ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

ПРАКТИКУМ

СОТНИКОВ А.А.,
КИМ Т.А., АРЕЩЕНКОВ Д.А.

А.А. Сотников, Т.А. Ким, Д.А. Арещенков

Цифровая обработка сигналов информационно-управляющих систем

Практикум

Электронное издание
локального распространения

Санкт-Петербург
Наукоемкие технологии
2022

ISBN 978-5-907618-10-7

УДК 004.94
ББК 30в6
С67

Рецензент:
доктор технических наук, профессор
Сюзев Владимир Васильевич,
МГТУ им. Н.Э. Баумана

С67 Сотников А.А., Ким Т.А., Арещенков Д.А. Цифровая обработка сигналов информационно-управляющих систем [Электронный ресурс]: практикум / А.А.Сотников, Т.А.Ким, Д.А.Арещенков. – Электрон, текстовые дан. (8,8 Мб). – СПб.: Наукоемкие технологии, 2022. – 96 с. – 1 электрон., опт. диск (CD-ROM).

ISBN 978-5-907618-10-7

В основе книги лежит восемь практических задач из теории цифровой обработки сигналов информационно-управляющих систем: различные методы фильтрации, алгоритм Герцеля, цифровое гетеродинирование, демодуляция фазоманипулированных сигналов и преобразование Гильберта-Хуанга. Решение каждой задачи сопровождается необходимыми краткими теоретическими сведениями, а также кодом в программной среде MathWorks® MATLAB®.

Книга представляет собой практический курс, ориентированный на студентов — бакалавров и магистров, специалистов, проходящих подготовку по направлениям обучения «Информатика и вычислительная техника», «Информационные системы и технологии», «Компьютерная безопасность», «Информационная безопасность» и «Прикладная математика и информатика», а также по другим направлениям, учебные планы которых содержат дисциплины, использующие цифровую обработку сигналов. Она будет полезна также аспирантам, инженерам и научным работникам, специализирующимся в области цифровой обработки сигналов информационно-управляющих систем.

Текстовое электронное издание
Минимальные системные требования:

- процессор: Intel x86, x64, AMD x86, x64 не менее 1 ГГц;
- оперативная память RAM ОЗУ: не менее 512 МБайт;
- свободное место на жестком диске (HDD): не менее 120 МБайт;
- операционная система: Windows XP и выше;
- Adobe Acrobat Reader;
- дисковод CD-ROM;
- мышь.

УДК 004.94
ББК 30в6

ISBN 978-5-907618-10-7

Учебное издание

Сотников Алексей Александрович
Ким Тамара Александровна
Арещенков Дмитрий Александрович

Цифровая обработка сигналов
информационно-управляющих систем

Практикум

Текстовое электронное издание

Издательство «Наукоемкие технологии»
ООО «Корпорация «Интел Групп»
https://publishing.intelgr.com
E-mail: publishing@intelgr.com
Тел.: +7 (812) 945-50-63

Подписано к использованию 02.09.2022
Объем издания — 8,8 Мб
Комплектация издания — 1 CD
Тираж 100 CD

ISBN 978-5-907618-10-7

Оглавление

Введение в цифровую обработку сигналов информационно-управляющих систем

Сигнал — это физический процесс, являющийся средством переноса информации [1]. Если о сигнале заранее неизвестно абсолютно ничего, то его нельзя принять. Если о сигнале заранее известно все, то его не нужно принимать.

В окружающем нас мире существуют всевозможные сигналы различной формы и природы происхождения. Часть сигналов являются естественными, а часть сигналов создана человеком. Сигналы окружают нас повсюду. Они исходят от радиопередатчиков, телевизоров, смартфонов и радаров — это лишь малая часть источников. Оповещения смартфона, звуковые сигналы автомобилей, сообщения на табло вокзала, данные, передающиеся по высокоскоростным сетям. Некоторые сигналы обеспечивают нашу жизнедеятельность (речь, жесты или мимика человека), некоторые приносят удовольствие (музыка, фильмы), а некоторые нежелательны в какой-то конкретной ситуации. В контексте информационно-управляющих систем сигналы являются носителями информации от датчиков к вычислительной подсистеме, от вычислительной подсистемы к исполнительным устройствам, при сетевом взаимодействии вычислительных подсистем между собой. В электрической системе примерами таких сигналов могут быть напряжение, ток, количество заряда. В механической системе — координаты положения объекта, его скорость или масса. В финансовой системе сигналами могут являться цена акции, процентная ставка или обменный курс [2].

Один и тот же сигнал в зависимости от поставленной перед разработчиком задачи может нести полезную информацию, то есть быть *целевым*, или наоборот затруднять приём информации, то есть представлять собой *шум* (в случае естественного происхождения сигнала) или *помеху* (в случае искусственного происхождения сигнала) [2].

Цифровая обработка сигналов занимается вопросами получения информации, передаваемой сигналом на фоне помех.

Обобщенная схема цифровой обработки сигналов представлена на рисунке 1.

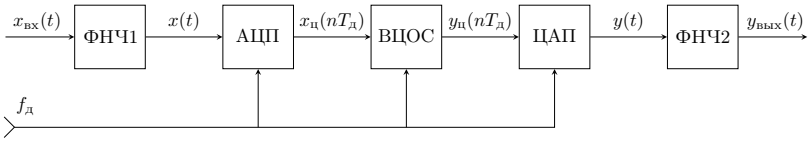


Рис. 1. Обобщенная схема цифровой обработки сигналов

Фильтр нижних частот (ФНЧ1) препятствует пропусканию гармоник с частотами выше половины частоты дискретизации, что обеспечивает выполнение условий теоремы Найквиста-Котельникова.

Аналого-цифровой преобразователь АЦП и цифро-аналоговый преобразователь ЦАП реализуют преобразование аналогового сигнала в цифровой и обратно соответственно.

Фильтр нижних частот ФНЧ2 обеспечивает сглаживание ступенчатого выходного сигнала ЦАП (рисунок 2).

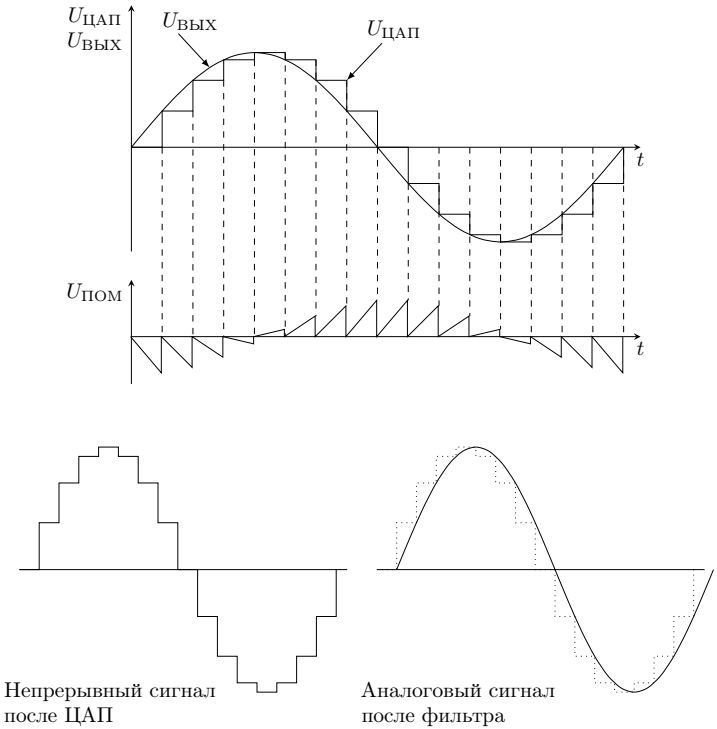


Рис. 2. Иллюстрация результатов работы сглаживающего фильтра на выходе цифро-аналогового преобразователя

Вычислитель цифровой обработки сигналов (ВЦОС) реализует непосредственно требуемый алгоритм цифровой обработки сигналов. Физически в качестве ВЦОС, как правило, используется либо процессор цифровой обработки сигналов со специализированной архитектурой (DSP – Digital Signal Processor), либо программируемая логическая интегральная схема (PLD - programmable logic device; FPGA – Field Program Grid Array), либо специализированная интегральная микросхема цифровой обработки сигналов.

В данном учебном издании предложено к практической реализации восемь алгоритмов цифровой обработки сигналов. Для простоты реализации реальная

программно-аппаратная система обработки сигналов, соответствующая схеме, изображенной на рисунке 1, заменена на персональный компьютер с установленным пакетом прикладных программ MathWorks® MATLAB®. В качестве алгоритма цифровой обработки сигнала используются алгоритмы цифровой фильтрации сигнала (лабораторная работа «Метод скользящего среднего», лабораторная работа «Фильтрация методом когерентного накопления», лабораторная работа «Нерекурсивный цифровой фильтр», лабораторная работа «Согласованная фильтрация»), преобразователь частоты (лабораторная работа «Цифровое гетеродинирование»), детектор (лабораторная работа «Алгоритм Герцеля»), кодек (лабораторная работа «Цифровой демодулятор фазоманипулированных сигналов»), преобразование сигнала (лабораторная работа «Преобразование Гильберта-Хуанга»).

Лабораторная работа №1

Метод скользящего среднего

Основные теоретические сведения

Цифровая фильтрация является одной из наиболее распространённых операций цифровой обработки сигналов, широко применяемых в науке и технике [1]. Фильтрация направлена на преобразование входного сигнала к заданному виду. Например, фильтр низких частот предназначен для пропускания только определенных гармоник сигнала, частота которых f_i удовлетворяет условию $f_i < f_{cp}$, где f_{cp} — заданная частота среза фильтра низких частот. Наиболее простой с точки зрения реализации фильтр низких частот реализуется с использованием метода скользящего среднего. Суть метода заключается в том, что для реализации сигнала производится последовательная замена каждого отсчёта сигнала на некоторое среднее значение M соседних точек. Данный метод эффективен для сглаживания сигнала и устранения высокочастотных шумов. При этом с увеличением длины окна M эффект сглаживания усиливается — все большее количество высокочастотных гармоник подвергается затуханию (рисунок 3).

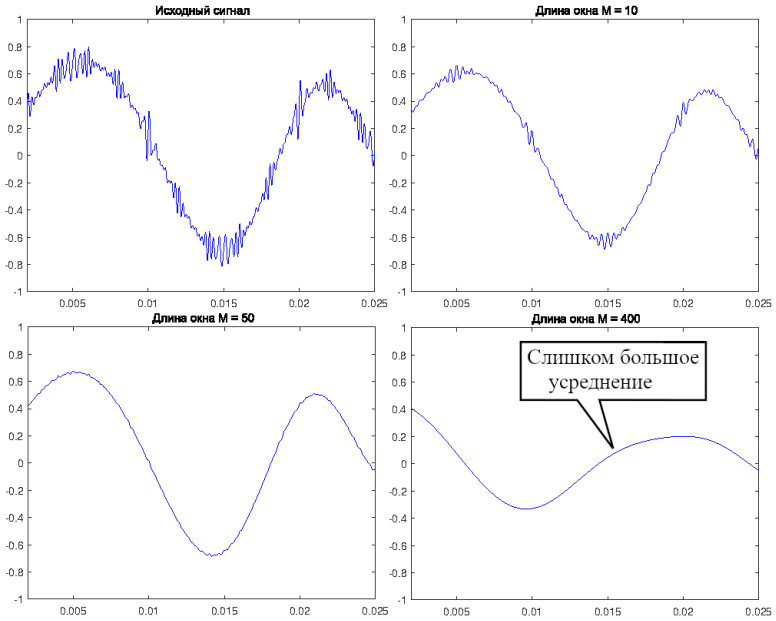


Рис. 3. Результаты фильтрации сигнала методом скользящего среднего

Метод скользящего среднего обладает рядом недостатков:

– скользящее среднее нельзя продлить на $M/2$ первых и последних отсчетов сигнала;
– метод не позволяет установить степень затухания для заграждаемых частот, а также сформулировать аналитическое выражение для частоты среза (другими словами не позволяет сформировать требуемую амплитудно-частотную характеристику фильтра).

Для метода простого скользящего среднего усреднение реализуется через обычное среднее арифметическое, а результат фильтрации описывается выражением

$$y(n) = \frac{1}{2m+1} \sum_{j=-m}^{m} x(n+j), \tag{1}$$

где $M = 2m + 1$ — длина окна фильтра; $x(n)$ — последовательность отсчетов входного сигнала; $y(n)$ — последовательность отсчетов выходного сигнала.

В том случае, если есть необходимость изменить вес определённых отсчётов при усреднении (например, увеличить вес более актуальных отсчётов), то применяется метод взвешенного скользящего среднего.

$$y(n) = \frac{\sum_{j=-m}^{m} x(n+j) \cdot \omega(n+j)}{\sum_{j=-m}^{m} \omega(n+j)}, \tag{2}$$

где $\omega(n+j)$ — дискретная функция весовых коэффициентов.

Выражение (2) является обобщением выражения (1) и в случае $w(n) = 1$ взвешенное скользящее среднее преобразуется в простое скользящее среднее. В зависимости от вида весовой функции $w(n)$ метод взвешенного скользящего среднего также может иметь различные частные случаи. Наиболее распространенными являются метод линейного скользящего среднего и метод экспоненциального скользящего среднего. Также, кроме среднего арифметического усреднения может применяться среднее квадратическое, среднее геометрическое и другие виды усреднений.

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанной речью или мелодией длительностью $8 - 15$ с.
2. Построить графики сигнала во временной и частотной области.
3. Добавить высокочастотную шумовую составляющую с прямоугольной функцией спектральной плотности от 10 до 15 кГц и мощностью сопоставимой с

мощностью звукового сигнала.

4. Построить графики сигнала во временной и частотной области.
5. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.
6. Провести фильтрацию сигнала методом скользящего среднего окном длительностью M.
7. Построить графики сигнала во временной и частотной области.
8. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

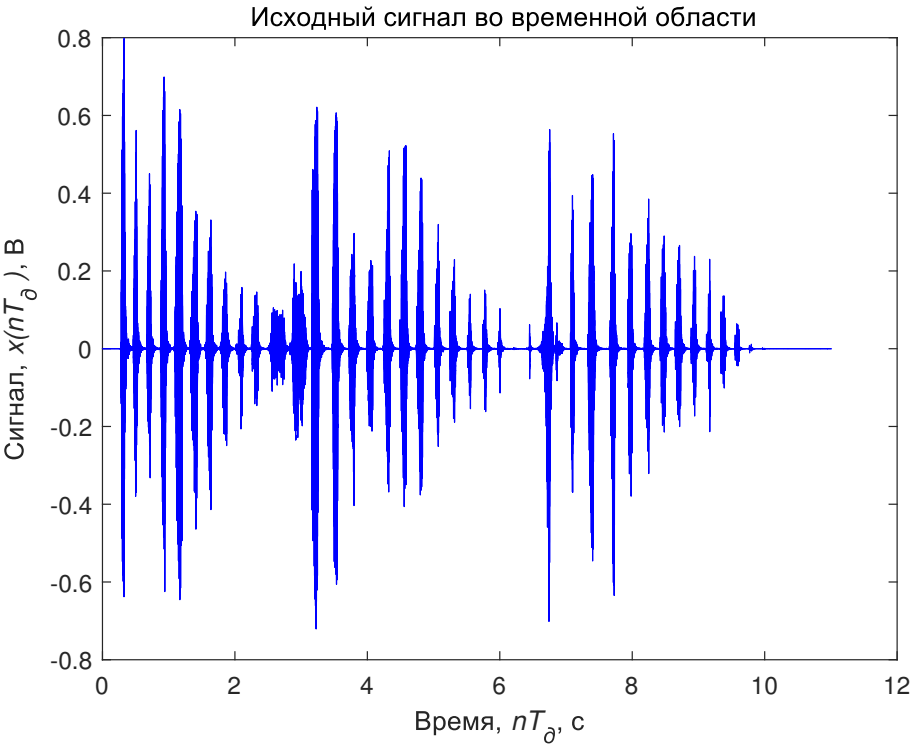
1 Фильтрация сигнала методом скользящего среднего

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
A = 0.006; % Амплитуда шума
f0 = 5000; fn = 10000; df = 500; % Частота шума, Гц
w = 40; % Размер окна фильтра
```

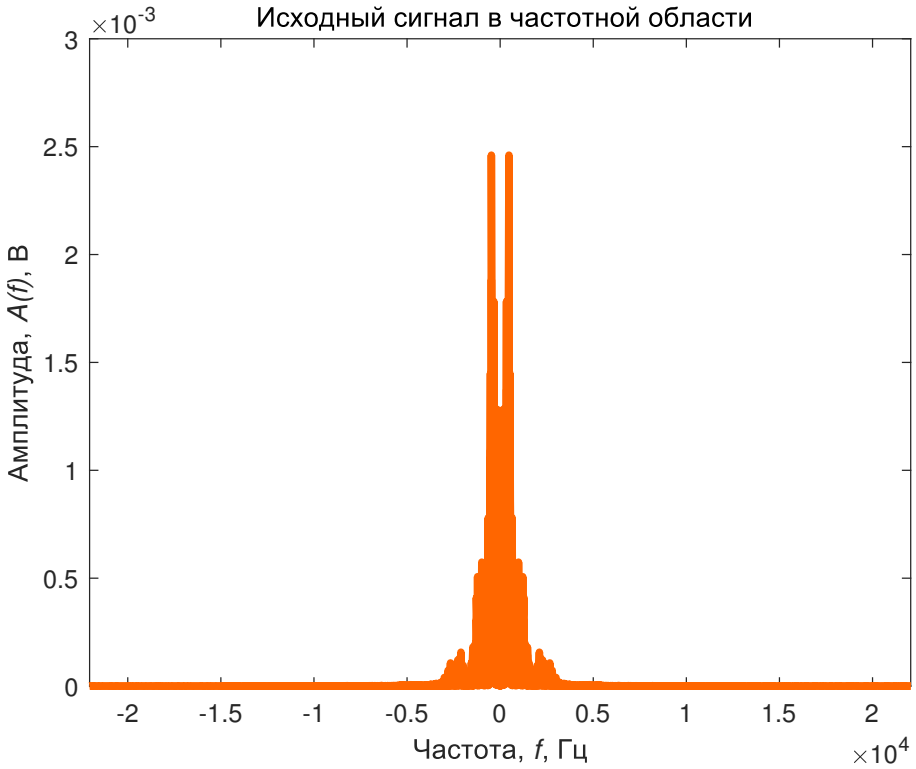
1.2 Чтение звукового файла

```
[data,rate] = audioread('sound.wav'); % Чтение отсчётов и частоты дискретизации
t = linspace(0, length(data)/rate, length(data)); % Формирование области определ
ения
figure; plot(t, data, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



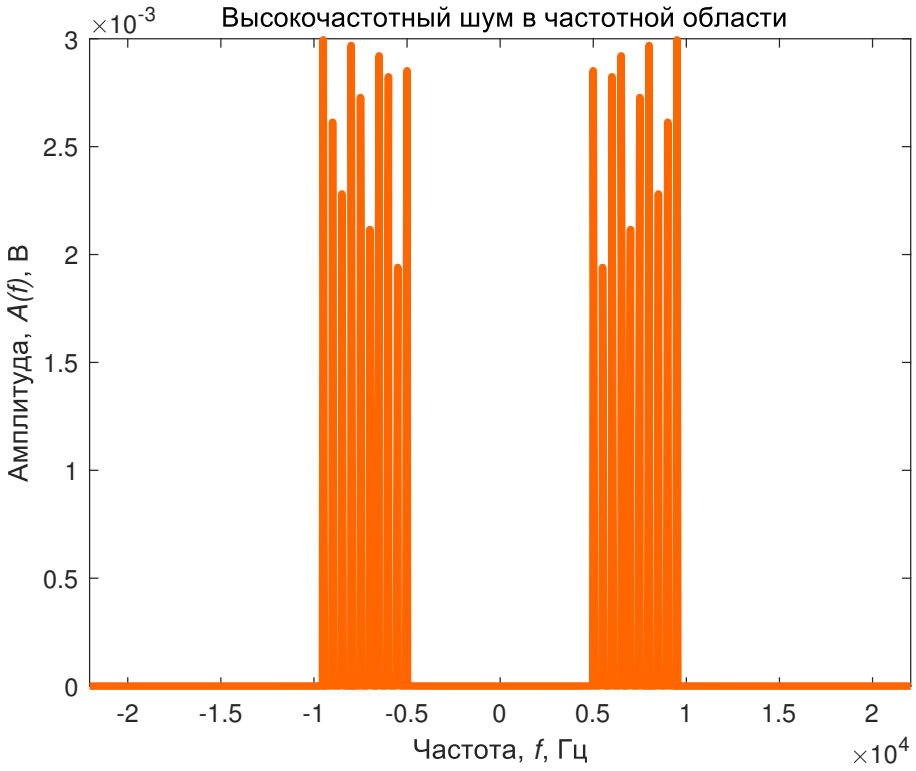
1.3 Построение амплитудного спектра исходного сигнала

```
f = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
axis([-rate/2 rate/2 0 A/2]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



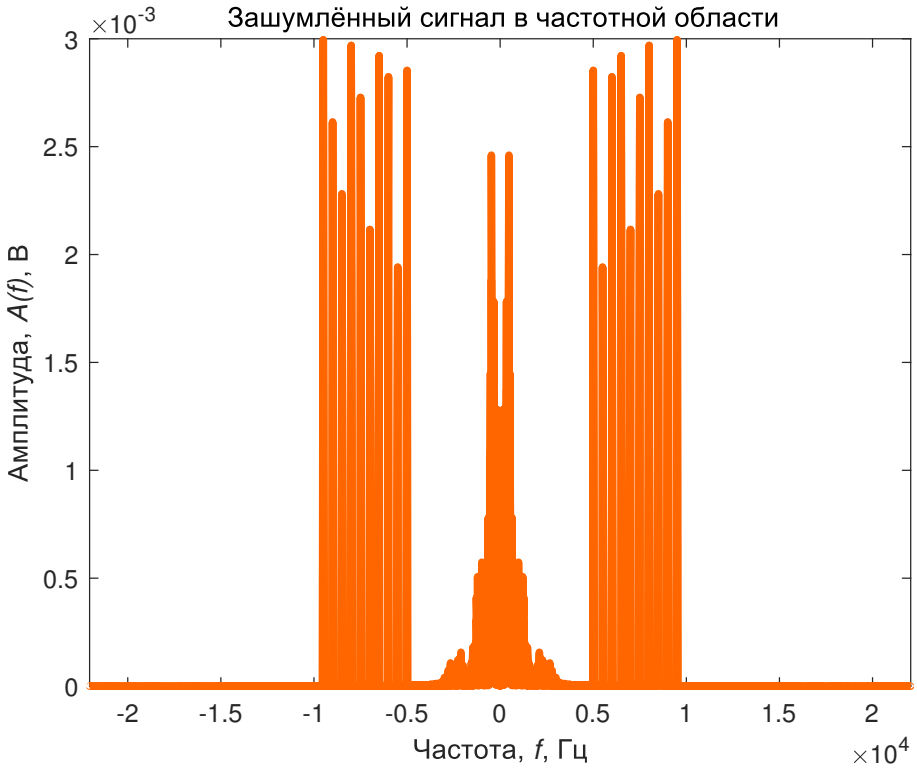
1.4 Формирование высокочастотного шума

```
ny=zeros(length(data),1); % Массив отсчётов шума
while f0 < fn
    ny=ny+A.*sin(2*pi*f0*t.');
    f0=f0+df;
end
fny = abs(fft(ny)/length(ny)); % Массив отсчётов спектра шума
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fny),...
    'Color', fColor, 'LineWidth', 3);
axis([-rate/2 rate/2 0 A/2]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Высокочастотный шум в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



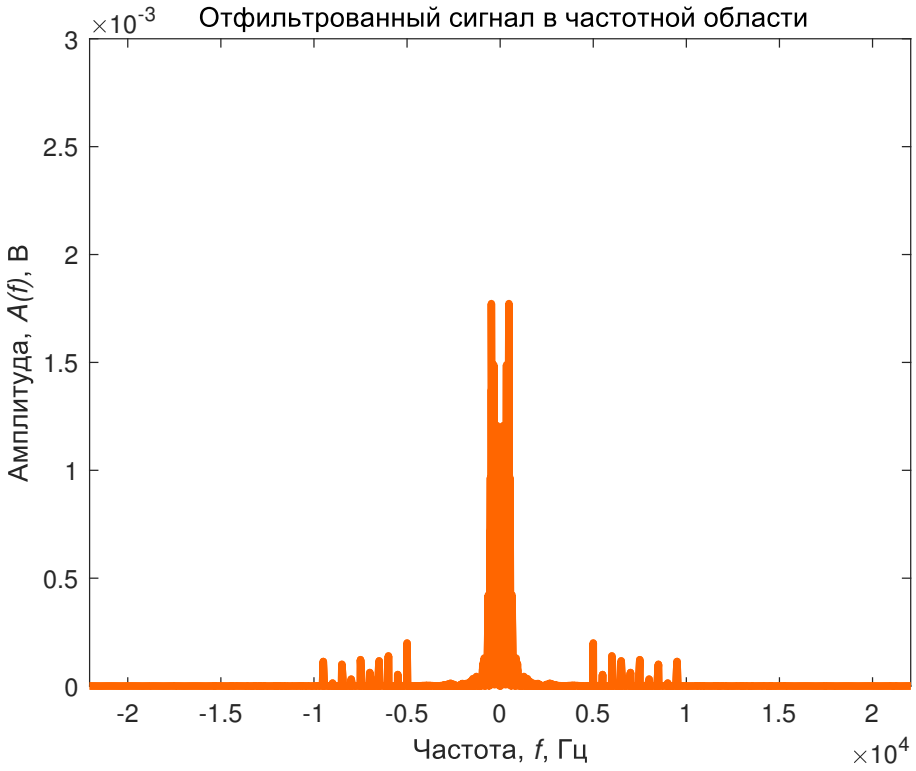
1.5 Формирование зашумлённого сигнала

```
ndata = data; % Массив отсчётов зашумлённого сигнала стереоканала
ndata(:,1) = data(:,1) + ny; % Добавление шума к исходному сигналу
ndata(:,2) = data(:,2) + ny;
% Запись звукового файла с зашумлённым сигналом
audiowrite('noise.wav', ndata, rate);
% Массив отсчётов спектра зашумлённого сигнала
fdata = abs(fft(ndata)/length(ndata));
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
axis([-rate/2 rate/2 0 A/2]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Зашумлённый сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.6 Фильтрация сигнала методом скользящего среднего

```
madata = ndata; % Массив отсчётов отфильтрованного сигнала стереоканала
% Фильтрация методом скользящего среднего с размером окна w
madata(:,1) = cumsum(ndata(:,1));
madata(:,2) = cumsum(ndata(:,2));
madata(w:end) = madata(w:end) - madata(1:end-w+1);
madata(w-1:end) = madata(w-1:end)./w;
% Запись звукового файла с отфильтрованным сигналом
audiowrite('filter.wav', madata, rate);
% Массив отсчётов спектра отфильтрованного сигнала
fdata = abs(fft(madata)/length(madata));
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
axis([-rate/2 rate/2 0 A/2]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```


Отфильтрованный сигнал в частотной области
×10-3
3
2.5
2
1.5
1
0.5
0
Амплитуда, A(f), В
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Частота, f, Гц
×10^4

Контрольные вопросы

1. Какие достоинства и недостатки цифрового фильтра, реализованного методом скользящего среднего Вы можете назвать?
2. Какую область частот пропускает фильтр, реализованный с помощью метода скользящего среднего?
3. Какие параметры фильтра, реализованного методом скользящего среднего Вы знаете?
4. Каким образом параметры фильтра, реализованного методом скользящего среднего, влияют на его АЧХ?
5. Какие разновидности метода скользящего среднего Вы знаете? Чем они отличаются?

Лабораторная работа №2

Фильтрация методом когерентного накопления

Основные теоретические сведения

Накопление импульсов — это метод улучшения вероятности обнаружения целевых параметров сигнала за счет использования энергии нескольких зондирующих импульсов (рисунок 4) [3]. Наибольшее применение метод находит в локации [4].

При когерентном накоплении сигнала выполняются следующие операции:

– коррекция доплеровского набега фазы сигнала за период повторения;
– совмещение во времени одиночных сигналов;
– синфазное (когерентное) сложение N сигналов на всем интервале наблюдения.

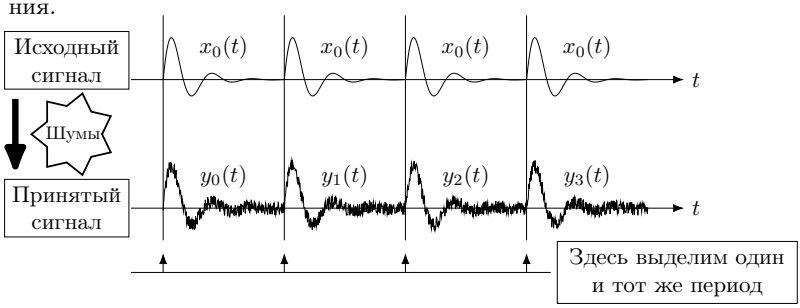


Рис. 4. Иллюстрация использования нескольких зондирующих импульсов для когерентного накопления

С учётом наложения шумов на зондирующий сигнал $x_0(n)$ принимаемый j-й импульс сигнала определяется выражением

$$y_j(n) = x_0(n) + x_{nj}(n),$$

где $x_{nj}(n)$ — реализация шума, наложенная на j-й зондирующий импульс.

Тогда нормированный результат когерентного сложения определяется выражением

$$y(n) = \frac{1}{N}\sum_{j=1}^{N} y_j(n) = \frac{1}{N}\sum_{j=1}^{N}(x_0(n) + x_{nj}(n)) =$$

$$= \frac{1}{N}\sum_{j=1}^{N} x_0(n) + \frac{1}{N}\sum_{j=1}^{N} x_{nj}(n) = x_0(n) + \frac{1}{N}\sum_{j=1}^{N} x_{nj}(n).$$

В том случае, если шумы являются некоррелированным стационарным случайным процессом, то выражение

$$\lim_{N\to\infty} \frac{1}{N} \sum_{j=1}^{N} x_{nj}(n)$$

стремится к 0, а следовательно, с увеличением количества зондирующих импульсов N выходной сигнал стремится к зондирующему сигналу $x_0(n)$.

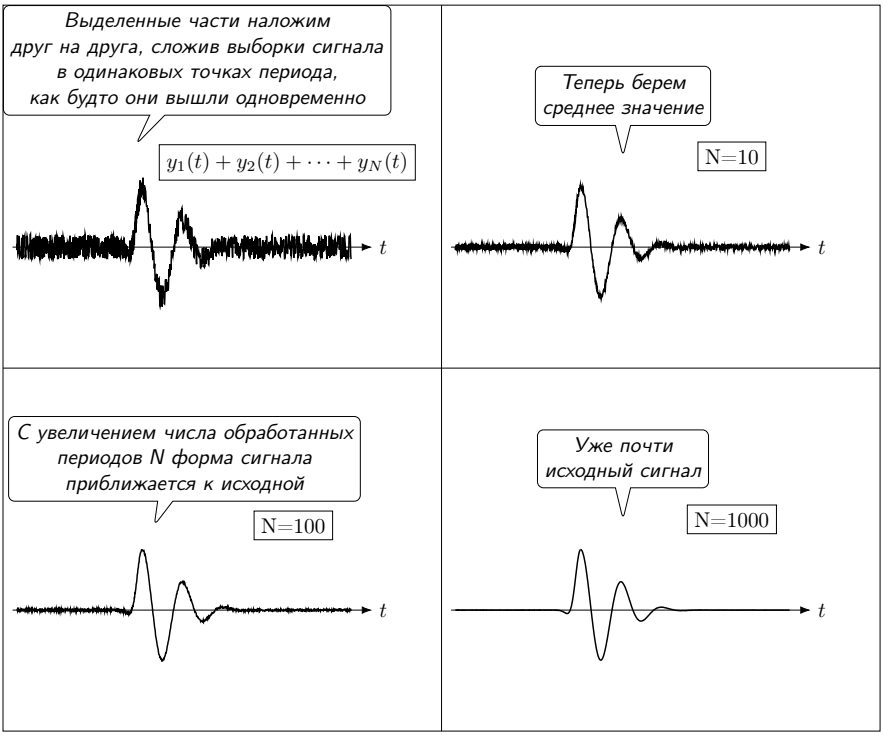


Рис. 5. Результаты фильтрации сигнала методом когерентного накопления

Когерентное накопление является линейной операцией цифровой обработки сигналов, поэтому критерием эффективности метода фильтрации когерентного накопления является увеличение соотношения сигнал/шум, обеспечиваемое накопителем.

Амплитуда когерентно суммируемых сигналов увеличивается при фильтрации данным методом в N раз, следовательно мощность — в N^2 раз. Мощность шума, у которого междупериодная корреляция отсутствует, в результате накоп-

ления увеличивается в N раз (в соответствии со свойством дисперсии суммы независимых случайных величин). В итоге отношение сигнал/шум по мощности возрастает пропорционально числу накапливаемых сигналов N (рисунок 5) [3, 5].

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанной речью или мелодией длительностью $8 - 15$ с.
2. Построить графики звукового сигнала во временной и частотной области.
3. Добавить нормальный шум мощностью сопоставимой с мощностью звукового сигнала.
4. Построить графики сигнала во временной и частотной области.
5. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.
6. Провести фильтрацию сигнала методом когерентного накопления для количества реализаций N.
7. Построить графики сигнала во временной и частотной области.
8. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

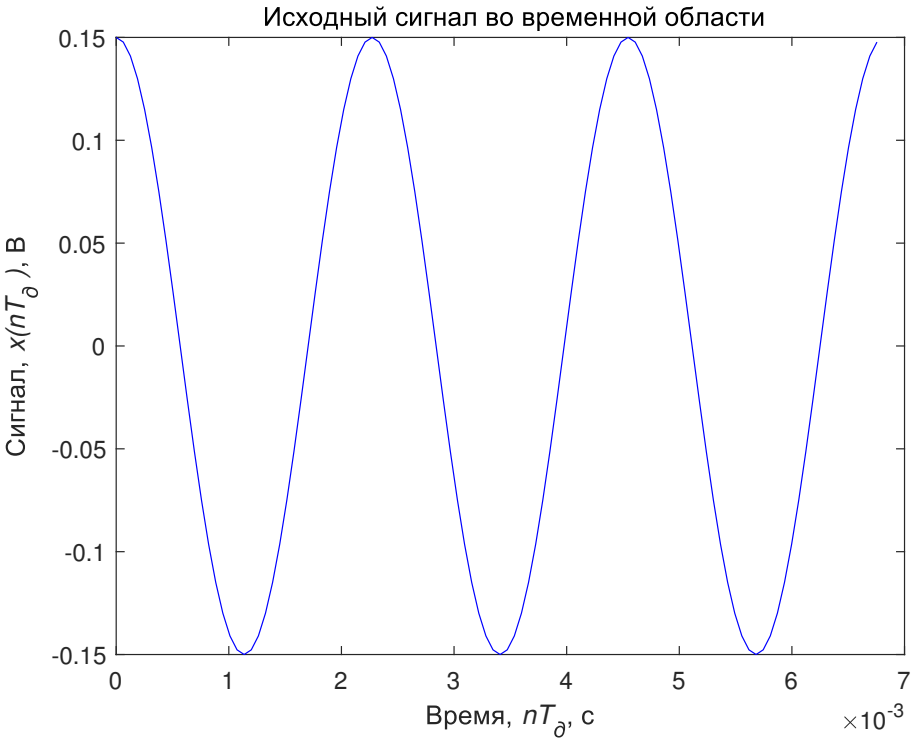
1 Фильтрация сигнала методом когерентного накопления

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
rColor = [0.2 0.7 0.3]; % Цвет графиков отфильтрованного сигнала
volume = 0.15; % Амплитуда исходного сигнала
f0 = 440; % Частота сигнала, Гц
fd = 16000; % Частота дискретизации, Гц
mean = 0; % Математическое ожидание шума
std = 0.3; % Стандартное отклонение
nK = 100; % Количество накоплений
nT = 20; % Количество периодов отфильтрованного сигнала
dt0 = nT/f0; % Длительность отфильтрованного сигнала, с
N0 = nT*round(fd/f0); % Количество отсчетов отфильтрованного сигнала
dt = nK*dt0; % Длительность зашумлённого сигнала, с
N = N0*nK; % Общее количество отсчетов сигнала
gT = 3; % Количество периодов для визуализации
```

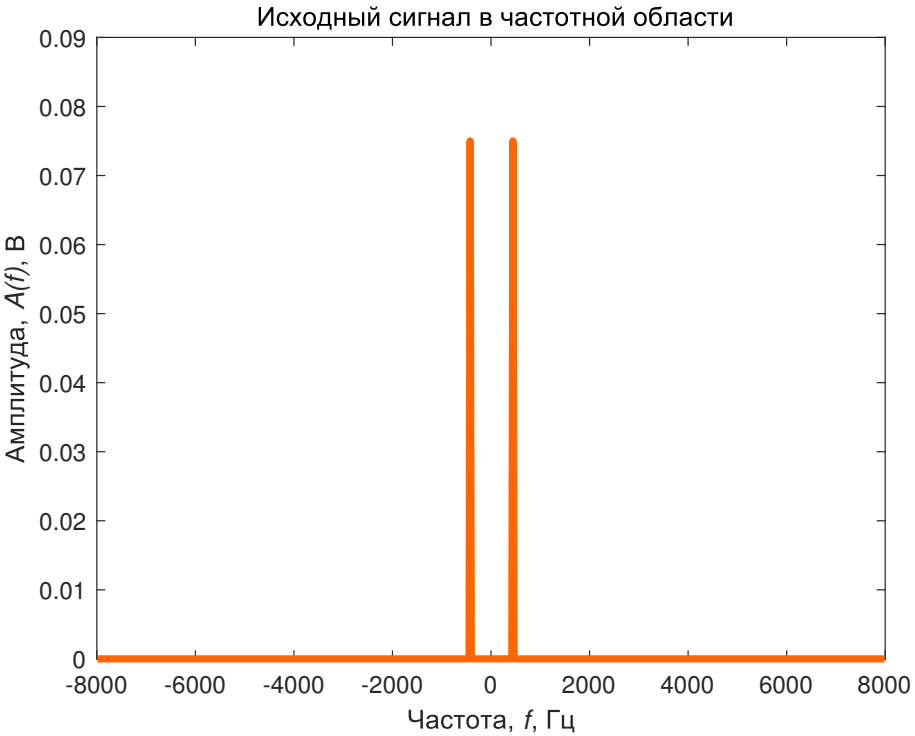
1.2 Моделирование тонального сигнала

```
t = linspace(0,dt,N); % Формирование области определения во временной области
data = volume*cos(2*pi*f0*t); % Формирование тонального сигнала
% Запись звукового файла с исходным сигналом
audiowrite('sound.wav', data, fd);
figure; plot(t(1:gT*round(fd/f0)), data(1:gT*round(fd/f0)), 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



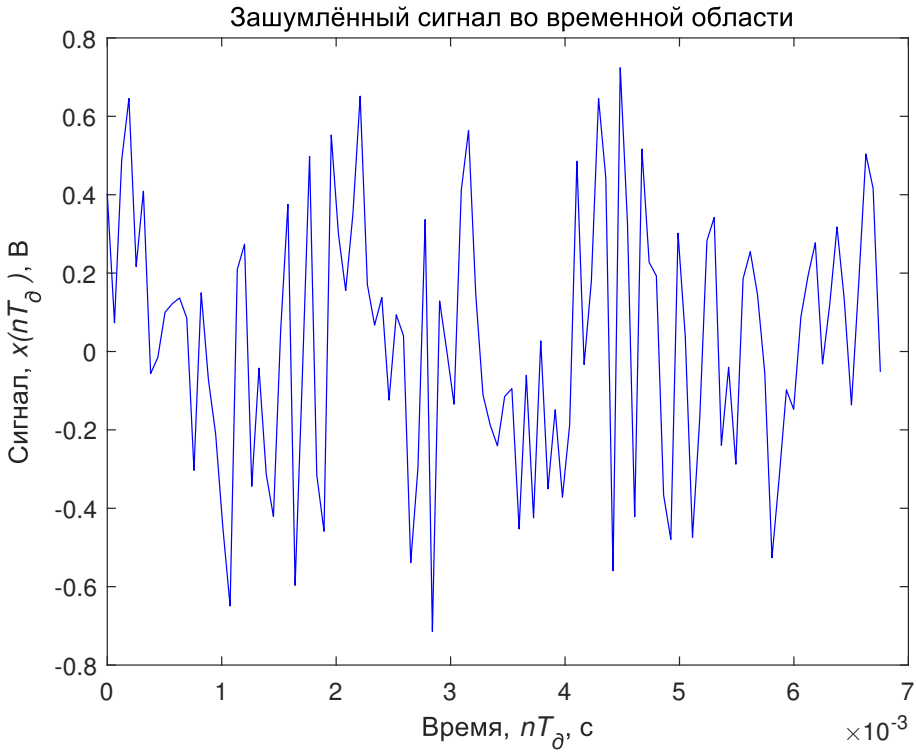
1.3 Построение амплитудного спектра тонального сигнала

```
f = linspace(0, fd, N0); % Формирование области определения в частотной области
% Массив отсчётов спектра исходного сигнала
fdata = abs(fft(data(1:N0))/N0);
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
axis([-fd/2 fd/2 0 volume*0.6]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



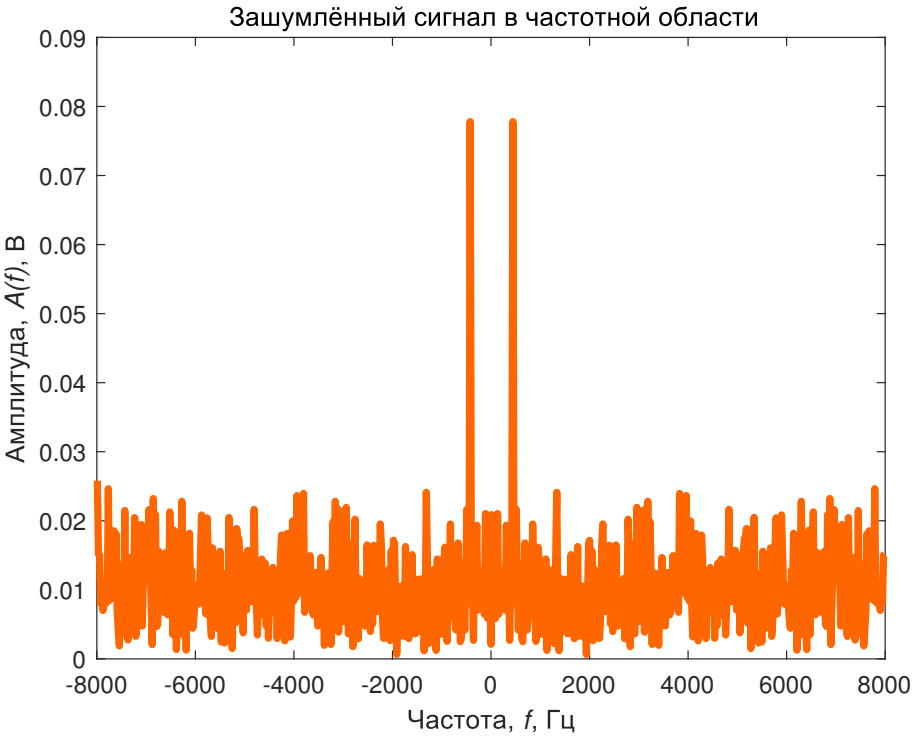
1.4 Формирование зашумлённого сигнала

```
ny = normrnd(mean, std, [1,N]); % Массив отсчётов шума по норм. распределению
ndata = data + ny; % Добавление шума к исходному сигналу
% Запись звукового файла с зашумлённым сигналом
audiowrite('noise.wav', ndata, fd);
mdata = reshape(ndata, [N0,nK]).'; % Разделение сигнала на nK накоплений
figure; plot(t(1:gT*round(fd/f0)), mdata(1,1:gT*round(fd/f0)), 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Зашумлённый сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



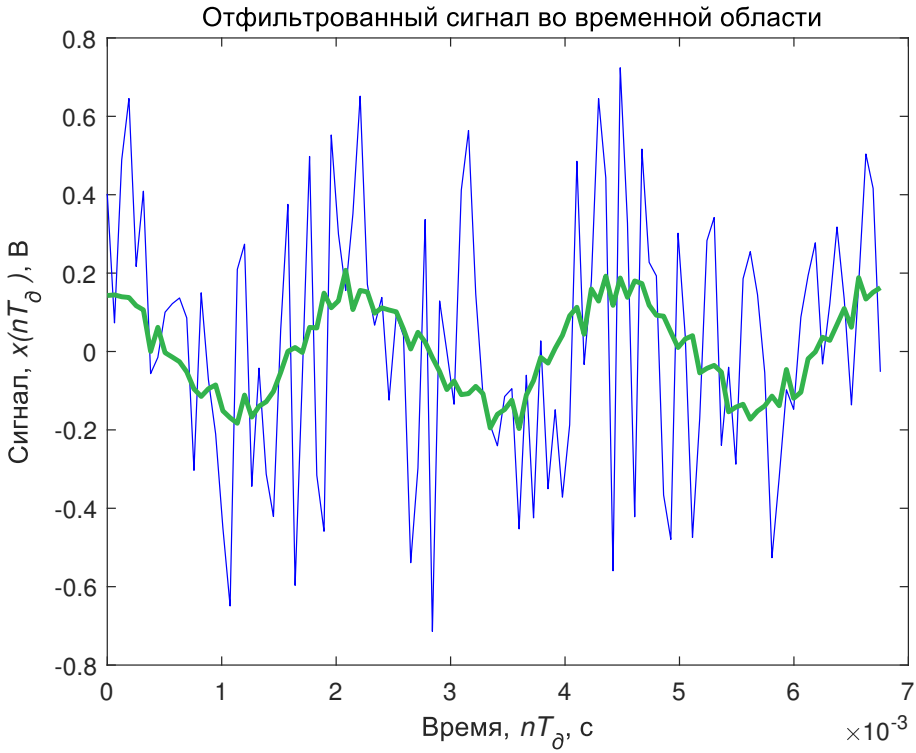
1.5 Построение амплитудного спектра зашумлённого сигнала

```
% Массив отсчётов спектра зашумлённого сигнала
fmdata = abs(fft(mdata(1,:))/N0);
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fmdata),...
    'Color', fColor, 'LineWidth', 3);
axis([-fd/2 fd/2 0 volume*0.6]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Зашумлённый сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



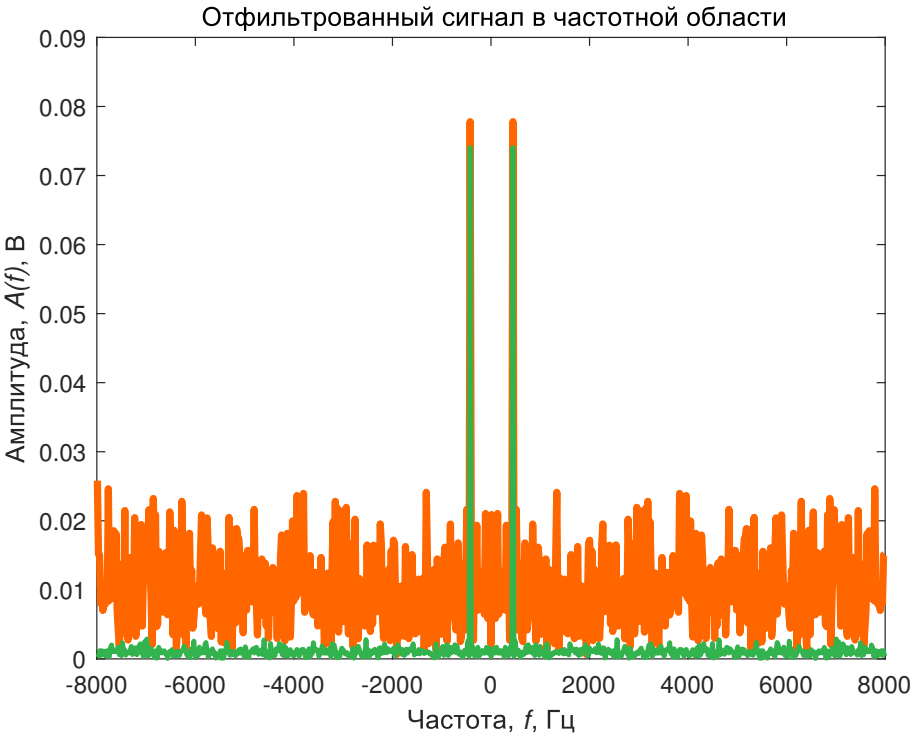
1.6 Фильтрация методом когерентного накопления

```
% Фильтрация методом когерентного накопления с количеством накоплений nK
sdata = sum(mdata)./nK;
% Запись звукового файла с отфильтрованным сигналом
audiowrite('filter.wav', sdata, fd);
% Построение графика с зашумлённым и отфильтрованным сигналами
figure; plot(t(1:gT*round(fd/f0)), mdata(1,1:gT*round(fd/f0)), 'Color', tColor);
 hold on;
plot(t(1:gT*round(fd/f0)), sdata(1:gT*round(fd/f0)), 'Color', rColor, 'LineWidth
', 2);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



1.7 Построение амплитудных спектров зашумлённого и отфильтрованного сигналов

```
% Массив отсчётов спектра отфильтрованного сигнала
fsdata = abs(fft(sdata)/N0); % Формирование значений
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fmdata),...
    'Color', fColor, 'LineWidth', 3); hold on;
plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fsdata),...
    'Color', rColor, 'LineWidth', 2);
axis([-fd/2 fd/2 0 volume*0.6]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```


Отфильтрованный сигнал в частотной области
Амплитуда, A(f), В
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000
Частота, f, Гц

Контрольные вопросы

1. Назовите ограничения применения метода синхронной фильтрации (когерентного накопления).
2. Какой параметр метода синхронной фильтрации (когерентного накопления) определяет качество результирующего сигнала?
3. Приведите пример прикладного использования метода синхронной фильтрации (когерентного накопления).
4. Для какого типа шума эффективно применение метода синхронной фильтрации (когерентного накопления)?
5. Попробуйте кратко пояснить суть метода синхронной фильтрации (когерентного накопления), используя термины «детерминированный сигналы» и «случайный сигналы».

Лабораторная работа №3

Нерекурсивный цифровой фильтр

Основные теоретические сведения

Ограничения фильтрации методом скользящего среднего, связанные с невозможностью априорного задания параметров амплитудно-частотной характеристики, побуждают в ряде случаев применять другие методы проектирования цифровых фильтров.

В общем случае цифровой фильтр представляет собой линейную дискретную систему (рисунок 6) с входным сигналом $x(n)$ и выходным сигналом $y(n)$, которая описывается уравнением

$$\sum_{m=0}^{M} a_m y(n-m) = \sum_{k=0}^{K} b_k x(n-k), \tag{3}$$

где a_m, b_k — некоторые весовые коэффициенты, соответствующие отсчетам сигналов с номерами m и k соответственно.

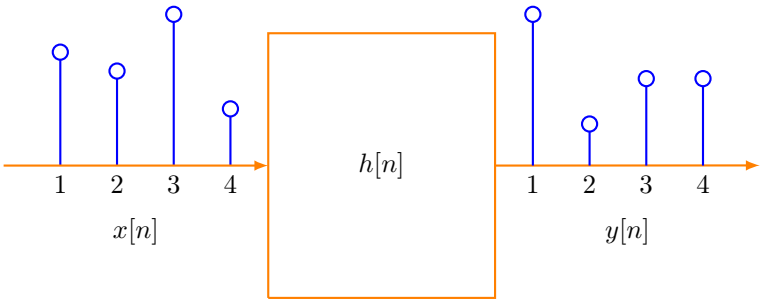


Рис. 6. Структурная схема дискретной системы

Используя (3), получим выражение для текущего значения (с нулевой задержкой) выходного сигнала

$$y(n) = \sum_{k=0}^{K} \frac{b_k}{a_0} x(n-k) - \sum_{m=1}^{M} \frac{a_m}{a_0} y(n-m). \tag{4}$$

Таким образом, мы получили, что текущее значение выходного сигнала цифрового фильтра в общем случае определяется линейной комбинацией текущего входного значения сигнала, предыдущих отсчётов входного сигнала и предыдущих отсчётов выходного сигнала. Такой фильтр называется рекурсивным, так выходной сигнал зависит не только от входного, но и от предыдущих значений выходного сигнала.

В том случае, если $\forall m > 0 : a_m = 0$, то выражение (4) принимает вид

$$y(n) = \sum_{k=0}^{K} \frac{b_k}{a_0} x(n-k). \tag{5}$$

Фильтр, описываемый выражением (5), называется нерекурсивным. Такое название подчеркивает, что выходной сигнал такого фильтра зависит только от отсчётов входного сигнала и не зависит от отсчётов выходного сигнала, а $h(n)$ представляет собой импульсную характеристкику нерекурсивного цифрового фильтра.

Используя понятие и обозначение свертки математического анализа [6], выражение (5) можно переписать в виде

$$y(n) = h(n) * x(n).$$

Операция свертки является коммутативной, поэтому

$$h(n) * x(n) = x(n) * h(n),\ a$$

$$\sum_{k=0}^{K} h(k)x(n-k) = \sum_{k=0}^{K} h(n-k)x(k).$$

Импульсная характеристика фильтра — это отклик фильтра при нулевых начальных условиях в виде выходной последовательности во временной области при подаче на вход единичного импульса.

Комплексной частотной характеристикой (КЧХ) $H_д(w)$ цифрового фильтра называется зависимость от частоты отношения комплексной амплитуды выходного дискретного гармонического сигнала к комплексной амплитуде входного дискретного гармонического сигнала в стационарном режиме. КЧХ определена выражением

$$H_д(w) = \frac{\dot{Y}}{\dot{X}}$$

и, как видно, является спектральной функцией для импульсивной характеристики $h_д(t)$.

Амплитудно-частотной характеристикой (АЧХ) $|H_д(w)|$ цифрового фильтра называется зависимость от частоты отношения амплитуды выходного дискретного гармонического сигнала к амплитуде входного дискретного гармонического сигнала в стационарном режиме:

$$|H_д(w)| = \frac{Y}{X}.$$

Фазочастотной характеристикой (ФЧХ) $\varphi_{H_д}(w)$ цифрового фильтра называется зависимость от частоты разности начальных фаз выходного и входного дискретных гармонических сигналов в стационарном режиме:

$$\varphi_{H_д}(w) = \varphi_Y(w) - \varphi_X(w) = arg H_д(w).$$

Частотные характеристики цифрового фильтра являются переодическими функциями частоты с периодом $w_д$, однако, их чаще всего бывает удобно рассматривать как 2π-переодические функции нормированной частоты wT [7].

Примеры частотных характеристик приведены на рисунке 7.

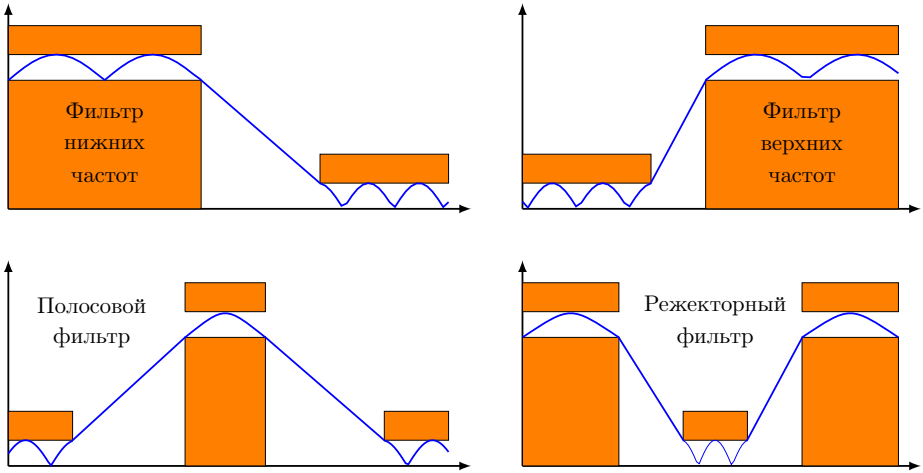


Рис. 7. Основные типы амплитудно-частотных характеристик цифровых фильтров

Нерекурсивный фильтр в отличие от рекурсивного:

1. Имеет конечную импульсную характеристику.
2. Имеет линейную фазо-частотную характеристику.
3. Является устойчивым.

Линейная фазо-частотная характеристика обеспечивает равномерную задержку всех гармоник сигнала вне зависимости от значения их частоты.

Фильтр называется *устойчивым*, если при любых конечных начальных условиях и любом ограниченном входном сигнале выходной сигнал также остается ограниченным.

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанной речью или мелодией длительностью $8 - 15$ с.
2. Построить графики звукового сигнала во временной и частотной области.

3. Добавить высокочастотную шумовую составляющую с прямоугольной функцией спектральной плотности от 10 до 15 кГц и мощностью сопоставимой с мощностью звукового сигнала.
4. Построить графики сигнала во временной и частотной области.
5. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.
6. Спроектировать цифровой фильтр с конечной импульсной характеристикой.
7. Провести фильтрацию звукового сигнала с помощью полученного цифрового фильтра с конечной импульсной характерстикой.
8. Построить графики сигнала во временной и частотной области.
9. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

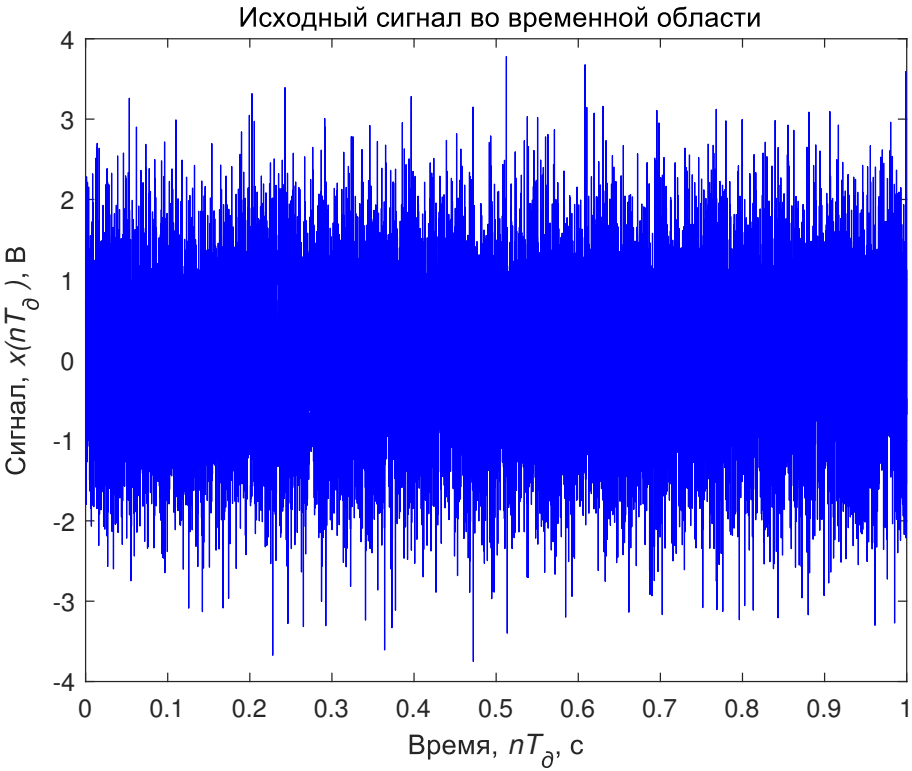
1.1 Инициализация и формирование значений основных параметров

1 Применение фильтра с конечной импульсной характеристикой

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
fd = 16000; % Частота дискретизации, Гц
dt = 1; % Длительность исходного сигнала, с
N = 16000; % Количество отсчётов исходного сигнала
```

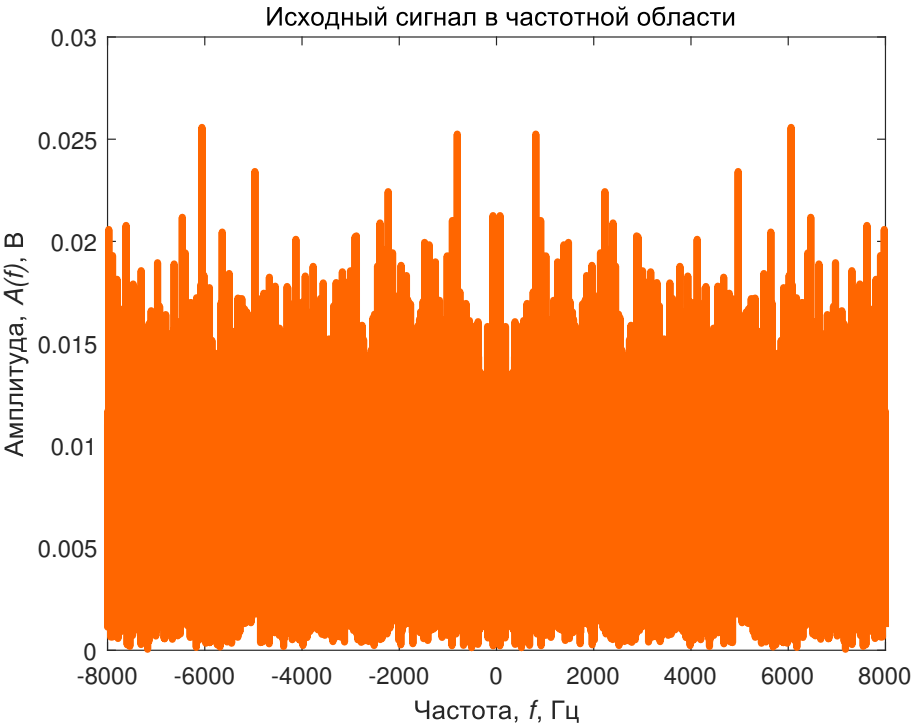
1.2 Моделирование белого шума в качестве исходного сигнала

```
t = linspace(0,dt,N); % Формирование области определения во временной области
noise = wgn(1,N,0); % Массив отсчётов белого шума - исходного сигнала
figure; plot(t, noise, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```

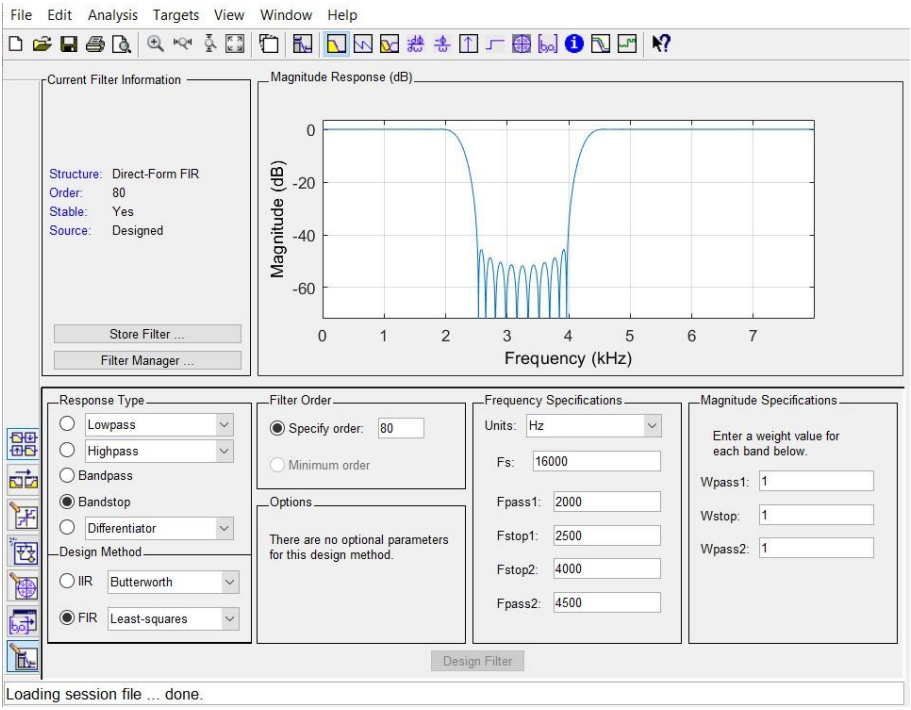


1.3 Построение амплитудного спектра белого шума

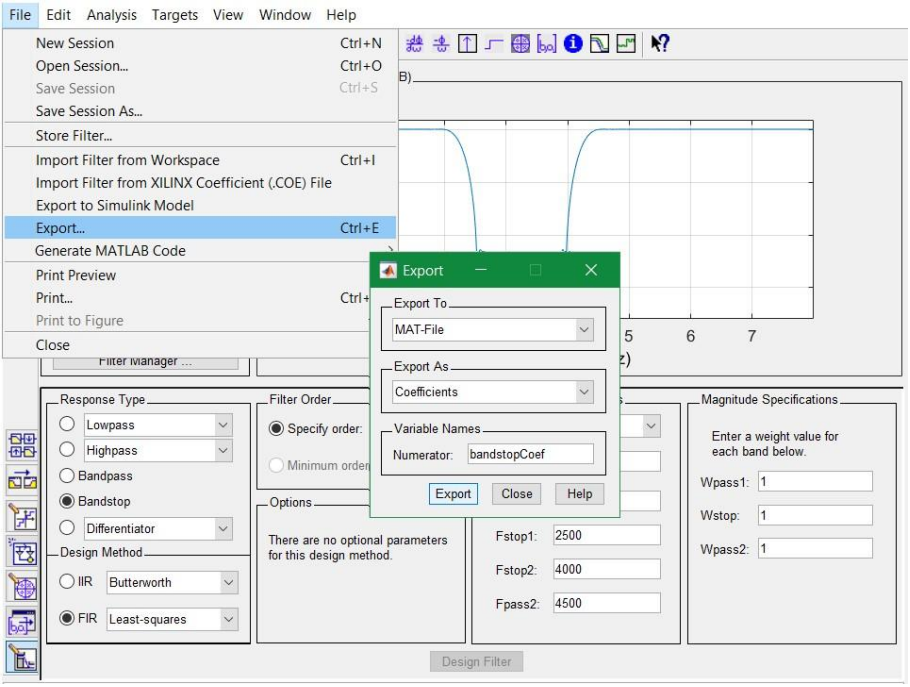
```
f = linspace(0, fd, N); % Формирование области определения в частотной области
% Массив отсчётов спектра исходного сигнала
fnoise = abs(fft(noise)/N);
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fnoise),...
    'Color', fColor, 'LineWidth', 3);
%axis([-fd/2 fd/2 0 volume*0.6]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.4 Создание режекторного фильтра в Matlab Filter Designer

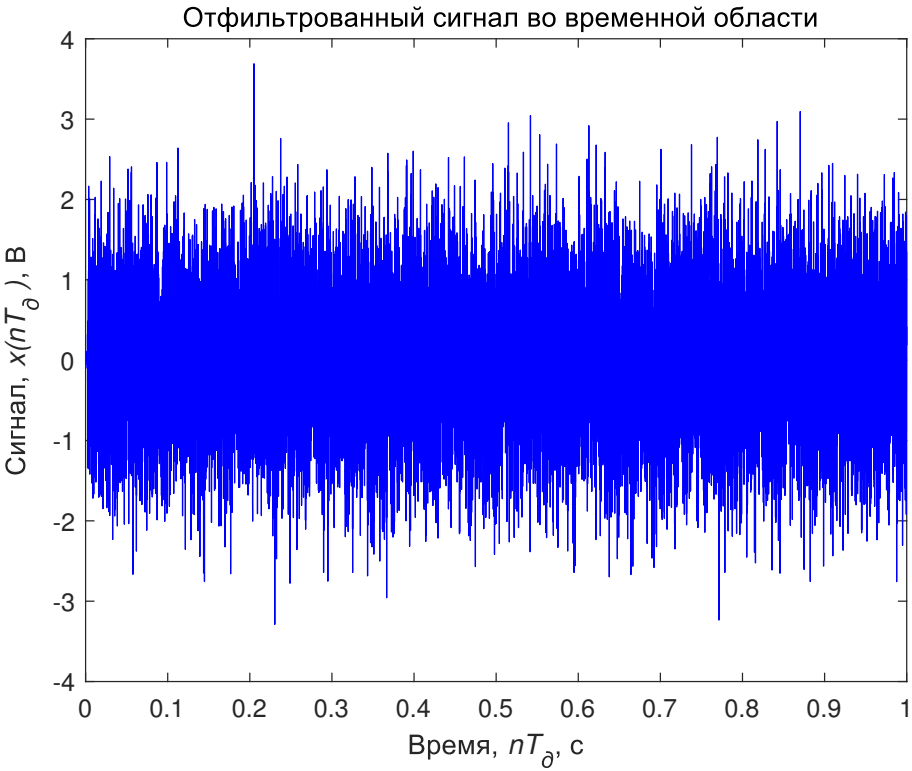


1.5 Экспорт коэффициентов фильтра в файл



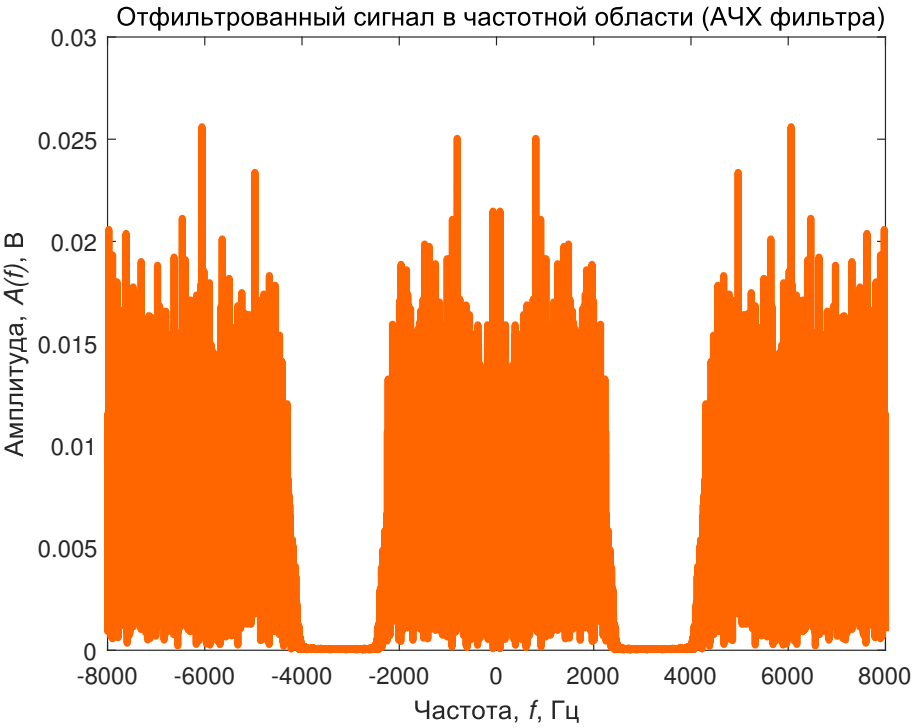
1.6 Фильтрация белого шума

```
load('bandstopFilter.mat'); % Загрузка файла с коэффициентами фильтра
global bandstopCoef; % Массив коэффициентов КИХ-фильтра
data = filter(bandstopCoef, 1, noise); % Применение фильтра к белому шуму
figure; plot(t, data, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



1.7 Построение амплитудного спектра отфильтрованного сигнала

```
% Массив отсчётов спектра отфильтрованного сигнала
fdata = abs(fft(data)/N);
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
%axis([-fd/2 fd/2 0 volume*0.6]); % Диапазон значений осей
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал в частотной области (АЧХ фильтра)'); % Заголов
ок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```


Отфильтрованный сигнал в частотной области (АЧХ фильтра)
Амплитуда, A(f), В
Частота, f, Гц
0.03
0.025
0.02
0.015
0.01
0.005
0
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000

Контрольные вопросы

1. Какими преимуществами и недостатками обладает цифровой фильтр по сравнению с аналоговым?
2. Какие виды фильтров по пропускаемой ими полосе частот вы можете назвать?
3. Какие параметры АЧХ фильтра нужно задать при проектировании?
4. Чем рекурсивный фильтр отличается от нерекурсивного?
5. Что такое импульсная характеристика фильтра?
6. Каким соотношением связаны импульсная и частотная характеристики фильтров?
7. Что такое устойчивость цифрового фильтра? Какие виды цифровых фильтров являются устойчивыми?
8. О чем говорит линейность фазо-частотной характеристики цифрового фильтра?
9. Каким методом можно реализовать цифровой фильтр без задержки выходного сигнала относительно входного?
10. Какая математическая операция позволяет получить выходной сигнал фильтра при определенном входном сигнале и известной импульсной характеристики фильтра?

Лабораторная работа №4

Согласованная фильтрация

Основные теоретические сведения

Напомним, что сигнал на выходе цифрового фильтра определяется выражением

$$y(n) = \sum_{k=0}^{K} h(k) \cdot x(n-k) = \sum_{k=0}^{K} h(n-k) \cdot x(k). \tag{6}$$

Если предположить, что импульсная характеристика определяется выражением

$$h(k) = x(K-k),$$

то есть является зеркальной копией сигнала, то выражение (6) примет вид

$$y(n) = \sum_{k=0}^{K} x(K-k) \cdot x(n-k).$$

В результате такой настройки фильтра на сигнал отклик получился пропорциональным автокорреляционной функции (АКФ) сигнала $x(n)$

$$R(n) = \sum_{k=0}^{K} x(n) \cdot x(n-k).$$

Рисунок 8 иллюстрирует вычисление отсчетов АКФ опорного сигнала $x(n)$ в различные моменты времени.

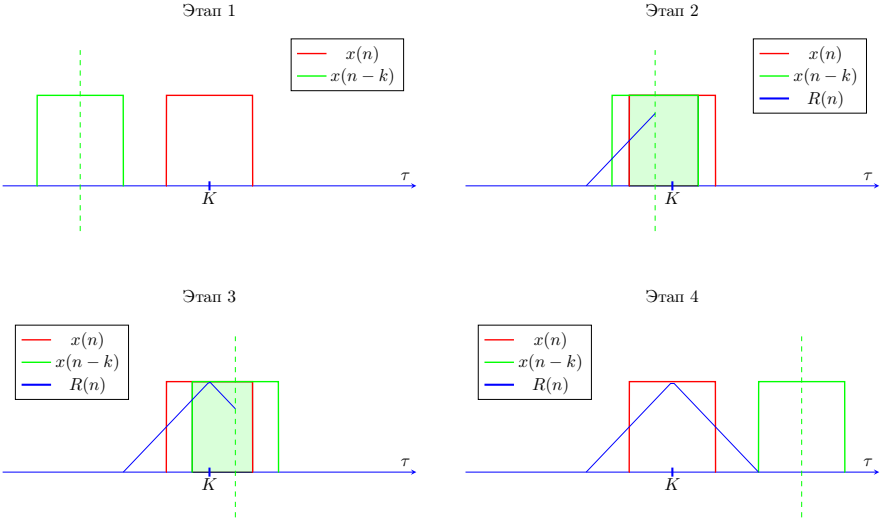


Рис. 8. Иллюстрация вычисления отсчетов АКФ опорного сигнала $x(n)$ в различные моменты времени

В момент времени $n = K$, когда сигнал совпадает с импульсной характеристикой (опорным сигналом), формируется максимум, соответствующий максимуму АКФ

$$y(K) = \sum_{k=0}^{K} x^2(k) = R_{max}.$$

Таким образом, под согласованным фильтром понимается фильтр, импульсная характеристика которого согласована с опорным сигналом. Результатом работы согласованного фильтра является накопление информации о наличии и временной задержки опорного сигнала на входе фильтра.

Практическое применение согласованная фильтрация находит в обработке сигналов в радио- и гидролокации, цифровой связи, ультразвуковой диагностике, а также других областях, где есть априорная информация об опорном (зондирующем) сигнале.

Практический эффект от применения согласованной фильтрации также существенно зависит от выбора опорного (зондирующего) сигнала, основанного на анализе его функции неопределенности [2].

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанной речью или мелодией длительностью $8 - 15$.
2. Построить графики звукового сигнала во временной и частотной области.
3. Добавить нормальный шум мощностью сопоставимой с мощностью звукового сигнала длительность $0{,}1 - 0{,}5$ с начиная с временной метки сигнала Tau_0.
4. Построить графики сигнала во временной и частотной области.
5. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.
6. Реализовать согласованную фильтрацию полученного звукового сигнала с опорным сигналом в виде наложенного нормального шума.
7. Построить график сигнала, полученного в результате согласованной фильтрации.
8. С помощью построенного графика определить временную метку начала шумового фрагмента сигнала и сравнить с исходным значением τ_0.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

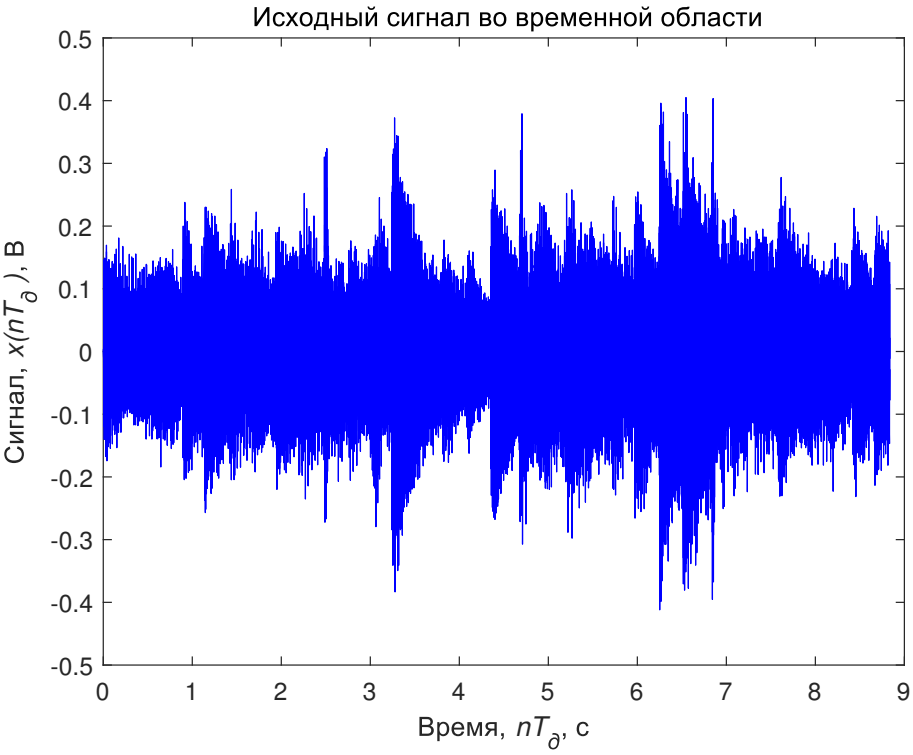
1 Согласованная фильтрация

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
```

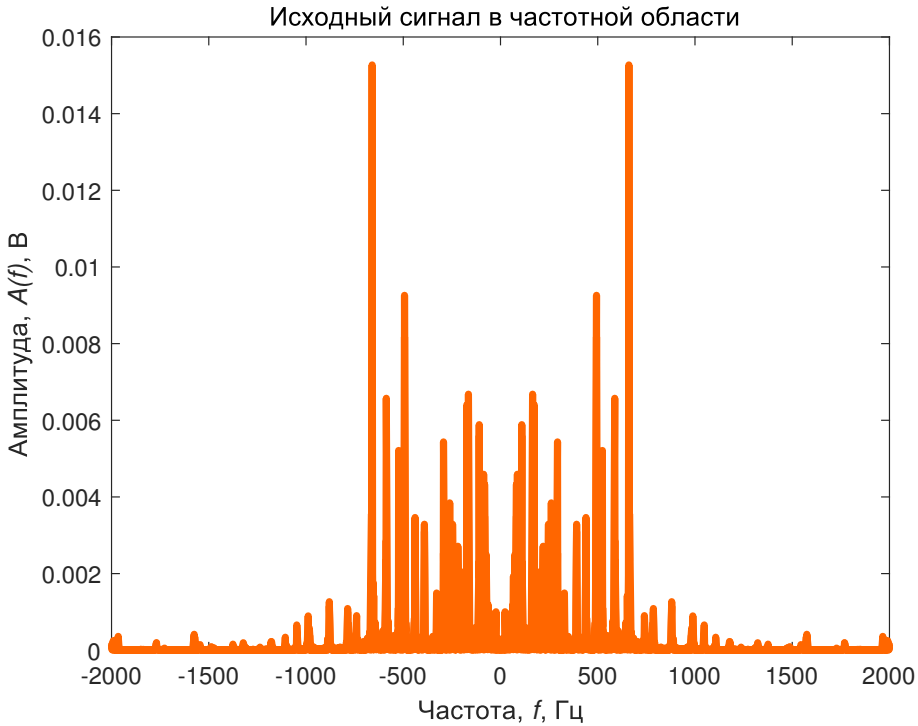
1.2 Чтение файла с мелодией

```
[data,rate] = audioread('melody.wav');
t = linspace(0, length(data)/rate, length(data)); % Формирование области определ
ения
figure; plot(t, data, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



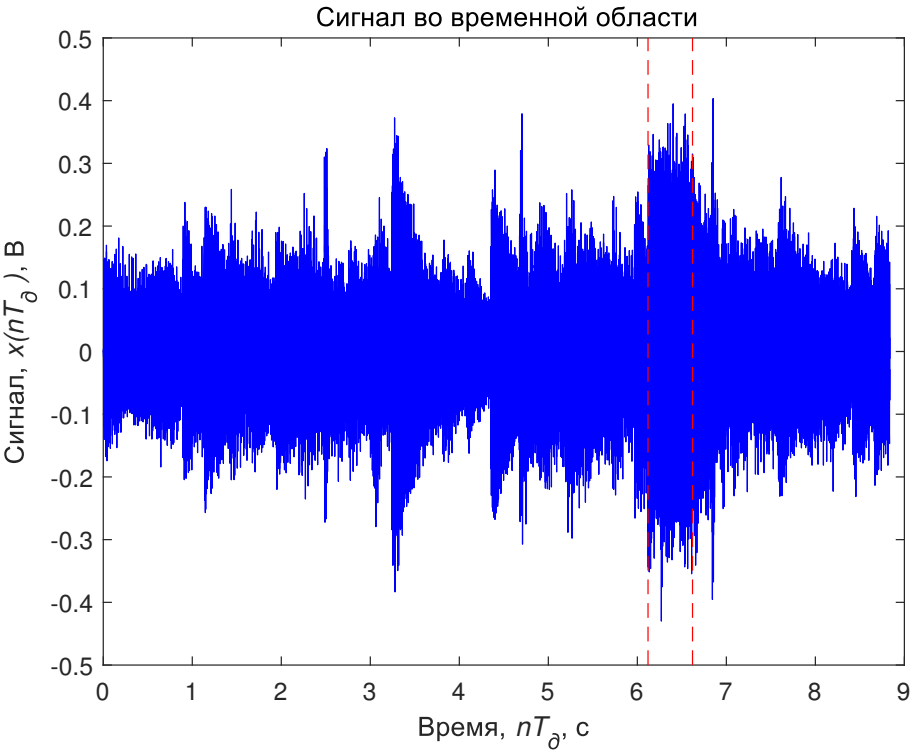
1.3 Построение амплитудного спектра сигнала

```
f = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([-2000 2000]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



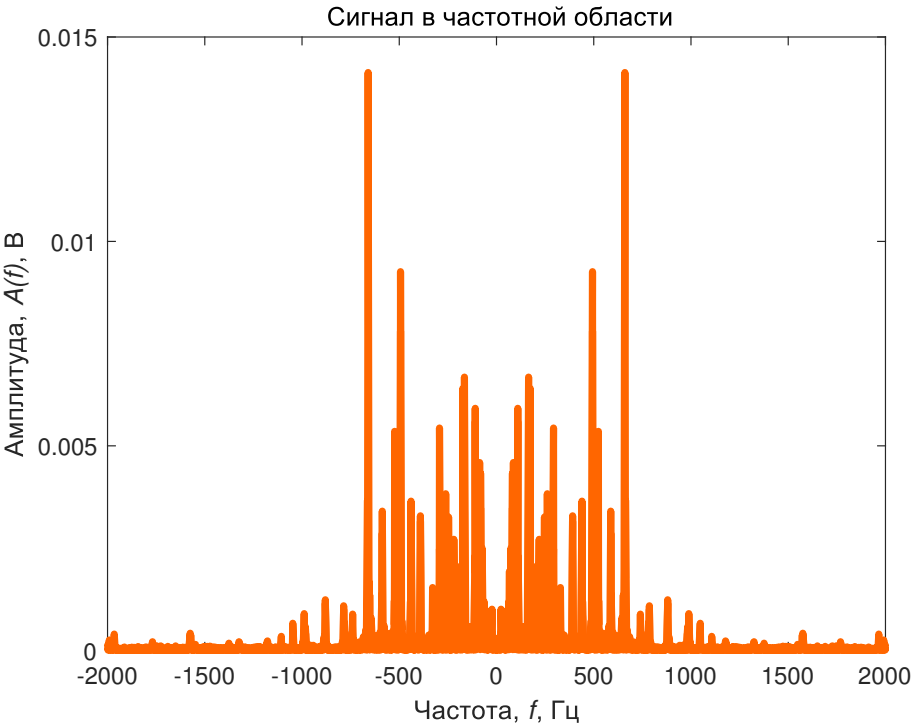
1.4 Добавление нормального шума к фрагменту мелодии

```
noise_dur = 0.5; % Длительность шума 0.5 секунд
noise_len = noise_dur*rate; % Количество отсчётов шума
noise = randn(noise_len, 1); % Формирование вектора отсчётов с нормальным распре
делением
t_0 = 6.12; % Метка начала шумового фрагмента с 6.12 секунд
offset = 6.12*rate;
for i = 1 : noise_len
    data(offset + i) = noise(i)/10; % Добавление шума к мелодии от 6.12 секунд
end
% Построение графика с выделенным шумовым фрагментом мелодии
figure; plot(t, data, 'b', [t_0 t_0+noise_dur; t_0 t_0+noise_dur], [-0.5 -0.5;
0.5 0.5], 'r--');
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



1.5 Построение амплитудного спектра сигнала

```
f = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([-2000 2000]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```

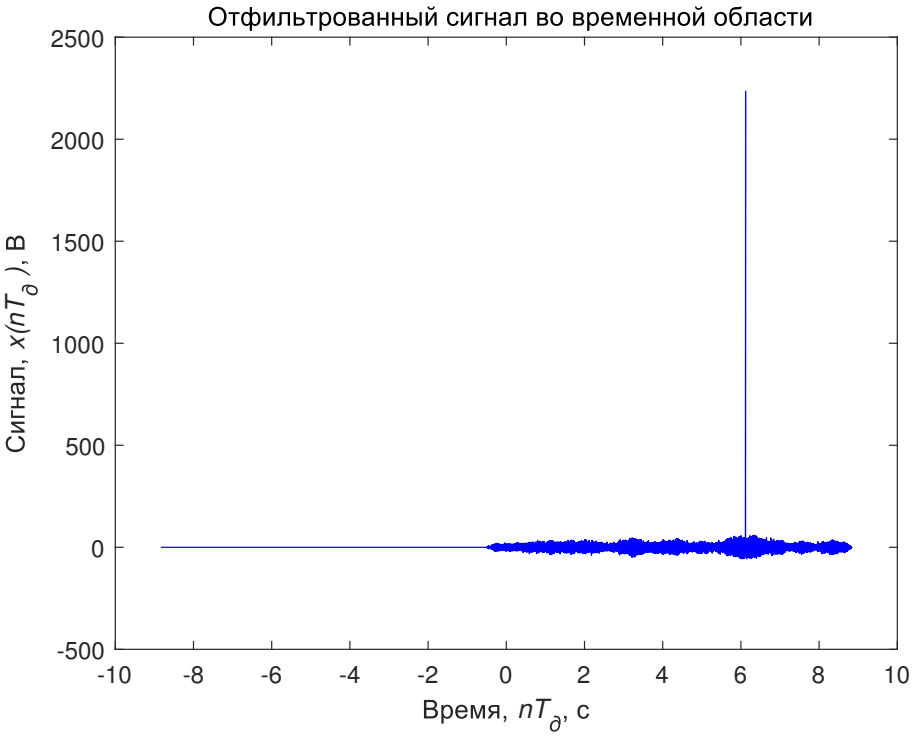


1.6 Сохранение сигнала с шумом в файл

```
audiowrite('melody_noise.wav',data,rate);
```

1.7 Согласованная фильтрация

```
% Взаимная корреляция между мелодией и шумовым фрагментом
[y_corr, x_corr] = xcorr(data(1:end,1), noise);
t_corr = x_corr/rate; % Перевод в секунды
figure; plot(t_corr, y_corr, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```




```
[value, index] = max(y_corr); % Нахождение максимума корреляционной функции
t_1 = t_corr(index); % Вычисление временной метки начала шумового фрагмента
fprintf(['Рассчитанная метка начала шумового фрагмента: %.3f с. ' ...
    'Разность между рассчитанной и установленной: %.3f с.\n'],t_1, t_1 - t_0);
```

Рассчитанная метка начала шумового фрагмента: 6.120 с.

Разность между рассчитанной и установленной: 0.000 с.

Контрольные вопросы

1. Какая информация является результатом работы согласованного фильтра?
2. Чем обусловлено название согласованного фильтра?
3. Приведите пример практического использования согласованной фильтрации.
4. Как правильно выбрать опорный сигнал при обработке сигналов методом согласованной фильтрации?
5. Чему соответствует максимум выходного сигнала результата работы согласованного фильтра?

Лабораторная работа №5

Алгоритм Герцеля

Основные теоретические сведения

Как известно, одним из наиболее эффективных методов спектрального анализа является быстрое преобразование Фурье (БПФ) [8], позволяющее получить полный набор частотных отсчетов $X(k)$ в диапазоне от нуля до половины частоты дискретизации для временной последовательности $x(n)$ сигнала. На практике в некоторых случаях нет необходимости в получении полного частотного спектра сигнала, но требуется определение мощности одной или нескольких определенных гармоник.

Одним из наиболее широко используемых примеров устройств, применяемых в повседневной жизни и использующих алгоритмы выборочного спектрального анализа, является приемник DTMF (Dual-Tone Multi-Frequency) сигналов мобильного или стационарного телефона. DTMF-сигнал представляет собой двухтональный многочастотный аналоговый сигнал, используемый для набора телефонного номера и ручного ввода команд при использовании абонентов интерактивных телефонных систем. Для кодирования любого символа клавиатуры телефона необходимо использовать сигнал, представляющий собой сумму двух синусоид, частоты которых определяются с помощью таблицы 1.

Таблица 1. Соответствие клавиши клавиатуры и частоты

	1209 Гц	1336 Гц	1477 Гц	1633 Гц
697 Гц	1	2	3	A
770 Гц	4	5	6	B
852 Гц	7	8	9	C
941 Гц	*	0	#	D

Соответственно, для декодирования DTMF-сигнала необходимо провести оценку мощности каждой из восьми гармоник. В данном случае применение алгоритма БПФ является нерациональным. Альтернативным вариантом может являться применение дискретного преобразования Фурье

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn},$$

но более эффективным является применение алгоритма Герцеля [9].

Алгоритм Герцеля заключается в следующем:

1. Последовательно вычисляются члены последовательности s_n для $n = 0, \ldots, N-1$ по рекуррентной формуле $s_n = 2cos(\frac{2\pi k}{N}) s_{n-1} - s_{n-2} + x_n$,

где $s_{-1} = s_{-2} = 0$.

2. Искомое значение частотного компонента получается как $X_k = e^{\frac{2\pi i}{N}k} s_{N-1} - s_{N-2} + s_{N-2}^2$.

В случае, когда требуется вычислить только мощность сигнала, а его фаза не важна, на втором этапе алгоритма вместо комплексного значения частотного компонента вычисляется квадрат его модуля по формуле

$$|X_k|^2 = s_{N-1}^2 - 2cos(\frac{2\pi k}{N}) s_{N-1} - s_{N-2} + s_{N-2}^2.$$

По своей сути алгоритм Герцеля осуществляет фильтрацию сигнала цифровым фильтром с бесконечной импульсной характеристикой (фильтр Герцеля). Пример амплитудно-частотной характеристики фильтра Герцеля приведен на рисунке 9.

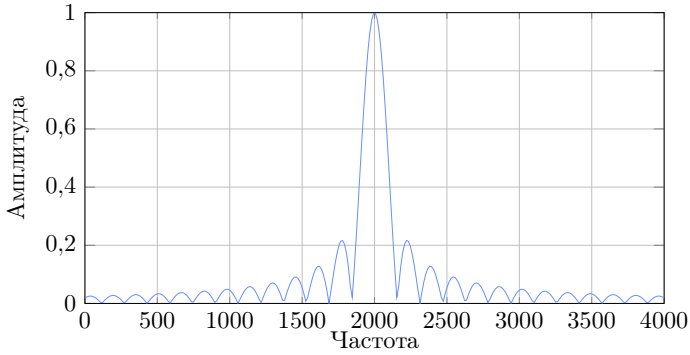


Рис. 9. Амплитудно-частотная характеристика фильтра Герцеля с резонансной частотой 2 кГц

Наряду с преимуществами, связанными с вычислительной сложностью, алгоритм Герцеля имеет недостатки в части устойчивости, свойственные всем фильтрам с бесконечной импульсной характеристикой.

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанным битональным сигналом, соответствующем одной из клавиш мобильного телефона.
2. Построить графики звукового сигнала во временной и частотной области.
3. С помощью алгоритма Герцеля детектировать частоты гармоник битонального сигнала и определить соответствующий символ клавиатуры телефона.
4. Сравнить полученный символ с исходным.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

1 Декодирование DTMF сигнала с помощью алгоритма Герцеля

1.1 Инициализация и формирование значений основных параметров

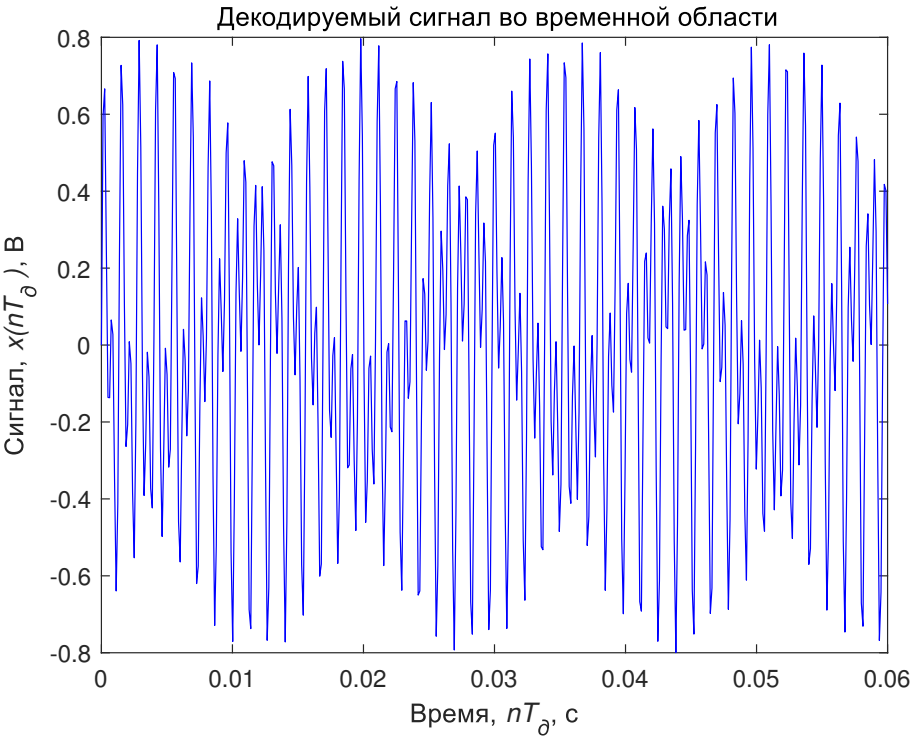
```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
DTMFfreqs = [697 770 852 941 1209 1336 1477 1633]; % Тоновый набор DTMF
```

1.2 Запись аудиосигнала с микрофона

```
Fs=8000; % Частота дискретизации аудиозаписи
Nseconds = 2; % Время записи сигнала
% Запись сигнала с микрофона
recorder = audiorecorder(Fs,16,1);
recordblocking(recorder,Nseconds);
% Чтение отсчётов аудиосигнала в массив
x = getaudiodata(recorder,'uint8');
% Запись звукового файла с записанным сигналом
audiowrite('DTMF\record.wav',x(Fs/2-1:end),Fs);
```

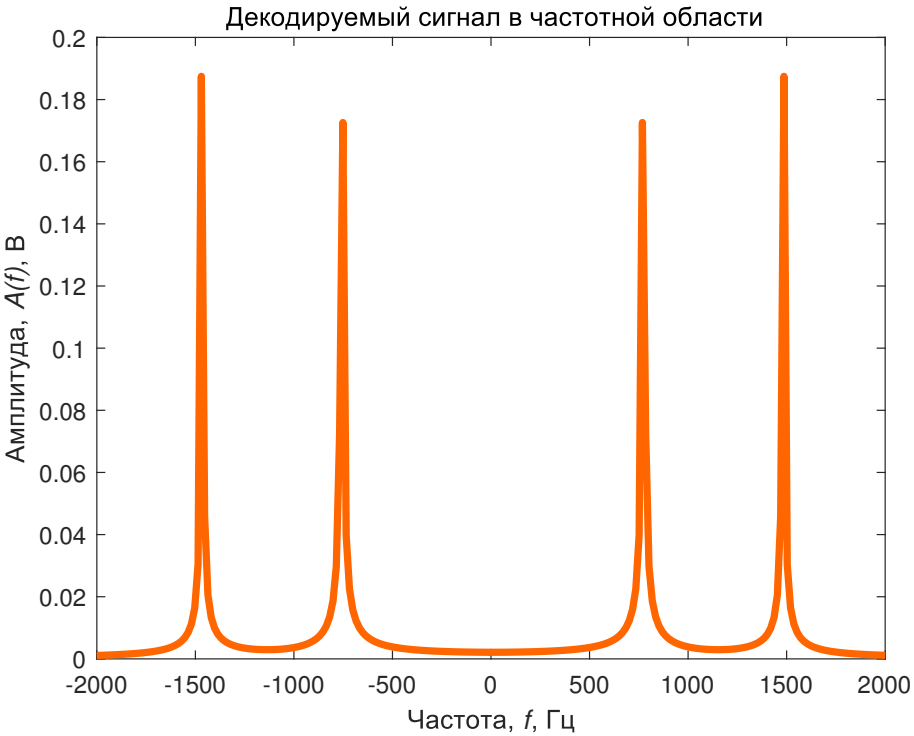
1.3 Чтение аудиосигнала для декодирования

```
[data,rate] = audioread('DTMF\6.wav'); % 0..9, A..D, star, sharp или record.wav
t = linspace(0, length(data)/rate, length(data)); % Формирование области определ
ения
figure; plot(t, data, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Декодируемый сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



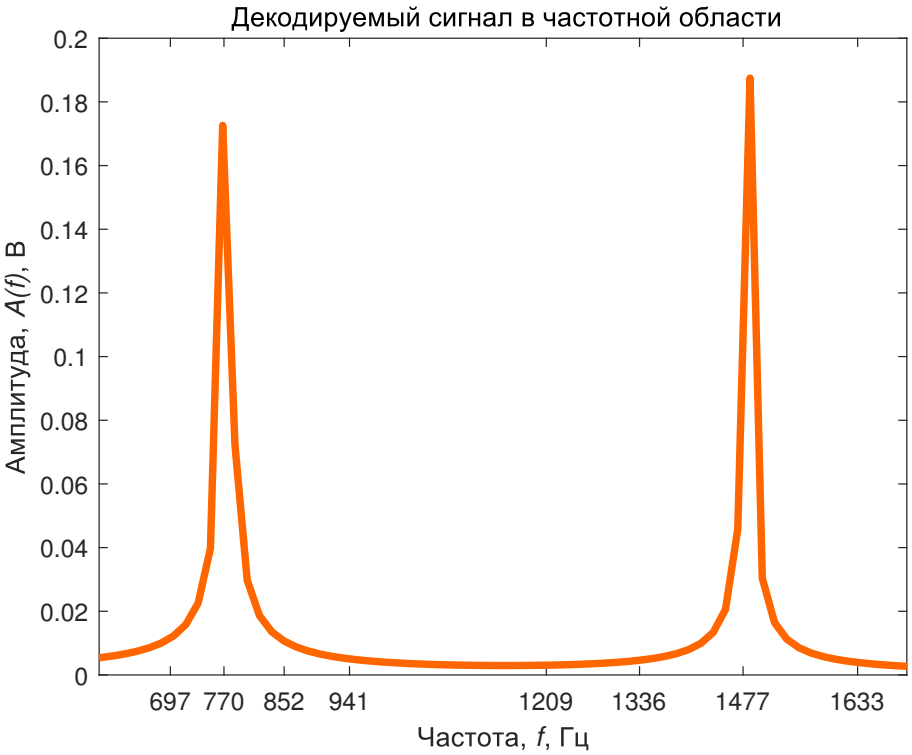
1.4 Построение амплитудного спектра декодируемого сигнала

```
f = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([-2000 2000]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Декодируемый сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



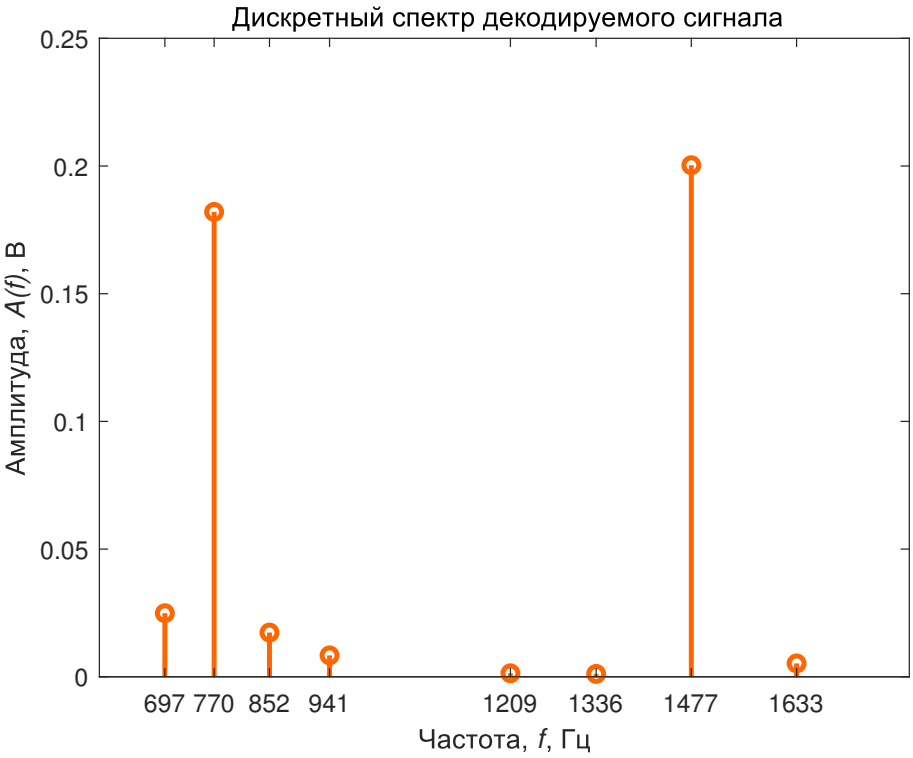
1.5 Построение амплитудного спектра в области тонового набора DTMF

```
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([600 1700]); % Ограничение области определения тоновым набором
xticks(DTMFfreqs); % Подписать только частоты из тонового набора DTMF
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Декодируемый сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.6 Построение дискретного амплитудного спектра с помощью алгоритма Герцеля

```
N = 205; % Количество точек ДПФ
% Расчёт номеров спектральных отсчётов для тонового набора
freq_indexes = round(DTMFfreqs/rate*N) + 1;
% Вычисление значений спектра по алгоритму Герцеля
gdata = abs(goertzel(data(1:N),freq_indexes))/N;
% Построение графика полученного спектра
stem(DTMFfreqs, gdata, 'Color', fColor, 'LineWidth', 2);
ax = gca; % Получение текущей системы координат
ax.XTick = DTMFfreqs; % Подпись делений оси абсцисс
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Дискретный спектр декодируемого сигнала'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.7 Декодирование сигнала по рассчитанным значениям спектра

```
% Матрица символов DTMF
DTMF_symbols = ['1' '2' '3' 'A';
                '4' '5' '6' 'B';
                '7' '8' '9' 'C';
                '*' '0' '#' 'D'];
% Нахождение двух наибольших частот
[value, row] = max(gdata(1:4));
[value, column] = max(gdata(5:8));
% Вывод декодированного символа
fprintf('Распознанный символ: %s\n', DTMF_symbols(row,column));
```

Распознанный символ: 6

Контрольные вопросы

1. В чем преимущество и недостатки алгоритма Герцеля по сравнению с преобразованием Фурье?
2. Приведите пример практического применения алгоритма Герцеля.
3. Каким типом фильтра по пропускаемой частоте является фильтр Герцеля?
4. Какими недостатками обладает фильтр Герцеля?

Лабораторная работа №6

Цифровое гетеродинирование

Основные теоретические сведения

На практике для обработки высокочастотных полосовых сигналов возникает необходимость смещения их спектра в область низких частот. Так, например, полоса частот радиосигналов сети Wi-Fi находится в диапазоне от $f_{\text{н}} = 2432$ МГц до $f_{\text{в}} = 2443$ МГц. Даже преобразование сигнала такой частоты из аналогового вида в цифровой представляет собой сложную с точки зрения применяемых технологий задачу. Обработка же сигнала, оцифрованного с частотой дискретизации около 5 ГГц (исходя из требований теоремы Котельникова) накладывает крайне серьезные требования вычислительной мощности и объемы памяти аппаратуре приемника. С другой стороны, если выполнить перенос заданной полосы частот в область низких частот, например, таким образом, чтобы нижняя граничная частота совпала с нулевой частотой, то минимальная частота дискретизации полученного сигнала будет совпадать с удвоенной шириной полосы сигнала

$$f_{\text{д}} = 2(f_{\text{в}} - f_{\text{н}})$$

и для предложенного примера составит всего 22 МГц. Очевидно, что подбор аналого-цифрового преобразователя в данном случае существенно упрощается. В радиотехнике такой процесс понижения частоты реализуется посредством гетеродинирования.

Гетеродини́рование — это преобразование сигнала одной частоты в пару сигналов с разными частотами (промежуточные частоты) с сохранением фазы исходного сигнала.

Гетеродинирование осуществляется с помощью вспомогательного генератора гармонических колебаний — *гетеродина*, а в основе этого процесса лежит тригонометрическое выражение

$$\sin\theta \sin\varphi = \frac{1}{2}\cos(\theta - \varphi) - \frac{1}{2}\cos(\theta + \varphi),$$

которое применительно к математическим моделям сигналов может быть переписано в виде

$$\sin(2\pi f_1 t)\sin(2\pi f_2 t) = \frac{1}{2}\cos[2\pi(f_1 - f_2)t] - \frac{1}{2}\cos[2\pi(f_1 + f_2)t]. \tag{7}$$

Если применить к результату умножения сигналов (7) фильтр низких частот, то высокочастотную составляющую в виде гармоники $f_1 + f_2$ можно устранить, тогда на выходе получим исходный сигнал с частотой f_1, но смещенный в область низких частот на частоту f_2. Структурная схема гетеродинирования приведена

на рисунке 10.

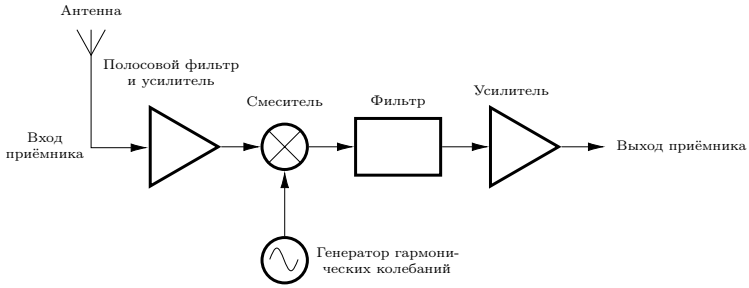


Рис. 10. Структурная схема гетеродинирования

Если по каким-либо причинам возникает необходимость в увеличении частоты, то к результату умножения сигналов (7) необходимо применить фильтр высоких частот и устранить низкочастотную составляющую в виде гармоники $f_2 - f_1$, тогда на выходе получим исходный сигнал с частотой f_1, но смещенный в область высоких частот на частоту f_2.

Иллюстрация результата переноса спектра полосового сигнала из области несущей частоты в область промежуточной (пониженной) частоты приведена на рисунке 11.

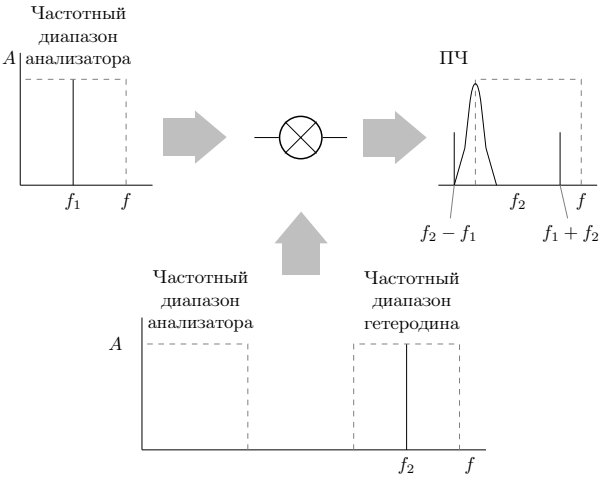


Рис. 11. Иллюстрация переноса спектра полосового сигнала из области несущей частоты в область промежуточной частоты

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанным тональным сигналом со значением частоты в диапазоне от 10 кГц до 15 кГц.
2. Построить графики звукового сигнала во временной и частотной области.
3. С помощью метода гетеродинирования модифицировать сигнал переносом его частоты в диапазон от 500 Гц до 5500 Гц.
4. Построить графики звукового сигнала во временной и частотной области.
5. Записать полученный сигнал в новый WAV-файл. Прослушать полученный файл средствами операционной системы и сравнить с исходным.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

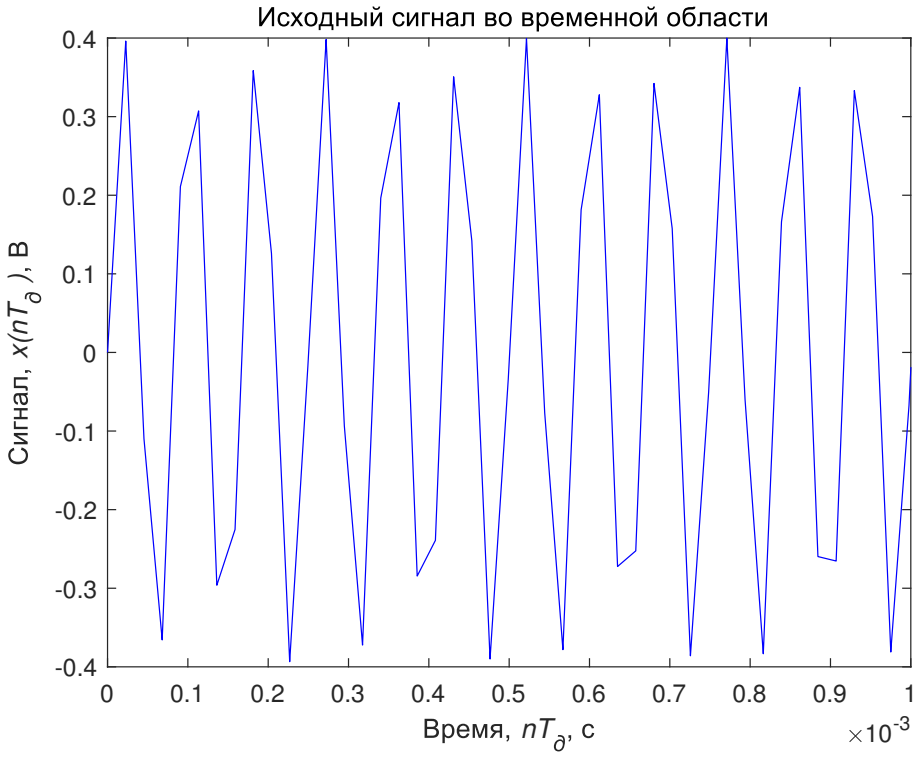
1 Гетеродинирование

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
```

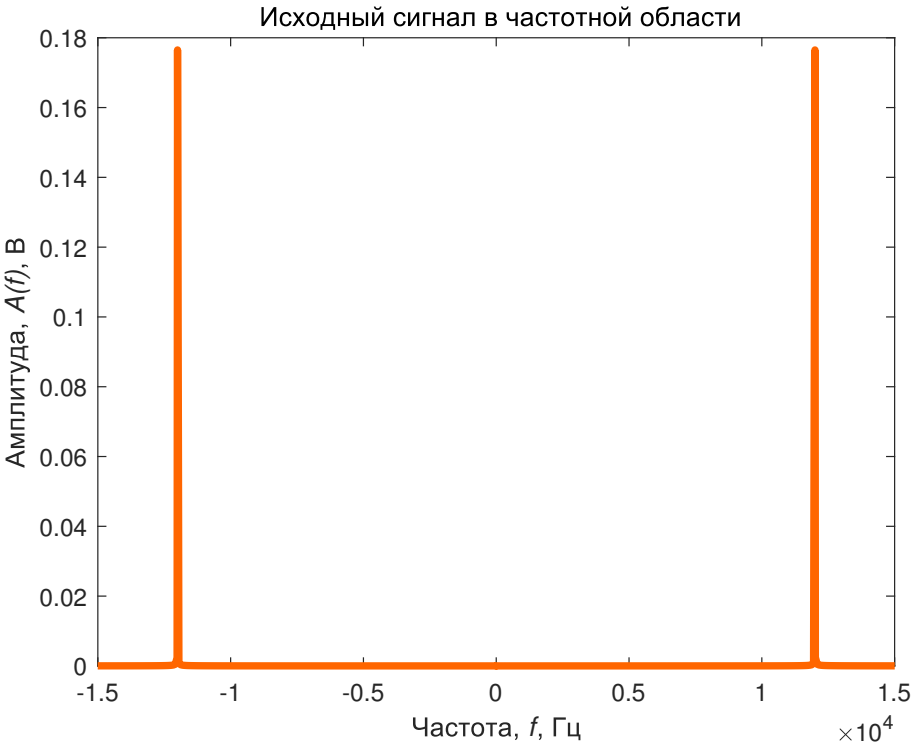
1.2 Чтение звукового файла с тональным сигналом

```
[data,rate] = audioread('12kHz.wav');
t = linspace(0, length(data)/rate, length(data)); % Формирование области определ
ения
figure; plot(t, data, 'Color', tColor);
xlim([0 0.001]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



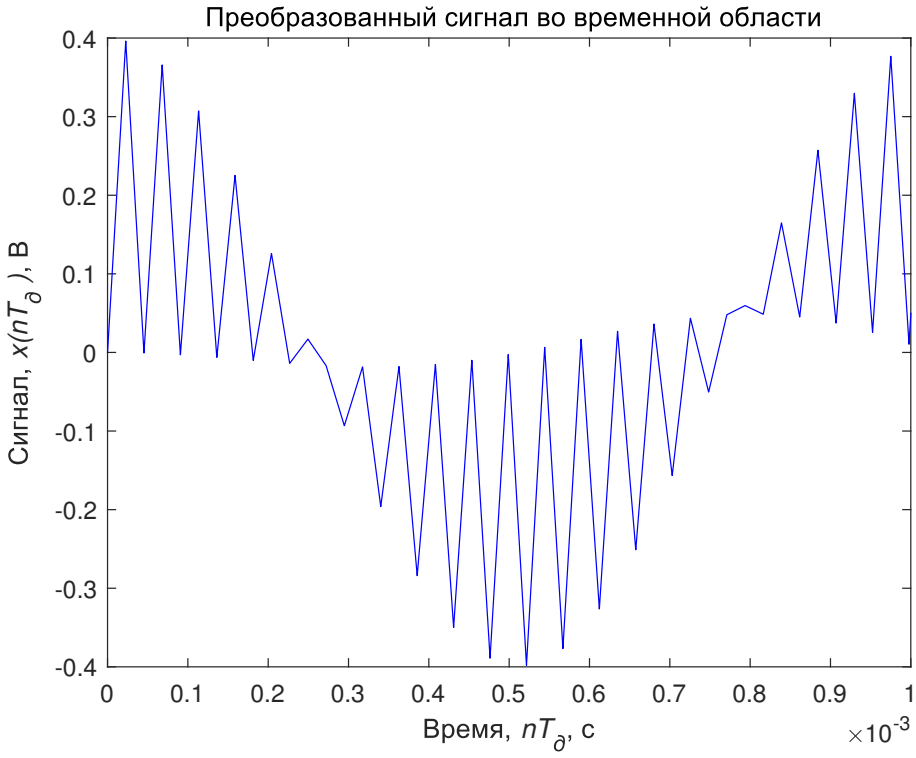
1.3 Построение амплитудного спектра сигнала

```
f = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([-15000 15000]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



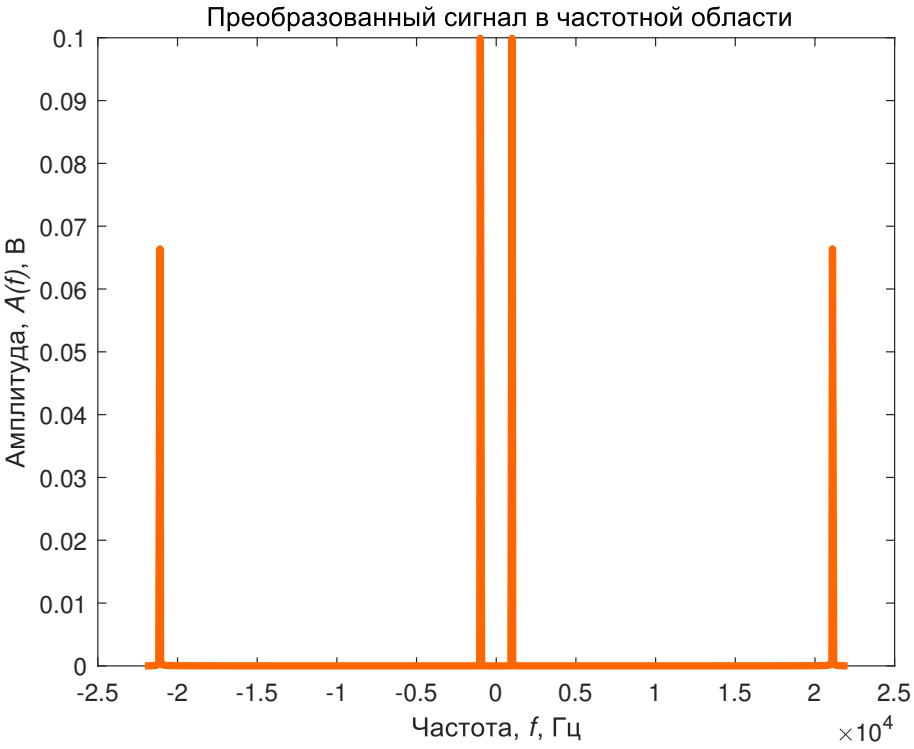
1.4 Модификация сигнала с помощью гетеродинирования

```
% Формирование вспомогательного гармонического сигнала частотой 11 кГц
t = linspace(0, length(data)/rate, length(data));
sin_11kHz = sin(2*pi*11000*t);
% Гетеродинирование
result = zeros(length(data),1);
for i = 1 : length(data)
    result(i) = data(i)*sin_11kHz(i); % Умножение исходного и вспомогательного
end
% Построение графика полученного сигнала
figure; plot(t, result, 'Color', tColor);
xlim([0 0.001]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Преобразованный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



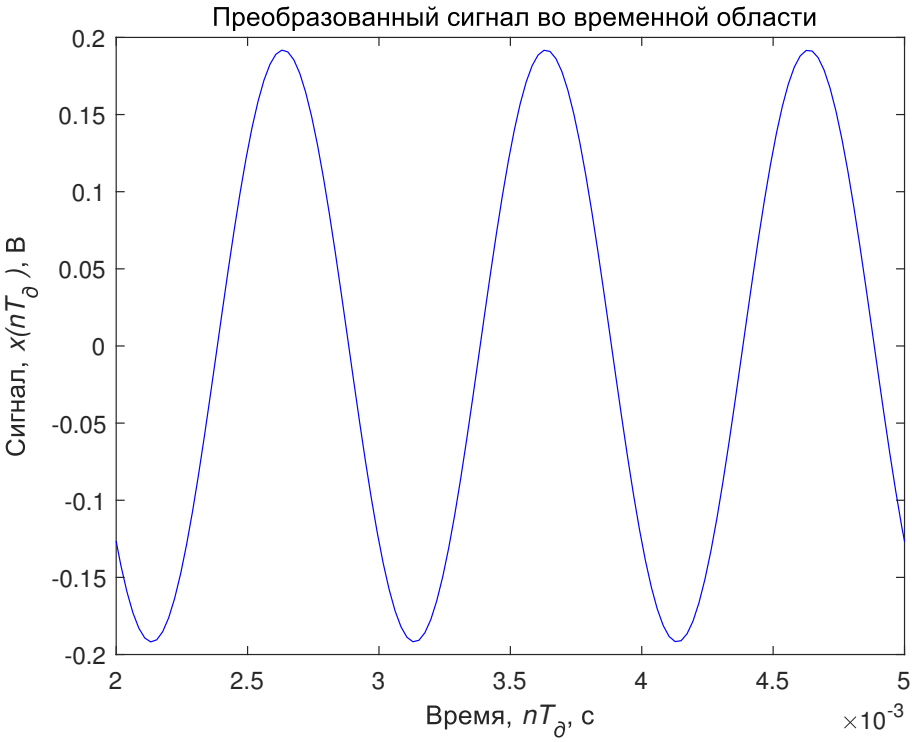
1.5 Построение амплитудного спектра преобразованного сигнала

```
f = linspace(0, rate, length(result)); % Формирование области определения
fdata = abs(fft(result)/length(result)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Преобразованный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



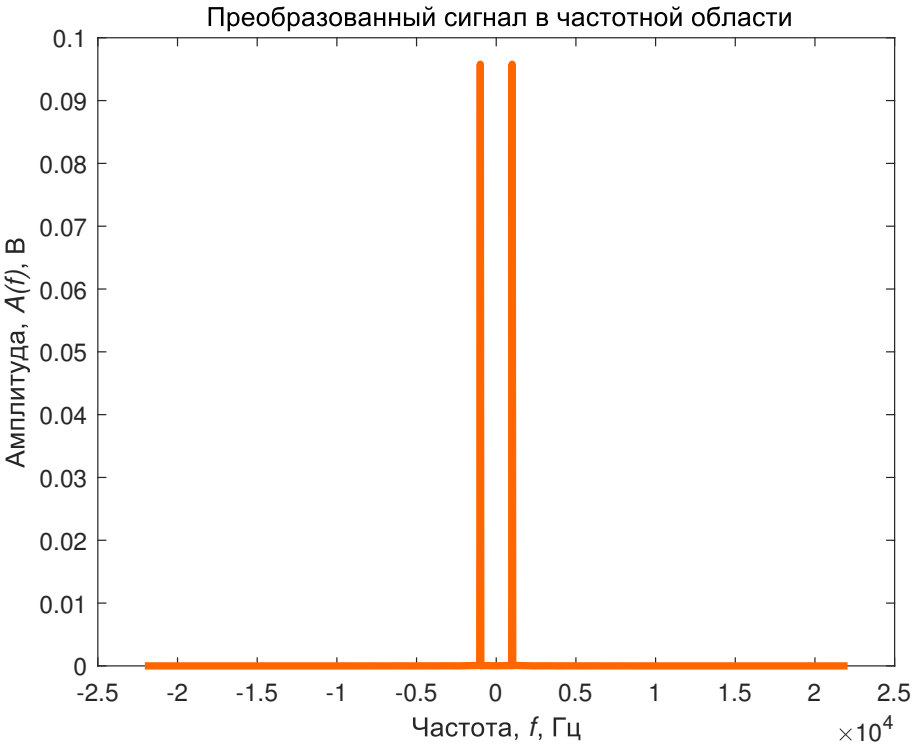
1.6 Фильтрация сигнала

```
load('LowPassFilter.mat'); % Загрузка файла с коэффициентами фильтра
global LowPassCoef; % Массив коэффициентов КИХ-фильтра
filtered = filter(LowPassCoef, 1, result); % Подавление высокочастотной составляющей
% Построение графика полученного сигнала
figure; plot(t, filtered, 'Color', tColor);
xlim([0.002 0.005]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



1.7 Построение амплитудного спектра отфильтрованного сигнала

```
f = linspace(0, rate, length(filtered)); % Формирование области определения
fdata = abs(fft(filtered)/length(filtered)); % Формирование значений спектра
figure; plot([-fliplr(f(1:end/2)) f(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Отфильтрованный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.8 Сохранение полученного сигнала в звуковой файл

```
1 audiowrite('result.wav', filtered, rate);
```

Контрольные вопросы

1. Что такое гетеродинирование?
2. Приведите пример практического применения гетеродинирования.
3. Почему при переносе частоты сигнала методом гетеродинирования необходимо дополнительно применять фильтр?
4. Как выбрать величину частоты гетеродина?

Лабораторная работа №7

Цифровой демодулятор фазоманипулированных сигналов

Основные теоретические сведения

Как известно, для передачи цифровых данных по радиоканалу широко применяются фазоманипулированные гармонические сигналы с заданной несущей частотой колебания [10].

Опишем сигнал несущей частоты выражением

$$S(t) = A\cos(2\pi ft + \theta),$$

где A — амплитуда, f — частота, θ — начальная фаза несущего колебания.

Тогда выражение для фазоманипулированного сигнала будет иметь вид

$$S_m(t) = A\cos(2\pi ft + \theta_k(t)),\ k = \overline{1, N},$$

где k — порядковый номер импульса.

В пределах каждого импульса начальная фаза сигнала не изменяется, а в начале каждого нового импульса принимает одно из дискретных значений, определяемых выражением

$$\theta_j = \frac{\pi(2j-1)}{J},\ j = \overline{1, J}, \tag{8}$$

где j — порядковый номер значения из дискретного множества начальных фаз сигнала, J — общее количество элементов множества начальных фаз.

В зависимости от значения J из формулы (8) различают виды фазовой манипуляции: двоичная ($J = 2$), квадратурная ($J = 4$), восьмеричная ($J = 8$). В современных системах связи данные виды манипуляции обозначают также BPSK, QPSK и 8-PSK соответственно. При этом количество бит B, кодируемых одним импульсом определяется выражением

$$B = \log_2 J. \tag{9}$$

Из выражения (9) следует, что с увеличением J ёмкость закодированной информации, а для систем цифровой связи и теоретическая скорость передачи данных возрастает. Теоретически J может принимать значения и больше 8, тогда в общем виде модуляцию можно обозначить как M-PSK. На практике в системах цифровой связи с уменьшением соотношения сигнал-шум или с увеличением J на стороне приемника становится сложнее различать значения фаз [2].

Для получения цифровых данных, закодированных с помощью фазоманипулированных сигналов на приемной стороне используется демодулятор, структурная схема которого приведена на рисунке 12.

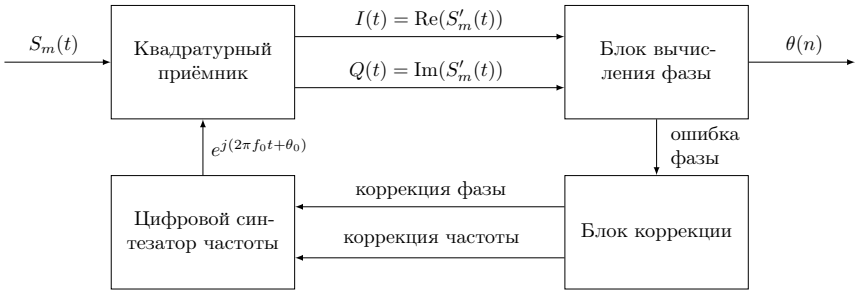


Рис. 12. Структурная схема демодулятора приемника фазоманипулированных сигналов

На вход «Блока вычисления фазы» поступают оцифрованные отсчеты синфазной (I) и квадратурной (Q) компонент комплексного сигнала, полученного после умножения входного сигнала на гармонический комплексный сигнал опорной частоты в квадратурном приемнике.

Значение фазы комплексного числа (рисунок 13) вычисляется исходя из выражения

$$M = \sqrt{I^2 + Q^2};$$
$$\phi = \mathrm{arctg}\left[\frac{Q}{I}\right].$$

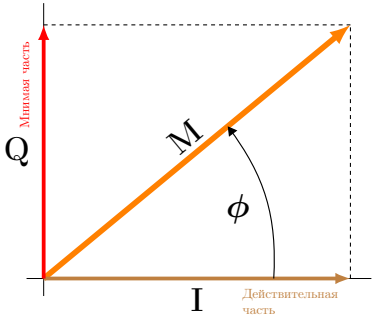


Рис. 13. Изображение комплексного числа

На практике прямое вычисление функции арктангенса затруднено, поэтому используется алгоритм CORDIC (Coordinate Rotation Digital Computer) [11].

В реальных физических условиях в начальный момент частота опорного генератора может довольно сильно отличаться от частоты несущего сигнала в силу, например, эффекта Доплера. Оценка смещения разностной частоты и ее

знака позволит внести корректировку частоты и фазы опорного генератора в блоке коррекции и уменьшить разностную частоту [12].

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного фазоманипулированного сигнала с длительностью чипа τ, несущей частотой 50 Гц, кодирующего последовательность $A = a_0, a_1, ..., a_N$ значениями фаз $\phi_0 = 0$, $\phi_1 = \pi$.
2. Построить графики звукового сигнала во временной и частотной области.
3. Провести декодирование последовательности.
4. Сравнить полученную последовательность с исходной.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

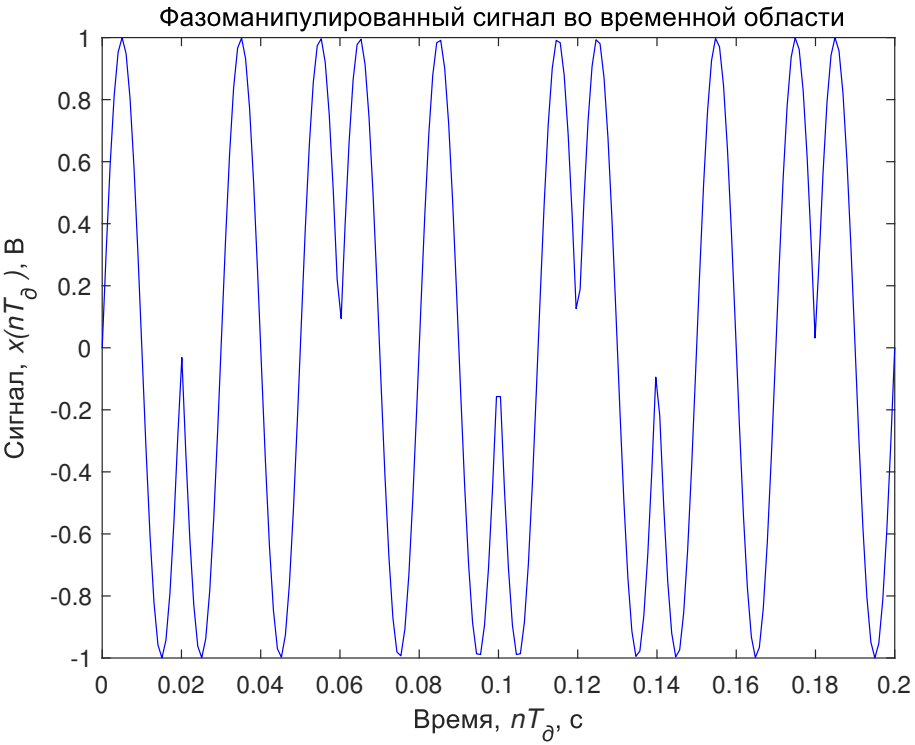
1 Декодирование фазоманипулированных сигналов

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
fd = 1000; % Частота дискретизации
f = 50; % Частота фазоманипулированного сигнала
T_points = fd/f; % Отсчётов за период
A = [0, 1, 1, 0, 0, 1, 0, 1, 1, 0]; % Кодируемая последовательность
tmax = length(A)/f; % Длительность сигнала
N = tmax*fd; % Количество отсчётов сигнала
t = linspace(0, tmax, N); % Формирование области определения
```

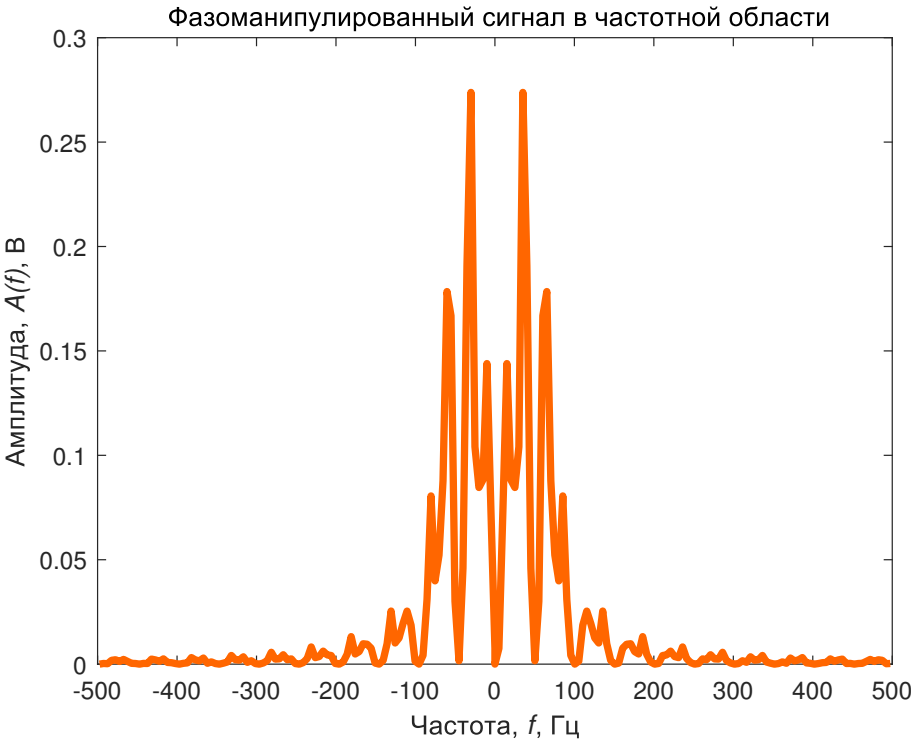
1.2 Моделирование фазоманипулированного сигнала

```
% Массив отсчётов фазоманипулированного сигнала
codedSignal = zeros(1, N);
for i = 0 : length(A)-1
for j = 1 : T_points
codedSignal(i*T_points + j) = sin(2*pi*f*t(i*T_points + j) + A(i+1)*pi);
end
end
figure; plot(t, codedSignal, 'Color', tColor);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Фазоманипулированный сигнал во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
```



1.3 Построение амплитудного спектра сигнала

```
f_axis = linspace(0, fd, length(codedSignal)); % Формирование области определени
я
fdata = abs(fft(codedSignal)/length(codedSignal)); % Формирование значений спект
ра
figure; plot([-fliplr(f_axis(1:end/2)) f_axis(1:end/2)], fftshift(fdata),...
'Color', fColor, 'LineWidth', 3);
xlim([-500 500]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Фазоманипулированный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



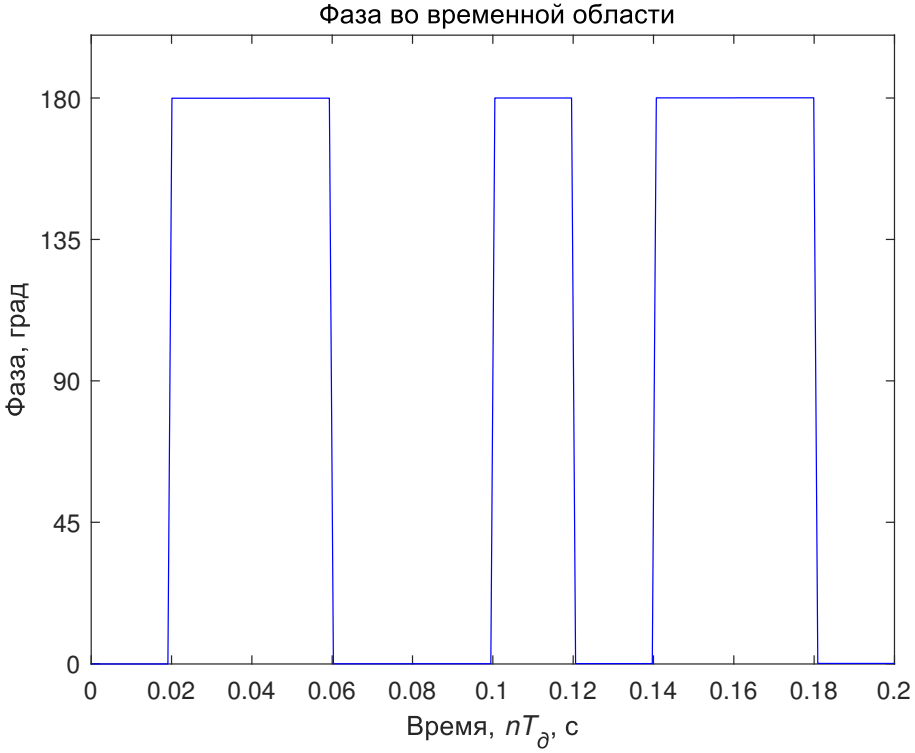
1.4 Декодирование последовательности

```
% Формирование вспомогательного комплексного гармонического сигнала
signal_Q = sin(2*pi*f*t+pi);
signal_I = cos(2*pi*f*t);
% Гетеродинирование
result_Q = zeros(1,N);
result_I = zeros(1,N);
for i = 1 : N
result_Q(i) = codedSignal(i)*signal_Q(i); % Умножение исходного и вспомогательно
го
result_I(i) = codedSignal(i)*signal_I(i);
end
% Декодирование
phase_code = zeros(1,length(A));
flag = false; % Флаг неравенства последовательностей
for i = 0 : length(A)-1
% Подавляем высокочастотную составляющую после гетеродинирования
meanQ = mean(result_Q(i*T_points+1:(i+1)*T_points));
meanI = mean(result_I(i*T_points+1:(i+1)*T_points));
for j = 1 : T_points
result_Q(i*T_points + j) = meanQ;
```

```
result_I(i*T_points + j) = meanI;
end
% Вычисляем фазу и преобразуем в код
phase_code(i+1) = round((atan2(meanQ,meanI)+pi/2)/pi);
if phase_code(i+1) ~= A(i+1)
flag = true; % Найдено несоответствие
end
end
```

1.5 Просмотр результата декодирования

```
phase = (atan2(result_Q,result_I)+pi/2)/pi*180;
figure; plot(t, phase, 'b');
yticks([0 45 90 135 180]);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Фаза во временной области'); % Заголовок
xlabel('Время,\it nT_д\rm, с'); % Надпись оси абсцисс
ylabel('Фаза\rm, град'); % Надпись оси ординат
```



Кодируемая последовательность:

```
disp(A);
```

0 1 1 0 0 1 0 1 1 0

Декодированная последовательность:

```
disp(phase_code);
```

0 1 1 0 0 1 0 1 1 0

Результат сравнения последовательностей

```
if flag
disp('Декодированная последовательность не соответствует исходной');
else
disp('Декодированная последовательность соответствует исходной');
end
```

Декодированная последовательность соответствует исходной

Контрольные вопросы

1. Какие бывают виды модуляции (манипуляции) сигнала?
2. Какие преимущества и недостатки сопутствуют увеличению количеству градаций фаз при фазоманипулированном кодировании сигнала в цифровой передаче данных?
3. Каким образом можно вычислить фазу отсчета комплексного сигнала?
4. Чем определяется скорость беспроводной цифровой передачи информации при использовании фазмоианипулированного кодирования?
5. Отличаются ли спектры гармонического сигнала и фазоманипулированного сигнала при использовании одной и той же несущей частоты?

Лабораторная работа №8

Преобразование Гильберта-Хуанга

Основные теоретические сведения

Спектральный анализ на базе преобразования Фурье имеет ограничения применения для линейных систем и стационарных сигналов. На практике это условие не всегда может быть выполнено, что приводит к необходимости ряда допущений, которые влияют на точность полученных результатов.

Преобразование Гильберта-Хуанга (Hilbert-Huang transform, HHT) было предложено Норденом Хуангом в конце XX века и основано на спектральном анализе сигналов Гильберта. В отличие от преобразования Фурье, а также подобных преобразований, использующих определенный базис, преобразование Гильберта-Хуанга не требует определенного аналитическим образом базиса и может применяться для нестационарных и нелинейных данных. Базисные функции являются адаптивными и носят название эмпирических мод, а процесс их получения, предложенный Хуангом, называется эмпирической модовой декомпозицией (Empirical Mode Decomposition, EMD).

В процессе эмпирической декомпозиции полученные моды должны представлять собой линейные или нелинейные внутренние колебания (intrinsic mode functions, IMF), для которых с помощью преобразования Гильберта можно получить значения мгновенных частот. Выполнение данного условия обеспечивается, если функции внутренних колебаний обладают свойствами [13]:

1. Количество локальных экстремумов и количество пересечений нуля не должны отличаться более, чем на единицу.
2. В любой точке функции среднее значение огибающих, определенных локальными максимумами и локальными минимумами, должно быть нулевым.

Допустим, что имеется произвольный сигнал $y\,(t)$. Сущность метода заключается в последовательном вычислении функций эмпирических мод $c_j(t)$ и остатков $r_j\,(t) = r_{j-1}\,(t) - c_j\,(t)$, где $j = 1,2,3,\ldots,n$ при $r_0 = y(t)$. Результатом разложения будет представление сигнала в виде суммы модовых функций и конечного остатка:

$$y\,(t) = \sum_{i=1}^{n} c_j\,(t) + r_n\,(t)\,, \tag{10}$$

где n — количество эмпирических мод, которое устанавливается в ходе вычислений.

Алгоритм разложения произвольного сигнала на моды определяется следующей последовательностью действий [14]:

Действие 1. Находим в сигнале $y(k)$ положение всех локальных экстремумов, максимумов и минимумов процесса (номера точек $k_{i.ext}$ экстремумов), и значения $y(k_{i.ext})$ в этих точках (рисунок 14). Между этими экстремумами

сосредоточена вся информация сигнала. Группируем раздельно для максимумов и для минимумов массивы координат $k_{i.ext}$ и соответствующих им амплитудных значений $y(k_{i.ext})$. Число строк в массивах максимумов и минимумов не должно отличаться более чем на 1.

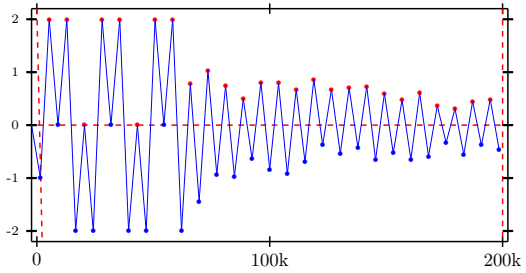


Рис. 14. Локализация экстремумов

Действие 2. Кубическим сплайном (или каким-либо другим методом) вычисляем верхнюю $u_t\,(k)$ и нижнюю $u_b\,(k)$ огибающие процесса соответственно, по максимумам и минимумам, как это показано на рисунке 15. Определяем функцию средних значений $m_1\,(k)$ между огибающими.

$$m_1 = \frac{u_t(k) + u_b(k)}{2},$$

Разность между сигналом $y\,(k)$ и функцией $m_1\,(k)$ дает нам первую компоненту отсеивания $(Sifting)$ – функцию $h_1\,(k)$, которая является первым приближением к первой функции IMF:

$$h_1(k) = y(k) - m_1(k).$$

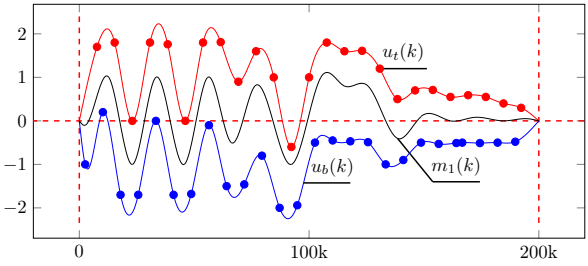


Рис. 15. Интерполяция экстремумов

Действие 3. Повторяем действия 1 и 2, принимая вместо $y\,(k)$ функцию $h_1\,(k)$, и находим второе приближение к первой модовой функции — функцию

$h_2(k)$:

$$h_2(k) = h_1(k) - m_2(k).$$

Последующие итерации выполняются аналогично:

$$h_i(k) = h_{i-1}(k) - m_i(k).$$

Одним из наиболее эффективных критериев останова итераций является задание предела, вычисленного с использованием двух последних приближений

$$\delta = \frac{\sum_k |h_{i-1}(k) - h_i(k)|^2}{\sum_k {h_{i-1}}^2(k)}.$$

Последнее значение $h_i(k)$ итераций принимается за наиболее высокочастотную функцию $c_1(k) = h_i(k)$, которая непосредственно входит в состав исходного сигнала $y(k)$. Это позволяет вычесть $c_1(k)$ из состава сигнала и оставить в нем более низкочастотные составляющие

$$r_1(k) = y(k) - c_1(k).$$

Функция $r_1(k)$ обрабатывается как новые данные по аналогичной методике с нахождением второй модовой функции — $c_2(k)$, после чего процесс продолжается:

$$r_2(k) = r_1(k) - c_2(k), \ldots$$

Таким образом, достигается декомпозиция сигнала в n — эмпирическом приближении, соответствующее (10). В дальнейшем каждая из модовых функций $c_j(t)$ подвергается преобразованию Гильберта H, то есть свертке с функцией $1/(\pi t)$, т.е.

$$x_j(t) = H[c_j(t)] = c_j(t) * \frac{1}{\pi t} = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{c_j(\tau)}{t - \tau} d\tau.$$

По сути преобразование Гильберта изменяет фазу всех частотных составляющих сигнала на $\pi/2$, и позволяет получить сигнал ортогональный исходному сигналу Это позволяет сформировать из сигналов $c_j(t)$ и $x_j(t)$ комплексный аналитический сигнал $z(t)$ (рисунок 16), как

$$z(t) = x_j(t) + ic_j(t).$$

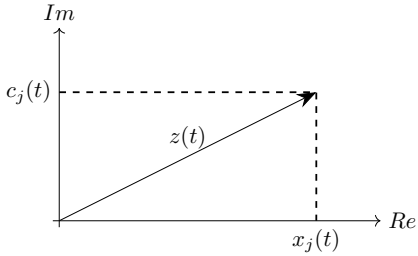


Рис. 16. Изображение комплексного числа

Подобное представление позволяет легко определить огибающую $A(t)$ и мгновенную фазу $\varphi(t)$ сигнала $z(t)$:

$$A(t) = \sqrt{x_j^2(t) + c_j^2(t)},$$

$$\varphi(t) = \operatorname{arctg}(c_j(t)/x_j(t)).$$

Мгновенная частота же в этом случае может быть определена как первая производная от мгновенной фазы.

Итоговый результат преобразования Гильберта-Хуанга графически может быть представлен либо в виде набора двумерных графиков частоты от времени для каждой моды, либо сведен на трехмерный график путем совмещения мод.

Задачи и порядок выполнения работы

Для успешного выполнения работы необходимо:

1. Получить массив отсчётов исходного сигнала из заранее подготовленного звукового WAV-файла с записанным ЛЧМ-сигналом с девиацией частоты от f_0 до f_1 длительностью $8 - 15$ с[1].
2. Построить графики звукового сигнала во временной и частотной области.
3. Построить спектрограмму сигнала.
4. Выполнить эмпирическую модовую декомпозицию сигнала.
5. Построить графики эмпирических модовых функций.
6. Построить график зависимости частоты сигнала от времени. Сравнить полученный график со спектрограммой.

После выполнения экспериментальной части необходимо ответить на предложенные контрольные вопросы для закрепления пройденного материала и установления взаимосвязи между полученными результатами практических работ и теоретическими знаниями.

Результаты работы рекомендуется оформить в виде отчета, в котором должна содержаться следующая информация: цель работы; решённые в процессе

[1]Исходный wav-файл можно получить с помощью бесплатной утилиты LabChirp

её достижения задачи; основные математические выражения, использованные при решении задач; текст программы или схема моделирования, результаты моделирования в виде графиков и заключение, позволяющее сделать вывод о сопоставимости результатов практической работы с теоретическими сведениями.

Пример выполнения работы в среде MathWorks MATLAB

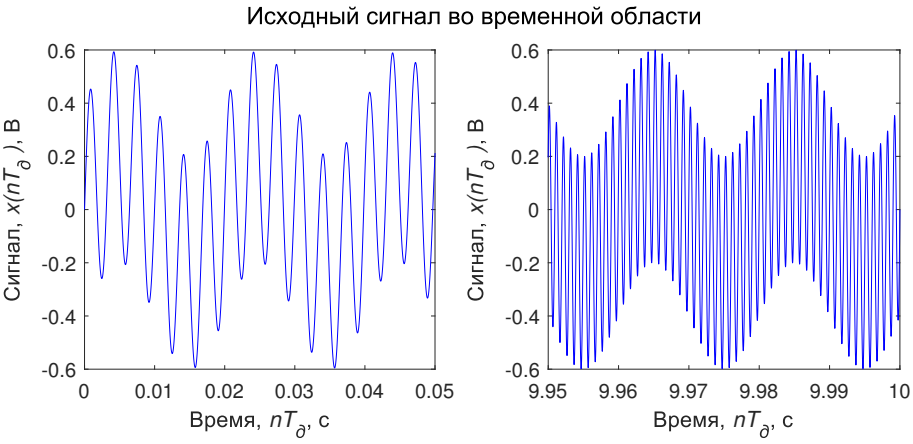
1 Анализ исходного сигнала

1.1 Инициализация и формирование значений основных параметров

```
clear all; % Очистка памяти
close all; % Закрытие всех окон с графиками
clc; % Очистка окна команд и сообщений
fontSize = 10; % Размер шрифта графиков
tColor = 'b'; % Цвет графиков во временной области
fColor = [1 0.4 0]; % Цвет графиков в частотной области
xlimit = 0.05; % Ограничение области определения на графике
```

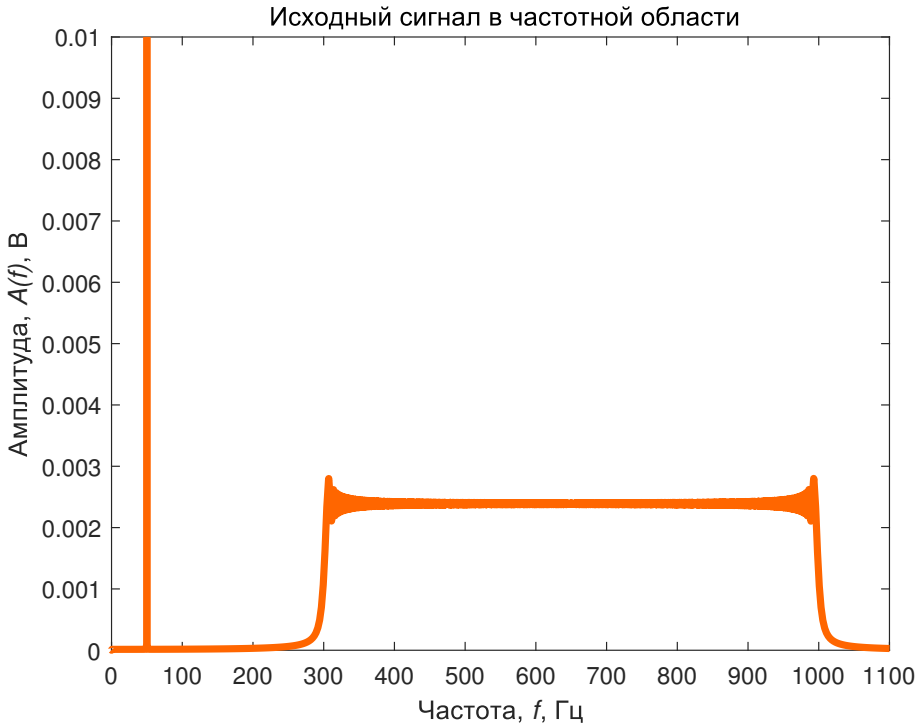
1.2 Чтение файла с мелодией

```
[data,rate] = audioread('complex.wav');
t = linspace(0, length(data)/rate, length(data))'; % Формирование области определения
figure('Renderer', 'painters', 'Position', [0 0 700 300]);
subplot(1,2,1);
plot(t, data, 'Color', tColor);
xlim([0 xlimit]); % Показать сигнал в начале мелодии
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, B'); % Надпись оси ординат
subplot(1,2,2);
plot(t, data, 'Color', tColor);
xlim([t(end)-xlimit t(end)]); % Показать сигнал в конце мелодии
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Сигнал,\it x(nT_д )\rm, B'); % Надпись оси ординат
sgtitle('Исходный сигнал во временной области');
```



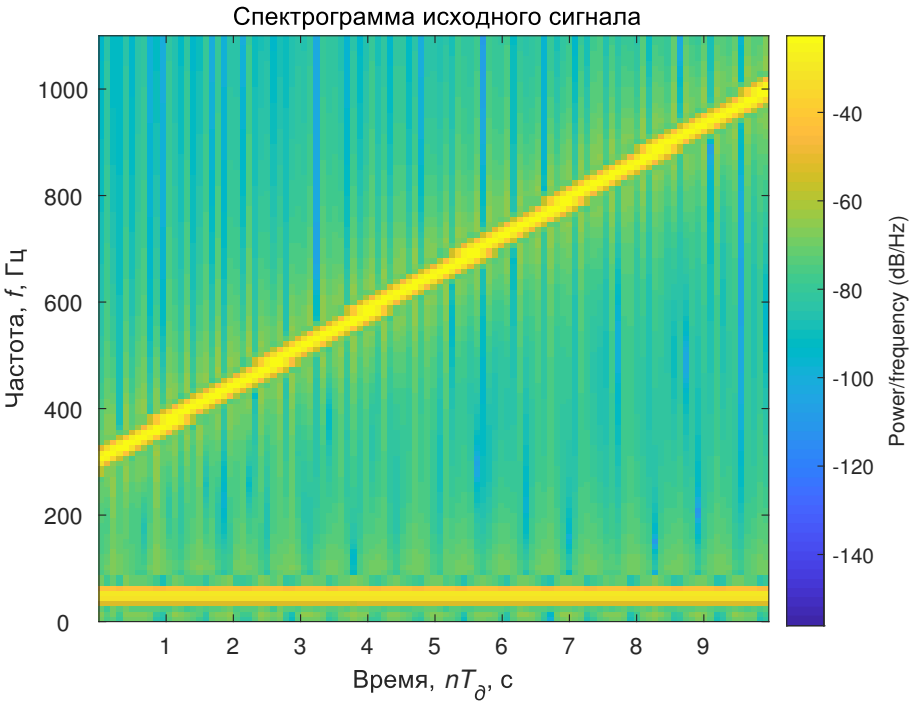
1.3 Построение амплитудного спектра сигнала

```
f_axis = linspace(0, rate, length(data)); % Формирование области определения
fdata = abs(fft(data)/length(data)); % Формирование значений спектра
figure; plot([-fliplr(f_axis(1:end/2)) f_axis(1:end/2)], fftshift(fdata),...
    'Color', fColor, 'LineWidth', 3);
xlim([0 1100]); % Ограничение области определения
ylim([0 0.01]);
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Исходный сигнал в частотной области'); % Заголовок
xlabel('Частота,\it f\rm, Гц'); % Надпись оси абсцисс
ylabel('Амплитуда,\it A(f)\rm, В'); % Надпись оси ординат
```



1.4 Построение спектрограммы исходного сигнала

```
spectrogram(data,4096,64,4096,rate,'yaxis');
ylim([0 1.1]); % Ограничение области определения в кГц
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
title('\rm Спектрограмма исходного сигнала'); % Заголовок
xlabel('Время,\it пТ_д\rm, с'); % Надпись оси абсцисс
ylabel('Частота,\it f\rm, Гц'); % Надпись оси ординат
yt = get(gca, 'YTick'); % Перевод единиц измерения частоты в Гц
set(gca, 'YTick', yt, 'YTickLabel', yt*1E+3);
```

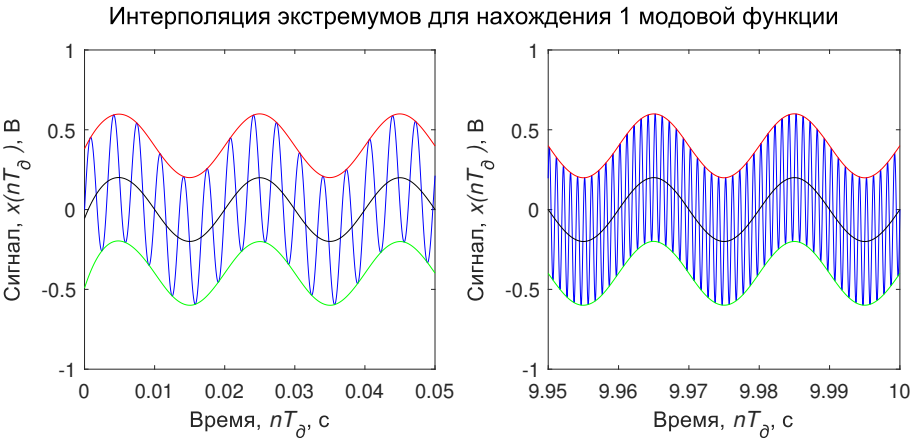


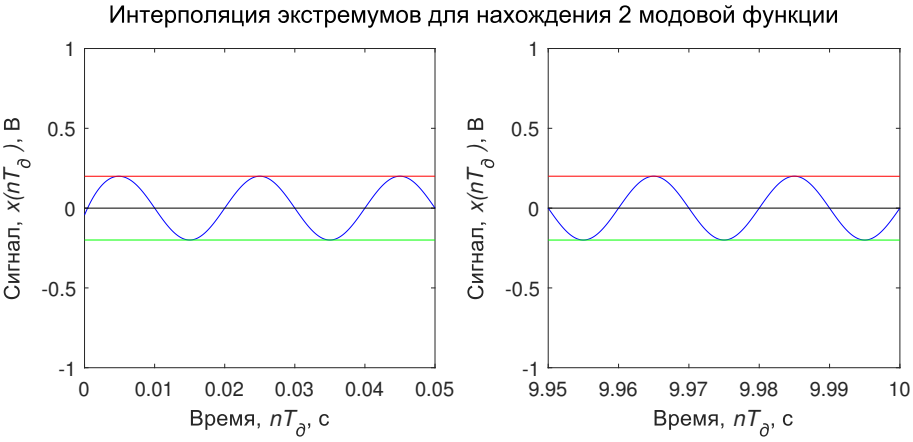
2 Преобразование Гильберта-Хуанга

2.1 Выполнение эмпирической модовой декомпозиции сигнала

```
c = zeros(length(data),5); % массив для пяти эмпирических модовых функций
r = zeros(length(data),5); % массив для пяти остатков
c_index = 1; % индекс эмпирической модовой функции
y = data; % y - раскладываемый сигнал
while 1
h_prev = y; % h_prev - предыдущее приближение модовой функции
h_index = 1; % номер приближения модовой функции
while 1
    [pks_max,locs] = findpeaks(h_prev);
    t_pks_max = t(locs); % Нахождение локальных максимумов
    [pks_min,locs] = findpeaks(-h_prev);
    pks_min = -pks_min;
    t_pks_min = t(locs); % Нахождение локальных минимумов
    polynom_max = spline(t_pks_max,pks_max,t); % Вычисление огибающих
    polynom_min = spline(t_pks_min,pks_min,t);
    m = (polynom_max + polynom_min)./2; % Вычисление функции средних значений
    if (h_index == 1) % Построить графики вычисленных функций для первого прибли
жения
    figure('Renderer', 'painters', 'Position', [0 0 700 300]);
    subplot(1,2,1);
```

```
    plot(t, h_prev, '-b', t, polynom_max, '-r', t, polynom_min, '-g', t, m, '-k'
);
    xlim([0 xlimit]); % Ограничение области определения
    ylim([-1 1]); % Ограничение области определения
    set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
    xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
    ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
    subplot(1,2,2);
    plot(t, h_prev, '-b', t, polynom_max, '-r', t, polynom_min, '-g', t, m, '-k'
);
    xlim([t(end)-xlimit t(end)]); % Ограничение области определения
    ylim([-1 1]); % Ограничение области определения
    set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
    xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
    ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
    sgtitle(sprintf('Интерполяция экстремумов для нахождения %d модовой функции'
,c_index));
    end
    h = h_prev - m; % Нахождение следующего приближения
    eps = sum((h_prev - h).^2)/sum(h_prev.^2);
    if(eps < 1e-6) % Выход из цикла при малой разности между двумя приближениями
        break;
    end
    h_prev = h; % Переход к расчёту следующего приближения
    h_index = h_index + 1;
end
c(:,c_index) = h; % Сохранить последнее приближение как модовую функцию
r(:,c_index) = y - c(:,c_index); % Вычисление остатка
y = r(:,c_index); % Остаток - следующий сигнал для разложения
eps2 = sum(r(:,c_index).^2)/length(r(:,c_index));
if(eps2 < 1e-6) % Выход из цикла при нулевом остатке
    break;
end
c_index = c_index + 1; % Переход к вычислению следующей модовой функции
end
```

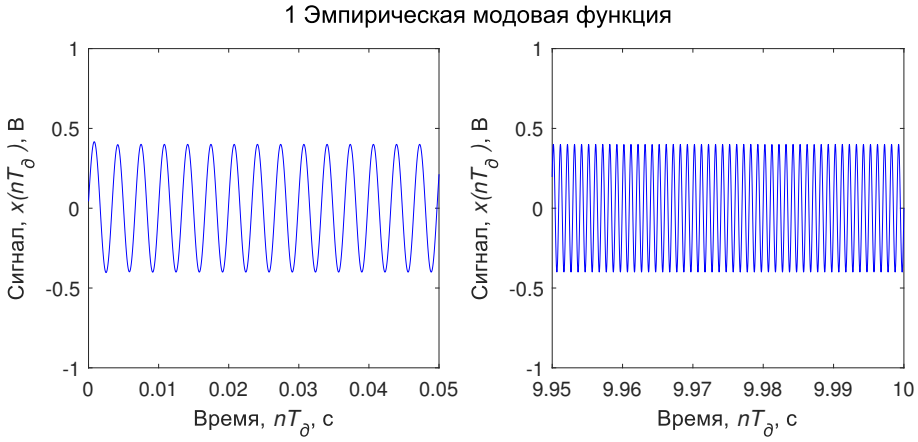


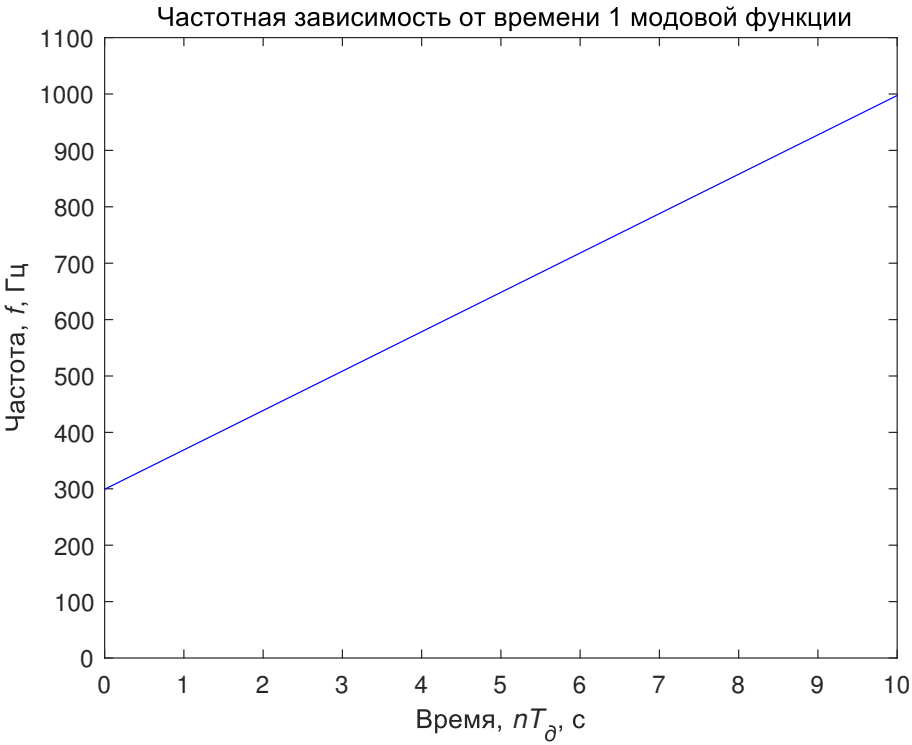


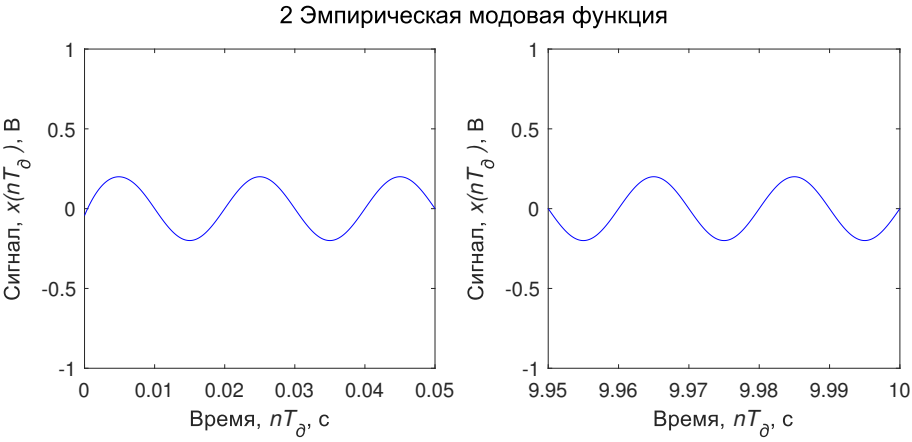
2.2 Построение графиков эмпирических модовых функций и зависимостей частоты от времени

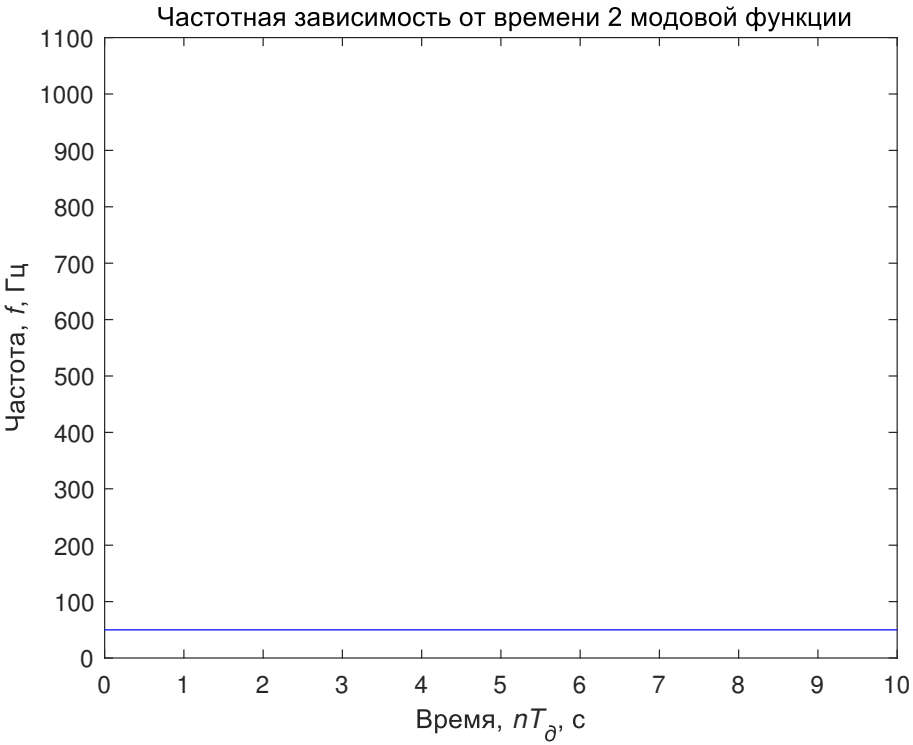
```
freq = zeros(length(data),5); % Значения частоты в зависимоти от времени
for index = 1:c_index % Отобразить каждую модовую функцию и зависимости частоты
от времени
    figure('Renderer', 'painters', 'Position', [0 0 700 300]);
    subplot(1,2,1);
    plot(t, c(:,index), '-b');
    xlim([0 xlimit]); % Ограничение области определения
    ylim([-1 1]); % Ограничение области определения
    set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
    xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
    ylabel('Сигнал,\it x(nT_д )\rm, B'); % Надпись оси ординат
    subplot(1,2,2);
```

```
    plot(t, c(:,index), '-b');
    xlim([t(end)-xlimit t(end)]); % Ограничение области определения
    ylim([-1 1]); % Ограничение области определения
    set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
    xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
    ylabel('Сигнал,\it x(nT_д )\rm, В'); % Надпись оси ординат
    sgtitle(sprintf('%d Эмпирическая модовая функция',index));
    % Нахождение зависимости частоты от времени
    hx = hilbert(c(:,index)); % Преобразование Гильберта
    phi = angle(hx); % Вычисление фазы
    phi2 = unwrap(phi);
    vector = linspace(0, length(c(:,index))-1, length(c(:,index)))';
    p = polyfit(vector,phi2,3); % Аппроксимация полиномом третьей степени
    dp = polyder(p); % Производная полинома
    freq(:,index) = polyval(dp, 0:length(c(:,index))-1).*7000; % Вычисление значений частоты от времени
    figure; plot(t, freq(:,index), 'Color', tColor);
    ylim([0 1100]); % Ограничение области определения
    set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
    title(sprintf('\\rm Частотная зависимость от времени %d модовой функции', index)); % Заголовок
    xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
    ylabel('Частота,\it f\rm, Гц'); % Надпись оси ординат
end
```




Частотная зависимость от времени 1 модовой функции
Частота, f, Гц
Время, nT∂, с
0
100
200
300
400
500
600
700
800
900
1000
1100
0
1
2
3
4
5
6
7
8
9
10

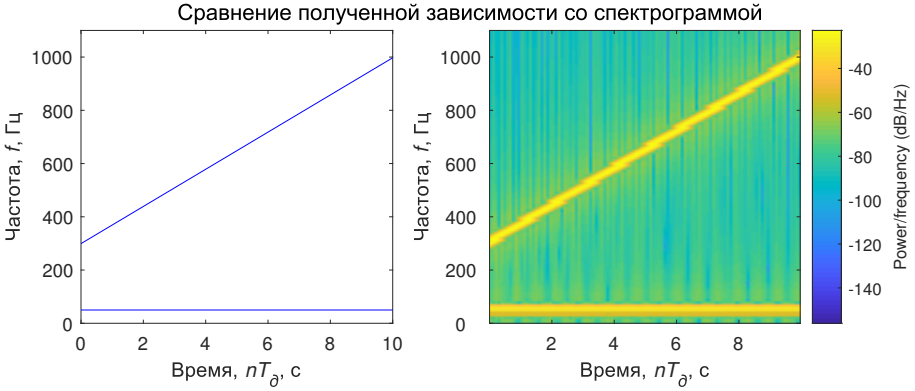

2 Эмпирическая модовая функция
Сигнал, x(nT∂), В
Время, nT∂, с
1
0.5
0
-0.5
-1
0
0.01
0.02
0.03
0.04
0.05
9.95
9.96
9.97
9.98
9.99
10



3 Сравнение зависимости частоты сигнала от времени со спектрограммой

```
figure('Renderer', 'painters', 'Position', [0 0 700 300]);
subplot('Position',[0.1 0.2 0.32 0.7]);
plot(t, freq(:,1), 'Color', tColor); % Построение зависимости для 1 модовой функ
ции
hold on; % Одновременное построение нескольких функций на одном графике
for index = 2:c_index % Построение зависимостей остальных модовых функций
    plot(t, freq(:,index), 'Color', tColor);
end
ylim([0 1100]); % Ограничение области определения
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Частота,\it f\rm, Гц'); % Надпись оси ординат
hold off;
subplot('Position',[0.52 0.2 0.4 0.7]);
spectrogram(data,4096,64,4096,rate,'yaxis'); % Построение спектрограммы
ylim([0 1.1]); % Ограничение области определения в кГц
set(get(gcf, 'CurrentAxes'), 'FontSize', fontSize); % Изменение шрифта
xlabel('Время,\it nT_д\rm, c'); % Надпись оси абсцисс
ylabel('Частота,\it f\rm, Гц'); % Надпись оси ординат
yt = get(gca, 'YTick'); % Перевод единиц измерения частоты в Гц
```

```
set(gca, 'YTick', yt, 'YTickLabel', yt*1E+3);
sgtitle('Сравнение полученной зависимости со спектрограммой');
```



Контрольные вопросы

1. Чем отличаются области применения преобразования Фурье и преобразования Гильберта-Хуанга?
2. Приведите аналитическое выражение для преобразования Гильберта-Хуанга.
3. Что такое эмпирическая модовая декомпозиция и для чего она нужна?
4. Опишите алгоритм получения эмпирических модовых функций.
5. Каким выражением определяется мгновенная частота при использовании преобразования Гильберта-Хуанга?

Список литературы

[1] Сюзев В.В. *Основы теории цифровой обработки сигналов. Учебное пособие.* Издательство «РТСофт», 2014, с. 752.

[2] Розанов И.А. Сотников А.А. Ким Т.А. *Имитационное моделирование сигналов информационно-управляющих систем: практикум.* Издательство «Наукоемкие технологии», 2022, с. 147.

[3] Юкио Сато. *Без паники! Цифровая обработка сигналов.* М.: Додэка-XXI, 2010, с. 176.

[4] Харкевич А.А. *Основы радиотехники.* М.: Связьиздат, 1962.

[5] Охрименко А.Е. *Основы извлечения, обработки и передачи информации. (В 6 частях).* Минск, МРТИ, 2004.

[6] Фомин С.В. Колмогоров А.Н. *Элементы теории функций и функционального анализа.* Москва, Наука, 1976, с. 496.

[7] Исаков В.Н. *Радиотехнические цепи и сигналы. Ч. 2: методические указания по выполнению лабораторных работ по курсу «Радиотехнические цепи и сигналы».* Москва : МГТУ МИРЭА, 2014, с. 28.

[8] Генри Нуссбаумер. *Быстрое преобразование Фурье и алгоритмы вычисления сверток.* Радио и связь, 1985.

[9] Gerald Goertzel и др. “An algorithm for the evaluation of finite trigonometric series”. в: *The American Mathematical Monthly* 65.1 (1958), с. 34—35.

[10] Окунев Ю.Б. *Цифровая передача информации фазомодулированными сигналами.* М: Радио и связь, 1991, с. 296.

[11] Байков В.Д. и Смолов В.Б. *Аппаратурная реализация элементарных функций в ЦВМ.* Л.: ЛГУ, 1975, с. 96.

[12] Шахматов А.В. “Алгоритм цифровой демодуляции фазоманипулированных сигналов с произвольным индексом модуляции, ориентированный на использоавение цифрового синтезатора частоты”. в: *Доклады ТУСУР* №2(24) (2011), с. 74—77.

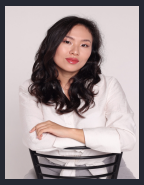
[13] Norden E Huang и др. “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis”. в: *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences* 454.1971 (1998), с. 903—995.

[14] Давыдов В.А. Давыдов А.В. “Уменьшение краевых эффектов при выполнении эмпирической модовой декомпозиции сигналов преобразования Гильберта-Хуанга”. в: *Актуальные инновационные исследования: наука и практика* 1 (2011), с. 11—11.

ОБ АВТОРАХ

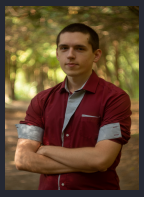
СОТНИКОВ АЛЕКСЕЙ АЛЕКСАНДРОВИЧ

КАНДИДАТ ТЕХНИЧЕСКИХ НАУК, ДОЦЕНТ КАФЕДРЫ «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ», НАЧАЛЬНИК СЕКТОРА НАУЧНО-ИССЛЕДОВАТЕЛЬСКОГО ИНСТИТУТА ИНФОРМАТИКИ И СИСТЕМ УПРАВЛЕНИЯ МГТУ ИМ. Н.Э. БАУМАНА



КИМ ТАМАРА АЛЕКСАНДРОВНА

АСПИРАНТКА И АССИСТЕНТ КАФЕДРЫ «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ», ПРОГРАММИСТ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОГО ИНСТИТУТА ИНФОРМАТИКИ И СИСТЕМ УПРАВЛЕНИЯ МГТУ ИМ. Н.Э. БАУМАНА



АРЕЩЕНКОВ ДМИТРИЙ АЛЕКСАНДРОВИЧ

МАГИСТРАНТ КАФЕДРЫ «КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ» МГТУ ИМ. Н.Э. БАУМАНА

ИЗДАТЕЛЬСТВО
«НАУКОЕМКИЕ ТЕХНОЛОГИИ»

САНКТ-ПЕТЕРБУРГ, 2022