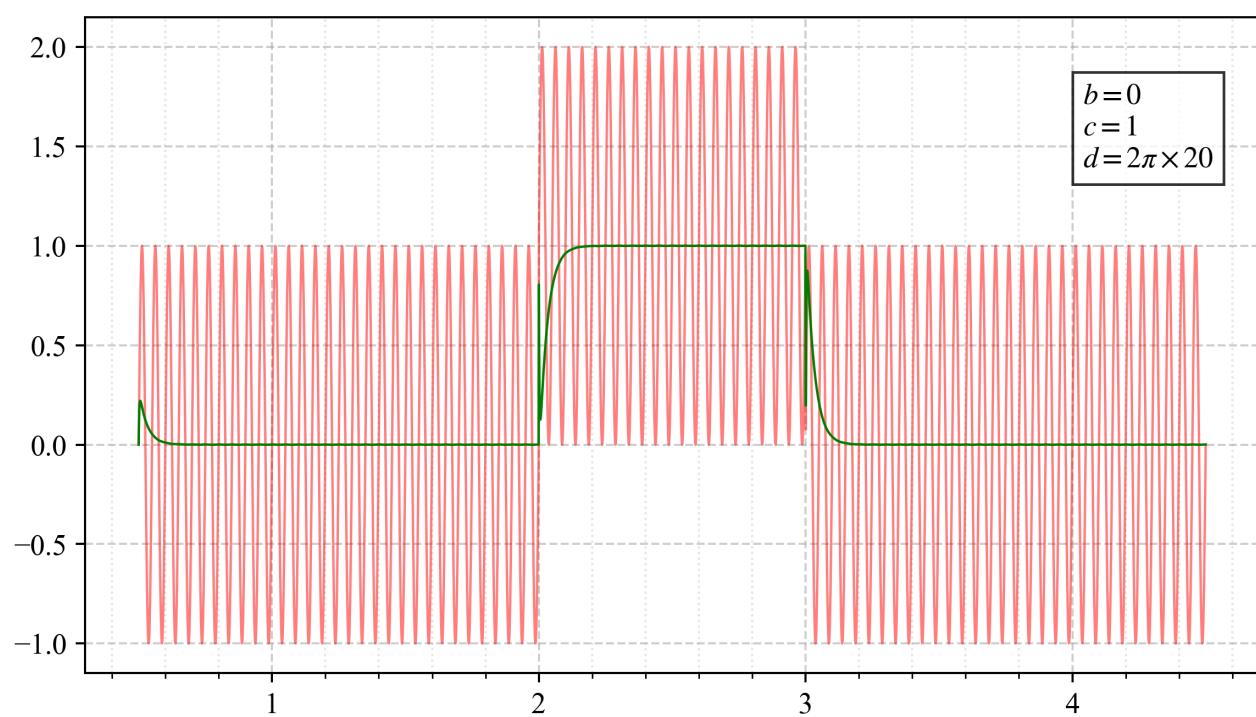


Рис. 16. Сравнение образов зашумленного и фильтрованного сигналов
при $b_1 = 100$

$b_1 = 500$

Рис. 17. АЧХ фильтра второго порядка при $b_1 = 500$ Рис. 18. Результат фильтрации при $b_1 = 500$

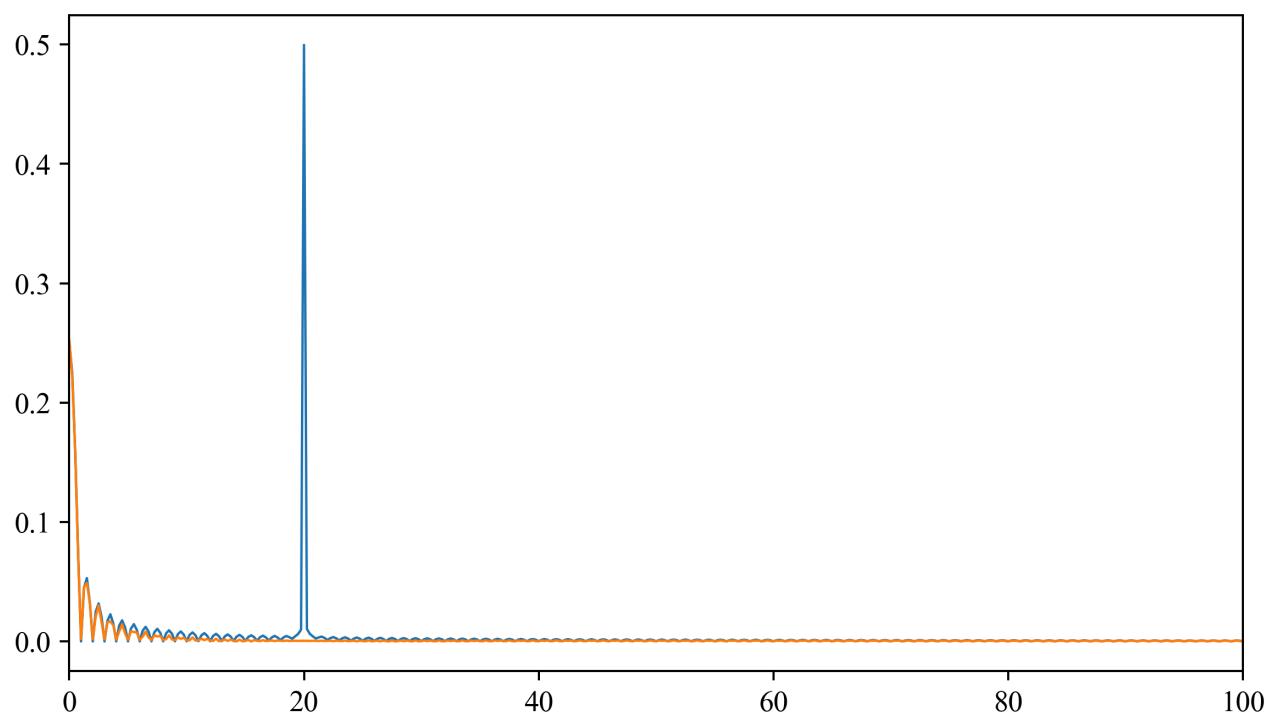
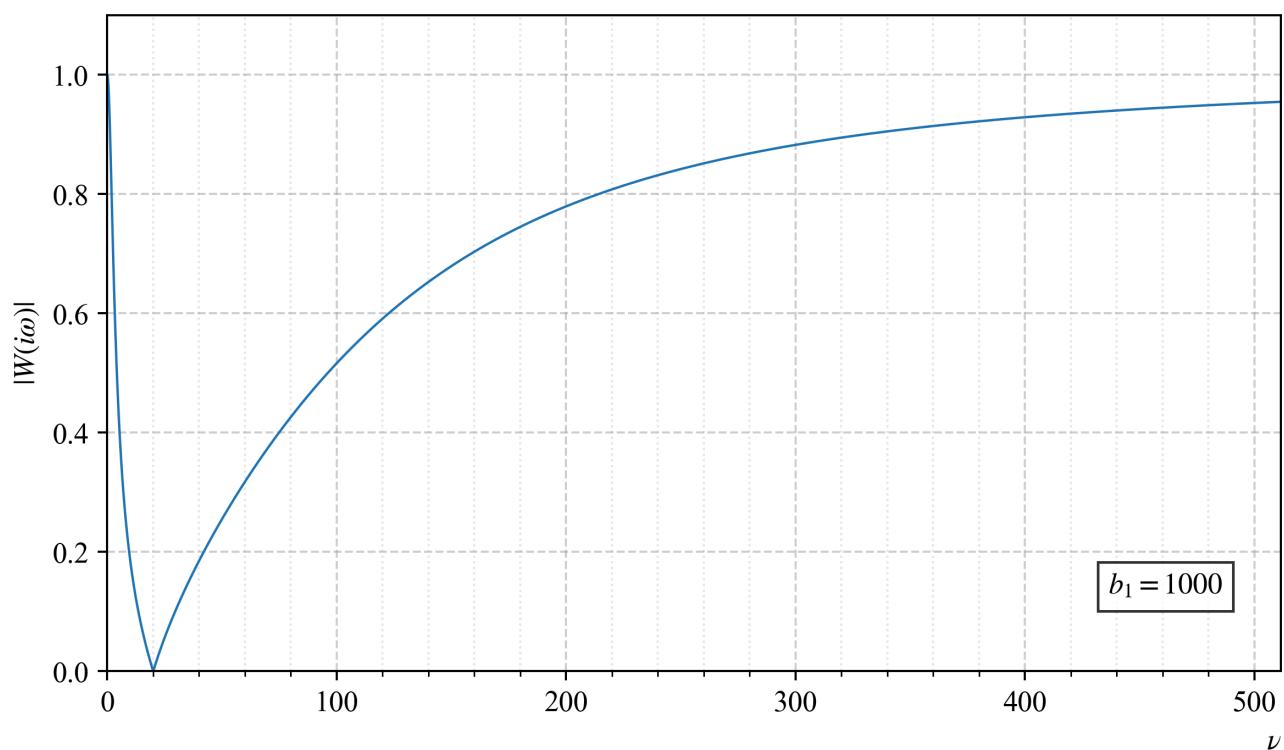
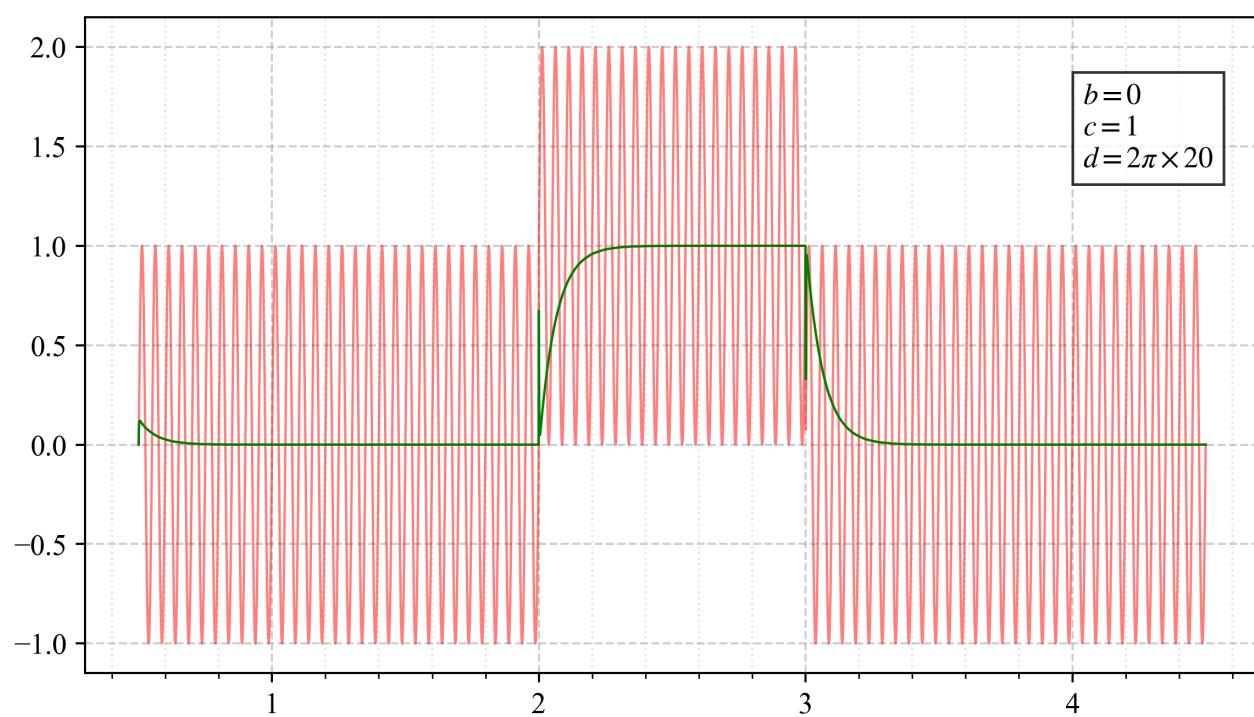


Рис. 19. Сравнение образов зашумленного и фильтрованного сигналов
при $b_1 = 500$

$b_1 = 1000$

Рис. 20. АЧХ фильтра второго порядка при $b_1 = 1000$ Рис. 21. Результат фильтрации при $b_1 = 1000$

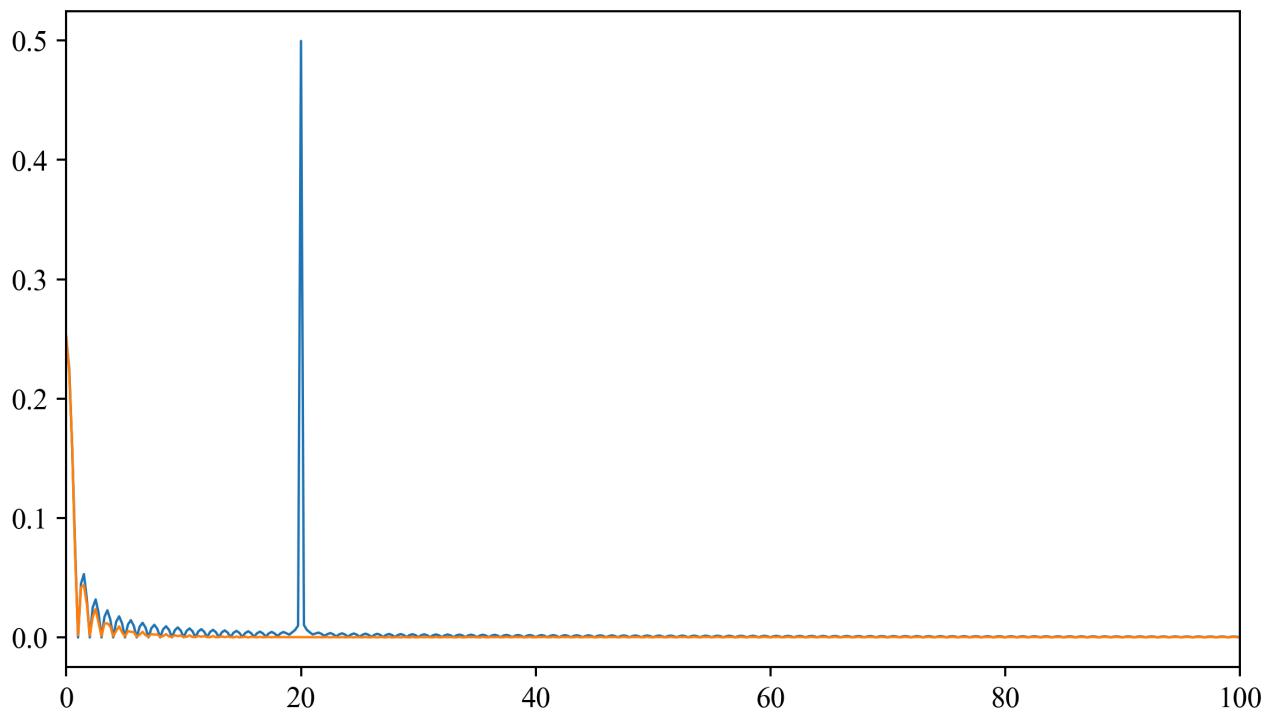


Рис. 22. Сравнение образов зашумленного и фильтрованного сигналов
при $b_1 = 1000$

Сравнив Графики АЧХ рассмотренных конкретных фильтров с разным значением параметра b_1 видим, что он отвечает за ширину заграждаемого окна. Для рассматриваемого зашумленного сигнала наилучшим стало значение $b_1 = 100$.

На всех предыдущих рисунках был рассмотрен случай $d = \omega_0$, теперь рассмотрим как повлияет изменение d на результат фильтрации.

$$d = 2\pi \times 25$$

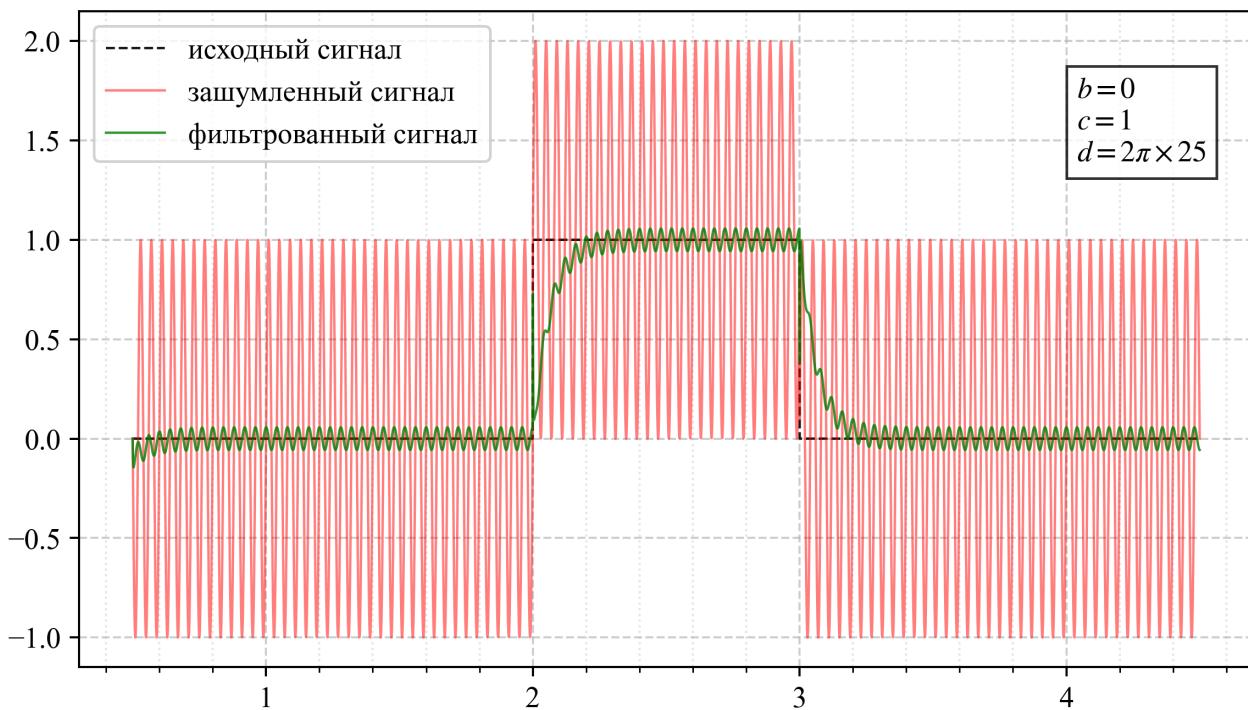


Рис. 23. Сравнение зашумленного и фильтрованного сигнала при $d = 2\pi \times 25$

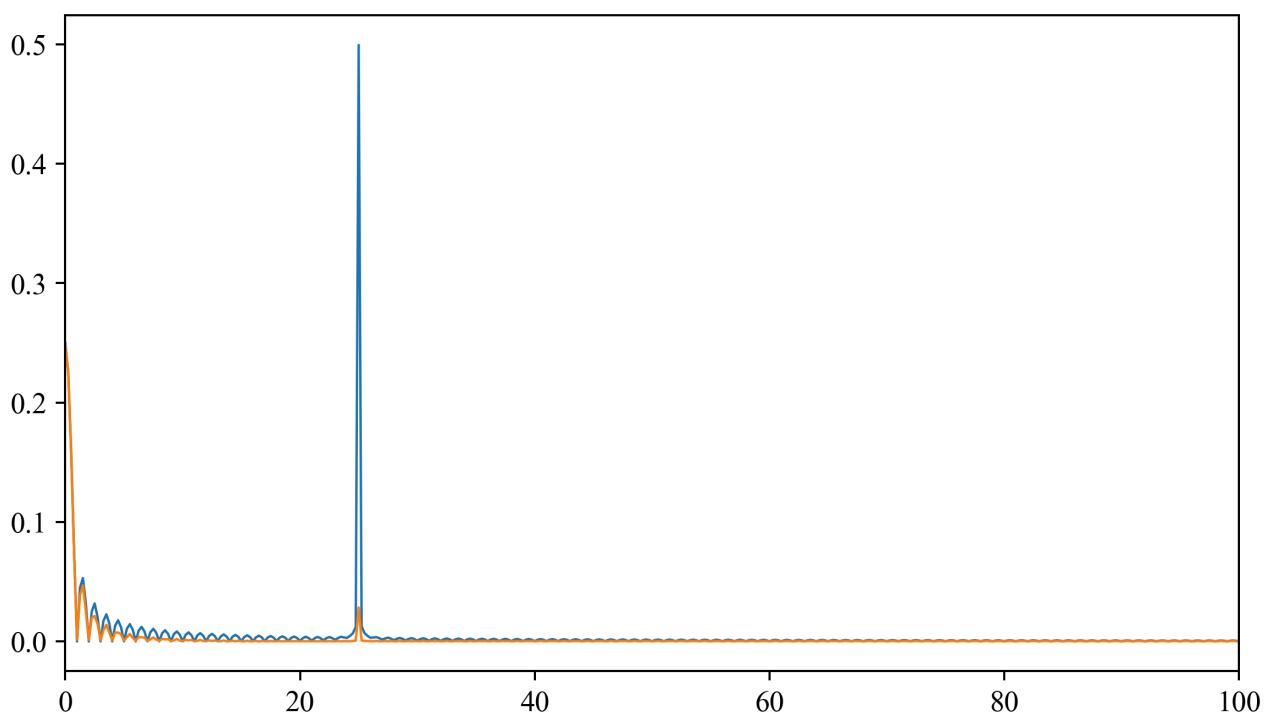


Рис. 24. Сравнение образов зашумленного и фильтрованного сигнала при $d = 2\pi \times 25$

$$d = 2\pi \times 30$$

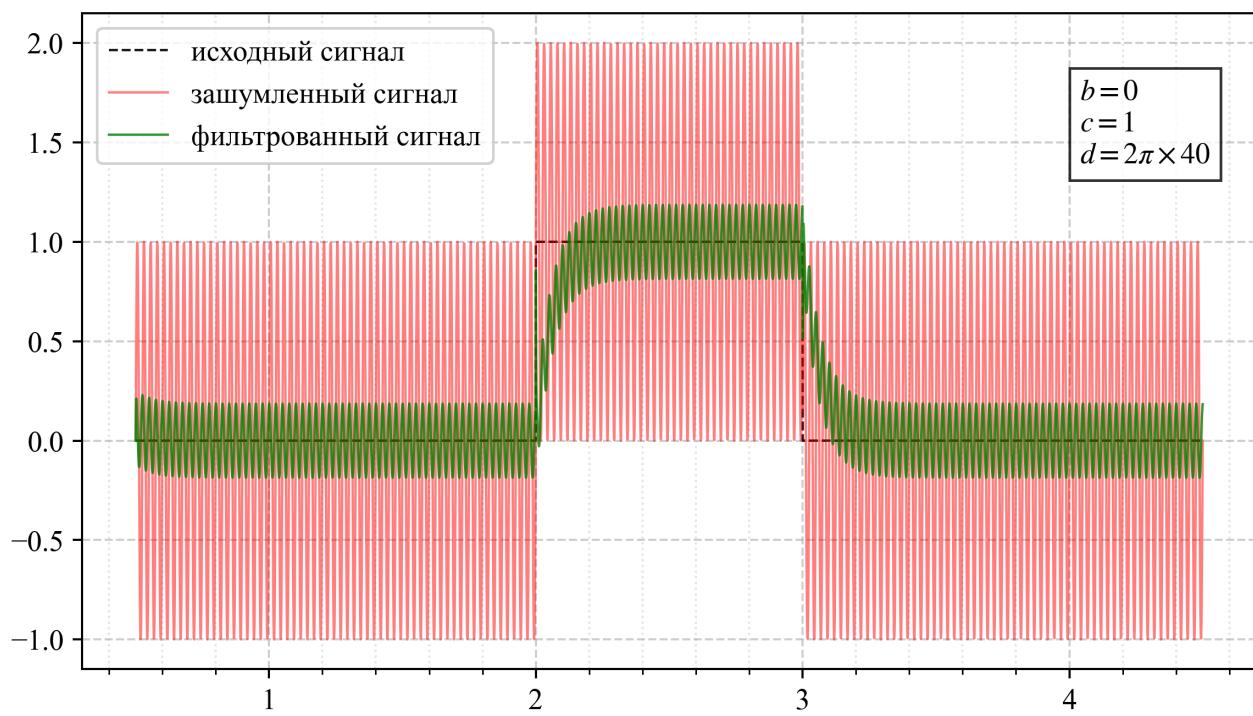


Рис. 25. Сравнение зашумленного и фильтрованного сигнала при $d = 2\pi \times 30$

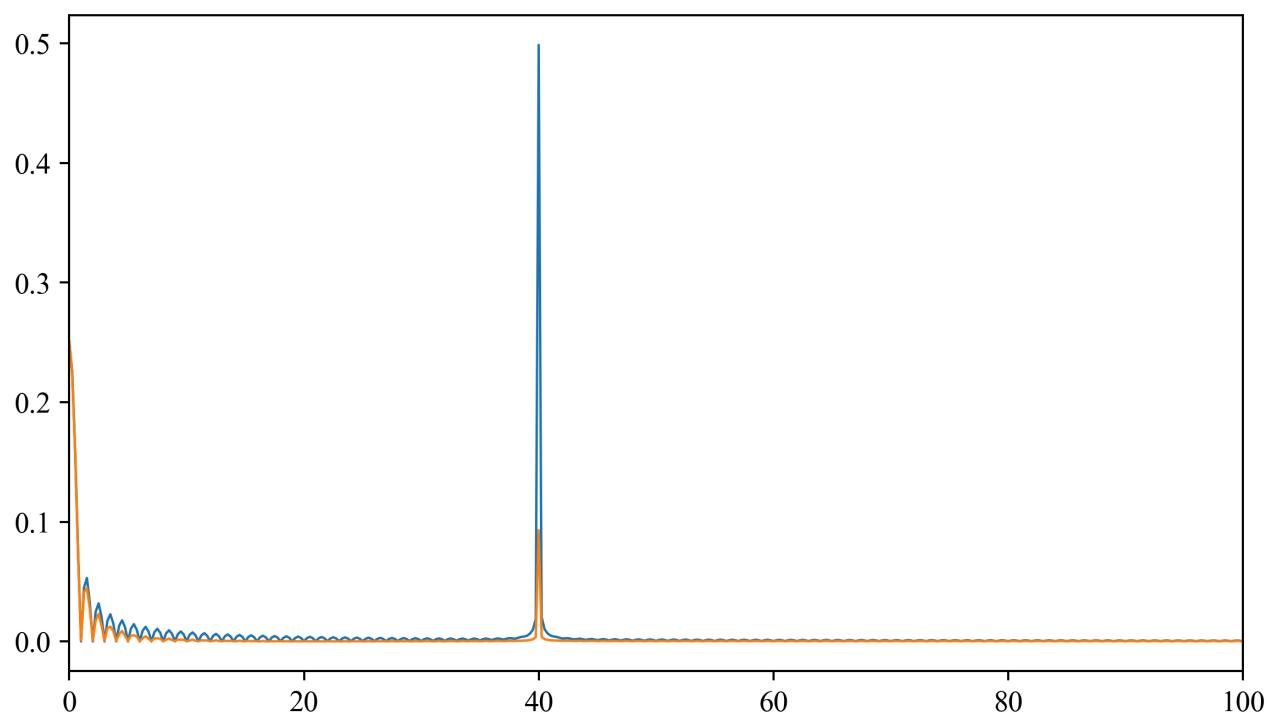
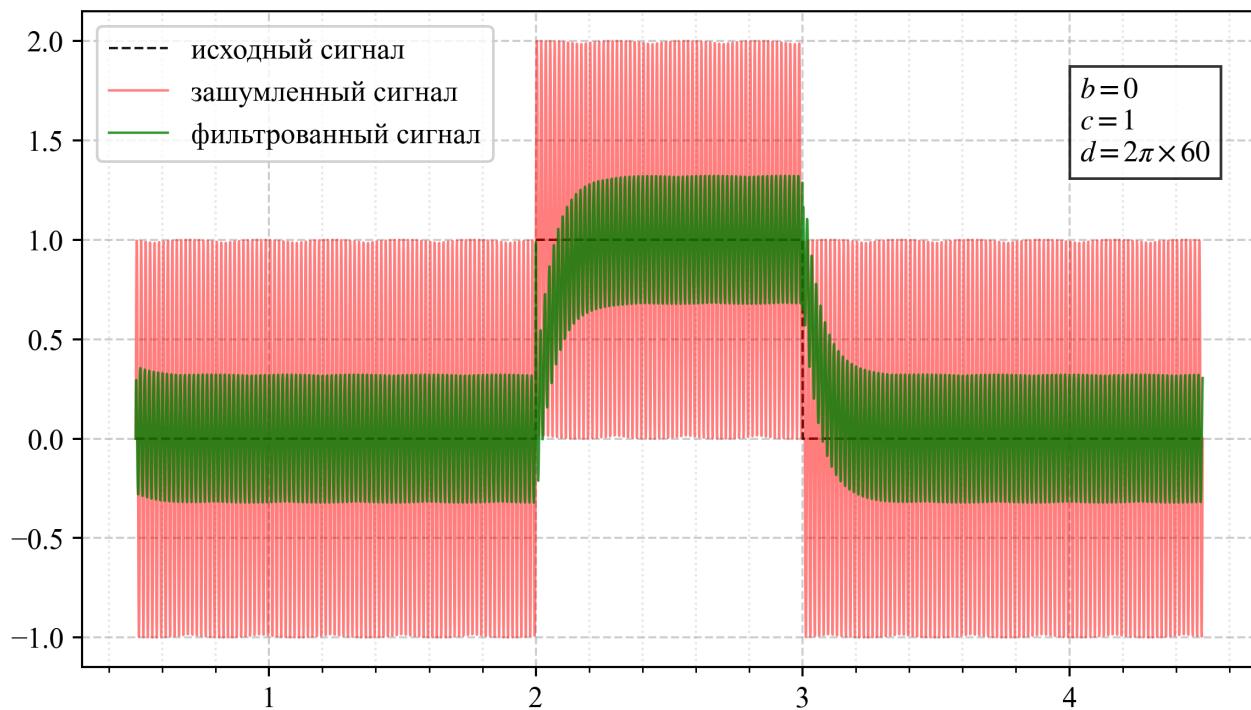
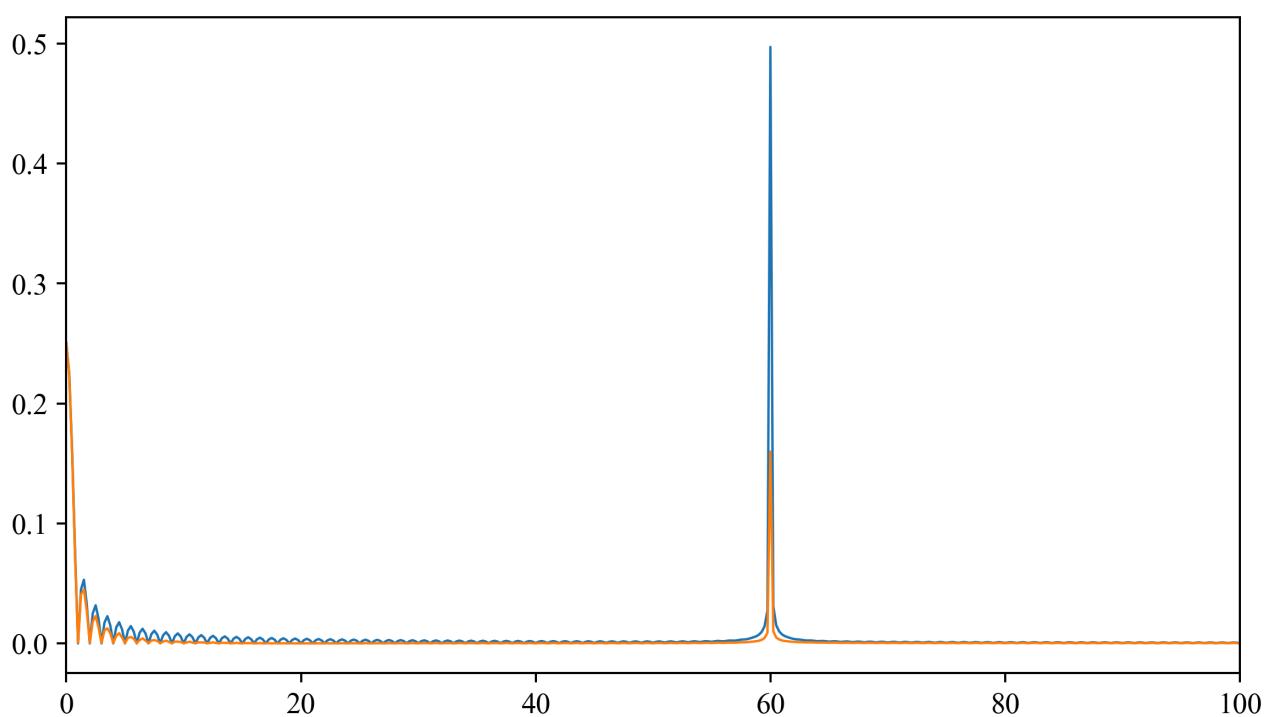


Рис. 26. Сравнение образов зашумленного и фильтрованного сигнала при $d = 2\pi \times 30$

$d = 2\pi \times 60$

Рис. 27. Сравнение зашумленного и фильтрованного сигнала при $d = 2\pi \times 60$ Рис. 28. Сравнение образов зашумленного и фильтрованного сигнала при $d = 2\pi \times 60$

По рисункам видно, что чем больше разница $|d - \omega_0|$, тем хуже удаляется шум.

Сравним, также как в предыдущем задании, результат фильтрации с обратным преобразованием произведения $W_2(i\omega)\hat{u}(t)$

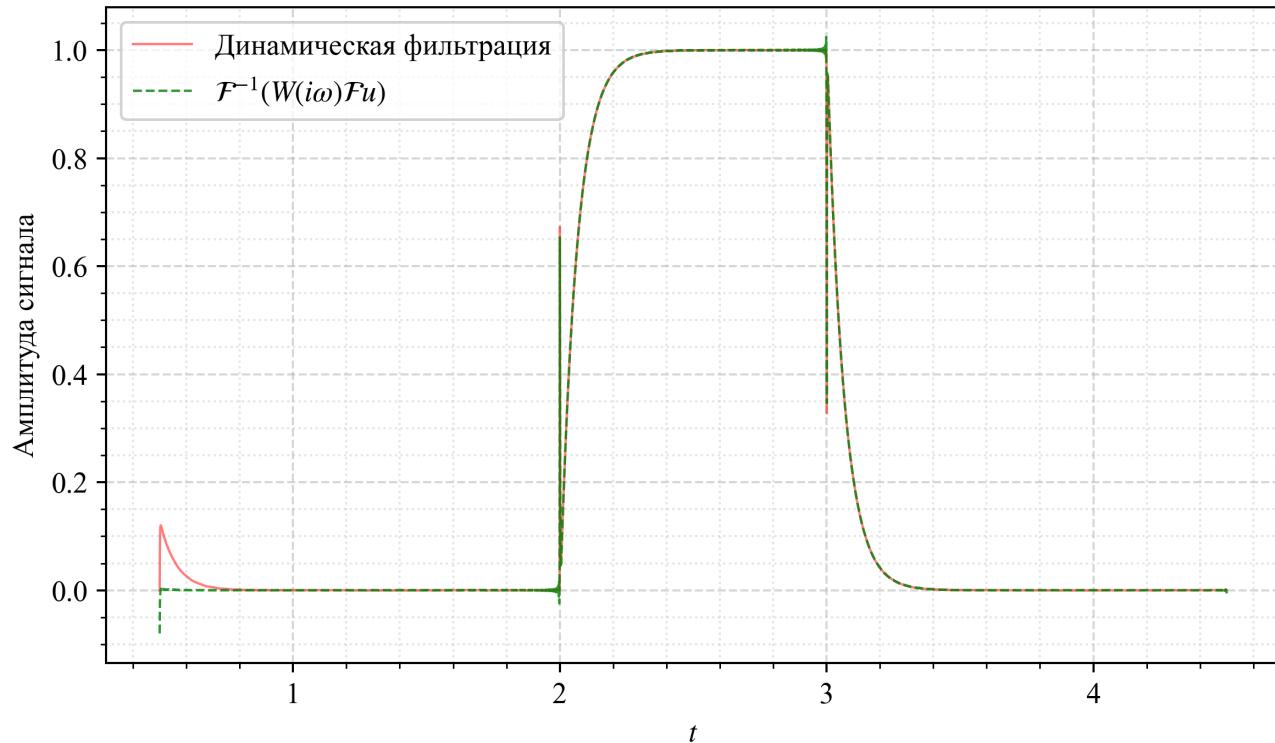


Рис. 29. Сравнение фильтрованного сигнала и обратного преобразования произведения передаточной функции

Графики совпадают почти на всем интервале t , но Динамическая фильтрация в силу ее природы, не знает о начальных условиях сигнала, поэтому странно ведет себя в окрестности $t = 0$.

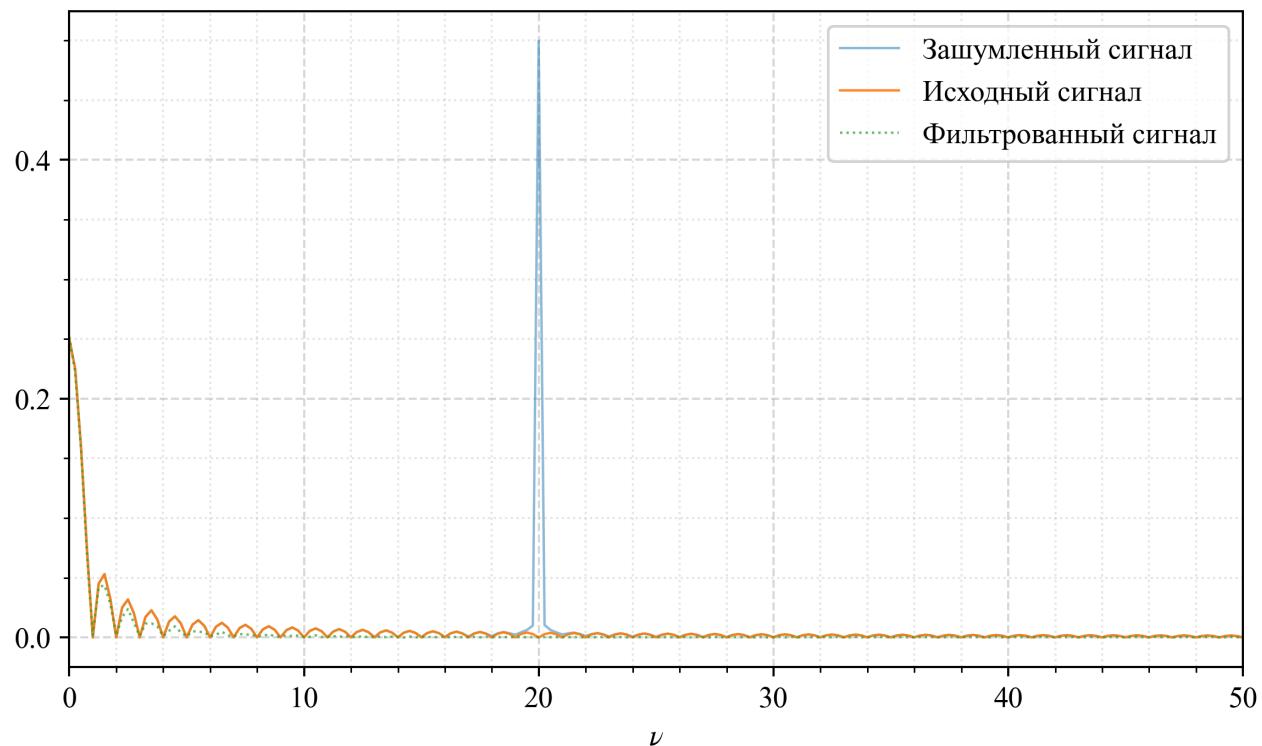


Рис. 30. Сравнение образов чистого, зашумленного и фильтрованного сигналов

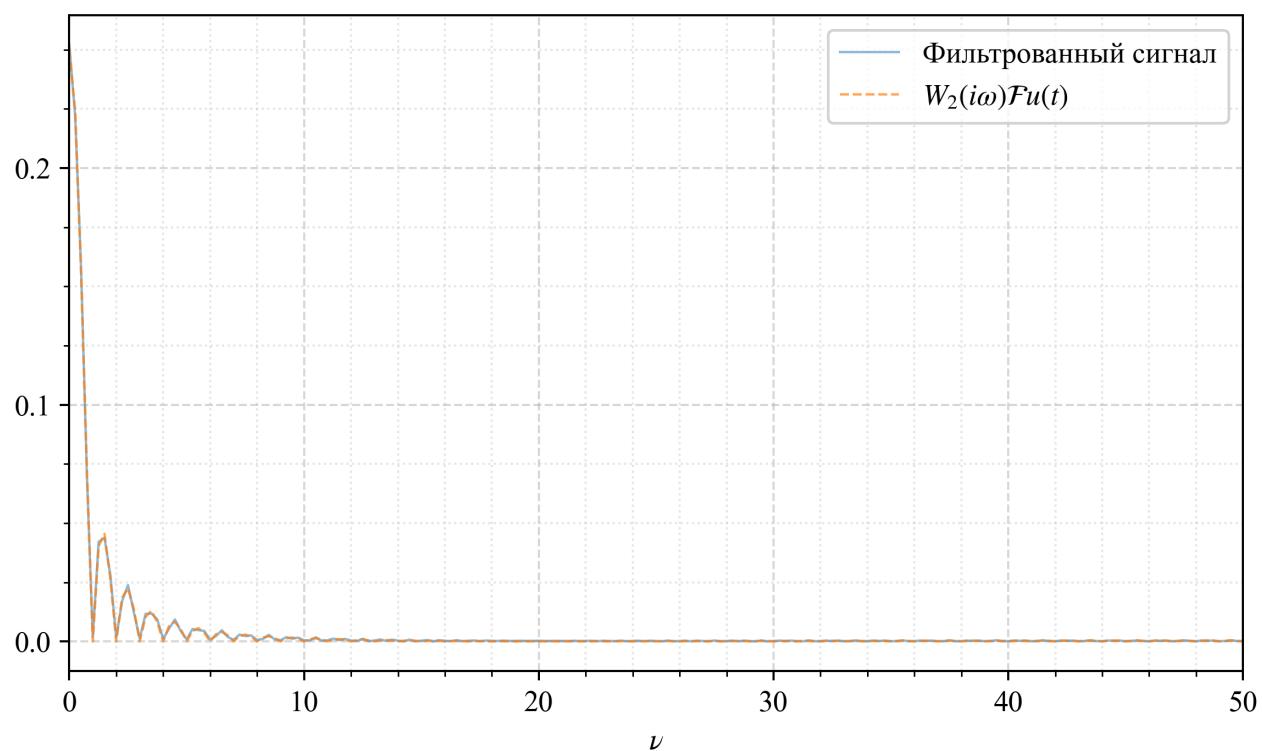


Рис. 31. Сравнение образов фильтрованного сигнала и произведения передаточной функции

Образы сигналов после динамической фильтрации и после произведения передаточной функции на образ зашумленного сигнала снова совпали.

Задание 2. Фильтрация биржевых данных

В этом задании воспользуемся линейным фильтром первого порядка и попробуем сгладить реальные данные. В качестве исходных данных нам было предложено использовать цену акции компании ПАО Сбербанк, но решили использовать стоимость криптовалюты BTC¹ за период с ее появления до 2023-02-21T00:33.

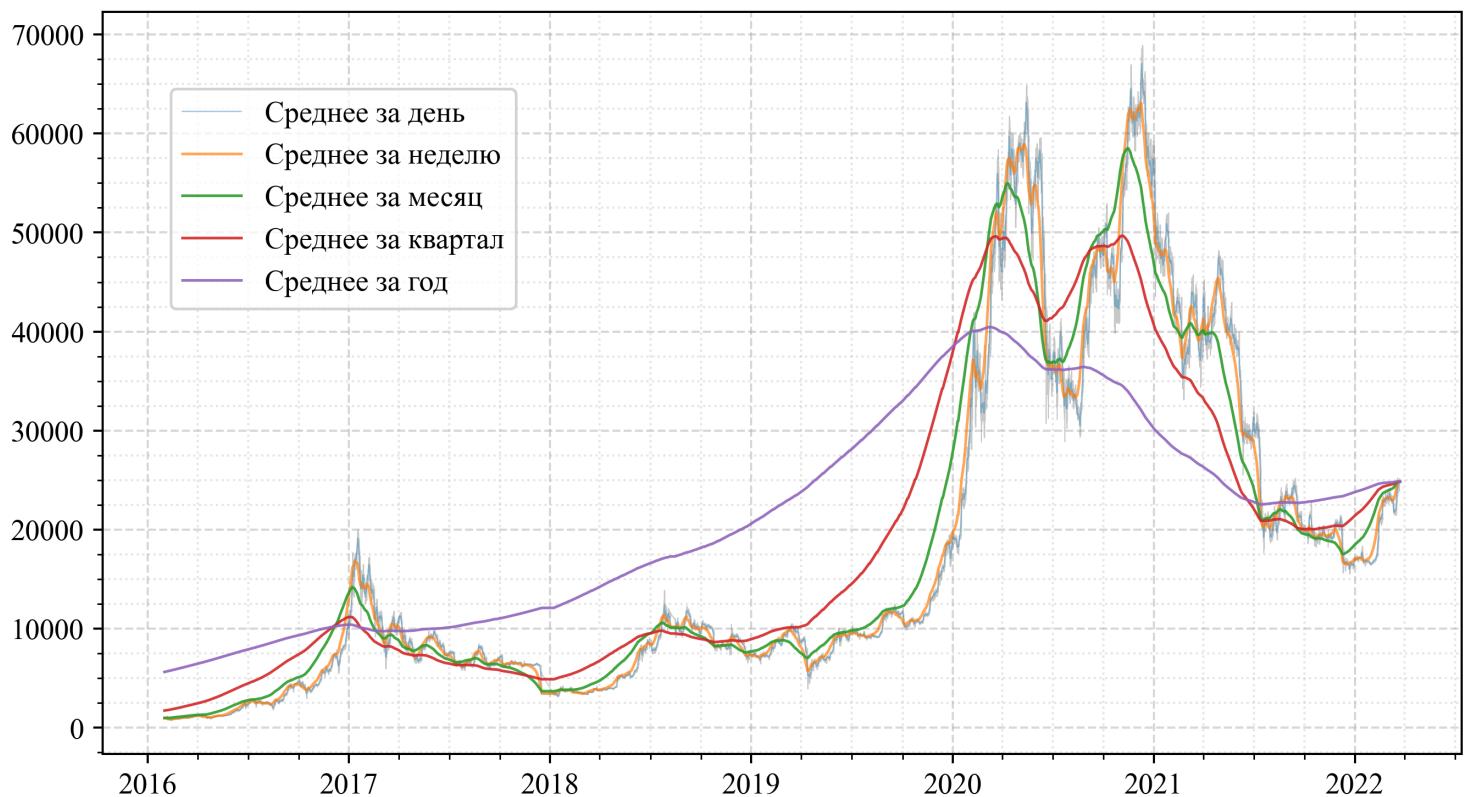


Рис. 32. Стоимость BTC с 2016-09-09 по 2023-06-13

¹<https://www.kaggle.com/datasets/swapr/training-bitcoin-historical-data>

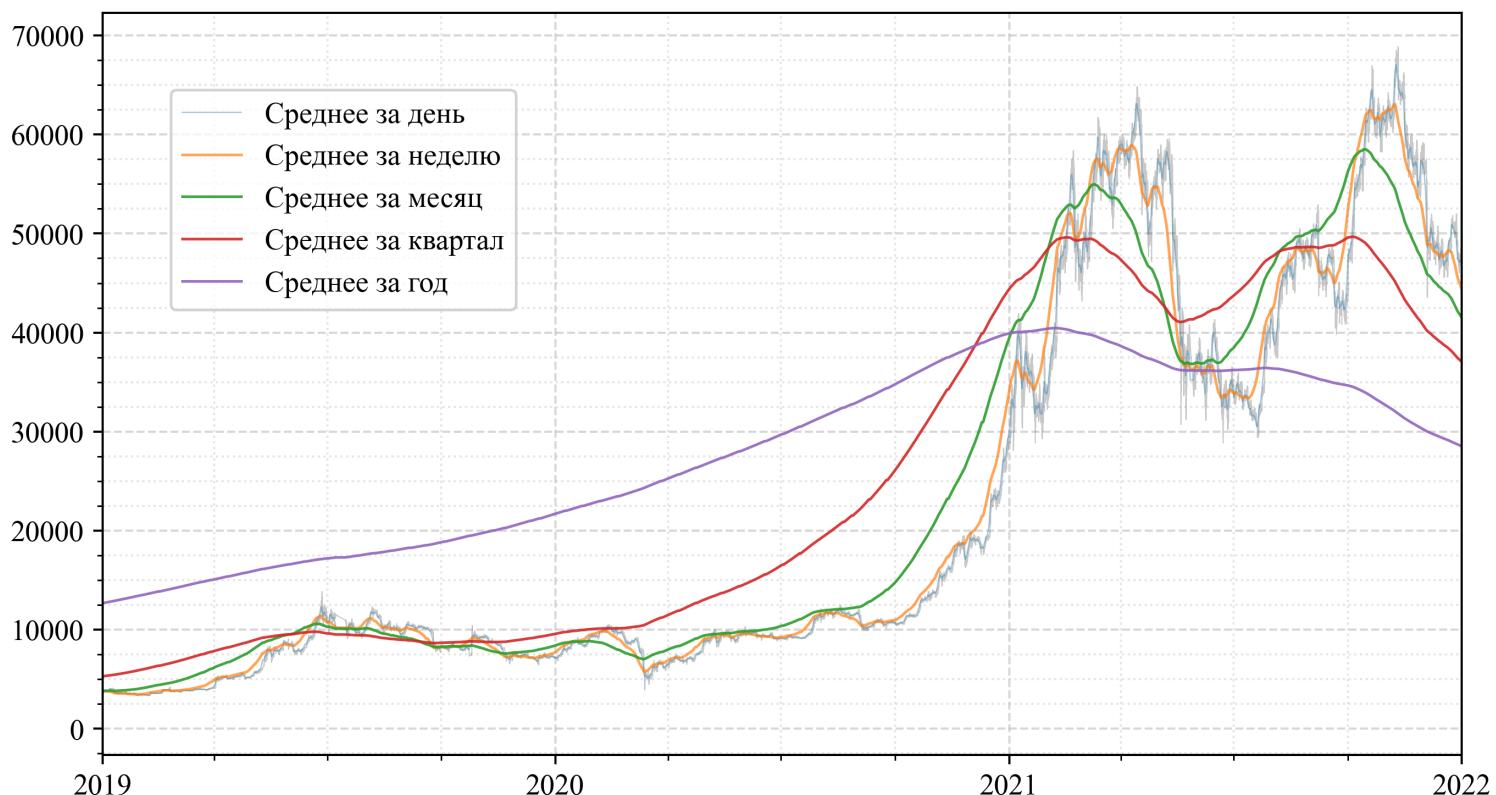


Рис. 33. Стоимость BTC с 2019-01-01 по 2022-01-01



Рис. 34. Стоимость BTC с 2021-01-01 по 2021-07-01

Нетрудно заметить, что чем больше взятый период T , тем плавнее получается график и тем медленнее он отвечает на изменение входных данных. Подобная фильтрация часто встречается в финансовых инструментах и называется *moving average* и позволяет получить больше информации о наборе данных.

Листинги

Ниже представлен исходный код Jupyter блокнота со всеми заданиями лабораторной работы, разбитый на листинги по ячейкам

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

import scipy as sp
def figsize(width, ratio):
    width /= 25.4
    height = width / ratio
    return (width, height)

fig16by9 = figsize(200, 16/9)

plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.family'] = ['Times New Roman', 'Helvetica', 'serif']

plt.rcParams['font.size'] = 11
plt.rcParams['figure.figsize'] = fig16by9
plt.rcParams['figure.dpi'] = 350

plt.rcParams['lines.linewidth'] = .9
```

Листинг 1.

```
def g_(a, t1, t2):
    return lambda t: np.piecewise(t,
        [t > t2, t <= t2, t < t1],
        [0, a, 0]
    )

g = g_(1, 2, 3)

def xi(t):
    return np.random.uniform(-1, 1, size=t.size)

def wave(t, d):
    return np.sin(d*t)

def noisy_signal(t, b, c, d):
    return g(t) + b*xi(t) + c*wave(t, d)
```

Листинг 2.

```
N = 2**10 * 4
t = np.linspace(0.5, 4.5, N)
signal = noisy_signal(t, 1, 0, 1)
clean_signal = g(t)
```

Листинг 3.

```
w_tf = lambda num, den: (num, den)
W1_tf = lambda T: w_tf([1], [T, 1])

# передаточная функция от произвольного аргумента
def W_poly(W):
    return lambda p: sum(p**i*coef for i, coef in
enumerate(reversed(W[0]))) / sum(p**i*coef for i, coef in
enumerate(reversed(W[1])))
```

Листинг 4.

```
# АЧХ
def freq_response(W, t):
    poly = W_poly(W)

    sr = t[1] - t[0]
    N = t.size

    fr = np.fft.fftfreq(N, sr)
    fr = np.fft.fftshift(fr)

    return fr, np.abs(poly(1j*np.pi*2*fr))
```

```
#график АЧХ
def plot_freq_response(W, ax, t):
    fr, response = freq_response(W, t)

    ax.set_xlabel('$\nu$', loc='right')
    ax.set_ylabel('$|W(i\omega)|$')
    ax.plot(fr, response)

    ax.set_xlim(left=0, right=fr[-1])

plot_freq_response(W1_tf(.01), plt.axes(), t)
```

Листинг 5.

```
# фильтрация через произведение перед. функции на образ
def apply_W_nondynamic(t, signal, W, dt=None):
```

```

if dt is None:
    dt = t[1] - t[0]

fr = np.fft.fftfreq(signal.size, dt)
fft = np.fft.fft(signal)

return np.real(np.fft.ifft( W_poly(W)(1j*2*np.pi*fr)*fft ))

```

честная динамическая фильтрация дискретного сигнала

```

def apply_W_dynamic(t, signal, W, dt=None):
    if dt is None:
        dt = t[1] - t[0]

    b, a = sp.signal.bilinear(W[0], W[1], 1/dt)
    y_out = sp.signal.lfilter(b, a, signal)

    return y_out

```

Листинг 6.

```

%mkdir -p ../fig/1/1/T/func
%mkdir -p ../fig/1/1/T/fr

for i, N in enumerate((1e-3, 1e-2, 1e-1)):
    plt.figure()

    filtered_signal = apply_W_dynamic(t, signal, W1_tf(N))

    plt.plot(t, signal, 'r-', alpha=.5, linewidth=.9, label='зашумленный
сигнал')
    plt.plot(t, filtered_signal, 'g-', alpha=.8, linewidth=.9,
label='фильтрованный сигнал')
    plt.plot(t, clean_signal, 'k--', alpha=.6, linewidth=.9,
label='исходный сигнал')
    plt.legend()

    plt.savefig(f"../fig/1/1/T/func/{i}.png")

    plt.figure()

    plot_freq_response(W1_tf(N), plt.gca(), t)

    plt.gca().xaxis.set_minor_locator(plt.MultipleLocator(100))
    plt.gca().yaxis.set_minor_locator(plt.MultipleLocator(0.1))
    plt.grid(which='minor', linestyle=':', alpha=0.4)
    plt.grid(which='major', linestyle='--', alpha=0.6)

```

```
plt.legend([f'$T={N}$'])
plt.savefig(f"../fig/1/1/T/fr/{i}.png")
```

Листинг 7.

```
%mkdir -p ../fig/1/1/a

noise = xi(t)
w = W1_tf(.01)

for i, a in enumerate((1, 2, 3), start=3):

    clean_signal1 = g_(a, 2, 3)(t)
    noisy_signal1 = clean_signal1 + noise

    filtered_signal = apply_W_dynamic(t, noisy_signal1, w)

    plt.figure(num=i)
    plt.plot(t, noisy_signal1, 'r-', alpha=.5, label='зашумленный
сигнал')
    plt.plot(t, filtered_signal, 'g-', label='фильтрованный сигнал')
    plt.plot(t, clean_signal1, 'k--', linewidth=1.5, alpha=.6,
label='исходный сигнал')
    plt.legend()

    plt.savefig(f"../fig/1/1/a/{i}.png")
```

Листинг 8.

```
%mkdir -p ../fig/1/1/comp

plt.figure()

y_out = apply_W_dynamic(t, signal, W1_tf(.01))
out2 = apply_W_nondynamic(t, signal, W1_tf(.01))

plt.plot(t, signal, 'r-', alpha=0.5, label='Зашумленный сигнал $u(t)$')
plt.plot(t, y_out, 'g-', alpha=.4, label='Фильтрованный сигнал')
plt.plot(t, out2, 'b:', alpha=.4, label="$\mathcal{F}^{-1}\{W_1(i \omega)\mathcal{F}u(t)\}$")

plt.xlabel('$t$')
plt.grid(which='major', linestyle=':', alpha=.3)
plt.legend()
```

```
plt.savefig("../fig/1/1/comp/1.png")
```

Листинг 9.

```
plt.figure()

clean_signal = g(t)
noisy_signal1 = g(t) + xi(t)
filtered_signal = apply_W_dynamic(t, noisy_signal1, W1_tf(0.01))

fr = np.fft.fftfreq(t.size, d=t[1]-t[0])
fr = np.fft.fftshift(fr)

clean_fft = np.abs(np.fft.fft(clean_signal))/t.size
clean_fft = np.fft.fftshift(clean_fft)

noisy_fft = np.abs(np.fft.fft(noisy_signal1))/t.size
noisy_fft = np.fft.fftshift(noisy_fft)

filtered_fft = np.abs(np.fft.fft(filtered_signal))/t.size
filtered_fft = np.fft.fftshift(filtered_fft)

plt.xlim(0, 100)

plt.plot(fr, clean_fft, alpha=.9)
plt.plot(fr, noisy_fft, alpha=.5)
plt.plot(fr, filtered_fft, alpha=.7, linestyle=':')

plt.xlabel('$\nu$')
plt.legend(['Исходный сигнал', 'Зашумленный сигнал', 'Фильтрованный сигнал'])

plt.gca().get_xaxis().set_major_locator(plt.MultipleLocator(20))
plt.gca().get_xaxis().set_minor_locator(plt.MultipleLocator(10))
plt.gca().get_yaxis().set_major_locator(plt.MultipleLocator(.05))
plt.gca().get_yaxis().set_minor_locator(plt.MultipleLocator(.025))

plt.grid(which='major', linestyle='--', alpha=.5)
plt.grid(which='minor', linestyle=':', alpha=.3)

plt.savefig("../fig/1/1/comp/2.png")
```

Листинг 10.

```

fr = np.fft.fftfreq(t.size, t[1]-t[0])
fr = np.fft.fftshift(fr)

noisy_fft = np.abs(np.fft.fft(noisy_signal1))
noisy_fft = np.fft.fftshift(noisy_fft)

filtered_fft = np.abs(np.fft.fft(apply_W_dynamic(t, noisy_signal1,
W1_tf(.01))))
filtered_fft = np.fft.fftshift(filtered_fft)

product_fft = W_poly(W1_tf(.01))(1j * 2*np.pi * fr) * noisy_fft
product_fft = np.abs(product_fft)

plt.figure()

plt.xlim(0, 100)
plt.xlabel("$t$")

plt.gca().get_xaxis().set_major_locator(plt.MultipleLocator(25))
plt.gca().get_xaxis().set_minor_locator(plt.MultipleLocator(5))
plt.gca().get_yaxis().set_major_locator(plt.MultipleLocator(.05))
plt.gca().get_yaxis().set_minor_locator(plt.MultipleLocator(.025))

plt.grid(which='major', linestyle='--', alpha=.5)
plt.grid(which='minor', linestyle=':', alpha=.3)

plt.plot(fr, filtered_fft/fr.size, alpha=.7, label='фильтрованный
сигнал')
plt.plot(fr, product_fft/ fr.size, alpha=.99, linestyle=':',
label='$W(i\omega)\mathcal{F}u(t)$')

plt.legend()
plt.savefig("../fig/1/1/comp/3.png")

```

Листинг 11.

```
%mkdir -p ../fig/1/2/
```

```
noisy_signal1 = noisy_signal(t, 0, 1, 2*np.pi * 20)
```

Листинг 12.

```

def check_roots(b, c):
    D = np.complex128(b**2 - 4*c)
    roots = np.real((-b + np.sqrt(D))/2), np.real((-b - np.sqrt(D))/2)
    return all(map(lambda x: x < 0, roots)), roots

```

```

def W2_tf(a1, a2, b1, b2):
    assert check_roots(b1, b2), "Фильтр неустойчивый"

    return W_tf([1, a1, a2], [1, b1, b2])

def W2(a1, a2, b1, b2):
    roots = check_roots(b1, b2)
    assert roots[0], ("Фильтр неустойчивый", roots[1])

    return lambda p: (p**2 + a1*p + a2)/(p**2 + b1*p + b2)

```

Листинг 13.

```

%mkdir -p ./fig/1/2/b1

for i, _b1 in enumerate((10**2, 5*10**2, 10**3)):
    #
    # АЧХ Фильтра
    #
    plt.figure(num=3*i)
    ax = plt.gca()

    ax.xaxis.set_major_locator(plt.MultipleLocator(100))
    ax.xaxis.set_minor_locator(plt.MultipleLocator(20))
    ax.grid(which='major', linestyle='--', alpha=0.6)
    ax.grid(which='minor', linestyle=':', alpha=0.3)

    ax.set_ylim(top=1.1)

    # См.:
    # https://en.wikipedia.org/wiki/Band-stop_filter

    omega0 = 2*np.pi * 20

    a1 = 0
    a2 = omega0**2
    b1 = _b1
    b2 = omega0**2

    _W = W2_tf(a1, a2, b1, b2)

    ax.text(0.95, 0.15, f"$b_1 = {b1}$", transform=ax.transAxes,
            verticalalignment='top', horizontalalignment='right',
            bbox=dict(facecolor='white', alpha=0.8))

    plot_freq_response(_W, ax, t)

```

```

plt.savefig(f"../fig/1/2/b1/{3*i}.png")

# -----
# Сравнение зашумленного и фильтрованного сигнала
# -----
plt.figure(num=3*i+1)
ax = plt.gca()

ax.xaxis.set_major_locator(plt.MultipleLocator(1))
ax.xaxis.set_minor_locator(plt.MultipleLocator(0.2))
ax.grid(which='major', linestyle='--', alpha=0.6)
ax.grid(which='minor', linestyle=':', alpha=0.3)

out_signal = apply_W_dynamic(t, noisy_signal1, _W)

ax.plot(t, noisy_signal1, 'r-', alpha=.5)
ax.plot(t, out_signal, 'g-', alpha=1)

ax.text(0.85, 0.90, f"$b=0$\n$c=1$\n$d=2\pi\times 20$",
transform=ax.transAxes,
       verticalalignment='top', horizontalalignment='left',
       bbox=dict(facecolor='white', alpha=0.8))

plt.savefig(f"../fig/1/2/b1/{3*i+1}.png")

# -----
# Сравнение зашумленного и фильтрованного образа
# -----
plt.figure(num=3*i+2)
ax = plt.gca()

ax.set_xlim(right=100)

fr = np.fft.fftfreq(t.size, t[1] - t[0])[0:t.size//2]
noisy_fft = np.abs(np.fft.fft(noisy_signal1)[0:t.size//2]) / t.size

filtered_fr = np.fft.fftfreq(t.size, t[1] - t[0])[0:t.size//2]
filtered_signal = np.abs((np.fft.fft(out_signal))[0:t.size//2]) / t.size

plt.plot(fr, noisy_fft)
plt.plot(filtered_fr, filtered_signal)

plt.savefig(f"../fig/1/2/b1/{3*i+2}.png")

```

Листинг 14.

```
%mkdir -p ../fig/1/2/d

# См.:
# https://en.wikipedia.org/wiki/Band-stop_filter

omega0 = 2*np.pi * 20

a1 = 0
a2 = omega0**2
b1 = 10**3
b2 = omega0**2

_w = W_tf(a1, a2, b1, b2)

for i, _f0 in enumerate((25, 40, 60)):

    _wave = wave(t, 2*np.pi*_f0)
    _noisy_signal = clean_signal + _wave

    # -----
    # Сравнение зашумленного и фильтрованного сигнала
    # -----
    plt.figure(num=2*i)
    ax = plt.gca()

    ax.xaxis.set_major_locator(plt.MultipleLocator(1))
    ax.xaxis.set_minor_locator(plt.MultipleLocator(0.2))
    ax.grid(which='major', linestyle='--', alpha=0.6)
    ax.grid(which='minor', linestyle=':', alpha=0.3)

    out_signal = apply_W_dynamic(t, _noisy_signal, _w)

    ax.plot(t, clean_signal, 'k--', alpha=.9, label='исходный сигнал')
    ax.plot(t, _noisy_signal, 'r-', alpha=.5, label='зашумленный
сигнал')
    ax.plot(t, out_signal, 'g-', alpha=.8, label='фильтрованный сигнал')

    ax.text(0.85, 0.90, f"$b=0$\n$c=1$\n$d=2\pi\times{_f0}$",
transform=ax.transAxes,
           verticalalignment='top', horizontalalignment='left',
           bbox=dict(facecolor='white', alpha=0.8))

    plt.legend()
    plt.savefig(f"../fig/1/2/d/{2*i}.png")

    # -----
```

```

# Сравнение зашумленного и фильтрованного образа
# -----
plt.figure(num=2*i+1)
ax = plt.gca()

ax.set_xlim(right=100)

fr = np.fft.fftfreq(t.size, t[1] - t[0])[0:t.size//2]
noisy_fft = np.abs(np.fft.fft(_noisy_signal)[0:t.size//2]) / t.size

filtered_fr = np.fft.fftfreq(t.size, t[1] - t[0])[0:t.size//2]
filtered_signal = np.abs((np.fft.fft(out_signal))[0:t.size//2]) / t.size

plt.plot(fr, noisy_fft)
plt.plot(filtered_fr, filtered_signal)

plt.savefig(f"../fig/1/2/d/{2*i+1}.png")

```

Листинг 15.

```

omega0 = 2*np.pi * 20

a1 = 0
a2 = omega0**2
b1 = 10**3
b2 = omega0**2

_w = w_tf(a1, a2, b1, b2)

_noisy_signal = noisy_signal

filtered_signal_nonfair = apply_w_nondynamic(t, _noisy_signal, _w)
filtered_signal_fair = apply_w_dynamic(t, _noisy_signal, _w)

plt.plot(t, filtered_signal_fair, 'r-', alpha=.5, label='Динамическая
фильтрация')
plt.plot(t, filtered_signal_nonfair, 'g--', alpha=.8, label='$\mathcal{F}^{-1}(W(i\omega)\mathcal{F}u)$')

plt.xlabel("$t$")
plt.ylabel("Амплитуда сигнала")

ax = plt.gca()
ax.xaxis.set_major_locator(plt.MultipleLocator(1))
ax.xaxis.set_minor_locator(plt.MultipleLocator(0.2))
ax.yaxis.set_major_locator(plt.MultipleLocator(0.2))

```

```

ax.yaxis.set_minor_locator(plt.MultipleLocator(0.05))
plt.grid(which="major", linestyle="--", alpha=0.5)
plt.grid(which="minor", linestyle=":", alpha=0.3)
plt.legend()

%mkdir -p ../fig/1/2/comp

plt.savefig('../fig/1/2/comp/1.png')

```

Листинг 16.

```

fr = np.fft.fftfreq(t.size, d=t[1]-t[0])[:t.size//2]
_noisy_fft = np.abs(np.fft.fft(_noisy_signal)[:t.size//2])/t.size
_clean_fft = np.abs(np.fft.fft(g(t))[:t.size//2])/t.size
_filtered_fft = np.abs(np.fft.fft(apply_W_dynamic(t, _noisy_signal, _W))[:t.size//2])/t.size

plt.plot(fr, _noisy_fft, alpha=0.5, label='Зашумленный сигнал')
plt.plot(fr, _clean_fft, alpha=0.9, label='Исходный сигнал')
plt.plot(fr, _filtered_fft, alpha=0.7, linestyle=':', label='Фильтрованный сигнал')

plt.xlabel('$\nu$')
plt.legend()

plt.xlim(left=0, right=50)

ax = plt.gca()
ax.xaxis.set_major_locator(plt.MultipleLocator(10))
ax.xaxis.set_minor_locator(plt.MultipleLocator(2))
ax.yaxis.set_major_locator(plt.MultipleLocator(0.2))
ax.yaxis.set_minor_locator(plt.MultipleLocator(0.05))
plt.grid(which="major", linestyle="--", alpha=0.5)
plt.grid(which="minor", linestyle=":", alpha=0.3)

plt.savefig('../fig/1/2/comp/2.png')

```

Листинг 17.

```

fr = np.fft.fftfreq(t.size, d=t[1]-t[0])
_fr = fr[:t.size//2]
_filtered_fft = np.abs(np.fft.fft(apply_W_dynamic(t, _noisy_signal, _W))[:t.size//2]) / t.size
_product_fft = np.abs( W_poly(_W)(1j * np.pi*2 * fr) *
np.fft.fft(_noisy_signal) )[:t.size//2] / t.size

```

```

plt.plot(_fr, _filtered_fft, alpha=0.5, linestyle='-' ,
label='Фильтрованный сигнал')
plt.plot(_fr, _product_fft, alpha=0.7, linestyle='--', label='$W_2(i\omega)\mathcal{F}u(t)$')

plt.xlabel('$\nu$')
plt.legend()
plt.xlim(left=0, right=50)

ax = plt.gca()
ax.xaxis.set_major_locator(plt.MultipleLocator(10))
ax.xaxis.set_minor_locator(plt.MultipleLocator(2))
ax.yaxis.set_major_locator(plt.MultipleLocator(0.1))
ax.yaxis.set_minor_locator(plt.MultipleLocator(0.025))
plt.grid(which="major", linestyle="--", alpha=0.5)
plt.grid(which="minor", linestyle=":", alpha=0.3)

plt.savefig('../fig/1/2/comp/3.png')

```

Листинг 18.

```

import pandas as pd

frame = pd.read_csv('../sup/data.csv')
frame["Date"] = pd.to_datetime(frame["Date"])
frame["Average"] =
(frame['Open']+frame['Close']+frame['High']+frame['Low']) / 4
subset = frame[frame["Date"] > pd.to_datetime("2016-12-31")]

time = dict(
    minute=1,
    hour =60,
    day =60*24,
    week =60*24*7,
    month =60*24*30,
    year =60*24*365
)

```

Листинг 19.

```

# Исходные данные
average_orig = subset["Average"].to_numpy()
_average_orig = average_orig - average_orig[0]

average_day = apply_W_dynamic(None, _average_orig, W1_tf(time['day']),

```

```
dt=1) + average_orig[0]
average_week = apply_W_dynamic(None, _average_orig, W1_tf(time['week']),
dt=1) + average_orig[0]
average_month = apply_W_dynamic(None, _average_orig,
W1_tf(time['month']), dt=1) + average_orig[0]
average_quarter = apply_W_dynamic(None, _average_orig,
W1_tf(time['month']*3), dt=1) + average_orig[0]
average_year = apply_W_dynamic(None, _average_orig, W1_tf(time['year']),
dt=1) + average_orig[0]

plt.plot(subset['Date'], average_orig, 'k-', linewidth=.4, alpha=.2)

plt.plot(subset['Date'], average_day, '-.', linewidth=.5, alpha=.4,
label='Среднее за день')
plt.plot(subset['Date'], average_week, '-.', linewidth=1, alpha=.7,
label='Среднее за неделю')
plt.plot(subset['Date'], average_month, '-.', linewidth=1, alpha=.9,
label='Среднее за месяц')
plt.plot(subset['Date'], average_quarter, '-.', linewidth=1, alpha=.9,
label='Среднее за квартал')
plt.plot(subset['Date'], average_year, '-.', linewidth=1, alpha=.9,
label='Среднее за год')

plt.legend(loc=(.05, .6))

ax = plt.gca()
ax.xaxis.set_major_locator(mpl.dates.MonthLocator(interval=12)) # каждые 3 месяца - квартал
ax.xaxis.set_minor_locator(mpl.dates.MonthLocator(interval=3))
# минорные - каждый месяц
ax.xaxis.set_major_formatter(mpl.dates.DateFormatter('%Y'))

ax.yaxis.set_major_locator(mpl.ticker.MultipleLocator(10000))
ax.yaxis.set_minor_locator(mpl.ticker.MultipleLocator(2500))

ax.grid(which='major', linestyle='--', alpha=0.5)
ax.grid(which='minor', linestyle=':', alpha=0.3)

plt.margins(.05)
plt.tight_layout()

%mkdir -p ../fig/2
plt.savefig('../fig/2/1.png')
date_range = mpl.dates.num2date(ax.get_xlim())
print("Figure ../fig/2/1.png date range:", date_range[0].strftime('%Y-
```

```
%m-%d'), "to", date_range[1].strftime('%Y-%m-%d'))
plt.savefig('../fig/2/1.png')

start_date = pd.to_datetime("2019-01-01")
end_date = pd.to_datetime("2022-01-01")

ax.set_xlim(start_date, end_date)
plt.savefig('../fig/2/2.png')
date_range = mpl.dates.num2date(ax.get_xlim())
print("Figure ../fig/2/1.png date range:", date_range[0].strftime('%Y-
%m-%d'), "to", date_range[1].strftime('%Y-%m-%d'))

start_date = pd.to_datetime("2021-01-01")
end_date = pd.to_datetime("2021-07-01")

ax.set_xlim(start_date, end_date)
plt.savefig('../fig/2/3.png')
date_range = mpl.dates.num2date(ax.get_xlim())
print("Figure ../fig/2/1.png date range:", date_range[0].strftime('%Y-
%m-%d'), "to", date_range[1].strftime('%Y-%m-%d'))
```

Листинг 20.