МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И
КОМПЬЮТЕРНОЙ ТЕХНИКИ

ДОМАШНЯЯ РАБОТА №1

по дисциплине

«ТЕОРИЯ ФУНКЦИЙ КОМПЛЕКСНОГО ПЕРЕМЕННОГО»

Выполнил:
Рязанов Никита Сергеевич,
студент группы P3207,
поток 23.1

Проверил:
Краснов Александр Юрьевич

г. Санкт-Петербург, 2025

Содержание

Задание 1. Линейные фильтры

Задание 1.1 Фильтр первого порядка

Исходный код для построения графиков:

```
# Возьмем следующие числа для задания функции g(t)
a = 3
t1 = 1
t2 = 33

# 1. Линейные фильтры
T_total = 100 # большой интервал времени
dt = 0.05 # маленький шаг дискретизации
t = np.arange(-T_total/2, T_total/2, dt) # набор временных шагов

g = np.zeros_like(t)
g[(t >= t1) & (t <= t2)] = a # значения функции g(t)
xi = np.random.uniform(-1, 1, size=t.shape) # дискретные значения
шума
u = g + b*xi + c*np.sin(d*t) # зашумленная версия g(t)

# 1.1 Фильтр первого порядка
c = 0
T_const = 3.0
# Определение фильтра 1-го порядка W(p) = 1 / (T*p + 1)
num = [1]
den = [T_const, 1]
filter1 = signal.TransferFunction(num, den)

t_sim = t - t[0] # сдвигаем в [0, 100], чтобы не вылетала ошибка
(ограничение библиотеки)
# Пропускание сигнала u через фильтр
tout, u_filtered, _ = signal.lsim(filter1, u, t_sim)

freqs = fftfreq(len(t), dt)  # массив циклических частот (Гц)
omegas = 2 * np.pi * freqs  # массив угловых частот (рад/с)
```

```
u_hat = fft(u)  # образ Фурье зашумленного сигнала
g_hat = fft(g)  # образ Фурье исходного сигнала

W_iw = 1 / (T_const * 1j * omegas + 1)


ach = np.abs(W_iw) # АЧХ

# Теоретический образ Фурье
u_spec_hat = u_hat * W_iw

# Обратное преобразование Фурье
u_spec = np.real(ifft(u_spec_hat))
```

Рассмотрим различные значения T при фиксированном a и значения a при фиксированном T.

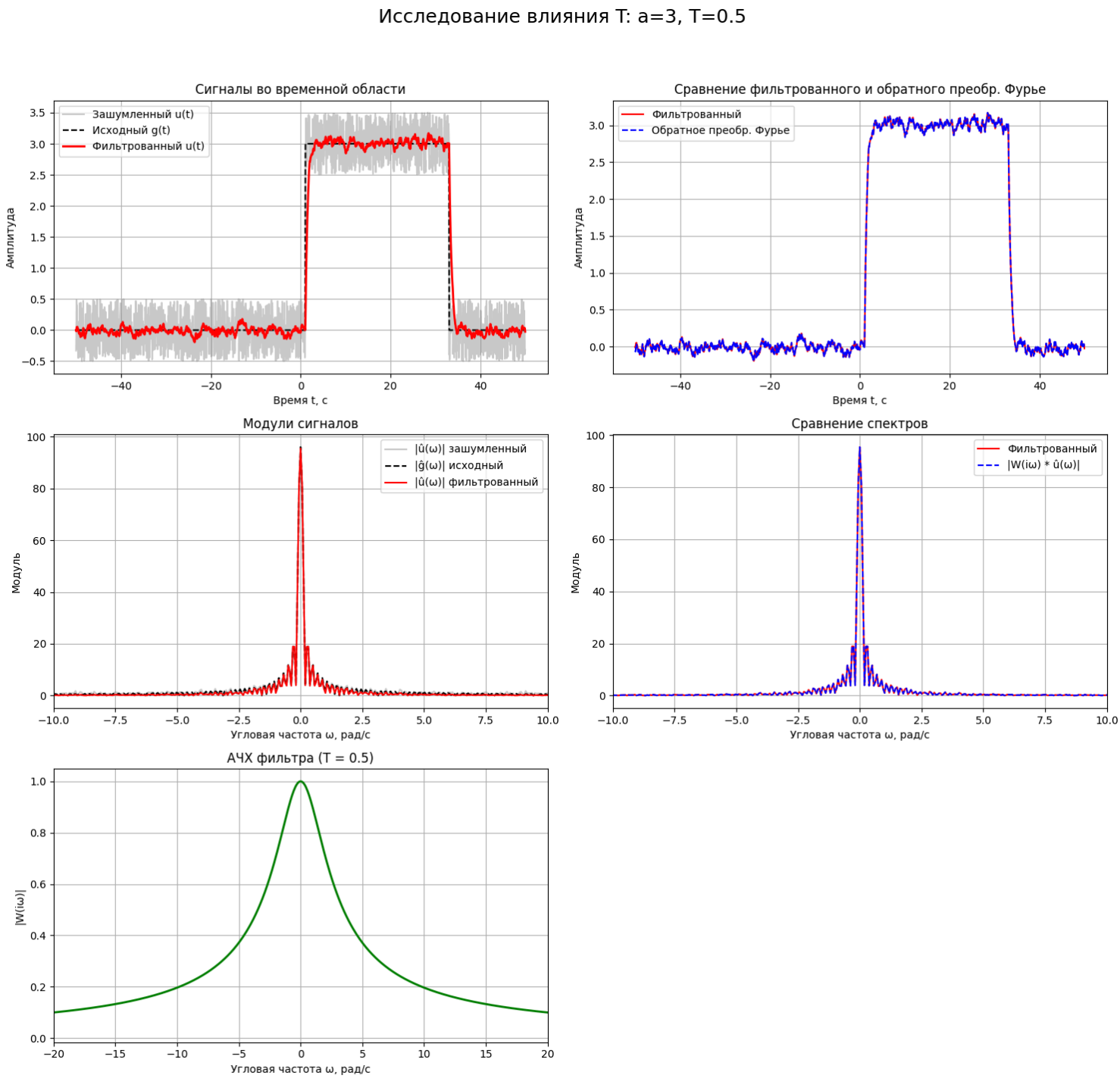

Рис. 1. Исследования влияния T: $a = 3, T = 0.5$

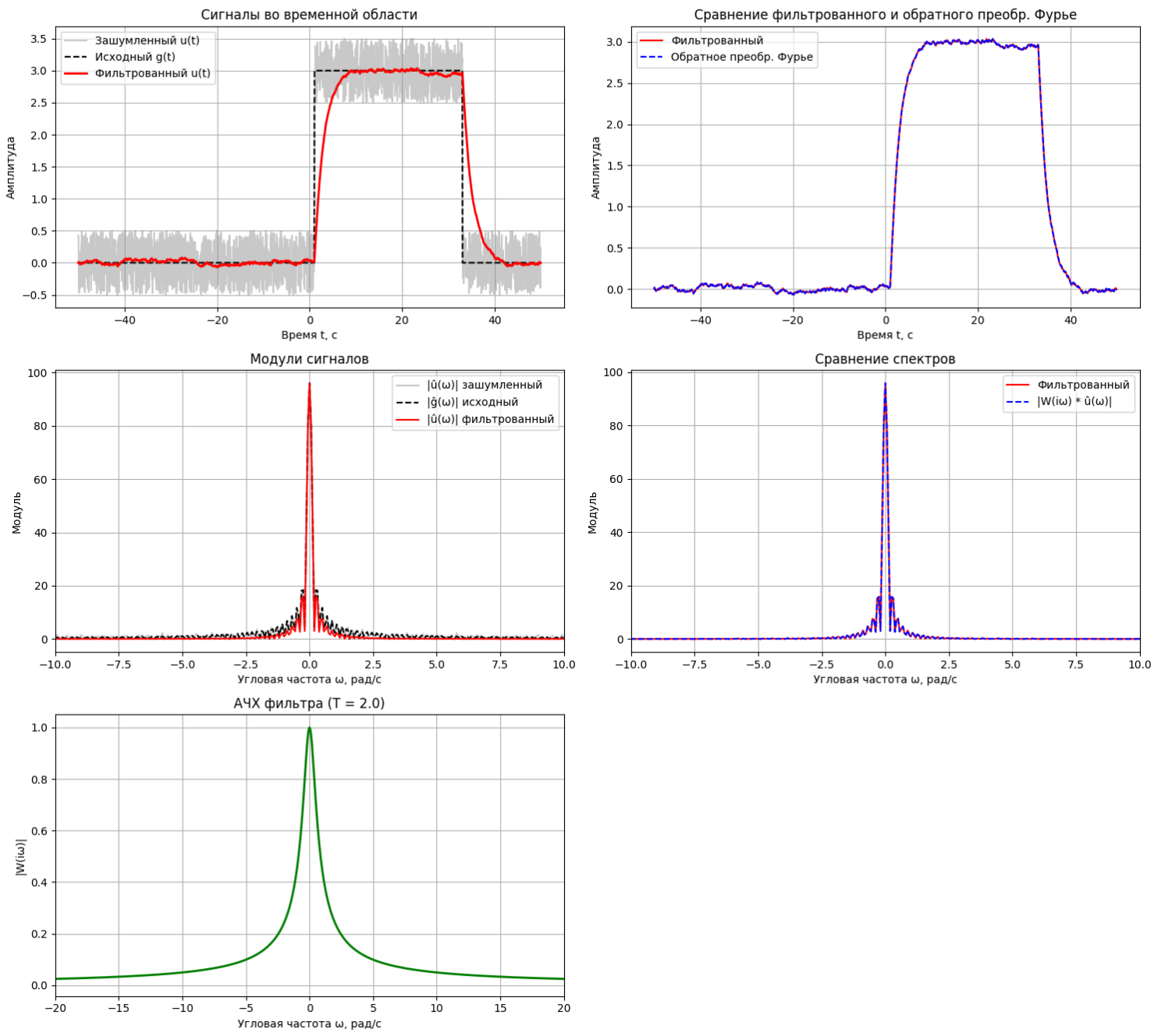

Рис. 2. Исследования влияния T: $a = 3, T = 2$

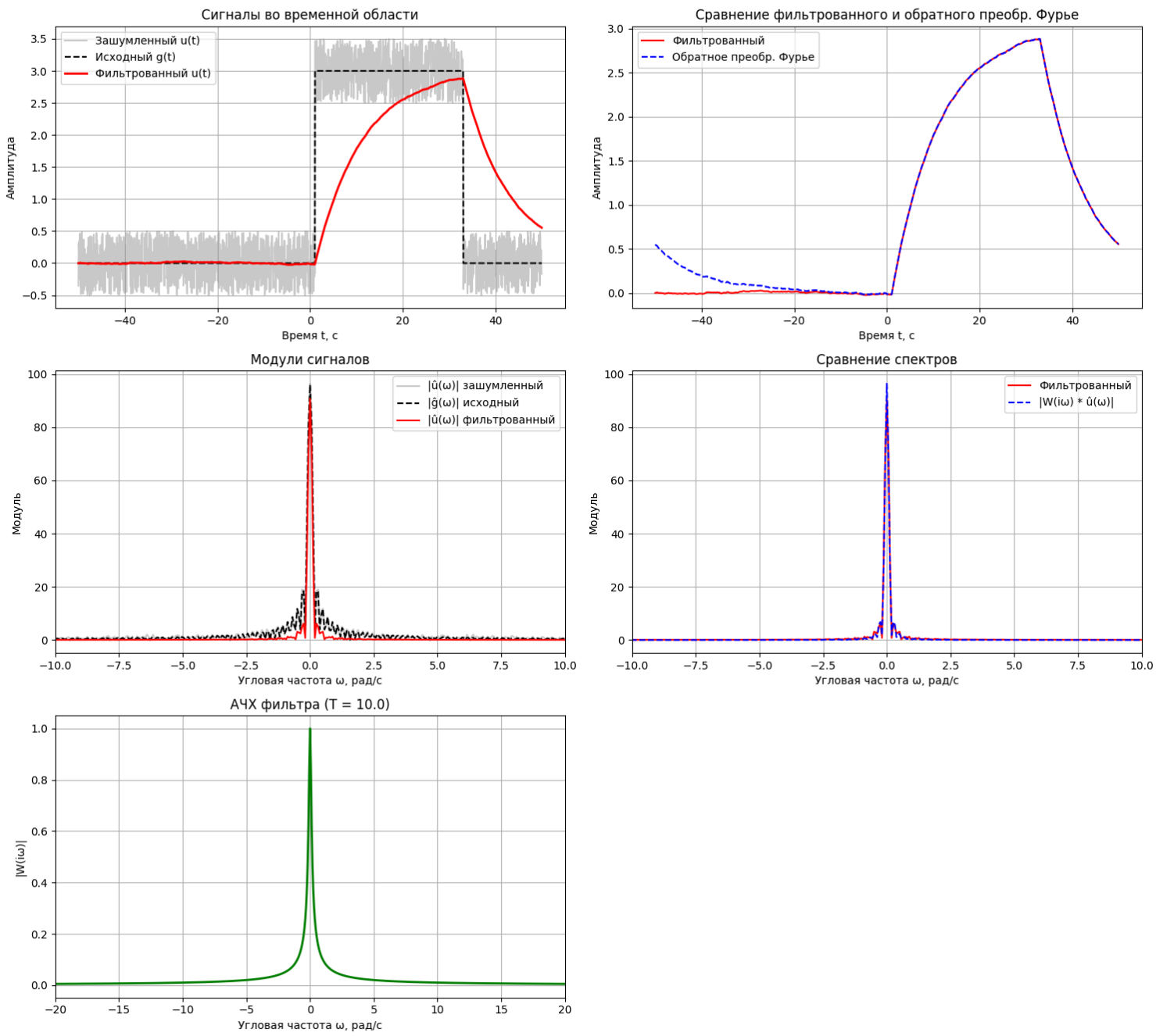

Рис. 3. Исследования влияния T: $a = 3, T = 10$

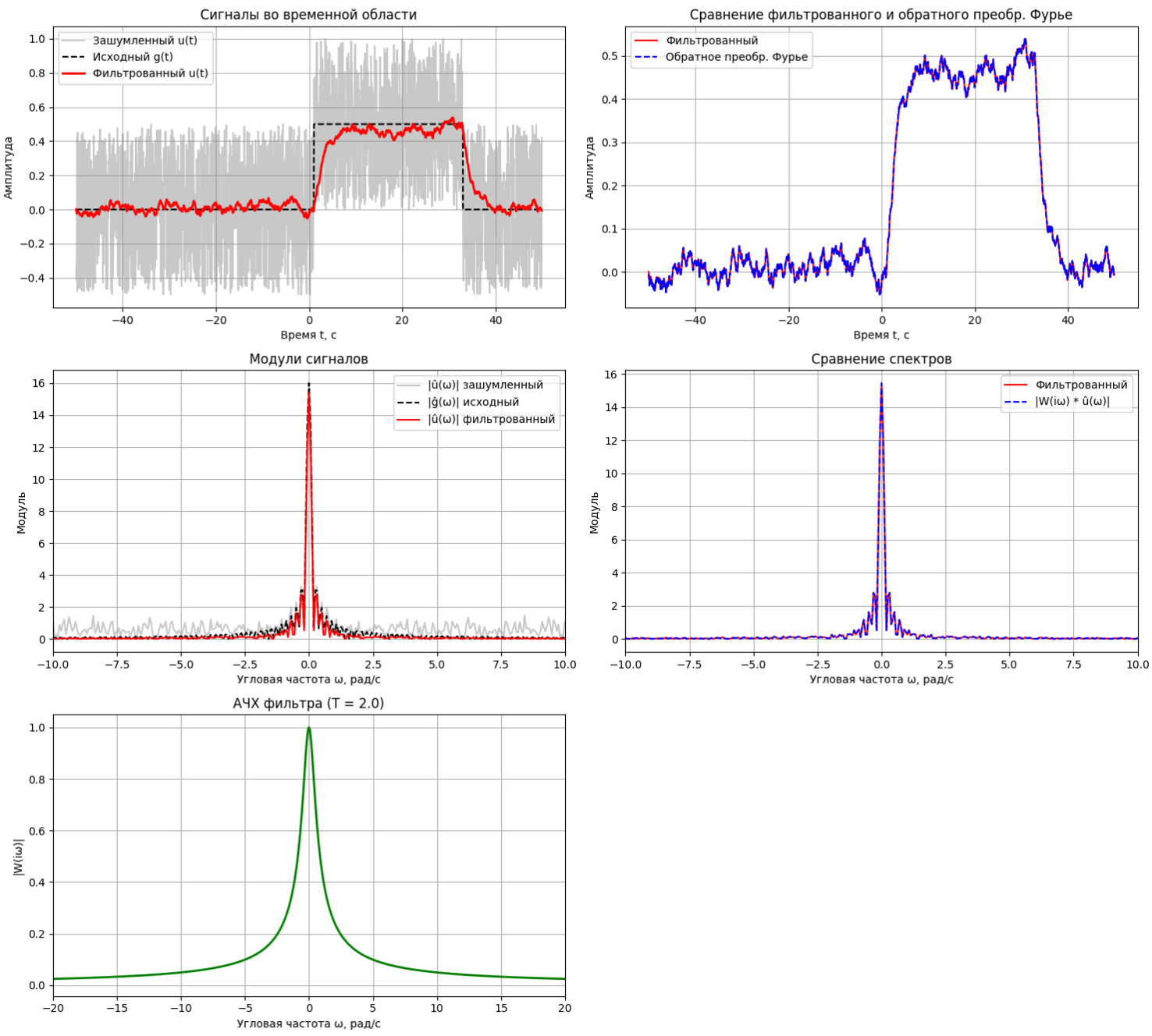

Рис. 4. Исследования влияния T: $a = 0.5, T = 2$

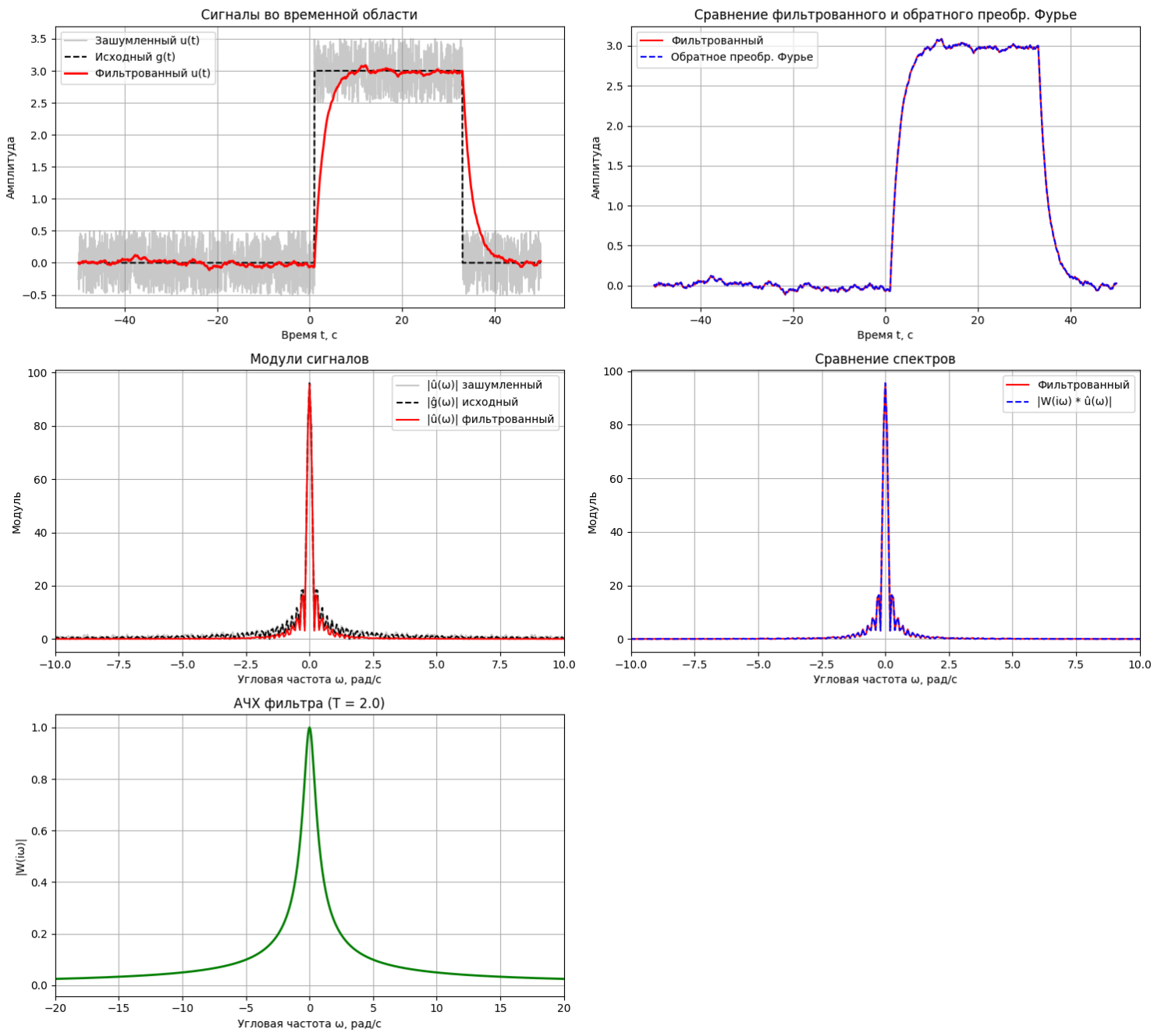

Рис. 5. Исследования влияния T: $a = 3, T = 2$

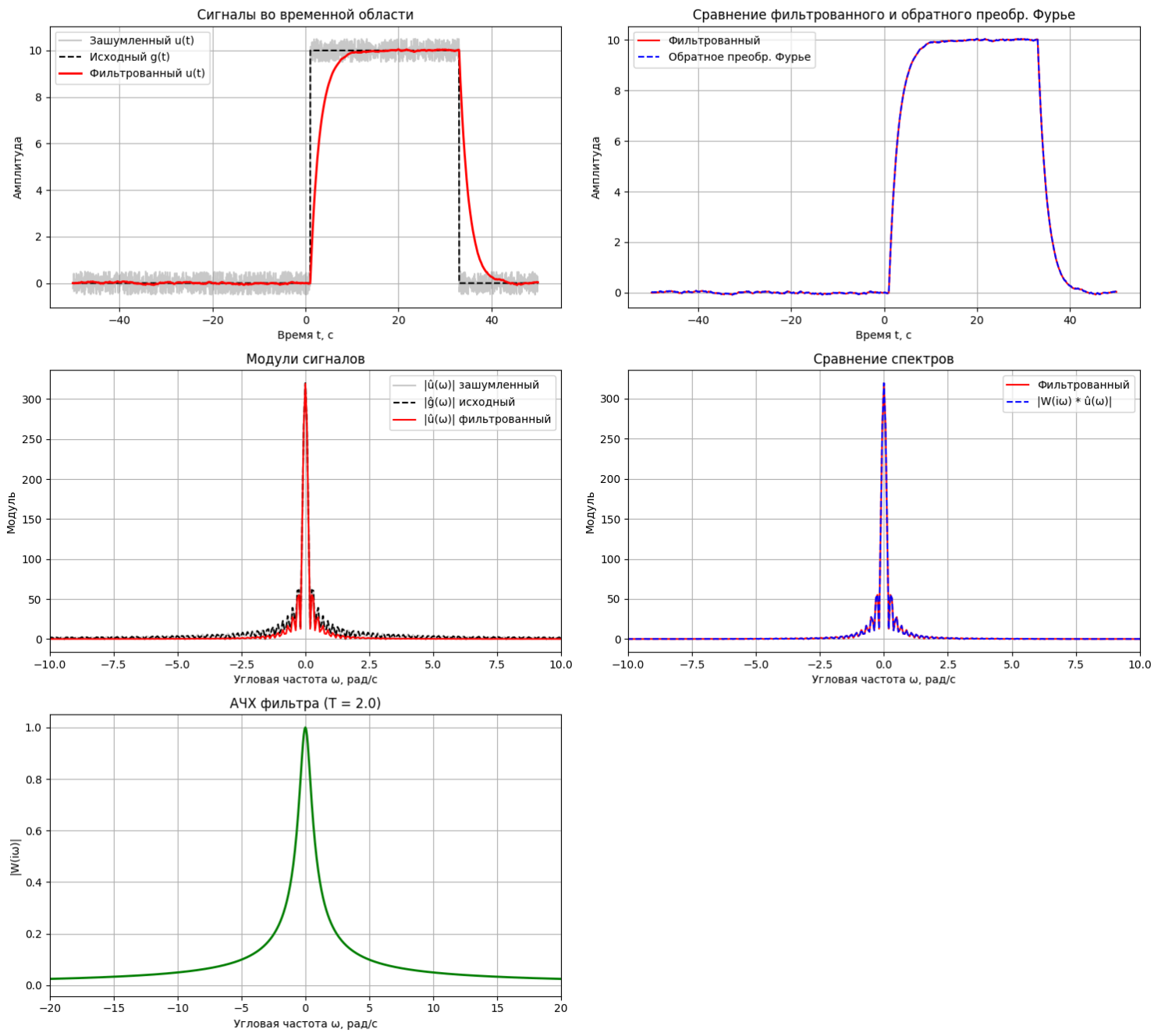

Рис. 6. Исследования влияния T: $a = 10, T = 2$

Изучив приведенные графики, можно заметить, что функция

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

эффективно «отсекает» высокочастотные помехи. Однако при этом возникает искажение формы прямоугольного импульса. С увеличением значения параметра T фильтр работает сильнее и тем самым сильнее искажает исходный сигнал.

Задание 1.2. Режекторный полосовой фильтр

Анализ и выбор значений параметров:

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

При $\omega \to \infty$ $\quad W_2(\omega) = \frac{(i\omega)^2 + a_1(i\omega) + a_2}{(i\omega)^2 + b_1(i\omega) + b_2} \to 1$ (условие выполняетя)

При $\omega \to 0$ $\quad W_2(\omega) = \frac{(i\omega)^2 + a_1(i\omega) + a_2}{(i\omega)^2 + b_1(i\omega) + b_2} \to \frac{a_2}{b_2} = 1$ (по условию) $\Rightarrow a_2 = b_2$

Чтобы при ω_0 $W_2(\omega) = 0$ числитель должен быть равен нулю:

$$(i\omega_0)^2 + a_1(i\omega_0) + a_2 = -\omega_0^2 + i\omega_0 a_1 + a_2 = 0 \Rightarrow \begin{cases} -\omega_0^2 + a_2 = 0 \\ \omega_0 a_1 = 0 \end{cases} \Rightarrow \begin{cases} a_2 = \omega_0^2 \\ a_1 = 0 \end{cases}$$

Получаем, что

$$W_2(p) = \frac{p^2 + \omega_0^2}{p^2 + b_1 p + \omega_0^2}$$

При этом, чтобы фильтр был «устойчивым», должно выполняться

$$\Re(p) = \Re\left(\frac{-b_1 \pm \sqrt{b_1^2 - 4\omega_0^2}}{2}\right) < 0$$

Если $b_1^2 < 4\omega_0^2$, то

$$p = -\frac{b_1}{2} \pm i\frac{4\omega_0^2 - b_1^2}{2} \Rightarrow \Re(p) = -\frac{b_1}{2} \Rightarrow \text{Условие выполняется при } b_1 > 0$$

Если $b_1^2 \geq 4\omega_0^2$, то

$$p_1 = \frac{-b_1 - \sqrt{b_1^2 - 4\omega_0^2}}{2} \text{ очевидно отрицателен при } b_1 > 0$$

$$p_2 = \frac{-b_1 + \sqrt{b_1^2 - 4\omega_0^2}}{2} \text{ отрицателен, если} \sqrt{b_1^2 - 4\omega_0^2} < b_1$$

$$\Rightarrow b_1^2 - 4\omega_0^2 < b_1^2 \Rightarrow -4\omega_0^2 < 0 \text{ выполняется всегда}$$

Таким образом, дополнительным условием является то, что $b_1 > 0$

Исходный код для построения графиков:

```
# Возьмем следующие числа для задания функции g(t)
a = 3
t1 = 1
t2 = 33

# 1. Линейные фильтры
T_total = 100 # большой интервал времени
dt = 0.05 # маленький шаг дискретизации
t = np.arange(-T_total/2, T_total/2, dt) # набор временных шагов

g = np.zeros_like(t)
g[(t >= t1) & (t <= t2)] = a # значения функции g(t)
xi = np.random.uniform(-1, 1, size=t.shape) # дискретные значения
шума
u = g + c * np.sin(d_freq * t) # b = 0

# 1.2 Режекторный полосовой фильтр

# Определение режекторного фильтра W(p) = (p^2 + w0^2) / (p^2 + b1*p
+ w0^2)
num = [1, 0, omega0_const**2]
den = [1, b1_const, omega0_const**2]
filter2 = signal.TransferFunction(num, den)

# Пропускание сигнала u через фильтр
tout, u_filtered, _ = signal.lsim(filter2, u, t_sim)

freqs = fftfreq(len(t), dt)  # массив циклических частот (Гц)
omegas = 2 * np.pi * freqs  # массив угловых частот (рад/с)

u_hat = fft(u)  # образ Фурье зашумленного сигнала
g_hat = fft(g)  # образ Фурье исходного сигнала

W2_iw = ( (1j * omegas)**2 + omega0_const**2 ) / ( (1j * omegas)**2 +
b1_const * (1j * omegas) + omega0_const**2 )
```

```
ach = np.abs(W2_iw) # АЧХ

# Теоретический образ Фурье
u_spec_hat = u_hat * W2_iw

# Обратное преобразование Фурье
u_spec = np.real(ifft(u_spec_hat))
```

Рассмотрим различные значения b_1 при фиксированном d и значения d при фиксированном b_1.

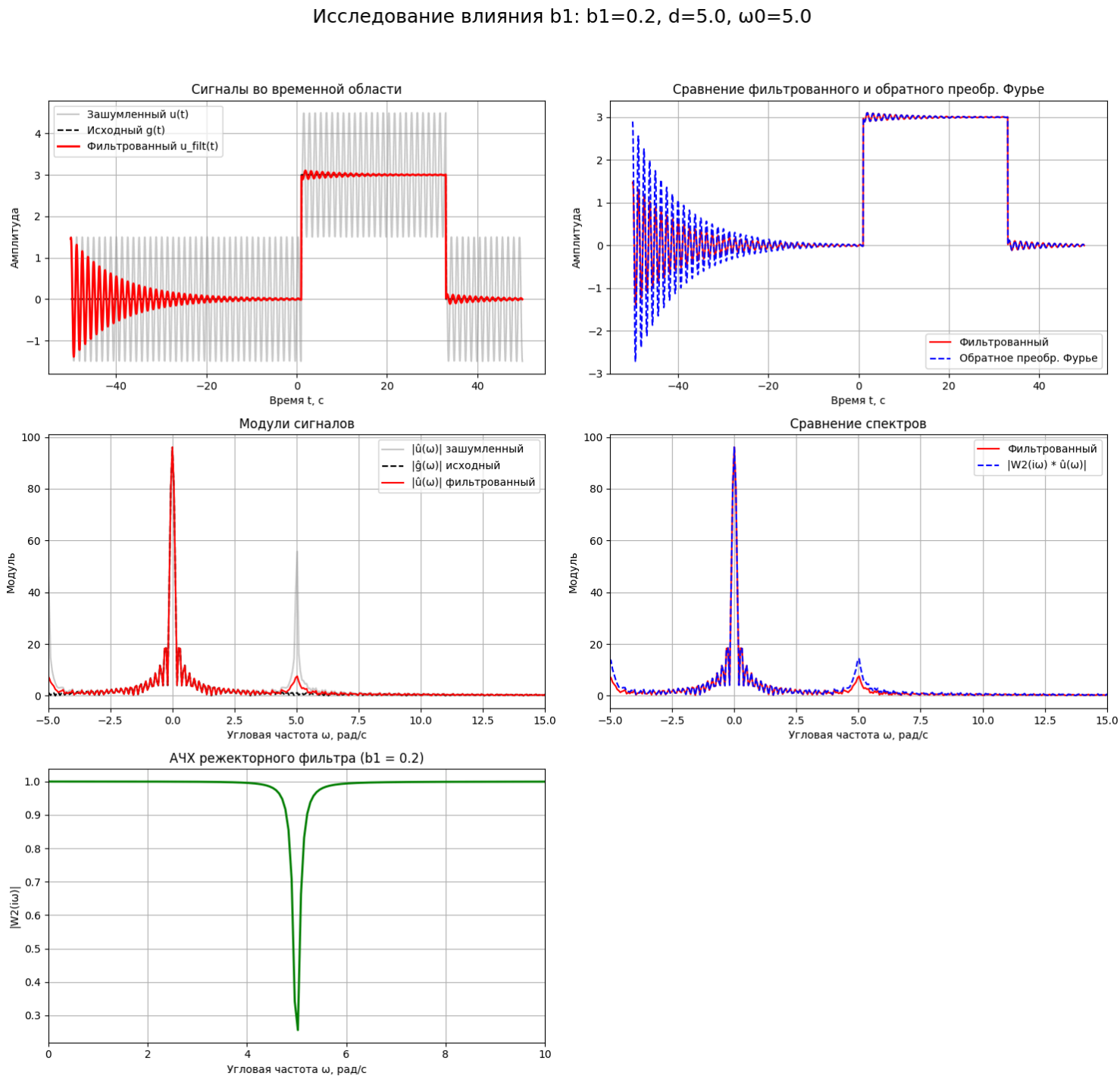

Рис. 7. Исследования влияния b_1: $b_1 = 0.2, d = 5, \omega_0 = 5$

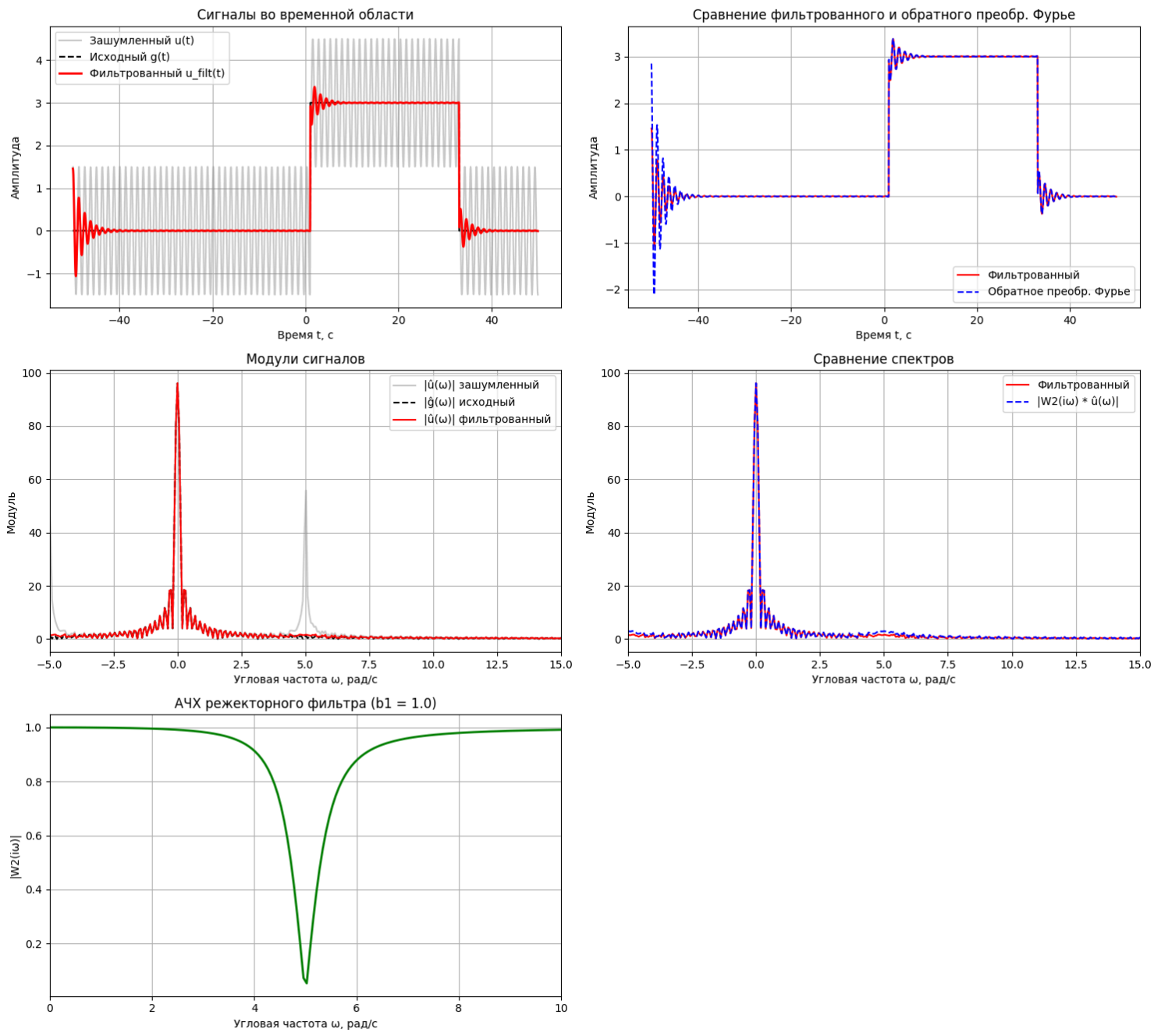

Рис. 8. Исследования влияния b_1: $b_1 = 1, d = 5, \omega_0 = 5$

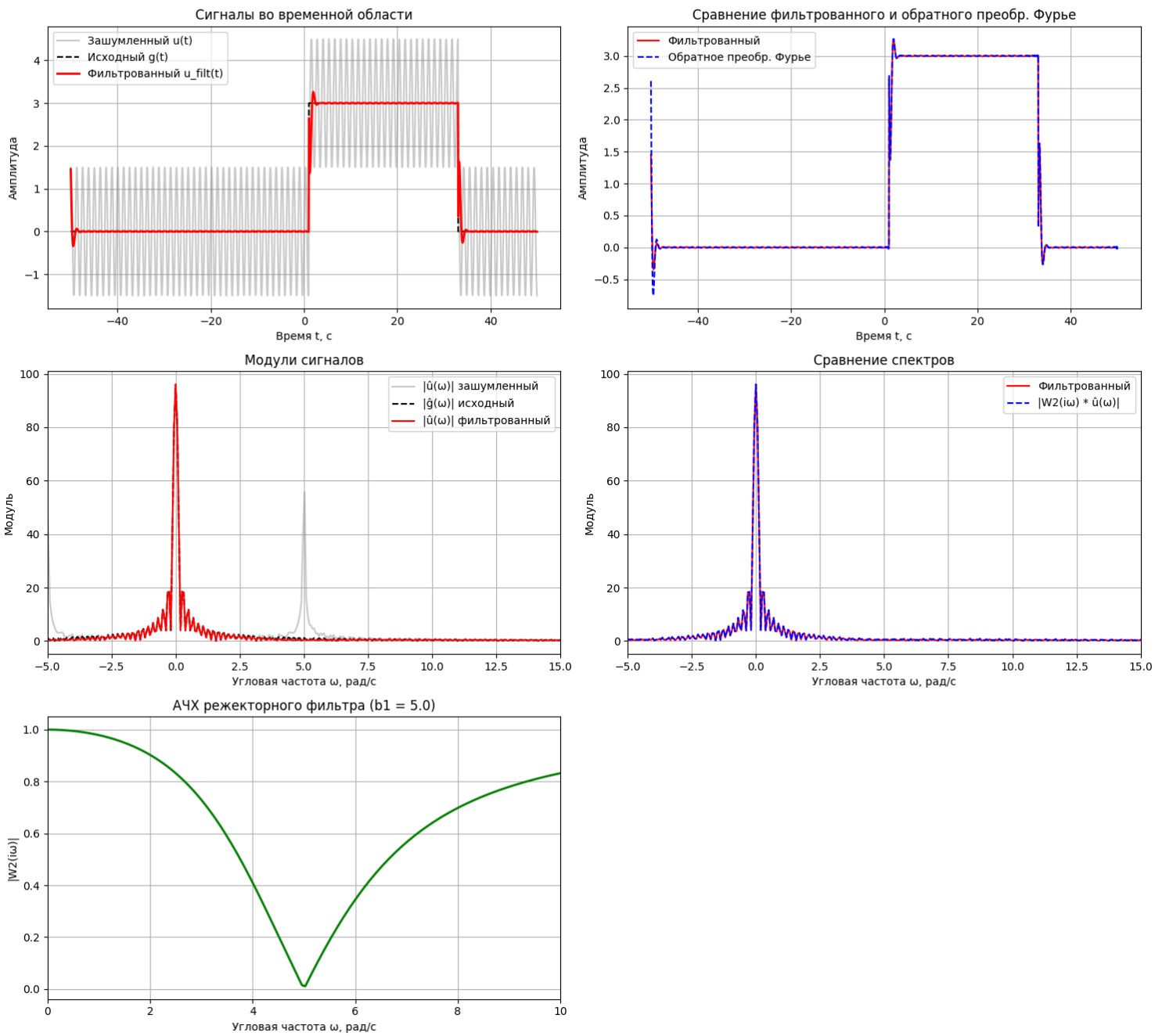

Рис. 9. Исследования влияния b_1: $b_1 = 5, d = 5, \omega_0 = 5$

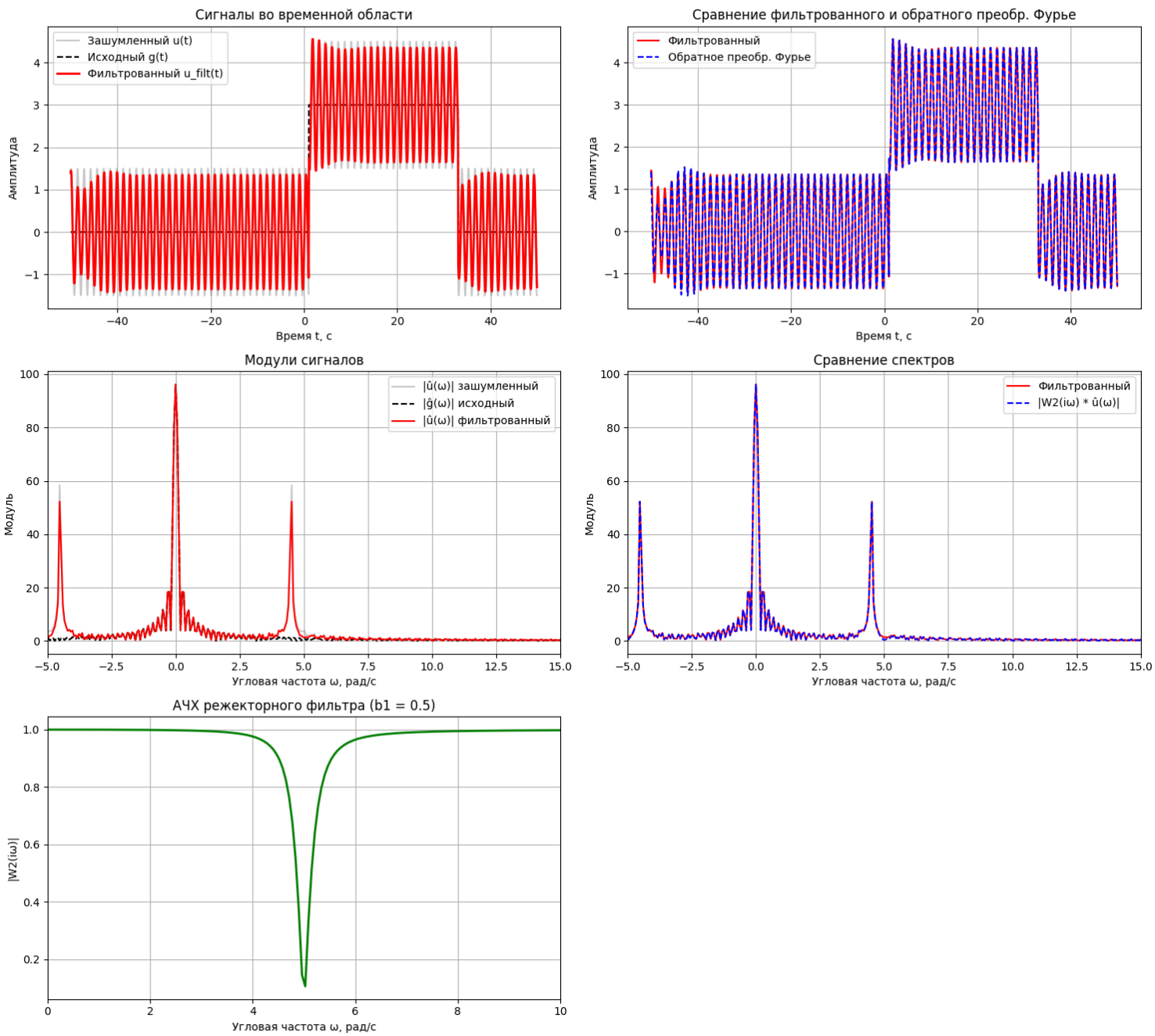

Рис. 10. Исследования влияния d: $b_1 = 0.5, d = 4.5, \omega_0 = 5$

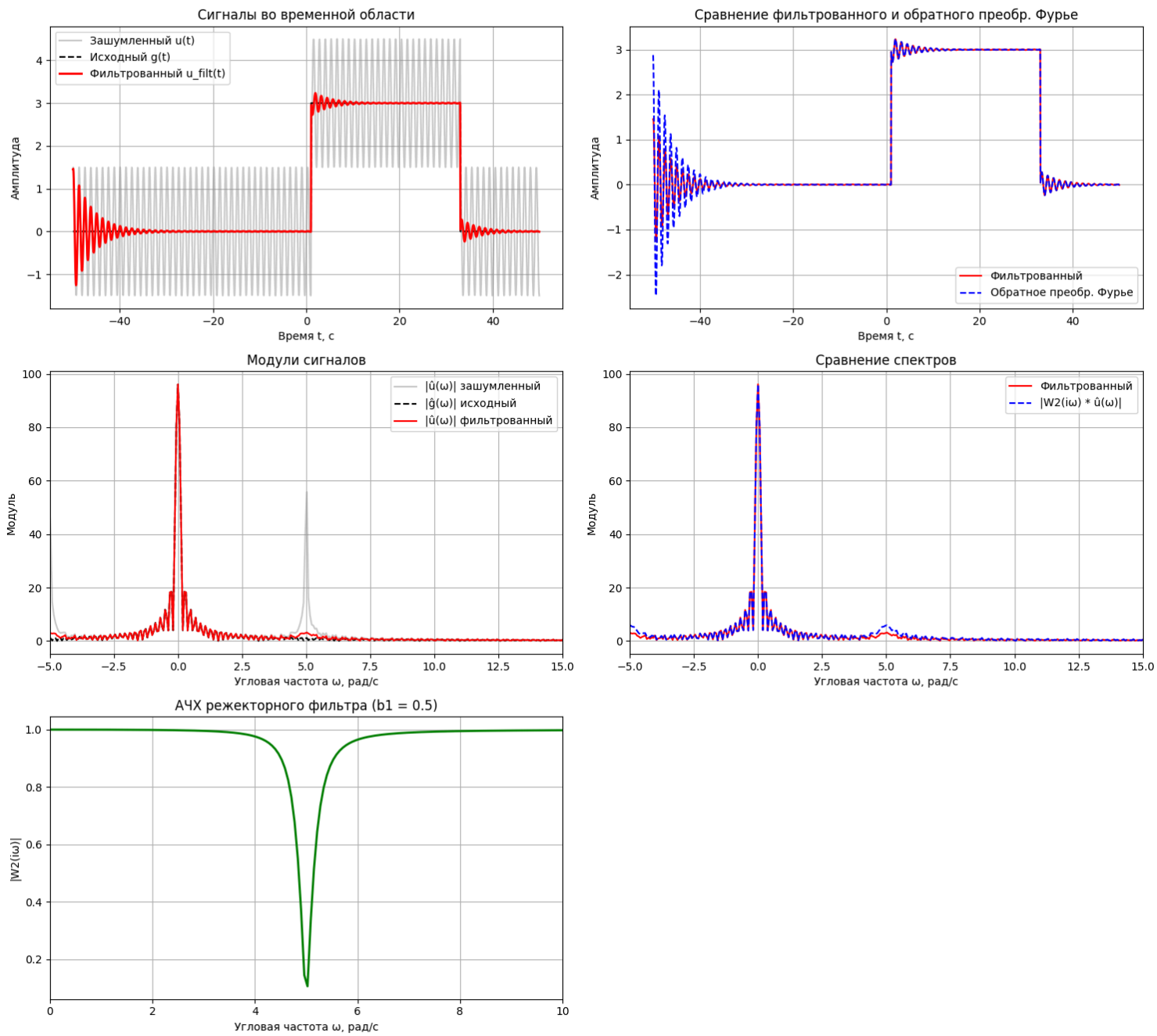

Рис. 11. Исследования влияния d: $b_1 = 0.5, d = 5, \omega_0 = 5$

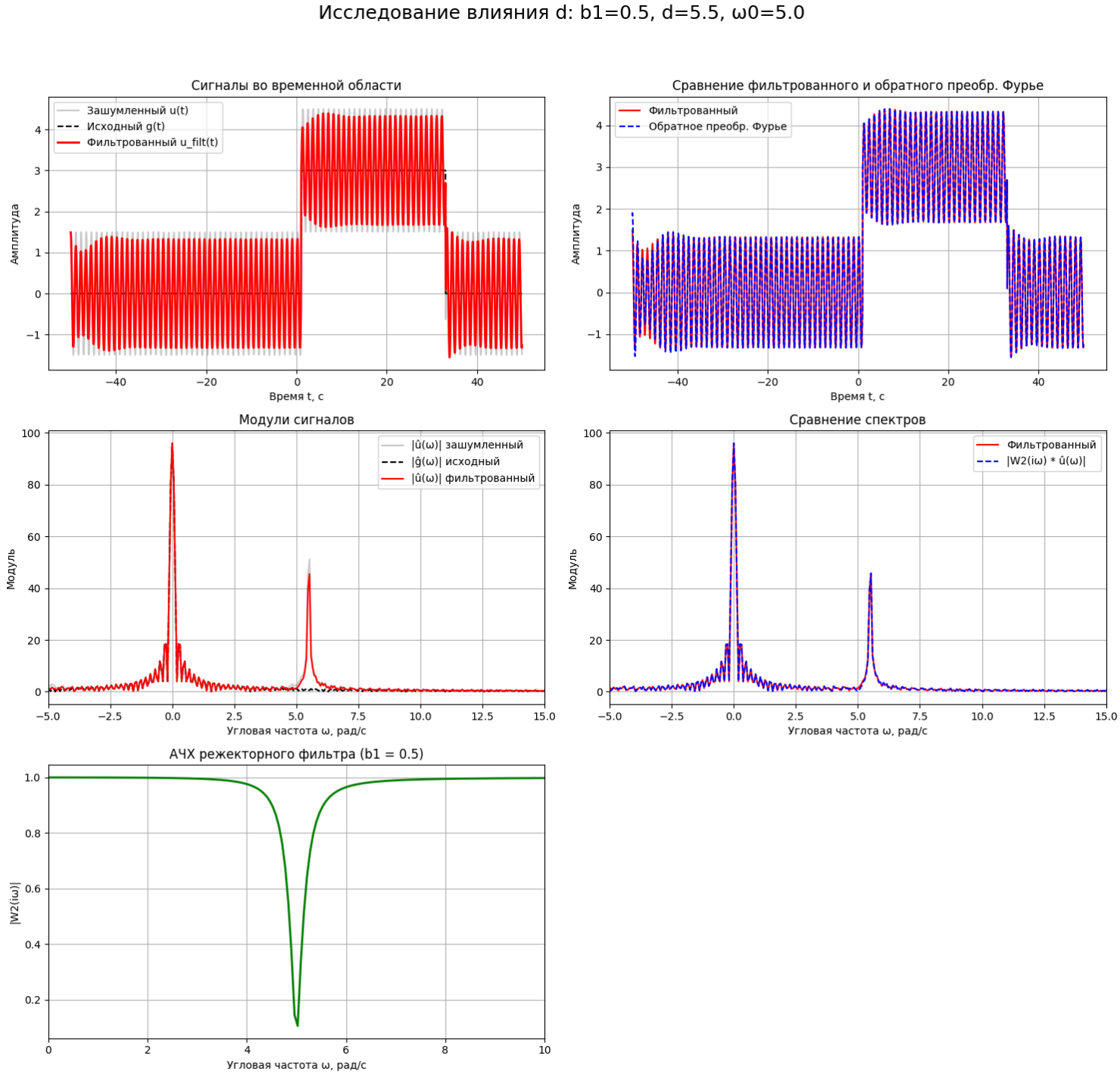

Рис. 12. Исследования влияния d: $b_1 = 0.5, d = 5.5, \omega_0 = 5$

Изучив приведенные графики, можно заметить, что функция

$$W_2(p) = \frac{p^2 + a_1 p + a_2}{p^2 + b_1 p + b_2} = \frac{p^2 + \omega_0^2}{p^2 + b_1 p + \omega_0^2}$$

эффективно очищает помехи, но для работы требуется знать точное значение частоты помехи. Также немаловажен выбор параметра b_1, так как слишком «узкий» фильтр дает долгие переходные процессы.

Задание 2. Сглаживание биржевых данных

Исходный код для построения графиков:

```
df = pd.read_csv('../docs/stocks.csv', sep=';')
prices = df['<CLOSE>'].values
t = np.arange(len(prices))  # Временная шкала в днях

# Определение констант времени (в торговых днях)
time_constants = {
    "1 день": 1,
    "1 неделя": 5,
    "1 месяц": 21,
    "3 месяца": 63,
    "1 год": 252
}


# Настройка фильтра первого порядка W(p) = 1 / (Tp + 1)
num = [1]
den = [T, 1]
filter1 = signal.TransferFunction(num, den)
tout, y, x_out = signal.lsim(filter1, U=prices, T=t, X0=[prices[0]])
```

Рассмотрим графики для заданных периодов T

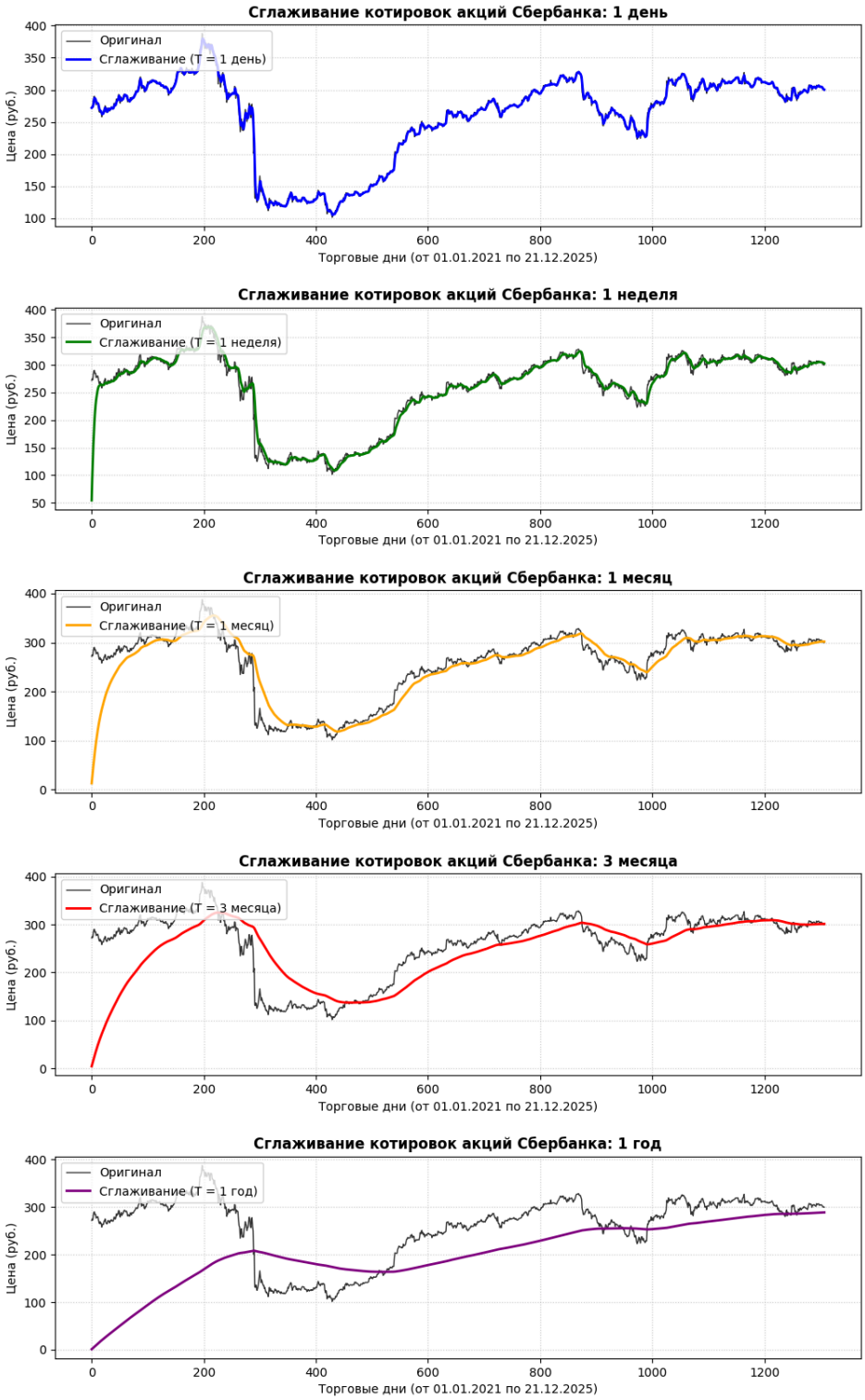

Рис. 13. Сглаженные котировки акций Сбербанка

Изучив приведенные графики, можно заметить, что чем больше значение T, тем сильнее подавляются высокочастотные колебания (шум) и тем более гладким становится график. Однако платой за это является возрастающее запаздывание — отфильтрованный сигнал «отстает» от реального во времени.