МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств

(ТС и ВС)

Лабораторная работа № 1

По дисциплине

«Основы систем мобильной связи»

«Временная и частотная формы сигналов. Преобразования Фурье.

Дискретизация сигналов»

Студент:

Группа № ИА331 Р. К. Рубцов

Предподаватель: В. Г. Дроздова

Новосибирск 2025 г.

Цель занятия:

- Получить представление о формах радиосигналов, их частотном и временном представлении, а также о преобразованиях Фурье и аналоговоцифровых преобразованиях сигналов, частоте дискретизации сигналов.

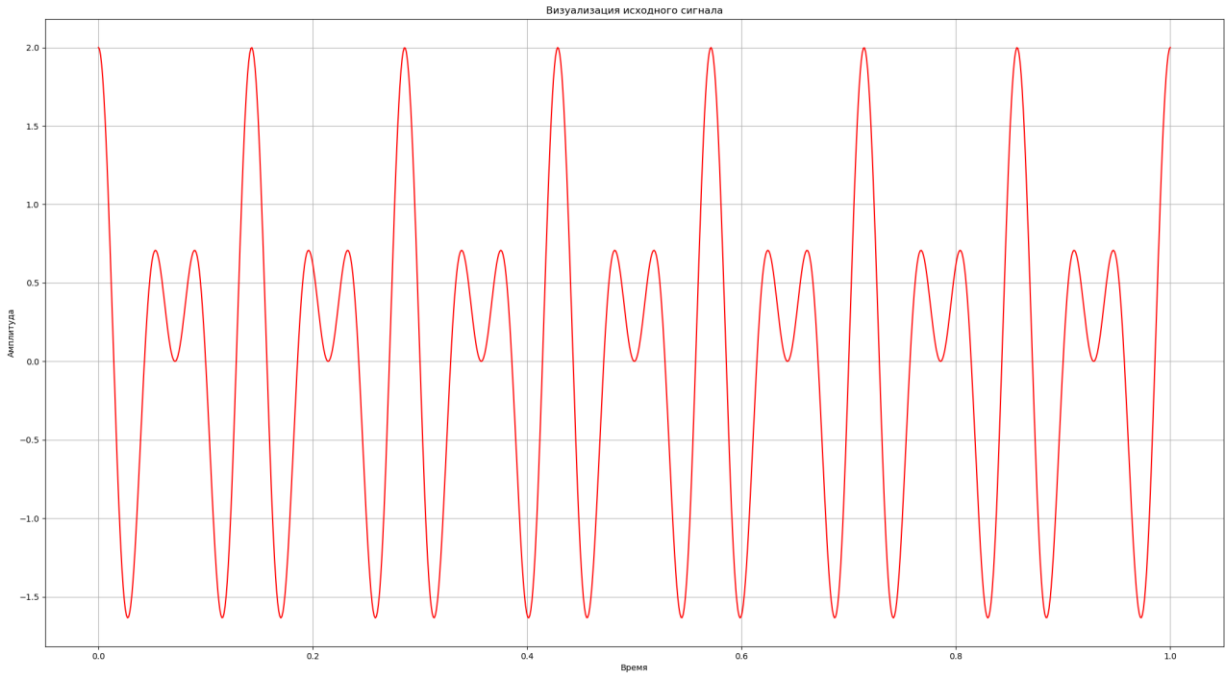
Вариант 26

26	$y(t) = \cos(4\pi f t) + \cos(6\pi f t), \quad f = 7$

Порядок выполнения работы:

1. Необходимо сгенерировать и визуализровать (вывести на график) непрерывный сигнал, в соответствии с вариантом (таблица 1). Номер варианта – порядковый номер в журнале группы.

```
f = 7
t = np.linspace(0, 1, 1000)

s_t = np.cos(4*np.pi * f * t) + np.cos(6*np.pi*f*t)

plt.plot(t, s_t, 'r')
plt.title("Визуализация исходного сигнала")
plt.xlabel("Время")
plt.ylabel("Амплитуда")
plt.grid()
plt.show()
```



2. Определить максимальную частоту в спектре данного сигнала.

w = 2pi*f f = w/2pi 42pi/2pi = 21 fmax = 21

3. Определить минимальную необходимую частоту дискретизации полученного сигнала (теорема Котельникова).

Fs >= 2 * fmax = 42 Гц.

4. Оцифровать сигнал с полученной частотой дискретизации, выбрав требуемое число отсчетов сигнала на длительности 1 секунда, сохранить полученные значения в массив, пока не озадачиваясь разрядностью АЦП, просто выбранные с частотой дискретизации значения с теми уровнями, которые имеет функция в полученных точках.

```
fs = 42
N = fs


t_sampled = np.linspace(0, 1, N, endpoint=False)
s_t_sampled = np.cos(4*np.pi * f * t_sampled) + np.cos(6 * np.pi * f *
t_sampled)
```

5. Выполнить прямое дискретное преобразование Фурье для массива временных отсчетов сигнала и оценить ширину данного спектра. А также объем памяти, требуемый для хранения данного массива (тип переменных в массиве – на ваше усмотрение, float, int и пр.).

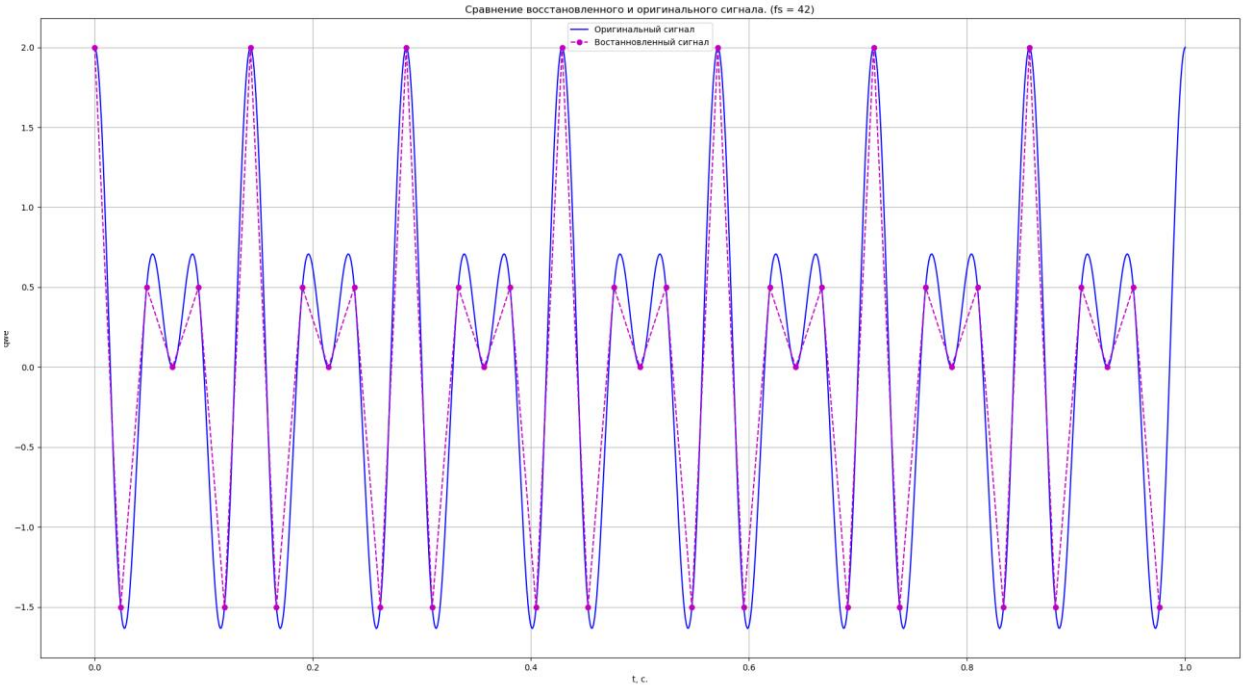
```
# ДПФ

s_t_sampled_f = np.zeros(N, dtype=complex)

for k in range(0, N):
    sum_val = 0
    for n in range(0, N):
        sum_val = sum_val + (s_t_sampled[n] * np.exp(-1j*2*np.pi*((k*n)/N)))
    s_t_sampled_f[k] = sum_val

s_t_sampled_f_amplitude = np.abs(s_t_sampled_f[:N//2 + 1]) # Амплитудная ось
s_t_sampled_f_amplitude_N = np.abs(s_t_sampled_f[:N//2 + 1])/N # Амплитудная
ось (Нормированная)

s_t_sampled_f_freq = np.zeros(N//2 + 1) # Частотная ось
for k in range(0, N//2 + 1):
    s_t_sampled_f_freq[k] = (k/N)*fs

# Находим ширину спектра
peak_freqs = s_t_sampled_f_freq[s_t_sampled_f_amplitude_N > 0.4]

bandwidth = peak_freqs.max() - peak_freqs.min()

print(f'Ширина спектра: {bandwidth}.')

# Находим обьем памяти занимаемый s_t_sampled_f

print(f'Объем памяти выделенной под массив временных отсчетов сигнала:
{s_t_sampled_f.nbytes} байт.')
```

Ширина спектра: 7.0.

Объем памяти выделенной под массив временных отсчетов сигнала: 672 байт.

6. Восстановите оригинальный аналоговый сигнал по массиву имеющихся у вас отсчетов, соединив их непрерывной линией и оцените визуальное сходство оригинального сигнала и восстановленного после оцифровки

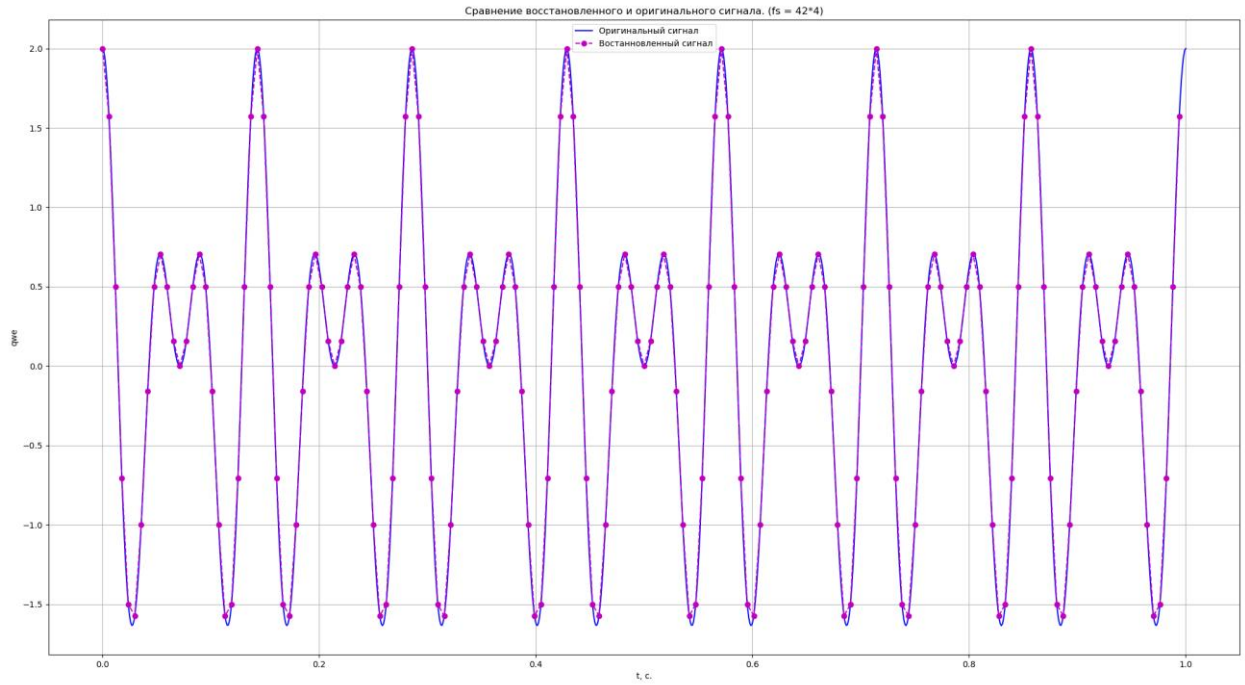
```
plt.plot(t, s_t, 'b', label="Оригинальный сигнал")
plt.plot(t_sampled, s_t_sampled, 'm--', marker='o', label="Востанновленный
сигнал")
plt.title("Сравнение восстановленного и оригинального сигнала. (fs = 42)")
plt.xlabel("t, c.")
plt.ylabel("qwe")
plt.grid()
plt.legend()
plt.show()
```



7. Увеличьте частоту дискретизации в 4 раза и проделайте задания из п.4-6.

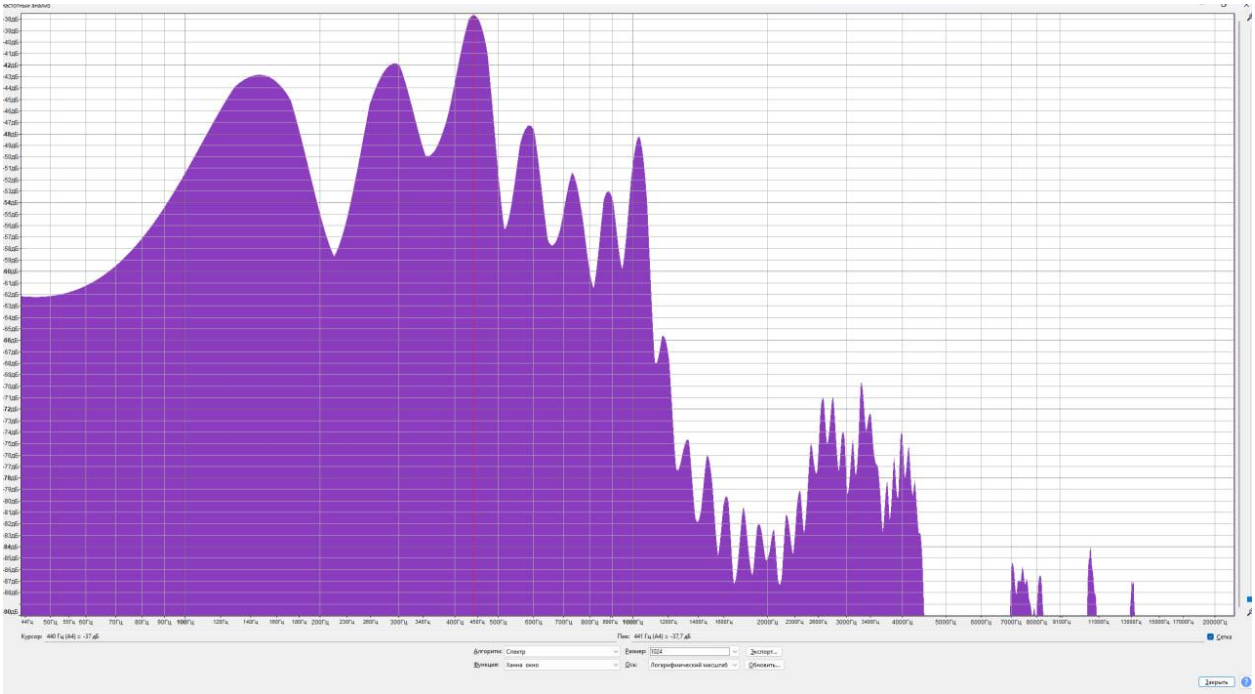
Ширина спектра: 7.0. (fs = 42*4)

Объем памяти выделенной под массив временных отсчетов сигнала: 2688 байт. (fs = 42*4)



Как мы видим частоты дискретизации 42*4 хватило чтобы весьма хорошо восстановить оригинальный сигнал

8. Запишите аудиофайл со своим голосом (например, в формате wav). Проанализируйте визуально спектр голоса, используя, например, приложение 12 Audacity (вкладки Анализ -> График спектра). Определите максимальную частоту в спектре данного сигнала и выберите требуемую для оцифровки частоту дискретизации.



Максимальная частота в спектре около 4.5 кГц, выше есть какие-то пики но это скорее фон или обертона. Частота дискретизации нужна около 9 кГц

9. Используя библиотеки работы со звуком Matlab (или Pyton) проанализируйте имеющуюся у вас запись голоса. Считайте аудиофайл с помощью функции Matlab (файл с голосом должен лежать в рабочей директории Matlab): (в моем случае python)

```
Fs, y = wavfile.read('vocal.wav') # fs - частота дискретизации, y - массив отсчетов
```

10. Определите частоту дискретизации, которая была использована при записи голоса на цифровой носитель. Для этого возьмите количество элементов в файле с записью и разделите на длительность записи в секундах. Например, если у вас файл состоит из 441 000 элементов (отсчетов) и имеет длительность 10 секунд, то частота дискретизации Fs будет равна 441 000/10 = 44 100 Гц. Сравните вычисленное значение с тем, что выводится в Matlab для Fs.

```
=== ШАГ 10: Проверка частоты дискретизации ===
Длительность записи: 2.627 секунд
Частота дискретизации из файла (Fs): 44100 Гц
Вычисленная частота дискретизации (Fs_calc = N / T): 44100.0 Гц
Совпадает — частота дискретизации определена корректно.
```

```
N = len(y) # кол-во отсчетов

duration = N / Fs # длительность записи

Fs_calc = N / duration


print("Частота дискретизации из файла:", Fs)

print("Частота дискретизации вычисленная:", Fs_calc)
```

Частота дискретизации из файла: 44100

Частота дискретизации вычисленная: 44100.0

11. Проредите полученный массив y с помощью функции downsample (иными словами – уменьшите частоту дискретизации) и воспроизведите полученный сигнал с помощью matlab-функции play() или ее аналогами в других языках программирования. Обратите внимание на качество звучания при неверно взятой частоте дискретизации:

```
y1 = y[::10]  # срез с шагом 10

Fs1 = Fs // 10  # новая частота дискретизации


wavfile.write('vocal_downsampled.wav', Fs1, y1)


print(f'Новая частота дискретизаии {Fs1}')
```

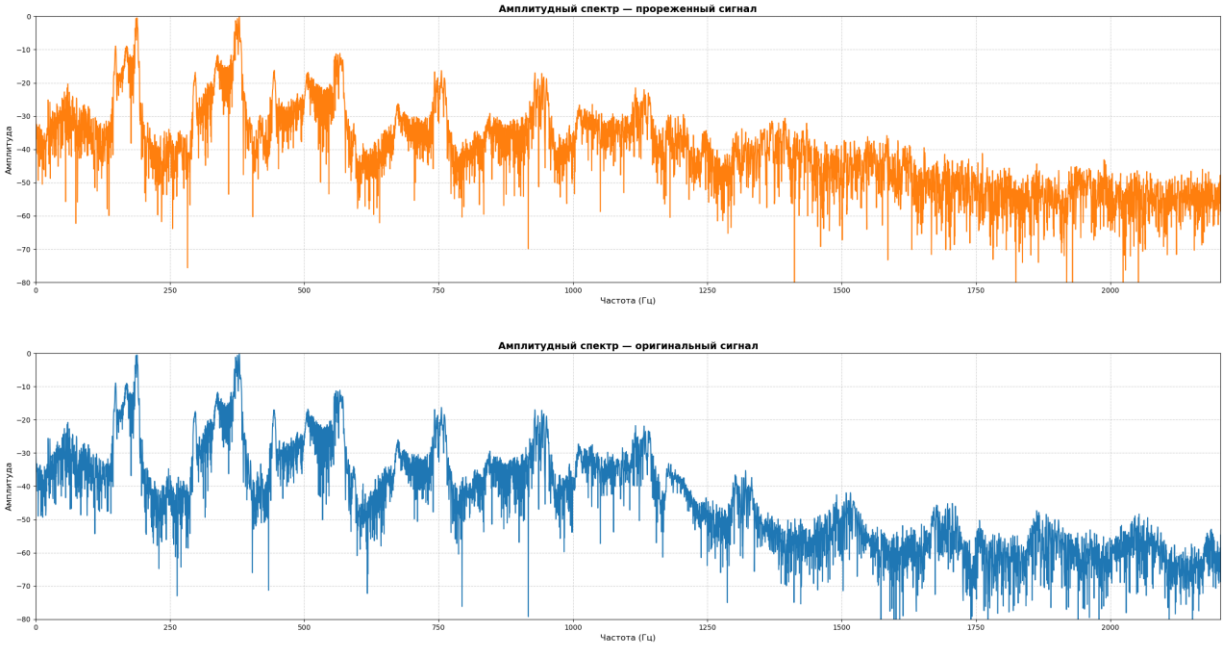
Новая частота дискретизаии 4410

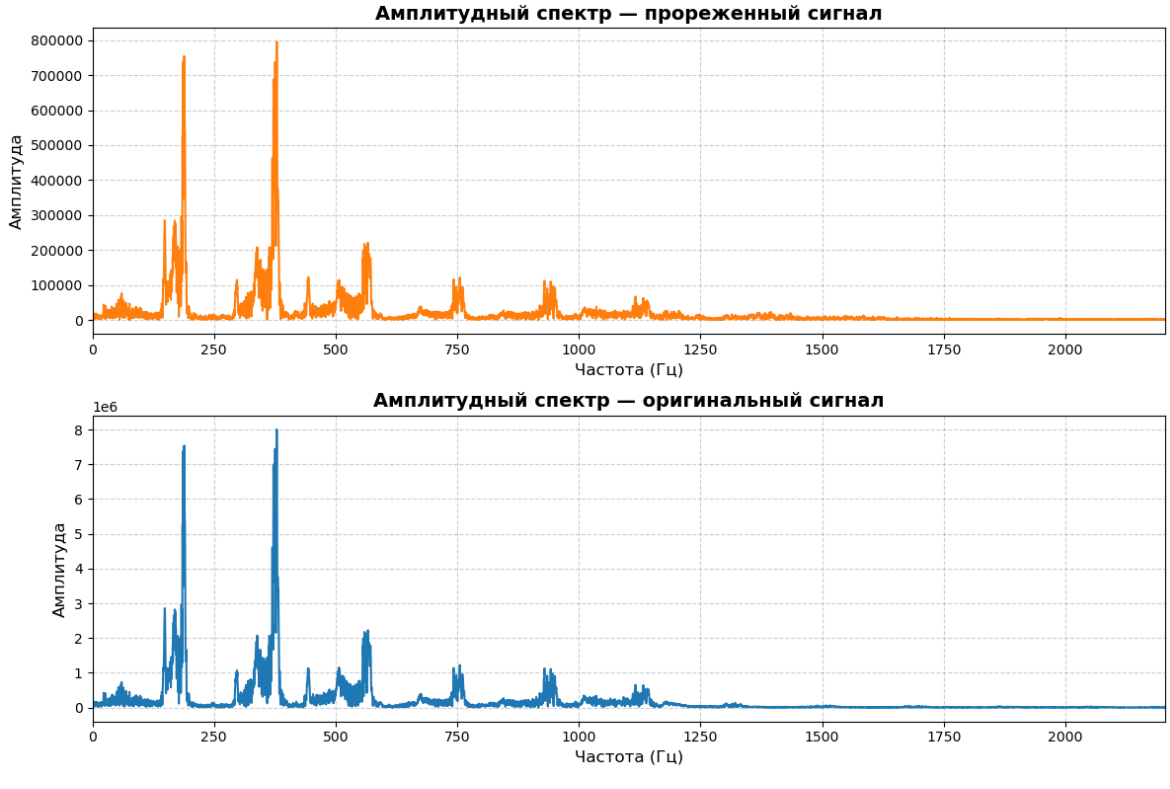
Сигнал стал глухим, но его все еще можно разобрать

12. Выполните прямое дискретное преобразование Фурье для оригинального звучания и для прореженного сигнала, выведите на график амплитудный спектр сигнала, определите его ширину.

```
# 12.1 для оригинального

y_fft = fft(y)
y_fft_freq = fftfreq(N, 1/Fs)
y_fft_half = y_fft[:N//2 + 1]
y_fft_freq_half = y_fft_freq[:N//2 + 1]

amp = np.abs(y_fft_half)

amp_db = 20 * np.log10(amp/amp.max())

freqs = y_fft_freq_half[-40 < amp_db]
bandwidth = freqs.max() - freqs.min()
print(f'Ширина {bandwidth:.1f} Гц.')

# 12.2 для прореженного

N1 = len(y1)

y_fft1 = fft(y1)
y_fft_freq1 = fftfreq(N1, 1/Fs1)
y_fft_half1 = y_fft1[:N1//2 + 1]
y_fft_freq_half1 = y_fft_freq1[:N1//2 + 1]

amp1 = np.abs(y_fft_half1)

amp_db1 = 20 * np.log10(amp1/amp1.max())

freqs1 = y_fft_freq_half1[-40 < amp_db1]
bandwidth1 = freqs1.max() - freqs1.min()
print(f'Ширина {bandwidth1:.1f} Гц.')
plt.figure(figsize=(12, 8))

# прореженный сигнал
plt.subplot(2, 1, 1)
plt.plot(y_fft_freq_half1, amp_db1, color='tab:orange', linewidth=1.5)
plt.title('Амплитудный спектр — прореженный сигнал', fontsize=14,
fontweight='bold')
plt.xlabel('Частота (Гц)', fontsize=12)
plt.ylabel('Амплитуда', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlim(0, Fs1/2)
plt.ylim(-80, 0)

# оригинальный сигнал
plt.subplot(2, 1, 2)
plt.plot(y_fft_freq_half, amp_db, color='tab:blue', linewidth=1.5)
```

```
plt.title('Амплитудный спектр — оригинальный сигнал', fontsize=14,
fontweight='bold')
plt.xlabel('Частота (Гц)', fontsize=12)
plt.ylabel('Амплитуда', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlim(0, Fs1/2)
plt.ylim(-80, 0)

plt.tight_layout()
plt.show()
```





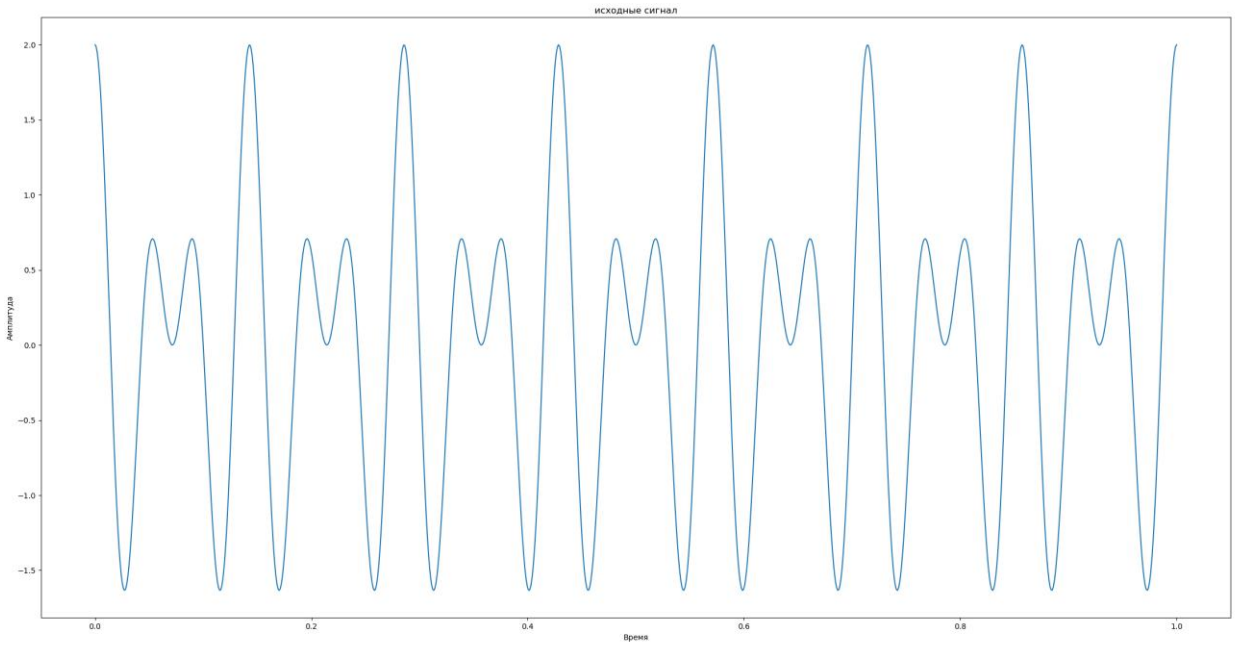
Ширина оригинального 4372.4 Гц.

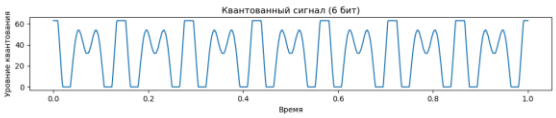
Ширина прореженного 1653.3 Гц.

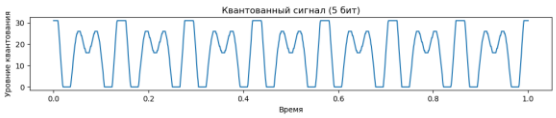
13. Оцените влияние разрядности АЦП на спектр сигнала. Для этого нужно написать функцию, которая бы округляла значения отсчетов сигнала, заданного в вашем варианте, до какого-то числа, определяемого разрядностью АЦП. Допустим если у АЦП всего 3 разряда, то диапазон возможных дискретных значений амплитуд временных отсчетов сигнала – это 0..7 (то есть, 7=2 3 -1). Все значения больше 7 округляются до 7. Для результирующего дискретного сигнала требуется выполнить прямое преобразование Фурье. Сравнить полученный спектр со спектром исходной синусоиды, отсчеты которой не подвергались квантованию по уровню. Вывести среднюю ошибку квантования для случаев, когда разрядность АЦП равна 3/4/5/6

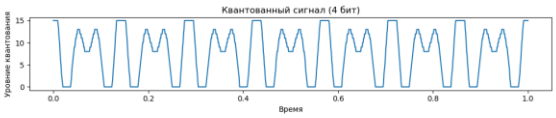
```
## 13 Оцените влияние разрядности АЦП на спектр сигнала.

def MSEcalc(signal, signal_quant):
    total = 0
    for i in range(0, len(signal)):
        total = total + np.abs(signal[i] - signal_quant[i])

    return (1/len(signal))*total

def sround(signal, bits):
    signal = np.clip(signal, -1, 1)
    levels = 2**bits-1
    signal_quant_lev = np.zeros(len(signal))
    signal_quant_amp = np.zeros(len(signal))
    norm = np.zeros(len(signal))

    min = signal.min()
    max = signal.max()

    norm = (signal - min) / (max - min)

    signal_quant_lev = np.round(norm * levels)

    signal_quant_amp = ((signal_quant_lev / levels) * (max - min)) + min

    mse = MSEcalc(signal, signal_quant_amp)

    print(f'Ошибка квантования при {bits} битах: {mse}')

    return signal_quant_lev

f = 7
t = np.linspace(0, 1, 1000)

s_t = np.cos(4*np.pi * f * t) + np.cos(6*np.pi*f*t)
```

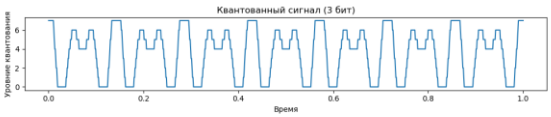
```
49.
50.plt.plot(t, s_t)
51.plt.title("исходные сигнал")
52.plt.xlabel("Время")
53.plt.ylabel("Амплитуда")
54.
55.plt.tight_layout()
56.plt.show()
57.
58.s_t_quant6 = sround(s_t, 6)
59.plt.subplot(4, 2, 1)
60.plt.plot(t, s_t_quant6)
61.plt.title("Квантованный сигнал (6 бит)")
62.plt.xlabel("Время")
63.plt.ylabel("Уровние квантования")
64.
65.s_t_quant5 = sround(s_t, 5)
66.plt.subplot(4, 2, 2)
67.plt.plot(t, s_t_quant5)
68.plt.title("Квантованный сигнал (5 бит)")
69.plt.xlabel("Время")
70.plt.ylabel("Уровние квантования")
71.
72.s_t_quant4 = sround(s_t, 4)
73.plt.subplot(4, 2, 3)
74.plt.plot(t, s_t_quant4)
75.plt.title("Квантованный сигнал (4 бит)")
76.plt.xlabel("Время")
77.plt.ylabel("Уровние квантования")
78.
79.s_t_quant3 = sround(s_t, 3)
80.plt.subplot(4, 2, 4)
81.plt.plot(t, s_t_quant3)
82.plt.title("Квантованный сигнал (3 бит)")
83.plt.xlabel("Время")
84.plt.ylabel("Уровние квантования")
85.
86.plt.tight_layout()
87.plt.show()
```











Ошибка квантования при 6 битах: 0.005211954748172533

Ошибка квантования при 5 битах: 0.010781475616119687

Ошибка квантования при 4 битах: 0.022444356248891285

Ошибка квантования при 3 битах: 0.04515611921668369

В ходе выполнения лабораторной работы были изучены основные принципы представления сигналов во временной и частотной областях, а также влияние параметров дискретизации и квантования на качество цифрового сигнала.

На примере сигнала y(t) = cos(4π·7·t) + cos(6π·7·t) (частоты 14 Гц и 21 Гц) было показано:

1. Спектр сигнала содержит две дискретные частоты — 14 Гц и 21 Гц. Максимальная частота f_max = 21 Гц, следовательно, по теореме Котельникова, минимальная частота дискретизации должна быть больше или равна 42 Гц. Были выбраны fs = 42 Гц (корректно) и fs = 42*4 Гц (с запасом) — в обоих случаях спектр восстанавливался точно, а сигнал — без искажений
2. При увеличении частоты дискретизации улучшается точность восстановления сигнала (меньше "ступенек"), но возрастает объём занимаемой памяти — с 672

байт (при 42 Гц) до 2688 байт (при 42*4 Гц). Это демонстрирует компромисс между качеством и ресурсам

3. При прореживании сигнала до fs = 4410 Гц (ниже 2·f_max для голоса) было наглядно продемонстрировано нарушение теоремы Котельникова:
    - Спектр исказился – выскоеи частоты наложились на низкие (алиасинг)
    - Звук потерял информацию на высоких частотах, нет сибилянтов, он сам звучит тусклее
    - Ширина спектра уменьшилась, тк частоты выше половины новой частоты дискретизации были потеряны или искажены

4. Анализ разрядности АЦП показал, что при малом числе бит (3–4) в спектре появляются шумы и искажения, а при 5–6 битах — сигнал восстанавливается практически без потерь. Ошибка квантования (MSE) уменьшается в ~4 раза при каждом добавлении бита — что соответствует теории.
5. Для голосового сигнала было показано, что его спектр — широкополосный, содержит множество частотных компонентов, в отличие от простых синусоид. Использование порога -40 дБ позволило корректно определить ширину полезного спектра (~4372 Гц), что подтвердило необходимость частоты дискретизации не менее 9–10 кГц для качественной оцифровки речи.

- Преобразование Фурье — мощный инструмент для перехода от временного представления сигнала к частотному. Оно позволяет выявить скрытые частотные компоненты, оценить ширину спектра и принять обоснованные решения по параметрам оцифровки.
- Теорема Котельникова — фундаментальный закон цифровой обработки: при нарушении условия fs > 2·f_max возникает алиасинг — необратимое искажение сигнала.
- Разрядность АЦП напрямую влияет на точность представления сигнала: малая разрядность вносит шум квантования, большая — требует больше памяти.
- Голос — сложный сигнал, и его спектр кардинально отличается от спектра синусоид. Для его качественной оцифровки нужны и достаточная частота дискретизации, и адекватная разрядность АЦП.

Контрольные вопросы

1. Для чего используются прямое и обратное преобразование Фурье?
    - Прямое преобразование Фурье нужно для того чтобы перевести сигнал из временной области в частотную, показывая из каких гармоник он состоит.
    - Обратное преобразование Фурье восстанавливает временную область сигнала по его частотному спектру.
2. Что такое ошибка квантования и дискретизации?
    - Ошибка дискретизации – это искажение, возникающее из-за недостаточной частоты дискретизации
    - Ошибка квантования – это разница между настоящим значением сигнала и по дозволенному уровнем значению.
3. Какое количество разрядов АЦП требуется, чтобы оцифровать голос?

В случае лабораторной работы 5-6 бит было достаточно, но например для телефонной связи обычно используют 8 бит, для хорошей записи речи 16 бит, этого более чем хватает, чтобы шум квантования был не слышен на фоне речи.

4. Как математически получить дискретные отсчеты непрерывного сигнала?
   Путем умножения аналогового сигнала на решетчатую функцию (гребенку Дирака) с периодом 1/fs - $s_{d(t)} = s(t) * \sum_{-\infty}^{\infty} \delta(t - nT)$, но на практике это значит просто взятие значений сигнала в моменты x[n] = x(n*T), n = 0, 1, 2…
5. Какой спектр у периодического сигнала sin(10πt+π/2)?

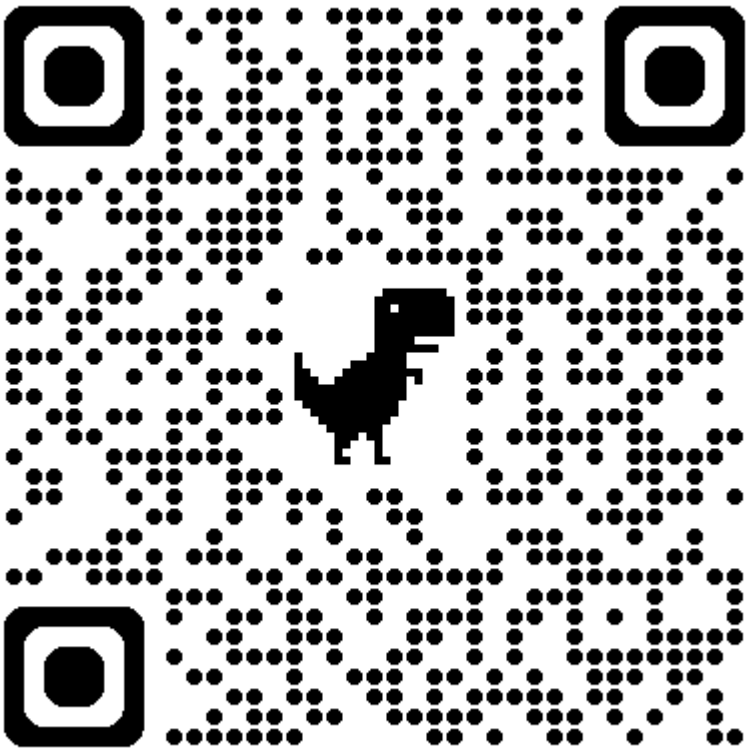
sin(10πt+π/2) = cos(10πt) f0 = 5 Hz.

$$cos2pift \rightarrow \frac{1}{2}\delta(f - f0) + \frac{1}{2}\delta(f + f0)$$

→ спектр это две дельта функции на частоте $\pm$ 5 Гц.

6. Что такое быстрое преобразование Фурье?
   Быстрое преобразование Фурье (FFT) – это более быстрая вариация ДПФ, он рекурсивно разбивает сигнал на четные и нечетные отчеты вычисляет спектры для этих половин и потом объединяет их с учетом фазовых сдвигов, что снижает вычислительную сложнуть с O н^2 до O NlogN
7. Как определяется минимальная требуемая для оцифровки частота дискретизации сигнала?
   Fs >= 2fmax

Гитхаб



https://github.com/ohfuckinglucy/OSMS