



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ДГТУ)**

Кафедра «Кибербезопасность информационных систем»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по дисциплине

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ И ИЗОБРАЖЕНИЙ

Ростов-на-Дону
ДГТУ
2026

Составитель: А.А. Ляпин

Методические указания для выполнения контрольной работы по дисциплине «Цифровая обработка сигналов и изображений» для обучающихся заочной формы / сост. А.А. Ляпин – Ростов-на-Дону: Донской государственный технический университет, 2026. – 61с.

Содержат краткую теорию по изучаемым вопросам, примеры применения методов обработки сигналов и изображений применительно к проблемам машинного обучения. Приведены варианты индивидуальных заданий.

Предназначены для обучающихся магистратуры направления подготовки 09.04.02 заочной формы обучения

УДК 004.02

Ответственный за выпуск: профессор кафедры «Кибербезопасность информационных систем» д-р ф.-м. наук, профессор А.А. Ляпин

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

© Донской государственный технический университет, 2026

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ.....	4
1. ОСНОВЫ ПРОГРАММИРОВАНИЯ В MATLAB.....	5
2. РАЗЛОЖЕНИЕ ДИСКРЕТИЗИРОВАННЫХ СИГНАЛОВ В ДЕЙСТВИТЕЛЬНЫЙ И КОМПЛЕКСНЫЙ РЯД ФУРЬЕ.....	20
3. ЦИФРОВАЯ ФИЛЬТРАЦИЯ ШУМОВ В СРЕДЕ MATLAB.....	31
4. БАЗОВЫЕ СРЕДСТВА ФИЛЬТРАЦИИ ШУМОВ НА ИЗОБРАЖЕНИЯХ.....	50
Литература	56
ПРИЛОЖЕНИЕ. Форма отчета по контрольной работе.....	57

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

Задания контрольной работы выполняются по индивидуальным вариантам. Вариант соответствует последней цифре в зачетке обучающегося: цифра 1 – вариант 1, ..., цифра 0 – вариант 10.

Отчет по контрольной работе оформляется в виде электронного документа MS Word. Содержит:

- титульный лист;
- постановку задачи, незначительные сведения из теории, код программы или скриншоты выполнения в рекомендованной программе на компьютере, результаты вычислений и сформулированные выводы по каждой задаче;
- список использованной литературы, включая интернет-источники.

Отчет распечатывается и предоставляется на кафедру «Кибербезопасность информационных систем» для проверки и оценивания.

1. Основы программирования в MATLAB

Целью лабораторной работы является изучение методики проектирования программ цифровой обработки сигналов (ЦОС): моделирования сигналов и нормально распределенного и белого шума, программирования операций математической обработки, операций графического вывода данных, проектирования средств графического пользовательского интерфейса.

Теоретические основы

Программные средства систем ЦОС обычно создаются на языке C/C++, так как в этом языке предусмотрены средства, обычно необходимые при аппаратной реализации на базе программируемых логических интегральных схем (ПЛИС) и сигнальных процессоров: удобные средства работы с битами, логические операции, средства работы с аппаратными прерываниями и др. Кроме того, компилятор C обеспечивает формирование исполняемой программы, близкой по скорости исполнения к программе на ассемблере.

При разработке систем жесткого реального времени на базе ПЛИС и быстродействующих сигнальных процессоров необходимо производить отработку алгоритмов ЦОС на универсальных средствах (с помощью компьютерных программ) используя низкоуровневое программирование без библиотечных функций (на C/C++) с тем, чтобы на следующем этапе можно было реализовать разработанные алгоритмы ЦОС на ПЛИС и сигнальных процессорах. Кроме того, как известно, ЦОС должна быть выполнена в течение критического срока обслуживания и это так же важно, как и корректность алгоритма ЦОС. Невыполнение задачи реального времени в течение критического срока обслуживания равносильно невыполнению задачи в целом. Поэтому очень важно, чтобы на втором этапе разработки отработка алгоритмов ЦОС производилась с использованием соответствующей аппаратуры и скорость выполнения операций ЦОС контролировалась, например, с использованием системного таймера компьютера.

Конечной целью разработки и исследования различных алгоритмов ЦОС является их практическая реализация в устройствах на базе ПЛИС и сигнальных процессоров, таких, например, как модуль ЦОС SAMC-401, содержащий ПЛИС серии Virtex-4, в которой интегрированы два процессора

Power PC с тактовой частотой до 450 МГц, и сигнальный процессор Texas Instruments TMS320C6455, работающий на частоте 1.2ГГц с submodule АЦП SAMC-ADC, реализующий аналого-цифровое преобразование 12/14 бит с предельной частотой тактирования до 210 МГц.

Основные стадии разработки алгоритмов ЦОС:

1. Высокоуровневая программная (MATLAB, LabView).
2. Низкоуровневая программная (C/C++).
3. Аппаратно-программная (C/C++, LabWindows/CVI).
4. Аппаратная (VHDL, C/C++).

На первом этапе разработки наиболее совершенным средством является MATLAB, так как он содержит библиотеки функций для сложных видов математической обработки, таких как быстрое преобразование Фурье, цифровая фильтрация, корреляционная обработка, преобразование Гильберта и др., библиотеки программ для создания объектов графического пользовательского интерфейса (панелей, кнопок управления, окон цифрового ввода/вывода, окон графического вывода и др.).

Программные средства, создаваемые для работы в среде MATLAB, содержат две составляющие:

1. Собственно программу (mat-файл) на языке C, в которой содержатся необходимые функции математической обработки, отображения таблиц и графиков. При ее создании следует пользоваться описаниями функций библиотек математической обработки и графического пользовательского интерфейса.
2. Файлы ресурсов графического пользовательского интерфейса <имя программы>.fig.

Создаваемая пользователем прикладная программа представляет проект, содержащий два файла:

- файл основной программы <имя>.mat;
- файл макета <имя>.fig.

Программа на языке C может быть написана и редактироваться пользователем. Файлы макета <имя>.fig создаются автоматически при создании и редактировании пользователем графических панелей. Эти файлы нельзя редактировать!

Создание средств графического пользовательского интерфейса в традиционных системах программирования, таких как Visual C++ возможно, но достаточно сложно, так как для этого необходимо создание большого количества нестандартных графических объектов.

Создание прикладной программы ЦОС в среде MATLAB

1. Создайте панель интерфейса пользователя: **File/New/GUI/Blank GUI**. В результате появится всплывающее диалоговое окно `untitled.fig`
2. Разместите элементы управления (кнопки управления `PushButton`, окна ввода/вывода текста `EditText`, окна графического вывода `Axes` и др.), например, так, как показано на рис. 1.1.

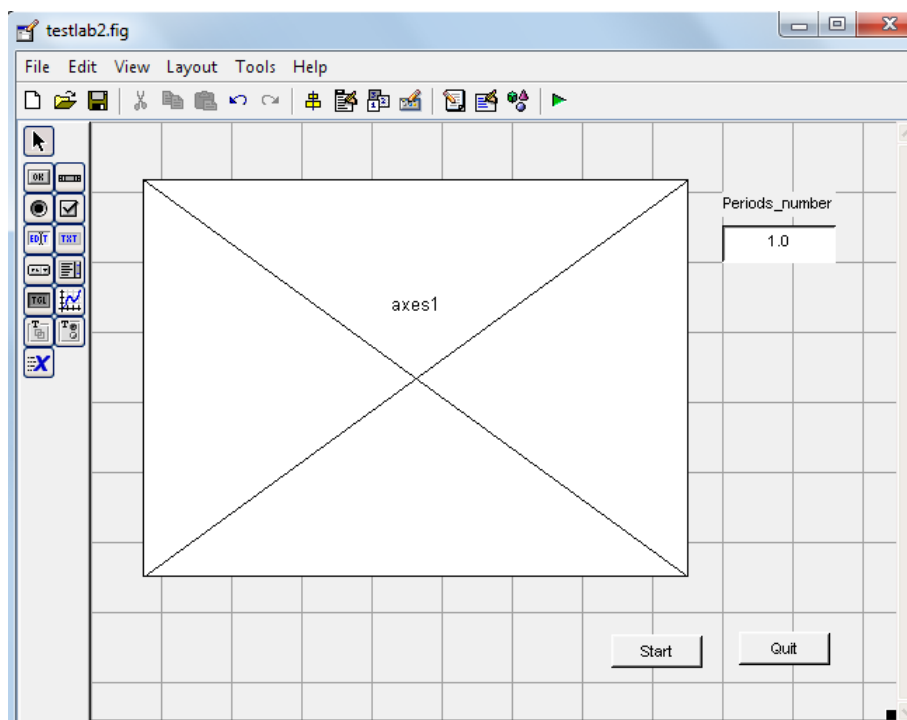


Рис. 1.1

и задать их атрибуты. Панель атрибутов элемента управления вызывается двойным щелчком левой кнопки мыши на изображении элемента. Основные атрибуты кнопки управления и окна ввода/вывода текста – `Callback`, `CreateFcn` и `tag`. Например, при задании атрибуту `tag` окна ввода/вывода текста значения `freq` имя окна будет `freq`. Такое же имя может быть дано и переменной, значение которой читается из этого окна.

Атрибуты Callback, CreateFcn создаются автоматически после задания *tag*.

3. Сохраните untitled.fig в файле с именем <имя>.fig. В результате будет не только сохранен файл интерфейса пользователя, но и создан и сохранен в файле <имя>.m «скелет» программы.

4. Создайте содержательную часть программы в блоке функции, вызываемой, например, нажатием кнопки управления (см. Приложение 2).

В итоге будут созданы два файла: файл программы с расширением .m и файл GUI с расширением .fig. Для внесения изменений в интерфейс пользователя нужно вызвать файл командой >>guide ('<полное имя файла, включая путь>.fig').

Загрузка и запуск MATLAB

Для загрузки и запуска MATLAB в среде ОС Windows XP/7 нужно вызвать из меню «Пуск»:

Programs/MATLAB

Создание программы

Создайте файл макета.

а) **File/New/GUI/Blank GUI ...**

б) сохраните созданный файл макета

File/Save Untitled.fig As.../{имя файла}

В результате на экране дисплея появится изображение панели GUI. Размеры и место расположения панели на экране можно установить с помощью мыши. Далее нужно установить атрибуты панели. Для этого перейти в режим редактирования панели двойным щелчком мыши и во всплывающей панели задать:

- Name - идентификатор панели, который определяет имя m-файла и fig-файла.
- и другие.

4. Создайте элементы GUI (графические окна вывода, текстовые окна ввода/вывода, командные кнопки и т.д.) из библиотеки элементов и разместить их на созданной панели.

Объекты GUI выбираются из меню и размещаются на экране методом "drag and drop" с помощью "мыши".

MATLAB предоставляет широкий выбор объектов:

- Edit text - окно ввода/вывода текста;
- Static Text – окно вывода текстового сообщения;
- Push Button – кнопка с двумя состояниями;
- Toggle Button – кнопка с двумя состояниями (включено/выключено);
- Check Box – флажок;
- Radio Button – аналог Option Button;
- Pop-up-menu – меню;
- Slider – линейка прокрутки;
- List box – поле со списком;
- Axis1 - окно вывода массивов данных в графической форме.

При выполнении лабораторной работы рекомендуется использовать следующие элементы управления и ввода/вывода: Edit text, Axis1, Pushbutton.

4.1 Задание атрибутов элементов GUI.

Сделайте двойной щелчок левой клавишей мыши на объекте GUI, далее во всплывающей панели задать его атрибуты, в первую очередь tag.

Далее нужно сохранить сделанные установки.

File/Save

/Save As.../{имя файла}

5. Создайте программный код. Создание fig-файла производится автоматически при сохранении GUI-образа. Одновременно создается m-файл.

В созданном программном коде m-файла объектам GUI будет соответствовать "программная оболочка" функций на языке C++. Эти функции (за исключением функции работы таймера) будут запускаться на исполнение пользователем с панели GUI. В приложении 1 приведен пример

созданной таким путем «программной оболочки».

Далее в эту оболочку вносится программный код прикладной программы на языке C++.

Если производится редактирование ранее уже созданного программного кода, например, при добавлении какого-либо объекта GUI, то для добавления в ранее созданный программный код изменений, связанных с добавленными элементами GUI нужно открыть файл <>.fig для редактирования командой:

>>guide ('<полное имя файла, включая путь>.fig')

в окне Command Window. Путь к файлу можно не указывать, если поместить его в папку MATLAB.

6. Добавьте в программу функцию выхода.

Для этого в уже сгенерированную по п.5 функцию выхода программы необходимо внести программный код:

```
fclose('all');  
close ('all');
```

7. Добавьте в функцию, вызываемую кнопкой на панели GUI (в приведенном примере - Start), строки обращения к элементам ввода/вывода GUI, текст программы. Ниже приведен пример программирования ввода текста из окна ввода/вывода (TextBox) с преобразованием в число типа double, вывода значения с преобразованием в текст, чтения номера элемента из поля со списком (ComboBox), вывода числового значения в TextBox:

```
periods_number=str2double(get(handles.periods_number,'string'));  
set(handles.chastotan,'String',fr_int);  
regim=get(handles.regim,'Value');  
set(handles.lampa,'Value',1);
```

Примечания:

1. Элементы поля со списком нумеруются, начиная с 1. Так же в MATLAB нумеруются и элементы массивов.

2. В атрибутах окна, предназначенного для вывода текстовых сообщений, задайте атрибут Style –Edit.
3. Для имитации сигнального индикатора можно использовать элемент radiobutton.

Описания типов переменных не обязательны, но возможны, если требуется. В случае необходимости передавать значения параметров из одной функции в другую нужно использовать описания глобальных переменных, например:

```
global N
```

В результате будет получен полный программный код прикладной программы. Простой пример такой программы генерации и графического отображения массива случайных чисел приведен ниже. Количество периодов сигнала задается переменной freq, получаемой с элемента GUI:

```
freq=str2double(get(handles.freq,'string'));  
for i=1:1000  
    y(i)=sin((2*pi*i*freq)/1000.0);  
end  
i=1:1000;  
plot(i,y);
```

Редактирование программного кода, в том числе строк обращения к объектам GUI производится точно так же, как в любом C/C++.

10. Запустите программу.

Debug/Run

Если компилятор найдет ошибки, то устранить ошибки и повторить предыдущее действие.

Программа работы

1. Изучите состав и функции библиотек программ MATLAB для работы с аппаратурой, математической обработки, создания средств графического пользовательского интерфейса.
2. Разработайте программу генерации гармонического сигнала.

3. Разработайте программу генерации стандартных сигналов: гармонического, пилообразного, треугольного, прямоугольного с нормально распределенным и белым шумом. Предусмотрите средства выбора вида сигнала, амплитуды сигнала, вида и уровня шума.

Указания к выполнению лабораторной работы

1. При выполнении п. 1 Программы изучите основы программирования в MATLAB по описанию к лабораторной работе и пользуясь справочной системой.
2. При выполнении п.2 Программы создайте графический пользовательский интерфейс и программу генерации синусоидального сигнала. Количество периодов сигнала сделайте регулируемым.
3. При выполнении п.3 Программы разработайте прикладную программу генерации различных стандартных сигналов:

- ◆ гармонического (синусоидального);
- ◆ пилообразного;
- ◆ треугольного;
- ◆ прямоугольных импульсов.

без шума и с нормально распределенным и белым шумом. Выбор вида сигнала, амплитуду, частоту сигналов и среднеквадратическую величину шума сделать регулируемыми.

- 3.1. Для генерации нормально распределенного и белого шума используйте функции `randn` и `wgn`. Пример фрагмента программы генерации модельного сигнала с шумом приведен ниже.

```
%Генерация нормально распределенного и белого шума
%noise=randn(points_number);%нормально распределенный
noise=wgn(points_number,1,0);%белый Гауссов шум
for i=1:points_number %генерация модельного сигнала с шумом
    x(i)=sin(2*3.14*kp*i/kt)+noise_sko*noise(i);
end
```

Здесь `points_number` – количество элементов массива (количество значений сигнала), `noise_sko` – среднеквадратическое значение шума.

3.2. На панели графического пользовательского интерфейса предусмотрите (см. рис. 2):

- ◆ переключатель выбора вида сигнала;
- ◆ элемент ввода уровня шума;
- ◆ элемент ввода количества точек за период генерируемого сигнала;
- ◆ окно графического вывода генерируемого сигнала.

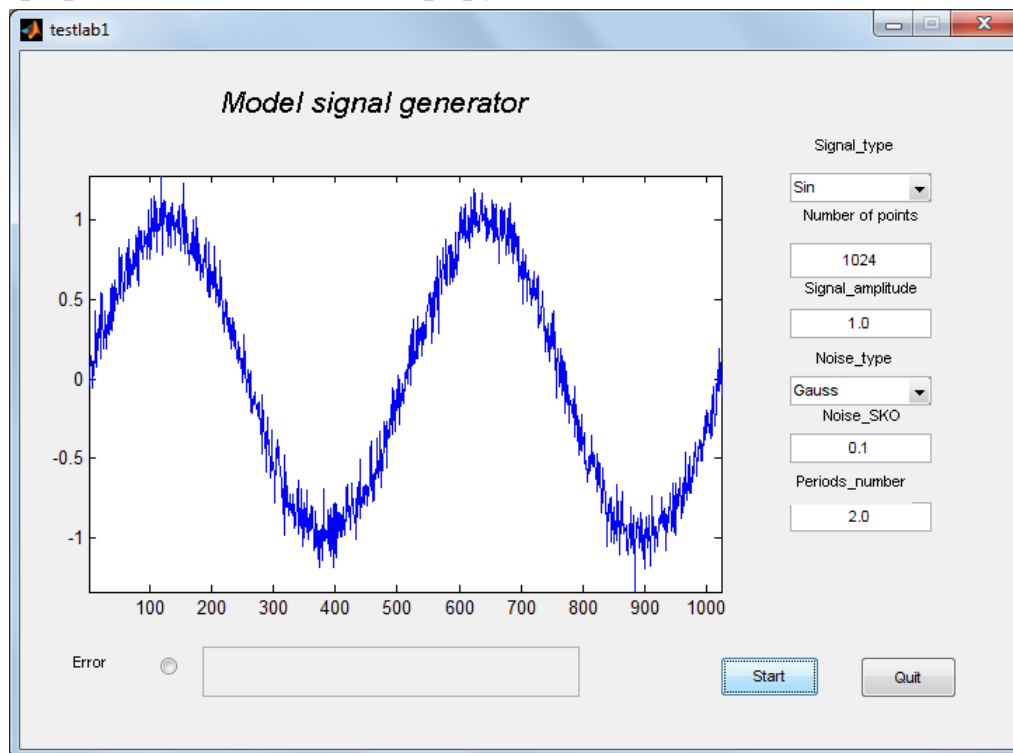


Рис. 1.2

3.3. Дополните программу, предусмотрев в ней возможность накопления (суммирования с усреднением) генерируемых сигналов.

На панели графического пользовательского интерфейса дополнительно предусмотрите переключатель задания количества накоплений.

Содержание отчета

1. Задание к работе.
2. Текст программы с комментариями.
3. Вид панели интерфейса пользователя при генерации различных сигналов с различным шумом.

Задание является общим для всех студентов группы.

Приложение 1

Пример программного кода, генерируемого автоматически после создания панели и объектов GUI в среде MATLAB

```
%Служебная часть М-файла, создаваемая автоматически после сохранения в
файле
%панели интерфейса пользователя <имя>.fig
%В данном примере панель интерфейса пользователя содержит кнопку
управления
%PushButton, окно ввода/вывода текста EditText (tag – КР), окно графического
%вывода Axes (tag – axis1)

function varargout = testlab2(varargin)
% TESTLAB2 M-file for testlab2.fig
%   TESTLAB2, by itself, creates a new TESTLAB2 or raises the existing
%   singleton*.
%
%   H = TESTLAB2 returns the handle to a new TESTLAB2 or the handle to
%   the existing singleton*.
%
%   TESTLAB2('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in TESTLAB2.M with the given input
%   arguments.
%
%   TESTLAB2('Property','Value',...) creates a new TESTLAB2 or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before testlab2_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to testlab2_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

```

% Edit the above text to modify the response to help testlab2

% Last Modified by GUIDE v2.5 03-Feb-2012 21:54:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @testlab2_OpeningFcn, ...
    'gui_OutputFcn', @testlab2_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before testlab2 is made visible.
function testlab2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to testlab2 (see VARARGIN)

% Choose default command line output for testlab2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```

% UIWAIT makes testlab2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = testlab2_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function KP_Callback(hObject, eventdata, handles)
% hObject handle to KP (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of KP as text
% str2double(get(hObject,'String')) returns contents of KP as a double

% --- Executes during object creation, after setting all properties.
function KP_CreateFcn(hObject, eventdata, handles)
% hObject handle to KP (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)

```



```

% hObject    handle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
KP=str2double(get(handles.KP,'string'));
for i=1:1000
    y(i)=sin((2*pi*i*KP)/1000.0);
end
i=1:1000;
plot(i,y);

% --- Executes on button press in Quit.
function Quit_Callback(hObject, eventdata, handles)
% hObject    handle to Quit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fclose('all');
close ('all');

```

Пример программного кода в функции Start

```
% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject    handle to Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
KP=str2double(get(handles.KP,'string'));
for i=1:1000
    y(i)=sin(2*pi*i*KP./1000.0);%KP – кол-во периодов
end
i=1:1000
plot(i,y);
```

После запуска программы Debug/Run диалоговое окно программы с результатами выполнения будет выглядеть, как показано на рис.1.3.

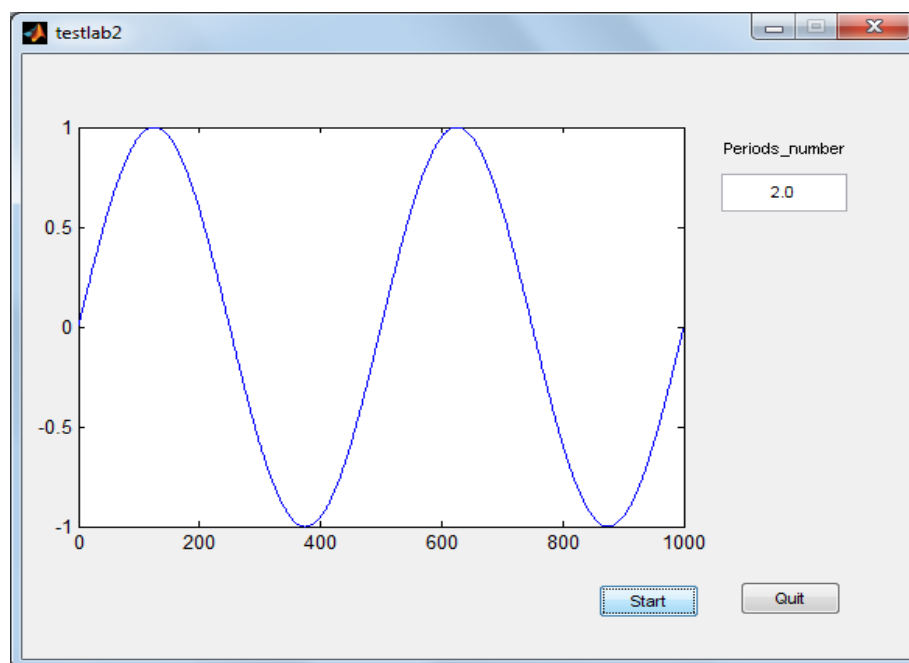


Рис. 1.3

Приложение 3. Построение и оформление графиков в MATLAB

```
i=1:N;  
y=A*sin(6.28*KP*i/N); %создание N значений KP периодов модельного sin  
%сигнала,  
figure %создание окна графического вывода  
plot(i,y(1:50));%вывод графика y, кол-во точек – 50.  
plot(i,y) ;% вывод графика y (все N точек)  
plot (i,2*y(1:N),'r-');%отображение графика y линией красного цвета  
semilogx(i,abs(y(1:200))),grid; %то же, что и plot, но в логарифмическом  
% масштабе по X  
loglog(i,abs(y(1:200))), % в логарифмическом масштабе по X и Y  
stem(A(1:K)); %вывод графика дискретной последовательности данных,  
%например, коэффициентов ряда Фурье  
[C,h] = contour(X,Y,Z);%построение контурной карты (сетки изолиний)  
hist(<имя массива данных>,<кол-во столбцов>);%построение гистограммы  
grid on; %отображение линий координатной сетки  
title('сигнал до фильтра');%заголовок графика  
xlabel('номер отсчета'); % подпись по оси X  
ylabel('амплитуда'); % подпись по оси Y  
legend('до фильтра');%подпись легенды  
axis tight; %диапазон X и Y по осям точно соответствует Xmax и Ymax  
%(автомасштабирование)  
hold on; % «удержание» окна вывода для следующего графика  
hold off; % отмена «удержания»  
close all; % закрытие всех открытых окон графического вывода  
clear; %очистка Workspace  
clc; %очистка Command Window
```

2.Разложение дискретизированных сигналов в действительный и комплексный ряд Фурье

Целью лабораторной работы является изучение методики разработки программ разложения дискретизированных сигналов в действительный и комплексный ряд Фурье на интервалах $[-\pi, \pi]$ и $[-\frac{T}{2}, \frac{T}{2}]$ с конечным числом членов.

Задание к работе

Имеется дискретизированная функция в виде числового массива. Требуется спроектировать на внутреннем языке MATLAB программу цифровой обработки данных, реализующую разложение этого сигнала в ряд Фурье с конечным числом членов и исследовать зависимость точности представления этой функции с помощью ряда Фурье от числа членов разложения и от шага дискретизации исходного непрерывного сигнала.

Теоретические основы

Известно, что любую функцию $f(t)$ можно представить в виде:

$$f(t) = c_0\varphi_0(t) + c_1\varphi_1(t) + c_2\varphi_2(t) + \dots + c_k\varphi_k(t) + \dots = \sum_{k=0}^{\infty} c_k\varphi_k(t)$$

где φ_k - ортонормированные функции.

Коэффициенты c_k вычисляются по формуле:

$$c_k = \langle f(t), \varphi_k(t) \rangle = \frac{1}{b-a} \int_a^b f(t) * \varphi_k(t) dt$$

Условие ортонормированности выполняется, если скалярное произведение любых двух функций, входящих в набор, равно нулю, а норма любой функции равна единице:

$$\langle \varphi_m(t), \varphi_n(t) \rangle = \frac{1}{b-a} \int_a^b \varphi_m(t) * \varphi_n(t) dt = 0, \quad m \neq n$$

$$\|\varphi_m(t)\| = \langle \varphi_m(t), \varphi_m(t) \rangle = \frac{1}{b-a} \int_a^b \varphi_m^2(t) dt = 1$$

Разложение непрерывной функции на интервале $[-T/2, T/2]$ в действительный ряд Фурье

До этого момента мы рассматривали функцию переменной t на отрезке $[-\pi, \pi]$. В случае периодического сигнала с периодом 2π мы брали этот интервал за основной. В общем случае периодического сигнала с периодом T при разложении в ряд Фурье мы должны использовать интервал $[-T/2, T/2]$. Если интервал $[-\pi, \pi]$ расширить (или сократить) до интервала $[-T/2, T/2]$, то и период первой гармоники увеличится (или уменьшится) от 2π до T . Поскольку кратность этого преобразования равна $(T/2) * \pi$, то составляющие первой гармоники примут вид:

$$\cos \frac{2\pi}{T} t \quad \sin \frac{2\pi}{T} t.$$

Для составляющих k -й гармоники можно записать:

$$\cos \frac{2\pi}{T} kt \quad \sin \frac{2\pi}{T} kt.$$

Следовательно, если функцию $f(t)$ разложить в ряд Фурье на интервале $[-T/2, T/2]$, получим:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left\{ a_k \cos \left(\frac{2\pi}{T} kt \right) + b_k \sin \left(\frac{2\pi}{T} kt \right) \right\}.$$

Если обозначить угловую частоту через ω_0 , то поскольку $\omega_0 = \frac{2\pi}{T}$, последнее выражение можно записать и в таком виде:

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \{ a_k \cos \omega_0 kt + b_k \sin \omega_0 kt \}$$

В соотношении, определяющем коэффициенты Фурье на отрезке $[-\pi, \pi]$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos kt dt.$$

произведем замену переменной

$$t \rightarrow \omega_0 t,$$

а также замену отрезка, на котором берется интеграл

$$[-\pi, \pi] \rightarrow [-T/2, T/2].$$

Оставив функцию $f(t)$ без изменения, получим

$$a_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos \omega_0 k t dt \quad (k = 0, 1, 2, \dots)$$

Аналогичным образом выводится следующее соотношение:

$$b_k = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin \omega_0 k t dt \quad (k = 1, 2, \dots).$$

Разложение функции, представленной в дискретизированном виде, в действительный ряд Фурье на интервале $[-T, T]$

В дискретизированном виде (т.е. в виде набора дискретных значений или, что то же, в виде числового массива, содержащего N значений) функция $f(t)$ на интервале $[-T, T]$ будет иметь вид:

$$f(t) \rightarrow f\left(\frac{2Ti}{N}\right), \quad i = \left[-\frac{N}{2}, \frac{N}{2}\right]$$

где N – количество дискретных значений сигнала. При $n \rightarrow \infty$ дискретизированная функция будет приближаться к непрерывной $f(t)$.

Разложение в ряд Фурье будет иметь приведенный ранее вид:

а коэффициенты α_0, α_k и β_k :

$$\alpha_0 = \left\langle f\left(\frac{2Ti}{N}\right), 1 \right\rangle = \frac{1}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right);$$

$$\alpha_k = \left\langle f\left(\frac{2Ti}{N}\right), \sqrt{2} \cos\left(\frac{2\pi k}{N}\right) \right\rangle = \frac{\sqrt{2}}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right) \cos\left(\frac{k * 2\pi i}{N}\right);$$

$$\beta_k = \left\langle f\left(\frac{2Ti}{N}\right), \sqrt{2} \sin\left(\frac{2\pi k}{N}\right) \right\rangle = \frac{\sqrt{2}}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right) \sin\left(\frac{k * 2\pi i}{N}\right).$$

и коэффициенты a_0, a_k и b_k :

$$a_0 = 2\alpha_0 = \frac{2}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right)$$

$$a_k = \sqrt{2}\alpha_k = \frac{2}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right) \cos\left(\frac{k * 2\pi i}{N}\right)$$

$$b_k = \sqrt{2}\beta_k = \frac{2}{N} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} f\left(\frac{2Ti}{N}\right) \sin\left(\frac{k * 2\pi i}{N}\right)$$

а общий вид разложения:

$$f\left(\frac{2Ti}{N}\right) = \frac{a_0}{2} + a_1 \cos\left(\frac{2\pi i}{N}\right) + a_2 \cos\left(\frac{4\pi i}{N}\right) + a_3 \cos\left(\frac{6\pi i}{N}\right) + \dots \\ + b_1 \sin\left(\frac{2\pi i}{N}\right) + b_2 \sin\left(\frac{4\pi i}{N}\right) + b_3 \sin\left(\frac{6\pi i}{N}\right) + \dots$$

Разложение функции в комплексный ряд Фурье

Система функций $\{e^{jk\tau}, k = 0, \pm 1, \pm 2, \dots\}$ на отрезке $[-\pi, \pi]$ образует ортонормированную систему функций. Значит, произвольная функция $f(t)$ может быть представлена по этой системе следующим образом:

$$f(t) = \sum_{k=-\infty}^{\infty} C_k e^{jk\tau}$$

Это и есть разложение в комплексный ряд Фурье. Коэффициенты C_k называются комплексными коэффициентами Фурье и, подобно действительным коэффициентам Фурье, вычисляются как скалярные произведения $f(t)$ и e^{jkt} :

$$C_k = \langle f(t), e^{jk\tau} \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-jk\tau} dt$$

Если период функции не равен 2π , а, например, равен T , то получим следующее общее выражение для комплексных коэффициентов:

$$f(t) = \sum_{k=-\infty}^{\infty} C_k e^{j\omega_0 k t} \\ C_k = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j\omega_0 k t} dt \quad (\omega_0 = 2\pi / T)$$

Содержание работы

1. Разработайте программы разложения дискретизированного сигнала

№ варианта	Вид сигнала $f(x)$	№ варианта	Вид сигнала $f(x)$
1	x	6	$x \cdot \cos(x)$
2	x^2	7	$\exp(0.5x)$
3	$ x $	8	$\ln[\cos(x/2)]$
4	$ \sin(x) $	9	$ \cos(x) $
5	$x \cdot \sin(x)$	0	$\text{sign}[\sin(x)]$

в дискретизированный и комплексный ряд Фурье с конечным числом членов.

2. Исследуйте зависимость точности представления заданной функции с помощью ряда Фурье от числа членов разложения и от шага дискретизации функции.

Указания к выполнению

Используйте в качестве основы программы, приведенные в приложении.

Содержание отчета

1. Задание к работе.
2. Тексты разработанных программ с комментариями.
3. Графики зависимости погрешности представления заданной функции в действительный и комплексный ряд Фурье на интервале $[-\pi, \pi]$ и $[-1, 1]$ от количества членов разложения в ряд Фурье и от шага дискретизации.
4. Выводы

Приложение. Тексты базовых программ

Приложение 1. Программа вычисления коэффициентов разложения в

действительный ряд Фурье для функции $y(t)=t \rightarrow \frac{2\pi i}{N}, i = [-\frac{N}{2}, \frac{N}{2}]$.

% Разложение функции t в ряд Фурье

% в дискретизированном виде на интервале $[-T, T]$, например, $[-\pi, \pi]$

N=255; %Количество отсчетов (элементов массива y(t))

K=16; %Количество членов ряда Фурье

T=pi; %диапазон изменения функции f(i) равен +/-T

kp=2.4; %количество периодов гармонической функции

y=zeros(1,N+1);

Sa = zeros(1,K);

Sb = zeros(1,K);

p=1;% показатель степени функции t^p

f=zeros(1,N+1);

Sa0=0;

for i=1:N+1

 f(i)=sin(2*pi*kp*(i-1)/N); % гармоническая функция

 % f(i)= (2*T*((i-1-N/2)/N))^p; %функция t^p

 Sa0=Sa0+f(i);

end

Sa0=Sa0/N

for i=1:N+1

 for j=1:K

 Sa(j) = (Sa(j)+f(i)*cos((j)*2*pi*(i-1-N/2)/N));

 Sb(j) = (Sb(j)+f(i)*sin((j)*2*pi*(i-1-N/2)/N));

 end

end

for j=1:K

 Sa(j)=Sa(j)*(1/(N/2));

 Sb(j)=Sb(j)*(1/(N/2));

end

%Вычисление и отображение спектра амплитуд (начало)

```

for j=1:K
Sab(j)=sqrt(Sa(j)^2+Sb(j)^2);
end
i=1:K;
figure
plot(i,Sab);
stem(Sab(1:K)); %вывод графика дискретной последовательности данных
axis([1 8 -0.2 1.2]); %задание осей: [xmin xmax ymin ymax]
title('Амплитуды частотных составляющих спектра');
xlabel('Количество периодов')
axis tight;
%Вычисление и отображение спектра амплитуд (конец)
y=zeros(1,N+1);
for i=1:N+1
    for j=1:K
        y(i)= y(i)+Sa(j)*cos(j*2*pi*(i-1-N/2)/N)+Sb(j)*sin(j*2*pi*(i-1-N/2)/N);
    end
    y(i)=Sa0+y(i);
end
i=1:N+1;
figure
plot(i,f);
axis tight;
hold on;
plot(i,y,'r-')
hold off;
pause;
close all;
clear; %очистка Workspace

```

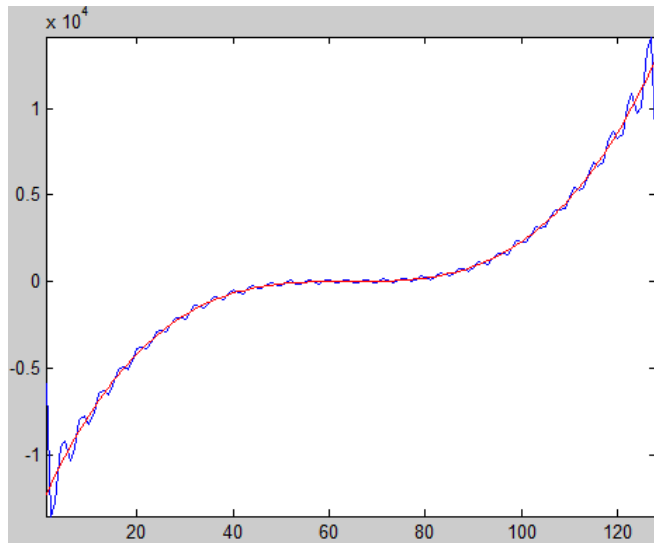


Рис. 2.1. Исходная и восстановленная функция t^3 при $N=128$, $K=32$, $p=3$.

Примечание. Для разложения четной функции из ряда можно исключить члены, содержащие синусы, для разложения нечетной – косинусы. Но можно оставить ряд разложения полностью.

Приложение 2. Программа вычисления коэффициентов разложения в действительный ряд Фурье для функции $y(t) = t \rightarrow \frac{2Ti}{N}$, $i = [-\frac{N}{2}, \frac{N}{2}]$.

```
% Разложение функции y(t)=t в ряд Фурье
% в дискретизированном виде на интервале [0,T], например, [0,  $\pi$ ]

N=255; %Количество отсчетов (элементов массива y(t)=t)
K=64;%Количество членов ряда Фурье
T=pi;%диапазон изменения функции f(i)+/-T
kp=2.0
y=zeros(1,N+1);
Sa = zeros(1,K);
Sb = zeros(1,K);
p=3;%показатель степени функции t^p
f=zeros(1,N+1);
Sa0=0;
for i=1:N+1
    f(i)=sin(2*pi*kp*(i-1)/N); % гармоническая функция
    % f(i)= (T*(((i-1))/N))^p; %функция t^p, i-1, если p>0, i, если p<0
    Sa0=Sa0+f(i);
end
Sa0=Sa0/N
for i=1:N+1
    for j=1:K
        Sa(j) = (Sa(j)+f(i)*cos((j)*2*pi*(i-1)/N));
        Sb(j) = (Sb(j)+f(i)*sin((j)*2*pi*(i-1)/N));
    end
end
for j=1:K
    Sa(j)=Sa(j)*(1/(N/2));
    Sb(j)=Sb(j)*(1/(N/2));
end
%Вычисление и отображение спектра амплитуд (начало)
for j=1:K
    Sab(j)=sqrt(Sa(j)^2+Sb(j)^2);
```

```

end
i=1:K;
figure
plot(i,Sab);
stem(Sab(1:K)); %вывод графика дискретной последовательности данных
axis([1 8 -0.2 1.2]);%задание осей: [xmin xmax ymin ymax]
title('Амплитуды частотных составляющих спектра');
xlabel('Количество периодов')
axis tight;
%Вычисление и отображение спектра амплитуд (конец)
y=zeros(1,N+1);
for i=1:N+1
    for j=1:K
        y(i)= y(i)+Sa(j)*cos(j*2*pi*(i-1-N)/N)+Sb(j)*sin(j*2*pi*(i-1-N)/N);
    end
    y(i)=Sa0+y(i);
end

i=1:N+1;
figure
plot(i,f);
axis tight;
hold on;
plot(i,y,'r-')
hold off;
pause;
close all;
clear; %очистка Workspace

```

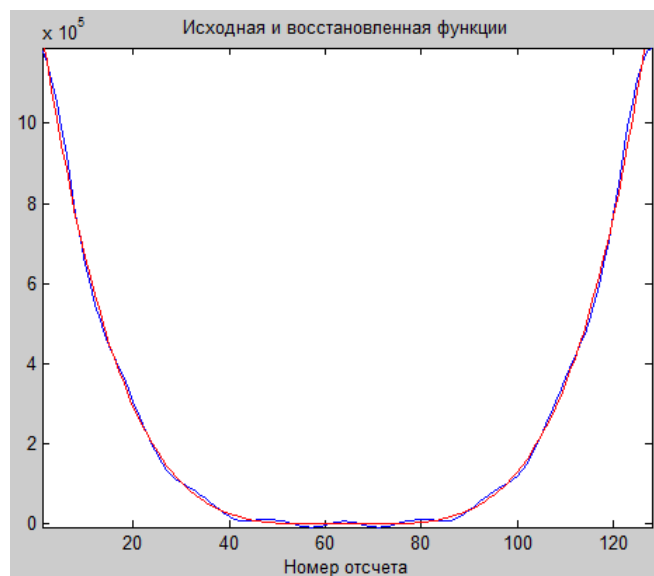


Рис. 2.2. Исходная и восстановленная функция $y=t^p$ после разложения в ряд Фурье, $N=128$, $T=5$, $p=4$, $K=16$.

Приложение 3. Программа вычисления коэффициентов разложения в

комплексный ряд Фурье для функции $y(t)=t^2 \rightarrow (\frac{2Ti}{N})^2, \quad i = [-\frac{N}{2}, \frac{N}{2}]$.

%Разложение функции t^p в комплексный ряд Фурье

%в дискретизированном виде на интервале $[0, T]$

%Восстановление функции производится по формуле

% $f_b(i)=y(i)=\sum(c(k)*\exp(j*2*\pi*i*0/N)), k=[1,M], i=[0,N-1]$

%Чем больше M, тем точнее восстановление

T=pi;%Значение T (произвольное)

N=128;%количество значений функции на интервале $[0, T]$

M=6;%количество членов ряда Фурье

p=1;%показатель степени функции x^p

kp=2.4;%количество периодов гармонического сигнала

C0=0;

for i=1:N+1 %генерация модельной функции

 f(i)=sin(2*pi*kp*(i-1)/N); % гармоническая функция

 % f(i)= (T*((i-1)/N))^p; %функция t^p

 C0=C0+f(i);

end

C0=C0*(2/N);

for k=1:M

 C(k)=0;

end

for i=1:N+1

 for k=1:M

 C(k)=C(k)+f(i)*exp(-j*2*pi*k*(i-1)/N);

 end

end

for k=1:M

 C(k)=C(k)*(2/N);

end

%Вычисление и отображение спектра амплитуд (начало)

for k=1:M

 Cab(k)=abs(C(k));%коэффициенты Cab(k)- комплексные числа вида $a+jb$,

 %функция abs вычисляет $\sqrt{a^2+b^2}$

```

end
k=1:M;
figure
plot(k,Cab);
stem(Cab(1:M)); %вывод графика дискретной последовательности данных
axis([1 8 -0.2 1.2]);%задание осей: [xmin xmax ymin ymax]
title('Амплитуды частотных составляющих спектра');
xlabel('Количество периодов')
axis tight;
%Вычисление и отображение спектра амплитуд (конец)
for i=1:N+1
    y(i)=0;
    %f3(i)=0;
    for k=1:M
        y(i)=y(i)+C(k)*exp(j*2*pi*k*(i-1)/N);
    end
    y(i)=C0/2+y(i);
end
i=1:N+1;
figure
plot(i,f);
axis tight;
title('Исходная и
восстановленная функция')
xlabel('Номер элемента массива')
hold on;
plot(i,real(y),'r-');
axis tight;
hold off;
pause;
close all;
clear;

```

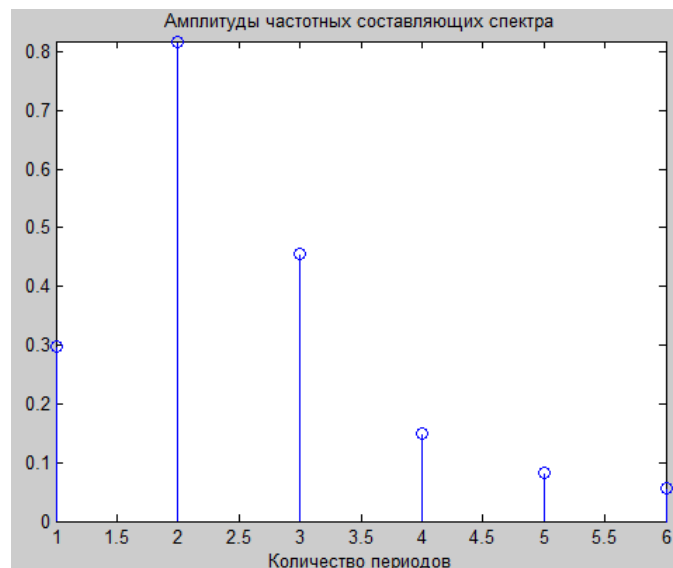


Рис. 2.3. Спектр амплитуд гармонического сигнала с количеством периодов $k_p=2.4$

3. Цифровая фильтрация шумов в среде MATLAB

Целью лабораторной работы является изучение методики разработки программ цифровой обработки сигналов, включающей различные способы улучшения отношения сигнал/шум (накопление, использование НЧ и ВЧ-фильтров, оптимального фильтра Колмогорова-Винера, прямого и обратного БПФ).

Задание к работе

Имеется набор экспериментальных данных в виде числового массива. Требуется спроектировать на внутреннем языке MATLAB программу цифровой обработки данных, реализующую различные способы улучшения отношения сигнал/шум:

1. Накопление.
2. НЧ-фильтр.
3. Фильтр Баттерворта.
4. Фильтр скользящего среднего.
5. Фильтр БПФ - ОБПФ.
6. Оптимальный фильтр Колмогорова-Винера.

и оценить сравнительный эффект улучшения отношения сигнал/шум и степень искажения сигнала в результате обработки.

Теоретические основы

Цифровая обработка сигналов решает задачи обнаружения и определения параметров информативных сигналов и изображений, искаженных шумами и помехами. Для этой цели используются различные средства:

- накопление (временная фильтрация);
- цифровые частотные фильтры (высокой частоты, низкой частоты, полосовые фильтры, фильтр Баттерворта);
- сглаживающие фильтры (скользящего среднего, медианный);
- оптимальные фильтры (фильтр Колмогорова-Винера, LMS и RLS-фильтры);
- адаптивные фильтры;

Выбор способа борьбы с шумами должен производиться с учетом свойств и особенностей информативного сигнала и помехи. Чем в большей степени свойства сигнала и шума априори известны, тем может быть получен больший эффект от цифровой обработки. Кроме того, несмотря на наличие стандартных программ цифровой обработки, с учетом конкретных априори известных свойств информативного сигнала и шума может оказаться полезным разработка новых методов борьбы с шумами.

ЦИФРОВОЙ НИЗКОЧАСТОТНЫЙ ФИЛЬТР

Для фильтрации высокочастотного шума может быть применен фильтр низких частот (ФНЧ). Частотная характеристика ФНЧ выражается как

$$K(\omega) = \frac{1}{1 + j\omega}, \quad \text{где } \omega - \text{частота среза НЧ - фильтра}$$

Для фильтрации сигнала нужно вычислить частотный спектр сигнала с помощью преобразования Фурье, затем перемножить частотный спектр сигнала и частотную функцию фильтра и выполнить обратное преобразование Фурье. Второй способ – вычислить импульсную переходную характеристику фильтра (реакцию на единичный импульс), затем выполнить операцию свертки входного сигнала с импульсной переходной характеристикой фильтра.

В программах обработки дискретизированных сигналов, представленных в форме числовых массивов, цикл

```
for k=1:N
```

```
x(k) = A*sin(2*pi*KP*k/N);
```

```
end
```

создает KP периодов дискретизированного синусоидального сигнала в числовом массиве, содержащем N значений, понятие частоты в этом случае отсутствует и появляется только в том случае, если задать шаг дискретности по времени. Аналогично этому, в результате выполнения быстрого преобразования Фурье

```
I=1:N;
```

```
Y=fft(x,N);
```

мы получаем числовой массив, который может содержать N элементов, причем информативной будет только первая половина массива, вторая будет зеркально отображать первую половину. В первой половине массива, содержащей N/2

элементов, будет представлен «частотный» спектр. Спектр будет отражать не частоту сигнала, установить которую по числовому массиву, представляющему сигнал во временной области, невозможно, а количество периодов. Т.е. в приведенном выше примере пик в массиве частотного спектра будет в элементе с номером КР. Таким образом, положение пика (номер элемента массива частотного спектра) укажет на количество периодов сигнала во временной области.

Частотная характеристика линейного фильтра низких частот может быть вычислена следующим образом:

```
for i=1:N  
H(i)=1/((1+j*i/NC));  
end
```

Здесь NC - полоса пропускания фильтра по уровню 0,7 амплитуды выражена в количестве отчетов спектра БПФ, пропускаемых фильтром. Остальные отсчеты в массиве частотного спектра будут ослабляться по амплитуде. Таким образом, понятие постоянной времени фильтра, равно как и полосы пропускания, при дискретизированном представлении линейного фильтра отсутствует.

Ниже приведены программы фильтрации сигналов и временные диаграммы.

%Низкочастотный фильтр

A=1; %амплитуда сигнала

Q=0.05; %амплитуда шума

KP1=5;% - количество периодов первого сигнала

KP2=5;% - количество периодов второго сигнала

N=1024;%количество отсчетов

NC=5;

%NC - полоса пропускания фильтра по уровню 0,7 амплитуды

% выражена в количестве отчетов спектра БПФ, пропускаемых фильтром

% остальные отсчеты (в частотном спектре!) будут ослабляться по амплитуде

for k=1:N % генерация сигнала и шума

s(k) = A*sin(2*pi*KP1*k/N);%+ A*sin(2*pi*KP2*k/N);

q(k)=Q*(randn(size(N))); %СКО шума равно Q

x(k)=s(k)+q(k); % суммирование сигнала и шума

```

end
figure
plot(x);
axis tight; %диапазон X и Y по осям точно соответствует Xmax и Kmax
title('Зашумленный сигнал до фильтра');
Y=fft(x,N); %БПФ сигнала с шумом
i=1:N/2;
figure
% semilogy(i(1:200),2*abs(SS1(1:200)));
%plot(i(1:100),2*abs(SS1(1:100)));
plot(i(1:N/2),abs(Y(1:N/2)));
title('Частотный спектр сигнала с шумом');
for i=1:N;
H(i)=1/((1+j*i/NC)); %передаточная функция фильтра НЧ 1-го порядка
%в частотной области
end
h=ifft(H);
% HH=fft(h,N);
i=1:N;
plot(i(1:20),abs(h(1:20))); %импульсная характеристика фильтра
title('Импульсная характеристика фильтра');
i=1:200;
figure
%plot(i,abs(H(1:200)));
semilogx(i,abs(H(1:200))); %то же, что и plot, но в логарифмическом
%масштабе по X
grid on;
title('Частотная хар-ка НЧ-фильтра');
i=1:N;
XX1=fft(x,N); %частотный спектр сигнала с шумом
Z=ifft(XX1.*H); %свертка зашумленного сигнала с частотной хар-кой фильтра
XX2=fft(s,N); %частотный спектр сигнала
Z2=ifft(XX2.*H); %свертка незашумленного сигнала с частотной хар-кой
фильтра
DZ(i)=(2*real(Z(i))-2*real(Z2(i)))*100./max(real(Z2)); %случайная погрешность
DZ1(i)=(2*real(Z(i))-x(i))*100/max(x); %полная погрешность

```

```

SKO=std(DZ)
SKO1=std(DZ1)
i=1:N;
yy=A*sin((6.28*KP1*i/N));
figure
plot(i,x); %вывод сигнала до фильтра
title('Сигнал до фильтра');
xlabel('Номер отсчета'); % подпись по оси X
ylabel('Амплитуда'); % подпись по оси Y
axis tight; %диапазон X и Y по осям точно соответствует Xmax и Ymax
hold on; % "удержание" окна вывода для вывода следующего графика
i=1:N;
plot(i,2*real(Z(1:N)),'r-'),grid; %вывод отфильтрованного сигнала
%представление графика линией красного цвета, отображение сетки
title('Сигнал до и после фильтра');%подпись названия графика
hold off;
i=1:N;
figure
plot(i,DZ(1:N)); %вывод случайной погрешности отфильтрованного сигнала
title('Случайная погрешность отфильтрованного сигнала');
ylabel('Случайная погрешность, %'); % подпись по оси Y
axis tight;
i=1:N;
figure
plot(i,DZ1(1:N)); %вывод случайной погрешности отфильтрованного сигнала
title('Полная погрешность отфильтрованного сигнала');
ylabel('Полная погрешность, %'); % подпись по оси Y
axis tight;
pause;
close all;%закрытие окон графического вывода
clear;%очистка Workspace

```

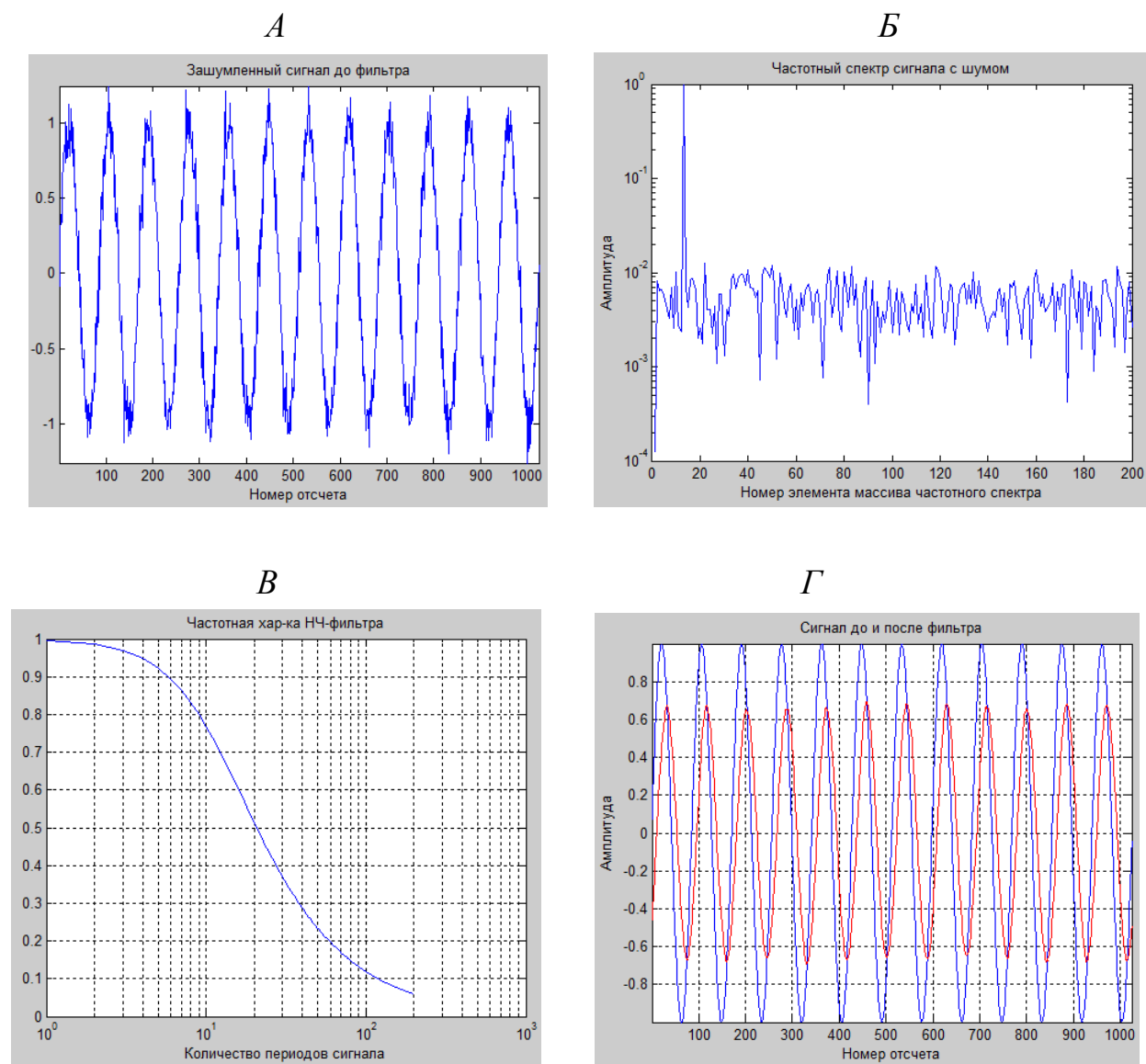


Рис.3.1. Исходный сигнал с шумом (А), его частотный спектр (Б), частотная характеристика фильтра в полулогарифмическом масштабе (В) и сигнал до и после фильтра (Г).

ФИЛЬТР БАТТЕРВОРТА

Недостатком линейного цифрового НЧ-фильтра первого порядка является существенная неравномерность амплитудно-частотной характеристики в полосе пропускания, малая крутизна ее спада. Эти недостатки могут быть в некоторой степени преодолены при использовании фильтров более высоких порядков, в значительной мере указанные недостатки преодолены в фильтрах Баттерворта.

Ниже приведена программа фильтрации сигналов с помощью фильтра Баттерворта 4-го порядка и временные диаграммы.

%Фильтр НЧ Баттерворта 4-го порядка

A=1; %амплитуда сигнала

Q=0.3; %СКО шума

KP1=5;% - количество периодов первого сигнала

KP2=5;% - количество периодов второго сигнала

N=1024;%количество точек расчета

NC=15;

%NC - полоса пропускания фильтра по уровню 0,7 амплитуды

% выражена в количестве отчетов спектра БПФ, пропускаемых фильтром

% остальные отсчеты (в частотном спектре!) будут ослабляться по амплитуде

for k=1:N % генерация сигнала и шума

s(k) = A*sin(2*pi*KP1*k/N); %+ A*sin(2*pi*KP2*k/1000);

q(k)=Q*(randn(size(N))); %СКО шума, амплитуда равна 3Q

x(k)=s(k)+q(k); % суммирование сигнала и шума

end

figure

plot(x);

title('Зашумленный сигнал до фильтра');

Y=fft(x,N)/N; %БПФ сигнала с шумом

i=1:N/2;

figure

% semilogy(i(1:200),2*abs(SS1(1:200)));d

%plot(i(1:100),2*abs(SS1(1:100)));

plot(i(1:N/2),real(Y(1:N/2)));

title('Частотный спектр сигнала с шумом');

for i=1:N;

H(i)=1/(sqrt(1+(j*i/NC).^4)); %передаточная функция фильтра Баттерворта 4-го порядка

end

i=1:200;

figure

%plot(i,abs(H(1:200))); %вывод частотной хар-ки фильтра

```

semilogx(i,abs(H(1:200))),grid;%то же, что и plot, но в логарифмическом
%масштабе по X
%loglog(i,abs(H(1:200))),grid;% в логарифм. масштабе по X и Y
axis tight;
title('Частотная хар-ка фильтра Баттерворта');

i=1:N;
XX1=fft(x,N); %частотный спектр сигнала с шумом
Z=ifft(XX1.*H);%свертка зашумленного сигнала с частотной хар-кой фильтра
XX2=fft(s,N);%частотный спектр сигнала без шума
Z2=ifft(XX2.*H);%свертка сигнала с частотной хар-кой фильтра
DZ=(2*real(Z(i))-2*real(Z2(i)))/A*100;%вычисление случайной составляющей
погрешности
DZ1=(2*real(Z(i))-s(i))/A*100;%вычисление полной погрешности
SKO=std(DZ)
SKO1=std(DZ1)
figure
plot(i,x); %вывод сигнала до фильтра
%title('Сигнал до фильтра');
xlabel('Номер отсчета'); % подпись по оси X
ylabel('Амплитуда'); % подпись по оси Y
legend('до фильтра');
axis tight; %диапазон X и Y по осям точно соответствует Xmax и Ymax
hold on; % "удержание" окна вывода для вывода следующего графика
plot(i,2*real(Z(1:N)),'r-'); %вывод отфильтрованного сигнала
%представление графика линией красного цвета
grid on; % отображение сетки
title('Сигнал до и после фильтра');%подпись названия графика
hold off;

i=1:N;
figure
plot(i,real(DZ(1:N))); %вывод случайной погрешности
title('Случайная погрешность отфильтрованного сигнала, %');
i=1:N;
figure

```

```
plot(i,real(DZ1(1:N))); %вывод полной погрешности
title('Полная погрешность отфильтрованного сигнала, %');
```

```
pause;
close all; %заккрытие окон графического вывода
clear; %очистка Workspace
```

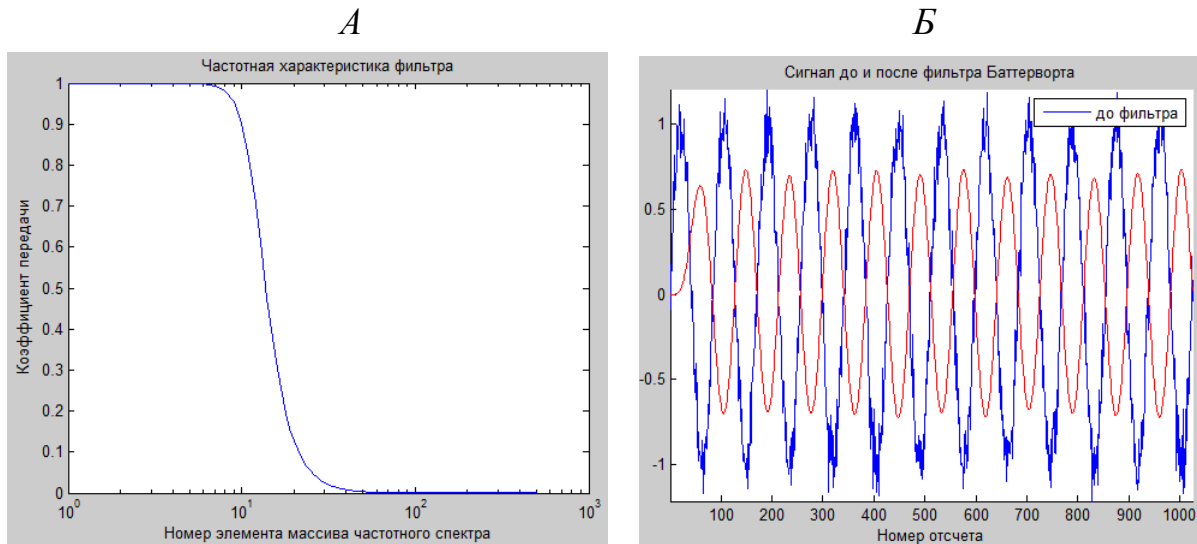


Рис.3.2. Частотная характеристика фильтра в полулогарифмическом масштабе (А) и сигнал до и после фильтра (Б).

ОПТИМАЛЬНЫЙ ФИЛЬТР КОЛМОГорова-ВИНЕРА

Фильтры низкой частоты, высокой частоты и полосовые фильтры эффективны в том случае, когда частотные спектры сигнала и шума не перекрываются.

Наилучшее разделение сигнала и шума цифровыми методами обеспечивает оптимальный фильтр Колмогорова-Винера.

Частотная характеристика фильтра Колмогорова-Винера:

$$H(w) = W_s(w) / [W_s(w) + W_q(w)] ,$$

где $W_s(w)$ и $W_q(w)$ - энергетические спектры (плотности мощности) сигнала и помех.

Спектр плотности мощности W в MATLAB рассчитывается по формуле:

$$W(k) = \frac{1}{N * F_s} X(k) .* conj(X(k))$$

где $X(k)$ – N-точечное ДПФ N-точечной последовательности $x(n)$, N-размерность ДПФ, F_s – частота дискретизации.

$W(k)$ можно вычислять по более простой формуле:

$$W(k) = X(k) \cdot \text{conj}(X(k))$$

т.к. множитель $\frac{1}{N \cdot F_s}$ в выражении $H(w)$ равен единице.

Программа, реализующая оптимальный фильтр Колмогорова – Винера в среде MATLAB для трех часто встречающихся видов сигнала: гармонического, колоколообразного и прямоугольного, приведена ниже.

%Фильтр Колмогорова-Винера

A=1; %амплитуда сигнала

Q=0.1; %амплитуда шума (в долях СКО)

N=1024;%количество точек расчета

kp1=12;

kp2=24;

for k=1:N %цикл вычисления сигнала и шума

%s1(k)=A*exp(-0.0003*(k-200)^2.0); %колоколообразный сигнал

s1(k)=A*sin(2*pi*kp1*k/N);%+A*sin(2*pi*kp2*k/N);%гармонический сигнал

% s1(k)=0; % сигнал прямоугольной формы

% if (k>100)&(k<300) % сигнал прямоугольной формы

% s1(k)=A;

% end

q(k)=Q*(randn(size(N))); %шум

x1(k)=s1(k)+q(k); % суммирование сигнала и шума

end

figure

plot(x1(1:N));

title('Зашумленный сигнал до фильтра');

axis tight;

Y=fft(x1,N)/N; %БПФ сигнала с шумом

SS1=Y.*conj(Y)/N; %спектр мощности

i=1:200;

figure

%plot(i,SS1(1:200));

semilogy(i,SS1(1:200)); %вывод спектра мощности сигнала с шумом

title('Частотный спектр сигнала с шумом');


```

Y=fft(s1,N)/N; %БПФ сигнала без шума
f = N/2*linspace(0,1,N/2);%вычисление шкалы для частотной области
SS1=Y.*conj(Y)/N; %спектр мощности сигнала без шума
Y1=fft(q,N)/N; %БПФ шума
f = N/2*linspace(0,1,N/2);
SS2=Y1.*conj(Y1)/N; %спектр мощности шума
for i=1:N
H(i)=SS1(i)/(SS1(i)+SS2(i));%частотная характеристика оптимального фильтра
end
i=1:200;
figure
%plot(i,abs(H(1:200)));
semilogx(i,abs(H(1:200)));
%hold on
title('Частотная характеристика оптимального фильтра');
i=1:N;
XX1=fft(x1,N); %частотный спектр сигнала с шумом
Z=ifft(XX1.*H);%свертка зашумленного сигнала с частотной хар-кой фильтра
axis tight;
figure
plot(i,s1(1:N)); %вывод незашумленного сигнала до фильтра сигнала
title('Незашумленный сигнал до фильтра');
axis tight;
figure
plot(i,Z(1:N)); %вывод отфильтрованного сигнала
title('Сигнал после свертки с част. хар-кой оптимального фильтра');
axis tight;
i=1:N;
DZ(i)=Z(i)-s1(i);
DZ1=DZ*100/max(s1);
SKO=std(DZ1)

i=1:N;
figure
plot(i,DZ1(1:N)); %вывод случайной погрешности отфильтрованного сигнала
title('Погрешность отфильтрованного сигнала');

```

```

ylabel('Полная погрешность, %'); % подпись по оси Y
axis tight;
pause;
close all; %закрытие окон графического вывода
clear; %очиска Workspace

```

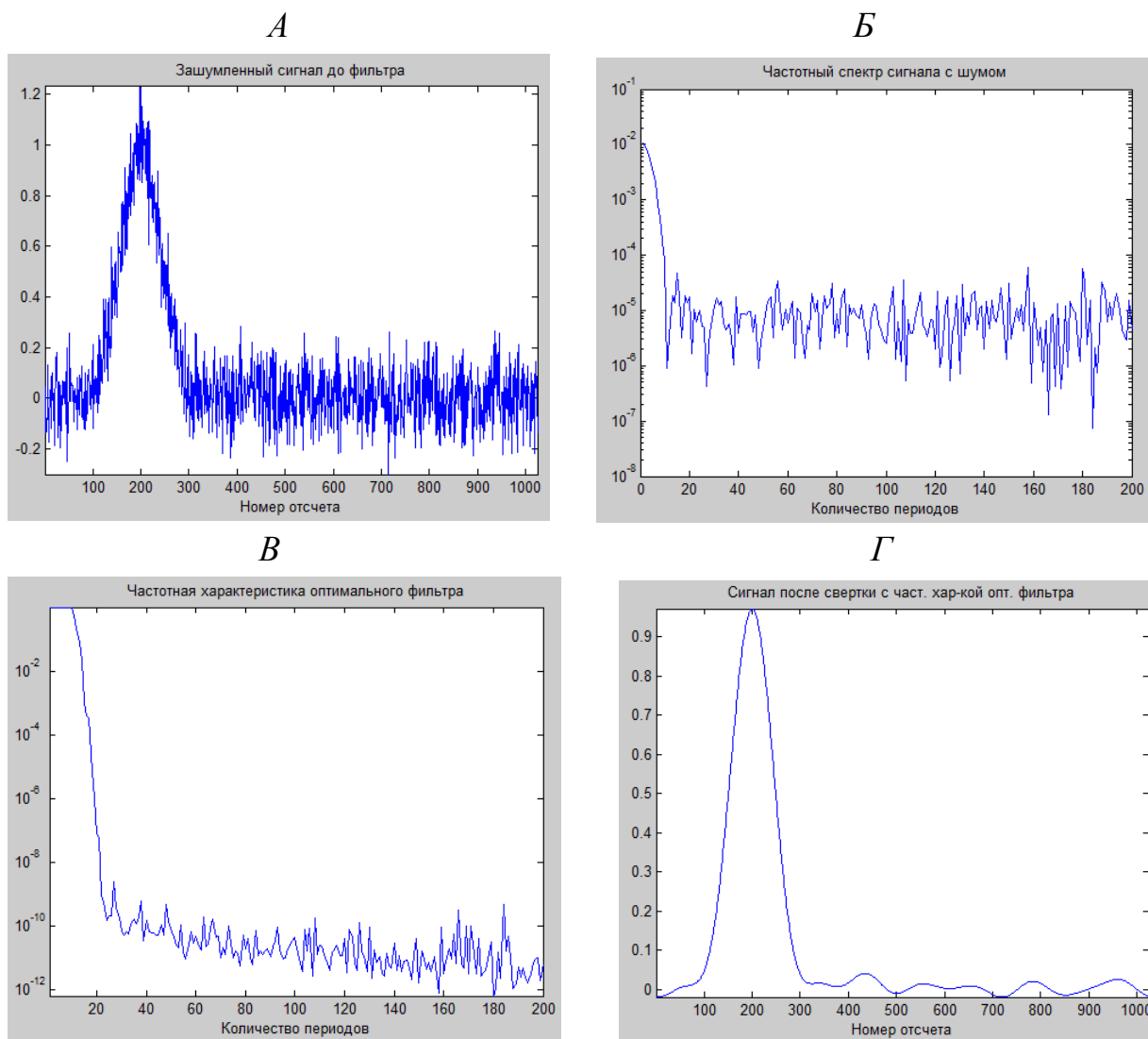


Рис. 3.3. Исходный колоколообразный сигнал с шумом (А), его частотный спектр (Б), частотная характеристика оптимального фильтра Колмогорова-Винера (В) и сигнал после фильтра (Г).

ФИЛЬТРАЦИЯ ШУМОВ С ИСПОЛЬЗОВАНИЕМ ПРЯМОГО И ОБРАТНОГО БПФ

В том случае, когда частотные спектры сигнала и шума не перекрываются, фильтрация шума может быть произведена путем выполнения БПФ, обнуления спектральных линий шума и последующего обратного БПФ.

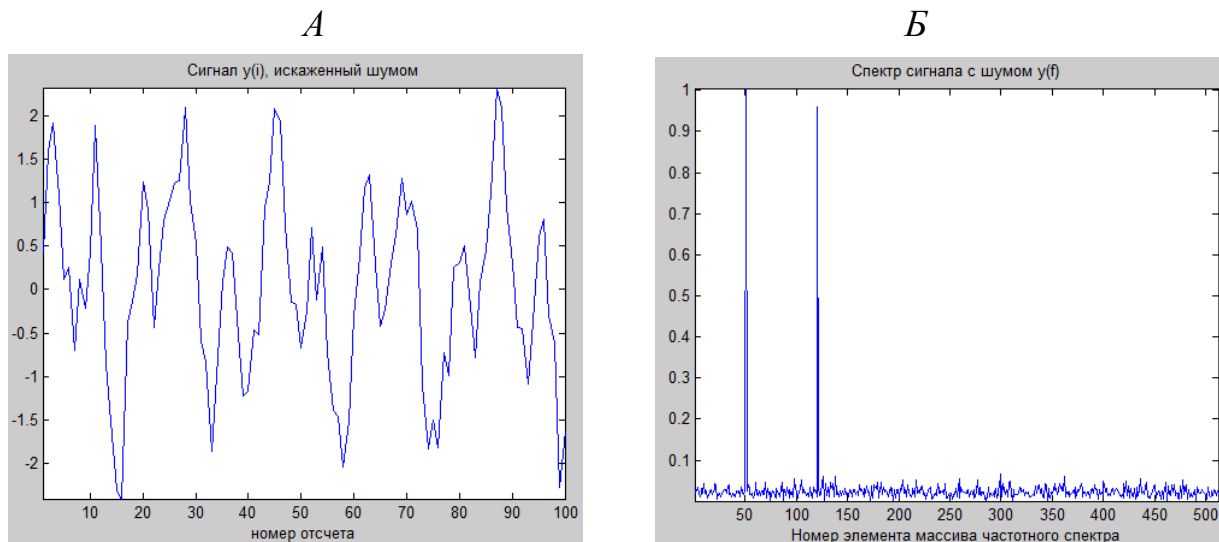


Рис. 3.4. Зашумленный сигнал (А), представляющий сумму двух синусоидальных сигналов разных частот (количество периодов во временном окне 50 и 120), и частотный спектр, полученный с помощью БПФ (Б).

Если теперь обнулить участки спектра от 0 до 40, от 55 до 110 и от 125 до 500, а затем выполнить обратное преобразование БПФ (ifft), то получим спектр и сигнал, представленные на рис. 3.5.

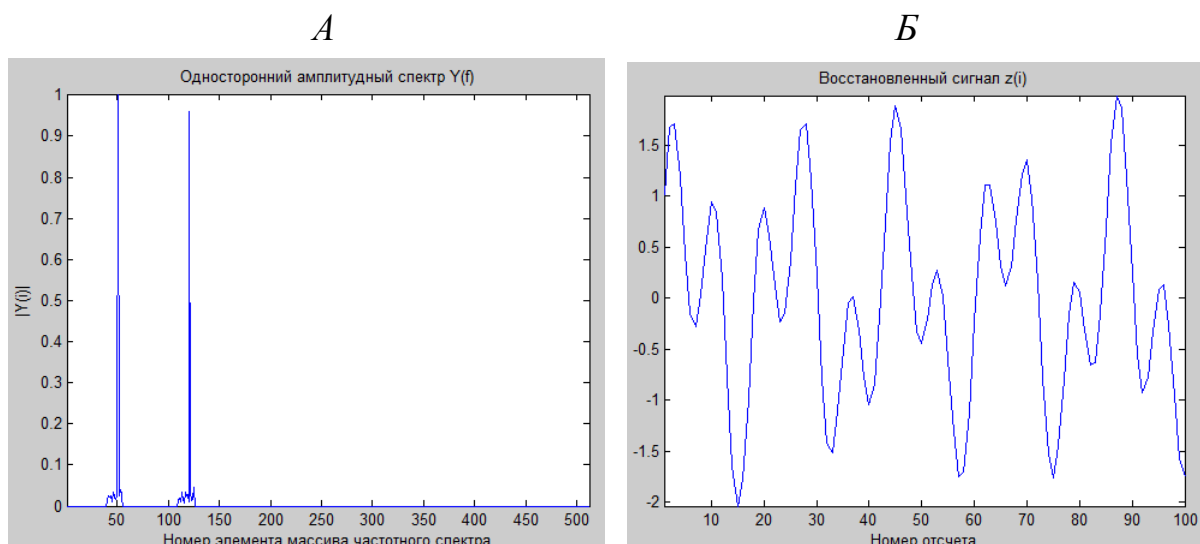


Рис. 3.5. Частотный спектр после удаления спектральных составляющих шума (А) и восстановленный с помощью обратного преобразования Фурье сигнал (Б)

Текст программы фильтрации помех с помощью прямого и обратного преобразования Фурье приведен ниже.

```
%Программа фильтрации помех с использованием БПФ-ОБПФ
```

```
A=1;%амплитуда первого сигнала
```

```
B=1; %амплитуда второго сигнала
```

```
Q=0.4;%уровень шума, выраженный в долях СКО. СКО шума равно 1.
```

```
N=1024;%количество отсчетов сигнала
```

```
kp1=50;%количество периодов 1-го сигнала
```

```
kp2=120;%количество периодов 2-го сигнала
```

```
% Сумма сигналов 1 и 2
```

```
for i=1:N
```

```
x(i) = A*sin(2*pi*kp1*i/N) + B*sin(2*pi*kp2*i/N);
```

```
z(i)=Q*randn(size(N));%
```

```
y(i) = x(i) + z(i); % Суммирование сигнала и шума
```

```
end
```

```
Y = fft(y,N)/N;%вычисление спектра сигнала с шумом
```

```
i=1:N/2;
```

```
figure
```

```
plot(i,2*abs(Y(1:N/2)));
```

```
title('Спектр сигнала с шумом y(f)')
```

```
xlabel('Номер элемента массива частотного спектра')
```

```
axis tight;
```

```
for i=1:N %"вырезание" спектральных составляющих шума
```

```
if (i>125)|(i<40)|((i>55)&(i<110))
```

```
Y(i)=0;
```

```
end
```

```
end
```

```
i=1:N/2;
```

```
figure
```

```
plot(i,2*abs(Y(1:N/2)));
```

```
title('Односторонний амплитудный спектр Y(f)')
```

```
xlabel('Номер элемента массива частотного спектра')
```

```
ylabel('|Y(i)|')
```

```

axis tight;
%восстановление исходного сигнала после "вырезания"
%спектральных составляющих шума
z=ifft(Y)*2*N;
i=1:N;
figure
plot(i(1:100),x(1:100));
title('Исходный сигнал x(i)')
xlabel('Номер отсчета');
axis tight;
figure
plot(i(1:100),y(1:100));
title('Сигнал y(i), искаженный шумом')
xlabel('номер отсчета')
axis tight;
figure
plot(i(1:100),z(1:100));
title('Восстановленный сигнал z(i)')
xlabel('Номер отсчета');
axis tight;
for i=1:N
DZ(i)=x(i)-real(z(i));%уровень зашумления в сигнале после фильтра
%результат обратного преобразования Фурье - массив комплексных чисел
%Восстановленный сигнал - действительные части комплексных чисел
%Вычисляются с помощью функции real
end
DZ1=DZ*100/max(x);
SKO=std(DZ1)
figure
i=1:N;
plot(i,DZ1(i:N)); %вывод погрешности отфильтрованного сигнала
title('Погрешность отфильтрованного сигнала');
ylabel('Случайная погрешность, %'); % подпись по оси Y
xlabel('Номер отсчета');
axis tight;

```

```
pause;  
close all; %заккрытие всех окон графического вывода  
clear;%очистка Workspace
```

ФИЛЬТР СКОЛЬЗЯЩЕГО СРЕДНЕГО

Пусть мы имеем массив N значений измеренного сигнала, представленный в цифровой форме:

$$\{f_1, f_2, \dots, f_N\}, i=1, 2, 3, \dots, N$$

Для нахождения скользящего среднего в окрестности точки f_i берем среднее арифметическое от K предыдущих и K последующих точек, включая и f_i . Таким же образом производим обработку для всех значений i . В результате вычисляем новый массив g_i :

$$g_i = \frac{1}{2K+1} (f_{i-K} + f_{i-K-1} + \dots + f_i + \dots + f_{i+K})$$

или

$$g_i = \frac{1}{2K+1} \sum_{j=-K}^K f_{i+j}$$

Текст программы фильтрации помех с помощью фильтра скользящего среднего приведен ниже.

```
%Фильтр скользящего среднего  
  
A=1; %амплитуда сигнала  
Q=0.05; %СКО шума  
KP1=10;% - количество периодов первого сигнала  
KP2=5;% - количество периодов второго сигнала  
N=1024;%количество точек расчета  
  
for k=1:N % генерация сигнала и шума  
s(k) = A*sin(2*pi*KP1*k/N);%+ A*sin(2*pi*KP2*k/N);  
q(k)=Q*(randn(size(N))); %СКО шума равно Q  
x(k)=s(k)+q(k); % суммирование сигнала и шума
```

```

end

S=x(1)+x(2)+x(3)+x(4)+x(5)+x(6)+x(7);
y(4)=S/7;
S1=s(1)+s(2)+s(3)+s(4)+s(5)+s(6)+s(7);
y1(4)=S1/7;
for i=1:N-7 %сглаживание зашумленного сигнала
    S=S-x(i)+x(i+7);
    y(i+4)=S/7;
end

for i=4:N-4
    DZ(i)=s(i)-y(i);%уровень зашумления в сигнале после фильтра
end
DZ=DZ*100/max(s);%остаточная погрешность после фильтрации
SKO=std(DZ)

i=4:N-4;
figure;
plot(x);
hold on;
title('Зашумленный сигнал до фильтра');
plot(i,y(4:N-4),'r-');
title('Сигнал после фильтра');
xlabel('Номер отсчета'); % подпись по оси X
axis tight;
hold off;
figure
plot(i,DZ(4:N-4)); %вывод погрешности отфильтрованного сигнала
title('Погрешность отфильтрованного сигнала');
xlabel('Номер отсчета'); % подпись по оси X
ylabel('Полная погрешность, %'); % подпись по оси Y
axis tight;
pause;
close all; %закрытие окон графического вывода
clear; %очистка Workspace

```

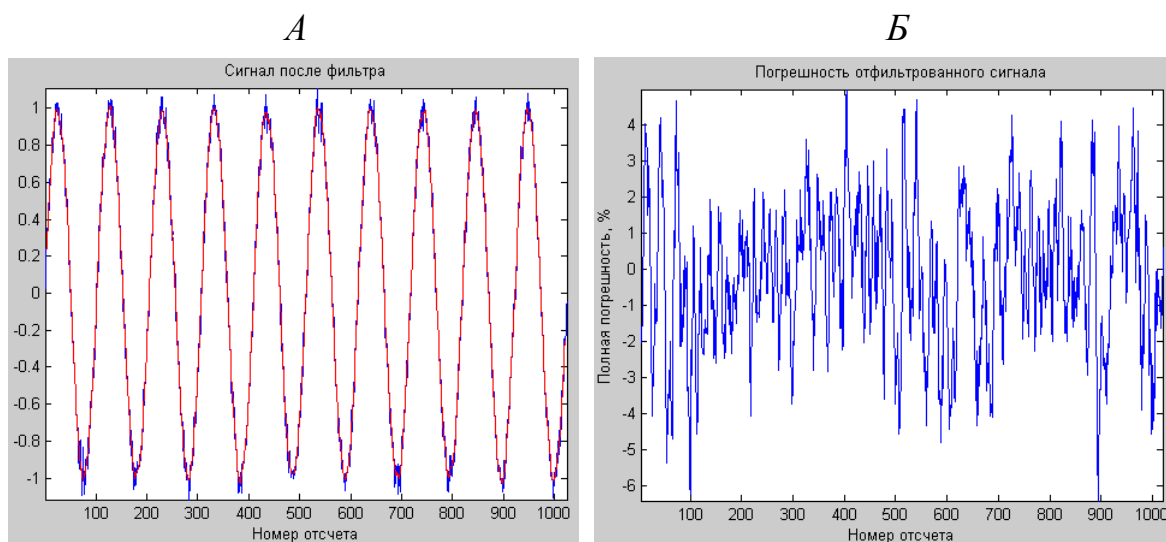


Рис. 3.6. Сигнал до и после сглаживания (А) и полная погрешность после сглаживания (Б).

Указания к выполнению работы

1. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в сравнении с исходным благодаря накоплению постройте зависимость отношения сигнал/шум и разности исходного и обработанного сигналов от количества накоплений.

2. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в сравнении с исходным благодаря использованию фильтров (НЧ, Баттерворта, БПФ-ОБПФ и др.) найдите оптимальную полосу пропускания фильтра, при которой эффект будет наибольшим с точки зрения улучшения отношения сигнал/шум и уменьшения полной погрешности обработанного сигнала отдельно.

3. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала в сравнении с исходным благодаря использованию фильтра скользящего среднего (Smoothing) определите зависимость степени подавления шумов от ширины окна (при рамере окна 3, 5, 7 элементов) при различном уровне шумов и влияние параметров фильтра на изменения амплитуды и фазы сигнала на выходе фильтра.

4. При исследовании эффекта улучшения отношения сигнал/шум и уменьшения погрешности обработанного сигнала произведите сравнение эффективности исследуемых цифровых фильтров с оптимальным фильтром Колмогорова-Винера. Во всех пунктах задания количество периодов исходного

сигнала КР1 следует выбрать равным $2+2*N$, где N - номер варианта для студента, соответствующий последней цифре в зачетке.

Содержание отчета

1. Задание к работе.
2. Таблица исходных данных.
3. Тексты программ фильтрации с комментариями
4. Графики исходного и обработанного сигналов, график погрешности при различных параметрах обработки.
5. Выводы.

4. Базовые средства фильтрации шумов на изображениях

Целью лабораторной работы является изучение методики использования базовых средств фильтрации шумов на изображениях в среде MATLAB, включающих усредняющий и медианный фильтры, фильтр Гаусса, адаптивный фильтр Винера, фильтр повышения резкости и ранговый фильтр и оценки их сравнительной эффективности.

Задание к работе

1. Исследуйте связь эффективности фильтрации зашумленных изображений с визуальным восприятием изображений и степенью сходства отфильтрованного изображения с исходным незашумленным изображением, найдите оптимальные параметры фильтров;
2. Исследуйте связь эффективности фильтрации от показателей яркости и контрастности исходного изображения;
3. Спроектируйте на внутреннем языке MATLAB программу цифровой фильтрации изображений, содержащей средства интерфейса пользователя, позволяющие выбирать вид и задавать параметры фильтрации.

Описание программного обеспечения и порядок работы

Программное обеспечение включает базовую программу фильтрации шумов и файлы фотоизображений. Программа фильтрации шумов позволяет загрузить Ваш собственный файл и осуществить его зашумление, произвести фильтрацию зашумленного изображения с помощью различных фильтров, вычислить коэффициент корреляции зашумленного и отфильтрованных изображений с исходным, выполнить построение 2D и 3D графика коэффициента корреляции. Виды и параметры шума и параметры фильтров могут задаваться. Текст программы приведен в приложении.

Указания к выполнению работы

1. При выполнении п.1 Программы установите в программе уровень шума 0.005, вид шума – Гауссов, запустите программу Filtering.m, произведите кадрирование изображения, выделив центральную часть лица, и зафиксируйте в таблице получаемые в окне Command Window при выполнении программы максимальные значения коэффициентов корреляции зашумленного изображения и отфильтрованных разными фильтрами изображений. Затем, изменяя параметры фильтров, добейтесь получения наиболее высоких значений коэффициентов корреляции отфильтрованных сигналов с исходным.

2. Повторите выполнение п. 1 при шуме `salt & pepper` с параметрами $d=0.01$, 0.05 и 0.1 и при мультипликативном шуме `speckle` с нулевым средним значением и среднеквадратичным отклонением $v=0.02$, 0.04 и 0.06 .

3. При выполнении п. 2 Программы спроектируйте на внутреннем языке MATLAB программу цифровой обработки изображений, реализующую уменьшение шумов на изображениях, используя стандартные базовые программы:

- `average` – усредняющий фильтр;
- `medfilt` – медианный фильтр;
- `gaussian` – фильтр Гаусса;
- `ordfilt` – ранговый фильтр;
- `unsharp` – фильтр повышения резкости;
- `wiener2` – адаптивный фильтр Винера;

Программа должна позволять производить интерактивный выбор вида фильтра и параметров фильтра.

Содержание отчета

1. Задание к работе.
2. Таблицы с показателями корреляционного сравнения изображений по п. 1 и 2 Указаний по выполнению работы с комментариями.
3. Текст программы по п. 3 Указаний по выполнению работы и экранная форма.
4. Выводы.

Приложение. Текст базовой программы фильтрации шумов

```
%Программа производит зашумление изображения
%фильтрацию зашумленного изображения с помощью
%различных фильтров, определение коэффициента корреляции
%зашумленного и отфильтрованных изображений с исходным

img=imread ('your_image.tif');

% У каждого студента должен быть собственный файл с изображением

I=imcrop(img);
figure;
imshow (img);
title('Исходное изображение');
img1=imnoise(img,'gaussian',0,0.015);% 0.005
figure;
imshow (img1);
title('Зашумленное изображение');

%Медианный фильтр 'medfilt2'
hsize=[3 3 ];
F4= medfilt2(img1,hsize);
figure;
imshow(F4);
title('После медианного фильтра');
pause;

%Усредняющий фильтр 'Average'
hsize=[3 3 ];
h= fspecial('average',hsize);
F1=imfilter(img1,h,'replicate');
figure;
```

```

imshow (F1);
title('После усредняющего фильтра');
pause;

ncorr = normxcorr2(I(:,:,1),img2(:,:,1));
figure, surf(ncorr), shading flat;
title('Сравнение с изображением другого человека');
max_c_other_man = max(abs(ncorr(:)))
pause;

ncorr = normxcorr2(I(:,:,1),img1(:,:,1));
figure, surf(ncorr), shading flat;
title('После зашумления');
max_c_noised = max(abs(ncorr(:)))
pause;

ncorr = normxcorr2(I(:,:,1),F4(:,:,1));
figure, surf(ncorr), shading flat;
title('После медианного фильтра');
max_c_median = max(abs(ncorr(:)))
pause;

ncorr = normxcorr2(I(:,:,1),F1(:,:,1));
figure, surf(ncorr), shading flat;
title('После усредняющего фильтра');
max_c_average = max(abs(ncorr(:)))
pause;

%Фильтр Гаусса 'gaussian'
hsize=[9 9];
sigma=0.99;
h= fspecial('gaussian',hsize,sigma);
F2=imfilter(img1,h,'replicate');
figure;
imshow(F2);title('После фильтра Гаусса');
pause;

```

```

ncorr = normxcorr2(I(:,:,1),F2(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра Гаусса');
max_c_gaussian = max(abs(ncorr(:)))
pause;

```

```

%Фильтр Лапласа 'laplacian'
alpha=0.5;
h= fspecial('laplacian',alpha);
F3=imfilter(img1,h,'replicate');
figure;
imshow(F3);title('После фильтра Лапласа');
pause;

```

```

ncorr = normxcorr2(I(:,:,1),F3(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра Лапласа');
max_c_laplasian = max(abs(ncorr(:)))
pause;

```

```

%Фильтр повышения резкости 'unsharp'
alpha=0.2;
h= fspecial('unsharp',alpha);
F4=imfilter(img1,h);
figure;
imshow(F4);title('После фильтра unsharp');
pause;

```

```

ncorr = normxcorr2(I(:,:,1),F4(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра unsharp');
max_c_unsharp = max(abs(ncorr(:)))
pause;

```

```

%Ранговый фильтр

```

```

m=5;n=5;
J = imnoise(img,'gaussian',0,0.015);
imshow(J);
J6 = ordfilt2(J, 7, ones(4,4));
figure;
imshow(J6);title('После рангового фильтра');
pause;

ncorr = normxcorr2(I(:,:,1),J6(:,:,1));
figure, surf(ncorr), shading flat;
title('После рангового фильтра');
max_c_ordfilt2 = max(abs(ncorr(:)))
pause;

%Фильтр Винера
m=5;n=5;
[J,noise] = wiener2(img1,[m n]);
J6 = wiener2(J,[5 5]);
figure;
imshow(J6);title('После фильтра Винера');
pause;

ncorr = normxcorr2(I(:,:,1),J6(:,:,1));
figure, surf(ncorr), shading flat;
title('После фильтра Винера');
max_c_wiener = max(abs(ncorr(:)))
pause;

close all; %закрытие всех окон графического вывода
clear; %очистка Workspace

```

Литература

1. Сергиенко А.Б. Цифровая обработка сигналов. Учебник для вузов. 3-е изд.-СПб.: Питер, 2013.-768с.
2. Тутыгин В.С. Цифровая обработка коротких сигналов. Издательство Политехн. ун-та, 2012 – 164с.
3. А.И.Солонина, С.М.Арбузов. Цифровая обработка сигналов. Моделирование в MATLAB: Учебное пособие. – СПб.: БХВ-Петербург, 2008 – 816с. ISBN 978-5-9775-0259-7
4. Юкио Сато Без паники! Цифровая обработка сигналов: Пер. с яп.. – М.: Изд. дом Додэка – XXI, 2010 - 176с. ISBN 978-5-94120-251-5.
5. С.Л.Марпл-мл. Цифровой спектральный анализ и его приложения: Пер. с англ. – М.:Мир, 1990 – 584с. ISBN 5-03-001191-9
6. Давыдов А.В. Цифровая обработка сигналов. Конспект лекций.
<http://prodav.narod.ru/textbook/index.html>
7. Давыдов В.А., Давыдов А.В. Очистка сигналов от шумов методом эмпирической модовой декомпозиции в диалоговом режиме.
www.geoin.org/hht/app/hht6.doc
8. Тутыгин В.С. Способ очистки сигналов от шумов с использованием неполной декомпозиции Хуанга. В сб. докладов X Международной научно — практической конференции «Актуальные вопросы науки, технологии и производства», СПб., 2015.
9. Гонсалес Р., Вудс Р. Цифровая обработка изображений.// Техносфера. – 2006 г.-1072с.
10. Гонсалес Р., Вудс Р., Эддинс С., Цифровая обработка изображений в среде MATLAB.// Техносфера. – 2006 г.-616с.
11. В.Дьяконов, И.Абраменкова. MATLAB. Обработка сигналов и изображений. Специальный справочник. –СПб.: Питер, 2002.-608с.

ПРИЛОЖЕНИЕ

Форма отчета по контрольной работе



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ДГТУ)**

Наименование образовательной программы Информационные системы и технологии

Кафедра Кибербезопасность информационных систем

КОНТРОЛЬНАЯ РАБОТА

по дисциплине

«Цифровая обработка сигналов и изображений»

Обучающийся _____
подпись, дата

Обозначение 09.04.02.NN0000.000 Группа МИТ21

Направление подготовки 09.04.02 Информационные системы и технологии

Профиль Технологии искусственного интеллекта в
информационных системах

Руководитель _____
подпись, дата

г. Ростов-на-Дону
2026 год

Содержание

Задание	3
Процесс выполнения.....	4
Программный код.....	5

					09.04.02.NN0000.000	Лист
						2
Изм.	Лист	№ докум.	Подпись	Дата		

Задание

Разработать ...

					<i>09.04.02.NN0000.000</i>	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

Процесс выполнения

На первом этапе ...

					<i>09.04.02.NN0000.000</i>	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

Программный код

61

					<i>09.04.02.NN0000.000</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		